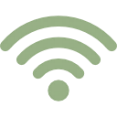
Corso di Sistemi Embedded e Internet-Of-Things

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](#_Toc1676186)

[1.1 Specifiche progetto 3](#_Toc1676187)

[1.2 Schema del circuito 3](#_Toc1676188)

[1.3 GreenHouse Controller (Arduino) 4](#_Toc1676189)

[1.3.1 Detection Task 5](#_Toc1676190)

[1.3.2 Irrigation Task 5](#_Toc1676191)

[1.4 GreenHouse Server (PC) 6](#_Toc1676192)

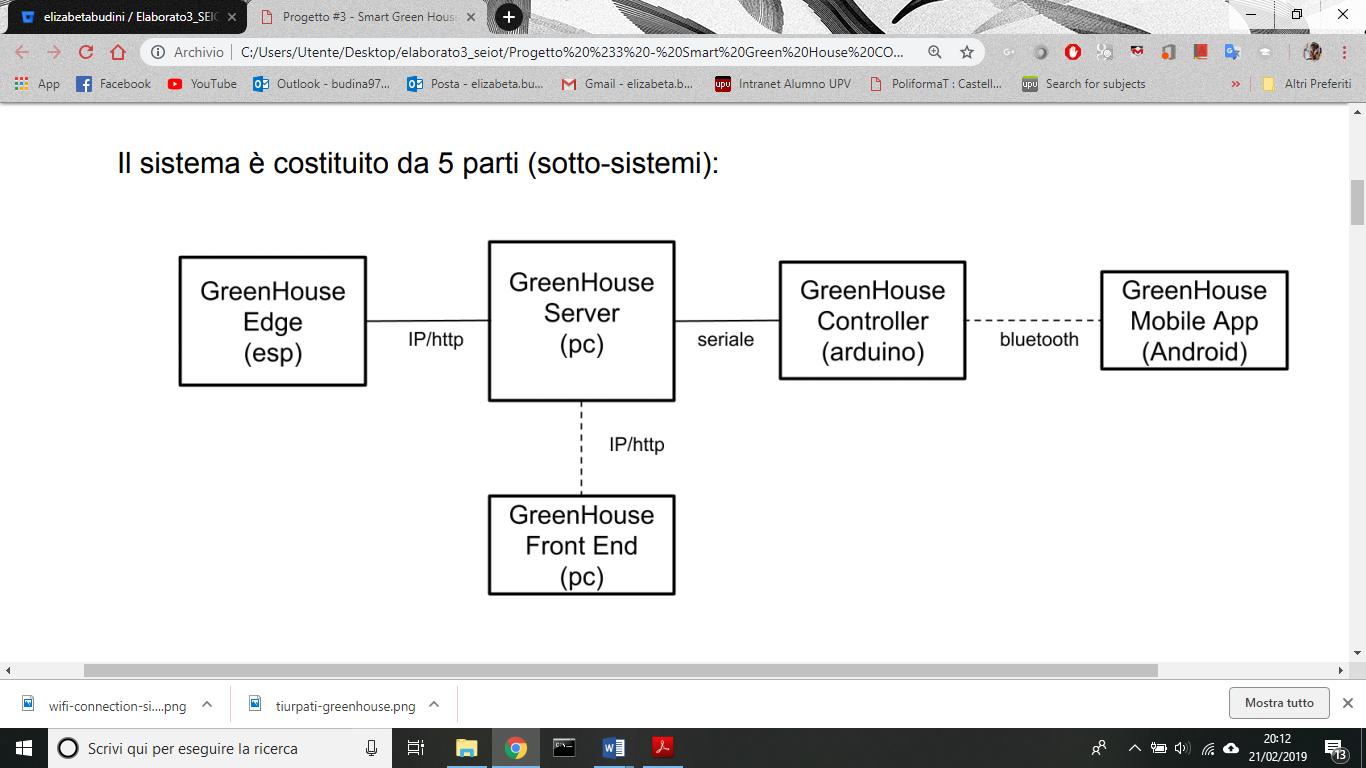
[1.5 GreenHouse Edge (ESP) 6](#_Toc1676193)

[1.6 GreenHouse Mobile App (Android) 6](#_Toc1676194)

[1.7 GreenHouse Front End (PC) 6](#_Toc1676195)

# 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):

- **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

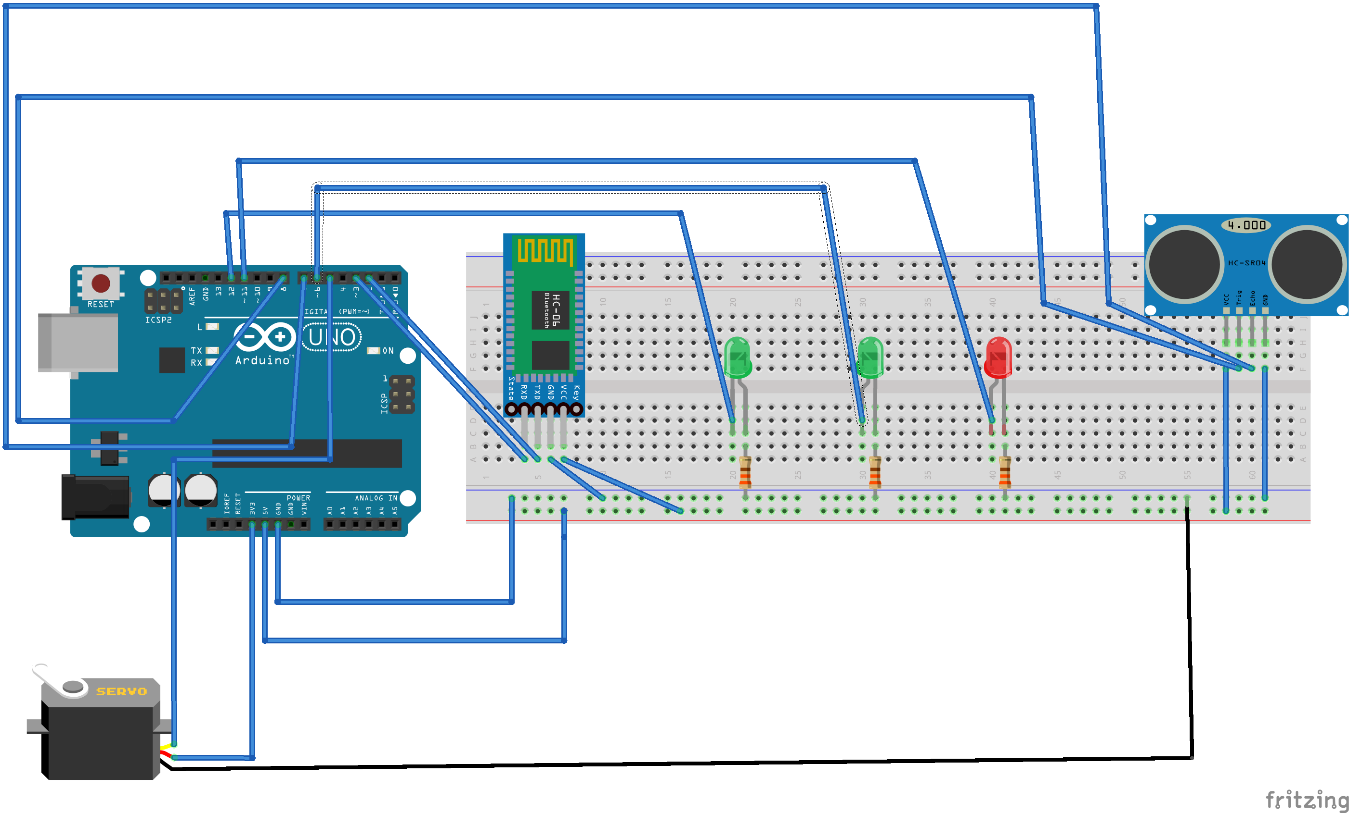
- **GreenHouse Mobile App (Android)**

- permette di controllo manuale della serra

- **GreenHouse Front End (PC)**

- Front end per visualizzazione/osservazione/analisi dati

# 1.2 Schema del circuito



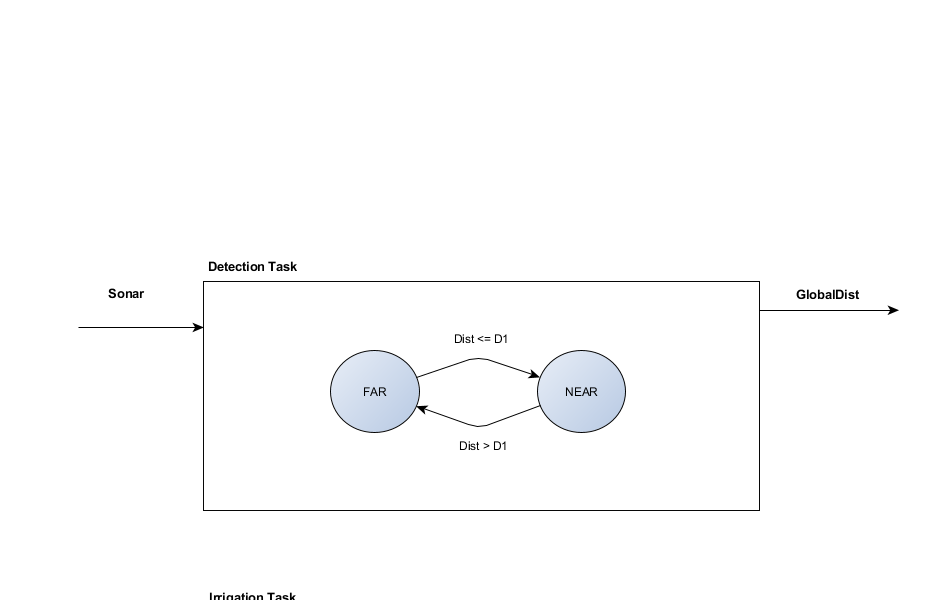
*Osservazioni:*

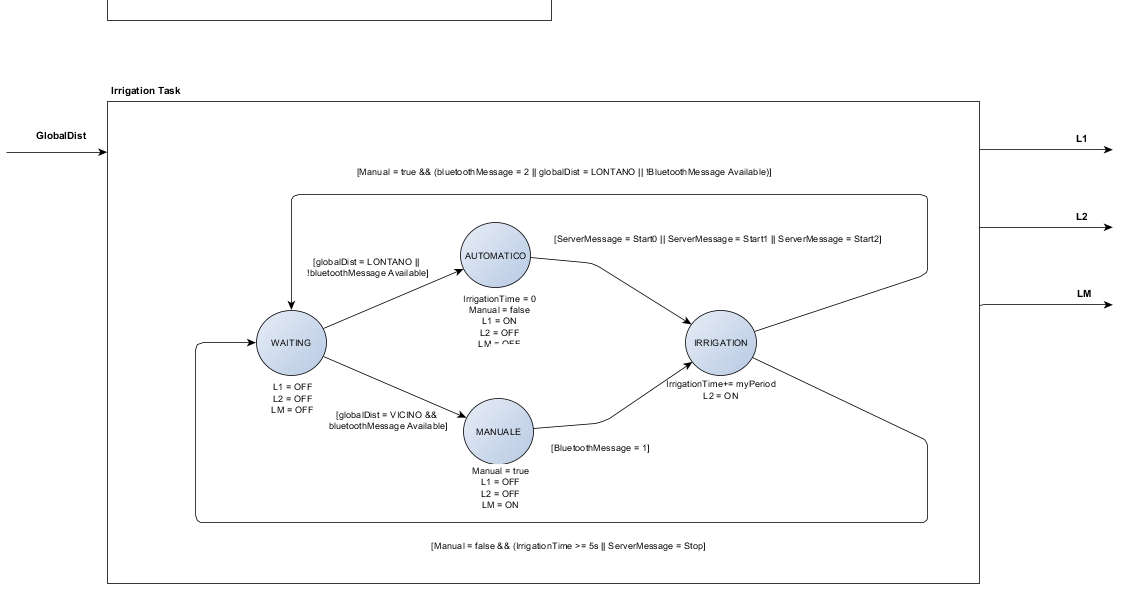
L’unica problematica riscontrata è stata quella riguardante il pin PWM utilizzato per il LED che riflette la portata e si accende con un’intensità variabile.

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.

# 1.3 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 panoramica del Controller e il funzionamento generale del sistema nelle sue modalità.

Il sistema presenta due modalità: MANUALE e AUTOMATICA. Il sistema parte in modalità AUTOMATICA, in cui avviene 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 .

## 1.3.1 Detection Task

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.

## 1.3.2 Irrigation Task

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.

# 1.4 GreenHouse Server (PC)

# Il lato server e’ stato realizzato in java implementando un’architettura ad eventi che comunica con l’arduino e riceve messaggi dall’ESP.

# Vi sono 4 classi principali:

# -GreenHouseController: riceve gran parte degli event generati dalle altre classi e se e’ in modalita’ AUTO attua le opprtune azione

# -GreenHouse tramite la propria classe MsgService gestisce gli ev 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 creata per comodita’ dato che era necessario avere una classe Observable relativa alla comunicazione con l’esp, si occupa di controllare i valori di umidita’ ricevuti e nel caso notificare la necessita’ di irrigazione

# Per quanto riguarda l’arduino vengono ricevuti messaggi quando esso passa dalla modalita’ manuale a quella automatica e viceversa, quando accende e spegne effettivamente l’irrigazione in modo da poter salvare le relative date in particolare differenziando quando l’arresto dell’irrigazione e’ avvenuto per Overtime.

# Per l’ESP abbiamo invece un Dataservice che riceve tutti i messaggi inviati a ngrok sulla porta 8081, il messaggio ricevuto dovra’ essere un JSON con un valore Umidita’ pari ad un int rappresentante la percentuale, una volta ricevuto il messaggio tramite la classe observable “ESP” viene controllato se e’ sotto il valore minimo e nel caso viene mandato un notifyEvent per richiedere l’irrigazione. Il controller principale “GreenHouseController” che memorizza lo stato dell’Arduino (AUTO,MANUAL) nel caso sia AUTO una volta ricevuto l’event crea a sua volta una notifyevent contenente le informazioni riguardanti l’apertura della pompa di irrigazione per il controller della comunicazione con l’arduino “GreenHouse” e sara’ esso infatti a richiederla mandando un messagio “Start”.

# ATTENZIONE: PER AVVIARE IL LATO SERVER SARA’ NECESSARIO MODIFICARE IL VALORE DELLA PORTA CONNESSA CON ARDUINO E PORTA RELATIVA A NGROK DAL MAIN.

# PER QUANTO RIGUARDA I DATI SALVATI ESSI VERRANNO SALVATI IN DOCUMENTS NEL CASO SI USI LINUX ALTRIMENTI E’ RICHIESTO DI MODIFICARE LA PATH DAI FILE Dataservice.java E GreenHouse.java

# 1.5 GreenHouse Edge (ESP)

Il funzionamento dell’ESP è molto semplice.  
Ogni 1,5 secondi leggerà il valore rilevato dal potenziometro, che in questo caso simula un sensore di umidità, e lo invierà al server tramite Ngrok in un messaggio strutturato come un Json.

# 1.6 GreenHouse Mobile App (Android)

# 1.7 GreenHouse Front End (PC)