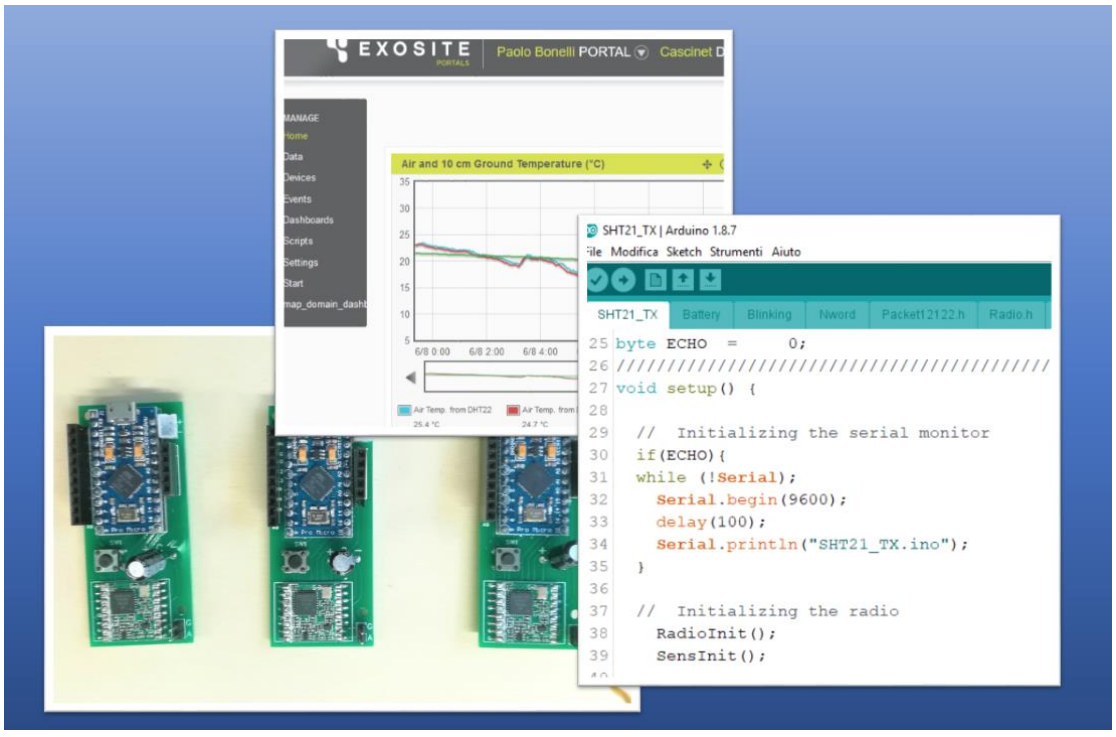
Mini Rete LoRa
il Software

Versione 1.1

Paolo Bonelli

paolobo87@gmail.com

08/12/2019



Sommario

Mini Rete LoRa

Quella che io chiamo “Mini Rete LoRa” è una tecnologia che consente l’acquisizione e trasmissione di grandezze rilevate da sensori, posti in località prevalentemente all’aperto, quindi prive di connettività WiFi, richiedendo poca energia e bassi costi.

Mentre il termine “LoRa” indica una tecnologia di modulazione radio brevettata dalla ditta Semtech, il termine “Mini Rete” è stato coniato da me, per definire un tipo di architettura di dispositivi che sfruttano la tecnologia LoRa per trasmettere dati tra molti trasmettitori e un ricevitore, usata in un contesto di monitoraggio ambientale. La Mini Rete LoRa non va confusa con il protocollo LoRaWAN creato per un’infrastruttura di rete più complessa per applicazioni di vario tipo.

I dati sono trasmessi per mezzo di una frequenza radio libera in tutta Europa (868 MHz) modulata con tecnologia LoRa, che consente lunghe distanze e bassa energia di emissione. La Mini Rete è stata concepita per funzionare senza infrastrutture di terze parti, quindi può essere impiegata ovunque. La Mini Rete prevede una trasmissione di tipo broadcast, dai trasmettitori (TX) al ricevitore (RX) a senso unico. Quest’ultimo può essere connesso ad una rete WiFi esistente sul posto dove è situato ed inviare i dati ad un server Internet.

TX e RX sono costituiti dalla scheda ProMicroLoRa, a cui ho dedicato una pubblicazione a parte. In questo documento si descrive il software di tipo generale necessario a far funzionare le schede ProMicroLoRa. Per seguire le spiegazioni è importante avere qualche conoscenza sulla programmazione di Arduino.

Il pacchetto (payload) inviato dal TX è semplice ed essenziale perché costituito da una testata di 8 byte contenente l’identificativo del TX (`ID_TX`), la lunghezza in byte del pacchetto trasmesso (`Length`), la tensione della batteria in mV (`vBatTX`) e il numero progressivo dei pacchetti (`npacket`), seguono le variabili misurate in qualsiasi formato (`float, int, word, byte`…). La periodicità dei pacchetti è tipicamente di 1 o più minuti.

La trasmissione può essere avviata a tempo o su evento esterno.

La lunghezza massima di un pacchetto è di 50 byte.

Testata

Byte

0	1	2	3	4	5	6	7

ID_TX (int)	Length (int)	vBatTX (int)	Npacket (int)

8	9	10	…	….	…	…	…	…

Variabili misurate in qualsiasi formato (`float, int, word, byte`…)

Il pacchetto viene trasmesso come vettore di `byte` di lunghezza `Length`.

L’associazione delle variabili del sensore ai byte del vettore è fatta tramite le istruzioni `typedef struct` e `typedef union`.

La mini-rete LoRa è costituita da:

- una o più unità trasmittenti, basate da una scheda ProMicroLoRa con alcuni sensori collegati,
- una unità ricevente composta da una ProMicroLora, una scheda ESP8266, per il collegamento ad Internet, e un display. Si può aggiungere un DataLogger Arduino del tipo Adafruit o DeekRobot.

Il Software

Il software per la Mini Rete LoRa, versione senza datalogger, è costituito da tre sketch principali scritti in C costruiti con la IDE di Arduino e alcune librerie open source scaricabili da github.

Lo sketch del TX è adattabile a qualunque sensore. In questo documento si considera il sensore DHT22 che misura la temperatura e l'umidità.

Lo sketch del RX si riferisce alla configurazione con il display OLED a 128 x 32 punti con protocollo I2C

La configurazione presente nel KIT non contempla la trasmissione dei dati ad una scheda ESP8266. Si riporta comunque la documentazione del software compatibile anche con la configurazione completa.

TX: `DHT22_TX.ino`

RX: `ProMicroLora_RX_OLED.ino`

ESP: `ESP_exosite_General_V2.ino` (non compreso nel KIT)

Tutti gli sketch sono scaricabili dal sito:

https://github.com/paolometeo/Outdoor-Sensing

Le librerie

Le librerie extra Arduino, oltre a quelle specifiche per il sensore sono:

per TX

```
#include <RH_RF95.h>
```

fa parte del pacchetto RadioHead e serve a far funzionare il modulo radio

```
#include "Adafruit_SleepyDog.h"
```

contiene la funzione di sleep utilizzata per diminuire i consumi di energia tra una trasmissione e l'altra.

```
#include "DHT.h"
```

Serve a leggere il sensore DHT22 e a ricavare i valori di temperatura e umidità.

per RX

```
#include <RH_RF95.h>
```

fa parte del pacchetto RadioHead e serve a far funzionare il modulo radio

```
#include <SoftwareSerial.h>
```
(solo se presente la scheda ESP8266)

gestisce la comunicazione seriale tra le schede ProMIcroLoRa e ESP8266 (non inclusa nel KIT)

```
#include <Adafruit_SSD1306.h>
```

gestisce il display OLED 128 x 32 via I2C

per ESP8266 (non incluso nel KIT)

```
#include <ESP8266WiFiMulti.h>
```

gestisce il collegamento alla rete WiFi e la ricerca della rete WiFi tra quelle di una lista

```
#include <SoftwareSerial.h>
```

gestisce la comunicazione seriale con la RX

```
#include <Exosite.h>
```

gestisce la costruzione delle stringhe da inviare al server Exosite via Internet

Una trattazione completa sulla libreria RadioHead con molte informazioni sui parametri radio e loro significato si trova qui:

https://www.airspayce.com/mikem/arduino/RadioHead/index.html

Sketch per trasmettitore

A titolo di esempio si riporta lo sketch di un trasmettitore dotato di sensore di temperatura e umidità DHT22:

```
DHT22_TX.ino
```

Lo sketch è strutturato in più schede della IDE. Alcune di esse devono essere modificate se si vuole cambiare sensore o aggiungerne altri, altre rimangono uguali per quasi tutte le applicazioni. La tabella seguente spiega le varie schede

scheda	**Cosa contiene**	**Da modificare quando si cambia sensore**
DHT22_TX	Main program	Si Modificare le chiamate allo sleep. Settare la variabile ECHO. Aggiornare gli #include
Packet30003.h	Dichiarative delle variabili misurate dal sensore e da trasmettere, in questo caso Temperatura e umidità 30003 è l'identificativo del TX	Si Inserire le variabili lette dai vari sensori, che si vuole trasmettere
Radio.h	Dichiarative riguardanti la radio	No

Sens.h	Dichiarative riguardanti il sensore (librerie), numero di cicli di sleep, assegnazione valori a ID_TX e Length	Si Inserire i parametri necessari agli altri eventuali sensori
RadioInit	Funzione di inizializzazione Radio	No
SensInit	Funzione di inizializzazione sensore	Si Inserire le inizializzazioni necessarie agli altri eventuali sensori
ReadSensor	Funzione per la lettura sensore	Si Inserire le istruzioni di lettura degli altri eventuali sensori, oppure creare altre schede
Battery	Funzione per leggere la tensione della batteria	No
Blinking	Funzione lampeggio del LED	No
Nword	Funzione per approssimazione all'intero più vicino	No
Transmit	Funzione per trasmette il pacchetto di dati	No
altDelay	Funzione di ritardo	No

Sens.h

```
1  //  DHT22
2
3  #include "DHT.h"
4  #define PINdata 6        // DHT22 data pin
5  #define DHTTYPE DHT22    // DHT 22  (AM2302), AM2321
6   DHT  dht(PINdata,DHTTYPE) ; // Creats a DHT object
7  //
8  word ncicli =    0;
9  word ncicliMax = 1;
10 word ID_TX = 30003;
11 word Length = 16;  // payload Length
```

Il sensore di temperatura e umidità DHT22 viene collegato tramite tre fili: due per alimentazione a 3.3V e uno per i dati (Pindata); viene letto tramite le funzioni della sua libreria <DHT.h>. Le dichiarative sulle linee 5 e 6 servono a definire il tipo di sensore, visto che la libreria è fatta per diversi sensori, e creano l'oggetto dht.

Le dichiarative sulle linee 8 e 9 riguardano rispettivamente la variabile globale che conta i giri di loop e il suo valore massimo. Quando ncicli raggiunge ncicliMax, viene trasmesso il pacchetto con i dati.

Le dichiarazioni alle righe 10 e 11 non saranno cambiate nel programma, sono rispettivamente l'indicativo numerico del TX e la lunghezza del pacchetto trasmesso.

Packet30003.h

```
//  30003;    DHT22        byte 8 + 8;
typedef struct packet30003{
float      UMID;
float      TEMP;
};

//  these declarations must not be changed
 typedef union pluto {
  packet30003 Data30003;            //  Dataxxxxx is the name of the structure to be joined to
  byte bufVar[MXVAR];               //  array of bytes to be joined with the structure
};
pluto minni;                       //  minni is the name of the union type pluto
```

In questa scheda vengono dichiarate le variabili globali che ospiteranno i dati provenienti dal sensore: UMID e TEMP, ambedue a virgola mobile (`float`). Ovunque nello sketch le 2 variabili potranno essere chiamate con: `minni.Data30003.TEMP` e `minni.Data30003.UMID`

La scelta dei nomi `packet30003`, `Data30003`, `pluto`, `minni` è opzionale, ma conviene usare questo metodo di assegnazione dei nomi per semplificare il lavoro di adattamento dello sketch ad altri sensori.

SensInit

```
//  put here instructions for initializing sensors and values for ID_TX and Length
void SensInit(){
     // Call dht.begin() to initialize the sensor and our data pin
  dht.begin();
  altDelay(1000);
}
```

In questa scheda sono inserite le istruzioni necessarie ad inizializzare il sensore e le librerie a questo necessarie.

ReadSensor

```
1  void ReadDHT() {
2   float t = dht.readTemperature(); // Gets the values of the temperature
3   float h = dht.readHumidity(); // Gets the values of the humidity
4      minni.Data30003.UMID = h;
5      minni.Data30003.TEMP = t;
6      if(ECHO){
7      // Now print the values:
8      Serial.println("Humidity: " + String(minni.Data30003.UMID, 1) + " %");
9      Serial.println("Temp (C): " + String(minni.Data30003.TEMP, 1) + " deg C");
10     }
11 }
```

Scheda con le istruzioni per la lettura del sensore e l'elaborazione del dato

Alle linee da 2 a 5 vengono assegnate alle variabili del sensore, dichiarate nella struttura packet30003, i valori di temperatura e umidità estratti dal sensore tramite le funzioni della sua libreria. Le linee dalla 6 in poi contengono le istruzioni di stampa sul monitor seriale del PC, solo se la variabile ECHO è messa a "TRUE" o 1.

DHT22_TX.ino

```
15 #include "Radio.h"
16 #include "Sens.h"
17 #include "Packet30003.h"
18 ////////////////////////////////////////////
19 byte ECHO  = 0  ;
20 ////////////////////////////////////////////
21 void setup() {
22
23   //  Initializing the serial monitor
24   if(ECHO){
25   while (!Serial);
26     Serial.begin(9600);
27     delay(100);
28     Serial.println("DHT22_TX.ino");
29   }
30
31   //  Initializing the radio
32     RadioInit();
33     SensInit();
```

```
35 //   First transmission with all the variables at 0
36     minni.Data30003.UMID = 0;
37     minni.Data30003.TEMP = 0;
38     Transmit();
39     altDelay(5000);
40 }
41 ////////////////////////////////////////////////////////////////////////////
42 void loop() {
43   ncicli++;
44   if(ncicli >= ncicliMax){
45
46 //  Read DHT22
47     minni.Data30003.UMID = 0;
48     minni.Data30003.TEMP = 0;
49     ReadDHT();
50
51 // Transmit data
52     Transmit();
53     ncicli = 0;
54   }
```

```
56   if(ECHO){
57      altDelay(5000);
58   }else{
59
60 //  The radio goes to sleep
61      rf95.sleep();
62 // The ProMicro goes to sleep for 20 seconds.
63      Watchdog.sleep(4000);
64      Watchdog.sleep(8000);
65      Watchdog.sleep(8000);
66   }
67 }
```

La scheda del main program contiene i vari `#include` delle schede `".h"`. Alla riga 19 c'è l'assegnazione alla variabile `ECHO` del valore 1 o 0 che serve, nel caso 1, allo sketch per attivare la scrittura sul monitor seriale della IDE. In ogni parte dello sketch, anche nelle altre schede, qualsiasi scrittura sul monitor seriale è contenuta in un blocco `if(ECHO)`.

Nel caso in cui `ECHO` è posto a 0, il monitor seriale non viene nemmeno inizializzato, quindi se, durante l'esecuzione è aperta una finestra del monitor, la CPU si blocca. Il cavo USB può essere ancora collegato alla board ma non devono essere aperte finestre monitor.

Nel blocco `setup`, vengono eseguite le varie inizializzazioni.

Nelle righe 36, 37 e 38 sono poste a 0 tutte le variabili del pacchetto trasmesso e chiamata la funzione `Transmit` che trasmette un primo messaggio tutto a 0.

Il blocco `loop` contiene le istruzioni eseguite ciclicamente. Il contatore ncicli serve ad eseguire il blocco di lettura e trasmissione dei dati periodicamente con un periodo voluto. Questo periodo è dato dal tempo in cui la CPU "dorme", dato dalle righe 63-65, moltiplicato per `ncicliMax`. Durante le prove e la messa a punto dello sketch, con `ECHO = 1`, è conveniente rallentare il loop con un ritardo, dato dalla funzione altDelay. Durante l'esecuzione operativa, con `ECHO = 0`, entrano in gioco le funzioni di sleep: quella alla riga 61 che mette a "dormire" il modulo radio e quella alle righe 63-65 che mettono a dormire la ProMicro.

In queste ultime il "sonno" dura i millisecondi scritti in argomento, con un massimo di 8000. Pertanto la ripetizione di questa istruzione consente di allungare questo tempo.

Il sonno del modulo radio dura invece fino alla chiamata di una qualsiasi funzione della libreria radio.

E' lecito chiedersi perché non inserire la chiamata alla funzione `Watchdog.sleep(8000)` in un ciclo `for` anziché riscriverla più volte. E' stato verificato che questa modalità non funziona. Inoltre l'esecuzione non è più stabile se si riscrive troppe volte questa funzione, questo spiega l'introduzione delle variabili `ncicli` e `ncicliMax`.

Skech per ricevitore

Il ricevitore ProMicroLoRa può essere inserito in diverse architetture a seconda delle necessità:

1. ProMicroLora + Display LCD 16x2;
2. ProMicroLoRa + ESP8266 + Display LCD 16x2;
3. ProMicroLoRa + ESP8266 + Arduino + DataLogger Shield + Display LCD 16x2;
4. ProMicroLoRa + Display OLED 128 x 32 (KIT);

Il Display OLED è del tipo con interfaccia I^2C

Le configurazioni 1 e 4 hanno solo la funzione di visualizzare sul display i dati ricevuti; la 2, oltre a questo, può connettersi ad Internet tramite un WiFi ed inviare i dati ad un server; la 3, oltre alle funzionalità della 1 e 2, memorizza i dati su una SD card tramite datalogger, tipo Adafruit. Di seguito si spiegherà il software per la configurazione 4.

Da tener presente che un ricevitore come questo capta qualsiasi segnale sulla frequenza 868 MHz, ma lo sketch è in grado di selezionare il pacchetto da processare in base all'identificativo di provenienza `ID_TX`. Inoltre la libreria e il modulo radio sono in grado di controllare la correttezza del messaggio per mezzo di tecniche CRC, restituendo allo sketch solo i messaggi corretti.

Si riportano qui di seguito le spiegazioni delle schede relative allo sketch per la configurazione 4 che comprendono, ma commentate, anche le istruzioni necessarie all'eventuale aggiunta di una scheda ESP8266. Saranno spiegato solo le schede che dovranno essere modificate nel caso della ricezione di altri TX.

scheda	Cosa contiene	Da Modificare se si aggiungono altri TX da ricevere
ProMicroLoRa_RX_OLED	Main program	No Settare la variabile ECHO secondo necessità
Packet.h	Dichiarative delle variabili misurate dai vari sensori TX che vogliamo ricevere. Le strutture sono uguali a quelle dei singoli sketch dei TX.	Si
RadioRX_ESP.h	Dichiarative per la radio	No
Action	Funzione che decide le azioni da fare in quando è ricevuto un messaggio da un TX	Si
Blinking	Funzione per lampeggio	No
Nword	Funzione per la trasformazione di una variabile float in un intero troncato senza segno	No
ESPTransmit	Funzione che trasmette i bytes ricevuti via radio alla scheda ESP8266 via protocollo seriale	No
Printxxxxx con xxxxx = 30003…	Funzione per stampare su PC e su Display. Ce n'è una per ogni TX, chiamata con la ID del TX (ultime 5 cifre)	Da creare per ogni nuovo TX solo se si vuole visualizzare quello che si è ricevuto su Display e PC
RadioRX_ESP_Init	Funzione per l'inizializzazione della radio	No
Receive	Funzione per la ricezione dati via radio	No

Packet.h

```
1  //  30003;     DHT22         byte 8 + 8;
2  typedef struct packet30003{
3  float       UMID;
4  float       TEMP;
5  };
6
7  //  30004;     Sonar SR04         byte 8 + 8;
8  typedef struct packet30004{
9  float       Dist;
10 long        Duration;
11 };
12
13 typedef union pluto {
14   packet30003 Data30003;
15   packet30004 Data30004;
16   byte bufVar[MXVAR];    //  array of bytes to be joined with the structure
17 };
18 pluto minni;
```

Contiene le strutture di ogni TX che deve essere ricevuto; in questo caso il 30003 e il 30004 (non compreso nel KIT)

Le strutture delle variabili, dichiarate negli sketch dei vari TX sono riportate tali e quali, una dopo l'altra, nella scheda `Packet.h`. La `typedef union` del TX deve essere riportata In fondo alla scheda (dalla linea 13), dove compare la definizione "union" con tutti i tipi dei pacchetti dichiarati precedentemente accanto al nome del pacchetto `Dataxxxxx`. Alla linea 16 è dichiarato un array byte `bufVar` dove saranno "travasati" i byte contenuti nel pacchetto ricevuto dal numero 8 in poi, `MXVAR` è la dimensione massima, posta a 50. Alla linea 32 il tipo `union pluto` viene attribuito alla variabile `minni`.

Action

```
void Action(){

//  Identification and re-transmission
          if(ID_TX == 30003){
               Print30003();

// Uncomment if you need to transmit data to the ESP8266
//              ESPTransmit();
          }else if(ID_TX == 30004){
               Print30004();
// Uncomment if you need to transmit data to the ESP8266
//               ESPTransmit();
          }else{
             if(ECHO){
                Serial.println(" ID_TX not permitted ");
             }
          }
      }
```

La funzione `Action()` smista il pacchetto ricevuto alle funzioni `Printxxxxx()` e `ESPTransmit()` che provvedono rispettivamente a stamparlo sul monitor seriale e sul display LCD e trasmetterlo alla board ESP8266. Quindi nella `Action()` è possibile decidere quali pacchetti ricevuti sono da passare alla ESP8266 per l'eventuale invio su Internet. Inoltre nella funzione Printxxxxx è possibile aggiungere o togliere altre modalità di visualizzazione dei dati o altre elaborazioni.

Print30003

```
//  SHT21
void Print30003(){
      if(ECHO){
         Serial.print(minni.Data30003.TEMP);Serial.print(" ");
         Serial.print(minni.Data30003.UMID);Serial.print(" ");
         Serial.println();
      }
//  Display OLED
    display.clearDisplay();
    delay(500);
    display.setTextSize(1);
    display.setTextColor(WHITE);
    display.setCursor(0,0);
    display.print("#");display.print(ID_TX);
//
  if(npacket > 9999){
    npacket = 0;
  }
```

```
   display.print(" N");display.print(npacket);
   display.print(" R");display.print(nRSSI,0);
   display.setCursor(0,16);
//   float VV = float(vBatTX/1000)
   display.print("V");display.print(vBatTX);
   display.print(" T");
   display.print(minni.Data30003.TEMP,1);
   display.print(" U");
   display.print(minni.Data30003.UMID,1);
   display.display();
}
```

Questa funzione ha il compito di stampare su monitor seriale i valori delle variabili TEMP e UMID e sul display OLED, oltre a questi valori anche l'indicativo del nodo TX, la tensione della batteria, l'intensità del segnale e il numero del pacchetto.

Sketch per ESP8266

ESP_exosite_General_V2.ino

La board riceve il pacchetto via seriale dal ricevitore ProMicroLoRa_RX e provvede ad inviare i dati ad un server Internet utilizzando una connessione WiFi.

scheda	Cosa contiene	Da modificare
ESP_exosite_General_V2	Main program	No

ExositeKeys.h	Contiene le chiavi per entrare nei device di Exosite	Si se si configura un nuovo device su Exosite
Packet.h	Uguale a quella del Ricevitore ProMicroLoRa	No
Variables.h	Dichiarative varie	No
SerialReceive	Funzione che riceve i dati dalla seriale.	No
WiFi_Transmit	Funzione che smista i vari pacchetti verso le funzioni di tipo http…	Si in base ai settaggi fatti su Exosite
connectWifi	Funzione che provvede a connettere ESP8266 alla rete WiFi	Si Solo per introdurre le credenziali di una nuova rete
http30003	Funzione che costruisce la stringa http da inviare al server Exosite. Una scheda di nome http(ID_TX) per ogni TX che si vuole inviare a Exosite	Si per ogni nuovo TX

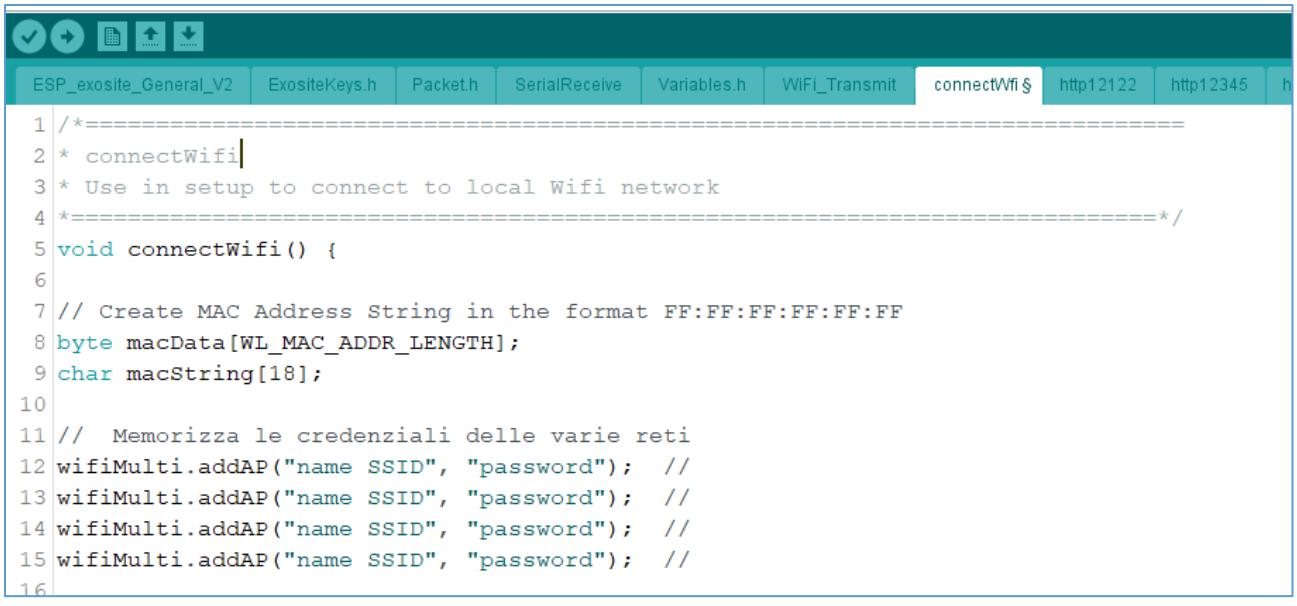
ExositeKeys.h

Qui sono riportati i "cik" assegnati da Exosite ogni qualvolta si definisce un "Device" su questo server. Per il server Exosite il Device è la board ESP8266 da cui arrivano i dati. I cik sono le chiavi per entrare nel server.

Lo sketch può inviare i dati solo ad un Device di Exosite. Ogni pacchetto ricevuto viene "inglobato" in una stringa ed inviato al server.

connectWifi

Per la connessione WiFi è necessario inserire nella scheda `connectWifi` le credenziali della rete WiFi (SSID e password) a cui si vuole connettersi. E' possibile inserire più reti, infatti le funzioni della libreria `<ESP8266WiFiMulti.h>` provvedono a collegare la board alla rete con segnale più forte.



WiFi_Transmit

```
void WiFi_Transmit(){
    buono = 0;

    //   in base all'identificativo chiama la funzione giusta
    if (ID_TX == 12345) {
      http12345();
      buono = 1;
    }
    if (ID_TX == 54321) {
      http54321();
      buono = 1;
    }
    if (ID_TX == 12122) {
      http12122();
      buono = 1;
    }
    if (ID_TX == 34567) {
      http34567();
      buono = 1;
    }
```

Per ogni identificativo di TX da ricevere si introduce in questa scheda un blocco:

```
if(ID_TX == xxxxx){
   httpxxxxx();
   buono = 1;
}
```

Le funzioni `httpxxxxx()` sono quelle che provvedono a costruire la stringa http da inviare su internet al server desiderato.

http30003

File Modifica Sketch Strumenti Aiuto

ESP_exosite_General_V2 | ExositeKeys.h | Packet.h | SerialReceive | Variables.h | WiFi_Transmit | connectWfi | http300

```
//  Temp Humi
void http30003(){

   String writeString1 = "SH_vBatTX_1=";
   String writeString2 = "&SH_npacket_1=";
   String writeString3 = "&SH_Temp_1=";
   String writeString4 = "&SH_Humi_1=";

//  trasformazione da formato numerico a carattere parte fissa
   String SvBatTX = String(vBatTX);
   String Snpacket = String(npacket);

//  trasformazione da formato numerico a carattere parte variabile
   String SH_Temp = String(minni.Data30003.Temp);
```

```
   TotalString = "";
   TotalString = writeString1 + SvBatTX +  writeString2 + Snpacket +
   writeString3 + SH_Temp  +  writeString4 + SH_Humi ;
;
    readString =  writeString1 + writeString2 + writeString3 + writeString4;
   if(ECHO){
     Serial.println(); Serial.println(" String transmitted to Exosite");
     Serial.println(TotalString);
   }
}
```

Questa scheda è stata scritta per il server Exosite (https://portals.exosite.com). Alle linee da 4 a 7 si assegnano alle variabili "stringa" `writeString1…` i nomi delle variabili così come sono state definite sul server Exosite (Data source). Dalla `writeString2` in poi si inizia con una "`&`" che serve al protocollo usato da Exosite. Dalla linea 10 alla 15 vengono trasformati i valori delle variabili numeriche in formato carattere, per mezzo della funzione `String`. Questa funzione trasforma in stringa di caratteri sia le variabili intere che float, a quest'ultime può essere specificato il numero di cifre dopo la virgola.

Alle linee 22 - 24 viene formata la stringa `TotalString` concatenando tutte le stringhe definite precedentemente. `readString` contiene invece le prime tre `writeString1,2,3 e 4.`
`TotalString` è quella che verrà inviata al server per fargli acquisire i dati, mentre `readString` viene usata per richiede indietro i valori (query) a scopo di verifica.