CPEN 291 2016W2
Lab 4

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

Student name	*Student number*	*Contribution percentage*
Ziqiao Lin	10668168	33.3%
Yuhao Huang	55562152	33.3%
Yuxiang Huang	14605159	33.3%

Contribution summary:

Ziqiao Lin: Part of circuit layout; part of the lab report; code for Arduino.
Yuhao Huang: Most of circuit layout; Wiring; schematic in Fritzing; part of the lab report;
Yuxiang Huang: Code for Processing and MATLAB; part of lab report;

A. Introduction and motivations

This report discusses a project to implement a simple weather monitoring system, which detects ambient temperature, humidity, and light level, and plots the temperature, humidity, and light level over time in MATLAB and Processing. During this lab, we learned about how to use the digital humidity sensor (DHT11), the analog temperature sensor (LM35), the photocell, and the on-off switch; we also learned to interface Arduino with MATLAB and Processing. This report presents the procedures for the project, and a conclusion and analysis of reflections.

B. Lab Description

1. **Circuit Layout:** We connected the components to the Arduino board according to the datasheets and instructions in the experiment info, and organized the layout so that they can fit onto the breadboard compactly, which was proved to be a challenging task. The DHT11 and the switch were connected to digital pins, with the DHT11 protected by a pull-up resistor, while the LM35 and the photocell were connected to analog pins. All components were connected to 5V output and ground, with the switch and the photocell each protected by a 10k ohm resistor. We managed to place each component in the positions that were closest to their respective I/O pins, making the placement of their respective wires compact and organized.

2. **The Weather Monitoring System:**

   a) **DHT11 Temperature Sensor:** We implemented two functions, "calling" and "read_data", to read from the DHT11 component. The function "calling" first sends signal to DHT to prepare for receiving input data from it by setting the pin to low for 40 ms, and then high for another 40 ms. After doing so, it sets the mode of the pin to input, and calls the function "read_data" in a while loop to read 40 bits from the component. The function "read_data" reads 8 bits, or one byte, from DHT11 each time in a for loop.

   b) **LM35 Analog Temperature Sensor:** We implemented a function "read_LM35_Temp" to read from this component. This function simply reads from the analog pin connected to LM35 and convert the data to degree in Celsius by multiplying 100 and divide by 255, which is the scale of the LM35 component.

   c) **Photocell:** The function "read_photocell" is used to receive data from the photocell; it reads the input data from the analog pin connected to the component.

   d) **On-off Switch:** The on/off state of the switch is read by a "digitalRead" function in the main program .

   The four components are implemented separately, and finally controlled simultaneously by the main program. In the main program, we read first from the switch, and then from the other 3 components every 3 seconds. For different status of the switch, the data is displayed in the serial monitor in 2 different patterns.

3. **Processing:** Processing is used to create a user interface, where users can view the graphs plotted according to the data received from the serial port, as well as control the plotting process. In this lab, we created 4 real time plots (temperature received from 2 sensors, humidity, and ambient light level) on the user interface, and user can choose to start or stop the plotting process simply by clicking 2 buttons on the right side of the interface. we used an external library "ControlP5" to implement buttons. Also, we consult online resources and tutorials in order to have a better understanding

of Processing, and how it interacts with Arduino.

4. **MATLAB:** Matlab is used to create real time plots according to the data received from the Serial port. Similar to what we do in Processing, in MATLAB we plot 4 graphs simultaneously in one interface. In order to make the plots more professional, we added titles, units and legends on our graphs. In order to simultaneously plot multiple graphs, we used arrays to store the data received from the serial port, and graphic objects and plot functions are used for plotting the data from the array. We consulted online resources and tutorials in order to use Matlab properly.

C. Conclusions and Reflections

Overall the program works well and our circuit looks compact and organized. Having the experience from the previous lab, we designed the circuit layout on Fritzing and before wiring the components onto the breadboard; by doing so we saved lots of time and materials. However, we still encountered some problems in this experiment. Firstly, we spent quite a lot of time on finding why the DHT library doesn't work in our laptop. Next, we tried very hard on DHT datasheet but the data we received are quite different. Finally, we also did not figure out why processing and matlab sometimes do not work but sometimes work fine. An aspect we could improve on for next time would be to saving one copy of code before we add more things. This would improve our progress of coding and also save much time after.

D. References

Arduino language reference: http://arduino.cc/en/Reference/HomePage

LM35 temperature sensor component datasheet: LM35ComponentDatasheet.pdf

LM35:https://arduino-info.wikispaces.com/LM35

DHT11 datasheet: DHT11_datasheet.pdf

DHT11 datasheet another version: https://cdn-shop.adafruit.com/datasheets/DHT11-chinese.pdf

DHT sensor library: https://github.com/adafruit/DHT-sensor-library

Graphing in Processing: https://www.arduino.cc/en/Tutorial/Graph

Push Botton: https://www.arduino.cc/en/Tutorial/Button

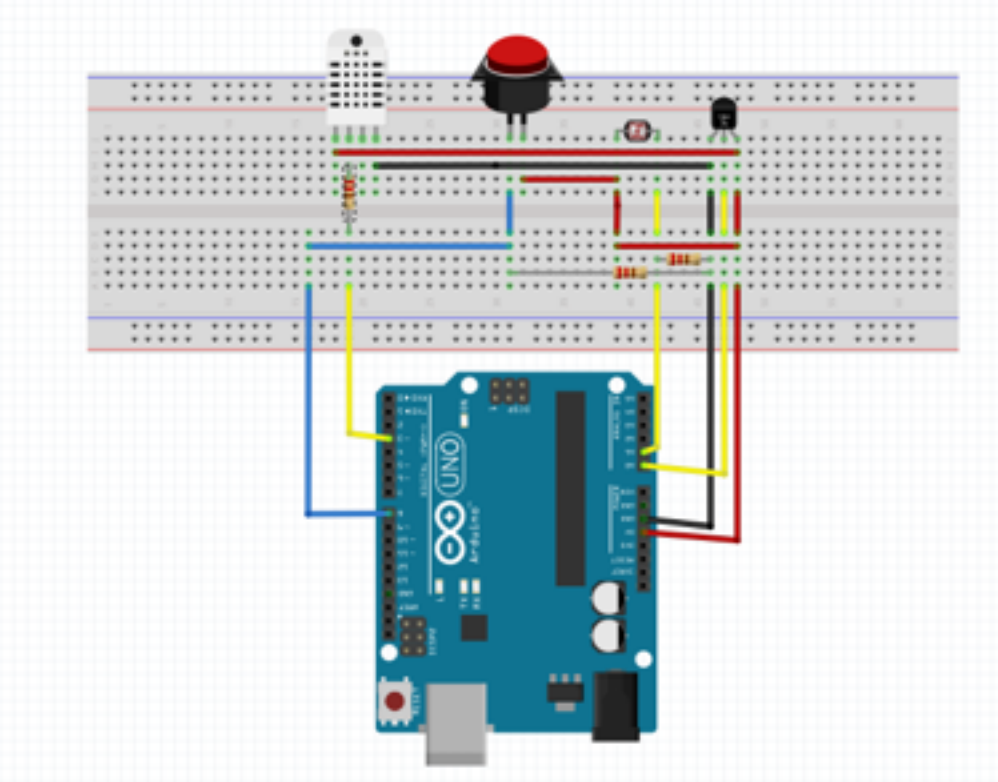
DHT library implementation: https://github.com/brucetsao/techbang/blob/master/201510/DHT_OK_For_86duino_Product/DHT_OK_For_86duino_Product.ino

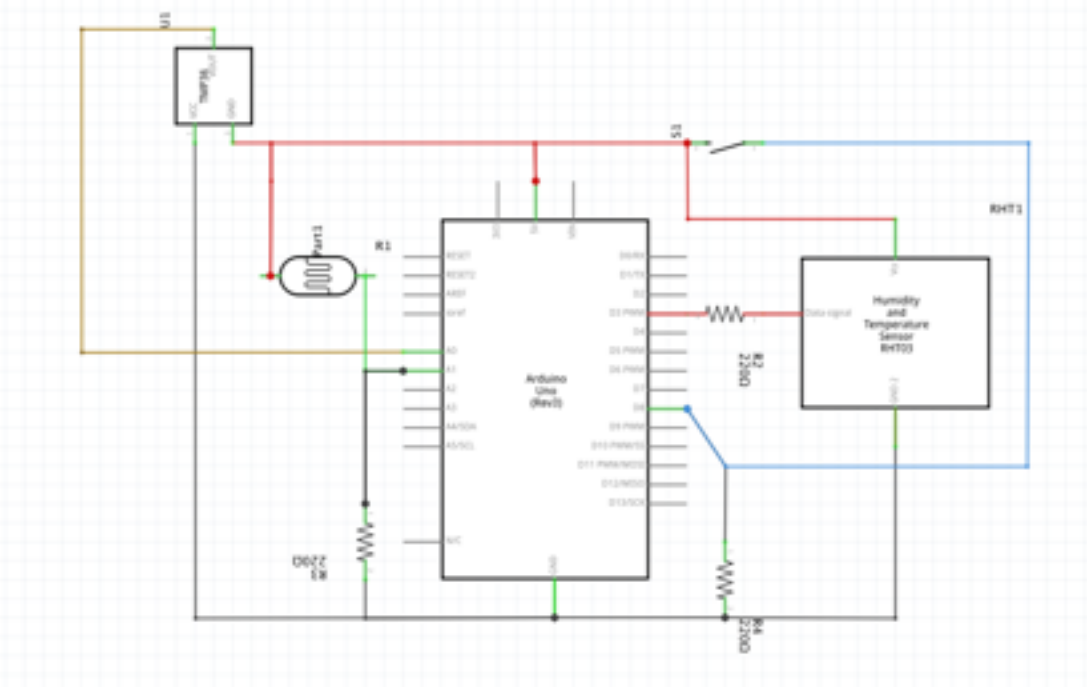
http://www.techbang.com/posts/39932-how-to-be-a-professional-maker-rewrite-the-program-to-use-the-library-syntax

https://www.noao.edu/education/QLTkit/ACTIVITY_Documents/Safety/LightLevels_outdoor+indoor.pdf

http://www.hobbyist.co.nz/?q=documentations/wiring-up-dht11-temp-humidity-sensor-to-your-arduino

Appendix I




U1
S1
RHT1
R1
Humidity and Temperature Sensor RHT03
Arduino Uno (Rev3)
R2 220Ω
R3 220Ω
R4 220Ω

Appendix II (Arduino code)

```
/             /
************
*********
```

```
int DHTPin = 8;             // DHT Pin number
byte bits[5];               // 40 bits read from DHT
//**********************
/*
* setup for LM35
*/
float LM35temp;
float LM35reading;
int LM35Pin= A1;            // analog Pin for LM35
//************************
/*
* setup for phtocell
*/


int photocellPin = A0;      // analog Pin for photocell
int photocellReading;       // photocell reading


//************************
/*
* setup for button
*/
const int buttonPin = 11;     // button Pin number
```

```
int buttonState = 0;          // button State



//**************************
void setup() {

 Serial.begin(9600);
 analogReference(INTERNAL);
 pinMode(buttonPin,INPUT);
 pinMode(DHTPin,OUTPUT);
}




void loop() {

 buttonState = digitalRead(buttonPin);   // read button
 read_LM35_Temp();                    // calling the method to read data from LM35
 read_photocell();                   // calling the method to read data from photocell
 calling();                         // calling the method to read data from DHT
 if(buttonState == HIGH){                  // if the signal is high for button, we just
print raw data
          Serial.print(bits[2],DEC);    // print digital position of temperature
          Serial.print(".");           // print decimal
          Serial.print(bits[3],DEC);  // print number after decimal
          Serial.print(" ");
```

```
        Serial.print(bits[0],DEC);    // print digital position of humandity
        Serial.print(".");            // print decimal
        Serial.print(bits[1],DEC);  // print number after decimal
        Serial.print(" ");
        Serial.print(LM35temp);     // print LM35 data
        Serial.print(" ");
        Serial.println(photocellReading);  // print photocell data
    }else{
        Serial.print("Temperature: ");    // print temperature with text
        Serial.print(bits[2],DEC);
        Serial.print(".");
        Serial.print(bits[3],DEC);
        Serial.println(" *C");           // print unit for temperature
        Serial.print("Humidity: ");       // print humidity
        Serial.print(bits[0],DEC);
        Serial.print(".");
        Serial.print(bits[1],DEC);
        Serial.println(" %");            // print unit for humidity
        Serial.print("Temperature from LM35:");  // print temperature from LM35
        Serial.print(LM35temp);
        Serial.println(" *C");                  // print unit of temperature
        Serial.print("ambient light level: ");   // print ambient light level
        Serial.print(photocellReading);          // print data
        Serial.println(" foot candle");         // print unit of ambient light level
    }
  // Delay between measurements.


  delay(3000);
}
```

```
//*****************************************

/*
* this method is to get reading from LM35 and then convert to degree Celsius
*/
void  read_LM35_Temp(){
 LM35reading = analogRead(LM35Pin);
 LM35temp = LM35reading *100 / 255;
 }


//*****************************************
/*
* this method is to read data from photocell
*/
void read_photocell(){
 photocellReading = analogRead(photocellPin);
 }


//*****************************************
/*
* this is the main method to start to call DHT, give signal as format for reading data,
* first we will send out 40 ms low signal to ensure DHT11 can receive the signal.Then we
* have to set high for another 40 ms. After this, change the pinMode to input and start to
```

```
* receive data. DHT will give 80 ms low signal as react, and then pull up to high for other
* 80 ms. Next, we are good to start to receive data
*/
void calling(){

 digitalWrite(DHTPin,LOW);          // set as low signal
 delay(40);                         // keep this signal for 40 ms
 digitalWrite(DHTPin,HIGH);         // then set the signal as high
 delayMicroseconds(40);             // keep another 40 ms
 pinMode(DHTPin,INPUT);             // change mode to be ready to receive signal from DHT
 while(digitalRead(DHTPin) == HIGH);   // Keep the signal as high until signal changes to low for acting
 delayMicroseconds(80);             // the DHT will give low signal for 80ms as reaction
 if(digitalRead(DHTPin) == LOW );     // wait until DHT pull up the signal
 delayMicroseconds(80);             // wait another 80 ms and then we are good to receive data


 for( int i = 0; i < 5; i++){       // receive 40 bits data together
   bits[i] = read_data();           // calling read_data method to read data
   }

 pinMode(DHTPin,OUTPUT);            // after receiving data, set the pinmode back to output
 digitalWrite(DHTPin,HIGH);         // pull up the DHT
 }


//**********************************
/*
```

```
* this method is mainly to just read data from DHT11. In the previous method, we already send the signal
* to DHT and get reply from DHT11, and after getting signal from DHT11, we are ready to receive 40 bits
* data from DHT11. Since we determine signal "0" or "1" depends on the duration of singal "1" after 50ms
* signal "0". And each digit we receive should follow the sequence from highest priority to lowest priority.
* THerefore, we let data each time or with new input and left shift it for 7-i bits.
*/
byte read_data()
{
  byte data;                    // define a byte type
  for(int i=0; i<8;i++)         // loop for receiving signal for each bit in one byte
  {
    if(digitalRead(DHTPin)==LOW)    // before start the signal, DHT will pull down for 50 ms,
                                    // this is to check whether is ready to receive signal
      {
        while(digitalRead(DHTPin)==LOW);    // wait until the DHT turn to high for 50 ms
          delayMicroseconds(30);            // delay for 30 ms to determine the signal whether is high or low
          if(digitalRead(DHTPin)==HIGH)     // after 30ms, if the signal is still high, which means this bit is high
            data |=(1<<(7-i));              //left shift 7-i bit, since we receive priority position first
          while(digitalRead(DHTPin) == HIGH);  //after determine signal as high, just wait to finish
      }
  }
  return data;                      // wait until the data return
}
```

Appendix III (MATLAB code)

```
% Date: Febrary 3rd, 2017
% Purpose: Get familiar with Matlab and finish CPEN 291 lab4


s1 = serial('COM7', 'BaudRate', 9600, 'Parity', 'none', 'DataBits', 8, 'StopBits', 1, 'FlowControl', 'none');
fopen(s1);
x = 1:10;
val=fscanf(s1);
result = 0*x;


%Create a graphic object
H = gobjects(1, 4);


%Create 4 subplots to plot 4 types of data received
subplot(2,2,1);
%Initialize one element in the H graphic object
H(1) = plot(x, result, 'r');
ylim([0 50]);
%Turn on the grid
grid on
%Add the title, and x, y label
title('Temperature');
xlabel('time (s)'); % x-axis label
ylabel('Degree(*C)'); % y-axis label


subplot(2,2,2);
H(2) = plot(x, result, 'm');
ylim([0 50]);
```

```
grid on
title('Humidity');
xlabel('time (s)'); % x-axis label
ylabel('Percent(%)'); % y-axis label

subplot(2,2,3);
grid on
H(3) = plot(x, result, 'Color',[0.2,0.1,0.9]);
ylim([0 100]);
title('Temperature from LM35');
xlabel('time (s)'); % x-axis label
ylabel('Degree(*C)'); % y-axis label
grid on;

subplot(2,2,4);
grid on;
H(4) = plot(x, result, 'Color',[0.2,0.5,0.4]);
ylim([0 1050]);
title('Ambient Light Level');
xlabel('time (s)'); % x-axis label
ylabel('LightLevel(foot candle)'); % y-axis label

StripChart('Initialize',gca);

%Use a nested loop to plot 4 liines of data simoultaneously
   for i = 1:100
     %loop through four elements in the reslut array, plot each of them
     %one by one
     for x = 1:4
       %store the received the data in the val string
```

```
        val=fscanf(s1);
        %use an array to store the float data in the array
        result = sscanf(val, '%f');
        StripChart('Update', H(x), result(x));
        %Display the result on the comment window
        display(result);
    end
    end

%close the serial port in the end
fclose(s1);
```

Appendix IV (Processing code)

```
/* Author: Yuxiang Huang
 * Date: February 3th, 2017
 * Purpose: To get a better understanding of designing
 * a user interface and Processing; finsih CPEN 291
 *Lab 4
*/

import controlP5.*;
import processing.serial.*;
Serial myPort;
//This is an external libray for button controlling
ControlP5 gui;

int start;

/*********************************************************************/
int numValues = 4; // number of input values or sensors
// * change this to match how many values your Arduino is sending *

//Create arrays to store data received from the serial port,
//As well well as some other parameters such as colors
float[] values = new float[numValues];
int[] min = new int[numValues];
int[] max = new int[numValues];
color[] valColor = new color[numValues];

//Define the height of the title
int titleH = 100; //title height
float partH; // partial screen height
```

```
float xPos = 0; // horizontal position of the graph
boolean clearScreen = true; // flagged when graph has filled screen

void setup() {
  //Define the size of the window
  size(1000, 700);
  gui = new ControlP5(this);
  gui.addButton("Start")
     .setPosition(870, 240)
     .setSize(80,80)
     .setValue(0)
     .activateBy(ControlP5.RELEASE);

  gui.addButton("Stop")
     .setPosition(870, 400)
     .setSize(80,80)
     .setValue(0)
     .activateBy(ControlP5.RELEASE);

  partH = (float)(height - titleH) / numValues;

  // List all the available serial ports:
  printArray(Serial.list());
  // First port [0] in serial list is usually Arduino, but *check every time*:
  String portName = Serial.list()[0];
  myPort = new Serial(this, portName, 9600);
  // don't generate a serialEvent() until getting a newline character:
  myPort.bufferUntil('\n');
```

```
textSize(12);

background(0);
noStroke();

// initialize:
values[0] = 0;
min[0] = 0;
max[0] = 50; // full range example, e.g. any analogRead
valColor[0] = color(255, 0, 0); // red

values[1] = 0;
min[1] = 0;
max[1] = 30;  // partial range example, e.g. IR distance sensor
valColor[1] = color(0, 255, 0); // green

values[2] = 0;
min[2] = 0;
max[2] = 300;    // digital input example, e.g. a button
valColor[2] = color(0, 0, 255); // blue

// example for adding a 4th value:
values[3] = 0;
min[3] = 0;
max[3] = 1040; // custom range example
valColor[3] = color(255, 0, 255); // purple

 /*values[4] = 0;
 min[4] = 0;
 max[4] = 1023; // custom range example
```

```
  valColor[4] = color(255, 0, 255); // purple*/
}


void draw() {
  //Set the backgound color (black) when the draw function is first executed
     if (clearScreen) {
       // erase screen with black:
       background(0);

       // or, erase screen with translucent black:
       //fill(0,200);
       //noStroke();
       //rect(0,0,width - 200,height);
       clearScreen = false; // reset flag
     }

     //Set up the title
     textSize(50);
     fill(102,204,255);
     text("CPEN 291 Lab4 User Interface", 140, 60);
     textSize(12);

    //If the start button is pushed, start to plot the graph
    if(start == 1){
      for (int i=0; i<numValues; i++) {

        // map to the range of partial screen height:
        float mappedVal = map(values[i], min[i], max[i], 0, partH);
```

```
// draw plot lines:
stroke(valColor[i]);
line(xPos, partH*(i+1) - mappedVal + titleH, xPos, partH*(i+1) - mappedVal + 1 + titleH);

// draw dividing line:
stroke(255);
line(0, height - partH*(i+1), width - 200, height - partH*(i+1));
line(0, titleH, width, titleH);
stroke(255);
line(800, titleH, 800, height);

// display values on screen:
fill(50);
noStroke();
rect(0, partH*i+1+titleH, 230, 15);
fill(255);

//Draw additional dividing lines according to the value of i
//(which indicates which element it is plotting
switch(i){
  case (0): text("Temperature ( *C ):", 2, partH*0+12+titleH);
  case (1): text("Humidity ( % ):", 2, partH*1+12+titleH);
  case (2): text("Temperature (LM35)(*C):", 2, partH*2+12+titleH);
  case (3): text("Relative Light Level: ", 2, partH*3+12+titleH);
}

text(round(values[i]), 160, partH*i+12+titleH);
fill(125);
text(max[i], 200, partH*i+12+titleH);
```

```
      //print(i + ": " + values[i] + "\t"); // Used for debugging
      //println("\t"+mappedVal); //Used for debugging
    }

    // increment the graph's horizontal position:
    xPos = xPos + 0.5;
    // if at the edge of the screen, go back to the beginning:
    if (xPos > width - 200) {
      xPos = 0;
      clearScreen = true;
    }
  }
}



void serialEvent(Serial myPort) {
  try {
    // get the ASCII string:
    String inString = myPort.readStringUntil('\n');
    //println("raw: \t" + inString); // Used for debugging

    if (inString != null) {
      // trim off any whitespace:
      inString = trim(inString);

        // split the string on the delimiters and convert the resulting substrings into an float array:
      values = float(splitTokens(inString, " ")); // delimiter can be comma space or tab
      for(int i = 0; i < numValues; i++){
        println(values[i]);
```

```
      }
     }
    }
    catch(RuntimeException e) {
      // only if there is an error:
      e.printStackTrace();
    }
}

//This function is used to read the button state; it calls the Start and Stop functions whenever
//The button state changes
public void controlEvent(ControlEvent theEvent) {
  println(theEvent.getController().getValue());
}

// function Start
public void Start(int theValue) {
  println("a button event from Start: "+theValue);
  start = 1;
}

// function colorB will receive changes from
// controller with name colorB
public void Stop(int theValue) {
  println("a button event from Stop: "+theValue);
  start = 0;
}
```