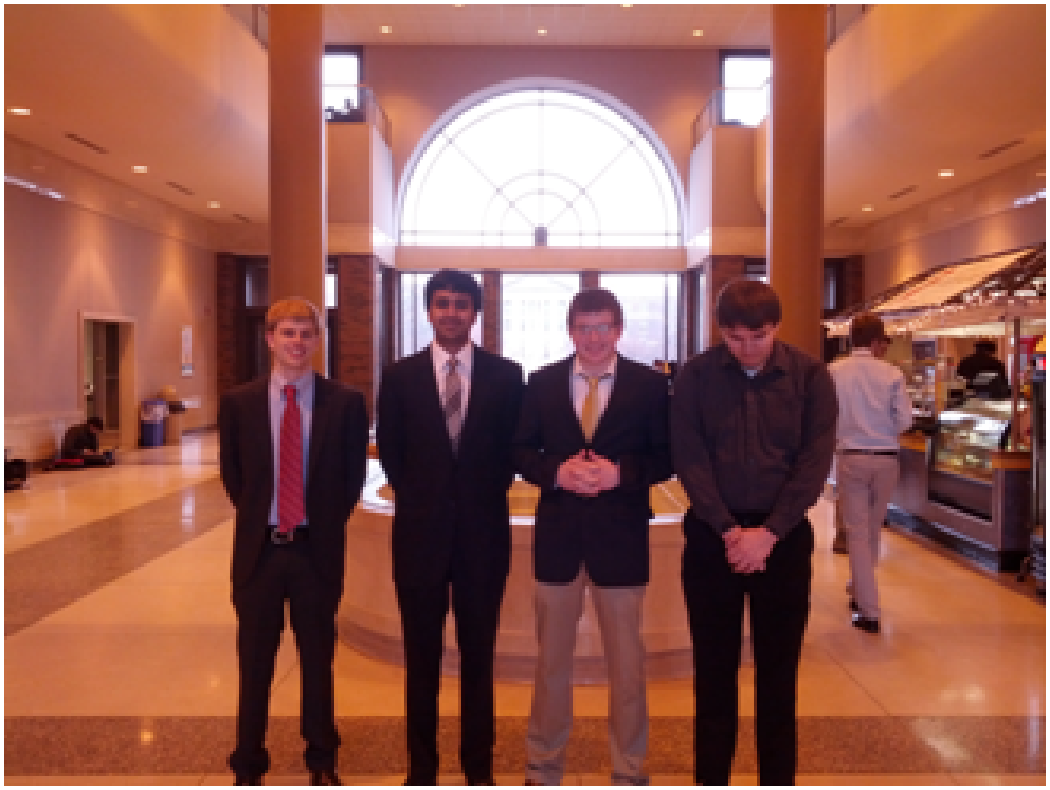
ECE 477 Final Report Spring 2013

Team # 16: Project Minotaur



Left to Right: Scott Stack, Neil Kumar, John Hubberts, Jon Roose

Team Members:

#1: _____Scott Stack_____ Signature: _____ Date: _____

#2: _____John Hubberts_____ Signature: _____ Date: _____

#3: _____Jon Roose_____ Signature: _____ Date: _____

#4: _____Neil Kumar_____ Signature: _____ Date: _____

CRITERION	SCORE	MPY	PTS
Technical content	0 1 2 3 4 5 6 7 8 9 10	3	
Design documentation	0 1 2 3 4 5 6 7 8 9 10	3	
Technical writing style	0 1 2 3 4 5 6 7 8 9 10	2	
Contributions	0 1 2 3 4 5 6 7 8 9 10	1	
Editing	0 1 2 3 4 5 6 7 8 9 10	1	
Comments:	*TOTAL*		

Table of Contents

Abstract

1.0 Project Overview and Block Diagram

1.1 Sample Heading 2

1.1.1 Sample Heading 3

2.0 Team Success Criteria and Fulfillment

3.0 Constraint Analysis and Component Selection

4.0 Patent Liability Analysis

5.0 Reliability and Safety Analysis

6.0 Ethical and Environmental Impact Analysis

7.0 Packaging Design Considerations

8.0 Schematic Design Considerations

9.0 PCB Layout Design Considerations

10.0 Software Design Considerations

11.0 Version 2 Changes

12.0 Summary and Conclusions

13.0 References

Appendix A: Individual Contributions

A.1 Contributions of Neil Kumar:

A.2 Contributions of John Hubberts:

A.3 Contributions of Scott Stack:

A.4 Contributions of Jon Roose:

Appendix B: Packaging

Appendix C: Schematic

Appendix D: PCB Layout Top and Bottom Copper

Appendix E: Parts List Spreadsheet

Appendix F: FMECA Worksheet

Abstract

Project Minotaur is a home security drone that lets users control it from anywhere in the world. Customers can load a web page that enables them to either manually control the drone with a live video feed or set the drone to patrol and alert users of a human being in the frame. The main idea is that while the user is away, they will be able to log onto a website and monitor their home or place of business. The robot has a patrol mode that allows it to autonomously patrol a building and search for intruders. It can detect and avoid objects and is able to differentiate between inanimate objects and a human form even if the person is not moving. If an intruder is detected, the robot will take a picture and alert the user of the intrusion via email. This will be accomplished with the help of a Microsoft Kinect. The Kinect which will act as both a camera for manual remote control as well as an object recognition sensor that will be able to detect obstacles and human forms.

1.0 Project Overview and Block Diagram

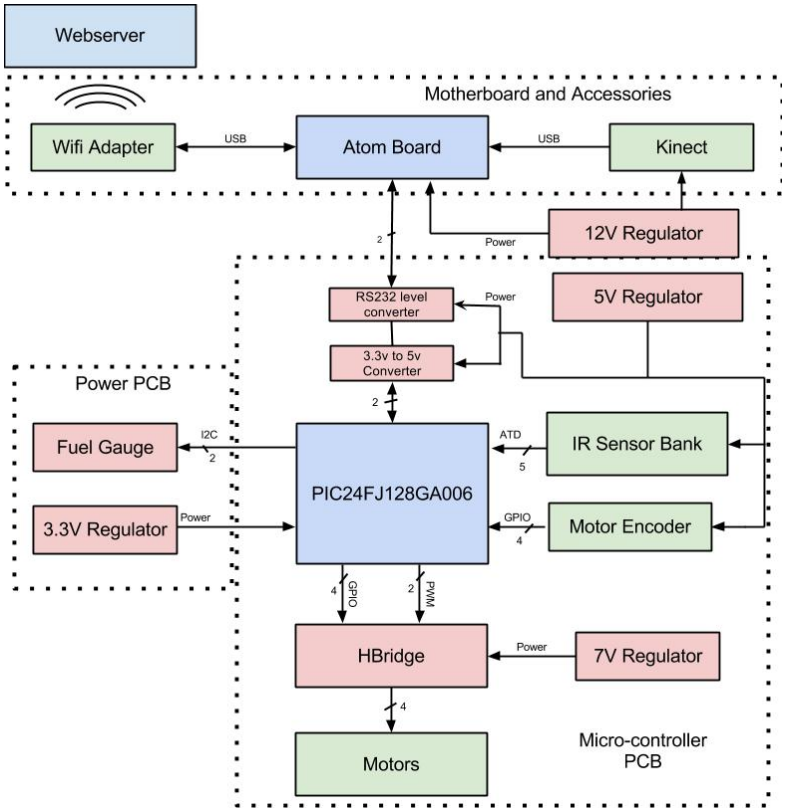


Figure 1.1 - Minotaur Block Diagram

There are 3 main control components of the overall system: the microcontroller and the ATOM board and the webserver. The webserver serves the command and control webpage to the end user and handles communication between this page and the actual robot. The server sends commands to the ATOM board on the robot through the internet. The server also receives the video stream that is being sent from the ATOM board and displays it in the web page. The next major component of the system is the ATOM board. The atom is in charge of receiving commands from the user and passing them on to the microcontroller to be executed. It is also responsible for interfacing with the

kinect. That means that it must grab the video/depth feed and either forward it to the command and control server, or use the information to avoid objects and patrol around a room. If the robot is in patrol mode, it must also interpret the depth data to search for a human form. If one is detected, it has the ability to capture a picture and send an email alerting the user of the intrusion. On the next level there is the microcontroller which communicates with the ATOM board using RS232 serial. The microcontroller receives specially formatted data packets from the ATOM board and responds accordingly. For example the ATOM board could tell the microcontroller to drive the motors forward at full speed, or it could request all of the sensor data. The microcontroller is responsible for keeping track of all of the various sensors and controlling the speed and direction of the motors. It can also do very simple obstacle avoidance if something gets very close. There are two PCBs on the robot that house all of the power supplies. There are four power supplies that supply 3.3 volt, 5 volt, 12 volt, and 7.2 volt power to various components in the system. A 14.8 volt 4400mAh Lithium Ion battery was used to power the whole robot.

2.0 Team Success Criteria and Fulfillment

1. An Ability to control the speed and direction of a robot

This criteria was met fully in the finalized package - the magnitude and direction of robot travel could be provided by either the web client or a direct serial connection to a computer, and the robot would move accordingly.

2. An Ability to automatically detect and avoid obstacles

This criteria was met fully in the finalized package - using the front and rear mounted sensors, if the drone detected an obstacle, it would ignore its previously provided command, and instead act to avoid that obstacle by backing away (if in front) or moving forward (if behind).

3. An Ability to capture and transmit live video from a Kinect to a Web Server

This criteria was met in the final package, but with some minor stipulation - the robot was able to transmit Kinect video data to a webclient (as

RGB, IR, depth fields, and other forms), but it wasn't entirely live, there was a persistent ~750ms lag between the occurrence of an event and that event's replaying on the screen. It is worth noting, however, that the lag was small enough that Jon Roose could drive our robot from baltimore airport, and make sharp turns and delicate maneuvers without having to engage the collision detection and avoidance mechanisms.

4. An Ability to control the movement of the robot through a web interface

This criteria was fully met in the final package - instructions could be relayed through the command and control website, the the robot would respond accordingly as outlined in criteria #1's evaluation.

5. An Ability to identify and respond to the detection of a human

This criteria was fully met in the final package - upon detection of a human skeleton, the robot would relay an email to the user. The address at which they would be emailed could be specified at the command and control website. Additionally, the robot would send this email through a 3rd party gmail account in order to avoid getting marked as spam.

3.0 Constraint Analysis and Component Selection

Many parts had to be chosen for this robot. The main components of this design are the motherboard that will be used to handle very intensive calculation and the microcontroller that control all of the low level peripherals on the robot. First the criteria for selecting a motherboard will discussed, then the microcontroller selection criteria will be discussed. Finally, power constraints and packaging constraints will be outlined.

Several tasks that the robot must perform are very computationally intensive. In particular, live video compression and object recognition will require heavy computation. The robot must grab video frames from the Kinect, compress them, and transmit them over the internet to the user. The Kinect camera captures 640x480 pixel images in 24 bit color which means that the size of each frame is 900kB. Since the speed of the internet is not guaranteed and the video is intended to be live, the compression of this data needs to take as little time as

possible to minimize the total latency. Object recognition is the other computationally intensive task that the robot must perform. The Kinect sensor will capture depth map images of a scene and it is the receiving computer's job to make sense of that data. While the robot is patrolling, it will take in these depth images and attempt to identify a human form based on these depth maps. A decent amount of computing power will be required to analyze the incoming frames in real time. This data is not quite as time sensitive as the compression and transmission of video data, but it still has to be performed quickly enough so that any human entering the scene will be identified. To perform these tasks, an Atom based motherboard will be used. The Atom processor will be capable of performing these time sensitive calculations in a reasonable amount of time. The robot will also have to receive commands from the user through the internet and respond accordingly. The amount of data will be relatively small, so no real computational power will be required for this task. Additionally, it will need to sample various sensors in order to detect objects and control the motors accordingly. This will also require very little computational power because sensors only need to be sampled on the order of every hundred milliseconds. Both motor control and sensor sampling will occur on a relatively low speed microcontroller since there is no heavy computation associated with those peripherals.

There were several main constraints that were taken into account when choosing the motherboard for this project. Firstly, it had to be as fast as possible to perform the necessary calculations a quickly as possible. Second, it must have at least 3 USB ports and 1 COM port for a wifi adapter, USB flash drive to run the OS off of, the Kinect, and communication with the microcontroller. Lastly, it must be able to run off of 12 volt DC power. With that in mind, two boards were chosen. The first choice was the Advantech MIO-5250 with an Atom N2600 processor and the second choice was the NORCO BIS-6630 with an Atom D2700 processor. The NORCO board was chosen for several reasons. The main reason was that the NORCO board incorporated a dual core 2.13 GHz processor while the Advantech board had a single core 1.6 GHz processor [16][17]. For this application, it would be best to get the most powerful board possible while still maintaining low power. Both boards provided sufficient peripheral interfacing, however the NORCO board was significantly cheaper than the slower Advantech. This is why it made more sense to choose the NORCO board.

The robot’s microcontroller requires several functions that came into play when choosing one. It will use general purpose I/O to drive the left and right motors through an H-Bridge integrated circuit (IC). This H-Bridge IC will require 6 output pins. Two pins will be used to enable the left and right motors, and 4 pins will be used to dictate the direction that the motors will rotate. The general purpose I/O pins on our

chosen microcontroller (PIC24FJ128GA) operate at 3.3 volts and are capable of sourcing or sinking 18mA [8]. The inputs to the chosen L298 H-Bridge are compatible with the outputs of the PIC microcontroller with a DC noise margin of 0.1 [8][9]. If this noise becomes a problem, there are spare channels on the 3.3 volt to 5 volt level converter that could be used to interface with this IC [12]. The microcontroller will need to communicate with the Atom board through an RS232 interface which requires a chip to convert 5 volt logic levels to the +/- 9 volt differential levels that RS232 requires. The MAX233 chip that was chosen to perform this task has CMOS inputs that are not compatible with the I/O pins on the PIC microcontroller. An intermediate buffer will be required in order to convert the 3.3 volt level signals of the microcontroller into the 5 volt signals required by the MAX233 chip. A TXB0108 level converter chip has been chosen to perform this task [12]. The microcontroller requires 2 PWM output channels to drive the left and right motors at variable speeds. The PWM granularity is of little importance in this application because the PWM will not need to be very accurate. The microcontroller also requires at least 1 UART interface in order to communicate with the Atom board. This takes up 2 pins on the microcontroller, one for the transmit line and one for the receive line. The robot has 5 infrared sensors to facilitate obstacle avoidance and navigation. Those IR sensors output an analog voltage that is proportional to the distance from an object to the sensor [23]. This requirement means that 5 A/D channels and pins are required on the microcontroller. Finally, a battery fuel gauge will be required to monitor the battery life of the robot and relay that information to the user. To accomplish this task the team has decided to use a fuel gauge integrated circuit that communicates with the microcontroller via an I2C interface [13]. In addition to the battery fuel gauge, the robot incorporates 2 motor encoders that send pulses to track distance and speed traveled. The microcontroller requires a timer and GPIO pins to run these encoders.

After examining all of the constraints for a microcontroller, the team came up with several options. After much searching, the team came up with two choices that fit all of the constraints. The first choice was an Atmel ATmega128 microcontroller and the second was a PIC24FJ128GA microcontroller. Both microcontrollers meet the peripheral constraints, so the decision was based on several other factors. In the end the PIC24FJ128GA was selected to be the robot's microcontroller for several reasons. One factor in choosing the PIC over the Atmel was its programmable internal oscillator. The microcontroller does not need very much processing power, so the high clock speed of the Atmel would just be wasting power. The PIC can be easily clocked at a lower frequency which will save power. The PIC also offered more peripheral features so the design could be easily expanded upon. Finally, a development board for the PIC was readily available to the team so prototyping

could begin immediately.

Power is a large constraint with this robot. It has many components that consume a large amount of power. The robot is required to be battery powered which means that power efficiency is extremely important. The battery that was chosen, a 4400mAh LiIon battery, has a fairly high capacity because there is no way to improve the efficiency of some if the parts that are being used. For example, the Kinect and Atom board have a minimum power requirement that cannot be negotiated. In addition, the application of this robot requires that it be active for an extended amount of time. The user will not be home when using the robot, so ideally it should be operational until the user gets home. Physical size is not a limitation of the battery due to the large chassis that will be used for the robot.

The robot will need to be able to hold the weight of all of the components on board and be able to withstand some minimal abuse. It should also be able to quickly and easily traverse any indoor terrain. To fulfill these requirements, the team has chosen to use the VEX robotics to obtain a chassis, motors, wheel encoders, and tank style treads [14]. The footprint of this robot will be about 1.5x1.5 feet. This is small enough to navigate around home, but large enough to carry all of the electronics reliably.

4.0 Patent Liability Analysis

After searching for patents pertaining to our robot through freepatentsonline.com it is clear that the project will be at great risk of infringing upon no fewer than three patents. The first two patents pertain to software algorithms meant to improve navigation accuracy by utilizing planar landmarks within the home. The third and most concerning patent, owned by Fujitsu, concerns the overall design of our project. Together all of these patents mean that there is a high probability that the team will be forced to license some of the patents from their respective owners.

The first patent, developed by Gutmann et al., was filed on March 17, 2005 and issued on November 8, 2007 as US0257910A1. This patent is for an algorithm which detects planes within three dimensional depth sensor data. The initial claim of the patent does not place a

requirement upon the method of obtaining these points, which makes the initial claim a purely algorithmic patent. Later claims require the use of laser finders or parallax detecting range finders to generate the 3D data, but the initial claim creates no such restriction. The patented algorithm within claim 1 essentially starts off by estimating which points lie within a single plane through some unspecified means. Our algorithms do this via least-squares regression and line fitting. The algorithm then grows the estimated plane to include as many distance points as possible. Finally, the algorithm fits lines to the detected planes through linear regression techniques and applies the lines to find the closest geometric approximation of the plane. Later claims involve the refining of this overall algorithm, for example through the grouping and elimination of lines and points within the planes, and the merging of similar planes. It is possible that our design has worked around this algorithm by finding only vertical or horizontal planes within 3d space. It appears that the algorithm patented here is for a more generic algorithm which identifies the planes in true 3d form.

The second patent, developed by the Honeywell International Inc., was filed on September 10, 2009 and issued on March 24, 2010 as EP2166375A2. According to claim 1 the patent is for using features of planes detected within an "image sensor operable to obtain range data" and uses the plane features to identify the current "reference orientation" of a robot. The patented algorithm also stipulates that it will be combining the measurements of several walls to identify the reference orientation of the robot. Essentially the algorithm described in claim 1 will use the normals of detected planes to identify the position and orientation of the robot in relation to the planes. Claim 2 expands this algorithm to include an estimation of the distance traveled by the robot between two successive "scenes", or sets of range data. Claim 3 then expands the algorithm to be potentially coupled with a third sensor for estimating motion. Claims 4-7 then proceed to offer several methods of identifying the normal vectors of planes for use by the algorithms described in the earlier claims. The final claim, claim 8 proposes using only the most "powerful" selected planes to determine the orientation of the robot. Once again, it appears that the simplifying assumption that walls are either purely vertical or purely horizontal will allow us to bypass this patent. This is because we are never computing normals within a 3d space, but instead we are simplifying 3d features into a 2d floorplan before we calculate our reference orientations and distances to the walls. Additionally, although the wall detection orientation identification portion of the code was completed, we have not yet implemented a method of using the walls for navigation purposes.

The third patent, developed by Fujitsu Limited, was filed on June 15, 2007 and issued on January 4, 2005 as US7218993. Claim one

describes an abstract system where an "autonomous mobile robot" communicates wirelessly with a "base station". The described robot then proceeds along a "predetermined path at predetermined times" and records images of the location before transmitting them to the base station capable of feeding these images to an external device. Claims 10, 18, 21, and 22 essentially re-list the same robot as claim 1 with slight variations upon the robot's relationship to the base station and the base station's relationship to the external devices. Other claims expand the various iterations of the robot to include "moving images", sound sensors, and notably the detection of "suspicious objects". Because of the broad system level language used within this patent, it is truly neither an algorithmic nor a hardware patent. This creates quite a challenge because the system that is being described within the patent is actually the sum total of the project our team has set out to create. It can additionally be noted that Fujitsu released a product which implemented the listed system as MARON-1 in 2002. It appears that individual pieces of this overall system may have been previously patented within the Japanese patent system several years before the US patent application was filed. It appears that our inability to complete the robot in its entirety (as of this date) has allowed us to bypass this patent liability. Because we have not yet been able to implement patrol mode in our design, the robot cannot infringe upon the "predetermined path at predetermined times" requirement of the patent. Additionally, the algorithm which the team intends to implement after this report will utilize an A* algorithm to identify the best patrol path for maximum coverage. This differs from the patent in that no human assistance is required for the implementation of patrol mode.

The first patent by Gutmann et al. applies to the planar detection system of our robot. The robot will use the planar detection within later algorithms which will allow us to more accurately reckon the current location of the robot within the patrollable area. Without this feature, the robot would have grave inaccuracies within its room mapping and navigation functionalities which would result in a significant loss of value. Thankfully, it appears that our project may be too simplistic to infringe upon the entirety of this patent. Initially it appeared that our robot would not be using the distance to the wall to identify its orientation, but that has since been added to the algorithm. The patent's usage of linear regression and has quite a bit of similarity to our algorithm, but our product will not utilize cross products within its wall-identifying algorithms. We managed to accomplish the lack of cross-products by simplifying all walls into lines on a two dimensional floormap of the patrollable area. This has the useful repercussion of eliminating our infringement upon this patent because of the patent's reliance upon tracking planes within three dimensional space.

The second patent, by Honeywell International Inc., applies to using the planar surfaces to reckon the orientation and location of the robot within 2D or 3D space. This algorithm is required as part of the same features listed as pertaining to the Gutmann patent, so again the ideas related to this patent are key to a significant amount of the value of this project. Although I was originally doubtful of the team's ability to work around this patent, it appears that our system of using simplifying the walls into two dimensional coordinates has saved us once more. The Gutmann patent no longer applies to our project because we never actually implement a system for calculating the planes in three dimensional space, even though we do go as far as calculating the y/x, z/x, and z/y slopes and intercepts of lines that measure the planes.

The third patent, by Fujitsu, was a source of much concern in the original patent liability analysis. This patent dictated a patrollable robot implemented using a user's mobile device or PC, a middle-man server much like our C&C server, and a robot which uploaded images, sounds, and other information to the user through the middle-man server. Thankfully, their robot patrolled a "predetermined path at predetermined times" whereas our robot would utilize automatic patrol algorithms in order to establish the best patrol path. Although we would ideally enable the user to set up customized patrol paths, that feature is not entirely necessary for the project. To get around this issue, we could allow the user to log in and set patrol paths manually, without them being initiated at "predetermined times", however if the waypoins are sequentialized, it may still infringe upon the patent because a sequence could be considered a measure of time.

Because the recommended legal actions and their rationale have already been detailed above, I will use this section to summarize the recommended legal actions.

Gutmann patent: US0257910A1 – Low risk, designed workaround simplifications

Honeywell patent: EP2166375A2 – Low risk, designed workaround simplifications

Fujitsu patent: US7218993 – Possibly require a license depending upon features of final consumer-ready design.

5.0 Reliability and Safety Analysis

In analysis of the reliability of our design, four primary components were determined to be particularly at risk; the PIC24FJ128GA microcontroller, the L298P013TR H-bridge, the TPS5450-Q1 7.2V switching regulator, and the LM25085 12V switching regulator. The microcontroller was determined to have a relatively high likelihood of failure, it is the highest complexity circuit on the device's primary printed circuit board, and has a large pin count. Additionally, investigation of the microcontroller's reliability seemed prudent from the standpoint of preserving the device's general functionality, as it is directly responsible for the interpretation of all sensor data, and control of the motors. In continuation with the identification of at-risk components, in-depth analysis was run on the h-bridge and the 12V/7.2V switching regulators, as they all handle large amounts of current (on the order of amps). Like the microcontroller, each of these components is directly responsible for a large portion of the device's general functionality; if the 7.2V switching regulator or the h-bridge were to fail, the motors would malfunction. If the 12V switching regulator failed, the ATOM motherboard would lose power, and any communication with the web server, along with autonomous navigation, would be lost. Shown below are the tabularizations of the parameters required for, and results derived from, the calculations of λp and MTTF for the four previously identified components.

Figure 2.0.1 Microcontroller (PIC24FJ128GA) [8]

MIL-HDBK-217F Class		Monolithic Bipolar and MOS Digital Microprocessor Devices	
Determination Equation		$\lambda p = (C1 \times \pi\tau + C2 \times \pi E) \times \pi Q \times \pi L$	
Parameter	Description	Value	Comment
C1	Die Complexity	0.28	Value for 16 bit microprocessors
$\pi\tau$	Temperature Coefficient	3.075	Ea=0.35 (CMOS Logic), TJMAX=125
C2	Package Failure Rate	0.032	64-Pin Non-Hermetic SMT
πE	Environmental Constant	4	Ground Mobile
πQ	Learning Factor	1	More than two years in production

πL	Quality Factor 10	Commercial Quality	
λp	Failures per 106 Hours	9.897	
MTTF	Mean Time to Failure (hr)	101083	11.53 Years

Figure 2.0.2 H-Bridge (L298P013TR) [9]

MIL-HDBK-217F Class	Monolithic MOS Digital and Linear Gate/Logic Array Devices		
Determination Equation	λp = (C1 x πτ + C2 x πE) x πQ x πL		
Parameter	Description	Value	Comment
C1	Die Complexity	0.01	Logic device with less than 100 transistors
πτ	Temperature Coefficient	10	Ea=0.4 (TTL Logic), TJMAX=150
C2	Package Failure Rate	0.0082	18-Pin Non-Hermetic SMT
πE	Environmental Constant	4	Ground Mobile
πQ	Learning Factor	1	More than two years in production
πL	Quality Factor 10	Commercial Quality	
λp	Failures per 106 Hours	1.328	
MTTF	Mean Time to Failure (hr)	753012	85.96 Years

Figure 2.0.3 7.2V Switching Regulator (TPS5450-Q1) [10]

MIL-HDBK-217F Class	Monolithic MOS Digital and Linear Gate/Logic Array Devices		
Determination Equation	λp = (C1 x πτ + C2 x πE) x πQ x πL		
Parameter	Description	Value	Comment
C1	Die Complexity	0.04	Logic device with less than 1000 transistors
πτ	Temperature Coefficient	3.075	Ea=0.35 (CMOS Logic), TJMAX=125

C2	Package Failure Rate	0.0025	6-Pin Non-Hermetic SMT
πE	Environmental Constant	4	Ground Mobile
πQ	Learning Factor	1	More than two years in production
πL	Quality Factor	10	Commercial Quality
λp	Failures per 106 Hours	1.33	
MTTF	Mean Time to Failure	751655	85.80 Years

Figure 2.0.4 12V Switching Regulator (LM25085) [11]

MIL-HDBK-217F Class		Monolithic MOS Digital and Linear Gate/Logic Array Devices	
Determination Equation		λp = (C1 x πτ + C2 x πE) x πQ x πL	
Parameter	Description	Value	Comment
C1	Die Complexity	0.04	Logic device with less than 1000 transistors
πτ	Temperature Coefficient	3.075	Ea=0.35 (CMOS Logic), TJMAX=125
C2	Package Failure Rate	0.0034	8-Pin Non-Hermetic SMT
πE	Environmental Constant	4	Ground Mobile
πQ	Learning Factor	1	More than two years in production
πL	Quality Factor	10	Commercial Quality
λp	Failures per 106 Hours	1.36	
MTTF	Mean Time to Failure	731852	83.54 Years

As is shown in the previous calculations, all of the device's most at-risk components are projected to have a mean lifetime in the order of decades. The most at-risk component (the microcontroller) has the shortest lifetime, and while its failure would render the device unusable, the failure wouldn't cause any harm to the user, and ten years seems like a reasonable expected lifespan for a consumer electronic device. The three

high-current devices analyzed (the switching regulators and the h-bridge) all had higher mean times to failure, but their failures are capable of generating considerable heat. Their reliability could be improved through inclusion of redundant parts, multiplexed and controlled by a hardware watchdog which would look for chip malfunctions. While this would add slightly more cost to the device, it would minimize the time during which the device was malfunctioning, and thus would prevent any further damage caused by massive heat dissipation. Seeing as their mean times to failure were all above eighty years, which considerably outlast the microcontroller, it seems equally reasonable to assume that their lifespans are sufficient for applications in consumer electronics.

In order to evaluate the potential failure modes of our device, it was prudent to divide it into smaller, easy to work with blocks. After the division, the circuit was subdivided into battery block, 12V switching regulator block, 7.2V switching regulator block, microcontroller block, h-bridge block, RS-232 block, and sensor block. Appendix F contains FMECA charts corresponding to the information provided..

As a part of the Failure Mode, Effects, and Criticality Analysis data generation process, formal definitions of criticality levels had to be established. The classification of high criticality was given to any failure which held the potential to put the end user in physical danger. In order to avoid these failures as frequently as possible, the acceptable threshold for λp was set as 10-9. The classification of medium criticality was given to any failure which could cause irreparable damage to the device, but would pose no danger. It's acceptable threshold was set to λp = 10-7. Finally, low criticality failures were defined as failures which may limit but not eliminate the device's capabilities. These could tolerate λp values of 10-6 or lower.

6.0 Ethical and Environmental Impact Analysis

Throughout its entire lifecycle, our product, has a very low negative impact on the Environment. In all three stages of our products life: Manufacturing, Use, and Disposal, our project attempts to make as minimal an impact on the environment as possible; however, there are still harmful chemicals used and potentially dangerous metals in components of our project. In the manufacturing stage both the printed circuit board and the integrated circuits require hazardous chemicals to produce. In addition, integrated circuits can require millions of gallons of water that

cannot be recycled. The manufacturing of the other components of our project (motors, chassis) is done using environmentally safe practices. The motor and internals of our chassis are produced by VEX Our solution to these environmental manufacturing issues, if we were to put our product into production, would be to ensure that all integrated circuits are manufactured in a RoHS compliant manner, and ensure that the manufacturers that we choose to fabricate our printed circuit board complies with the same RoHS standards. With a larger budget we would have used Chassis materials that are 100% biodegradable/recyclable, and lighter like tempered aluminum and/or carbon fiber, so as to decrease the power required by the motors further decreasing the negative impact that the drone has on the environment. As a result of our project being electrically powered, the drone runs with zero emissions. Also, we have a large (14.8V 4400mAh) lithium ion battery, the rechargeable battery type with the most charges, that should give our drone approximately one hour of battery life. These two factors, along with our recommendation to users to use an efficient smart charger (since there is no internal charging circuit) that will allow the battery to be used for years before it degrades, will allow for our project to have a very long period of use, making it even more environmentally friendly. All of the materials in our drone, with the exception parts of the printed circuit board and the integrated circuits are completely recyclable. Contrary to popular belief, Lithium Ion batteries are considered non-hazardous waste by the United States Government and can actually be thrown away. However, if lithium ion batteries are disposed of in a landfill the metals from the battery can, over time, seep into nearby water supplies and can cause harmful contamination to the water. Our solution to this environmental issue is to put a label on the the end product that correctly alerts users on proper recycling practices for every component of the product. Overall our project, a kinect powered wireless home security drone that alerts users of a human security breach, has a very low negative impact on the environment.

When designing our product we were faced with many ethical issues. For example, communication with between the client, the control server, and the drone has to be secure. If there is a security breach in any part of this communication line hackers could potentially gain manual access to control of the drone, a floor plan of the room that the drone is in, and a live video feed of the room that the drone is in. As a result we were ethically faced with the challenge of protecting these security threats. Our solution to the security of our drone is multi faceted. First, we are going to require that all users create a unique username and password to access the client application that gives users direct control of the drone. In order to ensure that passwords cannot be stolen we will store them as a cryptographically secure hash. A cryptographically secure hash ensures that it is computationally infeasible to recover the original password from the hash, to accomplish this we are going to implement a

version of the SHA hashing algorithm. In addition to the possibility of usernames and password being stolen there is also the possibility that path of communication from the control client to the drone is breached at some point. In order to ensure that these control packets cannot be read or manipulated, we are going to use an RC4 byte-wise stream cipher to ensure that packets are only readable by the drone. As a result of these two solutions, it would be computationally infeasible to either gain access through the site with a stolen username or password, or try and sniff control packets. One more ethical concern that we face is that this drone can require up to 60 watts of power at a time at four different voltage levels, specifically, the motors can draw up to four amps each at 7.2 volts. This high power requirement is dissipated as heat by most of the integrated circuits. Ethically we are required to ensure that none of the integrated circuits overheat causing a fire or explosion. Our solution to this problem is to limit the the current through the motors using a polyswitch. The polyswitch acts like a “resettable fuse” which will trip when either of the motors draw more than 1.5 Amps of current. This allows us to limit the total amount of current through the H-bridge and basically create a safety shutoff. Also since the H-bridge is the primary source of heat dissipation we are going to use a heatsink to further assist in efficient heat dissipation. Finally we would ensure that all parts are manufactured by RoHS compliant manufacturers that use ethical methods to manufacture all the integrated circuits and printed circuit boards. Overall our design had a couple of major ethical issues, however, we have taken the appropriate measures to ensure success.

7.0 Packaging Design Considerations

The treads of our robot are anchored to the outer side of a 12.5”x12.5” VEX robotics chassis[14], which contains 0.182” thick holes distributed at the frequency necessary to house VEX robotics’ gear kits[14]. Two cross-beams run aside the inside of the square vex chassis, and the VEX 7.2V motors[14] and motor encoders[14] are attached to these beams via the aforementioned VEX robotics holes. A layer of 3/16” thick plexiglass rests on top of the VEX chassis, and is secured to the chassis by screws. Three sets of 4 metal pegs are screwed into the

top of the plexiglass board - one for securing the main PCB, one for securing the power and battery PCB, and one for preventing the battery itself from moving. The two PCBs are secured by screws onto these raised pegs, and the battery is lodged snugly between its four pegs such that it is unable to come loose when further level of the robot are placed. A second layer of plexiglass is suspended directly above the first by a four 4" metal risers, which are attached by screws to both the bottom and middle plexiglass layers. The second level of plexiglass is 3" shorter than the first, with the missing length coming from the back of the robot (the side where the Kinect isn't looking). On the second level are four more metal pegs, for securing the Intel Atom board to the plexiglass. Additionally, on this level are four 6" long threaded metal rods, which are secured by hex nuts to the plexiglass. On top of these is a third and final plexiglass layer, which is also secured to the threaded rod via hex nuts. This level is an additional 3" shorter than the level before it, again, with the missing space coming from the rear end of the robot. On top of this, the Kinect is mounted facing forward. Additionally, there is a hole for wires to come through to the top (for the Kinect and the USB wifi adaptor). Outside of the internal skeleton of the robot is an aluminium sheet metal 'wedge' casing, which protects the robot from potential damage. This case is attached to the threaded rods sticking slightly out of the top layer via hex nuts. Holes are carved into the left side of the aluminum casing, to allow room for the system power button, and the ATOM power reset button.

8.0 Schematic Design Considerations

Our design is made up of five major subsections: power, i2c, PWM, UART (RS232), Input Capture, and ADC. Our circuit requires 4 different operating voltages, 12 volts for the Intel Atom board and the Kinect (each requiring up to 1.5 amps), 7.2 volts for the motors, 5 volts for all integrated circuits other than the microcontroller, and 3.3 volts for the microcontroller. All of these voltages will be obtained by using 4 separate switch mode voltage regulators. To drive all of these voltages we will be using a 14.4V 5200mAh Lithium Ion battery that is made up of eight total cells, four in series and two in parallel. Our initial design was to provide an unregulated 14.4V to the motors, this would have been too close to the maximum rated voltage of the motors of 15V. As a result, we decided to provide the motors with a regulated 7.2 volts from the 14.4 volt battery, we chose 7.2 volts because that is the actual rated voltage of the motors. We will not include a re-charging integrated circuit because we were unable to find one that suited the size of our battery. Because of this we will use a commercial charger and have 2 separate

printed circuit boards, one that contains most of our microcontroller circuitry as well as the 12, 7.2, and 5 volt power regulators, and another board that has the fuel gauge circuitry and the 3.3V regulator. The reason for this secondary board is due to the fuel gauge, the gauge has to be connected to the battery during charging and discharging in order to maintain accuracy, and this particular fuel gauge integrated circuit operates at 3.3 volts. As a result of these two factors we need to create board with both the fuel gauge on it and the 3.3 volt regulator that will stay permanently attached to the battery. Both of these boards will be connected via a barrel connector that provides the unregulated 14.4V for the other regulators as well as the regulated 3.3 volts, ground, and the SCL/SDA lines used by i2c. The i2c module of our design will be used to receive data from the fuel gauge. We also had to use a 3.3 to 5 volt logic converter for the H-Bridge control pins because the H-Bridge operates at 5 volt logic. The H-Bridge will take in the PWM signals from the microcontroller and the control data from the GPIO pins and accordingly move each motor forward or backwards at the appropriate speeds. We will be using the UART module to communicate with the Intel Atom Board via RS232. This communication will consist of control data sent to the microcontroller from the Atom designating speed and direction of both motors, and the fuel gauge and tachometer data from the microcontroller to the Atom. The input capture module of the microcontroller is used by the motor tachometers. Each tachometer is attached directly to the drive shaft on each motor and produces 360 active high pulses on every rotation of the motor. We will be using five ADC inputs that will read in analog inputs from our five infrared sensors, each of use 5V. As a result we will be providing our microcontroller with an analog reference voltage of 5 volts.

For our microcontroller we specifically chose the smallest package available, a 64-pin package, because our subsections require very few pins. We will be using four subsections of the microcontroller, the PWM, the ADC, i2c, and UART. for our ADC we will be using pins: AN5, AN4, AN3, AN2, AN1, and AN9 (pins 11-14, 22). This is because the other available ADC pins conflict with the pins used by the PIC in-circuit debugger. Specifically the ADC will be used to take in data (10 bit resolution) from the infrared sensors that will be placed in the front and rear of the drone. After testing the ADC the drone will be able to detect an object up to approximately 15 feet away. The microcontroller will use this data to give the Atom information about nearby objects and how close they are via the UART (RS232) interface. We will be using the UART1 pins (U1TX, U1RX) which will be connected to the RS232 level converter, allowing communication between the microcontroller and the Atom. We chose these pins because they do not interfere with any of the other microcontroller subsections that we are using and they are separated from the other pins being used on the physical integrated circuit. The PWM we are using has a resolution The RS232 level converter

requires 5 volts to operate so it will draw power from the 5 volt regulator. The PWM has a resolution of 16-bits and will be used to control the speed of the motors through the H-Bridge. The PWM only requires two pins to control the two motors, one for each motor; the PWM pins that we are using do not conflict with any of the other modules we are using and are also close to the GPIO pins used to control the H-Bridge. The 4 GPIO pins that we plan on using are RE4, RE3, RE2, and RE1. These were chosen because they are located near the PWM pins which also run to the H-Bridge. The i2c subsection of our microcontroller is used to collect fuel gauge readings that will be sent back to the Atom Board. The i2c pins chosen are in the first i2c module (SDA1, SCL1), this is due to a conflict with the second i2c module and one of the UART lines that we are using. The microcontroller operates at an i2c logic level of 3.3 volts which is directly compatible with the fuel gauge because both operate at 3.3 volts. All of the extra unused pins on the Microcontroller will be padded out to headers in case they are necessary after the PCB has been designed. The power circuitry was specifically difficult to design. In order for the Fuel Gauge to maintain accuracy it has to be attached to the battery during charging and discharging. However we will not have an internal charging circuit so the battery will have to be removable. As a result, we will have to create a PCB separate from the main PCB that contains the microcontroller, H-Bridge, level converters and headers. The fuel gauge PCB will also contain the 3.3 volt regulator because the Fuel Gauge IC requires a regulated 3.3 volts, the other 3 regulators will be on the microcontroller board. This results in an interconnection between these two boards consisting of a cable that provides the unregulated voltage from the battery to the microcontroller board through a barrel connector, as well as the 3.3 volt regulated output and i2c headers on both boards. This four wire bus is necessary for the microcontroller to receive 3.3 volts from the regulator on the battery PCB as well as i2c data from the Fuel Gauge.

9.0 PCB Layout Design Considerations

There were several design choices to be made for the overall PCB layout. The first thing that was considered was the placement of large functional blocks on the PCB. On the main PCB there are three linear switching regulator power supplies which generate a lot of noise and handle a lot of current. These circuits need to be placed away from the more sensitive analog and digital circuits. In addition, the H-Bridge

circuit should be placed far from the digital and analog lines due to the large amount of current that runs through it. The position of the headers also had to be considered when placing functional blocks. There had to be enough room on the edge of the board for any circuit's headers or connectors. On the main PCB, there are 3 barrel connectors for an unregulated battery input,12 volt output, and 3.3 volt input. There is an RS232 connector, 0.1" headers for connection to sensors and the other PCB, an RJ-11 connector for programming the microcontroller, and 0.1" headers for connections to the motors. For the main PCB, it was decided that all of the power supply circuits and H-bridge would go towards the bottom of the PCB while the microcontroller, digital logic ICs, and analog sensor lines would go towards the top. Specifically, the analog IR sensor headers would go in the top left of the board, the micro would go towards the top center, and the RS232 circuit would go in the top right. The overall size of the main PCB is 5 inches x 4 inches.

After the position of the functional blocks was determined, several other things had to be considered. First, some circuits have a lot of current flowing through them, so traces need to be wide enough to accommodate this. Polygon copper pours were used where possible to reduce the noise produced by a circuit and a ground plane was used on the bottom copper layer.. The linear switching regulators are particularly picky about layout, so extra care was taken to make sure that there were copper pours where possible and the all of the components were as close to each other as possible. There should also be no acute angles in any traces or right angles in any data lines. The minimum recommended trace width is 12 mils, so that's what was used for data lines. However, thicker lines were used wherever possible. Additionally, 100 mil traces were used for power wherever it was possible. There is no real size constraint on the main PCB, so the space that traces occupy was not a large concern. The battery PCB board has very few components, so traces could be make as large as necessary. There was actually excess space available when all of the parts were laid out on a 2 inch by 2 inch board which was the target size for the board. This extra space was filled with copper pours, and a ground plane was put in to reduce noise. All of the current that flows from the battery goes through this board, so close attention was paid to the width of the traces. All external connections to this board are made with two pin 0.1 inch spaced headers.

The microcontroller chosen for this board is the PIC24FJ129GA006 which comes in a 64 pin TQFP package [8]. The first consideration in laying out the microcontroller on the PCB was where to place it on the board relative to all of the other components. At first the microcontroller was placed in the exact center of the board, but it quickly became apparent that several of the other ICs needed to be closer together than that layout would allow. Therefore, the microcontroller was moved closer to the IR sensor headers in on the left side of the board.

This allowed the level translator chip and the RS232 level converter to be closer together which made connections between them easier. Moving the microcontroller over also had the added benefit of being closer to the analog signals coming from the IR sensors. This will mean more accurate readings as they the traces will be subjected to less noise. The analog lines from the sensors are also decoupled with capacitors to obtain more accurate readings. The datasheet recommends decoupling capacitors for the analog to digital signals because the analog to digital channels themselves produce significant noise. Another important aspect of the microcontroller layout was the ability to access all of the pins of the microcontroller either to gain access to unused pins or test the status of signals coming out of the microcontroller. All of the pins on the microcontroller have been padded out using four sets of two row 0.1 inch spaced headers. This configuration saves a lot of space which allowed every pin on the microcontroller to be padded out as opposed to just the ones that are not being used. One challenge that was faced when laying out the microcontroller was routing the power and ground rails to the center of the microcontroller. This was because the headers are through hole and take up space on both sides of the board. To solve this the headers were moved back from the micro and the power and ground traces were run between them. Another major consideration when laying out the microcontroller was the location of bypass capacitors. In order to save space, the bypass capacitors were placed under the microcontroller on the bottom side of the board. The power and ground pins of the microcontroller were connected to the other side through vias on the inside of the chip. The 3.3 volt and ground rails were routed to the center of the microcontroller on the bottom plane. There is also a bypass capacitor for the 5 volt A/D reference pin on the microcontroller. Unfortunately, this trace had to be cut in the end because it was discovered that the reference pin was not 5 volt tolerant. The last consideration was that the microcontroller had to be rotated in such a way that the pins were closest to where they had to be routed. The optimal orientation was with the first pin towards the top of the board.

There were many factors to take into account when laying out all of the power supplies. The first was the placement of all of the power supplies. The fuel gauge IC needs 3.3 volt power so the 3.3 volt regulator was placed on the battery PCB. This circuit uses the TPS62160 linear switching regulator IC for which there is a suggested layout in the datasheet. This layout was followed very closely except for one resistor which was not in the suggested layout [27]. All of the other power supply circuits are located on the main board. The 12 volt source is in the bottom right corner of the board because it does not power anything actually on the board, but rather powers the Atom motherboard and the Kinect. The 7.2 volt source is located at the bottom of the board next to the h-bridge circuit. The 7.2 volt source and 12 volt source were

placed far from the actual digital logic circuits to reduce interference. Finally, the 5 volt source was placed in the center on the board. The 5 volt source uses the same IC as the 3.3 volt circuit and, as such, has a very similar layout.

All of the power supply IC datasheets had requirements for PCB layouts. Only one of the IC datasheets actually had a suggested layout. The 12 volt source uses the LM25085 switching regulator [11]. It uses the most external components of all of the other switching regulators. The datasheet specifies that the parts should be as close to each other as possible and the traces should be as wide as possible. In the layout of this circuit, copper pours were used wherever possible and the components were placed as close to each other as possible. Some of the 1206 type components were placed on the backplane to ensure that everything was as close as possible. The output of the 12 volt power supply goes directly off the board to through a barrel connector to the Atom motherboard and Kinect. The 7.2 volt power supply uses a TPS5450-Q1 switching regulator IC. The requirements for this circuit were almost the same for this circuit as the 12 volt one. All components need to be as close as possible to one another, and traces should be as wide as possible. As with the 12 volt source, all components were placed as close to each other as possible and copper pours were used extensively. This is especially important for this circuit because it has the potential to source a lot of current. One tricky aspect of this particular IC is that it has a thermal pad on the bottom to aid with heat dissipation. This pad needs to be tied electrically to ground. The datasheet also recommends placing vias in this pad to further aid in heat dissipation, so two vias were added to the underside of the IC [10].

The actual power traces that deliver power to the loads on the board had several considerations as well. The traces carrying power throughout the board were 100 mils where there was space to minimize the resistance and inductance of the lines. For the most part, power and ground rails go directly to where they need to go to minimize the length of the traces. The ground plane helps immensely with noise reduction on ground. The exception to this is the 5 volt rail that gets routed to the IR sensor headers which had to be routed around the microcontroller. The analog sensors get their own power and ground rails to minimize noise on the traces. Additionally, decoupling capacitors were placed next to every IC as close to the IC as possible to subdue any noise on the power and ground lines.

10.0 Software Design Considerations

To implement this design, the team has decided upon a four tiered software stack. The four tiers are best separated logically through by their physical processing location, and the tiers are best ordered by their distance from the hardware at the bottom of the tier. From lowest level to highest level the tiers are A) a PIC24 microcontroller for GPIO and ATD connectivity, B) an atom board for utilizing complex path finding algorithms and interfacing with the Xbox Kinect utilizing Ubuntu Linux, C) a command and control server which doubles as a webserver for the user interface running on an x86 PC utilizing Ubuntu Linux, and D) the user's web browser which will serve as the primary interface for the robot. Please see appendix B for a visual representation of these software tiers.

The MicroController code in layer (A) will be written in C for the purpose of ease of development and code organization. Because of its simplicity this program will be written using an interrupt flag driven event loop. This organization will cause the program to behave in a reactionary manner. This fits the purpose of the microcontroller well as its primary function will be reacting to commands registered by the atom board over RS232. These commands implemented for use on the microcontroller are motor control, battery level retrieval, IR sensor polling, and motor encoder polling. Another primary feature of the microcontroller includes a "fail-safe" stop feature that will monitor the IR sensors on the bottom and sides of the robot and keep it from colliding with walls or falling down stairs. Because both of these functionalities rely upon inputs that can be caught by interrupts, the interrupt driven flag event loop is a perfect fit for its code base. The interrupts used by the microcontroller include the i2c interrupt, the adc interrupt for the IR sensors, UART TX and RX interrupts for RS232, and two timer interrupts for debugging and capturing the state of the motor encoders.

The software running on the Atom board within layer (B), which we will refer to as the Intelligent Drive System (IDS), is significantly more complicated, and will therefore be implemented in the C++ programming language. This layer of the software stack has many responsibilities including:

1. Retrieving depth frames from the Kinect for navigation
2. Retrieving video frames from the Kinect for the user display
3. Transcoding video frames into a YUV based format from the initial RGB format
4. Maintaining spatial coordinates on a two dimensional floor map
5. Navigating around objects within the room to set waypoints when in navigation mode
6. Reacting to commands via a TCP network socket from the C&C server in layer (C)
7. Polling for and monitoring sensor data from layer (A)

Because of these many processor intensive responsibilities we have decided upon a much more complicated code organization which will allow us to take full advantage of the dual-core processor on the Atom board. The Intelligent Drive System will implement a dualistic control flow separated into two primary threads and two "listener" threads. . The first primary thread will be a timer driven synchronous loop. The main loop here runs at 24 iterations per second because it is responsible for outputting the 24FPS video stream. This loop will handle responsibilities 1-3 which are less time sensitive, but will still take up a significant amount of processing power. The two helper threads will be running on the same core as the video output thread and handle responsibilities 6 and 7 for communicating with the adjacent layers of the software hierarchy (A and C). The second primary thread which handles responsibilities 6 and 7 is organized as a queue based event loop. The team developed a queue of commands which will be serviced by the loop. The command data structure contains a virtual method "action()" which allows it to implement the strategy design pattern. Commands are added to this queue through timers which will allow for a heightened flexibility for the frequency of execution of tasks that may have varying degrees of time sensitivity. Commands are also added to the queue by the helper threads in order to serve as an inter-thread communication pipeline. Additionally, the action( ) member accepts a parameter which is a singleton class used to represent the overall program named IDS. This singleton class provides access to an instance of the singleton Kinect class which composes frame dumps and processed video output over to primary thread 2 upon request. The IDS class also provides access to a class named Minotaur which contains current positional reference information within the 2d floor plane. Finally, thread

one will be outputting its video feed onto stdout, which will be piped into the open source program AVConv (formerly named FFMpeg) which will package the video frames into a video container and transmit them to the AVServer (formerly named FFServer) running within layer (C) via the FFStream feed (not renamed) protocol. Additionally, the goal of integrating human detection into the IDS program was not achieved due to conflicts between the library chosen to implement it (OpenNI) and the library chosen to implement raw data access from the Kinect sensors (LibFreenect). This feature was therefore confined to a secondary program which accomplished the PSSC associated with it.

One technique used within the IDS in layer (B) deserves extra attention. Although the team was unable to complete the room mapping algorithm in time for this report due to problems with interpreting motor encoder data, some of the more advanced features of the room mapping software were successfully implemented, though they have not had a chance to be integrated into the final design of the robot. In order to accomplish wall detection for use in the unfinished room mapping feature, the attempts to identify groups of pixels within the Kinect's depth sensor image which correspond to walls within the patrollable area. The patrol design called for using the walls within the room to reduce the accumulating error created by the motor encoders. To this end, the WallFrame command maps each pixel of depth data to a 3D coordinate system with the Kinect as its reference point (See X,Y,Z definitions below). The command then groups the points generated by an 8x8 block of pixels and executes a linear regression to identify their slope in the y/x, z/x, and z/y planes. The command then uses the z/x and z/y slopes in conjunction with the Z-variance to identify which 8x8 blocks belong to vertical walls or horizontal floor planes. The command can make immediate use of the floor planes restricted to certain heights below the Kinect to identify floorspace that is valid for travel and use it to avoid obstacles. The more algorithm deals with vertical wall detection. In order to detect walls suitable for orientation change detection, the command creates a 2d matrix of counts of 8x8 blocks whose y/x slope and distance from the Kinect are similar. We then use this 2d 256x256 array of counts of slopes x distance and run it through a fourier transform and low pass filter. This functionality allows us to find significant walls to use in position detection. In order to find the center of the cluster we simply find the top of the gradient within each cluster. We calibrated the magnitudes returned by the low pass filter to pick out significant walls that are currently within sight of the kinect. The result is a list of highly visible walls organized by their slope and y-intercept within the y/x plane. By comparing this list between iterations of the command we are able to detect changes within in position of the robot.

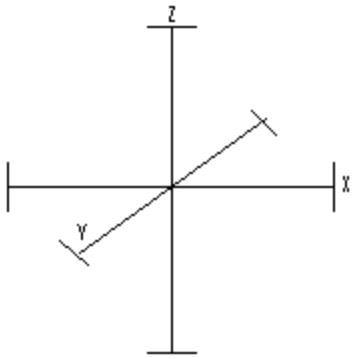


Figure 10.1 - 3D Dimensions as seen by Kinect

Layers (C) and (D) are tightly coupled, as the Command & Control server will double as a web server which serves layer (D) to the user's web browser. Software within layer (C) is split into three distinct pieces: the Apache webserver, the AVServer/FFServer video streamer, and the C&C server which will be written in Python. Being open source projects, Apache and AVServer/FFServer should require no more explanation within this documentation. The user's web browser in layer (D) will build its user interface via HTML5, CSS, and JavaScript loaded from the Apache server (C), and uses a VLC plugin to display the live video feed from the AVServer (C). Additionally, the client leverages the WebSockets library to create a communication stream between the C&C server in layer (C) and JavaScript in layer (D). WebSockets allows for direct TCP socket communication over a TCP/IP connection from within JavaScript. The WebSockets library within JavaScript is a great help to this project because it allows for communication with the webserver without invoking the overhead of the HTTP protocol usually associated with JavaScript network communications. The C&C server in layer (C) uses the Python Twisted library to implement a threaded socket server loop that waits for incoming connections on a TCP port from the robot and web client. The server then retrieves commands from the client connection and forwards them to the proper Minotaur robot. One could argue that the introduction of layer (C) into the software hierarchy is superfluous because the webserver and C&C server could be implemented directly within layer (B), but there are several good reasons for its implementation. Firstly, layer (C) moves the burden of serving potentially numerous user connections onto a more powerful machine. This allows the project to incorporate more advanced methods of authentication, firewalling, and user management into

the C&C server design. It has an additional security benefit of keeping the physical robot controlled by layers (A) and (B) from being directly exposed to the Internet.

The team has also made several features available within the design for debugging purposes. This "design for test" mentality should help us to quickly troubleshoot problems within our complex software stack. Hardware debugging features include a reset pushbutton, a debugging LED, exposure of the RJ11 interface of the microcontroller so that it can be programmed directly with the ICD3 in-circuit debugger, and the breaking out of micro controller pins to various headers. Our microcontroller debugging is also assisted via communication over the RS232 port, which will allow us to output detailed debugging information. Layer (B) also has several debugging features implemented such as an SSH server and an ability to dump log files and performance metrics to layers (C) and (D) via a TCP connection.

Within the software hierarchy above, the Low Level Controller (LLC) of layer (A) can be broken down further into individual code modules. The primary code modules of the LLC include the I2C fuel gauge driver (incomplete due to circuit problems), the IR sensor bank timer ISR, the RS232 driver, the motor encoder sensor ISR, and the motor motion controller. The I2C fuel gauge driver was supposed to be initiated upon requests over RS232. The driver polls the fuel gauge via I2C, and then awaits another interrupt representing the I2C response. Once the fuel gauge responds, the micro controller would respond to layer (B) via RS232. The IR sensor bank's IR sensors will utilize the ADC module of the microcontroller and trigger whenever a capture compare completes. The interrupts captures each of the IR sensor's readings and store them into a variable in which each sensor's "tripped" status is represented in one bit of five. This value is returned to layer (B) via RS232 upon request, and will force a "stop" fail-safe for the motors unless overridden by layer (B). The RS232 driver will be set various flags whenever the "incoming" interrupt triggers. If the request is for a piece of immediately accessible sensor data it also responds to the request by responding over RS232 with the requested data. The motor encoder driver increments a counter via a GPIO interrupt which represents the current distance driven by the motor encoder and store the count into a global variable. The motor controller then utilizes current speed settings from a global variable after checking the "force stop failsafe" variable. It then relays these speed settings to the motors on each iteration of the loop, if they have changed. Finally, if the microcontroller has not received any commands via RS232 within some specific timeout window, the microcontroller will initiate the "failsafe stop" sequence to avoid a runaway robot.

The Intelligent Drive System (IDS) of Layer (B) can also be broken down into code modules. Within thread one the duties of

capturing depth sensor and image sensor data is completed via a callback within the Freenect library. These callbacks retrieve the data via USB from the Kinect and store it into a memory buffer. The task of converting the video from RGB format into the YUV frame format is also in its own function and has been completed. Polling for sensor data from the LLC over RS232 was completed and is updated 5 times per second.. The TCP client which connects to the C&C server is maintained within the main loop of a helper thread at a rate of 30 Hz.

11.0 Version 2 Changes

When constructing the drone, we decided to use the combination of a high-power low-speed gear ratio, VEX motors, and treads to allow for movement of the robot. In the future, we might replace these with a higher-speed wheel-based system with a weaker gear ratio in order to accomplish speeds more comparable to a remote-controlled car. On the PCB, we would utilize different 12V switching regulators and fuel gauges, as the two that we selected never fully functioned, and we needed to find workarounds. The RX and TX lines on the RS-232 to microcontroller would be switched, as they were incorrectly assigned (inconvenient, but not detrimental). Since the h-bridge could run off of 3.3V logic, we would remove the 5V-3.3V level converter, and find an RS-232 differential level converter that could function at 3.3V. Finally, we would incorporate the polyswitches from the H-Bridge into the circuit board, and we'd merge the two PCBs into a single PCB for the sake of convenience.

12.0 Summary and Conclusions

Overall we were able to accomplish our goal of designing a wireless home security drone that uses a Microsoft Kinect to give users a live video feed, mapt the room its in, and avoid objects. The drone wirelessly interfaces with a command and control server through the internet that relays commands from the end user client. We were able to successfully design two printed circuit boards that control the lower level functions of the drone (motor speed/direction, rs232 communication, IR sensor input) and interface these boards with an Intel Atom board that was also interfacing with the Microsoft Kinect. In addition to successfully attaining functionality of the drone we were able to create a sturdy and visually

appealing package for the drone. Through this design process we, as a team, learned how to design a microcontroller based project that interfaces with a fully functioning web application.

13.0 References

1. M. Goosey, "Environmental best practice in the PCB industry," Circuit World, Vol. 26 iss: 3, pp.30-33

2. Secure Hashing Standard, Federal Information Processing Standards Publication 180-1, 1993.

3. RC$, Public-Key Cryptography Standards PKCS #7, 1991.

4. A. Kak. (2013, February 26). *Public-Key Cryptography and RSA* [Online]. Available: http://engineering.purdue.edu/kak/compsec/NewLectures/Lecture12.pdf

5. A. Kak. (2013, March 21). TCP Vulnerabilities: IP Spoofing and Denial-of-Service Attacks

6. Aavid Thermalloy. (2013). *"Heatsink Multiwatt, SIP"* [Online] Available: www.aavid.com/products/standard/566010b03400g

7. Department of Defense. (1990). MIL-HDBK-217F. [Online]. Avaliable: https://engineering.purdue.edu/ece477/Homework/CommonRefs/Mil-Hdbk-217F.pdf

8. Microchip. (2005). PIC24FJ128GA Family Data Sheet. [Online]. Avaliable: http://ww1.microchip.com/downloads/en/devicedoc/39747a.pdf

9. ST Microelectronics. (2000). L298 Dual Full Bridge Driver. [Online]. Avaliable: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/CD00000240.pdf

10. Texas Instruments: (2012). 5-A, WIDE INPUT RANGE, STEP-DOWN SWIFT CONVERTER. Avaliable:

http://www.ti.com/lit/ds/synlink/tps5450-q1.pd

11. Texas Instruments: (2012). LM25085 42V Constant On-Time PFET Buck Switching Controller. Avaliable: http://www.ti.com/lit/ds/symlink/lm25085.pdf

12. Texas Instruments: (2012). 8-BIT BIDIRECTIONAL VOLTAGE-LEVEL TRANSLATOR. [Online]. Avaliable: http://www.ti.com/lit/ds/symlink/txb0108/pdf

13. Texas Instruments: (2012). Wide Range Fuel Gauge with Impedance Track Technology. [Online]. Avaliable: http://www.ti.com/product/bq34z100

14. Vex Robotics: (2013). [Online] Avaliable: http://www.vexrobotics.com

15. Atmel. (2011). 8-bit Atmel Microcontroller with 128KBytes In-System Programmable Flash [Online]. Available: http://www.atmel.com/Images/2467S.pdf

16. Advantech. (2013). MIO-5250. [Online]. Available: http://www.advantech.com/products/MIO-5250/mod_EEA5E107-B1E6-4989-A8E0-03A042F5DBEF.aspx

17. NORCO. (2012). BIS-6630. [Online]. Available: http://www.norco-group.com/products_detail/&productId=27b4a88d-0404-4a48-8727-76f584f834a4.html

18. OpenNI The open framework for 3D sensing, OpenNI, [online] 2013, http://www.openni.org/ (Accessed: 22 March 2013).

19. The WebSocket API, W3C, [online] 2013, http://dev.w3.org/html5/websockets/ (Accessed: 22 March 2013).

20. Welcome, OpenKinect, [online] 2013, http://openkinect.org/ (Accessed: 22 March 2013).

21. FFMpeg Documentation, FFMpeg, [online], http://www.ffmpeg.org/documentation.html (Accessed: 22 March 2013).

22. HTTP Server Project, Apache, [online] 2013, http://httpd.apache.org/ (Accessed: 22 March 2013).

23. Sharp. (2006). GP2Y0A02YK0F. [Online] Available: http://www.sharpsma.com/webfm_send/1487

24. Gutmann et al., "Method and Apparatus for Detecting Plane, and Robot Apparatus Having Apparatus for Detecting Plane," U.S. Patent 0 257 910 A1, Nov. 08, 2007.

25. Honeywell International Inc., "System and method of extracting plane features," European Patent 2 166 375 A2, Mar. 24, 2010.

26. Fujitsu Limited, "Robot system and autonomous mobile robot," U.S. Patent 7 218 993, June 15, 2007.

27. Texas Instraments: (2012). 3V-17V 1A Step-Down Converters with DCS-Control TM. Available: http://www.ti.com/lit/ds/symlink/tps62160.pdf

Appendix A - Individual Contributions

A.1 - Contributions of Neil Kumar:

For this project I was most heavily involved with the Hardware, Packaging, and Embedded system design. I helped Scott find our microcontroller and other parts for each of the subsections of the microcontroller circuit (IR sensors, H-bridge, RS232 converter, level translator, fuel gauge, battery, motors, chassis). I helped design the overall power architecture of the design using the TI WebBench to find IC's for the 4 regulated voltage levels we required. After we, as a group, determined the overall architecture of the project I helped to design the circuits and subsystems for each part of the microcontroller. I put together the schematic for our Main Printed circuit board and helped to create most of the parts in eagle for the PCB layout. After finishing the schematics I created the layout for the battery PCB which included the fuel gauge and the 3.3 volt source, and assisted Scott with the layout for the Main PCB. After Helping to finalize the PCB I assisted Scott in prototyping and testing some of the subsystems of the microcontroller circuit (H-Bridge, Motors, IR Sensors) using the PIC microcontroller Development board. I also helped to write some of the embedded code for the ADC IR sensors as well as the H-Bridge and i2c fuel gauge. When the PCB came back from fabrication I did most of the soldering of on both of our boards and helped to create many cables used by the sensors and motors. When the PCB came in we had many issues with getting the microcontroller programmed, and burned out multiple microcontrollers and IC's in the process; I was heavily involved with the removal and replacement of these multiple microcontrollers and Integrated Circuits. Through this process I was also heavily involved in debugging both printed circuit boards, I helped Scott to find out that the common port on the power source was actually 12.5 volts above ground which was causing many of our microcontroller issues. I also helped debug the UART module where we found that the RX and TX lines were accidentally switched, and also helped find that the H-Bridge was accidentally using ground as its Digital Logic Voltage and thus not working. I helped to fix these errors by fly wiring cables on our board. I also wrote all of the i2c code, we were able to successfully send the command to recieve the fuel level from the battery. However, since we did not have access to the proprietary fuel gauge development module we were unable to initialize the fuel gauge and finally unable to use it overall. In

addition to helping design, fabricate and test the microcontroller and all of its subsystems I was heavily involved with the actual construction of our chassis and packaging. I helped to drill the holes in the plexiglass sheets to mount the three printed circuit boards (battery, microcontroller, atom board). I was able to assist in cutting, bending and drilling the aluminum sheet that we used to develop the shell that housed all of the electronics. In addition to fabricating the shell I painted it matte black and painted the orange logo on the back. Though I did not write a lot of the software I was heavily involved in debugging both the embedded software as well as the higher level software. I also created the template which was used for our Team website and helped to construct the layout of the end user client for the actual project.

A.2 - Contributions of John Hubberts:

Within this project, my general focus was in high level software design and support, as well as packaging design, though I additionally lent some assistance in debugging hardware and embedded software issues. I assisted in the decisions on which microcontroller to select based on the stipulations of our project, as well as the VEX components which comprised the chassis, motor encoders, treads, and h-bridge. In addition, I worked with Jon Roose to select an ATOM motherboard that would have low enough power consumption to give us sufficient battery life, but a high enough clock speed and computational parallelism to allow us to run complex graphical computations (Kinect-to-image encoding and transmission) in tandem with control statement handshaking with the command and control server.

The bulk of my work was put towards furthering the functionality of the high-level code (ATOM board, command and control server, the web client, and their interactions). Jon Roose and I worked together to set up the code hierarchies, and lay out a preliminary block diagram. After spring break, when our ATOM board came in, myself and Jon installed 32-bit Ubuntu 12.10, configured it to run without GUI (so-as to reduce the processing overhead), and set up an ssh server so that we could remotely access the board. I took Jon's prototype of the web-client, and added some additional functionality, including speed control and direct user feedback of packet transmissions for debugging purposes. The two of us (in conjuncture with Scott on the embedded systems side) developed a packet protocol for sending data between the various layers of the software. I encoded the fields of the web-client into a JSON object, and configured python JSON libraries to take that data, and decode it on the side of the command and control server. Jon Roose and I took this object, and configured directional movement

analysis to translate keystrokes or sets of keystrokes {W, A, S, D, Space (brake)} into motor magnitude and direction translations. After this was configured, I spent time with Scott Stack debugging issues in the bidirectional RS-232 communication, and modifying mine and Jon's code accordingly.

In addition to this, I built a few c++ libraries that were used for various high-level functionality. I constructed the library for conducting and decoding/encoding RS-232 transactions with the microcontroller (as used by the IDS running on the atom board). I also built a library for sending time-stamped warning emails to a user after an intruder was detected in the premises. I used the OpenNI library to modify code that would display and detect skeletal forms seen by the Xbox kinect. This code was stripped of all graphical calls (from OpenGL) so-as to reduce the overhead. Finally, I injected the aforementioned email warning library calls into the skeletal detection events, so that the detection of human forms would result in a message being transmitted from the drone to a specified email address, through a gmail address configured particularly for the drone.

A.3 - Contributions of Scott Stack:

My focus on this project was all of the hardware, embedded software and packaging although I also helped out with a little bit of the high level software. During the early stages of the projects I chose many of the parts that were going to be used on our PCB. This included the H-Bridge, microcontroller, IR sensors, RS232 level converter, and battery fuel gauge. Neil and I both selected the motors, chassis and battery that the robot would use as well. Using Ti's WeBench tool, Neil and I designed all of our 4 power supplies that we would need. I helped Neil complete a good portion of the schematics for our PCBs. I created about half of the parts in Eagle and Neil did the other half. Once the schematic was done, I designed the main PCB board. This required building several more parts and finding connectors and passive components. Laying out the three power supplies on the main PCB was a challenge because the parts have to be so close together.

In addition to designing the main PCB, I wrote almost all of the embedded code for the microcontroller. I wrote the code for most of the main functions that our board would be performing. I wrote code to initialize and run the PWM for the motors, the A/D conversion for all of the sensors, the UART transmission to the ATOM board, and the two timers that would be used for updating the motor encoders and timing

out commands received from the ATOM board. The code ended up being a kind of hybrid between flag driven and interrupt driven architectures. Any non time sensitive functions would set a flag in their ISR and be serviced in a main polling loop, but time critical functions such as UART and motor encoder logic had to be serviced in their respective ISRs. I wrote a circular buffer for the UART transmission and receiving so that everything could be processed without data loss. I also wrote the code to implement the protocol that would be used to communicate between the ATOM board and the microcontroller as well as basic obstacle avoidance using the IR sensors.

Once the PCBs came in I helped populate both of the boards. Neil and I worked together to get all four of the power supplies working, but in the end we were only able to get three of them working. After the power supplies were working, Neil and I populated the rest of the boards. I worked closely with Neil to debug all of the problems with our PCB and get it working. We went through about 5 microcontrollers because the bench DC power supply that we were using to power the board had a common that was 12.5 volts above ground, so whenever we would plug in anything that was earth ground the board would fry. I helped create a bunch of the cables and connectors that the robot would use as well. There were only a couple things that needed to be fly wired on our board (for example we switched our RX and TX lines going into the microcontroller).

Once the PCBs were both put together and working, Neil and I built the packaging for our robot. I got several sheets of plexiglass cut at the machine shop to use as our layers of the robot. Neil and I constructed the robot using these plexiglass sheets and standoffs between the layers. We drilled holes to mount all of the electronics and the battery on. Neil and I also got metal sheets that we had machined into an outer shell for the robot. Neil and I spray painted the shell to make it look a little better. I helped make all of the final cables and switches that needed to be made for the final packaging as well.

Once all of the hardware was in place I took the basic website that Jon built and helped make it look better by organizing it and adding colors, instructions, settings, and more functionality to the page.

Appendix A.4 - Contributions of Jon Roose:

Over the course of developing project Minotaur I was primarily responsible for implementing the high level software stack (excluding human detection), but I also assumed the role of software technician within our group. Some of the primary features of the software design that I implemented were the C&C server, the IDS program, the web interface, Kinect camera calibration, conversion of depth data into 3d coordinate systems, and conversion of motor encoder values into X/Y coordinates within the floor plane and orientation changes. I also implemented prototyping software for wall detection and final wall detection software utilizing 2 dimensional fourier transforms and low pass filters. In terms of software infrastructure I also set up the Atom board and C&C servers using minimalistic and full Ubuntu implementations respectively. I was also in charge of configuring and installing the AVServer implementation on our systems along with supporting software for Python, websockets development, and C++ libraries for fourier transforms and pthreads. I chose and implemented all of these libraries.

Much of the first half of the semester was spent displaying video from the Kinect into the VLC software used by the client. A significant portion of the challenge was eliminated through our use of the AVConv/FFMpeg software package that is designed to transmit video from raw input sources, transcode it, and make it available over the network to end users. Setting up the video transcoding was a laborious task that required the implementation of an RGB to YUV video conversion process within the IDS before it was piped back out to AVConv via stdout. It was unforunate that we had to utilize stdout in order to communicate with AVConv, but setting up fifo nodes within the kernel or finding alternative methods of communicating with the software package for live video would have greatly increased the work requirement for its administration. I also spent a significant amount of time researching methods of live streaming videos (specifically checking into various protocols) before settling upon the RTSP protocol which was supported by AVConv. Using this method we were able to achieve a video latency of less than 1.5 seconds over a Wifi network link.

In terms of the IDS software I planned and implemented the overall architecture of the software for our design, with input on some of the big picture specifics from the rest of the group. I made extensive use of my knowledge of object orient programming and design patterns in order to create the best possible design for the IDS architecture and facilitate interthread communication. To this end I made extensive use of the strategy and singleton design patterns in order to keep the code as modular as possible. I also did some extensive research into Makefile

design in order to make the build process as painless as possible while working with a myriad of object files for each of the required classes. To this end I made effective use of the G++ -M option for dependency generation.

I worked closely with John Hubberts in implementing the C&C server interface for the webserver communications. I discovered an optimal method of detecting key presses within the client's web browser and sending them to the C&C server and made the decision that the C&C server should directly control the command packets sent to the robot based on the key inputs sent to it by the user interface. This enhanced security by allowing us to have better control over what data and commands are sent to the robot. I also discovered, with the help of John Hubberts, the twisted packages which allowed us to easily implement a webclient/IDS communication link by connecting raw TCP communications to the websockets client used by the user's web browser.

In terms of setting up hardware, I was tasked with enabling the Atom board. This quickly became a significant challenge because the graphics drivers used by the atom board were not available to Ubuntu. Rather than spending months identifying the problem with the graphics, I decided to use a separate computer to set up the flash drive by booting into it with my laptop. There I was able to make the necessary configuration changes and install an SSH server that allowed us to control the Atom board remotely. I also made use of the cron @reboot function to implement the network interface and run the IDS when the Atom board booted. I also enabled the automatic fixing of errors on boot via FSCK on the file system for the purpose of removing interaction from the bootup process. Finally, I was tasked with configuring the Atom board for WPA wifi through the command line on Purdue's wireless network.

Finally, I spent much of the last half of the semester working on wall detection and position coordination. I accomplished the wall detection with the help of the fourier series algorithm detailed in the software narrative in section 10.0 This was a fulfilling and interesting software implementation, but unfortunately it required large amounts of time required by it. I spent the last weekend of the project implementing the motor encoder calibration and position tracking features using turn radii equations. Essentially, this piece of software simulated the motors as two wheels following circles whose radii mapped to the equation Ro + Ri = W where Ro is the radius of the outer circle, Ri is the radius of the inner circle, and W is the width of the robot's turn radius. I decided to have this feature returning a series of structures containing timestamps, X,Y locations, and the robot's orientation within the XY plane. However, I was unable to complete the implementation of the A* algorithm due

to time constraints and unforeseen problems with my serial connection circular buffer driver.

Appendix B - Packaging

Figure B-1. Front view of the robot without protective frame

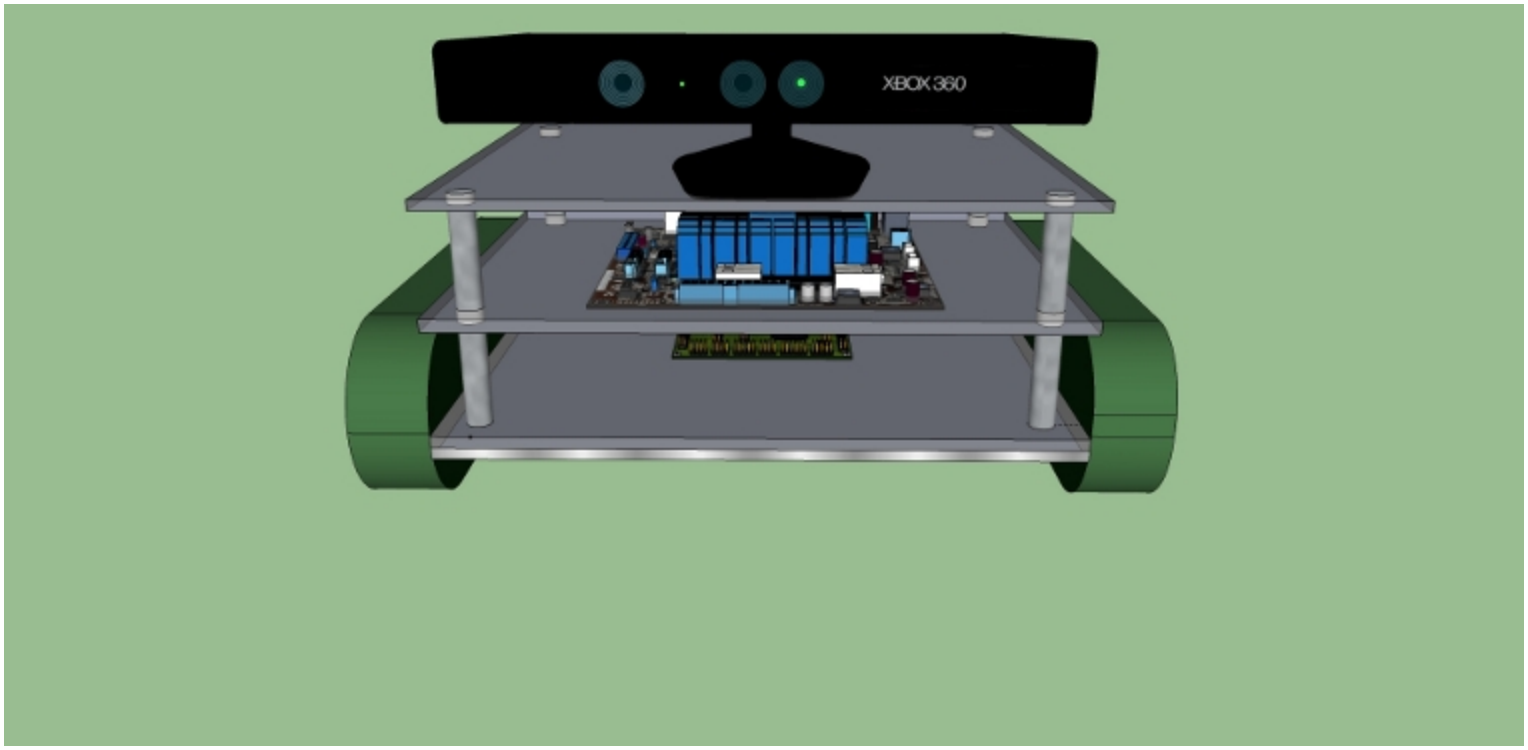
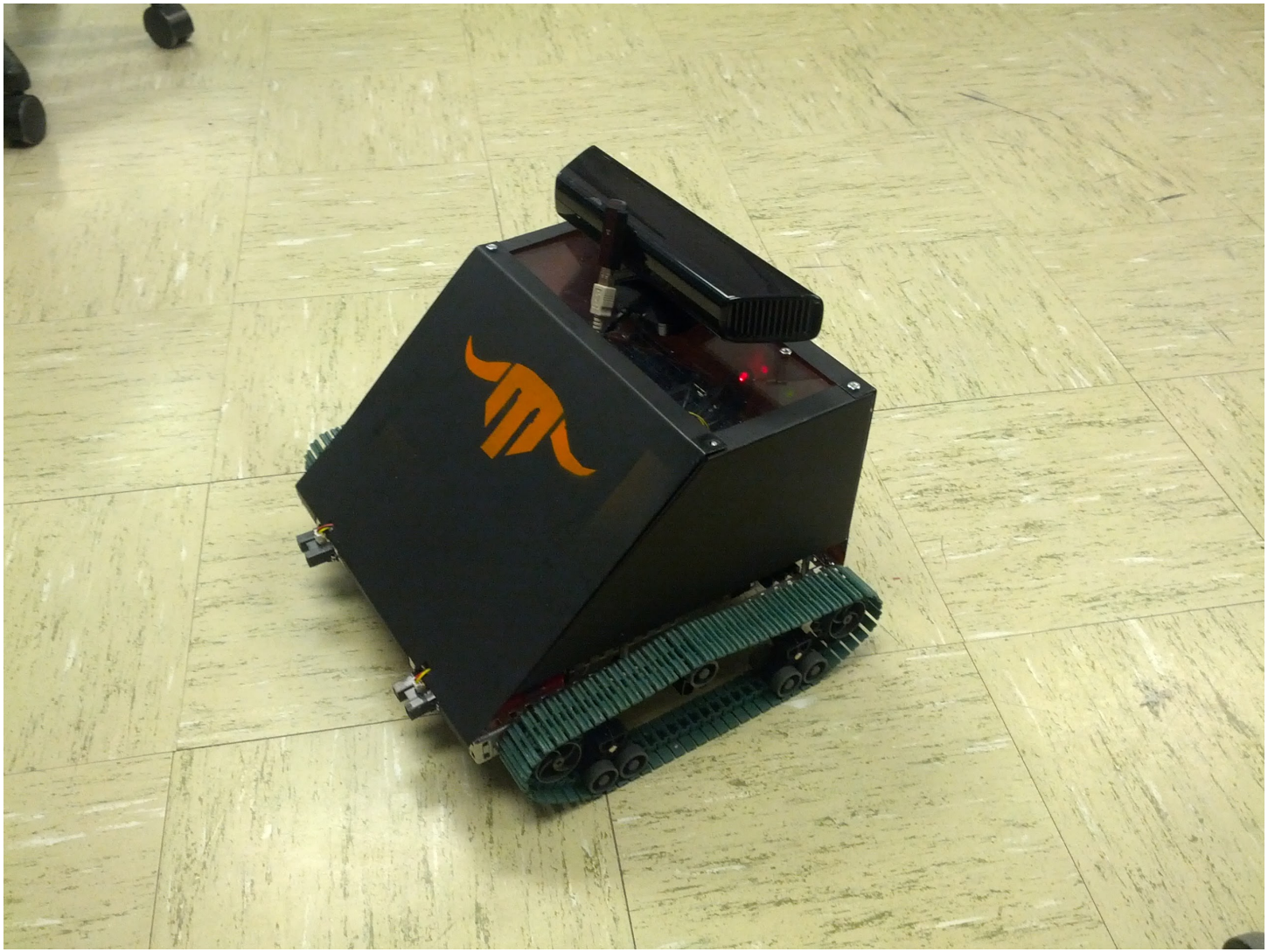


Figure B-2. Side view of chassis with frame



Appendix C - Schematic

Figure C-1. Battery Schematic

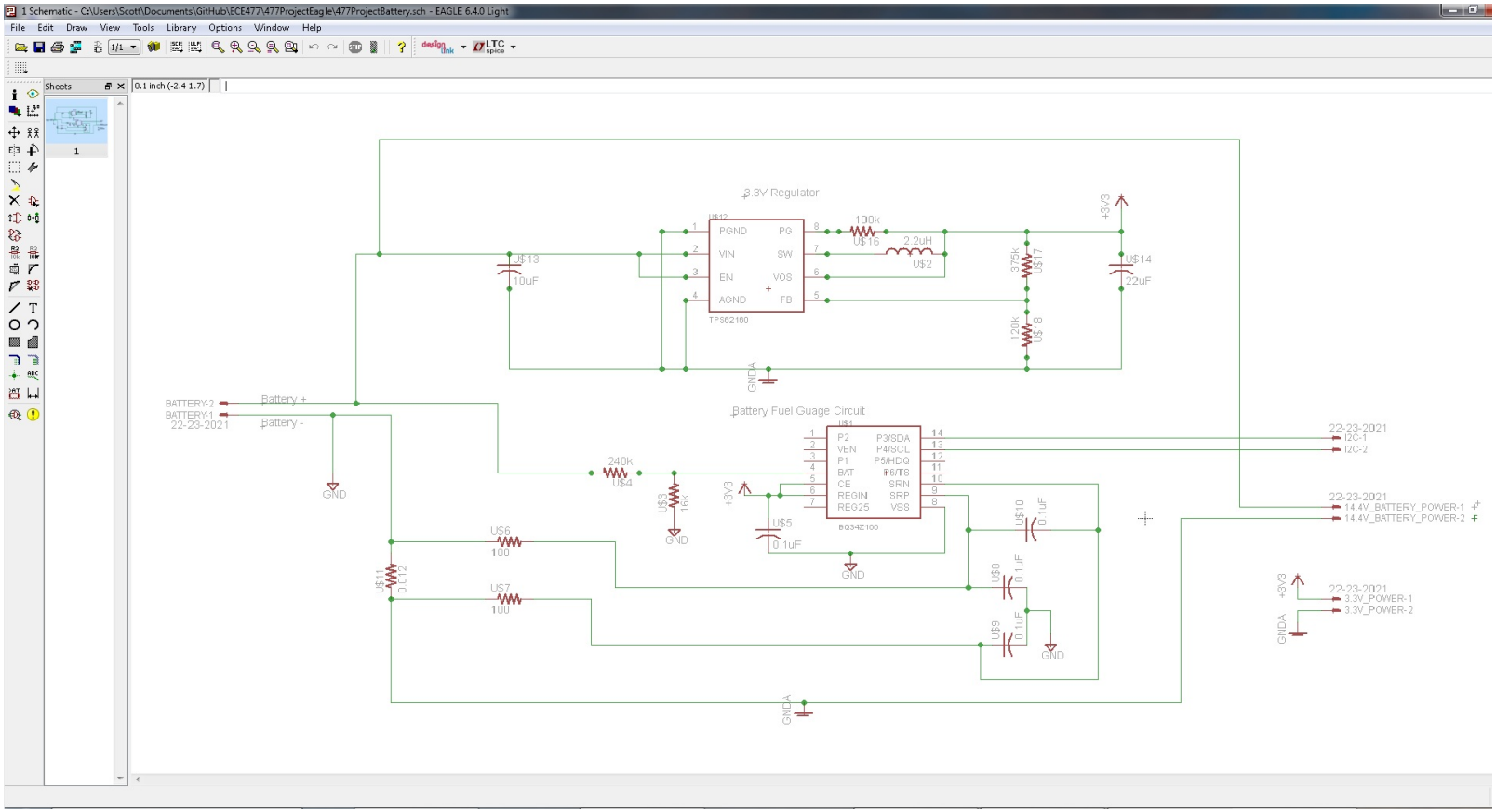
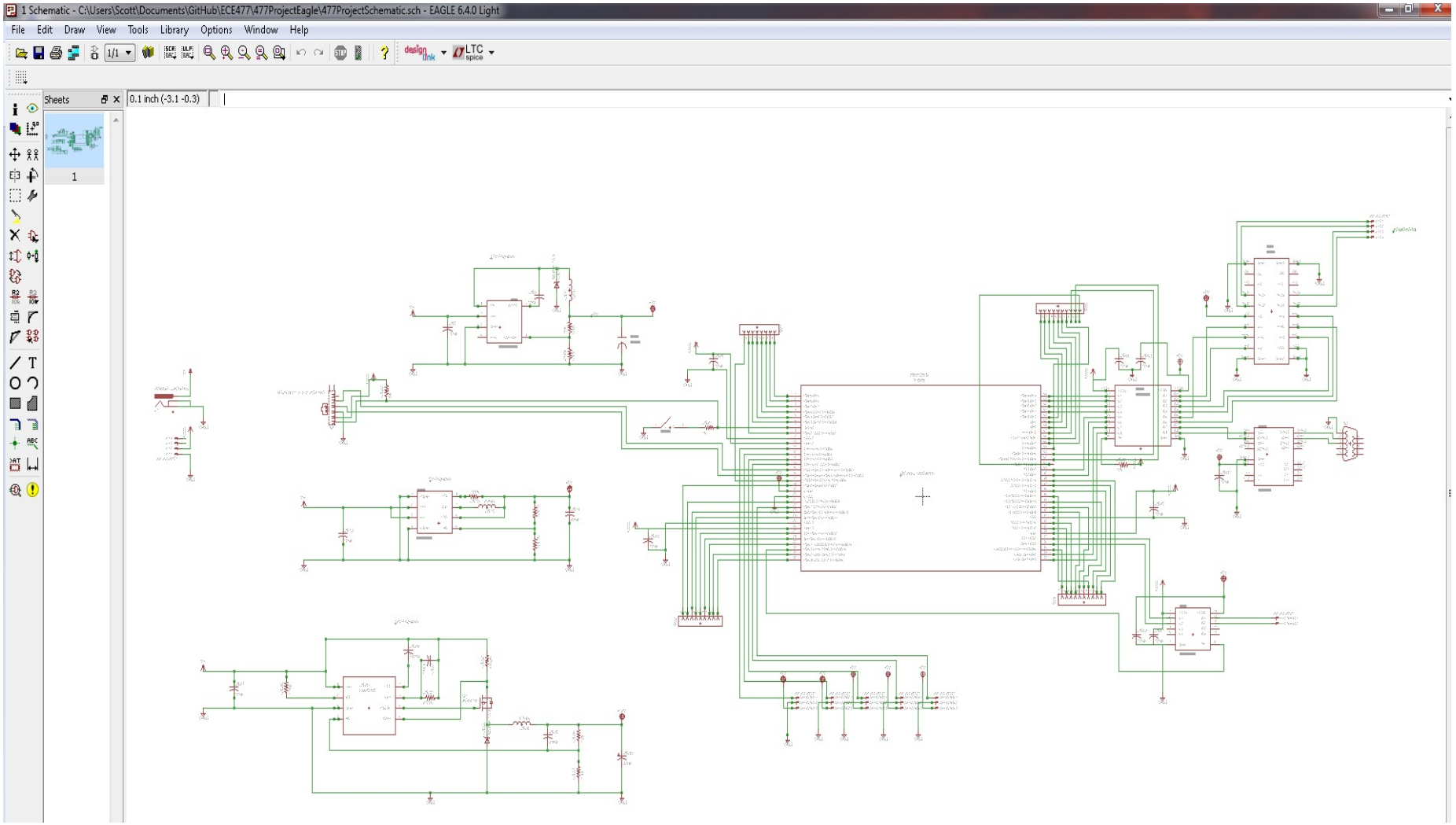


Figure C-2. Main PCB Schematic



Appendix D - PCB Layout Top and Bottom Copper

Figure D-1. Battery PCB layout.

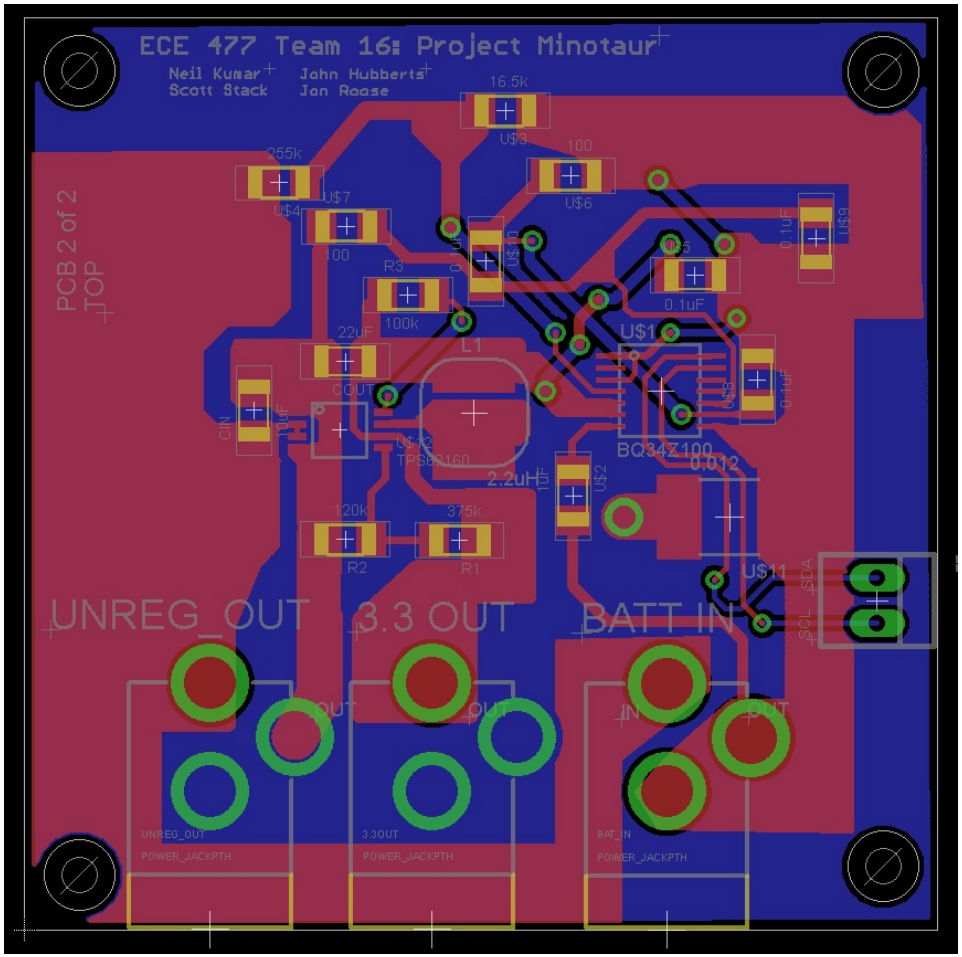
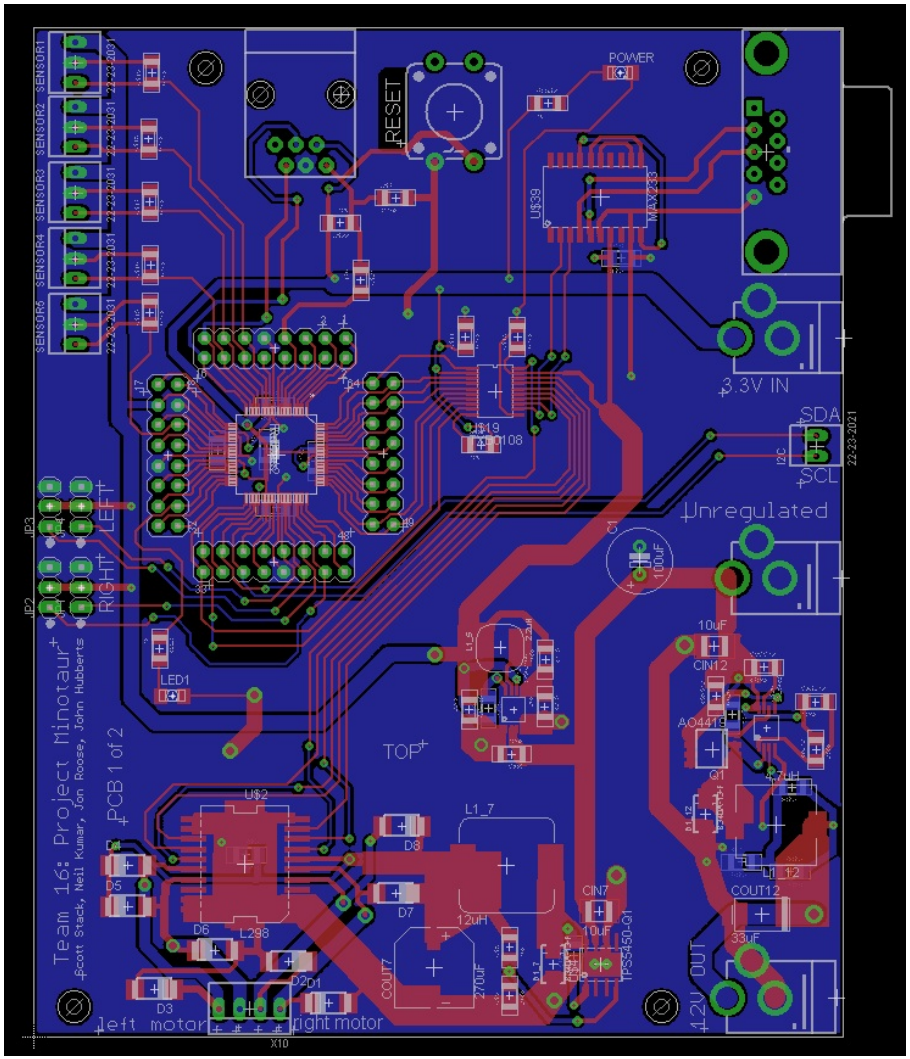


Figure D-2. Main PCB Layout



Appendix E - Parts List Spreadsheet

Appendix F - FMECA Worksheet

Figure F-1. FMECA Sheet

Failure No.	Failure Mode	Possible Causes	Failure Effects	Method of Detection	Criticality	Remarks
A1	Fuel Gauge stuck empty	Internal failure of BQ34Z100, burnout of resistor U$3	Power to the device is cut	Observation, device stops functioning	Moderate	
A2	Fuel Gauge falsely registers charge	Internal failure of BQ34Z100	Battery is overdrawn, potential damage to the battery	Gradual decline in battery life	Low	Should be addressed , but wouldn't destroy the battery instantaneously
A3	3.3V out goes to 0	Burnout of R1 and R2, internal damage to TPS62160	The Microcontroller loses all functionality	Observation	Moderate	
A4	3.3V out short circuits	Overdraw from battery, TPS62160 failure	Microcontroller would be sourced with 14.8V, would likely burn out	Observation	High	Could cause massive heat dissipation

B1	12V output goes to 0V	Internal damage to switching regulator, PGate stuck at logic '0'.	Loss of ATOM board and Kinect functionality	Observation, no packet transaction	Moderate	
B2	12V output becomes short circuit	Internal damage to switching regulator, PGate stuck at logic '1', AO4419 transistor blows out, malfunction in battery IC causing overdraw	ATOM board and Kinect could get burned out	Observation, rapid fuel gauge depletion	Moderate (High)	The Kinect and the Atom Board have internal regulators, but a high enough current could potentially overcome these regulators, and create enough heat to start a fire.
C1	7.2V output goes to 0V	Internal damage to switching regulator, B340A-13-F diode breaks down resistors U$8 and U$9 burn out.	Loss of ability to drive motors	Observation, no motor movement	Moderate	

C2	7.2V output becomes short circuit	Internal damage to switching regulator	Source 14.8V current to the H-Bridge, could potentially cause overheating or loss of motor control.	Observation, smoldering H-Bridge, unstoppable motor	High	The Motors are only rated to handle 7V, so they'd be able to drive at 14.8V for a while, but eventually they might (likely would) overheat, which could cause a fire hazard.
D1	Microcontroller Overheats	Failure in the 3.3V switching regulator, or battery management IC	Loss of motor/sensor control, large heat dissipation	Observation (all functionality lost)	High	Total functionality of the device would be unpredictable, could cause a fire.
D2	H-Bridge Control Outputs stuck at '1'	Internal chip damage	The device would drive indefinitely at full speed	Observation	Moderate (High)	The reason that this is considered a high criticality failure is because a motor stall could cause massive heat dissipation from the h-bridge and motors, leading to a fire.

D3	Sensor Inputs stuck at '0'	Internal chip damage	Proximity sensors wouldn't register obstacles. Motors might stall, device might drive off of a ledge	Observation	Moderate (High)	This could be considered high criticality, as it could lead to a stall (as mentioned in D2), or could lead to the device driving off of a staircase, potentially tripping and injuring the user
E1	Chip acts as a net short circuit	Failure of 7.2V switching regulator	Overdriving the motors, causing stall or overheat	Observation	High	Overheat and stall could lead to a fire hazard
E2	Unable to source current	Failure of 7.2V switching regulator, internal failure of the H-Bridge	No control over the Motors	Observation	Moderate (High)	Could potentially drive off of a staircase, injuring the user as described in D3.
F1	T1out stuck at '1' or '0'	Failure of 5V regulator damage to MAX233	Improper transmission of data to ATOM Board	Packet data incorrect in ATOM board	Low	Could lead to obstacle improper exportation of sensor data, giving atom board an incorrect view of its surroundings

F2	R1in registering incorrect value	Failure of 5V regulator damage to MAX233	Corrupted packet data leading to potentially incorrect instructions	Erratic behavior of device	Low(High)	Could potentially give an instruction the device to run into an obstacle, or off of a ledge, causing a fire or injuring a user located on a staircase as described in D2/D3
G1	Sensors always register maximum range found	Failure of 5V regulator internal damage	Inability to detect obstacles	Observation, viewing control packets	Moderate (High)	See D3
G2	Sensors always register minimum range found	Obstruction to sensors due to dust/environmental factors, failure of 5V regulator, internal damage	Perpetual state of obstacle collision, device freezes in place.	Observation, viewing control packets	Low	