PROJECT TITLE

"TEMPERATURE AND HUMIDITY MONITORING AND ALERT SYSTEM"

Submitted By

ALI ASGAR TASHRIFWALA **ai229@njit.edu**

YASH MAHAJAN **ym248@njit.edu**

ADITYA BANSODE **aab87@njit.edu**

UNDER THE GUIDANCE OF

PROF. SUBRAMANIAN NAGANATHAN

18 S - ECE 692102 - Embedded Computing Systems
Department of Electronics & Computer Engineering

ACKNOWLEDGEMENTS

We take this opportunity to express our profound gratitude and deep regards to our guide **Prof. Subramanian Naganathan** for his exemplary guidance, monitoring and constant encouragement throughout the course of this project work.

We also take this opportunity to express a deep sense of gratitude to **Barsha Jain,** Teaching Assistant of ECE-692 course for her cordial support and valuable information which helped us in completing this task through various stages.

We are obliged to stock room staff members of **New Jersey Institute of Technology**, for providing us the microcontroller boards and equipment. We are grateful for their cooperation during the period of our project work.

Lastly, we thank almighty, our parents, our family and friends for their constant encouragement.

CONTENTS

Chapter 1

Project Overview

Home is the place which protects us and our loved ones from cold weather, rainy days and hot summer. Thus, we must take steps to prevent it from disaster like fire. The already existing system only informs the fire department in case of fire. The residence and neighbors do not know of the situation if they are outside. A real time notification will help if there are any children inside the house.

This project aims to notify the residents and neighbors regarding the FIRE ALERT through SMS and call. This will help to prevent any major loss to life or property. The neighbors will be notified which will help them to leave the building immediately and carry any important and personal belongings with them.

Chapter 2

Analysis & Design

2.1 Theory

2.1.1 FRDM KL25z

The FRDM-KL25Z is an ultra-low-cost development platform for Kinetis L Series KL1x (KL14/15) and KL2x (KL24/25) MCUs built on ARM® Cortex™-M0+ processor. Features include easy access to MCU I/O, battery-ready, low-power operation, a standard-based form factor with expansion board options and a built-in debug interface for flash programming and run-control. The FRDM-KL25Z is supported by a range of NXP and third-party development software.

FRDM-KL25Z can be used to evaluate the KL1 and KL2 Kinetis L series devices. It features a KL25Z128VLK, a KL2 family device boasting a max operating frequency of 48MHz, 128KB of flash, a full speed USB controller, and loads of analog and digital peripherals. The FRDM-KL25Z hardware is formfactor compatible with the Arduino™ R3 pin layout, providing a broad range of expansion board options. The on-board interfaces include an RGB LED, a 3-axis digital accelerometer, and a capacitive touch slider.

The FRDM-KL25Z has been designed by NXP in collaboration with mbed for prototyping all sorts of devices, especially those requiring the size and price point offered by Cortex-M0+ and the power of USB Host and Device. It is packaged as a development board with connectors to break out to strip board and breadboard and includes a built-in USB FLASH programmer.

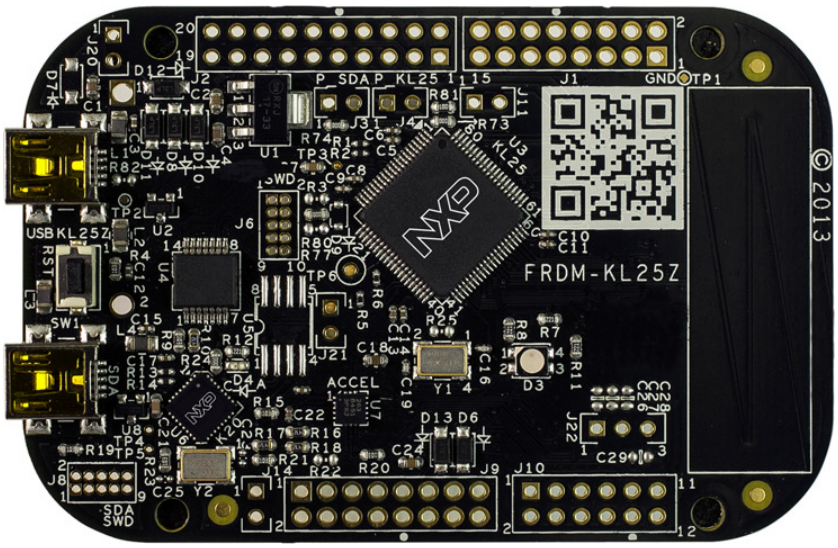


Figure 1 FRDM-KL25Z Board.

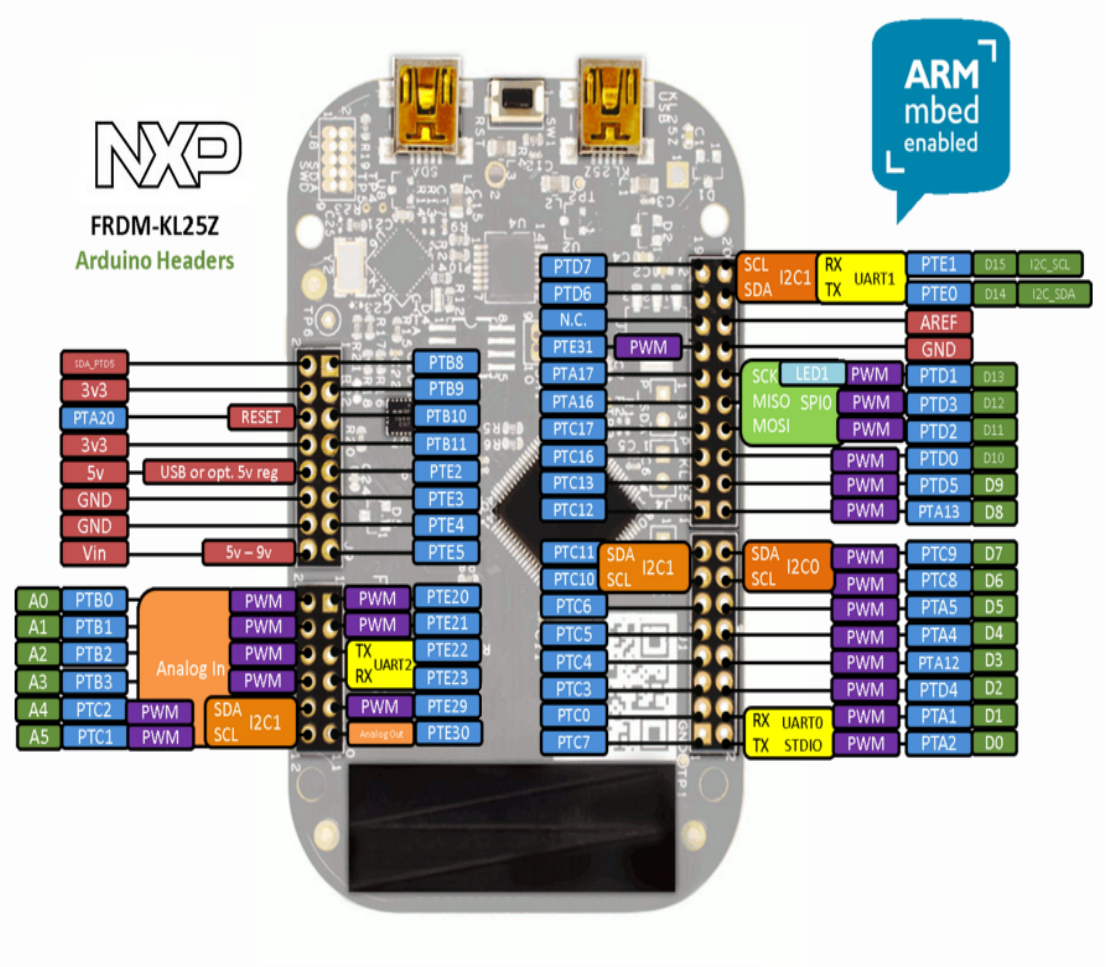


Figure 2 PIN Diagram

The features of the FRDM-KL25Z include:

- MKL25Z128VLK4 in an 80 LQFP package
- Capacitive touch slider - MMA8451Q accelerometer
- Tri-color (RGB) LED
- Flexible power supply options – USB, coin cell battery, external source
- Battery-ready, power-measurement access points
- Easy access to MCU I/O via Arduino ™ R3 compatible I/O connectors
- Programmable OpenSDA debug interface with multiple applications available including: - Mass storage device flash programming interface - P&E Debug interface provides run-control debugging and compatibility with IDE tools - CMSIS-DAP interface: new ARM standard for embedded debug interface - Data logging application

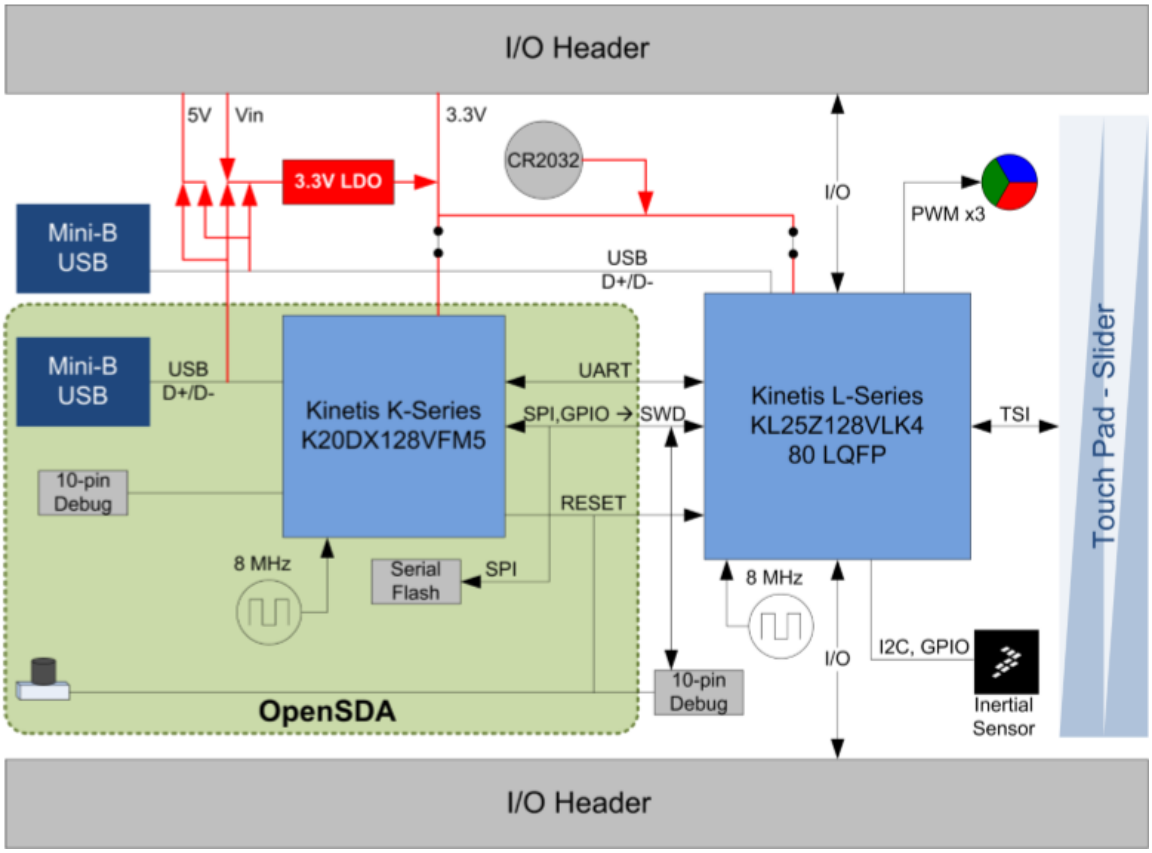


Figure 3 The primary components and their placement on the hardware block diagram.

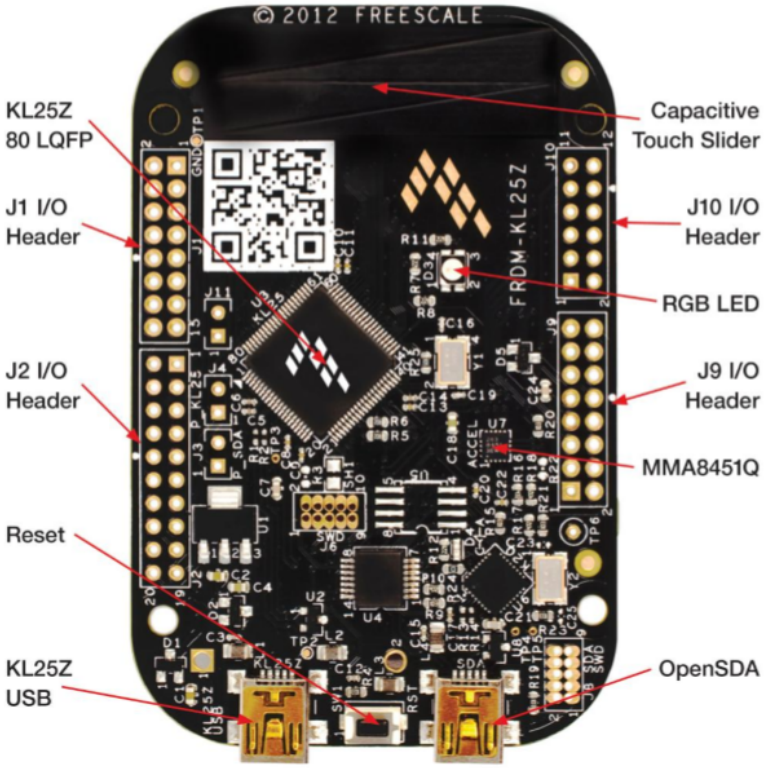


Figure 4 Important pins and their placement on the board.

2.1.2 Atmega 328p (Arduino UNO):

Arduino is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform Microcontroller ATmega328.

The ATmega328 is a single-chip microcontroller created by Atmel in the megaAVR family.

Specifications:

- Operating Voltage 5V
- Supply Voltage (recommended) 7-12V
- Maximum supply voltage (not recommended) 20V
- Digital I/O Pins 14
- Analog Input Pins 6
- DC Current per I/O Pin 40 mA
- DC Current for 3.3V Pin 50 mA
- Flash Memory 32KB
- Clock Speed 16 MHz

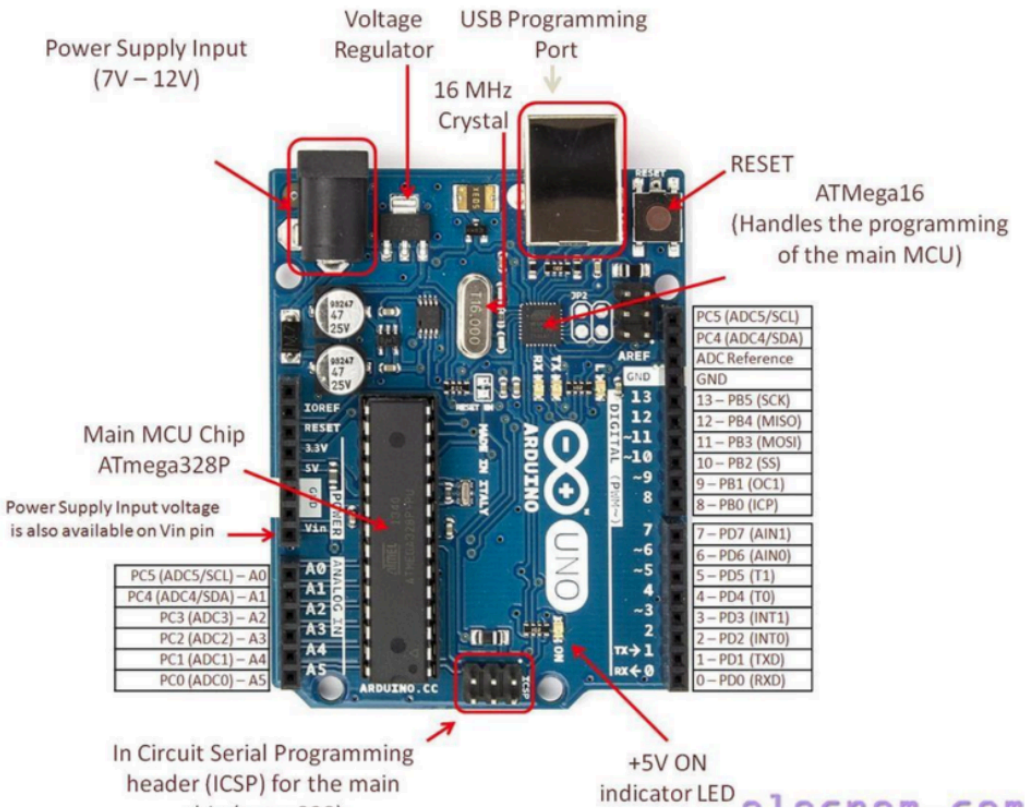


Figure 4 Arduino UNO Board.

2.1.3 NRF Module

The nRF24L01+(nRF24L01p) is a single chip 2.4GHz transceiver with an embedded baseband protocol engine (Enhanced ShockBurst™), suitable for ultra low power wireless applications. The nRF24L01+ is designed for operation in the world wide ISM frequency band at 2.400 - 2.4835GHz. To design a radio system with the nRF24L01+, you simply need an MCU (microcontroller) and a few exter-nal passive components. The high air data rate combined with two power saving modes make the nRF24L01+ very suitable for ultra low power designs. nRF24L01+ is drop-in compatible with nRF24L01 and on-air compatible with nRF2401A, nRF2402, nRF24E1 and nRF24E2. Intermodulation and wideband blocking values in nRF24L01+ are much improved in comparison to the nRF24L01 and the addition of internal filtering to nRF24L01+ has improved the margins for meeting RF regulatory standards.

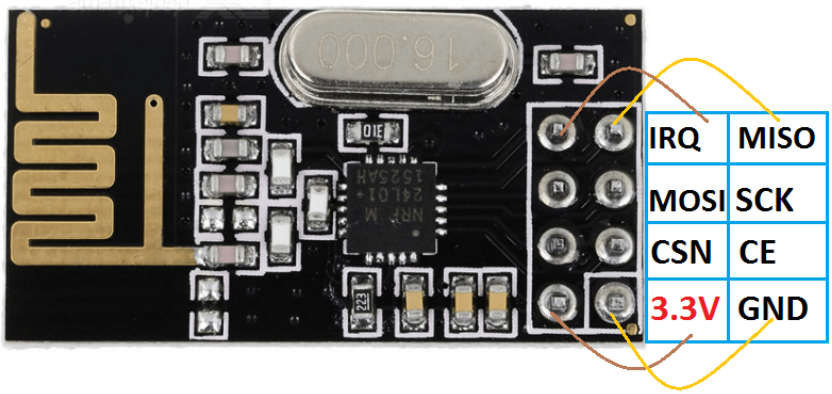


Figure 5 NRF Module.

Features

- Worldwide 2.4GHz ISM band operation, Free license to use.
- 126 RF channels.
- High air data rate: 250kbps, 1 and 2Mbps.
- Transmitter: 11.3mA at 0dBm output power.
- Receiver: Fast AGC for improved dynamic range.
- Receiver: Integrated channel filters.
- Enhanced ShockBurst™:1 to 32 bytes dynamic payload length,6 data pipe MultiCeiver™ for 1:6 star networks.
- Host Interface:4-pin hardware SPI,3 separate 32 bytes TX and RX FIFOs.
- Low Power Management:1.9 to 3.6V supply range.
- GFSK modulation.
- Auto packet transaction handling.
- Easy for designed.
- Small size:15mm*29mm.

2.1.4 Wi-Fi Module

ESP8266 series wireless module is a series of cost-effective Wi-Fi SOC module which can be developed independently. The series modules support the standard IEEE802.11 b/g/n protocol, built-in complete TCP/IP protocol stack. Users can use this series of modules to add networking capabilities to existing devices, or to build standalone network controllers.

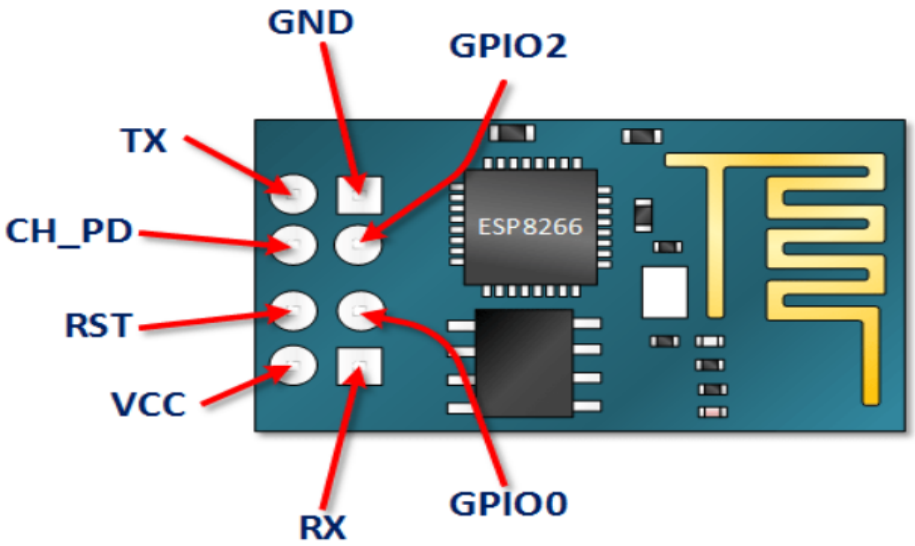


Figure 6 Wi-Fi Module Pin Diagram.

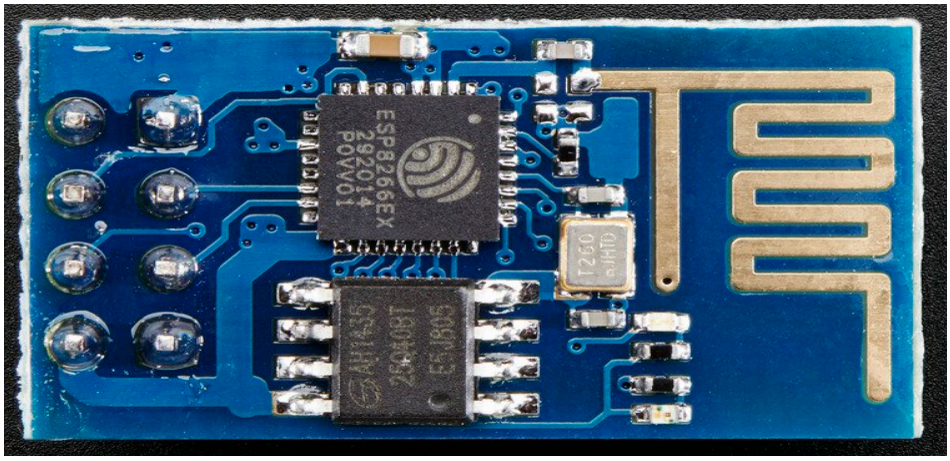


Figure 7 Wi-Fi Module.

Features

- The smallest 802.11b/g/n Wi-Fi SOC module
- Using low-power 32-bit CPU, can also serve as the application processor
- Clocked at up to 160MHz z Built-in 10 bit high precision ADC
- Support UART/GPIO/IIC/PWM/ADC/HSPI and other interfaces
- Integrated Wi-Fi MAC/BB/RF/PA/LNA
- Supports multiple sleep modes, deep sleep current as low as 20uA
- Embedded Lwip protocol stack
- Support STA/AP/STA + AP work mode z Supports Smart Config/AirKiss key distribution network
- Serial port rate up to 4Mbps
- General AT commands can be used quickly

- Support SDK secondary development
- Supports serial local upgrade and remote firmware upgrade (FOTA).

2.1.5 GSM Module

GSM (Global System for Mobile Communications, originally Groupe Special Mobile), is a standard developed by theEuropean Telecommunications Standards Institute(ETSI) to describe the protocols for second-generation (2G) digital cellular networks used by mobile phones, first deployed in Finland in July 1991.[2] As of 2014 it has become the default global standard for mobile communications - with over 90% market share, operating in over 219 countries and territories.

2G networks developed as a replacement for first generation (1G) analogy cellular networks, and the GSM standard originally described a digital, circuit-switched network optimized for full duplex voice telephony. This expanded over time to include data communications, first by circuit-switched transport, then by packet data transport via GPRS (General Packet Radio Services) and EDGE (Enhanced Data rates for GSM Evolution or EGPRS).

Subsequently, the 3GPP developed third-generation (3G) UMTS standards followed by fourth-generation (4G) LTE Advanced standards, which do not form part of the ETSI GSM standard.

"GSM" is a trademark owned by the GSM Association. It may also refer to the (initially) most common voice codec used, Full Rate.

A GSM modem is a wireless modem that works with a GSM wireless network. A wireless modem is like a dial-up modem. The basic difference between them is the dial-up modem sends and receives data through a fixed telephone line while the wireless modem sends and receives data through waves. Like a GSM mobile phone, a GSM modem also requires a SIM card from a wireless carrier to operate.

AT commands are used to control MODEMs. AT is the abbreviation for Attention. These commands come from Hayes commands that were used by the Hayes smart modems. The Hayes commands started with AT to indicate the attention from the MODEM. The dial up and wireless MODEMs (devices that involve machine to machine communication) need AT commands to interact with a computer. These include the Hayes command set as a subset, along with other extended AT commands.

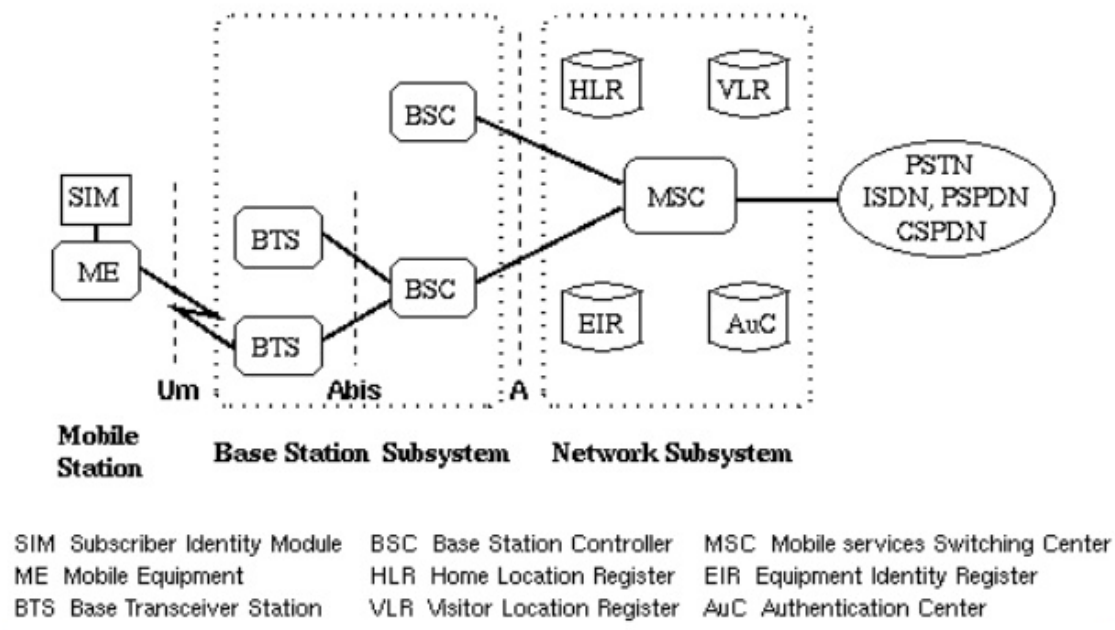


Figure 8 GSM Architecture (Courtesy: Google).

AT commands with a GSM/GPRS MODEM or mobile phone can be used to access following information and services:

1. Information and configuration pertaining to mobile device or MODEM and SIM card.
2. SMS services.
3. MMS services.
4. Fax services.

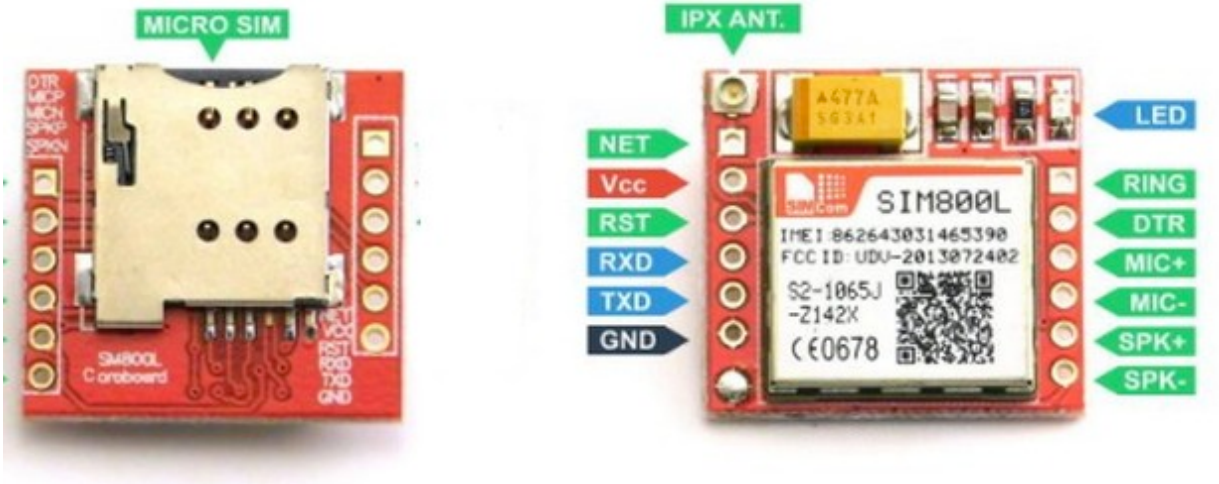


Figure 9 GSM Module Pin Diagram.

GSM/GPRS module is used to establish communication between a computer and a GSM-GPRS system. Global System for Mobile communication (GSM) is an architecture used for mobile communication in most of the countries. Global Packet Radio Service (GPRS) is an extension of GSM that enables higher data transmission rate. GSM/GPRS module consists of a

GSM/GPRS modem assembled together with power supply circuit and communication interfaces (like RS-232, USB, etc) for computer. GSM/GPRS MODEM is a class of wireless MODEM devices that are designed for communication of a computer with the GSM and GPRS network. It requires aSIM (Subscriber Identity Module) card just like mobile phones to activate communication with the network. Also they have IMEI (International Mobile Equipment Identity) number similar to mobile phones for their identification.

A GSM/GPRS MODEM can perform the following operations:

1. Receive, send or delete SMS messages in a SIM.
2. Read, add, search phonebook entries of the SIM.
3. Make, Receive, or reject a voice call.

2.1.6 Temperature Sensor

DHT 11 Sensors DHT11 digital temperature and humidity sensor is a composite Sensor contains a calibrated digital signal output of the temperature and humidity. It ensures high reliability and excellent long-term stability. This sensor is resistive-type humidity and an NTC temperature measurement component and connects to a high performance 8-bit microcontroller, offering excellent quality, fast response, anti-interference ability and cost effectiveness.

It has a small size, low power consumption and signal transmission up-to-20. DHT11 is a 3-pin single row pin package. Its features are as follows.

It works on 5V.
Temperature range: 0 - +50 °C.
Temperature accuracy: ± 2.0 °C.
Humidity range: 20-95% RH.
Humidity accuracy: ± 5.0%
RH Response time: < 5 Sec

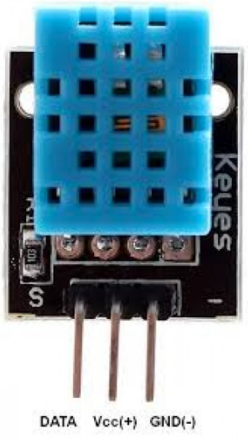
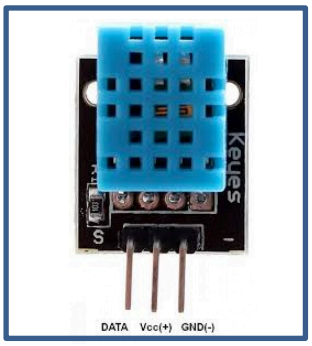
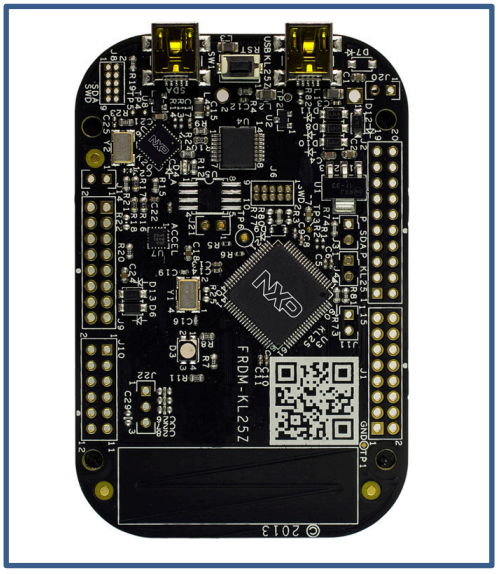


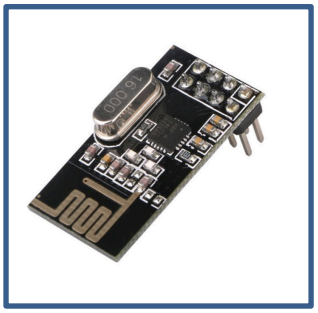
Figure 10 DHT11 Temperature Sensor

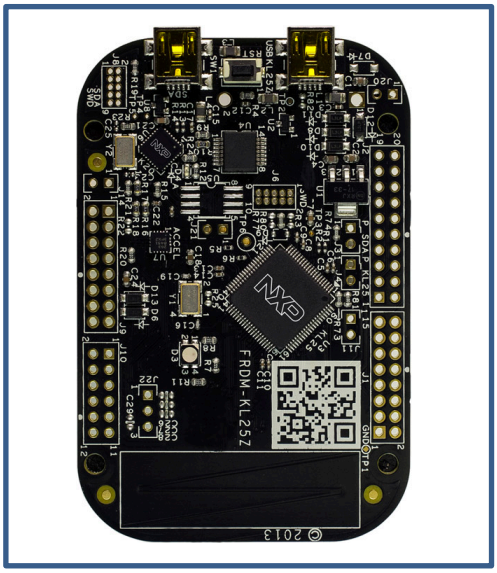
Chapter 3

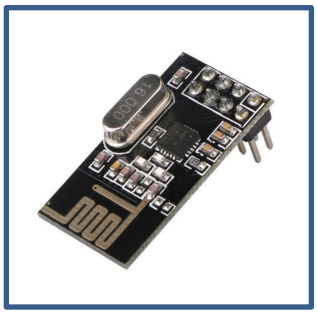
Block Diagram

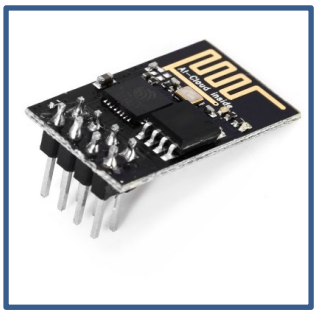


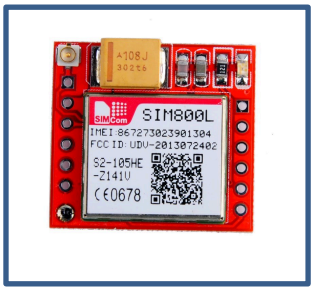












Chapter 4

Working

4.1 Transmitter Board

The FRDM KL25z are use on both sides of the wireless communication. The Transmitter side includes an NRF module and Temperature sensor. NRF Module sends the values received from the temperature module DHT11 through the FRDM board. The Serial Monitor Reflects the real time temperature and humidity values.

4.2 Receiver Board

The Transmitter side includes an NRF module and Wi-Fi module. The temperature values received from the NRF module are sent to the IoT cloud through the ESP8266 Wi-Fi module which is connected to the FRDM board. The Wi-Fi module is connect to the internet through the AT commands in the Embedded C code. Once the connection is established, the current temperature values received are sent to the IoT cloud and the updated values get reflected on the graph.

4.3 Alerting Board

When the temperature rises above the normal standards, a signal will be sent to the Arduino UNO to notify the residents and neighbors about a possibility of FIRE and inform to take preventive measures through the GSM module SIM800l. This signal will be turned on when the if condition if satisfied and it goes to the Arduino UNO. The UNO runs the code and sends SMS and calls to notify.

4.4 Software Specification

```
/****PIPE0
Transmitter****/


                    #include "mbed.h"
                    #include "nRF24L01P.h"
                    #include "Dht11.h"

                    Serial pc(USBTX, USBRX); // tx, rx
                     Dht11 sensor(PTD7);

                    nRF24L01P my_nrf24l01p(PTD2, PTD3, PTD1, PTD0, PTD5, PTD4);    //
                    mosi, miso, sck, csn, ce, irq
                    DigitalOut RedLED(LED1);
```

```
int main()
{
    char count[1];
    char TxDataCnt;
    char temp;

    //set tx and rx address for pipe0: 5 bytes long; different for
every pipe
    long long TxAddress_PIPE0 = 0xC2C2C2C2C2;
    long long RxAddress_PIPE0 = 0xC2C2C2C2C2;

    my_nrf24l01p.powerUp();
    my_nrf24l01p.setRfFrequency(2492);

    //setting tx and rx address
    my_nrf24l01p.setTxAddress(TxAddress_PIPE0);

    //set rx address with default width and pipe number
    my_nrf24l01p.setRxAddress(RxAddress_PIPE0,
DEFAULT_NRF24L01P_ADDRESS_WIDTH, NRF24L01P_PIPE_P0);

    // Display the (default) setup of the nRF24L01+ chip
    pc.printf( "nRF24L01+ Frequency    : %d MHz\r\n",
my_nrf24l01p.getRfFrequency() );
    pc.printf( "nRF24L01+ Output power : %d dBm\r\n",
my_nrf24l01p.getRfOutputPower() );
    pc.printf( "nRF24L01+ Data Rate    : %d kbps\r\n",
my_nrf24l01p.getAirDataRate() );
    pc.printf( "nRF24L01+ TX Address - PIPE0  : 0x%010llX\r\n",
my_nrf24l01p.getTxAddress() );

    pc.printf( "Wirelesss sensor network \r\n");

    TxDataCnt = 1;

    //set transfer size for specified pipe
    my_nrf24l01p.setTransferSize(TxDataCnt, NRF24L01P_PIPE_P0);
    my_nrf24l01p.enable();

    char pot_val;

    while (1) {

        //adjusting data to 0-255 with char data type
        sensor.read();
        pc.printf("Current Temperature is %f degrees in fahrenheit
and Humidity is %d%\r\n", sensor.getFahrenheit(),
sensor.getHumidity());
        pot_val = sensor.getFahrenheit();
```

```
            count[0] = pot_val;

            //transmit on specified pipe
            temp = my_nrf24l01p.write( NRF24L01P_PIPE_P0,count, TxDataCnt
);

            pc.printf( "Sending %d bytes from Transmitter FRDM Board to
Receiver FRDM Board through PIPE0; Temperature =
%d\r\n",temp,count[0]);

            // Toggle LED1 (to help debug Host -> nRF24L01+
communication)
            RedLED = !RedLED;

            wait(2);
        }
    }
```

```
/*****Multiceiver
wireless
network*****/
```

```
#include "mbed.h"
#include "ESP8266.h"
#include "nRF24L01P.h"

Serial pc(USBTX, USBRX); // tx, rx

nRF24L01P my_nrf24l01p(PTD2, PTD3, PTD1, PTD0, PTD5, PTD4);
// mosi, miso, sck, csn, ce, irq
ESP8266 wifi(PTE0, PTE1, 115200);


char snd[255],resp[1000];
char http_cmd[300], comm[300];
int timeout = 5000;
DigitalOut myled(PTA5);


#define SSID "yash"
#define PASS "12345678"


//Remote IP
#define IP "34.231.200.253"


//ldrvalue global variable
```

```
int pot1Val = 0;



//Public and private keys for phant
char* Update_Key = "T2UOHRRW96LYCTE8";
//blue for pipe1 and green for pipe0
DigitalOut GreenLED(LED2);
DigitalOut RedLED(LED1);


//Wifi init function
void wifi_initialize(void){

    pc.printf("\n******** Resetting wifi module ********\r\n");
    wifi.Reset();

    //wait for 5 seconds for response, else display no response
receiveed
    if (wifi.RcvReply(resp, 5000))
        pc.printf("%s",resp);
    else
        pc.printf("No response");

    pc.printf("******** Setting Station mode of wifi with AP
********\r\n");
    wifi.SetMode(1);    // set transparent  mode
    if (wifi.RcvReply(resp, timeout))    //receive a response
from ESP
        pc.printf("%s",resp);    //Print the response onscreen
    else
        pc.printf("No response while setting mode. \r\n");

    pc.printf("******** Joining network with SSID and PASS
********\r\n");
    wifi.Join(SSID, PASS);
    if (wifi.RcvReply(resp, timeout))
        pc.printf("%s",resp);
    else
        pc.printf("No response while connecting to network
\r\n");

    pc.printf("******** Getting IP and MAC of module
********\r\n");
    wifi.GetIP(resp);
    if (wifi.RcvReply(resp, timeout))
        pc.printf("%s",resp);
    else
        pc.printf("No response while getting IP \r\n");
```

```
    pc.printf("******** Setting WIFI UART passthrough
********\r\n");
    wifi.setTransparent();
    if (wifi.RcvReply(resp, timeout))
        pc.printf("%s",resp);
    else
        pc.printf("No response while setting wifi passthrough.
\r\n");
    wait(1);

    pc.printf("******** Setting single connection mode
********\r\n");
    wifi.SetSingle();
    wifi.RcvReply(resp, timeout);
    if (wifi.RcvReply(resp, timeout))
        pc.printf("%s",resp);
    else
        pc.printf("No response while setting single connection
\r\n");
    wait(1);
}


void wifi_send(void){

    pc.printf("******** Starting TCP connection on IP and port
********\r\n");
    wifi.startTCPConn(IP, 80);    //cipstart
    wifi.RcvReply(resp, timeout);
    if (wifi.RcvReply(resp, timeout))
        pc.printf("%s",resp);
    else
        pc.printf("No response while starting TCP connection
\r\n");
    wait(1);

    //create link

sprintf(http_cmd,"/update?key=%s&field1=%d",Update_Key,pot1Val);
    pc.printf(http_cmd);

    pc.printf("******** Sending URL to wifi ********\r\n");
    wifi.sendURL(http_cmd, comm);   //cipsend and get command
    if (wifi.RcvReply(resp, timeout))
        pc.printf("%s",resp);
    else
        pc.printf("No response while sending URL \r\n");
```

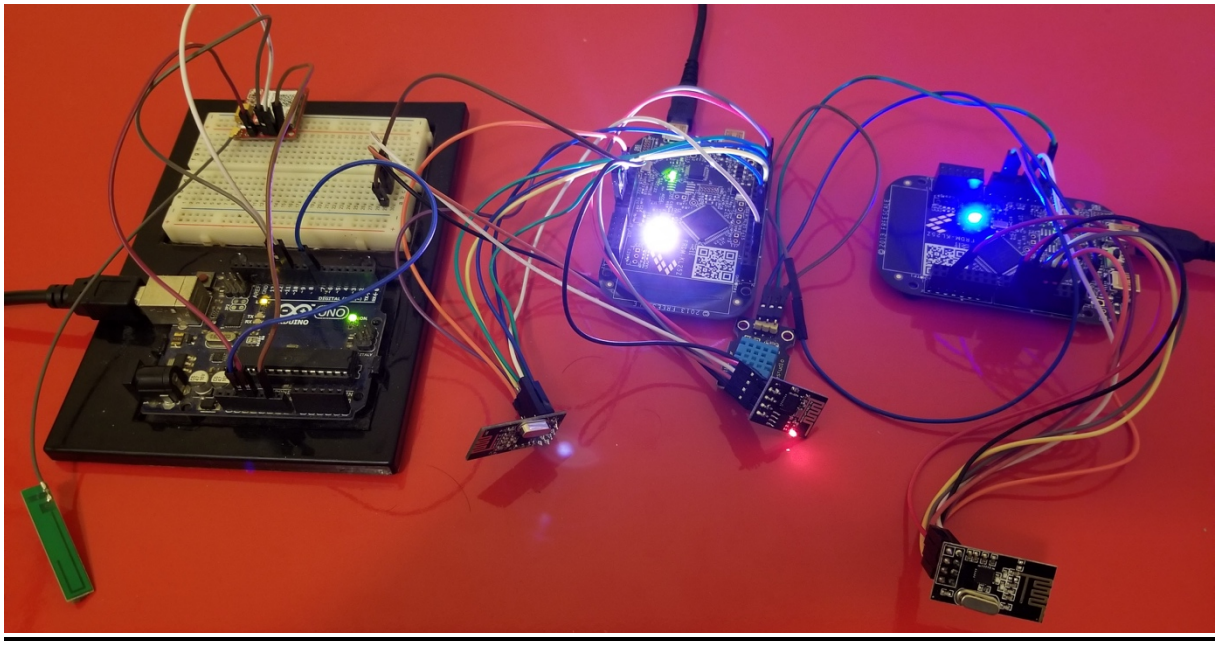
```
    //wifi.SendCMD("AT+CIPCLOSE"); //Close the connection to
server
    //wifi.RcvReply(resp, timeout);
    //pc.printf("%s", resp);
}

int main()
 {



    char count[1];
    char RxDataCnt_PIPE0, RxDataCnt_PIPE1;
    char temp;

    //specifying address same as transmitter for pipe0 and pipe1
    long long RxAddress_PIPE0 = 0xC2C2C2C2C2;

    wifi_initialize();
    my_nrf24l01p.powerUp();
    my_nrf24l01p.setRfFrequency(2492);

    //set rx address with default address and for specified pipe
    my_nrf24l01p.setRxAddress(RxAddress_PIPE0,
DEFAULT_NRF24L01P_ADDRESS_WIDTH, NRF24L01P_PIPE_P0);

    // Display the (default) setup of the nRF24L01+ chip
    pc.printf( "nRF24L01+ Frequency    : %d MHz\r\n",
my_nrf24l01p.getRfFrequency() );
    pc.printf( "nRF24L01+ Output power : %d dBm\r\n",
my_nrf24l01p.getRfOutputPower() );
    pc.printf( "nRF24L01+ Data Rate    : %d kbps\r\n",
my_nrf24l01p.getAirDataRate() );

    //display rx address for both pipes
    pc.printf( "nRF24L01+ RX Address - PIPE0  : 0x%010llX\r\n",
my_nrf24l01p.getRxAddress(NRF24L01P_PIPE_P0) );
    pc.printf( "Wireless Sensor Network - Multiceiver\r\n" );

    RxDataCnt_PIPE0 = 1;

    //set transfer size explicitly for both pipes
    my_nrf24l01p.setTransferSize(RxDataCnt_PIPE0,
NRF24L01P_PIPE_P0);

    my_nrf24l01p.setReceiveMode();
    my_nrf24l01p.enable();

    while (1)
    {
```

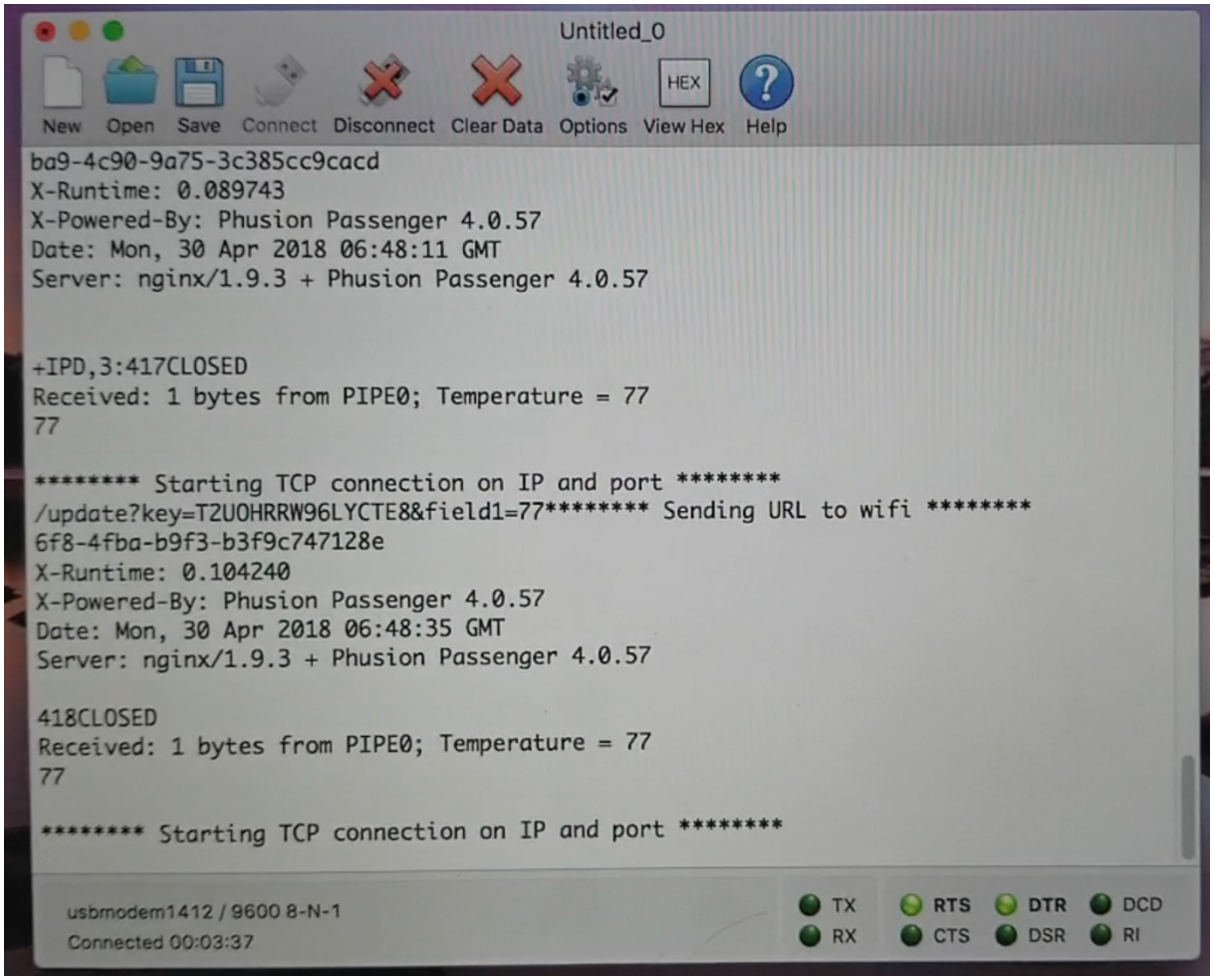
```
                        //check if data is available in pipe0
                        if ( my_nrf24l01p.readable(NRF24L01P_PIPE_P0) ) {

                            // ...read the data into the receive buffer
                            temp = my_nrf24l01p.read( NRF24L01P_PIPE_P0, count,
                    RxDataCnt_PIPE0 );

                            pot1Val = count[0];

                            pc.printf("Received: %d bytes from PIPE0;
                    Temperature = %d\r\n",temp, count[0]);

                            // Toggle LED2 (to help debug nRF24L01+ -> Host
                    communication)
                            GreenLED = !GreenLED;
                        }
                        pc.printf("%d \n\r",pot1Val);

                        wifi_send();
                        if(pot1Val>80) {
                           myled=1;
                        }
                        else {
                            myled=0;
                        }
                        wait(5);
                    }
                    }
/*
ALERTING -
GSM Module
            * RX is digital pin 10 (connect to TX of GSM
            Modem)
            * TX is digital pin 11 (connect to RX of GSM
            Modem) */



            #include <SoftwareSerial.h>


            SoftwareSerial mySerial(10, 11); // RX, TX
            const int pina = 7;
            int val = 0;
```

```
void setup() {

pinMode(pina,INPUT);

// Open serial communications and wait for port
to open:

Serial.begin(9600);

Serial.println("Calling through GSM Modem");

}



void loop() {

val = digitalRead(pina);

if (val == HIGH) {

// set the data rate for the SoftwareSerial port
            mySerial.begin(9600);
            delay(2000);
            mySerial.println("ATD+18628720305;");

                        Serial.println("Calling
8628720305");

// print response over serial port
             if (mySerial.available())
             Serial.write(mySerial.read());
             delay(10000);
  }

}
```

Chapter 5

Results





Chapter 6

Conclusion

The main idea behind implementation of this project was to introduce a system that would monitor and control the mechanism of maintaining the house temperature and alerting the residents in case of calamity. Thus, at any instant of time, if the house temperature rises beyond the set threshold level, the residents will be alerted with call notification on their mobile devices which makes it possible for them to take the necessary measures so as to avoid/reduce loss of property and even life!!

For this prototype, we made use of DHT11 sensor to monitor only the temperature values. We are further working on this prototype to inculcate more sensors and introduce more parameters such as humidity and smoke to further enhance the safety mechanism.

We are also working on upgrading the existing use GSM technology to notify the residents with AWS in order to have continuous real monitoring of temperature values.

Chapter 7

References

1. https://thingspeak.com/
2. https://os.mbed.com/
3. http://freeware.the-meiers.org/
4. http://freeware.the-meiers.org/
5. https://mcuoneclipse.com/2013/07/20/tutorial-ultra-low-cost-2-4-ghz-wireless-transceiver-with-the-frdm-board/
6. https://www.teachmemicro.com/arduino-gsm800l-tutorial/