EECE 281 L2B
University of British Columbia

Dr. Jesús Calviño-Fraga
Dr. Joseph Yan

Electromagnetic Tether Robot

April 8, 2013

Andy Au
Daniel Chong
Gregory Lee
Jae Yeong Bae
Jeffrey Chan
Red (Kyle) Garsuta

Contents

1 Introduction 2
1.1 Specifications 2
1.2 Overview and Design Approach 3
2 Investigation 3
2.1 Idea Generation 3
2.2 Investigation Design 4
2.3 Data Collection 4
2.4 Data Synthesis 5
2.5 Analysis of Results 5
3 Design 6
3.1 Use of Process 6
3.2 Need and Constraint Identification 6
3.3 Problem Specification 7
3.4 Solution Generation 8
3.5 Solution Evaluation 10
3.6 Detailed Design 10
3.6.1 Transmitter 10
3.6.2 Receiver 11
3.6.3 Motors 12
3.6.4 Software 13
3.7 Solution Assessment 15
4 Life-long Learning 15
4.1 Software 15
4.2 Hardware 16
5 Conclusions 17
6 References 18
7 Bibliography 18
8 Appendices 20
8.1 Appendix A: Robot Movement Sample Code 20

1 Introduction

This project involves designing, building, and testing an electromagnetic tether robot. The Tether robot must keep a fixed distance from an electromagnetic beacon. This section describes the requirements and specifications for the Electromagnetic Tether Robot.

1.1 Specifications

The robot must be implemented with the following specifications[1]:

1. **AT89LPC828 Microcontroller Systems.** Microcontroller systems must be built and assembled using a solder-board.
2. **Battery operated.** The robot must be battery operated.
3. **Robot construction.** Any materials may be used in implementing the chassis of the robot. A default chassis is provided in the EECE machine shop but you may choose to design your own.
4. **C programming.** The software for this project must be written in C.
5. **MOSFET Drivers.** You must use Metal Oxide Semiconductor Field Effect Transistors (MOSFETs) must be used to drive and control the motors of the robot. Both NMOS and PMOST transistors are available and may be used.
6. **Transmitter.** A separate microcontroller system must be used to transmit an electromagnetic signal to the robot. The transmitter must be able to send commands to the robot:
    a. Move closer
    b. Move farther

[1] Jesus Calvino-Fraga, *Project 1: Automatic Soda Fountain Dispenser* (University of British Columbia, 2013), p.1.

7. **Distance.** The robot must be able to track the distance from the transmitter at a minimum separation of 20 cm.

1.2 Overview and Design Approach

We started by estimating the size of the main circuit boards to determine the overall dimensions of the project. Once this was done, we created 3D models of the project using AutoCAD Electrical, and agreed upon the general appearance of the robot. The team was split into two main groups: 4 members in charge of software, 2 members in charge of hardware.

Once the project functionality was decided, each group worked simultaneously while coordinating project developments to assure overall compatibility. Section 2 of this report delves into the design process, while detailed descriptions of the software and hardware implementations are described in Section 3.

2 Investigation

This section describes our team's approach in gathering relevant data for the Electromagnetic Tether robot. The design process can be broken down as follows: (1) Idea Generation, (2) Investigation Design, (3) Data Collection, (4) Data Synthesis, and (5) Analysis of Results. The following subsections discuss each of the above in detail.

2.1 Idea Generation

We initialised this project through brainstorming sessions where ideas were pooled, discussed and refined; ideas unreservedly thrown at each other until a consensus was achieved. To distribute work more effectively, the workload was split into hardware and software; this allowed our group to utilise each member's talent more effectively. Drafting our ideas into

schematic diagrams and flowcharts allowed us to solidify our ideas and prepare to move onto the next step.

2.2 Investigation Design

To construct our project, we first researched the specifications of the parts that were available to us. Values such as maximum current and voltage allowed across components, were important to prevent damage to the design of the circuits. This allowed us to prototype circuits on the breadboard to test and collect data about the components and parts.

Inquiries were made to different retailers for parts that were required to complete certain designs, for instance: one of our designs required tank treads instead of the wide wheels that we were given, however we were unable to find a retailer that sells the parts at a reasonable price; eventually we had to abandon the design and switch to a simpler one.

2.3 Data Collection

Data of components and parks was collected from using equipment available in the lab. An example of this would be the determination of the resonant frequency of the transmitter circuit. The resonant frequency of the circuit was determined through:

$$f_0 = \frac{\omega_0}{2\pi} = \frac{1}{2\pi\sqrt{LC}}.$$

This formula allows us to determine the theoretical resonant frequency when the inductance and the capacitance is known. However, since there are deviations between the stated value and the actual value of the components, we first used a capacitance meter and inductance meter to determine the values, then used a signal generator for fine tuning, and found the

frequency to be 6907.5 Hz. Then by using this frequency, we were able to program the transmitter to transmit at maximum peak to peak

2.4 Data Synthesis

Data synthesis for our electromagnetic tethered robot was done by examining and combining the major parts of the project. To begin with, the major pieces of the project included the transmitter, the receiver, the motors, and the software. The transmitter and receiver had to be synthesized by determining how far the receiver would detect the transmitter, which ultimately determined our range. The transmitter produced a sine wave with a peak to peak value of approximately 120V, which was taken into account when determining the amplitudes for closer positions. The receiver then had to send data to the motors, where the motors would have to turn in a certain direction based on the received signal from the receiver. This was done by determining the peak value of the received sine wave which was then converted into an integer value by an ADC. All three of these parts work in harmony with the software. The software controls the strength of the transmitter, determines how to interpret the receiving signal, and finally controls the directions of the motors.

2.5 Analysis of Results

The results produced from each major component were generally acceptable with some fall backs. In particular, the founded 120V peak to peak value of the transmitter was much smaller than expected. While this was useable, a much higher peak to peak value was needed to obtain the minimum range requirement (around 180V was proven to be adequate). Furthermore, this came to affect our receiver which generated more noise at farther distances due to the relatively weak source. Ultimately, this cause erratic movements in our robot at times. While most of this

was dampened with tolerance levels in the software itself, it was impossible to eliminate due to the original problem of the weak transmitter.

3 Design

This section describes our team's approach in designing the Electromagnetic Tether robot. The design process can be broken down as follows: (1) Use of Process, (2) Need and Constraint Identification, (3) Problem Specification, (4) Solution Generation, (5) Solution Evaluation, (6) Detailed Design, and (7) Solution Assessment. The following subsections discuss each of the above in detail.

3.1 Use of Process

With both software and hardware components to complete, our team derived the work into two main components. While each member played a significant role in either software or hardware, everyone was given the opportunity to work on both sections. This allowed for our team to draw ideas from all members of the group while being able to focus on more specific tasks. By separating the project into multiple components, the project also became very modular which allowed for our group to easily test any components that were not working. This also made it easy to understand what each component did as we only needed to know the inputs and outputs of any part.

3.2 Need and Constraint Identification

The identified needs and constraints are as follows:

1. **Robot Movement.** We designed our robot to use two distances between two solenoids on our robot and a solenoid on our transmitter. This meant that our robot would need to,

using only two distances, follow the transmitter regardless the position of either the robot or the remote.

2. **Physical Appearance.** As a group, we wanted a robot that was physically unique. We did not want a project that was similar to everyone else's. From a consumer's point of view, something different that stood out would be more interesting than something that was common and plain. Using the chassis that was provided for us was too ordinary and easy, so we decided to come up with a different looking design. We came up with the idea of making the appearance of our robot close to that of a Roomba, a type of robot by iRobot. Making our robot in the shape of a circle gave it a much sleeker look than using a simple rectangular chassis that was given, which would allow for our project to physically stand out more than others. In addition, a remote control had to be constructed for the user to operate our robot. This was also done in the machine shop but was trivial to produce.
3. **Battery Limitation.** Unfortunately, our group discovered that using a single 9V battery was not enough to power the robot as well as extra LED's that we had planned to include. Because of this, we had to either add more batteries to the robot or limit the power consumption.

3.3 Problem Specification

Identified problem specifications are as follows:

1. **Robot Movement.** It was easy to determine if the robot was not facing the remote directly as any difference between the two distances meant that the robot needed to adjust its position by turning. However, once it reached this position, whether it would need to move forward or backwards was a problem as the distance measured was independent of whether the robot was in front or behind the remote.

2. **Physical Appearance**. Designing our robot in a circular shape present a couple of problems for us. Other than the water jet cutter, we do not have access to tools that would allow us to make circular shape for our robot. As a result, we had to send in our design to the machine shop and have them make it for us as we do not personally have access to the water jet cutter. The controller did not need to be sent in. Instead, we built it ourselves, but ensuring that it would not be too big for one's hand but not too small so the required circuitry would not fit.
3. **Battery Limitations.** We had to choose between either adding batteries to the robot or limiting the robot's power consumption. Since we chose to not add any more batteries, an additional requirement that we had was to ensure the robot used only a certain amount of voltage.

3.4 Solution Generation

1) **Robot Movement.** In order to determine whether the robot was in front of the remote or behind it, it was decided that a solution would be to move the robot forward for a very short period of time. The robot could then determine re-determine the distance between itself and the remote to see whether it had moved closer or farther away from the remote. This would allow the robot to then change directions if it was moving in the wrong direction.
2) **Physical Appearance.** All we had to do to ensure our robot looked like how we designed it was to send the design into the machine shop. The water jet cutter gives precision cuts, allowing us to accurately design our robot. As for the controller, we simply kept the dimensions of the circuit board in mind as we were constructing it. This ensures that the circuit, along with all its buttons and such, would fit.

3) **Battery Limitations.** Certain cutbacks, such as not having strobe LEDS, had to be made because of the maximum voltage that the 9V battery provided. Although this limited what we could include in the actual robot, it was more important that it got the necessary power to function properly.

This project had several specifications that had to be met in order to be successful. Using MOSFETs to drive the motors of the robot was relatively easy to do as well as. A previous module that we had to do involved getting 1 servo motor running with P and N MOSFETs. All we had to do was build an identical circuit for the other servo motor.
Having to construct the robot ourselves while being battery operated were not hard requirements for us. Putting some AA batteries in series and a 9V battery was all that was needed to power the robot. As for the construction, we chose not to use the chassis that was provided as we wanted our robot to be different. As a result, we came up with a circular design.

However, some requirements were harder to meet. The main problem that our group encountered was with the transmitter as the minimum 20cm distance was difficult to achieve. The solution that we came up with was to use an H-bridge configuration for the transmitter. Instead of using a full H-bridge, we built a half H-bridge circuit for the transmitter. This resulted in our transmitting distance being relatively short.

Using the AT89LPC828 Microcontroller System and using C programming was difficult as well. If this was not the case, our group would have used a much easier prototyping platform such as Arduino which would have saved us a lot of time, allowing us to implement more features into our project.

3.5 Solution Evaluation

The slight movement forward by the robot allowed for it to easily determine whether it was moving farther away from the remote or closer towards it. Although it did not seem like an elegant solution at first, since the remote only needed to move forward for a very short period of time, the period required for the robot to determine the direction was short and did not impair movement.

For the actual robot, we had a couple of designs ranging from R2-D2 to Wall-E. In the end, we chose to go with a Roomba-looking robot. There is no requirement in this project for the robot to look a certain way so we chose this design as it was much simpler than Wall-E.

We had several designs for the electromagnetic field generator, or transmitter. Including designs utilizing op amps, comparators, transistors, and MOSFETs to amplify the signal generated by the microcontroller. As final design, we used a MOSFETs H bridge and series LC circuit with resonance frequency to amplify the signal generated from microcontroller. With this design, the fast speed of MOSFETs can provide the big peak to peak voltage for greater electromagnetic field.

3.6 Detailed Design

3.6.1 Transmitter

The transmitter circuit composed of an AT89LPC828 Microcontroller System and a series LC resonant tank circuit. This is ideal as for series LC circuits, at resonance the impedance of the circuit goes to zero. The opposite is true for parallel LC circuits, to which the receiver was based off of. The series LC circuit was run off a half H bridge MOSFET class-D driver system (Figure 1). The microcontroller was used to create a 50% duty cycle

square wave signal at varying frequencies to determine the amplitude of the transferred sine wave. Two buttons on the transmitter were used to change the frequency either higher or lower, which affected the robot's distance. Instead of BJTs, an LTV-847 opto-isolator IC was used instead. In order to avoid hardware dependencies, a 630V 0.43uF capacitor was used.

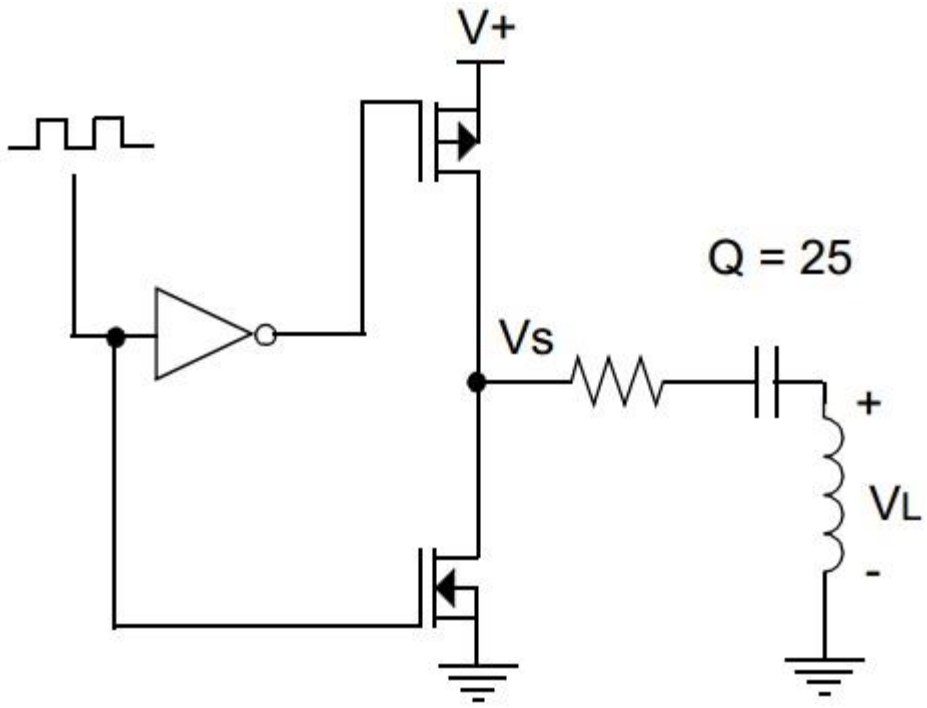

Figure 1: Half H-Bridge MOSFET Class-D Driver

3.6.2 Receiver

Each receiver circuit consisted of: A parallel LC circuit, a simple peak detector, a logarithmic and inverting amplifier (Figure 2). The result of this amplifier was fed into MCP3008 ADC to be measured and sent to microcontroller via SPI communication. We needed 2 of these circuits to triangulate the electromagnetic signal and find the direction

as well as distance. The logarithmic amplifier was used to bring the exponential function more linear, and have bigger readable range.

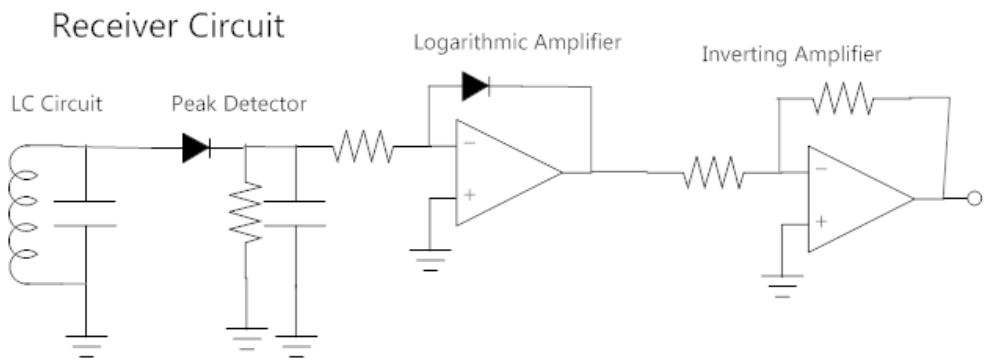

Figure 2: Logarithmic and Inverting Amplifier

3.6.3 Motors

In order to control each motors with variable speed and direction, we used the MOSFET H-bridge (Figure 3). Opto-isolators were used to separate the H-Bridge from microcontroller to protect the circuit from any interference. With this design, we only required 4 PWM signals from microcontroller to control 2 motors at fully adjustable speed and direction, or 2 PWM signals for each motor. The difference in two PWM signals would generate current flow in the motor at proportional rate.

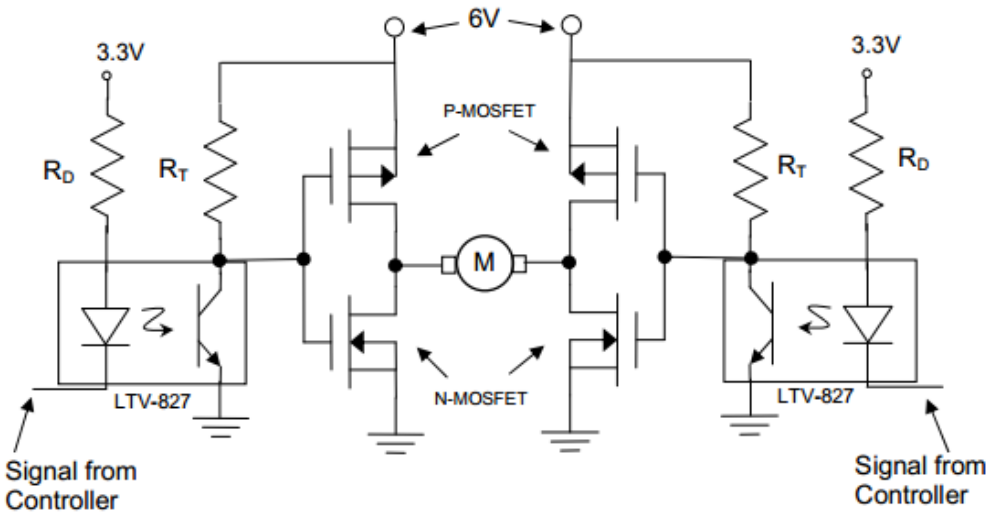

Figure 3: MOSFET H-Bridge

3.6.4 Software

We started with writing some functions that would help us write the main software, including functions to move the robot and functions to read and calculate distance. After writing these simple functions, our main loop looked very simple (Figure 4). First the robot would check to see it is pointing the right direction and turns if needed. After that, the robot would move forward, backward, or stay still depending on its distance from remote.

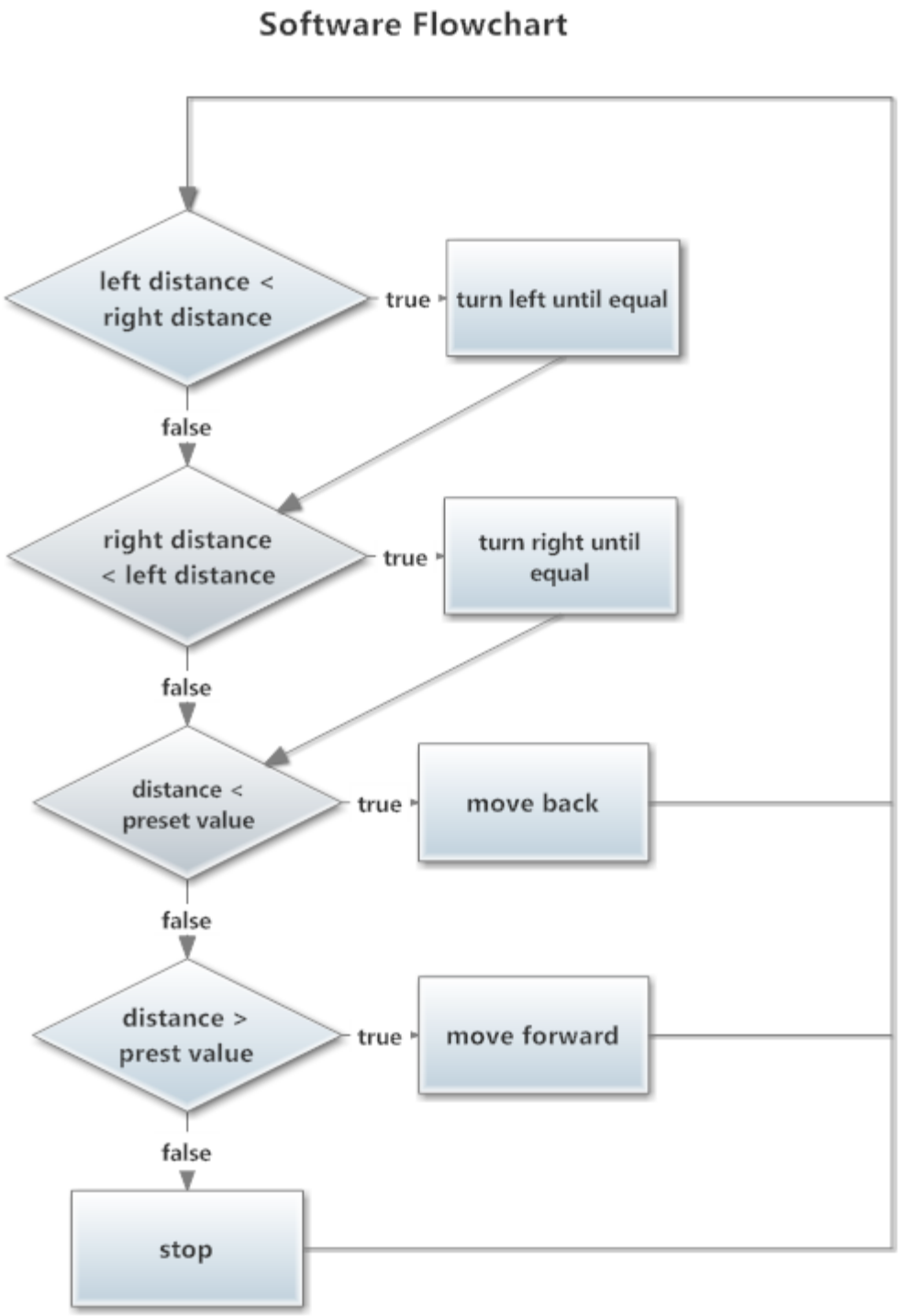

Figure 4: Software Flowchart

3.7 Solution Assessment

Below is a breakdown of our robot's strengths and weaknesses.

1) Strengths:
   - unique physical design
   - controller was easy to use
   - easily accessible on/off switches
2) Weaknesses:
   - range of the transmitter was short
   - constant movement stuttering

4 Life-long Learning

4.1 Software

With software, there is always something new to learn. With a vast variety of programming languages, the learning experience is endless. Through this project, we furthered our understanding of C in terms of interrupts and timers, where previously we had only used this concept in assembly. Other new aspects learned through this project with relation to software includes:

- Creating a PWM signal at a desired frequency, controllable through HyperTerminal or a switch
- Reading the values from an ADC to take samples in order to create a completely software based peak detector
- Building our own microcontroller system and flashing it

- Transferring the idea of a state-machine

While these may be tasks we have done similarly before (such as flashing the Intel 8052 microprocessor on our DE2), there are always key differences. The DE2 was a pseudo microcontroller, only an imitation, whereas we now have credible experience with the physical microcontroller itself.

Software proved to be a relatively simple task from previous experience in other courses as well as Project 1. The C programming language is heavily used in CPSC261 and CPSC259, so our group did not have any problems implementing code.

4.2 Hardware

Our group chose to make our robot from scratch, in order to achieve a unique appearance. All six member of the group received training in the following areas:

- Bending
- Drill Press
- Finishing
- Shearing
- Spot Welding

The two members in charge of hardware went on to learn the basics of 3D modelling software to aid the design process, namely AutoCAD Electrical. DraWinG (.dwg) files generated by the aforementioned software were also necessary in cutting material using the OMAX Waterjet Cutter available in the machine shop.

Aside from the standard machine shop training mentioned above (Bending, Drill Press, etc.), no further training is made available by the machine shop. Due to machine shop experience with the first project, the group was confident in working at the machine shop. There was a considerably increased sense of confidence which was lacking on the first project. We believe we were well prepared for this project, and effectively utilized prior knowledge and experience to our advantage.

5 Conclusions

The initial goals and specifications were accomplished, and the project performs as expected. As such, this project was an overall success. Furthermore, the aesthetics of the final product are quite admirable due in part to our unique design implementation - we did not use the default chassis as most groups chose to do. The team was able to effectively design and implement a hardware structure that effectively housed all the components, while displaying internal circuitry with the use of an acrylic glass top cover. Despite the challenges encountered in designing, building, and testing the project, our team successfully followed through and met initial goals within time and scope.

6 References

Calvino-Fraga, Jesus. *Project 2: Electromagnetic Tether Robot.* (Vancouver: University of British Columbia, 2013).

7 Bibliography

Arduino. *Servo Library*. Retrieved March 27, 2013, from http://arduino.cc/en/Tutorial/Knob.

Calvino-Fraga, Jesus. *Debugging Assembly Programs with Cmon51 in DE1-8052 and DE2-8052 Processors.* (Vancouver: University of British Columbia, 2013).

Calvino-Fraga, Jesus. *Module 4 – Embedded C Programming.* (Vancouver: University of British Columbia, 2013).

Calvino-Fraga, Jesus. *Module 5 – Magnitude and Phase Measurement.* (Vancouver: University of British Columbia, 2013).

Calvino-Fraga, Jesus. *Module 6 – Interfacing Using Transistors.* (Vancouver: University of British Columbia, 2013).

Calvino-Fraga, Jesus. *Project 2: Electromagnetic Tether Robot.* (Vancouver: University of British Columbia, 2013).

Microchip Technology Incorporated. *AN232: Low-Frequency Magnetic Transmitter Design.* Retrieved March 15, 2013, from http://ww1.microchip.com/downloads/en/AppNotes/00232B.pdf.

MIT FabLab. *A Gentle Introduction to Abrasive (Waterjet) Machining.* Retrieved March 23, 2013, from http://rpl.mit.edu/tools/tutorials/Waterjet-OMax-1/A_Gentle_Introduction_to_Abrasive_Machining_-_Part_I.pdf.

8 Appendices

8.1 Appendix A: Robot Movement Sample Code

The following code controls the movement of the robot and uses simple functions such as the distance and turning functions.

```
void move(void){

	int chosenDist = 600;

	while(1){
		if ((distance(0) > 0) && (distance(1) > 0))
		{
			if (distance(0)+250 < (distance(1))){
				turn(0);
				while (distance(0) < distance(1)){}
				stop();
			}
			if (distance(1)+250 < (distance(0))){
				turn(1);
				while (distance(1) < distance(0)){}
				stop();
			}
			if (distance(0)+200 < chosenDist){
				moveStraight(0);
				direction = 0;
				delay(1);
			}
			else if (distance(0) > chosenDist + 200){
				moveStraight(1);
				direction = 1;
				delay(1);
			}
			else
				stop();
		else{
			direction = 0;
			stop();
		}
	}
}
```