**CPEN 291 2016W2  
Lab 5**

*Lab #: 5 Lab section:L2A Team #: 5B*

*Student name Student number Contribution percentage*

Yuhao Huang 55562152 33.3%

Yuxiang Huang 14605159 33.3%

Ziqiao Lin 10668168 33.3%

*Contribution summary*:

*Yuhao Huang:* Code for Arduino; schematic in Fritzing for Arduino; circuit layout and wiring for Arduino; part of the lab report.

*Yuxiang Huang:* Code for Processing; part of the lab report.

*Ziqiao Lin:* Code for Raspberry Pi; schematic in Fritzing for Raspberry; circuit layout and wiring for Raspberry PI; part of lab report.

**A. Introduction and motivations**

This report discusses a project to implement a radar system with a HC-SR04 ultrasonic range finder sensor and a servo motor. The sensor rotates with the servo motor arm to sense 180 degree of its surrounding, and measures the distance from the ambient object to the sensor itself. The system also includes a Processing user interface, and is able to send the sensor info via internet using Twitter. During this lab, we also learned about Raspberry Pi. The task was to control the 7-segment LED with the Raspberry Pi to count from 0 to 9 repeatedly. This report presents the procedures for the project, and a conclusion and analysis of reflections.

**B. Lab Description**

1. **Arduino:** This part of the lab consists of 2 subparts: measuring the distance from ambient objects with HC-SR04 ultrasonic range finder sensor, and making the sensor to turn from 0 to 180 degree and back to 0 with a servo motor. On the hardware level, we connected the servo motor and the ultrasonic sensor onto the Arduino board according to the instructions in lecture slides and datasheets. We also used an LM35 temperature sensor in order to measure ambient temperature to determine the speed of sound, thus improving measurement precision of the distance.
2. **HC-SR04 Ultrasonic Range Finder Sensor:** We implemented a separate function *calculateDistance* to calculate the distance from an object to the sensor. This function first reads from the LM35 temperature sensor by calling *read\_LM35\_Temp,* a function we wrote in the last lab. It then calculates the speed of sound with the ambient temperature, and the time it takes for sound wave to travel 1 cm round trip in ms/cm. After that the sensor is triggered, and the sound wave travel time is read by function *pulseIn,* and the distance is calculated from travel time and time per centimeter. This function finally returns the distance to the main program. In the main program, function *calculateDistance* is called once for each degree the servo motor is in, and degree and distance data is also printed in the serial monitor. We set the best performance range of the sensor to be 2 to 200 cm (approximately 0 to 7 inch) according to the datasheet. As a result, if the distance calculated is beyond this range, a special message “Out of range/no object” is printed.
3. **Servo Motor:** The servo motor is used here to carry the sensor. We initialized a servo motor at the very beginning of the program and attached it to digital pin 10. In the main program, we first set the servo motor to the beginning point (0 degree), and then start to rotate its arm from 0 degree to 180 degree, and then back to 0, in two for loops. In each cycle of the for loop, the distance data is calculated and printed.
4. **Twitter and Processing:** The second part of this lab is about using Processing to design a graphical user interface, as well as to enable the device to send twitter warning message if there is an object in range.
5. **Twitter:** In order to send twitter message, we first created a twitter application, and got the authorization information (e.g. consumer key, consumer secret) for the app. Then, we used the twitter4j library to implement the functions that update our twitter status. Specifically, we check if there is an object approaching, and if an object is detected, a twitter message will be sent every a few seconds (the interval is defined by the user) using the function “twitter.updateStatus”.
6. **Processing:** In terms of the processing GUI, we used a radar shape to show the real time object distance, with green lines showing the normal condition, and the red lines showing the object in range. In addition, we implemented 3 buttons (start, stop, and refresh) and 2 sliders (twitter\_interval and sweeping\_speed) in enable user interaction with the Arduino and Processing. By pressing start, stop and refresh buttons, the user is able to start/stop the servo, and refresh the radar screen. By moving the twitter\_interval slider, the user is able to change the interval between which a twitter message will be sent, if an object is detected; and by moving the sweeping\_speed slider, the user can adjust the speed of the servo. The interactions strongly improves the functionality of this application as well as the user experience.
7. **Raspberry Pi:** This part of the lab makes 7-seg led display from 0 to 9 continuously. On the hardware level, we connect T-cobbler and 7-seg LED on the breadboard. Then we connect pin1 and pin6 to 5v pin of T-cobbler, pin A,B,C,DP,D,E,G,F which are in series with 330 ohms connected with GIPO 21,20,24,23,6,13,19,26 separately. For coding part, firstly we import time and GPIO libraries. Next we set up 7-seg leds and we use LedA, LedB, LedC, LedD, LedE, LedF, LedE and LedDP to represent every certain led. We also use an array with size 10 named “number” to store 10 hexadecimal numbers, which contains 8 bits to represent each led whether turn on or off. We define a method called counter. In this method, we first have an infinite while loop to make sure it can still counter even the counter reaches 9. Inside this while loop, we have a for loop and define x in range of 10 and call the method port. Port method will accept the number from array and determine which led will be turned on or off by if statement.

**C. Conclusions and Reflections**

Overall the whole lab goes well and our circuit looks compact and organized. Raspberry Pi, Arduino, and Processing GUI all work as desired, and the twitter messages are regularly sent. Having the experience from the previous lab, we found good online resources about designing radar interface more quickly, and designed the processing GUI more easily. However, we still encountered some problems in this lab. First, we noticed that the reading from the ultrasonic sensor was not accurate; we tried to change several parameters, and finally found that the imprecision was due to the excessively short delay time. Initially, we used “delayMicroseconds(10)” and “delayMicroseconds(2)” for measuring distance with the sensor; after we changed the delay to 30 and 60 microseconds, the readings became much more accurate. Second, we noticed that the temperature value on the Processing GUI kept changing; to solve this problem, we converted the actual temperature reading to type “int” in Processing, and we also added a filter function inside the Arduino function “read\_LM35\_Temp()”; please look at the code for more detailed information.

**D. References**

Radar system: <http://howtomechatronics.com/projects/arduino-radar-project/>

Raspberry Pi: <https://www.raspberrypi.org/learning/physical-computing-guide/test-led-python/>

Display 7-seg LED: <https://www.youtube.com/watch?v=66ZlukzPDgE>

<https://www.youtube.com/watch?v=RzdxfCg_jHo>

<https://www.youtube.com/watch?v=D2e8kz6sL0Y>

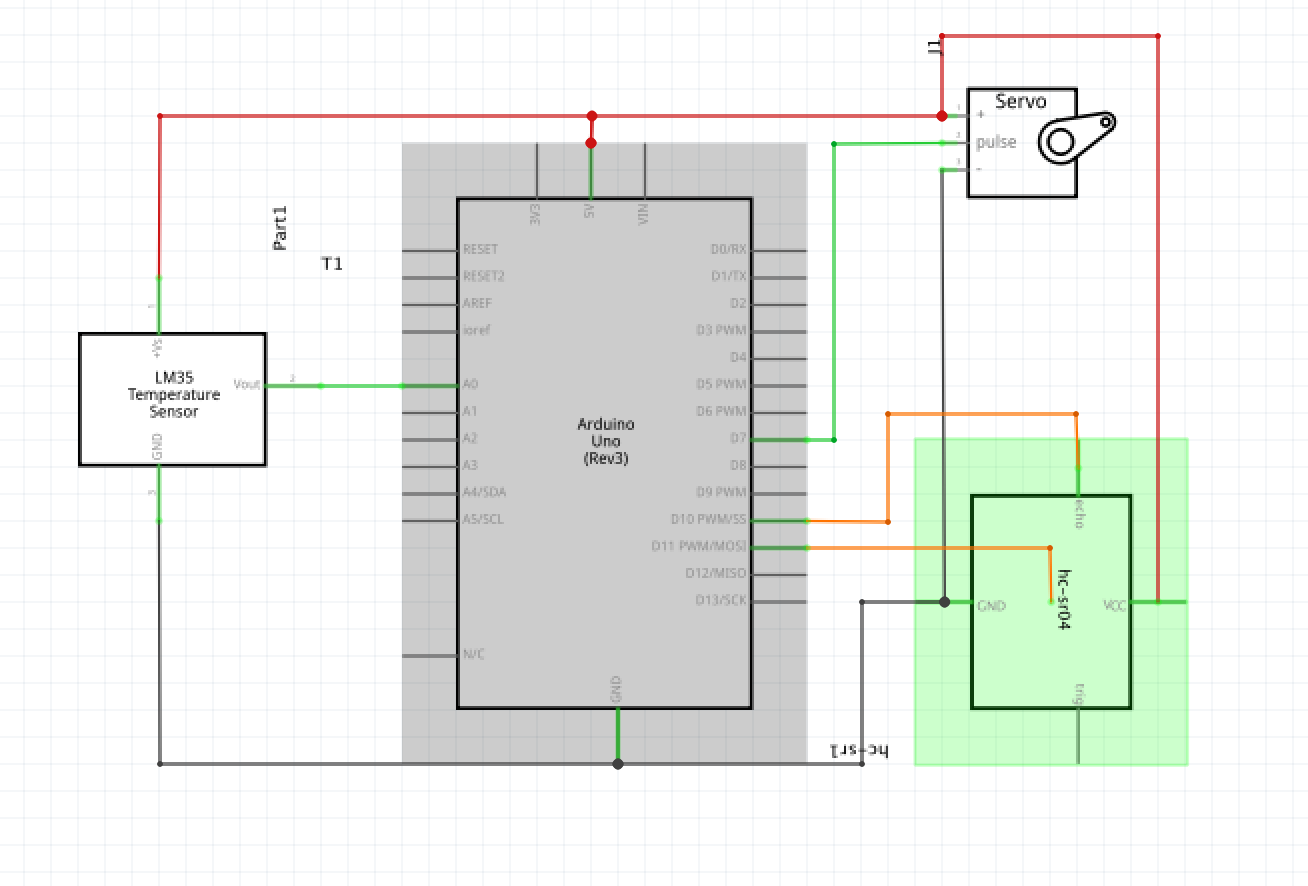
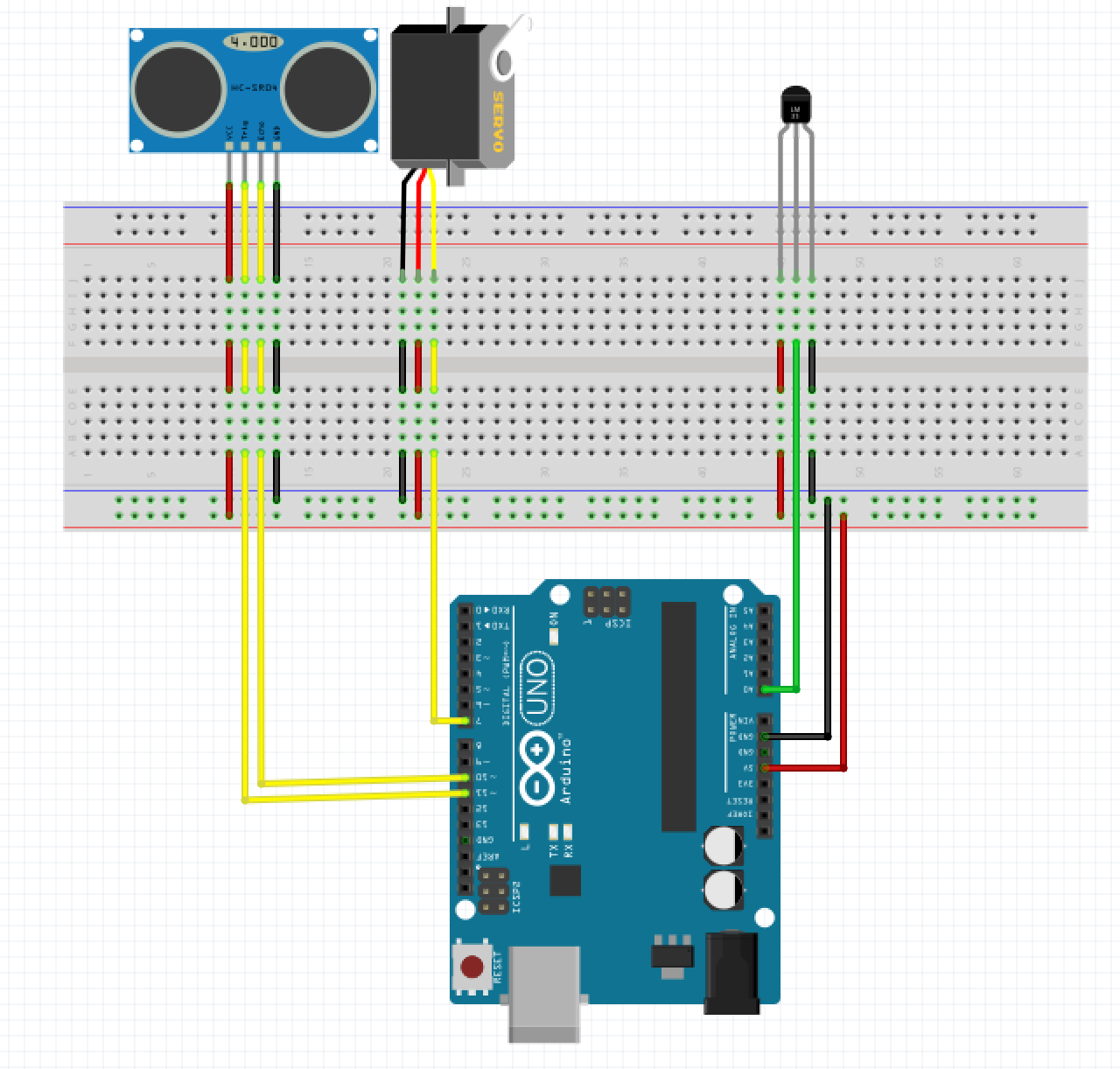
Python Tutorial: <https://www.zhihu.com/question/21123544>

<http://www.pythonforbeginners.com/comments/comments-in-python>

<http://raspi.tv/2015/how-to-drive-a-7-segment-display-directly-on-raspberry-pi-in-python>

<https://www.youtube.com/watch?v=9LgyKiq_hU0>

**Appendix I**



**Appendix II**

/\*Name: Yuhao Huang

Date: Feb 8th 2017

Purpose: Lab 5 Arduino part. This program controls the servo motor and ultrasonic range finder sensor.

The servo motor is set to the beginning point at the start of the loop, and then start to turn

from 0 degree to 180 degree. As the servo motor turns, the sensor measures distance for every

degree. The degree and distance data is printed in the serial monitor.

\*/

#include <Servo.h>.

// Defines Tirg and Echo pins of the Ultrasonic Sensor

const int trigPin = 7; //Define pin numbers of the ultrasonic sensor and the temperature sensor.

const int echoPin = 6;

const int LM35Pin = A0;

// Variables for the duration and the distance

long duration; //sound wave travel time in microseconds.

float distance;

float LM35temp; //Calculated temperature read from the LM35 sensor in degree Celsius.

float LM35reading; //Original data read from the temperature sensor.

float SpeedOfSound; //Calculated speed of sound.

float time\_cm; //Calculated time needed for sound wave to travel 1 cm round trip in μs.

int input = 0;

int stateFlag = 1;

int servoDelayTime = 30;

Servo myServo; //Initialize a servo motor.

void setup() {

pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output

pinMode(echoPin, INPUT); // Sets the echoPin as an Input

Serial.begin(9600);

myServo.attach(10); // Defines on which pin is the servo motor attached

}

void loop() {

//Check the input from Processing via the Serial port

if(Serial.available() > 0){

input = Serial.read();

}

//Excecute different commands according to the values

//If input us between 5 and 45, adjust the delay time of the servo rotation

if(input >= 5 && input <= 45){

servoDelayTime = 60 - input;

}else if(input > 0){

//If the input is 1, change stateFlag to 1

stateFlag = 1;

}else if(input == 0){

//If the input is 1, change stateFlag to 1

stateFlag = 0;

}

//If the stateFlag is 1, start the program

if(stateFlag == 1){

// rotates the servo motor from 15 to 165 degrees

for(int i=10;i<=170;i++){

myServo.write(i);

delay(servoDelayTime);

distance = calculateDistance();// Calls a function for calculating the distance measured by the Ultrasonic sensor for each degree

Serial.print(i); // Sends the current degree into the Serial Port

Serial.print(","); // Sends addition character right next to the previous value needed later in the Processing IDE for indexing

Serial.print(distance); // Sends the distance value into the Serial Port

Serial.print(","); // Sends addition character right next to the previous value needed later in the Processing IDE for indexing

Serial.print(LM35temp);

Serial.println();

//Check if there are new incoming bytes; if there is a 0 is received, pause the program

if(Serial.available() > 0){

input = Serial.read();

if(input == 0){

stateFlag = 0;

//keep checking if there are new incoming bytes; if there is a 1 is received, resume the program

while(1){

if(Serial.available() > 0){

input = Serial.read();

}

if(input == 1){

stateFlag = 1;

break;

}

}

}

}

}

//If input us between 5 and 45, adjust the delay time of the servo rotation

if(input >= 5 && input <= 45){

servoDelayTime = 60 - input;

}

// Repeats the previous lines from 165 to 15 degrees

for(int i=170;i>10;i--){

myServo.write(i);

delay(servoDelayTime);

distance = calculateDistance();

Serial.print(i); // Sends the current degree into the Serial Port

Serial.print(","); // Sends addition character right next to the previous value needed later in the Processing IDE for indexing

Serial.print(distance); // Sends the distance value into the Serial Port

Serial.print(","); // Sends addition character right next to the previous value needed later in the Processing IDE for indexing

Serial.print(LM35temp);

Serial.println();

if(Serial.available() > 0){

//Check if there are new incoming bytes; if there is a 0 is received, pause the program

input = Serial.read();

if(input == 0){

stateFlag = 0;

//keep checking if there are new incoming bytes; if there is a 1 is received, resume the program

while(1){

if(Serial.available() > 0){

input = Serial.read();

}

if(input == 1){

stateFlag = 1;

break;

}

}

}

}

}

}

//If the stateFlag is 0, pause the program

else if(stateFlag == 0){

//keep checking if there are new incoming bytes; if there is a 1 is received, resume the program

while(1){

if(Serial.available() > 0){

input = Serial.read();

}

if(input == 1){

break;

}

}

}

}

/\*

\* \*This function calculates the distance from an object in front of the sensor to the sensor itself.

\*/

float calculateDistance(){

float distance\_pre = distance;

read\_LM35\_Temp(); // calls to function to read from the temperature sensor.alculate the speed of sound from ambient temperature

SpeedOfSound = 331.5 + (0.6 \* LM35temp); // calculate the speed of sound from ambient temperature

time\_cm = 20000 / SpeedOfSound; // calculate the time it takes for sound wave to travel (round trip) per cm in μs

digitalWrite(trigPin, LOW); // clear the trig pin to low.

//delayMicroseconds(30);

delay(1);

digitalWrite(trigPin, HIGH); // trigger the sensor.

//delayMicroseconds(60);

delay(1);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound wave travel time in μs

distance= duration / time\_cm; // calculate the distance.

if(distance > 1 && distance < 200){

return distance;

}else{

return distance\_pre;

}

}

/\*

\* This function reads from the temperature sensor and calculate the ambient temperature.

\* Retrieved from lad 4 code.

\*/

void read\_LM35\_Temp(){

float temp\_pre = LM35temp;

LM35reading = analogRead(LM35Pin);

LM35temp = LM35reading /9.31;

//This functions as a filter; The LM35 temperature sensor sometimes reads a wrong value

//If this happens, this filter should filter out that value and use the previous value

//instead

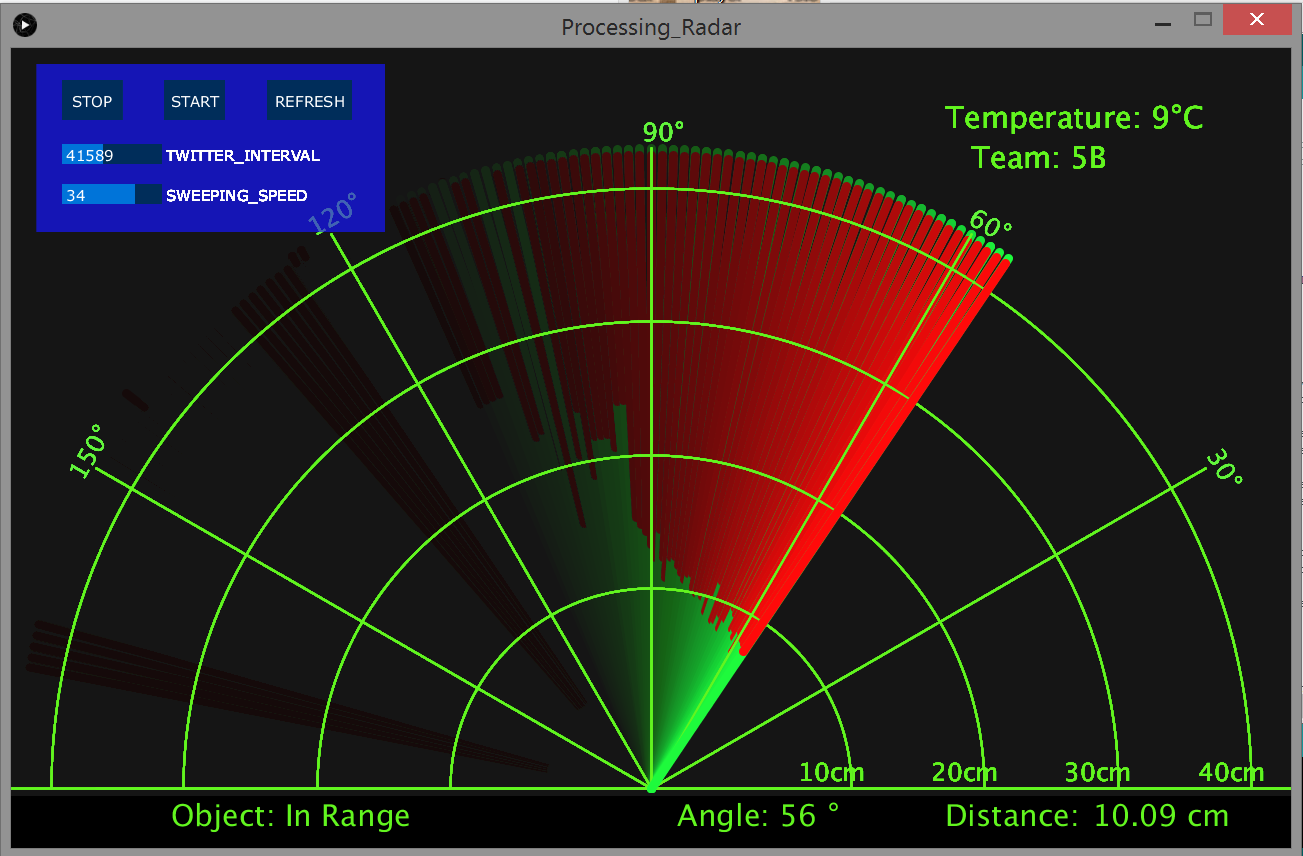
if(millis() > 500 && abs(temp\_pre - LM35temp) >= 0.5){

LM35temp = temp\_pre;

}

}

**Appendix III**

**(Snapshot)**

/\* Author: Yuxiang Huang

\* Date: 10, February

\* Purpose: Design a radar GUI for lab 5;

\* Get familiar with Processing

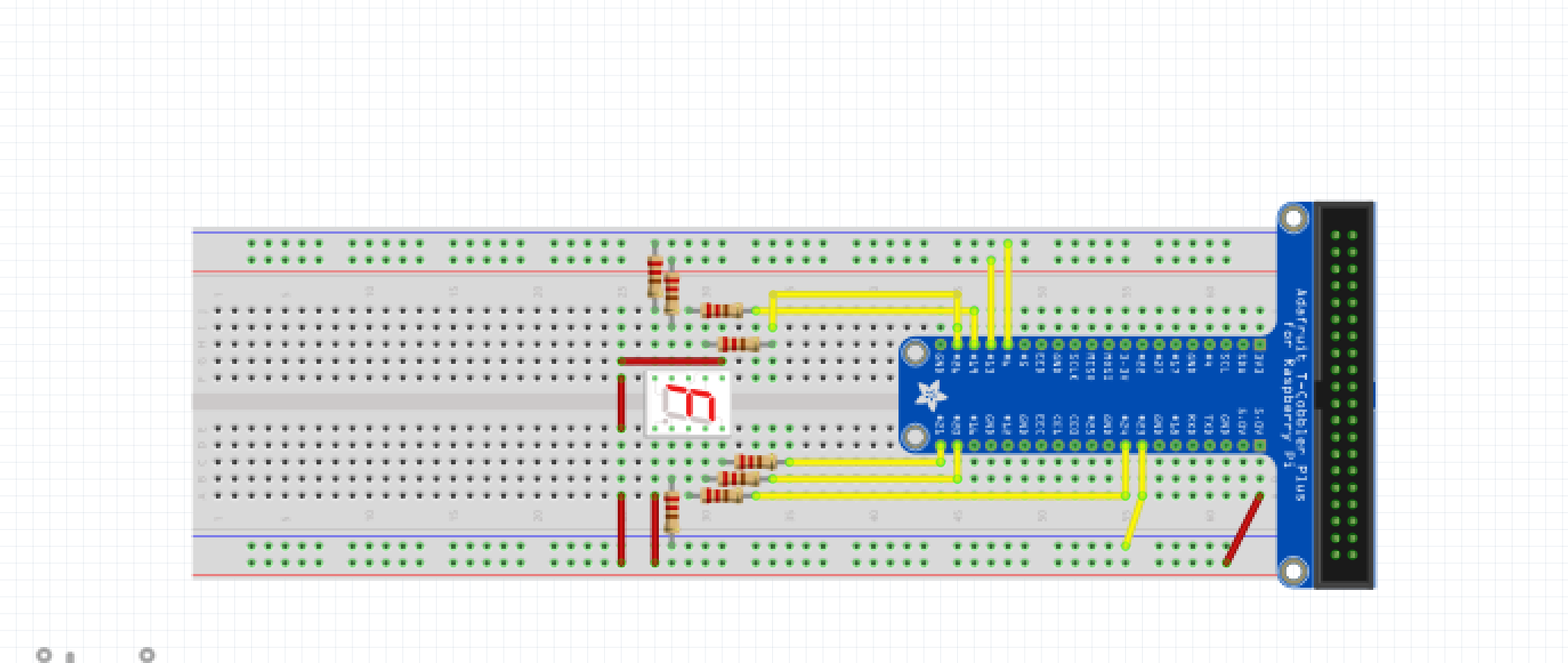
\*/

|  |  |
| --- | --- |
| **import g4p\_controls.\*;** |  |
|  |  |
|  | **import controlP5.\*;** |
|  |  |
|  | **import twitter4j.\*;** |
|  | **import twitter4j.api.\*;** |
|  | **import twitter4j.auth.\*;** |
|  | **import twitter4j.conf.\*;** |
|  | **import twitter4j.json.\*;** |
|  | **import twitter4j.management.\*;** |
|  | **import twitter4j.util.\*;** |
|  | **import twitter4j.util.function.\*;** |
|  |  |
|  | **import processing.serial.\*; // imports library for serial communication** |
|  | **import java.awt.event.KeyEvent; // imports library for reading the data from the serial port** |
|  | **import java.io.IOException;** |
|  | **Serial myPort; // defines Object Serial** |
|  | **Twitter twitter;** |
|  | **ControlP5 gui;** |
|  |  |
|  |  |
|  | **// defubes variables** |
|  | **String data="";** |
|  | **String noObject;** |
|  | **float pixsDistance;** |
|  | **int iAngle;** |
|  | **float iDistance;** |
|  | **int temperature;** |
|  | **int time = 0; //keep track of the time** |
|  | **int stop = 1;** |
|  | **int twitterInterval = 99999;** |
|  | **String[] values = new String[3];** |
|  | **PFont orcFont;** |
|  | **PImage img;** |
|  |  |
|  | **void setup() {** |
|  | **img = loadImage("Cosmos.jpg");** |
|  | **//tint(255, 10);** |
|  |  |
|  | **size (1280, 800); // \*\*\*CHANGE THIS TO YOUR SCREEN RESOLUTION\*\*\*** |
|  | **smooth();** |
|  | **myPort = new Serial(this,"COM7", 9600); // starts the serial communication** |
|  | **myPort.bufferUntil('\n'); // reads the data from the serial port up to the character '.'. So actually it reads this: angle,distance.** |
|  | **//orcFont = loadFont("OCRAExtended-30.vlw");** |
|  |  |
|  | **/\*\*\*The following is for Twitter\*\*\*/** |
|  | **ConfigurationBuilder cb = new ConfigurationBuilder();** |
|  | **cb.setOAuthConsumerKey("aW0c4ZRktv0DRjlPBJ1WCWenE");** |
|  | **cb.setOAuthConsumerSecret("LcxzcNqOeH1UYdWK3nmNMhoFEYnFN9ZLpnw1SqXT3yLuytbur8");** |
|  | **cb.setOAuthAccessToken("710166322907373569-bvkvV865MG02oqtzyLlZRNLqorZCav7");** |
|  | **cb.setOAuthAccessTokenSecret("d7UUybjwGJ9Xjg4QWJ9U3xHP2edmfOel48FXqyKZ3tAOh");** |
|  |  |
|  | **TwitterFactory tf = new TwitterFactory(cb.build());** |
|  |  |
|  | **twitter = tf.getInstance();** |
|  |  |
|  | **/\*\*\*The following is for Buttons\*\*\*/** |
|  | **gui = new ControlP5(this);** |
|  | **PFont p = createFont("Verdana",15);** |
|  |  |
|  | **gui.addButton("Stop")** |
|  | **.setPosition(width\*0.04, height\*0.04)** |
|  | **.setSize(60,40)** |
|  | **.setValue(0)** |
|  | **//add image to the button** |
|  | **.setImages(img)** |
|  | **.updateSize()** |
|  | **//set colors for the button** |
|  | **/\*.setColorForeground(0xffaa0000)** |
|  | **.setColorBackground(0xff660000)** |
|  | **.setColorValue(0xffff88ff)** |
|  | **.setColorLabel(0xffdddddd)** |
|  | **.setColorActive(0xffff0000)\*/** |
|  | **.activateBy(ControlP5.RELEASE);** |
|  |  |
|  | **gui.addButton("Start")** |
|  | **.setPosition(width\*0.12, height\*0.04)** |
|  | **.setSize(60,40)** |
|  | **.setValue(1)** |
|  | **.setImages(loadImage("Cosmos.jpg"))** |
|  | **.updateSize()** |
|  | **.activateBy(ControlP5.RELEASE);** |
|  |  |
|  | **gui.addButton("Refresh")** |
|  | **.setPosition(width\*0.20, height\*0.04)** |
|  | **.setSize(85,40)** |
|  | **.setValue(2)** |
|  | **.setImages(loadImage("Cosmos.jpg"))** |
|  | **.updateSize()** |
|  | **.activateBy(ControlP5.RELEASE);** |
|  |  |
|  | **gui.addSlider("Twitter\_Interval", 1000, 99999, 256, int(width\*0.04),int(height\*0.12),100,20)** |
|  | **.setValue(256);** |
|  |  |
|  |  |
|  | **gui.addSlider("Sweeping\_Speed", 5, 45, 128, int(width\*0.04),int(height\*0.17),100,20)** |
|  | **.setValue(128);** |
|  |  |
|  | **gui.setFont(p);** |
|  |  |
|  | **}** |
|  | **void draw() {** |
|  |  |
|  | **fill(30, 25, 255, 2);** |
|  | **rect(width\*0.02, height\*0.02, width\*0.272, height\*0.21);** |
|  |  |
|  | **if(stop == 0){** |
|  | **//image(img, 0, 0, width, height);** |
|  |  |
|  | **fill(98,245,31);** |
|  | **//textFont(orcFont);** |
|  | **// simulating motion blur and slow fade of the moving line** |
|  | **noStroke();** |
|  | **fill(0,6);** |
|  | **rect(0, 0, width, height-height\*0.065);** |
|  |  |
|  | **fill(98,245,31); // green color** |
|  | **// calls the functions for drawing the radar** |
|  | **drawRadar();** |
|  | **drawLine();** |
|  | **drawObject();** |
|  | **drawText();** |
|  | **}** |
|  | **}** |
|  |  |
|  | **void serialEvent (Serial myPort) {** |
|  | **try{** |
|  | **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*For radar\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/** |
|  |  |
|  | **// starts reading data from the Serial Port** |
|  | **// reads the data from the Serial Port up to the character '.' and puts it into the String variable "data".** |
|  | **data = myPort.readStringUntil('\n');** |
|  | **//data = data.substring(0,data.length()-1);** |
|  |  |
|  | **values = splitTokens(data, ",");** |
|  |  |
|  | **// converts the String variables into Integer** |
|  | **iAngle = int(values[0]);** |
|  | **iDistance = float(values[1]);** |
|  | **temperature = int(values[2]);** |
|  | **if(millis() - time > twitterInterval){** |
|  | **tweet();** |
|  | **time = millis();** |
|  | **}** |
|  | **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*For twitter \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/** |
|  | **}** |
|  | **catch(RuntimeException e){** |
|  | **e.printStackTrace();** |
|  | **}** |
|  | **}** |
|  |  |
|  | **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/** |
|  | **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*The following functions are for radar\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/** |
|  | **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/** |
|  |  |
|  | **void drawRadar() {** |
|  | **pushMatrix();** |
|  | **translate(width/2,height-height\*0.074); // moves the starting coordinats to new location** |
|  | **noFill();** |
|  | **strokeWeight(2);** |
|  | **stroke(98,245,31);** |
|  | **// draws the angle lines** |
|  | **line(-width/2,0,width/2,0);** |
|  | **line(0,0,(-width/2)\*cos(radians(30)),(-width/2)\*sin(radians(30)));** |
|  | **line(0,0,(-width/2)\*cos(radians(60)),(-width/2)\*sin(radians(60)));** |
|  | **line(0,0,(-width/2)\*cos(radians(90)),(-width/2)\*sin(radians(90)));** |
|  | **line(0,0,(-width/2)\*cos(radians(120)),(-width/2)\*sin(radians(120)));** |
|  | **line(0,0,(-width/2)\*cos(radians(150)),(-width/2)\*sin(radians(150)));** |
|  | **line((-width/2)\*cos(radians(30)),0,width/2,0);** |
|  | **// draws the arc lines** |
|  | **arc(0,0,(width-width\*0.0625),(width-width\*0.0625),PI,TWO\_PI);** |
|  | **arc(0,0,(width-width\*0.27),(width-width\*0.27),PI,TWO\_PI);** |
|  | **arc(0,0,(width-width\*0.479),(width-width\*0.479),PI,TWO\_PI);** |
|  | **arc(0,0,(width-width\*0.687),(width-width\*0.687),PI,TWO\_PI);** |
|  | **popMatrix();** |
|  | **}** |
|  |  |
|  | **void drawObject() {** |
|  | **pushMatrix();** |
|  | **translate(width/2,height-height\*0.074); // moves the starting coordinats to new location** |
|  | **strokeWeight(9);** |
|  | **stroke(255,10,10); // red color** |
|  | **pixsDistance = iDistance\*((height-height\*0.1777)\*0.025); // covers the distance from the sensor from cm to pixels** |
|  | **// limiting the range to 40 cms** |
|  | **if(iDistance<40){** |
|  | **// draws the object according to the angle and the distance** |
|  | **line(pixsDistance\*cos(radians(iAngle)),-pixsDistance\*sin(radians(iAngle)),(width-width\*0.505)\*cos(radians(iAngle)),-(width-width\*0.505)\*sin(radians(iAngle)));** |
|  | **}** |
|  | **popMatrix();** |
|  | **}** |
|  |  |
|  | **void drawLine() {** |
|  | **pushMatrix();** |
|  | **strokeWeight(9);** |
|  | **stroke(30,250,60);** |
|  | **translate(width/2,height-height\*0.074); // moves the starting coordinats to new location** |
|  | **line(0,0,(height-height\*0.2)\*cos(radians(iAngle)),-(height-height\*0.2)\*sin(radians(iAngle))); // draws the line according to the angle** |
|  | **popMatrix();** |
|  | **}** |
|  | **void drawText() { // draws the texts on the screen** |
|  |  |
|  | **pushMatrix();** |
|  | **if(iDistance>40) {** |
|  | **noObject = "Out of Range";** |
|  | **}** |
|  | **else {** |
|  | **noObject = "In Range";** |
|  | **}** |
|  | **fill(0,0,0);** |
|  | **noStroke();** |
|  | **rect(0, height-height\*0.0648, width, height);** |
|  | **fill(98,245,31);** |
|  | **textSize(25);** |
|  |  |
|  | **text("10cm",width-width\*0.3854,height-height\*0.0833);** |
|  | **text("20cm",width-width\*0.281,height-height\*0.0833);** |
|  | **text("30cm",width-width\*0.177,height-height\*0.0833);** |
|  | **text("40cm",width-width\*0.0729,height-height\*0.0833);** |
|  | **textSize(30);** |
|  | **text("Temperature: " + "9" + "°C", width\*0.73, height\*0.10);** |
|  | **text("Team: 5B ", width\*0.75, height\*0.15);** |
|  | **text("Object: " + noObject, width-width\*0.875, height-height\*0.0277);** |
|  | **text("Angle: " + iAngle +" °", width-width\*0.48, height-height\*0.0277);** |
|  | **text("Distance: ", width-width\*0.27, height-height\*0.0277);** |
|  |  |
|  | **if(iDistance<40) {** |
|  | **text(" " + iDistance +" cm", width-width\*0.215, height-height\*0.0277);** |
|  | **}** |
|  |  |
|  | **textSize(25);** |
|  | **fill(98,245,60);** |
|  | **translate((width-width\*0.4994)+width/2\*cos(radians(30)),(height-height\*0.0907)-width/2\*sin(radians(30)));** |
|  | **rotate(-radians(-60));** |
|  | **text("30°",0,0);** |
|  | **resetMatrix();** |
|  | **translate((width-width\*0.503)+width/2\*cos(radians(60)),(height-height\*0.0888)-width/2\*sin(radians(60)));** |
|  | **rotate(-radians(-30));** |
|  | **text("60°",0,0);** |
|  | **resetMatrix();** |
|  | **translate((width-width\*0.507)+width/2\*cos(radians(90)),(height-height\*0.0833)-width/2\*sin(radians(90)));** |
|  | **rotate(radians(0));** |
|  | **text("90°",0,0);** |
|  | **resetMatrix();** |
|  | **translate(width-width\*0.513+width/2\*cos(radians(120)),(height-height\*0.07129)-width/2\*sin(radians(120)));** |
|  | **rotate(radians(-30));** |
|  | **text("120°",0,0);** |
|  | **resetMatrix();** |
|  | **translate((width-width\*0.5104)+width/2\*cos(radians(150)),(height-height\*0.0574)-width/2\*sin(radians(150)));** |
|  | **rotate(radians(-60));** |
|  | **text("150°",0,0);** |
|  | **popMatrix();** |
|  | **}** |
|  |  |
|  | **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/** |
|  | **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*The following functions are for Twitter\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/** |
|  | **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/** |
|  |  |
|  | **void tweet()** |
|  | **{** |
|  | **try** |
|  | **{** |
|  | **if(iDistance < 30){** |
|  |  |
|  | **Status status = twitter.updateStatus("An object is approaching! Distance: " + iDistance + "cm");** |
|  | **System.out.println("Status updated to [" + status.getText() + "].");** |
|  | **}** |
|  | **}** |
|  | **catch (TwitterException te)** |
|  | **{** |
|  | **System.out.println("Error: "+ te.getMessage());** |
|  | **}** |
|  | **}** |
|  |  |
|  | **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/** |
|  | **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*The following functions are for buttons\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/** |
|  | **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/** |
|  |  |
|  | **public void controlEvent(ControlEvent theEvent) {** |
|  | **println(theEvent.getController().getValue());** |
|  | **}** |
|  |  |
|  | **// function Start** |
|  | **public void Stop(int theValue) {** |
|  | **println("a button event from Stop: "+theValue);** |
|  | **stop = 1;** |
|  | **myPort.write(theValue);** |
|  | **}** |
|  |  |
|  | **// function colorB will receive changes from** |
|  | **// controller with name colorB** |
|  | **public void Start(int theValue) {** |
|  | **println("a button event from Start: "+theValue);** |
|  | **stop = 0;** |
|  | **myPort.write(theValue);** |
|  | **}** |
|  |  |
|  | **public void Refresh(int theValue) {** |
|  | **println("a button event from Refresh: "+theValue);** |
|  | **background(25, 4);** |
|  | **}** |
|  |  |
|  | **public void Sweeping\_Speed(int theValue) {** |
|  | **println("a silder event from SweepingSpeed: "+theValue);** |
|  | **myPort.write(theValue);** |
|  | **}** |
|  |  |
|  | **public void Twitter\_Interval(int theValue) {** |
|  | **println("a silder event from Twitter\_Interval: "+theValue);** |
|  | **twitterInterval = theValue;** |
|  | **}** |

**Appendix IV**

**Reflections on the challenges and differences**

Since it was our first time to touch with Python and Raspberry Pi, we met several challenges feel differences between raspberry pi and Audrino. Firstly, on the hardware level, the GPIO pin and the pin in Audrino are different. Even though we were using breadboard to connect T-cobbler, it was still hard to organize wire and make the breadboard compact. Secondly, we found it challenging to code in Python. For example, Python does not have switch, and we tried to make the led selection clean and neat. We spent a lot of time on that but had to use if statement for 8 leds selection. Thirdly, when we did the experiment, two resistors touched each other, and this made the circuit short and the Raspberry Pi suddenly turned off when we connect T-cobbler to Raspberry Pi. Finally, the Decimal Point LED seemed to have some problem inside. After successfully run the program for one time, it always turned on and also made other leds wrong. Therefore, we had to change to another pin for Decimal Point Led.



**import** time *#import time library*

**import** RPi.GPIO **as** GPIO *#import GPIO library*

GPIO.setmode(GPIO.BCM)

GPIO.setwarning(**False**)

*#initialize for ledA - ledF*

LedA = 21

LedB = 20

LedC = 24

LedDP = 23

LedD = 6

LedE = 13

LedG = 19

LedF = 26

numbers = [0xFC, 0x60, 0xDa,0xF2,0x66,0xB6,0xBE,0xE0,0xFE,0xE6]

*#number array contains 10 hexadecimal numbers which indicates certain leds turn on*

*#and certain led turn off*

*#setup for LedA - LedDP*

GPIO.setup(LedA,GPIO.OUT)

GPIO.setup(LedB,GPIO.OUT)

GPIO.setup(LedC,GPIO.OUT)

GPIO.setup(LedD,GPIO.OUT)

GPIO.setup(LedDP,GPIO.OUT)

GPIO.setup(LedE,GPIO.OUT)

GPIO.setup(LedF,GPIO.OUT)

GPIO.setup(LedG,GPIO.OUT)

print(**"initial setup"**)

*#turn all leds off at the beginning*

GPIO.output(LedA,GPIO.HIGH)

GPIO.output(LedB,GPIO.HIGH)

GPIO.output(LedC,GPIO.HIGH)

GPIO.output(LedD,GPIO.HIGH)

GPIO.output(LedDP,GPIO.HIGH)

GPIO.output(LedE,GPIO.HIGH)

GPIO.output(LedF,GPIO.HIGH)

GPIO.output(LedG,GPIO.HIGH)

*# function counter accept a hexadecimal number pin, go through every*

*#if statement to determine certain led will turn on or off*

**def** counter(pin):

**if**(pin&0x80 == 0x80): *#determine whether 1st bit is 1 or not*

GPIO.output(LedA,GPIO.LOW) *# if 1st bit is 1, turn ledA on*

**else**:

GPIO.output(LedA,GPIO.HIGH) *# if not, turn ledA off*

**if**(pin&0x40 == 0x40): *#determine whether 2nd bit is 1 or not*

GPIO.output(LedB, GPIO.LOW) *# if 1st bit is 1, turn ledB on*

**else**:

GPIO.output(LedB, GPIO.HIGH) *# if not, turn ledB off*

**if**(pin&0x20 == 0x20): *#determine whether 3rd bit is 1 or not*

GPIO.output(LedC, GPIO.LOW) *# if 1st bit is 1, turn ledC on*

**else**:

GPIO.output(LedC, GPIO.HIGH) *# if not, turn ledC off*

**if**(pin&0x10 == 0x10): *#determine whether 4th bit is 1 or not*

GPIO.output(LedD, GPIO.LOW) *# if 1st bit is 1, turn ledD on*

**else**:

GPIO.output(LedD, GPIO.HIGH) *# if not, turn ledD off*

**if**(pin&0x08 == 0x08): *#determine whether 5th bit is 1 or not*

GPIO.output(LedE, GPIO.LOW) *# if 1st bit is 1, turn ledE on*

**else**:

GPIO.output(LedE, GPIO.HIGH) *# if not, turn ledE off*

**if**(pin&0x04 == 0x04): *#determine whether 6th bit is 1 or not*

GPIO.output(LedF, GPIO.LOW) *# if 1st bit is 1, turn ledF on*

**else**:

GPIO.output(LedF, GPIO.HIGH) *# if not, turn ledF off*

**if**(pin&0x02 == 0x02): *#determine whether 7th bit is 1 or not*

GPIO.output(LedG, GPIO.LOW) *# if 1st bit is 1, turn ledG on*

**else**:

GPIO.output(LedG, GPIO.HIGH) *# if not, turn ledG off*

**if**(pin&0x01 == 0x01): *#determine whether 8th bit is 1 or not*

GPIO.output(LedDP, GPIO.LOW) *# if 1st bit is 1, turn ledDP on*

**else**:

GPIO.output(LedDP, GPIO.HIGH) *# if not, turn ledDP off*

**while** 1 : *# infinite while loop*

**for** x **in** range(10): *# for loop make x in range of 10. from 0 - 9*

pin = numbers[x]

print(x)

counter(pin) *# call the counter function*

time.sleep(1) *# delay 1 second*