Name: Karthik Raja

Project 03 – Hoops Project

Design Overview:

High-Level Design Overview:

The design for my project will involve implementing a mini-basketball arcade game that would incorporate different game modes that the player(s) would be able to choose from. The purpose of my design is to allow players to be able to play different basketball games on a door mount basketball hoop. When I was a little kid, I would often play basketball on an indoor hoop and try to play different games and try to keep the score and time in my head. However, with this design, players can relax and not worry about keeping track of the score or time and only focus on trying to make a basket. The game will be built around a pre-built door mount basketball hoop. The design will be revolved around the ATMega328P chip, which is the chip used in Arduino Uno. The ATMega328P chip will be in charge of many different jobs, such as indicating the total points, indicating a make via a green LED, indicating game being over with a red LED, allowing the users to select among multiple game modes via buttons, outputting sound to the player in order to make the experience better, and indicate a good bucker via an ultrasonic sensor. The entire system will be powered by a 9V battery, which is stepped down to 5V by a 7805-voltage regulator, since all components have a 5V operating voltage level. I will be using a seven-segment display in order to display the total points for the player. The seven-segment display is an I2C module, which means that only four pins are required to power and operate the display. I will be using an LCD display in order to display the different available game modes, and the time remaining in those game modes as well. Also, I will be using a DFPlayer Mini module in order to output sounds to the two speakers I will be using to allow the players to be able to hear the sounds regardless of which direction they are in. Furthermore, a 16MHz oscillator and two 22pF capacitors are required in order to operate the ATMega328P chip as intended. I will be using green and red LEDs in order to detect a made basket and the game being over, respectively. There will be 100-ohm resistors for each of the LEDs to avoid short circuits. I will be using a left and right arrow buttons and a select button in order to allow the user to select and play their desired game mode. The reset button in the system will allow the player to return back to the game mode selection screen while they are currently in the middle of a game. As mentioned above, an ultrasonic sensor will be used in order to detect a basketball going through the hoop and it will only detect basketballs at most 15cm away from the sensor. Finally, I will using

screw terminals to help jump wires from the PCB to the components for the speakers, LEDs, buttons, sensor, and the 9V battery as well.

The different game modes I will have are the 30 second drill, 60 second drill, 3-point shootout, and Horse. The 30 second and 60 second drills are both game modes where the player has a set amount of time (30 seconds or 60 seconds) in order to try and get as many points as they can. For the 30 second drill, the player will be awarded with extra 10 seconds of time if they score more than 24 points, while, for the 60 second drill, the player will be awarded with extra 10 seconds of time if they score more than 50 points. For the last 7 seconds of the 30 second drill, the player will get 3 points for each made bucket, compared to the default 2 points per made bucket. For the last 15 seconds of the 60 second drill, the player will get 3 points for each made bucket, compared to the default 2 points per made bucket. Furthermore, the 3-point shootout game mode will allow the player to shoot as many 3-point shots as they can in 45 seconds. There is no extra time or point incentives in this game mode; it is simply to see how many 3-pointers the player can get in 45 seconds. Finally, the last game mode for the system is Horse. The logic works the exact same as normal Horse except for the fact that a missed shot is indicated as a timeout from the time limit of each shot. Since they are no way to detect a missed shot in the system, I had it so that a missed shot will be detected by a shot not going in within the time range of 15 seconds. But the rest of the logic is the same as normal Horse, where both players alternate taking shots, and if the current player misses, with the previous shot went in, then the corresponding player will have a letter added to their game. The first player to spell out HORSE will be the loser of the game. Hence, these are the four different game modes I have implemented in the system. For future iterations of this design, I plan to add a mode where the user can shoot for as long as they would and increment the score accordingly, and there will be no time limits in the game mode.

Name: Karthik Raja

Step 1: Circuit Schematic

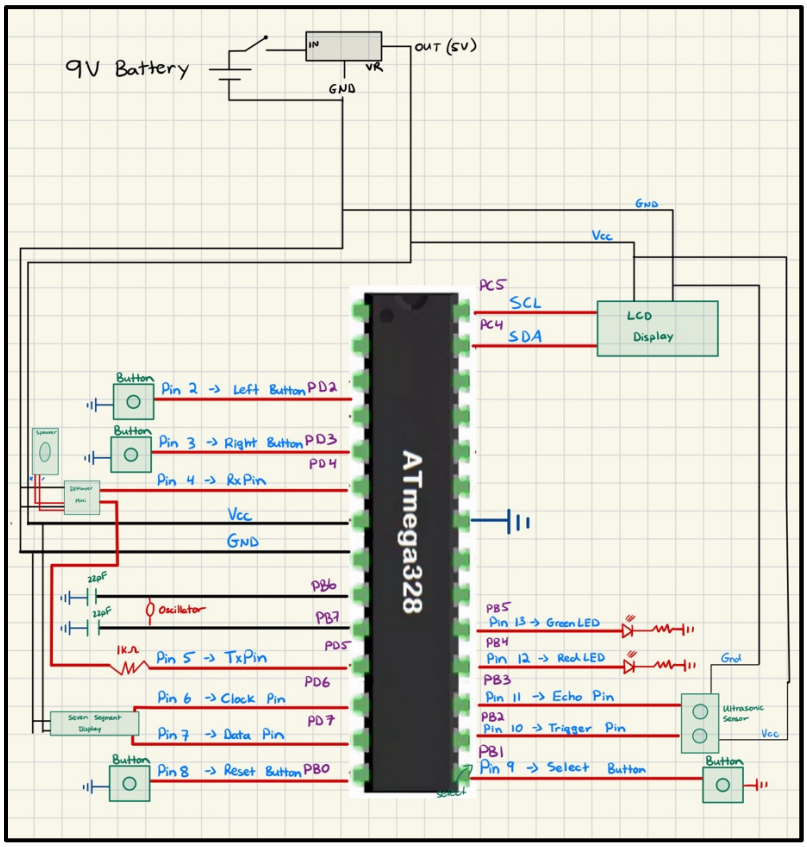


Figure [1]: Project 03 Schematic

The screenshot above represents the circuit level schematic for my Project. The schematic involves all of the components mentioned above: ATMega328P chip, seven-segment display, LCD display, ultrasonic sensor, left/right buttons, select button, reset button, DFPlayer Mini, and two speakers. There is also a 7805-voltage regulator having 9V as its input in order to step down the voltage from 9V to 5V for all of the components to function as expected. The schematic also involved three resistors, a 16MHz oscillator, and two 22pF capacitors.

The ATMega328P chip will act as the main logic controller for the game and will take in multiple inputs and put out multiple outputs depending on the scenario. I used the Arduino IDE to code all of the software for this design. The software of the project will be explained in detail later. The seven-segment display will display the total number of points the user as scored depending on the game mode, as explained above. The LCD display allows the player to select between different game modes, display the time remaining in the selected game mode, and also, if the game

mode selected is HORSE, then the display will help the two players play the game in an organized manner with instructions. The ultrasonic sensor will be used in order to detect a made bucket by the player. The software logic for the sensor was that if the sensor detected a basketball within 15cm of the sensor, then register that shot has a made bucket. This distance was found via trial and error to make sure that the sensor picks up a made shot from anywhere with the circumference of the rim and nothing outside of it. The left and right buttons are for the player to navigate through the game mode selection screen. The select button is for the player to select their desired game mode they would like to play. These buttons are not being polled during the actual gameplay, so if the ball accidentally hits the buttons, nothing will be triggered and the game will continue normally. The reset button allows the players to go back to the game mode selection screen while they are in the middle of a game mode, so that they do not have to wait until the game mode is over in order to go back to the game mode selection screen. The DFPlayer Mini module is used to output sound to two different speakers in order to allow the players in have a better experience. The module was output a buzzer sound whenever the game mode has started and another one when the game mode has finished. It will also output the infamous Mike Breen "BANG" sound every time the player has made a basket. Finally, the module was output little ticking sounds for the five second countdown between the game mode selection and the start of the game. The purpose of the voltage regulator is to step down the 9V input voltage down to 5V for all of the components explained above. The purpose of the two of the three resistors is to limit current go into the LED and the last resistor is placed between the TX pin of the ATMega and the RX pin of the DFPlayer Mini because the DFPlayer Mini's RX pin is designed to operate in 3.3V, instead of 5V. Finally, the 16MHz oscillator and the two 22pF capacitors are used to allow the ATMega chip have an external clock, which helps synchronized its components with precise timing and helping accurate execution of the software program loaded on the chip.

My original proposal for my project involved making a mini size basketball arcade game; however, the proposal consisted of a 3-D printed basketball hoop, break-beam sensors, joystick module, seven-segment display, passive buzzer, LCD display, and a reset button. The break-beam sensors would have been put on the rim on the model and would have detected if there has been a ball that has got past the rim, which indicates that the user shot it in the hoop successfully. The seven-segment display is to indicate the number of points the user scored in the time duration. This implementation is still in my final product as well. The LCD display is to help the user select the mode they would like to play in (30 sec or 60 sec). This implementation is still in my final product as well, but I added two game modes to the initial proposal: Horse and 3-point shootout. The passive buzzer would have been used to indicate a

made bucket and the time has ran out. The joystick would have been used to navigate through the different game modes (displayed on the LCD screen) and select the desired game mode. I decided to divert away from the initial proposal explained above due to the fact that I realized that having a small desk size 3-D printed basketball hoop would not have been fun compared to a much bigger indoor basketball hoop, where the player can stand up, run around, and shoot like they normally would. Since, I have always loved to play the basketball arcade games and figured that building one myself will be fun, I decided to stick with a basketball-themed project, but with a different hoop. As explained above, my final design also had a seven segment display and an LCD display, similar to my initial proposal; but the rest of the components used are not the same. The detailed design concepts of my final design are explained above.

Schedule (Apr 1st – May 3rd):

Week 1:

1. Prototyping of all the integrated sensors
2. Start working on the PCB
3. Start making initial 3D model designs

Week 2:

1. Finish working on the PCB and submit it to Dr. Dickerson to order it
2. Continue to work on software logic
3. Start printing 3D model designs

Week 3:

1. Receive PCB and start soldering to components together
2. Finish software logic
3. Finish printing all 3D model designs

Week 4:

1. At this point I should have all of the components I need 3D printed and the PCB
2. Assemble different parts of the 3D model together and put assembled PCD into the enclosure

This is the schedule I used in order to complete this final project. This final project only used one pre-existed components, which was the pre-built indoor basketball hoop I used. This pre-built basketball hoop was

found and bought on Amazon. Everything else in this project was not built on previous works, and following this schedule helped me to complete the project on time with everything working as intended.

Preliminary Design Verification:

Hardware Verification - Breadboard Prototype

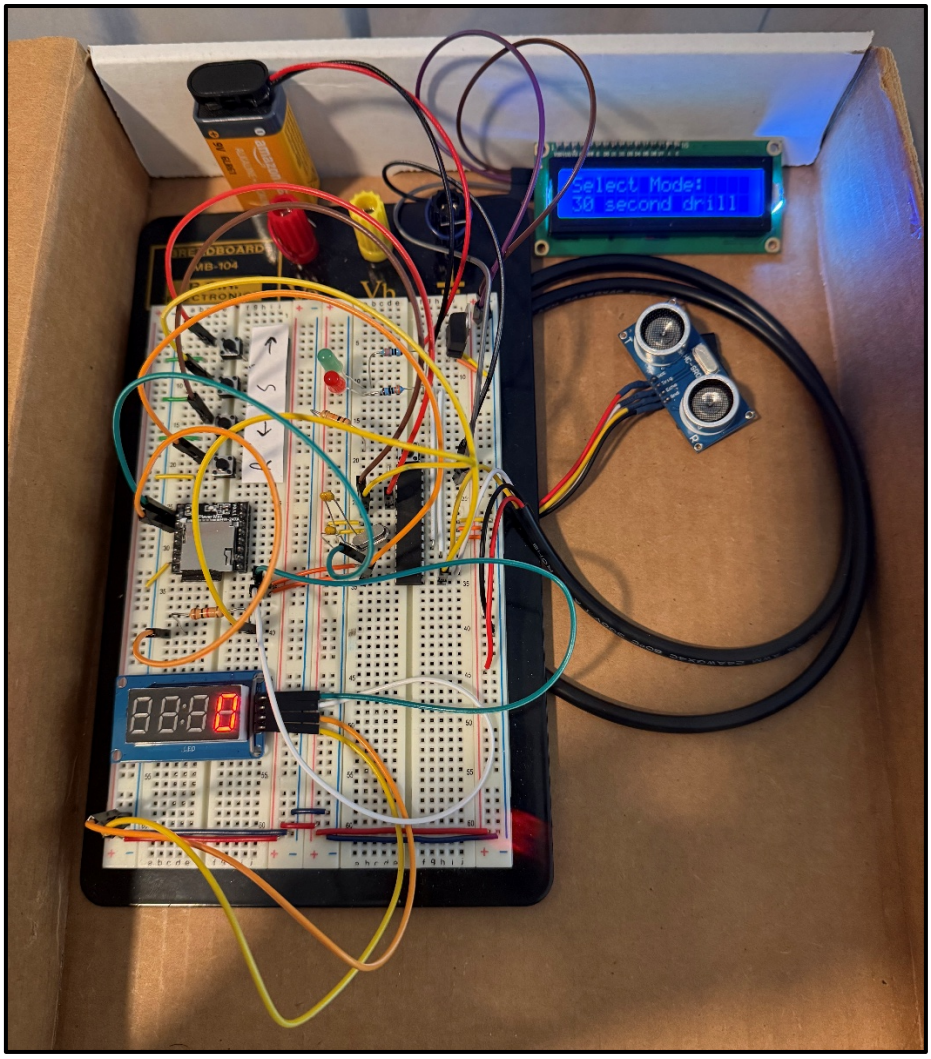


Figure [2]: Breadboard Prototype of Design

As seen in the screenshot above, this is my prototype of my design where I put my circuit schematic into a breadboard. I decided to verify all parts of my design on my breadboard to make sure all of the parts work as intended

before I design a PCB for this design. I did not want to design a PCB for a design that was not fully prototypes on a breadboard, since that could lead to certain parts of the design not working as intended. The schematic that I based my breadboard prototype off of was provided earlier in Figure 2.

In the left side of my breadboard, I implemented all of the different buttons, the DFPlayer Mini module, and the seven-segment display as well. For the buttons I will be using, I labeled them to make it easier for debugging purposes. The button labeled “R” represents the reset button, the button labeled "S” represents the select button, and the button labeled with the left and right arrows represent the left and right buttons. Regarding the DFPlayer Mini module, I used the diagram I found online to connect the module in the desired manner to play MP3 files. Lastly, I used a seven-segment display that only required four pins to operate, where only two are signal pins. Using this component was easier than using a seven-segment display with 10 pin for each of the segment on the display and it helped reduce the number of total pins needed as well.

In the right side of my breadboard, I implemented the ATMega chip, LCD display, ultrasonic sensor, and the two LEDs I will be using as well. I decided to use the I2C module of the LCD display to avoid using 16 different pins just for one component. As mentioned above, the ultrasonic sensor uses echolocation to detect an object in front of it. They emit high-frequency sound waves and measure the time it takes for the echo to return after bouncing off an object. This time between the emit and return is used to find the distance to the object. The two LEDs I will be using are to detect a made basket and the game being over. The green LED will be used to detect a made bucket, while the red LED will be used to detect that the current game mode is over. The ATMega chip has jumper wires going in and out of it to the respective components in order for the logic of the software loaded onto the chip to work as desired. I used a separate breadboard to help burn the bootloader and load the program onto the chip. I had some issues using the final breadboard as seen above to burn the bootloader, so I decided to make a separate breadboard just for uploading the code onto the chip. I believe the issue on the final breadboard was that I had component pins in the slots where there were bootloader pins, such as the MISO and MOSI pins.

Initially, to test this prototype, I had multiple helper files to test each component individually and understand how they work, which will be explained later. After individually testing them, I started to integrate them into one file and started on my main logic for the program. After hours of debugging and trying to perfect the code, I powered on the circuit and played the game as if it would be played if it was mounted on the pre-built basketball hoop. I was able

to select the different game modes and I would place my hand in front of the ultrasonic sensor to detect the ball going through the basketball hoop. I had noticed that there was something wrong with my logic for the HORSE game mode. Whenever Player 2 makes a shot and Player 1 misses that shot, it would go back to Player 1 when it is supposed to go back to Player 2. I had to change my logic in a way such that it always alternates between the players and would detect a letter being added to the player if the previous shot was made or not. In the video below, I tested that the players can switch between game modes in the selection screen, the sensor detects a made bucket with my hand playing the role of the ball, players can press the reset button to go back to the game mode selection screen, and the LEDs work properly and light up when expected. Below you will find the video to the functionality of the breadboard prototype that proves that all of tests work.

Video of Final Breadboard Prototype Functionality:

Click on this Link to Access Video of Functionality

As seen by the video above, the functionality of the game works as such:

- The player must select the desired game mode they would like to play.
- The game would provide the players with a 5-second countdown before moving onto the starting the game.
- Depending on the game mode logic, the game will increment the total number of points if the ultrasonic sensor detects a made basket, which is indicated by the green LED lighting up.
- The player is able to press the reset button in the middle of a game mode in order to go to the main game mode selection screen.
- If the sensor does not detect a made basket, then it does nothing.
- The red LED will light up if either the reset button is pressed or if the game mode as ended.
- With these results, I was able to determine that my design was indeed feasible to implement and would work on a PCB.

Name: Karthik Raja

Software Verification – Arduino Code

Regarding the software aspect of the design, I decided to use helper functions for all of the individual sensors and then integrate them together into one file and then add logic around them in order to get the game fully functional.

Helper Files:

The helper files below will help us integrate all of the different sensors into the main file after testing them individually.

Ultrasonic Sensor Code:

```
void loop() {

  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  duration = pulseIn(ECHO_PIN, HIGH, 10000);

  distance = duration * 0.0344 / 2; // cm

  if (distance < 13 && distance > 0) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(1000);
    digitalWrite(LED_BUILTIN, LOW);
  }

  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  delay(50);
}
```

Figure [3]: Ultrasonic Sensor Test File

The screenshot above shows the code needed to operate the ultrasonic sensor by itself. The ultrasonic sensor uses echolocation in order to measure the distance of an object in front of it. I start off by sending the trigger from the sensor for a certain amount of time. Then, I measure how long it takes for the sensor to detect the trigger signal back. Moreover, I use the distance formula shown above in order to find how far the detected object was. The conditional below it is the conditional I am using to detect the basketball going through the rim.

Name: Karthik Raja

Audio Output Code:

```
#include "mp3tf16p.h"

MP3Player mp3(8, 9);

int randomInt = 0;
bool playedOnce = false;

void setup() {
  Serial.begin(9600);
  mp3.initialize();
}

void loop() {

  if (!playedOnce) {
    randomInt = random(1, 3);
    mp3.playTrackNumber(randomInt, 10);
    mp3.serialPrintStatus(MP3_ALL_MESSAGE);
    playedOnce = true;
  }

}
```

Figure [4]: DFPlayer Mini Test File

The screenshot above shows the code used to play a pre-recorded MP3 audio file via a DFPlayer Mini module and a MicroSD Card. Since there is three different commands in our design, the software will randomly pick a number from 1-3 and the corresponding audio will be played in the speaker that is connected to the DFPlayer Mini module. This code was obtained from a YouTube video I was watching in order to get a better understanding of the module and its connection with the speaker.

LCD Display Code:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
    lcd.init();
    lcd.backlight();
    lcd.setCursor(0, 0);

    for (int i = 0; i < 11; i++) {
        lcd.print(i);
        delay(1000);
        lcd.clear();
    }
}

void loop() {

}
```

Figure [5]: LCD Display Test File

Name: Karthik Raja

The screenshot above shows the code used to test the LCD display by itself. The LCD display should display numbers 0-10 with each iteration being one second apart. Printing a statement to the LCD display is a simple as doing "lcd.print("STATEMENT")."

Main Logic Code:

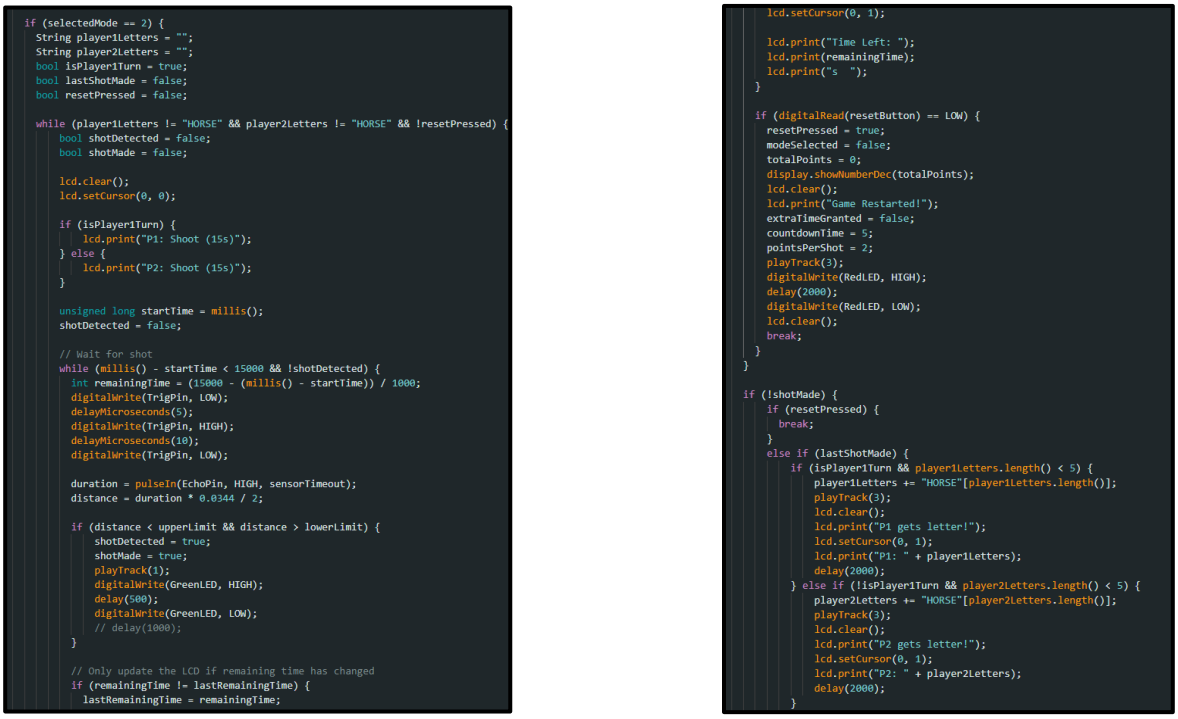
For the main logic file program, I implemented the different helper files into the main file and built the logic around those functions. Initially, I prototyped with my personal Arduino Uno dev board in order to be able to write to terminal and debug that way. After debugging was finished, I prototyped the software on my ATMega328P chip and double checked that everything works on that as well, since that is what I am using for my PCB.

```
// ********************************************************
// ***************** USER SELECTION MENU ******************
// ********************************************************

if (modeSelected == 0) {
  lcd.setCursor(0, 0);
  lcd.print("Select Mode:");

  bool rightPressed = digitalRead(RightArrowButton) == LOW;
  bool leftPressed = digitalRead(LeftArrowButton) == LOW;
  bool selectButtonPressed = digitalRead(selectButton) == LOW;

  // Right arrow logi
  if (rightPressed && !previousRightState && cursorPos < 3) {
    cursorPos++;
  }

  // Left arrow logic
  if (leftPressed && !previousLeftState && cursorPos > 0) {
    cursorPos--;
  }

  previousRightState = rightPressed;
  previousLeftState = leftPressed;

  lcd.setCursor(0, 1);
  lcd.print("                ");
  lcd.setCursor(0, 1);
  lcd.print(gameModes[cursorPos]);

  // Display Selected Mode to Player
  if (selectButtonPressed == 1) {
    modeSelected = true;
    selectedMode = cursorPos;
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Selected Mode:");
    lcd.setCursor(0, 1);
    lcd.print(gameModes[selectedMode]);
    delay(3000);
  }

  delay(20);

}
```

Figure [6]: Game Mode Selection Main File

The screenshot above shows the code used to allow the player to select which game mode they would like to play. The code initially the first game mode in the gameModes array, which is the 30-second drill mode. The left and right arrows essentially allow the players to move through the gameModes array and display the current index of the cursor to the player. When the select button is pressed, the system would take the game mode in the current cursor position index of the gameModes array and allow the player to play it.

Name: Karthik Raja



```
if (selectedMode == 2) {
  String player1Letters = "";
  String player2Letters = "";
  bool isPlayer1Turn = true;
  bool lastShotMade = false;
  bool resetPressed = false;

  while (player1Letters != "HORSE" && player2Letters != "HORSE" && !resetPressed) {
      bool shotDetected = false;
      bool shotMade = false;

      lcd.clear();
      lcd.setCursor(0, 0);

      if (isPlayer1Turn) {
          lcd.print("P1: Shoot (15s)");
      } else {
          lcd.print("P2: Shoot (15s)");
      }

      unsigned long startTime = millis();
      shotDetected = false;

      // Wait for shot
      while (millis() - startTime < 15000 && !shotDetected) {
        int remainingTime = (15000 - (millis() - startTime)) / 1000;
        digitalWrite(TrigPin, LOW);
        delayMicroseconds(5);
        digitalWrite(TrigPin, HIGH);
        delayMicroseconds(10);
        digitalWrite(TrigPin, LOW);

        duration = pulseIn(EchoPin, HIGH, sensorTimeout);
        distance = duration * 0.0344 / 2;

        if (distance < upperLimit && distance > lowerLimit) {
            shotDetected = true;
            shotMade = true;
            playTrack(1);
            digitalWrite(GreenLED, HIGH);
            delay(500);
            digitalWrite(GreenLED, LOW);
            // delay(1000);
        }

        // Only update the LCD if remaining time has changed
        if (remainingTime != lastRemainingTime) {
          lastRemainingTime = remainingTime;
          lcd.setCursor(0, 1);

          lcd.print("Time Left: ");
          lcd.print(remainingTime);
          lcd.print("s  ");
        }

        if (digitalRead(resetButton) == LOW) {
          resetPressed = true;
          modeSelected = false;
          totalPoints = 0;
          display.showNumberDec(totalPoints);
          lcd.clear();
          lcd.print("Game Restarted!");
          extraTimeGranted = false;
          countdownTime = 5;
          pointsPerShot = 2;
          playTrack(3);
          digitalWrite(RedLED, HIGH);
          delay(2000);
          digitalWrite(RedLED, LOW);
          lcd.clear();
          break;
        }
      }

      if (!shotMade) {
        if (resetPressed) {
          break;
        }
        else if (lastShotMade) {
            if (isPlayer1Turn && player1Letters.length() < 5) {
                player1Letters += "HORSE"[player1Letters.length()];
                playTrack(3);
                lcd.clear();
                lcd.print("P1 gets letter!");
                lcd.setCursor(0, 1);
                lcd.print("P1: " + player1Letters);
                delay(2000);
            } else if (!isPlayer1Turn && player2Letters.length() < 5) {
                player2Letters += "HORSE"[player2Letters.length()];
                playTrack(3);
                lcd.clear();
                lcd.print("P2 gets letter!");
                lcd.setCursor(0, 1);
                lcd.print("P2: " + player2Letters);
                delay(2000);
            }
```

Figure [7]: HORSE Main File

The screenshot above shows the code used to show how the logic for the HORSE game mode works. There is an encapsulating while loop that checks if a player has HORSE yet to see for a winner. Then, the logic essentially alternates between each player and allows 15 seconds for each player to shoot. If the current shot did not go in, then the logic checks to see if the previous shot was a made shot or not. If the previous shot was made and the current shot was missed, then a letter is added.

Name: Karthik Raja

```
else if (selectedMode == 0 || selectedMode == 1 || selectedMode == 3) {
// Game Loop
  while (millis() - gameStartTime < timeLimit) {
    int remainingTime = (timeLimit - (millis() - gameStartTime)) / 1000;

    // Calculate distance in real-time
    digitalWrite(TrigPin, LOW);
    delayMicroseconds(5);
    digitalWrite(TrigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(TrigPin, LOW);

    duration = pulseIn(EchoPin, HIGH, sensorTimeout);

    distance = duration * 0.0344 / 2;

    // Check if the ultrasonic sensor detects something within 14 cm away from it (Indicates Ball went through)
    if (distance < upperLimit && distance > lowerLimit) {
      playTrack(1); // BANG
      if (remainingTime <= 7 && selectedMode == 0) {
        pointsPerShot = 3;
      } else if (remainingTime <= 13 && selectedMode == 1) {
        pointsPerShot = 3;
      } else if (selectedMode == 0 || selectedMode == 1) {   // 30 second and 60 second drill
        pointsPerShot = 2;
      } else if (selectedMode == 3) {   // 3 Point drill
        pointsPerShot = 3;
      }

      totalPoints += pointsPerShot;
      digitalWrite(GreenLED, HIGH);
      delay(500);
      digitalWrite(GreenLED, LOW);
    }

    // Update the display with the total points
    display.showNumberDec(totalPoints);
```

Figure [8]: Rest of Game Modes Main File

The screenshot above shows the code used to show how the logic for the rest of the game modes works. The timeLimit variable is assigned above depending on the game mode selected by the user. Within the corresponding time limit, if there is a good bucket detected by the ultrasonic sensor, then the corresponding number of points is added to the total points and displayed in the seven-segment display.

Enclosure Verification – Pre-Built Basketball Hoop and 3D Prints

Regarding the enclosure aspect of the design, I decided to use Fusion 360 in order to build the PCB enclosure and the other parts that will make up the system.

Name: Karthik Raja

Figure [9]: Pre-Built Basketball Hoop

The image above shows the door mount basketball hoop that I bought to act as my hoop for the game. Initially, I planned on 3D printing a desk-size basketball hoop, but after some thought, I thought that would be too small and not as fun as being able to run around and play with a real-size indoor hoop. This was the only part of the design that was built off of previous designs, since it was bought off of Amazon.

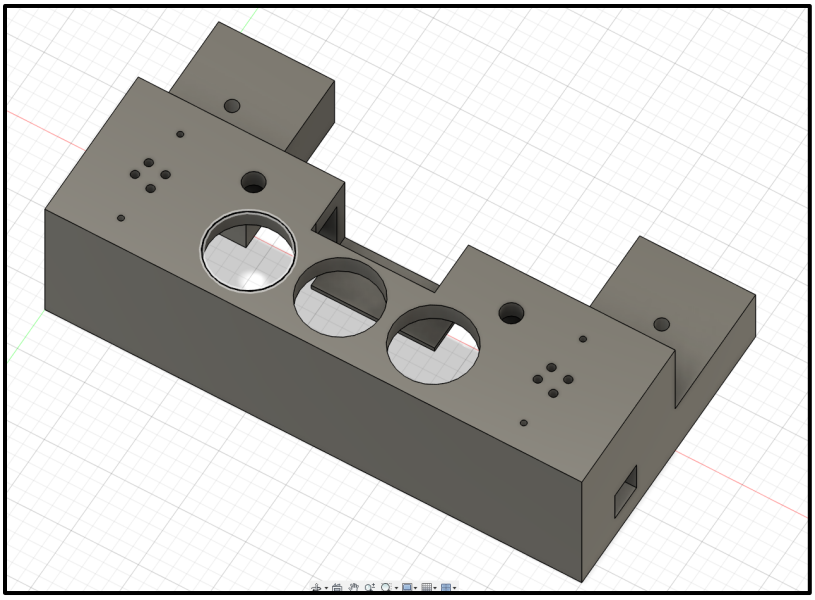


Figure [10]: Hoop Controller Enclosure

Name: Karthik Raja

The image above shows the enclosure I 3D printed in order to house all of the buttons, LEDs, and the speakers for the controller of the game. The three big holes in the middle of the enclosure will house the left, right, and select button. The two little holes right above those will house LED mounts, which will provide a cleaner look for the player. The little four circles on each end of the front will house the speakers and the sound will be heard through the four little holes. I did not want there to be a speaker exposed in the open to the player, so I decided to hide it in the back and allow the sound to go through the little holes. The two holes on the opposite of the front of the enclosure will be used to mount this controller onto the physical hoop. I will use M5 screws to make a secure connection between the hoop and the controller. The square hole on the right side of the enclosure will be for the reset button, while the hole on the left side of the controller will be for the on/off rocker switch.

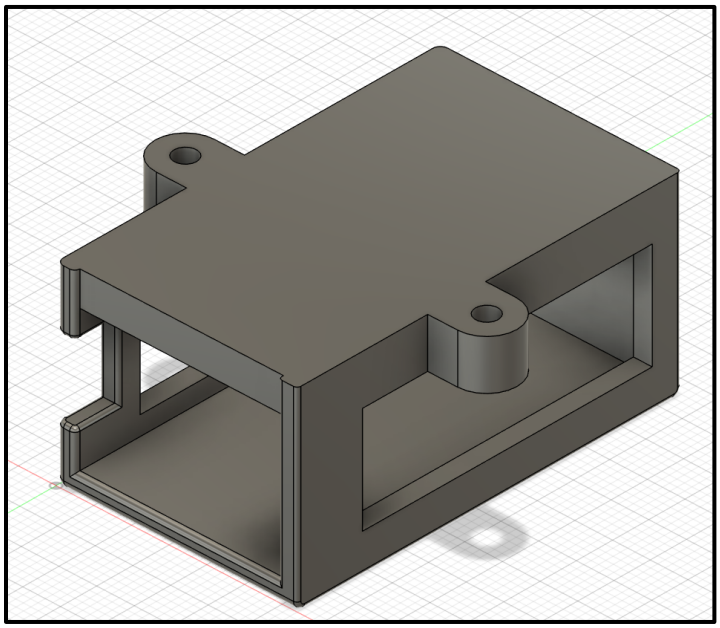


Figure [11]: 9V Battery Enclosure

I made a 3D model of a holder for the 9V battery that is needed to power on the circuit. I took the dimensions of the battery and made a hole for it to be slid into. I made a little passageway on the right side of the model in order for the wires connected to the battery clip to get out and connect to the 2- pin header in the PCB design. I plan on using the two holes on each side to mount this onto the physical hoop so that there is not a big battery on the top of my PCB and replacing the battery will be much easier. The rectangular holes on both sides of the holder are for the user to get the battery out for replacement with ease. Also, I plan to use Velcro that has a sticky side on both parts of it and attaching one side to the bottom of the battery and the other side to the deepest end of the model above. I decided to do this because I do not want the battery to slip out while users are playing the game.

Name: Karthik Raja

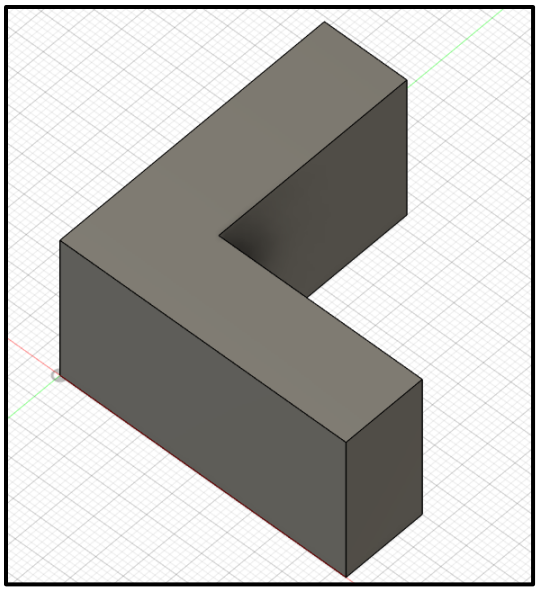


Figure [12]: Backboard Foam Spacer

I made a 3D model of a backboard spacer so that there is room to screw in the PCB enclosure to the back of the physical basketball hoop. I took the dimensions of the pre-existing foam pads on the hoop and made the spacer to same size. I super glued one side of this spacer to the pre-existing foam on the hoop and adding foam pads to the other side as well to prevent any damages the door that the basketball hoop will be resting on while players are playing the game.

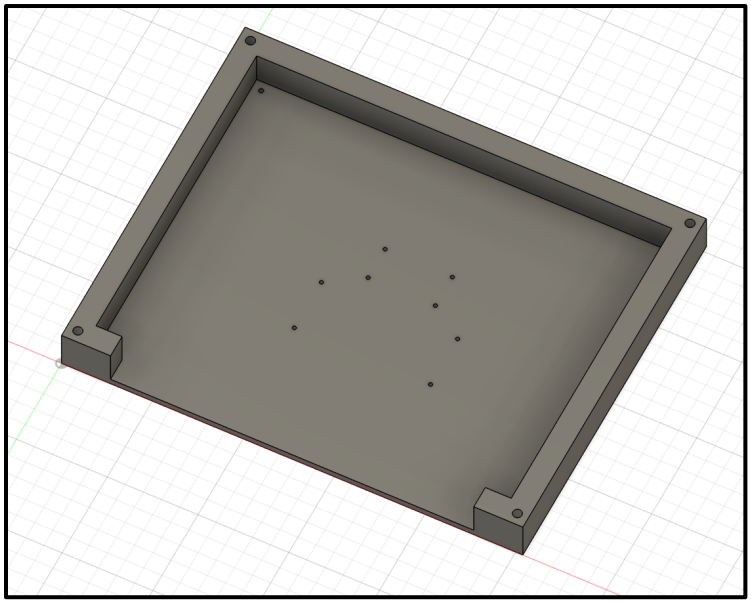


Figure [13]: Backboard PCB Enclosure

Name: Karthik Raja

I made a 3D model of a backboard PCB enclosure so that I can safely screw in my PCB into the holes on the enclosure to ensure a secure connection. In the PCB design, I added holes for all of the places I needed holes for. The holes in the four corners are for the PCB to be placed securely. The upper four holes that make up a rectangle are used to screw in the seven-segment display onto the PCB, while the lower four holes that make up a rectangle are used to screw in the LCD display onto the PCB. These screws will make sure that both displays are secured onto the enclosure as well as onto the PCB. The four corner holes on top of platform will be used to connect the enclosure with its lid and secure both of those onto the physical basketball hoop.

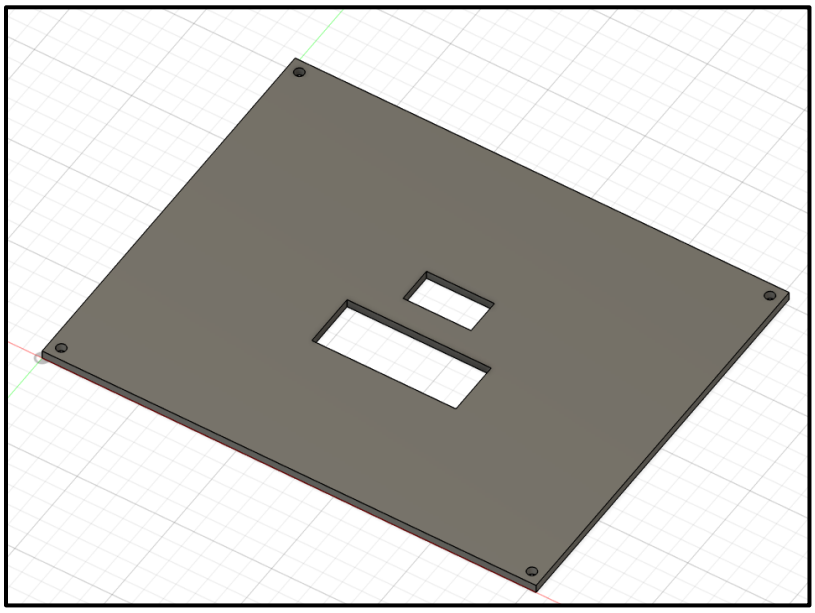


Figure [14]: Backboard PCB Enclosure

I made a 3D model of a backboard PCB enclosure lid so that all the wires and connections on my PCB are not revealed to the users. The only components that will be revealed to the players will be the seven-segment display and the LCD display, and both of those components will be seen through the holes made in the middle for them. The four corner holes will be used to connect the lid with the enclosure and secure both of those onto the physical basketball hoop.

Name: Karthik Raja

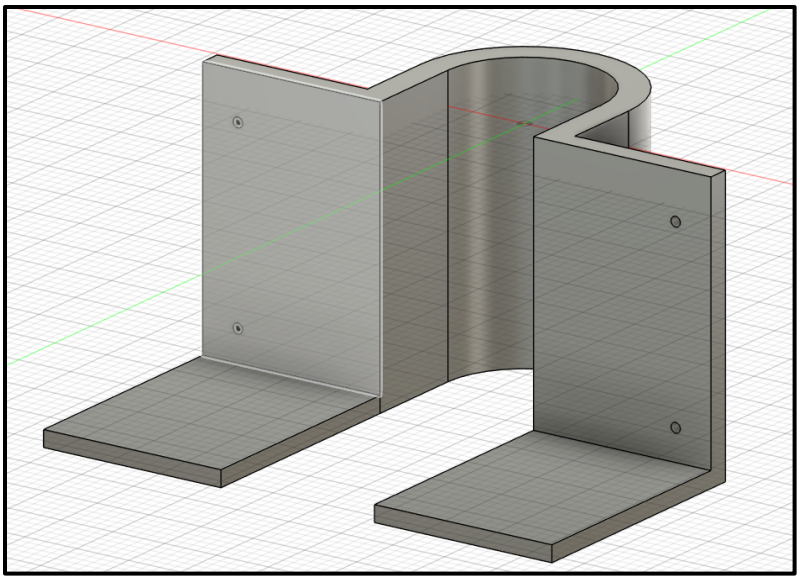


Figure [15]: U-Clamp for Ultrasonic Sensor

I made a 3D model of a u-clamp for the ultrasonic sensor, which will be housed around the spring in the bottom of the rim. The four holes are for the ultrasonic sensor and screwing the sensor into the clamp to ensure a secure fit. The platform that extrudes out of the face of the clamp is for the clamp to be securely glued onto the surface below the spring, which prevents the sensor to be moved at all.

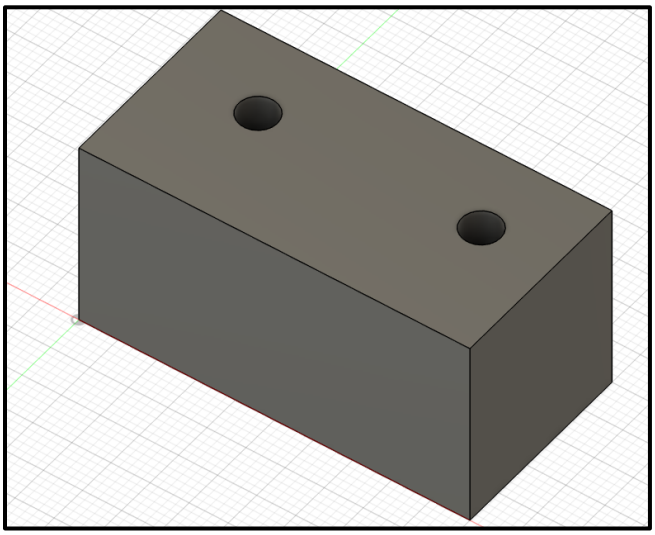


Figure [16]: Hook Spacer

I made a 3D model of a spacer for the two hooks of the physical basketball hoop so that the PCB enclosure has enough room to be housed and still be able to placed on top of doors to play. These spacers serve the same purpose

as the foam spacers mentioned above; they are only there to extend the space between the backboard and the door that it will be rested on in order to allow the PCB enclosure to be placed on the backboard.

Design Implementation

PCB Schematic

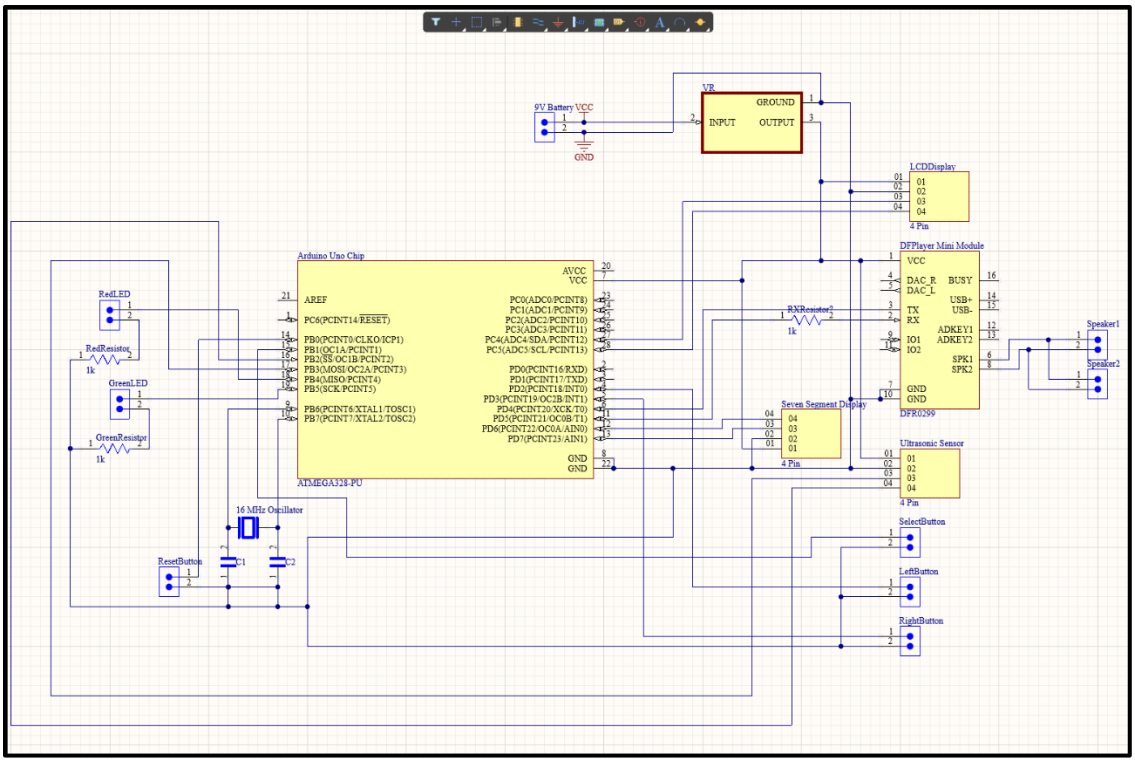


Figure [17]: Project 03 Schematic in Altium

The figure above shows the schematic for the project, which is similar to the initial proposed schematic in Figure 1. There were no changes made to the Schematic shown above from Altium compared to the initial schematic. I used screw blocks for all of the components except for the ATMega chip, seven-segment display, LCD display, the DFPlayer Mini module, the oscillator, the resistors, the capacitors, and the voltage regulator. Every other component in the schematic above uses a pin headers footprint, since I will be using screw blocks for easy wire placement. From this I plan on making a 3D model of a 9V battery holder that will be attached to the side of the PCB to hold the battery, so that there is not a big battery dangling from the PCB. This will make the product easier to manage, since the battery

has a little slot it can go in. I had to use a 2-pin screw terminal to power the circuit, since I am designing a PCB and I was unable to stick wires in a PCB for power like I can for a breadboard design. I used all THT components to assemble our schematic, which included the ATMega328P chip, the DFPlayer Mini, the voltage regulator, the oscillator, resistors, capacitors, 3 resistors, and 10 screw terminals. I used dip sockets to place the ATMega chip and the DFPlayer Mini onto the PCB, so that taking it out will be much easier and desoldering it won't be necessary. Since my PCB was going to be a THT board, I had to use THT components to create the schematic as well. As mentioned above, the initial schematic having the pin assignments already on it helped us create the schematic and connect the corresponding wires. The oscillators and capacitors on the ATMega chip are used in order to generate and stabilize the clock signal for its operation. The purpose for all of these components and why they are used are mentioned above in the design implementation explanation section.

Name: Karthik Raja

PCB Design Layout

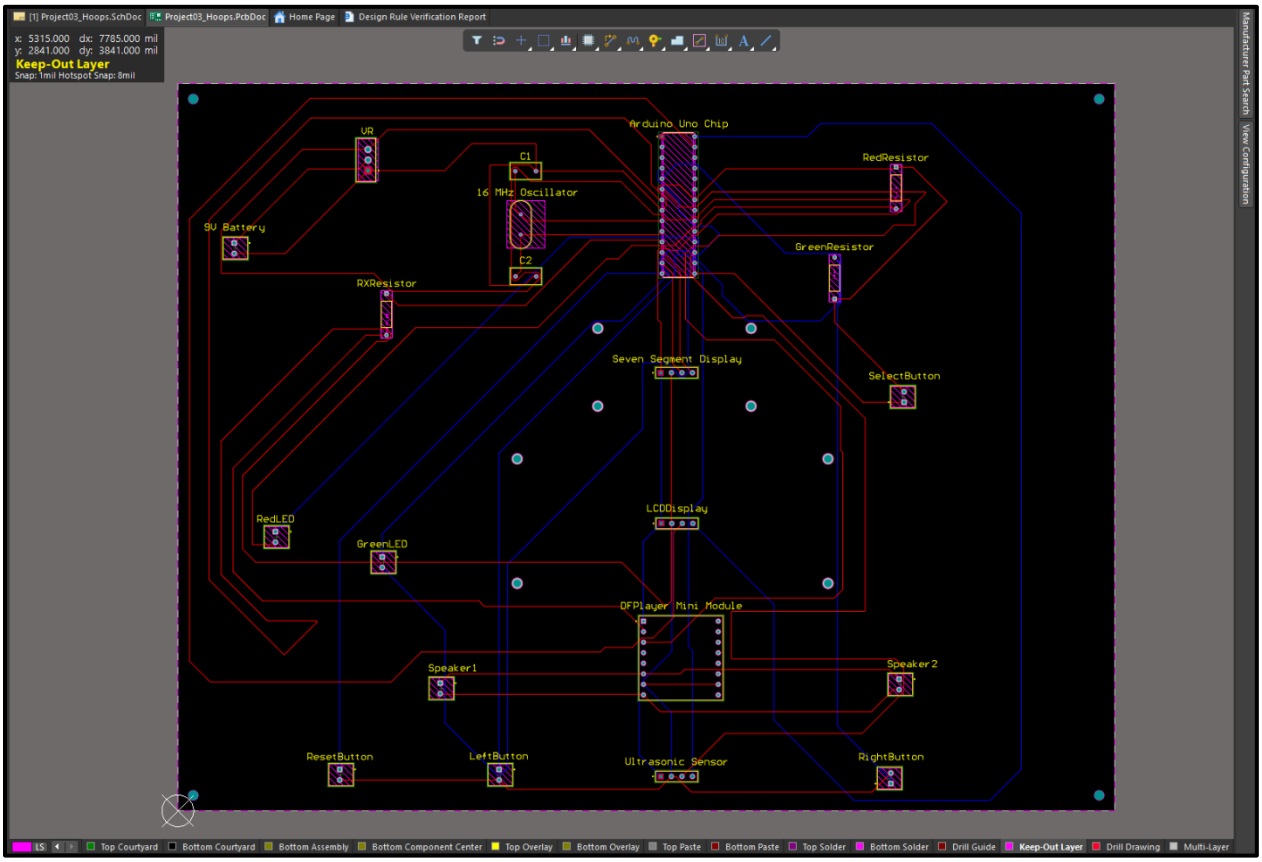


Figure [18]: 2-D View of Altium PCB Design

The figure above represents my PCB layout for the design I am implementing. I used all THT components to assemble my schematic, which included the ATMega328P chip, the DFPlayer Mini, the voltage regulator, the oscillator, resistors, capacitors, 3 resistors, and 10 screw terminals. Since I am using THT components, there is going to have vias in my PCB, which allows for components to go through the PCB and makes soldering easier. In the middle of the PCB, I decided to have the seven-segment display and the LCD display, since both of those components are going to be visible to the players. As mentioned above, I made holes for both of those components so that I can screw the components onto the PCB, which is screwed into the PCB enclosure. So, I decided to put the ATMega chip right on top of the displays and the DFPlayer Mini module on the bottom of the displays. Since, I am going to have the 9V battery holder 3D model to the left of the PCB, I decided to have the 9V battery input in the left side of the PCB and the voltage regulator right after that. Finally, I spread out the remaining components' screw blocks far apart,

since I knew that I would have a lot of wires going in and out of the PCB in order to connect the component to the screw blocks. Thus, I gave myself room to keep my wires organized. The main thing I was worried about was placing screw blocks for all of our components and just jumping wires out of the PCB and connecting it to the component. In my PCB design, since I will be pulling out jumper wires from the PCB, I used the footprints of pin headers in order to make vias for the different components that is going to be connected to the PCB. My main goal was to get the traces between each component correct and worry about having vias for each pin of the components. So, the positions of the components are not the best, since the PCB will not be seen by anyone and will be inside of the enclosure with wires coming of it to attach to the different components. There were no challenges in the layout of the PCB, since, as mentioned above, the placement of the components was not the main concern because I am just jumping wires from it. However, I couldn't have the screw blocks too far apart, because I am still constrained by the size of my PCB enclosure. This PCB will allow all of the different input and output components to be communicating with the ATMega chip in order to make the game to operate as smooth as possible.

Best Practice Incorporated in PCB Design: The "3W" Rule was the best practice I chose to incorporate into my PCB design, which states that the distance between traces should be at least 3 times its trace width. For example, all of my trace widths are 10mill, which means that the distance between all traces in my PCB design should be at least 30 mill. Due to the hole of the ATMega chip, I am using, there are a couple of instances where the distance between the traces is 20 mill, but that only occurs a couple of times and having twice the distance of the trace width is still acceptable for my design, since it does not require high speed communication. As mentioned above, I positioned my components in such a way that will make connecting traces relatively easy.

Name: Karthik Raja

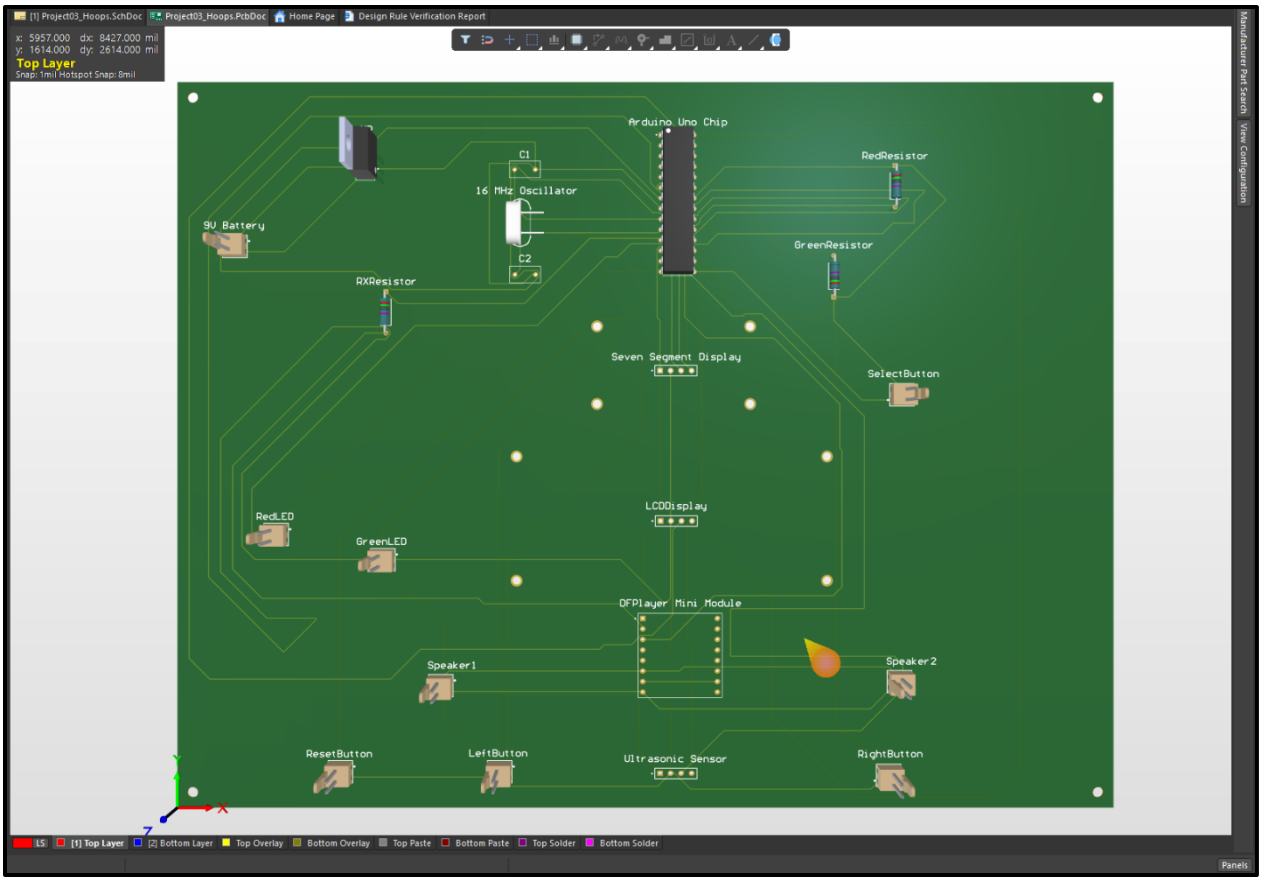


Figure [19]: 3-D View of Altium PCB Design

In the figure above, the PCB design is shown in a 3-D view with all of the components inserted as if they were already soldered onto the PCB. The layout of the PCB layout is the exact same as the layout in the 2-D view of it. The oscillator component on the left side of the PCB is not depicted properly due to the footprint I downloaded from the internet, but the size and number of holes match the chip. Since what matters most in PCB design is the holes being drilled, I am not worried about the chip being sideways in the 3-D view. The light green lines shown in the PCB are the traces in the PCB, which connect the whole circuit together.

Name: Karthik Raja

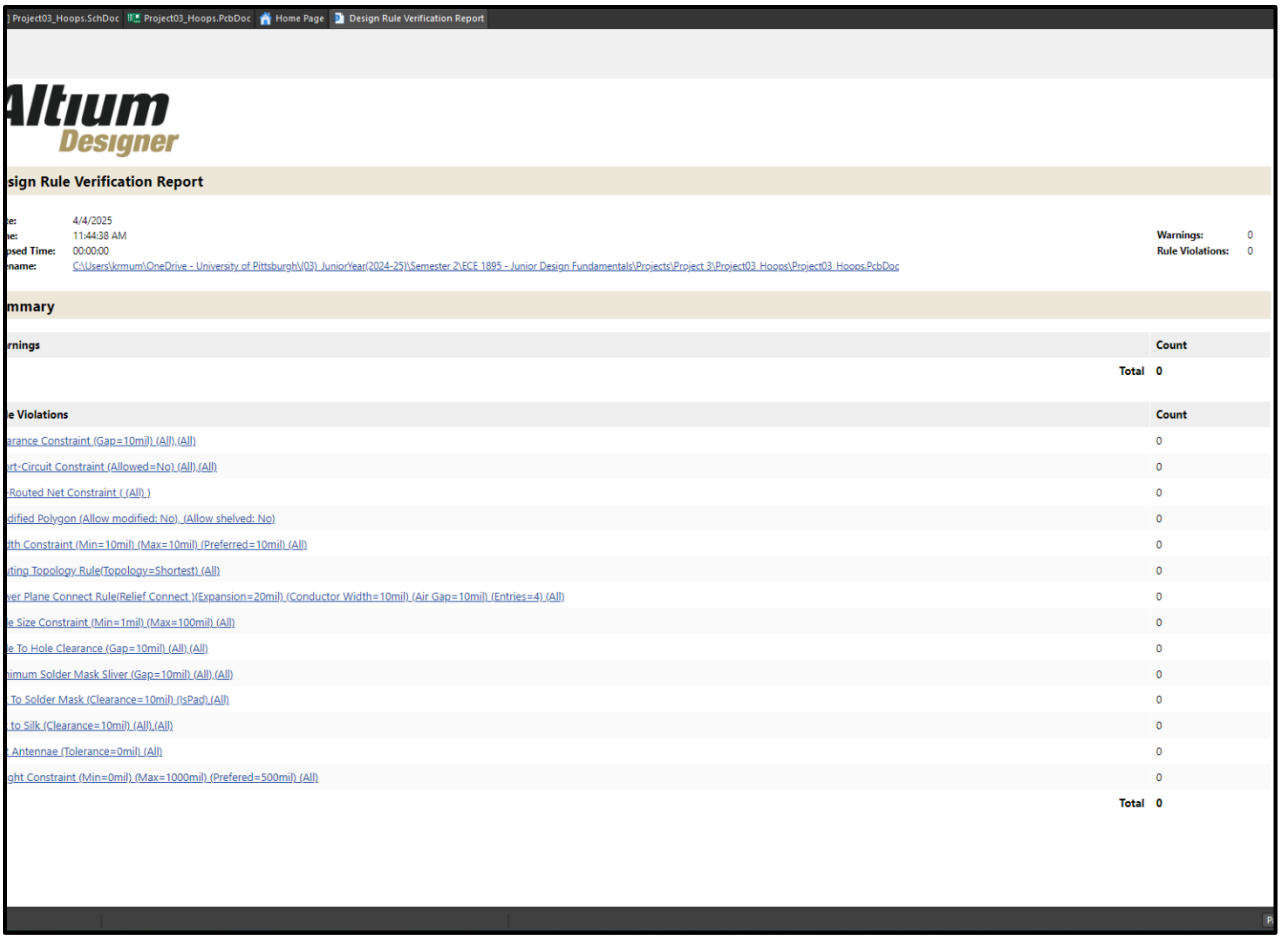
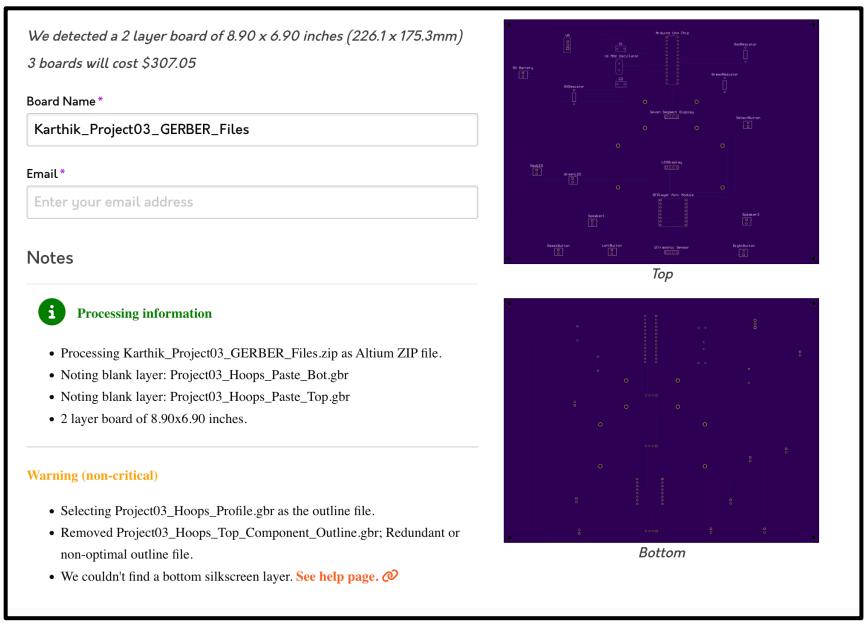


Figure [20]: DRC Check Results for PCB Design

After successfully arranging the components in the PCB design, I ran the Design Ruler Check with JLCPCB Rules file to make sure that I did not get any rule violations. Initially, I did get one error because I had overlapping text, but after I fixed that, I got an error count of 0, as seen in the screenshot above. This step was one of the most important steps of the PCB design process, since it checked for any errors that may prevent the printing of a PCB. Also, it is important to make sure that this step is taken after even the slightest change done to the PCB to ensure that the board can be printed.

Name: Karthik Raja



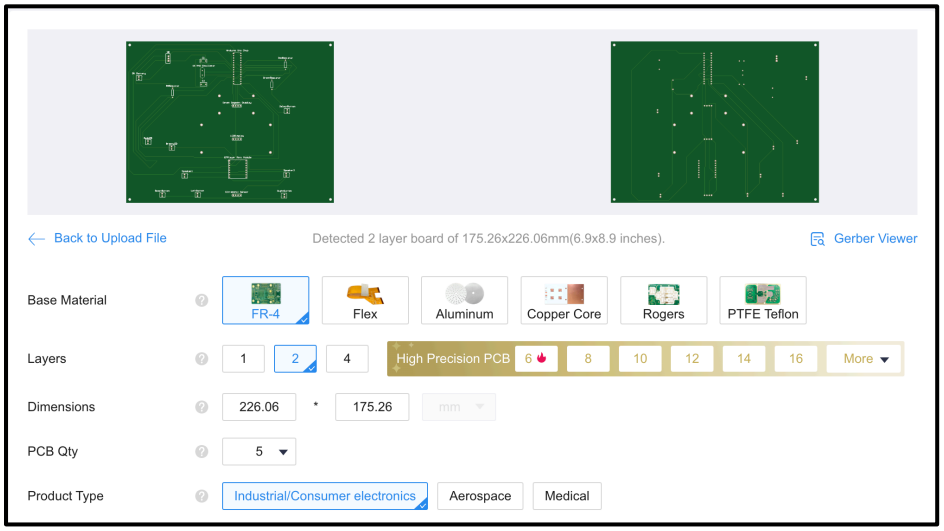


Figure [21/22]: Preview of PCB Design in Manufacturer Website

As seen in the figure above, my PCB design successfully gets read by oshpark, which verifies that my design is indeed manufacturable. I uploaded the compressed 'Project Output' folder into oshpark.com and JLCPCB and the screenshot above are the pages that resulted from it. You can clearly see the top and bottom of our PCB for this project along with the traces made in the top and bottom. The design of the top and bottom layer of the PCB is the exact same layout we had above in Altium Designer. The fact that we are able to see my PCB design when we uploaded the output folder to the websites indicates that our PCB can be manufactured by that manufacturer. Hence, after finishing this

step, in the PCB design process, you can go to a manufacturer's website, such as JLCPCB, to attach your Gerber file and order a PCB from them.

Design Manufacturing and Assembly

In my PCB assembly, all of the components were Through-Hole components and were clearly labeled in the PCB that was designed. Since the vias on my PCB were extremely small, I decided to use a skinny tip for my soldering iron in order to have enough room for the iron tip and the solder to make contact with the via. The first thing I did when my soldering iron heated up to the desired temperature of 375 °C was tinning the solder iron. The process of tinning the iron is feeding the tip of the iron with a chunk of solder. This process prevents oxidation and helps the solder to be melted easier. One trick that I used to help my components stay secure while flipping to the bottom of the PCB to solder was that I put a little bit of masking tape on top of the component. This causes the component to not move around or fall out of the vias when I flip the PCB over to solder. Using a clamp to hold my PCB while I was soldering was a suggestion I used. Having some extra 'helping hands' also helps with anything especially if the activity involves anything dangerous, such as a 375 °C solder iron. Throughout the assembly of the PCB, I did not encounter any major challenges, but I did, initially, find it difficult to solder into such a small via. Even though any solder would not go to any place other than the via due to the solder mask and silkscreen coating, it is still better to try and get it as close to the via as possible for better results. Furthermore, it was recommended that I use dip sockets for the ATMega chip and the DFPlayer Mini that I have in my design, so that if I have to replace the chip, then all I have to do is take it off of the socket and place another one in it. This results in replacing those components without any unsoldering and soldering involved, which saves a lot of little and does not pose any risk to the PCB. There were no modifications made to the design of the project for the purpose of testing. Although soldering does require some patience, I found that doing something that you like to do causes you to develop more patience for that activity. Overall, the process of assembling the PCB was relatively easy, since I had some previous experience with Through-Hole soldering.

The process of making the different subcomponents for the enclosure was explained in detail above as well as the process of implementing the software, including screenshots of the code as well.

Name: Karthik Raja

Design Testing

Images/Description of the Assembled PCB:

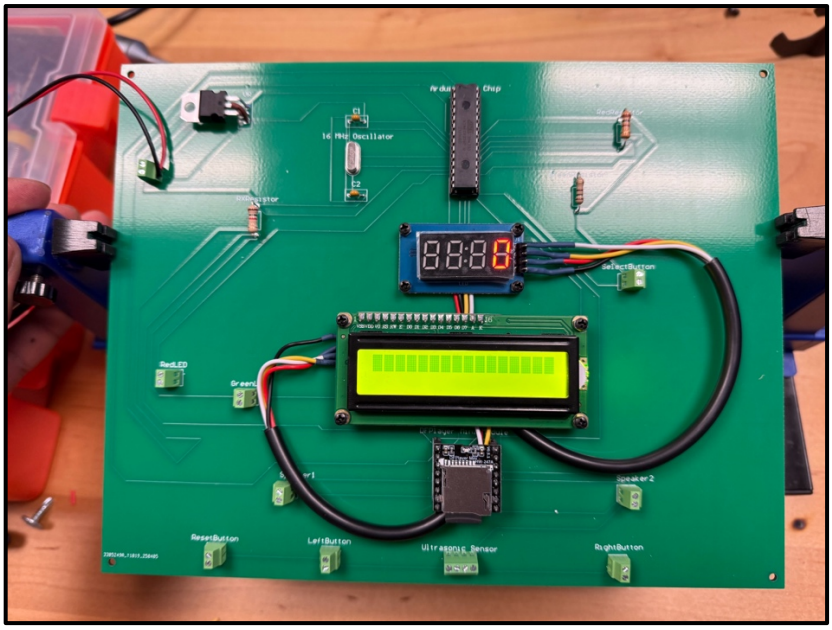


Figure [23]: Top View of Assembled PCB

The figure above shows the top view of the assembled PCB for Project 01. As mentioned above, I used all THT components to assemble my PCB, which included an ATMega328P chip, seven-segment display, LCD display, ultrasonic sensor, left/right buttons, select button, reset button, DFPlayer Mini, and two speakers. There is also a 7805-voltage regulator having 9V as its input in order to step down the voltage from 9V to 5V for all of the components to function as expected. The schematic also involved three resistors, a 16MHz oscillator, and two 22pF capacitors. In the middle portion of the PCB design, I organized the seven-segment display and the LCD display in such a way that it is visible to the user even with the enclosure lid I made. I decided to put the ATMega chip right on top of the displays and the DFPlayer Mini module on the bottom of the displays. Since, I am going to have the 9V battery holder 3D model to the left of the PCB, I decided to have the 9V battery input in the left side of the PCB and the voltage regulator right after that. Finally, I spread out the remaining components' screw blocks far apart, since I knew that I would have a lot of wires going in and out of the PCB in order to connect the component to the screw blocks. Thus, I gave myself room to keep my wires organized. I used 4 wire electrical cable in order to connect both displays shown above to the PCB.

Name: Karthik Raja

Best Practice Incorporated in PCB Design: The “3W” Rule was the best practice I chose to incorporate into my PCB design, which states that the distance between traces should be at least 3 times its trace width. For example, all of my trace widths are 10mill, which means that the distance between all traces in my PCB design should be at least 30 mill. Due to the distance between holes of the same chips, there are a couple of instances where the distance between the traces is 20 mill, but that occurred 2 times and having twice the distance of the trace width is still acceptable for my design, since it does not require high speed communication

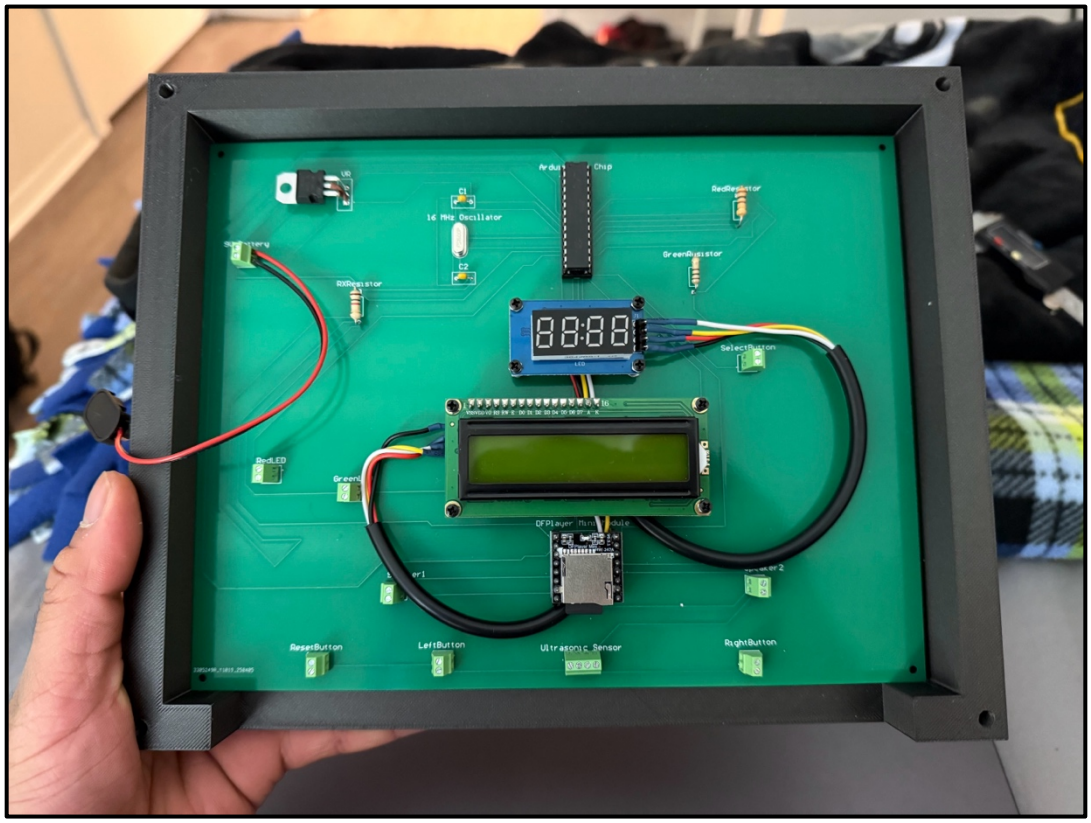


Figure [24]: Assembled PCB inside of Enclosure

The figure above shows the PCB soldered with all of the components inside of the PCB enclosure I had 3D printed. The seven-segment display and the LCD display are both seen to be screwed into the holes I made in the PCB for them and through the enclosure as well. The holes lined up perfectly and there was no need to reprint the enclosure. There was no need to reprint any 3D model made; everything came perfect on the first try.

Name: Karthik Raja

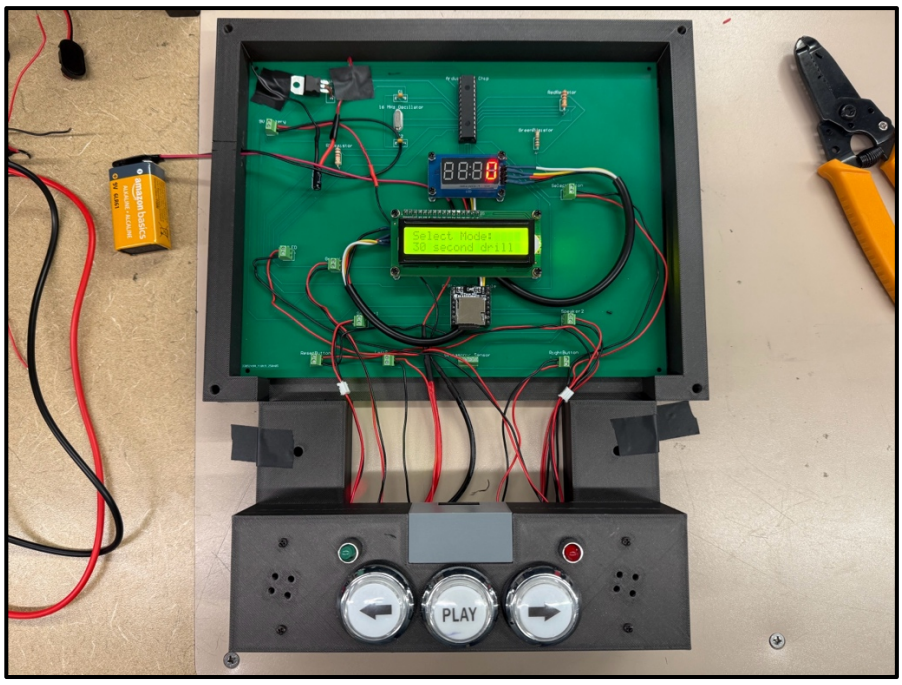


Figure [25]: Assembled PCB connected with all components

The figure above shows the system all connected together with all of the necessary connections coming out of the game controller into the corresponding screw blocks on the PCB. One of the ways I tested my connections was by using the continuity test on the multimeter. I would put the two multimeter probes on top of the screw blocks screws and then press the button and wait to see if I hear a beep, which indicates that continuity exists and it is working as expected. When I was testing the right arrow button, initially, there was no continuity on the input, and I realized that the spade connector I used for the button was not crimped properly. Also, one of the mistakes that I made during the PCB implementation was that I forgot to put a 10 uF capacitor between the output of the voltage regulator and ground. This caused issues for me, since I had to splice the 9V battery connector power wire in order to make it work with a rocker switch. The system would turn on if I directly connected the 9V battery to the battery connector, but would not turn on if I used the spliced battery connector. After some hours of debugging, I realized that I forgot to add the 10 uF capacitor between power and ground in my PCB and had to jump wires from the bottom of the PCB, as seen in the top right of the PCB, to add the 10 uF capacitor successfully. The last mistake I made regarding the PCB was that I was not aware that there was voltage regulators for negative voltages as well. So, I grabbed the first footprint

I could find for a voltage regulator not knowing that I grabbed one that was for negative voltage and the input and ground pins wee flipped. I did not know this and I fried my first PCB; thus, I had to grab one of the four remaining PCBs and resolder everything. The solution I used to fix this was I simply crossed the input and ground pins in the voltage regulator and put shrink tubes on those leads to prevent them from touching each other. This solution worked perfectly. The crossed leads voltage regulator can be seen clearly in Figure 24. To further test the rest of the PCB, I tried to replicate all of the test cases I made for the breadboard prototype, and it was all successful. The only thing I had left to test was the ultrasonic sensor distance detection, which will be explained later.

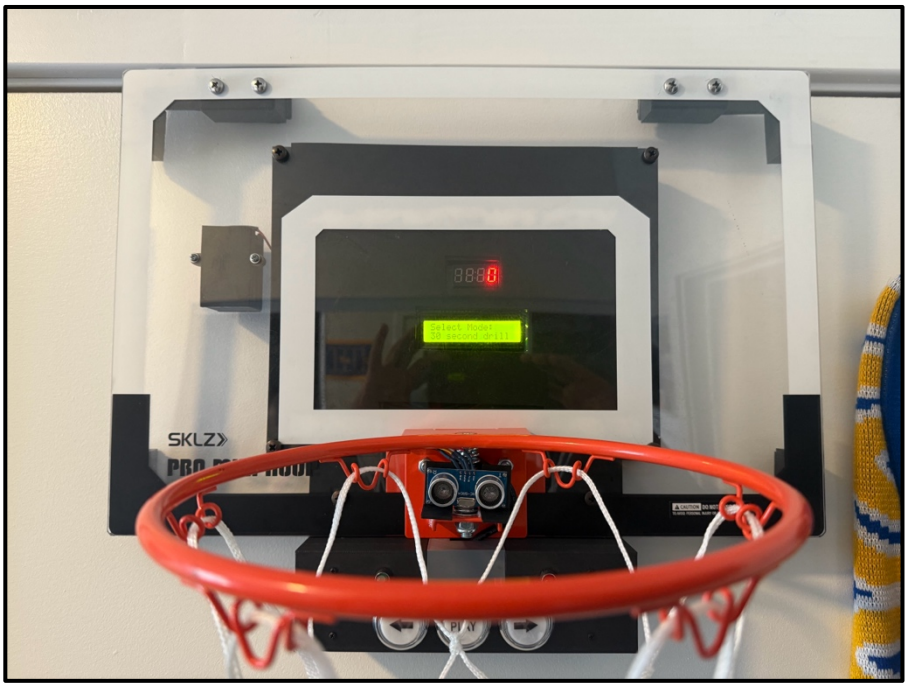


Figure [26]: Assembled PCB mounted behind Basketball Hoop

The figure above shows the fully assembled product with all of the 3D printed models mounted in the backboard and the system powered on. For all of the screws, I tapped all of them in so that they make natural threads for the screws to go into, which eliminates the need for any nuts on the other side of the screw. The 9V battery holder is also seen to the left of the main PCB enclosure, where there is a battery connector going from the PCB to the battery to power on the system. I had to drill a little hole for the battery connector wires to go into. I tried to place the seven-segment display and the LCD display as much in the center as I could for the player to be able see both of them.

Name: Karthik Raja

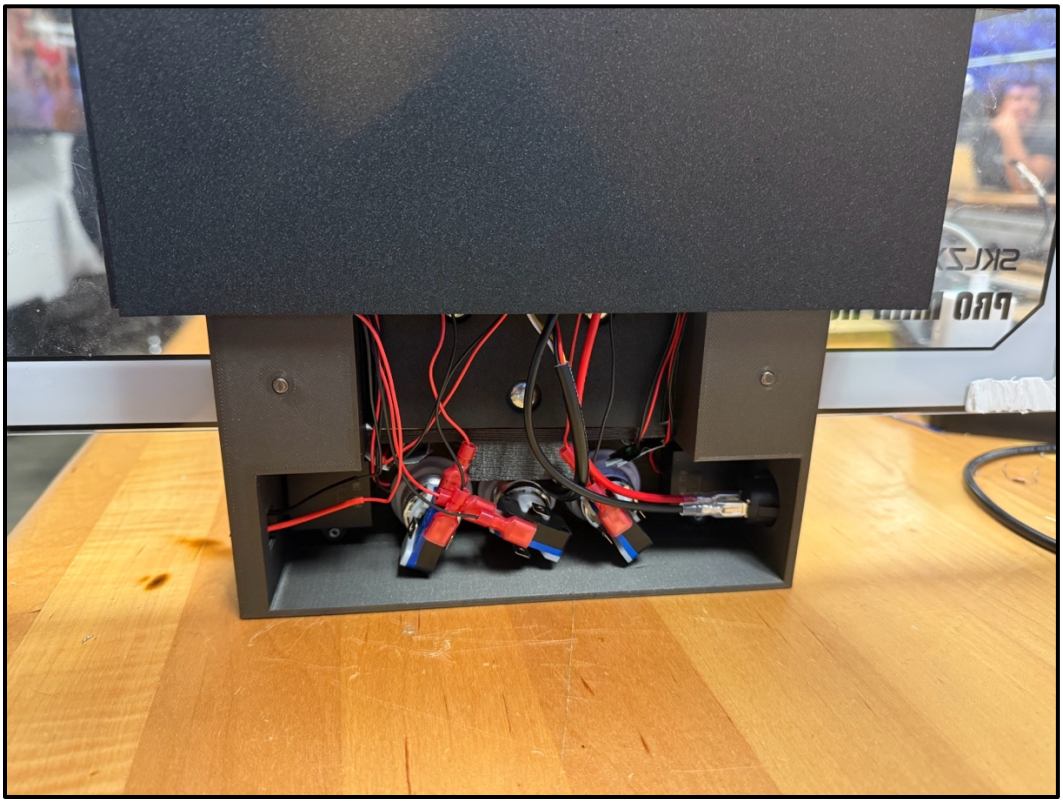


Figure [27]: Back of Controller with all connections

The figure above shows the back of the fully assembled controller mounted on the backboard of the hoop. The two screws on each side are the screws that were used to mount the controller onto the hoop. I used spade connectors for all of the buttons, since they were extremely easy to use and insert into the leads of the buttons. To prepare the spade connectors for use, I would have to crimp a wire into the hole of the connector and then insert the entire connector into the button lead. For other components, such as the LEDs and the reset button, I simply soldered electrical wire into the leads of the components and added shrink tubes to it to avoid any short circuits from happening. Finally, I used JST connectors for both speakers, since it was easier to just poke a wire into JST terminals and jump a wire from there to the screw blocks on the PCB.

Name: Karthik Raja

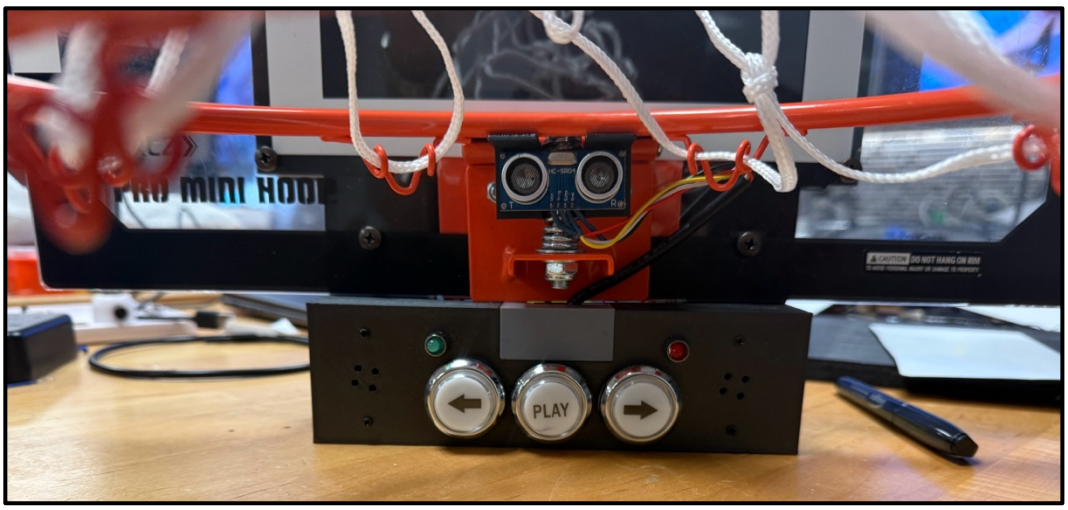


Figure [28]: Front of Controller after Assembly trial and test

The figure above shows the front of the fully assembled controller mounted on the backboard of the hoop as well as the ultrasonic sensor being held up by the 3D printed u-clamp. The front of the controller houses three different buttons (left, right, select), two LEDs mounted onto LED mounts, and the areas where sound will come from the speakers behind it. The right side of the controller will house the reset button, while the left side of the controller will house the on/off rocker switch. The ultrasonic sensor is screwed into the four holes of the u-clamp I made, which is explained earlier above. In order to test the ultrasonic sensor, I had to guess and check the distance I should measure from the sensor to detect the ball going in and how long the sensor should wait for the echo of the sound wave. This was the most difficult part of the project because I had screw and unscrew every time I wanted to test the updated code on the ATMega chip. For future iterations, I wish to place the chip in a place that does not require me to unscrew all of the screws in order to burn new code onto ATMega chip. Every time I tested the sensor, I would check to see if in-and-outs would count as made baskets and see if a made basket does indeed get registered as a make. After several attempts of trying to tune the sensor, I finally got an optimal value. I got the final value of 15cm for the distance and 10,000us for the time delay for detecting the echo of the triggered sound wave.

Figure [29]: Back of Basketball Hoop

The figure above shows the back of the fully assembled basketball hoop. As seen in the image, the PCB enclosure is mounted in the middle of the hoop, the foam spacers and amounted in the four sides of the hoop, the controller mounted in the bottom of the hoop, the 9V battery holder mounted to the left of the PCB, and the hook extensions are placed on the top right before the hooks itself. All of these 3D printed components are explained in detail earlier above. Furthermore, I added foam pads to the front and back of all of these 3D printed components as I did not want them to scratch either the backboard or the door that it is hanging on.

Video of Functionality - Click this link to access the video of functionality.

Summary, Conclusions, and Future Work

One of the most important things I learned throughout this project was that it is important to breadboard designs exactly as it is on the PCB schematic. Initially, I prototyped using the Arduino Uno dev board. This was one

of the mistakes I had made due to the fact that the DFPlayer Mini was working on the dev board; however, it was not operating on the PCB that just used the ATMega 328 chip. Another important thing we learned was that there are many different types of voltage regulators and not all of them have the same pinouts because some voltage regulators are for constant negative voltages. If we had used the correct footprint for the desired voltage regulator, we could have saved multiple hours of debugging. Another lesson learned was that having small capacitors, such as a 10uF capacitors between power and ground will help provide stable voltage levels and prevent fluctuations in voltage levels. When I was splicing the power wire of the 9V battery connector, none of the components would turn on because they were getting 4.5V, instead of closer to 4.9V. After I realized that I forgot to put a decoupling capacitor between power and ground, I jumped wires from the bottom of the PCB to add the decoupling capacitor in the PCB. After adding the capacitor, the spliced battery connector worked perfectly fine.

Overall, my project is functioning exactly how I intended it to. However, as mentioned above, I made some modifications to my initial proposal. My original proposal for my project involved making a mini size basketball arcade game; however, the proposal consisted of a 3-D printed basketball hoop, break-beam sensors, joystick module, seven-segment display, passive buzzer, LCD display, and a reset button. The break-beam sensors would have been put on the rim on the model and would have detected if there has been a ball that has got past the rim, which indicates that the user shot it in the hoop successfully. The seven-segment display is to indicate the number of points the user scored in the time duration. This implementation is still in my final product as well. The LCD display is to help the user select the mode they would like to play in (30 sec or 60 sec). This implementation is still in my final product as well, but I added two game modes to the initial proposal: Horse and 3-point shootout. The passive buzzer would have been used to indicate a made bucket and the time has ran out. The joystick would have been used to navigate through the different game modes (displayed on the LCD screen) and select the desired game mode. I decided to divert away from the initial proposal explained above due to the fact that I realized that having a small desk size 3-D printed basketball hoop would not have been fun compared to a much bigger indoor basketball hoop, where the player can stand up, run around, and shoot like they normally would. Thus, my project is functioning exactly how I expected it to be.

If I was to iterate upon the current design, I would make another game mode, called Freestyle, where the player can shoot for as long as they would and increment the score accordingly, and there will be no time limits in

the game mode. This would allow the players to play for as long as they would like without any timeouts and see how many points they can score before they get tired. Another thing I would like to iterate upon is placing the ATMega chip in a place that does not require me to unscrew all of the screws in order to burn new code onto ATMega chip. Every time I tested the sensor, I would have to unscrew and screw back all of the screws from the basketball hoop in order to upload new code to the ATMega chip. Finally, I would also try to reduce the size of the PCB as well. I gave myself a little too much room for all of the wires, thinking that having more space is better than not having enough. However, making a better PCB would be better to show to employers and organizing the PCB in a way that would be best to connect all of the components in the most optimal position. I am planning on doing these changes over the summer during my spare time.

Final Presentation

I presented to Dr. Dickerson in-person on May 1st at 9:30am. Dr. Dickerson and I were testing the game by playing HORSE and the 60-second drill game modes. The game is working as intended and, although it was the most tedious part of the project, tuning the ultrasonic sensor was worth the while, because it does an amazing job at detecting misses and makes now.