I. Purpose and Background

A digital coin bank is a contemporary spin on the classic piggy bank. It employs various forms of technology to enhance the individual's saving process. A digital coin bank, unlike traditional coin banks, enables the user to keep track of the amount of money already saved. There are multiple advantages to having a digital coin bank over a traditional one, including convenience and accuracy. It also teaches individuals how to manage their finances in an introductory manner.

The discovery that there are multiple methods to improve the traditional coin bank inspired the researchers. In order to accomplish this, they also desired to test the limits of their hardware and programming expertise. They accomplished this by integrating a PIC16F877A and Arduino UNO with a coin bank. The PIC functions and Arduino UNO as the system's central processing unit, containing the majority of the logic along with peripheral components such as a keypad, sensor, and LCD display. The keypad functions as a security key to ensure that each coin deposit is unique. The sensor and LCD are required to keep track of precisely how much has been saved.

This project's objectives are:

1. To design and develop an intuitive and user-friendly interface for a digital coin bank.
2. Use a PIC16F877A microcontroller and Arduino UNO to implement the coin bank.
3. Validate the dependability and precision of the newly created digital coin bank.

II. Designs and Program

This section describes the implementation methodology for the Digitize Coin Bank using a PIC Microcontroller and Arduino UNO.

PIC16F877A Microcontroller, US-100 ultrasonic sensor, 1602A LCD Display, 4x4 Matrix Keypad, and power supply were the primary components of this project. In addition, the prototype was constructed using jumper wires, breadboards, a 22MHz Oscillator, two 22pF capacitors, and a potentiometer. Before interfacing with the PIC microcontroller, the researchers implemented the digital coin bank using Arduino UNO. The programming software used was Arduino IDE written in the C programming language. After implementing the Arduino microcontroller, the researchers moved on to the PIC microcontroller. To prevent damaging the microcontroller, the entire circuit was simulated using Proteus Pro software. The microcontroller was also programmed with MikroC for PIC.

A. *Proteus Pro and MikroC for PIC*

The capability to install plugins and additional libraries is one of the features of Proteus Pro. These additional libraries include the 1602A LCD display and the Y401 ultrasonic sensor, which are not included in the base version. The 4x4 matrix keypad and potentiometer were included in the library of the base version. In addition to the simulation of PIC16F877A, the software has the capacity to transfer a hex file to the microcontroller.

The figure below depicts the simulated circuit of a PIC microcontroller with an LCD connected to port D, a 4x4 matrix keypad connected to port C, and an ultrasonic sensor connected to port B. To achieve an accurate simulation result and to avoid obtaining erroneous results, the components installed in the simulation were identical to and modeled after the components in the actual circuit.

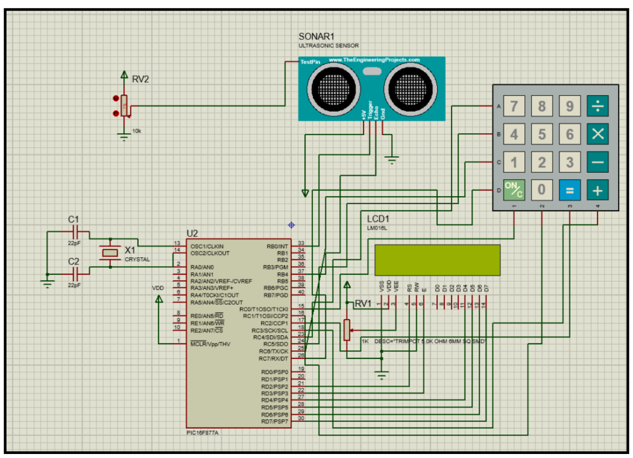


Figure 1. Simulated Circuit in Proteus Pro

C programming was used to create the.hex file for the microcontroller using the MikroC for PIC software to program and construct the simulated microcontroller. The code displayed below represents the effectively compiled and imported program for the PIC microcontroller simulation in Proteus Pro.

```
// LCD module connections
sbit LCD_RS at RD2_bit;
sbit LCD_EN at RD3_bit;
sbit LCD_D4 at RD4_bit;
sbit LCD_D5 at RD5_bit;
sbit LCD_D6 at RD6_bit;
sbit LCD_D7 at RD7_bit;
```

```
sbit LCD_RS_Direction at TRISD2_bit;
sbit LCD_EN_Direction at TRISD3_bit;
sbit LCD_D4_Direction at TRISD4_bit;
sbit LCD_D5_Direction at TRISD5_bit;
sbit LCD_D6_Direction at TRISD6_bit;
sbit LCD_D7_Direction at TRISD7_bit;
// End LCD module connections

// 4x4 Keypad
char keypadPort at PORTC;
unsigned short kp, cnt, oldstate=0;
//End 4x4 Keypad

char* password = "1245";
int positionCnt = 0;

void main()
{
 //LCD
 int a;
 int distance;
 char txt[7];
 Lcd_Init();
 Lcd_Cmd(_LCD_CLEAR);          // Clear display
 Lcd_Cmd(_LCD_CURSOR_OFF);     // Cursor off

 TRISB = 0b00010000;           //RB4 as Input PIN (ECHO)

 Lcd_Out(1,1,"Digitized");
 Lcd_Out(2,1,"Coin Bank");

 Delay_ms(500);
 Lcd_Cmd(_LCD_CLEAR);
 Lcd_Cmd(_LCD_CURSOR_OFF);     // Cursor off

 while(1){
  LCD_Out(1,1,"Welcome!");
  LCD_Out(2,1,"Enter Password");

  //Keypad
  cnt = 0;
```

```
Keypad_Init();
do{
  kp = 0;
  do
    kp = Keypad_key_Click();
  while(!kp);
  switch (kp){
    case 1: kp = 49; break; //1
    case 2: kp = 50; break; //2
    case 3: kp = 51; break; //3
    case 4: kp = 65; break; //A
    case 5: kp = 52; break; //4
    case 6: kp = 53; break; //5
    case 7: kp = 54; break; //6
    case 8: kp = 66; break; //B
    case 9: kp = 55; break; //7
    case 10: kp = 56; break; //8
    case 11: kp = 57; break; //9
    case 12: kp = 67; break; //C
    case 13: kp = 42; break; //*
    case 14: kp = 48; break; //0
    case 15: kp = 35; break; //#
    case 16: kp = 68; break; //D
  }
  if(kp == '*' || kp == '#' || kp == 'A' || kp == 'A' || kp == 'B' || kp == 'C' || kp == 'D'){
    positionCnt = 0;
    Lcd_Cmd(_LCD_CLEAR);
    LCD_Out(1,1, "INVALID PASSCODE");
    LCD_Out(2,1, "Please try again...");
    Delay_ms(300);
    Lcd_Cmd(_LCD_CLEAR);
  }

  if(kp == '1' ){
    Lcd_Cmd(_LCD_CLEAR);
    LCD_Out(1,1, "****VERIFIED****");
    Delay_ms(300);
    Lcd_Cmd(_LCD_CLEAR);


    //After Verified
```

```
while(1){
  TMR1H = 0;              //Sets the Initial Value of Timer
  TMR1L = 0;              //Sets the Initial Value of Timer

  PORTB.F0 = 1;           //TRIGGER HIGH
  Delay_us(10);           //10uS Delay
  PORTB.F0 = 0;           //TRIGGER LOW

  while(!PORTB.F4);       //Waiting for Echo
  T1CON.F0 = 1;           //Timer Starts
  while(PORTB.F4);        //Waiting for Echo goes LOW
  T1CON.F0 = 0;           //Timer Stops

  a = (TMR1L | (TMR1H<<8));   //Reads Timer Value
  a = a/58.82; //Converts Time to Distance
  a = a + 1;
  IntToStr(a,txt);
  if(a > 15){        //Check whether the result is valid or not
      LCD_Out(1,1, "LEVEL 1");
      LCD_Out(2,1, "SAVE UP!");
      Delay_ms(400);
      Lcd_Cmd(_LCD_CLEAR);
  }
  else if(a > 10 && a <= 15){        //Check whether the result is valid or not
      LCD_Out(1,1, "LEVEL 2");
      LCD_Out(2,1, "KEEP IT UP!");
      Delay_ms(400);
      Lcd_Cmd(_LCD_CLEAR);
  }
  else if(a > 5 && a <= 10){        //Check whether the result is valid or not
      LCD_Out(1,1, "LEVEL 3");
      LCD_Out(2,1, "GOOD JOB!");
      Delay_ms(400);
      Lcd_Cmd(_LCD_CLEAR);
  }
  else if(a > 0 && a <= 5){        //Check whether the result is valid or not
      LCD_Out(1,1, "LEVEL 4");
      LCD_Out(2,1, "GOOD JOB!");
      Delay_ms(400);
      Lcd_Cmd(_LCD_CLEAR);
  }
```

```
                Delay_ms(400);
            }
        }
    } while (1);
  }
}
```

Figure 2. MikroC for PIC Code

B. *PIC16F877A Microcontroller, PICKit 2, PICKit USB Programmer and MPLAB*

The PIC16F877A microcontroller has 40 pins with Ports A-D where various inputs and outputs can be connected, including a reset pin, VDD and Vss pins, and oscillator clock pins. The pinout and orientation of the aforementioned microcontroller are depicted in the figure below.

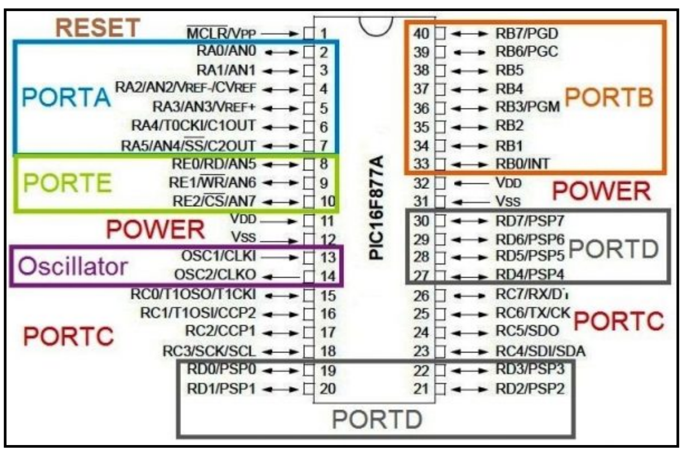


Figure 3. Orientation and Pin Diagram of PIC16F877A Microcontroller
Source: George, L.

The connection of the actual circuit corresponds to the connection in Figure 3's simulation circuit. In order to avoid damaging the actual microcontroller when interfacing with the PIC microcontroller, the microcontroller can be programmed using PICKit 2, PICKit USB Programmer, and MPLAB once the circuit has been successfully connected to the breadboard and the simulated circuit has executed the program as expected in Proteus Pro.

The PICKit 2 will be the primary programming tool for MPLAB microcontrollers. Additionally, the PICKit USB Programmer will be used to connect the PIC microcontroller to the PICKit 2 and the computer via a USB interface. The PICKit 2 connection is depicted in the figure below.

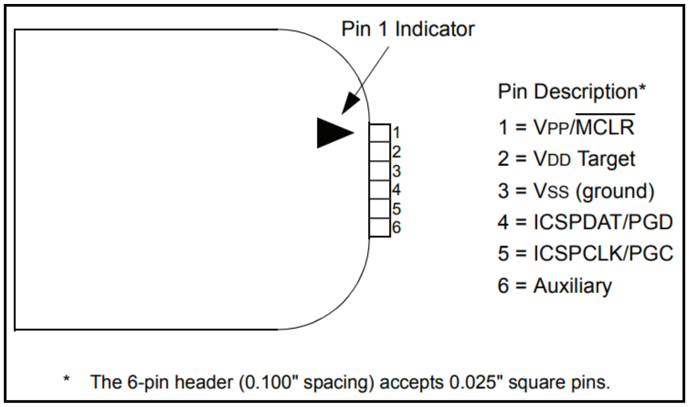


Figure 4. Pin Diagram of PICKit 2
Source: Microchip

Using the MicroC for PIC-generated hex file, the file was exported using MPLAB. In order to export the hex file from the computer, the software must be able to detect the PICKit 2. The exported microcontroller was anticipated to operate similarly to the simulated circuit's output.

C. *Arduino UNO; Arduino IDE*

The researchers also interacted with an Arduino UNO to evaluate the functionality of the individual components. To regulate the behavior of the components, the researchers created and uploaded programs to the Arduino UNO. Figure 5 depicts the arrangement of the components once they have been connected to the Arduino.

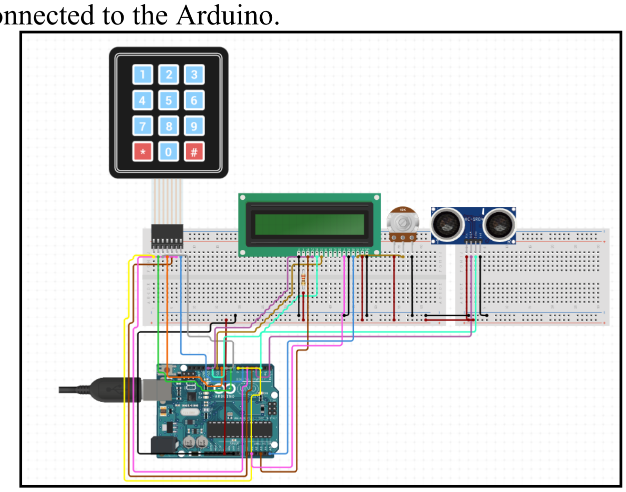


Figure 5. Arduino UNO Circuit Diagram

When interfacing the ultrasonic sensor, the VCC and GND pins of US-100 were connected, respectively, to the 5V and GND terminals of Arduino. The Trig and Echo pins were respectively connected to digital pins 9 and 10.

The VSS, VDD, and VO ports of the LCD were connected to GND, +5V, and the middle pin of a 10k ohm potentiometer for the LCD connections. The LCD's RS, RW, and E ports were connected, respectively, to digital pin A0, GND, and digital pin 11. The LCD's D4, D5, D6, and D7 pins were then connected to the Arduino UNO's A1, A2, A3, and A4 pins, respectively.

The connections for the 4x4 Matrix Keypad are: for its rows, digital pins 1, 2, 3, and 4; and for its columns, digital pins 5, 6, 7, and 8. The LiquidCrystal.h and Keypad.h libraries were included at the outset of the code using the Arduino IDE. The LCD pins, keypad pins, and password and position count variables were initialized, as well as the LCD and Keypad objects.

```
#include <LiquidCrystal.h>
#include <Keypad.h>
#define trigPin 9
#define echoPin 10

char* password ="1245";
int pozisyon = 0;

const byte rows = 4;
const byte cols = 4;

char keyMap [rows] [cols] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

byte rowPins [rows] = {1, 2, 3, 4};
byte colPins [cols] = {5, 6, 7, 8};

Keypad myKeypad = Keypad( makeKeymap(keyMap), rowPins, colPins, rows, cols);

LiquidCrystal lcd (A0, A1, A2, A3, A4, A5);
```

Figure 6. Code of the initialization of components in Arduino IDE

In the setup function, it begins by initializing the LCD and configuring the pinModes for the Trig and Echo pins of the ultrasonic sensor.

```
void setup() {
  lcd.begin(16, 2);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
```

Figure 7. Code of the setup function in Arduino IDE

In the loop function, the variable 'whichKey' identifies the key that was pressed using the getKey() function. Using lcd.print(), the LCD then prompts the user to input the password.

```
void loop() {
  char whichKey = myKeypad.getKey(); //define which key is pressed with getKey

  lcd.setCursor(0, 0);
  lcd.print("    Welcome");
  lcd.setCursor(0, 1);
  lcd.print(" Enter Password");
```

Figure 8. Code for welcome statement in Arduino IDE

If the user presses an invalid key such as '*', '#', 'A', 'B', 'C', or 'D', the if statement will display an error message on the LCD.

```
  if (whichKey == '*' || whichKey == '#' || whichKey == 'A' || whichKey == 'B' || whichKey ==
'C' || whichKey == 'D') {
    pozisyon = 0;

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("  Invalid Key!");
    delay(1000);
    lcd.clear();
  }
```

Figure 9 .Code for inputting invalid password in Arduino IDE

Figure 10's subsequent lines of code verify that the user has entered the correct password and increment the 'pozisyon' variable if the input key matches the password.

```
  if (whichKey == password[pozisyon]) {
```

```
  pozisyon++;
}
```

Figure 10 .Code for inputting correct password in Arduino IDE

The lines of code in Figure 11 display a message of success on the LCD if the correct password is inputted. The LCD will be reset and the while loop will continue to continuously measure and display the distance.

```
if (pozisyon == 4) {
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("*** Verified ***");
  delay(3000);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("    Digitized");
  lcd.setCursor(0, 1);
  lcd.print("    Coin Bank");
  delay(3000);
  lcd.clear();
```

Figure 11 .Code for displaying correct message in Arduino IDE if password is correct

The 'while' iteration within the 'if (pozisyon == 4)' block continuously measures the distance using the ultrasonic sensor and displays the appropriate message on the LCD based on the measured distance. The loop continues forever until the system is reset or switched off.

Figure 12 demonstrates that the trigPin was set to LOW and a 2 ms delay was applied to assure stability. The trigPin was then set to HIGH for 10 milliseconds before being promptly reset to LOW. Using the pulseIN() function, the duration of the pulse on the echoPin was determined and saved in the variable 'duration'. The pulse's duration is proportional to the coin's distance. The distance was calculated using the formula 'distance = duration * 0.034 / 2', which converts the duration of the pulse into centimeters. The value 0.034 represents the speed of sound in centimeters per microsecond, and the division by 2 accounts for the time required for the sound wave to travel in both directions.

```
while (1) { // infinite loop to continuously measure distance and display it
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
```

```
digitalWrite(trigPin, LOW);
long duration = pulseIn(echoPin, HIGH);
int distance = duration * 0.034 / 2;
lcd.setCursor(0, 0);
```

Figure 12 .Code for continuously measuring the distance in Arduino IDE

The coin bank's level of abundance was determined by comparing the distance value with predetermined threshold values. Using 'lcd.print()', the appropriate message was displayed on the LCD based on the distance value. There was a 3-second latency between each message. Finally, there is a one-second delay before the cycle repeats and remeasures the distance. Observe Figure 13.

```
if(distance > 15){
  lcd.setCursor(0,0);
  lcd.print("    LEVEL 1");
  lcd.setCursor(0,1);
  lcd.print("    SAVE UP");
  delay(3000);
  lcd.clear();
}
else if(distance > 10 && distance <= 15){
  lcd.setCursor(0,0);
  lcd.print("    LEVEL 2");
  lcd.setCursor(0,1);
  lcd.print("  KEEP IT UP!");
  delay(3000);
  lcd.clear();
}
else if(distance > 5 && distance <= 10){
  lcd.setCursor(0,0);
  lcd.print("    LEVEL 3");
  lcd.setCursor(0,1);
  lcd.print(" ALMOST THERE!");
  delay(3000);
  lcd.clear();
}
else if(distance > 0 && distance <= 5){
  lcd.setCursor(0,0);
  lcd.print("    LEVEL 4");
  lcd.setCursor(0,1);
  lcd.print("   GOOD JOB!");
```

```
      delay(3000);
      lcd.clear();

    }
    delay(1000);
   }
  }
}
```

Figure 13. Code for displaying the appropriate level depending on the distance detected in Arduino IDE

III. Testing Methodology and Data Gathering Structure

The same experimental procedure was utilized for interfacing the PIC16F877A and Arduino UNO. For the digital coin bank system, the distance between the ultrasonic sensor and the top of the coin stack was initially measured and recorded during prototype testing. Using the keypad, the system's password was then entered to authenticate the access. The LCD display was scrutinized to ensure that it accurately displayed the security authentication and distance level.

The ranges were as follows:

- distance > 15
- distance > 10 && distance <= 15
- distance > 5 && distance <= 10
- distance > 0 && distance <= 5

These ranges were represented by levels that encourage the user to continue saving.

- distance > 15
  - The LCD displayed “Level 1” and “SAVE UP”
- distance > 10 && distance <= 15
  - The LCD displayed “Level 2” and “KEEP IT UP!”
- distance > 5 && distance <= 10
  - The LCD displayed “Level 3” and “ALMOST THERE!”
- distance > 0 && distance <= 5
  - The LCD displayed “Level 4” and “GOOD JOB!”

A new ultrasonic sensor-to-coin stack distance measurement was recorded after additional coins were added to the coin bank. After each addition, the distance was measured. To ascertain the ultrasonic sensor's accuracy, the measured and actual distances were compared. Finally, the keypad and LCD display were inspected to ensure proper operation.

Overall, PIC and Arduino both configure a simple digital coin bank system with a password-protected interface and a display that signifies the level of fullness based on the ultrasonic sensor's measurement of distance. The user must enter a password to gain access to the system, and the system continuously updates the display based on the measured distance.

IV. Results of Tests

During the project's testing phase, a complete circuit was assembled. Despite a thorough examination of the connections, reprogramming of the PIC, and rewriting of the code, the circuit remained inoperable. LCD was the furthest that the circuit was able to advance. Following the diagram of the assembled circuit are the anticipated outputs.

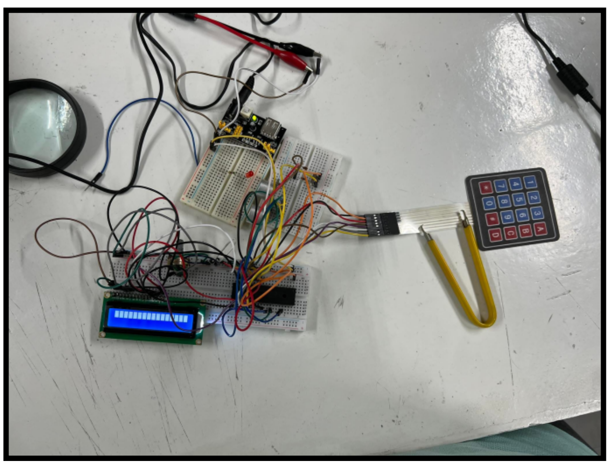


Figure 14. Interfacing PIC16F877A Microcontroller

The figure depicts a simulation of the circuit's initial power up. It contains the project's title and will be followed by another scene.

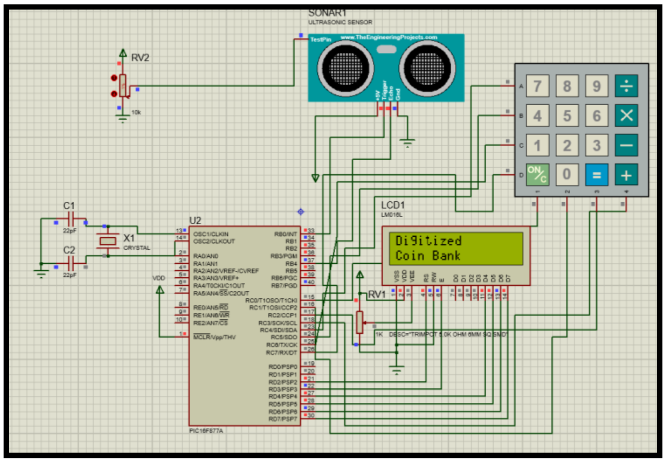


Figure 15. Introduction Prompt Simulation

Figure 16 is the next state after the initial boot up. It welcomes the user and asks for the password for the coin bank.

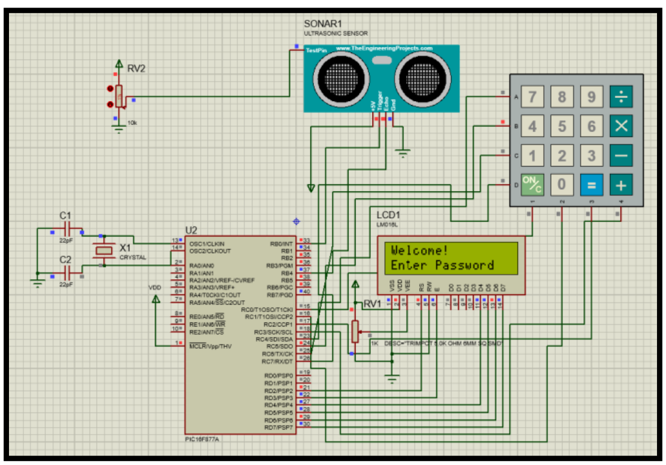


Figure 16. Enter Password Prompt Simulation

Figure 17 displays 'Verified' because the correct passcode was entered. The passcode is entered using a keypad. When the incorrect passcode is entered, the LCD displays an invalid prompt and prompts the user to re-enter their passcode.

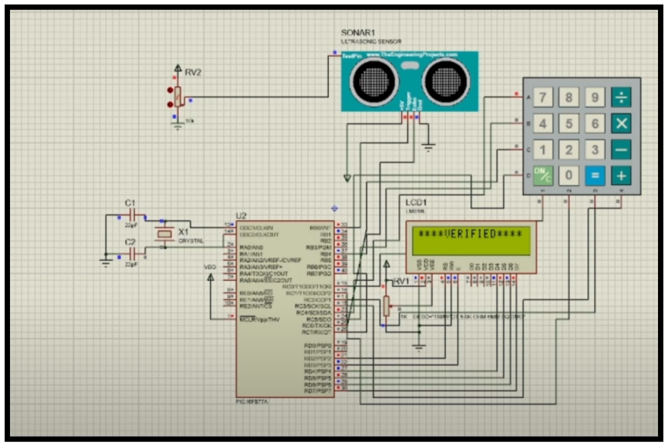


Figure 17. Password Verified Prompt Simulation

The outputs depicted in Figures 18 through 21 are expected. Level 1 is triggered when the sensor detects that the object is more than 15 cm away. If the distance is less than or equal to 15 centimeters but still greater than 10 centimeters, the distance is classified as level 2. When the distance is less than or equal to 10 cm but greater than 5 centimeters, level 3 is reached. The final level applies when the distance is less than or equal to 5 centimeters but greater than 0 centimeters.

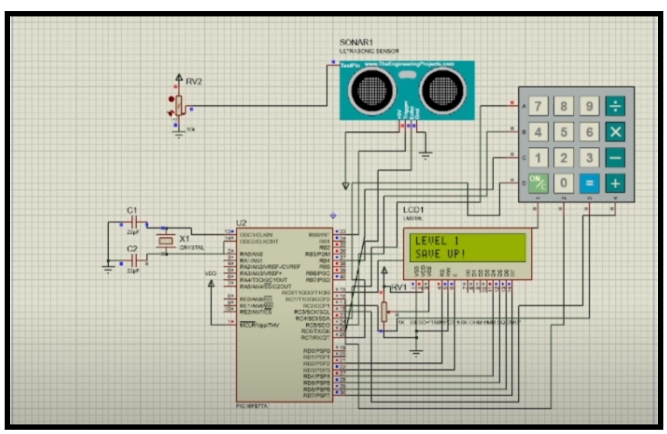


Figure 18. Level 1 Simulation

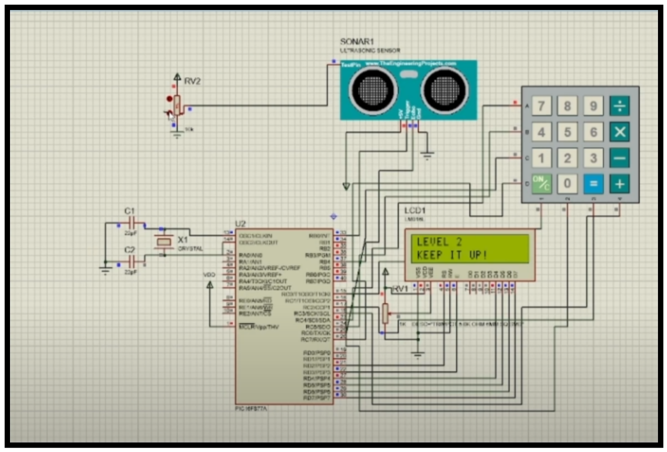


Figure 19. Level 2 Simulation

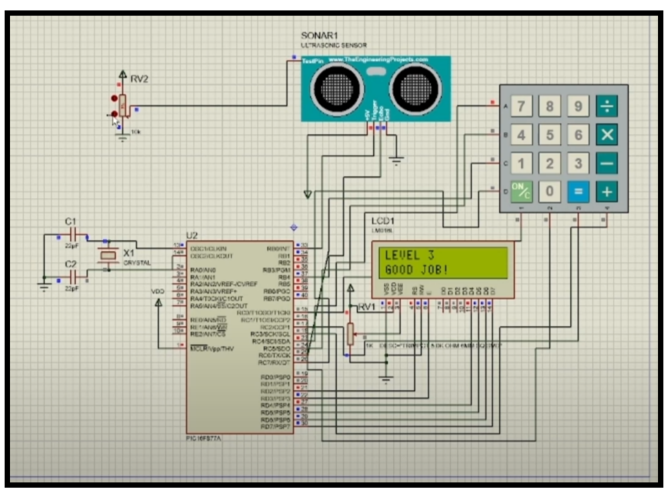


Figure 20. Level 3 Simulation

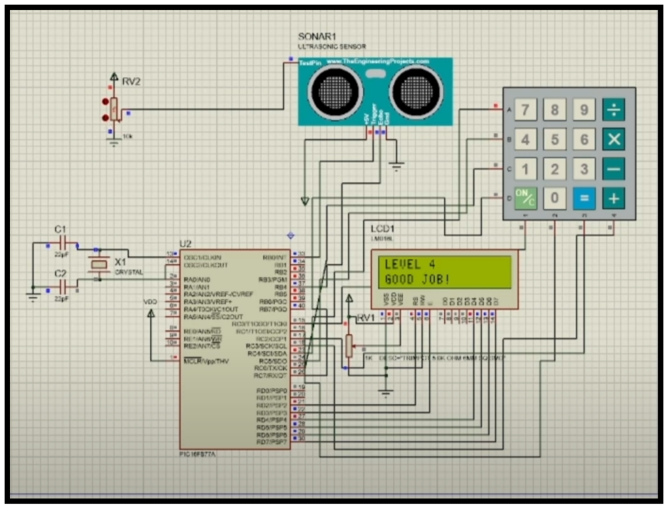


Figure 21. Level 4 Simulation

Figures 22-29 depict the expected output of the PIC circuit. Arduino was used to construct the circuit below, but the components used are all identical. The accompanying numbers provide a concrete illustration of the output. It can be seen that the passcode verification is functioning, as well as the various levels of the digital coin bank. The various levels of the coin bank are measured with a ruler to demonstrate that the sensor is accurate.

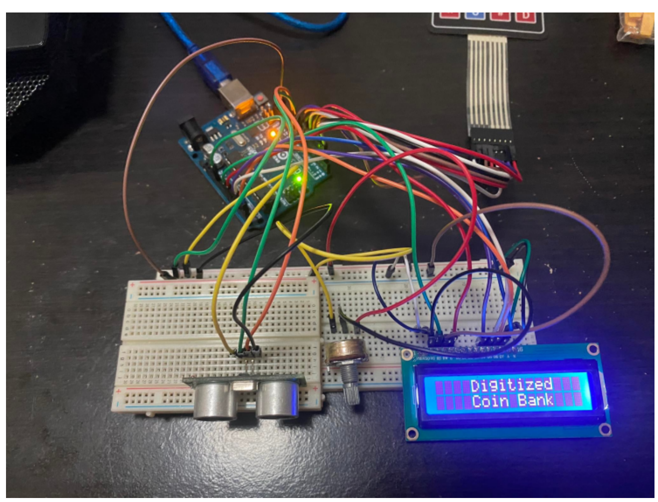


Figure 22. Introduction prompt

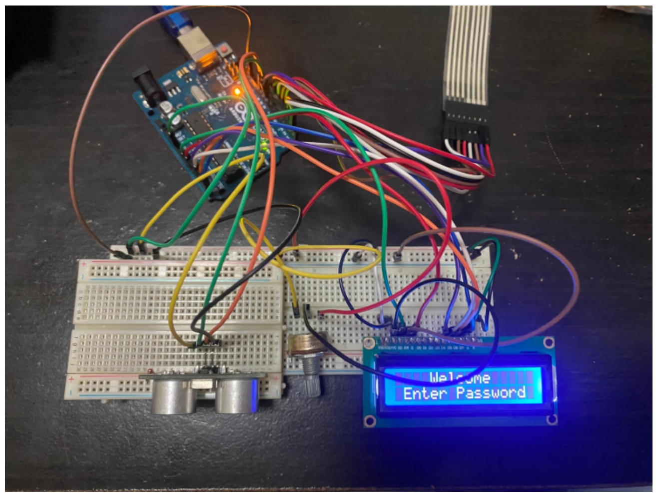


Figure 23. Enter password prompt

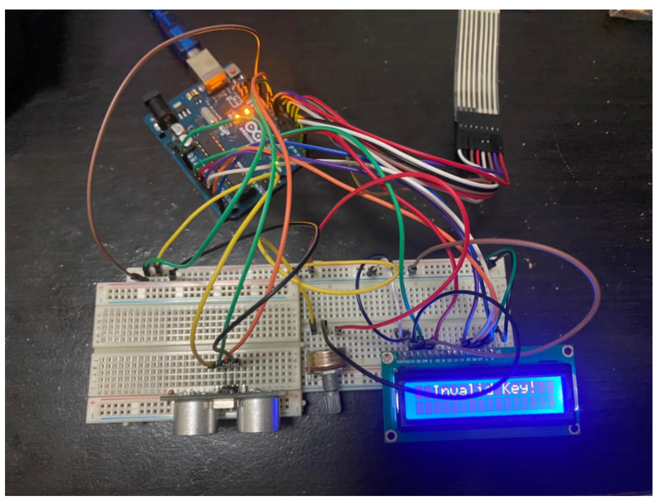


Figure 24. Invalid key prompt

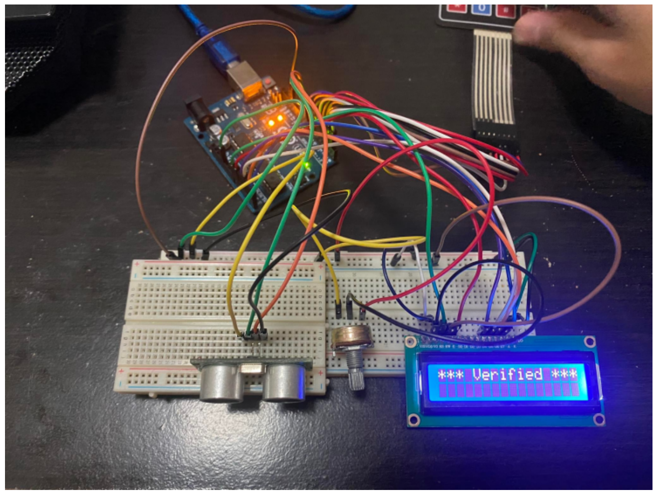


Figure 25. Password verified prompt

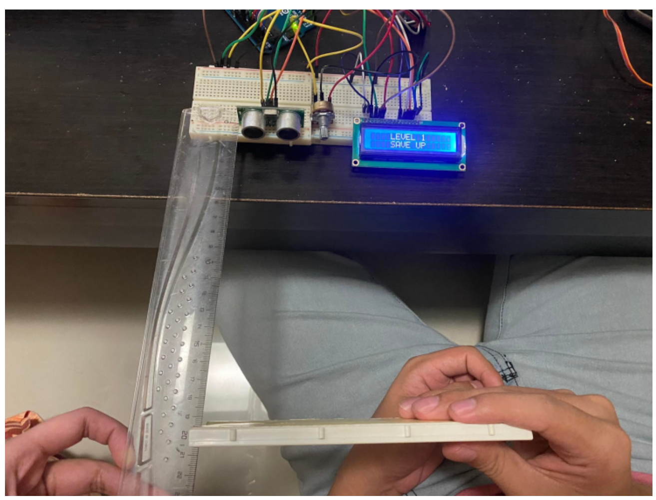


Figure 26. Level 1 prompt (distance > 15 cm)

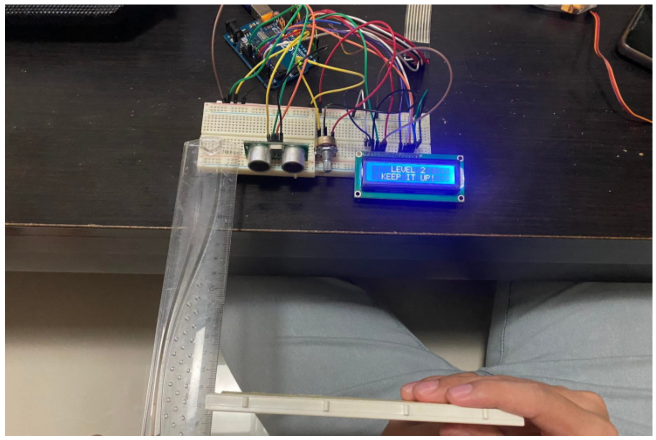


Figure 27. Level 2 prompt (distance > 10 && distance <= 15)

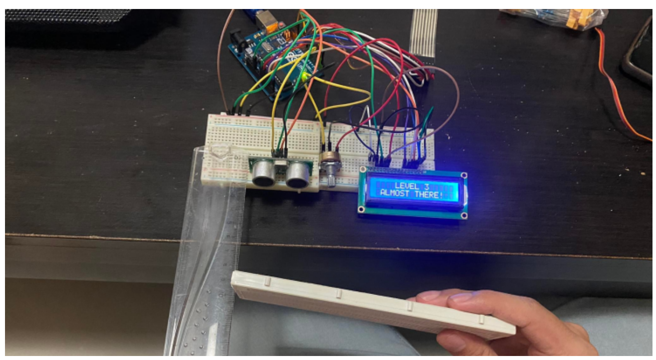


Figure 28. Level 3 prompt (distance > 5 && distance <= 10)

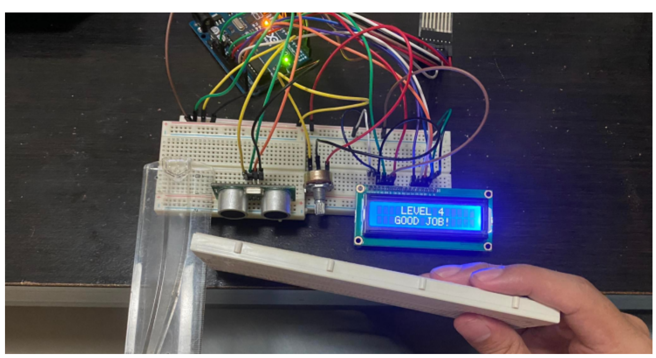


Figure 29. Level 4 prompt (distance > 0 && distance <= 5)

V. Discussion of Results (Comparison of Performance and Cost)

An Arduino board costs 650.00 pesos and a PIC microcontroller costs roughly 400.00 pesos. While an Arduino board already has the necessary components embedded onto it, the PIC needs a PCB or a breadboard with external components such as an oscillator and power supply for it to fully function. The researchers were unable to test the full functionality of the PIC microcontroller but they were able to create a working prototype for the Arduino. In terms of performance, the Arduino is much more straightforward and with just lines of coding it was able to perform what it needed to perform. On the other hand, the PIC microcontroller needed multiple initialization lines for it to function and in the end it was still lacking. Costwise, an Arduino is worth more for beginners and hobbyists as it is a much more straightforward piece of equipment rather than a PIC that needs more advanced skills and a bigger selection of materials.

VI. Conclusions on Device Viability, Selection of which device is likely a better item to field people to use.

In terms of device viability, for beginners like the researchers Arduino would be the choice. The strengths of Arduino is its less complex interface that allows simple wirings to achieve the necessary goal. The programming part of Arduino as well is much more straightforward as most of the digital ports already have expected functionality. Arduino also has a larger community that is much more active to this date that allows updated resources to anyone who needs them. That being said, PIC microcontrollers are not completely being looked down upon. As a microcontroller, it also has its strengths such as it being a much more powerful processor compared to the arduino. While it also has a large array of components that it can connect to, it might be a little bit troublesome as advanced programming is required. The researchers experienced this while interfacing the PIC with the LCD as there were some bumps that they encountered. All in all, both the PIC and Arduino are viable choices to developers like the researchers but it just depends on the intended use case.

Bibliography

George, L. (2018, January 14). Getting started with pic microcontroller - CCS C compiler. electroSome. Retrieved April 20, 2023, from https://electrosome.com/getting-started-pic-ccs-c/

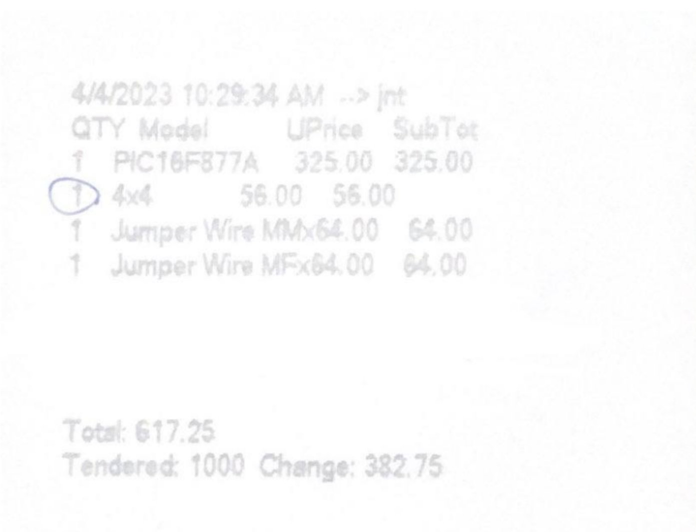
Microchip technology. (n.d.). Retrieved from https://ww1.microchip.com/downloads/en/DeviceDoc/51553E.pdf

Annex A. Schedule of Activities in terms of Day Basis

Activity	**Project Work Days**						
	1	**2**	**3**	**4**	**5**	**6**	**7**
Proposal	Done						
Design Schematic (Arduino UNO)		Done					
Program (Arduino IDE)		Done					
Fabrication (Arduino IDE)		Done					
Design Schematic (PIC16F877A)			In Progress	Done			
Program (PIC16F877A)			In Progress	Done			
Fabrication (PIC16F877A)			In Progress	Done			
Test Evaluation (Arduino UNO)		Done					
Test Evaluation (PIC16F877A)			In Progress	Done			

Result Analysis (Arduino UNO)		Done					
Result Analysis (PIC16F877A)			In Progress	Done			
Report Writing				Pending	In Progress	Done	
Submission of Project							Done

Annex B. Bill of Materials for Each Tested Design

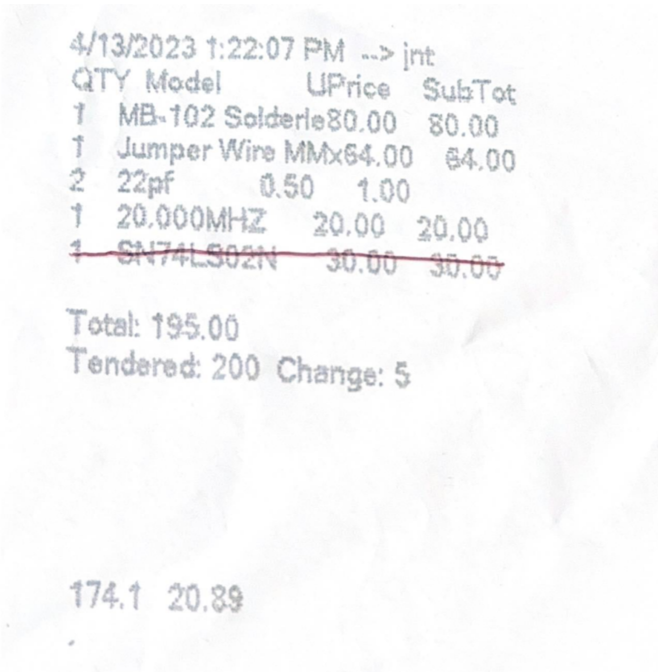


4/4/2023 10:29:34 AM --> jnt

QTY	Model	UPrice	SubTot
1	PIC16F877A	325.00	325.00
1	4x4	56.00	56.00
1	Jumper Wire MMx	64.00	64.00
1	Jumper Wire MFx	64.00	64.00

Total: 617.25

Tendered: 1000 Change: 382.75

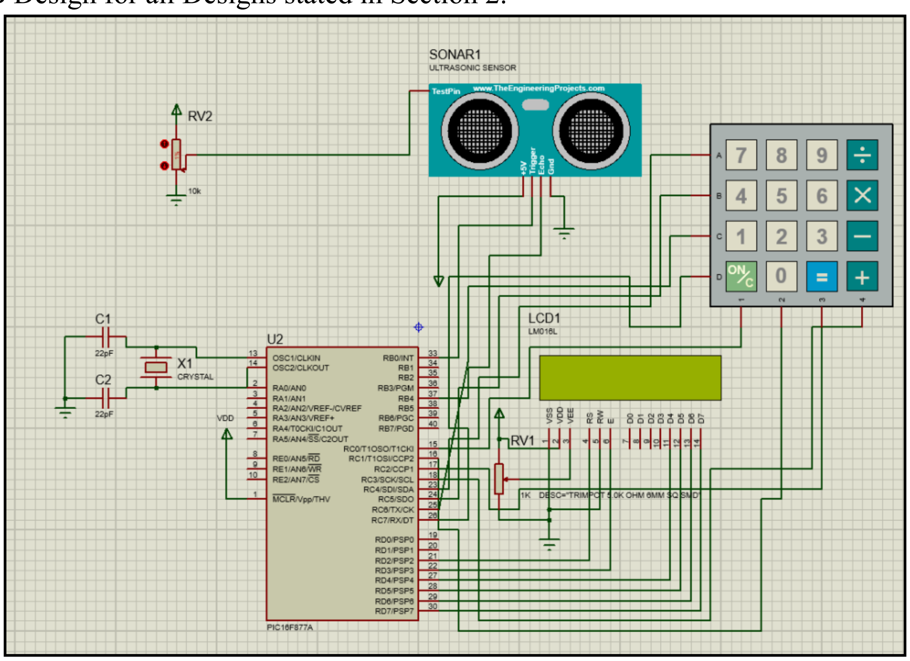

4/13/2023 1:22:07 PM --> jnt

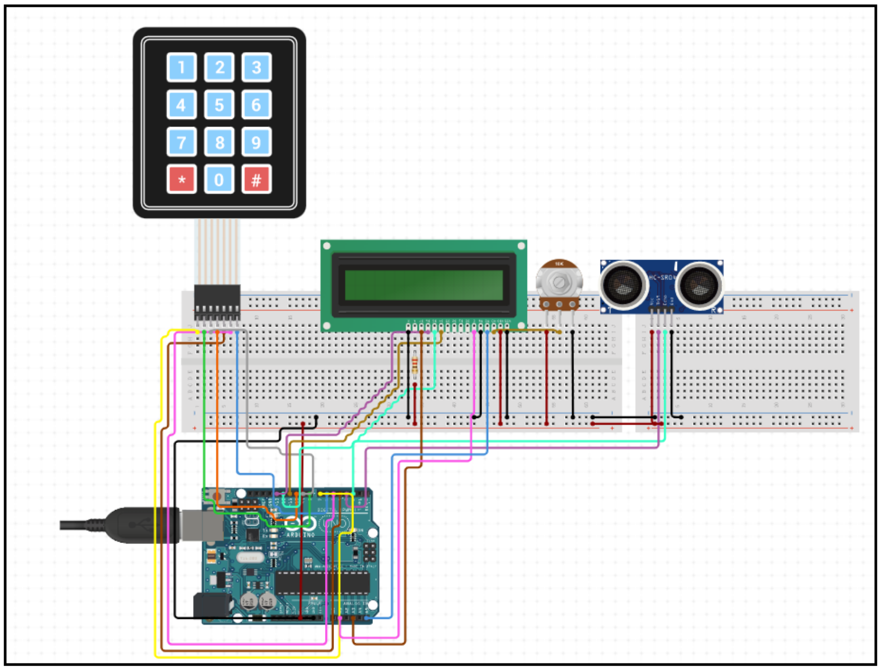
QTY	Model	UPrice	SubTot
1	MB-102 Solderle	80.00	80.00
1	Jumper Wire MMx	64.00	64.00
2	22pf	0.50	1.00
1	20.000MHZ	20.00	20.00
~~1~~	~~SN74LS02N~~	~~30.00~~	~~30.00~~

Total: 195.00
Tendered: 200 Change: 5

174.1 20.89

Annex C. PCB Design for all Designs stated in Section 2.





1
2
3
4
5
6
7
8
9
*
0
#