

S.H.A.S.bot

Group #9

Team Members:
Daniel Lanzone
Mike Roosa
Ryan Tochtermann
Charlie Grubbs

December 3, 2012

Table of Contents

Executive Summary	1
1 Overview	2
1.1 Motivation.....	2
1.2 Objectives	2
1.3 Requirements and Specifications	2
1.4 Division of Labor	3
1.5 Project Milestones	3
1.6 Project Financing	4
2 Research.....	5
2.1 Chassis	5
2.1.1 Introduction	5
2.1.2 Requirements.....	5
2.1.3 Wheels	6
2.1.4 Tracks.....	6
2.1.5 Motors	7
2.1.6 Mounted peripherals	8
2.2 Power	8
2.2.1 Battery Technologies	8
2.2.2 Current Sensing	10
2.2.3 Voltage Sensing	13
2.2.4 Reverse Polarity Protection	14
2.2.5 Voltage Regulation.....	17
2.3 Stabilization Platform Research	20
2.3.1 Inertial Measurement Unit	20
2.3.2 Servo Motors.....	22
2.3.3 Stabilization Platform Control	24
2.3.4 Microcontroller.....	26
2.4 Image processing	27
2.4.1 Analog image processing	27
2.4.2 Digital image processing	28
2.4.3 Video input	28
2.4.4 Video Compression	28

2.4.5	Compression Artifacts.....	29
2.4.6	Camera selection	29
2.5	Software	31
2.5.1	Image Processing Software.....	31
2.5.2	Control Software	35
2.5.3	Graphical User Interface Software	36
2.5.4	Processing	39
2.6	GPS.....	40
2.6.1	GPS – Introduction.....	40
2.6.2	GPS – Requirements.....	40
2.6.3	GPS Communication	42
2.7	Communication Systems.....	42
2.7.1	Wired Channel Communication.....	43
2.7.2	Wireless Channel Communications	45
2.7.3	Wireless connectivity (Wi-Fi) 802.11	46
2.7.4	Sensor and Data Fusion	47
2.8	Navigation	47
2.8.1	Autonomous Navigation	47
2.8.2	Routing	48
2.8.3	Perimeter search/scan.....	48
2.8.4	Obstacle avoidance	48
2.9	Audio	49
2.9.1	Audio Circuit.....	49
2.9.2	Microphone Modification.....	50
2.9.3	Microphone Mounting	53
2.9.4	Microphone Arrangement	54
2.9.5	Audio Amplification	58
2.9.6	Audio Filtering	60
2.9.7	Voltage Comparator.....	61
2.9.8	Audio Arduino Input (Analog)	62
2.9.9	Arduino Audio Control Loop	63
3	Design	64
3.1	Chassis	64
3.1.1	Frame	64

3.1.2	Wheels	65
3.1.3	Motor Selection	65
3.1.4	Motor Controller Selection	66
3.1.5	GPS	67
3.2	Power	70
3.2.1	Battery Selection	70
3.2.2	Current Sensing Design	70
3.2.3	Voltage Sensing Design	72
3.2.4	Reverse Polarity Protection Design	72
3.2.5	Voltage Regulation.....	73
3.3	Stabilization Platform Design	75
3.4	Navigation	77
3.4.1	Autonomous navigation mode.....	77
3.4.2	Auto-routing.....	77
3.5	Communication Systems.....	78
3.5.1	XBee-Pro Wireless Module	78
3.5.2	Embedded-side Communications	81
3.5.3	PC-side Communications	82
3.5.4	Wireless Communications System.....	83
3.5.5	Camera.....	86
3.5.6	Networking and security	86
3.6	Software	87
3.6.1	Graphical User Interface and PC-Side Control.....	87
3.6.2	Image Processing	90
3.7	Audio	93
3.7.1	Microphone Connection to Audio Board	93
3.7.2	Audio GUI Display.....	94
3.7.3	Defining a Critical Audio Event.....	95
3.7.4	Audio Parts Selection	97
3.7.5	Audio Board Circuit Schematics.....	97
4	Testing and configuration.....	98
4.1	Communications System.....	98
4.1.1	Camera link	102
4.1.2	Camera range	103

4.1.3	GPS link	104
4.1.4	GPS accuracy	105
4.2	User Interface and Controls Testing	105
4.2.1	LCD output	106
4.3	Image Processing	108
4.3.1	Motion detection.....	108
4.3.2	Gating	109
4.3.3	Centroid calculation	109
4.3.4	Tracking.....	110
4.4	Navigation	111
4.4.1	Auto-route.....	111
4.4.2	Obstacle avoidance	111
4.5	Stabilization Platform Testing Procedure.....	112
4.6	Audio System Testing.....	113
4.6.1	Microphone Voltage Testing Procedure (Audio Test 1)	113
4.6.2	Audio Board Testing Procedure (Audio Test 2)	115
4.6.3	Audio Alert System Testing Procedure (Audio Test 3).....	117
4.7	PCB Design	113
5	Project Summary.....	121

Table of Figures

FIGURE 2.1.1 – GENERAL DESIRED CHASSIS SHAPE AND DIMENSIONS (9.75" x 8" x 2.30")	7
FIGURE 2.2.2 – CLOSED LOOP HALL EFFECT DIAGRAM.....	12
FIGURE 2.2.3 – DIODE PROTECTION SCHEMATIC	15
FIGURE 2.2.4 – FULL-WAVE RECTIFICATION PROTECTION SCHEMATIC	16
FIGURE 2.2.5 – PMOS PROTECTION SCHEMATIC	16
FIGURE 2.3.1 – CONTROL LOOP FLOW CHART.....	25
FIGURE 2.4.1 – ANALOG TO DIGITAL VIDEO CONVERSION	27
FIGURE 2.4.2 – COMPRESSION ARTIFACTS CAN BE SEEN ON THE RIGHT IMAGE	29
FIGURE 2.5.1 – TRACKING SOFTWARE OVERVIEW	35
FIGURE 2.5.1 – SAMPLE PREVIEW LAYOUT OF GUI APPLET (WRITTEN IN ECLIPSE IDE)	37
FIGURE 2.5.3 – SAMPLE GUI APPLET WRITTEN IN PROCESSING USING CONTROLP5.....	40
FIGURE 2.9.1 – AUDIO SIGNAL PROCESSING CHAIN	50
FIGURE 2.9.2 – HEXAGONAL MICROPHONE PATTERN.....	51
FIGURE 2.9.3 – REDUCED HEXAGONAL MICROPHONE PATTERN	52
FIGURE 2.9.4 – SHOTGUN MICROPHONE MODIFICATION.....	52
FIGURE 2.9.5 – FREE FLOATING MICROPHONE MOUNT	53
FIGURE 2.9.6 – SECURE MICROPHONE MOUNT.....	54
FIGURE 2.9.7 – TRIAD MICROPHONE CONFIGURATION	55
FIGURE 2.9.8 – QUADRUPLE MICROPHONE CONFIGURATION	56
FIGURE 2.9.9 – QUINTUPLET MICROPHONE CONFIGURATION	57
FIGURE 2.9.10 – SEXTUPLET MICROPHONE CONFIGURATION	58
FIGURE 2.9.11 – AMPLIFICATION AND FILTERING CHAIN	60
FIGURE 2.9.12 – COMPARATOR I/O DIAGRAM	62

FIGURE 2.9.13 - AUDIO PROCESSING CONTROL LOOP	63
FIGURE 3.1.1 – NIKKO RC BUGGY CHASSIS	64
FIGURE 3.1.4 – PANASONIC 0.2 F CAPACITOR	68
FIGURE 3.1.5 – SCHEMATIC WITH ATMEGA328 (TX + RX)	69
FIGURE 3.1.6 – VENUS GPS MODULE HEADER SCHEMATIC	69
FIGURE 3.2.1 - MAX4070 SENSOR DIAGRAM	70
FIGURE 3.2.2 – MAX4373 SENSOR DIAGRAM	71
FIGURE 3.2.3 – VOLTAGE SENSE SCHEMATIC	72
FIGURE 3.2.4 – 5V REGULATOR SCHEMATIC	74
FIGURE 3.2.5 – 5V+ REGULATOR SCHEMATIC	74
FIGURE 3.2.6 – 3.3V REGULATOR SCHEMATIC	75
FIGURE 3.3.1 – STABILIZATION PLATFORM SCHEMATIC	76
FIGURE 3.4.1 – AUTO-ROUTING FLOWCHART	78
FIGURE 3.6.1 – GENERAL IMAGE PROCESSING SOFTWARE OVERVIEW	91
FIGURE 3.6.1 – LUKAS-KANADE METHOD OF OPTICAL FLOW TRACKING	92
FIGURE 3.6.2 – TRACK SOFTWARE ROUTINE	93
FIGURE 3.7.1 – AUDIO INPUT JACK BREAKDOWN	94
FIGURE 3.7.4 – SOUND LEVEL PICKUP REGIONS	96
FIGURE 3.7.5 – AUDIO AMPLIFICATION CIRCUIT	97
FIGURE 4.1.1 – XBEE MODULE TO XBEE EXPLORER CIRCUIT SCHEMATIC	99
FIGURE 4.1.2 – CHANGING THE XBEE'S FUNCTION SET TO ZNET 2.5 COORDINATOR AT	100
FIGURE 4.1.3 – VERIFYING XBEE PARAMETERS IN COMMAND MODE	101
FIGURE 3.1.2 – 16X2 CHARACTER MONOCHROME LCD MODULE	107
FIGURE 3.1.3 – EAGLE SCHEMATIC FOR LCD	107
FIGURE 4.4.1 - VOLTAGE LEVELS IN AUDIO PROCESSING CHAIN FOR 0.5 mV MICROPHONE OUTPUT	115
FIGURE 4.4.2 - GUI FRONT MICROPHONE AUDIO EVENT ALERT VISUAL	118
FIGURE 4.7.1 - POWER BOARD	119
FIGURE 4.7.2 – COMMUNICATION BOARD	120
FIGURE 4.7.3 – STABILIZATION BOARD	121

Table of Tables

TABLE 1.6-1 PROJECTED BUDGET TABLE	4
TABLE 2.2.1 – VOLTAGE SENSING REGULATION OPTIONS	14
TABLE 2.2.2 – REGULATION TYPE COMPARISON	19
TABLE 2.3-1.3.1 – IMU COMPARISON	21
TABLE 2.3.2 – SERVO MOTOR COMPARISON	23
TABLE 2.3.3 – ARDUINO FUNCTIONS	24
TABLE 2.3.4 – MICROCONTROLLER COMPARISON	26
TABLE 2.4.1 – DECISION MATRIX FOR CAMERA SELECTION	30
TABLE 2.6.1 – GPS RECEIVER MODULE COMPARISON	41
TABLE 2.7.1 – RS-232 PIN CONFIGURATION	43
TABLE 3.1.1 – NIKKO RC BUGGY SPECS	64
TABLE 3.1.2 – FRAME (ONLY) DIMENSIONS AND SPECIFICATIONS	65
TABLE 3.1.3 – MOTOR CONTROLLER COMPARISON TABLE	66
TABLE 3.3.1 – STABILIZATION PLATFORM CONNECTIONS LISTING	76
TABLE 3.5.1 – XBEE-PRO MODULE PIN CONFIGURATION	79
TABLE 3.5.2 – XBEE-PRO 2.5 MODULE ELECTRICAL CHARACTERISTICS	80
TABLE 3.6.1 – INPUT AND OUTPUT DATA STRUCTURE FOR MEAN SHIFT CALCULATION	91
TABLE 3.7.1 – DECIBEL LEVEL EXAMPLES	95
TABLE 5.1.1 - FINAL BUDGET	122

Executive Summary

In an increasingly technological world, the idea of using robots as an extension of the person becomes more and more prevalent. The inspiration can be traced back to the adolescence of today's senior engineering students, when many young, future engineers spent carefree days wasting away their youth playing video games, sending virtual robots to spy on their enemies and complete their mission. This project is the manifestation of that inspiration, to take those virtual wonders and bring them to the real world, and to make them readily accessible. Thus, the idea of the S.H.A.S.bot was born.

Using robots to survey an area is generally inconvenient if the video feed is shaky, the image is unclear, and if the user has to turn his or her head more than 45-degrees to not get disoriented by the picture. The S.H.A.S.bot is the solution to this problem. Designed with the intention of being placed on a mobile robot platform, this project will take a mobile, remotely-controlled ground vehicle and attach a camera system. The video-feed of the camera will be wirelessly transmitted to a laptop PC, to give the user a first-person, uneven terrain, the camera system will compensate based on the angle the base of the vehicle has with respect to a non-inclined surface. Thus, if the vehicle is on a 25-degree slope, the camera will compensate by orienting itself -25-degrees, giving the user a flat, non-disorienting perspective to continue operating with. A heads-up display will be generated to inform the user of the angle of the terrain they are on. The auto-correction will have the option of being toggled, so if the user wishes to get a visual idea of the surface they are on they may do so. The S.H.A.S.bot will be controlled by using a combination of gyroscopes and accelerometers to determine the angle of the camera with respect to the ground plane. The angle data will be passed along to the control loop, which will be a combination of analog and digital components, to then adjust the servo system.

We will integrate a stereoscopic sound detection system to interface with the video feed from the camera. The audio awareness system will be fine-tuned to ignore background noise while still being able to pick up specific frequencies of sound. The system will then alert the operator wirelessly via the HUD to the origin of the noise. Beyond those key features, autonomous tracking and routing would both be desirable product features if time constraints allow.

The end product will be a lightweight, portable platform that is sturdy enough to handle the various types of environments it will encounter, with a long enough battery life-span to complete whatever objective the user might have for it. Low cost is also ideal, as there is a high chance for nature to take a toll on the system. The applications itself are limitless. Scientists studying animals in their natural habitat, military reconnaissance, or even home security situations are all scenarios in which this system would be useful.

1 OVERVIEW

1.1 Motivation

The motivation for this project is to learn how to use robotics to interface with the outside world. Robots will become an everyday part of our lives in the future, and the lessons that will be learned throughout the duration of this project will prove invaluable for engineers beginning their career. Beyond the academic motivations, from a consumer standpoint, both the commercial and military markets would have an interest in a robotics platform that has the ability to survey an area and detect noises. For military applications, it could be used to survey a hot-area by showing the general landscape, pinpointing threats and targets, and providing an advantageous view for the warfighters to utilize. From a civilian and commercial standpoint, home and business surveillance/security situations could take full advantage of the system while removing overhead costs for increased security personnel.

1.2 Objectives

The desired technical objectives for the S.H.A.S.bot are as follows:

- Stabilize about the roll and pitch axes with a minimum resolution of 10-bits
- The ability to toggle camera control between user and auto-stabilization
- Obstacle avoidance in autonomous mode
- Directional sound detection
- Basic image processing motion detection and tracking
- Wireless vehicle and platform control
- GUI with power consumption, sound detection, and camera feed

1.3 Requirements and Specifications

- 14.8V lithium ion polymer power source
- Reverse polarity protection
- Current and voltage sensing capabilities with 10-bit resolution
- 5V and 3.3V regulation
- Independent stabilization platform
- Automatic camera correction about roll and pitch axes
- Camera control toggle
- Minimal power consumption
- Wireless vehicle navigation
- Track mobile objects while vehicle is stationary
- Autonomous navigation with accuracy of less than 3 meters
- Tri-directional audio detection with 50dB sensitivity

- Operator HUD with power consumption, audio detection, and vehicle control interfaces

1.4 Division of Labor

The project responsibilities were divided as shown below:

Ryan Tochtermann

- Stabilization platform
- Image processing
- PCB design
- Camera integration

Charles Grubbs

- Sensor and control interfacing
- Wireless integration
- GUI

Michael Roosa

- Audio detection system
- PCB design
- Power systems

Daniel Lanzone

- Power systems
- PCB design

1.5 Project Milestones

General testing and redesign of systems will be an ongoing, daily process. The following dates are general guidelines that we hope to be able to achieve.

- Ordered parts - August 17
- Began building of each system - August 24
- Worked prototype of each independent system - August 31
- Began system integration testing - September 7
- Develop working prototype - September 14
- Began system testing/tuning - September 14
- PCB design finalized - September 21
- PCB ordered - October 5
- PCB printed and received - October 20
- PCB testing begun - October 20
- Tuned software parameters - November 2
- Design Finalized - November 16

1.6 Project Financing

As of the completion of this paper, we have no sponsorships or funding options outside of our own means. The following table reflects the projected cost of our project:

Table 1.6-1 Projected Budget Table

Part name	Description	QTY	Total cost
RC chassis	4WD aluminum chassis w/ Motors	1	\$230.00
MPU-6050	6 Degrees of Freedom 3-axis gyro/accelerometer	1	\$19.95
Cisco Wireless-N Camera	Wireless IP camera	1	\$250.00
Voltage regulators	Voltage regulators for 5 and 3.3 volts	4	\$10.00
Servos	HS-422 Servo Motor	2	\$20.00
Motor controller	Sabertooth 10A R/C motor controller	2	\$139.99
Battery	14.4 V Li-Poly	2	\$99.95
Microphones	Directional (modified)	6	\$60.00
Audio Amplifiers	1.4W Analog Devices	12	\$36.00
Audio Filters	Maxim Active Low Pass Filter	12	\$30.00
Voltage Comparator	Philips 74HC	6	\$6.00
Audio Connector	1/8" Audio Three Pronged Socket	6	\$6.00
Microphone mounting	Mounting equipment	1	\$15.00
Series 2 XBee-Pro RF Modules	Wireless RF Communications Module	2	\$42.00
XBee Explorer USB	Serial to USB Interface for XBee	1	\$24.95
Arduino Uno	Development Board	1	\$26.56
General Electronic Components	Capacitors, resistors, LEDs, switches, transistors, diodes, etc.	1	\$30.00
Venus638FLPx-L	GPS module	1	\$39.95
Antenna	GPS antenna	1	\$9.95
ATMEGA328P-PN	Microcontroller - DIP	1	\$3.31
Total:			\$1216.48

2 RESEARCH

2.1 Chassis

2.1.1 Introduction

The chassis for the system is one of the most important and defining features of the vehicle. It needs to be large enough to support all the desired equipment, lightweight, and rigid. These requirements should be enough to provide a stable platform for all the systems to operate on. The material and costs are also an important concern. Materials too weak or subject to wear won't last long enough and materials too costly can restrict the project through funding.

The platform used dictates many of the components that are at our disposal and could either restrict or expand the overall features. The right size, ground clearance, and traction are all features that are addressed to develop a well-functioning design.

With the requirements in mind, the decision to use wheels was made in order to simplify the design and make more parts available for selection. Track component selection is limited and makes the construction of the chassis slightly more complicated. It also requires more power and weight, which would further limit the systems capabilities.

2.1.2 Requirements

The physical chassis should be at least 8 inches by 9 inches to provide enough area to incorporate the stabilization platform, printed circuit boards, battery, audio detection system, and any managing peripherals needed, such as Wi-Fi or other wireless communication modules. The overall height, length, and width of the system should be, at a minimum, a couple inches smaller in each dimension. This is required so that the wheels or tracks will have enough room to operate in all terrain or standard modes. Operating speeds and the turning radius are also functions of track and wheel diameters, it is important to choose an optimal combination in order to maximize performance in both areas.

The frame requires rigid materials with a hard plastic, aluminum, or lightweight steel construction. The advantages of having a metal construction over plastic or wood is overall strength and tolerances. These advantages allow the performance and versatility of the vehicle to be isolated from these potential hindrances. The wheel or track construction should contain a combination of metal and plastic, along with some other connecting pieces. The performance advantages from this design outweigh the disadvantages of cost associated with a metal construction.

The weight of wheels or tracks should be kept to a minimum. The effects of using wheels or tracks with a higher mass can drastically decrease motor performance and battery life.

Traction and maneuverability are directly related to the choice of wheels or tracks used in the design. Wheels provide a relatively cheap and easy way to interface with DC motors to achieve adequate speeds and maneuverability while also enabling the design to incorporate more rugged terrain modes with larger diameter wheels. Tracks provide better traction and load support, but are more expensive and complicated to configure and operate.

2.1.3 Wheels

Wheels provide a widely available solution for mobile driven platforms. They come in a wide spectrum of diameters, thicknesses, traction patterns, and mounting mechanisms. The driving mechanism behind wheels can be a simple solid axle driven system or a more complicated independently driven system with suspension or four wheel drive.

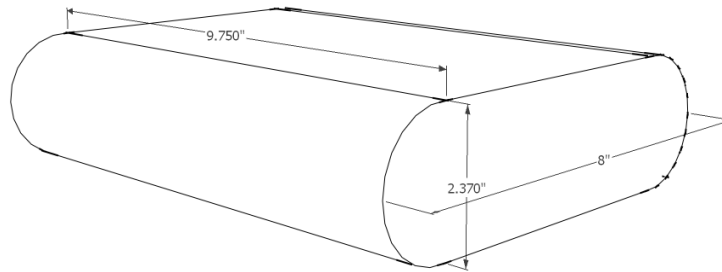
Depending on the maximum torque output of the motors, the wheel size needs to at a minimum provide enough clearance for the physical characteristics of the frame. Other requirements are derived from the torque output, max rpm, and wheel diameter. Equation 2.1.1 below shows the velocity for the vehicle in feet per second derived from physics of a wheel.

$$\text{Velocity} = d_{\text{wheel}} \pi \cdot \frac{\text{rpm}}{60} \quad (\text{Equation 2.1.1})$$

2.1.4 Tracks

Track systems are an ideal solution for vehicles with higher weight and dimension considerations combined with all-terrain applications. The main idea behind using track driven systems is to disperse the weight of the vehicle over a larger surface area, which provides more traction especially in poor driving conditions. Examples of this technology can be seen on almost all tank designs since the early twentieth century and vehicles that need to distribute their weight for terrain applications such as desert, swamp, or snow. The diagram shown in Figure 2.1.1 displays a general design that allows for surface area on the top to be utilized, wheel clearance on the front and back with beveled edges and substantial internal volume to be used for microcontrollers, embedded systems, and batteries.

Figure 2.1 1 – General Desired Chassis Shape and Dimensions (9.75" x 8" x 2.30")



2.1.5 Motors

Many design choices for our vehicle depend on the choice of the motor used to drive the system. The decision of the battery type to be used is conditioned primarily around the motor choice, as that will be the primary draw of power throughout the system. We need to take into account how much our system will weigh, the kind of speeds we wish for it to be able to attain and maintain, and the various types of terrain it will need to be able to successfully navigate. There are a wide variety of motors to choose from, and the most ideal motor is the one that can handle the most weight with the best efficiency versus cost.

2.1.5.1 Brushless Motors

Between brushed and brushless motors, brushless motors are more advanced in terms of complexity and design. The main advantages of using a brushless motor are:

- Greater lifespan
- Improved efficiency (greater torque per weight/watt)
- Decreased interference/noise
- Increased reliability
- Less mechanical dependency
- Programmability

The cost of these advantages is immediately apparent in the price tag associated with these motors. The implementation complexity is increased versus the brushed motor simply due to the electrical nature of the brushless variant, and is also susceptible to damage from too much heat. The heat generated, however, is relatively small due to the lack of friction within the motor, and the electrical to mechanical power conversion is extremely efficient. The reliance on magnets as opposed to brushes increases the lifespan, as the magnets will not erode as a brush will. We can also program the firmware of the brushless motor to limit its maximum speed or to fine-tune its control, which can greatly aide in our wireless control implementation. We also have less electromagnetic interference to be concerned with, as the magnetic design of the brushless motor eliminates most of it.

2.1.5.2 Brushed Motors

The use of a brushed motor is simple in comparison to the brushless motor, and often much cheaper. There are many negatives, however, that must be considered. The dependency on the brushes to carry the current from the commutator to the rotor coils means that over time, as the brushes begin to wear, efficiency decreases until the motor itself simply stops working. The application of our vehicle may require extended use that could ultimately lead to increased maintenance costs over time. The motors would need to be regularly switched out, and though they may be cheaper upfront, the cost of constantly replacing these could eventually exceed the cost of a brushless motor. The brushed motor also has the possibility of generating a spark that could damage our electronics.

2.1.6 Mounted peripherals

The stabilization platform will provide a useful area for many potential applications. The proof of concept of the stabilization platform makes previous difficult tasks possible and can be used for recreational, industrial, or military based products and designs.

One example that particularly underlines the utility in an auto stabilizing platform is used with satellite dishes or other directional dependent devices. If the platform where the satellite or similar device is mounted needs to have a constant direction or stabilization on a particular set of axes, then the platform can perform this function with only some initial calibration settings and a power source required.

2.1.6.1 Targeting

An application that seems obvious for surveillance and security is targeting. With the stabilization platform in mind, it makes the normally difficult and tedious task of keeping a surveillance camera, or even weapon, steady simple and automated.

The application of targeting is a natural fit for the stabilization platform and image processing. The target algorithms used can be chosen by the user, but the addition of have a stable platform to reduce unwanted jitter and movements will almost certainly improve performance algorithm side.

2.2 Power

2.2.1 Battery Technologies

The most important decision for the power systems of our project is the choice of a battery. In robotics, the key battery characteristics to pay attention to are:

- Charge Capacity
- Durability
- Cost
- Environmental/Health Issues
- Maximum Output

Charge capacity is one of the most important characteristics listed above, as robotics applications benefit from having a long lifespan per charge, and we need at least 12 volts of power available for our design. The durability of the battery must be taken into account as well. Being able to reuse the battery a sufficient number of times will save the user money, and increase the overall sustainability of our project. As with all aspects of our project, cost is always an important consideration, although trading money for a higher quality end product is an acceptable option. The environmental and health issues, as well as maximum output are qualities worth looking into as well. Though we do not require a particularly high current output for any aspect of our project, having the ability to output a decent amount of charge at any moment in time is not a bad thing. We also must take into consideration how to dispose of a battery when it no longer functions properly, and the potential for harm in the case of an accident.

2.2.1.1 Nickel Cadmium Battery

NiCD batteries use 1.2 volt secondary cells with an alkaline chemistry. They offer internal resistances that are less than half that of NiMH cells, have high charge and discharge rates, and can tolerate deep discharges. As far as durability is concerned, NiCD batteries typically sport a cycle life of over 500 cycles, and can be charged very quickly. Some shortcomings of the NiCd battery are the possibility, although less prevalent in current technologies, of memory effect, high cost compared to other cells, and general environmental hazard. The memory effect could affect the long-term durability of our project, the high cost would make it more difficult to maintain, and the environmental hazards would make its disposal troublesome. The battery may also be damaged by overcharging. [1]

2.2.1.2 Nickel Metal Hydride Battery

Another battery type that has an alkaline chemistry, NiMH batteries are well suited for high power applications. The cells are, like their NiCD counterpart, often 1.2 Volts. Compared to nickel cadmium batteries, the cycle life is much greater, often consisting of 3000 cycles. A downfall, however, is that these batteries have a larger internal impedance. The energy density is about 1.5 times that of the NiCD battery, and is much more environmentally friendly. These battery cells are susceptible to the memory effect, have a high self-discharge rate, and are less tolerant of overcharging. The 1.2 volt cells also make this a more expensive option than other battery types. [2]

2.2.1.3 Lithium Ion Battery

A strong selling point of the lithium-ion battery, when compared to the previous battery types, is that it does not suffer from memory effect. When not being used, these batteries lose charge at a much lower rate, and sport some of the best energy densities available. On top of the better storage characteristics and life span, these batteries are also environmentally safe, as there is no free lithium metal. They also sport high open circuit voltages, allowing for more power with less current. Lithium-ion cells are about 3.7 volts, and cost more per watt-hour than other options. They also have a much higher internal resistance than both NiMH and NiCad batteries, and if subjected to too much heat may combust. The light weight of these batteries makes it an attractive option for our application.

2.2.1.4 Lithium-Ion Polymer Battery

Lithium-Ion Polymer batteries come with a very low profile, making this an ideal battery for our chassis. The trade-off for the smaller package comes in the form of a smaller energy density compared to Lithium-Ion batteries, which equates to higher cost. They are less susceptible to overcharge, and there is a smaller chance for electrolyte leakage. These batteries have an expected life cycle of more than 500 cycles, and operate between 2.7 and 4.2 volts.

2.2.2 Current Sensing

Current sensing is an important application for robotics applications, as current derives half of the power equation. In order to accurately estimate the duration the vehicle will be able to run, we need to determine the power consumption at any point in time and compare it to our battery's specifications. There are multiple methods for monitoring current in a circuit, but to determine which one to use, we must take into account many factors, namely:

- Footprint
- Efficiency
- Complexity
- Cost
- Output

2.2.2.1 Shunt Resistor

The first method we will look at is that of using a shunt resistor along with an integrated circuit or an operational amplifier to measure current. The shunt resistor is typically a very small resistance, and the operational amplifier is usually configured as a difference amplifier, which measures the voltage drop across the resistor, amplifies that voltage, and passes the value along to an analog-to-digital converter. High side and low side current sensing are the two primary applications that involve the use of a shunt resistor. High-side sensing measures the current before the load it is being applied to, and low-side senses

the current after the load. Each method has its own distinct advantages and disadvantages.

2.2.2.2 Low Side Current Sensing

Low side current sensing measures the current after the load. Low side sensing is generally much easier to implement than its high side counterpart, as it rarely requires more than an operational amplifier. Because of this, it is typically inexpensive to apply to a circuit. It does have its disadvantages, however. The primary drawback to implementing low side current sensing is that it adds extra resistance to the ground path, generally pulling it higher than one would desire. This can cause interference with other aspects of the circuit that share the common ground. It may also add an additional wire to the load that it is in proximity with, which can also be undesirable [29]. This is a solid option if our circuit can tolerate the additional ground resistance.

2.2.2.3 High Side Current Sensing

High-side current sensing is done by measuring the current between the supply and load of interest, as shown in [figure]. A small resistor, usually a factor of a few milliohms, is placed between these two points, and a sensor measures the voltage drop across the resistor. The sensor can be as simple as a difference amplifier, or as complicated as an entire integrated circuit dedicated to the process. Designing the difference amplifier can be a challenge, as choosing the values of the resistors incorrectly can have enormous consequences. Because the current will be measured by a small resistance, the common-mode rejection ratio (CMRR) is of extreme importance. The higher the CMRR, the less error we will have while measuring the signal. According to Texas Instruments, the severity of choosing careful resistance values is a disadvantage with High Side sensing. A 0.01% deviation in any of the resistor values lowers the CMRR to 86dB, and falls by 20dB/decade of deviation [8].

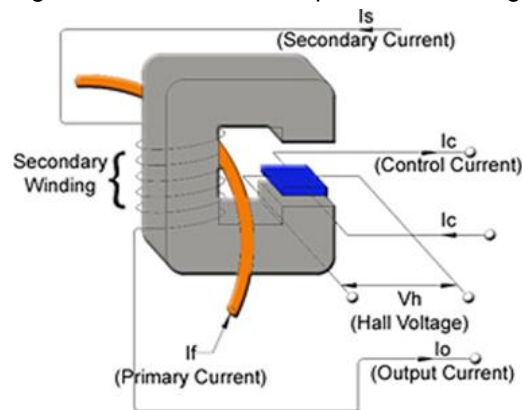
High side sensing's greatest strength is that, if one uses the information from the sensor actively, the designer can open the circuit and prevent any damage from a spike in the current. This method also does not add the extra resistance to the ground path that low side sensing does. Its weakness derives from its sensitivity to variations in resistance in the difference amplifier, but this issue is circumvented with the application of precision integrated circuits.

2.2.2.4 Hall Effect Sensors

Utilizing a Hall Effect sensor to measure the current through a circuit is the least intrusive method available, as Hall sensors measure the magnetic field generated by the current through a wire. Linear Hall sensor integrated circuits work by generating an output signal that is proportional to the applied magnetic field [3]. There are two types of Hall Effect sensors, those being closed-loop and open-loop. An open-loop Hall Effect sensor typically consists of a current

amplifier and an air gapped magnetic core. A potential difference is generated by the magnetic flux, and this potential is amplified to be proportional to the current that it is sensing. Closed-loop Hall Effect sensors add another coil into the equation, and also include feedback circuitry. The added coil is used to produce a current that opposes the primary current being monitored, which then minimizes the magnetic flux produced in the magnetic core. The output current of the closed-loop Hall Effect sensor is a factor of the primary current based on the number of turns the secondary coil possesses. Figure 2.2.2 illustrates a closed-loop Hall Effect sensor.

Figure 2.2.2 – Closed Loop Hall Effect Diagram



Used with permission from Tamura Corporation of America

What makes a Hall sensor an attractive option is that there is minimal power dissipation from the sensor, as there is no resistor required to create a voltage drop to measure. There is a small power requirement to turn the device's internal circuitry on, but this is a minimal cost. The downsides to using a Hall Effect sensor are that they are generally more expensive than other current sensing solutions. Closed-loop Hall sensors are more expensive than their open-loop counterpart, but are more convenient and provide better data. They are also susceptible to static and dynamic electric and magnetic fields, which may require loose cables to be twisted or shielded [4].

2.2.2.5 Current Transformers

Current transformers offer a solution for monitoring high voltage or high current circuits. The option for using 24V to power our electronics makes the use of a current transformer a viable option. A current transformer is made up of two windings, a primary winding and a secondary winding. The primary winding is a low impedance, typically one-turn coil that reflects the current through that part of the circuit. The low impedance allows for minimal power cost when using this sensing option. The primary winding has three components; a loss current, a magnetizing current, and the load current. The loss and magnetizing currents need to be minimized so that the load current actively reflects the primary current through the circuit. The secondary winding is for low currents, and effort is made

during the design of the transformer to keep the two windings independent of each other [5].

The current transformer can be used as an overcurrent protection device by connecting it to a relay. When the current exceeds its maximum specification, the relay will be powered on by the current transformer, activating protective circuitry and maintaining the integrity of delicate electronics. You can also use the transformer to safely step-down high current, generating a smaller current that can be used in conjunction with current or voltage sensor devices. Care must be taken when using these devices. Dangerously high voltages can be developed due to the turn ratio of these devices if the secondary load becomes an open with primary current still flowing. This could lead to a fire, could cause injury to the user, or simply destroy the circuitry it was a part of.

2.2.2.6 Summary

The current transformer and Hall Effect options considered in sections [# and #] are great for their ability to measure current without adversely affecting the power load of the system, but may require additional circuitry to manipulate the output signal into one compatible with our choice of microcontroller, and are generally more expensive. The shunt resistor methods discussed provide a convenient output method with an affordable price, but come at the cost of higher power consumption.

2.2.3 Voltage Sensing

A voltage sensing solution is desirable for our project as it gives us the second half of the power equation. With current and voltage measurements being taken simultaneously, we have the option of running the necessary calculations to display the power consumption to the user. There are a few different methods that can be used to sense the voltage the battery is supplying, such as using a combination of zener diodes and transistors or by using an analog-to-digital converter in conjunction with power regulation methods, but because we require digitized information our options are limited to the latter. Because our options are limited to the use of the analog-to-digital converter, we will look into different methods of regulating the input voltage. Some factors to consider when choosing are:

- Power requirements
- PCB space requirements
- Cost
- Reliability

We can use a zener diode connected to the supply terminal and ground, a voltage regulator in series with the converter, or a voltage divider network to limit the input voltage to the converter. Table 2.2.1 compares these different methods to each other:

Table 2.2.1 – Voltage Sensing Regulation Options

Method	Power Consumption	Cost	Space Requirements	Reliability
Zener Diode	Low	Low	Low	Medium
Voltage Regulator	Medium	High	High	High
Voltage Divider	Medium	Low	Low	Low

The voltage regulator provides the most reliable solution to the problem, but costs the most and requires the most space. These regulators also require a modest sum of power in order to be activated, which makes it a less than ideal solution. The voltage divider is an affordable way, in terms of both size requirements and monetary cost, to limit the voltage to the converter, but is the least reliable method to incorporate. Varying tolerances on the resistors could cause too high of a voltage to reach the converter, and more power than is necessary could be dissipated over these resistors. The zener diode combines the low cost and space requirements of the voltage divider network with a decent amount of reliability, making it a solid choice to regulate the input voltage. The diode will also never be conducting, which makes its power consumption minimal.

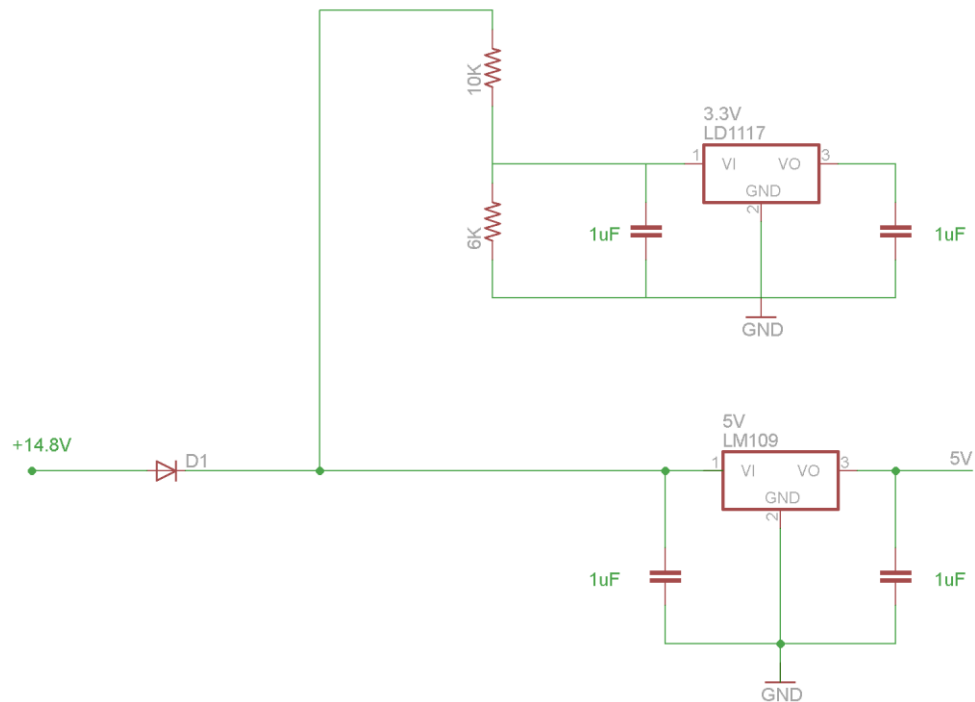
2.2.4 Reverse Polarity Protection

A potentially devastating event for any electronics application, reverse polarity voltages can wipe out all vital circuitry within a robotic device in a matter of seconds. To safeguard against this outcome, we will compare different methods of protection and choose the one most suitable for our application.

2.2.4.1 Diode Protection

The most basic of reverse polarity protection methods is the use of a diode in series with the power supply. Configured as shown in Figure 2.2.3, the diode will prevent the loop from completing when the power source is not properly connected.

Figure 2.2.3 – Diode Protection Schematic

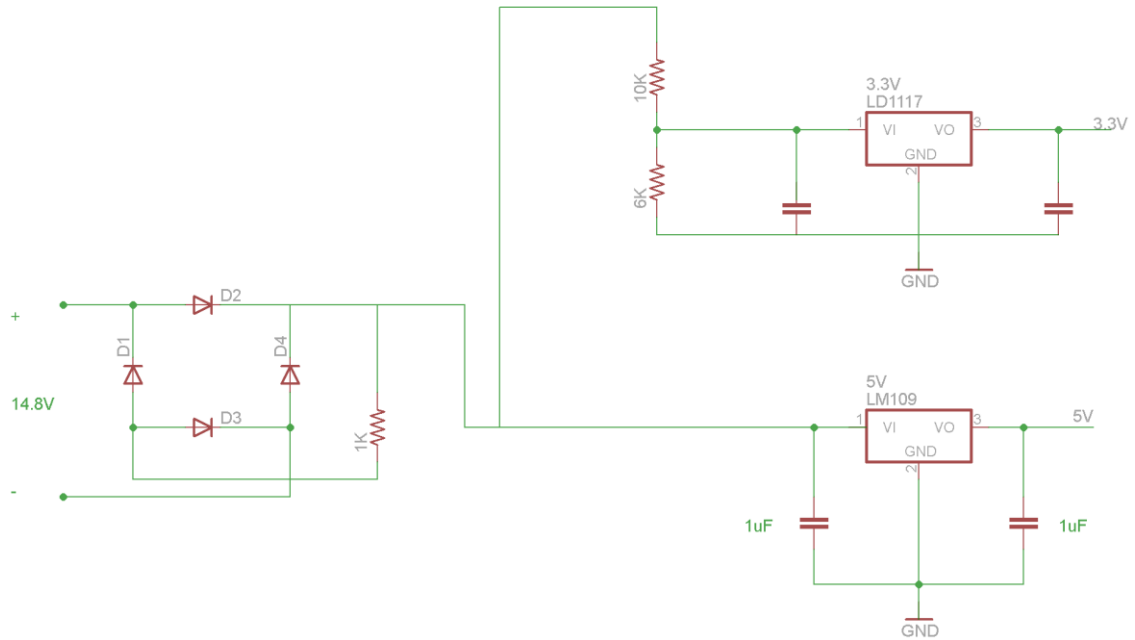


The selection of the diode is of importance, as the reverse polarity voltage limit needs to be able to handle the supply voltage, and a low forward voltage drop would be preferable. The forward voltage is perhaps the greatest drawback to this technique. A forward voltage will lower the input voltage of our power supply by 0.6 to 0.7 volts, resulting in the need for either a larger power supply, which is additional money with more regulation requirements, or components that operate on smaller supply voltages. This is the least desirable implementation.

2.2.4.2 Full-wave Rectification

Full-wave rectification circumvents the need for more advanced protection setups by simply converting reverse polarity signals into the proper polarity. Two well-known methods for full-wave rectification are using 4 diodes in a bridge setup or using a transformer and two diodes. The method we will exam here is the bridge setup, as shown in Figure 2.2.4.

Figure 2.2.4 – Full-Wave Rectification Protection Schematic

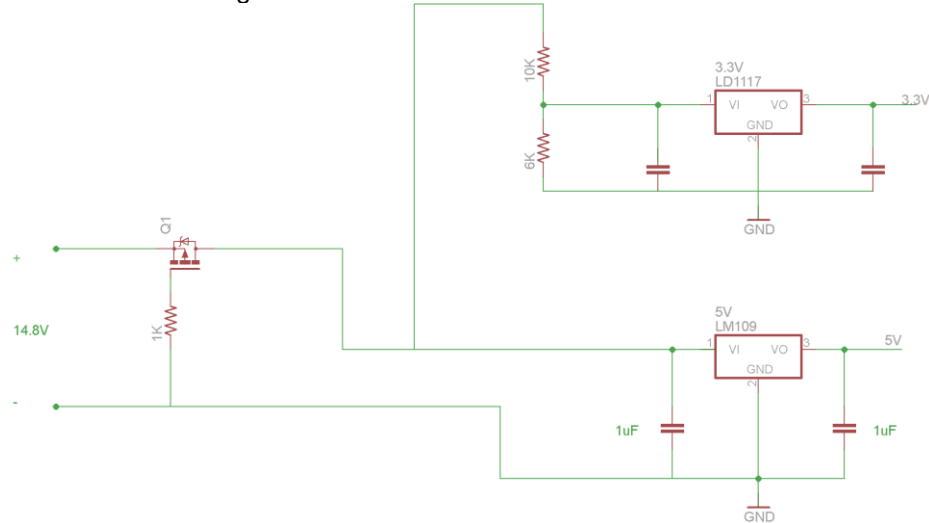


This method incorporates 4 diodes in the configuration as shown below. As discussed in [Diode section number], a major drawback to using a diode is the 0.7 volt drop in the forward direction. In this case, we have not just one, but two active diodes at all times, resulting in the loss of more than a volt for other applications onboard. The loss of so much power simply does not justify the convenience of not worrying about polarity.

2.2.4.3 FET Protection

The use of a field effect transistor as an off/on switch is an extremely viable method of protection. Figure 2.2.5 shows the proper configuration.

Figure 2.2.5 – PMOS Protection Schematic



While the gate is not properly powered, the transistor is simply in an off-state, resulting in an open circuit. When the polarity of the battery or power source is switched to the correct polarity, the gate is properly biased with voltage, resulting in the short opening between the drain and source, creating a closed path with the rest of the circuitry. What makes this method preferable over the diode method, as well as the full-wave rectification method, is the negligible voltage drop for our application. Dealing with a 12-volt power supply, a few millivolts drop is not something to be overly concerned with.

2.2.5 Voltage Regulation

As our power supply will be a 14.8V lithium ion polymer battery, we will need to utilize multiple voltage regulators in order to safely power our integrated circuits and electromechanical components. Based upon our parts, we will need to regulate our voltage down to 3.3V, 5V, and 12V. There are some key factors to pay attention to when considering specific voltage regulators. We need to consider the following:

- Output Voltage Range
- Input Voltage Range
- Efficiency
- Thermal Characteristics
- Current Rating
- Noise
- Package Type
- Footprint

Efficiency is of utmost importance, as we need to get the most use out of our power supply as possible in order to have a decent operating time. Noise is also important because we will be powering digital components with these regulators, and if any of the outputs are too inconsistent it could cause major problems for the rest of our system. The input voltage range must just be specific to the power supply we will be using – 14.8V. The package type is important in terms of being able to easily mount the part onto our circuit board, and footprint size ensures that we do not have to spend extra money fabricating a larger circuit board. Current rating will always be taken into account, as our parts will require a certain current draw in order to properly function. To a lesser extent, the thermal characteristics are worth looking into, but temperature should not be too much of a concern with our application. There are generally two types of regulators: linear and switching. We will discuss the pros and cons of each type specific to our application.

2.2.5.1 Switching Regulators

The three general switching regulator configurations are flyback, step-up, and buck converters. Switching regulators typically utilize a diode, a capacitor, and an inductor. We are most interested in the buck converters, as this configuration acts as a step-down for the supply voltages. The primary advantage to using switching regulators comes in the form of efficiency. Less energy is lost due to the fact that it is not constantly operating as a linear regulator does. The decrease in lost energy allows for less of a build-up of heat, allowing our circuitry to remain cooler and allowing us to use smaller components than a linear regulator. Switching regulators do have the tendency to have more output noise when compared to a linear regulator, which can adversely affect sensitive components [<http://www.maxim-ic.com/app-notes/index.mvp/id/2031>]. According to Maxim, “The parameters affecting the choice of a switching-regulator topology include the peak currents for the load and inductor, the voltage level on the power transistors, and the necessity for magnetic and capacitive energy storage.” [6]. Switching regulators are an ideal regulation solution to battery powered applications because of their efficiency.

2.2.5.2 Linear Regulators

There are three types of standard linear regulators; Standard, Low Dropout, and Quasi Low Dropout. Linear regulators operate by requiring a minimum voltage drop the device to maintain a constant output voltage. This voltage drop is what dictates the efficiency of the device, with the Low Dropout variant having the highest efficiency and the Standard having the lowest. Conversely, the Standard variant has the lowest ground current while the Low Dropout (otherwise known as LDO) has the highest. In the linear regulator spectrum, LDOs are the preferred choice for battery operated applications as they take the most advantage of the available voltage. The problem with an LDO regulator, however, is that the potentials between the battery we are using (14.8V) and some of the voltages we desire (3.3V) are too high, and thus would require a regulator that operates at a higher dropout voltage. This would also greatly decrease the efficiency of our system, and generate a greater quantity of undesired heat. Depending on the selection of the IC we would use, we may also require extra components, such as capacitors and/or resistors in order to stabilize output and decrease oscillation, or to limit the amount of current that can travel into and damage the IC. Table 2.2.2 shows a comparison between linear regulators and switching regulators [7].

Table 2.2.2 – Regulation Type Comparison

Regulation Type	Linear	Switching
Capabilities	Step-down voltage	Step-up, Step-down, or Invert
Efficiency	If $V_{in}-V_{out}$ is small, high, otherwise medium or low	High except at extremely low load currents
Thermal Efficiency	If $V_{in}-V_{out}$ is large, the thermal efficiency is low	High
Footprint	Without heatsink, small to medium	Small except at low power
Noise	Low	Medium to High, due to ripple
Complexity	Low, generally only requiring external capacitors	Medium to High, depending on necessary external components

2.2.5.3 12V Regulation

We must be able to power four 12V motors that are capable of drawing up to 233mA at a rated load of about 1.3 pounds. An option for achieving 12V regulation is to use an adjustable regulator, the LM317, in conjunction with two external resistors. This part has the capability of regulating between 1.25 and 37 volts and is capable of more than 1.5A output current. This is an attractive solution for its ease of use, low cost at \$1.95, and its three-pin package. The only problem is that these regulators are traditionally less stable than packaged IC regulators, and anything that affects the resistors could have potentially catastrophic consequences for the load we are supplying. Another option would be to use the MC34063A, which is designed to be configured to a function that the user defines. It can be used as a step-up, step-down, or inverter regulator, based upon external configurations, and is capable of 3 to 40 volt input, with switch currents up to 1.5A, and operates at a frequency of 100 kHz. The TPS54383 is a dual output, non-synchronous buck converter that can support up to 3A output current and can operate between 4.5 and 28 volts. The output voltage can be anywhere between 0.8V and 90% of the input voltage making it an ideal solution for our 14.8V lithium ion polymer battery. Each of the regulators listed above are capable of driving these motors, but their respective efficiencies and external component requirements, as well as other regulators, will be examined in the design section.

2.2.5.4 5V Regulation

We need 5V regulation in order to power our microcontrollers, audio detection filters and amplifiers, and our servo motors. These components are susceptible to noise, thus making a linear regulator the ideal solution to powering these

devices. The LM109 series of voltage regulators from Texas Instruments are designed for complete 5V regulation are capable of 35V input, and up to 3A output, all in a simple 3-pin package. Another option is to use the LM7805 regulator. The LM7805 is capable of up to 25V input and 1.5A output, and comes in an easy to use 3-pin package. We do not expect high-efficiency from these components, although it is desired, as we are more interested in the protection of these vital circuit components than we are in heat generation and lost power. We will examine these factors further in the design section.

2.2.5.5 3.3V Regulation

Our inertial measurement unit and GPS both require a 3.3V input, which requires the use of a 3.3V regulator. The sensor is susceptible to noise, thus making a linear regulator an obvious solution. The LM2937 is a 3-pin voltage regulator with a 3.3V output, allowable input voltage of up to 26V, and about 500mA of output current. The LD1117 is an adjustable voltage regulator that is available in a fixed 3.3V option. This regulator is capable of up to 15V input, 800mA output, and is capable of an output voltage tolerance of $\pm 1\%$ at 25°C and $\pm 2\%$ at full temperature range.

2.3 Stabilization Platform Research

There are three primary components to the design of the stabilization platform. These are the inertial measurement unit, or IMU, the servo motors, and the platform control. The inertial measurement unit is a collection of gyroscopes and accelerometers configured in such a way as to give the microcontroller the tilt angle of the vehicle, which is processed with the feedback implementation, and corrected through the servo motors. The goal of the stabilization platform is to provide steady camera feed by correcting the roll-axis, which hampers the jarring motion seen by the user through the video feed as the controller moves the vehicle across uneven terrain. We will discuss each of these three components in this section.

2.3.1 Inertial Measurement Unit

The primary concern of the inertial measurement unit is to provide accurate data to the microcontroller about the current roll-angle of the vehicle. A secondary concern is to provide adequate data of the current pitch of the vehicle. The roll stabilization provides steady and non-disorienting video feed to the user of the vehicle, while the pitch feed will allow the possibility of toggling the camera to be parallel with a non-inclined surface, allowing the use a straightforward view of their current location.

The Gyroscopes provide the rotation about an axis in degrees per second, which can be passed to a microcontroller via built-in analog-to-digital converters. Taking this raw data, we can run a numerical function to translate it into usable data that can counteract the effect of the vehicle's tilt on the camera. As discussed

previously, the two axes we wish to monitor are the pitch and roll axes. In order to be able to accurately measure movement about these axes, we will need a 2-axes gyroscope. 3-axes gyroscopes are regularly available, but the additional cost is undesirable considering we are not concerned about yaw. Positioning of the gyroscope with respect to the camera is of utmost importance. Any error in its mounting on the board will render it useless.

While the degrees per second of rotation provided by the gyroscope are useful, we can provide more simplistic and accurate information by the inclusion of an accelerometer. The inclusion of the accelerometer completes the creation of the inertial measurement unit. The accelerometer measures acceleration, and using this data in conjunction with the data passed from the gyroscopes, we can run an algorithm in our microcontroller that will allow us to determine the actual angle of the vehicle relative to the ground plane. Though the computation of this angle will take extra time, it will allow us to more accurately adjust the angle of the camera, and the inclusion of both gyroscopes and an accelerometer allows us to smooth error in the data between either devices. Table 2.3.1 provides specific parts and their qualities that would be applicable to this application:

Table 2.3-1.3.1 – IMU Comparison

Part Number	Axes	Type	Price	Convenience	Size
ITG-3200	3	Gyroscope	\$24.95	Low	Small
ITG-3200 BO	3	Gyroscope	\$49.95	Medium	Large
MPU-6050	3	Accelerometer / Gyroscope	\$39.95	High	Large
9DOF Razor	3	Accelerometer / Gyroscope / Magnetometer	\$124.95	Highest	Large
IDG1215	2	Gyroscope	\$24.50	Lowest	Small

The ITG-3200 / ITG-3200 BO are both the same triple-axis gyroscope one version (BO) coming with a breakout board to allow for more convenient prototyping and configuration. Both of these boards offer 16-bit analog-to-digital converters to digitize the outputs, an optional internal low-pass filter bandwidth, and a 400 kHz I2C interface. The ITG-3200 also only requires 6.5mA current consumption for operation, allowing for longer battery life. The MPU-6050 goes one step further than the ITG-3200 and provides a triple-axis accelerometer that can be programmed with a full scale range of ± 2 , 4, 8, or 16g. This unit comes as a breakout-board and for \$10 less than the ITG-3200. It also allows for I2C output data in rotation matrix, quaternion, Euler angle, or raw data format. The Razor IMU offers the most comprehensive solution yet, including a triple-axis gyroscope, magnetometer, and accelerometer. It comes programmed with an 8MHz Arduino bootloader and example firmware that demonstrates the outputs of the sensors. It even comes with a connector that automatically regulates any

power applied to it down to 3.3VDC. This board comes priced at \$124.95, but with built in USB compatibility and Bluetooth or XBee support, it is well worth the price. The simplest of the 5 options above is the IDG1215 dual-axis Gyroscope. It provides the rate of rotation about the X- and Y-axis, incorporates a low-pass filter for the sensor, and comes trimmed so that user calibration is unnecessary.

The selection criteria are based upon cost, convenience, and size. The size of the breakout boards are considered large, while the IDG1215 and ITG-3200 as individual units are considered small, as they will be mounted onto our PCB. The 9DOF Razor is the most convenient, offering three different measurement devices and a bootloader specific to an Arduino microcontroller. The MPU-6050 follows, offering all the components we would need to calculate the angle of the vehicle while maintaining a decent cost. The ITG-3200 breakout follows, with the ITG-3200 being the next least convenient and the IDG1215 as the least. The ITG-3200 and the IDG1215 are both capable of being mounted onto our board directly, but this will add complexity to our layout, and cost us more to fabricate. The advantage of these two individual units is that they cost less upfront. The MPU-6050 is the most logical choice, as it provides the accelerometer and gyroscope combination that we require, costs less than the ITG-3200 breakout board that only includes the three-axis gyroscope, and has programmable gyroscope and accelerometer ranges. The only downside is that it does not offer the analog-to-digital converters that the ITG-3200 does.

2.3.2 Servo Motors

The servo-motors are what will be used to correct the angle of the camera, based on the input from the IMU. The application of the servos is greatly simplified when using an Arduino microcontroller. Arduino code has built in libraries that allow the angle of the servos connected to it to be automatically converted into data usable by the microcontroller. The servos will then be controlled by the Arduino passing varying pulse-width modulated signals to the motors, which will cause them to rotate an amount specified by the width of the signals. The determination of the pulse-width signals is done by the feedback loop, which takes the information provided by the IMU and runs it through a PID controller. The selection of the servo motors depends on their cost versus the amount of torque they provide, the range of movement, weight and size, and the type of servo motors they are (continuous or non-continuous). Table 2.3.2 provides numerous servo-motors and their qualities:

Table 2.3.2 – Servo Motor Comparison

Part Number	Weight	Torque	Speed	Voltage	Price
Hitec HS-422	45g	3.3kg/cm	60°/0.21sec @ 4.8V 60°/0.16sec @ 6V	4.8V / 6V	\$12.10
Hitec HS-82MG	19g	2.8kg/cm @ 4.8V 3.4kg/cm @ 6V	60°/0.12sec @ 4.8V 60°/0.10sec @ 6V	4.8V / 6V	\$17.94
TGY-1501MG	60g	15.5-17.0kg/cm	60°/0.16sec @ 4.8V 60°/0.14sec @ 6V	4.8V / 6V	\$11.95
HK-933MG	12g	1.8kg/cm @ 4.8V 2.0kg/cm @ 6V	60°/0.12sec @ 4.8V 60°/0.10sec @ 6V	4.5V / 6V	\$11.20
Corona Digital Servo	12.5g	2.2kg/cm @ 4.8V	60°/0.11sec @ 4.8V	4.8V	\$6.41

The selection criteria for the proper servo motor are based upon its weight, torque, speed, price, and to a smaller degree, voltage. For convenience, the less the servo weighs the better, as this reduces the load on our vehicle. The TGY-1501MG is by far the heaviest of the servo motors referenced in [table], but has 17kg/cm of torque, making it the most powerful of them as well. Another drawback to the TGY-1501MG is that it is the second slowest servo listed, and for our particular application speed is a priority. The weakest of the motors is still capable of almost 4lbs of torque, which implies that any of the servos in the table will be capable of stabilizing the small camera we will be using. The Hitec HS-82MG provides a solid balance between torque, weight, and speed, but costs almost \$6 more than the second most expensive servo, while the HK-933MG weighs the least, has great speed, and costs little, but is only capable of 2kg/cm of torque at 6V, making it the weakest of the servos listed. The Corona Digital Servo suffers the same problems of the HK-933MG, but at almost half the cost, and the Hitec HS-422 is the second heaviest servo while also being the slowest. It should be noted, however, that the Hitec HS-422 offers the best resolution out of the aforementioned servo motors, and accuracy is more desirable than speed in our particular project. Relative to the rest of the system, 60g is a negligible weight, and 60°/120mS is more than fast enough to smoothly stabilize the camera.

2.3.3 Stabilization Platform Control

The Arduino platform provides many convenient methods for controlling servos and gathering information from external sources. There are two primary libraries available that we are interested in: Servo and Wire. Table 2.3.3 provides a list of relevant functions from each library and their applications.

Table 2.3.3 – Arduino Functions

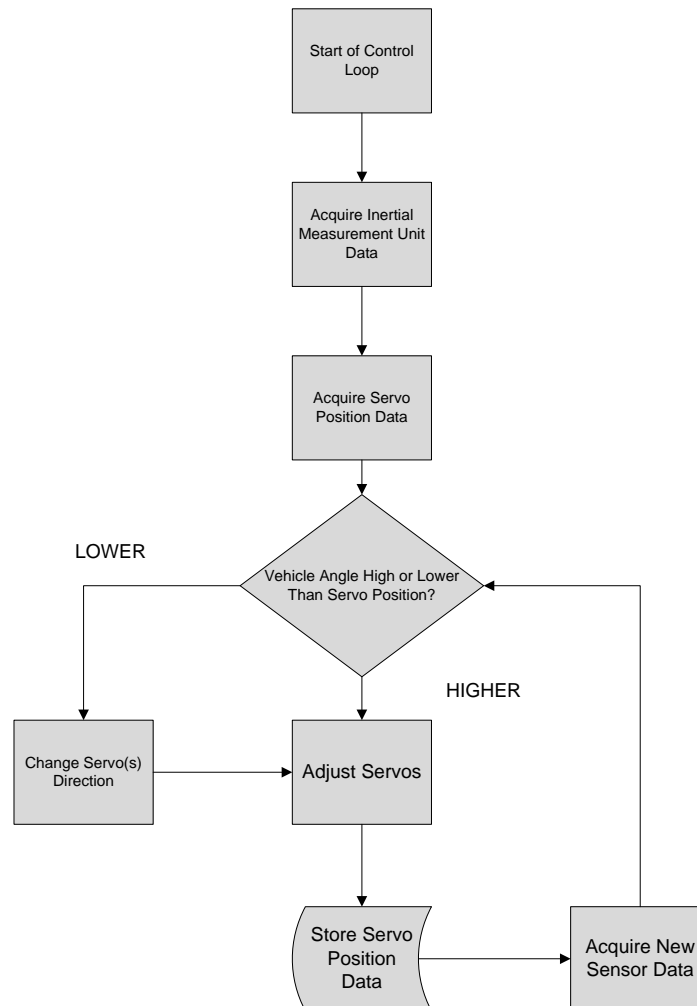
Library	Function	Application
Servo	Attach()	Connects servo variable to specified pin
Servo	Read()	Reads current angle of specified servo
Servo	writeMicroseconds()	Controls the servo angle
Wire	beginTransmission(address)	Begins communication with external device
Wire	requestFrom()	Requests bytes from external device
Wire	read()	Reads byte from external device after request command
Wire	write()	Writes data from external device
Wire	onReceive()	Registers a function to be called when external device receives a request

The functions attach, read, and writeMicroseconds are of utmost importance to our design. This allows us to easily communicate back and forth with the servos based on the information we are receiving from the IMU. Calculation for the writeMicroseconds parameter will be discussed in [section]. The Wire library gives us the necessary functions to communicate with external I2C devices. We can read and write the information we receive from the IMU with the read and write functions, and onReceive will allow us to take the appropriate actions when we receive data from the IMU using the requestFrom() command. BeginTransmission is a simple way to begin communicating with our IMU, and will be done upon power-up of the system.

While Arduino provides convenient commands for our use with servos, we still must develop a solution for creating a PID control loop within the microcontroller. Luckily, Arduino is an open-source platform, and a library exists to aid us with our PID implementation. Arduino PID Library contains a set of seven functions. PID()

creates a PID controller linked to a specified input, output, and setpoint. It also takes into account the tuning parameters associated with PID control theory, and a direction to determine the way output changes based on error. Compute() contains the PID algorithm itself, and is recommended to run at least one time per loop. SetMode() sets the PID controller to an on or off state. SetOutputLimits() limits the output to a specific range, which is defaulted to 0-255: the Arduino PWM range. This will be of great benefit when controlling our servos. SetTunings() dictates the dynamic behavior in general of the PID controller, and is primarily useful for situations in which the tuning needs to be changed during operation. SetSampleTime() controls how often the PID algorithm evaluates. It is defaulted to 200mS. SetControllerDirection() determines the direction of the output. Because our vehicle may be inclined at either a negative or positive angle, SetControllerDirection() is important because it allows us to accurately correct our platform. Though we could construct our own form of PID control with the tools available to us in Arduino, using the library allows us to maintain legibility in our software and makes debugging easier. Figure 2.3.1 shows the basic control loop flow:

Figure 2.3.1 – Control Loop Flow Chart



The beginning of the control loop is when we begin transmission with the inertial measurement unit, and we start off by acquiring both the current angle of our platform and the current position of the servos. The servo position is measured from 0-255, with 0 generally being 0° and 255 being about 180°. From there we have to determine whether the stabilization would have to decrease or increase the angle of the servo. If the setpoint angle is lower, we have to readjust the output direction using `SetControllerDirection()`. We then adjust the servos accordingly, store the new servo position, reacquire servo data, and repeat from the angle checking state.

2.3.4 Microcontroller

For the stabilization platform, we require at least two output pins capable of PWM for controlling the servos, and three input pins to receive angle position data from the servos and the angle position data from the inertial measurement unit. Since we are planning on using the Arduino platform, we will compare various Atmel microcontrollers in Table 2.3.4 and determine the chip best suited for our application.

Table 2.3.4 – Microcontroller Comparison

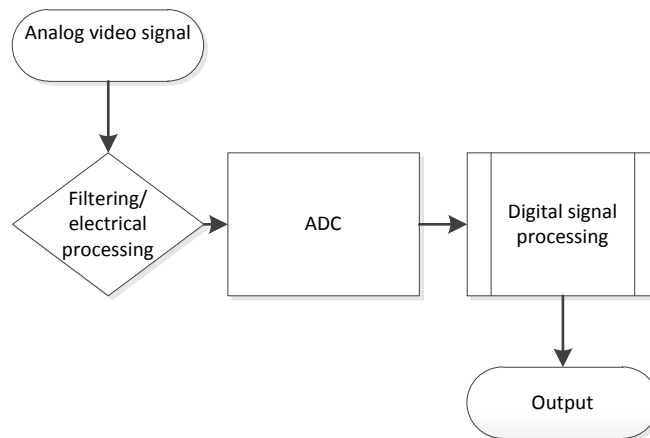
Model	Input Voltage	I/O Pins	Frequency	CPU	Flash
ATmega2560	1.8-5.5V	86	16MHz (Max)	8-bits	256KB
ATmega16U2	2.7-5.5V	22	16MHZ (Max)	8-bits	16KB
ATmega328	1.8-5.5V	23	20MHZ (Max)	8-bits	32KB

The ATmega328 offers 6 PWM channels in a 32-pin package, the ATmega16U2 offers 4 PWM channels in a 32-pin package, and the ATmega2560 offers 15 PWM channels in a 100-pin package. A 100-pin package is too extreme for our application, as having 86 I/O pins would mean a vast majority would go unused, and 256KB of flash memory is more than likely overkill for our control loop software implementation. The ATmega16U2 has too few PWM channels to be applicable, while the ATmega328 has 6, and double the amount of available flash memory. While all three of these operations have maximum frequencies of 16MHz or greater, the ATmega16U2 and ATmega328 are capable of 1 MIPS throughput, while the ATmega2560 is capable of a throughput of 16 MIPS. Weighing these options, the ATmega328 appears to be the best solution for our stabilization platform controller.

2.4 Image processing

Image processing will provide the security system with a means to see the world. The possibility of having a video stream, or multiple streams, available for post processing and viewing is a vital component in securing areas and resources through surveillance. Analog image processing has existed much longer than its digital counterpart and still provides a quick and effective way in transforming the electrical signals from an analog source signal. Digital image processing has many advantages over analog, such as compression, repeatability, and complex processing algorithms. With digital image processing, entire fields of computer vision and intelligent machines exist which can use video information to perform complicated tasks and learn. The signal flow diagram in Figure 2.4.1 below outlines Analog to Digital conversion. [39]

Figure 2.4.1 – Analog To Digital Video Conversion



2.4.1 Analog image processing

Analog image processing has the benefits of being able to use solely embedded systems, which provide very fast dedicated processors, to handle and manipulate the signal information real time. The signal can be filtered, modified, or even reduced, which can result in a more suitable signal for further processing digitally or may even be enough by itself through hardware such that the desired goal, for example motion detection, is possible. The analog signal components would have to be extracted from the composite signal, processed, and then be sent to the user or converted to a digital equivalent for further processing.

Using analog image processing provides an easy way to interface with a lot of the existing hardware and can also be used to process and reduce the data much faster than its digital equivalent. The actual number of operations and complexity of the processing that can be done is extremely limited compared to digital image processing, however, when available, the processing done on the analog side can save valuable time and resources for the system. [35]

2.4.2 Digital image processing

Image processing will provide the user with enhanced feedback and capabilities such as target tracking, motion detection, and pursuit mode. Image processing is required for the reason of surveillance and security. A picture element, or pixel, is a discrete element of the 2D array given as feedback by the camera. The amount of pixels available through the camera will be an important aspect of image processing as a decision needs to be made by the computer for each pixel. For a resolution of only 320x240, more than 75,000 pixels are available to be processed. A typical web camera has a resolution of at least 640x480, which will provide more than enough information for motion detection, blob detection, and object tracking. The more pixels provide more information to be processed, and more information usually means better decisions can be made. This becomes more computationally intensive as millions of elements need to be processed, and therefore a decision must be made based on the balance between performance and resolution. [39]

2.4.3 Video input

The video input stream or streams will be the user's remote vision connection and the data source for image processing. The camera in use will provide this stream at the desired frame rate and resolution, which will be set by the user. A higher resolution or frame rate will require higher bandwidth due to the increased data rate, which could drastically reduce performance. The real time application required in this processing and viewing stream will be a top priority, which will require a dedicated tuning process and plan to determine which combination of frame rate, quality, and resolution will provide the user with optimal results.

The most popular video streams that are directly accessible from webcams are the MJPEG and RTSP streams. The MJPEG stream is a buffered JPEG stream that can be accessed via the HTTP protocol. This allows for a high level of cross platform compatibility and many existing algorithms and software suites that already have HTTP protocol libraries in them have the ability to incorporate these streams into the code with very little code editing and manipulation.

For testing and tuning for the tracking algorithms, prerecorded videos can be used to provide a repeatable test for comparisons and performance improvements. This will also allow the user to test various applications and situations simultaneously through post processing.

2.4.4 Video Compression

The compression of video in the digital world has deemed to be a requirement and also provides an advantage compared to its analog counterpart. The ability to compress video reduces its size before sending and provides the user on the receiving end to optimize data flow and prioritize quality or bandwidth allocation.

The camera's ability to compress video using the best algorithm will be an important factor in deciding the part to be used. Each camera has the ability to compress the video data at various quality levels. The quality level determines how much compression will be applied to each frame and subsequently the data rate required by the stream.

2.4.5 Compression Artifacts

A result of the video compression is the loss of data information, especially in high frequency areas of the picture. This is where the detail or large changes in pixel information occur.

The ability to image process depends directly on image quality and subsequently the compression. If the data were sent as a raw image there would be far too much information to process in a reasonable amount of time and it would only be valid for post processing. To be able to have a real-time system process the video information without having a significant delay in the time between image receiving and completion of processing, the total data size must be reduced to manageable size. This image size is not a constant, as it can depend on many factors such as processing power, link bandwidth, and the algorithm that is being used. Using different levels of image compression, with all other parameters constant, it is possible to achieve a reduced image size without sacrificing too much valuable image information. If this compression level is taken too far, it will affect the capabilities of the image processing algorithms through deformation and compression artifacts. Figure 2.4.2 exhibits how compression can distort the image.

Figure 2.4.2 – Compression artifacts can be seen on the right image



2.4.6 Camera selection

A camera with video capability at minimum resolution of 160x120 was required for the image processing to function as an integrated part of the user interface, autonomous navigation, and tracking. Audio features will be a plus, but would not be required since there will be a dedicated platform for the audio system which

sufficed the system needs. The camera mounted on the stabilization platform needed to be light and small enough for the stabilization platform to support and maneuver it and be able to provide a video quality in day and night use that is adequate enough for image processing. Table 2.4.1 outlines different possible cameras.

Table 2.4.1 – Decision Matrix for Camera Selection (Note: Wireless option is not required, but preferred)

Model	Resolution	FPS	Range(ft)	Wireless	FOV	Price
Microsoft VX-3000	640x480	30	-8	No	55°	\$20.94
Cisco-Linksys Wireless-N	640x480	30	100	Yes	61°	\$100.00
Smarthome 75790	640x480	30	30	Yes	67°	\$91.94
D-Link DCS-930L	640x480	30	100	Yes	45°	\$59.99

The resolution of the cameras has an important implication of providing the information through the pixel count. The image processing techniques operate on the pixels, which are a discrete measurement of light recorded and processed into the RGB value, which is what the image processing side will see. As the resolution doubles, the total number of pixels will quadruple. This effect means the computer will receive four times the amount of information, which in most cases is an advantage as it can be often seen that the more information one has for processing, the better and more accurate of a decision can be made. The downside of this is that if there is too much data with this increase in resolution it will need to be reduced and it could also just be superfluous data. In many applications where image processing is successfully implemented, it can be seen that the kinds of images, backgrounds, and foregrounds are modeled and can be almost predicted by the computer. An example of this is the green screen application commonly seen on television performances and known in media. This uses a simple processing technique to replace the unique green color with the desired image where it is fitted and adjusted appropriately.

With the image processing in mind, a minimum of 320x240 resolution was anticipated to be able to have enough image information and resolution to yield desirable results with image processing technique. Since all the cameras have at least this resolution the Cisco-Linksys Wireless-N camera provided the largest wireless range and performance per dollar spent.

2.5 Software

2.5.1 Image Processing Software

2.5.1.1 C++/C

Another option that was investigated was using C++ and C standard libraries to handle and process incoming images and video streams. This is done widely in proprietary industries and is also the backend of similar open source solutions such as OpenCV.

The main advantage of writing the image processing algorithms from scratch is the ability to have complete control and a working knowledge of all the backend information. When each algorithm, function, and class is written by the user who is directly implementing them, it can be optimized for the specific application at hand and any overhead or redundancies can be reduced or even eliminated.

2.5.1.2 OpenCV

OpenCV is an open source computer vision (CV) set of libraries and APIs that allows users to quickly start using common image processing algorithms and functions such as blob detection, Kalman filtering, and motion detection. Since it is open source, everything is available for use without restriction and the software is free, which is a major benefit for student funded projects.

The OpenCV libraries are always expanding and changing due to it being open source and available freely for everyone to use, edit, and improve upon. At the time of writing, the current stable release is 2.4. This release contains mainly useful examples and introductory documentation which will be used later to demonstrate some of the algorithms that the S.H.A.S.bot will be implemented in image processing. [33]

2.5.1.3 Data organization

The data from the video camera will be broadcasted as a MJPEG stream or an RTPS and processed directly through OpenCV's provided libraries. Most of the actual byte by byte and class organization is done already in C++.

When an image or video stream is passed to the software side, it is important for the software to keep its memory usage to a minimum. When the functions are called, the large size of the information within the image object must be passed by reference in order to keep the propagation of duplicates to a minimum within the program.

Once the image data is in the program, it may need to be copied over for parallel processing, depending on the algorithm in use. The organization of data in this

case is important, since the processing time is now dependent on the memory available and the advantages that the parallel processing will provide.

2.5.1.4 Algorithms

The algorithms used in image processing can often be very mathematically intensive and difficult to explain in the general scope. What they often boil down to is the ability to detect unique features in an image or in a set of images, which could be a video, and give the user information on the location and movement of the pixels and how they change. For brevity, the algorithms won't be discussed in too much mathematical detail, however, their overall function, inputs, outputs, and procedure will be explained.

2.5.1.4.1 Blob detection

Blob detection provides an alternative approach to grouping, motion detection, line detection, and similar algorithms. The library in OpenCV has a number of preexisting settings and modes for tracking and blob detection based on

The background subtraction technique has been an image processing technique used even in the ages of analog image processing. The basic idea behind background subtraction is to get a mean value for each pixel and compare it to the new pixel. If this calculated difference is above a certain threshold, then the pixel has moved or some event has occurred to cause this change. It could be due to movement, or it could be a change in lighting.

The advantages of the blob detection technique are that it allows the organization and computation contours and lines temporally. This builds up a level for certain object locations and they can be traced over time and tracked. The information given by this can also be used to limit our data input for performance optimization and track information.

There are various blob detection libraries that can be used, each with their own advantages and performance increases or decreases. During the testing phase each library will be analyzed and a final library or libraries will be decided upon. [34]

2.5.1.4.2 Motion detection

Motion detection provides useful information that can be interpreted as movement within the image. This movement can be translation, rotational, or general plane motion. The scope of this application is to use the motion detection combined with other image processing algorithms to be part of the inputs for the tracking software as well as be displayed on the GUI.

The software can assist in accomplishing this goal with background subtraction using the mean value calculated from the images. There are some libraries within OpenCV which can provide a baseline for these algorithms and techniques. The majority of the time spent programming these algorithms will be spent tuning and

calibrating the settings for different environmental changes such as weather, lighting, terrain, and desired objects to be tracked. [33]

2.5.1.4.3 HSV conversion (Hue-Saturation-Value)

When dealing with computer vision and image processing, using the traditional RGB interpretation seems intuitive for humans being that we differentiate between colors regularly in this manner. For computers, on the other hand, are better at handling pixel information when it is represented as intensities and lighting differences.

The HSV conversion technique allows for this transition and provides the image information in a more useful way than a RGB representation. [36]

2.5.1.4.4 Filtering

Filtering in its most general form is the method of producing a new output(s) with a set of given input(s). In the application of image processing and target acquisition, the inputs of interest are limited to data from our sensors.

The Kalman filter is a well-known and widely used filter that has been researched and developed since the 1960s. The basic idea behind this filtering process is to use our native coordinate and constant velocity dynamic systems to provide us with an estimate of target position, from here on known as its state. The optimal filter can be found directly by analyzing its impulse response. It is not as simple of a task as one might think to synthesize the filter from such data. [30]

The Kalman filter is a linearly recursive filter. This has a couple important advantages that can be used on systems with limited memory and computational power. Since the equation is recursive and only requires one set of previous measurements and estimates, the amount of memory required for this is constrained only to the actual size of the single previous measurement and estimate.

There are many variants and derived filters from the Kalman filter, which can be applied to a wide variety of image processing, models, control loops, and countless other applications and fields. The advantages with the Kalman filter and other Bayesian related filters and equations are to reduce the noisy measurements and output a filtered state of the target that is combination of estimates and measurements.

The equation 2.5.1 on the next page is the basic Kalman position equation. The estimated state x_{EST} is a the filtered or “smoothed” position estimate. It can be computed from the previous position $x_{n,n-1}$ with the addition the measurement minus the previous position’s result weighted by k_n , which is the Kalman gain factor. The final factor ε_x is the inherent state error associated with the variance of our measurements.

$$x_{\text{EST}} = x_{n,n-1} + k_n(y_n + x_{n,n-1}) + \varepsilon_x \quad (\text{Equation 2.5.1})$$

The equations 2.5.2 and 2.5.3 use of kinematic equations derived from the basic Newtonian physics allows us to arrive at the follow implementation of our position state estimator.

$$x_{\text{EST}} = x_t = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x_{t-1} + \begin{bmatrix} \frac{T^2}{2} \\ T \end{bmatrix} a + \varepsilon_x \quad (\text{Equation 2.5.2})$$

$$G_t = [1 \quad 0] \bar{x}_t + \varepsilon_G \quad (\text{Equation 2.5.3})$$

The implementation of the Kalman filter is done with the error in state and measurement prediction in covariance representation for the position and measurement, as shown below in equations 2.5.4 and 2.5.5, respectively:

$$\varepsilon_x = E_x = \begin{bmatrix} \sigma_x^2 & \sigma_x \sigma_v \\ \sigma_x \sigma_v & \sigma_v^2 \end{bmatrix} \quad (\text{Equation 2.5.4})$$

$$\varepsilon_G = E_G = [\sigma^2] \quad (\text{Equation 2.5.5})$$

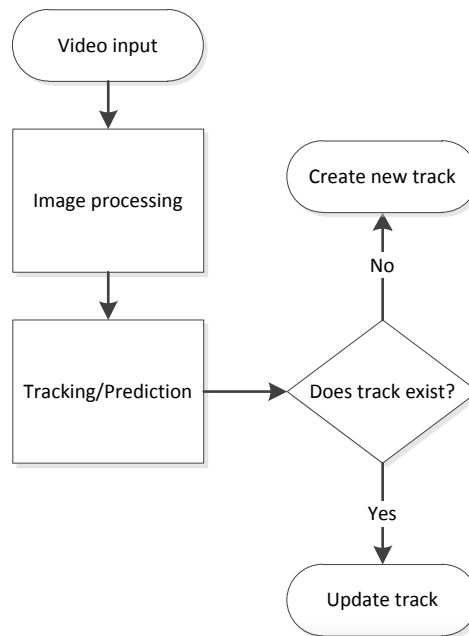
In the application for image processing, it is possible to use a simple position prediction based on pixel displacement after blob or object detection and a derived velocity estimate. The velocity estimate is more difficult to rely on in terms of accuracy as there will be no ranging equipment on board the S.H.A.S.bot to provide a measurement.

The use of these algorithms allows a basic smoothing effect from measurement to measurement, which can reduce jitter and even enable the software to predict a future position state. This is applicable to targeting systems and systems where tracking accuracy is needed. [31]

2.5.1.4.5 Tracking

The requirement of tracking in this image processing system will allow the robot to predict targets motion, velocity, and position in real time and give the user, who may be disposed with other tasks, a better idea of where the target's final position may be. The tracking portion of the software will be a fairly basic set up of creating new tracks and updating previous tracks when the gating or centroids of the measurement are within a threshold. The following signal flow diagram in Figure 2.5.1 outlines the operation of the image tracking software. [30]

Figure 2.5.1 – Tracking Software Overview



The centroid calculation is based on calculating the centered weight of the object in question. Weight could be the area of the object or some other similar parameter. Ultimately these definitions are defined by the user so the centroid could also be specialized and use only the calculation within a certain color range to refine and tune the calculation.

2.5.2 Control Software

Software selection is essential in both the design and function of the remotely-controlled surveillance vehicle. It includes language-specific integrated development environments (IDEs), which provide a helpful interface for writing the code that gives functionality to the microcontrollers, wireless modules, and other hardware. It also includes the native firmware used by these onboard devices. Potential software to be used in the development and control of the communication system are as follows:

Development Environment Software:

- Processing IDE for development of vehicle control Sketches in C
- Arduino IDE for development of vehicle control Sketches in C

Configuring and Testing Software:

- X-CTU for configuring and updating RF communication modules
- Arduino software for configuring and updating microcontroller

2.5.2.1 Processing

Processing is an open source programming language and environment that can be used to create graphics and software sketches. It uses a language similar to C. It has a vast API with basic control structures and functions, but also many functions that are highly dedicated to creating graphics and animations. This makes it an ideal platform for creating wireless control software for the vehicle as well as a simple GUI to go with it. Processing's simple serial I/O libraries also make it easy to communicate from PC-side to embedded-side using the XBee RF modules. [15]

2.5.2.2 X-CTU

X-CTU is an RF Module configuration and testing tool developed by MaxStream, which is used to program the XBee RF module. It features an interface to download and update firmware on the XBee module, an integrated terminal window for editing device parameters when the XBee is in Command mode, and a simple Range Test with an interactive RSSI display.

- **PC Settings:** The PC Settings tab allows the user to select a valid COM port to begin configuring a connected device.
- **Terminal:** The terminal window can be used to edit device parameters when it is in Command mode. It can also compose and transmit test packets between devices in either ASCII or Hexadecimal.
- **Modem Configuration:** The Modem Configuration interface allows the user to download all firmware updates for MaxStream devices and write them to a connected device. Devices can be configured to act as coordinators or routers and their specific addressing parameters can also be changed.
- **Range Test:** The Range Test tab allows the user to dynamically test the connectivity of a previously established network. The RSSI checkbox must be checked to show Signal Strength. [13]

2.5.2.3 Arduino

The Arduino software is a simple, open-source development environment based off of Processing, avr-gcc, and other open-source software. It is used in the design, configuration, and testing of an infinite range of electronics projects. It's interface and functionality is very similar to that of Processing.

2.5.3 Graphical User Interface Software

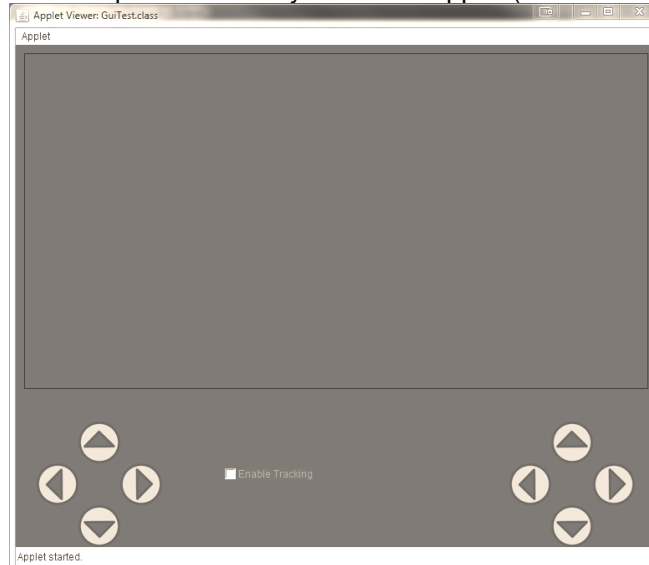
A virtual controller interface will be used via laptop or other portable computer for the purpose of executing orders to the robot as well as monitoring the display, audio, and other sensor data. There are several options concerning which

software and language that the Graphical User Interface (GUI) will be developed in. They are listed as follows:

- Java-based GUI: Would be developed using the Java .applet and .awt software libraries for the display itself along with the .io and .net libraries for incorporating data streams (control, display, audio, power sensing, etc.). The GUI software would be developed in Eclipse on an OSX, Linux, or Windows machine.
- Processing-based GUI: Would be developed using the controlP5 for the majority of the graphics and user controls and other native Processing libraries for everything else. The GUI software would be developed in the Processing development interface on an OSX, Linux, or Windows machine.
- C++-based GUI: Would be developed using the windows.h library for the majority of the display features and other native libraries for everything else. The GUI software would be developed in the Dev-C++ development interface on an OSX, Linux, or Windows machine.

The software will be working with large amounts of input and output data so a relatively high performance computer (dual core + 4 GB RAM minimum) will be required. Data will be incorporated from multiple sources. These include but are not limited to the IP Web Camera data via 3/4G or a local/other network that has internet connectivity, the camera positioning data, the voltage/current data, and the audio data all via a wireless RF communications module. All of this data will be compiled into a user-friendly interface with a centered picture with control options and status data below the picture. Figure 2.5.1 shows a sample layout of the desired user interface.

Figure 2.5.1 – Sample Preview Layout of GUI applet (written in Eclipse IDE)



2.5.3.1 Java

The following research outlines information necessary to create a functional graphical user interface in Java.

2.5.3.1.1 Java Libraries

The Graphical User Interface (GUI) will utilize several various Java software libraries. These include `java.applet`, `java.swing`, `java.awt`, `java.io`, and `java.net`. In addition to these libraries, the two codecs `JPEGImageDecoder` and `JPEGCodec` will be imported in order to properly display the camera images.

The `.applet` library will be used as a panel or “container” for the all of the functionality of the GUI. It will also serve to let the control interface be available online, in-web browser. [1] The `.swing`, along with the `.awt` (Abstract Windowing Toolkit) library will make up the central components of the GUI. It is simply a toolkit that allows the creation of buttons, check-boxes, and other useful components for the GUI.

The Java `.io` library provides for system input and output through data streams, serialization, and the file system. In order to take in and display data from the IP webcam, the ‘`BufferedInputStream`’ class in the `.io` library will be used. Using the `URLConnection` and `URL` classes in the `java.net` library, an input stream can be acquired from a specified IP address set up by the IP webcam. Once this picture data stream is set up, it can be properly displayed on the GUI. [14]

2.5.3.1.2 JPEG Buffer/MJPEG Stream

In order to successfully get a data stream from the IP address specified by the IP webcam, the `java.net.URL` and `java.net. URLConnection` classes will be used. A `URL` (Uniform Resource Locator) object represents a pointer to a resource on the World Wide Web. [4] The resource that will need to be accessed for the GUI is a stream of JPEG images. A `URLConnection` object is simply a `URLConnection` with support HTTP-specific features. A `URL` can be parsed and returned as a `URLConnection` for the purpose of opening a connection for that `URL`.

A `DataInputStream/InputStream` is used to let a program read primitive Java data types from a basic input stream in way that is universal to any computer. Once a class is initialized, it can read bytes of data sequentially. For the GUI, the bytes will represent image data from the camera. The `read()` method reads a non-specified or specified number of bytes, and stores them into an array to later be decoded.

Once a connection is made and the data input stream begins reading data, the `JPEGImageDecoder` interface and `JPEGCodec` class will be used in order to decode the image data stream so that the image can ultimately be displayed.

Once the `JPEGImageDecoder` is created, it takes an `InputStream` containing JPEG encoded data and decodes it based on the parameter specified in a `JPEGDecodeParam` object. Once the data is decoded, it is returned as a `Raster` or `BufferedImage` object. `BufferedImage` will be used for its simplicity. [14]

2.5.3.1.3 Action Listeners

In order to take in control data from the user through a button push or similar action in the GUI, action listeners will be used. `ActionListener` is an interface for receiving “action events.” Whichever class is associated with processing a certain action implements the `ActionListener` interface. Using the `addActionListener` method, an object created from the class associated with a particular action (ex. A button push) will invoke that object’s `actionPerformed` method when a particular action is performed. The `actionPerformed` method will include instructions to do something based on if the action is performed. For example, if an `actionPerformed` method includes instructions to tell the surveillance vehicle to move forward and this method is associated with a certain button push. When the button is pushed, the method is invoked, and the vehicle moves forward. [14]

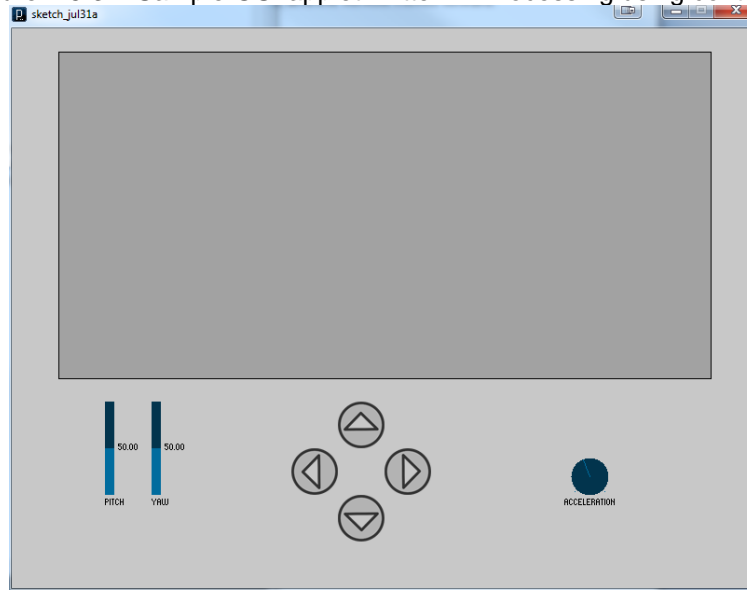
2.5.4 Processing

The Processing software platform makes it very simple to create and run sketches that can easily communicate with embedded devices. This makes it ideal for the surveillance vehicle, which will be controlled by a microprocessor. However, the native libraries of the Processing software do not contain many user control input options. This could make developing a GUI in Processing very difficult. Fortunately, the `controlP5` library was developed by Andreas Shlegel at Sojamo. [15]

2.5.4.1.1 ControlP5 Library

The `controlP5` library allows for the creation and configuration of a wide variety of GUI user input objects such as Buttons, Sliders, and more by simply adding them to the Processing sketch. They can be arranged in separate control windows, and can be organized in tabs or groups. They can then be associated to control events, which work almost exactly like action listeners in Java. Using control events, the GUI controls can be paired with actual control commands on the surveillance vehicle. Figure 2.5.3 shows a representation of the desired GUI written in Processing. [16]

Figure 2.5.3 – Sample GUI applet written in Processing using controlp5



2.5.4.2 C++

The OpenCV tracking software will be written in C++, so it would also be useful to develop the GUI in C++ as well. The GUI would be written using the windows.h library.

2.6 GPS

2.6.1 GPS – Introduction

Having a GPS is required for the S.H.A.S.bot and serves as a vital communication role in autonomous navigation. The main idea behind GPS is to enable users to have a passive low power sensing unit that will provide speed, latitude, longitude, elevation, time, and course information.

The number of channels available on a GPS receiver will affect its accuracy in the data given. Each channel represents a satellite lock and the number of locks directly affects the accuracy of the triangulation, and subsequently all derived information from the calculations using time and satellite position.

2.6.2 GPS – Requirements

Channels- The number of channels does affect the accuracy of the GPS output for location, however, for L1 civilian use there are only a maximum of 12 satellites available for use at any given moment. This will affect the decision such that any module over 12 channels will be able to provide comparable results, all else equal.

Signal quality – Each GPS receiver will be coupled with an antenna or multiple antennas. Different antennas, external or internal, will be able provide varying results in signal quality and will also have different form factor and sizing requirements. For the application in S.H.A.S.bot, it will be required that the antenna be mounted on the chassis and be able to provide the best signal to noise ratio available. High gain antennas will be considered, but the price to performance ratio will be the ultimate factor in the decision.

Ceramic antennas are somewhat bulky compared to the GPS IC, however, they provide the cheapest available option with reasonable performance. Helical antennas are another option available. The disadvantage to using active antennas is that they require external amplification and noise filtering which can add to power consumption. One of the derived goals for the S.H.A.S.bot is lower power consumption, so this option will be weighted accordingly.

Update rate – The fix rate, or update rate, for each GPS receiver should be at least 2 Hz. Some allow for update rates to 20 Hz, which provides a smoother transition between updates and better resolution when changing direction, speed, or elevation. This requirement is set at a minimum of 2 Hz, however, a receiver with a higher update rate will be preferred.

Form factor- The form factor is an important aspect in the PCB layout and design of incorporated the GPS and antenna into the S.H.A.S.bot's system. The GPS module should be small enough to fit onto the main PCB without obstructing and current designs or requiring any special fabrication techniques. A TQFP SMD is expected, as most modules listed in Table 2.6.1 have this option.

Table 2.6.1 – GPS Receiver Module Comparison

Model	Form factor(LxWxH)	Fix rate [Hz]	Power [mW]	Start time [s]	Cost
Venus638FLPx-L	10mm x 10mm x 1.3mm	20	67	Hot: 1 Cold: 29	\$39.95
Fastrax UP501	22mm x 22mm x 8mm	10	75–115	Hot: - Cold: -	\$49.95
u-Blox5H-GS407	47.1mm x 22.9mm x 8.9mm	4	247	Hot: 1 Cold: 29	\$89.95
Copernicus II	19mm x19mm x2.54mm	1	132	Hot: 1 Cold: 38	\$44.95

Start time – The start time for the GPS receiver reflects the amount of time in seconds that it takes to acquire a lock of the minimum number of satellites for a position calculation. The hot start time is used when the module is running in standby mode and has its last valid position saved, which it can then use to acquire the locks with the satellites at a much faster rate compared to a cold start. Cold start is where the GPS receiver is turned on without any previous data saved. Cold starts can take as long as 180 seconds before a signal with adequate quality is acquired, but typically is around 30 to 40 seconds and also

depends on where the GPS receiver is located with respect the satellites and signal quality received from the antenna.

Power consumption – The power consumption is another important aspect of each module that needs to be assessed for the decision process. Since the GPS only acts as a receiver in reference to the satellites, its power consumption is relatively low. Other satellite communication devices require much higher power consumptions due to the required connection. Cell phones use cell towers to distribute the network load and allow the cell tower to do all the high power transmission. [37]

2.6.3 GPS Communication

Global Position Systems used in North America usually use the National Marine Electronics Association (NMEA) 0183 protocol as the output from a GPS stream. This protocol provides a lot of useful information readily available in a serial ASCII stream. There are both newer and older protocols, the older ones are being phased out, which provide standards for protocols to use on the hardware. The NMEA version 0183 is dominant in the U.S. market and will be used in the design of the S.H.A.S.bot. Version 2000 is also available, but is not as readily available with the same volume of information and documentation available.

The NMEA protocol also allows for variable baud rates, which can range from 4800 to 38400. The minimum required baud rate will be set to 9600, for the reason of compatibility and availability among receivers. Although the NMEA protocol is proprietary, the hardware on the GPS receiver module does the decoding for the user real time, which allows the designers to bypass, and required interpreting and decrypting.

The baud rate is defined as the number of symbols per second. The exact definition and application of baud rate can vary based on the device. In the case of GPS, the baud rate is actually much high than the bit rate by using the chip method. [38]

2.7 Communication Systems

Electronic communications, both wireless and wired, will be used extensively in the design and control of the surveillance vehicle. In the following sections, different existing standards of electronic communication, as well as their significance to the project, will be outlined. The chosen microcontroller for the project is an Atmega328 by Atmel so all research will be related to that processor.

2.7.1 Wired Channel Communication

Wired communications will be used in the programming of the microcontrollers, relaying the current status of onboard sensors, and the transmission of data to and from the onboard wireless module.

2.7.1.1 Serial Communication

In serial communication, data is transmitted and received consecutively, one bit at a time over a single channel. This differs from parallel communication, where multiple bits are sent over multiple channels simultaneously. Typically, serial connectors have three wires: one for transmitting data, one for receiving data, and a common ground. The process by which data is transmitted utilizes bit periods, during which a high or low value is active on the wire. A (usually longer) start and stop bit period allows the receiver to know when to record the data bits between the start and stop. First, a start bit is active, to indicate to the receiver that a fragment of data is about to be sent. The receiver then takes in the following data bits until a stop bit is detected. For error detection, an extra bit is often used as a parity bit to sense whether there was an even or odd number of 1's (highs) sent. For instance, the receiver registers that four 1's were sent. The even parity bit sent is a 1. This means that there were an even amount of 1's sent and that the data stream is most likely error-free because four 1's were, indeed, sent.

2.7.1.1.1 RS-232

RS-232 stands for Recommended Standard number 232 and is a serial communications standard still widely used today. Most computers and microcontrollers today, including the Atmel chips used by Arduino, use a subset of the RS-232 standard. When it comes to RS-232, there are two types of devices. The first is Data Terminal Equipment (DTE), which uses a 9-pin connector and is commonly used for serial ports on most personal computers and microcontrollers. The other is Data Communications Equipment (DCE), which uses a 25-pin connector and is commonly used for external devices like modems and printers. The smaller, 9-pin connector is used for the Atmel chips. However, a typical wireless communication shield bypasses the 9-pin connector and is plugged directly into the onboard ports for the Atmel microcontroller. The following Table 2.7.1 illustrates the pin-out for a 9-pin DTE device. [9]

Table 2.7.1 – RS-232 Pin Configuration

Pin Number	Direction of signal
1	(CD) Carrier Detect (from DCE) Incoming Signal from a modem
2	(RD) Received Data (from DCE) Incoming Data
3	(TD) Transmitted Data (to DCE) Outgoing Data
4	(DTR) Data Terminal Ready (to DCE) Outgoing handshaking signal
5	(G) Signal Ground Common reference voltage
6	(DSR) Data Set Ready (from DCE) Incoming handshaking signal
7	(RTS) Ready To Send (to DCE) Outgoing flow control signal
8	(CTS) Clear To Send (from DCE) Incoming flow control signal

TD and RD are the data communication lines which actually send the start/stop bits as well as the symbol data. When these lines are idle, the DTE/DCE device sets it to a mark condition (high).

RTS and CTS are the Ready To Send and Clear To Send lines, respectively. RTS is used by the DTE device to tell the DCE device that it is ready to receive data and the CTS is used by the DCE device to tell the DTE device that it is ready to receive data. They are set to high when the respective device is ready to receive data and low when they are not ready to receive data (usually when the receive buffer is nearly full).

DTR and DSR are the Data Terminal Ready and Data Set Ready lines, respectively. They are similar to the RTS/CTS lines in that when set to high, they indicate that the two devices are connected and their ports are open. They are rarely used because RTS/CTS are used instead.

CD is the Carrier Detect line and it lets you know when a connection between two modems has been made. RI is the Ring Indicator line and a modem toggles the state of this line when an incoming call occurs. [9]

2.7.1.1.2 Baud Rate

The formal definition of the Baud unit (Bd) is the rate that a line changes between high and low states. This is not always the same as bits per second (b/s) because some modems use more sophisticated ways of transferring data like phase modulation, where highs and lows no longer directly represent binary digits. For two different devices to communicate serially, they must be assigned the same baud rate. For the Arduino architecture, you assign baud rates using the command 'begin.' For example, `Serial.begin(9600);` sets the baud rate at 9600 Bd. [12]

2.7.1.1.3 Asynchronous and Synchronous Communication

The two types of serial communications are asynchronous and synchronous. Synchronous is when two separate devices synchronize themselves before sending data, and then continuously send symbols to keep in sync. So, when actual data is not being sent, there still exists a constant character flow. This type of serial communication is faster than asynchronous because start and stop bits do not need to be sent for every character of data. Asynchronous serial communication doesn't require the two devices to be in sync by constantly sending from one to the other. Instead, one device sends a start bit to the other, indicating that it's about to send a character. Then, it sends a stop bit to indicate that all of the character bits have been sent. The advantage of asynchronous serial communication is that the microcontroller does not have to use processing power to constantly send and receive characters when it is idle. Instead, it just sets the serial line to a high ("1") to indicate that it is idle and sets the serial line to a low ("0") to indicate that it is about to transmit data. [11]

2.7.1.1.4 UART and USART

UART stands for Universal Asynchronous Receiver/Transmitter and acts as a buffer between devices in a serial communication system. At the transmitting end, it converts bytes of data into individual bits to be transmitted through a channel. At the receiving end, it converts the bits of data back into bytes of data to be used by the receiving device. The UART is only usable in asynchronous applications. The USART is also widely used and stands for Universal Synchronous-Asynchronous Receiver/Transmitter. It allows for synchronous as well as asynchronous serial communications. The UART/USART can also be used to indicate the state of the transmission media and to regulate the flow of data when the receiving device is not ready to accept more data. A UART is included with most microcontrollers today including the Arduino hardware. The built-in UART on the Atmega microcontroller allows for the chip to work on other tasks while processing other data unrelated to the serial communications. [11]

2.7.1.2 Arduino Communication

An Arduino development board utilizes the RS-232 standard for serial communications between itself and other devices. Its Atmega328 microcontroller uses pins 2 and 3, labeled as digital pins 0 (RX, receive data) and 1 (TX, transmit data), for serial I/O communication. The Arduino has its own software library dedicated to serial communications. This library is called the SoftwareSerial Library. Communication speeds up to 115200 bps can be achieved and the signals can be inverted for devices that use the opposite format. It is important to note that if using multiple serial ports, only one can actively receive data at a time. Also, not all of the pins on the Atmega chip support change interrupts. This is important because the sensors on the stabilization platform will be sending data about the current state and position of the platform constantly in order to keep the platform level. [10]

2.7.2 Wireless Channel Communications

The wireless communication system is essential in transmitting and receiving data to and from the vehicle. Control data for positioning the vehicle and camera, sensor data, and picture data will all be sent wirelessly. It is for this reason that the remote-controlled surveillance vehicle will require a high-quality, dependable wireless communication system. The wireless communications technologies in use today that are most relevant to the project include radio frequency (RF) communications, Infrared (IR) communications, and microwave communications. Due to the relatively short distance limitations of IR technology and the high reliance on line of sight of microwave technology, RF communications will be the primary focus for the surveillance vehicle. Many RF communications systems are low-power, low-cost, and long range. Among these is the Zigbee communications standard.

2.7.2.1 Zigbee

The Zigbee wireless standard is highly applicable to the design of the surveillance vehicle because of its high compatibility with various systems due to its requirement that all products utilizing the Zigbee standard be made accordant with one another. Among these vendors are Atmel, TI, Digi, Samsung, and many more. Also, the low-cost and low-power requirements for Zigbee make it ideal for a surveillance vehicle that has a high probability of being damaged in the field and that will be hundreds of feet from a recharging station. Zigbee operates in the 915 MHz band and is capable of transmission speeds up to 900 kilobits per second.

2.7.2.2 XBee Wireless Communications Module

The XBee Wireless module is a small, RF communications device that allows the transmission of data between itself and another RF module, which can be used to create a wireless channel between a host device and other devices. So the XBee can act as a replacement to the typical corded serial connection. It utilizes a set of communications requirements based on the 802 wireless standard. This set of requirements is called Zigbee. Its long range makes it highly desirable for a reconnaissance vehicle that will be hundreds of feet or more from its control transceiver. The range of the ZNet 2.5 XBee Pro Wireless Module is up to one thousand feet when line of sight is between the transceivers is maintained. [17]

2.7.3 Wireless connectivity (Wi-Fi) 802.11

Wireless connectivity to the bot's camera is requirement since the processing platform will be remote. WLAN networks provide an already well-established infrastructure to build on and allow the use of preexisting hardware that is compatible and secure with these networks. Many facilities already have WLAN networks set up, thus allowing a company to have a fast deployment of security bots available with a minimized set up time that would traditionally be set aside to installation of network cameras. Wi-Fi networks can provide up to a 300 ft range outdoors with proper antenna systems or up to 120 ft. range indoors. This wide range can help cover a very wide area which would be designated for surveillance or coverage.

The main advantage to using Wi-Fi over other systems such as ZigBee (IEEE 802.15.4) is the high data rates one can achieve with a Wi-Fi setup. Typically, bandwidth using WLAN connection can be as high as 54 Mbps, which is more than enough for a video stream with audio.

Along with the camera being wireless, the general communications between the user and robot will be wireless. The servo commands and instructions will be sent via ZigBee, discussed in the communication section, but the option to have it sent all over the 802.11 standard protocols is available. The ZigBee protocol is discussed in its own section, but it should be noted here as part of the research

that the ZigBee protocol, in general, won't be able to provide enough bandwidth for the desired type of video stream.

Many webcams come with onboard Wi-Fi modules which can connect them to infrastructures or as ad-hoc networks. This eliminates the need for networking cables and otherwise obtrusive wire setups to connect the cameras for networking. [17]

2.7.4 Sensor and Data Fusion

The concept of data fusion has a legacy within communication and detection systems. One simple example of this that we see every day is in our stereoscopic vision. This provides us with two unique viewpoints, which when processed by our brains, allow us to derive a third sense of depth of field.

This organic example can be extrapolated onto the autonomous robotic platforms that are arising in research and industry labs. The number of sensors on board a vehicle can provide a range of measurements and measurement types. It may be the case that a range sensor may also give velocity information, such as a radar system, and if another sensor exists aboard that gives related information such as a laser ranging system, it may be possible to have these measurements fused and obtain an average of the two.

2.8 Navigation

2.8.1 Autonomous Navigation

The navigation feature will be able to control the steering and motion of the S.H.A.S.bot through the use of software and sensors aboard the platform. The GPS will provide the data for its location relative to the earth and will also provide heading, speed, and elevation data. The software aboard the S.H.A.S.bot will be capable of making decision based on the information given and the goal at hand. The main idea behind the autonomous navigation feature is to aid in the surveillance mode and provide a larger perimeter for surveying. With the microcontroller making the majority of the decisions, the user can focus more on the external data being received such as the audio and visual information passed through the 802.11 and XBee protocols displayed in the GUI.

In order to be able to navigate itself, routing software must be implemented to tell the motors and servos when and where to turn. The data received on the microcontroller end will be way points associated with the desired locations to scan.

2.8.2 Routing

The given input data must have enough unique information packets, such as waypoints, for the S.H.A.S.bot to make a decision on the quickest and safest way for it to reach the destination or destinations. Without enough information, the S.H.A.S.bot may not be able to execute the commands properly and may encounter physical or electrical errors on its course. The accuracy of the data sent to the microcontroller depends on the signal quality and GPS accuracy, which was previously discussed in the GPS section.

The auto routing feature provides a solution for this. The basic idea behind an auto router is to define a path to the destination that, depending on the parameters given, will either reach the destination at the shortest time or be able to cover the most area for surveillance during its journey. Both of these objectives may be weighted as more important, but that must be decided by the user.

The data received by the microcontroller could be relative or absolute data. If the S.H.A.S.bot uses the GPS module, it can use its absolute location, given in latitude and longitude, to navigate a path given in similar parameters. When the data is relative information, for example, the S.H.A.S.bot must use ranging sensors to navigate in a certain direction or combination of direction and velocity for a given distance. The combination of these two sensors is likely the best option and will be explored further.

2.8.3 Perimeter search/scan

Routines and predetermined routes can provide a limited scope for the image processing to focus on. Even though this scope is limited in terms of area coverage, it is often important to, and advantageous, to focus on high priority areas. At given waypoints and intersections, it may be desired to have the S.H.A.S.bot stop and perform a perimeter search with motion detection enabled. A perimeter scan is when the bot will activate image processing routines and determine if there are any targets, and if so it will start any eligible tracks.

The advantage to have a perimeter scan, instead of constant scanning, allows the S.H.A.S.bot to travel at high speeds and reduce the time interval between high threat and high priority areas. The following Figure 2.8.1 shows the desired field of view.

2.8.4 Obstacle avoidance

Anytime an autonomous vehicle is operating under its own power and it is not being aided by human control, it is important to avoid any unforeseeable or unwanted outcomes. Obstacle avoidance is often used to provide a minimum protection scheme against physical problems and routing barriers.

Since the S.H.A.S.bot will be navigating on various types of terrain, it will be important to position any sensors in an orientation that allows them to operate to their maximum potential and also limit the possibility of false hits or misinterpreted information.

The obstacle avoidance system can be initiated through image processing or through other ranging or proximity sensors mounted on the chassis. The image processing technique could be considerably more complicated and processing intensive as compared to a simple array of range sensors, however, with these data combined it could be possible to improve our resolution and effective success rate associated with obstacle detection and avoidance.

With the S.H.A.S.bot in motion, it's possible to do some back background and foreground detection and identify some potential obstacles. This can be processed with the PC software side and the results can be relayed to other parts of the program or to the user directly.

2.9 Audio

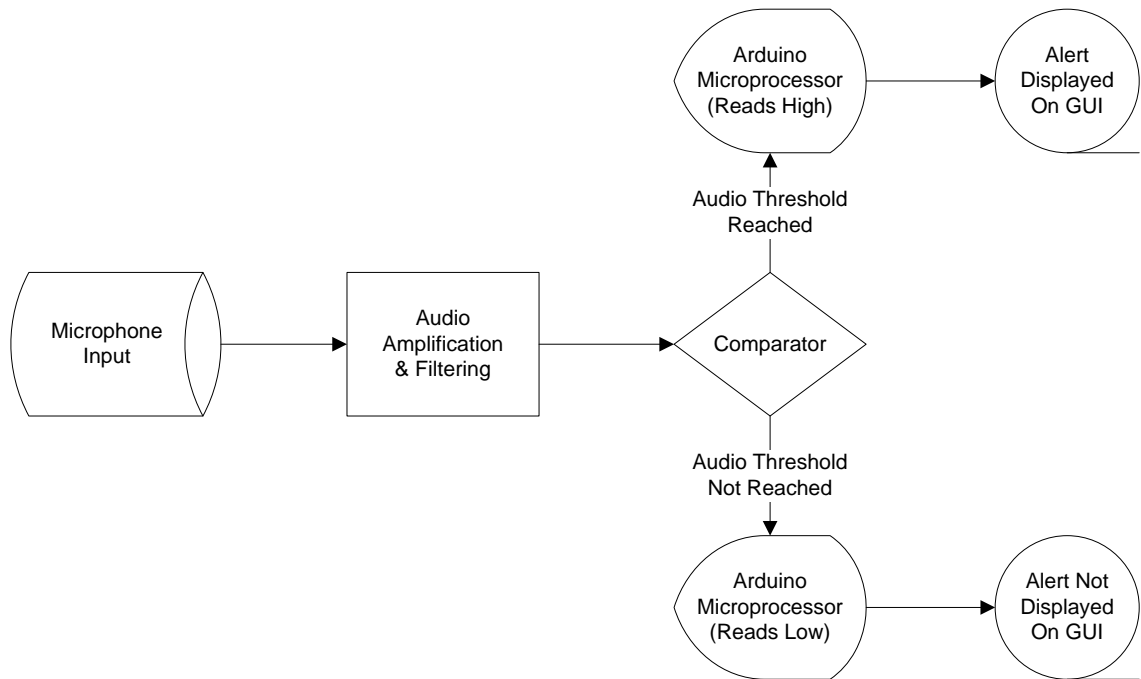
2.9.1 Audio Circuit

The audio circuit that we will design, build, and implement into our overall design will be the path the audio signal takes from the microphone to being digitally processed. Our audio circuit will be close to the design of a typical audio frequency amplifier, but the specifications will be altered seeing as the signal is being fed into our microprocessor instead of a speaker.

Overall, we will want to be able to amplify the signal to the point where it is voltage matched to our microprocessor's input range, and any noise outside of the audible frequency range is filtered out below the noise floor.

In a simple overview of our audio processing chain, there will really only be two steps; the first step being amplification and the second being the filtering. It is done in the order because it gives us the ideal attenuation of non-audible noise (electronic, spectral, etc.). If the amplification came after the filtering, we would be amplifying the noise floor and we would then be dealing with a lot of interference once the signal goes through a comparator. Below the chain is illustrated in Figure 2.9.1.

Figure 2.9.1 – Audio Signal Processing Chain



Our implementation of audio detection in our project greatly simplifies our audio processing needs. For example, if we were creating an audio amplifier that connected to a speaker, we would be very concerned with quality of sound. This would make for some quite complicated audio processing and circuitry. We are only concerned about audio quality to a certain extent, so we can use a simpler, straightforward audio processing circuit. Our biggest concern audio-wise is the gain, or how loud a sound is that was detected by our microphones.

2.9.2 Microphone Modification

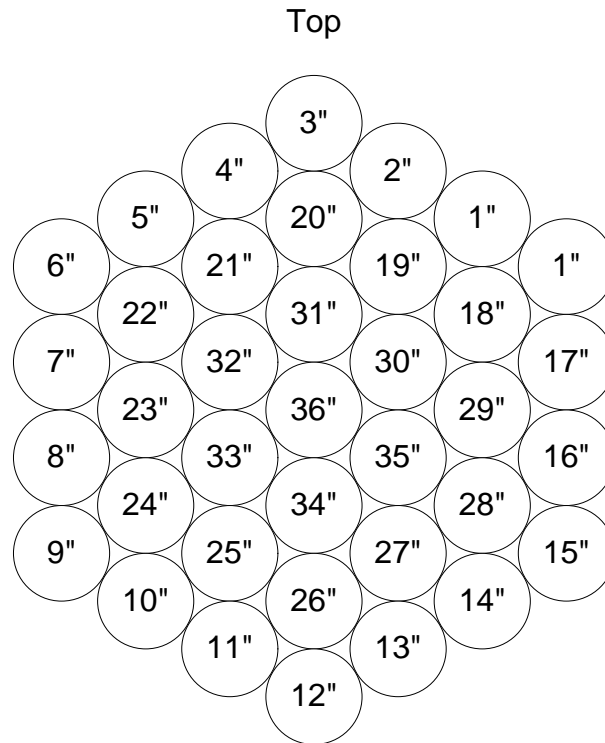
The goal of the audio portion of the project is to successfully detect sounds from specific locations relative to the vehicles position. This requires us to use microphones that pick up sounds in their “line of sight” so to speak. Directional or “shotgun” microphones are very expensive, and since sound quality is not as important to us as the gain of a sound source, paying for a directional microphone would be gratuitous. The much cheaper and feasible option is to purchase much less expensive omni-directional microphones and using the microphone cartridge to modify them into uni-directional microphones. As far as DIY shotgun microphones go, there are two stand out options.

2.9.2.1 Option 1: “Organ Pipe” Arrangement

The organ pipe microphone modification is the more precise of the two options because it allows for a more even pickup of the desired frequency range. Using a series of 3/8” diameter aluminum or plastic tubes, a hexagonal pattern is formed

with the largest tube in the center, with the successively smaller tubes spiraling outwards. The following diagram shows an example arrangement in Figure 2.9.2.

Figure 2.9.2 – Hexagonal microphone pattern



The tube length is determined by and proportional to the specific frequency (cycles per second) of sound you wish to detect. The following formula, equation 2.9.1, determines this.

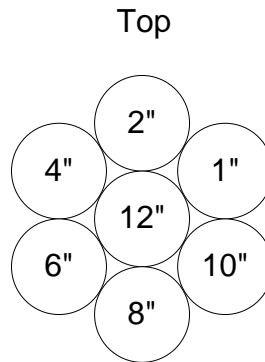
$$Tube\ length = \frac{Speed\ of\ sound}{2 \times Frequency} \quad (Equation\ 2.9.1)$$

In this case the example arrangement pictured above will more amply detect frequencies ranging from 183 Hz to 6.6 kHz.

The microphone cartridge may be mounted a series of different ways, but the most important aspect of the mounting is that the cartridge as close to the tubes as possible. For our applications, a shallow funnel would be the best option.

Since we will be limited on size, we will most likely be limited to 7 tubes. Therefore, our pickup frequency range will not be as precise. A possible arrangement for us could be as follows, seen in Figure 2.9.3.

Figure 2.9.3 – Reduced Hexagonal Microphone Pattern



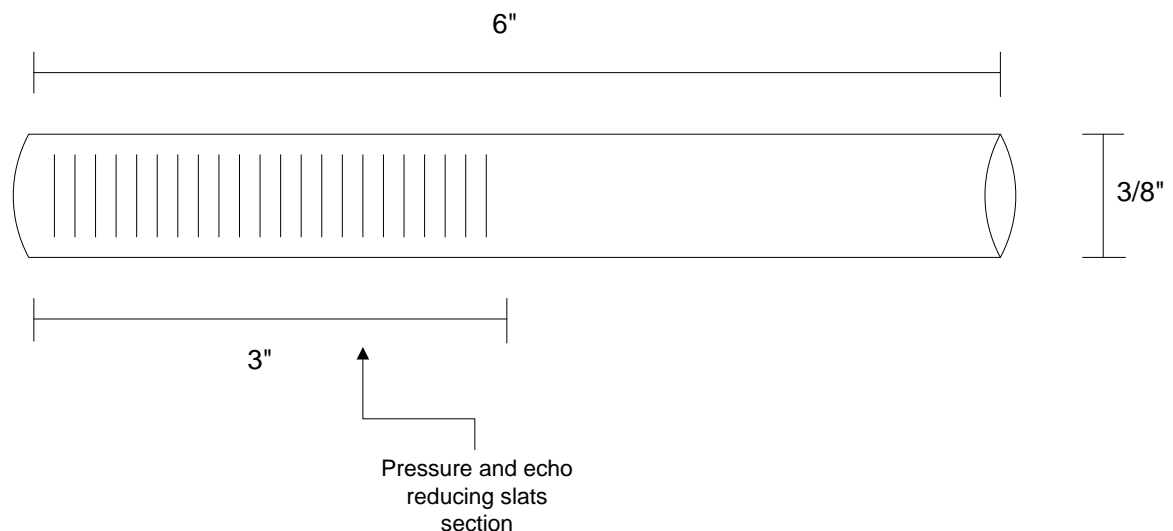
This will adequately pick up a frequency range of 550 Hz to 6.6 kHz. I have chosen the largest pipe to be 12" because any larger would hinder our design and limit mobility.

I want to reiterate that the organ pipe arrangement will not act as a filter, and the microphone will still pickup frequencies outside of the tuned range. What the tubes essentially do is allow the tuned frequencies to be picked up from farther distances and with more precision. [20]

2.9.2.2 Option 2: "Shotgun" Modification

The shotgun design is much more popular in the sound engineering industry because of its portability and functionality. It consists of only one tube with the microphone cartridge at the base. Although not as precise as the organ pipe arrangement, it still does an adequate job of directional sound pickup. The design can be seen below in Figure 2.9.4

Figure 2.9.4 – Shotgun microphone modification



In order to make the sound pickup more sensitive, parallel slats are cut into the sides. This eliminates any sort of echo inside the tube and equalizes the pressure. To fit our design, the length of the tube will have to be around 6" and the diameter will be around 3/8". Like the organ arrangement, the tube material can either be made out of aluminum or plastic. Mounting the microphone cartridge will be a bit easier as it can be inserted into the base of the tube. Mesh material will have to be used to cover the slats and the open ending to ensure that no foreign particles jeopardize the functionality of the microphone.

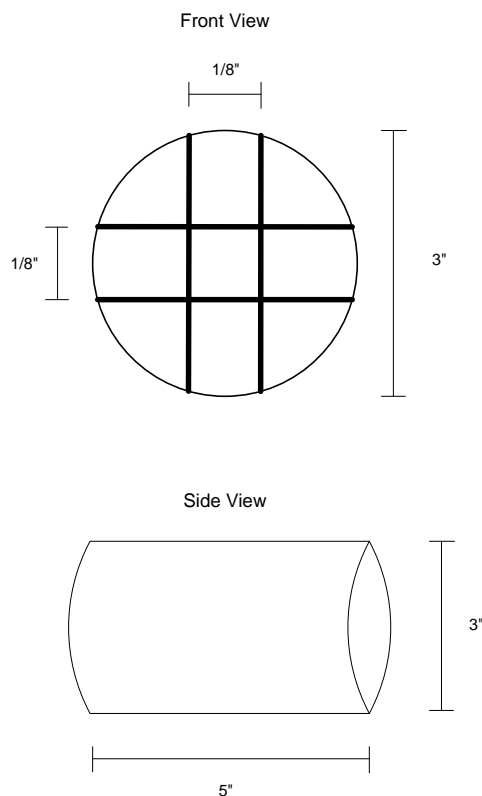
2.9.3 Microphone Mounting

Since our vehicle will encounter all sorts of rough terrain, the microphones will need to be mounted in a way that an adequate amount of shock is absorbed. Note: The organ pipe microphone will not be able to be shock-mounted. These design options only pertain to the shotgun design.

2.9.3.1 Option 1: Free-floating

Free floating designs allow for a large range of movement which keeps the noise level at a minimum. Using a series of several elastic bands in a criss cross pattern, the microphone will be suspended inside of a PVC tube. The diagram below depicts a possible design in Figure 2.9.5.

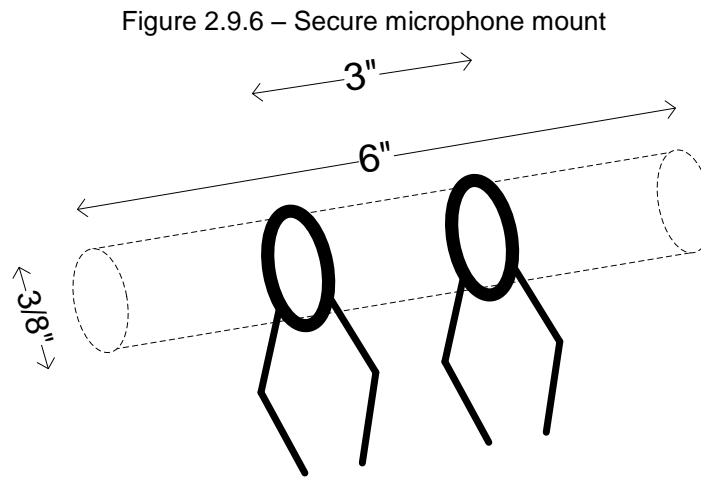
Figure 2.9.5 – Free floating microphone mount



The only concern would be the suspension being too flimsy, and the microphone shifting during vehicle movement. The biggest positive of this option is that it offers the largest noise attenuation. [21]

2.9.3.2 Option 2: Secure

The secure mount design will ensure that the microphone will not shift during vehicle movement. It also offers a smaller, more compact option. The main trade-off, however, is that it won't attenuate as much vehicle noise as the free-floating shock mount. The secure mount will use two circular rubber rings that are the same diameter as the microphone so that it fits extremely snug. The friction from the rubber will hold the microphone in place and will eliminate the concern of shifting during vehicle movement. Below is a diagram of the secure mounting option in Figure 2.9.6.



2.9.4 Microphone Arrangement

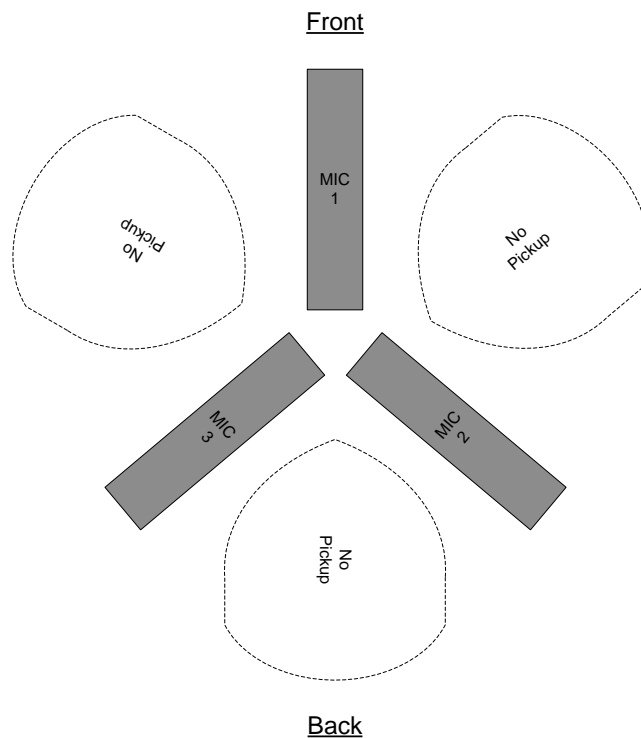
The purpose of having the microphones on our vehicle is to give the user another dimension of awareness while controlling the vehicle. It is of the utmost importance that the user feels as close to being in the vehicle's atmosphere as possible. Even though the user will be able to hear the sounds picked up by an external microphone, they will not have the luxury of a 360 degree stereophonic reference like they would if they were in the vehicle's atmosphere. So we will give the user the next best thing, a visual representation of where a sound is coming from relative to the vehicle's forward direction.

Although we have many options of how the microphones can be arranged, we have to consider many different factors that play into which option we will choose. These factors include, but are not limited to: size constraints, overall cost, overall effectiveness, power efficiency, etc.

2.9.4.1 Option One: Triad

When we want to maximize size, cost, and power efficiency we think of the triad microphone arrangement option. With only three microphones, we will have the least amount of parts to buy and the least amount of power to supply to the audio board. We will also have quite a bit of room to work with. But how well will the triad pick up 360 degrees of sound? The following Figure 2.9.7 shows what the triad arrangement will look like and the regions where sound will not be effectively picked up.

Figure 2.9.7 – Triad microphone configuration



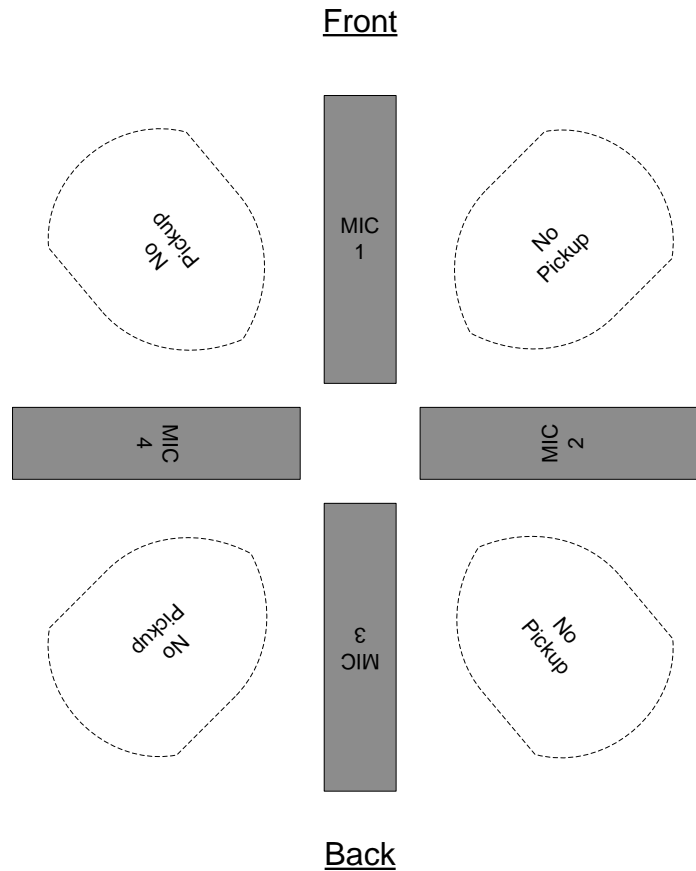
We see that we have run into a problem with overall effectiveness. It doesn't matter how much cost this design saves or how much space it saves, it just doesn't effectively give the user an accurate 360 degree sound detection representation. In the dotted line regions you can see the massive dead zones that this design leaves open. This design, at least for us, is a last resort in case something goes wrong.

2.9.4.2 Option Two: Quadruple

The next microphone arrangement calls for the addition of another microphone and gives us the ability to pick up sounds in the front, left, back, and right of the vehicle. Adding another microphone doesn't add to much to the cost or space of

the project, but will it add a significant enough amount of effectiveness to make this arrangement worth it? Below is a diagram of the design in Figure 2.9.8

Figure 2.9.8 – Quadruple microphone configuration

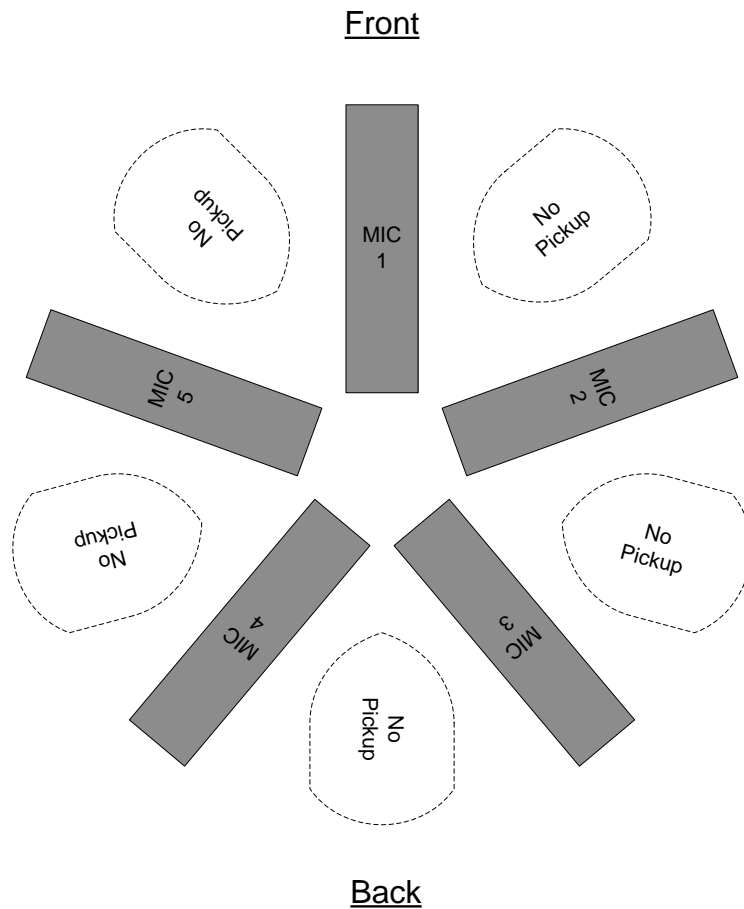


While this design does give the user a better directional sound assessment than the triad arrangement, we still have to remember that these microphones will be very much directional. This design still leaves the opportunity of certain sounds going undetected because they can fall in the direction of the no pickup region. This microphone arrangement still falls short of our effectiveness requirements, but still makes for a better fallback option than the triad arrangement.

2.9.4.3 Option 3: Quintuplet

In the quintuplet arrangement we add yet another microphone to the ensemble, giving the microphones now 72 degrees between them. At this point the addition of the fifth microphone does cut into the size and cost a bit, but gives us an ample boost of effectiveness. You can see by the following Figure 2.9.9 that having five microphones really helps fill in some of the gaps that the quadruple microphone arrangement left open.

Figure 2.9.9 – Quintuplet microphone configuration



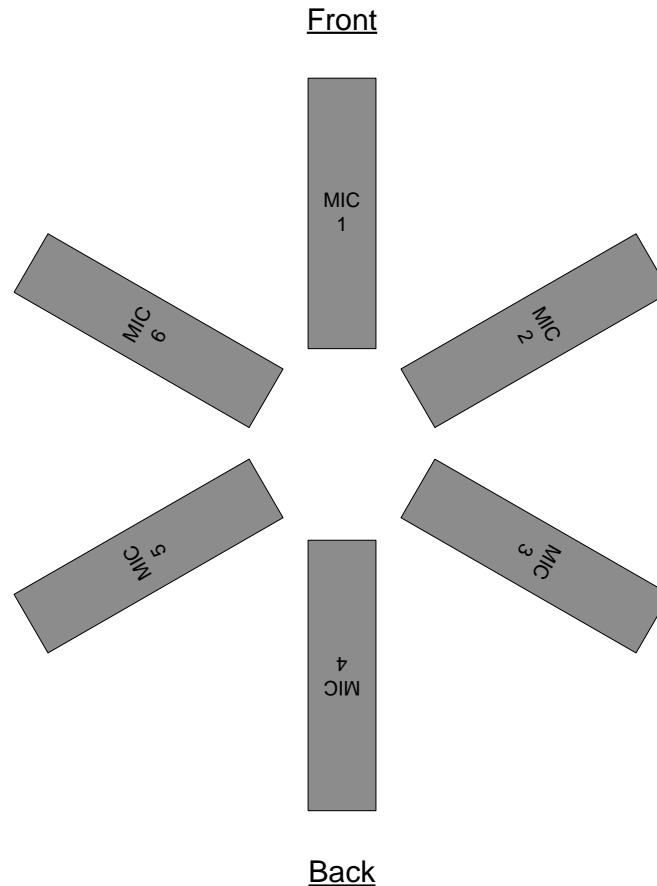
This is where we start to see real improvement in the microphone arrangement's effectiveness. Although there are still visible dead zones in the pickup pattern, they are significantly smaller and will affect the audio surveillance much less than the triad and quadruple arrangements. But this doesn't change the fact that they are still there, and will still affect quality to some extent. As far as an even compromise goes, this is the most balanced option, showing a fairly equal tradeoff in the aspects of size, cost and effectiveness. This arrangement makes for a feasible option for our design.

2.9.4.4 Option 4: Sextuplet

It can be seen in the diagrams of the last three microphone arrangement options that the effectiveness of the arrangement increases exponentially as we add more microphones into the mix. This brings us to our last option of having an arrangement of six microphones on our vehicle. With the microphones placed with equal spacing like the last three options, this leaves only 60 degrees between microphones. When we look at this arrangement visually in Figure 2.9.10 we can see that the effectiveness of the pickup pattern has indeed

increased to a point where the addition of anymore microphones would be gratuitous.

Figure 2.9.10 – Sextuplet microphone configuration



We can see that the no pickup zones have been effectively subdued by adding a sixth microphone to the arrangement. Although the microphones are directional, their “line of sight” so to speak will most likely overlap to the point where no sound will go undetected, especially with our voltage threshold settings (see the section on voltage comparators). The sextuplet option is gives us the highest amount of effectiveness, but does cut into the cost and size efficiency quite a bit. However, this is the option we will want to use if we are to ensure a full 360 degree analysis of the sounds around the vehicle. This is the option we will assume to use unless something changes; we can switch accordingly to one of the other options if need be.

2.9.5 Audio Amplification

When the audio signal from our microphones is received, the first thing that needs to happen to that signal is amplification. We need an amplifier that meets many important criteria. Those criteria are as follows:

- Amplifies frequencies in the audible range (20 – 20000 Hz)
- Amplifies the signal to the correct gain in preparation for filtering
- Operates at the correct voltage
- Has a low enough quiescent current

The microphone should be generating a small signal when picking up the sounds, in the realm of several millivolts. This signal will not be large enough to process once it enters the audio chain. We will need enough amplification to boost the signal enough to where it can withstand a stage of analog filtering while also being high enough to trigger the comparator it comes in contact with. We also need the amplifier chip to be fairly small since we will be dealing with fitting six of these audio chains on the audio board.

We will need to raise the microphone input signal to around line level which is between 0.5V to 2V; however, we could go higher if need be. This will be highly improbable with only one IC amplifier, so we will need an amplifier chain. A typical IC audio amplifier can shell out between 1.2W to over 10W of power. We have many options of audio amplification chains to use, but we must try to choose one where the power efficiency, size efficiency, and cost efficiency tradeoffs are appropriate.

Let's say our target line level is 2V. We would need a voltage gain close to 1400 to have the small microphone input signal amplified to the desired voltage.

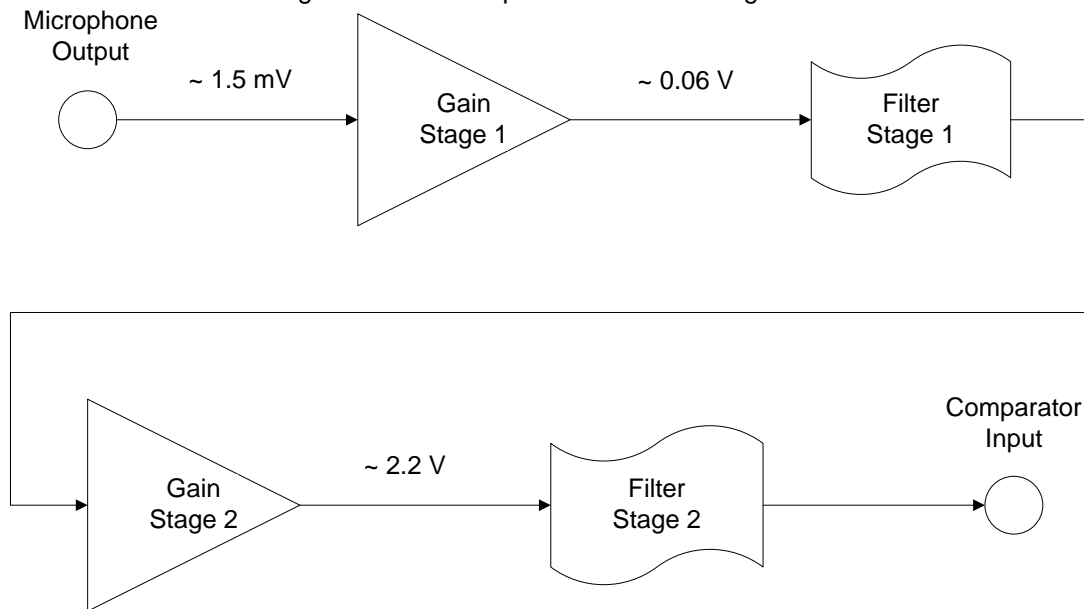
2.9.5.1 Feasible Amplification Option

Take for example an ideal 1.5W audio amplifier. This produces a gain of about 31.7 dBm, which is a voltage gain of 38.5. Since amplifiers in series multiply their respective gains together, we would only need two of them to reach a small signal voltage gain of around 1400. This may seem high, but we are dealing with very small signals. The signal going into the second amplifier stage will be about 0.06V, which is well within the input range of whatever amplifier we will use.

We need to be careful though, because there could be a cause for concern with the noise involved. Two amplifiers directly in series could pose problems with feedback or with the amplification of unwanted frequencies. This can be solved somewhat with a filter thrown in between the two amplifiers. Of course there will be another filter after the second amplifier but that will be covered mostly in the filter section.

The drawing below in Figure 2.9.11 depicts what the amplification and filtering chain will most likely end up being.

Figure 2.9.11 – Amplification and filtering chain



The voltage values shown are based on a 1.5 mV microphone input. This is the estimated threshold of the microphones we will use since they will be simple cartridges. The intermittent voltages are shown between the gain and filter stages. These voltages will obviously be different based on how powerful the sound is that the microphone picks up, but 1.5 mV is the estimated threshold.

2.9.6 Audio Filtering

After the incoming audio signal is amplified, we need to prepare it for the comparator that it will be processed by. This will be done through frequency specific filtering. In order to send a clean signal to the comparator, we will need to filter out any frequencies that are not within the realm of the human hearing frequency range, which is roughly between 20 Hz and 20000 Hz. [22]

We don't have to worry too much about the low threshold of human hearing, 20 Hz, because it is so low. For the high threshold however, we will need to filter out the frequencies above 20000 Hz. These higher frequencies are unwanted and can be the product of interfering electronic noise. Therefore, we are looking for some sort of low pass filter with a cutoff frequency preferably no more than 5000 Hz above the human threshold.

We have many options for parts since there are many filters out there that meet our criteria. However, we want to take size into account since we will be dealing with six audio chains, one for each microphone. We also want to take into account insertion loss, how much power our amplified signal will lose once it goes through filtering. The filter can either be passive or active, but active filters have better attenuation of out of band frequencies and will be the better option if powering them is feasible.

2.9.6.1 Filter Stages

Since we will have two amplifiers in series, we will need to place a filter in between them for practical purposes. We want to be able to filter out unwanted frequencies before the signal hits the second gain stage. Also, this intermittent filter will protect against unwanted feedback.

The second filter stage will be right after the second gain stage, to make sure that any more noise or unwanted frequencies are dealt with. The signal will then be clean enough to feed to the comparator.

2.9.7 Voltage Comparator

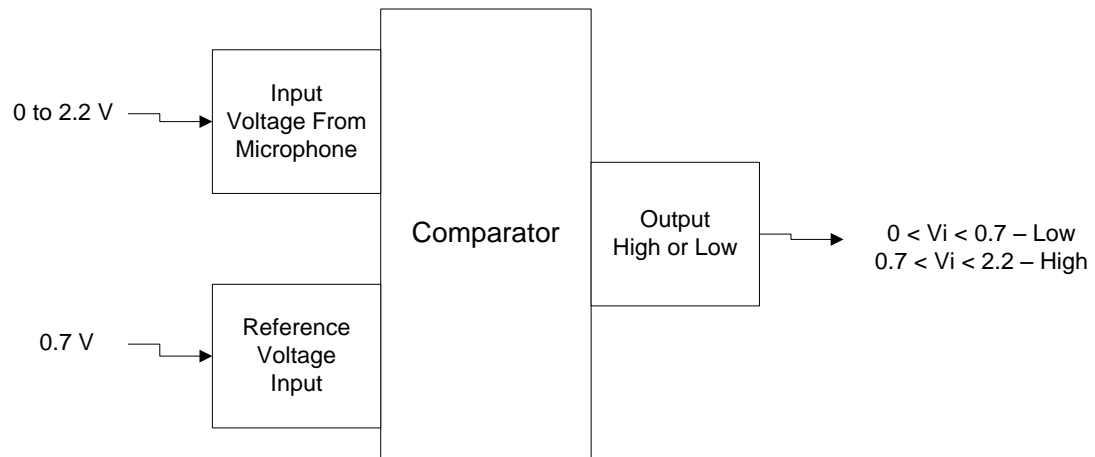
When sound waves come in contact with any one of our six microphones, the microphone cartridge will generate a small voltage signal. This signal will most likely be between 0 mV to 1.5 mV. It will be amplified to a scale of 1.5 mV to around 2.2 V. But the dilemma we run into here is, how high does the microphone input have to be to constitute a critical event? We don't want to accept ambient sounds that will be very common in our vehicle's atmosphere, but we also don't want to reject sounds that are significant enough to warrant an alert. Since we don't have the tools necessary to test this scenario and solve the dilemma at this point, we have to make an arbitrary guess as to where that desired user threshold will lie. Taking into account the magnitude of sounds this vehicle will detect, a safe guess for the low threshold would be around 0.5 mV.

Once the audio signal is amplified, the chosen threshold voltage of 0.5 mV will be increased to around 0.7 V. When this signal leaves the gain and filter stage, it must be processed by a voltage comparator to determine if the signal is high enough to warrant an alert.

No matter what comparator we end up using, we will only need to use three pins: one for the input from the microphone, one for a reference voltage, and one for the output. The reference voltage input will be 0.7 V, so if the voltage from the microphone exceeds the 0.7 V threshold, the output voltage will be high. If the voltage from the microphone is less than 0.7 V, then the output of the comparator will be low. The output signal will then be sent to the microprocessor.

The drawing below in Figure 2.9.12 shows how we will incorporate the comparator into our audio board design. There are no other pins included because no other pins will be used.

Figure 2.9.12 – Comparator I/O diagram



2.9.8 Audio Arduino Input (Analog)

Once the audio signal is fed through the comparator, it will output one of two possible voltages: either a high voltage (due to the microphone input being over the event threshold) or a low voltage (due to the microphone input being below the event threshold). Those high and low voltages will differ based on the comparator we use but the concept will be the same once the signal hits the analog input pin on the Arduino board.

The Arduino microprocessor has a set of analog input pins aptly named `analogRead()`. Depending on the type of Arduino chip we use, there could be 6 channels (on most boards), 8 channels (on the Mini and Nano), or 16 channels (on the Mega). Since we will only need 6 `analogRead()` channels, any type of Arduino chip will work to our advantage. [23]

Each one of the `analogRead()` inputs is equipped with a 10 bit analog to digital converter. This means that each `analogRead()` channel has a resolution of 1023 integer values. Each `analogRead()` input has a 0 V to 5 V input range. Therefore, each `analogRead()` channel will have an accuracy down to the $5/1023$ V, or about 5 mV. Obviously we don't need this kind of resolution as we will only need the channel to be able to differentiate between the high and low voltages coming out of the voltage comparator. On the software side, the resolution can be easily changed with the function `analogReference()`.

Since the difference between the high and low voltages from the comparator output will be relatively far apart (relative to the resolution), the highest resolution we would want would probably be around 0.05 V, although the increment could obviously be larger.

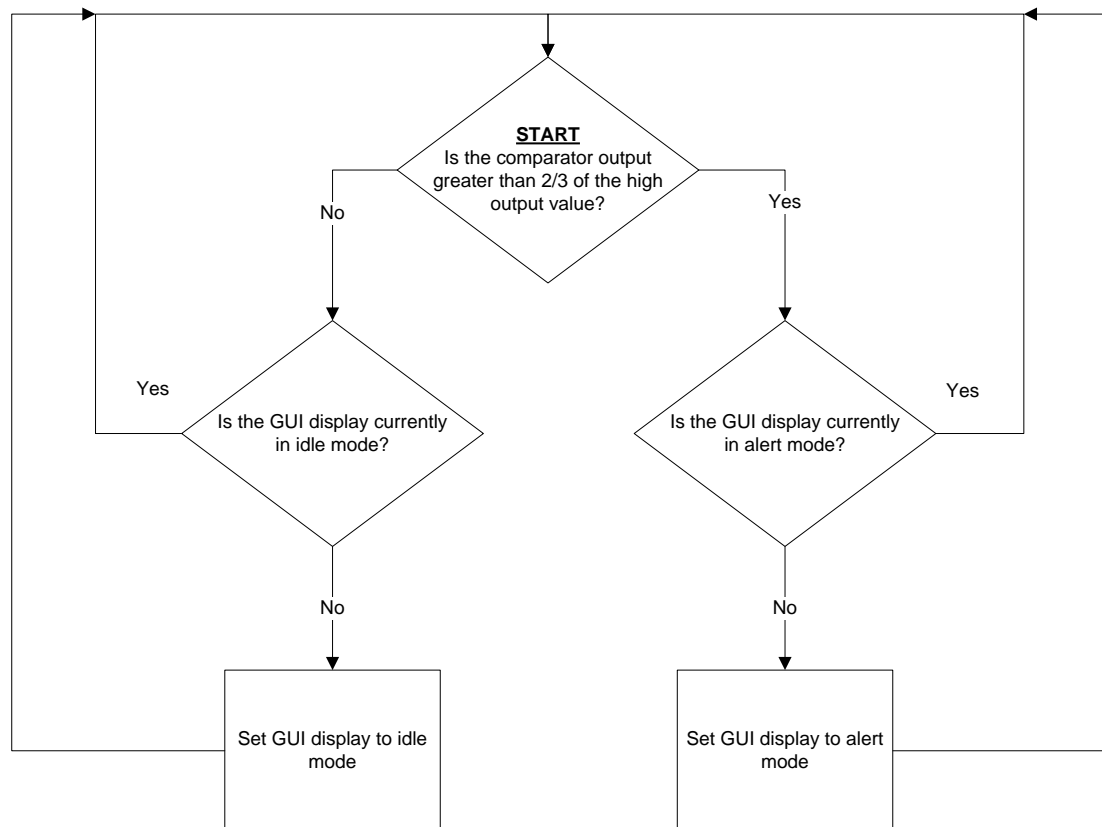
2.9.9 Arduino Audio Control Loop

On the software side of things, we need a constant control loop running to check for changes in the voltage on the analogRead() input pins. Obviously we need these loops (or single loop if possible) to be cycling through each of the six inputs. If the voltage is low, no changes need to be displayed on the GUI. If the voltage is high, some sort of alert needs to be displayed, but I will go into more detail on the display later on. We need to set the parameters on the software control loop based on what the low and high output voltages of the voltage comparator are.

We will need to choose a reference voltage as the threshold that will trigger a change on the GUI. It makes sense to choose this reference voltage somewhere below the high output voltage of the comparator since the difference between those high and low voltages will be relatively large. So we will choose a reference voltage to be about $\frac{2}{3}$ the value of the comparators high voltage output. The Arduino chip will be interfaced with the GUI, and the idle and alert modes will affect the GUI audio display accordingly.

Below is a diagram of the software audio control loop in Figure 2.9.13.

Figure 2.9.13 - Audio processing control loop



3 DESIGN

3.1 Chassis

3.1.1 Frame

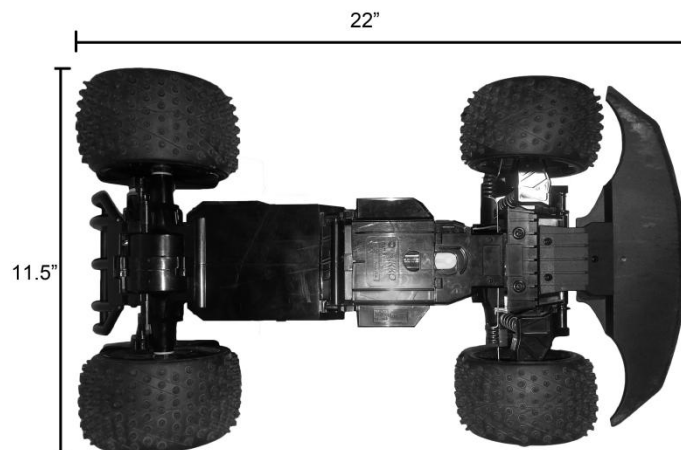
A prebuilt chassis called the Nikko RC buggy available for purchase online provided us with adequate dimensions, two wheeled drive, wheels, motors, and a plastic chassis to build our design off of. This, or the similar, allows our system to travel on different types of terrain with speeds up to 6 MPH. There are other options that were available offered different track systems and motors, however, for the price it was decided that this chassis provides the most amount of room for development and all-terrain capabilities. Table 3.1.1 outlines the basic specifications of the chassis.

Table 3.1.1 – Nikko RC buggy specs

Length	Width	Height	Weight	Speed
22.00"	11.50"	5.125"	4 lb. 2 oz.	6.8 MPH

With these specifications, it was possible to fully utilize at least 144 in² available on the upper deck for the stabilization platform, servos, antenna, camera, and audio detection equipment. The battery, microcontroller, and other equipment could be mounted inside the chassis. This allowed not only for maximum utilization of surface area and volume available, but also provided for a safe mounting of external hardware that could endure physical damage during testing if not safeguarded properly. The plastic chassis provided this protection. Figure 3.1.1 features the mechanical dimensions of the chassis

Figure 3.1.1 – Nikko RC buggy chassis



The frame chosen is made of a combination of plastic and aluminum mounts. It is prefabricated by Nikko. The dimensions just the frame alone provided enough

surface area and internal volume for the mounting of the sensors, microcontroller, motors, battery, and PCB. The stabilization platform was mounted on top of the chassis and bolted down using the fabricated mounting hardware available from Skycraft. Table 3.1.2 gives additional mounting platform dimensions and specifications.

Table 3.1.2 – Frame (only) Dimensions and Specifications

Length	Width	Height from ground
16.00"	8.00"	5.13"

The main advantage in using a prefabricated frame is the time saved in building a comparable option. The frame used met the minimum requirements and offers a good value for the money spent.

3.1.2 Wheels

The wheels used in the design for the rear were 5.00" in diameter and will be directly mounted on the RC car rear drive motor. This enabled the software to use front wheel steering driven system with 4.25" wheels which eliminated the need for any differentials. The wheels that were included in with the chassis were from the prebuilt RC chassis were built for all-terrain use and were lightweight and already tested for compatibility with the chassis and motors. Other wheels may be added on or used in place of the current set for a higher desired speed or for terrain specific applications. The mounting system was fairly basic and allows for these kinds of modifications.

3.1.3 Motor Selection

The RC car chassis used 2 DC motors. These brushed motors operate between 6 and 12V. They had a current draw at 12V with no load close to 0 mA. We used these motors because it alleviates the need for further mechanical analysis on our project, and we can get them in a package with our chassis, we decreased the overall cost. We tested the motors available with the motor controller and added heavy loads. The motors preformed adequately and were determined to be sufficient for our needs.

3.1.4 Chassis Frame Selection

After much consideration of time and budget, we decided to use a pre-built Nikko brand remote control car chassis frame. The chassis features a rigid, low-profile frame with built-in passive shock absorbers. This design is ideal for a stable video feed via the mounted camera. The chassis has two built-in motors, one high-torque steering gearbox motor and one rear vehicle drive motor. These will be controlled using a motor controller. Since it is not a servo motor, the gearbox

motor has a built in potentiometer for steering angle sensing and will be connected to an analog-to-digital line on the chosen microcontroller, an Atmega328.

3.1.5 Motor Controller Selection

We looked at a variety of motor controllers to handle our application based off of the following criteria:

- Cost
- Output Current
- Input Voltage Range
- Control

The motor controller needed to be able to handle up to 16.8V, as our Lithium Ion Polymer has the potential to output that amount of voltage, and the motor controller must also be able to output at least 6A of current for the worst case current draw of our motors. Control is important because the easier our microcontroller can communicate with the motor controller, the better. Cost is what was used to determine the right motor controller among a group of similarly designed controllers. Table 3.1.3 provides a variety of motor controllers and their respective characteristics.

Table 3.1.3 – Motor Controller Comparison Table

Motor Controller	Input Voltage Range	Output Current	Control Options	Cost
Seedstudio motor shield	6-15V	Up to 2A per channel	TTL Serial	\$59.99
Sabertooth 2X12	6-24V	Up to 12A per channel	Analog, R/C, TTL Serial	\$79.99
RoboClaw 2X15	6-30V	Up to 15A per channel	TTL Serial, RC, Analog	\$89.95
Pololu 24v23	5.5-40V	Up to 23A per channel	Analog, TTL Serial, RC	\$59.95
Devantech MD03	5-50V	Up to 20A per channel	I2C, RC, Analog, TTL Serial	\$108.95

Each of the listed motor controllers offered a convenient method for communication with our ATmega328 microcontroller. The Sabertooth 2x5 offered the least amount of current per channel, while the Pololu model offered the most. As stated previously, we had to be able to handle at least 6A of current with our motor controller in order to have full functionality. Even if each of these motor controllers only had a single channel, we would easily be able to handle our current needs with these. The Sabertooth 2x5 an ideal solution because it can handle our required input voltage range and costs the second least. It also can output more than enough amperage per channel, so the power of our motor

controller will not be wasted. The Devantech was the most expensive and did not output the most amperage per channel, but it can easily handle the most input voltage, with the Pololu coming in second in terms of most input voltage, but as the least expensive option. The Sabertooth models were desirable because they had protection against lithium ion polymer batteries outputting too much current to the device, and the Sabertooth 2X12 came as the recommended motor controller for our chassis. The Sabertooth is also in the middle in terms of price. We chose the Seeedstudio motorcontroller simply because it satisfied our requirements set for each motor and was one of the cheaper motorcontroller shields available.

3.1.6 GPS

In order to meet the requirements and specifications for the S.H.A.S.bot's performance, cost, and power consumption, the requirements were set at having the lowest power consumption possible while maintaining accuracy less than 3 meters or 9.84 feet. For GPS receivers this accuracy is usually possible with all 12 satellite links established along with a superior link quality, which can be attained with a receiver that has high sensitivity and low susceptibility to noise and interference.

The GPS module used was the Venus638FLPx-L surface mount receiver made by SkyTraq. This module has a very small PCB footprint of only 100 mm² but still provides high performance and low power consumption. The antenna in this version can be either passive or active.

The breakout board for this board was used. The output of all the pins are available via a header pin, which can be plugged directly into our board as an expansion feature.

3.1.6.1 Antenna

The antenna requirements were set for the antenna to achieve the maximum performance for the GPS receiver and minimize power consumption. As noted in the research, an antenna that is active will use considerably more power, which may put the total power consumption over specifications. In order to circumvent this, a passive antenna will be used.

The passive antenna was a simpler and cheaper design, compatible with many different modules. The antenna used was able to prove a cold start time of less than 60 seconds. This time length was set due to the fact that the specifications in the manual may be set using a GPS module that has superior sensitivity, which would reduce the time it takes during testing.

3.1.6.2 Back up battery – Super Capacitor

The VBAT pin on the Venus GPS receiver allowed for a backup battery to be used in conjunction with a power supply. This allowed the internal circuitry and memory to stay in operation while the main system is shut down intentionally or due to power outage or power supply instability.

The backup battery chosen was the surface mount Panasonic 0.2 F capacitor. This capacitor can withstand voltages up to 3.3 V, which matched our voltage requirement and has enough energy to power the internal circuitry for up to 7 hours. If desired, these can be placed in parallel to double or even triple the battery life. Figure 3.1.4 features a picture of the supercapacitor.

Figure 3.1.4 – Panasonic 0.2 F capacitor

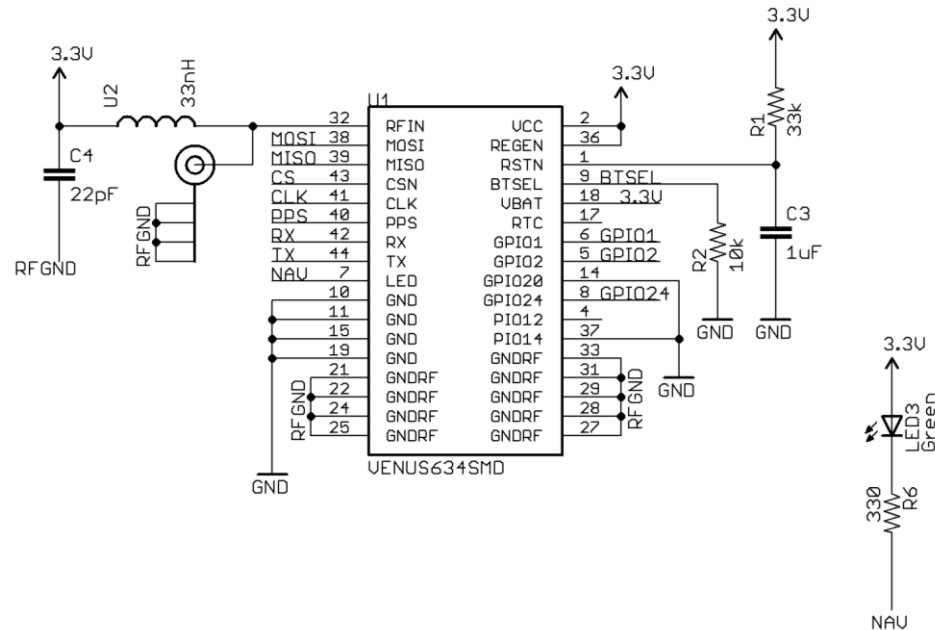


As shown above, the capacitor will be placed in the circuit with the positive terminal connected to the VBAT input on the GPS and the negative terminal to GND. This supercapacitor ended up not being used, as we decided to have the GPS module as an expandable option for the stabilization board. The female header pins were included in the PCB design and soldered on.

3.1.6.3 Mounting

The GPS had the ability to be mounted on the main PCB. Pin number 57, which is the TX1 pin, will be sending the serial GPS data to the microcontroller. Depending on which pin is used on the microcontroller end, the pin will need to be setup to read and interpret the GPS data at the appropriate baud rate. By default the baud is set to 9600, however, this could be modified by connecting the GPS RX1 and TX1 pins. The schematic in Figure 3.1.5 for the Venus GPS modules below shows how the module only requires one pin for the connectivity between the microcontroller and GPS.

Figure 3.1.5 – Schematic with ATmega328 (TX + RX), Venus GPS, and active/passive antennas

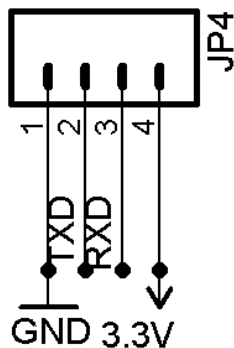


Used under Creative Commons permission from SparkFun

The power and ground were regulated externally and the antenna used did not require use with an inductor, which is used for active antennas. The antenna will have its own ground for RF communication and isolation. The backup capacitor is also indicated in the schematic above. The following schematic in Figure 3.1.6 shows how the Venus GPS Module will be wired to the ATmega328p. As seen below, only the RX and TX lines are needed for communications to the MCU.

Figure 3.1.6 – Venus GPS Module Header Schematic

GPS HEADER



3.2 Power

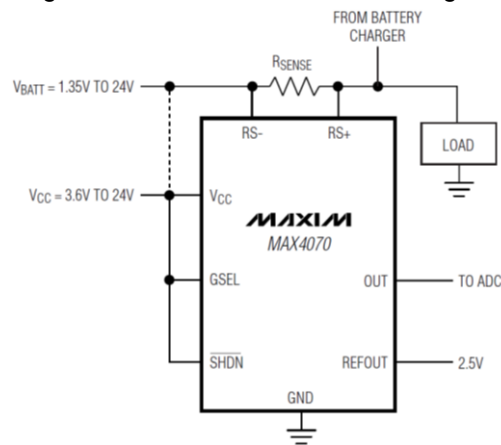
3.2.1 Battery Selection

Originally, we were planning to use a 14.8V, 5.7Ah rechargeable lithium ion polymer battery pack. The battery, the PLH-7051119-4S-WR, came with many features, such as a power control module to balance all of the cells, a 7A polyswitch to protect from reverse polarity and limit maximum discharge, and a lightweight frame. Instead, we chose to go with a Tenenergy 11.1V, 6500mAh LIPO battery. It had similar features as the PLH-7051119-4S-WR, but with a lower voltage and a higher current rating. The PCM unit requires 30 minutes after the battery is fully charged to balance all of the cells. A 7A polyswitch also comes preinstalled to protect from reverse polarity and limit the maximum discharge current. The battery pack is 126mm long by 44mm wide by 62mm high, and weighs only about 1.2lb, so it was ideal for our chosen chassis. Because the battery is lithium ion polymer, we paid special attention to the pack while it was being charged, and handled it carefully. These types of batteries are known to combust when overcharged, and safety is of utmost importance. The estimated charge time is 5.7 hours. The battery pack comes pre-wired with a 6 inch 18 AWG wire without a connector, so we would need to find an appropriate header or connector kit to use in conjunction with this battery.

3.2.2 Current Sensing Design

Based on the information provided in section 2.2.2, we decided that high-side current sensing is the best current sensing method for our application. The Maxim MAX4070 offers a compact current sensing solution with a total output error of less than 1.5%, a selectable gain of 50V/V or 100V/V, with a possible voltage supply of 1.35 to 24V. This is an ideal current sensing IC because it only requires 100uA supply current, and comes in a small 8-pin package. Figure 3.2.1 shows the general configuration of the MAX4070 with respect to a circuit.

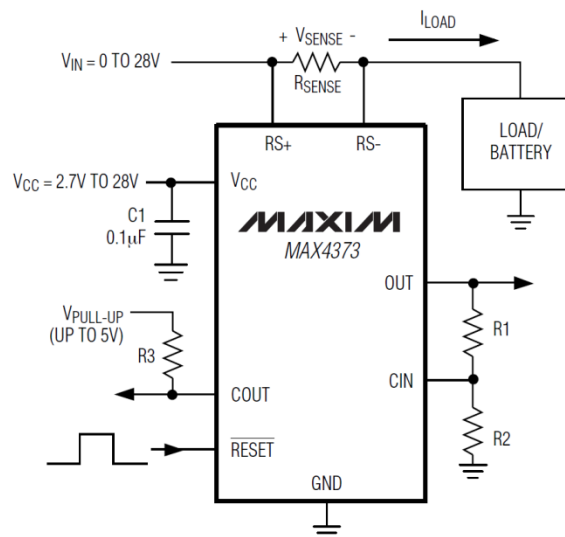
Figure 3.2.1 - MAX4070 Sensor Diagram



Used with permission from Maxim Integrated Products

Another viable option from Maxim is the MAX4373. This is a low-cost, low-power, high-side current sensor that contains the current sense amplifier, a bandgap reference, and a comparator with latching output. It requires a single 2.7 to 28V power supply and only consumes 50uA supply current. It also comes in 8- or 10-pin packages, with a 1mV input offset voltage, 2% full-scale accuracy, and has three gains available at 20V/V, 50V/V, or 100V/V. Figure 3.2.2 shows a typical circuit using the MAX4373.

Figure 3.2.2 – MAX4373 Sensor Diagram



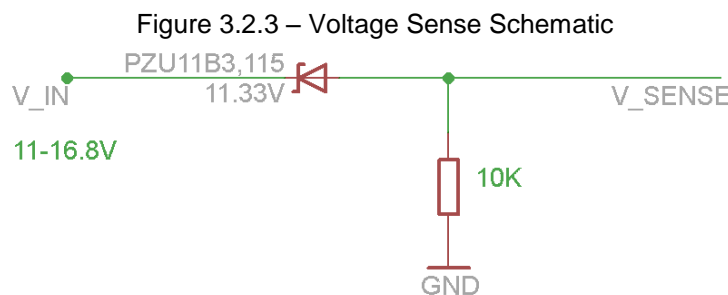
Used with permission from Maxim Integrated Products

The LT6100 from Linear technology is a unidirectional current sense amplifier that monitors the voltage across a sense resistor. It has selectable gains of 10, 12.5, 20, 25, 40, and 50V/V, chosen by strapping or floating two gain select pins. This IC operates from 2.7 to 36V and has a current draw of 60uA. It is available in 8-lead DFN and MSOP packages and has a maximum input offset voltage of 300uV. Each of these packages requires an external analog-to-digital converter in order to create a fully functioning current sensor. If we choose to go for an all-in-one package, the MAX9611 is an ideal solution. It comes with an integrated 12-bit analog-to-digital converter with a gain block that can operate as either an op-amp or a comparator. The analog-to-digital converter is controlled via I2C and is compatible with 1.8 or 3.3V, which will be provided by our 3.3V regulator. This device operates between 2.7 and 5.5V, and comes in a 3x5mm, 10-pin package. The ADC has current sensing resolution down to 13uV, making it extremely accurate. Among these choices, the MAX9611 appeals the most because it offers an inclusive package that eliminates the need for further research for applicable converters and saves board space. The two downsides is that it comes at a price point of \$2.68 for a single unit, and that the increased complexity of the IC means there is the potential for increased difficulty in making the device operate correctly. However, we decided in the end to use the

ISL28006 current sensor from Intersil. We already had this IC available to us, and the convenience proved to be worth the tradeoff. It is a low current consumption device, requiring only 50uA in order to operate, can be powered from anywhere between 2.7V and 28V, and had a 100V/V gain voltage output representation of the current. If we had to buy the device, it would only cost us about \$0.99, which means we would have saved money with this selection anyways.

3.2.3 Voltage Sensing Design

Our battery has a 9V cutoff voltage, 11.1V working voltage, and 12.6V peak voltage, therefore creating a difference of 3.6V between its minimum and maximum ratings. The Atmega328 can take voltages up to 5.5V in an I/O pin. In order to properly design the voltage sensing scheme, we needed to find a way to limit the voltage that the analog-to-digital converter of the Atmega328 could receive. Mouser Electronics offers a zener diode rated at 11.33V, allowing us to safely scale the input voltage that the microcontroller can receive to about 5.5V. The Atmega328 uses 10-bit analog-to-digital converters, which means that we will be able to sense the voltage with a resolution of about 5.37mV per bit, which is more than acceptable accuracy on a power source that operates at 11.1V. We could have used the PZU11B3,115 zener diode from NXP, which has a rated voltage of 11.33V with a tolerance of 2%, 550mW of power dissipation, and comes in a surface mounting package, and we could have used a 10kΩ resistor in parallel with the connection to the Atmega328. Figure 3.2.3 shows the schematic for the voltage sensing circuit. We decided in the end to simply limit the voltage down to a safe level for the Atmega328 by using a voltage divider to bring the voltage down to about 5V.



3.2.4 Reverse Polarity Protection Design

The selection criteria for our reverse polarity protection implementation method were as follows:

- Space requirement
- Monetary Cost
- Power Cost
- Complexity

Of the criteria listed, power cost stood as the most important, followed by monetary cost, space requirement, and complexity of implementation. The diode method was the easiest to implement, had the smallest space requirements, and the lowest cost. Its drawback came from having a high voltage requirement, which unfortunately was too great to consider as the final solution. Full-wave rectification was a convenience factor to it beyond the other methods listed, but cost more money than a single diode, and required more space to implement as well. The drawbacks were the unspeakably high voltage cost, practically double the single diode voltage cost, and the greater space requirements. The FET protection, on the other hand, had an extremely small cost in terms of both voltage requirements and monetary cost, and a decently small circuit footprint as well. Ignoring other effects, the FET was extremely simple to implement as well. For these reasons, the FET protection method was utilized in our design.

3.2.5 Voltage Regulation

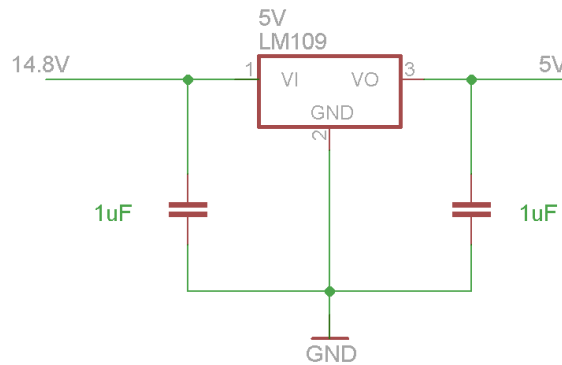
3.2.5.1 12V Regulation

The use of a 12V regulator may not be required for our application, as the motors and motor driver we will be using may require too much current to be handled safely and effectively on our PCB. We chose a regulator, however, in order to prevent a rushed design based on future evidence supporting the onboard regulation requirement. The TPS54383 was a solid choice due to its dual-output channels that each allow up to 3A of current. This eliminated the need for multiple regulators, as each output terminal could be used to power two of our motors. The disadvantage to using this device is that it is much more complicated than other types of regulators, such as a traditional LDO fixed voltage regulator. The possibility of output noise due to it being a switching regulator is accounted for due to the motor controller being able to accept voltages well beyond the 12V output we desire. The desire for using this regulator to begin was simply to be able to effectively measure the current traveling to the motors for a more accurate power consumption reading. We did not need 12V regulation for our final project.

3.2.5.2 5V Regulation

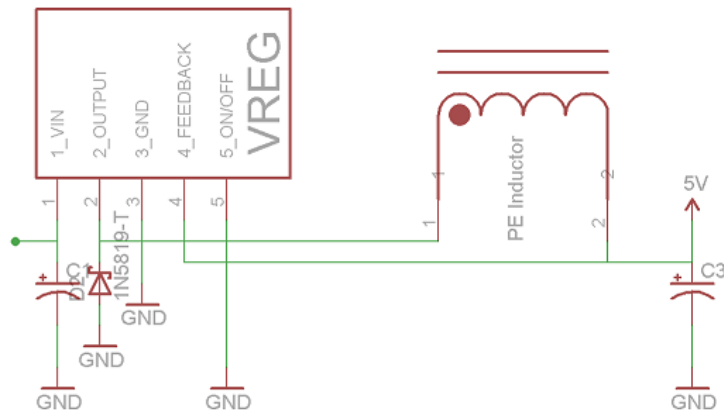
We originally decided to use the LM109 5V regulator from Texas Instruments for its convenient packaging, thermal overloading protection, and current limiting capabilities. Figure 3.2.4 shows how the regulator would have been configured on our PCB.

Figure 3.2.4 – 5V Regulator Schematic



The output capacitor is there to ensure a smooth transient response, while the input capacitor is recommended by the datasheet for filtering purposes. The datasheet recommends that these capacitors be solid tantalum. The LM109 is also capable of an adjusted output voltage with the use of a potentiometer and resistor network. This type of setup would have allowed us to achieve regulated voltages of over 5V, while we could use a simple resistor divider network to achieve output voltages of lower than 5V. This may have been useful if future changes required voltages in the 4V range as opposed to just the 5V and 3.3V range. But we realized that switching voltage regulators, though they are more complex and require more and larger components, offer a much more stable voltage line capable of providing higher currents with less heat dissipation. So we used another product from Texas Instruments, the LM2576 5V switching regulator. This regulator required the use of an inductor, a diode, and two capacitors in order to function properly, as shown in figure 3.2.5.

Figure 3.2.5 – 5V+ Regulator Schematic

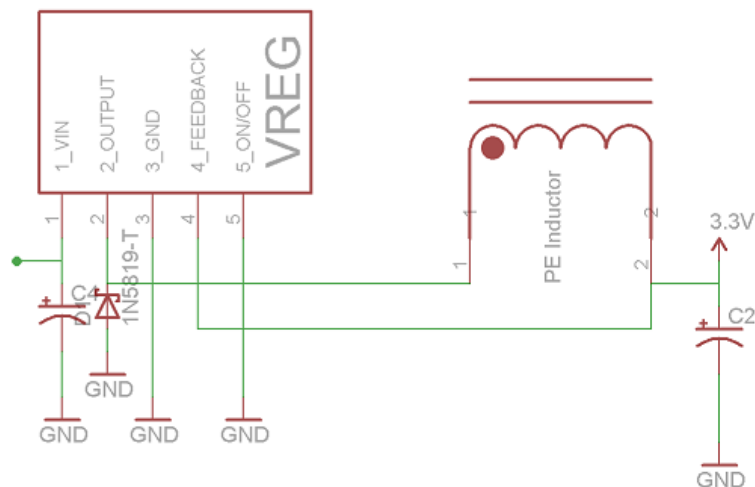


We decided to use a 330uH inductor on the output, along with a Schottky diode and a 10uF capacitor. We used another 10uF capacitor on the input of the device. This voltage regulator was capable of limiting voltages as high as 40V, and was able to supply 3A of consistent output current if necessary.

3.2.5.3 3.3V Regulation

The LD1117 is a great choice for 3.3V regulation as it has the increased available output current of 800mA over the LM2937's 500mA output. The application of this device is very similar to the application of the LM109 used for 5V regulation. The datasheet recommends the use of an input filtering capacitor and an output capacitor to aid in transient response. The increased output current guarantees that all of the devices that will be operating off of the 3.3V will be able to acquire sufficient operating current. Although the LD1117 is marketed as an adjustable regulator, the LD1117V33 is a fixed 3.3V option that removes the need for external components. LD1117V33 comes in a typical 3-pin package that will allow us to easily mount the component to our PCB. Typical output noise is 100uV, which indicates that there will be minimal impact on the operation of our digital devices operating off of this regulator's output. However, for reasons mentioned in 3.2.5.2, we decided to use a switching regulator for this voltage level as well. We chose the TL2575-33 from Texas Instruments. It had a similar layout as the LM2576 discussed in 3.2.5.2, and was capable of consistently providing 1A in output current, which was more than necessary for our 3.3V power line. Figure 3.2.6 shows the topology of the 3.3V switching regulator.

Figure 3.2.6 – 3.3V Regulator Schematic



3.3 Stabilization Platform Design

If we isolate the stabilization platform, the schematic appears as follows in Figure 3.3.1:

[illegible]

The pin connections are also show in Table 3.3.1:

Table 3.3.1 – Stabilization Platform Connections Listing

From	To
MPU-6050 Pin 10 (VDD)	+3.3V
MPU-6050 Pin 4 (VIO)	+3.3V
MPU-6050 Pin 24 (SDA)	ATmega328 Pin 27 (A4)
MPU-6050 Pin 9 (GND)	Ground
MPU-6050 Pin 23 (SCL)	ATmega328 Pin 28 (A5)
ATmega328 Pin 7 (VCC)	+5V
ATmega328 Pin 20 (AVCC)	+5V
ATmega328 Pin 8 (GND)	Ground
ATmega328 Pin 22 (GND)	Ground
ATmega328 Pin 16 (B2)	Roll Servo Signal Wire
ATmega328 Pin 15 (B1)	Pitch Servo Signal Wire

Pins 15 and 16 on the ATmega328 were utilized by the `writeMicroseconds()` Arduino command in the Servo library. We needed to be able to determine the current position of the servo motors by utilizing the `read()` command. The MPU-6050 was powered by a +3.3V power source, which was supplied through power regulation from our battery, and the ATmega328 was powered with +5V. The MPU-6050 provided its information through the SDA and SCL pins, which the ATmega328 processed using the Arduino I2C library. The Hitec HS-422 servo motors were powered with +5V in order to keep voltage regulation board space to a minimum and reduce costs. These servos also connected to the board via 3-pin headers rated to handle the maximum PWM output from the

ATmega328. In order to test this system, we designed the PCB to have a 10-pin header to allow for a connection to an MPU-6050 breakout board.

The mechanical portion of the stabilization arm was built using an assembly of brackets and horns designed specifically for our HS-422 servo motors. The weight of the mechanical portion was of high interest. The lighter our mechanical assembly was, the faster our servos will be able to turn and stabilize the platform, and the less wear that would occur on our motors. Because of this, aluminum was the ideal choice for the material the brackets and horns were made out of. Images Scientific Instruments and RobotShop both offer multiple packages containing a different set of brackets and servo horns specific to different applications, and can be used to create our roll and pitch platform. We chose a set of servo brackets called Servoblocks to directly couple our servo motors together in the 3-axis fashion required.

3.4 Navigation

3.4.1 Autonomous navigation mode

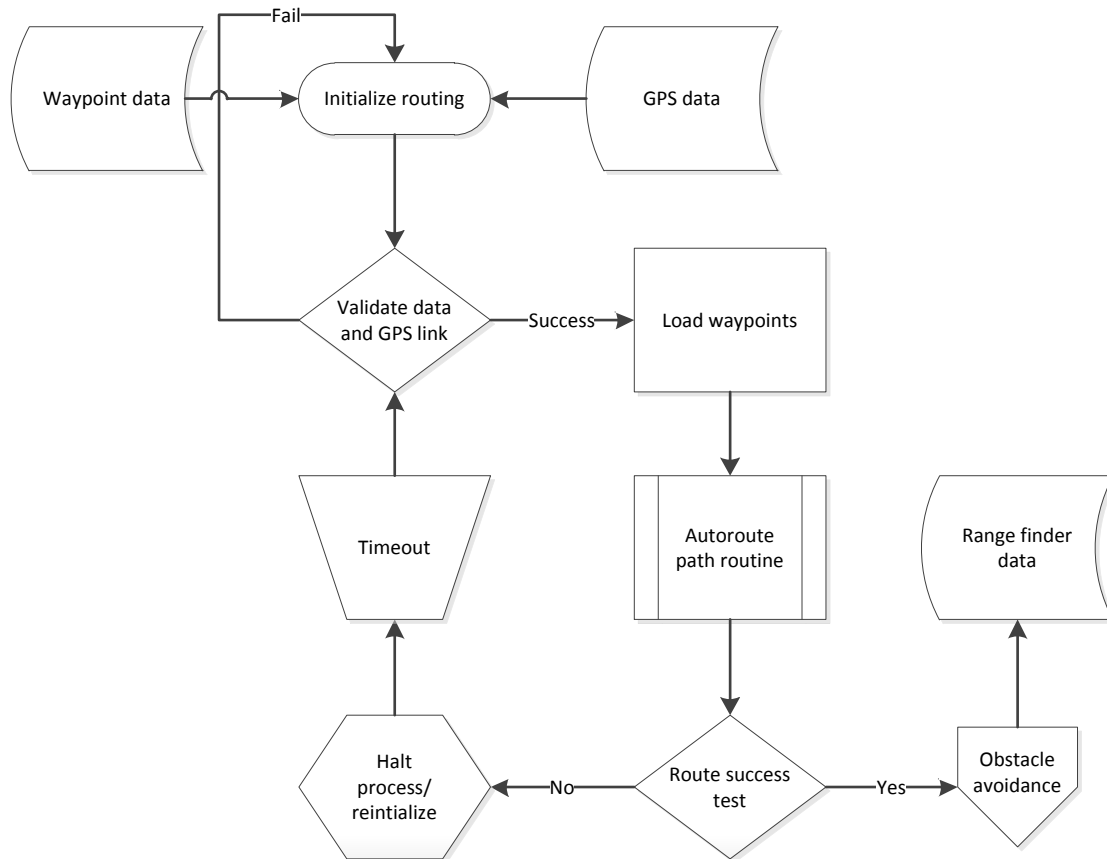
The navigation done in this mode is controlled directly by the software without any assistance from the user. The path is loaded from the user by waypoints and the path is calculated by the auto-routing function. The navigation mode then takes control of the motors and sends the appropriate commands for navigating along the path returned from the auto-router. Due to time constraints, autonomous navigation modes were not included. However, since this is all software based, it could still be implemented if desired.

3.4.2 Auto-routing

The auto-routing function allows the S.H.A.S.bot to route itself along a given perimeter or path defined from the user or by using other modes such as the waypoint mode. The general idea behind the auto routing mode is to assist the user in area coverage.

The diagram below shows the data fusion from the waypoint data, which is an input, and the GPS data from the Venus module. After the link and waypoint data are validated. The auto route routine will take place to determine the best route to take between waypoints. This can either be interpolated between points or can be set manually by the user. The flow chart in Figure 3.4.1 shown below illustrates the general techniques and logic behind the navigation auto-routing mode.

Figure 3.4.1 – Auto-Routing Flowchart



3.5 Communication Systems

The following will outline the design specifications for the wired and wireless communications systems for the surveillance vehicle. After consideration of different RF devices, the XBee-Pro RF module will be used for its low power consumption and extensive range. For wired connections, the system consists of the embedded-side communications between the wireless XBee module and the host microcontroller and the PC-side communications between the other XBee module and the computer. For wireless connections, the system consists of the RF communications between the two XBee modules as well as the 802.11 Wi-Fi communications between the IP web camera.

3.5.1 XBee-Pro Wireless Module

Information essential to the integration of the XBee RF module into our communications system is outlined in the following sections. This includes pin assignments, DC electrical characteristics, and modes of operation.

3.5.1.1 Pin Assignments

The following Table 3.5.1 outlines the useful pin signals for the XBee-Pro module. [17]

Table 3.5.1 – XBee-Pro Module Pin Configuration

Pin Number	Name	Direction	Description
1	Vcc	-	Power Supply
2	Dout	Output	UART Data Out
3	Din/ $\overline{\text{CONFIG}}$	Input	UART Data In
4	DIO8	Either	Digital I/O 8
5	$\overline{\text{RESET}}$	Input	Module Reset
6	PWM0/RSSI	Output	PWM Output 0 or RX Signal Strength Indicator
7	PWM 1	Output	PWM Output 1
8	Reserved	-	Do not Connect
9	$\overline{\text{DTR}}$ / $\overline{\text{SLEEP_RQ}}$ /DI8	Input	Pin Sleep Control or Digital Input 8
10	GND	-	Ground
11	DIO4	Either	Digital I/O 4
12	$\overline{\text{CTS}}$ / DIO7	Either	Clear To Send or Digital I/O 7
13	ON/ $\overline{\text{SLEEP}}$	Output	Module Status Indicator
14	Reserved	-	Do not Connect
15	Associate / DIO5	Either	Associated Indicator, Digital I/O 7
16	$\overline{\text{RTS}}$ / DIO6	Either	Request To Send, Digital I/O 6
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0	Either	Analog Input 0 or Digital I/O 0

It is important to note that the minimum connections required for wireless communication are Vcc, GND, Dout, and Din. A 50k ohm pull up resistor comes attached to $\overline{\text{RESET}}$. Pins that are not used are left disconnected. [17]

3.5.1.2 DC Characteristics

The following Table 3.5.2 includes the DC characteristics and requirements for the Xbee Pro Module. [17] These electrical characteristics are important and must be taken into account in order to keep the wireless module as well any other devices from being damaged.

Table 3.5.2 – Xbee-Pro 2.5 Module Electrical Characteristics

Symbol	Parameter	Condition	Min	Typical	Max	Units
V_{IL}	Input Low Voltage	All Digital Inputs	-	-	$0.2 \cdot V_{CC}$	V
V_{IH}	Input High Voltage	All Digital Inputs	$0.8 \cdot V_{CC}$	-	$0.18 \cdot V_{CC}$	V
V_{OL}	Output Low Voltage	$I_{OL} = 2 \text{ mA}$, $V_{CC} \geq 2.7 \text{ V}$	-	-	0.5	V
V_{OH}	Output High Voltage	$I_{OH} = -2 \text{ mA}$, $V_{CC} \geq 2.7 \text{ V}$	$0.82 \cdot V_{CC}$	-	-	V
I_{IN}	Input Leakage Current	$V_{IN} = V_{CC}$ or GND, all inputs, per pin	-	-	0.5	μA
TX	Transmit Current	$V_{CC} = 3.3 \text{ V}$	-	215	-	mA
RX	Receive Current	$V_{CC} = 3.3 \text{ V}$	-	55	-	mA
PWR-DWN	Power-Down Current	SM Parameter = 1	-	<10	-	μA

3.5.1.3 Operational Modes

The XBee wireless communications module operates in five different modes. These include Idle mode, Transmit mode, Receive mode, Sleep mode, and Command mode. These modes and their specified conditions are as follows:

- Idle Mode: Default, low-power state, active when there is no condition
- Transmit Mode: Entered when serial data is received from host device
- Receive Mode: Entered when valid RF Data is received from RF device
- Sleep Mode: Entered when sleep pin is active
- Command Mode: Entered when Command Mode Sequence is received

3.5.1.3.1 Idle Mode

When the module is not in Transmit, Receive, Command, or Sleep mode, it remains in Idle mode. It shifts into other modes of operation depending on a specified condition variable.

3.5.1.3.2 Transmit Mode

In Transmit mode, the module receives serial data from its connected host device. It then transmits the data across a wireless channel to a receiving RF device.

3.5.1.3.3 Receive Mode

In Receive mode, the module receives valid RF data from another transmitting RF device and relays it serially to a connected host device.

3.5.1.3.4 Sleep Mode

In Sleep mode, the module enters one of three states of low power consumption based on its SM (Sleep Mode) variable. These include Hibernate, Doze, or Cyclic Sleep.

- Hibernate (SM = 1): Characterized by having the lowest power consumption of the three sleep states. Its on/off is determined by the host device via the Sleep_RQ pin (pin 9).
- Doze (SM = 2): Characterized by having the fastest wake-up time of the three sleep states. Its on/off is also determined by the host device by the Sleep_RQ pin.
- Cyclic Sleep (SM = 4-5): Characterized by periodic sleep intervals where it will check for RF data. Its on/off is pre-programmed by its ST(Sleep Time) parameter.

Leaving SM = 0 will cause the device to remain in Idle mode. In this mode it will always be ready to receive and transmit RF data.

3.5.1.3.5 Command Mode

In Command mode, the XBee module's operational parameters can be changed. When in Command mode, the device accepts incoming characters as commands. To transition into AT Command mode, simply input the character sequence "+++". Once in Command mode, commands can be sent in the following format:

ATxx yy<CR>

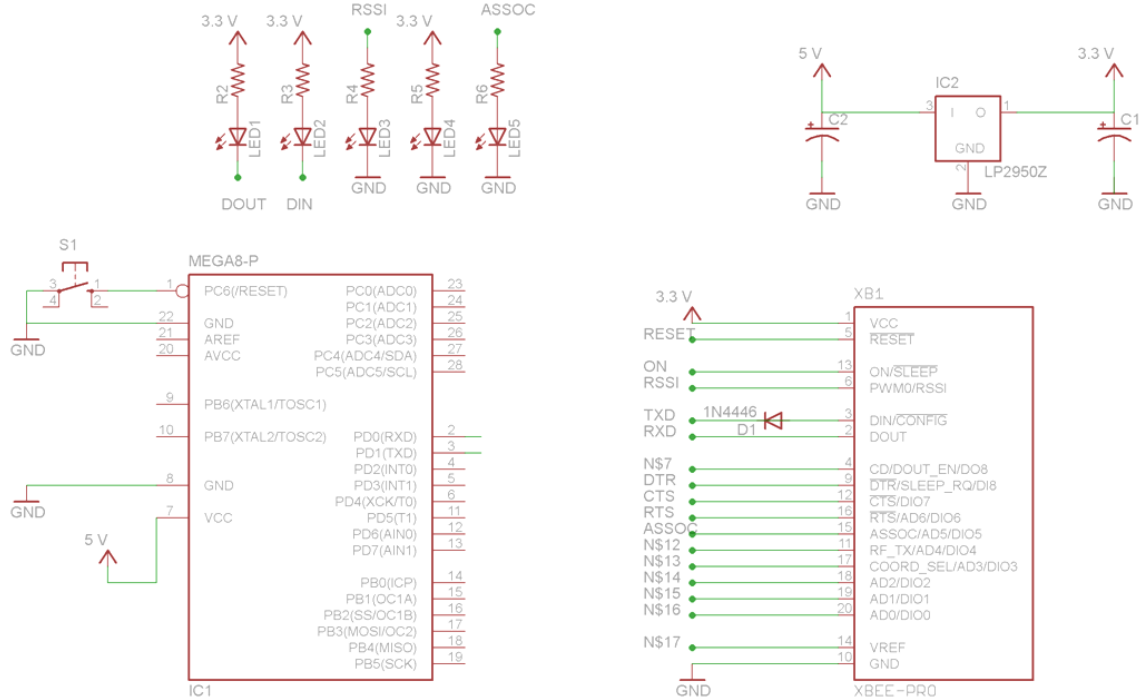
Where AT is the required prefix, xx is the parameter to be changed (in ASCII), yy is the new value (in HEX), and <CR> is a carriage return. For example, ATDL 1F<CR> changes the Low bytes of the Destination Address to 1F. The command "ATWR" writes a previous command into non-volatile memory and is necessary when configuring the RF device. To exit Command mode, enter the command "ATCN". A table of all AT Commands used by the XBee is supplied in the XBee ZNet 2.5 manual. [17]

3.5.2 Embedded-side Communications

The embedded-side communications system consists of a Wireless XBee Pro module and the chosen microcontroller, which is an Atmega328 MCU. The Atmega328 has a built in, logic level 3.3 V UART port which is connected to the D_{in} and D_{out} of the XBee module. These are pins 3 (D_{in}) and 4 (D_{out}) on the XBee module. The module is very specific about its operating electrical characteristics. The XBee Pro requires a transmit current of 270 mA at 3.3 V and a receive current of 55 mA at 3.3 V. It also requires a V_{cc} between 2.8 and 3.3 V. A 3.3 V voltage regulator will be used to achieve this input level. The following schematic in Figure 3.5.1 outlines the embedded-side communications circuit. An interface circuit will be used to connect the XBee module to the host microcontroller. It

includes the built-in voltage regulation as well as breakouts for all of the XBee module's pins. Filter capacitors for the regulator outputs as well as resistors to regulate the current flow on the data in and out pins are built in as well.

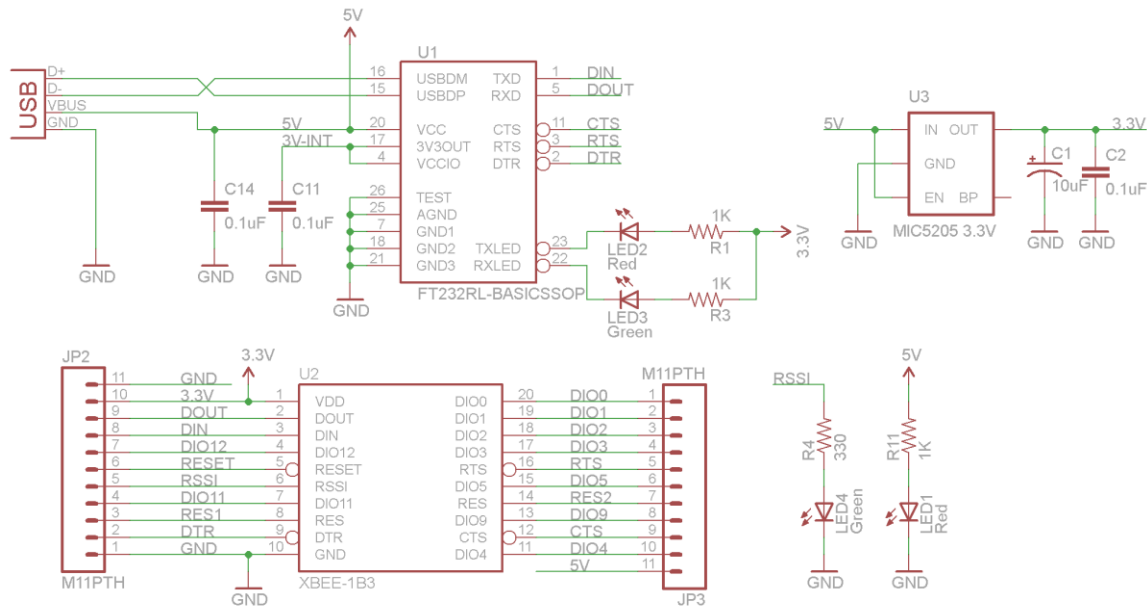
Figure 3.5.1 – Embedded-side XBee Interface Schematic



3.5.3 PC-side Communications

The PC-side communications system is almost identical to the Embedded-side. It consists of a wireless XBee Pro module and a USB-to-Serial interface. The USB-to-Serial module contains a header socket for the XBee module, a built-in 3.3V regulator with output filter capacitors, an FT232RL UART for converting from USB to serial, a USB hub, and LED indicators to indicate data transfer and power. Resistors to regulate the current flow in the data in and out pins are also included. The following schematic in Figure 3.5.2 outlines the PC-side communications circuit. The operation of the module on the PC-side is almost identical to the embedded-side. The only difference is that they use different UARTs as the FT232R ICs are designed for USB to serial conversion.

Figure 3.5.2 – USB to Serial Schematic

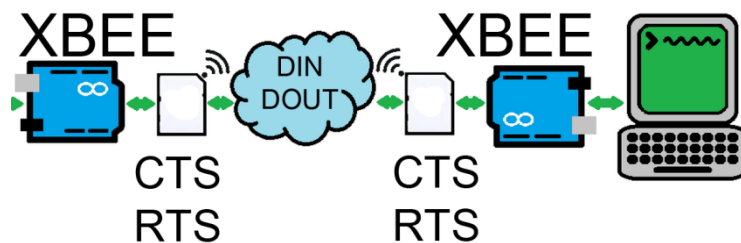


Used Under Creative Commons Attribution Share-Alike 3.0 License by N. Seidle at Sparkfun [18]

3.5.4 Wireless Communications System

The full wireless control system consists of two XBee Pro wireless modules interfaced between the vehicle-side microcontroller and the PC, respectively. Its basic functionality is outlined in the data flow chart below. Data in and data out lines are regulated to 3.3 V. The flow control lines CTS (Clear To Send) and RTS (Request To Send) may or may not be used depending on whether or not I/O buffers are overflowing and data is being lost. The following signal flow diagram in Figure 3.5.3 outlines the system functionality.

Figure 3.5.3 – Wireless Communications Flow Chart



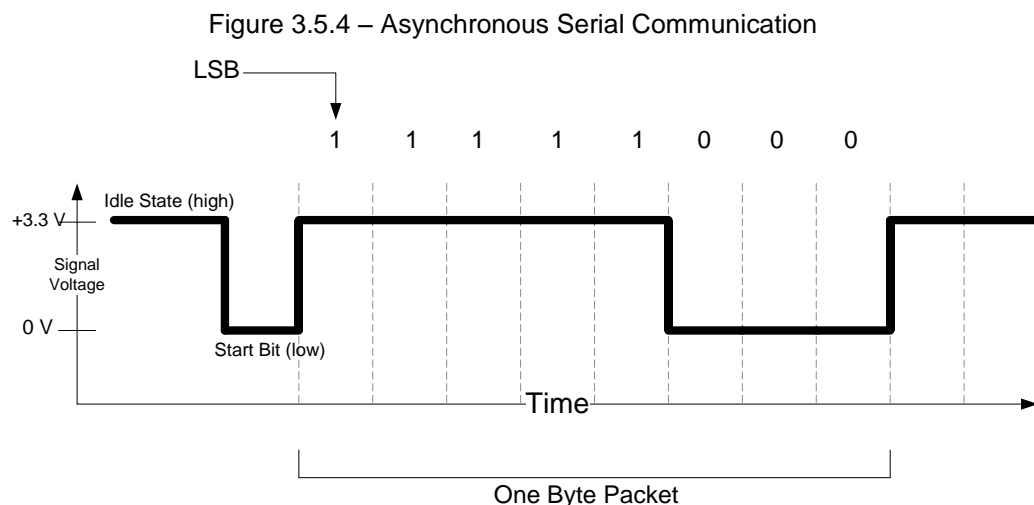
Used under Creative Commons license

Ideally, the RTS and CTS lines would be excluded, as to save pin space on the microcontroller. To do this, it is necessary to send messages that are smaller than 34 bytes (the size of the I/O Buffers on the XBee Modules). It also may be required that the interface baud rates be lower than that of the XBee to compensate for slower throughput data rates. The surveillance vehicle will be

sending sensor data to the PC for monitoring and it will be receiving control data from the PC for vehicle movement. This creates a situation where data might be trying to transmit from one serial interface to one XBee Module while it is receiving RF data from the other XBee Module. Because data can only be transmitted or received over the RF channel at one time, the incoming serial data will be stored in the Data Input buffer to send once the RF receive process is finished. This creates an opportunity for possible overflow if the Input buffer fills up before the RF receive process is done and it can transmit the serial data in the Input buffer. [17]

3.5.4.1 Operation

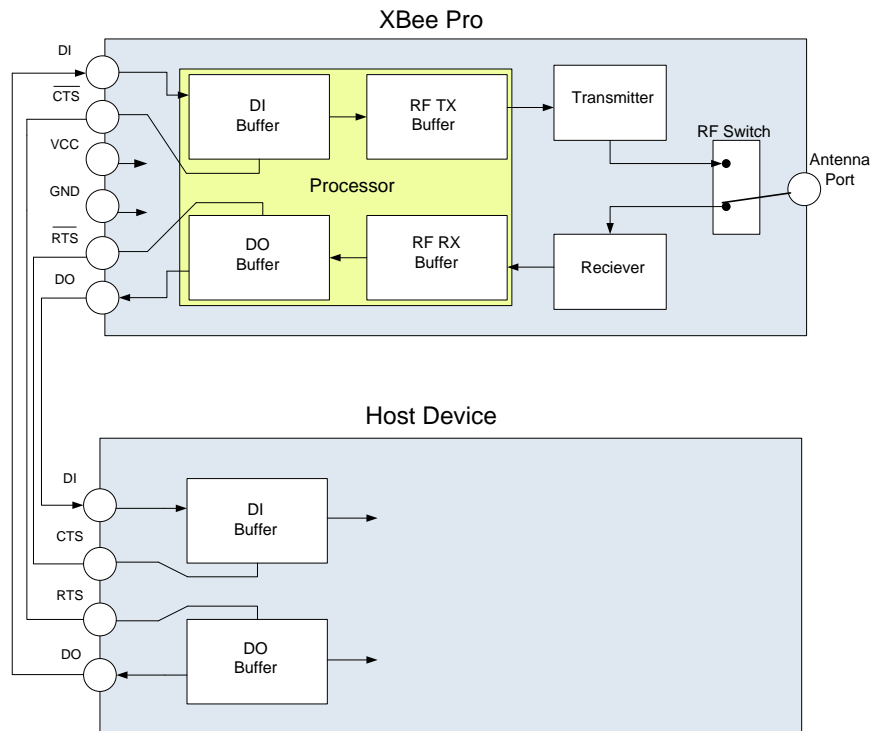
The XBee Module sends data to and from a microcontroller as an asynchronous signal at a 3.3V logic level. This means that it is compatible with any UART with the same electrical characteristics. Once the XBee module is correctly connected to a compatible UART, the data lines should be high when not in use. When data is about to be transferred, the signal is set to low to indicate a start bit. The receiving end recognizes this and begins accepting data bits. The baud rate determines the time intervals during which the UART checks for a high or low signal and processes the bits accordingly. Once the UART has experienced a full time period it checks for one or more stop bits to verify that the particular data packet has been sent. The least significant bit is always sent first, right after the start bit. The UART will always handle timing and error checking. The following graph in Figure 3.5.4 shows an example transmission of the single data packet '0x1F' hex (= '31' decimal = '00011111 binary) from the XBee Module Pin 4 (Dout) to a compatible UART. The UART converts the serial bits into a parallel hex or decimal value. Always ensure that the baud rate for the microcontroller or PC is set to that of the XBee module (default BD parameter = 3; 9600bps). [17]



3.5.4.2 Data Flow Control

For the wireless communications system between the PC and Microcontroller to function correctly, data flow control must be handled correctly or buffer overflow will occur, causing any additional packets of data that are sent to be lost. The data flow control system consists of input and output buffers for temporarily storing data until it is ready to be processed and transmitted between the XBee module and either the host device or another RF module. The following signal flow diagram in Figure 3.5.5 outlines the connections between the XBee module and a host device.

Figure 3.5.5 – XBee Module Signal Flow Control Diagram



Data enters the Data Input Buffer through DI (pin 3), where it is stored until it can be processed and transmitted. The $\overline{\text{CTS}}$ line is set to low by the module when the DI Buffer is 17 bytes away from being full. It signals the connected device to stop sending data until the buffer has 34 bytes of space open.

RF data enters the Data Output Buffer until it is processed and sent through DO (pin 4) to the connected device. If the DO buffer is full and RF data is still streaming to the device, that data will be lost. The $\overline{\text{RTS}}$ line is set to low by the receiving (host) device to signal the DO Buffer to stop sending data.

In typical serial communications systems, the CTS line of one device is connected to the RTS line of the other device and vice versa. The flow control lines CTS and RTS are not always required for successful data transmission. But

they can be very useful when trying to achieve the fastest and most efficient communication speeds possible. [17]

3.5.4.3 Set Up and Configuration

Once a serial connection between the XBee and a PC is established, the XBee's operational parameters can be tailored to function specifically for the surveillance vehicle. In order to program and update the firmware of the XBee, software called X-CTU by MaxStream was needed. The following steps outline the preparations needed to begin configuring the XBee:

1. Install X-CTU software by MaxStream.
2. Connect the XBee module to the serial-to-USB interface board.
3. Launch X-CTU and, via the "PC Settings" tab, set the baud and parity settings of the desired Com Port to match those of the XBee. The default baud setting is BD = 3, which is 9600 bps.
4. Download and upgrade the firmware of the XBee in the Modem Configuration Tab
5. Enter Command mode in the Terminal tab and change desired parameters of the module

A highly detailed procedure of configuration and testing of the wireless network is included in the Testing section.

3.5.5 Camera

The camera chosen to be used provided the best performance and versatility for a reasonable price of \$100.00. The Cisco Wireless-N camera provided an ad-hoc connection with the user's PC and could stream the data in various qualities and frame rates, allowing for mitigation of performance issues.

The resolution can range from 160x20 to 640x480, which provided a wide range of choices for the performance and processing techniques. The quality level, or compression level, could also be varied and controlled. This affects the bandwidth required and may allow for faster processing if the image being processed is reduced in total size.

3.5.6 Networking and security

3.5.6.1 IP/networking configuration

Since the camera being used is an IP camera, a networking interface and connection must be established between the image processing platform and camera. This could be done through an infrastructure or ad-hoc connection.

The advantage of using the infrastructure mode is it can be used with many established network setups without a lot of network configuration. The downside

to this option is the overhead and traffic increase over the local area network. Every router is different and there is no general solution for every network.

Using the Ad-hoc mode provided by the camera allows a PC to connect to the camera directly and manage the stream without any external hardware. This solution seems like the best option, as reducing propagation times and external variables was likely to reduce the total processing time as well as reduce and potential problems and bugs with the software.

3.5.6.2 Security and jamming mitigation

An advantage of using a secured Wi-Fi network setup was its inherent ability to resist and mitigate jamming and interference. The natural RF noise that can be seen at any time will not significantly affect the performance of the video link between the camera and the PC.

The GPS module also used hardware and firmware based jamming resistance. This is for a civilian module. Even with substantial amounts of RF noise present from other machines in operation, there should not be a noticeable level of performance decrease by using the devices indoors or around other electronic equipment.

3.5.6.3 Data logging

Data logging was a feature that can be provided by the GPS module and as a general software feature for debugging. The data log could be stored locally on the included flash module. It was logical to store the IP data PC side since that information is being processed and created on the PC. Some data from the hardware, such as GPS, can be stored on a memory chip provided on the PCB, which is directly connected to the GPS using a serial interface. The GPS would handle all the required techniques for logging the data, the user needs to only enable the data logging feature.

The main advantage for data logging is reserved for its application in development and debugging, which the team will be utilizing. It can be invaluable information in figuring out these bugs and better the understanding of the software and hardware flow.

3.6 Software

3.6.1 Graphical User Interface and PC-Side Control

3.6.1.1 Graphical User Interface

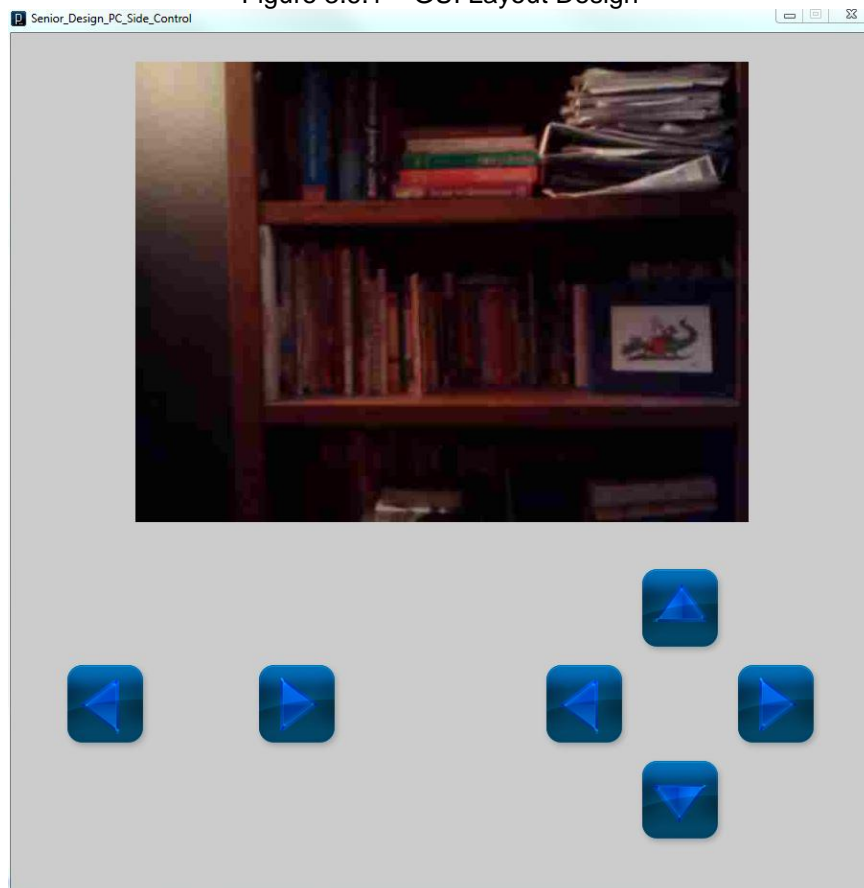
After consideration of GUI designs in Processing, Java, and C++, Processing was used due to its simplicity and direct compatibility with the control code for the surveillance vehicle. The GUI utilizes the native Processing API as well the

controlp5 library to create an accessible, user-friendly interface. The GUI is in the form of an interactive Heads-Up Display (HUD) and has the following functionality available for the user:

- A. 1 – Central high-resolution video display that gives the user real-time vision of the vehicle's surroundings via its mounted, stabilized camera.
- B. 2 – Hot-keyed virtual directional buttons to control the camera's panning (left/right) movement.
- C. 4 – Hot-keyed virtual directional buttons to control the vehicle's movement. Up and down arrows causes the vehicle to travel forward or in reverse. When the up and down arrows are not being pressed, power is not delivered to the wheels. Left and right arrows control the steering servo to cause the vehicle to turn right or left while moving. When the left and right arrows are not being pressed, the steering servo returns to a user-specified central position.

All of these controls and display features are organized in an easily-accessible format for user. The diagram below in Figure 3.6.1 shows the desired layout design of the graphical user interface.

Figure 3.6.1 – GUI Layout Design



For the functions above to work correctly, they are integrated into the final PC-side control code using control events, Processing's version of action listeners/events.

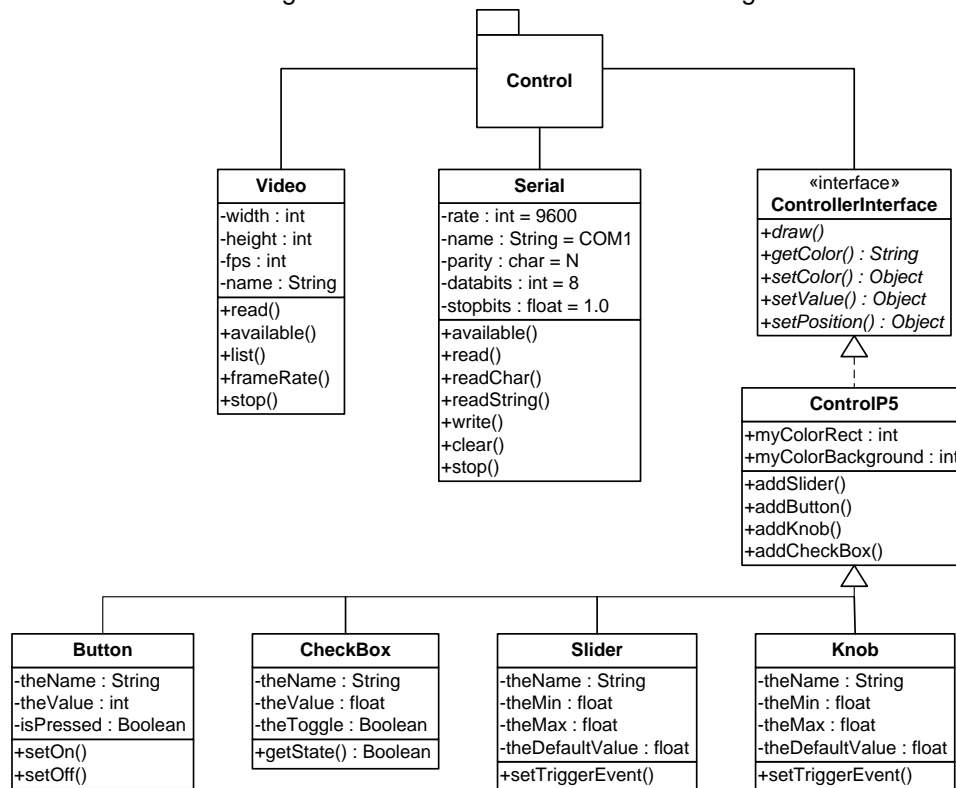
3.6.1.2 PC-Side Control Software

The full PC-side control software is outlined in Figure 3.6.1 and consists of the following libraries and functionality:

- Controlp5 library for GUI design and implementation.
- Video library for video feed image.
- Processing.serial library for serial communications between the Coordinator XBee module and the PC. The PC-side Coordinator XBee module is able to communicate with the embedded-side Router XBee to obtain sensor data and other feedback.

The following UML diagram in Figure 3.6.2 outlines the API of the control code. The entire package was exported as an executable file so that it is usable on other systems.

Figure 3.6.2 – Control Software UML Diagram



A sketch written in Processing has two major methods. The setup() method, and the draw() method.

The setup() Method

The majority of the GUI/HUD layout code as well as the Serial I/O setup code is written in the setup() method. Elements such as buttons and sliders are created using a particular add method. The parameters of an elements add method determine its labels, positioning, and size. Setter methods such as setColor can be used to change other features of a particular element. The setup() method only runs once and prepares the program for the control loop.

The draw() Method

The control code that links to the GUI elements created in the setup method is written in the draw() method. The draw() method is basically the main control loop for the entire system. It continuously checks input variables from the PC system such as ASCII data from a key press. It also reads and check input variables from the vehicle's embedded system such as an IMU sensor data packet. When the control loop receives data from one of these sources it will respond accordingly. The read and write methods are used to send numerical data to and from the PC and the embedded system. PC read/write commands work by first identifying a valid COM port, sending or receiving the data serially to or from that port, and transmitting/receiving it across a wireless network. The PC or embedded device will then decide what action to take based on the transmitted data.

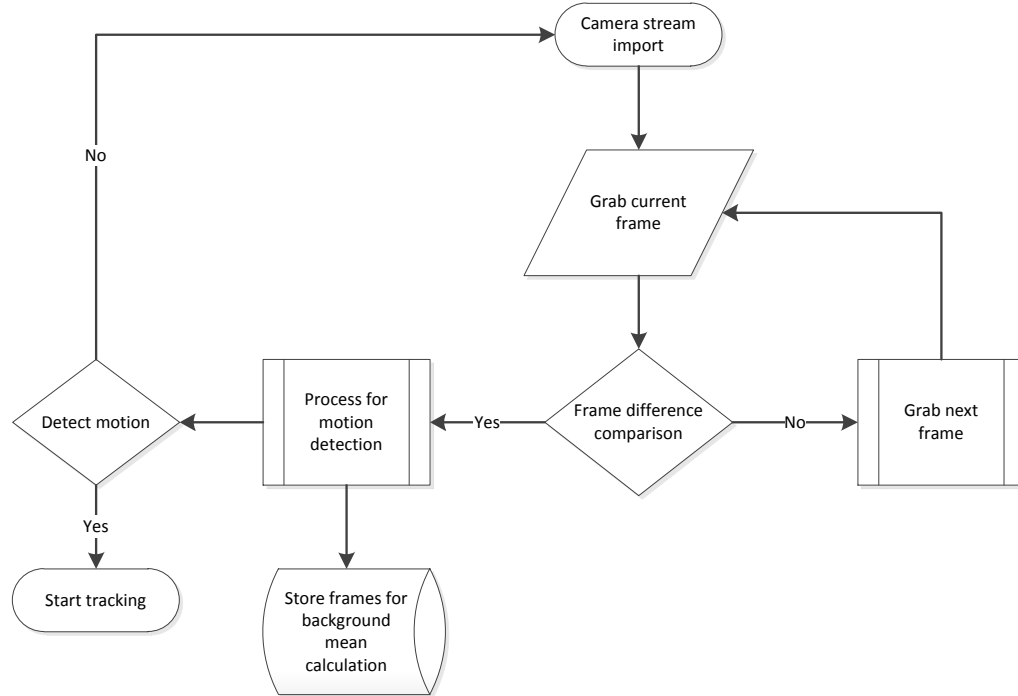
3.6.2 Image Processing

3.6.2.1 OpenCV

OpenCV provides many open source algorithms available for use. The mean shift calculations, motion detection, and blob detection used will all be using libraries from OpenCV. This allows the development team to focus on important aspects of tuning these algorithms for use with the wireless camera and for the application at hand. The data fusion and tracking methods behind these algorithms would be external to OpenCV and would provide a useful application for these image processing techniques. The use of image processing in the final design was not used, but similarly to the tracking and auto-routing features, it can still be programmed and implemented on the PC side and used with our final design.

The image processing libraries from OpenCV would not be enough for the end result, however, they will provide the main core algorithms for use in providing the information needed to create these tracks. The following Figure 3.6.1 outlines the functionality of the OpenCV tracking software.

Figure 3.6.1 – General Image Processing Software Overview



3.6.2.2 Mean-shift based tracking

The mean-shift based tracking algorithm initializes the detection on a mean value for the background image. This is saved in the memory and any subsequent frames from the video are compared to this image. Pixel values are subtracted and if they will form what's called a foreground mask. This is created by a pixel by pixel subtraction and a given threshold. Table 3.6.1 shows the I/O data structure for mean-shift calculation.

Table 3.6.1 – Input and Output Data Structure for Mean Shift Calculation

Inputs	Outputs
Video frames	Foreground mask
Threshold	Background mask
Tuning specifications	Video out with applied mask

3.6.2.3 Motion detection

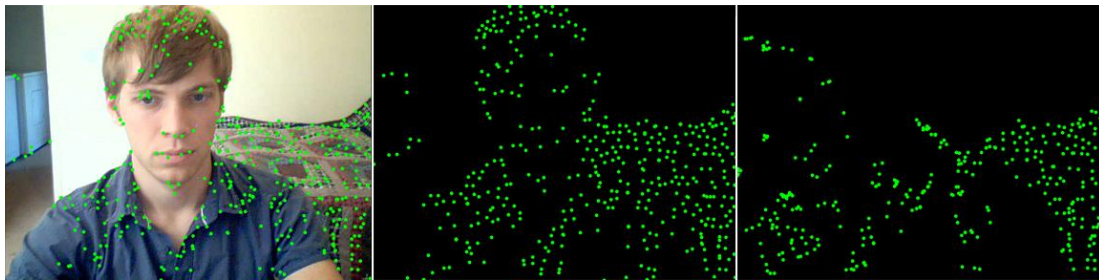
The motion detection algorithms included with OpenCV provide the software with a garden variety of basic motion detection schemes. These were enough for what the software needs to accomplish in terms of complexity, since there wouldn't be any complex motion detection when the S.H.A.S.bot is moving.

The Lukas-Kanade optical flow algorithm could be implemented fairly easily using OpenCV's libraries. The basic idea behind this algorithm is follows a basic procedure:

1. Identify unique points in the image based on pixel combination and local neighborhood features of the pixel.
2. Points are saved and the optical flow equations are solved for pixels in the given neighborhood by the least squares criterion.
3. The solution provides an optical flow, which can show the motion flow of the image that is less susceptible to noise than point-wise methods [3].

An example of the tracking procedure can be seen below in figure. The dots represent the unique points chosen by the optical flow algorithms. The middle picture shows only the points and the final picture on the right shows the man turning his head to his right, with the points following the optical flow of the video.

Figure 3.6-1 – Lukas-Kanade method of optical flow tracking



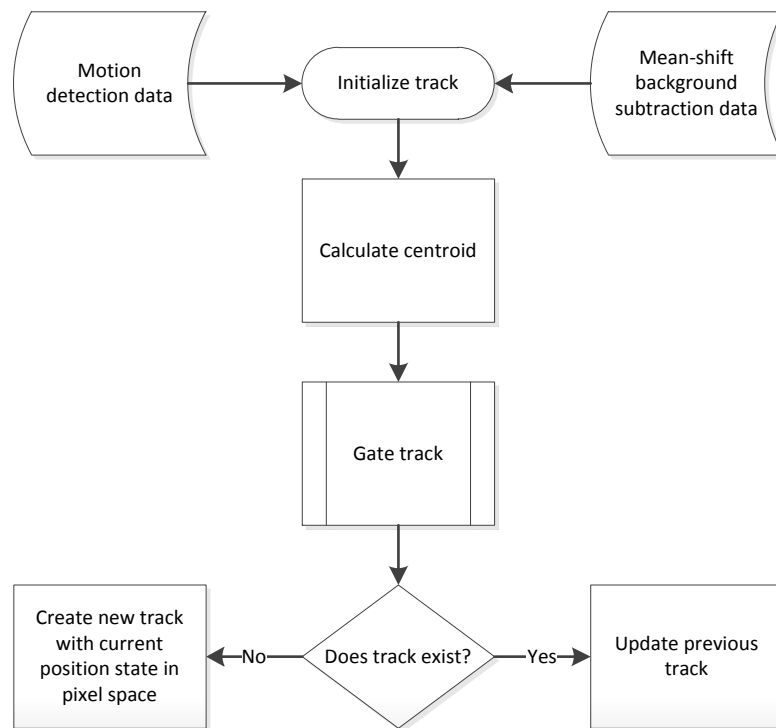
3.6.2.4 Tracking

The tracking software would be implemented with a combination of the libraries from OpenCV along with some homebrew software written in C and C++. The main objective of the tracking software was to use the image processing motion detection algorithms and tracking algorithms discussed in the research section to produce a usable and informational output to the user. In this particular case it will be a track that displays to the user of where the motion is detected and its predicted motion.

The mean-shift detection algorithm uses the HSV conversion technique combined with an image scan to detect, group, and contours the resultant. This output was an enclosed shape that is defined by this algorithm, which can be interpreted and passed on as a track.

The fusion of this data and the motion detection data would provide for enough information for the tracking software to establish the current state of position and velocity and predict the next state $t+1$.

Figure 3.6.2 - Track software routine



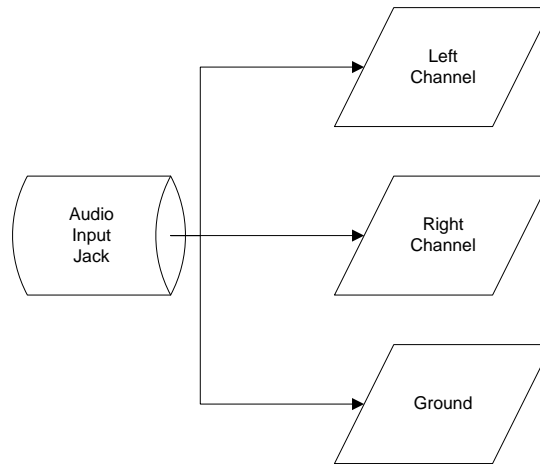
We began testing of the image processing and tracking portion of the project and decided to leave it as a separate expandable feature for later development. We wanted to focus on creating a solid foundation for the vehicle and wanted to have the image processing as an optional portion to sit on top of the video feed, which is still possible.

3.7 Audio

3.7.1 Microphone Connection to Audio Board

It is important to understand how the microphones we use connected to our audio board before we start designing the audio board itself. The microphone cartridges that used had XLR plugs, therefore we needed to have XLR connection sockets to mounted on the board. The design of the microphone plug was as follows, seen in Figure 3.7.1.

Figure 3.7.1 – Audio input jack breakdown



There are three incoming signals: left channel, right channel, and ground. Since we are not interested in a stereo signal, but a mono rather, we only needed the left channel signal. The right channel was run to ground.

The most useful mount for the XLR socket in our project is a simple solder. The design had a socket for the plug and three prongs corresponding to the three signals (L, R, Ground). We needed to create soldering terminals on the board to connect the sockets; one terminal to the audio chain and one terminal to ground.

The reason we wanted to be able to connect the microphones this way was because we wanted to easily be able to disconnect the microphones for any kind of maintenance they may require. If we were to hardwire them to the board, it would make it very difficult to fix any problems with the microphones, or to be able to replace a microphone completely.

3.7.2 Audio GUI Display

When the user is controlling the vehicle, they needed to be able to see what is going on audio-wise; they needed to be alerted if a sound is detected by one or more of the two microphones that the vehicle is equipped with. But the user also needed to be able to see some sort of representation of how the microphones are arranged on the vehicle so that they will know where the sound came from in reference to the direction the vehicle is facing. We originally wanted to display a bar on each side of the video feed in order to alert the user to sounds from that location. We were unable to implement this at time of presentation due to unforeseen circumstances and time constraints, but we indicated audio detection with blinking LEDs, which verified the functionality of the audio detection circuit.

3.7.3 Defining a Critical Audio Event

When our vehicle was out and working, it could be for a number of reasons. Our vehicle could be used for many different reconnaissance applications, but the threshold of sound detection will always be the same. The vehicle will pick up sound in any environment that it encounters, but what level of sound will we consider enough to alert the user of a critical event? The decibel scale is most likely the best way to physically look at a solid reference for which level of sound will relate the most to us. In the following, Table 3.7.1, we can see some solid examples of certain significant decibel levels. [24]

Table 3.7.1 – Decibel Level Examples

10 dB	Rustling of Leaves
20 dB	Human whisper
40 dB	Ambient Noise in a Library
60 dB	Normal Human Conversation
70 dB	Busy Street Traffic
80 dB	Vacuum Cleaner
100 dB	Large Orchestra

We must remember that while our vehicle needed to be able to pick up significant sounds, the vehicle could run onto all sorts of noisy environments. If the environment is noisy enough, it would effectively render our audio detection fairly useless since it would be picking up ambient sound from every direction.

However, even if our vehicle encounters a noisy environment, the issue of our audio detection effectiveness is completely dependent on the decibel level of the ambient noise of the environment, and the decibel level that we choose to be enough to warrant a critical event. So we must first define a level of environmental ambient sound that will be the loudest we will account for on our audio detection system.

As we can see on the decibel level examples table, the decibel level of a normal human conversation is around 60 dB. This is definitely a level of sound that we want our audio system to be able to pick up. We also see that the ambient noise in a library is around 40 dB. We can say that we definitely want our ambient noise level to be above this. So somewhere in between 40 dB and 60 dB was our ambient environmental noise level threshold. We decided on 45 dB.

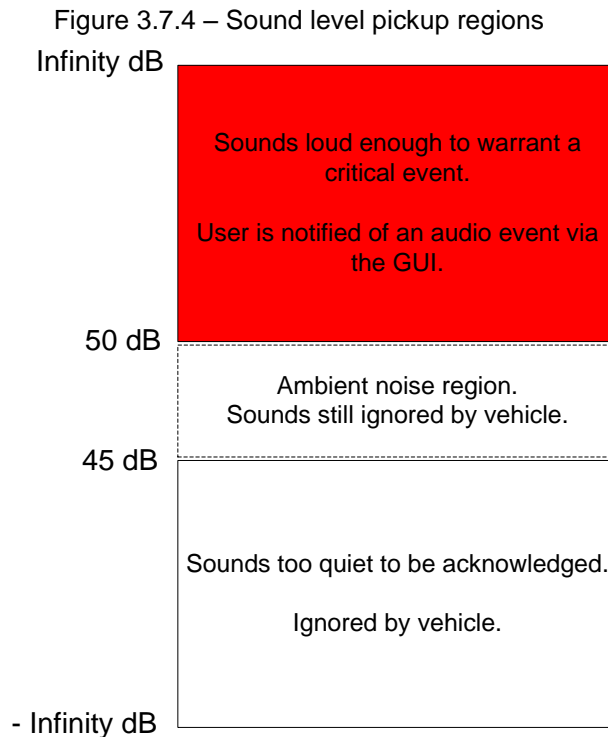
Now that we have determined our ambient environmental noise level threshold to be 45 dB, this also decided that the level of sound gain that would warrant our alert system of a critical event. Keep in mind that we want the vehicles audio system to be able to pick up sounds that are a certain distance away as well. Let us reference the equation 3.7.1 that calculates sound dissipation in a free field. [25]

$$L_2 = L_1 - 20 \cdot \log \left(\frac{r_2}{r_1} \right) \quad (\text{Equation 3.7.1})$$

The first point from the sound source is distance r_1 , the second point from the sound source is r_2 . L_1 is the level of sound that the point at r_1 experiences, and the sound level at r_2 is L_2 .

From this equation we can determine that a doubling of distance from the sound source always produces a difference in sound level of - 6 dB between the points. So let's say that at point r_1 there is a sound heard at 55 dB and that point is 5 meters away from the sound source. If the vehicle is 10 meters away at r_2 , it will experience that same sound at 48.98 dB, so we can say 49 dB. This gives us a very good reference for what the vehicle will be able to detect if we set the threshold of a critical event at around 50 dB. The distance equation shows us in this example that a sound less than a normal human conversation heard 5 meters away from the vehicle with the sound source being 10 meters away from the vehicle, it can still detect this while our critical event threshold remains about 5 dB above our decided ambient sound threshold.

After all of these calculations and hypotheticals, we can safely say that the level of sound our vehicle will pick up to determine a critical event is 50 dB. Below is a diagram, Figure 3.7.4, that illustrates the pickup regions relative to sound level.



3.7.4 Audio Parts Selection

Based on our needs for the audio processing chains, we selected specific IC parts. Our parameters have been discussed in the individual sections for amplifiers, filters, and comparators, but we will go into more detail about the parameter specifics in this section.

3.7.4.1 Amplifier Selection

Our goal was to choose an amplifier that is efficient, low noise, and operates efficiently in the audio bandwidths. We also wanted an amplifier that puts out close to 1.5 W, as explained in the amplifier section. The amplifier we chose is the model LM386. It requires a 5V supply. It is a specialized audio amplifier and will work well for our applications.

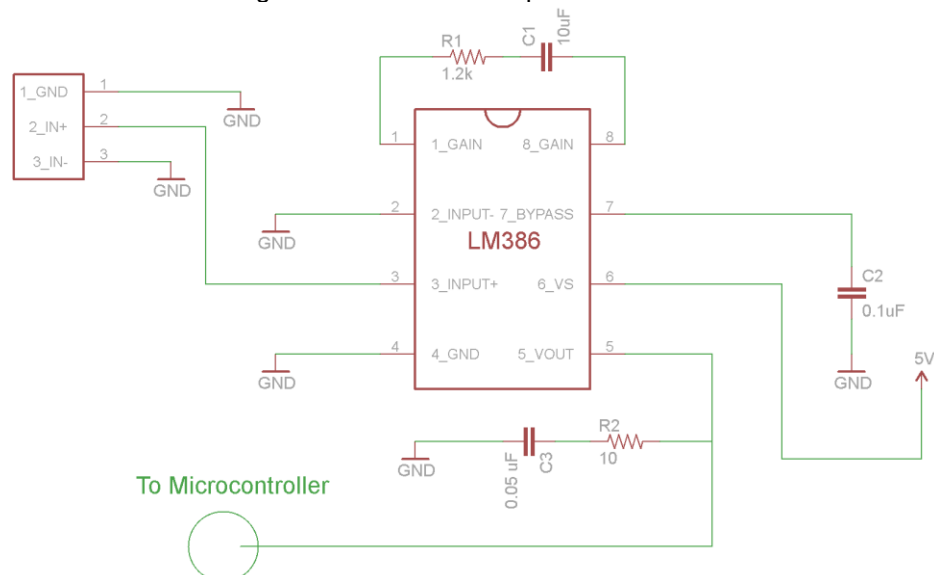
3.7.4.2 Microphone Selection

For our two stereo microphones, we chose the Audio Technica M4000s because they output an ample 5 mW signal max and they have a beefy construction. Also, they have a removable pop filter that allows for even more sensitivity when removed. They boast an XLR output which matches our audio board connection jack.

3.7.5 Audio Board Circuit Schematics

Figure 3.7.5 shows our final audio amplification circuit design schematic.

Figure 3.7.5 – Audio amplification circuit



4 TESTING AND CONFIGURATION

4.1 Communications System

Purpose and objective: The goal of the configuration and testing apparatus is to create and test a fully-functioning long-range network capable of transmitting and receiving control and sensor data. It consists of the following:

Hardware:

- 2 – Series 2 XBee-Pro RF Modules (Digi International P/N: XB24-BCIT-004)
- 1 – XBee Explorer USB (Sparkfun P/N: WRL-08687)
- 1 – Arduino Uno or similar Development Board (Digi-Key P/N: 1050-1024-ND)
- 1 – Arduino XBee Shield (Digi-Key P/N: 1050-1006-ND)
- 2 – LEDs
- A Windows-based PC
- USB mini-B cable
- Breadboard

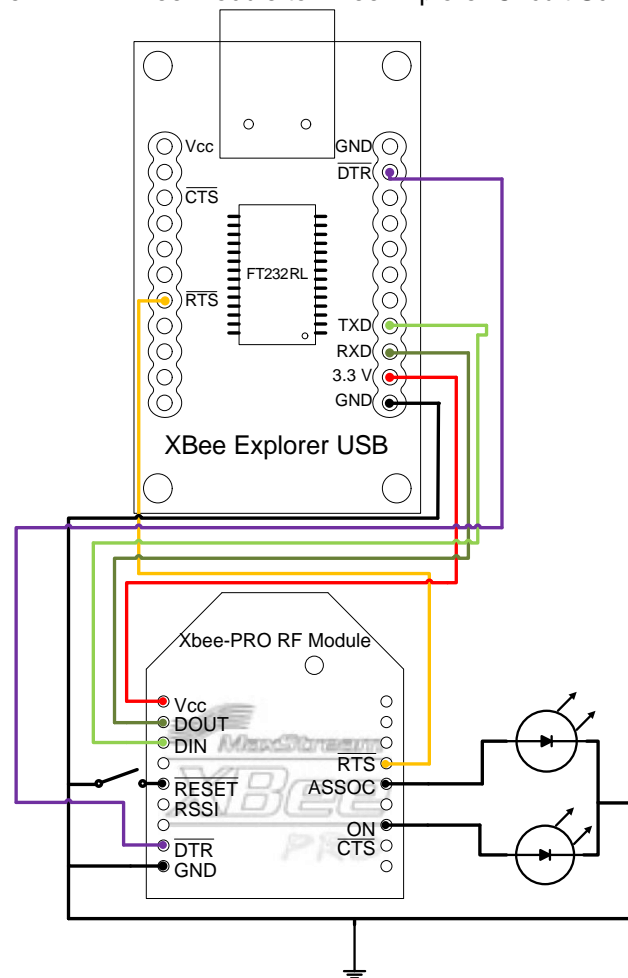
Software:

- X-CTU software by Digi
- Processing software by Ben Fry and Casey Reas

It is composed of two separate circuits. The first is the PC-side circuit that includes the XBee Explorer USB, one XBee RF Module, two indicator LEDs, and a Windows-based PC. The second is the embedded-side circuit which includes the Arduino Uno development board and the other XBee RF module. The following steps explain the preparation, configuration, and testing methods:

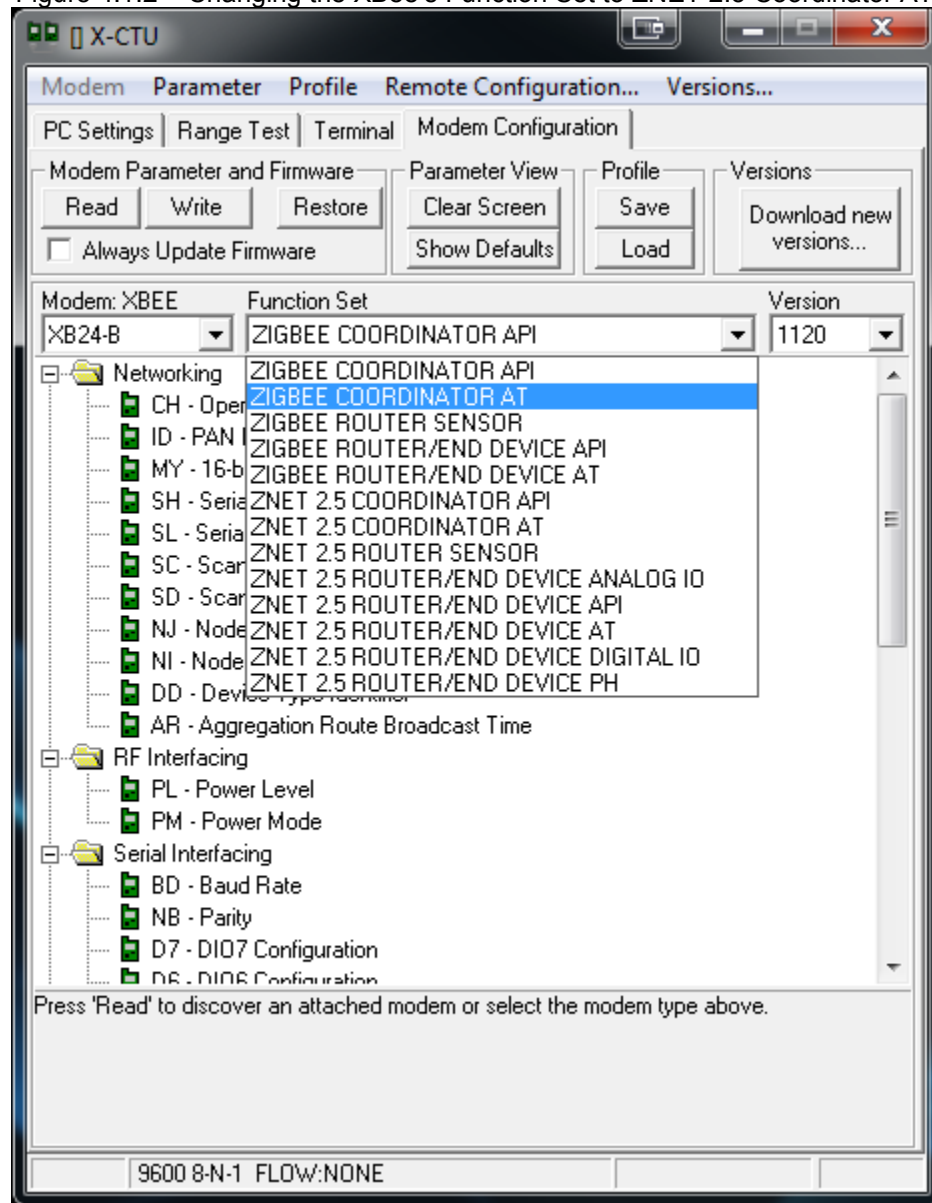
1. Construct the circuit in Figure 4.1.1 on the breadboard, connecting the XBee module to the XBee Explorer USB breakout board and the LEDs to the ASSOC (DI05 on breakout board) and ON (DI09 on breakout board) pins of the XBee RF module. The reset pin can be activated via a jumper cable instead of wiring in a switch.
2. Connect the XBee Explorer USB to the PC using the USB mini-B cable. Both LEDs connected to the ASSOC and ON pins of the XBee should turn on and remain lit.

Figure 4.1.1 – XBee Module to XBee Explorer Circuit Schematic



- 3.
4. Open X-CTU. In the “PC Settings” tab, select each COM port and push the “Test/Query” button to see if the module is connected to that port. Once the port that the XBee module is connected to is found and selected, click the “Modem Configuration” tab and then click the “Download new versions” button at the top right of the window. This will download any new firmware for the RF device. Once the download is finished, click “Read” at the top left of the window. All of the current parameters of the module will be displayed. Hit the dropdown button on the “Function Set” dropdown menu and select “ZNET 2.5 ROUTER AT”. This is shown in Figure 4.1.2. Change the Node Identifier NI parameter to unique name such as ‘router’. Change the PAN ID parameter to an exclusive number. Click the “Always update firmware” checkbox if it is not already checked and click “Write”. X-CTU might request that the XBee module be reset while the firmware is upgrading. Simply touch a jumper cable that is connected to ground to the RESET pin of the module.

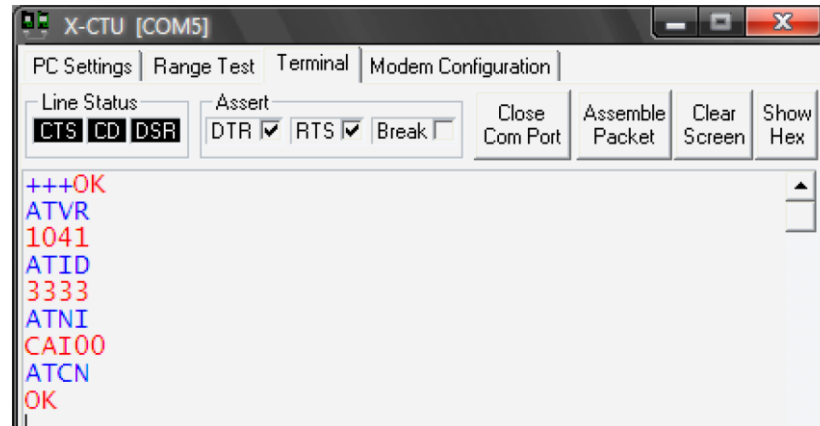
Figure 4.1.2 – Changing the XBee's Function Set to ZNET 2.5 Coordinator AT



5. When the firmware is finished upgrading, the XBee should be checked for its new firmware version, its network ID and its new Node ID. Click the "Terminal" tab. The following commands need to be entered within five seconds of one another or the device will exit Command mode and return to Idle mode. Type "+++". The module should respond after a couple seconds with "OK". Type "ATVR" and after a couple seconds the module should respond with the correct firmware version that it was just upgraded to. Type "ATID" and the module will return the network ID. The default ID is "123". Type "ATNI" and the module should return the Node Identifier that was previously set in step 3. Finally, to exit Command mode type "ATCN" and the module will return "OK", indicating that the device has

successfully exited command mode. This process is shown in Figure Figure 4.1.3.

Figure 4.1.3 – Verifying XBee Parameters in Command mode

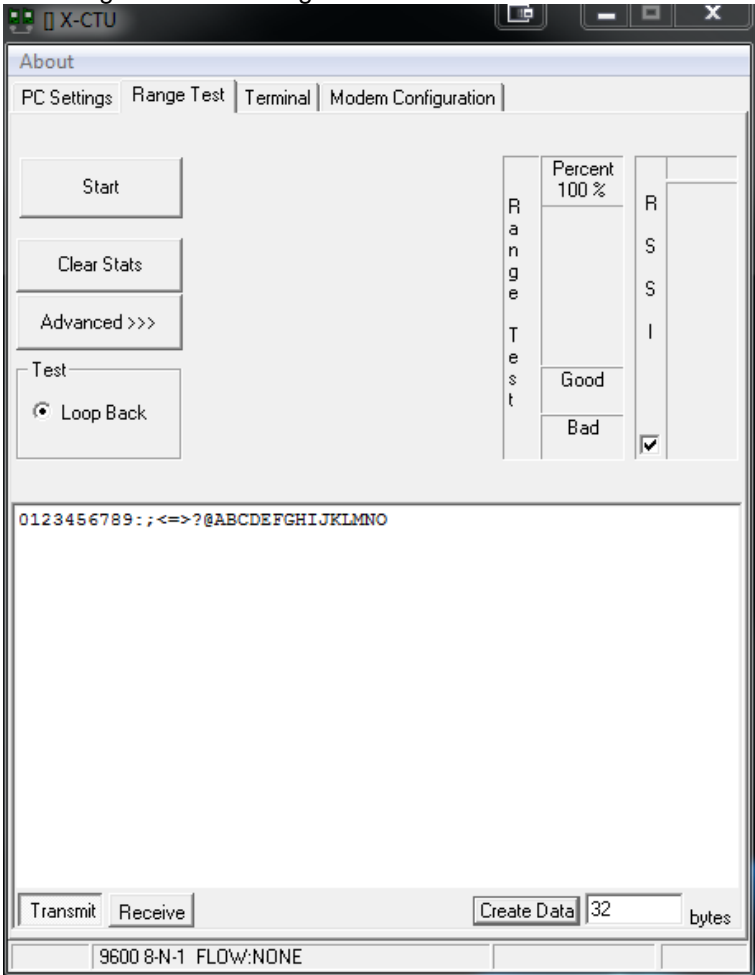


Used with Permission from Cai Maver [19]

6. Repeat steps 1-4, only now on step 3 change the Function Set to “ZNET 2.5 COORDINATOR AT” instead of router and change its Node Identifier to an exclusive name like “coordinator.” Change the “PAN ID” parameter to the same value as before with the “router” module. When the coordinator firmware is finished upgrading, the LED connected to the “ASSOC” pin should be blinking periodically.
7. To test that the network connection between the two XBee modules, remove, but do not discard the jumper pins on the XBee shield marked “XBEE/USB.” Connect the “router” XBee module to the XBee Shield and connect it to the Arduino development board. Connect the other “coordinator” XBee module to the PC via the Xbee Explorer interface. Ensure that the “Associate” LED on the PC-side XBee is blinking periodically. Connect the Arduino to a power source. If the “Associate” LED on the XBee Shield is blinking periodically, the two modules have successfully created a network. If the network connection is not correctly established, check the following parameters:
 - The PAN ID (ID) parameter must be the same value for both XBee modules
 - Destination Address Low (DL) parameter on the coordinator module set to FFFF
 - Destination Address Low (DL) parameter on the router set to 0000
 - The Baud Rate (BD) parameters of both modules are equal (9600 bps default). [1]
8. Once a successful network has been established, a range test is performed to verify functionality of the communications system at long

ranges. The range test should be performed in a flat, open field for minimal signal obstruction. To perform a range test, connect one XBee module to a laptop running X-CTU. Connect the other to the Arduino development board and connect it to a power source. In X-CTU, click the “Range Test” tab and check the “RSSI” checkbox to display signal strength. Figure 4.1.4 shows the range test user interface.

Figure 4.1.4 – Range Test GUI with RSSI Selected



With both RF devices still in close proximity, click the “Start” button in X-CTU. Move the embedded-side router device to the maximum distance where full functionality is still required and monitor the Range Test GUI. Signal strength should remain in the green throughout the range of separation between the two modules. [19]

4.1.1 Camera link

Purpose and objective: The camera link quality is an important step and component in many subsequent image processing and user interface systems. The main objective of this test is to verify the link quality of the wireless camera

and verify connectivity in different environments, conditions, and loads. With the camera link properly established within specifications, the user is able to further tune and configure the settings for image processing and video relay.

Supplies:

- Cisco wireless-N camera
- Desktop or laptop PC with wireless connectivity
- OpenCV and user GUI software

Preparation:

- Connectivity and power for supplies, check specifications
 - Start software and prepare for initialization of connection with camera
1. Setup Cisco Wireless-N camera to Ad-hoc connection mode with image quality and frames per second set at desired testing rate.
 2. Set the correct networking settings for the camera and connecting devices, such as a laptop or desktop computer.
 3. Turn on laptop and establish connection between camera and laptop or desktop computer
 4. Attempt to stream the MJPEG stream through the browser.
 5. Verify stream in external program such as OpenCV.
 6. Analyze bandwidth calculation

The required outcome for the success of this test is a successful retrieval of the video stream in the browser and in OpenCV. If this test fails, then further action will needed to be taken to establish this link.

4.1.2 Camera range

Purpose and objective: The test objective is to verify that the link with the camera can be established within specifications given in design. The camera range should be established and known before any tuning or test runs are performed to maintain connectivity with the camera

Supplies:

- Cisco wireless-N camera
- Desktop or laptop PC with wireless connectivity
- OpenCV and user GUI software

Preparation:

- Connectivity and power for supplies, check specifications
- Start software and prepare for initialization of connection with camera
- Make sure there aren't other devices attempting to connect or large amounts of RF noise in the area during testing

Procedure:

1. Perform the camera link test to verify link is being established.
2. After camera link is established, distance the camera outdoors at the maximum specified distance. A good starting point for Wi-Fi networked camera is 30 ft.
3. Verify link quality with the built in link indicator or some external program within the OS.
4. Retest the stream and image quality within OpenCV and verify that stream is not jittery or has a significant lag of more than 100 milliseconds.
5. Repeat steps 1 – 4 for ranges up to maximum distance of link established outside, and then repeat again for indoors.

The outcome required for success is that outdoor range is a minimum of 30 feet and indoor range is a minimum of 20 feet. If the link quality diminishes significantly after the minimum ranges, then an external antenna may be needed to enhance link quality.

4.1.3 GPS link

Purpose and objective: With a strong signal for the GPS, the accuracy can be heavily relied on as well as changes in environmental conditions can be predicted and adjusted for. The GPS link test objective is to establish a quality link in less than 60 seconds cold start and 5 seconds hot start.

Supplies:

- Venus GPS module
- Antenna
- SkyTraq software

Preparation:

- Start SkyTraq software and connect the module with a serial link
- RF noise should be kept to a minimum during testing to achieve the most accurate results

Procedure:

1. Turn on GPS module with attached passive ceramic antenna outdoors.
2. After 60 seconds, a basic link should be acquired from cold start mode. This should give position within 10 meters of accuracy.
3. Depending on the time of day, latitude and longitude of the module, a 12 satellite link should be acquired within a couple minutes.
4. If link is not acquired test has failed. A modification to the antenna or weather conditions may affect actual performance.

The GPS link needs to be acquired in less than or equal to 60 seconds on cold start and less than 5 seconds on hot start. If the link is not acquired in these times or less, the test has failed.

4.1.4 GPS accuracy

Purpose and objective: The accuracy of the GPS unit allows for the following of waypoints for perimeter navigation and scanning. The main objective is to test the GPS accuracy for less than 3 m accuracy outdoors for position and less than 1 meter per second velocity error.

Supplies:

- Venus GPS module
- Antenna
- SkyTraq software

Preparation:

- Start SkyTraq software and connect the module with a serial link
- RF noise should be kept to a minimum during testing to achieve the most accurate results

Procedure:

1. GPS module should be turned on and a 12 satellite tracking link should be acquired. Follow steps from GPS link test procedure.
2. The location of a known point should be recorded externally and verified for at least 10 locations. The test should pass with 90% confidence level.
3. The module and antenna should be placed directly over the test location and the latitude and longitude from the GPS should be recorded. An average may need to be taken.
4. These locations can be defined by the test engineer or may be located for geodetic survey databases available.

The link should provide location accuracy of less than 3 meters position error and velocity accuracy of ± 1 meter per second. If these parameters are not met and cannot be met, then the software may need to be tuned for these inaccuracies.

4.2 User Interface and Controls Testing

Purpose and objective: After the wireless network has been configured and tested to ensure proper communications, the user interface and control code can be tested for correct functionality. The following steps outline the test procedure:

1. Connect the Coordinator RF module to a PC with the control software installed. The LED connected to the Associate pin of the RF module should be blinking periodically.
2. Turn on the Surveillance vehicle. Check to see that the Associate pin LED on the board is blinking. If it is not, a network connection between the Coordinator and Router has not been established.

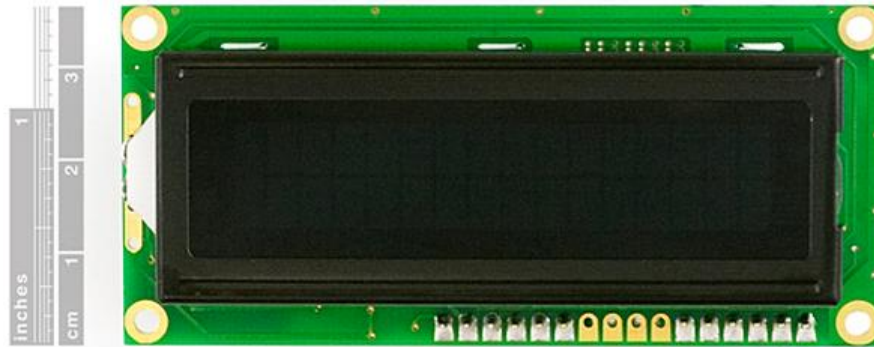
3. On the PC, run the control software executable and select the correct COM port in the command prompt. The GUI should load and a real-time video image streaming from the vehicle's camera should appear.
4. Turn the "Steering Idle" knob left and right. The vehicle's front wheels should turn left and right as you turn the knob. When the knob is released, the wheels should remain in whatever position specified by the knob.
5. Push the left and right arrows buttons on the GUI. The wheels should turn left and right when a left or right arrow button is pushed, respectively. When the button is released, the wheels should return to the position specified by the "Steering Idle" knob.
6. Push the up and down arrow buttons on the GUI. The vehicle should move forward and backward, respectively.
7. Make sure that the "Enable Stabilization" checkbox is checked. Lift the vehicle and, with the camera facing you, rotate (roll) the vehicle left and right about 45 degrees. Pitch the vehicle forward and backward about 45 degrees. Finally, spin (yaw) the vehicle about its vertical axis. The camera platform should remain upright, facing the same direction, and stable throughout the last three axial movements.
8. Uncheck the "Enable Stabilization" checkbox and repeat the movements described in step 6. The platform should make no attempts to stabilize itself.
9. Move the "Pitch", "Yaw", and "Roll" controls on the PC User Interface. The camera platform should respond accordingly.
10. Set the "HF Bound" slider to its maximum value and the "LF Bound" slider to its minimum value. Set the "SPL Level" slider to its Maximum value and observe the image. No flashing indicator arrows should appear at the edges of the screen. Now lower the "SPL Level" slider until sound detection indicator arrows begin to flash at the edges of the screen. Slowly raise the SPL slider until no flashing arrows appear. Tap each onboard microphone on the vehicle and check to see that sound detection arrows flash accordingly on the screen.

4.2.1 LCD output

Purpose and objective: For development and real time feedback during testing and modifications, a 16x2 monochrome LCD display will be used to communicate with the system. This has the advantage of a direct connection with the S.H.A.S.bot, which may be advantageous if the wireless systems need to be shut down temporarily for modifications or if there is a software or firmware upgrade

that needs to be uploaded and immediate reliable feedback is desired. Figure 3.1.2 features a picture of the LCD display.

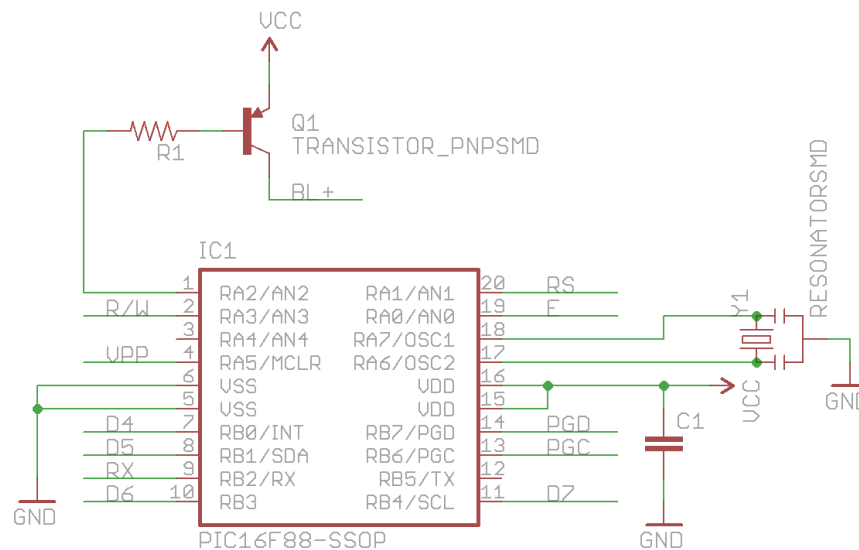
Figure 3.1.2 – 16x2 character monochrome LCD module



Printed under a Creative Commons license

Since the LCD is being used for development and debugging, it will be connected to an external power supply and have a debug mode set in software to utilize a single RX pin for a serial 3.3V TTL connection. The following schematic in Figure 3.1.3 shows how the LCD will be wired.

Figure 3.1.3 – Eagle schematic for LCD



Printed under a Creative Commons license

General Procedure:

1. Connect LCD to external power or through ATmega328 PCB. The VCC pin should be connected to a 5 V rail, while the TTL logic is at 3.3 V.
2. Software should be set manually to debug mode, or debug procedures should be in place while using the output LCD.

3. In the programming software for the ATmega328, the output pin and libraries for the serial communication for the LCD need to be loaded and compiled.
4. The pin for the output also needs to be set to an open digital pin on the ATmega328.
5. Once connected the program can run as normal and any debug outputs can be displayed during runtime on the module.

4.3 Image Processing

4.3.1 Motion detection

Purpose and objective: The test objective for motion detection is to be able to test motion detection of a person down to a resolution of at least 1 mph(1.47 ft/s) at 10 feet laterally. After the power systems are turned on and the connectivity between the camera and user is established, the procedure for testing can begin.

Supplies:

- Cisco wireless-N camera
- Desktop or laptop PC with wireless connectivity
- OpenCV and user GUI software

Preparation:

- Connectivity and power for supplies, check specifications
- Start software and prepare for initialization of connection with camera
- Verify camera link within range specifications is established with a strong signal
- OpenCV software should handle any video stream drops or errors

Procedure:

1. Camera should be placed outside with adequate lighting for nominal conditions.
2. If the testing conditions need to be changed, it should be noted in the testing documentation under which settings the test took place.
3. Turn on the camera and start the motion detection algorithm user side.
4. The algorithm or algorithms being used should be tuned for the application and should be ready to start.
5. An average sized person will walk at approximately 1 mph at a distance of 10 feet away from the camera perpendicularly.
6. The motion detection algorithms should be able to detect the motion continuously and provide the desired output to the user's GUI.
7. Further motion detection information should be passed along to tracking algorithms.

For the motion detection test procedure to provide acceptable results the motion detection must be successful at the specifications provided in the objective. As the distance increases from the camera the effective image resolution size of the object will decrease. For each application and environment the desired effective image resolution may differ, but tuning within the specifications given allows the user to have a well-established baseline.

4.3.2 Gating

Purpose and objective: The gating test will measure the effectiveness of our gating parameters for tracking and targeting.

Supplies:

- Cisco wireless-N camera
- Desktop or laptop PC with wireless connectivity
- OpenCV and user GUI software

Preparation:

- Connectivity and power for supplies, check specifications
- Start software and prepare for initialization of connection with camera
- Verify camera link within range specifications is established with a strong signal
- OpenCV software should handle any video stream drops or errors

Procedure:

1. Set up camera and S.H.A.S.bot for motion detection procedure.
2. Pull up GUI and start the motion detection
3. Gating should start and each track should gate properly with gate size and prediction

The gates should follow the target and encompass the centroid. For a target with a velocity the gating should follow the target until the target breaks the threshold velocity.

4.3.3 Centroid calculation

Purpose and objective: The objective of the centroid calculation test is to verify that the centroid calculation is accurate.

Supplies:

- Cisco wireless-N camera
- Desktop or laptop PC with wireless connectivity
- OpenCV and user GUI software

Preparation:

- Connectivity and power for supplies, check specifications

- Start software and prepare for initialization of connection with camera
- Verify camera link within range specifications is established with a strong signal
- OpenCV software should handle any video stream drops or errors

Procedure:

1. With blob detection algorithm started, the target should be within recognition range of the vehicle.
2. The target should begin translation across the camera's field of view.
3. The centroid of the blob should be calculated and displayed when in development mode. Normally disabled for reducing clutter on display

The gates should follow the target and encompass the centroid. For a target with a velocity the gating should follow the target until the target breaks the threshold velocity.

4.3.4 Tracking

Purpose and objective: The image processing algorithms on the PC side will require a lot of tuning and testing to perform within specifications. The objective of this test is to receive successful tracking of the targets.

Supplies:

- Cisco wireless-N camera
- Desktop or laptop PC with wireless connectivity
- OpenCV and user GUI software
- Tracking software

Preparation:

- Connectivity and power for supplies, check specifications
- Start software and prepare for initialization of connection with camera
- Verify camera link within range specifications is established with a strong signal
- OpenCV software should handle any video stream drops or errors

Procedure:

1. Camera should be placed outside with adequate lighting for nominal conditions.
2. If the testing conditions need to be changed, it should be noted in the testing documentation under which settings the test took place.
3. Turn on the camera and start the motion detection and tracking algorithms user side.
4. The target should be distanced from ten to twenty-five feet away.
5. The motion detected from the target should create a track, which will be indicated on the GUI. If it is not, tuning may be required on the gating algorithms.

6. As the target moves off the field of view the track can either follow it through motor control or the track will be lost.

The tracks created must be able to show the targets location, measurement, and prediction.

4.4 Navigation

4.4.1 Auto-route

Purpose and objective: The auto-route feature is part of the autonomous mode which involves the S.H.A.S.bot routing itself to the destination successfully. The objective for this test is to perform some basic routing and successfully reach the destination.

Supplies:

- Assembled S.H.A.S.bot vehicle
- Microcontroller loaded with navigation software
- PC loaded with ranging software

Preparation:

- Load vehicle with navigation software and set desired parameters
- Power on vehicle and verify systems are functional

Procedure:

1. After systems are functional the user should load desired waypoints into the navigation software via the GUI
2. Set the S.H.A.S.bot into autonomous mode and the auto-route function should begin
3. The user can view the video feed as the S.H.A.S.bot runs through the perimeter, looking for any suspicious activity.
4. As the auto-route function routine completes the user can take control or set the routine to auto-route and patrol the area.

The criteria for success is that the S.H.A.S.bot completes the perimeter in a reasonable amount of time, not more 50% increase in time it would take the user to navigate, and that it does not encounter and routing errors or anomalies.

4.4.2 Obstacle avoidance

Purpose and objective: During auto-route autonomous mode the S.H.A.S.bot will need to automatically avoid and obstacles that could obstruct its path. The obstacle avoidance routine should modify the path defined by the user so that the S.H.A.S.bot is able to self-navigate around the obstacle and complete the perimeter routine. The objective is to do the above mentioned successfully and avoid hitting obstacles. Moving obstacles will not be tested.

Supplies:

- Assembled S.H.A.S.bot vehicle
- Microcontroller loaded with navigation software
- PC loaded with ranging software
- Obstacle avoidance software

Preparation:

- Load vehicle with navigation software, obstacle avoidance software, and properly set desired parameters for the run
- Power on vehicle and verify systems are functional
- Enable the obstacle avoidance mode and set obstacles

Procedure:

1. After systems are functional the user should load desired waypoints into the navigation software via the GUI.
2. Set the S.H.A.S.bot into autonomous mode and the auto-route function should begin.
3. The user can view the video feed as the S.H.A.S.bot runs through the perimeter, looking for any suspicious activity.
4. The obstacle avoidance feature should have been set in the parameters. If this is not set, it may be turned on or off during the run.
5. With the obstacle avoidance feature enabled, an obstacle will be placed in the path of the S.H.A.S.bot.

The vehicle should maneuver around the obstacles by checking for a clear path and navigating appropriately, then resuming the closest approximation to the previous path and ultimately reach the destination.

4.5 Stabilization Platform Testing Procedure

Purpose and objective: The testing of the platform involves two systems: the microcontroller and inertial measurement unit system, and the servo motors and microcontroller system. For the purposes of this testing, using an Arduino Uno breakout board and a MPU-6050 breakout board is acceptable, as this allows us to ensure our design integrity before printing a circuit board. It also expedites the testing process and decreases the complexity involved in configuring the testing system. Before we begin testing these individual systems we must ensure that the proper prerequisites are being met to supply the system. Refer to [stabilization design decisions schematic] for the proper setup of the system, and ensure the pins are connected as described in [stabilization platform design decisions table]. The testing procedure is as follows:

Procedure:

1. Ensure that the supply voltages for the servo motors and the ATmega328 are all 5V.

2. Ensure that the supply voltage for the MPU-6050 inertial measurement unit is 3.3V.
3. Run a simple Arduino loop on the ATmega328 utilizing the writeMicroseconds() and Read() functions from the Servo library. The attach() function should allow us to properly control each servo using the writeMicroseconds() command. Use a simple Arduino sketch and output a Read() command after each servo command to the display.
4. Utilize a simple Arduino sketch to output data from the MPU-6050, and ensure that communication is occurring between the inertial measurement unit and the ATmega328.
5. Utilize the DMP capability of the MPU-6050 or create your own algorithms to transform the raw sensor data into an angle reading, output this information to the screen, and ensure relative angle calculation accuracy.
6. Using the software created in step 5, run a basic loop that uses a user-defined setpoint and the angle data we now have to properly control the correction of the platform.
7. Begin implementing the PID Arduino library and experiment with different control loop parameters, utilizing a similar algorithm from step 6, to maximize the efficiency of the stabilization platform and guarantee smooth movement of the servo motors.
8. Test the control loop at frequencies ranging from 50Hz to 100Hz, observe and record response time, and locate the optimal correction rate.

The successful completion of these steps ensures that the stabilization platform system is ready for integration with the chassis. These steps also ensure that the circuitry behind the control of this unit is well-designed and suited for creation on a printed circuit board.

4.6 Audio System Testing

4.6.1 Microphone Voltage Testing Procedure (Audio Test 1)

Purpose and objective: A microphone's functionality is quite simple; sound waves cause a coil to oscillate past a magnet, therefore generating an electrical current. All microphones are different though; some are expertly hand crafted with the finest materials to be super sensitive to sound waves and generate a larger electric current. These microphones are also extremely expensive. The microphone cartridges we will be using will not be as expensive nor will they generate a very large current, so we must be able to determine the voltage that our microphones will generate at certain levels of sound gain.

In the section "defining a critical audio event," we determined that 50 dB was an appropriate sound level for the vehicle to alert the user of. The only problem with that, though, is that we have no way of knowing what kind of voltage that translates to on our microphone without any sort of testing. So, we have come up with a testing procedure that will tell us what kind of voltage a 50 dB sound (at the location of the microphone) will produce on the microphone.

Determining this voltage is very important because in order to know how to (in analog) program the comparator to alert the microprocessor of a critical audio event, we must know the voltage that a 50 dB level sound will produce on the microphone.

Supplies needed:

- The directional microphone we will be using
- Multimeter with a 0.1 mV sensitivity (since the signal will be very small)
- Pos. and neg. multimeter probes
- Laptop with the ability to run audio through an output
- Laptop program that will output user defined frequencies through audio output
- A single speaker that will connect to the laptop's audio output
- A decibel meter, with attached microphone

Preparation:

- Connect the speaker to the audio output jack of the laptop.
- Make sure the multimeter is calibrated.
- Make sure that the testing environment is completely silent.

Procedure:

1. Turn on multimeter, decibel meter, and laptop program.
2. Place the decibel meter's microphone 1 inch away from the speaker.
3. Turn laptop volume to its highest setting, and set the program to output a constant tone at 440 Hz.
4. Observe the reading on the decibel meter.
5. Adjust the volume on the laptop until the decibel meter reads 50 dB.
6. Keep the laptop's audio output at this volume for the rest of the experiment.
7. Attach the positive multimeter probe to the 1/8" jack extending from our directional microphone. Attach the negative multimeter probe to ground.
8. Place the end of our directional microphone 1 inch away from the speaker.
9. Set the laptop program to output a constant tone at 440 Hz, at the 50 dB adjusted volume.
10. Observe the voltage reading on the multimeter.
11. Record the voltage reading.

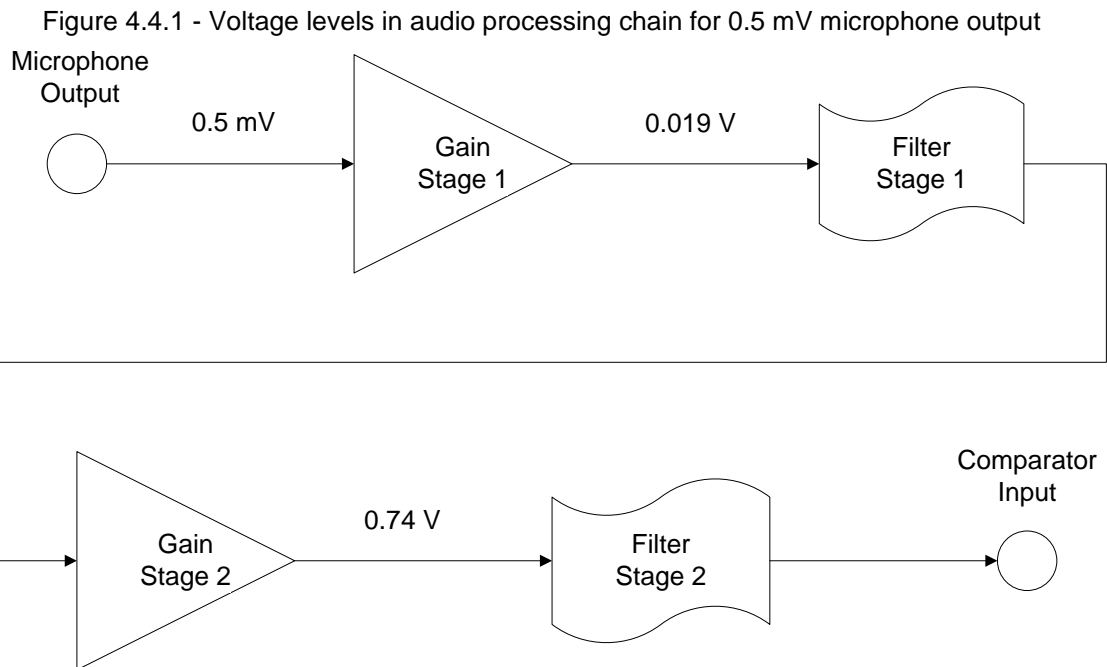
Once we know the voltage that the microphone produces whilst picking up a 50 dB sound, we can determine what the threshold voltage will be for our comparator. Then we can successfully alert the user of a 50 dB or more sound detection.

This is the first test we will perform because before we do any other testing or make any adjustments on the audio board, we must know the exact voltage that our microphones produce when picking up a 50 dB sound.

4.6.2 Audio Board Testing Procedure (Audio Test 2)

Because there are so many aspects of the audio board on our vehicle that have to be right in order for the audio alert system to work correctly, we must have a rigorous testing procedure in place that insures the proper function of the audio board and GUI connection as well. We will not only have to test the connections on the board itself, but also test the functionality of the microphones and see if they are alerting properly to a critical audio event.

One problem we run into is the fact that, at this point, we have no idea what kind of voltage our microphones will produce at the threshold sound level of 50 dB. Therefore, for continuity's sake, we will approximate that voltage to be 0.5 mV. Before we get into the details of the experiment, here is a diagram, Figure 4.4.1, of what the voltages should be at each step of the audio chain assuming that the output from the microphone is 0.5 mV. We will also assume, for now, that there is no insertion loss due to the filters.



Let us also assume that the comparator's low output voltage is 0 V, and its high output voltage is 0.7 V.

Supplies:

- Our vehicle's audio board, with microphones properly hooked up
- Multimeter with a 0.1 mV sensitivity (since the signal will be very small)
- Pos. and neg. multimeter probes
- Laptop with the ability to run audio through an output

- Laptop program that will output user defined frequencies through audio output
- A single speaker that will connect to the laptop's audio output
- A decibel meter, with attached microphone

Preparations:

- Connect the speaker to the audio output jack of the laptop.
- Make sure the multimeter is calibrated.
- Make sure that the testing environment is completely silent.

Procedure:

1. Turn on the multimeter, decibel meter, and laptop program.
2. Turn on the vehicle's power systems.
3. Connect the positive multimeter probe to the Vcc pin of the first amplifier (gain stage 1). Connect the negative multimeter probe to ground. Ensure that the voltage on the Vcc pin is 5V.
4. Disconnect the positive multimeter probe from the first amplifier's Vcc pin and connect it to the Vcc pin of the first filter (filter stage 1). Keep the negative multimeter probe connected to ground. Ensure that the voltage on the Vcc pin is 5V.
5. Disconnect the positive multimeter probe from the first filter's Vcc pin and connect it to the Vcc pin of the second amplifier (gain stage 2). Keep the negative multimeter probe connected to ground. Ensure that the voltage on the Vcc pin is 5V.
6. Disconnect the positive multimeter probe from the second amplifier's Vcc pin and connect it to the Vcc pin of the second filter (filter stage 2). Keep the negative multimeter probe connected to ground. Ensure that the voltage on the Vcc pin is 5V.
7. Disconnect the positive multimeter probe from the second filter's Vcc pin and connect it to the voltage comparator's Vcc pin. Keep the negative multimeter probe connected to ground. Ensure that the voltage on the Vcc pin is 5V.
8. Disconnect the positive multimeter probe from the voltage comparator's Vcc pin and connect it to the voltage comparator's reference voltage pin. Keep the negative multimeter probe connected to ground. Ensure that the voltage on the reference voltage pin is 0.74V.
9. Place the decibel meter's microphone 1 inch away from the speaker.
10. Turn laptop volume to its highest setting, and set the program to output a constant tone at 440 Hz.
11. Observe the reading on the decibel meter.
12. Adjust the volume on the laptop until the decibel meter reads 50 dB.
13. While the tone is still constantly playing out of the speaker at 50 dB, place the speaker 1 inch away from the front of the directional microphone.
14. Connect the positive multimeter probe between the microphone output and the first amplifier (gain stage 1). Connect the negative multimeter probe to ground.

15. Verify on the multimeter that the voltage is 0.5 mV.
16. Connect the positive multimeter probe between the first amplifier (gain stage 1) and the first filter (filter stage 1). Keep the negative multimeter probe connected to ground.
17. Verify on the multimeter that the voltage is 0.019 V.
18. Connect the positive multimeter probe between the second amplifier (gain stage 2) and the second filter (filter stage 2). Keep the negative multimeter probe connected to ground.
19. Verify on the multimeter that the voltage is 0.74 V.
20. Connect the positive multimeter probe between the voltage comparator output and the arduino microprocessor. Keep the negative multimeter probe connected to ground.
21. Verify on the multimeter that the voltage is 0.7 V.
22. Turn the tone output program on the laptop off and ensure that there is no sound coming out of the speaker by verifying that the decibel meter reads 0 dB.
23. Keep the positive multimeter probe connected between the voltage comparator output and the arduino microprocessor. Keep the negative multimeter probe connected to ground.
24. Now with no sound emitted from the speaker, verify that the voltage on the multimeter is 0 V.
25. Repeat steps 3 through 24 for each of the next five microphones and their respective audio chains.

After successfully completing this testing procedure you will have verified that:

- All active parts are receiving their necessary power.
- All stages of the audio chain are at the correct voltage with a 50 dB sound pickup.
- The voltage comparator is outputting the correct voltage with a 50 dB sound pickup and a 0 dB sound pickup.

This is the second test that we will perform, with the first test being the microphone voltage test. Once we have verified that everything is correct on the audio board, we can then move to test 3 which is the audio alert system functionality test. This final test will show us whether or not our GUI is functioning and communicating properly with our audio board.

4.6.3 Audio Alert System Testing Procedure (Audio Test 3)

Objective: When our vehicle's audio system is completely connected and integrated with the GUI, we will have to test the functionality of system. We want the system to alert the user if one or more of the microphones pick up a sound at 50 dB or more. To test this we must sweep the microphones with a continuous range of volume to see if the alert display on the GUI properly informs the user such a sound.

Supplies:

- The directional microphone we will be using
- Multimeter with a 0.1 mV sensitivity (since the signal will be very small)
- Pos. and neg. multimeter probes
- Laptop with the ability to run audio through an output
- Laptop program that will output user defined frequencies through audio output
- A single speaker that will connect to the laptop's audio output
- A decibel meter, with attached microphone

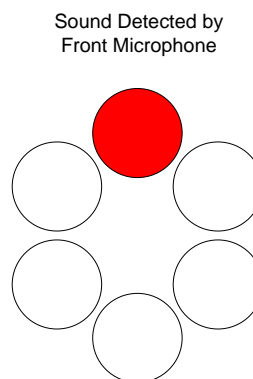
Preparation:

- Connect the speaker to the audio output jack of the laptop.
- Make sure the multimeter is calibrated.
- Make sure that the testing environment is completely silent.

Procedure:

1. Turn on multimeter, decibel meter, and laptop program.
2. Place the decibel meter's microphone 1 inch away from the speaker.
3. Also place the end of one of the vehicle's directional microphones 1 inch away from the speaker.
4. Turn on all vehicle power systems.
5. Power on the GUI.
6. Turn the volume on the laptop all the way down.
7. Set the program to output a constant tone at 440 Hz.
8. Slowly turn the laptop's volume up, observing the decibel meter and the GUI interface at the same time.
9. Once the sound level reaches 50 dB on the decibel meter, verify that the GUI interface has alerted the user of a critical audio event. The alert should present itself as the white circle corresponding to the directional microphone being tested turning bright red. See Figure 4.4.2 below for a visual example.

Figure 4.4.2 - GUI front microphone audio event alert visual



This is the last test to be done for the audio portion of our vehicle. Once the functionality of the connection between the audio board and the GUI is verified, this means that all audio systems are go for the overall use of the vehicle. The vehicle can now be run with certainty that the audio system will alert correctly for a critical audio event.

4.7 PCB Design

This project had three separate printed circuit boards, in order to have three separate systems easier to troubleshoot. These include the power board, which contained our voltage regulators, current sensor, and voltage sensing network, the stabilization board, which contained a breakout for our IMU, GPS, and the stabilization microcontroller, and the communication board, which contained the xbee module breakout, the wireless communication microcontroller, and the audio detection circuitry. The power board supplied power to each of the other boards, to the motor controllers, and to the camera, the stabilization board controlled our platform, and the communication board controlled our vehicle and relayed information back to the user. The boards were each a two layer design, with components being placed on the top of each board. Each layer had a ground plane surrounding the majority of the components, and we used mostly through-hole parts in order to make swapping out components easier. We designed the circuit boards in the Eagle software, and had them fabricated by Advanced Circuits. Figure 4.7.1 shows the power board layout. As you can see, we had breakouts for the communication board and the stabilization board to receive all the power they required and we had the voltage and current sensing lines going to the communication board as well. We also had a barrel jack connection for the camera, and screw terminals for the motor controller and the battery.

Figure 4.7.1 - Power Board

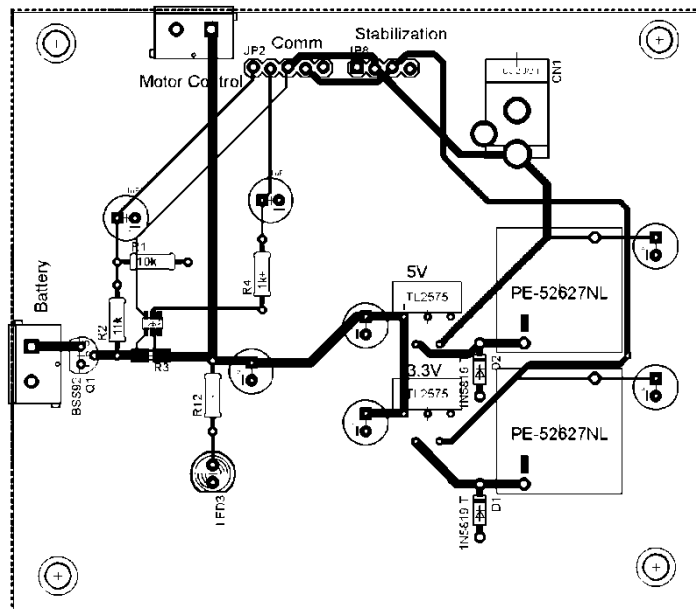
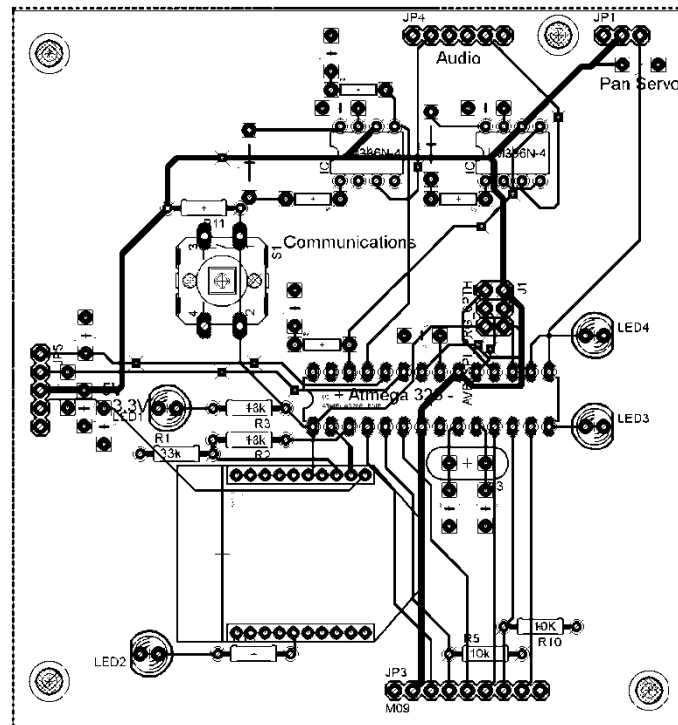


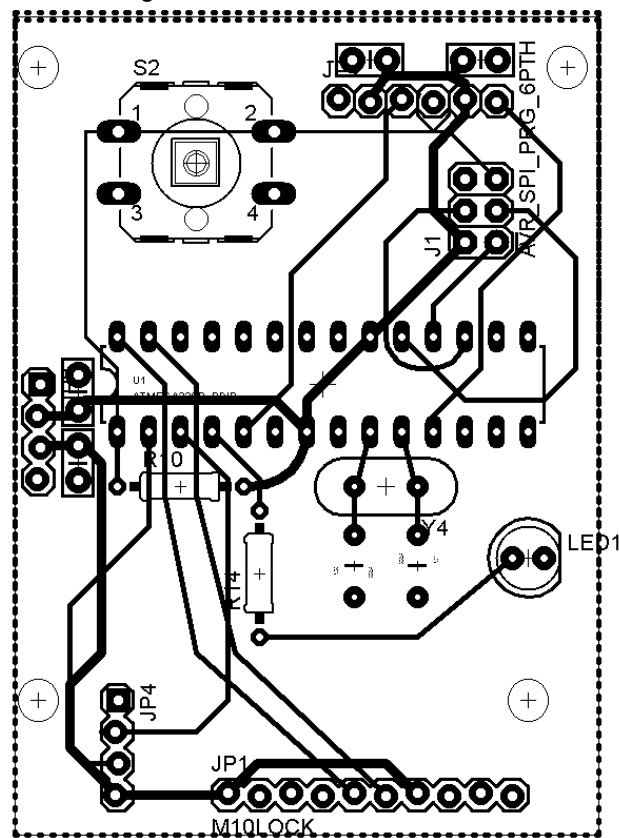
Figure 4.7.2 shows the communication board. As you can see here, we had breakout connections to connect the pan servo, the audio microphone inputs, the xbee board, the motor controller, and inputs for the voltage, current, and voltage and current sensing connections. We had impedance matching on the traces that connected the crystal in order to ensure that our timing was as precise as possible for our microcontroller.

Figure 4.7.2 – Communication Board



Finally, figure 4.7.3 shows the stabilization board. As you can see, here we had a breakout header for connections to the inertial measurement unit, an optional GPS unit, and headers for our pitch and roll servos, and for input power. Here we also used impedance matching for our traces going to the crystal for the microcontroller clock.

Figure 4.7.3 – Stabilization Board



5 PROJECT SUMMARY

This document outlined our design for the S.H.A.S.bot, a complete solution for reconnaissance and surveillance scenarios. The combination of the stabilization platform, the audio detection system, and wireless control capabilities, all coupled with an intuitive GUI provides a unique solution for industrial and military, as well as recreational applications. The product is relatively inexpensive to build considering its capabilities, is easily portable, and is scalable to suit your needs. The final working design of our project was unable to include the current sensing section, as the package designed in Eagle for the part was not the version of the part we had available. The audio detection was functional, but at the time of presentation we did not have its features in the GUI. However, we did include the functionality test for it by having LEDs blink each time noise was picked up. The panning servo was also unavailable in the GUI due to time constraints, but the functionality was there.

Table 5.1.1 - Final Budget

Part name	Description	QTY	Total cost
MPU-6050	6 Degrees of Freedom 3-axis gyro/accelerometer	1	\$19.95
Voltage regulators	Voltage regulators for 5 and 3.3 volts	4	\$10.00
Motor controller	Seedstudio motor shield	1	\$19.99
Battery w/ contr.	14.4 V Li-Poly	1	\$99.95
Microphones	Directional (modified)	2	\$50.00
Audio Amplifiers	1.4W Analog Devices	6	\$25.00
Audio Filters	Maxim Active Low Pass Filter	6	\$25.00
Voltage Comparator	Philips 74HC	6	\$6.00
Audio Connector	1/8" Audio Three Pronged Socket	6	\$6.00
Microphone mounting	Mounting equipment	1	\$15.00
Series 2 XBee-Pro RF Modules	Wireless RF Communications Module	2	\$42.00
XBee Explorer USB	Serial to USB Interface for XBee	1	\$24.95
Arduino Uno	Development Board	1	\$26.56
USB to Mini-B Cable	USB A to Mini B cable	1	\$1.77
General Electronic Components	Capacitors, resistors, LEDs, switches, transistors, diodes, etc.	1	\$30.00
Antenna	GPS antenna	1	\$9.95
ATMEGA328P-PN	Microcontroller - DIP	2	\$10.00
Total:			\$422.12

6 BIBLIOGRAPHY

- [1] Electropaedia - Battery and Energy Technologies, "Nickel Cadmium Batteries," 2005, <http://www.mpoweruk.com/nicad.htm>
- [2] Electropaedia - Battery and energy Technologies, "Nickel Metral Hydride Batteries," 2005, <http://www.mpoweruk.com/nimh.htm>
- [3] P. Emerald, "Non-Intrusive Hall-Effect Current-Sensing Techniques," 1998, <http://www.allegromicro.com/en/Products/Design/non-intrusive-currentsensing/>
- [4] Tamura Corporation of America, "Current Sensor Info," 2012, <http://www.tamuracorp.com/products/current-sensor-info/>
- [5] Toroid Corporation of Maryland, "Technical Topics: AC Current Transformers or Current Sensors (CTs)," 2004, http://www.toroid.com/knowledge_base_/AC_current_transformers.htm
- [6] Maxim Integrated Products, "DC-DC Converter Tutorial," 2001, <http://www.maxim-ic.com/app-notes/index.mvp/id/2031>
- [7] Maxim Integrated Products, "Regulator Topologies for Battery-Powered Systems," 2001, <http://www.maxim-ic.com/app-notes/index.mvp/id/660>
- [8] Texas Instruments, "Traditional High Side Current Sensing," <http://focus.ti.com/analog/docs/microsite.tsp?sectionId=560&tabId=2182µsiteId=7>
- [9] "Introduction to Serial Communications." *Taltech Instrumental Software Solutions*. N.p., 02, Feb 2011. Web. <http://www.taltech.com/support/entry/serial_intro>.
- [10] "begin()." *Arduino*. N.p., 30, Jan, 2012. Web. <<http://arduino.cc/en/Serial/Begin>>.
- [11] Durda, Frank. "Serial and UART Tutorial." . N.p., 13, Jan 1996. Web. 1 Aug 2012. <http://www.freebsd.org/doc/en_US.ISO8859-1/articles/serial-uart/index.html>.
- [12] "SoftwareSerial Library." *Arduino*. N.p., 25, May 2012. Web. <<http://arduino.cc/hu/Reference/SoftwareSerial>>.
- [13] "X-CTU." *Software Informer*. N.p., 2012. Web. <<http://x-ctu.software.informer.com/>>.


- [14] "Java SE Technical Documentation." *Oracle Technology Network*. N.p., n.d. Web. <<http://docs.oracle.com/javase/>>.
- [15] Fry, Ben, and Casey Reas. "Overview: A Short Introduction to the Processing Software and Projects from the Community." *Processing*. N.p., n.d. Web. <<http://processing.org/about/>>.
- [16] Schlegel, Andreas. *ControlP5*. N.p., n.d. Web. 1, Aug 2012. <<http://www.sojamo.de/libraries/controlP5/>>
- [17] "XBee™ ZNet 2.5/XBee-PRO™ ZNet 2.5 OEM RF Modules." *Digi: Your M2M Solutions Expert*. Digi, 02, Nov 2008. Web. <<http://www.embedded.arch.ethz.ch/xbec-setup.pdf>>.
- [18] Seidle, Nathan. "XBee-Explorer-v15 Datasheet." (2010): n.pag. Database. <<http://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-Explorer-v15.pdf>>.
- [19] "How-to: Setting up XBee ZNet 2.5 (Series 2) modules." *EmbeddedLab*. N.p., 8, Aug 2008. Web. <<http://www.embedded.arch.ethz.ch/xbec-setup.pdf>>.
- [20] James R. Hollinger and John E. Mulligan, "Build the Shotgun Sound Snooper," 1965. <http://colinhartonline.com/wordpress/wp-content/uploads/2010/03/ShotGunMike.pdf>
- [21] Tyler Cartner, "How to Make a Shotgun Shock Mount," 2011. http://www.microfilmmaker.com/tipstrick/Issue10/shock3_1.html
- [22] Hyperphysics Educational, "Audible Sound," 2005. <http://hyperphysics.phy-astr.gsu.edu/hbase/sound/earsens.html>
- [23] Arduino Tutorial, "Analog Input," 2012. <http://arduino.cc/en/Tutorial/AnalogInput>
- [24] Sengpei Audio, "Decibel Table – SPL – Loudness Comparison Chart," 2012. <http://www.sengpielaudio.com/TableOfSoundPressureLevels.htm>
- [25] Sengpei Audio, "Damping of Sound Level with Distance," 2012. <http://www.sengpielaudio.com/calculator-distance.htm>
- [26] Analog Devices, "SSM2301 Datasheet," 2007. http://www.analog.com/static/imported-files/data_sheets/SSM2301.pdf

- [27] Maxim Semiconductor, "MAX291 Datasheet," 2010. <http://pdfserv.maxim-ic.com/en/ds/MAX291-MAX296.pdf>
- [28] Philips Semiconductor, "74HC Datasheet," 1990. http://www.nxp.com/documents/data_sheet/74HC_HCT85_CNV.pdf
- [29] Texas Instruments, "Low Side Current Sensing," 2012, <http://www.ti.com/analog/docs/microsite.tsp?sectionid=560&tabId=2181µsiteId=7>
- [30] R. Kalman, "A New Approach to Linear Filtering," *Transactions of the ASME—Journal of Basic Engineering*, 82 (Series D), 1960. Texas Instruments, "Low Side Current Sensing," 2012, <http://www.ti.com/analog/docs/microsite.tsp?sectionid=560&tabId=2181µsiteId=7>
- [31] E. Brookner, *Tracking and Kalman Filtering Made Easy*, New York: John Wiley & Sons, Inc, 1998. Texas Instruments, "Low Side Current Sensing," 2012,
- [32] D. H. Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes*," University of Rochester. Computer Science Department., New York, 1979.
- [33] G. Bradski and A. Kaehler, *Learning OpenCV*, Sebastopol, CA: O'Reilly Media Inc., 2008.
- [34] G. Hoffman, "Midterm Progress – OpenCV Blob Detection with Kinect," 5 March 2011. [Online]. Available: <http://itp.nyu.edu/~gh726/ITProcess/2011/03/midterm-progress-opencv-blob-detection-with-kinect/>. [Accessed 31 July 2012].
- [35] V. Kasik and T. Peterek, "Video Processing Toolbox for FPGA Powered Hardware," in *2011 International Conference on Software and Computer Applications*, Singapore, 2011.
- [36] S. Emami, "HSV color conversion," 4 October 2010. [Online]. Available: <http://www.shervinemami.co.cc/colorConversion.html>. [Accessed 1 August 2012].


- [37] SparkFun, "GPS Buying Guide," [Online]. Available: https://www.sparkfun.com/pages/GPS_Guide. [Accessed 27 07 2012].
- [38] P. Bennett, "The NMEA FAQ," 15 09 1997. [Online]. Available: <http://www.kh-gps.de/nmea.faq>. [Accessed 27 07 2012].
- [39] T. Acharya and A. Ray, Image Processing: Principles and Applications, New Jersey: John Wiley & Sons, Inc., 2005.

Permissions

Re: A Sincere Request for Permission to use Data from Your XBee Configuration/Setup TutorialHide Details

FROM: Cai Maver 

TO: Charlie Grubbs

Tuesday, July 31, 2012 11:53 PM 

Hi Charles-

Thank you for the e-mail. You can absolutely use any text, photos or anything else from the XBee tutorial I wrote. Sounds like an interesting project :)

- Cai

On Mon, Jul 30, 2012 at 9:07 PM, Charlie Grubbs <cafire4414@yahoo.com> wrote:

Hello,

I recently read your article on embedded.arch.ethz.ch about configuring and testing two XBee RF Modules and it was very informative! I am a student at the University of Central Florida currently designing an autonomous surveillance vehicle for a school project. I'm requesting if I may use the technical info as well as the picture of the X-CTU terminal in your document in my final paper. The project is completely non-profit and anything taken from your article would be properly cited. I appreciate your time and look forward to your response.

Thanks,

Charles Grubbs

Interest: *

First Name: *

Last Name: *

Company:

Email: *

Country: *

Phone:

Message:

Hello,

I am an Electrical Engineering student at the University of Central Florida currently working on my Senior Design project. I am requesting permission to reprint a graphic from the XBee™ ZNet 2.5/XBee-PRO™ ZNet 2.5 OEM RF Modules Product manual in my final report. All material used will be credited and cited properly throughout the documentation. The project and report are not for profit and will not be published.

Regards,

Charles Grubbs


*

Permission for picture use - Google Chrome

https://by2prd710.outlook.com/owa/7ae5Item8a=Open&it=IPM.Note&id=RgAAAAD4mWDQalhvQLcRbmbdMcNbwAVH0e

Reply Reply All Forward Chat

Permission for picture use

 dlanzone
Thursday, August 02, 2012 6:35 PM

To: webinfo@tamura-ss.co.jp

To whom it may concern,

My name is Daniel Lanzone and I am a senior electrical engineering student at the University of Central Florida. I am working on a senior design project, and I would like to request permission to use content and pictures from your website. All of the material will be cited properly and referenced back to the website and the location of the website from which I received it.

I appreciate your time,

Daniel Lanzone

Lab 5
katmorales@knights.ucf.edu

pics
dlanzone@knights.ucf.edu

Older

Checking In
herb.gingold@ti.com

Lab 4 Report

sign out dlanzone

Find Someone Options

Actions

6:35 PM

Mon 1:58 PM

Sun 11:35 AM

7/16/2012

7/7/2012

7/4/2012

7/2/2012

6/30/2012

Mail

Calendar

Contacts

Tasks

Staff Comment

2012-08-03 10:36:49 PST

By: Moe R

Thanks for asking.

Yes, you may use the material from the website. Please complete the attached form and return via scan-and-e-mail, mail, or fax, as instructed on the form. Please attribute the quoted material with: "Copyright Maxim Integrated Products (<http://www.maxim-ic.com>). Used by permission."

You may use the material as soon as you send the form (you do not have to wait for reply).

Submit Request

2012-08-02 14:52:53 PST

By: dlanzone@knights.ucf.edu

Hello,

My name is Daniel Lanzone and I am a senior year electrical engineering student at the University of Central Florida. I am working on my senior design project, and I would like to request permission to use two of your figures. These figures will be properly cited, and the document they are going into will not be published. The figures of note are the following:

From	Ryan Tochtermann <tochtermann@gmail.com>
To	info@skytraw.com.tw Add Cc Add Bcc
Subject	Permission to reprint Venus638FLPx Attach a file Insert: Invitation

[Rich formatting »](#) [Check Spelling -](#)

To whom it may concern:

Im a student at the university of central florida and I wanted to reprint the wiring schematic on page 10 of the V638FLPx-D_AP001 data sheet. This is for educational purposes only and it is used to show how we are using the GPS module in our design.

Thank you in advance,
Ryan Tochtermann