M.A.S.S.

(Mobile Aerial Surveillance System)

Senior Design

GROUP 6

College of Engineering
and
Computer Science

Henry Chan
Derrick Shrock
Eric Hernandez
Sanjay Yerra

Table of Contents

SECTION 1: EXECUTIVE SUMMARY

In today's world, there are many things that go wrong in life. The outcome on arrays of events that occur all the way from terrorism to as simple as traffic accidents can dramatically change if cameras were implemented more in our society. Even though there have been more applications of their usage, the idea of them being mobile would be more idyllic. Introducing the Mobile Aerial Surveillance System (MASS), a robotic blimp camera system designed to be an all-in-one observation system. The research behind this project will have the prospective to make important contributions to the advancement of self-directed robotics as well as being enjoyable to us engineering students.

Refined machines are able to do crazy tasks that most people cannot even begin to think about much less have people try to complete the inconceivable tasks, which would take extended periods of time to accomplish, and in the end may even led to the overall failure of the tasks in the first place. With that being said, you can pretty much determine that it would be nearly impossible to complete many of the mundane tasks we have today if it weren't for the advancements in technology. For example, if we still did not have cellular phones, communication would be limited to land lines wiping out nearly almost all the data we use in today's world with phones (internet, texting, hot spots, networking, remote controlling, etc.). This productivity in the technological industry has led to a dependency on electronics us humans have.

Since the development of aviation, there have also been many advances. We have used aviation in war, civilian transportation, the transportation of goods such as food and items we buy and use, sporting events, all the way down the list to even being an extracurricular activity. Now that we live in a world where everything is controlled by technology and electronics, why not combine several features together in order to have one design to help in a bunch of different ways.

The main objective of this project is to design a mobile device that will be able to observe and capture events as they happen while being connected to an aviation envelope. It will have several main functions, which are as follows: video surveillance, camera stabilization system, power directed propellers (on a rotating axis), GPS, user control by remote control, user control by inputting coordinates into system, user control by android phone, and video recognition. This system has the intentions to be just a fancy surveillance system; however, it will also act as a type of revenue for some people by advertising their company's logo on the side, or even sending data to a centralized base such as weather readings, body counts at special events, or simply just recording your favorite football team to watch on television.

SECTION 2: PROJECT DESCRIPTION

The main goal of this project is to have a personal surveillance system that can work anywhere and can be used by anyone for his or her own use. Nonetheless, in today's frontline of defense the ability of having visual and communication support from the skies. Unfortunately, this project will be limited due to funds, but can be taken to a level of surveillance that is undetectable by the best technology out there.

Section 2.1 Project Motivation and Goals

Many contributions go into the motivations and goals of this project, ranging from particular applications and personal use, to realistic, real world functions. Although there are a plethora of uses, they pretty much all coincide together. For example, having the MASS at a sports event, not only can you have it observing the entire place watching for any disturbances and capturing it all, the blimp can either have advertisement displayed on the side of it or have itself be the advertisement, meaning the shape of the airship can be something like an animal or object (being a sports event it would most likely be the team's mascot).

This project has qualities that are not only good for interest of having security, but from engineering stand point it is also very green because this system requires no fuel or natural resources. The only resource it needs is the electricity to charge the batteries required to power the propellers and other fundamental parts it requires.

From a personal standpoint, we as engineering students have been studying the fundamentals for a while now. This project will require us to not only use the rudiments we have been learning through our education, but will also require us to ascertain skills that we have to research and further investigate in order to complete the main task at hand and not to mention will be a great stepping stone for the key into being successful engineers.

Section 2.2 Objectives

The objective of the MASS is to have an autonomous blimp that can not only take and record video and send it to a central computer station, but also give data such as weather specifications, as well as be controlled by an Android phone device. The reasoning behind this is to give someone the sense of having their very own personalized surveillance system that they control with their cell phone. Not only that, but the system will be a fun little RC hobby to have for every hobbyist out there.

***Section 2.3** Project Requirements and Specifications*

In this section we will discuss the requirements for this project. By not achieving these specifications, the MASS will not be presentation ready. The following requirements will be our guidelines:

- MASS will entail each member to at least 200-300 dollars
- Approximately 70-80 cubic feet of helium
- Hold at least 4-5.5 pounds of equipment
- Rechargeable Batteries
- Wireless Control
- GPS enabled
- Video camera system

Section 3: Research Related to Project

Most of the background on blimps and airships is mainly full of history from the military and inventions and all the way to civilian passenger airships. These vessels are typically circumnavigated by using either petrol powered or electronically powered, engine-driven, propellers. The three types of airships are as follows: rigid, semi-rigid, and non-rigid. The definitions are really self-explanatory. An airship that is rigid is precisely that, it has an internal frame, generally made from metal, to uphold the shape of the ship. A semi-rigid airship also has a frame except the frame is only partially there and instead has firm beams holding its shape. Lastly, the non-rigid airships are just that, no frame. Instead of using a frame as the shape holder for the envelope, the natural lift and figure of the envelope is produced from the helium induced into the balloon. Since we are limited with not only cost, but size as well, the motivation behind our project is going to be directed toward the non-rigid model.

***Section 3.1** History Behind Airships*

Like all other inventions, they have to come from somewhere. France is where the hot air balloon was first created and also envisioned as a use for military. Meusnier, a French officer back in the 1780's, had designed the first airship that had a lot of the same features as the airships of today's world. It had a drawn-out, ellipse-shaped envelope, propellers, and a rudder. Even though he designed this in documentation, it was never built. However, in 1852, an engineer (naturally) named Henri Giffard, built the first real-world airship. It was filled with hydrogen gas and was able to accomplish lift-off, even though it wasn't completely controllable. Two men, Renard and Krebs, whom, by default, were also Frenchmen, built the first actual controllable and effectively piloted airship in 1884.

One of the first airships for a military type use came along when David Shwartz, a German, built a rigid bodied airship. With his design being the fist, a great success story of rigid bodied blimps would be that of the Zeppelin. The Zeppelin was constructed by Count Zeppelin, which led to the initial military advantage for Germany when they built twenty other ships just like this one in WWI. Witnessing the success the Germans had with the airships, the British decided to develop some as well except these were non-rigid aircrafts.

Without a doubt, most of the early airships, especially those of the 1800's, where used primarily for military use. Despite this fact, they did eventually start being ships for civilians. This happened around the early 1900's. Some of the first countries to implement these as human carrying airships are Britain, Germany, and the United States. The United States, however, was a little different from the other to and used a design like one of which we will be using; a helium filled non-rigid airship (blimp) to attain the proper lift.

Section 3.1.2 Early Blimps

The start of air surveillance began shortly after the 1930's as a use for the military. The reasoning behind this is because of the tragedies happening with civilian carriers from the 1920's-30's, so since they still wanted to use them, people came up with making surveillance systems. Thus, success came out of these tragedies because airborne surveillance systems were a huge hit with the blimps. This is where the heart of the MASS design comes into play. These early-days surveillance systems watched over military missions, was also used as warning systems and radar systems, as well as observing scientific experiments.

Section 3.1.3 Modern Uses of the Blimp

The main reasoning behind the development of the MASS is the fact that in today's world the blimp plays a huge roll in not only surveillance, but also in the advertising industry. You see blimps at the beach when you're relaxing with words sprayed on the side and banners, also at sporting events usually capturing the game as it goes on, and even designs of movie ideas flying around when a new movie comes out. They are excellent for advertisement based on the fact that the pretty much can hover in one spot for long periods of time. They are also usually large in size, thus being able to be noticed with ease. They also, but not lastly, are no very noisy, keeping the disturbance level to a minimum.

Electronics play a huge role in this industry as well with the airship. Apart from the actual controls system itself some ways of advertising have come into play by using LED lights and LED screens at night implemented on the blimp. So not only will you see the advertised logo on displayed on the sides, but now you have a programmable unit to change anything up.

Section 3.1.4 Materials Usually Used

The main part of the blimp is called the envelope. This envelope is usually made from a list of materials including, but not limited to: Dacron, Mylar, polyester, different types of Nylon, etc. The dependency of the material changes depending on what applications you need to use the system for. Even though the MASS will not be using a bladder, which a balloon like material that fits into the envelope to make sure it holds shape and can withstand the load of the pressure.

Section 3.1.5 Design

Many of the blimps today are made from pretty much the same basic design. They have the basic shape of an ellipsis and involve an envelope. For our design this is the only thing we will be using. Just an envelope designed to hold a small amount of helium, at least compared to the civilian carriers. The design is basically the same, but includes an inner layer, a bladder, and an envelope. The reason why we do not need the inner layer or bladder is because our system will not have enough pressure to really need the other two segments. For the larger designs, the bladder fills up with air and holds its form by the envelope, which also acts as a protection layer.

The nose cone pretty much serves as two different functions. Not only does it offer a point of attachment (like anchoring to the ground or for the crew to guide when it is on the ground before take-off and after landing), but it also provides better protection of the entire design being as the ends of the blimp system are the weakest, also having the extra protection in case of front end abrasions.

The main powerhouse of the entire system is located in the rear of the gondola. The gondola also contains the controls for the structure. These controls include the communications, electrical systems (prop pitch, throttle controls, temperature controls, helium and regulation of the balloon air pressure, etc.), and fuel.

Section 3.2 Relevant Technologies

Aviation has been around since the late 1700's, which started with the hot air balloon. Since then there has been many different designs and ideas on airships used for all kinds of applications. Today, there are many different ways to obtain your very own airship whether it's for a hobby or, for the subject at hand, security purposes. A few of similar project and products are www.eblimp.com (eblimp), www.surveyor.com (surveyor), Lockheed Martin's HALE-D, among many others.

www.eblimp.com

Of the three examples listed, two are available to purchase for personal use. Although each of these, for the first example, we are going to discuss www.eblimp.com's blimps. They design and make indoor and/or outdoor blimps for your specific use. This service is designed based off of velocity, wind conditions, run time, altitude, cargo requirements, cost, maneuverability/aerobatic performance, and size requirements or constraints. Their product description is as follows: "eBlimps are manufactured from Nylon, Urethane, or PVC, they can be printed, painted, stickered, or equipped with banners. The Motor system is manufactured using ultra lightweight carbon fiber composites, and aluminum motor mounting hardware." (http://eblimp.com/eblimp/Outdoor_Blimps.html)
These complete systems include, but are not limited to:

- Blimp envelope (eight to thirty feet)
- Vinyl graphics(advertising purposes)
- Internal lights for a glow effect at night time
- Drop device to drop items from such as coupons and prizes, etc.
- Batteries and chargers
- Video system

This system is very close to the one we will be developing.

www.surveyor.com

Surveyor Corporation is a company which has a robotics forum and has many different projects listed on their site that they have done. Among these projects and info logs is the YARB which is a robotic blimp system they built. This project is also just like the MASS. With a few minor differences and the fact that we use different parts and camera system, this is the most relevant technology to our project out there.

The Lockheed Martin High Altitude Airship (HAA™):

Lockheed Martin is one of the top leaders in almost every engineering realm out there. Even though they have had a few mishaps which their own versions of airships they have created very successful projects as far as surveillance goes. The following is one of their projects which even though was successful at doing its job, had to make a crash landing due to a malfunction. "The Lockheed Martin High Altitude Airship (HAA™) – and its sub-scale demonstrator, the High Altitude Long Endurance-Demonstrator (HALE-D) – is an un-tethered, unmanned lighter-than-air vehicle that will operate above the jet stream in a geostationary position to deliver persistent station keeping as a surveillance platform, telecommunications relay, or a weather observer."
(http://www.lockheedmartin.com/us/products/lighter-than-air-vehicles/haa.html)

The MASS will be something similar to this, minus the fact that it will not crash. Obviously, we will not have the same advantage points as the much larger and much more expensive Lockheed Martin system, but will implement some of the same main functions. Some of the similar feature of the HAA are it has surveillance capabilities with system camera, it records data and sends it to a main communication server, also tracks the location of where it is and the communications it has between the central hub.

Even though our system will not be communication with satellites, it's still on the same page as far as communication goes. With the invention of these types of these advanced airships, we can improve communications on the battlefield; collect solar energy, and key in-flight operations to help succeed in various military applications

Albeit Lockheed Martin created a magnificent airship, the fact that it crashed shows that they may not have tested it properly. They could have prevented this by running several tests before actual lift off occurred.

Section 3.3 Microcontrollers Comparison and Research

A microcontroller is a miniature computer on a single integrated circuit that contains memory, a processor core, and programmable input/output pins. Memory is normally stored as NOR Flash (which is when there is a cell that one part is connected straight to the ground and the other a bit line) and also contains a small amount of RAM. Almost anything that is able to interact with its user will have an embedded system inside of it which would be controlled by a microcontroller. For example a number of everyday items have a microcontroller in them ranging from microwaves, remote controls, dishwashers amongst many other everyday use items. Microcontrollers are typically given one specific tasks and will do this and only this tasks which they are specialized for, for their lifetime. There are several important factors to consider when looking for a microcontroller and considering all the things that will be connected to ours we had to get something that can handle multiple signals at once.

The signs of a good microcontroller depend on the size of its memory, the number of input/output pins it has, the physical packaging it uses, peripherals(like what can be read from the ports) and the overall architecture of the microcontroller. When dealing with the architecture we also need to choose certain features associated with the controller like if it is a complex instruction set computer or if it is a reduced instruction set computer. Microcontrollers come with a library that assists in helping the user program the microcontroller to the other parts it is connected too. They also tend to have integrated development environments that allow users to create computer programs for software development.

For our project, making this decision was the most difficult because a lot of assuming had to be made about how much memory we needed, how many input and output pins were needed among other things. For this reason it was very important for us to become knowledgeable about multiple different types of microcontrollers and all of their functions in case we needed to switch to a new one. For this reason we checked out numerous microcontrollers that we thought would suffice for our project then narrowed it down to only four. A comparison table will be displayed next which will help discover the strengths and weaknesses of every microcontroller and help us decide which one will be perfect for our project. When searching we wanted to narrow down the choices to have at least 1kB of RAM memory, at least 36 input/output pins, and be able to read peripherals which was the most important aspect considering that we will have an accelerometer, gyroscope, GPS module, and servo motors attached to our microcontroller. We also wanted to have an ADC on chip just in case we end up using an analog out for the motor controls or if any of other devices use analog instead of digital. We want to cover as many of possible bases right now so in 3 months when everything is being put together we do not have to buy extra parts and worry about shipping time. Shipping time is also one other aspect to consider seeing as most take at least 6 weeks to ship even upon saying that it is ready to go. The factory lead time is a very misleading disclaimer. On **[Table 3.3.1]**, it highlights the microcontroller speeds as well as the memory sizes of the microcontrollers we are considering.

Microcontroller Part Number	**ATMEGA328P-PU**	**MC9S08AC128MFGE**	**LC87F5LP6AU-CH-E**	**ATmega128-16PU**
Manufacturer	Atmel	Freescale Semiconductors	On Semiconductors	Atmel
Core	AVR	S08	On chip	AVR
Data Bus Width	8 bit	8 bit	8 bit	8 bit
Maximum Clock Frequency	20 MHz	40 MHz	12 MHz	16 MHz
Program Memory Size	32 KB	128 KB	256 KB	128 KB
Ram Size	1 KB	8 KB	8 KB	4 KB
Maximum Operating Temperature	85 C	125 C	70 C	85 C

***Table 3.3.1**: Details of Different Microcontrollers in Consideration (Part 1)*

On **[Table 3.3.2]** below, the highlights of the table are the interface types available on each microcontroller. When we pick modules such as the accelerometer and GPS, we have to make sure that the digital pin for it to communication is available on the microcontroller or else we cannot use it. The number of timers are important for keeping track of any counting we have to do, such things that the timers are important for are generating the PWM signal for the motor controls. So we have to make sure we have enough timers for any device that needs them, but we can also share the generated clock of one across many devices if we time it correctly.

Microcontroller Part Number	**ATMEGA328P-PU**	**MC9S08AC128MFGE**	**LC87F5LP6AU-CH-E**	**ATmega128-16PU**
Interface Type	2 Wire, SPI, USART	I2C, SCI, SPI	UART	2 Wire, SPI, USART
Mounting Style	Through Hole	SMD/SMT	SMD/SMT	SMD/SMT
ADC	Yes	Yes	Yes	Yes
A/D Channels	6	16	15	15
A/D Bit Size	10 bit	10 bit	8 bit	8 bit
Data ROM Size	1 KB	1 KB	4 KB	4 KB
Number of Programmable I/Os	23	70	64	53
Number of Timers	3	3	8	4
Minimum Supply Voltage	1.8 V	2.7 V	2.5 V	4.5 V
Maximum Supply Voltage	5.5 V	5.5 V	5.5 V	5.5 V

***Table 3.3.2**: Details of Different Microcontrollers in Consideration (Part 2)*

When searching for microcontrollers there were about 15,761 different models that came up. We liked the idea of using an Atmel processor because they use the Harvard architecture and are commonly used. This project will require a microprocessor; fortunately there is an entire market of microprocessors that will fit our needs and specifications. The microprocessor that our group requires will need to have three specifications, namely, a serial transmission translator such as SPI, I^2, or USART, 16 or more I/O pins, and will need to be able to hold all of our code. In terms of specifications all of the microcontrollers listed were very similar. All have the same 8 bit width for the bus. The maximum clock frequency is one of the few things that every microcontroller has different values. Obviously the higher clock frequency means the faster things will be done. The Freescale Semiconductor has the fastest microcontroller at 40 MHz which is twice as fast as any other microcontroller that we are looking at, however 16 MHz should be sufficient for the things we need it for. The memory size ranges for our microcontrollers as well with us having to estimate how much memory we will actually need. The controllers need to have either SPI or I2C interfaces, including have UART as well which will connect to our peripherals. The number of programmable pins became a major player when we were deciding between our two favorite microcontrollers. All microcontrollers have the same maximum supply voltages which will end up playing no role in our decisions.

At the end of the deal our familiarities with the Atmel Processor lead us to pick between the two different ATMEGA processors. So now we will go more in-depth between the ATMEGA 128 and the ATMEGA 328 processor.

Section 3.3.1 Microcontroller Information

Through looking at the microcontroller comparison, we have chosen the AVR ATmega128 as the processor that we will use in our project. It has all the digital I/O pins that we need for our project as well as has a large community support for help in coding that we might need in interfacing different modules.

AVR ATMEGA 128

The ATmega microprocessors are famous for being in the Arduino Microcontroller boards. These lines of microprocessors are made by AVR and have Harvard RISC architecture. Because this microprocessor is so famous, many programs have been written and placed on the internet for free use to the general public. The AVR atmega128 has 32 general registers that all connect to the Arithmetic Logic Unit. This allows two independent registers to be accessed in one single instruction on a single clock cycle. This allows for super-fast computing, up to 1 MIPS per MHz! This allows the programmers to optimize power consumption and processing speed.

Power

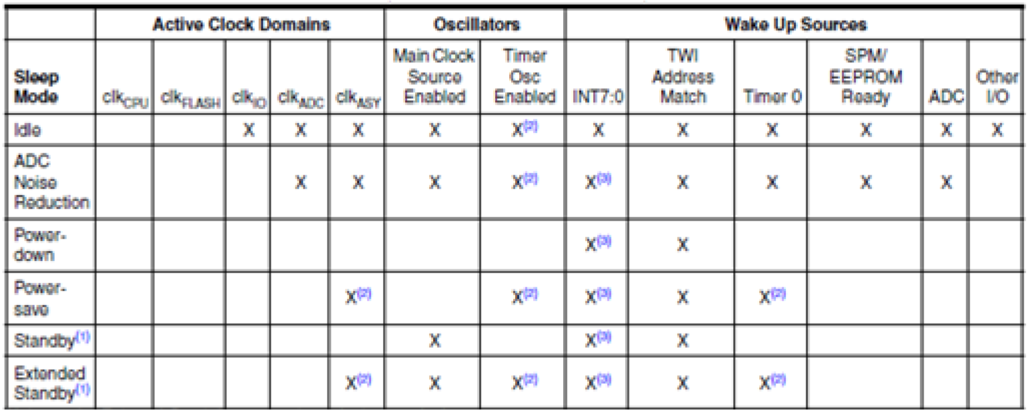
The AVR ATmega128 has a decent to high operating voltage of 4.5 - 5.5V. However there are many features that are built into the system that allow for making it a very lucrative option for being our microcontroller. The AVR has many various sleep modes that let us reduce the power consumption from the device. This power management comes from the MCU control Register. It can allow for a sleep mode if Bit 5 is enabled. Note when programming the sleep enable mode it is recommended to write the bit to one before its execution. An even better feature is in the MCU register bits 4-2. These bits when enabled in a certain way make certain parts of the microcontroller sleep. this will be very useful for the times when the Blimp does not need to do anything but wait to receive a signal from the ground station.in the **Figure 3.3.3**, below shows the select sleep modes that are available and which pins would be needed to be enabled.

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC Noise Reduction
0	1	0	Power-down
0	1	1	Power-save
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby(1)
1	1	1	Extended Standby(1)

Figure 3.3.3*: Modes on Microcontroller Pin to save Power (Sleep Mode). Reprinted with Permission by Atmel*

In idle mode, the CPU is stopped but the SPI, USART, analog comparator, analog to digital Converter, two-wire serial interface, timer/counters, watchdog, and the interrupt system continue operating. This will likely be the mode at which our blimp will be operating majority of the time. The noise reduction mode is when everything except the analog to digital converter, external interrupts, the two-wire serial interface and timer zero will be turned off. This does not seem like it would be useful for use because the serial communications will likely always need to be on for the control of the blimp. The other two modes, power save and power down, might be useful when the blimp is at a low altitude and docked at the power station. Even then it might be better to just turn the power off to the device. We will decide this when the project is actually implemented.

Figure 3.3.4 below, shows the various pins that will wake up the MCU out of the various sleep modes.



	Active Clock Domains					Oscillators		Wake Up Sources					
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Osc Enabled	INT7:0	TWI Address Match	Timer 0	SPM/ EEPROM Ready	ADC	Other I/O
Idle			X	X	X	X	X[2]	X	X	X	X	X	X
ADC Noise Reduction				X	X	X	X[2]	X[3]	X	X	X	X	
Power-down								X[3]	X				
Power-save					X[2]		X[2]	X[3]	X	X[2]			
Standby[1]						X		X[3]	X				
Extended Standby[1]					X[2]	X	X[2]	X[3]	X	X[2]			

***Figure 3.3.4**: Modes on Microcontroller Pin to save Power (Wake Up Mode). Reprinted with Permission by Atmel*

Input/ Output Lines

This microprocessor has 53 programmable input and output lines. Each of these ports has a true Read-Modify-Write functionality when only used as general digital I/O ports. So the direction of a pin with SBI and CBI instructions will not be affected by a change to a single pin. There are three memory address locations for each port. One for the Data Register- PORTx, Data direction Register-DDRx, and the Port Input Pins-PINx. Note that the Data register and the data direction register are read and write, but the port input pins are read only. a great feature about using the I/O ports as Digital ports is that the pins are automatically multiplexed with alternate functions. The general Digital I/O bus is as shown in **Figure 3.3.5**.

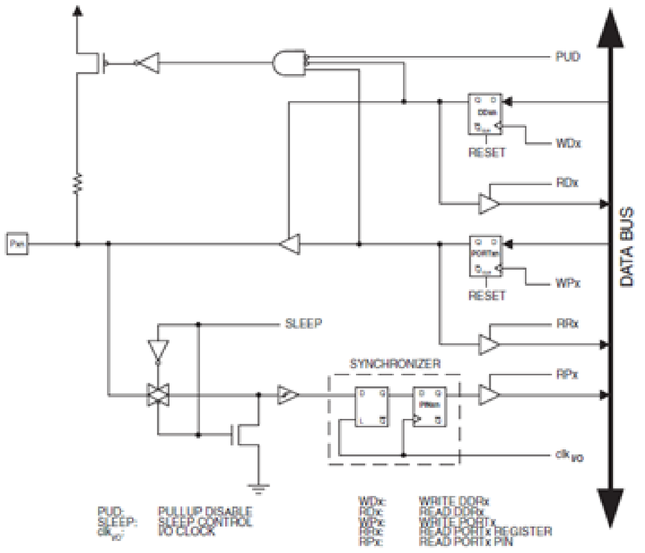


***Figure 3.3.5**: General Digital I/O Bus. Reprinted with Permission by Atmel*

Pins and Configuration

The ATmega has 53 input pins. Let's talk about them. Each port takes in or gives out 8 bits of data. The microcontroller has 32 general purpose registers for operations. Each are connected to the ALU, so all many operations can be carried out at a single time. The Pin configuration is as shown in **Figure 3.3.6.**

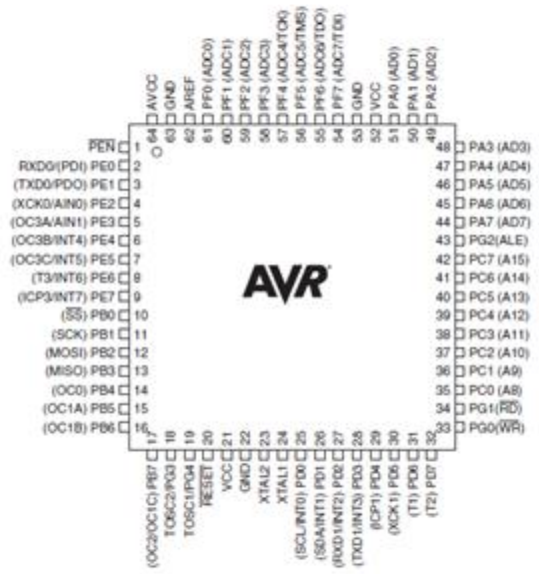
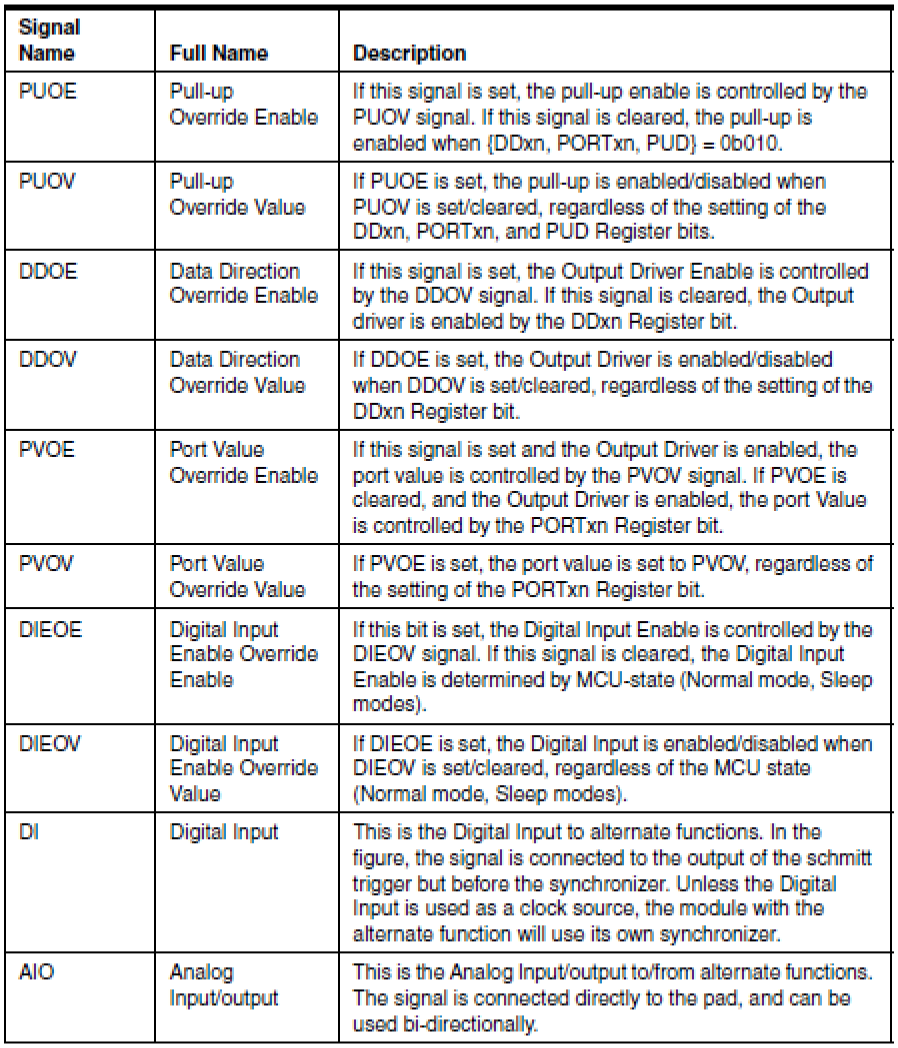


***Figure 3.3.6**: Pin Layout of Atmel AVR ATmega128.*
Reprinted with Permission by Atmel

This microcontroller has been optimized for the AVR Enhanced RISC instruction set. So from an assembly programmer's standpoint, it looks very similar to MIPS. The instruction set might not be that important, seeing how we'll most likely program in C. However, the possibility of programming in assembly is open because assembly code takes up less space in the memory of the microcontroller and tends to be faster as well. Each of the ports have a specific purpose, however if they are used as general purpose input and output pins, the microprocessor will allow the use of the ports alternative functions as well. The microcontroller has a VCC pin for digital voltage supply. The ground pin is for excess voltage and current. There is an AVCC pin for the F port and for the Analog to digital converter. The pin has to be connected to the VCC externally, even if the ADC pin isn't used. However, if the ADC pin is used, VCC and ADC should be connected through a low-pass filter. The AREF pin is used as an analog reference signal for the analog to digital converter. The XTAL1 is a pin that is the internal clock operating circuit input as well as the inverting Oscillator amplifier circuit. XTAL2 is the output from the inverting Oscillator amplifier circuit. The Reset pin is short for reset input. A low level, as in "off" for a prolonged period of time, generally the minimum pulse width length, will call a reset, even if the clock is not running. Note: shorter pulses will not guarantee a reset. The PEN pin is a pin that is programming enable for the SPI serial programming mode and is internally pulled high. Meaning that if the pin is low during the Power-on-Reset, the device will enter the SPI programming mode which will be used to connect to our transceiver and ground station. Note: that the PEN pin has no function during normal operation. The ATmega128 has 8 ports, Port A-G. Each of these ports is an 8-bit bidirectional I/O port with internal pull-up resistors (selected for each bit). All of the ports' output buffers have symmetrical drive characteristics with a high sink and source capability. Every port's pins that, if set as inputs that are externally low will source current if the pull-up resistors are activated. Note all the port pins will be tri-stated when a reset condition becomes active, regardless if the clock is running or not. Now specifically each port has alternative functions. All of these ports have different alternative functions but in essence the functions all the same, however some pins will have the function capability turned off. The **Figure 3.3.7** below shows these pin alternative functions.

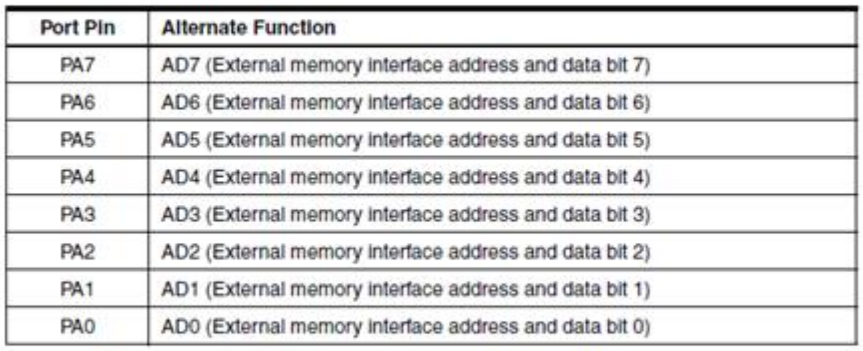


Signal Name	Full Name	Description
PUOE	Pull-up Override Enable	If this signal is set, the pull-up enable is controlled by the PUOV signal. If this signal is cleared, the pull-up is enabled when {DDxn, PORTxn, PUD} = 0b010.
PUOV	Pull-up Override Value	If PUOE is set, the pull-up is enabled/disabled when PUOV is set/cleared, regardless of the setting of the DDxn, PORTxn, and PUD Register bits.
DDOE	Data Direction Override Enable	If this signal is set, the Output Driver Enable is controlled by the DDOV signal. If this signal is cleared, the Output driver is enabled by the DDxn Register bit.
DDOV	Data Direction Override Value	If DDOE is set, the Output Driver is enabled/disabled when DDOV is set/cleared, regardless of the setting of the DDxn Register bit.
PVOE	Port Value Override Enable	If this signal is set and the Output Driver is enabled, the port value is controlled by the PVOV signal. If PVOE is cleared, and the Output Driver is enabled, the port Value is controlled by the PORTxn Register bit.
PVOV	Port Value Override Value	If PVOE is set, the port value is set to PVOV, regardless of the setting of the PORTxn Register bit.
DIEOE	Digital Input Enable Override Enable	If this bit is set, the Digital Input Enable is controlled by the DIEOV signal. If this signal is cleared, the Digital Input Enable is determined by MCU-state (Normal mode, Sleep modes).
DIEOV	Digital Input Enable Override Value	If DIEOE is set, the Digital Input is enabled/disabled when DIEOV is set/cleared, regardless of the MCU state (Normal mode, Sleep modes).
DI	Digital Input	This is the Digital Input to alternate functions. In the figure, the signal is connected to the output of the schmitt trigger but before the synchronizer. Unless the Digital Input is used as a clock source, the module with the alternate function will use its own synchronizer.
AIO	Analog Input/output	This is the Analog Input/output to/from alternate functions. The signal is connected directly to the pad, and can be used bi-directionally.

***Figure 3.3.7**: Generic Description of Overriding Signals for Alternative Functions*
Reprinted with Permission by Atmel

Port A

The alternative functions for Port A are that it can function as the address low byte and data lines for the External Memory Interface. If the groups need to have an external memory device, this is the port we would connect it to. Each pin corresponds to a data bit and interface address. Port A goes from PA0-PA7. Pins are showed as follows in **Figure 3.3.8.**



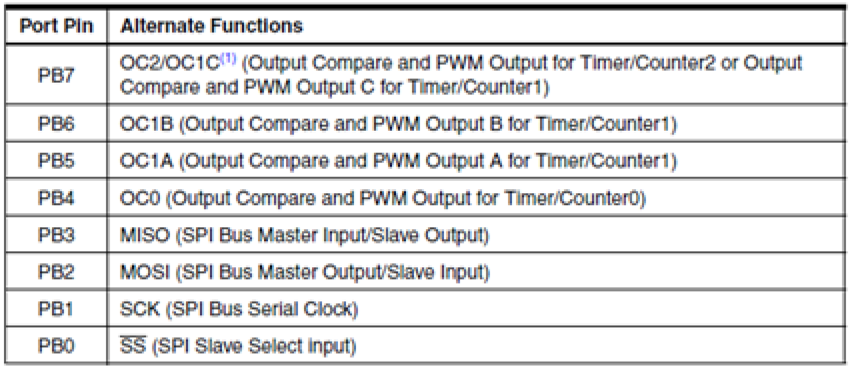
Port Pin	Alternate Function
PA7	AD7 (External memory interface address and data bit 7)
PA6	AD6 (External memory interface address and data bit 6)
PA5	AD5 (External memory interface address and data bit 5)
PA4	AD4 (External memory interface address and data bit 4)
PA3	AD3 (External memory interface address and data bit 3)
PA2	AD2 (External memory interface address and data bit 2)
PA1	AD1 (External memory interface address and data bit 1)
PA0	AD0 (External memory interface address and data bit 0)

Figure 3.3.8*:* Functions of the Port A Pins
Reprinted with Permission by Atmel

Port B

Port B has an alternative functionality that allows for serial input operations. Pin B0 is the SS, the selective slave input. So if the SPI is enabled as a slave, this will be the pin that will be configured as an input, regardless of pin DDB0. As a slave, the SPI is low active. However, when the SPI is enabled as master, the data direction is actually controlled by the DDB0 pin. Note that when the pin is configured to be an input, the pull-up can be controlled by PORTB0 bit. PORT B, bit 1, is the SCK pin. It is used as the master clock output or slave clock input for the SPI channel. Note, when the SPI is enabled as a slave, the pin is configured as an input, regardless of the setting of the DDB2 pin. When the pin is configured to be an input, the pull up can still be controlled by the PORTB1 bit. PORTB2 bit is the MOSI, also known as the SPI Master output/ slave Input channel. This pin works when the pin is configured as an input regardless of the setting of pin DDB2. When the pin is an input fully, the pull-up can be configured and controlled by the PORTB2 bit. PORTB3 bit is the MISO, also known as the SPI Master Input/ Slave Output channel. This pin works when the pin is configured as an input regardless of the setting of pin DDB3. When the pin is an input fully, the pull-up can be configured and controlled by the PORTB3 bit. Bits 7-4 in PORTB are all output compare match output pins. The pins can serve as external outputs for the Timer or Counter compare bits. The pins are also the output for pulse width modulation mode timer function.

Figure 3.3.9 below shows the port B functions of the Atmel ATmega128. We are most interested in the PWM output signals to control the motors and servers and the four digital SPI pins to connect to either a sensor module or the transmitter and receiver combo.

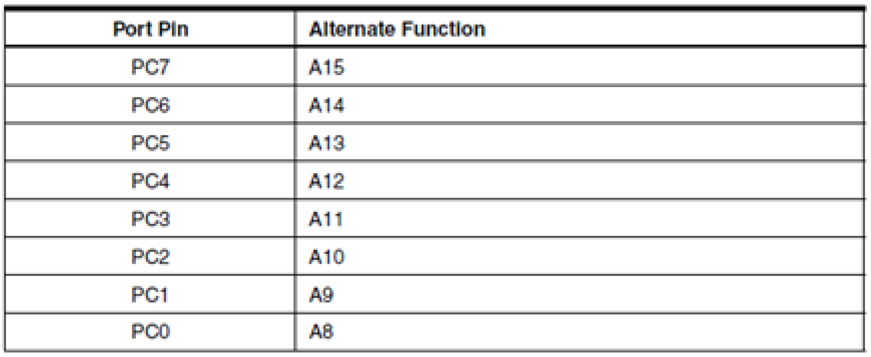


Port Pin	Alternate Functions
PB7	OC2/OC1C[1] (Output Compare and PWM Output for Timer/Counter2 or Output Compare and PWM Output C for Timer/Counter1)
PB6	OC1B (Output Compare and PWM Output B for Timer/Counter1)
PB5	OC1A (Output Compare and PWM Output A for Timer/Counter1)
PB4	OC0 (Output Compare and PWM Output for Timer/Counter0)
PB3	MISO (SPI Bus Master Input/Slave Output)
PB2	MOSI (SPI Bus Master Output/Slave Input)
PB1	SCK (SPI Bus Serial Clock)
PB0	$\overline{SS}$ (SPI Slave Select input)

Figure 3.3.9*:* Functions of the Port B Pins
Reprinted with Permission by Atmel

Port C

Port C functions alternatively as output only. Initially, the ATmega 128 is shipped in compatibility mode, meaning if the part are not programmed before they are connected on the PCB, Port C will be an output on the first boot up. This port is used for external memory similar to port A. however this port is for address high bytes in the external memory interface. The table in **Figure 3.3.10** shows the pins and the alternative functions for each.



Port Pin	Alternate Function
PC7	A15
PC6	A14
PC5	A13
PC4	A12
PC3	A11
PC2	A10
PC1	A9
PC0	A8

Figure 3.3.10*:* Functions of the Port C Pins
Reprinted with Permission by Atmel

Port D

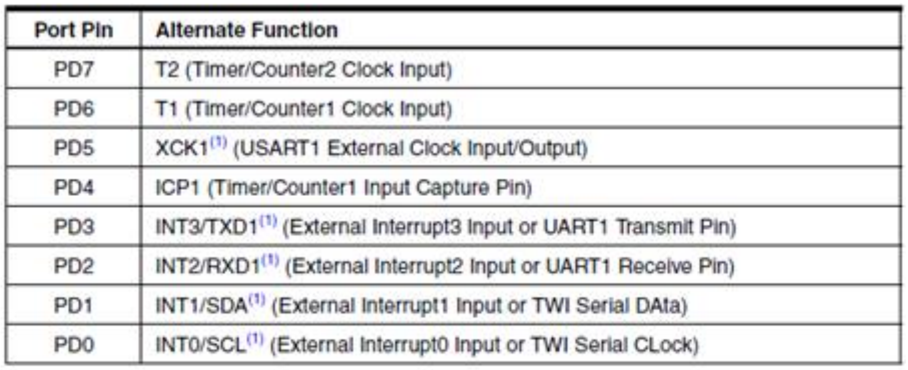
The pins in port D are used for various different purposes. In general, they are used for timing and for causing interrupts. PORT D bit 7, is called T2. T2 is the Timer/Counter2 counter source. PORT D bit 6, is called T1. T1 is the Timer/Counter1 counter source. PORT D bit 5 is called the XCK1, USART1 external clock. Note that the DDR4, the data direction register controls whether the clock is an input (DDD4 cleared) or output (DDD4 set). Only if the USART1 is operating on synchronous mode, will the XCK1 pin be active.

PORT D bit 4, is called the ICP1, which stands for the Input Capture Pin 1. The PD4 pin acts as the input pin for the time/ counter1 pin also known as PD6.

PORT D bit 3 is the INT3/TXD1. The INT3 is an external interrupt source 3, which can serve as an external interrupt to the MCU. TXD1, stands for transmit Data, specifically data output for the USART1. If the USART1 is enabled, this pin will be an output pin regardless of DDD3.

PORT D, bit 2 is called the INT2/ RXD1. The INT2 stands for external Interrupt source 2. It serves as an External Interrupt to the MCU. RXD1, stands for Receive Data, specifically from the Data input pin for the USART1. The USART receiver will be enabled as an input regardless of the value of DDD2. Note, even if the USART forces the pin to be an input, the pull-up can be controlled by the

The INT1 stands for external Interrupt source 1. It serves as an External Interrupt to the MCU. SDA is two-wire Serial Interface Data. It works when the TWEN bit in the TWCR is set high (one) to enable the Two-wire Serial Interface. Pin D1 then disconnects from the port and becomes an I/O (input or output) pin for the two-wire serial interface. If this enabled, the built-in spike filter will filter out any spike that are shorter than 50 ns in the input signal. PORT D0 aka INT0/SCL, the INT0 stands for external Interrupt source 0. It serves as an External Interrupt to the MCU. SCL is two-wire Serial Interface Clock. It works when the TWEN bit in the TWCR is set high (one) to enable the Two-wire Serial Interface. Pin D1 then disconnects from the port and becomes an I/O (input or output) pin for the two-wire serial interface. If this enabled, the built-in spike filter will filter out any spike that are shorter than 50 ns in the input signal. The figure below, **Figure 3.3.10** shows the functions of PORT D, in general terms. It can be assumed that this port will be connected to our serial transceiver and to our GPS module units.



Port Pin	Alternate Function
PD7	T2 (Timer/Counter2 Clock Input)
PD6	T1 (Timer/Counter1 Clock Input)
PD5	XCK1[(1)] (USART1 External Clock Input/Output)
PD4	ICP1 (Timer/Counter1 Input Capture Pin)
PD3	INT3/TXD1[(1)] (External Interrupt3 Input or UART1 Transmit Pin)
PD2	INT2/RXD1[(1)] (External Interrupt2 Input or UART1 Receive Pin)
PD1	INT1/SDA[(1)] (External Interrupt1 Input or TWI Serial DAta)
PD0	INT0/SCL[(1)] (External Interrupt0 Input or TWI Serial CLock)

Figure 3.3.10: Functions of the Port D Pins
Reprinted with Permission by Atmel

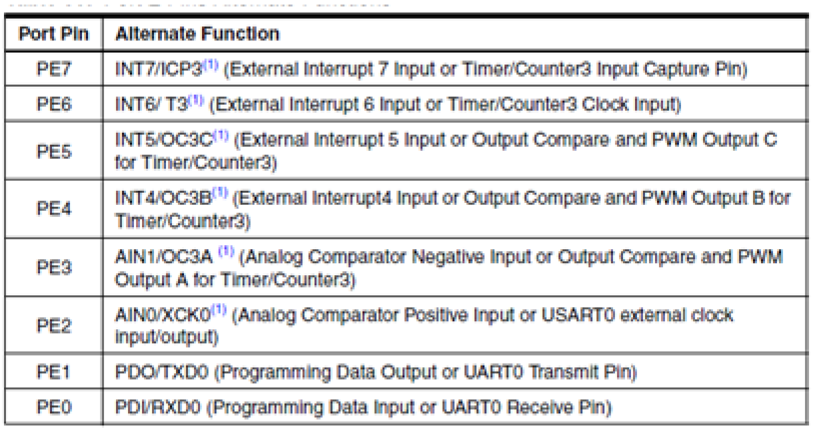
Port E

Port E has alternative functions that make it like the ADC. It is used for analog signals. PORT E, bit 7 has the IN7/ICP3 functions built in for the pin. INT7, external interrupt source means that it can serve as an external interrupt source. ICP3, input capture pin 3 means that the pin can act as an input for the timer/ counter 3 in pin PORT 3 bit 6 if needed. PORT E, bit 6 has the IN6/T3 functions built in for the pin. INT6, external interrupt source means that it can serve as an external interrupt source. T3. Timer/ counter, is an input source.

PORT E, bit 5 has the IN5/OC3C functions built in for the pin. INT5, external interrupt source means that it can serve as an external interrupt source. OC3C, which stands for the output compare match C output. The PE5 has the capability to serve as the External output for the timer/counter 3 output compare C; however it must be configured to be an output. Note the pin can also be an output pin for the Pulse width modulation mode timer function. PORT E, bit 4 has the IN4/OC3B functions built in for the pin. INT4, external interrupt source means that it can serve as an external interrupt source. OC3B, which stands for the output compare match B output. The PE4 has the capability to serve as the External output for the timer/counter 3 output compare B; however it must be configured to be an output. Note the pin can also be an output pin for the Pulse width modulation mode timer function. PORT E, bit 3 has the AIN1/OC3A functions built in for the pin. AIN1, Analog Comparator Negative input, the pin itself is directly connected to the negative input of the Analog comparator unit. OC3A, which stands for the output compare match A output. The PE3 has the capability to serve as the External output for the timer/counter 3 output compare A. However, it must be configured to be an output. Note the pin can also be an output pin for the Pulse width modulation mode timer function. PORT E, bit 2 has the functions AIN0 and XCK0 built in. AIN0 is the Analog Comparator Positive input. It is connected to the positive input for the Analog comparator unit.

XCK0 or USART0 is an external clock. The built in DDR, DDE2, controls whether the pin is an input or an output. This function will only run when the UASRT is in a synchronous mode however. PORT E, bit 1 has the function of PD0 and TDX0, an SPI serial programming data input. This pin will connect to our transceiver and to our GPS module through a mux. Note that this pin will be used as a data output line for the microprocessor. The TDX0 function is for UART0 transmitting. PORT E, bit 0 has the function of PD1 and RDX0, an SPI serial programming data input. This pin will connect to our transceiver and to our GPS module through a mux. Note that this pin will be used as a data input line for the microprocessor. The TDX0 function is for UART0 receiving. If the pin is set up as an input it will be an input regardless of how DDRE0’s direction. This means that the USART0 forces the pin to be an input and PORT E0 will turn on the built in internal pull-up.

Figure 3.3.11 shows a chart representation of the functions of the various pins. The pin that we care about on this port are the UART pins and the PWM pins, the GPS module that we are looking into using is going to be plugged into these UART transmit and receive pins. While more motor controls or servo motors are going on the PWM pins.



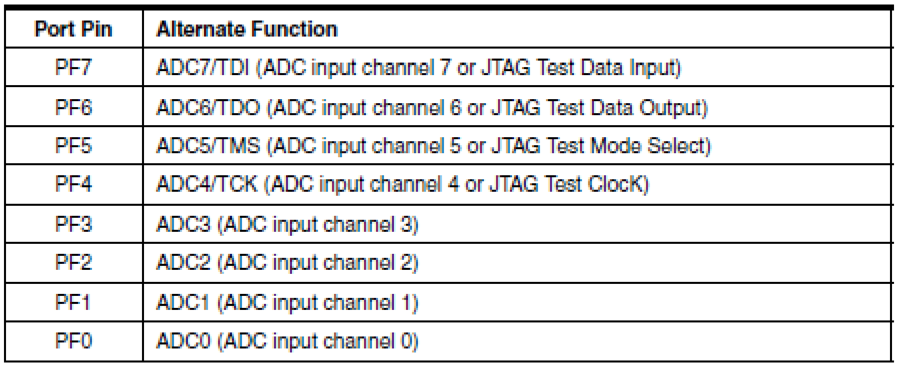
Port Pin	Alternate Function
PE7	INT7/ICP3[(1)] (External Interrupt 7 Input or Timer/Counter3 Input Capture Pin)
PE6	INT6/ T3[(1)] (External Interrupt 6 Input or Timer/Counter3 Clock Input)
PE5	INT5/OC3C[(1)] (External Interrupt 5 Input or Output Compare and PWM Output C for Timer/Counter3)
PE4	INT4/OC3B[(1)] (External Interrupt4 Input or Output Compare and PWM Output B for Timer/Counter3)
PE3	AIN1/OC3A [(1)] (Analog Comparator Negative Input or Output Compare and PWM Output A for Timer/Counter3)
PE2	AIN0/XCK0[(1)] (Analog Comparator Positive Input or USART0 external clock input/output)
PE1	PDO/TXD0 (Programming Data Output or UART0 Transmit Pin)
PE0	PDI/RXD0 (Programming Data Input or UART0 Receive Pin)

Figure 3.3.11: Functions of the Port E Pins
Reprinted with Permission by Atmel

PORT F

This port will be the port that will takes in all of the analog signal inputs for the Analog to Digital Converter, also known as the ADC. If this functionality is not used then the port can be used as a typical 8 bit I/O port with internal pull-up resistors. This port is different than the other port in that its output buffers have symmetrical drive characteristics. These pins are tri-stated, meaning that there is an active, low and reset setting. The alternative functions for these ports pins are very volatile. They cannot be switched when a conversion is occurring or else the converted signal will give a distorted signal. Note: if the JTAG pin interface is enabled on, the resisters on the pins PF7, PF5, and PF4 will be activated. PORT F, bit 7 is called the ADC7, which stands for Analog to Digital converter, channel 7. The pin has alternative functions of TDI, Test Data In, in which a serial input signal is shifted to an Instruction register or the Data register. This function occurs only when the JTAG interface is enabled. PORT F, bit 6 has alternative functions of TDO and ADC6. The ADC6 function is an analog to digital converter. TDO stands for test data out. This function will send serial data out from the Instruction or Data Registers. This function only works when JTAG is enabled. Note that TDO is a tri-stated pin that is if TAP states otherwise. PORT F, bit 5 has an ADC, also known as analog to digital converter (ADC) channel 5. The pin also has a built in TMS function which is short for Test mode Select. This pin is used for navigation in the TAP-controller state machine. Like other pins in this port, the function only works, when the JTAG interface is enabled. PORT F, bit 4 has the function capabilities of TCK and ADC4. ADC4 is the analog to digital converter channel 4.TCK is the JTAG Test Clock: the JTAG function is synchronous to the TCK. This like in other pins in PORT F will occur when the JTAG interface is enabled. The PORT F, bits 3-0 all only have the functionality to be analog to digital converters.

Figure 3.3.12 below shows the functions built into the pins of PORT F. Port F has all the analog to digital converter pins that we will not be used since we are planning to use all digital sensors. We might connect the JTAG port for programming the microcontroller on the board itself without having to lift off the microcontroller.



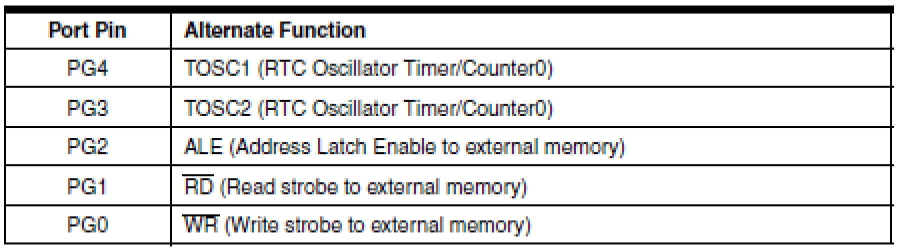
Port Pin	Alternate Function
PF7	ADC7/TDI (ADC input channel 7 or JTAG Test Data Input)
PF6	ADC6/TDO (ADC input channel 6 or JTAG Test Data Output)
PF5	ADC5/TMS (ADC input channel 5 or JTAG Test Mode Select)
PF4	ADC4/TCK (ADC input channel 4 or JTAG Test ClocK)
PF3	ADC3 (ADC input channel 3)
PF2	ADC2 (ADC input channel 2)
PF1	ADC1 (ADC input channel 1)
PF0	ADC0 (ADC input channel 0)

Figure 3.3.12: Functions of the Port F Pins
Reprinted with Permission by Atmel

PORT G

PORT G is different than the other ports. Its alternative functions are all really just timer signals or enable bits. Also PORT does not have 8 pins, but rather 5. PORT G, bit 4 is called the TOSC1. TOC1 stands for Timer Oscillator pin 1, which has the functionality to enable asynchronous clocking of Timer/Counter0 when the AS0 bit in ASSR is set (one),and when pin PG4 is disconnected from the port. It then starts inverting the output for the Oscillator amplifier. In this mode, a crystal oscillator is connected to this pin, and as such, the pin cannot be an I/O pin. PORT G, bit 3 is called the TOSC2. TOC2 stands for Timer Oscillator pin 2, which has the functionality to enable asynchronous clocking of Timer/Counter0 when the AS0 bit in ASSR is set (one),and when pin PG3 is disconnected from the port. It then starts inverting the output for the Oscillator amplifier. In this mode, a crystal oscillator is connected to this pin, and as such, the pin cannot be an I/O pin. PORT G, bit 2 is called the ALE. ALE stands for the Address Latch Enable. This pin tells the microprocessor that it is connected to external memory and indexes its memory location. PORT G, bit 1 is called RD and is the external data memory read control switch of the port. This port in tells if the data can be modified or not. PORT G, bit 0 is called the WR and is the external data memory write control switch of the port. This port in tells if the data can be modified or not.

In **Figure 3.3.13** below, the functions of the port are shown more clearly. In these pins we only need to worry about using the 2 Oscillator pins (PG4 and PG3) which we need to connect a resonator or a crystal to use as an external clock.



Port Pin	Alternate Function
PG4	TOSC1 (RTC Oscillator Timer/Counter0)
PG3	TOSC2 (RTC Oscillator Timer/Counter0)
PG2	ALE (Address Latch Enable to external memory)
PG1	$\overline{\text{RD}}$ (Read strobe to external memory)
PG0	$\overline{\text{WR}}$ (Write strobe to external memory)

Figure 3.3.13: Functions of the Port G Pins
Reprinted with Permission by Atmel

Memory

The ATmega128 has two main on-chip memory spaces, the Data memory and the Program memory. There is also an EEPROM memory for additional memory storage. The program storage on the chip is 128 Kbytes of In-System Reprogrammable Flash memory. The Flash is organized as 64K by 16 bit because all instructions are either 16 or 32 bits long. The Flash memory has a lifetime of about 10,000 write/erase cycles. The program counter (PC) has 16 bits, so it can address 64K memory locations easily. In the normal configuration for the SRAM, there are 4096 internal SRAM data memory locations. All of which is a bit wide. There are 32 registers and 64 I/O registers. Although there are many pins and functions for specifically external memory, our project would not need them so I will not talk about them in detail. The program memory will have to hold all of the hardware functions that will be needed to successfully control the blimp. As of right now the programmer can conceive of at least 20 functions that will need to be coded with the hardware programming. The first will be a function that can read in all of the inputs from the various modules that are built onto the PCB. This function will tell the modules to turn on and most likely reset every one of them so they get rid of any garbage data that they currently have and get new data instead. Another function that will have to be coded into the program is the servo motor controls. All of the servo motor controls are controlled by a PWM signal that tells the motor when to move and by how many degrees. This will be used in the servo motor in the tail wing, the servo for the propulsion fan rotor, and in various servos the camera system. There will need to be a function that can turn GPS coordinates from the transceiver and make the numbers into an array that will store the number and also be able to compare them as well. The comparison will tell us how much farther and in what direction the blimp will have to go. There will likely be more programs that will be written on the internal memory. The few functions that were stated above were just examples of such.

System Clock and Clock Operations

Because we are using I^{2}C serial communication for our blimp, we must have synchronous timers and counters. The ATmega128 microcontroller is controlled with an AVR clock control unit. This unit gets inputs from the timer/counter Oscillator and a clock multiplexer that connects the External RC Oscillator, External Clock, crystal oscillator, low frequency crystal Oscillator, and finally a Calibrated RC Oscillator. In the **Figure 3.3.14** below shows the clock unit in detail.

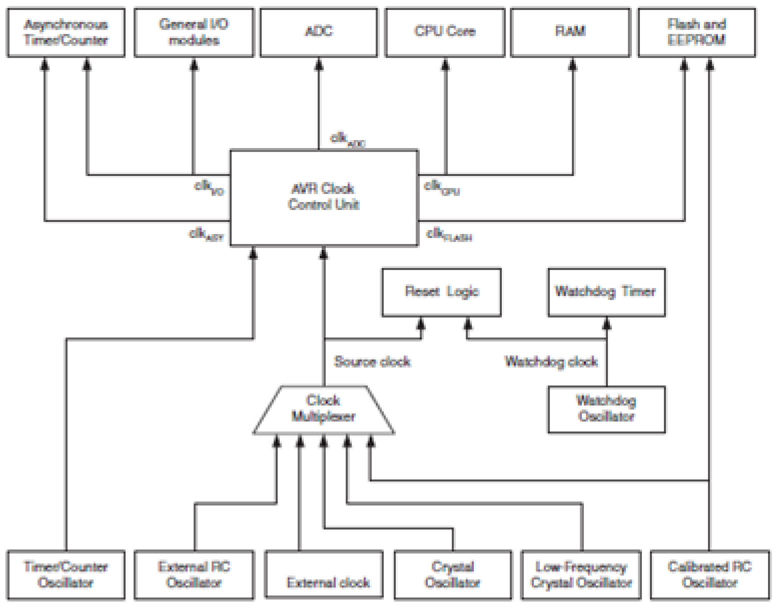


Figure 3.3.14: Clock Layout of ATmega128
Reprinted with Permission by Atmel

Serial Communication

The AVR ATmega has two USART inputs. The USART serial communication is a very old and becoming more and more obsolete now that there are easier and simpler serial communication languages. For example, I^{2}C is now becoming the industry standard in serial communications. The conundrum that is occurring in our project is that most of our modules have I^{2}C and our microprocessor does not have such built in ports. We will likely connect this to our transceiver module so we can send and receive data to our microprocessor. There is however, SPI serial communication pins and these can go very well with I^{2}C. The only drawback might be for us to use a serial signal converter that will be on our PCB. This would take up quite a lot of space but would be very handy for us.

The AVR uses SPI (Serial Peripheral Interface) for serial communications. This allows for high-speed synchronous data transfer between devices and the microcontroller. The interface has many features that will help us. including, a Full-duplex, three wire Synchronous Data transfer, Master and Slave Operations, a Write Collision Flag protection feature and a Double Speed (CK/2) Master SPI Mode. **Figure 3.3.15** below shows the SPI Block diagram and the pins that will need to be programmed for the microcontroller interface.

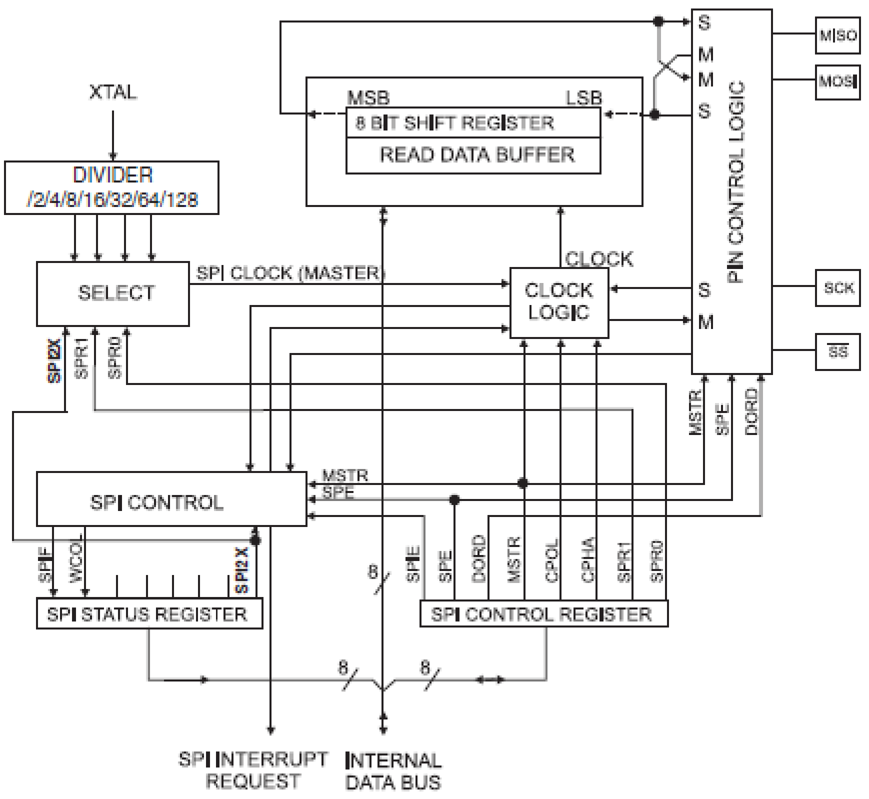


***Figure 3.3.15**: SPI Block Diagram of ATmega128*
Reprinted with Permission by Atmel

Interrupts

In the AVR ATmega, there are special functions that will tell the microcontroller that something is not acting properly and if the function that is currently being carried out is not as important, the microcontroller will interrupt the current program and go to the more important one. This special function is called the interrupt, and it is vital to any and all programs. Because our blimp is fragile, there are many things that have to be taken into consideration when attempting flight. I can think of at least five interrupt functions that will be needed in our microcontroller. The first one we will need is one that sends a flag when there is too little power. This will be useful to tell the client or programmer that there will either need to go back to the ground station for a battery replacement or to recharge the camera. There is also the possibility of the blimp to just use the last remaining units of power to just lower in altitude so it can be grabbed by the client or programmer. This is a good idea because the blimp is lighter than air so it should in theory not fall to the ground and just stays suspended in air, another interrupt that we will need is one that will tell the programmer or client that the blimp has gone out of range of the ground station. This will be huge for our project. If this occurs, the blimp should turn around and head for the last GPS coordinate that is in the memory. Another possibility is that the blimp can remember the GPS coordinates of the ground station and try to return to it automatically. There should be an interrupt when the gyroscope is giving off a strange input signal. This would tell us that the blimp has gone off balance and would need to correct itself. There would need to be many functions that would do the self-correcting of the blimp. Depending on the direction at which the gyroscope is point towards, there should be a function to counterbalance the blimp. Another great interrupt that should be implemented is a slow down or even stop when the accelerometer reads a number that is too high. This will help the client and programmers because although it guarantees our project won't be speeding away like a bullet (not that a blimp could) it will be much more secure and we would risk less damage to our prototype. The last interrupt that should be immediately implemented into the program is a flag that tells if the power to the ground station has been cut off for some reason or another. If this occurs, the blimp should end the current function it is in and lower altitude. This function would be a lifesaver because the ground station is really the heart of the blimp. It gets the controls from the user and transmits it to the blimp. If power were to be lost in the blimp there would be serious repercussions when it comes to getting controls from the user or to getting GPS coordinates for the blimp to fly to. Another interrupts that could be implemented is if any of the modules start to malfunction. To implement the interrupts, there will flags set up in the program and they will connect to INT0-7 source pins. These are the External Interrupt request pins. There might be a need for the analog comparator and other timer/ counter pins for some of the other interrupt functions requests. The AVR has 32 sources that will allow us to have lots of freedom in deciding how to implement the interrupt functions.

Pulse Width Modulation Needed for Servo Motors

The pulse width modulation signal is needed to control the servo motors. The servo motors are controlled by a rising edge clock cycle. This means when the PWM wave goes into the servo motors input pin, the servo will turn only when it hits the high part of the wave. It will also stop moving when it hits another high rising edge. The pulse width modulation control is needed for only this purpose. The pin that controls the pulse width modulation is the counter/timer0 in PORT D. this pin also has a frequency generator as well so this will be great in controlling the length and how long the servo is supposed to turn. To make the Pulse Width modulation wave, the wave first comes from the waveform generator that is built into the pin. In **Figure 3.3.16**, below shows the block diagram of the pin and the components of what makes the Pulse width modulator. The question of this will be how we will connect the pins to every one of the servo motors. And an even better question would be how will we decide which servo motor will be controlled. These questions will probably be answered in the section highlighting the servo motors.

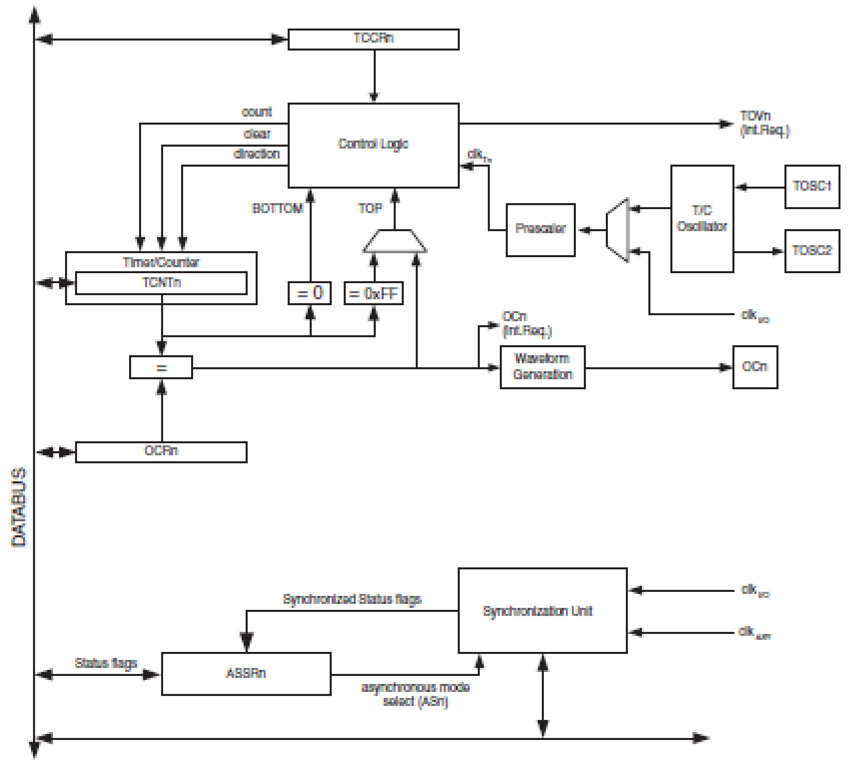


Figure 3.3.16: Pulse Width Modulation Block Diagram of ATmega128
Reprinted with Permission by Atmel

AVR ATMEGA 328

This microprocessor is also famous for being in the Arduino Microcontroller boards. This line of microprocessors is also made by AVR and has Harvard RISC architecture. Because this microprocessor, like the 128, is so famous, many programs have been written and placed on the internet for free use to the general public. The AVR ATmega328 has 32 general registers that all connect to the Arithmetic Logic Unit. This allows two independent registers to be accessed in one single instruction on a single clock cycle. This allows for super-fast computing, up to 1 MIPS per MHz! This allows the programmers to optimize power consumption in the program. It is however much smaller than that of the ATmega128. Because of which many of the initial components that were in the blimp’s PCB would have to be moved to the ground station. The ATmega328 is being considered being used as a backup microcontroller is we run into any major problems with the ATmega128.

In **Figure 3.3.17** below, the Pin Layout of the ATmega328 is shown below, although it has less pins than the ATmega128 it still serves the purposes of our project to a certain extent. We will just not have extra pins in case we need extra GPIO ports for unforeseen connections.

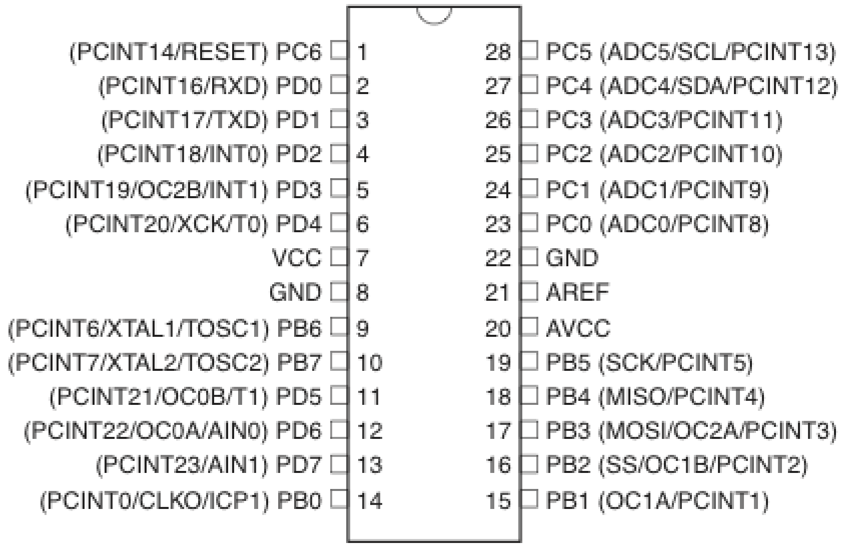
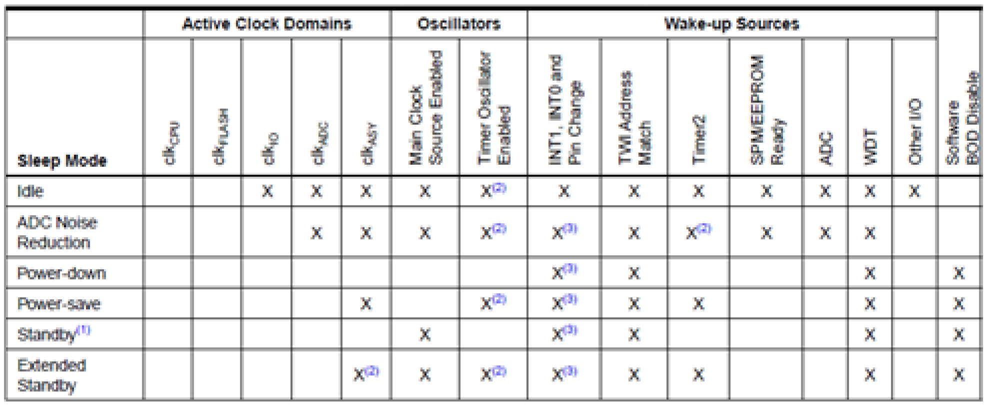


Figure 3.3.17: Atmel ATmega328 Pin Layout
Reprinted with Permission by Atmel

Power

The AVR ATmega328 is very similar to the 128 power consumption wise. It has a low to decent operating voltages of 1.8 - 5.5V. However there are many features that are built into the system that allow for making it a very lucrative option for being our microcontroller. The AVR has many various sleep modes that let us reduce the power consumption from the device. This power management comes from the MCU control Register. It can allow for a sleep mode if Bit 5 is enabled. Note when programming the sleep enable mode it is recommended to write the bit to one before its execution. An even better feature is in the MCU register bits 4-2. These bits when enabled in a certain way make certain parts of the microcontroller sleep. This will be very useful for the times when the Blimp does not need to do anything but wait to receive a signal from the ground station. In idle mode, the CPU is stopped but the SPI, USART, and Analog comparator; analog to digital Converter, two-wire serial interface, timer/counters, watchdog, and the interrupt system continue operating. This will likely be the mode at which our blimp will be operating majority of the time. The noise reduction mode is when everything except the analog to digital converter, external interrupts, the two-wire serial interface and timer zero will be turned off. This does not seem like it would be useful for use because the serial communications will likely always need to be on for the control of the blimp. The other two modes, power save and power down, might be useful when the blimp is at a low altitude and docked at the power station. Even then it might be better to just turn the power off to the device. We will decide this when the project is actually implemented. **Figure 3.3.18** below shows the various pins that will wake up the MCU out of the various sleep modes.



	Active Clock Domains					Oscillators		Wake-up Sources							
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Oscillator Enabled	INT1, INT0 and Pin Change	TWI Address Match	Timer2	SPM/EEPROM Ready	ADC	WDT	Other I/O	Software BOD Disable
Idle			X	X	X	X	X(2)	X	X	X	X	X	X	X	
ADC Noise Reduction				X	X	X	X(2)	X(3)	X	X(2)	X	X	X		
Power-down								X(3)	X				X		X
Power-save					X		X(2)	X(3)	X	X			X		X
Standby(1)						X		X(3)	X				X		X
Extended Standby					X(2)	X	X(2)	X(3)	X	X			X		X

***Figure 3.3.18**:* Atmel ATmega328 Sleep/Wake-up Mode
Reprinted with Permission by Atmel

I/O ports

The AVR has 23 general I/O ports and pins and they are needed to get information from and to the various modules. Amazingly, like the 128, each pin has alternative functions on top of the general purpose I/O ports.

The alternative functions also allow for very many possibilities when programming the device. Port A has an alternate function as analog input for the ADC. The input channel can take in from PA0-PA7 analog input bits. It also allows for comparison between the various channels.

Each pin takes in or gives out 8 bits of data. The microcontroller has 32 general purpose registers for operations. This microcontroller has been optimized for the AVR Enhanced RISC instruction set. So from an assembly programmer's standpoint, it looks very similar to MIPS. The instruction set might not be that important, seeing how we'll most likely program in C. However, the possibility of programming in assembly is open because assembly code is takes up less space in the memory of the microcontroller and tends to be faster as well.

Memory

The AVR ATmega 32 microprocessor has 3 linear and regular main spaces for memory, the Data Memory and the Program Memory and EEPROM memory for data storage. The memory in the microprocessor will be used for three main purposes:

1. To store the program into the hardware. This will be done on the In-system Reprogrammable Flash memory.
2. To store interrupt and functions to call them in memory.
3. To set up the functions that will control the servo and modules that will ultimately fly the blimp.

SERIAL TRANSMISSION

The AVR ATMEGA has one USART serial input pin. But because we have two modules that send serial communications there might be issues with we will likely connect this to our transceiver module so we can send and receive data to our microprocessor. If we set up the transceiver to the only serial port then the GPS module will have to be connected to the ground station. Which in all honesty would not be such a bad idea, but that would mean that the project would always need to be connected to the ground station and would need a bit of tweaking in the functions to decide whether or not it has completed the flight path correctly.

Section 3.4 PIC Family of Microcontrollers Comparison and Research

For the ground station design, we decided to use a PIC microcontroller made by Microchip in its design layout with the communication modules and a USB connection to the computer. We were considering using an Atmel AVR but decided that it had too many features that we were not going to use on the ground station. The ground station is designed to act as an intermediary device between the blimp and the computer or android phone, so all we need it to do is send and receive signals. We would like a decently powered microcontroller with only the features we need, we want to send the data as fast as we can but it is not required since the blimp will only need to send back its information every 2 or 3 seconds.

The PIC microcontroller is perfect for what we need since has lower power consumption, cheap price, and it uses a small Dual Inline Pin (DIP) packaging. The pricing for the PICs are between $2-5 each for the low to mid-range ones that we are considering. The power needed to power a PIC is from 2 to 5 Volts and we will be using a small 9V battery to plug into the ground station which will supply enough power to the PIC. The speeds of the PICs go up to 50 MHz which is a lot higher for the low cost than the Atmel ATtiny which only go up to 20 MHz.

In **Table 3.4.1** below, we will be comparing 2 PIC Family Microcontroller, the PIC16F887 and the PIC18F2550, and the 1 Atmel ATtiny, the ATtiny2313. These microcontrollers were picked out by their input pins features, such as they must have the SPI pins, UART pins, and enough memory to support our programs. Another key feature in picking out these are the high amount of support in forums with a wide variety of transmission and receiving modules such as Bluetooth and wireless communications. Since this is used in the ground station and not the blimp we do not have to take the weight into consideration and the power consumption is not that important because a 9 volt lithium battery or a 9 volt wall plug AC adapter will be used to power the device and can be easily replaced on the ground if it dies.

	PIC16F887	**PIC18F2550**	**ATtiny2313**
Speed	20Mhz	48Mhz	20Mhz
Pin Package	40-pin DIP	28-pin DIP	20-pin DIP
Digital Communication	1-UART 1-A/E/USART 1-SPI 1-I2C 1-MSSP(SPI/I2C)	1-UART 1-A/E/USART 1-SPI 1-I2C 1-MSSP(SPI/I2C)	1-UART 2-SPI 1-I2C
USB Interface	None	USB 2.0 Supported	None
Program Flash Memory	14KB	32KB	2KB
Data EEPROM	256 Bytes	256 Bytes	128 Bytes
Operating Range	2 to 5.5 V	2 to 5.5 V	1.8 to 5.5V
Current Consumption	- 11µA @ 32 kHz, 2.0V, typical - 220µA @ 4 MHz, 2.0V, typical	- 11µA @ 32 kHz, 2.0V, typical - 220µA @ 4 MHz, 2.0V, typical	- 230 µA @ 1 MHz, 1.8V - 20 µA @ 32 kHz, 1.8V
Timers	2 x 8-bit, 1 x 16-bit	1 x 8-bit, 3 x 16-bit	1 x 8-bit, 1x16-bit
Price	$ 2.63	$ 4.13	$ 2.88

Table 3.4.1 *- Comparison of PIC Microcontrollers*

Section 3.4.1 PIC Programming

Since we are most likely going to go with the Peripheral Interface Controller (PIC) microcontroller over the Atmel ATtiny, we will discuss the PIC programming. To program the PIC, we need software provided by Microchip called MPLabs, it's their Integrated Development Environment (IDE) to develop applications for Microchip microcontrollers and digital signal controllers. The current version we are going to use in MPLabX and it is required that we have a programmer kit to load the program onto the PIC. The latest version is the PICkit3 but we will be using a Canakit PICkit2 Programmer since we have access to it. The drawback is that we can't program some of the newer PICs but the PIC16F887 and the PIC18F2550 are both supported on the PICkit2.

The programming language we decided to use is C18 which is just the C programming language extension for MPLabX, it is very similar to C programming we have all done before and has a lot of support on forums. If there are problems with programming in C18, we will use PIC Pro Basic, which our group is more familiar with. With either language we need to read over the Datasheet for the PIC to find out the Port name for each Pin and program the direction of the pin (input or output). We will need to use TRISA through TRISD to find out which channel the pins we want are on and set it to a binary number to configure it to input and output. Then there is a method to assign a variable name to the channel of the Port name we need to use for the pin. We need to read through the Datasheet to also figure out how to setup the interrupt pin and then during the interrupt period we will use that to change either change the digital transmission lines to send out to SPI or UART for the ground station. The nice thing about using C18 is that there are for loops and while loops that we can use to accomplish this task. If we choose to use PIC Pro Basic, we will have limited access to the use of doubles, so we would lose precision in our calculations but that will not matter for our purposes.

Section 3.5 Programming Languages (C++ versus Java)

For designing an easy to use program for the user to interface with our blimp, we decided to implement a Graphical User Interface (GUI) for them to use. The requirements for this program are that it must be able to use libraries for low-level interfaces to hardware. The hardware considerations we need are the USB connection to the ground station which have to use UART libraries to send commands to and from the blimp, and then a USB controller that can interface the GUI. There is an alternative where we write the libraries ourselves but it will be easier to use premade libraries and just modify them for our program as needed. We also want the programming language to be object-oriented so we can easily separate out our project into different modules. An object-oriented language is nice so that we can have multiple people working on different sections and then merge them together as long as we know the inputs and outputs of each method. The final requirement is that it must be able to support OpenCV libraries to use the video feed from the blimp camera and then recognize objects within it in real time. OpenCV libraries were design for and compiled in C++, but now they offer interfaces for C, Python and Java interfaces. The OpenCV libraries are also supported on all the major Operating Systems such as Windows, Linux, Mac OS, iOS and Android so the programming languages' development operating system environment does not matter. Another interface the programming language must support is the Logitech Controller we are implementing, we want to use DirectInput which is the native support for programming with the controller, and its base programming language is C++. For programming the GUI, we are considering using either C++ or java; both have its pros and cons.

Section 3.5.1 C++

The C++ programming language is considered an intermediate programming language since it has high level function such a premade classes for structures but also has some low level programming design which is perfect for coding for hardware. The C++ programming language can compile C code and most of the C programs can be ported easily over to a C++ program. C++ has many dependencies that must be on the computer before you install your C++ program, such as some DLL window files or large collections of libraries such as .NET Framework. The Integrated Development Environment (IDE) that we would use with C++ is Microsoft's Visual Studio Professional 2010. There will be no cost using this IDE as it is free to students from Microsoft's DreamSpark Website.

Section 3.5.2 Java

The Java programming language come later and was influenced by C++, it is an object oriented language and focuses on high level program, so the hardware programming with Java is non-existent. Java is more self-contained than C++ and its dependencies are just the Java runtime environment. Many of C++'s libraries are ported over to Java or has a way to implement things like DirectInput which is called JXinput for java. If we wanted looks over functionality, Java would be a very compelling choice since it provides excellent graphical user interface design with the Swing, AWT, or JavaFX classes.

Section 3.5.3 *Differences in Java and C++*

In **Table 3.5.1** below, it shows the comparison of C++ and Java; many of the features are the same but some many times one coding language does better than the other in dealing with the matter. The main thing to note in this first half of the chart is the use of the overloading methods and the pointers as well as the speed. Since the OpenCV library will require a great deal of processing power the speed of each compiler is very important and we would want to go with the compiler that can utilize the most of the processor.

	Java	**C++**
Pointers	No Pointers, Java creates references to an object that you create.	Supports Pointers, Can select any point in memory
Speed	Runs slower than C++, but some Just In Time Compilers provides some speed up.	Runs 20x faster than Java because of the Windows Environment Support.
Deconstructors	Objects cannot be destroyed, an automatic garbage collector destroys objects for you at random times when they are not used again. Need to instead create a cleanup method.	Objects need to be destroyed manually by the user and can be done at any time to be freed from memory.
Primitive Types	Both support the same primitive's types.	These types include Boolean, char, byte, short, int, long, float, and double.
Method Overloading	Both have the same support for creating methods with the same name.	This is important when we have multiple people working on the program (as long as the methods have different inputs/outputs).

***Table 3.5.1** - Advantages and Disadvantages of Using Java and C++ (Part 1)*

In **Table 3.5.2** below, it shows the comparison of C++ and Java continued. The main thing to take away from this chart is that both have a hierarchy system, but C++ is more flexible with their inheritance. But Java has a superior Visual Design capabilities compared to C++ which might be more intriguing to us.

	Java	**C++**
Hierarchy	All objects inherit from the root class Object. (Single Forest)	Can start new inheritance tree from anywhere. (Multiple Forests)
goto Method	Does not Exist. (uses "Break" instead)	Can use goto to return to any part of the code.
inline Methods	User cannot create their own functions within a method without calling the method from another place.	Available in C and C++ compiler.
Exception Handling	Superior to C++. Exceptions are caught and enforced at run-time.	Calls a function when an exception is thrown.
Hardware Access	Cannot directly access hardware. Java has to use native methods to call a function written in C or C++.	Can run and compile direct hardware access on its own.
Visual Design	Superior to C++. Can work with visual programming environments across from a variety of vendors. Uses Java Bean Standard, making it easier to learn.	Harder to learn vendor-specific environments. The availability of visual components is limited on C++.

***Table 3.5.2** - Advantages and Disadvantages of Using Java and C++ (Part 2)*

Section 3.6 *Android programming*

Android is a Linux based OS and is currently one of the main operating systems in the mobile market. It uses a modified version of java language to program its applications, called "apps". These apps are bought and downloaded on the Google play market. As of May 2013, there have been over 48 billion apps downloaded. In terms of market share, Android is on its way to becoming the dominant OS in the mobile and smartphone market. There are many complex reasons why this is occurring, but to a programmer the reason is a simple one, android is open source and allows free access to its programming language. This translates to letting programmers mess around and gives them an opportunity to make something with little to lose when an app fails or does not work properly. It is because of this freedom that the group decided to try our luck with android programming. The group is under the impression that android will be the dominant OS in the market similar to how Windows is to the PC world. We think getting into that world would be vital to our careers as Electrical and Computer Engineers. In order to become familiar with the new and uncharted world of Android programming, we decided in the final version of our project, we will be sending the video input to an android capable device and that device will also be able to control the direction of the flight or the blimp and the view of the camera feed. Since the programmers are in very preliminary stages at the moment, we are not sure how we will implement all of the user inputs into the small space of an android phone, but we will eventually put it together.

LG Optimus V

The phone we will be using will be the LG Optimus V. It is a low end smartphone that runs the Android OS. The Optimus was known to be a very good smartphone for people just entering the smartphone and android market. Now there are many smartphones and tablets in the Optimus line, but the Optimus V will always be one of the forebears. Because the Optimus V is so old, and cannot run the latest versions of Android OS, it uses the Android 2.2 (Froyo) Operating system. The video resolution for the phone is 640x480 by default and can be set to lower resolutions such as 320x240 and 176x144 pixels. The phone has built in Bluetooth capabilities. The Bluetooth version it has is the version 2.1 +EDR (Enhanced Data Rate). There is also Wi-Fi connect ability to the industry standard IEEE 802.11b/g.

Specs

The phone’s radio communication technology is CDMA. The frequencies it can read are-1.9GHz CDMA PCS. The processor in its system is the MSM7627 dual core processor: 600 MHz applications processor and a 400 MHz modem processor. The phone’s memory has an internal memory of 160 MB. With a micro SD Memory slot that can expand it to 32 GB.

Section 3.7 Communication

The communication to and from the PCB will be vital to the success of the surveillance blimp. As of now, there are going to be one input signal and two output signals. The input signal will be from the user. This signal will tell the microcontroller to move servo motors, either the ones that are latched onto the GoPro (if the user wants to get a better view) or the ones that are attached to the propellers (the user wants to change the direction of the blimp) depending on what the user inputs are. As for the output signals, the first of the signals will be of a video output from the GoPro camera. This will be sent directly to the user by an IP or a Wi-Fi connection. From the company website, this signal will be able to broadcast up to 100 yards!

Section 3.7.1 Transmission Modes

There are several different transmission methods available to us with different transceiver modules. In these sections we will explore an old standard called narrow band and some of the newer standards Direct Sequence Spread Spectrum and Frequency Hopping Spread Spectrum. Then we will look at the data modulation used in this transmission mode, specifically how Gaussian-frequency-shift-keying (GFSK) works and how it is based off of frequency shift keying (FSK).

Section 3.7.1.1 Narrow Band Transmissions

Previous radio frequencies worked in narrow band which only allows for one frequency to be used at a time on each channel, these frequencies worked on the lower end of the spectrum, running at 27, 35, 36, 40, 41 or 72 MHz. The problem with old narrow band radio transmissions is that the channels that the frequencies are split up into are only a tiny sliver of space. This causes interference between two channels if they are very close to each other, the two kinds of interference that will be caused is glitches or lock-outs.

Section 3.7.1.2 Spread Spectrum Transmissions

Spread spectrum is a transmission technique that spreads a narrowband communication signal over a wide range of frequencies for transmission then combines it into the original data bandwidth at the receiver. There are two kinds of spread spectrum we are interested in, one is Direct Sequence Spread Spectrum and another is Frequency Hopping Spread Spectrum. Wireless transmission devices at the 2.4 GHz ISM band will use one of these two Spread Spectrum techniques. The Bluetooth module we are using will use the Frequency Hopping Spread Spectrum but be adaptive, so it listens in on the range of channels and picks the one that is clear to use. Frequency Hopping Spread Spectrum spreads the signal by hopping from one frequency to another across a bandwidth of 83 Mhz. Direct Sequence Spread Spectrum spreads the signal by adding redundant bits to the signal prior to transmission which spreads the signal across 22 Mhz.

In **Table 3.7.1** below, it shows the how the spread spectrum technologies clearly are more advantageous to use over the old standard of narrow band. We will look into only using spread spectrum transmission modules for our project so we will not suffer from all the disadvantages of narrow band.

Narrow band	**Spread Spectrum**
• Uses only enough frequency spectrum to carry the signal • High peak power • Easily jammed	• The bandwidth is much wider than required to send to the signal. • Low peak power • Hard to detect • Hard to intercept • Difficult to jam

Table 3.7.1 *- Comparison of Narrow Band vs. Spread Spectrum*

Section 3.7.2 Data Modulation for Transmission

We need to look at what two types of data modulation, the Frequency-Shift-Keying Modulation (FSK) and Gaussian-Frequency-Shift-Keying Modulation (GFSK) modulation modes. These modulation modes work by varying the amplitude, phase, or frequency of a high frequency carrier with the amplitude of the message signal. The spacing between the sine wave will determine if the signal is a 1 or a 0 for FSK modulation, the difference with GFSK is that it uses a Gaussian filter to smooth out the waveform to better determine the 1 or 0. The FSK modulations are done by having a 1 output when there is a 180 degree phase shift in the signal and a 0 output when there is no phase change.

Section 3.7.3 Interference and Issues with 2.4 GHz Band

When many people are using the same frequencies there can be cross-communication issues and errors. Although the 2.4 GHz band is spread out from a lot of channels we can still run into interference. There are some interference issue if there is a narrow band transmission running at 2.4Ghz, but the spread spectrum transmission should look for its own channel to use, this makes it more resistance to interference but not totally safe from it. The Bluetooth module uses Adaptive Frequency Hopping, so this should search and detect used channels so it will have higher resistance to interference from other devices using the 2.4 GHz Band.

Section 3.8 Transceiver Modules Comparison and Research

To communicate between our blimp and our computer/phone we need a long range communication module. We explored several option of communication which included Bluetooth, wireless and using transmitter and receivers. The Bluetooth module we decided will not get the range we needed to communicate with the blimp, so we instead are using the Bluetooth module to communicate from the ground station to the phone. Wireless also has a range issue and we have the IP camera using wireless to send the video feed on the blimp, so we did not want to use extra bandwidth on the wireless signal. So we decided to look into transmitters and receiver modules and found they have many frequencies available to use and have the long distance we need. Transceiver modules have been proven to work in the RC cars and planes models, so we are planning to model our transmission design after that.

When looking for transceiver modules we have a few requirements that must be met in order to achieve the altitude and the speeds we want for transmissions. We need the transmitter to transmit at least 100 meters if not more, the longer the distance the better. We also are looking for any test data on attenuation through buildings/objects, basically, will the blimp antenna lose track of the receiving antenna if the blimp does not have line of sight on the ground station, obstructions can include, tall buildings or trees, as well as other high rising structures. Another requirement is a low power consumption, if there is a high current draw, it will drain the battery without even having an hour of flight time, then we will not be able to power the blimp controls at all causing serious problems in retrieving the blimp. Then we would prefer the transceiver to use the 2.4Ghz ISM band since it is free for public use and we won’t have to get a radio license to broadcast on other frequencies, there is also the option of using the 900Mhz or 434 MHz frequency but it suffers from lower data rates for the cost of longer range that we do not even need.

In **Table 3.8.1** below shows three modules we are considering and compares the specifications we are looking for. The most important thing for us to consider is the transmission range of the transmitter, the data rate is also important but we rather have farther transmission range than a faster speed, though both would be nice to have but would be too expensive for this project.

	nRF24L01	**XBee 1mW Series 1**	**RFM22B-S2**
Frequency	2.4 Ghz	2.4 Ghz	260MHz to 960MHz (434 MHz Standard)
Antenna	External	On-Chip	On-Chip
Weight	4.3 grams	5.7 grams	5.6 grams
Power Consumption	11.3mA Radio TX at 0dBm 13.3mA Radio RX at 2Mbps on-air data-rate	3.3V @ 50mA	100mW at 3.3V
Max Range	750 meters	100 Meters	40 Kilometers
Data Rate	250kbps, 1Mbps or 2Mbps	250kbps Max data rate with 128-bit encryption	0.123 to 256 kbps
Transmission Protocol	Gaussian-Frequency-Shift-Keying	802.15.4 stack	FSK, GFSK, and OOK modulation. Digital RSSI.
Digital Communication Signal	4-pin SPI	8 digital IO pins 6 10-bit ADC input pins	16-pin with 8-bit ADC input
Price	$19.84	$ 29.95	$25.95

Table 3.8.1 *- Comparison of Transceiver Modules*

Section 3.9 *Bluetooth Modules Comparison and Research*

Since we wanted to implement an android phone application to control the blimp, we needed a wireless communication module to connect to the ground station. The alternative to a wireless communication is just to use a USB connection to the phone. Since we will be using a USB plug to connect the ground station to a Graphical User Interface (GUI) on a laptop computer, it will not be hard to just replace that with a female to mini-USB connection to connect to a phone.

When choosing the Bluetooth modules to consider we need to look at a couple things about the Bluetooth modules. The first thing we want is that the Bluetooth standard is at least Bluetooth v2.0, this is required so that the protocol used has adaptive frequency hopping so that we can avoid the interference from other 2.4Ghz devices that we are using such as the transmitter and the wireless IP camera. The range for the Bluetooth module is not that important since we do not plan to walk that far away from the ground station to control the blimp. The data rate speed is important though, we want the data speed at least as fast as the transceiver module so everything syncs together and no buffering is required in the microcontroller. To achieve these speed requirements we want the Bluetooth module to at least transmit at 2 Mbits/s, but the greater available speeds the better. An on-chip antenna is also fine for the purposes of the ground station as we will not need the ability to change the antenna and increase the gain of the Bluetooth module, if any tweaking of this setting needs to be done, the Bluetooth module should have software to change it.

In **Table 3.9.1** below shows two modules we are considering and compares the specifications we are looking for. When comparing the Bluetooth modules it is important to look at the range and the Transmission protocol. Most devices we saw would use the 2 pin UART digital serial communication method. The range on Bluetooth is pretty short but since it just has to communicate with a relatively close phone, for our purposes, this short range would be fine to work with.

	Bluegiga WT12	**Roving Networks RN-42**
Bluetooth Standard	Bluetooth v. 2.1 + EDR Class 2	Bluetooth v. 2.1 + EDR Class 2
Frequency Range	2.4 Ghz	2.4 Ghz
Power Consumption	Link active min: 7mA, max: 60mA Link active in sniff mode: 2.5mA Link active in park mode: 2.5mA Idle w/ deep sleep:<1mA	Standby/Idle: 25 mA Normal mode: 3 mA Low power Sniff: 8 mA Standby/Idle (Deep sleep enabled): 26 µA
Transmission Protocol	HCI over UART or USB	UART (SPP or HCI) and USB (HCI only)
Transmission Range	30 meters	18 meters
Antenna	Internal	Internal
Data Rate	2 Mbps or 3 Mbps	With onboard stack: 300Kbps HCI mode: 1.5Mbps sustained, 3Mbps burst
Price	$ 50.00 with breakout board ($27.00 without)	$ 15.94 without breakout board

Table 3.9.1 *- Comparison of Bluetooth Modules*

Section 4: Structural Design Details

In this section of the paper, we will cover the design that we are planning to make, since it is too costly to outright buy a blimp found online, we chose to make it ourselves and fill it with helium. Other things the section will cover are the anything going on or attaching to the blimp structure. We need stabilizing wings, as well as a gondola to hold all our materials onto the blimp. Then we need a camera holder which will house the rotating camera so we can add the surveillance part to our system.

Section 4.1 *Initial Blimp Design*

There are many different designs as far as airships go. There are, of course, the main three: rigid, semi-rigid, and non-rigid. Then there are the others which aren't really airships, but more of a weather balloon type. The reasoning behind the fact that we aren't using a rigid design is because our design is simply too small and won't be able to handle the weight restrictions. The same pretty much goes for the semi rigid; however, we could pull some strings and make it work. The only way to make that work would be to design a slightly larger scale system. Last we come down to the weather balloon style. Even though we could implement a camera system along with a propulsion system it would not be as ideal of a surveillance system because of the aerodynamics. That's why we are going to the non-rigid blimp design.

The main design will incorporate a ten-foot-long helium-filled blimp with four aerodynamic stabilizing wings attached, a main cabin-like unit, which will contain the electronics for the airship, two propellers that will be connected to one shaft, which will be conducted so it can rotate in order to control the flight patterns of the aircraft, and lastly, a camera stabilization system with camera.

Section 4.2 *Components of Blimp Design*

There are a few things that make up our MASS blimp.

- Envelope
- Welding material (glue)
- Stabilizing Wings
- Propulsion system
- Blimp car (Gondola)
- Camera Stabilization System
- Camera
- GPS

Section 4.2.1 Envelope Design

The first step into implementing the entire design of the MASS together is to have the envelope perfected. The scheme, material, weight factor of the material, weld (glue), and the factors of the other parts of the airship are all in play and must be precisely calibrated for the entire system to work as a whole. The scheme of the blimp design will be in the traditional ellipse-shape; big, round, and wide in front, which in turn, gets skinnier towards the tail. The envelope will be made from several pieces welded together in order to make one, smooth looking 'balloon' when pressure from the gas added. Most likely, the magic number for this project implementation will be eight identical pieces due to the fact that the more pieces there are the smoother it will look when under pressure (kind of like making a circle out of a bunch of straight lines). The number of sections around the circumference multiplies any inaccuracy when not measured correctly, thus the sections have to be particular. **Figure 4.2.1** shows how the design.

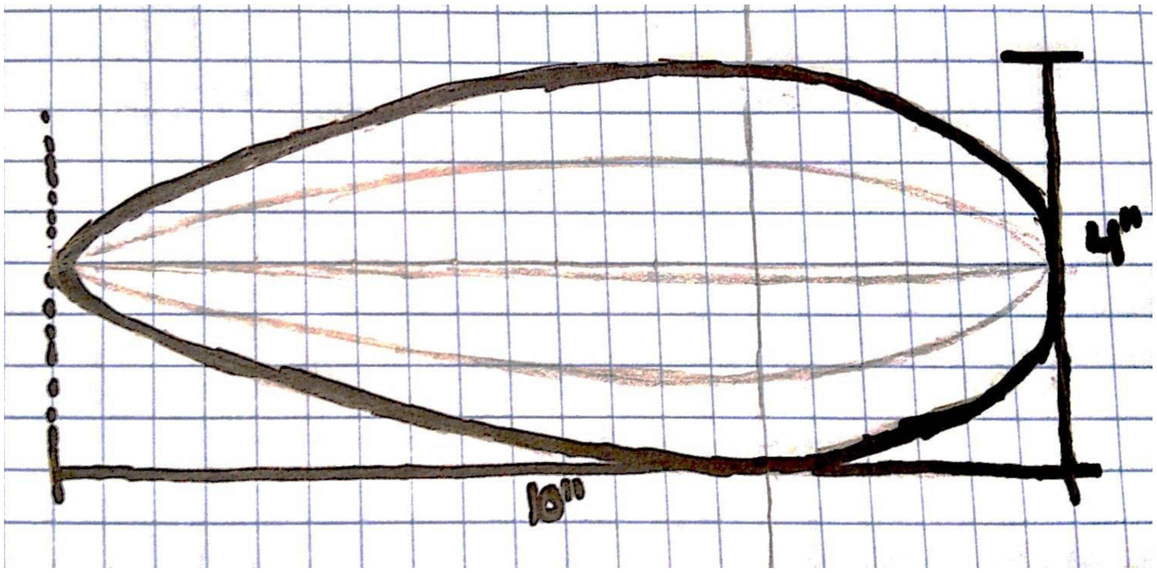


***Figure 4.2.1**: The figure above is the main profile of the blimp design. When the design is complete we should have an envelope with a total length of 10 feet and a total width of three to four feet.*

The other main focus of the build for the envelope is on the type of material we need to use. There are many factors in choosing the material including weight, ply, texture, and the fact if it is air tight. For this we will be using a type of nylon. Nylon is great for holding air and is also light. The problem with these materials is they are hard to find from an actual supplier. Many businesses sell the envelopes; however don't sell the actual material in order to build your own.

Section 4.2.1a Section Design

The design of the sections will be determined depending on the full size of the blimp. Considering we are designing the blimp to be about ten-foot long, we are looking for the blimp to be around three to four feet wide. In conclusion, to design the sections, we will draw the final envelope size and shape, cut it in half with a line (long ways like a hot dog), and use that as a template for the design [**Figure 4.2.2]**.The section design will be according to arc length (A) from the front tip of the blimp and the widest radial part (R) to the maximum top point of the blimp. After taking the measurements from Figure 1, we take the A and the R and use the equation [2*(pi)*R]/X, where 'X' is the number of sections we are using (in our case X=8). Calculating the equation, we have our final width for our sections **[Figure 4.2.3]**. Now, we just connect the points of interest with a smooth curved line and we will have everything we need to implement the final design. The sections will be made from a material only being about twenty to twenty five microns thick. In order to make the final envelope work with the eight different sections and be light enough to be 'stable' (just about floating) when filled with the appropriate amount of helium along with the added weight of the electronics and mechanics we need to weld the sections together.

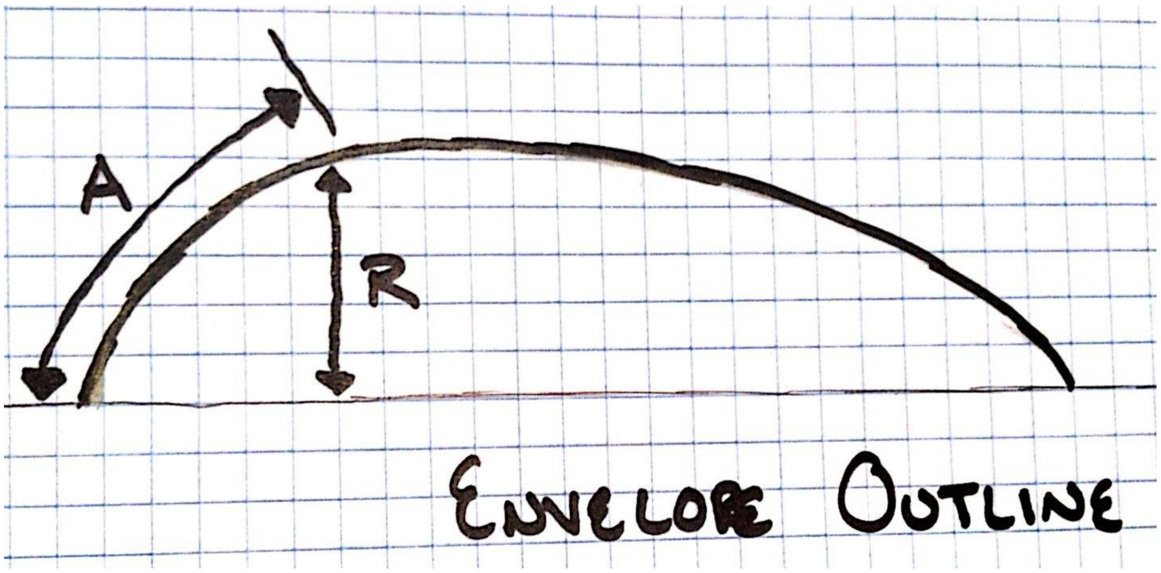


Figure 4.2.2*: Having the profile of the total size of the envelope we want to use as a template and having a large sized piece of paper laid out we need to sketch out the profile in its entirety. This will allow us to collect the right information for our sections by cutting the profile in half and using the arc length (A) and the largest radial section (R) and using those for our values in the section creation.*

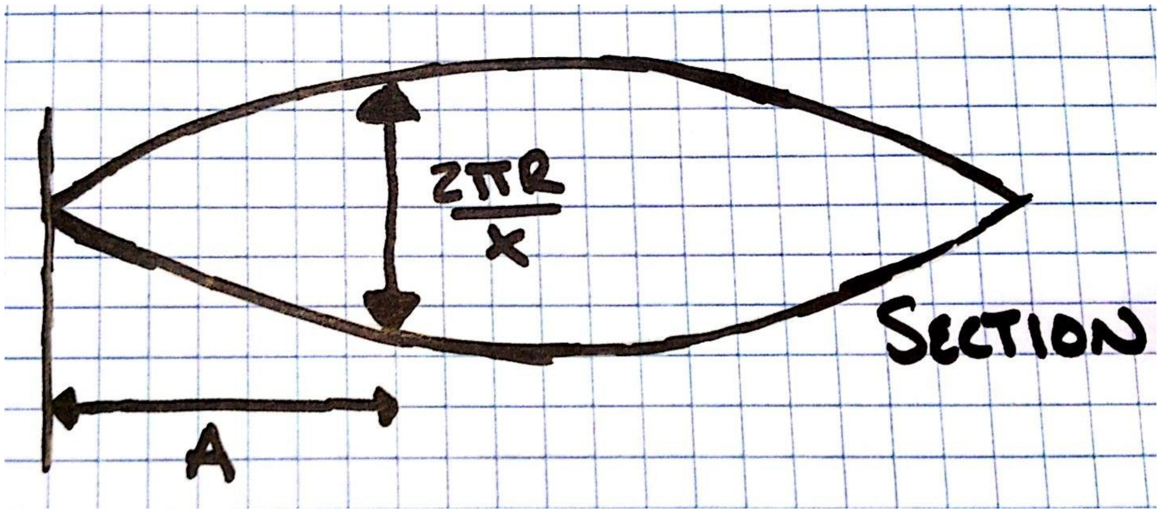


***Figure 4.2.3:** After finding the arc length and radial value of the profile we will start measuring to get our sections. By using the design in the figure above for each of our sections, we will make a total of eight and then start the mending process.*

Section 4.2.1b Joining the Sections

One of the trickiest parts in designing the blimp will be creating the perfect envelope in order for flight to be flawless. So far, we have how we will create the sections, but now it's time to discuss how we will join the sections together. In order to weld the pieces together we will use one of two designs. The first design is just simply over lapping the edge of the consecutive pieces with glue under the edge of the overlapped section which is shown in **Figure 4.2.4** below.

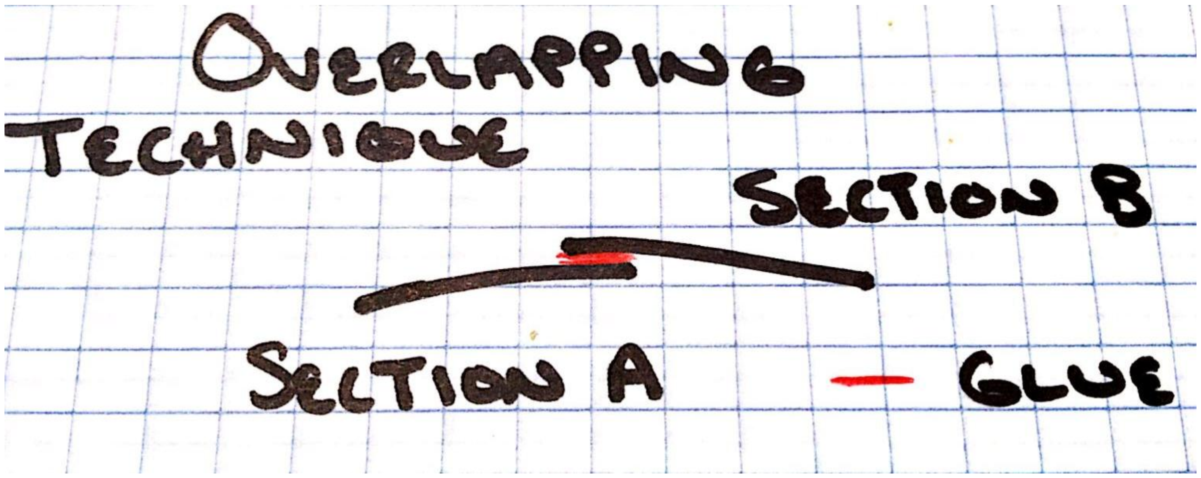


Figure 4.2.4: The figure above shows how the overlapping technique will be done with the sections. By overlapping one section over the other we will have a smaller radial profile, but will also reduce some of the weight from the total design.

The second, and most likely the one to be used in the final design, is a webbing type fuse **[Figure 4.2.5]**. The best example of this is the webbing tape you put on drywall joints. The tape goes over the crack; you put speckle spread over the top, sand it down and you're good to go on painting. In our case, we will be using this method except the difference is the 'tape' will be just a strip of the material, and the glue will go on the down side of the strip attaching the two joints together. On top of this process, heat will be applied to get the tensile strength up on the joints and kind of melding the nylon material as one. However, this process isn't going to be as easy as painting a wall.

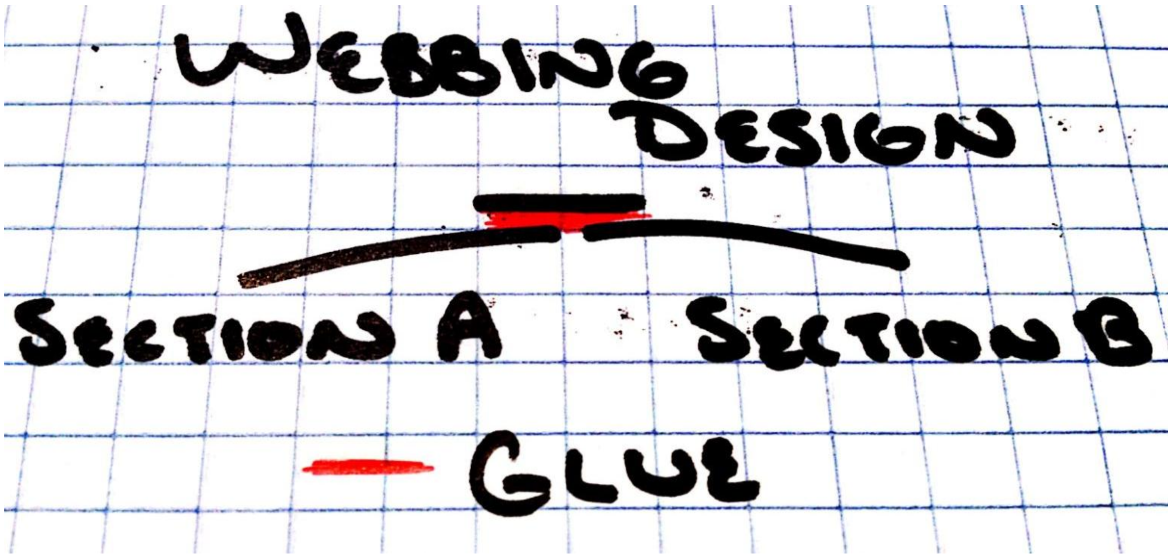


Figure 4.2.5: The figure above shows the design we will be using in order to complete our envelope. Even though using more material will add some weight, it will be a more suitable option in order to seal our envelope completely and will hold the pressure when the helium is added.

The sequence we will be taking in joining these sections together will need some type of contraption in order to hold everything in place as the welding process takes hold. The blueprint for this union is to have something in the shape of the envelope in order to guide the joining process as shown in **Figure 4.2.6** below.

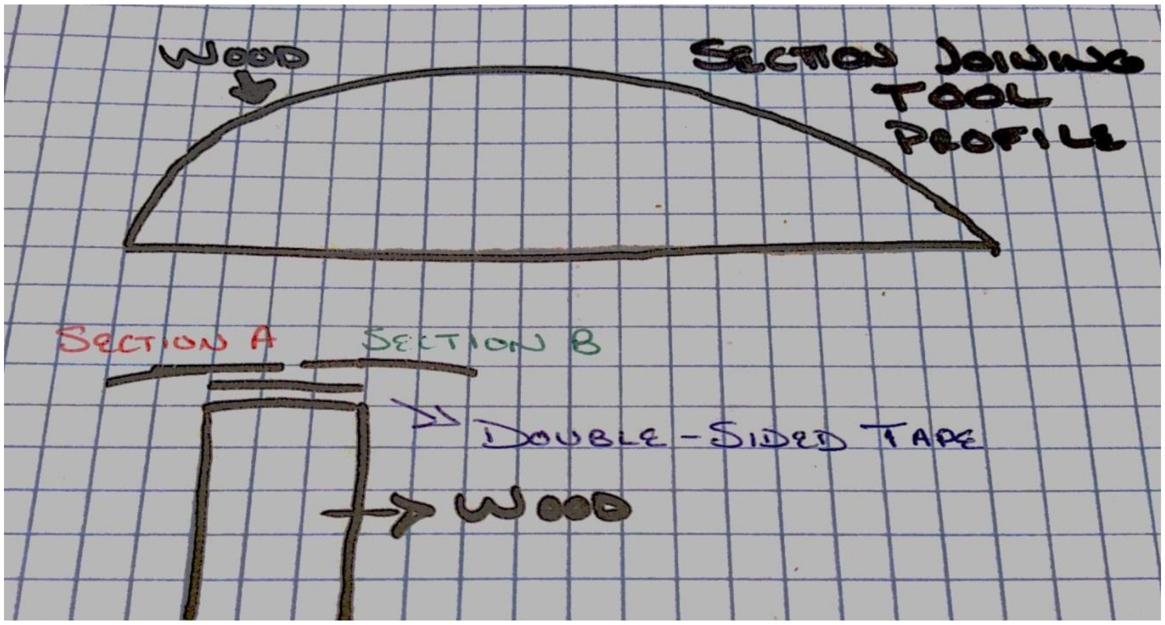


Figure 4.2.6: The tool we will be using has the same profile as shown in the above figure. In order to have the two sections held on to the tool while we mend them together the double-sided tape will act as a holder.

The arc will provide the extra room for being able to stretch the material and put it on double-sided tape (for a quick hold) as the webbing process with the glue can ensue. Using the weak, double-sided tape, we will be able to take the envelope off the mechanism without any damage to the material with also trying to prevent any sagging or wrinkles. Starting at the front tip, the sections will be lined up and applied to the tape with absolutely no overlap in addition to not having any gaps. Accurateness will need to be had to get the most sleekness when the envelope is finished. Once the two sections are lined up we will apply adhesive to the joint. Cutting a strip of 'tape' from the same material we used for the envelope sections, we will tack on a layer of glue on one side of the strip (the downside) and place it over the joint. After we set everything we will hit the seam with some heat. When all is said and done, we should be able to pull the two now combined sections off the contraption with no difficulties. All the sections will be done this way until the final result. **Figure 4.2.7** shows how the tool will be used to combine the different sections of nylon.

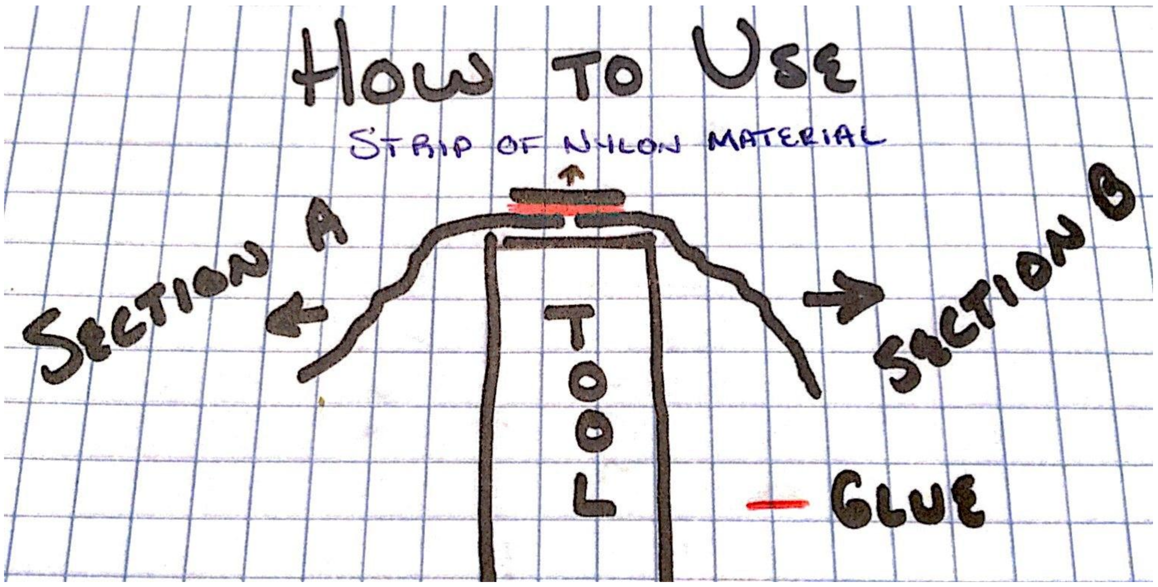


Figure 4.2.7: The figure above shows how we will use the tool. Put the two sections on the tool, stretch them out and them use the strip of nylon material with glue and mend the two sections together.

The tail end of the blimp will be left a little open for us to be able to work inside the envelope if need be (LED, wiring, etc.). Once we are ready to seal up the balloon, we will need to put the inflation tube in the end. By cutting off the very ends of the sections (because they are probably not going to meet), we will design a cone type disc section with the inflation tube sticking out of it and glue it to the end of the envelope. After this process is complete we will be ready for testing by filling the MASS with air and checking for leaks. Once the leak check is finished this will complete the envelope building method we will be using.

Section 4.2.2 Stabilizing Wings (Tail Fins)

Like many of the blimps you see in everyday use in today's world, we will be using the common, yet, very effect design of implementing four tail wings into our architecture of the MASS **[Figure4.2.7]**. They will all be equal in size; however, the only difference will be the tail wing which will be attached toward the rear underside of the aircraft. This tail will have our third and very important rear rotor propeller in it. This not only will help stabilize the entire system, but will also aid in the navigation of the blimp from left to right **[Figure 4.2.8].** The other three stabilizing wings will be in their usual placement; one on each side and one on top.

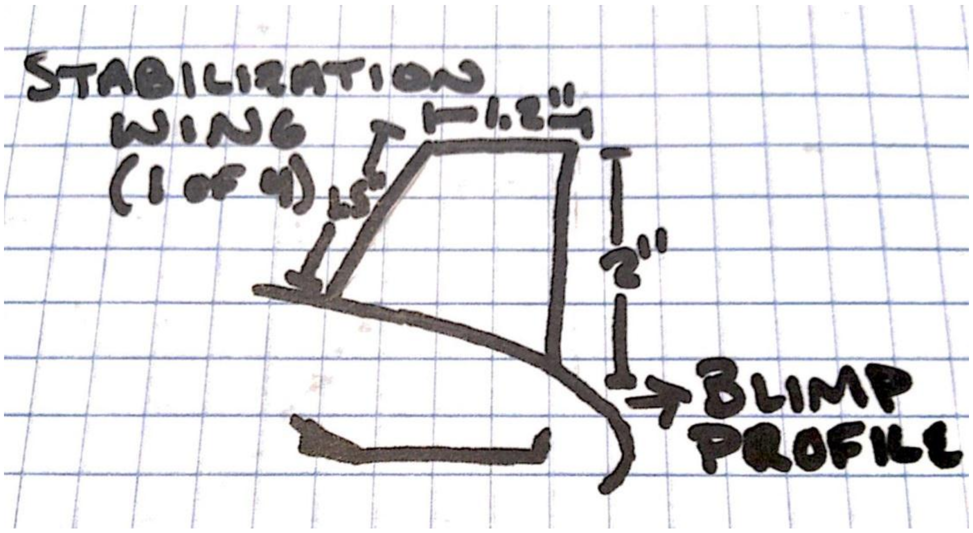


***Figure 4.2.7:** The figure above shows the top stabilizing wing. There will be four total; top, left, right, bottom.*

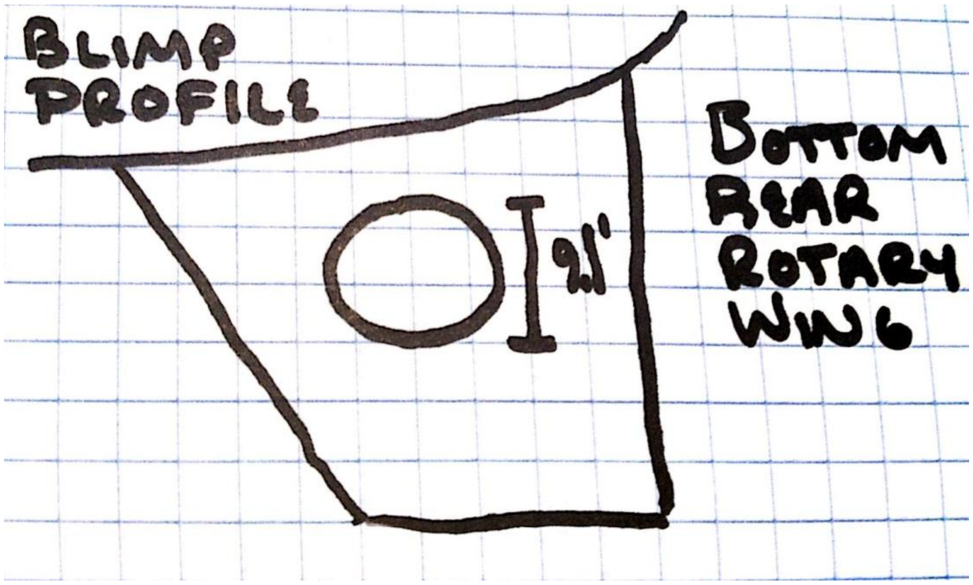


***Figure 4.2.8**: The bottom rear rotary wing will have the third propeller designed into it. Since the propeller is nine inches long we will cut the hole to be slightly bigger in order to be able to fit the motor assembly.*

The materials used to make these can, as well, be made out of several different types of materials. Since we are working with a smaller type aircraft we will want something that is sturdy, but yet very light. Again, weight restrictions are in place. We can use carbon fiber, lightweight wood, aluminum, plastic, etc. Even though all the materials listed are only among the few that could be used, we will most likely use a dense and thin style of Styrofoam since not only is it cheap, but it is replaceable with ease. The one consideration, however, is for the bottom wing. Carbon fiber molding is going to be our best bet when it comes to weight as well as being able to mold a clean looking structure for the stabilization wing with the propeller. Since the bottom wing will implement the rear stabilization rotary propeller, we may consider using a form of plastic. Further design details are needed to come to this decision.

Section 4.2.3 The Gondola (Hull)

When designing the style of airship we were going to use, we decided that the elliptical shaped blimp will suit our surveillance system the best. Not only do you have aerodynamics, which will help the system stay straight and true, but you also have extra space to implement many things into the overall system with less hassle. Among these things is the gondola. As explained before in the history section of this report, the gondola is pretty much the central headquarters of the airship. In the manned versions of a blimp, the gondola is usually where the pilot(s) and any passengers are located and then toward the rear of the gondola would be where the engines, electrical components, burners, etc. would be. However, in our version, since the gondola is obviously a smaller, unmanned part of our surveillance system, this is where all the electrical components will be located. Everything from the microcontroller to the servo gear working the main propellers will be located in the gondola and it will act as a housing unit **[Figures 4.2.9 and 4.2.10].**

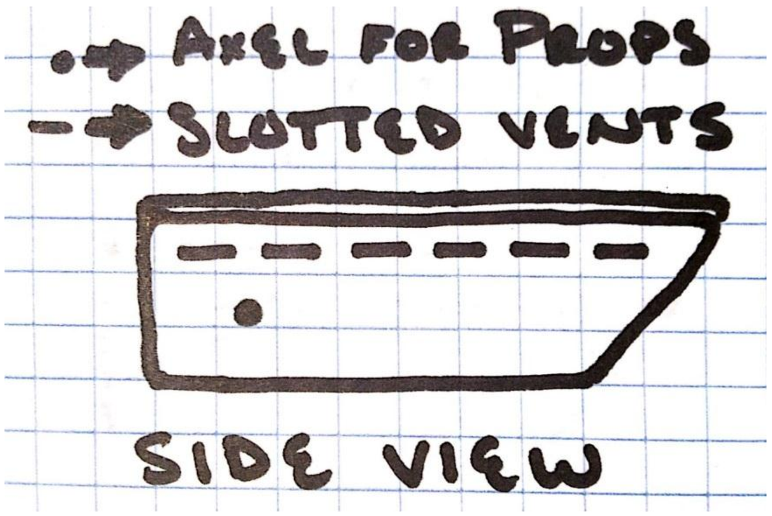


Figure 4.2.10*: This is the side view of the Gondola. As you can see in the figure, there are slotted vents toward the top of the design. The dot depicts where the axel will come through for the propellers.*

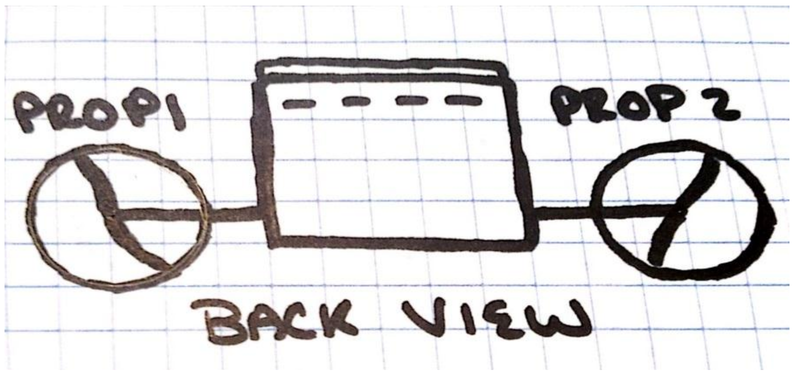


Figure 4.2.9*: This is the rear view of the Gondola. Here you can see where the axel for the propeller comes out. It will be able to rotate freely within the Gondola.*

The gondola can be made of many different types of materials. Most people in the RC (remote control) community use premade hulls, which are made out of a lightweight plastic then, once they build their circuitry, paste the gondola to the blimp's envelope. Other materials used can be anywhere from fiber glass to carbon fiber. Most likely, the material we will be using is going to be made from a very lightweight plastic, but will implement a top design that does not completely seal, however, it must latch. The basic design will be melting pieces together to make one solid unit with a latch system for the top piece. The reason we want some sort of top is so we can anchor the gondola to the envelope with a cross-member-like foundation in order to guarantee that we can take the unit off, but we do not want it to completely seal on the top in order to be able to dissipate heat properly to safeguard proper and continuous functionality **[Figure 4.2.11]**. Going back to the material for the gondola, even though plastic would be the more ideal way to go; carbon fiber material will be lighter. So, with that being said, we are going to build the plastic hull and make a carbon fiber replica with that piece. After each piece is made we will way the options on which to use.

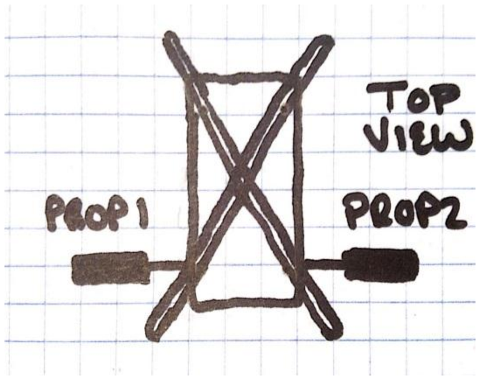


Figure 4.2.11: *This is the top view of the Gondola. The two bars creating an X will be configured in order to be attached to the top part of the control box and will also have an anchor system set up attached to the bottom of the envelope.*

Section 4.2.4 The Camera Stabilization System

The camera stabilization system will be gone over in detail in a later section; however the basic focus and details can be reviewed now. We have a couple different designs in mind. However, the design we will be using is a two part design. The first part **[Figure 4.2.12]** is the housing unit which stabilizes itself on a rotational stand point. It attaches to the bottom of the blimp. At the moment, we are planning on putting the camera system on the bottom of the Gondola, but we may put it near the front of the system.

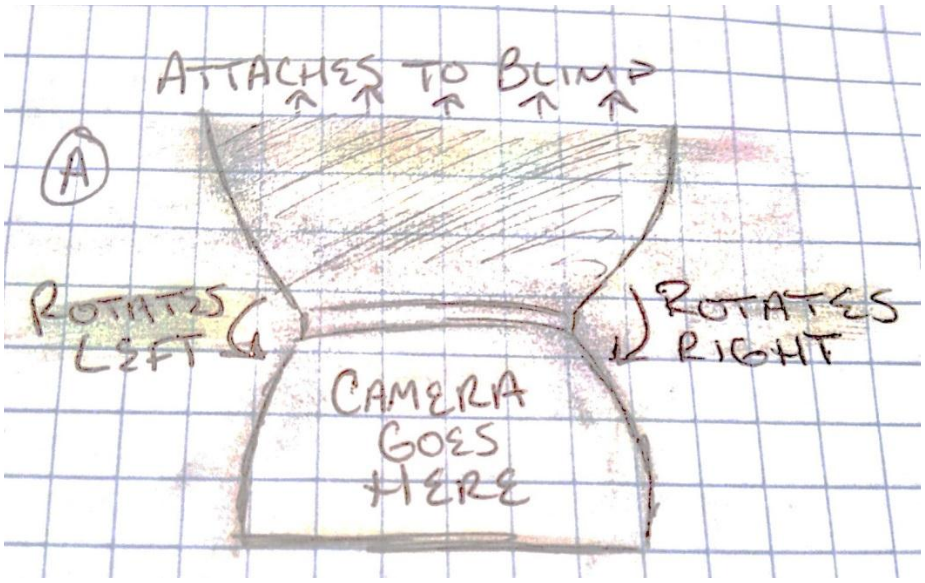


Figure 4.2.12: Part A of the camera stabilization system. Part B in the next figure goes on the platform depicted in the Figure above.

The second part of the camera stabilization system is the vertical component **[Figure 4.2.13]**. As the first part controls the rotation of the stabilizing, this part controls the vertical stabilization of the system. Thus, giving the system a smooth and fluid camera video feed to the computer system.

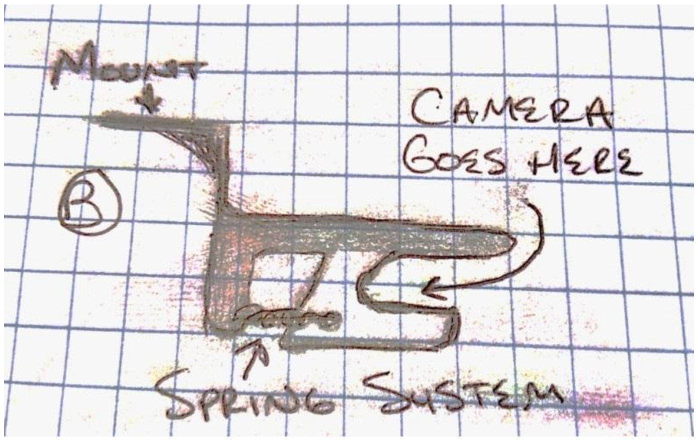


Figure 4.2.13: Part B of the camera system.

Section 4.3 Detailed Blimp Schematics

Since we have gone over the main parts of the MASS blimp it's time to review how the entire system will look. The following two figures are the side view and front view of the MASS. In each of the figures, everything is labeled and clearly shown that will be used for the completion of the project.

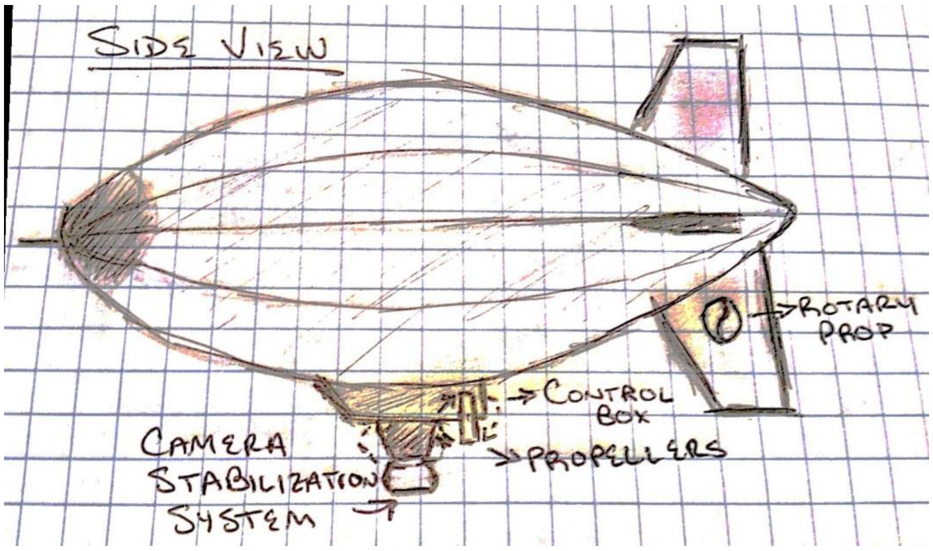


Figure 4.3.1: *Figure above shows the side view of the MASS.*

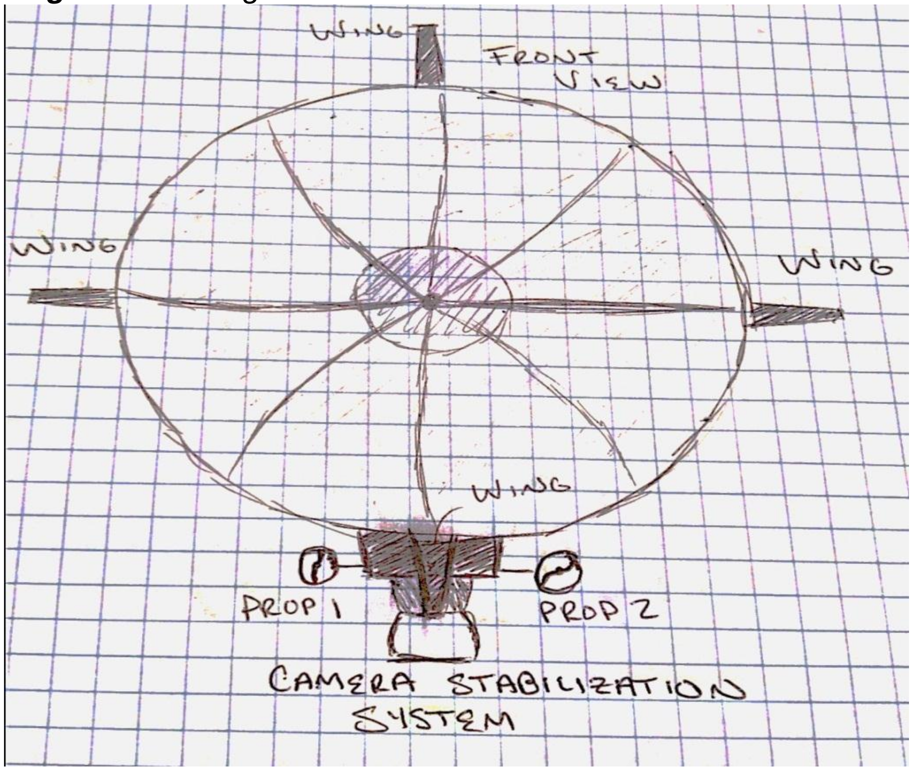


Figure 4.3.2: *Figure above shows the front view of the MASS.*

Section 4.4 Motors and Prop Design

For the motors and propeller design we are using a simple, but yet ingenious design in order to achieve maximum and optimal lift and turning power. The system will be a totally of three motors and propellers. Two of the three will be powering the MASS's lift, decline, and thrust rates. The other, as mentioned in the previous sections, will be controlling the left and right directions. For the motors, we are planning on using the Turnigy L2210C-1200 brushless motor (150w) **[Figure 4.4.1]** for our design basically because it not only packs a punch for the little guy, but it's also cheap, seems reliable from the reviews, and is only fifty eight grams in weight. The design we will be using is the two motors will be attached to an axis rod on either end. In the middle of this rod will be a servo gear motor, which is connected to the microcontroller but a small gear set and ultimately controlling the rotation and direction of the thrust.

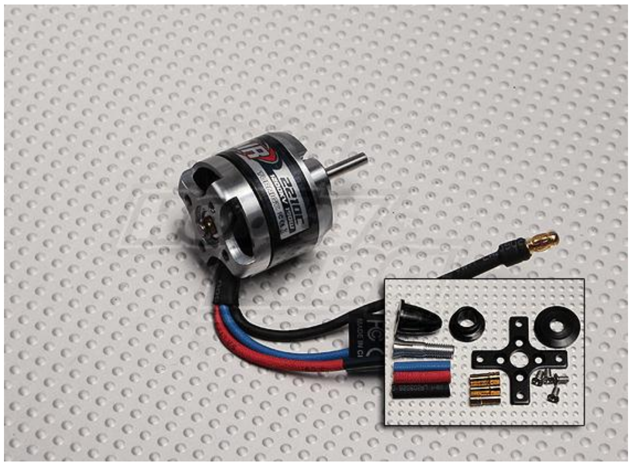


Figure 4.4.1: *Turnigy L2210C-1200 motor. Reprinted with Permission by HobbyKing*

The servo gear in which we will be using to control the rotation of the thruster axis is going to be the HXT900 9g / 1.6kg / .12sec Micro Servo **[Figure 4.4.2]**. It is not only one of the most famous servo motors used by hobbyists globally, but it is known to be one of the most trusted and reliable motors not to mention one of the cheapest out there. 9g stands for the weight of the motor, 1.6kg stands for the torque rating, and the .12 sec stands for it being able to change the rotation by sixty degrees in twelve one-hundredths of a second. Below are the tables and diagrams for the motors and servos.

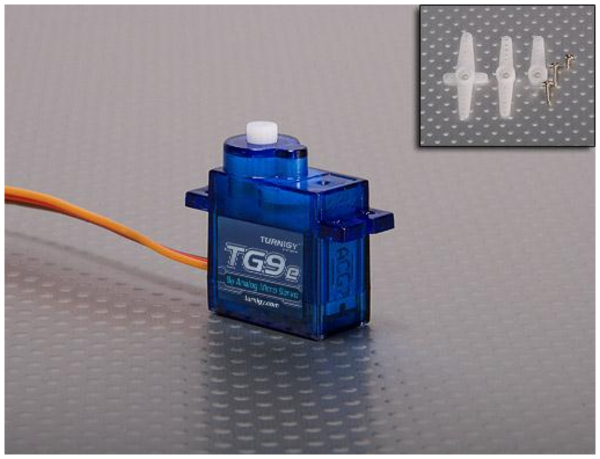


***Figure 4.4.2**: HTX900 servo motor. Reprinted with Permission by HobbyKing*

Although we have the power house behind this set up, we still need the thrusters themselves. In the case will be attaching nine inch propellers to the motors **[Figure 4.2.3].** We chose the nine inch propellers for a few different reasons. For one, it gives us plenty of torque availability and will be able to produce a decent amount of thrust. Secondly, if any alterations need to be made we can do so accordingly. For example, if there is a balance issue of some sort we can always cut the propellers to be shorter. This kit provides us with three pusher and three puller propellers. Since we have two main thrusters we will be covered on either option, and as far as the rear rotary propeller, the circuitry design will come into play on which way the thrust direction is produced.

***Figure 4.4.3:** H9011A 9x5 Propellers. Reprinted with Permission by HobbyKing*

Section 4.4 Electronic Speed Controllers (ESC)

One consideration we forgot to take into account for in senior design 1 is the electronic speed controllers (ESC) required to send the PWM signals to the brushless motors. The amp requirement to run the motors were around 15-20 amps and we could not power that directly from the microcontrollers. So we had to look up and find information about a suitable motor controller. We tinkered with the idea of using a MOSFET circuit to provide the PWM signal but we did not want to risk the amp requirement and possibly burn out the motors. The electronic speed controllers are connected to the motors, which are connected to the batteries while the input wires are connected to the microcontroller, thus being controlled by that said microcontroller. The motors we are using need an arming sequence, the arming sequence consist of sending a PWM signal of 0 to the electronic speed controllers and then sending the minimum and maximum PWM signal lengths, which is implemented by the microcontroller. Our ESC's are rated at 30 amps on 3 cells and is controlled by pulse wave modulation, which is well over the requirements to control our 20 amp rated brushless motors.

Section 4.4 Final Blimp Structure Design

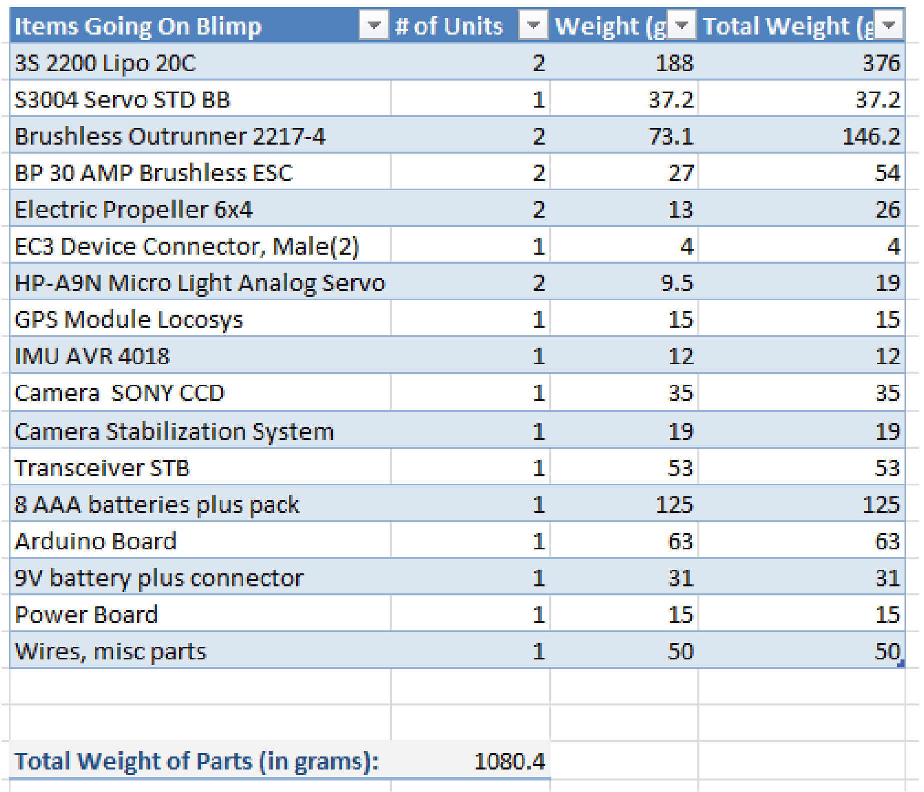
For the final blimp design we ended up venturing off the generic build that was originally planned. We still went with the elliptical design, however, instead of welding the sections together with glue we ended up melting the material edges of the sections together with a heated iron and using the over lapping technique. Also, instead of eight skinner sections, we ended up going with six fatter type sections. So instead of the final design being the original ten feet it is now around seven and a half feet by four feet wide. The nylon material ended up being more of a plastic urethane material. Some of the issues we are having with this material is the fact that the viscosity of the material is not good enough to properly hold and maintain the pressure from the air. In order to fill the envelope with air for testing we made a makeshift tube using the same material the blimp system is made from. Then in order to close it we roll it and clamp it with a strong, lightweight clip.

As far as the gondola we did end up building it out of balsa wood. However, instead of having the motors and pitch axel designed into it we decided to make a separate motor mount. The gondola is now just holding the main electronics. The motor mounts hold the motors and pitch axel with the main servomotor and is connected to the electronics using wire extensions from the gondola system.

Instead of using a third propeller to help turn the blimp system we decided to cut that part and just use the two main motors for the main steering controls considering the system doesn't need to turn quickly. The final design uses just four smaller fins for stabilization with no modifications to them with them being anchored down to anchors, which are beads glued to the proper areas. In order to anchor everything to the blimp we ended up using industrial Velcro so we can easily remove the gondola, motor mount, and stabilization wings without causing any damage to the envelope.

Section 4.4 Weight Considerations

The point of having a blimp system is to have a sense of weightlessness. That is why we have to have certain weight considerations. We added up all of our relevant parts including wires and miscellaneous parts and we came up with a figure of about 1080.4 grams. This does not include the envelope structure nor does it include the balsa wood pieces and small gear pieces. Since one cubic foot of helium will lift about 28.2 grams we projected our lift to be anywhere from 1900 to 2500 grams. **Figure 4.4.1** below shows the weight consideration table in which we calculated the weight of the system, we left a lot of weight in order to account for unforeseen addition weight like the material of the balloon or any extra modules we need to put in.



Items Going On Blimp	# of Units	Weight (g	Total Weight (g
3S 2200 Lipo 20C	2	188	376
S3004 Servo STD BB	1	37.2	37.2
Brushless Outrunner 2217-4	2	73.1	146.2
BP 30 AMP Brushless ESC	2	27	54
Electric Propeller 6x4	2	13	26
EC3 Device Connector, Male(2)	1	4	4
HP-A9N Micro Light Analog Servo	2	9.5	19
GPS Module Locosys	1	15	15
IMU AVR 4018	1	12	12
Camera SONY CCD	1	35	35
Camera Stabilization System	1	19	19
Transceiver STB	1	53	53
8 AAA batteries plus pack	1	125	125
Arduino Board	1	63	63
9V battery plus connector	1	31	31
Power Board	1	15	15
Wires, misc parts	1	50	50
Total Weight of Parts (in grams):	1080.4		

Figure 4.4.1: Estimated Weight of Components to go on Blimp

SECTION 5: Project Hardware and Software Design Details

In this section, we will cover the hardware layout on our system and give the technical features that we suit our purposes. We will cover the blimps' camera mount, the ground station, the blimp controller on the blimp and its attached modules such as GPS and accelerometer.

As can be seen by the following drawing we will have a stabilization system that will reside right on the underside of the middle of the blimp. There will be two main parts to the undercarriage of the blimp: the carrier which will reside right under the blimp and the camera stabilization part which will be under the carrier. Although both go together this section is specifically referring to the lower portion of the stabilizer. **Figure 5.0.1** shows the estimated dimensions of the blimp structure.

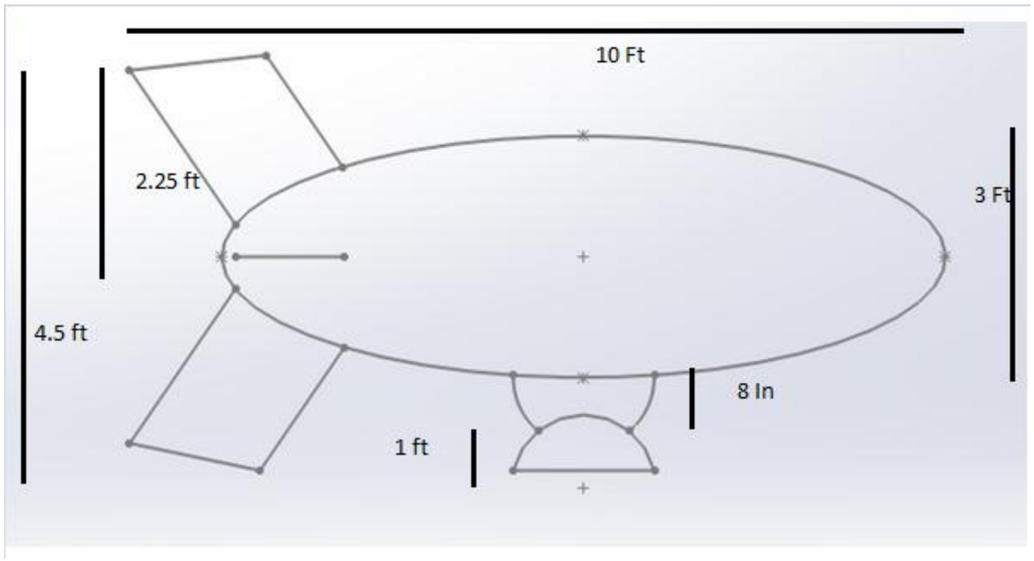


Figure 5.0.1: *Blimp with expected dimensions and expected mount and carrier*

***Section 5.1** Stabilization System*

Seeing as this is a blimp and only two small motors will be operating it the most important thing to once again consider is weight restrictions. A couple of materials we considered were Aluminum, Carbon Fiber, Wood and PVC. Starting with Aluminum, it is very durable, ductile, pliable and lightweight. It would be strong enough to hold the camera and malleable enough to make it into any shape necessary. Aluminum also has a strong elastic modulus which means that it will not deform when extreme pressure is put upon it. Approximately 1,000 kilo-pound per square inch of strength, which would be more than enough and honestly a lot more than, would be needed. As far as price goes, for an aluminum sheet of thickness .040 with dimensions of 2ftx8ft would be 51.20 not including shipping. That would probably be more material than needed however with mistakes prone to happen it would be better to buy more than needed. Aluminum would be slightly difficult to shape with the tools that we have and a lack of any molds but definitely not impossible to work with. Overall aluminum would seem to be a very good option but there are still three other options. Next option would be that of Carbon Fiber.

Carbon Fiber is extremely strong fiber reinforced polymer that has carbon fibers (hence the name). Typically the polymer is epoxy but other polymers are sometimes used. All upper end blimps that sell online currently use Carbon Fiber because of the strength, durability and reliability. It has a high strength to weight ratio and strong rigidity. CFRP which is Carbon Fiber Reinforced Polymer has a tensile strength of 3000 MP and would be used to reinforce any molding we use and automatically make it incredibly strong. However with all the strength comes a higher price tag as expected. The thinnest carbon fiber possible for purchase is .025" and then a sheet 2'x 3' has a price tag for 186.63 dollars. Also molding the sheets into the form we need would be extremely difficult and probably not worth all the hard work in the end. The desirability to say our blimp is reinforced with carbon fiber would add intrigue however on our college budgets it is just not realistic. Next on the list would be wood.

Wood is defined as a hard, fibrous structural tissue found in trees. Wood naturally is found everywhere as approximately one trillion tons of wood is contained on Earth. Naturally, seeing as there are millions of trees that leads to a lot of different types of woods. The "strongest" wood would be considered the Teak wood which comes from the Teak Tree and is considered the strongest because of its ability to bend but not break while facing high winds. Even though it's considered the strongest it is still very easy to cut and sculpt into any form. The Teak tree must be drained of water before it can be cut down which takes 1 to 2 years. It has a high resistivity towards water and because of this fact is usually used for outdoor furniture. A sheet of 1/42" and 2'x8' runs 56.99 with free shipping. The relative ease of sculpting and strength of wood is definitely desirable for our project. Chroma Pyramidal or Balsa Tree is another type of wood that was looked into. It is native to Mexico and Brazil and is known for its extremely lightweight material. The surface is made of a cellulose lignin mix which is resistant to rot and highly durable in water. Balsa wood is commonly used for model airplanes which speak to the ease of cutting and shaping of the wood. A cubic foot of balsa typically weight less than 6 lbs. The most attractive offer about balsa wood is the price tag as a .125" thickness by 12"X48" is only 12.51 dollars which gives us the possibility of messing up numerous times while perfecting our molding for our undercarriage and camera stabilization system.

Finally PVC is the last thing that was researched for the project. PVC is usually a very rigid but lightweight material. With the addition of certain plasticizers the plastic becomes very flexible and easier to work with. A plasticizer is an additive that adds flexibility and fluidity to a material. PVC can be shaped relatively easily but the moldings for our project would be quite difficult to shape. PVC was actually more expensive than anticipated ranging 46 dollars for a .25”x 24”x48” sheet, which wouldn’t leave us with any leeway if we happened to fail on construction our carrier the first time.

Considering all of the factors above the group decided to choose Balsa wood as our material. Aluminum was very tempting considering the strength, the pliability and the overall look of metal compared to wood but working with a smaller budget tends to make a larger impact than the overall look of the blimp. By using the Balsa wood the blimp would not work any different and would be almost just as light for a fraction of the price. The less power the blimp needs to move the longer the battery life and the longer the blimp will be able to survey the area. The strengths of the Balsa wood are the flexibility, the ability to shape it, the price and the durability. Balsa wood will definitely be strong enough to support our GoPro Hero 3 Black Edition as well as any weather it would face while roaming the skies. The idea is to make the camera stabilizer an upside down parabola with a bar connecting the ends of the parabola. While the following drawing is just a basic sketch it shows how the camera will be supported on the bar and how it will fit in the middle of the semi-circle. **Figure 5.1.1** shows the camera mount system.

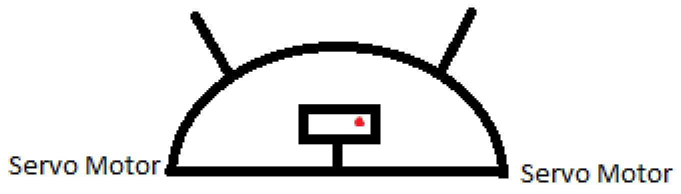


***Figure 5.1.1**: This is a zoomed in front view of how the camera will be kept in the camera stabilization system. Two Servo Motors will be able to turn the device in order to keep video still as blimp flies in the air*

The Balsa wood will be flexible enough to maintain this shape but also strong enough to crack or start becoming weaker in structure. The GoPro has a mount that will attach adhesively to the wood. The camera will the just slide into the grooves and be perfectly attached with no concern that it will slide out of the mount. The mount comes with the camera so no purchase is necessary to ensure that our camera is safe. The adhesive works in water and the camera has a water protective housing so it will work in all weather.

The stabilization part will be connected to the undercarriage part by a type of adhesive glue. Something strong enough to ensure that it will not fall and end up breaking are new GoPro Hero 3. The most complicated feature of our camera stabilization system is to actually keep the image still so the receiver gets a nice clean video feed from the camera. A shaky nonprofessional video feed will definitely take away from the overall impression of the surveillance blimp. The mount should ensure no panning but the tilting might be a slight problem. We also would like to rotate the stabilization system so that we do not have to turn the blimp in order to see what is on the left and right. To help us with this problem of turning the camera and also stabilizing the camera at the same time we will turn to Servomotors.

A Servomotor starts out with a regular motor that is powered by a voltage. Usually a servo is connected to a microcontroller and the microcontroller will tell the servo what position it should be located in. The servo has a disk with little slots in it and each slot represents a position relative to a voltage. If the microcontroller tells the servo go to position five the servo will turn to position five by way of using a Potentiometer to know the correct slow by using the voltage of the slot as a reader.

The servomotor then uses a combination of gears to create torque and rotation. The servomotor has a failsafe peg on one of the gears that does not let it spin 360 degrees, instead it spins 270 and anytime it hits that peg it can no longer rotate. The servo has a DC motor in it that spins the gears at a high revolution per minute. By turning those gears torque force is created which we can use to turn out system. Servos can be found with plastic gears which our lighter and cheaper or they can be made with metal gears which are obviously stronger, but also more expensive. We will undoubtedly go with plastic gears seeing as we want to be cost effective and lighter. There are also three different types of servos that we can possibly use: Continuous, Positional and Linear are all servos. The most common is the positional servo in which it can rotate 180 degrees and is restricted by pegs put on the gears. Continuous Servos have no restrictions like the positional ones. They can turn clockwise or counterclockwise as long as they want. They are very useful for radar satellites which would need a full range of motion. Linear servos are the least used and least popular. They use a special mechanism which will cause their force to be linear instead of rotational. We will use the most common type of servo, the positional one.

We will use two servos, one on each side of the carrier. The two servos will be connected to the Balsa wood and their power supplies will be in the undercarriage which is talked about in another section. A two axis system is what will stabilize the camera to make sure the shot stays focused and not jumpy. A digital servo drive will control the pan and tilt of the camera which will counteract any movement made by the blimp. In essence if the blimp starts heading left the carrier will counteract by going right to ensure a still shot. Servomotors are relatively cheap and effective and we have a few different ones to choose from. When choosing we need to consider a few things about the servo. We need to know how fast the servo will rotate, how much voltage it consumes, how hard it needs to push or pull, and will you have overshoot.

After comparing several of them we decided to go with the Parallax Servo Motor. It can hold any rotation between 0-180 degrees. It has a 38oz-in torque at 6 VDC. It has a current draw of 140 mA and only 15 mA in static state. The Servo motor is very susceptible to spikes in current which will fry the servo and make it virtually useless. All of our Servos will be connected with a voltage regulator and given the same voltage in order to minimize these spikes. The following picture is a quick and easy to read diagram of the connection of the servo to the microcontroller. **Figure 5.1.2** shows the connection of the servo to the microcontroller.

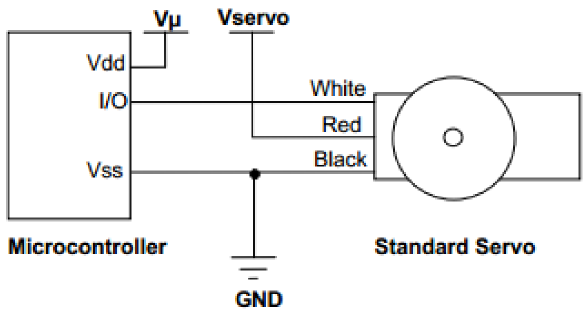


Figure 5.1.2*: How to connect the Servo motor to the Microcontroller.* Reprinted with Permission by *Parallax*

The white wire represents the signal wire; it runs as an input wire with a voltage of 3.3volts. The red wire connects to the voltage of the servo with a voltage regulation of 4.0 volts. The black wire would obviously go to the ground. We also have a picture that shows the sample timing diagram for the servo. If the servo isn't receiving a pulse every 20 mms it will not be able to hold its position. **Figure 4.1.3** shows the PWM signal timing we have to take into consideration when controlling the servo to make a smooth transition.

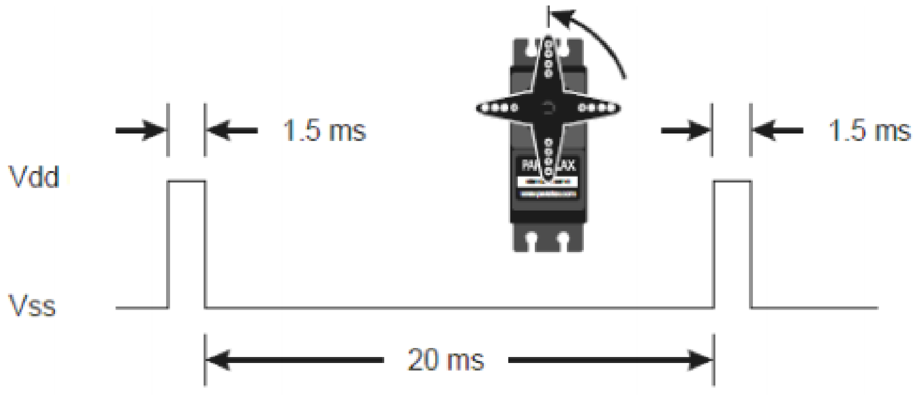


Figure 5.1.3: If the PMW does not cycle every 20 MS the Servo will lose power Reprinted with Permission by Parallax

A slip ring and slipper clutches will be used as a safety mechanism to ensure that none of our gears become stripped and no wires become tangled. A transmitter will be attached to the bottom of the carrier in order to ensure direct access with the remote control, or Android controller in our case. Two different power systems should be installed and be carried by the blimp, one which controls the servo motors and then one for the transceiver. The servomotors should be able to be screwed into the Balsa wood very easily and allow for easy turning if the PCB required it.

Ideally we would have safety precautions to ensure the servomotors never turned too far which could destroy the structure of the Balsa wood so we would use slip rings to ensure not too much damaged could possibly happen. By installing the slip rings the camera stabilizer would be able to turn 360 degrees as well if we decided to break off the safety peg that keeps the servomotors from turning fully. This slip ring also helps with keeping wires from tangling and possible splitting. All power systems for these devices will be located right about in the undercarriage of the blimp and thus why the protection of the wires needs to be at the foremost of concern. Slipper clutches will keep the camera from being knocked off balance in case something unexpected hits it. The slipper clutches will be put on the sides of the camera so that if it starts to learn forward to far it will be caught and brought back to the correct position. Considering that it will be on a mount it shouldn’t be a problem but with an expensive camera every precaution is a good one.

Section 5.1.2 The Mount

When referring to the heart of the blimp you could refer to the motors. The motors keep the blimp in the air along with the helium and basically pump the blimp through the air. If we are comparing parts to the human body and the mount would be the head and inside would be the brains. All of our important hardware that runs the software will be located here. This will be the place of the PCB board, the antenna, the GPS module, the gyroscope, the accelerometer, the batteries that will run everything and the transceiver. The majority of the high cost items of our blimp will be contained inside our mount and thus this needs to be the strongest, most durable and safest section of our blimp. If a tragedy happens and our blimp gets shot down, we need to make sure that this section remains intact and unfazed. In other words we need to spend the most money on the mount to ensure that everything inside is safe.

The mount will be fastened to the bottom of the blimp by using an epoxy or glue to ensure utmost connection between the polyester of the blimp and the material of the mount. The material of the mount we will discuss in the next paragraph. The mount will also have the camera stabilizer attached to it so the glue holding the top of the mount needs to be extremely strong to hold the mount, the stuff inside the mount and the camera. We will try to keep the mount as small as possible seeing as weight is very important to our design. If the blimp is approximately 10X3X3 which we expect it will be then it will have a lift capacity of 4.8 lbs. based on the fact that helium has a lift of .067 lbs./ft^3. So, when considering the materials for our mount we need to remember the lighter the better, but we also have to keep in mind that the MASS camera system needs to be very secure.

We have a few different materials to choose from that I think would suffice in filling these two requirements. First and what immediately comes to mind is carbon fiber which is an extremely lightweight and durable material which has a price tag that matches its effectiveness. We could also use the same material which we used for the camera stabilizer which would be balsa wood. Balsa wood is a good option as well that is also very light and durable but with a less steep price tag. Kevlar would definitely work in terms of keeping our PCB board and all of its components safe, especially if someone started shooting at our blimp. I would do a quick explanation of all three materials before we decide which material is the best for our group to use.

Carbon fiber as was briefly touched on earlier is a reinforced polymer that has carbon fibers to give it extra strength. Carbon fiber reinforced polymers or simply CFRP is normally used in the automotive industry because of its strength and rigidness. Carbon fiber can also be readily molded which is something that would be very important for us and the specific shape we need. Making the molding on the other hand might be extremely difficult considering the tools we have and we might have to buy extra tolls which would have to go into the cost of the carbon fiber in general. Carbon fiber is also very desirable because of how many other blimps use carbon fiber, typically when every other blimp that has a camera attached to it use a carbon fiber mount and this is your first go around it is always best to look at what everyone else did to get the best idea of how to do yours. I know cost has been the main determinant for every other part but for this one the stuff inside the mount is definitely worth spending a little extra. The sheeting will be the most expensive for all three cases but we will get into price later.

The balsa wood which is what we will be using for the camera stabilization system would also be a good option for the mount. As previously stated it is lightweight, cheap, and pliable and we can save money by ordering extra from the stabilization system. A foot only weighs 6 lbs. which will hopefully use a little less because our blimp will only be able to hold approximately 5 lbs. That would be a real downer to finish constructing all of our stuff and realize our blimp cannot fly without an extra propeller propelling upwards. This however is the backup plan in case we are not able to make it weightless.

Finally is the last option would be to use Kevlar material which I haven't mentioned prior in any other section. Kevlar is a Para-aramid synthetic fiber which has been used in various facets ranging from bicycle tires to bullet proof vests. Kevlar has a tensile strength of 3,620 MP and a relative density of 1.44

Seeing as Kevlar is used for bulletproof material it would be more than strong enough to keep our hardware safe and secure. Kevlar also will not mess with interference the way that most metals do, including carbon fiber which will slightly interfere with signal transmissions. It typically runs a price range of about 3.50 a square foot which is actually quite affordable. The molding of the Kevlar would be quite difficult and as for carbon fiber would have to be considered when purchasing the Kevlar. Overall Kevlar seems to be the best option for security and safety with a price that is very affordable.

After all the considerations and debating we choose to continue with the use of the balsa wood. It just made a lot of sense seeing as we had already chosen to use the balsa wood as our camera mount. We can order in bulk and then cut our pieces accordingly and use them for two different aspects of our blimp. The tools we need would remain the same so we wouldn't to buy different moldings for the carbon fiber or Kevlar material. The balsa wood in all honesty is probably not the best choice, but it is the safest choice and considering we just need to be able to get this blimp off the ground that became a major argument in favor of using balsa wood.

Section 5.2 *First Subsystem (PCB/Control System)*

In this section we will go over each module attach to the blimps' microcontroller. We plan for the microcontroller to be mounted on a Printed Circuit Board which is currently under construction in Eagle Schematic Capture Software and PCB Layout.

Section 5.2.1 *GPS Module*

GPS or global positioning system is a way of using satellites to relay where something is on Earth. GPS modules are great for tracking, positioning, and timing. Upon picking GPS module there are numerous things to consider which are: Size, Update Rate, Power Requirements, Number of Channels, and Antenna Size. Size is a very simple issue seeing as every cell phone has a GPS in it and cellphones have become extremely small so the size and weight will not be a problem for our project. In terms of update rate it basically tells you how fast the GPS recalculates where it is located. Standard is approximately 1 Hz or one time per second. The higher the Hz reading the more information will be flowing into the microprocessor which could cause it to burn out. GPS use a large amount of power when relaying their position so battery life and power saver modes will be at a premium when comparing different modules. The number of channels mainly deals with when you first turn on your GPS how long does it take to configure where it is. Average time seems to be around 30 seconds so anything less than that would be good for our project. All antennas are basically the same, made out of ceramic, there are other options but they tend to be more expensive and not justifiable for purchase when dealing with our project. We compared four different GPS modules and will chart and explain why we choose the one we did. **Table 5.2.1** shows the comparisons of the GPS modules that we are considering and would fit our needs for the project.

GPS MODULE	237-M10382-A1	927-A2100-B	916-UC530	916-IT430
Maker	Antenova	Maestro Wireless	Fastrax	Fastrax
Frequency Band	1.575 GHz	1575 MHz	32768 Hz	1.575 GHz
Interface	SPI, UART, USB	UART	UART	UART, I2C, SPI
Number of Channels	50	48	66	48
Acquisition Sensitivity	-146 dBm	-163 dBm	-165 dBm	-147 dBm
Horizontal Position Accuracy	2.5 m	2.5 m	3 m	2.5 m
Maximum Operating Temperature	85 C	85 C	85 C	85 C
Minimum Operating Temperature	-40 C	-40 C	-40 C	-40 C
Operating Supply Current	40 mA	30 mA	25 mA	47 mA
Operating Supply Voltage	1.8V, 3.3 V	1.8V	3-4.3 V	1.8 V
Cold Start	37 s	35 s	31 s	35 s
Price	$29.89	$17.57	$34.95	$31.35

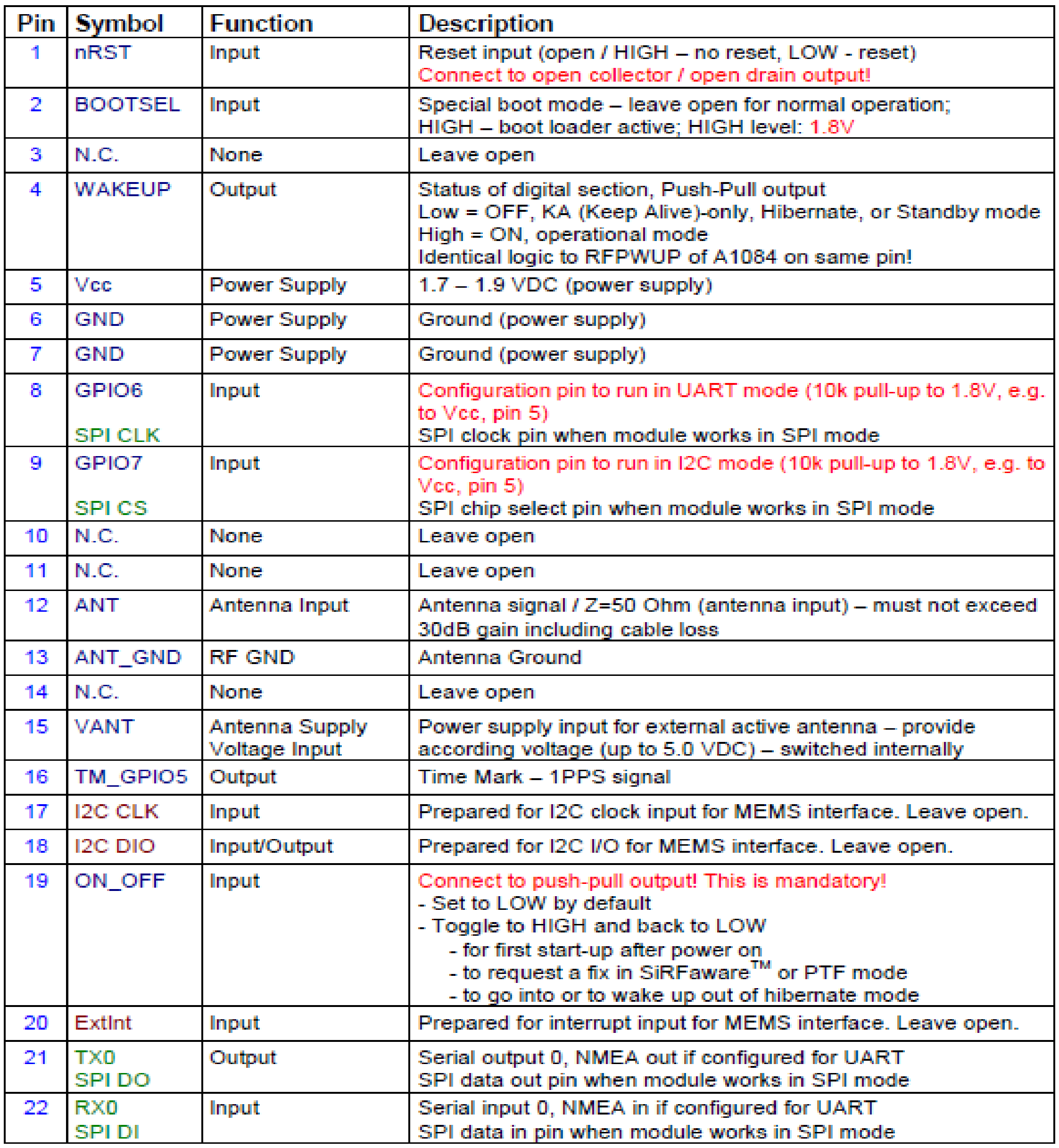
***Table 5.2.1**: Comparisons of different GPS modules*

We wanted to stay along the same path of finding a good part that fits our needs and wasn't very expensive. Unfortunately the GPS modules tended to be around the 30 dollar mark which was a little higher then we allotted in the budget, but sometimes things don't come out perfectly. We needed something with good sensitivity, good horizontal positioning and a good read time. The sensitivity would not be a problem seeing as every module we looked at ran into -150 dBm range which would be more than enough. The horizontal sensor was great as well within 3 meters would do more than suffice seeing as the module will be in the middle of the blimp and the blimp will extend approximately three feet either way in both directions. The read time was the only one that varied and this was actually one of the few times that a smaller Hz rating or in other terms a slower reader was better. We discovered that a GPS that read too often sent too much data to the microprocessor and thus would waste battery or cause it to overload.

Considering that 1 Hz equals 1 second of send time and we are dealing with a large slow blimp nothing considerably larger would be needed. The sensors with the 1.575 GHz readings are great for large and fast microprocessors that can handle that amount of data but considering all of the previous things mentioned we cut those modules from the list. That left two possible options for us. It was between the Maestro Module and the Fastrax one. Both had adequate specifications for us but one thing that stood out was the voltage needed, the Maestro only needing 1.8 Volts compared to the Fastrax needing 3.3 V ended up being the main reason we choose the 927-A2100-B model by Maestro Wireless. The A2100-B model is the first model made by Maestro using CSR's SiFRstarIV chip on GPS modules. It is a fast, highly responsive GPS that has accuracy down to -163 dBm.

It has access to 48 channels meaning 48 different satellites and requires only minimal amps to begin for a cold start (50 micro amps). The 48 different channels are done through parallel tracking. It has a jammer blocker that removes in=bad jammers of 80dB/Hz. As previously stated it uses the UART interface which our chosen microcontroller has special pins that read this interface. It has direct antenna support. The GPS correlates are approximately 400,000.

The frequency is 1,757 MHz which will be more than enough to support our blimp when it's traveling through the atmosphere at light speeds. The times for first fix are roughly 35 seconds but after the first reading every reading after that takes less than one second. The calibrated clock will never stop and is accurate to within microseconds. The receiver has date, time, and position along with a valid almanac. The only negative about this module is that it doesn't come with an antenna which will inevitably be the next section. The following chart will represent what each pin is used for. **Table 5.2.2** shows the description of each pin and if they are an input or output.



Pin	Symbol	Function	Description
1	nRST	Input	Reset input (open / HIGH – no reset, LOW - reset) Connect to open collector / open drain output!
2	BOOTSEL	Input	Special boot mode – leave open for normal operation; HIGH – boot loader active; HIGH level: 1.8V
3	N.C.	None	Leave open

4	WAKEUP	Output	Status of digital section, Push-Pull output
			Low = OFF, KA (Keep Alive)-only, Hibernate, or Standby mode
			High = ON, operational mode
			Identical logic to RFPWUP of A1084 on same pin!

5	Vcc	Power Supply	1.7 – 1.9 VDC (power supply)
6	GND	Power Supply	Ground (power supply)
7	GND	Power Supply	Ground (power supply)
8	GPIO6 SPI CLK	Input	Configuration pin to run in UART mode (10k pull-up to 1.8V, e.g. to Vcc, pin 5) SPI clock pin when module works in SPI mode
9	GPIO7 SPI CS	Input	Configuration pin to run in I2C mode (10k pull-up to 1.8V, e.g. to Vcc, pin 5) SPI chip select pin when module works in SPI mode
10	N.C.	None	Leave open
11	N.C.	None	Leave open
12	ANT	Antenna Input	Antenna signal / Z=50 Ohm (antenna input) – must not exceed 30dB gain including cable loss
13	ANT_GND	RF GND	Antenna Ground
14	N.C.	None	Leave open
15	VANT	Antenna Supply Voltage Input	Power supply input for external active antenna – provide according voltage (up to 5.0 VDC) – switched internally
16	TM_GPIO5	Output	Time Mark – 1PPS signal
17	I2C CLK	Input	Prepared for I2C clock input for MEMS interface. Leave open.
18	I2C DIO	Input/Output	Prepared for I2C I/O for MEMS interface. Leave open.
19	ON_OFF	Input	Connect to push-pull output! This is mandatory! - Set to LOW by default - Toggle to HIGH and back to LOW - for first start-up after power on - to request a fix in SiRFaware^{TM} or PTF mode - to go into or to wake up out of hibernate mode
20	ExtInt	Input	Prepared for interrupt input for MEMS interface. Leave open.
21	TX0 SPI DO	Output	Serial output 0, NMEA out if configured for UART SPI data out pin when module works in SPI mode
22	RX0 SPI DI	Input	Serial input 0, NMEA in if configured for UART SPI data in pin when module works in SPI mode

***Table 5.2.2**: Explanation of all the pins and what they represent. Reprinted with Permission by Maestro Wireless*

Then we have the following picture which displays where the pins are located on the chip and brings more clarity to the table we posted above. In the following picture the green pins represent the host interface pins, the red pins represent the MEMS interface pins and the blue pins represent the other stuff. The numerous pins located in the center are for shock and vibration stoppage purposes. **Figure 5.2.1** below outlines the purpose for every pin and what it entails.

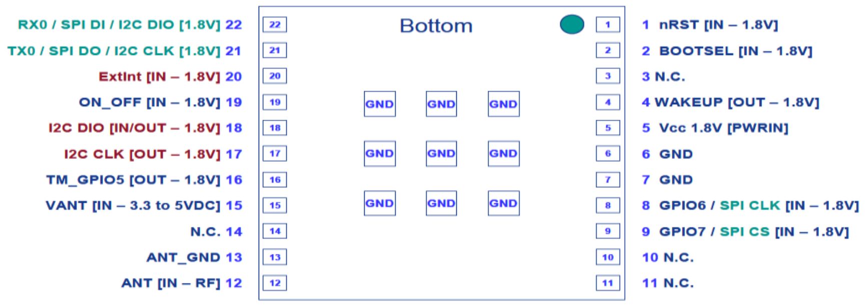


***Figure 5.2.1**: Pin diagram of GPS Module 927-A2100-B*
Reprinted with Permission by Maestro Wireless

The chart is very user friendly and very easy to read which is good for college students doing our first real soldering. When referring to special features and helpful modes the GPS module has multiple special features. It starts with a special firmware called SiRFaware which is a low power operating that is maintained throughout the duration of the flight in order to keep the power level low but still maintain certainty in position, time and frequency. There is also a “Push to Fix” mode which does infrequent checks on position for up to two hours which will be good for us to maintain that the blimp has not left the designated zone when we leave it unattended and will not waste the battery. There is also a hibernate mode which will as expected cause the chip to sleep and can be woken up by the toggling of the ON_OFF pin.

Even with all of these features our GPS will be of no use if we don’t have a successively mounted antenna and a good antenna at that. The ANT pin is used to connect the antenna to the receiver. When connecting we have to use Radio Frequency rules. We have to use a 50 Ohm PCB strip. The manual reminds us to keep the strip as small as possible to ensure that the maximum efficiency is experienced. Also we have to remember to ground the antenna into the ground of the PCB board. **Figure 5.2.2** below, we are shown how to mount the antenna to our pin and the correct place to attach it.

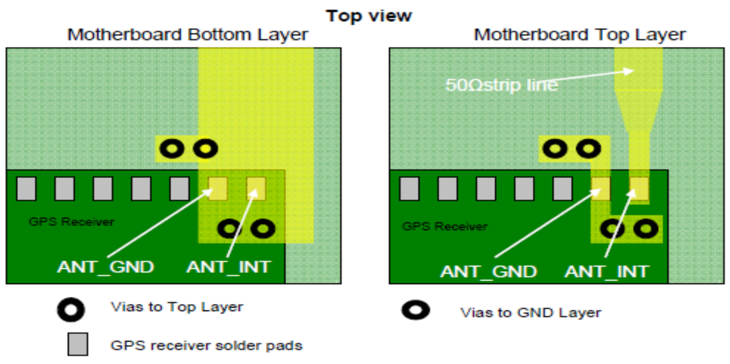


***Figure 5.2.2**: How to connect the antenna into the GPS Module*
Reprinted with Permission by Maestro Wireless

One rule to remember is that a calculation is needed to reach an impedance of 50 ohms. The width of the antenna should be approximately 1.8 times the height of the PCB board in order to keep positive reception without the antenna sticking out too far. I will get more in-depth with the antennas and the qualifications needed in order to make our GPS run to its fullest capabilities in the next section. Finally the last thing when worrying about the GPS is the actual interfacing and testing of the GPS. All boards have been tested numerous times before we get them and interfacing is done by UART which everyone in our group has used numerous times in Embedded Systems lab with Professor Weeks.

Section 5.3 Second Subsystem (Camera System)

The camera system consists of the modules that control the stabilization of the Camera Mount. The commands to send out the stabilization is calculated on the microcontroller and the moves the servos on the stabilization specific amounts related to the gyroscope and accelerometer readings to get it to stop it from shaking and imitate a steady camera feed.

Section 5.3.1 The Gyroscope

A gyroscope is used to measure the rotation around an axis, also known as the angular velocity. Gyroscopes are not affected by gravity and can monitor an object in motion in reference to its original orientation. It holds units in revolutions per minute and can represent all three axes commonly referred to roll, pitch and yaw. A gyroscope is a spinning wheel in which the axle is free to rotate in any orientation. There are four main components of the gyroscope including the gimbal, rotor, frame and spin axis. With this system you can tell how the object is moving rotationally compared to the platform and this mixed with the accelerometers from the previous section will be able to you exactly where you are heading. Our blimp will use this technology to ensure it remains on course and doesn’t hit any obstacles that would cause substantial damage.

For our gyroscope we have a certain amount of needs and concerns that need to be addressed. The most important things that we have to consider when choosing a gyroscope are the following: Range, Interface, Number of axes, and Power usage. When it comes to range we are talking about the range of values the gyroscope can read. We cannot get one that does not have a large range or one that is too sensitive. Interface is next on the list and although a majority of gyroscopes are analog we will try to find the right one that is digital. Prefer to keep it along the same lines as the accelerometer and use SPI programming if possible.

The number of axes measured would have to be three as we will be monitoring our blimp in all three dimensions. Power usage is important when considering the battery and how long do we want the blimp to run for. Obviously any gyroscope with a special sleep mode or power saver mode would be a premium option. We wanted to have a good amount of different gyroscopes to check out the different specifications and then we narrowed it down to our favorite four which will be depicted in the following **Table 5.3.1**.

Gyroscope	511-L3G4200D	20T511-L3GDR	700MAX21000+	511-L3G4IS
Manufacturer	STS	STS	Maxim Integrated	STS
Sensitivity	70mdsp/digit	17.5 mdps/digit	960 dsp/digit	8.75 mpds/digit
Supply Current	6.1mA	6.1mA	100mA	6mA
Digital Output	16bits	16bits	16bits	16bits
#of Axes	3	3	3	3
Max Op Temp	85 C	85 C	85 C	85 C
Supply Voltage	3.6v	3.6 V	3.6 V	3.6 V
Supply Voltage Min	2.4V	2.4V	2.4V	2.4V
Non-Linearity	.2%	.2 %	.2%	.2%
Range	+/- 2000 dps	+/- 2000 dps	+/-2000 dps	+/-2000 dps
Acceleration	None specified	1,000 g	None specified	10,000g
Bus Interface	I2C, SPI	I2C, SPI	I2C	I2c, SPI
Bandwidth	50 Hz	95 Hz	400 Hz	140 Hz
Price	$11.61	$6.76	$5.10	$6.45
Size	4X4X1(mm)	4X4X1(mm)	3X3X.9(mm)	4X4X1(mm)
Output Lines	2	2	2	2
Power Saver/Sleep Mode	Yes	Yes	Yes	Yes

***Table 5.3.1**: Compare differences between possible gyroscopes*

As can be seen from the above table most of the values for our gyroscopes are very similar. Almost all have the same range for non-linearity, bandwidth, price and voltages. Sensitivity for all of them is very close numerically as well. Because we choose an STS accelerometer I wanted to keep everything as simple as possible and realize the communication coming from the same company would probably be better than two coming from separate entities. So that leaves three of our gyroscopes still to choose from. Seeing as the only real separation between all of our gyroscopes was the sensitivity I decided to go with the most sensitive one which would be the 511-LG34200D which is made by STMicroelectronics. Now we will go into detail of what this particular gyroscope has and all the special features that made it the optimum choice.

This STMicroelectronics product has numerous features that made it desirable for our project. First it has three operating modes ranging from 250dps/digit to 2000 dps/digit. It runs on low power while having a three axis angular sensor that maintains stability of zero rate level. It also measures sensitivity when compared with time and temperature. We have multiple bandwidths we can use that I will talk about later when we go into interfacing and communication with microprocessor. As noted earlier it has the same manufacturer as the accelerometer and thus the same interface; which will be the I2C interface. This interface works mainly like C programming in the wording and the structure in the way that it appears and the use of variables. The temperature range is more than sufficient enough to ensure that we do not destroy our gyroscope due to overheating or freezing. In **Figure 5.3.1** below, you will be able to decipher how the gyroscope is made and all of the components inside.

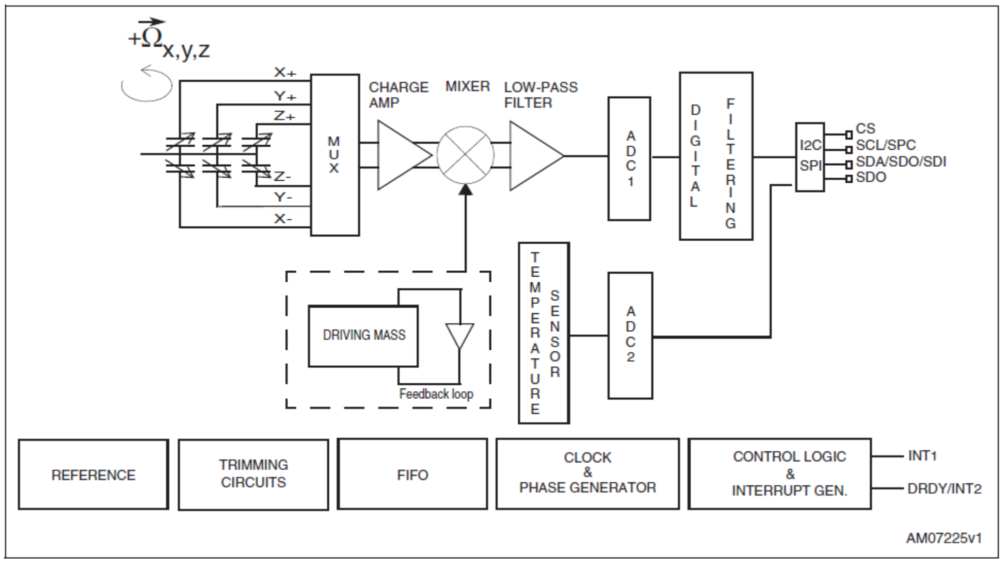


***Figure 5.3.1**: Hardware inside gyroscope, shows emphasis on feedback loop Reprinted with Permission by STMicroelectronics*

This picture represents the block diagram of what is actually contained in our gyroscope. The feedback loop controls the vibrations within the drive circuitry, more specifically in the driving mass portion of the circuit. The sensing signal will be filtered and then produced as a digital output. Also to help explain how our gyroscope will work I will also include the pin diagram in the **Figure 5.3.2**.

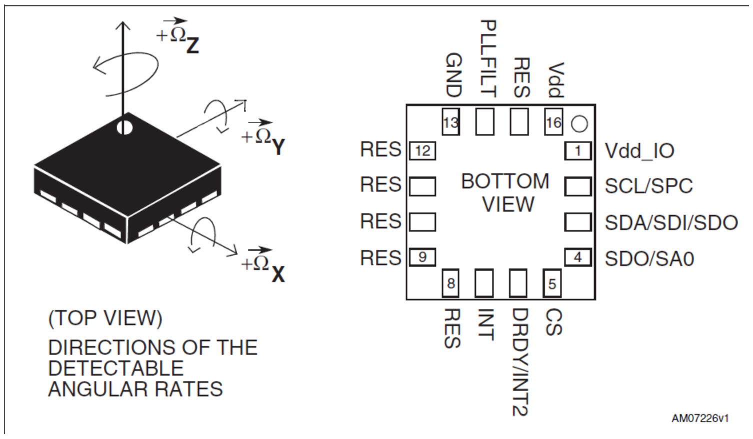
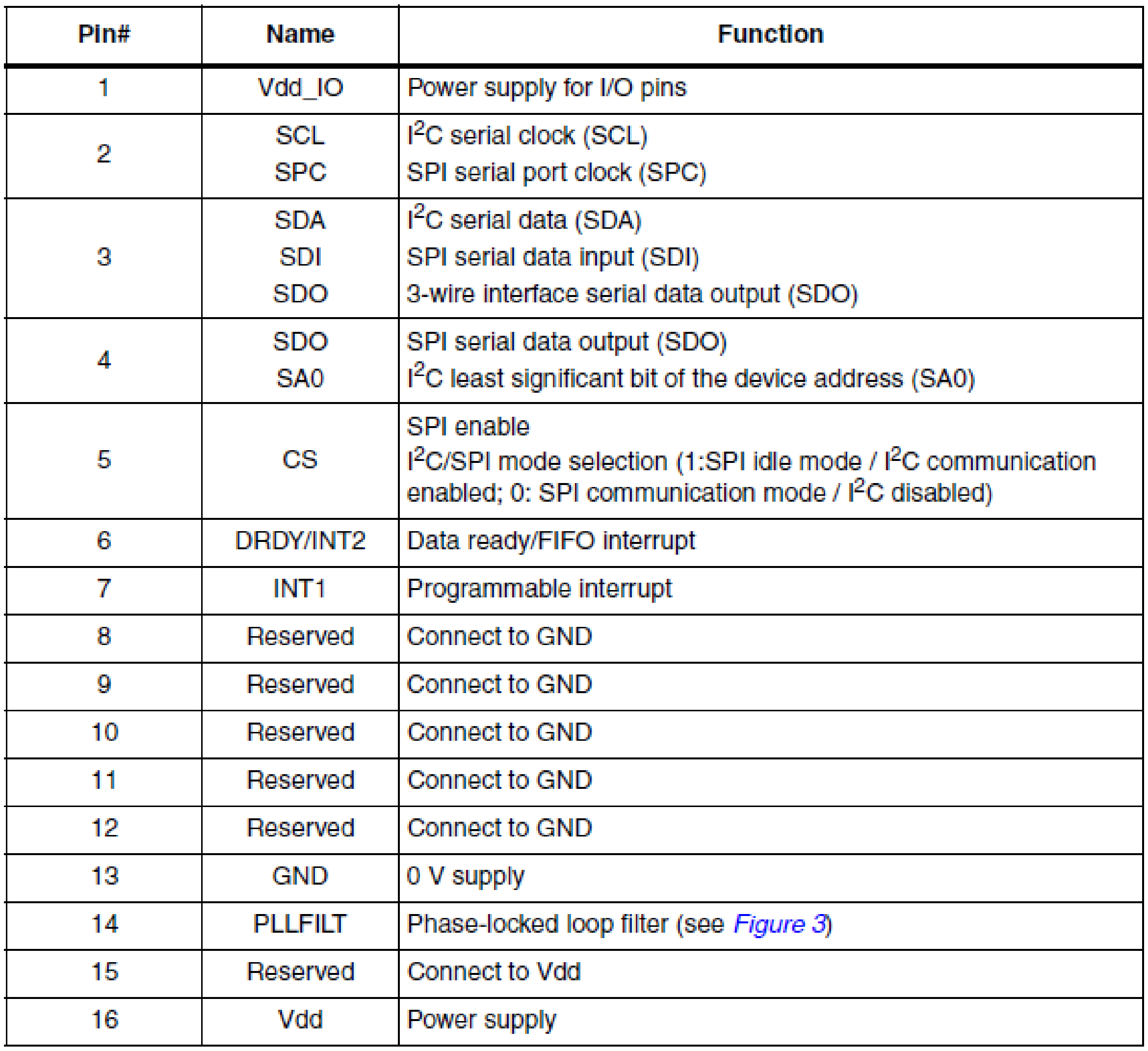


Figure 5.3.2*: Pin connections for 511-L3G4200D Gyroscope. Reprinted with Permission by STMicroelectronics*

The gyroscope has 16 pins which can be counted from the previous picture. It has a power supply in the last pin powered by the Vdd. The picture also shows how the Gyroscope will have full turning radius on all three dimensions, while the first pin is the power needed for the input output lines to work. In ***Table 5.3.2*** it shows how a different pin will be responsible for its own axis



Pin#	Name	Function
1	Vdd_IO	Power supply for I/O pins
2	SCL SPC	I^2C serial clock (SCL) SPI serial port clock (SPC)
3	SDA SDI SDO	I^2C serial data (SDA) SPI serial data input (SDI) 3-wire interface serial data output (SDO)
4	SDO SA0	SPI serial data output (SDO) I^2C least significant bit of the device address (SA0)
5	CS	SPI enable I^2C/SPI mode selection (1:SPI idle mode / I^2C communication enabled; 0: SPI communication mode / I^2C disabled)
6	DRDY/INT2	Data ready/FIFO interrupt
7	INT1	Programmable interrupt
8	Reserved	Connect to GND
9	Reserved	Connect to GND
10	Reserved	Connect to GND
11	Reserved	Connect to GND
12	Reserved	Connect to GND
13	GND	0 V supply
14	PLLFILT	Phase-locked loop filter (see *Figure 3*)
15	Reserved	Connect to Vdd
16	Vdd	Power supply

***Table 5.3.2**: Pin connection description of what every pin represents. Reprinted with Permission by STMicroelectronics.*

When hooking up the gyroscope to our development board we have a few items to discuss in order to figure out what settings will be the most ideal and accurate for our blimp. We choose the sensitivity of the blimp which ranges from 250 DPS with an 8.75 MDPS/digit, 500 DPS with a 17.5 MDPS/digit or a 2000 DPS with a 35 MDPS/digit. Seeing as most of these sensitivities are very accurate and normally used for higher force projects we will use the lowest selectable option which will be the 250 DPS option. The digital zero rate level is directly correlated to the setting that we choose so since we have the 250 option we have the smallest possible range for +/- 10 DPS. The sensitivity is described the gain of the sensor and can be determined by applying an angular velocity and this value is very resistant to temperature and time. The zero rate level as mentioned earlier is when there is no angular rate present. These values are slightly susceptible to positioning on the printer circuit board or mechanical stress but still maintain its original value quite well. The gyroscope has a memory allocation of FIFO when it comes to measure the three dimensions of roll, yaw, and pitch. By using the FIFO the gyroscope is able to go into power save mode and thus keep our battery life a little higher. The gyroscope can do this by going into sleep mode and only "waking up" when it needs to send new data to the processor. The FIFO feature has five different setting capabilities: Bypass Mode, FIFO Mode, Stream Mode, Bypass to Stream Mode and Stream to FIFO Mode. In bypass mode the FIFO is not used and old data is written over. In FIFO mode new data from the yaw, pitch, and roll are saved in the FIFO. The FIFO continues collecting data until the memory allocation is full and then the mode needs to be changed to bypass mode in order to take in new information. Stream mode uses the FIFO feature and just replaces the oldest data with the newest data when the memory is full. The bypass to stream mode makes the FIFO start operating in the bypass way and needs a trigger to be set in order to move onto the streaming way of collecting data. For the final mode the FIFO saves all of the information until a trigger is set and then it starts deleting the older data. We will most likely use the second option so we can see for our whole flight how the blimp's values change.

For the digital interface the gyroscope uses the same interface as the accelerometer which is partially one of the reasons we choose it. We can use either the SPI or I2C interface but seeing as we are more familiar with the SPI interface this is the one we will be using. It uses a master and slave system in which the master has a set of seven bits and if the data code matches these 7 bits then the master will control the "slave" bits. The SPI reads very comparably to the C programming in terms of language and readability and seeing as everyone in our group has experience with c programming it seems like a natural fit. Both our accelerometer and our gyroscope will be attached to our printed circuit board which will be installed in the mount of our blimp, which will hold the most important part of our blimp, all of the hardware which will control the software. One other piece of hardware that will be installed inside the mount and attached to the printer circuit board will be the GPS.

Section 5.3.2 Accelerometer

In order to keep our blimp level and upright we have to add a few features that ensure the blimp is working at full potential. We will have to incorporate rotational and motion sensors to keep our blimp from turning over or from doing another undesired trait. We want to be able to control positioning, velocity and orientation and by using two different sensors we should have this aspect of the blimp covered. I will talk about the rotational sensor later which would be a gyroscope. For now the motion of the blimp is read by an accelerometer. Basically, the accelerometer will work with an inertial frame that is equal to gravity. Meaning that when activated the accelerometer will automatically read the 9.8m/s^2 of force from gravity downward and will take it out of the calculation to determine the position and orientation of our aircraft. Overall our accelerometer is expected to perform a couple of functions namely: tilt, roll, vibration, impact and feedback for active control systems.

An accelerometer works by using proper acceleration. Proper acceleration is the actual physical acceleration an object has forced upon it. In essence it is acceleration that ignores free fall from gravity however it will acknowledge drift and drag. Another way of explaining it is the accelerometer works in terms of frame of reference. If the observer and accelerometer both feel the same force than to the observer the accelerometer or the device that had the accelerometer in it would not be moving and proper acceleration relatively works this same way. For commercial use which would be slightly extravagant for our project but effective, there are three main accelerometers used. Piezo-electric, Piezo-resistive, and Capacitive are the types of accelerometers and these will change a mechanical motion into an electric signal. A piezoelectric accelerometer works by having crystal structures that become stressed by accelerative forces which naturally causes voltages and this can be read. A capacitive accelerometer would have two small conductive surfaces which will naturally have a capacitance between them and when one of the structures moves it will create a change in capacitance which can be lead to find the voltage.

When choosing an accelerometer, there were a couple of items to consider. The first being do we need digital or analog outputs which will depend on the hardware we use. Since we are using a microprocessor with digital inputs we will definitely get an accelerometer that is digital. We need to decide whether we want three axes for all three dimensions or only two. We can buy special accelerometers or we can just put two regular ones at a 90 degree angle from each other. How strong of an accelerometer do we want meaning do we want one that will measure up to 2gs or 4gs or possibly more. Considering that we will be working with a blimp and only slight tilts and turns a 2g accelerometer should be more than strong enough to measure all the acceleration our blimp will feel. Sensitivity, bandwidth and impedance are other important issues that must be dealt with as well. Obviously the better the sensitivity the more correct our accelerometer will work and more accurate it will be as well. Sensitivity is measured in the change of the voltage out divided by the changed in g-force. Bandwidth typically ranges from 50 Hertz to several hundred depending on the size of the project and once the lower end will be sufficient for a project like ours.

Another thing to consider is the orientation of the accelerometer in respect to our blimp. There are three different ways to orient the accelerometer in response to the g effect. Base connector being horizontal represents a 0g force, having it facing downwards will give it a -1 g force and then facing upwards will give it a +1g force. It is mathematically more intelligent to have the base connector be positioned in the vertical direction or the 1g direction because of the ease of having a mistake. While in the horizontal position the Gn = g*sin(theta) while in the vertical position or +/-1 g direction Gn = g*cos(theta) which in English means that for a 1 degree change from level positioning the horizontal base connector will give a 57X larger incorrect reading than a vertical based connector that was knocked 1 degree off alignment.

Our accelerometer will be attached to our Printer Circuit Board and thus be directly connected with the microcontroller. The printer circuit board being the most important part of our project will be encapsulated in our balsa wood mount and hopefully remain securely stored there seeing as any force will throw off calculations of the accelerometer and make it less effective. The mount will be attached to the bottom of the blimp and will feel the least amount of force of any part of the blimp. Now we will talk about the most important part of the section and that is the choosing of the actual accelerometer. We were looking at a few different types accelerometers and which one would fulfill our needs the most accurately. Here in ***Table 5.3.3*** displays the different ones we looked at and the specifications for each one.

Accelerometer	LIS3DH (ST)	MMA8453Q (Free scale Semiconductor)	LIS331DLH (ST)
# of Axises	3(X,Y,Z)	3(X,Y,Z)	3(X,Y,Z)
Acceleration(g)	1000	2,4,6,8	2,4,8
Sensitivity	0.06 mg/digit to 0.73 mg/digit	256 count/g, 512 count/g, 1024 count/g	.9 mg/digit to 4.3 mg/digit
Digital Output	I2C, SPI	I2C	I2C, SPI
Max Temp	+ 85 C	+ 85 C	+ 85 C
Min Temp	-40 C	-40 C	-40 C
Supply Voltage Max	3.6 V	3.6 V	3.6 V
Supply Voltage Min	1.71 V	1.95 V	2.16 V
Price	3.25	1.31	2.83
Output Pins	16 bit	8 bit	16 bit
Size	3X3X1 mm	3X3X1 mm	3X3X1 mm

Table 5.3.3*: Comparison of all potential accelerometers*

As the previous table noted most of the qualifications are the same for the three accelerometers. All of them are tri-axial which is extremely important seeing as we have a flying object so we would need information on all three dimensions. The LIS3DH is a little too powerful for our project however seeing as it has a rating of up to 1000 g force acceleration. If a top dragster race car has a horizontal for of 5.7 g force we probably won't need more than 2 g force in our blimp. The g force is a measurement of weight/-mass. The interfaces are all the same as well using the SPI. All have similar maximum and minimum temperature and considering most of the flying will be indoors this is unimportant. The supply voltage is relatively the same on all of them as well. All of them have the same size and relatively the same weight so all in all the MMa8543Q and the LIS331DLH would both work adequately for our project. We are going to with the MMA8543Q mainly because it is cheaper. We will buy three of them in anticipation of blowing them up, dropping them in water and any other form of human error. The following is the specifications for the MMA 8543Q. And the pin view of how we will be connecting it. I have also accompanied **Table 5.3.3** that lists what all of the pin connections represent.

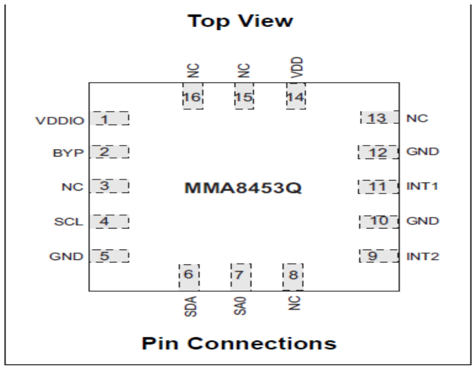


***Figure 5.3.3**: Pin connection for MMA8543Q accelerometer*
Reprinted with Permission by Free-Scale Semiconductor

Figure 5.3.4 below shows the application of the MMA8453Q accelerometer. The MMA8453Q is powered through the VDD line. For power supply there are decoupling capacitors (100 nF ceramic plus a single 4.7 µF ceramic) and for the best possible results the power supply needs to be as close to the pins 1 and 14 of the MMA8453Q. The control signals SCL, SDA, and SA0 cannot obtain a voltage of VDDIO+.3V. If the VDDIO is disconnected, the control signals that were listed previously will hold any logic signals with their internal electrostatic discharge protection diodes. The threshold and the timing will be controlled by the use of two pins (INT1 and INT2) and both of which are user programmable through the SPI interface.

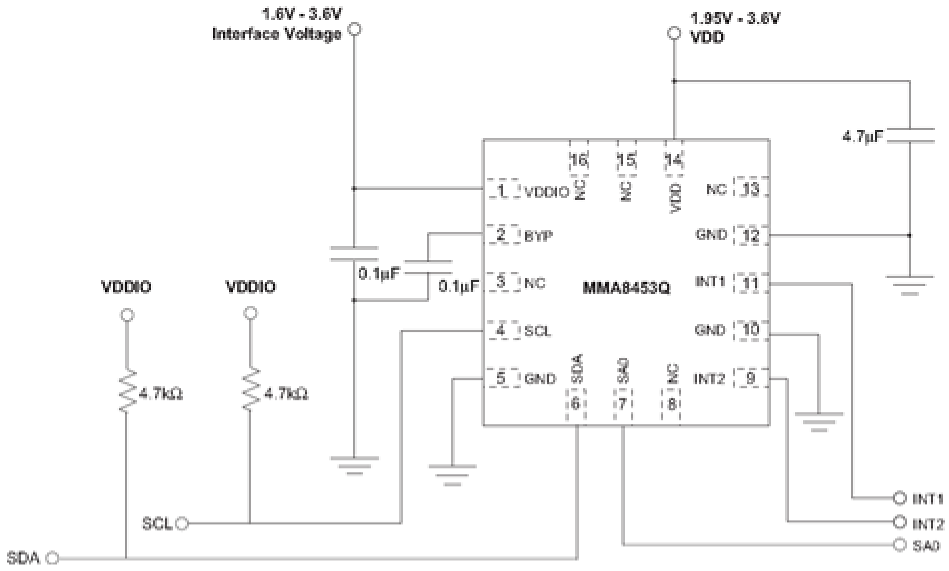
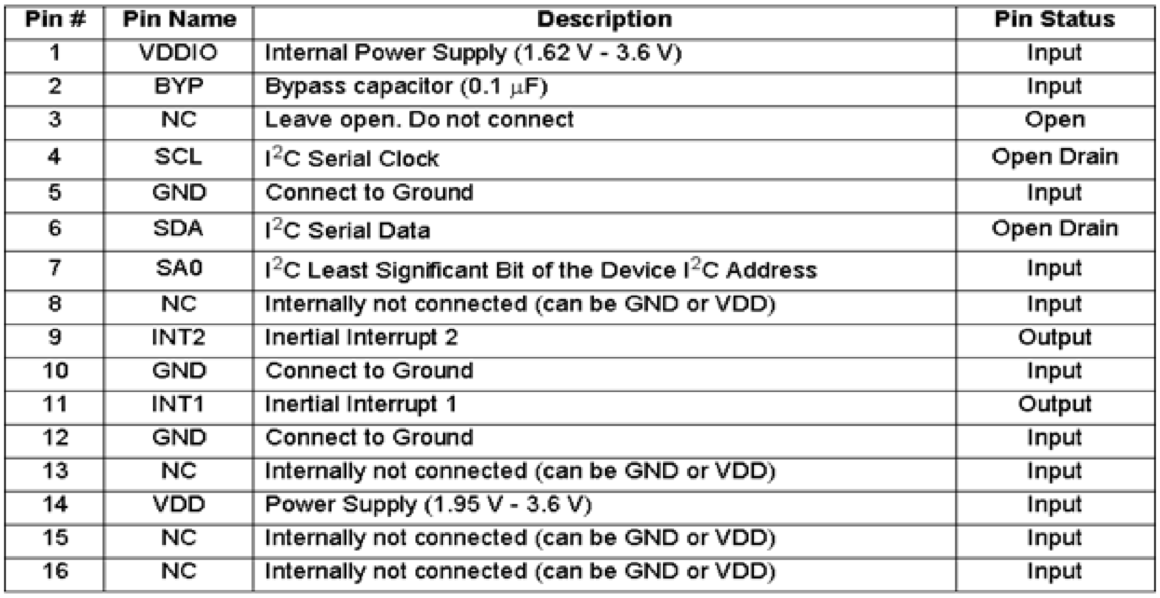


Figure 5.3.4: *Pin diagram matched up with application diagram(16 pins, 14 input and 2 output pins).* Reprinted with Permission by *Free Scale Semiconductor.*

Just to add more clarity to the accelerometer we are using the following chart will distinguish all of the pins and all of the statuses of what they do. The two most important pins that will be connected to the microcontroller will be pins 9 and 11. Those two pins allow the users to set up an interface and program them to tell the accelerometer how we want the blimp controlled and what specifications we will need. By using the SPI interface, (which is very close to C programming) we will setup the specific interrupts to measure the motion and vibrations of our blimp. Pin 4 and pin 6 are distinguished by having an Open Drain which requires the use of a pull up resistor. Some of the items we can control by using the SPI interface involve the SCL clock frequency, the stop and start condition, the stop data hold time, the SCL clock low and clock high time, the SDA and SCL rise and fall time, the capacitive load for each bus and finally the pulse width of spikes on SDA and SCL that must be suppressed by internal input filter. We can control the sensitivity as well which would also force us to change the counts. In 2g mode we are able to do 256 counts per g, in 4 g mode we are able to do 128 counts per g and finally for 8 g we are able to do 64 counts per g. In ***Figure 5.3.5*** below, we see the pins that we need to connect the accelerometer module to for either a breakout board or so we can put it directly onto the Printed Circuit Board (PCB).



Pin #	Pin Name	Description	Pin Status
1	VDDIO	Internal Power Supply (1.62 V - 3.6 V)	Input
2	BYP	Bypass capacitor (0.1 μF)	Input
3	NC	Leave open. Do not connect	Open
4	SCL	I^2C Serial Clock	Open Drain
5	GND	Connect to Ground	Input
6	SDA	I^2C Serial Data	Open Drain
7	SA0	I^2C Least Significant Bit of the Device I^2C Address	Input
8	NC	Internally not connected (can be GND or VDD)	Input
9	INT2	Inertial Interrupt 2	Output
10	GND	Connect to Ground	Input
11	INT1	Inertial Interrupt 1	Output
12	GND	Connect to Ground	Input
13	NC	Internally not connected (can be GND or VDD)	Input
14	VDD	Power Supply (1.95 V - 3.6 V)	Input
15	NC	Internally not connected (can be GND or VDD)	Input
16	NC	Internally not connected (can be GND or VDD)	Input

Figure 5.3.5: Table explaining all pin connections for MMA8453Q accelerometer. Reprinted with Permission by Free Scale Semiconductor

All of these images and tables help describe what's inside the accelerometer but now let us talk about the functionality of it and how it will help our project along more specifically and why we choose it over the others. First and foremost it uses very little voltage which will deal more with the battery, however the smaller the battery the less our blimp will weight which once again is very important to our project. We can function between 8 bit and 10 bit data which we will end up using the 8 bit due to familiarity but options are always good to have. We can cycle between the different g settings to ensure we are getting the most accurate reading from our blimp. It also carries a low power setting which will save battery and allow our blimp to fly in the air longer. Also in dealing with battery life it has an auto power wake/sleep mode which will save even more battery power. It has motion detection in the free-fall direction. All functions will work on the different setting that we give the accelerometer so no matter the force we put it on everything will be accounted for.

The following image is useful more in terms of the coding of our accelerometer. As previously stated Pins nine and eleven represent the interrupt pins that will be programmed using the SPI interface. Acceleration data is able to be retrieved by using the SPI and will be directly linked to the microcontroller. The accelerometer has an automatic update that lets the microcontroller know when new data is available. This simplifies the data synchronization in the digital system of our blimp. To enable the SPI interface VDDIO line must be on the high end to the supply voltage. There are three different serial interface pins. First is the SCL pin which is the serial clock, and then there is SDA pin which is the serial data, and finally the SA0 pin which is associated with the least significant bit of the two devices. The first two are associated with the bus SPI bus while the SDA pin is bidirectional which is used to send and receive data to/from the interface. In ***Figure 5.3.6*** we see the interrupt controller being used with the accelerometer to pull data from it.

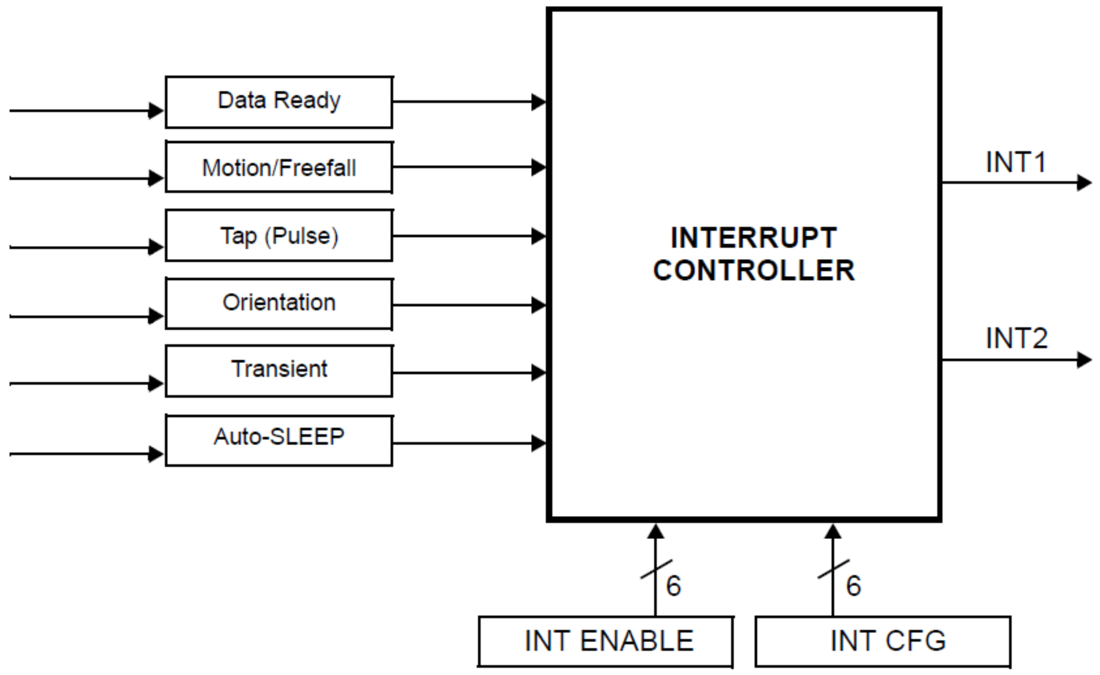


Figure 5.3.6*: Displays all possible interrupts and the order in which they are read in. Reprinted with Permission by Free Scale Semiconductor*

Like is previously stated the accelerometer and the gyroscope will make sure our blimp stays in perfect condition and does not run into any unwanted outside forces and if it does the corrections we can make on the fly to stabilize our blimp. The use of these two mechanisms together will actually make an IMU or inertial measurement unit which is often used in personal navigation tools and also body tracking and measuring amongst a sleuth of other things. Now I will focus on the second half of our stability equation which is the gyroscope.

Section 5.4 Third-Subsystem (Ground Station)

The purpose of the ground station is to act as a base station for all communication to the blimp and to the controlling programs. We want to interact with a computer program and an android application, so we need a USB connection for the computer and a Bluetooth connection to interface with the Android phone. The ground station will also house the second transceiver, the first being on the blimp, for long range communication (up to 750 meters, but will be limited to 100 meters). To control all these communication modules we will need a microcontroller to direct out all the signals to the corresponding places as well as set the transceiver to transmission (TX) or receiver (RX) mode. This section will cover all the devices used to make up the ground station and how they will work together at the end.

Section 5.4.1 PIC Microcontroller (PIC18F2550)

For the microcontroller in the ground station we decided to use the PIC18F2550, for its low cost, good speeds, and pin out availability. The nice thing about the PIC microcontroller is through the software setting we can make any pin mimic a UART, SPI, or I^{2}C connection through the use of software. We will probably have to do this for some of the pin since we have learned that we need 2 UART connections and that SPI and UART share the same hardware pins.

In **Diagram 5.4.1** below, we see that UART is on pin 18 and 17, but then SPI is also on pin 22, pin 21, pin 18, and 1 pin of our choosing for slave select. SPI can handle multiple devices on so if we need more SPI devices to plug in we can use this line for them, so we focused on using the hardware SPI over the software SPI configuration. The original idea was to use hardware UART and hardware SPI, and since we only have 1 UART pin to use on the PIC18F2550, we needed to use a multiplexer to select which device the UART will transmit and receive from.

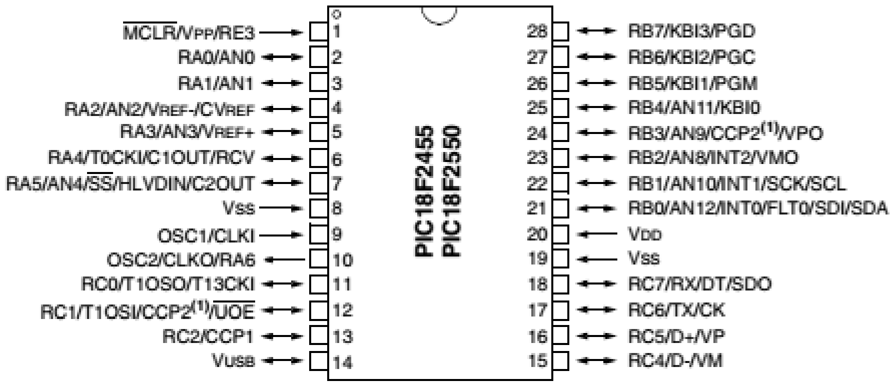


Diagram 5.4.1 *- Pin Layout of PIC18F2550 “Reprinted with Permission by MicroChip”*

The first thing we need to consider is the power pin of the PIC18F2550; we need to supply a 2 to 5.5 volt to this pin for the PIC to operate. The pin we use to power the PIC is Pin20, which is labeled V_{DD}, and our voltage regulator output will supply this line with 5 volts. With all the power line we cannot forgot to connect the ground pin of the PIC18F2550 to the common ground of the whole circuit. The ground pin is labeled V_{SS} and there are two available grounds on either side of the PIC microcontroller, the pin available are pin 8 and pin 19. In **Table 5.4.1** below, it shows the PICs pin location and the label name and also a reference to which range of pins would be best to connect the ground to if the device connected to that pin has a ground we need to connect.

Pin	**Label**	**Function**	**Pin Range**
20	V_{DD}	2 to 5.5 volt	N/A
8	V_{SS}	Ground	1 - 14
19	V_{SS}	Ground	15-28

Table 5.4.1 *- PIC Pin Layout for Power and Ground*

To connect our PIC18F2550 to our transceiver module, we will need to use the 4 SPI pins on the microcontroller. In **Table 5.4.2** below, it specifies the 4 pin number that we use and their purpose for SPI master/slave selection.

PIC Pin	**Label**	**Master**	**Slave**
22	RB1/SCK	Serial clock output from master [SCK]	Input to slave [SCK]
18	SDO	Serial data output from master [MOSI]	Input to slave [MOSI}
21	RB0/SDI	Serial data input from slave [MISO]	Output from slave [MISO]
4	RA2/SS	Optional slave (chip) select [SS]	Slave select [SS]

Table 5.4.2 *- PIC Pin Layout for SPI Connection*

Then we will worry about which pins are going to be dedicated to UART interfaces. Since now we are using hardware SPI and that blocks off the hardware UART pins. First we need to find out which digital i/o pins are TTL compatible, the list of pins that we can use for this is RA6 (pin10), RA1-3 (pin 3, 4, and 5), RA5 (pin 7), RB2-7(pin 21-28), and RC7 and 6 (pin 18 and 17) are all digital I/O pin available for TTL transmission. The ruled out pins are RA2, RC7 and RC6 because those are needed for SPI, and pin 10 cannot be used because it is needed for the crystal oscillator we need for the timer. So we decided we are going to use the PortB (TRISB) pins for any UART connection we need, with this we have 6 pins which come out to 3 different devices able to connect to the PIC18F2550. Another plus to this is that when we write the software we only need to make one call to TRISB to set up all the I/O pins to either input or output. Another thing we need to worry about is that these two modules never communicate to the ground station at the same time. To make sure that only one is transmitting and receiving, we will implement a switch which will give either a 1 for the USB-to-UART module to get data from the computer GUI or a 0 for the Bluetooth module to start sending out data to the android phone. In **Table 5.4.3** below, it shows the software UART connection we will make and to which device it goes to.

PIN	**Label**	**Device Direction**	**Device**
28	RB7	RX	Bluetooth Module
27	RB6	TX	Bluetooth Module
26	RB5	RX	CP2102 USB Con.
25	RB4	TX	CP2102 USB Con.
24	RB3	Input (1 or 0)	Switch
23	RB2	Free	Free

Table 5.4.3 *- PIC Pin Layout for UART Connections Used*

Then we need an 8 MHz crystal oscillator in our circuit to synchronize the transmissions and for the timer of the circuit. With the 8 MHz crystal oscillator and the right FOSC setting in software we can do transmission speeds of 2.64Mbps which will be perfect for the 2Mbps on the transceiver and the Bluetooth module. The crystal oscillator we have picked out from digikey is a 2 pin “CRYSTAL 8MHZ 20PF SMD”. In **Table 5.4.4** below, it shows the pins it will be connected to on the PIC18F2550, we do not have much experience working with an external oscillator but it will mostly be configured by the software in C18.

Pin	**Label**	**Component**
9	OSC1/CLK1	8Mhz Crystal Oscillator
10	OSC2/CLK0/RA6	8Mhz Crystal Oscillator

Table 5.4.4 *- PIC Pin Layout for Crystal Oscillator*

Section 5.4.1.1 *Programming the PIC18F2550*

Since this is a Microchip product, to program this PIC microcontroller we need to use their software called MPLabX. The programming language we chose to use is the C18 library for MPLabX, since it is close to the C programming language we are all familiar with. The coding style of the C18 language is very familiar to when we were programming the MSP430 using C on Code Composer. First we need to set the pins to input and output using TRISA, TRISB, TRISC and these ports correspond to RA, RB, and RC respectively. Then we need to setup the external oscillator to use the 8 MHz crystal this will be related to the OSC setting. For the software UART we need to set that up by using the SerOut and SerIn methods for C18 to use the TRISB ports (RB7 to RB4). We need to read in-depth more about the C18 libraries so we can use it correctly and the method inputs for something like SerIn and SerOut are correct. The C18 library has a reference manual that we are currently looking through and we will start coding the PIC once we get all the modules and the components to put it together on a breadboard. Once we start the C18 code we can use the PicKit2 programmer to program the PIC18F2550 with our code and start the debugging process for the pins and the program.

Section 5.4.2 *Nordic Semiconductor Transceiver (nRF24L01)*

For the receiver and transmitter module we decided to use the nRF24L01 transceiver module to meet both our needs. We need the blimp to be able to receive signals and we need the ground station hooks up to the computer to send and receive signals. We decided to make the blimp be able to send signals too just as a “Ping-Pong” system to make sure it can reach the ground station once in a while in case it is in patrol path mode and will not receive controls from the ground station.

The nRF24L01 is a Nordic Semiconductor transceiver that is on a single chip. Our model will have an external antenna so we can control the gains of the signal, but there are chips that have internal antennas. The internal antennas seem to have problems with controlling the output power, while with the external antenna we can swap out antennas to get a 0dB gain to get maximum output and range. This Nordic transceiver boasts its easy hardware setup with most microcontrollers and the need of very few external passive components to design the radio system with the nRF24L01. The transceiver is configured and operated through the Serial Peripheral Interface (SPI) and the ATmega128 microcontroller we are using has the pin for us to connect to. The ISM frequency band for this Nordic transceiver is 2.400 - 2.4835GHz. Packet encapsulation and data transmissions is handled by the Nordic’s Enhanced ShockBurst Protocol, which controls the way the high-speed link transmits and supports up to 6 transmitters sending to 1 receiver using multiple nRF24L01s in a star network topology. Our Nordic Transceiver Module (nRF24L01) will use an older standard called

Gaussian-frequency-shift-keying, this module has many good reviews from users and has proven to work at its recorded distances and frequencies in our required testing environment.

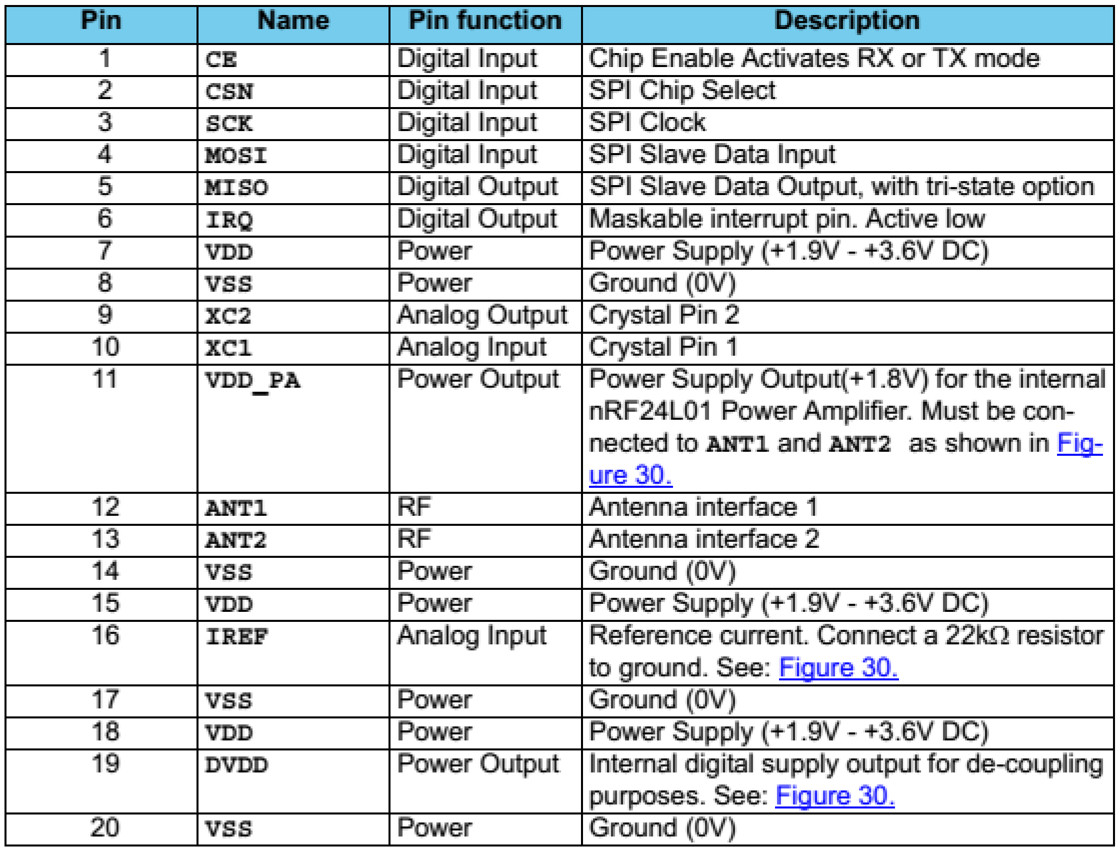
This transceiver is desirable because it designed for ultra-low power wireless applications and it is perfect for our blimp which we want to stay up in the air for the longest time possible. The input voltage for the transceiver is 1.9 to 3.6 volts, and we will supply a 3.3 volt power line on the microcontroller to the nRF24L01. The current draw for the transceiver module is 11.3mA at our optimal 0dBm output power and for the receiver the maximum it will be is 13.3mA at the maximum on-air data rate of 2Mbps. The data rate of the SPI connection to the transceiver module can go up to 10 Mbps. These statistics are shown formatted nicely in **Table 5.4.5** below.

Frequency	2.400 - 2.4835GHz
RF Channels	126
Pins	20
Data Rates	256 kbps, 1 and 2 Mbps
SPI Data Speed	Up to 10 Mbps
Transmitter Output Power	11.3 mA
Receiver Output Power	13.3 mA at 2 Mbps
Interface	SPI (Max: 8 Mbps) using 5V Logic
Power Supply Range	1.9 to 3.6V
Packet Handling	Enhanced ShockBurst (Packets: 1 to 32 bytes)

Table 5.4.5 *- Specifications Information for nRF24L01*

We decided to use a frequency that many RC cars use the 2.4 GHz frequency. This ISM band is desired because we don't need a license to operate at this frequency. The Federal Communication Commission (FCC) regulates all these frequencies and puts this one up as free for public use. The transceiver has two modes for sending out data, one at 1Mbps and another at 2 Mbps. The 1 Mbps transmissions have a range of 300 meters, and the 2 Mbps transmission has a range of 750 meters. If we need more range we can also opt for the nRF24L01+ which sends out data at 256 kbps but has a range of 1000 meters. These range tests are open area range test, which means that there are no trees or buildings in the way.

In the **Table 5.4.6** below, it shows all the pin functions of the nRF24L01. We will focus on only using 8 out of the 20 available pins as the other pins will be already used in the breakout board for the module such as the connections to either the antenna (Pin 13 and 13) or the internal oscillator clock (Pin 9 and 10).



Pin	Name	Pin function	Description
1	CE	Digital Input	Chip Enable Activates RX or TX mode
2	CSN	Digital Input	SPI Chip Select
3	SCK	Digital Input	SPI Clock
4	MOSI	Digital Input	SPI Slave Data Input
5	MISO	Digital Output	SPI Slave Data Output, with tri-state option
6	IRQ	Digital Output	Maskable interrupt pin. Active low
7	VDD	Power	Power Supply (+1.9V - +3.6V DC)
8	VSS	Power	Ground (0V)
9	XC2	Analog Output	Crystal Pin 2
10	XC1	Analog Input	Crystal Pin 1
11	VDD_PA	Power Output	Power Supply Output(+1.8V) for the internal nRF24L01 Power Amplifier. Must be connected to ANT1 and ANT2 as shown in Figure 30.
12	ANT1	RF	Antenna interface 1
13	ANT2	RF	Antenna interface 2
14	VSS	Power	Ground (0V)
15	VDD	Power	Power Supply (+1.9V - +3.6V DC)
16	IREF	Analog Input	Reference current. Connect a 22kΩ resistor to ground. See: Figure 30.
17	VSS	Power	Ground (0V)
18	VDD	Power	Power Supply (+1.9V - +3.6V DC)
19	DVDD	Power Output	Internal digital supply output for de-coupling purposes. See: Figure 30.
20	VSS	Power	Ground (0V)

Table 5.4.6 *- Table of Pins for nRF24L01. Reprinted with Permission by Nordic Semiconductors.*

The **Figure 5.4.1** below shows the Pins and their names of the nRF24L01 transceiver module that we will be using to connect it to the PIC18F2550. We will use Pin 2 (CSN) to select the transceiver as the current device on the SPI line (in case there are multiple devices). We will use Pin 3 (SCK) as the SPI clock. Then we have Pin 4 (MOSI) as the Data Input and Pin 5 (MISO) as Data Output. Pin 1 (CE) will be used to select if the chip is in Standby mode (active low - set to 0), or if this pin will enable the transmission mode (TX) and receiving mode (RX) (active high - set to 1). We will be using Pin 7 (VDD) to supply the 3.3V DC power supply to the module. We will be using the Pin 6 (IRQ) as the interrupt signal for the transceiver, the interrupt is active low (set to 0). Pin 8 (VSS) will be used as the ground that we will connect to the common ground of our ground station.

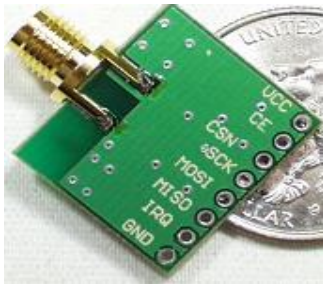


Figure 5.4.1 *- Usable Pins on the nRF24L01 Module. Reprinted with Permission by Nordic Semiconductors.*

There is an extra feature called Enhanced ShockBurst™ hardware protocol accelerator on the nRF24L01 transceiver that we are debating if we want to use. If we want to use this feature we have to edit the configuration setting in the header file for the nRF24L01 on the PIC18F2550, to enable it and then figure out the controls the sending and transmission (they are different from regular RX and TX mode). The features provided by the Enhanced Shock Burst protocol include: Automatic packet assembly (Preamble, Address, CRC), Automatic packed detection and validation, Dynamic payload length (1 to 32 Bytes), Auto retransmit, Auto Acknowledgment with optional payload, 6 data pipe MultiCeiver™, and 3 separate 32 Bytes TX and RX FIFOs.

Section 5.4.3 Bluetooth Module (Bluegiga WT12)

Since we want the blimp to communicate with an android phone, we decided to add a Bluetooth connection so we don't have to have a wire going to the phone. Out of the two Bluetooth modules we looked at we chose to go with the Bluegiga WT12, since it has more support and positive reviews online.

The features of this Bluetooth module that are important to note are that the Bluetooth Version is 2.1 + EDR. With version 2.1 +EDR, we have two data rates available to us in this, there is a 2Mbit/s and a 3 Mbit/s, to synchronize all our speeds we will be using 2Mbit/s. This device is a Bluetooth class 2 radio, it uses the 2.4 GHz ISM Band, for this class the specifications are +3 dBm Transmit power, -86 dBm Receiver sensitivity, and the range is 30 meters line-of-sight. Other noteworthy features include an integrated chip antenna, 802.11 coexistence interface, six software programmable IO pins, and the use of integrated I WRAP Bluetooth stack. Then to counteract interference from other 2.4 GHz ISM band devices, the module comes with a metal shielding to prevent RF interference.

In the **Diagram 5.4.2** below, we have the pin diagram of the Bluegiga WT12 Bluetooth module; it shows all the useable pins that we can plug into the microcontroller. Some of the features we will consider using on this device are host processor interface with UART or USB, SPI interface for firmware and parameter upgrades, and the 6 GPIO pins which are laid out and labeled for us in the breakout board (RTS, RXD, TXD, GND, 3V3, and CTS signals).

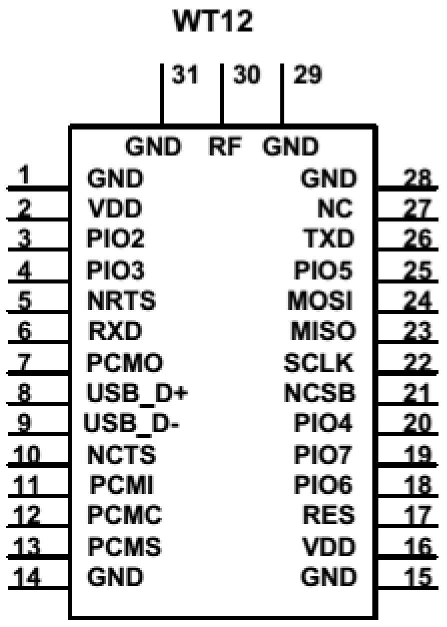


Diagram 5.4.2 *- Pin Layout of WT12 Bluetooth Module. Reprinted with Permission by Bluegiga.*

The **Figure 5.4.2** below shows the breakout board with the Bluegiga WT12 module, we are using the full model with extra pins instead of just the UART only model. We are using UART connection for the Bluetooth module but just in case we cannot get both the connection to the computer and the Bluetooth working at the same time we plan to change the pin interface of the Bluetooth module since we can easily change what kind of serial communication it is using by using different pin set. We can either create the breakout board for the module ourselves or use a premade one like the figure below. The nice thing about this breakout board is that all 31 pins from the module are available to use with a pin hole, but we will only use the 6 pins (RTS, RXD, TXD, GND, 3V3, and CTS signals). The request to send (RTS) signal will be tied to an interrupt pin and the clear the send (CTS) pin will be tied to ground since we do not need flow control.

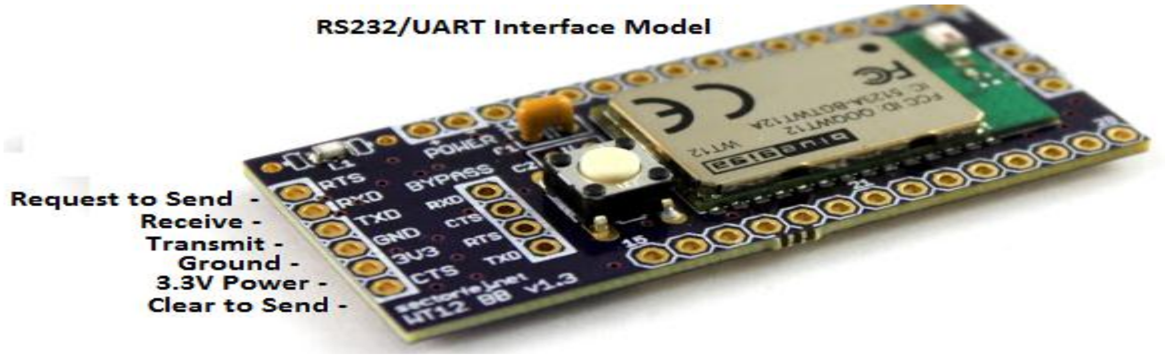


Figure 5.4.2 *- Usable Pins on the WT12 Bluetooth Module. Reprinted with Permission by Bluegiga.*

The Bluegiga WT12 Bluetooth module is pretty low power consumption, it is not as low as the Nordic Semiconductors' nRF24L01 transceiver but with the 9V lithium battery we plan to use it should last us a couple of hours. The **Table 5.4.7** below shows the current draw of the WT12 module.

Link active:	Minimum: 7 mA	Maximum: 60 mA
Link active in sniff mode:	2.5mA	
Link active in park mode:	2.5mA	
Idle with deep sleep:	<1mA	

Table 5.4.7 *- Power Consumption for the WT12 Bluetooth Module*

Section 5.4.4 Computer Connection Module (CP2102)

For the connection to the computer from the ground station we will use the CP2102, this is a USB to TTL/UART Module. This module will work with the transceiver module on the ground station to translate the signal to and from the blimp for the use of our C++ Program. We need to test this but I believe we should be able to get the UART signal commands translated by using this module. If the signal is different, we will need to look into using a logic level shifter op-amp in order to get the right signal format we need, this will be tested by using an oscilloscope and looking at the peak to peak values of the signal we get out of the transceiver and then the signal we get out of this CP2102 module.

The **Figure 5.4.3** below shows the CP2102 module that we will be using for the connection to the computer; this module turns the signal from our receiver and provides the UART serial connection to USB interface. The drivers needed for this module is supplied by Silicon Labs and can be downloaded from their website by searching for “CP210x USB to UART Bridge VCP Drivers”. We will only be using 4 of the pins. We will use the 5V pin to supply power to the module, since we will be using a 9 to 5 volt regulator in our circuit. Then we will be using the TXD and the RXD for the transmission and receiving pin respectively. On the PIC18F2550 that we are using, the TXD pin will be connected to Pin 17 to transmit to the PIC, and the RXD pin will be connected to the Pin 18 to receive data from the PIC. Then it is required we use the GND pin and connect that to the common ground with our circuit

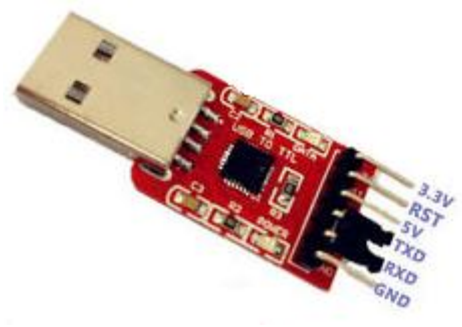


Figure 5.4.3 *- USB to TTL/UART Module With Connector with Pin Assignments. Reprinted with Permission by Silicon Laboratories.*

Section 5.4.5 9 to 5 Volt Regulators (LM7805C)

In the ground station, we need a voltage regulator to make sure that the voltage and current supply into the transmission modules and the microcontroller is not too high and stays constant. We are using either a 9 Volt wall plug AC adapter or a 9 Volt battery to power the USB ground station. Either way we need a regulator to drop the voltage down to 5 Volts because that is the power requirements in the PIC18F2550 is required to be powered by 5 Volts. Then for the Nordic Transceiver (nRF24L01) and the Bluegiga Bluetooth (WT12) Modules we need an input voltage of 3.3 Volts, for these two power connections we will simply put a resistor in series with the power lines to step down the voltage to 3.3 volts.

The LM7805C is a 3 pin voltage regulator, the first pin is an input pin in which we will supply our 9 volt DC power to, the second pin is the ground, and the third pin is the output pin which outputs 5 volts and 1 ampere.

Section 5.5 Graphic User Interface (GUI) Design

The program to interface with the Surveillance Blimp will be programmed in Visual Studio C++ in order to create a graphical user interface for the user to easily use the blimp. The primary features of this program will include a link to the IP Camera on the Balloon and a user control system for the acceleration/deceleration/ascent/descent/turning of the blimp which will include a controller connected to the computer and a patrol path through the drawing of a circle on a map interface.

Section 5.5.1 Creating Displays in the GUI

The Graphical User Interface (GUI) will be coded in Visual Studio C++ for the Integrated Development Environment (IDE). This is so we can compile the GUI in C, C++, and C++/CLI programming languages. The main purpose for choosing this language was because it can be used for low level and high level programming task. The low level task would be interfacing with the hardware and hopefully give us an easier time in communicating with a ground station that houses the receivers for the PCB and camera transmitters. The high level task would be designing a graphical user interface (GUI) that would give the user an easier time in controlling the blimp (through button presses) and handle the visual detections of the camera.

Visual C++ provides the user with a basic setup of the C++ graphical design in the “Win32 Project” design option. With these basic options most of the design will go into the method “*LRESULT CALLBACK WndProc*” that will create and give actions to the Buttons, Text Boxes, and Text then set them to certain locations of the Window Display.

Section 5.5.1.1 Creating Buttons for Blimp Controls

For Visual C++, we use the “*case WM_Create*” to create the all the buttons, text boxes, and text. For Buttons, each need to be defined using #define with a given name and value, the value should start at 1 and increment for each additional button used.

Then to give functionality to these buttons, we will use “*case WM_Command*” to make the designated button presses send commands to the blimp, which will be routed to the universal asynchronous receiver/transmitter (UART). To look for the event of the button being pressed we use “LOWORD(wParam)” in the IF’s case statement with the appropriate action within the IF statement. There will be a problem if two of these buttons are pressed at the same time, but some testing will go into it and make sure that it sends a combined action to the blimp to go in both directions if it is valid. Some invalid statements include trying to accelerate and decelerate at the same time, or going left and right at the same time.

Section 5.5.1.2 Displayed Sensor Values on GUI and Position Auto-Correct

We want to keep track of the speed and position of the blimp on the GUI in case we cannot see it. So if any of these values for speed or position of the blimp is faulty we want it to send a command to the blimp to go to a “safe” value. These values displayed are mainly for debugging the PCB as we test out where the position of the blimp is as we send out commands to make it move either through the GUI buttons, automatic path patrol or the controller. The methods below modify the initialized text boxes from the “*LRESULT CALLBACK WndProc()*” method.

Section 5.5.1.3 Display Sensor Value Methods

UpdateAccelerometer() - After every data packet of information is received from the blimp, we will parse out the UART receive message and take the Accelerometer data and determine the speed of the blimp and if its tilt or pitch is in the correct position. From this method it will call InitAutoCorrect() method if something is faulty.
UpdateGPSLoc() - Along with the same data packet as above, the GPS coordinates will be displayed on the GUI, this will also update the Map API on the GUI, and visually shows the user where on a big area map the blimp is located. If the GPS coordinates are out of the range we want it, it will also call the InitAutoCorrect() method to send controls to the blimp to reposition itself.

Section 5.5.1.4 Position Auto-Correct Methods

InitAutoCorrect() - This method will use the UART methods to send controls to the blimp to reposition it within the area we deem safe. This method will be checked after every send or receive to the blimp and should happen every 2 or 3 seconds. The safe values that the blimp goes to are 100 meters from the ground station is a spherical space. If the blimp is moving too fast to the spherical perimeter it will slow down and alert the user that one of the connections will be lost (these values will come from the max range test of the hardware).

Section 5.5.2 - Transmitter/Receiver Link

The GUI will communicate to the printed circuit boards' (PCB) blimp and camera controls by using the universal asynchronous receiver/transmitter (UART). The UART will be designed in C++ by opening up a serial COM port and sending data out or receiving data.

Serial Port Desired Options:

Bits Per Second (Baud Rate):	115200
Data Bits:	8
Parity:	None
Stop Bits:	1
Flow Control:	None

We want the baud rate to be as high as possible, so this variable will change through the testing phase of the transceiver. The higher the transfer rate we can achieve the faster we can send the transmitter/receiver to sleep to save power.

To transmit and receive from the ground station we need to use a Serial C++ library. We have decided to use an open source free High-Performance Serial C++ and the header file that contains all these methods is "StdAfx.h". We will use these methods to buffer the data received from the ground station when the transceiver is in TX mode, and the method will write to the ground station when the transceiver is in RX mode.

Section 5.5.2.1 Transceiver Link Methods

Serial Communication Methods

Serial.Open() - This method has a input of the COM port number and used to detect which COM port the Windows Device manager sees that the ground station is installed on.
Serial.Setup() - This method has a input of the Baud Rate, Data Bits, Parity, Stop Bits, and Flow Control. It is used to setup the transmission speed and how large the data packets will be. (These setting should match the PIC microcontroller settings or data will be corrupted when sending/receiving)
Serial.SetupHandshaking() - This method control what kind of buffering method we use for the transmission line. We need to make sure that the device is not trying to send and receive at the same time; the data is buffered if this case tries to happen and retransmitted later. So there are many Event Handling Methods that is associated with this, to make sure the line is clear to send or if the line is requesting a packet.
Serial.Write() - This method writes out to the serial communication device (the ground station) for the Blimp to receive.
Serial.Read() - This method waits to receive from the serial communication device (the ground station) and buffers all the data into an RXbuffer until the full message is received.

Section 5.5.3 - OpenCV

We chose to use OpenCV because it was a free open source computer vision software and has algorithms to detect and recognize faces, identify objects, and classify human actions in videos, track camera movements, and track moving objects. OpenCV also has a marker system which will provide an overlay for a augmented reality environment. OpenCV is perfect for our C++ language because OpenCV was written in optimized C/C++ and its library has can take advantage of multicore Systems. It will take more research or use of the library to see if this multi-core feature is automatically implemented or if we have to actually call algorithms on our own created threads for our C++ Application.

OpenCV also has libraries for Android development. Depending on how complete these libraries are we will also implement the android app to have Human/Face Recognition and use the target and follow system explained below.

Section 5.5.3.1 - Human/Face Recognition

Since OpenCVs' version 2.4 has introduced a new Class called FaceRecognizer, it has become easy to just call methods that will recognize that there are faces on the video stream. The three algorithms that can do face recognition in this class are Eigenfaces "*createEigenFaceRecognizer()*", Fisherfaces "*createFisherFaceRecognizer()*", and Local Binary Patterns Histograms "*createLBPHFaceRecognizer()*". There are examples of these working with still pictures but it will take some research and testing out to get it to work with a live video feed. We will probably want to only sample 1 frame a second as we do not expect someone to be moving across the video feed at a very fast past. Although we want everything to be real time, we expect some lag in the face recognition and the marker system we will implement highlighting the people in the video feed.

Section 5.5.3.2 - Infrared Alternative Full Body Recognition

We have a possibility of having the Go-Pro Camera to use an infrared lens instead of the normal optical lens. If this is the case we will go into using the algorithms to look for certain shapes and colors to find out which objects are people and which are just environmental objects. The targeting system will be changed from face recognition to just full body recognition because it would be hard to make out faces through the infrared lenses at such high altitudes. Using this system would make it easier to differentiate between actually people and just images of people like on billboards or posters.

Section 5.5.3.3 - Target and Follow System

This program future will be an overlay on top of OpenCV video feed. To provide surveillance of one person we wanted a feature to allow for the user to make the blimp follow someone dependent on the video feed. For this targeting system, we will use TemplateMatching feature of OpenCV, this allows us to compare an image (of a face) we have defined and look at another image to see if we can find any instance of that image (of a face) within it. If we have enough time after implementing all of the current features, we would like to implement a feature where we can insert a picture of someone into the C++ Application, through a file or a digital camera picture and the C++ application will search for that person within the video feed and automatically mark and follow it.

To debug this system and mark sure that the person its tracking kept within the screen there will be a small arrow pointing to where the person is moving within frame if they are moving still. The targeting will be done by OpenCV by the following of the target will be done by a combination of camera movements and blimp movements. To determine whether the camera moves or the blimp moves it will be dependent on the camera's current and maximum rotation and position.

Section 5.5.3.4 OpenCV Methods

Face Recognizer Class

createEigenFaceRecognizer() - This is the method in OpenCV is used to recognize the faces in the video frame. We will then use a polygon draw method to encase the faces we found in a square, where the user can select that person to make the blimp follow them (Calls on the Blimp Control method to keep the person centered on the screen). We will also save the selected persons' image within the program and use it in the template matching method for comparison.

Template Matching Class

image.CurrentImage.MatchTemplate() - This is a method in the OpenCV library that is used to match the current image saved to the video feed. This is the second part to the tracking system where the match template must be in the center of the screen or close to it, or else it will call the motor control methods to position the center of the face to the center of the screen. This method gives a percent match of the image stored with the current image on the video feed and then we need to write an auxiliary function to this to return the pixel position values so we know where the image is.

Section 5.5.4 - IP Camera Link

The IP Camera will have a separate link to send data over to the receiving station and then to the computer. Since the camera module should have a encoder to send over the bit stream of video over to the receiver. Encoding the video data on the Blimp's PCB will ensure that bandwidth transmission link of the video transmitter will be optimized. The video encoding and the output of the GoPro Camera are not currently known as we do not have an exact model number yet. The receiving station link should have a video decoder attached to it to match the encoding scheme of the camera and its transmission method.

The IP camera should have a IP Address we can connect to it. For the IP Camera to be detected in OpenCV's list of cameras to stream from we need to install GStreamer, ffmpeg and xine. Once the setup and changes to the library of OpenCV for the address of the IP Camera are made we can access the IP Cameras' Video Link by making a variable of type VideoCapture and linking it to the HTTP address of the IP Camera complete with Administrator Password and Video Resolution size.

Section 5.5.5 - Map Patrol Path Interface

The map patrol path interface will allow the user to draw a circle in the interface in a map, the interface will then send out a couple of these points to the Printed Circuit Board (PCB) to circle the patrol path. While the map patrol path interface is active the surveillance blimp will be in an autonomous mode in which it will not take any controls from the controller or interface until an interrupt is sent to the PCB telling it to turn off autonomous mode.

The basics of this will start out as just a map that will have a marker pointing out the location of the blimp. The map will be displayed and pulled from OpenStreetView Maps, this is an open source map source since Google maps requires a license for using their maps outside their java scripted Google maps API.

Section 5.5.5.1- Map Patrol Path Methods (COSMCtrl Library)

InitializeMap() - This method will setup the map on the GUI interface, it will retrieve the map from the OpenStreetMap HTTP website and start the starting location as the Orlando area Map.
SetLocation() -This method will require the input of a longitude and latitude value which would be received from the ground station from the blimp. This will fix the map position to that point on the map and then set the zoom levels to show at least 150 meters around that position.

DrawPolygon() - This method will be focused around the MouseClick Event Key, clicking 5 to 6 times on the map will create a patrol path and save the coordinates to a double variable to send out to the blimp. Once the final click is made it will draw in a filled in polygon on the map, each coordinate clicked will insert a pin onto the map.
SendOutPath() - This method will take the stored 5 to 6 GPS coordinated from the DrawPolygon() method and send them out the serial transmission line in a data packet to set the blimp to automated control and tell it to stop accepting controller inputs.

Section 5.5.6 - Blimp Controls

For the controller inputs, we will use Microsoft DirectInput component of the Microsoft DirectX application programming interface (API). The controller we will be using is the Logitech® Gamepad F310, which has support for game developers using DirectInput and XInput. In case a controller is not detected, we will instead have a few buttons for the acceleration/deceleration/ ascent/descent/turning of the blimp on the Graphic User Interface (GUI).

The setup for using DirectInput C/C++ Reference requires the programmer to know the version of DirectX they want to use. For this GUI, the GamePad Controller will provide support for DirectX 9.0c and 10.1, even though the current version is DirectX 11.1. This is in order to provide support for older computer hardware. This should not affect the performance of the release program but there is a known problem for a delay in the debugging process of using DirectInput.

Section 5.5.6.1 Blimp Control Methods

These Blimp control methods will send out a message to the serial.write method for the blimp to receive the motor control string on its UART pin from the wireless transceiver.
BlimpTurn() - This method will take the input of a integer 0 to 360 corresponding to the degree in which the blimp should turn from its current heading. The degree is then used to send a data packet to the Blimp motor controls to turn its servo motors to the specified degree.
BlimpAsc() - This method will tell the blimp the point the servo motors in an up direction, raising the blimp to a higher altitude. The duration of the blimps' ascension rate depends on how long the user holds his button.
BlimpDsc() - This method will tell the blimp the point the servo motors in a down direction, making the blimp fall to a lower altitude. The duration of the blimps' descension rate depends on how long the user holds his button.
BlimpAccDsc() - This method will tell the blimp to either accelerate or decelerate, the motor speeds are controlled by a variable speed IC connected to the two motors. The motors will have 3 to 5 speed maximum settings and this method will tell the motor at which speed the motors will run at. The input for this

method is an integer from 1 to 5 and can either speed up or slow down the motors depending on the blimps current value.

BlimpCamTurn() - This method will tell the blimp to turn the blimp's camera carrier system to a specified degree. The degree the camera system can turn is between 0 to 360 degrees but we are not too sure at the moment if we want the camera to turn all the way behind to its rear or just cover these angles by rotating the blimp.

BlimpCamZoom() - This method will set the optical and digital zoom of the camera, the field of vision (input for this method) will be from 0 (0x zoom) to 100 (10x zoom). (If this feature is available on the camera we are using).

BlimpAutoMode() - This method is used to tell the blimp if it should be on automatic or manual control mode. This blimp inputs true or false and if set to true it takes the GPS coordinates of the map path interface to set an automatic patrol path for the blimp to circle around.

Section 5.5.6.2 Logitech F310 Controller Setup

Using DirectInput, we will map the Logitech F310 controller buttons to each controller function for the C++ GUI Application. The **Table 5.5.1** below maps out what each button will do.

Button	**Function**
Directional Buttons	Flight Direction of Blimp (Up, Down, Left, Right Turns)
Left Analog Stick	Flight Direction of Blimp (Up, Down, Left, Right Turns)
Right Analog Stick	Moving Camera Position
Y, B, A, X Buttons	(No Current Implementation Planned)
Right Button (RB)	Camera Zoom In
Right Trigger (RT)	Camera Zoom Out
Left Button (RB)	Blimp Increase Speed (Acceleration)
Left Trigger (RT)	Blimp Decrease Speed (Deceleration)
Back Button	Switch between Patrolled Flight and User Controlled

***Table 5.5.1** - Logitech Controller Button Layout for our Program*

DirectInput Controller Methods

The event keys for the Logitech F310 Controller will be linked to the blimp control methods listed above. When the button is pressed the value returned corresponds to that event key button being pressed. For the joystick return values it is a bit more difficult as it returns an axis that the joystick position is currently in. It will take a lot of trial and error testing to make sure the range of the axis will correspond to the degree of how much we want the blimp to turn along with doing up or down. For the buttons we get the value with the Current State() method but with joystick axes we use Get Slider() to return the value of the axes.

Section 5.6 Android Sub-Controller

The Android App will be a secondary Sub System to the main Graphical User Interface (GUI) which will provide some features like button controls using an overlay of a gamepad on the touch screen phone screen and a connection to the IP Camera. The Android programming structure and syntax is really close to the object-oriented language Java, so developing how the program will function and how event keys work should coincide with our previous experience with Java.
The challenging part of the Android App will be making it communicate with the ground station wirelessly. At the moment I plan for a direct connection using a mini-USB on the phone to the USB to TTL port of the ground station. But the current plan is to have a wireless system to connect the phone to the ground station. We want to link a short range Bluetooth connection to the ground station but that might require an additional microcontroller on the ground station to include a Bluetooth “server” device and transmitter to communicate to the phone “client”.

Section 5.6.1 Gamepad Overlay Controls

The gamepad on the android phone will use the same buttons as the Logitech controller above. The only difference will be it will be an onscreen touch pad. This will have clear buttons outlined in the different shapes of the controller and overlaid on top of the video feed from the blimp. The gamepad of the android device will work like DirectInput on the controller and we will basically mirror the methods made on the GUI and port them over to the Android phone except we won’t use a serial line to send out the message but instead the Bluetooth connection. The values returned on the android phone will be a little different as we need to determine the position of where the user touched on the screen and then determine which direction the user is gesturing their fingers on the onscreen joystick (this will correspond to up, down, left, right). The shapes on the screen will correspond to one-click buttons that will accelerate/decelerate or zoom in/zoom out.

Section 5.6.2 *IP Camera Feed*

The Android application will pull its video from the HTTP Address provided by the IP camera. It will work the same way as the C++ GUI Application as the OpenCV library is supported on Android development environment. The camera feed will cover the whole screen and the estimated size of the screen 800x480 as that is a common and default setting for the resolution on smart phones.

Section 5.6.3 *Bluetooth Connection Setup*

We need methods in the android application to setup the Bluetooth connection between the ground station and the android phone. We will use the ID set up by the ground station Bluetooth module (the Bluegiga WT12) and matched that with an ID on the android phone. This will create a pairing of the two devices and allow communication between the two devices. We will setup the connection to have a data rate of 2Mbps or 3Mbps that complies to the speed of the Bluegiga WT12 module.

Section 5.7 *– Final First Subsystem (Blimp Control System)*

The Blimp Control System had to be redesigned because of some issues programming the Atmel ATmega128. We decided to go with the other considered part which was the Atmel ATmega328p. The ATmega328p is the microcontroller used in the Arduino Uno Development board and to quickly program and solve our problems we had with the other design we sent out a design to be made with this new microcontroller. While we were waiting for the Printed Circuit Board Design to come back we developed the code to control the servos, brushless motors, and sensor modules on the Arduino Uno.

The Arduino UNO board was used for developing and testing code for the blimp and for the ground station. Powering the Arduino Uno is a USB type A port that supplies a 5v input into the microcontroller. The microcontroller that is in the Arduino board is the Atmega328P which is what was used in the ground station and in the blimp’s PCB. The Arduino UNO was also used a programmer to burn the code into the microcontroller.

Section 5.7.1 – *Microcontroller – Atmel ATmega328p*

The Atmel ATmega328p pin layout is shown in **Figure 5.7.1** below. A special feature of using the ATmega328p to program our Printed Circuit Board is that we need a special boot loader (Optiboot) loaded into the microcontroller before we have upload Arduino sketches into it. So we could not use the sampled ATmega328s from Atmel in our project by directly putting them on the Arduino Uno as a programmer. Instead we had to buy the Optiboot loaded ATmega328p microcontrollers from an Amazon Supplier.

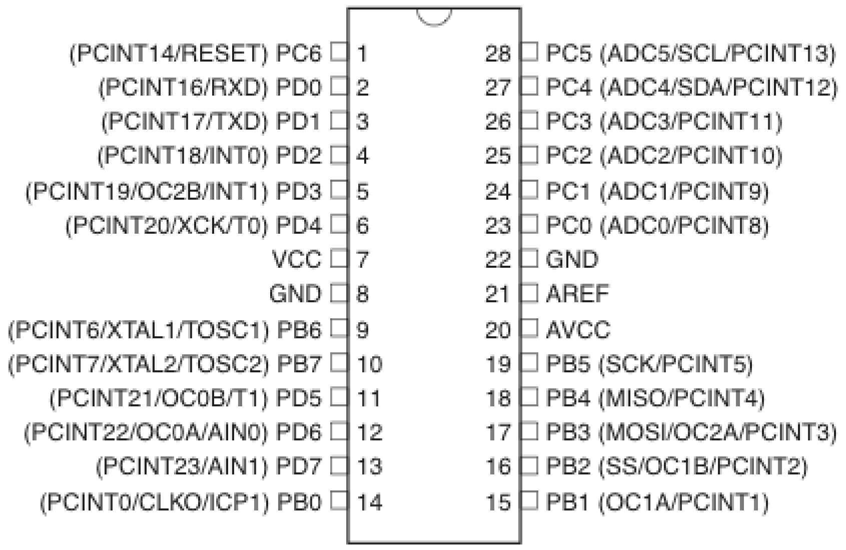


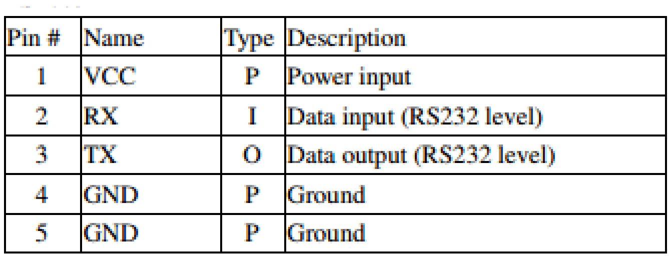
Figure 5.7.1: Pin Layout of Atmel ATmega328p

The AVR has 23 general I/O ports and pins and they are needed to get information from and to the various modules. Amazingly, like the 128, each pin has alternative functions on top of the general purpose I/O ports. There were many different pins used in the blimp and in the ground station. In the blimps PCB, the ports that were used were ports B and D. The pins that were used for servo motor and propulsion motor controls were the following: PD3, PD5, PD6, PB1, and PB2. These pins all produce a PWM signal and every one of them are controlled by an external 16hz crystal. The crystal is connected to the pins in PB6 and PB7. The IMU is connected to the PC5 and PC4 in the SDA and SCL ports respectively. These pins allow for the microcontroller and the IMU to serially communicate through I^2C. The XBEE transceiver is connected to the microcontroller through UART port in pins PD0 and PD1. The last serial communication device is the LOCOSYS GPS module and it is connected through software UART so in theory any I/O pins would work but it ended up being connected to pins PD2 and PD4.

Section 5.7.2 – *Global Positioning Module*

The LOCOSYS LS20031 GPS 5 Hz receiver is a GPS module and smart antenna receiver all in one. The module uses LOCOSYS 66 channels meaning it can sink up 66 different satellites although it will only use 22 at a time. The GPS module is surface mount which was actually adjusted to use headers on it for quick programming possibilities. This GPS was effective for the blimp because of the fast time to first fix and the one second navigation update with low power consumption. It uses a 10 Hz update rate which is more than enough, in fact that will be stepped down to the more reliable rate of 5 Hz. The GPS has a maximum altitude of 18,000 M which will be more than enough for the blimp. The baud rate is 9600 bps with 8 data bits and 1 stop bit. The GPS uses all of the basic NMEA codes including the two which are going to be used by the blimp of GSA and GLL. GSA will tell the number of active satellites while the GLL will tell the geographic position of longitude, latitude and altitude. The GPS also has a red led that flashes when it is acquiring GPS positioning data. The input voltage for the GPS module is 3.3 volts and a current draw of only 41 mA.

The GPS module will be attached to the gondola to ensure it gets the best readings possible from the satellites. The GPS module is probably the most essential piece of the PCB minus the microcontroller. The GPS will be responsible for the position of the blimp and the auto-pathing of the blimp. The GPS will use its data along with the magnetometer to ensure that the best possible route for the auto-path is being taken. The GPS module uses surface mount technology, but it needed to be programmed first and the development board was quite expensive so instead a five pin male header was soldered to the pads and plugged into a breadboard so it could be programmed by the Atmega328. As seen in **Figure 5.7.2** below.



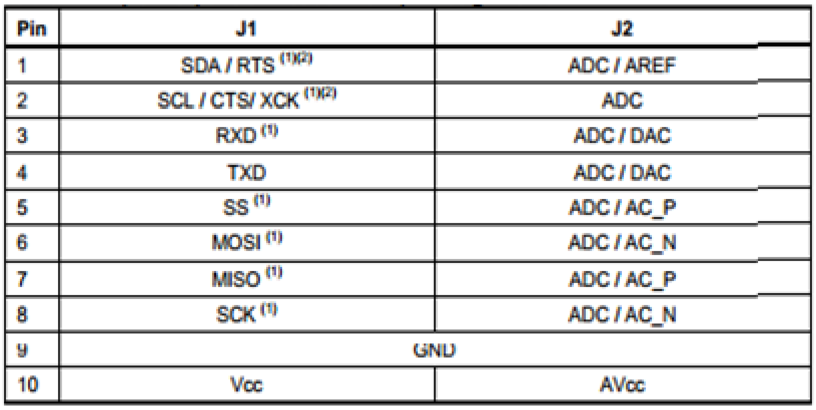
Pin #	Name	Type	Description
1	VCC	P	Power input
2	RX	I	Data input (RS232 level)
3	TX	O	Data output (RS232 level)
4	GND	P	Ground
5	GND	P	Ground

Figure 5.7.2: Pin Layout of LOCOSYS GPS Module
Reprinted with Permission by LOCOSYS.

The GPS uses UART communication for hardware UART however it was decided to save those UART pins for other modules the GPS was put on software serial UART on regular digital input output pins. PD2 and PD4 to be exact, any input output lines would have worked these two were chosen because they do not output PWM signals. For testing with the GPS module, random walks around UCF were taken to test the accuracy of the GPS module and it definitely sufficed for what we needed. The GPS also had a micro battery that will save what is in the register so no data will be instantly lost due to a loss of power.

Section 5.7.3 – *Inertia Measurement Unit (IMU)*

The Atmel AVRSBIN2 is a complete IMU board. For the IMU it includes an accelerometer, magnetometer and a gyroscope. The board is known for having a full nine degrees of freedom. It has an IMU3000 with three degrees of freedom, an accelerometer (KXTF9) which is three axes and finally a magnetometer or electronic compass which is also three degrees of freedom (HMC5883). These boards work extremely well with the Xplain MCU boards which were not used in the project, which had to be accounted for. This will be discussed a little later with the Logic Level Convertor. The IMU has female headers so it is removable from the XPLAIN board. The female headers are standard breadboard size or 1 mil away from each other. The VCC is 3.3 volts for the IMU. The IMU has two sets of 2X5 female headers which can be easily removed from the board. The pin diagram in **Figure 5.7.3** follows along with an image of the actual IMU.



Pin	J1	J2
1	SDA / RTS (1)(2)	ADC / AREF
2	SCL / CTS/ XCK (1)(2)	ADC
3	RXD (1)	ADC / DAC
4	TXD	ADC / DAC
5	SS (1)	ADC / AC_P
6	MOSI (1)	ADC / AC_N
7	MISO (1)	ADC / AC_P
8	SCK (1)	ADC / AC_N
9	GND	
10	Vcc	AVcc

Figure 5.7.3: Pin Layout of Atmel AVRSBIN2 IMU
Reprinted with Permission by Atmel

The IMU was not the first choice from senior design, originally the plan was to buy separate components and place them all together making a makeshift IMU. The following **Table 5.7.1** explains why that was not the best idea.

Design	Own IMU	AVR IMU
Cost	$14.60	$24.99
Accelerometer Values G	2,4,6,8	2,4,6,8
DOF	9	9
Advantages	Familiar with all parts	All parts are on same board. Noise Regulator is included
Disadvantages	Would have to buy separate breakout boards for programming of all modules.	More money, longer wait for parts

Table 5.7.1: Comparison Chart of Old vs New IMU.

The AVR IMU just seem to make more sense with the noise regulator included and not having to buy all of the breakout boards for all of the modules. Now just to get a bit more descriptive each module of the IMU will be broken down here.

Gyroscope

The IMU 3000 has three degrees of freedom, mainly an X, Y, and Z axis. The gyroscope uses I2C to communicate with the microcontroller. The IMU-3000 features a 3-axis digital gyro with programmable full-scale ranges of ±250, ±500, ±1000, and ±2000 degrees/sec (dps), which is useful for precision tracking of both fast and slow motions. Rate noise performance comes in at 0.02 dps/√Hz. The IMU-3000 was designed to connect directly with a third-party 3-axis digital accelerometer. The V_{DD} analog supply voltage range of 2.1V to 3.6V. The pin diagrams and outputs are not important because the IMU board already takes care of that.

Accelerometer

The KXTF9 is a tri-axis +/-2g, +/-4g or +/-8g silicon micro-machined accelerometer with integrated orientation. The accelerometer is delivered in a 3 x 3 x 0.9 mm LGA plastic package operating from a 1.8 – 3.6V DC supply. I2C interface is used to communicate to the chip to configure and check updates to the orientation, Directional TapTM detection and activity monitoring algorithms. This accelerometer was not the first choice however because it came on the IMU it will work perfectly fine for this project. Once again no schematics are needed because the IMU board already takes care of this.

Magnetometer

The magnetometer is a 12-bit ADC that enables 1° to 2° compass heading accuracy. The I2C serial bus allows for easy interface. The HMC5883L is a 3.0x3.0x0.9mm surface mount 16-pin leadless chip carrier. The magnetometer has a very low Voltage Operation (2.16 to 3.6V) and low power Consumption (100 µA). Extremely fast 160 Hz Maximum Output Rate. The magnetometer is especially important because the data will help with the auto-routing system by the use of its headings. The magnetometer as well does not need the pin diagram or pin layouts because the IMU board takes care of that.

Logic Level Converter

The level logic converter was one of those unexpected buys that needed to happen. The AVR Xplained board which runs at 3.3 volts matches up perfectly with the IMU because they were designed for each other. The Arduino Uno board is not compatible straight up with the IMU board because of the difference in voltages. A 5 volt source on a 3.3 volt module could potentially ruin or damage a module making the whole board worthless. The level logic converter takes the Vin from the Arduino board and converts everything to 3.3 volts. It also connects straight to the SDL and SCL ports. With the use of this board it ensures no frying of modules will happen.

Section 5.8 *– Final Second Subsystem (Camera System)*

In order to put the surveillance in MASS a camera system was needed. The system included a camera and a stabilization system. The camera module that was used for the MASS was the SONY SC2000 CCD 540 TVL MINI. This camera met all the specifications that were needed and had numerous examples of how it worked online. The camera system will be placed just under the gondola in the middle of the blimp. The system will be held in place by a strip of Velcro. The Velcro attaches harmlessly to the plastic of the blimp and can be easily adjusted if need be. The Stabilization System is currently able to move in a pan (left and right) but cannot move in a tilt (up and down) because of the servo mounts being too tight. The final part to the camera system is the transmitter and receiver combo that will take the video from the blimp and transmit to the ground station. The transmitter and receiver combo is a 5.8 GHz system still but is different than the one originally planned. We are using the 5.8 GHz STB Wireless Sharing Device AV Transmitter / Sender & IR Remote Extender.

Section 5.8.1 – Camera

The 1/3” Sony CCD had a good amount of features as well and was the first camera purchased. After four weeks of fighting with the international website all hope was lost on this camera and a change to domestic cameras become a necessity. This lead to choosing the SC2000, which met the three criteria that were most important: cheap, light and fast shipping. **Figure 5.8.1** is an exact picture of what the camera looks like and the measurements for it.

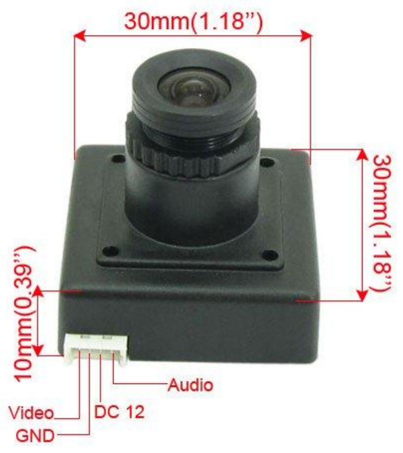


Figure 5.8.1: Sony CCD SC2000 Camera Dimensions.
Reprinted with Permission by Sony.

The camera measures at a miniscule 30mmX10mmX10mm which is approximately the size of a quarter. It weights a slight 32 grams and seeing as this is a blimp and weight is at a premium this was a major win. It would also ship from a domestic warehouse courtesy of Amazon and would be in within a few days. The camera has a 540 horizontal resolution which sounded on the lower end of clarity, however after some comparing video quality with other cameras it matched up very well. It will definitely be strong enough to identify people and items from several hundred feet up in the air. The camera came ready-made to be hooked up to a transmitter or a wall plug in. It came with a set of RCA cables that typically go straight into a TV monitor and it came with a wire connection which is to plug directly into a transmitter which was the original plan for the project. The camera uses a 12 Volt power source which is by far the largest voltage consumed on the blimp as well as a 150 mA of current. Due to these larger values the camera uses its own power source of eight AAA batteries. This battery pack was very inconvenient due to the weight of the batteries but the 150 mA current draw is extremely high for lighter batteries. The camera also had a very minimal delay coming straight from the transmitter/receiver combo to the laptop. The camera will be set up on a stabilization system.

The original plan from Senior Design I was to use a GoPro Hero 3 Black Edition due to its superior resolution and Wi-Fi capability. The GoPro, although more expensive, had numerous good features about it including mounts and software support. The price tag was steep but a member of the group could get 40 percent off which took a large part of the bill off. In the end the GoPro was not chosen because of video transmission delay that comes by the use of Wi-Fi. A three second delay was common amongst users of the GoPro camera and that was considered too drastic of a delay. Although all the research from Senior Design 1 said the GoPro was the camera of choice it was decided as a whole group that money could best be spent elsewhere. **Table 5.8.1** below shows a comparison of the old camera compared to the new camera module.

Camera	GoPro Hero 3 Black Edition	2200 CCD Sony Video Camera
Weight	204g	35g
Price	$240	$32.76
Resolution	1080p	540TV
Wifi Compatible	Yes	No
Power Supply	Battery Included	12V with 150mA current draw
Flight Time	90 Minutes	Dependent on Power Supply
Delay for Video Strea	3 seconds	Delay of Transceiver <1 second
Selected		This One!! ALL about the money

Table 5.8.1: Comparison of Old Camera Vs. New Camera

Section 5.8.2 – Stabilization System

The stabilization system is a very simple set up made of two open metal boxes that have grooves set up on the flaps that slide together in order to give movement. The following picture will help clarify the exact description of how the stabilization system looks. It is held in by several self-tapping screws.

As can be seen in **Figure 5.8.2**, the camera stabilization system also has two mini servos attached to it. The two mini servos are 9g rated and run on 5.5 volts. One servo controls the rotational movement of the camera or the panning motion while the upper servo controls the verticality of tilting of the camera. The camera system was designed to be super lightweight but still dependable enough that if some rain or windy weather came through the camera would survive.

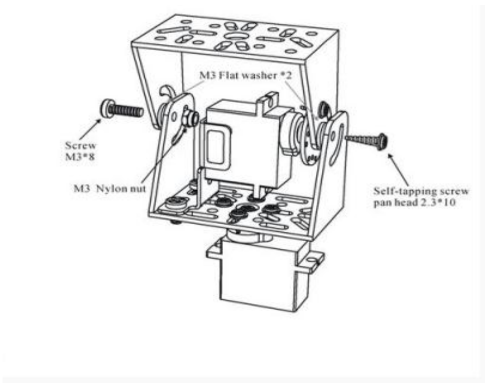


Figure 5.8.2: Layout of Camera Stabilization System with Servos.
Reprinted with Permission by Hobby King.

The camera system will be placed just under the gondola in the middle of the blimp. The system will be held in place by a strip of Velcro. The Velcro attaches harmlessly to the plastic of the blimp and can be easily adjusted if need be. The camera system should have two ways of being moved. When on the autopilot feature the IMU or more specifically the gyroscope should sent data to the ATmega 328 microcontroller and the ATmega would then account for the tilting in the blimp and move the system accordingly. When on user controlled interface mode the camera system can be controlled by using the left trigger stick to pan up to 180 degrees or tilt180 degrees. The 180 degree limit worked well because too much tilting will just show a video of the blimp while too much panning will lead to see a balsa wood gondola which does the flyer no good when trying to survey the ground. The servos will be controlled by the microcontroller by the use of pulse width modulation signals. Depending on the input from the user the servo will pan or tilt at a determined amount of degrees. For instance a push to the left will rotate the pan servo 3 degrees. The servos will be powered by a Li-Poly 11.1 Volt 2200 maH battery. The battery has two output voltages, one will go to the motors and one will go to the PCB board. The PCB board will have three, three pin headers that will be for all the servos including the two for the stabilization system. The three pins consists of a voltage, a ground and a control cord. The output voltage will be 5 volts which corresponds perfectly to the voltage needed for the servos. The servo motors are hooked up to PWM pins on the ATmega 328 which include pins 3, 6, 9, 11, and 12. The code for the servos will be covered in a different section.

Section 5.8.3 – Transmitter/Receiver

The final part to the camera system is the transmitter and receiver combo that will take the video from the blimp and transmit to the ground station. The transmitter and receiver combo is a 5.8 GHz system because of the 2.4 GHz interference. The combo that was purchased was a 5.8 GHz STB Wireless Sharing Device AV Transmitter / Sender & IR Remote Extender With 200M Transmitting Distance. The 200 meter distance was originally too short for what was needed but after some experimenting 200 meters was decided that it was sufficient. The modules are slightly bulky and the antennas are large and plastic but the overall weight of the transmitter is very light and will be placed on the outside of the gondola. The size was a little troubling but with attaching the transmitter on the outside of the gondola with Velcro that took care of that concern and the weight of 72 grams with the antenna included was decided light enough to work with. The camera will be directly connected to the transmitter via RCA cables. The transmitter then will transmit the picture to the receiver which had to be connected to an analog to digital convertor in order to produce a picture on the laptop.

The transmitter has a 5 Volt and 1 A current draw. To account for that large current draw a separate 9V battery was attached to the PCB for its own circuit. The PCB will have a two PJ-063AH power jacks which will do the powering for most of the PCB. A 9 to 5 LDO voltage regulator will be used in order to step down the voltage. A LM2940 will be the 9 to 5 voltage regulator used. By giving the Transmitter its own circuit it ensures that enough current will go to every other component connected to the PCB. The receiver is powered by a 5V 1A wall mount which is a lot easier to account for seeing as it can be plugged into an electrical receptacle. After the transmitter transmits the signal to the receiver, the receiver through the use of RCA cables will connect to an analog to digital convertor for signal to the laptop.

In order to process the video received from the video receiver on the ground, we had to connect a video decoder for OpenCV to recognize the video format. Finding the convertor was the most difficult part of the whole process. After numerous hours of searching online to found a way to transmit an analog feed to a digital computer, the video capture card was the answer. Originally the plan was to use the GoPro which was decided against due to the delay so the Wi-Fi feed that was going to be used was no more. This lead to the needing of a video capture card; Elgato was the first one that was purchased. It was going to be used for the conversion but it only recorded in mpeg4 format. The OpenCV program was not able to read this format so instead EZCAP was bought. EZCAP along with VLC made it possible to stream video into a program called direct show. This will all be discussed in the GUI section later. The EZCAP uses ULEAD software to show and edit video which would not work with the software that was needed. The EZCAP will just be plugged into the laptop and no powering is necessary.

Section 5.9 – *Final Third Subsystem (Ground Station System)*

The ground station design has changed due to complications in getting the PIC microcontroller to communicate to the Atmel microcontroller through the Nordic nRF24L01+. We decided to ditch the Bluetooth module and thus removing the Android Phone application as a secondary controller since the cost of the Bluetooth module was $50 and the added programming and testing of the Android Phone Application proved to be too much to handle.

The final ground station design to communicate between the computer and the blimp was changed to use the Atmel ATmega328p microcontroller. This microcontroller was also used on the Blimp Controller Board so it was a quick redesign in making them communicate with the transceiver. The nRF24L01+ had some trouble switching between transmit and receive still because of the IRQ pin, so we decided to change it to the XBee Pro 2.4 GHz transmitter. The USB connection to the computer stayed the same though since it was just using the UART pins and had no problems sending or receiving, we kept the CP2102 UART to USB module in the design.

Section 5.9.1 – *Microcontroller – Atmel ATmega328p*

The Atmel ATmega328p is the same microcontroller used in the Blimp Controller System. We have to connect the power using a 5 volt rail and we are achieving this using the LM2940 9 volt to 5 volt linear voltage regulator. Then we also have to connect the 16 MHz crystal oscillator or resonator to the pins 8 and 9 on the microcontroller. The serial communication pins we use are the hardware UART pins 2 and 3 for the CP2102, and then we have to use software UART to connect the XBee Pro transceiver which is on Pin 13 and 14.

Section 5.9.2 – *UART to USB Module – CP2102*

The CP2102 module is used to connect the microcontroller data stream to the computer's USB port. The module has pins to provide 5 volts and 3.3 volts out to power anything on the board from the computer but we do not use these as we found the Linear Voltage Regulators a more stable source of power. The USB module sends and receives data to and from the computer just fine so we kept it from the old ground station design. The nice thing about these modules is that it does not require an external 48 MHz crystal on the microcontroller or a dedication of 48 Mhz for the full USB transfer speed, it is instead all built onto the CP2102 module.

Section 5.9.3 – *XBee Pro 2.4 GHz Transceiver*

The XBee Pro 2.4 Ghz Transceivers are basically plug and play modules when used with the Arduino Programming on the Atmel ATmega328p microcontrollers. The XBee modules use the serial communication line (either hardware or software UART serial work fine) to send and receive data. When we switched over to the XBee we saw the specs and it actually boast a high maximum range at 1500 meters than the 1000 meter range nRF24L01+s. **Table 5.9.1** below shows the major differences between the two modules. Some things to note between the two modules are that the prices of the XBees are higher and that the power consumptions of the XBee modules are over 22 times higher than the nRF24L01+. The XBee was not ideal for our transceiver but it works better than the nRF24L01 at sending and receiving data.

	Frequency	Antenna	Weight	Power Consumption	Max Range	Data Rate	Price
nRF24L01+	2.4Ghz	External SMA	4.3 grams (no Antenna)	11.3mA TX Mode 13.3mA RX Mode	1000 meters	250kbps, 1Mbps or 2Mbps	$19.84
XBee 60 mW PRO	2.4Ghz	On-Chip	5.7 grams	3.3V @ 250mA	1500 Meters	250kbps Max data rate with 128-bit encryption	$ 37.95

Table 5.9.1: Comparison of Old vs. New Transceiver

Section 5.10 – *Final Graphical User Interface (C# Programming)*

For the final design of the GUI, we tried to keep most of the features the same. We have a human detection on the video feed but we had to change this to a one person on the screen at a time for the tracking feature to correctly follow them. We also implemented a map where the user can click for the auto-pathing feature. We had to change the programming language from C++ to C# in order to use the wrapper for OpenCV called EmguCV, using this library enabled us to access all the OpenCV library as well as keep the many interfaces and GUI designer features that make Visual Studios great at creating a User Interface.

Section 5.10.1 – *Emgu CV Features*

EmguCV is a wrapper library for OpenCV which installs quite easily onto the hard drive and can be configured for 32-bit or 64-bit program that want to use the Computer Vision libraries. We use this library for its HOGdescriptor which is the pedestrian detector for the OpenCV library. The people detector has quite a lot of false positives in detecting the people in the screen. There are many shadows of objects around that the human detector thinks is an actual person. The requirements for detecting a person on the screen is quite simplistic, it only requires, the head, arms, and feet to be on the screen for the detector to high light them in the video feed.

When we tried to improve the person detector using a mix of color detection and blob detection we got a lot of false positives in the screen because when looking at people from high above we get more shadows on the ground and the amount of different colors on the screen also increases. The blob detection was set in two different shapes, the overhead view of a body, which is mostly a oval shape with arms extending out, and the full body view that the HOGdescriptor is set to.

Because of the false positive rate and the already high processing power required to detect a person we had to abandon the "trained" version of the human detector and go back to the default HOGdescriptor. A problem with the HOGdescriptor is that it won't detect people smaller than 128x64, so we had to change this setting for our views of people at a higher altitude. We wanted to make this change dynamically according to how far up the blimp was but we could not fill up the blimp in time and send it up there while testing out this camera feed, other buildings on campus could not give us the constant variation we needed to test out the feature.

Section 5.10.2 – Mapping and Auto-Pathing

The mapping library we switched to is called GMaps.NET which implements maps from a great deal of sources such as Bing, Yahoo, and OpenStreetMaps. With this library we can plot points using GPS coordinates, refocus the map to a specific location, and draw a path using GPS coordinates. We were playing around with drawing the blimp path and have a line going from one point to the next that the blimp flies to, but with testing we found it cluttered the screen too much and it looked much nicer if the blimp's GPS dot on the screen was just updated to its next location all the time.

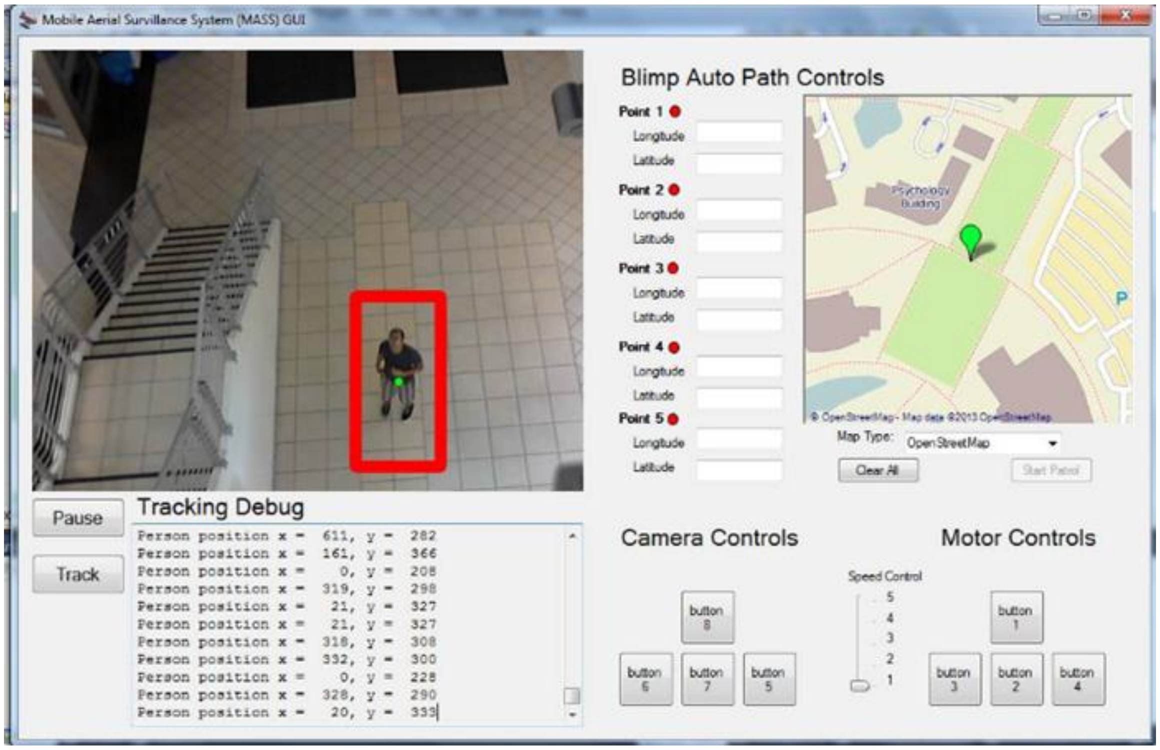
Section 5.10.3 – Movement Controls

For the movement controls from the GUI for the camera servos and the brushless motors, we are limiting it to be only being able to be controlled by the Button on the screen. We eliminated the Logitech Controller from the design because mapping out the analog joystick was not working out too well. The precision was not good and it was sending commands too often. Sending commands too often to the transmitter buffer was becoming a real problem for us, so we wanted to limit it by the user slowly clicking on the screen to move the blimp or the turn the camera. This also eliminated the need to program what happens if the user clicks on both the up and the left arrow at the same time since on the screen the user can only click one button at a time. The user can still rapidly click one button and ruin the transmission string but with us controlling and demoing the system it should not be a problem with a slow moving blimp receiving the slow turning or rising/falling commands for the blimp.

Section 5.10.4 – Computer System

The computer that was used to connect to the ground station's perforated board was the Toshiba Satellite L505. The Toshiba Satellite has 2GB of RAM and 288 GB of Memory. The vast amounts of memory were used to write and compile the code for the ground station and the blimp. The Toshiba was also used for testing and serially sending data to and from the GUI. The ground station was connected to the Toshiba was connected by a USB to UART converter. The computer was used to test the OpenCV libraries for the blob detection. Also connected to the Toshiba Satellite is the EZCap video capture card. This is used for receiving analog video signals from the receiver, and converting them into MPEG4 video format.

Figure 5.10.1 below shows the final layout (excluding the pictures on the directional buttons) of the GUI design as well as one of our group members being detected from the third floor of the Harris Building.



Map Type: OpenStreetMap

```
Person position x =  611, y =  282
Person position x =  161, y =  366
Person position x =    0, y =  208
Person position x =  319, y =  298
Person position x =   21, y =  327
Person position x =   21, y =  327
Person position x =  318, y =  308
Person position x =  332, y =  300
Person position x =    0, y =  228
Person position x =  328, y =  290
Person position x =   20, y =  333
```

Figure 5.10.1: MASS User Interface (Contains Video Feed, Control Buttons, and Auto-Pathing Controls)

Section 6: Design Summary of Hardware and Software

In this section we go over the subsystem design summary of the camera system and the ground station. The camera system will have an auto-stabilization system. So it is important we pick the right camera for it as there might be a lot of turbulence or shacking of the blimp as it flies up and around. Then the ground station schematic has to be laid out and the program on the microcontroller has to connect to the blimp microcontroller to take advantage of all the features we plan on implementing.

Section 6.1 *Blimp Design Summary*

The purpose of the Blimp controller is to gather all the data in the air and either use the information for auto-routing on the blimp itself or to send back the information to the computer so the user can see the positioning, speed, or orientation of the blimp. The hardware we picked has low noise in its sensor data, so we hope that we can get a clear reading and it won't jump with anything more than 5% error. The GPS is the most worrying part as in order to get a very accurate reading, we have to buy a very expensive or military grade GPS module, but being within a couple of meters hopefully does not affect the positioning much. The transmitter on the Blimp is the other side of the one on the ground station and will transmit 1000 meters away, but we only expect to get at most half of this range even transmitting at the lowest speed. The blimp design that goes in the carriage under the blimp was designed to be light weight but more so in low cost design. The amount of lift we get with the balloon size being 4 feet by 10 feet makes the weight not so important, but we still can't put too much on there.

Section 6.2 *Camera System Summary*

What we needed was a camera that was first and foremost light weight. Ideally it would be inexpensive, Wi-Fi compatible or an internet protocol camera, and have inferred technology so the blimp could be used at night. There were three cameras that fit the specifications that was needed: Gore Hero Black Edition, Panasonic - V201 HD Camcorder, and the Canon - VIXIA HF R42 32GB HD Flash Memory Camcorder. Each camera had their pros and cons and will run a few tables to display the differences and pinpoint exactly why we chose the camera we did.

Camera	Gore	PANASONIC	CANON
Price	$240 (with 40% discount)	$229	$499
Weight	7.2 ounces	6.5 ounces	8.3 ounces
Wi-Fi Compatible	Yes	Has Eye-Fi*	Yes
IR compatible	Yes with 109 dollar part	Yes	No
Resolution	1080p 60fps	1080 60fps	1080p 60fps
Battery Life	90 Minutes	135 Minutes	90 Minutes
Optical Zoom	0X	38X	52X

Table 6.2.1: *A way to compare all of the potential camera candidates in a quick way*

**Eye-Fi for the Panasonic is a wireless card that comes with a CD that has to be uploaded onto the computer and will work as a Wi-Fi substitute as the name would suggest.*

All three cameras would have sufficed for this project. Each had their qualities that made them more desirable for us, especially the time difference in terms of battery life for the Panasonic along with price tag. Or things that were a little more negative like the Canon ended up being a little too expensive for our budget and didn’t have enough specifications that made it stand out for that price tag. All 3 cameras were really surprising in the fact of how light they were. The LCD screen on the Canon and Panasonic made them a little more expensive and that was a useless feature considering the camera would be would be mounted fifty feet in the air on a blimp. The Eye-Fi from Panasonic sounded a little sketchy and without it being Wi-Fi compatible it made it a lot less desirable as well. The Canon was really a great camera but a lot of this project will be based on money decisions and this is one of those times. If the group was going to spend 500 dollars on a camera it would really need to have more optional features and special effects especially when the GoPro had so many things we liked. Not being waterproof also hurt the two other cameras mainly because a surveillance blimp should be able to go outside into most weather conditions(not including a hurricane of course) and be able to take video and monitor the area. With it not having a waterproof housing the camera could get caught in a rainstorm and that money we just spent on a new camera would be wasted. The GoPro normally retails for 399 dollars, but one of our group members sells them and gets a 40 percent discount on them. The remote control was a nice feature which would be slightly less work for us to program and the included mounts was easily the best selling point for the GoPro in comparison with the other cameras.

The GoPro in the end had a few too many positives to not end up choosing it. The GoPro will have to be ordered specially through his company so it will take an extra week or two to ship but that will be well worth the savings. The GoPro Hero 3 black edition is currently the bestselling action sports camera on the market and it's really easy to see why. Now to get down to the specifics of the camera and all of positives that made us choose it, as well as the negatives that we will have to overcome.

GoPro Hero 3 Black Edition

The camera is withheld inside a waterproof polycarbonate casing that is lightweight and can be removed by unhooking the black latch at the top of the camera. The camera is currently sitting on a mount that clicks into the stand and the mount will be very important for how we connect our camera to the carrier on our Blimp. The mount will be permanently affixed to our carrier but the camera can easily be removed by pressing the clips closed and removing the camera. For extra security the camera also has a screw to ensure no movement in the pan or tilt direction. The camera when dressed with the waterproof casing is able to go 60 meters deep which is perfect for our blimp unless it decides to become a submarine. The sleek and stylish camera only weighs 7.2 ounces and turned it to be the best camera we could find.

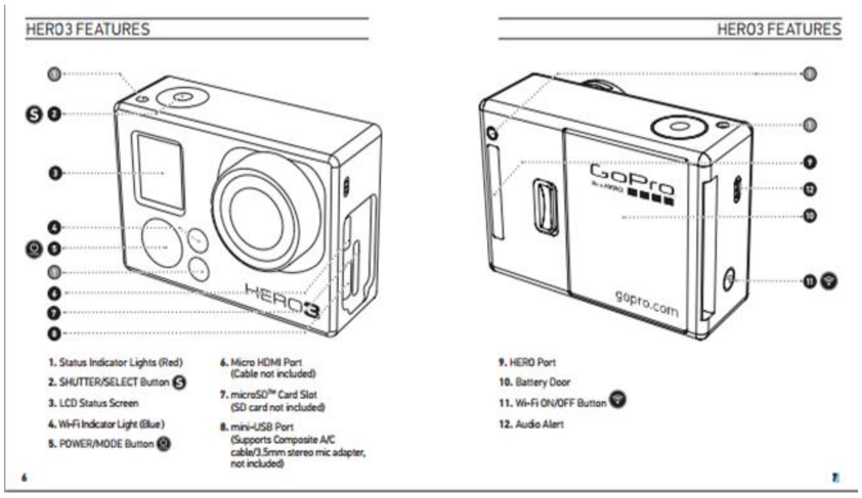

HERO3 FEATURES
HERO3 FEATURES
HERO3
GoPro
gopro.com
1. Status Indicator Lights (Red)
2. SHUTTER/SELECT Button
3. LCD Status Screen
4. Wi-Fi Indicator Light (Blue)
5. POWER/MODE Button
6. Micro HDMI Port (Cable not included)
7. microSD™ Card Slot (SD card not included)
8. mini-USB Port (Supports Composite A/C cable/3.5mm stereo mic adapter, not included)
9. HERO Port
10. Battery Door
11. Wi-Fi ON/OFF Button
12. Audio Alert
6
7

Figure 6.1.1*: GoPro Hero 3 features and diagram of what button does and corresponds too. Reprinted with Permission by GoPro*

Figure 6.1.1 above pictured is the actual camera outside of its casing and all of the input/output items it possesses. The camera has all of the typical buttons and ports that most cameras would have; mainly power, shutter button, HDMI port, Microcard port and a Wi-Fi indicator light. The Black edition can shoot in multiple resolutions ranging from 4k all the way down to 720p. The 4k it shoots in only can handle 15fps which is normally to slow for most action sports in which the camera is designed for but for us it would be ok. The higher resolution would waste the battery life quicker so in retrospect we would probably use a lower resolution with a higher frame per second to save battery life. 1080p and 60 fps seems to be the normal shooting resolution for most cameras and we will use the same.

A cool feature that the has it that it can overwrite automatically what it saved on the SD card if the loop feature is selected which means we wouldn't have to return the camera to the ground every time the card was full. The micro SD card can be up to 64 GB in size. Obviously the larger the more expensive and seeing as the camera only last 90 minutes before dying we probably will not need such a large SD card. The SD card unfortunately will have to be bought separately but they tend to be very cheap in general so a good amount of memory for 20 dollars should be very easy to find. All local electronic stores or even a Wal-Mart would have them for a reasonable price. A 32 GB would probably be best due to the fact that the battery life is marginal at best when dealing with the GoPro Hero 3 black edition. The Wi-Fi will also cause the battery to die quickly and if the LCD touch backpack is purchased as well that will drop the 90 minute runtime to approximately 60 minutes.

The camera also comes with some assorted accessories. These accessories include the following: an LCD touch backpack which is 80 dollars, Wi-Fi remote control, 3 way pivot arm, extended arm, frame, one flat mount, and one curved mount. One of our secondary objectives was to control the blimp with an Android phone and seeing as the GoPro has an APP that directly links to the Android phone this might make this task slightly less strenuous. The range for the APP is roughly 100 yards which would more than suffice the 50 foot goal we had previously set up. The camera will be able to run a program that will be able to use colors to identify and track certain objects and people. By using color recognition program the camera will notice that color and follow it until it goes out of focus for the camera. This is our way of tracking because a surveillance camera that cannot track is not a very productive thing to have. Our camera will be controlled a Wi-Fi remote that can work from the ground so we can turn on the camera whenever we want or we can use the Android app that comes is downloaded from the GoPro website. The camera also can send a direct Wi-Fi feed to our phones so no direct IP is needed in terms of streaming.

The only drawbacks to the camera are the facts that it is not Infrared ready and has a short battery life. This would make our blimp completely useless at night but there is an extension to the lens that we can buy that will ensure we can use our blimp at night. It is a little on the expensive side retailing for 109 dollars but will can add a necessary instrument to our surveillance blimp. The battery time for the camera is 90 minutes, which is not spectacular and if anything would like to be improved which is possible with the extra battery backpack which is an extra 50 dollars and will add some weight. The weight would not be substantial enough to not buy the backpack but the dollar value will probably make our group talk about how long we really need the blimp to be up in the air. Ideally the blimp would fly for at least 8 hours as if it were working a full security guard shift. We want to make it as marketable as possible to make it enticing to people who would like to have their own surveillance blimp. All GoPro videos can be uploaded and edited on the computer as well.

Overall in terms of performance price and specifications this camera is definitely the best camera we could get. It shots in high performance 1080p which is equal to high definition. It can take pictures while in video mode with its 12 megapixel camera which is equal in quality to much more expensive cameras. It has a high color quality optic control which enhances colors and makes a more noticeable difference in pictures. The extremely lightweight of the camera combined with the fact it has a waterproof housing that is also lightweight really just made this camera stand out a lot more and that's why we choose it. Although the discount we had on the camera did not hurt its chances either. Like it was stated previously we have to order and wait for the camera to come in but as soon as it does we can start testing our streaming and actually see what it's like to ride on a makeshift RC Surveillance Blimp.

Section 6.3 Ground Station Design Summary

The purpose of the ground station is to act as a base station for all communication to the blimp and to the controlling programs. We want to interact with a computer program and an android application, so we need a USB connection for the computer and a Bluetooth connection to interface with the Android phone.

Section 6.3.1 Schematic Diagram

The **Diagram 6.3.1** below shows all the components connected together as well as the pins that are used. The pins that are not used are not displayed to save space in the pictures. There will be revisions to this design as we get the parts and test it out. A larger version of these schematics is available on **Appendix F**.

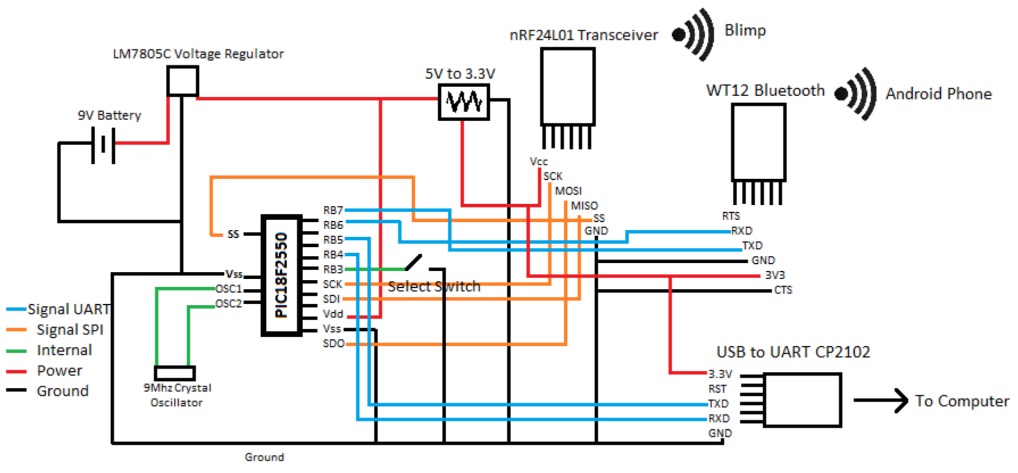


Diagram 6.3.1 *- Circuit Layout for Ground Station Design*

6.3.2 Ground Station Program

For the ground station we need to program the PIC microcontroller to use all the I/O pins of our transmission modules in Microchip MPLabX with the C18 programming language. There is one issue with the UART connection and we need to learn how to use the multiplexer select line with the PIC and there should be no problem with the serial protocol we send to the two devices on the multiplexer since they both use UART. But, we have recently found a solution to this problem by turning the unused pin 28 and 27 pin on the PIC18F2550 to mimic a UART connection (use TTL logic) to transmit (TX - pin 28) and receive data (RX - pin 27). This solution is a software solution where we set the pins virtually to mimic UART and we have to look into the C18 programming language for MPLabX to figure out the exact commands and pin setup.

nRF24L01 Header File

RF24_initPorts() - This method initializes the pins to output or input on the PIC18F2550 and sets up the SPI pins for the nRF24L01 transceiver.
RF24_default_config() - This method configures the packet size and the speed of transmission, we will change the default configuration to use baud rate of 115200, data bits at 8 bits, and hopefully achieve the 2 Mbps on-air speed.
RF24_TX_SET() - This method changes the nRF24L01 to transmission (TX) mode, since we cannot transmit and receive at the same time, we need to use this method to send data out the TX pin.
RF24_RX_SET() - This method changes the nRF24L01 to receiving (RX) mode, we need to use this method to have the transceiver listen to the airwave for data packets.

PIC18F2550 Methods

PinInit() - This method sets the direction (input or output) for the pins corresponding the RA through RC ports.
SwitchSel() - This method listens in on the Pin where the switch is connected to and switches the transmission from Computer’s USB connection to the Wireless Bluetooth (or Vise Versa).
DataSetup() - This method sets up the 8 Mhz oscillator and FOSC setting to output the data at 2Mbps. This will also set the baud rate, packet size, and buffering for the UART connections to the Bluetooth and USB connections.
Data_Send() - This method uses SerOut to change the pin to use TTL logic and send data from the Blimp to the computer GUI or android phone. Switches ground station transceiver to Receive (RX) Mode.
Data_Rec() - This method uses SerIn to change the pin to use TTL logic and then receives data from the computer GUI or android phone and buffers it until all

of it is received, then it will transmit the data to the Blimp. This switches transceiver to Transmit (TX) Mode.

The **Diagram 6.3.2** below shows the flow of the ground station program and how it should direct traffic as well as some of the method that is related to each module.

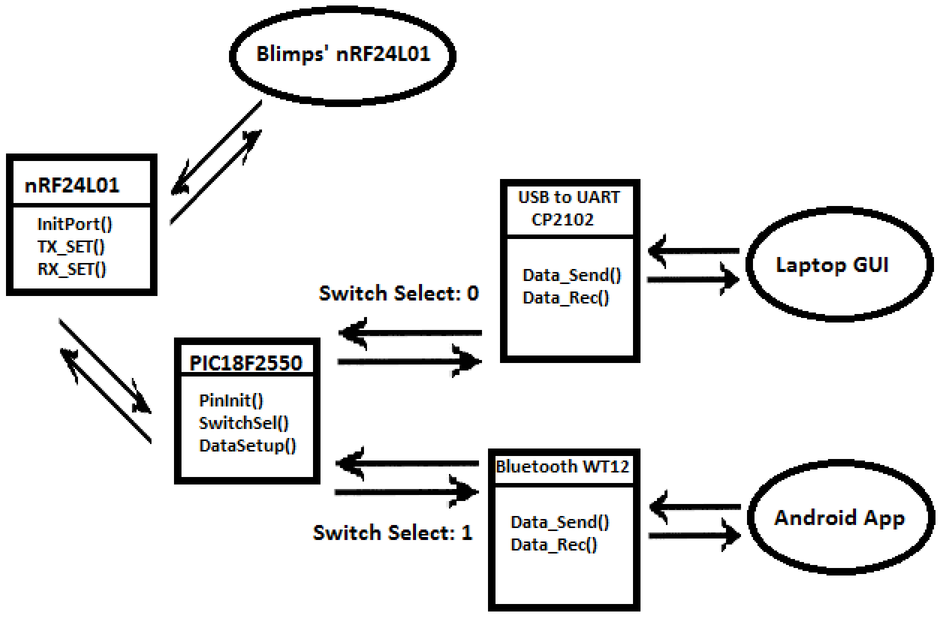


***Diagram 6.3.2** - Flow Layout of Ground Station Program*

Section 6.4 Software Summary

In this section the program on the microcontroller for the Blimp Controller Board and the program on the computer/android phone are explained. The main function of the ground station is to receive incoming data and either spit it out to the computer GUI or to the Android phone. The computer and the phone are responsible for recognizing the timing of the transmission and how to process it.

Section 6.4.1 Low Level Programming (Microcontroller Programming)

The blimps' microcontroller, the Atmel ATmega 128, will be programmed using the Atmel programmer. The software on the Atmel process needs the libraries for the nRF24L01 transceiver; we will create our own programs and methods to receive the GPS NMEA codes and the Accelerometer values. The motor controls will most likely be created by us unless we can find the libraries for the speed controller IC unit and the servo motors turning the propeller motors. For the Atmel ATmega 128, we need to figure out how to set the Ports A through D to either input or output and be able to connect the sensor and motor modules to these pins. The program for taking in all this data will work like a finite state machine with interrupts that can happen at any point. The interrupts can tell the blimp to either go to auto-pilot mode or to request a ping from it so the ground station and the blimp know that they are still connected to each other. The main program will start by taking sensor values and sending them to the ground station, and depending on either the sensor values or the controller connected to the ground station it will tell the camera system to move or the motor system to start moving.

The ground stations' microcontroller, the PIC18F2550, will be programmed on MPLabX using the PicKit2. The program of the ground station basically just acts as an intermediate device between the blimp and the controller, this ground station will redirect signals to either the GUI or the android phone application. The most important programming portion is to make sure that the GUI and the android phone cannot send data to the blimp at the same time. A second important feature is that the ground station will control the RX and TX modes as well as the speeds at which the blimp will be send data and the speeds at which the computer or android phone will be send data, the maximum transmission speed for any of the devices will be 2Mbps.

Section 6.4.1 High Level Programming (GUI and Android Programming)

The GUI will be programmed in Visual Studio 2010 Professional and the language it will be coded in is C++. The GUI program will work as a controller device as well a video output so that user can recognize which objects on the screen are people, then they will be able to select the people to make the blimp follow them. The video processing will be done through the OpenCV library and this will provide us with the people detection and the overlay on the video feed. The program will send out data to the ground station using the serial library for C++ with a certain string to define if it is either a controller input or a automatic movement input. There will be on screen controller button presses and the feature to connect a Logitech F310 controller to control the blimp manually. Then we will have an OpenStreetMap that will display the area the blimp is currently in and then be able to set the blimp to autopilot mode where it just circles the area that the user has selected. In **Figure 6.4.1** below, it shows the flow of the program and how each class interacts with each other. The methods listed are covered under the GUI and Android application design sections.

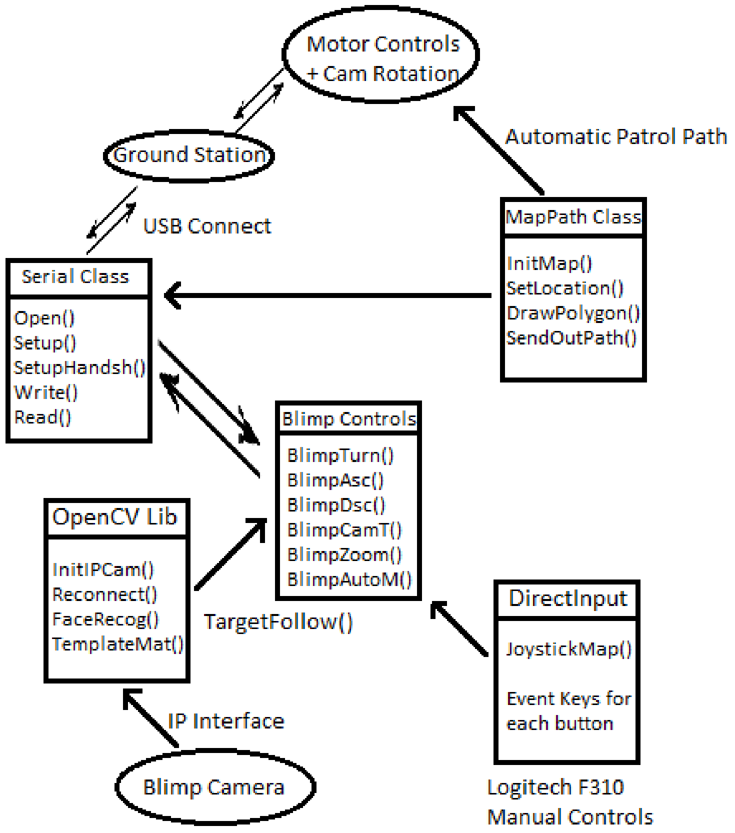


Figure 6.4.1 - Flow/Class Diagram of Computer GUI

The android programming will be done on the Android SDK programming environment. The coding style used in android coding is a mix of XML and Java. The android application that we create will use the OpenCV library since it supports android programming, this will control the video feed from the IP camera like in the computer GUI. The video feed will cover the whole screen of the android phone and will have an overlay for the joystick objects that will mimic the Logitech F310 layout. This android phone application will communicate to the ground station through a Bluetooth module. In **Figure 6.4.2** below, it shows the classes that are going to be created in the Android Application and the flow of the program.

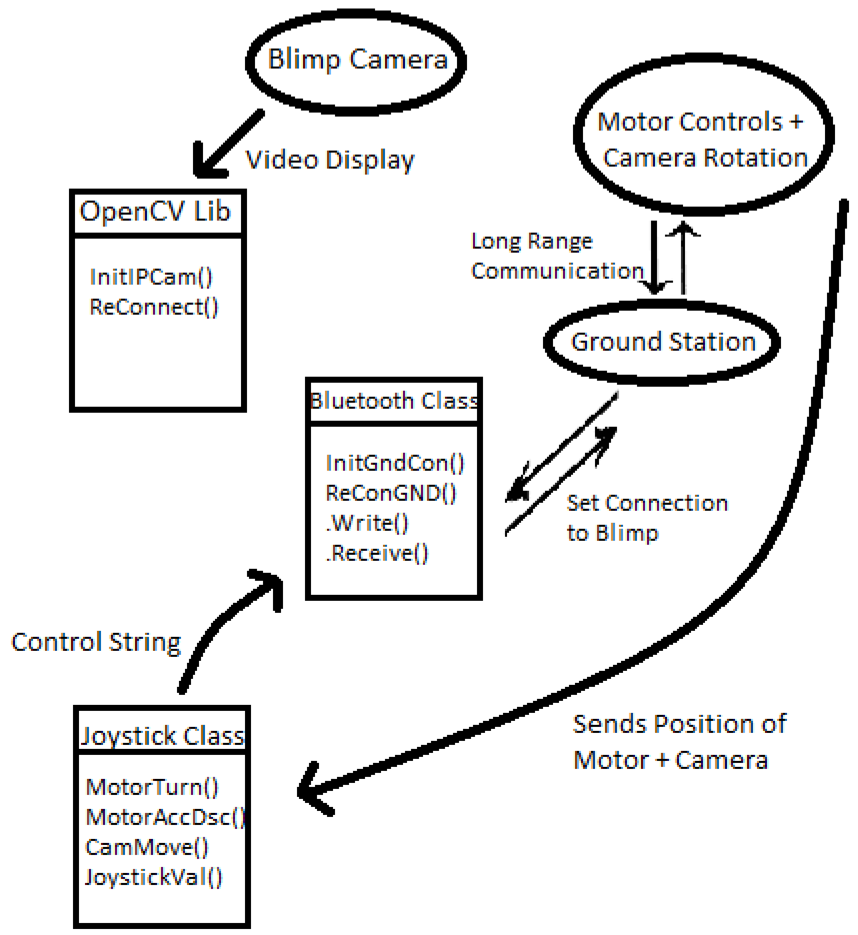


Figure 6.4.2 - Flow/Class Diagram for Android Application

Section 6.5 *Final Blimp Design Summary*

The final blimp code for the blimps onboard microcontroller was written after all of the coding and testing was done for all of the peripheral modules. Generally speaking, the inputs for the blimp were serial commands to the microcontroller. The outputs from the microcontroller would be GPS and IMU data transmitted to the ground station as well as motor and servo controls.

To get the final program together, the motor controls were the first thing to be coded up and tested. Ultimately this is the only output from the microcontroller that really mattered. To get the motors running the Electronic Speed Controllers needed to be tested and controlled. The ESCs give current to the motors only when they have been armed and when they have been given the proper signals from a RC transmitter. Through pulse width modulation and coding, the microcontroller could emulate the signals from a transmitter and control the motors! Through testing, it was found that the ESC will only work once, the arming sequence was sent. This was a PWM with a duty cycle of zero percent was sent for five seconds. After that, it was found that the ESCs would only give current to the motors if there was a duty cycle of greater than 55 percent and any higher than 70 percent would not increase the speed of the motors at all. In code it was decided that there would be 4 speed settings and they would all be increments from 55 to 70 percent duty cycles. There were also functions put in to reset the motors and call the arming sequence in case the motors lost the same revolutions per minute. Functions were written to turn the blimp left and right. They were tested extensively in code and in the air. These functions would be called when the microcontroller would read in the proper 8 character string.

Similarly, the code for the servo motors was tested in a similar vein. The motors would connect to the PWM sending digital pins and would be controlled when the user or the auto control would tell it to. This code was tested by cycling through the various degrees of motion.

Written in the blimps final code is the perfected GPS code. To get the final product, the testers needed to read in the data from the satellites by calling the proper the NMEA codes and using the Arduino libraries for the LOCOSYS GPS module. Besides from the longitude, latitude, and altitude other functions could be called and the data would be sent from satellites if they were called properly. Some of the data other data that we actually used in the final code was the trajectory and the speed functions. These input data would be read into the microcontroller only when the user specified that the blimp was in Auto control. The microcontroller would then read in the GPS coordinates every 4 seconds and compare with the users chosen GPS point.

To get the input data from the IMU, extensive testing was surprisingly not needed. The magnetometer built in was surprisingly stable. The gyroscope and the accelerometer was very noisy, but essentially were not used at all. There was code to act as a low pass filter but it was not used in the final code. To use this Wire library was used from Arduino.

The microcontroller was simply connected to the XBEE through digital I/O pins and then it worked out of the box basically. The testing for the XBEE was done by gradually increasing the distance and testing the lag on the transmitters. The code used the SoftwareSerial library from the Arduino library.

Section 6.6 *Final Camera System Design Summary*

Design for the camera system was pretty much the same as the old design except using new camera and a different mount for the stabilization system. The design was to be made simple and lightweight. Thin plated metal mixed with two mini servos distinctly gave the requirements desired for the system. It was put together by screws, self-tapping screws and nuts. The system is also user friendly seeing as almost any servo size can be used in case they are broken during installation. The implementation for the camera video feed will go from the camera to the transmitter through AV cords. Then the transmitter will send it to the receiver which will then send the signal to the EZ CAP. This video capture card turns the signal digital so it is readable on a laptop. Finally VLC makes the program runnable with OpenCV and the video feed is completely streamed. The implementation of the stabilization system involves having user controls or automated controls which run through the microprocessor to tell the servos which way to pan or tilt. The user will press a button on the joystick and the GUI will transmit the data to the transmitter and the microcontroller will then tell the servo motors to move accordingly.

Section 6.7 *Final Ground Station Design Summary*

The final ground station design changed to only having a transceiver and a USB module attached to it because the price of the Bluetooth module and the time it would take to develop the Android application with its own testing phase. We switched to the Atmel ATmega328p microcontroller in order to better communicate with the Blimp Controller Board from the Ground Station. We chose to use the XBee Pro 2.4 GHz transceiver instead of the nRF24L01+ because we were having address problems and could not get the board on the blimp to communicate with the ground station transceiver. Matching the address and channel of communication between the two still had no communication but when the boards had their own code between the same PIC or between the same ATmega it would communicate just fine. So we decided to just make it the same microcontroller.

The Ground Station is still used to send movement data from the user controlled Graphical User Interface (GUI). But the Ground Station can also receive location data from the Blimp so the GUI can process it and find out the Blimp's GPS location to display on the auto-path map. Receiving accelerometer and gyroscope data from the blimp is still up in the air and has not fully been implemented due to dirty data and being unable to clean up the readings.

Section 6.8 *Final C# GUI Software Summary*

The Graphical User Interface (GUI) that we have the user use to control the blimp and view the surveillance system has changed from using a C++ coding style to a C# coding style. We had to change since the EmguCV wrapper for OpenCV was much friendlier in creating buttons and interfaces for the user to use. The computer visions part of the blimp is able to detect people on certain conditions such as requiring the GUI to see the person's head, arms, and legs in the camera view before it will box them off and giving their center location. Any attempts made to improve the person detection feature provided to increase the false positive rate and gave the center location at many different locations on the screen. So in a perfect detection, it would only box off 1 person on the screen and try to track that person as they move on the screen. The GUI still has all the features of trying to track the person in the camera view and has a map to display the blimps current location and tries to map out the path that blimp should go to when the user initiates the auto-pathing system. There are button controls on the GUI screen for the user to press in order to turn the blimp's motors or make it go up and down. The buttons are split up between camera controls and motor controls and has a slider for the throttle which gives the user 5 motor speeds (3 of them being pretty similar in speed).

Section 7: Project Prototype and Construction

The project prototype will be done on perforated boards for the ground station and printed circuit boards (PCB) ordered from Advanced Circuits for the blimp controller board. The blimp controller board cannot be designed on the perforated board unless we buy a lot of extra break out boards so we plan to have the PCB design done as soon as possible so we can attach all the modules such as the accelerometer, global positioning system (GPS), and the gyroscope. Then we will start programming direction on the PCB and test things out. The ground station board on the other hand will be first bread boarded, then perforated boarded, and if we have the time and money, we will send out the complete and tested design to become a printed circuit board.

Section 7.1 – *Parts Acquisition and BOM*

The parts that we need to acquire mostly will come from online ordering, but there are also a couple of hobby shops around the Orlando Area that we can get parts quickly from. Looking at the parts at the hobby shop some fit our needs such as brushless motors, development boards, servo motors, and various cables. But they are more expensive here and we think we can wait a week on the orders from online. Main components such as resistors, capacitors, crystals, and other board components will come from online suppliers such as Digikey or Mouser Electronics, but as a backup, we have Skycraft as a local source so we don't have to wait around again if we missed some components.

Some of the parts we are ordering over the break are parts from China, these are the cheapest parts, but they also take the longest to arrive here. So we are ordering these cheap parts as backups as it would not cost us much money. Hobby King is also one of our main suppliers as an online supplier but they recently opened a US warehouse so we are ordering from there. Some parts might be unavailable so it would be coming from the China warehouse, and we are ordering those parts as soon as possible also.

Some of the common components we have learned that we can grab from the Senior Design Lab, such as resistors and capacitors, but the University does not supply the crystals, Linear Voltage Regulators, or LEDs we need. In the lab, we can rent out Soldering Irons, Breadboards, Wire Cutters, and Wires (to keep). So we are planning to fully take advantage of this.

Section 7.2 – PCB Vendor and Assembly

For the Printed Circuit Boards (PCB) vendor, we are going to order our PCB through Advanced Circuits on the 4pcb.com website. This site was recommended by Dr. Richie since they offer our school the PCB at a discounted price of $33 per board and you can order only 1 board at a time if need be. Looking at the site they also have the same deal but at a 4 board minimum, so it is great that we don't have to order the board at 4 per schematic and possibly have wasted boards. We plan to have the schematic done and checked over by everyone in the group to make sure we have all the connections done and then we will move onto switching that schematic to a PCB file design on the Eagle Software.

The Printed Circuit Board is for the Blimp Controller Board which will house connections to all the modules (Accelerometer, GPS, and Gyroscope), Servo Motors, Brushless Motors, and the transmitters. We also plan to house the power system for the Blimp's Microcontroller on the PCB, which will include either switching regulators or linear ones. The printed circuit board will mostly consist of header pins for the servo motors and brushless motors, but the modules such as GPS, are all surface mount parts and require a direct connection to the printed circuit board.

We are expecting failures in the assembly of the PCB and we are going to send out a second design and maybe even a third design to be made. The most troubling part of this is the time it takes to receive the PCB back from the Vendor. We expect it take a little over a week to receive and if need be we are willing to pay extra for the overnight or 2 day shipping. We are thinking about connecting extra wires on the PCB to have the extra GPIO pins for us to use in case we need them if we switch modules. This will only be for testing, because we might have to cut traces and reconnect them with wires. If this is the case and the board comes out too messy we will order another PCB design from that.

We are planning to do all the assembly ourselves, such as putting in the resistors, capacitors, headers, and modules onto the board. Although no one in the group has experience in surface mount soldering we are planning to learn from another group or from the Radio Club that is in the back of the Senior Design Lab. We have experience with through-hole soldering and we are planning on having everyone in the group brush up or learn how to solder so we don't mess up the perforated board or the Printed Circuit Board (PCB)

Section 7.3 – *Eagle PCB Designs*

As every project has to have a PCB design this one was no different. This PCB design was actually very easy considering all of the components were going to be put on male or female header pins. The Eagle design had only a handful of components including an XBEE transmitter, a Locosys GPS Module, Atmega 328 microcontroller and IMU module. The PCB will only be using two layers with very limited and simple connections. **Figure 7.3.1** below shows the Schematic capture done in eagle for the Blimp Controller Board. (Larger Version in **Appendix C**)

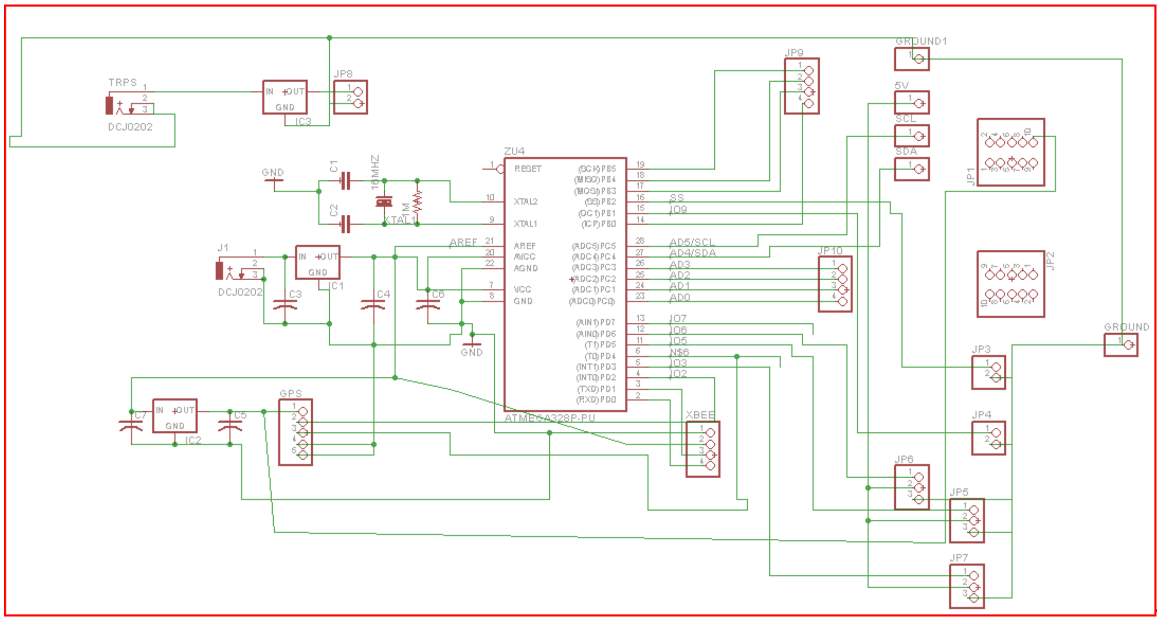


Figure 7.3.1 – Schematic Capture of Blimp Controller Board

The design of the schematic was decided to set the microcontroller (Atmega328) in the middle of the board in order to place everything around it in order to clean up the schematic. The ATmega had numerous important ports as expected which will just be run through real quick. The reset button was left untouched due to the fact that this PCB will be on the blimp and no way to press the button. The XTAL1 and XTAL2 ports were used with an external 16 MHz oscillator. All PWM signals were used with the servos and electronic speed controls. Everything else that was unused was plugged into a female header just in case something happened and a pin was needed. The first ordered PCB had this problem of not having all the pins needed and grounding unused pins which ended up ruining the PCB. The PCB has two power jacks connected to it, one for the camera transmitter and one to power the PCB. There is also a place for the 5volt out from the electronic speed controller which will help power the PCB. There are a couple of 9 to 5 Volt LDO regulators that will step down the voltage and a 5 to 3.3 volt regulator to power all of the components that run at 3.3 volts. The servos use three pin headers while the motor controllers use two pin headers. All power supplies are regulated by capacitors and grounded appropriately. . The microcontroller uses DIP packaging and will have 28 pins. Most of the pins will be unused but were hooked up to open headers just in case one of them is needed. Most of the unused pins are ADC pins. The actual board will be 3.26” by 3.93”. This will fit into the gondola perfectly and leave more space for other components.

The final PCB design is just the schematic put into board form, the following pictures will show some of the solder masks and the actual connections made for the PCB.

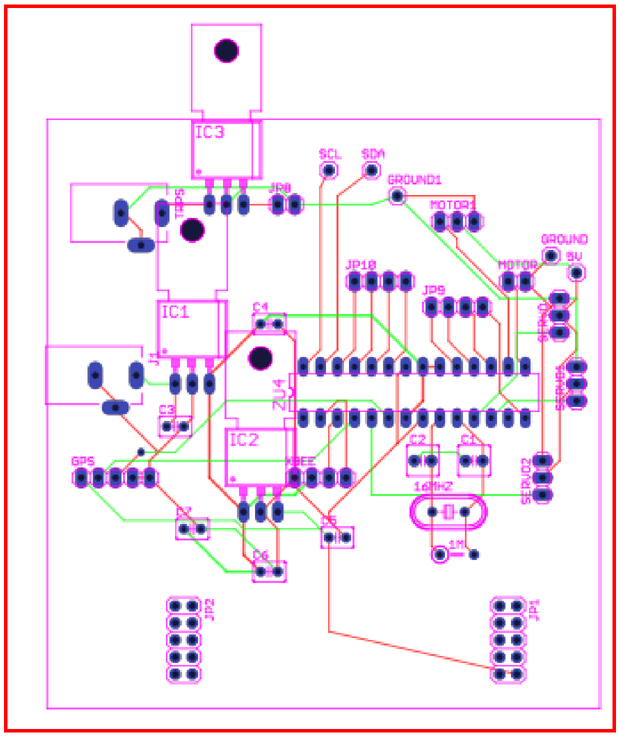


Figure 7.3.2 – PCB Layout of Blimp Controller Board

The actual board design was mainly done through auto routing. The pieces were placed logically next to the devices they were connected with and then auto routing did the rest of the traces. Some necessary clean-up had to happen in order to make sure the board passed all of the checks set forward from EAGLE and 4PCB.com which is where it was ordered from. 4PCB.com has a DFM partner that checks to make sure all board components make sense and there is enough spacing. After the board passed it was purchased for 33 dollars plus 40 dollars for shipping. The actual placement of the PCB will go into the gondola which will hold almost all of the other pieces it will connect to or be connected too. The PCB being the most delicate will have a little shelving to ensure it remains safe throughout the flying of the blimp. The PCB is still in testing mode to make sure everything works according to plan. All pieces are soldered on and are tested with the blimp in motion.

Figure 7.3.3 shows the final ground station schematic (Larger Version in **Appendix D**).The ground station also uses an Atmel ATmega328p but it only uses 4 pins to connect external modules. These pins are configured to be the UART connections to the XBee Pro Transceiver and the CP2102 UART to TTL module. The XBee Pro Transceiver uses Pin 2 and Pin 3 as hardware UART. The CP2102 uses Pin 13 and Pin 14 as software UART.

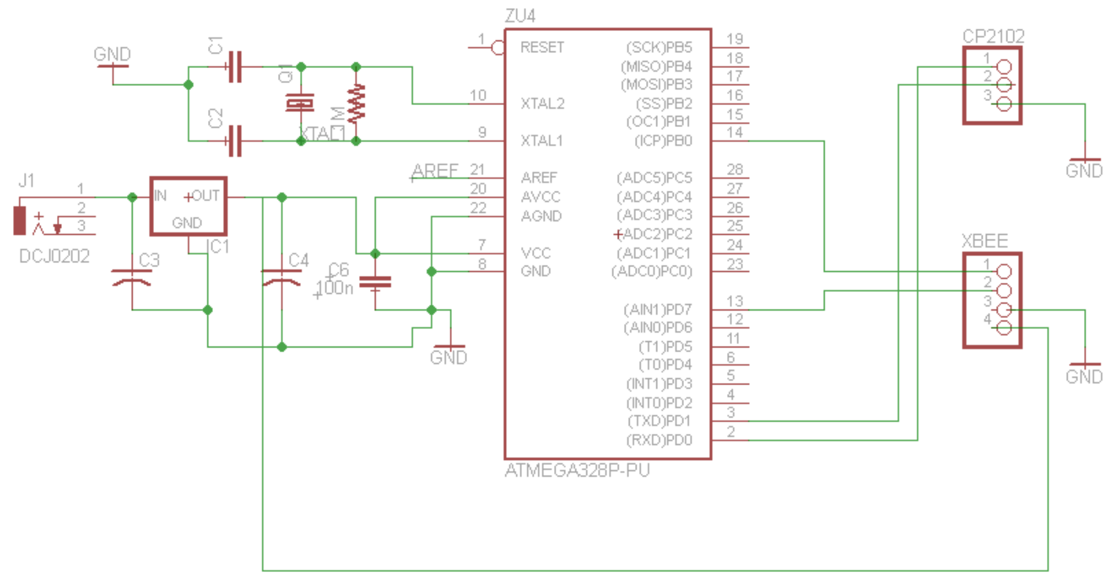


Figure 7.3.3 – Schematic Capture of Ground Station Board

Section 8: Project Prototype Testing

This section is dedicated to testing all the parts of our Blimp. The tests cover all the components working on the ground to all the components on the blimp and working in the air. For all these tests we need to make sure we don't burn out any of the components and if it does happen we are ready with a backup module or can get one shipped within 2 days. Items that are to be tested most extensively are the blimp structures, since if we have any leaks in the structure the blimp will not stay afloat for long and we won't be able to demo or even test the other components in the air. When thinking about the structure we need to make sure everything is balanced out with the carriages, if something is unbalanced, when we start to make a turn, it could cause the blimp to flip upside down or even possibly hit some innocent bystander.

Components that are tested before getting on the blimp and will be mostly tested on the ground will be the Blimp Control Board with all its modules and motors. We need to make sure that the transmission works across line of sight on the ground and see how much signal degrades through a building.

The ground station and the computer graphical user interface (GUI), will stay on the ground and are tested with the air components extensively. We need to make sure the video is displayed in the GUI and we can control the blimp through the GUI. Communication with the blimp all depends on the ground station transmitter and we need this to connect to the blimp constantly and efficiently.

Section 8.1 *Flight Test Environment*

One of the most important parts of developing a project or product or anything that has a use to it or that will be sold for consumers is the testing behind it. Not only do you have to make sure it works properly you also need to test it properly multiple times to ensure constant use. However, being just as important as testing the product, picking the correct and appropriate environment to conduct the test in is a simple, but yet, great task. For instance, since the MASS will be a decent sized aircraft we will ultimately want to be testing in an outdoor area, but we don't want it to be damaged from weather, or trees, or really anything that can harm the system.

First thing first, we will be running the very basic preliminary testing at the location of the build. Note it the **[Figure 8.1.1]** below for this will be where the main build takes place. As far as testing the system indoors, we will have it anchored to the ground just to check everything over. Once each and every group member feels comfortable with the results of that first initial testing we will move the project testing to the university in order to have a large enough area to make assessments without problems.

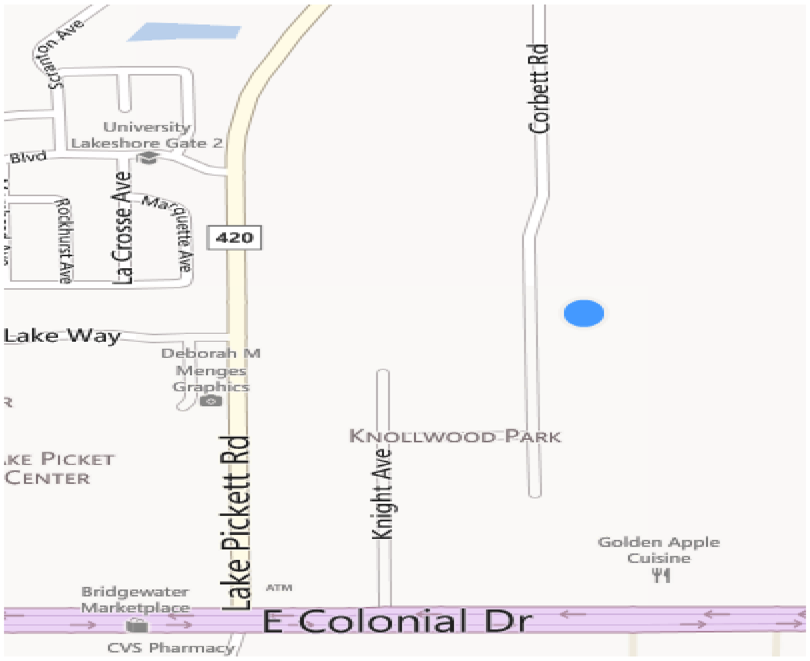


***Figure 8.1.1**: Location of initial build. This will be where the first testing phases will be conducted. Directions to Build Location: Head South on 434/Alafaya Trail from the University of Central Florida. Turn East onto HWY 50/East Colonial. Make U turn after passing Lake Pickett Rd. and proceed to turn right onto Knight Ave. Make another Right onto Belles then left onto Corbett Rd. ending at 2019 Corbett Rd. Orlando, FL.*

Since we will be doing the preliminary testing indoors we will need a big enough space to do so. The actual site we conduct this in does not matter, but most likely we will be using the Engineering 2 building main corridor **[Figure 8.1.2]** since it is not only roomy enough to do full circles with the blimp, but is also tall enough to test the our signal transmissions and will correspondingly prevent any "flying away" situations of any sort if the signal to the motor functions were to malfunction. We will also need to test it near a power source for our computer system and any other equipment we need to plug in. Conducting the initial testing inside will also alleviate any weather conditions and will prevent the system from enduring any other variables besides the ones we input from the controls system. If anything goes wrong or doesn't work properly we will continue to run the testing and make corrections until desired results are apprehended. We will also have a type of string connected in order to anchor the blimp to the ground in the instance of a flyaway. If something goes wrong it is always good to have assurance that the situation can and will be controlled for the case in which we can simply just pull the blimp down to safety. Assuming nothing goes horribly wrong (i.e. crashing and needing reconstruction) we will move on to further testing.

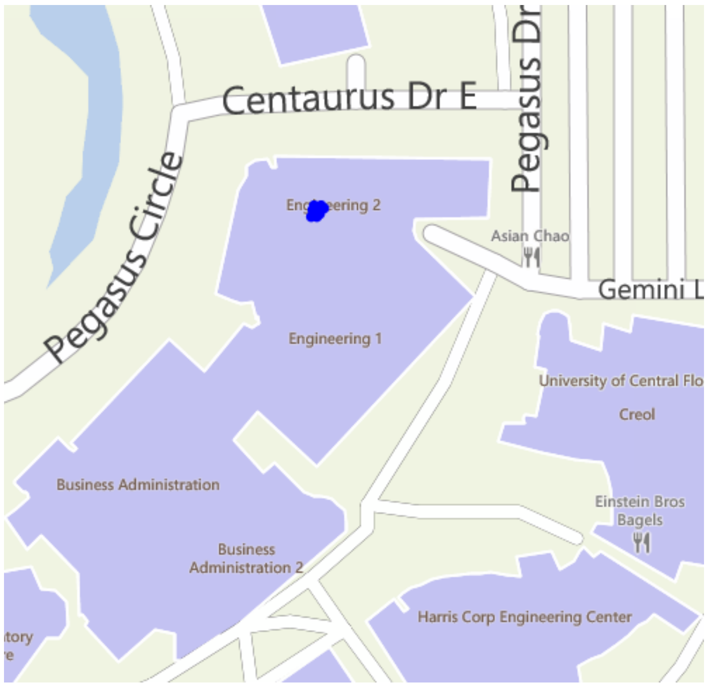


Figure 8.1.2: *In the figure above you can clearly see the building we will be doing the first inside testing phase in. The Engineering 2 building is large and open enough to do the first testing phase. The building in the figure is marked by a blue dot.*

Once the preliminary testing is finalized and we have gathered all the data needed we will start conducting the same tests we did in the prelims outside. The tests being the same, we will collect our data based off of the added variables of the outside conditions. Some of the following conditions will be in the added variables: humidity, wind, temperature, distance, etc. For some of the main testing we will be in the area in front of the UCF arena called the Memory Mall **[Figure 8.1.3]**. In the preliminary testing, as mentioned beforehand, we will be doing most of the basic testing at the location of the build. Of course when we finally get the project secure and together enough to feel confident of higher altitudes and not just the gondola module we will be doing the first testing when the weather conditions are perfect. Therefore, it must be a forecast showing absolutely no rain, very low humidity, and preferably no wind. The wind, however, will also act as a test in itself considering our system will basically be for outdoor use. Once we test, retest, and test the MASS many times again and again, we will throw several different types of weather conditions at it during our final testing phases in order to ensure full practicality of its use.

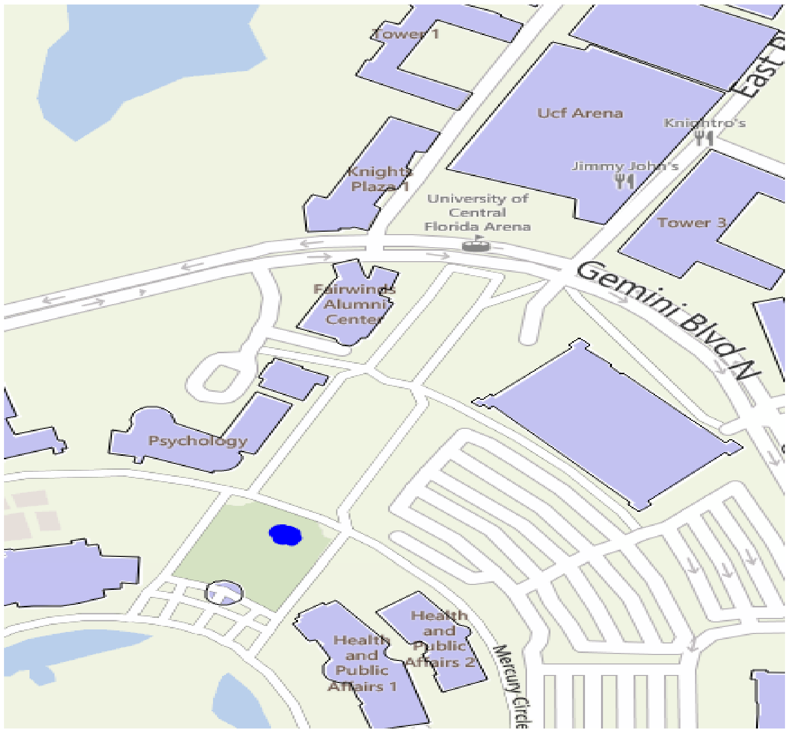


Figure 8.1.3: *In the figure above the Memory Mall is marked by a blue dot. This will be the testing area of the MASS for the outside trials.*

Section 8.2 *Flight Specific Testing*

Before we make plans to fly up in the air we need to make sure all the things work on the ground when the blimp is not floating. The servos must properly turn in the right directions (up, down, left, right) and the motor speeds must go fast or slow enough for our purposes. The blimp structure must have no leaks to prevent the helium from escaping and the envelope must be big enough for the required lift of the gondola, the camera system, and the motors/servos.

Section 8.2.1 *Pre-Flight Testing*

The pre-flight testing procedure will be set in place in order to prevent any disastrous situations from happening, as well as checking all of our components and moving parts to maintain consistency not only in our aircraft, but also in our data analysis. For example, if we didn't check the battery levels before every flight, we would not know the approximate amount of time we had available in the air and could possibly have a situation ending in harm to not only the MASS, but perhaps to someone at the scene of the event. Even though we will implement this routine into out pre-flight humdrum we understand this does not replace the main testing needed to be done. This list simply will be there to accommodate our data collecting. The following **[Table 8.2.1]** will merely act as a checklist for the pre-flight testing:

Task	Complete?
Check weather for go flight (outside)	
Check area for safe go flight environment	
Check battery for voltage reading	
Check all soldering work	
Check props to ensure they are tight	
Check for leaks and or abrasions in envelope	
Check connectivity of all wiring	
Check sensors	
Check for responsiveness from controller	
Check gondola for a secure structure	
Check batteries for a secure structure	
Check if video is rolling and responsive	
Check if GPS is pinging and responsive	
Check camera positioning responsiveness	
Check camera for secure structure	
Check overall integrity of full structure	
Check android phone for connectivity	

Table 8.2.1: *The figure about is the checklist that will be used for pre-flight testing.*

Once checklist is complete we will commence with the scheduled flight. If a failed or faulty element ensued during the checklist process, pre-flight testing would be considered a fail and until the part was replaced or fixed we would not have a 'go flight'. These are the steps that will be taken for a proper initial walkthrough of the MASS to not only give the project members a sense of optimism about the flight that is about to take place, but will also act as a safety precaution for the members as well as any witnessing peers.

Section 8.2.2 In-Flight Testing

This section is meant to explain the testing procedure once in flight. The MASS blimp must do the following:

- Hover
- React to every signal given
- Turn a complete circle
- GPS receive signal
- Full Power
- Video Feed

Section 8.2.2.1 Physical Testing Standards

This section will briefly describe our testing criteria of how our blimp system will be graded during testing. The following steps will be part of the procedure of arranging the prototype for go flight. These are the guidelines we will make an effort to meet with attentiveness before the ultimate demonstration stage. Following these steps the project at hand should be working and fully functional. It is our accountability as engineering students to demonstrate the project as so. We will be going to complete the following bullets before the final presentation and overall grading of the project.

- All electronics should respond to battery power and turn on and off when signaled to do so.
- Envelope must be able to hold the appropriate amount of gases and air without ripping, leaking, or seeping and also must maintain the main elliptical shape intended.
- The control unit for the airship must have a response for every command given. For example, if left turn has been initiated, the props should act accordingly making the MASS turn left.
- Motor controls system must work properly. Motors must accelerate and decelerate as controls are pressed with a certain amount of leverage. Servo motor guiding the electronic motors controlling the two main propellers should act as a result of changing altitude by achieving proper rotation of the axis from zero to one hundred and eighty degrees. Rear motor, just like the two main motors, must drive into forward or reverse motion in order for rear of vessel to be pushed left or right helping to lower the overall turn radius of the MASS. All three motors must act and respond correctly together in order to ensure proper rotary drive.
- With all motor drives working without malfunction, such as coding, sparking, smoking, etc., we will need to test the integrity of the blimp. We will do a series of tests, testing the turning radius, stopping distance, speed factor, elevation and de-elevation.
- Camera must give and maintain a responsive signal to the computer control system.
- Camera stabilization system must recognize commands given in order to maintain a smooth rotation and drag-free path if desired or system must respond to the home function feature if desired.
- GPS system must maintain a responsive signal to the computer station
- If a desired course is set at the computer station the MASS must recognize the desired path specified.
- Remote control must give the correct command when desired.
- Android controlling system must give the correct command when desired.
- Battery system must be put through every scenario in order to achieve values for the amount of flight time. Hypothetical results can be determined just based off of the manufacture's notes and data given with

the battery; however, these figures will not give us accurate 'real-time' results, thus the batteries need to be put through a scenario test we the engineering student will put together. The lithium polymer (lipo) batteries (two-three) will need to be able to handle the total current pull and voltage output essential to the project. The following scenario tests will be given to determine the values associated with **[Table 8.3.1]:**

- Standby Power
- Half Power
- Maximum Power

Battery Scenario Test Values

Test Type	**Time**
Estimated Standby	
Actual Standby	To Be Determined
Estimated Half Power	
Actual Half Power	To Be Determined
Estimated Max Power	
Actual Max Power	To Be Determined

Table 8.3.1*: The table above shows the estimated and actual operating time for the MASS for three different scenarios. Once data is fully recorded we can have an approximated run-time by averaging the three fields together.*

Section 8.3 Hardware Test Environment

Just like the flight testing environment we will need to test the hardware indoors and outdoors. Both the indoor test has to be more detailed than the test outdoors. If we do not test the system indoor and on the ground enough, we could have a fatal problem in controlling the blimp. We would then have the possibility of losing the blimp by it just floating away, but we will take a safety measure of having the blimp tethered to the ground and have someone hold onto it in case there is a possibility of the tether getting caught up in the motors.

Section 8.4 Hardware Specific Testing

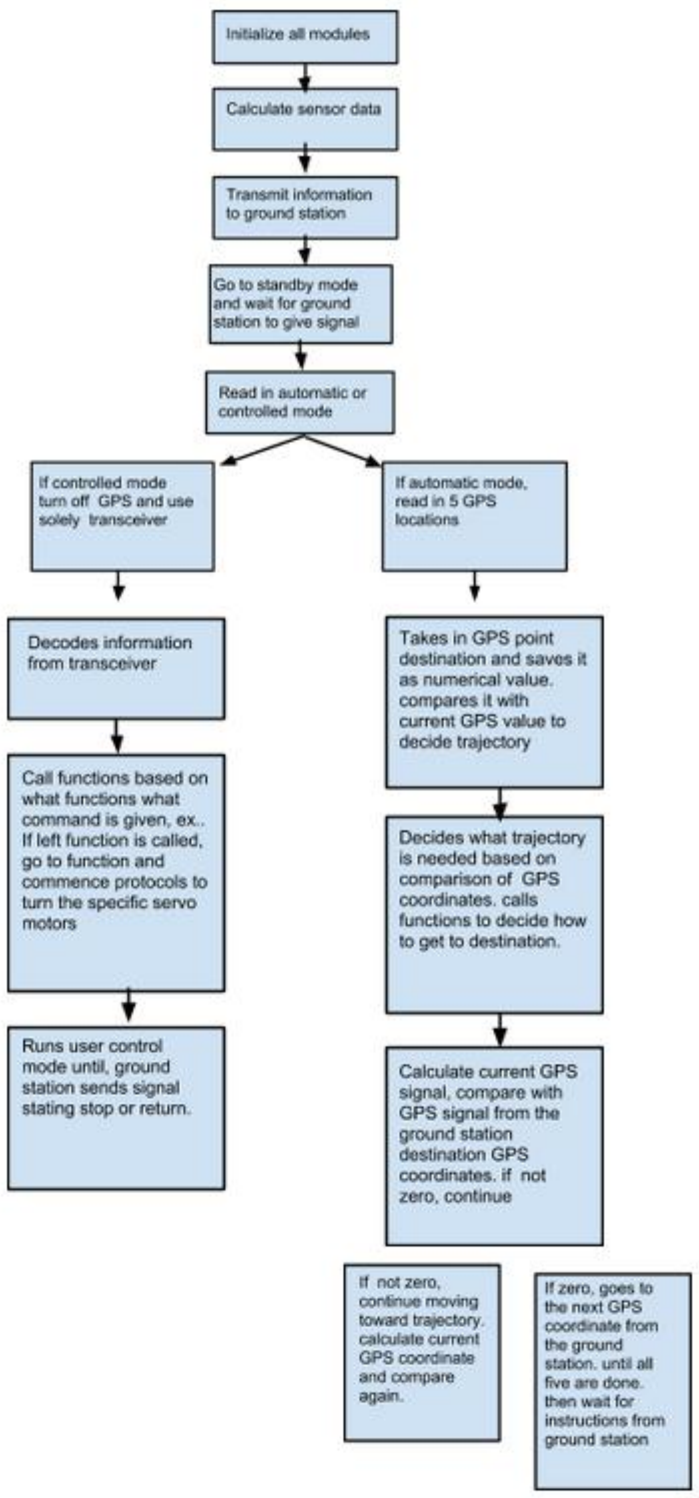
In order to ensure the MASS will work to our high standards we will first check to make sure all the electronics hardware work. By using digital multi meters the group will check each of the parts specifications. Once data is collected on the chips and individual parts we will move to the motors and props then further move on to the camera system and GPS and furthermore. Buy testing the motors we just need to connect them to batteries to ensure the work in both directions. To further test the motors we will connect them to a throttle control system that will be in turn connected to the microcontroller, which will allow us to change and vary the speed. We will have to test the GPS system by connecting it to a computer system and changing locations of the module and observe if it records the alteration. The same type of test will be implemented to the camera. The video transmitter will be connected to the camera and we will just note if it records to the computer screen. However, there will be different testing for the camera stabilization system.

The camera stabilization system will have a couple different settings. It will have a “home” setting making it only set to record in one direction into a locked position and by hitting a button on the control system it will change from automatically stabilizing and switching to the home location. The automatic stabilizing setting will allow it to record in one direction, but instead of being locked in the front facing direction it will be a smooth centralized recording in the area in which the button was pushed. The testing will be to observe if it works the way it’s intended to work. Basically we will set the camera in the pad, move the blimp one way or another, and observe on the computer system if it’s a smooth and level recording. Then, hit the home button and see if it returns to a front facing position and holds there until the button is pressed once again. While the blimp is put into motion we will test to see if the camera is streaming and continuously streaming without any interruptions.

The batteries will be the main power of the MASS and will also need to be tested several times in order to ensure the maximum amount of flight time. In order to test this we will first test standby time. We will turn on the MASS and just let it sit with everything running, but just inactive. Do this a few times and just record the run time. Then we will anchor the blimp and run everything all at the same time and then record that runtime. Once both data samples are taken we can make a hypothesis on how long the MASS will run will moderate use.

The remote control will be an old game console wireless controller and will control the obvious direction and propeller speed.

Hardware program flowchart



Initialize all modules
Calculate sensor data
Transmit information to ground station
Go to standby mode and wait for ground station to give signal
Read in automatic or controlled mode
If controlled mode turn off GPS and use solely transceiver
If automatic mode, read in 5 GPS locations
Decodes information from transceiver
Call functions based on what functions what command is given, ex.. If left function is called, go to function and commence protocols to turn the specific servo motors
Runs user control mode until, ground station sends signal stating stop or return.
Takes in GPS point destination and saves it as numerical value. compares it with current GPS value to decide trajectory
Decides what trajectory is needed based on comparison of GPS coordinates. calls functions to decide how to get to destination.
Calculate current GPS signal, compare with GPS signal from the ground station destination GPS coordinates. if not zero, continue
If not zero, continue moving toward trajectory. calculate current GPS coordinate and compare again.
If zero, goes to the next GPS coordinate from the ground station. until all five are done. then wait for instructions from ground station

Figure 8.4.1: Description of how the Blimp Controller's Program will be laid out.

8.4.1 Hardware flow chart details

The flowchart **[Figure 8.4.1]** describes the algorithm that will be running when the blimp is on. When the blimp is turned on, it first turns on all of its external modules, initializes and reads in the data from them. It then calculates all of the data, to find out the current position and its other standby information. Once this is all calculated, transmit all data to the ground station for further directions. This transmitting should be done with the serial port in either, SPI, I2C or with USART. The ground station then gets control input from the user. If the user decides to manually control the blimp, that signal will be relayed to it. If not, the user will type in 5 GPS coordinates for the blimp to go to. In user control mode, the GPS module will be for the most part, turned off so more power will go to the transceiver and ultimately for the user input signal to be clearer to read in. The MASS will wait and read in and decode information from the transceiver in real time. Based on that Information, the microcontroller will decide what functions will be called to implement movement. These functions will be like for example, turnLeft(), which will be calling the certain pins to turn the servo motors in the tail fin and to increase or decrease power into the motor. In all there should be around 10 directional functions. Which include turnRight(), go(), uTurn(), goUp(), goDown(), etc. All of these functions will essentially, turn servo motors and change the power of the blimp. The user control mode will stop once the user decides that user control is over and would opt to automatic control or to make it come back toHome(). If the user decides to make the blimp go in automatic mode, the user then picks 5 GPS coordinates for the blimp to go to. These coordinates will be read into the blimp and saved as numbers in the data memory (most likely in an array of ints). The blimp will also save its current GPS location so it can compare its current location with its GPS destination. If the comparison of the two locations does not give a zero, then the blimp will keep continuing on until its current GPS point and the GPS destination are the same. Once the blimp has gotten to the destination, the blimp will go to the next GPS destination in the array and repeat until the process until all of the numbers in the array are all gone. Then the blimp will wait for more instructions from the ground station. This will make the program go back to the beginning of the process.

Section 8.5 Ground Station Environment

In the lab, it is possible to test out the range by doing an attenuation test. The RF attenuator will pass the RF signal through a serial of resistors simulating long distance. If we can get non-corrupt data at the other end of the attenuator we will know that the signal will work at long ranges. In the senior design lab, it will be easier to modify our circuit and change the peak-to-peak values or the logic of our signal to give it longer range or clean up the noise cause since we will have access to an oscilloscope to look at the waveform.

Then the plan is to go to a large park with open fields and test out our design, we want to first test out when our blimp has line of sight (open area testing) with our receiving station. Then we will test long range at UCF while we have buildings in the way of our transmission. We hope the blimp will either not have problems getting transmissions through the buildings/structure or have our automated movement kick in and have the blimp reposition itself to move back into transmission range or sight of the antenna.

Section 8.5.1 Ground Station Transmission Prototype Testing

The ground station will need to be tested in several parts for each transmission unit. First we will build the ground station unit on a breadboard and connect each pin of the transceiver module and Bluetooth module using alligator clamps and prototyping wire to the ATMEL ATMEGA328. We need to make sure the design actually works before we send it the final eagle schematics to the PCB vendor.

In case the blimp microcontroller and attached modules are not ready by the completion of the ground station we will need to build a fake blimp microcontroller using an ATMEL ATMEGA328. The fake blimp data will just have a few cables and mainly consist of the ATMEL ATMEGA328 connected to the SPI pins of the nRF24L01 transceiver on a breadboard. This fake blimp design will have a programmed ATMEL ATMEGA328 to output data that will imitate the GPS NMEA codes and the accelerometer data, this will fake the position (roll and tilt) and coordinates of the blimp (longitude and latitude) and the speed its going at.

Section 8.5.1.1 Ground Station Power Testing

The ground station needs to be supplied with the correct amount of voltage in order to power all the modules and the microcontrollers. For many of the transmission devices the wrong voltage to it could result in a fried unit and would be rendered useless. We really need to avoid this as these modules such as the Bluetooth gets expensive and takes a long time to deliver a replacement unit to us (up to 2 months).

First we need to check that the voltage regulator is working properly, we need to connect a 9 volt battery or a 9 volt wall plug AC Adapter to the input pin of the LM7805C and use a multi-meter to measure the DC voltage that it is output, the result should be 5 volts. Then we will check the nRF24L01 transceiver power input, we will have a resistor to change the 5 volts to 3.3 volts for the transceiver power, this is to meet the required supply range of 1.9 to 3.6 volts. The transceiver power on the ground station should also match the voltage input on the transceiver (3.3 volts) as a different voltage can result in different transmission speeds and one of the transceiver will have to buffer a lot of data as it will not be able to keep up. Then we will need to test out the power going to the WT12 Bluetooth module, the input specification require a 3.3 volt input, anything higher will fry this module and we need to be very careful in measuring this. The last module, the CP2102 USB to TTL/UART module requires a 3.3 volt power connection, which is the requirement for powering USB devices, without this minimum input the USB connection to the computer will not be detected by Windows Device Manager.

Next we need to stress test that the ground station will be able to be powered by the battery or wall plug for the duration of the flight. With the LM7805C, there is a chance it could overhead, there have been reported problems of if you supply a voltage over 10 volts for a long enough time, the voltage regulator will overheat and fail until it is cooled down. This should not be a problem since we plan to use a 9 volt lithium battery to power the ground station if we do not have access to a wall plug AC adapter. With the LM7805C, it has a current draw of 1 amp this should be enough for a lithium battery to run for at least 4 to 5 hours even at maximum current draw, but we will test out the full battery life of the battery that we choose to use.

Section 8.5.1.2 PIC18F2550 Microcontroller Testing

To test out the microcontroller we first need to hook up all the I/O pins with the communication modules, such as the nRF24L01 transceiver and the CP2102 USB to TTL/UART connections. We will use a multi-meters' connectivity test to test out if the connection was soldered on or breaded boarded properly. We also need to make sure the ground connections of all devices are connected together. Then we will test out the data transmission coming out of the PIC18F2550, using an oscilloscope we need to make sure the data coming into and going out of the PIC's SPI connection is correct (RX and TX pins for SPI checked) and that the PIC's UART connection is sending out properly. The PIC's UART connection should be outputting a TTL signal which looks like a square wave with it going from 0 as a low and from 3.3 volts as a high.

Another important feature of this PIC18F2550 that we need to test is the interrupt pins which will coincide with the programing of the PIC. We need to make sure that when a certain data transmission is received that it will activate the interrupt pin on the PIC, we will check this by hooking up an oscilloscope to the interrupt pin and making sure it goes high and low at the proper times of receiving an interrupt message from the UART pin.

Section 8.5.1. Nordic Semiconductor Transceiver (nRF24L01) Testing

The PIC18F2550 that we are using has a UART pin which we will be using for testing with HyperTerminal. Since we cannot get the computer to read the SPI transmission coming directly from the Nordic transceiver, we need this UART pin for debugging. We need to make sure that the waveform coming out of the blimp is received properly and have the same peak to peak values; if the values are different we could have a problem with corrupt data. We will send out fake test strings and not actual data from the blimps' transceiver using the Atmel ATmega 128 microcontroller to test out the receiver on the ground station. Then we will send out data on the transmitter on the ground station and see if it is properly received on the blimp. These two tests will test out the RX and TX mode of the transceiver is working properly.

Section 8.5.1.4 Bluegiga WT12 Bluetooth Module Testing

If the TTL connection on the two pins (pin 28 - RX and pin 27 - TX) we designated for the Bluetooth module do not work we will have to use a multiplexer. But first we will test out the easier implementation of converting these pins to use UART which is a TTL connection. We will plug in this RX and TX line into the Bluetooth module and make sure we can transmit to and from them. Before we actually transmit the data we need to make sure the data being outputted on the line is actual TTL and the logic is correct using an oscilloscope.

The alternate testing for the WT12 Bluetooth module will be a little harder, since we will be using the same UART pin on the ATMEL ATMEGA328 for the Bluetooth and the USB connection to the computer. We need to make sure that 1 to 2 multiplexer is working properly on the line, this includes making sure the pin we connect to the control of the multiplexer is selecting the Bluetooth module for transmission. We know pins 17 and 18 are the RX and TX of the UART pins for the ATMEL ATMEGA328 but we will check the logic and the data signal on an oscilloscope to make sure it was properly set right to output to the Bluetooth module.

When using the Bluetooth module, it is very important to always check that only 3.3 volts is connected to the power input of the breakout board. The design specification specifically says that inputting more voltage than 3.3 volts will burn out the WT12. We will use a multi-meter to continuously check this when we are designing the circuit and when we order the Printed Circuit Board (PCB) design.

The final design of our project excluded the Android Phone since the Bluetooth module cost too much and there were other problems with the design. So the Bluetooth module was removed from the ground station.

Section 8.5.1.5 USB to TTL/UART Module (CP2102) Testing

To test the CP2102 module, it will just be a simple plug into the USB port of the computer and see if it detects the ground station device. There will be a notification of installing the device and initializing it to a COM port, but we need to make sure the drivers from Silicon Labs are installed on the computer before we plug into the USB port. If we do not see the Windows Notification pop up we can use the Windows' device manager and see if the COM port for the USB device is there, the COM port number can vary greatly from 1 to 50 and change each time you plug it into the computer.

Section 8.5.2 Ground Station Printed Circuit Board Design (PCB) Test

Once all the parts above have been confirmed working on a breadboard we will then make the schematics for it on eagle and send it out to get the PCB made. Once the PCB arrives, we need to take all the modules from the breadboard that is confirmed working and move them onto the finalized PCB design. Then we will basically repeat every test for all the modules and double check that it's all working the same. If there are any problems we need to double check the traces in the PCB and make sure there are no manufacturing mistakes.

Section 8.6 Software Testing Environment

All software will first be tested indoors with direction connections or with short ranged transmissions. We will first test out that the software on the Blimp microcontroller and the Ground station microcontroller are working properly by sending out basic commands directly to the UART pin in the Atmel ATmega328 on the Blimp and the Atmel ATmega328 on the ground station. Once everything works properly in the first tests, we will increment the distance from 1 meter away out to 100 meters to make sure the transmitter is sending everything properly.

Section 8.6.1 GUI Environment

The GUI will be tested and debugged in Visual Studio 2010 Professional. The system environment the GUI will be tested on is provided on the **[Table 8.6.1]** below. The system we are using will be the minimum specifications for the GUI to run in, but it could probably run with lower specs.

	Development Environment
Operating System:	Windows 7 Professional
Processor:	Quad Core 2.4 GHz
RAM:	2 GB
Video Card:	DirectX 10 Supported

Table 8.6.1 *- Our Development/Testing Environments Computer Specs*

For the GUI debugging, the environment will consist of the development desktop computer which has Visual Studio 2010 Professional and a laptop which the code will be published and deployed on. This laptop will have all of the software required for the code to run on (.NETFramework 4, ASP.Net, Video Encoding/Decoding Software, DirectX 10.1) including the drivers for the ground stations' TTL to USB Connector. The development software will not install on the laptop for development incase Visual Studio 2010 installs extra software that we do not know about.

Section 8.6.2 Android Environment

The Android Application will be tested in the Android's SDK development environment. First the app will be emulated on the development IDE, making sure there are no bugs on the virtual phone. Then we will move to using a physical Android phone that will be updated with the latest Android OS version (Jellybean version 4.2). We will first be testing out the android phone indoors in the Engineering building at UCF in the Senior Design Lab with the ground station and the blimps' microcontroller. Then we will move outside to an open field and test the android phone where we will have the blimp up in the sky and going out to a maximum range of 100 meters. The android program should not be affected by being inside a building with no signal but we wanted to make sure if there are any differences in the phone having signal or not.

Section 8.6.3 Range Testing Software Environment

Ranges will be recorded by GPS position of the receiving station to the blimp. We will start the blimp on top of the receiving station and record its GPS position, then we will fly the blimp out until it loses signal and record that GPS position. From these two GPS position we will calculate the distance in meters on how far out the blimp can go out in one direction. Then we will test out the altitude limitations and send the blimp directly upward of the receiving station and record that distance.

These range tests are important for the Patrol Pathing Software the C++ GUI will implement. Once we know the possible range the blimp can reach we will be able to limit the area where the user can draw a patrol path for the blimp radiating out from the receiving stations location.

The connection to OpenCV and how fair the C++ GUI can pick up the video feed is also important to test. As of now the GoPro's IP Camera Wi-Fi Range is supposedly ranged tested for 91 meters, while our transmitters can reach up to 300 meters depending on the transmission speed. We want to test that the IP Camera will not lose connection within the 91 meter range and will continue to send out a reliable infrared or optical video feed. If we do lose the camera feed we want to make sure our recovery program to recover the camera link will work too.

Section 8.7 *Software Specific Testing*

The programming for the hardware will be tested by making sure the Blimp PCB correctly receives the data from the sensors, the motors, the camera, the transceiver, and the C++ GUI Application. Then we need to test the transmission of the data from the Blimp's PCB to the ground station and finally to the GUI Application.

Section 8.7.1 *Microcontroller Software Debugging*

The hardware testing section should cover some of the programming of the PCB but when writing the C code for the PCB to run, we need to make sure the automated system works. The Automated System will kick in when it cannot receive transmission data from the ground station, we will implement this by moving the Blimp out of the range of the ground station, then seeing if it can find its way back to transmission range. For the Atmel microcontroller on the PCB we will need to make sure it compiles correctly through the Atmel programmer and in the IDE we choose to use. We need to make sure the libraries we load onto the microcontroller uses the correct pins and set the ports for the pins to the correct input and output for the module we connect to that specific pin. No one has worked with the Atmel programmer before so it will be a new experience and a lot of troubleshooting through forums to get the program on the Atmel ATmega 128 to work properly with our circuit setup and the modules it uses.

Section 8.7.2 Ground Station PIC Software Debugging

The ground station's software is programmed on the PIC18F2550 on MPLabX using the C18 programming language. We just need to make sure everything compiles correctly and there are no errors thrown when building the program file. Any errors thrown can be fixed easily by looking at the line that the error is thrown and correcting it by using the C18 reference manual and looking up that the method we are using is used correctly. Then after everything compiles correctly, we need to test out the software works the way we want it to, like when we are transmitting or when we are receiving and when an interrupt signal is sent to the pin from receiving a certain code from the two controlling devices (computer GUI or android application).

Section 8.7.3 Graphical User Interface (GUI) Testing

When testing out the GUI, we need to make sure each button does what is supposed to. All keyboard key presses and mouse clicks will be tested on each clickable surface of the GUI. The only reactions from the program should be from the Arrow Keys and the mouse Clicks on the Patrol Path Interface.

Section 8.7.3.1 GUI To Hardware Testing (Blimp Controls)

For the C++ GUI application testing, we will hook up a direct connection without the ground station to the PCB and make sure all the commands echoes back correctly into HyperTerminal. Once direct connections to each device proves to be correctly outputting or inputting the right signals we will be moving onto short range indoor tests of the blimp with the same echoing to HyperTerminal setup.

Section 8.7.4 Secondary Controller (Android Application) Testing

To test out the Android Application we need to make sure the camera video feed works correctly. The android application should first connect the ground station through its mini-USB port. The android OS should then recognize that there is a device connected to this port and it is trying to communicate to it serially (using UART). Then we need to start up the application on the phone, make sure it goes to full screen and starts up its video feed with the gamepad overlap for the blimp controls. We will test out each button to make sure that the blimp moves in the right directions. Then we need to make sure as a secondary controller if the C++ GUI application is running at the same time, the android application cannot overtake command. This is to avoid confusing the blimp by not sending two different movement commands to it.

Next is the wireless Android connection to the receiving station, this will test the Bluetooth connection to the phone. We want to make sure the Bluetooth ID are paired correctly to the ground station and we want to make sure that only a phone with our application can connect to the Bluetooth in the ground station. Then we will test out the video feed and the blimp controls using the Bluetooth connection once it is established. The Bluetooth connection should work independently from the cell phone carrier's signal or a Wi-Fi signal, so it should work as long as it can connect to the ground stations' WT12 Bluetooth Module. The Bluetooth module has a maximum range of 30 meters with sight, so we will also figure out what will happen when we move out of that range. If something catastrophically fails with the program when moving out of range, we will need to suspend the android application and make it go into a search mode for the Bluetooth signal and make the blimp go on an auto patrol path mode or return to the ground station.

Section 8.7.5 Full Deployment Software Testing (C++ GUI)

Finally full deployment into a long range outside test will be held. This test needs to test out the maximum range of the blimp and how the blimp will react when it is out of transmission range. Every component in the blimp, the ground station, C++ application, and android application will be completed by now. We will run test for 3 separate days, first day we will be testing out C++ application, then the second day we will test out with the android application, and finally we will switch between using the C++ Application and the Android App for the third day, seeing if there are any problem with dropped connections to be blimp between switching devices.

Section 8.8 Final Hardware/Software Version Testing

Many things in our final design have changed and so we have to change the testing methods. Although we went through a lot of the testing methods above before we had to change. We found the final versions of each of the changed modules to be a better fit and in some cases easier to integrate together in our project.

Some of the testing changes include switching from C# programming language to the C# programming language. We switched both microcontrollers on the blimp board and the ground station board to the Atmel ATmega328. Next we switched the transceiver from the nRF24L01 to the XBee Pro 2.4 GHz Transceiver.

Section 8.8.1 *EmguCV C# Graphical User Interface*

Originally the tests were done with a C# application but we changed the wrapper for OpenCV to use EmguCV which is C# version of OpenCV to be used in C# Applications. All the functionality is the same as the C# version, first we test the motor control buttons, and then we test the camera video feed with the EmguCV person detector. And finally we have GMaps library on the GUI displayed for Auto-Pathing. This application is run on a laptop and we can have to connect it to the Video Transmitter using the EZcap video decoder and to the ground station which has the Atmel microcontroller and the XBEE Pro 2.4Ghz Transmitter.

Test with the OpenCV detector provide to be very sporadic, but it was the best results we could get. We change the detector multiple times but found the default detector still the most accurate with the least false positives. The video feed actually lags at time but still boxes off the target pretty well, only quick movement of the camera really effects it a lot, on the slow moving blimp, we should not have that many issues with processing power.

Interfacing with the blimp controller through the ground station also provided to have positive results. The movement buttons correctly moved the blimp's brushless motors, each of the servo motors on the camera system, and the axel servo. We saw on the debugging window on the blimp controller board that we were correctly receiving the 8 character commands. But for a more intensive test of a constant stream of commands, the buffer on the XBee modules could not handle rapid tapping of the GUI buttons. A character of the command ultimate misses and the command gets shifted over, causing the blimp controller board to not recognize any command that comes afterwards.

Section 8.8.2 *Final Camera System Testing*

Testing for camera was simple; first it was plugged straight into a TV monitor through the use of RCA cables just to make sure it works. Then things became slightly more complicated because the camera had to be connected through the transmitter that was purchased. After numerous attempts at wiring the transmitter to the camera with no successful video output it was decided that the transmitter was the problem. Finally after ordering a new transmitter, a successful video feed was showing up on the monitor. Next was the step to actually get the camera to work while hooked up to a laptop which involved using an EZCAP analog to digital convertor. After purchasing the EZCAP convertor all testing for the camera was finished with only a miniscule delay. The final procedure was to get the camera running in the gondola while being powered by the power source of eight AAA batteries run in series. That was simple enough involving splicing wires and soldering them together.

Building the Camera Stabilization System was a hard task but testing it was troubling. We only got 1 axis working with the servo motors. The difficulties they discussed were the same issues faced when building the system. The screws had a tendency to break while tightening leaving the system a few screws short. The overall reliability and strength of the system was not compromised. For testing the only way to test was to hook the camera system to the bottom of the blimp and power it through the PCB and start issuing it commands. A few problems arose because of the rigidness of the system; it had trouble turning due to how tight it become when tightening the screws. By not having the tilting feature the system would lose its tracking ability and thus makes the tracking feature completely worthless. Slight tinkering is still happening and the tracking ability is still hopeful to happen for the project.

Testing the new camera system transmitter and receiver combo with the encoder took a couple times to get it to recognize on the OpenCV video stream. The original plan was to use a TX5800s and RCS5800 combo transmitter. The TX5800s had a much larger range at approximately 2000 meters compared to the 200 meters from new transmitter used now. It was smaller in dimension and weighed less. However after numerous attempts to get it to work the transmitter just wasn't compatible with the camera. The camera worked separately from the transmitter and wouldn't when connected so naturally the transmitter was the problem. Every channel was checked multiple times for compatibility and the picture never showed up on the monitor. Eventually it was decided that a new transmitter was needed and the new one worked perfectly as soon as it was hooked up. Some range testing was used for the camera to maximize the distance the blimp could travel and still get picture from the transmitter. After numerous testing the picture stayed relatively clear for around 150 meters.

Section 8.8.3 *Final Structural Test*

To test the blimp structure before it got filled with helium and sent up into the air with the controller. We had to make sure that there were no holes in the balloon, helium is expensive and we did not want to lose a single drop unless it was necessary. We constantly filled the balloon with air and then used the soap and water spray bottle trick to see if any bubbles appeared on the surface. We did this multiple times as we had to deflate the balloon in order to seal it properly. There were problems with the material and the leaks appearing randomly throughout the front end and the backend of the design. We calculated that it would take the balloon about 40 minutes before it loses its flight capabilities.

A full test of the blimp has not currently been implemented as we want to wait for the last couple of days to fill it up with helium and float it up into the air with the motor controls. We have the motors moving as we want it on the ground but as for full deployment up in the air, the testing results are still coming.

Section 9: Administrative Content

The administrative content section we will cover the projected plan for our projected in how we will organize who does which part of the blimp. Then we will discuss the budget in which we want this project to be as low cost as possible but also considering the low cost items will be low grade items. We have a limited selection of suppliers to get items from and we want to get all the materials in within 2days of ordering it and at least have all the materials arrived to us within 1 month of Senior Design 2. We must also consider the other assignments in Senior Design 2 that will most likely hold us up and will take some time with the team meeting to complete it, such as a presentation and a conference paper.

Although we emphasize on trying to save money since we do not have a sponsor, we plan to spend nearly one thousand dollars if not more on this project. We are then planning to gather the receipts and pay back everyone who had a greater spending cost than the other to make things fair. We currently do not know what we are going to do with the project after we complete it. Most likely we will split it into the components that we designed and unassembled the project. Since we have worked together with each other's part we can then reassemble the blimp later is anyone wants to own their own spending their own money.

Section 9.1 Milestone Discussion

One of the most important factors in working on a group project is the planning behind the goals and tasks which need to be completed. In order to effectively complete the project we have to plan for failures. We do not expect everything to work as soon as we get it and piece it together, this is why our testing section is extensive and we will be giving ourselves ample time to test each of our parts before we put it together and do even more testing when we want to communicate between our parts.

Below is **[Table 9.1.1]**, which explains how we plan to execute our fixed time available in order to finish the project in its entirety. We plan to have everyone working together on each part after each person who is administratively responsible for that part to say they are ready to integrate it with the rest of the blimp. The administrative responsibility of each section is listed off in the next section called Administrative Tasks.

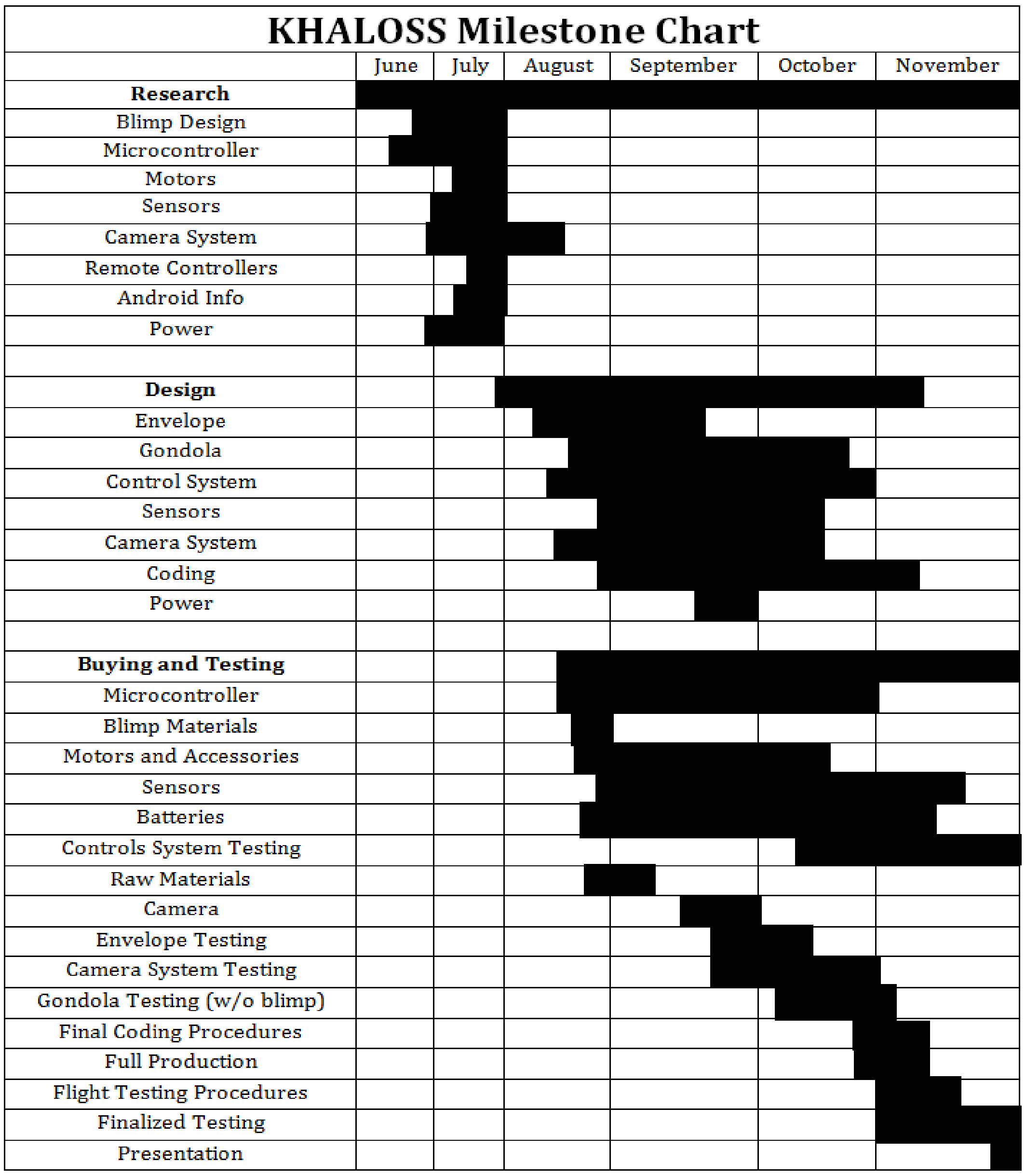


Table 9.1.1*: Milestone Chart*

Section 9.2 *Administrative Task and Responsibilities*

Business is a term used in order to understand this project should not only be clever in its design, but also performed out to be a professional asset. In this section we will briefly explain each of the group member's responsibilities. We will all be working diligently as a group and will help each other out with every objective, but it is the responsibility of the engineering student assigned to the task to make sure it gets completed. This will help well in the completion process not only for the project, but also for the documentation process, shown in [**Table 9.2.1**].

Task	Member
Research	ALL
Envelope Design	Derrick
Gondola and internals	ALL
Microprocessor	Sanjay
Power	Derrick
Camera System	Eric
Ground Station	Henry
Wireless and GPS communications	Henry
Coding	ALL
Sensors	Eric
Android Controls System	Henry
Documentation	ALL
Documentation Formatting	Derrick

Table 9.2.1 *: Given responsibility for each task.*

Section 9.3 Budget and Finance Discussion

Since we have no sponsor for the project, we decided that we will pay for the project ourselves. The final amount of the budget will be divided evenly among the four individuals. The most money about 50% of the budget will be towards the Blimp Structure.

Table 9.3.1-9.3.4 below shows the projected budget for the project. Each part is split into their own sections so we can better gauge how much we spent on each part. The most expensive parts should be the Blimp structure since it contains the helium we need to fill the balloon.

Blimp Structure

Part Name	Model Number	Quantity	Cost Per Unit	Total Cost
Nylon Material for Envelope	-	Est. of 12”x12”	~$40-100	$100
Turnigy Servo Motor	TG9e	1	$1.99	$1.99
Turnigy 150 watt Brushless Motor	L2210C-1200	3	$9.99	$29.97
9x5 Props	H9011A	1	$2.66	$2.66
Helium Tank	-	2	~$60-100	$200
Balsa Wood for Mount	1’X4’ .125” thickness	2	$12.51	$25.02
Epoxy and Nails, Screws, Bolts	-		$10.00	$10.00

Table 9.3.1 - Projected Blimp Structure Cost

The blimp microcontroller is also an important part to look at, in **Table 9.3.2** and **Table 9.3.3** it shows all the components we are buying to put in the blimps carriage. To get all the functions of tracking and auto-pathing it is crucial that there is no noise between the module and the microcontroller.

Blimp Microcontroller Board

Part Name	Model Number	Quantity	Cost Per Unit	Total Cost
Atmega128	Atmega128L-8PU	2	$12.17	$24.34
Printed Circuit Board	Advanced Circuits	1	$ 33.00	$33.00
Capacitors/Resistors/LEDs	-	-	$15.00	$15.00

Table 9.3.2 - Projected Blimp Controller Cost

Blimp Modules (Sensors + Camera)

Part Name	Model Number	Quantity	Cost Per Unit	Total Cost
GoPro Hero 3 Black Edition	Black	1	$ 240.00	$240.00
GoPro Extra Battery	-	1	$ 30.00	$30.00
Accelerometer	MMA8453QT	3	$1.31	$3.93
Gyroscope (STMicroelectronics)	L3G4200D	2	$11.51	$23.02
GPS Module	927-A2100-B	2	$17.57	$35.14
Antenna	APAMP-113	2	$13.01	$26.02

Table 9.3.3 - Projected Blimp Controller Modules Cost

In **Table 9.3.4**, it shows all the components that are going into the Ground station which is the main form of communication between the computer (which the user will be on) and the Blimp (in the air). The transmitter is not cheap since we went all out and got the Low Noise Amplifier and Power Amplifier Version of the Transceiver to improve transmission range.

Ground Station

Part Name	Model Number	Quantity	Cost Per Unit	Total Cost
Voltage Regulator	LM7805C	1	$ 0.29	$0.29
Transceiver Module	nRF24L01+	2	$19.84	$39.68
Bluetooth Module	WT12 w/ Breakout Board	1	$ 50.00	$50.00
Microchip PIC Microcontroller	PIC18F2550	1	$ 4.13	$4.13
8 MHz Crystal Oscillator	ATS08ASM-1	1	$ 0.38	$0.38
USB to TTL Module	CP2102	1	$ 7.25	$7.25
Printed Circuit Board	Advanced Circuits	1	$ 33.00	$33.00
Battery	9V Energizer Lithium	1	$ 7.28	$7.28
USB Extension Cable	Belkin F3U134-10 10 ft.	1	$ 6.99	$6.99
Power Snap (Battery Holder)	BS4T-HD-ND	1	$ 0.70	$0.70

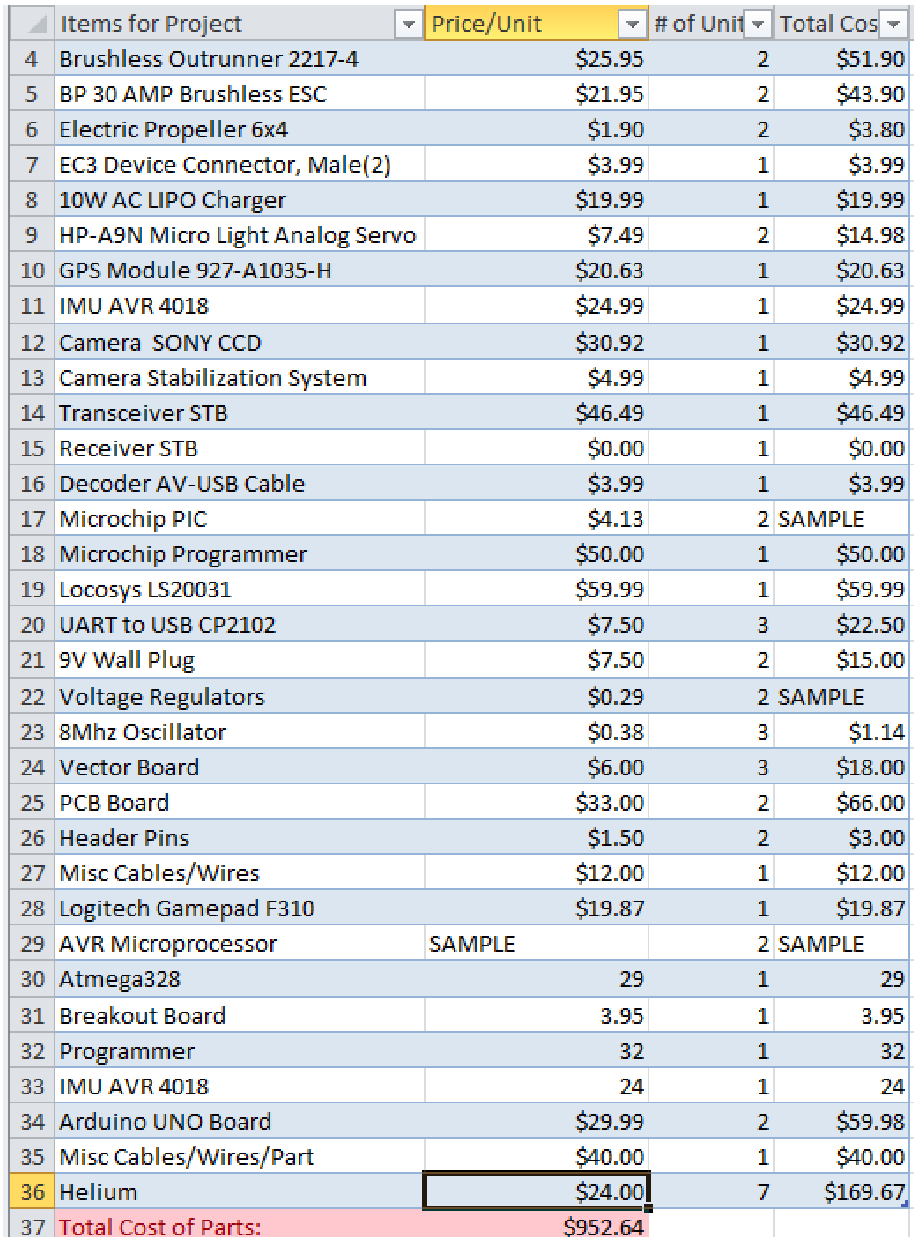
Table 9.3.4 - Projected Ground Station Cost

Estimated Shipping Cost: ……………………………………………….	$50.00
Total Estimated Cost:	**$999.79**

***Table 9.3.1-9.3.4**: Budget Table with Total Cost*

Section 9.4 Total Cost for Final Design

The projected cost of the project build was actually pretty close to the parts that were used on the blimp, but because we switched parts and ordered extra modules for backup we ended up spending about double the estimated cost (~$2000). **Figure 9.4.1** shows the cost of the blimp with only the components that are used either on the ground station, blimp structure, or the blimp controller.



	Items for Project	Price/Unit	# of Unit	Total Cos
4	Brushless Outrunner 2217-4	$25.95	2	$51.90
5	BP 30 AMP Brushless ESC	$21.95	2	$43.90
6	Electric Propeller 6x4	$1.90	2	$3.80
7	EC3 Device Connector, Male(2)	$3.99	1	$3.99
8	10W AC LIPO Charger	$19.99	1	$19.99
9	HP-A9N Micro Light Analog Servo	$7.49	2	$14.98
10	GPS Module 927-A1035-H	$20.63	1	$20.63
11	IMU AVR 4018	$24.99	1	$24.99
12	Camera SONY CCD	$30.92	1	$30.92
13	Camera Stabilization System	$4.99	1	$4.99
14	Transceiver STB	$46.49	1	$46.49
15	Receiver STB	$0.00	1	$0.00
16	Decoder AV-USB Cable	$3.99	1	$3.99
17	Microchip PIC	$4.13	2	SAMPLE
18	Microchip Programmer	$50.00	1	$50.00
19	Locosys LS20031	$59.99	1	$59.99
20	UART to USB CP2102	$7.50	3	$22.50
21	9V Wall Plug	$7.50	2	$15.00
22	Voltage Regulators	$0.29	2	SAMPLE
23	8Mhz Oscillator	$0.38	3	$1.14
24	Vector Board	$6.00	3	$18.00
25	PCB Board	$33.00	2	$66.00
26	Header Pins	$1.50	2	$3.00
27	Misc Cables/Wires	$12.00	1	$12.00
28	Logitech Gamepad F310	$19.87	1	$19.87
29	AVR Microprocessor	SAMPLE	2	SAMPLE
30	Atmega328	29	1	29
31	Breakout Board	3.95	1	3.95
32	Programmer	32	1	32
33	IMU AVR 4018	24	1	24
34	Arduino UNO Board	$29.99	2	$59.98
35	Misc Cables/Wires/Part	$40.00	1	$40.00
36	Helium	$24.00	7	$169.67
37	Total Cost of Parts:	$952.64		

Figure 9.4.1: Actual cost of components to make the Blimp

Section 10: Final Summary

Many things came to mind when we entered this senior design class. We were not sure of what project we wanted to do since there are so many out there as well as so many still to come up with a do. However, when we as a group finally decided to put the Mobile Aerial Surveillance System into the planning stage we knew we were in for some fun. What we did not anticipate is how much work is actually behind the planning process. At first it seemed kind of insignificant to a point. Put some air in a balloon, attach a motor to it powered by a battery and you are good to go, right? The answer is a big, yet simple, NO!

While we have finished the design behind the MASS we fully understand that there is still much work to be done. We still have to further research how all these parts and electronics go together. It's not like a simple system integration "plug and chug" concept. Parts have to be calibrated, code has to be written, and signals have to be transmitted. We will have to debug the problem that will inevitably occur. We will have to put in some sleepless nights in order to get this this running correctly. However, we believe that our project fully covers a little bit of everything as far as being engineering students.

Closing thoughts to this project, we have learned that aerial projects are a lot of fun but also a lot of work. It takes a lot of information from structural engineering, aerospace engineer, as well as electrical engineering to get any aerial project off the ground. We would definitely be playing around with RC cars and trying to design them after this project. But we still learned a great deal of the systems required for movement in the air and communication over long distances. Picking out the right parts and not going with the cheapest parts were definitely a lesson that we learned. With the final testing of the projects, we see that it takes weeks of testing to get the correct program and the correct circuit in order to power the system and make it perform the way you want it to.

There were a lot of features cut out during the project and we learned that we set unrealistic goals in the beginning since we were all ultimately new to creating this kind of project. But the features that were left such as the auto-pathing and just plain moving the blimp in the air were still really cool to implement. Picking out parts and having them work together was a good experience and we are all sure to continue this kind of project as a hobby that we can play around with at home.

Working together as a group was a fun experience and everyone seemed to learn a lot about making a Blimp fly in the air and controlling it. Everyone followed their administrative task but also worked together on each of the parts when it came to integrating the parts together. We shared a lot of difficulties in the project but learned that communication between team members and research is the key to success in this project and ultimately in any project as a group.

Appendix A – Copyright Permissions

For Microchip's PIC Family Datasheet

Microchip documentation (including but not limited to data sheets, manuals, etc.), images, website, and other original creations are valuable assets protected by copyright law. Microchip aims to protect these assets while encouraging the broad dissemination of product literature and related information in order to reach the widest market for Microchip products. To that end, we often consider requests by customers, distributors, and other parties to reproduce, translate, and/or reprint our copyrighted material in a book, CD, magazine, or other reference material to be sold or distributed on the open market. When we approve such requests, Microchip still owns all rights to the copyrighted material including any translations of such material.

If you would like to reproduce, translate, and/or reprint Microchip copyrighted material for commercial purposes you must request Microchip's written permission by following the 3-step process described below. Microchip's written permission is not required for personal use or educational (non-profit) use of copyrighted material.

Educational and Non-Profit Use of Copyrighted Material: If you use Microchip copyrighted material solely for educational (non-profit) purposes falling under the "fair use" exception of the U.S. Copyright Act of 1976 then you do not need Microchip's written permission. For example, Microchip's permission is not required when using copyrighted material in: (1) an academic report, thesis, or dissertation; (2) classroom handouts or textbook; or (3) a presentation or article that is solely educational in nature (e.g., technical article published in a magazine). Please note that offering Microchip copyrighted material at a trade show or industry conference for the purpose of promoting product sales does require Microchip's permission.

For BlueGiga Bluetooth Module (WT12 Datasheet)
Tim Eskew <tim.eskew@bluegiga.com>
Mon 7/29/2013 9:12 AM

Hello Henry,

Thank you for your email. In terms of your request to use materials from our data sheet, this is okay as long as it is noted that is copyrighted material from Bluegiga and cannot be redistributed or copied without our approval.

Kind regards,

Tim Eskew
Vice President Americas
Bluegiga
770-291-2181

On 7/28/13 7:25 PM, "henry.chan@knights.ucf.edu"
<henry.chan@knights.ucf.edu> wrote:

>First Name : Henry
>Last Name: Chan
>E-mail : henry.chan@knights.ucf.edu
>Company : University of Central Florida
>Country : United States
>Bluegiga Product : WT12
>Question: Hello, I am a student at the University of Central Florida and
>I need to use the Datasheet for the Bluetooth module WT12 for my research
>project. We need permission to use the diagrams, tables, and schematics in
>the datasheet for our research paper. We will properly cite the
>documentation but we still need approval from BlueGiga to use the
>pictures.

For Texas Instruments (LM7805C Datasheet)
support@ti.com
Tue 7/30/2013 4:24 PM

Hello Henry,

Please refer to our Use Restrictions section under the Terms of Use off our website. The link is listed below:

http://www.ti.com/corp/docs/legal/termsofuse.shtml

Thank you for contacting Texas Instrument Customer Support. If you have any additional questions or concerns please feel free to contact us.

Regards,
Sarah Miksa
TI Customer Support
Americas Customer Support Center
512-434-1560

Use Restrictions

The Materials contained on this site are protected by copyright laws, international copyright treaties, and other intellectual property laws and treaties. Except as stated herein, these Materials may not be reproduced, modified, displayed or distributed in any form or by any means without TI's prior written consent.

TI grants permission to download, reproduce, display and distribute the Materials posted on this site solely for informational and non-commercial or personal use, provided that you do not modify such Materials and provided further that you retain all copyright and proprietary notices as they appear in such Materials. TI further grants to educational institutions (specifically K-12, universities and community colleges) permission to download, reproduce, display and distribute the Materials posted on this site solely for use in the classroom, provided that such institutions identify TI as the source of the Materials and include the following credit line: "Courtesy of Texas Instruments". Unauthorized use of any of these Materials is expressly prohibited by law, and may result in civil and criminal penalties. This permission terminates if you breach any of these terms and conditions. Upon termination you agree to destroy any Materials downloaded from this site.

For Silicon Labs (CP2102 Datasheet)

Tabitha Parker (SiLabs) <MCU.Support@silabs.com>
Tue 7/30/2013 10:43 AM

Hi Henry,

Yes, you may use the diagrams and schematics from the datasheet for your documentation.

Please let us know if you have any other questions.

Best Regards,
Tabitha

Below is a summary of your request:

Ticket ID : MCU-17235
Part Number : CP2102
Support Request: Request to Use the Diagram/Pictures of the CP2102 Datasheet
Priority : Trivial

Request Details

Hello, I am a student at the University of Central Florida and I need to use the Datasheet for the USB to USART bridge (cp2102) for my research project. We need permission to use the diagrams and schematics in the datasheet for our research paper. We will properly cite the documentation but we still need approval from Silicon Labs to use the pictures.

CC: Don P. Williams; Chris M. League

Full Name

For Locosys Wireless
From: Tom W. Warner <TommyW@locosystech.com>
Sent: Tuesday, October 29, 2013 4:27 AM
To: e.hernandez2009@knights.ucf.edu
Cc: Support GPS
Subject: RE: [support-gps] Using Pictures and Tables from your GPS Receivers

Hi Eric,
Yes you may.

Best Regards,

Tom W. Warner
Technical Support Manager
Mobile: (+852) 9759 9622
Email: TommyW@locosystech.com

From: e.hernandez2009 [mailto:e.hernandez2009@knights.ucf.edu]
Sent: Monday, October 28, 2013 10:43 AM
To: support@locosystech.com
Subject: [support-gps] Using Pictures and Tables from your GPS Receivers
To Whom it may concern,

I currently am a senior at The University of Central Florida taking a senior design course in which I would like to use your GPS module A2100-B. We are making a controlled surveillance blimp that would basically be able to survey the area given by GPS coordinates. I am writing to you because I am interested in using some pictures and tables in your user manual for this device in my report and would need permission to include them. I would reference that the pictures were taken from your user manual and include credit in the appendix section of our report.

Thank you,

Eric Hernandez
Undergraduate Electrical Engineering

Sanjay
Email Address
s.yerra24@knights.ucf.edu
Address
FLORIDA 32817
Telephone Number
Mobile Phone

Subject
request use of information on your datasheet
REVIEW YOUR MESSAGE
Hello, I am a student studying electrical engineering at the University of Central Florida. I am currently in a senior design group that is trying to develop a program on the Android platform. Because I am an owner of this phone, I opted to learn to develop on this particular phone. I request permission for the use of the information that is on your datasheet. The information will be cited properly. Thank you, Sanjay Yerra

STATUS: Approved

Web Site Content

To:

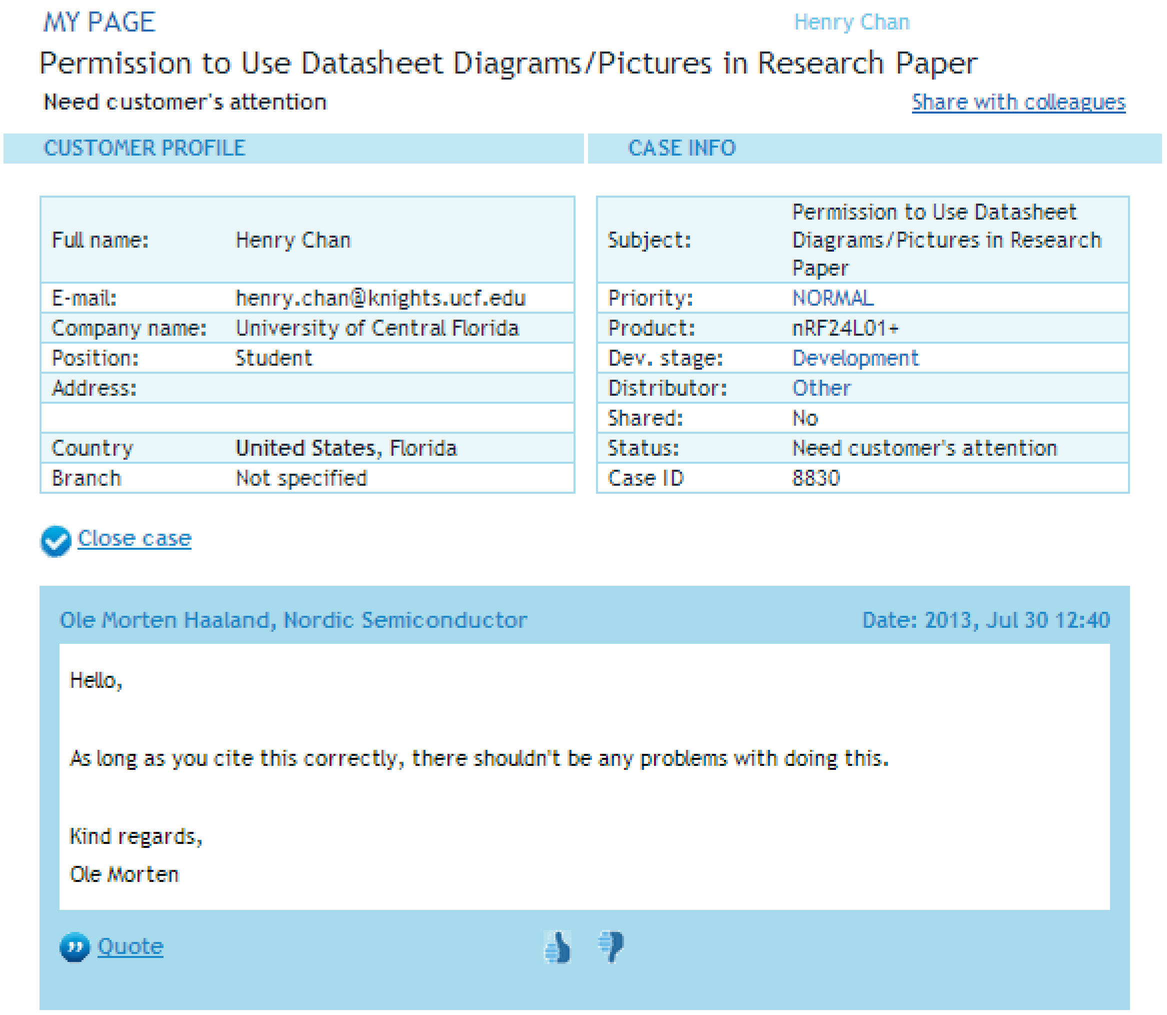
customerservice@atmel.com;

Hello, I am a student studying electrical engineering at the University of Central Florida. I am currently in a senior design group that is trying to design and build an Autonomous Surveillance Blimp. I have heard great things about your products and was recommended the AVR brand of microcontrollers by the professor himself. I request permission to use data and information from the data sheets for your AVR ATmega128 and your AVR ATmega 328.
thank you,
Sanjay Yerra
STATUS: Approved

LIMITED LICENSE TO ATMEL'S MATERIALS

Materials from this website www.atmel.com and any other website owned, operated or controlled by Atmel and/or its affiliated or subsidiary companies (together, Atmel) are owned and copyrighted by Atmel. Unauthorized use of such Materials (e.g., information, documentation and software), including these Terms, may be a violation of Atmel's intellectual property rights or other applicable laws. If you agree to these Terms, you may download (on a single computer), copy or print a single copy of all or a portion of the Materials for informational, non- commercial, lawful purposes only. You may distribute free copies of the documentation available at this website only to customers and prospective customers of Atmel's products. Any other distribution to third parties is strictly prohibited unless you obtain the prior written consent of Atmel. You may not modify in any way any of the Materials contained herein, or delete or modify any of Atmel's copyright, trademark or other proprietary notices. Unauthorized commercial use of these Materials includes without limitation, public display, performance, sale or rental. This Limited License, as well as the Materials, are non-transferable and does not give you any title or intellectual property rights to the Materials. You may not decompile, disassemble, reverse engineer, or otherwise convert Atmel's software Materials except and only to the extent permitted by law.

For Nordic Semiconductors (nRF24L01 DataSheet)



MY PAGE Henry Chan

Permission to Use Datasheet Diagrams/Pictures in Research Paper

Need customer's attention Share with colleagues

CUSTOMER PROFILE	
Full name:	Henry Chan
E-mail:	henry.chan@knights.ucf.edu
Company name:	University of Central Florida
Position:	Student
Address:	
Country	United States, Florida
Branch	Not specified

CASE INFO	
Subject:	Permission to Use Datasheet Diagrams/Pictures in Research Paper
Priority:	NORMAL
Product:	nRF24L01+
Dev. stage:	Development
Distributor:	Other
Shared:	No
Status:	Need customer's attention
Case ID	8830

Close case

Ole Morten Haaland, Nordic Semiconductor Date: 2013, Jul 30 12:40

Hello,

As long as you cite this correctly, there shouldn't be any problems with doing this.

Kind regards,
Ole Morten

Quote

For Maestro Wireless
From: Larry M. Patricio <larry.patricio@maestro-wireless.com>
Sent: Monday, July 29, 2013 4:27 AM
To: e.hernandez2009
Cc: Support GPS
Subject: RE: [support-gps] Using Pictures and Tables from your GPS Receivers A2100-A/B
Hi Eric,
Yes you may do so and would you send us a copy of your user manual once you are able to publish it.
Best Regards,

CW 100 maestro empowering wireless M2M Gateway - GPS Receiver - Design Services *	**Larry M. Patricio** **Technical Support Manager** **Mobile: (+852) 9759 9635** **Email: larry.patricio@maestro-wireless.com**
For more information, visit: Website | Twitter | News Feed	

From: e.hernandez2009 [mailto:e.hernandez2009@knights.ucf.edu]
Sent: Monday, July 29, 2013 10:43 AM
To: support-gps@maestro-wireless.com
Subject: [support-gps] Using Pictures and Tables from your GPS Receivers A2100-A/B

To Whom it may concern,

I currently am a senior at The University of Central Florida taking a senior design course in which I would like to use your GPS module A2100-B. We are making an Android controlled surveillance blimp that would basically be able to survey the area given by GPS coordinates. I am writing to you because I am interested in using some pictures and tables in your user manual for this device in my report and would need permission to include

them. I would reference that the pictures were taken from your user manual and include credit in the appendix section of our report.

Thank you,

Eric Hernandez
Undergraduate Electrical Engineering

Appendix B – References

"C# Versus Java." *C# Versus Java*. The Object - Orientated Project Software Engineering CA4, n.d. Web. 14 July 2013. <http://www.computing.dcu.ie/~renaat/projects/cvjava.html>.

Microchip Datasheet for PIC18F2455/2550/4455/4550. Document DS39632C, 2006. Microchip Technology Inc. 10 July 2013. <ww1.microchip.com/downloads/en/devicedoc/39632c.pdf>

Accelerometer Datasheet for MMA8453Q. Document MMA8453Q, 2013. Freescale Semiconductor. 17 July 2013. http://www.mouser.com/ProductDetail/Freescale-Semiconductor/MMA8453QT/?qs=sGAEpiMZZMvwE4h8i4g3cgOIRT3qYnDDNy2U%252bA9j0jg%3d

Gyroscope Datasheet for L3G4200D. Document ID17116Rev 3, 2010. STMicroelectronics. 17 July 2013. http://www.st.com/web/en/resource/technical/document/datasheet/CD00265057.pdf

www.surveyor .com

GPS Module Datasheet for GPS Receivers A2100-A/B. Document GPS Receiver, 2011.
A Description of Maestro's GPS Receiver Module A2100-A/B User's Manual Version 0.8. 17 July 2013.
http://www.mouser.com/catalog/specsheets/A2100_B.pdf

Antenna Datasheet for GPS Active Antenna Mode. Document ISO9001, 2013. GPS Active Antenna Module. 18 July 2013
http://www.mouser.com/ds/2/3/APAMP-113-245294.pdf

Bluegiga Datasheet for WT12 Bluetooth Module. Version 2.4, 11 January 2007. Bluegiga Technologies. 11 July 2013. <http://www.bluegiga.com/WT12_Class_2_Bluetooth_Module>

Silicon Labs Datasheet for CP2102. Rev. 1.8, 5/07. Silicon Laboratories. 11 July 2013.
<https://www.silabs.com/Support%20Documents/TechnicalDocs/CP2101.pdf>

www.lockheedmartin.com

Texas Instruments Datasheet for LM7805C. Rev D, 08 Apr 2013. Texas Instruments Inc. July 13, 2013.
<http://www.ti.com/product/lm7805c>

Nordic Semiconductors Datasheet for nRF24L01. Version 2, July 2007. Nordic Semiconductors. 11 July 2013.
<http://www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01>

Spread Spectrum Technologies Powerpoint Presentation. 1 September 2006. NVCC. 10 July 2013.
<http://engweb.info/courses/wdt/resources/Lecture%204%20-%20Spread%20Spectrum%20Technologies.ppt>

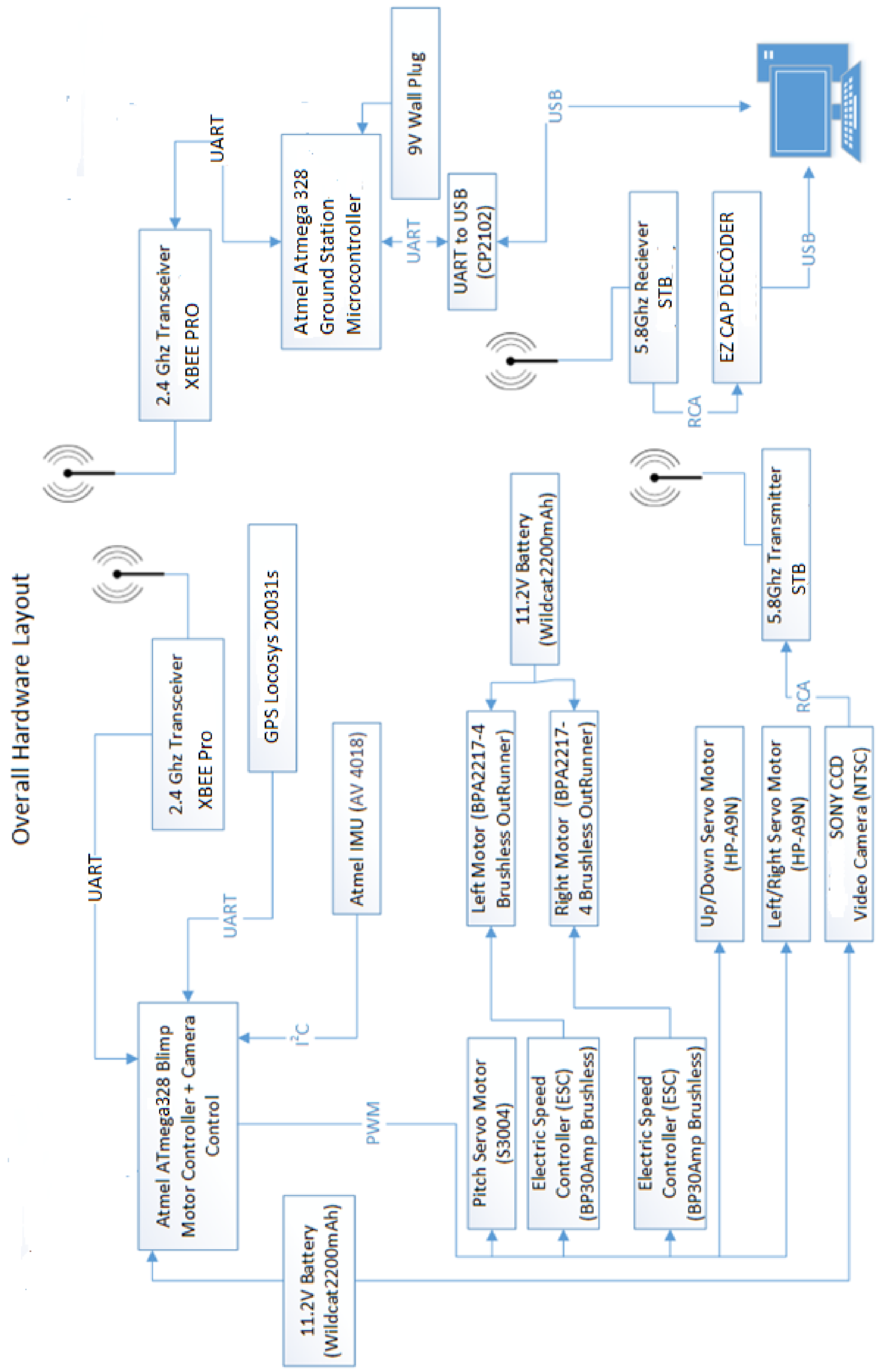
Atmel,2013.AVR ATmega128 Datasheet. Rev. 2467X-June 2011. Atmel Corporation. 20 July 2013.
<http://www.atmel.com/Images/doc2467.pdf>

www.eblimp.com

Atmel,2013.AVR ATmega328 Datasheet. Rev. 8161D July 2009. Atmel Corporation. 20 July 2013.
<http://www.atmel.com/Images/doc2467.pdf>

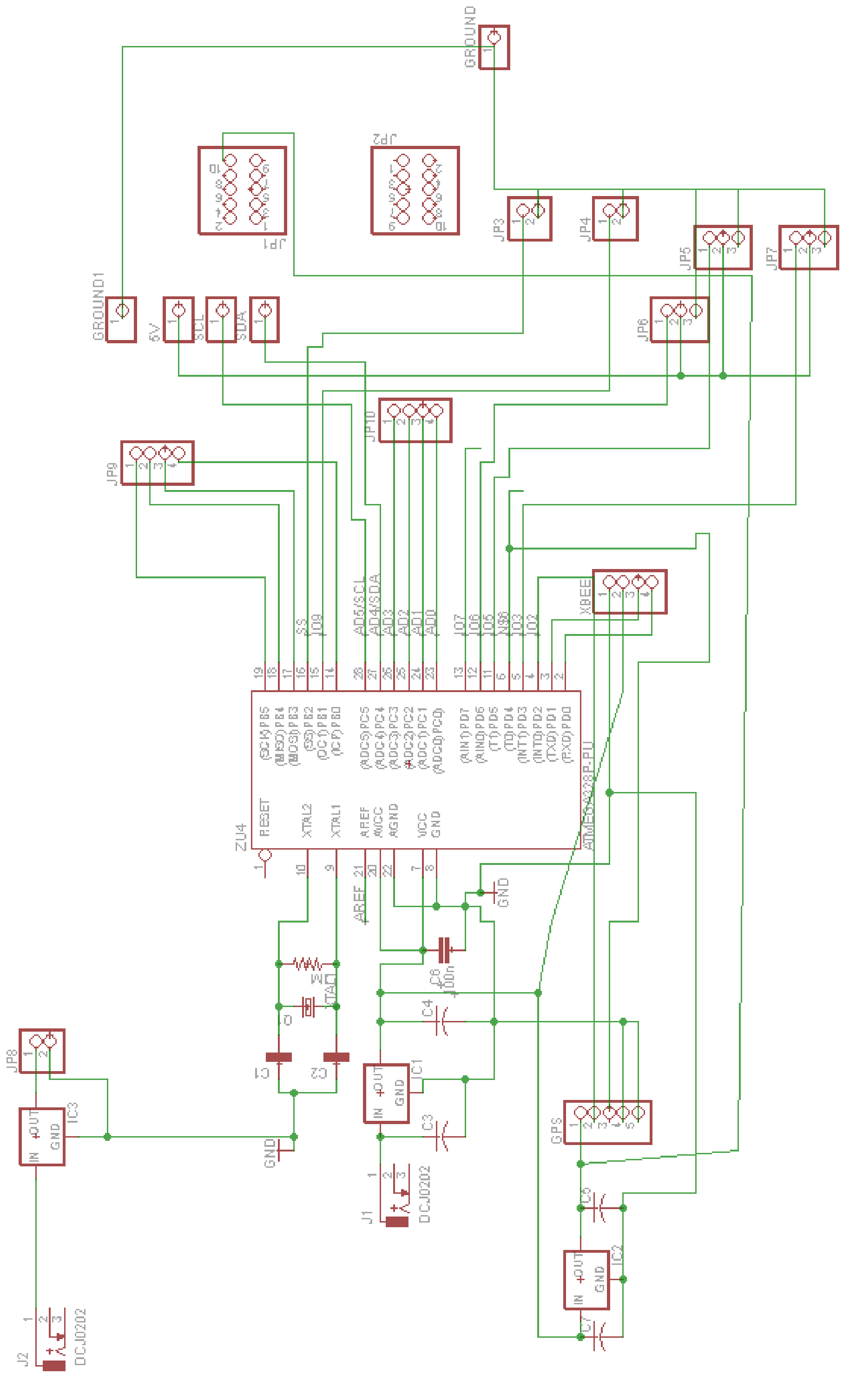
"e_Datasheet for Optimus_V_LG_Final." *LG*. LG. Web. 22 July 2013.
<http://www.virginmobileusa.com/resources/phones/prepaid/manual/lgoptimusv.pdf>.

Appendix C – Full Hardware Block Diagram

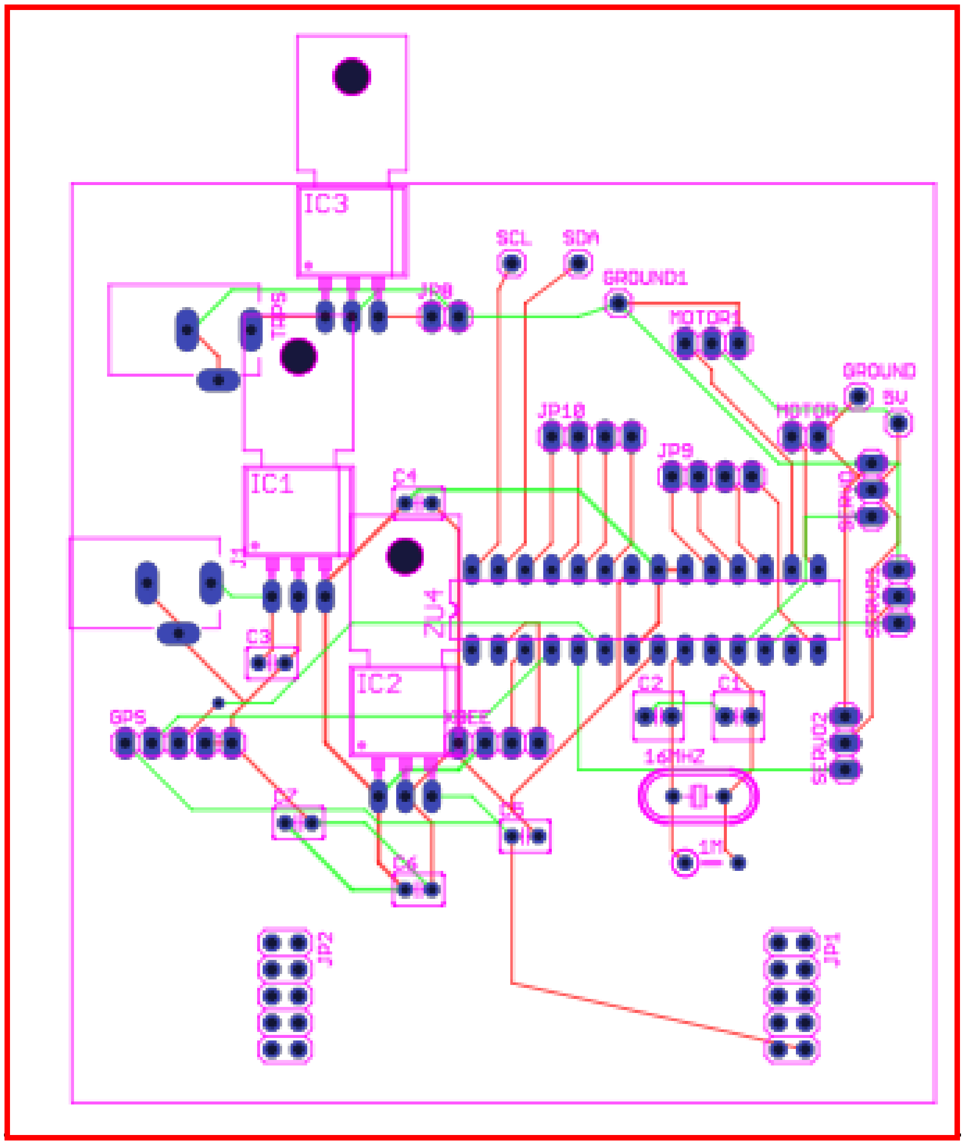


Overall Hardware Layout
2.4 Ghz Transceiver XBEE Pro
UART
GPS Locosys 20031s
UART
Atmel IMU (AV 4018)
I²C
Atmel ATmega328 Blimp Motor Controller + Camera Control
PWM
11.2V Battery (Wildcat2200mAh)
Left Motor (BPA2217-4 Brushless OutRunner)
Right Motor (BPA2217-4 Brushless OutRunner)
11.2V Battery (Wildcat2200mAh)
Pitch Servo Motor (S3004)
Electric Speed Controller (ESC) (BP30Amp Brushless)
Electric Speed Controller (ESC) (BP30Amp Brushless)
Up/Down Servo Motor (HP-A9N)
Left/Right Servo Motor (HP-A9N)
SONY CCD Video Camera (NTSC)
RCA
5.8Ghz Transmitter STB
2.4 Ghz Transceiver XBEE PRO
UART
Atmel Atmega 328 Ground Station Microcontroller
9V Wall Plug
UART
UART to USB (CP2102)
USB
5.8Ghz Reciever STB
RCA
EZ CAP DECODER
USB

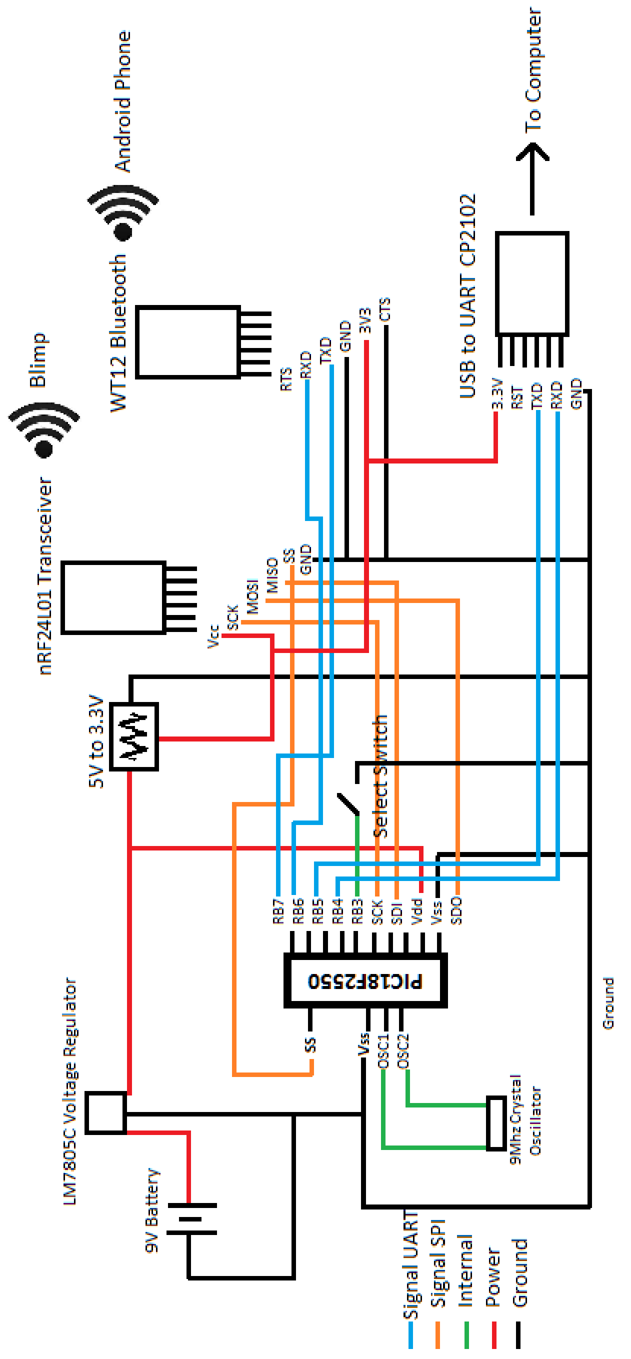
Appendix D-1 -Full Layout of Blimp Controller Board Design



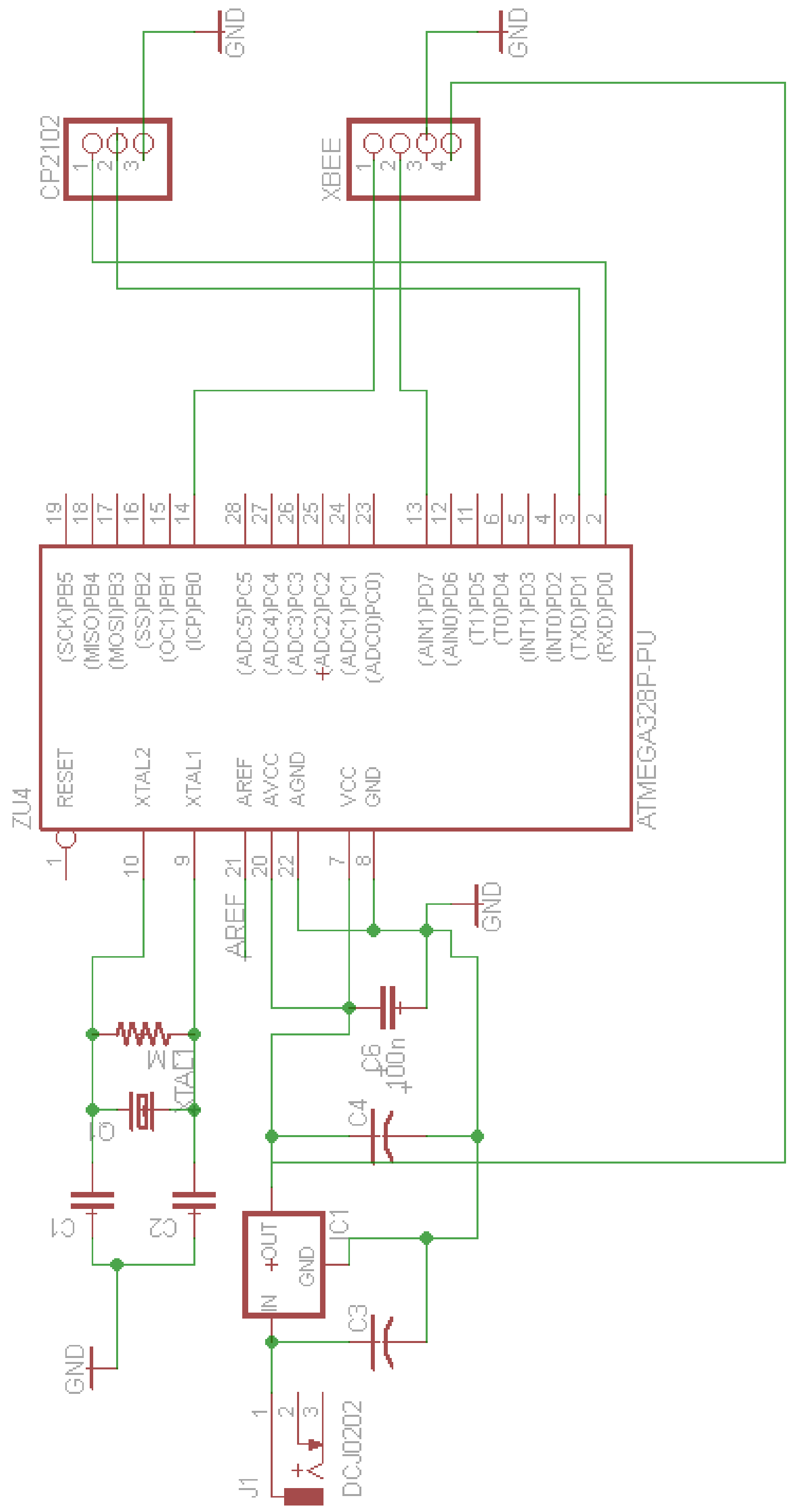
Appendix D-2 – Printed Circuit Board Layout



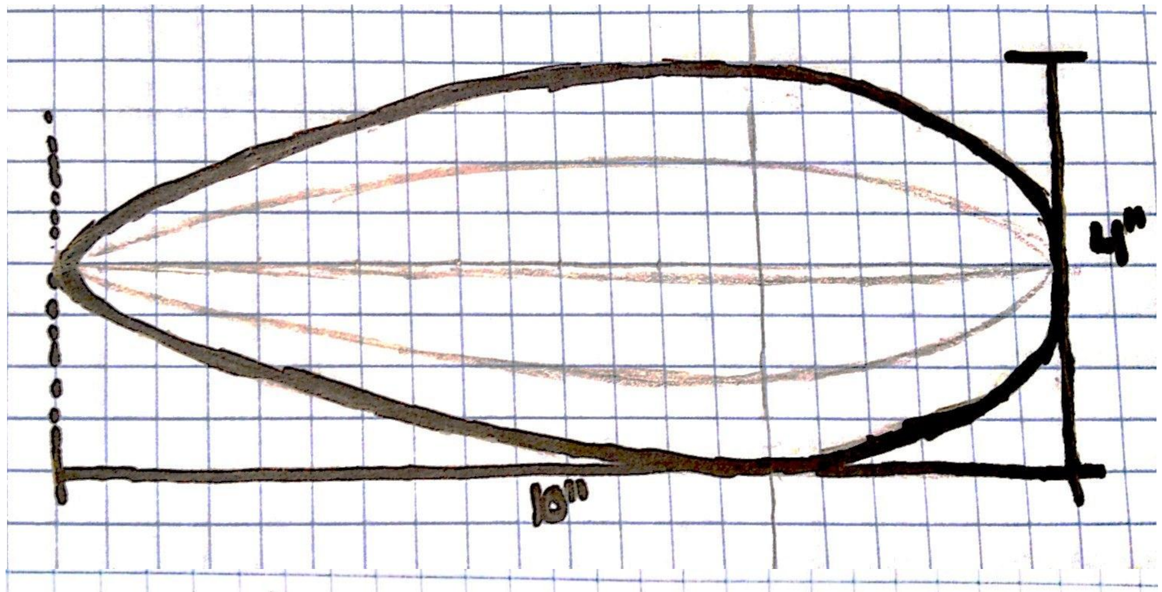
Appendix E-1 - Old Layout of Ground Station Design

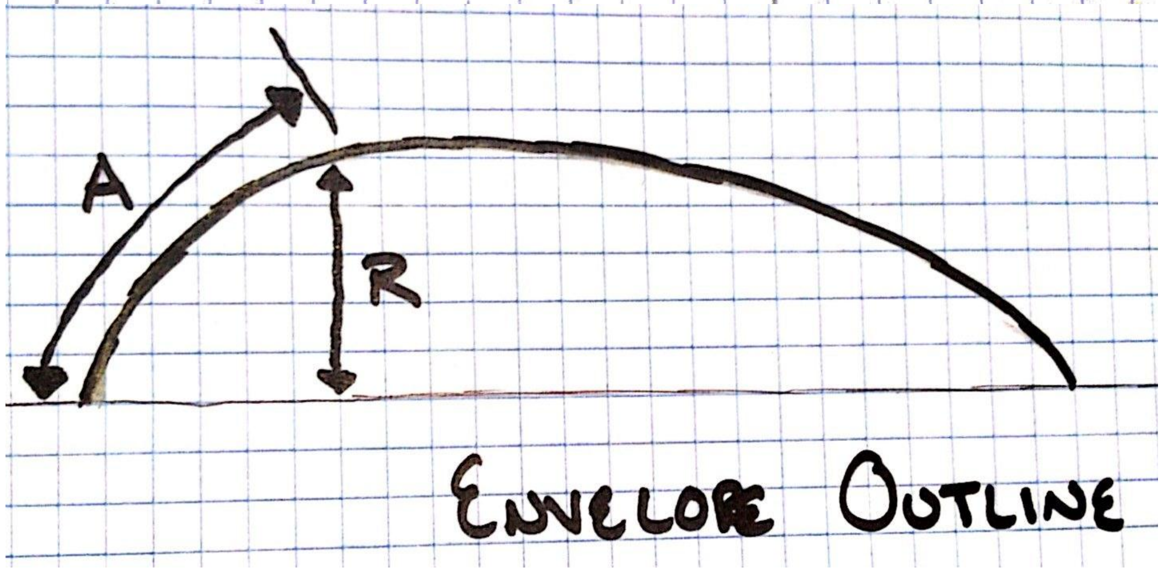


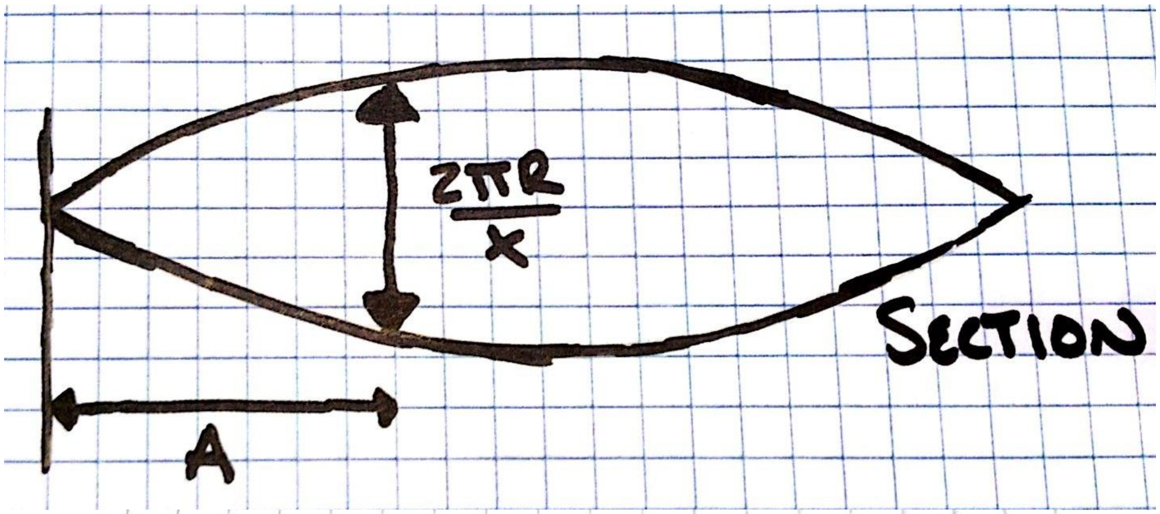
Appendix E-2 – Final Layout of Ground Station

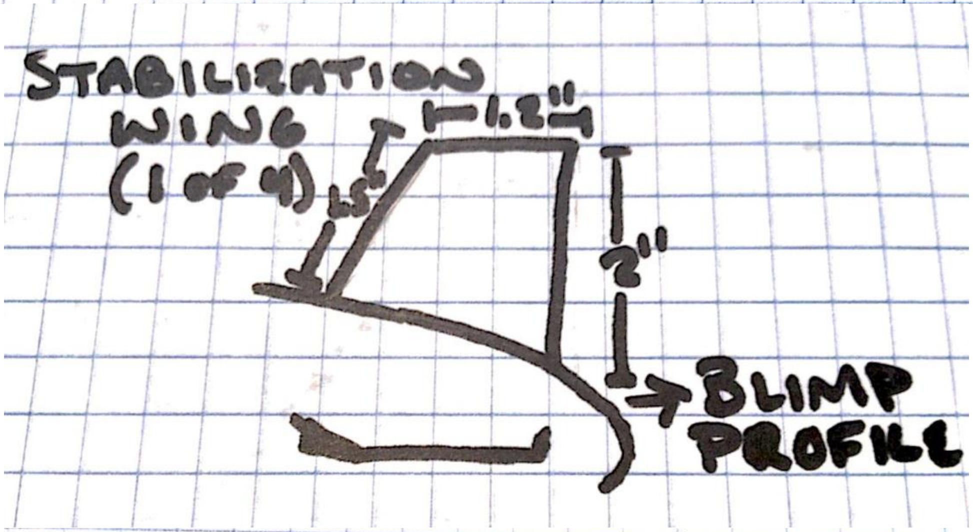


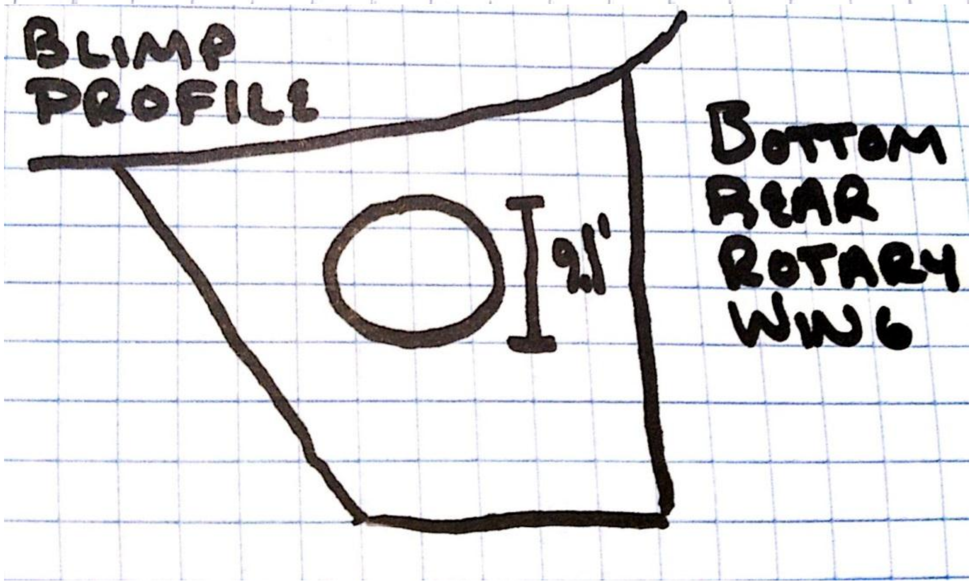
Appendix F – Full Layout of Blimp Design

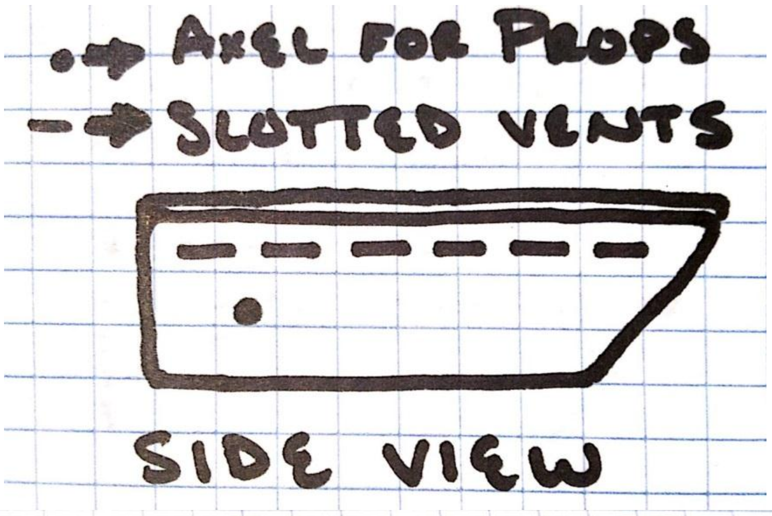


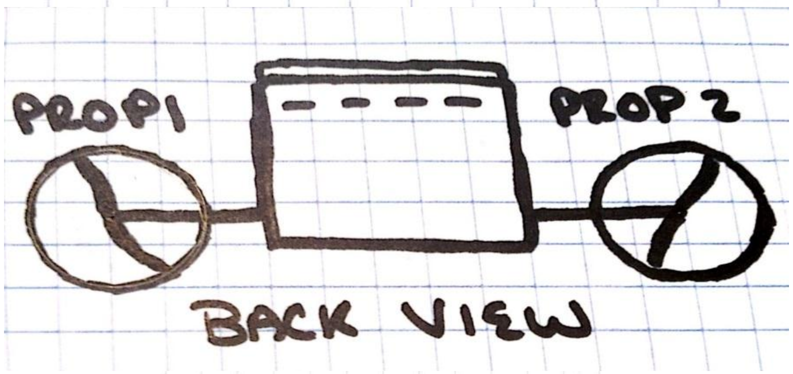


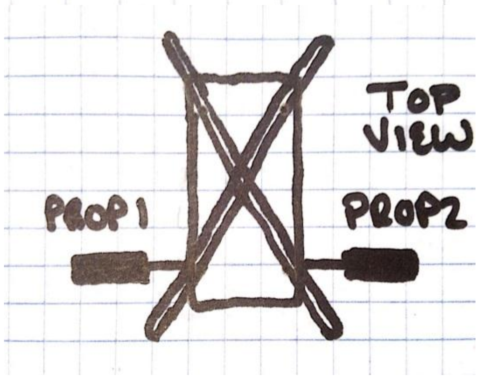

$\frac{2\pi R}{x}$
A
SECTION

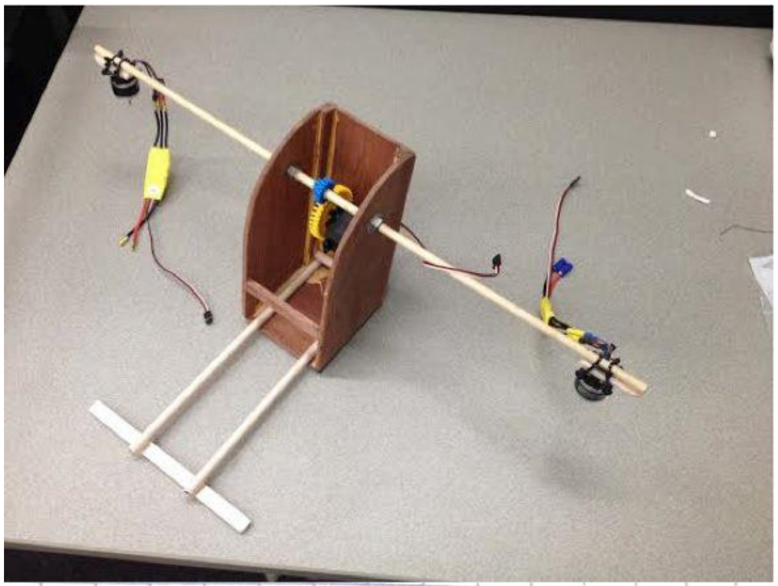

STABILIZATION WING (1 OF 4)
1.2"
1.5"
2"
BLIMP PROFILE

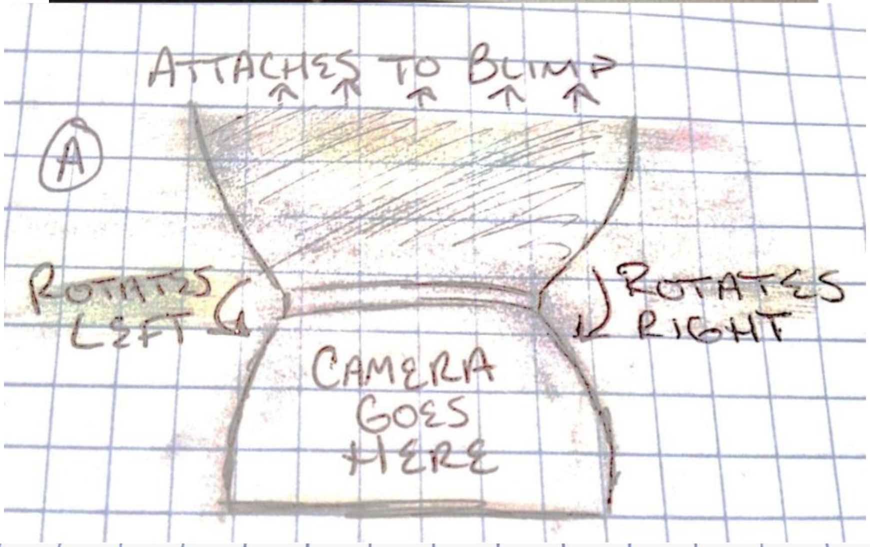

BLIMP PROFILE
2.5"
BOTTOM REAR ROTARY WING

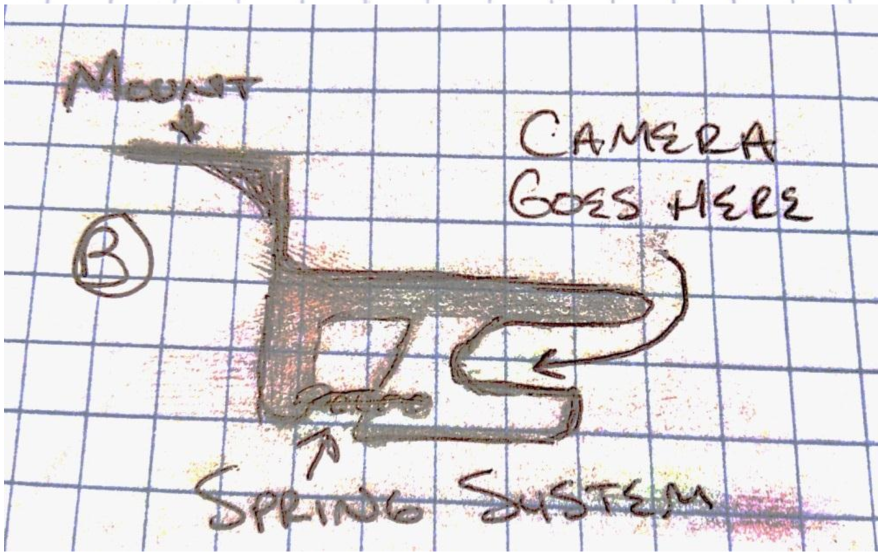

AXEL FOR PROPS
SLOTTED VENTS
SIDE VIEW

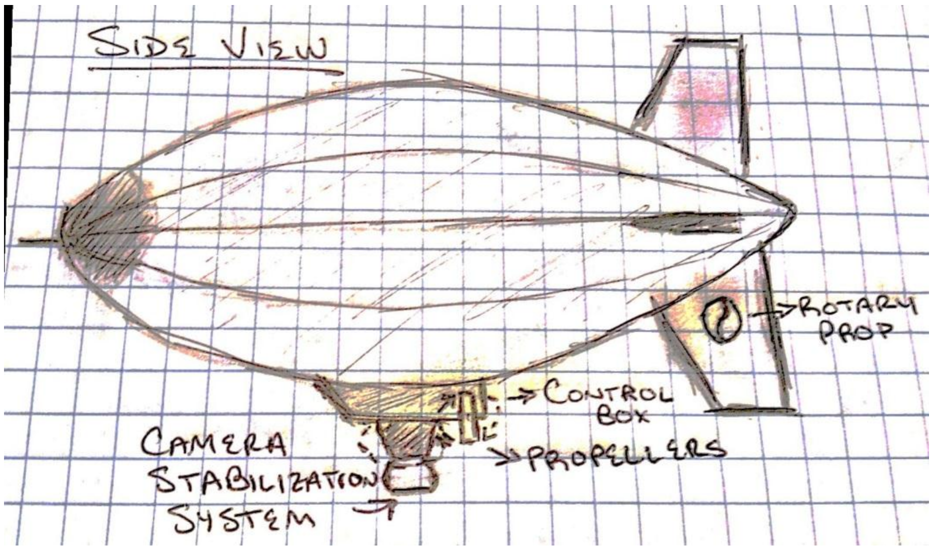

PROP1
PROP 2
BACK VIEW

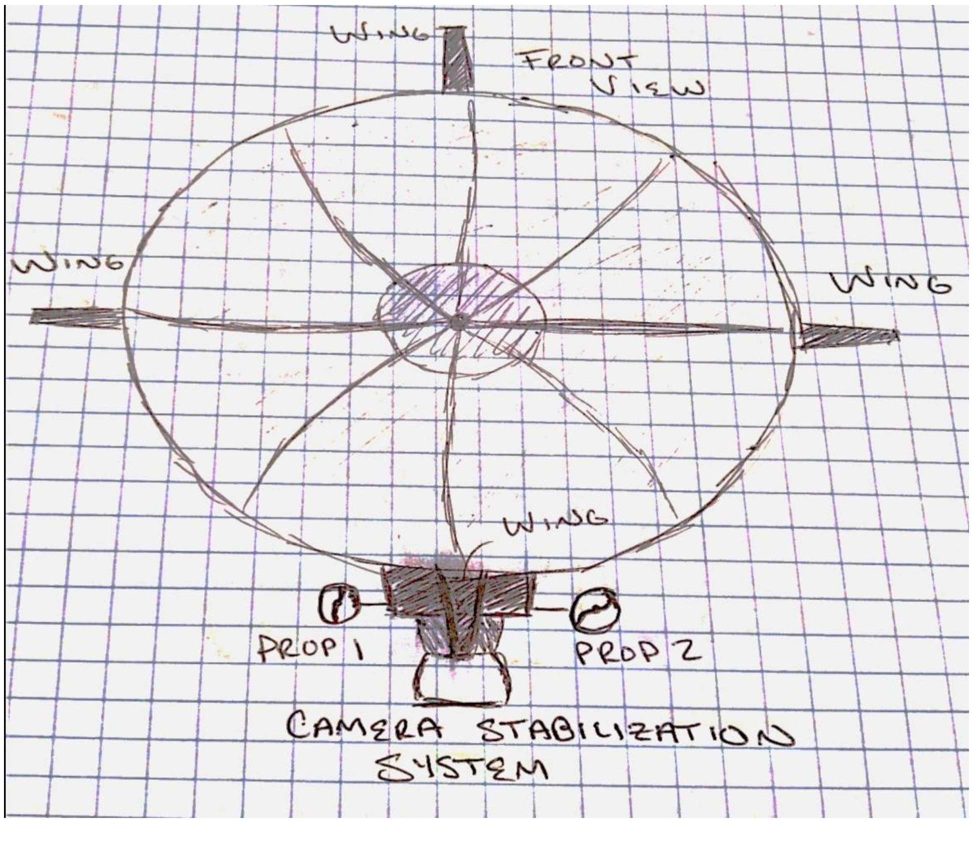

TOP VIEW
PROP1
PROP2




ATTACHES TO BLIMP
A
ROTATES LEFT
ROTATES RIGHT
CAMERA GOES HERE


MOUNT
B
CAMERA GOES HERE
SPRING SYSTEM


SIDE VIEW
ROTARY PROP
CONTROL BOX
PROPELLERS
CAMERA STABILIZATION SYSTEM


WING
FRONT VIEW
WING
WING
WING
PROP 1
PROP 2
CAMERA STABILIZATION SYSTEM