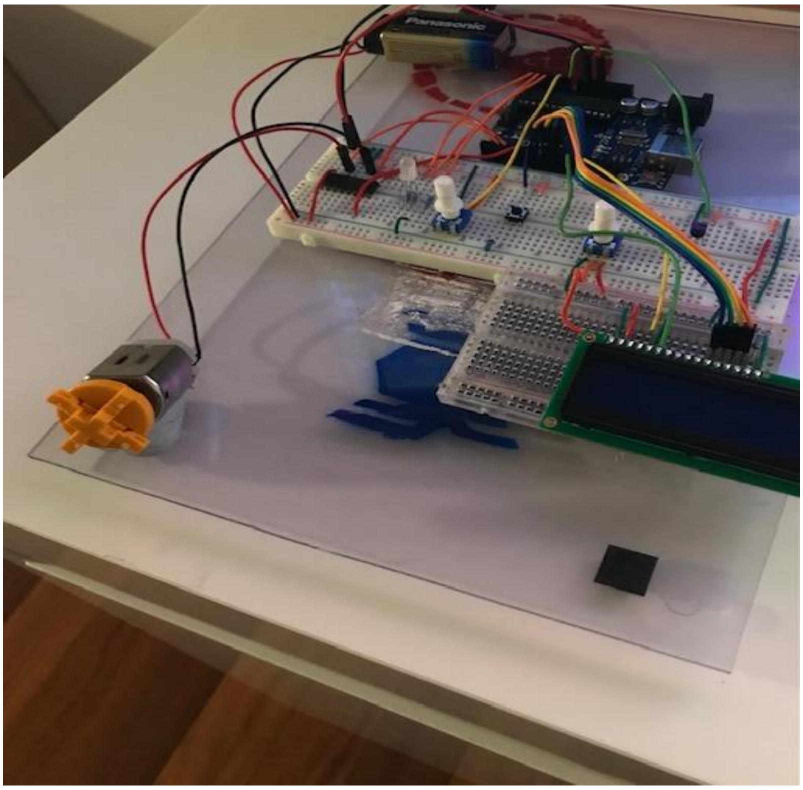
ELE1301 Computer Engineering

Portfolio 3

Temperature sensing and fan system.



Contents

1. Specification

a. Problem Statement

The system is a desk assistant with fan and temperature sensor. When the temperature is over 26 degrees it enables the fan, the red RGB led, displays the current temperature and a randomly chosen message from an array to assist the user in what they might need to do according to the temperature sensed. When the temperature is under 17 degrees, the system will disable the fan, illuminate the blue RGB led and display a randomly chosen message from an array to assist the user again. Otherwise, will randomly flash the RGB and the system will display a randomly chosen messages of affirmations from an array, as well as the current temperature.

b. Input and Output Connections

Arduino Digital Pins:

0	Unused
1	Unused
2	Output, Motor control Pin (on H Bridge motor driver)
3	Output, Motor Control Pin (on H Bridge motor driver)
4	Unused
5	Output, Motor enable Pin(on H Bridge motor driver)
6	Input, Start/stop Pushbutton (for Motor)
7	Output, LCD DB7(Data output line)
8	Output, LCD DB6(Data output line)
9	Output, LCD DB5(Data output line)
10	Output, LCD DB4(Data output line)
11	Output, LCD Enable Signal
12	Output, LCD Register Selection

Arduino Analog Pins:

A0	Potentiometer for fan speed
A1	Temperature Sensor
A2	Unused
A3	Output, RGB Red
A4	Output, RGB Blue
A5	Output, RGB Green

c. Requirements

This system will utilise the temperature sensor to determine which state it will be in. When a desired temperature has not been reached, or the temperature sensed is above 26, the user will be able activate the DC motor by holding the pushbutton long enough to bypass a delay. The motor will be able to be switched off by the same system. The DC motor will run in proportion to which the potentiometer value has been set, which is to be used in addition to the on/off push button.

When the Temperature is below 17, then the DC motor system will be deactivated.

2. Program Development

a. Analogue Input scaling

The system utilises a temperature sensor (TMP 36). The following information was extracted from the data sheet provided by Arduino starter kit. The TMP 36 has a typical accuracy of ±1°C however with a max of ±3°C. To obtain a scaled value from the temperature sensor, the equation becomes

(voltage – offset)/gradient, to evaluate the voltage the system takes the raw value (from 0 – 1023) multiplies it by 5 then divides it by 1024. The offset value of the TMP 36 is 0.5V as stated in the datasheet. The gradient is calculated by, shown in figure 1, the run and rise of the temperature divided by the output voltage. Where the difference of the output voltage is then divided by the difference in the temperature values. (1.2/125) Which I then tinkered with and ended up with 1.2/121.125.

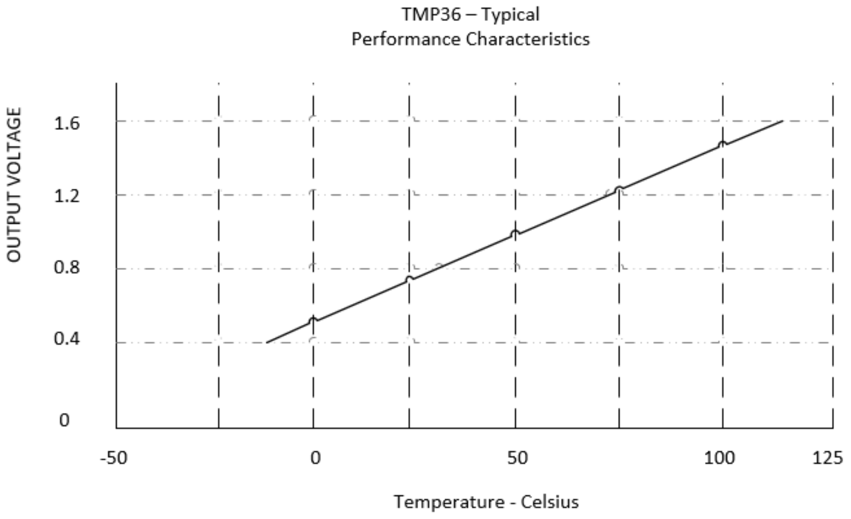


Figure 1. Typical Performance Characteristics

TMP 36 Output Characteristics		
Offset Voltage(V)	**Output Voltage Scaling(mV/°C)**	**Output Voltage @ 25°C (mV)**
.5	10	750

b. Analogue Output scaling

For the motor speed control, the system utilised the H-bridge motor driver in conjunction with a potentiometer. For the conversion of potentiometer for the motor driver, for the system I have implemented the use of the map built in function. Which takes the in raw information from the potentiometer (0 -1023) and converts it into an 8 bit binary number (0 - 255), utilising Pulse Width Modulation (PWM).

c. *Assumptions and Clarifications*

The Following assumptions and clarifications are to be made:

- The system has allocated breaks where reasonable to account for the inaccuracy of the TMP 36. The system has accounted for the worst possible accuracy, to try and make less jumps between states whilst the temperature changes.
- For clarification the RGB LED can be used as a signal to real time changes that are happening to the temperature sensor. As the messages have a timer attached to them, if there is a sudden change in the temperature sensor then a delay will be detected.
- This is system is built under the assumption that unlike the testing conditions the temperature will not vary as quickly as I have been testing it.

3. Program Documentation

a. *Program Description*

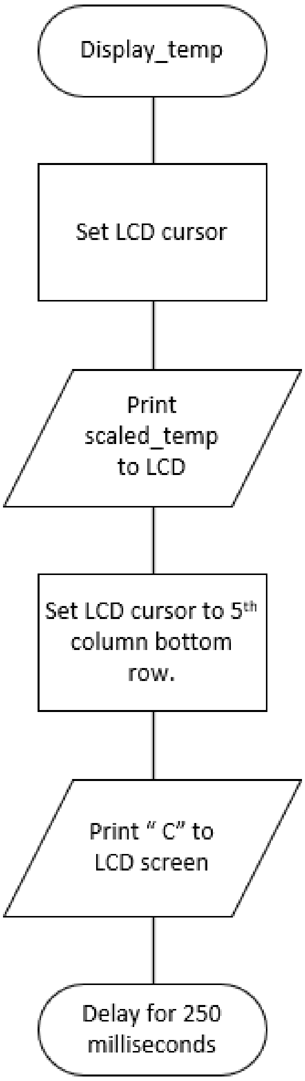
The program utilises arrays to display messages for the user, initialises constants to be used throughout the program and declares global variables to be used within functions and the main loop.

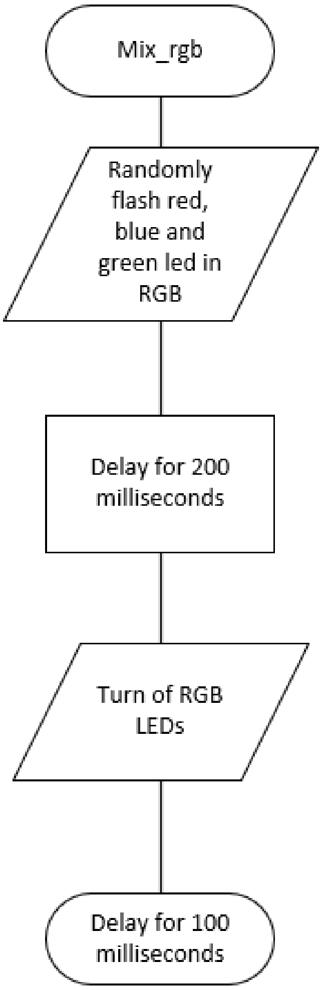
The setup function runs once, configures the pins, and initialises the LCD, as well as displaying a welcome message for the LCD on start-up.

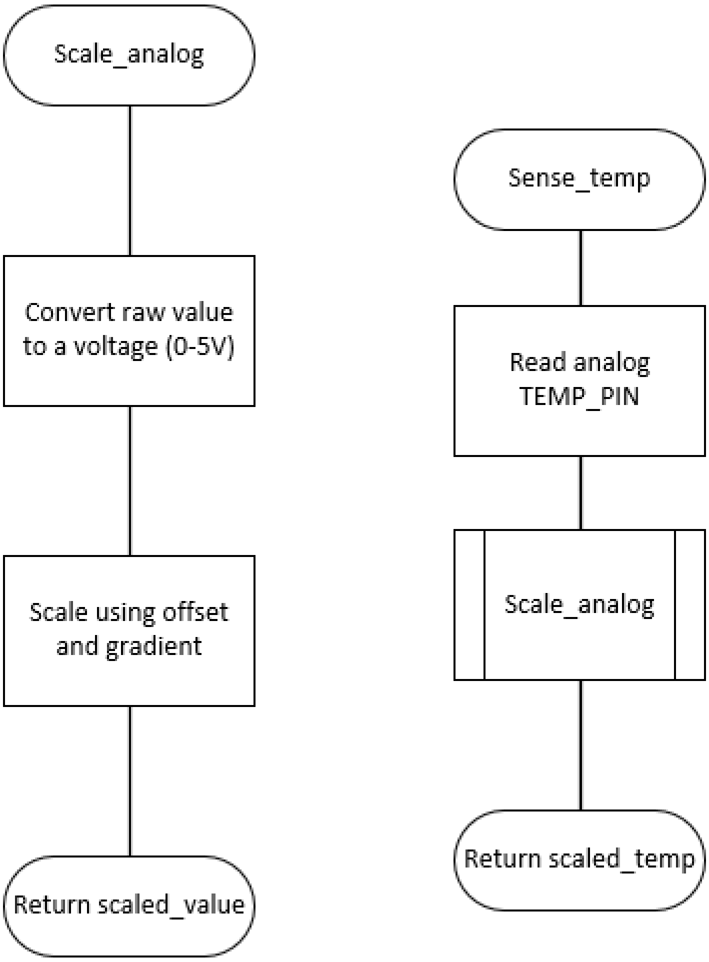
Within the loop function, the built in millis() function will be called to establish a current time to be used instead of a delay, so as not to put too much delay on the push button start motor function. The function to sense the temperature will be called and stored in a float value. Then a random message picked from an array and the current temperature will be displayed on the LCD. Using a while loop for the conditions that are needed to be met in order to change the functionality of the system. One of the stated conditions, is the temperature sensor reading over 26 degrees. When this happens the RGB LEDs blue and green will be switched low, and the red will be switched to high. Subsequently the motor function will be enabled, push button and potentiometer can be utilised to turn on and change the speed of the fan. Then the temperature will be sensed and assigned to a variable, a break is included to check if the temperature has dropped more than 4 degrees, to allow for the max ±3°C accuracy of the TMP36. Shortly thereafter if the time requirements have been met the LCD will illuminate with the current temp and a random message, warning the user of the heat. If the conditions of next while loop have been met, which is the temperature has dropped below 17 degrees. The motor will be disabled immediately and the RGB LEDs red and green will be switched low however the blue will be switched high. The temperature will be read and stored. Again, for the same reasons as above a break is include however, it is used if the temperature sensed has risen 4 degrees.

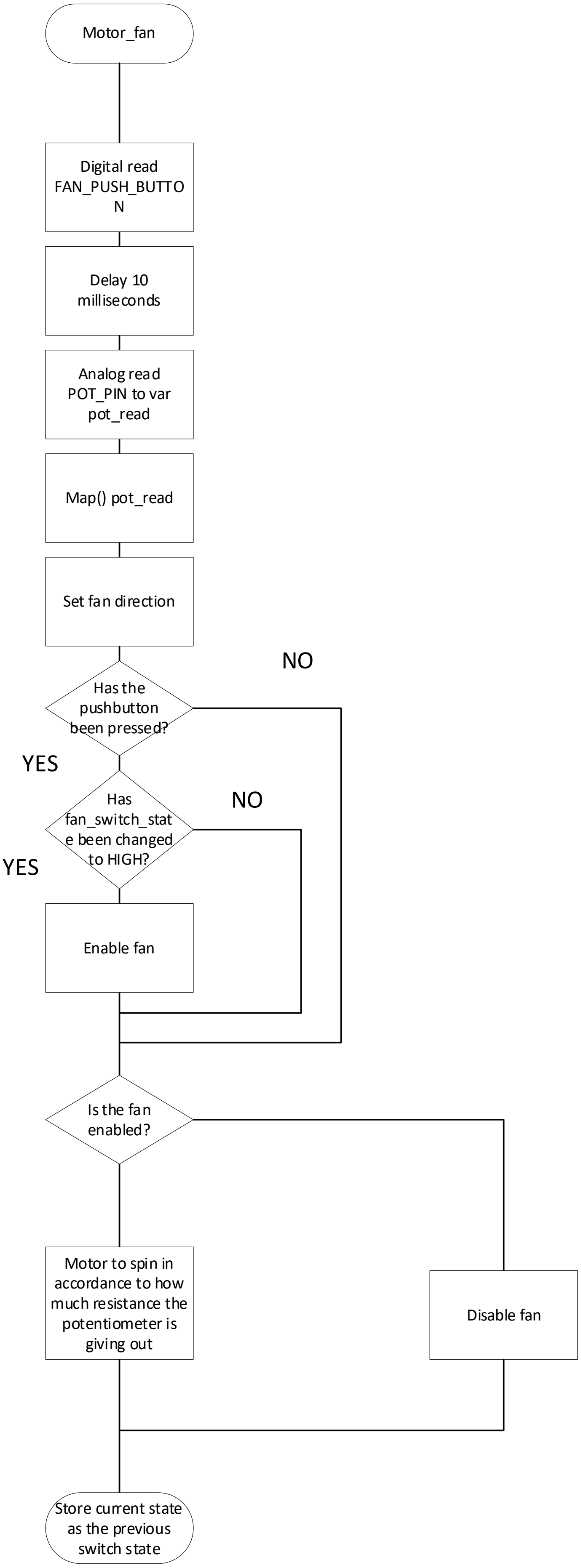
If neither condition has been met the program will loop back to the start again.

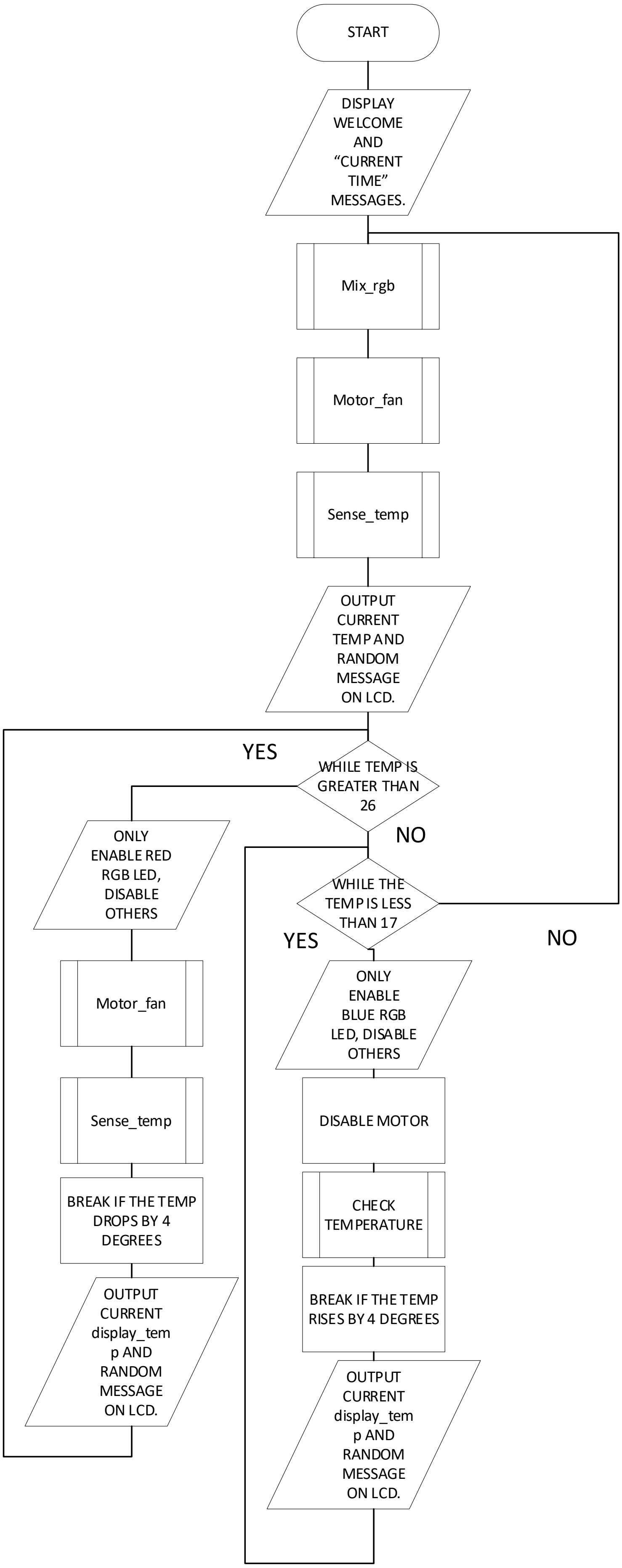
b. Flowcharts




Mix_rgb
Randomly flash red, blue and green led in RGB
Delay for 200 milliseconds
Turn of RGB LEDs
Delay for 100 milliseconds


Scale_analog
Convert raw value to a voltage (0-5V)
Scale using offset and gradient
Return scaled_value
Sense_temp
Read analog TEMP_PIN
Scale_analog
Return scaled_temp


Motor_fan
Digital read FAN_PUSH_BUTTO N
Delay 10 milliseconds
Analog read POT_PIN to var pot_read
Map() pot_read
Set fan direction
Has the pushbutton been pressed?
NO
YES
Has fan_switch_stat e been changed to HIGH?
NO
YES
Enable fan
Is the fan enabled?
Motor to spin in accordance to how much resistance the potentiometer is giving out
Disable fan
Store current state as the previous switch state


START
DISPLAY WELCOME AND "CURRENT TIME" MESSAGES.
Mix_rgb
Motor_fan
Sense_temp
OUTPUT CURRENT TEMP AND RANDOM MESSAGE ON LCD.
YES
WHILE TEMP IS GREATER THAN 26
NO
ONLY ENABLE RED RGB LED, DISABLE OTHERS
Motor_fan
Sense_temp
BREAK IF THE TEMP DROPS BY 4 DEGREES
OUTPUT CURRENT display_temp AND RANDOM MESSAGE ON LCD.
WHILE THE TEMP IS LESS THAN 17
YES
NO
ONLY ENABLE BLUE RGB LED, DISABLE OTHERS
DISABLE MOTOR
CHECK TEMPERATURE
BREAK IF THE TEMP RISES BY 4 DEGREES
OUTPUT CURRENT display_temp AND RANDOM MESSAGE ON LCD.

4. Testing Procedure

a. Temperature sensor accuracy

To ensure that the TMP 36 was reading as accurately as possible, the Tinkercad website was utilised with its ability to change the degrees it was potentially reading. From there I took the information gathered from the datasheet and slowly changed the gradient until I reached a number that was acceptable.

b. Motor speed development

For the motor speed to potentiometer variation, I utilised a built-in function called map (). Which takes the in raw information from the potentiometer (0 - 1023) and converts it into an 8 bit binary number (0 - 255), utilising Pulse Width Modulation (PWM). After some testing, and finding that near the lower end of 50% duty cycle was necessary to get the motor to spin was I then implement that 50% duty cycle was a better idea for the system than starting from zero.

c. Motor to push button delay timing.

The system I had originally built left some residual delay in the code. Therefore, creating a long wait time while starting the motor through pressing the push button. The delays that had been utilised in the system had to either be shaved down or replaced with a timer. I then continued that process until the push button delay was an acceptable amount of time for the user.

d. Program Correctly Shifted between states.

For the appropriate shifting between temperatures, I have conducted a series of tests and recorded those below.

Test ID	Objective	Procedure	Expected Result	Actual Result	Pass/Fail	Notes
1	Start up into a temperature range between 25.9 and 17.1.	USB to be inserted into Arduino	1. Arduino starts up 2. LCD to turn on and welcome message to appear.	Result as expected	pass	

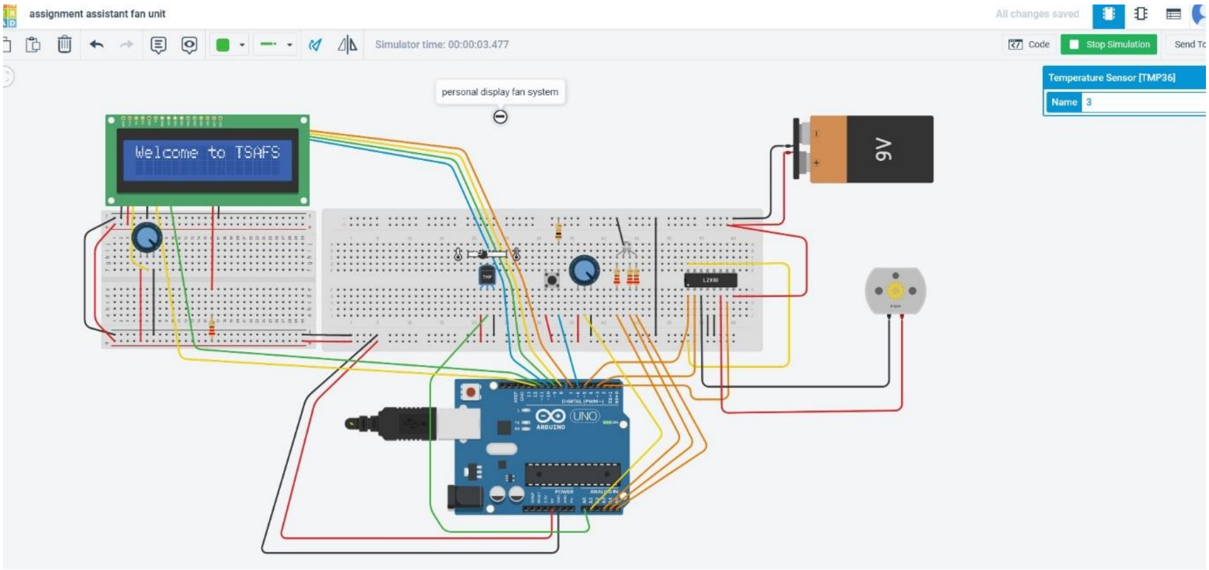
			3. Current temp message to appear. 4. RGB Flashing lights 5. Motor enables and able to activate/deactivate. Change speed as required. 6. Scaled_temp to appear on bottom row of LCD to appear every 5 seconds. 7. Random message from “norm ” array to appear on top row of LCD every 5 seconds.			
2	Shifting from normal range to greater than 26°C	a) Raise temp for the temp sensor by holding sensor. b) Activate push button to test motor. c) Turn the potentio meter to test fan speed	1. RGB to display solid red LED. 2. Motor enables and able to activate/deactivate. Change speed as required. 3. Scaled_temp to appear on bottom row of LCD to appear every 5 seconds. 4. Random message from “hot” array to appear on top row of LCD every 5 seconds.	Results as expected	PASS	To conduct this test I held on to TMP36. To mitigate the inaccuracy of the TMP 36 I have set a break that is 4 degrees below the entry conditions of the while loop.

		d) Press push button again to deactivate.				
3	Shifting from the normal range to the less than 17°C range.	a) Hold cold item to temp sensor b) Press push button to test the motor is disabled.	1. RGB to display solid BLUE LED. 2. Motor disabled. 3. Scaled_temp to appear on bottom row of LCD to appear every 5 seconds. 4. Random message from "hot" array to appear on top row of LCD every 5 seconds.	Results as expected.	PASS	I used an ice cube in a bag to test. To mitigate the inaccuracy of the TMP 36 I have set a break that is 4 degrees above the entry conditions of the while loop. There is some delay between messages on the LCD however it is to be expected as the LCD refreshes every 5 seconds.
4	Shifting from the normal range to the less than 17°C range. With the motor running.	a) In normal range switch motor on. b) Hold cold item to TMP 36 c) observe	1. Motor to be on from normal range. 2. RGB Flashing 3. Once cold item is held then the temperature should drop below 17. 4. RGB solid blue. 5. Once that happens the motor should switch off by itself	Results somewhat as expected.	PASS	whilst the temperature was changing back I noticed that the RGB LEDs had reverted to the "normal" state, however the motor had not been able to be witched back on and I believe it has

						to do with the delays in place and the TMP 36s inaccuracy.

e. Screen shots of Tinkercad Circuit in Operation

I. Welcome message



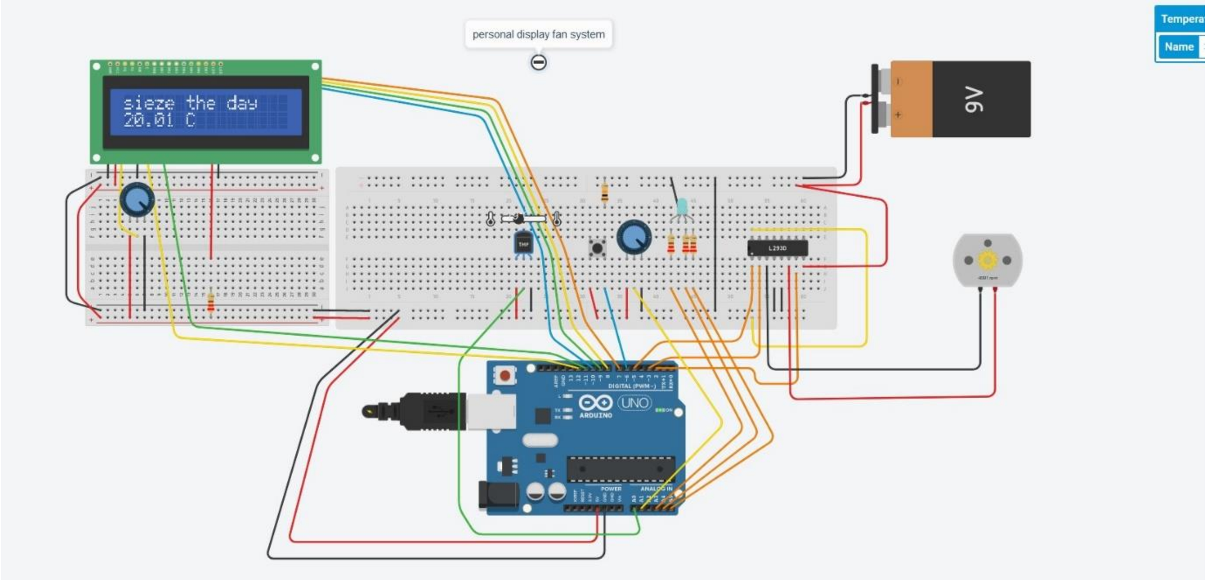
Welcome to TSAFS

Temperature Sensor [TMP36]

Name 3

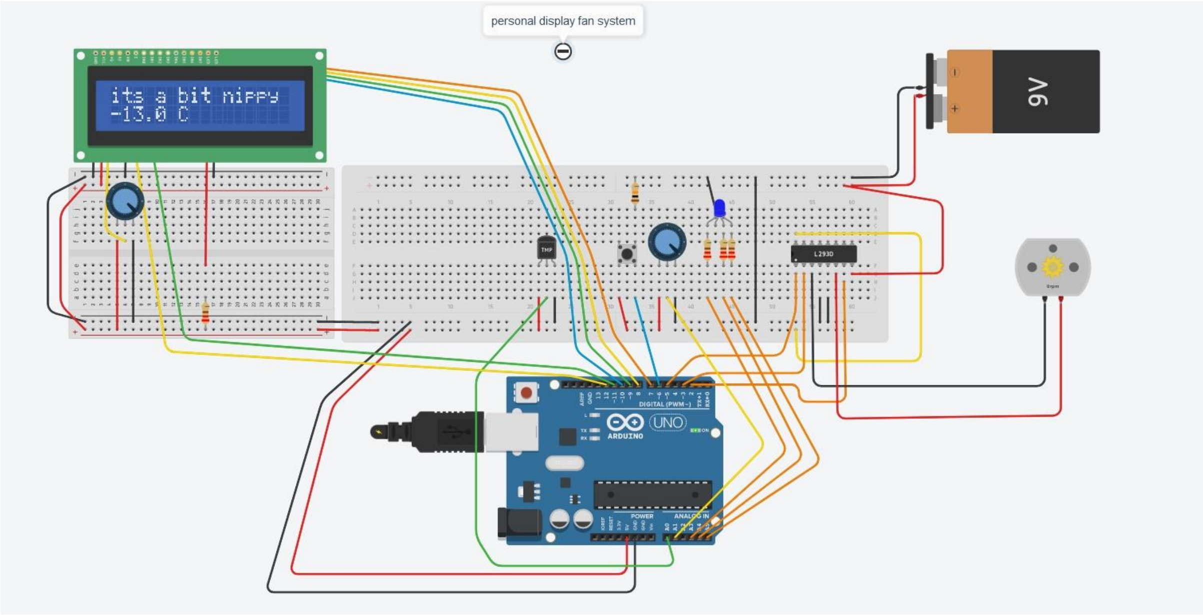
assignment assistant fan unit
All changes saved
Simulator time: 00:00:03.477
Code
Stop Simulation
Send To
personal display fan system
9V
L293D
UNO
ARDUINO

II. Normal State with motor spinning.

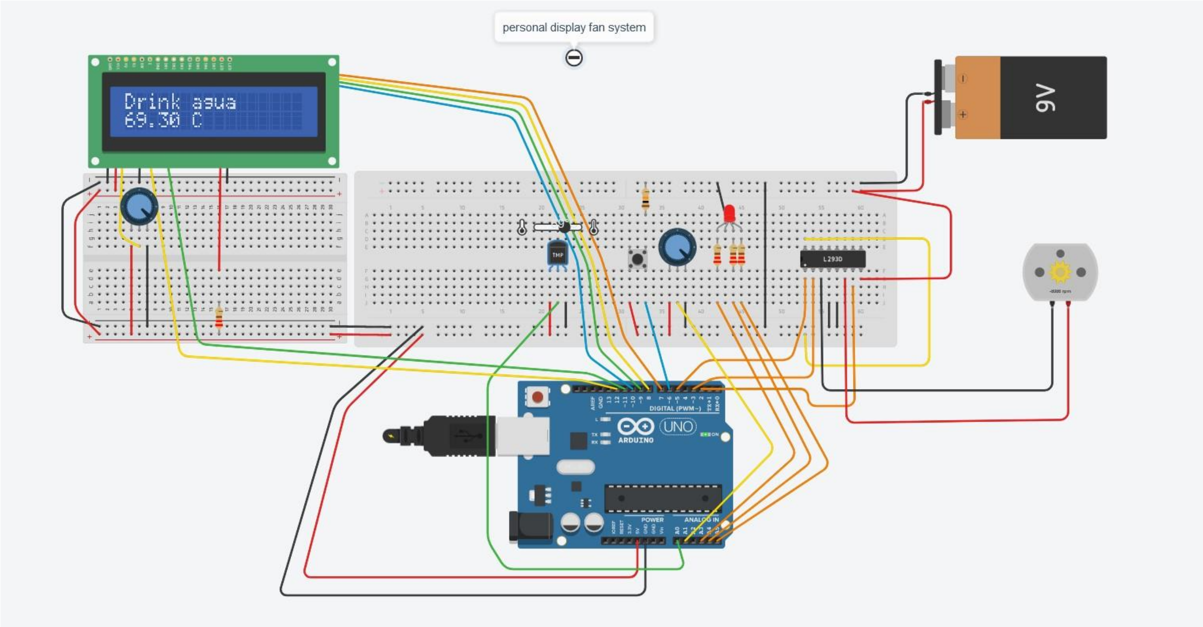


sieze the day
20.01 C

III. State where the Temperature sensor is below 17 degrees



IV. State where Temperature sensor is greater than 26 degrees



f. Tinkercad share link.

https://www.tinkercad.com/things/2s4Uc4aS2Z4

5. Program Code Listing

```
/*
 * Portfolio 3
 * Written By Andre Garcia
 *
 * Temperature sensing fan system
 *
 * Senses the current temperature, displays a random message drawn from the array, then displays it and
 * the current temperature on the LCD.
 *
 */
//declare libraries
#include <LiquidCrystal.h>

//declare arrays
String hot[4] = {"Keep cool","Drink agua","its hot","Stay Hydrated"};
String cold[4] = {"brrrrr!!","its a bit nippy","blanket time?","cold yet?"};
String norm[10] = {"b-e-a-u-tiful","you do you boo","sieze the day","keep going!", "you're Great!","go get em tiger!","( {- O__O -} )","( o _ o )","Nice Job!",""};
```

```
//declare constants
#define POT_PIN A1
#define TEMP_PIN A0
#define RED_RGB_LED A3
#define BLUE_RGB_LED A4
#define GREEN_RGB_LED A5
#define FAN_PUSH_BUTTON 6
#define FAN_ENABLE_PIN 5
#define FAN_CONT_PIN_1 3
#define FAN_CONT_PIN_2 2
double const OFFSET = 0.5;
double const GRADIENT = 1.2/121.125;
int const disp_period = 5000;

//declare variables
int fan_enabled = 0;
int fan_switch_state = 0;
int prev_fan_switch_state = 0;
double pot_read = 0;
unsigned long prev_time = 0;


//assigning pins on the arduino to the LCD.
```

```
LiquidCrystal lcd(12, 11, 10, 9, 8, 7);


void setup() {
 //serial is being used for troubleshooting
 Serial.begin(9600);

 //setup for lcd
 lcd.begin(16, 2);
 lcd.clear();

 //declaring outpus
 pinMode(FAN_ENABLE_PIN,OUTPUT);
 pinMode(GREEN_RGB_LED,OUTPUT);
 pinMode(RED_RGB_LED,OUTPUT);
 pinMode(BLUE_RGB_LED,OUTPUT);
 pinMode(FAN_CONT_PIN_1,OUTPUT);
 pinMode(FAN_CONT_PIN_2,OUTPUT);

 // declaring inputs
 pinMode(FAN_PUSH_BUTTON, INPUT);

 //display hello message on lcd top row
 lcd.setCursor(0, 0);
 lcd.print("Welcome to TSAFS");

 delay(5000);



 //display message for current temp on lcd bottom row

 lcd.clear();

 lcd.print("The current ");

 lcd.setCursor(0, 1);

 lcd.print("Temperature is ");

 delay(2500);

 lcd.clear();

}





void loop() {

 // initialising millis, to establish a current time

 unsigned long current_time = millis();



 // running of the random RGB function

 mix_rgb();



 //functiion to be able to turn the fan on and off as well as change the speed.

 motor_fan();
 //capturing the from the function reading the sensor information for the temperature sensor.
 float scaled_temp = sense_temp();


   //display temp then randomly choose a message from a designated array to display on lcd, if the timing requirements are met.
   if(current_time - prev_time >= disp_period){
      lcd.clear();
      display_temp(scaled_temp);
      lcd.setCursor(0, 0);
      lcd.print(norm[random(0, 9)]);
      delay(150);
      prev_time = current_time;
   }


  //while statement to check temperature is greater than 25, illuminate red RGB led, display a random message and the temperature.
  while(scaled_temp > 26){
    // to keep track of timer.
    current_time = millis();


    //blue and green led to be switched off and red switched on to further signify the heat
    digitalWrite(BLUE_RGB_LED, LOW);
    digitalWrite(GREEN_RGB_LED,LOW);
    digitalWrite(RED_RGB_LED, HIGH);


    //functiion to be able to turn the fan on and off as well as change the speed.

    motor_fan();


    // reading the sensor information for the temperature sensor

    scaled_temp = sense_temp();


      //insert a break so we do not get stuck in while loop

    if(scaled_temp < 22)

      break;


      //display temp then randomly choose a message from a designated array to display on
   lcd, if the timing requirements are met.

      if(current_time - prev_time >= disp_period){

       lcd.clear();

       display_temp(scaled_temp);

       lcd.setCursor(0, 0);

       lcd.print(hot[random(0, 3)]);

       delay(150);

       prev_time = current_time;

    }
  }


  //if state to check temp is less than 17 then disable fan, illuminate blue RGB led display lcd message, and temperature.
  while(scaled_temp < 17){
    // to keep track of timer
    current_time = millis();


    //disabling the fan
     analogWrite(FAN_ENABLE_PIN, 0);


    //blue led to be on, to visually show its cold. and turn red off
    digitalWrite(BLUE_RGB_LED, HIGH);
    digitalWrite(GREEN_RGB_LED,LOW);
    digitalWrite(RED_RGB_LED, LOW);



    // reading the sensor information for the temperature sensor
    scaled_temp = sense_temp();


    //insert a break so we do not get stuck in while loop
    if(scaled_temp > 21)
      break;
    //display temp then randomly choose a message from a designated array to display on lcd, if the timing requirements are met.
    if(current_time - prev_time >= disp_period){
      lcd.clear();
      display_temp(scaled_temp);
      lcd.setCursor(0, 0);
      lcd.print(cold[random(0, 3)]);
      delay(150);
      prev_time = current_time;
    }

 }

}

/*
*Function to scale analog input
*/

double scale_analog(int raw, double offset, double gradient)
{
  double scaled_value, voltage;
  voltage = double(raw)*5/1024;
  scaled_value = (voltage - offset)/gradient;
  return(scaled_value);
}


/*
 *Function to display current temperature on the LCD.
 */
void display_temp(float temp)
{
  lcd.setCursor(0, 1);
  lcd.print(temp);
  lcd.setCursor(5, 1);
  lcd.print(" C");
  delay(250);
}


/*
 * Function designed to activate the motor to act as a fan, at the press of the PushButton.
 * once activated the potentiometer connected can be used as a fan speed dial.
 * check if push button for fan has been pressed and then store the change into fan_enabled variable.
 */
void motor_fan(){
  fan_switch_state = digitalRead(FAN_PUSH_BUTTON);

  delay(10);

  pot_read = analogRead(POT_PIN);

  float pot_map = map(pot_read, 0, 1023, 127, 255);


  digitalWrite(FAN_CONT_PIN_1, HIGH);

  digitalWrite(FAN_CONT_PIN_2, LOW);


   if(fan_switch_state != prev_fan_switch_state){

       if(fan_switch_state == HIGH){

         fan_enabled = !fan_enabled;

       }

   }


   if(fan_enabled == 1){

     analogWrite(FAN_ENABLE_PIN, pot_map);

   }

   else{

       analogWrite(FAN_ENABLE_PIN, 0);

   }


  prev_fan_switch_state = fan_switch_state;

}
```

```
/*
*Function to mix rgb randomly for a normal states aesthetic.
*/
void mix_rgb(){
  digitalWrite(RED_RGB_LED, random(0,2));
  digitalWrite(GREEN_RGB_LED,random(0,2));
  digitalWrite(BLUE_RGB_LED,random(0,2));
  delay(200);
  digitalWrite(RED_RGB_LED, 0);
  digitalWrite(GREEN_RGB_LED,0);
  digitalWrite(BLUE_RGB_LED,0);
  delay(100);
}
/*
*Function to read the TEMP_PIN and read the current temperature.
*/
float sense_temp(){
  int raw_read = analogRead(TEMP_PIN);
  float scaled_temp = scale_analog(raw_read, OFFSET, GRADIENT);
  return(scaled_temp);
}
```