Arduino Yún Plant Datalog

Si vuole realizzare un dispositivo in grado di monitorare la crescita di una pianta, scattando foto ad intervalli prestabiliti e mostrando su un'interfaccia web la pianta in tempo reale.
Inoltre verranno mostrati in tempo reale i valori di temperatura, luminosità e umidità del sistema.

Materiale necessario:

Arduino Yún
Scheda microSD
Webcam Logitech compatibile con i driver *UVC* (**click qui per elenco dettagliato**)
Sensore di temperatura e umidità DHT-11.
Fotoresistenza da 50 KΩ
Resistenza da 50 KΩ
Breadboard
Computer collegato via wi-fi o ethernet.
Cavo *USB A - micro USB*

Presentazione Arduino Yún:

Arduino Yún è una scheda a microcontrollore basata sull'ATmega32u4 e sull'Atheros AR9331.
Il processore Atheros supporta una distribuzione Linux basata su OpenWrt, chiamata OpenWrt-Yun. La scheda possiede connettività via Ethernet e via WiFi, una porta USB-A, alloggiamento per una card micro-SD, 20 pin digitali di input/output (di cui 7 possono essere utilizzati come uscita PWM e 12 come ingresso analogico), oscillatore al quarzo f=16MHz, connessione USB, connettore ICSP, tre pulsanti di reset.

Configurazioni preliminari:

1. Accendere il computer e collegarlo ad una rete internet.
2. Scaricare i seguenti software: Arduino IDE e WinSCP (per Windows©) o Cyberduck (per Mac© OS X 10.x) e installarli.
   N.B. nel caso si voglia utilizzare Cyberduck, seguire la procedura al paragrafo corrispondente.
3. Inserire nell'apposito slot dell'Arduino la microSD
4. Alimentare la scheda via cavo USB
5. Con il computer disassociarsi dalla rete iniziale e collegarsi via wi-fi all'hotspot creato automaticamente dalla Yún.
6. Aprire un browser internet (internet explorer, mozilla firefox, google chrome, ecc…) e digitare nella barra degli indirizzi il seguente indirizzo IP:

   192.168.240.1
   oppure:
   arduino.local

7. Effettuare l'accesso con la seguente password:

   arduino

8. Una volta eseguito l'accesso, fare click su "Configure".
   Dal pannello di configurazione, nella sezione **"Wireless parameters"** selezionare il nome della propria rete wireless e, nel caso di rete protetta, inserire la password per l'accesso.
   È consigliabile cambiare anche la password di configurazione predefinita.
   Salvare le modifiche facendo click su "Configure & restart"
   N.B. nel caso non si disponga di una rete wireless si può collegare la scheda attraverso un cavo ethernet. In tal caso saltare il passo n° 5.

9. Scollegarsi dall'hotspot della scheda e ricollegarsi alla propria rete.
10. Aprire l'IDE di Arduino. Per ritrovare successivamente la propria scheda Arduino all'interno della propria LAN, selezionare l'Arduino Yún dalla lista che si può ottenere a:

Strumenti > Porta > Porte di rete

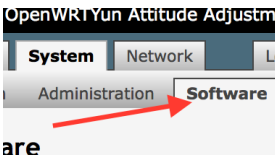
Installazione pacchetti aggiuntivi:

Una volta seguiti tutti i passi della sezione precedente, è possibile iniziare con il progetto vero e proprio.
Come presentato all'inizio, il dispositivo per prima cosa dovrà essere in grado di fare foto e inoltre di mostrare la pianta in tempo reale, per questo bisogna installare due pacchetti prima di collegare la webcam alla porta USB presente sulla scheda stessa.

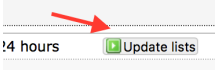
I pacchetti sono i seguenti:
kmod-uvc-video: modulo kernel per supportare dispositivi la *"USB video class"* (UVC). Può essere paragonato ad un "driver".
motion: pacchetto per la gestione della fotocamera quando qualcosa si muove davanti.

Per installare i pacchetti collegarsi con un browser all'interfaccia grafica di Arduino e, dopo aver inserito la password, clicchiamo "Configure"; quindi "advanced configuration panel".

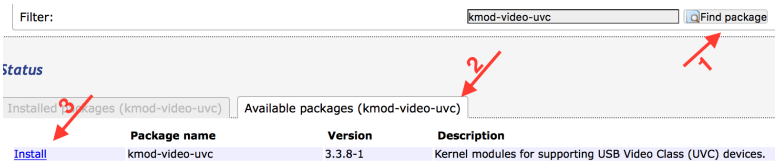
Dalla scheda "System" selezioniamo la selezioniamo "Software".



Premere il pulsante "Update lists" per aggiornare la lista dei software disponibili.



Cercare il pacchetto kmod-video-uvc, spostarsi nel tab "Available packages" e installarlo.



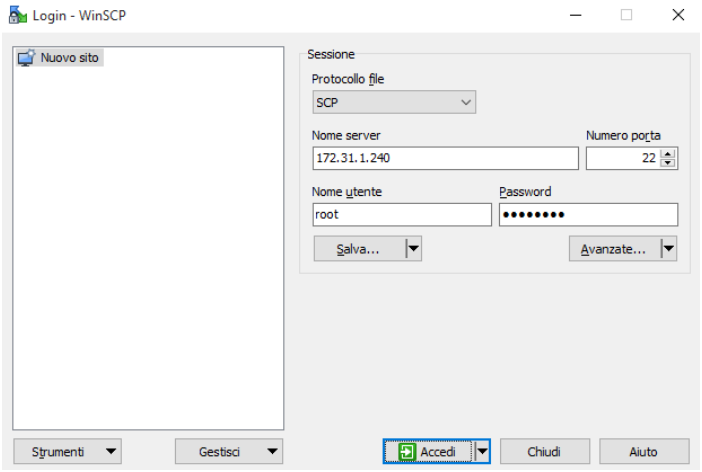
Fare la stessa cosa con motion.

WinSCP:

WinSCP è un client grafico open source per Windows, scaricabile gratuitamente da internet.
La sua funzione principale è quella di copiare file in modo sicuro tra un computer locale e uno remoto. WinSCP offre due interfacce utente: la prima, visualizzata all'inizio dell'esecuzione del programma, richiede l'inserimento di alcuni parametri per avviare il client.
La seconda interfaccia permette di comunicare con il computer remoto attraverso uno locale, rendendo possibili all'utente alcune attività base quali:

- Caricamento di file
- Download dei file
- Modifica delle proprietà dei file remoti
- Eliminazione di file
- Spostamento e duplicazione di file remoti
- Creazione di nuovi oggetti:
    - Creazione di file
    - Creazione di directory
    - Creazione di collegamenti e scorciatoie

Per questo progetto è stato scelto di utilizzare WinSCP per comunicare tra il computer locale e il server *dropbear* di Arduino Yùn.
Aprendo il file eseguibile (.exe) di WinSCP verrà chiesto chiederà di effettuare il login.
Come è possibile notare nella figura sottostante bisogna innanzitutto selezionare il tipo di protocollo. Si utilizzerà *SCP* (Secure Copy Protocol).
Nella sezione "Nome server" inserire l'indirizzo IP della propria scheda Arduino. Nell'esempio è stato assegnato alla Yún l'indirizzo 172.31.1.240, ma esso cambia ogni volta che la scheda viene collegata alla rete. Inserire come nome utente *root* e poi la password impostata all'inizio della configurazione. É necessario effettuare l'accesso come utente root perché sarà possibile godere di tutti i permessi d'amministratore potendo agire su file e directory anche alla "radice" del sistema. Inoltre, di default, linino (il sistema operativo che "gira" sull'Atheros AR9331) possiede solo un utente abilitato, l'utente root appunto.



Il terminale:

Per terminale si intende l'ambiente di lavoro che permetta di inserire dati in input ad un computer e riceverli in output per la loro visualizzazione.
Per alcune operazioni si possono impartire comandi alla Yún utilizzando l'*advanced configurazioni panel (luci)*, per altre invece è opportuno comunicare in modo più efficace ed istantaneo via terminale, utilizzando un protocollo di comunicazione.

Windows: aprendo WinSCP è già presente un client SCP, PuTTY. Esso si può aprire premendo sulla seguente icona dal pannello "Commands Buttons" :  di WinSCP.
PuTTY volendo è scaricabile separatamente, ma non è necessario
Mac: aprire "Terminale" da Launchpad. Digitare:

ssh *indirizzoIpYún* -l root

inserire la password e a questo punto si è in grado di comunicare con la scheda.

Cyber duck:

Scaricare l'ultima versione del software dal sito originale. La versione utilizzata per il seguente manuale è la 4.7.3 .
Linino di default ha Dropbear, un server SSH che consente l'utilizzo del protocollo SCP. Cyberduck nonostante possieda diversi protocolli non ha l'SCP; è possibile quindi risolvere il problema installando nell'O.S. della Yún un server SFTP digitando i seguenti comandi:

opkg update
opkg install openssh-sftp-server

La prima riga di comando permette di aggiornare le liste dei repository, la seconda di installare il server SFTP.
A questo punto avviare Cyberduck e fare click su "Nuova Connessione", inserire nel campo "Server" l'IP della scheda, nel campo "Nome utente" il nome utente cioè *root* e come "Password" la password definita dall'utente.

Il pacchetto "motion":

Motion si occupa di gestire tutta la parte video del progetto. Di default questo pacchetto permette di scattare foto ogni volta che un oggetto si muove davanti al campo d'azione della webcam.
L'algoritmo funziona nel modo seguente: viene fatta una scansione pixel per pixel e viene salvato il colore di ciascuno di essi. Al termine della scansione ne viene fatta un'altra; se il colore del px è differente al colore precedente viene incrementata una variabile contatore. Nel caso la variabile contatore sia maggiore di una soglia prestabilita, viene scattata una foto. Questa funzionalità però non è necessaria ai fini del progetto. Per disabilitarla non esiste un comando vero e proprio, quindi si ricorre al seguente espediente: imporre una soglia maggiore del numero di pixel che compongono l'immagine, in questo modo il contatore sarà sempre inferiore alla soglia e quindi non verrà mai scattata la foto.
Per modificare la soglia, via terminale spostarsi nella cartella /etc/ con il comando:

cd /etc/

una volta nella cartella digitare:

nano motion.conf

e andare alla riga seguente:

```
…
############################################################
# Motion Detection Settings:
############################################################
# Threshold for number of changed pixels in an image that
# triggers motion detection (default: 1500)
threshold 80000
```

```
…
```

Per cambiare il valore bisogna ragionare sulla dimensione del frame: andare a riga 90 e 93 e leggere i valori di *width* e *height*. Moltiplicarli fra loro. La webcam qui utilizzata è una Logitech C310 e i valori sono configurati in questo modo: 320 e 240. Il prodotto è 320*240 = 76800, il valore di *threshold* è stato modificato con 80000.

Un altro parametro da configurare è la cartella di destinazione in cui verranno salvate le foto scattate dalla webcam. Di default esse vengono salvate nella memoria NAND della scheda ma essa in breve tempo si riempirebbe di immagini e comprometterebbe il corretto funzionamento della board; vantaggioso è invece salvarle su una scheda *µ*SD. Le cartelle in cui poi verranno salvate le immagini si possono creare da terminale mediante il comando *mkdir* seguito da *nome-directory*. Per spostarsi di cartella in cartella utilizzare il comando *cd*; per visualizzare i file o le directory all'interno di una cartella si utilizza il comando *ls*.

```
…
# Target base directory for pictures and films
# Recommended to use absolute path. (Default: current working directory)
target_dir /mnt/sda1/arduino/www/webcam
…
```

Infine configurare ogni quanti secondi verrà scattata un foto. Andare alla riga:

```
…
# Snapshots (Traditional Periodic Webcam File Output)
############################################################
# Make automated snapshot every N seconds (default: 0 = disabled)
snapshot_interval 3600
…
```

Sostituire 0 al numero di secondi desiderato. Trattandosi di una pianta, una foto all'ora è più che sufficiente (= 3600 s).

Un altro elemento da tenere in considerazione è che ogni volta che si alimenta la scheda dovrà automaticamente partire il pacchetto motion, per garantire che l'utente possa visualizzare lo streaming della pianta senza dover fare altre operazioni.
Per fare ciò modificare il file *rc.local* che si trova alla directory */etc/* digitando i seguenti comandi:

```
cd /etc/
nano rc.local
```

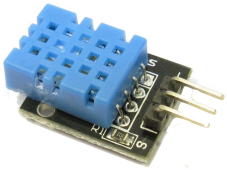
Andando a capo e sopra a *"exit 0"* aggiungere la seguente stringa:

```
motion
```

Sensore DHT-11:

Il sensore DHT-11 è un sensore sia di temperatura sia di umidità. A seconda della casa costruttrice può essere dotato di tre o quattro piedini. Esso comunica via seriale.
Ecco la funzione dei quattro pin:

1-Vcc: +5V
2-Data: pin da cui vengono letti i dati della lettura
3-null: pin non funzionante per la corrente applicazione
4-GND
Molte volte l'ordine dei piedini può mutare, seguire sempre il datasheet fornito dal produttore del sensore.



Caratteristiche:		
Range temperatura:	0°C - 50°C	± 2°C
Range umidità:	20%RH - 90%RH	± 5%RH
Tensione di alimentazione:	5V	
Corrente (running)	0.5 mA	(max 2.5 mA)
Corrente (stand-by)	100μA	(max 150μA)
Peso:	2.7 g	

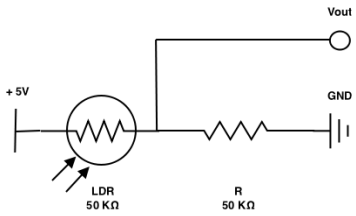
La gestione del sensore è affidata ad una libreria dedicata reperibile all' indirizzo:
http://www.dfrobot.com.cn/image/data/DFR0067/dht11.zip

Le istruzioni della libreria sono illustrate nella sezione in cui viene descritto il software per il microcontrollore 32u4.

Sensore di luminosità:

Il sensore di luminosità è stato realizzato mediante una fotoresistenza in serie ad una resistenza. Dato che non si potrà leggere un valore di luminosità preciso, espresso in lux ad esempio, si proporrà il valore in percentuale.

Il circuito del sensore di luminosità è il seguente:



Per il dimensionamento della resistenza è vantaggioso scegliere un valore ohmico uguale a quello della fotoresistenza in presenza di buio, in modo che il valore letto dall'adc (*analog to digital converter*) in uscita dal partitore di tensione sia al massimo metà di Vcc (legge del partitore di tensione).

Descrizione software microcontrollore:

Il software è stato scritto nell'ambiente di sviluppo di Arduino, Arduino IDE.
Questo programma mette in comunicazione il microcontrollore 32U4 con quello su cui gira Linux, inviando i dati letti dai sensori DHT-11 (temperatura e umidità) e di luce (fotoresistenza con resistenza in serie).
La libreria per gestire il DHT-11 è scaricabile da questo sito.
Ecco il codice:

```
#include <Bridge.h>
#include <YunServer.h>
#include <YunClient.h>
#include <dht11.h>

#define DHT11_SENS 7            // 32U4 pin where is connected dht-11 sensor

YunServer server;               // object variable creation
```

```
dht11 DHT;                        // object variable creation

int UmidityRead(void);            // dht-11 umidity read function prototype

void setup() {
  Bridge.begin();                 // starts and facilitates communication between 32U4 and
  server.listenOnLocalhost();     // doesn't listen request on 5555 port, not coming from uHttpd
  server.begin();                 // tells the server to begin listening for incoming connections
  Serial.begin(9600);             // serial enabled for debug mode
  Serial.println("Bridge, server and serial started");
}

void loop() {
  YunClient client = server.accept();
  if (client) {
    DatasSend(client);
    client.stop();                // disconnects client from the server
  }
  delay(50);
}

void DatasSend(YunClient client) {
  static float value;

  value = TemperatureRead();
  client.print("temperature");
  client.print("=");
  client.print(value);
  client.print("#");
  Serial.print("Temperature: ");  // prints temperature value for debug mode
  Serial.print(value);
  Serial.print(" \260");
  Serial.println("C");

  value = analogRead(A0)/5.12;
  client.print("light");
  client.print("=");
  client.print(value);
  client.print("#");
  Serial.print("Light: ");        // prints light value for debug mode
  Serial.println(value);

  value = DHT.humidity;
  client.print("humidity");
  client.print("=");
  client.print(value);
  Serial.print("Humidity: ");
  Serial.println(value);     // prints humidity value for debugging mode
  client.println("");
}

int TemperatureRead(void)
{
  static int debug_dht;
  debug_dht = DHT.read(DHT11_SENS);   // reads dht-11 status
  switch(debug_dht)
  {
    case DHTLIB_OK:                   // if the read returns 0, read OK
                Serial.print("OK,\t");
                break;
    case DHTLIB_ERROR_CHECKSUM:       // if the read returns -1, checksum error
                Serial.print("Checksum error,\t");
                break;
    case DHTLIB_ERROR_TIMEOUT:        // if the read returns -2, read timeouted
                Serial.print("Time out error,\t");
                break;
    default:                          // if the read returns other values unknown error
                Serial.print("Unknown error,\t");
                break;
  }
  return DHT.temperature;
}
```

Lista delle funzioni utilizzate:

- Bridge.begin();
  Serve a mettere in comunicazione il processore di Arduino con quello di Linux.
  Nessun parametro e nessun ritorno.

- server.listenOnLocalhost();
  Indica al server di iniziare l'ascolto per connessioni in entrata. Nessun parametro e nessun ritorno.

- server.begin();
  Inizializza il server per la comunicazione. Nessun parametro e nessun ritorno.

- Serial.begin(9600);
  É stata inserita per inizializzare la comunicazione via monitor seriale e quindi effettuare un rapido debug di ciò che i sensori leggono. Parametro: velocità di comunicazione 9600 bit/s. Nessun ritorno.

- YunClient client = server.accept();
  Si apre un client per la gestione del collegamento. Se il client si connette correttamente può elaborare le richieste della funzione personalizzata (descritta sotto). Nessun parametro, ritorna 1 se la connessione è andata a buon fine, 0 in caso contrario.

- DataSend(client);
  Questa è una funzione realizzata ad hoc che contiene l'invio dei dati letti dai sensori sia al client che via monitor seriale. Come parametro ha lo stato del client in comunicazione creato precedentemente, chiamato appunto *client*. Nessun ritorno.

- value = TemperatureRead();
  Assegna alla variabile *value* il valore di temperatura letto dal sensore DHT-11. Questo sensore permette di misurare la temperatura e l'umidità dell'ambiente. Nessun parametro, ritorna il valore della temperatura rilevata.

- debug_dht = DHT.read(DHT11_SENS);
  Legge dalla sonda lo stato della stessa e memorizza il valore nella variabile debug_dht. Come parametro accetta solo il piedino a cui essa è collegata.

- switch(debug_dht)
  Lo switch case verifica il valore restituito dalla sonda prima di chiederle informazioni su umidità e temperatura. A seconda del valore che debug_dht assume si può identificare il tipo di errore o se la lettura è avvenuta correttamente.

- DHT.temperature;
  Ritorna il valore di temperatura rilevata dal DHT-11;

- client.print(dato);
  Invia al client il parametro passato, che può essere un dato di qualsiasi tipo (int, float, double….) oppure una stringa se scritta costante, se opportunamente scritta fra virgolette (esempio: *client.print("Ciao")*.
  Nel programma è presente *client.print("#");* al fine di realizzare un carattere separatore fra i tre tipi di dati da mandare.
  Parametro: dato da inviare, ritorno: nessuno.

- value = DHT.humidity;
  Legge il valore di umidità rilevata e lo associa alla variabile *value*.

- value = analogRead(A0)/5.12;
  Il convertitore analogico digitale di Arduino effettua la conversione di un segnale che può variare tra da 0 a 5 V con una risoluzione a 10 bit.
  Sul piedino analogico 0 è collegata l'uscita di un partitore di tensione costituito da

una fotresistenza e in serie una resistenza di egual valore massimo. Nel caso di buio totale, $R_{LDR} = R_{serie}$ perciò $V_{OUT} = Vcc / 2$. Il valore ritornato dall' *adc* potrà essere al massimo alla metà della risoluzione massima $2^{10} - 1$, quindi 512.
Per proporre all'utente il valore espresso in percentuale, il valore letto dall'adc, con tutte le semplificazioni finali, verrà esclusivamente diviso per 5.12 .

Descrizione software pagina internet:

La pagina internet è stata scritta utilizzando AJAX, acronimo di Asynchronous JavaScript and XML, cioè utilizzando una tecnica di sviluppo software per la realizzazione di applicazioni web interattive. Lo sviluppo di applicazioni HTML con AJAX si basa su uno scambio di dati "dietro le quinte" fra browser e server web che consente l'aggiornamento dinamico di una pagina web senza richiesta di una "ricarica" specifica da parte dell'utente. AJAX è asincrono nel senso che i dati sono richiesti al server e caricati senza interferire con il comportamento della pagina esistente.

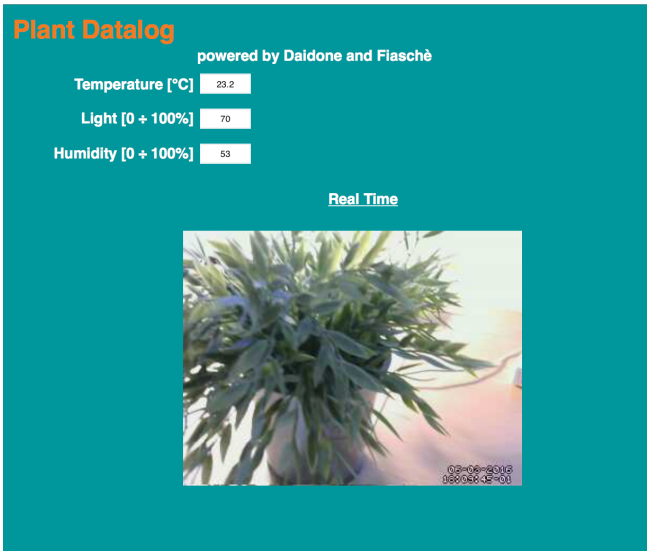
```
<html>
  <head>
    <title>Arduino Plant Datalog</title>                                    <!--Page's title-->
    <link rel="Shortcut Icon" href="http://172.31.1.240/sd/favicon.ico">
    <body bgcolor="#00979c" text="#ffffff">                                 <!--Arduino background color-->

    <script type="text/javascript">

              window.onload=status;         // update sensors status

              function status()
              {
                        setTimeout(status, 2000);
                        server = "/arduino/status/99";
                        request = new XMLHttpRequest();
                        request.onreadystatechange = updateasyncstatus;
                        request.open("GET", server, true);
                        request.send(null);
              }

              function updateasyncstatus()
              {
                        if ((request.readyState == 4) && (request.status == 200))
                        {
                                  result = request.responseText;
                                  labelsandvalues = result.split("#");

                                  for(i = 0; i < labelsandvalues.length; i++)
                                  {
                                            PinPair = labelsandvalues[i];
                                            singlelabelandvalue = PinPair.split("=");
                                            label = singlelabelandvalue[0];
                                            valueread = singlelabelandvalue[1];

                                            if(label.substr(0,1) == "t")
                                            {
                                                      temperaturevalue = parseFloat(valueread);
                                                      document.getElementById("temperature").value = temperaturevalue;
                                            }

                                            if(label.substr(0,1) == "l")
                                            {
                                                      lightvalue = parseFloat(valueread);
                                                      document.getElementById("light").value = lightvalue;
                                            }

                                            if(label.substr(0,1) == "h")
                                            {
                                                      humidityvalue = parseFloat(valueread);
                                                      document.getElementById("humidity").value = humidityvalue;
                                            }


                                  }
                        }
              }

    </script>

    <style>
    tr.prova {
     font-family:sans-serif;
     font-size:14px;
     border: 0px solid black;
    }
    </style>

    <style>
```

```
    table.tablestyle {
     border: 0px solid black;
    }
    </style>
  </head>

  <!--body-->
  <font face="sans-serif">

  <table name="Titolo" class="tablestyle" style="width: 400px; height: 50px;" hspace="3" cellpadding="0">
    <tr class="prova">
              <th align="left">
              <font size="5" color="#F47A20">
              <b>Plant Datalog</b>
              </font></th>
    </tr>
    <tr class="prova"> <th align="right">powered by Daidone and Fiasch&egrave;</th>
    </tr>


  </table>

  <table name="Dati" class="tablestyle" style="width: 250px; height: 100px;">

    <tr class="prova">
              <td>
              <th align="right">Temperature [&deg;C]</th>
              </td>

              <td align="left">
                        <input type="text" style="text-align: center;"
                         name="temperature" id="temperature" value="0" size="6" readonly/>

              </td>

    </tr>

    <tr class="prova">
              <td>
              <th align="right">Light [0 &divide; 100%]</th>
              </td>

              <td align="left">
                        <input type="text" style="text-align: center;"
                         name="light" id="light" value="0" size="6" readonly/>

              </td>
    </tr>

    <tr class="prova">
              <td>
              <th align="right">Humidity [0 &divide; 100%]</th>
              </td>

              <td align="left">
                        <input type="text" style="text-align: center;"
                         name="humidity" id="humidity" value="0" size="6" readonly/>

              </td>

    </tr>

  </table>

  <table name="Streaming1" class="tablestyle" style="width: 652px; height: 50px;">
    <tr class="prova">
              <td>
              <th align="center"><u>Real Time</u></th>
              </td>
    </tr>
  </table>

  <table name="Streaming2" class="tablestyle" style="width: 652px; height: 250px;">


    <tr class="prova">

              <td rowspan="2" align="center">
              <iframe style="width: 320; height: 240;" hspace="0" vspace="0"
               marginheight="0" matginwidth="0" frameBorder="0" src='http://172.31.1.240:8081'> <!--Webcam streaming-->
              </iframe>
              </td>

    </tr>

  </font>
  <!--/body-->
</html>
```

Se il codice verrà salvato con nome ed estensione *index.html* nella cartella della sd …/ www/arduino/ basterà mettere nella barra indirizzi del proprio browser il seguente indirizzo:

http://*indirizzoIpYún*/sd/

e verrà prodotta la pagina html in figura:



La pagina .html è costituita da due parti fondamentali:

1. Parte che descrive l'aspetto grafico delle informazioni
2. Parte racchiusa fra i tag *<script type="text/javascript">* e *</script>*

La sezione che descrive l'aspetto grafico può essere riassunta nel seguente modo: vengono create due tabelle, una che organizza testo, dati e campi di dati ed un'altra che gestisce la sistemazione grafica dello streaming-frame della pianta.
Prestare attenzione che ai campi che conterranno i dati è stato assegnato un id, ciò verrà comodo successivamente per fare l'aggiornamento di quel valore, proprio mediante l'id.

Funzionamento del codice scritto in javascript:
Ogni due secondi viene richiesto al client l'invio della stringa; essa viene letta e salvata.
La stringa inviata dal client sarà espressa in questa forma:

temperature=value#light=value#humidity=value

La stringa ora viene separata in tre parti utilizzando come carattere di separazione il cancelletto *#*, e ciascuna di esse viene salvata in un vettore.
Successivamente per ogni sottostringa, la parte letterale e quella numerica vengono separate, utilizzando come carattere di separazione l'uguale *=* .
Una volta fatte queste divisioni è effettuato un confronto tra la parte letterale e le lettere *t*, *l* e *h*.
Se, ad esempio, la prima lettera della parola è una *t*, sicuramente il valore che è stato salvato all'indirizzo immediatamente successivo dello stesso vettore si riferirà alla temperatura. Verrà quindi cambiato, nella tabella, il valore numerico apparente al campo individuato dal corrispondente id (in questo caso id=temperature).
Il meccanismo si ripete con luce ed umidità.