# Project Report SIT-2018

Name : Niti Patel  Group no. : 64  Roll no.: 12

KJ SOMAIYA COLLEGE OF ENGINEERING

# WIRELESS DATA ACQISATION SYSTEM

- ## Introduction/Project Description:

In modern wireless communication, GSM, CDMA, 3G, and Wi-Fi become the mainstream solution of wireless data transmission network because of their high speed and reliable quality. They also have the shortcomings of high cost, so wider application would cause a great waste of resources, and they cannot be promoted in small regional, low speed data communications.

The system we have created supports point-point and point to multipoint communication. The advantage of designing a short-range wireless communication network is to provide short-distance broadband wireless access to mobile environment or formulation of temporary network, it is the further development of internet in mobile environment. The main advantage of short-range wireless communication network is lower cost and more flexible use.

In our system we have used the FRDM KL25Z board, nRF24L01+P transceiver modules, ESP8266 WIFI modules and a shield that includes sensors, LEDs etc. We have developed a system to upload and download data from a could service called thingspeak based on the input received by the user on the serial terminal.

This system also includes the use all the protocols of communication namely UART, I2C  and SPI for different aspects in the system. However the system operates in such a way that we can download or upload only one data at a given time.

- APIs used:

- The following header files are included in the transmitter and receiver codes :

| TRANSMITTER | RECEIVER |
|---|---|
| #include "mbed.h"(Includes APIs related to FRDM) | #include "mbed.h"(Includes APIs related to FRDM) |
| #include "nRF24L01P.h" (Includes APIs related to NRF2401+P module) | #include "nRF24L01P.h"(Includes APIs related to NRF2401+P module) |
| #include "ESP8266.h"(Includes APIs related to ESP8266 wifi module) | #include "ESP8266.h"(Includes APIs related to ESP8266 wifi module) |
| | #include "TSISensor.h"(Includes APIs related to TSI touch sensor) |
| | #include"MMA8451Q.h"(Includes APIs related to Accelerometer) |

- The following are the APIs used in the Transmitter code:

| *TRANSMITTER* | *CLASS VARIABLE PIN DECLARATION* |
|---|---|
| *DigitalOut GreenLED(PTA12);* | Class: DigitalOut(Sets the mentioned pin as digital output)<br>Variable: GreenLED<br>Pin: PTA12 (Now set as digital output ) |
| *Serial pc(USBTX, USBRX);* | Class: Serial (Sets pins as TX and RX for Serial Communication using UART)<br>Variable: pc<br>Pins: USBTX(Transmitter), USBRX(Receiver) |
| *nRF24L01P my_nrf24l01p(PTD2, PTD3, PTD1, PTD0, PTD5, PTD4);* | Class: nRF24L01P (Sets pins to states required for SPI Communication)<br>Variable: my_nrf24l01p<br>Pins: PTD2(Master Out Slave In), PTD3(Master In Slave Out), PTD1(Serial Clock), PTD0(Chip Select Not), PTD5(Chip Enable), PTD4 (IRQ Interrupt) |
| *ESP8266 wifi(PTE0, PTE1, 115200);* | Class: ESP8266(Sets pins as TX and RX for communication through the wifi module also sets baud rate as 115200)<br>Variable: wifi<br>Pin: PTE0(Transmitter), PTE1(Receiver) |

- The following are the APIs used in the Receiver code:

| *RECEIVER* | *CLASS VARIABLE PIN DECLARATION* |
|---|---|
| *DigitalOut GreenLED(PTA12);* | Class: DigitalOut(Sets the mentioned pin as digital output)<br>Variable: GreenLED<br>Pin: PTA12 (Now set as digital output ) |
| *Serial pc(USBTX, USBRX);* | Class: Serial (Sets pins as TX and RX for Serial Communication using UART)<br>Variable: pc<br>Pins: USBTX(Transmitter), USBRX(Receiver) |
| *nRF24L01P my_nrf24l01p(PTD2, PTD3, PTD1, PTD0, PTD5, PTD4);* | Class: nRF24L01P (Sets pins to states required for SPI Communication)<br>Variable: my_nrf24l01p<br>Pins: PTD2(Master Out Slave In), PTD3(Master In Slave Out), PTD1(Serial Clock), PTD0(Chip Select Not), PTD5(Chip Enable), PTD4 (IRQ Interrupt) |
| *ESP8266 wifi(PTE0, PTE1, 115200);* | Class: ESP8266(Sets pins as TX and RX for communication through the wifi module also sets baud rate as 115200)<br>Variable: wifi<br>Pin: PTE0(Transmitter), PTE1(Receiver) |
| *AnalogIn pot(A0);*<br>*AnalogIn ldr(A1);* | Class: AnalogIn (Sets the mentioned pins as analog input)<br>Variable: pot(To input pot value in this case)<br>ldr(To input ldr value in this case)<br>Pins: A0(For POT), A1(For LDR). |
| *MMA8451Q acc(PTE25,PTE24, (0x1d<<1));* | Class: MMA8451Q (Sets Pins to enable I2C Communication between accelerometer and the board)<br>Variable: acc<br>Pins: PTE25(Serial Clock),PTE24(Serial Data Line),(0x1d<<1)(Slave Address for MMA8451Q) |
| *TSISensor tsi;* | Class: TSISensor(Sets the tsi pin for sensing the touch on the touch sensor)<br>Variable: tsi |

- Algorithm

## TRANSMITTER:

1. Start
2. Declaration of libraries, APIs, buffers, SSID and password, IP address, variables etc.
3. Initialize nRF24L01+ module:
   - Power Up the module.
   - Set RF Frequency.
   - Display nRF24L01+ default setup.
   - Set Transmitter Transfer-size.
   - Enable the module.
4. Initialize ESP8266-WIFI module:
   - Reset the module.
   - Check and display for remote response, wait for 5 seconds and print no response if no response is received.
   - Set Station mode to 1 and check again for response, print no response while setting station mode if no response is received.
   - Join the network using SSID and password and check for response again, print no response while connecting to network if no response is received.
   - Get the IP and MAC address of the module and check again for response, print no response while getting the IP if no response is received.
   - Set WIFI UART passthrough to transparent to print verbose response, and check again for response, print no response while setting WIFI passthrough if no response is received.
   - Set Single Connection mode i.e. connect only with one receiver no hoping and check again for response, print no response while setting single connection mode if no response is received.
5. Enter an infinite loop and get the user input character on serial terminal software and check the character received.
6. If character received is any from P, p, L, l, T, t, X, x, Y, y, Z or z then go to (a) else go to (b):
   a) Sending character to Receiver for file Upload:
      - Send the Transmit buffer via the nRF24L01+.
      - Toggle LED1 (to help debug Host: nRF24L01+ communication).

   b) Send Character to WIFI send function for Downloading:
      - Set i equal to the character received if character received is between 1 to 6 else display INVALID INPUT.
      - Call the WIFI send function and break the loop:
        I. Start TCP connection on IP and port i.e. cip-start ( connection with server/ip /thingspeak) 80:http and check for response.
        II. Create the link to be sent using the 'GET' method and 'i' i.e. the character received.
        III. Send the URL to WIFI to download the respective value based on character received and check for response again, print no response while sending URL if no response is received.

## RECEIVER:

1. Start
2. Declaration of libraries, APIs, buffers, SSID and password, IP address, variables, Update key etc.
3. Initialize nRF24L01+ module:
    - Power Up the module.
    - Set RF Frequency.
    - Display nRF24L01+ default setup.
    - Set Receiver Transfer-size.
    - Set module to receive mode, since acts as a transmitter by default.
    - Enable the module.
4. Initialize ESP8266-WIFI module:
    - Reset the module.
    - Check and display for remote response, wait for 5 seconds and print no response if no response is received.
    - Set Station mode to 1 and check again for response, print no response while setting station mode if no response is received.
    - Join the network using SSID and password and check for response again, print no response while connecting to network if no response is received.
    - Get the IP and MAC address of the module and check again for response, print no response while getting the IP if no response is received.
    - Set WIFI UART passthrough to transparent to print verbose response, and check again for response, print no response while setting WIFI passthrough if no response is received.
    - Set Single Connection mode i.e. connect only with one receiver no hoping and check again for response, print no response while setting single connection mode if no response is received.
7. Enter an infinite loop and get check if we've received anything in the nRF24L01+.
8. Read the data into the receive buffer
9. If character received is any from P, p, L, l, T, t, X, x, Y, y, Z or z then read sensor value based on the received character and the call the WIFI send function.
    a) Send Character to WIFI send function for Downloading:
        - Call WIFI send function and break the loop:
            I. Start TCP connection on IP and port i.e. cip-start ( connection with server/ip /thingspeak) 80:http and check for response.
            II. Create the link to be sent using the sensor value that we read i.e. based on the character received.
            III. Send the URL to WIFI to Upload data to that respective sensor and check for response again, print no response while sending URL if no response is received.

- Errors observed:
    1. "busy" means the previous AT command is not finished yet, so we cannot accept the next AT command .
    2. "busy p" means command is doing; "busy s" means sending data is doing.
    3. "ERROR" means you may input an incorrect AT command.

- Learning outcome:.

We learnt to use various APIs in-order to control multiple modules, we also learnt various communications protocols namely UART, I2C  and SPI for different aspects in the system. The system easily downloaded or uploaded any one data at a given time based on the user input.

It also helped us to learn and get hands-on experience with embedded systems, IOT and cloud services, and create an efficient system for short range communication.

- Output observed:

## 1. Transmitter Part:

There will be two set of outputs for the Tx module.

1. To send character to Rx to Upload on thingspeak.

   When any of the values from below was entered on the terminal on tx end:

   P - Pot
   L - Ldr
   X - x-axis
   Y - y-axis
   Z - z-axis
   T - TSI values

The RX received the character, and read the data and uploaded the respective data on thingspeak.

2. To download data from thingspeak

   When any of the values from below was entered on the terminal on tx end:

   1 - Pot
   2 - LDR
   3 - x-axis
   4 - y-axis
   5 - z-axis
   6 - tsi values

Then the respective data was downloaded from thingspeak and displayed it on Terminal.

## 2. Receiver Part:

Rx received upload command from Tx. Upon receiving command, Rx module read that respective sensor data and uploaded it on thingspeak. Also the previous values of sensors whose data is not requested should was maintained and uploaded every time.

- Program code:

# Receiver Code:

```
#include "mbed.h"
#include "nRF24L01P.h" //Receiver
#include"MMA8451Q.h"
#include "TSISensor.h"
#include "ESP8266.h"
TSISensor tsi;
MMA8451Q acc(PTE25,PTE24,(0x1d<<1));
AnalogIn pot(A0);
AnalogIn ldr(A1);
Serial pc(USBTX, USBRX); // tx, rx
nRF24L01P my_nrf24l01p(PTD2, PTD3, PTD1, PTD0, PTD5, PTD4);    // mosi, miso, sck, csn, ce, irq
DigitalOut GreenLED(PTA12);
ESP8266 wifi(PTE0, PTE1, 115200);
//buffers for wifi library
char resp[1000];
char http_cmd[300], comm[300];
int timeout = 8000; //timeout for wifi commands
#define SSID "iPhone"
#define PASS "12345678Np"
#define IP "184.106.153.149" // "api.thingspeak.com"
float p=0,t=0,l=0,x=0,y=0,z=0;
  char count[1];
char* Update_Key = "R7O654V084X5Q1XA"; //write key if you are uploading
//Wifi init function(USER DEFINED BOTH INIT AND SEND)
void wifi_initialize(void){
   pc.printf("******** 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);    //  0 ,1 OR 2 (1 station/ 0 access point / 2 switching between the two)  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(); // to print verbose respose meaning detailed version of response
  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();     // Single means only connect to one acess point(no hopping)
  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(){

    pc.printf("******** Starting TCP connection on IP and port ********\r\n");
    wifi.startTCPConn(IP,80);    //cipstart (connection with server/ip/thingspeak) 80:http
    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?api_key=%s&field1=%f&field2=%f&field3=%f&field4=%f&field5=%f&field6=%f",Update_Key,p,l,t,x,y,z);
    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");
}
int main() {
      wifi_initialize();
    char RxDataCnt;
    char temp;
      char c;
```

```
my_nrf24l01p.powerUp();
my_nrf24l01p.setRfFrequency(2470);

// 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   : 0x%010llX\r\n", my_nrf24l01p.getTxAddress() );
pc.printf( "nRF24L01+ RX Address   : 0x%010llX\r\n", my_nrf24l01p.getRxAddress() );

pc.printf( "Simple 1 Byte Receiver\r\n" );
RxDataCnt = 1;
my_nrf24l01p.setTransferSize( RxDataCnt );
my_nrf24l01p.setReceiveMode();
my_nrf24l01p.enable();
while (1) {

   // If we've received anything in the nRF24L01+...
   if ( my_nrf24l01p.readable() )
 {    // ...read the data into the receive buffer
      temp = my_nrf24l01p.read( NRF24L01P_PIPE_P0,count, RxDataCnt );
      pc.printf( "%d Value of Chatacter Received is %c \r\n",temp,(count[0]));
     c=count[0];
      // Toggle LED2 (to help debug nRF24L01+ -> Host communication)
      GreenLED = !GreenLED;
      wait_ms(10);
      if(c == 'P' || c == 'p')
      {
         p=pot.read();
         wifi_send();
          wait(20);
      }
      else if(c == 'L' || c == 'l')
      {
         l=ldr.read();
         wifi_send();
```

```
            wait(20);
          }
          else if(c == 'T' || c == 't')
      {    t=tsi.readPercentage();
            wifi_send();
            wait(20);
           }
          else if(c == 'X'||c=='x')
          {
            x=acc.getAccX();
            wifi_send();
            wait(20);
          }
          else if(c == 'Y' || c == 'y')
          {
            y=acc.getAccY();
            wifi_send();
            wait(20);
          }
          else if(c == 'Z' || c == 'z')
          {
            z=acc.getAccZ();
           wifi_send();
            wait(20);
          }
       }
    }
}
```

# Transmitter Code:

```
#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
DigitalOut GreenLED(PTA12);

//wifi UART port and baud rate
ESP8266 wifi(PTE0, PTE1, 115200); //TX,RX,BaudRate

//buffers for wifi library
char resp[1000];
char http_cmd[300], comm[300];

int timeout = 8000; //timeout for wifi commands

//SSID and password for connection
#define SSID "iPhone"
#define PASS "12345678Np"
//Remote IP
#define IP "184.106.153.149" // "api.thingspeak.com"
int i;

//Wifi init function(USER DEFINED BOTH INIT AND SEND)
void wifi_initialize(void){

    pc.printf("******** 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);    //  0 ,1 OR 2 (1 station/ 0 access point / 2 switching between the two)  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(); // to print verbose respose meaning detailed version of response
 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();     // Single means only connect to one acess point(no hopping)
    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(){

    pc.printf("******** Starting TCP connection on IP and port ********\r\n");
    wifi.startTCPConn(IP,80);    //cipstart (connection with server/ip/thingspeak) 80:http
    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,"/channels/531282/fields/%d.json?api_key=YZUW176NQ67RZY3E&results=1",i);
    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);    //add text here
    else
        pc.printf("No response while sending URL \r\n");
}


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

```
   my_nrf24l01p.powerUp();
   my_nrf24l01p.setRfFrequency(2470);

   // 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   : 0x%010llX\r\n", my_nrf24l01p.getTxAddress() );
   pc.printf( "nRF24L01+ RX Address   : 0x%010llX\r\n", my_nrf24l01p.getRxAddress() );

   pc.printf( "Simple Transmitter (0 - 9 Counter) \r\n");

   TxDataCnt = 1;
   my_nrf24l01p.setTransferSize(TxDataCnt);
   my_nrf24l01p.enable();
   wifi_initialize();
        pc.printf("\r\n Upload any character from:\r\n P or p(For Pot Value)\r\n L or l(For Ldr Value)\r\n T or t(For TSI Value)\r\n X or x(For X axis Value)\r\n Y or y(For Y axis Value)\r\n Z or z(For Z axis Value)\r\n");
      pc.printf("\r\n Download any value between 1 to 6:\r\n 1.For Pot Value\r\n 2.For Ldr Value\r\n 3.For TSI Value\r\n 4.For X axis Value\r\n 5.For Y axis Value\r\n 6.For Z axis Value\r\n");

   while (1)
   {
         if(pc.readable())
     {
        count[0]=pc.getc();



     if( count[0] == 'P' || count[0] == 'p' || count[0] == 'L' || count[0] == 'l' || count[0] == 'T' || count[0] == 't' || count[0] == 'X' || count[0] == 'x' ||count[0] == 'Y' ||count[0] == 'y' ||count[0] == 'Z' ||count[0] == 'z')
     {
        // Send the Transmit buffer via the nRF24L01+
        temp = my_nrf24l01p.write( NRF24L01P_PIPE_P0,count, TxDataCnt );
        pc.printf( "Sending %d - Input Value: %c \r\n",temp,count[0]);
        // Toggle LED1 (to help debug Host -> nRF24L01+ communication)
```

```
      GreenLED = !GreenLED;
      wait_ms(10);
    }
    else
    {
      switch(count[0])
      {
        case '1':i=1;
              wifi_send();
               wait(10);
               break;
        case '2':i=2;
              wifi_send();
               wait(10);
               break;
        case '3':i=3;
              wifi_send();
               wait(10);
               break;
        case '4':i=4;
              wifi_send();
               wait(10);
               break;
        case '5':i=5;
              wifi_send();
               wait(10);break;
        case '6':i=6;
              wifi_send();
               wait(10);break;
       default: pc.printf("INVALID INPUT");
                 break;
      }

    }
  }
}
```

}