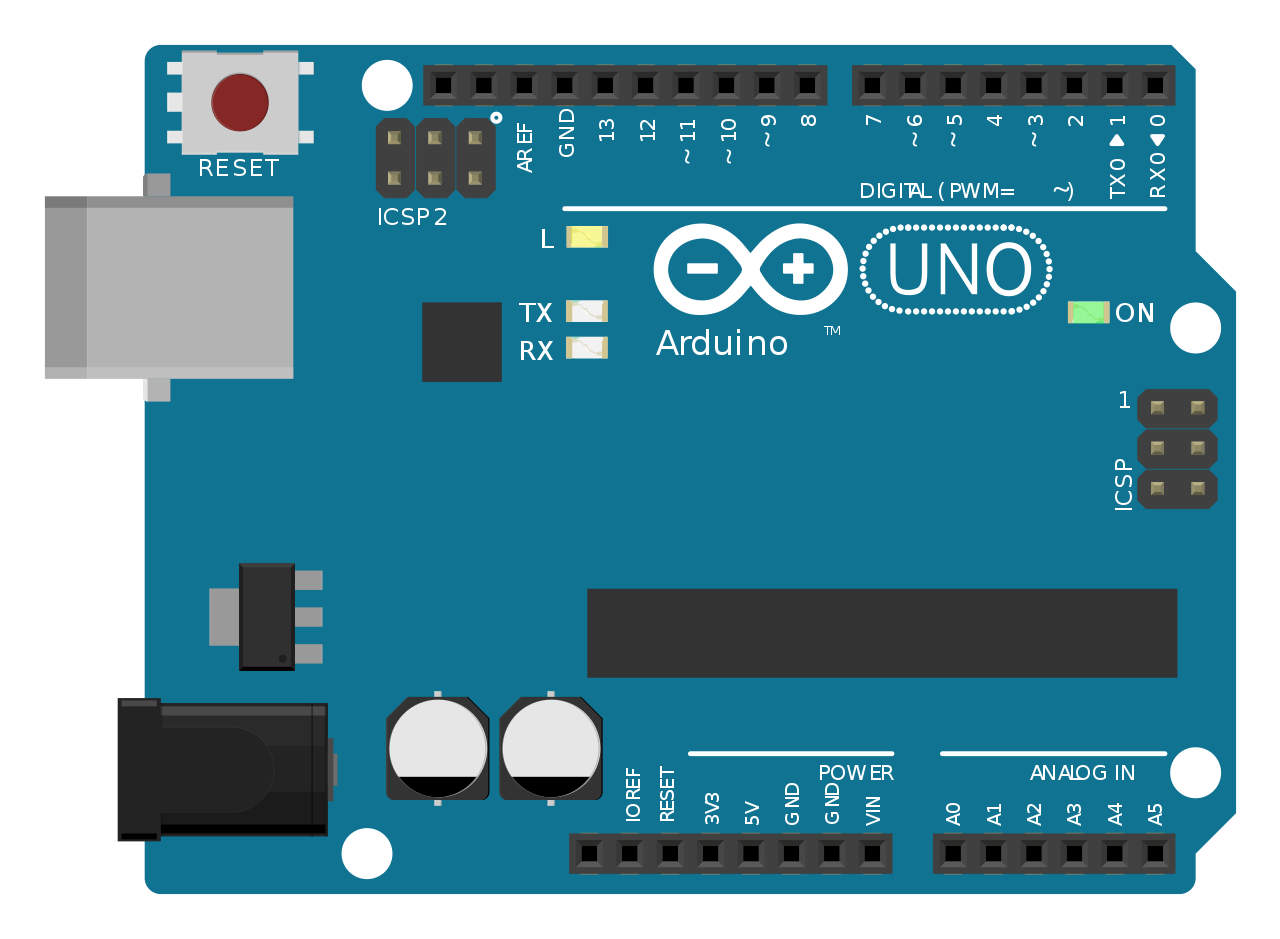
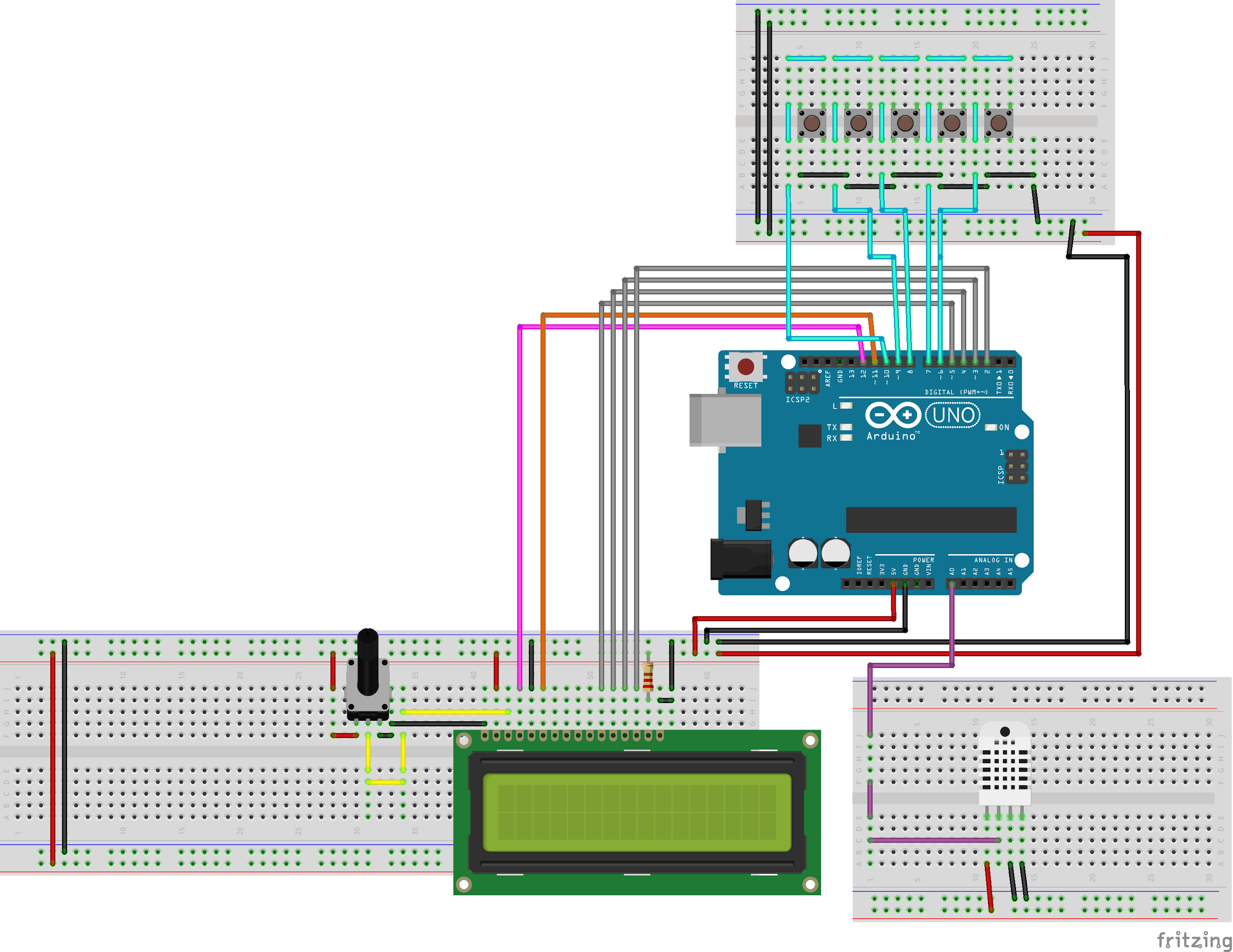
****

**Final Report**

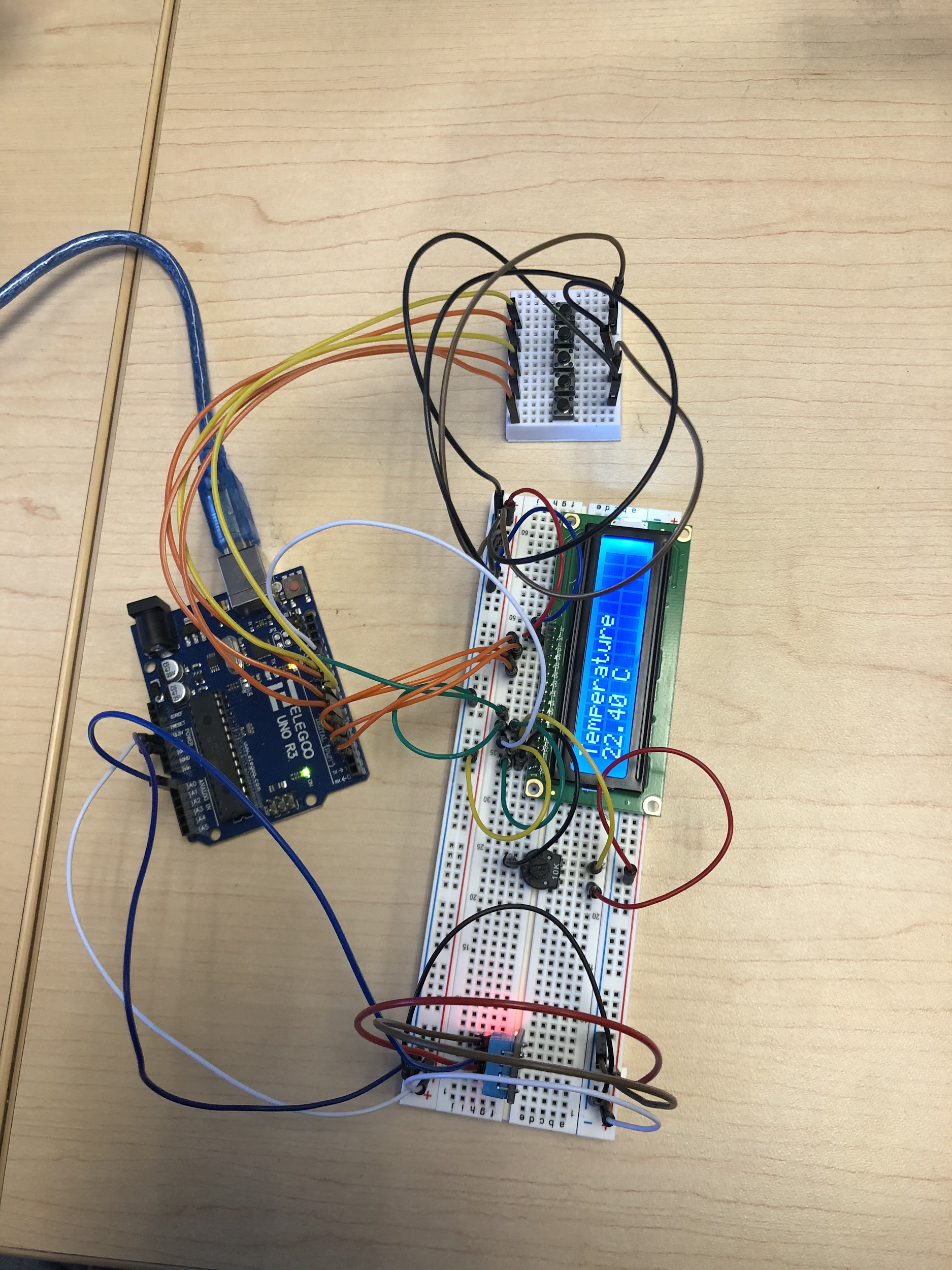
Arduino Clock with Temperature

Adrian Hernandez & Dominic DiPiero | ECET-311-01 | 12/11/2019

**Wiring Diagram**

****

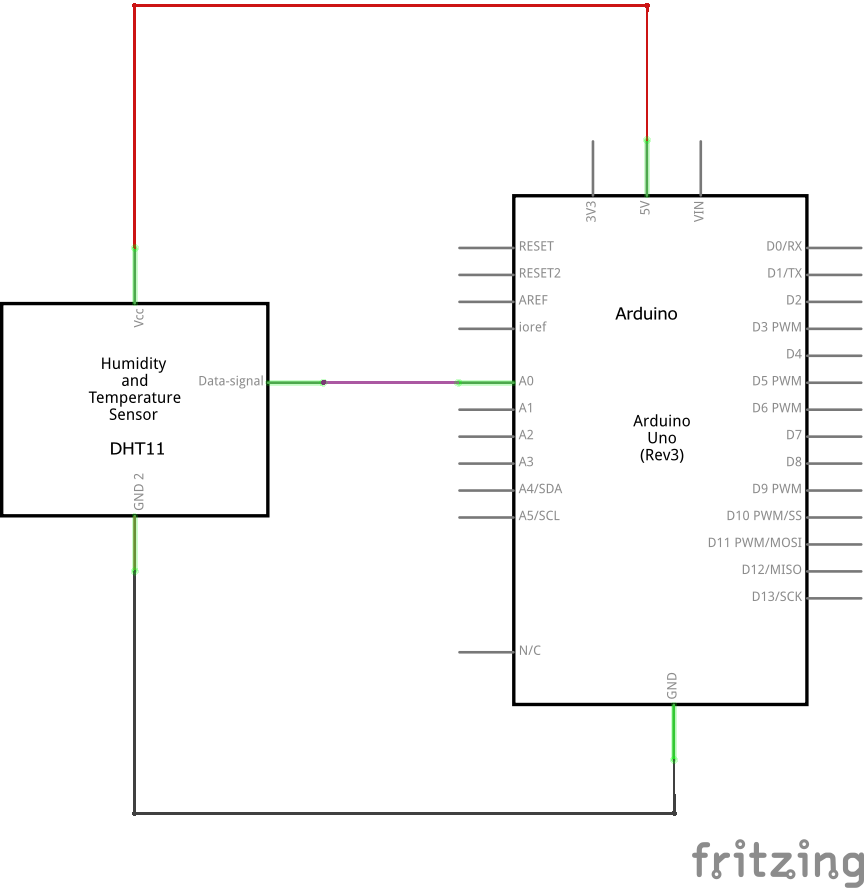
**Circuit Photo**



**Components list**

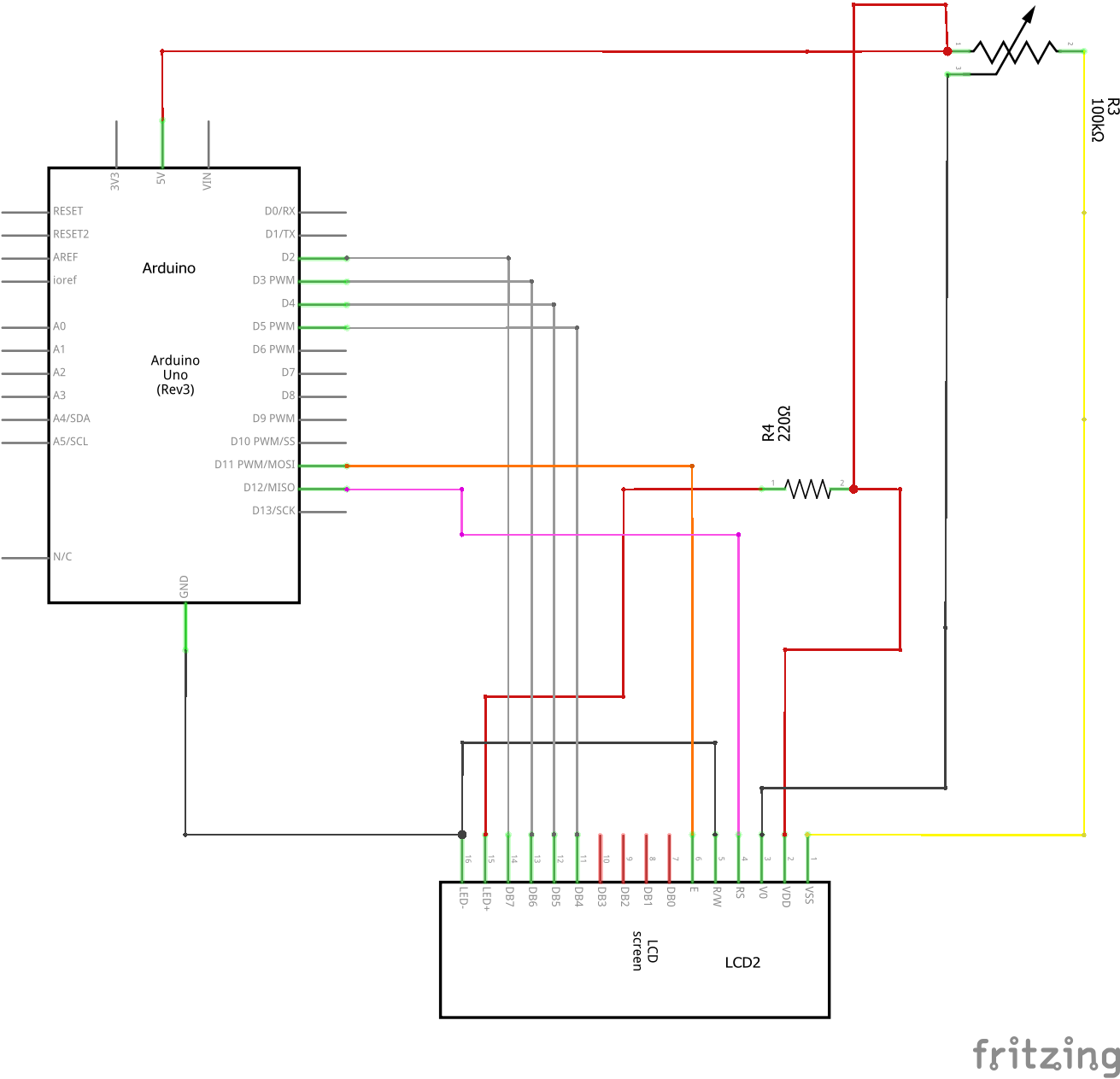
**DHT11 Temperature and Humidity monitor -**

this is the component at the bottom of our circuit picture, it is a small blue rectangle that serves as our temperature monitor. It has three pins on it, the leftmost pin goes to ground, the middle pin goes to analog output and the right in goes to 5 volts.



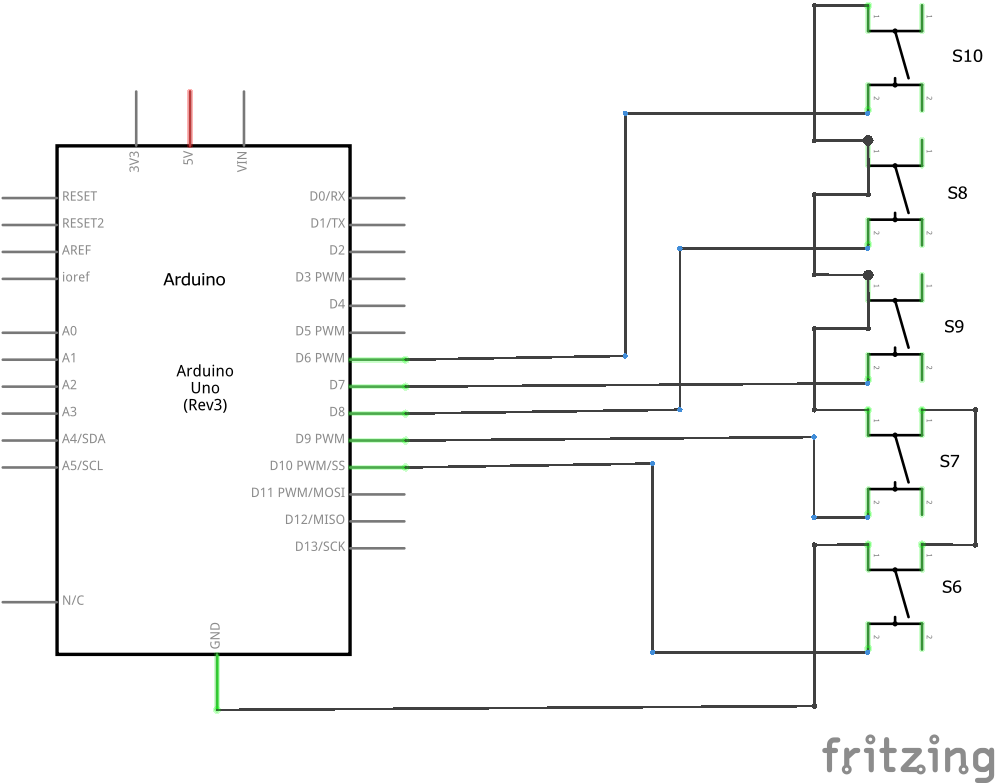
**Liquid Crystal Display-**

The LCD is where we view our output of our whole circuit, whether it be time or temperature. The following photo is a picture of the schematic for the lcd display. The LCD can display information on its 2 rows and 16 columns of display area. The LCD has 16 total pins, not all of which were used. 6 pins labeled D4 -D7(for reading and writing data), E (enabling pin that tells the LCD to carry out instructions such as clearing the display) and RS(register select, which controls where in the LCDs memory we are giving data to) are all connected to digital output pins on the arduino. The K pin which controls the backlight of the LCD, VSS, and RW which controls whether the LCD is in reading or writing mode are all connected to ground. The VDD pin, and the A pin which controls the backlight of the LCD are both connected to 5 volts. Lastly there is a potentiometer connected to the VO pin. The 10KΩ potentiometer has 3 pins, the middle is the one connected to the VO pin, and the other two pins are hooked up to ground and 5V.



**Buttons-**

This circuit required the use of 5 push buttons connected in parallel to one another as shown below. On each button there are 4 pins, one on each corner of the button. One on corner they are hooked up to digital output pins 6-10 on the Arduino. The pin on the opposite corner is connected to ground. In the code these buttons are used different things depending on what mode the code is in, temperature or time. The first button is always used for switching between these two modes and nothing else. When in time mode, the next two buttons are used to increment and decrement the hours of the clock between 1 and 24. The final two buttons increment and decrement the minutes of the clock between 0 and 59 minutes. When in temperature mode, the second button changes between the units of fahrenheit and celsius. Below is a picture of the wiring diagram for just the buttons.



**Code (line by line)**

#include <DHT.h> //includes library for using the DHT sensor

#include <LiquidCrystal.h> // includes the library used for working with LCD

#define DHTPIN A0 // what digital pin we're connected to

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE); // include the library code: // initialize the library with the numbers of the interface pins

LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // tells the arduino which pins to send information to the LCD

long hour = 3600000; // defines the length of an hour in milliseconds

long minute = 60000; // defines the length of a minute in milliseconds

long second = 1000; // defines the length of a minute in milliseconds

int hoursmod = 0; // defines the hours modification variable

int minutesmod = 0; // defines the minutes modification variable

int secondsmod = 0; // defines the seconds modification variable

boolean hourlock=false; // sets a boolean value used for a while condition

boolean clockortemp=true; // sets a boolean value used for a while condition

void setup() {

Serial.begin(9600); // begins the serial monitor at 9600 baud

lcd.begin(16, 2); // set up the LCD's number of columns and rows:

dht.begin(); // tells the dht to begin collecting data and reading the temperature

pinMode(10,INPUT\_PULLUP); // initializes button 1 to pin 10 in pullup mode

pinMode(9,INPUT\_PULLUP); // initializes button 2 to pin 9 in pullup mode

pinMode(8,INPUT\_PULLUP); // initializes button 3 to pin 8 in pullup mode

pinMode(7,INPUT\_PULLUP); // initializes button 4 to pin 7 in pullup mode

pinMode(6,INPUT\_PULLUP); // initializes button 5 to pin 6 in pullup mode

}

void loop() { // beginning of the loop section

lcd.setCursor(0, 1); // sets the lcd cursor to the top left corner

int sensorValue1 = digitalRead(10); // reads the state of button 1 and stores it as a variable

int sensorValue2 = digitalRead(9); // reads the state of button 2 and stores it as a variable

int sensorValue3 = digitalRead(8); // reads the state of button 3 and stores it as a variable

int sensorValue4 = digitalRead(7); // reads the state of button 4 and stores it as a variable

int sensorValue5 = digitalRead(6); // reads the state of button 5 and stores it as a variable

int swap = 0; // initializes the variable that will change the mode from time to temperature

while (sensorValue1 == LOW && swap == 0){ // while loop with the conditions of button 1 has been pressed and the program

// is in time mode

lcd.clear(); // clears display of LCD

lcd.setCursor(0, 0); // sets cursor to the top left

lcd.print("Time"); // write the word time on LCD

lcd.setCursor(0, 1); //sets cursor to bottom left corner

time(); // run time function

delay(1000); // delay one second

sensorValue2 = digitalRead(9); // assigns the state of button 2 to a variable

int sensorValue11 = digitalRead(10); // assigns the state of button 1 to a variable

if (sensorValue11 == LOW){swap = 1;} // if button 1 is pressed, set the swap variable to 1

// if the if statement is satisfied, stop this while loop and start the next one for temperature

}

while (swap == 1){ //used for changing from clock mode to temperature mode

lcd.clear(); //clears the LCD

lcd.setCursor(0, 0); //sets the LCD cursor to row 0 column 0

lcd.print("Temperature"); // the word Temperature is displayed on the LCD

lcd.setCursor(0, 1); // sets the LCD cursor to column 0 row 1

float temp = dht.readTemperature(); //stores the temperature reading from the sensor in a float variable

lcd.print(temp); // This prints the value in the temp variable to the LCD

if (swaptwo == 1){lcd.clear();} // if the condition for Fahrenheit is met clear the LCD

lcd.print(" C"); // The LCD prints C for Celsius

if (swaptwo == 1){lcd.clear();} // if the condition for Fahrenheit is met clear the LCD

delay(500); // delays the code 500 milliseconds (in order to not produce bad temperature values)

sensorValue1 = digitalRead(10); // assigns the state of button 2 to a variable

sensorValue2 = digitalRead(9); // assigns the state of button 1 to a variable

if (sensorValue2 == LOW){swaptwo = 1;} // if the button 2 is pressed the value of the variable swaptwo is changed to 1

while (swap == 1 && swaptwo == 1){ // while the swap varriable (the value that determines time or temp) and swaptwo (the variable that determines Celsius or Fahrenheit) is met do this function

lcd.clear(); //clears the LCD

lcd.setCursor(0, 0); //sets the LCD cursor to row 0 column 0

lcd.print("Temperature"); // the word Temperature is displayed on the LCD

lcd.setCursor(0, 1); // sets the LCD cursor to column 0 row 1

float temp = dht.readTemperature(); //stores the temperature reading from the sensor in a float variable

float tempf = (temp \* (9/5))+32; //this takes the Celsius value from the sensor and converts it via the C to F formula to Fahrenheit then stores it in tempf or Temperature Fahrenheit

lcd.print(tempf); // This prints the value in the tempf variable to the LCD

lcd.print(" F "); // The LCD prints F for Fahrenheit

delay(500); // delays the code 500 milliseconds (in order to not produce bad temperature values)

sensorValue1 = digitalRead(10); // assigns the state of button 2 to a variable

sensorValue2 = digitalRead(9); // assigns the state of button 1 to a variable

if (sensorValue1 == LOW){swap = 0;} //if button 1 is pressed go from temperature in F to Clock

if (sensorValue2 == LOW){swaptwo = 0;} // if button 2 is pressed go from temperature in F to temperature in C

}

if (sensorValue1 == LOW){swap = 0;} // button 1 is pressed go from temperature in C to clock

}

}

void time() { //the beginning of the time function

int sensorValue1 = digitalRead(10); // reads the state of button 1 and stores it as a variable

int sensorValue2 = digitalRead(9); // reads the state of button 2 and stores it as a variable

int sensorValue3 = digitalRead(8); // reads the state of button 3 and stores it as a variable

int sensorValue4 = digitalRead(7); // reads the state of button 4 and stores it as a variable

int sensorValue5 = digitalRead(6); // reads the state of button 5 and stores it as a variable

long timeNow = millis(); //sets the variable timeNow to the milliseconds that have passed since the Arduino board began running

int hours = timeNow / hour; // take the time now in milliseconds and converts it to hours

int minutes = (timeNow % hour) / minute; // takes the time now in milliseconds and converts it to minutes including the remainder from hours

int seconds = ((timeNow % hour) % minute) / second;// takes the time now in milliseconds and converts it to seconds including the remainder from minutes and the remainder from hours

if (sensorValue2 == LOW){hoursmod = hoursmod + 1;} //if button 2 is pressed the hours modification value increases by one which increments the hours displayed by one

if (hours > 24){hours = 0;} // resets hours back to 0 if it goes past 24

if (hoursmod > 24){hoursmod = 0;} // resets the hours modification value back to 0 if it goes past 24

if (sensorValue3 == LOW){hoursmod = hoursmod - 1;}//if button 3 is pressed the hours modification value decreased by one which decrements the hours displayed by one

if (hours < 0){hours = 24;} // if hours are decremented at 0 it goes to 24

if (hoursmod < 0){hoursmod = 24;}// resets the hours modification value back to 24 if it goes below 0

if (sensorValue4 == LOW){minutesmod = minutesmod + 1;}//if button 4 is pressed the minutes modification value increases by one which increments the minutes displayed by one

if (minutes > 59){minutes = 0;} // if minutes are increased past 59, it goes back to 0

if (minutesmod > 59){minutesmod = 0;}// if the minutes modification variable is increased past 59, it goes back to 0

if (sensorValue5 == LOW){minutesmod = minutesmod - 1;}//if button 5 is pressed the minutes modification value decreases by one which decrements the minutes displayed by one

if (minutes < 0){minutes = 59;} // if minutes are decremented below 0 time is reset to 59

if (minutesmod < 0){minutesmod = 59;}// if the minutes modification value is decremented below 0 time is reset to 59

minutes = minutesmod + minutes; //merges the minutes modification value and the real time minutes into the desired value

if (seconds > 59){seconds = 0;} // when the seconds go past 59 they restart at 0

if (minutes >= 59 && hourlock == true) {hourlock = false;} // if the value of the minutes variable to be displayed is greater than or equal to 59 and the boolean operator condition is met change the boolean value so the next line of code can become active

while (minutes >= 59 && hourlock == false && minutes != 0 && seconds >= 59){hoursmod= hoursmod + 1; minutes = 0; hourlock = true; seconds = 0;} // while the minutes variable is greater than or equal to 59 minutes and the boolean value is met and the minutes value does not equal 0 and the seconds value is greater or equal to 59 seconds. Then increment the hours modification value by one and change the minutes value to 0 and change the boolean variable to a state that ends the while and finally sets seconds to 0

if (minutes > 59){minutes = 0;} // if minutes go over 59, they are reset to 0

if (minutesmod > 59){minutesmod = 0;} //if the minutes modification variable goes over 59, they are reset to 0

if (minutes < 0){minutes = 59;} // if minutes are decremented to below 0 they go to 59

if (minutesmod < 0){minutesmod = 59;}// if the minutes modification value is decremented to below 0 they go to 59

if (hours < 0){hours = 24;} // if hours are decremented to below 0 they go to 24

if (hoursmod < 0){hoursmod = 24;}// if the hours modification value is decremented to below 0 they go to 24

if (hours > 24){hours = 0;} // if hours are incremented to above 24 they go to 0

if (hoursmod > 24){hoursmod = 0;} // if the hours modification value is incremented to above 24 they go to 0

hours = hoursmod + hours; //merges the hours modification value and the real time hours into the desired value

LCDprintDigits(hours); // print the hours

lcd.print(":"); // prints “:”

LCDprintDigits(minutes); // print the value in the minutes variable

lcd.print(":"); // prints “:”

LCDprintDigits(seconds); // print the value being stored in the seconds variable

}

void LCDprintDigits(int digits) { // a function used to print to the LCD

if (digits < 10) //if the digit is below 10 making it a single digit so a 0 may be placed in front of it

lcd.print("0"); //LCD print 0

lcd.print(digits); //print the value to the LCD

}

void printDigits(byte digitss) { // a function used to serial print values for troubleshooting purposes

if (digitss < 10) //if the digit is below 10 making it a single digit so a 0 may be placed in front of it

Serial.println('0'); //serial print 0

Serial.println(digitss, DEC); //serial print the value in Decimal form

}

**Segment Explanation**

The first section of the code everything before the “void setup()” line. This section is used as introducing the libraries that are being used in the code as well as which output pins are being used on the arduino and which component they are assigned to. Using the #include function we are telling the computer that we are using the DHT.v and the LiquidCrystal.h libraries that we have found online and downloaded onto the computer. Libraries are what allows the coder to use functions (usually related to a certain component) that are not integrated into the default language. The DHT library allows us to make the DHT read the temperature in the surrounding area and send the information to the arduino. The LCD library allows us to do things found in the code such as clearing the display as well as setting the cursor, which determines where on the LCD it will begin writing the information it is told. Also in this section is the initialization of hours seconds and minutes, which are measured in milliseconds. Finally some global variables are initialized to be used throughout the code which help with making the program fail proof.

#include <DHT.h> //includes library for using the DHT sensor

#include <LiquidCrystal.h> // includes the library used for working with LCD

#define DHTPIN A0 // what digital pin we're connected to

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE); // include the library code: // initialize the library with the numbers of the interface pins

LiquidCrystal lcd(12, 11, 5, 4, 3, 2); // tells the arduino which pins to send information to the LCD

long hour = 3600000; // defines the length of an hour in milliseconds

long minute = 60000; // defines the length of a minute in milliseconds

long second = 1000; // defines the length of a minute in milliseconds

int hoursmod = 0; // defines the hours modification variable

int minutesmod = 0; // defines the minutes modification variable

int secondsmod = 0; // defines the seconds modification variable

boolean hourlock=false; // sets a boolean value used for a while condition

boolean clockortemp=true; // sets a boolean value used for a while condition

The next section is the void setup section. Everything in this section of code is run just once at the beginning of the program. In this section we begin using the serial monitor to keep track of various variables and their current values, which are displayed in the serial monitor window on the computer. The DHT sensor and the LCD display are also told to start collecting data in this section.

void setup() {

Serial.begin(9600); // begins the serial monitor at 9600 baud

lcd.begin(16, 2); // set up the LCD's number of columns and rows:

dht.begin(); // tells the dht to begin collecting data and reading the temperature

pinMode(10,INPUT\_PULLUP); // initializes button 1 to pin 10 in pullup mode

pinMode(9,INPUT\_PULLUP); // initializes button 2 to pin 9 in pullup mode

pinMode(8,INPUT\_PULLUP); // initializes button 3 to pin 8 in pullup mode

pinMode(7,INPUT\_PULLUP); // initializes button 4 to pin 7 in pullup mode

pinMode(6,INPUT\_PULLUP); // initializes button 5 to pin 6 in pullup mode

}

The next section is the void loop segment. This is the piece of the code that is run in a loop, repeating over and over again. This is usually the largest section of code in a program as it is telling the arduino what it should be doing at all times. This section starts by telling the arduino to check the state of the 5 push buttons and store them as variables. The state of these buttons determines the direction the code goes in for that particular loop. Then for troubleshooting purposes, we used the Serial.println functions to show what the buttons were reading in the serial monitor. After the problems we had involving the buttons were solved, we commented these lines out so that they would not be executed. After this the code moves on to two while loops that tell the code to switch between the temperature and time functions. It does this by assigning the variable “swap” to change between 0 and 1 every time it is pressed. When swap is equal to 0 it clears the display and runs the time loop, when it is 1 arduino clears the display and runs the temperature loop. The while loop that contains the temperature is a little more complicated than the time one. This is because the time loop contains an external function that will be talked about later. In the temperature while loop it initializes the buttons that will be used, which is just the first two. Then some variables are set to be displayed in the serial monitor so that it is easier to keep track of them. Inside of this loop is another while loop which revolves around a variable called swaptwo. This is the variable that will help us switch between celsius and fahrenheit. Swaptwo uses the same concept as swap, when it is set to 0, it shows the temperature in Celsius on the LCD, and when it is 1 it converts the temperature from celsius to fahrenheit, and displays this value instead. This loop, when being ran, repeats every half second, updating the temperature each time. The time loop simply prints the word time followed by the time it gets from the external function, updating itself every second. The loop then calls the external function called time.

void loop() { // beginning of the loop section

lcd.setCursor(0, 1); // sets the lcd cursor to the top left corner

int sensorValue1 = digitalRead(10); // reads the state of button 1 and stores it as a variable

int sensorValue2 = digitalRead(9); // reads the state of button 2 and stores it as a variable

int sensorValue3 = digitalRead(8); // reads the state of button 3 and stores it as a variable

int sensorValue4 = digitalRead(7); // reads the state of button 4 and stores it as a variable

int sensorValue5 = digitalRead(6); // reads the state of button 5 and stores it as a variable

int swap = 0; // initializes the variable that will change the mode from time to temperature

while (sensorValue1 == LOW && swap == 0){ // while loop with the conditions of button 1 has been pressed and the program

// is in time mode

lcd.clear(); // clears display of LCD

lcd.setCursor(0, 0); // sets cursor to the top left

lcd.print("Time"); // write the word time on LCD

lcd.setCursor(0, 1); //sets cursor to bottom left corner

time(); // run time function

delay(1000); // delay one second

sensorValue2 = digitalRead(9); // assigns the state of button 2 to a variable

int sensorValue11 = digitalRead(10); // assigns the state of button 1 to a variable

if (sensorValue11 == LOW){swap = 1;} // if button 1 is pressed, set the swap variable to 1

// if the if statement is satisfied, stop this while loop and start the next one for temperature

}

while (swap == 1){ //used for changing from clock mode to temperature mode

lcd.clear(); //clears the LCD

lcd.setCursor(0, 0); //sets the LCD cursor to row 0 column 0

lcd.print("Temperature"); // the word Temperature is displayed on the LCD

lcd.setCursor(0, 1); // sets the LCD cursor to column 0 row 1

float temp = dht.readTemperature(); //stores the temperature reading from the sensor in a float variable

lcd.print(temp); // This prints the value in the temp variable to the LCD

if (swaptwo == 1){lcd.clear();} // if the condition for Fahrenheit is met clear the LCD

lcd.print(" C"); // The LCD prints C for Celsius

if (swaptwo == 1){lcd.clear();} // if the condition for Fahrenheit is met clear the LCD

delay(500); // delays the code 500 milliseconds (in order to not produce bad temperature values)

sensorValue1 = digitalRead(10); // assigns the state of button 2 to a variable

sensorValue2 = digitalRead(9); // assigns the state of button 1 to a variable

if (sensorValue2 == LOW){swaptwo = 1;} // if the button 2 is pressed the value of the variable swaptwo is changed to 1

while (swap == 1 && swaptwo == 1){ // while the swap varriable (the value that determines time or temp) and swaptwo (the variable that determines Celsius or Fahrenheit) is met do this function

lcd.clear(); //clears the LCD

lcd.setCursor(0, 0); //sets the LCD cursor to row 0 column 0

lcd.print("Temperature"); // the word Temperature is displayed on the LCD

lcd.setCursor(0, 1); // sets the LCD cursor to column 0 row 1

float temp = dht.readTemperature(); //stores the temperature reading from the sensor in a float variable

float tempf = (temp \* (9/5))+32; //this takes the Celsius value from the sensor and converts it via the C to F formula to Fahrenheit then stores it in tempf or Temperature Fahrenheit

lcd.print(tempf); // This prints the value in the tempf variable to the LCD

lcd.print(" F "); // The LCD prints F for Fahrenheit

delay(500); // delays the code 500 milliseconds (in order to not produce bad temperature values)

sensorValue1 = digitalRead(10); // assigns the state of button 2 to a variable

sensorValue2 = digitalRead(9); // assigns the state of button 1 to a variable

if (sensorValue1 == LOW){swap = 0;} //if button 1 is pressed go from temperature in F to Clock

if (sensorValue2 == LOW){swaptwo = 0;} // if button 2 is pressed go from temperature in F to temperature in C

}

if (sensorValue1 == LOW){swap = 0;} // button 1 is pressed go from temperature in C to clock

}

}

This function is where everything that has to do with the clock function is written. First up is the setting up of the normal, unmodified clock. This is done by dividing the time elapsed from when the program was uploaded by an hour which is 3600000 millisecond, the result is an integer that is used as hours, this number is then divided by 60000 and stored as minute, then divided by 1000 to get our second value. These values are locked in place so that they cant be modified until they are unlocked in an if loop in the next set of lines. These next few lines are a series of short if loops that state if one of the modification buttons are pressed, then that value is unlocked, changed and then locked again, without changing any of the other values. There are also limiting loops that when minutes get to 59 they do not go to 60, but instead back to 0. The same is done for hours but the limit is 24 in this case. The line by line comments on the code went into further detail on this part, as it is quite complicated.

void time() { //the beginning of the time function

int sensorValue1 = digitalRead(10); // reads the state of button 1 and stores it as a variable

int sensorValue2 = digitalRead(9); // reads the state of button 2 and stores it as a variable

int sensorValue3 = digitalRead(8); // reads the state of button 3 and stores it as a variable

int sensorValue4 = digitalRead(7); // reads the state of button 4 and stores it as a variable

int sensorValue5 = digitalRead(6); // reads the state of button 5 and stores it as a variable

long timeNow = millis(); //sets the variable timeNow to the milliseconds that have passed since the Arduino board began running

int hours = timeNow / hour; // take the time now in milliseconds and converts it to hours

int minutes = (timeNow % hour) / minute; // takes the time now in milliseconds and converts it to minutes including the remainder from hours

int seconds = ((timeNow % hour) % minute) / second;// takes the time now in milliseconds and converts it to seconds including the remainder from minutes and the remainder from hours

if (sensorValue2 == LOW){hoursmod = hoursmod + 1;} //if button 2 is pressed the hours modification value increases by one which increments the hours displayed by one

if (hours > 24){hours = 0;} // resets hours back to 0 if it goes past 24

if (hoursmod > 24){hoursmod = 0;} // resets the hours modification value back to 0 if it goes past 24

if (sensorValue3 == LOW){hoursmod = hoursmod - 1;}//if button 3 is pressed the hours modification value decreased by one which decrements the hours displayed by one

if (hours < 0){hours = 24;} // if hours are decremented at 0 it goes to 24

if (hoursmod < 0){hoursmod = 24;}// resets the hours modification value back to 24 if it goes below 0

if (sensorValue4 == LOW){minutesmod = minutesmod + 1;}//if button 4 is pressed the minutes modification value increases by one which increments the minutes displayed by one

if (minutes > 59){minutes = 0;} // if minutes are increased past 59, it goes back to 0

if (minutesmod > 59){minutesmod = 0;}// if the minutes modification variable is increased past 59, it goes back to 0

if (sensorValue5 == LOW){minutesmod = minutesmod - 1;}//if button 5 is pressed the minutes modification value decreases by one which decrements the minutes displayed by one

if (minutes < 0){minutes = 59;} // if minutes are decremented below 0 time is reset to 59

if (minutesmod < 0){minutesmod = 59;}// if the minutes modification value is decremented below 0 time is reset to 59

minutes = minutesmod + minutes; //merges the minutes modification value and the real time minutes into the desired value

if (seconds > 59){seconds = 0;} // when the seconds go past 59 they restart at 0

if (minutes >= 59 && hourlock == true) {hourlock = false;} // if the value of the minutes variable to be displayed is greater than or equal to 59 and the boolean operator condition is met change the boolean value so the next line of code can become active

while (minutes >= 59 && hourlock == false && minutes != 0 && seconds >= 59){hoursmod= hoursmod + 1; minutes = 0; hourlock = true; seconds = 0;} // while the minutes variable is greater than or equal to 59 minutes and the boolean value is met and the minutes value does not equal 0 and the seconds value is greater or equal to 59 seconds. Then increment the hours modification value by one and change the minutes value to 0 and change the boolean variable to a state that ends the while and finally sets seconds to 0

if (minutes > 59){minutes = 0;} // if minutes go over 59, they are reset to 0

if (minutesmod > 59){minutesmod = 0;} //if the minutes modification variable goes over 59, they are reset to 0

if (minutes < 0){minutes = 59;} // if minutes are decremented to below 0 they go to 59

if (minutesmod < 0){minutesmod = 59;}// if the minutes modification value is decremented to below 0 they go to 59

if (hours < 0){hours = 24;} // if hours are decremented to below 0 they go to 24

if (hoursmod < 0){hoursmod = 24;}// if the hours modification value is decremented to below 0 they go to 24

if (hours > 24){hours = 0;} // if hours are incremented to above 24 they go to 0

if (hoursmod > 24){hoursmod = 0;} // if the hours modification value is incremented to above 24 they go to 0

hours = hoursmod + hours; //merges the hours modification value and the real time hours into the desired value

LCDprintDigits(hours); // print the hours

lcd.print(":"); // prints “:”

LCDprintDigits(minutes); // print the value in the minutes variable

lcd.print(":"); // prints “:”

LCDprintDigits(seconds); // print the value being stored in the seconds variable

}

The other main function that is called whenever something needs to be displayed on the LCD is the LCDprintDigits function, which is used in most of the sections of the code where and output is needed. This function takes values given to it and either prints it as it is or prints it with a zero in front of it if the number is less than 10 or a single digit.

void LCDprintDigits(int digits) { // a function used to print to the LCD

if (digits < 10) //if the digit is below 10 making it a single digit so a 0 may be placed in front of it

lcd.print("0"); //LCD print 0

lcd.print(digits); //print the value to the LCD

}

The last function is a troubleshooting function used to make displaying variables in the serial monitor easier and more formatted. This function similar to the LCDprintDigits takes values given to it and either prints it as it is or prints it with a zero in front of it if the number is less than 10 or a single digit.

void printDigits(byte digitss) { // a function used to serial print values for troubleshooting purposes

if (digitss < 10) //if the digit is below 10 making it a single digit so a 0 may be placed in front of it

Serial.println('0'); //serial print 0

Serial.println(digitss, DEC); //serial print the value in Decimal form

}