Department of Mechanical Engineering,
Boğaziçi University

ME331 Course PG7 Term Project

Pinball Machine

İbrahim İşsever
Arda Ulutürk
Tolga Çatak
Ümit Henkoğlu
Yasela Mollaoğlu
Ömer Ercan

Project Advisor: Sinan Öncü

10.01.2022

Table of Contents

1 INTRODUCTION 3
2 MARKET ANALYSIS 3
3 SYSTEM REQUIREMENTS 4
4 DESIGN 6
4.1 Electro-Mechanical Design 6
4.1.1 Flipper Design 7
4.1.2 Game-Begin Shooter Design 10
4.1.3 Slingshot Design 15
4.1.4 Bumper Design 16
4.1.5 Lane Design 18
4.1.6 Spinner Design 20
4.1.7 Point Counter and Life Counter Design 21
4.2 Software Design 22
4.2.1 LCD 22
4.2.2 RGB LEDs 23
4.2.3 Slingshots 23
4.2.4 Lanes 24
4.2.5 Bumper 24
4.2.6 In-Game Shooter 24
4.2.7 Long Corridor 24
4.2.8 Out of play area 25
4.2.9 Problems and Solutions in Software Design 25
4.2.9.1 Piezoelectric Sensors Problems 25
4.2.9.2 Delay Problems 25
4.2.10 Arduino Code 27
5 SYSTEM COMPONENTS 33
6 FINAL PRODUCT 33
7 WORK DISTRIBUTION 38
8 CONCLUSION 38

1 INTRODUCTION

This project report aims to illustrate the design process of the Pinball Machine. The course of this report is as follows: First, market analysis is discussed, then, system requirements will be explained briefly. Afterwards, implemented design is displayed in a detailed way with related figures of schematics and circuit diagrams. Once design is clarified, the final product is displayed with photographs, and on the work distribution part, work distribution among group members is displayed. Lastly, there remains the conclusion part where this report is summarized.

2 MARKET ANALYSIS

As a team, we had conducted a market analysis to understand significant aspects of the pinball machine. This analysis contains two steps:

1) Data about the market price and growth
2) Significant questions for the Pinball Machine

In the first step, we found two data tables about the market: Global Arcade Gaming Market Table and Used Pinball Machine Price Table. Regarding Figure 1, between 2019 and 2024, market growth will be %2 and there will be a 1.56 billion dollar increase in the market size. Moreover, we analyzed the price table. As one can see clearly from Figure 2, the price of used pinball machines increases year by year. Today, the price is about 4003 dollars. Therefore, the pinball machine is a good investment and design for the university. Lastly, we decide to make the pinball machine as little price as possible.

Figure 1: The Market Values of Gaming Market

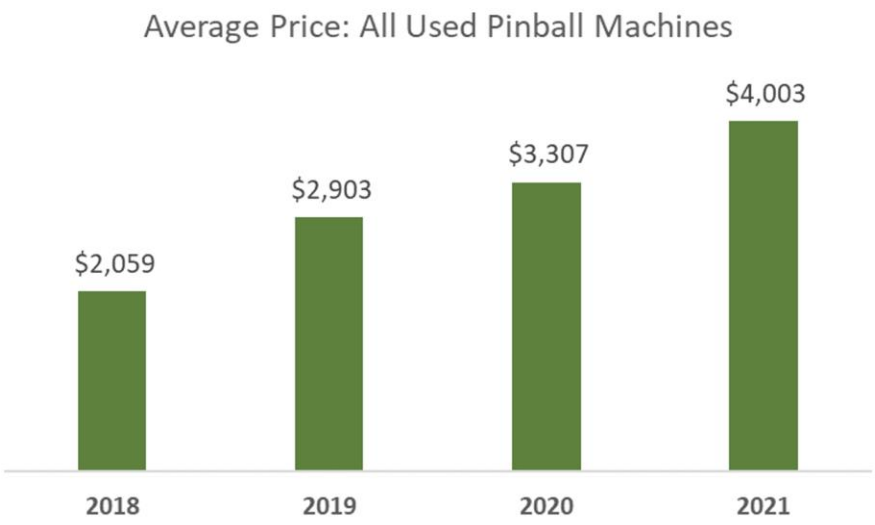

Figure 2: The Average Price Graph of Pinball Machines

In the second step, we analyzed customer feedback. As a result, we decided four most significant questions:

- What are you getting the pinball machine for?
- How Much is the Pinball Machine?
- Do You Have Enough Space to Fit the Machine?
- Do Students need a new machine for the fun?

At Bogazici University, there are several table foosball machines on the campus. However, there is no variety for the students. Therefore, the pinball machine will be a good alternative to have fun for students. Furthermore, this pinball machine costs not too much and is sustainable with the university budget. The original pinball machine size is very big for the university campus, therefore we decided to create a smaller and portable one. Lastly, we found that LED systems attract attention from the users. Therefore, we decided to attraction mode in the pinball machine. This mode will be enabled when there is no active game on the machine.

3 SYSTEM REQUIREMENTS

Our planned Pinball Machine is not so simplified from the infamous Pinball Machine. Dimensions of the machine board are going to be 90x50 centimeters. Our machine is going to have a game-begin shooter, an in-game shooter, 2 flippers, 2 slingshots, 3 lanes, 1 bumper, 1 LCD display as a scoreboard and life counter. Some of those parts can be seen from Figure 3. In Figure 3, part 1 is game-begin shooter lane, part 2 is lanes, part 3 is spinner corridor, part 4 displays the bumper, part 5 is in-game shooter lane, part 6 is slingshots. The 2 gray arms below are notorious flippers. Note that this Figure 3 misses some small parts which are added in the production process to the play area. Additional to the parts below, one can also have a more thorough understanding of the game and requirements from the design part of the report.

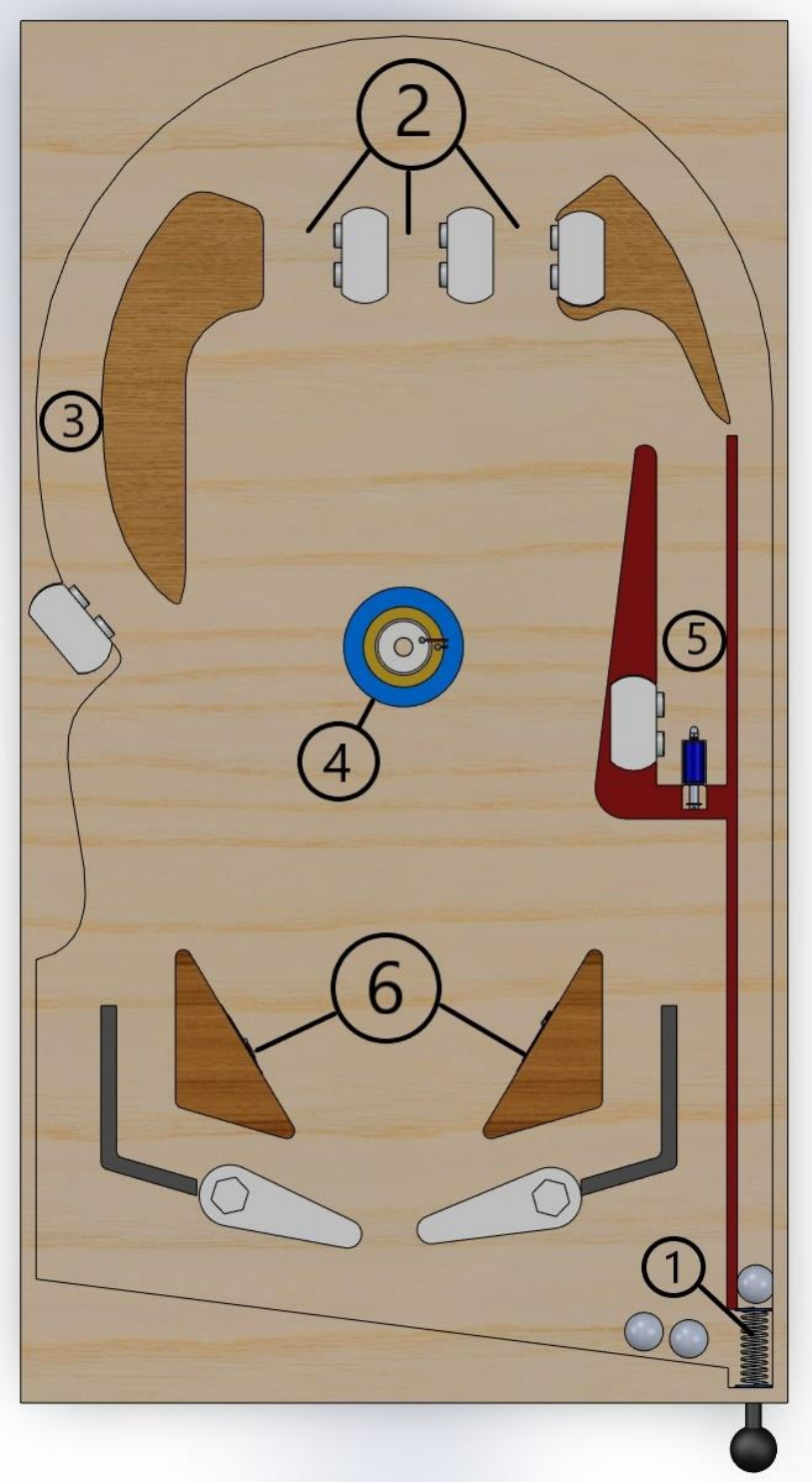

Figure 3: Parts of the Pinball Machine

4 DESIGN

In this section of the report, we divided the design into electro-mechanical and software design subsections. In those subsections, each part of the machine is discussed separately with related figures and schematics.

4.1 Electro-Mechanical Design

Below, in figures one can see the overall circuit diagram in Figure 4. As in Figure 4, we have used Arduino MEGA 2560 for our circuit as planned since Arduino UNO does not have adequate pins. Overall voltage of the system is supplied from 220V AC socket and transformed to 12V and 24V with a 12V/25A and a 24V adaptors. We needed to use an additional 24V adaptor since we upgraded some of our actuators since they were not able to supply adequate impulse. At most parts we have not used any breadboard since voltage and current might have damaged the breadboard, instead, we have built circuits with cables and solder.

For the game-table itself, first we designed the layout of the game area. Then we sent them to the CNC-Router in the .DWG format. For the floor we used an 8mm thick MDF board. For the other parts in the game, we used two 18mm thick MDF boards. The reason to use 2 boards on top of each other is to provide the necessary height since the ball has 40mm diameter. After the parts were produced, we had to make some corrections since there was a misunderstanding between the designer and the producer. They had given a 1cm offset to the outside frame. Therefore, we needed to make some new parts using the remaining boards with a Jigsaw cutting machine. To fix the parts onto the table, we used Mitreapel instant adhesive. Finally, we added RGB LED lights that changes color periodically to make it nostalgic.

We also had planned to use plexiglass to cover the table. This would prevent the ball from leaving the game area mistakenly and would make a more aesthetic appearance. We also wanted to put special paintings and stickers on the parts for a more entertaining display. We would simulate the campus of our university in the game area. For example, the in-game shooting area is designed to hold the ball for a few seconds so that the player can rest during the gameplay and this area could be made similar to the South Campus Square where people can relax and take a breath.

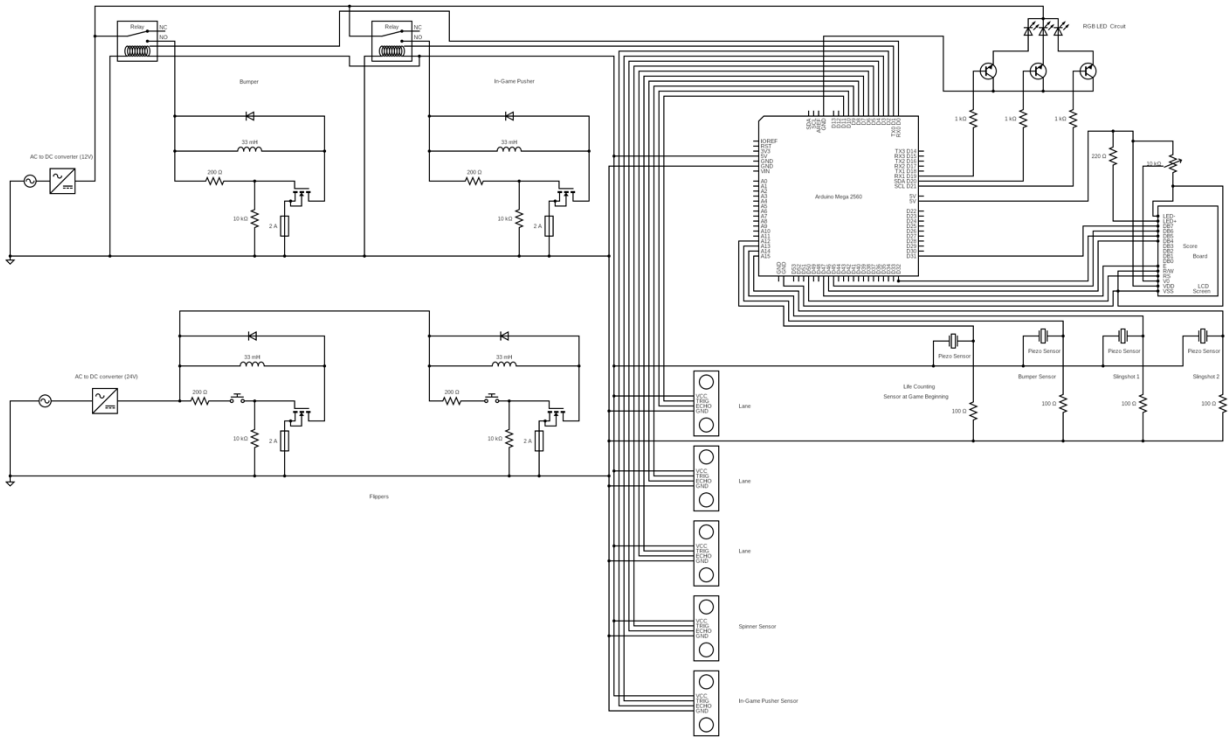

Figure 4: The Circuit Diagram

4.1.1 Flipper Design

In electrical design of flippers, solenoids are used as actuators. The activation of the solenoids is provided by the MOSFETs due to the fact that the direct activation of solenoids without using MOSFETs would cause solenoids to draw too much current and result in possible deformation of some components in the circuit. 2A fuse is used to protect the wiring and the appliance in the case of unwanted situations. If the current flowing through the fuse is higher than 2 amps, the wire in the fuse melts and breaks the circuit, guaranteeing the protection of other components and preventing the short circuit situation in pursuit of false wiring. The existing flyback diode is included for the current to pass through the opposite direction in the recharge phase of the solenoid.

The flippers are not connected to Arduino since it does not have automated process. The flippers are controlled by the user manually via push buttons. The power supply for the flippers was 12 volts before, since the initially chosen type of solenoid actuators could be driven with a potential difference of 12 volts. In the process of testing the actuators, although 10 mm stroke distance of 12V solenoids checked out to be adequate, the preferred impulse was not sufficient. In the virtue of the conducted market survey, the 24 volts push-pull solenoid actuator with the same stroke of 10 mm was decided on. This change required the new power supply of 24 volts. Since batteries and accumulators are off our budget, an adjustable voltage regulator was utilized.

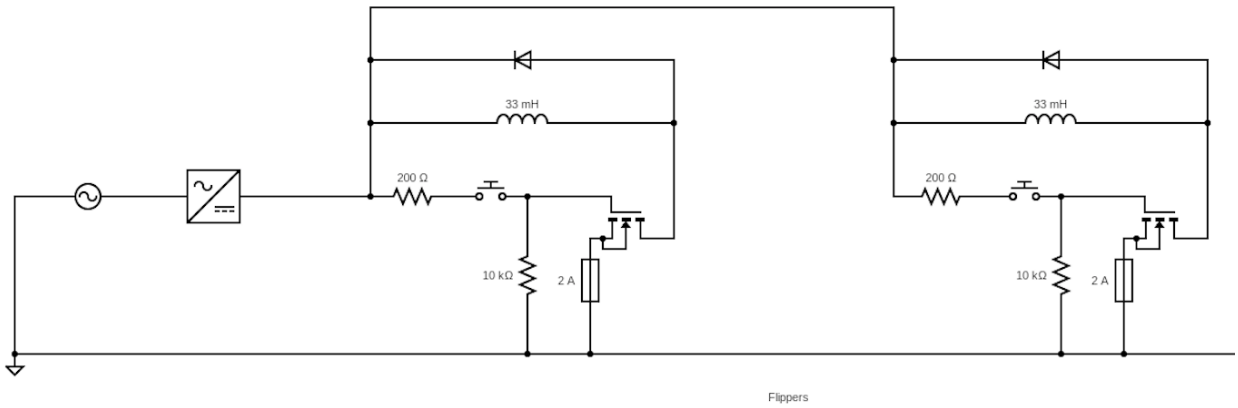

Figure 5: The Flippers' Circuit Diagram

The mechanical design of flippers can be seen in Figure 6. The design consists of 2 parts: the upper part and lower part. The reason why the hole in the upper part is hexagonal is that the rotational movement of the lower part can be transferred to the upper part. Only the part of the lower part that goes inside the upper part has a hexagonal structure. The rest of lower part is circular because the lower part must be able to rotate inside the bearing. A solenoid pushes the end of the lower part of the flipper, and it creates the moment about the vertical part of the lower part. Therefore, the translational motion of solenoids can be converted into rotational movement of flipper arms. To connect the solenoids to the lower part of the flippers we used strings so that they would work in different angles as well. The string and bearing connection can be seen in Figure 7.

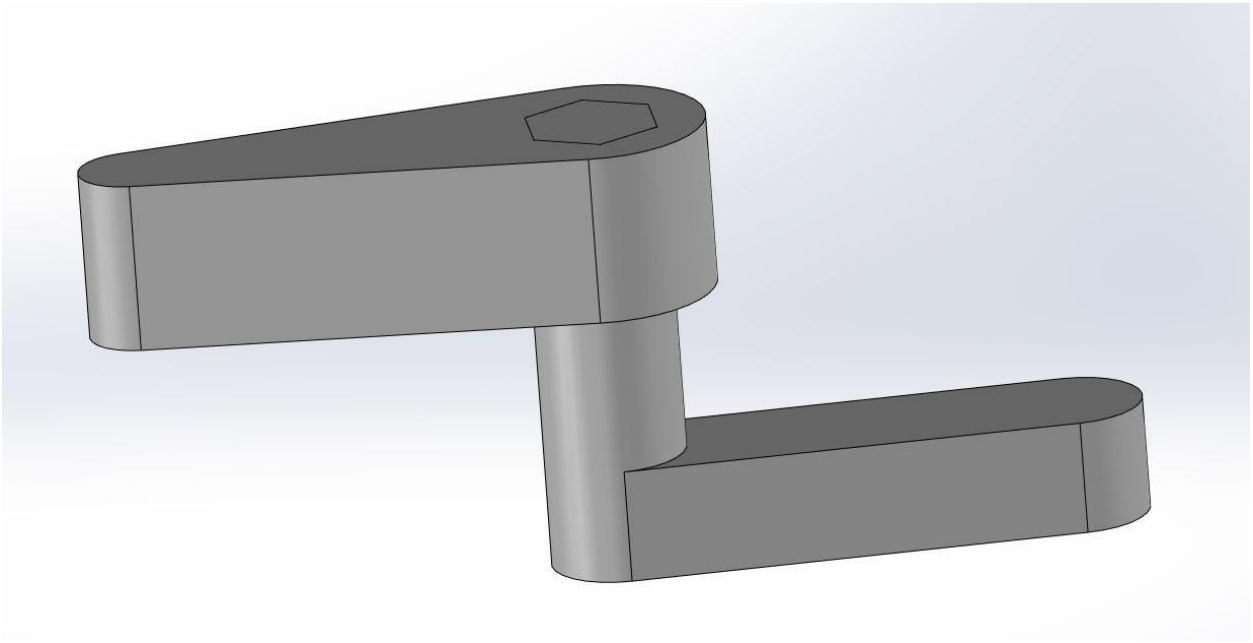

Figure 6: The Flipper in 3D Model

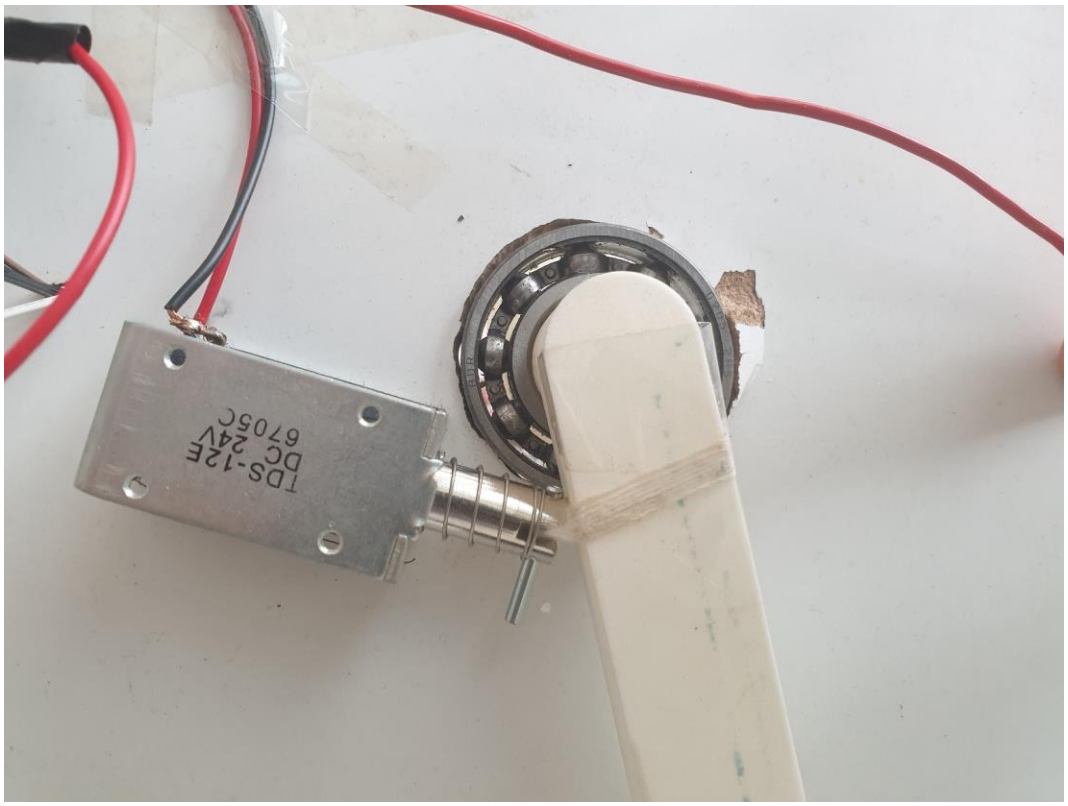

Figure 7: The connection between the flipper and solenoid via string and the connection between the bearing and the flipper.

After the parts were 3D printed, the hexagonal part of the lower part would not fit into the hole., therefore we had to grind and check again and again until it fitted perfectly. This taught us how important the hole and shaft tolerances are.

To fix the flippers into the playground, we needed to come up with a solution that enabled rotation only in upward direction but no translation in any direction. To achieve this, we used bearings but the bearings we bought were industrial and were designed for high rpm's. They had a grease in it that slowed them. So, we put them in thinner until they were clean and sprayed WD40 to reduce friction.

We also didn't think the bearing housing solution beforehand didn't design the necessary holes for them. That's why we used drills to make a hole and expand it until the bearings barely fit. We used a close fit method to fix them. This taught us how significant it is to think for the production while making designs. The final product of flippers can be seen in Figure 8.

Figure 8: The flippers which are fixed to the playing field.

4.1.2 Game-Begin Shooter Design

Game-Begin Shooter design is mostly mechanical rather than electrical. In here, only electrical component we have used was a piezo sensor in order count remaining lives of the player. We were planning to use an ultrasonic sensor, but we wanted to save some space, therefore we have changed our sensor from ultrasonic to piezo. Player has 3 lives in the game. Each time the ball passes on the sensor, life count is reduced and displayed in the LCD screen as red hearts.

The mechanical design of Game-Begin Shooter (GBS) can be seen in Figure 9. It consists of 3 parts: a spring, a rod and a handle. The main purpose of the handle is to provide convenience to the user while pulling the rod. The rod is connected to the spring by a circular object which the ball rests on. When the user pulls the handle, the user actually pulls the rod, the spring is compressed. Then, the spring gains potential energy due to compression. We get the energy needed to throw the ball from the potential energy of the spring.

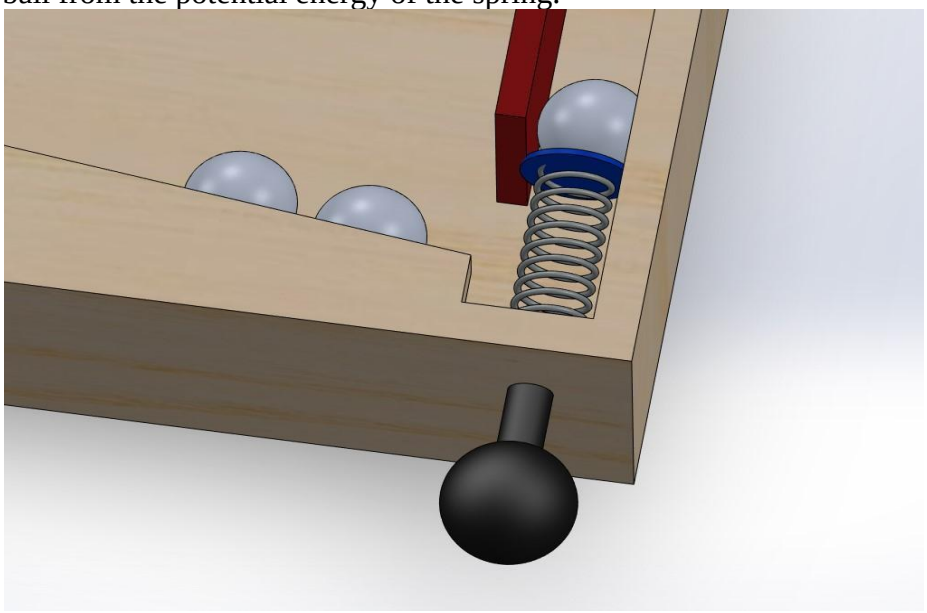

Figure 9: Game Begin Shooter (GBS)

Due to some small changes in the general layout of the game area, the ball would not come in front of the spring when the game was lost. Therefore, we had to change the location of the game-shooting mechanism and put it further down on the table. This required also making a new floor for this area.

To connect the new floor plate with the main plate, we used flat brackets. The brackets can be seen in Figure 10. We also tried different kinds of springs to balance the speed and force. In this part, we did not encounter many problems.

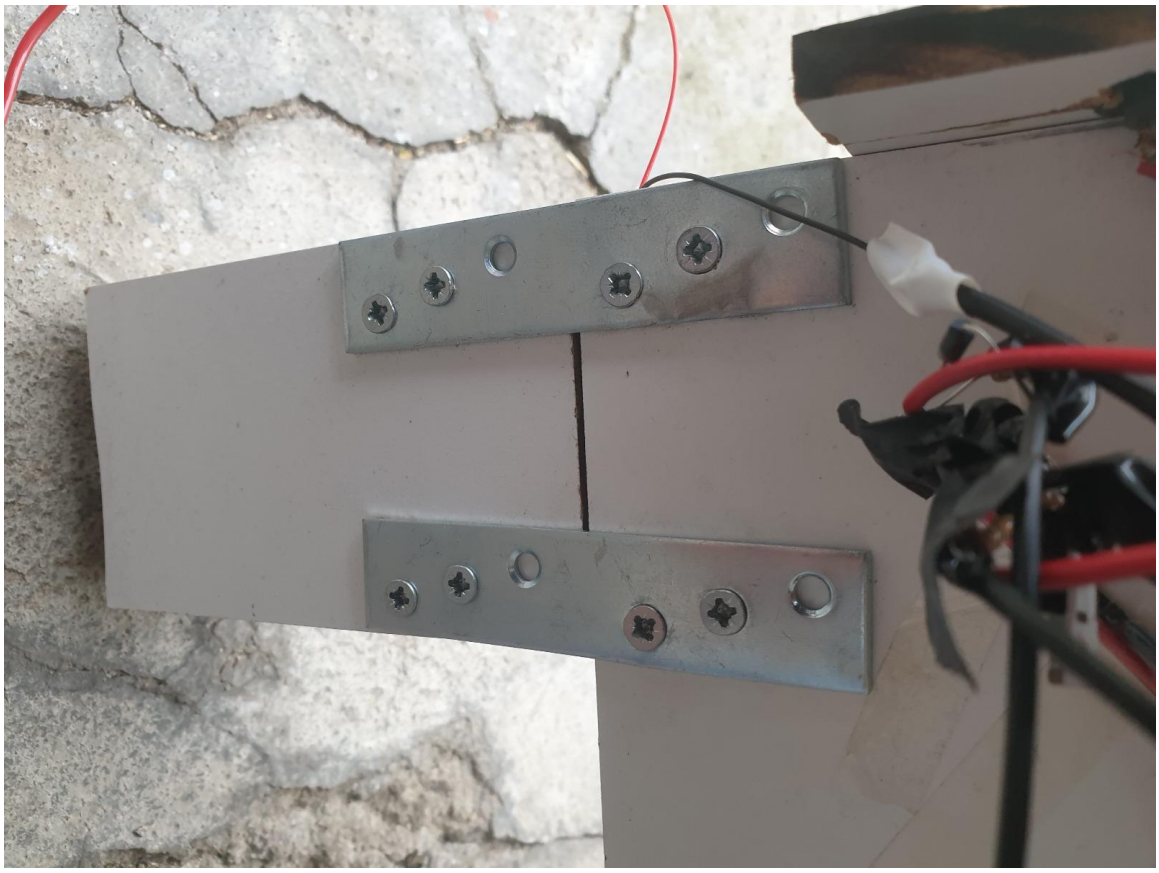

Figure 10: The brackets

We also put a piezo sensor in the game-begin area so that we could understand whether the game has ended or began. The final product of Game Begin Shooter can be seen in Figure 11.

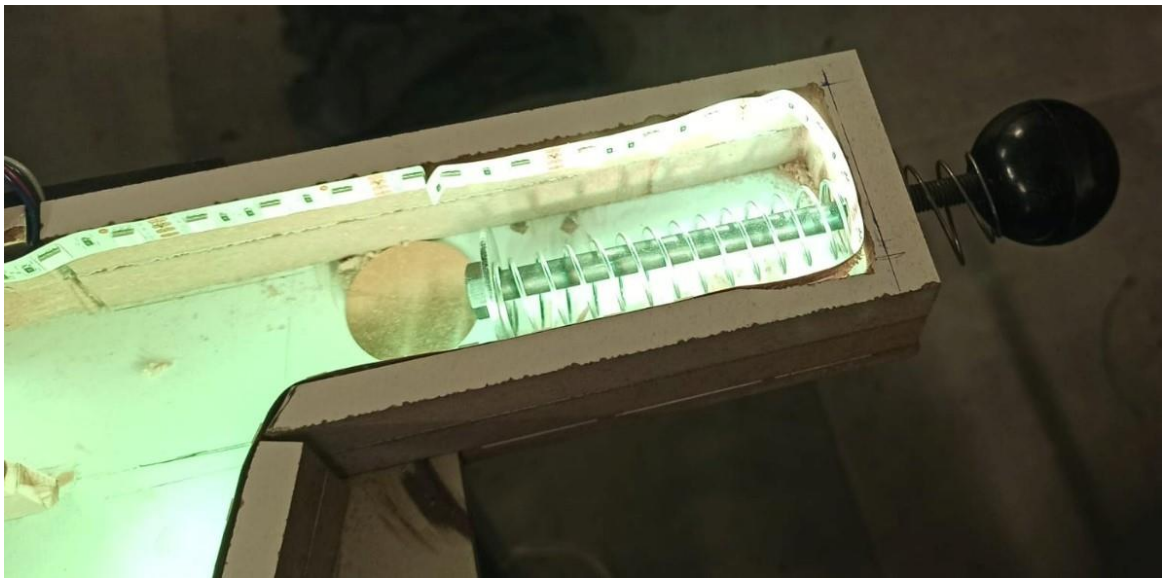

Figure 11: Game Begin Shooter

In-Game Shooter Design

The aim of the in-game shooter design is to shoot the ball into the game when the ball comes to a specified area. There is no manual intervention in this part, the whole process is automated via a closed loop feedback system. The circuit design of this part can be seen in Figure 12**.** We have used a small 12V pull-push solenoid in this part as an actuator. Solenoid is activated by the Arduino when the ultrasonic sensor detects the ball and 12V voltage difference acts on the terminals of solenoid. We have used a relay module to activate the solenoid by Arduino, the reason we used this configuration was because Arduino could not supply enough power to the actuator. Normally open (NO) output terminal of the relay module switch is on only when current passes through inputs coming from Arduino pins. When the ultrasonic sensor reads a predetermined value, the input signal from Arduino is HIGH, switch is closed and 12V acts on the circuit. MOSFET, fuse, resistors and flyback diode usage are for the same reasons with the flipper circuit. That part of the design will not be covered further in this report since the flipper section provides the necessary information. To summarize, when the ball comes to the special field in the pinball map, a solenoid gives a kick to the ball. The In-Game Shooter mechanical design can be seen in Figure 14**.**

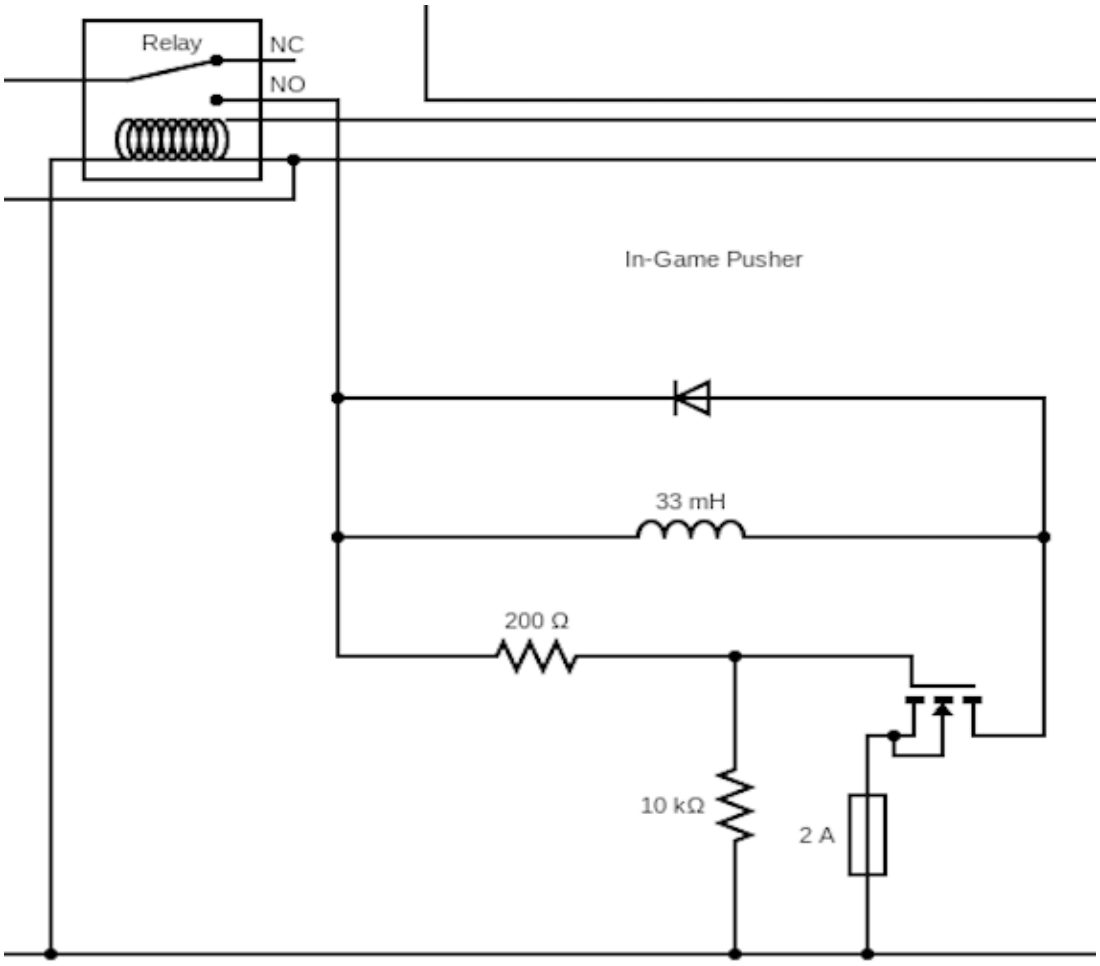

Figure 12: In Game Shooter Circuit Diagram

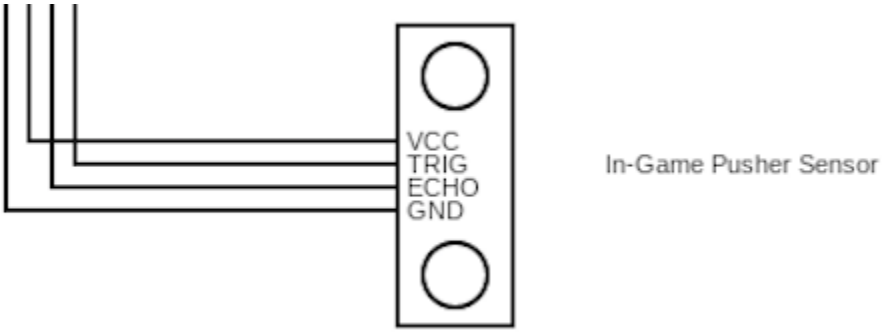

Figure 13: In Game Shooter Sensor Diagram

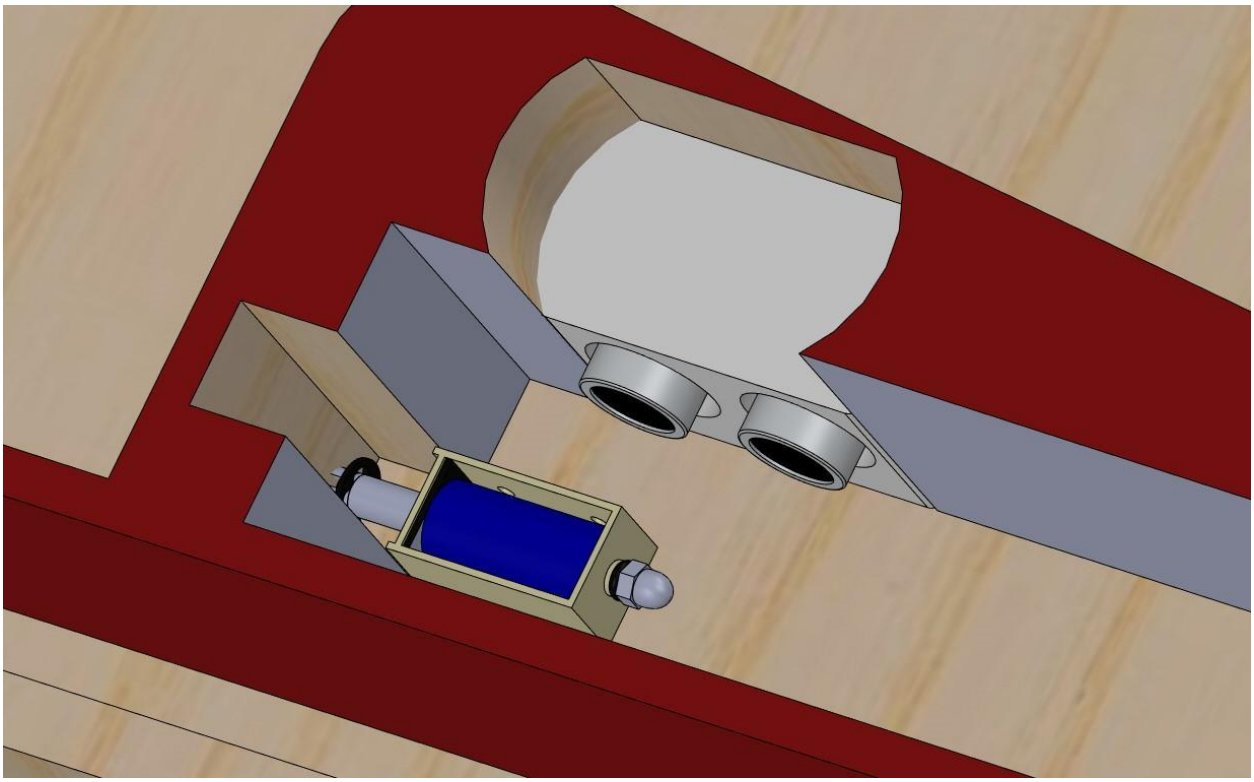

Figure 14: The In-Game Shooter 3D Design

A significant problem that we encountered in the in-game shooter part was that we could not read correct values since the ultrasonic sensor was reading the solenoid itself when solenoid is activated. We had to change the location of the solenoid to avoid this problem. But after small adjustment, it would also go into a continuous loop since the rod of the solenoid was misleading the sensor again. Therefore, we had to put it further back and the problem was solved. The final form of In Game Shooter can be seen in Figure 15.

Figure 15: The final form of In Game Shooter

4.1.3 Slingshot Design

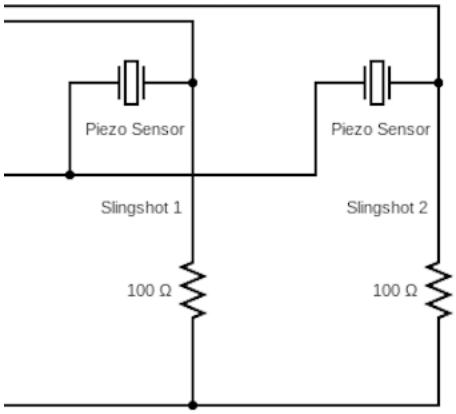

Figure 16: The Slingshot Circuit Diagram

In slingshots, we used piezo sensors inside. When the ball hits the hypotenuse of the slingshot, score is updated and displayed on the LCD screen. The Slingshot Design can be seen in Figure 17.

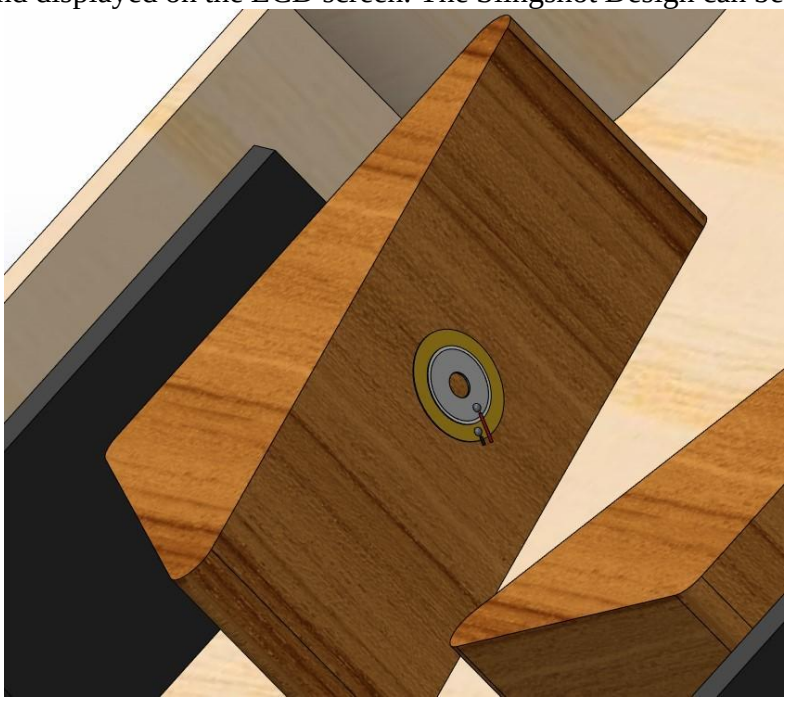

Figure 17: The Slingshot 3D Design

We used an angle grinder to make the necessary cuts for the lower part of the piezo sensors. Otherwise, the sensors would not fit perfectly, and the inputs were misleading because they were vibrating. We also made holes for the piezo cables in the slingshot parts and table. The final product of The Slingshot can be seen in Figure 18.

Figure 18: The Slingshots on the play field.

4.1.4 Bumper Design

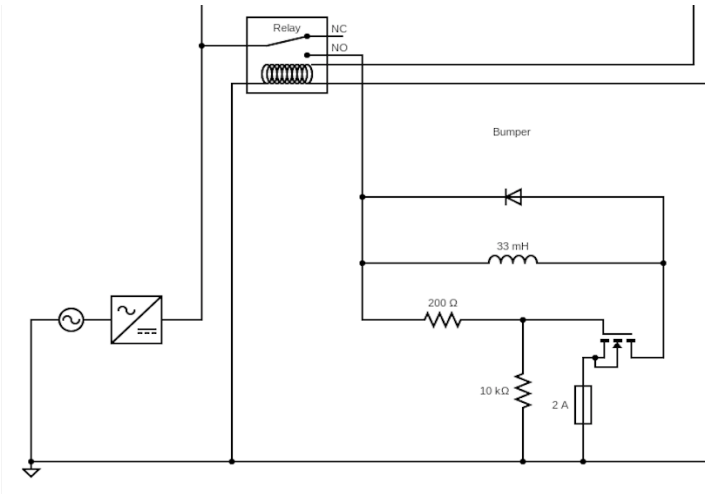

Figure 19: The Bumper Circuit Diagram

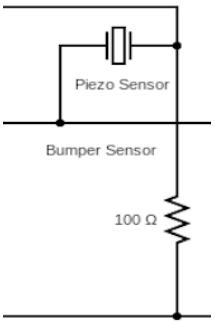

Figure 20: The Bumper Sensor Circuit Diagram

Bumper circuit design is similar to in-game shooter design electrically. Differently, the bumper is activated by the piezo sensor rather than an ultrasonic sensor. When ball touches the bumper, a piezo sensor placed on the top of the bumper detects pressure and activates the solenoid. Solenoid is connected to the bumper handle below the playground and pulls the bumper head downward, bumper head hits the ball and accelerates the ball. We were planning to design a mechanical switch, but it was not feasible, instead, we have used a relay module to activate the system. As an actuator, we have used a 24V pull solenoid. The Bumper Design can be seen in Figure 19.

In the mechanical design of the bumper, we used a conical shape so that the ball would be pushed outward to the direction where it came from. We put the piezo disk over it because of the flat surface. We made some cavities for the cables to go through the cylindrical hole below the bumper.

In the backside of the table, we used 3D printed parts to attach the solenoid to the table. We made a hole perpendicular to the cylinder and put a pin through the solenoid rod to assemble the solenoid rod and the bumper. With this configuration, we achieved the desired movement of the bumper. When the solenoid was activated with the piezo input, it would go down and go up to its original position. We adjusted the threshold such that the piezo sensor would detect any impact from any direction. The backside of the bumper can be seen in Figure 21.
Although this configuration works sufficiently, we could use many small piezo sensors around the cylinder of the bumper for higher precision, if we had a higher budget. This could be an area in which the project can be improved.

We also had planned to put 3 bumpers, but due to budget and time problems, we could only put 1 bumper. The mechanical design of bumper can be seen in Figure 22 and the actual product of bumper can be seen in Figure 23.

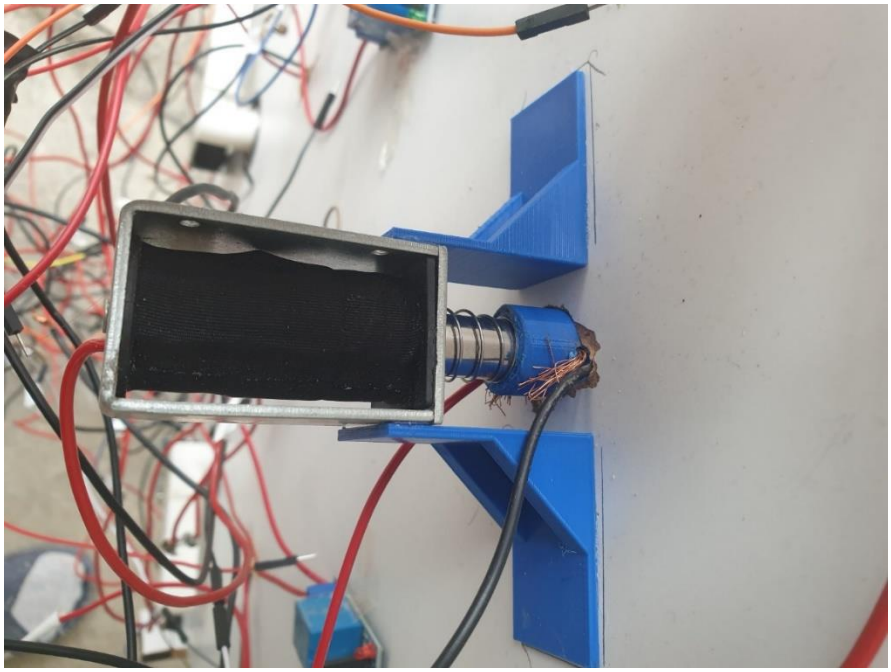

Figure 21: The backside mechanism of Bumper.

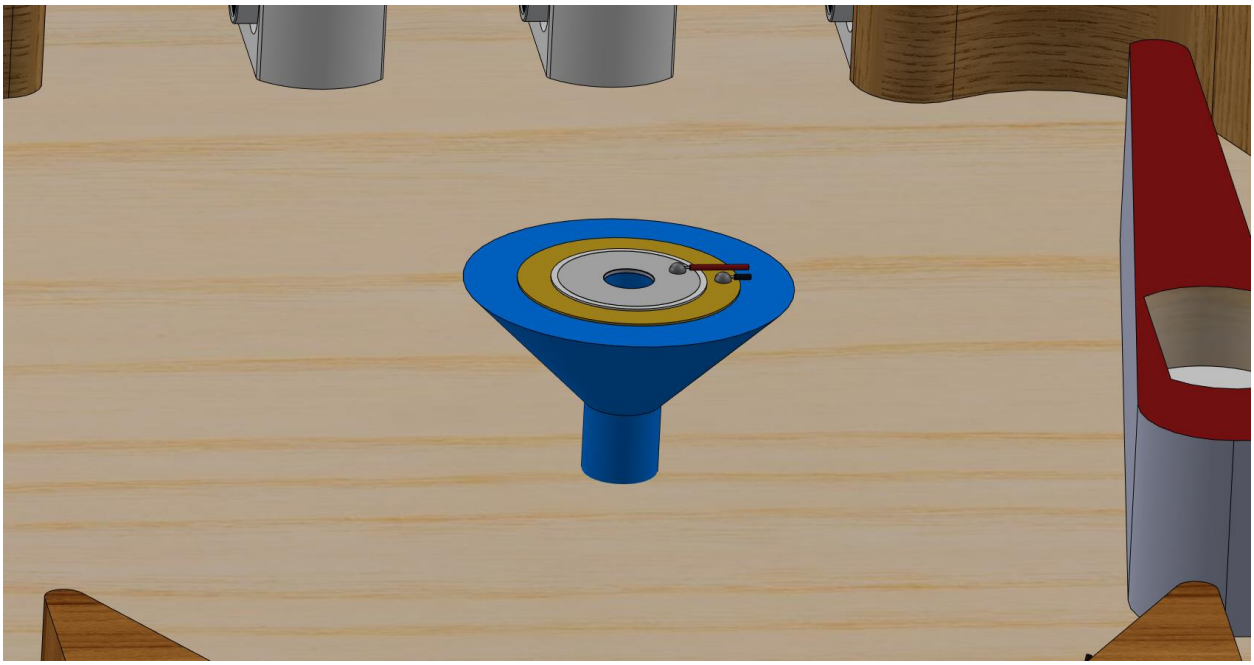

Figure 22: The Bumper 3D design

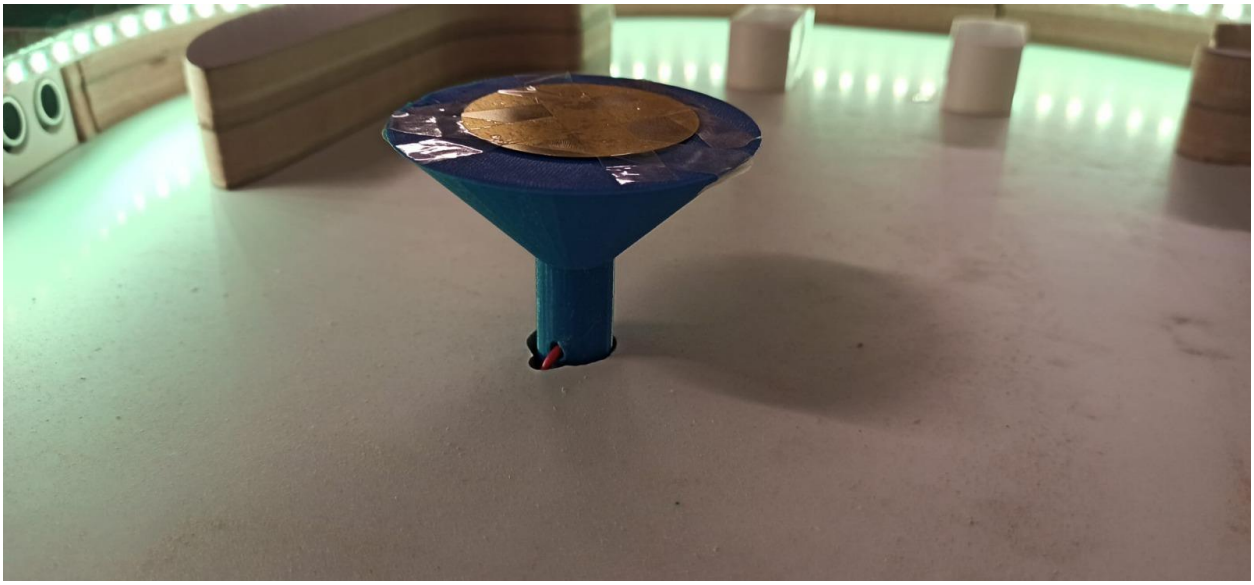

Figure 23: The actual product of the Bumper.

4.1.5 Lane Design

In lanes, we again used ultrasonic sensors to detect if the ball passes through lanes. When the ball passes through lanes, points are added to the scoreboard. The lanes can be seen in Figure 25.

In the lane and the other parts where we used other ultrasonic sensors, we designed, and 3D printed covering parts for the sensors so that they would not be damaged if the ball hit them. We didn't know the precise location of these sensors and could not design the holes. Therefore, instead of having them CNC cut, we had to drill the holes ourselves. This allowed us flexibility for choosing the locations. This gave us some advantages since we had to make arrangements in the layout according to the different ball options (metal ball, pinball ball, table tennis ball, foosball).

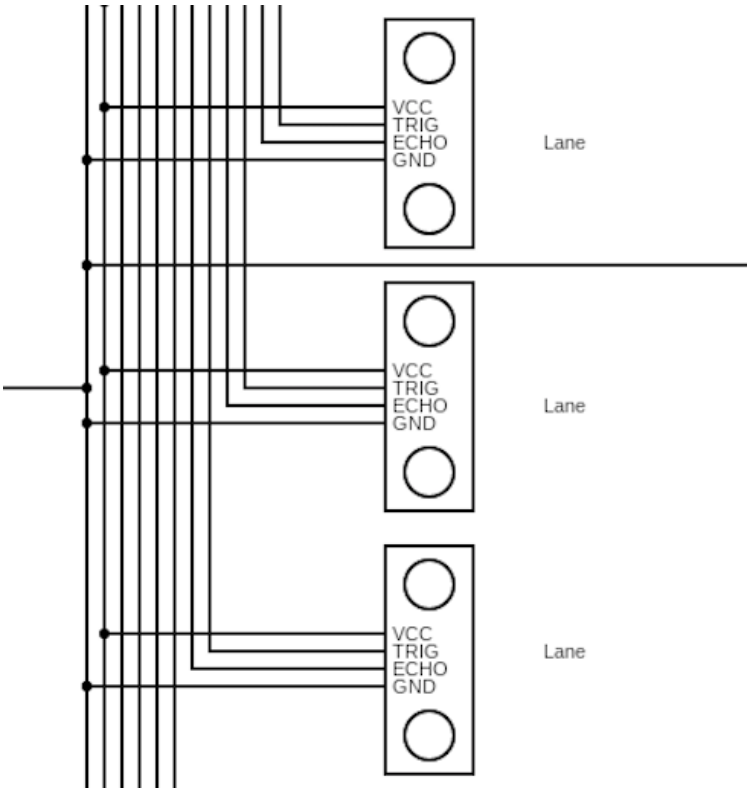

Figure 24: The Lane Circuit Diagram

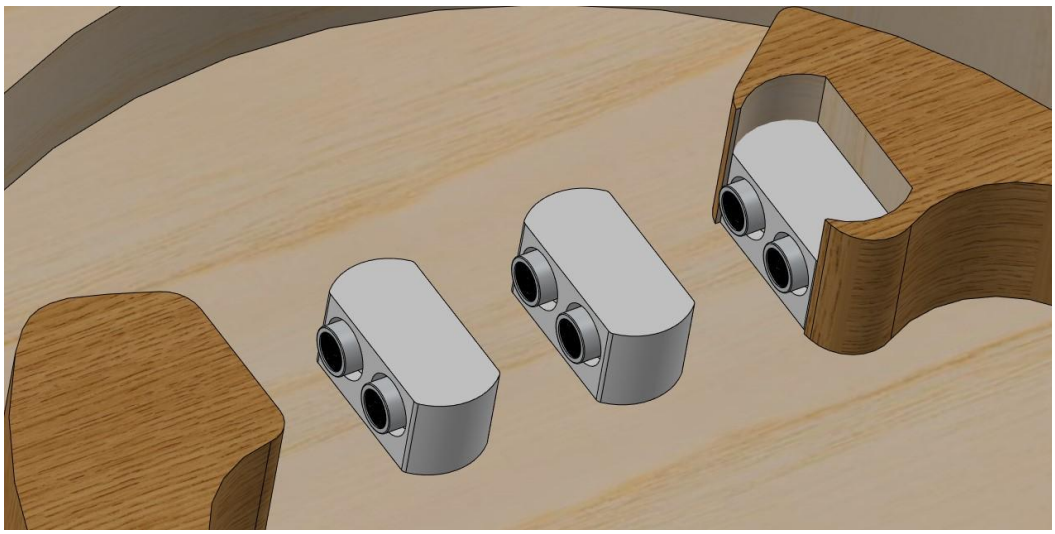

Figure 25: The Lanes

4.1.6 Spinner Design

In the spinner part, we had originally thought that the user can take 2 different scores. One would be given when the ball enters the spinner. The second would be given when the ball exits the spinner. If the ball could not leave the spinner even if it touches the exit door, only the first score would be given. This would be controlled via ultrasonic sensors. If the ball would pass through the first sensor which would be placed at the end of the first door, the initial point would be given. Then, if it passes through the second ultrasonic sensor which would be placed at the beginning of the exit door, the system would wait 2 seconds, and if the ball would not pass through the first sensor again, the system would give additional points. This mechanism would be to prevent miscalculation of points in situations in which the ball would barely reach the second sensor and would come back without exiting the spinner hall. The first door design we made of the spinner can be seen in Figure 27.

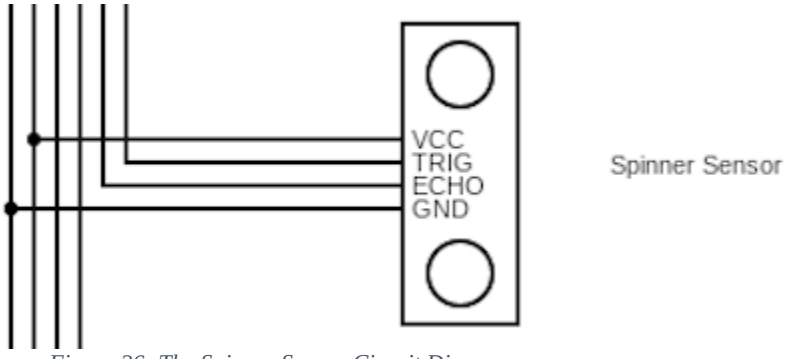

Figure 26: The Spinner Sensor Circuit Diagram

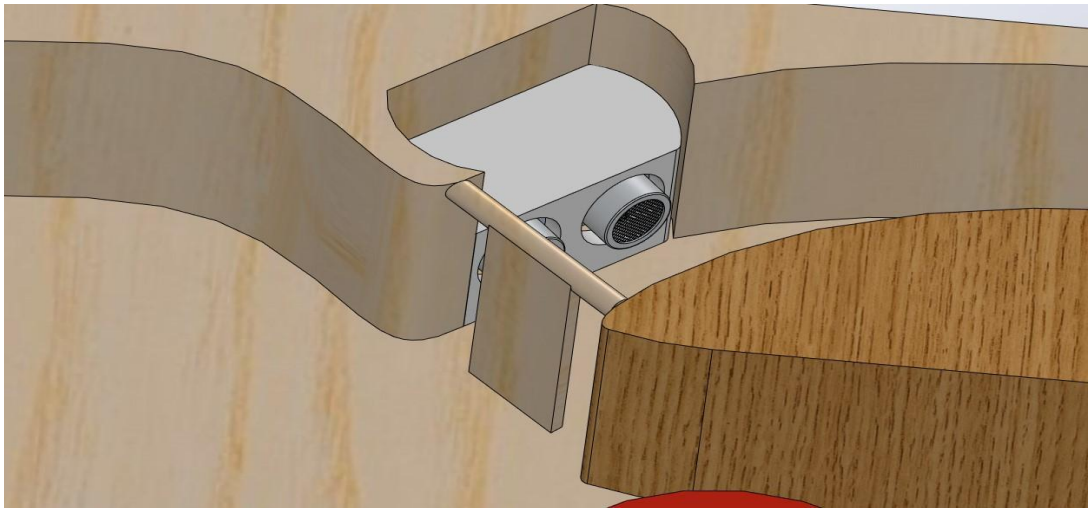

Figure 27: The first door design of The Spinner.

However, we didn’t realize our thoughts due to the following reasons.
-There was the risk of the ball being stuck behind the door since we chose the table tennis ball which is very light.
-It would create additional expenses because we would have to use additional sensors.
-We needed to work very precisely since the part was too small. However, we didn’t have much experience nor the necessary tools.
-We decided to use our time in more important aspects.
The modified product of The Spinner and the comparison of the part which we produced ourselves due to error in manufacturing with the part we buy can be seen in Figure 28.

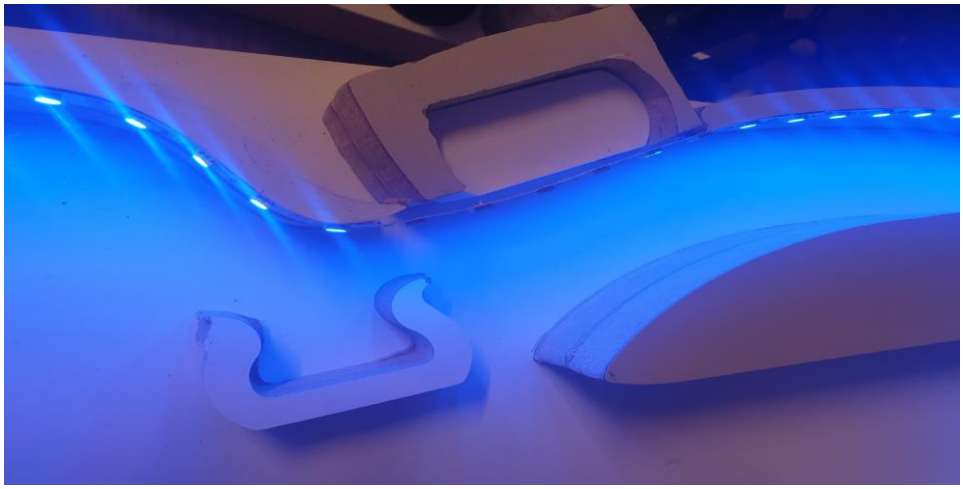

Figure 28: The final form of spinner and the comparison between old and new ultrasonic sensor home.

4.1.7 Point Counter and Life Counter Design

The scores are kept and added up by the Arduino code and with each change in the score the LCD screen display is updated, and the score is displayed. The LCD screen is fixed in the lower part of the table so that the player can easily follow its current point and number of lives. LCD pins are connected to the Arduino. The hearts in Figure 37 shows the remaining lives at the game start.

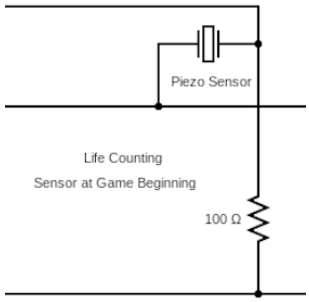

Figure 29: The Life Counting Sensor at Game Beginning

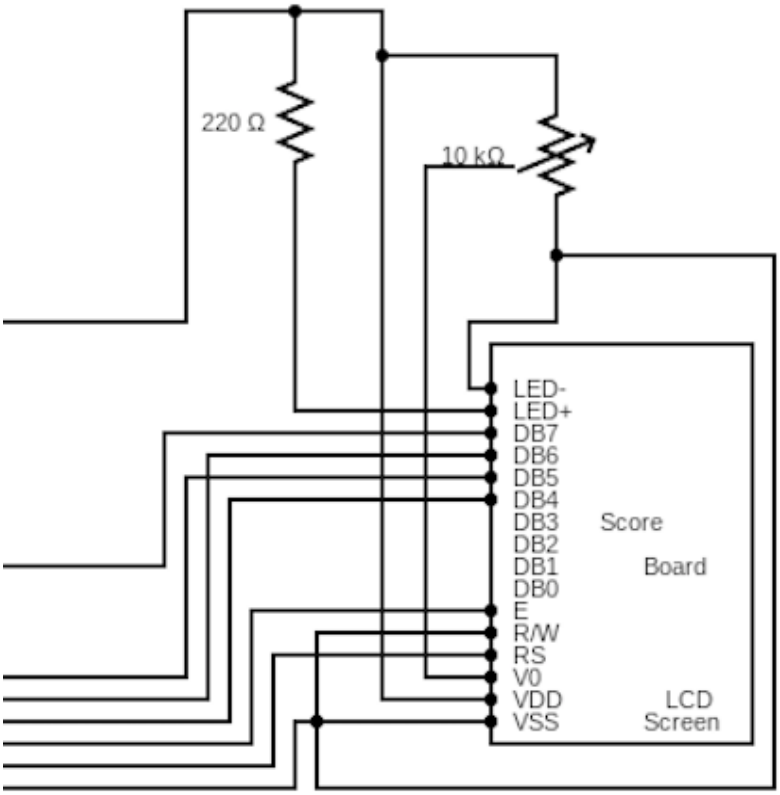

Figure 30: The LCD Circuit Diagram

4.2 Software Design

In this section of the Design heading, we illustrate the software design process of the project.

4.2.1 LCD

In the LCDdisplay function, firstly we used LiquidCrystal_I2C.h library. Secondly, Lcd was defined as LiquidCrystal_I2C lcd(0x27,16,2). Then, by using the lcd.createChar function, the heart shape was created. For Lcd, there are four modes:

- LCDdisplay
- LCDwaittime
- LCDdisplay_newhighscore
- LCDgame_over

In LCDdisplay mode, the Lcd shows the remaining life and current score. If the game doesn't start yet, it shows a high score and "Pull to Start." information. Secondly, in LCDwaittime mode, Lcd returns "Wait" information to show that the system needs time to activate the play mode. Thirdly, in LCDdisplay_newhighscore mode, Lcd returns the newest high score for the player. Lastly, when life is equal to zero, by using LCDgame_over mode, Lcd shows the total score and "Game Over " information.

4.2.2 RGB LEDs

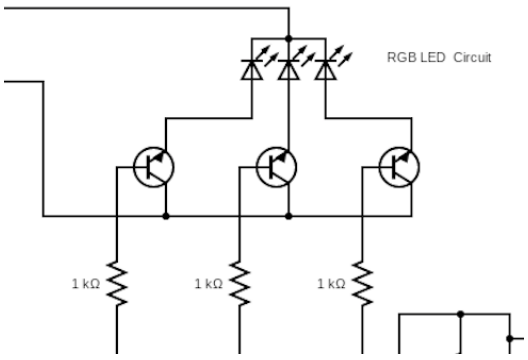

Figure 31: The RGB LEDs' Circuit Diagram

LEDs which show the situation of the game are the critical parts of the pinball machine. Before the setup code, three different colors were defined as random(256) that returns numbers between 0 and 256. Moreover, milis function is critical to count the seconds to change the lights for every three minutes. For the LEDs, there are two different functions:

- LEDs_NOGAME
- LEDS_INGAME

In LEDs_NOGAME mode, the LEDs light according to the random number that came from random() function. Therefore, this mode gives a different color each time. Secondly, in LEDS_INGAME mode, we used switch mode to create three cases according to randRGB = random(3) variable. In this mode, there are three different color types for three different cases. Every three minutes, the randRGB variable changes and the user observes different colors of the LEDs.

4.2.3 Slingshots

For the slingshots, the piezoelectric sensors were used to take input. The working principle of the code consists of three parts:

- Taking the initial inputs
- Calculating input difference
- Adding 30 points for suitable scenarios.

In the first step, piezoelectric sensors take initial inputs. After the game starts, the sensors continue to take input from the game. In the Arduino, the new and old values are stored. When the difference between new value and old value is more than 15, the score increases 30 points. When the difference between new and old value is smaller than 15, the loop will stop and the function continues to calculate input difference.

4.2.4 Lanes

The lanes are significant for the game mechanics. They have two significant roles:

- Taking input for the game start
- Taking the input for the score

For this part, ultrasonic sensors are used to take inputs to detect the position of the ball. If these sensors return true, the game starts immediately. Moreover, after the first passing, each pass increases the score by 20. If the ball passes all the corridors, the score increases by 100.

4.2.5 Bumper

Information of the ball hitting the bumper is gathered by a piezoelectric sensor placed on top of the bumper. As of all piezoelectric sensors, two successive inputs from the piezoelectric sensor are stored and compared with respect to the threshold value to determine whether an impact is present. If such impact is present on the ball,

- Increase score
- Initiate solenoid
- Wait for piezoelectric values to return neutral which are smaller than the threshold value.

4.2.6 In-Game Shooter

In game shooter is the part where the ball is led to the game automatically, shown as number 5 in Figure 3: Parts of the Pinball Machine. We read the input of an ultrasonic proximity sensor placed right next to the in-game shooter, or a solenoid. If ultrasonic sensor detects the ball,

- Increase score
- Delay for 1.5 seconds
- Then initiate solenoid

4.2.7 Long Corridor

Long corridor is the left long corridor designated as number 3 in Figure 3: Parts of the Pinball Machine. Presence of a ball in the corridor is determined by an ultrasonic sensor placed right after the beginning of the corridor. But since the ball can come back from the corridor, we don't want to increase the score twice, so we created a variable which stores an integer from 1 to 9 to indicate where the last input was acquired from. The ultrasonic sensor at the long corridor corresponds to the number 1 and if the last input equals number 1, presence is not counted as an input. If sensor detects the ball and last input variable is different than 1,

- Increase score by 20
- Assign last input to 1
- Increase corridor counter by 1.

If corridor counter equals 4,

- Increase score by 100.
- Set corridor counter to 0.

4.2.8 Out of play area

Ball hitting out of bounds is understood from the piezoelectric sensor placed right under the pulling spring. Since this piezoelectric is susceptible to greater impact than other sensors, the threshold value is larger for this sensor. If the sensor detects the presence of the ball,

- Decrease life by 1

4.2.9 Problems and Solutions in Software Design

In this section, we discuss the problems we have encountered during software design process of the project, and the corresponding solutions we were able to come up with.

4.2.9.1 Piezoelectric Sensors Problems

In this project, several piezoelectric sensors have been used. Before using it in the project, we decided to analyze them in the prototype. However, we observed that after the sensors read the input, they don't return to their initial value which is 0. For instance, the sensor reads the input as 500, however, after a few seconds, the Ardunio doesn't read 0 values. It reads 450 and this number decreases one by one.

Because of the problem in the sensors, we realized that the software design may help us to overcome this problem. In the code, before the loop, the sensors take input initially. Then, in the loop, they continue to take input. Therefore, we can store two different input values in Arduino : 1. New value 2. Old Value. When the difference between new value and old value is bigger than threshold, this means the sensors take vibration from the ball. As a result, by using this algorithm, we can detect the position of the ball easily and increase the point.

4.2.9.2 Delay Problems

Since the Arduino processor has only one core, Arduino can only process one task at a time. Therefore, delaying the compiler even for very little amounts can interfere with input readings, overall functions, structure etc. We wanted to use delay() function to flash LED lights in attraction mode and when a combo is made in game. But delaying in attraction mode can interfere with the input which starts the game and delaying in-game can interfere with all the inputs which determines the game functions. So, we used a delay only for the ultrasonic proximity sensor for 10 μs which is necessary for the sensor we used.

"One of the upper corridors" corresponds to the three corridors in the upper play area shown as number 2 in Figure 3: Parts of the Pinball Machine. "Long corridor" corresponds to the left corridor shown as number 3, bumper is shown as number 4, "in game shooter" as number 5, slingshots as number 6, in Figure 3: Parts of the Pinball Machine. "Out of play area" is determined by the sensor placed on number 1.

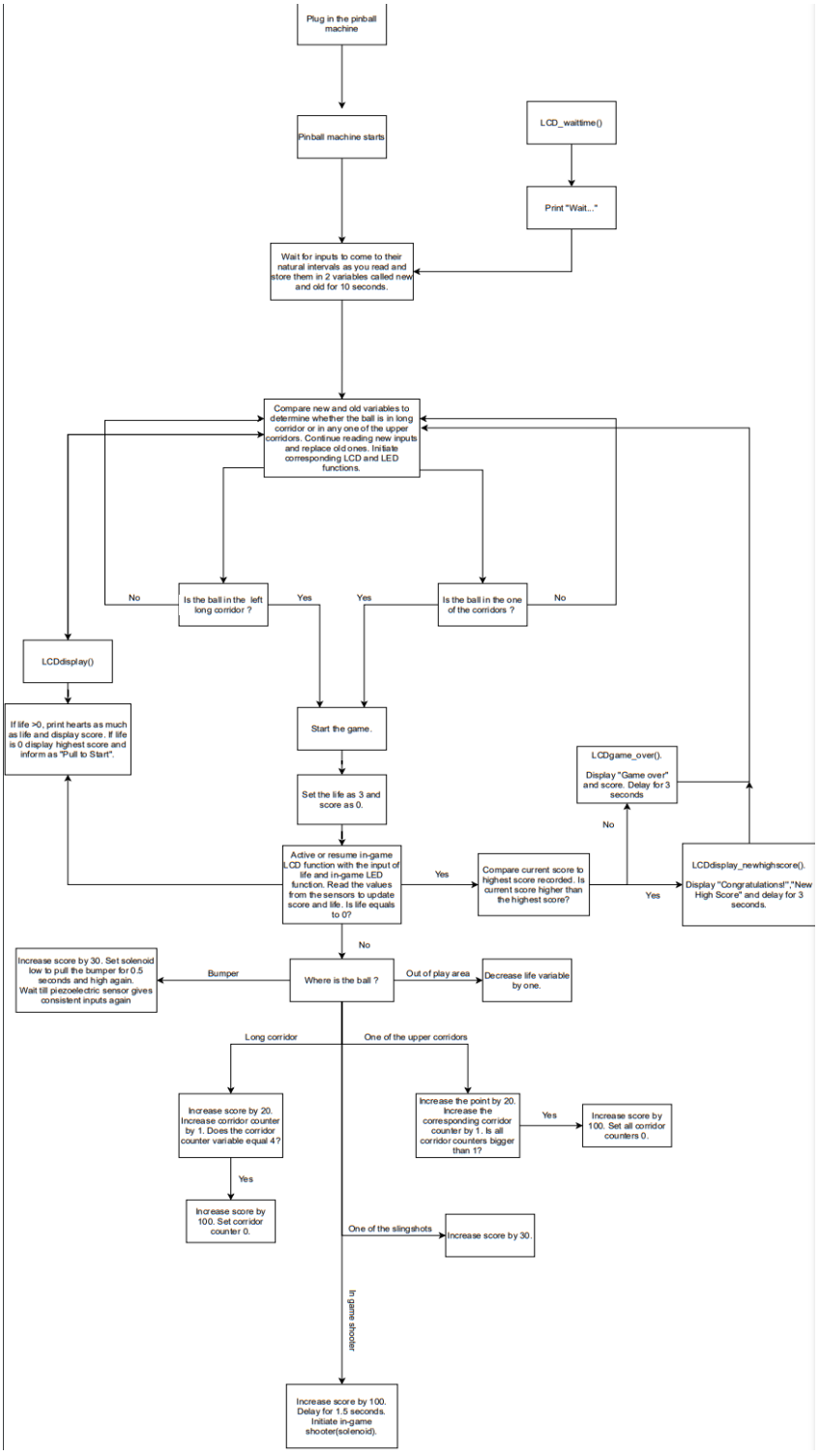

Figure 32 The Pseudo Code

4.2.10 Arduino Code

Pinball Machine Code

PG7

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);

int A_trig = 3; int disA; int corridorA =0;
int A_echo = 2;
int B_trig = 9; int disB; int corridorB =0;
int B_echo = 8;
int C_trig = 10; int disC; int corridorC = 0;
int C_echo = 11;
int D_trig = 12; int disD; int corridorD=0;
int D_echo = 13;
int SS_trig = 4; int disSS; int SS_Selonoid = 51;
int SS_echo = 5;
int distance;
long duration;
int last_input = 0;

int SlingSL = A0; int SlingLN; int SlingLO;
int SlingSR = A1; int SlingRN; int SlingRO;
int OOPA = A3; int OOPAN; int OOPAO;
int Bumperpiezo = A2; int BumperN; int BumperO; int BumperSelonoid = 53;

int starttime = 0; int time_new=0;
int Score = 0; int HighestScore = 0;
int life = 0;
int Start = 0;

int Green = 47; int GREENRAN = random(256); int randRGB = random(3);
int Red = 49;   int REDRAN = random(256);
int Blue = 7;   int BLUERAN = random(256);

byte heart[] = {
  B01010,
  B11111,
  B11111,
  B01110,
  B00100,
  B00000,
  B00000,
  B00000
};


void setup() {
  delay(1000);
  lcd.begin();
  lcd.createChar(0,heart);

  pinMode(A_trig, OUTPUT);
  pinMode(A_echo,INPUT);
  pinMode(B_trig, OUTPUT);
  pinMode(B_echo,INPUT);
  pinMode(C_trig, OUTPUT);
  pinMode(C_echo,INPUT);
  pinMode(D_trig, OUTPUT);
  pinMode(D_echo,INPUT);
  pinMode(SS_trig, OUTPUT);
```

```
58    pinMode(SS_echo,INPUT);
59
60    pinMode(SS_Selonoid, OUTPUT);
61    pinMode(BumperSelonoid, OUTPUT);
62    digitalWrite(SS_Selonoid,LOW);
63    digitalWrite(BumperSelonoid,LOW);
64
65    Serial.begin(9600);
66  }
67
68  void loop() {
69    while((millis()-starttime)<10000){
70      Ultrasonic(SS_trig, SS_echo);
71      disSS = distance;
72      Ultrasonic(D_trig, D_echo);
73      disD = distance;
74      Ultrasonic(C_trig, C_echo);
75      disC = distance;
76      Ultrasonic(B_trig, B_echo);
77      disB = distance;
78      Ultrasonic(A_trig, A_echo);
79      disA = distance;
80      SlingLN = analogRead(SlingSL);
81      SlingRN = analogRead(SlingSR);
82      BumperN = analogRead(Bumperpiezo);
83      OOPAN = analogRead(OOPA);
84
85      SlingLO = SlingLN;
86      SlingRO = SlingRN;
87      BumperO = BumperN;
88      OOPAO = OOPAN;
89      LCDwaittime();
90      time_new = millis();
91    }
92    if((millis()%time_new)>=10000){ // Change color every 10 seconds.
93      time_new = millis();
94      GREENRAN = random(256);
95      REDRAN = random(256);
96      BLUERAN = random(256);
97    }
98    LCDdisplay(life, Score, HighestScore);
99    LEDs_NOGAME(GREENRAN, REDRAN, BLUERAN);
100
101   Ultrasonic(SS_trig, SS_echo);
102   disSS = distance;
103   Ultrasonic(D_trig, D_echo);
104   disD = distance;
105   Ultrasonic(C_trig, C_echo);
106   disC = distance;
107   Ultrasonic(B_trig, B_echo);
108   disB = distance;
109   Ultrasonic(A_trig, A_echo);
110   disA = distance;
111   SlingLN = analogRead(SlingSL);
112   SlingRN = analogRead(SlingSR);
113   BumperN = analogRead(Bumperpiezo);
114   OOPAN = analogRead(OOPA);
115
116   if(disA>10 || disA<5){
117     Start = 1;
118     while(disA>10 || disA<5){
119       Ultrasonic(A_trig, A_echo);
120       disA = distance;
121     }
122   }
```

```
123     if(disB>10 || disB<5){
124       Start = 1;
125       while(disB>10 || disB<5){
126         Ultrasonic(B_trig, B_echo);
127         disB = distance;
128       }
129     }
130     if(disC>12 || disC<7){
131       Start = 1;
132       while(disC>12 || disC<7){
133         Ultrasonic(C_trig, C_echo);
134         disC = distance;
135       }
136     }
137     if(disD>12 || disD<7){
138       Start = 1;
139       while(disD>12 || disD<7){
140         Ultrasonic(D_trig, D_echo);
141         disD = distance;
142       }
143     }
144
145     SlingLO = SlingLN;
146     SlingRO = SlingRN;
147     BumperO = BumperN;
148     OOPAO = OOPAN;
149
150     if(Start == 1){
151       life = 3;
152       Score = 0;
153     }
154     while(life != 0){
155         LCDdisplay(life, Score, HighestScore); // LCD display
156         LEDS_INGAME(randRGB); // LEDs
157
158         Ultrasonic(SS_trig, SS_echo);
159         disSS = distance;
160         Ultrasonic(D_trig, D_echo);
161         disD = distance;
162         Ultrasonic(C_trig, C_echo);
163         disC = distance;
164         Ultrasonic(B_trig, B_echo);
165         disB = distance;
166         Ultrasonic(A_trig, A_echo);
167         disA = distance;
168         SlingLN = analogRead(SlingSL);
169         SlingRN = analogRead(SlingSR);
170         BumperN = analogRead(Bumperpiezo);
171         OOPAN = analogRead(OOPA);
172
173         if(disA>10 || disA<5){
174           Score += 20;
175           last_input = 2;
176           corridorA++;
177           while(disA>10 || disA<5){
178             Ultrasonic(A_trig, A_echo);
179             disA = distance;
180           }
181         }
182         if(disB>10 || disB<5){
183           Score += 20;
184           last_input = 3;
185           corridorB++;
186           while(disB>10 || disB<5){
187             Ultrasonic(B_trig, B_echo);
```

```
188             disB = distance;
189           }
190         }
191         if(disC>12 || disC<7){
192           Score += 20;
193           last_input = 4;
194           corridorC++;
195           while(disC>12 || disC<7){
196             Ultrasonic(C_trig, C_echo);
197             disC = distance;
198           }
199         }
200         if((disD>12 || disD<7) && last_input != 1){
201           Score += 20;
202           last_input = 1;
203           corridorD++;
204           while(disD>12 || disD<7){
205             Ultrasonic(D_trig, D_echo);
206             disD = distance;
207           }
208         }
209         if(disSS>10 || disSS<5 ){
210           Score += 100;
211           last_input = 5;
212           delay(1500);
213           digitalWrite(SS_Selonoid,LOW);
214           delay(500);
215           digitalWrite(SS_Selonoid,HIGH);
216           delay(500);
217           while(disSS>10 || disSS<5){
218             Ultrasonic(SS_trig, SS_echo);
219             disSS = distance;
220           }
221         }
222         if(abs(BumperN-BumperO)>15){
223           Score += 30;
224           last_input = 6;
225           digitalWrite(SS_Selonoid,LOW);
226           delay(500);
227           digitalWrite(SS_Selonoid,HIGH);
228           delay(500);
229           while(abs(BumperN-BumperO)>15){
230             BumperO = BumperN;
231             BumperN = analogRead(Bumperpiezo);
232           }
233         }
234         if(abs(SlingLN-SlingLO)>15){
235           Score += 30;
236           last_input = 7;
237           while(abs(SlingLN-SlingLO)>15){
238             SlingLO = SlingLN;
239             SlingLN = analogRead(SlingSL);
240           }
241         }
242         if(abs(SlingRN-SlingRO)>15){
243           Score += 30;
244           last_input = 8;
245           while(abs(SlingRN-SlingRO)>15){
246             SlingRO = SlingRN;
247             SlingRN = analogRead(SlingSR);
248           }
249         }
250         if(abs(OOPAN-OOPAO)>600){
251           life--;
252           while(abs(OOPAN-OOPAO)>600){
```

```
            OOPAO = OOPAN;
            OOPAN = analogRead(OOPA);
          }
        }
        if(corridorA && corridorB && corridorC){
          Score += 100;
          corridorA = 0;
          corridorB = 0;
          corridorC = 0;
        }
        if(corridorD == 4){
          Score += 100;
          corridorD = 0;
        }

        SlingLO = SlingLN;
        SlingRO = SlingRN;
        BumperO = BumperN;
        OOPAO = OOPAN;

        if((millis()%time_new)>=120000){ // Change colors every two minutes in game.
          time_new = millis();
          randRGB++;
          if(randRGB == 3){
            randRGB = 0;
          }
        }
        if(life == 0){
          Start = 0;
          if(Score > HighestScore){
            LCDdisplay_newhighscore();
            delay(3000);
            Score = 0;
          }else{
            LCDgame_over(Score);
            delay(3000);
            Score = 0;
          }
        }
    }
  }
  void Ultrasonic(int Trigpin, int Echopin){
    digitalWrite(Trigpin,LOW);
    delayMicroseconds(2);
    digitalWrite(Trigpin, HIGH);
    delayMicroseconds(10);
    digitalWrite(Trigpin,LOW);
    duration = pulseIn(Echopin,HIGH);
    distance = 0.034 * duration /2 ;
  }
  void LCDdisplay(int life, int Score, int HighestScore){
    if (life==3) {
       lcd.setCursor(4,0);
       lcd.write((byte)0);
       lcd.print(" ");
       lcd.write((byte)0);
       lcd.print(" ");
       lcd.write((byte)0);
       lcd.print(" ");
       lcd.setCursor(0,1);
       lcd.print("Score:");
       lcd.print(Score);}
       else if(life ==2){
       lcd.setCursor(6,0);
       lcd.write((byte)0);
```

```
318         lcd.print(" ");
319         lcd.write((byte)0);
320         lcd.print(" ");
321         lcd.setCursor(0,1);
322         lcd.print("Score:");
323         lcd.print(Score);}
324         else if(life == 1){
325         lcd.setCursor(7,0);
326         lcd.write((byte)0);
327         lcd.print(" ");
328         lcd.setCursor(0,1);
329         lcd.print("Score:");
330         lcd.print(Score);
331         } else {
332          lcd.setCursor(1,0);
333          lcd.print("HS:");
334          lcd.print(HighestScore);
335          lcd.setCursor(1,1);
336          lcd.print("Pull to Start.");
337         }
338   }
339   void LCDwaittime(){
340     lcd.setCursor(5,0);
341     lcd.print("Wait...");
342   }
343   void LCDdisplay_newhighscore(){
344      lcd.setCursor(0,0);
345      lcd.print("Congratulations!");
346      lcd.setCursor(1,1);
347      lcd.print("New High Score");
348   }
349   void LCDgame_over(int Score){
350     lcd.setCursor(3,0);
351     lcd.print(" Game Over");
352     lcd.setCursor(0,1);
353     lcd.print("Score:");
354     lcd.print(Score);
355   }
356   void LEDs_NOGAME( int GREEN, int RED, int BLUE){
357       analogWrite(Green, GREEN);
358       analogWrite(Red, RED);
359       analogWrite(Blue, BLUE);
360   }
361   void LEDS_INGAME(int var){
362     switch(var){
363     case 0:
364     analogWrite(Green, 255);
365     analogWrite(Red, 0);
366     analogWrite(Blue, 0);
367     break;
368     case 1:
369     analogWrite(Green, 0);
370     analogWrite(Red, 255);
371     analogWrite(Blue, 0);
372     break;
373     case 2:
374     analogWrite(Green, 0);
375     analogWrite(Red, 0);
376     analogWrite(Blue, 255);
377     break;
378     }
379   }
```

5 SYSTEM COMPONENTS

Before listing the components that are planned to use in our project, it is useful to explain how we came up with this component list first. In the beginning, we designed each part of the machine separately, then we combined all designs in a single circuit diagram as in Figure 4: The Circuit Diagram efficiently. Separate designs were consisting common components, by combining all designs we saved some of them and came up with less components. Our component list is as following:

- 1 12V/5A Adaptor
- 1 24V Adaptor
- 4 2A Fuse
- 3 IRF3205 MOSFET
- 1 IRLZ44 MOSFET
- 2 12V Push-Pull Solenoid
- 2 24V Push-Pull Solenoid
- 6 1N5819 Diode
- 5 HCSR-04 Ultrasonic Sensor
- 2 Pushbuttons
- 4 100Ω Resistor
- 4 200Ω Resistor
- 1 220Ω Resistor
- 5 10,000Ω Resistor
- 3 1000Ω Resistor
- 1 LCD Screen
- 3 TIP120 Transistors
- 5 meters RGB led strip
- 3 50mm Piezo Disc
- 1 35mm Piezo Disc
- 2 50x90 8mm Thick MDF Board
- 2 50x90 18mm Thick MDF Board

6 FINAL PRODUCT

In this section, we display some photographs of the end-product. Related figures can be seen below.

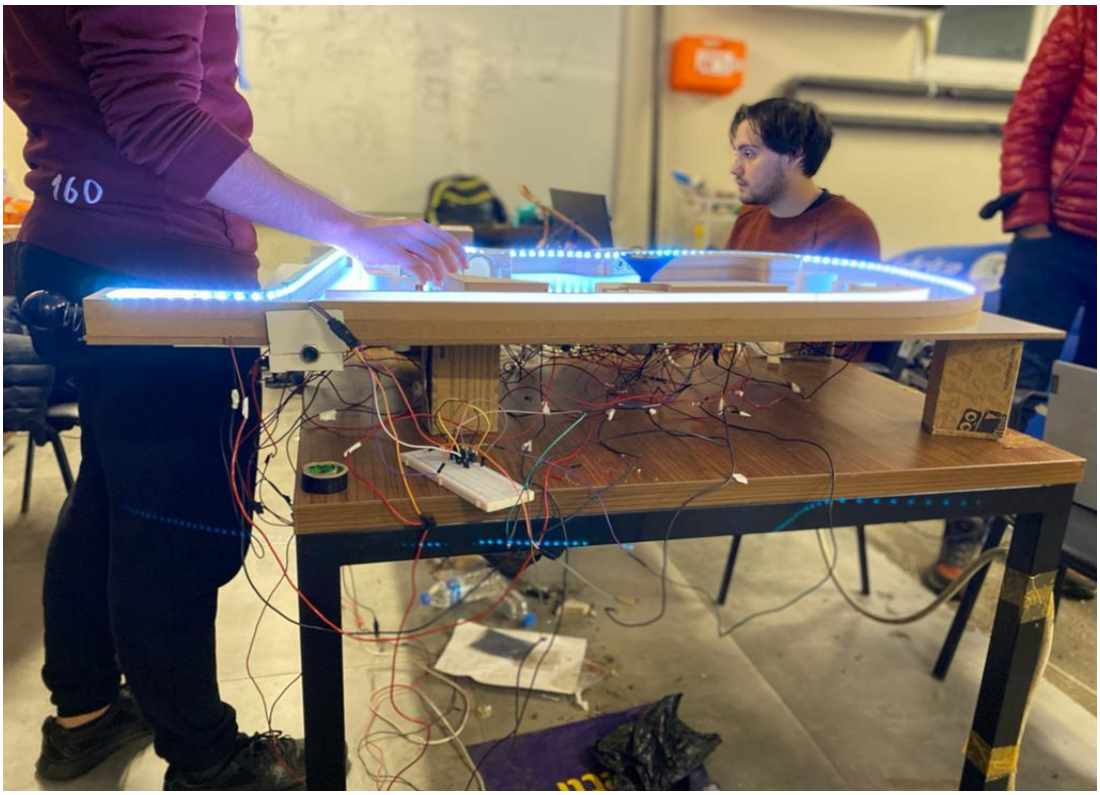

Figure 33: The Side View of Pinball Machine

Figure 34: The LCD Screen

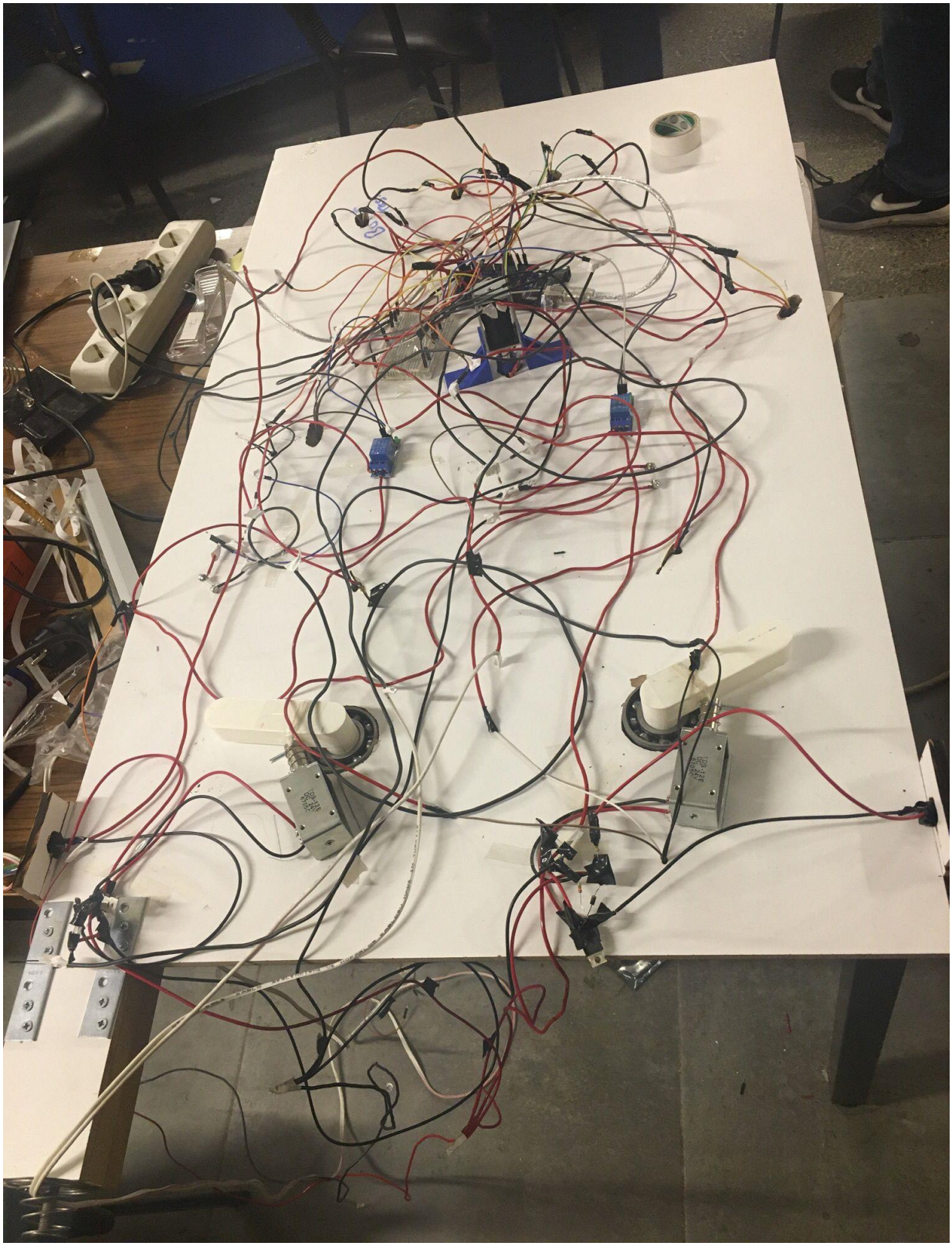

Figure 35: The Electrical Product of Pinball Machine(back side of the machine)

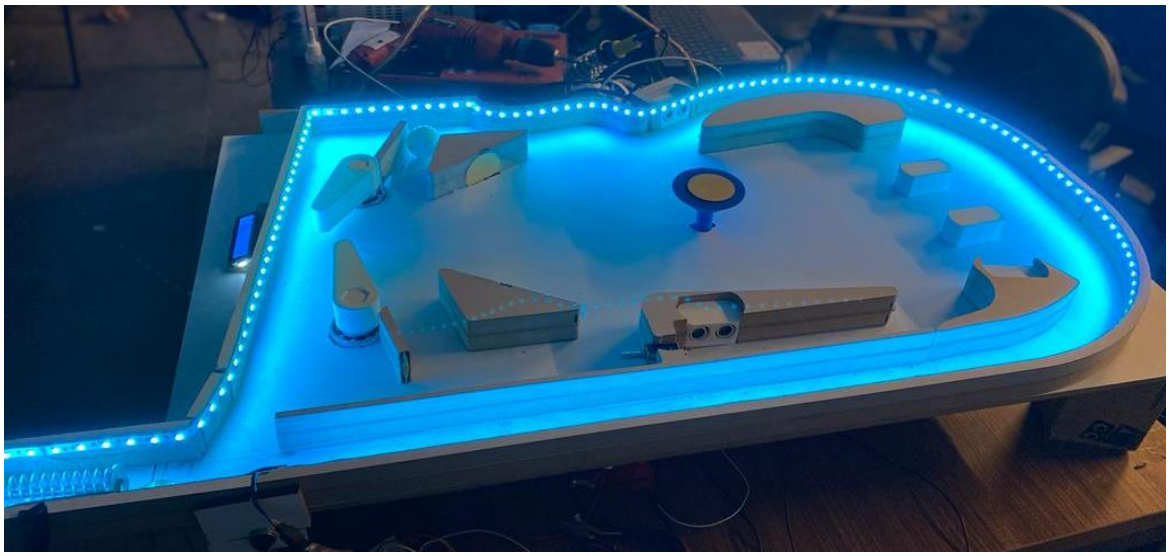

Figure 36: The Pinball Machine with Blue Lightning

Figure 37 – The LCD screen during game

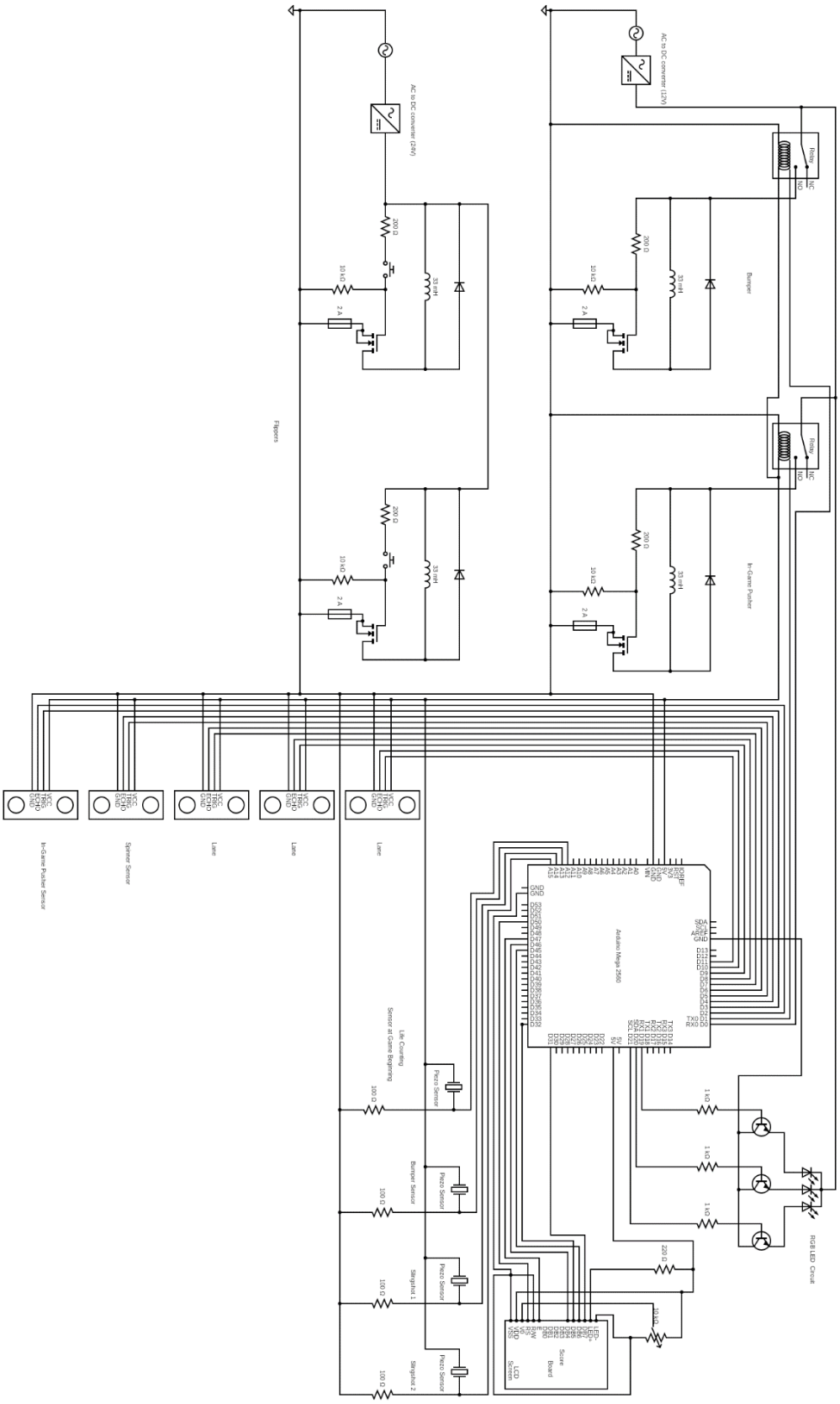

Figure 38– The Circuit Diagram (in bigger scale)

7 WORK DISTRIBUTION

Work distribution among group members is displayed below in Table 1: Work Distribution Table.

WORK DISTRUBITION TABLE

PROJECT NAME	PINBALL MACHINE	COURSE	
PROJECT GROUP	PG7	YEAR / SEMESTER	

STAGE		DETAILED WORK	MECHANICAL TEAM		ELECTRICAL TEAM		SOFTWARE TEAM	
			Tolga Çatak	Ümit Henkoğlu	İbrahim İşsever	Yasela Mollaoğlu	Arda Ulutürk	Ömer Ercan
1	Research	- Microcontroller Selection					■	■
		- Mechanical Component Selection	■	■				
		- Actuator Selection	■	■	■	■	■	■
		- Sensor Selection	■	■	■	■		
		- Selection of Electrical Components			■	■	■	
2	Shopping	- Electrical Shopping Karaköy 1			■	■	■	
		- Electrical + Mechanical Shopping Karaköy 2		■	■	■		
		- Electrical Shopping Karaköy 3			■			■
		- Electrical Shopping Karaköy 4				■		
		- Online Electrical Shopping 1	■					
		- Online Electrical Shopping 2			■		■	■
		- Mechanical Shopping 1	■					
		- Mechanical Shopping 2	■	■				
		- Mechanical Shopping 3	■	■				
3	Mechanical	- SolidWorks Design	■	■				
		- Mechanical Design	■	■	■	■	■	■
		- Cutting and Forming Playground Parts	■	■				
		- Building Playground	■	■	■	■		
		- Connecting Mechanical Parts	■	■				
4	Electrical	- Circuit Design			■	■		
		- Building Sub-Curcuits			■	■		
		- Testing and Analyzing Sub-Circuits	■		■	■		■
		- Connecting Sub-Circuits			■	■		
		- Connecting Arduino	■		■	■	■	■
5	Software	- Writing Code Sections					■	
		- Uniting Code Sections						■
		- Debugging					■	■
		- Final Clean Code					■	■
6	Documentation	- Proposal Preparation	■	■	■	■	■	■
		- 1st Report Preparation	■	■	■	■	■	■
		- Presentation Slides Preparation	■	■	■	■	■	■
		- Final Report Preparation	■	■	■	■	■	■

Table 1: Work Distribution Table

8 CONCLUSION

In this final report of the ME331 2021 Fall course term project, we have covered electrical, mechanical and software design of our “Pinball Machine” Project. We have displayed our circuit diagram, pseudocode flowchart and mechanical 3D design of some components of the machine. We also have added some figures of the final product to the report. Additionally, we have explained the working principle of each part thoroughly. Then, we have listed all components that are used in the project. Finally, we have displayed work distribution among group members during the project.