

Corso IOT, anno accademico 2022/2023

SMART GUARDIAN

Gruppo 04:

Andrea Alberti, Alessio Bosco, Alberto Cirola



PUNTI CHIAVE DELLA PRESENTAZIONE

- Descrizione del progetto
- Dispositivi utilizzati
- Protocolli di comunicazione
- Interfaccia (Dashboard e App)



Descrizione del progetto

Il progetto consiste nella realizzazione di una cassaforte smart, che garantisca all'utente di lasciare i propri beni in un ambiente sicuro, avendo la possibilità di gestirla tramite un'interfaccia web. **SmartGuardian** mette a disposizione dell'utente una dashboard (realizzata con node-red) che permette di inserire il codice di sblocco, di prelevare l'oggetto e di bloccare nuovamente la cassaforte. Inoltre è anche possibile cambiare il codice in caso di necessità e di visualizzare una tabella di record di allarmi in modo da tenere traccia di eventuali tentativi di intrusione. Il sistema di allarme viene attivato quando si verificano 3 errori consecutivi sul codice di sblocco, in seguito ad intrusioni o quando si rileva uno spostamento dell'oggetto con cassaforte bloccata. Oltre alla dashboard, prettamente utilizzata per garantire una comoda gestione lato utente, si è predisposta anche un'applicazione (basata su MQTTAlert) che invia notifiche all'utente sul proprio dispositivo mobile per informarlo di eventi salienti.

Dispositivi Utilizzati

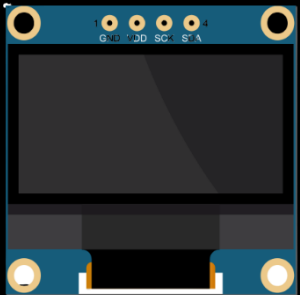
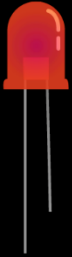


- **ESP-32:** Abbiamo utilizzato lo Zerynth Eva ZM1 che è un modulo IoT basato sul microcontrollore ESP32. ESP32 è un microcontrollore a basso consumo energetico con supporto integrato per Wi-Fi e Bluetooth. Supporta diversi protocolli di comunicazione tra cui I2C, che è stato utilizzato anche in altre fasi del progetto.



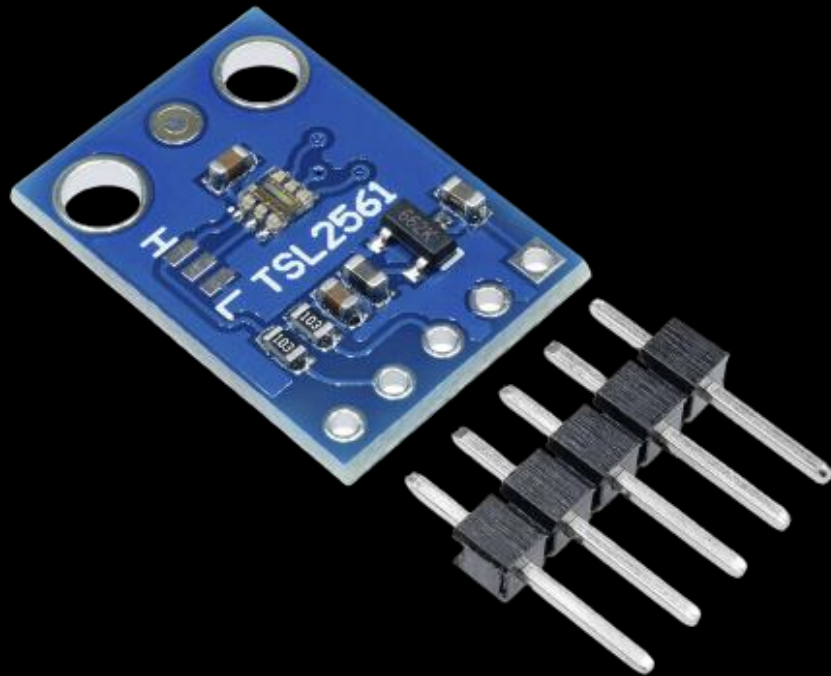
- **SERVOMOTORE:** Il motore servo SG90 è un piccolo motore ad azionamento elettrico, controllato attraverso un segnale di controllo PWM (Pulse Width Modulation) che segnale PWM determina l'angolo a cui il motore servo deve essere posizionato. Il motore dispone di tre fili di connessione: alimentazione (è stata utilizzata una tensione di 5V), messa a terra e segnale di controllo collegato direttamente all'Esp32. Il motore è stato utilizzato nel nostro progetto per l'apertura e la chiusura della porta della cassaforte.

Dispositivi Utilizzati



- **LED:** dispositivo elettronico a semiconduttore che converte l'energia elettrica in luce, costituito da due strati semiconduttori: il catodo e l'anodo. In questo progetto si è fatto uso di due led, uno verde e uno rosso per favorire una corretta comunicazione verso l'esterno per l'utente ed un led bianco per garantire un funzionamento coerente del sensore di luminosità ambientale anche in caso di assenza di luce quando la cassaforte è chiusa.
- **BUZZER PASSIVO:** componente elettronico che produce suoni o segnali acustici quando viene applicata una tensione o un segnale elettrico. È possibile modulare l'intensità del suono, cioè il volume percettibile, utilizzando il duty cycle (ciclo di lavoro) di un segnale a onda rettangolare. Il duty cycle rappresenta la percentuale di tempo in cui il segnale è "attivo" rispetto al suo periodo totale. Utilizzato per il sistema di allarme, produce un suono che segnala l'attivazione all'utente.
- **DISPLAY ssd1306:** Tipo di schermo a matrice di punti che utilizza la tecnologia OLED per visualizzare testo, grafica e immagini in modo chiaro e luminoso. Supporta il protocollo I2C (Inter-Integrated Circuit), protocollo di comunicazione seriale che consente la connessione tra dispositivi elettronici tramite due linee di segnale: una linea di dati (SDA) e una linea di clock (SCL). Utilizzato per mostrare brevi messaggi all'utente che indicano il verificarsi di alcune situazioni chiave come lo sblocco/chiusura della cassaforte ed eventuali allarmi, oltre alla modifica del codice.

Dispositivi Utilizzati



Il sensore di luminosità ambientale **TSL2561** è un dispositivo di rilevamento della luce ambientale a elevata precisione.

Il sensore comunica tramite il protocollo I2C. Pertanto, sono stati collegati i pin SDA (Serial Data Line) e SCL (Serial Clock Line) all'ESP32 assegnandogli dei pin specifici.

Questo sensore è stato utilizzato nel nostro progetto per verificare la presenza o meno dell'oggetto all'interno della nostra cassaforte.

Datasheet:

<https://cdn-shop.adafruit.com/datasheets/TSL2561.pdf>

Implementazione del Dispositivo

```
def verifica_presenza_objetto():  
    global sensore_lux  
    if(sensore_lux.read() < 0.7):  
        return True  
    else:  
        return False
```

Abbiamo implementato questa funzione per verificare la presenza dell'oggetto all'interno della cassaforte utilizzando il sensore di luminosità ambientale. In particolare è stato necessario scaricare una libreria online (fonte <https://github.com/adafruit/micropython-adafruit-tsl2561>) e importato la classe «TSL2561» dove è presente il metodo read().

Si è osservato che per valori minori di 0.7 il sensore rilevava la presenza dell'oggetto.

```
i2c=SoftI2C(sda=Pin(16),scl=Pin(17))  
sensore_lux=tsl2561.TSL2561(i2c) #sensore di luminosità ambientale
```

Dispositivi Utilizzati



Il sensore **TRCT5000** è un sensore di rilevamento di oggetti a riflessione infrarossa. Attraverso la riflessione della luce infrarossa emessa riesce a rilevare la presenza o l'assenza di un oggetto. Questo sensore verrà poi collegato al pin 34 dov' è implementato l'ADC per convertire il segnale analogico in input con un segnale digitale. Nel nostro progetto è stato utilizzato per rilevare intrusioni. Un allarme verrà attivato se la cassaforte è chiusa e il sensore rivela un movimento.

Datasheet:

<https://datasheetspdf.com/pdf/377371/VishayTelefunken/TCR/T5000/1>

Implementazione del Dispositivo

```
#Definizione classe per il sensore a infrarossi
class LDR:
    def __init__(self, pin, min_value=0, max_value=10):

        if min_value >= max_value:
            raise Exception('Min value is greater or equal to max value')

        # initialize ADC (analog to digital conversion)
        # create an object ADC
        self.adc = ADC(Pin(pin))
        self.min_value = min_value
        self.max_value = max_value

    def read(self):
        return self.adc.read()

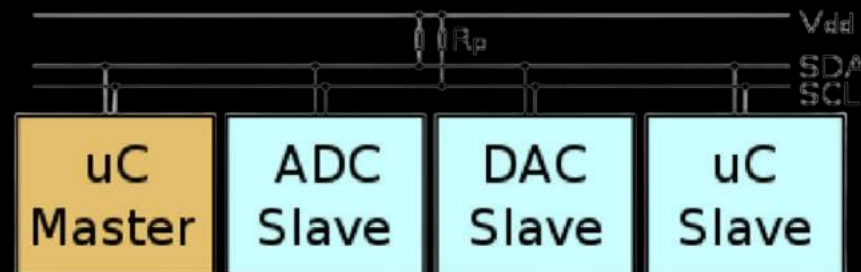
    def value(self):
        return (self.max_value - self.min_value) * self.read() / 4095
```

Abbiamo creato un apposita classe chiamata **LDR** per la lettura e la conversione del segnale in input. Nel metodo *value()* del codice, viene eseguita la divisione per 4095, valore che deriva dal fatto che si sta utilizzando un convertitore analogico-digitale (ADC) con una risoluzione di 12 bit.

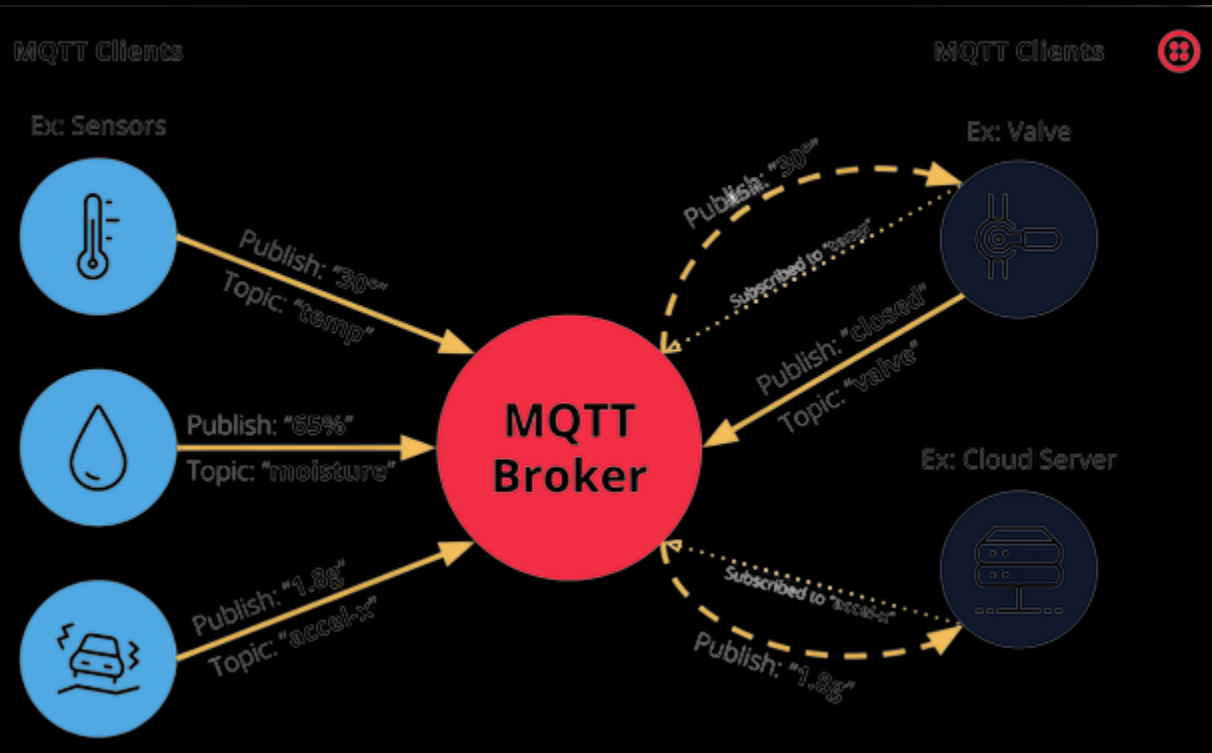
Protocolli di Comunicazione

I protocolli di comunicazione coinvolti nella realizzazione del progetto sono: **I2C** e **MQTT**.

- **I2C**: Utilizzato per il display e il sensore di luce, è un protocollo *sincrono*, *half duplex*, *master/slave* e con *linee condivise*. Le due linee sono chiamate: **SDA**=Serial Data, **SCL**=Serial Clock, la prima viene impiegata per la trasmissione di dati, l'altra è necessaria per favorire la sincronizzazione della comunicazione. Il funzionamento master/slave prevede che il trasmettitore invii dati sulla linea SDA e lo slave invii segnali di *ACK*, per operazioni in lettura il trasmettitore è lo slave, mentre per la scrittura è il master.



Protocolli di Comunicazione



MQTT: Protocollo di messaggistica asincrono basato sul modello **publish/subscribe** dove i dispositivi possono pubblicare un messaggio su un **topic** specifico e sottoscrivere a topic di interesse per ricevere i messaggi corrispondenti. Questo pattern richiede un **broker** di messaggistica, responsabile della distribuzione dei messaggi ai client destinatari. MQTT è progettato per essere efficiente in termini di risorse di rete e di dispositivi. MQTT offre diversi livelli di **QoS (Quality of Service)** per garantire la consegna affidabile dei messaggi. Contestualmente al progetto, il protocollo è stato impiegato per favorire lo scambio di messaggi con la dashboard sviluppata tramite node-red e con l'applicazione mobile MQTTAlert.

Interfaccia

Per favorire l'interazione con l'utente abbiamo realizzato due interfacce utilizzando **node-red** e l'applicazione mobile **MQTTAlert**.

- **DASHBOARD node-red:** Interfaccia grafica che permette all'utente di interagire facilmente con il sistema. Per la realizzazione è stato utilizzato node-red, un ambiente di sviluppo per la creazione di flussi di automazione e integrazione tra dispositivi. Formata da 3 schermate principali che mostrano all'utente le funzionalità principali sviluppate. Lo scambio di messaggi è gestito tramite protocollo MQTT, attraverso il broker test.mosquitto.org:1883 e una serie di topic a cui ci si è sottoscritti.
- **APP MQTTAlert:** Applicazione per Android basata su MQTT. È stata utilizzata per notificare all'utente in qualsiasi momento il verificarsi degli eventi più significativi (Allarmi e accessi), per implementare questa funzione è stato necessario sottoscrivere ai topic corrispondenti.

☰ SafeGuard App

Inserimento Codice

Inserire Codice (max 4 cifre)

Codice: *

SUBMIT

CANCEL

CHIUSURA CASSAFORTE

Vuoi modificare il codice? [CLICCA QUI!](#)

Per accedere al record degli allarmi clicca [QUI!](#)

VERIFICA PRESENZA OGGETTO

Cambio Password

Cambio codice

Inserisci vecchio e nuovo codice

Codice vecchio: *

Nuovo codice: *

SUBMIT

CANCEL

Torna alla [schermata di sblocco](#)

☰ Storico Allarmi

Record Allarmi

Motivazione	Data e Ora
Oggetto spostato!	5/21/2023, 6:22:54 PM
Oggetto spostato!	5/21/2023, 6:23:07 PM
Rilevato movimento!	5/21/2023, 6:24:01 PM
Oggetto spostato!	5/21/2023, 6:24:11 PM
Oggetto spostato!	5/21/2023, 6:24:21 PM
Codice sbagliato 3 volte!	5/21/2023, 6:24:45 PM
Codice sbagliato 3 volte!	5/21/2023, 6:25:17 PM
Rilevato movimento!	5/21/2023, 6:26:10 PM
Rilevato movimento!	5/22/2023, 12:05:01 PM

[Torna alla schermata di sblocco](#)

Node-RED

localhost:1880/#flow/2a535143036c70e1

Node-RED

Deploy

filter nodes

common

inject

debug

complete

catch

status

link in

link call

link out

comment

function

function

switch

change

range

template

delay

trigger

Flow 2

Flow 1

link_sblocco

Inserire Codice (max 4 cifre)

CHIUSURA CASSAFORTE

link_cambio

Inserisci vecchio e nuovo codice

Verifica Presenza Oggetto

Codice non valido

Codice valido

Chiusura effettuata

Presenza Oggetto

ALLARME

Codice aggiornato

ERRORE!

Lunghezza=4

Lunghezza=4

function 1

switch

switch

template

codice/sblocco

chiudi/cassa

cambio/codice

verifica/oggetto

show dialog

show dialog

show dialog

link_record


link_sblocco1

MQTT ALERT: Topic e notifiche

**Allarme(Intrusione!!!)** ...


Topic: ALLARME


☒

**Apertura SmartGuardian(Cassaforte aperta).**

Topic: Codice valido


☒


 MQTTAlert • MQTT Message... • ora

Allarme(Intrusione!!!) 

ALLARME:Rilevato movimento!


SILENCE


 MQTTAlert • MQTT Message... • ora

Allarme(Intrusione!!!) 

ALLARME:Oggetto spostato!


SILENCE


 MQTTAlert • MQTT Message... • ora

Allarme(Intrusione!!!) 

ALLARME:Codice sbagliato 3 volte!

SILENCE

 MQTT Message...

Apertura SmartGuardian(Cassaforte a.. 

Codice valido:APERTURA CASSAFORTE...

SILENCE