SmartRoom

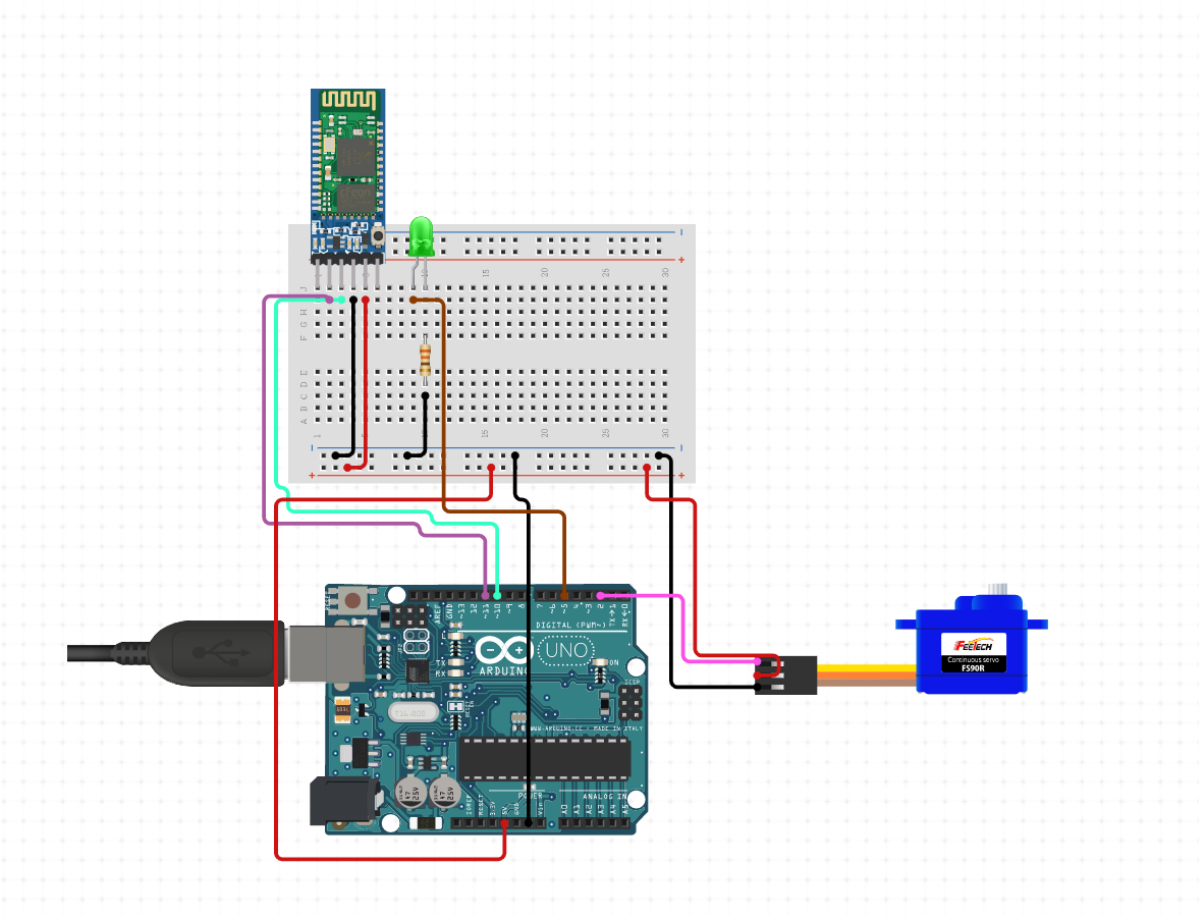
Embedded Systems And Internet of Things

Assignment #2

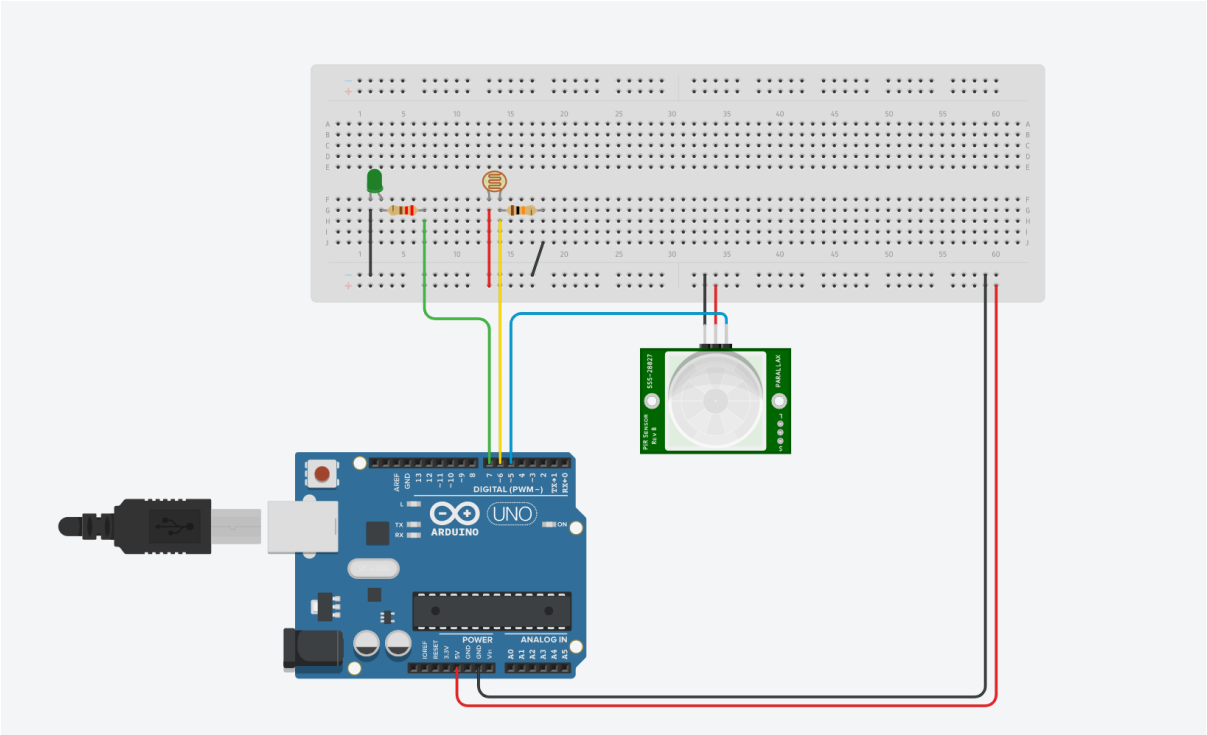
Davide Tonelli

Matr: 0000970464

1. Circuit:



Representation of the Room Controller schema



Representation of the Room Sensor-Board schema

Nota:Tinkercad non permette di realizzare schemi utilizzando un Esp32 quindi, per motivi didattici, lo schema presenta un Arduino Uno. Nella realtà si è usato un Esp32.

2. Architettura software:

Room Controller:

Scritto in C++, il software è organizzato nelle seguenti cartelle:

- **core**: contiene le classi ed interfacce alla base del suo funzionamento
- **devices**: contiene le classi ed interfacce dei singoli componenti hardware
- **model**: contiene le classi dei principali componenti del dominio applicativo
- **tasks**: contiene le classi delle tasks operanti nel sistema

SmartRoom è la classe rappresentante il dominio applicativo.
Agisce come un container memorizzando lo stato della stanza e dei suoi sensori.
Presenta 3 modalità di controllo:

- **AUTO**: il comportamento della stanza è automatizzato
- **MANUAL_SERIAL**: la stanza è controllata da remoto via seriale
- **MANUAL_BLUETOOTH**: la stanza è controllata da remoto via bluetooth

TimeService è una classe utilizzata come utility per mantenere e memorizzare l'ora attuale.

Si può utilizzare il file **config.h** per cambiare le impostazioni del software (es. la configurazione dei pin, il minimo livello di luce, …)

Room Sensor-Board:

La classe **RoomSensorBoard** ha un comportamento analogo a **SmartRoom**.
Per il trasferimento dati si è usato il protocollo MQTT.

Si può utilizzare il file **config.h** per cambiare le impostazioni del software (es. la configurazione dei pin, …)

Room Mobile App:

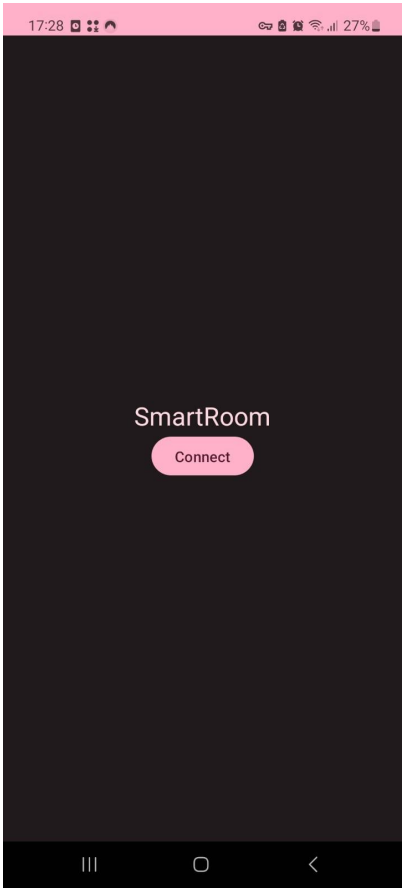
Scritta in Kotlin, il software è organizzato nelle seguenti cartelle:

- **bluetooth**: contiene le classi per il funzionamento del bluetooth
- **model**: contiene le classi dei principali componenti del dominio applicativo
- **ui.screen**: le funzioni riguardanti l'interfaccia grafica

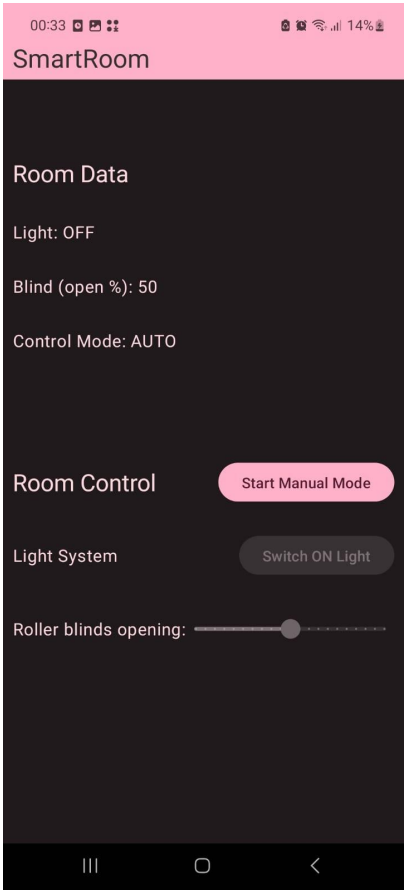
A livello di architettura delle classi si è utilizzato il pattern ViewModel:
La classe **SmartRoomViewModel** ha un funzionamento analogo a **SmartRoom** e presenta uno stato (composto dai valori della stanza) che permette di aggiornare dinamicamente l'interfaccia grafica.

Per ricezione e trasmissione dati viene utilizzato json.
L'unica eccezione sono i comandi per cambiare modalità di controllo della SmartRoom che presentano il seguente formato:

    chmod CONTROL_MODE



App welcome screen



App dashboard screen

Room Service:

Scritta in javascript utilizzando node js ed express, è un server che riceve aggiornamenti di stato dalla **Room Sensor-Board** e li propaga al **Room Controller**. Inoltre rende possibile utilizzare la **Room Dashboard**.

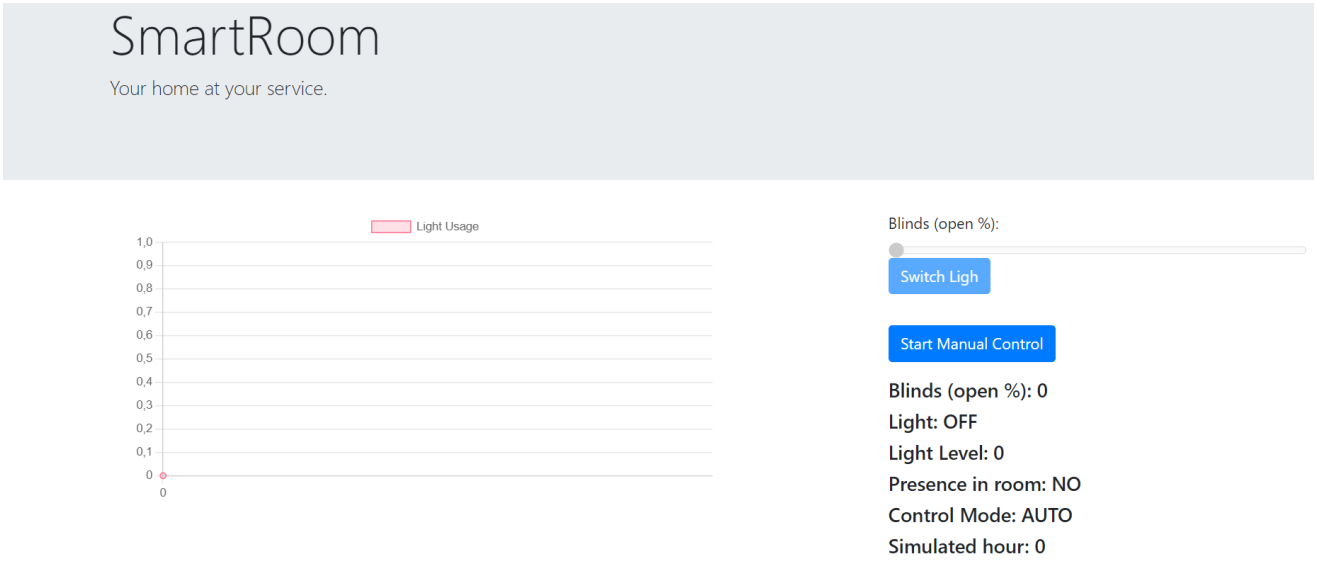
Ottiene dati dalla **Room Sensor-Board** utilizzando il protocollo MQTT.
Per ricezione e trasmissione dati viene utilizzato json.
L'unica eccezione sono i comandi per cambiare modalità di controllo della SmartRoom che presentano il seguente formato:

chmod CONTROL_MODE

Room Dashboard:

Scritta in javascript, html e css, è una semplice pagina web che permette di monitorare lo stato della **SmartRoom** e di controllarla.

Per ricezione e trasmissione dati viene utilizzato json.



Dashboard screen

3. Tasks and FSMs:

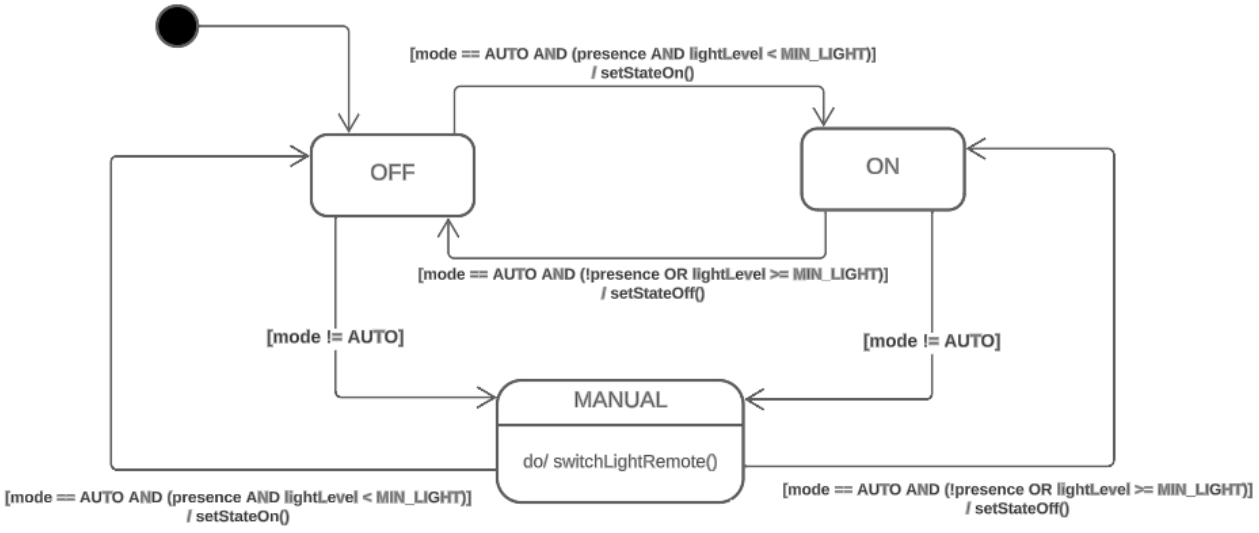
Room Controller:

Room Controller è composto da 4 Tasks:

- LightControlTask
- RollerBlindControlTask
- SerialComTask
- BluetoothComTask

LightControlTask:

Task che si occupa di controllare il light system della SmartRoom.

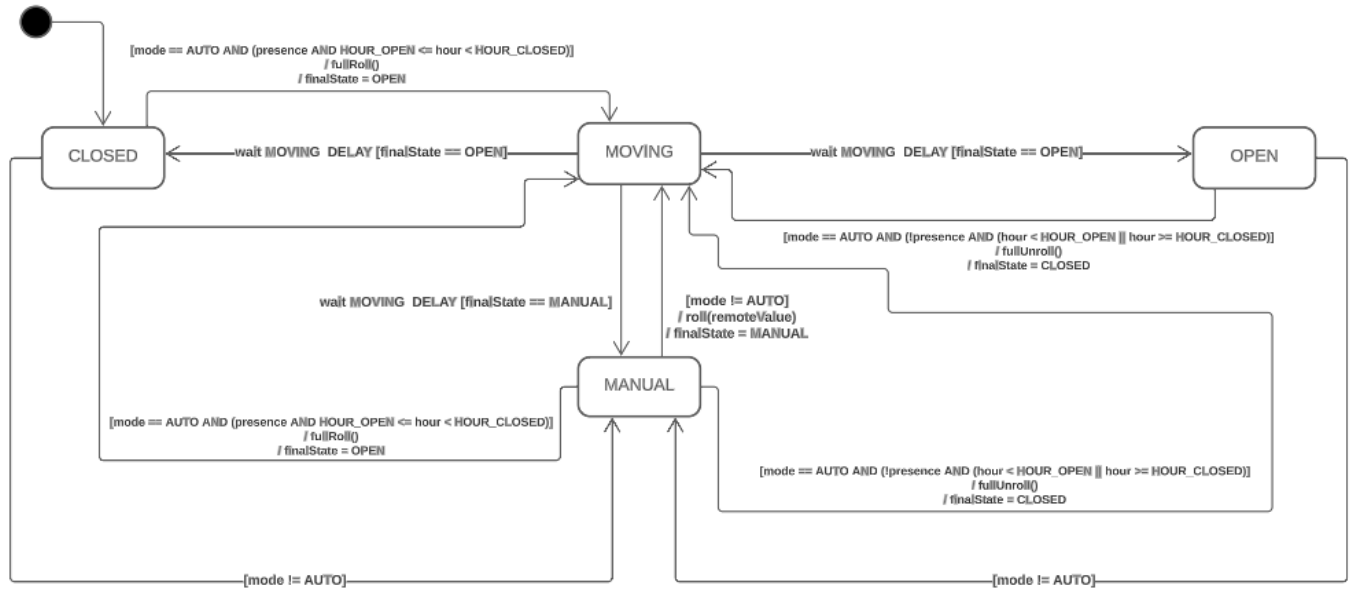


Presenta 3 stati:

- OFF: light system spento
- ON: light system acceso
- MANUAL: light system controllato da remoto

RollerBlindControlTask:

Task che si occupa di controllare le roller blinds della SmartRoom.



Presenta 4 stati:

- CLOSED: roller blinds completamente chiuse
- OPEN: roller blinds completamente aperte
- MANUAL: roller blinds controllate da remoto
- MOVING: stato utilizzato per permettere alle roller blinds di completare il movimento corrente prima di eseguirne un altro

SerialComTask:

Task che si occupa di gestire la comunicazione seriale della SmartRoom.
Esegue principalmente due azioni:

- Analizza i dati in ingresso
- Invia aggiornamenti sullo stato della SmartRoom ai componenti esterni

BluetoothComTask:

Task che si occupa di gestire la comunicazione bluetooth della SmartRoom.
Esegue principalmente due azioni:

- Analizza i dati in ingresso

- Invia aggiornamenti sullo stato della SmartRoom ai componenti esterni

SerialComTask e BluetoothComTask:

Usano json per inviare e ricevere dati.
L'unica eccezione sono i comandi per cambiare modalità di controllo della SmartRoom che presentano il seguente formato:

chmod CONTROL_MODE

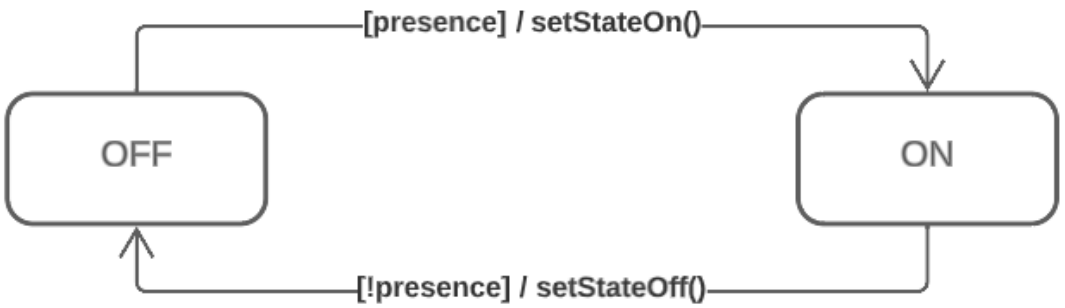
Room Sensor-Board:

Room Sensor-Board è composto da 3 Tasks:

- LightControlTask
- LightLevelSamplingTask
- PresenceSamplingTask

LightControlTask:

Task che si occupa di controllare il led della Room Sensor-Board.



Presenta 2 stati:

- OFF: led spento
- ON: led acceso

LightLevelSamplingTask:

Task che si occupa di campionare il livello di luce nella stanza e memorizzarlo nella Room Sensor-Board.

PresenceSamplingTask:

Task che si occupa di campionare la presenza di una persona nella stanza e memorizzarlo nella Room Sensor-Board.

4. Librerie utilizzate:

Room Controller:

- **ArduinoJson** per il parsing di dati
- **TimerOne** utilizzata nello Scheduler
- **ServoTimer2** utilizzata dal servomotore in alternativa a Servo

Room Sensor-Board:

- **WiFi**
- **PubSubClient** per utilizzare il protocollo MQTT

Room Mobile App:

- **Jetpack Compose** per l'interfaccia grafica
- **Material Design 3** come design system

Room Service:

- **node js** ed **express** per la creazione del server
- **mqtt** per utilizzare il protocollo MQTT
- **nodemon** per velocizzare lo sviluppo del software aggiornando il server ad ogni salvataggio
- **serialport** per utilizzare la comunicazione seriale

Room Dashboard:

- **Bootstrap** per l'interfaccia grafica