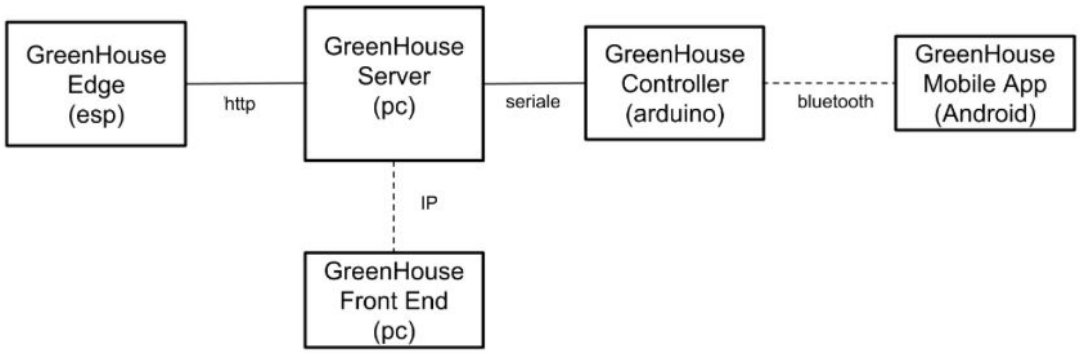
Mattia Vincenzi 792970
Rei Beshiri 789973

SmartGreenHouse

PANORAMICA

Nell'immagine sotto riportata abbiamo indicato i metodi di comunicazione adottati tra le varie componenti del sistema.



GreenHouseEdge (esp)

Questa componente rappresenta la parte di sistema adibita alla rilevazione dell'umidità e alla sua pubblicazione sul ***Data Service***. Per svolgere quest'ultimo compito abbiamo utilizzato come protocollo HTTP, che mediante il metodo POST pubblica il valore dell'umidità precedentemente ottenuto in formato JSON. Per la sua realizzazione ci siamo serviti dello stack espruino visto a lezione.

Per fare in modo di pubblicare i dati sul ***Data Service***, esponendo quest'ultimo nella rete pubblica, abbiamo utilizzato il servizio **ngrok**.

GreenHouseServer (pc)

1. Data Service

Utilizzato il codice fornitoci dal prof. abbiamo implementato il ***Data Service*** come servizio di supporto per il mantenimento dei dati relativi all'umidità raccolti dal **GreenHouseEdge**. Il ***Data Service*** si occupa di rispondere e gestire le richieste in GET e in POST che gli vengono inviate.

Per il deploy e la comunicazione con il ***Data Service*** abbiamo utilizzato il tool basato sulla JVM **Vertx**, fornitoci dal prof.

2. Back-end

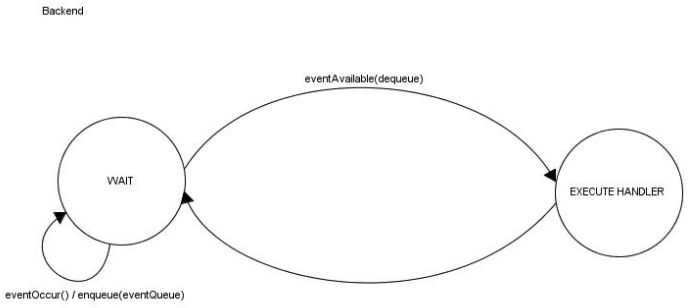
Per la realizzazione della logica server-side abbiamo utilizzato un approccio ad **Agenti**, basato sul pattern **event-loop**, il cui comportamento è determinato da una macchina a stati asincrona.

L' **Agente** essendo un'entità attiva diventa osservatore dell'ambiente (**Observer**), mentre gli oggetti quali canali di comunicazione, timer ecc.. sono le entità passive sorgenti di eventi (**Observable**). Questo comportamento è implementato mediante *event-loop*, dove si disaccoppia il flusso di controllo che genera gli eventi e quello che esegue gli handler, creando quindi una bufferizzazione di eventi mediante una coda bloccante.

Observable:

1. **SerialChannelObservable**: è l'entità adibita alla gestione del canale di comunicazione seriale, che alla ricezione di un messaggio crea un evento e lo accoda al buffer dell'agente.
2. **SocketChannelObservable**: è l'entità adibita alla gestione del canale di comunicazione mediante socket UDP, che alla ricezione di un datagramma accoda un evento.
3. **TimerObservable**: è l'entità che si occupa di generare un evento periodicamente, che detta i tempi per le richieste al Data Service.

L' **Observer** è una unica entità, il cui comportamento è descritto dalla FSM asincrona.

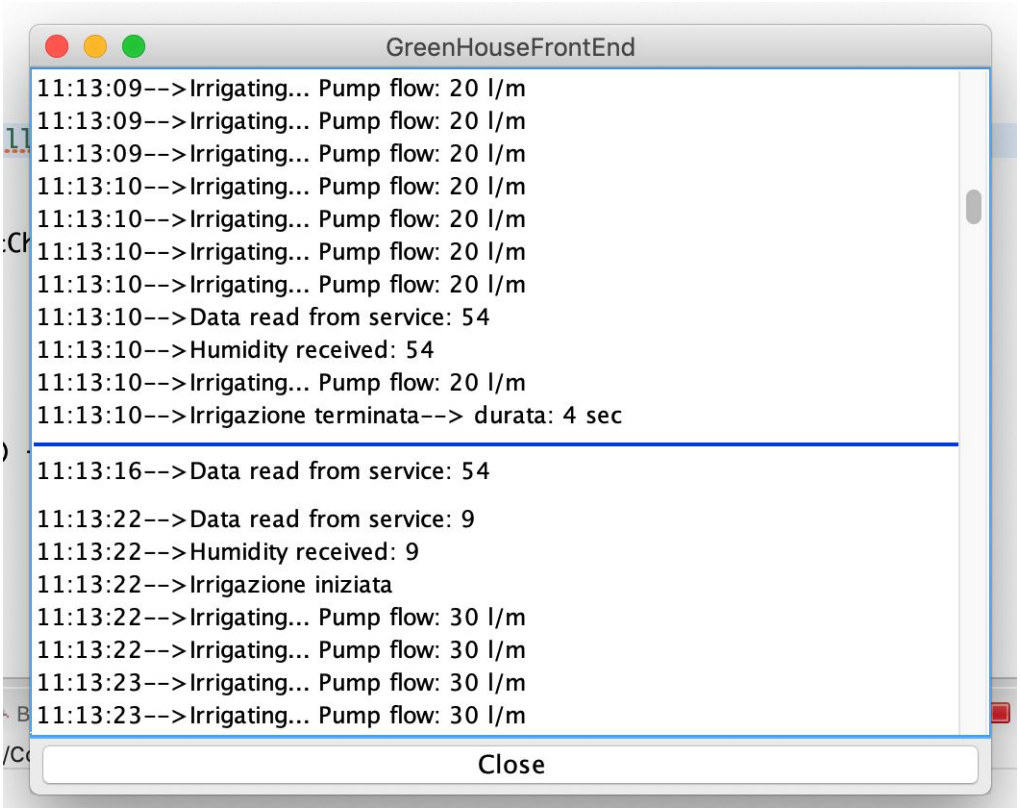


Internamente quando un evento viene estratto dalla coda e viene processato, allora viene eseguito l'handler corrispondente alla classe dell'evento. Questo implica una fase di registrazione iniziale degli Handler da eseguire in risposta ad ogni tipo di evento.

GreenHouseFrontEnd (pc)

Il front-end si occupa semplicemente della richiesta periodica di dati al server mediante una socket UDP e della loro visualizzazione.

Questa componente comunica solamente con il **GreenHouseServer** seguendo i principi dell'architettura *client-server,* di conseguenza non sarà mai il server ad iniziare una comunicazione, ma risponderà solo a fronte di richieste esplicite da parte del client.



GreenHouseController (arduino)

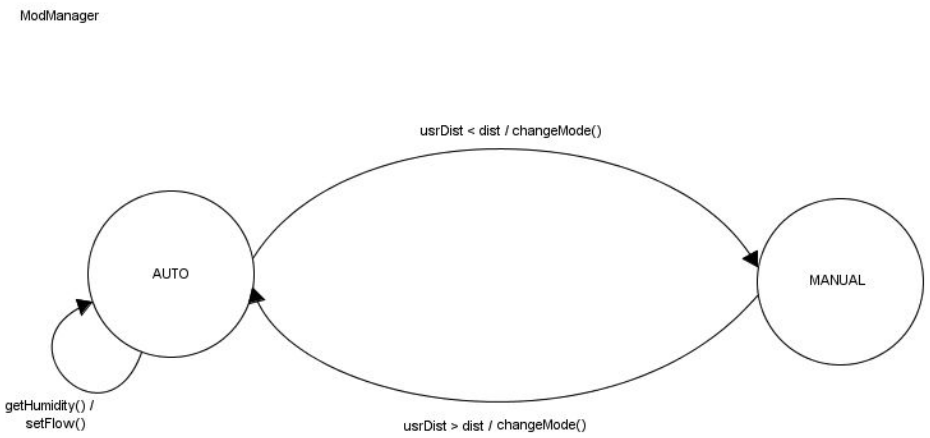
Abbiamo utilizzato un approccio a **task** con modello di comportamento basato su macchine a stati finiti sincrone.

1. ModManager Task

Questo task si occupa del mantenimento e del monitoraggio dello stato attuale della serra.

Il sistema parte in modalità automatica per poi passare a quella manuale alla rilevazione di un utente nelle vicinanze. Viceversa quando non viene rilevato nessuno nei pressi della serra per un certo periodo di tempo ritorna alla modalità automatica.

Abbiamo assegnato a questo task un periodo di 100mS per evitare la perdita di eventi; inoltre ci sembrava opportuno come frequenza per l'invio di messaggi al server. Essendo il task che gestisce le modalità del sistema a nostro avviso doveva avere la priorità massima (primo nella coda).



2. Irrigation Task

Questo task si occupa della gestione dell'irrigazione, e quindi dell'accensione e dello spegnimento della pompa dell'acqua.

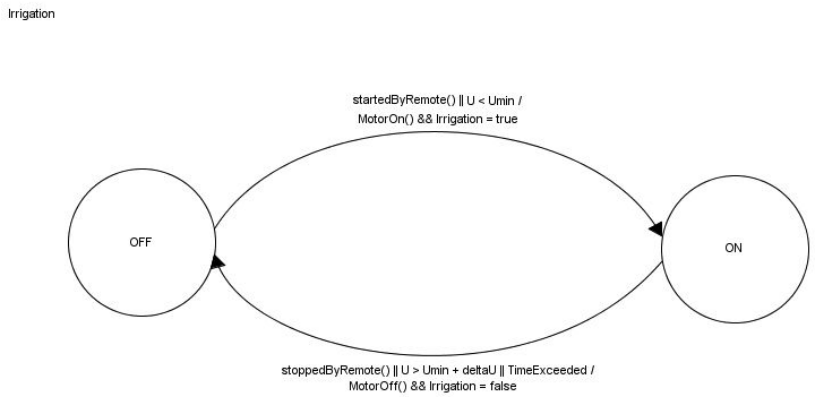
L’irrigazione può avvenire (con il passaggio della macchina dallo stato OFF a ON) quando:

1. La serra è in modalità automatica e l’umidità ricevuta dal **GreenHouseServer** è minore alla soglia minima.
2. La serra è in modalità manuale e l’utente invia il comando di irrigazione.

L’irrigazione può essere interrotta (con il passaggio della macchina dallo stato ON a OFF) quando:

1. La serra è in modalità automatica e viene percepita un’umidità superiore del 5% a quella minima.
2. La serra è in modalità automatica e l’irrigazione va avanti per un tempo superiore a quello massimo.
3. La serra è in modalità manuale e l’utente invia il comando di spegnimento.

Anche a questo task abbiamo assegnato il periodo di 100mS, perchè supponendo di essere in un sistema reale con una serra molto estesa e delle pompe a grande portata, la perdita di un evento quale il cambio di modalità del sistema, oppure la ricezione di una umidità sufficientemente alta all'interruzione dell'erogazione, potrebbe portare a grandi sprechi, per via del mantenimento della pompa aperta per un tempo maggiore del necessario.



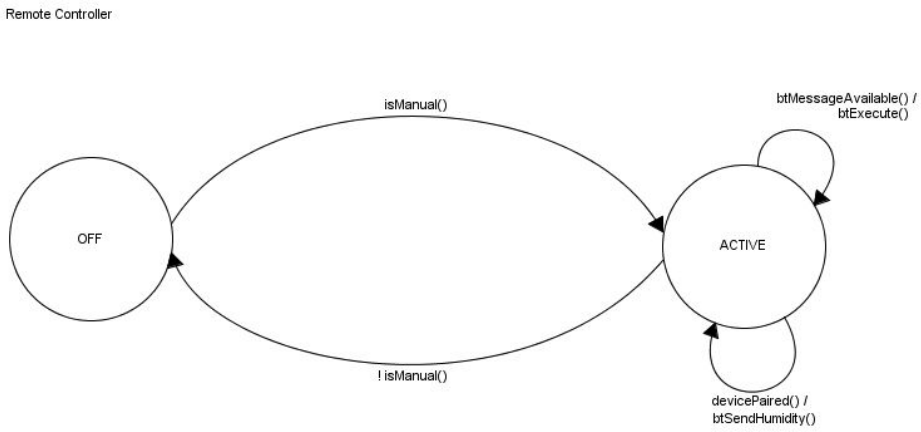
3. Remote Task

Questo task si occupa della comunicazione bluetooth tra il **GreenHouseController** e la **GreenHouseMobileApp**.

Questo task opera quando il sistema è in modalità manuale (passando dallo stato OFF a quello ACTIVE), e alla ricezione di un messaggio dal canale bluetooth esegue l'opportuna azione.

Inoltre, in questa modalità, verrà periodicamente inviato il valore dell'umidità (sempre mediante il canale bluetooth) alla **GreenHouseMobileApp**.

Anche per il Remote Task abbiamo optato per il periodo di 100mS, in modo che l'utente abbia un feedback/controllo immediato del sistema. (Ad esempio avendo il *ModManger* lo stesso periodo non ci perdiamo eventi come potrebbe essere l'avvicinamento dell'utente con il conseguente cambiamento di modalità).



Nota

Durante la progettazione di questo sottosistema, abbiamo mantenuto un alto livello di astrazione, cercando di rappresentare le risorse dal punto di vista dell'utente (ad esempio: userConsole, FeedbackState ecc...) e al loro interno utilizzare i vari sensori/attuatori.

GreenHouseMobileApp (android)

Per lo sviluppo dell'applicazione abbiamo utilizzato android e le classi di supporto forniteci dal prof.

L'applicazione permette di connettersi al modulo bluetooth collegato ad arduino e di inviare i comandi di irrigazione (con l'opportuna portata della pompa) e di spegnimento del sistema di irrigazione.

Abbiamo deciso di mantenere memorizzato l'accoppiamento dei due dispositivi mandando in fase di inizializzazione e di terminazione dei messaggi di sincronizzazione. Inoltre in un'apposita

area è possibile visualizzare il valore dell'umidità inviato periodicamente come feedback dalla serra .