Master's Project 2015
SIV-12-15

Idachaba Emmanuel (142938)
Egbunu Achema (142773)

Home Automation with Windows 10 IoT

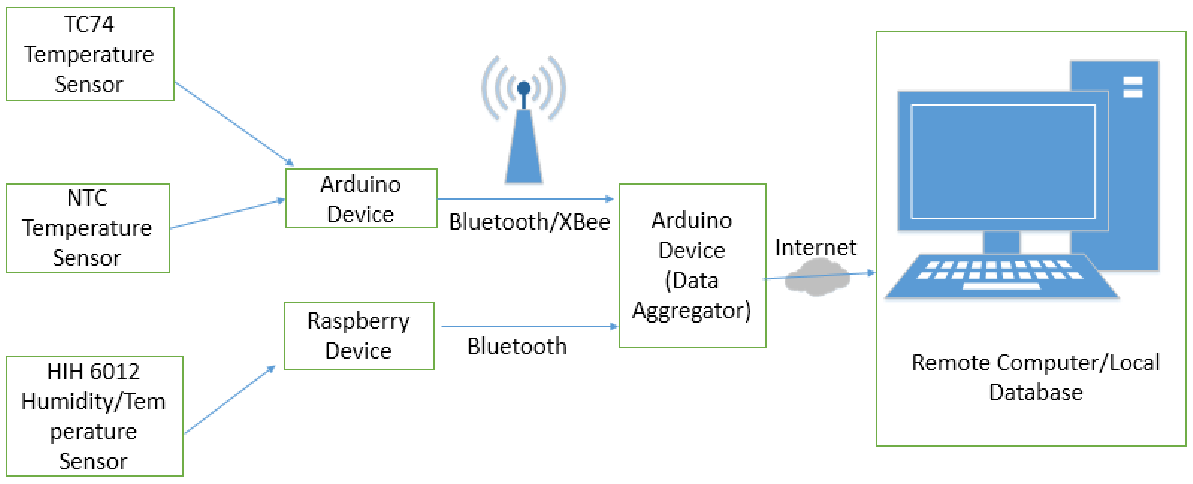


Figure 1-1: Overview of the Home Automation system

Telemark University College

Faculty of Technology

Kjølnes ring 56

3918 Porsgrunn

Version 0.9

http://www.hit.no

Egbunu Achema & Idachaba Emmanuel

Telemark University College

Faculty of Technology

M.Sc. Programme

PROJECT REPORT, COURSE CODE SCE4006

Students:	**Egbunu Achema (142773) & Idachaba Emmanuel** (142938)		
Project title:	**Home Automation Monitoring and with Windows 10 IoT**		
Signatures:	. . . .		
Number of pages:	102		
Supervisor:	Hans-Petter Halvorsen	sign.:	. . . .
2nd Supervisor:	Nils-Olav Skeie	sign.:	. . . .
Censor:	<name>	sign.:	. . . .
External partner:	<name>	sign.:	. . . .
Availability:	<Open/Secret>		

Archive approval (supervisor signature)**:** sign.: . . . . **Date :** . . . .

Abstract:

The home automation has becomes so vital due to the comfort and flexibility user derives from it. The design and implementation of this project is based on the use of Arduino and Raspberry Pi for reading temperature and humidity values, logging locally and remote data database, and the overall system management over internet.

They Arduino and Raspberry devices are located at different point interfacing with the sensors and collecting the sensor data. The sensor data collected is wirelessly transferred to data aggregator system which is an Arduino UNO device through XBee and Bluetooth wireless communication.

The data aggregator is directly connected to Wi-Fi (Router) where the sensor data is logged to the local database (SD Card) and remote database. The user can access these data from anywhere using smart phones, iPad or computer.

The overall management of the Home Automation system is carried out either remotely or locally using Tablet or Mini PC. For the remote management, the user can monitor the data logged to the data over the internet using Tablet or any other devices. The project focus on developing a reliable, effective and robust system of using Arduino and Raspberry Pi that will enables the user to monitor temperature and humidity data logged to the local database and remote database from anywhere.

Contents

1 Introduction

This project work focuses on automation of monitoring and Logging of sensor data to a local memory location on a Next Unit of Computing (NUC) device and to a database in a remote server via Wi-Fi connection. It combines the functionalities of an Arduino board and a Raspberry Pi 2 for monitoring and logging data.

In this project, we will be using an XBee, Wi-Fi and Bluetooth wireless module based on IEEE 802.15.4 , IEEE 802.11, and IEEE 802.15.1 networking protocol respectively in transmitting the data from the Arduino and Raspberry development board mounted with sensor to a remote server. The choice of the XBee and Bluetooth were made in order to benefit from their functionality low energy use, low power consumption, support data rates up to 115200bps which is pretty fast and it is relatively cheap. While Arduino UNO Wi-Fi shield will be used in order to provide wireless communication between the sensors connected to the Raspberry Pi 2 and Arduino and the remote server. The use of wireless communication will reduce greatly the use of communication wires that can make communication system messy.

It is observed from Figure 1-1 that the design of the smart home involves monitoring temperature data by the sensors on Arduino and Raspberry development board before logging to the remote computer/local database through Bluetooth LE, Wi-Fi and XBee.

1.1 Home Automation

Home Automation refers to the automatic and electronic control of household features, activity, and appliances. It can also be seen as the use of one or more computers to control basic home functions and features automatically and sometimes remotely [1]. This is made possible by making use of the data logged to a remote server from the monitoring sensors.

1.1.1 Existing Home Automation

There are various Home Automation system in existence because the desire to achieve a smart home is always on the rise as a result of technology advancement. The term "smart house" was first coined by the American Association of Housebuilders in 1984 and the first “wired homes” were built by hobbyist in the early 1960s [2].

Different professional solutions to home automation have been proffered by many giant bodies such as Samsung, Google and Apple HomeKit etc.

1.1.1.1 Apple HomeKit

HomeKit was meant to improve the experience and offer a secure method for managing accessories to the users. It is an app available on Apple store. The Home app is a user interface for controlling HomeKit-enabled products. It provides users with many monitoring and controlling functionalities about home, zone and room. HomeKit is a framework for communicating with and controlling connected accessories in a user's home[3]. It uses Wi-Fi and Bluetooth Low Energy (BLE), and can be integrated with Apple's phones, tablets and Apple Watch. The user has a choice of adding devices that needs to be controlled. It is activated using Siri[1] option in the app's menu. Within the app's setting page, the home, zone and the room can be given a unique name and the devices that need to be controlled are identified. And the controlling of the added items can commence with series of commands by simply pressing a button to give an order and see the order carried out immediately. This is done through wireless communication. This functionality can also be extended to a longer distance, meaning distance cannot be a barrier to what can be controlled. To control appliances from outside the room, home or zone, the user will need an Apple TV (3rd generation or later) that shares the same iCloud account as their other iOS devices or using the Lutron app.

The Apple HomeKit is still undergoing development with good and improved upgrade still been added. Some challenges of voice recognition by Siri is noticed if the specific HomeKit voice command protocol laid down by Apple is not followed. However, simple commends as "turn off my lights", "turn on the heater" are recognized without problem. Figure 1-1[3] shows various appliances that can be connected to the HomeKit.

[1] Siri: It is a computer program that works as an intelligent personal assistant and knowledge navigator.

Figure 1-1: Various appliances on Apple HomeKit [3]

The functionalities are limited on iOS8 but iOS9 has an improved and better functionalities and support for more compatible accessories.

1.1.1.2 Google Brillo

Brillo is an Operating System (OS) developed based on Android for low power Internet of Things (IoT) devices. It is “lighter” in terms of storage and RAM (at least 128MB storage and 64 or 32MB RAM) required since it runs on low powered devices. This makes it possible to be used in controlling home devices. The birth of Brillo is the result of the synergy between Google and Nest. Brillo will be ran through a centralized console and will support Bluetooth Low Energy (BLE) and Wi-Fi for connectivity [4]. Brillo uses Weave command language communication protocol in talking with devices. Weave has cross-platform protocol functionality that enables device setup from a mobile phone, communication between devices and devices to the cloud, and user interaction from their mobile devices, the web or any other IOT OSes. Any device running Brillo and Weave will communicate effectively with other Android devices, this gives Brillo a very wide coverage and wider use.

Communication between the devices is not limited within the home or the range of coverage of the Wi-Fi alone or Bluetooth LE, devices that run on Weave can communicate through the cloud service from anywhere because devices that run on Weave protocol are automatically connected to the cloud service.

Figure 1-2 [5] shows a typical connectivity network of Brillo app to different devices.

Figure 1-2: Google Brillo Connectivity [5]

1.1.1.3 Samsung Smart Home

Samsung is the pioneer of Home Automation and its home automation solution has a very wide coverage in terms of devices that can be connected. It connects any device at home irrespective of its brands. Home appliances can be controlled when they are configured through the Samsung smartphone app for Smart Home. Devices connect through a hub device to the smartphone app and sensors are used to get the necessary information. The data obtained are then send to a cloud-based Smart Home Server. Figure 1-3 [6] gives an overview of the Samsung smart home. Samsung Smart Home is supported by SmartThings platform and it uses its Openness.

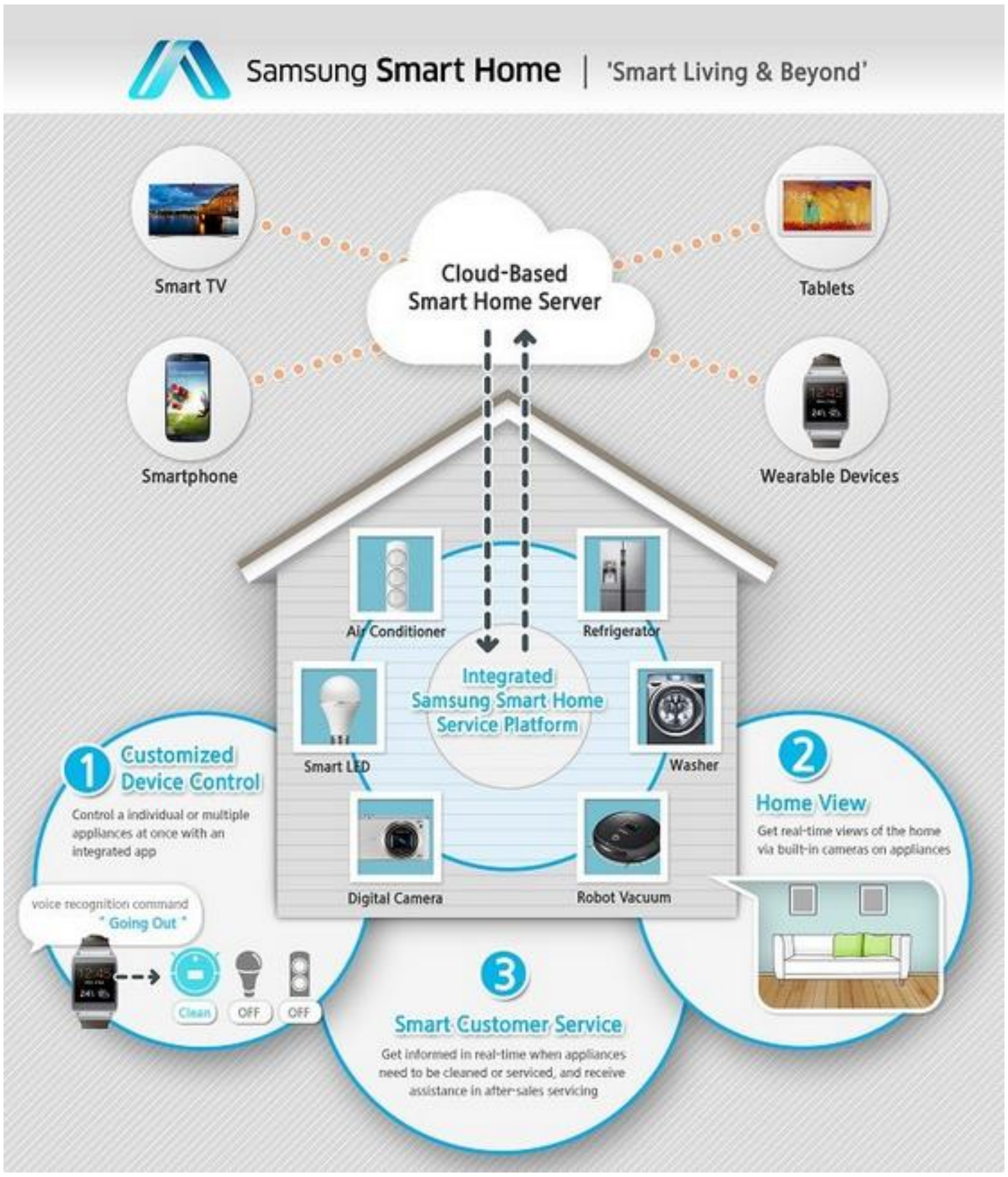

Samsung Smart Home | 'Smart Living & Beyond'
Smart TV
Cloud-Based Smart Home Server
Tablets
Smartphone
Wearable Devices
Air Conditioner
Refrigerator
Integrated Samsung Smart Home Service Platform
Smart LED
Washer
Digital Camera
Robot Vacuum
1 Customized Device Control
Control a individual or multiple appliances at once with an integrated app
voice recognition command " Going Out "
Clean
OFF
OFF
2 Home View
Get real-time views of the home via built-in cameras on appliances
3 Smart Customer Service
Get informed in real-time when appliances need to be cleaned or serviced, and receive assistance in after-sales servicing

Figure 1-3: Appliance that can be controlled by Samsung Smart Home [6]

1.1.2 Smart Home

A home that is completely automated is referred to as a Smart Home. A smart home, or smart house, is a home that incorporates advanced automation systems to provide the inhabitants with sophisticated monitoring and control over the building's functions [7]. For example a smart home may control lighting, temperature, multi-media, security, window and door operations, as well as many other functions. Smart homes use 'home automation' technologies to provide home owners with 'intelligent' feedback and information by monitoring many aspects of a home. Many smart home technologies are available as a result of advancement in computer and technology generally. This has created a challenge of compatibility of the various devices used, therefore there is a drive to standardize home automation technologies.

In this project, we will be using some fundamental components such as computer(s) equipped with the necessary software, Arduino board, Raspberry Pi, various devices to be controlled, sensors (temperature, camera, humidity, photo detection, motion detection), high-speed internet connection, interconnecting cables or wireless links and power source.

Home Automation involves the monitoring and controlling of various appliances such as lighting, security locks on doors and gates, surveillance cameras, Heater, TV set etc. But in this project, attention will majorly be on monitoring and logging of data from Temperature sensor (NTC, TMP-36 and TC74A0), Humidity sensor (HIH6120).

1.1.3 Why Home Automation?

Regardless of the technology used, home automation provides numerous benefits and importance. Outlined are some of the benefits:

1. **It reduces Energy Consumption:** Home automation saves energy on the use of utilities because the issue of forgetting to switch off light or any appliances is eliminated with the use smart appliances as the can be controlled from anywhere.
2. **Security and safety:** Smart Home provides the privilege of Lock and unlock doors from anywhere, control and automate lights from any location, trigger an alarm to ring if there's unwanted motion or entry, and get immediate alerts if doors or windows open unexpectedly. This can be achieved using the following; security camera, entry sensors, smoke detectors and doors and gates locks.
3. **Convenience:** Home automation offers automatic assistance system for able and disabled people who live alone at home as well as old people.
4. **Comfort:** Home Automation offers great deal of comfort as appliances can be controlled from any location. The stress of going to the control switch is eliminated.
5. **Emergency Aids to aged:** Home automation can serve as a rescue/emergency aids to aged people that are in danger and leave alone.

The interesting thing about this project that differs from some of the existing home automation is that it will be implemented on Windows 10 IoT Core which is installed on the Raspberry Pi with various functionalities and features.

Home automation technology has developed so far that the only limit is your level of creativity and imagination.

1.2 Hardware

In this project, the major hardware used are the Arduino and Raspberry Pi 2 development board, and the various temperature sensors

1.2.1 Arduino UNO Device

There are several makes of Arduino device such as Arduino Nano, Arduino Mega, Arduino Mini and Arduino YÙn etc. In this project, Arduino UNO will be used because it has all the functionalities required for the project work.

Arduino UNO has an onboard microcontroller that is based on ATmega328P[2] . *Table 1* [8] shows the technical properties of Arduino UNO device. Figure 1-4 shows an Arduino UNO device.

Table 1: Technical Details of Arduino UNO

Feature	Description
Microcontroller	ATmega328P
Digital I/O Pins	14 (with 6 PWM outputs)
Operating Voltage	5V
Input Voltage (recommended)	7 – 12V
Input Voltage (limit)	6 – 20V
Digital I/O Pins	14 (of which 6 provide PWM output)

[2] ATmega328P: A high performance, low power AVR 8-Bit microcontroller.

PWM Digital I/O Pins	6
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader6
Analog Input Pins	6
DC Current per I/O Pin	20mA
DC Current for 3.3V Pin	50mA
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Figure 1-4 shows an Arduino UNO device drawn from *fritzing* with the different I/Os and connection points.

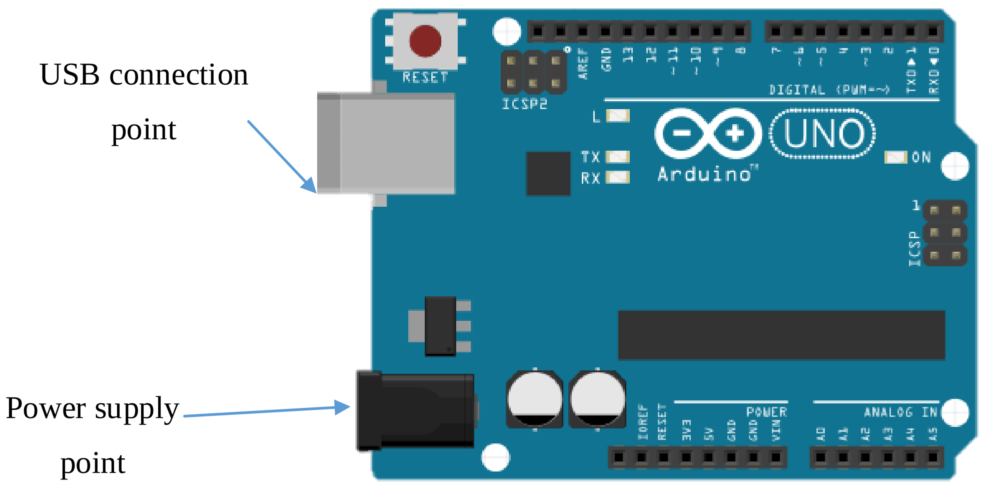


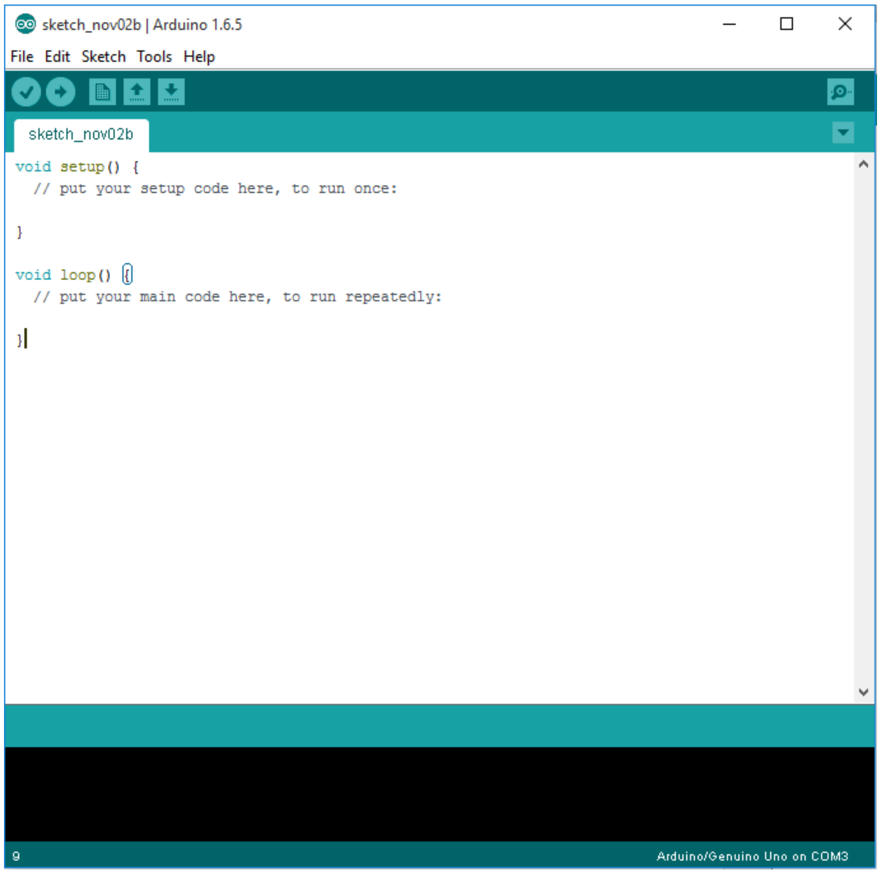
Figure 1-4: Arduino UNO Device

The Arduino board can be powered by three basic ways:

i. Through USB connected to a computer
ii. Using a voltage adapter that converter AC to DC voltage. A DC voltage of 7 – 12V is recommended but it has a limit of 6 – 20V.
iii. It can be powered by a battery.

1.2.1.1 Arduino IDE

The IDE contains the menu bar, tool bar and the editor. The sketch is written in the editor, verified and uploaded to the board. Figure 1-5 shows the Arduino IDE with the default sketch structure.


sketch_nov02b | Arduino 1.6.5
File Edit Sketch Tools Help
sketch_nov02b
9
Arduino/Genuino Uno on COM3

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Figure 1-5: Screen shot of Arduino IDE

1.2.2 Raspberry Pi 2 Device

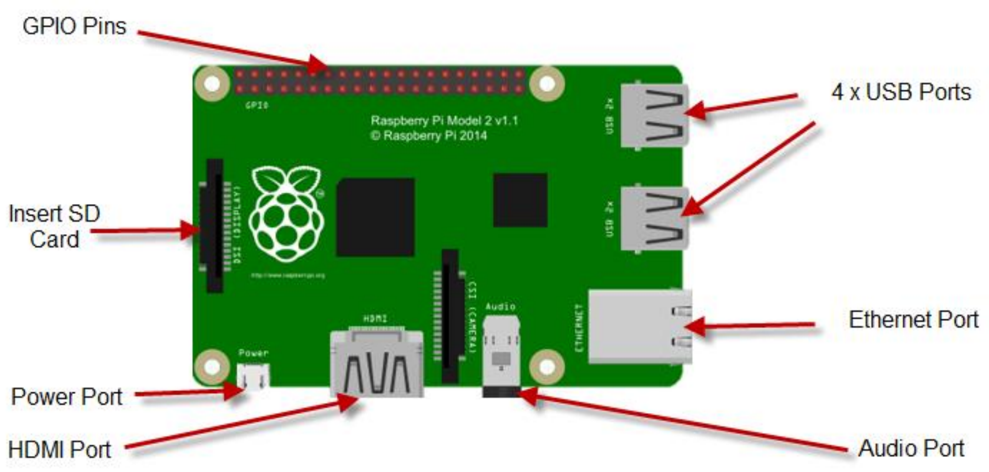
This is the latest make of Raspberry device, it was released in February 2015. It has some added functionalities that are not included in the first one Raspberry Pi 1 Model B+. It is around a BCM2835 system-on-a-chip (SoC) ARMv7 processor running at 700MHz microprocessor, hence it can run the full range of ARM GNU/Linux distributions, including Snappy Ubuntu

Core, as well as Microsoft Windows 10 IoT Core[9]. In order to use Raspberry Pi 2 conveniently, it is very important that the various systematic and sequential steps in setting it up should be followed carefully. The steps are provided at the implementation section. Its technical details are provided in the Table 2.

Table 2: Technical Details of Raspberry Pi 2 Device

Features	Description
CPU	A 900MHz quad-core ARM Cortex-A7
RAM	1GB
USB ports	4
GPIO pins	40
HDMI port	1
Ethernet port	1
Audio jack and composite video	1 Combined 3.5mm
Camera interface (CSI)	1
Micro SD card slot	1
VideoCore IV 3D graphics core	1

Figure 1-6 shows a Raspberry Pi 2 device, picture taken from *fritzing* with the various connection



points.

Figure 1-6: Raspberry Pi 2 Device

Figure 1-7 [10] shows the various GPIO pinout of Raspberry Pi 2

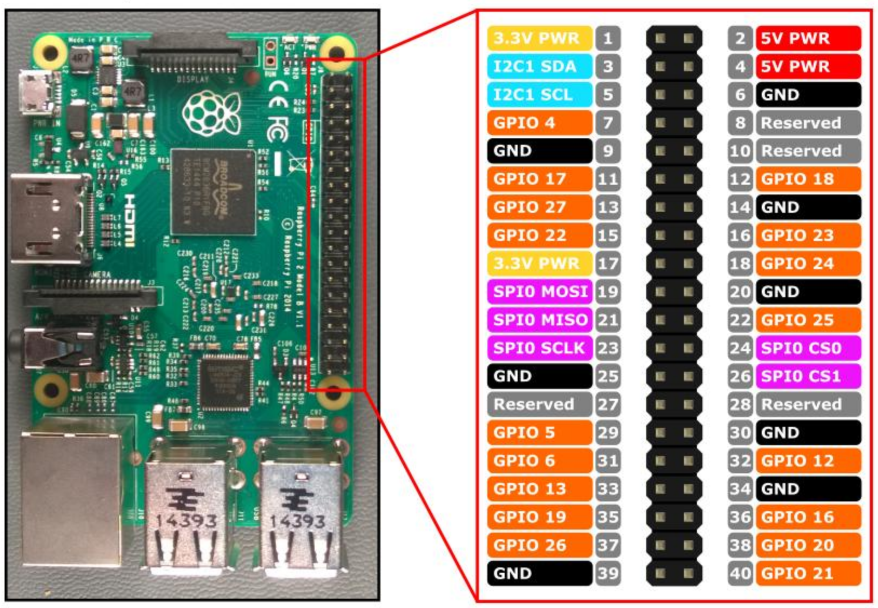


Figure 1-7: Pinout of Raspberry Pi 2 [10]

1.2.3 Why both Arduino and Raspberry Pi 2?

Technically it might sound as a redundant system because the two devices are used to measure the same quantities but the inclusion of the two is necessary because there are some features that are present in one that are not available in the other. Such features are:

i. Arduino uses a microcontroller while Raspberry Pi uses an applications processor which is system on a chip (SoC) makes it possible for it to run an application such as home automation application if developed.
ii. The inclusion of Raspberry Pi makes it possible for the monitored data to be viewed on a screen through the HDMI cable because that of Arduino UNO can be viewed through the serial monitor.

1.2.4 Comparison of Arduino UNO and Raspberry Pi 2 Device

The two devices are very similarly technically and functionally but there exist some differences between them that can influences one choice. Table 3 provides the comparison between Arduino UNO and Raspberry Pi 2 devices.

Table 3: Comparison of Arduino UNO and Raspberry Pi 2 Devices

	Arduino UNO	Raspberry Pi 2
Price	$30	$35
Size	7.6 x 1.9 x 6.5 cm	8.6 x 5.4 x 1.7 cm
Memory	0.002MB	512MB
Clock Speed	16MHz	700MHz
Onboard Network	None	10/100 wired Ethernet RJ45
Multitasking	No	Yes
Input Voltage	7 – 12V	5V
Flash	32KB	SD Card (2 to 16G)
USB	One, Input Only	Two, Peripherals Ok
Operating System	None	Windows 10 IoT Core
Integrated Development Environment (IDE)	Arduino	Any compatible IDE e.g. Geany, BlueJ, Visual Studio

		etc.

1.2.5 XBee Device

There are various communication modules through which the data monitored can be transferred to the controller Arduino such as wireless, wired, infrared etc. In this project, XBee module is used because of its low power consumption of 50mA @ 3.3V and its high range of reach of 300ft. XBee module is based on Zigbee communication protocol. Other details technical and functional details of XBee can be found at [11]. Figure 1-8 shows an XBee mounted on adapter board.

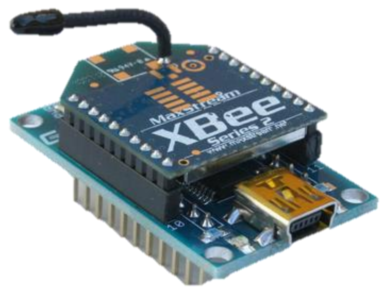


Figure 1-8: XBEE module mounted on Adapter board

1.2.6 Bluetooth Device

The communication between the Raspberry Pi 2 at point 2 to the central device (Arduino device) is through Bluetooth communication protocol. A USB Bluetooth dongle that transmit Bluetooth signal is connected to the Raspberry device while the central device will be mounted with Arduino Bluetooth shield. Figure 1-9 shows the USB Bluetooth dongle that was purchased to be used with the Raspberry Pi 2 in this project while Figure 1-10 shows Bluetooth Smart Shield mounted on Arduino Bluetooth Shield purchased.

Figure 1-9: USB Bluetooth dongle

Figure 1-10: Bluetooth Smart Shield mounted on Arduino Bluetooth Shield

But due to challenge of limited available information about the two devices, we could not use them effectively to achieve the intended purpose, however, order was made for a more efficient, easy to use and cheap Bluetooth module (HC-05). Figure 1-11 shows a HC-05 with the pin assignment. The pins are connected accordingly to the Arduino development board depending on the mode that it is configured as (master or slave mode).

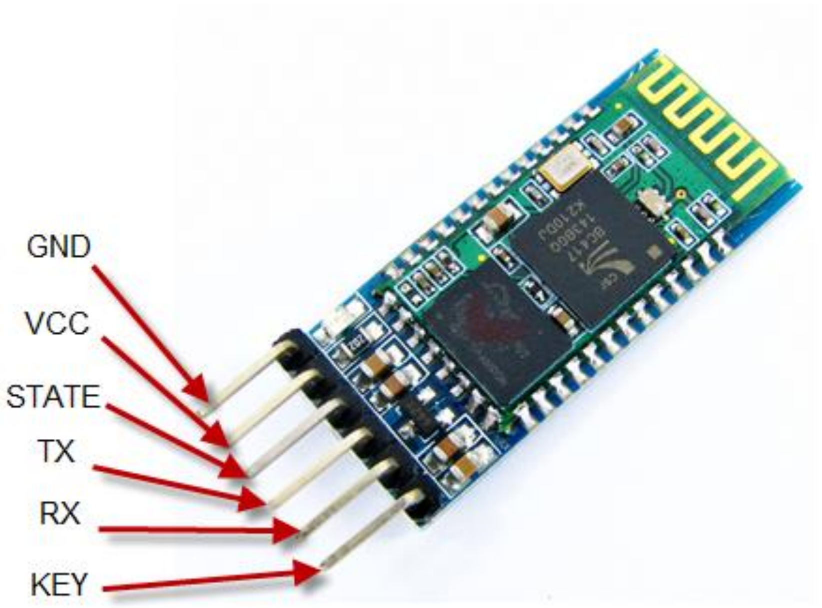


Figure 1-11: HC-05 Bluetooth Module with pin assignment

The data obtained from both point 1 and point 2 are first logged to the SD card on the central Arduino device (known as data aggregator) before logging to the local database on the NUC computer through wireless communication protocol.

The Arduino and Raspberry Pi 2 are connected to XBee or Bluetooth LE module and Raspberry USB Wi-Fi dongle respectively which help in transmitting data from the sensor to the remote server in the control center. Each point also consist of various sensors needed for taking measurements for onward transfer to the control center's Raspberry Pi via wireless internet service. All the readings are then logged to a remote data base through a secured and high speed internet service which can be accessed and control through PDAs[3] (IPad) from any location by an authorized person.

1.2.7 Temperature Sensors

These are devices used in measuring the temperature of a medium. They can be categorized in to contact and contactless sensors based on their proximity to the medium. Also, they can be categorized based on the physical property (resistance, current, voltage, level of liquid) the measured into thermometer, RTDs and thermocouple.

There different types of Temperature sensors based on the physical property the measured (resistance or voltage) for measuring temperature. They are:

1. TMP36 temperature sensor.
2. NTC temperature sensor.
3. PT-100 temperature sensor.
4. Thermocouple

1.2.7.1 TMP36 Temperature Sensor

It is a low cost analog temperature sensor. It works on the principle that the voltage between the Base and the emitter, V_{BE} is linearly proportional to temperature in degree Celsius (℃). It is a solid state component and easy to use.

It has three terminals as shown in Figure 1-12 namely:

i. Collector
ii. Emitter
iii. Base

[3] PDA: Personal Digital Assistance

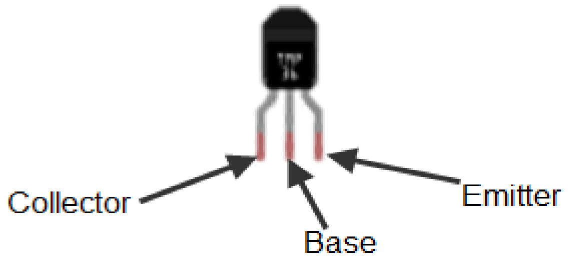


Figure 1-12: TMP36 Temperature Sensor

1.2.7.1.1 Features of TMP36

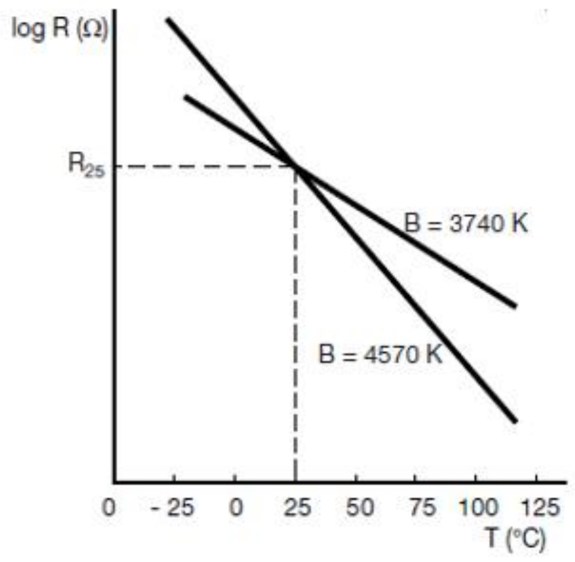
TMP36 has many functional features that make it suitable for temperature measurement.

i. It has a low operational voltage (2.7 V to 5.5 V)
ii. It is calibrated directly in °C.
iii. It has a 10 mV/°C scale factor
iv. It is ±2°C accurate over temperature
v. It has ±0.5°C linearity
vi. Stable with large capacitive loads
vii. It can measure temperature in the range of −40°C to +125°C
viii. It has less than 50 µA quiescent current
ix. Shutdown current 0.5 µA max
x. It is low self-heating

1.2.7.2 NTC Temperature Sensor

A Thermistor is a term coined out of thermal and resistor. Its works based on the principle that its resistance varies with changes in temperature. Technically, all resistors values varies with temperature but thermistors are designed in such a way that their resistance varies drastically with changes in temperature[12]. NTC temperature sensor has 0 ℃ at 100 Ω[13]. Figure 1-13

[14] shows the relationship between the resistance and the



temperature.

Figure 1-13: Temperature – Resistance Relationship

It is observed from Figure 1-13 that relationship has a negative gradient, which buttress the negative proportionality of NTC temperature sensor's resistance with temperature.

There two different types of thermistors:

i. PTC: Positive Temperature Coefficient. It has a positive temperature coefficient i.e. its resistance value increases as the temperatures increases. They are majorly used as resettable fuses because of their positive relationship with temperature. Hence, they are used for protecting circuit.
ii. NTC: these are used for temperature measurement. It has a negative temperature coefficient i.e. its resistance value decreases as the temperatures increases. It has a negative electrical resistance variation with temperature as it is observed in Figure 1-13.

Since the objective of this project is to measure temperature, NTC is used. The nominal value of the thermistor is normally 25 degrees Celsius, in this case we will use a 10K thermistor. It can read temperatures between -40 and +125 degrees Celsius

1.2.7.2.1 Features of NTC Temperature Sensor

NTC has many benefits over other types of temperature sensor [12]. Among the features are:

i. They are also much easier to waterproof since they just a resistor.
ii. They work at any voltage (digital sensors require 3 or 5V logic).
iii. Compared to a thermocouple, they don't require an amplifier to read the minute (very small) voltages - you can use any microcontroller to read a thermistor.
iv. They can also be incredibly accurate for the price. For example, the 10K 1% thermistor in the shop is good for measuring with ±0.25℃ accuracy.
v. They are difficult to break or damage - they are much simpler and more reliable

1.2.8 HIH 6120 Humidity/Temperature Sensor

It is a cheap digital output-type relative humidity and temperature sensors combined together in the same package. It has an accuracy of +/-4.0 %RH and a temperature accuracy level of +/-0.5 ℃. It has I^2C and SPI functionalities which make it possible to integrate with microprocessor. Figure 1-14[15] shows HIH-6120 temperature and humidity sensor with the pinout.

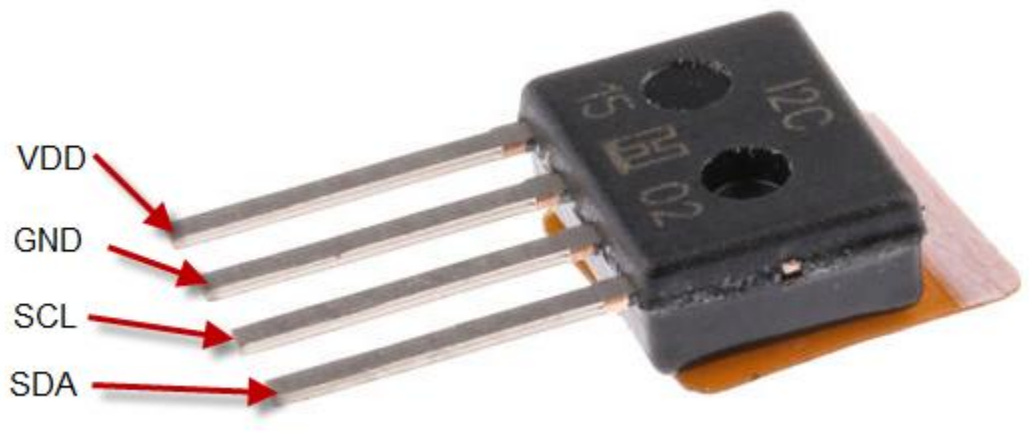


Figure 1-14: HIH-6120 Temperature and Humidity

1.2.8.1 Specification of HIH-6120

The technical specification of HIH-6120 temperature and humidity sensor are:

i. It gives digital output
ii. It has resolution of 14 bit
iii. It has accuracy of +/-4.0 %RH and +/-0.5 ℃.
iv. Minimum operating temperature is -25 ℃.
v. Minimum operating supply voltage is 2.3V
vi. Maximum operating temperature is +85 ℃.

vii. Maximum operating supply voltage 5.5V.
viii. Interface type I2C
ix. Package type SIP

2 Exploring Different Means of Communications for Home Automation Systems

There are different means of communications available for the implementation of Home Automation systems, and in order to be able to choose a viable, robust and reliable one for the implementation of this project, different solutions were analyzed and discussed as shown Figure 2-1.

2.1 Wired (Ethernet Cable) Home Automation System

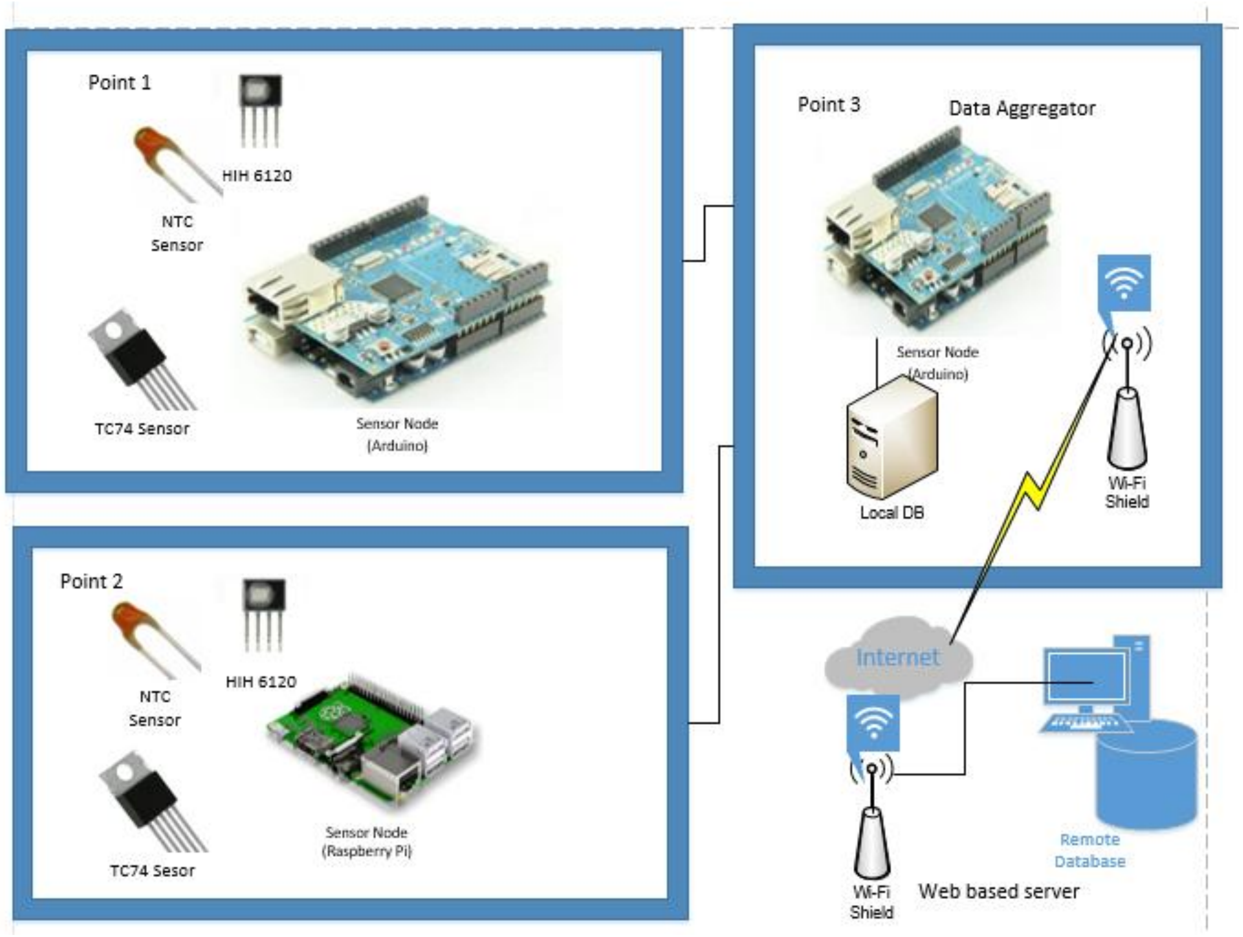


Figure 2-1: Wired Home Automation System using Ethernet Cables

Figure 2-1 shows an overview for the wired means of communication for the Home Automation system. In this topology, each sensor node (Arduino and Raspberry Pi) is connected to the data aggregator (Arduino system) using Ethernet cables (10BASE-T/100BASE-TX). As shown in the figure, the Arduino device has an Ethernet shield that is mounted on it to enable Ethernet connection between point 1 and point 3. The Raspberry Pi on the other hand has an inbuilt ethernet port that enables ethernet communication. The communication protocol between the nodes is based on Ethernet standard, IEEE 802.3 (10BASE-T, up to: 100BASE-TX). And each device can communicate with each other with maximum distance of about 100 meters using Category 5 (CAT5) cable supporting higher bit rate of about 100Mbps and longer link distances [16].

The data transmission between the nodes is divided into shorter pieces called frames and each frame contains the source and destination addresses with error checking to enables damage frames to be detected, discarded and retransmitted. The Ethernet communication between the nodes also provides services up to and including the data link layer as per the OSI model.

Twisted pair Ethernet cabling is used between the nodes for the purpose of electromagnetic interference cancellation from external sources. The color code for the twisted pair Ethernet cable for both ends is shown in Figure 2-2.

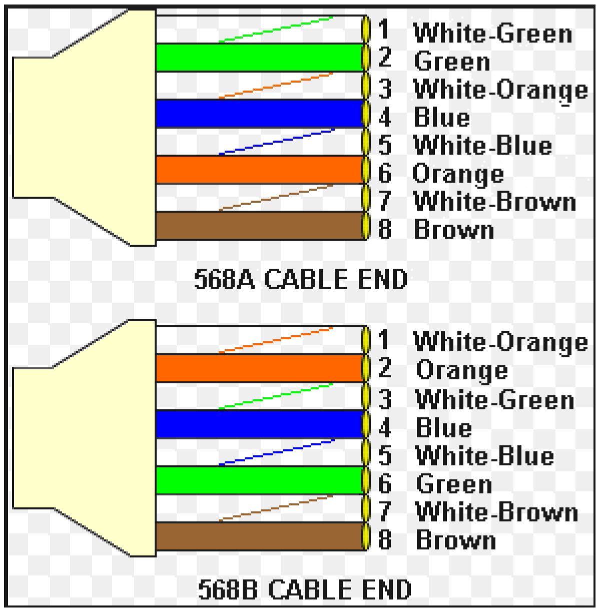


Figure 2-2: Color Codes for the Twisted Pair Ethernet Cable [17]

Figure 2-2 shows the color codes for the twisted cable that can be used for the communication between the nodes in Figure 2-1. Both ends of the ethernet cable are crimped into RJ-45 pins using 568A and 568B configuration respectively. The Wi-Fi shield on the data aggregator (point

3) connects directly to the router through which the sensor data is logged to the remote data base, DB over the internet.

2.1.1 Advantages

The advantage of this solution is that using ethernet cable as a connection medium between the sensor nodes and the data aggregator node provides a reliable communication channel without any fluctuations in the communication link. High data transmission rate can also be maintained between the nodes.

2.1.2 Disadvantages

For long distance connection the cable resistance can offer a significant setback as the link tends to attenuates. Cost of purchasing the cables is expensive. Furthermore, laying of cables is also labor intensive and does not make the house neat.

Table 2-1: Specifications for the Wired Solution [16]

Cable Type	Connector Type	Maximum Speed (Mbps)	Maximum Distance (meter)	Communication Protocol
Ethernet Cable (CAT-5)	RJ-45	100	100	Ethernet

2.2 Wireless Home Automation Systems

The wireless Home Automation systems that are discussed here are based on the Wi-Fi, Bluetooth and ZigBee communications. Figure 2-3 shows the different wireless means of communication between the sensor nodes (point 1 and point 2) and the data aggregator (point 3). In this figure different means of communications between the sensor nodes and the data aggregator are incorporated together and discussed separately.

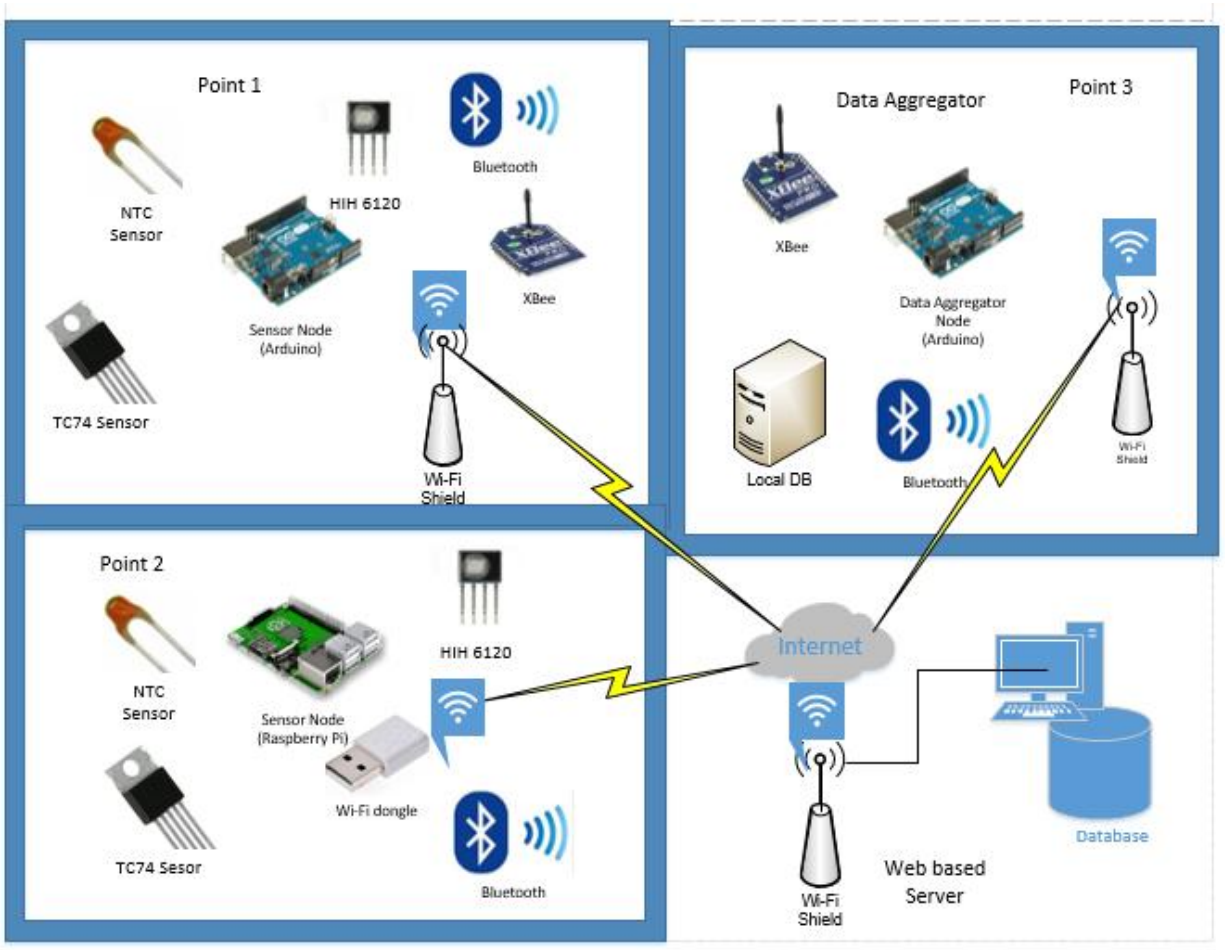


Figure 2-3: Different Wireless Means of Communication between the Sensor Nodes (point 1 and point 2) and the data Aggregator

2.2.1 Wi-Fi Communication

Error! Reference source not found. shows the Wi-Fi solution for Home Automation system. This solution provides Wi-Fi communication medium between the sensor nodes (Arduino and Raspberry Pi) and the remote database, DB. Since the communication is Wi-Fi, there is no need for the third device which is the data aggregator. Each sensor nodes is connected to the router using Wi-Fi communication protocol through which the data is logged to remote database. The Wi-Fi protocol is a medium range wireless communication system with a coverage area of about 100 meters. It operates in frequencies of 2.4GHz and 5GHz [18] which enhance the device to transmit higher data rate of about 54Mbps and the communication protocol is based on IEEE 802.11 standard.

The specifications for the solution are shown in Table 2-2.

Table 2-2: Specifications for the Wi-Fi Communication [18]

RF Output Power (dbm)	Maximum Coverage (meter)	Operating Frequency (Hz)	Maximum Speed (Mbps)	Communication Protocol	Power Consumption (mA)	Supported Security Protocols	Wi-Fi Shield Dimension (mm)
8dBm ± 1dBm	100	2.4 - 2.497 GHz	54	Wi-Fi (IEEE 802.11b/g/n)	TX: 135.0 RX: 125.0	WEP, WPA/WPA2 –PSK	59 x 54

2.2.1.1 Advantages

The advantage of this means of communication is that it is flexible, easier to add new device(s) and less labor intensive.

2.2.1.2 Disadvantages

The main disadvantage of this means of communication is interference and bandwidth issues. If the house is full of Wi-Fi connected gadgets then the Wi-Fi device will competes with the other devices for bandwidth and will be slower to respond. The interference on the other hand causes fluctuations of signal and affects the quality of the link. Another drawback to this solution is that the Wi-Fi shield consumes a lot of power.

2.2.1.3 Bluetooth Communication

Figure 2-3 also shows the Bluetooth means of communication for the Home Automation system. The communication between the sensor nodes and the data aggregator is based on Bluetooth communication protocol. This enables the user to access the sensor data from any of the node via Bluetooth once within the coverage area without the need to connect through the internet. However, the logging of the data to remote database is based on the Wi-Fi communication protocol between the router and the data aggregator. The data aggregator receives the sensor data (Temperature values) via Bluetooth and logged the value to the remote database using Wi-Fi.

This solution has a communication distance in house up to 10 meters without obstacle with a frequency range of 2.4G. In this solution up to 8 cell nodes can be connected to each other. The specifications for the communication are given in Table 2-3.

Table 2-3: Specifications for the Bluetooth Solution [19]

TX Power (dBm)	RX sensitivity (dBm)	Coverage Distance (meter)	Operating Frequency (Hz)	Communication Protocol/ Interface	Antena (dBi)	Baudrate:	Bluetooth Shield Dimension (mm)
[-27 to +3]	-90	10	2.4 GHz	Bluetooth V2.0	2	9600/9200 /38400 etc	57.4x45.3x19.4

2.2.1.4 Advantages

The chief advantage here is that the user can use a smart phone/Tablet to access sensor data using both Bluetooth and Wi-Fi communication. The solution is suitable for low power consumption project(s) and low data rate.

2.2.1.5 Disadvantages

The main disadvantages of this solution are the limitation in distance and power consumption.

2.2.2 ZigBee Communication (XBee)

Figure 2-3 shows the network topology for XBee. The communication between the sensor nodes and the data aggregator is based on Zigbee protocol using XBee communication device. This protocol is designed to communicate under Personal Area Network with data rate of about 250kbps. The data aggregator received the sensor values from the sensor nodes via ZigBee communication protocol and logged the value to the remote database using Wi-Fi. The communication is based on mesh topology with each device communicating with each other and through each other to the destination and can support up to 65000 cells nodes [20].

This solution is robust and provides a stable network due to the mesh communication method implemented by the protocol. If any of the neighboring nodes is faulty, the faulty or damaged neighboring nodes will create a reverting loop, preventing any information or data to be sent to it. This is known as self-healing. There are three different types of nodes can be configured using ZigBee: the ZigBee Coordinator, ZigBee Router, and ZigBee Endpoint. The ZigBee router transmits the data to the coordinator where the data is transmitted to the data aggregator, then to the local database and remote database using Wi-Fi.

2.2.2.1 Advantages

The means of communication is more efficient and consumes less energy when compare to Bluetooth. It is also self-healing network and supports multiple nodes.

2.2.2.2 Disadvantages

It cannot be utilized for a network that requires higher data rate.

2.3 Comparison between the Four Different Solutions

Based on the analysis of the various means of communication, Table 2-4 shows the summary of the comparison among the different solutions discussed.

Table 2-4: Comparison between Wired, Wi-Fi, Bluetooth and Xbee Home Automation Solution

Wired (Ethernet)	Wi-Fi (Wi-Fi Shield)	Bluetooth (Bluetooth Shield)	ZIGBEE (XBee)
• High data rate(100Mbps) • Required high power consumption • Can connect to the internet using Ethernet cable connected to	• High data rate (54Mbps) • Required high power consumption • Used to connect directly to the internet • Suitable for sensor networks due to high data	• Very low data rate(1Mbps) • Required very low power consumption • Cannot connect to the internet directly • Due to low data rate, it is not suitable for sensor	• Low data rate (250kbps) • Required low power consumption • Cannot connect to the internet directly • Suitable for

the router • Suitable for sensor networks due to high data rate. • Uses star topology network	rate. • Uses star topology network	networks but good for controlling devices (like switching ON and OFF light from phone) • Uses point to point master-slave network.	sensor networks • Uses mesh topology. Each device communicates with each other and through each other to the destination device.

Based on the analysis of the various means of communication of Home Automation systems discussed above, Bluetooth and XBee wireless communication solutions will be integrated and used for the Home Automation under design.

3 Analysis and Design of Home Automation Systems

3.1 Analysis

This section uses the concepts of Object-Oriented Analysis (OOA) to analyze the software under design. The overview of the phase is shown in Figure 3-1 [21].

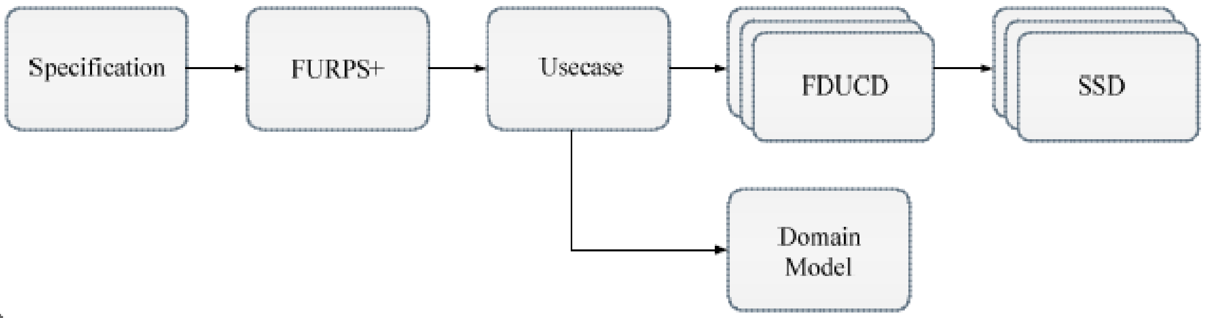


Figure 3-1: Analysis Phase of the Home Automation software development [21].

3.1.1 Requirements

Collection of requirements and specifications were carried out using FURPS+ as shown in Table 3-1.

Table 3-1: List of requirement using FURPS+

F: Functional	• The Home Automation system should get the sensor configuration and type; monitor the temperature and display the values on the screen • It should be able to log the temperature value to the SQL server • Using IIS Web service to monitor the temperature from the SQL database and SD Card
U: Usability	• Display • SQL database • SD card
R: Reliability	• Running system all the time and logging temperature value data

	anytime. • Temperature range -30℃ – 100℃.
P: Performance	• Sampling and logging temperature values every 1 seconds • Low pass filter or average filter for filtering out noise from sensor data to improve accuracy
S: Supportability	Windows 10 IOT
Design limitation	-

3.1.2 Use cases

The use case defines what the system shall do and treat the system as a black box [22]. It defines the functions of the system and each use case represents a function. It also describes how the system is going to work. It shows the interaction between the actors and the system in order to achieve the objective or goal. The components that are external to the system but have some roles that relates to the system are known as the actors.

The use case diagram for the Home Automation system is shown in Figure 3-2 and is explained as follows.

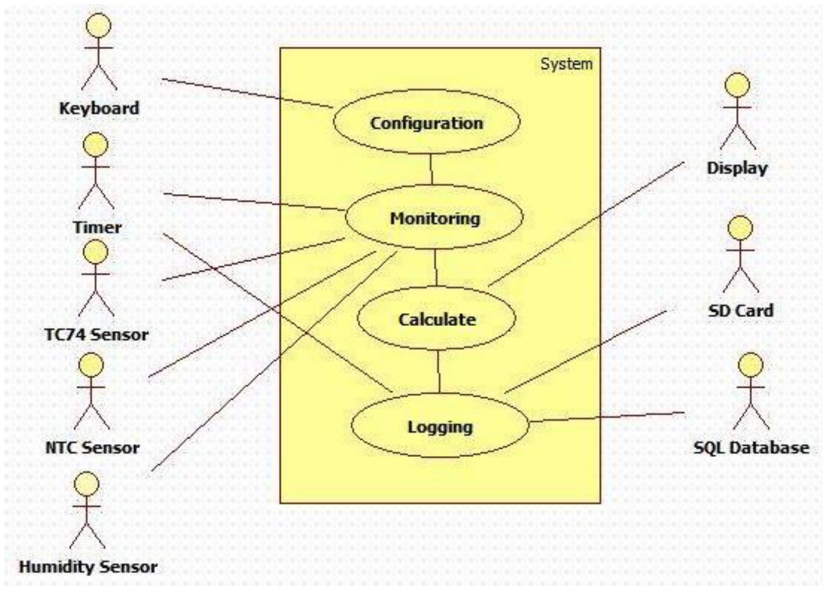


Figure 3-2: Use case diagram for Home Automation System

- *Configuration*: It is responsible for obtaining the number of sensors, sensor Id, and setting up pins as input and output pins.
- *Monitoring:* It is responsible for reading sensor values based on sampling time, implementing low pass filter to get rides of unwanted signal(s) from the sensor value.
- *Calculate Value:* This perform signal conditioning and display the temperature value on screen.
- *Logging:* Responsible for getting temperature values from 'CalculateValue' and log the data to SQL database and SD Card.

3.1.3 Domain Model

The domain model shows the conceptual classes that is based on static information and gives a good view of the inheritances between classes. It is a conceptual model of the domain that incorporates both behavior and data[23].

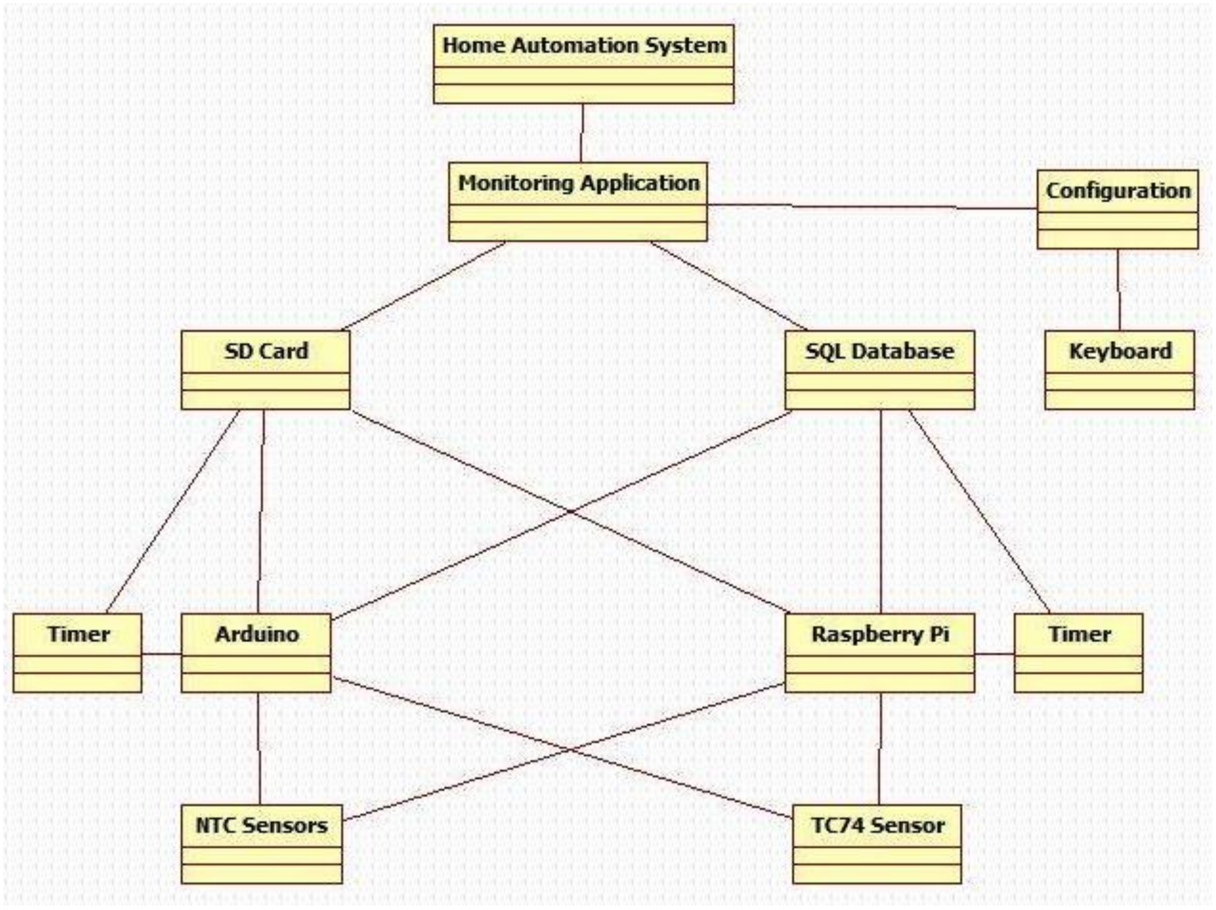


Figure 3-3: Domain Model for Home Automation Systems

It gives an overview of the static information. The domain model for the Home Automation system is shown in Figure 3-3

3.1.4 Fully Dressed Use Case Document (FDUCD)

The fully dressed use case document is a text that describes a more detailed structure for a use case [24]. It shows how software shall accomplish the specific functions. The FDUCD for the use cases are presented in Appendix 3.

3.1.5 System Sequence Diagram (SSD)

The system sequences deals with modeling the behavior of the use case. It focuses on what the system is doing and not how is doing it. Events between the actors and the system are one of the main functions of the SSD [22].

The system sequence diagram for the use cases are drawn with reference to the information documented in the main success scenario of the FDUCD. The SSDs for the system are shown in Appendix 4.

3.1 Design

Design is a process of preparing solutions to the problems and requirements, and make model of the design using UML diagrams. It defines the software objects and how these objects are collaborating to satisfy the requirements. It involves making a conceptual solution by defining the objects and assign responsibilities to these objects [25]. The design phase consists of interaction diagrams, class diagrams and object diagrams.

3.1.1 Interaction Diagram

The interaction diagram consists of the sequence diagrams and the collaboration diagrams. The sequence diagram is dynamic system which is time dependent. It shows the sequence of messages in time. The collaboration diagram shows collaboration with focus on data. The system under design is time dependent hence; our focus will be on sequence diagrams (SD).

3.1.2 Sequence Diagram

The sequence diagram for the system under design is shown in Appendix 5.

3.1.3 Class Diagram

A class diagram is a static model containing class name, attributes and messages for a class and consists of one or several classes [25]. The class diagram for the system is provided in Appendix 6.

3.1.4 Object Diagram

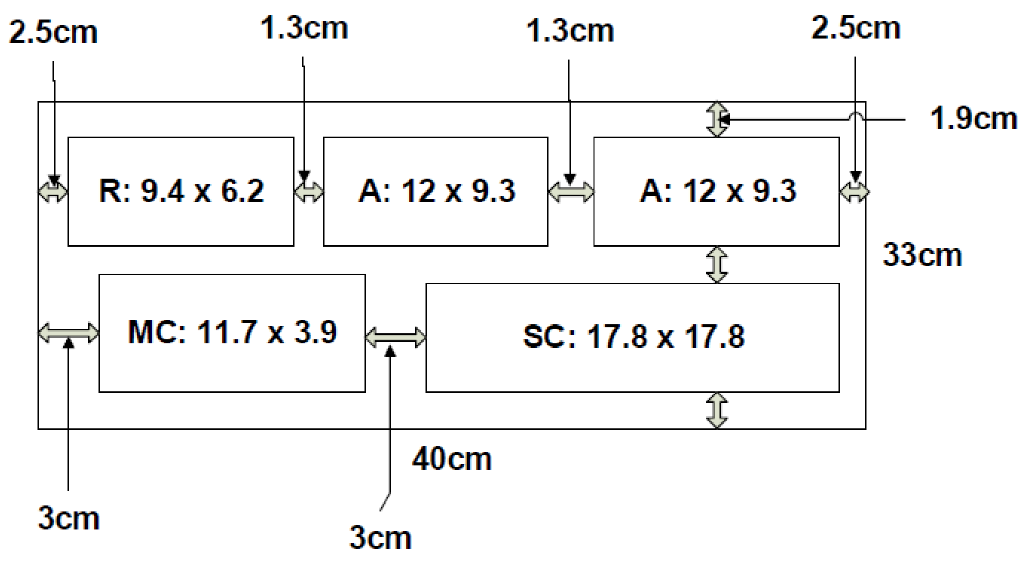
An object diagram represents an instance of a class which is derived from class diagrams. It is dependent on the class diagrams. The object diagram for this system is shown in Appendix 7.

3.2 Suitcase Design

The project is planned to be built into a suitcase. The design and the selection of the suitcase is based on the equipment that are be built into it and these include

- One Raspberry Pi with TC74 temperature sensor and MCP3002 analog to digital converter (Raspberry Pi size: 9.4 x 6.2 cm)
- Two Arduinos with TMP36/NTC temperature sensors (Arduino size: 12 x 9.3 cm)
- Arduino Bluetooth shields for wireless communication between devices
- Two Xbee devices for wireless communication between devices
- Mini Computer of size 11.7 x 3.9 cm
- Monitor (Screen) of size: 17.8 x 17.8

Based on the selected equipment and their corresponding sizes the suitcase was designed and purchased. Figure 3-4 shows the suitcase design for the project.



R – Raspberry Pi
A – Arduino
MC – Mini Computer
SC – Monitor (Screen)

SUITCASE SIZE:
L = 40cm
B = 33cm
All units are in cm

RASPBERRY PI:
(Sensor Node)
Breadboard is place directly on it.

Arduino:
(Sensor Node)
It comes with the Breadboard attached with it, the space above is included with breadboard

Figure 3-4: Suitcase Design for Building Home Automation

Based on this design, the suitcase in Figure 3-5 was purchased. The inside view of the suitcase is shown in Figure 3-6.

Figure 3-5: Purchased Suitcase for building Home Automation System

Figure 3-6: The Inside View of the Suitcase

3.2.1 Advantages of using suitcase

The main advantages of building Home Automation system into a suitcase are

- Ease of presentation
- Portability.
- It makes testing and maintenance of the whole system easy.
- Protection for the various components and devices.

3.3 Equipment List

The list of equipment and budget recommended for purchase for the system under design is attached with this report. This list includes different sensors, Arduino and Raspberry Pi, etc. with their prices and where to purchase them.

4 Implementation and Results

The coding based on software analysis and design is shown in Appendix 9 as master and slave codes respectively.

The circuits for measuring temperature using the different sensors are implemented in this chapter. The implementation starts with proper wiring of the sensors on the breadboard after looking up the datasheet to identify the terminals. This is followed by connecting the right signal to the Arduino board. The output of the measurement can then be read using the serial monitor in the Arduino IDE.

4.1 Connection of Temperature Sensors

The temperature sensors are hooked to either Arduino board or a Raspberry Pi 2 via the project board. The Arduino and Raspberry Pi 2 board provide the necessary signal for efficient operation of the respective sensors.

4.1.1 Connection of TMP36 Temperature Sensor with Arduino Board

For accurate and efficient temperature measurement, it is necessary that the sensor is wired correctly and supplied with the proper signal. Figure 4-1 shows the connection of TMP36 to Arduino board drawn in *fritzing*. The collector terminal is connected to 5V power supply, the emitter terminal is connected to ground and the base terminal is connected to analog A0 pin on the Arduino board.

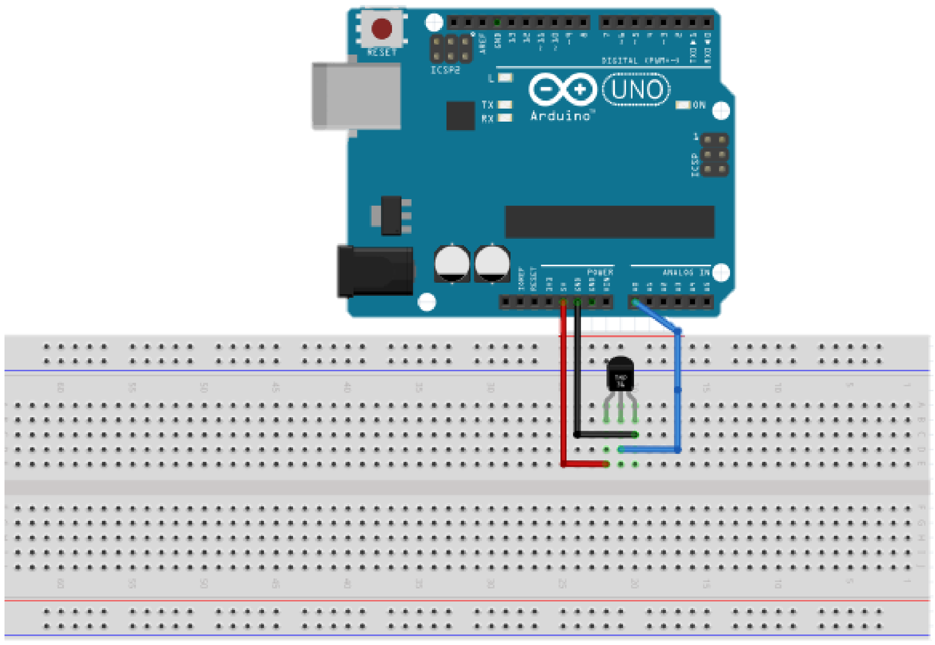


Figure 4-1: TMP36 Connection to Arduino Board

4.1.1.1 Sketch for TMP36 Temperature Reading

The sketch used in measuring the temperature is written in the editor of the Arduino IDE, verified by clicking the verify icon, ✓ and uploaded to the microcontroller by clicking the upload button, ➜ using Arduino bootloader. Proper COM port should be selected using the tool menu dropdown and select the appropriate port. The code is provided below:

```
/* This code is for TMP 36 temperature sensor to read temperature
 *  both in degreeC
 */

const int temperaturePin = A0;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
}

void loop() {
  // Declaration of variables voltage, degreeC
  float voltage, degreesC;
  float getVoltage(int pin);
  voltage = getVoltage(temperaturePin);

  degreesC = (voltage - 0.5) * 100.0;   //converting from 10 mv per degree with 500 mV offset
  //to degrees ((voltage - 500mV) times 100)
```

4.1.1.2 Conversion of Voltage to Temperature

TMP36 does not measure temperature directly; hence the voltage value it measures is converted to temperature in ℃ by the following lines in the sketch:

```
 degreesC = (voltage - 0.5) * 100.0;   //converting from 10 mv per degree with 500 mV offset
                                       //to degrees ((voltage - 500mV) times 100)
```

Offset of 500mV is used in order to create allowance for measuring negative temperature.

Before the conversion, the voltage value from the *pin* is scaled by finding the equivalent to the bit. The bit ranges from 0 – 1023 and the voltage range is 0 – 5V.

This was implemented in the sketch in the *getVoltage() class* that was implemented in a new tab.

The sketch used is provided below:

```
float getVoltage(int pin)
{
  return (analogRead(pin) * 0.004882814);

  // This equation converts the 0 to 1023 value that analogRead()
  // returns, into a 0.0 to 5.0 value that is the true voltage
  // being read at that pin.
}
```

4.1.1.3 Connection of NTC Temperature Sensor with Arduino Board

The connection of the NTC temperature sensor to the Arduino board is easy and straight forward.

Start by connecting one end of the 10K resistor to 5V, connect the other end of the 10K 1% resistor to one pin of the thermistor and the other pin of the thermistor to ground. Then connect Analog A0 pin to the junction of the resistor and NTC sensor of the two. This form a voltage divider. The connection is shown in Figure 4-2 drawn using *fritzing*.

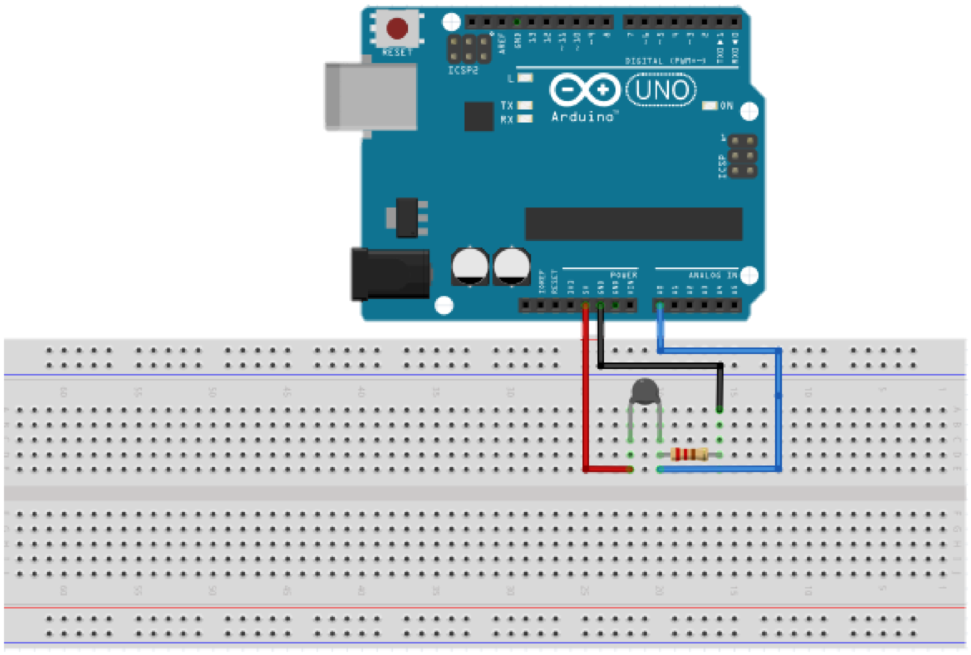


Figure 4-2: NTC Temperature Sensor Connection to Arduino Board

4.1.1.4 Determination of Temperature by NTC Temperature Sensors

The resistance value of NTC temperature sensor will need to be converted [26] to temperature. There are basically two ways of temperature determination when using NTC temperature sensor:

i. Steinhart-Hart Equation method.
ii. Beta Factor equation method.

Steinhart-Hart equation method is used in determining the temperature values in this project. The Steinhart-Hart equation is provided in equation (1).

$$\frac{1}{T} = A + B\ln(R) + C(\ln(R))^3 \qquad (1)$$

Where: $A = 0.001129148, B = 0.000234125 \; and \; C = 8.76741E - 08$

This is implemented in the sketch by the *getTemp() class*. The sketch is provided as:

```
double getTemp()
{
  // Inputs ADC Value from Thermistor and outputs Temperature in Celsius
  int RawADC = analogRead(temperaturePin);
  long Resistance;
```

```
 double Temp;
 // Assuming a 10k Thermistor. Calculation is actually: Resistance = (1024/ADC)
 Resistance = ((10240000 / RawADC) - 10000);
 // Utilizes the Steinhart-Hart Thermistor Equation:
 // Temperature in Kelvin = 1 / {A + B[ln(R)] + C[ln(R)]^3}
 // where A = 0.001129148, B = 0.000234125 and C = 8.76741E-08
 Temp = log(Resistance);
 Temp = 1 / (0.001129148 + (0.000234125 * Temp) + (0.0000000876741 * Temp * Temp * Temp));
 Temp = Temp - 273.15; // Convert Kelvin to Celsius
 return Temp; // Return the Temperature
}
```

4.1.1.5 Sketch for Reading Temperature by NTC Sensor

The sketch used in measuring the temperature is written in the editor of the Arduino IDE, verified by clicking the verify icon, ✓ and uploaded to the microcontroller by clicking the upload button, → using Arduino bootloader. Proper COM port should be selected using the tool menu dropdown and select the appropriate port. The code is provided below:

```
// Read Temperature Values from NTC Temperature Sensor
const int temperaturePin = 1;
void setup()
{
 Serial.begin(9600);
}
void loop()
{
 float temperature;
 temperature = getTemp();
 Serial.print("Temperature Value: ");
 Serial.print(temperature);
 Serial.println("*C");
 delay(1000);
}
```

4.1.2 Connection of Temperature Sensor Using Raspberry Pi 2

In order to use the Raspberry Pi 2 development board effectively and efficiently, it is very necessary that the various steps in setting it up are strictly adhered to.

4.1.2.1 Setting up Raspberry Pi 2

The following steps [10] should be followed in setting up the PC and Raspberry Pi 2:

4.1.2.1.1 Setting up the PC

In setting up the PC, the following steps are observed:

Step 1: Log on to https://dev.windows.com/en-us/iot and click get started.

Step 2: Select your device. In this case Raspberry Pi 2

Step 3: Setting Up PC: Download and install Windows 10 setup or upgrade the PC to Windows 10 then install Visual studio 2015 if you don't have them on your system.

To setup your Windows 10 IoT Core development PC, you first need to install the following:

i. **Make sure you are running the public release of Windows 10 (version 10.0.10240) or better**: If you are already running Windows 10, you can find your current build number by clicking the start button, typing "winver", and hitting enter.

ii. **Install Visual Studio 2015**
   a. Visual Studio Community Edition is recommended but Visual Studio Professional 2015 and Visual Studio Enterprise 2015 will work as well.
   b. If you have to install Visual Studio, make sure to do a Custom install and select the checkbox **Universal Windows App Development Tools** -> **Tools and Windows SDK**.
   c. If you already have Visual Studio, you will be prompted to download the needed tools when attempting to run our solution in the next part of the tutorial.

iii. **Install Windows IoT Core Project Templates.** Alternatively, the templates can be found by searching for *Windows IoT Core Project Templates* in the Visual Studio Gallery or directly from Visual Studio in the Extension and Updates dialog (Tools > Extensions and Updates > Online).

iv. **Enable developer mode** on your Windows 10 device. The relevant portion of the linked instructions is the "Windows 10 Desktops/tablets" section, as you should be attempting setup with one of these devices.

v. Open Visual Studio 2015 and **create a Universal Windows Platform (UWP) App** by selecting 'File > New > Project > Visual C# > Windows > Universal > Blank App (Universal Windows)'. This is required to ensure that the framework dependencies are available for our samples to build.

4.1.2.1.2 Setting up the Device

Before proceeding with the device setup, the following items are needed to be in place:

1. **A PC running Windows 10** - as prepared in the previous step.
2. **Raspberry Pi 2**
3. **5V Micro USB power supply** - with at least 1.0A current. If you plan on using several power-hungry USB peripherals, use a higher current power supply instead (>2.0A).
4. **8GB micro SD card** - class 10 or better.
5. **HDMI cable and monitor**

6. **Ethernet cable**
7. **Micro SD card reader** that is compatible with Raspberry Pi 2

Step 1: Install the Windows 10 IoT Core Tool

i. **Download a Windows 10 IoT Core image**. Save the ISO to a local folder.
ii. Double click on the ISO (IoT Core RPi.iso). It will automatically mount itself as a virtual drive so you can access the contents.
iii. Install **Windows_10_IoT_Core_RPi2.msi**. When installation is complete, flash.ffu will be located at **C:\Program Files (x86)\Microsoft IoT\FFU\RaspberryPi2**.
iv. Eject the Virtual CD when installation is complete - this can be done by navigating to the top folder of File Explorer, right clicking on the virtual drive, and selecting “Eject”.

Step 2: Put the Windows 10 IoT Core image on your SD card

i. **Insert a micro SD card** into your SD card reader.
ii. **Use IoTCoreImageHelper.exe** to flash the SD card. Search for “WindowsIoT” from start menu and select the shortcut “WindowsIoTImageHelper”. Figure 4-3 shows the screen shot of “WindowsIoTImageHelper”.

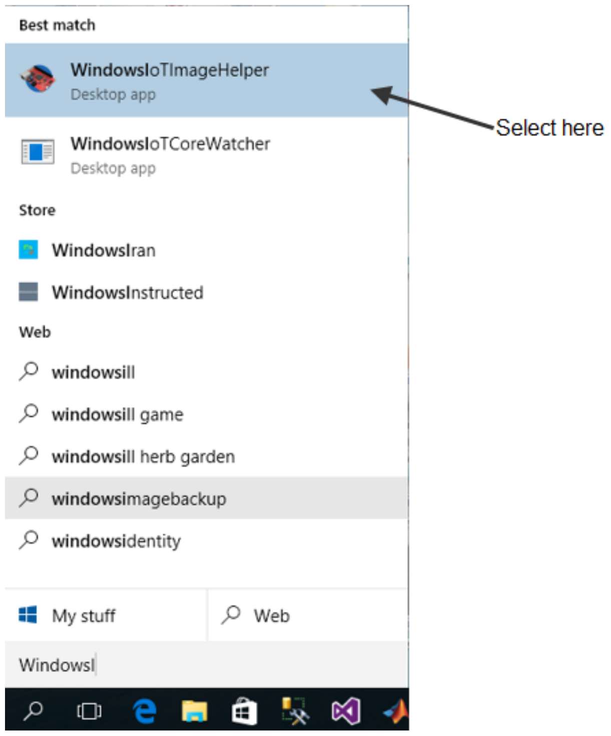
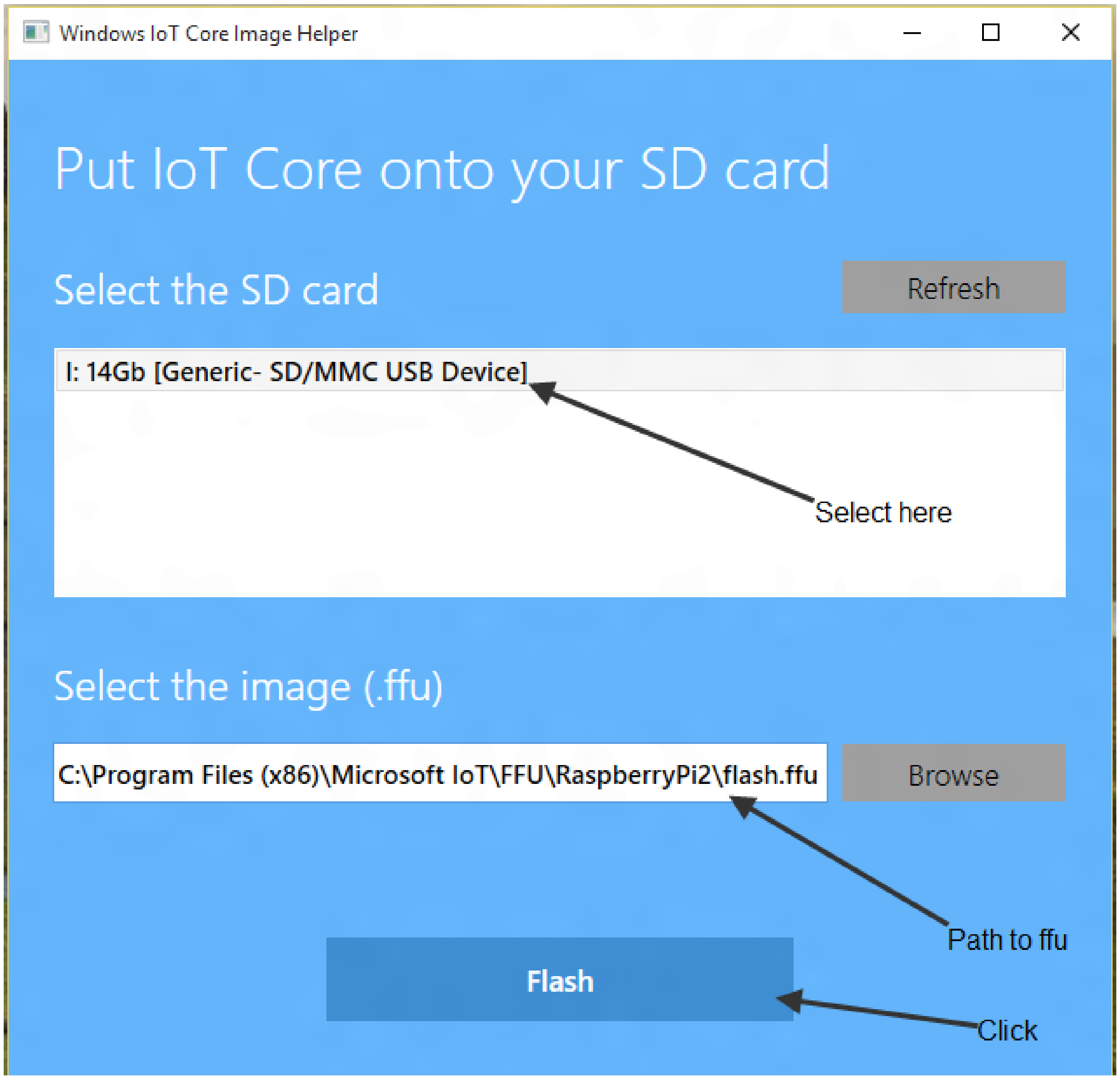


Figure 4-3: Screen shot of “WindowsIoTImageHelper”

iii. The tool will enumerate devices as shown. Select the SD card you want to flash, and then provide the path to the ffu to flash the image. Figure 4-4 shows the screen shot of the interface for Windows IoT Image Helper.


Windows IoT Core Image Helper

Put IoT Core onto your SD card

Select the SD card

I: 14Gb [Generic- SD/MMC USB Device]

Select the image (.ffu)

C:\Program Files (x86)\Microsoft IoT\FFU\RaspberryPi2\flash.ffu

Figure 4-4: Screen shot of Windows IoT Image Helper

iv. **Safely remove your USB SD card reader** by clicking on “Safely Remove Hardware” in your task tray, or by finding the USB device in File Explorer, right clicking, and choosing “Eject”. Failing to do this can cause corruption of the image.

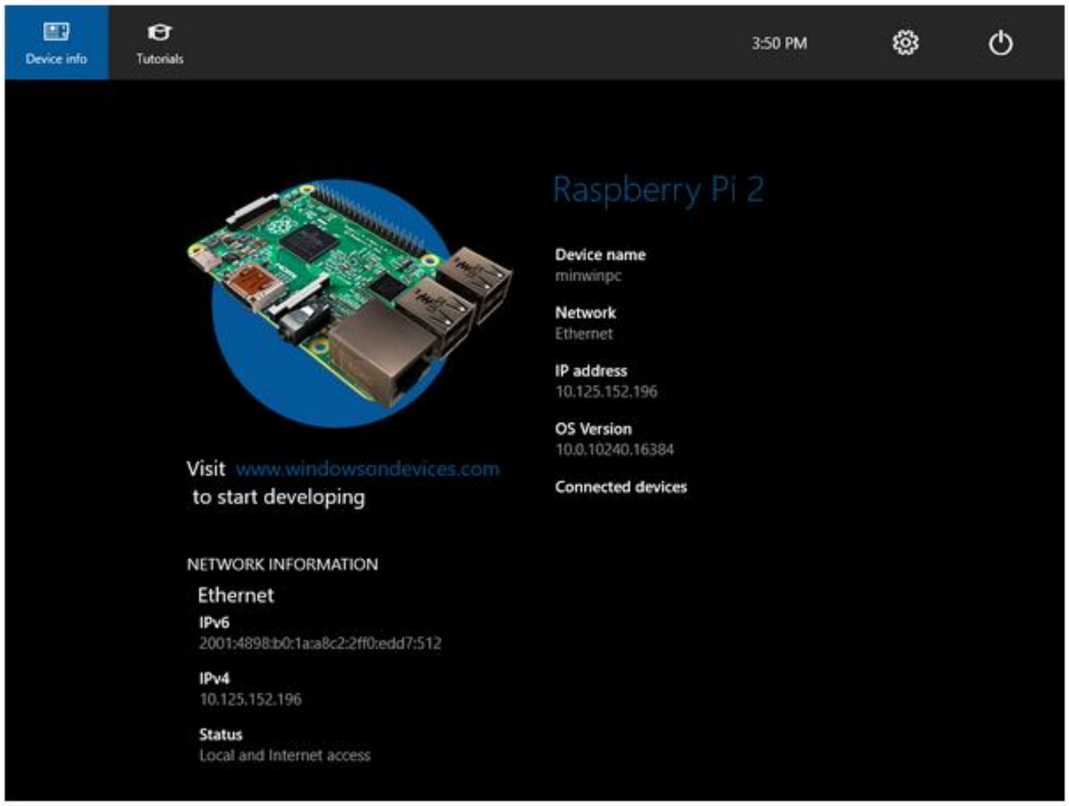
Step 3: Hook up the Raspberry Pi 2 Development Board

All the various connection points are shown in Figure 1-6

i. **Insert the micro SD card** you prepared into your Raspberry Pi 2.
ii. **Connect a network cable** from your local network to the Ethernet port on the board. Make sure your development PC is on the same network.
iii. **Connect an HDMI monitor** to the HDMI port on the board.
iv. **Connect the power supply** to the micro USB port on the board.

Step 4: Boot Windows 10 IoT Core

i. Windows 10 IoT Core will boot automatically after connecting the power supply. This process will take a few minutes. After seeing the Windows logo, your screen may go black for about a minute - don't worry, this is normal for boot up. You may also see a screen prompting you to choose a language for your Windows 10 IoT Core device - either connect a mouse and choose your option, or wait about a 1-2 minutes for the screen to disappear.
ii. Once the device has booted, the DefaultApp will launch and display the IP address of your RPi2.



Raspberry Pi 2

Device name
minwinpc

Network
Ethernet

IP address
10.125.152.196

OS Version
10.0.10240.16384

Connected devices

Visit www.windowsondevices.com to start developing

NETWORK INFORMATION

Ethernet

IPv6
2001:4898:b0:1a:a8c2:2ff0:edd7:512

IPv4
10.125.152.196

Status
Local and Internet access

Figure 4-5: Screen Shot of Device information on Screen display

In order to connect a running device, we use PowerShell. The steps in configuring PowerShell are enumerated in the preceding section.

4.1.2.1.3 Using PowerShell to connect and configure a device running Windows 10 IoT Core

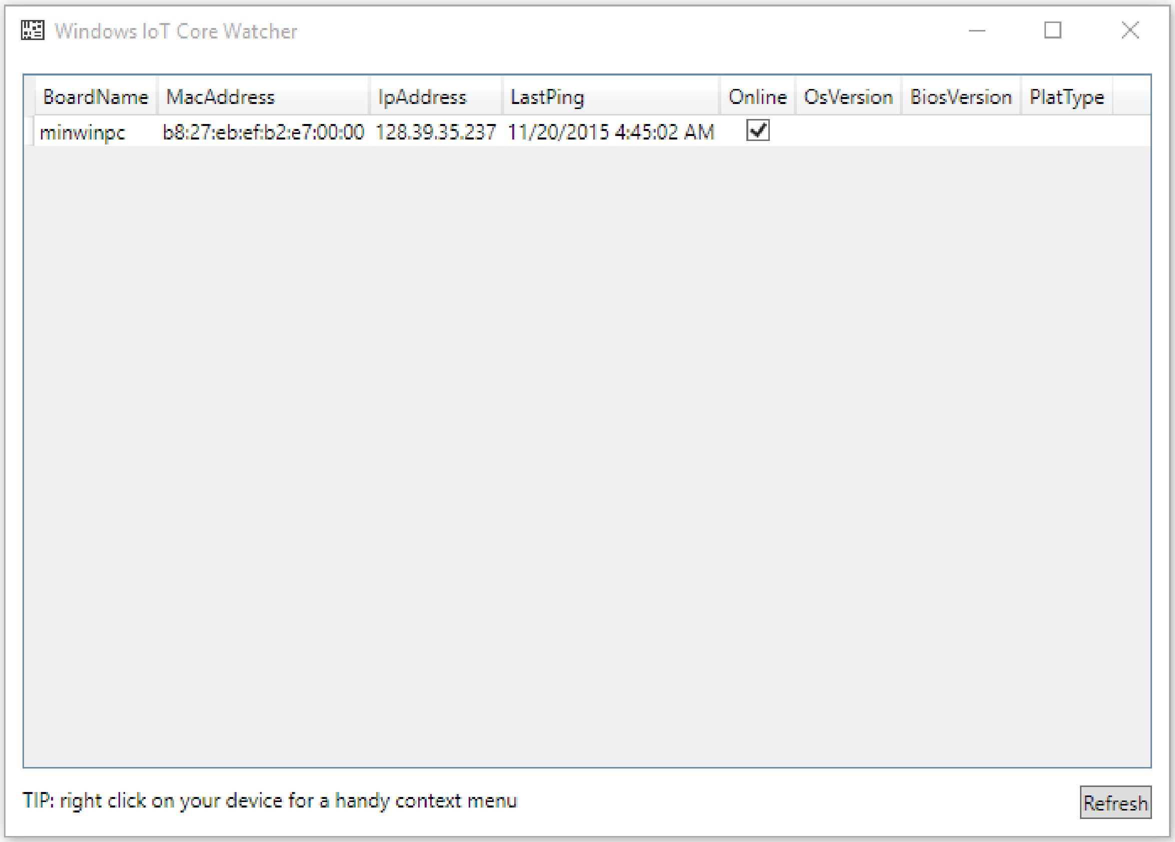
In order to run the device on Windows 10 IoT core, it will be first configured as a remote Administration.

4.1.2.1.3.1 Remote Administration and Configuration

PowerShell is a task-based command-line shell and scripting language, designed especially for system administration. It can be remotely configured and manage by any Windows 10 IoT Core device using Windows PowerShell.

4.1.2.1.3.2 Initiating a PowerShell (PS) Session

To start a PS session with your Windows 10 IoT Core device, you'll first need to create a trust relationship between your host PC and your device. After booting your Windows IoT Core device, an IP address will be shown on the screen attached to the device as in Figure 4-5 or alternatively the same information can be obtained from Windows 10 IoT Core Watcher utility that pops up on booting the device as shown in Figure 4-6.



BoardName	MacAddress	IpAddress	LastPing	Online	OsVersion	BiosVersion	PlatType
minwinpc	b8:27:eb:ef:b2:e7:00:00	128.39.35.237	11/20/2015 4:45:02 AM	✓			

TIP: right click on your device for a handy context menu

Figure 4-6: Screen shot of Windows IoT Core Watcher

i. Launch an administrator PS console on your local PC by searching for it at search the web and windows.

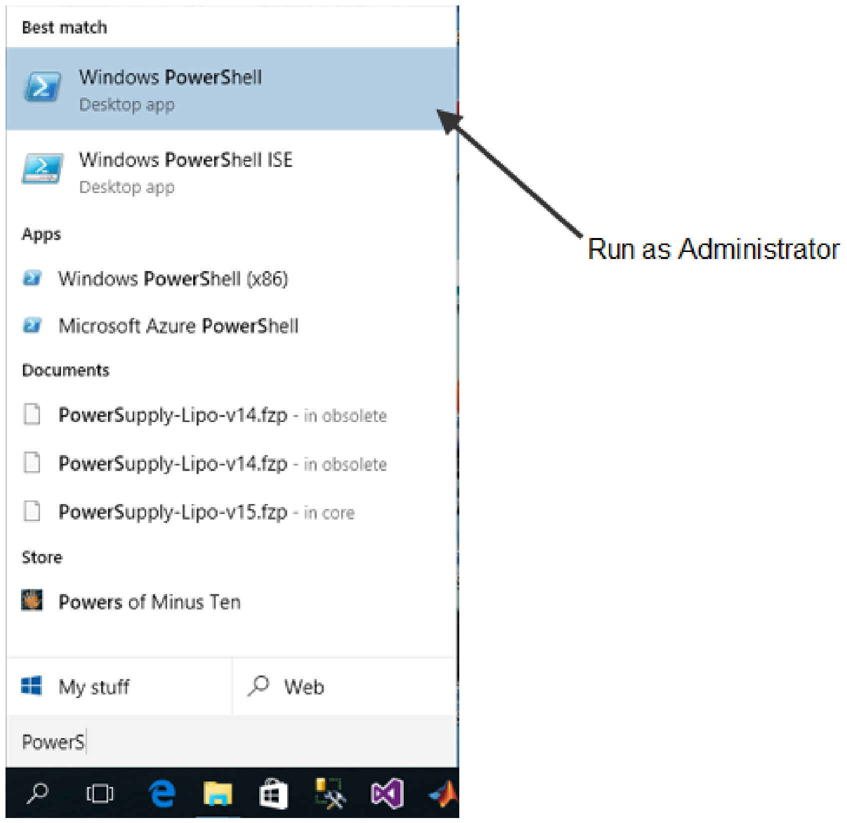


Figure 4-7: Screen shot of Windows PowerShell

4.2 Logging

Logging of temperature and humidity data obtained from the sensors through Arduino device and Raspberry Pi 2 is done in two locations. Locally on a NUC computer and externally on a remote server database through internet. The data can be accessed by an authorized person from any location for control purpose. Three communication means; Bluetooth, Wi-Fi and Xbee communications are used in transmitting the temperature data to the different storage location.

4.2.1 Bluetooth Communication

Bluetooth communication is a wireless means of communication that can be employed in transmitting voice and data at high speeds using radio wave. It operates at a frequency of 2.45 GHz frequency band and a typical data transfer rate of about 2 megabits per second (Mbps)[27]. It employs a master-slave protocol, with one master and one or more devices as slaves. The master device uses link manager software to identify other Bluetooth devices to create links with them to be able to send and receive data [27]. It has a range of approximately 10 meters perimeter. Bluetooth uses a spread-spectrum frequency hopping technology which makes it possible for multiple devices to connect at different frequencies without interference.

4.2.1.1 Why Bluetooth Communication

The choice of using Bluetooth communication was necessitated by the enormous advantages it has over other means of wireless communication in short distance transmission.

i. It uses less power for transmission
ii. Its transmission speed is reasonably high
iii. It can connect up to 7 devices
iv. It does not require a direct line of sight for communication
v. It does not require synchronization

4.2.1.2 HC-05 Bluetooth Module

HC-05 Bluetooth module is an easy to use and cheap device. It can be used to setup a wireless serial Bluetooth connection between a host device and a slave device or vice versa; as a master it can initiate a connection and as a slave, it can only accept connections. It is a serial port Bluetooth module that is fully qualified Bluetooth V2.0+EDR (Enhanced Data Rate) 3 Mbps Modulation with complete 2.4GHz radio transceiver and baseband[28]. Its connection with Arduino UNO device drawn in *fritzing* is shown in Figure 4-8.

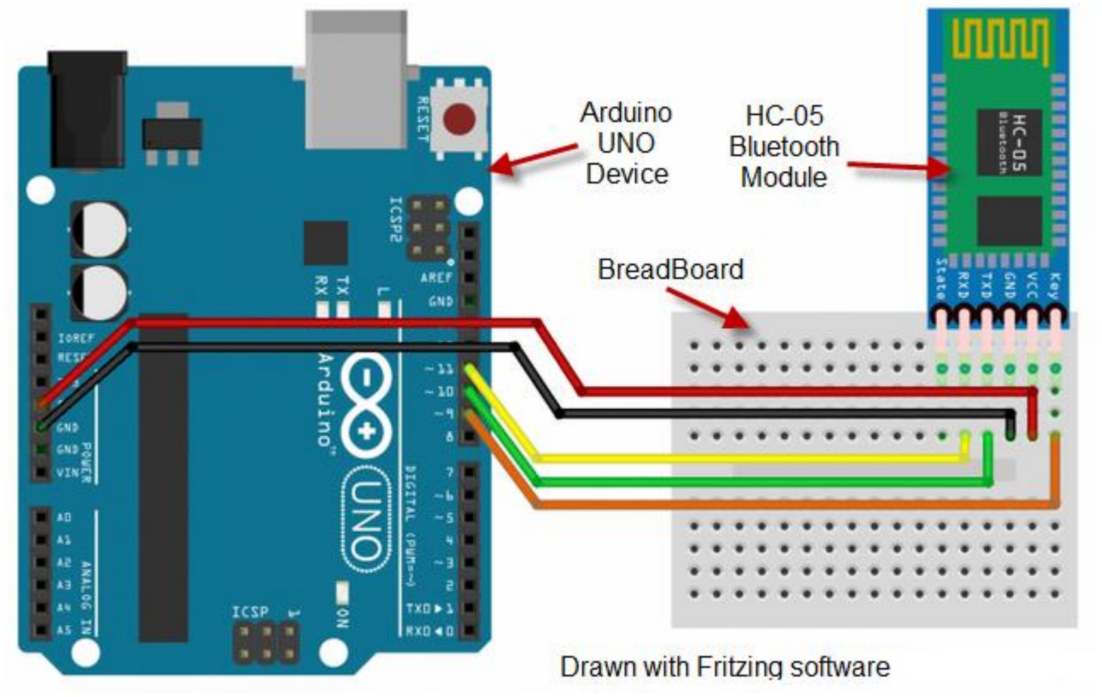


Figure 4-8: Connection of HC-05 with Arduino UNO device

It is observed that the TX and RX pins of HC-05 Bluetooth module are connected to digital pin 10 and 11 on the Arduino UNO device respectively. HC-05 Bluetooth module has several software and hardware specifications[28] that made it very useful for this project.

4.2.1.3 Hardware Specifications of HC-05 Bluetooth Module

The Hardware specifications obtained from the datasheet are:

i. Low power from 1.8V to 3.6V, 1.8V Operation
ii. Parallel Input Output (PIO) control
iii. UART interface with programmable baud rate
iv. Integrated antenna
v. Edge connector
vi. +4dBm RF transmit power
vii. -80dBm sensitivity

4.2.1.4 Software Specifications of HC-05 Bluetooth Module

The software specifications obtained from the datasheet are:

i. Default baud rate: 38400, Data bits: 8, Stop bit: 1, Parity: No parity, Data control: has.
ii. Supported baud rate: 9600, 19200, 38400, 57600, 115200, 230400, and 460800.
iii. Given a rising pulse in PIO0, device will be disconnected.
iv. Auto-pairing PINCODE: “0000” as default.

v. Permit pairing device to connect as default.
vi. Auto-reconnect in 30 min when disconnected as a result of beyond the range of connection.

4.2.1.5 Setting Up HC-05 Bluetooth Module

HC-05 Bluetooth module can be setup either as host (master) or slave device. The AT default commands used in setting HC-05 mode as a server (master) are outlined below:

1. Connect PIO11 to high level
2. Power on, module in to command state.
3. Using baud rate 38400, sent the "AT+ROLE = 1\r\n" to module, with "OK\r\n" means setting successes.
4. Connect PIO11 to low level, repower the module, the module work as server (master).

4.2.1.6 Pairing HC-05 with PC

Before pairing the HC-05 with the PC, we need to first establish a link between the Arduino and the Bluetooth module. This can be achieved by the following steps:

i. Search for Bluetooth icon on the computer, right click and select add a device.
ii. Search for new device and add HC-05.
iii. Entering the default pairing code of "1234". If it was not changed during the Bluetooth module setup.
iv. Sketches can now be uploaded to send and receive data from the computer.

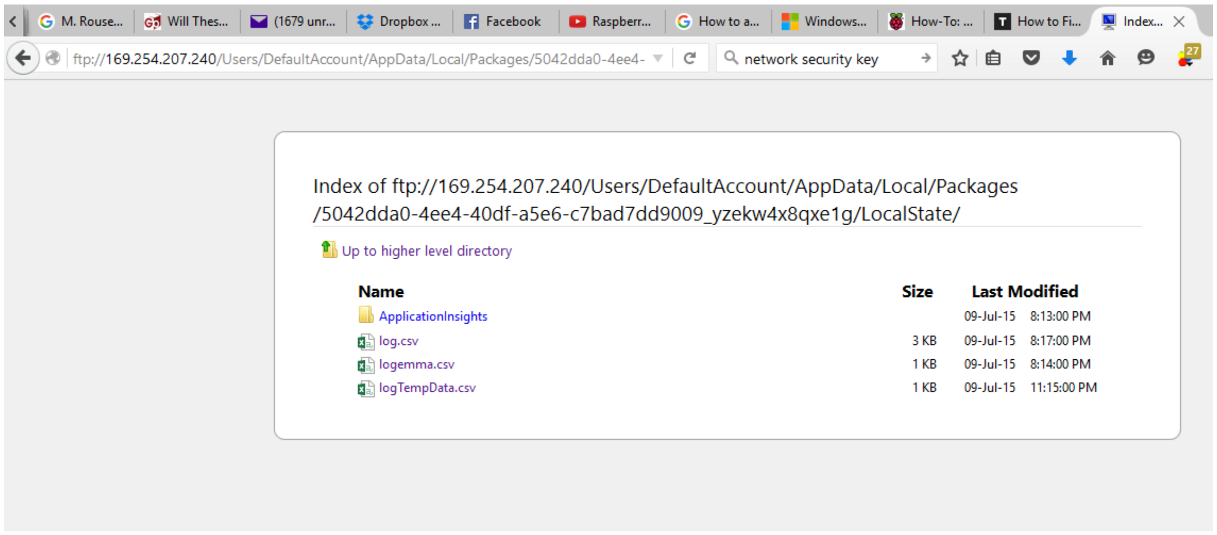
For the HC-05 to communicate with the PC, a terminal emulator software such as tera term, Realterm etc. needs to be installed on the PC. In this project, tera term terminal emulator software is used. Search for the Bluetooth signal on the PC and pair with it by entering the same passcode and connecting with the right port. Tera term is opened on the PC and the passcode, baud rate, and the port number is selected. The PC is now paired with the Arduino board, hence the temperature data measured can now be received on the PC.

4.2.2 Raspberry Pi Wi-Fi Adapter

The temperature data measured by the Raspberry Pi 2 is logged to the SD Card and transmit through Wi-Fi to the remote server. The code used in achieving was done in C# and it is located in Appendix 8 in the Appendix section.

The data logged can be accessed through the internet using the FTP[4] protocol and adding the IP address assigned to the Raspberry device.

(ftp://169.254.207.240/Users/DefaultAccount/AppData/Local/Packages/5042dda0-4ee4-40df-a5e6-c7bad7dd9009_yzekw4x8qxe1g/LocalState/). Figure 4-9 shows the csv file of temperature data logged obtained from the browser.



Index of ftp://169.254.207.240/Users/DefaultAccount/AppData/Local/Packages
/5042dda0-4ee4-40df-a5e6-c7bad7dd9009_yzekw4x8qxe1g/LocalState/

Up to higher level directory

Name	Size	Last Modified	
ApplicationInsights		09-Jul-15	8:13:00 PM
log.csv	3 KB	09-Jul-15	8:17:00 PM
logemma.csv	1 KB	09-Jul-15	8:14:00 PM
logTempData.csv	1 KB	09-Jul-15	11:15:00 PM

Figure 4-9: Screen shot Temperature Data Logged

[4] FTP: is a standard network protocol used to transfer computer files from one host to another host over a TCP-based network, such as the Internet

5 Database

The implementation of Home Automation involves the read of different sensor measurement (monitoring) and the write to number of actuators for control purpose. These various sensor measurements needs to be structured and organized in such a way that access to the data, management of information and update of measurement can easily be carried out, hence, the need for a database.

A database is an organized structured collection of data or information for easy management, accessibility and update.

In this project, the database is designed using SQL server and it is structured in to different tables which are 2-dimensional with rows and columns that are inter related. The column has different attributes and the row has specific information. The logging of data in to the database in this project is carried out at 2 different stages:

1. Local on a NUC computer
2. Server based database.

Each of the devices (Arduino and Raspberry Pi) used for monitoring the sensors are connected to a NUC computer where the data are logged locally. The information are then transmitted to the server based Database through XBee communication from the Arduino device and Raspberry USB through the Wi-Fi dongle communication which help in transmitting data from the sensor to the remote server in the control center. Figure 5-1 shows the overview of the database system with the various interconnection of logging process in to the local database and the server based database.

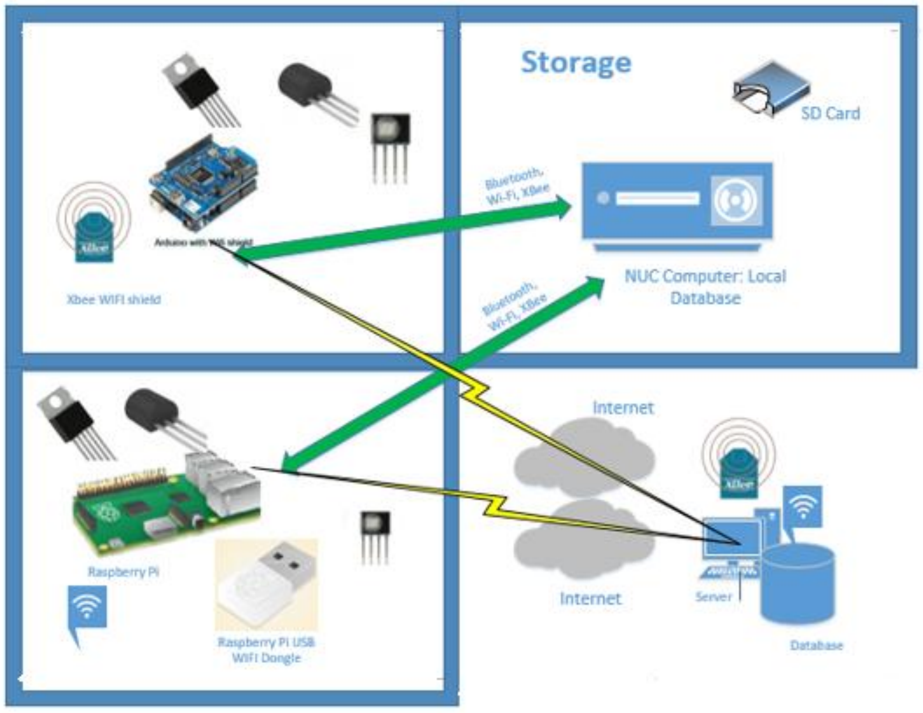


Figure 5-1: Over view of the Database Storage System

5.1 Database Model

The model of the database was achieved using CA Erwin Data Modeler (ERwin). Erwin is a software tool used basically for data modeling. Erwin's data modeling engine is based upon IDEF1X[5]. The choice of Erwin was necessitated by the fact that the Community Edition that is used in this project is free and support up to 25 tables which is within the requirement of the project. However, it can be increased to accommodate more than 25 tables on further expansion of the project. This will requires a different version of Erwin like standard Edition, Navigator Edition or Workgroup Edition which will attract some license fee. Figure 5-2[29] shows the model of the database using the Erwin Community Edition modeler.

[5] IDEF1X: Integration DEFinition for Information Modeling. It is a data modeling language for development of semantic data models.

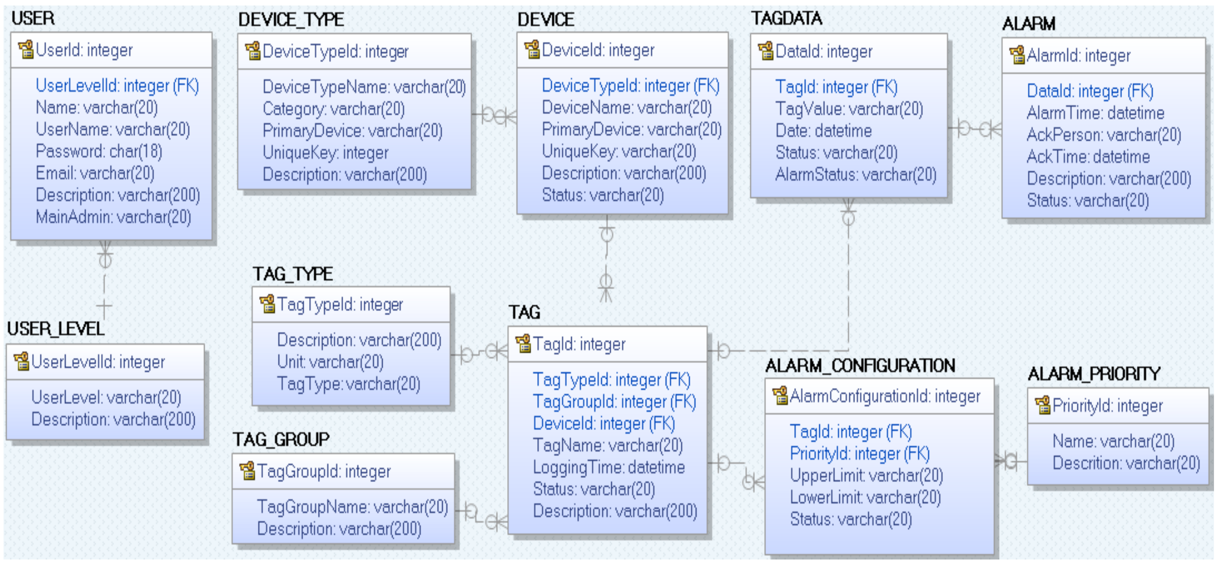


Figure 5-2: Model of Database in ERwin

It is observed from Figure 5-2 that the database has five main tables; USER, DEVICE, ALARM, TAG and ALARM_CONFIGURATION.

- USER: It has UserId as its Primary key and the data type is integer. It has seven attributes out of which UserLevelId is a Foreign key.
- DEVICE: It has DeviceId as its Primary key and the data type is integer. It has six attributes out of which DeviceTypeId is a Foreign key.
- TAG: It has TagId as its Primary key and the data type is integer. It has seven attributes out of which TagTypeId, TagGroupId and DeviceId are a Foreign keys.
- ALARM_CONFIGURATION: It has AlarmConfigurationId as its Primary key and the data type is integer. It has five attributes out of which TagId, PriorityId are Foreign key.
- ALARM: It has AlarmId as its Primary key and the data type is integer. It has seven attributes out of which DataId is a Foreign key.

5.2 SQL

The original database (DMM) was designed using in MySQL by the previous student (Kishan Prajapati) but it has been so far converted to SQL[6]. This was obtained using the functionality of the Erwin modeler to generate the code which was executed in the SQL IDE to create the

[6] SQL: Structured Query Language

different tables in the database. The code used in generating the database is provided in Appendix 8.

6 Discussion and Suggestion

Discussion of success story, challenges and suggestion to further work as observed during the implementation of each task are categorized below:

6.1 Monitoring System

In the project only temperature and humidity were measured, in subsequent task more quantities like wind speed, Atmospheric pressure etc. can be measured. Also, all the sensors used according to the task description were wired, attempt can be made in using wireless sensors in future work.

6.2 Logging System

Temperature and humidity values were measured and logged to locally on SD Card and can be monitored over the internet. Due to time constraint and some challenges experienced logging to NUC could not be realized. This could be implemented in future work. NUC computer will give the best solution for the local data storage because it has a larger storage capacity, many peripherals can be connected to it and input device such as keyboard can be used with it.

6.3 Control System

In this project, the major focus was on data monitoring and logging. Future work can be extended to control.

6.4 Challenges

Several challenges were experienced during the course of the project. They challenges are categorized into different subunits as enumerated:

6.4.1 Loss Human Capacity

Originally, the project work was allocated to 4 students but along the line one student drop for reason best known to him. The task was divided among the three remaining students but 20 days to the hand-in of the project, another student dropped the course leaving behind 2 students. This project report is based on the part undertaken by the two remaining students in the group, time could not allow us to undertake the work left by the departing student. We will suggest that before student are allowed to join another group, there should be an opposite movement of student to the group he/she is leaving in order to compensate for the loss of capacity.

6.4.2 Software Compatibility

Compatibility of the various sensors with Windows 10 IoT were major problems because of Windows 10 being in its development stage most of the sensors components were not compatible with it.

6.4.3 Good Feasibility Study / Limitation on Purchase Channel of Components

The features and specifications of the components need to be checked against the communication protocols before procurement of the devices and sensors purchased. This was due to non-availability of the intended components and sensors in the Norwegian shops that the school has memorandum of understanding with and also due to challenge of compatibility issue with Windows 10 mentioned in 6.4.2.

7 Conclusion

The objective of the project task of Analyze, design and development of a prototype that can be built into a suitcase was not fully implemented as some task that were allocated to the student that left the group could not be undertaken due to the limited time, challenge on using the purchased components and limited information about Windows 10 IoT. However, the Analysis and design were fully implemented and the development of a prototype was 60% completed.

Appendices

The list of appendices are outlined below:

1. Appendix 1: Abstract
2. Appendix 2: Project Task Description
3. Appendix 3: Fully Dressed Use case Document (FDUCD)
4. Appendix 4: Use Case
5. Appendix 5: Sequence Diagram
6. Appendix 6: Domain Diagram
7. Appendix 7: Class Diagram
8. Appendix 8: Codes
9. Appendix 9: Coding for Analysis and Design of the Home Automation System

Appendix 1: Abstract

Telemark University College

Faculty of Technology

M.Sc. Programme

PROJECT REPORT, SCE4006

Students:	**Egbunu Achema (142773), Idachaba Emmanuel (142938)**		
Thesis title:	**Home Automation with Windows 10 IOT**		
Signatures:	. . . . …………………………………..		
Number of pages:	102		
Keywords:	. . . .		
Supervisor:	Hans-Petter Halvorsen	sign.:	. . . .
2nd Supervisor:	Nils-Olav Skeie	sign.:	. . . .
Censor:	<name>	sign.:	. . . .
External partner:	<name>	sign.:	. . . .
Availability:	<Open/Secret>		

Archive approval (supervisor signature)**:** sign.: . . . . **Date :** . . . .

Abstract:

The home automation has becomes so vital due to the comfort and flexibility it gives the user for using home devices. The design and implementation of this project is based on using Arduino and Raspberry Pi for reading temperature values, local and remote data logging, alarm system and the overall system management over internet.

These devices have been distributed in each room interfacing with the sensors and collecting the sensor data. The sensor data collected is wirelessly transferred to data aggregator system (Aggregator: Arduino) using XBee and Bluetooth.

The data aggregator is directly connected to Wi-Fi (Router) where the sensor data is logged to the local database (SD Card) and remote database. The user can access these data from anywhere using smart phones, iPad or computer.

The overall management of the Home Automation system is carried out either remotely or locally using Tablet or Mini PC respectively. For the remote management, the user can monitor the data logged to the data over the internet using Tablet or any other devices.

Hence, this project focus on developing a reliable, effective and robust system using Arduino and Raspberry Pi that will enables the user to monitor temperature data logged to the local database and remote database from anywhere.

Appendix 2: Project task description

Telemark University College

Faculty of Technology

SCE4006 Project

Title: Home Automation Platform using Windows 10 IoT

TUC supervisor: Hans-Petter Halvorsen, Nils-Olav Skeie

External partner: National Instruments

Task description:

Home automation (also known as Smart House, Smart Home, etc.) solutions has greatly increased in popularity over the past several years. Home Automation may include centralized control of lighting, heating, ventilation and air conditioning, appliances, security locks of gates and doors and other systems, to provide improved convenience, comfort, energy efficiency and security.

Keywords: Industrial IT, Raspberry Pi, Arduino, IoT, Windows 10 IoT, Wireless Data

Some of the following topics should be selected and investigated in this project:

- Study of existing Home Automation Systems and explore the Arduino and the Raspberry Pi platforms to see how they can be integrated and used for Home Automation.
- Explore functionality using Raspberry Pi with Windows 10 IoT.
- Logging, Monitoring and Control of typical Data in Homes, e.g., Temperature, etc.
- A Web based logging service similar to e.g., Xively or Temboo should be developed.
- Further development of a platform for monitoring and presentation of data.
- Use of RFID for Access Control
- Camera surveillance with Raspberry Pi
- Explore Wireless Communication, such as Wi-Fi, Bluetooth SMART, XBee, RFID together with Raspberry Pi and Arduino. RFID should be used for Access Control (e.g. open doors). XBee should be used for Wireless Sensor Communication.
- Using Arduino/Raspberry Pi for Monitoring and Data logging using Web Services and "Data Dashboard for LabVIEW" (App for Smartphones and Tablets).
- Development of PID control and other Control strategies like MPC for temperature control in houses.
- Using OPC UA together with Raspberry Pi
- Weather Data Monitoring and Prediction for Temperature Control

- Using Arduino within LabVIEW; LabVIEW LINX
- Explore possibilities for using LabVIEW within Raspberry Pi 2
- Security issues within Home Automation solutions
- Analyze, design and development of a prototype that can be built into a suitcase.

Based on the student's interest, they should select some of the topics above in collaboration with the supervisor for further investigation.

Task background:
Home Automation Systems have been very popular today and many vendors have solutions within this area. This must also be seen in connection with Internet of Things (IoT).

Arduino is a low cost open-source electronics prototyping platform. The Arduino has analog and digital I/O.

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and may be used with a standard keyboard and mouse.

Student category: SCE students

Practical arrangements: None

Signatures:

Supervisor (date and signature):

Students (date and signature):

Appendix 3: Fully Dressed Use Case Documents (FDUCD)

Table 0-1: Fully Dressed Use Case Document of Home Automation System for ’Logging’ use case

Use case name	Logging
Scope	The system under design
Level	User goal
Primary actor	SQL databse, SD Card
Stakeholders	-
Precondition	-
Success garantee	-
Main success scanario	1. Get logging time 2. Get temperature values from ‘Calculate’ 3. Get humidity value from ’Calculate’ 4. Log the data to SD Card 5. Log the data to SQL database 6. Wait for next sampling time 7. GOTO 2.
Extensions	2a. Warning if connection error and re-try 5times and stop 3a. Warning if connection error and re-try 5times and stop
Frequency of occurance	While the system is operational

Miscellaneous	-

Table 0-2: Fully Dressed Use Case Document of Home Automation System for 'Monitoring' use case

Use case name	Monitoring
Scope	The system under design
Level	User goal
Primary actor	Display, NTC Sensor, TMP 36 Sensor, Timer, SQL database, SD Card
Stakeholders	-
Precondition	-
Success garantee	-
Main success scanario	1. Get sensor type 2. Get sampling time 3. Read TC74 sensor value 4. Read NTC sensor value 5. Read Humidity sensor 6. Filter sensors value 7. Wait for next sampling time 8. GOTO 2.
Extensions	3a. Warning if connection error and re-try 5times and stop 4a. Warning if connection error and re-try 5times and stop

Frequency of occurance	While the system is operational
Miscellaneous	-

Table 0-3: Fully Dressed Use Case Document of Home Automation System for 'Calculate' use case

Use case name	Calculate
Scope	The system under design
Level	User goal
Primary actor	Display
Stakeholders	-
Precondition	-
Success garantee	-
Main success scanario	1. Get sensors value from 'Monitoring' 2. Scale sensors value to corresponding temperature value in degrees 3. Display the sensors values 4. Wait for next sampling time 5. GOTO 1.
Extensions	-
Frequency of occurance	While the system is operational
Miscellaneous	-

Table 0-4: Fully Dressed Use Case Document of Home Automation System for 'Configuration' use case

Use case name	Configuration
Scope	The system under design
Level	User goal
Primary actor	Keyboard
Stakeholders	-
Precondition	-
Success garantee	-
Main success scanario	1. Get sensors number 2. Get sensors Id 3. Get sensors type 4. Setup sensors
Extensions	-
Frequency of occurance	Only once while the system is operational
Miscellaneous	-

Appendix 4: System Sequence Diagram (SSD)

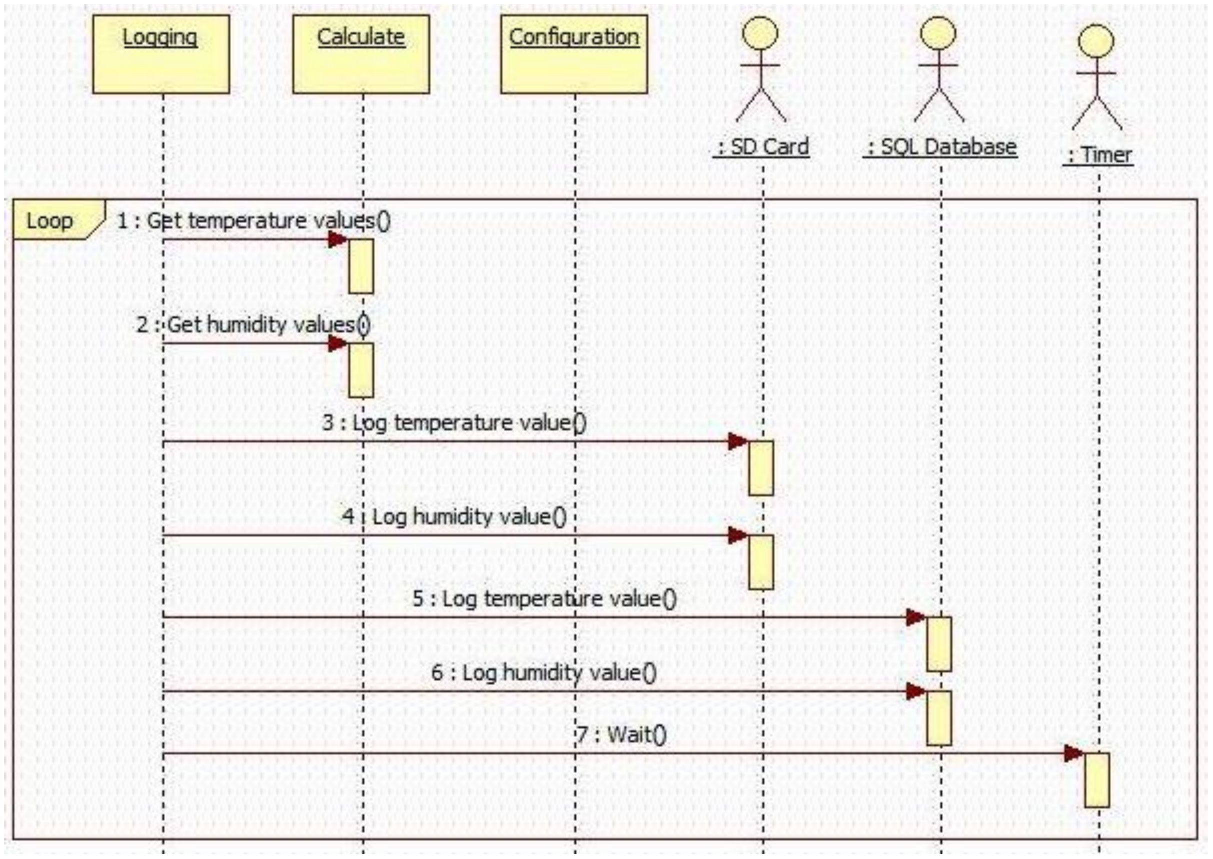


Figure 0-1: System Sequence Diagram (SSD) for 'Logging' Use Case Diagram

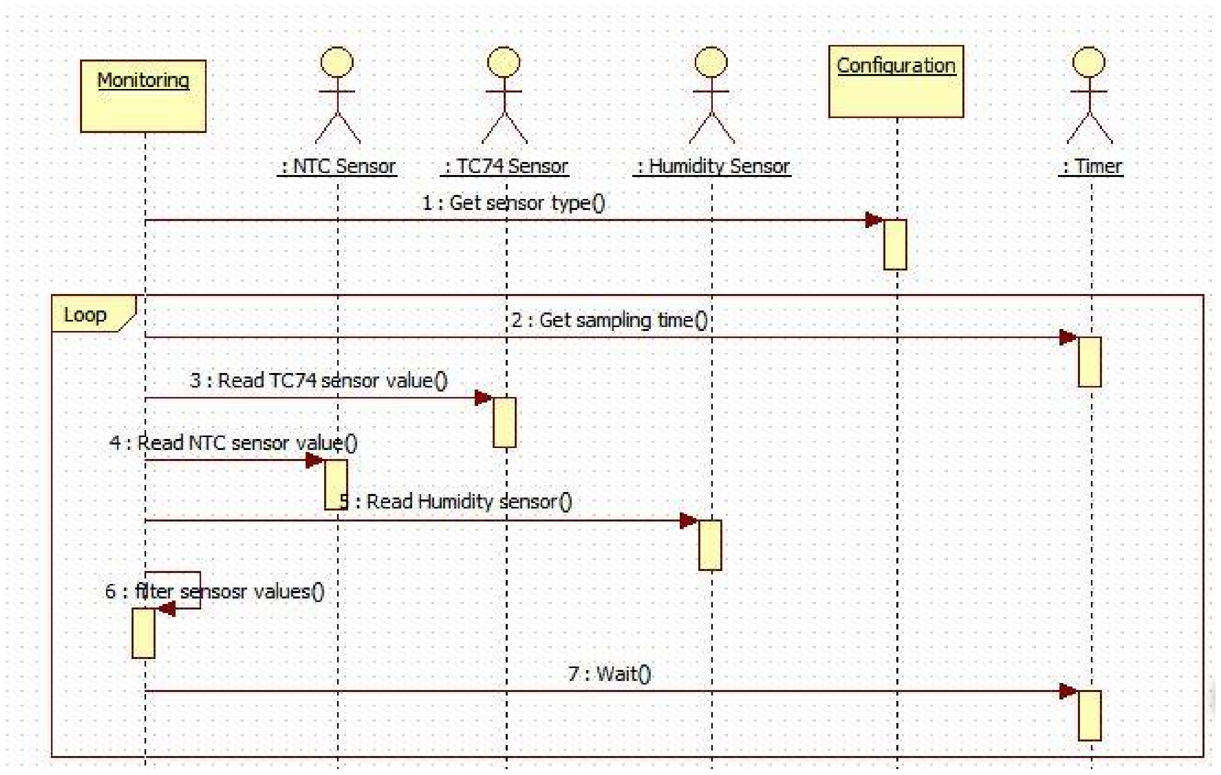


Figure 0-2: System Sequence Diagram (SSD) for 'Monitoring' Use Case Diagram

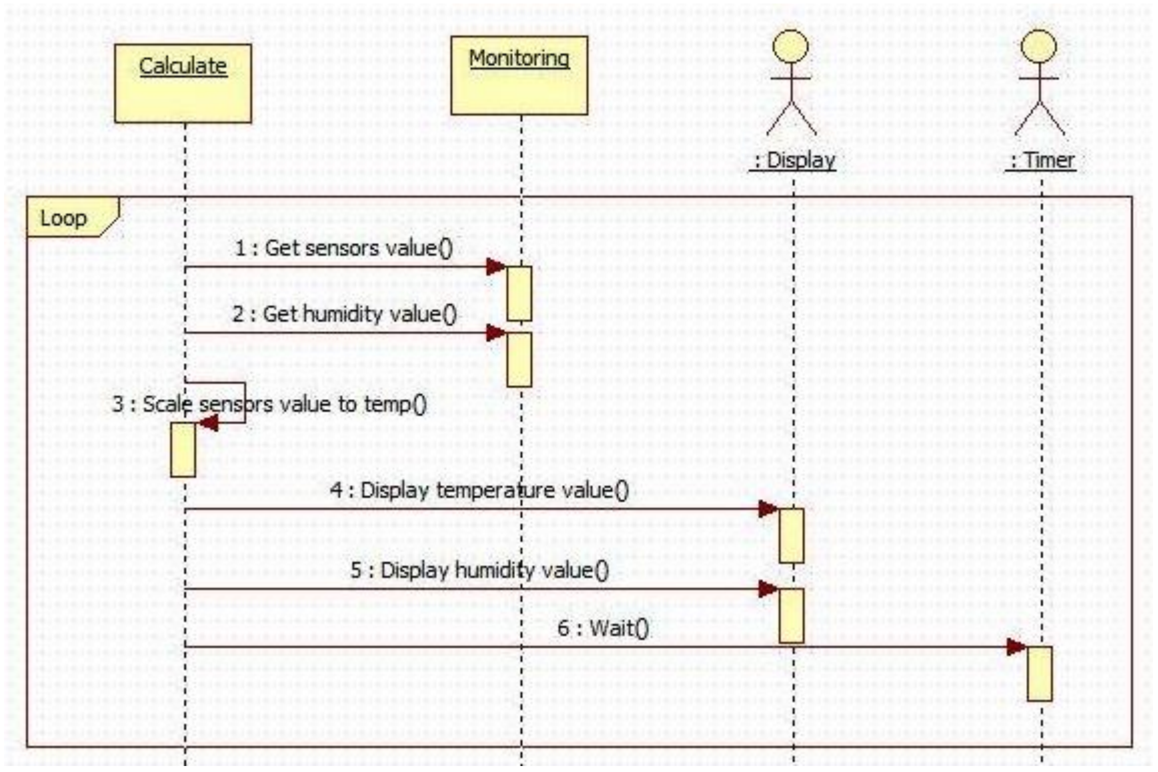


Figure 0-3: System Sequence Diagram (SSD) for 'Calculate' Use Case Diagram

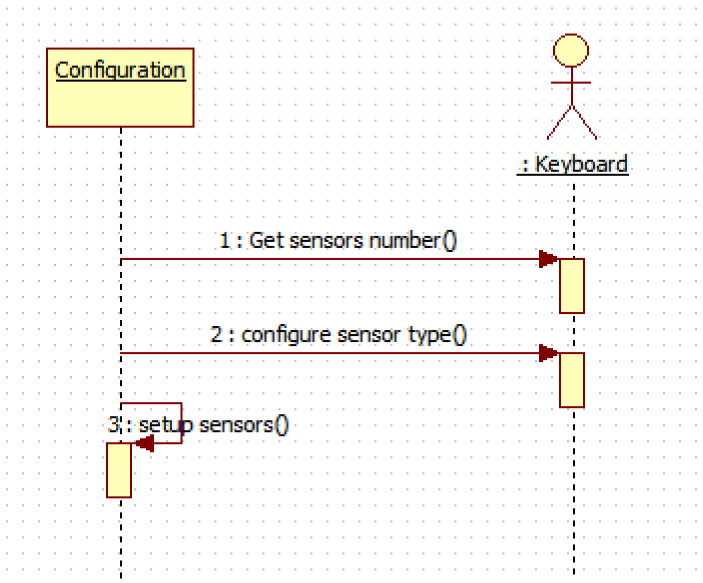


Figure 0-4: System Sequence Diagram (SSD) for 'Configuration' Use Case Diagram

Appendix 5: Sequence Diagrams (SD)

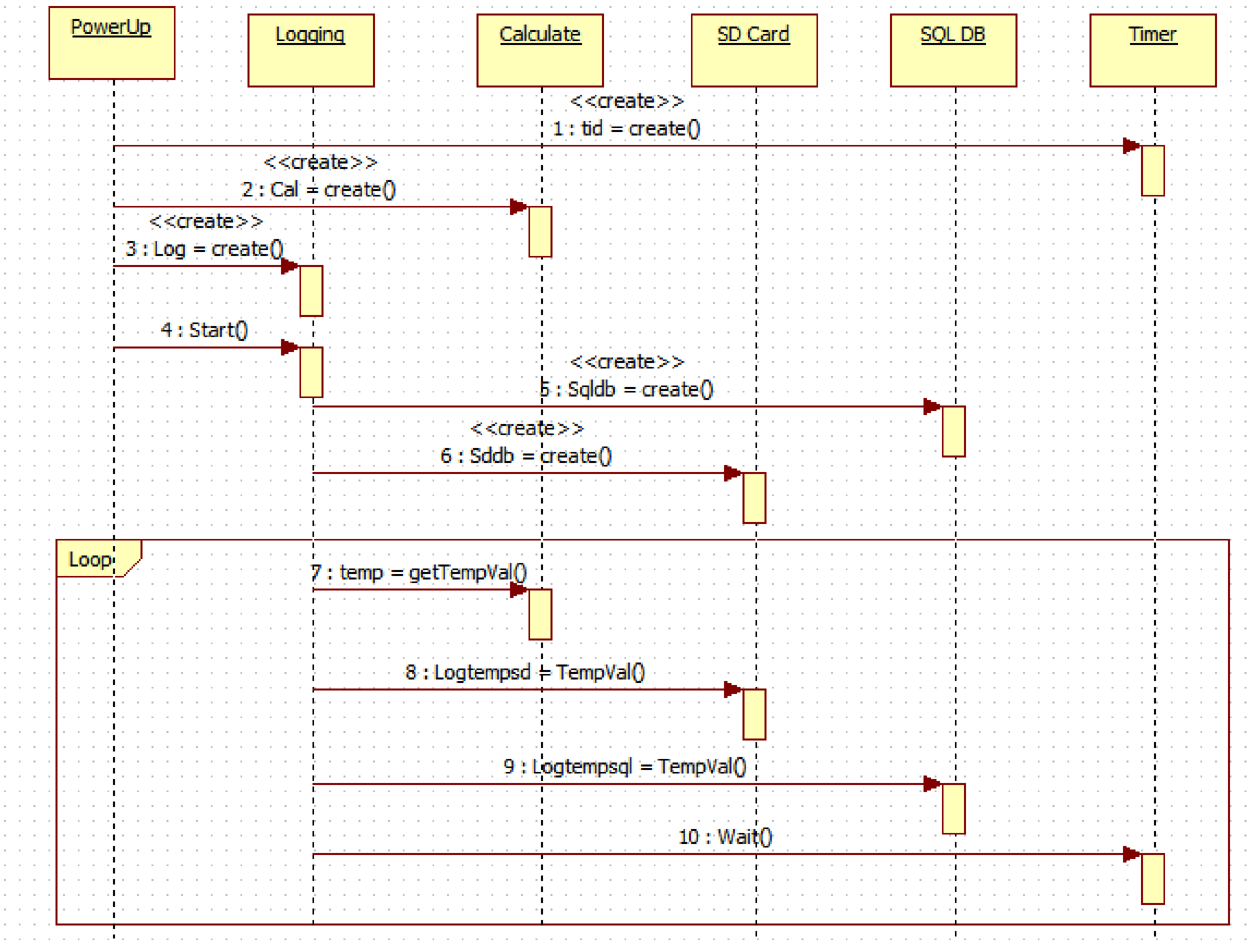


Figure 0-5: Sequence Diagram for 'Logging' Use Case

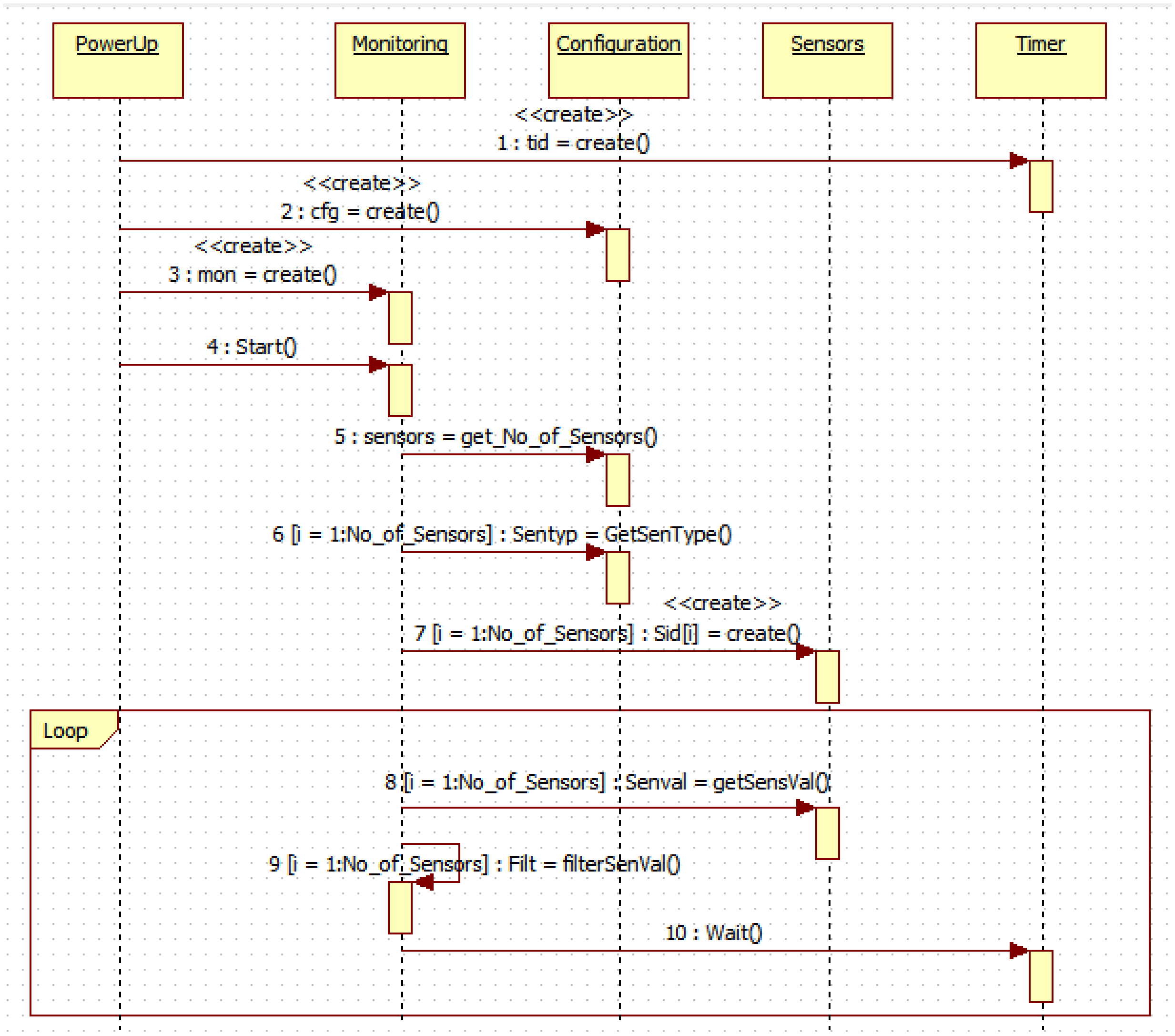

PowerUp
Monitoring
Configuration
Sensors
Timer
<<create>>
1 : tid = create()
<<create>>
2 : cfg = create()
<<create>>
3 : mon = create()
4 : Start()
5 : sensors = get_No_of_Sensors()
6 [i = 1:No_of_Sensors] : Sentyp = GetSenType()
<<create>>
7 [i = 1:No_of_Sensors] : Sid[i] = create()
Loop
8 [i = 1:No_of_Sensors] : Senval = getSensVal()
9 [i = 1:No_of_Sensors] : Filt = filterSenVal()
10 : Wait()

Figure 0-6: Sequence Diagram for 'Monitoring' Use Case

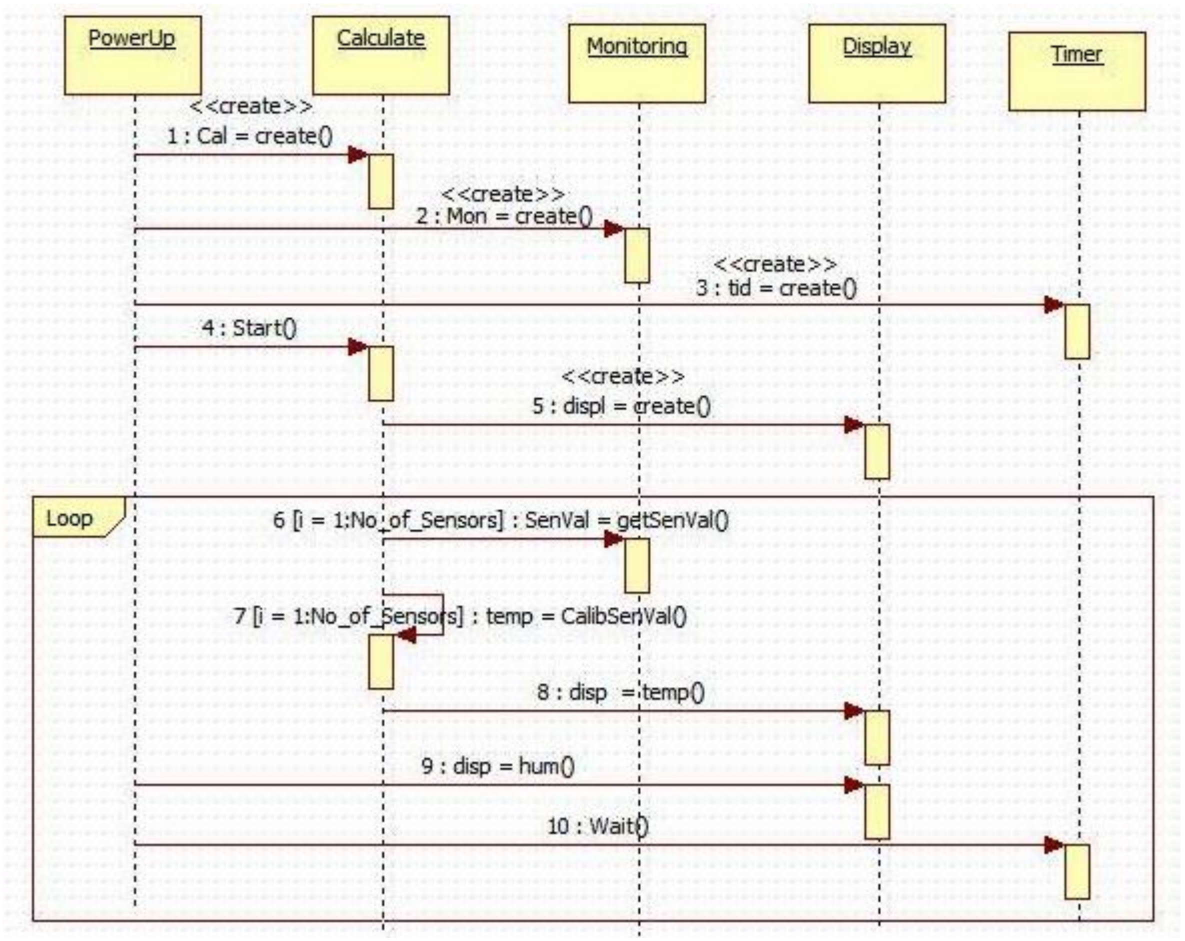


Figure 0-7: Sequence Diagram for 'Calculate' Use Case.

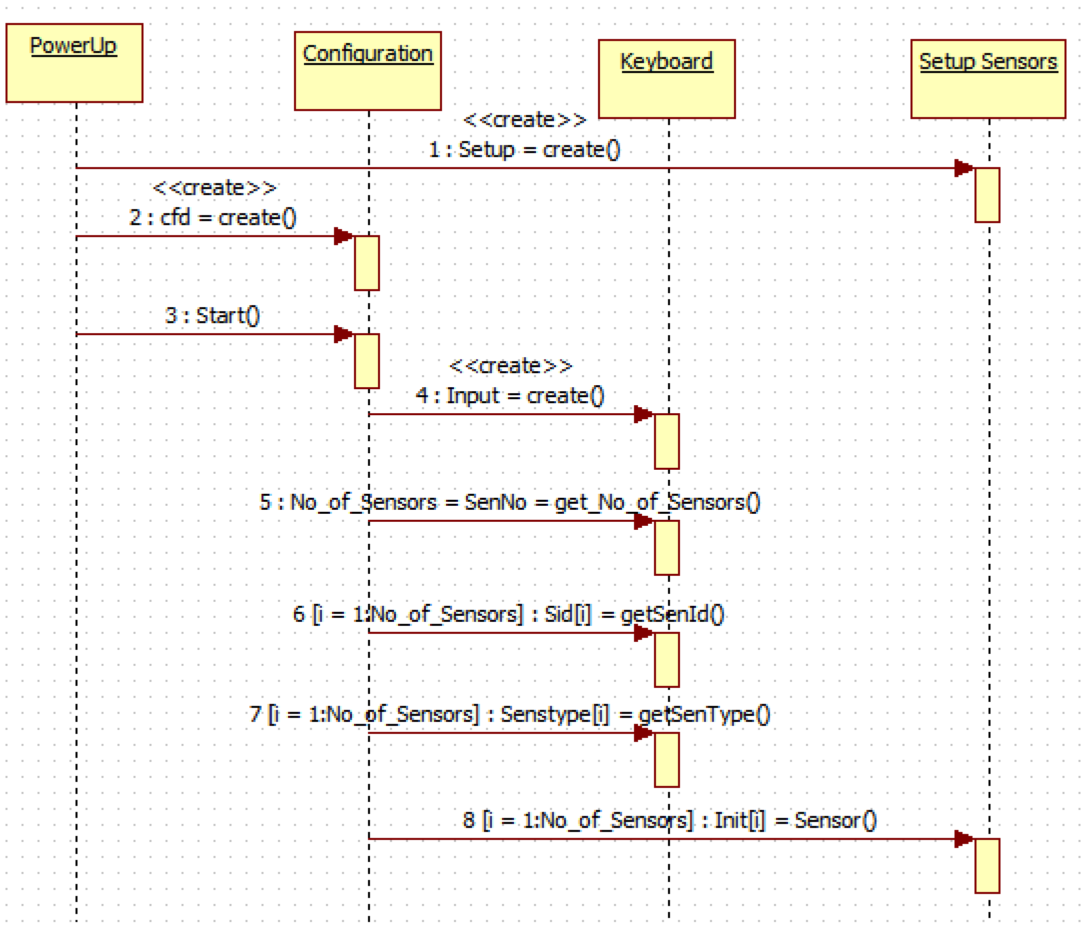

PowerUp
Configuration
Keyboard
Setup Sensors
<<create>>
1 : Setup = create()
<<create>>
2 : cfd = create()
3 : Start()
<<create>>
4 : Input = create()
5 : No_of_Sensors = SenNo = get_No_of_Sensors()
6 [i = 1:No_of_Sensors] : Sid[i] = getSenId()
7 [i = 1:No_of_Sensors] : Senstype[i] = getSenType()
8 [i = 1:No_of_Sensors] : Init[i] = Sensor()

Figure 0-8: Sequence Diagram for 'Configuration' Use Case

Appendix 6: Class Diagram

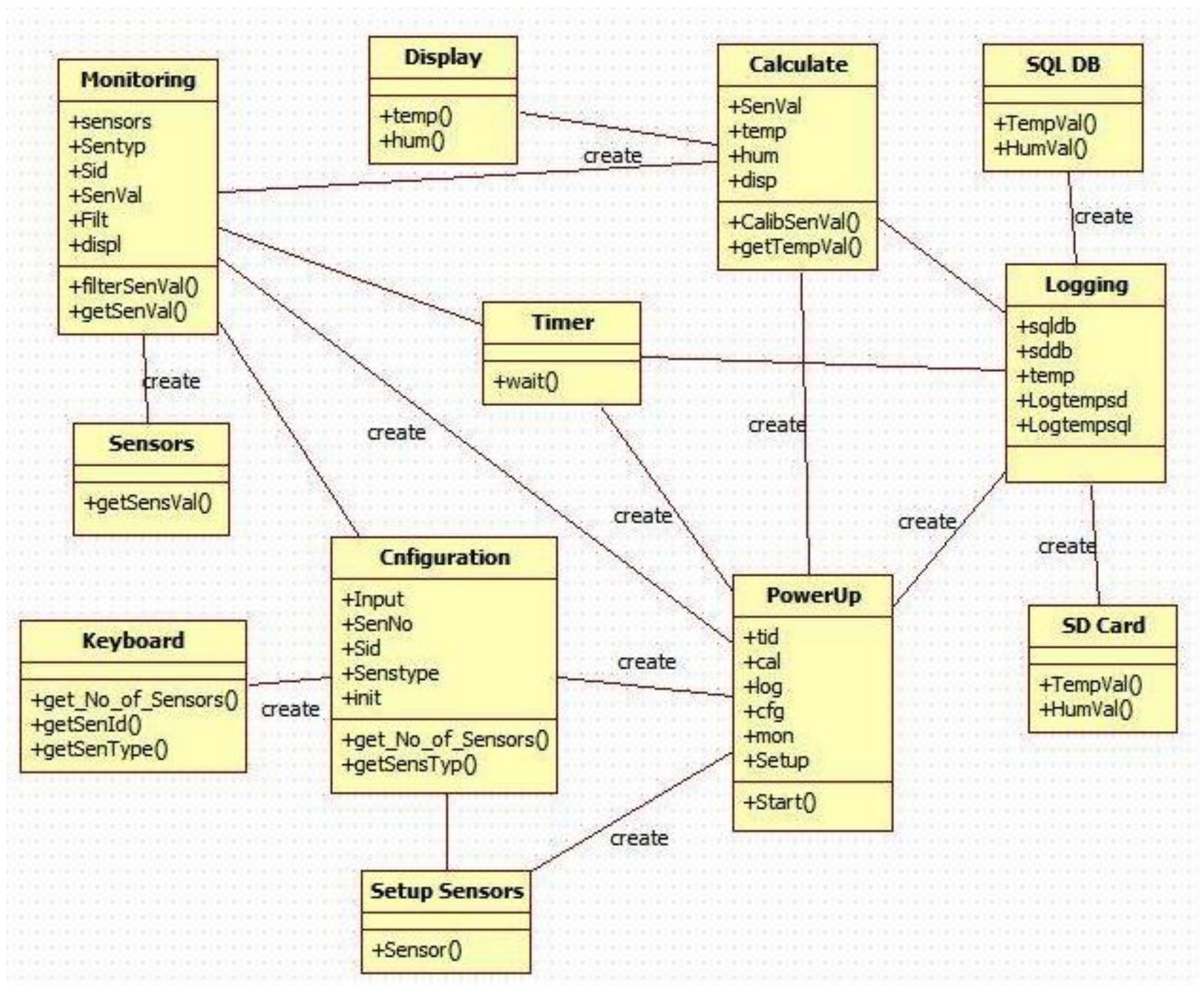


Figure 0-9: Class Diagram for Home Automation System

Appendix 7: Object Diagram

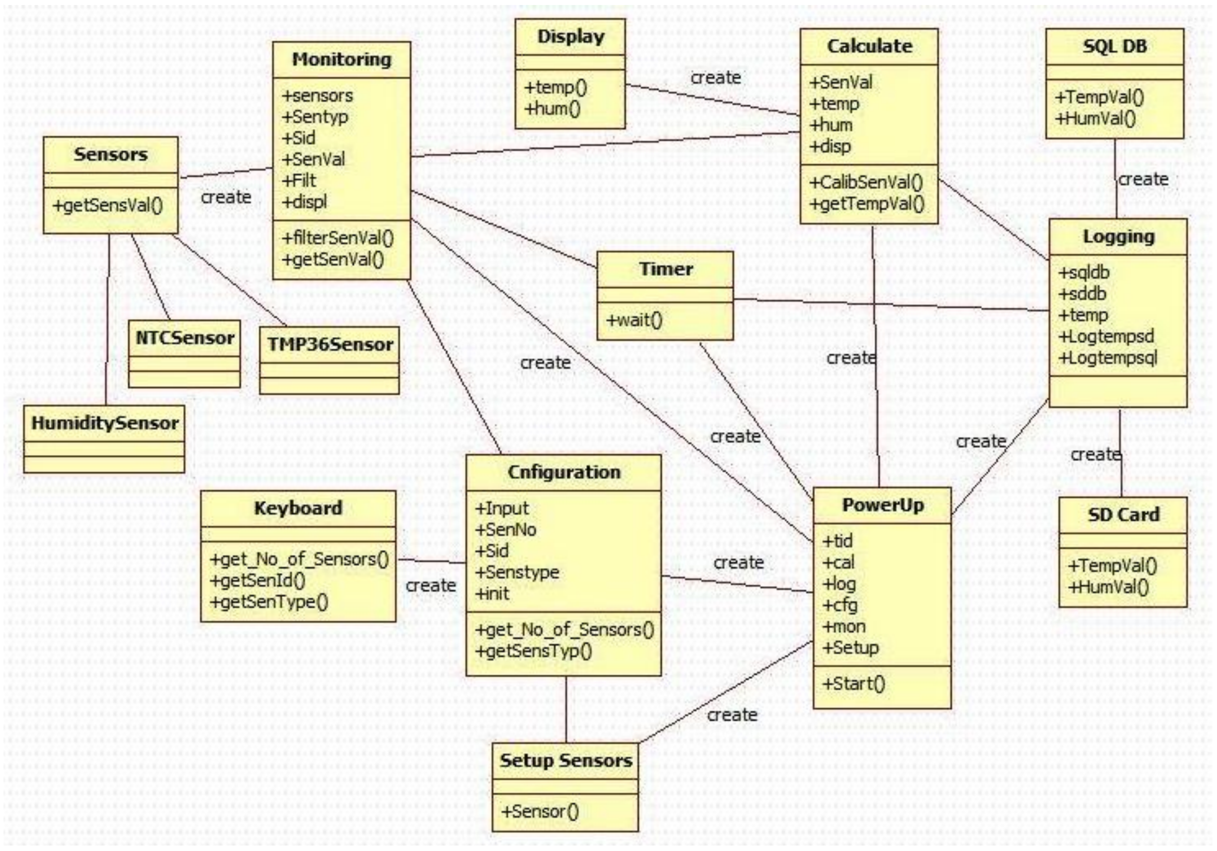


Figure 0-10: Object Diagram for the Home Automation System

Appendix 8: SQL Codes

Code for generating Database in SQL server

```
CREATE TABLE [ALARM]
(
	[AlarmId]          integer  NOT NULL ,
	[DataId]           integer  NULL ,
	[AlarmTime]        datetime  NULL ,
	[AckPerson]        varchar(20)  NULL ,
	[AckTime]          datetime  NULL ,
	[Description]      varchar(200)  NULL ,
	[Status]           varchar(20)  NULL
)
go

ALTER TABLE [ALARM]
	ADD CONSTRAINT [XPKALARM] PRIMARY KEY  CLUSTERED ([AlarmId] ASC)
go

CREATE TABLE [ALARM_CONFIGURATION]
(
	[AlarmConfigurationId] integer  identity (1,1) NOT NULL ,
	[TagId]            integer  NULL ,
	[PriorityId]       integer  NULL ,
	[UpperLimit]       varchar(20)  NULL ,
	[LowerLimit]       varchar(20)  NULL ,
	[Status]           varchar(20)  NULL
)
go

ALTER TABLE [ALARM_CONFIGURATION]
	ADD CONSTRAINT [XPKALARM_CONFIGURATION] PRIMARY KEY  CLUSTERED
([AlarmConfigurationId] ASC)
go

CREATE TABLE [ALARM_PRIORITY]
(
	[PriorityId]       integer identity (1,1) NOT NULL ,
	[Name]             varchar(20)  NULL ,
	[Descrition]       varchar(20)  NULL
)
go

ALTER TABLE [ALARM_PRIORITY]
	ADD CONSTRAINT [XPKALARM_PRIORITY] PRIMARY KEY  CLUSTERED ([PriorityId] ASC)
go

CREATE TABLE [DEVICE]
(
	[DeviceId]         integer  identity (1,1) NOT NULL ,
	[DeviceTypeId]     integer  NULL ,
	[DeviceName]       varchar(20)  NULL ,
	[PrimaryDevice]    varchar(20)  NULL ,
	[UniqueKey]        varchar(20)  NULL ,
	[Description]      varchar(200)  NULL ,
        [Status]          varchar(20)  NULL
)
go

ALTER TABLE [DEVICE]
        ADD CONSTRAINT [XPKDEVICE] PRIMARY KEY  CLUSTERED ([DeviceId] ASC)
go

CREATE TABLE [DEVICE_TYPE]
(
        [DeviceTypeId]      integer identity (1,1) NOT NULL ,
        [DeviceTypeName]    varchar(20)  NULL ,
        [Category]          varchar(20)  NULL ,
        [PrimaryDevice]     varchar(20)  NULL ,
        [UniqueKey]         integer  NULL ,
        [Description]       varchar(200)  NULL
)
go

ALTER TABLE [DEVICE_TYPE]
        ADD CONSTRAINT [XPKDEVICE_TYPE] PRIMARY KEY  CLUSTERED ([DeviceTypeId] ASC)
go

CREATE TABLE [TAG]
(
        [TagId]           integer identity (1,1) NOT NULL ,
        [TagTypeId]       integer  NULL ,
        [TagGroupId]      integer  NULL ,
        [DeviceId]        integer  NULL ,
        [TagName]         varchar(20)  NULL ,
        [LoggingTime]     datetime  NULL ,
        [Status]          varchar(20)  NULL ,
        [Description]     varchar(200)  NULL
)
go

ALTER TABLE [TAG]
        ADD CONSTRAINT [XPKTAG] PRIMARY KEY  CLUSTERED ([TagId] ASC)
go

CREATE TABLE [TAG_GROUP]
(
        [TagGroupId]      integer identity (1,1) NOT NULL ,
        [TagGroupName]    varchar(20)  NULL ,
        [Description]     varchar(200)  NULL
)
go

ALTER TABLE [TAG_GROUP]
        ADD CONSTRAINT [XPKTAG_GROUP] PRIMARY KEY  CLUSTERED ([TagGroupId] ASC)
go

CREATE TABLE [TAG_TYPE]
(
        [TagTypeId]       integer identity (1,1) NOT NULL ,
        [Description]     varchar(200)  NULL ,
        [Unit]            varchar(20)  NULL ,
        [TagType]         varchar(20)  NULL
)
```

```
go

ALTER TABLE [TAG_TYPE]
	ADD CONSTRAINT [XPKTAG_TYPE] PRIMARY KEY  CLUSTERED ([TagTypeId] ASC)
go

CREATE TABLE [TAGDATA]
(
	[DataId]           integer identity (1,1) NOT NULL ,
	[TagId]            integer  NULL ,
	[TagValue]         varchar(20)  NULL ,
	[Date]             datetime  NULL ,
	[Status]           varchar(20)  NULL ,
	[AlarmStatus]      varchar(20)  NULL
)
go

ALTER TABLE [TAGDATA]
	ADD CONSTRAINT [XPKTAGDATA] PRIMARY KEY  CLUSTERED ([DataId] ASC)
go

CREATE TABLE [USER]
(
	[UserLevelId]      integer identity (1,1) NOT NULL ,
	[Name]             varchar(20)  NULL ,
	[UserName]         varchar(20)  NULL ,
	[Password]         char(18)  NULL ,
	[Email]            varchar(20)  NULL ,
	[Description]      varchar(200)  NULL ,
	[MainAdmin]        varchar(20)  NULL ,
	[UserId]           integer  NOT NULL
)
go

ALTER TABLE [USER]
	ADD CONSTRAINT [XPKUSER] PRIMARY KEY  CLUSTERED ([UserId] ASC)
go

CREATE TABLE [USER_LEVEL]
(
	[UserLevelId]      integer identity (1,1) NOT NULL ,
	[UserLevel]        varchar(20)  NULL ,
	[Description]      varchar(200)  NULL
)
go

ALTER TABLE [USER_LEVEL]
	ADD CONSTRAINT [XPKUSER_LEVEL] PRIMARY KEY  CLUSTERED ([UserLevelId] ASC)
go


ALTER TABLE [ALARM]
	ADD CONSTRAINT [R_1] FOREIGN KEY ([DataId]) REFERENCES [TAGDATA]([DataId])
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
go


ALTER TABLE [ALARM_CONFIGURATION]
```

```
        ADD CONSTRAINT [R_2] FOREIGN KEY ([PriorityId]) REFERENCES [ALARM_PRIORITY]([PriorityId])
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
go

ALTER TABLE [ALARM_CONFIGURATION]
        ADD CONSTRAINT [R_3] FOREIGN KEY ([TagId]) REFERENCES [TAG]([TagId])
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
go


ALTER TABLE [DEVICE]
        ADD CONSTRAINT [R_4] FOREIGN KEY ([DeviceTypeId]) REFERENCES
[DEVICE_TYPE]([DeviceTypeId])
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
go


ALTER TABLE [TAG]
        ADD CONSTRAINT [R_5] FOREIGN KEY ([TagTypeId]) REFERENCES [TAG_TYPE]([TagTypeId])
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
go

ALTER TABLE [TAG]
        ADD CONSTRAINT [R_6] FOREIGN KEY ([DeviceId]) REFERENCES [DEVICE]([DeviceId])
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
go

ALTER TABLE [TAG]
        ADD CONSTRAINT [R_7] FOREIGN KEY ([TagGroupId]) REFERENCES [TAG_GROUP]([TagGroupId])
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
go


ALTER TABLE [TAGDATA]
        ADD CONSTRAINT [R_11] FOREIGN KEY ([TagId]) REFERENCES [TAG]([TagId])
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
go


ALTER TABLE [USER]
        ADD CONSTRAINT [R_8] FOREIGN KEY ([UserLevelId]) REFERENCES [USER_LEVEL]([UserLevelId])
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
go
```

Appendix 9: Coding based on Analysis and Software Design

Code for the Aggregator Node (Arduino Bluetooth_ Master)

```
/*
The Arduino is configured as Aggregator (master) receiving temperature values
from the Arduino sensor node (Slave).
*/

#include <SoftwareSerial.h>                    // Software Serial Port

#define Rx 6
#define Tx 7



#define DEBUG_ENABLED  1

String retSymb = "+RTINQ=";          // start when any return is observed
String slaveName = ";BLUBTSlave";    // Name of the Arduino sensor node (Slave name)
int Indexname = 0;   //nameIndex = 0;
int addrIndex = 0;

String recvBuf;
String slaveAddr;
String connCmd = "\r\n+CONN=";

SoftwareSerial blueToothSerial(Rx,Tx);

void setup()
{
   Serial.begin(9600);
   pinMode(Rx, INPUT);
   pinMode(Tx, OUTPUT);
   Bluetoothconfiguration();

   /*wait 1000ms to flush the serial buffer*/
   delay(1000);
   Serial.flush();
   blueToothSerial.flush();
}

/* Checking if there is any Bluetooth serial available */
void loop()
{
   char recvChar;
   while(1)
   {
        if(blueToothSerial.available())         //check if there's any data sent from the remote
bluetooth shield
        {
            recvChar = blueToothSerial.read();
            Serial.print(recvChar);
        }
        if(Serial.available())                  //check if there's any data sent from the local serial terminal,
you can add the other applications here
        {
            recvChar  = Serial.read();
            blueToothSerial.print(recvChar);
        }
    }
    delay(2000);
}

void Bluetoothconfiguration() //setupBlueToothConnection()
{
    Serial.println("t");  //Newly Added
    blueToothSerial.begin(38400);                               // Bluetooth BaudRate
    blueToothSerial.print("\r\n+STWMOD=1\r\n");                 // setting Bluetooth to work in
master mode
    blueToothSerial.print("\r\n+STNA=BLUBTMaster\r\n");         // setting Bluetooth name as
"BLUBTMaster"
    blueToothSerial.print("\r\n+STAUTO=0\r\n");                 // Forbiden Auto Connection
```

```
  delay(2000);                                    // delay for 2seconds.

  blueToothSerial.flush();

  blueToothSerial.print("\r\n+INQ=1\r\n");          //make the master inquirable

  Serial.println("Master is inquirable!");

  delay(2000);


  /* Locating the target sensor node (slave) */

  char recvChar;

  while(1)

  {

    if(blueToothSerial.available())

    {

      recvChar = blueToothSerial.read();

      recvBuf += recvChar;

      Indexname = recvBuf.indexOf(slaveName);          //get the position of slave name


                                          //Indexname -= 1;


      if ( Indexname != -1 )

      {

        Serial.print(recvBuf);

        /* geting the start position of sensor node (Arduino slave) address */

        addrIndex = (recvBuf.indexOf(retSymb,(Indexname - retSymb.length()- 18) ) +
retSymb.length());
```

```
            /* geting the string of slave address */
            slaveAddr = recvBuf.substring(addrIndex, Indexname);
            break;
        }
    }
}

/* The connection command */
connCmd += slaveAddr;
connCmd += "\r\n";
int connectionOK = 0;
Serial.print("Connecting to slave:");
Serial.print(slaveAddr);
Serial.println(slaveName);

//connect the sensor node (slave) till they are connected
do
{
    blueToothSerial.print(connCmd);//send connection command
    recvBuf = "";
    while(1)
    {
        if(blueToothSerial.available()){
            recvChar = blueToothSerial.read();
            recvBuf += recvChar;
            if(recvBuf.indexOf("CONNECT:OK") != -1)
            {
              connectionOK = 1;
              Serial.println("Connected!");
              blueToothSerial.print("Connected!");
              break;
            }
            else if(recvBuf.indexOf("CONNECT:FAIL") != -1)
            {
              Serial.println("Connect again!");
              break;
            }
          }
        }
    }while(0 == connectionOK);
}
```

Code for Sensor Node Arduino (Slave)

```
/*
Temperature Measurement using Arduino Sensor Node (Slave) and transmit values via
Bluetooth shield
to Agregator Node for Logging

*/

/* using Software Serial Port */
#include <SoftwareSerial.h>
#define RxD 6 //9
#define TxD 7 //8

#define DEBUG_ENABLED  1

#define NTC_PIN_TEMP    A1

SoftwareSerial blueToothSerial(RxD,TxD);

void setup()
{
   Serial.begin(9600);
   pinMode(RxD, INPUT);
   pinMode(TxD, OUTPUT);
```

```
    configuration();
}


/* Reading NTC sensor values */
int getSensVal()
{
   int a = analogRead(NTC_PIN_TEMP);


   return (a);
         }


/* Implementing filter */
int filterSenVal()
{
    double FilterOutput;
    double Tf = 0.00005;
    double Ts = 1;
    double Filt;
    double y_init = 0;                  //Initial filter output
    double a = Ts/(Tf+Ts);
    double u_k;                         //Sensor output signal
    u_k = getSensVal();


    Filt = (1-a)*y_init + a*u_k;        //y_k is the output filter
```

```
        y_init = FilterOutput;


    return ((int)Filt);                //Return filter output
}


/* Calibrating or Scaling of NTC sensor values to corresponding temperature */
int CalibSensVal()  //getTemp()
{
    /* Getting the analog filtered sensor output */
    int a = filterSenVal();     //analogRead(NTC_PIN_TEMP);
    //int B=3975;  //Constant value for used for conversion


    /* Converting the anlaog filtered sensor output to corresponding resistance value
       Using Steinhart-Hart Thermistor Equation:
       Where A = 0.001129148; B = 0.000234125; C = 0.0000000876741;
    */
    float resistance = (float)(1023-a)*10000/a;
    float LogRes = log(resistance);


    /* Converting rsistance value to the corresponding temperature value in degrees */
    float Temperature = 1 / (0.001129148 + (0.000234125 * LogRes) + (0.0000000876741 *
LogRes * LogRes * LogRes));


    /* Converting temperture in Kelvin to Celsius */
    float temp = Temperature - 273.15;

    /* Alternative equation here can also be used where B=3975:
    float temp = temp=1/(log(resistance/10000)/B+1/298.15)-273.15;
    */

    return (int)temp;
}

void loop()
{
  Serial.print("NTC Temperature Value[Degree C]: ");
  //Serial.println(CalibSensVal() /*getSensVal()*/ /* filterSenVal()*/);

    char recvChar;
    while(1)
    {
        if(blueToothSerial.available())
        {//check if there's any data sent from the remote bluetooth shield
            recvChar = blueToothSerial.read();
            Serial.print(recvChar);

            if(recvChar == 't' || recvChar == 'T')
            {
```

```
            blueToothSerial.print("NTC Temperature Value[Degree C]: ");
            blueToothSerial.println(CalibSensVal());
            Serial.print(CalibSensVal());
            blueToothSerial.print("\n");
             }
        }
        if(Serial.available())
        {//check if there's any data sent from the local serial terminal, you can add the other
applications here
           recvChar  = Serial.read();
           blueToothSerial.print(recvChar);
        }
   }
   delay(2000);
}

void configuration()
{
   blueToothSerial.begin(38400);                          // Bluetooth BaudRate
   blueToothSerial.print("\r\n+STWMOD=0\r\n");            // setting Bluetooth to work in slave
mode
   blueToothSerial.print("\r\n+STNA=BLUBTSlave\r\n");     // setting Bluetooth name as
"BLUBTSlave"
   blueToothSerial.print("\r\n+STOAUT=1\r\n");            // Allowing Paired device(s) to connect
to this Bluetooth
```

```
blueToothSerial.print("\r\n+STAUTO=0\r\n");             // Forbiden Auto Connection

delay(2000);                                        // delay for 2seconds.

blueToothSerial.print("\r\n+INQ=1\r\n");                // make the slave bluetooth inquirable

Serial.println("The slave bluetooth is inquirable!");

delay(2000);                                        // delay

blueToothSerial.flush();

  }
```

Code for Monitoring Temperature using Raspberry Pi

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Windows.Storage;
using System.IO;

namespace temp_TC74
{
    class LogData
    {
        string _filename = "logTempData.csv";
        StorageFile file;
        Stream _write;
        int nLineCount = 0;
        public string Filepath { get; private set; }
        public LogData(string filename = null)
        {
            if (filename != null)
                _filename = filename;
            InitFile();
        }
        async void InitFile()
        {
            // create a file with the given filename in the local folder; replace any
existing file with the same name
            file = await
Windows.Storage.ApplicationData.Current.LocalFolder.CreateFileAsync(_filename,
CreationCollisionOption.ReplaceExisting);
            Filepath = Windows.Storage.ApplicationData.Current.LocalFolder.Path;
            _write = await file.OpenStreamForWriteAsync();
        }
        public void WriteData(double fTempTC74)
        {
            nLineCount++;

            string content =  fTempTC74.ToString("0.0")   + "*C ,"     +
DateTime.Now.ToString()+ "%\r\n";//fTempTC74.ToString("0.0") + "C ," +
fTempHIH6100.ToString("0.0") + "C ," + fHumidityHIH6100.ToString("0.0") + "%\r\n";
            // saves the string 'content' to a file 'filename' in the app's local storage
folder
            byte[] fileBytes = System.Text.Encoding.UTF8.GetBytes(content.ToCharArray());

            // write the char array created from the content string into the file
            _write.Write(fileBytes, 0, fileBytes.Length);
            _write.Flush();

        }

    }

}
```

References

[1] M. Rouse, "Home Automation," 2010.

[2] R. Harper, *Inside the Smart Home*, 2003.

[3] A. Inc. (2015). *HomeKit*. Available: https://developer.apple.com/homekit/

[4] A. Grush. (2015). *Project Brillo is Google's new Internet of Things OS*. Available: http://www.androidauthority.com/project-brillo-google-io-612159/

[5] M. Elgan. (2015). *How Google and Apple will smartify your home*. Available: http://www.computerworld.com/article/2925959/internet-of-things/how-google-and-apple-will-smartify-your-home.html

[6] W. India. (2015). *Will These 3 Biggies Change The Future of Smart Homes?* Available: http://gosmartbricks.com/the-future-of-smart-homes/

[7] S. H. Energy. (2015). *What is a Smart Home*. Available: http://smarthomeenergy.co.uk/what-smart-home

[8] Arduino. (2015). *Arduino*. Available: https://www.arduino.cc/Hacking

[9] Raspberry, "Raspberry Pi 2 Model B," 2015.

[10] Microsoft. (2015). *Windows Dev Center*. Available: http://ms-iot.github.io/content/en-US/win10

[11] D. I. Inc, "XBEE RF Module," 2009.

[12] Adafruit. (2015). *NTC Temperature Sensor*. Available: https://learn.adafruit.com/thermistor/overview

[13] EPCOS. (2015). *NTC Datasheet-NTC thermistors for temperature measurement*. Available: www.farnell.com/datasheets/1598315.pdf

[14] vishay. (2012). *Resistive Products-NTC Thermistors*. Available: www.vishay.com

[15] RS-Online. (2015). *Temperature and Humidity sensor*. Available: http://no.rs-online.com/web/p/temperature-humidity-sensors/7811365/

[16] Wikipedia. (2015, November 4). *Ethernet*. Available: https://en.wikipedia.org/wiki/Ethernet

[17] Google. (2015, November 4). *Twisted pair cable color code*. Available: https://www.google.no/search?q=twisted+pair+cable+color+code&rlz=1C1CHJW_enNO660NO660&espv=2&biw=1366&bih=623&tbm=isch&imgil=Kv3Pu7rk63W11M%253A%253B9zRbgJh8RXhwsM%253Bhttp%25253A%25252F%25252Fwww.incentre.net%25252Ftech-support%25252Fother-support%25252Fethernet-cable-color-coding-diagram%25252F&source=iu&pf=m&fir=Kv3Pu7rk63W11M%253A%252C9zRbgJh8RXhwsM%252C_&dpr=1&usg=__d-kSBK1JZ9nD4GouPwP0qLrOjS8%3D&ved=0CC0QyjdqFQoTCJ-E1aPJgckCFcOQLAodWm4Onw&ei=BKQ_Vp_yHMOhsgHa3Ln4CQ#tbm=isch&q=twisted+pair+cable+color+code&imgrc=gxOo5pRf1jNomM%3A

[18] Study.Com. (2003, October 8, 2015). *Medium-Range Wireless Communication: Wi-Fi & Hotspots*. Available: http://study.com/academy/lesson/medium-range-wireless-communication-wi-fi-hotspots.html

[19] Geeetech. (2013, November 11). *Arduino Bluetooth shield*. Available: http://www.geeetech.com/wiki/index.php/Arduino_Bluetooth_shield

[20] EngineersGarage. (2012, November 11). *Difference between Bluetooth and Zigbee Technologies*. Available: http://www.engineersgarage.com/contribution/zigbee-vs-bluetooth

[21] N.-O. Skeie, *Object-Oriented Analysis, Design, and Programming using UML and C#*, 2014.

[22] N.-O. Skeie, *'SCE1306: Object-oriented analysis, design, and programming', Lecture notes for the master course SCE1306 at the Telemark University College (TUC).* 2015.

[23] Wikipedia. (2015, November 17). *Domain model*. Available: https://en.wikipedia.org/wiki/Domain_model

[24] Wikipedia. (2010, November 16). *Use case*. Available: http://en.wikipedia.org/wiki/Usecase#Actors.

[25] N.-O. Skeie, *Object Oriented Analysis Design and Programming*, 2014.

[26] Asus. (2013). *Keyboard Device Filter*. Available: http://rog.asus.com/forum/showthread.php?31509-Is-the-ASUS-Keyboard-Device-Filter-Needed

[27] P. Zandbergen. (2003 - 2015). *Short-Range Wireless Communication: Bluetooth, ZigBee & Infrared Transmission*. Available: http://study.com/academy/lesson/short-range-wireless-communication-bluetooth-zigbee-infrared-transmission.html

[28] I. Studio. (2010). *HC-05 Bluetooth Module*.

[29] K. Prajapati, *Process Control and Monitoring using Arduino and Raspberry*, 2015.