Assignment#3
Corso: Sistemi Embedded e Internet of Thing

Andrea Micheli

23 marzo 2023

1 Analisi

Si vuole realizzare un sistema IoT che implementi una versione semplificata di uno smart garden. Il sistema è composto da cinque sottosistemi:

- ESP, utilizzato per rilevare dati riguardanti temperatura e luminosità ambientale e comunicarli tramite MQTT al backend (applicazione in esecuzione sul PC).
- Backend - PC, utilizzato per la gestione del sistema; comunica tramite serial line all'Arduino, tramite MQTT alla sensor board (ESP) e gestisce la pagina Web.
- Arduino, sistema embedded che gestisce il sistema di irrigazione e le luci del giardino; comunica tramite serial line con il Backend e tramite bluetooth all'applicazione cellulare.
- Applicazione cellulare, sistema in grado di ottenere controllo manuale delle luci o del sistema di irrigazione e di disattivare l'allarme di sistema; comunica tramite Bluetooth con Arduino.
- Applicazione web, permette di visualizzare lo stato del sistema, delle luci e il sistema di irrigazione; comunica con il Backend per ottenere le informazioni necessarie.

Il giardino viene gestito in due diversi modi: automatico o manuale.

- In modalità automatica il sistema rileva periodicamente luminosità e temperatura, ottenendo le rilevazioni dall'ESP, e aggiusta in modo autonomo le impostazioni di luci e irrigazione.
  Le luci seguono il ciclo del giorno e quindi vengono spente durante le ore più luminose della giornata e vengono accese nel momento in cui la luminosità ambientale supera un soglia preimpostata. Due luci possono essere solo accese o spente mentre le restanti due hanno un'intensità regolabile.
  Il sistema di irrigazione viene acceso durante la notte e la velocità viene dettata dalla temperatura rilevata. Una volta terminato il ciclo di irrigazione il sistema entra in uno stato di riposo. Nell'evento in cui la temperatura superi una soglia prefissata e il sistema di irrigazione fosse a riposo, lo stato del giardino entrerebbe in allarme richiedendo la supervisione di un utente.
- In modalità manuale è l'utente che gestisce ogni elemento del sistema. Una volta ottenuto il controllo del sistema tramite l'applicazione cellulare, l'utente sarà in grado di spegnere o accedere le luci, impostare una velocità di irrigazione, decidere se irrigare o meno il giardino e, in caso fosse necessario, eliminare uno stato di allarme.
  Una volta terminato l'uso dell'applicazione e rilasciato il controllo, il sistema tornerà in modalità automatica.

2 Architettura

Dopo un'analisi del comportamento dettagliato del progetto, ho deciso di iniziare dallo sviluppo di Arduino e del backend: Arduino per la gestione dei componenti e il backend per la gestione del sistema e la relazione tra tutti i sottosistemi.

Per la gestione autonoma del sistema, ho deciso di implementare una macchina a stati finiti che combaciasse con il ciclo del giorno/notte di ogni giorno: durante le ore più luminose della giornata i componenti rimangono inattivi per poi attivarsi uno dopo l'altro con il calare del sole, le luci in primis e il sistema di irrigazione durante le ore più buie. Ogni stato, una volta completato il proprio compito, riporta il sistema in uno stato di attesa del prossimo evento (cambio di luminosità ambientale).
A differenza della gestione autonoma, quella manuale è stata suddivisa in base alle possibili richieste al sistema, poste dall'utente: accendere, spegnere o cambiare l'intensità delle luci e la gestione del sistema di irrigazione. Una volta completata ogni azione anche questo sistema torna in uno stato di attesa del comando successivo.
In caso il sistema dovesse entrare in uno stato di allarme viene impedita qualsiasi modifica ai componenti fino a quando lo stato non viene risolto da un utente.

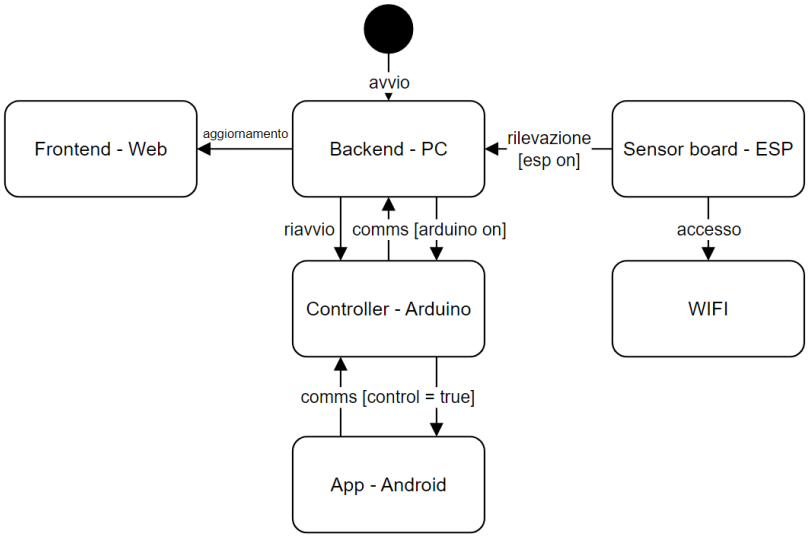
Per quanto riguarda l'implementazione del backend, inizialmente ho deciso di focalizzarmi sullo sviluppo del sistema di comunicazioni fra serial line, Bluetooth e MQTT e la realizzazione di codici concisi, utili alla trasmissione di dati.
Una volta che sono stato in grado di comunicare i dati necessari al funzionamento e al monitoraggio del corrente stato di ogni componente, mi sono dedicato alla loro rappresentazione sotto forma di file JSON, utilizzato per la realizzazione della pagina web. Ogni volta che viene apportata una modifica ad un componente, il file JSON viene sovrascritto con i nuovi dati. Alla chiusura dell'applicazione di backend, il file JSON viene riportato al suo stato predefinito (tutti i componenti spenti e il sistema in uno stato nominale).
Prima di implementare la sezione di MQTT, mi sono dedicato allo sviluppo della sensor board (ESP).

Per poter adattare al meglio le specifiche di progetto con quanto imparato a lezione, ho deciso di suddividere le due operazioni, di monitoraggio e di comunicazione, in due processi: così facendo ogni compito viene eseguito su un core differente dall'altro.
Sul primo core ho implemento le operazioni di rilevazione di temperatura e luminosità ambientale, suddivise nei propri task. Il core prende turni nel rilevare e calcolare i dati raccolti tra temperatura e luminosità; per ottenere valori più fedeli alle condizioni ambientali, alle quali viene sottoposto il sistema, ho deciso di accumulare i dati raccolti ed eseguire una media nel momento in cui si vogliono comunicare i valori registrati. L'ESP invierà un messaggio, contenente i risultati ottenuti, ogni 5 secondi.
Sul secondo core ho invece implementato il sistema di messaggistica MQTT. Per ottenere i dati raccolti sul primo core dell'ESP, tramite una programmazione ad oggetti, ho passato, come attributi i puntatori ai due task su esso in esecuzione (raccolta dati di temperatura e luminosità), ciò al momento dell'inizializzazione del servizio di comunicazione.
Questo approccio di programmazione mi ha però portato ad avere difficoltà di istanziazione dei task nei core dell'ESP: utilizzando una programmazione ad oggetti il tipo di parametro passato per generare un task su uno dei core, non è accettato in quanto viene richiesto un riferimento ad una funzione di tipo "classless" (o *(void)(*)). Per ottenere il risultato atteso, ho quindi dovuto implementare funzioni di tipo "inline", ma esterne alla classe del task.

Completata l'implementazione dei sistemi appena esposti, e testato la modalità automatica del sistema, mi sono dedicato alla programmazione dell'applicazione Android.
L'applicazione comunica tramite Bluetooth con Arduino. Quando l'utente necessità della gestione del sistema, l'applicazione richiede controllo ad Arduino che, a sua volta, notifica il backend di questa modifica. Una volta confermato il passaggio del testimone, l'utente sarà in grado di applicare modifiche allo stato corrente dei componenti del giardino. Per notificare Arduino di una richiesta di cambiamento, l'applicazione Android, ad ogni pressione di un pulsante sulla GUI, invia un messaggio Bluetooth con il relativo comando.

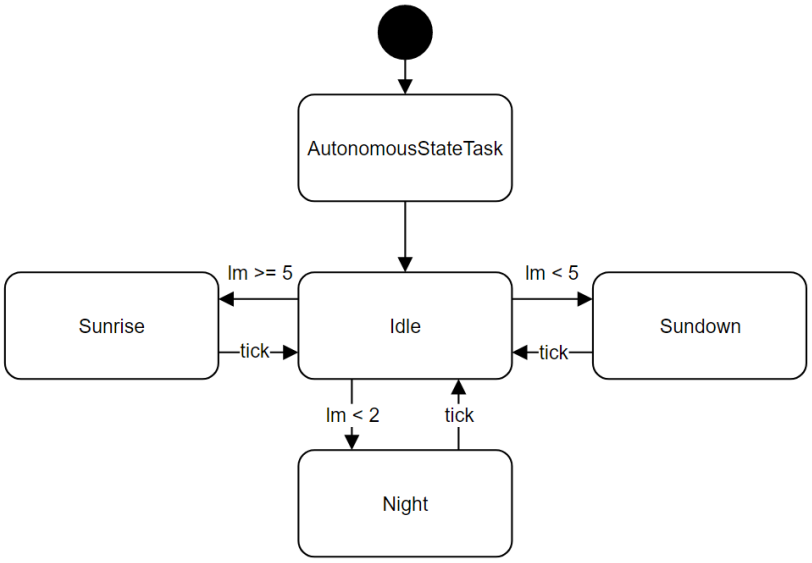
Successivamente mi sono dedicato alla creazione della pagina web, la quale mostra il corrente stato del sistema, le luci e il sistema di irrigazione; per fare ciò ogni secondo aggiorna il proprio contenuto raccogliendo i dati presenti all'interno del file JSON che il backend modifica ad ogni cambiamento del sistema.
Infine mi sono dedicato all'aspetto grafico del sito.

3 Diagramma del progetto

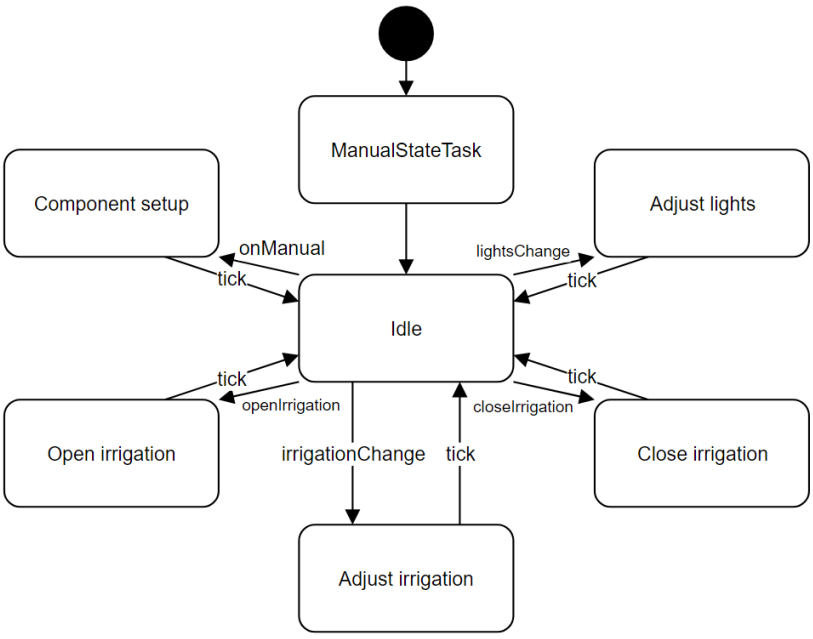


4 Diagramma degli stati

4.1 AutonomousStateTask

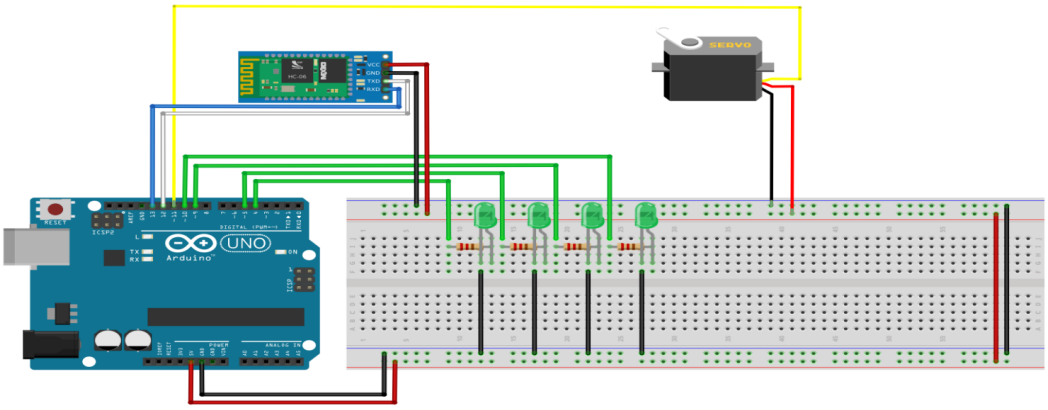


4.2 ManualStateTask



5 Schema

5.1 Arduino



5.2 ESP