KnightCop

Remote Controlled Scout Robot

Senior Design I

Summer 2013

Group 11

Elean Atencio

Nitin Kundra

Aug 1, 2013

Table of Contents

1.0.0 Executive Summary

The primary goal of Knight Cop is to be an extension of law enforcement in the field. Our aim is to build an affordable robot that is functional enough to merit usage in life threatening scenarios but is also dispensable, unlike an officer.

Our robot will be based on a chassis we will be building ourselves. This will allow us to conform to the weight and size restrictions imposed by other components. The chassis will also house the PCB, a video camera and temperature, light and proximity sensors that will continually provide feedback to the user. The video camera is necessary to survey crime scenes or disaster zones. Temperature sensors would allow us to determine if the site is safe for deploying personnel. Ambient light sensors would allow us to switch on the onboard light or toggle the camera to ‘night vision’ mode. Proximity sensors allow us to add automation to the robot. It will be able to assist the operator in navigating across inhibiting terrain even if communication link with the operator is broken. This will also potentially allow a cluster of KnightCops to be used in a recon mission without burdening multiple operators to control each and every robot.

The defining feature of KnightCop is the robot arm that will be mounted on the base. We will be purchasing this arm and modifying it to suit our needs. Instead of going for a powerful arm that can carry heavy payloads (which would needlessly add more bulk to the robot). We chose a lighter arm which has more degrees of freedom and allows us to showcase the dexterity and precision we wish to implement in the robot. The arm has a split hook that allows it to interact with foreign objects. The arm will also demonstrate several pre-programmed movements - such as moving the payload from point A to point B. The robot undercarriage and the arm will be remote controlled, allowing the operator to stay out of harm's way. A MS Windows based PC will act as an intermediary between the robot and the controller, which will be an Android phone. All communication will be handled over the Wi-Fi protocol. This allows us sufficient range to operate the robot from and enough bandwidth to relay video and sensor feeds. It also keeps the communication channel between the devices uniform.

The software development would be primarily handled in Atmel Studio 6 and Eclipse IDE. The languages of choice for these are C and Java respectively. Microcontroller programming would be done in C since most libraries relevant to our chosen components are available in C. Android development is done in Java and we decided to preserve code uniformity and portability by implementing the desktop software in Java as well.

We wish to accomplish aforementioned goals while respecting the time and monetary constraints, but without sacrificing on features and performance. Thorough testing of the hardware and software components will be done to ensure that the robot functions reliably and will not fail at a critical moment out in the field.

2.0.0 Introduction

2.1.0 Motivation

On a daily basis, Law enforcement members face numerous threats as they perform their duties. Law enforcement members routinely face risk such as gunfire, explosions, and suspicious packages. Even the risk associated with the aftermath of natural disasters can be quite high since these situations often bring officers in contact with such things as live electric wires, gas leaks that cannot be seen easily sensed, and unstable structures. Wide spread damage to areas brings about a sense of lawlessness in some individuals. Officers are often first responders and placed in danger while in this role. Some of these threats can result in serious bodily harm or the loss of a law enforcement member's life. We believe that some of these threats can be effectively dealt with by machine. Using technology with the aim of creating a robot capable of performing specific high-risk task, we believe, is in the best interest of society. Whenever the threat of loss of life or great bodily harm can be reduced, everyone benefits.

We envision KnightCop as a means to accomplish this threat. We are motivated to use the technological skill we have acquired to lessen these risk to life. To make KnightCop possible, our group will have to come together as a unit. Our experiences and learning will have to work for us now. We will design KnightCop to enter those potentially dangerous situations and respond back to its user. Several functions will allow KnightCop to relay important information. KnightCop will be capable of determining the temperature and CO levels of its environment. A transmission device mounted on KnightCop will be used to relay back what the sensors detect. KnightCop will need to be mobile enough to enter areas containing debris. Video and audio signals will allow law enforcement members to see and hear activities within the red zone.

The group's intention is for Knightcop to be a solution that is affordable enough so that the financial risk of having it destroyed or damaged is negligible when compared with the alternative of having an officer seriously injured or killed. Although the effort to preserve human life is of the highest importance, the cost of needed devices must still be considered or the design would be useless. We will pay attention to not only our development cost, but we will also consider the cost of reproducing KnightCop for others. Thus, we intend on making Knightcop affordable enough that even smaller law enforcement departments with small budgets can easily obtain it. We strive to build a robust yet affordable robot.

Knightcop will be designed to take the place of the officers in response to certain high-risk situations. It would need to function well enough that officers feel confident about remaining in a safe zone, while directing Knightcop to perform whatever duty is needed at the time. It would need to be relatively easy to operate and direct during an emergency situation. During these types of situations, the law enforcement member may not only need to be operating

Knightcop but also staying aware of the surroundings so that KnightCop isn't placed into unnecessary harm.

2.2.0 Objectives and Goals

Through remote operation, KnightCop can be deployed into areas deemed having an unnecessarily high risk for human injury human. The movement of KnightCop itself as well as its arm will be controlled independently. Remote control operation will be possible in several ways. By either using a modified game controller, a smartphone or our team-designed graphics user interface commands can be sent to KnightCop. No matter which mode is used to interface KnightCop, all data communication will be performed wirelessly.

KnightCop will be expected to travel into areas having very different terrain. We aim to design KnightCop to be able to move over various terrains.

The video and audio signals of the surveillance area will allow a user to monitor the hazardous area remotely. The user will have the ability to view a color video feed with a 640x480 resolution. The device will be capable of being programmed to move as directed or allowed to perform limited, autonomous actions. KnightCop will be equipped with sensors. The sensors will detect heat, CO_2 levels and objects.

The arm will have the capability of moving several degrees. The split-hook hand will be capable of grabbing objects with its pincher-type grip.

Any information gathered from any video, audio, or sensor readings will be shown on a screen. Information gathered by KnightCop will be sent to a remote location wirelessly.

2.3.0 Project Requirements and Specification:

2.3.1 General

We came together to think of what capabilities KnightCop should possess. We knew it could be equipped with a variety of sensors. KnightCop would be expected to travel across many different types of surfaces. We focused on how KnightCop would interact with its user. After exchanging ideas, a list of KnightCop's requirements began taking shape.
KnightCop will be required to:

- navigate across smooth and debris-filled surfaces
- possess wireless control
- stream video to base station
- possess night vision video capabilities
- possess an intuitive user interface
- possess a panning camera
- possess an external light
- have the ability to transfer small items

Specifications

The general desires were from above were made into specific requirements. Our initial specifications are displayed in table1

Maximum Dimensions	**l x b x h**	30 x 30 x 30 in
Operating range		50 m
Operating frequency		2.4 GHz
Power supply voltage		12 V
Minimum Speed		1 m/s
Maximum Robot weight		30 Kg
Arm		
Minimum Lifting capacity		100 g
Rotation span		120 degrees
Gripper states		open/close
Video feed	**30 fps**	640 x 480 px

Table1 – Specifications

2.3.2 Wireless Communication

Our chief requirement in a communication system is wireless connectivity, decent range and bandwidth. Most modern protocols have a decent power footprint. Availability, too, is not a major concern. Infrared is ubiquitous in remote controls. RF is extensively used in RC cars and planes. Bluetooth is found in virtually all smart phones and portable computers. Wi-Fi is a staple wherever higher range and bandwidth is desired. Our primary wants in a communication system were:

a. Native encryption support:

Encryption is a must since we do not typically want video and other sensitive information from a disaster or crime scene compromised. Although this adds an overhead, modern PCs and smart phones are more than capable of handling it.

b. Long range operation:

The robot must be controllable over a distance deemed safe enough from armed criminals, radiation or minor explosions. Furthermore, there must not be any degradation in communication quality over this specified range.

c. High bandwidth:

The communication channel must be able to support two way exchange of data without choking. One of our primary features is video feedback, which would take up considerable bandwidth if the capture resolution is increased. We would also like to maintain some breathing room to add more sensors and features.

d. Low power consumption:

Since the robot is remote controlled, the wireless communication system will always be powered on. It then becomes critical that it not draw too much power from the battery.

e. Inter-operability:

The communication system should ideally stay the same among the three devices (KnightCop/PC/Android phone). This gives us the option of using the PC as the controller in case the phone is not available.

f. Scalability:

The communication system must allow multiple devices to connect to each other. This will let a swarm of KnightCops carry out missions while remaining in contact with each other.

The specifications are listed in Table 2 below:

operational range	100 m
minimum bandwidth	10 Mbps
maximum power consumption	500 mW
encryption	yes

Table 2 – Wireless Communication Specifications

2.4.0 Risks

Anytime a group of people work together for a common goal, there are numerous risks that can derail the goal. Our group is aware of this and has decided to move forward with each of us sharing responsibility to achieve our goal. To avoid the risk of unnecessary confusion and one person being overload, we've

agreed on making regular updates to a project plan to make sure all members are on the same page. Governing the manner by which the development of the project moves along, we believe, will help the project move forward at an acceptable pace and increase our chances of finishing on time.

We are a 3-member group and, of course, our time is limited, so it is imperative to remain on schedule or we could face the risk of running into all sorts of problems. For example, the risk of the project not being completed in time is great if we were to lose a member. If an emergency were to arise that forced a team member's exit, additional work would be placed upon the remaining members. So, the work that was to be done by three would have to be completed by two. Since we each chose to work in areas we felt strongest in, the loss of a group member would force the remaining two to work in areas they weren't as familiar or comfortable in.

Also, we have little room for personal issues that could interfere with a member's ability to complete their share of work. We all realize the need to put aside egos for the good of the group. Infighting that distracts from the ultimate goal cannot be tolerated. Our Senior Design project will span almost seven months. Many things can serve as distractions and risk over that period of time. These sorts of risk must be managed well. For example, if a group member is not meeting stated goals, the other members can ask that member to submit regular updates. Finding out at the last minute that a group member will not be able to meet their obligation is too costly. This can lead to tension and hard feelings. The project's success cannot be sacrificed for one individual. Each member must be accountable to the group.

There is the also a risk that no matter how many times the project has been tested it could malfunction during the presentation. Because of this, we will devise a contingency plan to overcome possible malfunctions. These safeguards would be easy on-the-spot fixes to help us wart off frustration, annoyance, embarrassment or, worst of all, an unprofessional appearance. The pressure of the moment brings its own problems. We would like to avoid things that could inhibit our ability to think clearly and find any problem efficiently. With a checklist each person would know the job they would play in looking over the project if this were to happen.

The number of technical and personal issues that can arise are numerous. Having awareness and a way of dealing with potential risks is important to completing the project. Our group has discussed this. We have completed many things in the past, and we all bring a certain degree of ability of seeing things through to the table.

2.5.0 TEAM MANAGEMENT:

There are many different management approaches that we could use to aid in the coordination and the organization of the team. From the moment the group was formed, we were in understanding that we were all equally responsible for

the outcome of the project. The reason behind this is that the project is going to be given a single grade no matter how works on what. That being said, it was important for us to be as effective and efficient in the utilization of our resources as we could. We did this by trying to utilize our strengths and past experiences as well as we could. In order to do this we decided to make Elean the project manager of the group, based on him having a more extensive experience with robotics and engineering projects in and outside of school alike. It doesn't mean that he would be responsible for the whole project success but would be utilized more to keep an organized collaborative effort and overseeing the overall status of the projects in order to stay on pace to meet the required deadlines. We stressed the important of keeping a healthy channel of communication between all members of the group. Not only in order to make sure deadlines would be met, but to create an environment that encourages collaboration in order to achieve the highest quality product possible. Since we all lived on campus, we heavily relied on virtual means of communication like email correspondence, online chats, and Skype for video conferences in order to achieve that open lane of communication. Furthermore, we scheduled regular weekly meetings where we could discuss what we had worked on and help each other if there was anything that any of us was having problems.

The next step was to figure out what storage method we were going to use and how the version control of our documentation was going to work. We decided to use Google Drive, a service provided by Google that allows us to store and share our files in one centralized location. Moreover, it has the additional benefit of enabling us to collaboratively edit our documents at the same time. We decided to create separate, more manageable in size, documents that we could individually work on and keep information resulting from our research. For instance, we kept a separate document with information of similar products, and another for keeping the parts with the parts numbers that we including in our design. We also kept separate documents with our responsible topics to document, which could be seen by all members to then be merged into our final document.

When it came to the separation of labor, which includes research, documentation and design, we thought it was important that we choose something that plays to our strengths by having prior experience or knowledge of the specific task that needed to be accomplished. Additionally, if none of us had experience with a subject, the person most interested in it would get it. Furthermore, it was important to understand that the team is composed of two computer engineers and one electrical engineering. It would not have made too much sense to give the majority of the software design to the electrical engineer and the electrical drawings or schematics to one of the computer engineers. The hardware block diagram (Figure 1) and software block diagrams (Figure 2) illustrate those principles. In addition it shows the importance of having a modular design, which allowed us to separate the work into modules that we could work on in parallel due to the fact that they have very few links between them.

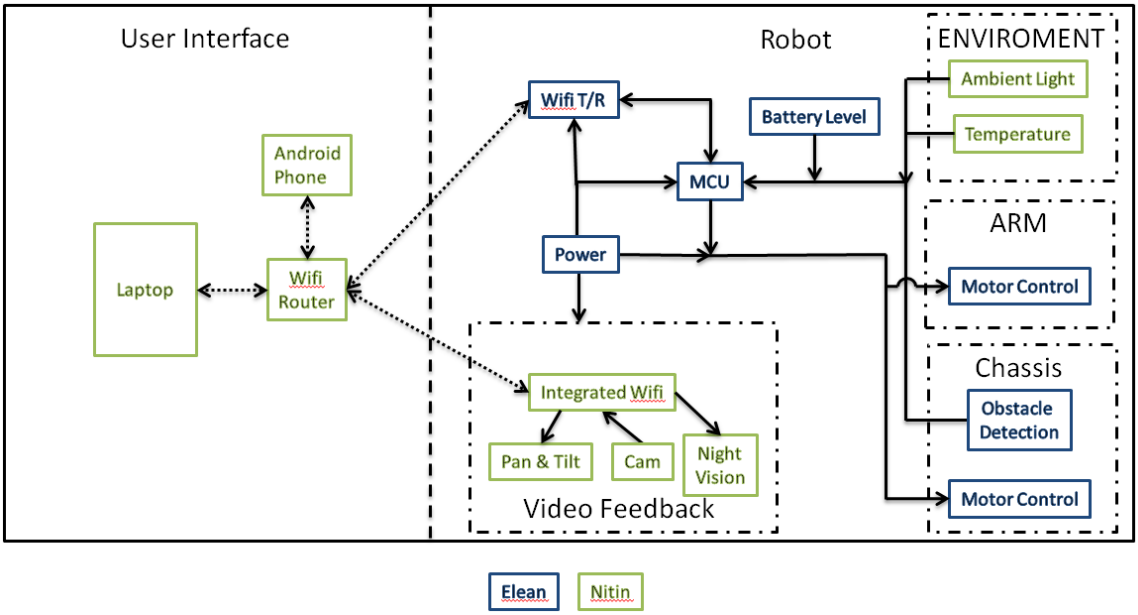


Figure 1 - Hardware Block Diagram

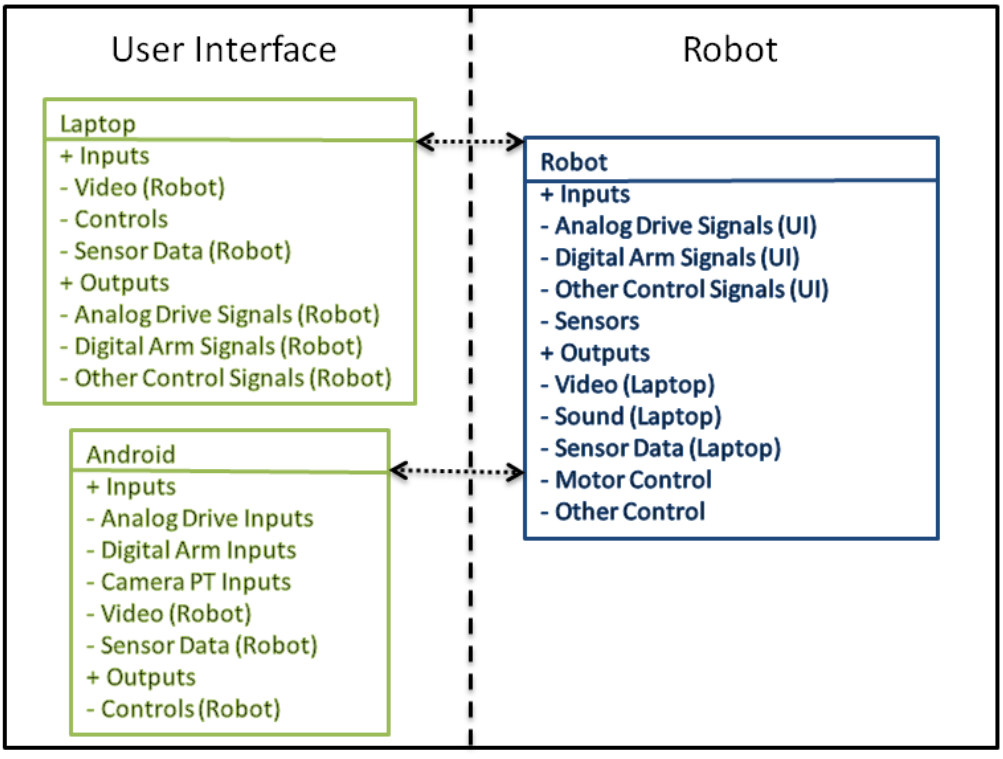


Figure 2 - Software Block Diagram

3.0.0 Research

3.1.0 Existing Similar Projects and Products:

3.1.1 Armored Remote Monitoring and Operated Recon Device A.R.M.O.R.D.

The group designing the A.R.M.O.R.D. robot (Figure 3) had wanted to test the limits of their knowledge in the area of robotics.

The group members, who all had experience within the defense industry, were aware of the need to minimize human casualties and the rapidly expanding use of robotics by the military. With this knowledge they chose to design a robot that would be helpful in carrying out military operations.

They decided to base their robot design on “The Dragon Runner,” which is a robot designed by Carnegie Mellon University (CMU) for military use. A.R.M.O.R.D. would have the capability of being used in combat situations. The group determined they would keep production cost to a fraction ($32,000) of what the C.M.U. robot cost. This would greatly affect what their robot could do. A.R.M.O.R.D was equipped with a video/audio transmitter and a global positioning system (GPS) unit.

Due to the nature of its duties, the robot needed to be able to withstand tremendous force from various events. The frame was constructed from aluminum and supported by a suspension system to help absorb excessive force in the event of an explosion, fall, or collision. There was concern that there may be interference between the motors and the electrical parts, so two different power sources will be used to try to avoid this. It is user-controlled by a remote device which was an IPhone. A laptop or desktop was used as a gateway that receives transmission from the robot systems and sends the data to the user. If needed, the user can use an application on the Iphone to directly control the robot.

Once assembly began the team worked from the inside out assembling first the electrical systems, then the protective exterior of the robot. The remaining steps of programming and syncing all the devices and perform rigorous testing to ensure that the robot correctly complied with all of the objectives were completed to the group’s satisfaction.

Figure 3 - A.R.M.O.R.D.

Reprinted with permission from Andrew Lichenstein

3.1.2 Autonomous Brilliantly Engineered Cooler (A.B.E.C.)

The designers of A.B.E.C. (Figure 4) were happily immersed in all aspects of college life, especially the infamous activity of “tailgating”. They recognized much of the setbacks they faced when trying to carry out a good tailgating party could become the source for inspiration for their senior design project.

Carrying all the needed components needed for the party, the annoyance of having to “unplug” from personal electronic devices for a couple of hours due to a shortage of places to “charge up”, and a desire to do their part to “save the earth” by reducing the use of carbon emitting power sources spurred the team into creating the ultimate tailgating accessory….the Autonomous Brilliantly Engineered Cooler (A.B.E.C.)

A.B.E.C. was designed with features to meet all of these needs and more. The frame work chosen for the A.B.E.C. was a used rideable toy car. The time saved from not having to build one from scratch, along with being able to find a used one made this the practical choice. It had most of the specifications needed and the things it lacked could be modified.

The team considered how the A.B.E.C. would be transported to its destination. The ability to carry large cooler full of ice and have room for food were considered. They determined that it would need to be able to carry at least a hundred pounds and preferably move faster than the 5mph that the motor would currently allow. Because they wanted it to be powered with renewable energy, it was equipped with an electric lawn mower motor and solar panels. This allowed two possible ways to charge it up. Since it would also have a station to connect things such as cell phones and amplified speakers, there would need to be enough power generated to cover these drains on power also. The finished

product would need to be able to remain charged for 3 full hours under expected the conditions without any need for further recharging.

A.B.E.C. had motion sensors to help it avoid collision with anything in its path and be able to reroute itself. A.B.E.C. would operate with tracking systems to be able to navigate by itself to the controller. It would have the ability to track from at least 20 feet to within 10 ft. accuracy. All of the features would be controlled by an attached microcontroller programmed in high level C programming language. The user would be able to control it remotely using an app on a smartphone operating with the Android system.

Figure 4 – A.B.E.C

Reprinted with permission from Marc Bianco

3.1.3 Disaster Zone Emergency Response Vehicle (DZERV)

This group decided on building their project around the idea that it would work for the good of society. With the possibilities to choose from being endless, they decided to design a robot that could assist in search and rescue missions. They believed that by doing this they could minimize human involvement in dangerous situations. This in turn would increase the chances of the overall search and rescue operation of being successful.

The group used various sensors to help communicate back to the user such things as temperature, barometric pressure and humidity. They faced a challenge in figuring out how to design the robot so that it could be controlled by a microcontroller that would sync with various sensors, video equipment, and be able to relay this data in real-time. A radio frequency transmitter and receiver provided the means for communication between the base and the robot. A laptop computer was used as the base. The user could view the area DZERV, seen in figure 5, directly on the laptop's monitor. Control of the robot could be performed by using the laptop's keyboard. This gave the user the ability to both guide the robot where it needs to go and to determine whether or not it was safe for a person to enter into the area.

When up and running DZERV would be able to proceed into danger zones and locate survivors or possible dangers such as downed live electrical wires, gas leaks, or fire. The data would be read on a computer screen, so the user can better access the situation and have a better idea of how to approach. A wireless camera was attached and will not only be able to send visual feedback to the controller, but it will also allow the controller to be able to guide DZERV without actually having DZERV in view. Although this will help guide DZERV, there was an ultrasonic range finder added as a second line of defense in avoiding obstacles. There were safety mechanisms added to keep the DSERV at a low temperature. Alert notifications would let the controller know of any issues that arose and needed to be taken care of.

The team successfully designed and built DSERV. They meet their objectives of designing DZSERV to consume as little or power as possible, while minimizing cost. They determined that DZERV could aide in military, fire, and other rescue operations.

Figure 5 – DSERV

Reprinted with permission from Rob Smith

3.1.4 iRobot Defense

iRobot is an American based advanced technology company that designs both autonomous and remotely operated robots. The products we will be showcasing in this section are part of their defense and first response line of products. They are aimed to aid military and civilian personnel and had unique features and uses that were part of the inspiration for this project.

3.1.4.1 110 FirstLook

The 110 FirstLook (Figure 6)is a recognizance and surveillance robot. It makes its impact by being small, with a footprint of only four inches in height, nine inches in width and ten inches in length. This, in addition to its weight being less than six pounds, makes it really portable. This portability allows it to be carried by one person and be placed in circumstances that would otherwise not be possible with a bigger frame. Moreover, it is very durable, as it can be thrown through

windows, can survive up to sixteen foot drops onto concrete and is not only water resistant but waterproof up to three feet. Its battery allows it to be used for up to six hours of continuous operation, which is very useful when it is important to maintain surveillance for an extended amount of time. It also has great maneuverability by being able to maneuver a large variety of environment and overcome curbs and steps up to seven inches tall. Its only autonomous capability is to self-right itself if it becomes flipped over. All of these features make it an ideal robot for recognizance of hard to access places, on difficult to traverse or unknown terrain.

Figure 6 - 110 FirstLook

Reprinted with permission from iRobot.com

3.1.4.2 SUGV

The SUGV (Figure 7), or Small Unmanned Ground Vehicle, is a portable and versatile robot with a manipulator on it. It keeps a small footprint while the manipulator is in the stowed position coming in at nine inches tall, fourteen inches in width and thirty inches in height. Its weight also increases to twenty nine pounds, so it loses some of the portability compared to the 110 FirstLook but it gains the ability to manipulate objects with a highly dexterity manipulator. It still fits on a backpack so it remains portable enough. It has great mobility and maneuverability by traversing terrain at six miles per hour and is able to go up forty degree slopes. Additionally, it can overcome twelve inch obstacles and climb ten inch stairs. It waterproof ability is double as that from 110 FirstLook with an impressive 6 inches. Its battery power is able to sustain one and a half hours of operation with an optional battery upgrade for up to six hours of continuous operation.

The manipulator has a high amount of dexterity by providing a turret with three hundred and sixty continuous rotation, a shoulder with a hundred and eighty degrees pitch, an elbow with positive and negative a hundred degrees pitch and a wrist with three hundred and sixty continuous rotation. The maximum extension in the most extreme manipulator configuration is twenty four inches. Its lifting capacity is seven pounds while fully extended and fifteen pounds while close in. The standard gripper configuration is a 5 inch parallel jaw opening with thirty five pounds of grip strength. There are additional configurations for the gripper that allow it to better grip smaller objects and wire. There is also special tool kit that among other things cuts wire, removes obstructions and moves threads and debris.

The controller is very portable by being wearable and having a heads up display. The game style hand controller and heads-up glasses come in at a very respectable six pounds. The familiarity of the game style controller makes it very user friendly and cuts down on training time. The range of the wireless communication is one thousand meters, which comes very handy a threat is better kept at a very long distance from the personnel.

Figure 7– SUGV

Reprinted with permission from iRobot.com

3.1.4.3 510 PackBot

The 510 PackBot (Figure 8) is one of iRobot's most battle tested and versatile robots in their inventory. The key to this robot is its modularity and adaptability to a wide range of scenarios. Depending on the configuration it can be used an Explosive Ordnance Disposal, or EOD, robot by identifying and neutralizing improvised explosive devises. In a different configuration, it can be used as a

recognizance and surveillance robot by being able to reach hard to get and hostile environments. In order to achieve this level of adaptability to the needs of missions that take place in ever changing environments, both its chassis and its digital infrastructure had to be developed to accommodate many different types of payloads, tools and sensors.

iRobot's Aware 2 robot intelligence software does just its part. In part by creating a more adaptable and user friendly graphical user interface that allows the user to switch between configurations and automatically switching controls and displays to match them. It also has the ability to create pre-set poses for prompt positioning of the robot. Furthermore, it keeps an active three dimensional model of the robot, showing its configuration, heading, and position status. On top of the user interface, this architecture allows for features like being able toreverse the path to its original location in case of a loss in the communication link between the user and the robot. Autonomous self-righting in case of being flipped over or maintaining a given heading set by the user are never the less very much a part of the overall system. As far as interfacing with different tools and sensors it allows the user to be much more in aware of the robots surroundings and allows them to perform their task with greater precision. For instance, the relative force being applied by a gripper system can be relayed back to the controller and indicated on the graphical user interface.

The chassis 510 PackBot is incredibly portable for being as modular and full of features as it is. Its foot print with no payload comes in with a height of seven inches, width of twenty inches, and twenty seven inches in length. Weighing in at twenty four pounds, keep it in a portable range as well as allowing its top speed of six miles per hour. It is also submersible up to six feet under water and can climb grades of up to sixty degrees. Additionally, it comes fully loaded with sensors like a global positioning system (GPS), compass, accelerometers, inclinometer, and eight payload bays.

Some of the possible additional accessories are explosive detection kits, two way audio communication, thermal and wide angle cameras, radiation and temperature sensors, hooks and flashlights among many others. Perhaps the most important accessories would be its two manipulators. The Manipulator 1.0 is a three link arm with a targeting-head tracking gripper. It has the ability to have multiple pre-set positions, extends up to an amazing seventy four inches, and has a lifting capacity of ten inches fully extended and thirty pounds at close range. Not only does it have an extraordinary reach and lifting capacity but it also has an incredible amount of dexterity with eight independent degrees of freedom. Including, a three hundred and sixty continuous head pan and rotation, as well as a head tilt of two hundred and twenty. The second arm has less but similar capabilities but it weighs in at six pounds instead of twenty.

Figure 8 –510 PackBot

Reprinted with permission from iRobot.com

3.1.4.4 710 Warrior

The 710 Warrior (Figure 9) trades off portability for raw power, ruggedness and strength. With a footprint of thirty inches tall, thirty inches in width, forty one inches in length and weighing in at five hundred pounds, it is definitely not a portable robot. Its raw driving power allows it to overcome its weight to be able to achieve its maximum speed at eight miles per hour. Its increased mobility allows it to traverse more challenging and aggressive terrains. It retains the ability overcome vertical obstacles of up to eighteen inches, climb stairs and slopes with a gradient of forty five degrees. The most amazing part of its mobility is that is able to do all of this while carrying a payload in excess off a hundred and fifty pounds.

It is also very modular by running the same Aware 2 intelligent robot software as the 510 PackBot. Its two-link, heavy-lifting manipulator has the ability to extend up seventy five inches. The more impressive feature of this manipulator, much like this robot itself, is its power. With a lifting capacity of over seventy pounds while fully extended, three hundred pounds at close proximity and a grip strength of seven hundred pounds, it is able to tackle tasks that are simply not possible with a smaller more portable design.

Figure 9 – 710 Warrior

Reprinted with permission from iRobot.com

3.2.0 Microcontrollers:

Undoubtedly the most integral part of KnightCop would be the microcontroller we end up choosing. The microcontroller would have to interface with all motor controllers, sensors and the PC/Android application we throw at it. It must, therefore, have sufficient amounts of memory and input/output pins. Processor speed is not a major concern since we will not use the robot for number crunching.

Most microcontrollers we considered were available as free samples on request, so cost was thankfully not a concern when homing in on our preference.

3.2.1 MSP430G2553

The 16 bit MSP430 line from Texas Instruments is extremely power efficient microcontrollers. The MSP430G2553 in particular is a popular microcontroller. It is touted as a cost efficient alternative to Atmel and PIC microcontrollers on several websites and forums. It had several notable features like:

Power Saving Modes: The microcontroller is fully operational at 2.2 Volts, drawing 230 µA. it could stay in Standby while only drawing 0.5 µA of current or hibernate at a measly 0.1 µA.

Capacitive Touch Sense: it also provides 24 input/output pins - all of which have 'Capacitive Touch'. Capacitive Touch provides the ability to roll out touch applications with the free software library from Texas instruments. It would have meant that we could use a generic touch sensitive LCD instead of an Android phone. This would have come at the expense of a more expensive development

board, negating one of the major advantages the MSP430 had over other microcontrollers.

Ease of development: The microcontroller can be programmed using a serial interface. The development board is very affordable ($4.30 at the time of writing) and Texas Instruments is generous enough to provide their IDE Code Composer Studio free of cost. We were already proficient in developing Assembly and C programs for MSP430 boards thanks to laboratory sessions in the Embedded Systems class we took last semester.

This particular microcontroller also featured an integrated temperature sensor, comparators and USB support. More details are described in table 3 below:

Architecture (bits)	16
Frequency (MHz)	16
Max Operating Voltage (V)	3.6
Program Memory (KB)	16
RAM (KB)	0.5
USART / SPI	1
I2C	1
I/O Pins	24
Analog to Digital Convertors	8 channels 10-bit each

Table 3– MSP430G2253 Specs

3.2.2 ATmega328

Atmel's 8 bit ATmega328 microcontroller is massively popular among robotics hobbyists and enthusiasts. This translates into a robust community which means help is a click away in most cases.

There are a ton of resources to learn from and to be wary of mistakes others have made while building their robots. As with the ATmega2560, development can be facilitated on an Arduino board, but while using Atmel's tool chain for software development. The tool chain in itself propels Atmel's offerings ahead of

the competition. It is highly polished and available for free from Atmel. Other features:

picoPower: Atmel's proprietary power saving technology that enables the megaAVR microcontrollers to be operable at 1.62 V. The user can switch through various available modes using high end software. It also enables the microcontroller to go to 'sleep' and minimizes power consumption.

QTouch: Atmel's equivalent of Capacitive Touch Sense from Texas Instruments. A distinguishing feature is 'proximity mode' where QTouch capacitive sensors go into high sensitivity mode and can detect the user's approaching finger.

SleepWalking: This technology implements intelligent AVR peripherals which allows megaAVR microcontrollers to suspend the CPU into a deep sleep state if it is determined that incoming data does not require CPU use.

The microcontroller is more than sufficient if the robot is only required to move and interface with a couple of sensors. However, our goal of having a fully functional arm requires more I/O pins and USART than what is offered by the ATmega328 (as seen in table 4 below):

Architecture (bits)	8
Frequency (MHz)	20
Max Operating Voltage (V)	5.5
Program Memory (KB)	32
RAM (KB)	2
USART / SPI	1 / 1
I2C	1
I/O Pins	23
Analog to Digital Convertors	8 channels, 10-bit

Table 4 - Atmega328 Specifications

Like the MSP430, ATmega328 also comes with a temperature sensor. Arduino support is a huge plus for Atmels because it translates into a huge selection of peripherals and add-ons. Although design is said to be portable among PICs and AVRs, we shall see below how development is easier using an AVR chip. With

Arduino rolling out new development boards frequently, driver and firmware issues are non-existent on the newer AVRs.

3.2.3 PIC16F886

Microchip's PIC16F886 is an 8 bit microcontroller representing the other major player in the market. Also backed by a thriving community, our research showed that help and guidance would be as abundant as (if not better than) Atmels.

mTouch Sensing: Microchip's touch and input sensing technology. Apart from support for touch screens, proximity detection, sliding and scrolling controls, these also offer GestIC. GestIC enables 3D tracking and gesture sensing to compatible devices, exponentially increasing the possible input stream.

Hardware USB Support: USB-OTG enables 16-bit PIC microcontrollers to interface with USB natively. There are extensive free libraries to aid with software development using these.

Hardware Audio Support: PIC16 family boasts of native audio playback and record using Adaptive Differential Pulse Code Modulation. Speech from a microphone can be recorded using peripheral Analog to Digital converters. Sound files or voice messages may then be stored on the Flash memory within the microchip.

The PIC falls prey to the same shortcomings as aforementioned microcontrollers: low RAM and not enough I/O pins, as evident from table 5 below:

Architecture (bits)	16
Frequency (MHz)	8
Max Operating Voltage (V)	5.5
Program Memory (KB)	14
RAM (KB)	0.359
USART / SPI	1 / 1
I2C	1
I/O Pins	25
Analog to Digital Convertors	11 channels, 10-bit

Table 5 - PIC16F886 Specifications

The Microchip controllers had a distinct advantage over others because they can be easily interfaced with a PC using PICkit which facilitates easy programming and debugging using only 2 I/O pins.

3.2.4 ATmega2560

Atmel's ATmega2560 is very popular for higher functioning robots due to the Arduino Mega 2560 development board. Even at a first glance, the ATmega2560 looked like everything we wanted in a microcontroller. The only downside is the power consumption on this chip. It operates between 4.5 V - 5.5 V at its peak. Although it doesn't have picoPower, our applications demand constant CPU use and we did not feel picoPower was worth giving up extra I/O pins. We do not foresee switching between 'awake' and 'sleep' states often enough to have a signigficant impact on battery life.

ATmega2560 offered us everything we wanted in a convenient package. There are more than sufficient USARTs, I/O pins and ADCs as listed in table 6 below:

Architecture (bits)	8
Frequency (MHz)	16
Max Operating Voltage (V)	5.5
Program Memory (KB)	256
RAM (KB)	8
USART / SPI	4 / 5
I2C	1
I/O Pins	86
Analog to Digital Convertors	16 channels, 10-bit

Table 6 - ATmega2560 Specifications

The ATmega2560 does not come with a temperature sensor. It does, however, have 4 UART and 5 SPI interfaces. This means we are free to choose whatever sensors best fit our needs rather than fret over lack of I/O pins.

The obvious choice for a development board will be the Arduino Mega 2560 Revision 3 board. It is affordable and has excellent documentation and software libraries available.

3.2.5 PIC18F47J53

This offering from Microchip is another highly popular microcontroller in robotics societies. The PIC18F47J53had some distinct advantages over the ATmega2560:

nanoWatt XLP: This technology allows the microcontroller to go into deep sleep, draining only 9 nA of current, thereby maximizing battery life.

USB 2.0: Integrated full-speed USB 2.0 means we could allow users to expand (or even upgrade) the robot themselves. They could, for example, connect a night vision camera for the video feed instead of relying on the camera used by us.

The PIC18F47J53 offered just enough interfaces/peripherals and I/O pins as can be seen in table 7 below. It was a tough choice between this and ATmega2560.

Architecture (bits)	8
Frequency (MHz)	8
Max Operating Voltage (V)	3.6
Program Memory (KB)	128
RAM (KB)	3.71
USART / SPI	2 / 2
I2C	2
I/O Pins	44
Analog to Digital Convertors	13 channels, 12-bit

Table 7 - PIC18F47J53 Specifications

Both the PIC18F47J53 and the ATmega2560 had even footing on the hardware side. Both microcontrollers could be obtained as free samples. Both have been used in robotics for quite some time and extensive resources were available for both online. Our only reservation was that we would end up using two microcontrollers due to a shortage of interfaces on this PIC.

On the software side, the PIC family was found to have a tradition of lackluster compilers. In our research we saw that some of the past robotics projects which used PIC microcontrollers also spent hundreds of dollars on compilers and

debuggers. This is because the free options do not measure up to the paid offerings. Microchip does provide a free IDE (MPLAB X) to program their microcontrollers. However, the free version of MPLAB only provides level 2 optimizations during compile. This is in stark contrast to the software toolchains available for AVR microcontrollers which provide full optimization and are all available for free as well. In summation, PIC has a time-honored but restrictive software development environment, while AVRs offer variety and conform to contemporary IDEs like Eclipse and Visual Studio.

3.2.6 Deciding on a Microcontroller

As mentioned before, the CPU speed ranges from 5 MIPS (on the PIC16F886) to 16 MIPS on the ATmega2560. Although a substantial difference, it was not paramount to us since data processing would be handled by the Android device and the PC. The operating voltage was also comparable, besides the MSP430 beating all others at peak operating voltage of 3.6 V. Battery life, although important, was not enough to sacrifice the RAM and program memory the other chips offered.

It was a close decision between the PIC18F47J53 and ATmega2560. On the one hand, the ATmega offered us all the IO pins and RAM we needed, plus excellent development support with the Arduino environment for both Android and PC. On the other hand, it looked like an overkill compared to the PIC, which also had the advantage of USB to serial interface and native USB 2.0 support. The final hardware comparison is shown in table 8 below (bold specs are more desirable):

Microcontroller	**ATmega2560**	**PIC18F47J53**
Frequency (MHz)	**16**	8
Max Operating Voltage	5.5	**3.6**
Program Memory (KB)	**256**	128
Program Memory (KB)	**256**	128
RAM (KB)	**8**	3.71
USART / SPI	**4 / 5**	2 / 2
I2C	1	**2**
I/O Pins	**86**	44

Table 8 - Final Hardware Comparison

As evident from table 8 above, the ATmega2560 is a better choice for KnightCop, simply because it offers us more options and scalability. We wish to keep the design simple and effective and not have to use multiple processors when one would do the job.

As a team we decided to go with the ATmega2560 because of the toolkit and peripherals it offers.

3.3.0 Power Unit:

3.3.1 Power Supply

Early into our research, we realized that KnightCop could be powered using a countless amount of configurations. For example, we had the option of having a single power source power all of our subcomponents. We could also go as far as having a separate power source for each subcomponent. While these two examples represent extreme design, a practical design would require a great deal of consideration and planning. Some of our initial considerations included the voltage and current requirements of each subsystem so as to avoid brownouts, minimizing the amount of power-related noise entering the circuit, and the length of time KnightCop could operate before the battery completely drains. We decided to focus on learning about the types of batteries we had to choose from. From there, we would look at ways we could design our power supply. We believed this would give us enough information to design our power supply.

3.3.2 Battery

We did not consider powering Knightcop using alternating current because it would simply impossible to meet our design specification that way. A battery would be the way to go. There was a shortage of batteries to choose from. We investigated lead, lithium, nickel cadmium, alkaline, and nickel metal hydride batteries. Our desires for a battery that had a sufficient amount of spare capacity and be commonly available were two of our immediate concerns. With that, we looked at the major types of batteries.

Lead Battery

We learned there are two types of these batteries: starting and deep cycle. Starting batteries are designed to deliver a quick burst of energy for things such as the starting of a car's engine. A deep-cycle battery is designed to deliver energy over a long period of time. Of these two major types of lead batteries, each comes in three different cell structure types: gel, wet and an absorbed glass mat.

Absorbed Glass Mat

These types of batteries have a significantly longer storage capacity with respect to wet cell batteries.

An important measure, known as cold cranking amps, of lead batteries is the number of amps it's able to deliver at a 0°F without dropping below 7.2V. Cranking amps is a measure of how many amps a battery is able to deliver at 32°F without dropping below 7.2V. Reserve amps is a measure of how many minutes a fully charged battery at 80°F can discharge 25 amps before dropping below 10.5V.

Deep-cycle Batteries

Amp hour (Ah) is an important rating of deep-cycle batteries. It can be used to find both the energy capacity and the maximum current discharge rate of the battery. So because we already know the battery's voltage, knowing the amp hours makes it easy to determine the battery's energy capacity. C is a symbol associated with a battery's maximum discharge rate. C-rate describes the battery's maximum discharge rate with respect to its maximum capacity for some given length of time. From this discharge rate, the maximum current of the battery can be found. The maximum current is the product of the amp and C. An E-rate describes the amount of power a battery can deliver with respect to its maximum charge capacity.

The shortest amount of time a battery can be safely discharged in is an important measure. This discharge rate can be given in terms of current (C-rate) or power (E-rate). Knowing the C-rate as well as the amp hours of a battery, the maximum discharge current can be found. Similarly, the maximum amount of power that can be drawn from the battery can be calculated knowing its E-rate, voltage and amp hours.

The battery's capacity is a measure of how long the battery will operate at the specified voltage for a specified discharge rate.

Memory Effect

Recharging a battery before it is fully discharged can be a problem for some types of batteries. Subsequent recharges will not be able to store as much energy as previous charges. This is known as the memory effect. To avoid this problem, these types of batteries should be fully discharged before attempting to recharge it. This is certainly an undesirable property. It poses an inconvenience and can prove to be a costly mistake to commit.

Ratings

There are several ratings that are used to help compare different batteries. Some of the ratings used are the capacity, discharge rates, internal resistance, and nominal voltage.

The capacity of a battery refers to amount of energy that a particular battery can deliver over a given period of time. Capital letter **C** is used to represent capacity. Capacity is measured in amp-hours. The capacity of a battery is typically determined by discharging a battery over a 20 hour period. A battery with a C

rating of 4 suggests that the battery can reliably deliver 4 amps of current for one hour. This rating can also be used to give a very loose estimate of the battery's ability to deliver higher or lower current draws over different periods of time. We might expect the same battery rated at 4C to deliver 2 amps over a two-hour duration or 8 amps for a half hour. For us, our specification require KnightCop to operate over an extended period of time. We will need a battery with a C rating of at least 5.

We also need to be concerned with the amount of voltage the battery can reliably maintain over a certain period of time. This brings us to the battery's nominal voltage. Since a battery cannot maintain an exact voltage under varying load conditions, it is important to know how much its value can fluctuate and when it drops below a certain value. A battery with its voltage below a certain value is considered dead. If a component requires 3.5V to operate correctly, voltages below that may result in malfunctioning of the component. Electronics are particularly vulnerable to voltage levels. Voltage below the required level could mean that KnightCop can no longer move or that the microcontroller processes data incorrectly. We must have a battery that can maintain its voltage for the length of time required by out specification.

The discharge rate

The internal resistance of a battery is an important factor because it affects the battery's ability to deliver current. Given all other things the same between two batteries, the battery with a high internal resistance will not deliver the same amount of current as the one with a lower internal resistance. This is an important consideration for us because very often the battery will be asked to deliver larger-than-normal amounts of current. For example, when KnightCop is called upon to begin some motion, more current is required to start the motion than is required to continue the motion. A battery unable to deliver higher burst of current for short periods will not meet our needs.

Lithium ion

Lithium polymer can produce a higher current output as compared to lithium ion batteries. Lithium ion batteries have been used in projects similar to Knightcop with great success. They are light weight and possess a high energy capacity. These batteries do not suffer from memory effect, so a full discharge does not have to be performed before attempting a recharge.

Nickel Cadmium

These batteries have the highest current output of all the batteries. They are also the most affordable. Rechargeable nickel cadmium batteries can be recharged several thousand times over its lifetime. No other type of rechargeable battery can be recharged as much. Newer nickel cadmium batteries do not suffer from memory affect as much as earlier ones did. Unfortunately, nickel-cadmium

batteries release toxic materials when left to decompose. This makes them environmentally unfriendly and to be avoided if possible.

Alkaline

An alkaline battery is a dry cell battery. These reliable batteries to have a high energy density, long shelf life, and low discharge rates under a wide range of temperatures. These types of batteries are most often disposable. Once it is drained, it is discarded; however, there are rechargeable alkaline batteries. These types require a special recharger that can be much costlier than other types of rechargers.

Nickel-Metal Hydride

Nickel-Metal Hydride batteries suffer from even less memory effect than nickel-cadmium batteries; however, these batteries are heavier than nickel-cadmium batteries. They have lower internal resistance, which allows them to deliver relatively high amounts of current. This makes them well suited for high drain devices. Otherwise, they discharge slowly, allowing the battery to maintain its voltage until the end of the cycle. They can be recharged hundreds of times. Unfortunately they have a very long recharge time, up to 10 hours. Their self-discharge rate is also very high. This means that during storage, they'll lose much of their charge within two to three weeks.

Battery Charging

Great care should be considered when choosing how to recharge a battery. Battery chargers are not to be used interchangeably. Using a Nickel Cadmium charger to recharge a nickel metal hydride battery can damage the battery. Also, the manner in which the battery is recharged is important. Smart Chargers recharge batteries in a series of three steps: bulk, absorption, and float. During the bulk stage, 80% of the battery is charged. During the absorption stage, the battery's voltage is held constant while the current decreases until a 98% charge is achieved. The voltage and current are again lowered once the float stage begins. During the float stage, the battery is charged in a way that avoids it from overheating or its liquid content from boiling while maintaining a 100% charge.

3.3.3 Voltage Regulation

KnightCop has multiple subcomponents. Each of the subcomponents have its own voltage requirement. To ensure that we meet each of these subcomponent's correct voltage requirement, we need to come up with a way to accomplish this. It was decided that by using voltage regulators we could make sure the subsystems would remain within their voltage range. We considered several designs: voltage-divider circuit, diodes, Zener diode, linear voltage regulation, switching voltage regulation

Diodes can be used to effectively regulate voltage. Several diodes wired in series can be used to achieve a desired voltage. Although this is possible, in fact

using a diode(s) to step down the voltage can be a bit tricky. Care must be taken to not exceed the diode's current limitation. The types of diodes that would be used are relatively inexpensive.

A Voltage-divider circuit can also be used to step down the voltage.

Zener diodesare helpful when trying to ensure that the voltage doesn't exceed a certain value. Voltages above a certain value, the breakdown voltage, stop the flow of current through the diode. A resistor placed in series with the diode (Figure 10) helps limit the current through the diode. These diodes are also relatively inexpensive.

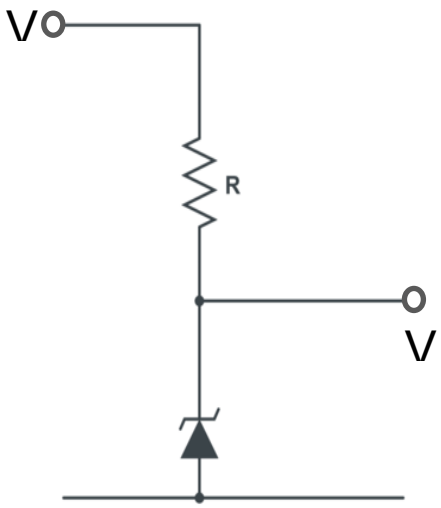


Figure 10 – Zener Diode with Resistor in Series

Another way to regulate voltage is by using a linear voltage regulator. They configure relatively easy and provide a reliable way to maintain a minimum voltage. Linear voltage regulators, however, are not very efficient. This is mainly because linear regulator are essentially always on, regardless the input voltage's value. As these regulators step down the voltage, heat is constantly being released. The greater the voltage drop, the greater the amount of heat generated. A heat sink is recommended when using these types of regulators. A voltage regulator could meet our need.

The most efficient way of voltage regulation is obtained by using a switching regulator (Figure 11). These regulators differ from linear regulators because they are not constantly performing voltage drops. The switch goes on and off at a fixed rate. This reduced the amount of heat generated. The result is that switching regulators are much more efficient than linear regulators. Considering that KnightCop is battery driven, special consideration will go to devices that draw less power.

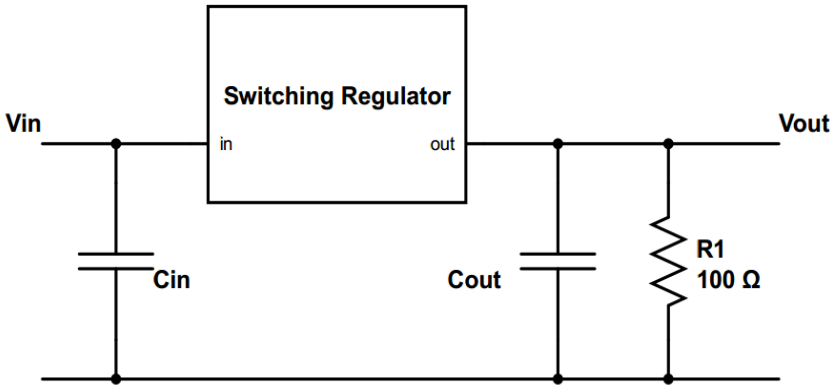


Figure 11 - Switching Regulator

3.4.0 Communication Systems:

The communication system of choice seemed to be Wi-Fi from the beginning, but we decided to look into other options to be prudent.

3.4.1 Bluetooth

Bluetooth operates on the 2.4 GHz spectrum at the IEEE 802.15.1 standard. Bluetooth has an edge over Wi-Fi due to its lower power consumption and affordability. Although the bandwidth is low, we would not be transmitting huge amounts of data. Bluetooth, like Wi-Fi, was available on the laptop and Android phone we were planning to use. Table 9 shows the different Bluetooth transceivers we looked at:

Device	**Roving Networks RN-42**	**Bluegiga WT12i**
Bluetooth Class	2.1	2.1
Max Over Air Data Rate (Mbps)	2	3
Max Supply Voltage (V)	3.3	3.4
Interface	UART/USB	UART/USB
Antenna	PCB Trace	Chip
Price (USD)	19.95	29.91

Table 9 - Comparison of Bluetooth Transceivers

The only drawback is the range - most Bluetooth transceivers peak at a range of 10 -15 meters. The data transfer rate fell drastically. ZigBee protocol claims to have similar data transfer rates and greater range, so we decided to go for XBee modules instead.

Class 1 Bluetooth devices which boast of ranges approaching 100 meters are prohibitively expensive and draw considerable power. Bluetooth also does not scale nearly as well as Wi-Fi, in case we decide to increase the number of sensors on the robot. It will also fall short on bandwidth if image/video processing was done in situ.

3.4.2 Radio Frequency

Radio Frequency solutions for wireless communication took us by surprise. They were very affordable and offered a huge range of operation. On more research, however, we discovered that implementing RF transceivers was an ordeal in itself. The frequencies used (315 MHz and 434 MHz) are notoriously prone to signal noise. We definitely did not have spare time to tinker with filters and sweating over interference from GPS or amateur radio broadcasts. We looked at a very popular RF transceiver during our research, the **RFM12B-S2 Wireless Transceiver.** While it was very low cost compared to other options ($6.95 per transceiver) and supported SPI, it was only capable of transmitting 115.2 Kbps. We found this severely restricting considering the number of peripherals that needed to be in constant communication with the PC Hub.

RF was also discarded as an alternative.

3.4.3 Infrared

Infrared has some crippling limitations that were enough for us to turn our back on it. The protocol works on Line-of-Sight which means any obstacle will break communication with the robot. This is unacceptable for the kind of missions we have in mind. We thought about programming some autonomy within the robot so it could function without signal from control, but another limitation of Infrared devices is that unlike newer Bluetooth and Wi-Fi implementations, Infrared devices only work in pairs. One receiver can only acknowledge one transmitter. This limits our options on scaling and is not practical for real world implementation where obstacles are bound to interrupt communication between KnightCop and the controller. Infrared too was rejected as a viable option.

3.4.4 Wi-Fi

Wi-Fi was our original choice for interfacing the PC to the Android phone among all the wireless protocols. It offers excellent bandwidth (11 to 54 Mbps) and range (100 meters typically). It also fits our criteria of scalability - if we intend to add more sensors or functionality to the robot we do not want to be limited by the bandwidth available to us. Wi-Fi range can also be extended out in the field by using access points as intermediaries. In case connection with a robot is

compromised (if it stumbles or is knocked over the com range) another robot out in the field can help it re-establish communication. Wi-Fi also operates on the 2.4 GHz spectrum, but consumes more power than Bluetooth. Considering how well modern Lithium Polymer batteries perform, we were confident that we would meet our target battery life for the robot even with Wi-Fi.

Table 10 below shows how different Wi-Fi modules we found in our research compare to each other:

Manufacturer	**Roving Networks**	**Texas Instruments**	**Bluegiga Technologies**
Model	RN171XVW-I/RM	CC3000MOD	WF111-A
Protocol (802.11)	b/g	b/g	b/g/n
Frequency (GHz)	2.4	2.4	2.4
Transmit Power (dBm)	12	18	17
Data Rate (Mbps)	11, 54	54	11, 54, 72.2
Interface	UART	SPI	CSPI, UART, USB
Antenna Type	Wire	U.FL	Integrated
Security	WEP, WPA, WPA2	WEP, WPA, WPA2	WPA / WPA2, WEP, CCMP, TKIP, WPS
Price (USD)	37.48	23.56	28.16

Table 10- Comparison of Wi-Fi Modules

In the end, we chose Roving Networks' RN171XVW-I/RM module to facilitate Wi-Fi on KnightCop.

It was a through-hole part, allowing for easier prototyping compared to surface mount which would have required soldering. Moreover, the extraneous interfaces available on other modules would be wasted since we have sufficient UARTs on the ATmega2560.

Both the Android device (HTC One V) and the laptop we intend to use as a PC Hub (Lenovo Y500) had integrated Wi-Fi solutions. We also chose to go for a Wi-Fi capable IP surveillance camera which could broadcast over HTTP or connect to the devices in ad-hoc mode.

3.4.5 ZigBee

ZigBee is yet another wireless protocol working over the same frequency as Wi-Fi and with comparable range, but lower throughput. It's lower power consumption and ease of implementation (XBee modules are the de facto standard for wireless robots) made it the obvious choice for communication between KnightCop and the PC. The following table 11 shows a comparison between the various XBee models we considered:

Device	Series 1	Series 2	Pro
Range (feet)	300	400	5280
Power Consumption (mW)	165	132	973.5
Frequency (GHz)	2.4	2.4	2.4
Data Rate (Kbps)	250	250	250
Antenna	Wire/Chip/Trace	Wire/PCB/External	Wire/External
Cost (USD)	22.95	25.95	42.95

Table 11- Comparison between XBee Modules

As can be seen, data rate remains constant over all different variations of XBee modules. The Pro version of XBee modules was excessive for our needs. The power consumption was too high and data transfer rates dropped over increasing range. These were almost twice as expensive as the other kinds and required an external antenna (sold separately) for optimal performance.

The difference between Series 1 and Series 2 is that the latter offers better performance (greater range and lower power consumption) at the cost of more complicated setup and pairing. For a simple two device pairing setup like ours (point to point communication), Series 1 is sufficient. However, the cost of buying two Series 1 XBee modules, plus an additional XBee Explorer module (to connect one of the modules to our PC Hub) was too high compared to a simple Wi-Fi module found in research.

3.5.0 Mobile Base

The mobile base of the robot is constituted by the drive train or platform, drive motors, drive motor controllers, and the sensors used for detecting the surrounding environment. In essence, it is the base that will not only carry our electronics and our manipulator but also what allows it to move. It has been clear to us from the beginning that one of the most important factors for the success of

KnightCop is its mobility. It would not matter if have great software, advance sensors and the most sophisticated manipulator in the market, if our base can't traverse different kinds of terrains effectively and aid the user in avoiding obstacles and other hazards that effectively. In order to do this, all aspects of the mobile base most work together in harmony.

3.5.1 Platform

There are a lot of requirements and factors that have to be taken into consideration when deciding on the type of platform that has to be selected for this project.

First the size of the platform was an important factor to take into consideration. If the platform's physical dimensions are too large, it would hinder its capability to go through door openings or go under tables which would severely limit its ability to be an effective scout and surveillance robot. Additionally, being too large has the potential of limiting its portability. For instance, it might not be able to fit completely in the trunk or back seat of a car but instead a pickup truck would be needed for transportation. This would create an unnecessary requirement for the user which would make KnightCop less appealing. On the other hand, its physical dimensions can't be too small. It needs to have a big enough area to mount all of our components. Our manipulator, probably being out largest component, would need to have a flat area on top of the platform in which it would be installed securely and be stable. Furthermore, it would need to be able to house all of the electronic components and the video feedback system. Moreover, it would be beneficial and a great advantage to have the space for tactical payload like sensors, radio for two way communication, provisions, important documentation, and more. That being said, the platform's physical dimension is not the only thing that affect it in all those areas. It is also very important to take into consideration the weight that it would be able to carry. It needs to be able to withstand the weight of all of the aforementioned components and also its own. For this reason the weight of the batteries used, which will be described in one of the sections to follow, had to be carefully monitored.

We also had to take into consideration if we were going to be using wheels or caterpillar treads. Whichever one we pick it needed to be able to traverse a variety of terrains like grass, concrete, wood flooring and other types of flooring among others. It also has to be capable to handle driving over small obstacles like rocks and sticks. Caterpillar treads have a number of advantages. They are able to handle uneven surfaces, small objects, rough and soft terrains due to the continuous band of treads distributing the weight of the robot more evenly. They also provide a greater amount of traction due to their greater area of contact with the ground. In addition, depending on the type of track used, they are usually stronger and more resilient to being punctured. On the other hand, the additional amount of traction gained by the tracks is lost in top speed and it creates a greater amount of strain that it puts on the drive transition, shortening its life span. In addition, the far greater mechanical complexity creates its own number

of disadvantages. Assembly and installation of the tracks is a labor intensive and complicated procedure. This also increases the amount of maintenance required on the number of parts required for assembly and makes that maintenance more problematic. The last but not least important factor that we took into account when making the decision between was cost which is definitely a clear advantage for a system with wheels. Taking everything into account we decided that the best course of action was to have wheels in our base. The biggest factors that contributed to this decision were the cost, simplicity and flexibility with respect of choices that we get with a wheeled system.

Having a good idea of what kind of platform we wanted to get for KnightCop, we had to make the decision of whether it was better to build and design a new platform or to simply purchase one. There were four major aspects that we took into consideration when making this decision, flexibility, complexity, cost, and labor or man hours. First we looked at which path would give us the greater amount of flexibility and expandability. Having the ability to be able to expand on our design by either upgrading our current subsystems or by adding new subsystems was of high importance to us. One of the advantages of choosing a robot for our senior design was that they are incredibly scalable and we wanted to take full use of this advantage. The second factor that we took into account was the complexity of the system. We had to take into consideration that the team is comprised of electrical and computer engineers and thus making us lean towards a system that won't require a huge amount of mechanical engineering involved. A built system might be simple enough that would not require a lot of mechanical changes to it but this depends on the availability of these systems as others might be more complex than building our own. The biggest complexity for a built system would be the understanding of the interactions between the mechanical forces involved and the many different types of designs and materials that could be used. The third factor that we took into account was the cost. We sought to make the price of the robot as low as we could for several reasons and the platform could be one of the most expensive components. One reason to keep the price low on the platform was to make the project more affordable for all of the team members and to allow us to spend more money on other subsystems like the manipulator and sensor systems. The final major factor that we took into consideration was the amount of labor that it would take for us to make the platform work with our specific design. It would be beneficial for us to get a platform that comes with everything we need and with all of the specifications required to make our design work, without doing any modifications. We will be looking at some of the products that we could find in the market and come to a conclusion based on which path will give us the best balance among all four of the deciding factors.

The first idea we got was to buy a radio controlled toy truck. We thought this would give us the best price possible, especially if we could find it used. To get a general idea of what we would like we went into different only toy stores and browsed their RC section. We found out that in order to get a truck with dimensions and specifications that we would need in order to fit all of our

components, we had to search in hobby stores, both online and in person. The following is an analysis of the specifications of a sample radio control toy truck that we found to be a good exemplification of the rest of the radio controlled toy trucks we looked at. In addition, we will be analyzing how this product compares to the rest of the options available by comparing how the deciding factors were balanced.

The Super Speed 1:10 Electric RTR RC Monster Truck is a hobbyist model monster truck that would generally be used for racing and off-roading entertainment. The radio controlled trucked turned out to be the most inexpensive option as the RC truck came with a cost of $34.95. This was a great advantage as it would allow us to spend the money on other components. The size was on the small realm. Having a physical footprint of sixteen inches in length, twelve inches in width and eight inches in height, would make it pretty difficult to be able to fit the electronics and especially the manipulator as it is our biggest subsystem. The system on itself is not really complicated but the modifications that would be needed to fit everything make it a more daunting task. This also reduces the flexibility and scalability possibilities as adding the current subsystems would be difficult enough and putting in extra components would just compound to the problem. Its weight is low but it might be so low that adding heavier components on top would upset its balance and make it too heavy. This would create a susceptibility to tip over when a collision occurs, turning to fast or by just simply stopping too abruptly. The motors used to drive the truck are effective at giving it enough strength to traverse different types of terrain but this is because of the lightweight materials that are used in its construction. The motors would not be as efficient or effective if the weight balance is upset and would not have enough torque to allow our platform to drive over softer terrains with a heavier payload. The wheels have a good off-road design but they fall short when it comes to their quality as the material used is a rubberized plastic that could be easily punctured and would not have the strength to hold larger loads as it would be required by our other components and requirements. Which brings us to our next point, the amount of effort and time spend to make all of the modifications necessary in order to make this work. Due to the fact that the radio controlled trucks are not design to be modified to their core, it would take a large amount of effort just to make the modifications necessary to be able to install our components to it. Not to mention the time that it would take to change out the motors, as they are connected to the drive transmission and if that gets damaged then we would be back at square zero. We would also have to spend time trying to find parts like wheels and specific size and shaft length motors in order to make it work with the existing platform.

Making an effort in order to keep the mechanical design aspects as much out of our hands as possible we looked into other possibilities of systems that we could buy that might fit our needs better than the radio controlled truck. We found that there is a community of sites that sell robotic platforms for this exact purpose. We provided to search these sites for products that would be specific to our needs as to keep the amount of labor in making modifications down as possible but also

trying to keep the price as close to our budget as we could. The following is an analysis of the specifications of a sample platform that we found to be a good exemplification of the rest of the robotic platforms that we looked at. In addition, we will be analyzing how this product compares to the rest of the options available by comparing how the deciding factors were balanced.

The standard all terrain mobile robot (Figure 12) from SuperDroid, was a platform that we thought suited our needs really well. It comes in two main configurations, a four wheel drive and a six wheel drive; we considered and analyzed both options. The six-wheel drive has the advantage of more points of contacts with the ground that are powered but it is also heavier and more complex as more mechanical parts are needed. The four-wheel drive would be simpler and lighter and also significantly cheaper. Now that we have made a decision on which configuration to further analyze we proceeded to try and understand how each of our deciding factors fared against the system. The physical footprint of the all-terrain platform is great by being seventeen inches wide, twenty inches long, and six inches tall. It has enough flat areas for the installation of all of our components including the manipulator and the camera. Its low center of gravity and weight of twenty six pounds means that it will remain stable and not top heavy even with the addition of our components on top of it. Its portability is not seriously compromised compared to the smaller radio controlled truck as it still fits on the back seat or trunk of a sedan and can be carried comfortably by one or two people. The fact this platform is created and designed for the very purpose that we are looking for makes it a blank canvas on which it only occupies the space it needs and allows for flexibility and expandability on it. The wheels are specifically design to withstand the punishment that most hazardous terrains can throw at it. Which combined with its size gives it a good amount of the mobility that is essential for the user of KnightCop to accomplish their tasks. This would defiantly be the fewer amount of energy and labor that we would spend on any of the platforms as the modifications are restricted to only the things needed for installing or components. The major problem with this system would most certainly have to be its price tag, at eight hundred dollars it is almost three times the amount we had budgeted for it and might be enough to deter us from it depending on what our other options might be.

Figure 12 – SuperDroid's All-Terrain Platforms
Reprinted with permission from superdroidrobots.com

The final idea that we took into consideration was to build our own platform. There are a number of advantages and disadvantages to building our own platform instead of buying one. The biggest disadvantages of building our own platform would be the complexity of the mechanical design aspect and the amount of time it would take to assemble it. Thanks to one of our members experience with robotics we think that the complexity aspect of it can be reduced by researching the different modules needed in order to have a complete design. On the other hand, there are many advantages in building our own. First and foremost it would give us the ability to build it to our specifications in order to meet all of our requirements. This would even be less complex than with some systems that we could by as we don't have to figure out how to make changes to an existing system that was design for a different purpose in order to meet our needs. Additionally it would reduce in the amount of time that it will take for assembly as all the parts will be built to fit each other a not as a modification. The flexibility that a self-built system would give us is the greater out of any system. We will purposely design the platform in such a way that additional modules and update to existing module will be an easy task. Another big advantage over systems that fit our requirements is the cost as we would only be paying for the price of the materials used. Some of these materials could be provided by business partners and some of the current member's connections and would be described more in detail in the budget and finance sections. These partners would also be a great resource in giving tips and guidance on the specifics of the materials and mechanical design details that are harder to research as they are learned through experience.

The first decision in building a platform of our own was what materials we were going to use and how we were going to put this material together. There are many different types of materials that could be used. We decided to evaluate this materials based on their strength to weight ratio, ease of use, and cost. We needed something that was strong enough to support the weight of all of our components and to withstand some impacts that might be experienced by KnightCops during its missions. Most compound plastics and materials,

especially ones in our price range, would not be able to withstand these kinds of impacts. That left us with the possibility of using either wood or different kind of metals in our construction with the exception of wheels, which we would be discussing shortly, and other miscellaneous components that won't be under much strain. Metals have the ability to have more textile strength in a denser and lighter package than wood would have. It comes with the disadvantage of requiring specialized equipment to cut and mold the metal to the shape desired. This disadvantage is not much of a factor as we have access to this kind of equipment. The cost is comparable if we compare wood with aluminum angle, we looked into steel but the price too steep especially when aluminum and wood would work just fine with our application. Another resource that we have available and would be very useful in the manufacturing of our parts is a 3-D printer. This tool allows us to use our experience with 3-D modeling software and create the parts exactly to the specifications we need for our design. This is a huge advantage as we could create parts to help with the installation of our sensors and components at practically no cost. That leads us to our next point of how we would assemble things together. We would use the aluminum as our main structural material and for brackets to support things like our battery and other heavy components like our manipulator. These structure elements would be bolted together as a substitute of welded together as none of the members have experience welding nor have access to the necessary equipment to weld. The wood and plastic printed part would be used for smaller components due to its cost and it them not carrying a significant weight load. These parts will be screwed, bolted, glued or fastened by Velcro depending on its use.

The second decision that we had to make in building our own platform was which type of transmission we were going to be using. This area has the potential to be very complex depending on the type of transmission of the motor's shaft rotational output to the wheels. For this reason we decided to look for the more simplistic approaches to get this accomplished. The simplest way would be to connect the motors directly to the wheels utilizing the right hub and couplers. This has the limitation that the output power and torque of the motor cannot be modify to suit our needs. This means that we would be limited in our motor selection to just those that has the correct torque output. The way to solve this but still keeping it simple would be to utilize a single speed gearbox in between the motor output and the wheel. Depending on the gear ratio used, the output torque of the motor is either reduced or increased. This could also be represented by the difference between the difference of revolutions per minute between the motor output shaft and the gearbox output. Ignoring frictional losses in the system, the torque and the revolution rate are inversely proportional. In other words if we want to increase the torque by a factor of two then the revolutions per second will be decreased by a factor of two as well, this would be a two-to-one gear ratio. This would increase the complexity of the transmission system but it is a necessary step as we want to gain torque to overcome hazardous and difficult to traverse environments. We also would have to make the decision of whether to utilize one motor per wheel or one per side. This would be determined by the type of motors and gearboxes utilized. If we would utilize

one motor per side we would have to make the decision of whether to chain the front and back wheels together or to only power the back or the front. Powering only the back or the front of the robot would mean that the opposite site would have to either steer, which adds too much mechanical complexity to our system, or its wheels would have to be omnidirectional to allow it to turn. This would increase the price of the system and reduced maneuverability and powered points of contact with the ground which are necessary to traverse uneven environments. Therefore if we decide to not power every wheel independently, they wheel all be powered by coupling them together with chains or belts.

The final major component that we had to make a decision on building our own platform was the type of wheels that would be used. The most important thing to consider when choosing the types of wheels to be used is the types of terrains that could be encountered by KnightCop in its missions. These terrains include but are not limited to grass, dirt, and concrete. The size of the wheels has to big enough in order to traverse through uneven surfaces and overcome small obstacles like rocks and sticks. They would also need to have threads on it so that it could have a better coefficient of friction with the ground, which allows for a better transition of torque. There are many sites available that provide wheels of this nature therefore it won't really be too much of a challenge.

In Conclusion, after analyzing all of our possibilities as throughout as we could, the decision was made to build our own platform. The affordability and flexibility that it allows us to have was greater than its disadvantages. In this way we can also be assured that all of our needs and specifications will be made as we would be designing it. Lastly, the experience gained in the process of designing and building our own platforms is one that we did not gain through school and we see as an opportunity to grow our careers.

3.5.2 Motors

The motors to be used by the mobile based are one of the most important components when it comes to mobility and power consumption. We will be covering power consumption on a different section. Without the use of the proper motors the robot won't be able to move effectively and thus not able to complete its missions.

The First decision that we need to make was to utilize either an alternating current (AC) motor or a direct current (DC) motor. We had three deciding factors in deciding which type of motor to use; method of control, price, and output.

The main difference between the two types of motors, AC and DC, is in the way they are controlled. AC motors, as it name may suggest, requires an alternating current to function. DC motors on the other hand, only require DC current. This is much more easily achieved as batteries output DC current. A DC to AC inverter would be needed in order to transform that DC current to AC, which would have to be corrected in amplitude and frequency in order to run the AC motor. Furthermore, all which is needed for controlling a DC motor is a motor controller

that will control the amplitude of the input voltage into the motor. The more the voltage is supplied to the DC motor, up to its rated maximum input voltage, the higher the output power. The way to achieve this result with an AC motor is by using an AC motor controller, or variable frequency drive, that utilizes the line frequencies to achieve the desire input voltage. That type of motor controller is more expensive and complicated. All of those additional components and complexity needed to control an AC motor give the DC motors the advantage in that aspect.

The cost for our application is comparable with the AC motor usually being cheaper. This doesn't include the cost of the additional components necessary to make the technology work. With this components included the DC motors were usually the better choice. Not to mention that they are more readily available to some of the members of the group.

The last but most certainly not least factor that we took into account when deciding between the AC and DC motor was its performance. The motors need to be powerful enough in order to be able to mobilize the weight of all of the components of the robot which with the addition of the battery is estimated at about seventy pounds. The motors will have to be able to overcome the static friction at first which will be the greatest force it will have to overcome as any of the coefficients of rolling resistance are significantly lower. The torque characteristics of the two types of motor is very similar up to the point that it reaches its base revolutions per minute, after that the DC motors torque decrease at a slower rate than that of an AC motor. This is an advantage to the DC motors as it increases their overall torque rating.

Looking at the results of the comparison against our deciding factors it is clear that utilizing DC motors was going to be our best alternative. Their simplicity of its power and control methods will directly translate into reliability and an overall better quality of the product. The performance would be very similar but having the overall cost of the system lower cemented the fact that it was the best course of action.

Now that we knew we would be using DC motors we had to decide between brushed or brushless motors. Brushed motors are simpler in design and implementation as it only needs two wires with a potential difference in voltage to be controlled. A brushless motor usually utilizes hall position sensors in order to know the position of the rotor. The absence of the brushes that carry the current in the brushed DC motor means that it requires less maintenance and has a longer life expectancy. The life expectancy of the brushed motor can usually be increase by getting replacement brushes. This process is not too complicated and for the scope of this project it would be a non-issue. Another difference between the two is the cost, as the brushed motors are cheaper. Since the quality of the product is not compromised by electing either of this motors, we elected the less costly and simpler to implement brushed DC motor.

Thanks to the previous experience of some of the members of the group we had some idea of the things to look for in the brushed DC motors and where to look for them as well. We found the AM802-001A motor commonly called a CIM motor, as CIM is original manufacturer. It is normally used for robotic competitions where the robots can weight upwards of a hundred and forty pounds. This is about double of what we are expecting for KnightCop to weight. This lets us know that they are able to move robots of our required size. In order to do this, the motors are coupled with gearbox of the appropriate gear ratio. This is necessary due to the fact that its typical performance (Figure 13) even under maximum power is does not provide the necessary torque to overcome the friction that it will encounter in some cases. We will try to work within its maximum efficiency range if possible. We will also be designing the system in order to minimize power consumption as the drive train will be the biggest drain on the battery. Additional hardware like couplers, hubs, shafts and brackets to hold the drivetrain and gearboxes together were available in that site and are not complicated to install. Moreover, this motor as it is more powerful than needed at the time for our application allows us some flexibility and scalability options. If we felt that adding a heavier manipulator or some other component with significant weight, we won't have to worry about the motors being capable to still move the load.

TYPICAL PERFORMANCE @ 12 Vdc

	TORQUE	SPEED	CURRENT	POWER	EFF'CY
	Oz-In	RPM (±10%)	A MAX	Wo	%
FREE LOAD	0	5310	2.7	0	0%
NORMAL LOAD	64.0	4320	27	205	63%
@MAX EFFICIENCY	45.0	4614	19.8	154	65%
@MAX POWER	171.7	2655	67.9	337	41%
@STALL	343.4	0	133.0	0	0%

HIPOT: 600 Vac/0.5 mA/1 sec
INSULATION RESISTANCE: > 10.0 M Ohm MIN
INSULATION CLASS: B

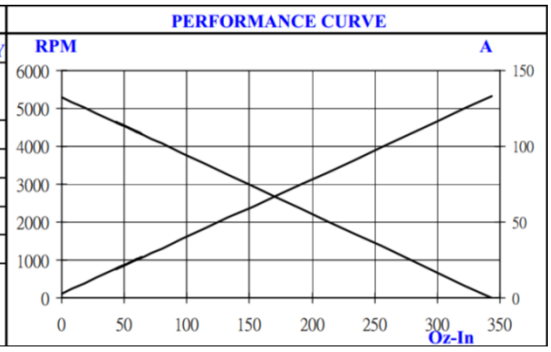


Figure 13 – CIM Motor Typical Performance

Reprinted with permission from Andymark.com

3.5.3 Motor Controller

Another key component for the mobility of KnightCop is going to be the motor controllers used to control the brushed DC motors. There are magnitudes of motor controllers out in the market and we have already made one decision that eliminated a portion of them from our list. Since we are controlling a DC, more specifically a brushed DC, motor, we know that all they require is a potential voltage differential between its two leads that is within its specifications. In other words, we could just connect a battery with the right amount of voltage supply and it would work. This is not very useful as it would only run at one speed and direction without ever changing, as long as the battery remains charged. We need to be able to control the direction and have at least the ability to turn it off in

order to have the basic functions of moving forward, backward, turn and stop. This can be accomplished with some very inexpensive and simple motor controller designs like an H-Bridge relay. We will not be taking this route alone as the control of the robot would be very rough as only one speed would be able to be used. What we need is to be able to control both the speed and direction of rotation. The more control we have in this two factors the more the potential for a smoother control of the platform we will have. Another really important factor in choosing motor controllers is their ability to deliver the current necessary to the motors.

Due to the availability to some of the members in the group and their proven effectiveness with similar applications, two motor controllers where researched and analyzed.

The first motor control module to be analyzed is commonly known as a Jaguar motor controller. It is utilized for variable speed control, both forward and reverse, of brushed DC motors. Its voltage and current specifications (table 12) match and surpass our needs as we are supplying twelve volts to our motors and don't plan on drawing more than ten amps in the worst case scenario. It is still good to have plenty of room when it comes to current consumption as current surges might occur depending of the environment we would be trying to traverse and the manner of which the user controls it. In addition to the spare room, the jaguar's firmware also has a software current limit capability to protect the hardware from being damaged. It has three current stages that cause different preemptive measures. If the current draw is under forty amps, the current limiting software will not shut the controller off. If the current is between forty and sixty amps, the current limiting software will only shut off the controller if this current draw is experienced for longer than two seconds. If the current is any larger than sixty amps for any amount of time, the current limiting software will shut power off the controller immediately. Moreover, it comes with a built in fan to keep down the temperature of the device, as heat is a common byproduct of high currents.

Minimum Supply Voltage (Vin)	0V
Maximum Supply Voltage (Vin)	30V
Minimum Output Voltage	0
Maximum Output Voltage	V_{in}
Continuous Current	40A
Surge Current (2 seconds)	60A

Table 12 – Jaguar Motor Controller Electrical Specifications

The jaguar has three options of input communication in order to control the speed or voltage output. The first is an industry standard RC style servo type interface, which utilizes a pulse width modulation (PWM) signal. The second option is a controller area network (CAN) interface. CAN bus is a standard message based protocol that allows for the microcontroller to communicate with multiple devices, in this case motor controllers, through the use of one bus. Each Jaguar module would be accessed with an initially assigned identification number. Lastly, the jaguar supports control and configuration over a standard serial, RS232C, interface.

The jaguar also allows for an array of features depending on the mode of communication used. As it can be seen by the comparison of control methods bellow (Table 13), Utilizing the CAN interface allows for close loop control of the speed, position and motor current. As opposed to the open loop control of the speed only by the PWM interface. The only problem with it is its added complexity at a software and hardware level as additional hardware is needed to implement the can interface. The PWM interface gives us the features that we need for our application without any extra complications. The speed control based on the difference of pulse widths is pretty simple to implement with our microcontroller. The limit switch feature is in case we would like to shut down the motor being controlled in either direction of rotation. The coast/break feature selects the dynamic behavior of the motor at the moment of deceleration or stopping and is selected by setting a jumper between the correct pin locations as shown in figure 14. While on a coast setting, the current is allowed to decay at a slow pace which provides the outcome of a gentler deceleration. While on a brake setting, the current generated by the motors is opposed, this provides a faster deceleration. In addition, the break setting provides additional holding capacity when the motor is stopped. The status LEDs indicate codes for normal operation, faults, and CAN conditions.

	Control Method	
	Servo-Style PWM Input	CAN Interface
Speed Control	Yes	Yes
Analog Position Control	No	Yes
Encoder Position Control	No	Yes
Configurable Parameters	No	Yes
Voltage, Current Measurement	No	Yes
Limit Switches	Yes	Yes
Coast/Brake Feature	Yes	Yes
Firmware Update	No	Yes

Table 13 – Jaguar Comparison of Control Methods

Reprinted with permission from vexrobotics.com

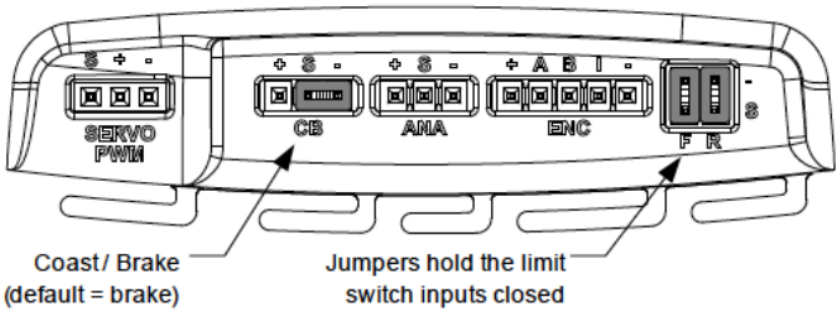


Figure 14 – Jaguar Jumper Configuration

Reprinted with permission from vexrobotics.com

The other motor control module that we analyzed is commonly referred to as a Victor 888 motor controller. The Victor has a lot of similarities to the Jaguar motor controller that we analyzed earlier, with the main difference being its design paradigm of simplicity. It is also utilized to control the voltage supplied to brushed DC motors. Its voltage and current specifications (table 14) match and surpass our needs as we are supplying twelve volts to our motors. It beats the Jaguar in allowable continuous and surge current by a significant margin. As mentioned before this is a good advantage as it prolongs the life expectancy of the device and prevents damages incurred by high current draws. It also comes with a built in fan to keep down the temperature of the device. It lacks software current limiting, but with the higher resistance to current surges and even continuous current, this software is most likely not needed. Another way this potential problem would be circumvented would be by running its input voltage to an appropriate circuit breaker first.

Minimum Supply Voltage (Vin)	6V
Maximum Supply Voltage (Vin)	15V
Minimum Output Voltage	0
Maximum Output Voltage	Vin
Continuous Current	60A
Surge Current (2 seconds)	150A

Table 14 – Victor Motor Controller Electrical Specifications

The victor provides a single medium of communication for control, the industry standard RC type PWM. It only includes one of the special features, the coast or break setting, which the jaguar provides. This is not a problem as this was the

only feature that we were planning on implement. What lacks in features it gains in size, as it accomplishes our requirements with a smaller package as it is only two inches tall, two and three quarters inches long, and two and a quarter inches wide. Compared to the two inches tall, four and a quarter inches long and three and three quarters inches wide jaguar. Its LED can indicate direction of motion and a lack of control input. Additionally it can be calibrated to the PWM input that the microcontroller provides. In calibration mode it will records the minimum, neutral, maximum PWM signal detected and adjusts its output to be full speed reverse, stop, and full speed forward accordingly.

After reviewing all of the factors and comparing the two controllers we decided to use the Victor 888. Since both controllers met our initial requirements we had to make the decision based on other factors that we felt were important. The fact that the Victor implemented less features was an advantage as its design was simpler. Simplicity usually leads to robustness and reliability, which is what we found to be the usual comments on reviews comparing the Victor and Jaguar controllers. The cost to the group was another factor that led to this decision. The Jaguar only costs ten dollars more than the Victor but the fact that one of our sponsors was willing to supply us with four Victors made it a huge advantage. All of this being said, the Jaguar is a great motor controller with a wealth of features and we will keep it as an optional upgrade if we decide that we would benefit from implementing them, but as the project stands we don't feel is a necessity.

3.5.4 Proximity Sensors

In order to achieve any level of obstacle avoidance or obstacle indication to the user, we need to be able to detect when those obstacles are in the vicinity of the user. In order to solve this problem we investigated three different technologies.

The first thing that we thought about was to utilize vibration sensors. The Minisense 100 from Measurement Specialties was the sensor we studied in order to make our analysis of this technology. The way this sensor works is by stressing a strip of piezoelectric material. When this material is stressed produces a voltage spike. In the case of the Minisense 100 this spike can be upwards of ninety volts. That means that in order to interface it to our microcontroller the voltages will need to be brought down to safe levels by using the right amount of resistance. These spikes in voltage can then be translated into the G-force experienced by the sensor. The way this technology can be implemented to detect obstacles is by contacting the obstacles before the platform of the vehicle does. Our inspiration for this idea was the way some nocturnal mammals use their whiskers to navigate their environment when they can't relay in their sight. A very specify configuration (Figure 15) will need to be used in order get the coverage needed to detect obstacles in KnightCop's surroundings. This means that the effective footprint of the robot is basically double which will limit our mobility and effectiveness in completing our mission. Additionally objects that are farther away won't be anticipated with as much time

as with a different system and the coverage of objects that might be taller or shorter than the vertical location of the “whiskers” might not be detected.

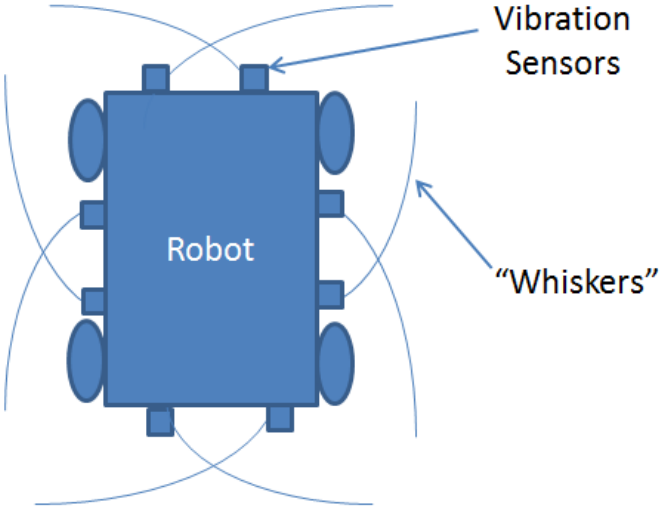


Figure 15 – Sample Vibration Sensor Configuration

Next we tried to find a different technology that would address the problem of the vibration sensors. An infrared solution (IR) resulted from the search. In specific we analyzed the GP2Y0A02YK0F from SHARP. This sensor allowed us to have a touch-less object detection which is a major advantage over our “whisker” solution. In essence it measures the distance between itself and the object directly in front of it. It has the ability to measure distances ranging from twenty to one hundred and fifty centimeters. That would be enough distance detection for early avoidance protocols to be very effective. Its low range might be a decrement as depending of the scenario we might want to know if an object is closer than twenty centimeters. Its supply current is very efficient as it only typically consumes thirty three milliamps. It utilizes a very common supply voltage of five volts plus or minus five volts. The last thing that we looked at for this sensor was the update time for the distance readings. Looking at the time diagram for the sensor (Figure 16) the readings will be updating about every forty milliseconds which is definitely fast enough for our application

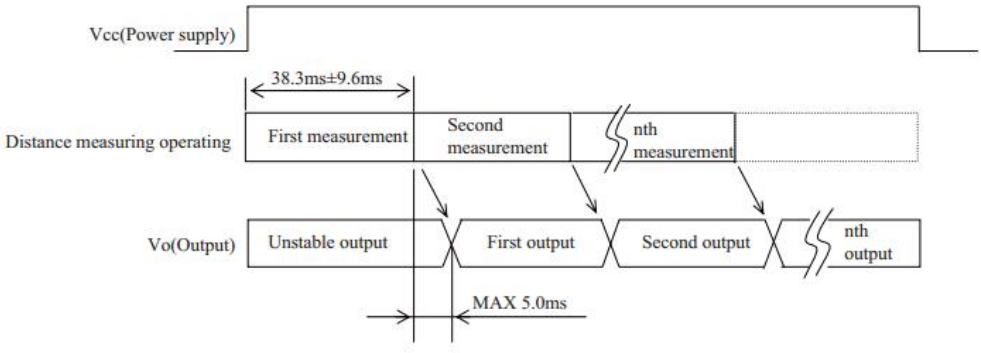


Figure 16– SHARP IR Sensor Timing Chart

Reprinted with permission from sparkfun.com

Even though the infrared solution seems to be an adequate solution for our design, we wanted to find a solution that would have better coverage of the surrounding areas and detect objects that were both close and relatively farther away from KnightCop. While investigating the IR solution we came across a variety of sensors that utilize ultrasonic technology to do their distance measurements. We analyzed two difference sensors in this instance as they had significantly different design paradigms and interfaces.

The first ultrasonic ranging module that we analyzed was the HC – SR04. This ranging module is a clear improvement when it comes to distance range as it can measures distances from two centimeters to four hundred centimeters. Not only this but it can offer and accuracy of up to three millimeters. It also provides a higher range of coverage as it has a measuring angle of fifteen degrees, which means that it won't only detect objects that are directly in front of it but also those that fall between fifteen degrees of the triggered signal. It also works on the very common five volts and consumes less energy than the IR solution at only fifteen milliamps. The way it works is by the microcontroller to set the trigger pin for at least ten microseconds, then the module would automatically send eight sonic signals at forty kilohertz and if there is a signal back it will set the echo pin to high for a duration proportional to the sound flight time. This process, as depicted by the timing diagram of the device (Figure 17), can take over sixty milliseconds as a hundred and forty eight micro seconds equals an inch on this device. Since this comes up to be just over sixteen times per second, it is still a good update time for our application.

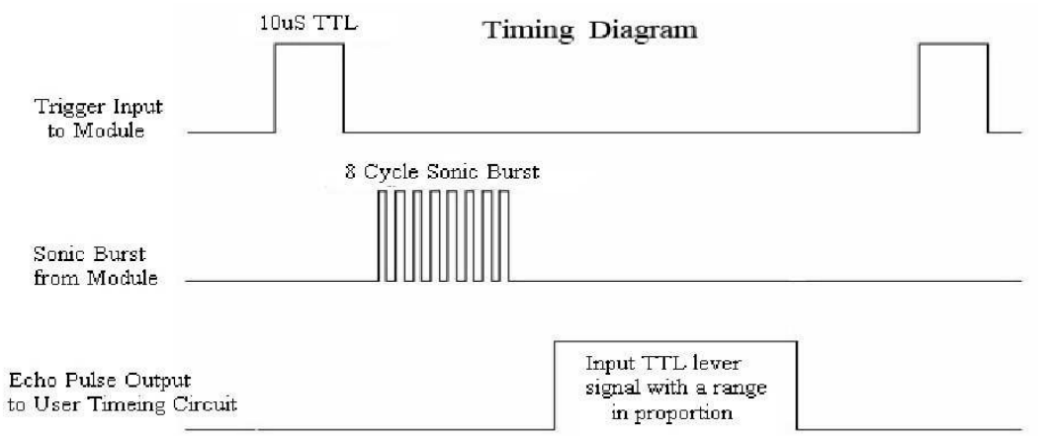


Figure 17– HC – SR04 Timing Chart

Reprinted with permission from sparkfun.com

The other ultrasonic ranging module that we analyzed was the LV-MaxSonar-EZ0. The documentation and help resources for the module were much greater as it is part of a much bigger family of range finders from Maxbotix. It has the ability to detect objects from zero inches to two hundred and fifty four inches but it will only provide sonar range from six to two hundred and fifty four inches. This

means that anything closer to six inches will be depicted as being six inches away. Its beam pattern on a twelve inch grid (Figure 18) depicts its characteristic with objects of different sizes, the smaller of which is a quarter of an inch. This showed us the best coverage out of any of the systems we had analyzed its operating voltage is very versatile ranging from three to five volts. A great advantage is gained in its power savings as it typically draws two milliamps, which is a lot lower than anything else we have seen. It is also very flexible in its modes of operation as it can be externally or internally triggered. When internally triggered the readings occur every fifty milliseconds and will be continually measured and output range information. When externally triggered the range readings can be delivered as desired up to every fifty milliseconds. It also has the advantage of being able to trigger other devices once it is done reading in order to get a cascading effect so that there is less noise created in the signal by neighboring devices. It has three different output modes that can be accessed by the user, serial, analog and pulse width. This means that the sensor reports range data directly instead of letting the microcontroller do the additional calculations necessary.

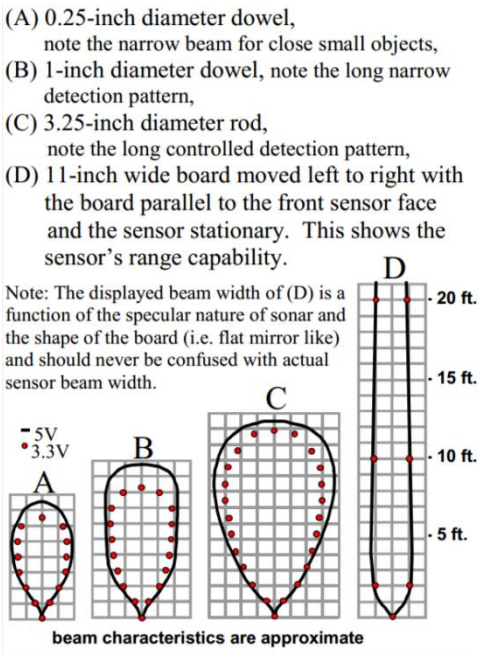


Figure 18 – EZ0 Beam Characteristics

Reprinted with permission from maxbotix.com

After comparing all of our options it was cleared that the "Whisker" solution had too many down falls. The Ultrasonic technology fared better than IR in almost every category. When it came to picking the specific ultrasonic ranging module the decision was not as clear but we came to a consensus that the EZ0 was the best for our application even though it was higher in price than the SR04. Its automatic calculation of range, ability to trigger other devices sequentially and flexibility of modes of operation made our decision easier.

3.5.5 Environment Sensors

One of the principal duties of KnightCop is to relay information about the surroundings it is in. This is easily achieved through environment and chemical sensors.

Biometric sensors: Fingerprint scanners and heart beat monitors were readily available, but the cost was too inhibiting. One of our major goals is keeping KnightCop affordable, and we could not justify spending 10% of our estimated budget over a superfluous add-on.KnightCop's primary mission is recon and not rescue missions, therefore we decided against adding these sensors to it.

Ambient Light Sensors: We decided to use an ambient light sensor to automatically toggle the light mounted on KnightCop's gripper as it enters a dark area. There will still be an option to manually switch this light ON/OFF. The following 15 shows various sensors we considered:

Sensor Make	**Vishay**	**SEN-09088 Photo Cell**	**ROHM Semiconductors**
Part #	TEPT4400	09088	BH1603FVC-TR
Peak Wavelength (nm)	570	550	560
Operating Temperature (°C)	-40 to +85	-30 to +70	-40 to +85
Mounting Method	Through Hole	Through Hole	Surface Mount
Peak Supply Voltage (V)	6	150	5.5
Power Dissipation (mW)	100	100	260
Cost (USD)	0.62	1.50	1.15

Table 15 - Comparison between Ambient Light Sensors

The Photo-resistor SEN-09088 seems to fulfill our needs best. It is a through-hole component, which are easier to work with. There is also no major power dissipation like in the case of the ROHM sensor.

Toxic Gas Sensors: There are various sensors available for detecting the presence of gases like Carbon Monoxide, Methane, LPG and even Hydrogen. While assembly is easy with breakout boards, the sensors have to be calibrated and adjusted for use, which needs to be done in a controlled environment. Furthermore, a hot and humid environment can cause the readings to go astray. Since none of our group members has access to a Chemistry lab, we thought it would be much easier to just use a commercial smoke/CO alarm and mount it to

the robot. A Bluetooth microphone can then periodically check for audio input and pass it on to the PC. The PC can in turn detect spikes in input audio levels and alarm the remote user.

Temperature Sensor: Since we have chosen the ATmega2560 as our microcontroller, we no longer have the privilege of an integrated temperature sensor. Temperature sensors are affordable and allow us to demonstrate the ease with which KnightCop can scale depending upon the user's environment. Some even come in a waterproof casing meaning they can be used in the humid Florida environment without concern. One concern regarding these sensors was that they had to be shielded from direct sunlight, but yet exposed to the atmosphere. This meant we could not enclose KnightCop's circuitry as we had initially planned to. They also had to be isolated from the rest of the electronics as some were susceptible to picking up heat from generated by KnightCop itself. Table 16 below describes the different temperature sensors we considered:

Sensor Make	**Dallas Semiconductor**	**Analog Devices**	**Maxim Integrated**
Base Part #	DS18B20	TMP36	DS18B20
Max Power Supply (V)	5.5	5.5	5.5
Min Temperature (°C)	-55	-40	-55
Max Temperature (°C)	125	+125	+125
Accuracy (°C)	0.5	2	0.5
Output	digital	voltage	digital
Price (USD)	4.25	1.5	9.95

Table 16 - Comparison between Temperature Sensors

TMP36 is a basic temperature sensor that provides output in corresponding voltage which can then be converted to °C using a scale factor of 10 mV/°C. The other sensors are based off the DS18B20 and provide digital output. The specs are identical except for the fact that Maxim Integrated's offering is waterproof, albeit at twice the price. We chose to go with Dallas Semiconductor's DS18B20.

Smoke Sensors: Smoke sensors are very important to have in a rescue robot like KnightCop. It is vital to know if the incident zone is affected by fire or an explosion of some sort. The sensors we looked at are listed in table 17 below:

Sensor Make	**Microchip**	**Microchip**	**Microchip**
Model	RE46C114E8F	RE46C121E16F	RE46C141E16F

Operating Temperature (°C)	-10 to +60	-10 to +60	-25 to +75
Sensing Method	Ionization	Ionization	Photoelectric
Supply Current (µA)	3	4.5	5.5
Mounting Method	Through Hole	Through Hole	Through Hole
Output	Digital	Temporal Tone	Temporal Tone
Max Supply Voltage (V)	12	-	12
Cost (USD)	0.68	0.88	0.96

Table 17 - Comparison between Smoke Sensors

Not only was RE46C114E8F the cheapest option, it was also the most simple to implement. However, we decided against using a smoke sensor since testing and demonstration would have been inconvenient compared to other options.

Misc sensors: A great advantage of working in the Arduino environment is the plethora of peripherals and sensors available to the end user like Voice Recognition shields that allow a person to voice control KnightCop. However, these are mostly plug and play and do not entail enough 'design value' to be incorporated into our project.

3.6.0 Manipulator

We saw the manipulator as an opportunity to improve our controls experience as well as adding a dimension that we hadn't seen with other mobile robot projects in the past.

3.6.1 Hardware

The first and most important module for the manipulator is the actual hardware that will be used to assemble it. There were many different factors that we took into account when deciding what the manipulator had to be able to do. Our initial idea was to have a manipulator with one joint and a gripper. It would have had to be able to lift a lot of weight, be made of really strong materials and driven by strong motors. Then we realized that in order for those requirements to me met, a lot of mechanical design was going to necessary. This did not make a lot of sense especially on a team composed of computer and electrical engineers. Everything from the materials used to achieve this task to the way the linkage was going to be made would have needed to be modeled and analyzed. The next thing that we realized was that in order to lift a lot of weight with a manipulator the gripper would have needed to be strong enough to deal with the

forces which meant that it was going to be another complete mechanical design necessary in order to have any type of dexterity and still achieve this type of strength. Moreover, the design of the mobile base would have been completely dependent of how high and how far away from the robot the weight was going to be lifted as it would need to be able to handle the significant shift in center of mass. This meant that we would have needed the platform to be really heavy and close to the ground. Taking all of this into consideration we had a shift of paradigm and decided to go for dexterity instead of strength, a task much more suitable for electrical and computer engineers. In order to do this we decided to that our manipulator needed to lift at least a hundred grams, have a gripper with that could grab a very small object, and have multiple, more than three not included those provided by the fact that it is on a mobile platform, degrees of freedom. It also had to be relatively small and lightweight so that it did not interfere much with the center of gravity of KnightCop.

The first option that we considered was to design our own manipulator. This would give us the assurance that all of our requirements would be met. Additionally, it would give us the flexibility to incorporate components that could be updated making the manipulator features very scalable. Since we would be designing our own hardware, we could choose more inexpensive materials which would also keep the cost down. The major downfall with this solution was again the lack of mechanical engineering experience within the team. We would need to figure out which materials were going to be used in order to meet our specification. Plastics and wood where the most likely candidates as they are easier to manipulate. The problem with this was that we found it very difficult to keep the size of the manipulator compact with the tools and experience at our disposal. Additionally, the mechanical design was becoming more complex as we wanted to add more degrees of freedom to the manipulator. This meant that more linkages and joints where necessary. This complications kept adding up the amount of possible failure points that the manipulator could have if designed by us. Taking all of this into account we made the decision to concentrate our design in the control of those degrees of freedom via the utilization of sensors and control algorithms instead of the mechanical design.

We reckoned that the best way to keep most of the mechanical design out of our hands was to buy an already built manipulator system. We found this task to be incredibly difficult as we couldn’t find almost any system that met our specific requirements. The ones that met our specifications and would have been ready to install were over our budget as they usually were priced in the thousands. Even the systems that did not meet all of our requirements were still very costly. The additional effort that we had to spend on this system to figure out a way to modify them into meeting our specification was still very high for our liking.

Our final idea was to try and meet the better of the two worlds and find a system that would give us some flexibility but also keep the mechanical aspects as low as we could. We found a solution that offered us just that. The OWI-535 robotic arm (Figure 19) is a robotic arm training kit that is intended for ages thirteen and

up. This kit is meant to be assembled and then used with the provided wired controller to control the manipulator. This gives us the opportunity to make modifications as is being built in order to adapt the system to our needs but doesn't require us to figure out how to design the mechanical linkages and joints. It is very compact with a footprint after assembly of nine inches long, six inches wide and fifteen inches tall in its upright position. It is also very light weight being six hundred and fifty grams. Additional footprint and weight reduction could be gained by eliminating its wired interface and power unit. It has a really good reach for its size at twelve point six inches horizontally and fifteen inches vertically. It is also able to lift a hundred grams through its range of motion. It meets and surpasses our need for degrees of motion as it has five. It has the ability for its gripper to open and close, has a wrist motion of a hundred and twenty degrees, an elbow range of three hundred degrees, a base rotation of two hundred and seventy degrees, and finally a base motion of a hundred and eighty degrees. The motors included with the kit are already the right size and can handle the load the manipulator advertises, thus we do not find it necessary to look for an alternative but is a possible upgrade that we could make in the future. Thanks to the wired controller only having switches that are connected to the motor lead they are very accessible which makes it really easy to adapt our wiring to the motors. It additionally comes with a led embed in the gripper which depending on the brightness might be sufficient for our application. What sealed our decision to utilize this system was the fact that it only costs forty five dollars, which when compared to similar systems at three hundred dollars and already build manipulators in the thousands make it the most inexpensive option.

Figure 19 - OWI-535 Robotic Arm

Reprinted with permission from owirobots.com

3.6.2 Motor Controllers

Due to the fact that KnightCop is design a remote platform, the built in wired controller interface won't be accessible to the user. In order to give the control back to the user the microcontroller on board that is connected wireless to the user interface needs to be able to control the manipulator's motors. Since the motors that come with the robotic arm for the kit are DC brushless motors, we can apply similar concepts as those that we did when investigating the control for the mobile platform. The main difference between the two would be the desired behavioral characteristic of the manipulator's motors. For our application only the control of the direction of the motor is crucial. In other words, a control of full speed forward, full speed reverse and stop is sufficient. Thanks to this, we could get a simpler and smaller controller, which would help us maintain simplicity on our overall design and help keep the packaging of the manipulator system down. The best way was to utilize a controller based on the simple design of an H-Bridge (Figure 20)

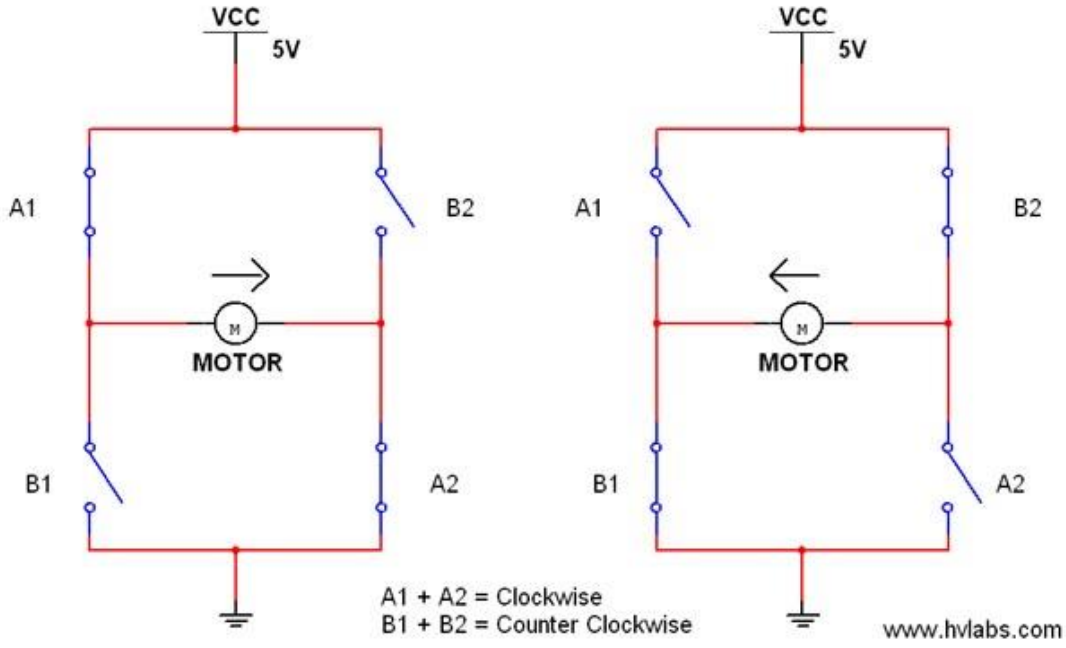


Figure 20–H-Bridge

Reprinted with permission from hvlabs.com

The First device that we found is called a Spike H-Bridge Relay. It is a relay module that is commonly used to drive small motors in forward, reverse or off depending on its three wire input (Table 18). It is designed to operate six and twelve volts and provides two outputs. The maximum amount of continuous current is regulated by an integrated twenty amp fuse to protect the hardware. Additionally it comes with an LED to indicate its operating status set by the inputs. Its footprint, about a two inch cube, is smaller than those of the more complex motor controllers analyzed with the mobile base research mainly

because it does not require an integrated fan. At a price of thirty five dollars was more than we were expecting for a simpler design.

INPUTS		OUTPUTS			
Fwd(Wht)	Rev(Red)	M+	M-	Indicator	Motor Function
0	0	GND	GND	Orange	OFF / Brake Condition (default)
1	0	+12v	GND	Green	Motor rotates in one direction
0	1	GND	+12v	Red	Motor rotates in opposite direction
1	1	+12v	+12v	Off	OFF / Brake Condition

Table 18 – Spike Method of Control

Reprinted with permission from vexrobotics.com

An alternative to the Spike relay was the DRV8833PWPR (Figure 21), which is a dual H-Bridge motor driver from Texas Instruments (TI). It has the ability to run either two DC brushed motors or a single stepper motor. As we are using DC brushed motors in our manipulators we would be controlling two for every driver, which would help us in both cost and space savings. In order to make sure that this IC could work for our application the first things that we looked at were the voltage and current recommended operating conditions. They were well within our range as the motor voltage range went up to eleven volts, the digital input voltage to six volts, and the output current ratings to one point five amps. Not only were they within range but the device features current control, overcurrent, under-voltage, and over-temperature protection. Additionally, it has an energy saving mode. Its footprint is incredibly small, about a quarter of an inch square. It uses two digital inputs in order to control each motor. Which could either use H-Bridge logic or PWM control as depicted in Table 19 This meant that we could also have the option to control the speed if of the motors if desired. The last and probably most shocking of its characteristics was its price, where depending on the distributor was between one and two dollars per device.

xIN1	xIN2	xOUT1	xOUT2	FUNCTION
0	0	Z	Z	Coast/fast decay
0	1	L	H	Reverse
1	0	H	L	Forward
1	1	L	L	Brake/slow decay

xIN1	xIN2	FUNCTION
PWM	0	Forward PWM, fast decay
1	PWM	Forward PWM, slow decay
0	PWM	Reverse PWM, fast decay
PWM	1	Reverse PWM, slow decay

Table 19 - DRV8833PWPR: H-Bridge (left) and PWM (right) logic

Reprinted with permission from ti.com

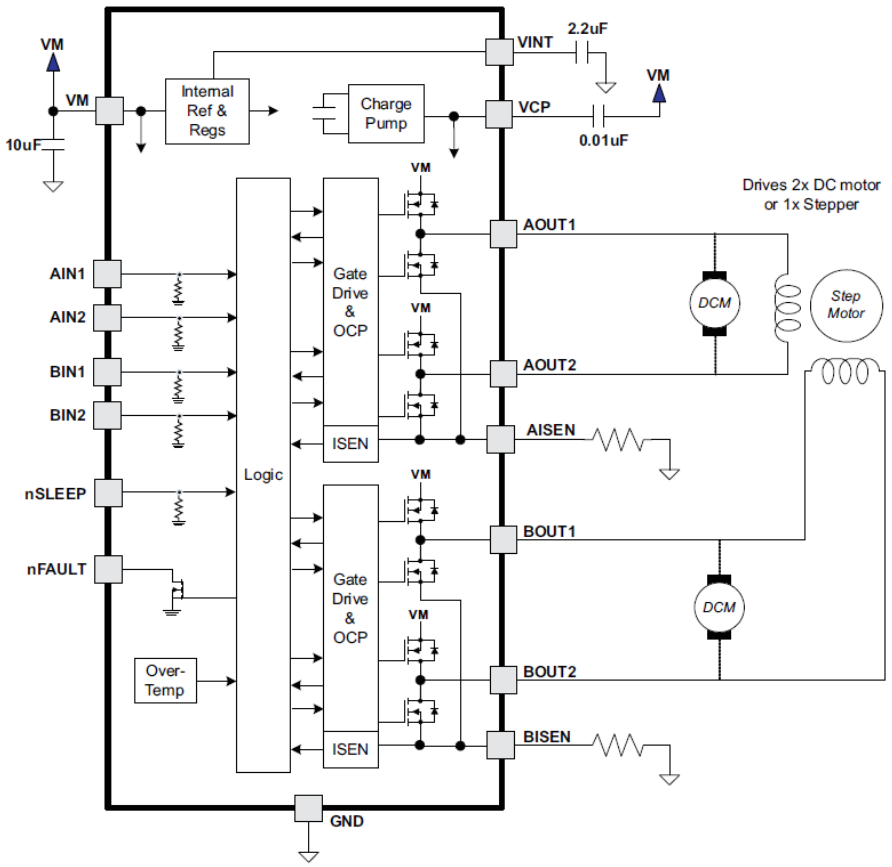


Figure 21 - DRV8833PWPR Functional Block Diagram

Reprinted with permission from ti.com

This was definitely the easiest decision we had up to this point as the DRV8833PWPR was an improvement to the Spike relay on almost every category. Even in those specifications that it did not come ahead, they were well within our requirements.

3.7.0 Video Feedback:

Video feedback could be provided by mounting a CCD camera on the robot and have the microcontroller encode/decode the images, but it would have been too taxing for the class of microcontrollers we were looking at. We also wanted to separate the video feedback entirely from the robot's operation so that video processing does not bottleneck, or worse, block control signals being sent to KnighCop.

This led us to look into IP cameras, which work over Wi-Fi and have embedded HTTP servers. This would allow us to maintain our modular design by keeping

video separate from robot operation. Table 20 summarizes the various offerings we looked at.

Make	Samsung	Foscam	HooToo
Model	SNH-1011	FI8910W	HT-IP210F
Wi-Fi	Yes	Yes	Yes
Night Vision	Yes	Yes	Yes
Resolution (px)	640*480	640*480	640*480
Frame Rate (fps)	30	30	30
Pan/Tilt	No	Yes	Yes
Video Codec	H.264, MJPEG	MJPEG	MJPEG
Cost (USD)	96.75	76	69.99

Table 20 - Comparing IP Cameras

We chose the Foscam FI8910W IP Camera mainly because it was very popular among hobbyists and robotics enthusiasts per our research. Even though its competition had similar specs and pricing, Foscam forums were very detailed and the community was very helpful. The documentation on the camera was abundant, including CGI commands that would allow us to pan/tilt, toggle night vision and control every other aspect of the camera.

3.8.0 Software:

3.8.1 Software Communication

KnightCop can be controlled from both the PC application and the Android app. Both devices have integrated Wi-Fi network cards that will communicate with KnightCop's Wi-Fi module over Java Sockets. KnightCop would receive control commands (for drive controls and data requests) and respond with appropriate movement and/or sensor feedback. The video stream and camera controls are independent and do not interfere with the robot's operation.

Figure 22 shows the direction of data flow between the three major components.

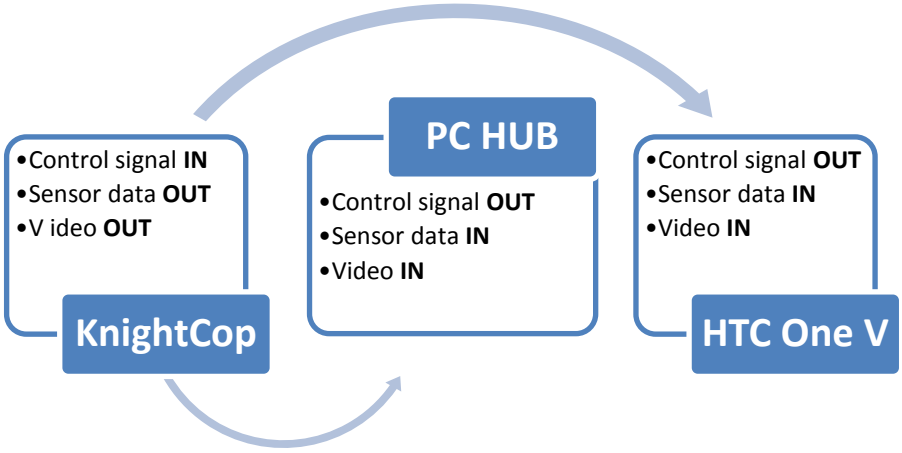


Figure 22 - Control and Feedback Flow

Both PC and Android apps will have full functionality when controlling the robot and its sub-systems. However, the Wi-Fi module we have chosen can only stream data over one port and one Wi-Fi channel, which effectively means that both applications may not interact with KnightCop at the same time.

This is to our benefit since we no longer have to implement conflict resolution in both applications. KnightCop would simply reject any attempts to open a new socket on its port and we can catch and handle this in an exception.

The primary reason behind this setup is that we will be using a wireless IP Video camera, which allows us to ramp up the video capture resolution and not create a bottleneck for sensors and control data being exchanged with the microcontroller. This also gives us the option of implementing video or image processing features to the project if time permits. The camera is capable of streaming to 4 devices concurrently.

3.8.2Android Development

Initially, we had the option of using an iOS or Android device as the controller in our project, given that Elean has an iPhone and Nitin has an Android phone. Development for iOS is done in Objective-C (which is a superset of C) and would nicely complement our microcontroller programming (which would be done in C/C++). Android development, on the other hand, takes place in Java which allows us to demonstrate proficiency in multiple programming languages.

Android's support of open source was another point in its favor. Within minutes we had downloaded Android Developer Tools (hereafter referred to as ADT) and were reading a sample open source project. ADT is bundled with Eclipse, which is a popular IDE we both are familiar with.

iOS Software Development Kit is only available on Apple computers running on Intel processors. Although this has workarounds like Virtual Machines or running

a Hackintosh (a PC running a Macintosh OS), these are neither supported by Apple nor legal. With this we had no choice but to go for Android.

App-development on Android is a straightforward process. Compiled Java code and edited XML documents can be run on a virtual device thanks to the bundled emulator which speeds up testing and debugging because software does not have to be uploaded to a real device every run.

Sensor data will be fetched from KnightCop and video stream will be captured directly from the IP Camera and displayed on the phone's UI. The UI will have controls translate movements to the robot's drive train and arm. The UI will also allow the camera to be panned or tilted.

UI controls would be run as a 'Foreground processes' since the user will directly interact with these. Data feed displays will be run as 'Visible processes' since these are for informational purposes only.

3.8.3 PC HUB

The Windows application will be written in Java using the Netbeans IDE. The GUI will be in Swing framework, since JavaFX and other modern frameworks are more rigid with respect to GUI rendering threads.

The PC will connect to KnightCop using Java Socket, which will remain open for the duration of each session. Terminating the application closes the socket, allowing the Android application to take over. Sockets also allow us to have multiple connections between the host and the client, which means we can have independent sensor data and video streams, while having a dedicated channel for transmitting controls to the robot.

Video feedback would have a dedicated thread process catering to it. This thread will be responsible for fetching the stream of Motion JPEGS and extracting and rendering these images to the GUI.

3.8.4 Microcontroller Programming

AVRs boast of GCC (GNU Compiler Collection) support, which translates to a hassle free coding environment, unlike the PIC family which was found to have notorious high level language compilers in our research. We also downloaded the freely available Atmel Studio 6 and WinAVR (for Atmels) and compared these. Table 20 shows a comparison between the major software toolsets.

Not only is Atmel Studio 6 the most polished product out of all these, it can also be used to program for Arduino boards (which will be the Arduino Mega 2560 in our case).

	WinAVR	Atmel Studio 6	Arduino IDE
Chips Supported	Atmel	Atmel	Atmel
Platform	Cross	Windows	Cross
GUI / IDE	Command Line	Yes	Yes
Higher Languages	C, C++	C, C++	C, C++
Code Optimization	Full	Full	Full

Table 21 - Comparison of Software Toolchains

We will bypass the default bootloader that comes installed on the microcontroller with Arduino Mega 2560. This will free up 8 KB of memory on the microcontroller. It will also remove the slight delay that happens when the development board is powered on or reset. The bootloader allows users to upload their code while the microcontroller is still plugged into the development board. However, for Senior Design we will be using In-Circuit Serial Programming (ICSP). This can be achieved by using external programmer devices like USBtinyISP (or derivatives of it). This is an open source project kit that must be assembled by users themselves. It uses SPI to interface with the microcontroller and connects to a PC with a USB cable.

In effect, we will code in the Arduino IDE, grab the hex dump and program the Atmgea2560 on our PCB with these using Atmel Studio. This is faster than programming the Arduino Mega as an ISP and also allows us to use the Arduino for testing code beforehand.

3.9.0 Computer Hardware:

The computer, being the hub of this project, needs to be able to delegate and relegate tasks as the need arises. The laptop we chose has an inbuilt Wi-Fi card to communicate with the other devices. It also has a USB 3.0 port which is capable of keeping the Arduino Mega powered on even if the laptop itself is turned off (as long as the laptop is connected to a power supply). This may come in handy in case the Windows system crashes.

It also has an Intel core i7-3630QM processor running at 2.4 GHz (each core) which would enable us to run our multi-threaded PC application seamlessly. A powerful processor would also be useful for encoding the video in real time for relay to the Android phone. We can designate one core each to a distinct function of the application, as shown in figure 23 below:

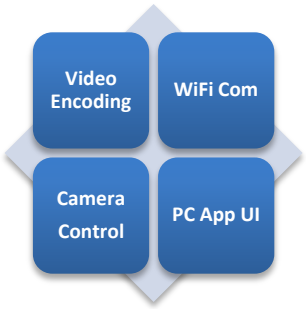


Figure 23 - Thread Designation to Processor Cores

4.0.0 Hardware and Software Design

4.1.0 Board Components:

4.1.1 Microcontroller: ATmega2560

We have made an effort to efficiently utilize all the pins and interfaces provided on the board. This is where choosing ATmega2560 pays off. As is evident in figure 24, we make use of almost all the digital PWM pins and well over half of the analog pins offered by the microcontroller. Other microcontrollers with similar specs would have run out of analog pins, forcing us to utilize digital pins for the ultrasonic sensors which needlessly complicates computation and design.

The robot arm uses 10 digital PWM pins for controlling the motors, plus an additional digital pin for sleep mode. Putting the arm to sleep (when not in use) should allow us to save a considerable amount of battery power. It will also use an additional pin to toggle the limit switch as well as the light bulb fixed inside the gripper cavity.

The drivetrain motors would use two additional PWM pins to operate. Potentiometers (for determining position and direction of rotation/movement) take up 5 analog pins. The ultrasonic sensors will be mounted in one direction each and require 5 additional analog pins. We will also need a digital pin as a trigger signal for these sensors. The temperature sensor is simple and only required one digital pin. The Wi-Fi module requires 1 digital pin for communication and an additional digital pin running to its GPIO 9 pin for toggling ad-hoc mode.

The microcontroller will be powered by a 5 V regulated connection from the Arduino/PCB. During development, we will use the clock provided in the Arduino Mega. Notice that not all pins are available since some are used by the ArduinoMega's peripherals. Extraneous/unused pins have been omitted.

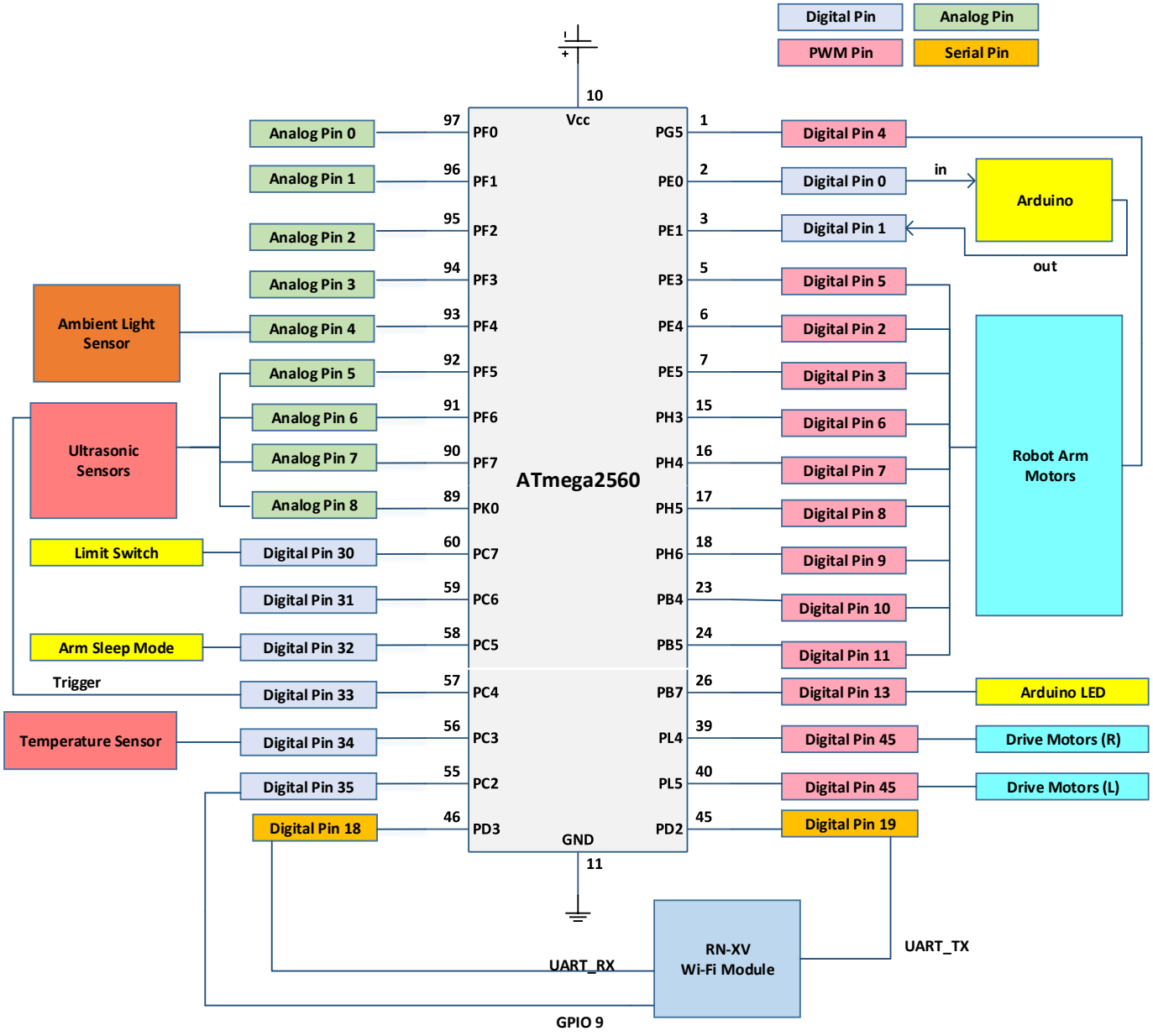


Figure 24 - ATmega2560 Block Diagram

4.1.2 Wi-Fi Module: RN-XVee

We have chosen the RN171XVW-I/RM Wi-Fi module from Roving Networks. It is a simple through-hole 802.11 b/g solution and is a third of the cost of implementing an XBee connection for our project. This particular module will use UART to interface with the microcontroller. It also has 8 general purpose I/O pins, something we do not have plans for yet. There are also 3 analog sensor interfaces, which means we can send sensor data from ultrasonic sensors directly to the PC Hub. This would be desirable in case we want to add more automation to the robot with the help of complex course-plotting algorithms.

Figure 25 below shows the block diagram of RN-XVee as we wish to implement it:

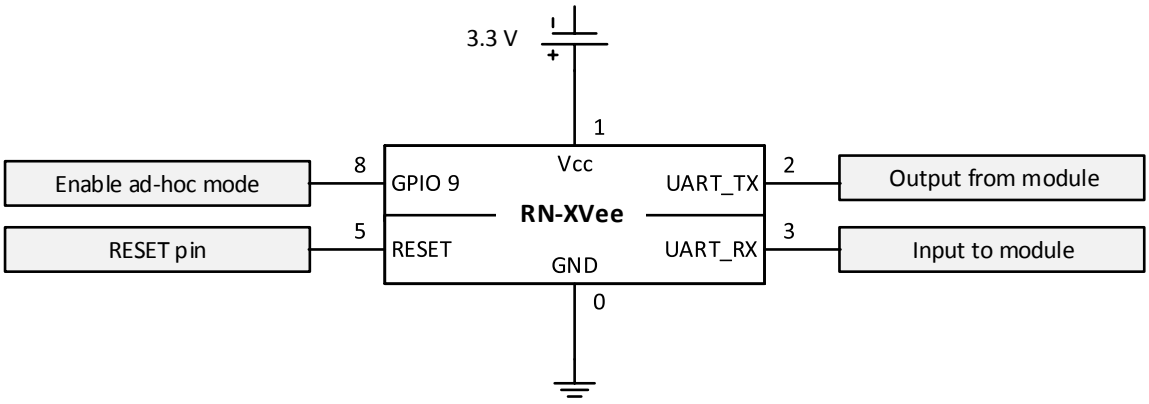


Figure 25 - RN-XVee Block Diagram

RN-XVee needs a regulated 3.3 V power source to function. Pin 2 is used to send output to the microcontroller while pin 3 is used to accept input from the microcontroller. Pin 9 can be used as a general purpose I/O pin, but we will utilize it to toggle the module into ad-hoc mode. Pin 5 can be used to reset the module in case Wi-Fi configuration is modified. This saves us from power cycling the whole unit.

4.2.0 Mobile Base

4.2.1 Platform Hardware

Since we decided to design and build our own platform, there were several aspects that we took into consideration. The first was the structural material that we were going to use to build the chassis with. Based on our research we knew that we wanted to use metal to construct the frame. We decided to use predrilled 5052-H34 aluminum angle for this application. The strength of this material and ease of use was what sold us on it. Its yield strength, the point at which permanent deformation starts, is in the range of thirty to thirty five thousand pounds per square inch. It is one inch tall, one inch wide, with a thickness of an eighth of an inch. We do not expect to experience any force close to its yield strength. 5052 aluminum is softer than other aluminum alloys which make it easier to bend if we needed. Additionally its pre-drilled quarter inch holes at an inch interval, on both its vertical and horizontal surfaces, allow us to assemble it without the need of welding the material together, as it can be seen in figure 26.

We will be using bolts and nuts with washers to hold the structural members together. Another advantage of this material is that it comes in members of seventy two inches long. This means that all we needed to do was figure out how big we wanted it to be and just cut the members to size with a band saw that was available to us. Additionally mounting of additional structural members for

support and construction and installation of brackets was made a breeze as we just cut the pieces to size and bolted them together.

We are only enclosing the boards we will be building and other small electronics as the environmental sensors, manipulator, camera, and antenna all have to be outside of an enclosing. The motors are sealed and so we did not feel the need to add them to the enclosing. It is a medium size robot as we wanted to have a good mix between maneuverability over objects and portability.The reason we kept the wheels inside of the frame is protection and durability to our design as it also allows us to have brackets that wheel have bearings to support the wheel axel from both sides of the wheel.We also have additional members in the middle of the robot for structural integrity to prevent bending and deformation. It additionally serves a double purpose to mount electronics and other modules like the Manipulator.

Figure 26 – Chassis

Another very important component that we took into consideration was the wheels we are going to use. We picked to utilize six inch plaction wheels with a roughtop tread from AndyMark. Plaction is a term used by the AndyMark to illustrate the fact that the wheels base is made of a strong plastic compound and that it is fitted with a high traction thread that gives it more traction than just the plastic would. Therefore plastic plus traction would equal placation.

The plastic base of the wheels is very strong, as it has an estimated load capacity of a hundred and fifty pounds. This is more than enough for us because the four wheels together would be able to hold six hundred pounds if the load was distributed evenly. The base also gives us flexibility as the bolt patter around the hub is standardized with their line of hubs that connect to a large range of different axel shapes and sizes. Thanks to this, we were not very limited as to what gearbox and axel shaft we could use.

Another important factor that led us to pick this wheel type was its high traction tread. Traction is important because we want to maneuver over many different types of surface. The main factor when looking for traction is the coefficient of friction between the tread and the different kinds of surfaces that we could encounter. The coefficient of static friction of the tread on dry concrete is around one and on sand is about three quarters. Having a large enough coefficient is important because we need to be able to apply enough force to get over obstacles and going up inclines. If the coefficient of friction was too low then we couldn't overcome these forces before the wheel started slipping. This is because the force that allows the wheel to propel the vehicle forward is its frictional force with the ground.

We also had to consider that the coefficient of friction couldn't be too high, as this might cause the motors to get to their stall torque. This would produce really high amounts in the amperage consumed, which might damage to its hardware and the hardware of what controls it. The overcurrent protection on the motor controller would probably shut down the motor before this occurred but this would hinder the robot inoperable until it is restarted. The ideal scenario is that the wheels start slipping before the torque and therefore amperage levels get too high. Six inches was tall enough to overcome the obstacle in the range that we wanted but also small enough to keep the speed down.

4.2.2 Motors, gearboxes, and Motor Control

Choosing the right motors for our drivetrain was extremely important. The CIM motors needed to have the amount of torque necessary in order to move the load we are expecting. This load is experienced the addition of all of the forces preventing the movement of KnightCop. These forces are dependent of the terrain condition, angle of incline that the robot wants to traverse, mechanical bindings in the hardware of the drive system, and possible obstacles that might have to be pushed or driven over. The minding inside of the gearbox can be minimized by greasing the wheels previous to usage. In order to maximize mobility and drive train strength that KnightCop needs to accomplish its task, we made sure that we could utilize all of the traction available to us. In order to do this we figured out the maximum amount of force that a robot with our physical characteristics could do. It turns out that this force is the static frictional force between KnightCop and the terrain it is navigating because if that force is overcome then the wheels will just start spinning in place and the sliding frictional force is always less than the frictional one.

For reliability and robustness in design we utilized worst case scenario numbers in our calculations. We used the biggest coefficient of friction between a terrain that we might encounter and the tread in our wheels, which was a one. We are expecting the weight of KnightCop with a regular amount of payload to be less than fifty pounds but for the purposes of our calculations we used seventy pounds. Because the frictional force is equal to the coefficient of friction, one, times the force applied to the ground, seventy, then the largest amount of force we will be able to propel KnightCop is equal to seventy pounds.

We want to be able to assure that the motors have the ability to provide enough torque at a safe amount of amps to overcome this force because if this is not the case and we encounter an object which we can't move and try to move it then either stall torque or dangerous amounts of current will be drawn. In the case of the CIM motors at a stall torque condition the maximum amount of current drawn will be a hundred and thirty three amps. This is definitely more current than most motor controllers in our price range can handle. The Victor motor controller can handle a hundred and fifty amps of surge current for two seconds but after that safety mechanism will shut it down. What we want to use as our worst case scenario is the maximum amount of current it can handle continuously which is sixty amps. Thanks to the CIM motor's performance curve we were able to plot its torque at a specific current draw. Its torque provided at sixty amps is equal to a hundred and fifty one inch ounce force. This means that the motor can provide a hundred and fifty ounces of force with a lever arm of 1 inch. This translates to about 50 ounces of force at our wheel radius of 3 inches. It was good that we picked a wheel of 3 inches in radius as the farther you get from the motor the less force it is able to provide to the ground. This meant that with four motors on a tank configuration, 2 on the left and 2 on the right, we would be able to provide twelve and a half pounds of force, which is lower than our worst case scenario of seventy pounds dictates. We were expecting this result as the motor is not geared which means that it is providing more speed than we needed.

We proceeded to calculate the speed that the ungeared motor would be propelling KnightCop, which we were able to do thanks to the CIM motor's linear relationship between torque and revolutions per minute (RPM). We calculated the RPM to be close to 3000 RPM, which with our three inch in radius wheel comes out to be over 50 mph. This was good because we realized that we had plenty of speed to loose which would be translated into torque with the right gearbox and gear ratio.

Elean had used a gearbox from AndyMark called the Toughbox Nano. It is a great gearbox for our application as it accepts the CIM motor innately and comes with the hardware necessary to make that connection. It also has a half inch diameter, 2024 aluminum, hex shaft that easily connects to the hubs provided on the site that connect to the wheels we are using (Figure 27). This greatly simplifies our design time and will save us time when assembling the parts as they are specifically made to accept each other. The gearbox is also very strong as it is made from 4140 steel. This is an advantage as we used it as a very

strong structural member because of the way it is connected to the frame (Figure 28).
The Toughbox has multiple options for the overall gear ratio that it uses, the standard being 12.75:1. Choosing the gear ratio is a crucial task that could mean the success of failure of KnightCop. The reason for this is that we need a really good balance of power and speed. Looking back at our worst case scenario the amount of force that the motors without gearing could exert was over 12 lbs of force. With the gear ratio of 12.75:1, this becomes a hundred and sixty pounds of force at sixty amps per motor. This is more than two times the amount force that we need in our worst case scenario which is great. The speed of the robot at these circumstances would be four miles per hour. That is plenty of speed considering that we will never face this force. At the 70 pounds of force that our worst case scenario presented our calculations showed that we would be consuming 28 amps and going at a speed of 6 mph. That is a very safe amount of amps for any of our components.

Even if they experience a surge, it will only be for a small moment as the wheels will start slipping. On the other extreme if the robot didn't experience any force opposing or helping its movement, perhaps on a slight downhill slope, it would consume 3 amps and have a speed of 7.5 mph which is equivalent to 11 ft/s. This is a good speed where the user can maintain control and be fast. The most likely scenario is when the rolling frictional force is the main force acting against the robot.

The worst coefficient of rolling friction for our wheels is with wet earth road, which is equal to 0.08. This means that in our worst case scenario weight of 70 lbs the force opposing movement is about 6 lbs if neglecting the friction between the mechanical components of the drive system. Based on our calculations this would equate to a current consumption of five amps per motor and a top speed of just over 7 mph.

Figure 27 – Motor, Gearbox, and Wheel Coupling

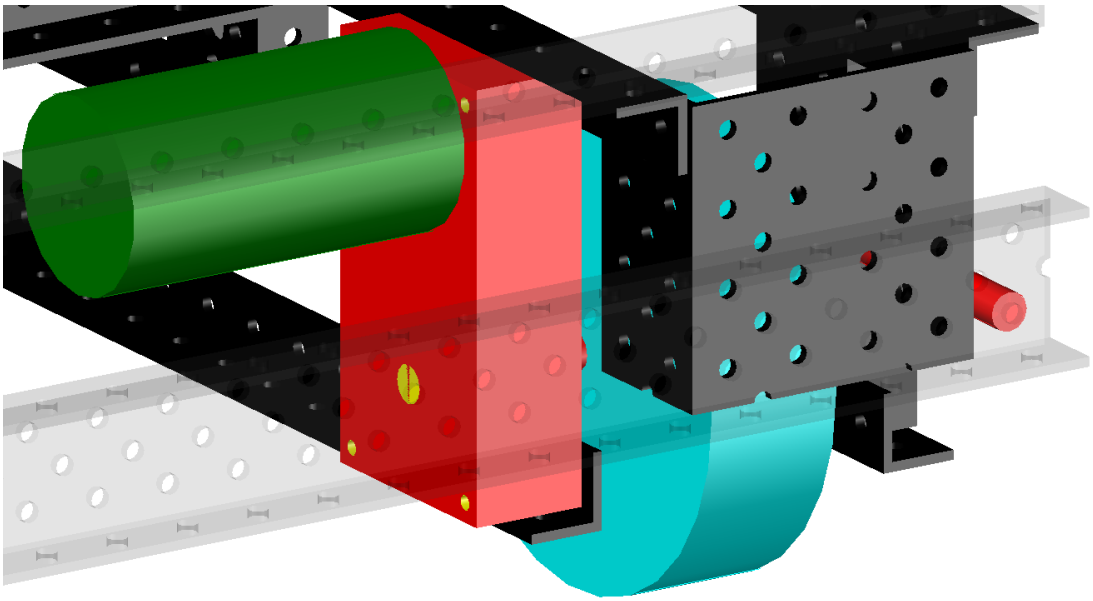


Figure 28 – Gearbox as a Structural Member

As it was implied to in previous sections, it was crucial for our motor controller to be able to handle the current draw that the drive motors might have. The Victor motor controller can handle more than twice our expected worst case scenario current draw. Its linear control behavior is very beneficial to our overall design as we will be able to have a good amount of precision in the controls. Wiring the controller (Figure 29) is very simple as it takes a PWM enabled digital input, logic power and logic ground from the microcontroller, power and ground for the motor, and it outputs power and ground for the motor that is based on the input from the microcontroller.

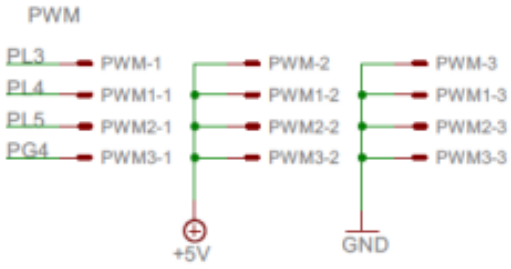


Figure 29 – Drive system wiring diagram

4.2.3 Proximity Sensors

The use of ultrasonic sensors that had a good range was important as the maximum speed of our KnightCop is going to be about ten feet per second. In other words, in order to stop or turn in time to avoid a collision we need to be able to anticipate the obstacles at a greater distance. The EZ0 allows us to do this as it can detect small obstacles up to ten feet and bigger obstacle up to

twenty feet. Its wide angle of coverage allowed us to cover the surroundings of the robot with only four sensors. Another important aspect at this speed is the update rate of its detected distance. Since its refresh rate is every fifty milliseconds, it will refresh its calculations every six inches. This is very good for our application as we will be taking avoidance actions well prior to this. We also have the options to trigger sensors that are next two each other at alternating sequence which would double the amount of time it will take, therefore doubling the distance traveled to a foot before the new distance is refreshed. We found the mounting brackets that fit the sensors perfectly and will work very well with our pre-drilled chassis. If for any reasons these don't work or are not available then we will just create our own out of wood or by printing them with the 3D printer.

4.3.0 Video System:

After much research, we have chosen the GA1A2S100LY from Foscam. It is a wireless IP camera capable of B/G/N class Wi-Fi. The camera supports hardware pan (300°) and tilt (120°), which means the viewfinder can be moved independently of the robot. It has an integrated microphone for audio support. The video resolution meets our specifications (640 * 480 pixels).

It also has better reviews than most other IP cameras in our budget (<$100). The following figure 30 shows the user interface of the bundled software (accessible through any modern web browser):

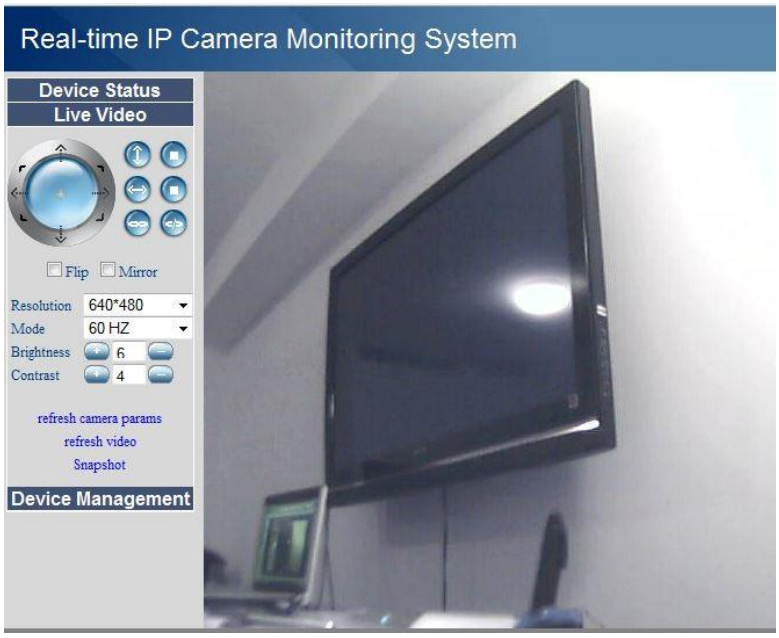


Figure 30 -Foscam GA1A2S100LY IP Camera UI

Reprinted with permission from Foscam Digital Technologies

The camera's HTTP server processes CGI commands and interprets these to camera controls like pan, tilt and patrol. These can be sent to the camera over a new socket (not the one being used for streaming video) using the open source Apache Framework which acts as a wrapper for these CGI requests. Alternatively, we can write a Javascript wrapper in our application and send commands from there.

4.4.0 Software:

4.4.1 Android Application

A screenshot of the Android application's user interface is given in figure 31 below:

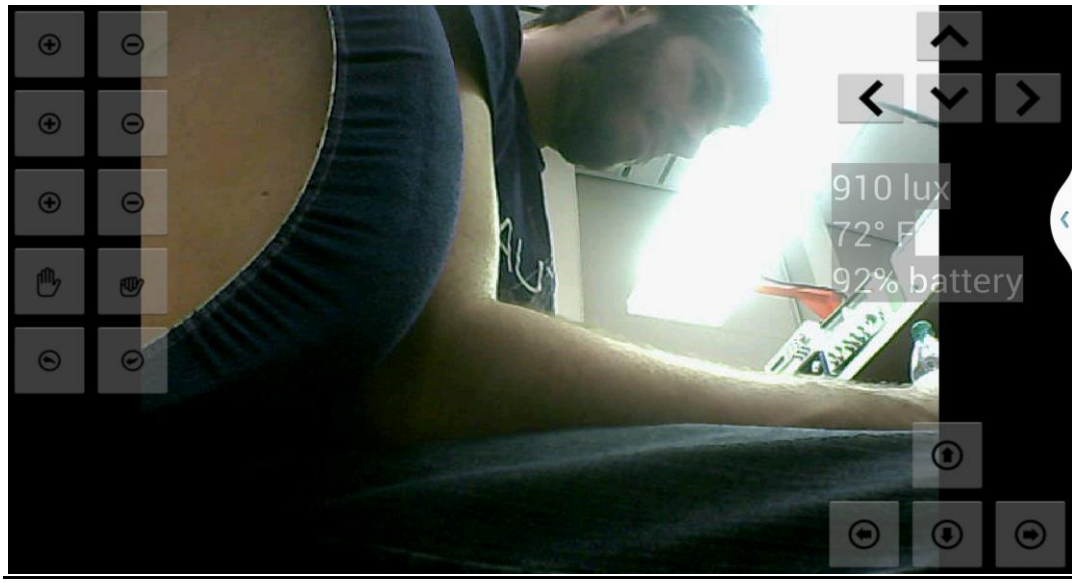


Figure 31 - Android Application UI

The Android Application UI will be divided into 3 distinct sections:

Video Feed (Visible Process): At the center of the screen will be a window with video stream from the IP camera via the PC. We will maintain an aspect ratio of 4:3 and will add the ability to pan and tilt the camera by swiping across this section of the UI. We expect some delay (mainly because the video is being re-encoded for streaming from the PC), but do not anticipate it to have drastic impact on performance.

Sensor Feedback (Visible Process): These are the temperature and ultrasonic rangefinder readings. We expect to be able to update these at an interval of 100 ms. Under normal conditions, temperature will not change significantly during such a small interval. However, a sudden rise in the reading can indicate an explosion or back draft from a fire. Similarly, while the robot will not cover much ground in the span of 100 ms, a spike in proximity readings can be the indication of collision with a foreign object or an earthquake.

Controls (Foreground Process): On the left hand side, we have controls for the drive train. Up and down arrows are for accelerating and decelerating respectively. Left and right arrows would turn the robot counter-clockwise and clockwise respectively. Rounded arrows to the right would do the same for the robot arm. Then we have controls to extend the arm out (holding this button would extend the arm as much as possible) and to contract it back in by controlling each of the three joint motors. The arm will have programmable positions. The + icon can be used to add a new custom position that the arm will remember. The lock icon will freeze current position as a new preset on the list. The user can pan or tilt the wireless camera by swiping gestures across or vertically, respectively, in the video feed region. The interfaces and classes we plan to use are listed below:

[android.view.View.OnClickListener]

Method	**Parameters**
onClick	View v
Executes code when a view (any UI element) is clicked on. This will be used to register user interaction with UI buttons and controls	

[android.view.View.OnFocusChangeListener]

Method	**Parameters**	
onFocusChange	View v	Boolean hasFocus
Executes code when a view loses or gains focus. This will be used to determine when the user wants to interact with the camera. A gesture in the video feed region will activate it and steal focus from controls.		

[android.net.wifi.p2p.WifiP2pManager]

Method	**Parameters**		
initialize	Context srcContext	LoopersrcLooper	ChannelListener listener
Attempts to register the Wi-Fi interface with current device			
discoverPeers	Channel c	ActionListener listener	
Tries to find Wi-Fi capable devices in vicinity			
connect	Channel c	WifiP2pConfig config	ActionListener listener
Attempts to connect to the device described the theconfig class			

[android.net.wifi.p2p.WifiP2pConfig]

Field	**Type**	**Description**
deviceAddress	String	The unique MAC address of the Wi-Fi peer
groupOwnerIntent	int	A value between 0 and 15 where a higher number signifies better affinity to be a ad-hoc connection group owner
wps	WpsInfo	Configuration of the protected Wi-Fi connection

Classes that we wish to implement in our Android application are listed in the following pages:

[public class VideoFeed]

Method	Parameters
onSwipeLeft	MotionEvent e
Directs IP Camera to pan 10 degrees left	
onSwipeRight	MotionEvent e
Directs IP Camera to pan 10 degrees right	
onSwipeUp	MotionEvent e
Directs IP Camera to tilt 10 degrees upwards	
onSwipeDown	MotionEvent e
Directs IP Camera to tilt 10 degrees downwards	
toggleNightMode	
Turns Night Mode on/off depending on last state	
reset	
Centers the IP camera by setting pan and tilt = 0	

[public class Position]

Field	Type	Description
jointAngle[0]	double	The angle arm joint 1 makes with the horizontal
jointAngle[1]	double	The angle arm joint 2 makes with the horizontal
jointAngle[2]	double	The angle arm joint 3 makes with the horizontal
baseAngle	double	The angle by which the arm base has been rotated from default position
gripperState	boolean	0 → gripper completely open, 1 → gripper completely closed

[public class DroidArmControls]

Method	Parameters		
changeAngle	intmotorNumber	double angle	booleanclockDirection
Changes the angle made by the arm joint with the horizontal. “motorNumber” is the motor of the corresponding arm joint (1,2 or 3). “angle” is the value by which the angle will change (default 1 degree). “clockDirection” is the direction in which the angle will change: 0 → clockwise, 1 → anti-clockwise.			
rotateArm	Boolean dir	double angle	
Rotates the arm in the direction specified by “dir” (0 → clockwise, 1 → anti-clockwise). Default angle increment is 10 degrees.			
toggleGripper	Boolean gripperState		
Attempts to open or close the gripper depending on what the last value of gripperState was.			
lockPosition	Position p		
Sets the current set of parameter values as the default position.			
addPosition	Position p		
Sets the current position values as a preset. 4 presets are available and are overwritten starting from preset 0 when capacity is reached.			

[public class DroidDriveControls]

Method	Parameters
moveForward	double 2_pi_r
Moves forward by ‘d’ units where ‘d’ is the circumference of the wheels.	
moveBackward	double 2_pi_r
Moves backwards by ‘d’ units where ‘d’ is the circumference of the wheels.	
turnRight	
Turns the robot right by 15 degrees around the center of rotation.	
turnLeft	
Turns the robot left by 15 degrees around the center of rotation.	
isBlocked	
Returns true if the sensors detect an obstacle 1 feet in front of the robot	

4.4.2 Microcontroller Code

Microcontroller code will be written in the form of sketches in the Arduino IDE. We plan on erasing the bootloader installed on the microcontroller that comes with the Arduino Mega 2560 and program it with an Atmel AVRISP mkII In-System Programmer (ATAVRISP2) via the ISP interface through Atmel Studio.

We will use the WiFlyHQWiFly RN-XV Arduino library to setup communications with the Wi-Fi module. Some relevant functions are noted in table 22:

Function	Return Type	Description
getSSID(char *buf, int size)	char*	Fetches the SSID of established Wi-Fi connection
getIP(char *buf, int size)	char*	Fetches the local IP address of the module
setIP(const char *buf)	boolean	Can be used to resolve conflicting IP addressing
setPassphrase(const char *buf)	boolean	Sets the WPA passphrase for the module, essential for security
join(const char *ssid, uint16_t timeout=20000)	boolean	Joins the specified wireless network

Table 22 -WiFlyHQWiFly Library

Our temperature sensor is quite popular among robotics enthusiasts and a popular library to use with it is the Dallas Temperature Control Library (GNU Lesser General Public License). Some relevant functions from this library are noted in table 23 that follows:

Function	Return Type	Description
setResolution(uint8_t);	void	Set global device resolution between 9 and 12 bits
getTempF(const uint8_t*);	float	Returns measured temperature in degrees Fahrenheit

Table 23 - Dallas Temperature Control Library

In addition, we will be using the Atmel Software Framework which contains drivers and open source code to implement motor controllers, PIDs and sensors. Atmel has recently also launched **Atmel Gallery** which is their equivalent of an “app store”. Developers can download real-time operating systems, communication stacks, plugins and extensions from within the Atmel Studio IDE.

In addition to this, users now have the option of using Atmel’s cloud based collaborative workspace – **Atmel Spaces**. It provides some essential features like access control, version control, and bug and feature trackers via an IDE extension. We plan on using Atmel Spaces to host our microcontroller code projects because we will have two members working on it. It also offers major advantages over using a service like Google Drive or Dropbox.

4.4.3 PC HUB Application

The PC application will be written using Java in Eclipse IDE. The reason for this is that it would interface well with the Android application which is also written in Java. Besides, Java is cross platform which means every group member can contribute. Java’s documentation is also one of the most detailed and beginner friendly out there. Figure 32 below shows how the PC application has turned out:

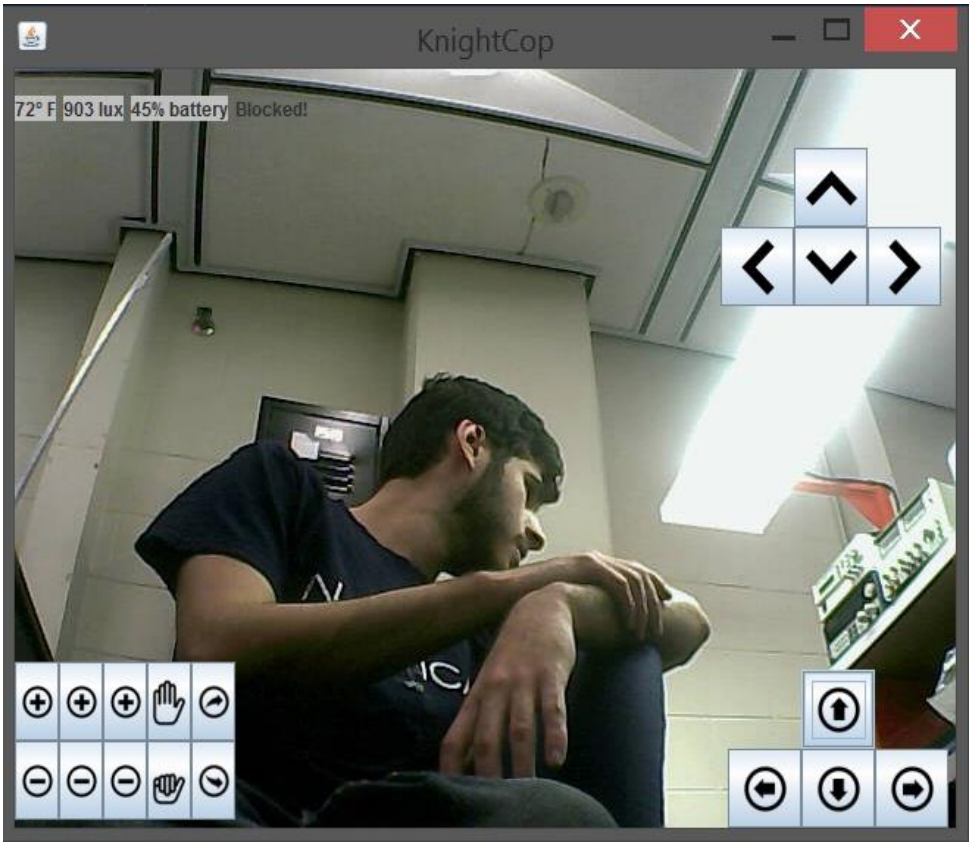


Figure 32 - PC Application UI

The background of the PC application will show live video feedback from KnightCop. Drive controls are in the bottom right corner, pan/tilt controls in the top right corner, sensor display in the top left corner and finally, arm controls in the bottom left corner.

The controls for the arm and drivetrain would remain the same and correspond to the controls in previous pages. Since the PC application would be able to accommodate more (due to larger screen size and resolution) we plan on adding more features to it. There is also an approximation of the time left on KnightCop's battery. All of this would be coded in Java.

The PC application will be developed as a Socket server. The Socket class in Java allows peer to peer communication between two nodes (devices). This was the main reason for choosing Java to develop the desktop app. A selective summary of the Socket class is given below:

[public class Socket]

Method	**Parameters**
bind	SocketAddressbindpoint
Sets the local address of the specified socket end point	
connect	SocketAddressbindpoint
Attempts to connect the specified socket to the established server	
getOutputStream	
Returns the output stream for [this] socket	
getInputStream	
Returns the input stream for [this] socket	

The GUI for the application will be developed using Java Foundation Classes (JFC) which encompasses the Swing API. Swing allows users to create user interface elements and arrange them on the GUI using frames and layout managers. We intend on implementing the following classes for the desktop application:

[public class Battery]

Method	**Parameters**
setDuration	int interval
After a thorough testing phase, we should have an approximate idea of how long the battery will last. This maximum value will be set as a starting point and the value will be decayed every interval (default 1 minute).	
getRemaining	
This will return the battery life remaining (in number of minutes). This will be done by subtracting the [intstartTimer] field from [int duration] field.	
isCharging	
This will return a boolean value depending on whether or not the battery is being charged by an external power source.	

[public class TempSensor]

Method	Parameters
getTemperature	
Attempts to get the output from the temperature sensor on board KnightCop	
toCelsius	double temp
Converts the specified degree Fahrenheit temperature to degree Celsius	
toFahrenheit	double temp
Converts the specified degree Celsius temperature to degree Fahrenheit	

[public class ProximitySensor]

Method	Parameters
pollSensor	int direction
Poll the sensor facing given direction: 0=N, 1=S, 2=E, 3=W	
increaseSpan	int degrees
Increment the field of view for [this] sensor by given degrees	
decreaseSpan	
Decrement the field of view for [this] sensor by given degrees	

[public class Navigation]

Method	Parameters
pauseForRead	
Halt KnightCop while sensors realign in target direction	
suppressSensor	int index
Discard readings from specified sensor	
activateSensor	int index
Allow readings from specified sensor	
setIntent	
Sets current direction as optimal path	
checkIntent	
Signifies if KnightCop is inclined to move in direction it currently faces	

4.4.4 Obstacle Avoidance

Our obstacle avoidance algorithm will work in has both an active and a passive way of avoiding collisions. The passive way will be by keeping the user informed of how far are objects from the robot at all times. This way the user can make an informed decision based on his line of site view to the robot, the video feedback being sent back by the video camera, and lastly the numbers depicting how close things are to the robot. This will be done by encoding the sensor information in the packages that are sent into the laptop from the microcontroller, decoding them in the laptop and displaying them in the graphical user interface and optionally in the android application if desired. There will also be messages and alerts being displayed when objects are close or getting increasingly closer. This will be done by keeping in the memory of the laptop a recollection of the most resent sensor data and user input. If the user is an object is detected to be

getting closer at a speed faster than expected by the user input an alerting light will be lit so that the user knows that something is approaching KnightCop. Additionally they will be able to see the sensor data changing so they can figure out how fast it is approaching.

The active way that we will avoid collisions is by giving the option to the user to activate evasive maneuvers. This will work by both avoiding increasingly approaching threat that are caused by KnightCop driving towards them and by them approaching KnightCop. There is two parts in determining is something is approaching the robot instead of the other way around. The first is to determine what direction the robot is moving. The second thing is determining if there is something approaching base on the first factor combined with the sensor data. We will use the recent history of the signals sent to the motors by the microcontroller which are initially determined on the laptop. We will average the values for each side individually giving more priority to the most recent values by weighting them. If we are above a certain threshold to be determined through testing, then we can say with a certain degree of accuracy that the motors on each side are either moving clockwise, counterclockwise, or really close to being stopped. Once we have this information we will be able to determine if we are going forward, stopped, backwards or rotating one way or the other. In other to keep complications down and add reliability to the system the thread avoidance algorithm will not try to actively avoid threads when rotating. Knowing if we are moving and at what orientation will then let us do a similar calculation as to whether we detect an object approaching on the wrong direction or not. Then the robot will simply try to drive autonomously, as long as the user doesn't override it, in the opposite direction. The speed at which it will try to get away from the thread is dependent on its approaching speed. Avoiding obstacles that the robot is approaching as we will simply try to go in the general direction if possible but if there are further obstacles that don't allow for evasive movement it will simply stop and alert the user that the path is blocked. The speed of the robot will also be reduced as obstacles are approaches to allow for more time to correct the path and to stop on time in order to avoid a collision.

4.5.0 Manipulator

Since we are using a robotic arm kit for the manipulator, we know exactly the minimum amount of weight that it can carry. It is rated at a hundred grams of lifting capacity but we found reviews online with other projects being able to do ten times as much. This was done by increasing the voltage of the motors provided from the original six volts that its battery pack provided to nine volts. This increased the torque as the motors are actually rated to work more efficiently on that voltage level. Additionally, changing the motor that controls the gripper for a stronger servo that provides more torque increased its strength. It is evident that the materials chosen to build the manipulator are strong enough to lift weights that are significantly higher than those that the kit is rated for. The motors and gears used to power the movement of the manipulator are the limiting factor to how much weight it can lift. That means that if desired,

modifications to all of the motors to stronger geared motors or stronger servo motors can be made to increase its lifting capacity. We do not plan to do this as we are not worried about how much weight we can lift. That is more of a mechanical constraint that as an electrical team we did not see fit to put as a top priority. Additionally, its size and weight were an advantage as we could mount it on top of the mobile base with ease (Figure 33). We decided to concentrate on providing a good interface with the user to control the multiple degrees of freedom of the manipulator with a high degree of precision and the addition of autonomous features. These features are comprised of precision control via a close loop control algorithm that will be used to control the manipulators movement taken into consideration the user's input and the sensed position of each of the joints that is taken from the mounted sensors. Furthermore, the ability to have preset positions for the arm gives it an autonomous capability. These positions can be left as the original pre-programmed positions or saved on the fly by moving the manipulator to a desired position and then saving that position over one of the original profiles. These allow the user to initiate a sequence of movements that the manipulator can then continuously perform. This gives the user the ability to perform a repetitive task by simply setting the positions once.

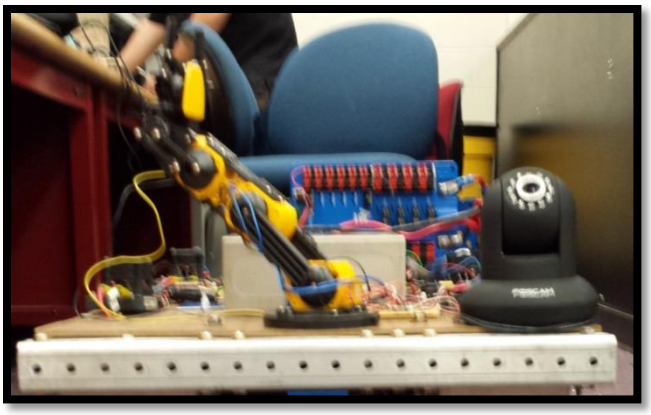


Figure 33 – Manipulator on Mobile Base

Even though we are currently not planning on making the modifications needed in order to increase the lifting capacity, we still wanted to keep that option as a possible upgrade if we had the time and budget space needed to do it. That was another reason why we picked the DRV8833PWPR motor controller. Its versatility means that it can handle the increase in voltage and current that we would need for the upgrade. Moreover, its energy saving feature called nSleep allows us to save very valuable battery life at the increased current draw. The wiring for the controller (Figure 34) is very simple. The motor to the connection is straight to its positive and negative leads. If we find that the orientation of rotation is backwards of that we are expecting then we can simply choose to swap the wires going into the leads or invert the values in the software. It is preferable to swap the wires going into the motor leads as inverting the values in the software would not be intuitive and might lead to confusion. The nSleep feature is activated through a single digital pin, which will be held at a low state when the

energy saving state is desired. Since we are not worried about current control to the motors, the xISEN pins are connect straight to ground without a resistor in series. The only additional components that are needed are three different capacitors that are connected in series with the voltage and ground inputs of the device. In order to control the motors efficiently we chose to use the PWM control method option provided by the controller. The reasoning for that was so that we can implement a closed loop control algorithm referred as a PID loop. PID stands for proportional, integral, and derivative. This are the three terms used to calculate the error between a measure variable and a desired set point. This calculated error will then be injected into the calculation of the value to set the motor. For the purposes of our application the use of the proportional term is enough. The reason for this is that the proportional value can be seen as the current error, the integral value as the accumulation of past errors, and the derivative value the error predicted on the future. In essence, we are controlling the motors to use more power and thus go faster when the current position of the joint is farther away from the desired position. This means that as the manipulator is getting closer to the desired position it will decrease its rate of movement, giving a smoother transition between positions than other control algorithms like bang bang controllers or the use of fuzzy logic.

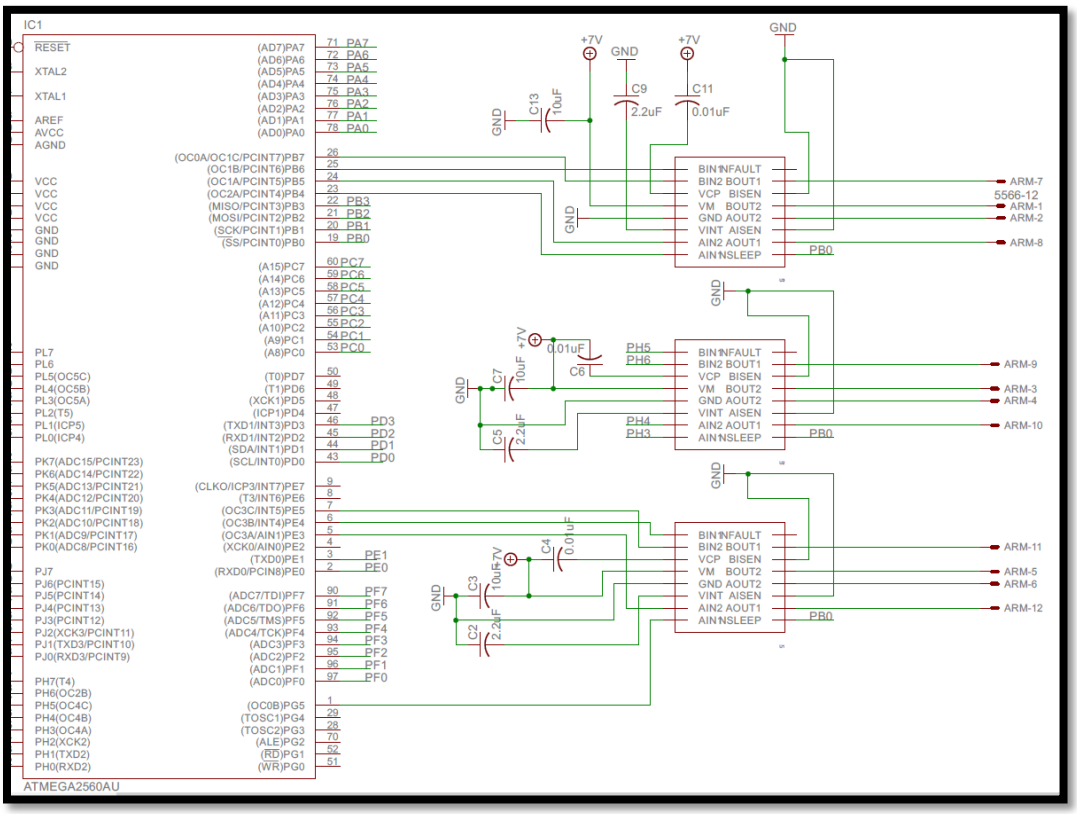


Figure 34 – Manipulator Motor Controllers Wiring Diagram

4.6.0 Environment Sensors:

4.6.1 Temperature Sensor: Maxim Integrated DS18B20

The temperature sensor is a 3 pin through-hole component that provides digital output and utilizes only one digital line for communication with the microcontroller. The pin-out diagram of the sensor is given in figure 36 below:

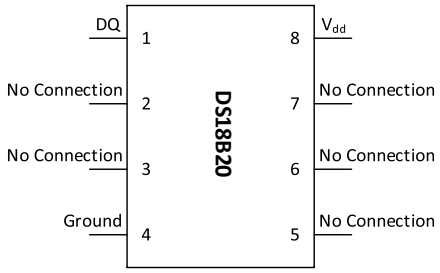


Figure 35 - Pin Out Diagram of Maxim DS18B20

Pin 8 (V_{DD}) is connected to an external line transferring 3.0 to 5.5 V (regulated). Pin 4 is ground. Pin 1 (DQ) is the digital input/output line.

Data can be read in a 16 bit format, where the user can specify 9 – 12 bits for temperature reading. Remaining bits are used as sign bits. The sensor output is linearly proportional to changes in temperature. 0° C is 0x000. The number increases to 0x07D0 up to 125° C and falls to 0xFC90h for -55° C. The relationship can be seen in figure 37 below. Notice that data points on the X Axis have been converted to decimal from hex since our version of MS Excel does not natively process hex values.

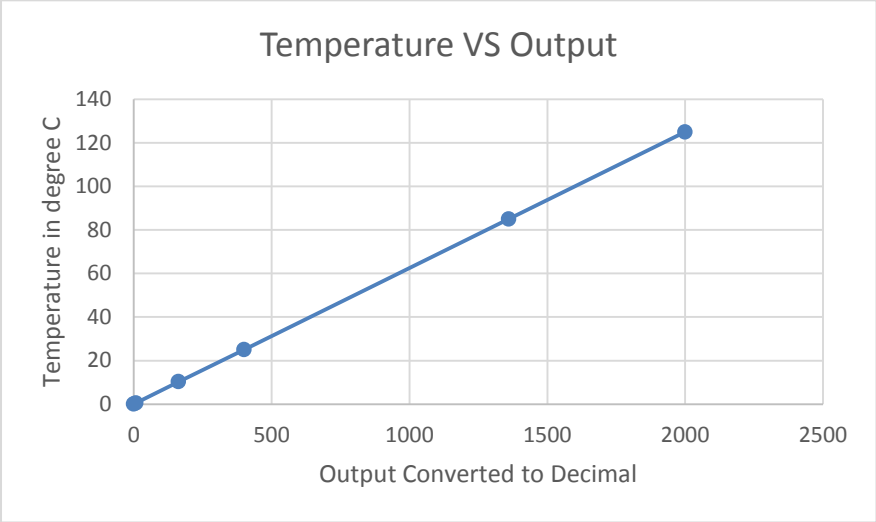


Figure 36 Temperature VS Output Value

The C library we intend to use with this sensor takes care of data exchange between the sensor and the microcontroller. It initializes the DS18B20, sends the ROM commands in sequence and follows up with a Function command to actually fetch data. Temperature conversions are also possible onboard, but we plan on implementing this functionality in the end-user application.

The sensor can also be used in parasite power mode using only one wire, but it complicates the data fetch process with no significant savings in power, so we ended up not using it. The highlight of using these particular sensors is the One Wire Bus technology that allows a user to chain multiple temperature sensors on one input line, thereby saving space and pins on the PCB. Any device can be polled on the bus wire for its reading without affecting input from other one wire sensors.

4.6.2 Ambient Light Sensor: SEN-09088 Photo Cell

This is a simple photo resistor. The output current is linearly proportional to the Illuminance (E_V, total luminous flux) on the device. In essence, output current is strong if area surrounding the sensor is brightly lit. The device was mounted in an exposed area on KnightCop to allow sufficient incidence of light. Output voltage is directly proportional to illuminance:

0V = 0 lux (dark)
2.5V = 1000 lux (typical indoor lighting)
5V = 10000 lux (daylight on a sunny day)

Figure 37 - SEN-09088 Photo Cell

The photo-resistor must be connected to a supply voltage of up to 150 V. Thorough testing determined the threshold for activating the light onboard KnightCop and for activating ‘night mode’ on the IP camera. The analog readings can be taken fast enough so that the delay between switching modes on the camera is minimum.

5.0.0 Design Summary

5.1.0 Hardware

As it can be seen in the schematic (figure 38) a single power connector is used to provide 12 volts in and then regulated to the 3.3, 5, and 7 volts needed for the different systems in KnightCop. On the right we can see the different motor controllers used to control the arm control connected to the microcontroller and the capacitors needed. We used a 16 Hz resonator instead of a crystal because of their ease of use since it comes with the 2 capacitors incorporated into it. The wifi module is connected to the microcontroller via the one of its serial receiving and transmitting line. There are several connectors that allow connecting all of our peripherals and other motor controller to the mcu. As it can be seen on the pcb layout (figure 39) there are extra connectors to allow for scalability that would allow us to add more sensors with access to more digital, analog, serial and power and ground lines.

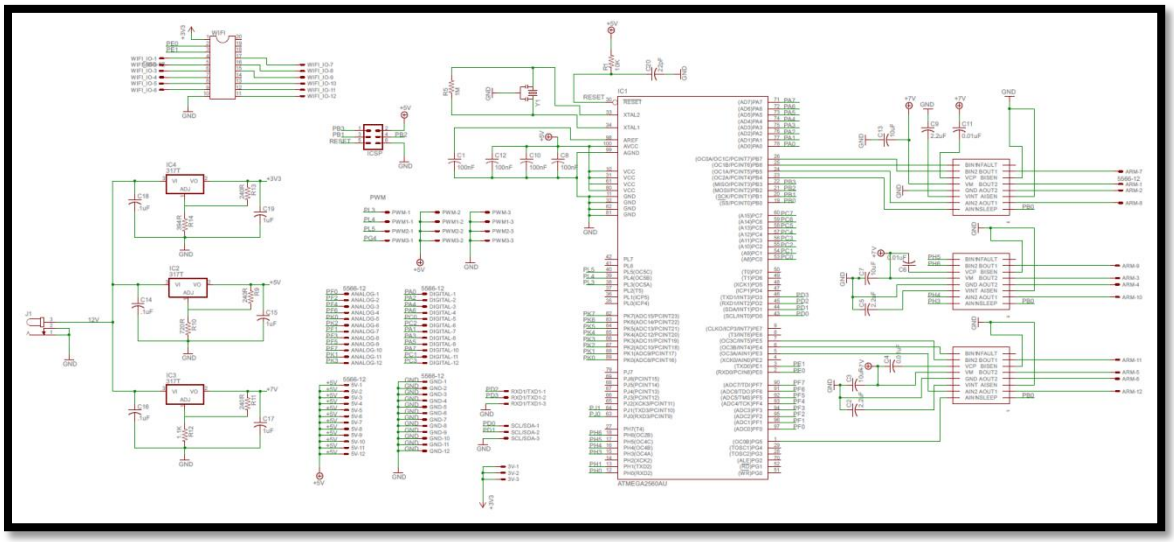


Figure 38 - Schematic

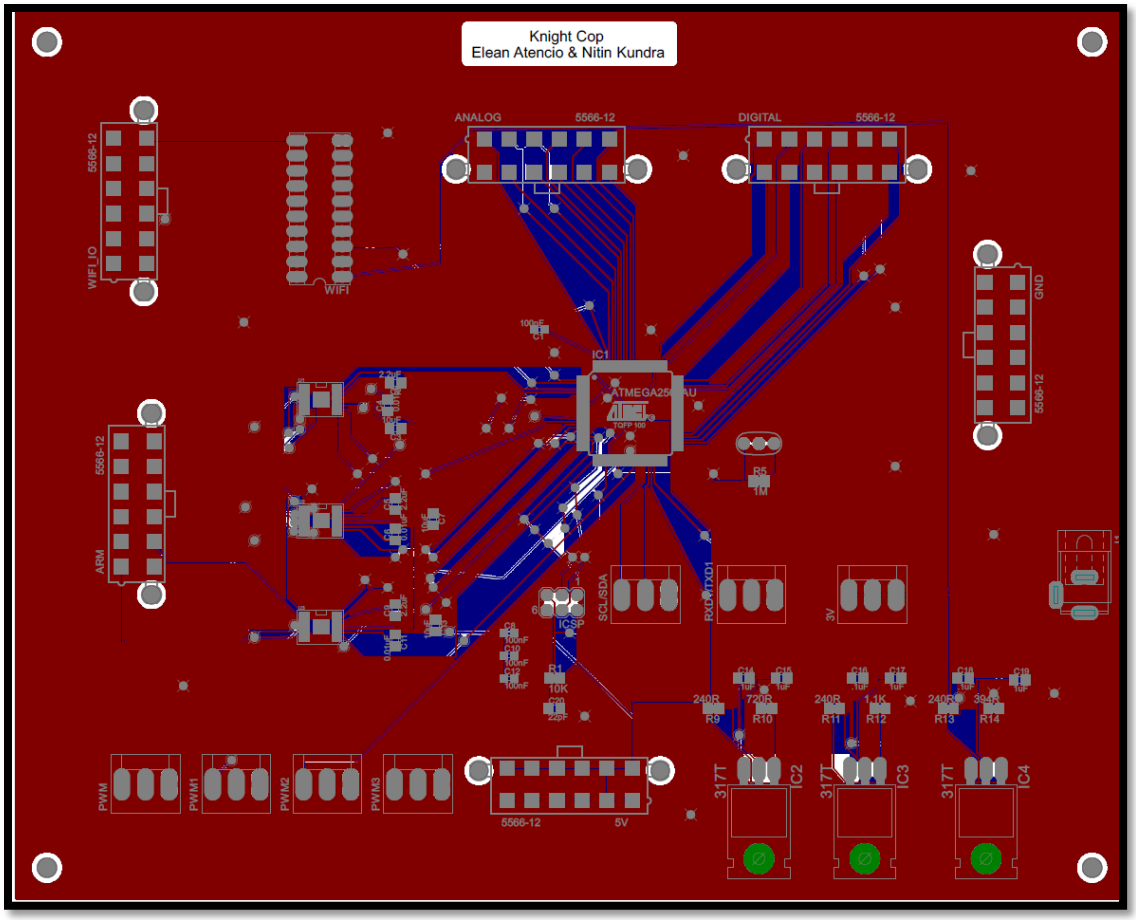


Figure 39 - Layout

Figures 40 and 41 show the top and front view of the robot. It shows the robustness of the mechanical design and how power and signals are distributed to through the robot. It also shows the mounting of all of our major components where the battery is the middle and the manipulator next to the camera up in the front. The pcb is mounted in the back of the robot just under the power distribution board for the 12 volt systems. Figure 41 shows the front instance of how the ultrasonic sensors are mounted on the underside of the base of the robot to allow for detection of objects that are closer to the ground.

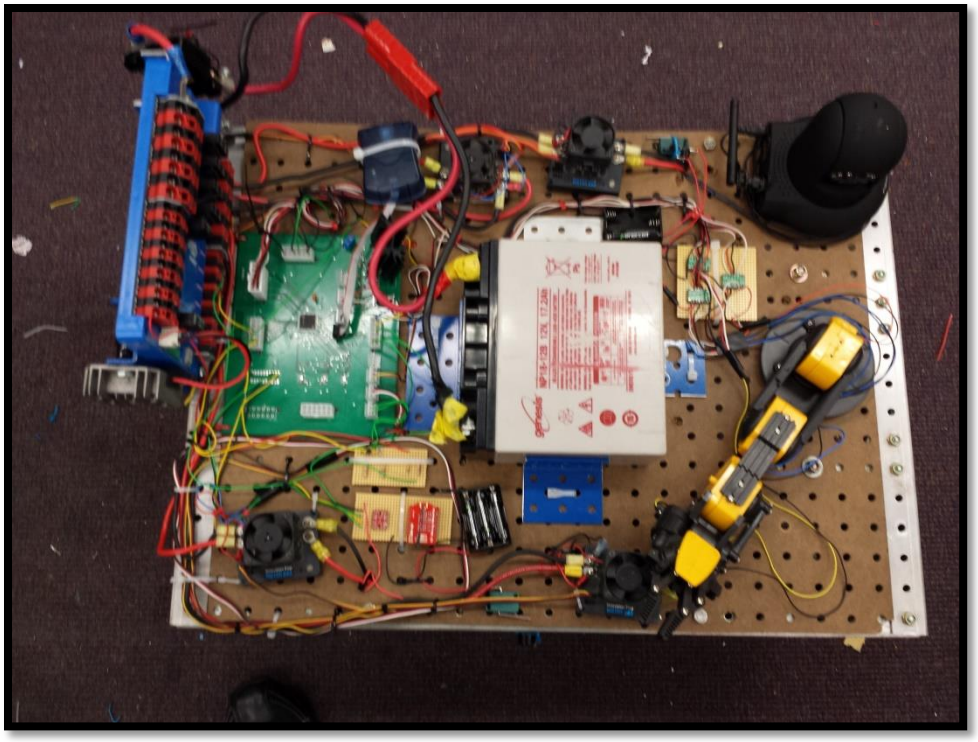


Figure 40- Robot Top View

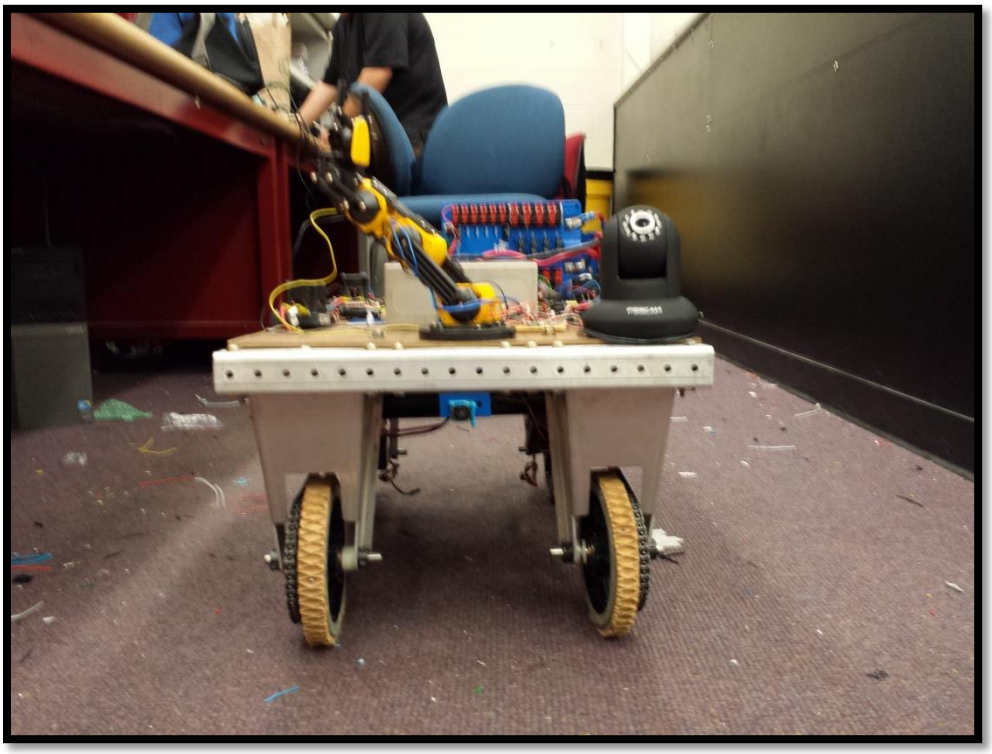


Figure 41 – Robot Front View

5.2.0 Software

Microcontroller Code Overview:

The microcontroller is controlled by a bootloader that directs control to the main function on startup. Thereafter, the main function would loop continuously until the microcontroller is put to sleep or turned off. It would poll the RN-XVee Wi-Fi module for received commands and analyze each one. Each command would have a prefix denoting which component it pertains to. The prefixes would range from 0x00 through 0x07, as described in table 24 below:

Hex Prefix	Component
0x00	drivetrain
0x01	arm control
0x02	Wi-Fi module
0x03	potentiometers
0x04	ultrasonic sensors
0x05	light sensor
0x06	temperature sensor
0x07	ATmega2560

Table 24 - Prefixes for Incoming Commands (ATmega2560)

The setup() loop in Arduino code declares pins as outputs and inputs as necessary and initializes them with safe or dormant values. For example, drive motors need to be given a value of 185 to make them stop. Deviation from 185 on either side initiates motor rotation.

The loop() function repeats until KnightCop is powered down and is responsible for fetching control commands from the controlling application. This is done via Serial Port UART0 which interfaces with the Wi-Fi module. Any data written or requested over UART0 in Arduino code is sent/fetched from the Wi-Fi module. The module, for all intents and purposes, replaces the Serial Monitor shipped with the Arduino IDE.

Iterations of this code are interrupted by commands from the controlling application, which means that KnightCop cannot simultaneously execute a turn and move command. However, owing to the high cycle frequency of the Atmege2560, this delay is not noticeable to the user. Polling sensors and relaying sensor data was also instantaneous over short to medium range. This, in effect, made the ultrasonic sensors very responsive and KnightCop was able to detect obstacles blocking its path in rapid succession.

Since our PCB design is derived from the Arduino Mega schematics, we were able to write and test sketches on the Arduino IDE. Every time this code is compiled, the IDE dumps hex data in a temporary directory. This hex file was used to program the ATmega2560 on our PCB using MKII serial programmer.

The main function is also be responsible for sequencing the flow of control between the components. Some probable scenarios are depicted in figure 39 and figure 40 below:

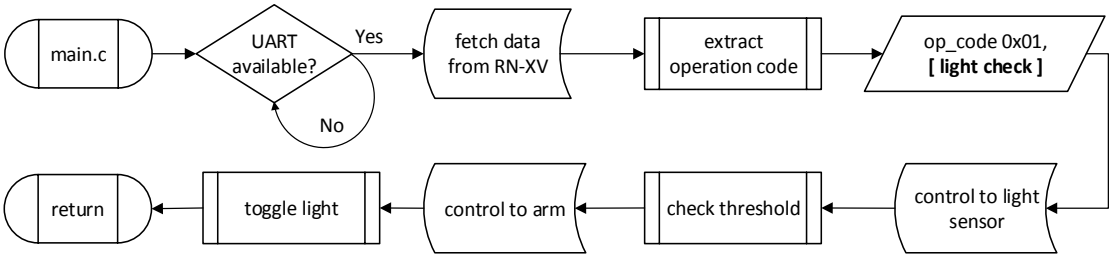


Figure 42 - Flow of Control [light check]

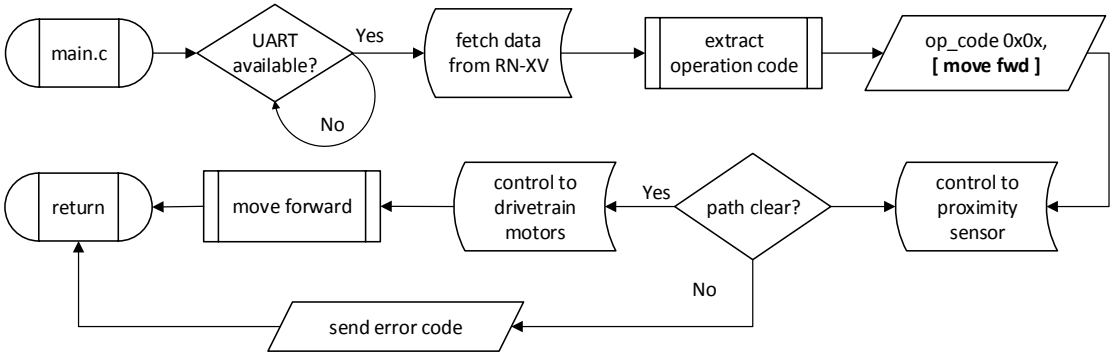


Figure 43 - Flow of Control [Attempt to move forward]

Due to our modular design approach, most decisions KnightCop makes are chained linearly, instead of forming complicated branches. This improves response time and lends to intuitive and frustration free control on the user's part. The user does not have to wait for KnightCop to finish executing requests. Some lag is noticeable over long range (> 100 feet) which is a limitation of Wi-Fi radio waves and not of the code. The microcontroller could not be saturated with processing requests in any of our tests.

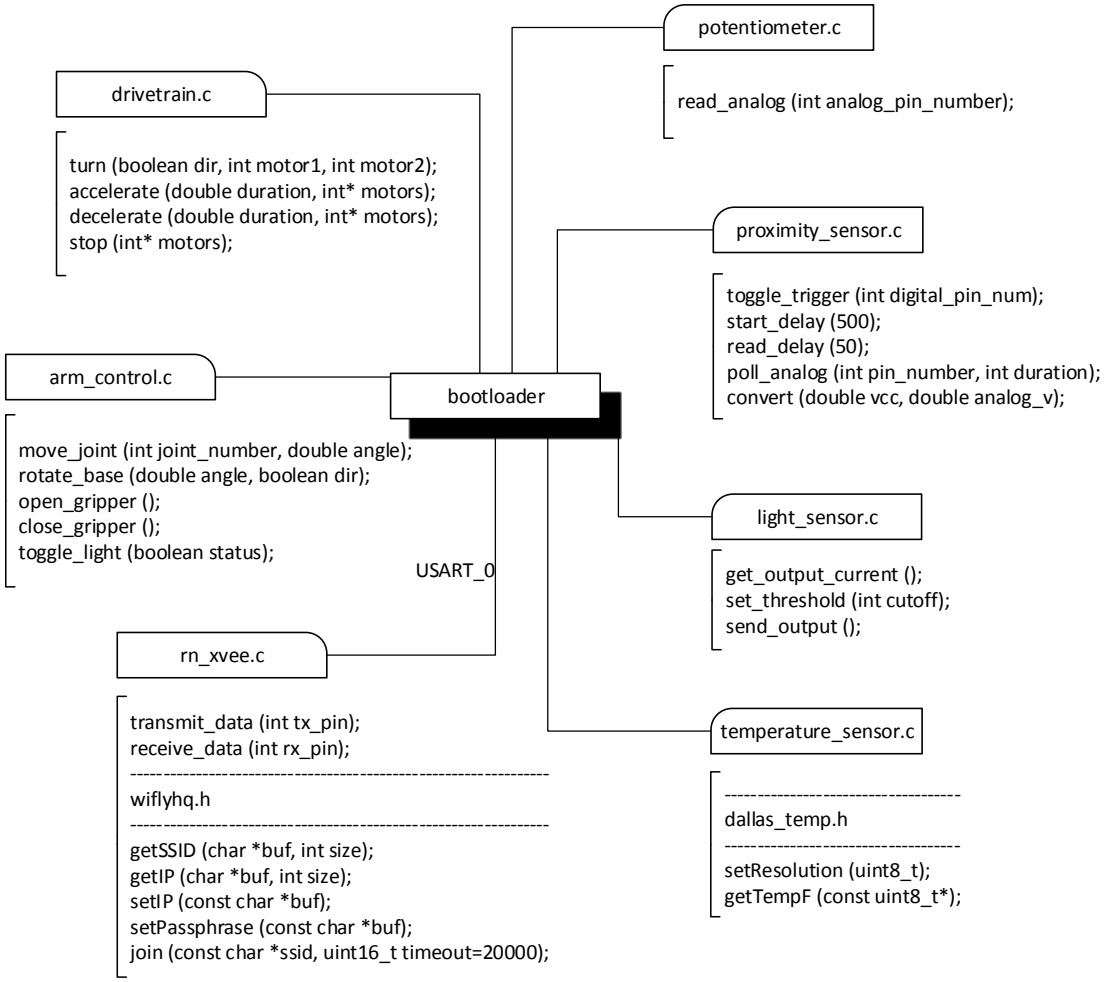


Figure 44 - Microcontroller Code Structure

PC Code Overview:

The PC Java application is primarily responsible for relaying user control to KnightCop. It would receive data from the user via the GUI and would forward requests to KnightCop via the Wi-Fi module RN-XVee. The PC is also running an independent thread responsible for fetching and rendering the MJPEG stream from the IP camera.

At any given time, there may be two to three threads running in the PC application - the GUI rendering thread, the video stream thread and an on-call thread to control camera actions and parameters. This structure ensures that data streams are independent and do not interfere with each other. Figure 42 shows the class diagram for the PC application.

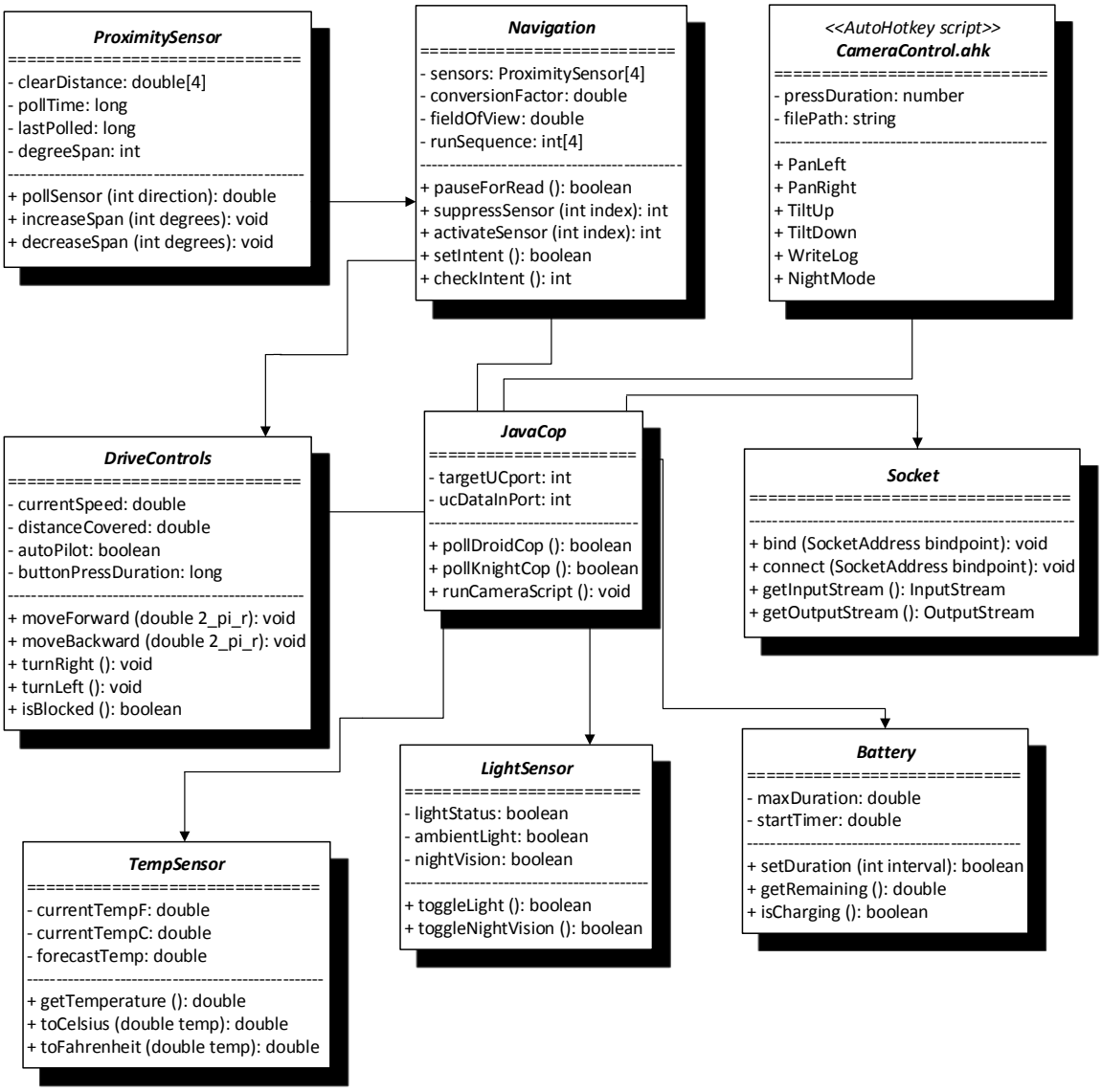


Figure 45 - PC Application Class Diagram

Android Phone Application Overview:

The application on the Android phone (HTC One V) is the primary way for the user to interact with the robot. It would receive live video feed from the IP camera. It would also send and receive control data via the Java Sockets. Programming for the HTC One V will be done using Android Development Tools, which is essentially a customized version of the Eclipse IDE. The PC application and the Android app will share a few classes and data structures as shown in figure 43 below:

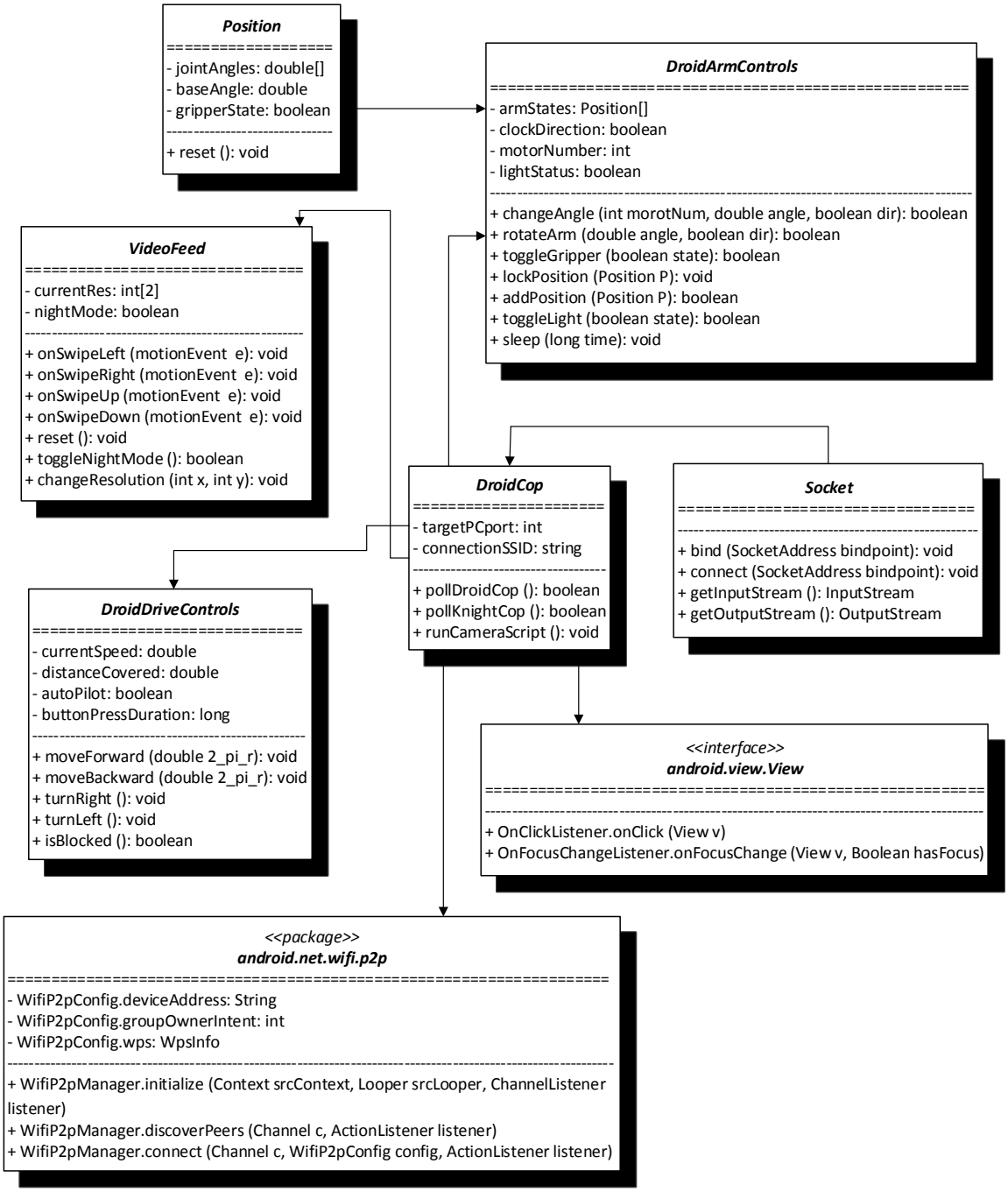


Figure 46 - Android Application Class Diagram

6.0.0 Project Prototype Construction

Project design and construction continued into the Fall 2013 senester. In order to do this we had to take two main things into account. The first was the locations at which the building and testing of our prototypes was going to take place. The

locations were going to be dependent as to the type of work we would be conducting as different tools would be necessary. The first task that we considered was the construction of the chassis and major mechanical components. We need specialized tools, like a band saw, to deal with the metals that we are going to be using. Thankfully Elean has access to these tools in his residence and that is where we will be doing the majority of the hardware build. The assembly of the electrical system needs a different kind of tool set that is required. We need oscilloscopes, multimeters, soldering irons, and other tools used the construction of electrical systems. Elean has access to some of these tools at home but not all of them. We decided to do most of this construction at the senior design laboratory for electrical and computer engineering students that is located in the engineering building at UCF's main campus. This laboratory contains all of the tools that will be needed in order to build our electrical systems. Additionally it is a very convenient location as we spend a good amount of time on campus as it is. Furthermore, since other teams will be doing a lot of their work in the same lab we will get the opportunity to collaborate with them if we have any questions. This will be really helpful, especially when going through the process of testing the system as other teams might have already troubleshooted what we are having difficulties with at the moment. Testing of the mobility of the system will be done outside and on different terrains.

The next step was to choose where we will be getting the parts that we had picked to use in our design. Table 25, depicts the list of parts and the vendors we will be using to get them. When picking the vendors we had three things in mind, the cost, timely delivery and availability of products. We wanted to make sure we picked vendors that had a surplus of availability of the products we were getting. The reason for this was that we were not getting these components until next semester and wanted to warrantee as much as we can that they will not be sold out when it was time for use to buy them. The majority of the vendors we looked at were in through their online interface where they have listed the availability for each of their products. Lead time in the shipping and receipt of the orders was also very important. We had to make sure that after an order had been place, we would get all of the components in a timely manner so that we could incorporate them in our prototype with enough time to test and decide if we need to order a replacement or different part if it doesn't work. The cost was one of the most important things as with a budget of close to a thousand dollars and only one sponsor that is only paying for a limited amount of the budget, we wanted to limit the amount of money each of the members of the team was going to have to pay out of pocket. The cost was also influenced by the price of shipping the components and so we decided to try to buy as many components from a single vendor as possible. A very important component for the incorporation of all of our electrical systems is going to be the printed circuit boards (PCB). We considered two vendors for our PCBs, ExpressPCB and PCBFabExpress. PCBfabExpress doesn't offer free PCB layout software. Additionally, we discovered that their pricing was really high as compared to ExpressPCB. They do offer an assembly service but that would defeat the purpose of using PCBs for our senior design project as it is a skill that every electrical engineer should learn. ExpressPCB on

the other hand, offers a free CAD software for PCB layout. Furthermore they also offer low cost but good circuit boards called their MiniBoard service. This service provides three identical two point five by three point eight inch boards for the low price of fifty one dollars plus taxes. That allows us to design our board with different options included in one board and have the effect of having three boards in it.

Part (Part #)	Vendor	Quantity
Arduino Mega 2560 (DEV-11061)	Sparkfun.com	1
Ultrasonic Maxbotix LV-EZ0 (SEN-085052)	Sparkfun.com	4
Potentiometer (COM-09939)	Sparkfun.com	1
Trimpot (COM-09806)	Sparkfun.com	4
Microswitch (COM-09414)	Sparkfun.com	2
Digital Temperature Sensor (SEN-00245)	Sparkfun.com	1
CIM Motors (am-0255)	Andymark.com	4
ThougboxNano (am-0482)	Andymark.com	4
PlactionRoughtop Wheel (am-0437)	Andymark.com	4
375 Hex Hub (am-2231)	Andymark.com	4
OWI Robotic Arm Edge (OWI-535)	Amazon.com	1
DRV8833PWPR (64T2941)	Newark.com	3
Miscellaneous Resistors, Capacitors, Wires and Other Electrical Supplies	Newark.com	NA
RN171XVW-I/RM (765-RN-XV)	Mouser.com	1
Ambient Light Sensor (GA1A2S100LY)	Mouser.com	1
Wireless Camera (FI8910W)	Foscam.us	1
ATAVRISP2 (ATAVRISP2)	Atmel.com	1
MiniBoards	ExpressPCB.com	3
Victor 888 (217-2769)	Vexrobotics.com	4
72” Aluminum Angle (217-0425)	Vexrobotics.com	3
Miscellaneous Sensor Brackets	Vexrobotics.com	NA

Table 25 – Part Acquisition table

7.0.0 Testing

7.1.0 Communication Range:

We shall be using Wi-Fi class G networks in all our communication needs. An ad-hoc, WPA encrypted network will act as the communication channel between KnightCop, the PC Hub and the Android phone. The network will be initiated on the PC, since the wireless network card on the PC (Intel® Centrino® Wireless-N 2230) has the highest output power out of all three. This means that it’s area of influence is the largest and it can support maximum span of distance between the other two devices.

Outdoor Testing:

Wi-Fi has a maximum range of 35 m (120 ft) indoors and 100 m (300 ft) outdoors. We plan on testing the robot around Harris Corporation Engineering Center. The outdoor test will be 3 phased:

I. Record maximum distance the robot can travel from the base station (PC) placed at the main entrance without losing video feed or control - Average 100 feet.

II. Record maximum distance the Android phone can be used from the base station without losing communications - Average 100 feet.

III. Attempt to control the robot (placed in the courtyard) while the PC is inside HEC, from the phone at the Arboretum behind HEC. This helps us foresee if the robot would function across several layers of walls in a building where Wi-Fi channels are heavily loaded - Passed.

The areas spanning the tests are depicted in figure 44 below:

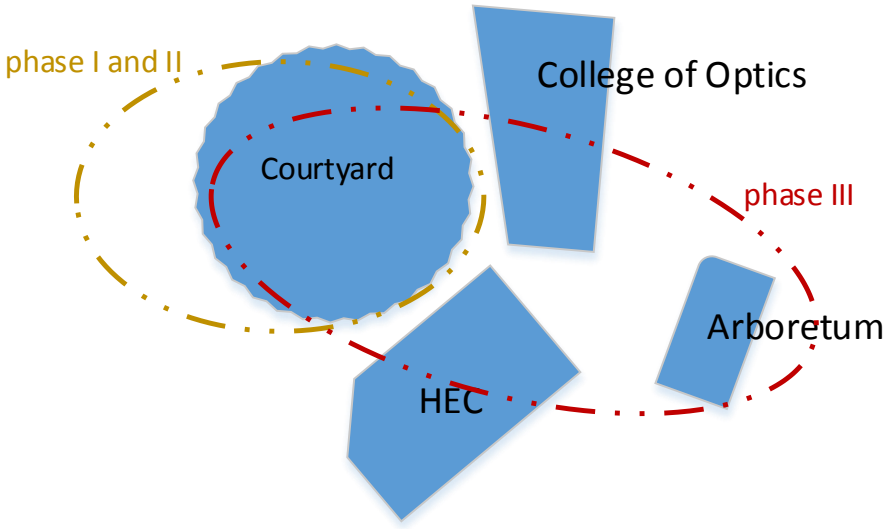


Figure 47 - Outdoor Communication Tests

Indoor Testing:

The indoor tests will be held in the corridors inside HEC at level 1. This gives us a chance to replicate a common urban emergency situation where the police personnel wish to assess the situation inside a building before entering it. The indoor tests will be two phased:

I. Base station and Android phone outside HEC, KnightCop used to navigate level 1 corridors and empty classrooms - Average 85 feet.

II. All three devices inside the building. Test maximum range attainable again - Average 85 feet.

Conclusions:

We anticipated that our test would be close to the indoor (120 ft) and outdoor ranges (300 ft) claimed by the Wi-Fi transceivers. Major discrepancies arose because the default communication channel was overloaded with other wireless and radio devices. Our testing away from the sources of noise showed better results, however those were dismissed because KnightCop is intended to work in an urban environment where such devices are in abundance and there is no escaping radio noise and congestion.

7.2.0 Maneuverability

In testing the maneuverability of KnightCop we were conserved with its ability to traverse several types of terrains while maintaining a good amount of speed and overcoming small obstacles that might get in its way. In order to test this ability we will place KnightCop in a grass field, parking lot, basketball court, and an area full of rocks. In order for the test to be a success the robot cannot get stuck and shall be able to traverse the terrains freely. Additionally, its top speed will be tested by measuring the amount of time it takes it to traverse a fourteen foot span after reaching its top speed. The reason for that number is that our calculated top speed is going to be about seven feet per second in concrete. This way we are expecting for the robot to be able to traverse the fourteen foot span in about two seconds. Lastly its ability to handle certain obstacles needs to be tested. First a two by one inch piece of wood will be held down in the path of KnightCop. It should be able to successfully drive up and over this obstacle at very low speeds. We can extrapolate from this that it would be able to also overcome this type of obstacles at higher speeds even though it is not recommended as the impact might cause damage to the wheels.

KnightCop should also be able to push things out of the way if necessary since we picked motors that were able to handle as much static friction as a worst case scenario of the robot can handle. We will put objects of increasing weight in front of the robot and record the point that the wheels start spinning and the robot is not moving. The contact surface between the objects and the ground should be the same as we want to keep the coefficient of friction to be the same in order to get comparable results with the different weights.

7.3.0 Manipulator

7.3.0 Degrees Of Freedom

One of the key features of the manipulator is that it has five degrees of freedom that the user can control while lifting a hundred grams of weight. In order to test this each joint will be commanded to go to its extreme values and compared against our rated rotational freedom (Table 26). It must also be able to this while

holding at least a hundred grams of weight. Additionally, the manipulator must do this safely. This will be tested by commanding the joints to keep moving in the same direction after its extreme value is reached and the motors shall remain motionless. Likewise, the gripper will be commanded to open and close all the way and must stop whenever the extremes are reached or objects of variable size are grabbed.

Joint	Rated rotational Freedom	Measured Rotational freedom
Wrist	120^0	120^0
elbow	300^0	300^0
Base Rotation	270^0	270^0
Base Motion	100^0	100^0
Gripper	Open/Close	Passed

Table 26- Rotational Freedom Log

7.3.0 Autonomous Features

The manipulator's autonomous features are easily testable as we set the arm joints to go to different angles and manually verify that the angles are accurate. Additionally, we will make sure that the sequencing of the different preset positions is being followed correctly by measuring the angles that it steps through at each step first and then validating that the sequence is in the proper order. We will also override these positions through the user interface and verify that the new positions are saved and carried out in the correct order.

7.4.0 Battery Life

The robot was run under a multitude of circumstances until the batteries were depleted. The time that this takes was recorded and checked against our desired specifications. We were able to power KnightCop in excess of 1 hour on average, with all systems functional.

7.5.0 Sensors:

Temperature Sensor:

In testing the temperature sensor we want to be sure that accurate measurements are reported across its range (-55° C to +125° C). To this extent, the sensor would be securely mounted in an exposed area of the robot where heat dissipated from the electronics or power supply would not affect it. At the same time, it must not be exposed to direct sunlight which could cause the device itself to heat up and measure the wrong temperature. The testing would be three phased:

I. Obtain hourly weather forecast from the National Weather Service at www.weather.govfor our locality. Take hourly temperature readings out in the open. The two sets of readings should not differ drastically if weather is clear.

II. Take readings in temperature controlled environments (laboratories and offices) where ambient temperature can be verified from a thermostat.

III. Heat water in a metal container and measure the temperature with a kitchen thermometer. When placed close to the sensor, we should get a reading close to the thermometer's.

IV. Subjecting the sensor to extreme heat (say, from a cigarette lighter) to ensure it can measure the claimed maximum temperature.

The sensors are cheap enough that we can afford several and are easy enough to implement (through hole, 1 data pin) that destroying one would not set us back on schedule. Proper care will be taken to ensure that other electronics are not affected in this testing.

Ambient Light Sensor:

The ambient light sensor measures the Illuminance or the total luminous flux on a surface, per unit area. The particular sensor we chose has light sensitivity similar to that of the human eye. This makes verifying test results easier. Care must be taken to ensure that the sensor is evenly lit (it is comprised of several photodiodes) and that no stray light may distort its measurements.

The testing will be done in five phases:

I. **Take measurements in open sunlight at noon.** Output current should be close to maximum.

II. **Take measurements in an evenly lit room.** Output current should stay high.

III. **Take measurements in shadow and poor lighting.** Output current should be low or non-existent.

IV. **Take measurements in a dark room with the onboard light and the IP camera off.** Output current should be non-existent.

V. **Take measurements while entering a dark room from a well-lit entrance.** Output current should fall immediately.

We intend to not only test the accuracy of the sensor, but also it's responsiveness. If lights go off in a disaster zone, it should respond quickly or the user might miss something important, since the night-vision on the video feed is only activated automatically when this sensor conveys a lack of light. We predict the responsiveness to be lower than 200 ms, since the device itself can be reliably polled every 20 ms *[1] and wireless data transfer and acknowledgement over 100 feet would not take more than 100 ms *[2].

*[1] *from device datasheet at: http://www.mouser.com/ds/2/365/GA1A2S100LY-180860.pdf*

*[2] *lower threshold on G class Wi-Fi = 10 Mbps, 1 bit = 10 $^{-7}$s, 16 bits ~ 1.6 µs. Even factoring in noise and data processing by the software, 100 ms is a conservative estimate.*

7.6.0 Obstacle Navigation

The testing of the obstacle navigation algorithms will be carried out outside and with a safety first mentality. We will put bumpers on the outside of the frame of KnightCop in order to decrease the possibility of anyone getting injured or anything getting broken in case of a runaway robot. Additionally to this, a failsafe algorithm will be used in order to try to avoid this from happening. Furthermore, a power switch will be hooked up to a cable that one of the members will be holding. If the robot bypasses all other measures, the cable will be pulled and the robot will be disconnected from its power source.

The first things we will test are the readings and update times of the ultrasonic sensors. To do this we will stand on all four sides of the robot when it is stationary. We will start by standing at a distance of twenty feet and getting closer in increments of five feet. When we reach the five feet mark we will start getting closer increments of one foot. The displayed distances must match our actual distance with an error of five to ten percent maximum at all times.

The second test is going to involve setting up an obstacle course that we will try to traverse while driving at obstacles. The robot should change its heading automatically to avoid colliding with those obstacles when the algorithm is enabled. Additionally, the end should be an array of obstacles that the robot cannot circumnavigate. In this instance the robot should stop before any collision and alert the user that there are obstacles blocking its way. The user could then choose to disable the algorithm and push the obstacles out of the way.

8.0.0 Administrative Content

8.1.0 Budget and Finance

Since we could not find sponsorship for Knightcop, we evenly divided the cost (Table 27). An early estimate of $1000 gave us a ball-park figure of what the entire project should cost. We knew we had to keep the cost within the ability of all group members, so we agreed on the goal of keeping the entire project cost under $1000. We exchanged ideas on how we could reduce the cost. We

thought of donors who would be interested in sponsoring the project. We learned that many companies offer samples for free. We agreed to fully explore this option. We also looked to family members and friends. Unfortunately, we weren't able to generate help this way. We knew that eventually we would have to come up with most, if not all, of the cost ourselves. The cost of the project would be split evenly between three members.

For parts that we could not get samples for, we were prepared to pay. Each of the arts we needed could be supplied by different vendors. But which one would be buy from? We took several factors into consideration before choosing which companies to purchase our parts from. Hearing stories of groups anxiously awaiting delivery of back-ordered items, unwisely buying from high-priced sellers, and other scare stories gave us a desire to plan ahead. We wanted to strike a balance in choosing which suppliers to purchase from. It could not simply be a matter of it being the lowest cost seller getting our business. Such things as a company's reliability, reputation and support, were important factors for us. We weighed these questions and decided to purchase as many items from Sparkfun and Digikey. These two companies have a very good reputation. Both Sparkfun and Digikey have well reputed support. They also have discussion board with members offering suggestions and their experience with products they've purchased. We knew that even if we didn't purchase from these two companies, having access to their community of hobbyist would be a big help for us. Shopping from Digikey or Sparkfun also gave us peace of mind in terms of them having the supply they indicated and it being delivered in the indicated time.

Although we were not able to get sponsorship, we were able to come up with creative ways to obtain things we needed. We approached people within the robotics community for donations and we were able to obtain some items this way. Scavenging and repurposing parts proved worthwhile. Doing these things helped us realize a substantial savings.

We are still hopeful that in the coming months, we can gain sponsorship. We will continue to approach people and organizations with the goal of achieving full sponsorship. A rough estimate of various parts required and the costs involved is given in table 27 below:

Part	Manufacturer	Quantity	Unit Price	Net Price
Ultrasonic Sensors	Maxbotix	4	$25.95	$103.80
Potentiometer	中国制造	1	$1.95	$1.95
Drive Motors	CIM	4	$25.00	$100.00
Manipulator	OWI Robots	1	$45.00	$45.00
Arm Motor Controllers	Texas Instruments	3	$2.00	$6.00
Drive Motor Controllers	Vex Robotics	4	$69.99	Donated
Micro Switch	Sparkfun	2	$0.95	$1.50
RN-XV Wi-Fi Module	Roving Networks	1	$34.94	$34.94
Temperature Sensor	Maxim Integrated	1	$4.25	$4.25
Ambient Light Sensor	Sharp Electronics	1	$0.81	$0.81
IP Camera	Foscam	1	$65.00	$65.00
AVR ISP	Atmel	1	$34.00	$34.00
Arduino Mega 2560	Arduino	1	$45.95	$45.95
ATmega2560	Atmel	1	$18.00	Sampled
MiniBoards	ExpressPCB	3	$33.00	$99.00
72” Aluminium Angle	Vex Robotics	3	$29.95	$89.85
Sensor Brackets	Vex Robotics	--	--	50.00
Trimpot 10K with Knob	Sparkfun	4	$0.95	$3.80
ToughBoxNano 500 Hex Shaft	AndyMark	4	$78.00	Donated
6” Plaction Wheel w/ Tread	AndyMark	4	$29.00	Donated
Hex Wheel Hub	AndyMark	4	$10.00	Donated
Misc Mechanical Parts	--	--	--	50.00
Misc Electrical Supplies (Resistors, Capacitors, Wires etc)	--	--	--	50.00
TOTAL				$785.85

Table 27 - Budget

8.2.0 Responsibilities

Divide and conquer is an effective method to help make complex task manageable. We employed this method by assigning responsibility based on each of the group member's strengths and interest. We stripped down the process of building Knighcop. Our block diagrams helped us understand the subcomponents needed. We looked at each of the subcomponents and divided responsibilities accordingly.

Elean was responsible for both hardware and software elements. The chassis design and construction was done by him. The motor, manipulator, gears, arm, obstacle avoidance, power distribution, motor controller and microcontroller were designed and built by him. On the software side, he was responsible for coding of the motor controller, obstacle avoidance and microcontroller.

Nitin also had responsibilities ranging both hardware and software. Nitin was responsible for the design and building of the sensors. On the software side, he was responsible for designing the motor controller, microcontroller, the hub and android applications.

Figure 45 gives a breakdown of each member's responsibility.

	Elean	Nitin
Chassis Design and Construction	X	
Communications		X
PC + Android Apps		X
Obstacle Detection	X	X
Microcontroller Programming	X	X
PCB Design	X	
Power	X	

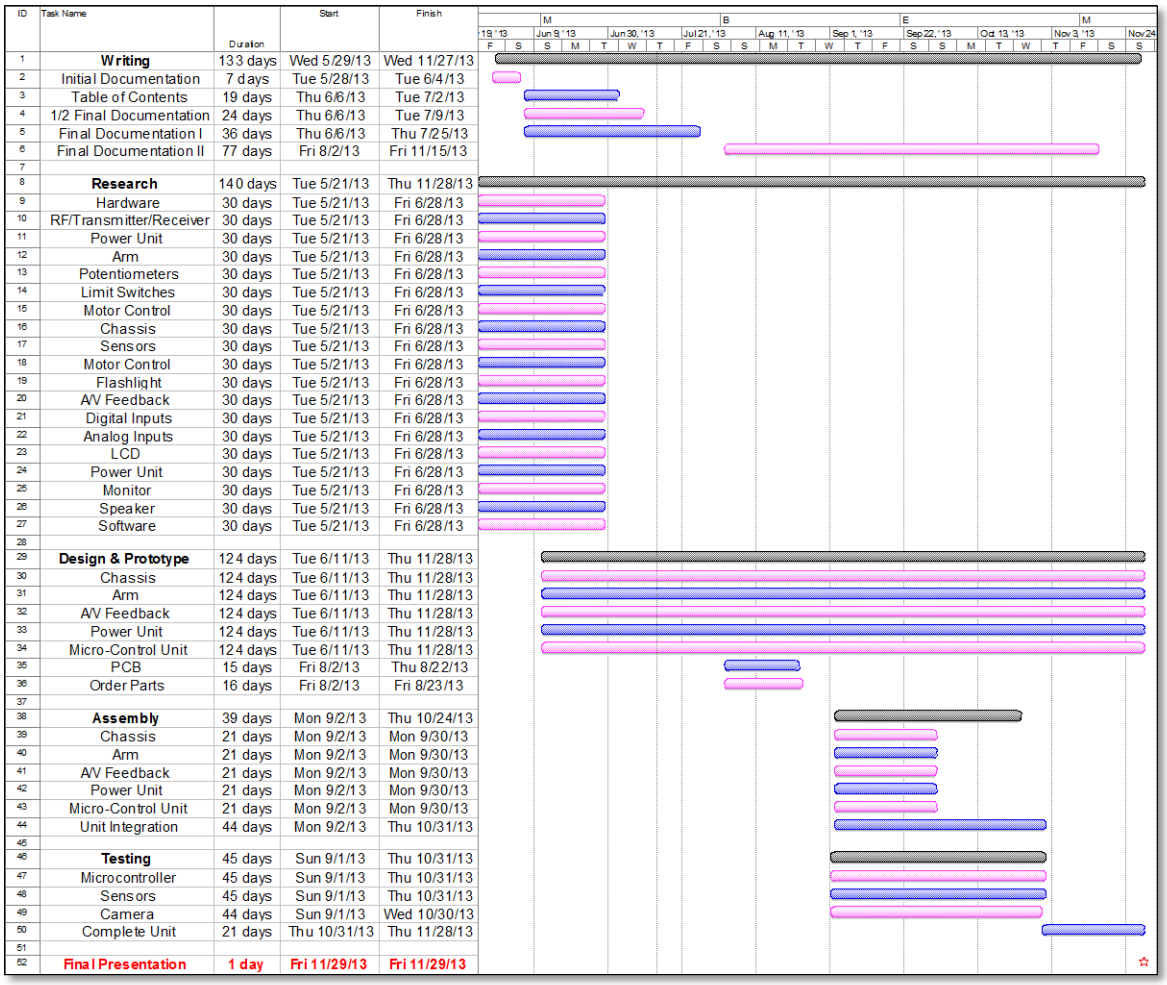
Table 28 - Members Responsibilities

8.3.0 Project Milestones

An easily overlooked but very important aspect to achieving a collective goal is synchronization. Care should be taken to prevent a project's development from stalling. A progress chart can be extremely helpful in doing this by coordinating project member's activities. Each group member would know what needed to be done and how much time should be spent on any given stage.

Our group decided a Gantt chart would be helpful in achieving our intermediate goals. We constructed the Gantt chart, Figure 46, to help organize the project.

From meeting our first deadline as a group to completing our Senior Design I paper, each task was mapped out using the Gantt chart. We agreed that our use of a Gantt chart helped us be more productive and avoid certain types of confusion. Our Gantt chart spans the entire length of time for senior design I and senior design II. It also includes tasks to be worked upon during our summer break. We agreed that working during the break would be a wise thing to do. We did not know the exact date for our final presentation, but we completed the Gantt chart using our best estimate.



ID	Task Name	Duration	Start	Finish
1	**Writing**	133 days	Wed 5/29/13	Wed 11/27/13
2	Initial Documentation	7 days	Tue 5/28/13	Tue 6/4/13
3	Table of Contents	19 days	Thu 6/6/13	Tue 7/2/13
4	1/2 Final Documentation	24 days	Thu 6/6/13	Tue 7/9/13
5	Final Documentation I	36 days	Thu 6/6/13	Thu 7/25/13
6	Final Documentation II	77 days	Fri 8/2/13	Fri 11/15/13
7				
8	**Research**	140 days	Tue 5/21/13	Thu 11/28/13
9	Hardware	30 days	Tue 5/21/13	Fri 6/28/13
10	RF/Transmitter/Receiver	30 days	Tue 5/21/13	Fri 6/28/13
11	Power Unit	30 days	Tue 5/21/13	Fri 6/28/13
12	Arm	30 days	Tue 5/21/13	Fri 6/28/13
13	Potentiometers	30 days	Tue 5/21/13	Fri 6/28/13
14	Limit Switches	30 days	Tue 5/21/13	Fri 6/28/13
15	Motor Control	30 days	Tue 5/21/13	Fri 6/28/13
16	Chassis	30 days	Tue 5/21/13	Fri 6/28/13
17	Sensors	30 days	Tue 5/21/13	Fri 6/28/13
18	Motor Control	30 days	Tue 5/21/13	Fri 6/28/13
19	Flashlight	30 days	Tue 5/21/13	Fri 6/28/13
20	A/V Feedback	30 days	Tue 5/21/13	Fri 6/28/13
21	Digital Inputs	30 days	Tue 5/21/13	Fri 6/28/13
22	Analog Inputs	30 days	Tue 5/21/13	Fri 6/28/13
23	LCD	30 days	Tue 5/21/13	Fri 6/28/13
24	Power Unit	30 days	Tue 5/21/13	Fri 6/28/13
25	Monitor	30 days	Tue 5/21/13	Fri 6/28/13
26	Speaker	30 days	Tue 5/21/13	Fri 6/28/13
27	Software	30 days	Tue 5/21/13	Fri 6/28/13
28				
29	**Design & Prototype**	124 days	Tue 6/11/13	Thu 11/28/13
30	Chassis	124 days	Tue 6/11/13	Thu 11/28/13
31	Arm	124 days	Tue 6/11/13	Thu 11/28/13
32	A/V Feedback	124 days	Tue 6/11/13	Thu 11/28/13
33	Power Unit	124 days	Tue 6/11/13	Thu 11/28/13
34	Micro-Control Unit	124 days	Tue 6/11/13	Thu 11/28/13
35	PCB	15 days	Fri 8/2/13	Thu 8/22/13
36	Order Parts	16 days	Fri 8/2/13	Fri 8/23/13
37				
38	**Assembly**	39 days	Mon 9/2/13	Thu 10/24/13
39	Chassis	21 days	Mon 9/2/13	Mon 9/30/13
40	Arm	21 days	Mon 9/2/13	Mon 9/30/13
41	A/V Feedback	21 days	Mon 9/2/13	Mon 9/30/13
42	Power Unit	21 days	Mon 9/2/13	Mon 9/30/13
43	Micro-Control Unit	21 days	Mon 9/2/13	Mon 9/30/13
44	Unit Integration	44 days	Mon 9/2/13	Thu 10/31/13
45				
46	**Testing**	45 days	Sun 9/1/13	Thu 10/31/13
47	Microcontroller	45 days	Sun 9/1/13	Thu 10/31/13
48	Sensors	45 days	Sun 9/1/13	Thu 10/31/13
49	Camera	44 days	Sun 9/1/13	Wed 10/30/13
50	Complete Unit	21 days	Thu 10/31/13	Thu 11/28/13
51				
52	**Final Presentation**	**1 day**	**Fri 11/29/13**	**Fri 11/29/13**

Figure 48 - Gantt chart

8.0.0 Conclusion

The experience of completing Senior Design I is unlike other engineering classes. In Senior Design I, we began a transition from academia to practical applications. Managing things such as cost, time, and competing visions proved to be a challenge. It was a challenge our group was prepared for. We also conducted a significant amount of independent research. All of our actions were coordinated by an ultimate goal. There were vast stretches for which we worked

independently. It was sort of interesting to see how working separately on each of our component would eventually lead to the creation of a much larger singular unit.

Working cooperatively over an extended period of time is an invaluable experience. Along the way there were many opportunities for challenges. Often times it is easy to know what you can do alone. A person can have a good sense of what they are capable of. It's not so easy to know what others that you're depending on is capable of, unless you have lots of experience working together. Throughout this semester, we were dependent on others to do their share. It was not possible for one person to do the work of all. This fact also helps avoid one person from overtaking the entire project. By the way the course was designed, we were forced to work cooperatively.

Dealing with the cost constraints of the project was not easy. Restricting our budget to $1000 ultimately affected what KnightCop could realistically do. But even with this restraint, there was an opportunity for creativity and resourcefulness.

We all had ideas of how one aspect or another of the project should be conducted. We worked through those ideas. It wasn't always easy for a member to give up an idea that he thought was important. At points along the process, decisions must be made. Exchanges on which way to go cannot go on indefinitely. We managed ourselves through those moments. We learned more about each other's style of doing things. We also learned about each other's strengths and weaknesses.

Senior Design I helped us gain a sense of what will be needed in Senior Design II. We expect Senior Design II to be even more challenging than Senior Design I. With Senior Design I, we defined and planned our project. In Senior Design II we will be expected to make it work. No matter how much sweat and effort we pour into the project, we must have it working for final presentation. That presents a great deal of pressure. Everything is on the line and we're playing for all the marbles. We look forward to next semester. KnightCop will go from being a concept to becoming a reality. We expect testing to be a difficult but rewarding stage in the development of KnightCop.

As we step closer to ending one journey, we are ready to beginning another.

Appendix A: Copyright Permissions

MaxBotix

Thanks for the email. I am glad to assist you with your questions.

All of our sensor images are available for download on the bottom of each product page, in a ZIP file. The file contains all the beam patterns and product images. We just ask that if the images are used, that we are sourced as the supplier. On each product page there is a link that says "High Resolution Pictures.zip".

Does this information help?

Please let me know if you have any questions.

Best regards,

Tom Bonar
Technical Support
of MaxBotix Inc.
Phone: (218) 454-0766 ext 2#
Fax: (218) 454-0768
Email: thomas@maxbotix.com

iRobot

Hello Kelly,

I am a computer engineering student at the University of Central Florida working on a senior design project. I would like permission to use some of the high definition images that are provided on this link:
http://www.irobot.com/us/Company/Press_Center/logos_photos_videos.aspx#

Thank you for your time and I look forward to hearing back from you soon,
Elean Atencio

Thank you for your quick reply Elean.
The project sounds great and you may use our photos.

Thank you,
Kelly

SuperDroid Robots

You can use them

SuperDroid Robots Inc. | www.SuperDroidRobots.com | www.SDRobots.com
224 Technology Park Lane | Fuquay Varina, NC 27526
919-557-9162 (phone) | 775-416-2595 (fax) | sdr@SDRobots.com (email)

-----Original Message-----
From: Elean Atencio [mailto:atencioelean@knights.ucf.edu]
Sent: Wednesday, July 24, 2013 10:13 PM
To: contact@sdrobots.com
Subject: www.superdroidrobots.com - Contact Us Form

www.superdroidrobots.com:
Email Address: - atencioelean@knights.ucf.edu
Name: - Elean Atencio
Phone/Fax: - 4075521552
Comments: - Hello,
I am a computer engineering student at the University of Central Florida working on a senior design project. I would like permission to use some of the images found on your site.
Thank you for your time and I look forward to hearing back from you soon, Elean Atencio

AndyMark

Dear Elean Atencio,

Thank you for this request. Yes, please use these images for your project as you wish. When you are finished, would you mind sending us a version of your completed project, or at least a summary?

Sincerely,
Andy Baker

Hvlabs

Hi Elean,
Thanks for your message. Yes you can use my images, providing the website address/logo is displayed.
Hope your project goes well!

Regards,
Oliver

Hvlabs. Com

Vexpro

Hello Elean, feel free to use our images and logos as long as they are supporting your events and are not on items for retail.

Best,

Brian

On Mon, Jul 22, 2013 at 8:51 AM, atencioelean <atencioelean@knights.ucf.edu> wrote:
Hello Brian,

I am a computer engineering student at the University of Central Florida working on a senior design project. I would like permission to use some of the images and datasheets provided on the vex pro site: http://www.vexrobotics.com/vexpro

Thank you for your time and I look forward to hearing back from you soon,
Elean Atencio

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.

OWI

From: kitsusa@hotmail.com on behalf of Jerry G <orders@omnitronelectronics.com>
To: atencioelean@knights.ucf.edu
Cc:
Subject: RE: New Contact Request: Image usage

HELLO ELEAN,

GO AHEAD...GOOD LUCK ON YOUR PROJECT

KIND REGARDS,

JERRY GRAYSON,
CUSTOMER SERVICE

Texas Instruments

Use Restrictions

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

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

FOSCAM

Foscam US Support via 7l6czvs6hisj.e-l5wvma0.e.bnc.salesforce.com

to me

Dear Nitin,

Yes that would be fine! I hope you're liking our cameras.

Please let us know if you have any questions or concerns!

Best Regards,
Zohaib

Foscam Digital Technologies LLC
Support Team
1-800-930-0949
support@foscam.us
www.foscam.us

Let me know how I'm doing! If you have any praises or complaints, please send them to feedback@foscam.us!

:::a0WE0000000351oMAA:::

All Battery

Wesley Edmund <wesley.b.edmund@gmail.com>

to service

Hello

I am an electrical-engineering student, attending the University of Central Florida. I have been researching for my senior design project and would like your permission to include an image(s) from your website. If you would like additional information concerning this request, I would be glad to provide it.

Thank You

All-Battery Service

to me

Hi Wesley,

Thank you for choosing our website. You have the permission to do it.

Kind Regards,
Customer Service
All-Battery

Cheap Battery Packs

Mike Frederick <mike@cheapbatterypacks.com>

to me

You have me permission to use an image from my site (as long as you include some reference to where it was obtained)

Mike

Name: **Wesley Edmund**
Email: **Wesley.B.Edmund@gmail.com**

Hello I am an electrical-engineering student, attending the University of Central Florida. I am researching for my senior design project and would like your permission to include an image(s) from your website. If you would like additional information concerning this request, I would be glad to provide it. Thank You Wesley Edmund

Battery Space

Wesley Edmund <wesley.b.edmund@gmail.com>

to sales

Hello

I am an electrical-engineering student, attending the University of Central Florida. I have been researching for my senior design project and would like your permission to include an image(s) from your website. If you would like additional information concerning this request, I would be glad to provide it.

Thank You
Wesley Edmund

Sales via yahoo.com

to me

Hi Wesley,

Yes, you can use our images from the website.

Thanks,

Jasmine

DZERV

Wesley Edmund <wesley.b.edmund@gmail.com>

to mrosario, octaroon, mblopez

Hello

I am an electrical-engineering student, attending the University of Central Florida. I have been researching past projects for my senior design project and would like your permission to include an image(s) from your project. If you would like additional information concerning this request, I would be glad to provide it.

Thank You

Rob Smith

to me, mrosario, mblopez

Of course! Best of luck to you and your project!

A.B.E.C

Wesley Edmund <wesley.b.edmund@gmail.com>

to drew_boyles, marc.bianco, echa

Hello

I am an electrical-engineering student, attending the University of Central Florida. I have been researching past projects for my senior design project and would like your permission to include an image(s) from your project. If you would like additional information concerning this request, I would be glad to provide it.

Thank You

---

marc.bianco

to me

Sure, feel free to use any of our images.

A.M.O.R.E.D.

Wesley Edmund <wesley.b.edmund@gmail.com>

to andrew.lichens.

Hello

I am an electrical-engineering student, attending the University of Central Florida. I have been researching past projects for my senior design project and would like your permission to include an image(s) from your project. If you would like additional information concerning this request, I would be glad to provide it.

Thank You

Andrew Lichenstein

to me

That's fine, may I ask which image?

Appendix B: Datasheets

Ultrasonic Sensor EZ0:
http://maxbotix.com/documents/MB1000_Datasheet.pdf

Drive Motors:
http://files.andymark.com/CIM-motor-curve.pdf

TI Motor Controller:
http://www.ti.com/lit/ds/symlink/drv8833.pdf

Victor Motor Controller:
http://content.vexrobotics.com/vexpro/pdf/217-2769-Victor888UserManual_20130118.pdf

Jaguar Motor Controller:
http://content.vexrobotics.com/docs/217-3367-VEXpro_Jaguar_Datasheet_20130220.pdf

Spike Relay:
http://www.vexrobotics.com/vexpro/motor-controllers/217-0220.html

Atmel:
http://www.atmel.com/dyn/products/datasheets.asp?family_id=607

MSP430:
http://www.ti.com/product/msp430g2553

PIC:
http://ww1.microchip.com/downloads/en/DeviceDoc/41291D.pdf,
http://ww1.microchip.com/downloads/en/DeviceDoc/39964B.pdf

Wifi Module:
http://www.mouser.com/ds/2/349/WiFly-RN-XV-DS-219924.pdf

Temperature Sensor:
http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

Ambient Light Sensor:
http://www.mouser.com/ds/2/365/GA1A2S100LY-180860.pdf

IR Sensor:

https://www.sparkfun.com/datasheets/Sensors/Infrared/gp2y0a02yk_e.pdf

Ultrasonic Sensor SR04:

http://users.ece.utexas.edu/~valvano/Datasheets/HCSR04b.pdf

Appendix C: Works Cited

"AC Motor Guide."
http://www.anaheimautomation.com/manuals/forms/ac-motor-guide.php

"Brushless vs Brushed Motors."
http://www.dynetic.com/brushless%20vs%20brushed.htm

"DC Motor Control Using an H-Bridge."
http://itp.nyu.edu/physcomp/Labs/DCMotorControl

"Differences of AC vs DC Motors."
http://www.ehow.com/info_8072798_differences-ac-vs-dc-motors.html

"How Serial Ports Work"
http://computer.howstuffworks.com/serial-port.htm

“IR Vs. RF: Which is the best for my needs?”
http://www.aclasstechnology.com/downloads/IRvsRF.pdf

"PID Velocity DC Motor Controller."
http://upgrayd.blogspot.com/p/pid-velocity-motor-controller.html

“Understanding Gear Reduction”
http://www.teamdavinci.com/understanding_gear_reduction.htm

“Arduino Playground: Interfacing with Hardware”
http://playground.arduino.cc//Main/InterfacingWithHardware#ui

“A Brief Tutorial on Programming the ATMega (Arduino) without Arduino Software”
http://brittonkerin.com/cduino/lessons.html

“WiFly RN-XV Arduino Library”
https://github.com/harlequin-tech/WiFlyHQ

“Arduino Playground: Dallas Semiconductor's 1-Wire Protocol”
http://playground.arduino.cc/Learning/OneWire

“Foscam IP Camera CGI V1.27”
http://www.foscam.es/descarga/ipcam_cgi_sdk.pdf

“Arduino Mega PinoutDiagram”
http://forum.arduino.cc/index.php?PHPSESSID=b0e2fe8d5f09a2f88f1bd7009cdc71d5&/topic,146511.0.html

“PWM Pins on Microcontrollers”
http://www.societyofrobots.com/robotforum/index.php?topic=14901.0

“Android App Development Cheat Sheet”
https://docs.google.com/spreadsheet/ccc?key=0Aiakq5EFgZZTdGlmblFNMFZLN3huRnNUNzdkWDRoekE#gid=0

“Using Wi-Fi Direct for Service Discovery”
http://developer.android.com/training/connect-devices-wirelessly/nsd-wifi-direct.html

“The Java Tutorials: What is a Socket?”
http://docs.oracle.com/javase/tutorial/networking/sockets/definition.html

Appendix C: List of Figures()

Appendix C: List of Tables