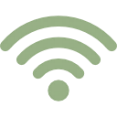
Corso di Sistemi Embedded e Internet-Of-Things

Laurea Triennale in Ingegneria e Scienze Informatiche

Università di Bologna - Cesena

anno accademico 2018/2019

*Progetto di una Smart Greenhouse*



Docente: Prof. Alessandro Ricci

Studenti:

Francesco Boschi ([francesco.boschi2@studio.unibo.it](mailto:francesco.boschi@studio.unibo.it))

Konrad Gomulka ([konrad.gomulka@studio.unibo.it](mailto:konrad.gomulka@studio.unibo.it))

Elizabeta Budini ([elizabeta.budini@studio.unibo.it](mailto:elizabeta.budini@studio.unibo.it))

Sommario

[Sommario 2](#_Toc2180357)

[Capitolo 1. Specifiche progetto 3](#_Toc2180358)

[Capitolo 2. Schema del circuito 4](#_Toc2180359)

[Capitolo 3. GreenHouse Controller (Arduino) 4](#_Toc2180360)

[3.1 Detection Task 5](#_Toc2180361)

[3.2 Irrigation Task 5](#_Toc2180362)

[Capitolo 4. GreenHouse Server (PC) 6](#_Toc2180363)

[4.1 Comunicazione seriale con Arduino 6](#_Toc2180364)

[4.2 Comunicazione IP/http con ESP 6](#_Toc2180365)

[Capitolo 5. GreenHouse Edge (ESP) 7](#_Toc2180366)

[Capitolo 6. GreenHouse Mobile App (Android) 7](#_Toc2180367)

[6.1 Connessione 8](#_Toc2180368)

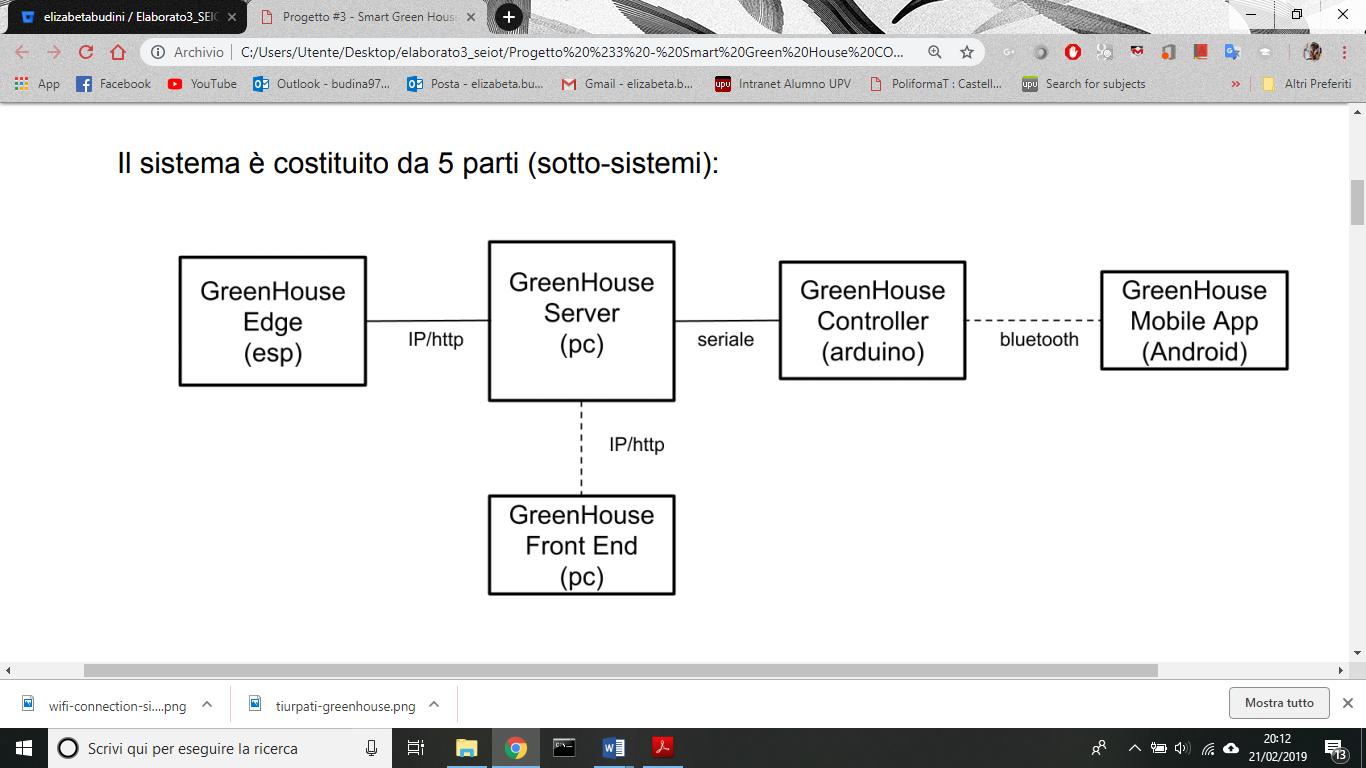
[6.2 Modalità manuale 9](#_Toc2180369)

[6.3 Comunicazione Bluetooth con Arduino 10](#_Toc2180370)

[Capitolo 7. GreenHouse Front End (PC) 10](#_Toc2180371)

# Capitolo 1. Specifiche 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 (sotto-sistemi):

Figura

- **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 (pompe acqua), quindi

della quantità di acqua erogata al minuto

- **GreenHouse Edge (ESP)**

- permette di percepire l’umidità del terreno

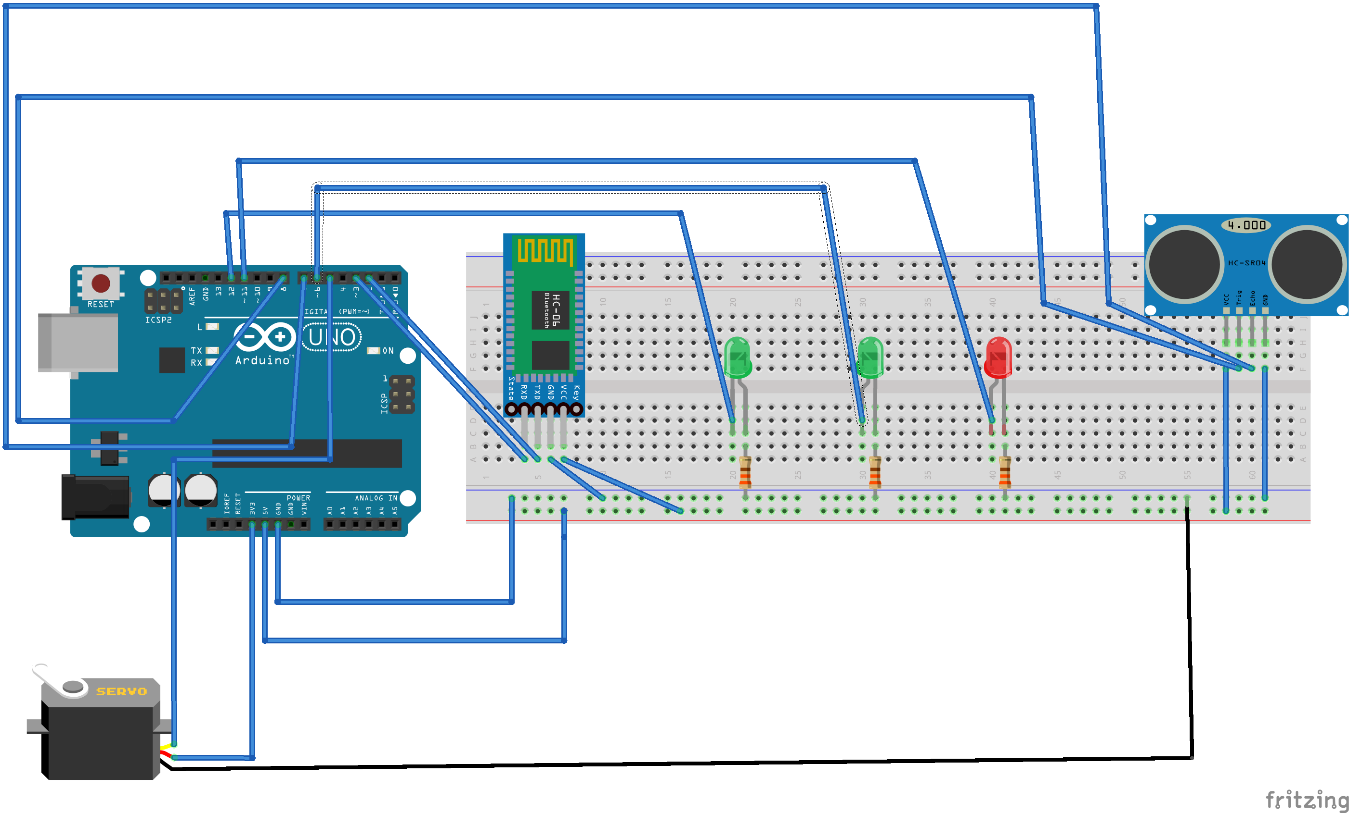
- **GreenHouse Mobile App (Android)**

- permette di controllo manuale della serra

- **GreenHouse Front End (PC)**

- Front end per visualizzazione/osservazione/analisi dati

# Capitolo 2. Schema del circuito



Figura

*Osservazioni:*

L’unica problematica riscontrata è stata quella riguardante il pin PWM utilizzato per il LED che riflette la portata e si accende con un’intensità variabile.

Inizialmente il LED utilizzava il pin 11, il quale è controllato dal Timer 2, che causava quindi conflitti con la libreria utilizzata per il servo motore.

# Capitolo 3. GreenHouse Controller (Arduino)

Per quanto riguarda l’implementazione lato Arduino, abbiamo deciso di utilizzare due task: *Detection Task*, utilizzato per controllare la vicinanza di qualcuno alla serra; *Irrigation Task*, incaricato di applicare le varie politiche di irrigazione in base alla modalità in cui si trova il sistema. Di seguito descriveremo più dettagliatamente l’implementazione di questi task e le rispettive macchine a stati finiti che descrivono il loro funzionamento. Prima di procedere, proponiamo una breve descrizione generale del sistema e del suo funzionamento.

Il sistema parte in modalità AUTOMATICA, in cui può avvenire l’irrigazione automatica. In questo caso il led L1 è acceso a indicare che il sistema è attivo, in modo AUTO, mentre L2 e Lm sono spenti. Quando viene percepito un valore di umidità U inferiore alla soglia Umin, viene aperta la pompa automaticamente erogando una certa portata (quantità di acqua nel tempo), pari a Y litri al minuto, dove Y può assumere tre valori: Pmin (portata minima), Pmed (media), Pmax (massima). Inoltre, quando viene erogata acqua, si accende L2 con intensità che riflette la portata. In modalità automatica l’erogazione si ferma quando o il valore supera la soglia Umin + un certo *DeltaU*, oppure se la durata dell’erogazione ha superato un tempo Tmax . In questo caso viene creata una segnalazione.

Collegandosi al Controller (Arduino) attraverso il bluetooth da Mobile App è possibile passare in modalità MANUALE se ci si trova nelle vicinanze della serra (la distanza rilevata dal sonar deve essere inferiore a 30 cm). In questa modalità è possibile manualmente aprire/chiudere/regolare l’erogazione dell’acqua, specificando la portata (litri al minuto) e visualizzare il valore in Real-Time dell’umidità percepita. Quando il sistema è in modalità di controllo manuale, il led L1 si deve spegnere e deve essere acceso Lm. In ogni momento è possibile visualizzare lo storico di irrigazione e umidità attraverso una pagina WEB descritta nel capitolo 1.7

## 3.1 Detection Task

Figura

Il primo task creato, nominato Detection Task, ha il solo compito di controllare la distanza percepita dal sonar, e settare una variabile globale usata per la comunicazione tra i due task.

## 3.2 Irrigation Task

Figura

Il secondo task invece, deve gestire la procedura di irrigazione, sia manuale che automatica.  
Abbiamo deciso di gestire in un solo task entrambe le modalità poiché condividono sensori e attuatori, oltre che la comunicazione seriale e bluetooth. Inoltre la logica delle due è analoga, con eccezione per le condizioni di inizio e fine irrigazione. Abbiamo inoltre deciso di aggiungere uno stato “IDLE” iniziale, che agirà da selettore per lo stato Automatico o Manuale, evitando di connettere questi ultimi direttamente. Come si può notare dallo schema sottostante, verrà utilizzata una variabile settata dagli stati che precedono l’irrigazione, così da considerare le condizioni adeguate per l’uscita dallo stato in cui viene innaffiata la serra.

# Capitolo 4. GreenHouse Server (PC)

Il lato server è stato realizzato in Java implementando un’architettura ad eventi che comunica con Arduino via seriale e riceve messaggi dall’ESP via IP/HTTP.

Vi sono quattro classi principali:

* **GreenHouseController**: riceve gran parte degli Event generati dalle altre classi e se il sistema è in modalità AUTOMATICA attua le opprtune azioni
* **GreenHouse** tramite la propria classe MsgService gestisce gli Event che richiedono la comunicazione con Arduino
* **Dataservice** gestisce l’ascolto della porta relativa a ngrok permettendo cosi’ di ricevere i messagi inviati dall’ESP
* **ESP** classe Observable adibita alla comunicazione con l’ESP, si occupa di controllare i valori di umidita’ ricevuti e se necessario di attivare l’irrigazione automatica

## 4.1 Comunicazione seriale con Arduino

Per quanto riguarda la comunicazione con il modulo Arduino vengono ricevuti messaggi quando esso passa dalla modalità manuale a quella automatica e viceversa, quando inizia e termina l’irrigazione in modo da poter salvare le relative informazioni su durata e portata, in particolare differenziando quando l’arresto dell’irrigazione è avvenuto per Over-time, caso in cui è necessario notificarlo all’utente attraverso l’interfaccia.

|  |  |  |  |
| --- | --- | --- | --- |
| **Messaggi inviati** | **Significato** | **Messaggi ricevuti** | **Significato** |
| “Start0” | Accendi pompa con portata minima | “ManIn” | Modalità manuale |
| “Start1” | Accendi pompa con portata media | “ManOut” | Modalità Automatica |
| “Start2” | Accendi pompa con portata massima | “Start” | Conferma inizio irrigazione |
| “Stop” | Spegni pompa | “Stop” | Conferma terminata irrigazione |
|  |  | *“StopT”* | Terminata irrigazione per Overtime |

Tabella

## 4.2 Comunicazione IP/http con ESP

Per la comunicazione con ESP utilizziamo invece un Dataservice che riceve tutti i messaggi inviati a Ngrok sulla porta 8081, il messaggio ricevuto dovra’ essere in formato JSON con un valore Umidita’ pari ad un numero intero rappresentante la percentuale di umidità rilevata da ESP. Una volta ricevuto il messaggio tramite la classe Observable “ESP”, si controlla se il valore ricevuto è inferiore al valore minimo di umidità e nel caso in cui questo avvenga, viene inviato un notifyEvent per richiedere l’irrigazione automatica. La classe controller “GreenHouseController”, nel caso il sistema sia in modalità AUTOMATICA, dopo aver ricevuto l’Event citato sopra, ha il compito di attuare l’irrigazione automatica comunicandolo al modulo Arduino.

*Note:* per avviare il modulo server sul proprio PC è necessario modificare nel codice del Main il valore della porta connessa a Arduino e la porta relativa a NGROK.

Se si sta utilizzando Linux i dati verranno salvati nella directory Documents, altrimenti è necessario modificare il path di destino del salvataggio nelle classi Dataservice.java E GreenHouse.java.

# Capitolo 5. GreenHouse Edge (ESP)

Figura

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.

*Note:* All’avvio del sistema dal proprio PC, è necessario modificare nel codice la rete wi-fi, la password e l’indirizzo del server Ngrok in esecuzione.

# Capitolo 6. GreenHouse Mobile App (Android)

L’applicazione permette il controllo manuale della serra quando si è nelle sue vicinanze a distanza inferiore di circa 30 centimetri. In modalità manuale è possibile controllare l’irrigazione e visualizzare il valore corrente dell’umidità percepita in real-time.

## 6.1 Connessione



Figura

All’avvio dell’applicazione nella parte superiore dello schermo è possibile connettersi alla serra cliccando sul bottone “Connetti alla serra”. Perché vada a buon fine questo tentativo è necessario:

- attivare il Bluetooth del proprio dispositivo Android

- effettuare l’accoppiamento tra il proprio dispositivo è l’HC-06 (prima volta)

-assicurarsi di aver collegato il modulo Arduino a un PC e di aver caricato sulla scheda il codice presente nella cartella sgh-controller

Si dà per scontato che l’applicazione sia sviluppata in stretta correlazione alla tipologia di serra, pertanto non è possibile da app collegarsi ad altri dispositivi differenti dall’HC-06 (serra in questione). Nel caso sia necessario modificare il nome del dispositivo 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";

Andando nello specifico, a questo punto 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** deputato ad 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. Il codice UUID è modificabile dalla classe com.example.sgh\_app.**Utils** con la stessa modalità con cui si può modificare il nome del server descritta precedentemente. Il metodo di creazione della socket createRfcommSocketToServiceRecord() può essere chiamato su una qualsiasi istanza di BluetoothDevice, ma nello specifico deve essere utilizzata quella dell’istanza di BluetoothDevice che fa riferimento al device che esegue l’applicazione server.

## 6.2 Modalità manuale

Figura



Avvicinandosi alla serra

A questo punto siamo connessi alla serra e se ci avviciniamo al sonar l’app rileva il passaggio alla modalità manuale e attiva tutti i comandi disponibili:

* Bottone “Accendi pompa”
* Bottone “Spegni pompa”
* SeekBar per modificare la portata

Nel caso ci si allontani dal sonar, la serra torna in modalità automatica e i comandi vengono disabilitati, ma la connessione Bluetooth persiste e quindi avvicinandosi alla serra si può usufruire nuovamente dei comandi disponibili. In modalità manuale, quando la pompa è spenta non è possibile utilizzare il bottone “Spegni pompa”, stesso discorso per il bottone di accensione quando la pompa è già accesa. Per quanto riguarda la modifica della portata, nonostante sia possibile cambiarla in qualsiasi momento purché l’applicazione si trovi in modalità manuale, l’effettivo cambiamento sulla serra avviene all’irrigazione successiva. Ovvero prima di ogni irrigazione si controlla a che portata bisogna irrigare e questa rimane invariata fino al termine dell’irrigazione stessa.

## 6.3 Comunicazione Bluetooth con Arduino

Lo scambio di informazioni che avviene tra Arduino e il dispositivo Android è limitato messaggi riportati nella tabella seguente, alcuni sono utilizzati allo scopo di avere un feedback di cosa sta succedendo nella serra ma non sono stati esplicitamente richiesti dalla consegna (ad esempio Start e Stop che si riferiscono allo stato di irrigazione).

|  |  |  |  |
| --- | --- | --- | --- |
| **Messaggi inviati** | **Significato** | **Messaggi ricevuti** | **Significato** |
| “1” | Accendi pompa | “ManIn” | Modalità manuale |
| “2” | Spegni pompa | “ManOut” | Modalità Automatica |
| “3” | Portata minima | “Start” | Iniziato irrigazione |
| “4” | Portata media | “Stop” | Terminato irrigazione |
| “5” | Portata massima | *“Numero”* | Umidità percepita |
| “6” | Connesso Bluetooth |  |  |

Tabella

# Capitolo 7. GreenHouse Front End (PC)

Per il modulo Front End abbiamo creato una pagina web (HTML/PHP) in cui l’utente ha a disposizione due tabelle una che offre informazioni sullo storico di APERTURA/CHIUSURA della pompa e le transizione tra modalità MANUALE/AUTOMATICA registrate. La seconda tabella contiene lo storico dell’andamento del valore dell’umidità. Ogni evento ha associato data e ora di quando è accaduto. Come richiesto dalla consegna quando l’irrigazione termina per Over time, ovvero sono passati 5 secondi da quando è stata aperta la pompa, si notifica all’utente questo avvenimento. Tramite le funzionalità di Ajax, la pagina controlla e aggiorna ogni 5 secondi i contenuti delle tabelle in base alle informazioni presenti sul server.