Project 2 - Coin Picking Robot

Group No X

Student No.	Student Name	Points Allocated	Signature
39875828	Colin Pereira	100	Colin Pereira
18987792	Mohamed Salah	100	Mohamed Salah
20333308	Umair Mazhar	100	Umair Mazhar
96925458	Der-Chien Chang	100	Der-Chien Chang
34532507	Will Chen	100	Will Chen
11056892	Nick Vo	100	Nick Vo

University of British Columbia
Electrical and Computer Engineering
ELEC 291 Winter 2022
Instructor: Dr. Jesus Calvino-Fraga
Section 201

Date of Submission: 08-April-2022

Table of Contents

1. Introduction

1.1. Objective

The objective of this project was to design and build an autonomous robot to detect and collect coins while being bounded by a guidewire perimeter.

1.2. Specifications

Hardware Specifications

- Refer to Appendix for detailed part list

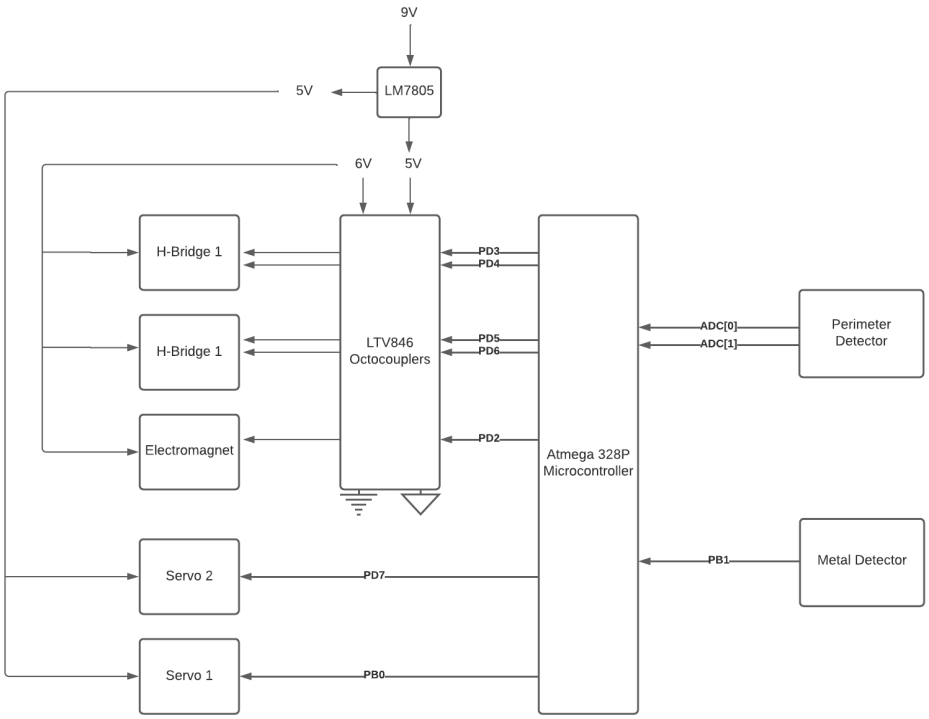

Figure 1: General block diagram of coin picking robot

Software Specifications

- All code was written using the C language
- The microprocessor supports upto 4 bits of binary information
- Supports the following commands:
    - Stop Robot
    - Turn Right, Left, Back
    - Make a U-Turn
    - Detect perimeter

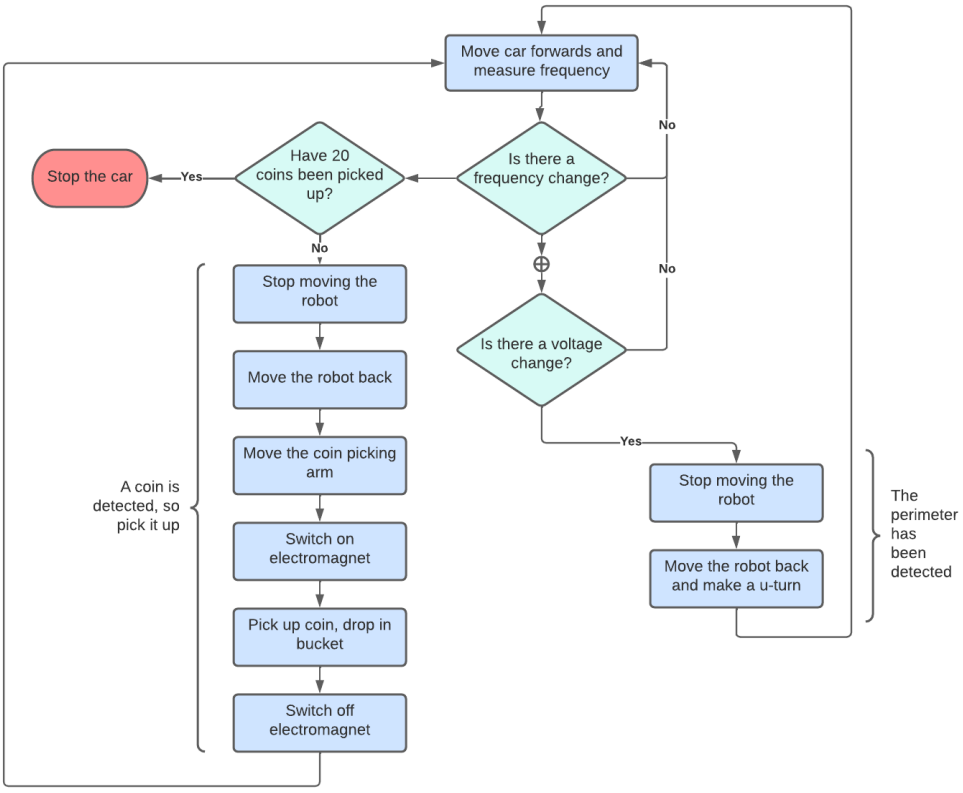

Figure 2: Robot logic

2. Investigation

2.1. Idea Generation

The idea generation process was followed by understanding the lecture slides. By fully understanding the requirements of the project, we are able to list out the potential difficulties we might face. At the early stage of the designing process, we predict that we might have a hard time assembling the hardware itself and also predict that it is hard to make the metal and perimeter detector as sensitive as it should be. In order to make the entire designing process work well, we make the hardware and software block diagram beforehand and generate different ideas on how to build the hardware within the smallest area possible it could.

2.2. Investigation Design

We investigate the design through a reading of the lab requirement and the lecture slides. We looked into each component and its manufacturer datasheet to find out how to incorporate them into the robots.

After we understand fully how each component of the coin-picking robot works, we then start to design the location of each component on the breadboard and the robot itself. We first determined which pin the power should go to, then where the inductor sensors are. And then build the circuit around the few requirements that we had to make sure the robot would function perfectly.

We also, in the process of investigation, realized that each component would create noise that may impact other components. Hence, we decided to keep the easily influenced components, such as a metal detector and the electromagnet, far from each other to minimize noise.

2.3. Data Collection

Most of the data collection occurs at the stage of debugging and understanding the sensitivity of the metal and perimeter detector. We experienced significant fluctuations in the frequency with the coin detector, which led to the coin detector being triggered when it's not supposed to be. The tools we used for the data collection are elaborated below.

PuTTY

We had PuTTY running when the robot was running, which enabled us to read the voltage and frequency collected from the robot. This also made it easier to adjust the threshold value to make both the metal detector and perimeter detector more sensitive.

Multimeter

We use a multimeter to measure voltages of various components of our circuits to make sure they have the correct voltage and impedance values that are aligned with our design.

Function Generator

We used the function generator to generate a constant 16kHz signal through the guidewire to test our robot's ability to detect the changing magnetic field as well as its behaviors after the detection.

Power Supply

We use an available power supply in the lab to generate 6V power to our robot motor in order to generate a reliable power source. At the same time, it also saves battery life as well as cost of battery replacement.

To detect frequency and voltage changes, we recorded test data on a table, as outlined in the appendix, to understand how the frequency and voltage change when our robot encounters a coin or the perimeter guidewire.

2.4. Data Synthesis

We mainly used the crossIDE function, multimeters, and the action of the robot itself to collect the data needed for this project.

The most useful tool that we have in this stage of the project is the ability to read data using putty. By using putty, we could gather the frequency reading of the metal detector as well as the parameter detector values.

Another data collection technique that we used is the multimeter. By using the multimeter we successfully found out the major mistakes that we had made during the circuit-making process and the healthiness of the batteries.

Lastly, we used the actual motion of the robots to determine whether or not our design actually works.

2.5. Analysis of Results

We analyze the data collected (as shown in 2.4) in many different ways. Yet the most effective one is by comparing the data collected and the movement of the car.

We use the data collected by the ATmega microcontroller and show the collected data on the putty simulation. In this way, we found out our threshold value. This way, we could easily gather the threshold value that we need for the robot to detect the coin and detect the parameters.

The other way we analyze the data is by observing the movement of robots. We observe the moment of the arm and see if the metal detector is actually able to detect the coin. And we observe the wheels to see if our code actually makes the robot move forward, backward, left, and right. This way we can find mistakes in our code more easily.

3. Design

3.1. Use of Process

Design requirements were based on the lab manual and lecture slides. The robot design requirements were:

- A microcontroller system that is not based in the 8051 microcontroller (we chose the ATMega328P by Atmel/Microchip)
- The robot has to be powered by batteries.
- Use the inductors to act as metal detectors to detect coins.
- Coin picking mechanism with 2 servos and an electromagnet.
- Must be able to pick all available Canadian coins and stop after 20 successes.

During the entire design process, our team revised software through a private repository on Github. This platform became extremely useful to us as we backtracked through code snippets that introduced bugs and allowed us to work on separate blocks of code before we merged them.

We drew out the block diagram of the hardware on pen and paper to understand the high level hardware design before connecting all the components and a simple flowchart was created on pen and paper to understand the flow of the program that we would have to create.

3.2. Need and Constraint Identification

To understand the needs, our team appreciated the fact that the stakeholders for this setup would be garbage processing companies to sort out metal from wastes. With this in mind, we understood that not only the cost efficiency needed to be a priority, but also the sensitivity of detecting the object has to be precise and efficient.

The constraints for this project was the interference between the wires and the inductors such that there's noise messing with the frequency measurements to impact our robot accuracy in properly detecting the coins. Our robot can also backing up over the guidewire when there's is a coin at the corner of the parameter because after the robot turns around and picks up the coin, it detects the guidewire and proceeds to back up further. The reason for this is because our inductors can only detect if there's a change in the magnitude of the guidewire magnetic field, but not

the direction of it, therefore causing our robot to back up over the guidewire and leave the preset parameter.

3.3. Problem Specification

Space management was a priority as there were only so many breadboards we could use.

3.4. Solution Generation

- To reduce signal noise from other components, the wires were covered with insulating tape and the electromagnet and motors are connected with an optocoupler.

- In addition, we use PuTTY to record the input frequencies from the metal detection and parameter detection system while it runs so we could check for the threshold frequency/voltage if the robot works as intended.

- After several attempts, we discovered the arm was moving too fast, so the arm would fling the coin off and cause it to fall off due to the extra force. Therefore, we not only slowed down the movement of the arm to prevent the coin from swinging away from the electromagnet, we also added code to sweep the arm from right to left when the arm is lowered to make sure the coin is actually picked up by the electromagnet.

- We discovered that during the demonstration, the threads on one of the 3D printed wheels was damaged and caused the wheel to loosen over time, which interfered with the robot's path and caused it to sway a little bit towards the left. To remedy this, before each demonstration, we would screw the wheel in as tight as possible.

3.5. Solution Evaluation

- Using PuTTY to record the frequencies greatly increased the easability of recording our data and finding the correct frequency.
- This greatly increased the smoothness of our arm and the coins were no longer being flung and were sticking to the electromagnet.

Covering loose/open wires with insulating tape had a small impact in reducing the signal noise. The optocoupler was somewhat effective in reducing the signal noise for the coin detector. Initially there were huge fluctuations, ~55700-58100, however after we added the optocoupler

circuit, the fluctuations decreased to around ~55700-55900. While this improved the effectiveness of our coin detector, the signal was still too noisy and our robot was still unable to effectively pick up dimes and nickels every time. Therefore, we had to increase the threshold in order to prevent false positives. In the end, our robot was only able to pick up quarters, loonies and toonies.

3.6. Detailed Design

Wheels, motor and H-Bridge

In our design, we connect the ATmega microcontroller to the H-bridge. Which is then connected to the two terminals of the wheel motor.

The H-bridge is composed of a basic CMOS circuit, and components of the optocoupler. The diode side of the optocoupler, would have one side connected to the 5V power source, and the other, to the ATmega signal. The switch side of the optocoupler is connected to the 6V power supply, and the gate of the CMOS circuit.

When the signal output by the microcontroller is logical 0, the optocoupler switch would open, and thus activate the pull-up network of the Cmos, which would give the mother terminal a logical 1 signal. Vice versa, when the microcontroller is outputting signal 1, the terminal of the motor would receive a 0 signal.

The wheels would move forward or backward, if the signals at the two terminals are different and stop when they are the same. Thus, we can control the direction that the card is moving.

The code we developed to control the movement of the car is referenced below.

```
void stopCar(){
      //PORTD = 0b00000000;
      PORTD &= ~(1<<3); // PD3=0
      PORTD &= ~(1<<4); // PD4=0
      PORTD &= ~(1<<5); // PD5=0
      PORTD &= ~(1<<6); // PD6=0
}
//Both wheels are 0, therefore stop
```

```
void moveCarBackwards() {
	//PORTD = 0b01010000;
	PORTD |= (1<<4); // PD4=1 --> 00010000
	PORTD |= (1<<6); // PD6=1 --> 00100000

	PORTD &= ~(1<<3); // PD3=0 --> 11110111
	PORTD &= ~(1<<5); // PD5=0 --> 11011111
}
//one wheel is 1, another 0. Therefore move backward
void moveCarForwards() {
	//PORTD = 0b00101000;
	PORTD |= (1<<3); // PD3=1
	PORTD |= (1<<5); // PD5=1

	PORTD &= ~(1<<4); // PD4=0
	PORTD &= ~(1<<6); // PD6=0
}
//0 and 1 in opposite directions. Therefore move forward

void turnright() {
	PORTD &= ~(1<<3); // PD3=0
	PORTD |= (1<<4); // PD4=1

	PORTD |= (1<<5); // PD5=1
	PORTD &= ~(1<<6); // PD6=0
}
//One wheel moving forward, another backward
void turnleft() {
	PORTD &= ~(1<<4); // PD4=0
	PORTD |= (1<<3); // PD3=1

	PORTD |= (1<<6); // PD6=1
	PORTD &= ~(1<<5); // PD5=0
}
```

Perimeter Detector

In our perimeter detector, we connected a tank circuit to the positive side of the amplifier. A non-inverting amplifier gain can be achieved using a one 100Ω resistor connected from the negative node of the amplifier to the

ground, and one 5.2kΩ resistor connected between the output side of the amplifer and the negative side of the amplifier. Lastly, the output side of the amplifier is connected with a diode that is connected to the ADC input of the ATmega microcontroller.

The tank circuit is a circuit that consists of a capacitor and inductor connected in parallel. When operating at the resonant frequency, the tank circuit absorbs maximum power. In our case, the inductor value is chosen as 1mH and the capacitor value is chosen as 100nF. The resonant frequency can be computed from $\frac{1}{\sqrt{LC}\times 2\pi}$ =15.9k. Therefore, when approaching the perimeter, which is generated by a function generator with 16kHz frequency, the LC tank circuit will absorb maximum power which makes the voltage increase significantly.

In the code shown below, we tested out the threshold votlage which the voltage should be higher than the threshold voltage when perimeter is detected. When the perimeter is detected, the car should do a u-turn.

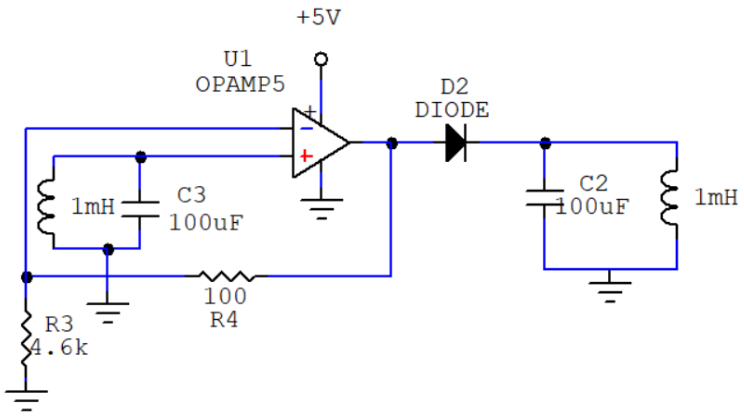

Figure 3: Circuit schematic for the perimeter detector

```
if ((v0 > thresholdVoltage) || (v1 > thresholdVoltage)) {
      stopCar();
      waitms(50);
      moveCarBackwards();
      waitms(500);
      turnright();
      waitms(1500);
      stopCar();
```

```
        waitms(50);
}
```

Metal Detector

The metal detector is composed of 2 capacitors, an inductor and a not-gate. In our circuit, the not-gate is made of a Cmos gate, which the drain of the circuit is connected to an input ADC pin of the ATmega microcontroller. (as shown in the graph below)

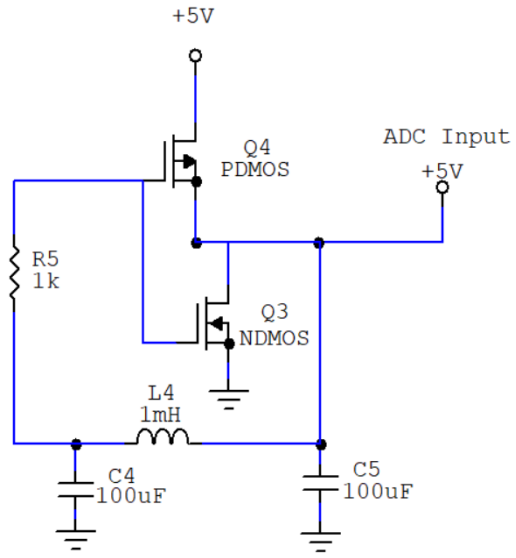

Figure 4: Circuit schematic for the metal detector

The ADC pin would measure the resonant frequency of the metal which is $\frac{1}{\sqrt{LC}\times 2\pi}$. When the inductor is near a coin, the inductor value would slightly change and thus change the input frequency to the microcontroller.

In the code, we have a threshold frequency that is higher than the frequency value without any coins, yet lower than the frequency value detected when the inductor is near a coin. If the microcontroller detects any value larger than that threshold frequency, it would then activate the servo motor and the electromagnet to pick up the coin.

The code for this process has been reference below.

```
if (f > 58050) {
      stopCar();
      waitms(100);
      moveCarBackwards();
      waitms(300);
      stopCar();
      moveArm();
}
```

Electromagnet

The electromagnet is powered by 6V and it is connected to an optocoupler as shown in the circuit below. The diode side of the optocoupler, would have one side connected to the 5V power source, and the other, to the Atmega signal. The transistor side of the optocoupler is connected to the electromagnet and a N-mosfet.

When the signal from Atmega turns to 0 (0V), the current is passed to the transistor side of the optocoupler, which makes the circuit on the right hand side shorted. On the other hand, if the signal from Atmega turns to 1 (5V), there is no current passed to the transistor side of the optocoupler, which makes the current pass to the gate of the N-mosfet and the electromagnet is turned on.

The code and circuit schematic is reference below.

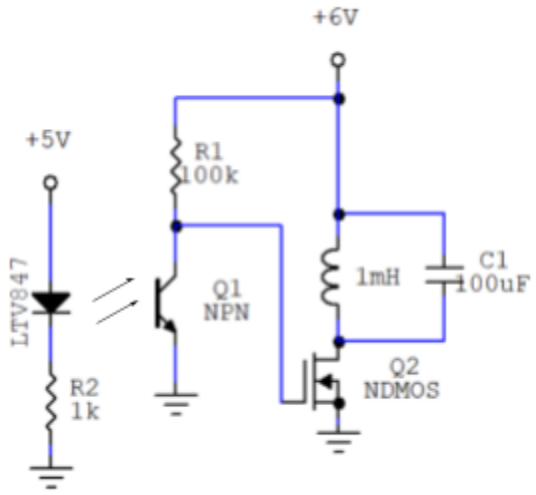

Figure 5: Circuit schematic for the electromagnet

```
void turnMagnetOff() {
      PORTD &= ~(1<<2); // PD2=0
}

void turnMagnetOn() {
      PORTD |= (1<<2); // PD2=1
}
```

Servo

We power the servo by 5V since it didn't work for us when we try to use the optocoupler to power the servo with 6V. The circuit, as shown below, is relatively simple when we power the servo with 5V. The source of the servo is connected to 5V, the ground is connected to the ground that corresponds to the 5V rail, and the PWM signal is connected to the Atmega signal.

We tested out the pulse width with the servo make file provided on canvas. Respective pulse width is tested for moving the servo to the OverBakestPosition, OverCoinPosition, Armup, Armdown. Furthermore, to make the servo moves slower and smoother, we use a for loop that change the pulse width by 10ms and also create delay of 200ms in each of the movement. Extra for loop is added to make the arm sweep while the arm is down. This prevents the arm missing the coin when the coin is detected.

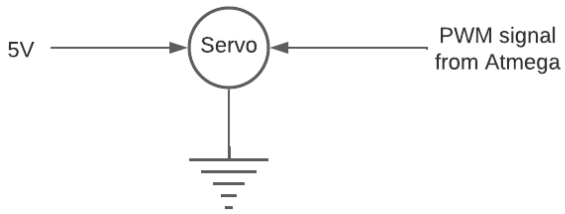

Figure 6: Circuit schematic for the servo

```
void moveArm() {
      int OverBasketPosition  = 110;
      int OverCoinPosition    = 170;
      int ArmDown             = 240; //position of arm lower position
      int ArmUp               = 90; //position of arm upper position

      turnMagnetOn();

      ISR_pw1 = ArmDown; //arm down

      waitms(1500);

      for (int i = OverCoinPosition; i > OverBasketPosition+10; i-=1)
{
            ISR_pw2 = i; //arm down
            waitms(10);
            //pw2_move500ms(i);
      }

      for (int i = OverBasketPosition + 10; i < 240; i+=1){
            //pw2_move500ms(i);
            ISR_pw2 = i; //arm down
            waitms(10);
      }

      waitms(500);

      for (int i = 230; i > ArmUp; i-=1) {
```

```
            ISR_pw1 = i; //arm down
            waitms(10);
            //pw1_move500ms(i);
      }


      for (int i = 240; i > OverBasketPosition; i-=1) {
            ISR_pw2 = i; //arm down
            waitms(10);
      }

      waitms(800);
      turnMagnetOff();
      waitms(200);
      ISR_pw2 = OverCoinPosition;
      waitms(1000);
}
```

3.7. Solution Assessment

For the microcontroller, we decided to use the ATmega microcontroller for a few reasons. First, the input power of the ATmega microcontroller is 5V which means that we can have the same voltage pin for other components such as the metal detector. Second, the group members are more familiar with the ATmega microcontroller for it is similar to the AT89 microcontroller that was used during the majority of the term. And lastly, the ATmega microcontroller does not require any soldering or composing and thus reduces chance for errors.

As for the metal detector, we had the choice between using a not-gate microcontroller, and a basic Cmos circuit. In the end, we used the Cmos circuit for it requires much less space than the not-gate microcontroller, thus easier to assemble the circuit.

For the servo motor that controls the electromagnetic, we decided to use the 6V source instead of the 5V. Even though, if we were to use the 5V power along with the optocoupler, we can reduce the noise on other components in the circuit, the idea was not working properly. Hence, we use the 6V source.

4. Life-Long Learning

Our team applied many technical concepts from prerequisites such as programming and hardware design skills. We used knowledge from past courses, especially CPEN 211 and ELEC 201, and integrated that with what we learned in ELEC 291 from problem detection to conducting our final project.

Additionally, using Github as a tool for software regression and tracking proved to be an unintentional yet experiential learning curve for us as it is a tool widely used in industry today and one that we will continue to use in the future.

We took this opportunity to learn more about shift registers. From this we gained experience with microprocessors , ATMega328P Assembly, hardware design and construction which will be very valuable for our future as electrical engineers.

5. Conclusions

Our coin picking robot is designed with programmer-set parameters for the inductors in order to ensure the readings are sensitive and stable. It is supposed to detect the smallest coin (the dime) without fail despite really minor changes in the detected frequency. Our designs ensure the robot can effectively detect the coins and pick them up using the electromagnet attached to the servo arm, and can continuously move within a preset parameter (using the guidewire) without crossing the boundary and backing up/turning when hitting the guidewire.

Despite inaccuracies in our robot's abilities to properly detect all the coins when running across it, it was still able to pick up almost all the coins (beside nickels and dimes). We as a team enjoyed playing the game and it was a very fun experience working on it. The work spent on our capacitance game project totals to 80 hours.

6. Appendix

- **Microcontroller**

Quantity	Part	Description
1	Atmel ATmega328P	8-bit AVR Microcontroller
3	BC1148CT-ND	0.1µF capacitor
1	1k ohm resistor	±5% tolerance

2	330 ohm resistor	±5% tolerance
2	5mm LED	Red
1	CTX 1085-ND	Crystal 16.00MHz
1	BO230XS USB adapter	Adaptor to interface with computer
1	Push button switch	2 pin push button

- **Robot (Electronics and**

Quantity	Part	Description
1	Ball caster kit	Ball caster to act as front wheel of car
2	Geared motor	5V DC motor
2	Servo Wheels	3D printed wheels
1	Robot chassis	CNC cut sheet metal chassis
1	4-40 screw / nut kit	
1	4 x AA battery holder	Battery holder
1	9V battery clip	Clip to attach 9V battery to chassis
1	DPDT Switch	Switch with two input and two output terminals
2	LTV847	Optocoupler
1	L7805CV	Positive Voltage Regulator
	55L104	N-Channel QFET MOSFET
	NVD2955	P-Channel QFET MOSFET
	1 mH inductor	Inductor
	IN4007 diode	Rectifier Diode
	1000 ohm resistor	±5% tolerance
1	Coin Picker Base	CNC cut sheet metal component
1	Dual micro servo bracket	CNC cut bracket for servo motors
1	Electromagnet arm	CNC cut sheet metal component

1	Coin bucket	CNC cut sheet metal component
4	2-56 machine screw / nut	Machine screws
5	4-40 machine screw / nut	Machine screws
4	Cable ties	Plastic one-time use cable ties
3	1 mH inductor	Copper coiled inductor
2	BC1148CT-ND	0.1μF capacitor
1	Electromagnet	Copper wire coiled electromagnet
2	Tower Pro MG90S	PWM interfaced servo motor with operating voltage range of 4-6V
2	Micro-servo motor horn kit	Electromagnet arm component

- **Team Designed Hardware**
  - H-Bridge
  - Metal Detector
  - Perimeter Detector
  - Electronic Magnet

- **Test Data**

	Trials #	**Frequency (Hz) Metal Detector**	**Voltage (V) Perimeter Detector**
Without Coin	1	*57200*	*0.024*
	2	*57000*	*0.022*
	3	*57500*	*0.021*
	4	*56200*	*0.023*
	5	*56800*	*0.024*
With Coin	1	*58200*	*2.012*
	2	*57900*	*2.019*
	3	*58100*	*2.120*
	4	*58800*	*2.002*

	5	*58400*	*2.032*

7. References

1. J. Calvino-Fraga. ELEC 291. Class Lecture 3, Topic: “Capacitive Sensor Reaction Game”. Department of Electrical and Computer Engineering, University of British Columbia. Jan. 28, 2022. https://canvas.ubc.ca/courses/83893/files/19111379?module_item_id=4191204
2. J. Calvino-Fraga. ELEC 291. Class Lecture 3, Topic: “555 timer and Capacitance Meter”. Department of Electrical and Computer Engineering, University of British Columbia. Jan. 28, 2022. https://canvas.ubc.ca/courses/83893/files/19249044?module_item_id=4208441
3. Atmel Corporation, “AT89LP52 Microcontroller IC 8051 MCU 8K FLASH 40-DIP” ATmel AT89LP51/2 Datasheet. https://canvas.ubc.ca/courses/83893/files/18731512?module_item_id=4101976