

**DEVELOPMENT OF WIRELESS SENSOR NETWORK (WSN) FOR A
BUILDING TEMPERATURE MONITORING SYSTEM**

JIMMY HEE CHIN YONG

**A project report submitted in partial fulfilment of the
requirements for the award of Bachelor of Engineering
(Hons.) Electrical and Electronic Engineering**

**Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman**

September 2016

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : _____

Name : Jimmy Hee Chin Yong

ID No. : 1200994

Date : 31/1/2016

APPROVAL FOR SUBMISSION

I certify that this project report entitled **“DEVELOPMENT OF WIRELESS SENSOR NETWORK (WSN) FOR A BUILDING TEMPERATURE MONITORING SYSTEM”** was prepared by **JIMMY HEE CHIN YONG** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Hons.) Electrical and Electronic Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : _____

Supervisor : _____

Date : _____

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2016, JIMMY HEE CHIN YONG. All right reserved.

Specially dedicated to
My beloved family members and Mr Chua Kein Huat who have given to me so much
support and guidance in ensuring this project a success.

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Mr Chua Kein Huat for his invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, I would also like to express my gratitude to my loving parent and friends who had helped and given me encouragement. I am very fortune to be blessed with the guidance and encouragement from my mentor, Choy Jun Yean. He has inspired and motivated me to persue a self-learning attitude regardless the project.

Last but not least, I wish to express my special appreciation and thanks to Universiti Tunku Abdul Rahman (UTAR) which has provided me with a good platform and training ground facilities to pursue this project with confidence.

DEVELOPMENT OF WIRELESS SENSOR NETWORK (WSN) FOR A BUILDING TEMPERATURE MONITORING SYSTEM

ABSTRACT

In this digital era, computer technology continue to push the boundaries of what people once considered as impossible. Wireless sensor Network (WSN) has been invented to get device communicated without cable. WSN can operate in a wide range of environments and has advantages in cost and size. The aim of this project is to develop a WSN to monitor the temperature data in a building. The Zigbee Module S2 has been deployed as a transmitter and receiver. The transmitter is attached with temperature sensor, motion sensor and voltage sensor whereas the receiver is attached to the Arduino Mega. The Arduino Mega functions to interact with users and process the data. Apart from that, the Arduino Mega enables the data storage in SD card and also the cloud. The developed system is capable of measuring 2 nodes and the accuracy of data obtained is approximately 90% as compare with the actual result.

TABLE OF CONTENTS

DECLARATION	ii
APPROVAL FOR SUBMISSION	iii
ACKNOWLEDGEMENTS	vi
ABSTRACT	vii
TABLE OF CONTENTS	viii
LIST OF TABLES	xi
LIST OF FIGURES	xii
LIST OF SYMBOLS / ABBREVIATIONS	xv
LIST OF APPENDICES	xvi

CHAPTER

1	INTRODUCTION	1
	1.1 Background	1
	1.2 Aims and Objectives	2
	1.3 The Scope of project	2
2	LITERATURE REVIEW	3
	2.1 Related projects	3
	2.2 Existing Implementations of Zigbee on WSN	4
	2.3 Wireless Sensor Network (Zigbee)	8
	2.3.1 XBee as Embedded Gateway	8
	2.3.2 Study of Zigbee	8
	2.3.3 Zigbee device	9
	2.3.4 Network topology of Zigbee	9

2.4	The Internet	10
2.4.1	The Internet Terms	10
2.4.2	The Internet Media	11
2.4.3	The computer versus dedicated devices (iDigi)	13
2.4.4	Sharing Data throughout the platform	14
3	METHODOLOGY	15
3.1	Introduction	15
3.2	System Specifications	17
3.2.1	Network topologies	18
3.2.2	Hardware	18
3.2.3	Software	21
3.2.4	Cloud	21
3.3	Installation venue	22
3.4	Constructed hardware	24
4	SOFTWARE IMPLEMENTATION	27
4.1	Configuration of the software	27
4.2	Xively Website Setup	32
4.3	Flow chart of the Arduino programming	33
4.3.1	Real Time Clock (RTC) module	34
4.3.2	Ethernet module	36
4.3.3	SD Card module	38
4.3.4	Track Device Module	42
5	RESULT AND DISCUSSION	46
5.1	Preliminary Results from 24 th of July to 26 th of July	46
5.2	Results from 31 st of July to 2 nd of July	49
6	CONCLUSION AND RECOMMENDATIONS	52
6.1	Conclusion	52
6.2	Recommendations	53
6.2.1	Increase the number of sensor nodes	53

6.2.2	Change the network topology of the system	53
6.2.3	Increase the parameter to study	54
6.2.4	Provide output signal from the cloud	54

REFERENCES	55
-------------------	-----------

APPENDICES	57
-------------------	-----------

LIST OF TABLES

TABLE	TITLE	PAGE
2.2.1	Evaluation of the Performance & Method Used of the Project Done	7
2.4.1	Comparison between Ethernet, WiFi and Mobile data	13
2.4.2	Different of Both The Internet Gateway	14

LIST OF FIGURES

FIGURE	TITLE	PAGE
2.1	Proposed Architecture (Piyare et al, 2013)	4
2.2	Proposed Architecture (Kasapovic et al., 2015)	5
2.3	Proposed Architecture (Nikhade, 2015)(Kasapovic et al., 2015)	6
2.4	Inverse Square Law (Faludi,2010)	8
2.5	Variety Type of Network Topologies of Zigbee (Faludi,2010)	10
2.6	Type of Internet Media at Network Interface Card (Makki et al., 2003)	12
3.1	The Proposed Methodology of the System	15
3.2	Arduino Uno	18
3.3	XBee Series Module S2	19
3.4	Temperature sensor (LM335)	19
3.5	PIR Motion Sensor	20
3.6	Ethernet Shield	20
3.7	XBee Shield	21
3.8	Installation Floor Plan During 24 th of July Until 26 th of July	23
3.9	Installation Floor Plan During 31 st of July Until 2 nd of August	24
3.10	Router or End Device Assembled Circuit	25

3.11	Coordinator Assembled Circuit	26
4.1	XCTU Software Layout	27
4.2	XCTU Software Layout (Update Firmware)	28
4.3	Arduino IDE Software Layout	29
4.4	Arduino Real Time Monitoring Layout	30
4.5	Ethernet Error in Real Time Monitoring Layout	31
4.6	Data Logging Error in Real Time Monitoring Layout	31
4.7	Xively Platform Layout	32
4.8	Flow Chart of Main Program	33
4.9	Flow Chart of RTC Module	35
4.10	Coding for RTC Module	35
4.11	Flow Chart of Ethernet Module	36
4.12	Setup Coding of Ethernet Module	37
4.13	Uploading Code of Ethernet Module	38
4.14	Flow Chart of SD Card Module	39
4.15	Initialize Coding of SD Card Module	40
4.16	Coding for Data Storing in CSV File	41
4.17	Flow Chart of Tracking Device	43
4.18	Coding of Presenting the Data	44
4.19	Coding of Presenting the Data	45
5.1	Temperature Data Recorded from Website During 24 th of July	46
5.2	Temperature data of Router from SD card during 24 th of July	47

5.3	Temperature Data of End Device from SD Card During 24 th of July	47
5.4	Temperature Data of End Device during 24 th of July until 26 th of July	48
5.5	Temperature Data of Router During 24 th of July Until 26 th of July	49
5.6	Temperature Data of End Device During 31 st of July Until 2 nd of August	50
5.7	Temperature Data of Router During 31 st of July Until 2 nd of August	50
5.8	Temperature Data Recorded from Website During 1 st of August	51

LIST OF SYMBOLS / ABBREVIATIONS

API	Application Programming Interface
ADC	Analog Digital Converter
IOT	Internet of Things
LCD	Liquid Crystal Display
SRAM	Static Random Access Memory
UDP	User Datagram Protocol
VOIP	Voice Over Internet Protocol
WIFI	Wireless Fidelity
WSN	Wireless Sensor Network

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
A	Graphs plotted from SD card	57
B	Graphs obtained from Xively	64
C	Computer Programme Listing	68

CHAPTER 1

INTRODUCTION

1.1 Background

Energy generation, consumption and preservation is one concern issue in this century. (Skeledzija et al., 2014) This is because throughout the world, the emission of carbon dioxide has increased and this has brought out a lot of environment issue such as greenhouse effect. Therefore, it can either preserve through energy generation by using renewable energy or use the energy conservative method.

When energy is utilised wisely, the overall efficiency can be improved as it can reduce the energy wasted in the residences or industrial. To reduce the wasted energy, monitoring process should be taken in this place. As can read through several journal, technology of Wireless Sensor Networks (WSN) has brought the convenience to the public. Apart from it, WSN has been studied the efficiency of the energy usage has been improved and also the cost of communication has reduced. (Huang et al., 2016)

WSN is best in monitoring system currently. Monitoring system is taking a big important role as it has the ability to check and measured the parameter such as humidity, light intensity and temperature. Comparing the old time method where using the wired connection, the wire has to be used up a lot if the distance is very far whereas by using the WSN where there are communication between radio signal, the data can be bouncing after each other until the destination. Thus, it has reduce the cost of the length and also the power consumption needed.

Apart from it, WSN can be fully utilized the work and reduce the man-power. It can be sending out the control signal at the base station and bounce through the transmission device until an end device and perform the work instead of calling a worker to walk to the end device to perform the job.

1.2 Aims and Objectives

The aim of this project is to develop a Wireless Sensor Network (WSN) for a building temperature monitoring system. This system is planned to be installed in one of the level in Universiti Tunku Abdul Rahman (UTAR)

The objectives are stated as below:

1. To investigate the WSN for a building temperature system.
2. To develop a building temperature monitoring system.
3. To develop a real time monitoring system
4. To evaluate the performance of the building temperature monitoring system.

1.3 The Scope of project

In this project, the Zigbee Module S2 is implemented for the communication between two devices. At the end device or router, the Zigbee Module S2 will be working as the transmitter to transmit the temperature data, motion data and voltage data to the receiver at the coordinator. Arduino Mega is used as the coordinator to carry out all the processing data after receive the data from the transmitter. Then, this data is integrated and posts to the serial monitor in the human readable language. In the meantime, the data also posts to the cloud and stores in the SD card.

CHAPTER 2

LITERATURE REVIEW

2.1 Related projects

Wireless Sensor Network (WSN) has been participated in a lot of projects and study. WSN is used in monitoring the environmental conditions at home, cities or warehouse. In order to provide convenience for monitoring, WSN can be working in the real time monitoring. According to the (Piyare et al., 2013), they have presented a WSN system with the aids of web-based interface, which is customizable and allows anyone from anytime and anywhere to access the data. (Kasapovic et al., 2015) have integrated the Arduino with the Zigbee series S2 to monitor the building environmental monitoring system. Their work is integrated with the cloud system based which a lot the users able to access the data using smart phones or via internet (Nikhade, 2015), has collected the data of light intensity in a room by implementing XBee series S2 of zigbee protocol with the aid of raspberry Pi.

The details on how the authors are implementing their projects of wireless sensor networks will be discussed in Section 2.2.

2.2 Existing Implementations of Zigbee on WSN

2.2.1 Implementation of the WSN

Piyare has set up their data collection system with the XBee ZB for the end device and arduino UNO as a base station (Coordinator) (Piyare et al., 2013). Arduino Uno is using ATmega328P processor with the operating voltage of 5V and also 32kB flash memory (“Arduino - ArduinoBoardUno,” n.d.). Coordinator will collect data of all the end device and upload this data to sensor data platform in the cloud. They are using representational state transfer (REST) based web service on an IP based low power WSN test bed. The reasons they propose this idea is because this allow users access the data from anywhere for the smart environment with also low power consumption of electrical energy as the end device can go into the sleep mode.

Piyare has claimed that it has been split into 3 parts as showed below the figure: Sensor layer where the XBee ZB take part in receiving data, Coordinator layer where Arduino Uno collect the data from all the end device and upload the data through the ethernet port to the cloud based web service (Piyare et al., 2013). Lastly, the supervision layer, Open.sen.se HTTP Service which provides a REST based Application Programming Interface(API) to publish and access the sensor data. The Figure 2.1 has illustrated proposed architecture of Piyare(2013).

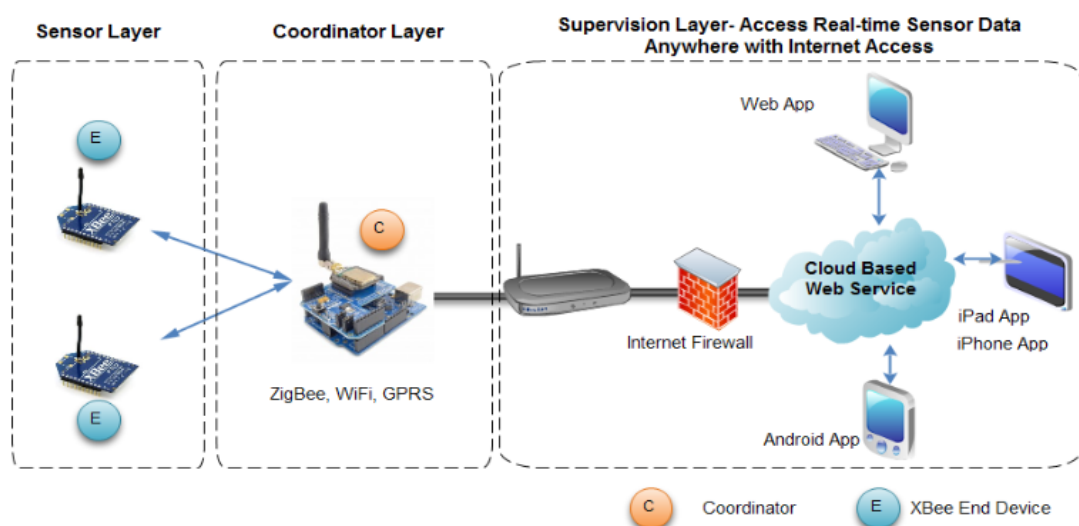


Figure 2.1 Proposed Architecture (Piyare et al, 2013)

On the other hand, Kasapovic et al. has designed a WSN home environment using ZigBee Protocol (Kasapovic et al., 2015). Kasapovic et al has stated that they are measuring the temperature in the smart home. Xbee Module series 2 with Zigbee communication protocol is used in the end device and also router. Apart from it, Kasapovic et al also used TMP36 precision temperature sensor with the sensitivity of $10\text{mV}/^{\circ}\text{C}$. Then, Arduino with microcontroller Atmega328 is used as the coordinator. The coordinator is the base station that will collect all data from other nodes. The communication of data between the coordinator and the end device is in the form of Application Programming Interface (API) packet. In kasapovic's paper, it also stated that Arduino with the Ethernet module and with the service of Xively platform where data is stored and the real time data can be showed in the server in the graphical form. From the paper of Simek et al. and Sinha et al. , they have stated that Xively is an internet of things(IoT) platform that allow end user connect to the server via a link activation method (Simek et al., 2013; Sinha et al., 2015). Sinha et al. also stated that Xively is able to track the exact location of the user that connected to the server (Sinha et al., 2015). The overall system of Kasapovic (2015) WSN has illustrated as Figure 2.2.

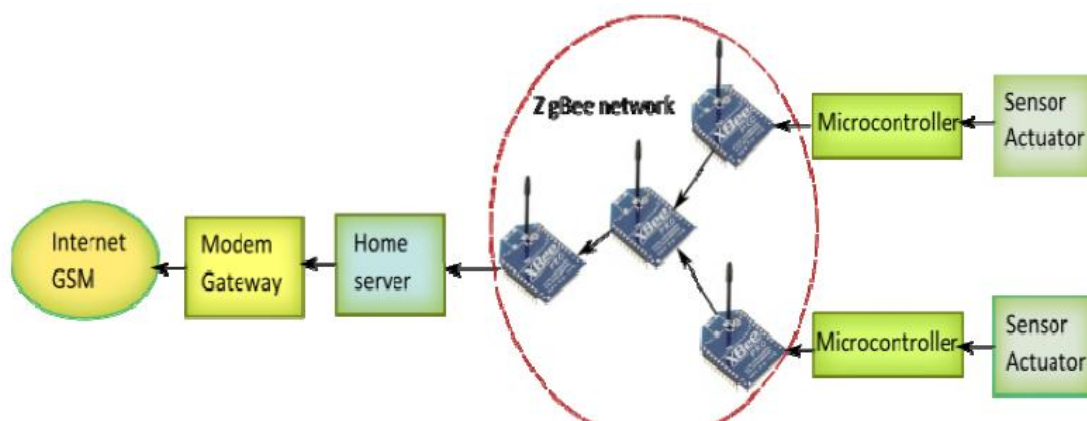


Figure 2.2 Proposed Architecture (Kasapovic et al., 2015)

From the (Nikhade, 2015) paper, he has implemented the way WSN with the aids of Raspberry Pi and XBee series S2 to collect the data of light intensity. According to Nikhade, raspberry pi is a low cost, low power single tiny computer where it is running with Linux operating system. He also stated that the board runs an ARM11 microcontroller @ 700MHz and come with a 512 Mega Bytes of RAM memory. So, he used the Raspberry Pi stacked with XBee series S2 (receiver) acts as

a base station which all the data will be collected from all the sensor node via the zigbee communication protocol. This data will store in the Mysql database form. Apart from it, he has used ATmega324PA which the chip has a 32KB flash program memory, 1KB EEPROM, 2KB internal SRAM, two 16 bit timer, programmable watchdog timer, 8 channel 10-bit ADC. When the data from the sensor node is collected, it will pass through the ATmega324PA microcontroller to process the data and will show the data at the LCD display and also send the data to the base station (Raspberry Pi) via Zigbee communication protocol. So, Figure 2.3 is the proposed architecture from Nikhade(2015).

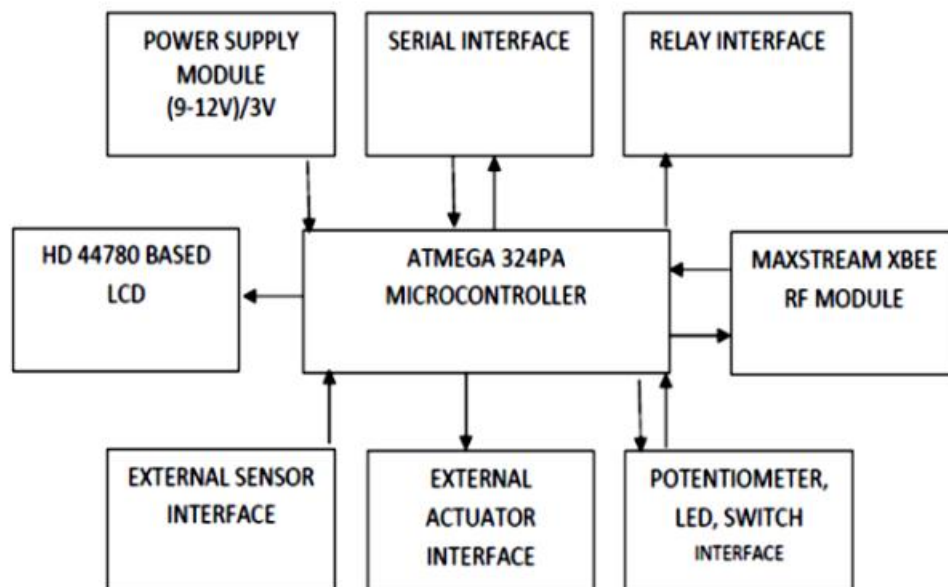


Figure 2.3 Proposed Architecture (Nikhade, 2015)(Kasapovic et al., 2015)

After revising several related projects with their implementation in the sub-section 2.2, evaluation of the performance and method used by all the authors has been done. This is to have clearer view of the pro and cons of the particular project. The evaluation of the performance and method used of the project are done in the form of table in the Table 2.2.1.

Table 2.2.1 : Evaluation of the Performance & Method Used of the Project Done

Authors	Piyare et al. (2013)	Kasapovic et al. (2015)	Nikhade (2015)
Parameters monitored	Unknown	Temperature	Light intensity
Sensor	Unknown	TMP36	Unknown
Implemented location	unknown	Smart home	Unknown
Microcontroller	Arduino	Arduino	Raspberry Pi & ATMEGA324PA
Radio module	XBee ZB	XBee Series module S2	XBee Series module S2
Data storage	Web page cloud service	Web page	Mysql database
Internet platform	Open.sen.se	Xively	Web browser/PHPMYAdmin
NOTES:			
<ol style="list-style-type: none"> 1. Piyare et al. (2013) – Integrating Wireless Sensor Network into Cloud Services for Real-time Data Collection 2. Kasapovic et al.(2015)- An Approach to Wireless Sensor Network Design in Home Environment using ZigBee Protocol 3. Nikhade (2015)- Wireless Sensor Network System using Raspberry Pi and Zigbee for Environmental Monitoring Applications 			

2.3 Wireless Sensor Network (Zigbee)

2.3.1 XBee as Embedded Gateway

There are various network protocols exist in this era. Zigbee, WiFi, Ethernet, Bluetooth are the well-known by public. IPv6,UDP and VoIP are the rarely known by the public. There are so many network protocols are because every network protocols have pro and cons. (Faludi,2010)

Zigbee is a protocol that able to creating radio sensor network. It allows to connect up to 65,000 nodes. With the number of the nodes, it can bring up to a big system. It has the low RF data rate at 250 kbps. Due to low data rate, the power consumption of the system is low too.

2.3.2 Study of Zigbee

In the paper of (Kasapovic et al., 2015; Nikhade, 2015; Piyare et al., 2013), they have monitored their wireless sensor network with the aids of wireless sensor network. From the (Faludi, 2010), the Zigbee is the low power communication protocol with the standard base on IEEE 802.15.4. It is designed with the inverse square law in mind. Inverse square law stated that when the distance is doubled, the power required is four time larger quantities needed. However, Zigbee is transferring the data to the nearest neighbour in the network thus this can be reduce the power consumption. Inverse square law can be discussed as Figure 2.4.

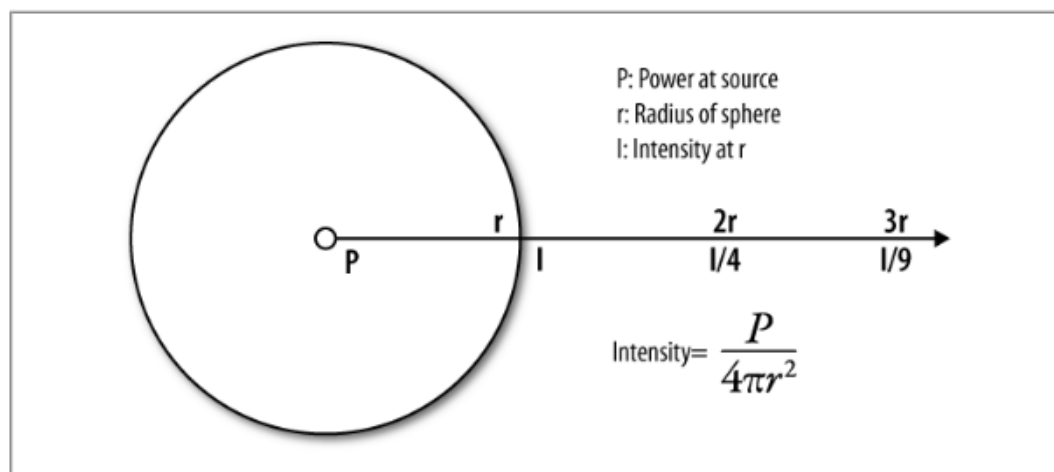


Figure 2.4 Inverse Square Law (Faludi,2010)

2.3.3 Zigbee device

In the network, there are three important roles Zigbee are playing which are Coordinator, router and end device (Faludi,2010). Every network, there must be only a coordinator device. It is acting as a parental device for all router and end device as it has to be collecting all the data from every end device and giving command. Without coordinator, the whole system won't be working. So it has to be turn on for all the time.

Then, router is a device to work between the end device and also coordinator. It bounces the signal from the end device to the coordinator or from the coordinator to the end device if the distance from the end device is too far away from the coordinator. It can be more than one in a system. Router has to be turned on for all the session.

Next, the end device is placed where the sensors are attached. It is the taking the role as to collect the data and send to the either to the router or directly to the coordinator if the distance is not too far or receive the command from parental device which are either coordinator or router. It can be a number of end device up to 65,000 nodes of them. The special things for end device that different from coordinator and router is, end device can enter to the sleep mode. Sleep mode is a power saver. It has allowed the device to be working on according to the pre-set time to gather the data then send the data to parental device and once it is done, it will run to the sleep mode again (temporary offline).

2.3.4 Network topology of Zigbee

In a system, there are several layouts or topologies of connection for Zigbee communication system. Basically, this topologies are indicating the arrangement of coordinator, router and also end device. The various type of topologies will discuss as below as what Faludi (2010) has stated.

- a. Pair Network: The most fundamental network where there are only two nodes, a coordinator and an end device.

- b. Star Network: The next level of pair network. The system can be a coordinator and many of end devices connect to the coordinator radially. In the rule that coordinator must in the centre and also end device must directly transmit the data to the coordinator.
- c. Mesh Network: In the mesh network, it is the same as the star network but the different is the end device are allowed to pass the data to the router and router will be working as bouncing the signal from the end device to the coordinator again.
- d. Cluster tree network: it can be the most complicated network if it is formed. It is basically slightly the same as the mesh network.

The system is illustrated in the Figure 2.4.2.

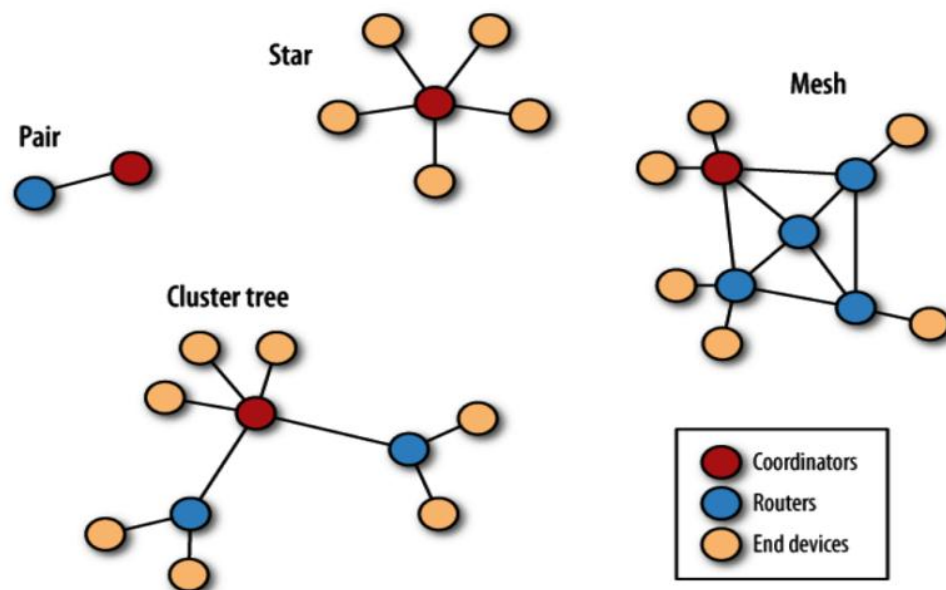


Figure 2.5 Variety Type of Network Topologies of Zigbee (Faludi,2010)

2.4 The Internet

2.4.1 The Internet Terms

Internet able to reaches almost everywhere and has the abilities to connect everything. There are a few fundamental terms that need to understand in this project. From the

book of building wireless sensor networks, Faludi (2010) has discussed in three basic terms:

Data storage – Sensors that connect to device must have reading and those reading will need a storage to be stored. So internet is the great things that able to store all this reading in everywhere. Data that store in the internet can be private such MySQL database or hosting provided such as Dreamhost or Host Pachube's system. With the available of hosting system, it allows other end user to access the data anywhere and anytime.

Data presentation – Once there are data recorded in the database, presentation have to be take note in the next step. As it allow end user to have better visual or view of the overall data in the internet. The data can be present in the table form or chart. There are several good service such as Google Chart which able to draw chart and also Pachube sites which able to share data, in the same time have visualization services.

Remote actuation- This is the platform where it is able to give command to the device to perform actions. For example, when the temperature is higher, from the remote site, the system can lower down the temperature of air conditioner and increase the fan speed. Thus the system will be equilibrium. This has ease up the user who are far away from the device and still able to monitor and control the working area.

2.4.2 The Internet Media

Internet media is the utilities that use to connect the whole system to the server. According to the Faludi (2010), three well known choice of physical networks by public are Ethernet, WiFi and Mobile data. The detail of it will discuss as below.

Ethernet – It is the first generation media with the standard as the IEEE802.3. that people connect the device to the server. It is wired type where the gateway will chain

at the router. It is relatively inexpensive, reliable and reasonably fast at the speed up to 10Mbps. Then, configuration is not essential all the time.

WiFi (Wireless Fidelity) – It is the second generation media with the standard IEEE802.11. that people connect the device to the server. It provides wireless connection but the connection distance is limited. Then, it is expensive due to the physical device and reliability of the WiFi is not very good as noise is found in the network system. The speed of data rate can up to 11Mbps(Lehr and McKnight, 2003). Lastly, the configuration of the system is compulsory.

Mobile data (2G/3G/4G) – It is the latest media compare to the other two media. It able to connect in most of the location where sometime it able to reach the area where WiFi and Ethernet cannot reach. Thus it makes the overall system reliable. On the other hand, the cost for the device of the mobile data is the roughly the same as WiFi but the connection itself is far more expensive. The speed of the data depends on the data plan itself.

So all the internet media explained above are the network connect method that connect the system at the network interface as illustrated in Figure 2.5.1.

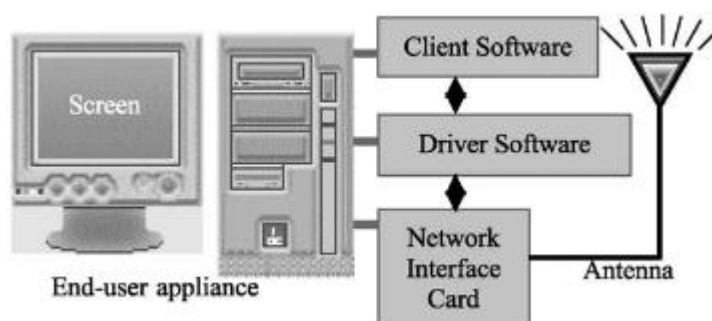


Figure 2.6 Type of Internet Media at Network Interface Card (Makki et al., 2003)

After all the studied of internet media, the comparison of the Ethernet, WiFi and mobile data will be showed in the Table 2.4.1

Table 2.4.1 : Comparison between Ethernet, WiFi and Mobile data

Media Specification	Ethernet	WiFi	Mobile Data (2G/3G/4G)
Configuration	Not necessary	Compulsory	Compulsory
Reliability	Reliable	Less reliable	Reliable
Cost	Cheapest	Expensive	Most expensive
Mobility	Static	Dynamic but limited distance	Dynamic as long as availability of data and service

2.4.3 The computer versus dedicated devices (iDigi)

In the book of ‘building wireless sensor networking’ by Faludi (2010), he has stated there are two types of internet gateways which are personal computer or manufactured as dedicated devices.

Computer – It is a good choice for gateways in quick prototype systems. It needed to connect to both Ethernet and WiFi for accessing to the Internet. It is also safe that it also has a USB serial port to plug into microcontroller. Besides, it is free as personal computer is used. On the other hand, long term work is not suitable if choosing computer as it run complex operating system. Other than that, personal computer also running other program thus it made the system to run slower. Then, it also consumes higher electricity compare to the dedicated devices from iDigi. Lastly, the mobility of the computer is static. It couldn’t shift to other places easily.

Dedicated devices (iDigi) – It is a Zigbee gateways there’s a radio to communication with the WSN. It is often to come with a microprocessor, a Ethernet, WiFi or mobile data module to navigate the system to the cloud. This device is needed to be configure but it doesn’t have to be a complex configuration as it is well done in a

web browser. The bright side of the device is, it tends to be small and able to move whenever it can. It costs will be cheaper than a desktop. Since it doesn't run a lot of program in a same time, it consumes less power in overall. So table 2.4.2 will illustrate the different of the both the internet gateway

Table 2.4.2 : Different of Both The Internet Gateway

Computer		Dedicated Devices	
Complex	System design	Simple	
More	Power consumption	Less	
Free (if own one)	Price	Expensive	
More	Maintenance	Less	

2.4.4 Sharing Data throughout the platform

There are plenty platform that can be found in the website. So through the books and journals, Faludi (2010) has mentioned about Pachube and also from (Simek et al., 2013; Sinha et al., 2015) has mentioned about Xively. The benefit of sharing this data in the platform is the data can be accessible whenever and wherever the users want.

Pachube – It is a platform that allow the public to upload, download and display of data for Internet-connected sensor networks. Data, information of the location, energy used, and results of environmental parameter can be stored and shown from anywhere or anytime in the world. This is the free platform that allow users to access once ID account is registered. However, this software cannot import a lot data in the same time if using the free version.

Xively – It is also a platform as a facility which allow the users to access the platform then to read the data such as environmental parameters, information of the location. This is also the free platform that allows users to access once ID account is registered. Privacy and extended features are available with paid accounts.

CHAPTER 3

METHODOLOGY

3.1 Introduction

The conceptual idea of the about this project is proposed as showed in the Figure 3.1. The data of the parameters of room temperature, closure status of the door and room occupancy able to monitor from the computer with the aids of wireless sensor network. Apart from it, the computer at the base station able to send signal either on or off to the air-conditioning system at the remote site through wireless sensor network.

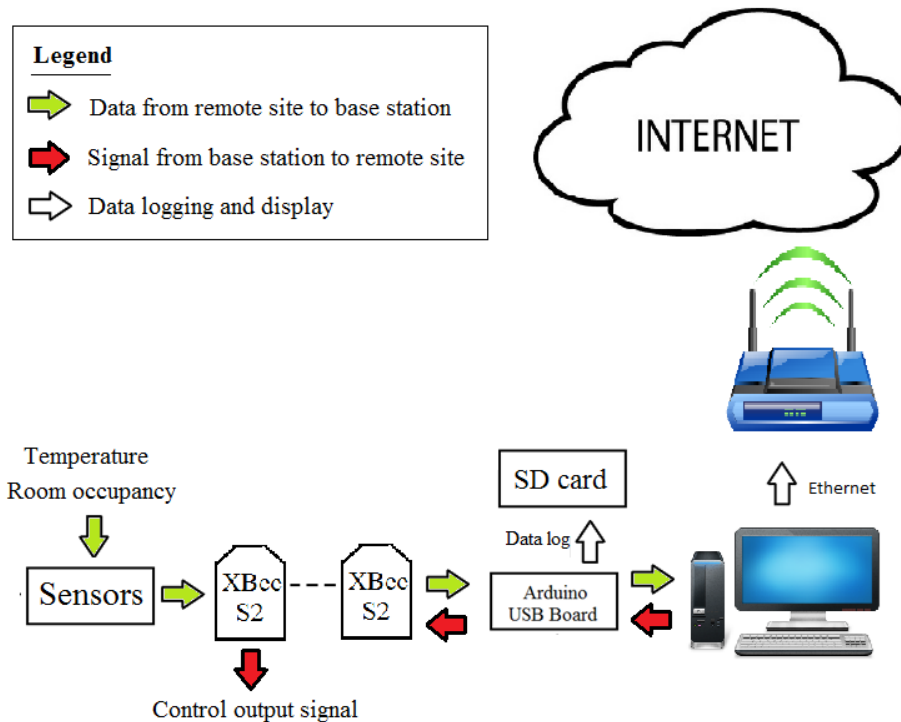


Figure 3.1 The Proposed Methodology of the System

From the diagram above, the wireless sensor network system is split into three layers which are the sensor layer, coordinator layer and also supervisory layer include data logging. There are three parameters can be found at the end device which are temperature sensor, magnetic door sensor and also motion sensor. The data measure by these three sensors will be pass through a RF module (transmitter) and the RF module will transmit the data to another RF module (receiver) at the coordinator.

Then at the coordinator, the microcontroller collects all the data from the transmitter by the receiver. The microcontroller also does all the calculation and produce out a readable data for human. Then the data can be stored in the SD card. Apart from it, the coordinator is able to send signal to adjust the air-conditioning system and act as a remote in this situation.

Lastly at the internet layer, the data from the coordinator can be store in SD card. Apart from it, coordinator with the Ethernet LAN can allow the data to be working in real time monitoring. This has allowed other users to have a look where there are not available at the base station to view the data collected. It is acting as a real time monitoring device for the whole wireless sensor network system. The summary of the components available in every layers of the system are showed in Table 3.1.

Table 3.1 : Components available in every layers

Location component	Sensor layer	Coordinator layer	Supervisory layer
Sensor	available	-	-
Wireless module	available	available	-
Microcontroller	-	available	-
Computer	-	available	available
Ethernet	-	-	available

3.2 System Specifications

XBee series module S2 is used in the wireless sensor network in the communication system. This XBee series module S2 is using Zigbee protocol as mentioned in the literature review. The XBee implements at the end device and also the coordinator in order to transfer the data. There can be a few number of end device in the system but only one coordinator. This coordinator acts as the parent device for all the end device where all the data be transmitter to the coordinator via the aids of XBee communication system. Apart from it, the end device is installed in several of the rooms whereas the coordinator is only installed in a base station that monitor all the sensor for all the end devices. The summary of the characteristics between end device and the coordinator is showed in Table 3.2.

Table 3.2 : The characteristic between end device and the coordinator

End Device	Characteristic	Coordinator
Obtain reading from the sensors and transmit to the coordinator	Role	Receive the reading from all the end devices
Can be more than one	Quantities needed in a system	Can only have one
at room that want to monitor	Location installed	Base station
1. XBee Series module S2 2. Temperature sensor 3. Motion sensor	Hardware	1. XBee Series 2 module 2. XBee shield 3. Arduino USB Board 4. Ethernet Shield 5. Ethernet cable

3.2.1 Network topologies

As discussed in the chapter 2, there are variety of network topologies (pair, star, mesh and cluster tree) but in this project, we first try out the pair topology where it is the fundamental and then only go to the complex which are the mesh, star or the cluster tree.

3.2.2 Hardware

There are a few important hardware that are needed for this project. So in this section, we will discuss in details about the hardware functionality and also apply at which layers.

- i. Arduino Mega – This microcontroller has to be used at the coordinator side. The specification is discussed at Section 2.2.1. It works as collecting the data and save in the SD card and also upload the data to the server. Figure 3.2 show the feature of the device.



Figure 3.2 Arduino Uno

- ii. Wireless Module – XBee series module S2 is used for the communication of the system network and transmission of the data. This device is installed at the end device, router and coordinator. Figure 3.3 show the figure of XBee series module S2.



Figure 3.3 XBee Series Module S2

- iii. Temperature sensor – LM335 is used for measuring the temperature with the sensitivity of 10mV/K. It is connected to the end device. The feature of the LM335 is illustrated in the Figure 3.4



Figure 3.4 Temperature sensor (LM335)

- iv. Motion sensor – PIR motion sensor is used to detect the habitation of the room. This sensor is also connected to the end device. Figure 3.5 has displayed the structure of the PIR motion sensor.



Figure 3.5 PIR Motion Sensor

- v. Ethernet Shield – it is used to allow Arduino to access into the Ethernet port where install only at the base station. Figure 3.6 has showed the Ethernet Shield.

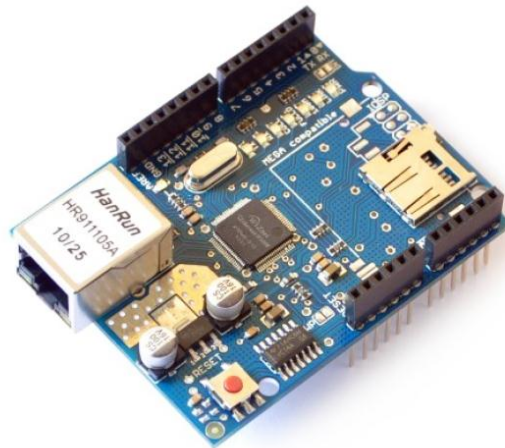


Figure 3.6 Ethernet Shield

- vi. XBee shield – it is used to for the connection between Arduino board and XBee series module S2. It is used for every each of the XBee series module S2. Figure 3.7 shows the XBee shield device.

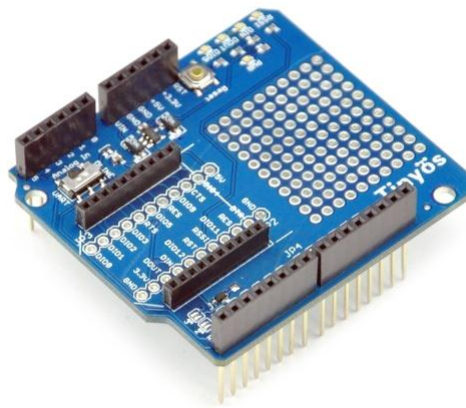


Figure 3.7 XBee Shield

3.2.3 Software

The software that are going to be used is going to be discussing in this section as below.

1. Arduino IDE – A software that allow user to edit the code and upload the code to the Arduino.
2. XCTU – A software to program for XBee radio.

3.2.4 Cloud

Ethernet LAN is used as the internet media instead of WiFi. This is because Ethernet can provide reliable network system which is very important towards the monitoring

system and it able to provide high speed data rate between the communication of the microcontroller and also the server. Apart from it, the system with the Ethernet is less complicated if compare to WiFi and also mobile data. Lastly, due to the budget is limited, the Ethernet is chosen as it is affordable.

Referring the Figure 3.1, there are two terminals for the data to be transferred, either SD card or webpage. Data can be stored in SD card and able to be read in the Comma Separated Values (CSV) file. This is able to show the data in the table form. In the file format, data such as time, data, sensor location, temperature, door sensor situation and also motion sensor situation can be from the file.

Referring the Figure 3.1, another terminal is transferred to the webpage. It is the platform where it enables the end user to access the data whenever or wherever they are. This has brought the convenience to the user who are not around at the base station. In order to make this work, this has to configure through the IP address of the router and set the system to the static IP address so there will no assign of different IP every time the system is restart again. Xively website has deployed for posting data to the server.

3.3 Installation venue

The experiment first setup at the residential building during 24th of July until 26th of July. Then there is another second setup of experiment at the same residential building during 31st of July until 2nd of August. The data has been recorded 72 hours without any disturbance that stop the process.

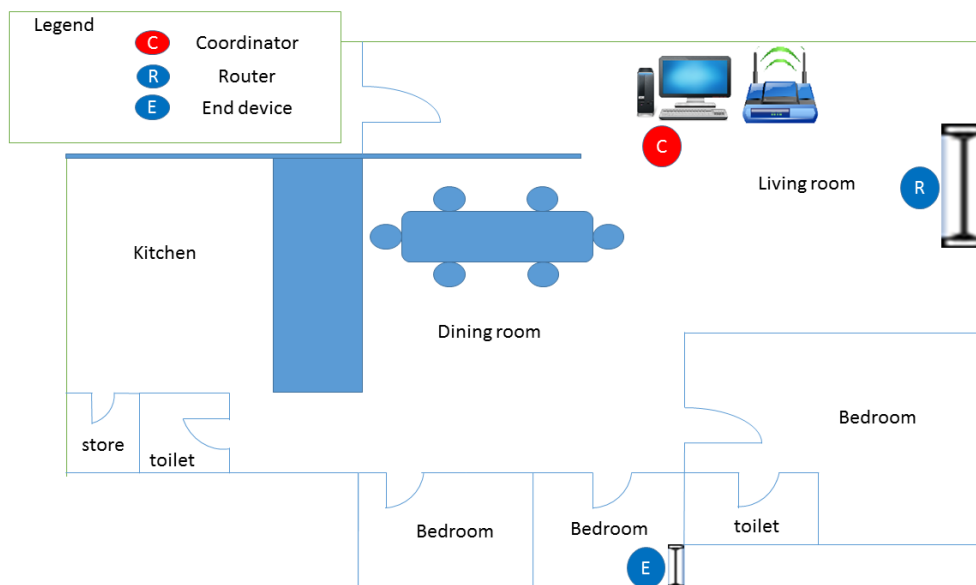


Figure 3.8 Installation Floor Plan During 24th of July Until 26th of July

Figure 3.8 shows the floor plan of installation venue. The router is placed beside the 24/7 open air balcony. The end device is installed at the bedroom and it is placed near to the window too. The Arduino mega, coordinator, is connected to the computer through the USB cable and the Ethernet shield is connect through RJ45 cable to the modem.

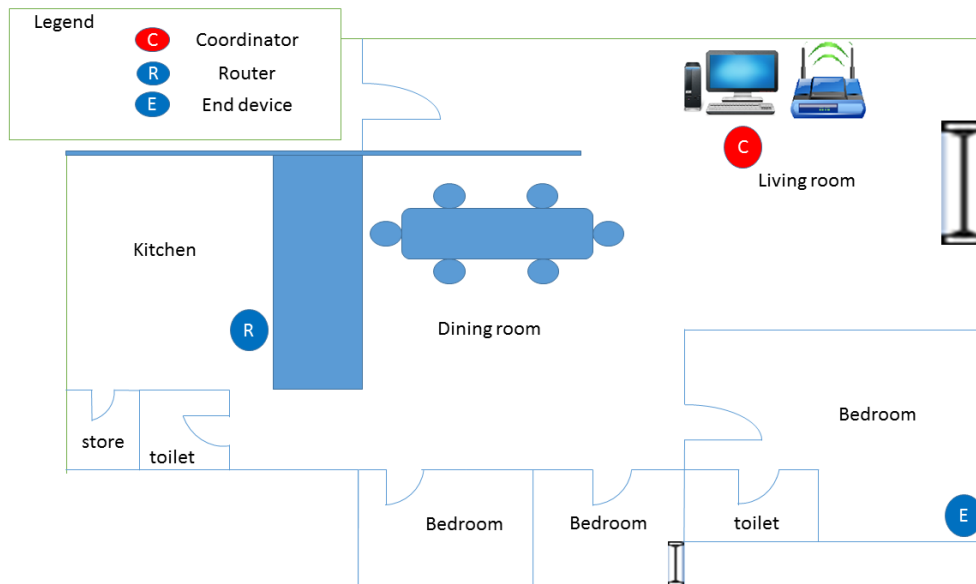


Figure 3.9 Installation Floor Plan During 31st of July Until 2nd of August

Figure 3.9, it is the second installation of the experiment. The router was installed behind the kitchen where a thick wall is block. Then the end device was placed under the bed of the bedroom. Coordinator remain the same place.

3.4 Constructed hardware

At the same time, the designed circuit with sensors of end device and router have to be assembled.

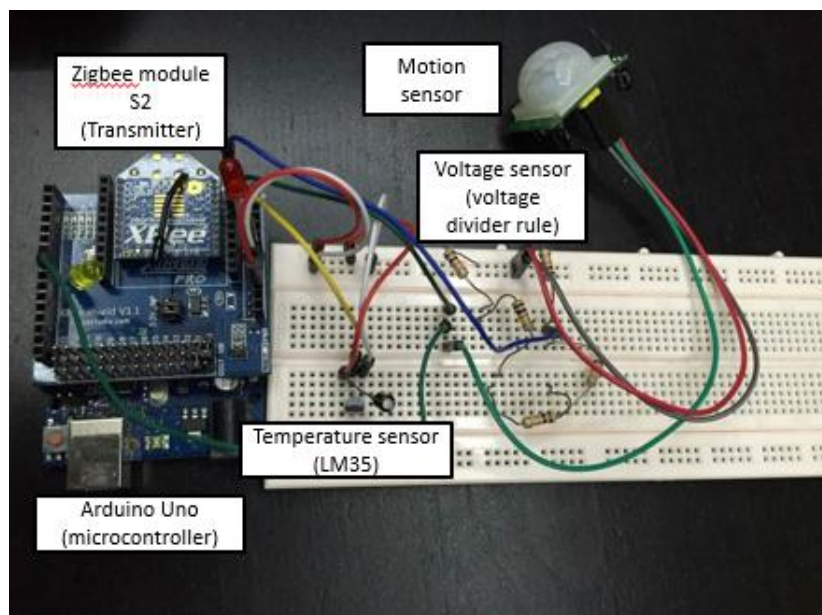


Figure 3.10 Router or End Device Assembled Circuit

Figure 3.10 shows the Zigbee module (transmitter) is stacked on the Zigbee adapter board and the Zigbee adapter board is then stacked on the Arduino Uno. The Arduino Uno functions as power up the Zigbee module in this situation. There are a few sensors available on the breadboard such as voltage sensor, motion sensor and temperature sensor. Those sensors are connected to the Zigbee adapter board at the Pin number of AD0 (analog input), AD2 (digital input) and AD3 (analog input).

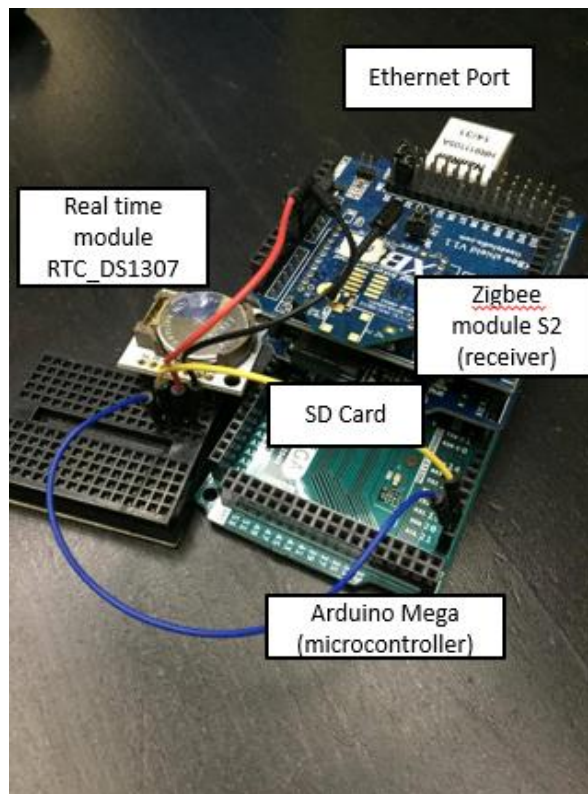


Figure 3.11 Coordinator Assembled Circuit

Figure 3.11 is the coordinator assembled circuit. There are Arduino shield attached to the Arduino Mega and it is stacked with another Zigbee Module (receiver). The Arduino Shield has the SD card slot and also the Ethernet port. Then there is a real time module, RTC_DS1307. This device is to generate the time clock system as the Arduino couldn't do clock setting.

CHAPTER 4

SOFTWARE IMPLEMENTATION

4.1 Configuration of the software

In this project, three importance devices have be brought in which are coordinator, router and end device. All the three device has been set up as their own identity by using the software called XCTU. As mentioned in 3.2.3, XCTU is a software to configure the firmware of the Xbee module.

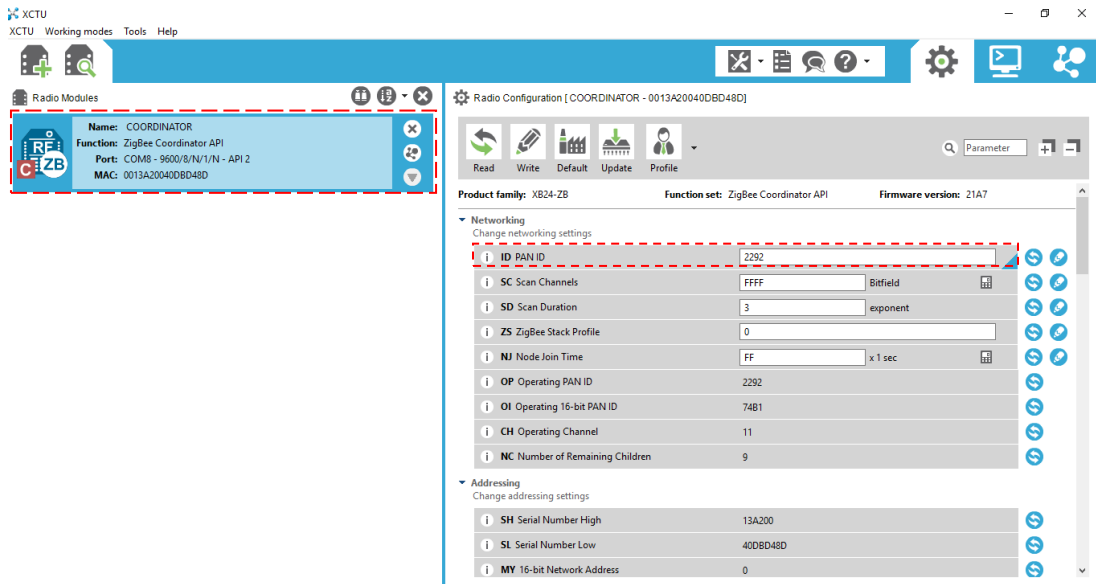


Figure 4.1 XCTU Software Layout

The left column of Figure 4.1 shows the radio module once the USB port is plugged in. Then the software read all the data of the Zigbee module and shows up at the right column. The right column allows users to edit the function of the Zigbee module device. The most important for the configuration at the right corner is the

PAN (personal area network) ID. Users can randomly set a number for it but if the other Zigbee module wants to communication with the system, it has to be the same PAN ID.

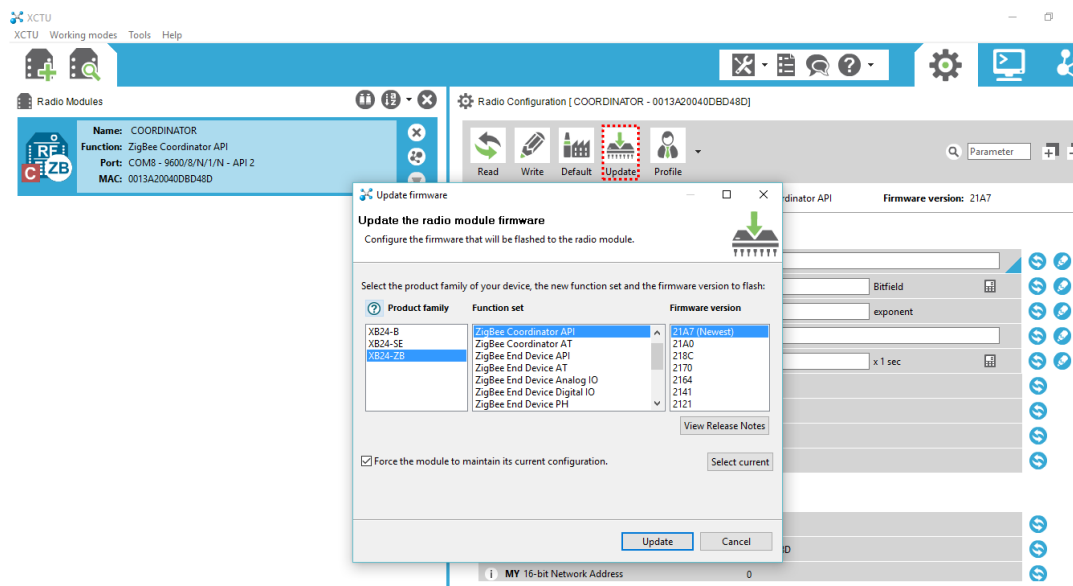



Figure 4.2 XCTU Software Layout (Update Firmware)

Figure 4.2 illustrates the method to update the device firmware. The Zigbee modules have to be using the same firmware only can be communicating to each other. In order to update the firmware, press the update button, then the software will update the system to the latest version of the firmware as show in Figure 4.2. Apart from it, it also can change the identity of the Zigbee module through this button. It can either be coordinator, router or end device. Once the Zigbee Modules are set up, move on to the Arduino Mega stacked with the Zigbee Modules which act as a Coordinator.

Then Arduino mega has to be program with the Arduino IDE as Arduino mega will be the coordinator at the base station. It has to be analysed and rewrite all the data received to human readable data.



```
sketch_jul13a | Arduino 1.6.7
File Edit Sketch Tools Help

sketch_jul13a
#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <Ethernet.h>
#include <HttpClient.h>
#include <Xively.h>
#include "RTCLib.h"

RTC_DS1307 rtc;

// MAC address for your Ethernet shield
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

// Your Xively key to let you upload data
char xivelyKey[] = "I3sAnej2eqrwDCXCriKIBZCSnYZrwlEGeo62cRIKEOhZiyhs";

// Analog pin which we're monitoring (0 and 1 are used by the Ethernet shield)
//int sensorPin = 2;

// Define the strings for our datastream IDs
char sensorId_1[] = "S1";
char sensorId_2[] = "S2";

float tempC;
float voltage;
int Voffset = 40;
float divider3V = 3.0;
```

Figure 4.3 Arduino IDE Software Layout

Figure 4.3 shows the layout of the Arduino IDE software layout. The full coding program of coordinator is attached in the appendix C. Once the code is uploaded to the Arduino mega, it starts to run. Then open up the serial monitor button at the top right corner.

```
Starting single datastream upload to Xively...

Initializing SD card...Card initialized
Title logged
2016/8/12 13:52:56
Location: ROUTER AT
Address: 40 79 D1 1C
Motion:
Supply voltage: 6.23 V
Temperature: 30.00 ^C
Read sensor value 30.00
Uploading it to Xively
xivelyclient.put returned 200

~0000@yÑ0yp00400

2016/8/12 13:53:11
Location: ROUTER AT
Address: 40 79 D1 1C
Motion:
Supply voltage: 6.23 V
Temperature: 30.00 ^C
Read sensor value 30.00
Uploading it to Xively
```

Figure 4.4 Arduino Real Time Monitoring Layout

Figure 4.4 shows the real time monitoring layout. In the serial monitor, at initial place, it shows up if the internet is working or not and it also detects the SD card is available for data logger. Afterwards, it shows up the time, location indicating either router or end device, Address (identity of router or end device), motion, supply voltage and also temperature. Even if the Ethernet cable is not connected to the Arduino Mega or SD card is not logged, the system still can work but the only drawback of the scenario is data is not able to back up or track back from history.

COM8 (Arduino/Genuino Mega or Mega 2560)

```
Error getting IP address via DHCP, trying again...
Starting single datastream upload to Xively...

Error getting IP address via DHCP, trying again...
```

Figure 4.5 Ethernet Error in Real Time Monitoring Layout

```
Initializing SD card...Card failed, or not present
2016/8/12 21:57:39
Location: ROUTER AT
Address: 40 79 D1 1C
Motion: Detected
Supply voltage: 4.74 V
Temperature: 32.34 ^C
error logging date and time
error logging location
error logging address
error logging address
error logging address
error logging address
error logging address
error logging voltage
error logging temperature
error logging motion

~          @y          D4
```

Figure 4.6 Data Logging Error in Real Time Monitoring Layout

Figure 4.5 and Figure 4.6 show the two examples of the error will occur in the system when the Ethernet cannot accessed or SD card is not available.

4.2 Xively Website Setup

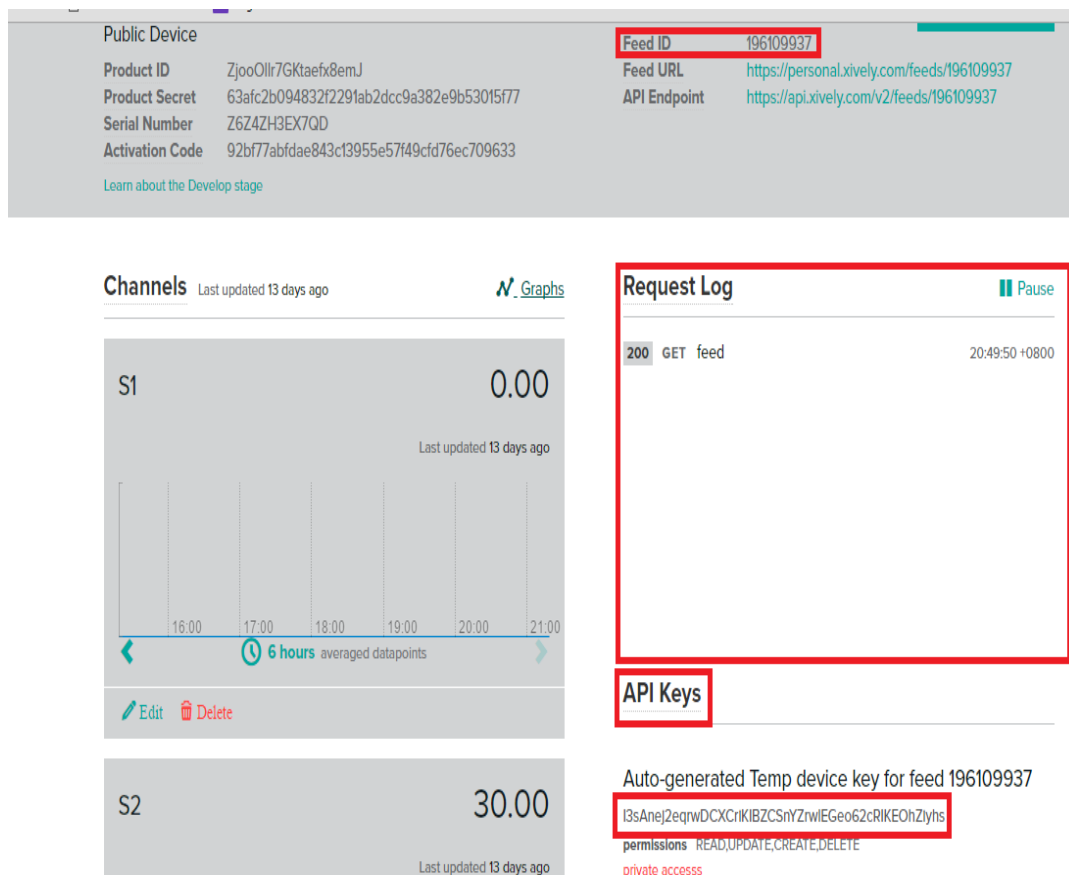


Figure 4.7 Xively Platform Layout

Xively website setup is discussed in this sub-title. Figure 4.7 shows the Xively platform layout. In this Xively’s layout, the three important parameter that is important to the project is the Feed ID, Request Log and also API Keys. The Feed ID and API Keys are used in the coding so this allows us to communicate with the device and also the website. The Request Log shows the data receive time. When there are data received by the website, the Request Log is updated in term of time and data.

4.3 Flow chart of the Arduino programming

In this sub-title, the flow chart of the main coding program is illustrated in the Figure 4.8.

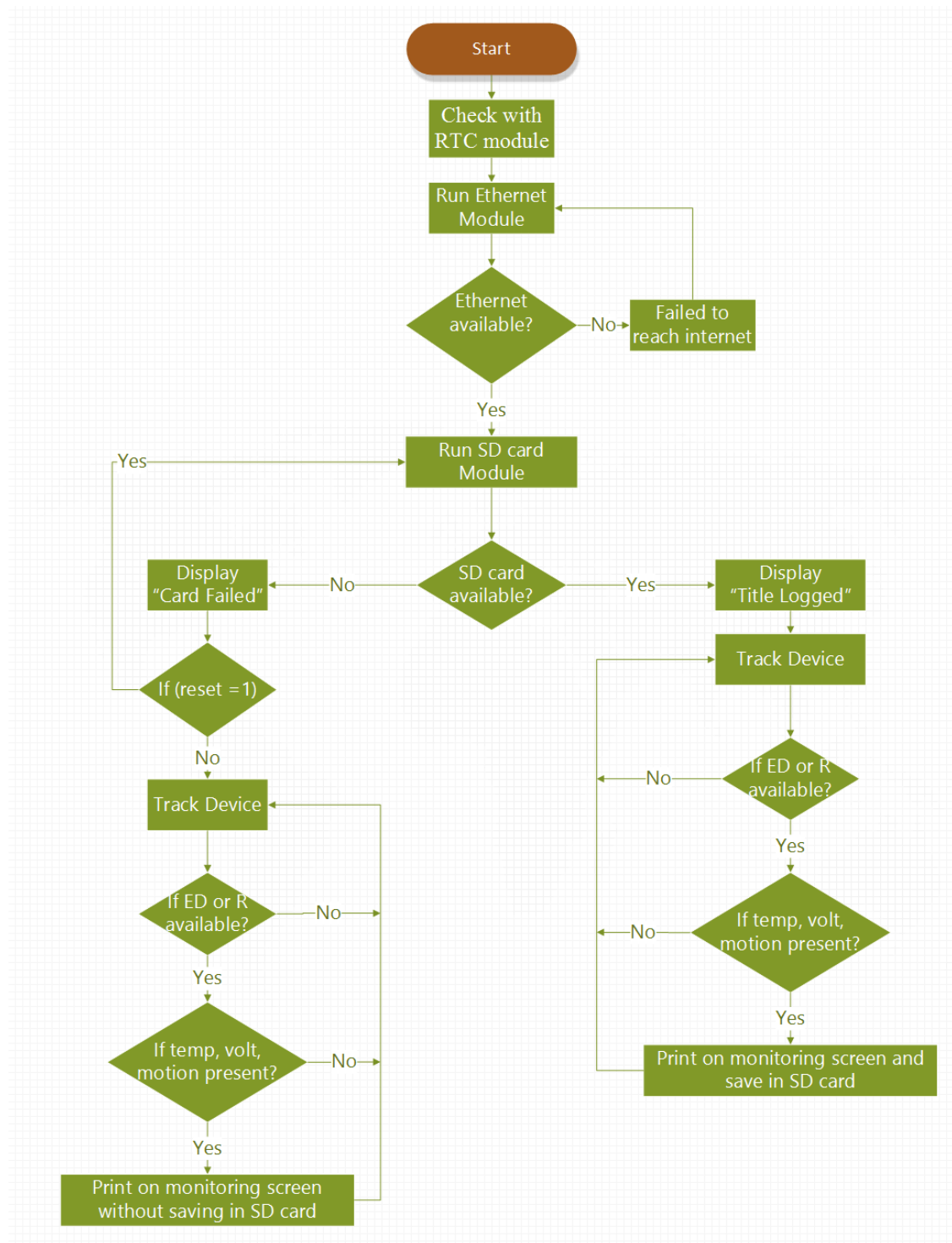


Figure 4.8 Flow Chart of Main Program

From the Figure 4.8, the first start of the program, the flow first runs the Real Time Clock (RTC) module. When the RTC module is valid, the program proceed to

the Ethernet Module. If the Ethernet is not valid, the system will keep on checking until the Ethernet is available. Then, the program is proceeded to the SD card Module. The program then checks if the SD card is available, then two outputs is generated from this situation, either “card failed” or “title logged”. The program continue to track the device no matter the output of the from the SD card module is “card failed” or “title logged”. If the program has tracked a device available, the program continue to read the temperature, voltage and motion reading. The program will keep on tracking the device if no device is available. Once the data has obtained, the program will print on the serial monitor and the program will repeat the flow of tracking device non-stop.

The more detail explanation of the flow chart has split into four sub-program in the section 4.3.1, 4.3.2, 4.3.3 and 4.3.4.

4.3.1 Real Time Clock (RTC) module

It is important to make sure that the Arduino has the same time as the computer. The time of the Arduino and the computer can be synchronized using the run time module. Figure 4.9 shows the sub-program for the run time module.

When power supply is provided, the system checks the time of the RTC module. If the time of the RTC module is equivalent to the current time, it will proceed to the next stages or else the program will correct the time until it generate the correct time. Normally error of RTC module is simple because the only problem that causes this situation to be occur is wrong connection of pin in the hardware. The coding of RTC module is showed in the Figure 4.10. It is easily to be understand for the RTC module.

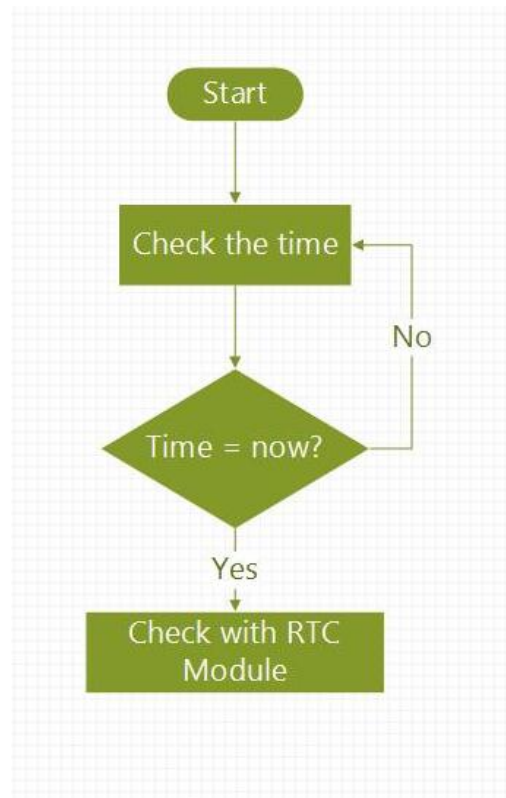


Figure 4.9 Flow Chart of RTC Module

```

void setup()
{
  pinMode(button, INPUT);

  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }
  Wire.begin();
  rtc.begin();

  if (!rtc.isrunning()) {
    Serial.println("RTC is NOT running!");
  }

  DateTime now = rtc.now();
  if ((now.year() != 2016)) {
    rtc.adjust(DateTime(2016, 1, 1, 23, 59, 0));
  }
}

```

Figure 4.10 Coding for RTC Module

4.3.2 Ethernet module

Ethernet Module helps to upload the data to the cloud. Without ethernet module, the data is unable to show up in the cloud. Figure 4.11 shows the flow of the sub-program. The Ethernet module will check the Ethernet connection, if the Ethernet is available, the program will continue to the next module or else the program will keep on check without progressing to further.

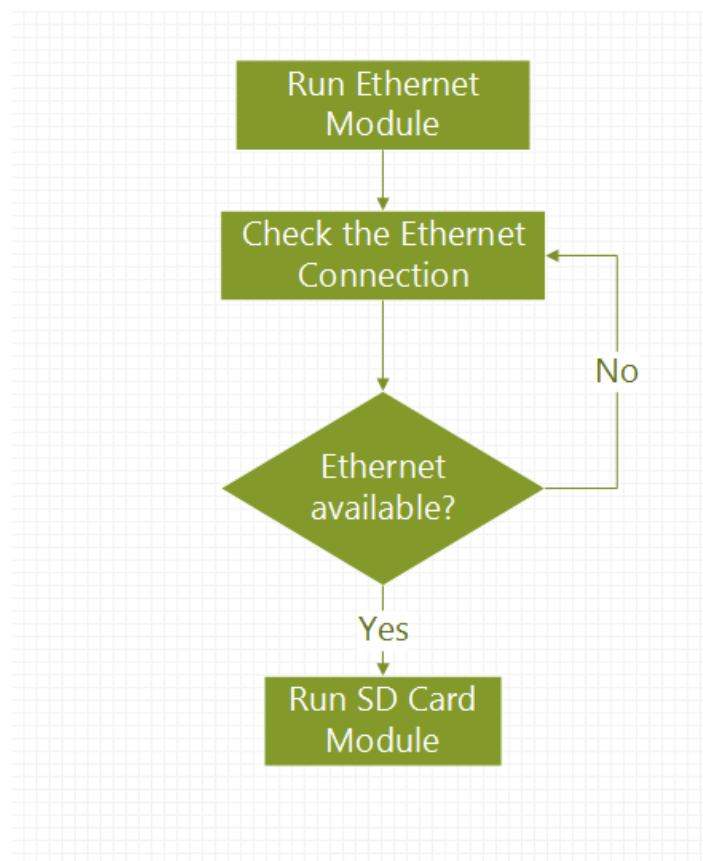


Figure 4.11 Flow Chart of Ethernet Module

Figure 4.12 shows the setup code of the Ethernet module. The code shows the initialize coding that allow Arduino Mega to communicate with the Xively platform. The FEED ID and API key that have mentioned in topic 4.2, is used in the initialize coding of Ethernet.

```

// Your Xively key to let you upload data
char xivelyKey[] = "I3sAnej2eqrwDCXCriKIBZCSnYZrwlEGeo62cRIKEOhZiyhs";

// Analog pin which we're monitoring (0 and 1 are used by the Ethernet shield)
//int sensorPin = 2;

// Define the strings for our datastream IDs
char sensorId_1[] = "S1";
char sensorId_2[] = "S2";
XivelyDatastream datastreams[] = {
  XivelyDatastream(sensorId_1, strlen(sensorId_1), DATASTREAM_FLOAT),
  XivelyDatastream(sensorId_2, strlen(sensorId_2), DATASTREAM_FLOAT)
};
// Finally, wrap the datastreams into a feed
XivelyFeed feed(196109937, datastreams, 2 /* number of datastreams */);

EthernetClient client;
XivelyClient xivelyclient(client);

void setup()
{
  pinMode(button, INPUT);

  // Open serial communications and wait for port to open:
  Serial.begin(9600);

  Serial.println("Starting single datastream upload to Xively...");
  Serial.println();

  while (Ethernet.begin(mac) != 1)
  {
    Serial.println("Error getting IP address via DHCP, trying again...");
    delay(15000);
  }
}

```

Figure 4.12 Setup Coding of Ethernet Module

Coding of uploading data has showed in the Figure 4.13. Basically this process is carried out when the device is tracked and the all the data has flow into the coordinator. The temperature data will then upload to the Xively website.

```

if (!(tempC >= 14.0 && tempC <= 40.0))
{
}
else
{
    if (location == 0x268)
    {
        datastreams[0].setFloat(tempC);

        Serial.print("Read sensor value ");

        Serial.println(datastreams[0].getFloat());
        Serial.println("Uploading it to Xively");
        int ret = xivelyclient.put(feed, xivelyKey);
        Serial.print("xivelyclient.put returned ");
        Serial.println(ret);
    }

    else if (location == 0x1A6)
    {datastreams[1].setFloat(tempROUTER);
        Serial.print("Read sensor value ");
        Serial.println(datastreams[1].getFloat());
        Serial.println("Uploading it to Xively");
        int ret = xivelyclient.put(feed, xivelyKey);
        Serial.print("xivelyclient.put returned ");
        Serial.println(ret);
    }

    Serial.println();
    delay(15);
}

```

Figure 4.13 Uploading Code of Ethernet Module

4.3.3 SD Card module

The SD card module is important as it can act as a storage to save the data. Without SD card module, the serial monitor screen still be able to monitor and print out the data. The only drawback without SD card is unable to track back the history data. Figure 4.14 shows the flow chart of SD card module. The program will check if the SD card available, if yes, the serial monitor will print out “title logged” or else “Card failed”. Proceed to the next condition, if the reset button is pressed, the system will run the SD card module again to check the availability of SD card. If it is not pressed, the system will start to track the device and print out the data without storing the data in SD card.

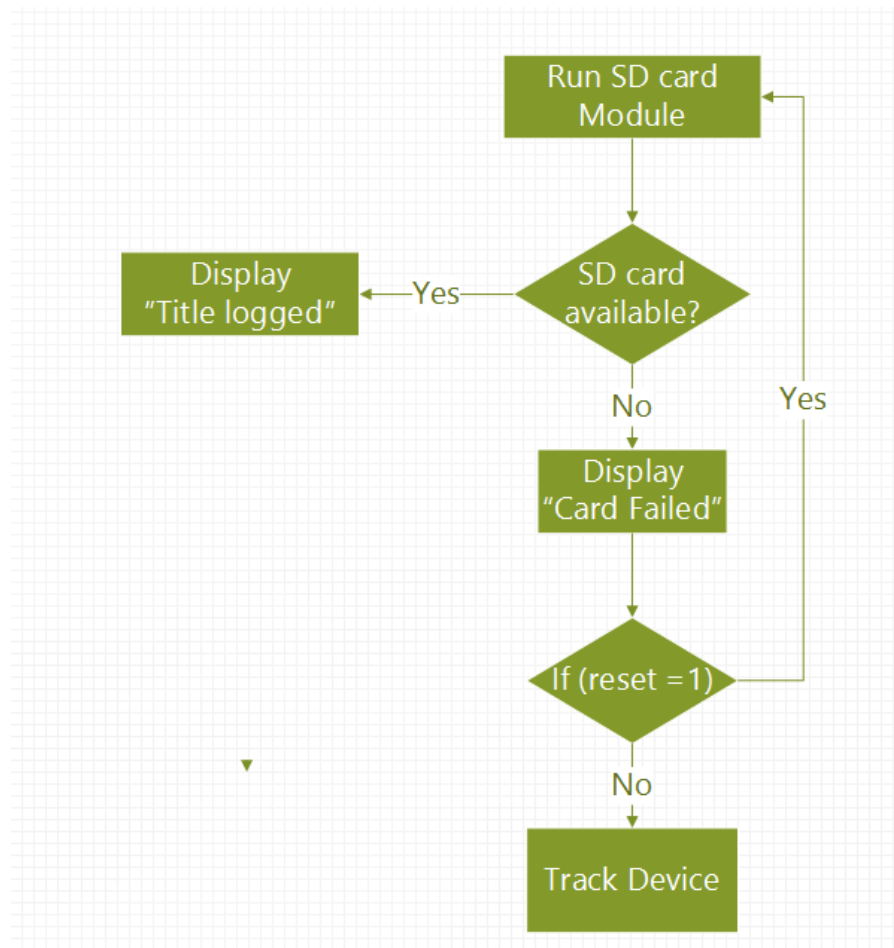


Figure 4.14 Flow Chart of SD Card Module

Referring to the Figure 4.15, the program log the title of the data of date, time, location, address, supply voltage (V), Temperature (°C) and motion to the CSV file. When the data is available to be stored, the next row of data is recorded below the title column by column. Figure 4.16 is the coding to log all the data recorded to the CSV file under the title column by column.

```
Serial.print("Initializing SD card...");

// see if the card is present and can be initialized:
if (!SD.begin(SDchip)) {
  Serial.println("Card failed, or not present");
  // don't do anything more:
  return;
}
Serial.println("Card initialized");

File dataFile = SD.open("datalog.csv", FILE_WRITE);

// if the file is available, write to it:
if (dataFile) {
  dataFile.println("Date,Time,Location,Address,,,Supply voltage (V),Temperature ('C),Motion");
  dataFile.close();
  Serial.println("Title logged");
}
// if the file isn't open, pop up an error:
else {
  Serial.println("error logging title");
}
```

Figure 4.15 Initialize Coding of SD Card Module


```

// ***** DATA LOGGING SESSION *****
File datetimelog = SD.open("datalog.csv", FILE_WRITE);

// if the file is available, log time
if (datetimelog) {
  datetimelog.print(now.year(), DEC);
  datetimelog.print("/");
  datetimelog.print(now.month(), DEC);
  datetimelog.print("/");
  datetimelog.print(now.day(), DEC);
  datetimelog.print(",");
  datetimelog.print(now.hour(), DEC);
  datetimelog.print(":");
  datetimelog.print(now.minute(), DEC);
  datetimelog.print(":");
  datetimelog.print(now.second(), DEC);
  datetimelog.print(",");
  datetimelog.close();
}
// if the file isn't open, pop up an error:
else {
  Serial.println("error logging date and time");
}
delay(10);

// Log location
locationlog(location);
delay(10);

// Log address
addresslog(Address_byte_1);
addresslog(Address_byte_2);
addresslog(Address_byte_3);
addresslog(Address_byte_4);
delay(10);

// Log voltage
voltage(voltage);
delay(10);

// Log temperature
temperaturelog(tempC);
delay(10);

// Log motion
motionlog(digitalin);
delay(10);

```

Figure 4.16 Coding for Data Storing in CSV File

4.3.4 Track Device Module

This section discusses the program detect either it is end device or router follow by tracking the temperature, voltage and motion data. The flow chart is illustrated in Figure 4.17

When the program detected the address of 40, 79, D1, 1C, the serial monitor will print out the device as router and followed by the temperature, voltage and motion reading. Meanwhile, if the address tracked is 40, 79, D0, DF, the serial monitor will print out the device as the end device and follow by all the data reading. If none of the address above is tracked, there is no temperature, voltage and motion data will not be appeared in the serial monitor and the program will keep tracking for the device.

Figure 4.18 and Figure 4.19 show the coding of the process presenting the data to the serial monitor. The program prints the date time in the first row of the serial monitor. Then in the coding, it shows how to differentiate the identity of the router or the end device by summing up the address for all the number of address. After the verification of router or end device, the program start to read the data of motion, voltage and the temperature as illustrated in Figure 4.19.

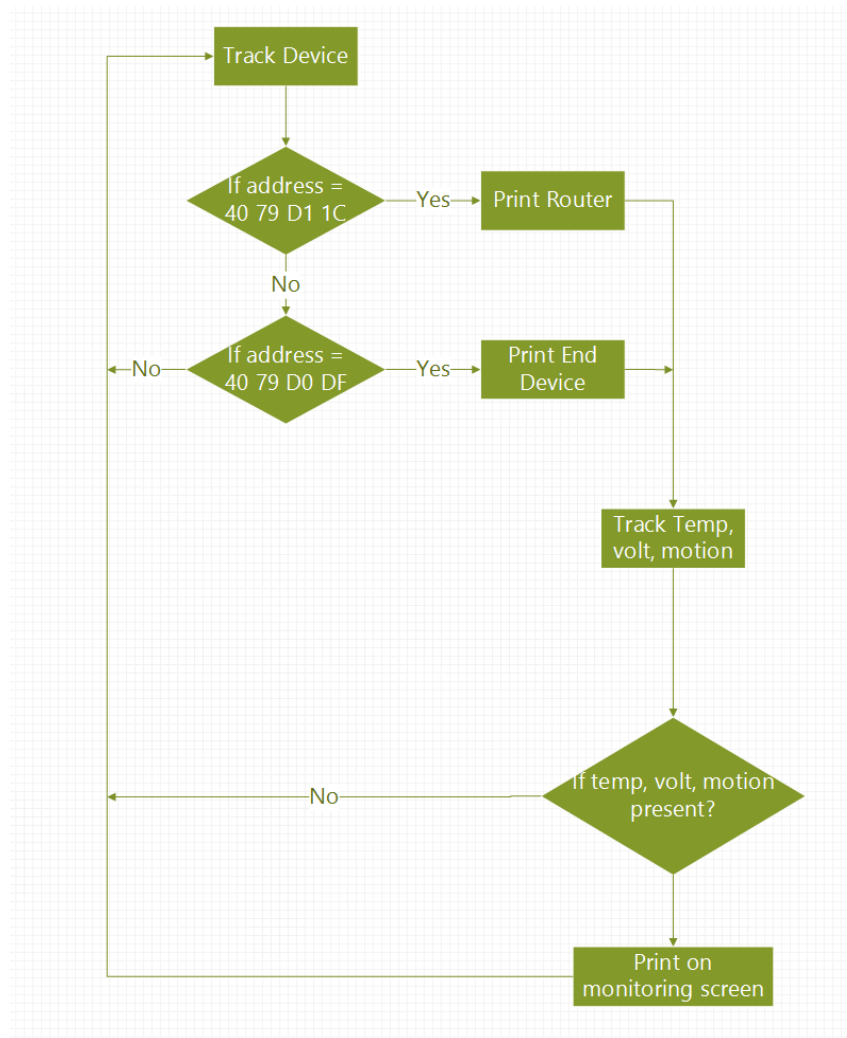


Figure 4.17 Flow Chart of Tracking Device

```

void loop()
{
  if (Serial.available() > 26) {
    if (Serial.read() == 0x7E) {
      digitalWrite(ReceiveLED, HIGH);
      delay(100);
      digitalWrite(ReceiveLED, LOW);

      // Obtain date and time from RTC module and show it in Serial monitor
      DateTime now = rtc.now();
      Serial.print(now.year(), DEC);
      Serial.print("/");
      Serial.print(now.month(), DEC);
      Serial.print("/");
      Serial.print(now.day(), DEC);
      Serial.print(" ");
      Serial.print(now.hour(), DEC);
      Serial.print(":");
      Serial.print(now.minute(), DEC);
      Serial.print(":");
      Serial.print(now.second(), DEC);
      Serial.println();
      delay(10);

      for (int i = 0; i < 8; i++) {
        byte discardByte = Serial.read(); // Discard unused byte
        delay(10);
      }

      // Read only lower 4 byte of source address
      int Address_byte_1 = Serial.read();
      delay(10);
      int Address_byte_2 = Serial.read();
      delay(10);
      int Address_byte_3 = Serial.read();
      delay(10);
      int Address_byte_4 = Serial.read();
      delay(10);

      int location = Address_byte_1 + Address_byte_2 + Address_byte_3 + Address_byte_4;
      //Serial.print("Location HEX: ");
      //Serial.println(location, HEX);
      Serial.print("Location: ");
      switch (location) {
        case 0x268:
          Serial.println("END DEVICE AT");
          break;
        case 0x1A6:
          Serial.println("ROUTER AT");
          break;
        default:
          Serial.println("UNKNOWN");
          break;
      }
      delay(10);
    }
  }
}

```

Figure 4.18 Coding of Presenting the Data

```

Serial.print("Address: ");
Serial.print(Address_byte_1, HEX);
Serial.print(" ");
Serial.print(Address_byte_2, HEX);
Serial.print(" ");
Serial.print(Address_byte_3, HEX);
Serial.print(" ");
Serial.println(Address_byte_4, HEX);
delay(10);

for (int j = 0; j < 8; j++) {
  byte discardByte1 = Serial.read(); // Discard unused byte again
  delay(10);
}

int digitalin = Serial.read();

Serial.print("Motion: ");
if (digitalin == 0x16 || digitalin == 0x14 || digitalin == 0x6 || digitalin == 0x4) {
  Serial.println("Detected");
} else {
  Serial.println("");
}
delay(10);

// Read analog samples from pin A0
// Then convert to battery voltage
int analog_1_MSB = Serial.read();
delay(10);
int analog_1_LSB = Serial.read();
delay(10);
int analog_1_reading = analog_1_LSB + (analog_1_MSB * 256);
voltage = (analog_1_reading + Voffset) * 1.2 / 1024.0 * divider5V;
Serial.print("Supply voltage: ");
Serial.print(voltage);
Serial.println(" V");

int analog_4_MSB = Serial.read();
delay(10);
int analog_4_LSB = Serial.read();
delay(10);
int analog_4_reading = analog_4_LSB + (analog_4_MSB * 256);
tempC = (analog_4_reading) * 1.2 / 1024.0 * 100.0;
Serial.print("Temperature: ");
if (location == 0x268) {
  if (tempC >= 14.0 && tempC <= 40.0) {
    Serial.print(tempC);
  }
} else {
  Serial.print(tempC);
}
Serial.println(" ^C");

// Store temperature value of router for sending control signal automatically
if (location == 0x1A6) {
  tempROUTER = tempC;
}

```

Figure 4.19 Coding of Presenting the Data

CHAPTER 5

RESULT AND DISCUSSION

5.1 Preliminary Results from 24th of July to 26th of July

The experiment has been taken out in the residential area. The results of the data has been recorded in the website and in the CSV file in the SD card.

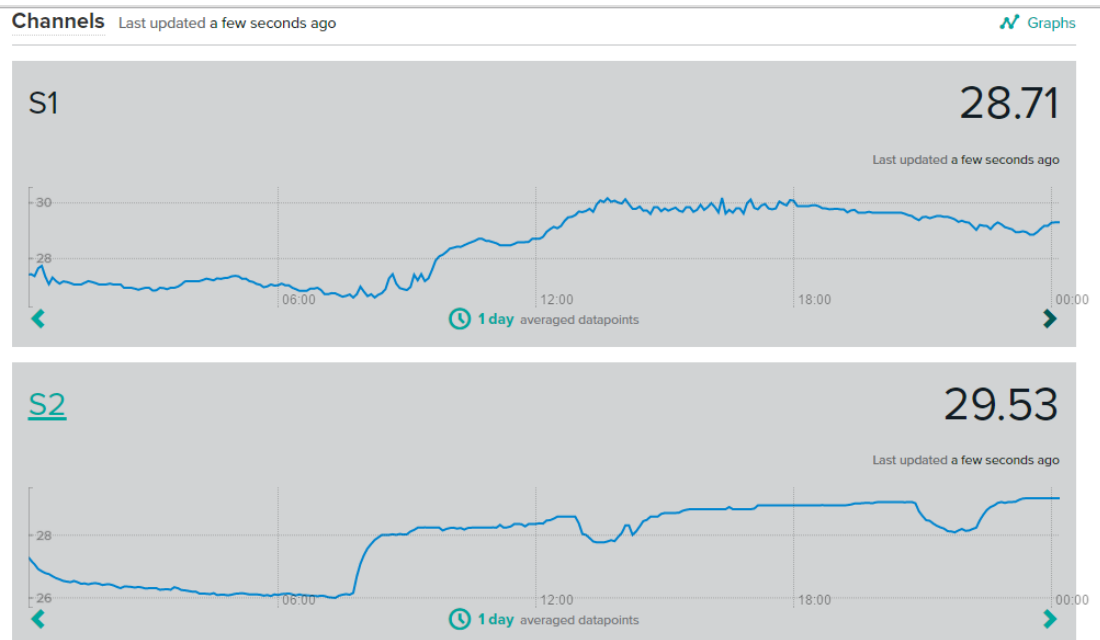


Figure 5.1 Temperature Data Recorded from Website on 24th of July

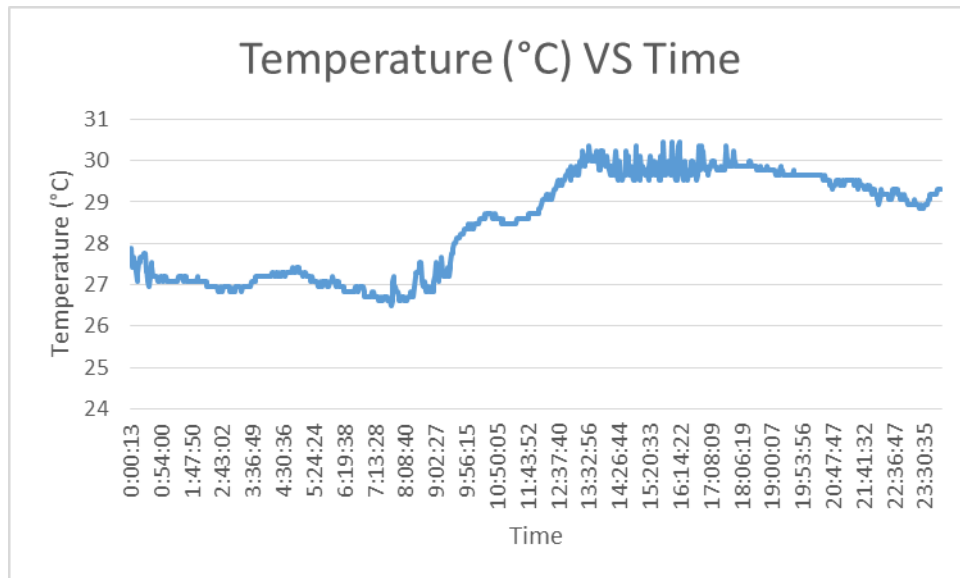


Figure 5.2 Temperature data of Router from SD card during 24th of July

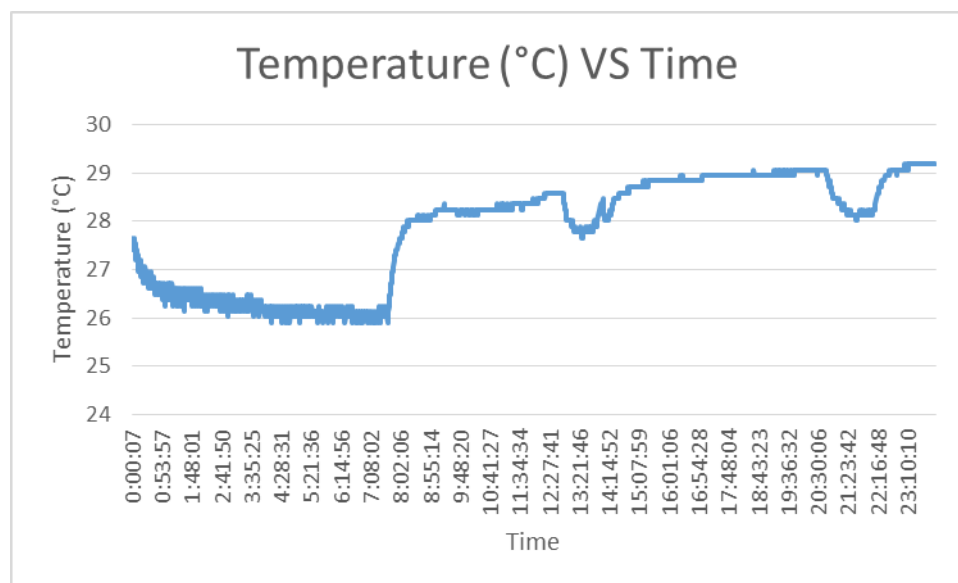


Figure 5.3 Temperature Data of End Device from SD Card on 24th of July

The data recorded from the website during 24th of July is shown in Figure 5.1. The S1 graph is the data collected from the router while the S2 graph is the data collected from the end device. At the same time, Figure 5.2 and Figure 5.3 are the results from the SD card, where Figure 5.2 is from the router and Figure 5.3 is from the end device. Through the comparison between the results from the website and the results from the SD card, the graphs which have been plotted out are identical. Thus, the results can conclude that there are no errors of data. (E.g. internet down causes the data cannot be uploaded)

From the floor plan Figure 3.3, the end device is installed in the bedroom then the router is installed to the window near the balcony. During 12am until 8am, the temperature for the router and end device are low. The temperature of the end device is low during this few hours is due to the running of air conditioners whereas the router temperature is low because the window is open and the cool air diffuse in to the living room during that few hours. After 8am, the temperature for both devices start to increase. Temperature of end device has increased rapidly as the air conditioner has switched off thus it back to normal room temperature whereas the temperature of router is increasing slowly as the temperature outside the window is cooler compare to the living room so cool air still diffuse in to the room but not as cold as during 12am to 8am. After 12pm, the temperature at the router get to the peak value which is around 30.5°C and the read is fluctuate in between 29°C to 30.5°C. On the other hand, the temperature of the end device, is remain in between 28°C to 29°C.

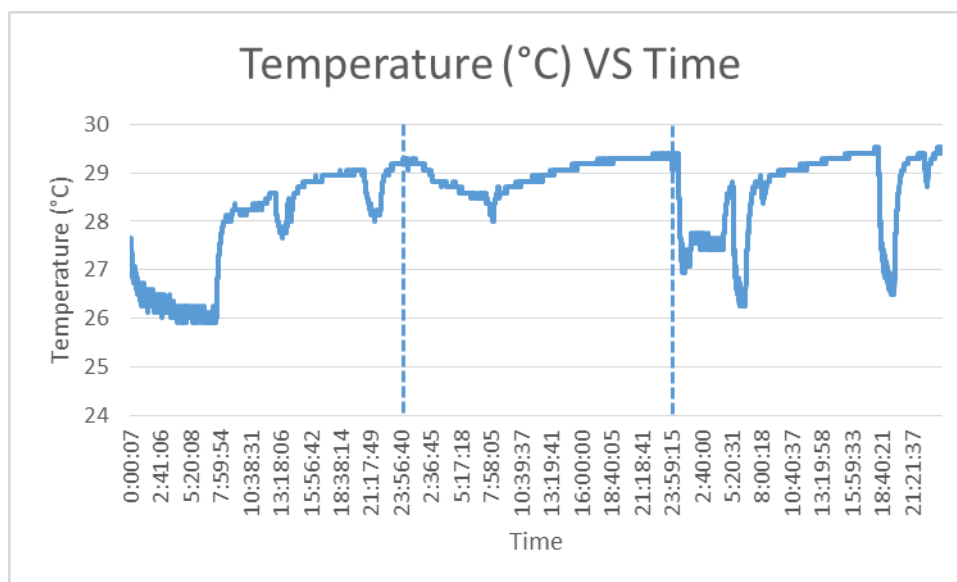


Figure 5.4 Temperature Data of End Device from 24th of July to 26th of July

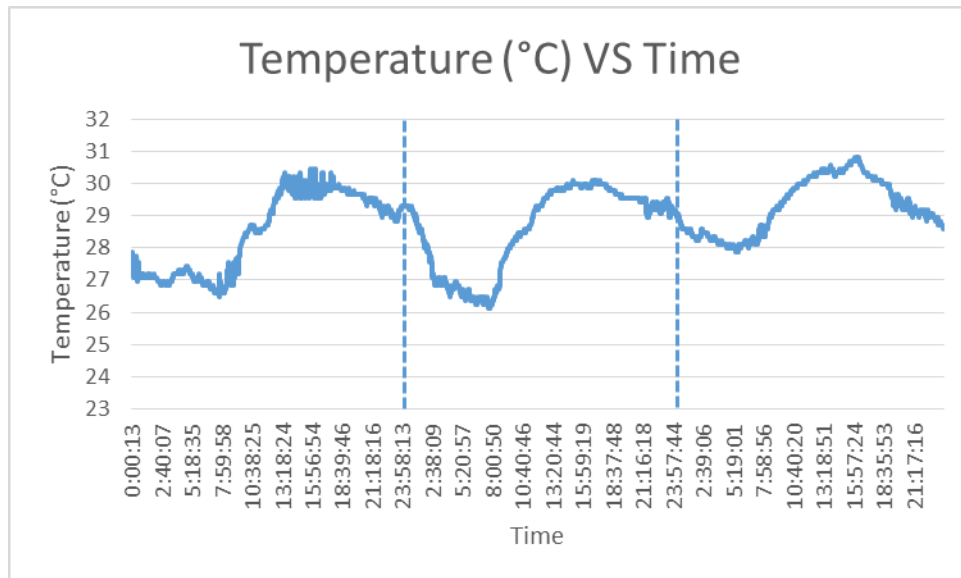


Figure 5.5 Temperature Data of Router from 24th of July to 26th of July

In the Figure 5.4 and Figure 5.5 are the temperature data for three days. Through the Figure 5.4, on the date of 25th of July, the temperature of end device during 12am to 8am, the temperature doesn't reach 26°C-27°C. From the temperature reading, the scenario of the air conditioner is not being switched on can be concluded for the cause of data obtained. Whereas, the reading for temperature at the router is having the trend as if on 24th of July. On 26th of July, temperature data for the end device is fluctuating. Suppose after 12am, air conditioner will be used, but during 26th of July, the air conditioner has been used on and off during 12am-7am. In between this hours, there are up and down of the reading, this is because the door is not closed when enter thus the cool air has diffuse out from the bedroom. Then during 4pm until 6.30pm, the air conditioner has been switched on again thus there is a sharp drop in the graph during that period of time. On the other hand, the temperature data obtained by the router is as smooth as usual. In next

5.2 Results from 31st of July to 2nd of August

During 31st of July until 2nd of August, the second experiment has been carried out. The experiment has been setup as in the Figure 3.4. The end device is placed underneath of the bed but still the bedroom is with the air conditioner. Then the

router is placed 12 meters apart from the coordinator and it is placed in kitchen where there is thick wall covered.

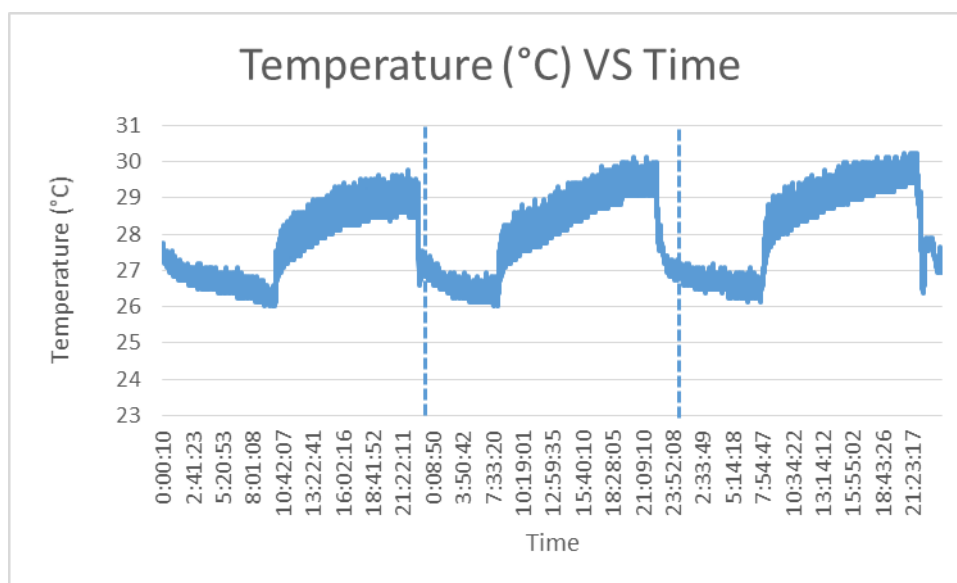


Figure 5.6 Temperature Data of End Device from 31st of July to 2nd of August

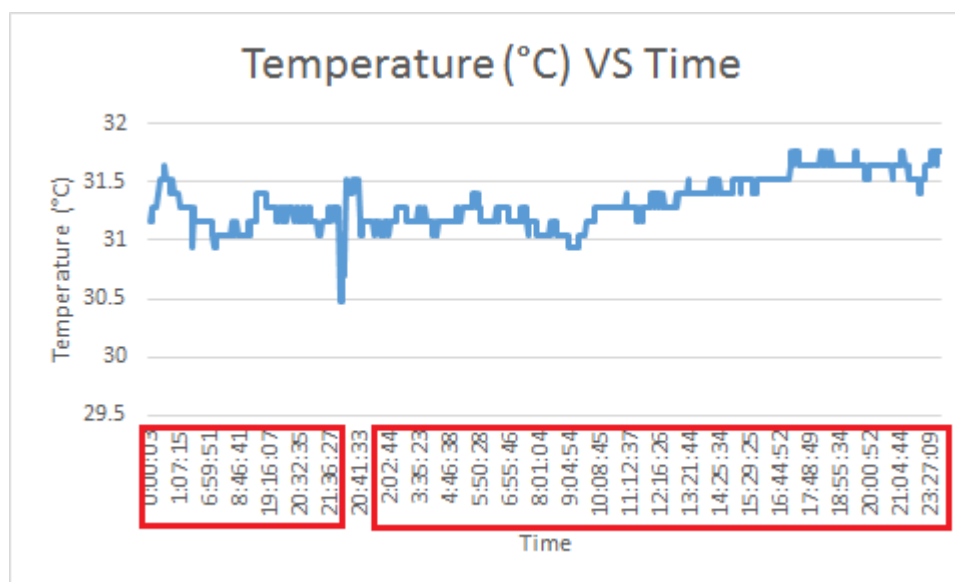


Figure 5.7 Temperature Data of Router from 31st of July to 2nd of August

Referring the graph in Figure 5.6, the trend of the result for the 3 days are the same. The result is consistent during day time and night time. However, referring to the Figure 5.7, the temperature data that has recorded from the router during 31st of July until 2nd of August, the data of 1st of August has missed out.

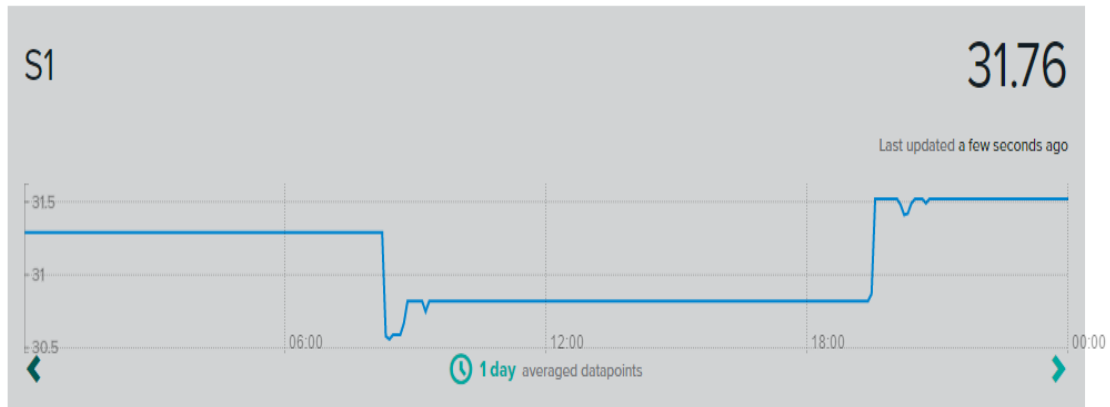


Figure 5.8 Temperature Data Recorded from Website on 1st of August

As the result can save in the SD card or in the website, the result from the website as illustrated in the Figure 5.8 has showed that the temperature data is not recorded consistently. The results in the graph with the horizontal line along the time has indicated no data flow in during that period of time.

Two scenarios can be predicted from this situation are:-

- The router has provided with the supply however, the communication between the coordinator is limited and thus the router has failed to communicate with the parent device.
- The router has not provided with any supply as the socket of the power supply might has some problem.

From this two scenarios, the first scenario probably has the higher chances to be occurred. From Figure 5.8, a few number of data is still able to be recorded. In this case, there still has the data able to flow into the coordinator. Apparently, the signal is not strong enough to reach the parent device. Thus, it lost connection during the transmission of data.

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion

WSN provides a lot of advantages in monitoring the building temperature. One of the major advantages it can forward the data through radio module to the base station instead of the wire-connected. This can reduce the installation cost of the overall system because the use of wire-connected is costly nowadays.

The temperature monitoring system with the implementation of Zigbee as communication between the coordinator with either end device or the router has been developed. It allows the real-time monitoring for voltage level of battery, the surrounding temperature and also motion detection. Apart from the real-time monitoring, the system allows the data to be stored in the SD card and this allows users to track back the history data. Besides, the temperature data is able to monitor through the cloud. This allows users to monitor the data no matter where they are.

After all the experiments have carried out in the residential area, the results obtained in the SD card is analysed by plotting into the temperature against time graph. Apart from it, the results in the cloud is captured. By comparing this two results, the data are identical. Besides, the results obtained allow users to know the duration of air conditioner is switched on and this can relate how much the cost

spend on the air conditioner. Lastly, the accuracy of the temperature data of the system has achieved 90% the same as compare to the thermometer.

In the nutshell, most of the problems face during the project have been encountered. However there are some of the problems are not solved due to the budget limited and also constraint in time. Last of all, all the stated objectives of the project are met and achieved.

6.2 Recommendations

The recommendations to improve the project in future is proposed in this section.

6.2.1 Increase the number of sensor nodes

In current project, there are only one coordinator, router and end device. Thus, the coverage of the wireless network is limited. In order to have larger coverage in a building, large quantities of the sensor nodes are required. After increasing the numbers of the sensor nodes in the wireless network, more data can be measured and recorded at different locations in a building. Therefore, most of the area in the building able to monitor through the base station.

6.2.2 Change the network topology of the system

In order to increase the number of sensor nodes, the network topology has to change from star network to cluster tree network. The data can only be transmitted through the end device or the router to the coordinator in current project. In order to change to cluster tree network, the router must have the function of bouncing the data from end device to the coordinator. This can be change in the software of XCTU and add in some programming code in the Arduino IDE. Therefore, cluster tree network topologies is able to deploy if the router setting has changed to bouncing the data instead of recording data.

6.2.3 Increase the parameter to study

The parameters recorded in current project is only voltage of battery, motion and also temperature data. There are a few more function-able pin from the Zigbee S2 module. It is good to fully utilise the pin available. Parameter such as humidity, light intensity can be added into the project.

6.2.4 Provide output signal from the cloud

The cloud is only able to generate the graphical data for users to analyse in current project. In future, the cloud can have the function of giving output signal. For example, providing output signal such as switching off the air conditioner when it is not using. This possesses tremendous advantages as it fully utilise the cloud service.

In order to achieve this recommendation, the current cloud service has to be neglected and the whole system has to start from the scratch again. The knowledge of Java language, HTTP, processing are required to achieve this recommendation.

REFERENCES

- Arduino - ArduinoBoardUno [WWW Document], n.d. URL <https://www.arduino.cc/en/Main/ArduinoBoardUno> (accessed 4.1.16).
- Huang, Y., Yu, W., Osewold, C., Garcia-Ortiz, A., 2016. Analysis of PKF: A Communication Cost Reduction Scheme for Wireless Sensor Networks. *IEEE Trans. Wirel. Commun.* 15, 843–856. doi:10.1109/TWC.2015.2479234
- Kasapovic, S., Doric, E., Banjanovic-Mehmedovic, L., 2015. An approach to wireless sensor network design in home environment using ZigBee protocol, in: 2015 23rd International Conference on Software, Telecommunications and Computer Networks (SoftCOM). Presented at the 2015 23rd International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pp. 185–189. doi:10.1109/SOFTCOM.2015.7314066
- Lehr, W., McKnight, L.W., 2003. Wireless Internet access: 3G vs. WiFi? *Telecommun. Policy, Competition in Wireless: Spectrum, Service and Technology Wars* 27, 351–370. doi:10.1016/S0308-5961(03)00004-1
- Makki, S.A.M., Pissinou, N., Daroux, P., 2003. Mobile and wireless Internet access. *Comput. Commun., Advances in Computer Communications and Networks: Algorithms and Applications* 26, 734–746. doi:10.1016/S0140-3664(02)00208-6
- Nikhade, S.G., 2015. Wireless sensor network system using Raspberry Pi and zigbee for environmental monitoring applications, in: 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM). Presented at the 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), pp. 376–381. doi:10.1109/ICSTM.2015.7225445
- Piyare, R., Park, S., Maeng, S.Y., Park, S.H., Oh, S.C., Choi, S.G., Choi, H.S., Lee, S.R., 2013. Integrating Wireless Sensor Network into Cloud services for real-time data collection, in: 2013 International Conference on ICT Convergence (ICTC). Presented at the 2013 International Conference on ICT Convergence (ICTC), pp. 752–756. doi:10.1109/ICTC.2013.6675470

Simek, M., Mraz, L., Oguchi, K., 2013. SensMap: Web framework for complex visualization of indoor outdoor sensing systems, in: 2013 International Conference on Indoor Positioning and Indoor Navigation (IPIN). Presented at the 2013 International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp. 1–5. doi:10.1109/IPIN.2013.6851425

Sinha, N., Pujitha, K.E., Alex, J.S.R., 2015. Xively based sensing and monitoring system for IoT, in: 2015 International Conference on Computer Communication and Informatics (ICCCI). Presented at the 2015 International Conference on Computer Communication and Informatics (ICCCI), pp. 1–6. doi:10.1109/ICCCI.2015.7218144

R. Faludi, *Building Wireless Sensor Network: With ZigBee, XBee, Arduino, and Processing*, O'Reilly Media, 2010.

APPENDICES

APPENDIX A: Graphs plotted from SD card

In this section, graph of temperature against time is arranged in the sequence below :

1. Preliminary Result in residential building

- 24th of July 2016 – End Device
- 24th of July 2016 – Router
- 25th of July 2016 – End Device
- 25th of July 2016 – Router
- 26th of July 2016 – End Device
- 26th of July 2016 – Router

2. Second Result in residential building

- 31st of July 2016 – End Device
- 31st of July 2016 – Router
- 1st of August 2016 – End Device
- 1st of August 2016 – Router
- 2nd of August 2016 – End Device
- 2nd of August 2016 – Router

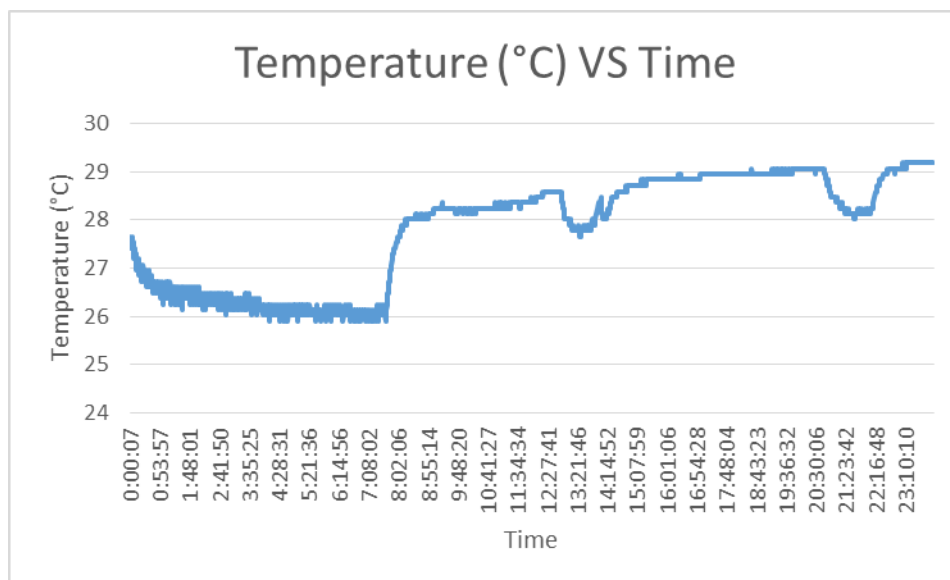


Figure A.1 Temperature vs Time of End Device Graph (24th July 2016)

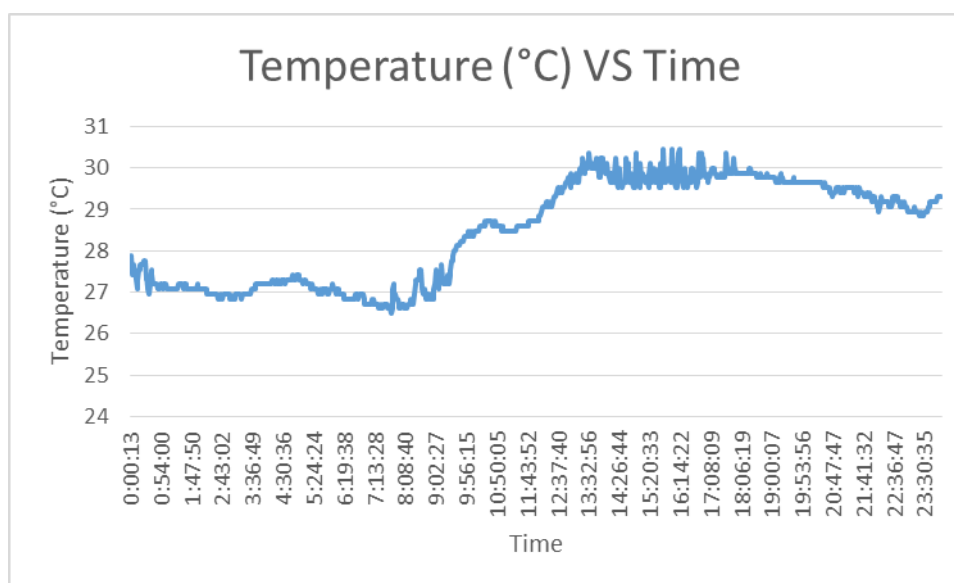


Figure A.2 Temperature vs Time of Router Graph (24th July 2016)

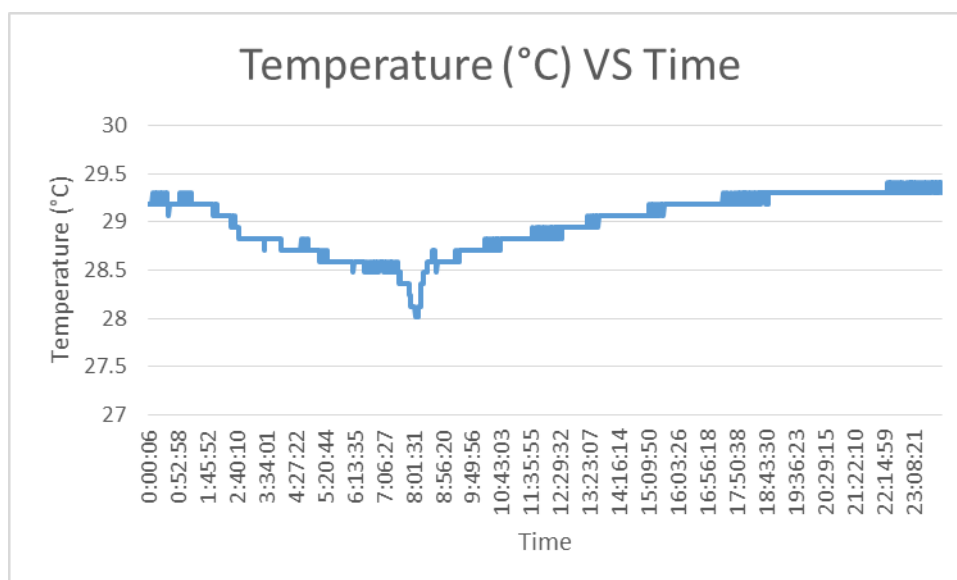


Figure A.3 Temperature vs Time of End Device Graph (25th July 2016)

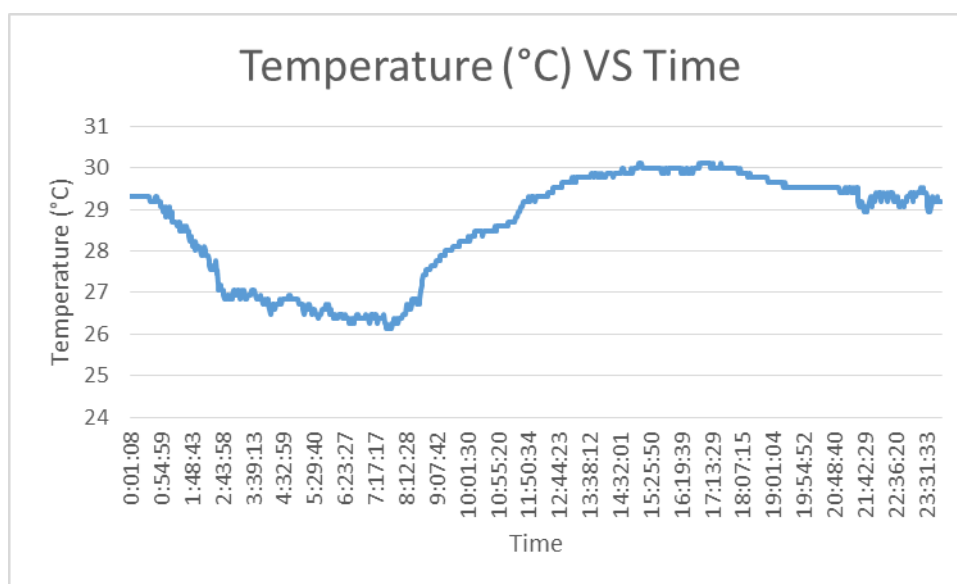


Figure A.4 Temperature vs Time of Router Graph (25th July 2016)

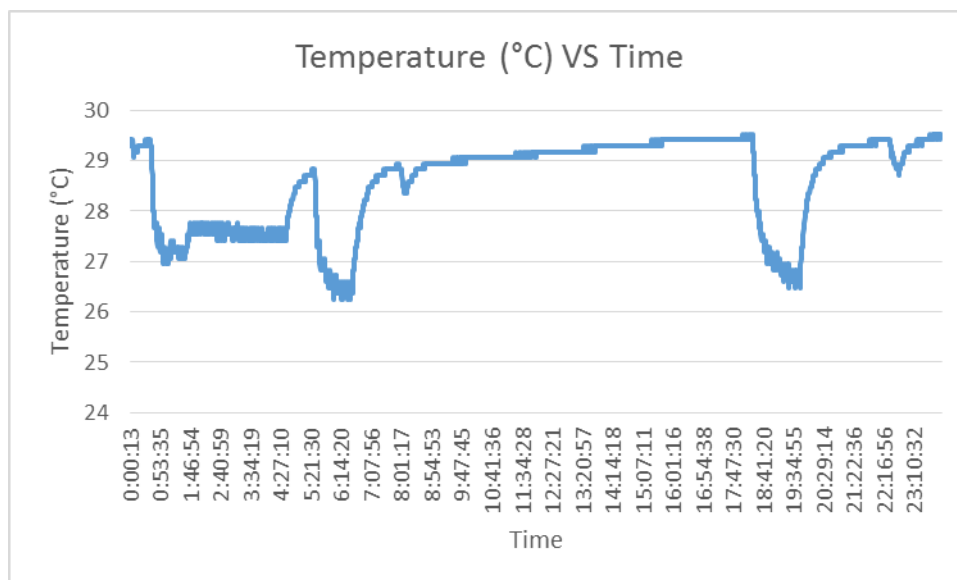


Figure A.5 Temperature vs Time of End Device Graph (26th July 2016)

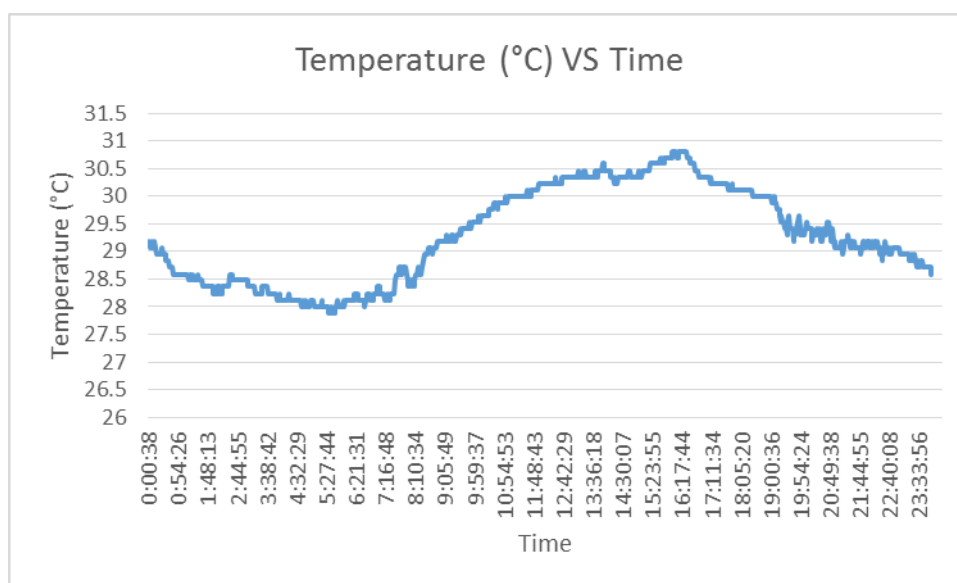


Figure A.6 Temperature vs Time of Router Graph (26th July 2016)

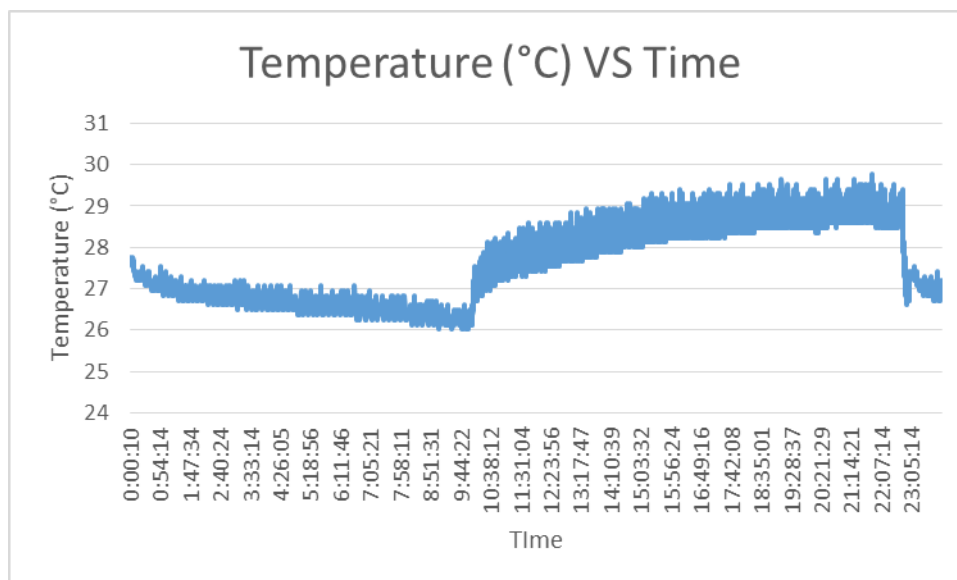


Figure A.7 Temperature vs Time of End Device Graph (31st July 2016)

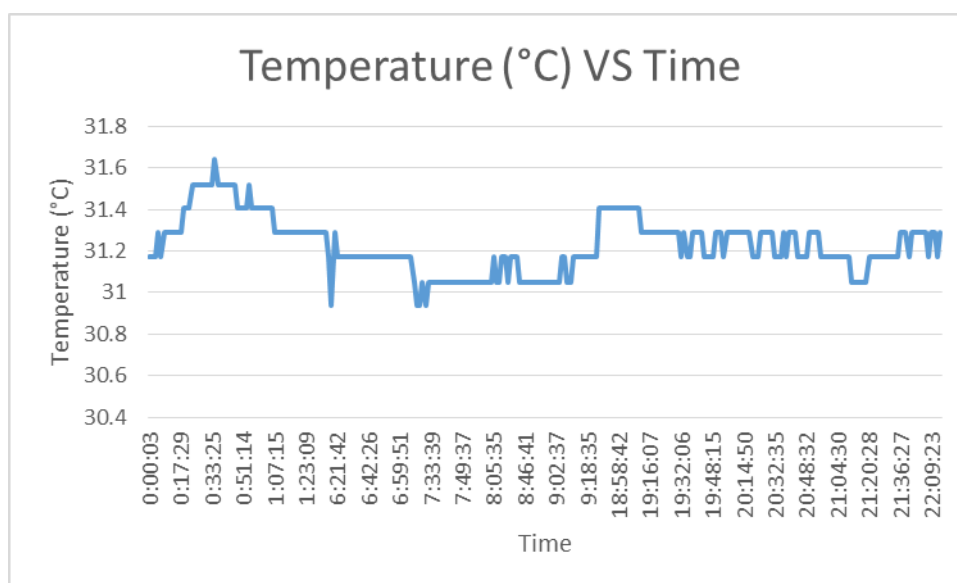


Figure A.8 Temperature vs Time of Router Graph (31st July 2016)

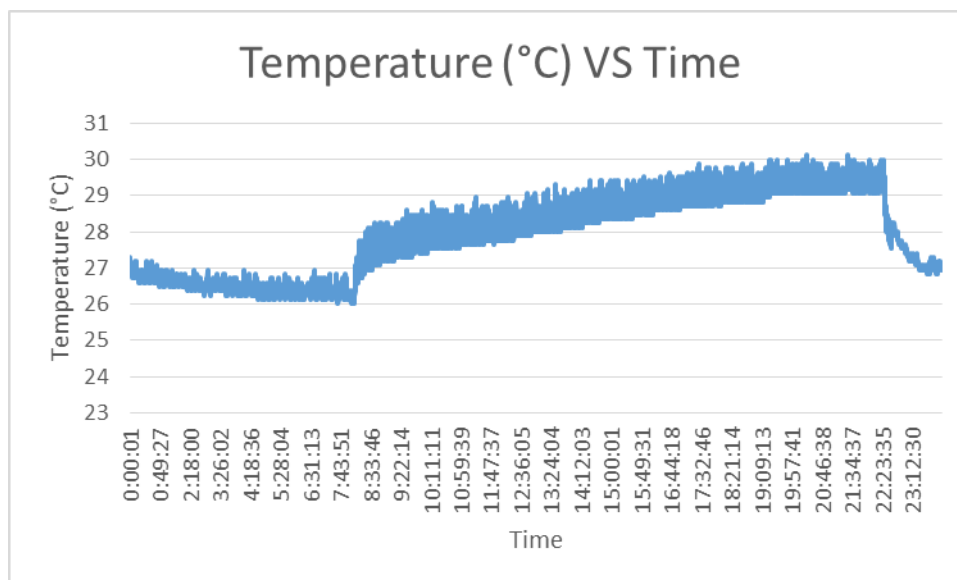


Figure A.9 Temperature vs Time of End Device Graph (1st August 2016)

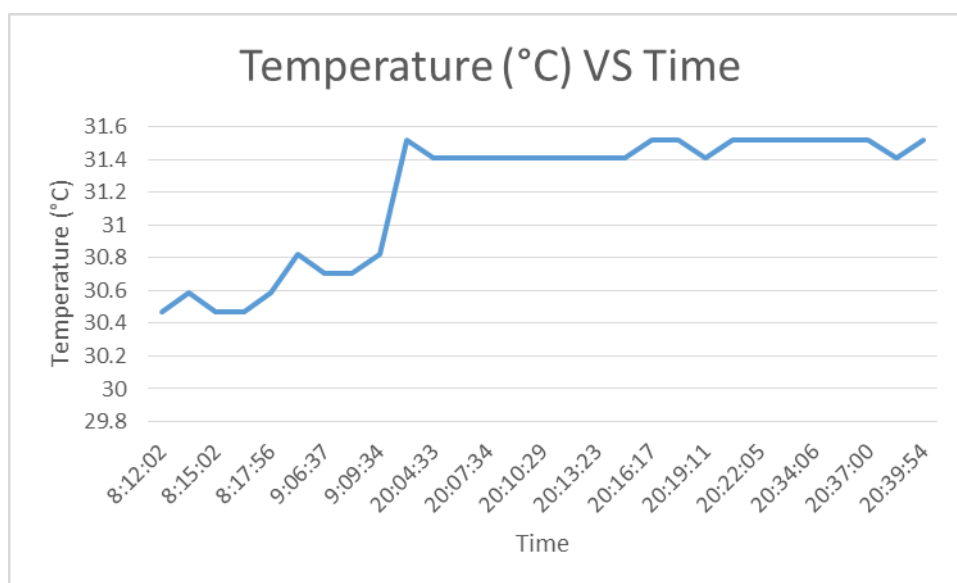


Figure A.10 Temperature vs Time of Router Graph (31st August 2016)

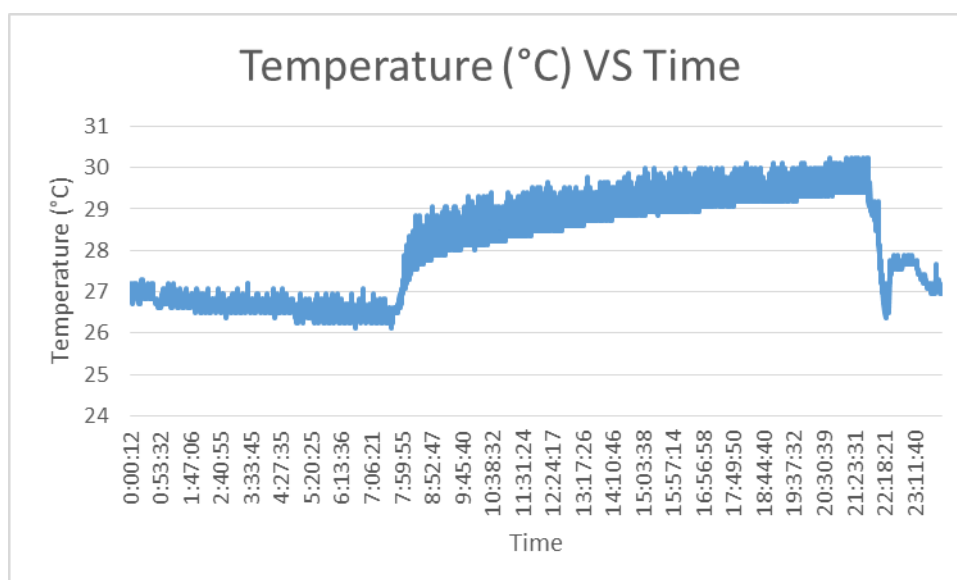


Figure A.11 Temperature vs Time of End Device Graph (2nd August 2016)

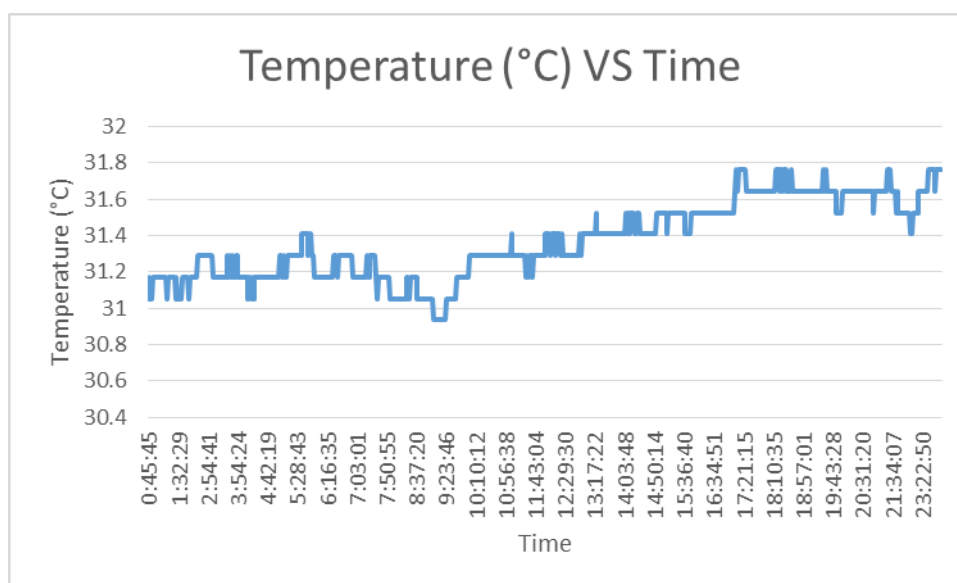


Figure A.12 Temperature vs Time of Router Graph (2nd August 2016)

APPENDIX B: Graphs obtained from Xively

In this section, graph of temperature against time is arranged in the sequence below :

3. Preliminary Result in residential building

- 24th of July 2016 – End Device & Router
- 25th of July 2016 – End Device & Router
- 26th of July 2016 – End Device & Router

4. Second Result in residential building

- 31st of July 2016 – End Device & Router
- 1st of August 2016 – End Device & Router
- 2nd of August 2016 – End Device & Router

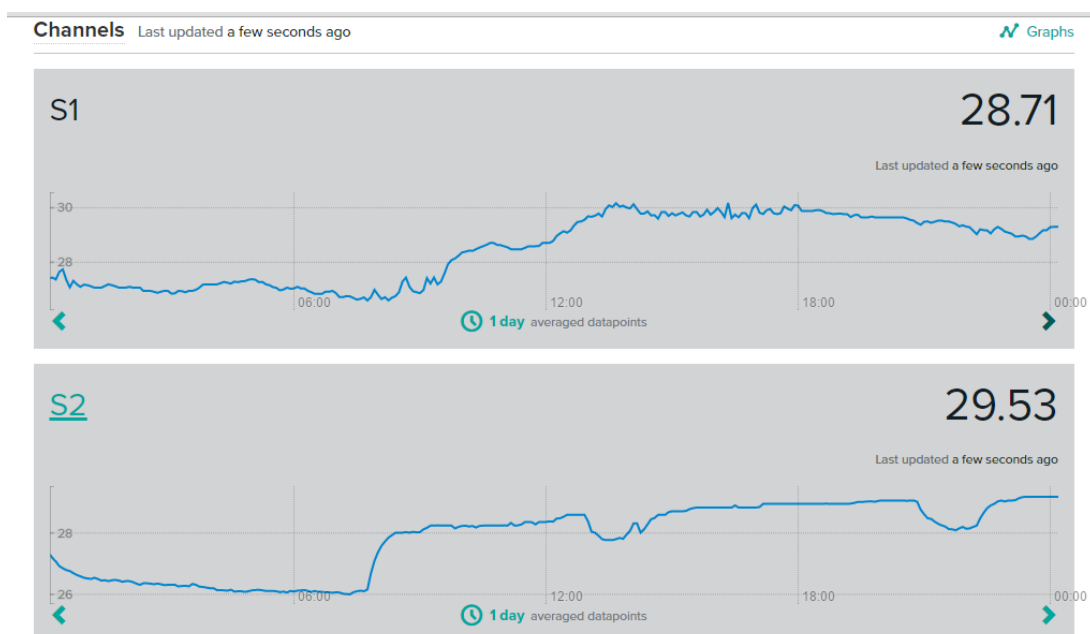


Figure B.1 Temperature vs Time of End Device and Router Graph (24th July 2016)

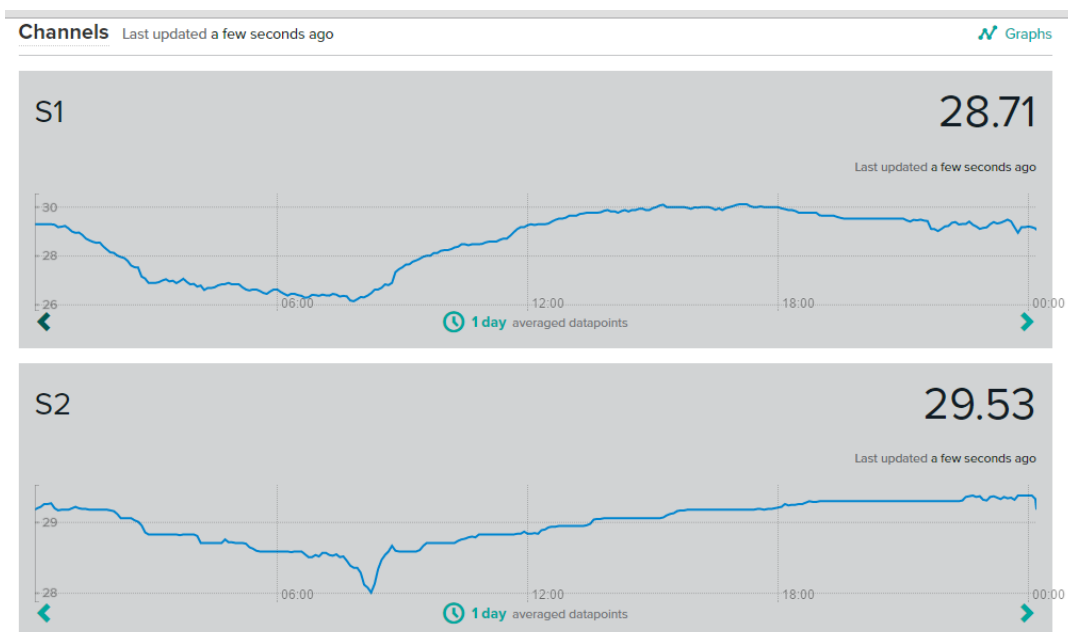


Figure B.2 Temperature vs Time of End Device and Router Graph (25th July 2016)



Figure B.3 Temperature vs Time of End Device and Router Graph (26th July 2016)



Figure B.4 Temperature vs Time of End Device and Router Graph (31st July 2016)

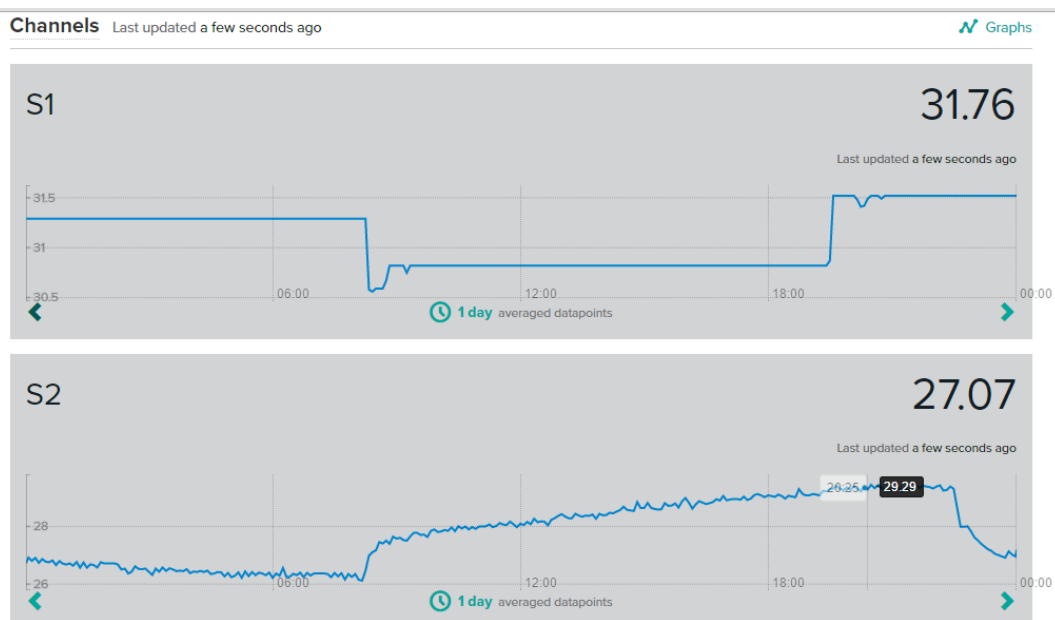


Figure B.5 Temperature vs Time of End Device and Router Graph (1st August 2016)

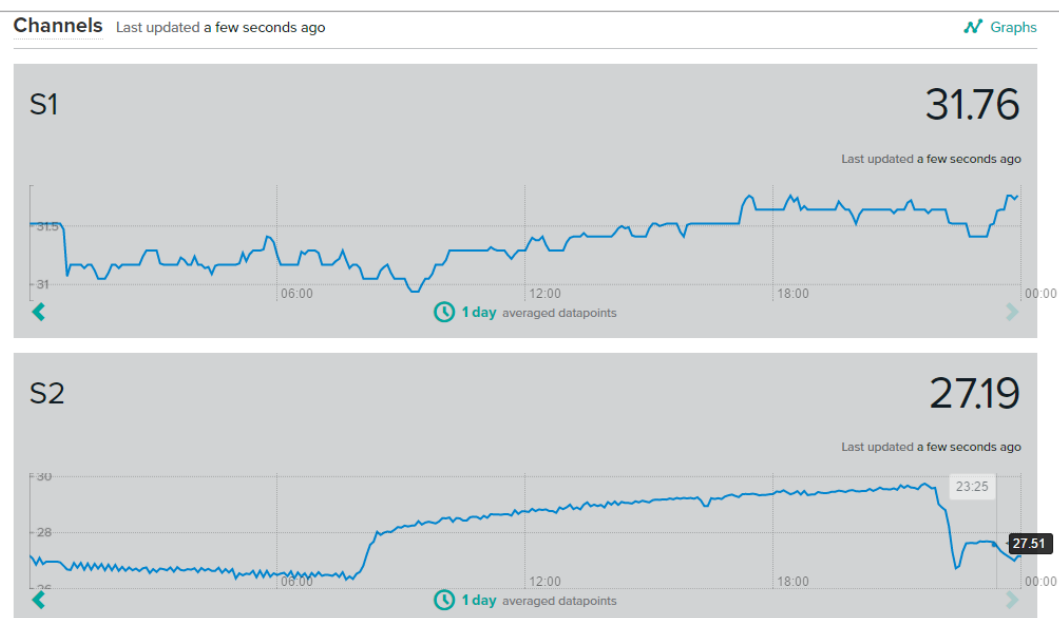


Figure B.6 Temperature vs Time of End Device and Router Graph (2nd August 2016)

APPENDIX C: Computer Programme Listing

The programs code are listed as below.

```
#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <Ethernet.h>
#include <HttpClient.h>
#include <Xively.h>
#include "RTCLib.h"

RTC_DS1307 rtc;

// MAC address for your Ethernet shield
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

// Your Xively key to let you upload data
char xivelyKey[] = "I3sAnej2eqrwDCXCriKIBZCSnYZrwlEGeo62cRIKEOhZiyhs";

// Analog pin which we're monitoring (0 and 1 are used by the Ethernet shield)
//int sensorPin = 2;

// Define the strings for our datastream IDs
char sensorId_1[] = "S1";
char sensorId_2[] = "S2";
```

```
float tempC;
float voltage;
int Voffset = 40;
float divider3V = 3.0;
float divider5V = 5.0;
int ReceiveLED = 13;
int SendLED = 12;
const int button = 7;
int buttonSTATE = 0;
float tempROUTER = 0;
const int SDchip = 4;

XivelyDatastream datastreams[] = {
  XivelyDatastream(sensorId_1, strlen(sensorId_1), DATASTREAM_FLOAT),
  XivelyDatastream(sensorId_2, strlen(sensorId_2), DATASTREAM_FLOAT)
};
// Finally, wrap the datastreams into a feed
XivelyFeed feed(196109937, datastreams, 2 /* number of datastreams */);

EthernetClient client;
XivelyClient xivelyclient(client);

void setup()
{
  pinMode(button, INPUT);

  // Open serial communications and wait for port to open:
  Serial.begin(9600);

  Serial.println("Starting single datastream upload to Xively...");
  Serial.println();

  while (Ethernet.begin(mac) != 1)
```

```
{
  Serial.println("Error getting IP address via DHCP, trying again...");
  delay(15000);
}

/*while (!Serial) {
  ; // wait for serial port to connect. Needed for Leonardo only
}*/
Wire.begin();
rtc.begin();

if (!rtc.isrunning()) {
  Serial.println("RTC is NOT running!");
}

DateTime now = rtc.now();
if ((now.year() != 2016)) {
  rtc.adjust(DateTime(2016, 1, 1, 23, 59, 0));
}

Serial.print("Initializing SD card...");

// see if the card is present and can be initialized:
if (!SD.begin(SDchip)) {
  Serial.println("Card failed, or not present");
  // don't do anything more:
  return;
}
Serial.println("Card initialized");

File dataFile = SD.open("datalog.csv", FILE_WRITE);

// if the file is available, write to it:
```

```

if (dataFile) {
    dataFile.println("Date,Time,Location,Address,,,,Supply voltage (V),Temperature
('C),Motion");
    dataFile.close();
    Serial.println("Title logged");
}
// if the file isn't open, pop up an error:
else {
    Serial.println("error logging title");
}
}

void loop()
{
    if (Serial.available() > 26) {
        if (Serial.read() == 0x7E) {
            digitalWrite(ReceiveLED, HIGH);
            delay(100);
            digitalWrite(ReceiveLED, LOW);

            // Obtain date and time from RTC module and show it in Serial monitor
            DateTime now = rtc.now();
            Serial.print(now.year(), DEC);
            Serial.print("/");
            Serial.print(now.month(), DEC);
            Serial.print("/");
            Serial.print(now.day(), DEC);
            Serial.print(" ");
            Serial.print(now.hour(), DEC);
            Serial.print(":");
            Serial.print(now.minute(), DEC);
            Serial.print(":");
            Serial.print(now.second(), DEC);
            Serial.println();

```

```
delay(10);

for (int i = 0; i < 8; i++) {
    byte discardByte = Serial.read(); // Discard unused byte
    delay(10);
}

// Read only lower 4 byte of source address
int Address_byte_1 = Serial.read();
delay(10);
int Address_byte_2 = Serial.read();
delay(10);
int Address_byte_3 = Serial.read();
delay(10);
int Address_byte_4 = Serial.read();
delay(10);

// Add all the lower 4 bytes of address
// To identify its installed location
int location = Address_byte_1 + Address_byte_2 + Address_byte_3 +
Address_byte_4;
//Serial.print("Location HEX: ");
//Serial.println(location, HEX);
Serial.print("Location: ");
switch (location) {
    case 0x268:
        Serial.println("END DEVICE AT");
        break;
    case 0x1A6:
        Serial.println("ROUTER AT");
        break;
```



```

default:
  Serial.println("UNKNOWN");
  break;
}
delay(10);

// ***** NOTE *****
// If any XBee module is added into the network,
// Comment the lower 4 byte of its source address in HEX below
// for reference
// 1) Router 1 (AT) = 40 79 D0 DF
// 2) End device 1 (AT) = 40 79 D1 1C
// 3) ...
// Check the data send from which XBee module
// Display the source address in Serial Monitor
Serial.print("Address: ");
Serial.print(Address_byte_1, HEX);
Serial.print(" ");
Serial.print(Address_byte_2, HEX);
Serial.print(" ");
Serial.print(Address_byte_3, HEX);
Serial.print(" ");
Serial.println(Address_byte_4, HEX);
delay(10);

for (int j = 0; j < 8; j++) {
  byte discardByte1 = Serial.read(); // Discard unused byte again
  delay(10);
}

// Read digital samples and display out

```

```

// according to different conditions:
// AD1 = digital input
// AD2 = motion sensor state (digital input)
//   has motion = 1 ; no motion = 0
// AD4 = control signal given (digital output)
//   default = 1 ; signal sent = 0;
// AD0 and AD3 are set as analog input, so always show zero
// *** EXAMPLE ***
// AD4 AD3 AD2 AD1 AD0
// 1  0  1  1  0  = 0x16
// Door open, Motion detected, Dout HIGH
int digitalin = Serial.read();

Serial.print("Motion: ");
if (digitalin == 0x16 || digitalin == 0x14 || digitalin == 0x6 || digitalin == 0x4) {
  Serial.println("Detected");
} else {
  Serial.println("");
}
delay(10);

// Read analog samples from pin A0
// Then convert to battery voltage
int analog_1_MSB = Serial.read();
delay(10);
int analog_1_LSB = Serial.read();
delay(10);
int analog_1_reading = analog_1_LSB + (analog_1_MSB * 256);
voltage = (analog_1_reading + Voffset) * 1.2 / 1024.0 * divider5V;
Serial.print("Supply voltage: ");
Serial.print(voltage);
Serial.println(" V");

```

```

// Read analog sample from pin A3
// Then convert to temperature
int analog_4_MSB = Serial.read();
delay(10);
int analog_4_LSB = Serial.read();
delay(10);
int analog_4_reading = analog_4_LSB + (analog_4_MSB * 256);
tempC = (analog_4_reading) * 1.2 / 1024.0 * 100.0;
Serial.print("Temperature: ");
if (location == 0x268) {
  if (tempC >= 14.0 && tempC <= 40.0) {
    Serial.print(tempC);
  }
} else {
  Serial.print(tempC);
}
Serial.println(" ^C");

// Store temperature value of router for sending control signal automatically
if (location == 0x1A6) {
  tempROUTER = tempC;
}

/*****data streaming
coding*****/
if (!(tempC >= 14.0 && tempC <= 40.0))
{
}
else
{
  if (location == 0x268)
  {
    datastreams[0].setFloat(tempC);

```

```

Serial.print("Read sensor value ");

Serial.println(datastreams[0].getFloat());
Serial.println("Uploading it to Xively");
int ret = xivelyclient.put(feed, xivelyKey);
Serial.print("xivelyclient.put returned ");
Serial.println(ret);
}

else if (location == 0x1A6)
{ datastreams[1].setFloat(tempROUTER);
  Serial.print("Read sensor value ");
  Serial.println(datastreams[1].getFloat());
  Serial.println("Uploading it to Xively");
int ret = xivelyclient.put(feed, xivelyKey);
Serial.print("xivelyclient.put returned ");
Serial.println(ret);
}
Serial.println();
delay(15);
}

// ***** DATA LOGGING SESSION *****
*****
File datetimedlog = SD.open("datalog.csv", FILE_WRITE);

// if the file is available, log time
if (datetimedlog) {
  datetimedlog.print(now.year(), DEC);
  datetimedlog.print("/");
  datetimedlog.print(now.month(), DEC);
  datetimedlog.print("/");
  datetimedlog.print(now.day(), DEC);
}

```

```
    datetimelog.print(",");
    datetimelog.print(now.hour(), DEC);
    datetimelog.print(':');
    datetimelog.print(now.minute(), DEC);
    datetimelog.print(':');
    datetimelog.print(now.second(), DEC);
    datetimelog.print(",");
    datetimelog.close();
}
// if the file isn't open, pop up an error:
else {
    Serial.println("error logging date and time");
}
delay(10);

// Log location
locationlog(location);
delay(10);

// Log address
addresslog(Address_byte_1);
addresslog(Address_byte_2);
addresslog(Address_byte_3);
addresslog(Address_byte_4);
delay(10);

// Log voltage
voltage(voltage);
delay(10);
```

```
// Log temperature
temperaturelog(tempC);
delay(10);

// Log motion
motionlog(digitalin);
delay(10);

// ***** Sending control signal from COORDINATOR *****
Serial.println();
delay(10);

buttonSTATE = digitalRead(button);
if (buttonSTATE == HIGH || tempROUTER > 35.0) {
    digitalWrite(SendLED, HIGH);
    setRemoteState(0x5);
} else {
    digitalWrite(SendLED, LOW);
    setRemoteState(0x4);
}
Serial.println("");
Serial.println("");
Serial.println("");
delay(1000);
}
}
}

void setRemoteState(char value) {
    Serial.write(0x7E);
    Serial.write((byte) 0x0);
```

```

Serial.write((byte) 0x10);
Serial.write((byte) 0x17);
Serial.write((byte) 0x0);
Serial.write((byte) 0x0); //64 bit destination address
Serial.write((byte) 0x13);
Serial.write((byte) 0xA2);
Serial.write((byte) 0x0);
Serial.write((byte) 0x40);
Serial.write((byte) 0x79);
Serial.write(0xD1);
Serial.write(0x1C);

Serial.write(0xFF);
Serial.write(0xFE);

Serial.write(0x02);

Serial.write('D');
Serial.write('4');

Serial.write(value);

long sum = 0x17 + 0x13 + 0xA2 + 0x40 + 0x79 + 0xD1 + 0x1C + 0xFF + 0xFE +
0x02 + 'D' + '4' + value;
Serial.write( 0xFF - (sum & 0xFF) );
}

void addresslog(int adr) {
  File adrlog = SD.open("datalog.csv", FILE_WRITE);

  // if the file is available, write to it:
  if (adrlog) {
    adrlog.print(adr, HEX);
  }
}

```

```
    adrlog.print(", ");
    adrlog.close();
}
// if the file isn't open, pop up an error:
else {
    Serial.println("error logging address");
}
}

void locationlog(int loc) {
    File loclog = SD.open("datalog.csv", FILE_WRITE);

    // if the file is available, write to it:
    if (loclog) {
        switch (loc) {
            case 0x268:
                loclog.print("END DEVICE AT");
                break;
            case 0x1A6:
                loclog.print("ROUTER AT");
                break;
            default:
                loclog.print("UNKNOWN");
                break;
        }
        loclog.print(",");
        loclog.close();
    }
    // if the file isn't open, pop up an error:
    else {
        Serial.println("error logging location");
    }
}
```



```
void vtagelog(float volt) {
  File voltlog = SD.open("datalog.csv", FILE_WRITE);

  // if the file is available, write to it:
  if (voltlog) {
    voltlog.print(volt);
    voltlog.print(",");
    voltlog.close();
  }
  // if the file isn't open, pop up an error:
  else {
    Serial.println("error logging voltage");
  }
}

void temperaturelog(float temp) {
  File templog = SD.open("datalog.csv", FILE_WRITE);

  // if the file is available, write to it:
  if (templog) {
    if (temp >= 14.0 && temp <= 38.0) {
      templog.print(temp);
    } else {
      templog.print("X ");
      templog.print(temp);
      templog.print(" X");
    }
    templog.print(",");
    templog.close();
  }
}
```

```
// if the file isn't open, pop up an error:
else {
  Serial.println("error logging temperature");
}
}

void motionlog(int motion) {
  File motlog = SD.open("datalog.csv", FILE_WRITE);

  // if the file is available, write to it:
  if (motlog) {
    if (motion == 0x14 || motion == 0x16 || motion == 0x4 || motion == 0x6) {
      motlog.print("Detected");
    } else {
      motlog.print(" ");
    }
    motlog.println(",");
    motlog.close();
  }
  // if the file isn't open, pop up an error:
  else {
    Serial.println("error logging motion");
  }
}
```