

22/04/2017

# Network-Nodes

Comunicazione ottica WLAN



**Network**  
Nodes

Pietro Rignanese, Andrea Polenta  
DOMOTICA WI-FI  
TESINA E PROGETTO ONLINE: [Github](#)  
Questo documento funge sia da tesina che da "Manuale Utente"

# INDICE

<b>Struttura</b>	4
<b>Comunicazioni</b>	5
<b>Terminale &gt; Nodo Master &gt; Nodo Interessato</b>	5
<b>Nodo Master &gt; Nodo Interessato</b>	6
<b>Schema Network</b>	7
<b>Occorrente Hardware</b>	8
<b>Occorrente Software</b>	8
<b>Cos'è l'ESP8266?</b>	8
Come installare la scheda su Arduino	9
Come utilizzare la scheda ESP8266	9
Come collegare la scheda ESP8266 alla rete WI-FI	9
Programma di lancio per il collegamento	10
<b>Cos'è Arduino?</b>	11
<b>Fase Operativa</b>	11
<b>Prima fase: Collegamento Scheda WI-FI ad una rete LAN</b>	12
<b>Seconda fase: Collegamento Ottico Nodo Master -&gt; Nodo 1</b>	20
Nodo Master: Collegamento ottico	20
Nodo 1: Collegamento ottico	22
<b>Terza fase: Collegamento Ottico Nodo Master -&gt; Nodo 1 -&gt; Nodo 2</b>	36
Primo prototipo: Simple Connection	36
Secondo prototipo: Connected Oriented	43
<b>Database (Firebase)</b>	60
<b>App Android</b>	61
<b>Problemi rilevati</b>	62
<b>Problemi</b>	62
<b>Soluzioni</b>	62
<b>Algoritmi</b>	63
<b>Nodo Master:</b>	63
<b>Nodo 1:</b>	63
<b>Nodo 2:</b>	64
<b>Conclusioni e costi complessivi</b>	64

**Costi:** ..... 64

**Conclusioni:** ..... 64

## Struttura

Abbiamo previsto un terminale PC, un nodo master e due nodi, schematizzati in questa maniera (classifichiamo questa rete come P2P):

- Terminale (Invia e riceve le informazioni dei nodi attraverso la rete accedendo all'indirizzo IP del Nodo Master)
- Router (Instrada le richieste fatte tra terminale e Nodo Master sulla rete domestica e anche quella esterna andando ad aggiornare un DB)
- Nodo Master (Nodo formato da un modulo ESP8266 ESP-12E per il collegamento WI-FI con il terminale, l'invio delle informazioni sul DB e la trasmissione di informazioni tra i vari nodi presenti nella rete locale).
- Nodo1 (Nodo formato da Arduino NANO che gestisce un attuatore/relè/led e, in futuro, un eventuale sensore)
- Nodo2 (Stessa cosa del nodo precedente)

*N.B: Lo storage delle informazioni su un DB, ci serve per controllare lo stato della rete, anche da smartphone o tablet, in remoto (Internetworking: il router fungerà da gateway). Il Nodo Master può essere, a sua volta, provvisto di attuatori e sensori. Il problema della scheda ESP-12E è la scarsità di PIN disponibili: nel nostro caso è stato possibile collegare un solo attuatore/relè a tale nodo a differenza dei due collegati sugli altri nodi.*

## Comunicazioni

La comunicazione fra i nodi e il terminale avviene in questa maniera:

- Il terminale comunica con il solo Nodo Master tramite la rete Wi-Fi, e riceve ed invia a questo nodo le richieste da fare agli altri due nodi della rete.
- Il Nodo Master comunica con il terminale, via Wi-Fi, e, con il Nodo1, attraverso un collegamento ottico (LED IR e Sensor IR), e la stessa comunicazione avviene tra il Nodo1 e Nodo2
- Ogni nodo può comunicare solamente con il suo nodo adiacente formando un ponte di nodi
- Se il terminale deve contattare il Nodo2, la sua richiesta passerà prima a tutti i nodi

Esempi di comunicazione:

*Terminale > Nodo Master > Nodo Interessato*

- Attraverso una pagina HTML, appositamente caricata in locale attraverso la scheda Wi-Fi, avremo la possibilità di inviare informazioni alla shield da qualsiasi terminale noi vogliamo (es. PC, smartphone, tablet); inoltre dalla stessa pagina sarà possibile reperire lo stato di ogni nodo(attuatori ON/OFF e sensori)
- Se volessimo inviare una certa informazione, come ad esempio l'accensione o lo spegnimento di un LED(Attuatore e/o Relè), non dovremmo far altro che selezionare l'apposito 'Radio Button' per poi determinare l'invio della Form attraverso il Bottone 'Invio'
- In questa maniera verrà mandata una richiesta tramite rete Wi-Fi al Nodo Master gestito dalla scheda ESP8266
- La scheda Wi-Fi riceverà la richiesta, la inoltrerà, tramite segnale infrarossi, al nodo adiacente(Nodo 1)
- Il nodo adiacente(Nodo 1) riceve questo segnale e lo elabora per capire se è lui il destinatario
- Se così non fosse lo manda all'altro nodo adiacente(Nodo 2) con il collegamento ottico
- Quando il messaggio è arrivato a destinazione, la scheda arduino presente su tale nodo spegnerà/accenderà l'attuatore o relè
- Verrà mandato, in ritorno, l'avvenuta/o accensione/spegnimento

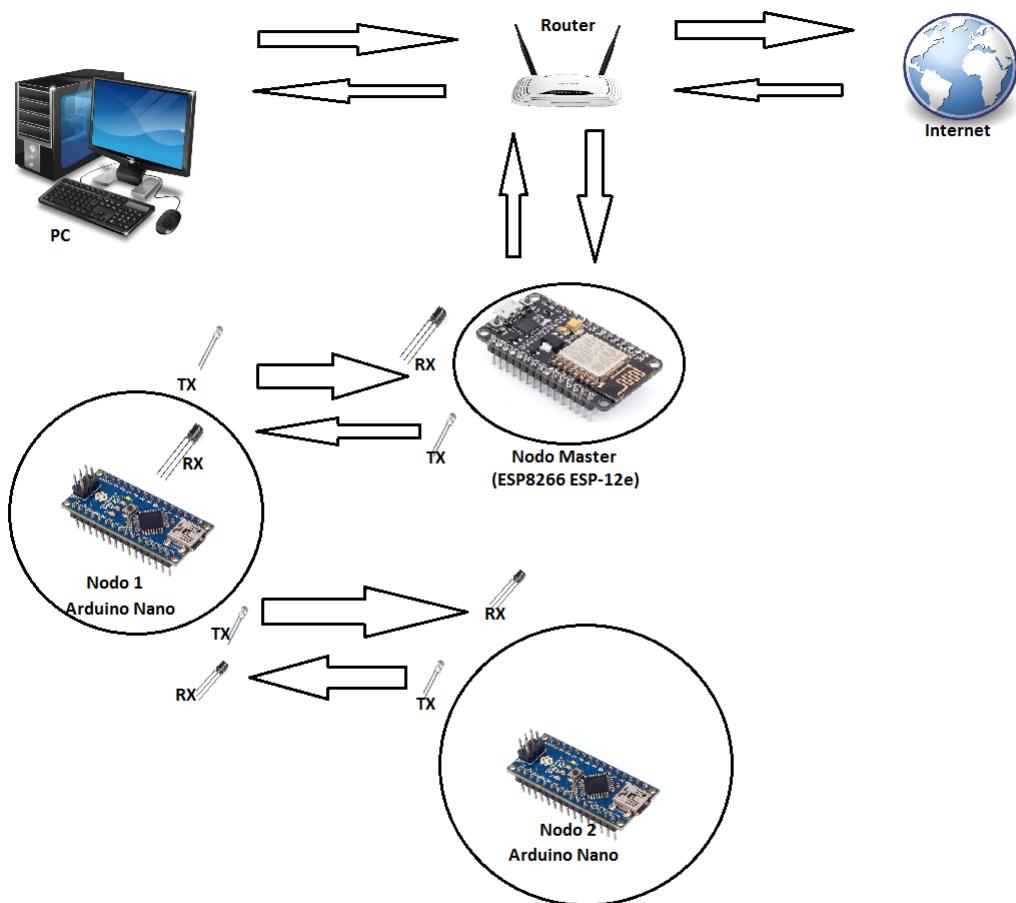
- Il messaggio di ritorno arriverà fino al Nodo Master che darà in risposta al PC un'altra pagina HTML aggiornata
- Il Nodo Master, successivamente, manderà questo dato sul database per poterlo consultare in remoto tramite smartphone
- Il PC riceverà la pagina HTML aggiornata e vedrà lo stato attuale di tutti i nodi per una prossima operazione

### **Nodo Master > Nodo Interessato**

- Il Nodo Master avrà un proprio indirizzo IP nella rete (es: 192.168.1.100)
- Il Nodo Master riceve la richiesta dal PC da instradare al giusto nodo
- Il Nodo Master comunicherà con il Nodol, essendo il suo adiacente
- Verrà mandato un segnale ottico al Nodol con il dato da elaborare
- Il Nodol riceve questo dato e lo elabora; una volta elaborato restituisce un ACK al Nodo Master
- Il Nodo Master chiude la comunicazione
- Il Nodol se dovesse ricevere un dato corrotto richiede il rinvio del dato
- Il Nodol elabora il dato ricevuto e capisce se appartiene a lui altrimenti lo manda, alla stessa maniera della prima comunicazione, al nodo adiacente (Nodo2)
- Così come il Nodol ha elaborato il dato all'andata, il Nodo Master lo farà per il ritorno
- Una volta chiusa la comunicazione il Nodo Master aggiorna la pagina HTML e comunica il cambiamento al database

*N.B: Se non dovesse esserci instradamento in uscita (connessione internet) sulla rete LAN, la connessione tra i nodi non ne risentirebbe e continuerebbe ad operare. Il database verrà aggiornato appena sarà presente una connessione in uscita. Per verificare lo stato di ogni nodo, si è pensato di collegare un Display LCD o OLED. Su questo display compariranno le azioni che sta eseguendo tale nodo. Esempio: Nodol - Display (Sto comunicando...) -> Nodo2 - Display (Sto ricevendo...)*

## Schema Network



Come possiamo vedere dall'immagine, abbiamo un PC che funziona da terminale: collegato alla rete WI-FI, invia e riceve informazioni attraverso il Nodo Master. Il Nodo Master, collegato alla rete, instrada le richieste e le informazioni, mandate dal terminale e dagli altri nodi, nella giusta direzione.

## Occorrente Hardware

- 1 terminale (PC, Tablet, Smartphone)
- 1 Router Wi-Fi
- 1 ESP8266 ESP-12E oppure una qualsiasi shield Wi-Fi
- 2 Arduino NANO/UNO/MEGA/Mino PRO
- 4 TX Diodo LED infrarossi
- 4 RX Sensori IR
- 3 Breadboard (830 fori)
- Cavetti e jumper per i collegamenti
- 3 Display LCD per verificare lo stato di ogni nodo
- 4 pulsanti

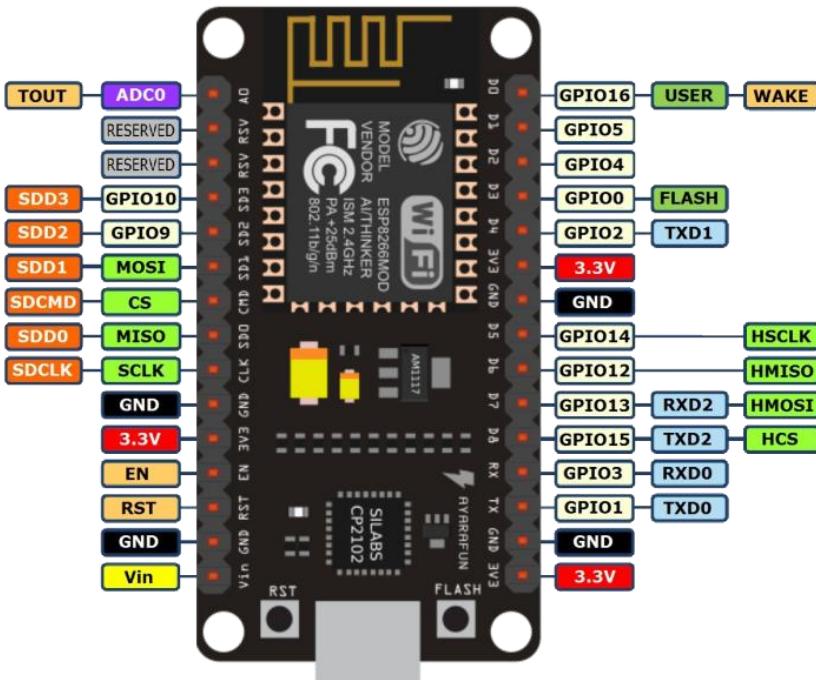
## Occorrente Software

- [Arduino IDE 1.8.0](#) o superiori
- [Firebase](#)
- [Core ESP8266](#)
- [Libreria gestione ESP8266](#)
- [Libreria gestione segnale IR](#)
- [Libreria gestione Display LCD](#)

## Cos'è l'ESP8266?

L'ESP8266 è una scheda che incorpora l'uso del Wi-Fi. Si hanno tanti tipi di ESP8266, cambiano forme e costi, ma essenzialmente svolgono lo stesso principio. Per questo progetto si è utilizzato un ESP8266 ESP-12E con scheda di programmazione già integrata, per facilitarne l'utilizzo.

La scheda presenta 30 pin e un'alimentazione attraverso micro USB. Questo tipo di scheda può sostituire o essere un degno alleato di arduino, perchè dispone di pin che possono essere utilizzati per la gestione di dati di INPUT e/o OUTPUT. La scheda, come si può vedere, è compatta, leggera, di buona fattura e può essere utilizzata per vasti utilizzi (domotica, robotica, automazione in generale).



### Come installare la scheda su Arduino

- Lancia l'IDE di Arduino dal tuo PC
- Vai su *File > Impostazioni*
- Inserisci questo link  
[http://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/stable/package_esp8266com_index.json) nelle URL aggiuntive per la gestione delle schede e clicca su OK
- Vai su *Strumenti > Gestione Schede* e installa la piattaforma "ESP8266"
- Installare la [libreria ESP8266](#) aggiuntiva della scheda

*N.B: Installazione di questo pacchetto aggiuntivo per l'IDE di Arduino, comprende tante schede WI-FI in commercio*

### Come utilizzare la scheda ESP8266

- Lancia Arduino
- Selezione *Strumenti > Schede* e seleziona la shcheda che vuoi utilizzare nella sezione "ESP8266 Modules"

Noi utilizziamo, per questo progetto, una ESP8266 ESP-12E, quindi selezioneremo la scheda ESP-12E MODE MCU 1.0.

### Come collegare la scheda ESP8266 alla rete WI-FI

Per collegare la shield Wi-Fi, ad un router, dobbiamo seguire questi passi:

- Collegare la shield, tramite cavo mini USB, al PC
- Aprire l'IDE di Arduino 1.8.0 o superiori
- Andare su *Strumenti* e selezionale la scheda ESP8266 corrispondente a quella inserita(nel nostro caso MODE MCU 1.0)
- Impostare la porta di collegamento seriale (Esempio COM4)
- Impostare il bound rate a 115200
- Lanciare il programma per la connessione

### Programma di lancio per il collegamento

Analizziamo, riga per riga, il programma di lancio per il collegamento della scheda al router.

*N.B: abbiamo utilizzato, per le varie prove, un hotspot Wi-Fi con un HUAWEI P8. Il telefono, in questo caso, funge da router...*

```
// Inclusione dell'apposita libreria per la Shield adoperata
#include <ESP8266WiFi.h>

//Definizione delle credenziali per la connessione per
//accedere al Router
const char* ssid = "*****";
const char* password = "*****";

//Istanziamo 'server(80)' come oggetto della classe
//WiFiServer.
WiFiServer server(80);

void setup() {
  // Connessione seriale a 115200.
  Serial.begin(115200);
  delay(10);

  //Connessione alla rete WiFi
  //Mostriamo l'IP della scheda connessa alla rete
  Serial.print("\n\nInizializzazione rete WiFi: ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connesso!");
}

void loop() {
  // Vediamo se esiste una connessione con un client. Se
  // non esiste ritorniamo al loop.
  WiFiClient cliente = server.available();
  if (!cliente)
  {
    return;
  }
  cliente.println("HTTP/1.1 200 OK");
  cliente.println("Content-Type: text/html");
  cliente.println("Connection: close");
  cliente.println();
  cliente.println("Hello World!");
}
```

Codice:

- Includiamo la libreria che abbiamo scaricato in precedenza, per utilizzare, più facilmente, le funzionalità della shield Wi-Fi (`#include <ESP8266.h>`)
- Creiamo due variabili di tipo string ed andiamo ad inserire le credenziali per l'accesso al router SSID e Password
- SSID è il nome della rete, nel nostro caso HUAWEI P8
- La Password è quella con cui accedete al router, per connettervi a internet
- Impostiamo la porta 80 per la comunicazione
- Nel ciclo di loop è presente la parte di programma che andrà a connettere la shield al router
- Una volta avvenuta la connessione verrà visualizzato, sulla porta seriale di arduino, l'indirizzo IP della scheda
- La scheda risulterà collegata alla rete LAN

*N.B: Potete verificare l'indirizzo IP, di ogni dispositivo connesso, andando sull'indirizzo IP del router, accedendo tramite browser. Di solito i router hanno come indirizzo 192.168.0.1. Potete verificare il vostro indirizzo di router sul manuale di quest'ultimo*

## Cos'è Arduino?

Arduino è una piattaforma hardware low-cost programmabile, con cui è possibile creare circuiti "quasi" di ogni tipo per molte applicazioni, soprattutto in ambito di robotica ed automazione. Si basa su un Microcontrollore della ATMEL, l'ATMega168/328: per esempio l'Arduino Uno monta un ATMega328. Nasce a Ivrea, nel 2005, da un'idea di un professore universitario, un Ingegnere Elettronico, Massimo Banzi, che decise di creare una piattaforma per i propri studenti, così da facilitarli nello studio dell'Interaction Design. Fu un completo successo, a tal punto da spingere l'ingegnere a rendere questa piattaforma, Open Source (in realtà è Open Hardware) cioè è possibile trovare sul sito ufficiale [www.arduino.cc](http://www.arduino.cc), i circuiti, i componenti e addirittura le istruzioni per realizzarla da soli. Ciò che dovrebbe interessare in realtà sono gli schemi circuitali: essendo Open, e quindi visionabili da tutti, possono essere continuamente migliorati dalla comunità e grazie ad essi sono state sviluppate un numero incredibile di librerie software che rendono davvero semplice l'interfaccia con periferiche di qualsiasi tipo.

Fu un gruppo di studenti della facoltà di Ingegneria Informatica a scrivergli la libreria, l'IDE (libreria portatile su ogni sistema operativo) e le prime API; grazie a questi pre-ingegneri, Arduino tutt'oggi programma in modo fluido, semplice e molto intuitivo. In Internet, addirittura, si possono trovare librerie già scritte in base al nostro bisogno. Per esempio se vogliamo fare qualche applicazione e ci serve qualche funzione in particolare o qualche supporto per sensori possiamo ricorrere, appunto, alla navigazione in Internet.

## Fase Operativa

## Prima fase: Collegamento Scheda WI-FI ad una rete LAN

- Una volta testata la connessione di questa shield, si è implementata un'interfaccia grafica in HTML per comunicare i cambiamenti di stati di alcuni attuatori e relè presenti sul Nodo Master (attuatori e relè sono sostituiti da semplici led per facilitare i collegamenti, ma il concetto non cambia).
- La pagina HTML viene mandata al terminale dalla scheda ESP8266 e aggiornata ogni volta avviene un cambiamento di stato dettato dal terminale.
- Il cambiamento di stato avviene attraverso una semplice FORM HTML che invia i dati cambiati all'URL della pagina. Ecco la pagina HTML con la relativa Form:

**The Sentinel**

**Controllo Nodi**

Nodo	Stato			
	Relè	Attuatore	Sensore n°1	Sensore n°2
Stazione Master	OFF	OFF	0.00	0.00
	<input type="radio"/> ON <input checked="" type="radio"/> OFF	<input type="radio"/> ON <input checked="" type="radio"/> OFF	/	/
Stazione n°1	Relè	Attuatore	Sensore n°1	Sensore n°2
	OFF	OFF	0.00	0.00
Stazione n°2	<input type="radio"/> ON <input checked="" type="radio"/> OFF	<input type="radio"/> ON <input checked="" type="radio"/> OFF	/	/
	OFF	OFF	0.00	0.00
<input type="radio"/> ON <input checked="" type="radio"/> OFF	<input type="radio"/> ON <input checked="" type="radio"/> OFF	/	/	

N.B: La pagina HTML ha tutte le funzionalità! Ma di queste solo l'accensione del relè e dell'attuatore sul Nodo Master funziona! Il resto verrà implementato man mano...

- Il terminale, per accedere a tale pagina HTML, dovrà allocarsi, tramite browser(Chrome, Firefox, Edge, ....) all'indirizzo IP associato alla shield Wi-Fi.
- Tale indirizzo sarà visualizzato su terminale, per le prime prove, e, in un secondo momento, su un display posizionato sul nodo.
- Il programma di lancio:

```
// Inclusione dell'apposita libreria per la Shield adoperata
#include <ESP8266WiFi.h>

//Definizione delle credenziali per la connessione per
accedere al Router
```

```
const char* ssid = "*****";
const char* password = "*****";

// Definiamo la mappatura della Shield
ESP8266(Datasheet)
```

```

#define D0 16
#define D1 5 // LED 1
#define D2 4 // LED 2
#define D3 0
#define D4 2
#define D5 14
#define D6 12
#define D7 13
#define D8 15
#define D9 3 // RX0
#define D10 1 // TX0

// Istanziamo 'server(80)' come oggetto della classe WiFiServer.
WiFiServer server(80);

void setup() {

// Connessione seriale a 115200.
Serial.begin(115200);
delay(10);

// Inizializziamo i pin per i LED.
pinMode(D1, OUTPUT);
pinMode(D2, OUTPUT);
digitalWrite(D1, LOW);
digitalWrite(D2, LOW);

// Connessione alla rete WiFi
// Mostriamo l'IP della scheda connessa alla rete
Serial.print("\n\nInizializzazione rete WiFi: ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("\nConnesso.");
Serial.print("IP del dispositivo: ");
Serial.print(WiFi.localIP());

// Inizializziamo il WebServer.
server.begin();
Serial.println("\nWebServer inizializzato.\n");
}

void loop() {

```

*// Vediamo se esiste una connessione con un client. Se non esiste ritorniamo al loop.*

```

WiFiClient cliente = server.available();

if (!cliente)
{
return;
}

// Nella caso ci sia una connessione con un client, mostriamo un messaggio di avvenuta connessione e mantenendo aperta la connessione.
Serial.println("Client connesso: ");
while (!cliente.available())
{
delay(1);
}

//Realizziamo la lettura della form.
String form = cliente.readStringUntil('\r');
Serial.println(form);
cliente.flush();

int RM = LOW; //Variabile per controllo stato Relè
int AM = LOW; //Variabile per controllo stato Attuatore

//Controlliamo il risultato della form e vediamo se è presente un cambiamento di stato dell'attuatore e relè
if(form.indexOf("RelayMaster=ON")!=-1)
{
digitalWrite(D1, HIGH);
RM = HIGH;
}
if(form.indexOf("RelayMaster=OFF")!=-1)
{
digitalWrite(D1, LOW);
RM = LOW;
}
if(form.indexOf("AttuatoreMaster=ON")!=-1)
{
digitalWrite(D2, HIGH);
AM = HIGH;
}
if(form.indexOf("AttuatoreMaster=OFF")!=-1)
{
digitalWrite(D2, LOW);
}

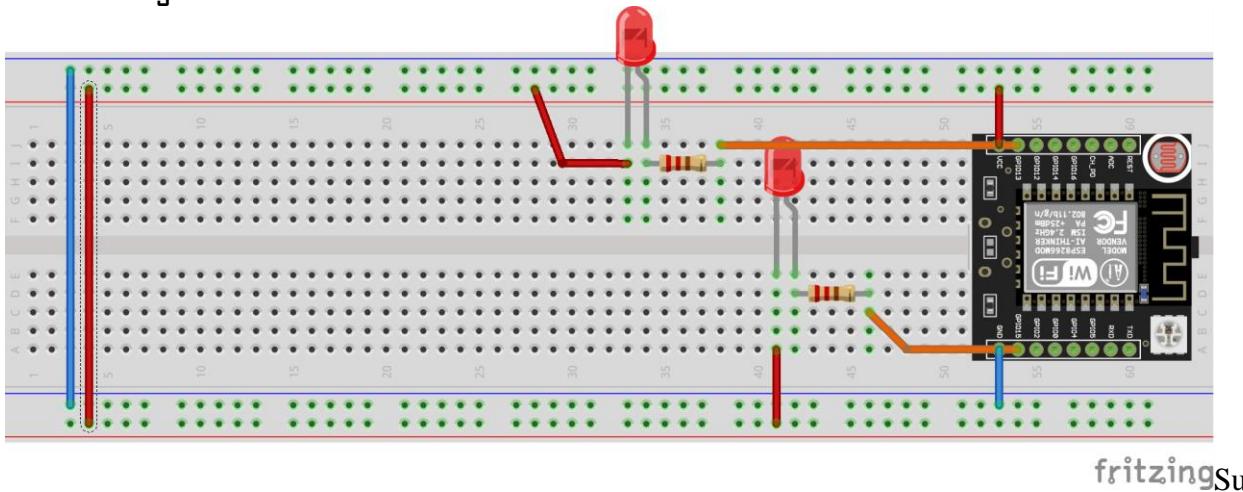
```

```
AM = LOW;
}

// Pagina WEB da mandare al Client
CODICE_PAGINA_HTML();
```

```
// Client disconnesso
delay(1);
Serial.println("Cliente Desconectado.\n");
}
```

Ecco i collegamenti:



Fritzing non è presente l'ESP-12E compreso di basetta per i collegamenti e USB incorporata, ma una semplice ESP-12E. Con un pò di fantasia immaginiamo che questa scheda sia quella che stiamo utilizzando noi (la scheda con 30 pin), per facilitarne la spiegazione.

I due collegamenti che vanno ai led sono i pin D1 e D2.

*N.B: Nella seconda fase lasceremo solo un led che rappresenterà un solo relè/attuatore, perché abbiamo poca disponibilità di piedini nella ESP-12E. In questo caso il led sarà collegato al piedino D4; i piedini D1 e D2 saranno riservati al display lcd I2C.*

*P.S. La pagina HTML che viene richiamata dal codice presente nel Nodo Master sarà sempre il seguente:*

```
----- Pagina HTML -----
-----
cliente.println("HTTP/1.1 200 OK");
cliente.println("Content-Type: text/html");
cliente.println(""); //Separatore
cliente.println("<!DOCTYPE html>");
```

```
cliente.println("<html>");
cliente.println("<head>");
cliente.println("<style type='text/css'>");
cliente.println("body");
-----
-----CSS-----
```

<pre> cliente.println("          {");  cliente.println("      background: #DCDCDC;");  cliente.println("      font-family: Rockwell;");  cliente.println("      }");  cliente.println("      h1");;  cliente.println("      {");  cliente.println("      color: #000000;");  cliente.println("      text-align: center;");  cliente.println("      }");  cliente.println("      h2");;  cliente.println("      {");  cliente.println("      color: #000000;");  cliente.println("      text-align: center;");  cliente.println("      }");  cliente.println("      table");     </pre>	<pre> cliente.println("      .tinto tr");;  cliente.println("      {");  cliente.println("      border: 1px solid #DCDCDC;");  cliente.println("      }");  cliente.println("      .tinto td");;  cliente.println("      {");  cliente.println("      font-size: small;");  cliente.println("      background: #DCDCDC;");  cliente.println("      border: 1px solid #000000;");  cliente.println("      {");  cliente.println("      .cinto");;  cliente.println("      {");  cliente.println("      padding: 0px 0px 0px 0px;");  cliente.println("      }");  cliente.println("      table, td");     </pre>
<pre> cliente.println("      {");  cliente.println("      border-collapse: collapse;");  cliente.println("      color: #000000;");  cliente.println("      }");  cliente.println("      .tinto");     </pre>	<pre> cliente.println("      {");  cliente.println("      border-collapse: collapse;");  cliente.println("      border: 3px solid #000000;");  cliente.println("      background:#FAFAD2;");  cliente.println("      {");  cliente.println("      table, tr");     </pre>
<pre> cliente.println("      {");  cliente.println("      border-collapse: collapse;");  cliente.println("      border: 1px solid #DCDCDC;");  cliente.println("      }");     </pre>	<pre> cliente.println("      {");  cliente.println("      background: #ffffff;");  cliente.println("      border: 3px solid #000000;");  cliente.println("      text-align: center;");  cliente.println("      }");     </pre>

```

cliente.println("  </style>");

//-----
-----HTML-----
-----


cliente.println("  <meta charset='utf-8'>");

cliente.println("  <title>The Sentinel</title>");

cliente.println(" </head>");

cliente.println(" <body>");

cliente.println("  <h1 id='titolo'>The Sentinel</h1>");

cliente.println("  <h2>Controllo Nodi</h2>");

cliente.println("  <form>");

cliente.println("    <table align='center'>");

cliente.println("      <thead>");

cliente.println("        <tr>");

cliente.println("          <td>Nodo</td>");

cliente.println("          <td>Stato</td>");

cliente.println("        </tr>");

cliente.println("      </thead>");

cliente.println("      <tr>");

cliente.println("        <td>Stazione Master</td>");

cliente.println("        <td class='cinto'>");

cliente.println("          <table align='center' class='tinto'>");

cliente.println("            <tr>");

cliente.println("              <td>Relè</td>");

cliente.println("              <td>Attuatore</td>");

cliente.println("              <td>Sensore n°1</td>");

cliente.println("              <td>Sensore n°2</td>");

cliente.println("            </tr>");

cliente.println("          </table>");

cliente.println("        </td>");

cliente.println("      </tr>");

cliente.println("    </table>");

cliente.println("  </form>");

cliente.println("  </body>");

cliente.println("  </html>");
```

```

//Controllo stato del relè e attuatore al nodo master

if(RM==HIGH)

{

  cliente.println("      <td>ON</td>");

  firebase_upload("NodoMaster","Relay","ON");

}

else

{

  cliente.println("      <td>OFF</td>");

  firebase_upload("NodoMaster","Relay","OFF");

}

cliente.println("      <td></td>");

cliente.println("      <td>o.oo</td>");

cliente.println("      <td>o.oo</td>");

cliente.println("      <tr>");

cliente.println("      <tr>");

//Stato del radio button

if(RM==HIGH)

{

  cliente.println("      <td><input type='radio' value='ON' name='RelayMaster' checked='checked'>ON</input><input type='radio' value='OFF' name='RelayMaster'>OFF</input></td>");

}

else

{

  cliente.println("      <td><input type='radio' value='ON' name='RelayMaster'>ON</input><input type='radio' value='OFF' name='RelayMaster' checked='checked'>OFF</input></td>");

}

cliente.println("      <td><input type='radio' checked='checked'>NO DEVICE</input></td>");
```

```

cliente.println("<td>/</td>");

cliente.println("<td>/</td>");

cliente.println("</tr>");

cliente.println("</table>");

cliente.println("</td>");

cliente.println("</tr>");

cliente.println("<tr>");

cliente.println("<td>Stazione n°1</td>");

cliente.println("<td class='cinto'>");

cliente.println("<table align='center' class='tinto'>");

cliente.println("<tr>");

//Se la ricezione dell'ACK non è avvenuto, mostra il valore non aggiornato in rosso

if(CRN1)

{

  cliente.println("<td>Relè</td>");

}

else

{

  cliente.println("<td bgcolor='#FF0000'><font color='#FF0000'>Relè</font></td>");

}

if(CAN1)

{

  cliente.println("<td>Attuatore</td>");

}

else

{

  cliente.println("<td bgcolor='#FF0000'><font color='#FF0000'>Attuatore</font></td>");

}
  
```

```

cliente.println("<td>Sensore n°1</td>");

cliente.println("<td>Sensore n°2</td>");

cliente.println("</tr>");

cliente.println("<tr>");

//Controllo stato Relè e Attuatore nodo 1

if(RN1==HIGH)

{

  cliente.println("<td>ON</td>");

  firebase_upload("Nodo1/","Relay","ON");

}

else

{

  cliente.println("<td>OFF</td>");

  firebase_upload("Nodo1/","Relay","OFF");

}

if(AN1 ==HIGH)

{

  cliente.println("<td>ON</td>");

  firebase_upload("Nodo1/","Attuatore1","ON");

}

else

{

  cliente.println("<td>OFF</td>");

  firebase_upload("Nodo1/","Attuatore1","OFF");

}

cliente.println("<td>0.00</td>");

cliente.println("<td>0.00</td>");

cliente.println("</tr>");

  
```

```

cliente.println("      <tr>");

//Stato radio button Nodo 1

if(RN1==HIGH)

{

  cliente.println("      <td><input type='radio'
value='ON' name='Relay1'
checked='checked'>ON</input><input type='radio' value='OFF'
name='Relay1'>OFF</input></td>");

}

else

{

  cliente.println("      <td><input type='radio'
value='ON' name='Relay1'>ON</input><input type='radio'
value='OFF' name='Relay1'
checked='checked'>OFF</input></td>");

}

if(AN1==HIGH)

{

  cliente.println("      <td><input type='radio'
value='ON' name='Attuatore1'
checked='checked'>ON</input><input type='radio' value='OFF'
name='Attuatore1'>OFF</input></td>");

}

else

{

  cliente.println("      <td><input type='radio'
value='ON' name='Attuatore1'>ON</input><input type='radio'
value='OFF' name='Attuatore1'
checked='checked'>OFF</input></td>");

}

cliente.println("      <td>/</td>");

cliente.println("      <td>/</td>");

cliente.println("      </tr>");

cliente.println("    </table>");

cliente.println("  </td>");

cliente.println("  </tr>");

cliente.println("      <tr>");
```

```

cliente.println("      <td>Stazione n°2</td>");

cliente.println("      <td class='cinto'>");

cliente.println("      <table align='center' class='tinto'>");

cliente.println("        <tr>");

//Se la ricezione dell'ACK non è avvenuto, mostra il valore non
aggiornato in rosso

if(CRN2)

{

  cliente.println("        <td>Relè</td>");

}

else

{

  cliente.println("        <td bgcolor='#FF0000'>font
color='FF0000'>Relè</font></td>");

}

if(CAN2)

{

  cliente.println("        <td>Attuatore</td>");

}

else

{

  cliente.println("        <td bgcolor='#FF0000'>font
color='FF0000'>Attuatore</font></td>");

}

cliente.println("      <td>Sensore n°1</td>");

cliente.println("      <td>Sensore n°2</td>");

cliente.println("      </tr>");

//Controllo stato Relè e Attuatore nodo 2

if(RN2==HIGH)

{
```

```

cliente.println("      <td>ON</td>);

firebase_upload("Nodo2/","Relay","ON");

}

else

{

cliente.println("      <td>OFF</td>);

firebase_upload("Nodo2/","Relay","OFF");

}

if(AN2 ==HIGH)

{

cliente.println("      <td>ON</td>);

firebase_upload("Nodo2/","Attuatore1","ON");

}

else

{

cliente.println("      <td>OFF</td>);

firebase_upload("Nodo2/","Attuatore1","OFF");

}

cliente.println("      <td>o.oo</td>);

cliente.println("      <td>o.oo</td>);

cliente.println("      <tr>");

cliente.println("      <tr>");

//Stato radio button Nodo 2

if(RN2==HIGH)

{

cliente.println("      <td><input type='radio' value='ON' name='Relay2' checked='checked'>ON</input><input type='radio' value='OFF' name='Relay2'>OFF</input></td>");

}

else
  
```

```

{

cliente.println("      <td><input type='radio' value='ON' name='Relay2'>ON</input><input type='radio' value='OFF' name='Relay2' checked='checked'>OFF</input></td>");

}

if(AN2==HIGH)

{

cliente.println("      <td><input type='radio' value='ON' name='Attuatore2' checked='checked'>ON</input><input type='radio' value='OFF' name='Attuatore2'>OFF</input></td>");

}

else

{

cliente.println("      <td><input type='radio' value='ON' name='Attuatore2'>ON</input><input type='radio' value='OFF' name='Attuatore2' checked='checked'>OFF</input></td>");

}

cliente.println("      <td>/</td>");

cliente.println("      <td>/</td>");

cliente.println("      </tr>");

cliente.println("      </tr>");

cliente.println("      </table>");

cliente.println("      </td>");

cliente.println("      </tr>");

cliente.println("      </td>");

cliente.println("      </table>");

cliente.println("      <p align='center'>");

cliente.println("      <input type='submit' value='Envio' align='center' />");

cliente.println("      <input type='reset' value='Reset' align='center' />");

cliente.println("      </p>");

cliente.println("      </form>");
```

```

String button = " <div align='center'><a href=''";
button += IP;
button += "><button>Aggiorna</button></a></div>";
cliente.println(button);
if(js)
{
  //Parte la pagina di errore
  cliente.println("<h1 align='center'>Attenzione!!!! La connessione non è avvenuta nel modo corretto!</h1>");
  cliente.println("<h2 align='center'>Potrebbe essersi verificato:</h2>");
}
  
```

```

cliente.println("<h3> 1. Ostacolo tra due nodi</h3>");
cliente.println("<h3> 2. Componente di comunicazione difettato di un nodo</h3>");
}

cliente.println(" </body>");
cliente.println("</html>");

//-----
-----
```

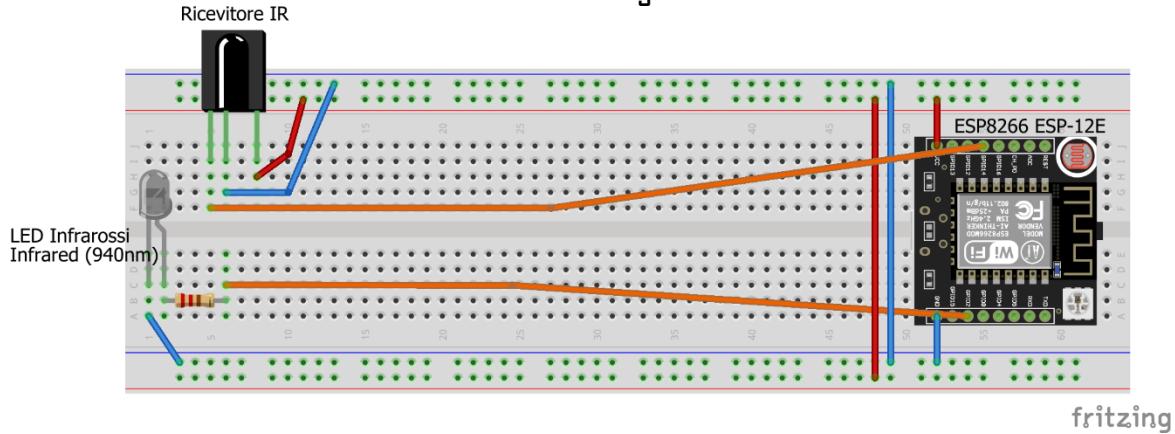
Per evitare ripetizioni, questo codice sarà sempre presente sul Nodo Master.

## Seconda fase: Collegamento Ottico Nodo Master -> Nodo 1

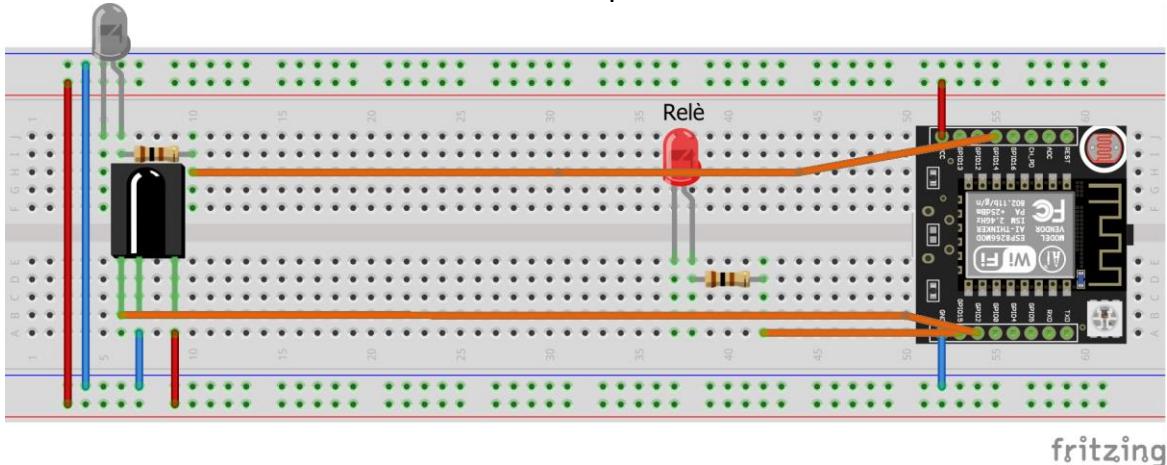
Per questa fase, avremo la comunicazione fra due nodi, quindi dividiamola, per semplicità, in due sotto-fasi.

Nodo Master: Collegamento ottico

Lo schema che andremo a realizzare sarà il seguente:



Tale schema dovrà essere incorporato con lo schema della prima fase; così facendo avremmo il nostro Nodo Master completo!



I collegamenti sono i seguenti:

	DISPLAY	LED	EMETTITORE IR	RICEVITORE IR
ESP8266	DI, D2	D4	DO	D3

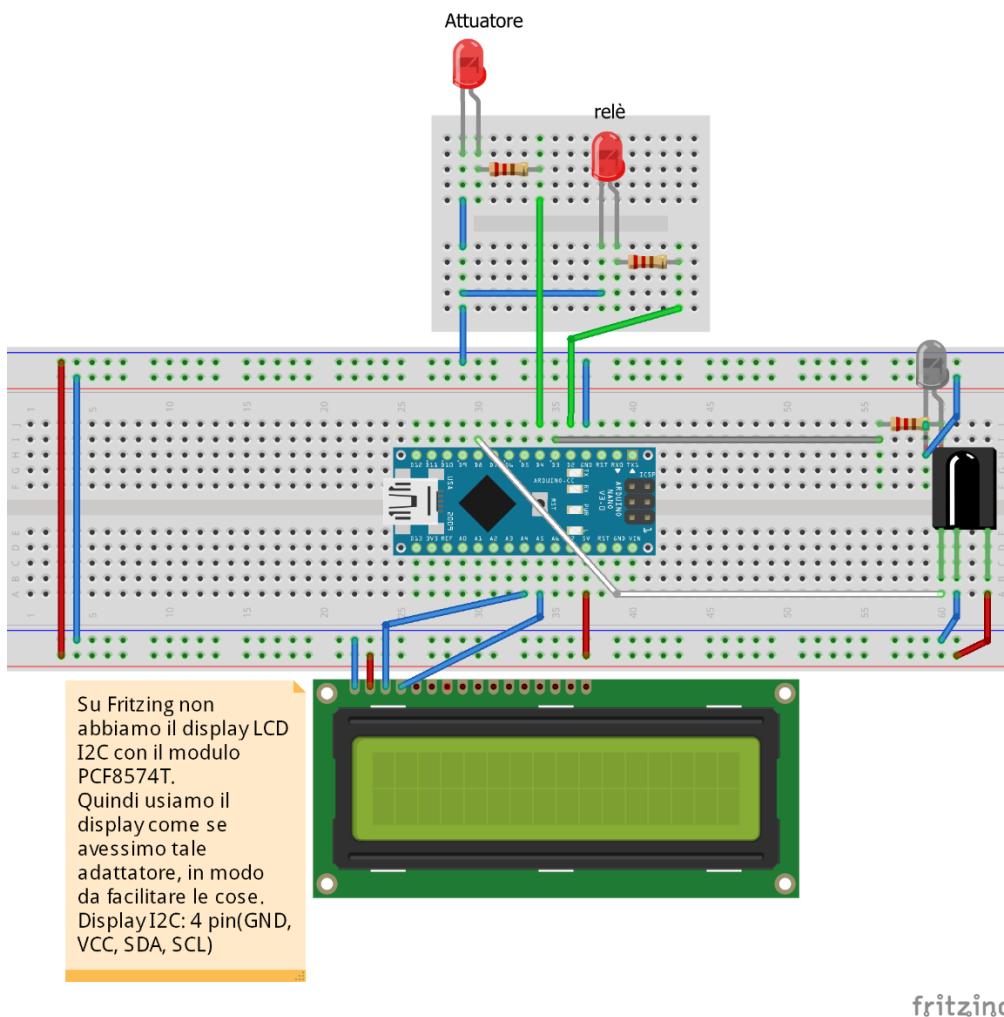
Così facendo occuperemo tutti i pin disponibili. Purtroppo ESP-12E non ha tanti pin da poter essere utilizzati, perchè gli altri sono tutti gestiti dalla scheda stessa.

*N.B: Possiamo anche non utilizzare i resistori, perchè la tensione di uscita sui pin non supera i 3volt. Se dovessimo usare resistenze alte la luce risulterebbe più fievole, comportando un possibile ostacolo alla comunicazione*

*Durante i collegamenti e i test abbiamo riscontrato un problema! Non sappiamo a cosa sia dovuto, ma se dovessimo alimentare il sensore IR con i 5v della shield WiFi, i dati risulterebbero molto corrotti! Utilizzando l'alimentazione di Arduino i risultati sono migliori... comunque abbiamo dei piccoli problemi, dovuti a disturbi e alla scarsa schermatura, nel rilevare il valore dal sensore. Spiegheremo questo più avanti...*

Da aggiungere all'occorrente software, per la realizzazione del Nodo Master, è la libreria per la gestione dei segnali infrarossi per la shield ESP8266 (purtroppo non può essere utilizzata la stessa libreria di arduino!).

## Nodo 1: Collegamento ottico



*N.B.: Può essere usato, tranquillamente, Arduino UNO! Arduino Nano è esattamente uguale (tranne nella dimensione) ad Arduino UNO! Noi per le prove abbiamo utilizzato Arduino UNO per comodità...*

## Connessioni:

Arduino UNO/NANO	
PIN 2	Relè
PIN 3	Emettitore LED IR
PIN 4	Attuatore
PIN 8	Ricevitore IR
PIN A4	Display SDA
PIN A5	Display SCL

5v	Breadboard
GND	Breadboard

Il collegamento dell'emettitore IR è obbligatorio farlo sul PIN3! Perchè la libreria di Arduino sull'invio del segnale infrarosso utilizza quel piedino per default. Se volessimo cambiare piedino dovremmo andare ad agire sulla libreria.

Per le prime prove, si è installato, sui due nodi, un emettitore(sul Nodo Master) e un sensore infrarossi(sul Nodo 1), per un semplice test di comunicazione. Si è inviato un piccolo dato, sottoforma di codifica NEC, dal Nodo Master al Nodo 1. Si hanno a disposizione tanti tipi di codifica: NEC, Sony, Sharp e tante altre...

Nel nostro caso, ogni nodo avrà almeno una coppia led-sensore per l'invio e ricezione infrarossi, quindi il procedimento è più complesso. Il punto che va a favore è che si conosce il dato che potrebbe arrivare, questo perchè possono verificarsi solo queste situazioni:

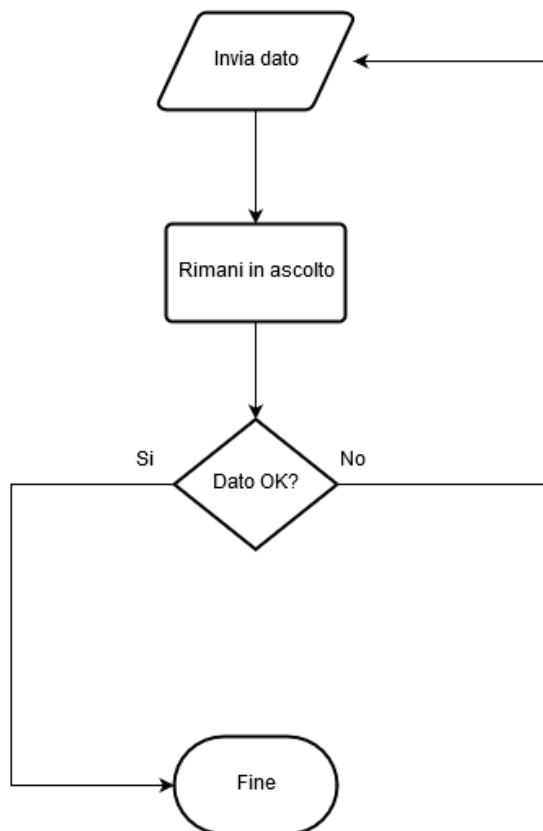
- Attivazione Relè Nodo 1;
- Attivazione Attuatore Nodo 1;
- Attivazione Relè Nodo 2;
- Attivazione Attuatore Nodo 2.

Quindi i segnali che possono essere inoltrati dall'emettitore IR(sul Nodo Master) sono essenzialmente 4; a questi c'è da aggiungere un solo segnale di ritorno(Nodo 1, Nodo 2) che verifica la corretta ricezione del dato.

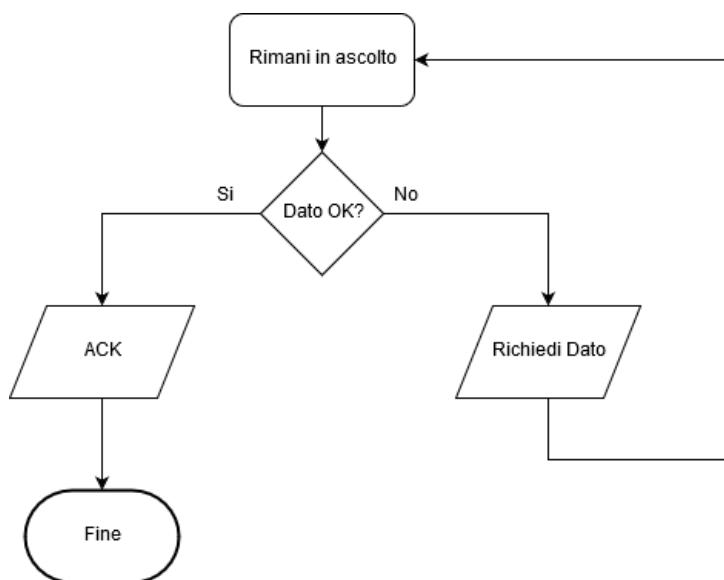
Procediamo per gradi...

- Il Nodo 1 riceve un dato dal Nodo Master attraverso la comunicazione infrarossi;
- Il Nodo Master, una volta inviato il dato, rimane in ascolto per l'ok(ACK);
- Il Nodo 1 elabora tale dato e verifica se può usarlo;
- Se può usarlo, lo elabora ed esegue un comando. Fatto questo manda l'ok al Nodo Master;
- Se non può usarlo manda al Nodo Master un dato per il rinvio del dato(il dato potrebbe essere corrotto);
- Il Nodo Master, in base alla risposta del Nodo 1, ritrasmette il dato oppure chiude la comunicazione.

Tutto questo dovrà essere implementato con un elegante e bell'algoritmo.



Questo è l'algoritmo che dovrà eseguire il Nodo Master una volta inviato il dato. Il Nodo I dovrà elaborare il dato inviato e mandargli la conferma.



*N.B: Se dovesse passare un certo lasso di tempo(es. 10 secondi) la comunicazione si interrompe!*

Una volta eseguiti tutti i collegamenti, possiamo eseguire gli sketch dei due nodi:

**Nodo Master:**

```

//-----CALCOLATORI
ELETTRONICI E RETE DI CALCOLATORI-----
-----


//-----PIETRO
RIGNANESE & ANDREA POLENTA 2017-----
-----


// Inclusione dell'apposita libreria per la Shield adoperata
#include <ESP8266WiFi.h>

// Libreria di gestione infrarossi per la shield esp8266
#include <IRremoteESP8266.h>

//Libreria per database esterno (Firebase)
#include <FirebaseArduino.h>

#define FIREBASE_HOST "*****.firebaseio.com"
#define FIREBASE_AUTH "*****"

//Librerie per gestione display
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

//Definizione delle credenziali per la connessione per accedere
al Router

const char* ssid = "*****";
const char* password = "*****";

String IP = "";

// Definiamo la mappatura della Shield ESP8266(Datasheet)
#define D0 16 // Trasmettitore IR
#define D1 5 // Display
#define D2 4 // LCD
#define D3 0 // Ricevitore IR
#define D4 2 // Relè
#define D5 14
#define D6 12
#define D7 13
#define D8 15
#define D9 3 // RX
#define D10 1 // TX

//Istanziamento le variabili di Controllo Invio/Ricezione
boolean controllo_invio = true;
boolean controllo_ricezione = true;

//Istanziamento un flag per verificare lo stato della connessione
boolean stato_conn = true;

```

```

// Istanziamo 'server(80)' come oggetto della classe WiFiServer.
WiFiServer server(80);

//Istanziamento display
LiquidCrystal_I2C lcd(0x27,20,4);

IRsend irsend(D0); //Instanziamo un oggetto per la
trasmissione dell'informazione tramite IR al pin D0

IRrecv irrecv(D3); //Instanziamo oggetto per ricezione segnale
infrarosso

decode_results results;

void setup()
{
//Display
lcd.init();
lcd.backlight();

// Instanziamo la comunicazione infrarosso
irsend.begin();
irrecv.enableIRIn();

// Connessione seriale a 115200.
Serial.begin(115200);
delay(10);

// Inizializziamo i dispositivi connessi al nodo allo stato di LOW
pinMode(D4, OUTPUT);
digitalWrite(D4, LOW);

lcd.home();
lcd.print("Benvenuto!");
delay(2500);
lcd.setCursor(0, 0);
lcd.print("Inizializzazione");
lcd.setCursor(0,1);

//Connessione alla rete WiFi
// Mostriamo l'IP della scheda connessa alla rete
Serial.print("\n\nInizializzazione rete WiFi: ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
lcd.print(".");
}

```

```

}

lcd.setCursor(0,0);
Serial.println("\nConnesso.");
lcd.print("Connesso ");
Serial.print("IP del dispositivo: ");
Serial.print(WiFi.localIP());
lcd.setCursor(0,1);
lcd.print(WiFi.localIP());

// Inizializziamo il WebServer.
server.begin();
Serial.println("\nWebServer inizializzato.\n");

Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH); //Inizializza
firebase

}

int i = 0;

//Funzione per la comunicazione tra Nodo Master e Nodo 1
durante l'invio del dato

void comunicazione()
{
Serial.println("Comunicazione");
do
{
if(irrecv.decode(&results))
{
Serial.println("Dato ricevuto:");
Serial.println(results.value);

//Dato ricevuto correttamente
if(results.value == 4369)
{
lcd.setCursor(0,0);
lcd.print(" ACK ");
lcd.setCursor(0,1);
lcd.print(" O K ");
controllo_ricezione = false;
controllo_invio = false;
stato_conn = true;
Serial.println("Dato valido");
delay(10000);
}
//Dato non ricevuto correttamente
}
}

```

```

else
{
controllo_ricezione = false;
stato_conn = false;
Serial.println("Dato non valido");
}

}

delay(1);

Serial.print(i++);
controllo_ricezione = false;

//Attesa di 10000 milli secondi = 10 secondi, dopo di che chiude
la connessione
if(i == 10000)
{
Serial.println("Connessione scaduta");
controllo_ricezione = false;
controllo_invio = false;
stato_conn = false;
}

irrecv.resume(); //Ricevi un altro valore

}while(controllo_ricezione);
Serial.println("Sono uscito... ");

}

int RM = LOW; // Relè nodo master
int RN1 = LOW; // Relè Nodo 1
int AN1 = LOW; // Attuatore Nodo 1
int RN2 = LOW; // Relè Nodo 2
int AN2 = LOW; // Attuatore Nodo 2

//Variabili di controllo per il cambiamento di stato effettuato\
Quando le variabili sono a TRUE il cambiamento è avvenuto
mentre a FALSE il cambiamento potrebbe non essere stato
effettuato

boolean CRN1 = true;
boolean CAN1 = true;
boolean CRN2 = true;
boolean CAN2 = true;

WiFiClient cliente;

String percorso = "NetworkNodes/"; //Stringa che conterrà il
percorso del database firebase

```

```

void firebase_upload(String nodo, String dispositivo, String
stato)
{
    percorso = "NetworkNodes/";
    // rimozione valore precedente
    percorso += nodo;
    percorso += dispositivo;
    Firebase.remove(percorso);

    // settaggio valore attuale
    Firebase.setString(percorso, stato);

    // Errore connessione
    if (Firebase.failed()) {
        Serial.print("setting /message failed:");
        Serial.println(Firebase.error());
        return;
    }

    boolean js = false; //Variabile per far partire o meno la modifica
    della pagina di errore

    boolean analogic_input = false; //Input analogico dato
    all'attuatore o relè dei vari nodi

    void loop()
    {
        i=0; //Settaggio a zero della variabile che conta il tempo di
        attesa

        // Vediamo se esiste una connessione con un client. Se non
        esiste ritorniamo al loop.
        cliente = server.available();
        if (!cliente)
        {
            if(irrecv.decode(&results))
            {
                Serial.println(results.value);
                if(results.value==4112)
                {
                    RN1=HIGH;
                    analogic_input = true;
                }
                if(results.value==257)
                {
            
```

```

RN1=LOW;
analogic_input = true;
}
if(results.value==1118481)
{
AN1=HIGH;
analogic_input = true;
}
if(results.value==1052688)
{
AN1=LOW;
analogic_input = true;
}
irrecv.resume();
}
return;
}

// Nella caso ci sia una connessione con un client, mostriamo un
messaggio di avvenuta connessione
// mantenendo aperta la connessione.
Serial.println("Client connesso: ");
while (!cliente.available())
{
delay(1);
}

String form ="";
if(analogic_input)
{
form = "http://";
form += IP;
}
else
{
//Realizziamo la lettura della form.
form = cliente.readStringUntil('\r');
}

analogic_input = false;

```

```

Serial.println(form);
cliente.flush();

// Controllo della form restituita dall'aggiornamento della
pagina HTML

if(form.indexOf("RelayMaster=ON")!=-1)

{
lcd.setCursor(0,0);
lcd.print("Relay ON ");
lcd.setCursor(0,1);
lcd.print("Attendere... ");
delay(100);
digitalWrite(D4, HIGH); // Accensione Relè nodo master
RM = HIGH;
}

if(form.indexOf("RelayMaster=OFF")!=-1)

{
lcd.setCursor(0,0);
lcd.print("Relay OFF ");
lcd.setCursor(0,1);
lcd.print("Attendere... ");
delay(100);
digitalWrite(D4, LOW); // Spegnimento relè nodo master
RM = LOW;
}

if(form.indexOf("Relay1=ON")!=-1)

{
lcd.setCursor(0,0);
lcd.print("Relay Nodo1 ON ");
lcd.setCursor(0,1);
lcd.print("Attendere... ");
delay(100);
Serial.println("NEC");
do
{
//Controllo tempo di attesa
if(i<1000)
{
irsend.sendNEC(0x0000000001, 32); //Invio dato al nodo 1 per
l'accensione del relè nodo 1
comunicazione();
i++; //Aumenta di +1 il tempo di attesa
}
}

```

```

}

else
{
controllo_invio = false;
}

}while(controllo_invio);

Serial.println("Anch'io...");

//Controllo stato connessione per modificare o meno la pagina
HTML

if(stato_conn)
{
RN1 = HIGH;
CRN1 = true;
}
else
{
RN1 = LOW;
js = true;
CRN1 = false;
}

//Resetto le variabili di controllo per una prossima
comunicazione
stato_conn = true;
controllo_invio=true;
controllo_ricezione=true;
i=0;
}

if(form.indexOf("Relay1=OFF")!=-1)

{
lcd.setCursor(0,0);
lcd.print("Relay Nodo1 OFF ");
lcd.setCursor(0,1);
lcd.print("Attendere... ");
delay(100);
Serial.println("NEC");
do
{
if(i<1000)
{
}
}

```

```

irsend.sendNEC(0x0000000101, 32); // Invio dato al nodo 1 per
lo spegnimento del relè nodo 1

comunicazione();

i++;

}

else

{

controllo_invio = false;

}

}while(controllo_invio);

Serial.println("Anch'io...");

//Controllo stato connessione per modificare o meno la pagina
HTML

if(stato_conn)

{

RN1 = LOW;

CRN1 = true;

}

else

{

RN1 = HIGH;

js = true;

CRN1 = false;

}

//Resetto le variabili di controllo per una prossima
comunicazione

stato_conn = true;

controllo_invio=true;

controllo_ricezione=true;

i=0;

}

if(form.indexOf("Attuatore1=ON")!=-1)

{

lcd.setCursor(0,0);

lcd.print("Attuat. Nodo1 ON");

lcd.setCursor(0,1);

lcd.print("Attendere... ");

delay(100);

Serial.println("NEC");

do

{

if(i<1000)

```

```

{

irsend.sendNEC(0x000000011, 32); //Invio dato al nodo 1 per
l'accensione dell'attuatore nodo 1

comunicazione();

i++;

}

else

{

controllo_invio = false;

}

}while(controllo_invio);

Serial.println("Anch'io...");

//Controllo stato connessione per modificare o meno la pagina
HTML

if(stato_conn)

{

AN1 = HIGH;

CAN1 = true;

}

else

{

AN1 = LOW;

js = true;

CAN1 = false;

}

//Resetto le variabili di controllo per una prossima
comunicazione

stato_conn = true;

controllo_invio=true;

controllo_ricezione=true;

i=0;

}

if(form.indexOf("Attuatore1=OFF")!=-1)

{

lcd.setCursor(0,0);

lcd.print("Attua. Nodo1 OFF");

lcd.setCursor(0,1);

lcd.print("Attendere... ");

delay(100);

Serial.println("NEC");

do

```

```

{
if(i<1000)
{
irsend.sendNEC(0x0000000010, 32); // Invio dato al nodo 1 per
lo spegnimento dell'attuatore nodo 1

comunicazione();

i++;
}

else
{
controllo_invio = false;
}

}while(controllo_invio);

Serial.println("Anch'io...");

//Controllo stato connessione per modificare o meno la pagina
HTML

if(stato_conn)
{
AN1 = LOW;
CAN1 = true;
}

else
{
AN1 = HIGH;
js = true;
CAN1 = false;
}

//Resetto le variabili di controllo per una prossima
comunicazione

stato_conn = true;
controllo_invio=true;
controllo_ricezione=true;
i=0;
}

if(form.indexOf("Relay2=ON")!=-1)
{
lcd.setCursor(0,0);
lcd.print("Relay Nodo2 ON ");
lcd.setCursor(0,1);
lcd.print("Attendere... ");
delay(100);
}

```

```

int invio = 0;

do
{
irsend.sendNEC(0x0000011000, 32);
invio++;
}while(invio<10);
RN2 = HIGH;
}

if(form.indexOf("Relay2=OFF")!=-1)
{
lcd.setCursor(0,0);
lcd.print("Relay Nodo2 OFF ");
lcd.setCursor(0,1);
lcd.print("Attendere... ");
delay(100);
int invio = 0;
do
{
irsend.sendNEC(0x0000010000, 32);
invio++;
}while(invio<10);
RN2 = LOW;
}

if(form.indexOf("Attuatore2=ON")!=-1)
{
lcd.setCursor(0,0);
lcd.print("Attuat. Nodo2 ON");
lcd.setCursor(0,1);
lcd.print("Attendere... ");
delay(100);
int invio = 0;
do
{
irsend.sendNEC(0x0001111000, 32);
invio++;
}while(invio<10);
AN2 = HIGH;
}

```

```

if(form.indexOf("Attuatore2=OFF")!=-1)
{
lcd.setCursor(0,0);
lcd.print("Attua. Nodo2 OFF");
lcd.setCursor(0,1);
lcd.print("Attendere... ");
delay(100);
int invio = 0;
do
{
irsend.sendNEC(0x0011100000, 32);
invio++;
}while(invio<10);
AN2 = LOW;
}

```

```

}

pag_html(); //Pagina HTML
js = false;

lcd.setCursor(0,0);
lcd.print("IP device: ");
lcd.setCursor(0,1);
lcd.print(WiFi.localIP());

// 
delay(1);

Serial.println("Client Disconnesso.\n");

}

```

## Nodo 1:

```

//-----CALCOLATORI ELETTRONICI E RETE DI
CALCOLATORI-----//


//-----PIETRO RIGNANESE & ANDREA POLENTA 2017-----
-----//


//Librerie per gestione display
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

//Libreria per comunicazione IR
#include <IRremote.h>

int RECV_IR = 8;
IRrecv irrecv(RECV_IR); //Ricezione del segnale IR sul pin 8
decode_results results;
IRsend irsend;

LiquidCrystal_I2C lcd(0x27,20,4);

int cont = 0; //Contatore per il conteggio dei secondi una volta
ricevuto il dato dal Nodo Master

int num_ricezioni = 1; //Conteggio delle ricezioni

//Mappatura pin Arduino
int relay = 2;
int attuatore = 4;

```

```

boolean dato_corrotto = true; //Flag per il controllo del dato
corrotto

void setup()
{
//Display
lcd.init();
lcd.backlight();
lcd.home();
lcd.print("Benvenuto!");

Serial.println("Benvenuto");

//Inizializzazione pin arduino
pinMode(relay,OUTPUT);
pinMode(attuatore, OUTPUT);

Serial.begin(115200);

irrecv.enableIRIn(); // Comincia la trasmissione del segnale
}

void loop()
{

//Controlla se è stato ricevuto un dato sul sensore IR
if (irrecv.decode(&results))

```

```
{
dato_corrotto = true;
Serial.println("");
Serial.println("");
Serial.println("Fase ");
Serial.print(num_ricezioni);
Serial.println("");
Serial.println("Dato in arrivo...\"");
Serial.println("");

num_ricezioni++;

lcd.setCursor(0,0);
lcd.print("Comunicazione:");
lcd.setCursor(0,1);

Serial.println(results.value); //Stampa su terminale il valore del segnale ricevuto

//-----Controllo del dato-----

//Accensione Relay
if(results.value == 1)
{
Serial.println("Accensione Relay");

int cont = 0;
lcd.print("R.ON ");
digitalWrite(relay,HIGH);
delay(20);
do
{
cont++;
irsend.sendNEC(0x0000001111, 32);// invio del dato per la conferma della purezza del dato = 4369
delay(20);
}while(cont<80);

//Reinstanziamento della ricezione
irrecv.enableIRIn();
irrecv.decode(&results);

dato_corrotto = false; //Pongo dato corrotto a false in modo da non richiedere l'invio del dato al Nodo Master
}
}

//Spagnimento Relay
}
```

```

if(results.value == 257)
{
Serial.println("Segnimento Relay");

lcd.print("R.OFF ");
digitalWrite(relay,LOW);
cont = 0;

do
{
cont++;
irsend.sendNEC(0x1000001111, 32);// invio del dato per la conferma della purezza del dato
delay(20);
}while(cont<80);

//Reinstanziamento della ricezione
irrecv.enableIRIn();
irrecv.decode(&results);

dato_corrotto = false; //Pongo dato corrotto a false in modo da non richiedere l'invio del dato al Nodo Master
}

//Accensione Attuatore
if(results.value == 17)
{
Serial.println("Accensione Attuatore");

lcd.setCursor(5,1);
lcd.print("A.ON ");
digitalWrite(attuatore,HIGH);
cont = 0;

do
{
cont++;
irsend.sendNEC(0x1000001111, 32);// invio del dato per la conferma della purezza del dato
delay(20);
}while(cont<80);

//Reinstanziamento della ricezione
irrecv.enableIRIn();
irrecv.decode(&results);
}
```

```

dato_corrotto = false; //Pongo dato corrotto a false in modo da
non richiedere l'invio del dato al Nodo Master

}

//Spegimento Attuatore
if(results.value == 16)

{
Serial.println("Spegimento Attuatore");

lcd.setCursor(5,1);
lcd.print("A.OFF");
digitalWrite(attuatore,LOW);
cont = 0;

do
{
cont++;

irsend.sendNEC(0x1000001111, 32); // invio del dato per la
conferma della purezza del dato

delay(20);

}while(cont<80);

//Reinstanziamento della ricezione

```

```

irrecv.enableIRIn();

irrecv.decode(&results);

dato_corrotto = false; //Pongo dato corrotto a false in modo da
non richiedere l'invio del dato al Nodo Master

}

//Richiedo l'invio del dato al Nodo Master, perchè il dato è
corrotto!

if(dato_corrotto)

{
irsend.sendNEC(0x0000000111, 32);
Serial.println("Messaggio corrotto!");

//Reinstanzio la ricezione
irrecv.enableIRIn();
irrecv.decode(&results);
}

irrecv.resume(); // Ricezione del prossimo valore
}
}
```

*N.B: Questi due programmi sono per la comunicazione fra Nodo Master e Nodo 1. Se dovessimo collegare un altro nodo(come faremo più in là) i due programmi cambieranno(cambierà solo il programma del Nodo 1).*

Con il lancio di questi due programmi otteniamo il risultato voluto:

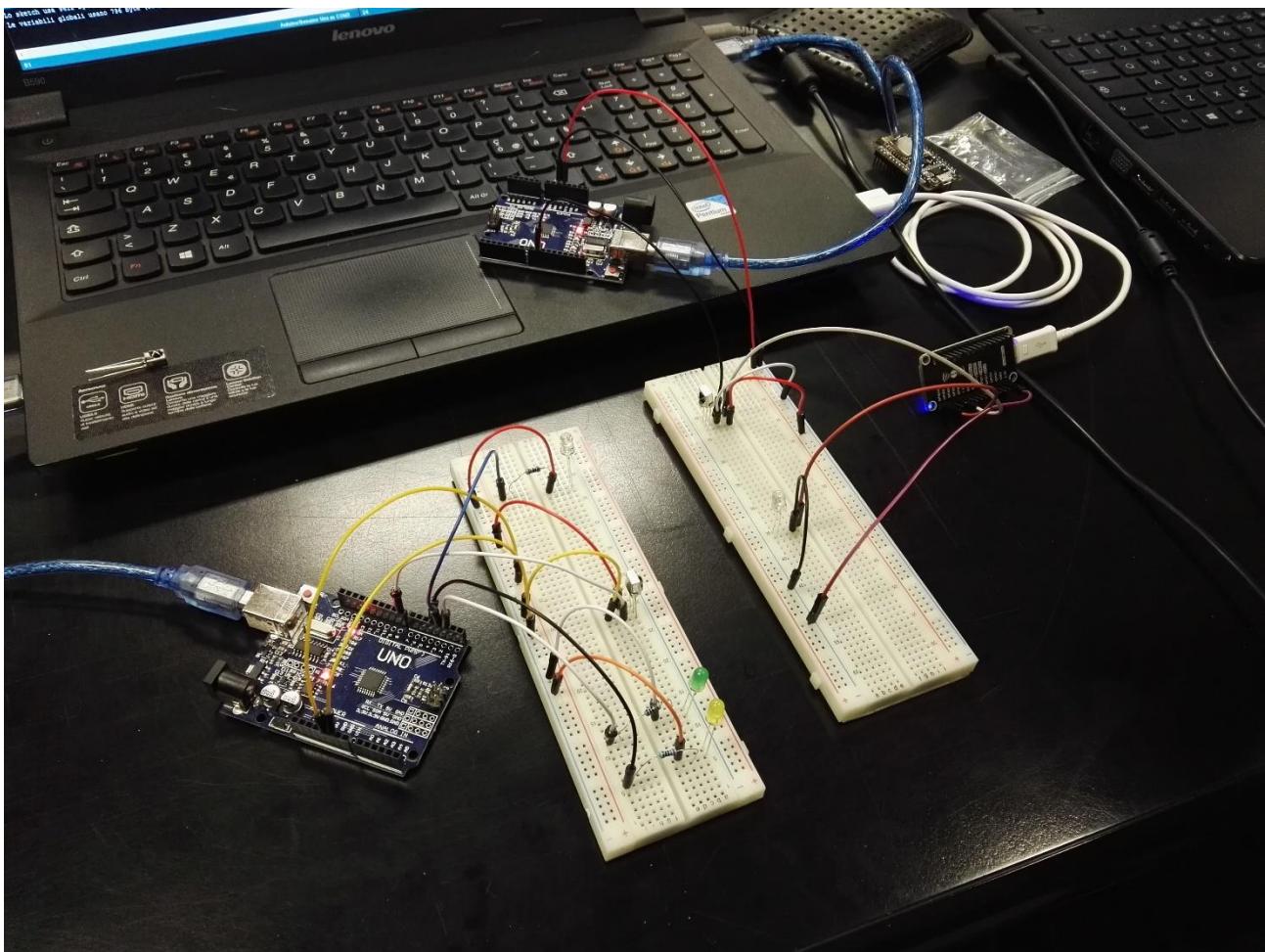
- Il client si connette all'indirizzo IP della scheda wi-fi;
- Decide di cambiare lo stato di un attuatore e/o relè sul Nodo 1;
- Il Nodo Master riceve il dato inviato e lo elabora inviandolo al Nodo 1;
- Il Nodo 1 elabora il dato ricevuto; in questa fase il Nodo Master rimane in ascolto per 10 secondi;
- Il Nodo 1, se accetta il dato, manda un ACK al Nodo Master;
- Il Nodo Master riceve l'ok e chiude la comunicazione aggiornando la pagina HTML che visualizzerà il client.

Il compito del Nodo Master è quello di un server web e di un "Ponte di comunicazione" tra il terminale e gli altri nodi.

*N.B: Potrebbe capitare che:*

- *La comunicazione sia interrotta da un ostacolo: in questo caso il Nodo Master invia per 10 secondi lo stesso dato fino a che non riceverà conferma. Se non dovesse ricevere conferma chiude comunicazione e la pagina HTML non viene aggiornata*
- *La comunicazione viene interrotta solo dopo che il Nodo I ha ricevuto il dato corretto (in questo caso non potrà comunicare l'ok): Potremmo risolvere questo problema mandando in ritorno una pagina con un WARNING di comunicazione attraverso un javascript. Purtroppo non possiamo prevenire ostacoli durante la comunicazione, ma su 100 comunicazione provate, solamente un 10%, non ha ricevuto l'ACK dal Nodo II. Mandando il WARNING il client sa che qualcosa non è andato nel modo giusto, perché potrebbe risultare anche un guasto a qualche componente di comunicazione. Questo problema potrebbe essere in parte gestito... Si è pensato di mandare, attraverso il Nodo Master, lo stato precedente del dato che non è stato ricevuto; così facendo si ha la sicurezza, in parte, di far tornare il Nodo I allo stato precedente. Questo è possibile se e solo se la comunicazione è interrotta solo nella ricezione della risposta e non nell'invio del dato!*

*PS: Durante la fase di ascolto del Nodo Master, il Nodo I manda un ACK per circa 10 secondi; nei dieci secondi di ascolto, da parte del Nodo Master, vengono prelevati valori random non inviati dal Nodo I. Fortunatamente, nell'arco dei 10 secondi di ascolto, il dato, inviato dal Nodo I, viene letto dal Nodo Master ricevendo l'ok di avvenuta ricezione.*



Come dicevamo in precedenza: il terzo arduino serve solo per alimentare il ricevitore del Nodo Master e in secondo a prelevare i valori del sensore, solamente per essere testati.

Il ricevitore IR, del Nodo Master, ci ha dato parecchi problemi:

- I valori rilevati erano falsati (riceveva valori anche quando non venivano inviati segnali infrarossi)
- L'alimentazione della shield Wi-Fi dava parecchi problemi al sensore durante la lettura

*Come si è risolto tutto ciò?*

Si è utilizzata, prima di tutto, un'alimentazione esterna, presa da un altro Arduino UNO (l'alimentazione può essere presa anche da una pila esterna), risolvendo in parte i problemi di lettura del sensore; in questo modo parte del primo problema è stata risolta, ma tuttavia continua, in modo random, a ricevere valori casuali. Non si è venuto a capo di questo!!! Si è cercato di risolvere la cosa nel migliore dei modi: Potrebbe essere un problema della scheda, un problema del sensore, un problema di alimentazione... Potrebbe essere tutto o niente...

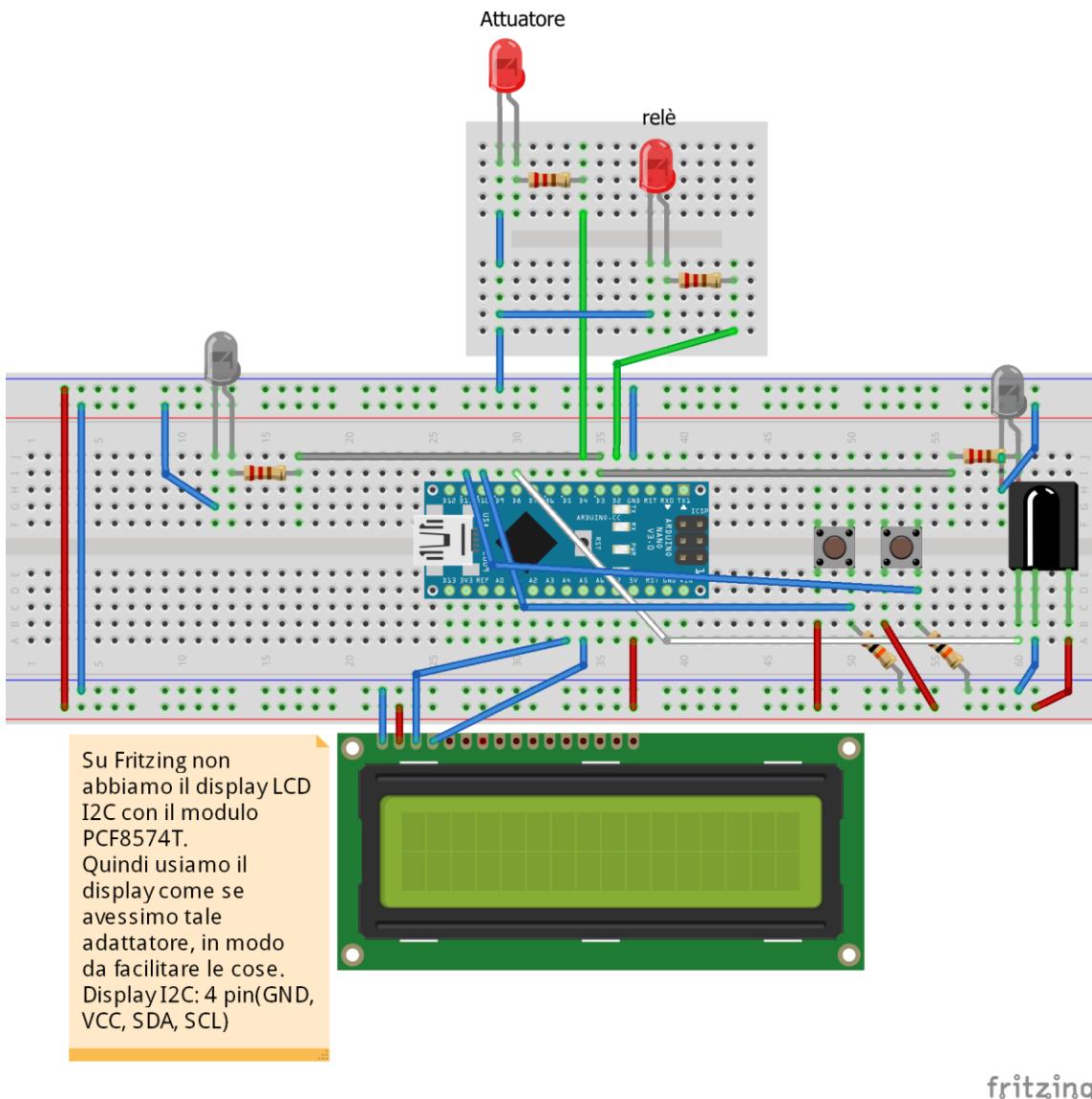
*N.B: potrebbe capitare che il dato inviato dal Nodo 1 non venga letto dal sensore del Nodo Master, ma su 100 prove non è mai successo! Il dato prima o poi verrà letto!*

### Terza fase: Collegamento Ottico Nodo Master -> Nodo 1 -> Nodo 2

Per questa terza fase implementiamo due prototipi: il primo avrà una semplice comunicazione senza risposta del Nodo 2 (cioè senza verificare la ricezione del dato), mentre il secondo sarà quello più completo e strutturato in cui anche il Nodo 2 interagisce attraverso l'architettura "Connected Oriented" sfruttata già dagli altri due nodi.

#### Primo prototipo: Simple Connection

- Lo schema per il Nodo Master rimane lo stesso, senza nessuna variazione neanche nel codice (essendo già tutta implementata)
- Lo schema del Nodo 1 subisce piccolissime variazioni (un'aggiunta di un solo LED infrarossi e due pulsanti per l'accensione dei dispositivi):



fritzing

Si è semplicemente aggiunto un altro diodo led infrarossi per inviare il segnale al Nodo 2. In questo caso abbiamo il LED IR aggiuntivo collegato allo stesso piedino del primo LED IR(in modo da sfruttare la libreria per l'invio del segnale IR, questo perchè, come dicevamo, la libreria IRremote utilizza, per default, il PIN3 di arduino). Il codice sorgente è leggermente diverso rispetto al primo.

```
-----CALCOLATORI ELETTRONICI E
RETE DI CALCOLATORI-----
--//  
  
-----PIETRO RIGNANESE & ANDREA
POLENTA 2017-----  
---//  
  
//Librerie per gestione display
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

```
//Libreria per comunicazione IR
#include <IRremote.h>  
  
int RECV_IR = 8;  
  
IRrecv irrecv(RECV_IR); //Ricezione del segnale IR sul  
pin 8  
  
decode_results results;  
  
IRsend irsend;  
  
LiquidCrystal_I2C lcd(0x27,20,4);
```

```

int cont = 0; //Contatore per il conteggio dei secondi
una volta ricevuto il dato dal Nodo Master

int num_ricezioni = 1; //Conteggio delle ricezioni

//Mappatura pin Arduino

int relay = 2;

int attuatore = 4;

//Controllo dei dispositivi del Nodo 1 quando queste
vengono attivate in modo analogico dal nodo stesso

boolean R = true;

boolean A = true;

boolean dato_corrotto = true; //Flag per il controllo
del dato corrotto

void setup()
{
pinMode(10,INPUT); //Bottone Relay
pinMode(11,INPUT); //Bottone Attuatore

//Display
lcd.init();
lcd.backlight();
lcd.home();
lcd.print("Nodo 1");

//Inizializzazione pin arduino
pinMode(relay,OUTPUT);
pinMode(attuatore, OUTPUT);

Serial.begin(115200);
Serial.println("Nodo 1");
Serial.println("");

irrecv.enableIRIn(); // Comincia la trasmissione del
segnale
}

//Funzione di risposta
void ack()
{
cont = 0;
delay(20);
do
{
  cont++;
}

```

```

  irsend.sendNEC(0x1000001111, 32); // invio del dato
per la conferma della purezza del dato = 4369

  delay(20);

}while(cont<80);

//Reinstanziamento della ricezione
irrecv.enableIRIn();
irrecv.decode(&results);

dato_corrotto = false; //Pongo dato corrotto a false
in modo da non richiedere l'invio del dato al Nodo
Master

}

void loop()
{
//Controllo dello stato analogico dei dispositivi
if(digitalRead(10)==HIGH)
{
if(R)
{
digitalWrite(relay,HIGH);
R = false;
irsend.sendNEC(0x1000001010, 32); // invio del dato
per la conferma della purezza del dato = 4369
delay(20);
irrecv.enableIRIn();
irrecv.decode(&results);
Serial.println("ON Relay");
delay(200);
}
else
{
digitalWrite(relay,LOW);
R = true;
irsend.sendNEC(0x1000000101, 32); // invio del dato
per la conferma della purezza del dato = 4369
delay(20);
irrecv.enableIRIn();
irrecv.decode(&results);
Serial.println("OFF Relay");
delay(200);
}
}

```

```

}

if(digitalRead(11)==HIGH)
{
if(A)
{
digitalWrite(attuatore,HIGH);

A = false;

irsend.sendNEC(0x1000111111, 32); // invio del dato
per la conferma della purezza del dato = 4369

delay(20);

irrecv.enableIRIn();

irrecv.decode(&results);

Serial.println("ON Attua");

delay(200);

}

else

{
digitalWrite(attuatore,LOW);

A = true;

irsend.sendNEC(0x1000101010, 32); // invio del dato
per la conferma della purezza del dato = 4369

delay(20);

irrecv.enableIRIn();

irrecv.decode(&results);

Serial.println("OFF Attua");

delay(200);

}
}

//Controlla se è stato ricevuto un dato sul sensore IR

if (irrecv.decode(&results))
{
dato_corrotto = true;

Serial.println("");
Serial.println("");
Serial.println("Fase ");
Serial.print(num_ricezioni);
Serial.println("");
Serial.println("Dato in arrivo...");

Serial.println("");

num_ricezioni++;
}

```

```

lcd.setCursor(0,0);
lcd.print("Comunicazione:");

lcd.setCursor(0,1);

Serial.println(results.value); //Stampa su terminale il
valore del segnale ricevuto

//-----Controllo del dato-----

//Accensione Relay

if(results.value == 1)
{
Serial.println("Accensione Relay");

lcd.setCursor(0,1);
lcd.print("Relay ON ");
digitalWrite(relay,HIGH);

R = false;

ack();
}

//Spegnimento Relay

if(results.value == 257)
{
Serial.println("Segnimento Relay");
lcd.setCursor(0,1);
lcd.print("Relay OFF ");
digitalWrite(relay,LOW);

R = true;

ack();
}

dato_corrotto = false; //Pongo dato corrotto a false
in modo da non richiedere l'invio del dato al Nodo
Master

}

//Accensione Attuatore

if(results.value == 17)
{
Serial.println("Accensione Attuatore");
lcd.setCursor(0,1);
lcd.print("Attuatore ON ");
digitalWrite(attuatore,HIGH);

A = false;
}

```



```
ack();  
}  
  
//Spegimento Attuatore  
if(results.value == 16)  
{  
Serial.println("Spegimento Attuatore");  
lcd.setCursor(0,1);  
lcd.print("Attuatore OFF ");  
digitalWrite(attuatore,LOW);  
A = true;  
ack();  
}  
  
//Accensione Relay nodo 2  
if(results.value == 69632)  
{  
irsend.sendNEC(0x0000011000, 32);  
Serial.println("Accensione Relay 2");  
lcd.setCursor(0,1);  
lcd.print("Relay Nodo2 ON ");  
  
//Reinstanziamento della ricezione  
irrecv.enableIRIn();  
irrecv.decode(&results);  
  
dato_corrotto = false; //Pongo dato corrotto a false  
in modo da non richiedere l'invio del dato al Nodo  
Master  
}  
  
//Accensione Attuatore nodo 2  
if(results.value == 17895424)  
{  
irsend.sendNEC(0x0001111000, 32);  
Serial.println("Accensione Attuatore 2");  
lcd.setCursor(0,1);  
lcd.print("Attua. Nodo2 ON ");  
  
//Reinstanziamento della ricezione  
irrecv.enableIRIn();  
irrecv.decode(&results);  
  
dato_corrotto = false; //Pongo dato corrotto a false  
in modo da non richiedere l'invio del dato al Nodo  
Master
```

```

//Spegnimento Relay nodo 2

if(results.value == 65536)
{
    irsend.sendNEC(0x0000010000, 32);
    Serial.println("Spegnimento Relay 2");
    lcd.setCursor(0,1);
    lcd.print("Relay Nodo2 OFF ");

    //Reinstanziamento della ricezione
    irrecv.enableIRIn();
    irrecv.decode(&results);

    dato_corrotto = false; //Pongo dato corrotto a false
    in modo da non richiedere l'invio del dato al Nodo
    Master
}

//Spegnimento Attuatore nodo 2

if(results.value == 286261248)
{
    irsend.sendNEC(0x0011100000, 32);
    Serial.println("Spegnimento Attuatore 2");
    lcd.setCursor(0,1);
    lcd.print("Attua. Nodo2 OFF");

    //Reinstanziamento della ricezione
    irrecv.enableIRIn();
    irrecv.decode(&results);

    dato_corrotto = false; //Pongo dato corrotto a false
    in modo da non richiedere l'invio del dato al Nodo
    Master
}

//Richiedo l'invio del dato al Nodo Master, perchè il
dato è corrotto!

if(dato_corrotto)
{
    irsend.sendNEC(0x0000000111, 32);
    Serial.println("Messaggio corrotto!");

    //Reinstanzio la ricezione
    irrecv.enableIRIn();
    irrecv.decode(&results);
}

```

```

    }

irrecv.resume(); // Ricezione del prossimo valore
}

```

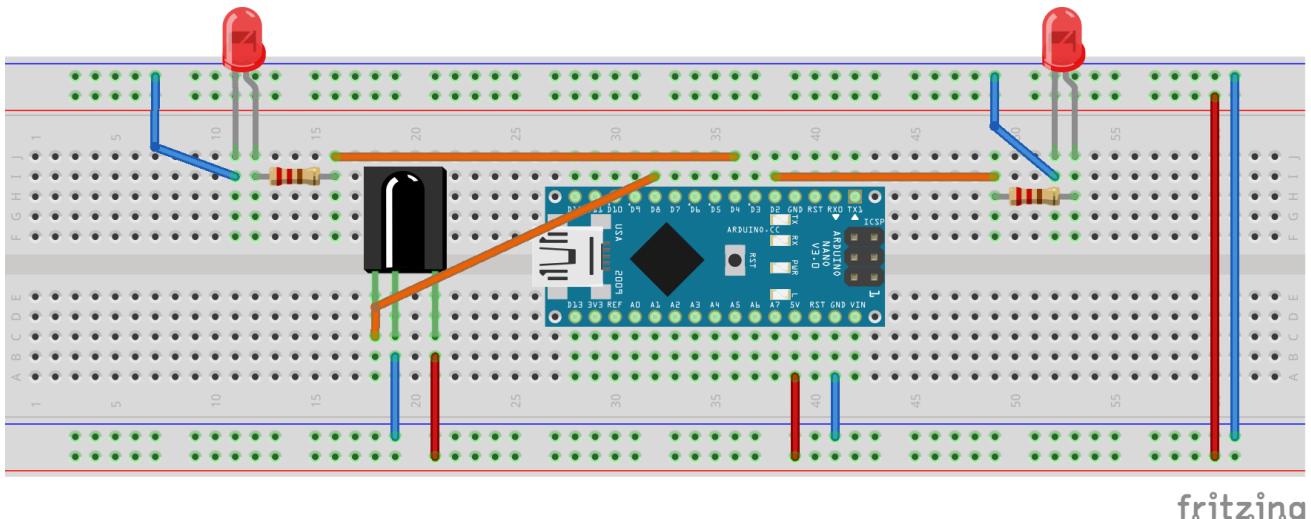
```

    }

```

Si sono aggiunti due pulsanti per controllare, in modo analogico, lo stato dei dispositivi collegati al Nodo 1. Una volta avvenuto il cambiamento viene comunicato lo stato al Nodo Master.

- Lo schema del Nodo 2 è il seguente:



I collegamenti:

Arduino NANO	
PIN 2	LED(Relay)
PIN 4	LED(Attuatore)
PIN 8	Ricevitore IR
PIN A4	Display SDA
PIN A5	Display SCL

Lo sketch da far eseguire a questa scheda è il seguente:

```

//-----NETWORK-
NODES-----


//-----PIETRO RIGNANESE
& ANDREA POLENTA-----


//Librerie per gestione display
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

//Libreria IR
#include <IRremote.h>

int RECV_IR = 8;
IRrecv irrecv(RECV_IR); //Ricezione del segnale IR sul pin 8
decode_results results;

//Mappatura pin Arduino
int relay = 2;
int attuatore = 4;

LiquidCrystal_I2C lcd(0x27,20,4);

void setup()
{
//Display
lcd.init();
lcd.backlight();
lcd.home();
lcd.print("Nodo 2");

Serial.begin(115200);
Serial.println("Nodo 2");
Serial.println("");


pinMode(relay, OUTPUT);
pinMode(attuatore, OUTPUT);

digitalWrite(relay,HIGH);
digitalWrite(attuatore,HIGH);
delay(2000);
digitalWrite(relay,LOW);
digitalWrite(attuatore,LOW);

irrecv.enableIRIn();
}

void loop()
{

```

```

lcd.setCursor(0,0);
lcd.print("Nodo 2 ");
lcd.print(" ");

//Controlla se è stato ricevuto un dato sul sensore IR
if (irrecv.decode(&results))
{
lcd.setCursor(0,0);
lcd.print("Comunicazione:");
lcd.setCursor(0,1);
Serial.println(results.value);
//Accensione Relay nodo 2
if(results.value == 69632)
{
lcd.print("Relay ON ");
Serial.println("Accensione Realy");
digitalWrite(relay,HIGH);
}

//Accensione Attuatore nodo 2
if(results.value == 17895424)
{
lcd.print("Attua. ON ");
Serial.println("Accensione Attuatore");
digitalWrite(attuatore, HIGH);
}

//Spegnimento Relay nodo 2
if(results.value == 65536)
{
lcd.print("Relay OFF ");
Serial.println("Spegnimento Realy");
digitalWrite(relay, LOW);
}

//Spegnimento Attuatore nodo 2
if(results.value == 286261248)
{
lcd.print("Attua. OFF");
Serial.println("Spegnimento Attuatore");
digitalWrite(attuatore, LOW);
}

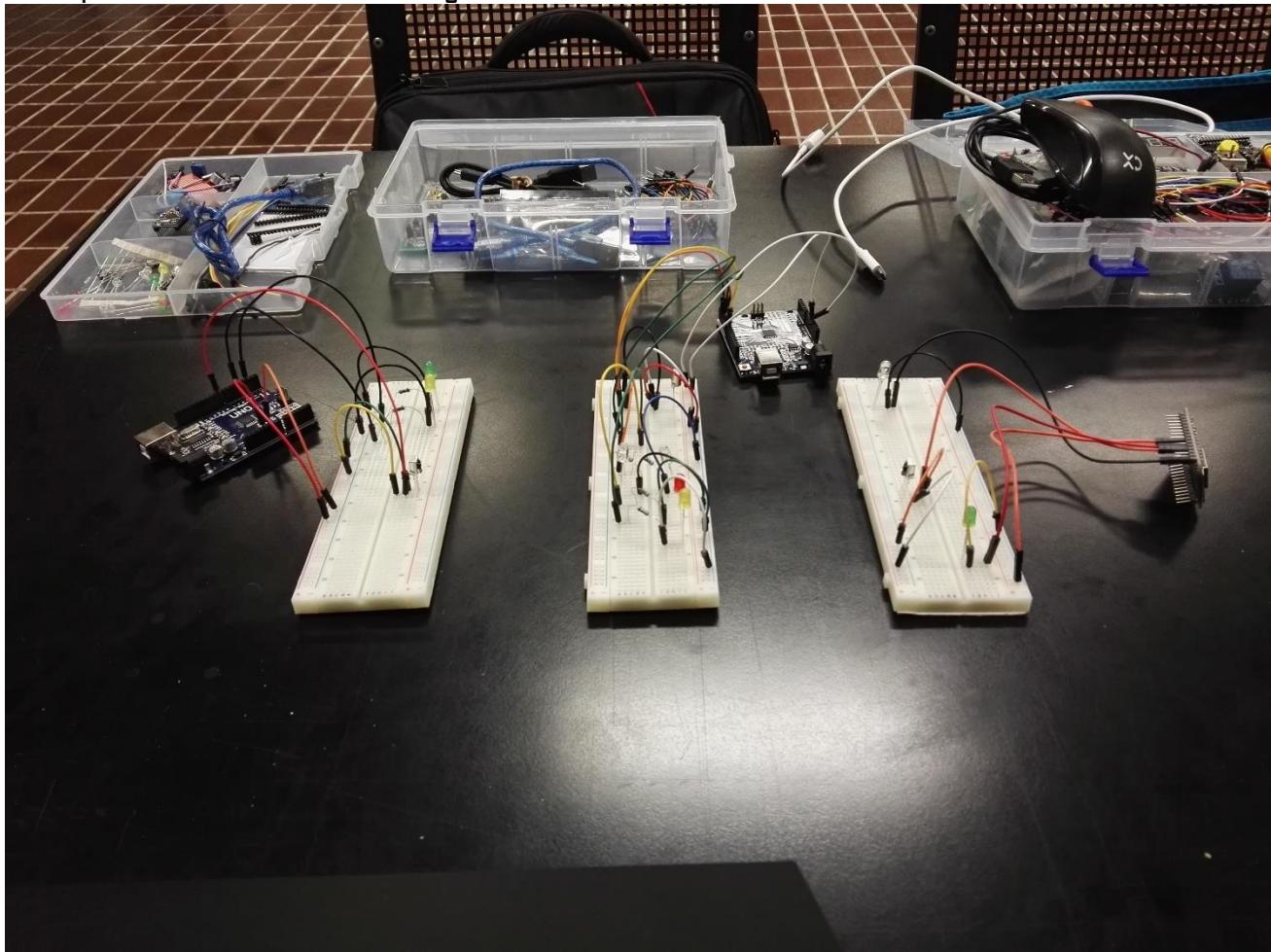
irrecv.resume();

```

}

}

La disposizione dei tre nodi è il seguente:



Da destra verso sinistra: Nodo Master (con modulo ESP8266), Nodo 1 (con Arduino UNO), Nodo 2 (con Arduino UNO).

*N.B: In questo caso la comunicazione, per attivare attuatori e relè sul Nodo 2 è del tipo semplice... cioè senza ack. Non si è ancora implementata una Connected Oriented tra Nodo 1 e Nodo 2.*

Secondo prototipo: [Connected Oriented](#)

Per il secondo prototipo si devono apportare delle piccole modifiche:

- Modificare lo sketch del Nodo Master;
- Modificare lo schema e lo sketch del Nodo 1;

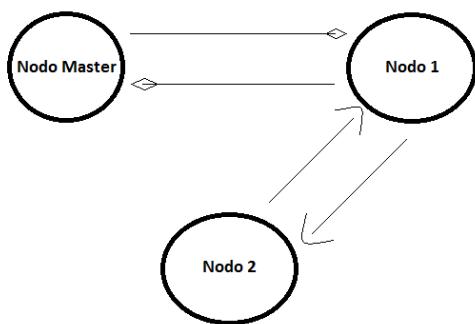
- Modificare lo schema e lo sketch del Nodo 2.

Per il primo prototipo non era richiesta la risposta dell'avvenuta ricezione del dato da parte del Nodo 2, mentre in questo prototipo vogliamo instaurare un "dialogo" tra Nodo 1 e Nodo 2, così come quello che avviene tra gli altri due nodi. Per far ciò dobbiamo collegare un altro sensore infrarosso, questa volta rivolto verso il Nodo 2, in modo da captare le frequenze infrarosse di quest'ultimo. Questo purtroppo non è possibile! Arduino non riesce a gestire due sensori IR perché nella parte di programmazione non è possibile creare due oggetti(o istanze) di due diversi sensori con la stessa classe! Questo descritto è un problema in ambito di programmazione ma si è pensato a delle possibili soluzioni:

- Un sensore IR sferico(360°) in modo da poter coprire, interamente, entrambi i nodi; risulterebbero dei problemi nella ricezione, perchè riceverà da entrambi i versi, ma logicamente potrebbe funzionare;
- Una serie di specchi che converge il fascio del segnale infrarossi del Nodo 2 verso il sensore del Nodo 1; potrebbe sembrare fantascienza ma potremmo usare lo stesso principio dei telescopi;
- Per ultimo, ma non meno importante, la "triangolazione".

Cos'è la "triangolazione"?

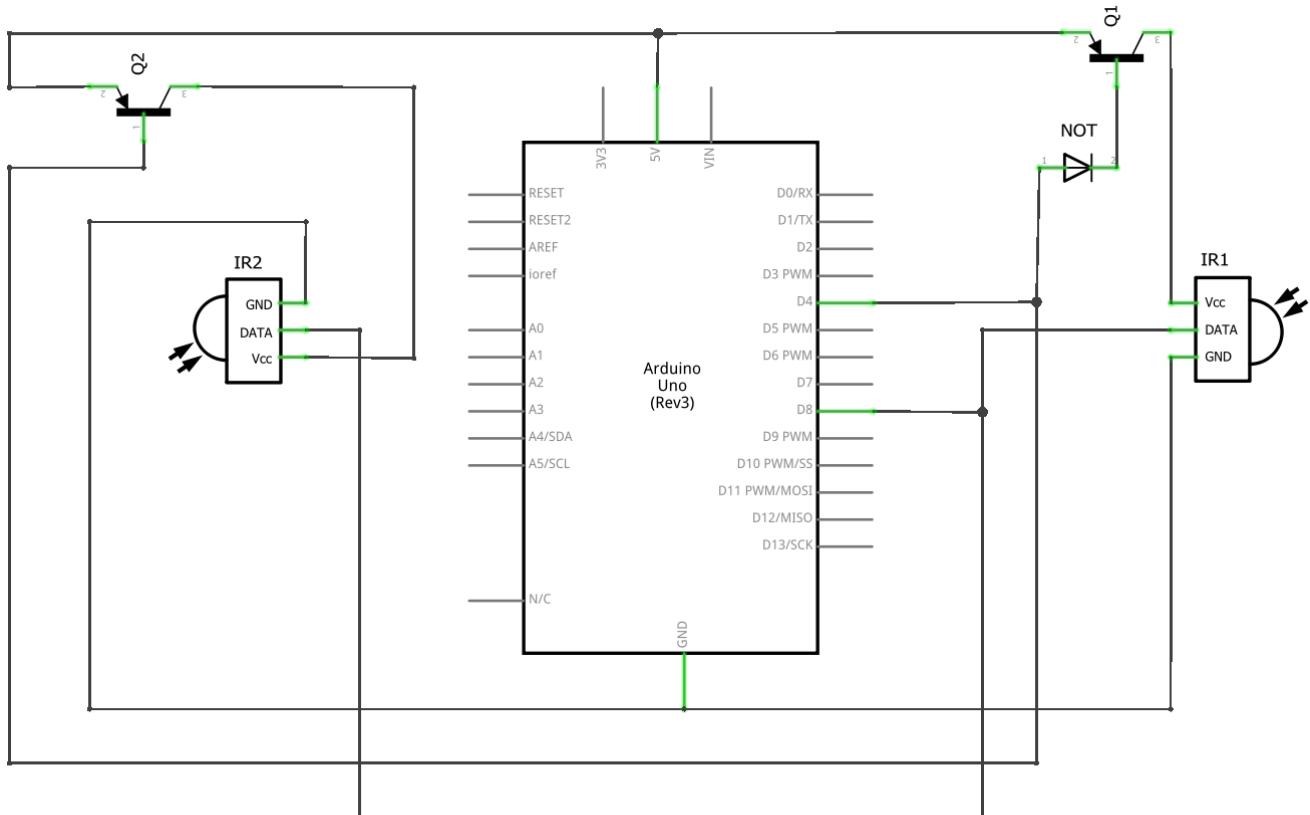
Con triangolazione intendiamo la disposizione dei nodi a formare un triangolo. Come in questa figura:



In questo modo i nodi sono disposti in modo tale da poter inviare al Nodo 1 i segnali infrarossi.

Logicamente è un modo corretto da realizzare, ma facendo alcune prove e test si è verificato che questa comunicazione non può avvenire! Il sensore del Nodo 1 è soggetto a troppe informazioni e non riesce a captare nel modo corretto il segnale di ritorno del Nodo 2. Si è pensato ad un altro tipo di soluzione, da aggiungere alle altre elencate poco fa:

- Gestire il cambio di sensore attraverso un circuito switch a transistor: due transistor per alimentare due diversi circuiti, in modo alternato, in base alla ricezione del dato (*soluzione hardware*).



**fritzing**

N.B: Questo potrebbe essere realizzato anche con un semplice relè! Basta collegare uno dei due sensori all'ingresso "Normalmente chiuso" e l'altro al "Normalmente aperto" del relè... In modo che solo uno dei due sensori è in funzione!

Si potrebbe progettare un "Secondo prototipo" con tutti i metodi risolutivi elencati e constatare quanti e quali di questi metodi funzionino... In questo caso realizzeremo l'ultimo metodo risolutivo:

*Risoluzione con "Switch a transistor":*

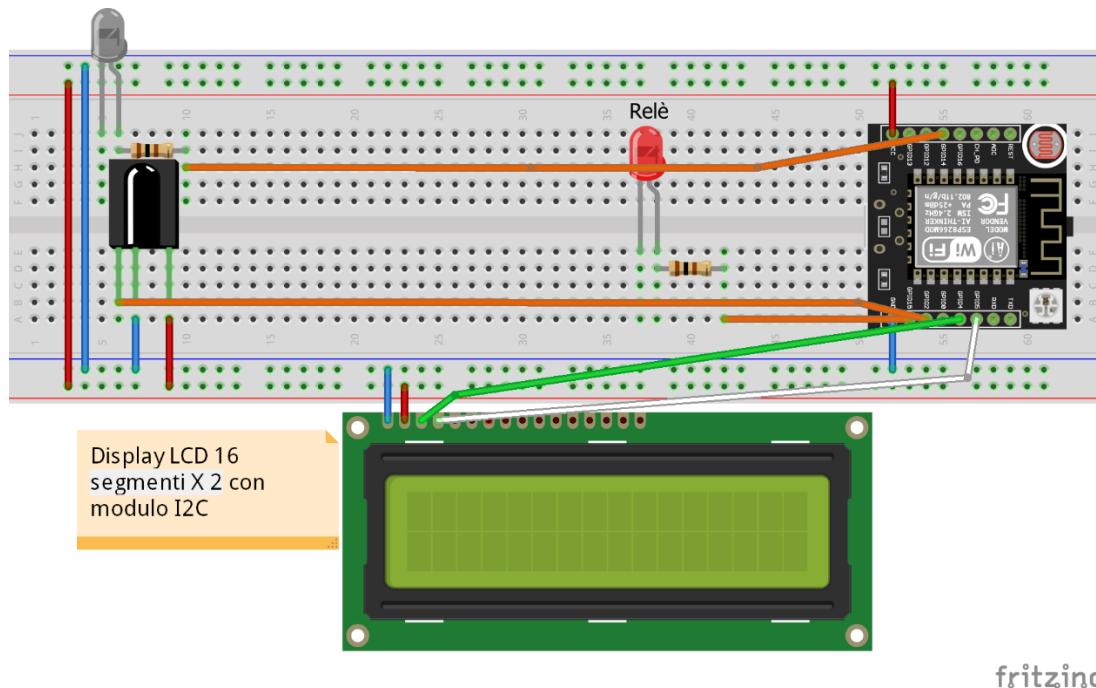
Per quanto questo metodo possa essere logicamente corretto purtroppo non può essere realizzato! Perchè non è possibile collegare, allo stesso pin di Arduino, i due pin dei sensori che trasportano il segnale... Quando uno dei due sensori è interdetto l'altro sensore non percepisce nessun dato... Come se il circuito fosse aperto!

Risoluzione con "Sensore Sferico" su Nodo 1:

Per questo tipo di risoluzione non abbiamo utilizzato un sensore sferico, ma ne abbiamo utilizzati due direzionali: uno rivolto verso il Nodo Master e l'altro verso il Nodo 2. In questo modo si ha una copertura totale della ricezione dei segnali provenienti da entrambe le direzioni.

Ecco i tre nodi completi:

Nodo Master



Connessioni:

ESP8266	
PIN D4	Relè
PIN D0	Emettitore LED IR
PIN D3	Ricevitore IR
PIN D1	Display SDA
PIN D2	Display SCL

Sketch:

```

// Inclusione dell'apposita libreria per la Shield adoperata
#include <ESP8266WiFi.h>
// Libreria di gestione infrarossi per la shield esp8266
#include <IRremoteESP8266.h>

//Libreria per database esterno (Firebase)
#include <FirebaseArduino.h>
//Credenziali di accesso al database
#define FIREBASE_HOST "sentinel-83e9f.firebaseio.com"
#define FIREBASE_AUTH "W7F3oHDG85MSrNDXiDnrCKbNCEBnrRfBDSSfful"

//Librerie per gestione display
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

//Definizione delle credenziali per la connessione per accedere al Router
const char* ssid = "TIM-67955105";
const char* password = "XDDaz2lzqaGGIQCfl97krEA";

//Variabile contenente l'indirizzo IP assegnato alla shield ESP-I2E
String IP = "";

// Definiamo la mappatura della Shield ESP8266(Datasheet)
#define D0 16 // Trasmettitore IR
#define D1 5 // Display
#define D2 4 // LCD
#define D3 0 // Ricevitore IR
#define D4 2 // Relè
#define D5 14
#define D6 12
#define D7 13
#define D8 15
#define D9 3 // RX
#define D10 1 // TX

//Istanziamento le variabili di Controllo Invio/Ricezione
boolean controllo_invio = true;
boolean controllo_ricezione = true;

//Istanziamento un flag per verificare lo stato della connessione
//Verifichiamo se la connessione è la ricezione del messaggio di ACK è avvenuta
senza problemi
boolean stato_conn = true;

// Istanziamento 'server(80)' come oggetto della classe WiFiServer.
WiFiServer server(80);

//Istanziamento display
LiquidCrystal_I2C lcd(0x27,20,4);

IRsend irsend(D0); //Instanziamo un oggetto per la trasmissione dell'informazione
tramite IR al pin D0
IRrecv irrecv(D3); //Instanziamo oggetto per ricezione segnale infrarosso
decode_results results;

void setup()
{
  //Display
  lcd.init();
  lcd.backlight();

  // Instanziamo la comunicazione infrarossi
  irsend.begin();
  irrecv.enableIRIn();

  // Connessione seriale a 115200.
  Serial.begin(115200);
  delay(10);

  // Inizializziamo i dispositivi connessi al nodo allo stato di LOW
  pinMode(D4, OUTPUT);
  digitalWrite(D4, LOW);

  Serial.println("-----NODE MASTER-----");
  Serial.println("");
}

```

```

//Stampiamo il benvenuto sul display appena la scheda sarà accesa
lcd.home();
lcd.print("Benvenuto!");
delay(2500);
lcd.setCursor(0, 0);
lcd.print("Inizializzazione");
lcd.setCursor(0,1);

//Connessione alla rete WiFi
//Mostriamo l'IP della scheda connessa alla rete
Serial.print("\n\nInizializzazione rete WiFi:");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
  lcd.print(".");
}

lcd.setCursor(0,0);
Serial.println("\nConnesso!");
lcd.print("Connesso    ");
Serial.print("IP del dispositivo: ");
Serial.print(WiFi.localIP());
lcd.setCursor(0,1);
lcd.print(WiFi.localIP());

// Inizializziamo il WebServer.
server.begin();
Serial.println("\nWebServer inizializzato.\n");

Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH); //Inizializza firebase
}

//Variabile contatore per verificare il tempo di attesa dell'invio del messaggio fino
alla ricezione dell'ACK
int i = 0;

//Funzione per la comunicazione tra Nodo Master e Nodo I durante l'invio del dato
void comunicazione()
{
  Serial.println("Comunicazione");
  do
  {
    //Viene controllato, attraverso il sensore IR, se viene ricevuto un dato
    if(irrecv.decode(&results))
    {
      //Ricezione avvenuta!
      Serial.println("Dato ricevuto:");
      Serial.println(results.value);

      //Dato ricevuto correttamente
      if(results.value == 4369)
      {
        lcd.setCursor(0,0);
        lcd.print(" ACK   ");
        lcd.setCursor(0,1);
        lcd.print("OK   ");
        controllo_ricezione = false;
        controllo_invio = false;
        stato_conn = true;
        Serial.println("Dato valido");
        delay(1000);
      }
      //Dato non ricevuto correttamente
      else
      {
        controllo_ricezione = false;
        stato_conn = false;
        Serial.println("Dato non valido");
      }
    }
    delay(1);
    Serial.print(i++);
    controllo_ricezione = false;
  }
}

```

```

//Attesa di 10000 milis secondi = 10 secondi, dopo di che chiude la connessione
if(i == 10000)
{
    Serial.println("Connessione scaduta");
    controllo_ricezione = false;
    controllo_invio = false;
    stato_conn = false;
}
irrecv.resume(); //Ricevi un altro valore

}while(controllo_ricezione);
Serial.println("Sono uscito... ");

}

int RM = LOW; // Relè nodo master
int RNI = LOW; // Relè Nodo I
int ANI = LOW; // Attuatore Nodo I
int RN2 = LOW; // Relè Nodo 2
int AN2 = LOW; // Attuatore Nodo 2

//Variabili di controllo per il cambiamento di stato effettuato
//Quando le variabili sono a TRUE il cambiamento è avvenuto mentre a FALSE il
cambiamento potrebbe non essere stato effettuato
boolean CRNI = true;
boolean CANI = true;
boolean CRN2 = true;
boolean CAN2 = true;

WiFiClient cliente;

String percorso = "NetworkNodes/"; //Stringa che conterrà il percorso del
database firebase

//Funzione che aggiorna il database Firebase
void firebase_upload(String nodo, String dispositivo, String stato)
{
    percorso = "NetworkNodes/";
    // rimozione valore precedente
    percorso += nodo;
    percorso += dispositivo;
    Firebase.remove(percorso);

    // settaggio valore attuale
    Firebase.setString(percorso, stato);
    // Errore connessione
    if (Firebase.failed()) {
        Serial.print("setting /message failed:");
        Serial.println(Firebase.error());
        return;
    }
}

boolean js = false; //Variabile per far partire o meno la modifica della pagina di
errore
//Input analogico dei vari dispositivi connessi al nodo(Input a pulsante)
boolean analogic_input = false;

void loop()
{
    i=0; //Settaggio a zero della variabile che conta il tempo di attesa

    // Vediamo se esiste una connessione con un client. Se non esiste ritorniamo al
loop.
    cliente = server.available();
    if (!cliente)
    {
        //Se la connessione al client non è presente, il Nodo Master è in ascolto verso il
        Nodo I per la ricezione
        //del dato analogico attraverso pulsante
        if(irrecv.decode(&results))
        {
            Serial.println(results.value);
            if(results.value==4112)

```

```

{
    RNI=HIGH;
    analogic_input = true;
}
if(results.value==257)
{
    RNI=LOW;
    analogic_input = true;
}
if(results.value==1118481)
{
    ANI=HIGH;
    analogic_input = true;
}
if(results.value==1052688)
{
    ANI=LOW;
    analogic_input = true;
}
irrecv.resume();

}
return;
}

// Nella caso ci sia una connessione con un client, mostriamo un messaggio di
avvenuta connessione
// mantenendo aperta la connessione.
Serial.println("Client connesso: ");
while (!cliente.available())
{
    delay(1);
}

String form = ""; //Stringa contenente l'indirizzo IP, e eventuale form, del Nodo
Master

//Controllo per vedere se l'input è avvenuto per via analogica o tramite device
if(analogic_input)
{
    form = "http://";
    form += IP;
}
else
{
    //Realizziamo la lettura della form.
    form = cliente.readStringUntil('\r');
}

analogic_input = false; //Resetiamo la variabile del controllo dell'input analogico

Serial.println(form);
cliente.flush();

// Controllo della form restituita dall'aggiornamento della pagina HTML
if(form.indexOf("RelayMaster=ON")!=-1)
{
    lcd.setCursor(0,0);
    lcd.print("Relay ON   ");
    lcd.setCursor(0,1);
    lcd.print("Attendere...   ");
    delay(100);
    digitalWrite(D4, HIGH); // Accensione Relè nodo master
    RM = HIGH;
}
if(form.indexOf("RelayMaster=OFF")!=-1)
{
    lcd.setCursor(0,0);
    lcd.print("Relay OFF   ");
    lcd.setCursor(0,1);
    lcd.print("Attendere...   ");
    delay(100);
    digitalWrite(D4, LOW); // Spegnimento relè nodo master
}

```

```

RM = LOW;
}

//-----NODO I-----
-----

if(form.indexOf("Relay=ON")!=-1)
{
lcd.setCursor(0,0);
lcd.print("Relay Nodal ON ");
lcd.setCursor(0,1);
lcd.print("Attendere... ");
delay(100);
Serial.println("ACCENSIONE RELAY NODO I");
do
{
//Controllo tempo di attesa
if(i<1000)
{
  irsend.sendNEC(0x0000000001, 32); //Invio dato al nodo I per l'accensione
  del relè nodo I
  comunicazione();
  i++; //Aumenta di +1 il tempo di attesa
}
else
{
  controllo_invio = false;
}
}while(controllo_invio);

Serial.println("Anch'io...");
//Controllo stato connessione per modificare o meno la pagina HTML
if(statoconn)
{
  RNI = HIGH;
  CRNI = true;
}
else
{
  RNI = LOW;
  js = true;
  CRNI = false;
}
//Resetto le variabili di controllo per una prossima comunicazione
statoconn = true;
controllo_invio=true;
controllo_ricezione=true;
i=0;
}

if(form.indexOf("Relay=OFF")!=-1)
{
lcd.setCursor(0,0);
lcd.print("Relay Nodal OFF ");
lcd.setCursor(0,1);
lcd.print("Attendere... ");
delay(100);
Serial.println("SPEGNIMENTO RELAY NODO I");
do
{
if(i<1000)
{
  irsend.sendNEC(0x0000000010, 32); // Invio dato al nodo I per lo
  spegnimento del relè nodo I
  comunicazione();
  i++;
}
else
{
  controllo_invio = false;
}
}while(controllo_invio);

Serial.println("Anch'io...");
//Controllo stato connessione per modificare o meno la pagina HTML
if(statoconn)

```

```

{
  RNI = LOW;
  CRNI = true;
}
else
{
  RNI = HIGH;
  js = true;
  CRNI = false;
}
//Resetto le variabili di controllo per una prossima comunicazione
statoconn = true;
controllo_invio=true;
controllo_ricezione=true;
i=0;
}

if(form.indexOf("AttuatoreI=ON")!=-1)
{
lcd.setCursor(0,0);
lcd.print("Attuat. Nodal ON");
lcd.setCursor(0,1);
lcd.print("Attendere... ");
delay(100);
Serial.println("ACCENSIONE ATTUATORE NODO 2");
do
{
if(i<1000)
{
  irsend.sendNEC(0x0000000011, 32); //Invio dato al nodo I per l'accensione
  dell'attuatore nodo I
  comunicazione();
  i++;
}
else
{
  controllo_invio = false;
}
}while(controllo_invio);

Serial.println("Anch'io...");
//Controllo stato connessione per modificare o meno la pagina HTML
if(statoconn)
{
  ANI = HIGH;
  CANI = true;
}
else
{
  ANI = LOW;
  js = true;
  CANI = false;
}
//Resetto le variabili di controllo per una prossima comunicazione
statoconn = true;
controllo_invio=true;
controllo_ricezione=true;
i=0;
}

if(form.indexOf("AttuatoreI=OFF")!=-1)
{
lcd.setCursor(0,0);
lcd.print("Attua. Nodal OFF");
lcd.setCursor(0,1);
lcd.print("Attendere... ");
delay(100);
Serial.println("SPEGNIMENTO ATTUATORE NODO I");
do
{
if(i<1000)
{
  irsend.sendNEC(0x0000000010, 32); // Invio dato al nodo I per lo
  spegnimento dell'attuatore nodo I
  comunicazione();
}
}
}

```

```

        i++;
    }
    else
    {
        controllo_invio = false;
    }
}while(controllo_invio);

Serial.println("Anch'io... ");
//Controllo stato connessione per modificare o meno la pagina HTML
if(stato_conn)
{
    ANI = LOW;
    CANI = true;
}
else
{
    ANI = HIGH;
    js = true;
    CANI = false;
}
//Resetto le variabili di controllo per una prossima comunicazione
stato_conn = true;
controllo_invio=true;
controllo_ricezione=true;
i=0;
}
//-----
-----NODO 2-----
if(form.indexOf("Relay2=ON")!=-1)
{int conta = 0;
Serial.println("ACCENSIONE RELAY NODO 2");
do
{
    if(i<1000)
    {
        if(conta == 15)
        {
            irsend.sendNEC(0x0000001000, 32); // Invio dato al nodo 1 per lo spegnimento dell'attuatore nodo 1
            conta = 0;
        }
        comunicazione();
        i++;
        conta++;
    }
    else
    {
        controllo_invio = false;
    }
}while(controllo_invio);

Serial.println("Anch'io... ");
//Controllo stato connessione per modificare o meno la pagina HTML
if(stato_conn)
{
    RN2 = HIGH;
    CANI = true;
}
else
{
    RN2 = LOW;
    js = true;
    CRN2 = false;
}
//Resetto le variabili di controllo per una prossima comunicazione
stato_conn = true;
controllo_invio=true;
controllo_ricezione=true;
i=0;
}

```

```

}

if(form.indexOf("Relay2=OFF")!=-1)
{int conta = 0;
Serial.println("SPEGNIMENTO RELAY NODO 2");
do
{
    if(i<1000)
    {
        if(conta == 15)
        {
            irsend.sendNEC(0x0000001000, 32); // Invio dato al nodo 1 per lo spegnimento dell'attuatore nodo 1
            conta = 0;
        }
        comunicazione();
        i++;
        conta++;
    }
    else
    {
        controllo_invio = false;
    }
}while(controllo_invio);

Serial.println("Anch'io... ");
//Controllo stato connessione per modificare o meno la pagina HTML
if(stato_conn)
{
    RN2 = LOW;
    CANI = true;
}
else
{
    RN2 = HIGH;
    js = true;
    CRN2 = false;
}
//Resetto le variabili di controllo per una prossima comunicazione
stato_conn = true;
controllo_invio=true;
controllo_ricezione=true;
i=0;

}

if(form.indexOf("Attuatore2=ON")!=-1)
{int conta = 0;
Serial.println("ACCENSIONE ATTUATORE NODO 2");
do
{
    if(i<1000)
    {
        if(conta == 15)
        {
            irsend.sendNEC(0x0000001000, 32); // Invio dato al nodo 1 per lo spegnimento dell'attuatore nodo 1
            conta = 0;
        }
        comunicazione();
        i++;
        conta++;
        delay(100);
    }
    else
    {
        controllo_invio = false;
    }
}
```

```

}while(controllo_invio);

Serial.println("Anch'io... ");
//Controllo stato connessione per modificare o meno la pagina HTML
if(stato_conn)
{
  AN2 = HIGH;
  CAN2 = true;
}
else
{
  AN2 = LOW;
  js = true;
  CAN2 = false;
}
//Resetto le variabili di controllo per una prossima comunicazione
stato_conn = true;
controllo_invio=true;
controllo_ricezione=true;
i=0;

}

if(form.indexOf("Attuatore2=OFF")!=-1)
{int conta = 0;
Serial.println("SPEGNIMENTO ATTUATORE NODO 2");
do
{

  if(i<1000)
  {
    if(conta == 15)
    {
      irsend.sendNEC(0x00110000, 32); // Invio dato al nodo 1 per lo
spennimento dell'attuatore nodo 1
      conta = 0;
    }

    comunicazione();
    i++;
    conta++;
  }
  else
}

```

```

{
  controllo_invio = false;
}
}while(controllo_invio);

Serial.println("Anch'io... ");
//Controllo stato connessione per modificare o meno la pagina HTML
if(stato_conn)
{
  AN2 = LOW;
  CAN1 = true;
}
else
{
  AN2 = HIGH;
  js = true;
  CAN2 = false;
}
//Resetto le variabili di controllo per una prossima comunicazione
stato_conn = true;
controllo_invio=true;
controllo_ricezione=true;
i=0;

}

//-----
-----
-----

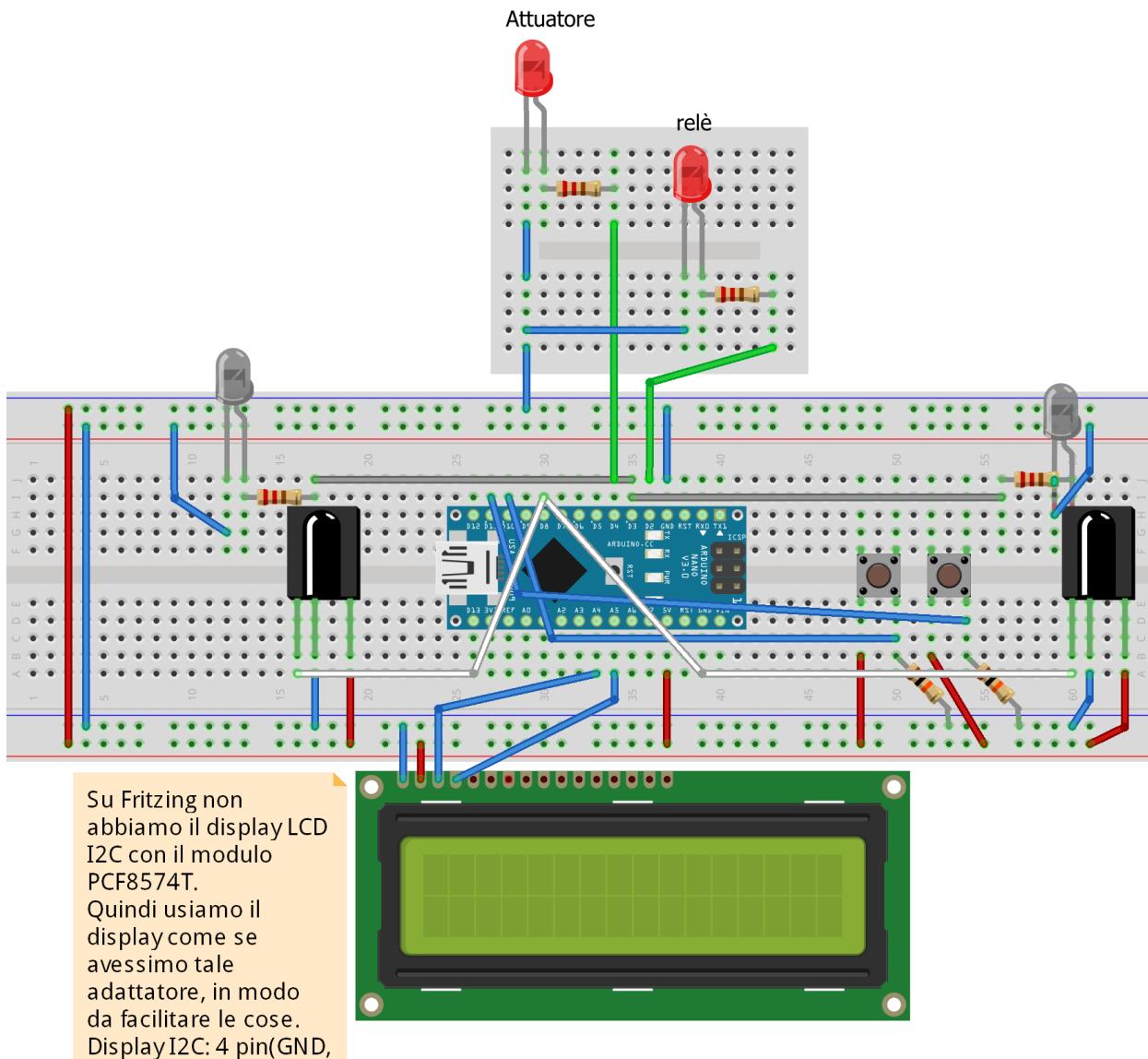

pag_html(); //Pagina HTML
js = false;

lcd.setCursor(0,0);
lcd.print("IP device:   ");
lcd.setCursor(0,1);
lcd.print(WiFi.localIP());

//
delay(1);
Serial.println("Client Disconnesso.\n");
}

```

## Nodo 1



fritzing

## Connessioni:

Arduino UNO/NANO	
PIN 2	Relè
PIN 3	Emettitore LED IR/LED IR2
PIN 4	Attuatore
PIN 8	Ricevitore IR/Ricev. IR2
PIN A4	Display SDA

PIN A5	Display SCL
PIN 10	Pulsante Relay
PIN 11	Pulsante Attuatore
5v	Breadboard
GND	Breadboard

Sketch:

```
//Libreria per gestione display
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

//Libreria per comunicazione IR
#include <IRremote.h>
int RECV_IR = 8;
IRrecv irrecv(RECV_IR); //Ricezione del segnale IR sul pin 8
decode_results results;
IRsend irsend;

LiquidCrystal_I2C lcd(0x27,20,4);

int cont = 0; //Contatore per il conteggio dei secondi una volta ricevuto il dato dal Nodo Master
int num_ricezioni = 1; //Conteggio delle ricezioni

//Mappatura pin Arduino
int relay = 2;
int attuatore = 4;

//Variabili per controllare il pilotaggio dei pulsanti
boolean R = true;
boolean A = true;

boolean dato_corrotto = true; //Flag per il controllo del dato corrotto

void setup()
{
  pinMode(10,INPUT); //Pulsante 1 = Relay
  pinMode(11,INPUT); //Pulsante 2 = Attuatore

  //Display
  lcd.init();
  lcd.backlight();
  lcd.home();
  lcd.print("Nodo 1");

  //Inizializzazione pin arduino
  pinMode(relay,OUTPUT);
  pinMode(attuatore, OUTPUT);

  Serial.begin(115200);
  Serial.println("-----Nodo 1-----");
  Serial.println("");
}

irrecv.enableIRIn(); // Comincia la trasmissione del segnale
```

```
//Funzione di risposta
void ack()
{
  cont = 0;
  delay(20);
  do
```

```
{
  cont++;
  irsend.sendNEC(0x1000000000, 32); // invio del dato per la conferma della purezza del dato = 4369
  delay(20);
}while(cont<80);

//Reinstanziamento della ricezione
irrecv.enableIRIn();
irrecv.decode(&results);

dato_corrotto = false; //Pongo dato corrotto a false in modo da non richiedere l'invio del dato al Nodo Master
}

//Funzione per avvenuta ricezione del dato da parte del Nodo 2
boolean comunicazione()
{
  Serial.println("Sensore 2");
  int sec = 0;
  do
  {
    if (irrecv.decode(&results))
    {
      Serial.println(results.value);
      if(results.value == 4369)
      {
        Serial.println("OK");
        return true;
      }
      delay(1);
      sec++;
      irrecv.resume();
    }
  }while(sec<100);
  return false;
}

void loop()
{
  //-----CONTROLLO ANALOGICO DEL NODO-----
  -

  if(digitalRead(10)==HIGH)
  {
    if(R)
    {
      digitalWrite(relay,HIGH);
      R = false;
      irsend.sendNEC(0x1000000000, 32); // invio del dato per la conferma della purezza del dato = 4369
      delay(20);
      irrecv.enableIRIn();
      irrecv.decode(&results);
    }
  }
}
```

```

Serial.println("ON Relay");
delay(200);
}
else
{
  digitalWrite(relay,LOW);
  R = true;
  irsend.sendNEC(0x1000000001, 32); // invio del dato per la conferma della
purezza del dato = 4369
  delay(20);
  irrecv.enableIRIn();
  irrecv.decode(&results);
  Serial.println("OFF Relay");
  delay(200);
}

}

if(digitalRead(II)==HIGH)
{
  if(A)
  {
    digitalWrite(attuatore,HIGH);
    A = false;
    irsend.sendNEC(0x1000000001, 32); // invio del dato per la conferma della purezza
del dato = 4369
    delay(20);
    irrecv.enableIRIn();
    irrecv.decode(&results);
    Serial.println("ON Attua");
    delay(200);
  }
  else
  {
    digitalWrite(attuatore,LOW);
    A = true;
    irsend.sendNEC(0x1000000001, 32); // invio del dato per la conferma della
purezza del dato = 4369
    delay(20);
    irrecv.enableIRIn();
    irrecv.decode(&results);
    Serial.println("OFF Attua");
    delay(200);
  }
}

//-----CONTROLLO AUTOMATIZZATO DEL NODO-----
//Controlla se è stato ricevuto un dato sul sensore IR
if(irrecv.decode(&results))
{
  dato_corrotto = true;
  Serial.println("");
  Serial.println("");
  Serial.println("Fase ");
  Serial.print(num_ricezioni);
  Serial.println("");
  Serial.println("Dato in arrivo...");
  Serial.println("");

  num_ricezioni++;

  lcd.setCursor(0,0);
  lcd.print("Comunicazione:");
  lcd.setCursor(0,1);

  Serial.println(results.value); //Stampa su terminale il valore del segnale ricevuto
}

//-----Controllo del dato-----

//Accensione Relay
if(results.value == 1)
{
  Serial.println("Accensione Relay");
}

```

```

lcd.setCursor(0,1);
lcd.print("Relay ON ");
digitalWrite(relay,HIGH);
R = false;
ack();
}

//Spegimento Relay
if(results.value == 257)
{
  Serial.println("Spegimento Relay");
  lcd.setCursor(0,1);
  lcd.print("Relay OFF ");
  digitalWrite(relay,LOW);
  R = true;
  ack();

  dato_corrotto = false; //Pongo dato corrotto a false in modo da non richiedere
l'invio del dato al Nodo Master
}

//Accensione Attuatore
if(results.value == 17)
{
  Serial.println("Accensione Attuatore");
  lcd.setCursor(0,1);
  lcd.print("Attuatore ON ");
  digitalWrite(attuatore,HIGH);
  A = false;
  ack();
}

//Spegimento Attuatore
if(results.value == 16)
{
  Serial.println("Spegimento Attuatore");
  lcd.setCursor(0,1);
  lcd.print("Attuatore OFF ");
  digitalWrite(attuatore,LOW);
  A = true;
  ack();
}

//Accensione Relay nodo 2
if(results.value == 69632)
{
  irsend.sendNEC(0x0000000000, 32);
  Serial.println("Accensione Relay 2");

  //Reinstanziamento della ricezione
  irrecv.enableIRIn();
  irrecv.decode(&results);
  if(comunicazione())
  {
    ack();
  }

  dato_corrotto = false; //Pongo dato corrotto a false in modo da non richiedere
l'invio del dato al Nodo Master
}

//Accensione Attuatore nodo 2
if(results.value == 17895424)
{
  irsend.sendNEC(0x0000000000, 32);
  Serial.println("Accensione Attuatore 2");

  //Reinstanziamento della ricezione
  irrecv.enableIRIn();
  irrecv.decode(&results);
}

```

```

if(comunicazione())
{
    ack();
}

dato_corrotto = false; //Pongo dato corrotto a false in modo da non richiedere
l'invio del dato al Nodo Master

}

//Spegimento Relay nodo 2
if(results.value == 65536)
{
    irsend.sendNEC(0x0000010000, 32);
    Serial.println("Spegimento Relay 2");

    //Reinstanziamento della ricezione
    irrecv.enableIRIn();
    irrecv.decode(&results);
    if(comunicazione())
    {
        ack();
    }

    dato_corrotto = false; //Pongo dato corrotto a false in modo da non richiedere
    l'invio del dato al Nodo Master

}

//Spegimento Attuatore nodo 2
if(results.value == 286261248)
{
    irsend.sendNEC(0x0011000000, 32);
}

```

```

Serial.println("Spegimento Attuatore 2");

//Reinstanziamento della ricezione
irrecv.enableIRIn();
irrecv.decode(&results);
if(comunicazione())
{
    ack();
}

dato_corrotto = false; //Pongo dato corrotto a false in modo da non richiedere
l'invio del dato al Nodo Master

}

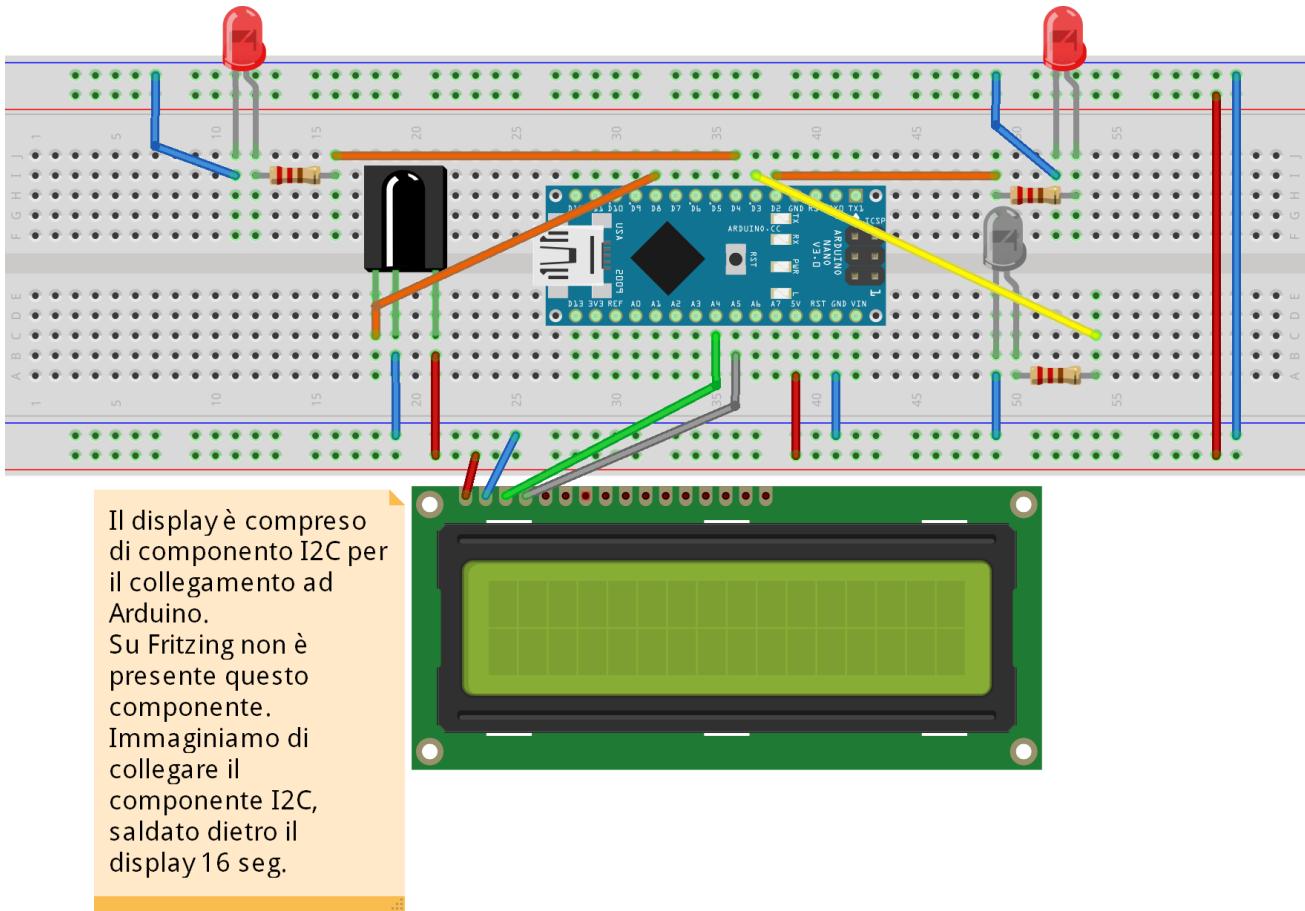
//Richiedo l'invio del dato al Nodo Master, perchè il dato è corrotto!
if(dato_corrotto)
{
    irsend.sendNEC(0x000000001111, 32);
    Serial.println("Messaggio corrotto!");

    //Reinstanzio la ricezione
    irrecv.enableIRIn();
    irrecv.decode(&results);
}

irrecv.resume(); // Ricezione del prossimo valore
}

```

## Nodo 2



fritzing

## Connessioni:

Arduino UNO/NANO	
PIN 2	Relè
PIN 3	Emettitore LED IR
PIN 4	Attuatore
PIN 8	Ricevitore IR
PIN A4	Display SDA
PIN A5	Display SCL
PIN 10	Pulsante Relay
PIN 11	Pulsante Attuatore

5v	Breadboard
GND	Breadboard

## Sketch:

```
//Librerie per gestione display
```

```
#include <Wire.h>
```

```
#include <LiquidCrystal_I2C.h>
```

```
//Libreria IR
```

```
#include <IRremote.h>
```

```
int RECV_IR = 8;
```

```
IRrecv irrecv(RECV_IR); //Ricezione del segnale IR sul pin 8
```

```
decode_results results;
```

```
IRsend irsend;
```

```
//Mappatura pin Arduino
```

```
int relay = 2;
```

```
int attuatore = 4;
```

```
LiquidCrystal_I2C lcd(0x27,20,4);
```

```
void setup()
```

```
{
```

```
//Display
```

```
lcd.init();
```

```
lcd.backlight();
```

```
lcd.home();
```

```
lcd.print("Node 2");
```

```
Serial.begin(115200);
```

```
Serial.println("Node 2");
```

```
Serial.println("");
```

```
pinMode(relay, OUTPUT);
```

```
pinMode(attuatore, OUTPUT);
```

```
digitalWrite(relay,HIGH);
```

```
digitalWrite(attuatore,HIGH);
```

```
delay(2000);
```

```
digitalWrite(relay,LOW);
```

```
digitalWrite(attuatore,LOW);
```

```
irrecv.enableIRIn();
```

```
}
```

```
void loop()
```

```
{
```

```
lcd.setCursor(0,0);
```

```
lcd.print("Node 2      ");
```

```
lcd.print("      ");
```

```
//Controlla se è stato ricevuto un dato sul sensore IR
```

```
if (irrecv.decode(&results))
```

```
{
```

```
int cont = 0;
```

```

lcd.setCursor(0,0);
lcd.print("Comunicazione:");
lcd.setCursor(0,1);
Serial.println(results.value);

//Accensione Relay nodo 2

if(results.value == 69632)
{
    Serial.println("Accensione Realy");
    digitalWrite(relay,HIGH);
    do
    {
        cont++;
        irsend.sendNEC(0x1000000000, 32); // invio del dato per la conferma della purezza del dato = 4369
        Serial.println("Invio OK");
        delay(20);
    }while(cont<80);
}

//Accensione Attuatore nodo 2

if(results.value == 17895424)
{
    Serial.println("Accensione Attuatore");
    digitalWrite(attuatore, HIGH);
    do
    {
        cont++;
        irsend.sendNEC(0x1000000000, 32); // invio del dato per la conferma della purezza del dato = 4369
        Serial.println("Invio OK");
    }while(cont<80);
}

```

```

delay(20);
}while(cont<80);
}

//Spegimento Relay nodo 2

if(results.value == 65536)
{
    Serial.println("Spegimento Realy");
    digitalWrite(relay, LOW);
    do
    {
        cont++;
        irsend.sendNEC(0x1000000000, 32); // invio del dato per la conferma della purezza del dato = 4369
        delay(20);
    }while(cont<80);
}

//Spegimento Attuatore nodo 2

if(results.value == 286261248)
{
    Serial.println("Spegimento Attuatore");
    digitalWrite(attuatore, LOW);
    do
    {
        cont++;
        irsend.sendNEC(0x1000000000, 32); // invio del dato per la conferma della purezza del dato = 4369
        delay(20);
    }while(cont<80);
}

```

```

}
irrecv.enableIRIn();
irrecv.decode(&results);
irrecv.resume();
}
}
}
}
```

```

}
}
}
}
```

La differenza, rispetto al primo prototipo, non è solo nello schema fisico, ma anche nella parte di codice per il controllo di ogni singolo nodo. Infatti gli sketch presentano forti variazioni rispetto agli sketch precedenti.

Attraverso questi collegamenti e gli sketch da inserire all'interno di ogni scheda sarà possibile realizzare una comunicazione "Domanda-Risposta" per la comunicazione dei dati. Infatti:

- Il client invierà la sua richiesta di modifica tramite la pagina HTML;
- Il Nodo Master riceverà il comando decodificando l'URL della pagina ed invierà le azioni da seguire al Nodo 1
- Il Nodo 1 elaborerà il dato inviato:
  - Esegue l'azione se fosse il destinatario (accensione/spegnimento attuatore e/o relè)
    - Restituisce l'ok di avvenuta lettura al Nodo Master
    - Il Nodo Master riceve l'ACK e aggiorna la pagina
  - Invia dato al Nodo 2 essendo quest'ultimo il destinatario
    - Il Nodo 2 elabora il dato ed esegue le azioni (accensione/spegnimento)
    - Restituisce l'ok di avvenuta ricezione al Nodo 1
    - Il Nodo 1 riceve l'ACK ed invia egli stesso la conferma al Nodo Master
    - Il Nodo Master riceve l'ACK di avvenuta ricezione
    - La pagina si aggiorna

In questo caso la comunicazione tra Nodo 1 - Nodo 2 avviene allo stesso e identico modo della comunicazione tra Nodo Master - Nodo 1:

- Il Nodo 1 manda il dato (continuamente per 8-10 sec.) al Nodo 2 e rimane costantemente in ascolto
- Il Nodo 2, una volta ricevuto il dato, manda continuamente l'ACK di avvenuta comunicazione
- La comunicazione si chiude all'attesa di 10 sec. o con l'invio dell'ACK.
- Se l'ok di comunicazione non avviene questo non giungerà nemmeno al Nodo Master, generando una pagina di errore!

*N.B: Per prevenire l'accavallarsi dei dati, inviati da entrambi i nodi, al Nodo 1 si è pensato di far mandare al Nodo Master il dato un tantum... In questo modo ci saranno dei buchi in cui il Nodo 1 percepirà solo il segnale inviato dal Nodo 2!*

## Database (Firebase)

Adesso non ci resta che implementare un database dove salvare i nostri dati, che saranno visibili in remoto attraverso un'app Android. Il database che si è utilizzato per questo progetto è [Firebase](#) (database NoSQL). Firebase è semplice ed intuitivo da utilizzare. La struttura che utilizza non è una struttura tabellare ma ad albero.

Una volta implementato il DB dobbiamo effettuare la connessione a Firebase grazie ad una libreria per l'ESP8266 che è presente nell'occorrente software elencato a inizio pagina. Abbiamo bisogno, per il collegamento, dell'HOSTNAME e dell'autorizzazione:

- Accedete allo sketch del Nodo Master attraverso Arduino IDE
- Aprite Firebase
- Andate al vostro DB creato in precedenza
- Copiate l'hostname del DB e incollatelo nello sketch alla variabile FIREBASE\_HOST senza https:// e lo / finale
- Andate su Firebase > Impostazioni > Database > Database Secret
- Cliccate su SHOW e copiate il codice di autorizzazione
- Incollate il codice nella variabile FIREBASE\_AUTH presente nello sketch

*N.B:Nello sketch postato su GitHub, non è presente ne SSID, Password, FIREBASE\_HOST, FIREBASE\_AUTH, perchè sono dati sensibili! Dovrete inserire al posto dei "/\*" le vostre credenziali.*

Elichiamo quello che succede nello sketch da eseguire sul Nodo Master(gli altri sketch, delle altre schede poste sugli altri nodi, rimangono invariati):

- Il client accede all'indirizzo IP della pagina HTML e fa una richiesta di modifica dei nodi
- Il Nodo Master riceve questa modifica attraverso la form e manda i dati, attraverso la connessione ottica, al Nodo 1
- Il Nodo 1 elabora i dati e li esegue o li invia al Nodo 2
- Il Nodo 1, una volta eseguito tutto, manda una risposta al Nodo Master, confermando l'avvenuta ricezione

- Il Nodo Master aggiorna la pagina HTML da mandare al client e contemporaneamente aggiorna il DB
- La pagina HTML viene aggiornata

In remoto sarà presente uno smartphone o tablet android con un'app per il controllo dello stato. Dalla semplice app, realizzata con [AppInventor](#), sarà possibile monitorare lo stato della rete e di tutti i nodi. In futuro potrebbe essere implementato anche un controllo, in remoto, dell'intera rete.

## App Android



Come dicevamo, l'app è stata realizzata attraverso un software open source online della Google: App Inventor. Tale software è molto semplice ed intuitivo, in futuro verrà fatta una guida su come utilizzarlo.

L'app presenta una schermata principale in cui ci sono due grossi button:

- Un occhio = rappresenta il controllo dello stato della rete
- Un joypad = rappresenta il controllo in remoto delle modifiche da portare ai nodi

Per il momento il controllo della modifica dei nodi non può essere effettuato (verrà implementato in un secondo momento) Cliccando sull'occhio abbiamo l'apertura della pagina "Controllo stato", in cui sarà presente lo stato di ogni nodo. Tale stato sarà prelevato da Firebase.

## Problemi rilevati

In questa parte verranno riportati tutti i problemi rilevati durante la realizzazione del progetto; questi problemi sono già riportati nella relazione, questa sezione è solo un resoconto di tutti i problemi e le loro risoluzioni.

### Problemi

- Alimentazione instabile shield ESP8266
- Lettura instabile sensore IR Nodo Master
- Interruzione comunicazione ottica da parte di un oggetto o altro
- Comunicazione "Connected Oriented" Nodo 1 - Nodo 2

### Soluzioni

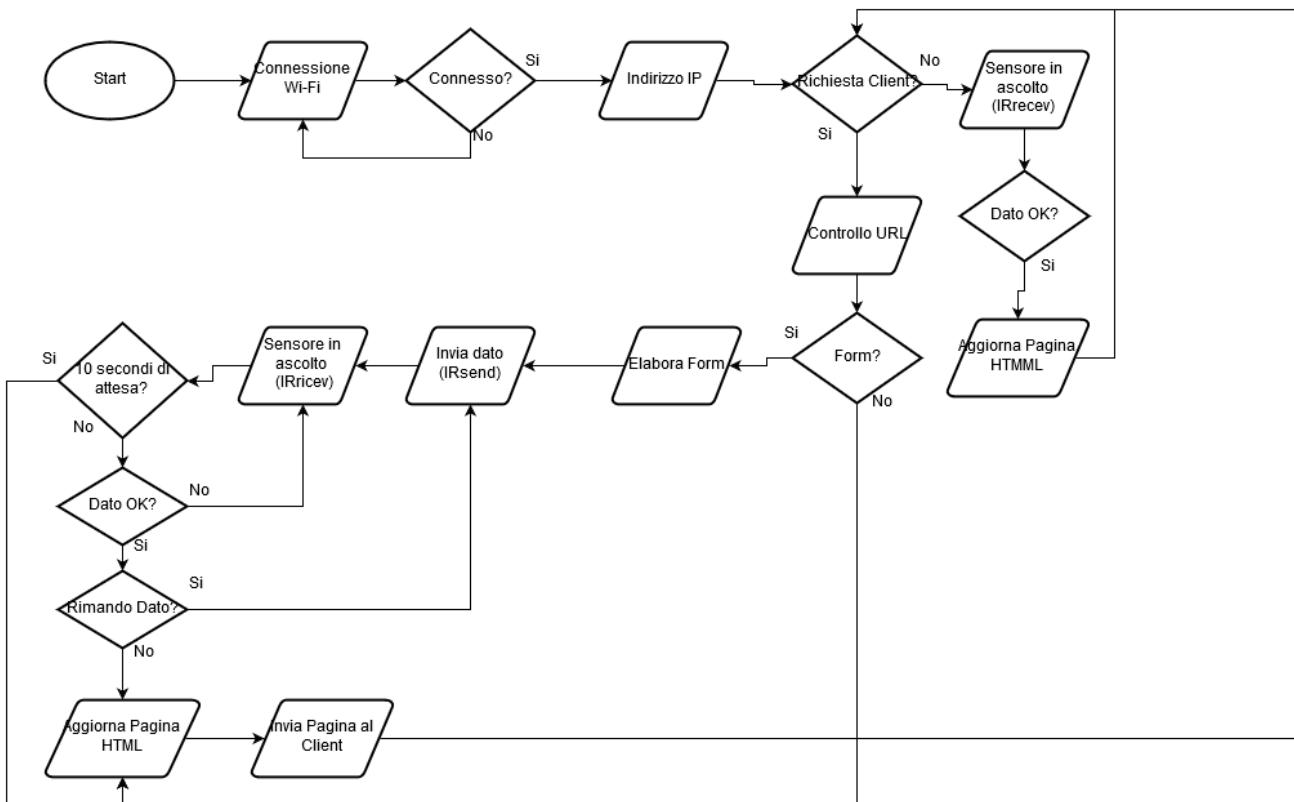
- L'alimentazione instabile (5v) della shield, non ci ha permesso di collegare il sensore IR con i 5v della scheda; si è risolto tale problema alimentando il sensore con i 5v di un'altra scheda Arduino.
- La lettura instabile del sensore IR sul Nodo Master non è stato risolto! Purtroppo la scheda rileva letture random di valori non conformi a quelli inviati dal nodo adiacente. Questo problema è dato solo dal Nodo Master, mentre la comunicazione tra il Nodo 1 e Nodo 2 avviene in modo impeccabile. Questo problema è comunque in parte risolto grazie all'invio continuo dell'ACK da parte del Nodo 1 al Nodo Master, in modo da fargli leggere, prima o poi, l'ok di avvenuta ricezione. Purtroppo la sicurezza assoluta non c'è!
- L'interruzione della comunicazione può avvenire in due modi:
  - Oggetto che ostacola la comunicazione ottica;
  - Malfunzionamento di uno dei componenti di comunicazione.

Per entrambi i casi si è previsto un javascript con un WARNING.

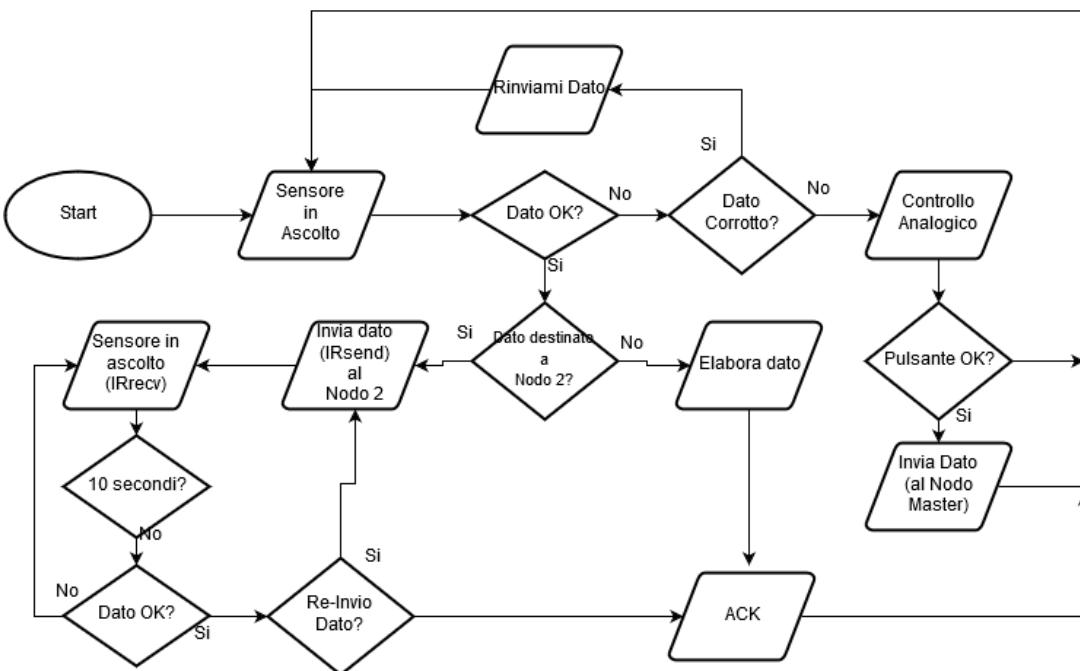
- Per la comunicazione "Connected Oriented" tra Nodo 1 e Nodo 2 si è pensato a una serie di soluzioni che potrebbero risolvere il problema:
  - Sensore sferico 
  - Triangolazione del segnale 
  - Serie di specchi per dirigere il segnale 
  - Schema di switch per doppio sensore 

## Algoritmi

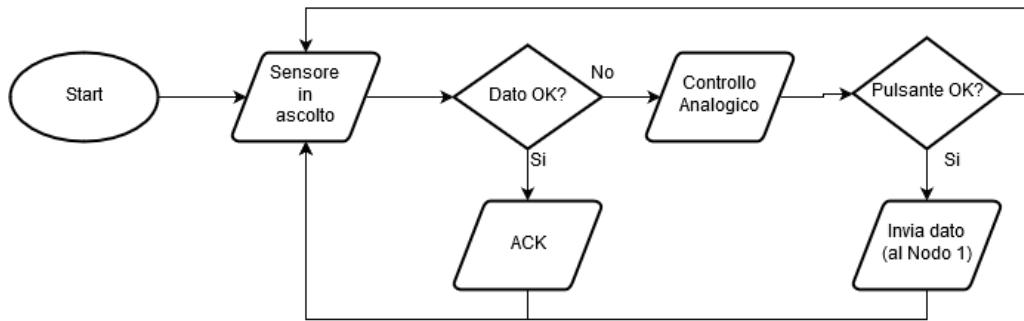
### Nodo Master:



### Nodo 1:



## Nodo 2:



## Conclusioni e costi complessivi

### Costi:

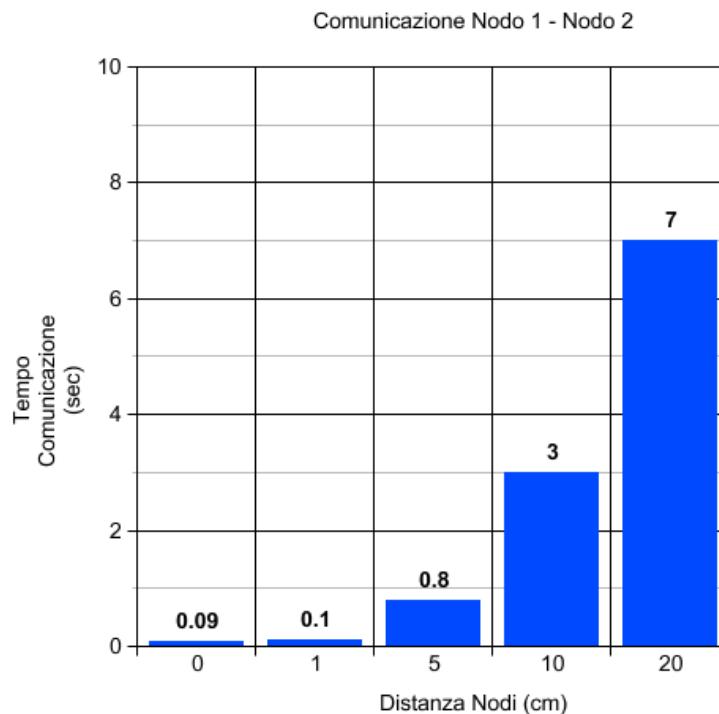
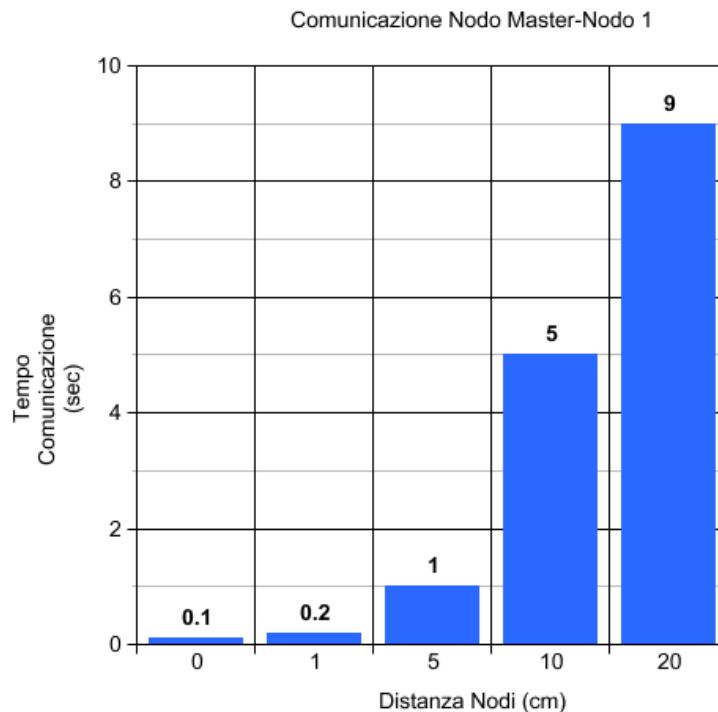
I costi dei vari moduli, schede, cavi, breadboard, led e altro si aggira intorno a questo range: 30€ - 70€. Il tutto dipende da dove comprate il materiale. Il consiglio che vi diamo è quello di comprare tutto su e-Bay... è un sito molto sicuro e trovate tutto quello di cui avete bisogno. Vi allego il link del venditore dove abbiamo reperito tutto il materiale:

[http://www.ebay.it/usr/nouteclab?\\_trksid=p2047675.l2559](http://www.ebay.it/usr/nouteclab?_trksid=p2047675.l2559)

### Conclusioni:

Riportiamo le conclusioni con un elenco puntato:

- Miglioramento capacità tecniche e pratiche in:
  - Informatica
  - Programmazione
  - Elettronica
  - Comunicazioni
- Rapporti di coesione e lavoro di squadra
- Rendere più performante la comunicazione ottica sostituendola con quella a onde radio
  - Utilizzando le onde radio si migliora la qualità del segnale inviato
  - La comunicazione può avvenire in modo diretto tra Nodo Master e Nodo 2
  - Si instaurerà un ponte di nodi, per la comunicazione, solamente se i nodi si trovassero fuori dal range di comunicazione
  - Se dovesse presentarsi un ostacolo permanente o no, il segnale passerebbe ugualmente.



L'intera relazione e manuale d'uso sono costantemente aggiornate e presente al sito di GitHub:  
<https://github.com/domoticawifi/Network-nodes>

