Cluster ambience monitoring

UP896692@myport.ac.uk

***Abstract*—This report presents an indoor IoT device that can measure the ambient temperature, humidity, pressure and light sensitivity for use in a machine room. It was successful in collecting, sending, presenting and alerting of the data. Desktop and mobile dashboards available. Physical sensory and digital notifications included.**

***Index Terms*—IoT, Arduino, Cluster, BMP180, DHT11, LDR, Docker, Grafana, InfluxDB, Alerts.**

I. Introduction

Data centres across the world rely on top performing servers to provide services, they are an intricate platform of computability with thousands of operations happening every second. Not only do they provide a service but they rely on the data of its surroundings to perform that service at the best that it can. IoT inside data centres can provide the needed readings of ambience for critical information against risk to that service and provide metrics for any remote or on-premises technician. With this data, risk of failure and thus downtime can be avoided which leads to the use of these IoT devices to be incorporated in the design of the server rooms.

II. Problem

For my final year project I have a computer cluster that is running I/O tests against different distributed file systems (see Figure 1, p1). These tests can range from two to three hours long and are continuously run throughout the day. Currently there is no ambient monitoring for this cluster. There are certain ranges of temperature and humidity that these computers should be running at to reach its maximum computing potential, whilst keeping the environment in a safe condition. Due to the amount of power that is being used it would be useful to reduce power consumption in any other possible way. Furthermore if a fire were to break out it would be useful to know if there was any early warning detection or way of measuring an ambient reading that may point to the possibility of this.

III. Aims and Objectives

The aim of this project is to have a working Arduino ambience reader that can record metrics from a live environment, display that data and alert a user if the data in certain areas go above or below a certain threshold.

- Measure the temperature and humidity from the environment around the device.
- Find a way to measure the ambience for fire detection and reduce power consumption.
- Send this data to a live dashboard that can be accessed by the user.
- Have the dashboard be accessible over the LAN.

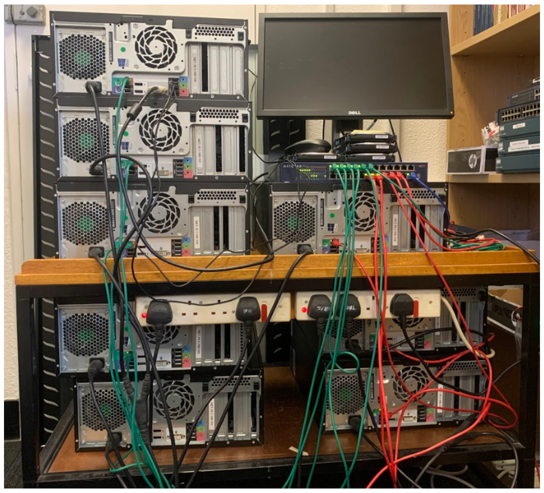


Figure 1. My final year project cluster.

- Send alerts to the user if temperature or humidity go above or below a certain threshold.
- Have a physical based alerting system if the live dashboard is unreachable.

IV. Design Discussion

A. Temperature

Temperature monitoring has been continuously on the radar for data centre technicians. Due to the size of the rooms there can be certain hot pockets of air. By monitoring the temperature of the area around the server racks further work around airflow in the room can be analysed and suggested (Hassan et al., 2013). To achieve this monitoring there are a few sensors on the market that can accommodate a small IoT device. DHT11 is a low cost, low power and reliably accurate sensor for temperature monitoring (Asair, 2020b) ranging from 0 to +50°C. Due to its size and cost this is usually shipped with starter kits for Arduino. Another good sensor for temperature monitoring is the BMP180, again a low cost, low power and accurate temperature range that runs from -40 to +85°C (Bosch, 2013). This is a small compact sensor that is similar in size to that of the DHT11. DS18B20 which is a probe style sensor allows for either air or water contact with the probe. It is a cost effective sensor for its specification with ± 0.5°C accuracy (within a certain range) and total range of -55 to +125°C. DS18B20 is a highly specialised sensor that can be programmed to your specific needs (Maxim-Integrated, 2019).

B. Humidity

Humidity in the data centre is a balancing factor, if your humidity is too high you will be more likely to run into condensation issues whilst a low humidity may cause higher static in the surroundings (Wan et al., 2013). There are sensors that offer humidity readings. DHT11's main purpose is to offer these readings. It can provide a measure in the range of 20-80% with ±5%RH accuracy (Asair, 2020b). The version up from this is the sister DHT22 sensor. It is a little larger and a bit more costly however it provides a larger humidity sensor range of 0-100% with a 2-5%RH accuracy (Liu, n.d.). Both of these are low power using maximum 5 volts. A newer updated version of the DHT22 is the AHT20 (Asair, 2020a). AHT20 is a smaller form factor with the same range as the DHT22 sensor yet the accuracy is sharper with a guaranteed ±2%RH and the module is physically smaller.

C. *Fire detection*

Safety and cause within the data centre environment is something that is needed for monitoring. If you can detect that cause early you may be able to save the service from danger. There are suggested ways such as gas sensing and even pressure sensing that can be flagged for a potential on going fire in the environment (Chen et al., 2019; Johnson, 2010). By using this knowledge, some sensors can be examined for potential use. MQ2 is a gas sensor that can detect butane, methane, alcohol, hydrogen and smoke down to the parts-per million (ppm) (Winsen, 2022). This particular sensor has a stable performance, low cost and long life with fast reaction to the environment. Alternatively the BMP180 has pressure also built into the sensor. It ranges from 300-1100 hPa with an absolute accuracy of ±1hPa (Bosch, 2013). BMP280 is the successor to BMP180 where both range and hPa accuracy are the same with just an additional SPI standard interface (Bosch, 2015).

D. *Power consumption*

There are a few ways in which power of a can be measured and ambient usage reduced. The one way would be to have a power draw sensor attached to one of the cluster machines. CT sensors such as the SCT-013-000 would allow for capture of the AC current (YHDC, n.d.). This would help calculate the overall rough power consumption of the cluster. Regarding ambience around the cluster, by reducing the light usage in the room. Power consumption can be reduced in this aspect. The simplest form of capturing this would be via an light dependent resistor (LDR) (Electronics, n.d.). Measuring the output strength of the LDR would give a rough light percentage when converted correctly.

E. *Data delivery*

Wireless IoT is very popular due to the affordability and scalability in the current climate (Al-Turjman, 2017). Ensuring a strong method of data collection and delivery certain aspects need to be taken into account. Wifi sensors can ensure this delivery of data by acting alone to send the metrics to the specific destination. The ESP-01 is a cheap simple module that runs with low power, allowing the 802.11 b/g/n protocol with top 2.4GHz frequency range (AiThinker, 2015). ESP32 is a newer module that not also supports the same protocols as the ESP-01 but also e/i for QoS and enhanced security. Not only that but ESP32 has bluetooth capabilities (Espressif, 2022). Another way of data delivery is via serial, this would allow for a physical link between client and host.

V. Design

A. *Selected Sensors*

For the sake of the project the BMP180 has been chosen due to its accurate and 3.3 volts low power temperature reading. The other reason behind this sensor is that it measures pressure. Due to time constraints and the ability to test components, pressure will be measured for its information regarding fire safety rather than gas. This means that this one sensor is providing two inputs for the project. DHT11 has been opted for humidity sensing due to already having the sensor in hand, its familiarity and reliability within the required humidity range. A CT sensor to reduce power consumption is not on par with the scope of this device, therefore an LDR would be better suited to reduce power consumption of the surroundings, power consumption of the cluster is needed to deliver the highest performance I/O, yet by measuring the ambient light sensitivity power can be saved by turning off the bulbs. For communication the ESP-01 is a good choice for this project due to its programmability and low cost. It has been adopted in a wide range of small projects where only a number of measurements are needed to be collected. A sensible library to use with this module is WifiESP (bportaluri, 2019), providing the ability for HTTP POST requests to external servers. Bluetooth in this case is not required. Serial is also being adopted as a back up form of data delivery. An Arduino is being used for the base device due to the simplicity of data collection and already having the device to hand. A Raspberry Pi would be too powerful for the job required.

B. *Interface*

In order to provide a graphical user interface a 16x2 LCD is being used. This will provide a clear in person view of the current readings and their retrospective metric names. The dashboard for LAN and WAN access will consist of a dash software and database. Namely Grafana and InfluxDB which have been used before in other IoT solutions and global cluster monitoring. Since these software require a Linux based host they will be set up inside a Docker container allowing them to be configured easily and deployed if needed.

C. *Alerting*

With the purpose of this gathered data being used to get an understanding of the current ambient state around the cluster some alerting should be considered. The proposed two solutions for this will be an LED to blink when certain values go above or below the threshold deemed. The other will be a notification with triggers from inside the Grafana dashboard, namely a Discord notification. The following thresholds will be on the alert "watch list" for both LED and Discord.

- Temperature

Table I
THE COST OF THE IOT DEVICE.

Item	Quantity	Cost
BMP180	1	£3.15
DHT11	1	£2.69
LDR	1	£0.21
LED	1	£0.16
ESP-01	1	£3.40
16x2 LCD	1	£3.45
Arduino Uno Rev3	1	£19.50
Wires & Resistors	∼30	∼£3.00
Total		**£35.56**

Below 15°C or above 29°C
- Humidity
  Below 40%RH or above 54%RH

These have been based around the Ashrae recommended thermal guidelines (Ashrae, 2022).

D. Architecture

With all these choices in mind the design will have a flow of data from the Arduino to the docker container. The user will be able to access data from the server that is hosting that container (see Figure 2, p3). A breakdown of the cost of this project (see Table I, p3) shows that compared to industry options this is much cheaper, there is also the ability to further add more sensors such as the DS18B20, allowing for more temperature readings at different ranges away from the device.

E. Communication and Measuring

```
// global vars
unsigned long previousMillis = 0; // update and
    store
const long interval = 1000;  // delay length

void loop() {
unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval)
    {
    // save the last time it was inside this
    statement
    previousMillis = currentMillis;}
    // do other code inside here every interval
    pass
}
```

Listing 1. Interval without physical delay. (Mellis et al., 2015)

Mentioned in the selected sensors (see subsection V-A, p2) POST requests will be needed to get the information over to the docker server. InfluxDB has a third party Arduino library however does not support ESP-01 (tobiasschuerg, 2020). Therefore the WifiESP library will be needed to send this data by POST. For serial communication measurements will be taken and pushed to the serial port. To get this serial data over to the database a python script will be run to connect to the port, parse and send to the database.

Measurements for BMP180 will be using the SFE_BMP180 library due to its diversity of functions that allow for quick and accurate measurements. DHT library will be used for the DHT11 sensor. All of this will be run along side a millisecond function (see Listing 1, p3) to allow for no physical delay within the code bar a one second physical delay for measurement start, stop. Finally LiquidCrystal library will be used for the setup and display of the LCD.

VI. IMPLEMENTATION

A. Measurements

The BMP library requires a start and record measurement. Delay is needed for accurate temperature readings, this was set to 1 second due to the accuracy given for this time being adequate. DHT uses a similar record function yet no start required, this was done right after temperature and pressure had been assigned. The light is measured via the analog pin, due to this pin having an analog to digital converter, 10-bit values from 0 to 1023 can be read and converted into a percentage using a map function (see Listing 2, p3). Once all variables have been collected they are assigned to global variables to be used elsewhere in the code.

```
int aL = analogRead(ldr);
L = map(aL, 0, 1023, 0, 100);
```

Listing 2. Work out % out of 100 of light sensitivity.

B. Communication

During setup the wifi module is probed to check if it is connected to the Arduino. If this is not true then a LCD

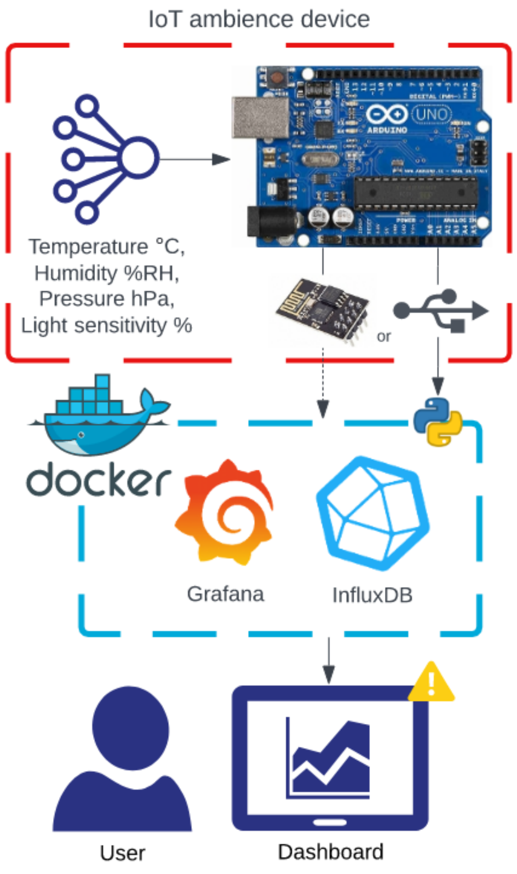


Figure 2. Technical design of the ambience device.

message alerts that the device does not have wifi connectivity, this is delayed for 20 seconds and LED on board lit for this time. Otherwise if successful a connected message will be shown and move forward. After the initial gathering of measurements that now are assigned to global variables a HTTP POST request is formed (see Listing 3, p5). This is set with an interval of 10 seconds to form the string and send to the database. This will only occur if the wifi connection has been verified as connected. The value F has been passed to the println function to exclude the string from the RAM, too many values would cause this to overload and start misbehaving.

The other method of communication was added via the serial. Again this uses the gathered global variables to print directly to serial separated by commas and will only occur if the WiFi module is not detected. This form of communication required a further python script to probe the port (115200), wait for incoming lines, parse the data into "points" and write to the database.

C. Display

To provide the collected data in person the LCD was split into two screens. One being the raw values that have been collected and the other being the names of the values that are being measured. These values are printed to the display and updated in real time (see Figure 3, p4), after 40 seconds has passed the second screen appears where due to the text value being longer than the display means it scrolls from left to right and back. This allows for the user to see the value name if it was missed the first time.

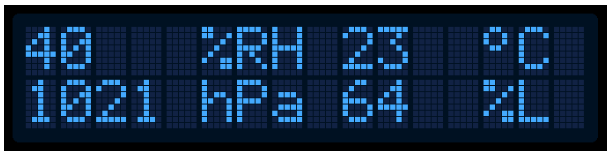


Figure 3. 16x2 LCD on screen example.

Grafana that has been setup with InfluxDB is queried for the populated data sent over from the device. This is a language called flux that is native to InfluxDB. From here a dash of temperature, humidity, pressure, and light sensitivity are set up for desktop view and mobile (see Figure 4, p5).

D. Alerting

On board the IoT device a LED has been included to notify if values exceed the threshold (see subsection V-C, p2). This has a blink with no delay applied to have the LED flash to draw attention to any in person users yet to not cause any physical delay to the rest of the program code (see Listing 4, p4).

Within Grafana, the ability to have alerts notified on many platforms is available. In this case the threshold values were used to alert via Discord. This used a Discord webhook to simply send an alert over the API meaning only configuration was needed for this to occur (see Figure 5, p5). Allowing for very easy set up.

```
void alertLED(){
  if (H < 40 || H > 54 || T < 15 || T > 29){
  // blink the LED when alert is true but without any program delay
  unsigned long currentMillisLED = millis();
  if (currentMillisLED - previousMillisLED >= LEDInterval) {
    previousMillisLED = currentMillisLED;
    if (ledState == LOW) {
      ledState = HIGH;
    } else {
      ledState = LOW;
    }
    }
  else{
    ledState = LOW;
    }
  digitalWrite(LED, ledState);
  }
}
```

Listing 4. Arduino LED alert threshold.

E. Docker and Physical Diagram

Docker compose was used to bundle Grafana and InfluxDB together. With the container running the database, dash and tokens generated. This allowed for the container to be started and stopped when required.

The resulting built diagram of the ambience IoT device is saturated with parts however works in a functioning and modular manner (see Figure 6, p5).

The following sources have aided in the completion of the implementation section. (Arduino, 2022; bportaluri, 2019; Mellis et al., 2015; Pérez, 2022; shazforiot, 2020; sparkfun, 2013)

VII. Conclusion

A. Project Reflection

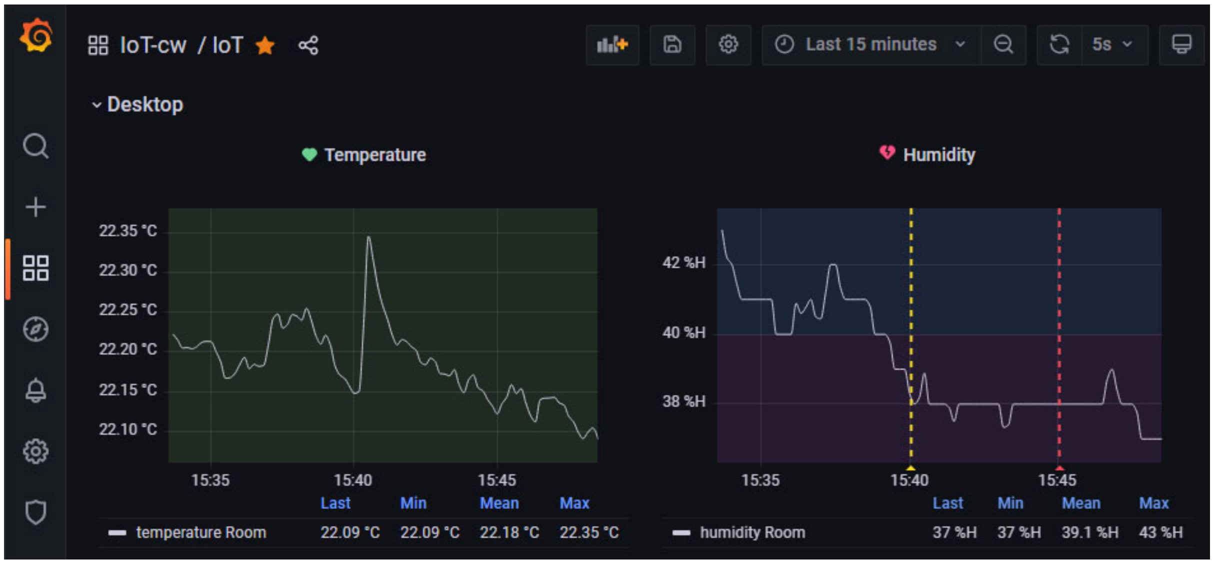
Overall the project has successfully been completed and provides the required data needed for cluster monitoring (see Figure 7, p6). Alerting has been achieved and provides the necessary means of communication to the end user. The used sensors have been reliable and provide accurate results required for the monitoring. Communication between the device and database was also successful yet had been very difficult to get working without running into memory issues on the Arduino itself. Due to the wiring of the device it is very sensitive to movement and can cause the WiFI module to drop off, this is why some error handling has been added to help detect that upon start up, however there is nothing to stop it from occurring during use.

B. Future Work

If re-conducting this work it could be necessary to include the MQ2 gas sensor with the rest of the used sensors. It would further add another layer of detection but due to the time constraint of this project was not possible. Other aspects such as the physical device like a Raspberry Pi could greatly improve this by becoming a self hosted means of being able to collect the data and host the docker container with dashboard.

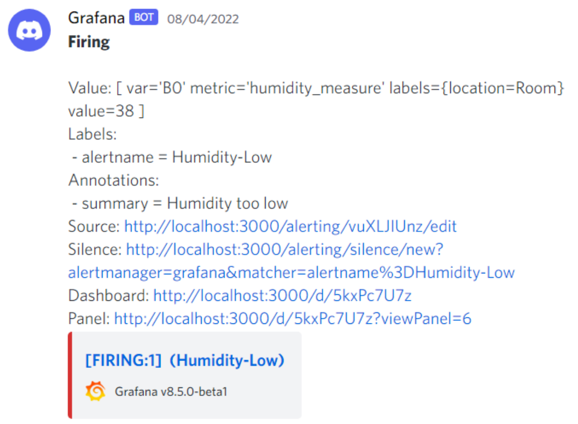
```
// https://docs.influxdata.com/influxdb/v2.2/reference/syntax/line-protocol/
String input = "measurement,location=IoT-ambience humidity=" +  String(H) +"\n";
input       += "measurement,location=IoT-ambience temperature=" + String(T,2) + "\n";
input       += "measurement,location=IoT-ambience pressure=" + String(P,2) + "\n";
input       += "measurement,location=IoT-ambience light=" +  String(L);
int inLength = input.length();

// send the HTTP POST request
client.println(F("POST /api/v2/write?org=ORG_NAME&bucket=DATABASE&precision=ns HTTP/1.1"));
client.println(F("Host: 100.1.0.100:8086"));
client.println(F("Authorization: Token XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"));
client.println(F("Content-Type: text/plain; charset=utf-8"));
client.println("Content-Length:" + String(inLength));
client.println();
client.println(input);
client.println("Connection: close");
// note the time that the connection was made
lastConnectionTime = millis();
```

Listing 3. Building and sending of the POST request to InfluxDB V2.



	Last	Min	Mean	Max
temperature Room	22.09 °C	22.09 °C	22.18 °C	22.35 °C
humidity Room	37 %H	37 %H	39.1 %H	43 %H

Figure 4. Grafana dashboard partially showing temperature and humidity measurements.



Grafana BOT 08/04/2022
Firing

Value: [var='B0' metric='humidity_measure' labels={location=Room} value=38]
Labels:
- alertname = Humidity-Low
Annotations:
- summary = Humidity too low
Source: http://localhost:3000/alerting/vuXLJIUnz/edit
Silence: http://localhost:3000/alerting/silence/new?alertmanager=grafana&matcher=alertname%3DHumidity-Low
Dashboard: http://localhost:3000/d/5kxPc7U7z
Panel: http://localhost:3000/d/5kxPc7U7z?viewPanel=6

[FIRING:1] (Humidity-Low)
Grafana v8.5.0-beta1

Figure 5. Grafana alert via the Discord webhook to channel.

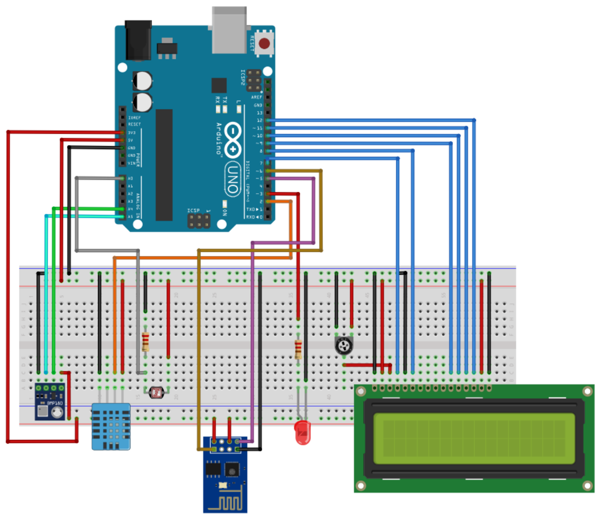


Figure 6. Final circuit of the device.

REFERENCES

AiThinker. (2015). *Esp-01 wifi module*. https : / / www . microchip.ua/wireless/esp01.pdf

Al-Turjman, F. (2017). Price-based data delivery framework for dynamic and pervasive iot. *Pervasive and Mobile Computing*, *42*, 299–316.

Arduino. (2022). *Liquidcrystal - arduino reference*. https://www.arduino.cc/reference/en/libraries/liquidcrystal/

Asair. (2020a). *Aht20 integrated temperature and humidity sensor-sensor-temperature and humidity-guangzhou aosong electronic co. ltd.* http://www.aosong.com/en/products-32.html

Asair. (2020b). *Dht11 sip packaged temperature and humidity sensor(discontinued replaced by dht20) -sensor-temperature and humidity-guangzhou aosong electronic co. ltd.* http://www.aosong.com/en/products-21.html

Ashrae. (2022). *2021 equipment thermal guidelines for data processing environments*. https://www.ashrae.org/file%5C%20library/technical%5C%20resources/bookstore/supplemental%5C%20files/referencecard_2021thermalguidelines.pdf

Bosch. (2013). *Bmp180 digital pressure sensor*. https://cdn-shop.adafruit.com/datasheets/BST-BMP180-DS000-09.pdf

Bosch. (2015). *Bmp280 data sheet document revision 1.14*. https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf

bportaluri. (2019). *Bportaluri/wifiesp: Arduino wifi library for esp8266 modules*. https://github.com/bportaluri/WiFiEsp

Chen, M., Liu, J., Ouyang, D. & Wang, J. (2019). Experimental investigation on the effect of ambient pressure on thermal runaway and fire behaviors of lithium-ion batteries. *International Journal of Energy Research*, *43*(9), 4898–4911.

Electronics, N. (n.d.). *5mm, 12mm, & 20mm ldr radial lead types*. https://www.nteinc.com/resistor_web/pdf/LDR-Series.pdf

Espressif. (2022). *Esp32 series datasheet*. https://www.espressif.com/sites/default/files/documentation/esp32%5C_datasheet%5C_en.pdf

Hassan, N., Khan, M. M. K. & Rasul, M. (2013). Temperature monitoring and cfd analysis of data centre. *Procedia Engineering*, *56*, 551–559.

Johnson, P. F. (2010). Fire detection in computer facilities: 25 years on. *Fire technology*, *46*(4), 803–820.

Liu, T. (n.d.). *Digital relative humidity & temperature sensor am2302/dht22*. https://cdn-shop.adafruit.com/datasheets/Digital+humidity+and+temperature+sensor+AM2302.pdf

Maxim-Integrated. (2019). *Ds18b20 programmable resolution 1-wire digital thermometer*. https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

Mellis, D., Stoffregen, P., Fitzgerald, S. & Guadalupi, A. (2015). *Blink without delay*. https://www.arduino.cc/en/Tutorial/BuiltInExamples/BlinkWithoutDelay

Pérez, I. (2022). *Iván g. pérez / our water footprint gitlab*. https://gitlab.com/ig-perez/our-water-footprint

shazforiot. (2020). *Shazforiot/grafana_influxdb-docker-compose: Docker compose file to create grafana and influxdb*. https://github.com/shazforiot/grafana%5C_influxdb-docker-compose

sparkfun. (2013). *Bmp180 breakout arduino library*. https://github.com/sparkfun/BMP180%5C_Breakout%5C_Arduino%5C_Library/blob/master/examples/SFE%5C_BMP180%5C_example/SFE%5C_BMP180%5C_example.ino

tobiasschuerg. (2020). *Help needed: Arduino uno (clone) with esp-01 to send sensor data to local influxdb instance issue #81 tobiasschuerg/influxdb-client-for-arduino*. https://github.com/tobiasschuerg/InfluxDB-Client-for-Arduino/issues/81

Wan, F., Swenson, D., Hillstrom, M., Pommerenke, D. & Stayer, C. (2013). The effect of humidity on static electricity induced reliability issues of ict equipment in data centers–motivation and setup of the study. *Ashrae transactions*, *119*(2).

Winsen. (2022). *Mq-2 smoke sensor-winsen*. https://www.winsen-sensor.com/sensors/combustible-sensor/mq2.html

YHDC. (n.d.). *Model sct-013-000 0-100a 0-50ma*. https://www.mcielectronics.cl/website%5C_MCI/static/documents/Datasheet%5C_SCT013.pdf

VIII. Final Device

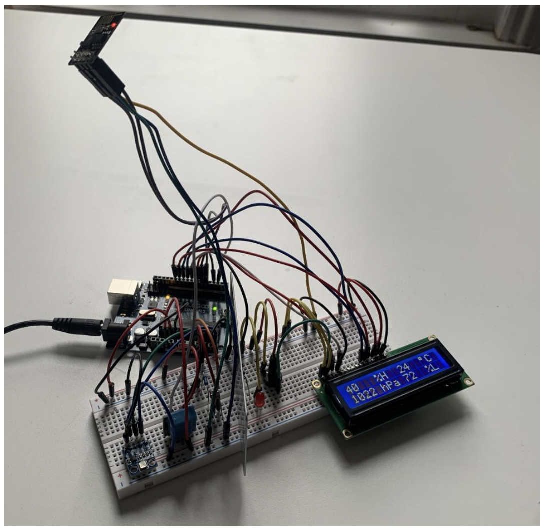


Figure 7. The final built device for monitoring ambience.