

Progettazione di una Smart GreenHouse.

Università degli studi di Bologna
Campus di Cesena
Facoltà di Ingegneria e Scienze Informatiche,
SEIOT A.A 2018/2019



Docente: Prof. Alessandro Ricci

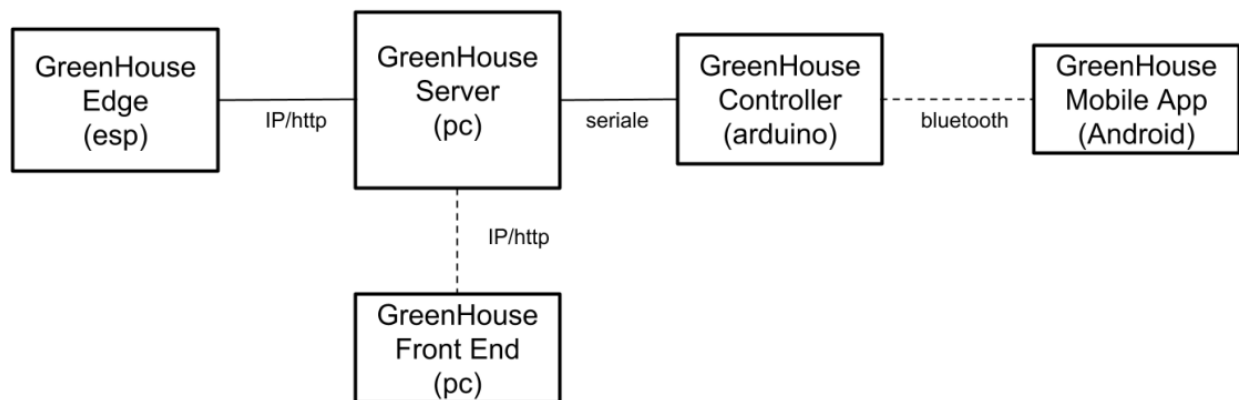
Studenti: Sokol Guri (sokol.guri@studio.unibo.it)
 Hamado Dene (hamado.dene@studio.unibo.it)
 Nicola D'Auria (nicola.dauria@studio.unibo.it)

Tabelle dei contenuti

<i>Capitolo 1 - Specifiche del progetto</i>	<i>3</i>
<i>Capitolo 2 - Progettazione</i>	<i>4</i>
2.2 Implementazione del circuito	4
<i>Capitolo 3 - SGH Controller</i>	<i>5</i>
3.0 Progettazione dei Task (PRO / CONTRO)	5
3.1 Distance Controller Task	6
3.2 Detect Irrigation Mode Task	7
3.3 Irrigation Task	8
<i>Capitolo 4 - SGH Edge</i>	<i>9</i>
4.1 Funzionamento Espruino	9
4.2 NGROK tunnel	9
<i>Capitolo 5 - SGH Android App</i>	<i>10</i>
8.1 Connessione	10
8.2 Core dell'app	11
8.3 Comunicazione Bluetooth con Arduino	12
<i>Capitolo 6 - SGH Font End</i>	<i>13</i>
<i>Capitolo 7 - SGH Server</i>	<i>15</i>

Capitolo 1 – Specifiche del progetto

Si vuole realizzare un sistema embedded integrato che rappresenti una versione semplificata di una serra smart. Il compito della serra smart è l'irrigazione automatizzata (di un certo terreno o pianta) implementando una strategia che tenga conto dell'umidità percepita, con la possibilità di controllare e intervenire manualmente mediante mobile app. Il sistema è costituito da 5 parti (sottosistemi):



- **GreenHouse Server (PC)**: contiene la logica che definisce e attua la strategia di irrigazione
- **GreenHouse Controller (Arduino)**: permette di controllare l'apertura e chiusura degli irrigatori
- **GreenHouse Edge (ESP)**: permette di percepire l'umidità del terreno
- **GreenHouse Mobile App (Android)**: permette di controllo manuale della serra
- **GreenHouse Front End (PC)**: Front end per visualizzazione/osservazione/analisi dati

Capitolo 2 - Progettazione

2.2 Implementazione del circuito

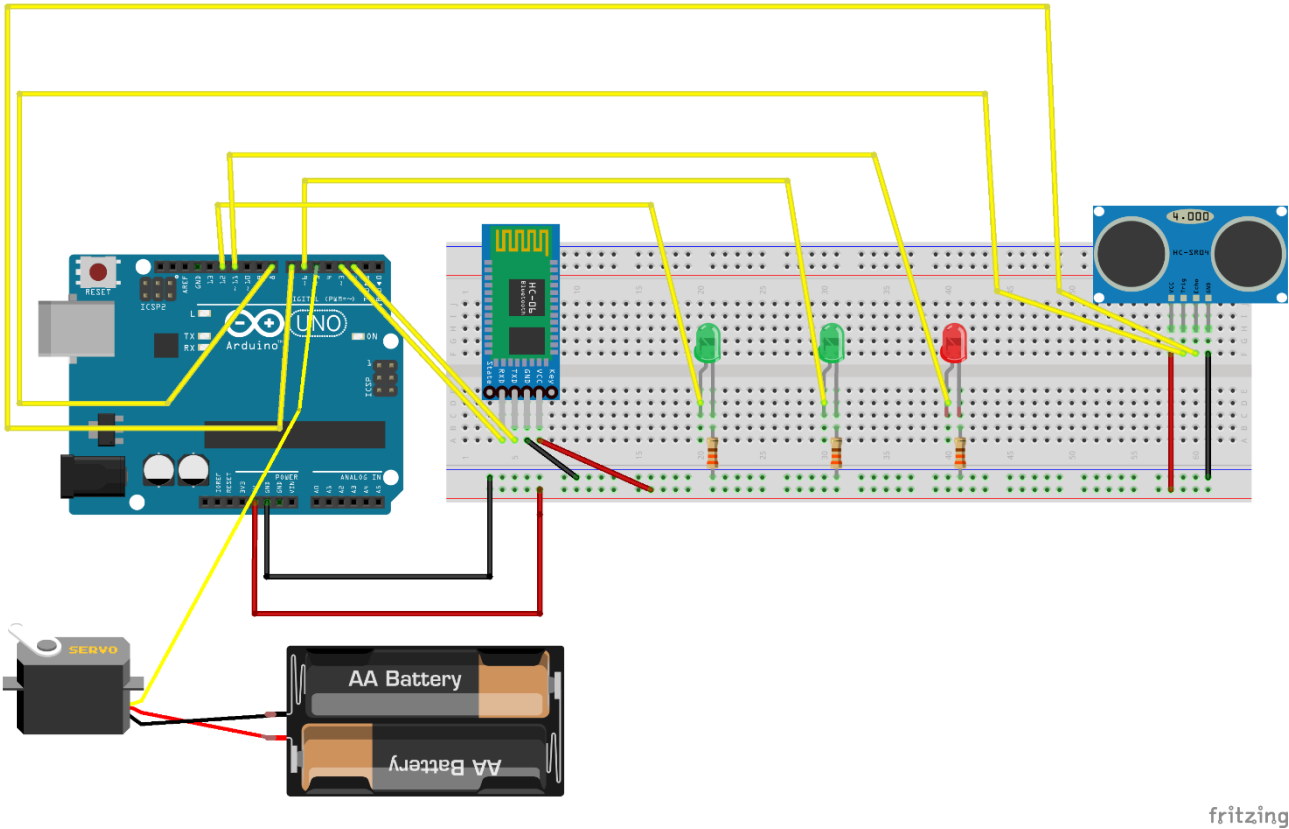


Figure 1, Circuito Arduino

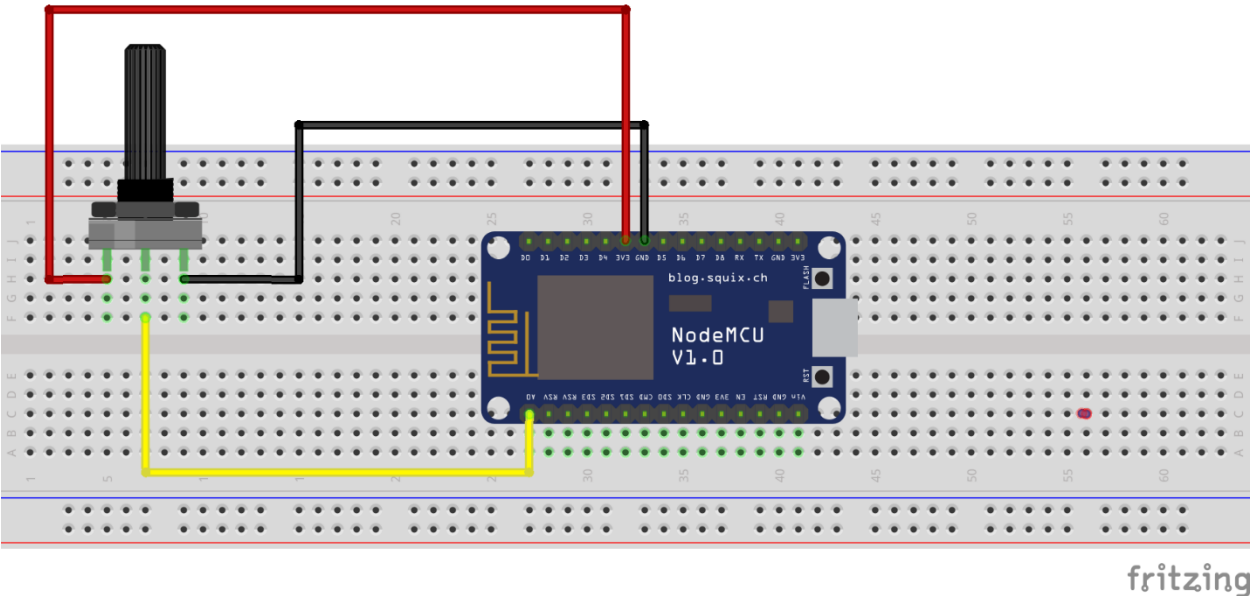


Figure 2, Circuito Esprimo

Capitolo 3 - SGH Controller

3.0 Progettazione dei Task (PRO / CONTRO)

All'inizio la parte del controller (ARDUINO) non era organizzata in tre task, ma in due. I due task erano DistanceControllerTask and IrrigationTask.

DistanceControllerTask rimane uguale, quello che cambia e IrrigationTask.

IrrigationTask all'inizio era pensato di svolgere più funzionalità dentro lo stesso task. Il task doveva fare dei controlli (la distanza dell'utente dalla sera, connessione Bluetooth etc.) per cambiare il modo di irrogazione dell'acqua. Poi il task doveva erogare acqua, tenendo conto del modo che aveva rilevato dai controlli che aveva fatto. Noi abbiamo deciso di dividere questi due concetti in due task diversi. Un task doveva rilevare il metodo di irrigazione e un altro task doveva gestire lo stato delle pompe. Certo che due task non potevano comunicare tra di loro, per questo le variabili usati per comunicare venivano salvati in uno spazio chiamato *SharedSpace*. SharedSpace offriva un'interfaccia con diversi metodi per far comunicare in modo sicuro i diversi task.

PRO: Abbiamo deciso di proseguire con questa strada perché:

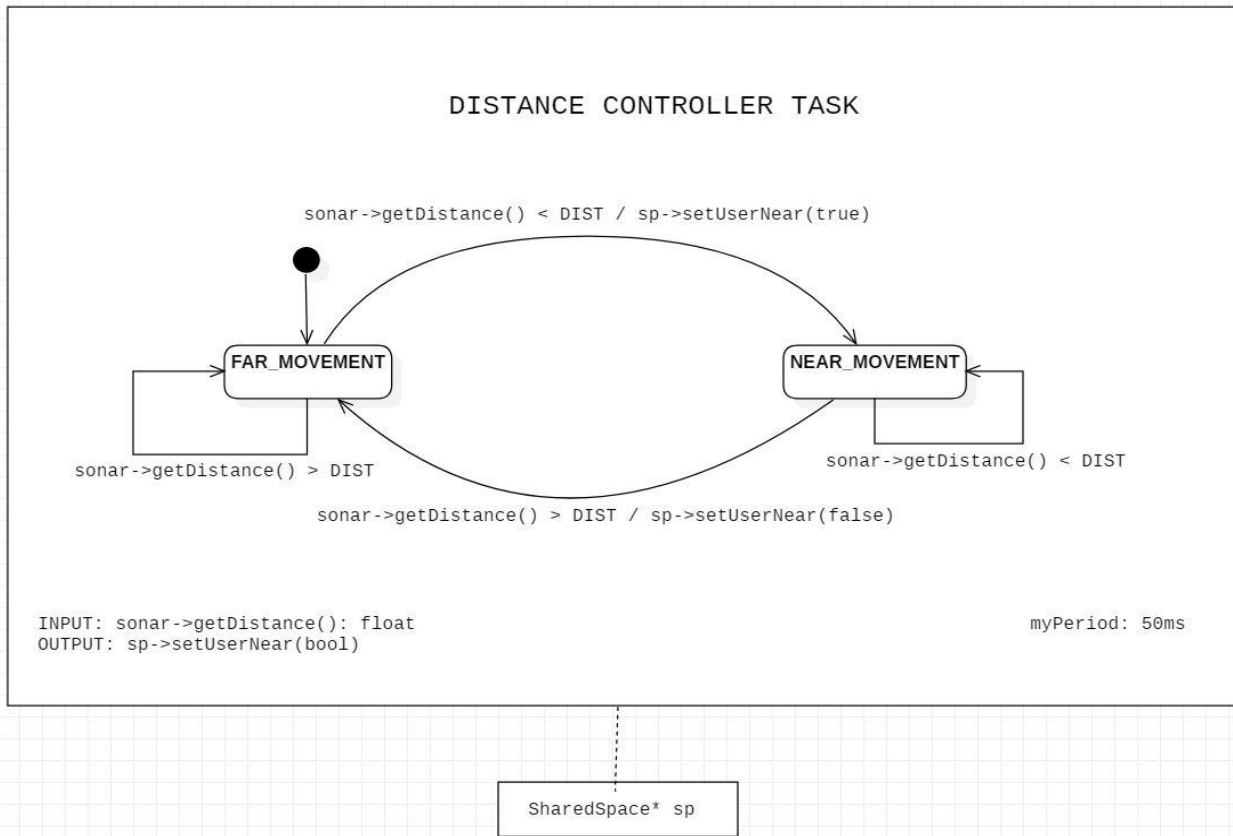
1- la modalità erogazione Automatica o Manuale (tramite l'app) e lo stato delle pompe sono due concetti diversi e potevano essere separati.

2- si poteva rappresentare meglio il cambiamento di stato. La modellazione con macchine a stati finiti (FSM) veniva più chiara e leggibile.

CONTRO: Tutta la logica del controller era incapsulata dentro ad un unico task. Prima si poteva gestire il modo di irrogazione e l'apertura delle pompe dentro lo stesso task. Tutte le informazioni necessarie erano nello stesso task, non dovevamo fare comunicare task diversi.

3.1 Distance Controller Task

Questo task svolge la funzionalità di controllare la distanza dell'utente dalla sera. Nell'Arduino abbiamo un sensore che misura la distanza tramite la propagazione del suono. Questo sensore si chiama *Sonar* e il termine nasce come acronimo dell'espressione inglese *sound navigation and ranging*.



Il task ha due stati:

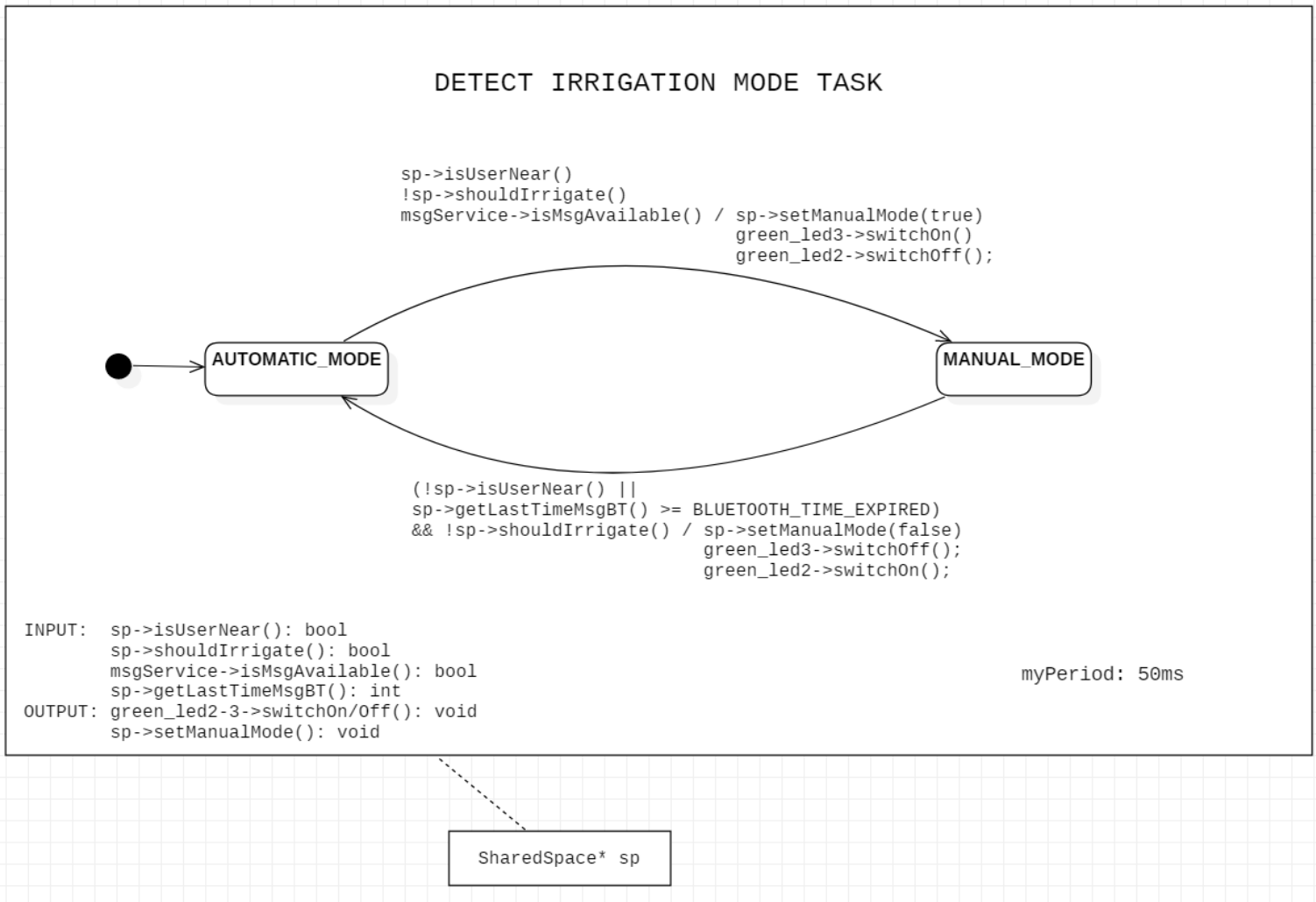
- * **FAR_MOVEMENT:** il task rimane in questo stato fino a quando non rileva una distanza minore di una costante `DIST`. Se viene rilevato una distanza maggiore di `DIST`, scriviamo nel `SharedSpace` `sp->setUserNear(true)` e passiamo in un altro stato.

- * **NEAR_MOVEMENT:** il task rimane in questo stato fino a quando non rileva una distanza maggiore di una costante `DIST`. Se viene rilevato una distanza minore di `DIST`, scriviamo nel `SharedSpace` `sp->setUserNear(false)` e passiamo in un altro stato.

3.2 Detect Irrigation Mode Task

Questo task ha la funzionalità di controllare in che modalità sarà fatta l'erogazione dell'acqua. Dopo aver rilevato la modalità accende il led corrispondente. Le due possibili modalità sono :

- 1- automatica
- 2- manuale.

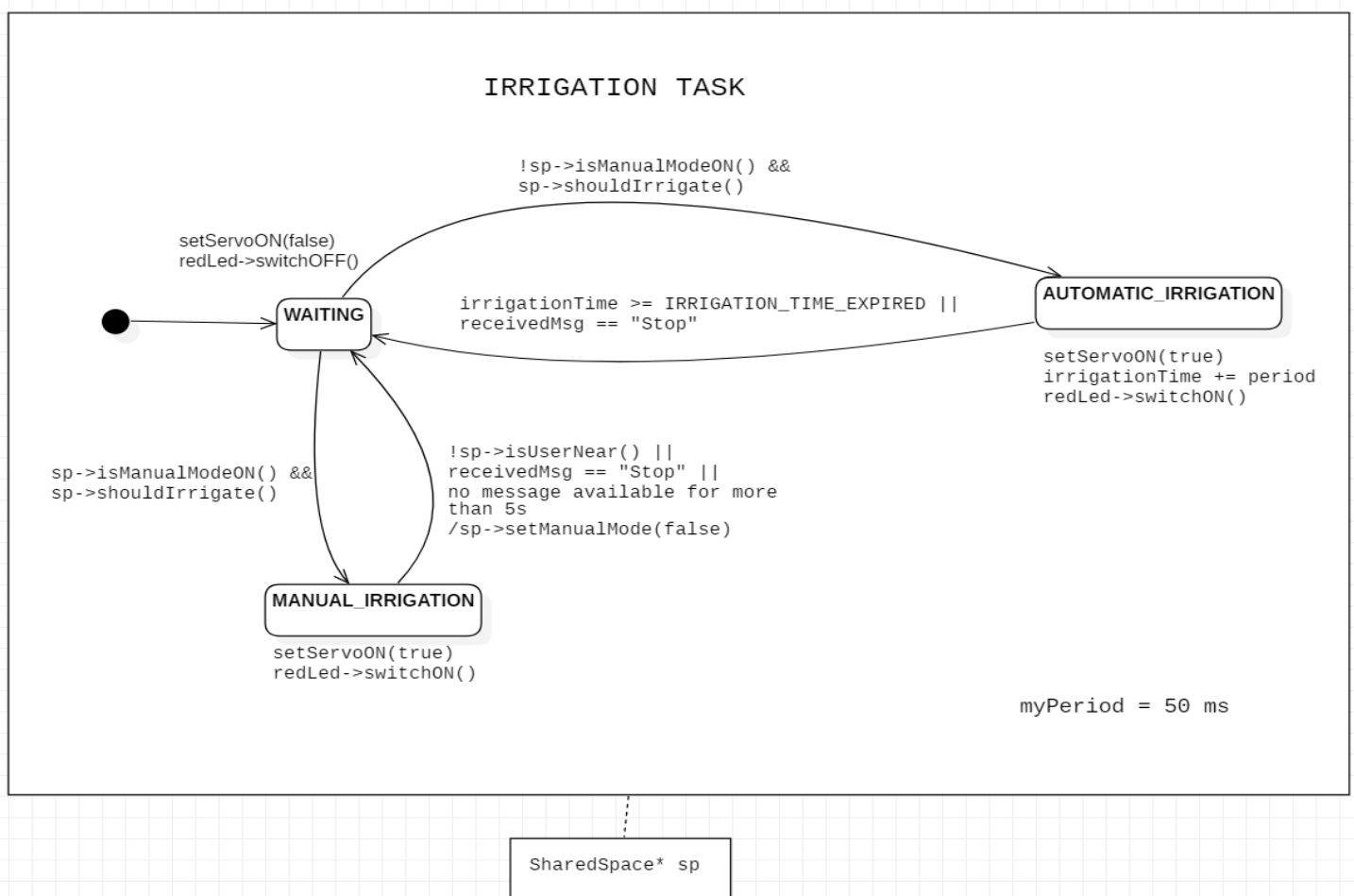


All'inizio io sono in AUTOMATIC_MODE. Io passo allo stato MANUAL_MODE quando rilevo un user nelle vicinanze e la connessione Bluetooth è possibile. Devo controllare anche se sp->shouldIrrigate() è falso, perché non posso cambiare stato durante l'erogazione. Se tutte queste condizioni vengono rispettate io passo in modalità manuale e il task cambia stato. Quando sto per passare in modalità manuale, devo settare nel SharedSpace sp->setManualMode(true). Con questa azione riesco a dire al prossimo task come procedere con l'erogazione.

Green_led2 e Green_led3 si accendono e si spengono in modo alternato quando cambio stato. Per poter passare alla modalità automatica basta che l' user si allontana oppure la connessione Bluetooth ci mette più di un tot di tempo. Certo che per passare dalla modalità manuale ad automatico, le pompe non devono essere nello stato di erogazione.

3.3 Irrigation Task

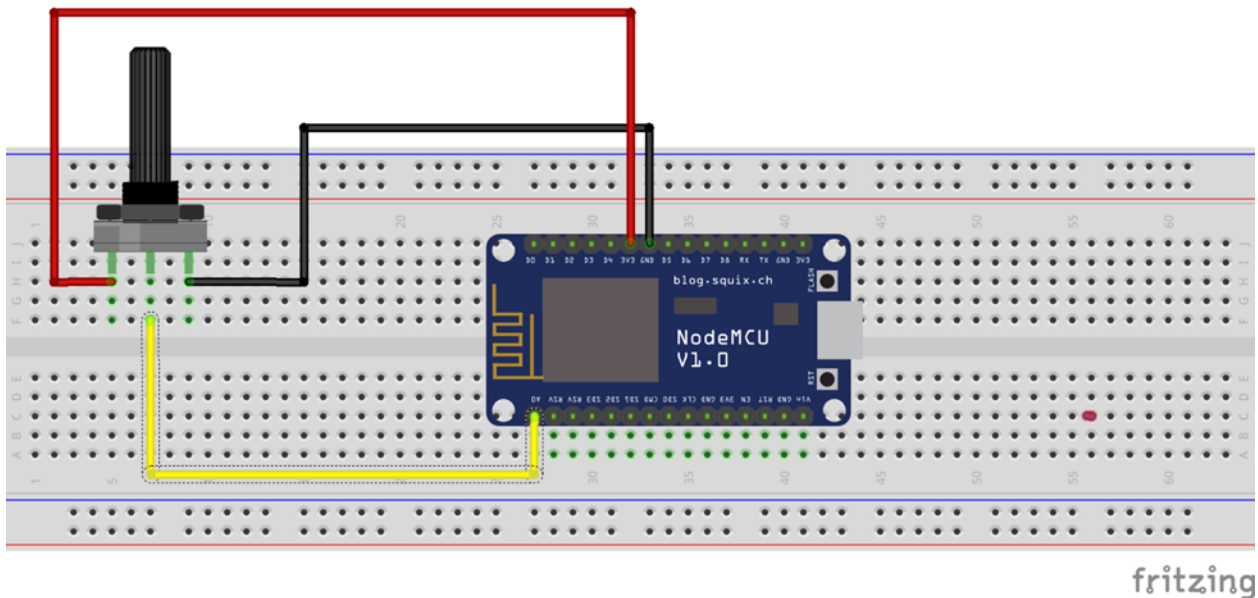
Questo è "il kernel" del tutto il controller. In questo task si gestisce l'apertura e la chiusura delle pompe e la quantità dell'acqua erogata. Per capire che modalità di apertura il programma deve usare, il task accede nello SharedSpace e tutto dipende dalla variabile booleana `sp->isManualModeON()`. Se questa variabile ritorna true, siamo in MANUAL_IRROGATION, altrimenti siamo in AUTOMATIC_IRROGATION.



Capitolo 4 – SGH Edge

4.1 Funzionamento Espruino

Lo scopo del modulo ESP è quello di percepire il valore dell'umidità del terreno e comunicarla al server, in modo che questa informazione sia fruibile da tutte le parti del sistema. In questo caso abbiamo scelto di simulare il valore letto dal sensore di umidità attraverso la lettura del valore di un potenziometro. Ogni 1,5 secondi il valore percepito verrà letto, mappato in un valore compreso tra 0 e 100 e inviato in formato Json al server attraverso Ngrok.



4.2 NGROK tunnel

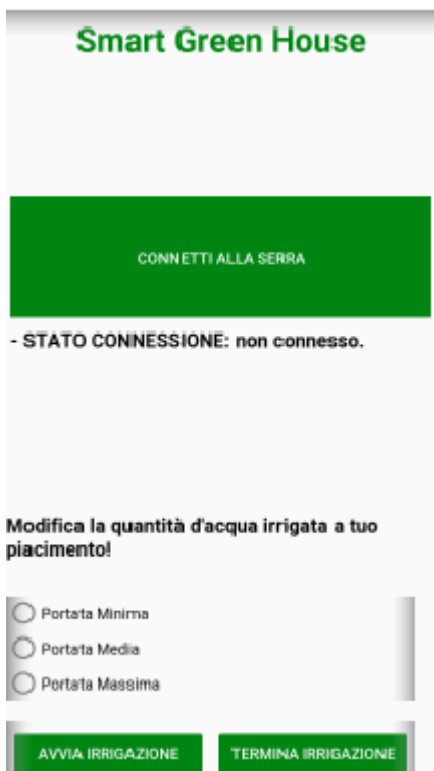
Questo servizio effettua un tunnel sicuro dell'indirizzo locale (localhost), anche se la mia rete è dietro un proxy oppure un firewall.

Lo scenario classico che andrà bene per molti progetti Web, è quello di avere un Web Server per esempio Apache, sulla porta standard 80, tramite il comando:

```
./ngrok http 80
```

Capitolo 5 – SGH Android App

L'applicazione è sviluppata per permettere il controllo della serra quando si è nelle sue prossimità (distanza inferiore ad x) e quindi la possibilità di decidere la quantità d'acqua utile all'irrigazione, mediante tre scelte con valori predefiniti, e l'apertura e la chiusura degli irrigatori.



8.1 Connessione

Per permettere all'applicazione di svolgere il proprio lavoro è necessario che stabilire una connessione Bluetooth tra la serra e il dispositivo, ciò è possibile grazie al bottone "Connetti alla serra".

Purché si verifichi questa connessione è necessario:

- Attivare il Bluetooth del dispositivo;
- Effettuare l'accoppiamento tra il dispositivo e la serra (HC-6);
- assicurarsi di aver collegato il modulo Arduino a un PC e di aver caricato sulla scheda il codice presente nella cartella sgh-controller.

N.B.: Non è possibile effettuare connessioni con dispositivi diversi dall'HC-6, nel caso si voglia cambiare dispositivo con cui stabilire la connessione è possibile farlo nella classe `com.example.sgh_app.Utils` sostituendo "HC-06" con il nome del dispositivo desiderato nella linea di codice riportata di seguito:

```
Public static final String
```

```
BT_DEVICE_ACTING_AS_SERVER_NAME = "HC-06";
```

Dunque, il modulo Android effettua il tentativo di attivare un canale di comunicazione basato sullo standard RFCOMM. In questo caso il dispositivo Android è il client e la serra è il server. A questo scopo si esegue il task **ConnectToBluetoothServerTask** il cui compito è eseguire il tentativo di connessione al server. L'istanza della socket su cui tentare la connessione al server è ottenuta mediante la funzione `createRfcommSocketToServiceRecord()` a cui si passa l'UUID condiviso con il server.

8.2 Core dell'app

Una volta stabilita la connessione Bluetooth e una volta passata in modalità manuale, è dunque possibile utilizzare i comandi disponibili:

- Scelta quantità acqua;
- Bottone "Avvia Irrigazione";
- Bottone "Termina Irrigazione";

Nel caso ci si allontani dalla zona di rilevazione l'app tornerà in modalità automatica e tutti i comandi verranno disabilitati ma senza perdere la connessione Bluetooth con la serra, in questo modo se il sonar rileva di nuovo la nostra posizione è possibile ritornare in modalità manuale senza dover ristabilire la connessione.

In modalità manuale non è possibile terminare l'irrigazione se essa è già terminata, stessa cosa per quanto riguarda se l'irrigazione è già avviata. Per quanto riguarda la portata è possibile modificare la quantità in qualsiasi momento, purché in modalità manuale, ma la modifica espressa durante un'irrigazione già avviata sarà effettuata all'irrigazione successiva. Dunque la quantità d'acqua irrigata rimane invariata durante tutto il periodo di irrigazione e non è possibile modificarla durante il processo stesso.

8.3 Comunicazione Bluetooth con Arduino

Lo scambio di informazioni che avviene tra Arduino e il dispositivo Android è limitato messaggi riportati nella tabella seguente, i messaggi aiutano a capire cosa succede nella serra.

Messaggi inviati	Significato	Messaggi ricevuti	Significato
"1"	Accendi pompa	"ManIn"	Modalità manuale
"2"	Spegni pompa	"ManOut"	Modalità Automatica
"3"	Portata minima	"Start"	Irrigazione Avviata
"4"	Portata media	"Stop"	Irrigazione Terminata
"5"	Portata massima	"Numero"	Umidità percepita
"6"	Connesso Bluetooth		

Capitolo 6 – SGH Font End

Il modulo del frontend è stato implementato utilizzando usato una web application che interagisce con il database del server tramite delle servlet http.

HOME

Greenhouse FrontEnd


Home

Umidity

pump status

Configuration

Greenhouse



WELCOME TO GREENHOUSE SERVER

If you are using this page then the server is in execution.

You can see information about the configuration of the service port, pump status and umidity value;
the data is taken from the server.

UMIDITA'

Greenhouse FrontEnd

Home

Umidity

pump status

Configuration

Greenhouse

	DATA	NAME	VALUE
Read in	2019-04-10 19:28:49.120	umidita	50
Read in	2019-04-10 19:29:07.143	umidita	50
Read in	2019-04-10 19:29:07.380	umidita	50
Read in	2019-04-10 19:29:07.577	umidita	50
Read in	2019-04-10 19:29:07.785	umidita	50
Read in	2019-04-10 19:29:07.928	umidita	50
Read in	2019-04-10 19:29:08.105	umidita	50
Read in	2019-04-10 19:29:08.271	umidita	50
Read in	2019-04-10 19:29:08.459	umidita	50
Read in	2019-04-10 19:29:08.657	umidita	50
Read in	2019-04-10 19:29:08.831	umidita	50
Read in	2019-04-10 19:29:09.080	umidita	50
Read in	2019-04-10 19:29:09.237	umidita	50
Read in	2019-04-10 19:29:09.427	umidita	50
Read in	2019-04-10 19:29:09.600	umidita	50

PUMP STATUS

Greenhouse FrontEnd

Home

Umidity

pump status

Configuration

☰ Greenhouse

DATA	INFO
2019-04-10 20:45:03.128	Send message for active pump,minimum flow rate
2019-04-10 20:46:58.713	Send message for active pump,average range
2019-04-10 20:47:41.013	Send message for active pump,maximum flow rate

SERVER CONFIGURATION

Greenhouse FrontEnd

Home

Umidity

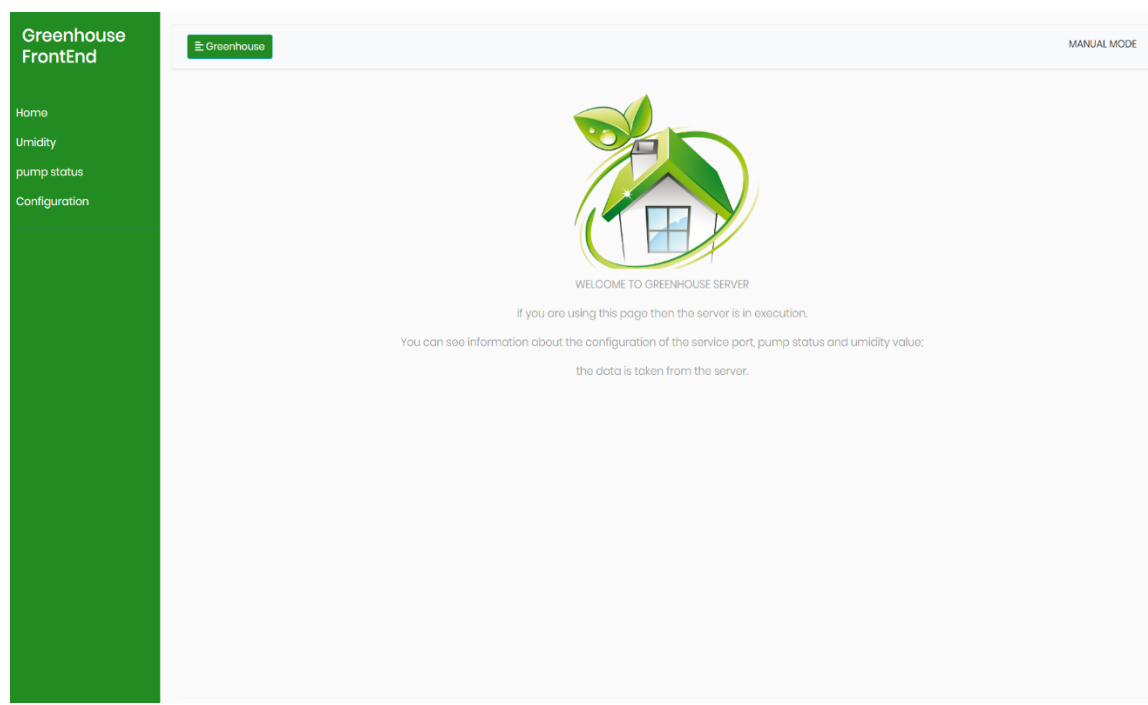
pump status

Configuration

☰ Greenhouse

SERVICE	PORT
Dataservice.port	8081
Frontend.port	8001
Messageservice.port	/dev/ttyACM0

MODALITA DI IRRIGAZIONE



Capitolo 7 – SGH Server

Il server è quello che implementa la logica applicativa. Il suo compito è quello di rimanere in ascolto su una determinata porta sulla quale riceverà i valori dell'umidità. I valori ricevuti vengono salvati su un database.

Il database è una database opensource che vive insieme al server, ovvero parte quando parte il server, e muore quando il server viene spento.

(<https://github.com/diennea/herddb.git>).

Tramite un message service, qualora la modalità risultasse manuale e il valore dell'umidità fosse compreso in una determinata soglia, il server informerà il controller di procedere con l'irrigazione.

Per inizializzare il server è necessario specificare la porta con la quale il server deve partire (nel caso non fosse specificato il server parte con il default 8081), la porta del

message service (se non specificato il server non sarà ingrato di informare il controller di avviare l'irrigazione) e la porta del frontend (se non specificato partirà con il default 8001). È possibile specificare questi valori in server.properties.

La procedura per avviare il server è la seguente:

NB. Questa procedura è valida solo per l'uso in un ambiente Linux. Se il build viene fatto in un ambiente windows, per avviare lo zip in un ambiente Linux è necessario eliminare i caratteri spuri del file in bin:

```
sed -i -e 's/\r$//' setenv.sh
sed -i -e 's/\r$//' sevice.sh
sed -i -e 's/\r$//' java-utils.sh
```

facendo il build del progetto (per esempio con Netbeans) viene creato uno zip (greenhouse-1.0-SNAPSHOT.zip) all'interno della cartella target (la logica di compilazione è già definita nei pom.xml con tutte le dipendenze).

Comandi:

1. `cp greenhouse-1.0-SNAPSHOT.zip /opt/`
2. `cd /opt/`
3. `unzip greenhouse-1.0-SNAPSHOT.zip`
4. `ln -s greenhouse-1.0-SNAPSHOT greenhouse`
5. `cd greenhouse`
6. se java non è settato come variabile d'ambiente, settare il percorso del jdk con /bin/setenv.sh
 - a. `vim bin/setenv`
 - b. `uncomment JAVA_HOME`
 - c. `Set JAVA_HOME → JAVA_HOME="/java/path/"`
 - d. `Settare Xmx e Xms e directmemory in base alla quantità di Ram che si dispone`
 - e. `Save`
7. Per personalizzare le porte e settare la porta del message service:
 - a. `Vim conf/server.properties`

- b. Fare le modifiche e salvare
- 8. `bin/service server start` → per avviare il server
- 9. `bin/service server stop` → per stoppare il server
- 10. `bin/service server restart` → per riavviare il server
- 11. `logs: service.server.log & logs/service.log`
 - a. il primo log contiene tutti i log riguardanti lo stato del server, del database ed eventuali exception.
 - b. Il secondo log risulta più leggera poiché contiene solo informazioni riguardanti le implementazioni di greenhouse
- 12. Per avviare il frontend →
`http://localhost:port/ui`

NB. Se si riavvia il server, i dati di umidità vengono persi. Ne rimane traccia comunque nei log

