Team Member 1:

- **Name:** Christopher Rattanasamay
- **NetID:** cratta3
- **Email:** c.rattanasamay@gmail.com

Team Member 2:

- **Name:** Aakash Kotak
- **NetID:** akotak4
- **Email:** akotak4@uic.edu

Team Member 3:

- **Name:** Patricia Guera
- **NetID:** pguera3
- **Email:** pguera3@uic.edu

LED Memory Game

For our Arduino project, we are making a game called the "LED Memory Game" and the purpose of this game is for the user to memorize and copy the random order that the LEDs that light up in a certain amount of time. The user will have three lives, and as you progress into the game, the harder the levels become.

Project Description:

Our project idea is to create a memory game that consists of the user matching the pattern of randomly lit LEDs. There will be five LEDs that will light up in a random sequence each round, and it is the player's job to correctly match the sequence of lights using the five push buttons that correspond with each LED. The user will have three lives, and the user will have to complete each round within a certain amount of time. The timer will be displayed on our 4-digit 7 segment display, the score and the user's lives will be kept track on an LCD screen and represented as three LEDs. If the user runs out of time before the sequence is complete, the player loses a life and the timer will reset back to 60 seconds, along with repeating the level's pattern again.

We will be using three Arduinos in our project, and each Arduino will be used for different parts of the game.

- **Arduino 1**

Arduino 1 will be used to keep track of all of our game mechanics and statistics. Arduino 1 keeps track of the random sequence of lights using a queue, the player input sequence, and keeping track of the game statistics, such as score, lives, and time the player has left in the game.

- **Arduino 2**

Arduino 2 is mainly used for output, and will have the LCD screen that will keep track of lives, and score. There will also be three LED lights to represent how many lives there are left. We also have the buzzer play sounds when the LEDs light up and when the player presses on a button.

- **Arduino 3**

Arduino 3 will output the digital timer on how long the player has, the time would be passed from arduino 1 to arduino 3.

Communication between Arduinos:

Arduino 1 will control mainly everything that goes on with the game such as printing out the sequence of lights via the LED's and take user input via the buttons. Also, Arduino 1 will be keeping track of all the game statistics such lives, time, and score and passing them on to Arduino 2 and 3 where Ardunio 2 and 3 will output the results.

Expected Inputs/Outputs:

We have a few expected input and output devices that we were planning on using. The first part is initializing the level, lives, and messages, and it will output on our LCD display from Arduino 2. Secondly, the LEDs will light up in a random sequence, and the user will be required to input the correct sequence using the push buttons corresponding to the five LEDs. If the user guesses the pattern incorrectly, the buzzer will go off, you will lose a life, and the level will reset. If the user inputs the correct pattern, your score will go up along with the level.

Original Work:
We feel that our project has some originality, such as the use of three Arduinos instead of one. We have also added features that aren't shown in other memory games such as our buzzer buzzing when the user incorrectly matches the pattern, each button having a different tone using the pitches library, three lives mechanic, patterns get progressively harder when you go to the next level, an LCD screen to display the player's statistics such as score and lives, and three LEDs that represent our lives. Also, our originality is in our code as well. We are using a queue where most memory games aren't as intricate in their code and use arrays.

Building the Project:
Our first step in creating our game was that we set up our Arduino 1 similar to our first and second lab where we set up our 5 buttons and 5 LEDs. Arduino 2 will just be the standard LCD Display along with our buzzer which is from both lab 3 and 5. Our Arduino 3 will have only the 4 digit 7 segment display to show the timer, and we figure out how to wire this by looking at the Arduino website.

How will this Project be Used?

Our project is supposed to be used for fun mainly since it's suppose to be a memory game. It could be used to improve and test memory.

Timeline:

- **Week 11/07**
  - Putting all Arduinos together / Coding [COMPLETED]
- **Week 11/11**
  - Putting all Arduinos together / Coding [COMPLETED]
- **Week 11/18**
  - Bug Fixing / Coding [COMPLETED]
- **Week 11/25**
  - Design Presentation [COMPLETED]
- **Week 12/02**
  - Project Demo [COMPLETED]

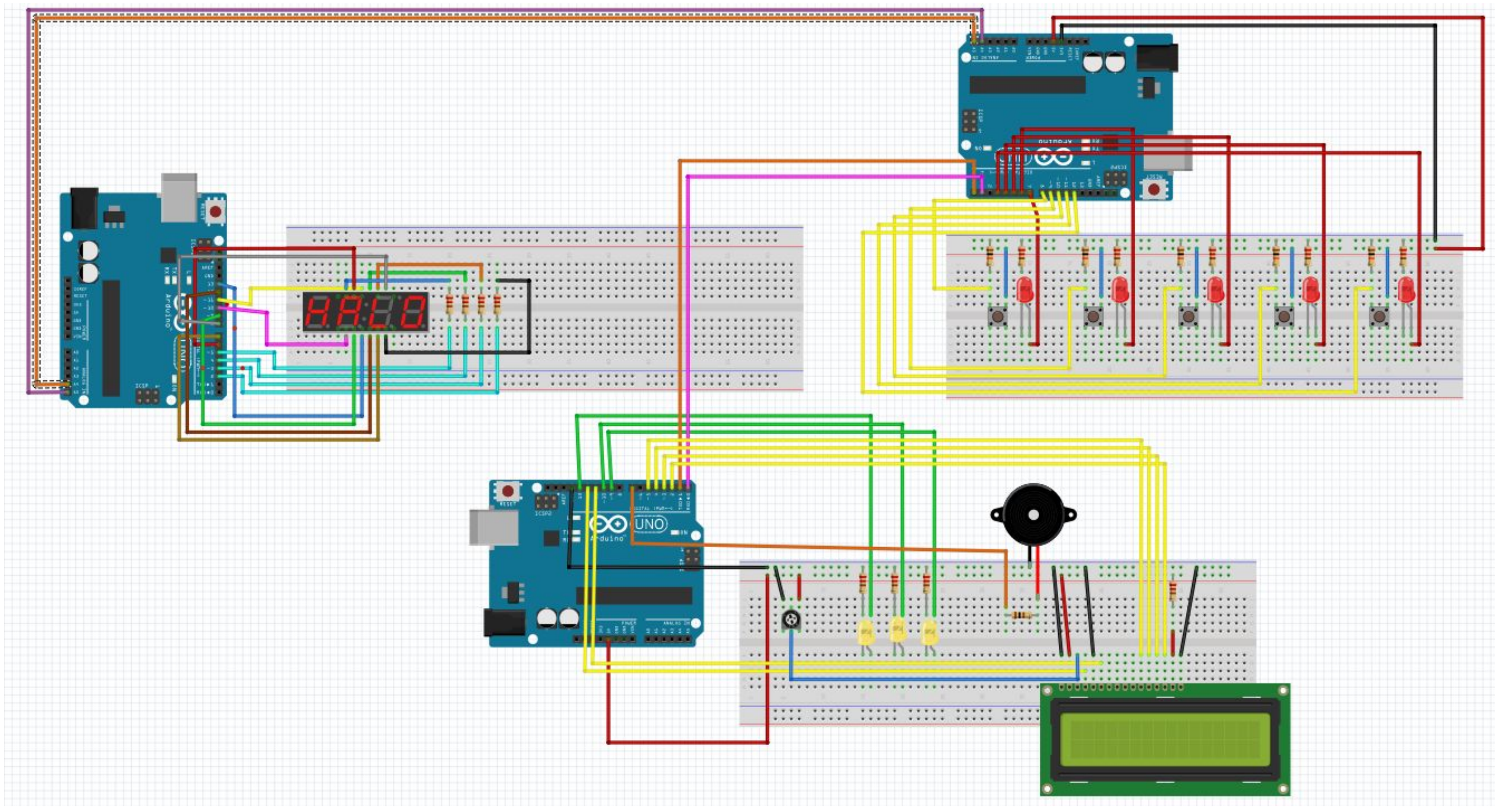
Materials:

- LED light x 8
- Push button x 5
- LCD Screen x 1
- Buzzer x 1
- 220 Ohm Resistor x 12+
- Potentiometer x 1
- 4 Digit 7 - Segment Display x 1
- Multiple Wires
- 10K Ohm Resistor x 2

List of References:

We've looked at this video of an LED memory game, similarly to ours, except we will have more features and Arduinos used, and also we are going to be using a Queue library.
https://www.youtube.com/watch?v=Tcp_6L80kY0&t=1s
https://www.youtube.com/watch?v=b7yCvvrDPSw
https://playground.arduino.cc/Code/QueueList/
https://www.arduino.cc/reference/en/language/functions/random-numbers/random/

Fritzing Diagram



Code:

arduino_main.Ino

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

#define TIMESIZE 2

QueueArray <int> queue;
QueueArray <int> playerInput;

int level = 1; // game starts at level 1, increments when a sequence is correct
int lives = 3; // player starts with 3 lives, decrements when a player gets pattern
incorrect or time runs out
int score = 0; // player starts at 0 points, increments when a player guesses a
sequence correctly

bool nextLevel = false; // set to false, becomes true when passes a level
bool gamePlay = true;

const int led1 = 7; // LED 1 - set to pin 7
const int led2 = 6; // LED 2 - set to pin 6
const int led3 = 5; // LED 3 - set to pin 5
const int led4 = 4; // LED 4 - set to pin 4
const int led5 = 3; // LED 5 - set to pin 3

const int button1 = 8; // button 1 - set to pin 8
const int button2 = 9; // button 2 - set to pin 9
const int button3 = 10; // button 3 - set to pin 10
const int button4 = 11; // button 4 - set to pin 11
const int button5 = 12; // button 5 - set to pin 12

int buttonState1 = 0; // reads button 1 status
int buttonState2 = 0; // reads button 2 status
int buttonState3 = 0; // reads button 3 status
int buttonState4 = 0; // reads button 4 status
int buttonState5 = 0; // reads button 5 status

byte Time = 60;
int Timer;

void setup() {
  pinMode(led1, OUTPUT); // set pin to led 1
  pinMode(led2, OUTPUT); // set pin to led 2
  pinMode(led3, OUTPUT); // set pin to led 3
  pinMode(led4, OUTPUT); // set pin to led 4
  pinMode(led5, OUTPUT); // set pin to led 5
  pinMode(button1, INPUT); // set pin to button 1
  pinMode(button2, INPUT); // set pin to button 2
  pinMode(button3, INPUT); // set pin to button 3
  pinMode(button4, INPUT); // set pin to button 4
  pinMode(button5, INPUT); // set pin to button 5
  Serial.begin(9600);
  Serial.setTimeout(100);
  Wire.begin();
  Wire.setTimeout(100);
}

void loop() {
  // Wait to move onto the next level
  if(!nextLevel){
    if(!queue.isEmpty()){
      while(!queue.isEmpty()){
        queue.pop();
      }
    }
    populateQueue(queue, level); // generates a new level
    nextLevel = true; // set to true when player goes to next level
  }
  while(!queue.isEmpty() && lives > 0 && gamePlay == true) // Add a time constraint
as well
  {
     Wire.beginTransmission(4);
     Wire.write(Time); // adds to timer to arduino 1
     Wire.endTransmission(4);
     user_input(queue, playerInput);
     Wire.beginTransmission(4);
     Wire.write(100);
     Wire.endTransmission(4);
     level++; // increment level
     turnOnAllLights(); // turn on LEDs
     nextLevel = false;
  }
}

void user_input(QueueArray<int>& queue,QueueArray<int>& playerInput){
  while(!queue.isEmpty() && lives > 0 && Timer > -1 && gamePlay == true){
    Wire.requestFrom(4,1); // grabs the time from arduino 3 to arduino 1
    while(Wire.available() > 0){
      Timer = Wire.read(); // reads the timer
      // player loses a life with timer hits 1
      if(Timer == 1){
        level--; // decrease level to stay in same level
        wrong_input(); // player enters wrong button
        nextLevel = false;
        // empty the queue
        while(!queue.isEmpty()){
          queue.pop();
        }
        Wire.beginTransmission(4); // reset the timer
        Wire.write(100); // ends the timer
        Wire.endTransmission(4); // exit
        delay(100);
        nextLevel = false;
        return;
      }
    }

    // reads each button pressed as a button state
    buttonState1 = digitalRead(button1);
    buttonState2 = digitalRead(button2);
    buttonState3 = digitalRead(button3);
    buttonState4 = digitalRead(button4);
    buttonState5 = digitalRead(button5);
    delay(10);

    if(buttonState1 == HIGH){
      // top of the queue is led 1
      if(led1 == queue.peek()){
```

```
      Serial.println("note 1"); // button 1 to buzz
      digitalWrite(led1, HIGH); // on
      delay(75);
      queue.pop();
      digitalWrite(led1, LOW); // off
      delay(75);
    }
    // wrong button
    else{
      wrong_input();
    }
  }

  if(buttonState2 == HIGH){
    // top of the queue is led 2
    if(led2 == queue.peek()){
      digitalWrite(led2, HIGH); // on
      Serial.println("note 2"); // button 2 to buzz
      delay(75);
      queue.pop();
      digitalWrite(led2, LOW); // off
      delay(75);
    }
    // wrong button
    else{
      wrong_input();
    }
  }

  if(buttonState3 == HIGH){
    // top of the queue is led 3
    if(led3 == queue.peek()){
      Serial.println("note 3");
      digitalWrite(led3,HIGH);
      delay(75);
      queue.pop();
      digitalWrite(led3,LOW);
      delay(75);
    }
    // wrong button
    else{
      wrong_input();
    }
  }
  if(buttonState4 == HIGH){
    // top of the queue is led 4
    if(led4 == queue.peek()){
      Serial.println("note 4");
      digitalWrite(led4, HIGH);
      delay(75);
      queue.pop();
      digitalWrite(led4, LOW);
      delay(75);
    }
    // wrong button
    else{
      wrong_input();
    }
  }
  if(buttonState5 == HIGH){
      // top of the queue is led 5
      if(led5 == queue.peek()){
        Serial.println("note 5");
        digitalWrite(led5,HIGH);
        delay(75);
        queue.pop();
        digitalWrite(led5,LOW);
        delay(75);
      }
      // wrong button
      else{
        wrong_input();
      }
    }
  }

  // game over
  if(lives == 0){
    Wire.beginTransmission(4);
    Wire.write(101);
    Wire.endTransmission(4);
    Serial.println("Over");
    delay(500);
    exit(0);
  }
  return;
}

// creates a random generated pattern between our 5 LEDs
void populateQueue(QueueArray<int>& queue,int level){
  // level increments when the player passes a level
  for(int i = 0; i < level; i++){
    int led = random(3,8); // generates a random number between our 3 and 7 pins,
sets it to led
    buttonPrint(led); // buzzer will make a sound when leds light up
    queue.push(led); // pushes the random generated LED into the queue
    Serial.println();
    turnLightOn(led); // turns on the LED
  }
}

// buzzer makes a noise when a led lights up
void buttonPrint(int led){
  // led 5
  if(led == 3){
    Serial.println("fake 5");
  }
  // led 4
  if(led == 4){
    Serial.println("fake 4");
  }
  // led 3
  if(led == 5){
    Serial.println("fake 3");
  }
  // led 2
  if(led == 6){
    Serial.println("fake 2");
  }
  // led 1
  if(led == 7){
    Serial.println("fake 1");
  }
}

// helper function to make LED blink
void turnLightOn(int led){
  digitalWrite(led, HIGH); // turns on LED
  delay(500); // wait half a second
  digitalWrite(led, LOW); // turns off LED
  delay(500); // wait half a second
}

// when the player passes a level, turns on all the lights to indicate it's correct
void turnOnAllLights(){
  digitalWrite(led1,HIGH); // turns on led 1
  digitalWrite(led2,HIGH); // turns on led 2
  digitalWrite(led3,HIGH); // turns on led 3
  digitalWrite(led4,HIGH); // turns on led 4
  digitalWrite(led5,HIGH); // turns on led 5
  delay(2000); // wait 2 seconds
  digitalWrite(led1,LOW); // turns off led 1
  digitalWrite(led2,LOW); // turns off led 2
  digitalWrite(led3,LOW); // turns off led 3
  digitalWrite(led4,LOW); // turns off led 4
  digitalWrite(led5,LOW); // turns off led 5
  delay(3000); // wait 3 seconds before next level generates
}

// when player presses an incorrect button for the pattern, have all the lights
blink to indicate player is wrong
void wrong_input(){
  digitalWrite(led1, HIGH); // turns on led 1
  delay(75); // wait .075 seconds
  digitalWrite(led1, LOW); // turns off led 1
  digitalWrite(led2, HIGH); // turns on led 2
  delay(75); // wait .075 seconds
  digitalWrite(led2, LOW); // turns off led 2
  digitalWrite(led3, HIGH); // turns on led 3
  delay(75); // wait .075 seconds
  digitalWrite(led3, LOW); // turns off led 3
  digitalWrite(led4, HIGH); // turns on led 4
  delay(75); // wait .075 seconds
  digitalWrite(led4, LOW); // turns off led 4
  digitalWrite(led5, HIGH); // turns on led 5
  delay(75); // wait .075 seconds
  digitalWrite(led5, LOW); // turns off led 5
  lives--; // decrement the amount of lives when player guesses wrong
  Serial.println("scor 0"); // Serial communication for Arduino 3 - set score to 0
when wrong
}
```

arduino_2.ino

```
#include <LiquidCrystal.h>
#include "pitches.h"

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
```

```
// Life Indicator
const int led1 = 13; // LED 1 - set to pin 13
const int led2 = 10; // LED 2 - set to pin 10
const int led3 = 9; // LED 3 - set to pin 9
const int buzzer = 7; // buzzer - set to pin 7

int score = 0; // display our scoreboard
int ButtonState = 0; // reads hint button status
int lives = 3; // player starts with 3 lives, an LED turns off if player gets
pattern incorrect

void setup() {
  // put your setup code here, to run once:
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  pinMode(led1, OUTPUT); // set pin to led 1
  pinMode(led2, OUTPUT); // set pin to led 2
  pinMode(led3, OUTPUT); // set pin to led 3
  pinMode(buzzer, OUTPUT); // set pin to buzzer
  Serial.begin(9600);
  Serial.setTimeout(100);
}

void loop(){
  lcd.clear(); // clear the LCD screen
  lcd.setCursor(0,0);
  lcd.print("Score: ");
  lcd.print(score); // print our current score
  lcd.setCursor(0,1);
  lcd.print("Lives: ");
  lcd.print(lives); // display number of lives

  // Three LEDs are on to indicate 3 lives left
  if(lives == 3){
    digitalWrite(led1, HIGH); // on
    digitalWrite(led2, HIGH); // on
    digitalWrite(led3, HIGH); // on
  }

  // allows us to communiate from Arduino 1 to Arduino 2
  if(Serial.available() > 0){
    String serialInput = Serial.readString();
    checkInput(serialInput);
  }
}

// helper function - plays a noise when an LED or button is used
void playTune(String num){
  // button 1 is pressed
  if(num == "1"){
    tone(buzzer, NOTE_C4);
    delay(150);
    noTone(buzzer);
  }
  // button 2 is pressed
  if(num == "2"){
    tone(buzzer, NOTE_G3);
    delay(150);
    noTone(buzzer);
  }
  // button 3 is pressed
  if(num == "3"){
    tone(buzzer, NOTE_A3);
    delay(150);
    noTone(buzzer);
  }
  // button 4 is pressed
  if(num == "4"){
    tone(buzzer, NOTE_B3);
    delay(150);
    noTone(buzzer);
  }
  // button 5 is pressed
  if(num == "5"){
    tone(buzzer, NOTE_G2);
    delay(150);
    noTone(buzzer);
  }
}

// checks the type of input that was sent from Arduino 1 to Arduino 2
void checkInput(String input){
  String type = input.substring(0,4); // 4 letter string is checked and what was
inputted
  String num = input.substring(5,6); // which button number that was inputted

  // if the LED is used, buzzer plays a noise
  if(type == "fake"){
    playTune(num); // buzzer is used
  }

  // if a button is pressed, add a point to score and buzzer plays a noise
  if(type == "note"){
    score++; // increment the score
    playTune(num); // buzzer is used
  }

  // player presses an incorrect button
  if(type == "scor"){
    // sets the score back to 0 if the player gets pattern wrong
    if(num == "0"){
      lives--; // decrement the lives you have
      score = 0; // set score back to 0

      // two LEDs are on if player has 2 lives
      if(lives == 2){
        digitalWrite(led1, HIGH); // on
        digitalWrite(led2, HIGH); // on
        digitalWrite(led3, LOW); // off
      }
      // one LED is on if player has 1 life
      if(lives == 1){
        digitalWrite(led1, HIGH); // on
        digitalWrite(led2, LOW); // off
        digitalWrite(led3, LOW); // off
      }
      // no LEDs are on and it's game over if 0 lives
      if(lives == 0){
        digitalWrite(led1, LOW); // off
        digitalWrite(led2, LOW); // off
        digitalWrite(led3, LOW); // off
        lcd.clear(); // clear the LCD screen
        lcd.setCursor(0,0);
        lcd.print("Game Over!!!!!"); // display game over on the screen
        delay(5000);
        exit(0); // exit the code
      }
    }
  }
}
```

Pitches.h

```
/*************************************************
 * Public Constants
 *************************************************/

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
#define NOTE_FS3 185
#define NOTE_G3  196
#define NOTE_GS3 208
#define NOTE_A3  220
#define NOTE_AS3 233
#define NOTE_B3  247
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#define NOTE_DS4 311
```

```
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_CS5 554
#define NOTE_D5  587
#define NOTE_DS5 622
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_FS5 740
#define NOTE_G5  784
#define NOTE_GS5 831
#define NOTE_A5  880
#define NOTE_AS5 932
#define NOTE_B5  988
#define NOTE_C6  1047
#define NOTE_CS6 1109
#define NOTE_D6  1175
#define NOTE_DS6 1245
#define NOTE_E6  1319
#define NOTE_F6  1397
#define NOTE_FS6 1480
#define NOTE_G6  1568
#define NOTE_GS6 1661
#define NOTE_A6  1760
#define NOTE_AS6 1865
#define NOTE_B6  1976
#define NOTE_C7  2093
#define NOTE_CS7 2217
#define NOTE_D7  2349
#define NOTE_DS7 2489
#define NOTE_E7  2637
#define NOTE_F7  2794
#define NOTE_FS7 2960
#define NOTE_G7  3136
#define NOTE_GS7 3322
#define NOTE_A7  3520
#define NOTE_AS7 3729
#define NOTE_B7  3951
#define NOTE_C8  4186
#define NOTE_CS8 4435
#define NOTE_D8  4699
#define NOTE_DS8 4978
```

Arduino_3.ino

```
#include <SevSeg.h>
#include <Wire.h>
SevSeg sevseg; //Instantiate a seven segment object
```

```
#define TIMESIZE 2

float displayTimeSecs = 1; //how long do you want each number on display to show (in
secs)
float displayTime = (displayTimeSecs * 5000);
long startNumber = 0; //countdown starts with this number
long endNumber = 0; //countdown ends with this number
bool gameplay = false;

void setup() {
  byte numDigits = 4;
  byte digitPins[] = {2, 3, 4, 5};
  byte segmentPins[] = {6, 7, 8, 9, 10, 11, 12, 13};
  bool resistorsOnSegments = false; // 'false' means resistors are on digit pins
  byte hardwareConfig = COMMON_ANODE; // See README.md for options
  bool updateWithDelays = false;  // Default 'false' is Recommended
  bool leadingZeros = true; // Use 'true' if you'd like to keep the leading zeros
  bool disableDecPoint = true; // Use 'true' if your decimal point doesn't exist or
isn't connected. Then, you only need to specify 7 segmentPins[]

  sevseg.begin(hardwareConfig, numDigits, digitPins, segmentPins,
resistorsOnSegments,
  updateWithDelays, leadingZeros, disableDecPoint);
  sevseg.setBrightness(90);
  Wire.begin(4);
  Wire.onReceive(receiveEvent);
  Serial.begin(9600);
  Wire.onRequest(requestEvent);
  Wire.setTimeout(100);
  Serial.setTimeout(100);
}

void loop() {
  Wire.beginTransmission(4);
  Wire.write(200);
  Wire.endTransmission(4);
  while(gameplay){
    // keeps the timer counting down
    if(startNumber >= endNumber){
      for (long i = 0; i <= displayTime; i++){
        sevseg.setNumber(startNumber, 0); // set the start timer to 30
        sevseg.refreshDisplay();
      }
      startNumber--; // count down
    }
    if(startNumber == 59){
      Wire.beginTransmission(4);
      Wire.write(200);
      Wire.endTransmission(4);
    }
  }
  sevseg.setNumber(0000,0); //after countdown shows endNumber, show this number.
  sevseg.refreshDisplay();
}

void receiveEvent(int howMany){
  // reads in Wire communication
  while(Wire.available() > 0){
    int c = Wire.read(); // set the timer to an int
    Serial.println(c);
    delay(50);
    // set the initial timer to 30
    if(c == 60){
      startNumber = 60;
      gameplay = true;
    }
    // get out of the loop
    if(c == 100){
      startNumber = 100;
      gameplay = false;
    }
    // game over
    if(c == 101){
      exit(0);
    }
  }
}

void requestEvent(){
  Serial.println(startNumber);
  Wire.write(startNumber);
}
```