Electronics and Computer Science
Faculty of Engineering and Physical Sciences
University of Southampton

Travis Han Yuen Lam

7 December 2020

LoRa based Weather Interpreter IoT Network

Project supervisor: Dr. Nicholas Harris

A project progress report submitted for the award of
MEng in Electrical & Electronics Engineering

Abstract

With current fast changing and advanced development of IoT protocol and platform, the implementation of IoT in each field became very predominant. To demonstrate the use case with one of the existing wireless communication technologies LoRaWAN, this project intends to implement a full integration a LoRa-based IoT network for a weather station, by that software are built on this and it will be optimized during first semester. During second semester, hardware interfacing circuit and power supply will be built on this. The structure of the project consists of two LoRa nodes connected to a LoRa-gateway/Raspberry Pi which upload the data to The Things Network and Microsoft Azure IoT hub to monitor the environment condition indoor and outdoor. The nodes are built to sense the temperature and humidity, also the sunlight. The gateway and end devices all built based on its LoRaWAN specification. Besides, this project also demonstrates the approach of data encapsulation as well as method of data transmission such as using different spreading factor and channels. The energy consumption is also an affecting factor to the device. In order to make an outdoor device suitable to for outdoor placement, a solar charger module will be built on this.

Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

You must change the statements in the boxes if you do not agree with them.

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

I have acknowledged all sources, and identified any content taken from elsewhere.

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

I have not used any resources produced by anyone else.

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

I did all the work myself, or with my allocated group, and have not helped anyone else.

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

The material in the report is genuine, and I have included all my data/code/designs.

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

I have not submitted any part of this work for another assessment.

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

My work did not involve human participants, their cells or data, or animals.

ECS Statement of Originality Template, updated August 2018, Alex Weddell aiofficer@ecs.soton.ac.uk

Contents

1 Project Goals

The objective of this project is to explore the existing communication technology LoRaWAN and create a LoRa based IoT system. It will start from building up the software by making a working end device node connects to the gateway, then improve optimize the latency also the energy efficiency of the device. The main goal for software implementation of the IoT network is that the device is able to communicate with the gateway without any problem and the data need to be shown on TTN at regular interval, it is best to upload the data to Microsoft Azure IoT hub so that the weather condition can be display on a graph and basic statistic can also be implemented on this. In second semester, an interfacing circuit will be built together with Power Supply Unit. The goal of making a good PSU is that the battery last long enough so that it does not need to replace regularly. By doing so, the power consumption needs to be estimated, A solar charger module also will be built on the outdoor end device node. During full implementation, the Raspberry Pi should act as a gateway powered by its AC powered charger, while the indoor and outdoor end devices should be Arduino stack up with a LoRa transmitter powered up by a rechargeable battery, with outdoor device able to charge by solar panel. The Gantt Chart in page X provides be the best estimated timeline, this project can only be considered as success when progression meets estimation.

2 Background and report of literature search

There are existing wireless communication technologies these days such as WiFi, Bluetooh 4.0 and ZigBee where all of them has a well- establish Standard documentation. LoRaWAN known as Long-Range Wide Area Network or Low-Power, Wide Area Network (LPWAN) it is network architecture in which each device connects to the gateway and the gateway relays all the message to a central network. LoRa refers to the physical layer of a radio wave modulation, it enables long range communication link up to scale of kilometre. It based on chirp spread spectrum modulation, which is an efficient modulation technique to achieve low power consumption [1]. Chirp spread spectrum modulation encodes the data symbol in a chirp, a sinusoidal wave whose has frequency sweep linearly within the bandwidth. Chirp signal sweeps frequency linearly starting at certain value, when it reaches the highest value , $f_c + BW/2$, then start from the lowest frequency $f_c - BW/2$, until the end of the chirp signal, with one symbol contains the spread factor(SF) bits. With different SF, LoRa signals are not perfectly orthogonal to each other, so the SF need to be tuned to make the chirp signals orthogonal to each other [3].

2.1. Network Architecture

Figure 1 shows the sample of LoRaWAN network architecture, in the figure there are various types of sensors known as end nodes which connects directly to each concentrator/gateway, the gateway relay the messages to the network server via TCP/IP SSL network protocol, the network server the provides services for the Application server.

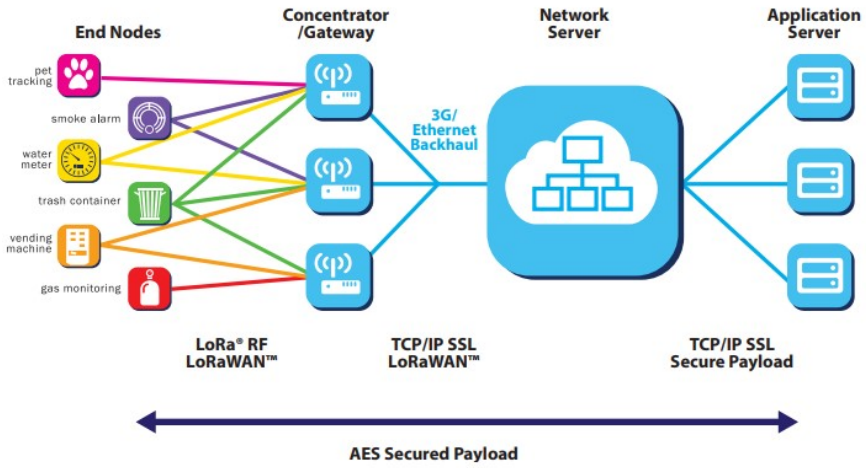

Figure 1. LoRaWAN Network Architecture [1]

2.2. Power consumption of LoRa

The nodes in LoRaWAN use asynchronous data communication and sends out data driven by event; this is also known as ALOHA method. To maximize a device battery life of an end device, the LoRa network infrastructure controls the data rate and RF output for each end node and transmit using any channel that available at any time [1].

2.3. Network Capacity

LoRaWAN can achieve high network capacity by adapting the data rate in different condition and by using multiple channel multi-modem transceiver in the gateway. The factors such as the number of concurrent channels, data rate, payload length, frequency of node transmit is affecting the capacity.

Based on Chirp spread spectrum modulation, the SF is tuned so that chirp signals having same orthogonality, speaking of change of SF the data rate also changed. This gives the benefit of having multiple data rate on one channel at the same time. If a node is close to gateway, there is unnecessary to keep the data signal strength so the data rate will be increased. This feature enables LoRaWAN to have a very high capacity and scalable network structure [1].

2.4. Device Class

Each device connected to the server may serves for different communication uses, for instance, there are devices save more energy but has higher latency such as a sensor and also device which required less communication latency but consumes more energy such as an actuator. Therefore, LoRaWAN classified different device classes as shown in Figure 2. In this project the device used is mainly A class end devices.

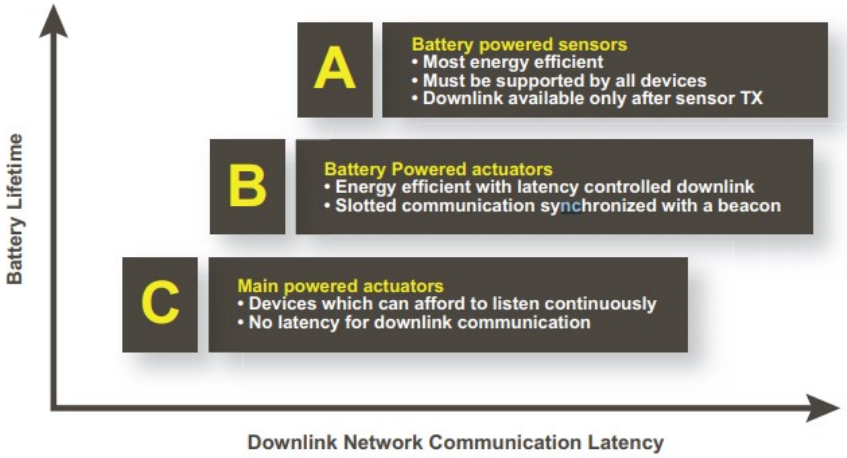

Figure 2. Device Classes and it uses [1]

2.5. Network Security

LoRaWAN provides two layers of security for network and application layer. LoRaWAN uses AES (Advanced Encryption Standard) encryption with the key exchange for IEEE EUI64 identifier. IEEE EUI64(64-bit Extended Unique Identifier) is an addressing format.

2.6. Physical Layer of LoRaWAN

2.6.1. Uplink Message

Uplink message is the data that sent by end node to the gateway and it relays to the server. Table 1 Shows structures of uplink message.

Preamble	PHDR	PHDR_CRC	PHYPayload	CRC

Table 1: Uplink PHY structure [2]

The uplink message starts with LoRa physical header (**PHDR**) plus a header CRC (**PHDR_CRC**), CRC stand for Cyclic Redundancy Check, an error detection technique. The Payload of the frame is the content of the message which also has its own CRC.

2.6.2. Downlink Message

Downlink message is sent by the network server to the end node relayed by gateway. Table 2 shows structure of downlink message.

Preamble	PHDR	PHDR_CRC	PHYPayload

Table 2: Downlink PHY structure [2]

The downlink message starts with LoRa physical header (PHDR) with header CRC (**PHDR_CRC**). However, the payload does not have CRC bits.

2.6.3. Regional Frequency

LoRaWAN frequency specification are different across the region. Table 3 shows the license frequency allowed to have a gateway across the region. In this project, LoRa radio module that able to transmit in 868MHz is chosen for sending uplink message to the LoRa gateway.

Europe	North America	China	Asia
867-869MHz	902-928MHz	470-510MHz	921-924MHz

Table 3: LoRaWAN Regional ISM band

2.7. Media Access Control (MAC) layer of LoRaWAN

2.7.1. MAC message

Physical Layer provides service to MAC layer, MAC layer is the sublayer between Network layer and Physical Layer. The MAC message is contained within PHY message the message format are shown in Figure 3. Further details of MAC message should refer to LoRaWAN specification [2].

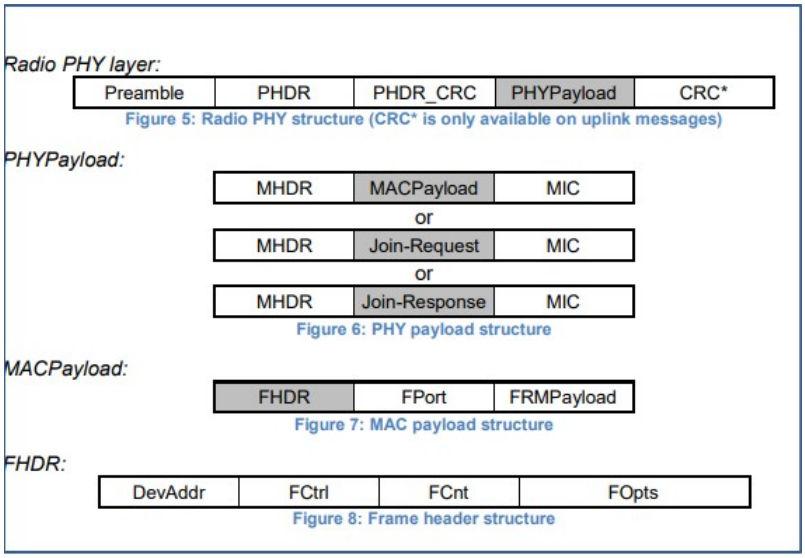

Figure 3. Format of MAC message [2]

2.8. Device Activation

In order to connect an end-device to the LoRa gateway, the device need to be activated. There are two ways to activate a device, either we reset the device via Over-The-Air Activation (OTAA), or Activation by Personalization (ABP) [2]. The following information is registered in the end device: Device Address (DevAddr), an application identifier (AppEUI), a network session key (NwkSKey) and application session key (AppSKey).

2.8.1. End-device address (DevAddr)

The DevAddr is an address to identify a device connected to the network. It consist of 32 bits where the most significant 7bits are used as network Identifier (NwkID) whereby the least significant 25 bits are used as network Address (NwkAddr). NwkID serves purpose of splitting addresses between different network operators overlapping network and utilize the roaming issues. NwkAddr is assigned by the network manager.

2.8.2. Application Identifier (App EUI)

The AppEUI is a global application ID in IEEE EUI64 address space, it is used to identify the status of JoinReq frame message in MAC message. AppEUI is stored in the end device before activation process.

2.8.3. Network session key (NwkSKey)

The NwkSKey is specific for device and used by both the network server and the end-device to ensure data integrity by compute and verify the MIC (message integrity code). It is also further implemented to decrypt and encrypt the payload field of MAC messages [2].

2.8.4. Application session key (AppSKey)

The AppSKey is specific to end-device and used by application server and end-device for en/decryption the payload of application data messages.

2.9. Reseacrh on Microsoft Azure

Microsoft Azure is a cloud computing platform aims to provide cloud computing service such as software as a service (SaaS), platform as a service (PaaS), also Infrastructure as a service. The focus of Azure of this project is using the Azure IoT hub. The Azure IoT hub offers users to monitor the sensor data in real time, also offers native support for MQTT 3.1.1 protocol[13].

3 Report on Technical Progress

3.1. Item Choice and budgeting

At the start of progression, the components are picked and planned to be used for the project. About the choice of component should refer to appendix A at page X. In the table, items such as Arduino Uno, Raspberry Pi 3, DHT11/22 are self-owned. In addition to reduce the cost, LoRa shield for Arduino and LoRa- Raspberry Pi HAT concentrator is used instead of buying the whole device.

3.2. Project Network Architecture

Figure 4. shows the network architecture of the project. The network topology of this weather interpreter system is a star centralised network, consisting two end devices connect to the centralised LoRa gateway. The gateway relays the data to internet server such as The Thing Network as well as Microsoft Azure IoT hub. So that it can be display on PC or mobile device.

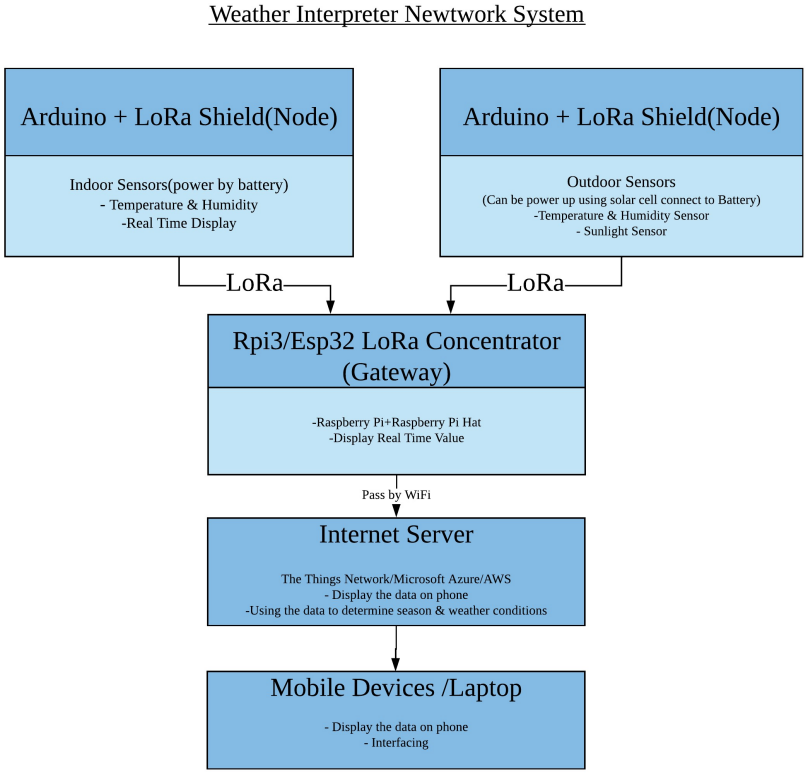

Figure 4. Network Architecture of weather interpreter network

3.3. Setting up a LoRA node

The end devices of a LoRaWAN are setup as shown in Figure 5. This is a simple build of each node, to make sure it can connect to the LoRa gateway and transmit data to The Thing Network. On the indoor and outdoor device, only DHT11/22 is connected to the LoRa shield stacked on the Arduino.

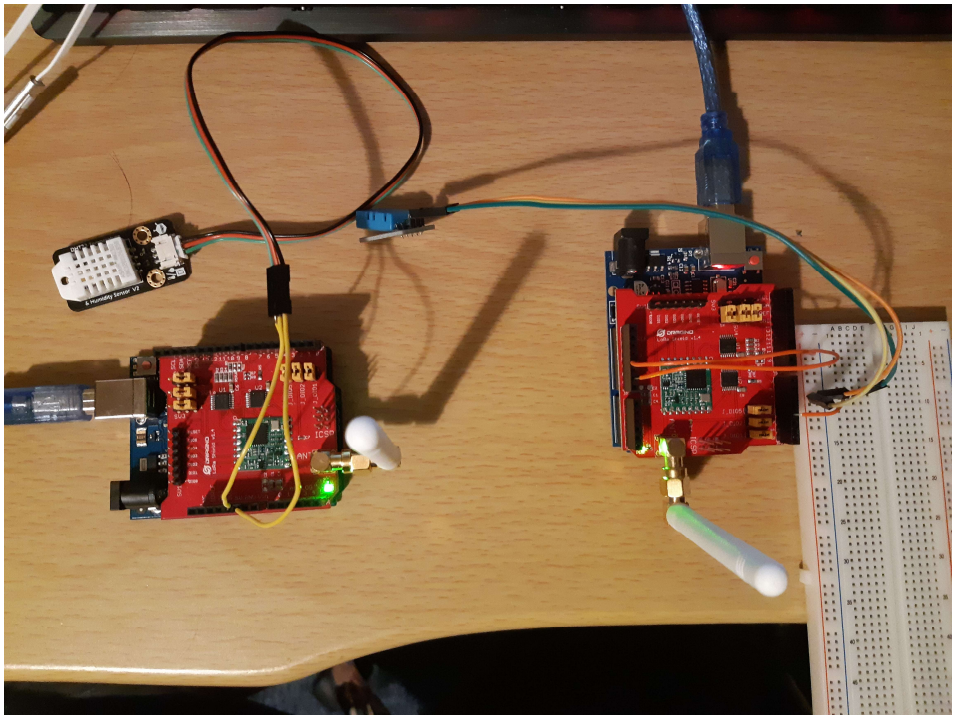

Figure 5. Hardware setup configuration of Indoor and outdoor end devices

The pin assignment of DHT11/22 and Dragino LoRa Shield are as shown in table 4 [11]. DHT sensor data pinout is connected to pin 3 it is not used by LoRa device [8][9]. During pin assignment for sensor data pin, the pinout should be checked so that it is not used by the transceiver.

Pin Name of respective device:	Pin on Arduino:
NSS (LoRa SPI Slave Select)	10
RXTX (LoRa Serial Comms)	LMIC_UNUSED_PIN
RST (LoRa reset)	9
Dio (LoRa digital I/O pin)	{2,6,7}
VCC (DHT11/22)	5V
Gnd (DHT11/22)	Gnd
Data (DHT11/22)	3

Table 4. Pin assignment between Arduino and its Components

The transceiver of LoRa module is based on RFM95W, it offers ultra-long-range spectrum spreads at low data rates, the library used to code the transceiver to is LMIC library created by IBM research lab. Github repository in [5] is referred to learn the basic idea to code up the end-device. The code written is shown in Appendix B. The sensor ID and the sensor value is encoded to hexadecimal and decoded using ASCII in application data. ABP is used as a method of device activation for the case of simplicity, as it will be replaced by OTAA as the project develop, it is because OTAA enable more data security over ABP.

Figure 6 shows activated end node, the device address is used to recognise the ID of the device, 0x26013923. Figure 7 shows the hexadecimal value transmitted to TTN by the device, and it is decoded by ASCII look up table, for example, payload: 48 3A 36 34 2E 30 30 2C 54 3A 32 31 2E 35 30 2C is decoded to "H:64.00,T:21.50" where 'H' represents humidity in percentage and 'T' represents temperature in degree Celsius.

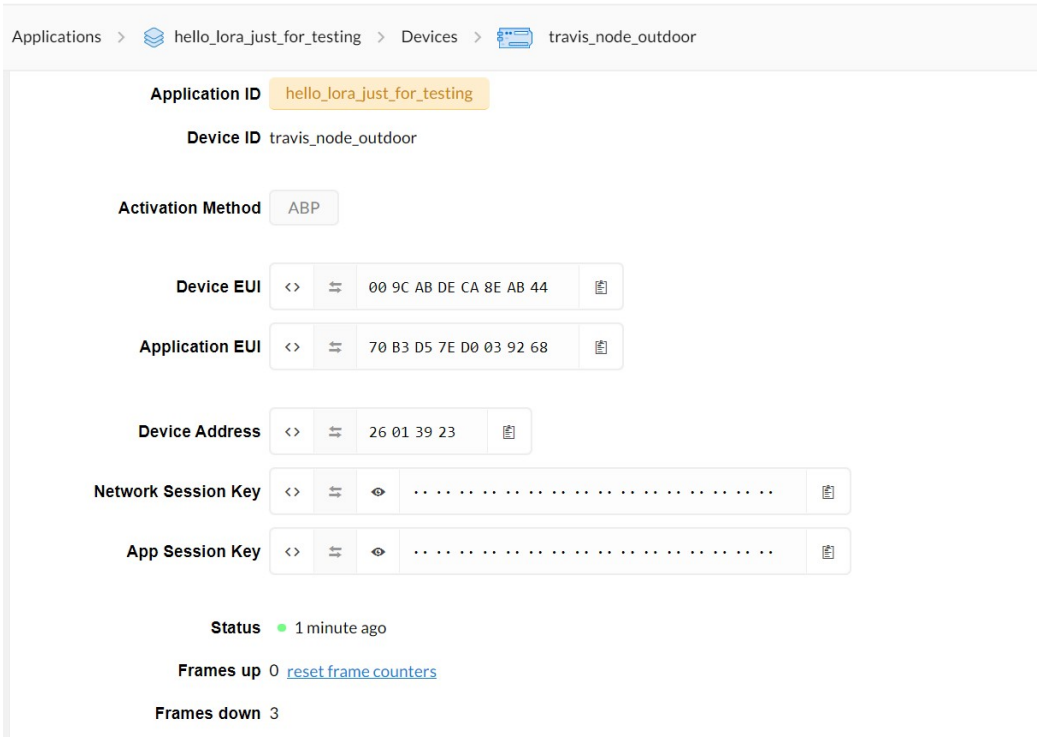

Figure 6. End-device registration [4]

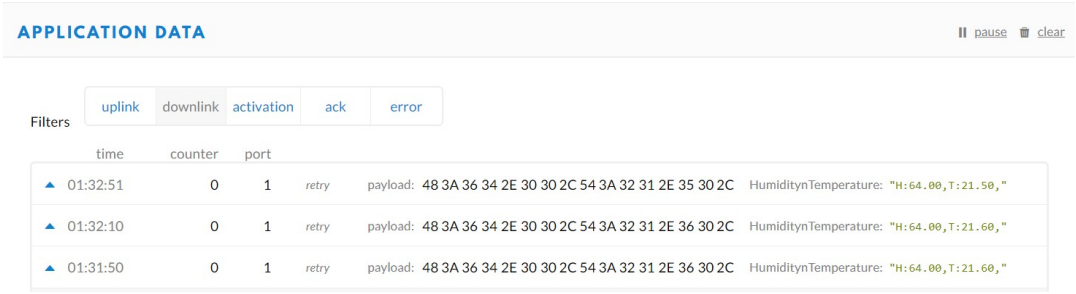

Figure 7. Application data for single node

3.4. Setting up a LoRa gateway

The physical setup of LoRa gateway can be shown on Figure 8, with LoRa RPi Hat stacked on Raspberry Pi 3B. Due to cost constraint the hardware used only can work in a single channel at a time, with fixed spreading factor. The LoRa RPi Hat is based on SX1276 transceiver, one of its features is it can achieve sensitivity at -148dBm with power amplifier of +20dBm [9]. The pin assignment between LoRa RPi HAT and Raspberry Pi shown on table 5 [10]. In current progression, the gateway only work until it able to receive signal at a single channel at fixed spreading factor, the GPS assignment is entered by manual so GPS module is not used.

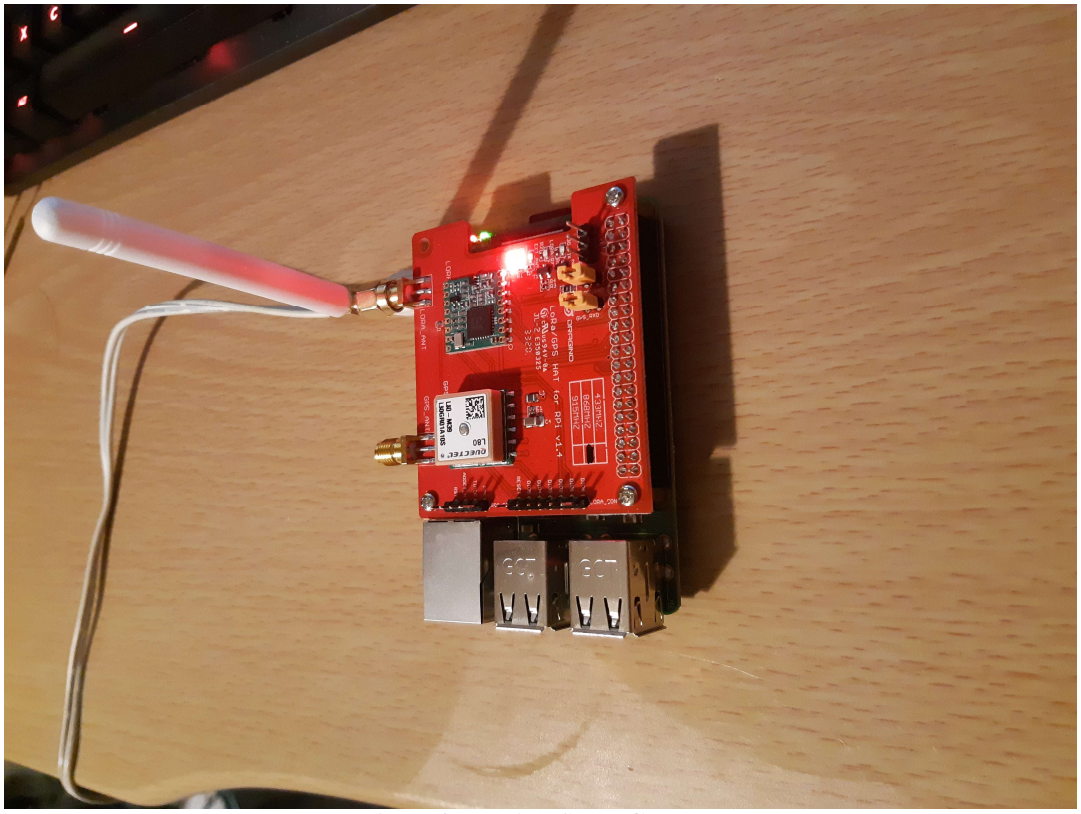

Figure 8. Physical Setup Gateway

LoRa RPi HAT	RaspberryPi Wiring Pi IO
3.3V	3.3V
5V	5V
GND	GND
DIO0	GPIO7
GPS_RX	GPIO15/TX
GPS_TX	GPIO16/RX
RESET	GPIO0
LoRa_NSS	GPIO6
LoRa_MISO	GPIO13/MISO
LoRa_MOSI	GPIO12/MOSI
SCK	GPIO14/SCLK
DIO1	GPIO4
DIO2	GPIO5
1PPS	GPIO1

Table 5. Pin assignment between LoRa RPi HAT and Raspberry Pi

To build up the gateway, the github library in reference [6] is clone to the Raspberry Pi. Besides, the SPI interface is enabled, and Wiring Pi is installed so that the HAT can communicate to Raspberry Pi. In the code "single_chan_pkt_fwd" in appendix B, the GPS is set by manual, the frequency(channel that the LoRa HAT listen to is set to 868.1MHz , Spread factor is set to SF7 so to listen to the end device.

The gateway is setup successfully with its Gataway ID, shown on Figure 9, while figure 10 demonstrates that it receive messages correct from the node created in section 3.2.

Figure 9. Single Channel Packet Forwarder LoRa Gateway

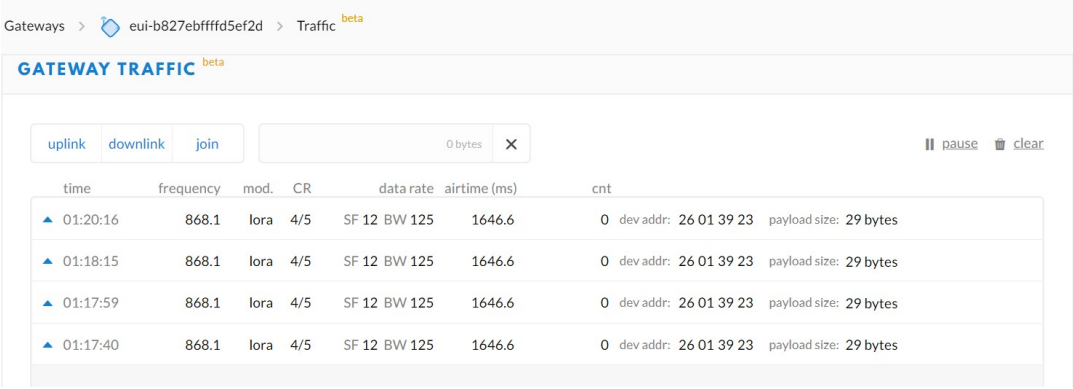

Figure 10. Gateway Traffic of the registered LoRa Gateway

4 Plan of remaining work

4.1. Optimize the Programme

The programme will be optimized by changing the activation method from ABP to OTAA method. At the same time, the power consumption needs to be minimized by turn off the all the LED that lights up during operation. The device transmits signal at 30second so during the 28seconds unused the device should put into sleep mode to save power consumption. The programme default run on spread factor 12 and bandwidth 125MHz, this makes the data rate slow but able to transmit the data to longer distance. For this project, the gateway is very near to the device around 100m, therefore there is no need to have a high SF in this situation. Therefore, need to look through the library in [5] and change the spreading factor to lower value.

4.2. Power Consumption Estimation

The power consumption of a single node needs to be estimated by looking up the datasheet of Arduino, LoRa shield, DHT sensor. If device have a scheduled sleep mode duration, the power consumption at the sleep duration should be consider.

4.3. PSU and interface circuit design

To make full implementation of the LoRaWAN system, the Power Supply Unit and interface circuit for Arduino will be designed. The PSU should be made such that it is able to power the Arduino for days before next recharge, the power consumption should as low as possible. The battery used can be LiPo battery charge by USB type B terminal port. Particular for outdoor device, a Solar panel should connect to the battery, such that it is able to charge the LiPo battery, when the LiPo battery is fully charged, the solar panel stops charging the battery. For interface circuit, the DHT sensor and the sunlight sensor should be embedded on board such that the data pin is connected Arduino digital IO and analogy pin. The interface circuit on the same board with PSU and this PCB should be stack on LoRa shield.

4.4. Full integration

When the PCB is soldered and assembled, it should work fine with the Arduino without circuit shortage. The final design should be 3D printed casing contained with the Arduino and its shield. The sensor data should be able to sends data to the gateway, and the dashboard or Microsoft Azure need to display the data correctly.

Appendix A: Gantt Chart

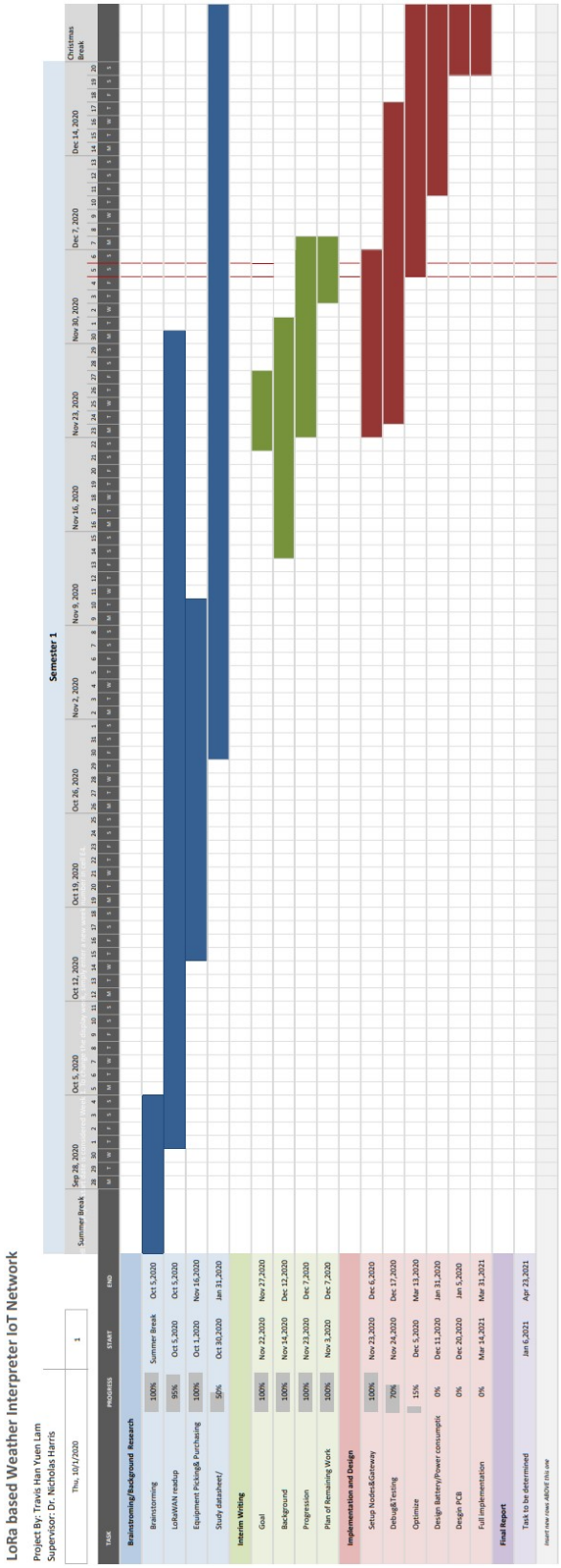

Figure 11. Gantt First semester

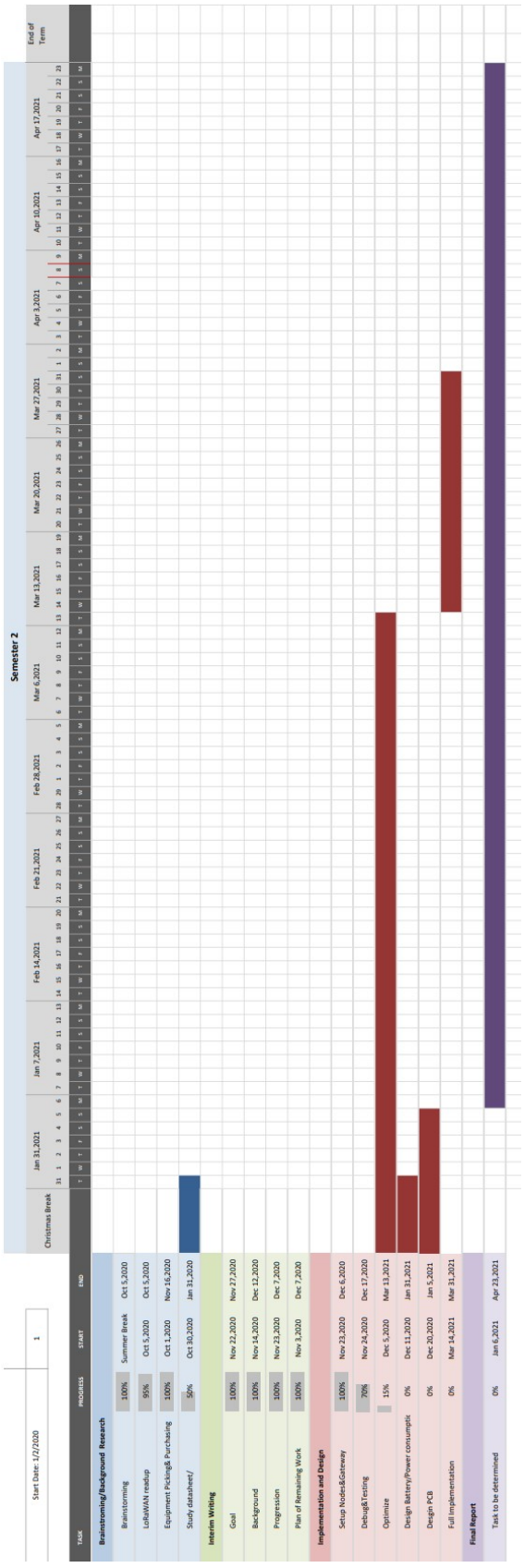

Figure 12 Gantt Chart Second Semester

Appendix B- Component Used and Costing

Component:	Unit Cost (£)	Quantity:	Total Price Inc. VAT (£):
868MHz Dragino LoRa Shield for Arduino	15.99	2	38.38
868MHz Lora/ GPS HAT for Raspberry Pi, 113990254	26.68	1	32.02
Add-On Board, Ambient Light Sensor Module, DFR0026	2.68	1	3.22
Delivery Charge	15.95		15.95
Arduino Uno R3	0	3	0
DHT 11	0	1	0
DHT 22	0	1	0
Raspberry Pi 3B	0	1	0
		Total:	89.57

Appendix C

(i) Code to Setup End devices

```
/*
 * Edited By: Travis Lam Han Yuen
 * Title: LoRa Based Weather Interpretor
 */
#include <lmic.h>
#include <hal/hal.h>
#include <DHT.h>
//#include <credentials.h>

#define DHTPIN 3
#define DHTTYPE DHT11 //using DHT22
DHT dht(DHTPIN,DHTTYPE);

#ifdef CREDENTIALS
static const u1_t NWKSKEY[16] = NWKSKEY1;
static const u1_t APPSKEY[16] = APPSKEY1;
static const u4_t DEVADDR = DEVADDR1;
#else
static const u1_t NWKSKEY[16] = { 0xF4, 0x41, 0x7C, 0x7C, 0x2F, 0x91, 0x59, 0xEB, 0x65,
static const u1_t APPSKEY[16] = { 0x5F, 0x75, 0x2F, 0x70, 0x34, 0xA1, 0x61, 0xDA, 0xF8,
static const u4_t DEVADDR = 0x26013923;
#endif

// These callbacks are only used in over-the-air activation, so they are
// left empty here (we cannot leave them out completely unless
// DISABLE_JOIN is set in config.h, otherwise the linker will complain).
void os_getArtEui (u1_t* buf) { }
void os_getDevEui (u1_t* buf) { }
void os_getDevKey (u1_t* buf) { }

// Sensor configuration
const char SensorIDTemperature[] = {"T"};
const char SensorIDHumidity[] = {"H"};

// Payload Configuration
const uint8_t spacebar = ':';
const uint8_t comma = ',';
const uint8_t max_message_len = 16;
uint8_t payload[max_message_len];
uint8_t payloadLength = 0;

static osjob_t sendjob;

// Schedule TX every this many seconds (might become longer due to duty
// cycle limitations).
const unsigned TX_INTERVAL = 20;

// Pin mapping Dragino Shield
const lmic_pinmap lmic_pins = {
```

```
41
42  static osjob_t sendjob;
43
44  // Schedule TX every this many seconds (might become longer due to duty
45  // cycle limitations).
46  const unsigned TX_INTERVAL = 20;
47
48  // Pin mapping Dragino Shield
49  const lmic_pinmap lmic_pins = {
50      .nss = 10,
51      .rxtx = LMIC_UNUSED_PIN,
52      .rst = 9,
53      .dio = {2, 6, 7},
54  };
55
56  void onEvent (ev_t ev) {
57      if (ev == EV_TXCOMPLETE) {
58          Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
59          // Schedule next transmission
60          os_setTimedCallback(&sendjob, os_getTime()+sec2osticks(TX_INTERVAL), do_send);
61      }
62  }
63
64  void PayloadAdd(const char *sensorID, float value, uint8_t decimalPlaces)
65  {
66      byte sensorIDLength = strlen( sensorID );
67      memcpy(&payload[payloadLength],sensorID,sensorIDLength);
68      payloadLength += sensorIDLength ;
69      payload[ payloadLength] = spacebar;
70      payloadLength += 1 ;
71      payloadLength += strlen( dtostrf(value, -1, decimalPlaces, (char *)&payload[payloadLength]));
72      payload[ payloadLength] = comma;
73      payloadLength += 1 ;
74  }
75  void PayloadReset()
76  {
77      for(int i=sizeof(payload);i>=0;i--) payload[i]='\0';
78      payloadLength = 0;
79  }
80  void do_send(osjob_t* j){
81      // Payload to send (uplink)
82      float h = dht.readHumidity();
83      float t = dht.readTemperature();
84      PayloadAdd( SensorIDHumidity , h , 2 );
85      PayloadAdd( SensorIDTemperature , t , 2 );
86       // Check if there is not a current TX/RX job running
87      if (LMIC.opmode & OP_TXRXPEND) {
```

```
void do_send(osjob_t* j){
    // Payload to send (uplink)
    float h = dht.readHumidity();
    float t = dht.readTemperature();
    PayloadAdd( SensorIDHumidity , h , 2 );
    PayloadAdd( SensorIDTemperature , t , 2 );
    // Check if there is not a current TX/RX job running
    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println(F("OP_TXRXPEND, not sending"));
    } else {
        // Prepare upstream data transmission at the next possible time.
        LMIC_setTxData2(1, payload, sizeof(payload), 0);
        Serial.println(F("Sending uplink packet..."));
        Serial.println(payload[0]);
        Serial.println(payload[1]);
        Serial.println(payload[2]);
        Serial.println(payload[3]);
        Serial.println(payload[4]);
        Serial.println(payload[5]);
        Serial.println(payload[6]);
        Serial.println(payload[7]);
        Serial.println(payload[8]);
        Serial.println(payload[9]);
        Serial.println(payload[10]);
    }
    PayloadReset();
    // Next TX is scheduled after TX_COMPLETE event.
}

void setup() {
    Serial.begin(115200);
    Serial.println(F("Starting..."));

    //  Setup DHT22
    dht.begin();
    delay(2000);

    os_init();

    // Reset the MAC state. Session and pending data transfers will be discarded.
    LMIC_reset();

    // Set static session parameters.
    LMIC_setSession (0x1, DEVADDR, NWKSKEY, APPSKEY);

    // Disable link check validation
    LMIC_setLinkCheckMode(0);

    // TTN uses SF9 for its RX2 window.
```

```
void setup() {
    Serial.begin(115200);
    Serial.println(F("Starting..."));

    //  Setup DHT22
    dht.begin();
    delay(2000);

    os_init();

    // Reset the MAC state. Session and pending data transfers will be discarded.
    LMIC_reset();

    // Set static session parameters.
    LMIC_setSession (0x1, DEVADDR, NWKSKEY, APPSKEY);

    // Disable link check validation
    LMIC_setLinkCheckMode(0);

    // TTN uses SF9 for its RX2 window.
    LMIC.dn2Dr = DR_SF9;

    // Set data rate and transmit power for uplink (note: txpow seems to be ignored by the library)
    LMIC_setDrTxpow(DR_SF12,14);

    // Start job
    do_send(&sendjob);
}

void loop() {
    os_runloop_once();
}
```

(ii) Code to configure Gateway

```
57  enum sf_t { SF7=7, SF8, SF9, SF10, SF11, SF12 };
58
59  /****************************************************************************
60   *
61   * Configure these values!
62   *
63   ****************************************************************************/
64
65  // SX1272 - Raspberry connections
66  int ssPin = 6;
67  int dio0  = 7;
68  int RST   = 0;
69
70  // Set spreading factor (SF7 - SF12)
71  sf_t sf = SF7;
72
73  // Set center frequency
74  uint32_t  freq = 868100000; // in Mhz! (868.1)
75
76  // Set location
77  float lat=0.0;
78  float lon=0.0;
79  int   alt=0;
80
81  /* Informal status fields */
82  static char platform[24]    = "Single Channel Gateway";  /* platform definition */
83  static char email[40]       = "";                        /* used for contact email */
84  static char description[64] = "";                        /* used for free form description */
85
86  // define servers
87  // TODO: use host names and dns
88  #define SERVER1 "54.72.145.119"    // The Things Network: croft.thethings.girovito.nl
89  //#define SERVER2 "192.168.1.10"      // local
90  #define PORT 1700                  // The port on which to send data
91
92  // #############################################
93  // #############################################
94
95  #define REG_FIFO                    0x00
96  #define REG_FIFO_ADDR_PTR           0x0D
97  #define REG_FIFO_TX_BASE_AD         0x0E
98  #define REG_FIFO_RX_BASE_AD         0x0F
99  #define REG_RX_NB_BYTES             0x13
100 #define REG_OPMODE                  0x01
101 #define REG_FIFO_RX_CURRENT_ADDR    0x10
102 #define REG_IRQ_FLAGS               0x12
103 #define REG_DIO_MAPPING_1           0x40
104 #define REG_DIO_MAPPING_2           0x41
```

References

[1] LoRa Alliance, " LoRaWAN What is it? " A technical overview of LoRa® and LoRaWAN™, November 2015, [Accessed 09/11/2020], Available at: https://lora-alliance.org/sites/default/files/2018-04/what-is-lorawan.pdf

[2] LoRa Alliance Technical Committee, "LoRaWAN™ 1.0.3 Specification", July 2018, [Accessed 09/11/2020], Available at: https://lora-alliance.org/sites/default/files/2018-07/lorawan1.0.3.pdf

[3] B. Dunlop, H. H. Nguyen, R. Barton and J. Henry, "Interference Analysis for LoRa Chirp Spread Spectrum Signals," 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), Edmonton, AB, Canada, 2019, pp. 1-5, doi: 10.1109/CCECE.2019.8861956.

[4] A. I. Petrariu, A. Lavric and E. Coca, "LoRaWAN Gateway: Design, Implementation and Testing in Real Environment," 2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME), Cluj-Napoca, Romania, 2019, pp. 49-53, doi: 10.1109/SIITME47687.2019.8990791.

[5] Matthijs Kooijman. "Arduino-LMIC library", 2015, [Accessed 23/11/2020], Available at: https://github.com/matthijskooijman/arduino-lmic#arduino-lmic-library

[6] Thomas Telkamp. "Single Channel LoRa Packet Forwarder" Dec 2015, [Accessed 6/12/2020], Available at: https://github.com/tftelkamp/single_chan_pkt_fwd

[7] Adafruit industry, "DHT sensor library" [Accessed: 25/11/2020], [Online]. Available at: https://github.com/adafruit/DHT-sensor-library

[8] Aosong Guangzhou Electronics Co. Ltd, " DHT11 product manual" [Accessed:25/11/2020], Available at: https://components101.com/sites/default/files/component_datasheet/DHT11-Temperature-Sensor.pdf

[9] Aosong(Guangzhou) Electronics Co.,Ltd, " DHT22 product manual" [Access:25/11/2020], Available at: https://cdn-shop.adafruit.com/datasheets/DHT22.pdf

[10] SEMTECH " SX1276/77/78/79 Low Power Long Range Transceiver" [Access:28/11/2020], Available at: https://semtech.my.salesforce.com/sfc/p/#E0000000JelG/a/2R0000001Rbr/6EfVZUorrpoKFfvaF_Fkpgp5kzjiNyiAbqcpqh9qSjE

[10] Dragino "LoRa GPS HAT Single Channel LoRa & GPS module User Manual "[Access:28/11/2020], Available at: http://www.dragino.com/downloads/downloads/LoRa-GPS-HAT/LoRa_GPS_HAT_UserManual_v1.0.pdf

[11] Atmel Corporation, "8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash" [Access:30/10/2020], Available at: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

[12] Z. Su, Y. Lin and V. R. L. Shen, "Intelligent Environmental Monitoring System based on LoRa Long Range Technology," 2019 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE), Yunlin, Taiwan, 2019, pp. 354-357, doi: 10.1109/ECICE47484.2019.8942778.

[13] Microsoft Azure, "Azure IoT Hub general availability overview" February 8,2016 [Access: 5/10/2020], Available at: https://azure.microsoft.com/en-us/blog/azure-iot-hub-ga-capability-overview/