Milestone 6 CS 362 Team 50

Team Members:

- Stefan Vukovic
  - NetID: svukov2
  - E-mail: svukov2@uic.edu
- Jan Kobylarz
  - NetID: jkobyl4
  - E-mail: jkobyl4@uic.edu
- Mashel Al-Nimer
  - NetID: malnim3
  - E-mail: malnim3@uic.edu

RGB Infinity Mirror Clock with Gesture Motion Control

Abstraction of Project:

This project will incorporate multiple major features available to the user within an alarm clock. A constant pattern of led lights will be on, but a set of unique light patterns will replace it when the alarm goes off. The user will be able to turn the alarm off with motion and the buzzer will turn off. Users will also be able to change the time through an online website connected via Bluetooth. This will add the ability to change the timezone. The infrared remote controller will set alarms for the clock.

Description of Project:

Our goal is to create a working alarm clock. We would like to design it with rotating rgb LEDs that correlate with the time such as a real analog clock.

This alarm clock will allow the user to:

- ❖ Change the time based on their time zone though an online website
- ❖ Set custom animations for the alarm using a button
- ❖ Set alarms using a remote controller
- ❖ Turn the alarm off using a motion sensor

Inputs/Outputs:

Motion detection sensor will allow the user to turn the alarm off. Led lights will be used to display the time. A buzzer will go off when the time matches the alarm set by the user. A remote will be used to set an alarm (IR Remote Transmitter). A button will be used to switch the pattern that is displayed when an alarm is triggered. A lcd will display the name of the pattern that is selected to be displayed when an alarm goes off. A bluetooth module to connect your phone to the alarm clock to change time based on your time zone.

Expected Plan For Communication:

We will be using bluetooth to connect to the users cellular phone. We decided to use bluetooth so the user can use their mobile phone to change the time of the clock through a website. Using their phone, the user will be able to turn the time off with a simple click of a button. Additionally, the user will be able to set the desired alarm clock time using an infrared remote controller. This provides the ability of actual two way communication between the user and clock.

Description of Original Work Being Attempted:

Our design is unique to other LED alarm clocks since it will have motion sensors allowing the user to turn off the alarm. Our design will also allow the user to change the time through an online website just in case they move to another location with a different time zone.

How to Build Our Project:

1. Gather all of the materials listed in the list of materials section

2. Once you have all the materials, begin building your hardware using the diagrams provided
3. If needed use this starting resource point for both hardware and arduino code.
4. The coding portion of this project can be broken down into following steps:
    a. Build a working alarm clock
        i. Create the input functionality to input desired clock times
        ii. Match the alarm clock time with the current clock time which will result in a buzzer sounding off indicating alarm time has been reached
    b. Connect a motion detection sensor which will stop the buzzer sound.
    c. Connect an IR Remote Controller and implement the functionality of setting the desired alarm clock times, which was created when building your clock.
    d. Create a LED light pattern on your LED strip, and add in to the current project database. Make sure to have a specific light pattern indicating when a specific time is matched.
    e. For the bluetooth part of the project, a separate html creation is needed. Within this file you need two specific libraries p5.min.js and p5.ble.js. You will also need to create a bluetooth.js, bluetoothGUI.js and the index.html files. We have provided code to help you create your documents

User Guide:

- ❖ When trying to set an alarm:
    - ➢ You will need to use the remote provided in your arduino kit
    - ➢ Use numbers 0 through 9 to set the time
    - ➢ Use EQ to indicate am and ST/REPT to indicate pm

    - You have a limit of 10 alarms
- When trying to change time zones:
    - Connect your arduino to the index.html file created
    - Select the time zone you would like to switch to
- When trying to change the alarm design:
    - The current style will be displayed on the screen
    - Click the button on the side of the clock until you find the alarm design you would like to have set
    - The new style will display on the screen
- When trying to turn alarm off:
    - Wave your hand in front of the motion sensor and halts the buzzer.

Timeline of Development:

- Completed:
    - Week 6-7:
        - Developed our initial designs and sketches
        - Developed an initial project plan of what should be done at certain time
        - Researched for available tools and resources
    - Week 8-10:
        - Incorporated our initial codebase which includes:
            - Setting up initial alarm time
            - Matching the alarm time to the clock time (parsing times), in order to turn on the buzzer
    - Week 11-12:

        - Led Light patterns configured to each time so that a unique pattern matches a unique time
    - Week 13-14
        - Create custom alarm clock animations
    - Week 15:
        - Connect our Arduino Infinity Mirror Clock to webpage via Bluetooth
- Milestone 3 - Sketches and Pseudocode done, along with abstract and list of materials needed.
- Milestone 4 - Have the hardware setup and begin coding.
- Milestone 5 - Have the hardware and most of the code completed. This also includes having our Design Presentation done.
- Milestone 6 - Have the project done and finish testing functionality. This includes getting the final design document done. Testing our projects among all functionalities will be part of this milestone as well.
- Milestone 7 - Have the project be done, fully functional and ready for use.

List of Materials:

- LED rgb strip
- 1 Mirror
- 1 Glass
- front plastic panel
- Circular loop for led encasement
- Breadboard
- Remote

- Motion Detection Sensor
- Buzzer
- Resistors
- Button
- Potentiometer
- LCD
- IR Receiver
- HC-05 Bluetooth Module
- 10k ohm resistor
- Adafruit NeoPixel Digital RGB LED Strip - White 30 LED

What Didn't Work:

- Bluetooth is not working fully. There is no way of knowing when the arduino is connected to the bluetooth so we need to display something on the screen saying the bluetooth has been connected. This would give an indication of bluetooth being connected in some way, and it would fulfill its functionality.
- There is no case for the clock. We initially thought we had all the pieces to create a case but when creating the case we were missing a piece. Initially our thoughts were to have one piece of glass to serve our infinity mirror however, upon further discussion and looking at references another glass component would be needed to make it dual. To fix this issue we need to buy the piece that was missing (the glass).
- The animations were not displaying correctly. To fix this issue we need to debug the animation code more and figure out why it isn't displaying the way we want it to

References:

Adafruit NeoPixel Digital RGB LED Strip - White 60 LED. (n.d.). Adafruit. https://www.adafruit.com/product/1138?length=1

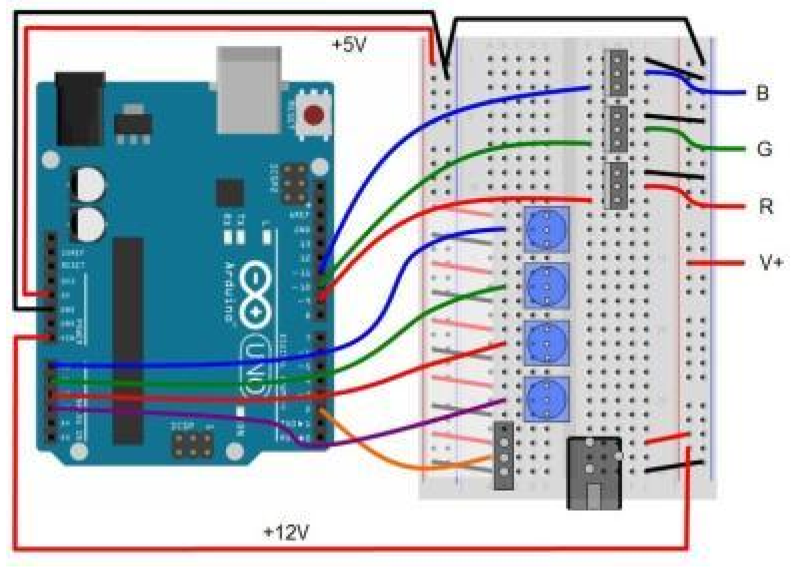
Arduino and HC-05 Bluetooth Module Complete Tutorial. (n.d.). HowToMechatronics. https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-05-bluetooth-module-tutorial/

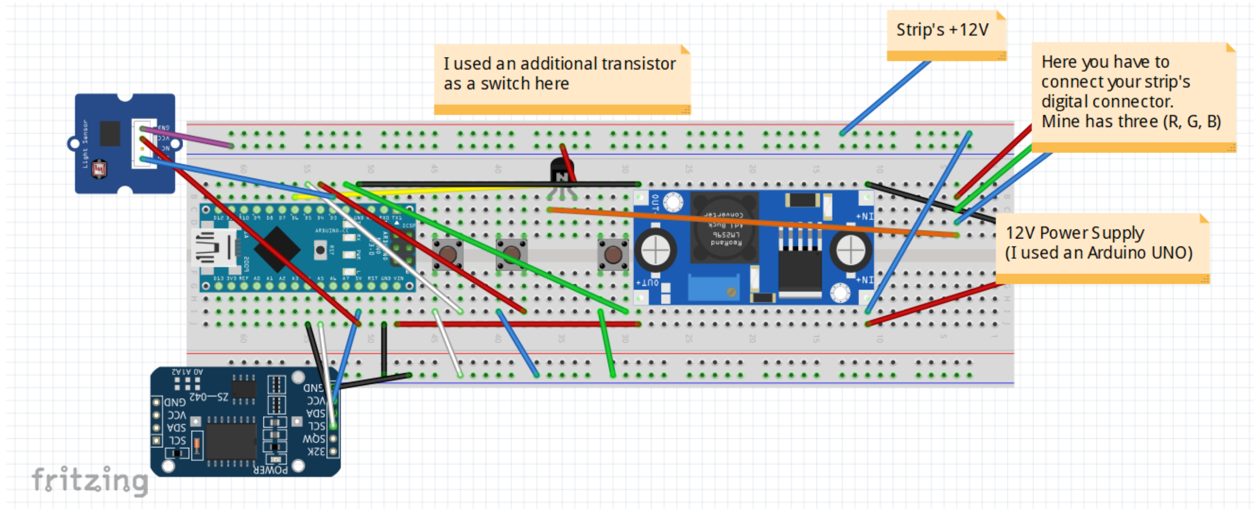
Darrah, Kevin, director. *Arduino Wireless Bluetooth from CHROME - EASY Tutorial! YouTube*, YouTube, 3 Oct. 2019, www.youtube.com/watch?v=w_mRj5IlVpg.

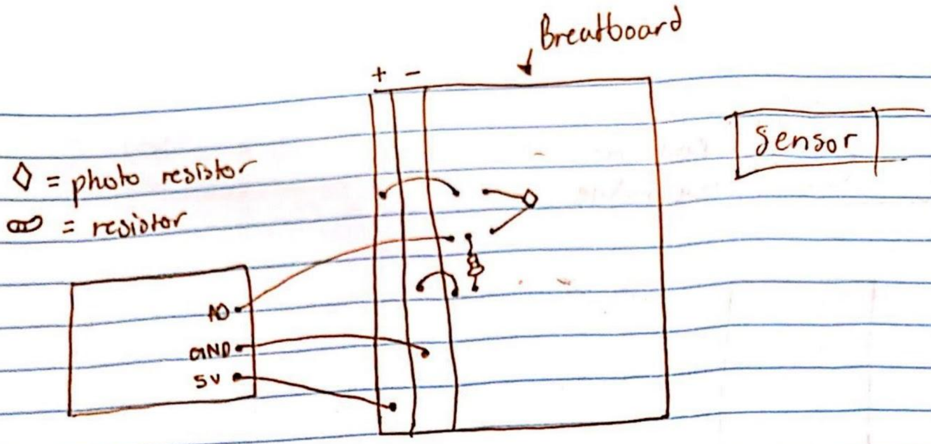
Infinity mirror clock. (n.d.). Arduino Project Hub. https://create.arduino.cc/projecthub/TheTNR/infinity-mirror-clock-b00c6a?ref=search&ref_id=Infinity%20mirror%20alarm%20clock%20+&offset=0

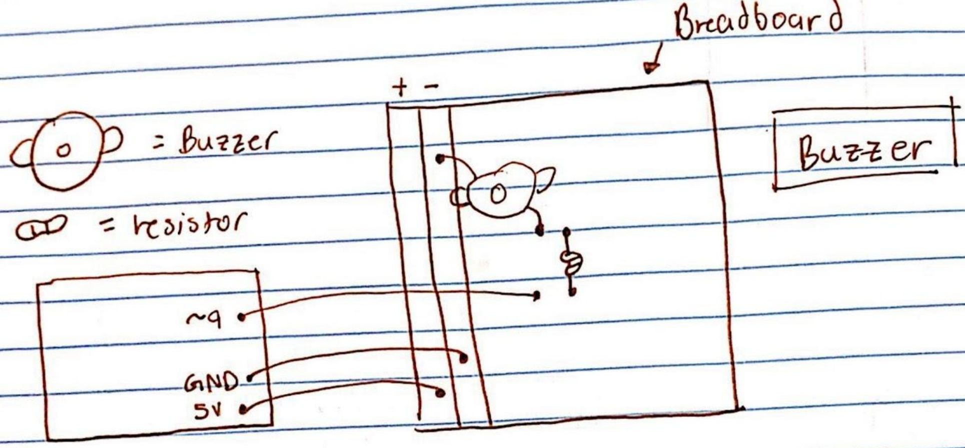
Use an IR Remote Transmitter and Receiver with Arduino. (n.d.). Arduino Project Hub. https://create.arduino.cc/projecthub/electropeak/use-an-ir-remote-transmitter-and-receiver-with-arduino-1e6bc8

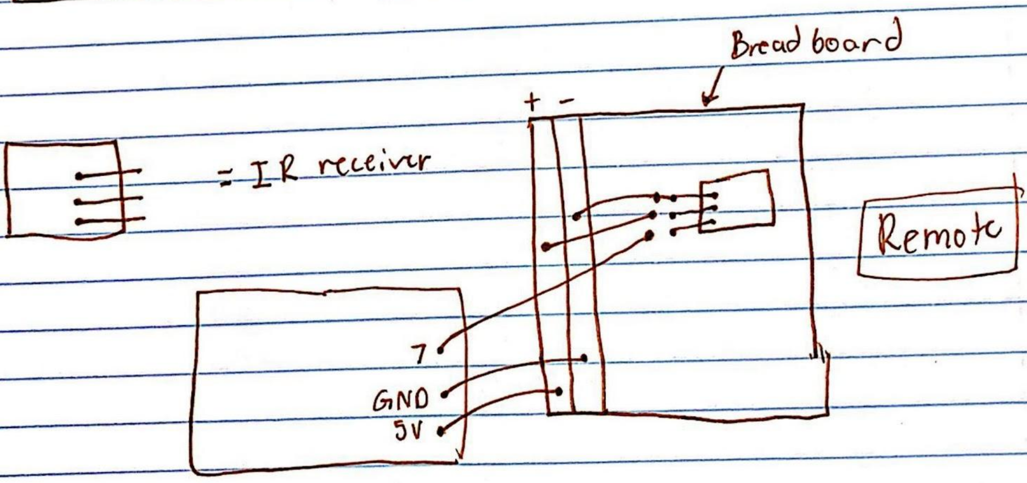
Diagram For Hardware:

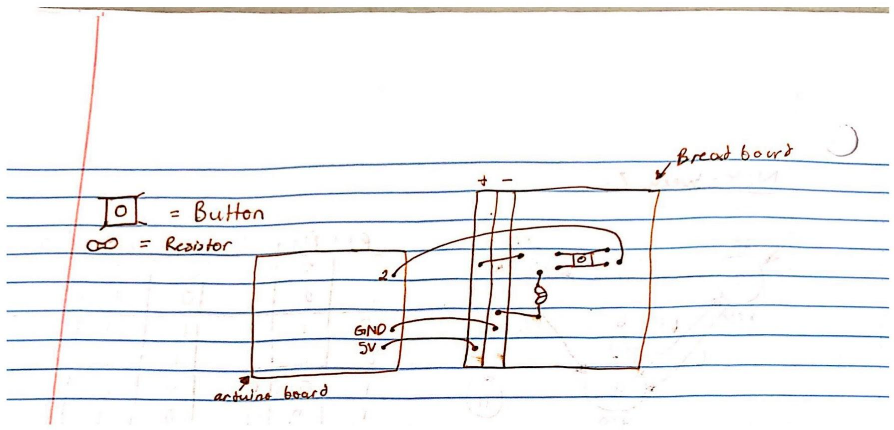



Strip's +12V
I used an additional transistor as a switch here
Here you have to connect your strip's digital connector. Mine has three (R, G, B)
12V Power Supply (I used an Arduino UNO)
fritzing


Breadboard
+ −
Sensor
◊ = photo resistor
= resistor
10
GND
5V


Breadboard
+ −
Buzzer
= Buzzer
= resistor
~9
GND
5V


Bread board
+ −
= IR receiver
Remote
7
GND
5V



Scanned with CamScanner

Code Sketch:

expoProject.ino:

```
#include <Vector.h>   // for vectors
#include <Wire.h>
#include <Adafruit_NeoPixel.h>
#include <stdio.h>
#include <IRremote.h>
#include <TimeLib.h>
#include <LiquidCrystal.h>
#include <SoftwareSerial.h> //For the bluetooth
LiquidCrystal LCD(12, 11, 5, 4, 3, 2);

using namespace std;

#define NEOPIN 13

#define STARTPIXEL 0
```

```
//These are the HEX values of the buttons being used
#define ONE_DAY   0xB946FF00 // VOL+ button (One occurrence of the alarm it will be
deleted once used)
#define EVERYDAY    0xB847FF00 // FUNCTION/STOP button (Everyday occurrence of the
alarm, will need to be deleted manually)
#define AM 0xE619FF00 // EQ button
#define PM 0xF20DFF00 // ST/REPT button
#define SLASH 0xBA45FF00 // SLASH button (for separating time)
#define ZERO 0xE916FF00
#define ONE 0xF30CFF00
#define TWO 0xE718FF00
#define THREE 0xA15EFF00
#define FOUR 0xF708FF00
#define FIVE 0xE31CFF00
#define SIX 0xA55AFF00
#define SEVEN 0xBD42FF00
#define EIGHT 0xAD52FF00
#define NINE 0xB54AFF00


 const int RECV_PIN = 11;       // IR Sensor pin
 int ledPin = 13;               // LED
 int pirPin = 12;               // PIR Out pin v
 int buzzer = 10;                // buzzer pin
 int pirStat = 0;               // PIR status

 //Setting variables needed for bluetooth
 uint8_t RXPin = 3;
 uint8_t TXPin = 2;
 SoftwareSerial bluetooth(TXPin, RXPin);

 //String alarmsSet[10]; //An array that will hold all the alarms (max of 10 alarms at once)
 Vector<String> alarmsSet;  //An vector that will hold all the alarms (max of 10 alarms at once)
 Vector<int> alarmHours;    //vector of parsed hours for alarm
 Vector<int> alarmMinutes;  //vector of parsed minutes for alarm
 Vector<String> alarmOc;  //vector of parsed occurence for alarm
 //Vector<int, int, String> conAlarm;
 int alarmNum = 0;
 int amPm = 0; //If 0 then not set yet, if 1 then am, if 2 then pm
 String totalClicked; //All the buttons clicked before am/pm clicked
 String curClick; //The button that just got clicked


//--------------------------------------------------------------------------------------------------------------
-----------------
```

```
  //Initialize global variables for time and date of the clock
  int months = 0;  // for settime
  int days = 0;   // for settime
  int years = 0;  // for settime
  int hours = 0;
  int minutes = 0;
  int seconds = 0; // for settime
  String occurrence = "";
  // Need to change these to vectors
  int parseAlarmsHours[10];
  int parseAlarmsMinutes[10];
  int parseAlarmsOccurence[10];


  // Prompt for user to enter the correct format of the string
  String Time = "Enter the time in this format:  hh/mm.";     // Will be manually entered into
Arduino ide everytime the clock needs to be turned on
  String Alarm = "Enter the alarm in this format: occurence/hh/mm. (OD = one day, ED =
everyday)"; // EX: OD/hh/mm       ED/hh/mm

  // For reading the strings and parsing it
  String myTime = "4/22";
  String myAlarm;

  //Keeps track of what type of zone we are currently in and the offset we need to add to current
time
  //Default is central
  String currTimeZone = "CENTRAL";
  int timeZoneOffset = 0; //difference in hour

//--------------------------------------------------------------------------------------------------------------
-----------------

void setup()
{
   pinMode(NEOPIN, OUTPUT);
   pinMode(pirPin, INPUT);
   LCD.begin(16, 2);
   Serial.begin(9600);
   IrReceiver.begin(RECV_PIN);
   pinMode(buzzer, OUTPUT);

  //Starting bluetooth
  bluetooth.begin(9600);

     strip.begin();
  //strip.show(); // Initialize all pixels to 'off'

  strip.setBrightness(DAYBRIGHTNESS); // set brightness

  // startup sequence
  delay(500);
  colorWipe(strip.Color(255, 0, 0), 20); // Red
  colorWipe(strip.Color(0, 255, 0), 20); // Green
  colorWipe(strip.Color(0, 0, 255), 20); // Blue
  delay(500);

}

//------------------------------------------------------------------------------------------------------------
-----------------
// Start of expoCode and my modified Lab6 Code
// loop also acts as a main function
void loop()
{
  //Checks to see if something was clicked on webpage
  readBluetooth();

  //Setting up alarm
  if(IrReceiver.decode()){
   if(alarmNum == 10){
    Serial.println("Sorry you already set 10 alarms!");
    IrReceiver.resume();
    }else{
      //Checks and converts the HEX to string
      switch(IrReceiver.decodedIRData.decodedRawData){
        case ONE_DAY:
        amPm = 1;
        curClick = "OD";
          //Serial.println(curClick);
          break;
        case EVERYDAY:
        amPm = 2;
        curClick = "ED";
          //Serial.println(curClick);
          break;
        case ZERO:
        curClick = "0";
          //Serial.println(curClick);
          break;
        case ONE:
        curClick = "1";
          //Serial.println(curClick);
          break;
        case TWO:
        curClick = "2";
          //Serial.println(curClick);
          break;
        case THREE:
        curClick = "3";
          //Serial.println(curClick);
          break;
        case FOUR:
        curClick = "4";
          //Serial.println(curClick);
          break;
        case FIVE:
        curClick = "5";
          //Serial.println(curClick);
          break;
        case SIX:
        curClick = "6";
          //Serial.println(curClick);
          break;
        case SEVEN:
        curClick = "7";
          //Serial.println(curClick);
          break;
        case EIGHT:
        curClick = "8";
          //Serial.println(curClick);
          break;
        case NINE:
        curClick = "9";
          //Serial.println(curClick);
          break;
        case SLASH:
          curClick = "/";
          //Serial.println("PM");
          break;
        }
      totalClicked.concat(curClick);
      Serial.println(totalClicked);

      //If the user selects ED/OD then it stores the time in the array
      if(amPm != 0){
        Serial.println("Adding to vector");
        alarmsSet.PushBack(totalClicked);
```

```
      Serial.println(totalClicked);
      amPm=0;
      totalClicked="";
      ++alarmNum;
    }
    IrReceiver.resume();
    }
  }else{
      Serial.println("Nothing was clicked");
      IrReceiver.resume();
    }
 //End of alarm function
 delay(1000);

  // If alarm size > 0 check it's validity
if (alarmsSet.Size() > 0)
{

  //Store user inputted Alarm, and check it's validity
  int i = 0;
  int n = alarmsSet.Size();
  while(i < n)
  {
    myAlarm = alarmsSet[i];
    int sizeAlarm = myAlarm.length();
    // Invalid Alarm entered
    if (sizeAlarm > 8)
    {
      Serial.println("Not A Valid Alarm");
      //LCD.setCursor(3, 0);
      //LCD.print("Invalid Alarm. Try Again! ");
      alarmsSet.Erase(i);
    }
    else
    {
      checkAlarm(myAlarm, i);
    }
      Serial.println(myAlarm);
      i++;
  }
}

// Wait for input
Serial.println(Time);
while (1)
{
    if (Serial.available() == 1)
    {
      break;
    }
  }

  // If input recieved for time
  if (Serial.available() == 1)
  {

    //Store user inputted Time, and check it's validity
    myTime = Serial.readString();
    int sizeTime = myTime.length();
    checkTime(myTime);
    Serial.println(myTime);


     // Invalid Time entered
    if (sizeTime > 9)
    {
      Serial.println("Not A Valid Time");
      LCD.setCursor(3, 0);
      LCD.print("Invalid Time. Try Again! ");
    }

    // Set the time using the setTime function
    setTime(hours, minutes, seconds, days, months, years);
    clock();
    //alarm();
    animationTime(hours, minutes, seconds, days, months, years);
  }



}
//------------------------------------------------------------------------------------------------------------------
-----------------
//Conversion function
//Based on the current time zone and the new time zone, we have functions which tell us the
difference between
//of the 2 time zones(in hours) so we can correctly change the time
void timeZoneConversion(String newZone){
  if(currTimeZone == "EASTERN"){
    if(newZone == "CENTRAL"){
      timeZoneOffset = -1;
    }else if(newZone == "MOUNTAIN"){
      timeZoneOffset = -2;
    }else if(newZone == "PACIFIC"){
      timeZoneOffset = -3;
    }
  }else if(currTimeZone == "CENTRAL"){
    if(newZone == "EASTERN"){
      timeZoneOffset = 1;
    }else if(newZone == "MOUNTAIN"){
      timeZoneOffset = -1;
    }else if(newZone == "PACIFIC"){
      timeZoneOffset = -2;
    }
  }else if(currTimeZone == "MOUNTAIN"){
    if(newZone == "CENTRAL"){
      timeZoneOffset = 1;
    }else if(newZone == "EASTERN"){
      timeZoneOffset = 2;
    }else if(newZone == "PACIFIC"){
      timeZoneOffset = -1;
    }
  }else if(currTimeZone == "PACIFIC"){
    if(newZone == "CENTRAL"){
      timeZoneOffset = 2;
    }else if(newZone == "MOUNTAIN"){
      timeZoneOffset = 1;
    }else if(newZone == "EASTERN"){
      timeZoneOffset = 3;
    }
  }
}

//---------------------------------------------------------------------------------------------------------------
-----------------
//Bluetooth function
void readBluetooth(){
  char buff[100];
  if(bluetooth.avaliable() > 0){
    int bytesGiven = bluetooth.readyBytesUntil('\n', buff, 99);
    buff[bytesGiven] = NULL;

    if (strstr(buff, "PACIFIC") == &buff[0]) {
      //Need to add the offset of pacific zone to current time zone
      timeZoneConversion("PACIFIC");
      currTimeZone = "PACIFIC";
    }else if (strstr(buff, "MOUNTAIN") == &buff[0]) {
      //Needs to add offset of mountain zone to current time zone
      timeZoneConversion("MOUNTAIN");
      currTimeZone = "MOUNTAIN";
    }else if (strstr(buff, "CENTRAL") == &buff[0]) {
      //Need to add offset of central to current time zone
      timeZoneConversion("CENTRAL");
      currTimeZone = "CENTRAL";
    }else if(strstr(buff, "EASTERN") == &buff[0]){
      //Needs to add offset of eastern to current time zone
      timeZoneConversion("EASTERN");
      currTimeZone = "EASTERN";
    }
  }
}

//-----------------------------------------------------------------------------------------------------------------
-----------------
//Function for Alarm validation.
void checkAlarm(String Alarm, int index)
{

  //Local String variables for parsing
  String parseOccurrence;
  String parseMinutes;
  String parseHours;


  //Parse through Alarm for Occurrence, minutes, and hours
  parseOccurrence = Alarm.substring(0, Alarm.indexOf("/"));

  //Parse through Time for Minutes
  parseMinutes = Alarm.substring(5, Alarm.indexOf("/") + 1);
  minutes = parseMinutes.toInt();

  //Parse through Time for Seconds
  parseHours = Alarm.substring(6);
  hours = parseHours.toInt();

  // INVALID VALUES for HOURS AND MINUTES
  if ((hours < 0) || (hours > 24) || (minutes < 0) || (minutes > 59) || occurrence != "OD" ||
occurrence != "ED")
  {
    Serial.println("Invalid Alarm. Removing it. Try Again !");
    Serial.println(Alarm);
    alarmsSet.Erase(index);
    //LCD.setCursor(3, 0);
    //LCD.print("Invalid Alarm. Try Again! ");
    /*
    while (1)
    {
      if (Serial.available() == 1)
      {
        break;
      }
    }
    if (Serial.available() == 1)
    {
      myAlarm = Serial.readString();
      checkAlarm(myAlarm);
      Serial.print(hours);
      Serial.print(":");
      Serial.print(minutes);
      Serial.print(":");
      Serial.println(occurrence);
    }
    */
  }
  else
  {
    alarmHours.PushBack(hours);
    alarmMinutes.PushBack(minutes);
    alarmOc.PushBack(parseOccurrence);
  }

}

//------------------------------------------------------------------------------------------------------------------------
-----------------
//Function for Time validation.
void checkTime(String Time)
{

  //Local String variables for parsing
  String parseHours;
  String parseMinutes;

  //Parse through Time for Hours
  parseHours = Time.substring(0, Time.indexOf("/"));
  hours = parseHours.toInt();

  //Parse through Time for Minutes
  parseMinutes = Time.substring(5, Time.indexOf("/") + 1);
  minutes = parseMinutes.toInt();
  // INVALID VALUES for DATE AND TIME
  if ((hours < 0) || (hours > 23) || (minutes < 0) || (minutes > 59))
  {
    Serial.println("Invalid Time. Try Again !");
    Serial.println(Time);
    LCD.setCursor(3, 0);
    LCD.print("Invalid Date. Try Again! ");
    while (1)
    {
      if (Serial.available() == 1)
      {
        break;
      }
    }
    if (Serial.available() == 1)
    {
      myTime = Serial.readString();
      checkTime(myTime);
      Serial.print(hours);
      Serial.print(":");
      Serial.print(minutes);

    }
  }
}




//------------------------------------------------------------------------------------------------------------
-----------------
// Updates Time

void clock()
{
  while (1)
  {
    time_t t = now();
    // delay is 1 second like a real clock
    delay(1000);

    //printClock(minute(t), second(t));
    // add alarm here once you implement the vector global variable  // Wait for input
    // Alarm code
    int i = 0;
    int n = alarmHours.Size();
    while(i < n)
    {
      int myHour = alarmHours[i];
      int myMinutes = alarmMinutes[i];
      String myOc = alarmOc[i];
      if(myHour == hour(t) && myMinutes == minute(t))
      {
          pirStat = digitalRead(pirPin);
          if (pirStat == LOW)
          {
            tone(buzzer, 1);               // if Alarm detected
          }
          else if (pirStat == HIGH)
          {
            noTone(buzzer);                 // turn off buzzer
          }
      }

    }
  }

}


//--------------------------------------------------------------------------------------------------------------
-----------------
// Once Alarm and time has been validated check if any of the current alarms match current time
/*
void alarm()
{
    Vector<int> alarmHours;    //vector of parsed hours for alarm
  Vector<int> alarmMinutes;  //vector of parsed minutes for alarm
  Vector<String> alarmOc

}
*/
//--------------------------------------------------------------------------------------------------------------
-----------------
```

```
//---------------------------------------------------------------------------------------------------------------
-----------------
void animationTime(int hours, int minutes, int seconds, int days, int months, int years)
{


      tinfo.tm_year = years;
        tinfo.tm_mon = months;
        tinfo.tm_mday = days;
        tinfo.tm_hour = hours;
        tinfo.tm_min = minutes;
        tinfo.tm_sec = seconds;

          // Convert time structure to timestamp
        initialt = timelib_make(&tinfo);

        // Set system time counter
        timelib_set(initialt);


        if (millis() - last > 1000) {
          // Keep track of the time we last sent data to serial monitor
          last = millis();
          // Get the timestamp from the library
          now = timelib_get();
          // Convert to human readable format
          timelib_break(now, &tinfo);
          // Send to serial port
          Serial.print(tinfo.tm_hour);
          printDigits(tinfo.tm_min);
          printDigits(tinfo.tm_sec);
          Serial.print(" ");
          Serial.print(tinfo.tm_mday);
          Serial.print(" ");
          Serial.print(tinfo.tm_mon);
          Serial.print(" ");
          Serial.print(tinfo.tm_year);
          Serial.println();
        }
        byte secondval = tinfo.tm_sec;   // get seconds
        byte minuteval = tinfo.tm_min;  // get minutes
        int hourval = tinfo.tm_hour;   // get hours

        // change brightness if it's night time
        // check less often, once per minute
        if (secondval == 0) {
```

```
    if (hourval < MORNINGCUTOFF || hourval >= NIGHTCUTOFF) {
      strip.setBrightness(NIGHTBRIGHTNESS);
    } else {
      strip.setBrightness(DAYBRIGHTNESS);
    }
  }

  hourval = hourval % 12; // This clock is 12 hour, if 13-23, convert to 0-11`

  hourval = (hourval * 60 + minuteval) / 24; //each red dot represent 24 minutes.

  secondval = (secondval) / 2; //each red dot represent 24 minutes.

  // arc mode
  for (uint8_t i = 0; i < strip.numPixels(); i++) {

    if (i <= secondval) {
      // calculates a faded arc from low to maximum brightness
      pixelColorBlue = (i + 1) * (255 / (secondval + 1));
      //pixelColorBlue = 255;
    }
    else {
      pixelColorBlue = 0;
    }

    if (i <= minuteval) {
      pixelColorGreen = (i + 1) * (255 / (minuteval + 1));
      //pixelColorGreen = 255;
    }
    else {
      pixelColorGreen = 0;
    }

    if (i <= hourval) {
      pixelColorRed = (i + 1) * (255 / (hourval + 1));
      //pixelColorRed = 255;
    }
    else {
      pixelColorRed = 0;
    }

    strip.setPixelColor((i + STARTPIXEL) % 60, strip.Color(pixelColorRed,
pixelColorGreen, pixelColorBlue));
  }

  //display
  strip.show();

  // printTheTime(theTime);

  // wait
  delay(100);

}

void printDigits(int digits)
{
  // utility function for digital clock display: prints preceding colon and leading 0
  Serial.print(":");
  if (digits < 10)
    Serial.print('0');
  Serial.print(digits);
}

timelib_t time_provider()
{
  // Prototype if the function providing time information
  return initialt;
}

// Fill the dots one after the other with a color
void colorWipe(uint32_t c, uint8_t wait) {
  for (uint16_t i = 0; i < strip.numPixels(); i++) {
    //strip.setPixelColor(i, c);
    strip.setPixelColor((i + STARTPIXEL) % 60, c);
    strip.show();
    delay(wait);
  }
}
```

commands.js:

```
function zonePacific() {

    sendData("PACIFIC\n");

}

function zoneMountain() {

    sendData("MOUNTAIN\n");

}

function zoneCentral() {

    sendData("CENTRAL\n");

}

function zoneEastern() {

    sendData("EASTERN\n");

}
```

Bluetooth.js:

bluetoothGUI.js:

```
//let blueToothCharacteristic;

let receivedValue = "";

let blueTooth;

let isConnected = false;

function setup() {



  createCanvas(windowWidth, windowHeight);



  // Create a p5ble class

  console.log("setting up");

  blueTooth = new p5ble();



  //Creating button which will allow connection of arduino

  const connectButton = createButton('Connect');
```

```
connectButton.mousePressed(connectToBle);

connectButton.position(15, 15);



//Creating buttons for time zones Pacific, Mountain, Central, and Eastern.

const pacificButton = createButton('Pacific');

pacificButton.mousePressed(zonePacific);

pacificButton.position(15, 60);



const mountainButton = createButton('Mountain');

mountainButton.mousePressed(zoneMountain);

mountainButton.position(pacificButton.x+pacificButton.width+10, 60);



const centralButton = createButton('Central');

centralButton.mousePressed(zoneCentral);

centralButton.position(mountainButton.x+mountainButton.width+10, 60);



const easternButton = createButton('Eastern');
```

```
        easternButton.mousePressed(zoneEastern);

        easternButton.position(centralButton.x+centralButton.width+10, 60);

      }
```

index.hmtl

```
<html>

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">



  <script language="javascript" type="text/javascript" src="libraries/p5.min.js"></script>

  <script language="javascript" type="text/javascript" src="libraries/p5.ble.js"></script>

  <script language="javascript" type="text/javascript" src="bluetooth.js"></script>

  <script language="javascript" type="text/javascript" src="commands.js"></script>

  <script language="javascript" type="text/javascript" src="bluetoothGUI.js"></script>



  <script src="p5.ble.min.js" type="text/javascript"></script>

  <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.7.2/addons/p5.dom.min.js"></script>
  <style> body { padding: 0; margin: 0; } </style>

</head>

<body>

</body>

</html>
```