Virtual Touring System

Final Report

Team 46

Kecheng Liu
Yubo Liu
Yigao Shao
TA: Mustafa Mukadam

University of Illinois at Urbana-Champaign
May 1, 2013

Abstract

Current microprocessors are dedicated to a users' specific application but may not be capable to handle multiple tasks. This project explores a real-time touring system that adopts a three-level hierarchy implementation of a system that utilizes the advantages of individual computational unit. This allows us to maximize each of the computational unit's capabilities in a system characterized by real-time tasks such as avoiding obstacles, and followed by large amount of computation such as analyzing the real-time data. This report documents the design, construction, verification, and cost of this system.

Table of Content

1.0 Introduction

1.1 Problem Description

Virtual tour systems are gaining popularity as a substitute for a person's actual presence to save time and money. However, current virtual tours are either at a fixed location or not real-time. This does not grant the users any freedom on choosing his or her experience. This project seeks to provide a real-time virtual tour system that allows the users to have their personal touring experience. The proposed implementation will include a moving vehicle (MV) that houses the touring hardware and a web server that allows the users to control the (MV). The system utilizes a three-level hierarchy that separates tasks into each level. The user level includes the web server and database. The real-time level includes a real-time controller (RTC) and peripheral devices. The intermediate level includes general purpose unit, which is responsible for the communication between the user level and real-time level. We are convinced this system will serve as a useful product to many users.

1.2 Objectives

In this project, we intend to create a complete virtual tour system that combines a robust network between our website and the moving vehicle (MV). In the backend, we will use a web server and a server on the MV to create a P2P connection between the user and the MV. Different users can safely control the MV from the users' computer. The user needs to create an account on our website, after which the user will be able to log in and take control of any idle MV. On the hardware side, we plan to implement various features that allow the MV to be easily controlled and autonomous. Considering the safety issue of the MV, we will implement an environment detector by using ultrasonic sensors. This allows our MV to automatically change speed and direction in emergencies such as seeing an obstacle. For friendly usage and control, a GPS will be installed on the MV to allow automatic controls and navigation.

1.2.1 Functions

The functions of our projects are listed below

1. Travels in different directions and speed
2. A live video camera to provide vision
3. A website for users to log in, choose the desired car which is provided by a car owner and control
4. Collision avoidance system to protect our car from unpredictable hazards
5. Four driving modes
   a. Full real-time manual control
   b. Programmed route control
   c. Automatic navigation via path memory to starting location
   d. Automatic control under GPS navigation.

1.2.2 Benefits

1. Access from long distance from any computers with internet connection
2. Robust internet connection
3. Robust software that is compatible with various network interface
4. Can have real-time tours through on-board live video camera
5. The MV has collision detect and avoidance system to protect the vehicle

1.3 Design Revisions

We faced many design challenges and revised our designs several times throughout the project. The major revisions are listed below.

1. Switch PIC to Arduino as the Real-Time Controller
2. Added 9V battery to power Arduino
3. Added digital compass HMC5883L to aid navigation, which turned out to not help too much
4. P2P network is changed RTMP broadcast due to University's firewall

2.0 Design

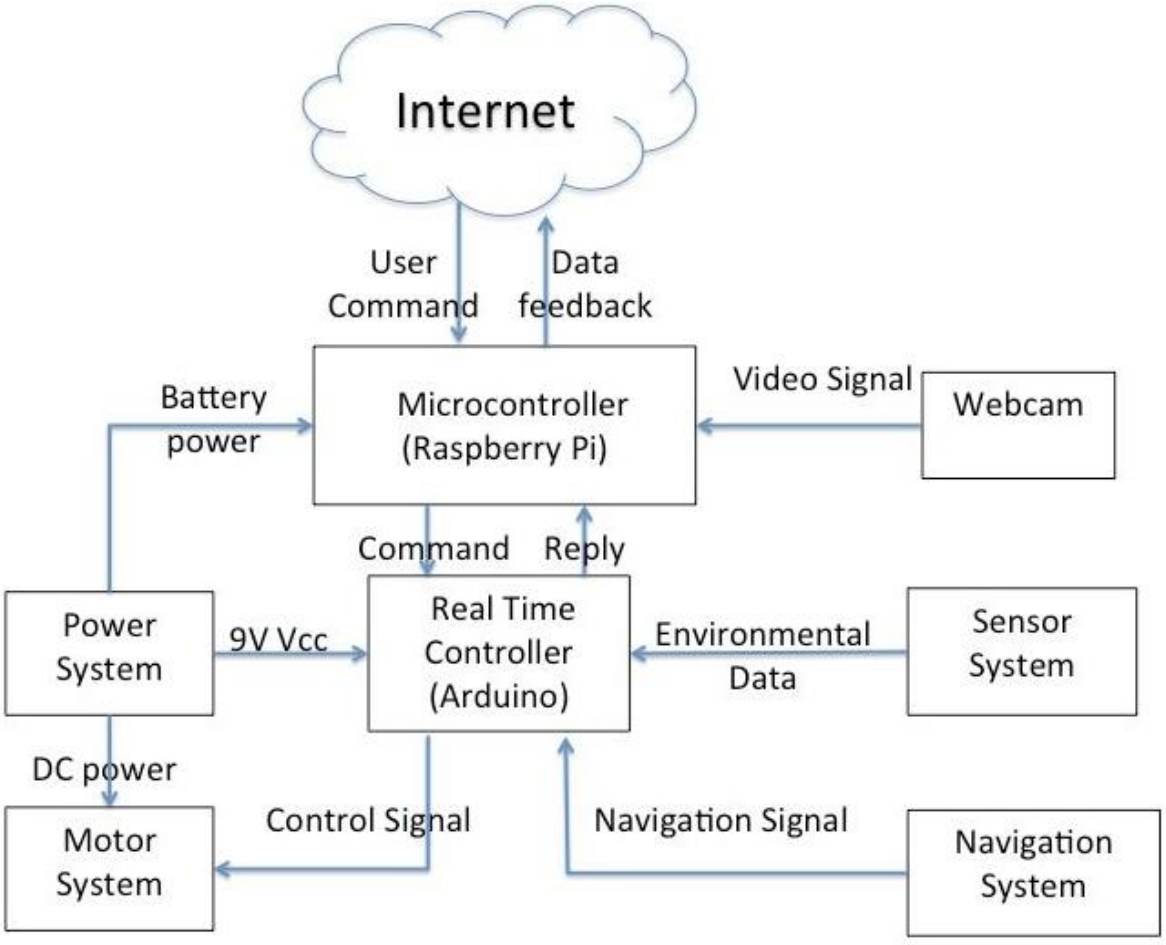

Figure 1 System Block Diagram

2.1 Power Module

2.1.1 Power Units

The Power Module's purpose is to deliver power to other electronic modules. The Power Module consists of three power sources to provide power to the different modules on the vehicle, which are 2200mAH MHK Power Pack battery, five 1.2V AA batteries, and a 9V alkaline battery. Three separate power supplies were used to try to minimize noise coupling to the electronic modules. The connection scheme is shown below in Figure 2 below.

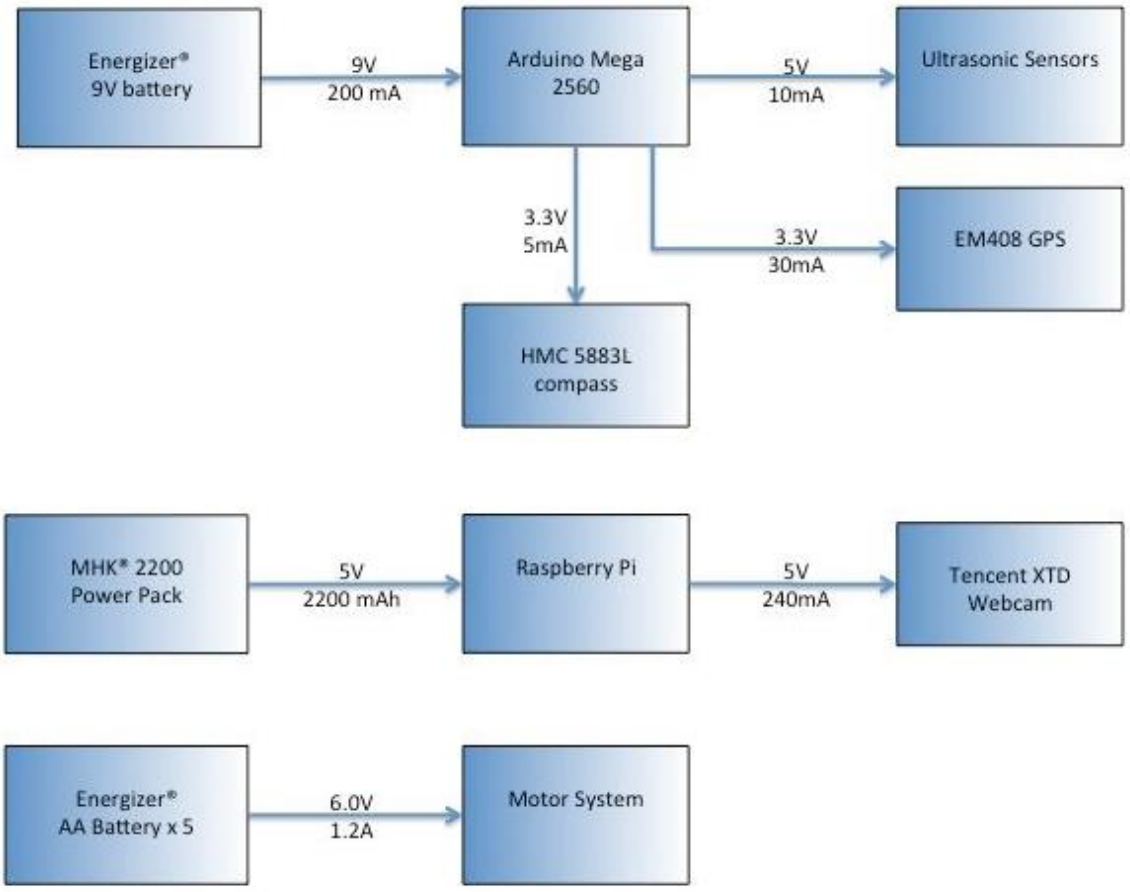

Figure 2 Power Flow for Each Supply

From the power supply chart, we derived a power budget. This budget table gives us a rough estimate on how much each power supply can support. Table 1 below shows the expected maximum ratings from each of the supply and the individual modules.

Table 1. Power Budget Table

Energy available		Energy consumption	
Type	$quantity \times rated\ voltage\ (V) \times ratedelectrical\ charge\ (Ah)$	Type	$quantity \times rated\ power\ (W) \times time(h)$
AA battery	$5 \times 1.2 \times 1.1 = 7.8\ Wh$	Motor (full load)	$4 \times 3 \times 1 = 12Wh$
MHK power pack	$1 \times 5 \times 2.2 = 11\ Wh$	RPI	$1 \times 3.5 \times 1 = 3.5\ Wh$
9V battery	$1 \times 9 \times 0.5 = 4.5\ Wh$	Arduino	$1 \times 3 \times 1 = 3Wh$
		Sensors	$4 \times 0.015 \times 1 = 0.06\ Wh$
		GPS	$1 \times 0.145 \times 1 = 0.145\ Wh$
		Compass	$1 \times 0.05 \times 1 = 0.05\ Wh$
Grand total	$7.8 + 11 + 4.5 = 23.3\ Wh$		18.775 Wh

2.2 Motor Module

2.2.1 Motor Control

The 5-6 VDC rated motors dictate the speed and the direction of our moving vehicle. The purpose of the H-bridge is to translate the PWM and control signals from the Arduino Mega 2560 to control the speed and direction of the DC motors.

The layout of the MV is shown in Fig 2. Two motors are placed on each side of the MV: Motor 1 (M1) is on the left side and Motor 2 (M2) is on the right side.
The MV moves forward if M1 spins counter clockwise (CCW) and M2 spins clockwise (CW), viewed from the front side of the MV. For backward motion, we simply flip the direction of motor spinning. The MV turns to the left if M2 spins CW and M1 remains stationary; it turns to the right if M1 spins CCW and M2 remains stationary.

The spinning direction of both motors is determined by the direction of current flow. M1 spins CCW (CW) and M2 spins CW (CCW) if current flows from their positive (negative) pin to negative (positive) pin and the MV moves forward (backward). Based on this setup, we can determine how M1 and M2 are connected with H-bridges.

The schematic of H-bridge is shown in Figure 3. H-bridge consists two n-type MOSFETs and two p-type MOSFETs The gate of each MOSFET acts as the input of the H-bridge and noted as A, B, C and D. Based on the fact that for n-type and p-type MOSFET, current will flow across the drain and source when gate voltage is respectively above and below threshold voltage (3.3V in our case), we can make up a truth table, as shown in Table 2, for the digital relationship between gate voltage level and MV moving directions.

Table 2 Truth Table for H-bridge Digital Control

A	B	C	D	MV Moving Directions
1	0	0	1	FWD
0	1	1	0	BWD
1	1	0	0	Break
0	0	1	1	Break

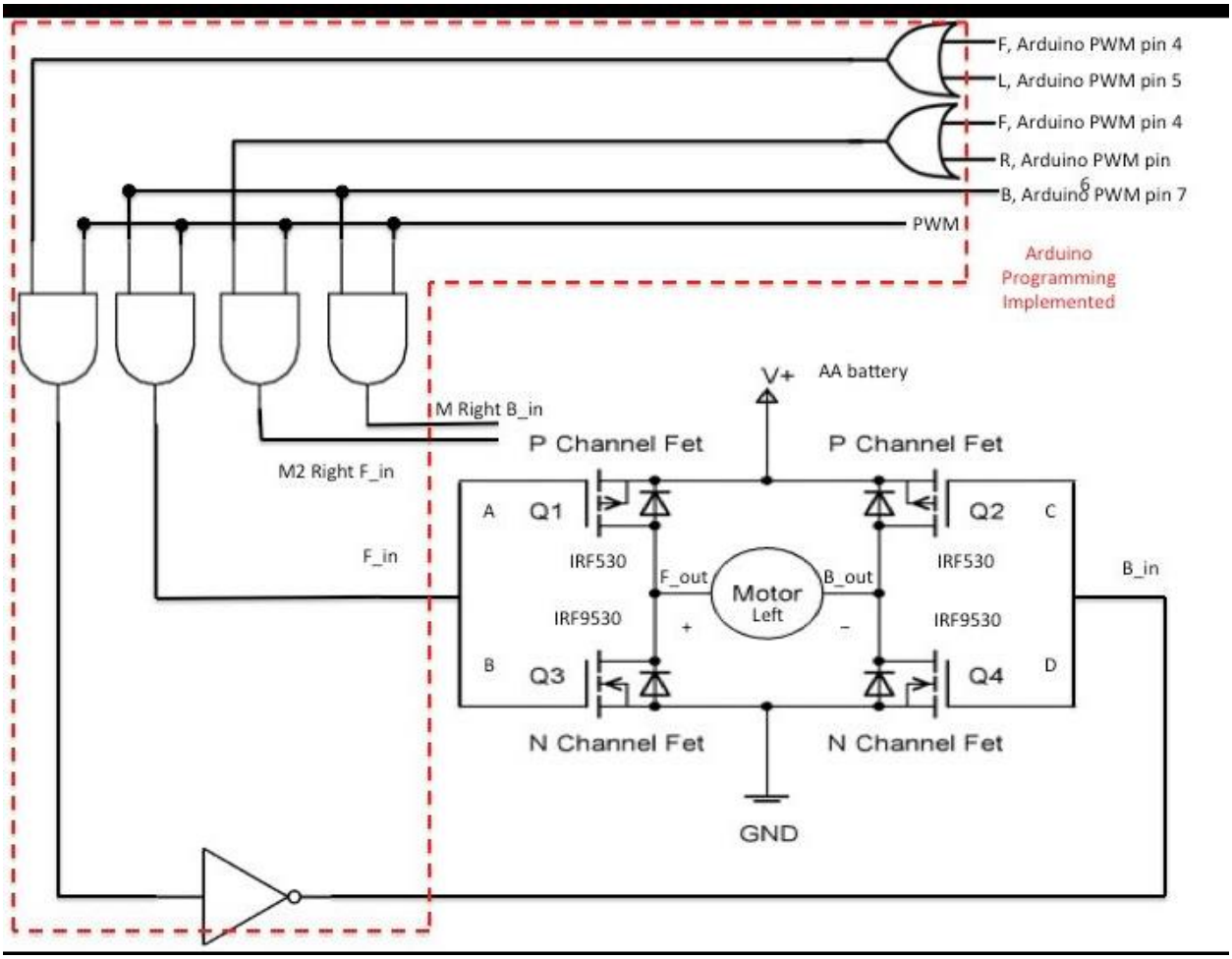

Fig 3 H-Bridge Schematic

Logic 1 and 0 represents voltage level above and below the gate threshold voltage, respectively. From the truth table we can acquire two Boolean expressions for MV motions as equation (1) and (2):

$$FWD = AB'C'D \quad (1)$$

$$BWD = A'BCD' \quad (2)$$

Because MV can only move when AD and BC are excited separately, we connect A and B, C and D together as two inputs of the H-bridge and noted as: forward input (F_in) and backward input (B_in). Two outputs of the H-bridge are: forward output (F_out) and backward output (B_out). F_in and B_in receive input signals from PIC; F_out and B_out are connected to the positive and negative pins of the motors. According to the discussion above, the MV will move forward when current flows from positive pin to negative pin for both M1 and M2; therefore, the positive pins for both M1 and M2 are connected to the F_out and the negative pins are connected to the B_out.

Below is the Control logic for the MV. Because MV will receive four commands from user: forward (F), backward (B), left turn (L) and right turn (R), and receive PWM signal from RTC, the truth table for MV can be constructed and shown in Table 3

Table 3 Truth Table for MV Digital Control Signal

F	B	L	R	PWM	M1	M2
1	0	0	0	1	FWD	FWD
0	1	0	0	1	BWD	BWD
0	0	1	0	1	FWD	Break
0	0	0	1	1	Break	FWD
X	X	X	X	0	Break	Break

From the truth table Boolean expressions can be determined as equation (3) to (6)

$$M1_{FWD} = (F + L)PWM \quad (3)$$
$$M1_{BWD} = B \times PWM \quad (4)$$
$$M2_{FWD} = (F + L)PWM \quad (5)$$
$$M2_{BWD} = B \times PWM \quad (6)$$

$M1_{FWD}, M1_{BWD}, M2_{FWD}, M2_{BWD}$ are four inputs for two H-bridges and will be connected to F_in and B_in separately.

Additionally, to prevent the hazard from backward EMF current due to motor spinning, we placed diodes in parallel with each MOSFET to protect our sources.

2.2.2 Motor Decoupling

DC motors are noisy components, meaning they will produce spurious signals when active. The spurious signals will cause erroneous data in communication. To minimize the effect of the motors' noise, decoupling capacitors are added to each motor to filter the noise signals.

The 0.1 uF capacitors are designed to filter out the noise higher than 1MHz. Large noise signals, which will contaminate our signal transmission, are mostly filtered out. However, smaller noise signals still exist at frequencies lower than 1MHz, and they can trigger erroneous data but at a much lower rate. The setup is shown in Figure 4 below.

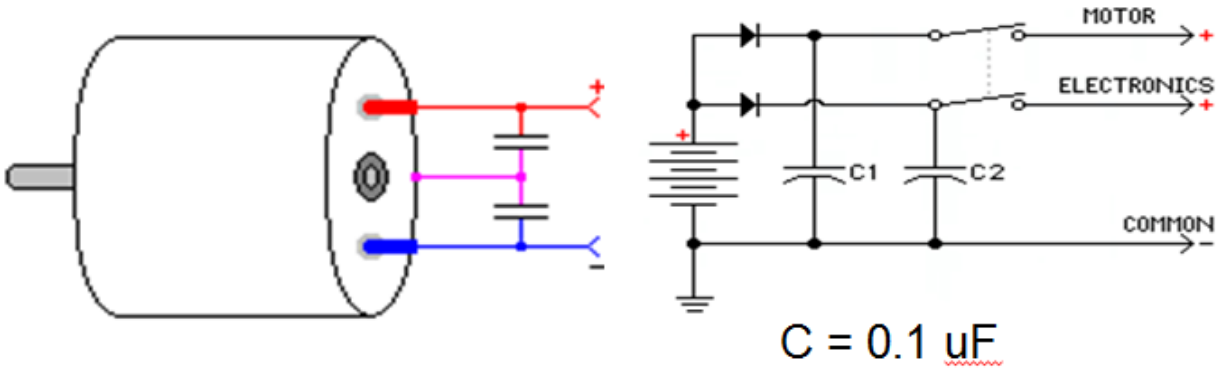

Figure 4 Decoupling Capacitor

2.3 Microprocessors

2.3.1 Raspberry Pi (RPi)

RPi serves as a general purpose processing unit that serves as the communication between the moving vehicle and the web server. It is a powerful unit that has growing community support. Its setup is shown in the Figure 5 below.

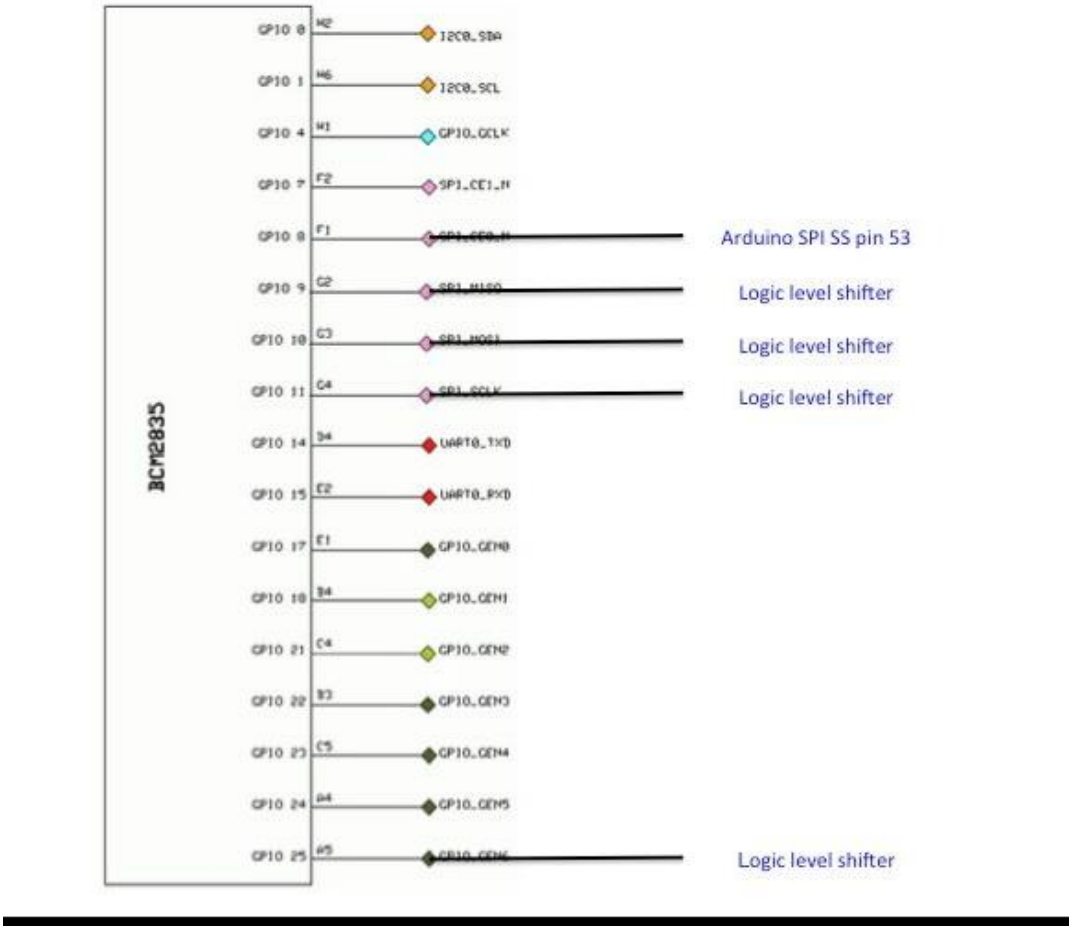

Figure 5 RPi

2.3.2 Arduino Mega 2560

The Arduino microprocessor is a weaker processing unit compared to the RPi. However, the Mega provides real-time functionalities, which something RPi lacks. The Mega is a dedicate microprocessor that handles the real-time data from its peripheral devices such as the sensors and the GPS. The original choice was a PIC. However, PIC had limited pins. The Mega was chosen for it abundance of pins. Its setup is shown in the Figure 6 below.

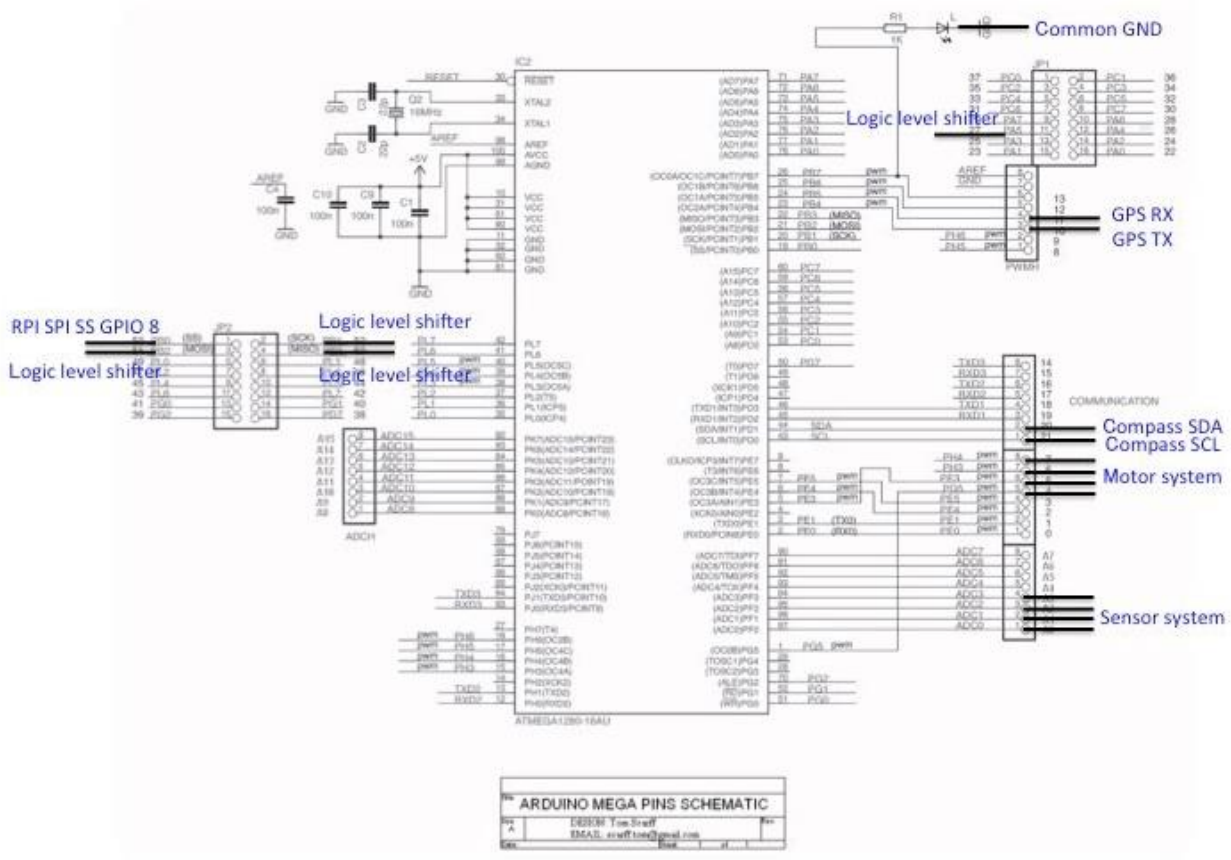

Figure 6 Arduino Mega

2.3.3 Voltage Level Shifter

To communicate between the Arduino and the Raspberry Pi, we adopted the Serial Peripheral Interface (SPI) for serial communication. However, the Arduino outputs a 5V as a '1', while the Raspberry Pi outputs 3.3V. The level shifter's purpose is to convert the 5V to 3.3V and 3.3V to 5V on each side. This prevents possible damages to each of the microprocessors. The schematic is shown below.

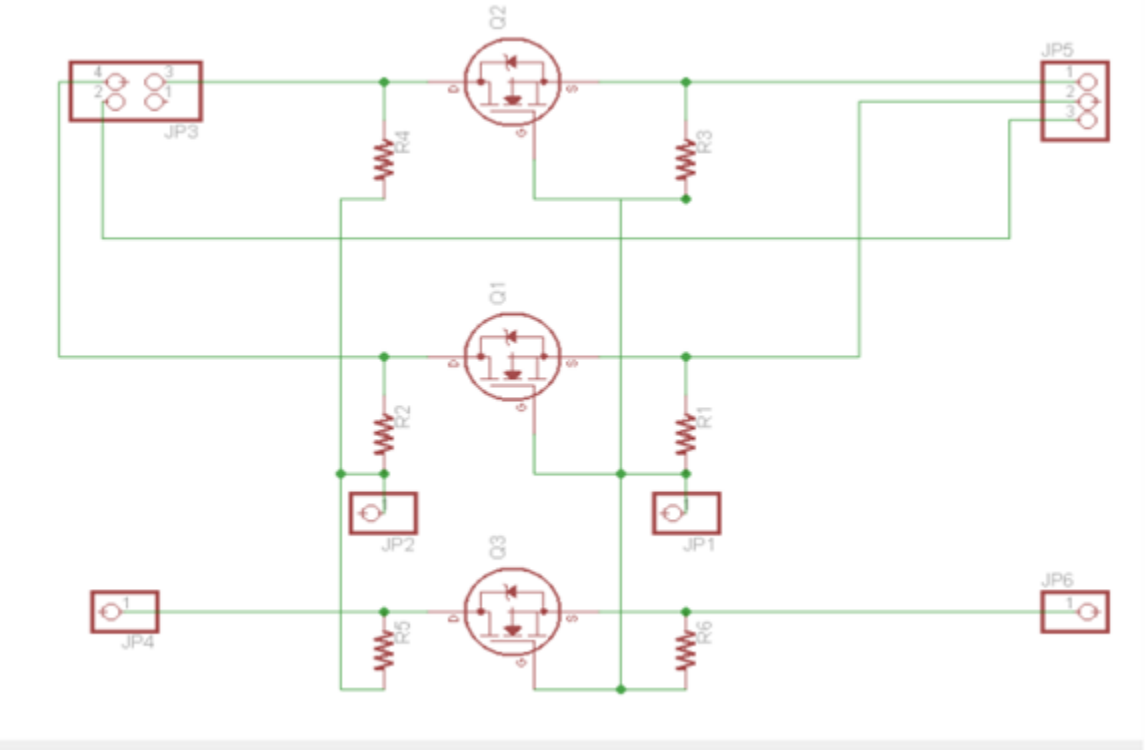

Figure 7 Voltage Level Shifter Schematic

2.3.4 Serial Peripheral Interface (SPI)

Communications between the two microprocessors are done via a serial link known as SPI. SPI provides large data throughput while uses less circuitry compared to other serial communication methods such as UART. The data transmission is performed by two serial shifters. The transmission process is illustrated in the Figure 8 below, in which the RPi acts as a Master, and the Mega acts as a Slave.

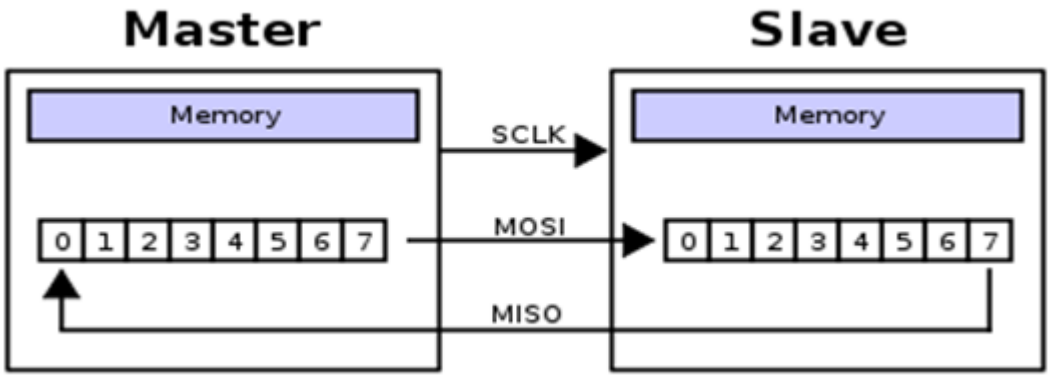

Figure 8 SPI Scheme

To set up the software handlers on the two microprocessors, we found supporting libraries on the RPi and the Mega. After software initialization, the communication is initiated by the RPi after it sends an edge on the clock line.

2.4 Sensor Unit

2.4.1 Sensors

Sensors were installed on the MV to protect the Moving Vehicle (MV). This ensures that the MV does not suffer damages from lousy user control or faulty data transmission/reception. The sensors chosen are MB1000 family ultrasonic sensors from MaxSonar. The sensors will transmit a signal and wait for its response. The distance can be determined from the signal's delay. The reason for choosing ultrasonic sensors is that the operating frequency is low enough so it is convenient to use. Its setup is shown in the Figure 9 below

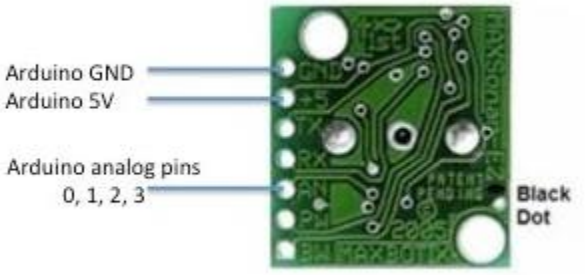

Figure 9 Sensors Setup

The sensors communicate with the Mega via an analog signal, which can be decoded by the Mega by its internal Analog-to-Digital Converter. With the signal, the distance of the obstacle can be decoded by software using the equation (7).

$$distance\ in\ cm = \frac{Vcc}{512 * 2.54} \quad (7)$$

2.4.2 Obstacle Avoidance Algorithm

Considering the safety of the Moving Vehicle (MV), an obstacle avoidance algorithm was implemented to protect the MV. The algorithm uses the ultrasonic sensor to locate obstacles. Our algorithm assumes small (less than 1 meter) obstacles as our modeling. The overall idea is demonstrated in the Figure 10 below.

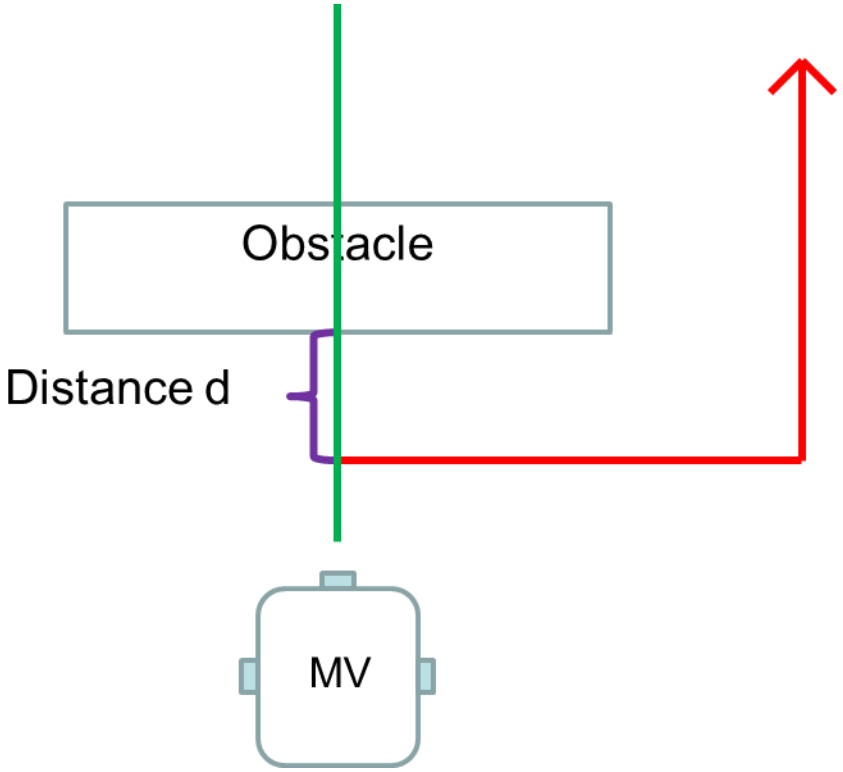

Figure 10 Obstacle Avoidance Algorithm Example

Upon seeing an obstacle at distance d (determined by user, but minimum of 16cm), the MV stop and turns 90 degrees. Then it will proceed to move straight as long as the sensor facing the obstacle (left sensor in the diagram) sees a distance less than d. As soon as the distance increases over d, the MV will go straight for another one second. Then, the MV will stop and turn back to its original path's heading. At this point, the avoidance has been completed.

2.5 Navigation Module

2.5.1 GPS

The hardware we used to implement the navigation system design is a GPS module and a digital compass module.

The GPS module we are using is EM408. We chose this module because it is sensitive to low power signal (-168 dBm) and it has high accuracy (5m). It will give us the global coordinates in latitude and longitude. The data transmission method between GPS and Arduino is called Universal Asynchronous Receiver/Transmitter RS-232 (UART). This method could send and receive data separately, which assures the reliability of transmission. The basic UART transmission diagram is shown below in Figure 11.

In addition to GPS module, the navigation system also contains a digital compass module, HMC5883L to acquire bearing. Bearing is the angle between “true north” and current heading of the platform. The data transmission method between compass and Arduino is called I^2C. It sends data synchronously with its own Serial Clock (SCL). The Serial Data Line (SDA) contains the message of either write or read at a specific slave address by encoding commands in the first byte it sends: the first seven bits of the first byte refer to the slave address and the last bit of the first byte refer to read if 1 or write if 0. The advantage of this method is that a master could interface with multiple slaves by using only two transmission lines. The basic I^2C transmission diagram is shown below in Figure 12

.

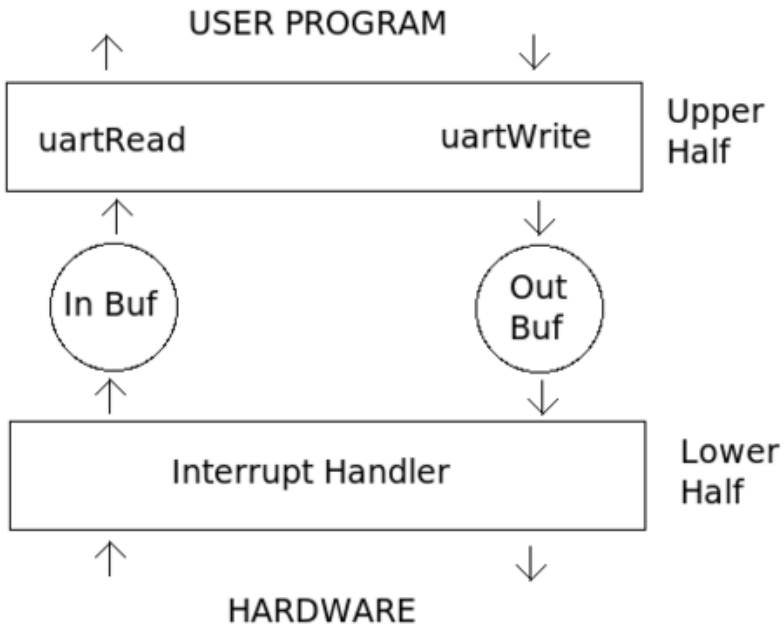

Figure 11. UART transmission diagram

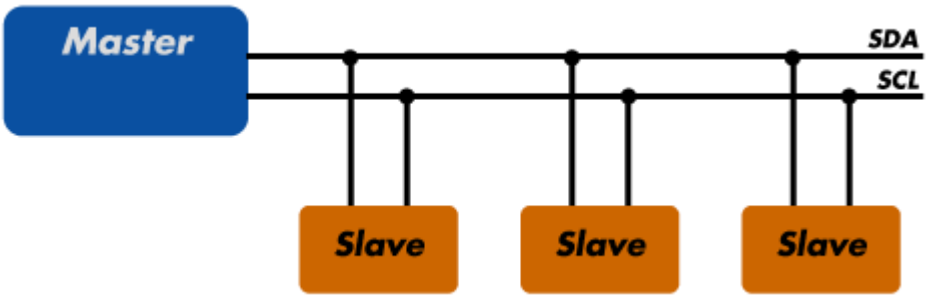

Figure 12. I^2C transmission dagram

2.5.2 Navigation

Google map is used as an assistance to implement the automatic navigation function. First of all, when navigation mode is chosen, user will need to tell the platform where to go by typing in the destination coordinates. Then RPI will request Arduino to feed coordinate data from GPS module and upload the coordinates to webserver. Webserver will use Google map to calculate the route in "walking mode" and the webserver will extract the route from Google map and generate a sequence of messages, which are in format of "heading, direction, distance". The example route from Google map, starting at Everitt Lab and ending at Engineering Hall, is shown in Figure 13. Those messages will be send from webserver to RPI and then sequentially to Arduino. Arduino will perform each sequence of routing command by using the current bearing from compass module as heading, and then turn to the specified direction and move the specified distance towards destination.
However, because of the uncertainty of road condition and the incapability to accurately move a specified distance, we will use an assisting algorithm to help approach desired destination.
After Arduino receives the first sequence of "heading, direction, distance" message from RPI, it will calculate the destination coordinates, latitude $\varphi_2$and longitude λ_2, by using its current coordinates, latitude $\varphi_1$and longitude λ_1, destination bearing and distance from current location to destination. The equations for this calculation are Equation (8), (9) and (10), where d is the distance, θ is the destination bearing and R is the radius of earth.

$$\varphi_2 = \sin^{-1}(\sin(\varphi_1)\cos\left(\frac{d}{R}\right) + \cos(\varphi_1)\sin(\frac{d}{R})\cos(\theta)) \quad (8)$$

$$\lambda_2 = \lambda_1 + atan2(\sin(\theta)\sin\left(\frac{d}{R}\right)\cos(\varphi_1), \cos\left(\frac{d}{R}\right)\sin(\varphi_1)\sin(\varphi_2)) \quad (9)$$

$$atan2(y,x) = 2\tan^{-1}(\frac{y}{\sqrt{(x^2+y^2)}+x}) \quad (10)$$

$$a = sin^2\left(\frac{\Delta\varphi}{2}\right) + cos(\varphi_1)cos(\varphi_2) * sin^2(\frac{\Delta\lambda}{2}) \quad (11)$$

$$dis = 2 * r * sin^{-1}(\sqrt{a}) \quad (12)$$

After the platform finish moving the first sequence of route specified by Google map, it will most likely end up at an undesired point, as shown in Figure 14. Before the RPI can send out the second sequence of route, the platform will use equation (11) and (12) to calculate the distance between its current location (the undesired location) and the destination location (the desired location). It will then turn to the correct bearing and move the calculated distance towards destination. The platform will keep performing this algorithm and approaching destination; it will stop moving only if its current coordinates are the same as the desired destination coordinates. This indicates the first sequence of routing command has been finished and Arduino will tell RPI to send out the second sequence of message. This method will guarantee the platform accurately reach the desired destination coordinates every time it receives a sequence of routing command so that error caused by unexpected uncertainties will not stack up.

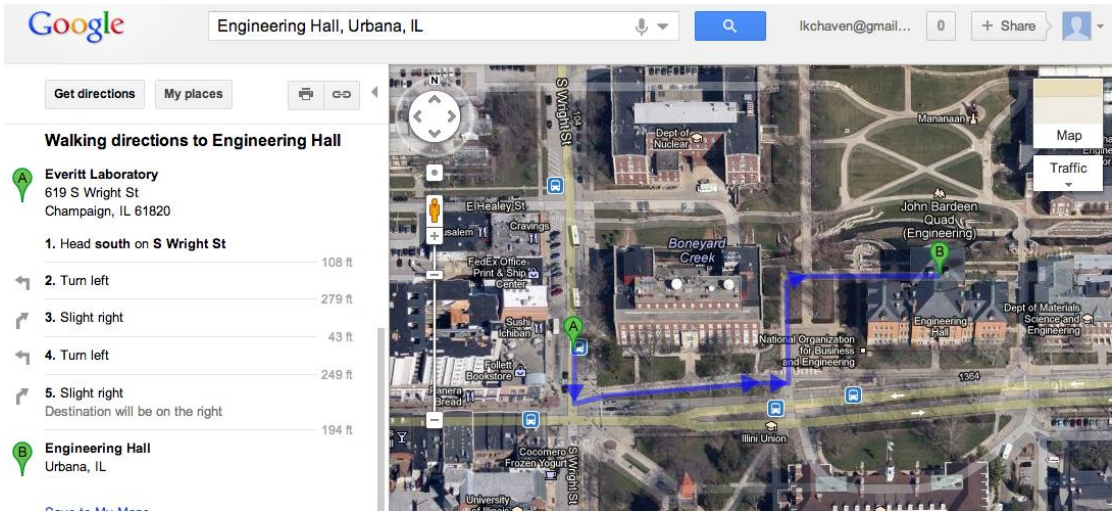

Figure 13. Sequential route generated by Google map

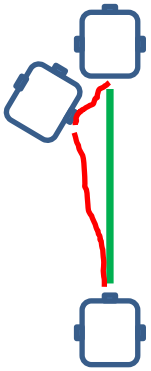

Figure 14. Approaching destination using algorithm assistance

3.0 Requirements and Verification

Because our system constitutes of numeral components, our verification is long. The Requirement and Verification tables are listed in Appendix A.

3.1 Failed Verification

The Motor Module was a major culprit that resulted in many of failed verifications. In essence, we failed to account for the spurious and noisy effects of the motors on rest of our circuits, mainly the serial communication between the Raspberry Pi and the Arduino Mega. The major fault lies in the progress of our design. We tested the link between the Arduino and the motor (PWM) and the link between Raspberry Pi and Aruduino individually; this did not translate to the fact that all three worked properly. However, we were not aware of that and designed our PCB without factoring testing the three modules together.

After receiving the PCB, when we began testing our project as a system, problems began to surface. The main problem is the serial communication errors between the Raspberry Pi and the Arduino with Serial Peripheral Interface protocol. We were very confused by this. As it turns out, the error stems from the noise of a running motor. At this point, we had spent too much debugging and did not have enough time to redesign our circuit. As a result, many of our major functionalities testing and debugging were hindered.

Another major implication was the method of determining the bearing. We have adopted a digital compass (HMC5883L). However, its readings fluctuate as it moves in various directions. This causes error in the determination of angles, making features such as guided navigation and GPS navigation difficult to implement.

3.2 Challenges

There are two challenging components in our project: video streaming and GPS navigation. Video streaming is challenging because of the need to minimize its data as well as ensuring its quality. GPS navigation's challenge lies in the precision of our system. The GPS works well by itself. However, integrating it onto a moving vehicle is limited by the inaccuracies from the motor and the compass.

3.3 Future Work

There are several major improvements that can be made to our system. The first is a better noise filtering circuitry for the motor. The motor noise is the root of failure for data failure between the two microcontrollers. Limiting the noise is the key to stable data transmission. The second improvement is to use shared memory rather than SPI for microcontroller communication. This allows for more efficient and reliable data transmission. Lastly, we need to improve the video streaming capability.

In addition for modifying our project, we believe that the three-level hierarchy can serve as a prototype for any related system. We are convinced that this prototype will have market value in the future.

4.0 Cost and Schedule

4.1 Labor cost

Name	(Rate/hour)×(Total Hours per week)×12	Total Price
Yigao Shao	$\$35 \times 30 hours \times 12\ weeks$	$12,600
Kecheng Liu	$\$35 \times 30 hours \times 12\ weeks$	$12,600
Yubo Liu	$\$35 \times 30 hours \times 12\ weeks$	$12,600
Labor total		$37,800

4.2 Part Cost

Part Name	Manufacture	Model /Part#	Description	Quan	Price /Unit	Total Price
Mr. Basic	DAGU	TR3	Vehicle	1	$30	$30
Raspberry Pi	Farnell and RS Component	Model B	Microcontroller	1	$35	$35
GPS	Global Sat	EM-408	GPS	1	$30 (445)	$30
Webcam	Tencent	QQ XTD	Webcam	1	$25	$25
WIFI Adapter	Sabrent	A11N	WIFI	1	$20.25	$20.25
Ultrasonic Sensor	MaxBotix	MB1000 Family	Sensors	5	$28 (445)	$140
AA battery	Energizer	Alkaline	Rechargeable	6	$3	$18
MHK Batter	MHK	Power Pack 2200mAH	Battery	1	$20	$20
PIC	MicroChip	PIC16F877	PIC	1	$4.50 (445)	$4.5
SD Card	SanDisk	008G-A11	Memory	1	$18	$18
IC & transistors					(445)	$10
Resistor, caps					(445)	$15
PCBs			Fabricated by Part Shop	3	$0	$0
					Total	$355.75

4.3 Total Cost

Part Cost	$365.75
Labor Cost	$37,800
Total Cost	$38,165.5

5.0 Conclusions

5.1 Wrap-UP

All the progress state in the presented report is fully tested. Only one of the functions, GPS navigation, from the original plan failed. This is due to not heeding the motor noise and taking the necessary steps to minimize that noise. In addition, some units were added/changed in accordance of this project. Overall, this was a successful project and laid some foundational ground work for future work.

5.2 Ethics

The purpose of this project is to develop a real-time Wi-Fi controlled moving platform. Its Internet remote control ability ensures that the vehicle could be used in places where potential hazards for human beings might occur. With such function, our device helps increase the safety and health of the user, which is consistent with the first code of the IEEE Code of Ethics:

1. To accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment.
2. Throughout the development of the platform, we will only make claims and estimates based on real data acquired from measurements. We will be honest and will not falsify the data acquired from our test procedures, as cited in the third code.
3. To be honest and realistic in stating claims or estimates based on available data.
4. After this project, we will have learned various real-world industrial systems such as the PIC emulation, Internet security and certification agreement, and the Raspberry Pi protocol usage. From experiments and testing procedures, our understanding of technology and comprehension of real-world application will be improved, as directed in the 5th and 6th codes of the IEEE Code of Ethics.
5. To improve the understanding of technology; its appropriate application, and potential consequences. To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations.
6. While working on the project, we accept criticism and suggestions from both group members and from outside. Additionally, since this project is a group project, we will help each other in his academic development and to support him in following the code of ethics.
7. To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others.
8. To assist colleagues and co-workers in their professional development and to support them in following this code of ethics.

5.3 Safety

Safety issue is very important because the work we performed in a laboratory can have a significant effect on us now and in the future. Our knowledge of safety is a critical element in protecting ourselves from potential hazards. Others who work in the laboratory—co-workers and service personnel—also rely on us to make choices that keep them safe.

All team members in our group have finished the *General Laboratory Safety Training* and *Electrical Safety for Labs* and have acquired *Certificate of Completion* provided by *Division of Research Safety*. We will closely follow the laboratory rules through design and be responsible for our own safety.

We will always keep safety in mind because the key to protecting our health and safety is to be able to identify hazards and take necessary precautions to protect ourselves and other people every time we work in lab.

In addition, the MV will be moving around the Engineering Quad, which could be a potential hazard to others and to itself. Thus, we will have a flag on the MV as an indicator to others.

We will be using lab equipment such as oscilloscope. We should be careful as to not eat or drink around the testing equipment. We should not make circuit changes or perform wiring when connected and power is on. Changes should be performed while power is off. In addition, we will be attentive with readings of the max input to the oscilloscope.

6.0 References

Beej's Guide to Network Programming. [Online]. Available: http://beej.us/guide/bgnet/

Chormium Community. [Online]. Available: http://www.chromium.org/

Division of Research Safety (DRS) [Online]. Available: https://www.drs.illinois.edu/tdb2/Quizzer/Presentations/GLS/GeneralLabSafety.html

Eagle User Guide,Cadsoft, 2010, [Online]. Available: http://www.cadsoft.de/wp-content/uploads/2011/05/manual_en.pdf

EM408 GPS Engine Board, Global Sat, Taiwan, 2009. [Online]. Available: http://www.usglobalsat.com/store/download/47/em408_ug.pdf

GCC Community. [Online]. Available:http://gcc.gnu.org/

HI-TECH C Compiler, Microchip Technology Inc, AZ, 2010. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/manual_PICC_983.pdf

ICE protocol RFC 5245, IETF, J.Rosenberg et.al, 2010,[Online].Available:http://tools.ietf.org/html/rfc5245

IEEE Code of Ethics [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html

Introduction to PIC Programming: [Online]. Available: http://www.gooligum.com.au/tutorials/midrange/PIC_Mid_C_1.pdf

Libcurl Community. [Online]. Available: http://curl.haxx.se/libcurl/c/libcurl-tutorial.html

Libjingle Community. [Online]. Available: https://developers.google.com/talk/libjingle/index

Libnice Community.[Online]. Available: http://nice.freedesktop.org/wiki/

LV-MaxSonar-EZ0 High Performance Sonar Range Finder, Maxbotix, MN, 2005. [Online]. Available: http://www.maxbotix.com/documents/MB1010_Datasheet.pdf

LV-MaxSonar-EZ1 High Performance Sonar Range Finder, Maxbotix, MN, 2005-2012. [Online]. Available: http://www.maxbotix.com/documents/MB1000_Datasheet.pdf

MPLAB tutorial, Inbredded Systemshttp. [Online]. Available://morrish.ca/beginner/interrupts.php

MPLAB X Integrated Development Environment, Microchip Technology Inc, AZ,2011-2012.[Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/52027B.pdf

Mr.Basic Model TR-3, DAGU Hi-Tech Electronic Co. LTD, China, 2009. [Online]. Available: http://www.robotshop.com/content/PDF/mr-basic-tr-3.pdf

PIC16F87X7 40-Pin 8-Bit CMOS FLASH Microcontroller, Microchip Technology Inc, AZ, 2001.[Online]. Available http://ww1.microchip.com/downloads/en/devicedoc/30292c.pdf

Raspberry Pi community. [Online]. Available: http://elinux.org/RPi_Community

Raspberry Pi Model B revision, University of Cambridge¡¯s Computer Laboratory, UK, 2012. [Online] Available:http://www.raspberrypi.org/wp-content/uploads/2012/12/quick-start-guide-v1.1.pdf and http://elinux.org/RPi_Community

STUN protocol (bis) RFC 5389, IETF J. Rosenberg et.al, [Online]. Available: http://tools.ietf.org/html/rfc5389

STUN protocol RFC 3489, IETF, J. Rosenberg et.al, 2003, [Online]. Available: http://tools.ietf.org/html/rfc3489

The LLVM Compiler Infrastructure Project. [Online]. Available:http://llvm.org/

TURN protocol RFC 5766,IFTF, R. Mahy et.al. 2010. [Online]. Available: http://tools.ietf.org/html/rfc5766

WebRTC community. [Online]. Available: http://www.webrtc.org

WebRTC tutorial, HTML5 Rocks, [Online]. Available:http://www.html5rocks.com/en/tutorials/webrtc/basics/

XC8 Compiler, Microchip Technology Inc, AZ. 2012. Online. Available:http://ww1.microchip.com/downloads/en/DeviceDoc/52053B.pdf

Appendix A R&V

A.1 Requirement and Verification

Table 4. Project requirement and verification table

Requirement	Verification	Verification status
1. Power Module **<u>1.a</u>** Five AA Batteries provide enough energy to run the motor at maximum load for at least 10 minutes	**1. Power module** **<u>1.a</u>** From Power Budget section, the five AA batteries have 7.8 Wh. The requirement passes If total peak power of subcomponents adds up to be less than 7.8W*6=46.8W.	1. Power module Verified (section A.2.1)
<u>1.a.1</u> Motor's peak power should be less than rated power: 4*(6+/-0.5)*0.47 =11.28+/-0.94W	**<u>1.a.1</u>** Measure each the motor's current and voltage using multi-meter (Fluke 12) with maximum loads running on pile floor, and then multiply the current and voltage to get peak power, which should less than 11.28 for four motors	Verified (Table 6)
<u>1.a.2</u> The output voltage needs to be above 3.0 V for the duration of 10 minutes; otherwise, motor will not move	**<u>1.a.2</u>** Use multimeter to take the voltage output after 10 minutes running at maximum loads and on pile floor, and then measure voltage level.	Verified (Figure 15)
<u>1.a.3</u> Output voltage's ripple from the AA batteries should vary within +/- 500mV	**<u>1.a.3</u>** Use oscilloscope to take of the voltage output for 10 minutes running at maximum loads and on pile floor	Verified (Figure 15)
<u>1.b</u> Microcontroller power pack provides enough energy to run the microcontrollers run at	**<u>1.b</u>** From the Power Budget section, the MHK power pack has 2.2 Wh and the 9V energizer battery has	

maximum power for at least 10 minutes while providing at least 700mA at 5V. **1.b.1** RPi's peak power should be less than 3.5W **1,b.2** Arduino's peak power should be less than 3W **1.b.3** Each sensor's total peak power should be less than 15 mW	0.5 Wh. The requirement passes if total peak power of subcomponents adds up to be less than 2.7*6=16.2W. **1.b.1** Use Tektronix TM504 instrument to measure the USB power input pin of RPi while performing data streaming on RPi. **1.b.2** Use Tektronix TM504 instrument to measure the USB power input pin of RPi while performing real time controlling **1.b.3** Use multi-meter to measure the voltage and current at the "Vcc" pin while letting the sensor detect a close ranged object.	Verified (Section A.2.2) Verified (Table 7) Verified (Table 7) Verified (Table 7)
2. Motor system Module **2.a** Motor is not damaged under supply voltage and current	**2. Motor system module** **2.a** Motors performance correctly is the basic requirement; ensure using five fully charged AA batteries to test. Disconnect motors from circuit, and then draw two wires from positive and negative sides of the battery carrier. Connect motors directly to AA batteries. Check and make sure motor could rotate; otherwise motors are damaged.	Verified

	2.b Ensure requirement **2.a** is verified; turn on vehicle and turn on microcontroller.	Verified (A.2.3)
2.b Ability to operate continuously between 50%—100% duty cycle; requirement will be verified if all sub-requirements are verified:	**2.b.1** No disconnection between gears will guarantee motors not slip at 100% duty cycle. Slowly rotate each tire by hands to check if there is any disconnection at gears by listening and observation	Verified
2.b.1 All gear teeth are fully engaged.	**2.b.2** Gate voltage higher than 3.3V will guarantee current flow across H-bridge; use multi-meter (Fluke 12) to test. Draw wires from pin 3 and 9 of each H-bridge. Draw a wire from RPI ground. Connect the positive port of the multi-meter to this wire and connect the negative port to common ground. Measure voltage and record data. Measurement should always be larger than 3.3V	Verified (Table 4)
2.b.2 Gate voltage at each MOSFET must be larger than 3.3V	**2.b.3** A 100% duty cycle square wave is necessary to provide full PWM; use oscilloscope (Agilent 54642A) to test.	

2.b.3 PWM pin is able to output a square wave with 100% duty cycle	Setup PWM at 100% duty cycle by set digital duty cycle at binary 1111. Then draw a wire from PWM output from Arduino unit. Connect positive wire to PWM output and negative wire to common ground. Press 'auto-scale' on oscilloscope. Press 'quick measurement' and chose "duty cycle". Measurement should be 100% +/- 0.1%.	Verified (Figure 16, Table 8)
2.b.4 PWM should have minimum frequency of 200Hz	**2.b.4** Minimum frequency of 200Hz will guarantee motor run continuously (without clicking); use oscilloscope (Agilent 54642A) to test. Draw a wire from PWM output from Arduino unit and connect positive wire to PWM output and negative wire to common ground. Press 'auto-scale' on oscilloscope. Press 'quick measurement' and chose 'frequency'. Measurement should have minimum of 200 Hz.	Verified (Figure 17, Table 9)
	2.c Ensure requirement **2.a** and **2.b** are verified; temperature limit must be verified to protect motor from damaging by heat. Set up PWM at maximum (100%) by set the digital duty cycle at binary 1111. Attach a mercury thermometer at the surface of the motor with tape. Measure the motor surface temperature before motor is turned on. Turn on motor and let it run until battery dies out. Record the temperature of the mercury thermometer, which is the maximum operation temperature. Measurement should not be 10 ± 10% degrees above ambient temperature	Verified (Table 9)

2.c Operate within reasonable temperature at maximum duty cycle (10 degrees +/- 10% above ambient temperature	**2.d** Ensure requirement 2.a and 2.b are verified; lagging tolerance must be verified to guarantee user-friendly control experience. Log in to our website and click User Driving Mode. Enter “w” and Server starts a stopwatch at t1. Server stop stopwatch at t2 when it receives the request from the car. Round Trip Time (RTT)=t2-t1. The RTT should not exceed 1 ± 0.1s	
	2.e Probe the motor’s positive and negative pins while running them; acquire waveform and measure frequency and maximum voltage level. The measurement result should be less than 1 Hz and 1 V.	Verified (A.2.4, Table 10)
2.d React to control signal within reasonable lagging tolerance (less than 1 +/- 0.1s)		
		Failed (A.2.5, Figure 18)
2.e Noise floor generated by motor spinning must be limited below 1 Hz and the noise signal amplitude must be limited below 1 V.		

Otherwise, noise will significantly affect the performance of the vehicle		
3. Navigation System **<u>3.a</u>** The Navigation System is functional on MV **<u>3.b</u>** User should be able to use auto-navigation mode. **<u>3.b.1</u>** GPS should be able to accurately track the moving platform within 10 meters accuracy	**3. Navigation System** **<u>3.a</u>** Read out input signal from Arduino, which returns if a signal has been received **<u>3.b</u>** Ensure Internet accessibility on the MV; checks users' browser for Google Map support. User enters a location, the MV should move toward that direction **<u>3.b.1</u>** Using GPS Navigation Mode, record longitude and latitude and enter a new long. And lat. within 2''. After the MV comes to a stop, record its new location. Finally, check the accuracy via Google Map. The difference should be within 10 meters.	Verified (A.2.6, Table 11, Figure 19) Failed (A.2.7) Failed
4. Webcam **<u>4.a</u>** Webcam with USB connection is able to perform correctly	**4.Webcam** **<u>4.a</u>** Webcam working normally is the basic requirement; ensure using a computer with Windows XP or higher version to test. Connect webcam to the computer USB port. Windows will automatically detect webcam as new hardware. Click "Open new hardware" at the prop-up interface. Webcam should be started; check the camera window on the computer screen to see whether it	Verified

	is working	
4.b Webcam should provide at least 10 frame/s at resolution less than 640X480 to guarantee fluent vision	**4.b** Ensure **4.a** is verified and connect webcam back to our project. Type 'Sudo apt-get fswebcam' to Run fswebcam(version 20110717) in shell : Fswebcam –r 640x480 –d /dev/video0 test.jpg –fps 10 Fswebcam –list-framesizes We can see the camera can support up to 640x480 and 10 frames/s is barely on the edge of the camera's ability. Grab image larger than 480*320 I.e. 960*480 can clearly see the webcam fails to respond.	Verified (A.2.8, Figure 20)
5. Sensor system **5.a** Sensor System should function and respond to obstacles.	**5. Sensor system** **5.a** Supply 5 V to Vcc of a sensor; connect AN output to oscilloscope, and then move a book back and forth in front of the sensor. The DC level on the oscilloscope should decrease and increase.	Verified (A.2.9)
5.a.1 Sensor should detect obstacles' range accurately (+/- 20%) down to minimum of 10-12cm. Below minimum distance, the DC level should stay within +/- 0.1%.	**5.a.1** Connect AN output of sensor to oscilloscope; then carefully bring a book from about 15 cm toward the sensor. Observe the DC level on the oscilloscope as the distance of the book approaches 10 cm. When It's at about 10 cm, the DC level should reflect the range within 2%. The reading should stay within +/- 1% of its DC level when the distance is less than 10 cm.	Verified (Table 12)

5.a.2 Sensor should detect obstacles' range accurately (+/- 10%) up to maximum of 50 cm. The distance away from sensor should reflect linearly on the oscilloscope reading at Vcc/512 per inch	**5.a.2** Connect AN output of sensor to oscilloscope; then carefully bring a book from about 15 cm away from the sensor. Observe and record the DC level on the oscilloscope. When the book is at about 25 cm away, check the DC level. It should be about 39 mV +/- 10% higher the original. In addition, when the book is held steady, the ripples of DC level should stay within +/-0.1%	Verified (Table 12)
5.b The Sensor System should correspond with the RTC at maximum of 1 msec (immediate)	**5.b** Connect Sensor System to the RTC and power on both. Move a book back and forth from the sensor. Set a timer program in Arduino to record the reaction. The result should be almost immediate to human reaction time (<1 msec).	Verified
6. Microcontroller	**6. Microcontroller**	
6.a RPi powers on and functions as normal	**6.a** Plug in power supply for RPi, and make sure it's powered on.	Verified
6.b WIFI adapter functions and passes information between RPi's server and user's server	**6.b** Use SSH to pin RPi and there should be a response	Verified
6.c SD memory card transfer rate is higher than class 4 and memory is larger than 4GB	**6.c** Insert SD card into PC and checks the storing capacity. Copy a large file to check the average transfer speed to be higher than 4 MB/sec.	verified

7. Website **<u>7.a</u>** Website should be functional. **<u>7.a.1</u>** The user can create account and log in to take control a of an available MV	7. **Website** **<u>7.a</u>** User can go to the website afallon.yzi.me **<u>7.a.1</u>** User goes to the website, creates an account, and then finds an available MV to control.	Verified (A.2.10) Verified

A.2 Functionality proposed and verification

Table 5 Functions proposed and verification

Proposed functions	Verification status
Travel in different direction and speed	Verified
Website	Verified
Live Webcam	Verified in local network, failed in World Wide Web
Real-time manual control	Verified
Programed route control	Verified
Automatic trace back	Verified
GPS automatic navigation	Failed
Collision avoidance system	Verified
Additional feature: Google talk control	Verified

A.2 Verification measurements and results

A.2.1 Power module 1.a measurements and verifications

We measured all motor voltages, currents and power by using multimeter and compared the total power with maximum power requirement. The result was consistent so this requirement had been verified. Measurement data is tabulated in Table 2.

From figure 1, we can see the battery voltage output after 10 mins motor spinning is 6.47 V, which is above the minimum requirement of 3V. The voltage ripple is equal to the detected maximum voltage level minus minimum voltage level. The result is 0.11 V and it is below the requirement of 0.5 V.

Table 6. Measurement of motor power parameters

Motor	Voltage(V)	Current (A)	Measurement Power (W)	Maximum power (W)
Front-left	6.49	0.37	2.40	-
Front-right	6.49	0.30	1.95	-
Back-left	6.49	0.35	2.27	-
Back-right	6.49	0.42	2.72	-
Total	-	-	9.34	11.34

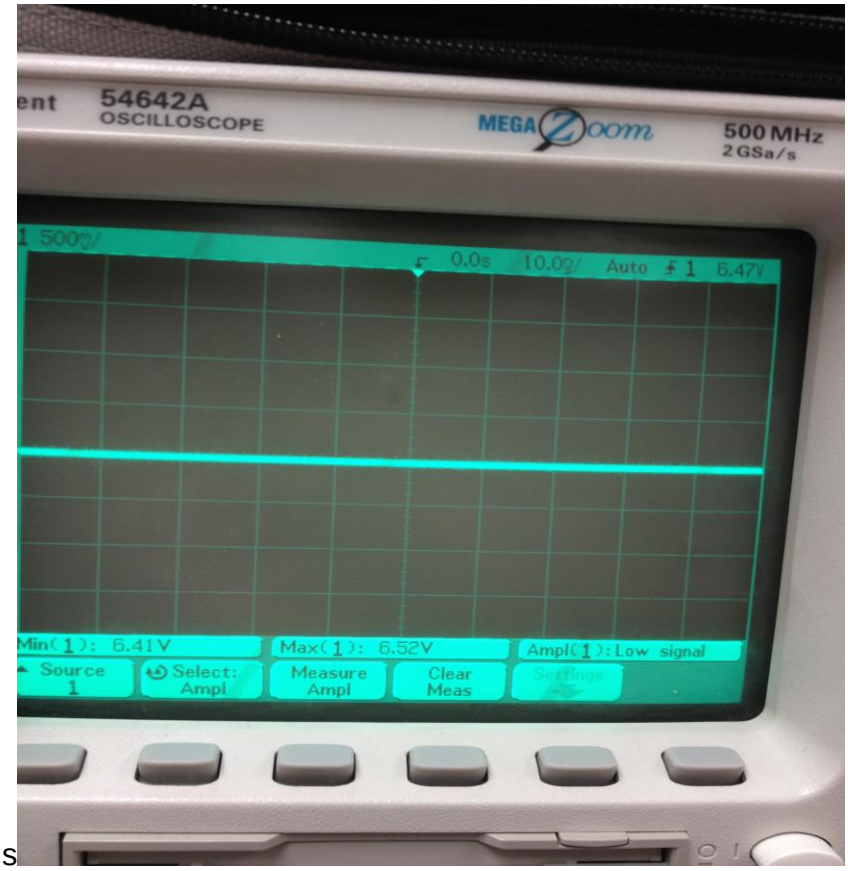

Figure 15. Battery voltage level after 10 minutes running

A.2.2 Power module 1.b measurements and verifications

We measured all microcontrollers' and sensors' voltages, currents and power by using Tektronix TM504 probe and compared the total power with maximum power requirement. The result was consistent so this requirement had been verified. Measurement data is tabulated in Table 3.

Table 7. Measurements of microcontroller power parameters

Components	Voltage(V)	Current (A)	Measurement Power (W)	Requirement power (W)	Maximum power (W)
RPI	5.2	0.42	2.18	3.5	-
Arduino mega 2560	8.9	0.20	1.78	3	-
Front-Sensor	4.8	0.002	0.010	0.015	-
Left-Sensor	4.8	0.003	0.014	0.015	-
Right-sensor	4.8	0.002	0.010	0.015	-
Bottom-sensor	4.8	0.003	0.014	0.015	-
Total	-	-	4.008	6.56	16.2

A.2.3 Motor module 2.b measurements and verifications

We measured each MOSFETs' voltage and measured the temperature of each motor. Then we compared the data with requirement. The result was consistent so these requirements had been verified. Measurement data is tabulated in Table 4 and 5.

From figure 2 and 3, we can see the maximum PWM duty cycle provided by Arduino is 100% and the frequency of the PWM is 490 Hz. Those measurement results can verify the basic requirements of having a 100% duty cycle PWM with frequency greater than 200 Hz.

Table 8. Measurements of H-bridge gate voltage

MOSFET	Left-1	Left-2	Left-3	Left-4	Right-1	Right-2	Right-3	Right-4
Measurement voltage(V)	4.8	4.8	4.8	4.8	4.9	4.9	4.9	4.9
Required voltage(V)	3.3	3.3	3.3	3.3	3.3	3.3	3.3	3.3

Table 9. Measurement result of rotor performance parameters

Motor	1	2	3	4	Requirement
Ambient temperature (F)	79	79	79	79	-
Measurement temperature (F)	85	85	85	85	89
PWM duty cycle measurement	100%	100%	100%	100%	100%+/- 0.1%
PWM (Hz)	490	490	490	490	200

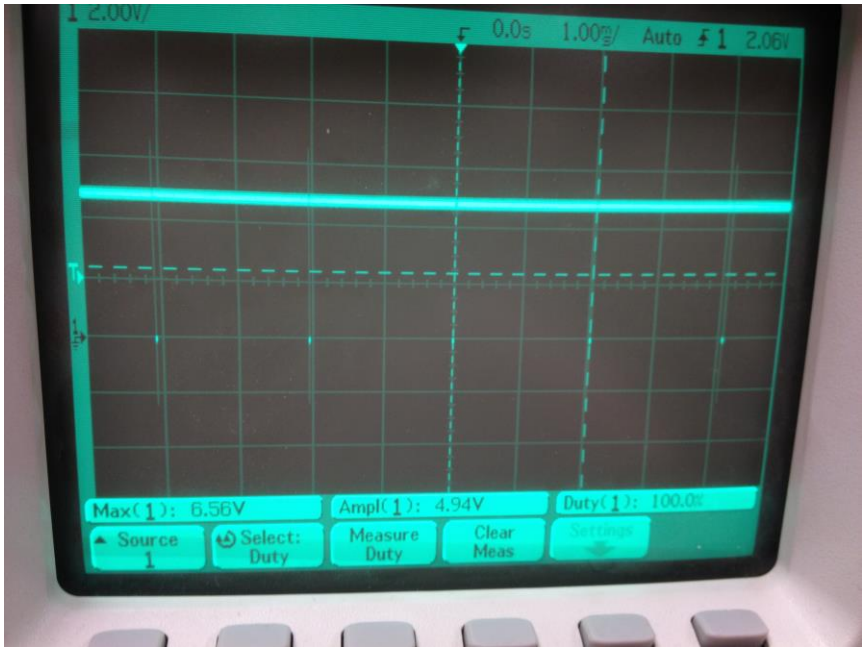

Figure 16.PWM maximum duty cycle measurement

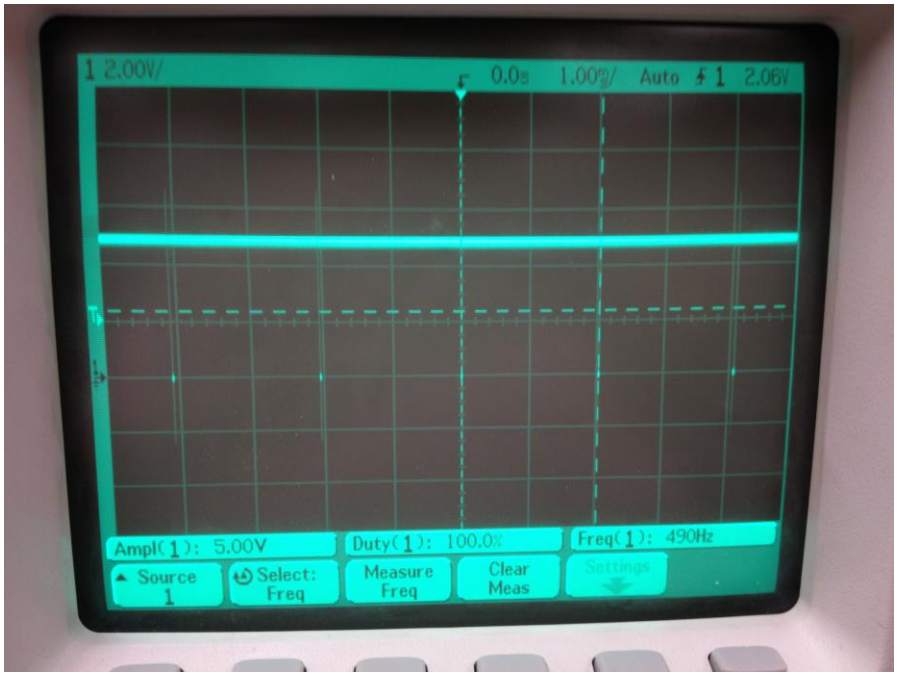

Figure 17. PWM frequency measurement

A.2.4 Motor module 2.e measurements and verifications

Table 10. Measurement of reaction time

Measured reaction time	Desired reaction time	difference
0.83s	1s +/- 0.1s	0.17s

A.2.5 Motor module 2.e measurements and verifications

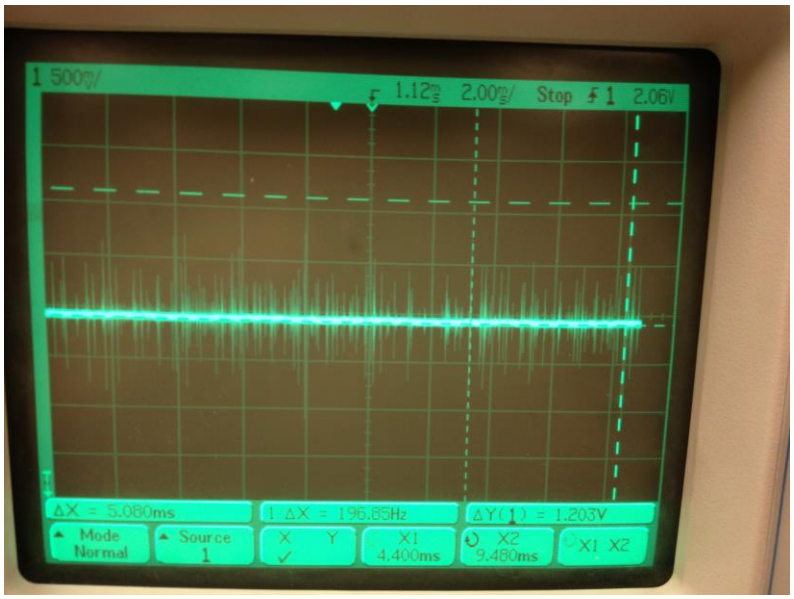

Figure 18. Measurement of motor noise floor

From Figure 19 we can see the measurement result of maximum noise amplitude is 1.23 V and the frequency for maximum noise is about 196.85 Hz. We failed this requirement because we did not anticipate the significance of noise floor interference with microcontroller. We did not specifically design a suitable low pass filter to reduce noise. As a result, the noise generated by motor spinning affected data transmissions between RPi and Arduino; therefore some of our functions may crash.

To compensate noise signal interference, we used several ways to limit its effects: we added 0.1 µF decoupling capacitors between every motor pins and ground; we tried to link all ground to a separate metal; we used a “handshake protocol” and “repetition error check” between two microcontrollers to secure data transmission. These methods greatly reduced the noise interference but they could not entirely prevent unexpected errors from happening.

A.2.6 Navigation module 1.a measurements and verification

We placed the antenna at the north window of room 246, Everitt lab. Comparing with Google map mapping, the received GPS data was accurate; therefore, GPS module was working and the this requirement was verified.

Table 11. GPS response from Arduino reading

	Measurement from EM408 GPS module
Latitude (degree)	40.1101406616
Longitude (degree)	-88.2282273437

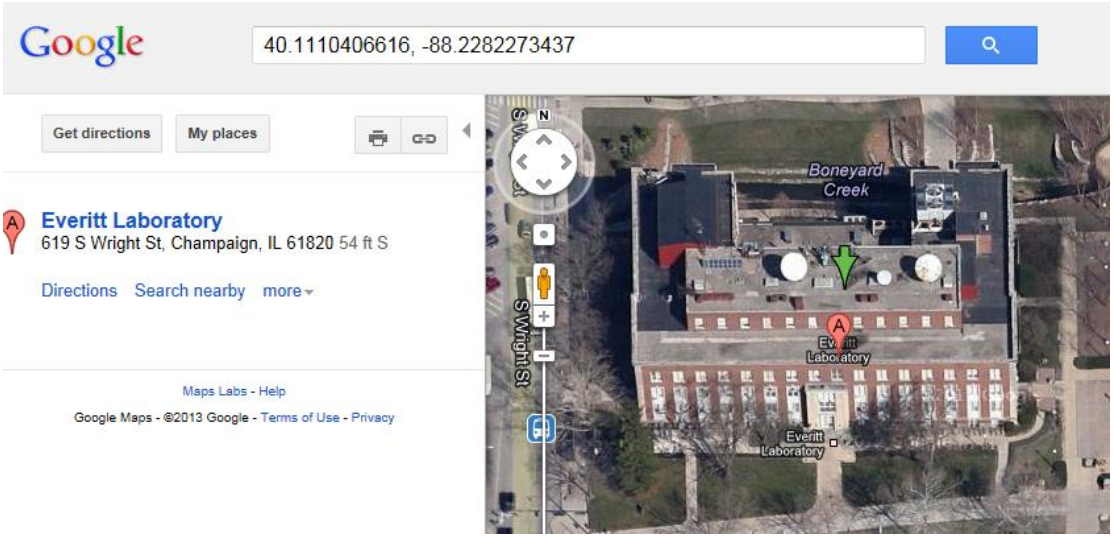

Figure 19. Coordinates measurement verified by Google map

A.2.7 Navigation module 3.b measurements and verifications

Automatic navigation algorithm failed due to unstable Serial Peripheral Interface Bus transmission. The compass module accuracy was also a big concern. The detailed discussion about all failures are attached as appendix at the end of the report.

A.2.8 Webcam module 4.b measurements and verifications

Figure 20: A random picture (above) with emphases shadow contrast taking from our webcam. We are compressing 320*240 JPEG image with 80% rate to ensure lower bandwidth consumption (transmitting 20KB*8 frames/s= 160 KB/s with wifi via ftp streaming). The result is considered satisfactory in streaming. In gstreamer, we are using raw image streaming with gst pipelines with about 200KB/s bandwidth consumption.

A.2.9 Sensor 5.a measurements and verifications

Table 12. Ultrasonic sensor distance measurement

	Sensor front	Sensor left	Sensor right	Sensor down
Digital Measurement (cm)	22.4	22.3	22.1	23.0
Actual distance (cm)	25.0	25.0	25.0	25.0
Error (%)	10.4%	10.8%	11.6%	8%
Digital measurement (cm)	10.1	10.2	9.9	10.0
Actual distance (cm)	10.0	10.0	10.0	10.0
Error	1%	2%	1%	0%

A.2.10 Website 7.a measurements and verifications

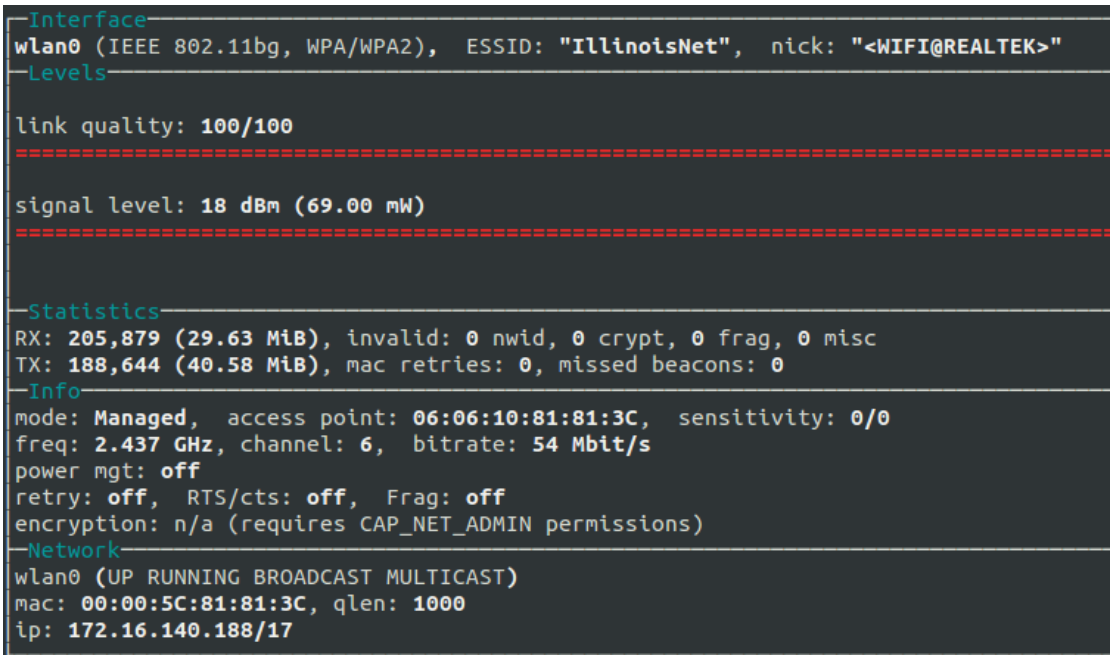

Figure 21: Linux wavemon check. RX/TX rates satisfy our program's running requirements, which is typically 2~3KB/s. Also the signal level 18dBm with link quality 100% guarantees fluent video streaming which is typically requires roughly 70% or above link quality.

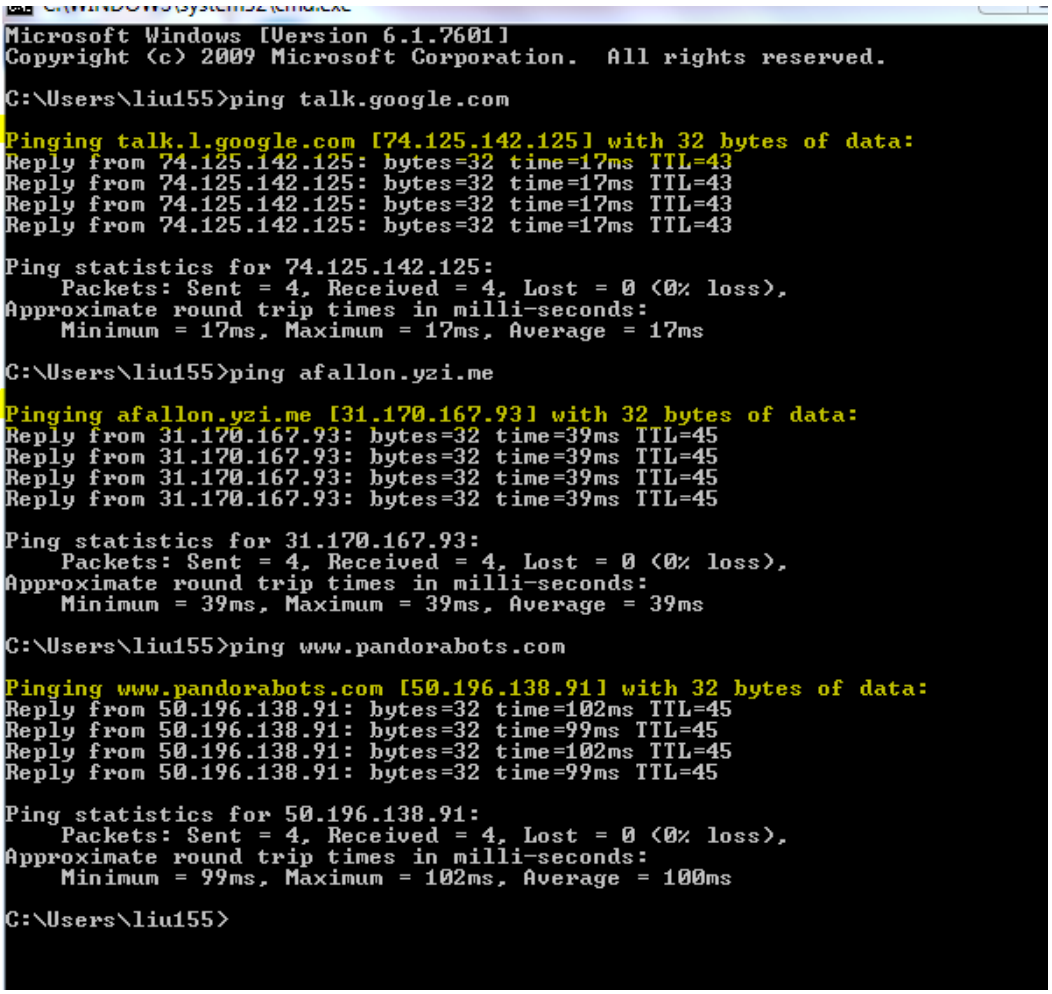

Figure 22: Pinging the three heart servers providing services to our program. talk.google.com relays our instant messages, afallon.yzi.me, which is our own server, hosting data management and common knowledge base API to our robot, and the pandorabots.com hosts the user-configured main artificial intelligence personality . Packets' RTT(Round-Trip time) are within 0.5 s.

Appendix B Network System

Network system is composed by two parts, the server side software and the client side software. The server side software, client side software and the real-time software all together form a Server-Client-Real-time three-level hierarchy architecture.
In this architecture, the server has the highest processing ability, the largest storage capacity, and the most convenient massive data exchange channel, so it is responsible for complex computing, large data storage and massive interaction with multiple clients/users. It is the brain of the robot.
The client software, which located on Raspberry Pi, is taking care of massive data interpretation, encoding/decoding, data transmit/data reception, and multi-thread tasks management. Specifically, it translates the instructions into pieces of real-time tasks for the lower level real-time system to do while performing data transmission. It is the cerebellum of the robot.
The real-time system on the lowest level, which has the lowest processing ability and zero storage, is responsible for real-time critical task and low level environmental data collection. It is the combination of spinal cord, sensors, and reflexive nerves of the robot.

B.1 Server Side Software

It composed by the application engine, the instant messaging relay and the real-time video stream broadcast

B.1.1 The Application Engine

The Application Engine is an abstraction of a combination of software services. It represents applications that separate in several geographically distributed hardware nodes.

B.1.2 Pandorabots.com

It is a XML server providing Artificial Intelligence Service. It is a service provided by A.L.I.C.E foundation. Server accepts and responses query following XML-RPC (Extensible Markup Language – Remote Procedure Calling Standard). Our Robot's AI is programmed using AIML (Artificial Intelligence Markup Language).

```
<pattern>WHOSE *</pattern>
<template>
        <random>
                <li>Do you mean "who is"?</li>
                <li>I don't know whose.</li>
        </random>
</template>
```

B.1.3 Afallon.yzi.me

It is a XML server providing various navigation related RESTful services. Our own server it provides RESTful services to HTTP GET, written in php.

B.1.4 Performing backward navigation algorithm based on trace analysis.

If backward navigation is triggered, the platform will automatically make a 180 degree turning and perform the recorded commands backward.
In detail, if the recorded command is go forward or go backward, at backward navigation mode, platform will perform the same command without any change; if the recorded command is go left or go right, at backward navigation mode, platform will revert the direction to its opposite direction.

B.1.5 Performing patterned navigation based on common knowledge system. (XML)

For example, consider the scenario:

1. When the user tells the robot to “draw a square with 1 meters edges on the floor”, the client software of the robot will ask the artificial intelligence engine “What does this sentence mean?”
2. The artificial intelligence engine will parse the sentence to the robot client software in the language it understands. The robot now knows it needs to “do something”. For “doing anything” more complex than direct instructions like “forward 10 meters/retreat 10 meters”, it is defaulted to query the common knowledge base located in remote server for answers.
3. It then asks the knowledge base about the method of “draw” with the parameter “circle” and “size 1 meter”. The server processes its requests and return a list of instructions in XML format:

```
−<response status="1">
    <step attr="0">0,0,0,1,0,1,0,0</step>
    <step attr="1">0,1,90,1,0,1,0,0</step>
    <step attr="2">0,1,90,1,0,1,0,0</step>
    <step attr="3">0,1,90,1,0,1,0,0</step>
  </response>
```

4. The robot “understands” the command and now it performs the sequence of instructions.

For now, our infantry robot only “understand” behavior “draw *” with “*” being square, triangle or simple Latin letters like “L, I”. In our architecture, with the help of distributed networking and data mining algorithms, our robot can break the bound of on-board processing and on-board storage and evolve to have infinitely exponentially growing knowledge with unlimited processing speed.

B.1.6 Data logging of instructions and robot status. (SQL database)

The logging triggers every time the robot status or the instruction changes. With the collecting of such data, the server can “learn” what the robot does and perform operation on path memory like backward navigation. This knowledge can even be transferred to the knowledge piece in common knowledge base if the robot is “trying” some new operations and the result of trying is successful. Such features are only available in GPS navigation currently.

B.1.7 Session Management

We identify each piece of information uploading with uniquely ID composing by “robot id, session number, operation’s sequence number” so that the non-collision concurrency is guaranteed so that it can support any arbitrarily number of robot accessing the service at the same time.

B.1.8 Google Map GPS Navigation.

Google map’s server for GPS navigation is used. It sends JSON data for navigation instructions in response to HTTP POST query.

B.1.9 Instant Messaging Relay:

The talk.google.com’s server is used. The server relays messages from client following google’s XMPP standard.

B.1.10 Real-time video streaming broadcast:

The Bambuser.com’s streaming server is used. The server broadcasts streaming from client.

B.2 Client side software:

B.2.1 Main Entity Client

An open source gtalk python library: PyGtalkRobot is used as the framework, adding our own video capabilities support. The main entity of the client software inherent the PyGtalkRobot class to interact with commands coming from users on gtalk. The main entity involves functions interacting with Application Engine and functions interacting with the lower level real-time system via SPI.

B.2.2 Video Streaming Client

Video streaming: A linux rtmp video streaming tool called FFMPEG is used to transmit video to the streaming server. The frame rate is set to 10 frames/s to save bandwidth for the main entity.

Appendix C PCB

Print Circuit Board (PCB)

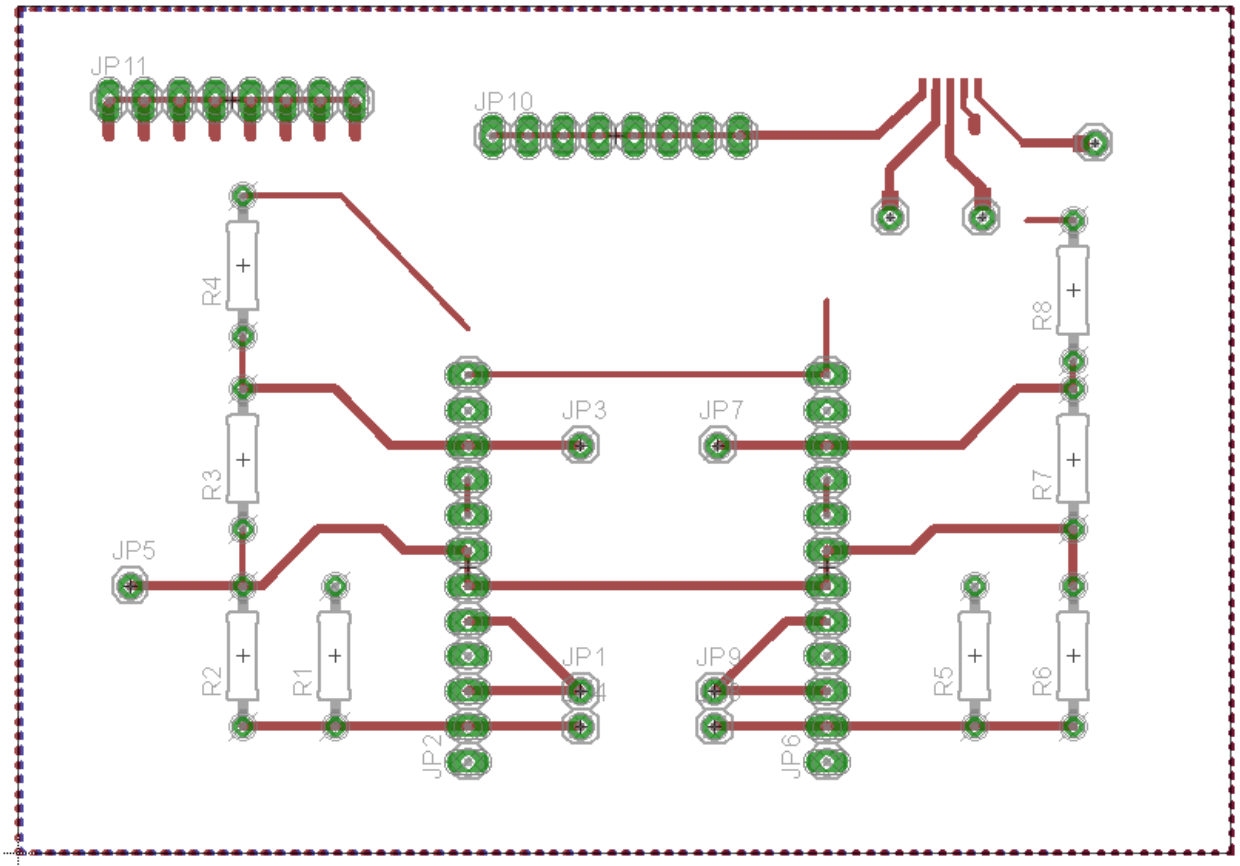

Figure 23 PCB of H-Bridge

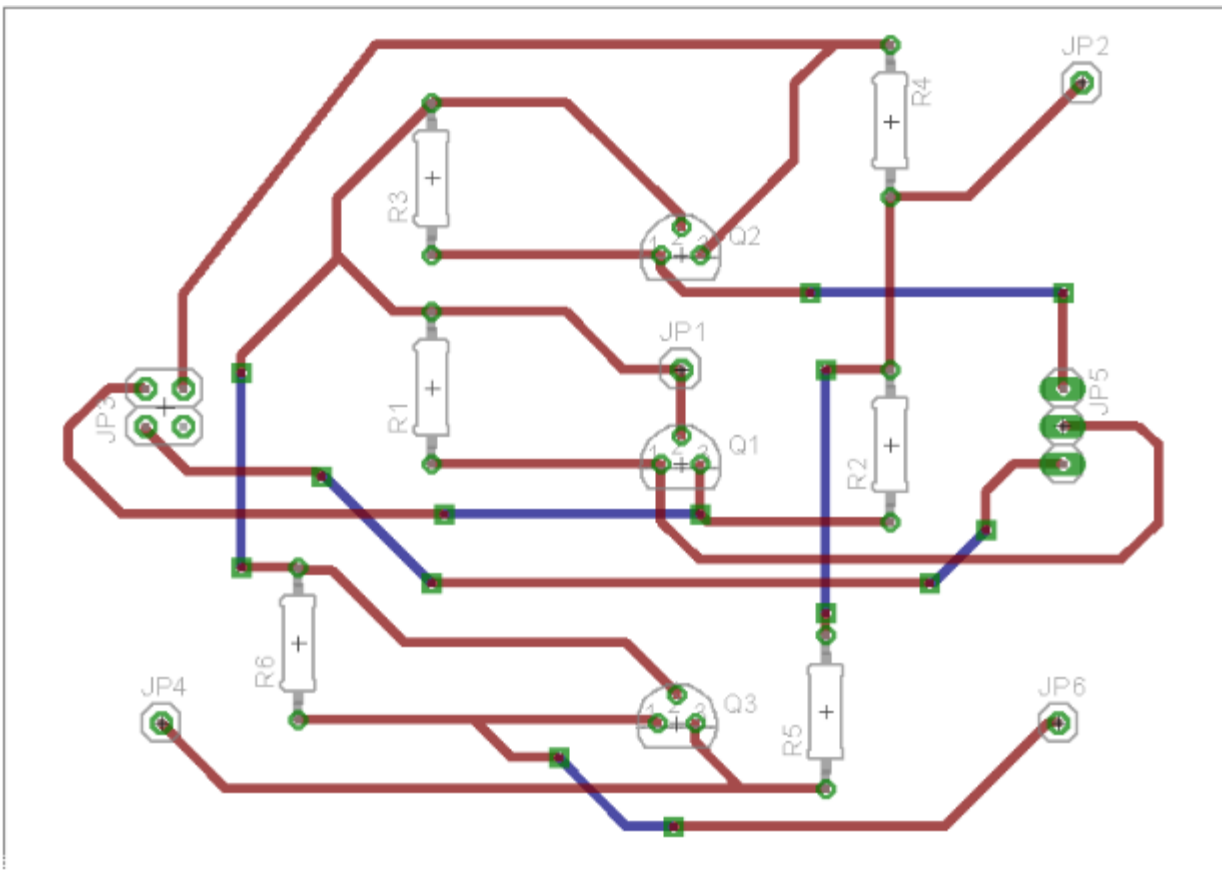

Figure 24 PCB of Level Shifter