

LEVEL 4 ASSESSMENT SPECIFICATION

Student name:	Heather Raymond	Student ID number:	2221721
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 competent knowledge of the basic underlying concepts and principles. You have included a description of the components used and justified why you have used them, whilst I agree using 3 separate LEDs is more complex from a wiring perspective, there is not additional demands on the memory than if using the RGB LED. Other components were considered and you have explained why these were not used. A seven-segment display is very limiting in terms of what it can output, whereas an LCD has several characters over several lines, depending on the LCD used. Another reason it would be a better option over the seven-segment display. Schematic correct- but in future try to use horizontal and vertical lines to join up components. 10/15 Software Demonstrates a reasonable knowledge of the basic underlying concepts and principles. Flowchart does not conform to standards. Use a rectangle for process, and where branching takes place a diamond box is used, with a question in the box. Any path coming from a decision box should be clearly labelled to make it clear which path is taken as a response to the question in the diamond box. The flowchart also contains dead ends. Pseudocode confuses loop constructs with selection, I think in the majority of this (having looked at your implementation) you meant to use IF and END IF. You have a great breakdown of the software functionality in the project notes in your appendices. 6/15	30	16
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 some of the useful features of the C language when working with embedded systems and reviewed the effectiveness of your designed system. Sensible improvements to the system are also included. 10/15	15	10
4. Practical Application Skills		
Implementation An advanced application of a range of basic methods taught, with an appropriate solution provided. Program is functional and meets the requirements of the assignment. You have annotated the code well, and made good use of constants, naming of pins is good	50	35

practice. Good use of the loop in the setup to initialise adjacent pins. Also, good practice to check the LEDs are functioning prior to getting into the main loop. Indentation is not correct - whilst this has no impact on the functionality of the program it aids readability. In the main loop you have also considered what happens if the DHT11 malfunctions and does not pick up a reading. Further marks would be gained had you used functions. 29/40 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, 18.1, 18, etc) in the table. 6/10		
5. Transferable Skills for Life and Professional Practice		
Work is coherent, very fluent and is presented professionally. Can work effectively independently. Work is formally set out, well structured, including contents pages, references and labelled diagrams. 4/5	5	4

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)		65%
	Up to 1 week late (40% Max)		
	Over 1 week late (0%)		



EMBEDDED SYSTEMS

module PCEE4004

Abstract

A report on the use of a microcontroller, software and hardware elements of an embedded system designed to control the temperature in an office environment.

Heather Raymond
95699@pembrokeshire.ac.uk

LEVEL 4 ASSESSMENT SPECIFICATION

Student name:	Heather Raymond	Student ID number:	95699
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:	(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%.		
Academic honesty / referencing:	Academic honesty is required. Append to your submission a reference list.		

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.

Introduction

The purpose of this report is to document the design and implementation of a micro-controller-based system. The device will monitor the room temperature in a workplace environment and will inform the user if the temperature is outside of predefined working conditions. Hardware design, Software design, implementation of the system and testing all come under the scope of this document. At the end of this report an evaluation of the project will be completed, considering all testing and design considerations throughout the project.

General design requirements/ considerations

Some initial notes on component selection, criteria and considerations are shown in *Appendix A*. According to HSE safety regulations, temperatures should not drop below 16°C, however there is no legal limit on how hot a workplace can be (Source: HSE, 2023).

But based on research into the topic, it would be wise to set an indication at 24°C or above, as the ideal temperature in an office space has been determined to be 22°C.

Ideally, the temperature range should be between 18°C and 24°C as a comfortable working range for most people. (Source: Nance-Nash, S. 2020). The system could use a lower temperature indication and a higher one, at the temperatures aforementioned. Additionally, there could be a message for when the temperature is below the legal limit.

Task 1 – Hardware Design

Micro controller selection

An Orangepip Kona328 board has been selected for this project, as this is what is available, is cheap to purchase and is compatible with Arduino software, as well as, meeting hardware requirements. This board is based around the AT- Mega328 microcontroller. It has 14x digital I/O pins (6 which may be used as PWM outputs), 6 analogue inputs and a 16MHz crystal oscillator. It has a USB port to connect to a laptop or computer, ground and power terminals suitable for powering LEDs, displays and temperature sensors via connection to a breadboard. It can be powered by battery as well, if portability is desirable for this application in the future. The data sheet for this component can be found in *Appendix B*.

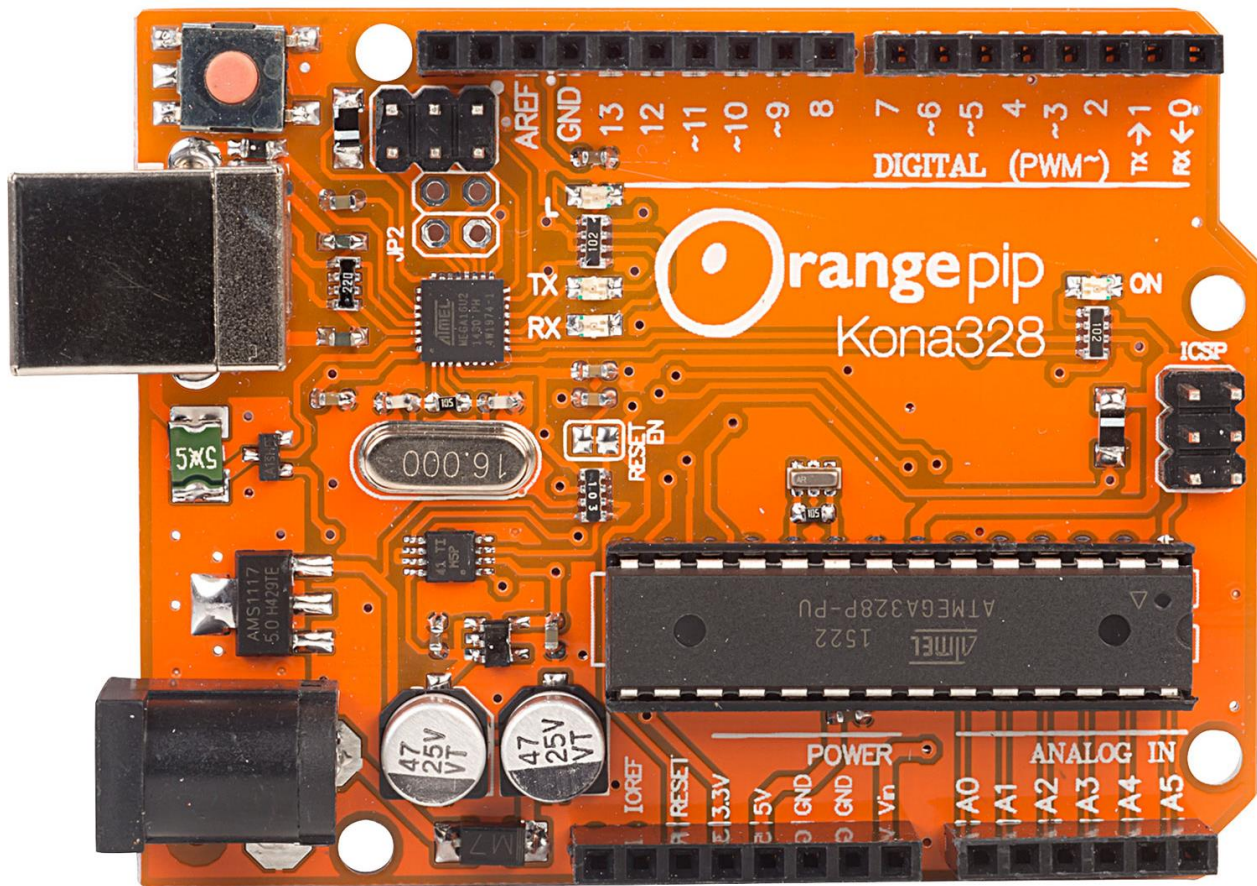


Figure 1, Source: <https://www.rapidonline.com/orangepip-kona328-arduino-uno-compatible-development-board-75-0550>

DHT11 Temperature and humidity sensor

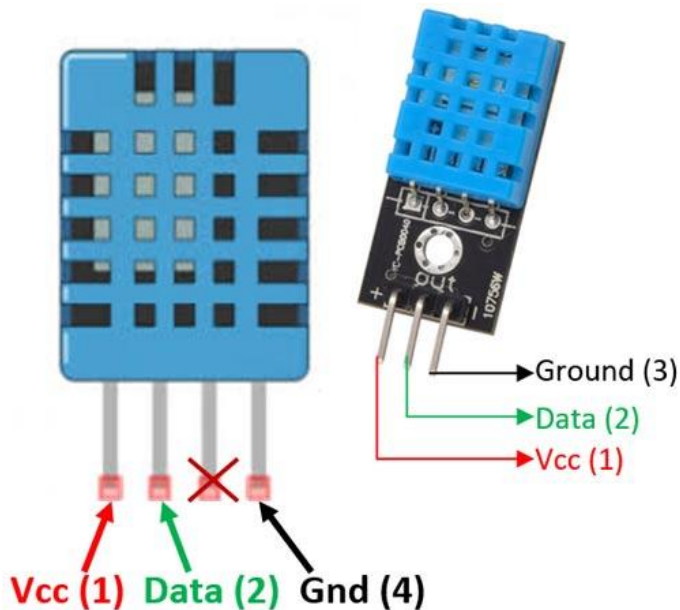


Figure 2, source: <https://components101.com/sensors/dht11-temperature-sensor>

For this project, the DHT 11 is a suitable temperature sensor. It is compatible with the Orangepip board, requires a 3.3 ~ 5.5V DC supply and has a working range of 0 to 50°C. It has an accuracy of $\pm 2^{\circ}\text{C}$, which is adequate for this application. The data sheet for this sensor can be found in *Appendix C*. The low cost, high availability and interchangeableness of this sensor makes it suitable for this application. In addition, the DHT11 uses just one signal wire to transmit data to the Arduino board.

Six other widely used temperature sensors include the DHT22, LM35, DS18B20, BME280 and BMP180. 0 to 50°C measurement is the ideal range for ambient temperature monitoring, the other sensors in this list range from -55 to 85°C, which are more than what is required of this application. The BME280 and BMP180 require more wiring than the DHT 11. These sensors also require a separate power supply for their chips, making the hardware setup more complicated. The LM35 and DS18B20 do not have dedicated libraries in the Arduino software. The DHT11 is cheaper than the DHT22 and its response time is faster. (Source: English, C. et al. 2019).

RGB LED and an LCD display

For this project, an RGB LED will be a suitable indicator of the room temperature. A red light could be used when the temperature reading goes above 24°C, a green indication could show the temperature is at an acceptable value and a blue indication could indicate the temperature has dropped below 16°C, the legal limit. One could use several LEDs of different colours, but this would require more space on the bread board and possibly more memory in the code. A buzzer is an option as well, but this could be an irritation in an office environment and would only provide an audible warning.

It would also be useful in this program to include an LCD display with the temperature value on it. This would enable the user to check the validity of the RGB LED indication and to give an idea of how much to change the thermostat/ air conditioning by in terms of degrees centigrade. The LCD display is a better option than a 7-segment display, as it requires less pins on the Arduino board and uses less code within the software program.

Power supply, wiring and protective considerations

Each LED colour leg will require protection via a 220Ω resistor. The DHT11 will require a 10KΩ resistor. The power supply will be via a laptop connected to the Orangepip board via its USB port. A solderless breadboard and wires will be used for the circuit, for ease of use and adaptability during this proto-type stage. It would be wise to colour code the wires to the RGB LED and other elements of the circuit, for ease of identification and modification. A rough drawing of the circuit can be seen in *Appendix D*.

Hardware system design drawing (fritzing)

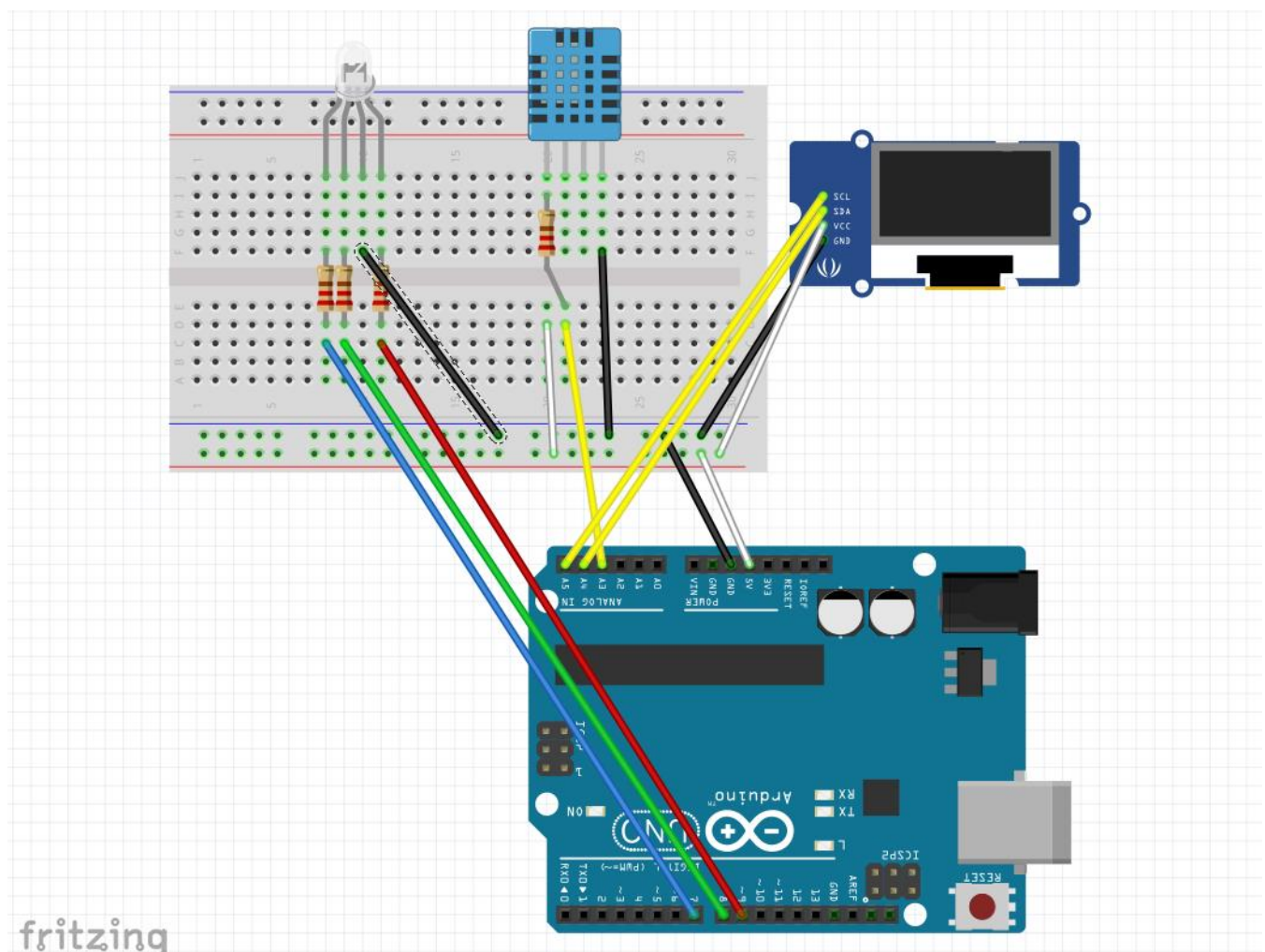
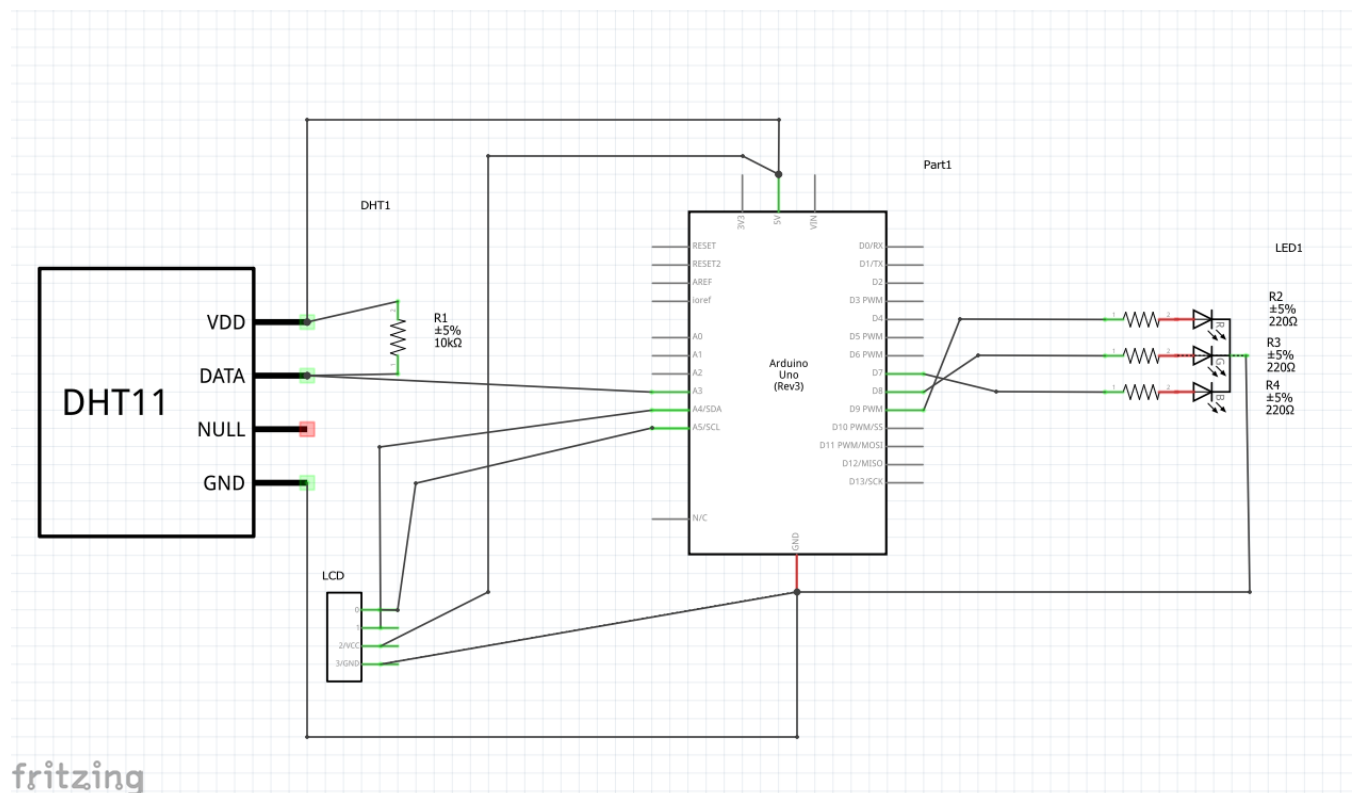


Figure 3

wiring diagram



fritzing

Figure 4

Task 2 – Software Design

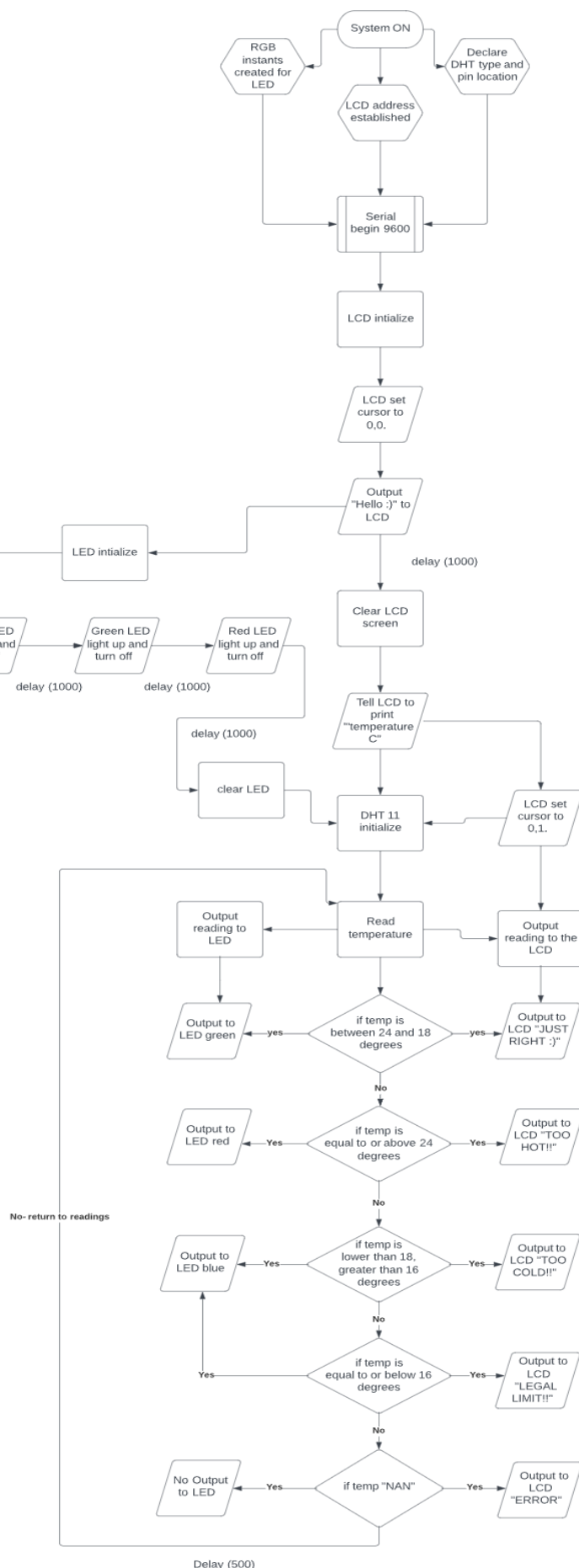


Figure 4, flow chart made in the Lucid Software

Arduino IDE program and Lucid program

The Arduino IDE software has been selected for this project due to it being free and readily available to download. It's compatibility with common windows and apple mac systems. This software is easy and intuitive to use for beginners to programming, the language is based on C++. There are also libraries available with pre-written code, which should help with some of the functions of the temperature indicating circuit. (Source: Kumar, A. 2023).

Lucid software is available online, is also open source, and will be used in the scope of this project to design a flow chart. This chart should help plan out the program from the start of its functions. Pseudo code will also be used to help plan the project, and will be compiled in a word document, using six common programming constructs. (Metwalli, S.A. 2022).

Figure 4 shows the flow chart design for this system, to help inform the code to be written in the arduino software, using common flow chart conventions.

INITIAL SETUP

INCLUDE DHT and LCD libraries.
DEFINE DHT pin location.
DEFINE DHT type- DHT11.
DEFINE LCD (liquid crystal) bus address.
CONST INT three colours of LED constants.
END.

SETUP SEQUENCE TO RUN ONCE

SET Baud rate to 9600.
INIT LCD backlight, interface.
SET LCD cursor position to 0,0.
PRINT LCD "Hello :)".
REPEAT-UNTIL RGB sequence has ended.
INCREMENT delay 1000 milli seconds.
ENDWHILE, run RGB test sequence.

SECURE pin addresses LEDs.

SHOW blue LED.
INCREMENT delay 1000 milli seconds.
ENDFOR switch off blue LED.

SHOW green LED.
INCREMENT delay 1000 milli seconds.
ENDFOR switch off green LED.

SHOW red LED.
INCREMENT delay 1000 milli seconds.

ENDFOR switch red LED.

FUNCTION LOOP

INCREMENT delay 1000 milli seconds.

INIT LCD clear screen.

SET LCD cursor position to 0,0.

PRINT LCD "Temp:"

PRINT LCD "t"

PRINT LCD "C"

IF temperature is above 24°C.

SHOW red LED on.

SHOW blue LED off.

SHOW green LED off.

SET LCD cursor position to 0,1.

PRINT LCD "TOO HOT!!"

REPEAT-UNTIL temperature is equal to or drops below 24°C.

ELSE-IF temperature is equal to or below 24°C, above 18°C.

SHOW green LED on.

SHOW blue LED off.

SHOW red LED off.

SET LCD cursor position to 0,1.

PRINT LCD "JUST RIGHT :)"

REPEAT-UNTIL temperature goes above 24°C or drops below 18°C.

ELSE-IF temperature is equal to or below 18°C, above 16°C.

SHOW blue LED on.

SHOW green LED off.

SHOW red LED off.

```
SET LCD cursor position to 0,1.  
PRINT LCD "TOO COLD!!"  
REPEAT-UNTIL temperature goes above 18°C or drops below 16°C.  
  
ELSE-IF temperature is equal to or below 16°C.  
SHOW blue LED on.  
SHOW green LED off.  
SHOW red LED off.  
SET LCD cursor position to 0,1.  
PRINT LCD "LEGAL LIMIT!!"  
REPEAT-UNTIL temperature goes above 16°C.  
  
CASE if temperature reading is lost.  
SHOW blue LED off.  
SHOW green LED off.  
SHOW red LED off.  
SET LCD cursor position to 0,1.  
PRINT LCD "ERROR!!"  
REPEAT-UNTIL a temperature is obtained.  
INCREMENT delay 1000 milli seconds.  
  
END-FOR.  
REPEAT-SEQUENCE.
```

Figure 5, pseudo code written to help design the main code

Task 3 – Implementation

```
#include <DHT.h> // Library for the DHT11 sensor.
#include "DHT.h" // Before Setup and Loop, the system needs to know what
functions to include via libraries.
#include <LiquidCrystal_I2C.h> // Includes the library for the LCD display.

#define DHTPIN A3 // This defines what pin the DHT11 is connected to on the
board.
#define DHTTYPE DHT11 // The type of sensor has to be defined before the
main code.

DHT dht(DHTPIN, DHTTYPE); // Creates a DHT object/ address. Parameters: (DHT
Pin, DHT Type).
LiquidCrystal_I2C lcd(0x27,16,2); // Creates a LiquidCrystal object/ address.
Parameters: (RS, Enable (E), 0x27,16,2).

const int greenLED = 8; // Activates a green led light to the RGB LED, pin 8,
names the variable.
const int blueLED = 7; // Activates a blue led light to the RGB LED, pin 7,
names the variable.
const int redLED = 9; // Activates a red led light to the RGB LED, pin 9,
names the variable.

void setup() {
    // The setup code is here, to run once:

    Serial.begin(9600); // Used to establish serial comms. Sets the BAUD rate.

    lcd.init(); // Initializes the interface to the LCD screen.
    lcd.backlight(); // Initializes the backlight of the LCD screen.
    lcd.clear(); // Clears any existing data/ serial prints on the LCD screen.
    lcd.setCursor(0,0); // Sets the cursor to column 0, line 0.
    lcd.print("Hello :)"); // The initial greeting on the LCD screen, to show it
is working.
    delay (1000); // Timed delay of 1000 milli seconds.

    // The following process is to test the RGB LED for functionality, before
starting the temp indication.

    digitalWrite(blueLED, HIGH); // HIGH is the condition to light the LED.
    delay (1000); // Timed delay of 1000 milli seconds.
    digitalWrite(blueLED, LOW); // LOW is the condition to turn off the LED.

    digitalWrite(greenLED, HIGH); // HIGH is the condition to light the LED.
    delay (1000); // Timed delay of 1000 milli seconds.
    digitalWrite(greenLED, LOW); // LOW is the condition to turn off the LED.
```

```

digitalWrite(redLED, HIGH); // HIGH is the condition to light the LED.
delay (1000); // Timed delay of 1000 milli seconds.
digitalWrite(redLED, LOW); // LOW is the condition to turn off the LED.

for (int DigitalPin = 7; DigitalPin <= 9; DigitalPin++) // Defines the Pin
addresses for the RGB LED.

{pinMode(DigitalPin, OUTPUT); // Used to configure the DigitalPin to an
OUTPUT.}

  lcd.begin(16,2); // Initializes the interface to the LCD screen, and
specifies the dimensions (width and height) of the display, after start
sequence.
  dht.begin(); // acitivates the DHT after the start up sequence.
}

void loop() {
  // The main code is here, to run repeatedly:

  delay(1000); // Timed delay of 1000 milli seconds.
  float t = dht.readTemperature(); // Reads the temperature as Celsius (the
default).

  lcd.clear(); // Clears any existing data/ serial prints on the LCD screen.
  lcd.setCursor(0,0); // Sets the cursor to column 0, line 0.
  lcd.print("Temp: "); // Tells the LCD to show "Temp:" on screen.
  lcd.print(t); // This prints the temperarture to the LCD display.
  lcd.print(" C"); // This prints the Celsius indication to the LCD display.

//The following is the setup for the RGB indication and LCD display messages.
// an "if" statement lets the program know if it should execute a particular
block of code, this relies on a variable.
// an "else if" statement lets the program know to run a block of code, if
conditions change from the "if" statement parameters.

if (t>24) // if the temperature rises above 24 degrees, the red LED becomes
active.
{digitalWrite(redLED, HIGH); // Red LED on.
digitalWrite(blueLED, LOW); // Blue LED off.
digitalWrite(greenLED, LOW); // Green LED off.
lcd.setCursor(0,1); // Sets the cursor to column 0, line 1.
  lcd.print("TOO HOT!!"); // prints a warning message below the temperature
indication.}

```

```

else if (t<=24, t>18) // else if the temperature is less than or equal to 24
degrees, and more than 28, the green LED becomes active.
{
    digitalWrite(greenLED, HIGH); // Green LED on.
    digitalWrite(blueLED, LOW); // Blue LED off.
    digitalWrite(redLED, LOW); // Red LED off.
    lcd.setCursor(0,1); //Sets the cursor to column 0, line 1.
    lcd.print("JUST RIGHT :)"); // Prints a message below the temperature
indication.
}

else if (t<=18, t>16) // else if the temperature is less than or equal to 18
degrees, and more than 16, the ble LED becomes active.
{
    digitalWrite(blueLED, HIGH); // Blue LED on.
    digitalWrite(greenLED, LOW); // Green LED off.
    digitalWrite(redLED, LOW); // Red LED off.
    lcd.setCursor(0,1); // Sets the cursor to column 0, line 1.
    lcd.print("TOO COLD!!"); // Prints a warning message below the temperature
indication.
}

else if (t<=16) // else if the temperature is less than or equal to 16
degrees, the ble LED remains active, different warning on LCD.
{
    digitalWrite(blueLED, HIGH); // Blue LED on.
    digitalWrite(greenLED, LOW); // Green LED off.
    digitalWrite(redLED, LOW); // Red LED off.
    lcd.setCursor(0,1); // Sets the cursor to column 0, line 1.
    lcd.print("LEGAL LIMIT!!"); // Prints a warning message below the
temperature indication.
}

else if (t=NAN) // If the temperature reading is not a number/ the DHT11 has
an issue, the LED will power off.
{digitalWrite(redLED, LOW); // Red LED off.
digitalWrite(blueLED, LOW); // Blue LED off.
digitalWrite(greenLED, LOW); // Green LED off.
lcd.setCursor(0,1); // Sets the cursor to column 0, line 1.
    lcd.print("ERROR"); // Prints a warning message below the NAN message, to
indicate a fault with the sensor.
    delay (500); // 500 milli second delay
}

}

```

A copy of the project code is included above. Comments explaining the code and algorithm are included alongside the code in grey coloured text. The program code/ sketch has used 7224 bytes (22%) of program storage space.

Task 4 – Testing

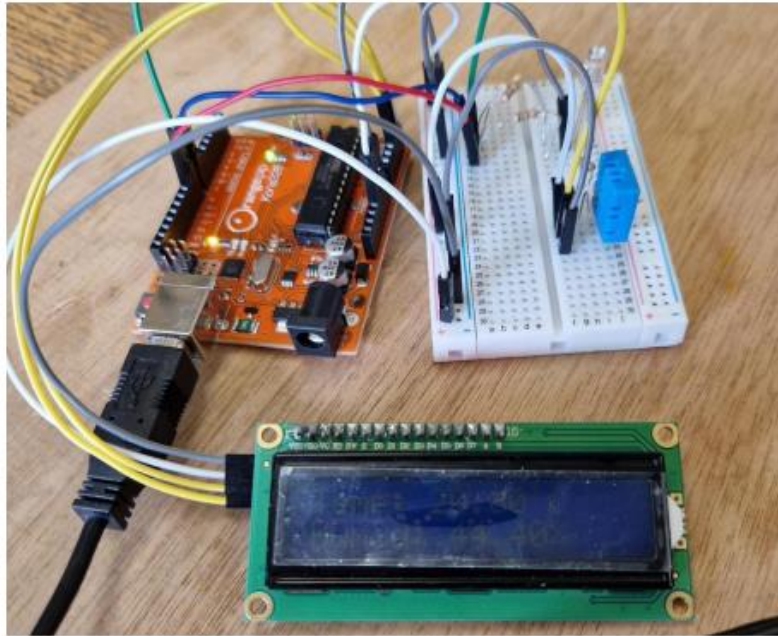


Image 1

Image 1 shows the set-up of the system, with the OrangePi board, breadboard, LCD display and USB connection all assembled, as per planning.



Image 2

Image 2, The system has been connected to the laptop for a power supply and the program has been uploaded to the board. As per the code, the LCD display powers up first and displays the "Hello :)" message.

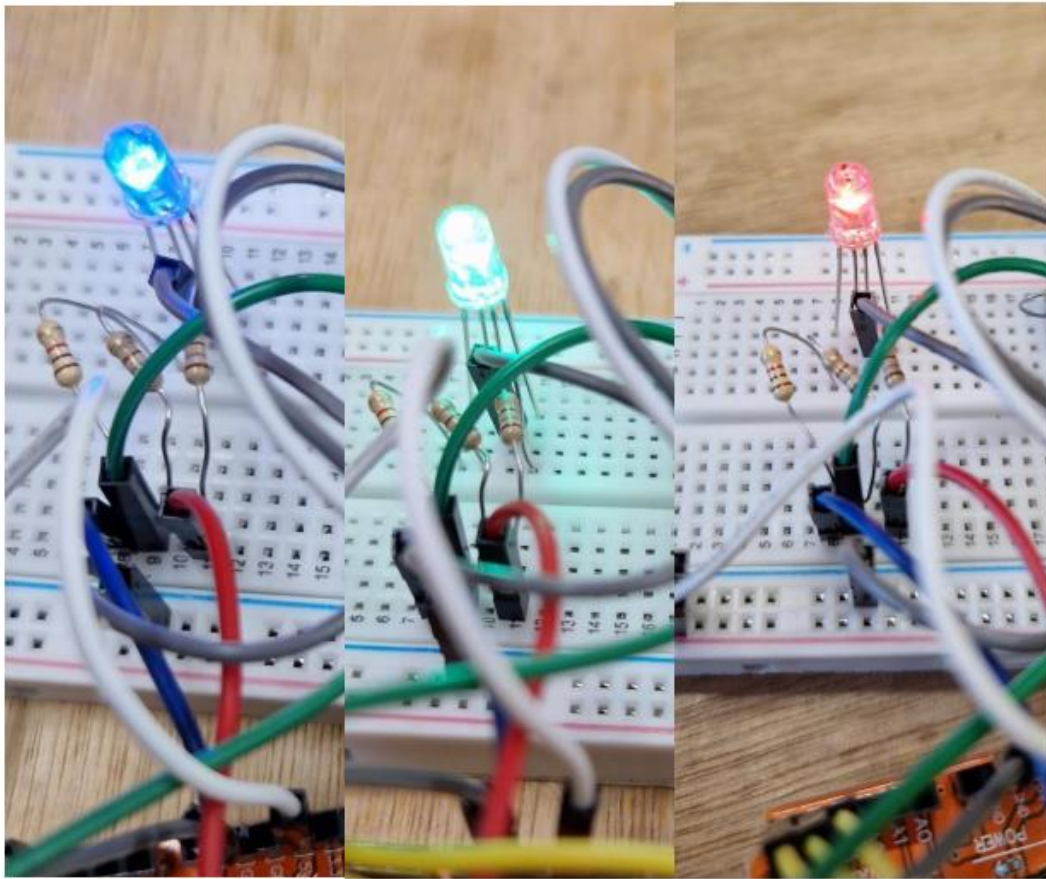


Image 3

Image 3 shows the LED going through it's startup sequence. This is to ensure all three colours are working before taking temperature readings. The blue light shows first, there is a delay, then the green light illuminates, another delay happens and finally the red light comes on. The LED then switches off before the sensor takes it's first reading.

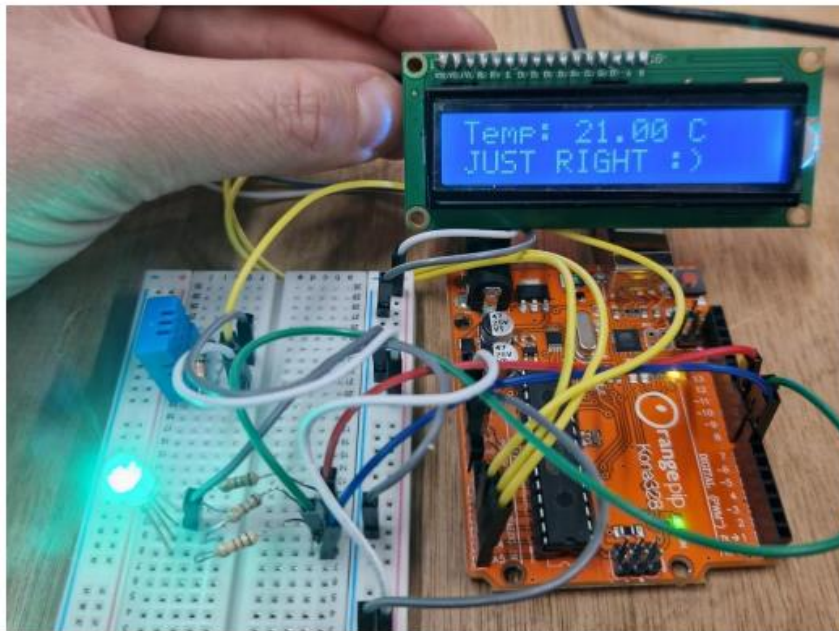


Image 4

Image 4 shows the reading from the sensor at ambient room temperature. The LED is glowing green as expected, and the LCD is showing a temperature of 21 degrees and the "JUST RIGHT :)" message.



Image 5

Image 5 shows the temperature sensor when heat has been applied (via holding the sensor). The temperature went above 24 degrees and as expected, the LED turned red and a warning was showing on the LCD display.

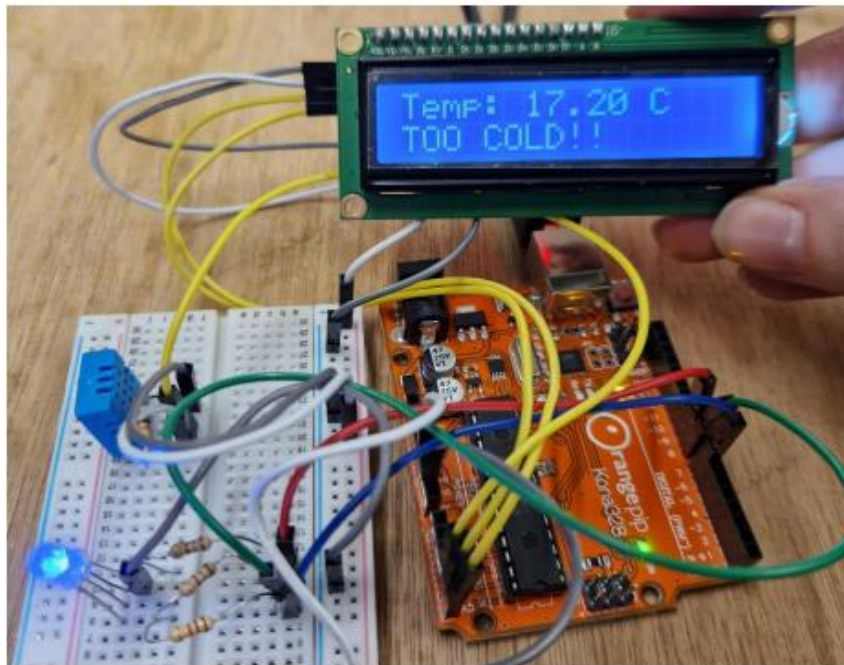


Image 6

Image 6 shows the readings when the temperature drops below 18 degrees. This was achieved by using chilled water in a plastic cup held up against the sensor. The LED turned blue and the LCD display showed the message "TOO COLD!!".

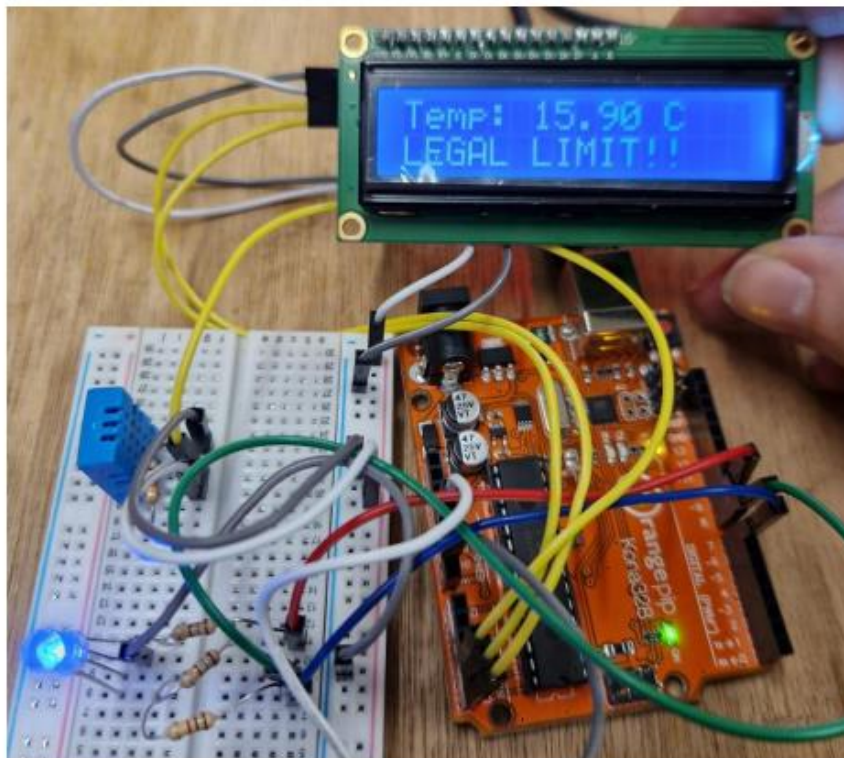


Image 7

Image 6 shows the LED glowing blue after the temperature dropped below 16 degrees. As this is the legal limit in an office space in the UK, it seemed important to include an additional warning of "LEGAL LIMIT!!"

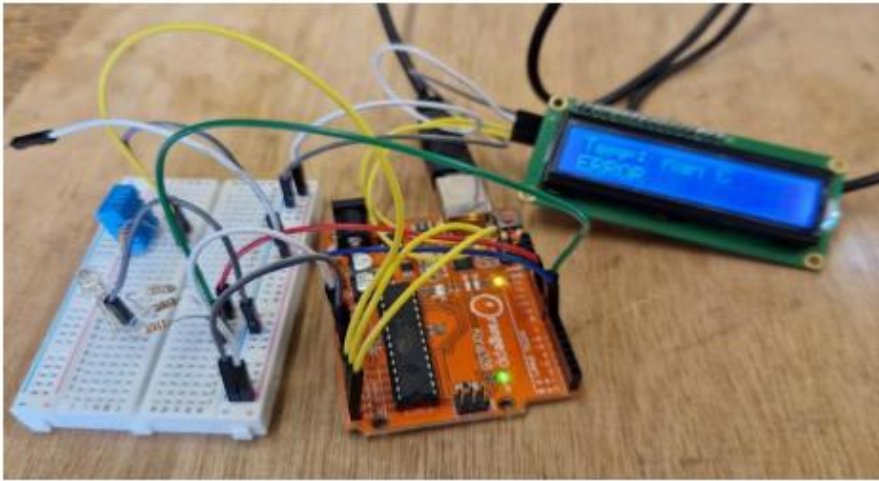


Image 8 shows the state of the LED and LCD display when the sensor is disconnected. The idea is an "ERROR" message will appear when there is no input from the sensor.

Image 8

Input	Output led	Output LCD	Tested Working?	Testing Method
Power On	3 colours with delays	Hello :)	Y	Connect to Laptop
Temp $\leq 24^{\circ}\text{C}$, $> 18^{\circ}\text{C}$	green	Just Right :)	Y	Room temperature
Temp $> 24^{\circ}\text{C}$	red	TOO HOT!!	Y	Warming sensor with finger tips
Temp $\leq 18^{\circ}\text{C}$, $> 16^{\circ}\text{C}$	blue	TOO COLD!!	Y	Using chilled water against the sensor
Temp $\leq 16^{\circ}\text{C}$	blue	LEGAL LIMIT!!	Y	Using chilled water against the sensor
No input/ fault	No output/ blank	Temp Nan C ERROR	Y	Disconnecting the sensor on the breadboard

Task 5 – Discussion, Evaluation and Conclusions

Discussion- what went well

The project was successful in creating a room temperature sensor and indication. The C programming language was relatively easy to learn, user friendly and intuitive. There were libraries available with pre-requisites for the project already programmed, making creating the code simpler. The choice of micro controller was a cheap, effective piece of hardware. The source code written using C programming language can work effectively on another operating system without suffering any changes to the code. The Arduino software was free to download and easy to use to program.

The temperatures selected for the range were based on HSE recommendations and other reputable sources- these could be adjusted for other countries with different laws and environmental conditions. The DHT 11 sensor was cheap and easy to source for the project. The use of an RGB LED reduced the need for 3 LEDs to show different indications and reduced the wiring required. The LCD display is a better option than a 7-segment display, as it requires less pins on the Arduino board and uses less code within the software program. The pseudo code and flow chart were both helpful in designing the code for the project.

Within the code, constants were created at the start to make it easier to read. Using the // symbol, there was opportunity to explain each part of the code, making it easier for the user to implement. Delays, print commands and different operating conditions were implemented within the code, to get the desired outputs to the LCD and LED indications. Temperature ranges were employed within the code successfully, as per the brief.

Evaluation- what could be improved

Concerning the indication of the ambient temperature in the room, a flashing element could have been introduced to the RGB LED when the temperature dropped below the legal limit, to differentiate the reading from the “TOO COLD!!” indication. The brightness of the LED could have been modulated by connecting a potentiometer in series with the LED to adjust resistance values within the circuit. Perhaps another LED could have been used to provide a different colour for the legal limit. Sound could have been implemented to give an audible warning when the temperature becomes too hot or too cold for a comfortable working environment, via using a buzzer element in the circuit.

Greater portability could be implemented via running the temperature sensing system using a battery, rather than a USB connection to a computer or a laptop. A case could be designed to contain and protect the wiring and components of the system. If the wires were available, colour coding could have been more exact in the project. For example, black wires for ground leads, red wires for positive ones could have been used if available. The DHT 11 sensor has a sensitivity of $\pm 2^{\circ}\text{C}$, a sensor with a great temperature range and improved sensitivity is the DHT 22, which has a $\pm 0.5^{\circ}\text{C}$ accuracy between 0 and 50°C .

During the testing stage, a thermometer or other temperature sensing device could have been used to validate the results. These tests could have been done several times to validate the results further. The code itself could possibly be streamlined or shortened to be more efficient and provide the same end result. The same could be said of the planning stage using flow charts and pseudo code.

Conclusions

Overall, the project was successful in producing a program and hardware that reads the temperature of an office space and gives different indications dependant upon the temperature range. When the temperature is outside of the acceptable range, the LED and LCD both show an appropriate indication. There is scope for further improvements to the code and hardware as discussed. Through testing, the program has been shown to be effective and working as designed. References are given in a separate document, appended to the report.

Design Considerations

- Turn the Device on
 - LCD display signifying on Hello 😊
 - LCD display showing temp 22°C
 - Sensing element 
 - power supply $\oplus$ $\ominus$
 - Bread Board for componentry
 - OrangePi micro-controller
 - LED indications
 - HIGH $>25^{\circ}\text{C}$ (RED?)
 - LOW 20°C (GREEN?)
 - LOW LOW / TOO COLD $<16^{\circ}\text{C}$ (BLUE?)
- 
- LCD display could also change? Low
 - Reset when temperature returns to acceptable range.
-  What about a multi-colour LED?
- Reduce Hardware requirements,
- wires/leads
 - resistors for LED and possibly the LCD display



Kona328 Arduino UNO Compatible Development Board

Order code: **75-0550**

Overview

The Kona328 is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Kona328 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the ATmega8U2 programmed as a USB-to-serial converter. Revision 2 of the Kona328 board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Power

The Kona328 can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

VIN The input voltage to the Kona328 board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.



5V The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.

3V3 A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.

GND Ground pins.

Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Kona328 can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

Serial: 0 (RX) and 1 (TX) Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.

External Interrupts: 2 and 3 These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.

PWM: 3, 5, 6, 9, 10, and 11 Provide 8-bit PWM output with the analogWrite() function.

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK) These pins support SPI communication using the SPI library.

Page 1 of 3



Kona328 Arduino UNO Compatible Development Board

Order code: **75-0550**



LED: 13 There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Kona328 has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the `analogReference()` function. Additionally, some pins have specialized functionality:

TWI: A4 (SDA) and A5 (SCL) Support TWI communication using the Wire library. There are a couple of other pins on the board:

AREF Reference voltage for the analog inputs. Used with `analogReference()`.

Reset Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between OrangePi pins and ATmega328 ports?

Communication

The Kona328 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Kona328's digital pins.

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library. Programming

The Kona328 can be programmed with the Arduino software. Select "Arduino Uno" from the Tools > Board menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Kona328 comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

The ATmega8U2 firmware source code is available. The ATmega8U2 is loaded with a DFU bootloader, which can be activated by connecting the solder jumper on the back of the board and then resetting the 8U2. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Kona328 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Kona328 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Kona328. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.



Kona328 Arduino UNO Compatible Development Board

Order code: **75-0550**

The Kona328 contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line.

USB Overcurrent Protection

The Kona328 has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The maximum length and width of the Kona328 PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.



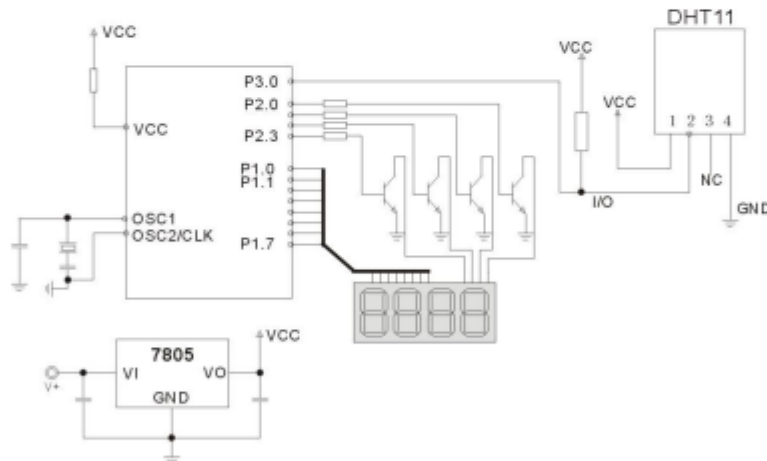
Supply current : Measure 0.3mA Standby 60μA

Sampling period : Secondary Greater than 2 seconds

Pin Description

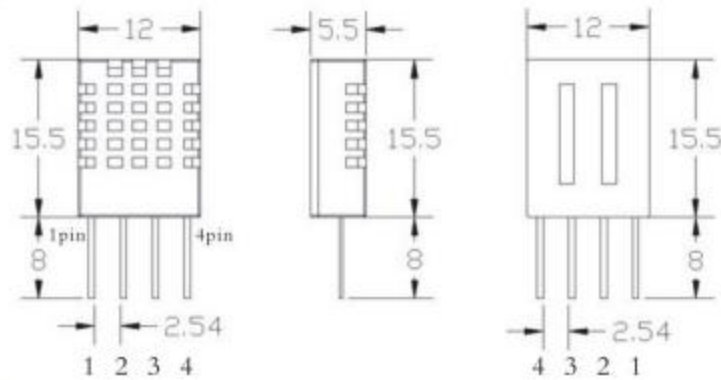
1. VDD supply 3.3 ~ 5.5V DC
2. DATA serial data, single-bus
3. NC NC
4. GND grounding, power negative

6. Typical circuit



Connecting the typical application circuit shown above the microprocessor and DHT11, DATA pull-up and microprocessor I/O port.

1. A typical application circuit recommended cable length shorter than 20 meters with a 5.1K pull-up resistor when greater than 20 meters when the pull-up resistor to reduce the actual situation.
2. When using a 3.3V voltage supply cable length must not be greater than 100cm. Otherwise it will lead to lack of line drop sensor supply, causing measurement bias.
3. Temperature and humidity values are read out every last measurement result, want to get real-time data, to be read twice in a row, but not recommended repeatedly read sensors, each sensor reading interval of more than 5 seconds to obtain accurate data.



5. Parameters

Relative Humidity

Resolution : 16Bit

Repeatability : $\pm 1\%RH$

Accuracy : $25^{\circ}C$ $\pm 5\%RH$

Interchangeability : Fully interchangeable

Response time : $1/e$ (63%) $25^{\circ}C$ 6s

1m/s Air 6s

Hysteresis : $< \pm 0.3\%RH$

Long-term stability : $< \pm 0.5\%RH/yr$

Temperature

Resolution : 16Bit

Repeatability : $\pm 1^{\circ}C$

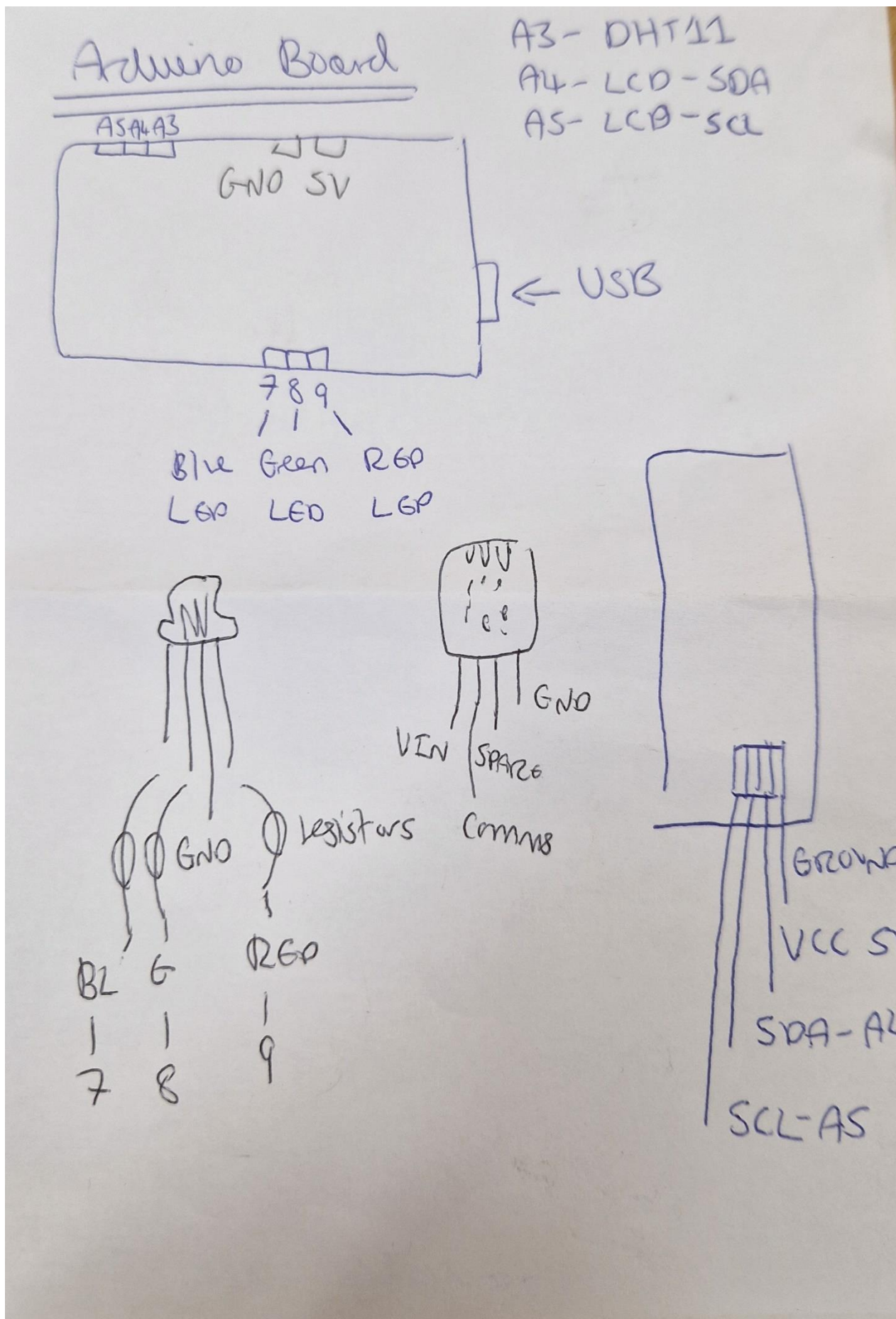
Accuracy : $25^{\circ}C$ $\pm 2^{\circ}C$

Response time : $1/e$ (63%) 10S

Electrical Characteristics

Power supply : DC 3.3 ~ 5.5V

Appendix C, DHT11 temperature sensor



Appendix D, rough circuit design drawing

References

English, C. et al. (2019) Dht11 vs dht22 vs LM35 vs DS18B20 vs BME280 vs BMP180, Random Nerd Tutorials. Available at: <https://randomnerdtutorials.com/dht11-vs-dht22-vs-lm35-vs-ds18b20-vs-bme280-vs-bmp180/> (Accessed: March 30, 2023).

HSE (2023) Thermal comfort, Temperature: Thermal comfort. Available at: <https://www.hse.gov.uk/temperature/thermal/index.htm> (Accessed: March 28, 2023).

Kumar, A. (2023) Why do we use the Arduino programming language? how is it helpful?, Emeritus Online Courses. Available at: <https://emeritus.org/blog/coding-arduino-programming-language/> (Accessed: April 3, 2023).

Metwalli, S.A. (2022) Pseudocode: What it is and how to write it, Built In. Available at: <https://builtin.com/data-science/pseudocode> (Accessed: April 3, 2023).

Nance-Nash, S. (2020) "The never-ending battle over the best office temperature," [bbc.com/worklife](https://www.bbc.com/worklife) [Preprint]. Available at: <https://www.bbc.com/worklife/article/20200826-how-corporate-diversity-initiatives-trap-workers-of-colour> (Accessed: March 28, 2023).