Department of Electrical Engineering and Computer Science

EEl-4915 Senior Design 2

Helmet Tracking System

Group #8

Andrés F. Suárez

Brian Maldonado

Rígel Jiménez

Table of Contents

List of Figures

List of Tables

Section 1: Introduction

1.1 Executive Summary

Electrical Engineers have always been recognized as main contributors in the technology sector. All of our group members used this opportunity to build a technological Senior Design Project that improved one of the many activities that a person can perform. There are quite a number of motorcycle riders around the world; so we decided to build a prototype that we named the Helmet Tracking System (HTS), and its main duty was to significantly enhance the motorcycle riding experience. Our plan was to make a modified version of the typical motorcycle helmet, with this motorcycle smart helmet design, the motorcyclist was able to perform a whole bunch of features that made his or her riding experience more fun, secure and sophisticated.

The Helmet Tracking System (HTS) was able to solve many of the problems that a motorcycle rider encountered on the daily basis. This Senior Design Project was composed of many different subsystems that responded according to the user needs. The system had different modules that allowed the biker to interact with his or her personal smart phone, it gave the rider the ability to make and receive phone calls while riding, as well as listening to music, or some other audio recording that the cell phone was able to provide. Finally, the Helmet Tracking System (HTS) also had the ability to track the position of the bike by having a GPS incorporated; the system was able to send information to the user and an emergency contact, in case of an accident.

Choosing a Senior Design project was not an easy task; in fact, it required a lot of brainstorming since we wanted to come up with something that would show our Electrical Engineering skills, but at the same time be useful for the community. The Helmet Tracking System (HTS) was basically an integration project that gave the opportunity to all the three members of the group to learn about different technologies that provide security and communication features which are quite popular in the current market. The project also provided a better insight on how to connect all the subsystems and make them all work as one; by implementing proper integration, coding, and manipulation of a microcontroller or unit to process data. Throughout this document, the reader will see step by step progress as to how the HTS was completed. It will start with some motivational aspects of the project, as well as goals and main function of the project; then it will show an extensive research of the different modules and subsystems required to integrate the main system; and finally, it will show the design of the different parts of the project, as well as the schematic of the full integration of all the parts for the Printed Circuit Board (PCB) design.

1.2 Motivation

Our team wanted to come with a solution to a problem. The problematic situation that we took into consideration was the simplicity of the commonly used motorcycle helmet. According to the National Highway Traffic Safety Administration (NHTSA), there were over 9 million motorcycles registered in 2012 in the United Stated, and the number just keeps increasing. As we can see there is a huge demand for motorcycles, therefore for safety regulations the government has established laws around the different states requiring the use of this protective headgear.

Rígel is not only an EECS student but also a huge motorcycle enthusiast. Riding is an everyday activity for him whether it be commuting, cruising or spirited track riding. Protective gear use is a must for a serious rider, when we think of protective gear three or four things come to mind: a jacket, gloves and the iconic full-face helmet. It is interesting to note that while other drivers on the road, texting, calling and listening to music is not such an out of the ordinary thing to do on an everyday commute but for a rider, these things are never really anything easily accomplished while riding. Headphone use is annoying as they become detached from the ear and wires run from your pockets to the helmet, texting is or calling is not even possible and attempting to even try any of these could potentially prove to be hazardous, if not deadly. Having all this said, we saw the helmet as a potential tool to provide the rider with not only adequate protection but additional features that motorcycle riders miss out on an everyday basis.

All group members decided to pick the Electrical Engineering discipline as a major because we want to be active contributors in this society, and our main motivation is to make this world a better, simpler and more efficient place. We might not be able to solve all the issues that the humanity currently has, but at least we can use our imagination and abilities to come out with ideas, products or projects that can make life somehow better; one example of this is the HTS.

1.3 Goals and Objectives

Our group wanted to learn about many different aspects of technology. One of the reasons we picked this design was to incorporate many different technologies that were available in the market, and at the same time learn as much as we could about each one of them. The HTS allowed us to use many different modules and provide different functions. By designing this project our group had very interesting experiences with the various types of technologies. Below there is a list of the goals and objectives that we achieved with this Senior Design Project.

- The group members were able to add electrical components to the helmet without increasing the weight of the helmet very much.
- The parts and tools to were cost efficient, and made it a worthwhile investment for companies to create and consumers to buy.
- The HTS maintained a simple user interface, and the interface did not require a lot of physical effort, the riding experience stayed as safe and comfortable as possible.
- The HTS made the riding experience safer and more multitasking, the motorcycle owner was able to use his bike as usual, and on top of that, the rider was able to enjoy the comfort and applications that a Bluetooth device was able to provide.
- The project made the riding experience more fun and productive by giving the opportunity to the motorcyclist to do stuff that he or she could not do before.
- The system had a quick emergency response.
- The system was able to text an emergency contact, when an accident occurred.
- HTS used a GPS system to acquire location.
- They system implemented a GSM module.
- A charging system was incorporated to the project.
- It was able to sync the helmet to a cell phone or some other Bluetooth device to make phone calls and listen to music, and it was also able to play music or make phone calls without having a Bluetooth headset or headphones.
- The system was able to send text messages.
- The HTS was able to synchronize to an iPhone and an Android cell phone.
- The system implemented an accelerometer.
- The helmet was kept as spacious as it was originally, even though it had all the extra electronic components.
- It had the ability to received phone calls from the emergency contact.
- A previously manufactured helmet was picked, and it had enough safety characteristics. We had to limit our total spending amount, so we had picked a used one that was in really good condition, and we painted and modified it to make it look great aesthetically.

1.4 Requirements and Specifications

Our design had several features on the helmet that require different types of sensors. A Global Positioning System was used to monitor the location of the motorcyclist. The GPS module kept track of the HTS location at all the time and it provided the location of the bike, if an accident occurred. A sensor or accelerometer module was placed in the helmet, and it had the ability to detect unauthorized movement. The accelerometer was set up in a way, that it had a specific threshold; once the threshold was breached it sent a command to the main microcontroller unit, informing that a crash took place. While the impact happened, the HTS automatically informed the emergency contact via text. Aside from security features the project also had some convenient features for the rider. For that a Bluetooth chip was placed to allow communication using the rider's cell phone. For this system to be complete it also had speakers to provide audio output and a microphone for audio input.

The HTS had its own features. As mentioned before it had a GPS chip to be able to track its location. An accelerometer was placed to monitor for sudden changes in g-force that would signify an accident. This triggered the GPS to send out the latitude and longitude location to an emergency response contact. In the case of possible unauthorized movement of the helmet.

As far as the power source, at first we thought that the motorcycle itself would be used to power the microcontroller and its components, but after doing some testing, the group decided to have an independent power source for the HTS. A voltage regulator was used to make sure each component got the right amount of input voltage. The helmet had a battery pack to power up its own components. The batteries were rechargeable and had a circuit to charge up the batteries so they did not have to be taken out each time.

1.5 Block Diagram

During Senior Design 1, our group thought that it would be a good idea to have two different sub-systems for our project. At that time, the Helmet Tracking System (HTS) was composed by two main sub-systems. The first sub-system had different modules and it was located on the bike; the second one was supposed to work together with the first one, and it was supposed to be placed on the motorcyclist's helmet. Figure 1 below depicts the old block diagram for the motorcycle side of the design. This design was supposed to be driven by the motorcycle voltage supply of the battery and centered about a main control module. This could be a microcontroller or an RF module of some sort depending on the degree of complexity of the processes being implemented. This diagram shows two external communications one that communicates to external emergency contacts in case of an accident for notification and safety purposes, the other external routes to the helmet through an RF medium of communication. The RF establishes the communication link to the helmet to send and receive data.

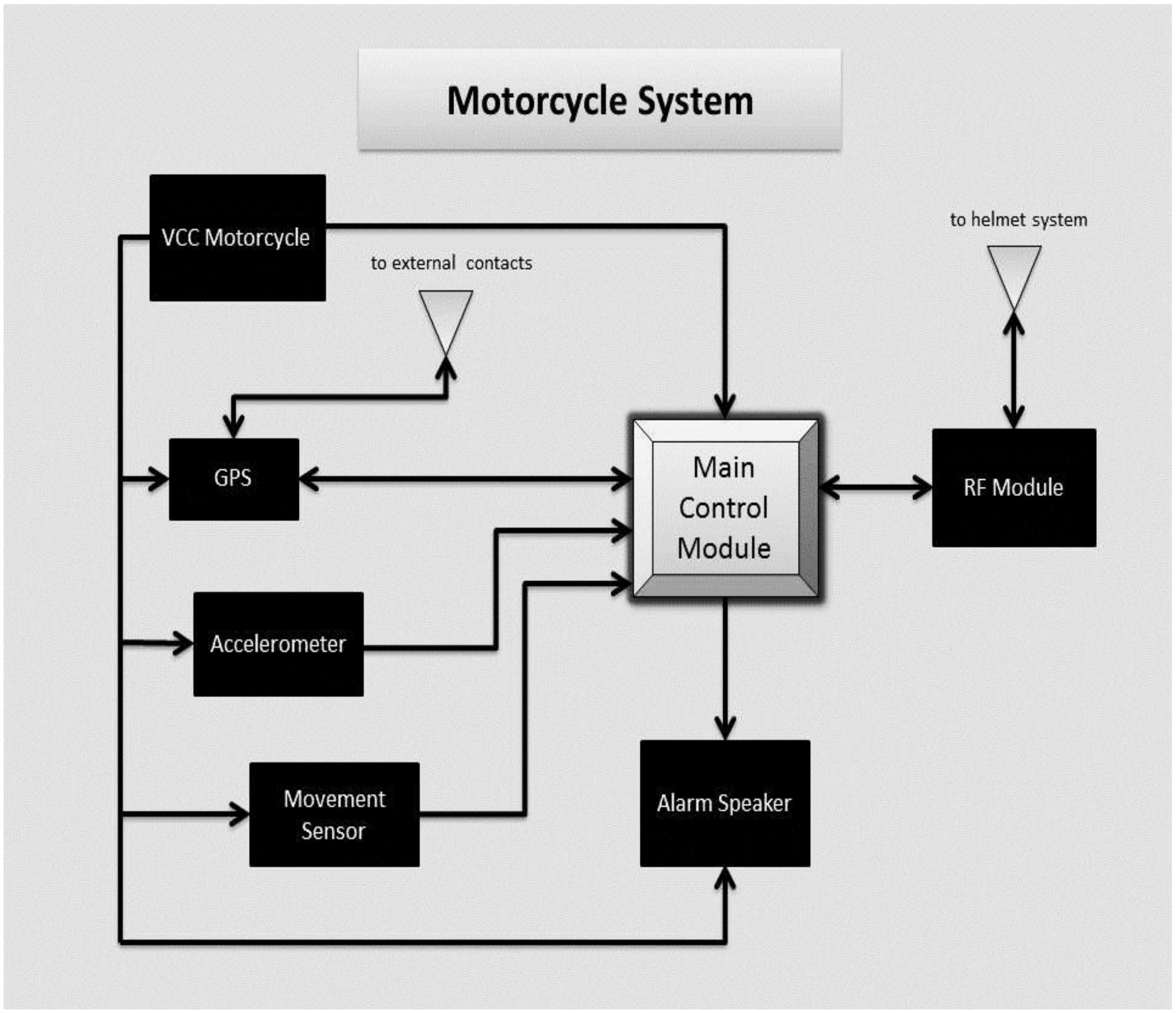


Figure 1: Block Diagram of the old design of the Motorcycle Sub-system

Figure 2 below depicts the block diagram from the old helmet design's perspective. This was the system that we wanted to implement around and within the helmet. Similarly to the motorcycle's block diagram this block diagram contained familiar elements including a secondary control module and further communication systems between the helmet and externals. The three communication peripherals included communication between the helmet and the user's cellular device via Bluetooth. The second peripheral was supposed to be connected to the motorcycle's control unit, which is the medium of data transfer between them.

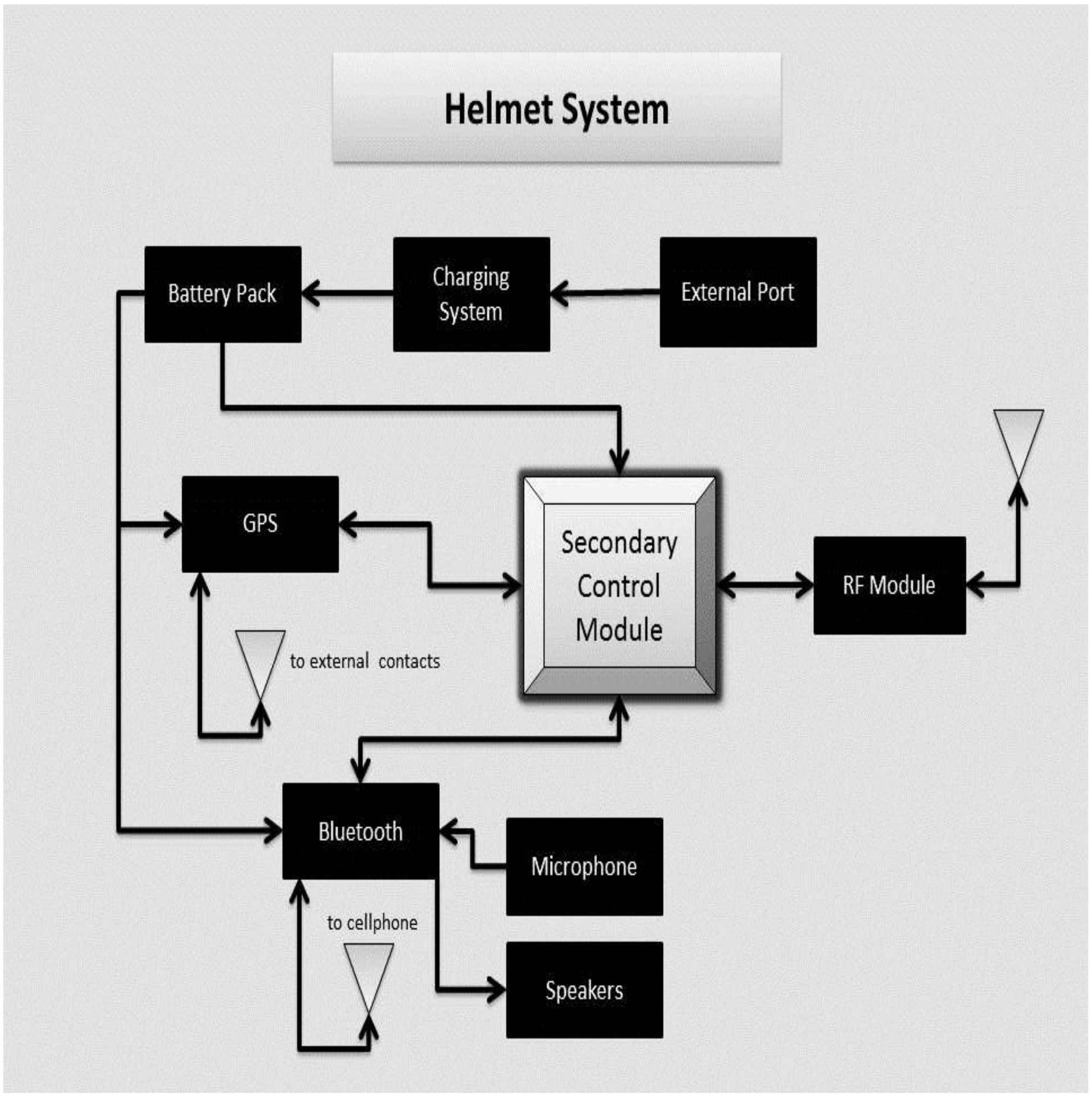


Figure 2: Block Diagram of the old design of the Helmet Sub-system

Once, we started working on the prototype of the Helmet Tracking System, we decided to merge the motorcycle and the helmet sub-systems into one. After brainstorming for a while, and doing different designs, our group members decided to put all the main modules on the helmet, so pretty much we had all the main features without the need of having to Printed Circuit Boards (PCB's), two microcontrollers, and an RF System. The block diagram shown in Figure 3 shows the final design of the HTS.

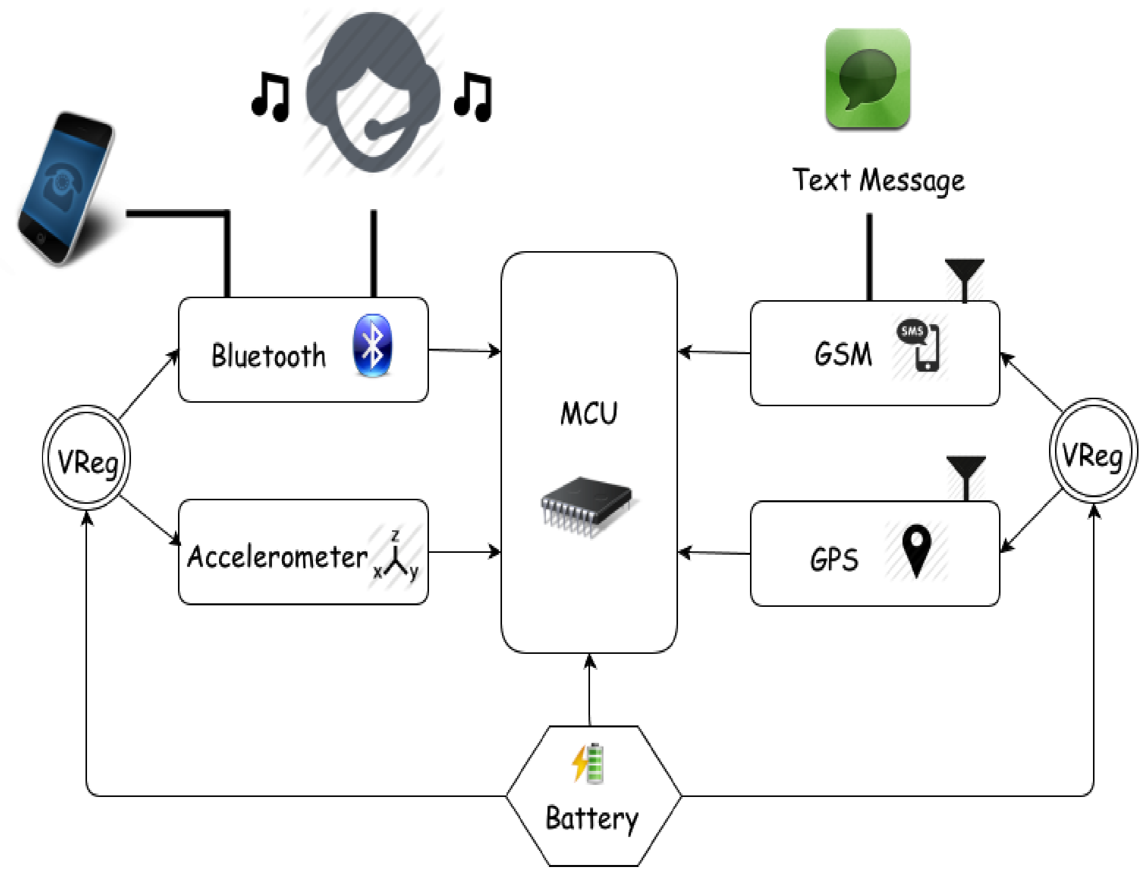


Figure 3: Block Diagram of the final design of the Helmet Tracking System

1.5.1 Distribution of Workload

The HTS was a project based on teamwork. A proper and well-balanced workload distribution was essential to obtain optimal results. Table 1 shows the topics selected by each member of the group.

	Rígel	Andres	Brian
MCU	X		
Software	X		X
Bluetooth Module		X	
GPS Module		X	
GSM Module	X	X	
Accelerometer Module			X
Battery and Recharge			X

Table 1. Workload Distribution

1.6 Project Function

The Helmet Tracking System incorporated a more technological version of the original bike helmet. Our design allowed the user to have access to Bluetooth technology without the use of a Bluetooth headset. As we mentioned before one of our group members was a motorcycle owner, and he tried many time to use his headphones or headset while riding, and they always fell off, also the reception was not that good. The Bluetooth incorporated in the helmet gave access to the user to a new amount of features that originally did not have while riding. The motorcyclist was now able to sync the helmet to his or her personal cell phone or electronic device with Bluetooth capability in general, depending on the device used, the person was able to make and receive phone calls, listen to music, or perform some other applications that used audio.

We also had to implement GPS technology to obtain the HTS location and to get information of the places we went. This helped us improve the riding experience. A microphone and a good audio system were added to the helmet too, so the user could efficiently interact with a cell phone. Our design had the ability to interact with iPhone and Android devices, which are the most popular cell phones in the market right now.

Most of our circuitry was located on the rear part of the motorcycle helmet, this designed allowed us to keep the helmet as specious as possible, which was one of our goals. All of our modules were placed on the PCB located on the back of the helmet, and this allowed a better reception from satellites and other external wireless sources. The PCB was relative small, so the helmet stayed as light as possible to keep its weight about the same.

The HTS basically could be synchronized to a Bluetooth device at any time; and it also had the ability to text an emergency contact, thanks to a GSM module that was added to the PCB. Pretty much, every time the threshold was breached, the accelerometer sent data to the microcontroller, and then the microcontroller woke up the GPS, and retrieved the location, and after that it texted the emergency contacted the latitude and the longitude, of the place where the rider was.

Section 2: Research

2.1 Existing / Related Projects

When we decided on this project to design we knew that there were many different variations out there and had to make ours more unique then what is out in the market already. We made it unique by either adding extra features that are not commonly found in these types of products or by combining features you found in different products into one. Making our design stand out and be different from anything that is available on the market. Security system products are readily available in many different forms. It is an area of market that has been targeted for many years since keeping things safe has always been a concern for belongings. That includes motorcycles, because we don’t live in a perfect world there will be people out there that are going to try and steal it and people want security to keep that from happening. There are a lot of systems out there so one of our goals is to make it one of the most advanced and most effective.

There was only one company that had a product very similar to what we made in this Senior Design project, for which made it that much easier to make ours more unique. The company Phantom Tracking offered a similar product to our design. It was a GPS motorcycle tracking and security system that offered protection to the motorcycle and safety to the rider. If the motorcycle was stolen even with the alarm on, it could track the location of it through GPS making only a matter of time before the thieves were caught. It came with crash detection, motion sensing, GSM/GPRS messaging. Some of these features were also featured in our design but where ours stand out that it’s the complete package where in not only included the motorcycle and rider, but it included the helmet as well. And by the complete package we meant all components needed for riding were protected with some kind of technology that we were implementing. The Phantom Tracking covers overall the motorcycle and rider but not the riders' helmet. Our project offered communication to the helmet for safety and security.

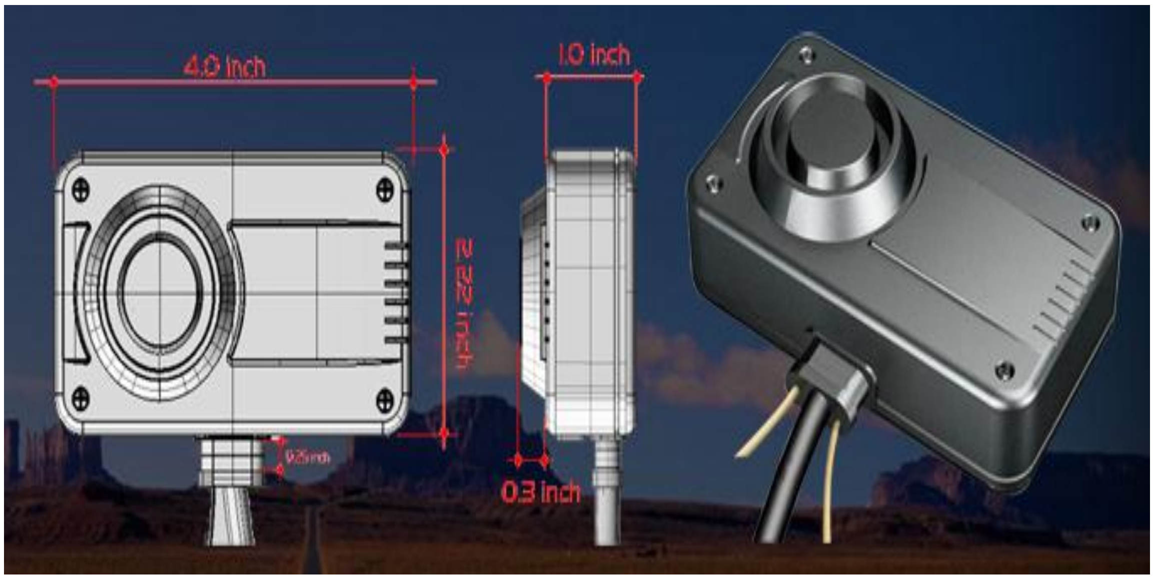


Figure 4: Phantom-Tracking System

(Posted with permission of phantomtracking.com)

How the Phantom Tracking focused on the motorcycle, there were many other products that focused only on the helmet. Implementing technology in a helmet has been a very popular demand especially since technology is becoming more and more advanced every year. The biggest implementation to be put in a helmet was Bluetooth. It allowed the rider to be able to communicate using the Bluetooth from their phone. Another feature that Bluetooth provided was the ability to listen to music while riding.

One company that was leading the competition in this was ChatterBox. They had several different type of product that center around Bluetooth technology. Their product was simple and small enough that it attached to the side of the helmet. Then once it was connected with the riders phone it could make phone calls, listen to music, and got GPS directions all wirelessly. Another cool feature they had is that it could communicate with other riders that have the same product with the same capabilities. Now where our design stood out was with extra features. The Bluetooth features were pretty much the same since our device had the same technology. GPS was an extra feature for the helmet that will be embedded in it. This was to protect your helmet from theft. With GPS the helmet could be tracked down to its location. It also had a GSM module to be able to transmit its location when needed. With these extra features it made our design different and unique from what is out in the market today.

Figure 5 Chatter Box

(Posted with permission of chatterboxusa.com)

2.2 Wireless Communication

In our current era wireless communication devices are pretty much everywhere; in fact, to successfully develop this Senior Design project, the use of this technology was essential. But realistically, even though the term sounded familiar, and as it was mention before, almost every single person interacts with one of these devices in a daily basis, there was a lack of knowledge from our part on how to build a communication system that implemented this technology, and on the different options that were in the market that could help build the HTS and add it to the motorcycle helmet. First of all, it was important to define this term; wireless communication is the technology exchanging of information between two or more points that are not connected by an electrical conductor. This kind of technology was a must have for the project because one of the main features of the helmet was the ability to connect to a cell phone, in order to make phone calls and listen to music. The research that follows looks into some of the most popular methods that were available for this project. Some of the wireless communication technologies that were looked into were Wi-Fi, Infrared, Bluetooth, and RF communication. Also, some key factors that were taken into consideration were the data rate, the range, and connectivity to the cell phone brand and model that the project was using.

2.2.1 Wi-Fi

Wi-Fi was the first protocol taken into consideration for this project, so let's go ahead and define it. In general, Wi-Fi is a very popular technology that allows transferring data wirelessly using radio waves over a computer network, including high-speed Internet connections. Wi-Fi stands for Wireless Fidelity, but it is also known as wireless local area network (WLAN), and it is based on the Institute of Electrical and Electronics Engineers (IEEE) 802.11 standards. Wi-Fi operates in the 2.4 GHz or 5 GHz radio bands, and it also provides a well-established connection and this is very helpful because it compensates network congestion and it helps minimize the errors on the connections. This technology is very popular among computer networks, as well as in video game consoles, cell phones, personal computers and tablets.

The main goal of Helmet Tracking System (HTS), which was designed by our group, was to make the riding experience safer and at the same time more enjoyable. Being realistic, who would not like to be able to ride a bike and at the same time listen to music, make phone calls and have security features. Being able to make the system wireless was a key factor in this design, because a person would not be comfortable driving a bike, while having a whole bunch of wires hanging, this could produce a hazardous situation. Wi-Fi is pretty much everywhere, so that is why it came to our mines as soon as we talked about implementing wireless communications, but what made it appealing to this project. A fast transferring of data was a factor that could make the performance of the design better, and among all the wireless technologies, Wi-Fi had the highest data rate transfer; in fact, its performance was very similar to a wired connection. Not only that, this protocol was extremely secure, and this was a feature that we wanted because we were sending personal information such as the location of the motorcycle rider, in case of an accident.

This project tried to keep a low consumption of power, so let's see how was the power consumption for this technology. The average power consumption for this technology was the highest compared to other wireless methods when transmitting over long distances. This was something that we really had to take into consideration because what was the point of creating a motorcycle and helmet security system that was very secure, but consumed a lot of power. If a Wi-Fi enabled transmitter was implemented in the senior design project, the battery life of the device could be highly affected, and that was something that a user would not like. The customers or users of any electronic device always want a product that is reliable and has a good performance.

Advantages:

- Wi-Fi technology provided a long-range of coverage.
- It had the highest data range. Its performance was very similar to a wired connection.
- Its signal was not easily blocked.
- It had a low cost compare to other technologies that offer a similar performance.
- It worked on the unlicensed 2.4 and/or 5 GHz range.
- It was very easy to integrate with a Smartphone. Even though the HTS is limited to iPhone and Android cell phones.
- This technology was quite popular in the market because it was very secure.

Disadvantages:

- It required high power consumption, which was something we could not afford
- Slight signal interference might occur.
- The configuration of a network based on it could be complex.

After understanding this wireless communication method better, it became clear to our group members that the implementation of this technology would not be such a great idea, because we were trying to minimized the power consumption as much as possible. But, this protocol could not be completely rejected, it also had some good features such as the high security levels, and we thought we might consider it as a back up plan later on when we were actually building the prototype, but this was not the case. Our knowledge about wireless communication increased after this, but we needed to keep researching about some other methods that we thought could be used in the project.

2.2.2 Infrared (IR)

Another wireless communication method available in the market was Infrared (IR). This protocol has been out there for quite a while, and we daily interact with products that use it. This technology allowed the transmission of data within a short range using infrared beams. Its current applications in the Electrical Engineering field showed us that it could also be an option when it came to building the helmet tracking system. Infrared was widely employed among computer peripherals and personal digital assistants. It was important for us to know that IR was limited to only two devices at the time; also, they needed to be in a direct line of sight so that the devices could detect each other and communicate.

Advantages:

- It had low rates of power consumption.
- Decent speed, it had a transmission speed up to 16 Mbit/s.
- It offered a safe transmission.
- The technology has been used for a long time, so it has been studied a lot, and there was a lot of information about it.

Disadvantages:

- The connection was limited to a short range, but in our case, we could deal with this.
- It was limited to only connect two devices at a time.
- It required a direct line of sight to operate properly.
- It did not go through objects, which was something that we needed to implement.
- It could produce a bad signal or the connection could be interrupted due to many factors such as: the distance between the two objects that were being connected, a wrong angle, noise, heat, or light waves.

Infrared was found to be successful in many electrical engineering projects, and it was a well-respected technology because it has been in the market for quite a while. Unfortunately, according to its characteristics, it had a lot of disadvantages that could make our project very inefficient when it came to performance. Even though it was able to transfer data in a short range, which was what we needed, it had a lot of weak points. We needed to have a reliable wireless communication module that was able to transmit the signal in an efficient way, so the motorcycle rider would feel confident and satisfied with the HTS.

2.2.3 Radio Frequency Transmission

According to webopedia.com RF or Radio Frequency was referred to as “any frequency within the electromagnetic spectrum associated with radio wave propagation.” Usually RF transmission follows standard electrical signal propagation format. An initial RF current is supplied to an antenna, which in turn creates an electromagnetic field that then begins to propagate through space. It was interesting to know that this simple idea is the most basic grounding foundation for many new modern wireless technologies out in the field today.

It should be noted that there were also different classifications of radio frequency oscillations. These range from ELF or extremely low frequency to THF with frequencies of 3-30 Hz and 300-3000 GHz respectively. As noted there were just too many different forms of frequencies propagating through the air and in turn this was where the use of a radio tuner came in. The radio tuner could enable us to tune into a particular frequency of preference. The tuner was usually a pretty simple resonating circuit (e.g. an LC circuit, which forms a filter and in turn helps with forming a pass-band and a rejection band to filter out unwanted frequencies and keep those that are of interest). Frequencies within our pass-band were accepted while all others were attenuated.

Perhaps one of the most important communication systems being used in our design could be the RF transmission module. We were faced with a variety of communication systems in our design. The use of Bluetooth was primarily useful for voice commands, calls, music and synchronization with the user's cellular device while the GSM chip was primarily useful for text messages and notifications. Aside from those two systems we had one of the most integral communication system involved in our design, the RF frequency transmission module. In essence, we wanted to dedicate the RF module to be used as a means of communication between the motorcycle and the helmet, but as the system was simplified, this method was disregarded.

If we go further into specifications we had a few parameters to consider and tailor to cater to our specifications. We did not want to have too much power in our hands or too little in which case we needed to take weight of what we needed and exactly just how much of it we needed. For our wireless module we were mainly concerned with range of transmission, data rate and network acquisition time. We desired to run our project in real time; therefore network acquisition served an important role.

One of the ideas that we had was that the rider could have a remote to control certain parameters of our system. The range did not need be very long, as ideally key-less systems were not designed for long-range applications. The idea here was that if we provided the user with a remote to control the system, we would intend it to be utilized in a very similar fashion to that of a key-less FOB remote of modern cars today. After testing, we decided to not use this idea because it was unnecessary for the HTS.

Following that, we ran into data rate, small packets could be sent back and forth between our nodes therefore data rate did not need to be very fast for effective use of our module. Low data rate was sufficient for the remote aspect of our project however for our Bluetooth module, data rate was more of a priority as opposed to range due to the fact that we were dealing with commands, phone calls, music streaming and general status notifications.

Different wireless technologies were considered when it came to choosing which fit the project best. There were just so many out there which many could be adapted to suit our project specifications however the group discussed a few options some more complex to implement than others but nevertheless feasible options. These few were Bluetooth, wireless USB or a ZigBee/Xbee module. Table 2 below shows a comparison chart between these modules.

Specification	**Wi-Fi**	**Bluetooth**	**ZigBee**
Data Rate	54 Mb/s	3 Mb/s	240 Kb/s
Range	100m	100m (class 1)	100m
Networking Topology	Point to Hub	Ad-Hoc	Ad-Hoc/PTP/mesh
Operating Frequency	2.4 GHz	2.4 GHz	2.4 GHz
Power Consumption	High	Medium	Very Low
Network Acquisition Time	3 – 5 s	< 10s	30ms

Table 2. RF Transmission Technology Options

2.2.3.1 ZigBee Module

On the same footnote, ZigBee was a module that really caught our attention because we thought it could be a good match for our project. ZigBee was a protocol that could be directly embedded into various applications at the general standard 2.4 GHz operating frequency. ZigBee was similar to Bluetooth as it can use radio frequency and had the ability to link to multiple nodes. It had of course low power consumption, which made it appealing due to the limited space in our design. ZigBee as well as XBee were considered for this project. ZigBee's parameters were reasonable to us as it is low power, low network acquisition time and low data rate. We decided to use an Arduino board, which did not require high data rate, so this made ZigBee a pretty good fit for our project.

ZigBee featured Carrier Sense Multiple Access/Collision avoidance mechanisms, which allowed communication to be very efficient and free of collisions for the most part. The interface would only send out single signals to the board only when user input was detected. Both nodes would be in sleep mode for most of the time until mutual acknowledgement was received. After that, nodes could communicate and return to a dormant state. ZigBee protocols tended to reduce the time devices were in active mode in order to manage power as efficiently as it did. Its low network acquisition of 30ms made it a perfect candidate for real-time applications.

Further scrutinizing revealed that ZigBee provided numerous options and features. Perhaps we could consider not only using one, but two of these modules to implement additional features or further optimize our project to increase efficiency and processing. Parameters considered here were frequency, RF line of sight range, transmission power, receiver sensitivity, data rate and cost. When these were taken into consideration we were able to narrow down some more possibilities for our RF modules. It should be noted that all of these also require a line of sight, which means that communication might be jeopardized by intervening objects. In the table below are the three different candidates found suitable for our project.

Specification	**XBee 802.15.4**	**XBee ZB**	**XBee Digi Mesh 24**
Topology	Point-to-Multipoint	Mesh	Mesh
Frequency	2.4 GHz	2.4 GHz	2.4 GHz
RF Line of Sight Range	90m	120m	90m
Transmission Power	0 dBm	3 dBm	0 dBm
Receiver Sensitivity	-92 dBm	-96 dBm	-92 dBm
RF Data Rate	250 Kb/s	250 Kb/s	250 Kb/s
Cost	$19.00	$17.00	$19.00

Table 3 ZigBee Wireless Module Specifications

2.2.4 Bluetooth

Now let see some of the information that we found about Bluetooth, which was one of the most popular wireless communication protocols in the market. One of our goals for this senior design project was to be able to transfer data wirelessly within a short distance, and this was basically the primary goal of the Bluetooth technology. This technology exchanged data using short-wavelength radio transmissions in the industrial, scientific and medical (ISM) band from 2400-2480 MHz from fixed and mobile devices; and it also created personal area networks (PANs) with high levels of security, which was something that we really liked to implement in our project. We wanted the user to feel as secure as possible when synching his or her devices with the HTS.

This was a wireless technology that enabled communication between Bluetooth compatible devices, and this was not a limitation because currently almost every cell phone or electronic device has a Bluetooth module on it due to its high efficiency and security features. The IPhone 5, which was one of the products that we were synchronized to the helmet, already had Bluetooth built-in and it offered an easy synching process, which was something that the motorcycle rider really liked when using the HTS. Bluetooth was used in cell phones to send files between two or more phones, or from the cell phone unit to a computer and vice versa. It was also used to connect to headsets or wireless earpieces, which was something that needed to be incorporated in the project. The helmet had an earpiece and a microphone to communicate to the external devices that were being synced. One thing that we took into consideration was the fact that was very common to use the same headset to connect with different Bluetooth devices, and this was definitively a feature that we wanted for our project. We limited the types of cell phones that could be synched to the helmet, but we still wanted it to sync to more than one device. In this case, multipoint pairing was essential. In order to achieve multi-pairing an industry standard color code was needed, this allowed the device to distinguish one pair from the others.

As it was mentioned previously, analyzing the range of coverage was important to choose the right wireless protocol for the project. Bluetooth was a protocol that was mainly designed for low power consumption, and the range changed according to its power class type. The different Bluetooth classes and its corresponding ranges are shown in Table 4 below.

Class	Maximum Power	Operating Range
Class 1	100mW (20dBm)	100 meters
Class 2	2.5mW (4dBm)	10 meters
Class 3	1mW (0dBm)	1 meter

Table 4: Bluetooth Classes

Table 4 clearly showed that a Bluetooth device of class 3 had a very small operating range, it only covered 1 meter; this limitation of range made the class 3 very unpopular and rare in the market, so class 2 and 1 were the most often purchased by designers that decided to transfer data via Bluetooth.

After selecting a range for a specific project, it was indispensable to use Bluetooth devices that had the same range. For example, and iPhone is a device that has a class 2 Bluetooth unit, so in order to successfully connect it to an external Bluetooth module, a class 2 unit needed to be chosen. They both provided the same range (10 meters) and maximum power (2.5 mW), and this would generate the desired output. When a motorcycle rider is on the road, most likely carries the cell phone used to sync with the helmet on his or her backpack or pockets, so the data only needed to be transferred across a short distance; in fact, in case a class 2 Bluetooth module was chosen the range would not be an issue because 10 meters were enough to complete the tasks that were being implemented in this design. So far this protocol seemed appealing to the project because most likely the project was limited to Apple IPhone, and Android cellular phones. Both of these cell phone brands that were being used to sync with the project already had a Bluetooth module incorporated, and this facilitated the syncing process.

<u>Advantages:</u>

- It had a very low power consumption compared to other wireless protocols.
- Did not require straight line of sight to transfer data.
- Little radio wave interference.
- Spread spectrum frequency hopping.
- Already incorporated in a wide range of devices.
- It had many robust profiles.
- It had the ability to penetrate surfaces.
- It was already incorporated on the cell phone used in the project.

- Bluetooth used a standard frequency, which allowed all the Bluetooth devices to be compatible with each other.
- “Piconets” were easily configurable. A Piconet is a network that is formed when to or more Bluetooth devices are connected to each other, one Piconet can have up to 8 devices.

Disadvantages:

- It covered a very low range, but this was not a issues in our case.
- It had relative slow data transmission speeds. The transmission speed was fixed based on the Bluetooth device that was being used. Almost all Android cell phones and IPhone’s had at least Bluetooth 2.0 + Enhance data rate (EDR). The nominal rate of EDR was about 3 Mbit/sec.
- It had low penetration qualities.
- On cluttered 2.4 GHz ISM band.
- The price of a module was not very cost effective compared to other protocols.

2.2.4.1 PAN1321i Bluetooth Module

Selecting the right module for Helmet Tracking System (HTS) facilitated the integration and made the performance more effective. The PAN1321i made by Panasonic was the first module that was going to be analyzed. This was a Bluetooth RF module compatible with Apple devices such as the IPod, IPhone, and IPad. This was possible because the module interfaced with the Apple authentication coprocessor and supports IPod Accessory Protocol (IAP) to enable Bluetooth Serial Port data communication with Bluetooth enabled Apple devices. As we mentioned before, we were mainly planning to sync our project with and IPhone 5, so this would be perfect. At the same time, it was very important to notice that the module was also compatible with Android, and most Bluetooth enabled devices currently available.

The PAN1321i was used on Smart phones, for proximity, in Heart Rate applications, Generic I/O, HVAC, battery monitor, and in most wireless applications. It also had a reasonable price of around $27.00, which was acceptable for our budget. Figure 6 illustrates this module.

Figure 6: PAN1321i

(Printed with permission of Open Source)

The figure above displays how the module looks like, but it does not really show its actual size. The unit was small, to be more precise its dimensions were 15.6 x 8.7 x 2.8 mm^3, which was a good size for our project. We knew that was compatible with devices that we wanted to connect to our senior design project, and we also knew that its size was acceptable; but we needed to know some more about the Bluetooth module specifications. This information is summarized on the table 5, shown below.

Parameter	**Value**	**Condition / Notes**
Receiver Sensitivity (BER = 10^-3)	-86 dBm	Ideal signal
Output Power	+ 3 dBm	@ 50 Ohm antenna pin
Power Supply	2.7 - 3.6 V	Single operation Voltage
Ultra Low Power Scan	80 uA	T=25 degrees C
ACL (Transmit 3-DH1)	40 mA	Enhanced Data Rate, 531.2 kb/s
ACL (Receive 3-DH1)	37 mA	Enhanced Data Rate, 531.2 kb/s
Operating Temperature Range	-40 to 85 degrees C	

Table 5: PAN1321i specifications

2.2.4.2 WT32 Bluetooth Audio Module

The main purpose of our Bluetooth module was to transfer Audio signals between the user's cell phone and the helmet, and vice versa. WT32 was a *Bluetooth* 2.1 + EDR module targeted for *Bluetooth* audio applications. In addition to *Bluetooth* radio, antenna and iWRAP *Bluetooth* stack, WT32 contained a DSP processor, a stereo audio codec and a battery charger that made it ideal for portable battery *Bluetooth* stereo or hands-free audio applications. The figure 6 shows the outside of the module.

Figure 7: Bluegiga WT32 Printed with permission of Open Source

This was a class 2 module, so according to our previous research it had the basic requirements that we needed to accomplish our goals, as far as, wireless communications was concerned. Here is a list of its main features:

- Plug and play Bluetooth solutions for mono and audio stereo solutions.
- Integrated Digital Signal Processor (DSP), stereo codec and battery charger.
- Integrated antenna.
- Class 2 module.
- Industrial temperature range from -40 C to 85 C.
- Low Power Consumption 1.8 V operation, 1.8 V to 3.6 V I/O.
- IWrap Firmware for controlling connections and configuring settings.
- Ten software compatible IO pins.
- Dimensions: 15.9 x 23.9 x 2.5 mm.

2.2.4.3 RN–52 Bluetooth Module

The previous two modules seemed to be good options for our motorcycle and helmet system. But, we also wanted to take a look at the RN-52 audio module because it had some good features for a very reasonable price; it could be purchased for as low as $24.95, and it was also a class 2 module that was compatible with the IPhone and the Android devices, and it provided audio capabilities, which was something that not all the modules could provide. We also liked the fact that it Supports multiple Bluetooth profiles such as SPP and HID and simple UART hardware interface, and it was simple to integrate into an embedded system or simply connect to an existing device, which was our case. The RN-52, illustrated in Figure 8, was FCC and Bluetooth SIG certified, and this made it a complete embedded Bluetooth solution.

Figure 8: RN-52 Module

Printed with permission of Open Source Sparkfun.com

Here is a list of the main features that made this device a good option for our project:

- Fully qualified Bluetooth module.
- FCC certified.
- Fully configurable UART.
- Dual-channel, differential input and output.
- Support iAP profile.
- Low power sleep mode.
- Maximum over air data rate of 3.0 Mbps.
- Compatible with all Bluetooth products that support SPP.
- Embedded Bluetooth stack profiles: A2DP, AVRCP, HFP/HSP, and SPP.
- 3.0 ~ 3.6 V operation.
- Bluetooth Technology v3.0 compatible.
- Class 2 power output.

2.2.4.4 Helmet Wireless Communication Module

Our group had a better understanding of what wireless communication was all about; so it was time for us to make a list of the requirements that we needed to meet for our project. Let's start with the range of coverage; most of the time the driver of the motorcycle would riding while using the different features of the helmet, but we also wanted the user to be able to connect with the electronic devices and have a good reception, even when they were a little far. Usually the cell phone was located inside of the rider's pocket or in a backpack so the distance from the cell phones to the Bluetooth module was not that much. We were not sure yet, if the module would located underneath the driver's seat, or it would be inside of the helmet; but one thing was for sure, we needed to select a component that was able to cover a range between 0 to 10 meters, the distance between the cell phone and the unit was less than a meter when the person was riding, so this would reception if the biker was walking on an area close to the bike, or in case the user fell off the bike, the module would still have signal, and it was be able to send a text message to an emergency contact to report an accident.

The next aspect we analyzed was the data rate. We wanted to make a product that was able to transfer data at least a rate of 2 Mbps, but if we could get one `

Let go back and analyzed Wi-Fi a little bit, we found out that it was a technology that offered a very fast transfer rate; actually, it was almost like using a wired connection, which was something that really caught our attention; unfortunately, we had one big issue with it, and that was the fact that it required a high power consumption, and that was something that we could not afford to have in this project; therefore, we had to let it go. After that we took a look at Infrared, and we were really impressed with the amount of information that there was about this technology, it was very solid and well known. We really liked the fact that was really good as far as power consumption, but we were really disappointed when we found out that the signal transmitted needed a straight line of sight to operate properly; and not only that, the signal could be interrupted or lost very easy. Even if the two devices that were connected were misplaced, the signal would be lost because they needed to be in a specific angle to have a clear and successful transmission. This was a big issue for our project because the motorcycle will be in motion and there was a lot of interference. Also, if we selected this technology the helmet would be limited to only synchronizing to two devices, and we wanted it to be able to sync to a few devices. As we previously stated, we decided to select Bluetooth because it was the wireless communication technology that seemed to be more well-rounded; it was not perfect, but according to the information obtained, it did match with the requirements that we were trying to meet, and it was also very popular, so we were able to choose from a variety of products and classes, and we had a lot of good information online about the building and integration of the unit in our own system.

To conclude with this part we wanted to reiterate that the decision was made after taking different things into consideration such us: range of coverage, security levels, data rate transfer, reliability, efficiency, power consumption, availability, compatibility with electronics devices being used to synch with the module, and the ability to send the signal through objects. The research really helped us because it incremented our knowledge about this topic and it helped us pick the wireless communication technology that seemed to be more appropriate for our senior design project. We still needed to look at the different Bluetooth Modules available in the market, to find out which one was the most adequate to use in our prototype; so far we did not know much about the Bluetooth products available, but we did know that we most likely had to get a class 2 Bluetooth module according to the characteristics analyzed before in the Bluetooth section. In the design section of the project, we will carefully examine the module selected for the HTS.

2.3 Global Positioning System (GPS)

Being able to locate a specific place, or to provide a current location has become a very popular feature among the newest and highest technological apparatuses; and of course this was one of the features that we wanted to have in the Helmet Tracking System (HTS). For our project, we needed to know the location of the motorcycle rider because this was a very important piece of information for the emergency contacts, in case of an accident. If the user was injured and it was not capable of making a phone call, we were planning on texting an emergency contact with the biker's location. But what technology could help us do this? Well, this answer to this question was summarized in three letters GPS. The technology known as GPS stands for Global Positioning System is a space-based satellite navigational system that allows land, sea, and airborne users to determine their exact location, velocity and time 24 hours a day, and it works under any weather conditions, anywhere on or near the Earth where there is an unobstructed line of site to four or more GPS satellites. In order to acquire the motorcycle location, we had to add a GPS receiver to our project. This receiver was integrated to the Helmet Tracking System (HTS), and it was next to the user at all the time, the primary duty of this receiver was to receive the data from the satellites, and then it used it to solve the navigation equations shown below.

$$p_i = \sqrt{(x-x_i)^2 + (y-y_i)^2 + (z-z_i)^2} - bc, i = 1,2,\dots,n$$

In the equation x, y, and z were the component of the satellite position and the time sent are designated by the i values in all the three directions. In our case, the result of these equations was the location of the receiver module that we were going to place on Helmet Tracking System (HTS). We were not going to fully analyze the Navigation Equations, but reading about them, and seeing some mathematical exercises online helped us better understand how a GPS works. We are Engineering Students, and we based our knowledge in Math. Now that we knew, how a GPS receiver works, and how it obtains the data to provide the location that we were going to use for our project; we wanted to analyze the different type of receivers that were available. The first type was the multiplexing receiver; this receiver used time division multiplexing to gathers data by switching at a high frequency, and it could track multiple satellites at the same time; another characteristic of the multiplexing receiver was that it used a single, dual or even multiple receiver units, which provided more options when it came to the hardware design. The second type was the parallel receiver; it had dedicated hardware to receive satellite information that was needed to determine a specific location. The last type was the sequential receiver; which was not very popular anymore because of its not so good performance compared to the previous two types. The sequential receiver switched and obtained information from multiple satellites, but it did not work in the same way that the multiplexing receiver did. The sequential receiver gathered all its data from one satellite and then it moved to the next one to continue to collect data to provide location. Among the three types of GPS receivers; the parallel and the multiplexing seemed to be the more reliable ones, we definitively did not one to implement sequential because we wanted to have an efficient project. To find the location of the rider, the GPS module needed to have access to at least 4 satellites at the same time, so the unit that had access to more satellites signals would be able to provide the more accurate data; this was a little bit easier for the multiplexing receiver because the parallel required more hardware.

To select the right GPS module some other features needed to be analyzed. These features were: size, update rate, power requirements, and number of channels, antennas, and accuracy. As far as the size, we needed to pick a module that was relative small, but we did not want to get something extremely small because it required a tiny antenna, which affected lock time and accuracy. Our project required at least an average update rate, because our helmet system was design to be on the road most of the time; the standard for most devices is 1 Hz (only one per second), and this was relative enough for our project, there were modules out there that provided a faster rate around 5 or 10 Hz, which was also good, the only problem was that they will require our microprocessor to process more data, the good thing about these faster modules was the fact that they gave more accurate data and they could usually be configured to run at an easier pace. Let’s take a look at the power consumption; on average a GPS module uses around 30 mA at 3.3 V, so we had to get something around that range, or if we could find something with a lower power consumption it would be phenomenal, but if the power consumption was very low, we had to make sure that the unit had an antenna attached to it. There were 35 systems of navigation satellites in space, and 24 of them were active at any given time; as we can see there were not that many, so the number of channels that the module ran affected the time to first fix. The GPS receiver did not know which satellites are in view, so the more frequencies that you could check at once, the faster you found a fix. On average, 12 to 14 channels worked fine for tracking, but if we wanted a faster lock, we needed a module with more channels. As far as the antennas, most of the modules came a chunk of ceramic on top, which was the antenna. Each antenna was finely trimmed to pick up the GPS L1 frequency of 1.57542 GHz; we tried to get a module that already had one incorporated to facilitate the integration. Finally, we talked about the accuracy, according to our research online, in average you could easily find the position of the module, any where in the world, within 30 seconds, down to + / - 10 m. But most module could get it down to + / - 3 m, which was a perfect for our project. The plus or minus sign stated that it could more or less vary according to the module, time of the day, clarity of reception, location of the module, etc.

Now that we had a better knowledge about the Global Positioning System technology, and the different types of receiver available in the market; we could confidently say that we could incorporate this technology into our project. As a matter of fact, we could implement a module that was extremely fast and accurate; but our only limitation was the amount of money that we were willing to spend. That did not mean we were going to get a bad unit, we did our best to select a unit that was reliable, effective, accurate, had a small size, a good update rate, good power requirements, and that of course was within our budget. In the following sections of our research we will look at few models of GPS receiver modules that seemed to be appealing to our senior design project.

2.3.1 D2523T Helical GPS Receiver

Figure 9 shows the D2523T dimension. This was a compact GPS smart-antenna engine board, which came equipped with a Sarantel GeoHelix high-gain active antenna and GPS receiver circuits. The module was based around the high performance 50-channel u-blox 5 platform, and its omni-directional antenna provided great sensitivity, even when you did not a have a clear view of the sky. This module could be really good for our project because it provided a very high and accurate performance, while keeping low power consumption. The only bad thing about was its price. It cost about $79.99, and even though it was not the most expensive module, there were quite a few that could do the task for a lower price. The helical antenna, which was not the basic model for antennas, was one of the reasons that made this receiver a little bit more expensive.

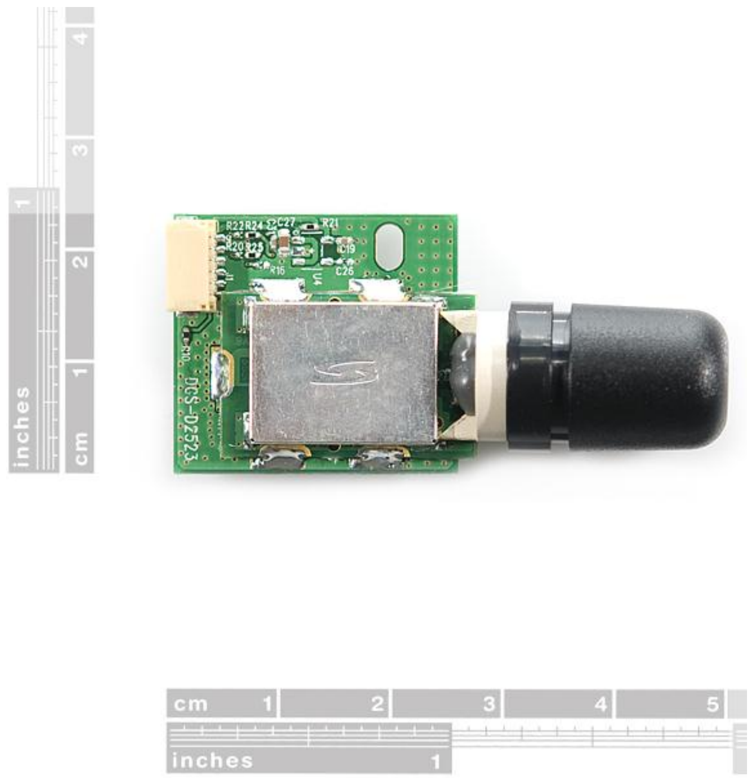


Figure 9: D2523T dimensions –

(Printed with permission of Open Source Sparkfun.com)

The advantages of this module were:

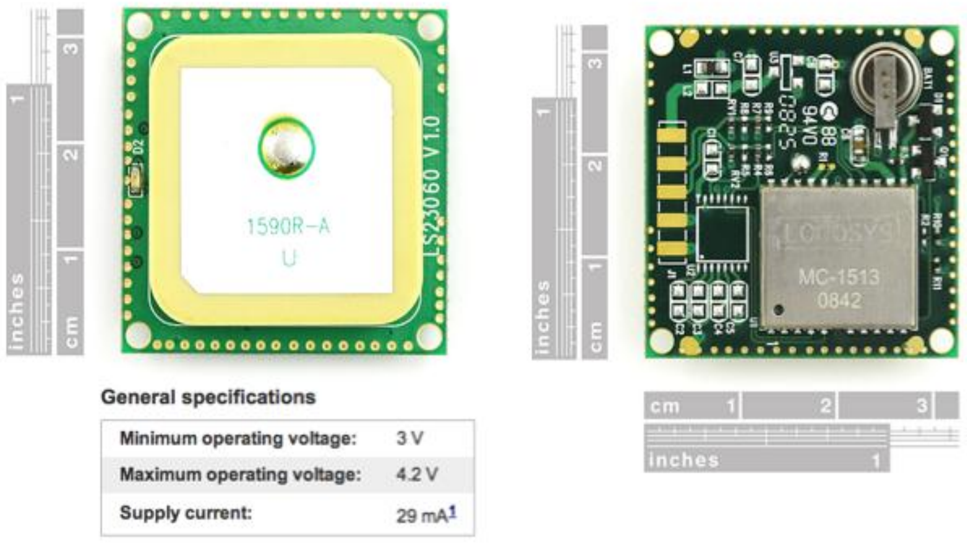
- 3.3 V supplied voltage for low power consumption.
- 50-channel u-box engine, which provided a faster and more accurate position, for the biker.
- It had a high performance antenna already integrated in the module.
- It had a high immunity to jamming, which was something that could come handy in our project because the bike was on the road.
- It offered an accelerated startup at weak signals.
- It usually offered a 1Hz update rate, but it could go up to 4 Hz, which was more than average.

Some of the disadvantages that we could encounter, if we picked this module for the helmet unit were the following:

- The cost of the unit was more expensive than average GPS modules.
- It had an update rate that was faster than average, so it would require the microprocessor to process more information.
- It did not have a LED.

2.3.2 LS20031 GPS 5Hz Receiver

Our second option was the Locosys LS20031 GPS receiver; this was a complete GPS smart antenna receiver that included an embedded antenna and GPS receiver circuits. This low-cost unit shown in Figure 10, outputted an astounding amount of position information 5 times a second. The receiver was based on the proven technology found in LOCOSYS 66 channel GPS SMD type receivers that used MediaTek chip solution.



General specifications

Minimum operating voltage:	3 V
Maximum operating voltage:	4.2 V
Supply current:	29 mA[1]

Figure 10. LS20031 with its dimensions and its main features

Printed with permission of Open Source Sparkfun.com

Advantages that made us consider this module:

- It had a LED indicator for fix or no fix.
- Its pads were easy to solder compared to other models.
- Average power consumption 3.3 V.
- Supported 66 channels for faster performance.
- It could go up to 10 Hz update rate, but it could also be configured to lower its update rate.
- Built in micro battery to preserve system data for rapid satellite acquisition.
- Reasonable price for its features $49.99.

On the other hand, some disadvantages were:

- It required voltage regulation if the module was implemented in a system that ran on a different voltage.
- It could transfer up to 57600 bps, which was a lot of data for the microprocessor to handle in a short period of time, but this could be modified.

2.3.3 Venus638FLPx-L GPS Receiver

The Venus638FLPx was a model that really caught or attention due to its size. It was not as powerful as the receivers previously reviewed, but it still was a high performance, low cost, single chip GPS receiver. It targeted mobile consumers and cellular handset applications due to its reduced size, and it also offered very low power consumption, high sensitivity, and best in class signal acquisition and time-to-first-fix performance. Venus638FLPx contained all the necessary components of a complete GPS receiver, included 1.2dB cascaded system NF RF front-end, GPS baseband signal processor, 0.5ppm TCXO, 32.768kHz RTC crystal, RTC LDO regulator, and passive components. It required very low external component count and took up only 100mm2 PCB footprint.

Figure 11, which is located below shows the unit. Here is a complete list of all the features that the receiver had to offer, and also here are a few advantages that we thought could make this module a good option for us:

- It was a very small object, its dimensions were 10 x 10 x 1.3 mm, and it only took up to 100 mm^2. Figure 11 can easily show how small it is compared to a coin.
- It offered jamming detection.
- It was compatible with active or passive antennas.
- Low price ($39.95)

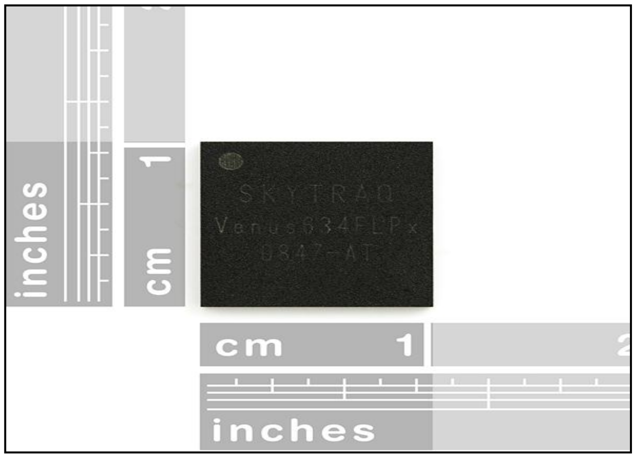


Figure 11. Venus638FLPx dimensions

Printed with permission of Open Source Sparkfun.com

As all the previous devices, the Venus638FLPx also had a few characteristics that could make it not so ideal to use in our senior design project. The disadvantages that we found are:

- A slower update rate compared to previous receivers.
- It did not have an antenna incorporated.
- It did not have an LED indicator for fix or no fix.

2.3.4 GP-2106 SiRF IV GPS Receiver

There were quite a few GPS receiver modules available in the market, but the GP-2106 was the last one that we were going to take into consideration. This module came with a built-in antenna and was also relative small compared to other designs; its dimensions were 21 x 6 x 6.2 mm. The SiRF Star IV powered the receiver, and the module could acquire satellites as low as -163dBm. It also had a hibernate mode which could go as low as 30uA while maintaining a hot start. It was important to know that the module ran at 1.8V VCC, however, you could use 3.3V TTL on the TX and RX pins. Also, the ON/OFF pin needed to be toggled high (to 1.8V) or low in order to turn on the module.

This module was not the most powerful one, but it did get the job done, as far as, locating the motorcycle rider while he or she was on the road. Let’s check out the main advantages that we found:

- It had a good number of channels.
- It offered an average update rate of 1 HZ, which would not transfer a lot of data at the same time.
- It had a built-in antenna.
- Cost $49.95, which was not that expensive.
- It’s smaller than quite a number of devices that offered similar characteristics. Figure 12 shows how this specific receiver looks, and some of its other features:

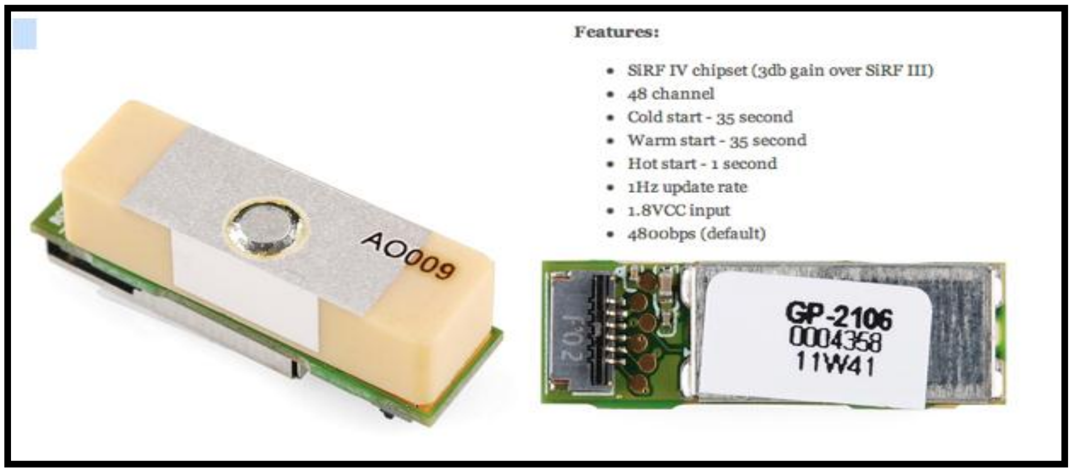


Figure 12: GP-2106 with its main features

(Printed with permission of Open Source Sparkfun.com)

This was a well-rounded receiver for its price, and it offered a good coverage, unfortunately, it also had some disadvantages such as:

- It did not have a LED.
- It had a limited update rate, so it would limit the speed of our design.
- It required voltage regulation if the module is implemented in a system that ran on a different voltage.

2.4 Accelerometer

An accelerometer is a special sensor that can detect accelerations and decelerations in any axis, depending of course on what that certain accelerometer is capable of. The accelerometer used in this project was given two main functions, the first one was to act as a motion sensor, and the second one was to detect possible accidents. In detecting accidents the accelerometers used to detect difference in accelerations. The accelerometer placed on the PCB was used as a motion sensor for our system and it was implemented for both crash and motion detection.

Since the accelerometer was able to pick up movement in different directions it was a good choice to use it as a motion sensor or a tilt sensor. Its requirement for a tilt sensor was to pick up minimal movement to the motorcycle or helmet while in the locked position, meaning with the motorcycle off. The direction did not really matter for this function only movement detection in any axis.

Another function of the accelerometer that was going to be applied was crash detection. Since one of the abilities of an accelerometer was to pick up decelerations, that could be used to detect any sudden high decelerations that are indicative of an accident. This was part of the safety system that helped the motorcyclist get a quicker response for help during an emergency, since motorcycle accidents could be deadly. Most of the time motorcyclists are by themselves and if by some chance they get in an accident with no one around how could they call for help? With this feature they could feel more at ease about that possibility since the safety system was able to call for them. The accelerometer was set with a certain threshold that once the amount of deceleration or g-force passed it, it triggered the safety system, and then the HTS sent a SMS with coordinates of the crash location. The accelerometer also had the ability to detect if the motorcycle was on its side and if the rider was flown from the bike, but this feature was not implemented on the final design. With all this information we were able to inform the emergency contact, and we provided the motorcyclist location. That way the motorcyclist got immediate help. The parameters of the accelerometer were chosen carefully for the system to work properly and accurately.

To be as accurate as possible using a three-axis accelerometer was the best choice. Since there were many factors to be accounted using a two-axis accelerometer would be a little inaccurate. For starters the way the module would be position in the HTS was not exactly on two axis, it might already had some tilt to it. Because of that we were not going to be able to know exactly in which axis the movement was on. The helmet itself might be modified different then when how it came from the factory. So to cover all different possibilities and still work like it’s supposed, the accelerometer had to be a three-axis sensor. The same applied to the crash detection function. Having a three axis allowed the system to be more accurate in calculating the g-forces in an accident, which was very important for the safety of the rider.

When the accelerometer was acting as the motion sensor of the system, sensitivity was very important and it needed to be high. It needed to be able to pick up the slightest movements for the system to work properly, but as far as when the accelerometer was being used as a detection device, sensitivity was not very important and it would needed to be low. When dealing with an accident we were dealing with high forces of deceleration, g-forces, thus the input received by the accelerometer was high. To be able to attain an appropriate output within the range desired, an accelerometer with low sensitivity was the best option. Sensitivity was the measure of how much an input signal was amplified, so depending how much we wanted to amplify the received input signal to the accelerometer that determined the sensitivity. This then made sense why we needed high sensitivity sensor for motion detection and low sensitivity sensor for crash detection.

Another important aspect of the accelerometer that was also crucial in its accuracy was its bandwidth. Bandwidth gave the range of the inputs that it could accept. Depending on what application it was going to be used knowing the type of inputs it will get helped us in choosing the bandwidth requirement for the accelerometer. In our project we were dealing with two different types of applications at first. The motorcycle had motion sensing to be able to detect any unauthorized tampering and be able to trigger a response. If someone were to try to steal the motorcycle they would need to move just slightly. Because the bike itself will be slightly tilted then the accelerometer would need to pick up very small movements. Having the right Bandwidth range would help in picking up such movements. Generally a small range of forty to fifty hertz would be good enough to pick up those small movements, unfortunate, after making the HTS a one PCB project, the motion sensing characteristics were taken out of the project. We did implement the accelerometer for crash detection; which was quite a different story. In crash detection the type of inputs that the accelerometer was receiving were much higher than in motion. An accelerometer of several hundred hertz was required to properly read the inputs and produce accurate outputs.

The communication of the accelerometer to the rest of the components depended on whether it was digital or analog. Those were the two forms that the information received can be transmitted. Analogs were constant readings. They could measure changes at a faster rate than digital could handle. It would excel over digital in cases where it was reading vibrations, or seismic activities. Basically in conditions that the inputs received by the accelerometer were dynamic at all times. It was basically sending a signal at all times of the input that it was receiving where as digital would send pulse width signals upon receiving an input. Depending on the acceleration would dictate how long the pulse width would be. For the task the accelerometer was doing in our project, digital was the best choice. There were no continuous input signals to be interpreted. If it was either for motion sensing and crash detecting the signals were short length in time making analog outputs not very efficient for this task.

Figure 13: Digital Accelerometer

(Printed with permission of Open Source Sparkfun.com)

The selected accelerometer was going to be integrated in the circuit and for it to be able to function it had to be able to communicate properly with the microcontroller. In other words the microcontroller and accelerometer needed to talk the same language so there could be communication. In communication of sensors there were two main protocols that were used, Serial Peripheral Interface (SPI) and Inter Integrated Circuit (I2C).

SPI protocol was a simpler form of communication than I2C, and Figure 14 shows a basic block diagram for it. It used three lines for data communication, which were Master In Slave Out (MISO), Master Out Slave In (MOSI), and a clock line. They also had a fourth line needed to be able to commence communication from the device. A Chip Select (CS) signal was needed to enable transfer of data. The clock line carried the clock pulses that provided synchronization of the data. The other two lines were the actual lines that transferred data out and in. The SPI 4-wire protocol supported full duplex communication with faster data transfers than the I2C. In a full duplex communication system, the device needed to have data coming in for it to send data out. One method to get around that was to send dummy bytes. Using SPI 3-wire protocol got around that problem as well. The 3-wire version used three lines by combining the MISO and MOSI into one line for data transfer. Instead of two it used one called Single Input Single Output (SISO). SISO was a single bidirectional data line that carried data in both directions. To be able to properly use the 3-wire version of SPI the device it was communicating to needed to support that version as well. If it supported only 4-wire version a method called bit banging could be used to be able to still use the 3-wire version.

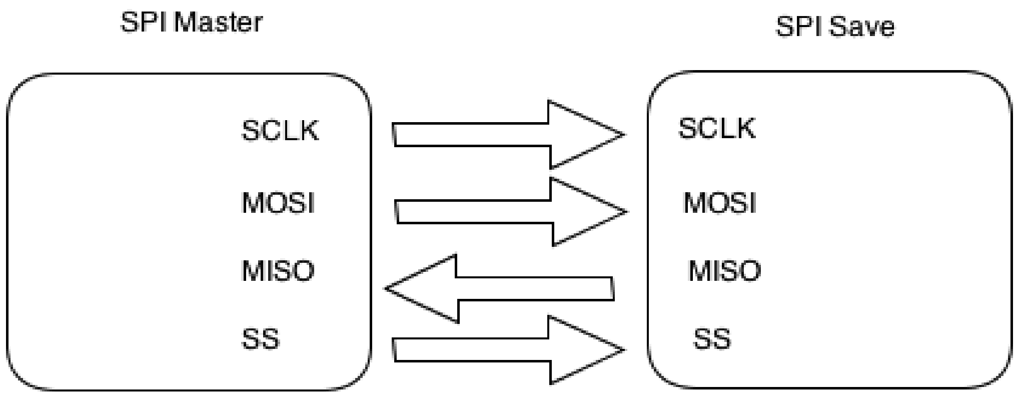


Figure 14: SPI communication system

The I2C protocol consisted of two bidirectional data lines that were pulled up to the voltage supply line usually called Vdd; figure 15 shows a block diagram of this. The two lines were the Serial Data line (SDA) and the Serial Clock line

(SCL). One of the big benefits over SPI it could communicate with many more devices. To start the communication between devices the master sent pulses of data with a START bit at the beginning, which indicated the beginning of the data. Then it was followed by either a WRITE bit or READ bit to know which action it needed to take. After that, it had the actual data it wanted to transmit, and the transfer stopped after a STOP bit was read. That START and STOP commands were just a transition from high to low on the SDA line with SCL high and low to high on the SDA line with SCL high.

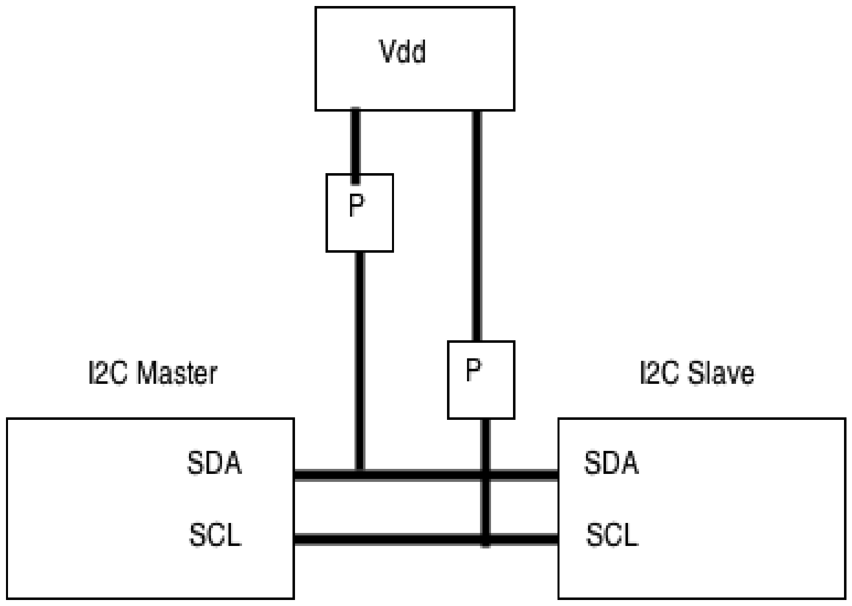


Figure 15: I2C communication system

By researching all the different specifications needed for our design to work as intended, at first we thought having two different accelerometers over one could be more efficient and accurate. We also thought that having just one accelerometer would be great since it was fewer components to deal with but since the parameters for the design were so wide it would be very hard to find one accelerometer that could do all the tasks accurately. To better focus we divided the tasks into two, one just for motion sensing and the other just for crash detection. We thought we would be able to better meet the parameters and to keep accuracy and performance as high as possible meeting the requirements of the design. Finally, after improving our design, we were able to only use one accelerometer on the helmet, and it performed all of these tasks effectively, and at the same time we kept the number of components to the minimum.

Another possibility to picking an accelerometer could be picking one that came with adjustable sensitivity and bandwidth. From the research taken to find the right component, we discovered that nowadays we could purchase an accelerometer with different settings of adjustability while keeping cost still close enough to that of a regular one. This was the best solution, it kept the components being used to a minimum thus also making it easier to make our module as small as possible. It was greatly help in achieving the goals of the project design.

We had to come up with a possible three different accelerometers that could work with our project. The table below shows the different specs of each of the accelerometer compared to each other. The first one was the ADXL345 by Analog Devices, and then the ADXL362 also by Analog Devices, and finally MMA8452Q by Free scale Semiconductors. Each of them had the potential to meet the requirement we had for our design.

Name	**ADXL345**	**ADXL362**	**MMA8452Q**
G-Scale and sensitivity	±2g, 3.9 mg/LSB ±4g, 7.8 mg/LSB ±8g, 15.6 mg/LSB ±16g, 31.2 mg/LSB	±2g, 1 mg/LSB ±4g, 2 mg/LSB ±8g, 4 mg/LSB	±2g, 1 mg/LSB ±4g, 2 mg/LSB ±8g, 3.9 mg/LSB
Output Data Rate	6.25-3200 Hz	12.5-400 Hz	1.56-800 Hz
Resolution	10-13 bits	12 bits	8-12 bits
Interface	SPI or I2C	SPI	I2C
Interrupt pins	2	2	2
Price	$27.95	$14.95	$9.95

Table 6: Accelerometer Specification Comparison

2.5 Global System of Mobile Communications (GSM)

As the project implemented Global Positioning System technology, the realization of a Global System of Mobile Communications (GSM) was a good idea because it allowed us to have access to a mobile communications network. GSM is a digital cellular phone technology based on time division multiple access (TDMA); GSM defines the entire cellular system, not just the TDMA air interface. Currently GSM utilizes a variation of TDMA, and is the most popular among the three digital wireless telephony technologies, which are: TDMA, GPS and Code division multiple access (CDMA). The basic concepts of a GSM were the followings: At first, the GSM unit digitalized and compressed data; then it sent this data down a channel with two other streams of user data, and each one had its own time slot. Finally, it was important to mention that the GSM operated in a number of different carrier frequency ranges (separated into GSM frequency ranges for 2G and UMTS frequency bands for 3G). Most of 2G GSM networks operated in the 900 MHz or 1800 MHz bands, and most of the 3G networks operated around the 2100 MHz frequency band.

The Helmet Tracking System (HTS) needed to have a GSM module, in order to have access to a network that allowed it to send text messages. Pretty much, our project sent a message to an emergency contact, informing that an accident occurred, and it provided the biker's location. There were quite a few networks available, from Edge to 4G LTE; but we did not actually need a super fast network such as 4G LTE, because we were only sending messages. As we mentioned before 3G operated around the 2100 MHz frequency band, and it was also well received around the market, so our group decided to get a module that implemented this network, or maybe a 2G. In order to send text messages the GSM module needed to be connected to a SMS module or have one already integrated. SMS stands for Short Message System; and it is a text messaging service component of a phone, web, or mobile communications systems. Originally the SMS technology was only used on cell phones that had GSM technology incorporated, but they have become so popular, that now they are supported by all the major cell phone systems.

Let us talk a little bit, about the services that the SMS module allowed us to have. This technology allowed us to send messages that could be up to 160 characters in length, but some used a 5 bit mode that allowed sending up to 224 characters in one message; we were only planning on texting the motorcycle location (longitude and latitude), and a small note that says, "an accident occurred". This technology was more than enough because we could totally do this in 160 characters. Now that we selected GSM and SMS as the technologies that we were going to use, and we also knew more about how they worked and the basic specifications that we needed. We were going to analyze a few modules that seemed to be appropriate for our Senior Design project.

2.5.1 GSM/GPRS Module - SM5100B

The SM5100B was the first module that caught our attention. It was a miniature, quad-band (frequency bands) GSM850 / EGSM 900 / DCS 1800 / PCS 1900 module that due its size and basic features could be compatible with our project. Our goal was to place this module on the PCB that is going to be located in the helmet system that was going to be located on the rear part of the protective gear; therefore, we were trying to save as much space as possible. Figure 16 shows the size of the SM5100 B module.

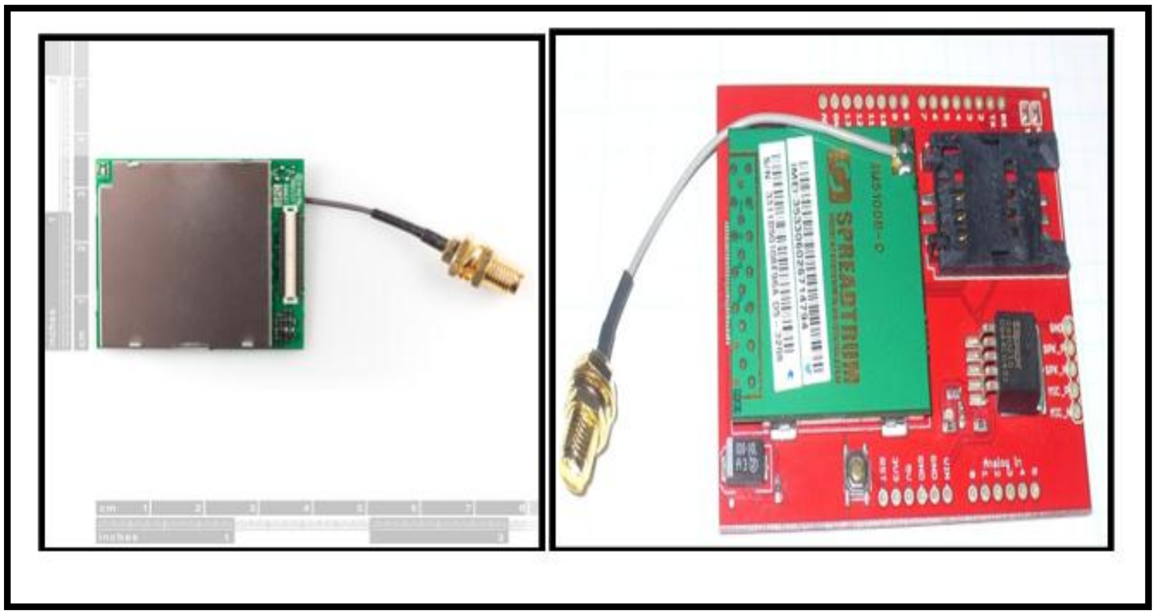


Figure 16: SM5100B dimensions & Arduino Cellular Shield with the module incorporated.

Printed with permission of Open Source Sparkfun.com

As it can be seen in the figure above the dimension of the module are quite small, so it fit our requirements for the PCB integration. Let us take a look at some other advantages that made this product appealing to us:

- It was a miniature module; its dimensions are: 35.0 x 39.0 x 2.9 mm. Also shown in Figure 16.
- No need of additional module to send SMS, this module was capable to provide almost all the basic features that a regular cell phone offers.
- As far as SMS technology. The module supported MO and MT for SMS. It also supported point-to-point short message broadcast. And it Supported TEXT and PDU modes as well.

- As far as the protocol, it supported GSM/GPRS 2/2+.
- It had an antenna already attached to the module, which facilitated the integrations.

Some disadvantages were:

- Its power voltage went from 3.3 V to 4.2 V (3.6 V is recommended), which was higher than the other components selected so far. Voltage regulator was required.
- The shape of the antenna required more space than what we wanted to use in our project, but it offered good reception.
- It did not have a SIM card socket, so it required an external SIM card socket, and it cost around $14.95.
- Its base price was $59.99 making this module more expensive than the one that we are going to check out next

We decided to go with this module, so it was a wise idea to use a Cellular Shield for testing, the Cellular Shield included all the parts needed to create an interface between the microprocessor and the cellular module. An example of this could be found on the previous Figure 16 that shows the dimension of the SM5200B, as well as an Arduino Cellular Shield with the Quad Band Module incorporated. The actual cellular shield illustrated in this figure has a price of $ 99.00 and it included the GSM module already.

2.5.2 ADH8066 GSM Module

The second GSM module that we analyzed was the AMH8066. It offered similar characteristics to the SM5100B; it was also a miniature, with quad-band GSM 850 / EGSM 900 / DCS 1800 / PCS 1900 module, and due to its dimension it could be easily and effectively integrated to our project, refer to Figure 17 below for actual dimensions.

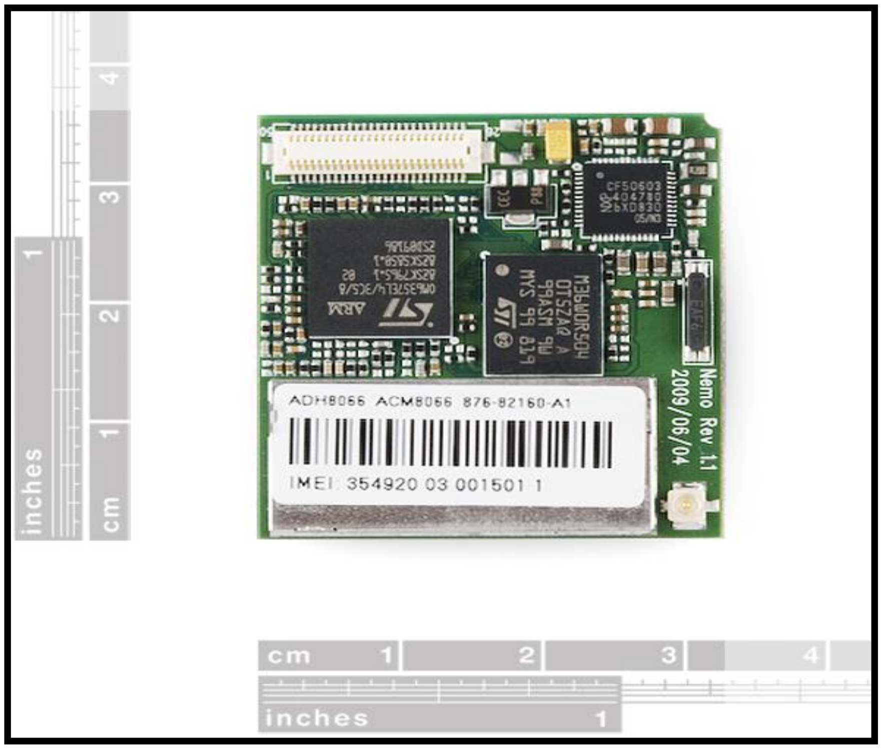


Figure 17: ADH8066 dimensions

(Printed with permission of Open Source Sparkfun.com)

Even though this module was not the most expensive or the most powerful in the market, it had more than enough power to send text messages, which was the main reason why we were incorporating this type of module into our Helmet system. Here are some of the advantages that the ADH8066 GSM module had to offer:

- Provided a good number of features for a reasonable price of $ 49.95.
- A little smaller than the previous GSM module. Its dimensions were 33.0 mm x 36.0 mm x 5.4 mm with the SIM cardholder.
- Included a SIM cardholder, which facilitated the integration process.
- Standard SIM interface (3V / 1.8V).
- Protocol supported GSM/GPRS Phase2/2+.

The two things that we did not like about this module were the following:

- Its power voltage went from 3.4 V to 4.5 V (4.0 V was recommended), which was higher than the other components selected so far. Voltage regulator was required.
- It did not have an antenna incorporated.

2.6 Battery

The type of battery selected for our design was a crucial part of the Helmet Tracking System (HTS). There were many different types of batteries in today's market to choose from and we had to pick one that had enough capabilities to meet the parameters that we had set for our design. Since in this generation we were exposed to so much technology we were already familiar more or less with what type were used to power up small devices. The main two types of batteries that were going to be considered for our design because of their many advantages and capabilities were lithium polymer and lithium ion. These two were the most commonly used for its size, and ability to recharge well. They were used in many application that we use nowadays, the most popular one were the batteries for hand held devices. Size, cost, and performance are some of the main points that we focused on for our design. The battery we selected needed to be light in weight and as small as possible, while it still met all the other requirements, and to be able to hold its charge for the time required.

For this design we were going to implement only one battery and it was located in the helmet. In the Helmet Tracking System (HTS) the battery was going to provide power to the Bluetooth module, the GSM module, the GPS, and other main components on the PCB during operation. The primary task of the system battery was to supply power to all the module that composed the HTS, without this power source, the system would not be able to acquire the location of the rider, which is key information for the SMS message, also while the rider was using the helmet he needs to be able to use the Bluetooth for outgoing and ingoing calls, GPS, or music purposes, and the only way of doing that is by properly charging and programming the main modules. The amount of charge the battery could provide had to be able to last longer than at least one entire days use. Even though it was highly unlike that someone would ride a motorcycle for one whole day, but that gave a good over calculation taken into account if they forget to charge it or other unforeseen possibilities. It was very important that the rider could be able to use its Bluetooth assistance in case of an accident; it would be the only way to have the ability to communicate with another person at anytime. If an accident were to occur an emergency response person could call ahead to check for consciousness of the rider. The rider being conscious could answer the call without moving to let help know of the severity of the accident, and if not then the previously set up Emergency Contact would know that he is dealing with a severe situation that requires immediate attention. Then for proper usage the next day it would need to be charged during nighttime or times of inactivity.

We started our research on this topic by looking at the lithium polymer batteries. They had many good advantages. One of them was the fact that they offered a very low profile that was small enough to resemble credit cards or even smaller if the demand was high enough. Manufacturers were not bound by any standard cell format. The polymers could be design to almost any shape and size. One advantage they had over lithium ion was that they used a gelled electrolyte over liquid. By having these designs, manufacturers did not need to put a hard metal enclosure and instead used heat-sealed foil. This allowed a space to be utilized a lot more efficiently making them much smaller than the lithium ion. Another benefit to its improved design was that it was less likely to leak electrolyte, which made it a much safer alternative. Li-ion had about the same benefits that Li-Po offered, except that Li-ion had higher energy density over Li-Po. They had a lower cost mainly because of higher supply and demand for Li-ion.

So after looking at its design benefits we could focus on its actual capabilities. A single cell was rated at 3.7 volts, which was the average rating for the battery. It was not going to operate at 3.7 volts its entire time of operation. When the battery was fully charged it would operate at around 4.23 volts and when it's discharged it would be around 2.7 volts. That means that whether the battery was at 4.23V or 2.7V all of our components that were running off of the battery still needed to operate properly and accurately. Their voltage capacities could vary depending on manufacturers' specs. If the input voltage needed to operate was greater than 2.7V then maybe a double cell battery would be more adequate. Lithium cells were set up in series and like stated before each cell rated at 3.7V and a Lithium pack could come with multiple cells depending of course on the parameters needed.

Li-Ion could provide a voltage of 3.7V, knowing how much current our design needed helped us pick the right capacity. Current was what dictated how much power was leaving the battery to supply the circuit. Our components were going to be low power for this main reason, that we wanted our battery to last as long as possible. Capacity was measured in milliampere hours (mAh). This rating allowed us to know how long the battery would last at a certain load. For example a battery rated at 1000 mAh with a load of 250 mA would only last four hours. So knowing that, the capacity for our design needed to be high enough to handle the load of our circuit for a full day. The discharge rate was related to the capacity but for our purposes it was not very important. All it told us was the maximum current load the battery could handle, and since our circuitry dealt with few low power components that was not an issue.

The recharging process of Li-Ion battery could be more intricate than others. To start, the charger used for this type of battery had to stop charging at 4.2V and not going anything beyond. The Li-Ion could not handle being charged at a voltage that was more than its rating. If the charge went beyond that voltage it could catch on fire or worse explode. Constant Current/Constant Voltage, CC/CV, method must be implemented to the charger or circuit connected to the battery to properly and safely charge. The way the method worked, at first the battery was charged with a constant current until it reached a voltage of 4.2V. Then it had to switch from constant current to constant voltage to keep at a constant 4.2V. The charger must gradually reduce the current until it gets to 2%-3% of the original constant current that it was providing. Charging a multi-cell Li-Ion required an extra procedure of monitoring each cell individually. No two batteries were ever the same meaning that they would have a different charge rate that must be monitored to not pass their charging capacity. Li-Po packs came with balancer plugs for the purpose of monitoring each cell to make sure that their voltage level be kept the same. The figure below shows the graph of the charging time of the Li-Ion battery implemented in the HTS.

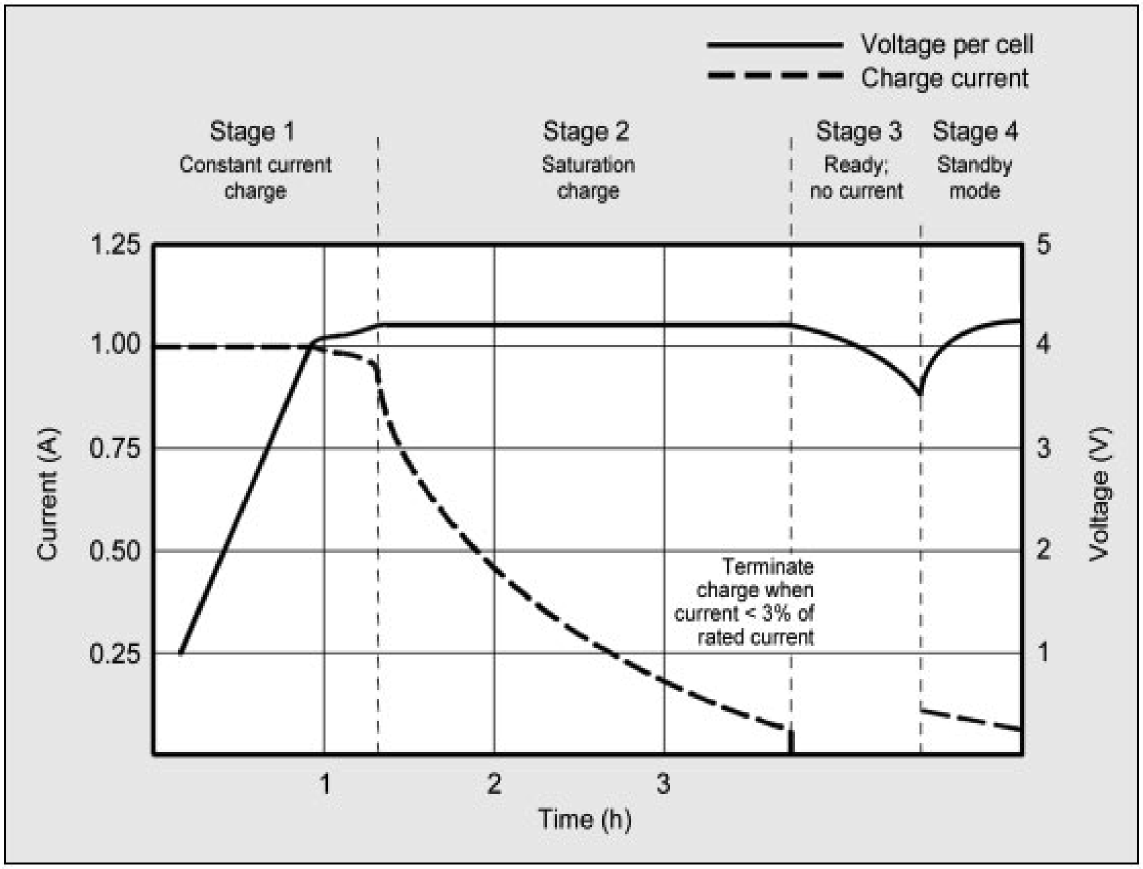


Figure 18: Charge Conditions

(Printed with Permission of Open Source)

Nowadays integrated circuit chips are manufactured just for the purpose of properly charging any kind of Li-Ion and Li-Po batteries. Most chips took either a DC voltage power supply or USB power supply and regulated it according to the CC/CV method to charge the lithium battery. This way any kind of overcharging and potential hazardous accidents were prevented. The specs of the chip had to be looked at carefully to make sure it worked well with the battery chosen. The rate of current at which it charged had to be below the max rating of the battery and the same for the voltage.

Looking back at both the advantages and disadvantages of the Li-Ion and Li-Po we saw that they were very evenly matched as far as what we needed them for.

The Parameters that needed to be met in our design could easily be met by either of the battery. But in one area that excels just a bit more than the other was in size. Size was a big issue in our design since this component was going to be put in a spot that has almost no space. The Li-Po excelled more on the size department because of their unique design. That way we can get the same performance of a Li-Ion but smaller in size. On the other hand, the Li-Ion offered a higher energy density. The task the battery had to perform was minimal and did not place a heavy burden on it. The main points were to provide a simple power source that could last for several uses before recharging and keeping its size as small as possible.

2.7 Microcontroller

Though we put copious amounts of research and conscious efforts in picking our parts to meet our design specifications and requirements, all would simply be futile without the direct control of a smart mind behind it all. Of course we could not control these components with our own brains while we wear the helmet, not yet at least. However adding intelligence, control and seamless integration to our design was nothing out of this world. This pivotal device to our design integration was of course a microcontroller.

The function of this microcontroller was truly the heart of our design as it was controlling, sending and receiving signals between our various peripherals being used in this project. For our project we did not need massive amounts of processing or computational power therefore a low cost low power microcontroller was suffice for the specific tasks we were implementing here. Some basic things we were looking for when choosing our microcontroller were the followings:

- It needed to have enough input and output pins to support our peripherals.
- It had to be compatible with our communication modules, accelerometer, GSM, GPS, and Bluetooth.
- It had to have sufficient memory and processing power to support our various triggering interrupts.
- It needed to be able to efficiently and seamlessly handle interrupts and exception handling.
- It had to offer low power and somewhat resistant to temperature and certain weather conditions.

The microcontroller was responsible for handling every signal sent from our peripherals. For one, it was directly connected to our accelerometer and was handling the signals sent and then decided whether or not conditions were normal or a special exception worthy of a software interrupt in which it would decide which action to take afterwards. For example, if our rider got into an accident our accelerometer triggers a tilt in motion, it then proceeded to triggering an alarm exception in which the microcontroller was then wake up the GSM module to send an SMS message to the owner notifying of the current location of the HTS.

We also took into account that none of us were computer engineers nor experienced programmers therefore our microcontroller had to be a programming friendly and intuitive one. We preferred to program in C rather than assembly and we decided to use an intuitive language as opposed to a cryptic one that would just end up giving us a hard time when we could have accomplished the same task using the simpler microcontroller from the start.

For our project we did not need a super powerful microcontroller but rather a solid, reliable and practical one. Basically any major microcontroller our there could suit our needs as we did not need a massive amount of I/O pins, memory or processing power therefore we focused our search in just finding the best fit microcontroller that allowed us to intuitively program it in an IDE environment and integrate it into a PCB board along with all our other peripherals.

Another design consideration was the architecture used to sufficiently fit our needs. More specifically we looked at how many bits would suffice to successfully implement our design. In the microcontroller world we were presented with very different varieties of controllers among these were the 8-bit, 16-bit 32-bit or even 64-bit architectures. For our design It was logical to pick an 8 or 16-bit architecture as it was sufficient yet not a complete waste of processing power. This will be explained with more detail further through the report.

Once the chip was programmed, we soldered it into the integrated circuit board as carefully as possible. Carefully being a key point as the pins and connections are sensitive and by doing this we ensured that our connections were free of shorts, disturbances or inhibitions of any sort in turn, our design was ensured to work as best as possible after we soldered and connected all of our components properly.

Before we began to even consider implementing much less choosing a specific microcontroller into our project, we first touched some bases and tried to get a strong foundation on what exactly a microcontroller was. Simply put, a microcontroller conceptually was a controlling brain that did what it was told to do by previously specified orders. It did nothing more and nothing less than what it was told to do. These specified orders were analogous to code, which could be as high leveled as assembly or machine code up to C, C++ or even java. It was designed to have connections or input and output ports through which it will receive or transmit signals and base its decisions with respect to the coded interrupt handling provided. I mentioned interrupts because the nature of our project is a fusion of sensors, which transmitted signals which then trigger certain interrupts on our microcontroller.

2.7.1 Researched Microcontrollers

Having taken the newest Embedded Systems course which was based around the ever popular Texas Instruments MSP430 our first logical step was to weigh our options and do some research on said powerful yet versatile device. The MSP430 was a top potential candidate due to it being so accessible, cheap and effective with even complex projects. With it being attainable at $4.30, serious consideration was not given a second thought. It is capable of ADC, had a respectable amount of flash memory and SRAM along with a timer. The MSP430 ships straight from TI at $4.30 brand new in the box with the following:

- MSP-EXP430G2 Development board
- M430G2553 MPU
- M430G2452 MPU
- 32.768 KHz oscillator crystal which you could solder onto the board for further timing accuracy
- Two sets of extra header pins
- USB cable used for power supply as well as programming
- Some neat TI Launch pad stickers

Right off the bat we could see that TI was pushing its MSP430 Launch pad into hobbyist's hands at a dirt-cheap price. This was very hard to pass off on in comparison to an Arduino board that ran you up a couple more and not include nearly as much for the price asked. In the long run, the MSP430 was a great bang for the buck.

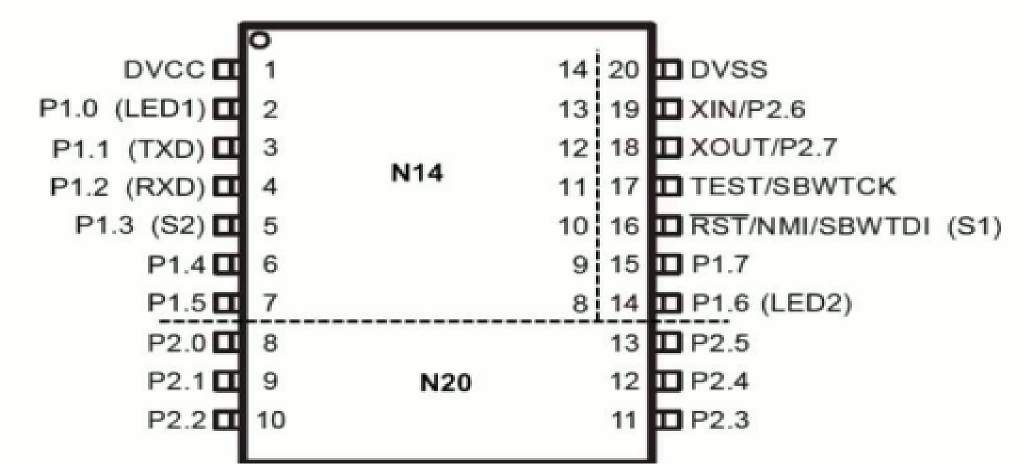


Figure 19: MSP430 device pin out

(Printed with permission of Open Source Sparkfun.com)

The layout of the board was split up into two logical sides. The USB interface was on the left hand side of the board and came already pre-jumpered with the USB interface connected. There was of course a power LED on the USB side and power could come via USB or the header pins on the other side of the board included in the package.

On the other side of the board we had two push buttons, two LED's, header breakouts for each of the 20 pins on the IC socket, 3 pins for power input header (VCC, GND, GND), and two jumpers to connect and disconnect the two integrated LED's. The board itself was a very simple circuit excluding the USB converter on the side of the board.

Out of the two buttons, one of those buttons was a reset button hardwired to pin 16 and of course there was no real way to change the function of that button. The other button switch went to pin 5 of the IC socket and that was of course an all-purpose programmable switch. There were also two programmable LED's, LED 1 (red) and LED 2 (green) and you had the option to easily disconnect them via jumpers.

Overall the board was cheap, well-designed and sturdy, small footprint and perhaps its most prominent figure of merit; power consumption. The MSP430 was extremely low power by nature and when it came to integrating it in our design, it would make for a ridiculously low power consumption feature that none of us would protest to.

The next slate to touch was to cross into the Arduino realm. The Atmel AVR controllers had tons of features to offer. The first upside being that they were the basis or the super popular and ever growing group of Arduino boards and as engineers, we could definitely agree that there was strength in the numbers. The Arduino community was unparalleled, which such a large crowd there was just a plethora of information on just about every product and task implementation ever thought of which made Arduino automatically attractive to amateur programmers like us.

Our search began with the ATMega328, a widely used microcontroller. It had an I2C port that allowed it to be used for any I2C peripheral needing the port. It had a decent amount of SRAM and flash memory meaning that it could hold modules of applications. As with the MSP430 the pin counts were unfortunately low but of course this did not limit its use as a secondary microcontroller in our system should we find the need to have one. The other varieties were the Atmel Xmega line of microcontrollers, which offered much more power and processing capabilities. With their superior memory capacity, speed and greater pin counts they were easily an opportunity to abuse power. They provided multiple I2Cs, UARTs and pin counts that exceeded our estimated pin counts. While tempting, the Xmega series just seemed like too much power for our intended purpose of use when it came down to a choosing a microcontroller.

While we were in at that high of a level we also swooned over to the Stellaris LM4F. This was a seriously powerful surface mount microcontroller designed by none other but TI. It provided multiple I2Cs and UART ports. It had more memory than the MSP and Xmega as well as a much faster clock speed. The evaluation board ran up to around $150 and that of course was a red flag in our search as we did not need that kind of power nor that fancy of a price tag.

Having both too much power at a high price and not enough for a really nice price tag left us wanting to finally touch some neutral ground with a good combination of the two. Looking further through the Arduino website we found the trademark ATMEGA 2560 and while we loved what it had to offer we saw a very similar compromise and this time we couldn't really nitpick it, as it had just what we needed. This microcontroller was referred to as the Arduino Mega.

The Arduino Mega was a microcontroller based on the Atmega1280. It had 54 digital input/output pins with 15 having the capability of being pulse width modulated outputs. 16 analog inputs, 4 UARTs, a 16MHz crystal oscillator, a USB connection, a power jack, an ICSP header and a reset button. At a cursory glance, this was a seriously potential final candidate for us.

While not being as small nor power efficient as the MSP430, it gave us a lot more flexibility when it came to pin counts and memory capacity which was always a good thing due to the fact that there were surely a lot of features and maybe even parts that we switched around, but the fact stayed that it was a capable microcontroller with a large community support base, compatibility and ease of use and programming, which all were a plus when it came to our limited programming experience.

The could not forget about the ATmega328, which was an excellent option, we were able to program it using the developing board Arduino UNO, and it offered just the right amount of pin and processing power for our design. In case, we wanted to go with a little bit more power, we could go with the Atmega1280, and Table 7 shows a chart of the basic features the 1280 had to offer, if it was chosen for the HTS.

Microcontroller	ATmega1280
Operating voltage	5V
Recommended Vin	7-12V
Input Voltage limit	6-20V
Digital I/O pins	54(15 being PWM)
Analog Input pins	16
DC current per I/O pin	40mA
DC current for 3.3V pin	50mA
Flash Memory	128KB, 4KB being used by boot loader
SRAM	8KB
EEPROM	4KB
Clock Speed	16MHz

Table 7: ATmega1280

In summary, while the MSP430 offered low cost, accessibility and functionality if fell short in pin counts and communication interfaces. The Xmega series we just too much as well as the Stellaris, as beautiful of a microcontroller as it. And finally the 1280 was found to be a more reasonable and definitely suitable neutral ground between the MSP and the Stellaris. It is important to notice that when we actually came to build our prototype, we decided to go with the ATMEL ATmega328, since the 1280 had more pins that we actually needed, and we were trying to keep a small PCB design. In general, the 328 had all the characteristics that we needed to successfully program all of the modules that composed the Helmet Tracking System (HTS).

2.8 HELMET

For security purposes, the government has created some rules and regulations for the motorcyclists around the United States; these regulations could change according to the state. The use of a protective headgear was one of these rules; in fact, this was a key factor that needed to be taken into consideration because the safety of the rider should always be a priority. For the Helmet Tracking System (HTS), we were planning on buying a helmet already built, but we were going to modify it, in order to properly add the required modules, buttons, speakers, microphone, and other components that made part of the helmet system. In order to keep our project as safe as possible, we were going to buy a brand new helmet unit that was certified with recognized safety standards and regulations. We could get a used headgear too, but it had to be in optimal conditions. For us, it would not acceptable to risk the security of the user just because we wanted to save a few bucks getting a bad helmet.

There were different sizes of helmets, but we were thinking about selecting full-face medium helmet, according to Rigel Jimenez, one of our group members, and motorcycle rider. This model offered a comfortable design, it was stylish, and it had enough empty space to locate the parts that need to be incorporated. We were contemplating the idea on having a small PCB outside of the helmet with a microcontroller, a Bluetooth module, a GSM module, a GPS module, a button, and some other components; as well as the microphone and speakers for interaction in the inside of the helmet.

Table 7 shows some helmet models that we were taking into consideration, and that were affordable. A motorcycle helmet could be very expensive; for example, Shoei was one of the best brands and it had some really good looking units, but their prices went above $400.00; which was way too pricey for us to afford. After looking online for quite a while, we decided to go with something not that expensive, but still fully qualified for our project; finally we came out with the following information. The brands listed below were also very good and offered a good amount of security features, while keeping a reasonable price. It is important to notice that all these are prices for new units; we still might be able to get a used helmet in optimal conditions.

Brand	**Model**	**Size**	**Price**
Scorpion	EXO 750	Medium	119.95
Scorpion	EXO 400	Medium	99
HJC	CS-R2	Medium	79.5
HJC	CL-16	Medium	125.99
Bell Arrow	Solid	Medium	89.95
Icon	Airmada	Medium	131.99
Nolan	N-43e	Medium	150

Table 8: Helmets and Prices

A full-face motorcycle helmet had a protective outer shell built that stand impacts of large magnitude without cracking, followed by a layer of insulating material usually made out of cloth of soft material; within the two holes that allow air through the helmet. Perhaps the most key features of this type of helmet were that it covered the full face of the motorcyclist, especially by a full visor that could be flipped, some came with tinted shields to protect your vision from the Sun while riding. Safety wise, these were the most protective helmets money could buy. Figure 20 shows an example of a Full-face Medium helmet; it can better demonstrate the features previously said about this type of helmet.

Figure 20: Full-face Medium Helmet (Scorpion EXO-750)

(Printed with Permission of Open Source)

Section 3: Design Details

3.1 GPS Selection

After doing some research about GPS technology; and also comparing four different GPS modules, our group decided to select the Locosys LS20031 GPS receiver for our project. This module was chosen because it offered some really good features for its price. This receiver was well balanced for our project, and its specifications met all the requirements that we needed to successfully build our Helmet Tracking System (HTS). Figure 21, displayed on the following page, shows that the module had 5 pins that needed to be connected, in order to be properly integrated into our electronic system. Figure 21 by itself does not describe each pin, but Table number 9 placed below the figure of the modules, gives this information. The process of soldering the unit was easy due to the shape and number of pins of the module. To solder down the receiver we used a L-shaped wire. Referring back to our research about the Locosys LS2003, the unit required an input voltage of 3.3 V; we were be able to achieve this exact voltage by adding a 3.3 V voltage regulator to our PCB final design.

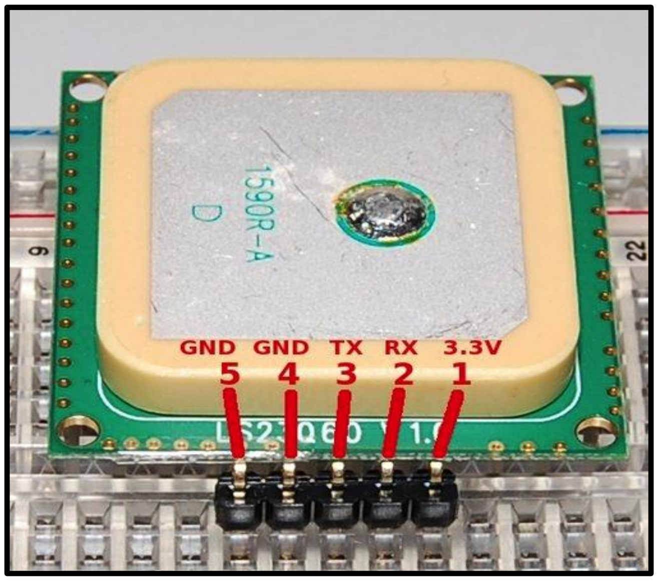


Figure 21: GPS LS20031Receiver Pin Layout

(Printed with permission of Open Source Sparkfun.com)

Pin #	Name	Description
1	VCC	Power Input
2	RX	Data Input (TTL level)
3	TX	Data Output (TTL level)
4	GND	Ground
5	GND	Ground

Table 9: GPS LS20031Receiver Pin Description

Another key factor that we took into consideration was the kind of information that the GPS receiver could give us. The LS20031 module provided NMEA (National Marine Electronic Association) massages as output, as most modules in the market do. The possible outputs that we could get from this receiver are shown below in Table 10.

NMEA record	Description
GGA	Global Positioning System Fix data
GLL	Geographic position (latitude / longitude)
GSA	GNSS DOP and active satellites
GSV	GNSS satellites in view
RMC	Recommended minimum specific GNSS data
VTG	Course over ground and ground speed

Table 10: GPS LS20031 Operation Modes

For our Helmet Tracking System, we only used the first two NMEA record outputs, which were GGA and GLL because they pretty much provided all the information that we needed, in fact they provided more than that since we only needed to acquire the longitude and the latitude of the receiver placed on the helmet. GGA stands for Global positioning system fixed data. Table 11 shows a more detail description of the output that the receiver provided us with the GGA NMEA record output.

$GPGGA,053740.000,2503.6319,N,12136.0099,E,1,081.1,63.8,M,15.2,M,,0000 *64

Name	Example	Units	Description
Message ID	$GPGGA		GGA protocol header
UTC Time	53740		hhmmss.sss
Latitude	2503.6319		ddmm.mmmm
N/S indicator	N		N = North or S = South
Longitude	12136.0099		ddmm.mmmm
E/W indicator	E		E = East or W = West
Position Fix Indicator	1		See Table Below
Satellites Used	8		Range 0 to 12
HDOP	1.1		Horizontal Dilution of Precision
MSL Altitude	63.8	meters	
Units	M	meters	
Geoid Separation	15.2	meters	
Units	M	meters	
Age of Diff. Corr.		second	Null fields when DGPS is not used
Diff. Ref. Station ID	0		
Checksum	*64		

Table 11: Global Positioning System Fixed Data (GGA) Output Description

To better understand the output provider by the module, we also need to understand the specific values that the GPS positioning fix indicator provides. Table 12 shows all the possible values.

Value	Description
0	Fix not available or invalid
1	GPS SPS Mode, fix valid
2	Differential GPS, SPS Mode, fix valid
3	Not Supported
4	Not Supported
5	Not Supported
6	Dead Reckoning Mode, fix valid

Table 12: Positioning Fix Indicators

The second NMEA that we needed to understand was the GLL (Geographic Position- Latitude / Longitude); this was some of the data that we needed to create the SMS message that we were going to send to the previously set Emergency Contact; in case of an accident. Of course this data had to be parsed and fully processed to provide and output in English that was clear enough for the person receiving the message to understand. Table 13 shows the code, in which, the GLL output is provided by the GPS receiver.

$GPGLL.2503.6319.N.12136.0099.E.053740.000.A.A*52

Name	Example	Description
Message ID	$GPGLL	GLL protocol header
Latitude	2503.6319	ddmm.mmm
N/S indicator	N	N = North or S = South
Longitude	12146.0099	ddmm.mmmm
E/W indicator	E	E = East or W = West
UTC Time	53740	hhmmss.sss
Status	A	A = Data valid or V = data not valid
Mode	A	A = autonomous, D = DGPS, E = DR
Checksum	*52	
<CR><LF>		End of message termination

Table 13: Geographic Position (latitude / longitude) Description

By obtaining the GLL and GGA output data; out project was able to get accurate data about how many satellites were currently being seen by the GPS receiver, as well as the current geographic position, which was the main purpose that made us integrate this module to our senior design project.

As we can see on the Figure 22, the first pin that we used was the VCC pin; we used it to supply the 3.3 V that the module required. The Pin 2 and 3, which were TX0 and RX0 respectably, were connected to the microcontroller to transfer data. Finally the pins 4 and 5 were grounded.

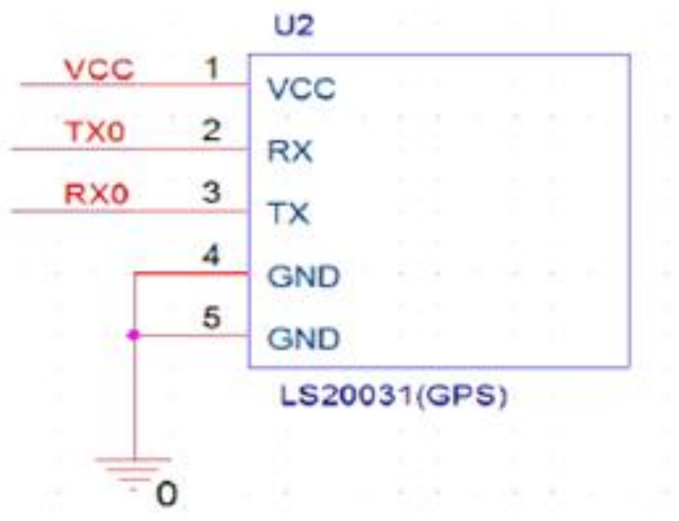


Figure 22: LS20031 GPS Receiver Schematic

(Printed with permission of Open Source Sparkfun.com)

3.2 Accelerometer Design

After the research was done and the parameters were narrowed down we were ready to pick the right accelerometer for our design. We looked at both options of using either two accelerometers to implement one function each and also one accelerometer to implement both functions. We were able to find one accelerometer with enough adjustability that can be used accurately to implement both options, and it was used for the Helmet Tracking System (HTS). This module was also fully compatible with our microcontroller unit and the other modules used. The ADXL345 accelerometer was chosen to function as the motion and crash detector. Their cost was $27.95 from the vendor Sparkfun.com, and was manufactured by Analog Devices. Since this was a very small chip we decided to get it with a breakout board for testing, and after making sure it worked properly we placed the unit on our PCB. This made the soldering process much simpler. The breakout board was soldered using headers to the circuit board for testing purposes only, and after that only the accelerometer was used. This part was also chosen because of the wide research and information available for it.

The ADXL345 met all the requirements that were needed for our design. It also had extra features that if needed could also be implemented. By having the extra features it gave more room for its implementation and possible modifications. It had three different protocols interface that could be used with our microcontroller, SPI 3-wire, SPI 4-wire, and I2C. Since our accelerometer was only needed to speak with the microcontroller we ended up using the SPI 3-wire version. It was of the simpler ones and required less connections. The sensitivity and g-force mode had adjustability of ±2g, ±4g, ±8g, and ±16g selectable by the user, which is shown on the Table 14 below. It has 10 to 13 bits of resolution.

Sensitivity Modes of Operation			
FS[1:0]	**Range**	**Sensitivity**	**Application**
Set to 00	±2g	3.9 mg/LSB	Motion/Tilt
Set to 01	±4g	7.8 mg/LSB	Crash Detection
Set to 10	±8g	15.6 mg/LSB	Not Used
Set to 11	±16g	31.2 mg/LSB	Not Used

Table 14: Accelerometer Sensitivity

As far as the power consumption, the parameters for the ADXL345 were way below our max threshold. It put very little strain on the battery that was used as the power source. Its supply voltage ranged from 2.0 V to 3.6 V and the current consumption ranged from 40 µA to 145 µA. It came with different modes of operation that conserve energy.

The accelerometer had a total of 14 pins, with two programmable internal interrupt pins that could directly communicate with a microcontroller when a certain threshold has been exceeded in the accelerometer. Also came with 12 different modes that could have a certain threshold and once it was met it registered to its corresponding address in the register. The two modes that apply to our project were the tap ready and Activity mode. In Single tap mode a bit was set when a single acceleration event was greater than the value of the threshold. The Activity mode triggers when acceleration were greater the value stored in the threshold register.

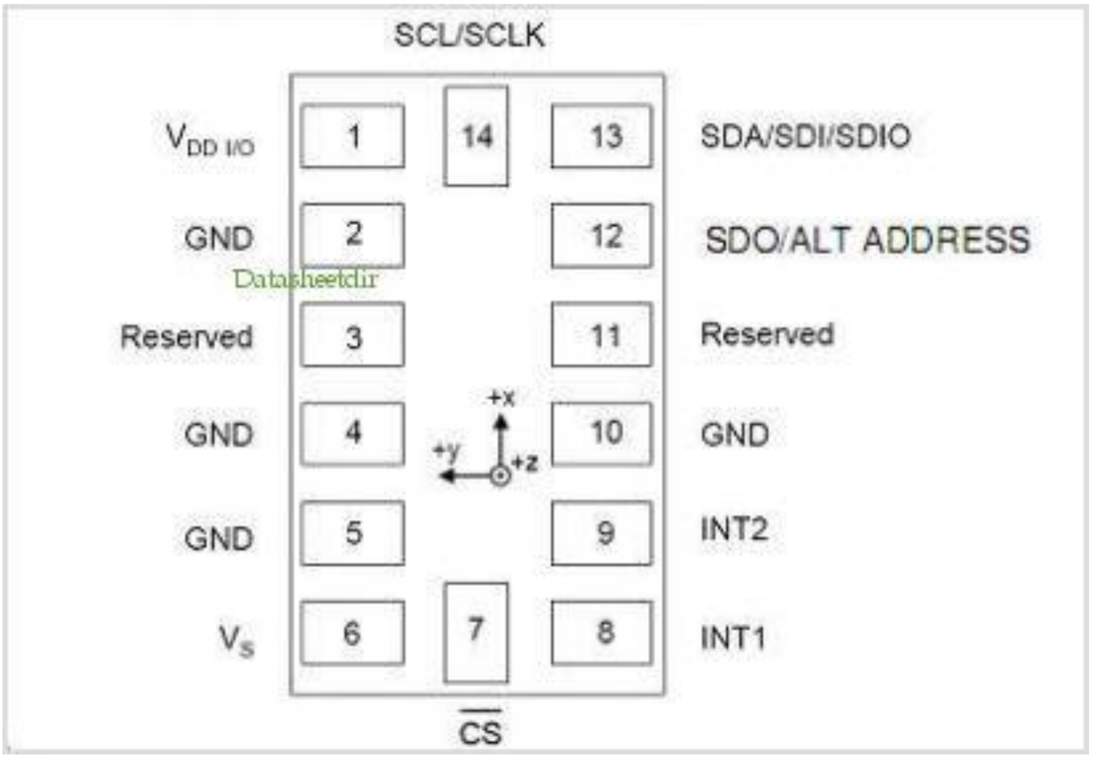


Figure 23: ADXL345 Pin Configuration

(Printed with permission of Open Source Sparkfun.com)

The accelerometer was surface mounted to the PCB board. Pins 7, 13, and 14 were the I2C interface pins that communicated with the microcontroller. Pin 7 was the CS Chip Select that allowed the communication for the accelerometer to start. This pin was set low for communication to pass. Pins 13 and 14 could also be set to communicate by the SPI format as well. Applying the appropriate bit in the SPI register helped select either the 3-wire or 4-wire communication. Any kind of information that needed to be sent into the accelerometer was done by either of these interfaces. It was also programmed through this interface to send interrupts if certain thresholds were met within the accelerometer. Pins 8 and 9, which were the interrupt pins, were going to be connected to the microcontroller. When a certain threshold was reached it sent either of these two as a confirmation signal to the microcontroller.

Pin #	Pin Name	Description	Pin Status
1	VDDIO	Digital Interface Supply Voltage	Input
2	GND	Must be connected to Ground	Input
3	Reserved	Reserved.	Open
4	GND	Must be connected to Ground	Input
5	GND	Must be connected to Ground	Input
6	Vs	Supply Voltage	Input
7	CS	Chip Select	Input
8	INT1	Internal Interrupt 1	Output
9	INT2	Internal Interrupt 2	Output
10	NC	Not internally connected	
11	Reserved	Reserved.	Open
12	SDO/ALT	Serial Data Output/Alternate I2C	Output
13	SDA/SDI/SDIO	(I2C)/ (SPI 4-wire)/ (SPI 3-wire)	Input/Output
14	SCL/SCLK	Serial Communications Clock	Input

Table 15: ADXL345 pin description

The accelerometer came with many different functions of interrupts to could be applied. Each of the interrupt functions could be used simultaneously, with the only exception being that a certain function may needed to share the interrupt pins to work properly. To enable interrupts the appropriate bit had to be set in the INT_ENABLE register at the address 0x2E and were mapped either interrupts by the INT_MAP register at address 0x2F. Analog Devices did recommend having an interrupt disable just to make sure there were no faulty interrupts set. They could be monitored by reading the data in the INT_SOURCE register at address 0x30. All the functions that provided interrupts are listed below with an explanation of what each function does.

Interrupt Functions

- **DATA_READY**: Bit was set in DATA_READY register when new data was available, and it was cleared when there was no new data available.
- **SINGLE_TAP:** Set when a single acceleration event was greater than the value set in the THRESH_TAP register at address 0x1D. It also must occur for less time than the specified value in register DUR.
- **DOUBLE_TAP:** Set when two acceleration events were greater than the value set in the THRESH_TAP register occurring for less time than the specified value in register DUR.
- **ACTIVITY:** Set when acceleration value was greater than the specified value in register THRESH_ACT in address 0x24.
- **INACTIVITY:** Set when acceleration less than the value stored in register THRESH_INACT at address 0x26.
- **FREE_FALL:** Set when acceleration less than the specified value stored in the register THRESH_FF at address 0x28 was experienced longer than the duration value stored in register TIME_FF.

To be able to use all the different functions of this accelerometer we had to have the right bits in the right address. The microcontroller through the SPI 3-wire interface had to give the data with the right address. Below is a Register Map that has the important registers that were needed for our project. In each bit of the register had a function that it implemented unless noted otherwise. Through the registers below we could adjust the sensitivity, picked a certain interrupt source, a certain power mode, and activated the axis that was needed.

REGISTER MAP

D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
1	1	1	0	0	1	0	1

Register 0x00-DEVID, Device ID

D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
ACT ac/dc	ACT_X enable	ACT_Y enable	ACT_Z enable	INACT ac/dc	INACT_X enable	INACT_Y enable	INACT_Z enable

Register 0x27-ACT_INACT_CTL, Axis enable control for activity and inactivity detection.

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	Suppress	TAP_X enable	TAP_Y enable	TAP_Z enable

Register 0x2A-TAP_AXES, Axis control for tap/double tap.

D7	D6	D5	D4	D3	D2	D1	D0
0	ACT_X source	ACT_Y source	ACT_Z source	Asleep	TAP_X source	TAP_Y source	TAP_Z source

Register 0x2B-ACT_TAP_STATUS, Source of tap/double tap.

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	LOW_POWER	RATE			

Register 0x2C-BW_RATE, Data Rate and power mode control.

D7	D6	D5	D4	D3	D2	D1	D0
0	0	Link	AUTO_SLEEP	Measure	Sleep	Wakeup	

Register 0x2D-POWER_CTL, Power-saving features control.

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	Suppress	TAP_X enable	TAP_Y enable	TAP_Z enable

Register 0x2E-INT_ENABLE, Interrupt enable control.

D7	D6	D5	D4	D3	D2	D1	D0
DATA_ READY	SINGLE_ TAP	DOUBLE_ TAP	Activity	Inactivity	FREE_ FALL	Watermark	Overrun

Register 0x2F-INT_MAP, Interrupt mapping control.

D7	D6	D5	D4	D3	D2	D1	D0
DATA_ READY	SINGLE_ TAP	DOUBLE_ TAP	Activity	Inactivity	FREE_ FALL	Watermark	Overrun

Register 0x30-INT_SOURCE, Source of Interrupts.

D7	D6	D5	D4	D3	D2	D1	D0
SELF_TEST	SPI	INT_INVERT	0	FULL_RES	Justify	Range	

Register 0x31-DATA_FORMAT, Data format control.

After knowing all the information on the accelerometer we needed to know how to put it on a circuit board properly, meaning we had to apply the right power source. The ADXL345 was quite simple to put together and apply the power it needed. As stated above the SPI interface pins and the Interrupt control went to the microcontroller. As far as the power the only precaution that needed to be taken was to place two capacitors, one for the Vs and the other for Vdd I/O. The Capacitor Cs must be rated for 1 µF and C_{IO} at 0.1 µF.

These two input pins also had modes that could be activated with the right signal. The modes were Power Off, Bus Disabled, Bus Enabled, and Standby or Measurement. The two Bus modes were of no concern for our project, only off and Standby or Measurement. Both of these inputs were going to be sharing the same power source meaning it could only be on those two modes mentioned. Once the voltage was applied the accelerometer went first into standby mode. It was recommended to configure the device in standby mode than by applying the right bit on the POWER_CTL register it could be switched to Measurement. In Measurement the device was fully on, receiving the inputs of the 3-axis.

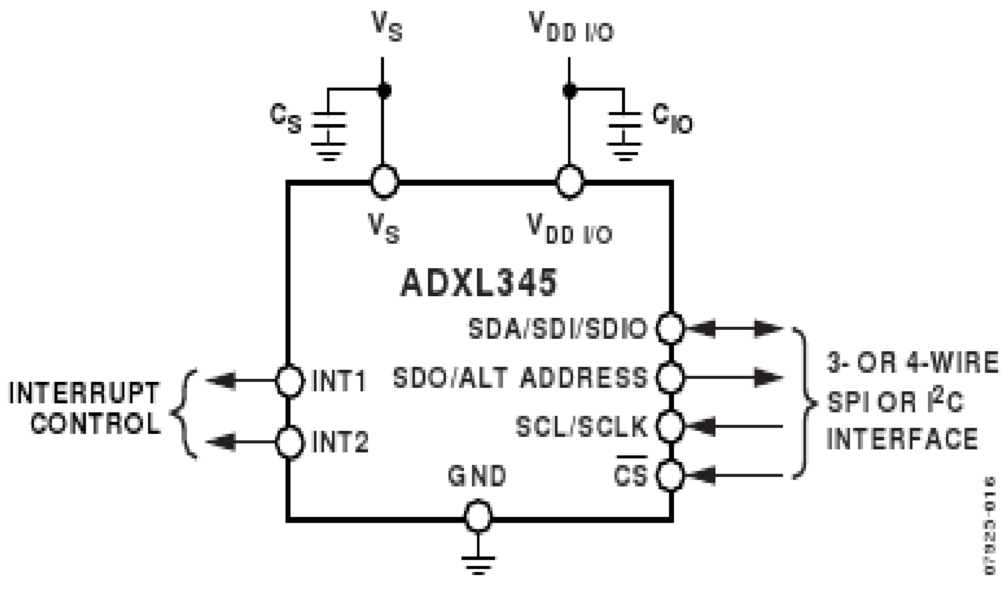


Figure 24: ADXL345 block schematic

(Printed with permission from an open source Sparkfun.com)

3.3 GSM Design

In the research part of the project, we analyzed the GSM and SMS technologies, and this improved our knowledge about the topic. We checked out two modules that met the basic requirements that we set up for the project, and we selected the SM5100B. Some of the things that made us choose this module were its specifications, its price, and compatibility with the selected microcontroller. As we mentioned before we liked the fact that the module had an incorporated antenna for better reception.

In order to properly integrate this part into the main system we needed to connect it properly to our PCB design located in the Helmet. Figure 25 shows the Eagle Cad schematics for the SM5100B, showing the name of all its pins.

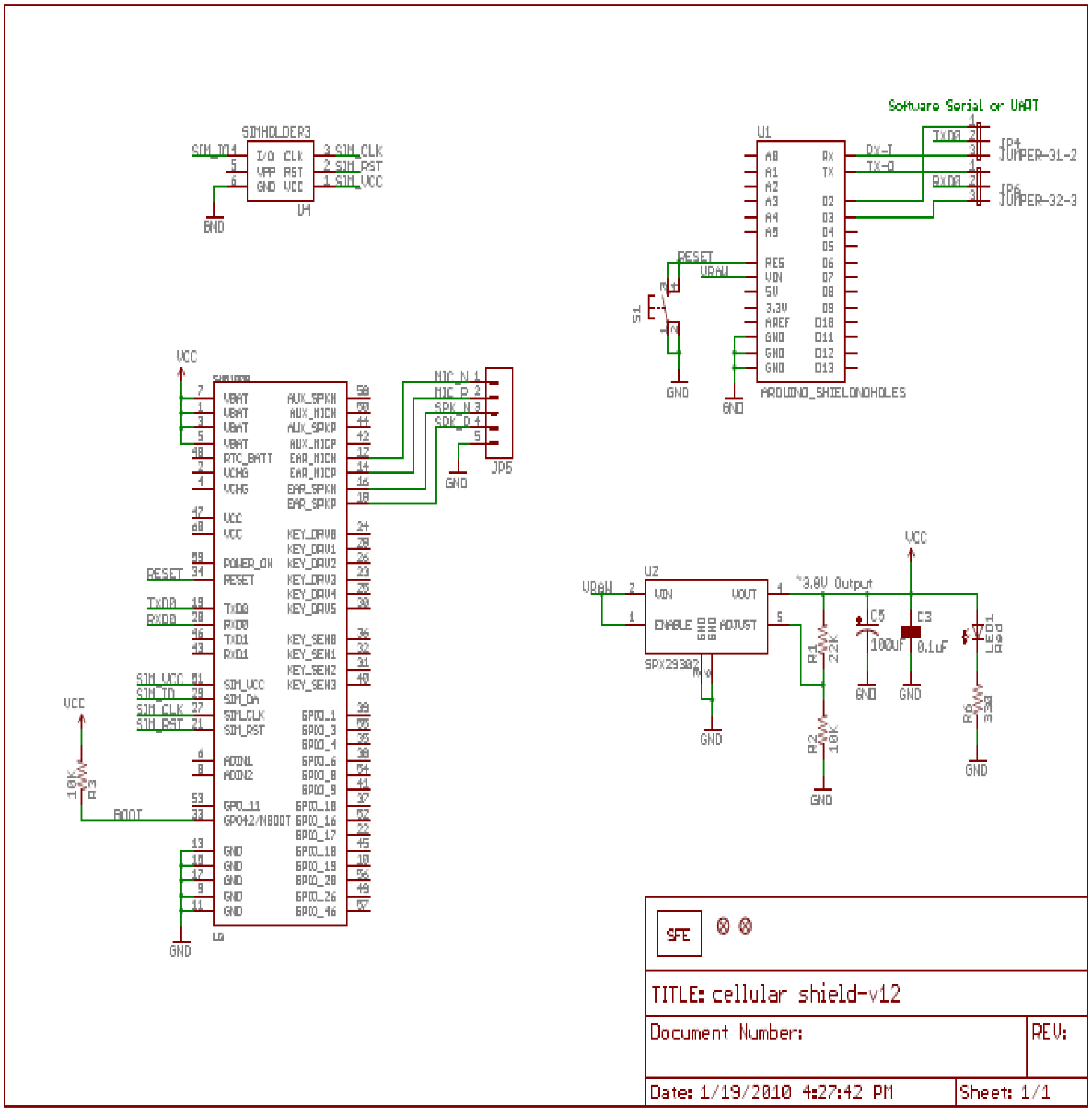


Figure 25: SM5100B Schematics

(Printed with permission from an open source Sparkfun.com)

The main purpose of using the module was to be able to send messages to the emergency contact, in case of an accident. In order to make this happen, we had to use the pins of the module that allows connecting to a cellular network by using a SIM card. Table 16 below shows the description of the pins that were used for this.

Pin number	Pin name	Type	Description
29	SIM_DA	I/O	SIM Serial Data
27	SIM_CLK	O	SIM Clock
21	SIM_RST	O	SIM Reset
51	SIM_VCC	P	SIM Power Supply

Table 16: SIM pin description

The module was also connected to the power pins and to the 6 pins for the UART0 interface to transfer and receive data. These pins were connected to the PCB and were connected to the microcontroller to process the data. Table 17 shows the UART0 pin description.

Pin number	Pin name	Type	Signal	Description
19	TXD0	O	UOTXDN	Transmit Data
20	RSD0	I	UORXDN	Receive Data
10	GPIO19/ U0_CTS/JTAG_TMS	I	UOCTSN	Clear to Send
22	GPIO17/ U0_RTS/JTAG_DI	O	UORTSN	Request to Send

Table 17: UART0 pin description

3.4 Bluetooth Design

The Audio Bluetooth module RN-52 was selected after reviewing a few modules in the research part of the project. This module was made by Roving Networks and it met our requirements as far as Bluetooth class, transfer rate, compatibility with cell phones being synch to the system, protocol compatibility, audio transfer capabilities, and range of coverage. Also, its input voltage could be stabilized with a voltage regulator to have a proper operation. One of the main reasons that made us picked this part was the fact that it was an audio module, not all the Bluetooth offered this feature, also the company that made the module is quite popular in the market, and it had a reasonable price. The Figure 26 shows the schematics of the RN-52 with the names of all its pins. This Class 2 module was purchased from Sparkfun.com, the site offers good info about the part, including its datasheet, and it also sells the unit. This product had a lot of tiny pins, but we were able to place it on the PCB, thanks to the help of the UCF Radio Club. This module was located on the Helmet Tracking System, and it was connected to the microcontroller, so it could be connected to a set of Speaker and a microphone that allowed the interaction with the Cell phone that was synchronized to the project.

Before we placed the module on the PCB, we programmed it and tested it with the microcontroller ATmega328. We decided to purchased the RN-52 Evaluation Board from Microchip Technology to make sure that everything was working as we wanted.

The RN-52 evaluation board was extremely helpful; with it we were able to set up the Bluetooth module according to our convenience. After working on the module for quite a while, we were capable of synchronizing the module with an iPhone 5, and an Android cell phone, to be more precise a Samsung Galaxy S4. Both of the cell phones were able to pair with the unit. The Evaluation Board had a microphone and a headphone jack, so we were able to interact with both of the phone with no problem. The cell phones were synchronized with the Bluetooth RN-52, and we were able to make phone call with them, as well as listening to music. When we integrated the RN-52 audio module to the PCB, we made sure that the unit was placed on a corner of the PCB, with a part of the module sticking out; this allowed a better reception.

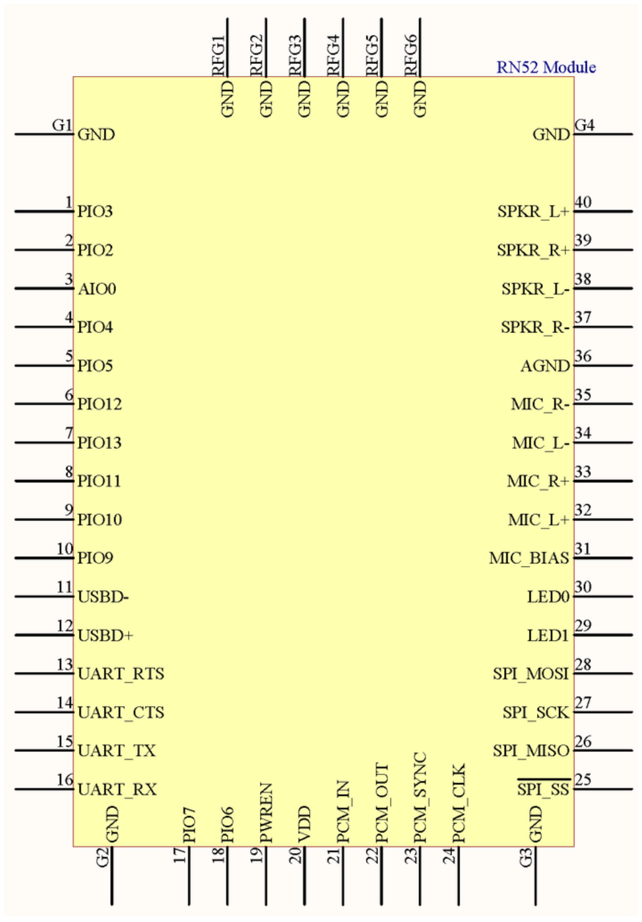


Figure 26: RN-52 Schematic

(Printed with permission from an open source)

Figure 27 shows a simplified block diagram of the audio module RN-52. The module had a lot of pins as we can see on the figure above, the block diagram in Figure 27 helped us analyzed the module in a much easier way. These were the main pins that we connected: The pins named GND were used to ground the needed digital and audio pins; the pin that started with VDD was used as input for internal power; POWERN was the pin that we connected to the external switch used for the system; and all the pins that started with the letters UART were connected properly to transfer data through the UART interface; the pins that we did not use were not connected, or they just ended up being grounded.

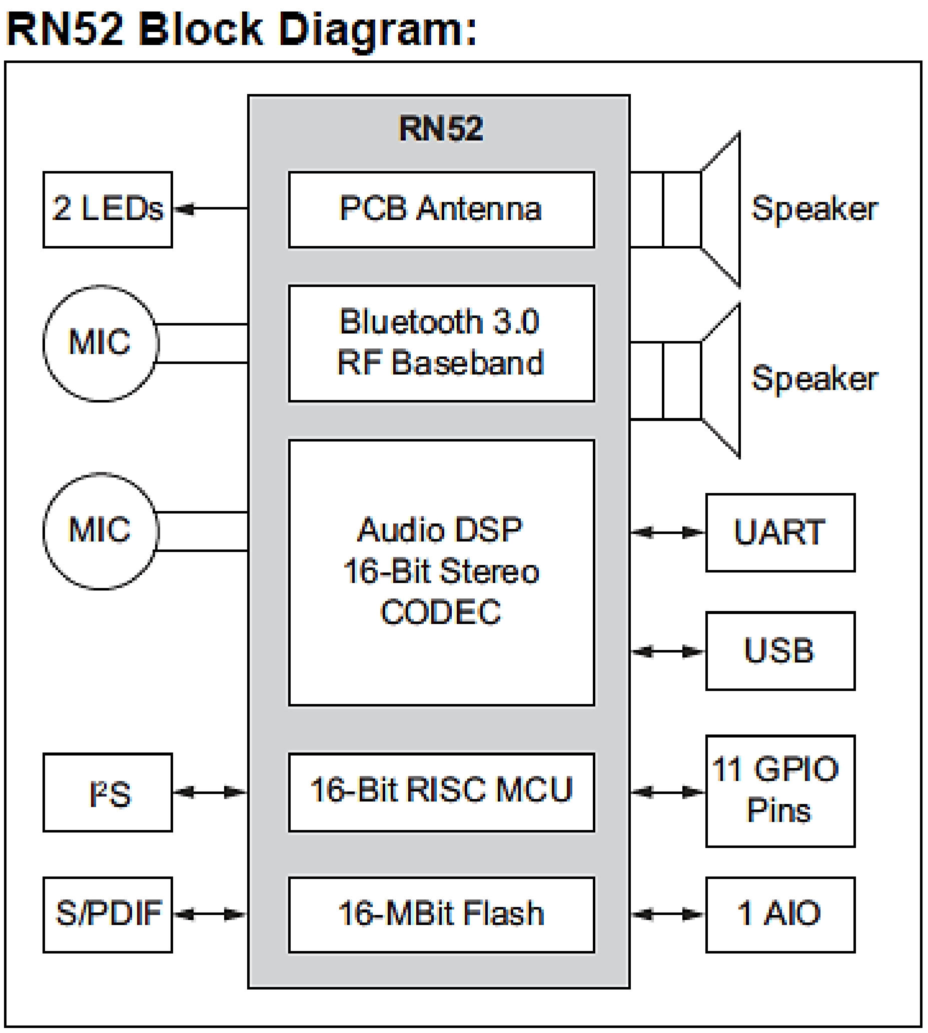


Figure 27: RN-52 Block Diagram

(Printed with permission from an open source Sparkfun.com)

3.5 Battery Selection

The Helmet Tracking System was composed of different modules, such as the Bluetooth, the GPS, the GSM, and the accelerometer; as well as the ATmega328 microcontroller. Before we selected the battery for the HTS we decided to analyze the power consumption of every single component to make sure that we were choosing the most appropriate unit. Table 18 shows our calculations.

Helmet Modules Power Consumption		
Modules	Active mode	Sleep mode
ADXL345	145 µA	40 µA
SM5100B GSM/GPRS	400 mA	2mA
LS20031 GPS	29 mA	2mA
ATmega328p	0.3 mA	0.8 µA

Table 18 HTS Components Power Consumption

After looking at several different types of batteries we decided to go with The Lithium-Ion Battery – 1800 mAh manufactured by Ultralife and sold by the vendor Sparkfun at a price of $29.95. This battery is commonly used in portable electronics and tracking applications, so it was a perfect match for the HTS. It met all the requirements needed for our design; it offered good performance while keeping a slim design profile that allowed us to easily place it on the helmet. The battery average voltage was 3.7 V, and it had a capacity of 1.8 Ah; it was also known for its high energy density, which was 146 Wh/kg. The size of this unit, it was also beneficial to our project; the Lithium-Ion battery 1800 mAh is able to efficiently supply power to the HTS, while keeping a reasonable weight of 46 g, so it is quite light weighted. The figure 28 shows the dimensions of Lithium Ion battery we picked and its dimensions.

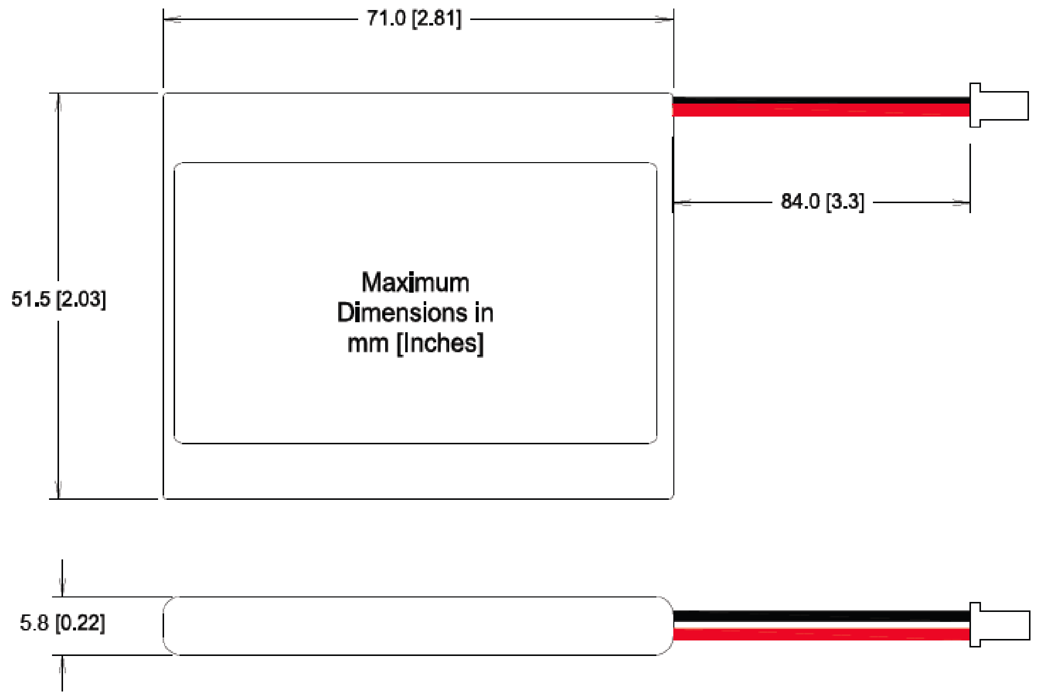


Figure 28: Dimensions of Li-Ion Battery

(Printed with permission from an open source)

After seeing the dimension of Ultralife Lithium Ion battery in mm [inches], it is always good to show the product to have a better idea of it; Figure 29 shows how the selected battery looked like. Also Table 18 below shows some of the main specifications that were important toward our design, and that definitively made us buy this unit. With a capacity rating of 1800 mAh the battery was able to last for 10 plus hours of operation. The components only pulled a small current discharge from the battery when it was in operation mode, also kept well below the max discharge threshold. It could supply what the components needed at a very safe margin.

Figure 29: Li-Ion Battery

Printed with permission from an open source

Specifications
Voltage Range between 3.0 to 4.2V
Average Voltage of 3.7 V
Capacity of 1.8Ah @ C/5 Rate @ 23°C±2°C
Energy Density 146Wh/kg, 317Wh/l
Weight 46 grams
Operating Temperatures between -20°C to 60°C
Self-discharged of less than 10% per month
Recommended charge rate is 900mA to 4.2V in a temperature range of 0°C to 45°C

Table 19: Li-Po battery specs

3.6 Recharge Circuitry

To be able to properly charge the Lithium Ion battery used in our project we had to get an IC chip programmed with the CC/CV method to charge the battery. This made sure to prolong the life of the battery to its maximum use. We decided to go with the MAX1555 SOT23 Dual-Input USB/AC Adapter 1-Cell Li+ Battery Charger manufactured by Maxim Integrated Products for $1.95. It met the requirements to properly charge our Li-Ion battery. The Figure below displays the MAX1555 with its pins labeled.

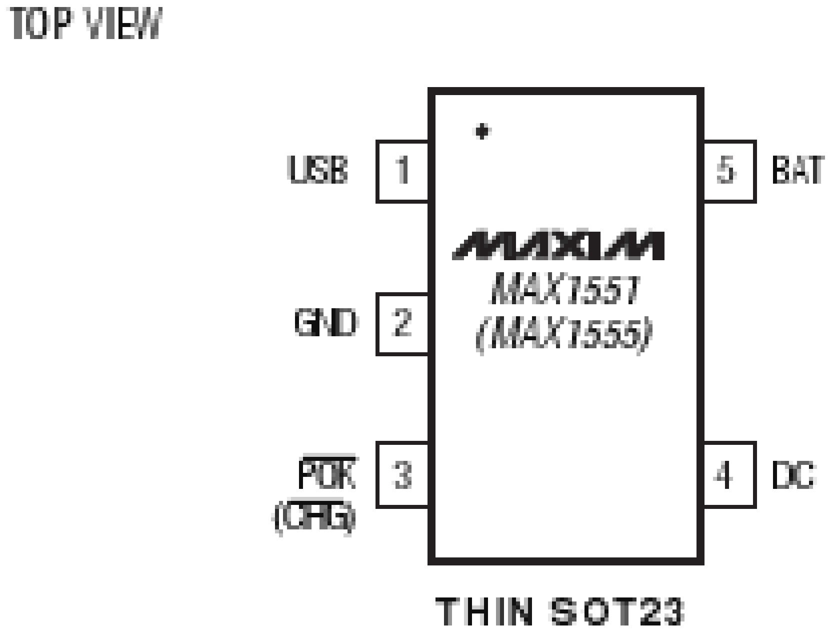


Figure 30: MAX1555 IC with pins

Printed with permission from an open source Sparkfun.com

The MAX1555 had the ability to take two different types of input sources, either a USB source, or a DC input from a wall plug. Depending on which source was used dictated the rate of which the battery was charged. The USB input charged at a current of 100 mA and DC input at a max current of 280 mA. We chose to charge with the DC input source because standard charge current was 200 mA; to be sincere, the USB was not be efficient enough to charge it. The DC source was connected to a USB mini-B since it was one of the most common connections. This helped to keep the device very universal; and it also allowed the device to be charged without the need of any kind of special charger.

Figure 31 shows the circuit suggested by MAXIM to properly install the MAX1555, which really helped us on doing our own circuit; and also the inner workings of the chip itself. The figure is follow by Table 20 that contains the functions of the pins. Where USB and DC were the input source, BAT connects to the battery being charged, ground, and CHG, which were the status indicator. CHG was the active-low Open drain status pin. Once the battery had been fully recharge CHG went high to provide indication.

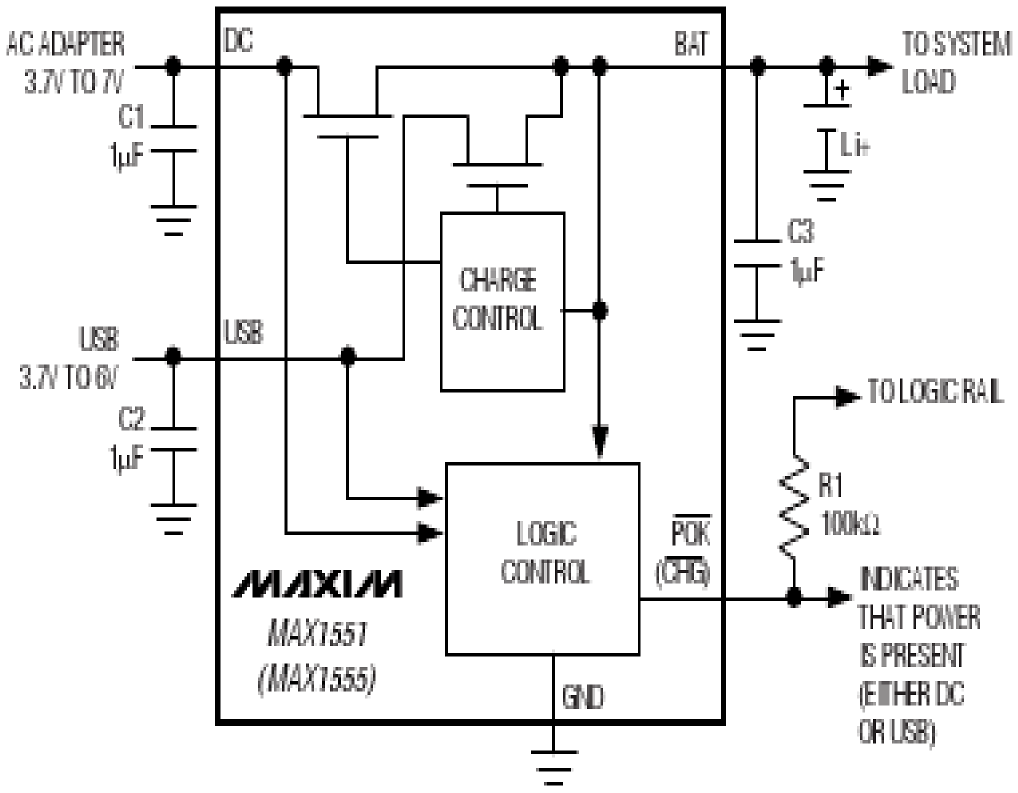


Figure 31: Suggested Circuit with description

(Printed with permission from an open source Sparkfun.com)

Pin	Name	Function
1	USB	USB Port Charger Supply Input
2	GND	Ground
3	CHG	Active-Low Open-Drain Charger Status Indicator pulls low when the battery is charging. CHG goes to a high-impedance state indicating the battery is fully charged, when the charger is in voltage mode and change current falls below 50 mA. CHG is high impedance when both input sources are low.
4	DC	DC Charger Supply Input for an AC Adapter.
5	BAT	Battery Connection.

Table 20: MAX1555 pin function

3.7 Microcontroller Selection

As noted in our research, lots of time was put into researching our options when it came to microcontroller. While some had too much power and or pins, others just simply did not. As far as finding a neutral balance, it was not too difficult to run into. We spoke to previous senior design students and discussed our project idea, function and goals and while we got some mixed opinions, some even a bit biased the majority sent us to the Atmel realm. This made perfect sense as none of us were programmers nor did we have any extensive programming experience. Atmel featured a familiar C language with some easy to use functions, which fit perfectly with what we were up against, as far as programming the GPS, the GSM, the accelerometer and the Bluetooth module, and also bring them together as a whole.

Finding the most adequate central processing unit for the HTS was one of our main concerns; therefore, lots of time was put into researching our options when it came to microcontrollers. While some had too much power or pins, others just simply did not; some of the microcontroller units that we took into consideration were the Texas Instruments MSP430 and some of the ATmega units done by the company Atmel. As far as finding a neutral balance, it was not too difficult to run into. Finally, on figure 32 below is our narrowed down finalized choice for a microcontroller. We chose to go with the ATmega320, and the developing board Arduino One to program it.

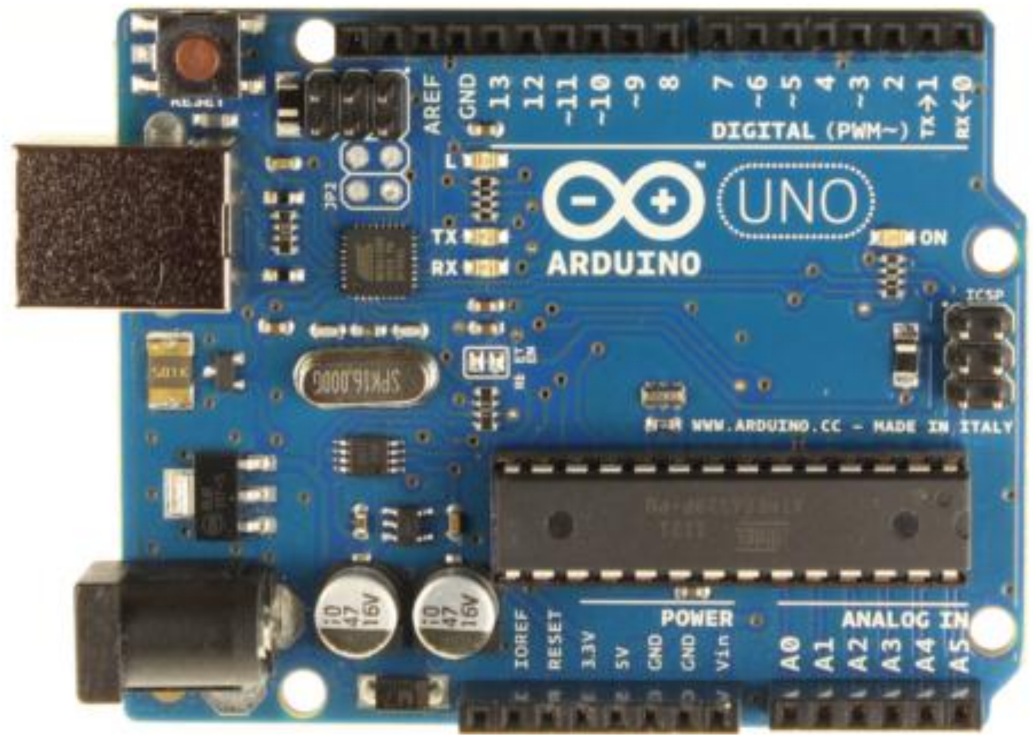


Figure 32- Arduino Uno with ATmega328

(Printed with permission from an open source Sparkfun.com)

This microcontroller unit was mainly chosen because it has a good amount of A/D and digital I/O pins, decent memory, clock speed and fits right in overall with our design specifications. With multiple UART and SPI interfaces and one I2C, it is compatible with ever device we are using in our design. The table 21 shows it main features.

FEATURES
1.8-5.5V Operating Range
Up to 20MHz processing speed
32kB Flash Memory & 1kB EEPROM
2kB Internal SRAM
8 Channel 10-bit ADC
Serial USART/SPI/2-wire I2C Interface
16 MHz of Clock Speed
23 IO lines

Table 21: ATmega328 Features

3.8 HTS Final Design

When it was all said and done, the HTS was an integration of various technologies doing their part to accomplish a subset of goals previously discussed. At a cursory glance, the HTS features an accelerometer, Bluetooth capabilities, GPS, SMS, user control buttons all of which are powered by a 3.8 V Li-ion battery coupled along with a recharge circuit IC. The PCB was designed using Eagle CAD software and built with the help of 4PCB.com. Figure 33 below shows the completed Eagle CAD schematic for the HTS. The components were soldered by our own group members but for smaller modules in the mm range such as the accelerometer and Bluetooth which had tiny pins, more precision was in order. The soldering was completed with the help of the ARC and machines. The PCB was then mounted to the back of the helmet with shock absorbers in their respective place to avoid any possible damage to the components already soldered on the PCB upon impact. The wires connecting the audio I/O were tucked into the cushioning along with the battery making an effort to expose as little as possible for aesthetic reasons.

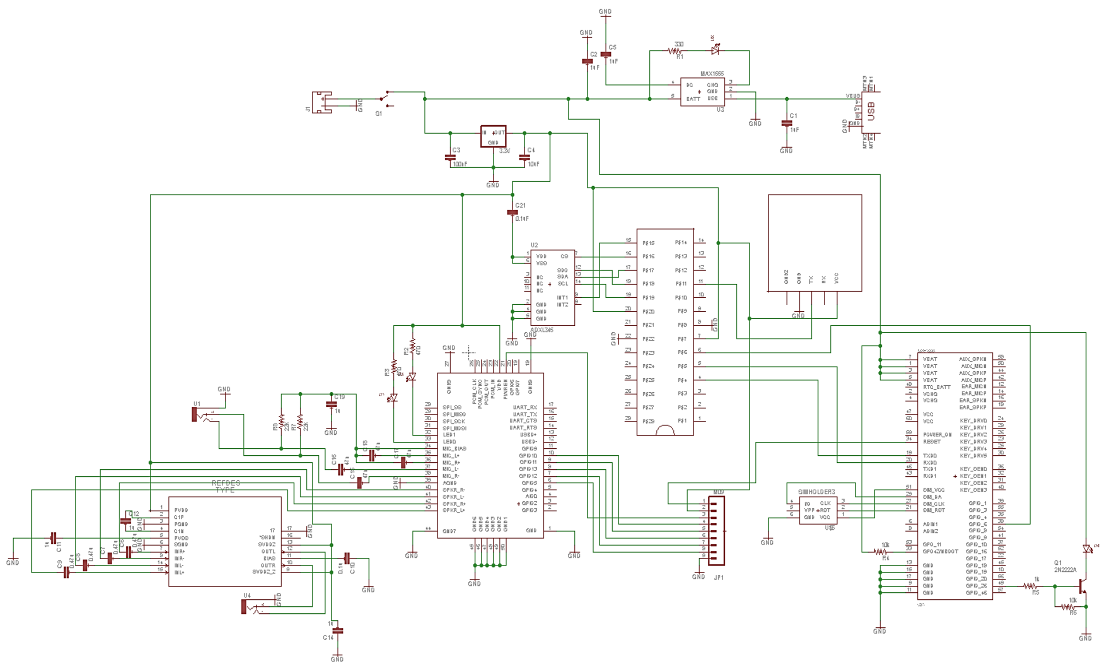


Figure 33- Finalized HTS Eagle CAD schematic

Like any engineering design, there were lots of obstacles in our path to completion among these were some that were able to be assessed and others that despite our best efforts were not able to be resolved within our due time constraints. As mentioned before, some of out main components were problematic when having to solder them on the PCB, the Bluetooth had to be soldered with the aid of a machine for proper precision and even then the end product was not even perfect but when it comes to making such a small PCB, tiny components were in order. Some of our main goals were to keep the PCB as small and unnoticeable as possible, no less than 300 g additional weight and an end profile that would not deform the helmet in any way. After finalizing our traces and double checking our CAD design we were able to come up with a PCB design that had the desired dimensions we had originally intended to aim for. The final dimensions of the HTS PCB came up to be 3.39 in X 3.08in as can be seen on figure 34 below.

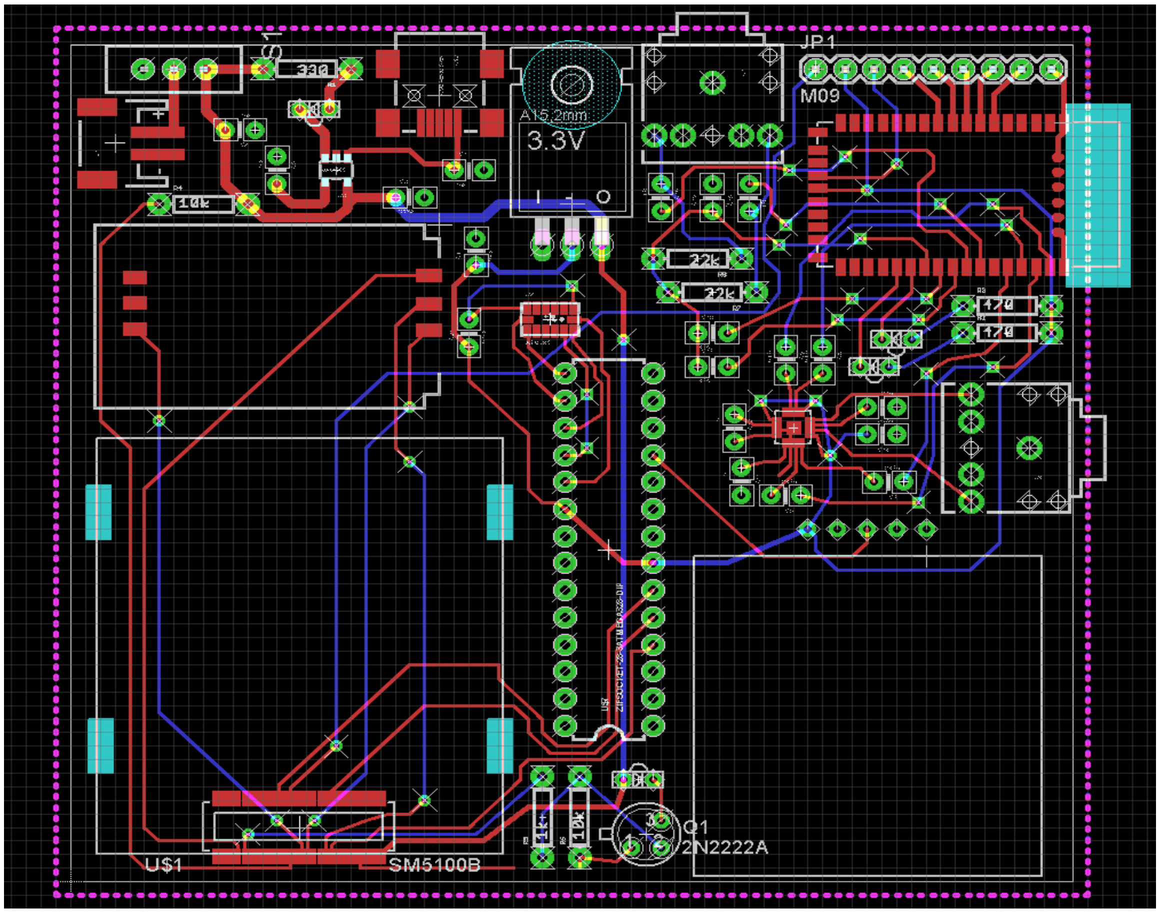


Figure 34- HTS PCB Design

Section 4: Design Summary

The design of the Helmet Tracking System was a collaborative effort as to help improve rider's safety and functionality in their everyday gear usage. We were a team of three Electrical Engineering students with different interests in the many subfields of the discipline therefore we chose a project that could further develop us as engineers and challenge our knowledge while learning about current topics in our respective technology subfields. The main requirements for the project were to incorporate technology and intelligence into an everyday riding gear. Nobody would assume that something so common could provide so much more functionality and further build on its original safety purpose.

As with many other designs, our system was of course a conglomeration of different subsystems but as usual, that was the basis of any engineering system. The main subsystems of our design include GPS, Bluetooth, GSM module, software/hardware, Accelerometer/Motion sensor, power system and a main brain to control all of the signals being sent back and forth; the microcontroller.

Our design is very accurate as far as provide Bluetooth capabilities for audio, and texting an emergency contact in case of an accident, but it is important to note that the user does not have too much control over customizing the system. The system worked and sent signals to the microcontroller with respect to the instructions pre-programmed into the board. As not that complex as the design was, if it were ever to make it into the market for regular consumer use, an additional peripheral would have to be implemented, this being a sort of user interface whether it be a touch screen, voice recognition or more buttons to control some of the basic functions of the HTS.

Perhaps the most basic feature we had on this design was the ability to text an emergency contact, which incorporated the accelerometer, the microcontroller, GPS, and GSM module.. The mechanism was simple, the accelerometer detected that the threshold was passed and then it signaled an interrupt to the microcontroller in which then the microcontroller could wake up the GPS to get the location, and then send data to the GSM module to send a SMS message to the emergency contact.

The GPS will serve a basic function of just keeping track of where the bike and helmet are at all times whether It be normal state or an altered emergency state. Of course the GPS module can give us much more functionality to exploit than what we're currently choosing but the design specification is just to keep track of the bike for monitoring purposes or accidents in which any case the user and his desired contacts will have the current latitude and longitude readings which can be plugged into any mapping software, such as Google Maps, that way the resultant location can be calculated to a respectable accuracy.

Besides providing safety and security the HTS had its features of convenience. For instance, Bluetooth was implemented in the helmet to provide a means of hands free communication for the rider in case of emergencies or leisure calls. The Bluetooth was able to pair to any of the smart phones out in the market that have the capabilities. Once the rider’s device was paired it could transmit the audio from the phone, meaning that it could also listen to music or any kind of audio file. A more important function it gave was that during a case of an accident, the emergency response received the message; it could call at the riders designated cell phone to check for the seriousness of the matter. If the rider was not in a serious accident it could pick up the phone call and explain the situation, but if the accident was severe most likely the rider would not be able to answer the call. When this happens the worst-case scenario will have to be assume and send for help as quickly as possible to the coordinates given.

When all components of the HTS were powered on, let us explain the basic function of the project. The accelerometer was used to detect accidents. The microcontroller gave the program to the accelerometer that changed the sensitivity and the mode the accelerometer was in. The trigger started with the accelerometer passing its threshold. If the threshold was passed, that meant that the rider experienced a g-force typical to that of an accident. Once the threshold was passed, it sent an interrupt signal to the microcontroller. Now the microcontroller sent out its given instructions. It retrieved the GPS coordinate from the module and transmitted it to the GSM to send the SMS message through the network. By having this information it could be relayed to the emergency response contact. The message relayed had information of rider, emergency response could then choose to call the rider if need be to assess further the situation. The Bluetooth integration made it possible for the rider to answer phone calls. The table below gives the task of each of the devices.

Device	**Description**
Accelerometer	It is set detect high levels of g-force. Once threshold is passed it sends interrupt signal to the microcontroller. Also sense for bike orientation after accident. The helmet accelerometer senses if rider is flown off the bike.
GPS module	Retrieves current coordinates and sends it to microcontroller; when the microcontroller receives the interrupt signal from accelerometer.
GSM module	Establishes connection with network. Once it receives signal from accelerometer it sends coordinates from the GPS through the network to its proper recipient.
Microcontroller	Transmits data with all modules of the system. Receives interrupt signal and sends out certain commands.
Bluetooth	To be able to receive calls hands free.

Table 22: HTS components tasks.

The blocks diagrams illustrated below in Figures 35 and 36 summarize the scenarios that could happen when the riders is using the Helmet Tracking System. It is important to notice that the scenario two shown in Figure 36 follow all the steps for Scenario 1, and then adds some more.

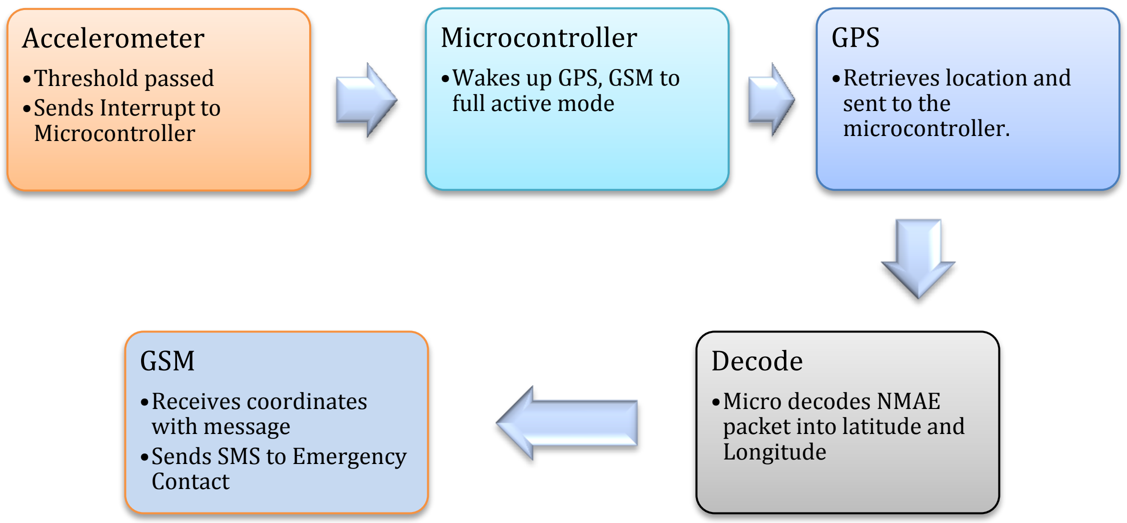

Accelerometer
•Threshold passed
•Sends Interrupt to Microcontroller
Microcontroller
•Wakes up GPS, GSM to full active mode
GPS
•Retrieves location and sent to the microcontroller.
Decode
•Micro decodes NMAE packet into latitude and Longitude
GSM
•Receives coordinates with message
•Sends SMS to Emergency Contact

Figure 35: HTS Scenario 1 – Reporting an accident

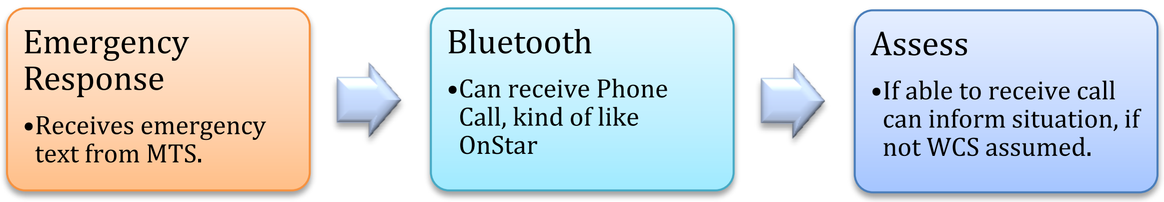

Emergency Response
•Receives emergency text from MTS.
Bluetooth
•Can receive Phone Call, kind of like OnStar
Assess
•If able to receive call can inform situation, if not WCS assumed.

Figure 36: HTS Scenario 2 – Respond from Emergency Contact

Section 5: Testing Procedure

5.1 GPS Subsystem Test

The GPS subsystem was one of the various subsystems that incorporated the Helmet Tracking System (HTS). Tracking the geographic location of the helmet was the main purpose of it; in fact, it was able to find the receiver anywhere in the world. The first thing that we tested was the ability of the receiver to effectively fix, which was pretty much making sure that the module was properly connecting to the satellites. The second aspect that we checked was how long the receiver took to lock on a satellite and started transmitting coordinates; this was a cold start test. To do the cold start test we unplugged the unit to make sure it did not get any power, and we left it like that for a day. After that, we checked the connecting time, we repeated this process for three days to get a little more accurate values. The third thing that we took into consideration was the location, the rider was utilizing our project in different places; therefore, we decided to test the GPS in different locations around town to make sure that we were getting the correct output at all the time, we would like to test the product on different cities or states, but it required us to spend more money to pay for the travelling expenses, so we did not do that. Some of the places that we used to test the unit were:

- The first floor of the UCF library. Testing in this place showed us that the unit was working on an indoor location.
- The second floor of the UCF parking garage C. This was a mixture of an indoor and outdoor location, which was one of the environments that we wanted to test. Also the motorcycles were often park in these places, so having a good reception on a parking garage was important.
- One of the streets of downtown Orlando. Making sure that the receiver was getting signal outdoors.

We checked the geographic location outputs that we got in all these places, and we compared them to the values that the free cell phone app Google maps provided, just to make sure our subsystem was accurate. We were expecting to not have a difference greater than 3 meters when we check latitude and longitude, and this was achieved.

5.2 Motion system testing

Before integrating the accelerometer to the project we first had to do the proper tests to make sure the sensor was working like it was supposed to. Each of the interrupt functions also had to be tested since it was a crucial part of the task the accelerometer had to provide. Luckily the accelerometer came with its own Self-Test feature that we could use to confirm the operation of the device. The Self-Test feature effectively tested its mechanical and electronic systems. The register DATA_FORMAT in address 0x31 held the bit that turned on the Self-Test bit. After the bit was set to 1 then the device put an electrostatic force on the mechanical sensor. This force applied to the accelerometer behaved in the same manner acceleration would. If the device was already undergoing some kind of acceleration, the force applied by the Self-Test was added on to the present acceleration. In other words the current acceleration at that time was offset by the value on the Self-Test. However these Self-Test values all had a certain threshold limit for each range of g-force that it was in. The Table below shows the upper and lower bounds of the Self-Test function. For the proper function of the Self-Test mode, the output data rate should be set to 100 Hz or higher.

G-Force Range	Axis	Min (LSB)	Max (LSB)
±2g	X	50	540
±2g	Y	-540	-50
±2g	Z	75	875
±4g	X	25	270
±4g	Y	-270	-25
±4g	Z	38	438
±8g	X	12	135
±8g	Y	-135	-12
±8g	Z	19	219
±16g	X	6	67
±16g	Y	-67	-6
±16g	Z	10	110

Table 23: Upper and Lower bounds of Self-Test mode

Before we could even try the Self-Test function we needed to initialize the accelerometer with the proper coding that made it able to store and see the data that the device was producing. The right file libraries needed to be uploaded and the right registers needed to be set. All of this was done through the code set in the microcontroller. We set up a very basic code to make sure that the accelerometer was working properly.

First we started with the header, where all the parameters were initialized. Below was a section of the code that showed the initialization process. The SPI.h library was added to the sketch.SPI, this was the communication protocol that the microcontroller used to be able send data and retrieve it from the accelerometer. Next thing we had to initialize was the CS pin, which it needed to be set for data to come out of the accelerometer. Now that we had established the communication, we had to create variables for the registers that were going to be used. Variables were also created to store temporary data and use it for certain functions.

This second section was used to configure the other different aspects that were needed for proper communication with the microcontroller. There were two functions that were going to be enabled to be able to get data from the accelerometer, the g-force range and measurement mode. With these two functions on we now implemented a loop that was going to be reading the data being produced by the x, y, and z-axis. The first 8 bits were put in its proper register and it took the remaining two, because this accelerometer read with 10 bits of resolution, and moved them to a different register. Following the loop were print functions that were going to display the data it was receiving from the accelerometer. By giving it a 10-millisecond delay it ensured that the loop was running at 100 Hz.

Now that we were able to read data from the axis and put them in the accelerometer, we needed to be able to extract that data into the microcontroller. The command that does this was read Register function, which needed three parameters to go with it. It needed the register address, the number of registers that should be read, and where the values should be stored. It started the communication between the two devices the CS pin must be set low. Once it was low the following commands took action and took data from the accelerometer. When the communication was finished the CS pin was set to high again.

Finally it displayed the data on the terminal window of the microcontroller. If we were getting readings it meant that the code was working and so was the accelerometer. But we still had to make sure they were appropriate readings. Now that the data was continually displayed, we could start by moving the accelerometer and see the changes that took place. Once that confirmed we could then set the Self-Test bit on to check if the values that were being displayed were the proper ones. If everything confirmed then we knew that the accelerometer was working as it was design too and the code could be changed to the one needed to perform the tasks for the design.

5.3 GSM Subsystem Test

In order to test the GSM module SM5100B, we used the Cellular Shield (SM5100B) with Arduino, this was pretty much a subsystem already build that had all the capabilities that a regular phone had; and it was comparable with our microcontroller, Figure 32 shows how this Shield looked like. The main components of the Cellular Shield were a 60-pin SM5100B connector, a SIM card socket, and an SPX29302 voltage regulator configured to regulate the Arduino's raw voltage to 3.8V. The board's red LED indicates power. The Arduino's reset button was also brought out on the shield. The Cellular Shield already had a socket for the SIM card for easy integration, and we were just going to use one of our already activated SIM cards to test if it did receive a signal and operated properly.

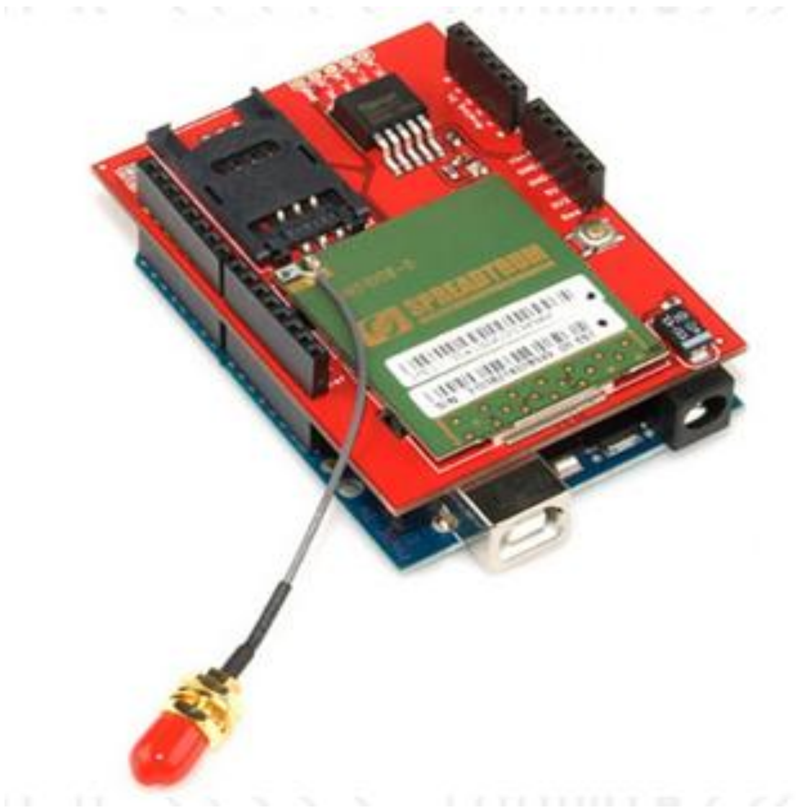


Figure 37: the Cellular Shield (SM5100B) with Arduino

(Printed with permission from an open source Sparkfun.com)

5.4 Bluetooth Subsystem Test

To test the Bluetooth module, we did a Loopback testing. First of all we wired up the module properly to the Arduino board, the Arduino board was getting the supply power from its power source, then we shorted circuit the rx and tx pin. By doing this we got exactly what we sent. Figure 33 below shows the standard proper wiring between and Arduino and a Bluetooth module.

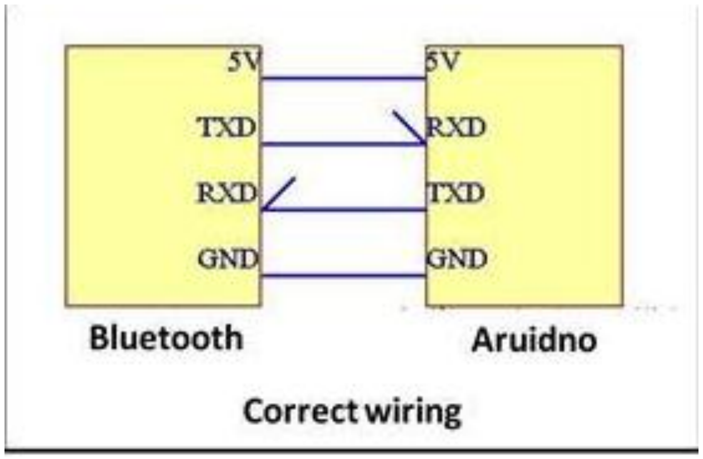
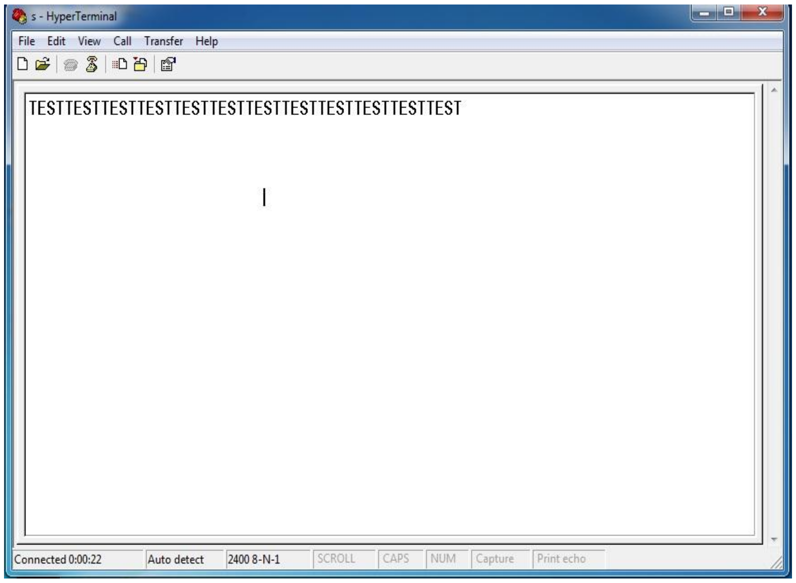


Figure 38: Correct Wiring

Printed with permission from an open source Sparkfun.com

The second part to the test was having the Bluetooth connected to the ATmega328 we needed to give it commands and see how the module responded to it. Bluetooth module came with Self-Test coding that needs to be implemented to the ATmega328 microcontroller.

The code was implemented using the Arduino environment, which was the one used by the ATmegaa328 microcontroller unit. The testing code started by configuring the ANSEL and ANSELH pins as digital pints; then the UART module was initialized at 9600 bps with a delay of 100 ms. Finally, a while loop was implemented and it displayed the word TEST every 2000 ms. The figure number 35 was an example of the output If device is working properly, and the testing went accordingly the desired output shown in the figure below will be displayed.



```
TESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTESTTEST
```

```
Connected 0:00:22   Auto detect   2400 8-N-1   SCROLL   CAPS   NUM   Capture   Print echo
```

Figure 39: Bluetooth output result

Finally, the RN-52 was a Class 2 Bluetooth module with Audio transfer capabilities, so once we got it working on the evaluation board, we were able to pair it to a cell phone to play audio and make and receive phone calls.

5.5 Microcontroller Testing

There were many different ways to test the microcontroller unit, one of course just ran some code in the IDE and checked that it compiled correctly and ensured that the microcontroller itself was working properly but in reality a more effective way would be to test every single peripheral along with the microcontroller to check and make sure that module was interacting correctly with the microcontroller. We proved this by testing with the gps module and the microcontroller together.

We also implemented a basic LED test with the Arduino. The purpose of this code was to transfer data; and every time a character was transfer, and LED that was connected to the Arduino MCU lighted up, as you can see on Figure 40.

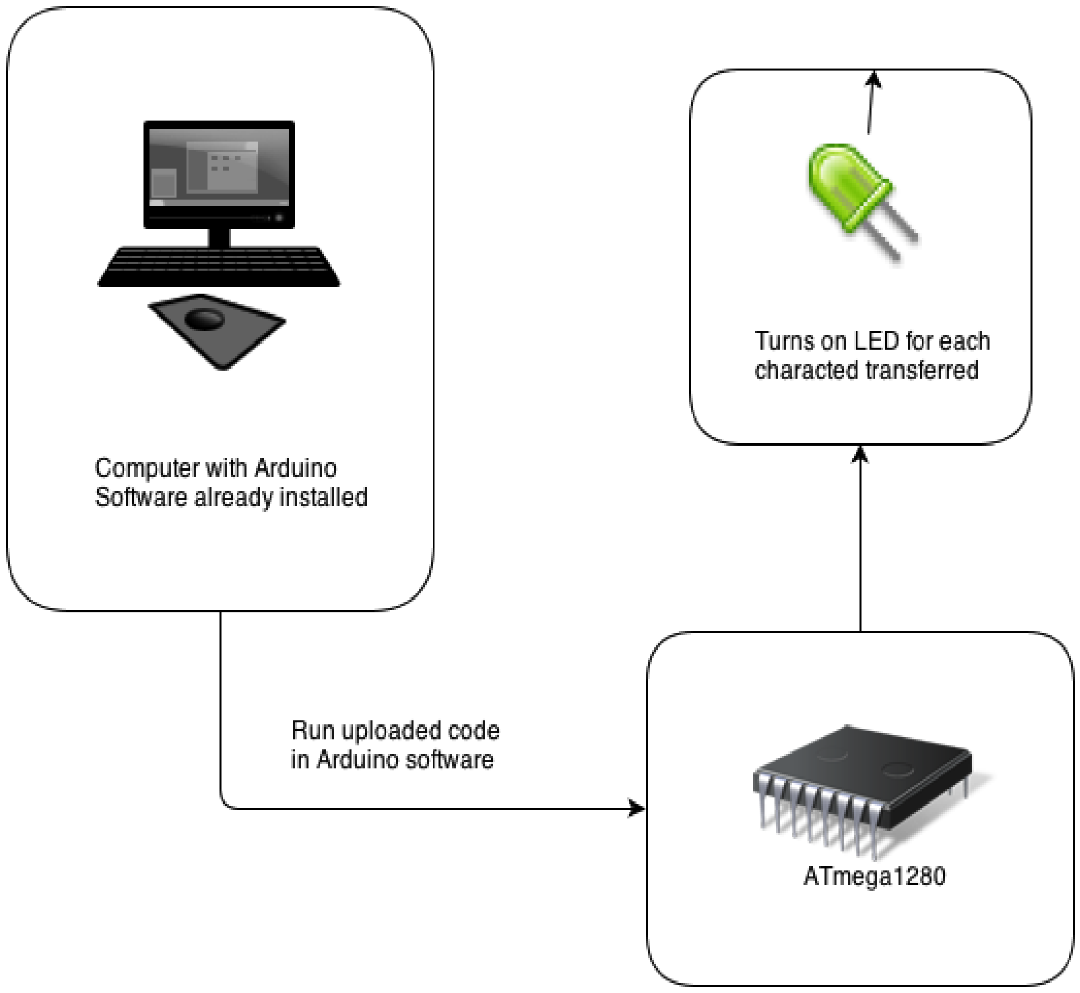

Computer with Arduino Software already installed
Run uploaded code in Arduino software
ATmega1280
Turns on LED for each characted transferred

Figure 40 Block Diagram of Arduino Testing Code for MCU

After we compiled this program and uploaded it to the Arduino. On the PC, we could use X-CTU software and click on the terminal tab to monitor data received wirelessly from the Arduino. On the Arduino side, we could go to the IDE and use the tools menu to open the console window to see the Arduino printed the characters as they were sent. Overall, the code that we were implementing made alphabet characters appear once a second in both windows. One was the console, attached to the transmitting Arduino; the other one was the terminal window of the X-CTU software on the PC side, receiving the characters sent wirelessly.

There were also some simple Examples that came with the Arduino software that could be used to test the microcontroller unit without the need to connect other external components to the MCU. One of them was the popular Blink code that turned on one of the little lights that were integrated in the ATmega328 board, when you connected the unit to the computer via USB, and of course you had to upload and run the code in Arduino.

Section 6: Administrative Content

6.1 Budget and Financing

The Helmet Tracking Security System was a Senior Design project that was self-funded; therefore, we tried to keep the spending within a reasonable amount. Some of the modules required to properly build this prototype were expensive, so we were as careful as possible when we dealt with them, so we did not have to replace them. The table below shows the prices of the parts that we selected for the HTS.

Item Description	Supplier	# Units	Cost
66 Channel LS20031 GPS 5Hz Receiver	Sparkfun	1	$59.95
SIM Socket	Sparkfun	1	$0.95
SM5100B Mating Connector	Sparkfun	1	$2.95
Cellular Shield with SM5100B	Sparkfun	1	$99.95
Quad-band Cellular Duck Antenna SMA	Sparkfun	1	$7.95
Arduino Uno - R3	Sparkfun	1	$29.95
Rn-52 Bluetooth Audio Module	Sparkfun	1	$24.95
Battery Charger	Sparkfun	1	$1.95
Lithium Ion Battery 1800 mAh	Sparkfun	1	$11.95
USB Mini-B connection	Sparkfun	1	$1.50
USB Mini-B Cable	Sparkfun	1	$3.95
ON/OFF switches	Sparkfun	2	$1.00
LEDs	Sparkfun	3	$1.20
ADXL345	Sparkfun	1	$14.95
Atmega328p	Microcontroller Pros	2	$9.00
RN-52 Evaluation Board	Microchip Technology	1	$186.93
PCB	4pcb.com	1	$37.50
		Total	$519.58

Table 24: Budget and Finance

6.2 Timeline

In order to have a successful Senior Design Project, the way we managed our time was extremely important. The tables 25 and 26 show how we organized our schedule for Senior Design 1 and 2.

Senior Design I

Date	Task Description
February 11th	Research on which microcontroller would be best for the project.
February 18th	Research on GPS/GSM module for both motorcycle and helmet and its circuit requirement
February 25th	Research on which type of module to be used for the helmet to motorcycle communication.
March 4th	Research Bluetooth module to implement in the helmet and the components that are compatible with it.
March 11th	Implementation of battery pack with a recharging circuit design.
March 18th	Accelerometer implementation and programming into microcontroller.
March 25th	Alarm System Programming.
April 1st	PCB layout Design to implement all components together properly.
April 15th	Put all materials together and touch up on the last details.
April 22nd	Turn in Final copy of Research Paper

Table 25: Senior Design I Timeline

Senior Design II

Date	Task Description
May 13th	Start Ordering and getting parts together.
May 20th	Start Programming each of the components and implementing them in the board.
June 3rd	Program the second board that goes into the helmet, and tune them to communicate with each other.
June 10th	Program all the components to work with the specified requirements.
June 24th	Place the board for the motorcycle in special modeled box, and place board for the helmet with components.
July 8th	Put it all together to have a working prototype
July 15th	Fix all the bugs to make sure it works to specifications
July 26th	Present working project.

Table 26: Senior Design II Timeline

Section 7: Appendix

7.1 Work Cited

Getting the proper information was a key factor to come out with the design of the Helmet Tracking System. These are the websites that we used to do our research.

[1] Arduino, Website:

http://www.arduino.cc

[2] Roving Networks, Website:

http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Wireless/Bluetooth/rn-52-ds-1.1r.pdf

[3] Accelerometer Datasheet Sparkfun, Website:

https://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf

[4] GSM Datasheet Sparkfun, Website:

https://www.sparkfun.com/datasheets/Cellular%20Modules/CEL-09533-User's%20Manual.pdf

[5] GPS Datasheet Sparkfun, Website:

https://www.sparkfun.com/datasheets/GPS/Modules/LS20030~3_datasheet_v1.2.pdf

[6] Code-42, Website:

http://www.code-42.blogstop.com/2011/01/li-po-batteries-explained.html

[7] Diagnosticnews, Website:

http://www.diagnosticnews.com/featured/parasitic-battery-drains/

[8] Dimensioning-Engineering, Website:

http://www.dimensionengineering.com/info/accelerometers

[9] Analog Devices, Website:

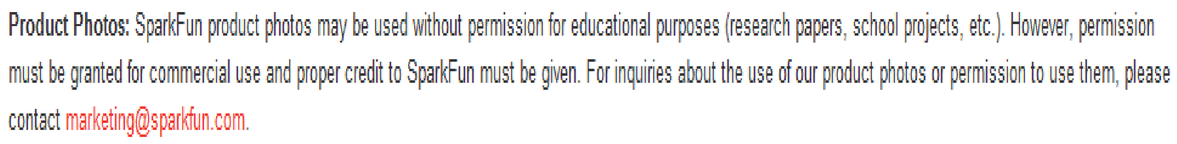
http://www.analog.com/en/content/td_accelerometer_specifications_definitions/fca.html

[10] Battery University, Website:

http://batteryuniversity.com/learn/article/understanding_lithium_ion

7.2 Permissions

Sparkfun



Product Photos: SparkFun product photos may be used without permission for educational purposes (research papers, school projects, etc.). However, permission must be granted for commercial use and proper credit to SparkFun must be given. For inquiries about the use of our product photos or permission to use them, please contact marketing@sparkfun.com.

Wikipedia

Terms of Use

This is a human-readable **summary** of the Terms of Use.

Disclaimer: This summary is not a part of the Terms of Use and is not a legal document. It is simply a handy reference for understanding the full terms. Think of it as the user-friendly interface to the legal language of our Terms of Use.

Part of our mission is to:

- **Empower and Engage** people around the world to collect and develop educational content and either publish it under a free license or dedicate it to the public domain.
- **Disseminate** this content effectively and globally, free of charge.

You are free to:

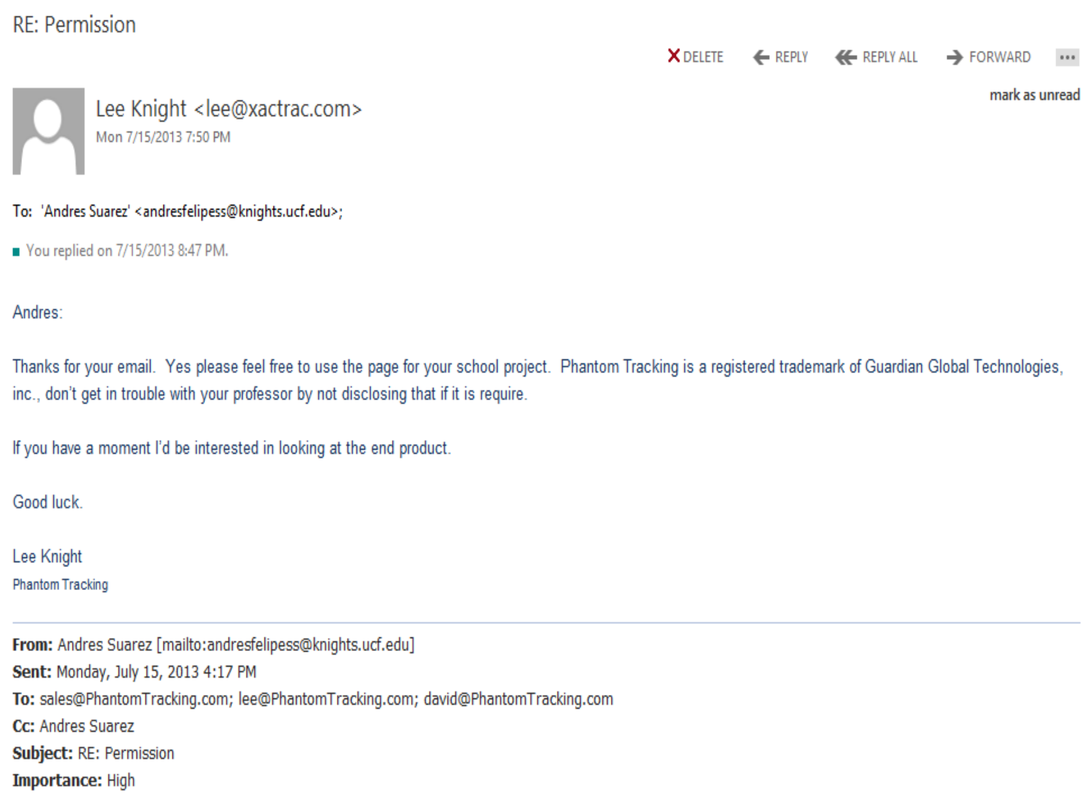
- **Read and Print** our articles and other media free of charge.
- **Share and Reuse** our articles and other media under free and open licenses.
- **Contribute To and Edit** our various sites or Projects.

Under the following conditions:

- **Responsibility** – You take responsibility for your edits (since we only *host* your content).
- **Civility** – You support a civil environment and do not harass other users.
- **Lawful Behavior** – You do not violate copyright or other laws.
- **No Harm** – You do not harm our technology infrastructure.
- **Terms of Use and Policies** – You adhere to the below Terms of Use and to the applicable community policies when you visit our sites or participate in our communities.

With the understanding that:

- **You License Freely Your Contributions** – you generally must license your contributions and edits to our sites or Projects under a free and open license (unless your contribution is in the public domain).
- **No Professional Advice** – the content of articles and other projects is for informational purposes only and does not constitute professional advice.


RE: Permission

DELETE REPLY REPLY ALL FORWARD ...

mark as unread

Lee Knight <lee@xactrac.com>
Mon 7/15/2013 7:50 PM

To: 'Andres Suarez' <andresfelipess@knights.ucf.edu>;

You replied on 7/15/2013 8:47 PM.

Andres:

Thanks for your email. Yes please feel free to use the page for your school project. Phantom Tracking is a registered trademark of Guardian Global Technologies, inc., don't get in trouble with your professor by not disclosing that if it is require.

If you have a moment I'd be interested in looking at the end product.

Good luck.

Lee Knight
Phantom Tracking

From: Andres Suarez [mailto:andresfelipess@knights.ucf.edu]
Sent: Monday, July 15, 2013 4:17 PM
To: sales@PhantomTracking.com; lee@PhantomTracking.com; david@PhantomTracking.com
Cc: Andres Suarez
Subject: RE: Permission
Importance: High

Webopedia.com

Stay Connected: Free Computer Dictionary Technology Tools and Widgets

Tech Tools For Web Publishers

Webopedia offers free Webmaster tools and RSS feeds so you can use our technology content on your own Web site or blog. Embed our computer dictionary search engine on your page with cut-and-paste code or educate your readers with the Webopedia Term of the Day button and tools. Using Webopedia links means that you can trust the content. With our search engine and gadgets on your site, your readers will see accurate technology references and definitions.

Connections for Readers

Our readers stay connected with Webopedia many ways: our daily and weekly newsletters, Webopedia Google gadget on your Google search homepage or our

RSS feed. We'd love to hear from you, so reach out to the Webopedia editorial staff on Twitter or join the discussion on the Webopedia Facebook Wall.

From: Felix Teodoro <felixt@chatterboxusa.com>
Sent: Tuesday, July 16, 2013 11:23 AM
To: Andres Suarez
Subject: Re: Permission

Hello Andres,

You have our permission to use our picture and information for your paper.

Best,

On Mon, Jul 15, 2013 at 4:35 PM, Andres Suarez <andresfelipess@knights.ucf.edu> wrote:
Hi,

My name is Andres Suarez and I am a Senior Electrical Engineering Student at University of Central Florida (UCF). My team is building a smart helmet system with GPS and some other features. We had to do a research paper, and wrote a little bit about similar projects. We copied a picture of your chatterbox, and spoke a little bit about it (about 1 page).

We just want permission to submit this to our professor.

Thank you

Andres Suarez, Brian Maldonado and Rigel Jimenez.

EE Students UCF