

ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

Corso The Internet Of Things

Smart GreenHouse

Emilio Biello
843567
emilio.biello@studio.unibo.it

Anno Accademico 2018-2019

Sommario

Il progetto in questione prende il nome di "*Smart GreenHouse*" e rappresenta un'implementazione di un sistema in grado di monitorare la coltivazione delle piante in un contesto casalingo, al fine di ottenere la massima crescita, il monitoraggio e il controllo continuo di parametri ambientali.

Il sistema prevede una tecnologia smart e l'utilizzo dell'Internet of Things per aiutare le persone a prendere decisioni basate sui dati.

Indice

1	Introduzione	2
2	Componenti	3
3	Test	5
3.1	Photoressitor	5
3.2	Soil Moisture Sensor	6
4	Modello	7
4.1	La comunicazione: <i>MQTT</i>	8
4.2	L'archiviazione: InfluxDB	9
4.3	La Visualizzazione: Grafana	10
5	Implementazione	12
5.1	ESP32	12
5.2	Raspberry	13
6	Conclusione	15

1 Introduzione

Lo scopo principale di questo progetto è implementare un sistema semplice, a basso costo, basato su Arduino, per monitorare i valori dei parametri ambientali, i quali sono continuamente aggiornati e controllati al fine di ottenere una crescita e una resa ottimale delle piante.

Condizioni ambientali, quali temperatura, umidità e luminosità, e umidità del terreno, sono continuamente monitorate poiché variazioni delle stesse innescano azioni automatizzate.

Smart GreenHouse è un progetto costituito da diversi sensori e un attuatore, un'elettrovalvola, per controllare l'irrigazione della pianta; è anche utilizzata una webcam per monitorare, tramite un'immagine, lo stato di salute della pianta.

Il sistema oltre ad un Arduino, prevede un Raspberry Pi. Quest'ultimo utilizzato sia per archiviare i dati provenienti dai sensori, sia per realizzare un server locale in grado di offrire un servizio per la visualizzazione di tali dati, sfruttando in tal caso Grafana ed una pagina HTML appositamente realizzata per interagire con il sistema di monitoraggio della pianta.

Prima della fase di sviluppo è stato necessario effettuare dei test per comprendere appieno il comportamento e i valori restituiti dai sensori di luminosità e di umidità del suolo. Tale fase è stata fondamentale per interpretare il valore e per definire opportune soglie sui dati analogici restituiti.

Nei prossimi capitoli verranno descritti brevemente i sensori utilizzati, i test necessari per comprendere i dati restituiti dai sensori, il modello del progetto e la fase implementativa.

2 Componenti

Le componenti fondamentali che costituiscono il sistema Smart GreenHouse, oltre ad un Raspberry Pi 3 ed un ESP32, sono:

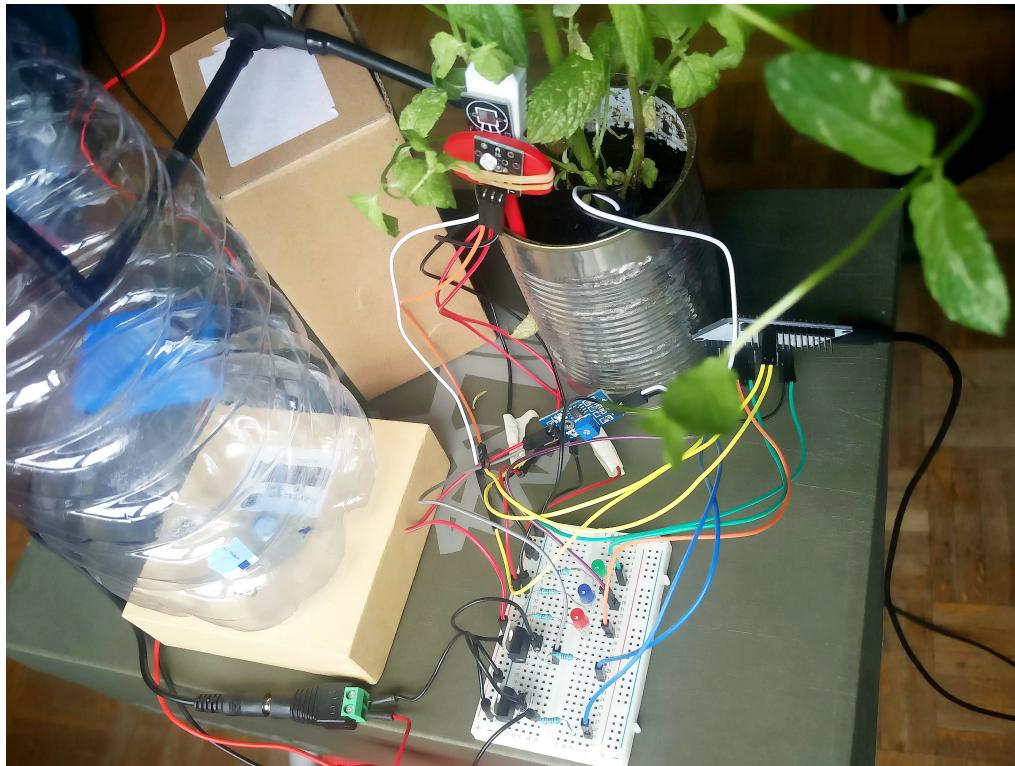


Figura 2.1: Smart GreenHouse

- L'elettrovalvola, funzionalmente un rubinetto che consente il passaggio di un fluido (liquido o gassoso) attraverso il varco individuato dalla valvola stessa. Il termine *elettrico* indica che l'attuatore dell'azionamento della valvola è comandato elettronicamente, solitamente un solenoide.
- L'igrometro, ovvero Soil Moisture Sensor, è uno strumento utilizzato per rilevare il contenuto volumetrico di acqua presente nel suolo. Per de-

terminare tali valori si sfruttano alcune proprietà del terreno, come la resistenza elettrica o costante dielettrica.

- Il sensore DHT22 permette di rilevare l'umidità relativa e la temperatura di un ambiente e trasmetterla digitalmente ad un microcontrollore. Permette di acquisire al massimo un campione ogni 2 secondi. La misurazione dell'umidità varia da 0 a 100% con un'accuratezza tipica pari al $\pm 2\%$. Per quanto riguarda la temperatura, il DHT22 ha un range di misurazione tra -40 e +80°C con un'accuratezza di $\pm 0.5\text{C}$ grazie al sensore DS18B20 integrato al suo interno.
- Il fotoresistore (LDR) è una resistenza variabile controllata dalla luce. La resistenza di un photoresistor diminuisce all'aumentare dell'intensità della luce. Il range di resistenza e la sensibilità di una fotoresistenza possono sostanzialmente differire tra i dispositivi dissimili.

Fondamentale per il funzionamento dell'elettrovalvola è stata l'introduzione di una fonte di alimentazione esterna, poiché la corrente erogata dal microcontrollore risulta essere insufficiente ad alimentarla. Difatti è stata inserita una batteria Lipo 12V. Poiché l'elettrovalvola viene azionata tramite il passaggio di corrente, è stato necessario introdurre un Transistor, controllato dall'ESP32 per indicare quando aprire e quando chiudere il circuito.

Un altro Transistor è stato introdotto per gestire l'alimentazione dell'igrometro. In modo da alimentare il sensore solo un istante prima di eseguire una misurazione, al fine di evitare l'ossidazione del sensore, visto che risulta sempre immerso all'interno del terreno.

Sono stati introdotti 3 LED per determinare alcuni stati del microcontrollore.

3 Test

Affinché i dati provenienti dai sensori siano interpretati in modo corretto è stato necessario eseguire dei test per il sensore della luminosità (Photoresistor) e per il sensore che misura il grado di umidità del terreno (Soil Moisture Sensor). Per gli altri sensori non è stato necessario eseguire tali test poiché restituiscono i risultati in digitale.

3.1 Photoressitor

Il sensore di luminosità restituisce i dati in analogico, quindi è stato fondamentale eseguire dei test per indicare i vari gradi di luminosità. Per determinare le due soglie estreme il sensore è stato usato in diverse condizioni di luce e sono stati analizzati i valori analogici restituiti. Così facendo è stato possibile convertire il valore analogico in percentuale. Per ottenere i valori in percentuale, è stata adoperata la funzione *map()* con i seguenti parametri:

$$lumen = map(value, 100, 4095, 100, 0)$$

- *value* corrisponde al valore restituito dal sensore
- 100 e 4095 corrispondono, rispettivamente alla soglia minima e soglia massima dei valori restituiti dal sensore di luminosità
- 100 e 0 indicano il range della percentuale
- *lumen* indica il valore espresso in percentuale della relativa misurazione

Da notare come le soglie relative alla scala percentuale sono invertite, con 0 come soglia massima e 100 come soglia minima, questo è stato necessario poiché il sensore utilizzato, Photoresistor, è un tipo di resistore la cui resistenza diminuisce all'aumentare dell'intensità della luce. Il flusso di corrente elettrica attraverso la fotoresistenza aumenta all'aumentare dell'intensità della luce.

3.2 Soil Moisture Sensor

L'igrometro, anch'esso restituisce dati in analogico, quindi anche per esso è stato necessario comprendere i dati prima di poterli adoperare. Il test consiste nel verificare quali fossero i valori estremi per tale sensore, ovvero alta presenza di acqua e poca presenza di acqua. Per individuare l'estremo che indicasse la massima umidità del terreno, il sensore è stato immerso in un bicchiere d'acqua e sono state eseguite diverse misurazione. Per individuare l'estremo che indicasse il massimo grado di siccità, il sensore è stato completamente asciugato e sono state eseguite altre misurazioni. Dai valori provenienti da tali osservazioni è stata impostata la funzione *map()* come di seguito, al fine di ottenere un valore percentuale che indicasse dei valori di umidità del terreno.

$$umdtrr = map(value, 700, 4095, 100, 0)$$

- *value* corrisponde al valore restituito dal sensore
- 700 e 4095 corrispondono, rispettivamente alla soglia minima e soglia massima dei valori restituiti dall'igrometro
- 100 e 0 indicano il range della percentuale
- *lumen* indica il valore espresso in percentuale della relativa misurazione

4 Modello

Di seguito viene illustrato il modello dello schema creato per collegare tutte le componenti all'ESP32.

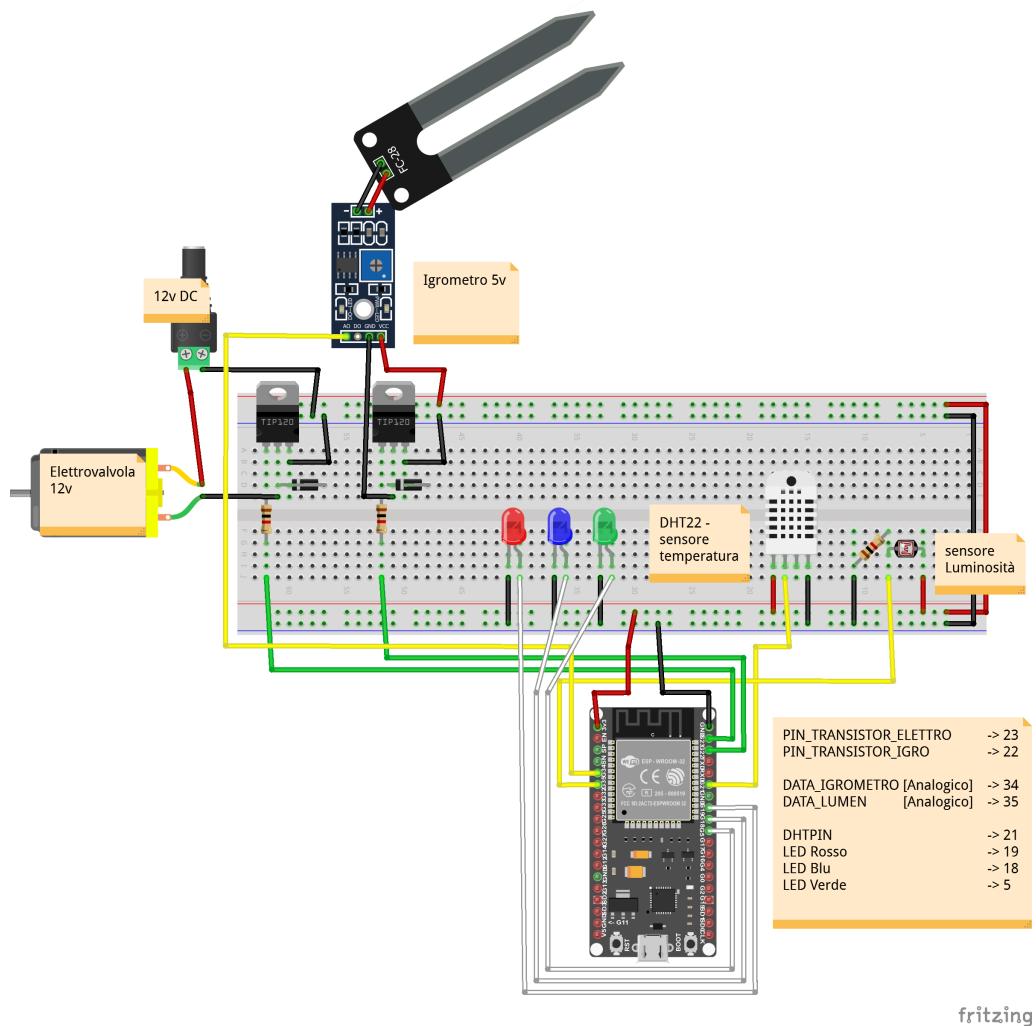


Figura 4.1: Schema sensori e attuatori Smart GreenHouse

Il modello inoltre prevede, l'utilizzo di MQTT per scambiare messaggi tra il microcontrollore e il Raspberry, un database time-series per l'archiviazione dei

dati ed Grafana per la visualizzazione dei dati.

4.1 La comunicazione: *MQTT*

La comunicazione tra il microcontrollore e il Raspberry Pi, avviene tramite il protocollo di comunicazione MQTT (Message Queuing Telemetry Transport), standard per lo scambio di dati nel mondo dell'IOT.

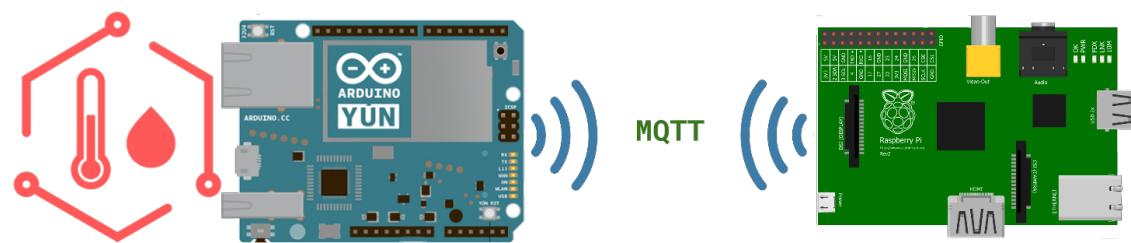


Figura 4.2: Comunicazione tra Microcontrollore e Raspberry Pi

Un protocollo specificatamente studiato per situazioni in cui è richiesto un basso consumo e una banda limitata.

Al posto del modello client/server di HTTP, il protocollo MQTT adotta un meccanismo di pubblicazione e sottoscrizione per scambiare messaggi tramite un apposito *"message broker"*.

Invece di inviare messaggi ad un determinato set di destinatari, i mittenti pubblicano i messaggi relativi ad un certo argomento (detto *topic*) sul message broker. Ogni destinatario si iscrive agli argomenti che lo interessano e, ogni volta che un nuovo messaggio viene pubblicato su quel determinato argomento, il

message broker lo distribuisce a tutti i destinatari. Con tale approccio è molto semplice configurare una messaggistica uno-a-molti. I principali attori presenti in una comunicazione tramite il protocollo MQTT:

- **Publishers**: genera dati e li invia ad un broker.
- **Subscribers**: si sottoscrive ad un topic di interesse, e riceve delle notifiche quando è disponibile un nuovo messaggio relativo al topic iscritto.
- **Broker** filtra i dati organizzandoli in topic e li distribuisce ai rispettivi subscriber.

Sul Raspberry è stato installato *Mosquitto*, un broker open source che implementa il protocollo MQTT, e Paho MQTT per poter realizzare un Client MQTT sfruttando il linguaggi di programmazione Python.

Sull'Arduino, invece, è stata installata la libreria *PubSubClient* per la realizzazione di un Clinet MQTT.

4.2 L'archiviazione: InfluxDB

I dati acquisiti dai vari sensori presenti in prossimità della pianta sono archiviate all'interno di un database Time-Series. Come suggerisce il nome, i database basati sulle serie temporali sono sistemi software ottimizzati per la memorizzazione di dati che evolvono nel tempo. Ogni dato viene archiviato assieme ad una marca temporale, in modo da identificare l'istante in cui è stata osservata tale misurazione.

Per archiviare i dati provenienti dai sensori è stato utilizzato InfuxDB. La scelta è ricaduta su tale sistema software poiché progettato e implementato appositamente per gestire serie di dati temporali, per il supporto al linguaggio SQL-Like per interrogare facilmente i dati consentendo ricerche veloci ed efficienti e per la compatibilità con i diversi linguaggi di programmazione.

InfluxDB è stato installato sul Raspberry ed al seguito è stato installato python-influxdb, un API per implementare un client Python in grado di interagire con il database precedentemente descritto.

4.3 La Visualizzazione: Grafana

Per la visualizzazione dei dati provenienti dai sensori è stato utilizzato Grafana, un tool basato su interfaccia web, in grado di monitorare e analizzare dati da tantissime fonti differenti.

La visualizzazione dei dati è offerta utilizzando delle dashboard interattive, ossia uno strumento che racchiude serie di informazioni acquisite eseguendo query in un database, solitamente di tipo time-series. Grafana mette a disposizione un generatore di query interattivo, in modo da rendere facile la generazione di grafici per l'utente finale.

Anche Grafana, come i precedenti tool, è stato installato sul Raspberry. Per esso non è stato necessario installare nessun modulo aggiuntivo, in quanto fornisce un'interfaccia web, accessibile sulla porta 3000, in grado di creare, gestire e visualizzare i propri cruscotti altamente personalizzati.



Figura 4.3: Grafana

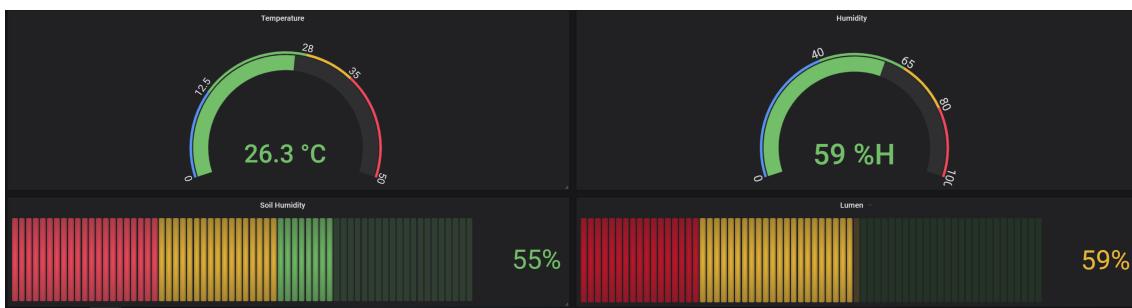


Figura 4.4: Grafana

5 Implementazione

Per questo progetto è stato realizzato un programma scritto in C++ ed eseguito sull'ESP32 per gestire i sensori prima citati. Uno script in Python eseguito sul Raspberry per ricevere i dati dal microcontrollore e memorizzarli nel database Influxdb, e una pagina HTML per consentire all'utente di sottoscriversi ad un apposito topic, per inviare una notifica al microcontrollore, la quale consente di azionare l'elettrovalvola o richiedere dati aggiornati dai sensori, e per visualizzare lo stato della pianta tramite una foto.

5.1 ESP32

Il programma eseguito sul microcontrollore, deve per prima cosa instaurare una connessione con il broker e successivamente procedere con l'invio dei dati acquisiti dai vari sensori in modo periodico. Come un qualsiasi programma eseguito da un microcontrollore, prevede 2 fasi.

Nella prima fase, di *startup*, dopo aver configurato la modalità di utilizzo dei vari pin, il dispositivo si connette alla rete, al broker e si sottoscrive al topic, *room/pianta_1/#*, tramite il quale è possibile comunicare delle azioni da eseguire. Nella seconda fase, di *loop*, il microcontrollore invia i dati acquisiti dai sensori, ogni 5 minuti, al broker utilizzando il seguente topic *room/pianta_1/* con i seguenti sottotopic: *temperature*, *humidity*, *lumen* e *igrometro*. All'interno del loop oltre a notificare i dati provenienti dai sensori è in attesa di notifiche relative al topic a cui si è iscritto. Alla ricezione di una notifica, verifica i sottotopic a cui si riferisce, e in base ad essi, innesca opportune operazioni.

I tre led, sono stati aggiunti, per verificare l'operazione corrente svolta dal dispositivo. L'accensione di un LED Rosso, identifica l'esecuzione della prima fase dell'algoritmo, l'accensione di un LED Blu identifica l'invio dei dati registrati

dai sensori al broker, mentre l'accensione di un LED Verde identifica l'esecuzione di un'azione richiesta dall'esterno, ovvero l'attivazione o la disattivazione dell'elettrovalvola per poter irrigare la pianta.

5.2 Raspberry

Il Raspberry, come descritto in precedenza dovrà consentire l'archiviazione e la visualizzazione dei dati. A tal proposito è stato realizzato uno script per poter utilizzare il protocollo MQTT e quindi sottoscriversi al topic *room/#*, in modo da poter ricevere notifiche da tutte le piante presenti all'interno di tale ambiente. Ricevute, una di seguito l'altra, quattro notifiche per ogni pianta relative a:*temperature, humidity, lumen, soil*, inviate dall'ESP32 ogni cinque minuti, crea un JSON per consentire la memorizzazione dei valori all'interno del database. A seguito dell'archiviazione, lo script, analizza il valore relativo all'umidità del terreno. Nel caso in cui risulta essere minore di un certo valore di soglia, su scala percentuale, invoca la funzione per irrigare il suolo della pianta corrispondente. Tale funzione prevede l'apertura dell'elettrovalvola, l'attesa di un intervallo di tempo e la chiusura della medesima valvola. A seguito di tale operazione viene comunicato il nuovo valore di umidità del terreno.

Sempre all'interno di tale script Python è stata implementata una RESTful API, usando *Flask*, in grado di fornire un'immagine, acquisita tramite una web-cam posizionata in prossimità delle piante.

Il Raspberry è stato configurato come un WebServer in grado, non solo di fornire i servizi precedentemente discussi ma, di fornire un'interfaccia web tramite la quale l'utente può sottoscriversi ad appositi topic, pubblicare dei messaggi al fine di attivare l'elettrovalvola o ottenere misurazioni da parte dei sensori e visualizzare una foto della pianta. Oltre a ciò, da tale pagine è possibile accedere a Grafana per la visualizzazione dei dati all'interno di cruscotti appositamente creati.

GreenHouse

[Link Grafana](#)

Hostname or Ip Address: Port:

Connected to: 192.168.1.137:9001 as clientID-ffMQR

CONNECT DISCONNECT PICTURE

Subscribe

Topic:

SUBSCRIBE UNSUBSCRIBE

#	Timestamp	Topic	Value
No data available in table			

Subscribing to room/#

Publish

Topic:

Message:

SEND

#	Timestamp	Topic	Value
No data available in table			

```

2019-7-15 19:37:42 - INFO - Connecting to: 192.168.1.137, on port: 9001 using the following client value]-
2019-7-15 19:37:42 - INFO - Connection Success [URI: 192.168.1.137:9001, ID: clientID-ffMQR]
2019-7-15 19:37:45 - INFO - Subscribing to: [Topic: room/#]

```

Figura 5.1: GreenHouse site

GreenHouse

[Link Grafana](#)

Hostname or Ip Address: Port:

Connected to: 192.168.1.137:9001 as clientID-ffMQR

CONNECT DISCONNECT PICTURE



Subscribe

Topic:

SUBSCRIBE UNSUBSCRIBE

#	Last measurement: 2019-7-15 19:51:26	Timestamp	Topic	Value
1	2019-7-15 19:41:25		room/planta_1/temperature	23.6
2	2019-7-15 19:41:25		room/planta_1/humidity	65.5
3	2019-7-15 19:41:25		room/planta_1/lumen	65
4	2019-7-15 19:41:26		room/planta_1/grometro	58
5	2019-7-15 19:46:25		room/planta_1/temperature	23.4
6	2019-7-15 19:46:25		room/planta_1/humidity	65.7
7	2019-7-15 19:46:25		room/planta_1/lumen	65
8	2019-7-15 19:46:26		room/planta_1/grometro	58

Subscribing to room/#

Figura 5.2: GreenHouse site

6 Conclusione

Tale progetto ha permesso di comprendere in modo migliore il mondo IoT e i vari paradigmi e protocolli legati ad esso.

Smart GreenHouse è un primo prototipo ma, molti sono gli sviluppi futuri che possono essere apportati affinché diventi più vantaggioso; di seguito si relazionano alcune idee:

- una luce per garantire sempre condizioni ideali di luminosità alla pianta.
- una ventola per garantire sempre delle temperature ideali al fine di incentivare la crescita della pianta.
- un sistema in grado di dare fertilizzante e sali minerali alla pianta al fine di tenerla sempre in forze ed un sistema in grado di analizzare biologicamente lo stato della pianta.

Tramite il link seguente è possibile reperire il codice utilizzato per tale progetto: IOT.git.