Bachelor's thesis

Degree programme in Information Technology

2018

Hoang Viet Anh Le

TEMPERATURE DATA COLLECTION DEVICE

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Degree programme in Information Technology

2018 | 29

Hoang Viet Anh Le

TEMPERATURE DATA COLLECTION DEVICE

The purpose of this project was to create a device to collect temperature data from underground and then send them to the user. The hardware I used to create this device include: Arduino UNO, Dallas digital temperature sensor, 4k7 resistor, electric wires, Raspberry Pi to Arduino Shields Connection Bridge and Sigfox module. The software used for implementing code in order to control the machine and to receive the temperature data include: an Arduino integrated development environment and Sigfox portal. The overall approach was to create an IoT device which uses less energy consumption and is affordable. The result was a device that was functional and easy to install.

KEYWORDS:

Arduino, Dallas digital temperature sensor, Sigfox technology, wireless connectivity, IoT, low power.

CONTENT

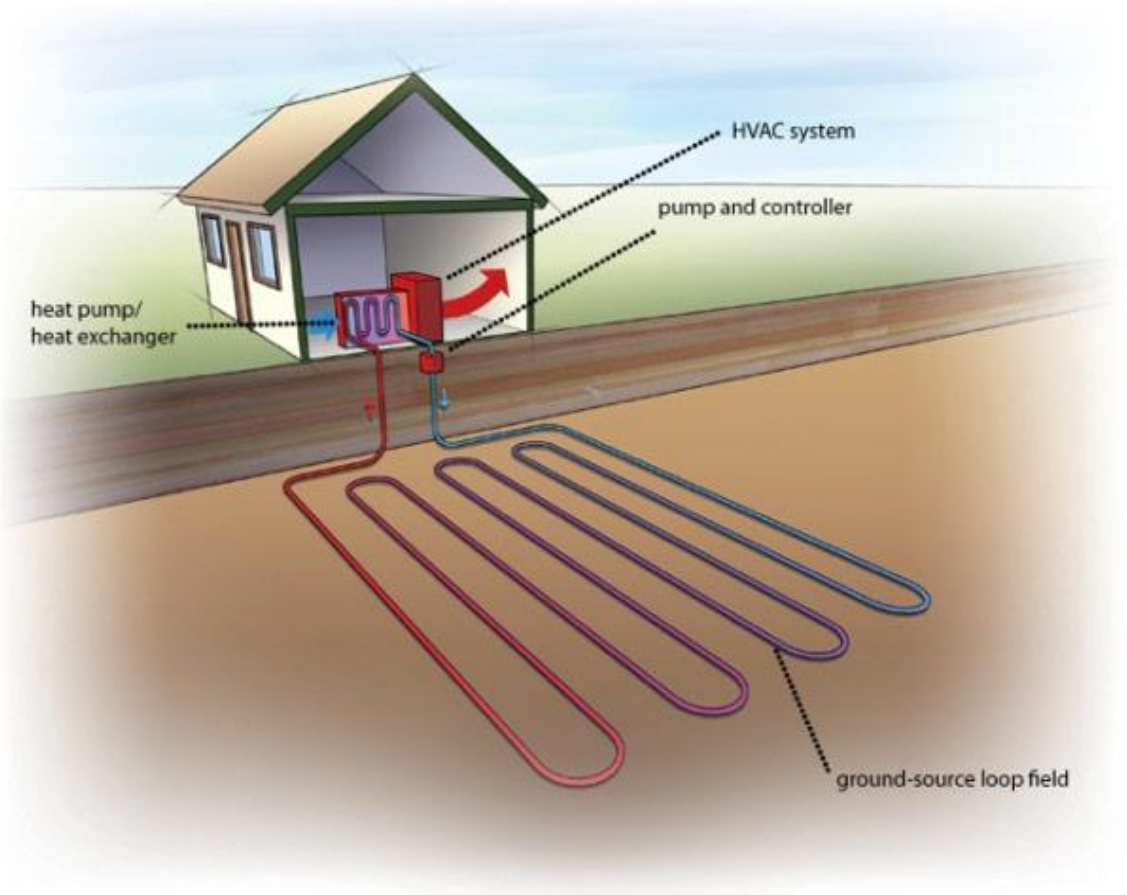
PICTURES

LIST OF ABBREVIATIONS (OR) SYMBOLS

IDE	Integrated design environment
ID	Identification
GND	Graduated neutral density filter
Vdd	Power supply pin
mA	Milliampere
Ah	Ampere per hour
mAh	Milliampere per hour
PVC	Polyvinyl clorua
ADC	Analog-to-digital converter
CPU	Central processing unit

1 INTRODUCTION

Besides solar energy, geothermal or geo-exchange is an efficient way to keep a house cool in summer and warn in winter (Picture 1). For instance, in winter, the heat pump heats by extracting the heat from the top layer of the earth's crust and transferring it into the building. On the opposite side, in summer, the heat pump takes the heat from the building and conduct it to the ground [1]. On average, with this system, the bills reduce more than 25% and the CO2 decreases 400g per one hour [2].



Picture 1. Ground source heat and cooling (https://www.ases.org/resources/solar-home-basics/ground-source-heat-and-cooling/).

Before the heat pump is installed, all the data in the desire area is collected and analyzed. One of them is the temperature data which is the main use case for the temperature data collection device project. There are seven digital temperature sensors in 18 meters cable which were connected in series. Therefore, at certain level in the ground, they can

measure and send the temperature data to the Sigfox backend, then from the cloud the data send back to the customer for analyzing.

The temperature data collection device includes Dallas temperature sensor and Arduino Uno which is the microcontroller. Furthermore, the Raspberry Pi to Arduino Shields Connection Bridge and Sigfox module were connected to the Arduino for sending temperature data to the cloud. Besides that, for reducing energy consumption, the sleep mode was used for the Arduino processors and some useless hardware on the Arduino was removed to decrease the current.

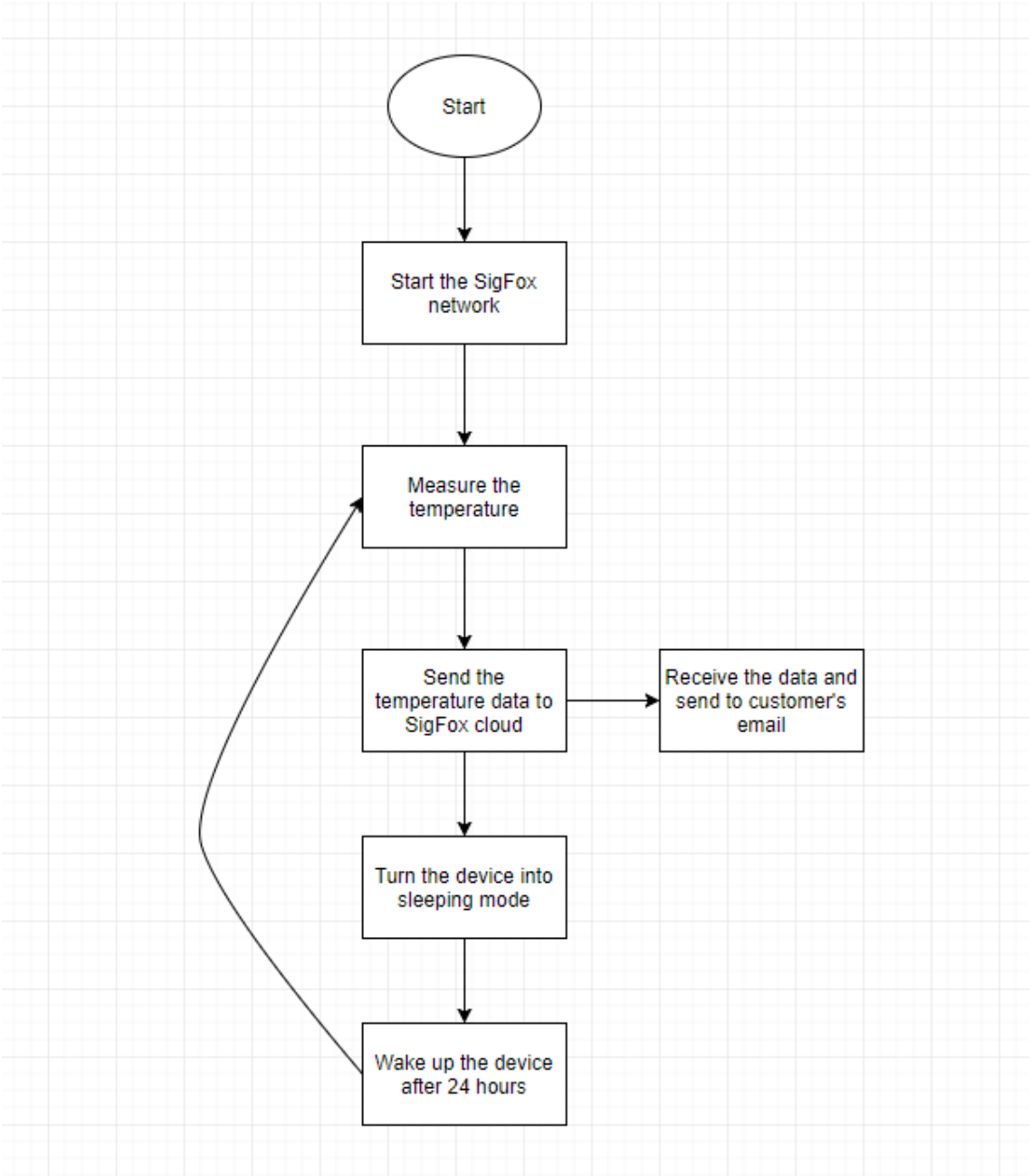
The stages of thesis are addressed as follows: The system design is introduced in Chapter 2 followed by system implementation Chapter 3. The testing and results for final project is covered in Chapter 4. Finally, the conclusion is presented in Chapter 5.

2 SYSTEM DESIGN

This chapter presents an overview about the integration of the temperature data collection device. According to the software, Arduino integrated development environment and libraries are easy to find on https://www.arduino.cc and Sigfox portal is set up on https://backend.Sigfox.com . Furthermore, the hardware components include an Arduino Uno, Dallas digital temperature sensor, Raspberry Pi to Arduino Shields Connection Bridge and Sigfox module. Regarding the design, the device aims to low cost and low energy consumption.

2.1 System software design

The diagram in Picture 2 illuminates the basic processing of temperature data collection device.



Picture 2. The block diagram of temperature data collection device.

First, the microcontroller which is Arduino checks and commences the Sigfox protocol. Then the Dallas temperature sensor measures the temperature and Arduino sends the

data to Sigfox back-end. Finally, for reducing the energy consumption, the device turns to sleeping mode and wakes up in 24 hours to continue the process again.

2.2 System hardware design

Basically, the microcontroller Arduino, the Raspberry Pi to Arduino Shields Connection Bridge and Sigfox module are assembled in a waterproof box, and the temperature sensors cable puts underground. According to this concept, the project aims to be user-friendly with Plug and Play technology.

2.2.1 Arduino Uno

Picture 3. Arduino Uno REV3 (https://store.arduino.cc/arduino-uno-rev3).

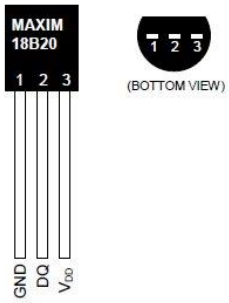
Arduino Uno (Picture 3), the microcontroller, the center of the devices, which based on ATmega328P. It contains 14 digital input or output pins, 6 analog inputs, a 16 MHz quartz crystal, a power jack and a USB connection [3]. Moreover, digital pins only have 0 (LOW) and 1 (HIGH) states. For example, when the current goes through the LED with 3.3 volts the state of the LED is 1 and vices era, it is 0 state when 0 volt in LED. Digital pins cannot read the variable of the sensors; however, analog pins can do it with the support value from 0 to 1023 bits. For instance, the rain sensor, which has the value about 350 when

it rains. Furthermore, the analog pins can measure the distance with ultrasonic sensor or the volumetric water in soil with moister sensor.

2.2.2 Dallas digital temperature sensor

There are seven DS18B20 (Picture 5) sensors in the cable (Picture 4). The length of the cable is around 20 meters and the distance between two sensors is three meters. The outstanding of DS18B20 is a unique 64-bit Serial number named into it, therefore, man sensors can connect on one bus data.

Picture 4. Dallas temperature sensors cable.



Picture 5. DS18B20 sensor (http://www.hobbytronics.co.uk/ds18b20-arduino).

However, this cable does not come with the datasheet which has the location and serial number of sensors. The more challenge project is the more I am interested, one solution came up which tested and checked each sensor one by one.

```
Found '1-Wire' device with address:
0x28, 0x14, 0xD6, 0x29, 0x08, 0x00, 0x00, 0x9D

Found '1-Wire' device with address:
0x28, 0xDC, 0xE7, 0x29, 0x08, 0x00, 0x00, 0xEA

Found '1-Wire' device with address:
0x28, 0xE2, 0xA5, 0x29, 0x08, 0x00, 0x00, 0x7E

Found '1-Wire' device with address:
0x28, 0x86, 0x1A, 0x2A, 0x08, 0x00, 0x00, 0xC9

Found '1-Wire' device with address:
0x28, 0x3B, 0xD1, 0x29, 0x08, 0x00, 0x00, 0x5E

Found '1-Wire' device with address:
0x28, 0x47, 0xFF, 0x5A, 0x08, 0x00, 0x00, 0xC1

Found '1-Wire' device with address:
0x28, 0x5F, 0x7F, 0x29, 0x08, 0x00, 0x00, 0xD0
```

Picture 6. Seven DS18B20 sensor addresses on the cable.

1	0x28, 0x5F, 0x7F, 0x29, 0x08, 0x00, 0x00, 0xD0
2	0x28, 0x86, 0x1A, 0x2A, 0x08, 0x00, 0x00, 0xC9
3	0x28, 0xDC, 0xE7, 0x29, 0x08, 0x00, 0x00, 0xEA
4	0x28, 0x47, 0xFF, 0x5A, 0x08, 0x00, 0x00, 0xC1
5	0x28, 0x3B, 0xD1, 0x29, 0x08, 0x00, 0x00, 0x5E
6	0x28, 0x14, 0xD6, 0x29, 0x08, 0x00, 0x00, 0x9D
7	0x28, 0xE2, 0xA5, 0x29, 0x08, 0x00, 0x00, 0x7E

Picture 7. Location and serial number of seven sensors.

First, Arduino was used to print out the addresses (Picture 6) of seven sensors. After that, on the cable, the sensors were marked down from one to seven (Picture 7). Finally, Arduino was used again to print out and see the fluctuation of the temperature, while hand was used to grab each sensor one by one.

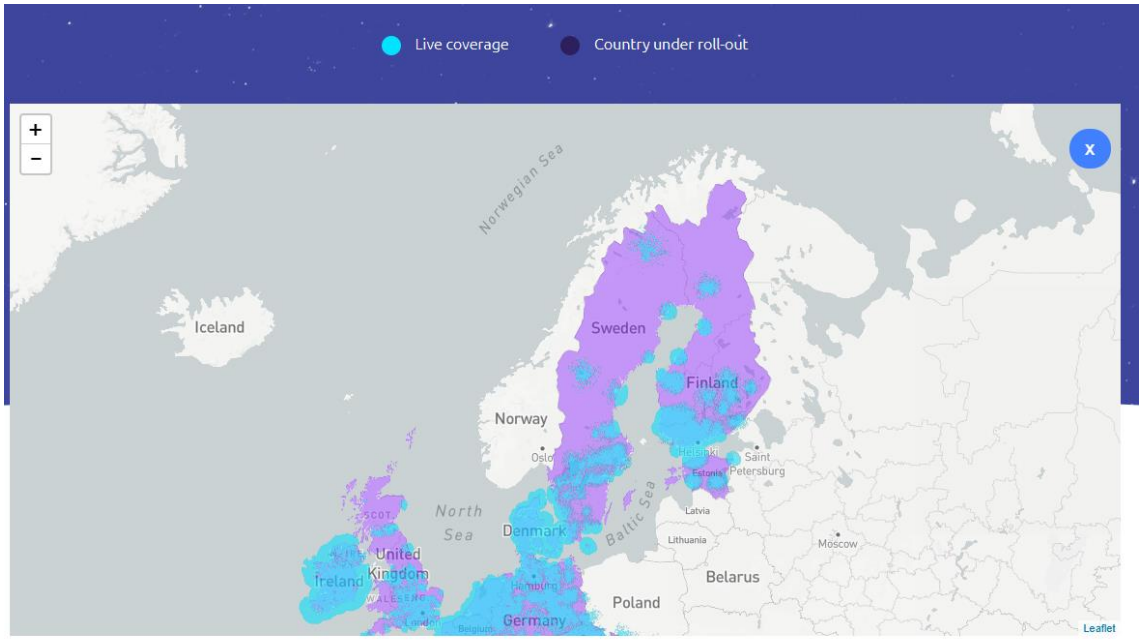
2.2.3 Raspberry Pi to Arduino Shields Connection Bridge and Sigfox module

This Connection Bridge (Picture 8) aims to use all the shields, boards and modules designed for Arduino in Raspberry Pi [4]. However, for the low energy consumption project, Arduino was used to connect to the Bridge which reduces the power consumption.

Picture 8. Raspberry Pi to Arduino Shields Connection Bridge and Sigfox module (https://www.cooking-hacks.com).

The Sigfox technology is the first IoT network to connect to billions of objects transmitting data in the world, without the process of launch and sustain network connections. On the other hand, Sigfox provides a back-end on the cloud, where all the network and calculating complexity are managed. For all that features, it harshly decreases energy consumption and fee of associated devices [5].

The Sigfox technology has approximately 800 million users, covers about 3.8 million km^2 and 45 countries. For instance, the Sigfox network is covered most of South Finland. The blue area in Picture 9 is the signal of Sigfox network.



Picture 9. The network of Sigfox in Finland (https://www.Sigfox.com/en/coverage).

Basically, in the coverage area, the devices with a Sigfox module (Picture 8) is possible to connect and send maximum 140 packets of 12 bytes of data per day [6]. Regarding to the project, it is perfectly for sending the temperature data once per day without getting the limitation of Sigfox network.

3 SYSTEM IMPLEMENTATION

Following the design of temperature data collection device, this chapter addresses the implementation of Arduino Uno, Dallas digital temperature sensors cable, Raspberry Pi to Arduino Shields Connection Bridge and Sigfox module. Furthermore, the embedded code and user interface are stated below.

3.1 Hardware implementation

At the beginning, the Sigfox module was placed upper the Raspberry Pi to Arduino Shields Connection Bridge. Then, all the Arduino pins were connected respectively to the Bridge with electronic wires, because it was unknown which pins of the Bridge is needed for the Sigfox module. For that reason, it was decided to check the pins which embedded the code and checked the status on monitor by using Serial.print(). Serial.print() is the command to print and illustrate on the Serial Monitor which uses to see the output of the Arduino.

```
72  void setup()
73  {
74    // start serial port
75    Serial.begin(9600);
76
77    Serial.println();
78    Serial.print("Sigfox ID: ");
79    Serial.println(Sigfox.getID());
80    pinMode(error_led, OUTPUT);
81
82    ////////////////////////////////////////////////
83    // 1. switch on
84    ////////////////////////////////////////////////
85    error = Sigfox.ON(socket);
86
87    // Check status
88    if ( error == 0 )
89    {
90      //"Switch ON OK"
91      digitalWrite(error_led, LOW);
92    }
93    else
94    {
95      //"Switch ON ERROR"
96      digitalWrite(error_led, HIGH);
97    }
98
99    // Start up the library
100   sensors.begin();
101
102 }
```

Picture 10. Get the Sigfox ID code.

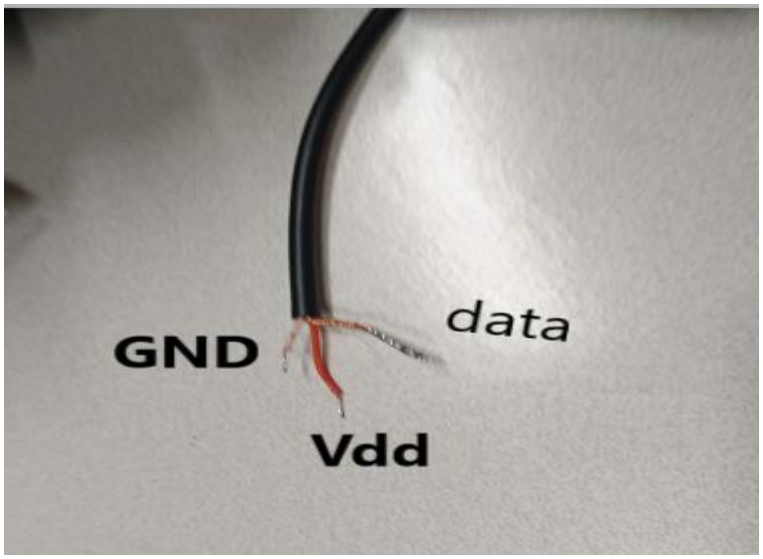
In this testing, the ID of Sigfox module printed out on Serial Monior of Arduino IDE (Picture 10). Besides that, one by one, the electronic wires which connected Arduino and the Bridge was unplugged. All the wires were removed when the ID still illustrate on the Monitor.

After the examination, only 9 pins were used to connected them, instead of 30 pins. For instance, two ground pins and 5 volts sources in the Arduino were connected to two GND pins and 5 volts sources in the Bridge. The ground pins in the Arduino were wired to the ground pins in the Bridge. Moreover, two analog pins A1 and A5 in the Arduino and the Bridge were connected respectively. The pins number 1, 2, 3 and 4 in the Arduino were wired to pins number 1, 2, 3 and 4 in the Bridge (Picture 11).

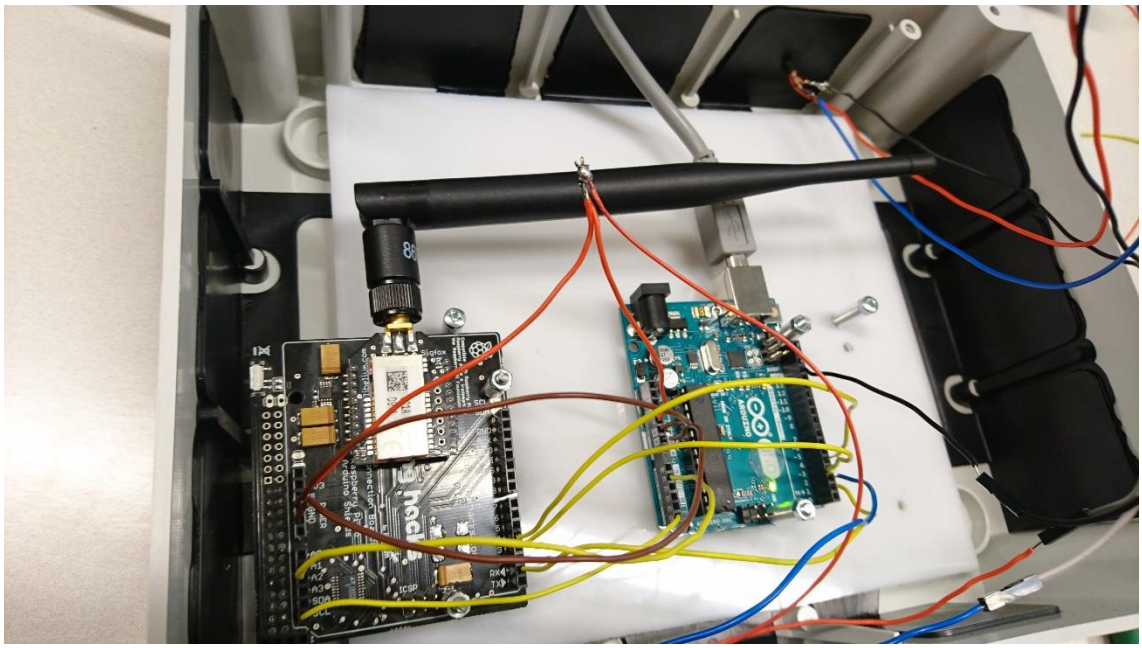
Arduino pins	Cooking hacks raspberry pi shield for Sigfox
5V	5V
GND	GND
GND	GND
A1	A1
A5	A5
GND	GND
4	4
3	3
2	2
1 (TX)	1 (TX)

Picture 11. Pins connection of Arduino and Raspberry Pi to Arduino Shields Connection Bridge.

Finally, the 4k7 resistor was soldered between the Vdd wire and the data wire of the cable. On the other hand, the data wire was connected to pin 2 in the Arduino. Furthermore, the GND and Vdd of the cable were connected to the ground pin and the 5 volts source in the Arduino respectively (Picture 12). The result after all the components were connected is in Picture 13.



Picture 12. Vdd, GND and data wires on Dallas cable.



Picture 13. The final design of Temperature data collection device.

3.2 Software implementation

All the code for Arduino was used C language to get temperature sensor data, send data to Sigfox cloud, set the microcontroller to sleep mode and wake it up once a day. The Sigfox packets maximum loads is 12 bytes [7], therefore, the sensor's data was sent one by one, from sensor 1 to 7.

The website https://backend.sigfox.com is used to watch the messages from Sigfox module send to Sigfox cloud. However, all the messages were encoded, so they need to decode in custom format of Sigfox backend. For instance, on the display custom page, the custom format was "Temp::float:32:big-endian Sensor::uint:8", "Temp" and "Sensor" are the name of the values. After "::" was the variable of the messages. For example, the receive message was "41ac000001", and it portrayed "Temp: 22.00 Sensor: 1".

Finally, the callbacks section was used for sending all data to customer, which was google mail in this project. For sending the messages to email, the messages on Sigfox backend page also need to change the format. For example, the sensor number from message was attained by using "Sensor: {customData#Sensor}" and for the temperature was "Temperature: {customData#Temp} Celsius degree". The "customData" get the data from the custom message, after "#" was the name of the data. So that, after the Sigfox backend demonstrates the message, immediately, it sent the data to the custom email.

4 LOW POWER SOLUTION

For the reducing energy consumption, there are software and hardware solutions. For instance, some unnecessary components on the Arduino can be removed to stop the flow of current, such as LEDs, NCP1117. Moreover, according to the datasheet of two processors on the Arduino which are Atmega16U2 and Atmega328P, the sleep mode can be used to decrease the consumption of the devices.

Picture 14. D, C, AA, AAA, AAAA and 9-Volt batteries (https://en.wikipedia.org/wiki/D_battery).

Regarding to the battery research, the D battery or IRC R20 (Picture 14) is used in high current drain devices, such as flashlight, radio receivers and transmitters, products with electric motors. The lifetime of battery depends on its cell chemistry and current draw. For instance, alkaline D battery of Energizer brand is at approximately 20Ah at 25mA draw and about 10Ah at 500mA draw. Normally, it has 12Ah capacity and 1.5 voltages [8].

Current per hour:

$$\frac{12\ Ah}{87659\ h} \approx 0.14\ mA$$

Sleep mode:

Current in 23.96 hours:

$$23.96\ h * \ 0.14\ mA \approx 3.36\ mAh$$

Active mode:

Current in 0.04 hours:

$$0.04\ h * \ 100\ mA \approx 4\ mAh$$

Picture 15. Desire current for 10 years D-battery.

Therefore, alkaline D battery which is the most suitable power supply for this project. Assuming the device is used three 1.5 voltages D batteries with 12Ah capacity for ten years which is 87659 hours, thus the current consumption in every hour is about 0.14 mA, in sleep mode for one day is approximately 3.36 mAh. Moreover, the device operates in 0.04 hours and uses 100mA, thus, it takes 4 mAh per day (Picture 15).

In order to decrease the power consumption to the nearest desire current, the hardware and software solutions were applied to the project.

4.1 Software solution

According to the schematic of Arduino Uno, there are two processors which are Atmega16U2 and Atmega328P. The Atmega16U2 (Picture 16) is the bridge between USB port of computer and serial port of main processor in the Arduino [9]. The Atmega328P is the main processor which controls all the ports of Arduino.

Regarding to the datasheet of Atmega16U2 and Atmega328P, the Atmega16U2 has five sleep modes and the Atmega328P has six sleep modes. However, in C header (avr/sleep.h), it supports only five sleep modes. For instance, they are Idle

(SLEEP_MODE_IDLE), ADC Noise Reduction (SLEEP_MODE_ADC), Power-save (SLEEP_MODE_PWR_SAVE), Standby (SLEEP_MODE_STANDBY) and Power-down (SLEEP_MODE_PWR_DOWN).

In idle mode, the CPU is stopped but it can be wake up by almost everything, such as connecting wire, interrupting pins, and input or output sources. The next sleep mode, ADC noise reduction mode, not only stops the CPU but also halts all the input or output clocks.

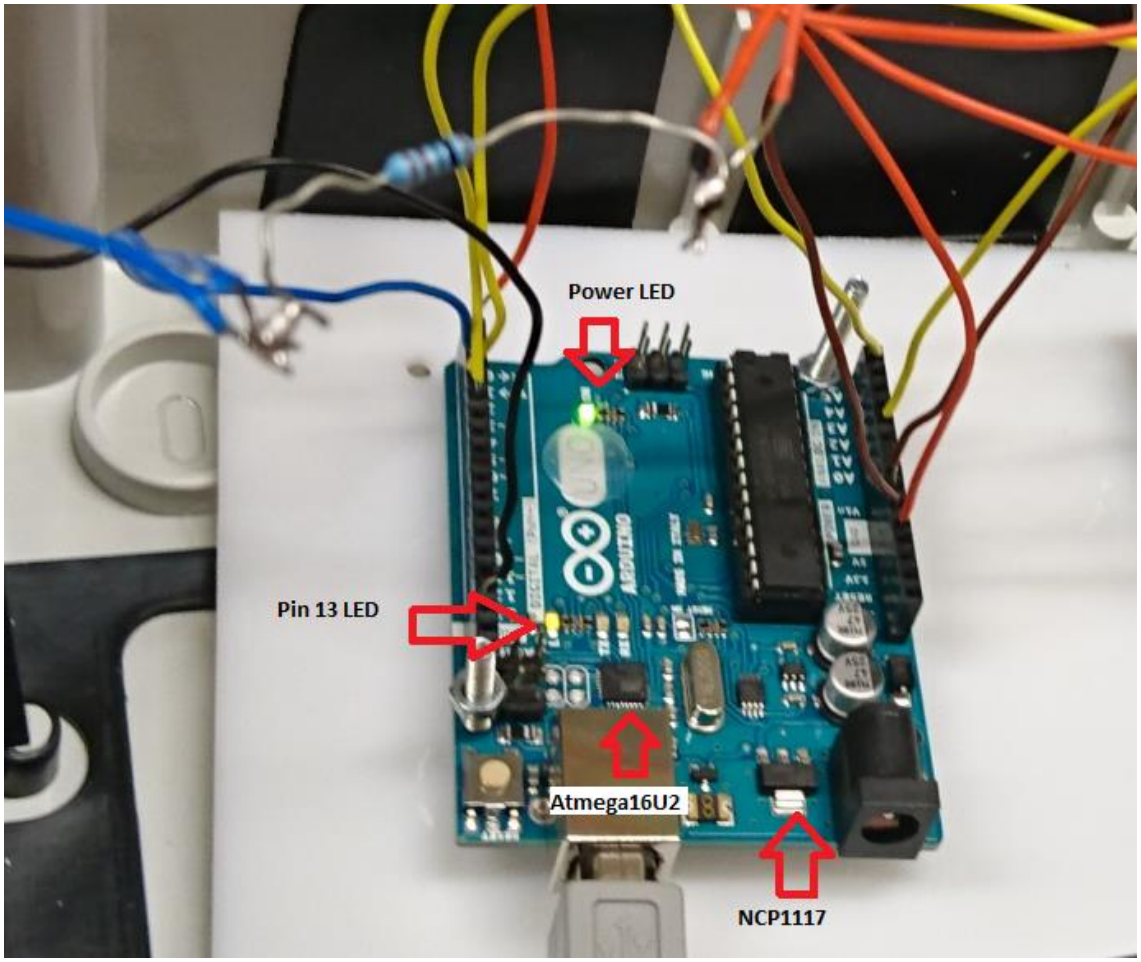
In the last three sleep modes, they all disable main processor and many components in the Arduino. The power-save mode is possible to wake the CPU up from timer interrupt. For instance, the Arduino wakes up to do some tasks and goes back to power-save mode after finishing all the tasks. The standby mode is as same as the power-save mode with the exception that the oscillator is kept running. Moreover, the CPU wakes up faster in six cycles clocks. The power-down mode is the lowest energy consumption mode which only wakes up from external interrupt [10].

Instead of using Atmega16U2 to connect between computer USB port and Atmega328P, the Atmel-ICE debugger was replaced to upload the code to the Arduino. Therefore, the Atmega16U2 was set to the power-down mode. Besides that, the LEDs of pin 13 (Picture 16) was turned off to save about 7mA. Furthermore, the main processor Atmega328P was implemented in the power-down to set the device in deep sleep mode after it sends the data to Sigfox backend.

4.2 Hardware solution

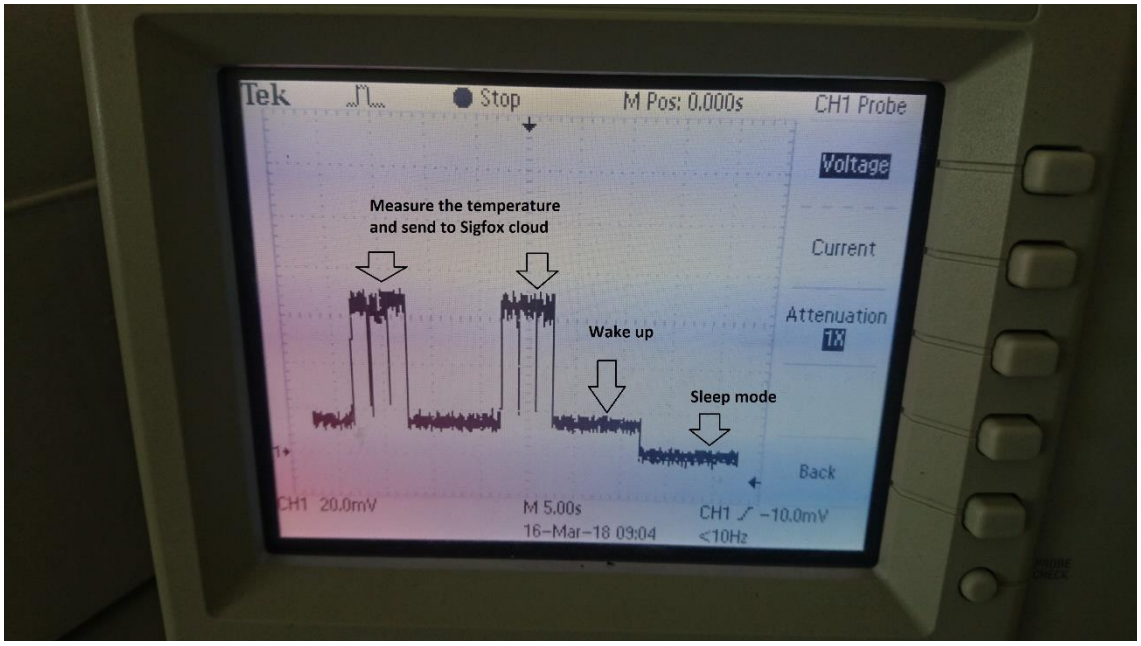
The power LED (Picture 16) uses approximately 7mA, it was unsoldered because it is connected directly to power source, thus, the main processor Atmega328P does not control it.

Moreover, the voltage regulator NCP1117 (Picture 16) which was removed to reduce the current flowed through it.



Picture 16. Position of LEDs, Atmega16U2 and NCP1117 on the Arduino.

4.3 Result of software and hardware solution

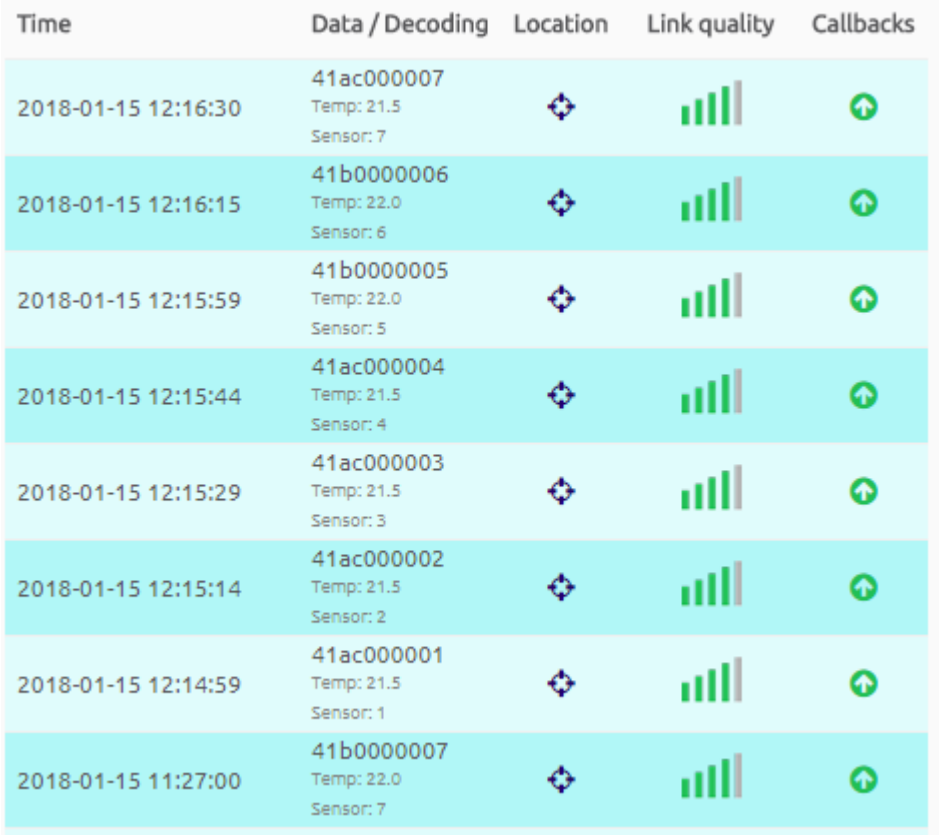


Picture 17. The result of sleep mode.

After implementing the sleep mode and removing some minor components of the Arduino Uno, the current decreases from 100mA to 70mA when measures and sends temperature data (Picture 17) to Sigfox cloud. Moreover, the most important part is the consumption while the device in deep sleep mode, which uses only 0.5mA (Picture 17). Therefore, the temperature data collection device can work approximately 24,000 hours continuously with three D batteries is connected in series.

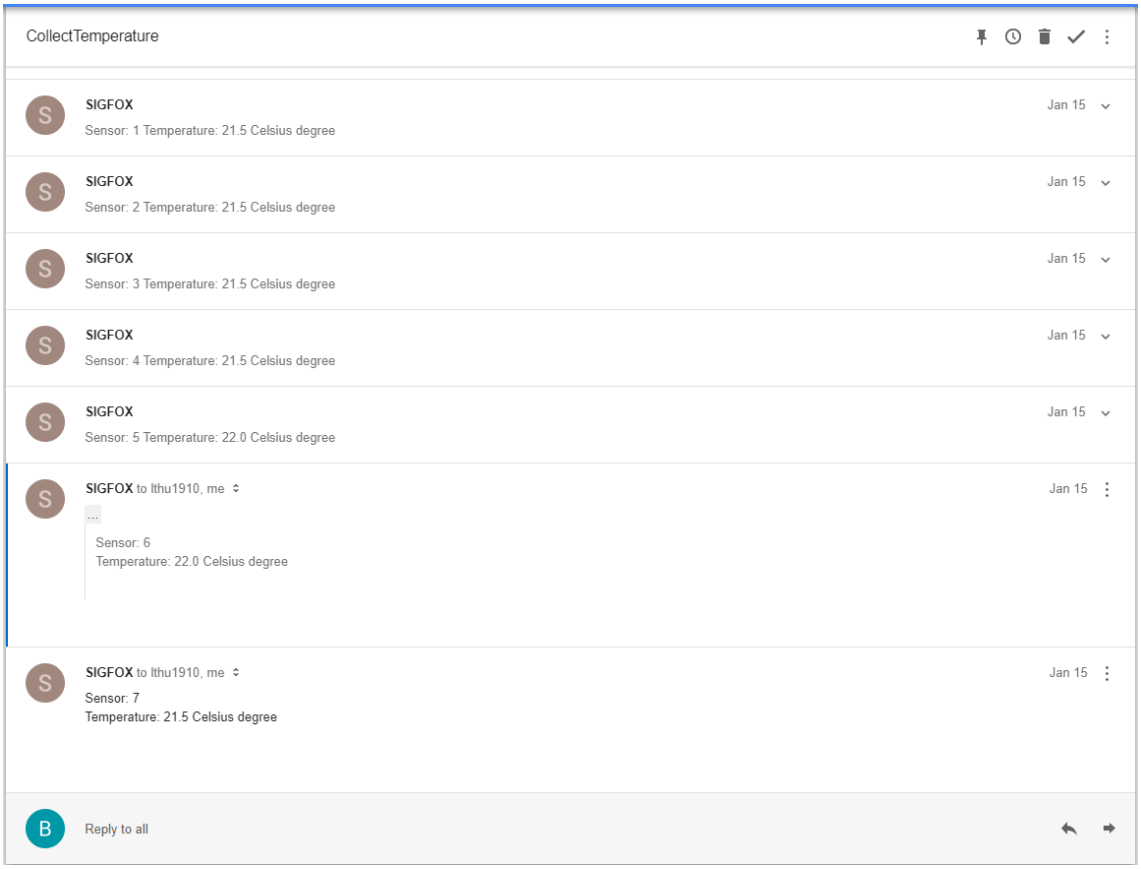
5 TESTING AND RESULTS

The unit test was applied for this project which means before implement components or the source code to the Arduino, the test case was created for each part. For instance, test case Sigfox ID which was embedded the code to check the Sigfox module operated well. After ensuring all components were processing well, the final code was implemented, the Sigfox backend was interfaced (Picture 18), the email (Picture 19) was ready and tested together. After the test, the device worked properly without any error.



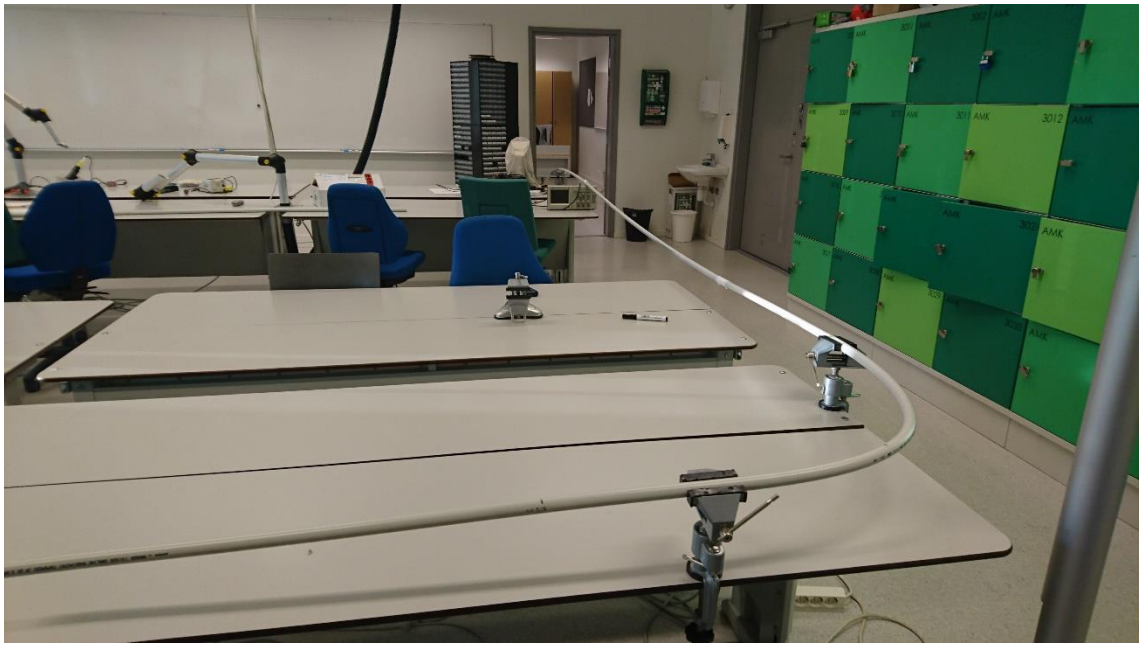
Time	Data / Decoding	Location	Link quality	Callbacks
2018-01-15 12:16:30	41ac000007 Temp: 21.5 Sensor: 7			
2018-01-15 12:16:15	41b0000006 Temp: 22.0 Sensor: 6			
2018-01-15 12:15:59	41b0000005 Temp: 22.0 Sensor: 5			
2018-01-15 12:15:44	41ac000004 Temp: 21.5 Sensor: 4			
2018-01-15 12:15:29	41ac000003 Temp: 21.5 Sensor: 3			
2018-01-15 12:15:14	41ac000002 Temp: 21.5 Sensor: 2			
2018-01-15 12:14:59	41ac000001 Temp: 21.5 Sensor: 1			
2018-01-15 11:27:00	41b0000007 Temp: 22.0 Sensor: 7			

Picture 18. The data portrayed on Sigfox backend website.



Picture 19. The temperature data on google mail.

After all the components operated correctly, the installation for the Dallas temperature sensors cable was tested. First, there are four PVC pipes were connected and bent 90 degrees in the middle of the last pipe (Picture 20). Then, the cable with the plastic cord were easily pushed into the pipe. Nevertheless, the cable without the cord was failed the test because it was stuck after went through the first pipe.



Picture 20. Testing the installation with 4 PVC pipes.

6 CONCLUSION

The temperature data collection device provided the main function of measuring and sending the temperature data from underground to the customer. With this data, the geothermal is easier to be implemented into the building in future. Furthermore, the device was low cost and low energy consumption so that it is suitable for all market. For instance, in Africa or in developing countries.

However, there are still limits in testing this device in real construction. Furthermore, the testing was not possible to cover all the condition. For example, in Winter, how the durability of the device in both inside and outside is. So, the feedbacks from customers in the future are very useful to improve and to develop this device further.

REFERENCES

[1] Geothermal heat pump [online] Available at https://en.wikipedia.org/wiki/Geothermal_heat_pump [Accessed 7th February 2018].

[2] Benefits [online] Available at https://www.geoexchange.com.au/technology/benefits/ [Accessed 7th February 2018].

[3] ARDUINO UNO REV3 [online] Available at https://store.arduino.cc/arduino-uno-rev3 [Accessed 8th February 2018].

[4] Raspberry Pi to Arduino Shields Connection Bridge [online] Available at https://www.cooking-hacks.com/raspberry-pi-to-arduino-shield-connection-bridge [Accessed 9th February 2018].

[5] Sigfox Technology Overview [online] Available at https://www.Sigfox.com/en/Sigfox-iot-technology-overview [Accessed 9th February 2018].

[6] Sigfox Shield for Raspberry Pi - EU [XBee Socket] [online] Available at https://www.cooking-hacks.com/shop/wireless/Sigfox-shield-for-raspberry-pi-868-mhz-xbee-socket [Accessed 9th February 2018].

[7] Sigfox Ready Technology for Arduino, Waspmote and Raspberry Pi [online] Available at https://www.cooking-hacks.com/documentation/tutorials/sigfox-connectivity-arduino-raspberry-pi-868mhz-europe-900mhz-us/ [Accessed 12th February 2018].

[8] D battery [online] Available at https://en.wikipedia.org/wiki/D_battery [Accessed 13th March 2018].

[9] Updating the Atmega8U2 and 16U2 on an Uno or Mega2560 using DFU [online] Available at https://www.arduino.cc/en/Hacking/DFUProgramming8U2 [Accessed 28th February 2018].

[10] Arduino Power Saving [online] Available at https://www.peterbeard.co/blog/post/arduino-power-saving/ [Accessed 19th March 2018].