UNIVERSITÀ
POLITECNICA
DELLE MARCHE

FACOLTA' DI INGEGNERIA
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea Triennale in Ingegneria Informatica
e dell'Automazione

Implementazione di un sistema domotico basato su framework open-source

Implementation of a home automation system based on open-source framework

RELATORE:
Prof. Ennio Gambi

CORRELATORE:
Ing. Adelmo De Santis

TESI DI LAUREA DI:
Francesco Cananà

Anno Accademico 2023/2024

PREFAZIONE

Questa Tesi si pone nell'ambito della domotica, ma cosa significa dire che qualcosa è "domotico"? Beh "*domotico"* innanzitutto fa riferimento al contesto casalingo (da "*Domus*", che in latino appunto significa "*Casa"*) e si riferisce essenzialmente al fatto che delle operazioni che normalmente sarebbero gestite dall'uomo fisicamente - cioè andando ad azionare "a mano" il comando - possono essere gestite in remoto, quindi senza lo sforzo fisico da parte delle persone. Questo tipo di tecnologia si occupa per definizione della pianificazione, dello studio e dell'implementazione di tecnologie e di soluzioni volte a migliorare la qualità di vita nella casa. Essa si inserisce inoltre nel grande contesto dell'IoT ("Internet of Things") il quale è un macro-termine che racchiude tutte le tecnologie capaci di raccogliere dati via internet o dall'ambiente circostante e di elaborarli. In questi ultimi anni questo tipo di tecnologia ha preso sempre più campo nel mercato globale: secondo uno studio di "Data Bridge" il mercato della domotica ha raggiunto un valore di 44,72 miliardi di dollari nel 2021 e stima, inoltre, che salirà a 93,10 miliardi di dollari entro il 2029, il tutto grazie ai molteplici vantaggi che essa offre. Tra questi possiamo citarne alcuni, come:

- Aumentare la sicurezza dell'ambiente casalingo
- Migliorare la qualità della vita delle persone
- Minimizzare gli sprechi e i consumi energetici

Questi rappresentano solo una piccolissima parte degli svariati vantaggi che la tecnologia domotica può offrire ed è sicuramente per ciò che essa sta acquisendo - e sicuramente acquisirà - sempre più consenso nell'ambiente domestico del futuro.

INDICE

ELENCO DELLE FIGURE

1. INTRODUZIONE

Il mio elaborato si pone l'obiettivo di spiegare alcune applicazioni che vengono realizzate in ambiente domotico. In questa tesi vengono infatti approfondite alcune funzionalità del software "Home Assistant", il quale è in grado di comunicare con diversi sensori, programmati utilizzando il software "Arduino IDE" e che implementano il protocollo di comunicazione "MQTT". Il tutto è stato implementato attraverso due setup all'interno del Laboratorio di ICT del Dipartimento di Ingegneria dell'Informazione di UNIVPM. Il setup sperimentale realizzato è stato concepito per simulare una rete domotica, alla quale sono stati collegati diversi attuatori, che nel mio caso si identificavano con delle luci, ma che nella realtà potrebbero essere attuatori di differente natura, quali temperatura, pressione, oppure sensori collegati all'impianto di video-sorveglianza e così via. Questa versatilità è garantita dal protocollo MQTT, che rende interoperabile Home Assistant con sensori di differente natura e vendor.

Figura 1. Rappresentazione generale di un sistema domotico

2. DESCRIZIONE DELLE COMPONENTI SOFTWARE E HARDWARE

In questo capitolo verranno descritte le componenti Software, ossia i programmi utilizzati per l'esecuzione del progetto e quelle Hardware, ossia le parti "fisiche" che hanno fatto parte dei test effettuati in laboratorio.

2.1 HOME ASSISTANT

Home Assistant è un software gratuito e open-source, potente e versatile, il quale consente di far funzionare vari dispositivi domotici, come sensori di movimento, di luminosità, ambientali, di rumorosità e così via. Esso è installabile su qualsiasi PC con Sistema Operativo Windows, MacOS o Linux, ma anche su mini computer a basso assorbimento elettrico (come ad esempio Raspberry Pi). Per quanto riguarda la mia fase di setup di Home Assistant è stato necessario avere a disposizione 3 cose:

1- Un PC su cui installare il software
2- Una rete locale e un cavo Ethernet a cui collegare il PC
3- I dispositivi da connettere e che potranno essere poi controllati direttamente dall'interfaccia di Home Assistant

Figura 2. Logo di Home Assistant

Nel mio caso il software è stato installato nel PC - avente Windows come sistema operativo - e in particolare in una Virtual Machine ("Macchina Virtuale"), creata utilizzando VirtualBox. Una Virtual Machine è un software che offre le stesse funzionalità di un computer fisico e come quest'ultimo esegue delle applicazioni e un sistema operativo. All'interno di VirtualBox sono state create due Virtual Machines, una delle quali serviva proprio ad avviare Home Assistant. Durante i miei test nella macchina virtuale di Home Assistant è stato impostato Linux come

Sistema Operativo (con versione “Other Linux” a 64 bit). Le funzionalità di Home Assistant sono veramente moltissime e sarebbe praticamente impossibile elencarle tutte in poche pagine: mi limiterò a descrivere quelle che sono state le più importanti per i miei scopi e per avere una maggior comprensione durante la lettura lo farò nei capitoli successivi, quando verranno descritti i test effettuati in laboratorio. L’installazione dell’ambiente virtuale Home Assistant - nonché il suo successivo funzionamento - avviene solo ed esclusivamente mediante l’uso di un cavo Ethernet connesso al router: questo poiché appunto la Virtual Machine di Home Assistant necessita della connessione fisica per avviarsi e funzionare. Nelle figure sottostanti è possibile vedere come si presentano le Macchine Virtuali prima e dopo l’avviamento. La figura 3 mostra come si presenta la schermata di avviamento della Virtual Machine, mentre nelle due immagini successive (figure 4 e 5) vengono mostrate le schermate avviate, rispettivamente per il software Home Assistant e per il Broker MQTT:

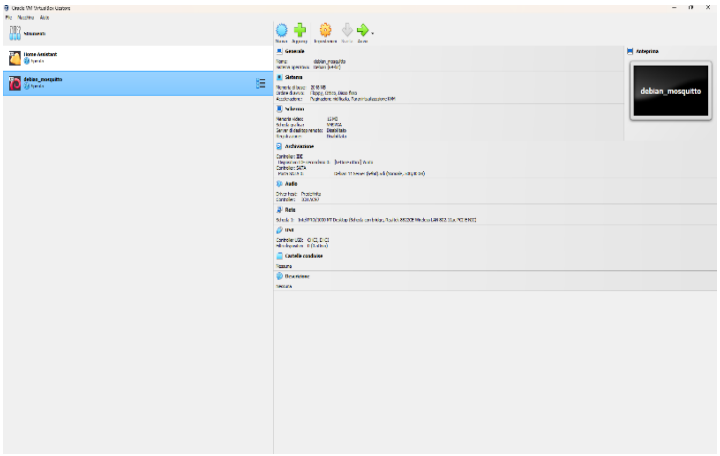

Figura 3. Schermata iniziale della VirtualBox prima dell’avviamento

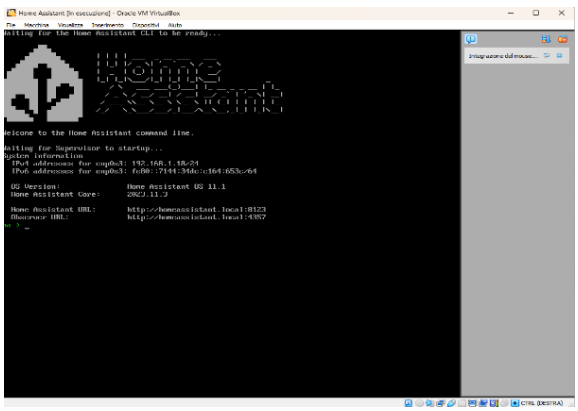

Figura 4. Virtual Machine avviata di Home Assistant

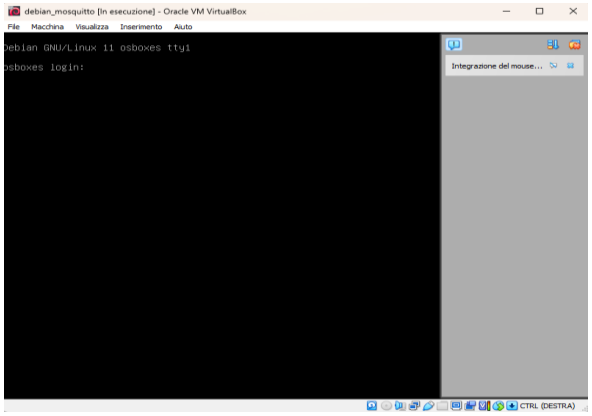

Figura 5. Virtual Machine avviata del broker MQTT

2.2 IL PROTOCOLLO DI COMUNICAZIONE MQTT

MQTT ("Message Queue Telemetry Transport") è un protocollo di comunicazione molto utilizzato nelle tecnologie IoT e in particolare è uno degli standard più apprezzati in ambito domotico. Nato ormai 25 anni fa - nel 1999 - esso si pone l'obiettivo di gestire le connessioni "machine to machine", ovvero serve a consentire la comunicazione tra le macchine in maniera estremamente efficiente. Per il mio lavoro ho scelto di utilizzare questo protocollo di comunicazione perché presenta alcune caratteristiche che a mio avviso sono molto vantaggiose, come:

1- Il fatto di essere un protocollo semplice e leggero
2- Richiedere poche risorse ai dispositivi che lo impiegano
3- Essere facilmente integrabile con Home Assistant e con la rete locale in cui sono stati eseguiti i test

Il protocollo MQTT segue un paradigma di pubblicazione e sottoscrizione classico definito "Publish and Subscribe". Questa logica fa sì che se due nodi (supponiamo ad esempio nodo A e nodo B) vogliano comunicare tra di loro, essi non lo faranno in maniera diretta come se fosse una telefonata tra uno e l'altro; la comunicazione avverrà infatti tramite un apposito Broker, il quale può essere visto come una sorta d'intermediario tra le due parti. Il Broker MQTT è essenzialmente un software - nel mio caso ho installato il "Mosquitto Broker MQTT" - che si occupa di ricevere dei messaggi dai mittenti e di renderli disponibili verso gli utilizzatori. La comunicazione mittente/utilizzatore avviene tramite dei canali sui quali viaggiano le informazioni chiamati topics e ai quali entrambi i nodi sono collegati. Quindi in breve è un po' come se noi avessimo: nodo A, nodo B e un topic, che io ho chiamato "led/stato".
Al fine di far comunicare i due nodi tramite MQTT, essi prima si dovranno connettere allo stesso topic e poi saranno in grado di comunicare tra loro scambiando dei messaggi a cui sono associati i relativi comandi. Riprendendo quanto detto sopra, per il mio lavoro è stato installato un broker MQTT all'interno di una Virtual Machine a cui ho associato un indirizzo IP locale collegato alla rete di laboratorio (che nel mio caso è stato 10.100.0.187), in modo che i dispositivi possano comunicarci senza passare attraverso il router. Il broker lavora quindi sulla Virtual Machine ed è stato installato con sistema operativo Linux (come per la Virtual Machine di Home Assistant) e utilizzando il package "Mosquitto". Ciò che avviene nella pratica è che quando verrà avviata la Macchina Virtuale si avvierà il servizio MQTT che è subito in grado di

ricevere e di smistare messaggi. La configurazione del broker è stata inoltre fatta in modo tale da consentire l'accesso a tutti i nodi, senza dover richiedere ogni volta l'autenticazione.

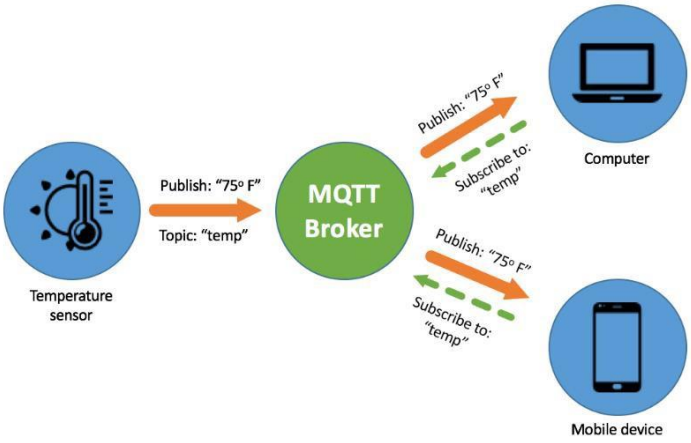

Figura 6. Schema generale di funzionamento del Broker MQTT

2.3 ARDUINO E SCHEDA UTILIZZATA

Per eseguire i test in laboratorio è stato necessario implementare del codice al fine di poter controllare l'accensione e lo spegnimento di un Led e questo è stato fatto tramite Arduino. Arduino è un piattaforma hardware dotata di una serie di componenti elettroniche, di un microcontrollore proprio e di un software personale che serve per la programmazione delle varie componenti. Per i miei test però non è sempre stato utilizzato il microcontrollore proprio di Arduino, ma anche un'altra scheda denominata "NodeMCU ESP8266". Questo perché quest'ultimo dispositivo, largamente utilizzato nell'ambito domotico, ha un modulo Wi-Fi integrato che gli permette di interagire direttamente con la rete locale a cui ero collegato per eseguire i test. Oltre a questo grande vantaggio la scheda NodeMCU offre anche un'altra grande possibilità: essa infatti può essere direttamente integrata con Home Assistant dopo aver installato l'apposito firmware all'interno del software. Per poter utilizzare questa scheda è stato opportuno installare inizialmente alcuni driver che erano descritti nella documentazione del microcontrollore, e poi successivamente impostare una baud rate di 9600.

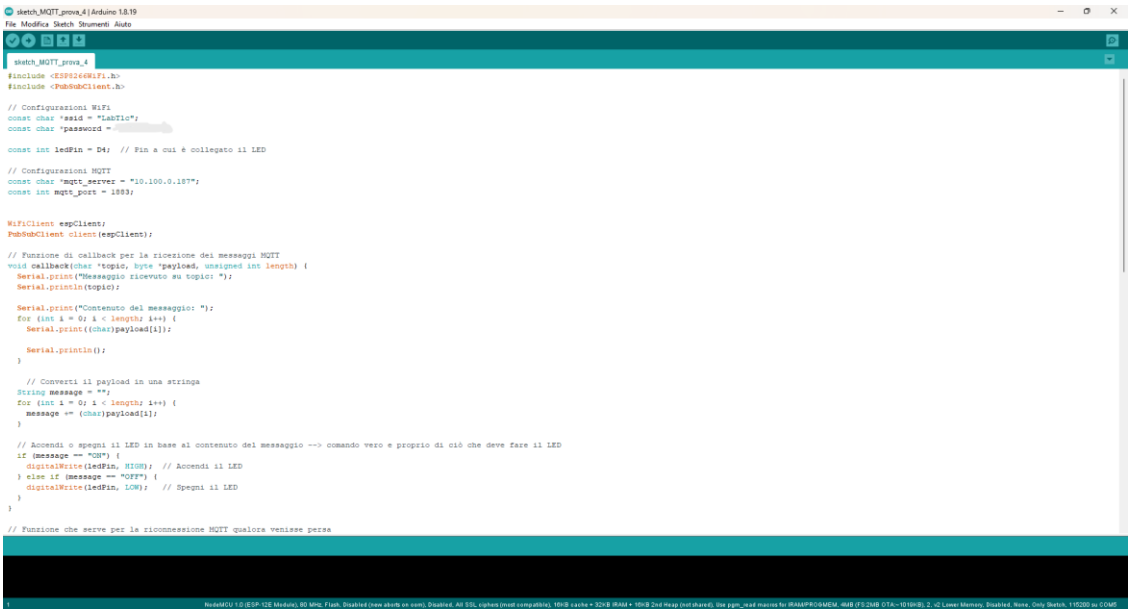

Figura 7. Esempio di una schermata di programmazione attraverso il software Arduino

3. INTEGRAZIONE HARDWARE E SOFTWARE

In questo capitolo verranno approfonditi i diversi aspetti di integrazione Hardware e Software attraverso la descrizione dei due test che ho effettuato in laboratorio: il primo riguardante la configurazione e il funzionamento dei dispositivi Tasmòta, il secondo invece relativo alla preparazione, configurazione e collegamento di due piattaforme (Arduino e Home Assistant) che attraverso il protocollo di comunicazione MQTT hanno permesso l'accensione e lo spegnimento di un Led in remoto.

3.1 1° TEST: LUCI TASMOTA

Il 1° test effettuato in laboratorio ha riguardato la programmazione di 5 dispositivi, in modo tale che essi potessero essere controllati in remoto direttamente dall'interfaccia di Home Assistant. L'obiettivo era sostanzialmente quello di riuscire a configurare manualmente i sensori e fare in modo che essi si potessero accendere e spegnere attraverso il software senza che ogni volta fosse necessario andare ad azionare l'interruttore a mano. I dispositivi in questione erano delle luci appositamente collegate alla rete locale e ognuna delle quali avente un proprio indirizzo IP all'interno della rete di laboratorio (gli indirizzi IP andavano da 10.100.0.209 a 10.100.0.213).

Figura 8. Luci Tasmòta

Il primo step è stato quello di configurare manualmente i 5 sensori andando a inserire come URL nel browser l'indirizzo IP di ognuno. Nella pagina di configurazione è stato poi necessario immettere l'IP del broker MQTT collegato alla rete di laboratorio e configurato su Home Assistant in modo tale che ci potesse essere un collegamento tra le due parti. Il broker MQTT è stato di fondamentale importanza poiché grazie a questo si riesce ad avere un collegamento tra le luci e il software. Successivamente è stato necessario andare su Home Assistant e avviare i comandi per far comunicare le due parti; per fare ciò però il software richiede che vengano create delle entità. Le entità, in Home Assistant, rappresentano già di per sé degli oggetti astratti dotati di alcune caratteristiche, quali:

1- La natura, che in questo caso sarà *interruttore*, o meglio ancora, *switch* (Switch è il termine che raggruppa questa categoria di interruttori in Home Assistant)
2- Lo stato, che sarà *"on"* o *"off"*
3- Gli attributi, che possono variare in base alla configurazione. In questo caso un attributo utilizzato è la visualizzazione della data e dell'ora dell'ultima volta in cui è stato eseguito il comando, ossia quando è stata accesa/spenta la luce.

Per capire meglio come queste entità appaiono graficamente nella pratica ho riportato qui in figura le 5 entità create relative ad ogni luce (la prima, ossia "Luci", è quella che serve per far accendere/spegnere tutti e 5 i dispositivi contemporaneamente).

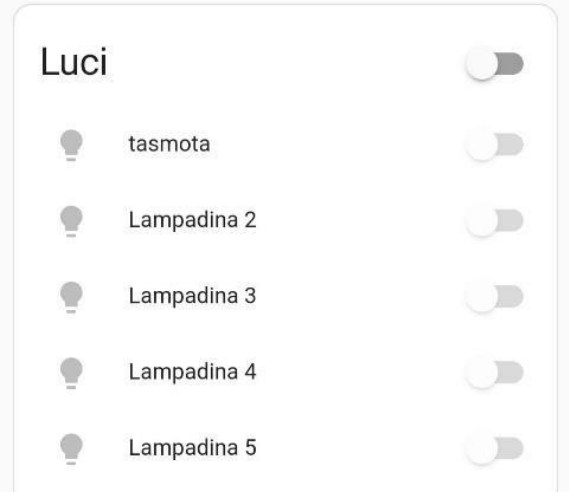

Figura 9. Entità in Home Assistant per luci Tasmòta

Una volta create le varie entità, queste sono state aggiunte alla schermata principale di controllo (chiamata *"plancia"*) e in questo modo sono stato in grado di controllare il tutto tramite il software andando semplicemente a cliccare sui vari pulsanti relativi alle entità.

3.2 2° TEST: ACCENSIONE E SPEGNIMENTO DEL LED DA REMOTO

Il 2° test ha riguardato invece l'accensione e lo spegnimento di un Led attraverso un comando mandato in remoto. Il primo passo è stato realizzare a livello hardware un Led che potesse essere controllato attraverso Arduino. Per i miei scopi ho utilizzato uno Starter kit di Arduino dai quali ho utilizzato le seguenti componenti: la breadboard (ossia la piattaforma hardware sulle quali mettere i vari componenti elettronici), un Led, una resistenza da 220 Ω (ohm) e vari connettori. Per testare che il codice del Led funzionasse ho inizialmente utilizzato la scheda propria di Arduino, per poi sostituirla successivamente con la NodeMCU. Di seguito in figura è riportata una delle prime prove fatte a livello hardware per il funzionamento del Led con la scheda di Arduino.

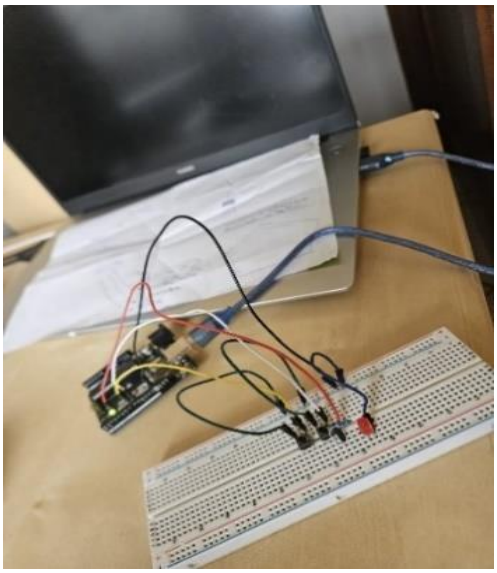

Figura 10. Collegamento hardware del Led con Arduino

3.2.1 Prima implementazione software con la scheda di Arduino

Una volta testato il funzionamento hardware - quindi dopo aver controllato che la scheda venisse effettivamente letta da Arduino IDE e che non fossero presenti errori nella realizzazione hardware - mi sono dedicato alla scrittura del codice. Qui sotto è riportato un esempio di codice su come è stato fatto funzionare il Led:

```
int led = 13;
//definisco led pin / define led pin
void setup() {
    pinMode (led, OUTPUT);
// inizializza il pin come un output /initialize the digital pin as an output
    }
void loop() {
    digitalWrite (led, HIGH);
// accende il LED (HIGH è il livello di voltaggio) / turn the LED on (HIGH is the voltage level)
    delay(1000);
// attende un secondo / wait for a second
    digitalWrite(led, LOW);
// spegne il LED rendendo il voltaggio LOW  / turn the LED off by making the voltage LOW
    delay(1000);
// attende un secondo / wait for a second
}
```

Come si può notare da questo esempio, il codice in Arduino è composto da due parti principali. La prima è quella contenente la funzione *"setup"*: al suo interno vengono inizializzate le variabili, impostati i vari PIN da utilizzare e in che modo. La seconda invece è la funzione *"loop"*: essa potrebbe essere definita come il vero e proprio corpo del codice poiché al suo interno viene specificato cosa effettivamente il dispositivo deve andare a fare. Nel mio caso il Led deve accendersi, aspettare 1000 ms (ossia 1 s), spegnersi, aspettare nuovamente 1000 ms e poi infine riaccendersi. E così in loop, appunto. Dopo aver verificato che il Led venisse riconosciuto

dall'IDE e che i comandi rispettassero effettivamente le istruzioni date, sono andato avanti con il test.

3.2.2 Implementazione software su NodeMCU ESP8266

Lo step successivo è stato quello di integrare il codice con la scheda "ESP8266". Questa scheda - come riportato nel capitolo 2.3 - è dotata di modulo Wi-Fi integrato e perciò è stato necessario prima di tutto collegarla alla rete locale di laboratorio. Per fare ciò ho installato i vari driver necessari, in modo tale che potesse essere letta dall'IDE di Arduino. Successivamente la scheda è stata configurata su Home Assistant e per fare questo è stato necessario accedere alle impostazioni del software e installare tra le componenti aggiuntive proprio quella relativa a ESPHome, la quale è in grado di leggere le varie schede ESP. Nelle due figure che seguono sono riportate a sinistra la scheda utilizzata ("NodeMCU ESP8266"), a destra invece il collegamento Hardware con il PC e con Arduino per testare il funzionamento del Led.

Figura 11. NodeMCU ESP8266

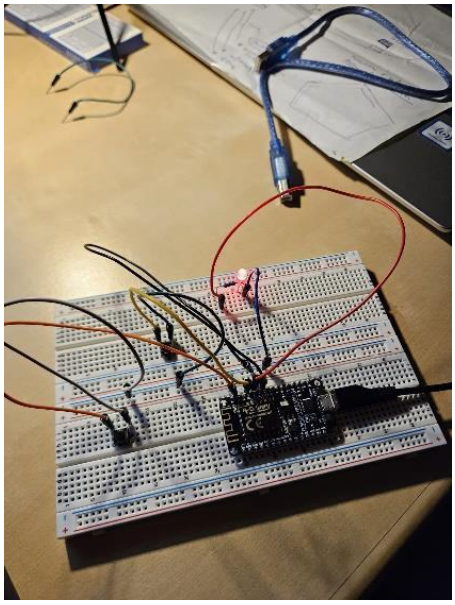

Figura 12. Collegamento hardware con la scheda

Installata la mia scheda all'interno del software e verificato che questa venisse effettivamente riconosciuta all'interno dell'ambiente Home Assistant ho proceduto con la scrittura del codice in Arduino IDE, ma questa volta utilizzando la NodeMCU. Proprio per questo la prima cosa che ho fatto è stata sfruttare questa possibilità andando a collegare la scheda alla rete locale di laboratorio in modo tale che a quest'ultima venisse assegnato un indirizzo IP all'interno

della rete stessa. Ho perciò integrato all'interno del codice di accensione e spegnimento del Led la funzione di connessione al Wi-fi. Qui sotto è riportato il frammento di codice relativo alla connessione Wi-Fi alla rete locale di laboratorio:

```
#include <ESP8266WiFi.h>

// Configurazione Wi-Fi / Wi-Fi configuration
const char *ssid = "LabTlc";
const char *password = "*****"
```

In questa parte di codice piuttosto basilare vengono solamente dichiarate e poi inserite sotto forma di stringa le credenziali della rete (SSID e Password). La cosa che però è importante sottolineare è ciò che è riportato nella prima riga. Qui infatti è stata importata la libreria *"ESP8266WiFi",* la quale è assolutamente necessaria per fare in modo che Arduino IDE possa riconoscere la scheda durante la connessione. Dopo aver eseguito il codice, il risultato ottenuto è quello riportato qui in figura:

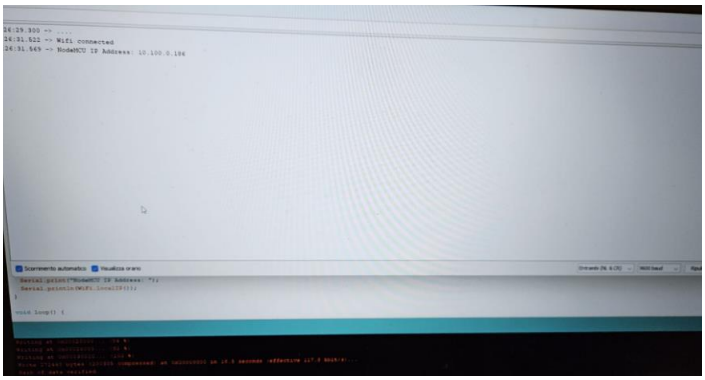

Figura 13. Collegamento al Wi-Fi della scheda ESP8266

Come è possibile vedere la connessione è stabilita ed è stata assegnato un indirizzo IP alla scheda (10.100.0.186) all'interno della rete locale di laboratorio. Dopo aver connesso la scheda alla rete è stato necessario implementare anche altre funzioni che mantengano viva la connessione e che facciano riconnettere i dispositivi automaticamente qualora la connessione dovesse per qualsiasi motivo perdersi.

```
// Connessione Wi-Fi / Wi-Fi connection
WiFi.begin(ssid, password);
      while(WiFi.status() != WL_CONNECTED){
            delay(1000);
Serial.println("Connessione in Corso…");
}
Serial.println("Connessione WiFi stabilita");
```

Questa parte di codice serve essenzialmente a farci capire che ogni volta che verranno inserite le credenziali di rete (SSID e Password) si avvierà il tentativo di connessione e apparirà la scritta "Connessione WiFi in corso..." fino a quando questa non sarà stabilita.

3.2.3 Collegamento al broker MQTT

Una volta connessa la scheda alla rete locale ho configurato la connessione al broker MQTT. Il broker MQTT - come specificato nel capitolo 2.2 - ha essenzialmente il compito di collegare il software Home Assistant con il codice del Led creato su Arduino e di far comunicare i due tramite un canale di scambio. Nello specifico del protocollo MQTT questi canali di scambio tra le parti sono chiamati *topics.* Definendo quindi uno stesso topic nelle due parti che vogliono comunicare tra loro si riesce a farli connettere e pertanto i comandi inviati da Home Assistant saranno smistati attraverso il broker MQTT e ricevuti da Arduino il quale, tramite il codice creato, eseguirà il comando impostato per il Led. Qui sotto è riportato graficamente in estrema sintesi il funzionamento di trasmissione delle informazioni tramite topic e protocollo MQTT.

Figura 14. Schema riassuntivo della trasmissione delle informazioni tramite MQTT

Una volta definita l'importanza del topic per questo tipo di comunicazione e avendo un'idea di come avviene lo scambio d'informazioni sono andato a implementare la connessione al broker da codice. Questo tipo di connessione è molto simile a quella vista in precedenza per la connessione al Wi-Fi e qui di seguito ho riportato il frammento di codice relativo:

```
#include <PubSubClient.h>

// Configurazione MQTT / MQTT Configuration
const char *mqtt_server = "10.100.0.187"
const int mqtt_port = 1883;
```

In questa porzione di codice, sono definiti l'indirizzo IP e la porta sulla quale è in ascolto il broker MQTT. Il Server corrisponde all'indirizzo IP di laboratorio a cui è associato il broker, mentre la Porta è usata per comunicare col Server MQTT. Nel protocollo MQTT le porte utilizzabili sono 2: "8883", usata per collegamenti con TLS (Transport Layer Security), la quale assicura più privacy e integrità dei dati per le comunicazioni Internet (ma viene per lo più utilizzata per lo sviluppo delle web App), o "1883" che invece è quella che ho utilizzato io, maggiormente utilizzata per collegamenti in chiaro e che sono sostanzialmente più semplici da effettuare. Infine anche qui, come per la connessione al Wi-Fi, risulta fondamentale la prima riga in cui è stata importata la libreria *"PubSubClient"*, la quale è stata creata proprio per permettere la messaggistica tramite protocollo MQTT. Infine è importante sottolineare che oltre alle righe di codice che implementano la connessione iniziale al broker, anche qui è stata implementata una parte relativa al mantenimento della connessione. Qui di seguito è riportato il frammento di codice utile per la riconnessione al broker MQTT:

```
// Mantiene la connessione MQTT attiva /  Keep the MQTT connection on
 if (!client.connected()) {
     reconnect();
 }
```

3.2.4 Sviluppo del codice in Arduino per la trasmissione delle informazioni tramite MQTT

Per fare in modo che Home Assistant e Arduino riescano a comunicare tra loro però non è sufficiente configurare la connessione alla rete e al broker e perciò arrivato a questo punto sono andato a implementare varie funzioni che mi hanno permesso di rendere possibile la trasmissione delle informazioni. Qui di seguito verrà spiegato gran parte del codice che ho creato per la trasmissione delle informazioni tramite MQTT e verranno illustrate le principali funzioni che sono state basilari a livello implementativo:

// Funzione di callback per la ricezione dei messaggi MQTT / Callback function for receiving // MQTT messages

```
void callback(char *topic, byte *payload, unsigned int length)
{
      Serial.print("Messaggio ricevuto su topic: ");
      Serial.println(topic);
      Serial.print("Contenuto del messaggio: ");
        for(int i = 0; i < length; i++)
    {
        Serial.print((char)payload[i]);
    }
      Serial.println();
}
```

Queste righe rappresentano una delle parti implementative più importanti. All'interno di questo frammento di codice è infatti descritta la funzione di callback, la quale è incaricata di occuparsi della ricezione dei messaggi MQTT. Questa funzione prende 3 argomenti: il topic, il payload (ossia il contenuto del messaggio) e la lunghezza del payload stesso. Poiché il messaggio che noi vogliamo trasmettere non è nel formato che desideriamo - infatti il payload è di tipo byte, rappresentato quindi come sequenza di bit - è stato necessario convertirlo in una stringa, ossia in una sequenza di lettere e/o numeri che possano essere leggibili dall'utente. Questo è stato possibile grazie alla conversione in un altro tipo di dato chiamato *"char"* (tipo utilizzato di default per la rappresentazione di stringhe contenenti caratteri

alfabetici, di punteggiatura ecc.). L'implementazione della conversione è descritta nelle righe di codice sottostanti:

```
// Conversione del payload in una stringa / Payload conversion to a string
String message = "";
  for(int i = 0; i < length; i++)
      {
         message += (char)payload[i];
      }
```

Successivamente è stata implementata la parte di codice relativa al comando vero e proprio che il led deve eseguire, ossia l'accensione e lo spegnimento:

```
 // Accendi o spegni il LED in base al contenuto del messaggio / Turn on/off the led according
// to the message topic
 if (message == "ON") {
       digitalWrite(ledPin, HIGH); // Accendi il LED / Turn the Led ON
      } else if (message == "OFF") {
       digitalWrite(ledPin, LOW); // Spegni il LED / Turn the Led OFF
      }
}
```

In questa parte troviamo una condizione che in informatica è molto utilizzata, chiamata istruzione *"if"*. Questa fa sì che se l'argomento passato all'interno delle parentesi tonde (che nel mio caso è (message == "ON")) corrisponda effettivamente a ciò che è stato inviato, allora verrà eseguito ciò che è scritto nella riga successiva e quindi verrà azionato il comando per far accendere il led. In caso contrario - quindi se l'argomento passato dovesse essere (message == "OFF") - il led si spegnerà. In estrema sintesi questo è un po' quello che quest'istruzione va a fare a livello implementativo ed è importante averlo descritto perché in ambito domotico gran

parte degli interruttori sono controllati proprio tramite la ricezione di messaggi come questo. Molti dispositivi elettronici infatti rispondono ad un comando che gli viene inviato eseguendo un'azione che è stata precedentemente programmata da qualcuno. Ma in questo caso specifico quindi cosa accade? Qui avviene che quando si va ad accendere l'interruttore dal software, quest'ultimo genererà automaticamente un messaggio - che nel mio caso è proprio la stringa "ON" - e questo farà in modo che il led si accenda. Quando poi invece il pulsante verrà premuto nuovamente il messaggio generato sarà "OFF" e quindi il led riceverà il comando di spegnersi; il tutto sempre grazie allo scambio d'informazioni che avviene attraverso il protocollo MQTT. Per rendere più comprensibile il funzionamento generale ho riportato qui sotto un piccolo schema riassuntivo per quanto riguarda l'accensione del led (un ragionamento analogo può essere fatto con lo spegnimento; cambierà solo il messaggio, il quale sarà "OFF" invece che "ON"):

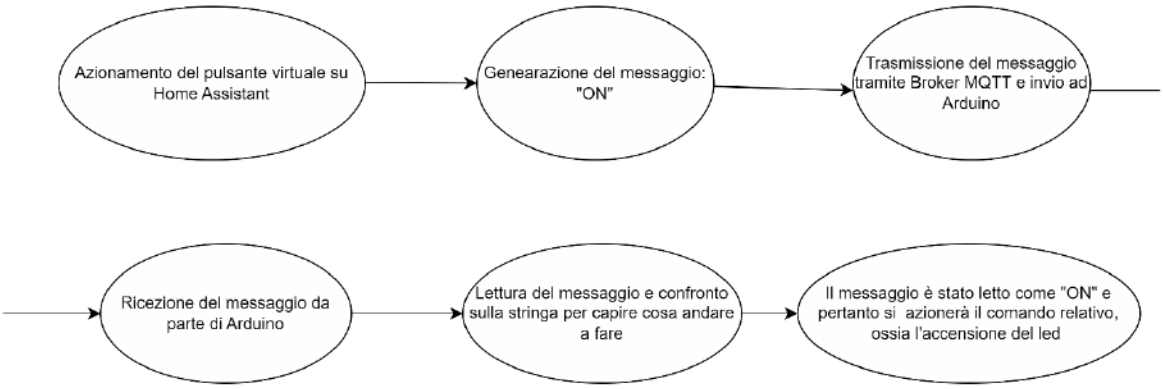

Figura 15. Schema del funzionamento di trasmissione delle informazioni da Home Assistant ad Arduino

3.2.5 Configurazione dei comandi nell'ambiente virtuale Home Assistant

Una volta spiegato il funzionamento a livello implementativo su Arduino è il momento di andare a vedere come funziona la configurazione sull'ambiente virtuale Home Assistant. Come prima cosa sono andato a creare due entità (la cui descrizione generale è riportata nel paragrafo 3.1): una per l'accensione del led e un'altra per lo spegnimento. Qui sotto in figura è riportato come appaiono le entità nel software:

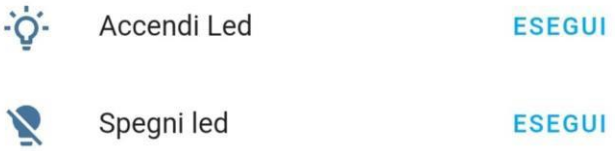

Figura 16. Pulsanti virtuali di accensione e spegnimento del Led in Home Assistant

Per fare in modo che queste entità potessero mettersi in comunicazione con Arduino attraverso il protocollo MQTT è stato necessario andare a configurare all'interno delle impostazioni del software alcuni parametri relativi alle entità. In particolar modo, accedendo alle impostazioni di Home Assistant si apre una schermata dalla quale è possibile controllare tutti i vari dispositivi che sono collegati alla rete domotica, tra cui appunto le entità. Andando a selezionare quella che ci interessa e scegliendo la voce "modifica entità" è poi possibile accedere a tutte le varie impostazioni dettagliate della stessa, avendo così la possibilità di modificare, aggiungere o rimuovere delle azioni. Nel mio caso è stato importante andare a inserire manualmente il topic giusto a cui collegare il led per il trasferimento delle informazioni – che nel mio caso ricordo chiamarsi "led/stato" – e scrivere il payload corrispondente, che per l'accensione del led era "ON". Le figure 17 e 18 mostrano le impostazioni dell'entità "Accendi led" all'interno del software per capire come effettivamente debbano avvenire i vari settaggi iniziali per consentire lo scambio d'informazioni attraverso MQTT tra Home Assistant e Arduino.

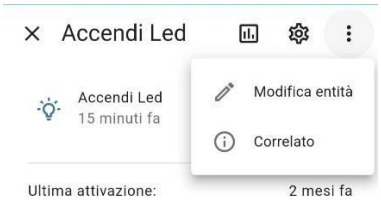

Figura 17. Schermata iniziale di un'entità

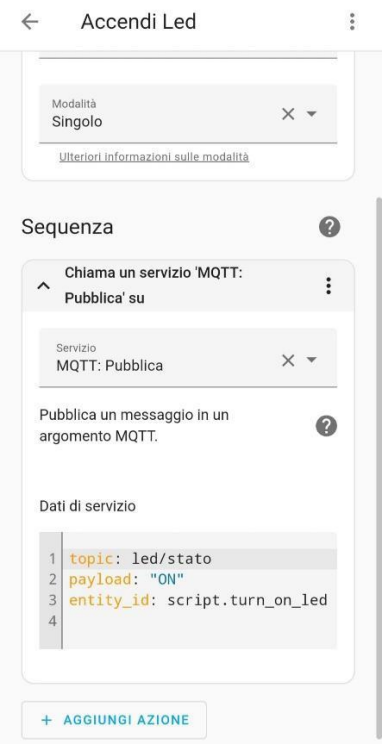

Figura 18. Schermata di configurazione di un'entità

Come si può vedere dalla figura 18, le righe 1 e 2 sotto la voce "dati di servizio" sono quelle più importanti a livello di configurazione ed è proprio grazie a queste che la connessione tra le parti può avvenire in maniera efficiente. Finita la configurazione bisogna riavviare il software per permettere il salvataggio delle varie modifiche. Effettuati tutti questi passaggi l'ambiente Home Assistant è pronto per l'interazione con Arduino e quindi per il funzionamento del Led.

4. INTERFACCIA PER AMBIENTE MOBILE

Dopo aver analizzato le varie configurazioni per Arduino e Home Assistant è doveroso soffermarsi un attimo su un altro aspetto a mio avviso di fondamentale importanza, ossia l'usabilità dell'ambiente domotico da parte dell'utente finale. Il software Home Assistant offre infatti la possibilità di controllare tutti i dispositivi attraverso una propria App (disponibile sia per Android che per iOS) e questo rende sicuramente più facile e veloce la gestione e il controllo di tutti gli apparati presenti in casa. Qui in figura sono riportate due foto relative alle schermate principali dell'applicazione mobile (Android):

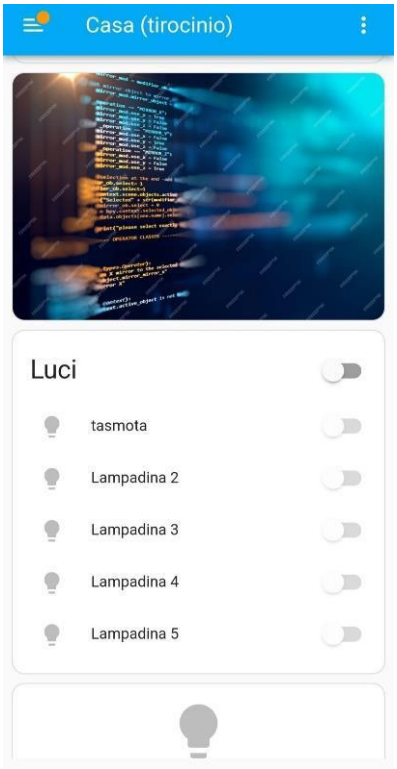

Figura 19. Dashboard dell'App

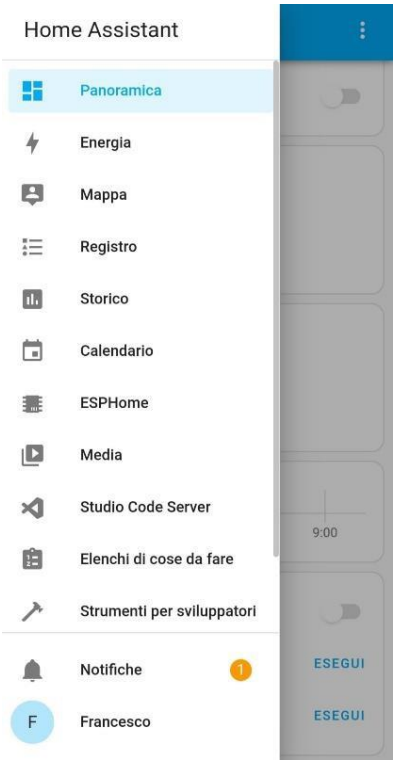

Figura 20. Pannello di controllo dell'App

Ovviamente queste 2 schermate danno un'idea abbastanza approssimativa di ciò che l'intera Applicazione è in grado di offrire, poiché all'interno della rete domotica che ho creato sono stati inseriti solo pochi sensori. Nella realtà, qualora si volessero integrare più dispositivi, l'Applicazione sarebbe in grado di controllare non solo le luci come nel mio caso, bensì di gestire anche sistemi di video-sorveglianza, monitorare e controllare sensori di temperatura per il riscaldamento/raffreddamento della casa ecc. Ma non solo: supponendo ad esempio di

disporre di un impianto fotovoltaico che aiuta nella gestione energetica casalinga, l'Applicazione potrebbe anche fornire in tempo reale una panoramica sulle condizioni energetiche per capire quanto si stia effettivamente consumando, o ancora meglio quanto sta consumando magari ogni singolo apparato (che può essere ad esempio la lavatrice, la lavastoviglie ecc.) e avere perciò un quadro generale sull'andamento della casa in cui si vive, che non è sicuramente cosa da poco.

5. CONCLUSIONI

Questo mio elaborato ha avuto l'obiettivo di approfondire alcuni aspetti del vastissimo mondo della domotica, mostrando la struttura e il funzionamento presenti dietro all'azionamento di determinati comandi per dei sensori. Il lavoro è proseguito integrando in piattaforma sia delle luci asservite ad attuatori SonOff (con firmware Tasmota) sia un sensore completamente custom, realizzato in piattaforma NodeMCU. I test ottenuti durante il periodo di ricerca e implementazione hanno dato i risultati sperati inizialmente e quindi posso affermare che questa modalità di sviluppo in ambiente domotico, valutata insieme alla difficoltà di realizzazione e implementazione, è molto soddisfacente ed efficace per questo tipo di tecnologia che sicuramente avrà moltissimo da riservarci nell'imminente futuro.

L'intero progetto è stato sviluppato utilizzando "macchine virtuali", un approccio sempre più diffuso nell'ambito ICT per contenere il consumo di energia elettrica e garantire migliori performances in termini di robustezza del sistema.

Figura 21. Potenzialità del controllo dei dispositivi da App mobile

6. SITOGRAFIA:

- **Virtual machine**

https://www.vmware.com/topics/glossary/content/virtual-machine.html

- **Home Assistant**

https://indomus.it/formazione/home-assistant-guida-rapida-ai-concetti-principali/

https://www.smartworld.it/guide/come-funziona-home-assistant.html#cose

https://www.home-assistant.io/installation/

- **MQTT**

https://www.internet4things.it/iot-library/mqtt-cose-e-come-funziona-il-protocollo-alla-base-delliot/

https://indomus.it/formazione/mosquitto-mqtt-broker-comandi-utili/

- **INCREMENTO DOMOTICA**

https://www.youtradeweb.com/domotica-il-mercato-e-in-crescita-del-123/#:~:text=Secondo%20Data%20Bridge%2C%20il%20mercato,tra%20il%202022%20al%202029