

LEVEL 4 ASSESSMENT SPECIFICATION

Student name:	Jaden Chapman	Student ID number:	2221720
Programme:	HNC Electrical/Electronic Engineering & Instrumentation Engineering		
Module:	Embedded Systems		
Module code:	PCEE4004	Contribution to Overall Module Assessment (%):	100%
Lecturer:	Sarah Williams	Internal Verifier:	Jason Pointer
Assignment Title:	Office Environment Monitor	Word count (or equivalent):	2000 words
Submission deadline:	12 th May 2023	Return date of provisional marks & written feedback:	10 th June 2023
Submission method:	All written assessments, where practical and possible, must be submitted via Turnitin unless otherwise instructed by the Lecturer. (Please DO NOT put this assessment specification into Turnitin or it will match many similarities with other students' submissions.) Alternative submission method (if applicable): Late submission of the assessment will result in a late penalty mark. Penalties for late submission: Up to one week late, maximum mark of 40%. Over one week late, 0%. Only the Extenuating Circumstances Panel may approve a change to submission dates.		
Academic honesty / referencing:	Academic honesty is required. In the main body of your submission you must give credit to authors on whose research and ideas your work is based. Append to your submission a reference list that indicates the books, articles, etc. that you have used, cited or quoted in order to complete this assessment.		

Module Learning Outcomes (from module syllabus)

1. Demonstrate a good understanding of the fundamental principles of programming an embedded system.
2. Describe the basic elements of typical embedded hardware and apply that knowledge in the analysis of a number of simple system applications.

STUDENT FEEDBACK FORM

This section details the *extent* to which the assessment criteria are demonstrated by you, which in turn determines your mark. The marks available for each category of skill are shown. Lecturers will use the space provided to comment on the achievement of the task(s), including those areas in which you have performed well and areas that would benefit from development/improvement.

Generic Assessment Criteria	Marks available	Marks awarded
1. Engagement with Literature Skills		
	0	
2. Knowledge and Understanding Skills		
Hardware Has exhibited a limited knowledge of the basic underlying concepts and principles. You have included descriptions of the components used in your system along with justifications as to why you chose them. You have also discussed some other components you had considered on the outset of this assignment. The schematic has missing wires to the RGB LED and Buzzer. 6.5/15 Software Competent knowledge of the basic underlying concepts and principles. Flowchart provided conforms to standards and is logically correct. Further marks here would have been gained by including pseudocode. Demonstrates a good knowledge of how the software will function. 9/15	30	15.5
3. Cognitive and Intellectual Skills		
Discussion, Evaluation and Conclusion Good evaluation of the underlying concepts and principles. Mostly valid arguments and logical judgements. You have briefly considered the usefulness of the C language but missed that it is very good when working with hardware and considerably lighter than other languages. You have given some considerations of the improvements that can be made to the system, although the addition of a battery so the device can be moved around freely and not tethered to a PC has been missed. 9/10	15	9
4. Practical Application Skills		
Implementation A very good application of a range of basic methods taught, with an appropriate solution provided. Program is functional and meets the requirements of the assignment. Your code is well commented, and you have followed conventions when using and naming your variables and constants. Your indentation isn't correct, where there is an IF you should indent and then return back in line with the IF when you close the bracket associated with it. Whilst this does not affect the functionality of the program it does aid in the readability. I liked the feature that you implemented where the buzzer just sounded	50	33

briefly when there was a change of state, much more suitable in an office environment than it sounding continuously. Further marks could have been achieved if you had used functions. 27/40 Testing A very good application of basic techniques for testing. Testing covers all possible states of the system, your images suggest you are attempting to work at the boundaries of your state changes, which is good practice. However, you should have explicitly stated what the values you were testing for (i.e. the boundaries i.e. 24.1, 24.0, rather than 23.7 etc) in the table. 6/10		
5. Transferable Skills for Life and Professional Practice		
Work is accurate, coherent, fluent, well-structured and organised. Can work effectively independently. You should include what research you did to determine the temperature ranges you have selected. I am not sure if the images have moved but both the breadboard and schematic should be in the hardware section. Otherwise, work is formally set out, including contents pages and labelled diagrams. 3/5	5	3

Assessment Mark (Assessment marks are subject to ratification at the Exam Board. These comments and marks are to give feedback on module work and are for guidance only until they are confirmed.)	Late Submission Penalties (tick if appropriate)		60.5%
	Up to 1 week late (40% Max)		
	Over 1 week late (0%)		

Jaden Chapman

Designing a Temperature Sensor

Embedded Systems

HNC Electrical / Electronic Engineering

Contents

- Introduction
- Hardware Design
- Software Design
- Implementation
- Testing
- Evaluation
- Appendix
- References

Introduction

Concerns have been raised regarding the heating and air conditioning system in the workplace. I have been tasked with designing a temperature sensor that informs people in the workplace when the environment's conditions are considered outside of the 'normal' range. There is a requirement for the microcontroller-based system to indicate when the conditions are not within range via LED, buzzer, LCD etc.

I will be configuring an LED and buzzer to coincide with the temperature sensor to display/sound when the component senses changes in temperature. I have decided that the 'normal' working range be between 16 – 24°C at which point a green LED will be displayed. Anything temperature lower than 16°C will result in the LED turning to blue and the buzzer briefly sounding. Anything higher than 24°C will result in the LED turning to red and the buzzer briefly sounding.

The code will be written using Arduino software as this will enable me to easily upload my code to the microcontroller and make changes as necessary. I will also be using Fritzing to design my schematics online so that I can visualise how the circuit will operate before wiring it to breadboard.

As well as software, I will be using a variety of hardware components including resistors, buzzers, LEDs and the sensor itself.

Task 1 – Hardware Design

Resistors – The resistor is used to regulate the amount of current flowing through a circuit. In my project, I used three 220ohm resistors and one 10000ohm resistor. The three 220ohm resistors were each connected to an individual pin of the RGB LED and the 10000ohm resistor was connected to the temperature sensor.

Temperature & Humidity Sensor – The DHT11 is a sensor capable of measuring temperature and humidity making it perfect for this application. It uses a capacitive humidity sensor and thermistor to measure the conditions of the environment and outputs a digital signal on the data pin. I considered using an LM35 sensor but, after comparing the data sheets of the two, decided the DHT11 would be a better fit for the application. Even though the LM35 is slightly more accurate, it's not capable of measuring both temperature and humidity.

LED – The RGB LED is a four-pin device that offers the user the opportunity of displaying three different colours: red, green and blue. I made full use of this function by configuring a different colour to each of the different temperature ranges. Instead of an LED, I could have used an LCD screen to display a temperature reading using text but eventually decided to go with a more simplistic approach.

Buzzer – The buzzer outputs a tone when a voltage is applied to it. When this voltage is applied will depend on how it is configured within the circuit. In my project, I've configured the buzzer to sound for one second whenever it enters a new temperature range. This was a component that wasn't necessarily required for the operation of the circuit, but I decided to implement it for the users who may not have the temperature in their view.

Microcontroller – The Arduino Uno is a microcontroller-based system that is responsible for receiving the code from the programming software and outputting this information to the correct pin. It's considered to be one of the more beginner-friendly microcontroller boards which proved to be a deciding factor when choosing which board to use for my project.

Breadboard – Breadboard is a semi-permanent construction base that can be used to design electronic circuits. It's easy to take circuits apart and re-configure them due to the board housing tiny holes where legs of components can be inserted and ejected as necessary. I did consider building the circuit on a stripboard but decided against this as it requires soldering and destruction of tracks in order to achieve a fully functioning circuit. If mistakes are made within the circuit on breadboard, it's a smaller issue.

Connecting cables – I've used colour co-ordinated cables to connect components to the required pins on the microcontroller. This allows the user to recognise which cable is connected to which component so that they can be changed with ease.

DATA SHEETS FOR EACH COMPONENT CAN BE FOUND IN THE APPENDIX.

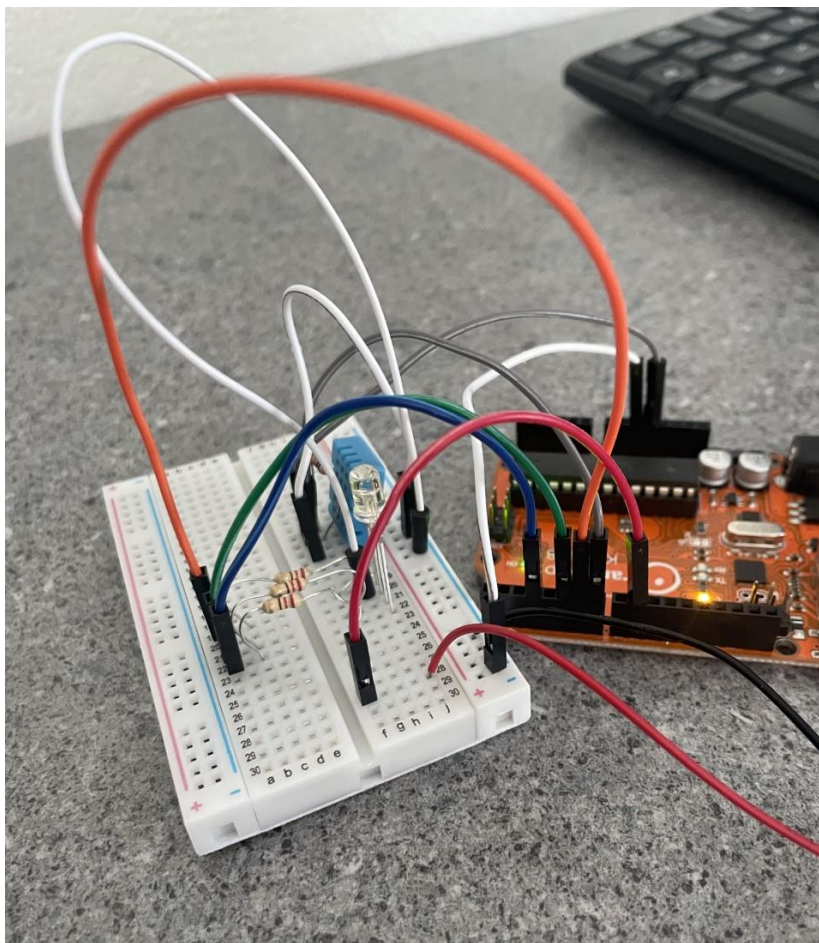
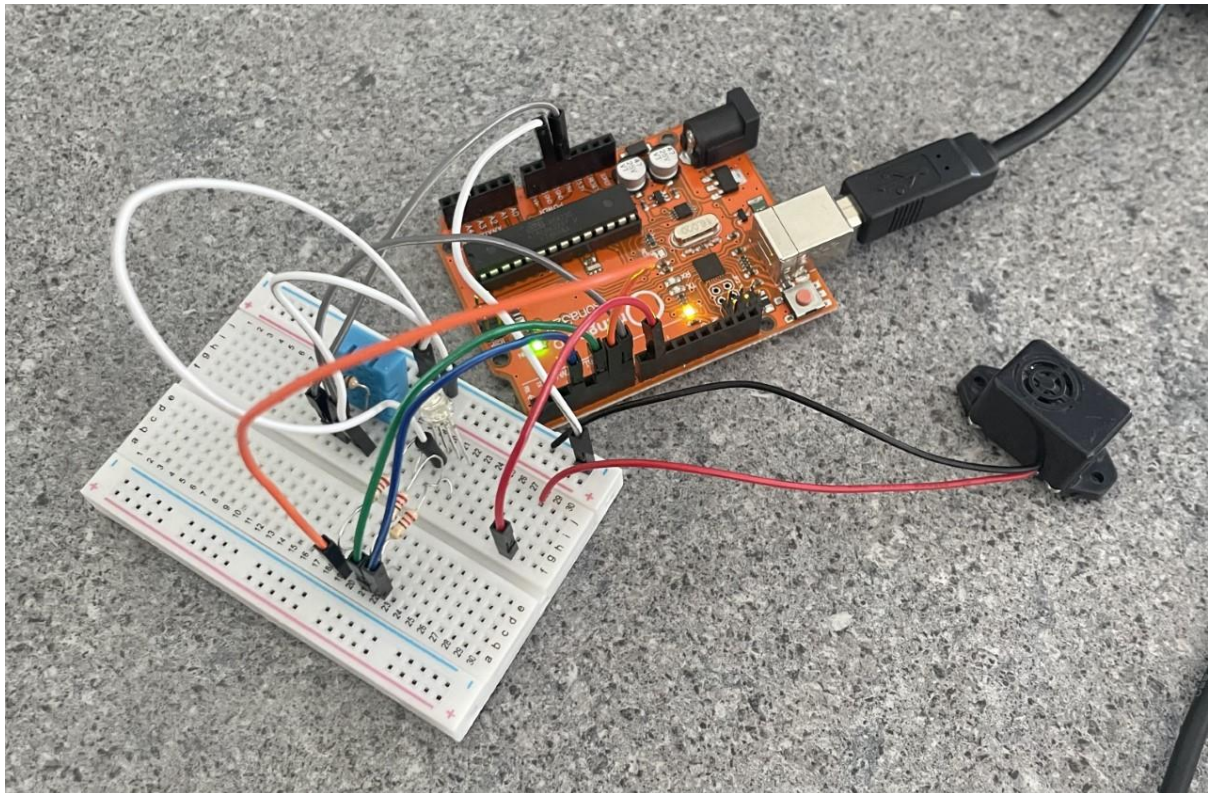


Figure 1 & 2 - Temperature Sensor Physical Circuit

Task 2 - Software Design

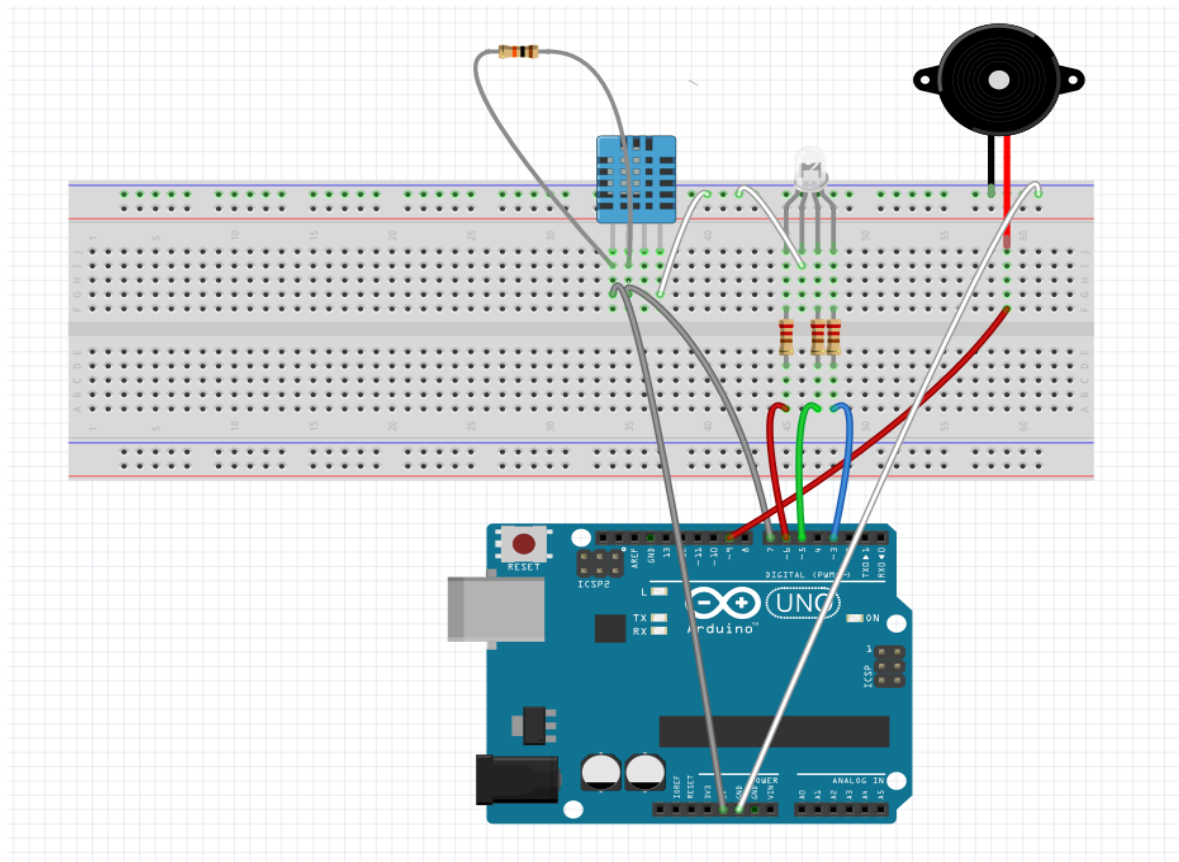


Figure 3 - Temperature Sensor Circuit Design

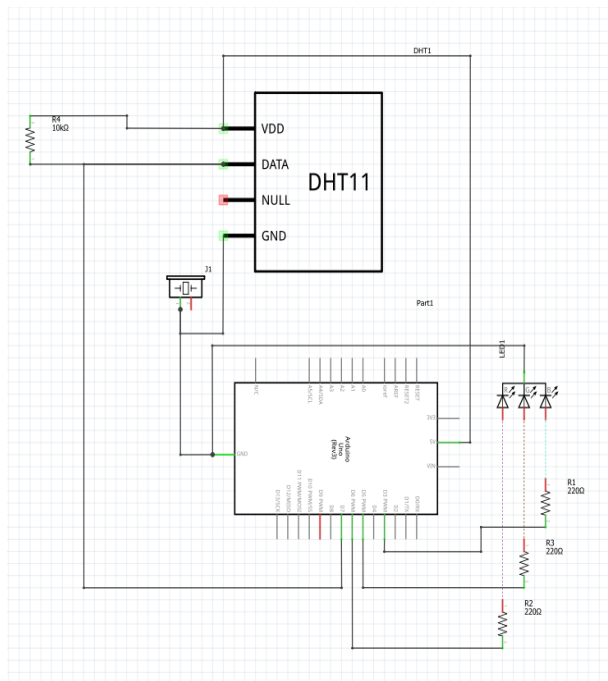


Figure 4 - Temperature Sensor Wiring Diagram

Seen above is the breadboard design for my temperature and humidity sensor. To the left is a wiring diagram of my sensor. I designed these prior to building the physical circuit using Fritzing, a software that allows users to design breadboard, schematic and PCB layouts so that they can be followed when it comes to physically building the circuit.

Fritzing enabled me to configure the circuit using different components so that I could test what worked best. The user interface made it easy for me to shuffle components around to find the best configuration for my project.

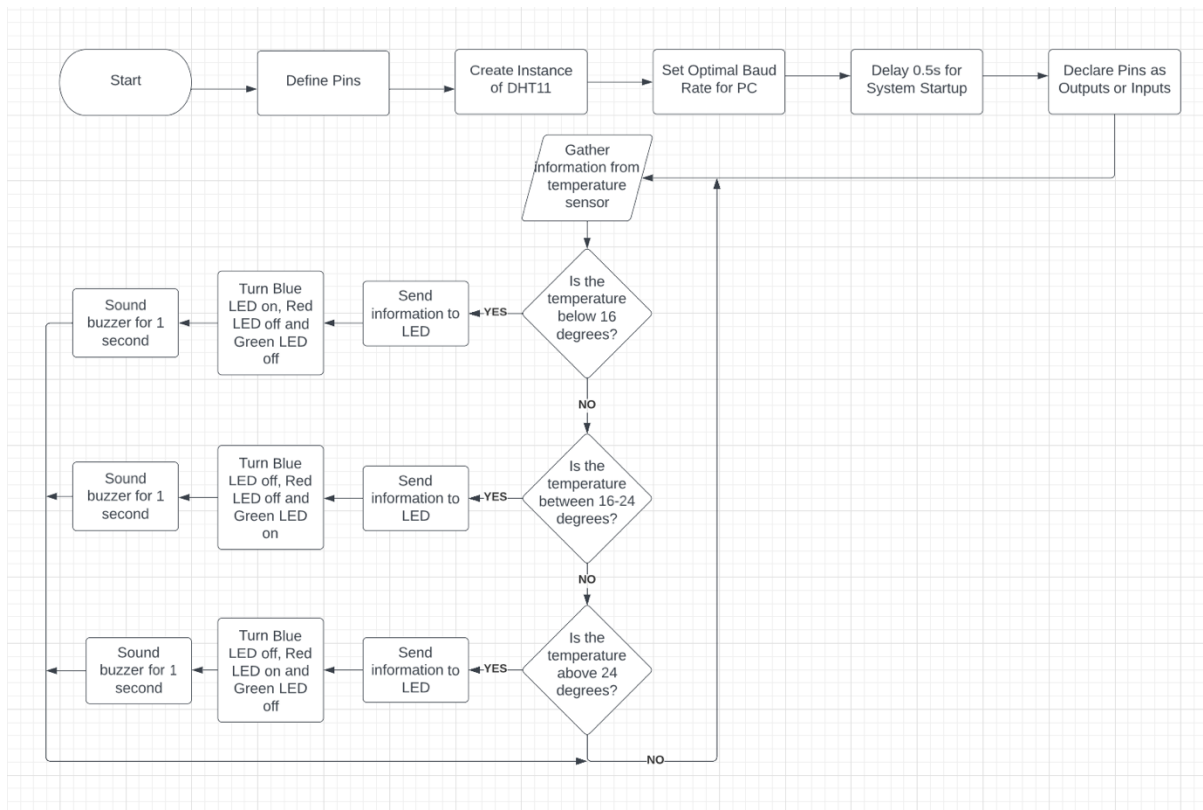


Figure 5 – Flowchart

Another piece of software I used was Lucid Chart, a web-based tool that allows users to build flow charts using different shapes, colours and connectors in order to build an accurate representation of how the programme is going to function once it has been written and implemented.

Designing the flowchart also allowed me to find solutions to problems in simple terms before they were transferred to code so that when it came to writing the code, there was less troubleshooting to do.

Task 3 – Implementation

```
#include "DHT.h" //bringing library in
#define DHT11_PIN 11 //configures DHT11 to pin 11
#define BUZ 9 //configures buzzer to pin 9
#define RED_LED 6 //configures red LED to pin 3
#define GREEN_LED 5 //configures green LED to pin 5
#define BLUE_LED 3 //configures blue LED to pin 6
DHT tempSensor (DHT11_PIN, DHT11);
char currentreading='x'; //sets current reading to x

void setup() {
    // put your setup code here, to run once:
    Serial.begin(9600);
    delay(500); //allows the system to start up
    Serial.println("DHT11 Humidity & Temperature Sensor\n\n");
    delay(500);
    tempSensor.begin();
    pinMode(BUZ, OUTPUT); //identifying buzzer as an output
    pinMode(RED_LED, OUTPUT); //identifying red LED as an output
    pinMode(GREEN_LED, OUTPUT); //identifying green LED as an output
    pinMode(BLUE_LED, OUTPUT); //identifying blue LED as an output

}
```

Figure 6 - Setup Code

```

void loop() {
    // put your main code here, to run repeatedly:
    float humidity = tempSensor.readHumidity();
    float temp = tempSensor.readTemperature();
    Serial.println(humidity); //testing if reading correct humidity
    Serial.println(temp); //testing if reading correct temperature

    if(temp>=16&&temp<=24) {
        analogWrite(GREEN_LED, 255); //display green LED when the temperature is between 16 and 24 degrees
        analogWrite(BLUE_LED, 0); //don't display blue LED when the temperature is between 16 and 24 degrees
        analogWrite(RED_LED, 0); //don't display red LED when the temperature is between 16 and 24 degrees
        if(currentreading!='n'){ //checks to see if temperature has changed state
            digitalWrite(BUZ, HIGH); //buzzer remains off when the temperature is within this range
            currentreading='n'; //records current temperature is normal
            delay(1000); //delay allows buzzer to sound for 1000ms
            digitalWrite(BUZ, LOW); //buzzer stops sounding after 1000ms

        }
    }
    else if(temp<16) {
        analogWrite(BLUE_LED, 255); //display blue LED when temperature is lower than 16 degrees
        analogWrite(GREEN_LED, 0); //don't display green LED when the temperature is above 16 degrees
        analogWrite(RED_LED, 0); //don't display red LED when the temperature is below 16 degrees
        if(currentreading!='c'){ //checks to see if temperature has changed state
            digitalWrite(BUZ, HIGH); //buzzer sounds when temperature decreases below 16 degrees
            currentreading='c'; //records current temperature is cold
            delay(1000); //delay allows buzzer to sound for 1000ms
            digitalWrite(BUZ, LOW);} //buzzer stops sounding after 1000ms
    }
    else if(temp>24) {
        analogWrite(RED_LED, 255); //display red LED when temperature is higher than 24 degrees
        analogWrite(GREEN_LED, 0); //don't display green LED when the temperature is above 24 degrees
        analogWrite(BLUE_LED, 0); //don't display blue LED when the temperature is above 24 degrees
        if(currentreading!='h'){ //checks to see if temperature has changed state
            digitalWrite(BUZ, HIGH); //buzzer sounds when temperature increases above 24 degrees
            currentreading='h'; //records current temperature is hot
            delay(1000); //delay allows buzzer to sound for 1000ms
            digitalWrite(BUZ, LOW);} //buzzer stops sounding after 1000ms
    }

    delay(5000);

}

```

Figure 7 - Loop Code

Task 4 - Testing

Test	Expected Result	Actual Result	Issues found	Issues fixed
Temperature reading 12 degrees	LED displaying blue	LED displayed blue	No issues found	
Temperature reading 23.7 degrees	LED displaying green	LED displayed green	No issues found	
Temperature reading 24.1 degrees	LED displaying red	LED displayed red	No issues found	
Temperature falls below 16 degrees	Buzzer sounds for 1 second and LED changes from green to blue	Buzzer sounded, LED changed	For as long as temperature was out of normal range, the buzzer would sound for 1 second every 5 seconds	Ammended code so that the buzzer only sounded again if it had since changed state
Temperature rises above 24 degrees	Buzzer sounds for 1 second and LED changes from green to red	Buzzer sounded, LED changed	For as long as temperature was out of normal range, the buzzer would sound for 1 second every 5 seconds	Ammended code so that the buzzer only sounded again if it had since changed state

Figure 8 - Testing Table

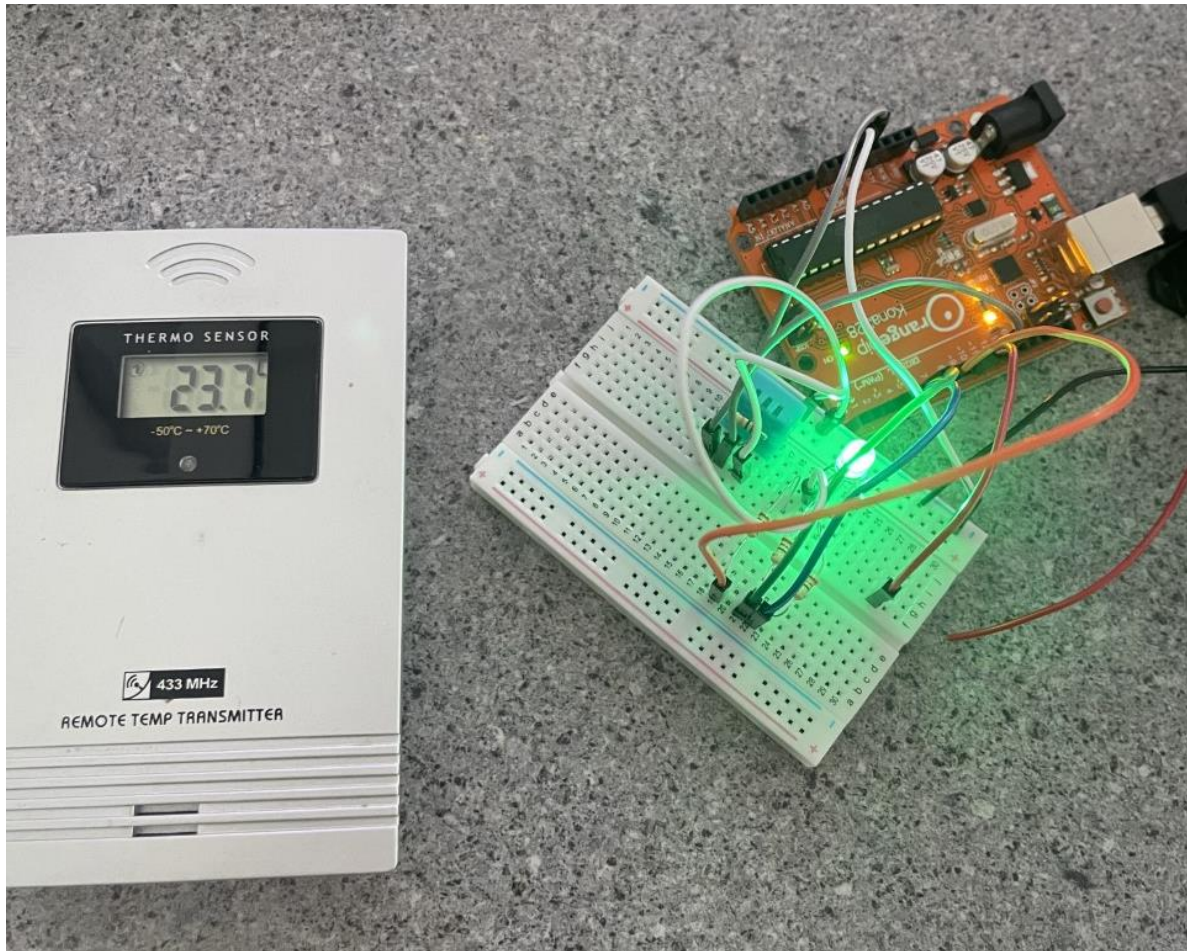


Figure 9 - Green LED Test

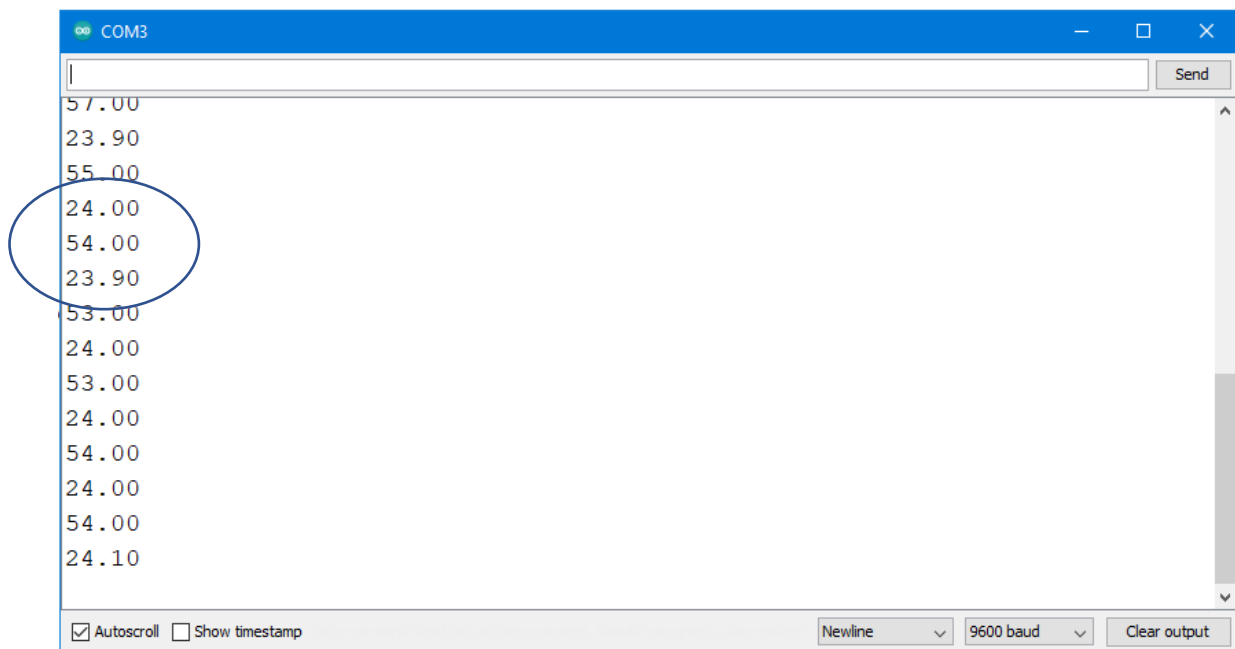


Figure 10 - Serial Print

For this test, I took readings from an external thermometer and the Arduino print monitor to ensure the sensor was measuring the temperature accurately. As seen in figure, the thermometer is reading 23.7°C and on serial print, the temperature has changed from 24°C to 23.9°C. This shows that the LED is capable of displaying green when the temperature is within the normal range.

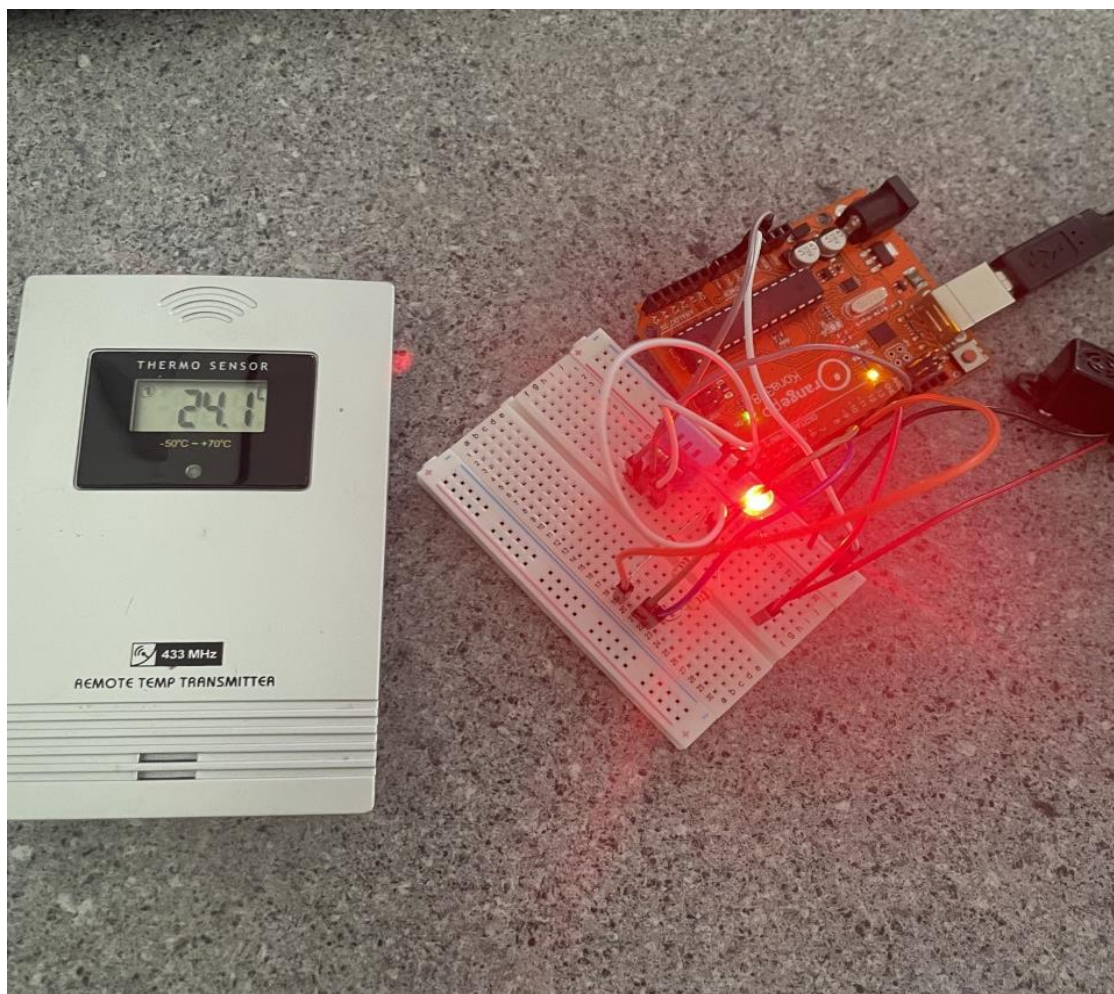


Figure 11 - Red LED Test

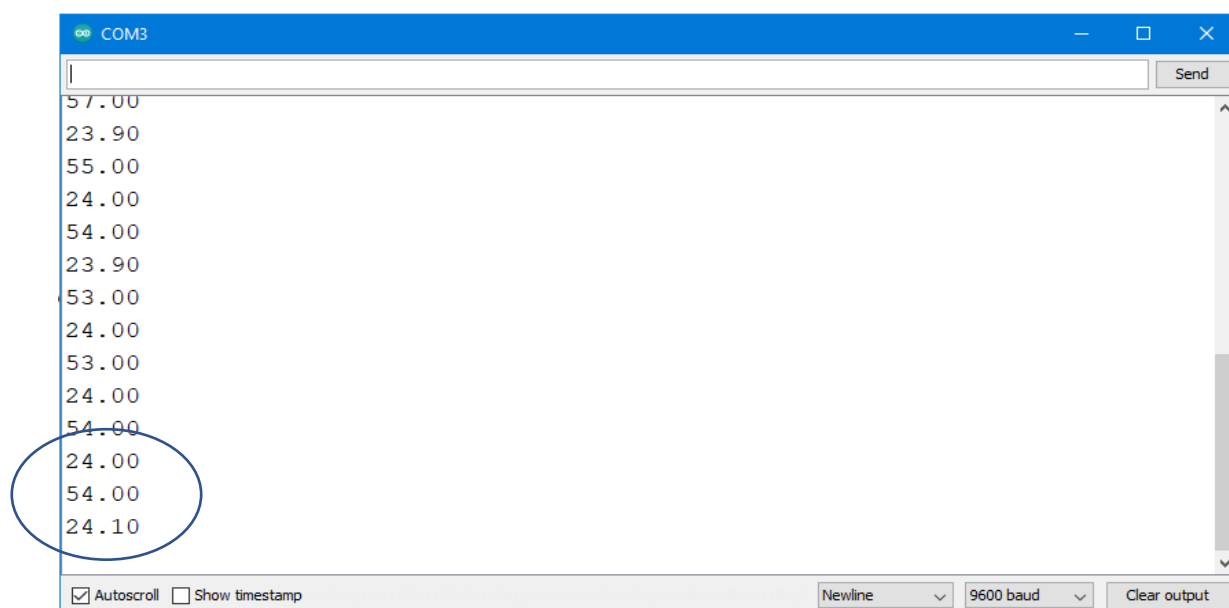


Figure 12 - Serial Print

I took readings from an external thermometer and the Arduino print monitor for this test also. As seen in figure, the thermometer is reading 24.1°C and on serial print, the temperature has risen from 24°C to 24.1°C. This shows that the LED is capable of displaying red when the temperature enters the high temperature range.

Issues

When I first uploaded the code to the microcontroller and ran it, the LED was operating as it should've been, but the buzzer would sound once every 5 seconds for 1 second no matter the temperature reading. After reviewing my code, I recognised I'd set a 1 second delay between the buzzer's on and off state and a 5 second delay before the loop was to run the code again so it made sense why the buzzer was operating in this way. Although, I didn't want it to be configured this way. I wanted it to be configured so that the buzzer would only sound when the temperature changed range.

This issue was amended by implementing a character statement and an 'if' statement to each section of the loop code.

In the setup code, whatever temperature range the temperature was currently reading was to be set as 'x'.

In the loop code, I implemented an 'if' statement that checks to see if the temperature reading is still 'x' or whether it has entered a different temperature range (not 'x'). The buzzer will only sound if the temperature isn't reading 'x'.

I named the current reading for the 16°C > range as 'c' (for cold), the 16°C < 24°C range as 'n' (for normal) and the 24°C < range as 'h' (for high).

An example of the implemented code for the high temperature range can be seen below.

```
char currentreading='x'; //sets current reading to x
```

Figure 13 - Character Statement

```
if(currentreading!='h'){ //checks to see if temperature has changed state
digitalWrite(BUZ, HIGH); //buzzer sounds when temperature increases above 24 degrees
currentreading='h'; //records current temperature is hot
delay(1000); //delay allows buzzer to sound for 1000ms
digitalWrite(BUZ, LOW);} //buzzer stops sounding after 1000ms
```

Figure 14 - If Statement

Task 5 – Evaluation

Result

I believe that I have completed the project to the highest standard I could have within the given timeframe. I have written a fully functioning program using C language that meets the criteria of the assignment brief. I've also uploaded this program to the Arduino Uno and proven throughout the assignment that the circuit functions as it should.

I've also shown that I can build schematics and wiring diagrams that can be later referred to when it comes to building the physical circuit.

C Language

Using C Language within the Arduino software was simple. The software contains a library of pre-written code and also offers verification of code before uploading to the Arduino Uno making it beginner friendly. The language also avoids keywords and symbols as much as possible making it easy to grasp concepts and relate code from one program to another. Another advantage is the ease of debugging as the language doesn't require the use of complex statements.

Improvements

By the end of the project, the sensor operated exactly how I wanted it to although if I was to design a temperature sensor in the future, there are some aspects I would make improvements to.

One way I believe the system could be improved is with the implementation of an LCD screen. This is something the temperature sensor I have designed doesn't offer as it is only capable of sounding a buzzer and changing the colour of an RGB LED when certain conditions are met. The implementation of an LCD screen would allow me the opportunity of configuring the sensor to display the current humidity and temperature readings through text. This would be a lot more convenient for the user as they would be made aware of the exact temperature rather than just the range that it's currently in.

Another improvement that could be made to the sensor regards the buzzer. In my project, the buzzer's tone remains the same whether the temperature enters the lower, normal or higher range. If the sensor isn't within the eyesight of the user, there is no way of distinguishing what range the temperature has just entered. If I was to design this sensor again, I would configure the buzzer to sound a different tone for each temperature range. To go even further, the sensor could be designed to give the user the capability of choosing what tone sounds at each temperature range.

The sensor could also be improved by employing a dead band. A dead band is a signal region that does not respond to small changes sensed by the instrument. This would be especially useful if the current temperature is sitting at the higher end of one range and the lower end of another. For example, if the temperature was to be reading 24 degrees, a fluctuation of 0.1°C will result in the buzzer sounding. The dead band could be configured so that the buzzer was to only sound if there was a fluctuation of 0.3°C essentially ignoring smaller changes in temperature.

Conclusion

Although I recognise that there are some areas that could be improved if I was to revisit to this in the future, I am happy with the outcome of the project. I've built a fully functioning circuit that adheres to the assignment brief whilst also adopting multiple ways of alerting the user of a temperature change.

Appendix

Specifications	
Resistance (Ohms)	220
Power (Watts)	0.25W, 1/4W
Tolerance	±5%
Packaging	Bulk
Composition	Carbon Film
Temperature Coefficient	350ppm/°C
Lead Free Status	Lead Free
RoHS Status	RoHS Compliant

Figure 15 - 220 Ohms Resistor Data Sheet

(220R Datasheet, 2023)

			
Sensor	DHT11	DHT22 (AM2302)	LM35
Measures	Temperature Humidity	Temperature Humidity	Temperature
Communication protocol	One-wire	One-wire	Analog
Supply voltage	3 to 5.5V DC	3 to 6V DC	4 to 30 V DC
Temperature range	0 to 50°C	-40 to 80°C	-55 to 150°C
Accuracy	+/- 2°C (at 0 to 50°C)	+/- 0.5°C (at -40 to 80°C)	+/-0.5°C (at 25°C)

Figure 16 - DHT 11 Data Sheet

(DHT 11 vs, 2023)

Pin Configuration

Pin No.	Pin Name	Description
1	R	This terminal used for glowing LED in Red color
2	Gnd	Common Cathode terminal (Ground)
3	G	This terminal used for glowing LED in Green color
4	B	This terminal used for glowing LED in Blue color

Figure 17 - RGB LED Pin Configuration

Features and Specifications

- Low Thermal Resistance
- No UV rays
- Super High flux Output and High luminance
- Forward Current for Red, Blue and Green color: 20mA
- Forward Voltage
 - Red: 2v (typical)
 - Blue: 3.2(typical)
 - Green: 3.2(typical)
- Luminous Intensity
 - Red: 800 mcd
 - Blue: 4000 mcd
 - Green: 900 mcd
- Wavelength
 - Red: 625 nm
 - Blue: 520 nm
 - Green: 467.5 nm
- Operating Temperature: -25 °C to 85 °C
- Storage Temperature: -30 °C to 85 °C

Figure 18 - RGB LED Data Sheet

(RGB LED Pinout Configuration, 2023)

Buzzer Pin Configuration

Pin Number	Pin Name	Description
1	Positive	Identified by (+) symbol or longer terminal lead. Can be powered by 6V DC
2	Negative	Identified by short terminal lead. Typically connected to the ground of the circuit

Buzzer Features and Specifications

- Rated Voltage: 6V DC
- Operating Voltage: 4-8V DC
- Rated current: <30mA
- Sound Type: Continuous Beep
- Resonant Frequency: ~2300 Hz
- Small and neat sealed package
- Breadboard and Perf board friendly

Figure 19 - Buzzer Data Sheet

(Buzzer Pinout Working Datasheet, 2023)

Board	Name	Arduino UNO R3
	SKU	A000066
Microcontroller	ATmega328P	
USB connector	USB-B	
Pins	Built-in LED Pin	13
	Digital I/O Pins	14
	Analog input pins	6
	PWM pins	6
Communication	UART	Yes
	I2C	Yes
	SPI	Yes
Power	I/O Voltage	5V
	Input voltage (nominal)	7-12V
	DC Current per I/O Pin	20 mA
	Power Supply Connector	Barrel Plug
Clock speed	Main Processor	ATmega328P 16 MHz
	USB-Serial Processor	ATmega16U2 16 MHz
Memory	ATmega328P	2KB SRAM, 32KB FLASH, 1KB EEPROM
Dimensions	Weight	25 g
	Width	53.4 mm
	Length	68.6 mm

Figure 20 - Arduino Uno Data Sheet

(Arduino Uno REV3, 2023)

References

220R Datasheet. (2023). Retrieved from Dig Chip:

<https://www.digchip.com/datasheets/parts/datasheet/1838/CFR-25JB-220R.php>

Arduino Uno REV3. (2023). Retrieved from Arduino: <https://docs.arduino.cc/hardware/uno-rev3>

Buzzer Pinout Working Datasheet. (2023). Retrieved from Components 101:

<https://components101.com/misc/buzzer-pinout-working-datasheet>

DHT 11 vs. (2023). Retrieved from Random Nerd Tutorials: <https://randomnerdtutorials.com/dht11-vs-dht22-vs-lm35-vs-ds18b20-vs-bme280-vs-bmp180/>

RGB LED Pinout Configuration. (2023). Retrieved from Components 101:

<https://components101.com/diodes/rgb-led-pinout-configuration-circuit-datasheet>