ELEC 291

Electrical Engineering Design Studio I

Section 201 L2A

Dr. Jesus Calvino-Fraga, P.Eng.

Project 2 - Coin Picking Robot

Date of submission: April 7, 2022

Table of Contents

1. Introduction

1.1 - Design Objective and Specifications

The purpose of this project is to design, build and program an autonomous coin picking robot. This robot is expected to move within a predefined area to detect coins with a metal detector constructed from inductors, pick them up with the use of servo motors, and identify the perimeter of its area defined by a wire carrying AC current.

1.2 - Project Requirements

The following requirements are taken from the lab manual [1]:

- ➔ *A microcontroller system that is not based on the 8051 microcontrollers:* Parts to assemble microcontroller systems based on some popular microcontrollers are provided with the Project 2 Kit.
- ➔ *Battery operated:* The robot must be battery operated. Both an AA battery holder and a 9V battery clip are included in the project kit.
- ➔ *Metal detection:* Coins must be detected using a detector that is assembled using the inductors provided in the Project 2 Kit. The metal detector must be able to detect all the Canadian coins in circulation (0.05$, 0.1$, 0.25$, 1$ and 2$).
- ➔ *Coin picker mechanism and electromagnet:* A coin picker mechanism consisting of your two servo motors and an electromagnet is provided with the Project 2 Kit. The coin picker must be capable of picking any of the current Canadian coins in circulation (0.05$, 0.1$, 0.25$, 1$ and 2$ provided they can be attracted by a magnet) and carefully deposit

them in a container carried by the robot.

➔ *Robot construction:* You can use any material you find available for the chassis of the robot (paper, cardboard, wood, plastic, metal, etc.). You can also use the materials and tools available in the MCLD workshop. If you are using the workshop you must complete the workshop safety training first. Also, you are required to wear safety glasses and shoes while in the workshop.

1.3 - Hardware and Software Designs

Below are graphical representations of both our hardware and software designs in block diagram format. For the hardware section, boxes represent components of our circuit, and arrows represent signals being passed in between them. For the software diagram, rectangles represent steps in the process of our code, diamonds represent decisions the program makes, and lines represent which step comes next.

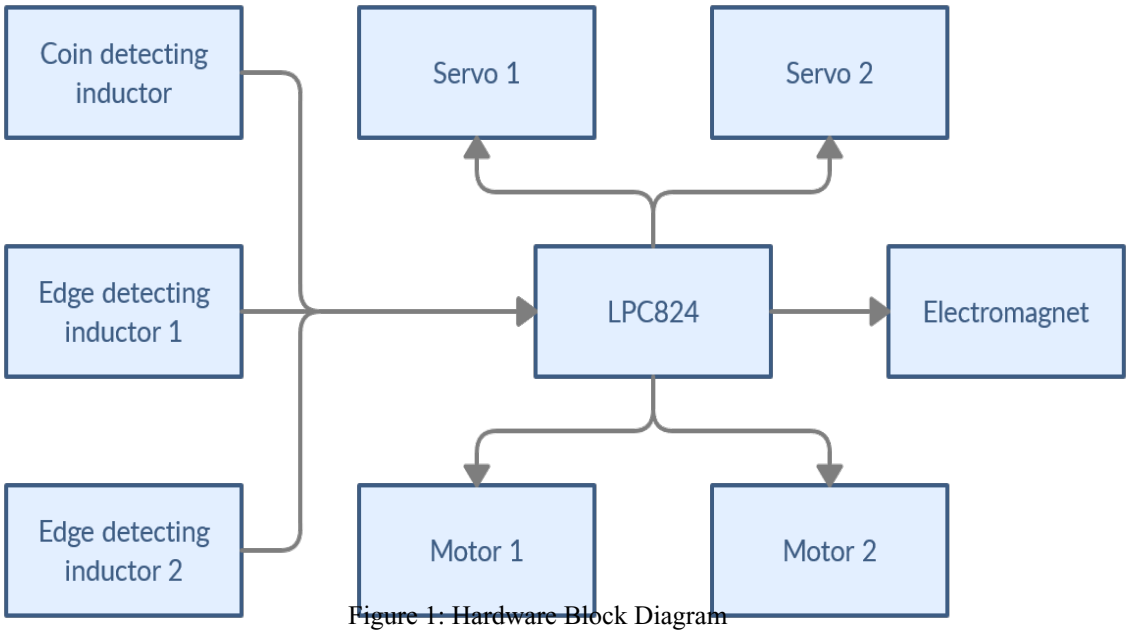


Figure 1: Hardware Block Diagram

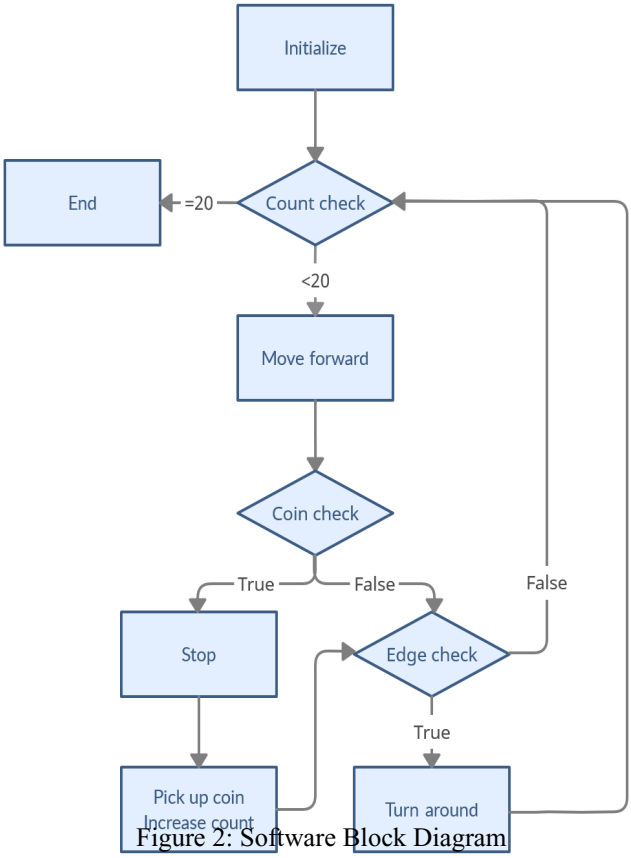


Figure 2: Software Block Diagram

2. Investigation

2.1 - Idea Generation

Our team first discussed which microcontroller we would use for this project based on our experiences from lab 6. We chose the LPC824 because it is compact, which means we would be able to save space on the breadboard. It is also powerful with sufficient pins for the scope of our project. Furthermore, the LPC824 uses a similar coding structure to the 8051, so not much extra learning is required to use the microcontroller sufficiently. Furthermore, it has an ARMv7

architecture, which can be very useful as many ARM architecture microcontrollers are used in the industry. After we decided on the microcontroller, we split up the functions of the robot and assigned it to group members. These roles were constructing the robot base, detecting the coins, detecting the perimeter, moving the servos to pick up the coins, and using the servos to move the robot. We figured that it would be difficult to fit all of the components onto one breadboard so we chose to use two instead. We then brainstormed how we would arrange the breadboards for a robust design. One hypothesis was that the voltage would increase when the AC current-carrying wire was brought close to the inductor. This hypothesis was proven to be correct after testing the robot we built.

2.2 - Investigation Design

The best guides for us during this project were the lecture slides [2] and sample code that was given to us. As a group, we constructed the main body and coin picker using the provided instructions and made the circuit by checking the datasheets for each circuit component. Our first step was comparing the main body and coin picker we made to the photos on the slides [2]. Next, we used the sample code that was given to us to check if our circuits were working. After ensuring they were, we realized that similar things were done in the previous labs that we could use in this project. We first made sure to understand them correctly and used the code as a reference in addition to the resources given by the professor.

2.3 - Data Collection

Our team spent a significant amount of time deciding what data we needed and which components they would be related to. We took advantage of the equipment provided in the ECE

labs, including oscilloscopes, power supplies, and function generators. One of the most important pieces of data we collected was obtaining threshold values of frequency for the metal detector. We recorded the values of frequency and count for each coin to ensure the metal detection system was accurate for all coin sizes.

	No Coin	**Dime**	**Nickel**	**Quarter**	**Loonie**	**Toonie**
Frequency (Hz)	00019706	00019849	00020001	00020106	00020011	00020011
Count	106538	105782	105142	104835	104435	104555

Figure 3: Metal Detector Frequency and Counts

2.4 - Data Synthesis

After enough data was collected, we verified that all of the constituent parts of the robot worked by themselves. We tested every single sample code given to us and figured out what they were needed for. First, we made sure the main body, coin picker, and electromagnet were built correctly and that we could detect the coins. Then, we built the microcontroller system and optocoupler using the lecture slides [2] and the microcontroller document [3] and tested to see if we were getting the correct voltages at different parts of the circuit. After, we constructed the servo arm and perimeter detection system. The only thing left was connecting all of the parts together. From measuring the frequency and count of each coin, we were able to conclude that a coin was near the robot when the count was above 106000. These findings allowed us to

successfully construct the code for our metal detector.

2.5 - Analysis of Results

We determined a threshold value for our metal detector. We tested it by displaying a success message on PuTTy, to see if our software could determine whether or not there was a coin underneath. We then progressed into connecting it with the electromagnet to determine if it would generate a current to pick up the coin. At first, when this process did not work, we debugged it by creating another message on PuTTy to determine if the voltage should be on or off.

3. Design

3.1 - Use of Process

The "Engineering Design Process" was heavily incorporated in the making of this project. As the first step, we determined the requirements in order to complete the project and brainstormed ideas to satisfy all these conditions. Everyone was assigned a different component of the project and we kept trying new ideas and iterating them if needed. When adding in new components, we utilized a regression testing method in which at each iteration we tested our design so that if the robot stopped working, we could easily identify at which stage or which component was causing the issue.

3.2 - Need and Constraint Identification

The lecture slides [2] and the lab document [1] were very helpful in the process of determining our requirements for the project. After carefully examining these documents, we made a list:

- ➔ Electromagnet must be able to detect all the coins that are magnetic.
- ➔ The servo arm should be able to pick up the coin and drop it into the coin bucket.
- ➔ Our robot should be able to work completely autonomously.
- ➔ Everything must be battery operated and coded in C.
- ➔ The robot must be able to detect the wire that marks the perimeter.
- ➔ The microcontroller must not be 8051.
- ➔ The voltage of the inductor should increase when the perimeter is detected.
- ➔ The wheels have to turn forward and backward (to turn).

3.3 - Problem Specification

The first additional function we thought about adding were small mechanical components and changing the code so that when the robot picks up all the coins, it can dump them out of the bucket by itself. To do this, we considered using a spring and a paperclip as a hinge. We figured that we could loosen the screw underneath the coin box and add a spring under one side that would tip the box when we triggered the hinge. Another piece of additional functionality we considered was connecting a buzzer to the robot and having it make a sound in morse code according to the amount of money it picked up. Our third idea was making the robot dance by moving the servo arm and wheels in a fun way. Unfortunately, we did not have time to implement any of these ideas.

3.4 - Solution Generation

After trying to fit every circuit component onto a single breadboard we decided that we would need two to comfortably fit everything. We had a few different ideas on how to balance

the two boards on the main body. These included: fitting cardboard inside the robot to balance both boards by sticking them sideways, sticking the boards to each other and balancing them on the main body, and having one board stay on the main body straight then sticking the second one on the side. We also tried different materials in the process like tape, blue tags, and the adhesive backing on the boards. In the end, what worked best was tape and arranging the breadboards in a triangular tent-like shape.

3.5 - Solution Evaluation

One of the problems we faced was the limited mechanical range of the servo arm. Since the gear of the servos can only rotate 180 degrees, we had to ensure the horns were installed such that they pointed in desirable directions over that range. It took much experimenting and many reinstallations to get them in an orientation where they could allow the arm to both pick up and drop the coins.

We noticed that the Vcc of the LPC824 dropped unexpectedly at times. In hindsight, we realized that this might have been caused by the PWM signals to the arm servos not being connected to the optocoupler as they were supposed to or connected to diodes. Servos produce noise through back emf, which if left unfiltered could have interfered with the microcontroller. A difficulty that we faced with the motors was that the signal being outputted from the microcontroller was too low to trigger the optocoupler. This would result in either one or both motors not functioning since they were not receiving a signal. After assessing the problem, and attempting multiple fixes, we realized that we had connected the microcontroller side of the optocoupler to the 6V battery, instead of the 3.3V source. Once we rewired the positive power rail to the 3.3V source instead of the 6V battery, the problem was resolved.

One of the most prominent issues we faced with coin detection was that the frequency threshold for detecting a coin was constantly changing. After some experimentation, we realized that the threshold was changing when we activated different parts of our circuit, specifically the motors and the electromagnet. This was due to the shared voltage source. The inductor would get less power the more robot components were operational. Eventually, once all circuit components were operational, we were able to obtain and use a reliable frequency threshold. We only needed to adjust this threshold a few times as the battery became weaker over time.

3.6 - Safety and Professionalism

During all of our work sessions, we followed every relevant COVID and electronics lab safety procedure. When meeting in person indoors, we made sure to wear masks and follow social distancing protocols. We also verified that anyone wanting to use equipment in the lab was aware of how to do so safely and properly. For example, anyone using the soldering iron made sure to wear safety glasses, turn on the fume extractor, and wear proper clothing (closed-toe s hoes, long sleeves). Group projects provide a unique opportunity to exchange ideas and to learn from the skillsets of others. For each group member to learn and contribute as much as possible, we fostered an inclusive work environment where everyone was encouraged to share their ideas. Work was divided equally and democratically, ensuring that each group member was put to work doing something they were interested in and/or skilled in.

3.7 - Detailed Design

Hardware

There were two types of batteries used to power the robot - four AA batteries and a single 9V battery - providing a 6V and a 9V voltage to the circuit. The batteries were first connected to a switch which allowed us to avoid wasting energy when it was not needed. The 6V batteries were used to power components such as the electromagnet, optocoupler, four servo motors, and two H-Bridges. The 9V battery was first connected to a high voltage regulator (LM7805), then using a diode (MBR150) alongside the USB connector with another diode (MBR150) producing a steady 5V powering the opamp (KA358) and then to a 3.3V voltage regulator (MCP1700) powering the microcontroller (LPC824). Many of the components used such as the LM7805 and MCP1700 needed filtering and decoupling capacitors of 0.1µF and 0.01µF at both the inputs and outputs of the components.

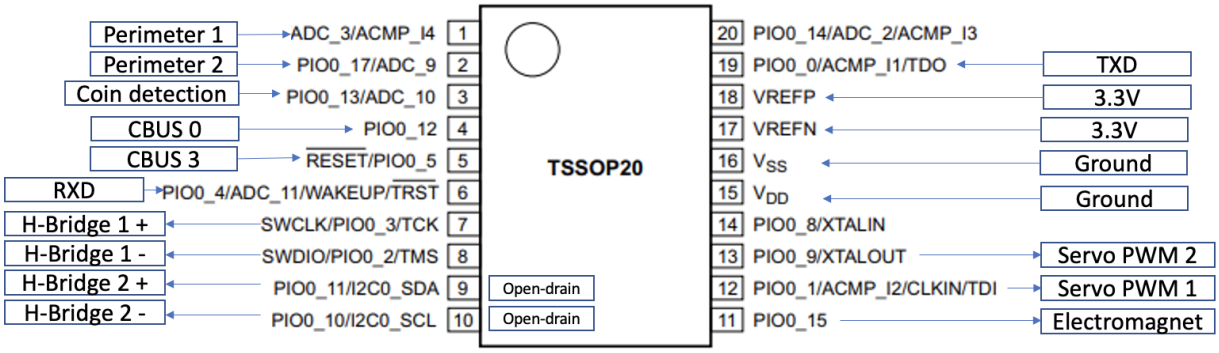
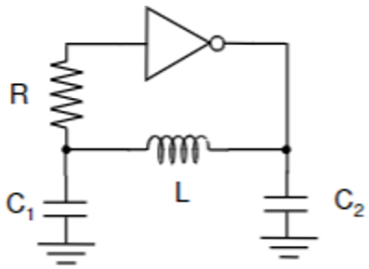


Figure 4: Microcontroller Pinout



The perimeter detector used an inductor in parallel with a capacitor resulting in an oscillator frequency of (). The CMOS inverter was made with PFETs and NFETs.

Figure 5: Metal Detector

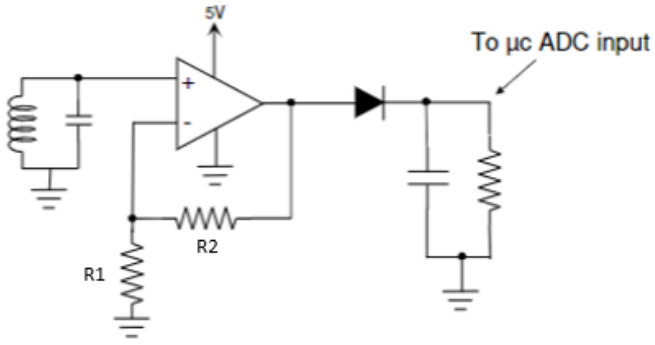


Figure 6: Perimeter Detector

The input to the opamp was built using an inductor in parallel with a capacitor. The resonant frequency between the two components is . With our values, the resonant frequency was around 16kHz. The gain of the opamp was $A_M = 1 + \frac{R2}{R1} = 1 + \frac{4.7k\Omega}{100\Omega} = 48\frac{V}{V}$. The RC circuit feeding into the microcontroller had a time constant of 10ms. The perimeter was made by forming wire into a rectangle of at least 0.5 m^2. A function generator produced a sinusoidal AC current through the wire of amplitude 20V peak to peak and frequency 16kHz.

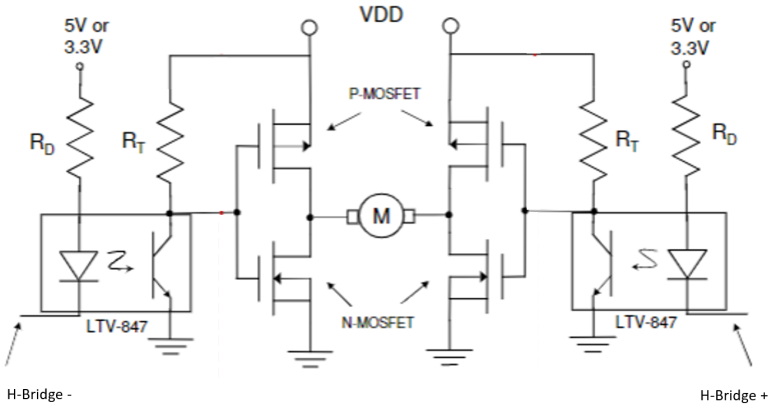


Figure 7: H-bridge with Optocoupler

The optocoupler is used to isolate the noise and high voltage of the motor from the microcontroller. When one H-Bridge was 5V and the other was 0V, then the motor would turn clockwise and vice versa.

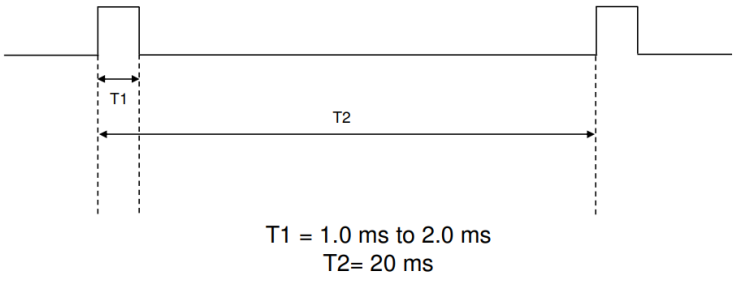


Figure 8: PWN Arm Servo Control Signal

Two servo motors used to move the coin picking arm was set to certain angles based on the separate pulse-width modulation (PWM) control signals that are delivered to them. The PWM signals had a 50Hz (20ms period) pulse frequency with pulse widths ranging from 1.0ms to 2.0ms. Varying the pulse widths varied the angle that the servos rotated to. The

servos had a range of 180 degrees, and the PWM signals had a resolution of 10μs. The upper and lower arm servos both received 6V, and were connected to pins 13 and 12 of the microcontroller respectively.

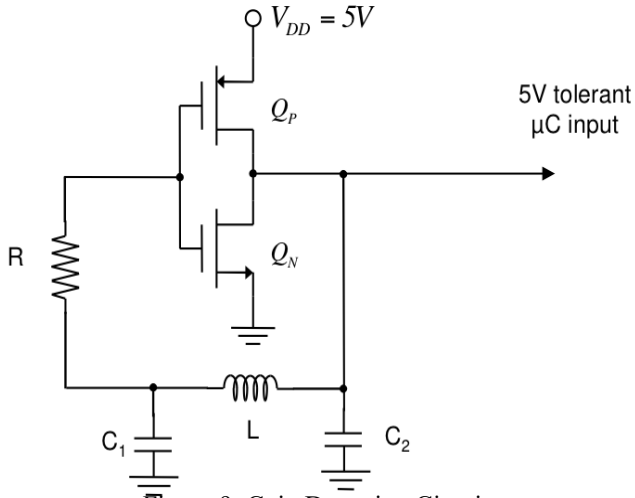


Figure 9: Coin Detection Circuit

At the end of the servo motor was an electromagnet made from an AWG 28 magnet wire roll, a steel machined screw, washers, a nut, and masking tape. The wire was wound around the screw with washers and a nut on the ends until the total resistance across the screw was 6 to 8 ohms. The ends of the wire were then twisted together and the coil of wire was wrapped in masking tape. When voltage was applied to the wire, current flowed through the wire and created a magnetic field that attracted the coin and picked it up. A metal detection circuit made of a Colpitts oscillator with a discrete CMOS inverter helped detect when a coin was near the robot. When a coin was placed close to the inductor in the circuit, the magnetic field and inductance would change, and the oscillator frequency went up. The microcontroller determined if the oscillator frequency had crossed a threshold value and applied a voltage to the electromagnet to pick up the coin. Furthermore, a back EMF diode was used between the electromagnet and the microcontroller ensuring noise from the electromagnet did not interfere with the microcontroller.

The wire was wound around the screw with washers and a nut on the ends until the total resistance across the screw was 6 to 8 ohms. The ends of the wire were then twisted together and

the coil of wire was wrapped in masking tape. When voltage was applied to the wire, current flowed through the wire and created a magnetic field that attracted the coin and picked it up.

Software

The *main.c* code first started with variable definitions, initializing timers, creating functions that control the movement of the wheel motor, servo motor initial positions and controlling functions, reading period for the metal detector, and calibrating the ADC pins to become an analogue input instead of the digital I/O. The main while loop continued until 20 coins had been collected and the counter *coins* were incremented each time a coin was successfully placed into the box. Continuously within the loop, the two ADC channels were read and a period reading was taken to determine if a coin was detected or not. Within an if statement, the pick-up coin sequence was triggered once the period was below a threshold value (36900 seconds). Within that if statement, the electromagnet was controlled using *GPIO_B15* to pick up the detected coin. In between the movement functions, *waitms()* was added to ensure the coin did not fall off during the movement sequence.

The coin detection if statement was then followed by an additional if statement which was responsible for perimeter detection. The statement checks both ADCs for a voltage spike. If the voltage of either of the ADCs surpassed the threshold value of 1.2 V, the turn around protocol was activated which consisted of going backward for half a second and then turning clockwise for half a second. Once this if statement was executed, the code restarted with checking the coin count, and then moving forward.

The main functions utilized in the loop were the *stop(), forward(), backward(), clockwise(),* and *counter_clockwise()* which were used to control the wheels of the robot and would continue continuously unless otherwise stopped or a different movement function was called. In addition, the *move_arm* function used hardcoded values that correlated to the servo arm picking up and dropping coins into the box depending on the index value specified from 0-2.

3.8 - Solution Assessment

In order to evaluate if our robot was meeting our design criteria, we developed several methods of assessment so that we could improve upon our design easily. The main components we focussed on during testing were the effectiveness of the perimeter detector, metal detector, and arm servo motors. For the perimeter detector, we used the oscilloscope to determine if the ADC was reading the current-carrying wire. In addition, when our perimeter detector was faulty, we used PuTTy to graph the voltage values so that the threshold value could be adjusted accordingly. Next, to test the metal detector, we used PuTTy to print out strings on the terminal which read "Coin Detected" or "No Coin Detected" to ensure the hardware and software were communicating properly. Similar to the perimeter detector, we used frequency values on PuTTy to adjust the threshold values if the detector failed to detect a coin.

The main method we used for assessing the servos in the arm was criteria-based. The arm needed to reliably pick up the coin that had been detected. Additionally, the arm needed to move slow enough so that the coin would not detach prematurely. Lastly, the arm needed to be correctly positioned over the box so that the coin would fall into the receptacle, no matter its position on the electromagnet. Once the arm met all three of these criteria, we were satisfied with its performance.

The microcontroller’ s validity can be tested by whether or not the makefile can be successfully built and flashed onto the microcontroller through the USART port connected to the microcontrollers through a series of serial ports. The microcontroller is always first tested to see if all of the digital GPIO ports are functional, as it can only produce a voltage that is equal to the Vcc of the microcontroller (3.3V) when on, and near a 0V value when off. If the GPIO output is not producing the correct voltage of 3.3V when on, then the output voltage of the voltage regulator should be checked and ensure that it produces a steady 3.3V fed into the microcontroller.

The metal detector was not functioning at first. Through measuring the output of the inductor directly it was found that the frequency was oscillating with the perimeter detector and thus the capacitance that is paralleling with the inductor would have to be adjusted until the resonance between the perimeter and the metal detector no longer existed. Without additional modifications, the perimeter wire and the metal detector were not very robust and often caused short-circuiting or poor conducting when they otherwise might have seemed to be properly installed into the breadboard. To increase the robustness of the wire, we soldered wires to the tip of the ends which made the connection much more stable. The functionality of the metal detector was first tested to ensure all parts were functioning by directly connecting the ends of the inductor and capacitor parallel to the oscilloscope and denoting the change of frequency directly.

The perimeter detection was also tested using a similar method where the individual inductors nodes were fed into the oscilloscope observing the voltage change when the perimeter was near. However, it was also observed that the output from the diode after the output of the opamp would be pulled-up whenever the ADC channel was connected. In order to lower that voltage to a voltage that could properly differentiate between a perimeter or not, a pull-down

resistor of 1kΩ was used. After the voltage was lowered to a voltage where the presence of the perimeter could be clearly seen, the threshold voltage was then logged into the microcontroller and could then reconfigure the movement of the robot when a perimeter was reached.

The four outputs from the microcontroller to the pair of H-bridges also faced some difficulties. At first, we did not realize that pin 9 and 10 were open-drain GPIOs and were pondering upon why the voltage output of those two pins were unreasonably low. After connecting it to the LED of the optocoupler, the resistance inside the LED was enough to act as a pull-up resistor and was rigidly functioning. However, it wasn't so simple with the other two GPIO outputs, where they would at most produce only 3.3V to the LED of the optocoupler. This was not enough to enable the output of the optocoupler and caused one set of the H-Bridge to not receive proper signals. Therefore, another set of pull-up resistors of 25Ω were used to bring the voltage up to trigger the optocoupler.

In general, the individual components were first tested and validated before being fully integrated into the circuit. Many of the components were first tested using voltage supplied from the power supply to ensure that ideal conditions (Vcc, potential difference) were met for the components and that the signal from the microcontroller was correct and activated the components. However, the more we combined constituent parts, the more issues arose. Many times, the voltage supplied from the 6V and 9V batteries would dip whenever multiple components (electromagnet, servo motor, H-bridge) were active. They were draining the Vcc so that small components such as the opamp, optocoupler, and FETs would not be able to receive a sufficient voltage to function. During the whole project, two sets of four AA batteries and two 9V batteries were replaced due to the insufficient voltages that it supplied after extensive usage. Although the robot could function, it was not a completely robust design as even minor changes

to the battery voltage could make the robot malfunction. Furthermore, the threshold measured from the perimeter and metal detector would have to be calibrated every once in a while. There were definitely parts of the solution that were quite robust For example, we almost never had issues with the microcontroller I/Os and their voltages. Also, the microcontroller code was quite robust as well.

4. Life-Long Learning

During this project, our team mainly applied skills learned from previous courses such as APSC 101, APSC 160, CPEN 211, CPSC 259, and ELEC 211. Similar to the autonomous claw project from APSC 101, the Coin Picking Robot project used servos to autonomously respond to the environment. Throughout the software portion, many of the debugging methods and syntax were pulled from our programming courses such as APSC 160, CPEN 211 and CPSC 259. Lastly, from ELEC 211 we had a general baseline on electromagnetics and how the metal detector/arm utilized physics concepts to pick up coins. One of the most valuable skills we learned during this project was how to build the motor used to control the robot. However, we would have found it helpful to have taken a course about motors beforehand. This way it would have been easier for us to figure out how to build and utilize the motor faster.

5. Conclusions

In the end, we chose a versatile design that detected coins, picked them up, collected them in the coin box, detected its perimeter and moved autonomously. The main problem we encountered was ensuring a compact wiring design since only two breadboards could

comfortably fit on the robot body. Additionally, faulty wiring proved to be problematic and the vast majority of our lab time was spent debugging the circuits. We were able to solve these problems by receiving guidance from the professor and using the lab equipment. In total, we spent approximately 30 hours in the lab working on the hardware and software of the robot and approximately 10 hours writing the lab report.

6. References

[1] Calvino-Fraga, Jesus, "ELEC 291 Project 2 – Coin Picking Robot", 2022

[2] Calvino-Fraga, Jesus, "ELEC 291 Project 2", 2022

[3] Calvino-Fraga, Jesus, "The LPC824 Microcontroller System", 2022

7. Bibliography

- *Datasheet for LTV-846 Lite-on optocouplers*. Octopart. (n.d.). Retrieved April 4, 2022, from https://octopart.com/datasheet/ltv-846-lite-on-704419
- *GM4 plastic geared motor - cdn.solarbotics.com*. (n.d.). Retrieved April 4, 2022, from https://cdn.solarbotics.com/wp-content/uploads/motor_data_summary-gm4.pdf
- *LPC82X data sheet - NXP*. (n.d.). Retrieved April 4, 2022, from https://www.nxp.com/docs/en/data-sheet/LPC82X.pdf
- *MA7800 series positive-voltage ... - SparkFun Electronics*. (n.d.). Retrieved April 5, 2022, from https://www.sparkfun.com/datasheets/Components/LM7805.pdf

➔ NTD2955, NVD2955 MOSFET - On semiconductor. (n.d.). Retrieved April 4, 2022, from https://www.onsemi.com/pdf/datasheet/ntd2955-d.pdf

➔ *NTD3055L104, NTDV3055L104 MOSFET - On semiconductor*. (n.d.). Retrieved April 4, 2022, from https://www.onsemi.com/pdf/datasheet/ntd3055l104-d.pdf