TESINA 2K15.06.

Autore: Davide Guastella.

Classe: 5° B. I.T.I.S

Indirizzo: Elettronica ed Elettrotecnica.

TECNOLOGIA & PROGETTAZIONE DEI SISTEMI ELETTRICI ED ELETTRONICI (TPSEE)

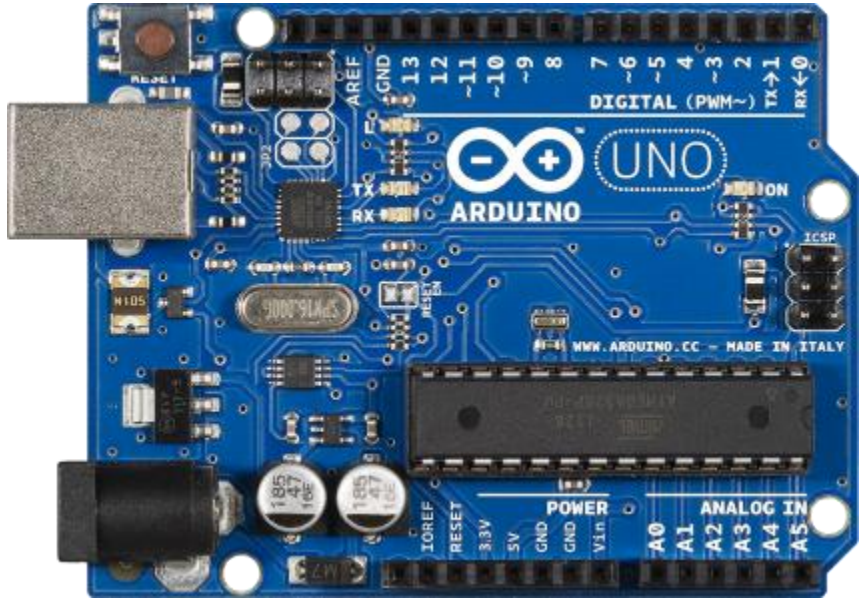
2

Arduino uno, scheda ethernet e rilevamento temperatura e luminosità

Alunno : Guastella Davide

ARDUINO UNO

Arduino è una piattaforma di sviluppo open-source basata su una semplice scheda I / O con un μC . Arduino può essere usato per sviluppare oggetti interattivi o può essere interfacciato a un PC per la sua programmazione tramite un normalissimo cavetto USB e il microcontrollore sulla scheda è programmato utilizzando un linguaggio adatto anche ai principianti.



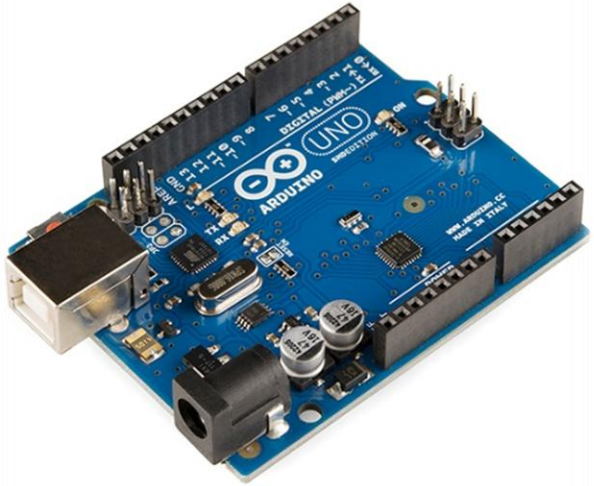
IL MICROPROCESSORE CONTIENE

- Una ALU (Arithmetic Logic Unit) dove avvengono i calcoli.
- Diversi registri per la memorizzazione temporanea dei dati e la gestione delle istruzioni.
- Un bus interno ad alta velocità.
- Circuiti di controllo e di temporizzazione per coordinare tutte le attività.
- Tre bus (Data Bus, Address Bus, Control Bus) grazie ad essi comunica col mondo esterno (dispositivi di memoria, dispositivi di ingresso uscita).

ARDUINO UNO

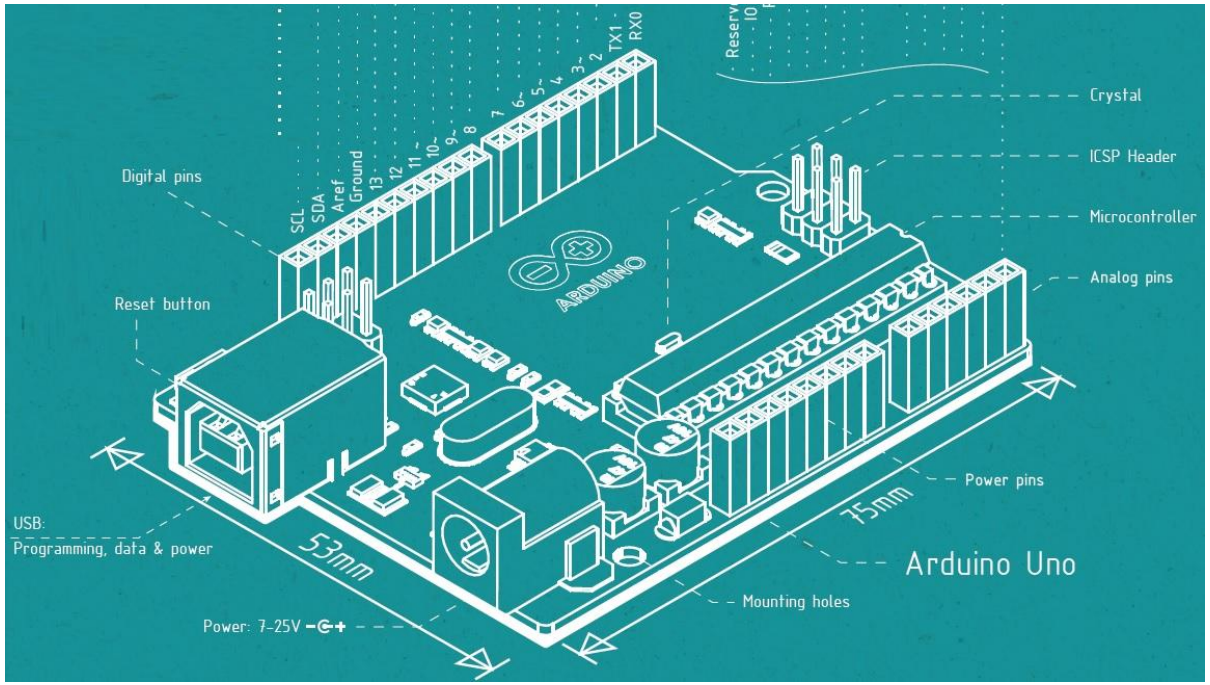
Può essere programmato attraverso un linguaggio del tutto simile al C basato sul www.processing.org language.

Una volta creato il codice lo si scarica sul μC della scheda che viene vista come una periferica dalla quale acquisire informazioni sulla porta seriale virtuale che il driver della scheda installa automaticamente. Scritto il programma Arduino potrà operare anche autonomamente eseguendo le istruzioni inserite al suo interno.



ARDUINO UNO OVERVIEW

The Arduino Uno is a microcontroller board based on the ATmega328 . It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.



ARDUINO UNO SUMMARY

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

ARDUINO UNO POWER

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector. The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

1. VIN. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
2. 5V. This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
3. 3.3V. supply generated by the on-board regulator. Maximum current draw is 50 mA.
4. GND. Ground pins.
5. IOREF. This pin on the Arduino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

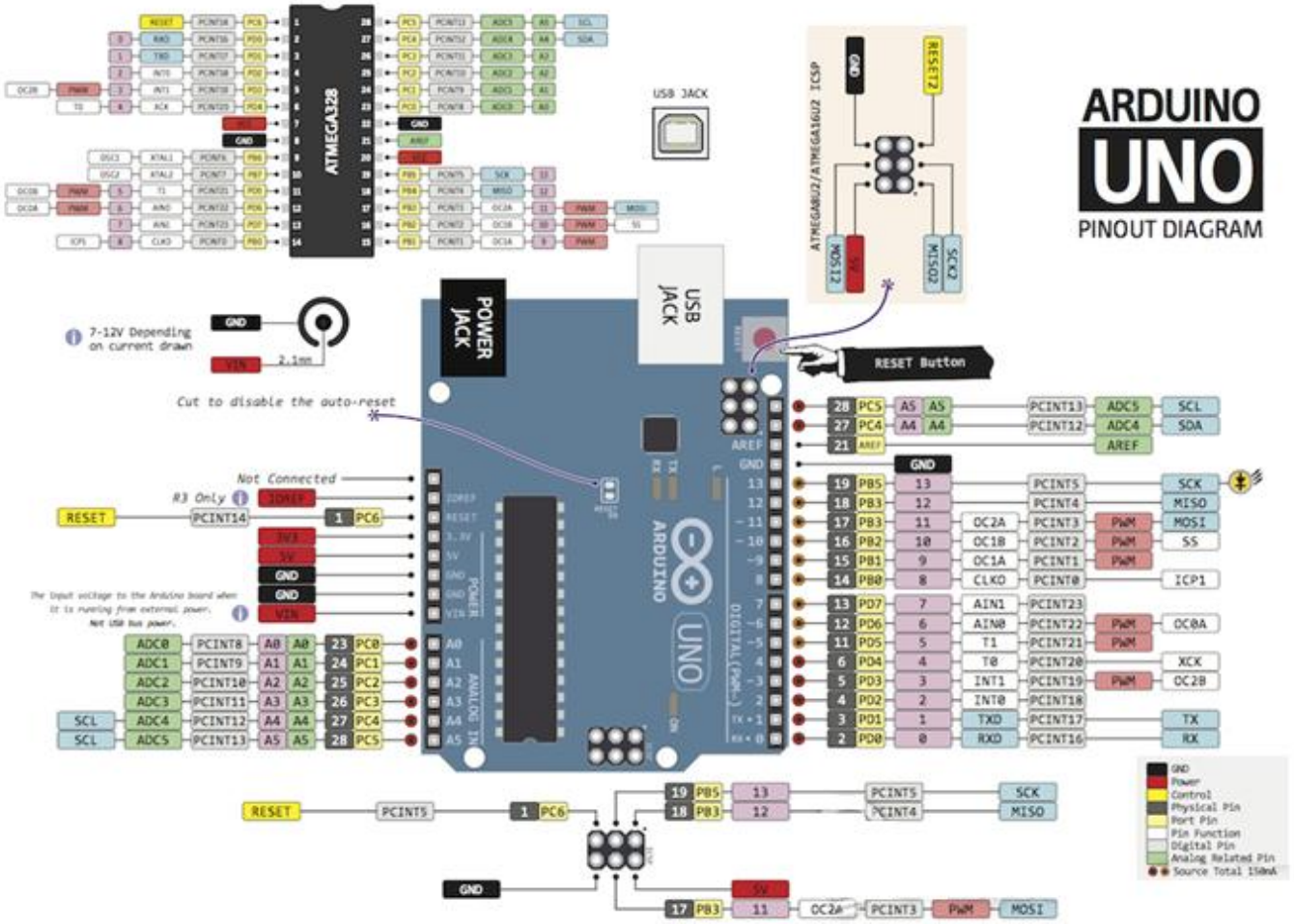
ARDUINO UNO INPUT AND OUTPUT

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

1. Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
2. External Interrupts: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt( ) function for details.
3. PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite ( ) function.
4. SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication using the SPI library.
5. LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

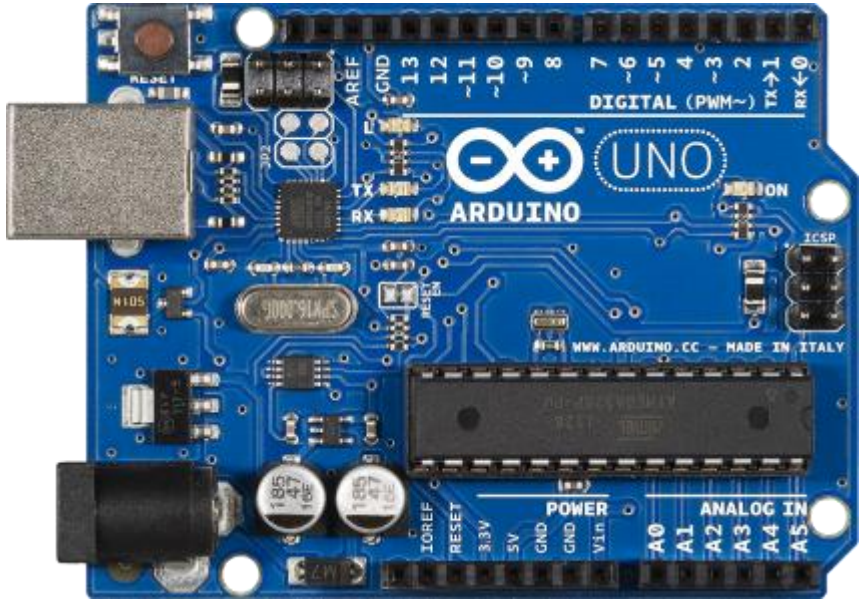
1. TWI: A4 or SDA pin and A5 or SCL pin. Support TWI communication using the Wire library.
2. There are a couple of other pins on the board:
3. AREF. Reference voltage for the analog inputs. Used with analogReference( ).
4. Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.


ARDUINO
UNO
PINOUT DIAGRAM
ATMEGA328
USB JACK
POWER JACK
RESET Button
7-12V Depending on current drawn
Cut to disable the auto-reset
Not Connected
R3 Only
RESET
ARDUINO
UNO
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
Source Total 150mA

ARDUINO UNO

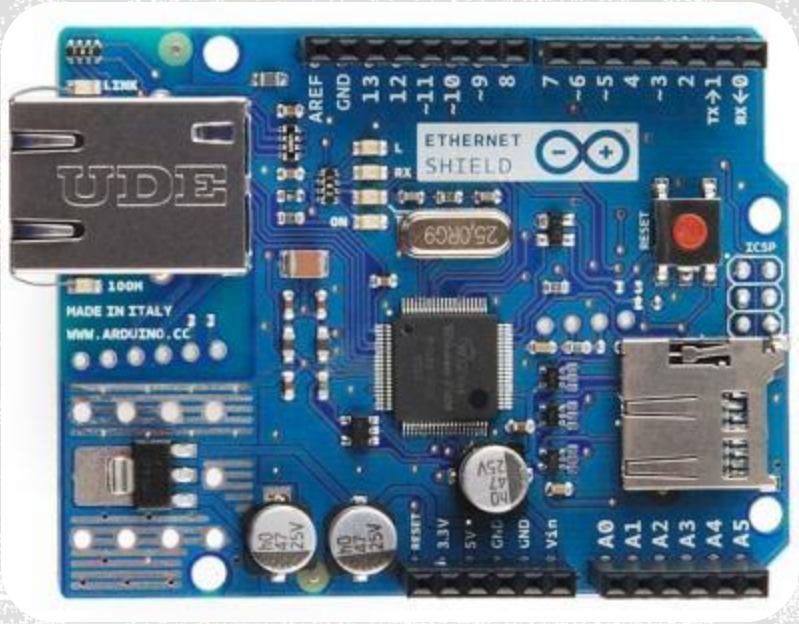
ARDUINO UNO

I programmi di Arduino si chiamano sketch o bozzetti. Collegando gli opportuni sensori , Arduino disporrà di tatto, udito, olfatto e vista ai quali si potrà aggiungere l'orientamento, il calcolo delle distanze , la vista agli infrarossi e altro ancora (giroscopi e accelerometri). Se un progetto richiede molti collegamenti che superano le porte di I/O disponibili in una sola scheda, allora si potrà intraprendere la strada a più schede Arduino UNO che colloquiano tra di loro e si dividono i compiti. Arduino dispone di soli 32KB e in questo spazio dovrà starci tutto il nostro programma.



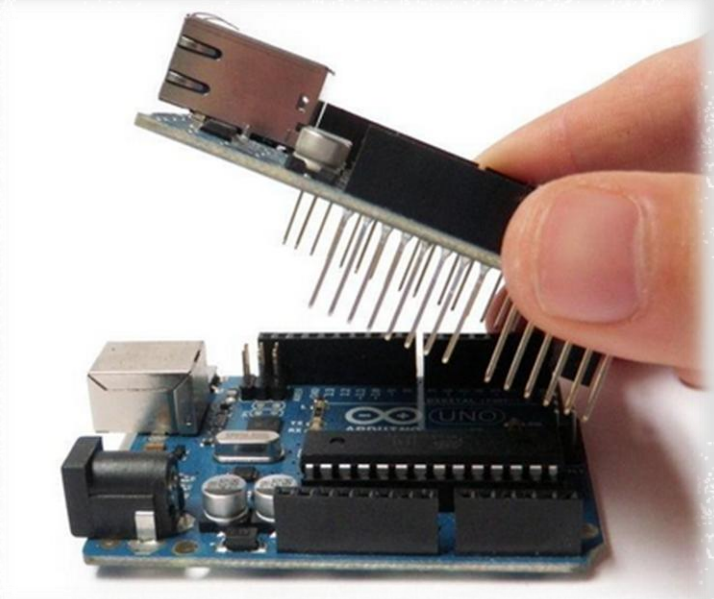
LA ETHERNET SHIELD

Una tra le più interessanti shield è la Ethernet Shield, una scheda che si innesta direttamente sulla principale e permette di collegare Arduino ad una porta Ethernet, in modo da poterlo controllare via rete locale o via Internet. La Ethernet Shield è provvista di un connettore RJ45 per la connessione diretta al modem di casa ed inoltre offre la possibilità di utilizzare uno slot microSD per lo scambio diretto di dati con un supporto di memorizzazione dedicato. La Ethernet Shield è dotata di un chip, il **W5100** che non svolge solamente il ruolo di controller Ethernet, bensì ci permette anche di utilizzare la libreria Ethernet che ad oggi rimane la più intuitiva tra le librerie utilizzabili con i moduli Ethernet. Wiznet W5100, che è in grado di gestire funzionalità di rete IP con il protocollo TCP , è anche in grado di gestire fino 4 connessioni simultanee.



LA ETHERNET SHIELD

La scheda si poggia sull'interfaccia SPI sui pin 10,11,12 e 13, con il pin 10 che serve da Slave Select, ovvero da segnale per selezionare la comunicazione con il chip Wiznet. Sulla scheda si trova anche una interfaccia Secure Digital che condivide i medesimi pin di comunicazione , ma ha sul pin 4 il segnale di Slave Select. Attraverso le opportune librerie si può quindi comunicare sia con il chip Ethernet, sia con la scheda di memoria, ma non contemporaneamente e dovrà essere lo sketch a evitare accessi sovrapposti. Ovviamente la scheda dispone di un indirizzo IP e di un MAC.



Importando la libreria ecco cosa viene inserito nello sketch:

#include <Dhcp.h>

#include <Dns.h>

#include <Ethernet.h> ; è la libreria principale

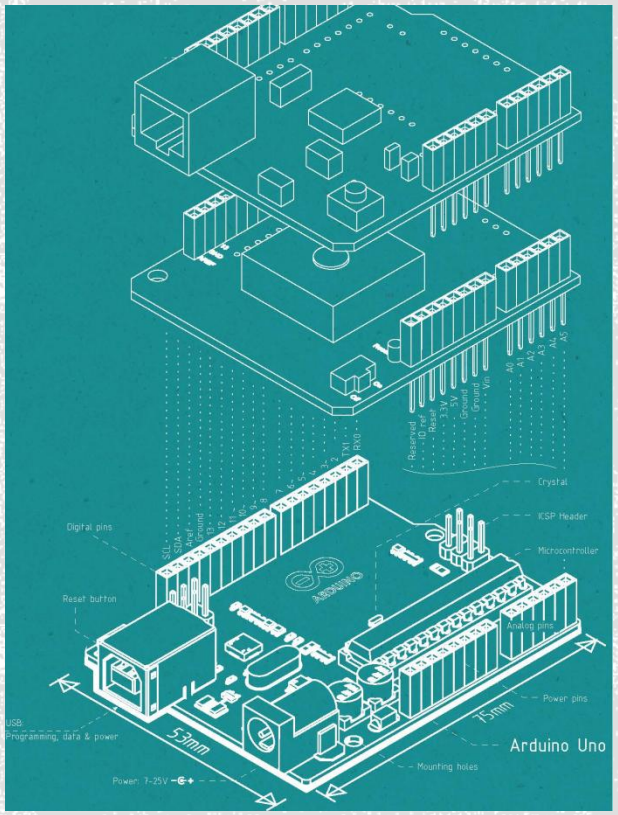
#include <EthernetClient.h>

#include <EthernetServer.h>

#include <EthernetUdp.h>

Il DHCP riguarda l'acquisizione di un indirizzo IP da un DHCP server,ovvero un server connesso alla rete e preposto all'assegnazione di indirizzi IP ai client che ne fanno richiesta . L'indirizzo può essere in uno spazio di indirizzi privato, ad es. del 192.168.x.x, se assegnato da un Router o da un proxy/firewall, oppure un indirizzo pubblico. Il DHCP oltre a comunicare l'indirizzo di rete assegnato al dispositivo, definisce anche il tempo di "lease", ovvero di affitto di quello specifico indirizzo al dispositivo. Trascorso il termine definito, quell'indirizzo sarà nuovamente assegnabile a qualche altro dispositivo. Chiaramente per tutto il tempo di "lease", l'indirizzo non cambia. DNS permette di interagire con i server in rete che trasformano gli host name in indirizzi IP

LA ETHERNET SHIELD



LA ETHERNET SHIELD

Ethernet è la libreria principale dalla quale dipendono tutte le altre.

Ethernet Client è la libreria grazie alla quale Arduino può collegarsi a server in rete facendo interrogazioni e leggendo dati.

Ethernet Server contiene le funzioni per realizzare server su una qualsiasi porta TCP/IP.

Ethernet UDP implementa le funzioni per la comunicazione con questo protocollo di basso livello.

Util ha il compito di gestire il formato e le conversioni da e per l'esadecimale.

Quando si inizializza il mac della scheda con la funzione viene anche inizializzato l'hardware e la scheda acquisisce l'indirizzo IP dal DHCP server se è stato specificato solo l'indirizzo della scheda con la funzione

Ethernet.begin(mac); // è l'indirizzo ottenuto al termine dell'inizializzazione si può leggere con la funzione

Ethernet.localIP(); //Se si specifica oltre al MAC address anche un indirizzo IP, un gateway e una maschera di sottorete, la scheda non utilizza il meccanismo DHCP ma si presenta in rete con i valori definiti dalla funzione:

Ethernet.begin(mac,ip,gateway,subnet);

MAC ADDRESS

Quando si inizializza il mac della scheda con la funzione viene anche inizializzato l'hardware e la scheda acquisisce l'indirizzo IP dal DHCP server se è stato specificato solo l'indirizzo della scheda con la funzione

Ethernet.begin(mac); // è l'indirizzo ottenuto al termine dell'inizializzazione si può leggere con la funzione

Ethernet.localIP(); //Se si specifica oltre al MAC address anche un indirizzo IP, un gateway e una maschera di sottorete, la scheda non utilizza il meccanismo DHCP ma si presenta in rete con i valori definiti dalla funzione:

Ethernet.begin(mac,ip,gateway,subnet);

IL MAC (MEDIA ACCESS CONTROL)

È un indirizzo di sei byte definito per tutti i dispositivi in fase di produzione che qualifica ciascun device in modo univoco. Nella scheda Ethernet ReV.3 l'indirizzo MAC si trova su un adesivo applicato nella parte inferiore della scheda, mentre se la scheda non riporta nulla, si può assegnare un indirizzo del tipo 90-A2-DA-xx-xx-xx. Si può recuperare l'indirizzo MAC di una vecchia scheda Ethernet non più in uso ed essere sicuri così di non avere un indirizzo utilizzato da un altro dispositivo in rete. Dopo la fase di configurazione e inizializzazione il sistema sarà pronto ad eseguire le funzioni della libreria per far funzionare Arduino da Server, Client o comunicazione e bidirezionale **UDP** (User Datagram Protocol) .

ARDUINO WEB SERVER

Quando un dispositivo deve poter rispondere a delle interrogazioni relative a dati raccolti o rilevati tramite sensori, assume il tipico ruolo da server. Praticamente mentre rileva dei dati, resta in attesa di interrogazioni su una porta e risponde inviando dati. Il tutto inserito in pagine HTML più o meno complesse memorizzate in memorie SD o micro SD da inviare ai client connessi. Questo è possibile tramite la classe "**EthernetServer()**" e le seguenti funzioni:

EthernetServer(porta) classe utilizzata per creare un server sulla porta specificata;

Esempio:

EthernetServer MioArduino = Server(80); //crea un server dinome MioArduino sulla porta 80

MioArduino.begin(); // il software inizia ad ascoltare sulla porta 80 per il collegamento di un client;

Arduino come web server

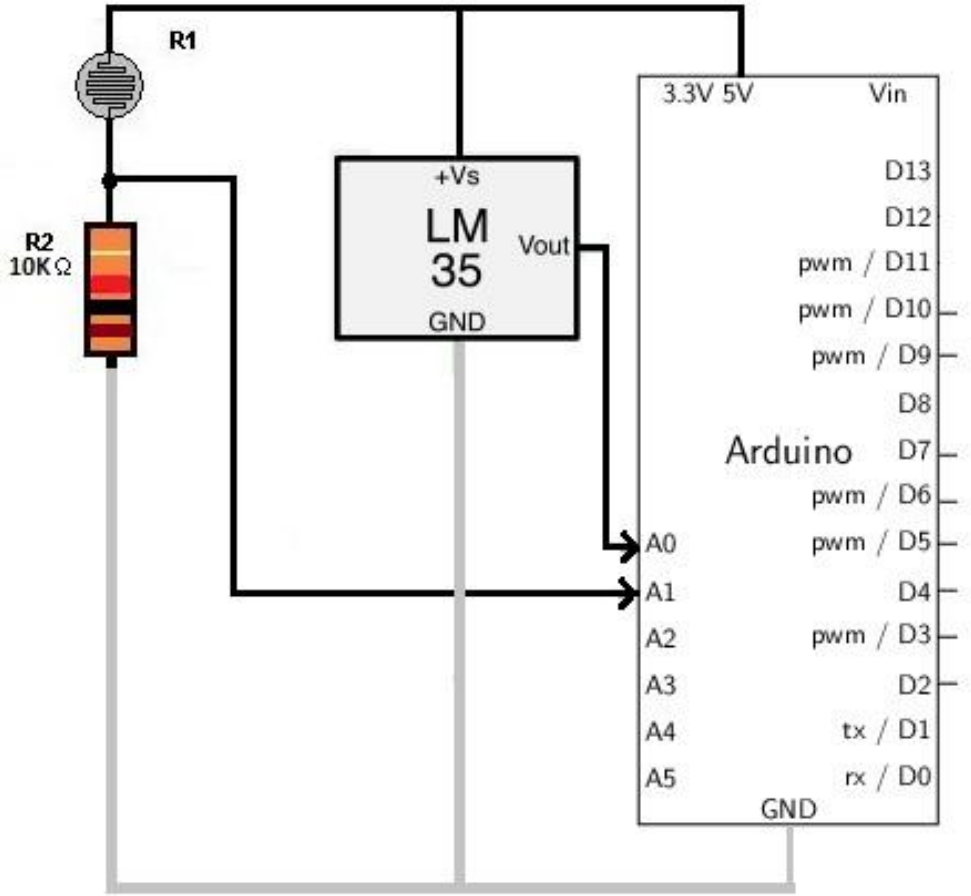
begin() //attiva il server in termini di monitoraggio della porta definita come attiva per collegamenti in entrata da parte di client

available() // la funzione restituisce un oggetto client con dei dati leggibili,altrimenti un valore false

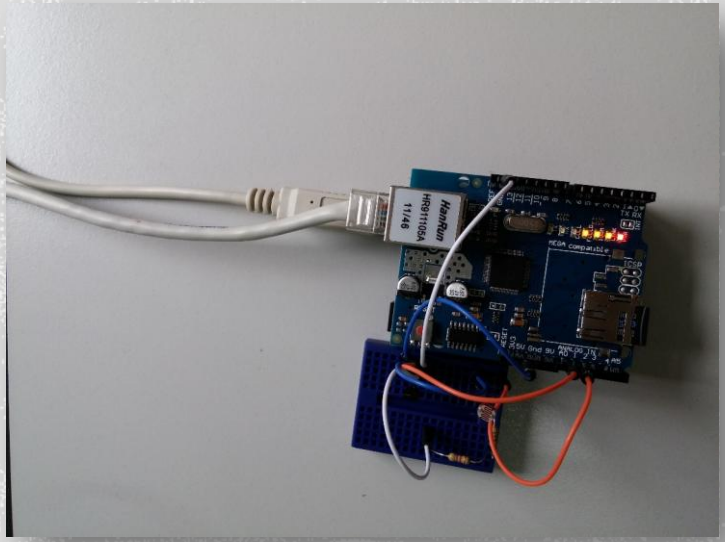
write(dato) // scrive i dati forniti come argomento (byte o char) a tutti i client connessi al server

print(dati, base) // la funzione trasferisce ai client connessi dei dati (numeri o stringhe). Come presentare il numero convertito in caratteri dipende anche dal parametro "base" (BIN,DEC,HEX,OCT). La funzione restituisce il numero di byte inviati al client.

println(dati, base) // aggiunge un ritorno a capo (CR LF) al termine del trasferimento dei dati.



PROGETTO



Servendoci di una foto-resistenza (da 10K) e di un sensore di temperatura (LM35) si vuole monitorare una piccola serra, scoprendo via internet se c'è abbastanza luce e se la temperatura è giusta. Si vogliono poi memorizzare i valori minimi e massimi di entrambe le grandezze per capire se ad esempio sono state superate certe soglie.

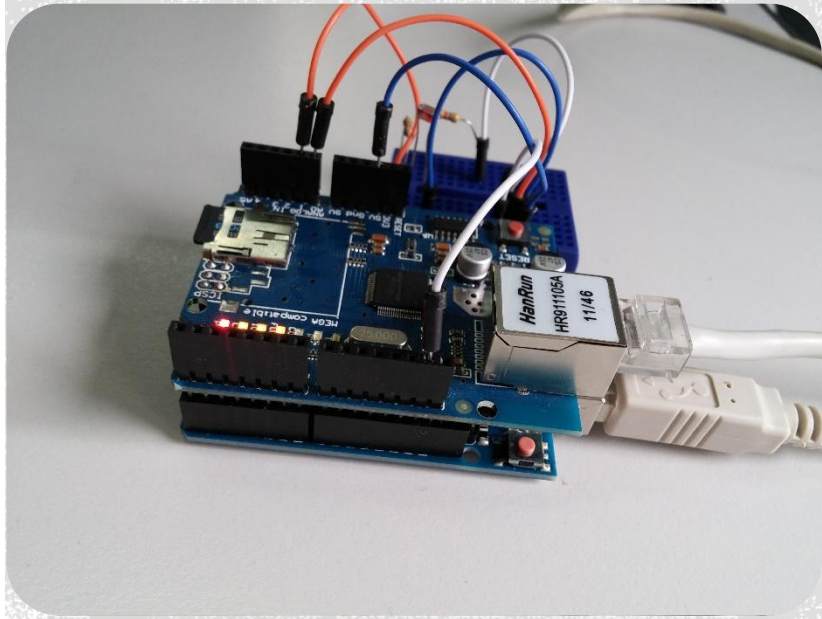
```
#include <SPI.h>

#include <Ethernet.h>

byte mac[ ]= {0x90,0xA2,0xDA,0x00,0x55,0xA0};

IPAddress ip(192,168,1,97);
```

// Come già detto, è necessario impostare l'inidrizzo MAC della scheda Ethernet, possibilmente rispettando quello proprio o quello di una vecchia scheda in disuso. L'indirizzo IP viene invece definito in modo arbitrario o tramite DHCP. Nel nostro esempio abbiamo scelto un IP fisso appartenente allo spazio degli indirizzi privati.

```
EthernetServer server(80);
```

// Creiamo l'istanza chiamata "server" che risponde al protocollo IP e che dialoga sulla porta 80 (se ne possono creare 5 su porte diverse)

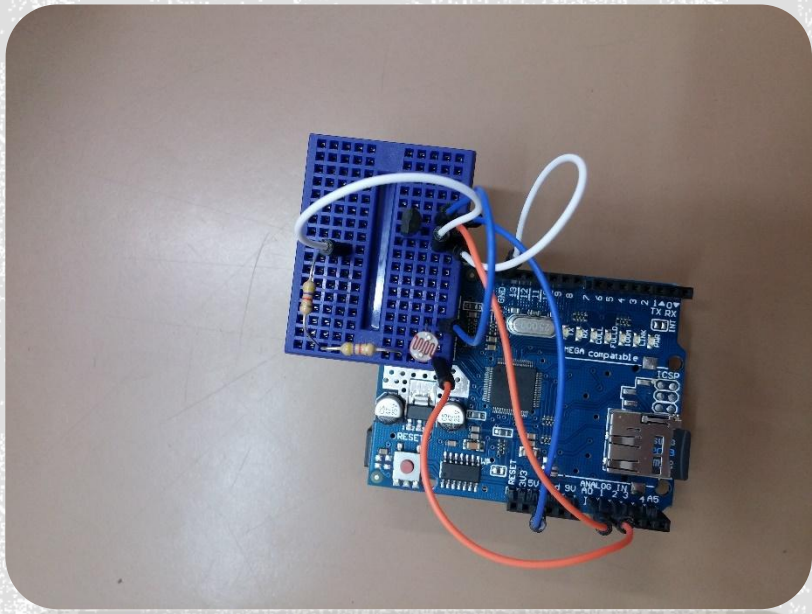
PROGETTO



Adesso definiamo le variabili e le costanti necessarie alla gestione della temperatura e della luce.

```
Int PinCalore =0; // A0 per la NTC o per l'LM35
Int PinLuce=1; // A1 per la fotoresistenza
int Luce;
double Celsius;
double Tmin=1000;
double Tmax=0;
int Lmin=1000;
int Lmax=0;
double R1=10000.0;
double V_in=5.0;
double A= 1.189148e-3; // parametri  sperimentali NTC
double B= 2.34125e-4;
double C= 8.76741e-8;
double K=6; // fattore di dissipazione dal Datasheet;
```

PROGETTO

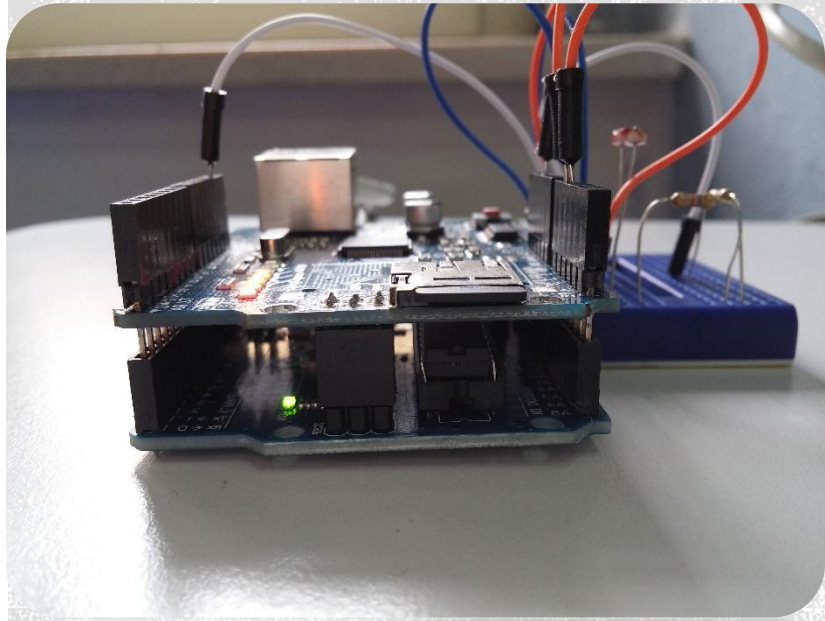


Nella parte di Setup() inizializziamo la porta seriale dalla quale potremo monitorare il traffico, sia in fase di test che di funzionamento normale. Con la funzione Ethernet.bgin (mac,ip); viene inizializzata la libreria e per essere certi che l'assegnazione dell'IP sia andata a buon fine o per vedere cosa ha assegnato un eventuale DHCP, ne facciamo una stampa con l'istruzione Ethernet.localIP().

```
void setup() {

Serial.begin(9600);

Ethernet.begin (mac,ip);

server.begin;

Serial.print ("il Server è a questo indirizzo ");

Serial.println(Ethernet.localIP());

}

void loop ( )

{
```

Misura( ); // la misurazione viene chiamata come funzione esternamente dal loop che fornirà i valori di luce e temperatura; questo per concentrarci sulla gestione del colloquio attraverso la creazione dell'istanza client se c'è una connessione che è stata attivata sul server. La ricezione dei dati dal client che si è connesso viene considerata terminata solo se viene ricevuta una riga vuota ed è questo il test che viene effettuato per continuare ad inserire caratteri in un astringa di ricezione "Response" prendendo in considerazione i primi 15 caratteri, dentro ai quali siamo sicuri che ci arriverà l'informazione significativa. Quello che ci aspettiamo è una richiesta di aggiornamento della pagina o una richiesta di azzeramento dei valori minimi e massimi memorizzati.(solo se troviamo il carattere "C" come invio dal client all'interno del messaggio http

PROGETTO



PROGETTO

```
Ethernet client = server.available(); //Si è collegato qualcuno?
If (client) {
Serial.println(“Nuovo client”);
Boolean currentLineIsBlank = true; // fine messaggio HTTP= linea blank
String Response; // qui ci vanno i caratteri ricevuti
Response= String(“”); // iniziamo con una stringa vuota
int RL=1; // var che conta i caratteri in arrivo dal client
while (client.connected()) // adesso che il client è connesso
{
if (client.available()) {
char c=client.read(); //leggiamo i caratteri provenienti dal client
Serial.write(c); // e intanto li scriviamo sulla seriale
if (RL<=15) {Response+=c; RL+=1; // incrementiamo RL sino a 15
}
```

// verifichiamo se è abbiamo ricevuto tutto il messaggio in arrivo dal client attraverso la riga vuota. Se la condizione si verifica, possiamo procedere alla gestione di quanto ricevuto

```
if (c=‘\n’ && currentLineIsBlank)
Serial.println(Response.substring(7,8));  //quale azione?
client.println(“HTTP/1.1 200 OR”);
client.println(“Content-Type: text/html”);
```

client.println(“Connection: close”);

client.println( );

client.println(“<!DOCTYPE HTML>”);

client.println(“<html>”);

client.println(“Benvenuto al micro server Guastella
 delle condizioni

ambientali.

”);

// Adesso controlliamo se prima di proseguire , dobbiamo azzerare i valori in quanto il

// client ci ha trasmesso la lettera “C”

if (Response.substring(7,8) ==“C” { // solo se c’è in una precisa posizione il carattere ”C” alla posizione 7,8 troveremo sempre il primo carattere inviato dal client

Tmin=1000;

Tmax=0;

Lmin=1000;

Lmax=0;

Misura( );

Client.print(“Massimi e minimi azzerati come richiesto”);

Client.println(“

”);

}

PROGETTO

```
// altrimenti non modifichiamo i valori minimi e massimi e stampiamo sul client le informazioni vere e proprie

client.print("Temperatura corrente: ");

client.print(Celsius);

client.print(" &deg; C ");

client.print("<br /> Temperatura minima: ");

client.print(Tmin);

client.print(" &deg; C ");

client.print("<br /> Temperatura massima: ");

client.print(Tmax);

client.print(" &deg; C ");

client.print("<br /><br />");

client.print("Luminosit&agrave corrente: ");

client.print(Luce);

client.print(" Lux");

client.print("<br />Luminosit&agrave minima: ");

client.print(Lmin);

client.print(" Lux");

client.print ("<br />Luminosit&agrave massima: ");

client.print(Lmax);

client.print(" Lux");

client.print("<br /><br />");
```

PROGETTO

```
// creazione dei pulsanti Aggiorna e Cancella
client.print("<form action='http://");
client.print(ip);
client.println("' method='post'>");
client.println("<input type=\"submit\"value=\"Aggiorna Dati\"/>");
client.print("</form> <br>");
client.println("</html>");
break;
}
```

PROGETTO

```
// Se non si raggiunge la fine della comunicazione la variabile
// urrentLinesIsBlank è false e si prosegue nella lettura dei caratteri
// di ingresso
if (c =='\n') { currentLineIsBlank=true; }
else if (c!='\r') {currentLineIsBlank=false;} // torniamo all'inizio a leggere un altro carattere dal buffer
}
}
delay(1);
client.stop();
Serial.println("Client disconnected"); // chiude la comunicazione col client
}
}
```

PROGETTO

PROGETTO

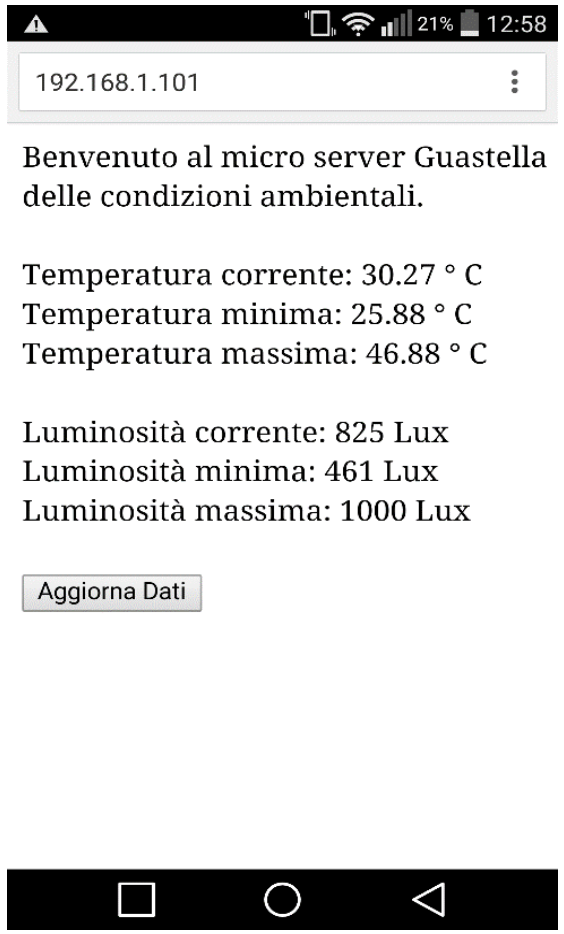
// Misura( ) legge i valori dei due sensori collegati alle porte analogiche A0 e A1. Ripetendo la lettura tre volte per la temperatura e due volte per la luce per poi farne la media. Sempre in questa funzione abbiamo il confronto dei valori memorizzati con quelli correnti per gestire i minimi e i massimi valori.

```
void Misura()
{
double adc_raw =analogRead(PinCalore);
delay(10);
Celsius= (adc_raw/1024)*500;
double Kelvin= Celsius+273.15;
Luce=analogRead(PinLuce);
delay(10);
```

```
Luce=(Luce +analogRead (PinLuce))/2;
if (Luce<=Lmin) {Lmin=Luce;}
if (Lmax<=Luce) {Lmax=Luce;}
if (Celsius<=Tmin) {Tmin=Celsius;}
if (Tmax<=Celsius) {Tmax=Celsius;}
}
```

PROGETTO

VISUALIZZAZIONE DATI



Dopo aver caricato il programma all'interno del nostro Arduino Uno, si passa alla fase successiva ovvero quella del collegamento alla rete tramite il cavo RJ 45.

Dopo la connessione alla rete locale (LAN) , il programma è pronto ed aspetta i comandi che arrivano da un qualunque client sempre all'interno della rete locale. Il client deve solo digitare l'IP su internet ed automaticamente compariranno i dati di luce e di temperatura del luogo dove Arduino uno è collocato.

FINE.

Autore: Guastella Davide

GRAZIE PER L’ATTENZIONE