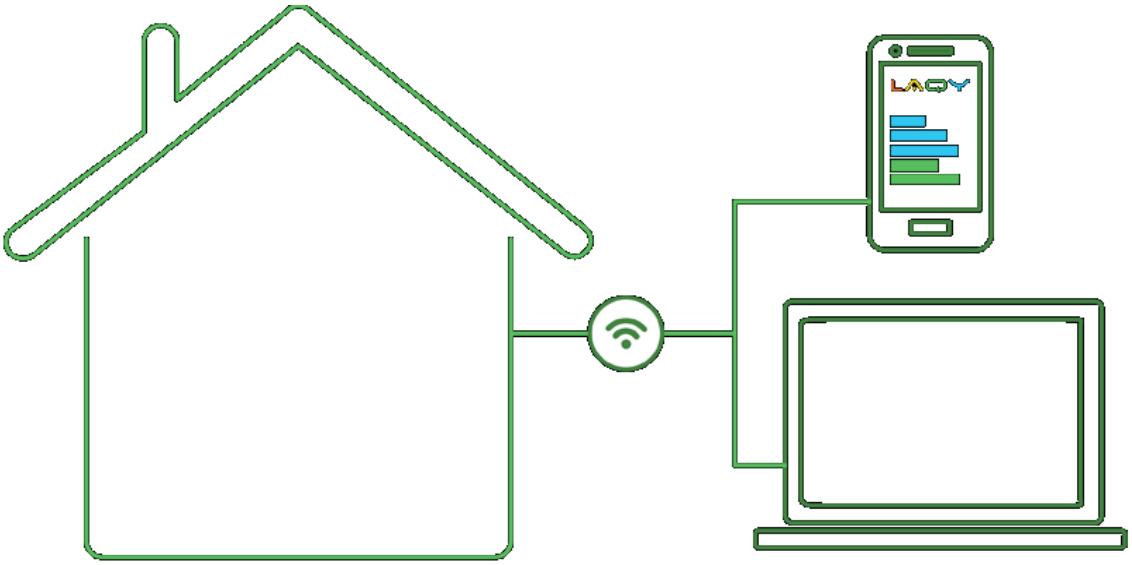
5^ELE
I.S.I.T.P.
VERRES

MONITORAGGIO E VISUALIZZAZIONE DATI DOMESTICI



GRUPPO: EMILE CHALLANCIN, SIMONE VUILLERMOZ E THIERRY BRUNIER

INDICE

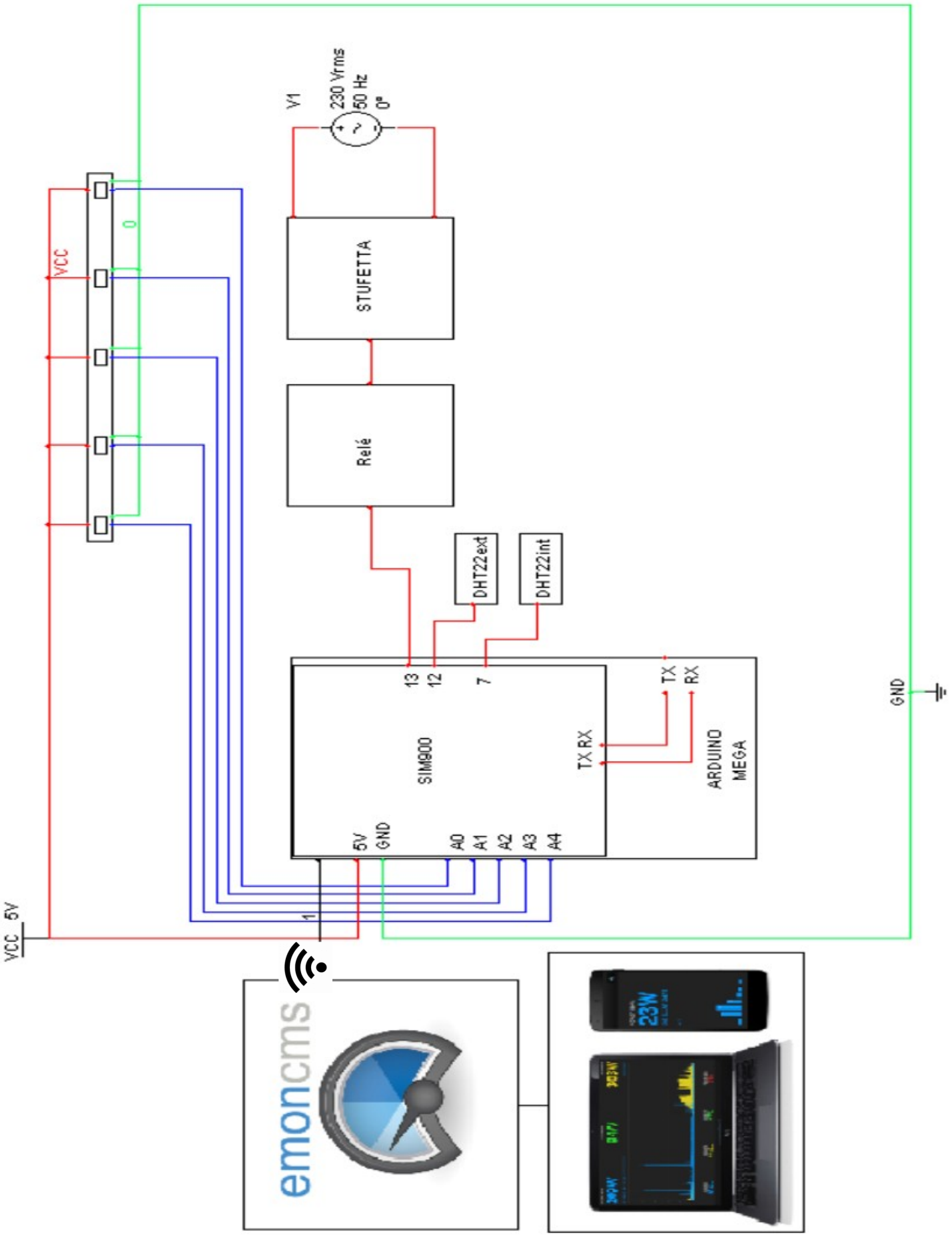
Introduzione

Per il nostro progetto di maturità abbiamo deciso di occuparci del rilevamento di dati della "Casa di Paglia", un edificio completamente verde, cioè un edificio che non provoca inquinamento ambientale e che migliora i consumi. Per questa ragione abbiamo deciso di visionare la temperatura, l'umidità, sia interni che esterni, e i consumi di potenza per poi poterli visualizzare e salvarli su un sito internet per dimostrare l'efficacia del progetto.

Perché abbiamo scelto questo progetto

Abbiamo scelto questo progetto poiché nella nostra scuola è stata installata una casa di paglia, allora abbiamo deciso di monitorare, con l'approvazione dei nostri professori, la temperatura e l'umidità di questa casa.
Quindi il nostro obbiettivo era quello di sfruttare una risorsa scolastica che, nel futuro, potrà essere ampliata e migliorata attraverso l'inserimento di altri sensori per renderla sempre più all'avanguardia.

SCHEMA A BLOCCHI

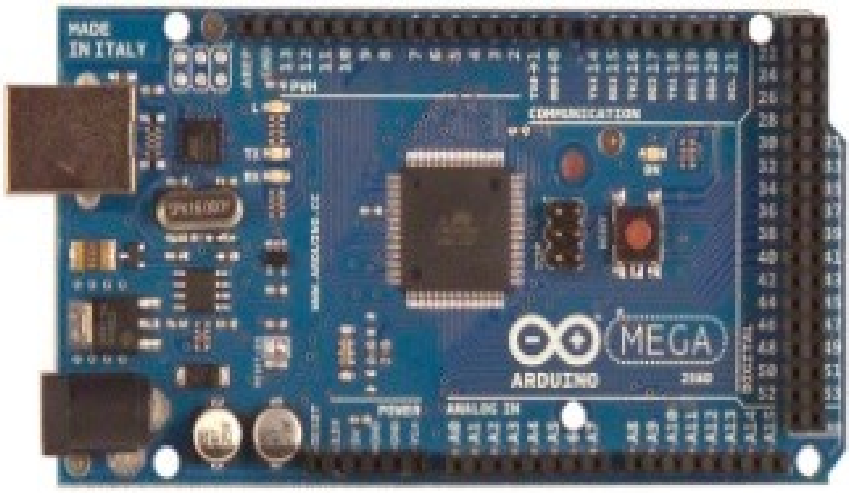


M UTILIZZATI

Per realizzare l'intero progetto abbiamo utilizzato diversi materiali:

Scheda Arduino Mega

L' Arduino Mega é una scheda elettronica basata su un microcontrollore ATmega. Possiede 54 pin digitali che possono essere utilizzati sia con funzioni di input sia come output. Possiede anche 16 pin analogici, che servono per trattare questi tipi di segnali. Rispetto all'Arduino UNO, quello più comune, possiede molti pin aggiuntivi che permettono l'utilizzo di più dati.



Per caricare il programma sulla scheda, il dispositivo viene collegato al computer con una presa USB e si utilizza un programma fornito dall'azienda. E' un linguaggio ad alto livello. Questa scheda elettronica rappresenta il cuore del nostro progetto poiché gestisce tutti i dati in entrata e permette l'invio di questi al server. In più con i pin 5V e GND alimenta tutto il circuito.

Abbiamo scelto questa versione di Arduino piuttosto che quella più comune (Arduino Uno) perché oltre ad avere più pin tratta meglio i dati (maggiore velocità e memoria)

Microcontroller	ATmega1280
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	128 KB of which 4 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

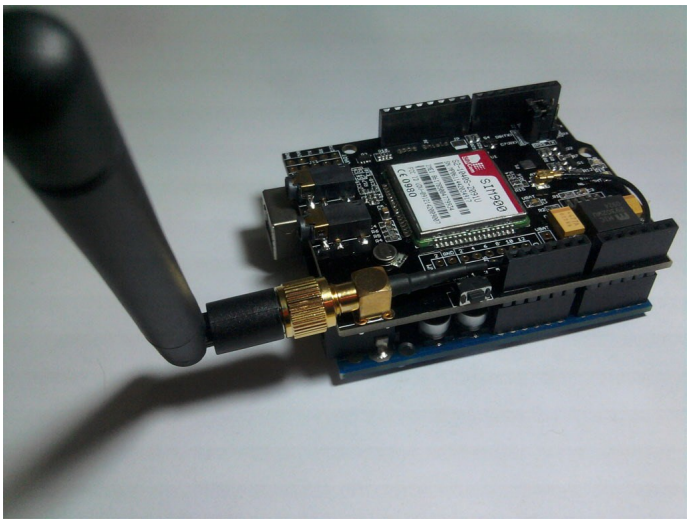
SIM900

Il modulo SIM900 è un modem seriale, ossia comunica con arduino attraverso due pin (2 e 3) in modalità Tx/Rx e si alimenta con i +5v e Gnd direttamente da Arduino.
Come tutti i modem seriali si possono utilizzare i comandi AT, con cui è possibile dialogare con il modem impostando modalità di connessione, velocità ecc..
I comandi AT non sono di certo semplici sopratutto se non conosci i protocolli e non hai chiaro come si esegue una connessione o cosa sia impostare i baud di una connessione GSM/GPRS.

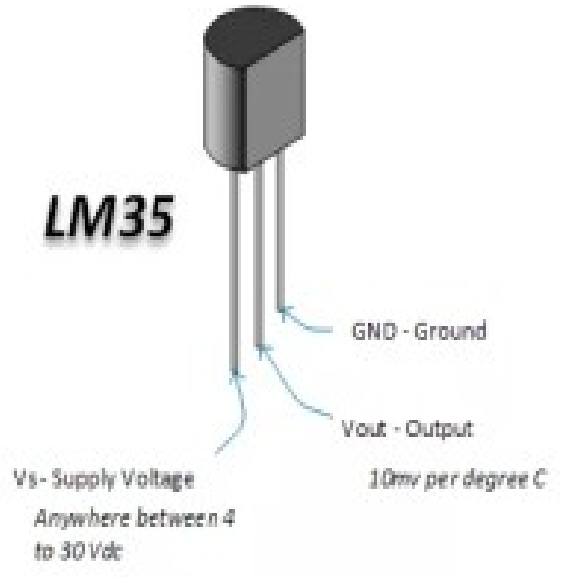
Montaggio della GPRS shield

La GPRS shield è già montata, quindi non servono saldature o complessi piani di assemblaggio, ma basta collegarla sopra ad una scheda Arduino Mega.

L'unica cosa che devi collegare è l'antenna, per farlo la GPRS shield dispone di un connettore SMA ed uno MMCX saldati sulla scheda e a cui puoi collegare l'antenna in dotazione

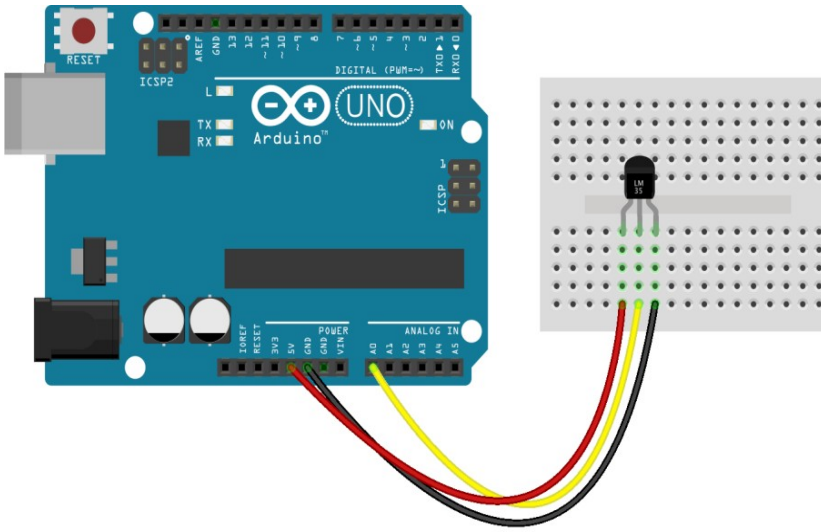


Sensori LM35



Il circuito integrato LM35 è un trasduttore di temperatura che fornisce una tensione d'uscita proporzionale alla temperatura espressa in °C (10mV/°C)
Il sensore possiede 3 terminali: alimentazione, massa, dato.
Abbiamo scelto questo tipo di sensore, poiché è semplice da utilizzare e convertire in digitale.
L'alimentazione del sensore va da 4V a 30 V.
Il range di temperatura misurabile da questo sensore varia da -55°C a 150°C circa.

Altezza	5.2mm
Profondità	4.19mm
Lunghezza	5.2mm
Funzione	Sensore di temperatura
Pin Count	3
Precisione	±0.6°C
Sensibilità	10mV/°C
Temperatura massima	150°C
Temperatura minima	-55°C
Tensione tipica di funzionamento	4-30V
Tipo uscita	Analogica



Sensori DHT22

DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND

Il DHT22 è un sensore di temperatura e umidità, semplice da usare, a basso costo e digitale .Utilizza un sensore capacitivo di umidità e un termistore per misurare l'aria circostante, e invia un segnale digitale al microcontrollore. Questo sensore invia i dati in seriale cioè un bit alla volta.

Il componente è composto da 4 pin: il primo viene collegato all'alimentazione, il quarto a massa, mentre il secondo è il dato in uscita, il terzo è un pin che non viene utilizzato

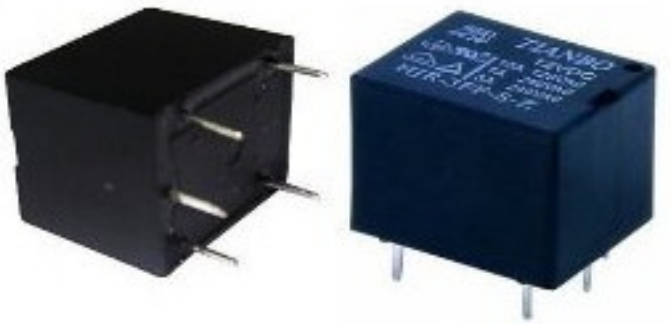
Abbiamo utilizzato questo dispositivo elettronico poiché rispetto al DHT11, questo sensore è più preciso e funziona in una più ampia gamma di temperatura / umidità.

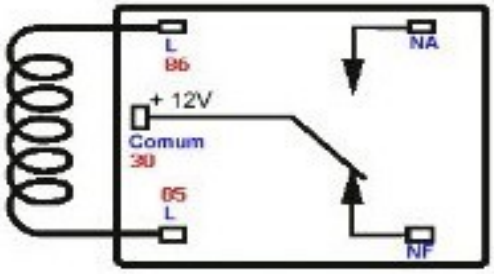
Rispetto al collegamento degli LM35 non abbiamo utilizzato dei cavi in comune per collegare massa e alimentazione.

Technical Specification:

Model	DHT22
Power supply	3.3-6V DC
Output signal	digital signal via single-bus
Sensing element	Polymer capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	humidity +-2%RH(Max +-5%RH); temperature <+-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Sensing period	Average: 2s
Interchangeability	fully interchangeable
Dimensions	small size 14*18*5.5mm; big size 22*28*5mm

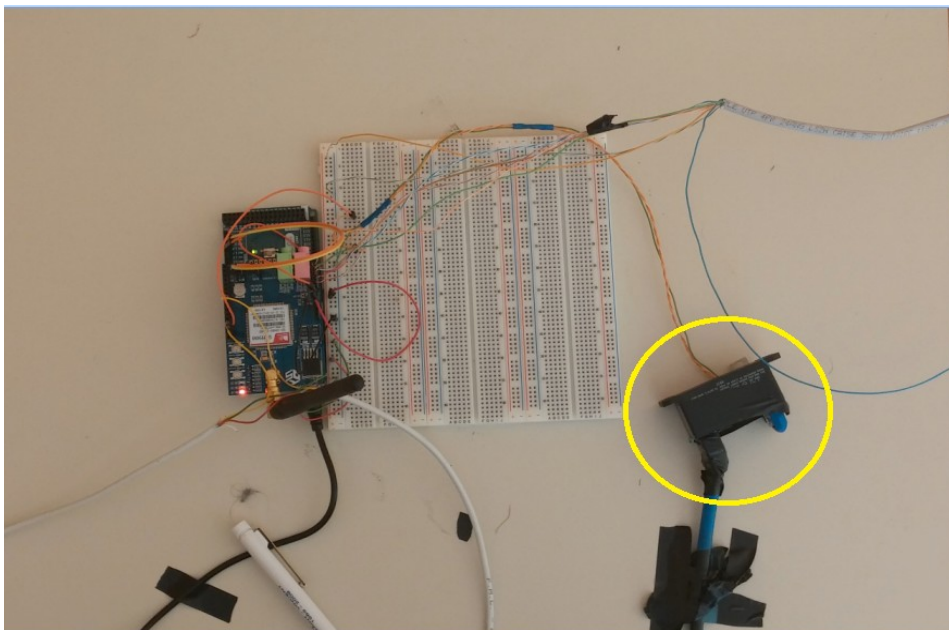
Relé





Il relè è un dispositivo elettrico comandato dalle variazioni di corrente per influenzare le condizioni di un altro circuito. In sostanza, il relè è un deviatore che non viene azionato a mano, ma da un elettromagnete.

Questo dispositivo ci serve per azionare la stufetta poiché quest'ultima richiede 230V d'alimentazione e la scheda Arduino non può fornirglieli. Allora il relé viene attivato, tramite un impulso del microcontrollore, il transistor si comporta come interruttore chiuso, e permette l'arrivo dell'alimentazione alla stufetta.

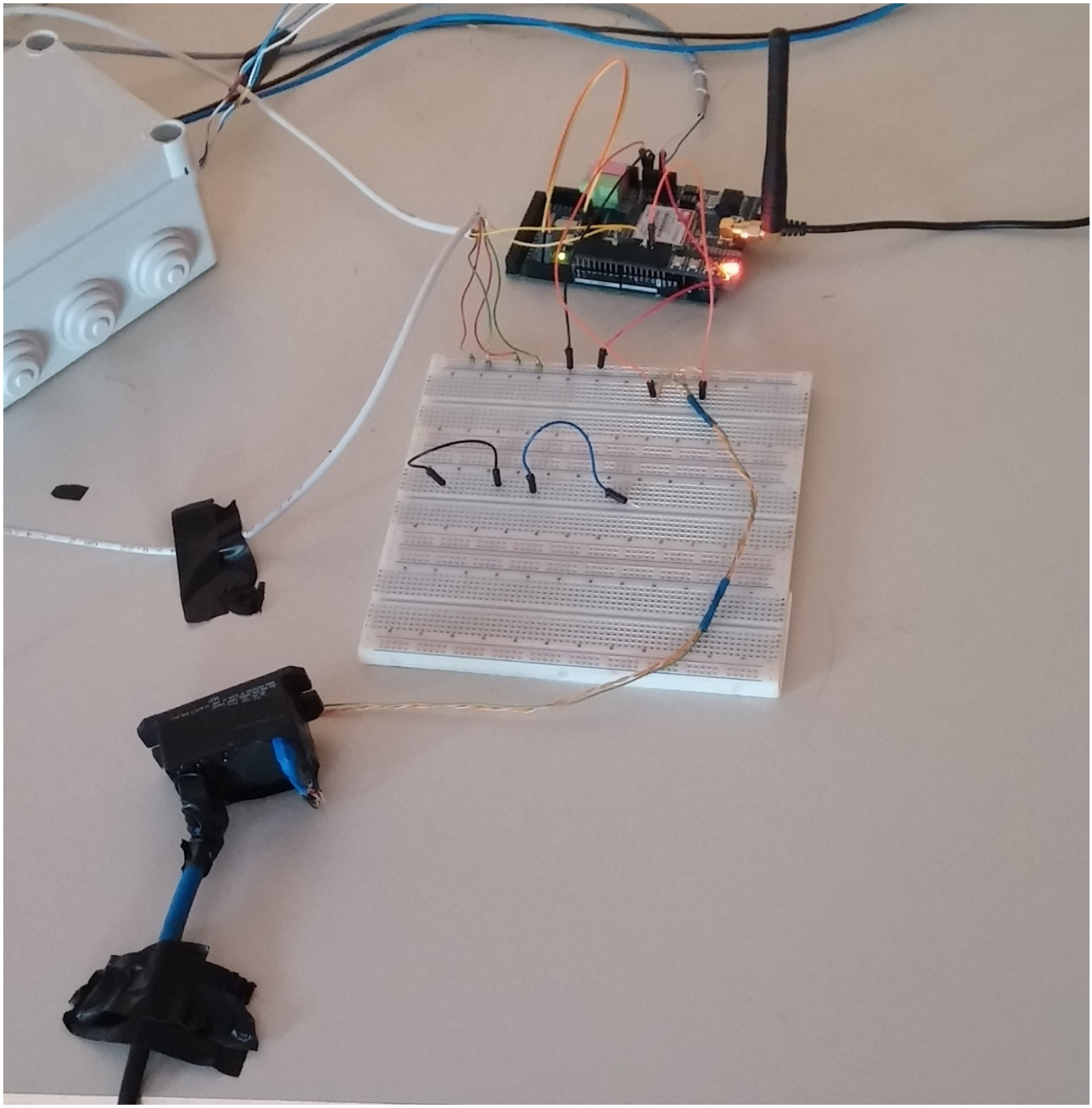


REALIZZAZIONE

GLI INIZI

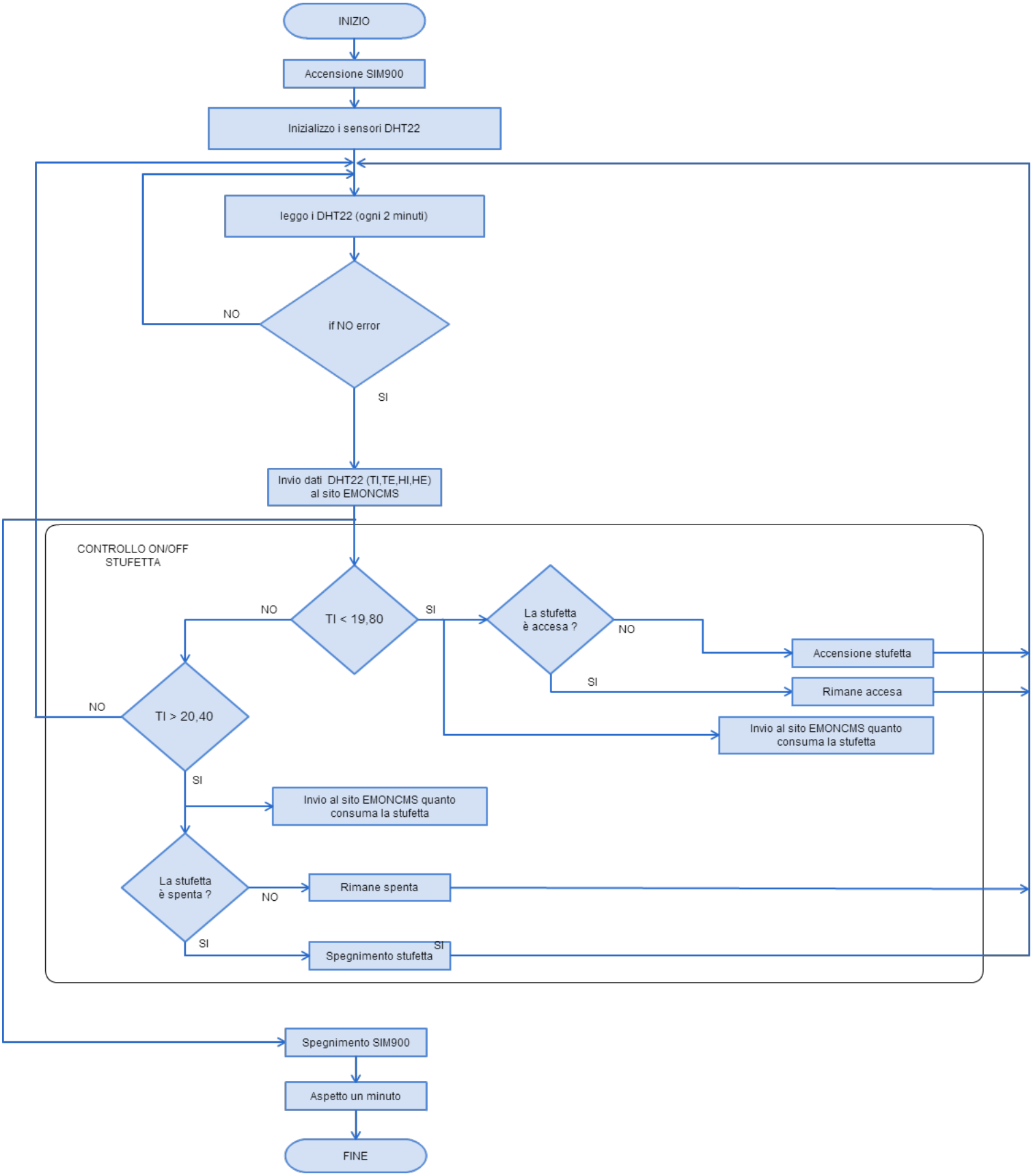
Innanzitutto, per svolgere il progetto che abbiamo deciso di portare avanti, abbiamo disegnato un semplice schema a blocchi del circuito che volevamo realizzare studiando quali dispositivi e quali materiali avrebbero fatto al caso nostro (vedi voce “schema a blocchi”).

In seguito, dopo aver scelto il tipo che materiali usare, abbiamo acquistato il necessario (vedi materiali utilizzati) e abbiamo fatto una prima costruzione di prova del circuito.

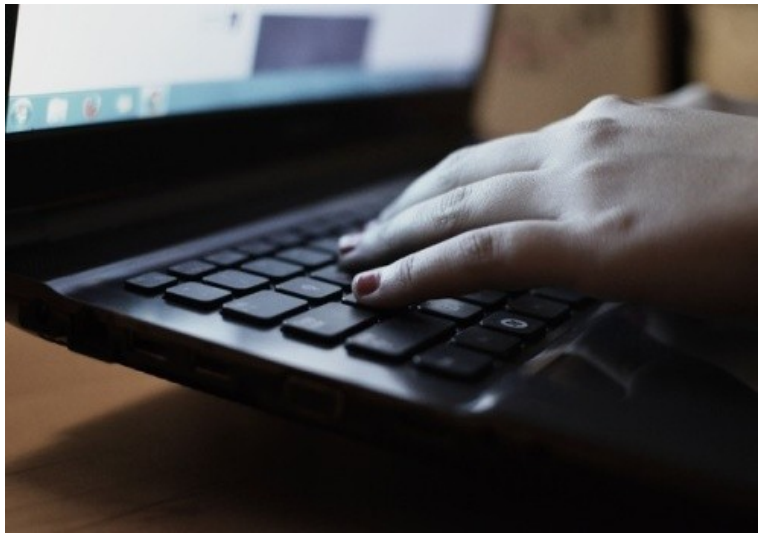


SOFTWARE - PROGRAMMA

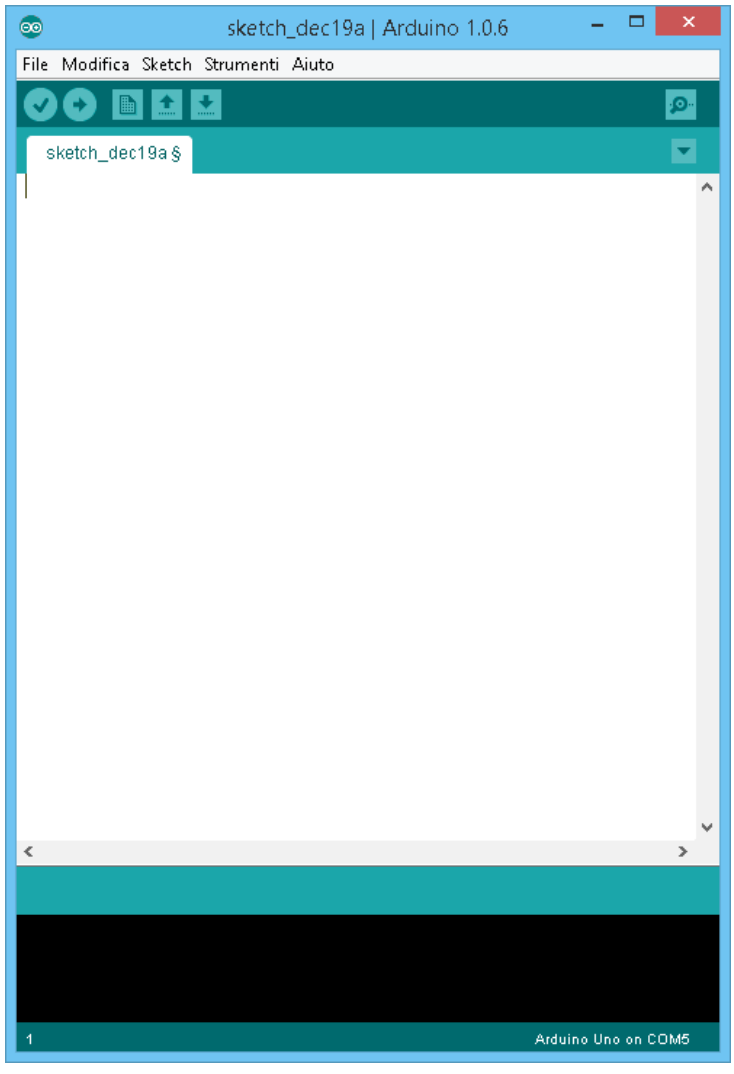
A questo punto abbiamo programmato la scheda Arduino Mega per ottenere i dati del nostro circuito di prova sul sito emoncms. Per fare ciò abbiamo iniziato a fare un flow-chart che ci permettesse di capire meglio quali funzioni di Arduino ci servivano.



Finito il flow-chart abbiamo scritto un primo programma



```
#include <DHT22.h>
#define SIM900_Power      8
#define DHT22_INT 7
#define DHT22_EXT 12
#define POWERRISC 13
#define TIntMin 19.80
#define TIntMax 20.40
//Dati SIM (TIM)
#define GPRS_APN        "ibox.tim.it"
#define GPRS_LOGIN      ""
#define GPRS_PASSWORD   ""
//FOR Arduino UNO
//SoftwareSerial SIM900(2,3);
//Prototipi Funzioni
void ShowSerialData();
void SIM900_Init();
void SIM900_ShutDown();
void SendDataToServer();
//Variabili Globali
boolean Start; //Primo Avvio Arduino
int RiscPrec;
//Variabili Database
String DatabaseIP = "emoncms.org";        //Indirizzo IP Server
String IndirizzoServer = "";              // per mettere l'indirizzo dell'APK
String IndirizzoServerRisc1 = "";
String IndirizzoServerRisc2 = "";
unsigned int DatabasePort = 80; //Porta Server
// Setup a DHT22 instance
DHT22 myDHT22INT(DHT22_INT);
DHT22 myDHT22EXT(DHT22_EXT);
DHT22_ERROR_t errorCodeint;
DHT22_ERROR_t errorCodeext;
//Per termostato
float  TInt;
////////////////////////////////////////////
///////Funzione di connessione GSM///////////NO
////////////////////////////////////////////
void SIM900_Init()
{
 pinMode(8, OUTPUT);
 digitalWrite(8,LOW);
 delay(1000);
 digitalWrite(8,HIGH);
 delay(2500);
 digitalWrite(8,LOW);
 delay(3500);
}
////////////////////////////////////////////
//////Funzione di disconnessione GSM/////////NO
////////////////////////////////////////////
void SIM900_ShutDown()
{
 pinMode(8, OUTPUT);
 digitalWrite(8,LOW);
 delay(1000);
 digitalWrite(8,HIGH);
 delay(2500);
 digitalWrite(8,LOW);
 delay(3500);
}
```



```
//////////////////////////////////////////
//////Funzione di Invio Dati Server//////////
//////////////////////////////////////////
void SendDataToServer(){
  static unsigned short int i;

  IndirizzoServer = "";
  errorCodeint = myDHT22INT.readData();
  errorCodeint = myDHT22EXT.readData();
  if(errorCodeint==DHT_ERROR_NONE && errorCodeext==DHT_ERROR_NONE )    //se non c'è nessun errore vuol dire che i dati sono affidabili e quindi li invia al server
     {

      IndirizzoServer="/api/post?json={TI:" + String(myDHT22INT.getTemperatureC(),2) + ",HI:" + String(myDHT22INT.getHumidity(),2) + ",TE:" + String(myDHT22EXT.getTemperatureC(),2) + ",HE:"  + String(myDHT22EXT.getHumidity(),2) + "}&apikey=bc6d7d7d92d5e5d7092a7e75c4311b80  HTTP/1.0";

      ShowSerialData();// this code is to show the data from gprs shield, in order to easily see the process of how the gprs shield submit a http request, and the following is for this purpose too.
      Serial1.println("AT+CGATT?");
      delay(100);

      ShowSerialData();
      Serial1.println("AT+SAPBR=3,1,\"CONTYPE\",\"GPRS\"");//setting the SAPBR, the connection type is using gprs
      delay(1000);

      ShowSerialData();
      Serial1.print("AT+SAPBR=3,1,\"APN\",\"");//setting the APN, the second need you fill in your local apn server
      Serial1.print(GPRS_APN);
      Serial1.println("\"");
      delay(4000);

      ShowSerialData();
      Serial1.println("AT+SAPBR=1,1");//setting the SAPBR, for detail you can refer to the AT command mamual
      delay(2000);

      ShowSerialData();
      Serial1.println("AT+HTTPINIT"); //init the HTTP request
      delay(2000);

      ShowSerialData();
      Serial1.print("AT+HTTPPARA=\"URL\",\"");// setting the httppara, the second parameter is the website you want to access
      Serial1.print(DatabaseIP);
      Serial1.print(IndirizzoServer);
      Serial1.println("\"");
      delay(1000);

      ShowSerialData();
      Serial1.println("AT+HTTPACTION=0");//submit the request
      delay(10000);//the delay is very important, the delay time is base on the return from the website, if the return datas are very large, the time required longer.
      //while(!SIM900.available());
      ShowSerialData();
```

```
////////////////////
    //Funzione Termostato
    ////////////////////
    TInt=myDHT22INT.getTemperatureC();

    if (TInt<TIntMin)
       {//Accendo Riscaldatore
       digitalWrite(POWERRISC,HIGH);

       if (RiscPrec==0)
          {
          //Invio stato riscaldatore al server
          IndirizzoServer="/api/post?json={pw:0,pw1:0}&apikey=bc6d7d7d92d5e5d7092a7e75c4311b80  HTTP/1.0";
          Serial1.println("AT+HTTPINIT"); //init the HTTP request
          delay(2000);

          ShowSerialData();
          Serial1.print("AT+HTTPPARA=\"URL\",\"");// setting the httppara, the second parameter is the website you want to access
          Serial1.print(DatabaseIP);
          Serial1.print(IndirizzoServer);
          Serial1.println("\"");
          delay(1000);

          ShowSerialData();
          Serial1.println("AT+HTTPACTION=0");//submit the request
          delay(10000);//the delay is very important, the delay time is base on the return from the website, if the return datas are very large, the time
required longer.
          //while(!SIM900.available());
          RiscPrec=1;
          }

          IndirizzoServer="/api/post?json={pw:829,pw1:829}&apikey=bc6d7d7d92d5e5d7092a7e75c4311b80  HTTP/1.0";
          ShowSerialData();
          Serial1.println("AT+HTTPINIT"); //init the HTTP request
          delay(2000);

          ShowSerialData();
          Serial1.print("AT+HTTPPARA=\"URL\",\"");// setting the httppara, the second parameter is the website you want to access
          Serial1.print(DatabaseIP);
          Serial1.print(IndirizzoServer);
          Serial1.println("\"");
          delay(1000);

          ShowSerialData();
          Serial1.println("AT+HTTPACTION=0");//submit the request
          delay(10000);//the delay is very important, the delay time is base on the return from the website, if the return datas are very large, the time
required longer.
          //while(!SIM900.available());

       }
    if(TInt>TIntMax)
       {//Spengo Riscaldatore
       digitalWrite(POWERRISC,LOW);

       if (RiscPrec==1)
          {
          //Invio stato riscaldatore al server
          IndirizzoServer="/api/post?json={pw:829,pw1:829}&apikey=bc6d7d7d92d5e5d7092a7e75c4311b80  HTTP/1.0";
          Serial1.println("AT+HTTPINIT"); //init the HTTP request
          delay(2000);

          ShowSerialData();
          Serial1.print("AT+HTTPPARA=\"URL\",\"");// setting the httppara, the second parameter is the website you want to access
          Serial1.print(DatabaseIP);
          Serial1.print(IndirizzoServer);
          Serial1.println("\"");
          delay(1000);
```

```
ShowSerialData();
            Serial1.println("AT+HTTPACTION=0");//submit the request
            delay(10000);//the delay is very important, the delay time is base on the return from the website, if the return datas are very large,
the time required longer.
            //while(!SIM900.available());
            RiscPrec=0;
            }

            IndirizzoServer="/api/post?json={pw:0,pw1:0}&apikey=bc6d7d7d92d5e5d7092a7e75c4311b80  HTTP/1.0";
            ShowSerialData();
            Serial1.println("AT+HTTPINIT"); //init the HTTP request
            delay(2000);

            ShowSerialData();
            Serial1.print("AT+HTTPPARA=\"URL\",\"");// setting the httppara, the second parameter is the website you want to access
            Serial1.print(DatabaseIP);
            Serial1.print(IndirizzoServer);
            Serial1.println("\"");
            delay(1000);

            ShowSerialData();
            Serial1.println("AT+HTTPACTION=0");//submit the request
            delay(10000);//the delay is very important, the delay time is base on the return from the website, if the return datas are very large,
the time required longer.
            //while(!SIM900.available());
          }

      ShowSerialData();
      Serial1.println("AT+CSQ");
     // SIM900.println("AT+HTTPREAD");// read the data from the website you access
      delay(300);a

      ShowSerialData();
      Serial1.println("");
      delay(100);
      } // if
}
void ShowSerialData(){
  while(Serial1.available()!=0)
    Serial.write(Serial1.read());
}
```

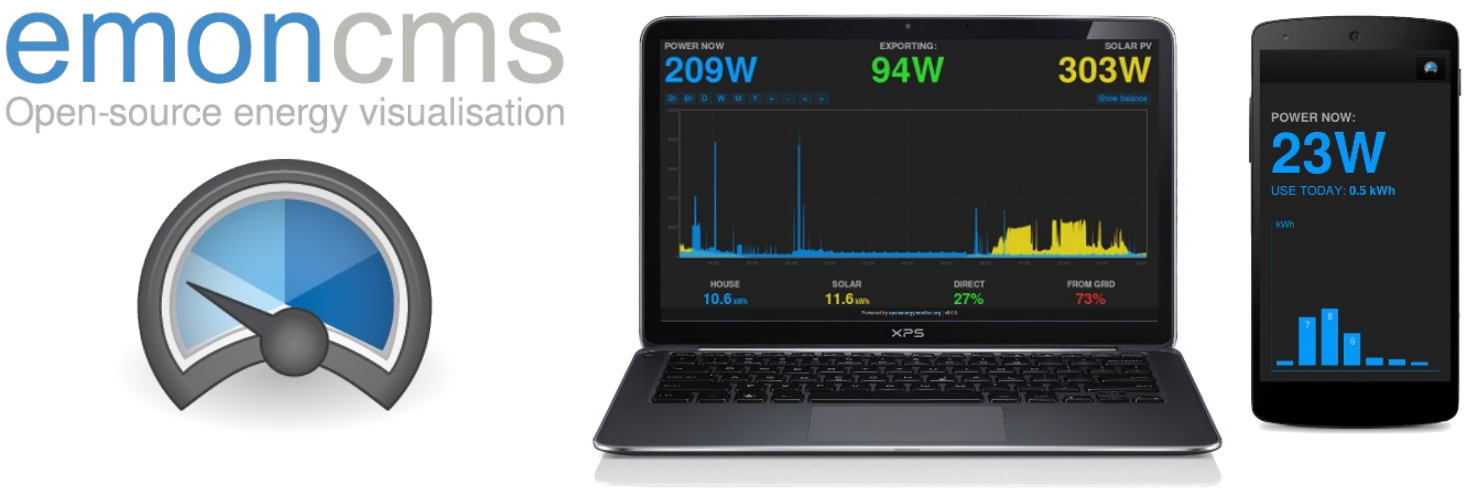
```
//////////////////////////////////////////
///////////////PROGRAMMA////////////////////
//////////////////////////////////////////
void setup()
{
  pinMode(SIM900_Power,OUTPUT);    //PIN Power SIM900 di Out
  pinMode(POWERRISC,OUTPUT);       //PIN Riscaldatore di Out
  Serial1.begin(9600);
  Serial.begin(9600);
  digitalWrite(POWERRISC,LOW);
  Start = true;
  RiscPrec=0;
}
void loop(){

    Serial.println("Avvio");
    Serial1.flush();
    SIM900_Init();
    delay(10000);
    ShowSerialData();

    //Invio dati al server
    Serial.println("primasendserver");
    SendDataToServer();
    Serial.println("doposendserver");
    //Fine invio dati al server
    ShowSerialData();
    SIM900_ShutDown();
    ShowSerialData();
    Serial.println("Fine");
    delay(60000);


}
//Fine programma
```

EMONCMS - INTRODUZIONE

Questo programma, come già detto in precedenza, invia i dati ricevuti dai sensori al sito Emoncms.

Cos'è Emoncms?

Emoncms è un sito web con un'interfaccia grafica costituita da una serie di moduli per archiviare i dati e le varie opzioni dell'interfaccia grafica stessa che è totalmente personalizzabile.



Emoncms è composto da vari moduli, ognuno deputato ad una specifica operazione:

Input: Questo modulo si occupa della manipolazione ed elaborazione vera e propria dei dati che possono essere memorizzati grezzi.

Feeds: Questo modulo gestisce la memorizzazione dei dati degli input e si occupa di passarli al modulo successivo. Sempre all'interno di questo modulo si può visualizzare l'attività di invio dei dati.

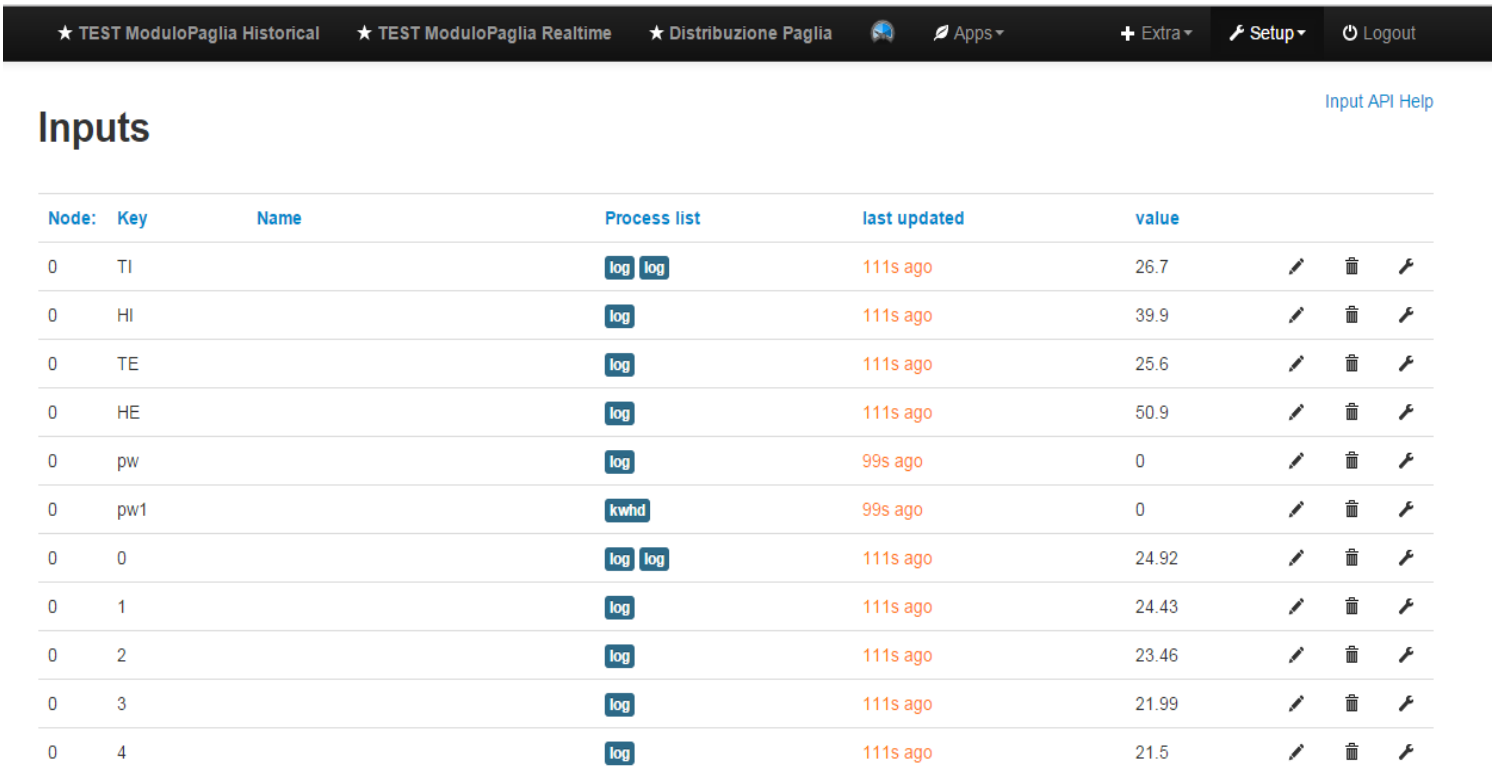
Vis (Visualisations): con questo modulo si possono visualizzare al volo i dati con dei grafici temporanei. Questo modulo è molto importante poiché lui imposta i grafici multilinea. Possiede inoltre la funzione di modifcare eventuali dati errati nel database.

Dashboard: E' la lavagna dove andiamo a posizionare i vari grafici e i dati da visualizzare, c'è la possibilità di ragruppare i dati dividendoli su più pagine.

Extras: Con questo modulo abbiamo la possibilità di visualizzare in modo diverso i dati a livello globale e generare dei report. Noi non lo abbiamo utilizzato, poiché per i nostri scopi era più che sufficiente la sola visualizzazione sulle Dashboards.

EMONCMS - INPUTS

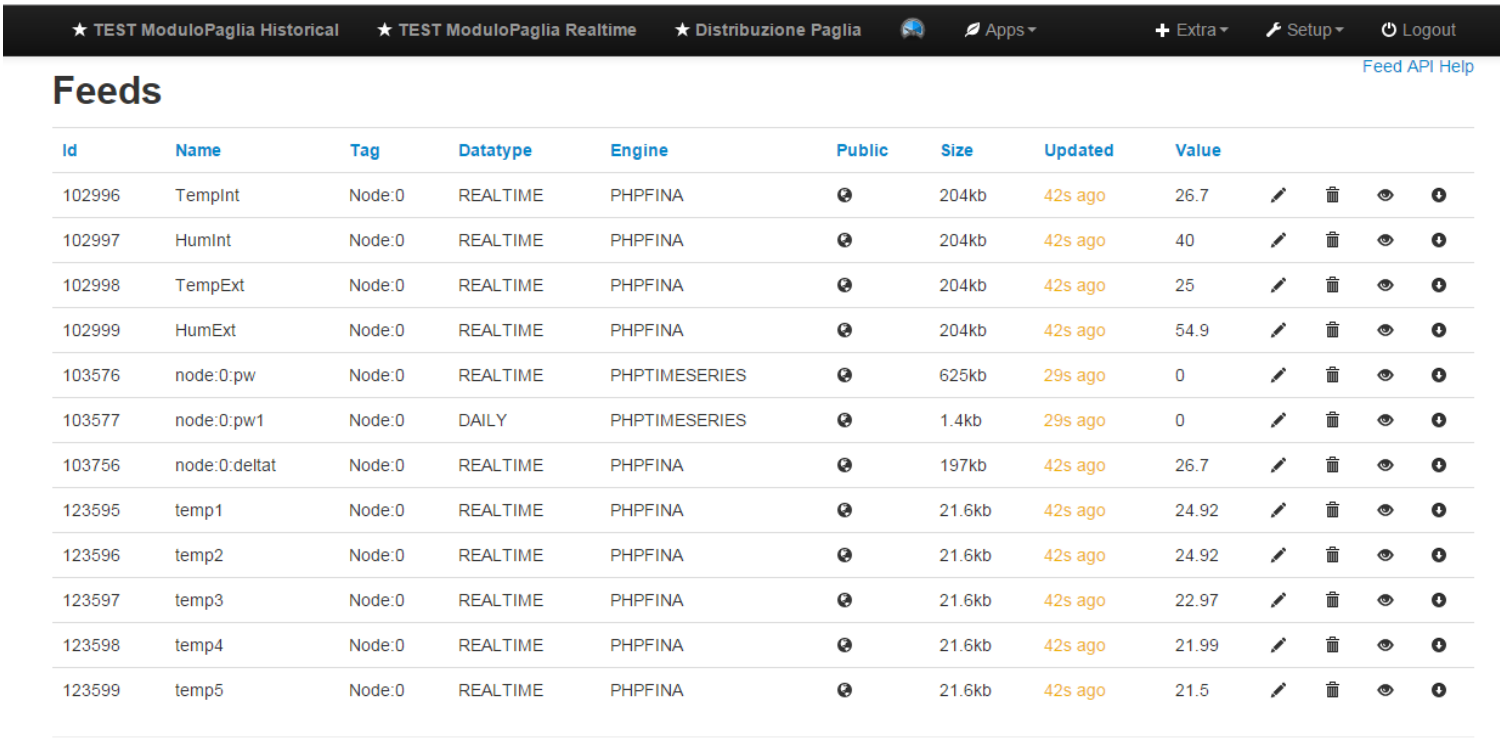
Questo modulo raccoglie i dati che arrivano dal monitoraggio e li invia ai feed per l'archiviazione, così come sono (grezzi), o formattati per poi inviarli. Qualsiasi elaborazione che si voglia fare con i dati deve essere fatta all'interno di questo modulo. I dati cosi processati vengono archiviati dai Feed e resi disponibili per gli altri moduli.
Per esempio si può convertire un dato in kW/h oppure svolgere pure dei calcoli matematici, come un addizione o una sottrazione (funzioni da noi non utilizzate).



Il nome della casella “Name” é quello che noi abbiamo dato nella stringa d'invio sul programma di Arduino, infatti mantengono i nomi originali (TI, HI, TE, HE...).
Ora bisogna formattare i dati, cliccando sull'icona a forma di chiave inglese della riga del dato che ci serve, si apre la finestra dei processi. Qui si imposta anche ogni quanto l'Input invia il segnale al feed.
Abbiamo quindi impostato prevalentemente la funzione “Log to Feed” che invia i dati direttamente al feed, poiché la conversione l’abbiamo già fatta sull'Arduino e i dati inviati sul sito sono già corretti e pronti alla visualizzazione tranne per la potenza consumata dalla stufetta, che é misurata in kWh e kWh/d per misurare il consumo energetico, abbiamo infatti sfruttato l'opportunità di mettere più funzioni per lo stesso dato.
In conclusione questa è la nostra pagina di input.

EMONCMS - FEEDS

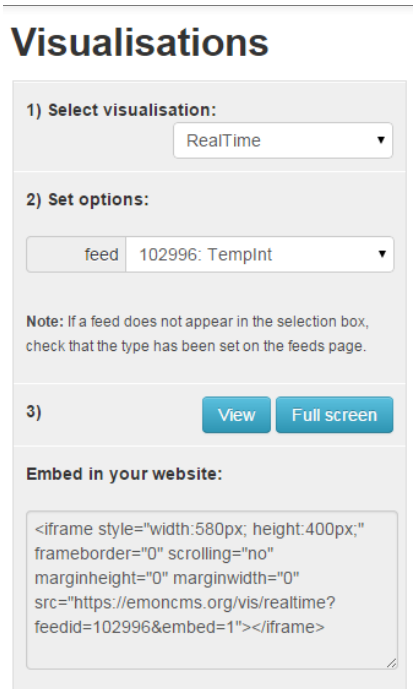
La pagina feed é quella dove è possibile trovare i dati formattati. Andando a cliccare sul menu vedremo tutti i nostri dati



Vediamo il contenuto di questa tabella:

- Id = identificativo del processo
- Name = nome assegnato al feed
- Datatype = tipo di dato contenuto (Realtime, daily, histogram), impatta sul modo di visualizzare il dato
- Public = imposta se il dato può essere pubblico o privato.
- Size = visualizza la dimensione dei dati salvati su questo feed (cresce col tempo)
- updated = intervallo di tempo trascorso dall'ultimo aggiornamento
- Value = ultimo valore menorizzato
- Icone = ogni riga ha a disposizione 3 icone:
  - Matita = cliccando si entra in modifica, si può variare "Name", "Datatype".
  - Cestino = serve a eliminare il feed.
  - Occhio = serve a verificare al volo il contenuto del feed, si apre un grafico e visualizza il contenuto.

EMONCMS - VISUALISATIONS



Su questa sezione, come già detto in precedenza, abbiamo la possibilità di visualizzare al volo i dati dei Feed mediante dei grafici. Oltre a questo possiamo, impostare i grafici multilinea. Passiamo ora a vedere le varie opzioni che abbiamo a disposizione.

1) Cliccando sul menù a tendina si apre un menù con tutti i grafici che abbiamo a disposizione.

2) Selezionando una voce del menù verranno aggiornate le opzioni che variano in funzione del grafico scelto

3) Impostate le varie opzioni cliccando su uno dei due bottoni del box vedremo apparire il grafico.

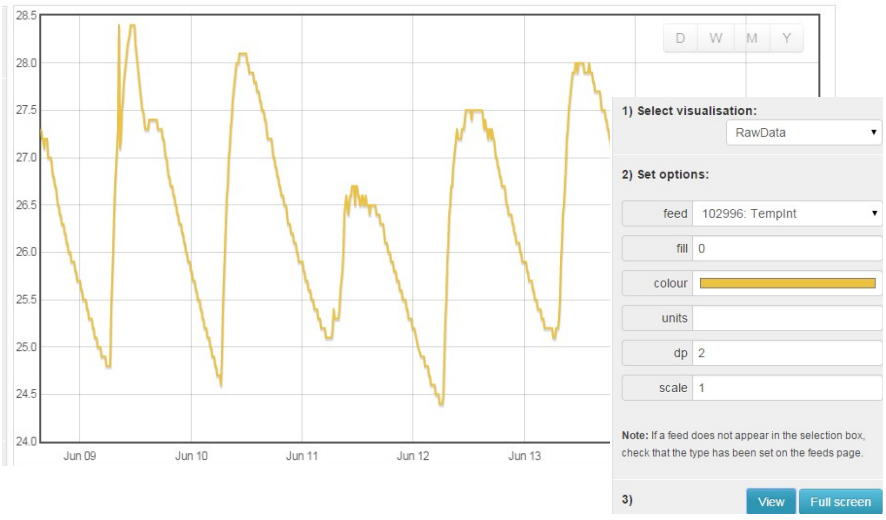
Per ogni tipo di grafico verranno proposti i feed compatibili, per questo motivo possiamo non vedere tutti i feed che abbiamo inserito.

Passiamo ora a vedere i vari grafici che abbiamo utilizzato.

REALTIME

Su questo grafico, scegliendo uno dei feed disponibili, apparirà l'andamento del valore in continuo aggiornamento.

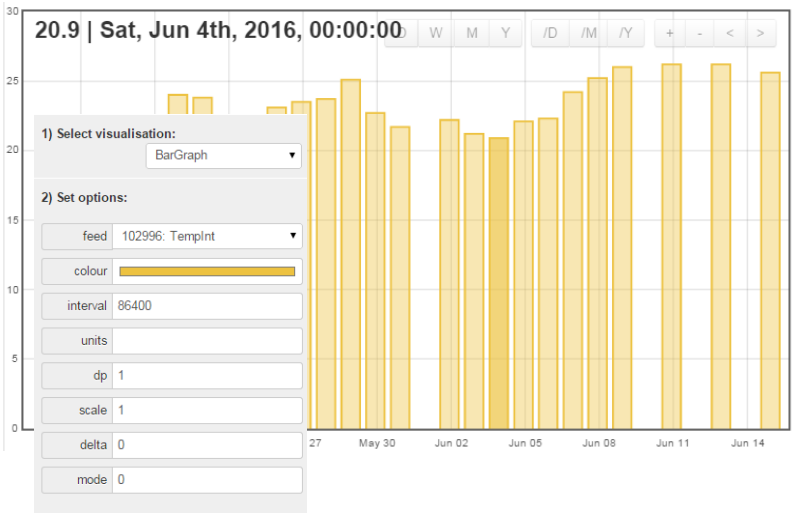
Sull'angolo alto sinistro del grafico ci sono dei bottoni per modificare il periodo di tempo visualizzato.



RAW DATA

Come al solito selezioniamo il grafico sul primo box e il feed sul secondo, su questo box ci sono due opzioni:

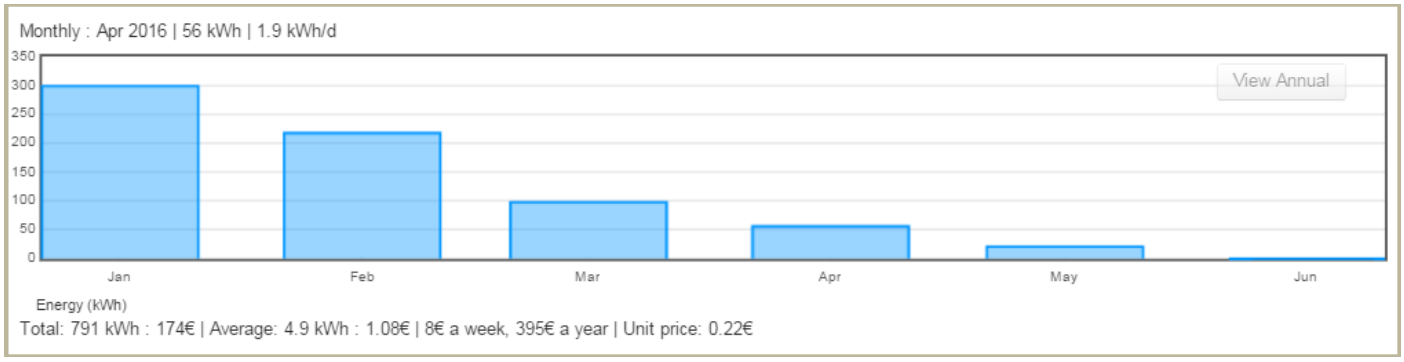
- fill, serve per riempire l'area del grafico sotto la linea dell'andamento del valore.
- units, imposta il tipo di unità da visualizzare sul grafico
- dp/scale, imposta il periodo da visualizzare.



BARGRAPH

(DIFFERENZA EXT-INT)
Questo grafico visualizzerà i dati in un formato a barre verticali con il valore giornaliero di quel dato. Anche qui è possibile scegliere il colore delle barre e il periodo da visualizzare.

ZOOM



In questo grafico si devono selezionare due feed, la potenza istantanea e la potenza giornaliera. Questo grafico é particolarmente utile poiché consente di visualizzare il consumo e il costo della potenza.

MULTIGRAPH

Questa visualizzazione ci permette di creare dei grafici con dati da visualizzare contemporaneamente. In questo caso vengono visualizzati i dati relativi all'umidità interna ed esterna

EMONCMS - DASHBOARD

Introduzione

La Dashboard serve a visualizzare mediante grafici o widget (impostazioni grafiche) i dati impostati da noi.. In pratica si tratta di impostare una o più pagina web che saranno visualizzate da chi entra nel nostro sito sul lato pubblico. La dashboard è una pagina vuota dove possiamo posizionare i vari oggetti. La gestione della dashboard è orientata agli oggetti, quindi per qualsiasi oggetto si seguirà questo procedimento:

- Seleziono un oggetto
- Lo posiziono sulla dashboard
- Gli do le dimensioni volute
- Gli imposto le opzioni disponibili

Come creare una pagina (Dashboard)

Entrati nella sezione Dashboard cliccare sul simbolo "+" in alto a destra e a questo punto verrà generata l'intestazione della prima pagina, vediamo le varie voci:

- Id: è il numero univoco con cui viene riconosciuta la pagina nel database.
- Name: nome della pagina.
- Alias: non utilizzata, nome alternativo per database.
- Main: identifica se la pagina è la Main o meno. Può esserci una sola Main.
- Public: identifica se la pagina è privata (lucchetto) o pubblica (mondo).
- Published: indica se la pagina è stata pubblicata o meno.
- Cliccando su questa icona si crea una copia della pagina.
- Matita: cliccando su questa icona si può modificare il nome.
- Cestino: elimina la pagina selezionata.
- Con questa icona si entra in modalità modifica della pagina corrente.
- Cliccando su questa icona a forma di occhio si passa a visualizzare l'anteprima della pagina cliccata.

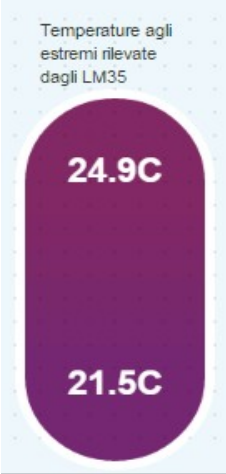
Text

Gli oggetti di tipo "Text" sono 3:

Paragraph: Permette di inserire del testo multiriga
Heading: Inserisce un titolo allineato a sinistra
Heading-center: Inserisce un titolo centrato

Widgets

Cliccando sul label "Widgets" si apre una tendina con tutti i vari elementi grafici che abbiamo a disposizione.

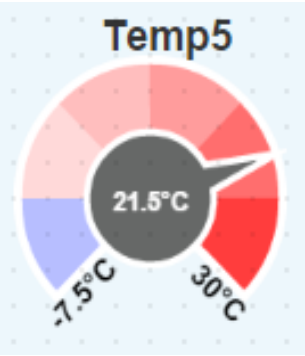


CYLINDER:

Abbiamo utilizzato questo widget per monitorare la differenza di temperatura da un punto all'altro della striscia di sensori LM35. In funzione dei due valori passati il widget si colora da Viola (freddo) a Rosso (caldo). Se le due temperature sono molto diverse colora il cilindro con gradiente di colore dal blu al rosso. Il quadrante può essere spostato e ridimensionato.

DIAL:

Si tratta di quadranti dinamici in tempo reale che si adeguano al valore del feed. Il quadrante cambia da un valore al successivo partendo velocemente e rallentando all'avvicinarsi del valore finale. Il quadrante può essere spostato e ridimensionato.



FEEDVALUE:

Questo widget è utilizzato per visualizzare il valore di un feed

JGAUGE:

E' un quadrante come i Dial ma con una grafica più accattivante, ne esiste un solo tipo e non è utilizzabile per i valori negativi.

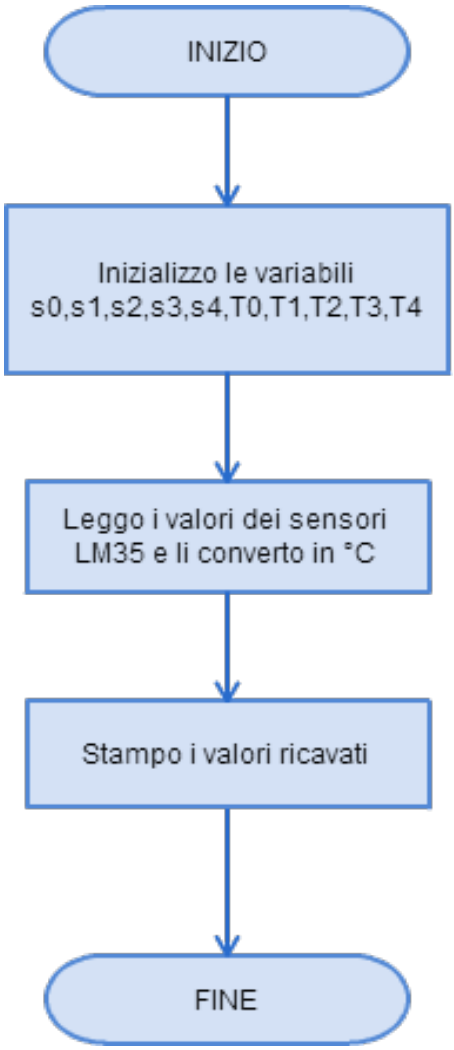
STECCA DI LM35

Dopo aver testato che tutto funzionasse, abbiamo deciso di ampliare il progetto. Per questo abbiamo creato una striscia di LM35 posta su una stecca di legno con l'obiettivo di vedere la distribuzione della temperatura all'interno di un muro di paglia

Inizialmente abbiamo scelto la lunghezza della stecca in base alla profondità del muro che è di circa 32 cm. A questo punto abbiamo preso un pezzo di legno che potesse fare al caso nostro e ne abbiamo definito le misure con "nome macchinario".

Fatto ciò, dopo aver progettato il circuito, abbiamo fatto dei fori sulla stecca per far passare i piedini degli LM35. Piedini che abbiamo poi collegato all'Arduino Mega (vedi come collegare LM35 nella descrizione del componente). I sensori sono stati posti sulla stecca ad una distanza di 8 cm l'uno dall'altro.
Dopo alcune prove utili a determinare che il circuito funzioni correttamente, abbiamo messo della colla a caldo tra i sensori per immobilizzarli

Terminata la parte hardware, ci siamo concentrati sul software




```
int s0;
int s1;
int s2;
int s3;
int s4;
float T0;
float T1;
float T2;
float T3;
float T4;
void setup()
{
 Serial.begin (9600);
}
void loop()
{
 s0=analogRead (A0);
 s1=analogRead (A1);
 s2=analogRead (A2);
 s3=analogRead (A3);
 s4=analogRead (A4);

 T0=s0*0.4887;
 T1=s1*0.4887;
 T2=s2*0.4887;
 T3=s3*0.4887;
 T4=s4*0.4887;

 Serial.println (T0);
 Serial.println (T1);

 Serial.println (T2);

 Serial.println (T3);
 Serial.println (T4);
 delay (5000);
}
```

Le funzioni per convertire il dato in °C sarebbero:

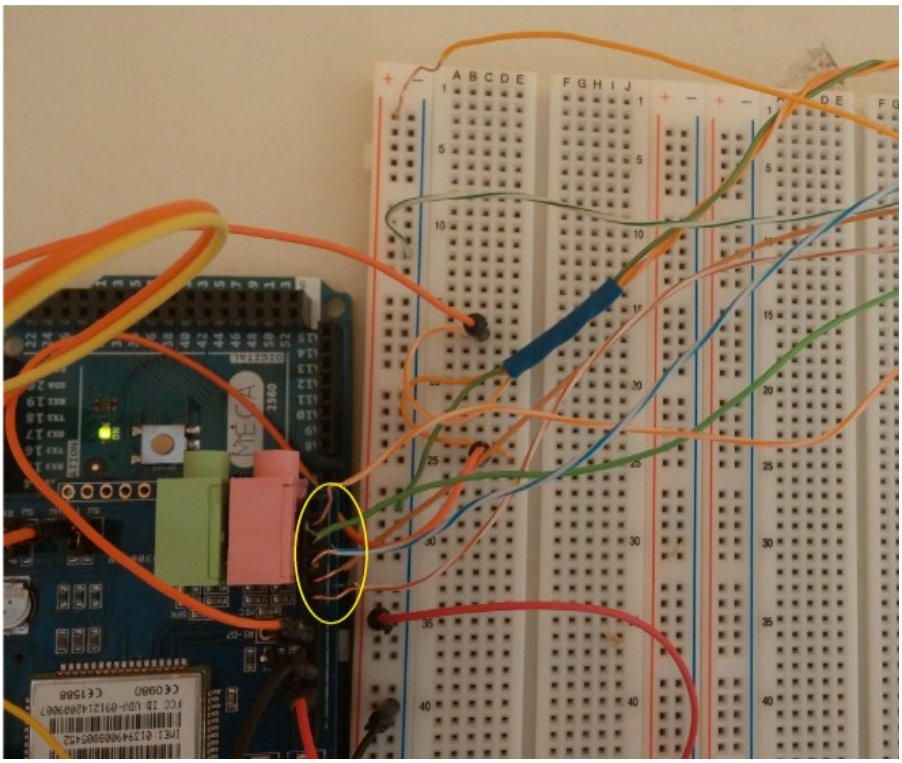
mV = (sensor/1023.0)*5000;
°C = mV/10

Inizialmente abbiamo utilizzato entrambe le formule, ma il programma ci dava inspiegabilmente dei problemi . Problemi che abbiamo risolto facendo la conversione in un unico passaggio.

Terminato sia l'hardware sia il software della stecca di LM35 abbiamo aggiunto tali linee di codice nel programma originale mettendo l'acquisizione dei dati e la loro conversione all'interno del primo [if] e facendo si che invece di stampare i valori ricavati, questi arrivino direttamente al sito Emoncms.

A questo punto abbiamo coperto tutta la stecca con del nastro isolante per evitare che eventuali fili si scolleghino

Infine abbiamo montato il tutto all'interno della casa di paglia. C'è da considerare che la striscia di LM35 è stata posta nella parte opposta rispetto ai DHT22 quindi è possibile che ci sia una leggera differenza di temperatura causata anche da una diversa sensibilità e precisione dei componenti.

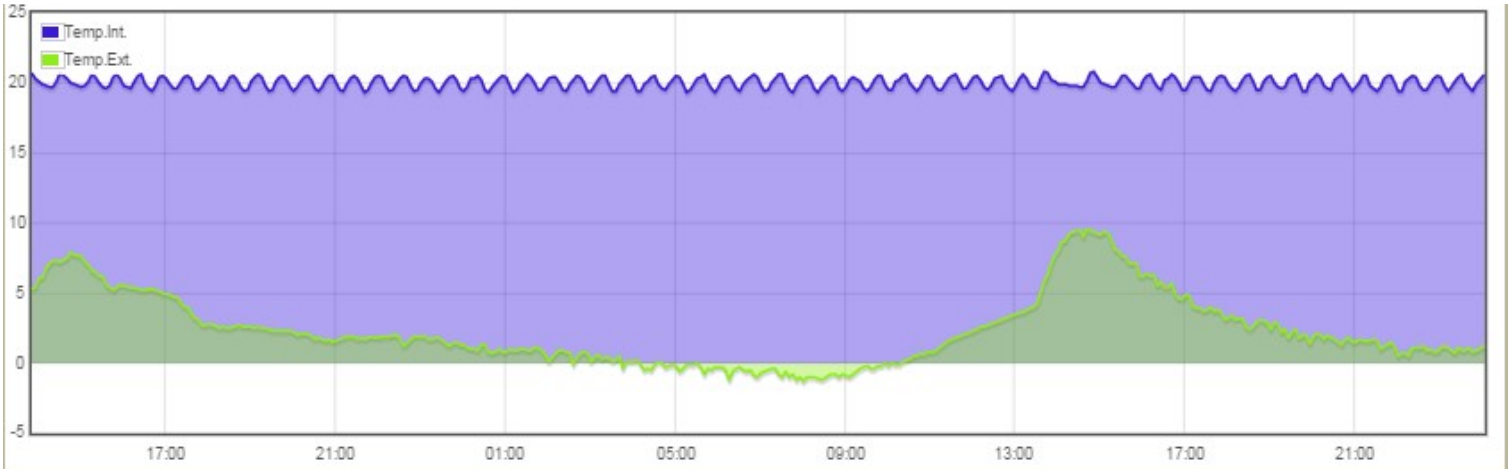


Dopo aver messo a posto tutti i collegamenti e modificato il programma con l'aggiunta delle linee di codice viste nella pagina precedente, possiamo finalmente vedere i valori di temperatura ricavati dagli LM35 sul sito emoncms.

ANALISI DEI RISULTATI

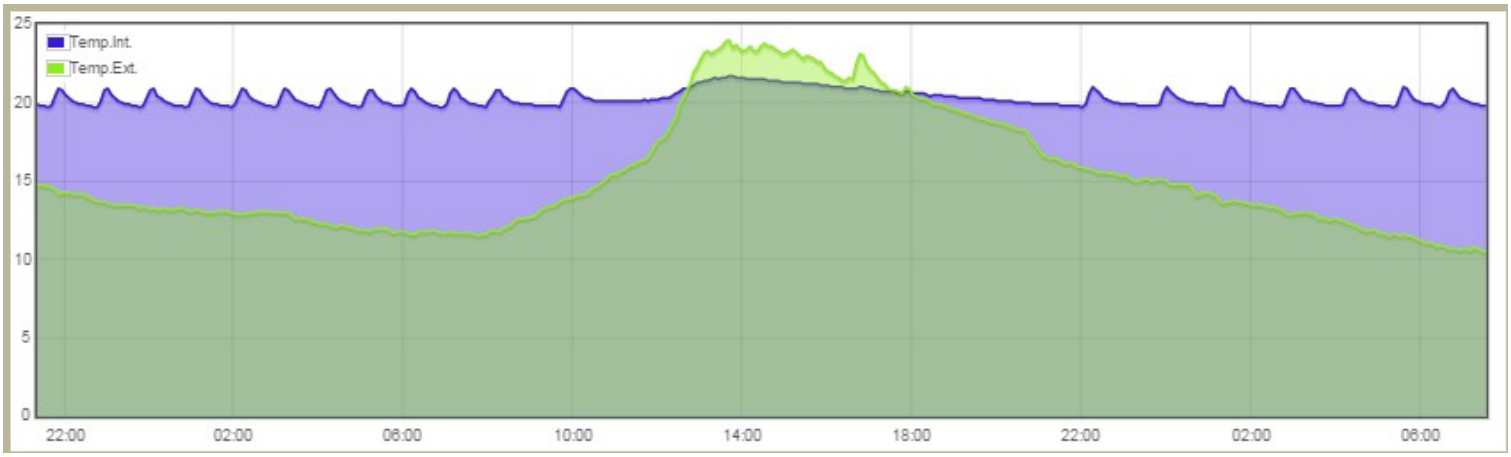
A questo punto, con i vari grafici creati sul sito Emoncms, possiamo analizzare i risultati ottenuti.

INVERNO



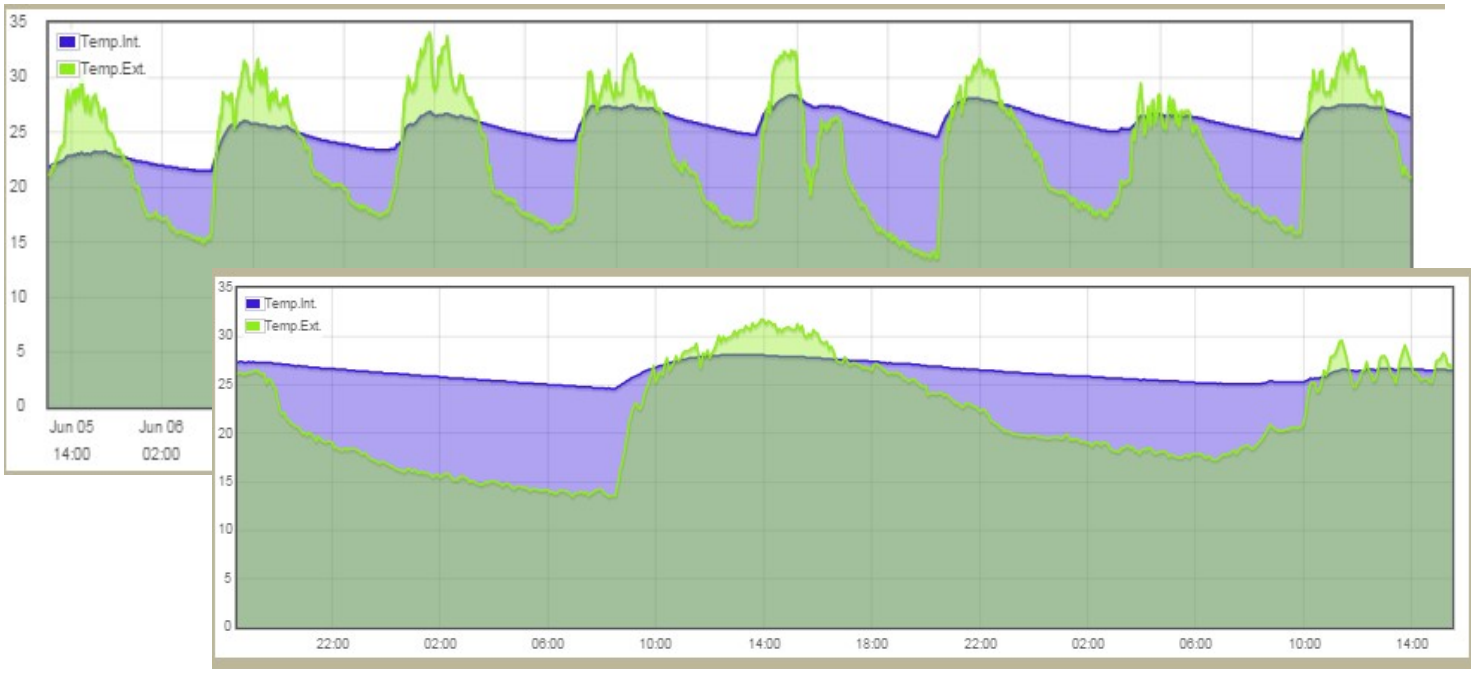
Come si può notare dal primo grafico, durante il periodo invernale la temperaturea interna passava continuamente tra i 19,80° e i 20,40° che rappresentano i limiti da noi imposti. Quindi essendo i limiti molto vicini, si può dire che la temperatura interna rimane quasi costante. Però si può notare un'oscillazione continua dovuta all'accensione e allo spegnimento della stufetta. Tuttavia il calo di temperatura era sempre continuo e notevole poiché la temperatura esterna, nell'arco di tempo da noi scelto, era sempre molto bassa (temp max.=10°C)

PRIMAVERA

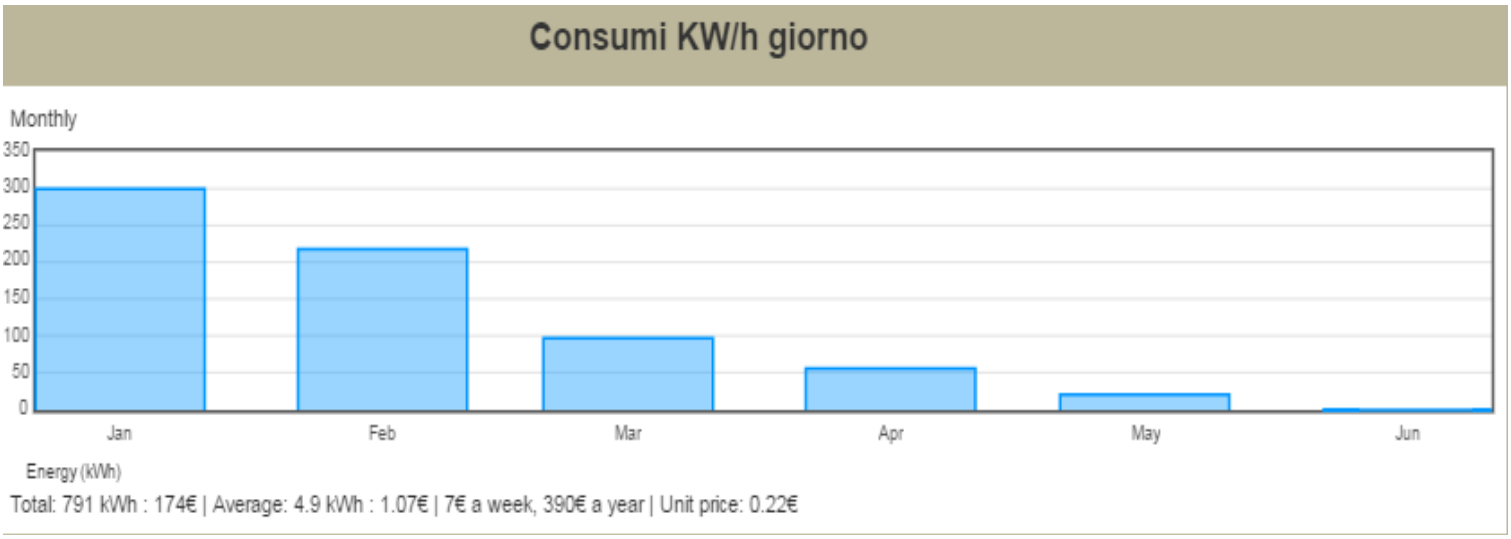


Come nel periodo invernale anche durante la Primavera la stufa si attivava, tuttavia al contrario di Gennaio, a causa della temperatura esterna più elevata, l'abbassamento di quella interna era molto più lento, e in alcuni lassi di tempo la temperatura saliva senza bisogno della stufetta (dalle 13 alle 17). Quindi si nota soprattutto una minore frequenza di accensione della stufa e un innalzamento della temperatura esterna.

ESTATE

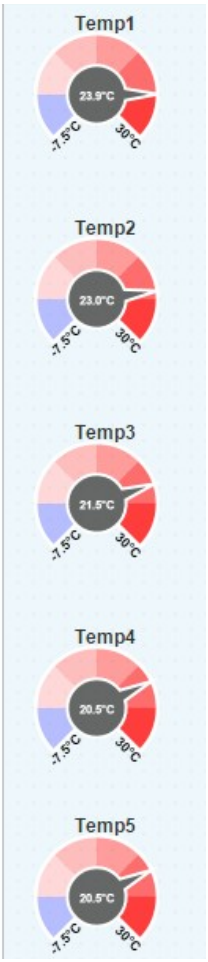


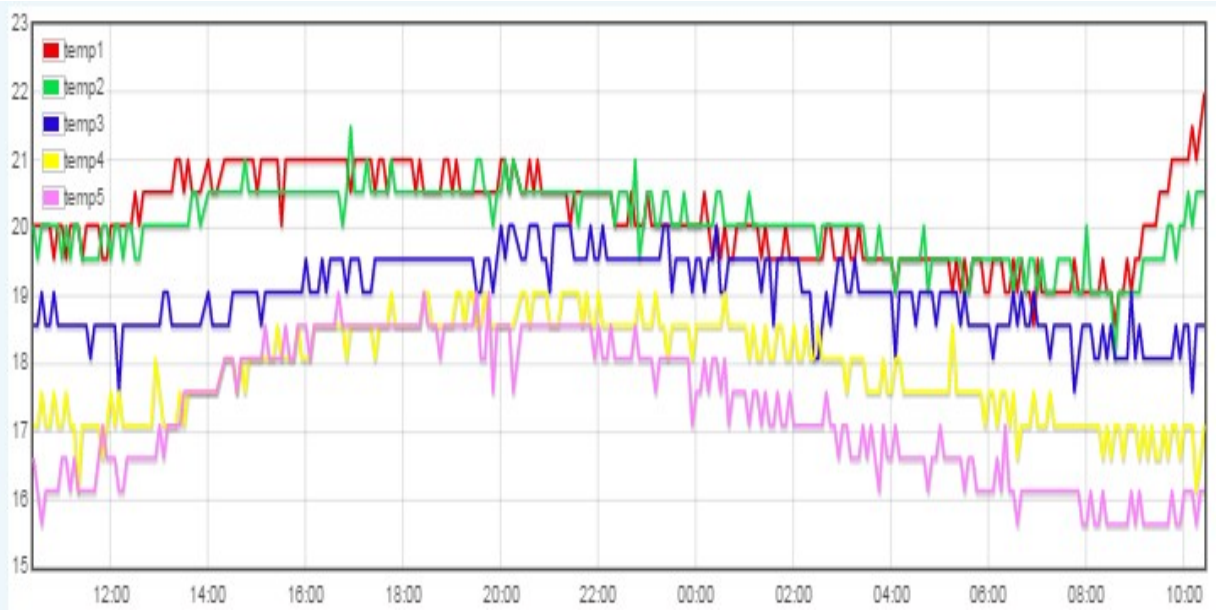
Durante il periodo estivo invece la stufa non viene praticamente mai attivata, infatti a causa dell'alta temperatura esterna, la temperatura della casa di paglia non raggiunge la soglia minima che determina l'attivazione del dispositivo. La temperatura è sempre sopra i 25° e l'andamento costante dove alle 11 inizia ad aumentare la temperatura interna per poi abbassarsi dalle 14 fino al giorno dopo è naturale e non manipolato da noi.



Come viene evidenziato dal grafico, al passare dei mesi e con l'aumento delle temperatura tra la primavera e l'estate, il consumo derivato dall'accensione della stufetta è sempre minore.

Avendo installato la stecca con i sensori LM35 più avanti, non abbiamo grafici relativi all'inverno.





Questo grafico rappresenta la distribuzione delle temperature all'interno del muro di paglia. Da qui è possibile comprendere come la paglia agisce sull'isolamento della temperatura esterna.
Dal grafico soprastante si può notare che la differenza di temperatura tra ogni sensore è di circa 1°C e quindi tra il primo sensore e l'ultimo sensore della stecca ci sono circa 4°C di differenza.

Questo grafico rappresenta l'umidità interna ed esterna misurata dai DHT22. Si può notare che l'umidità interna resta praticamente costante a circa 40%, mentre quella esterna varia continuamente. Dove sono presenti valori molto alti di umidità (da circa 80%) è molto probabile che stesse piovendo.

CONSIDERAZIONI FINALI

Il nostro progetto è stato concluso ben prima della fine della scuola ed è perfettamente funzionante sia per quanto riguarda la parte hardware sia la parte software.

Durante il progetto abbiamo riscontrato alcuni problemi soprattutto nella parte software del lavoro, ma nonostante ciò siamo riusciti a realizzare un progetto che ci ha dato delle competenze che ci potranno sicuramente servire in un futuro molto prossimo sia per quanto riguarda il mondo del lavoro sia per un utilizzo casalingo.

In più questo progetto può essere facilmente ampliato, per esempio aggiungendo un condizionatore per rinfrescare la casa durante l'estate oppure utilizzare un sensore magnetico che riveli se la porta è aperta o chiusa.

Dal punto di vista del lavoro all'interno di un gruppo abbiamo appreso quanto sia importante collaborare tutti assieme per realizzare un progetto e quanto sia importante stabilire le diverse mansioni che ogni singolo elemento del gruppo deve svolgere.