Esercizio 1:

BOT TELEGRAM + ARDUINO:

Il nostro obiettivo di questo primo esercizio è stato quello di ricevere valori dai sensori montati sull'Arduino Yun e fornire anche la possibilità di inviare comandi di attuazione, il tutto tramite un bot Telegram.

Il codice si sviluppa su due fronti:

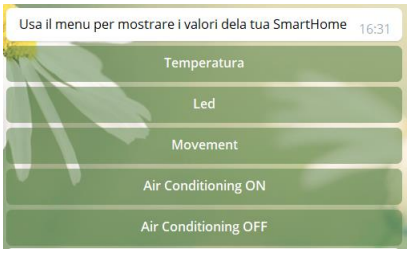
1. **Lato Telegram**: abbiamo sviluppato un bot con Python capace di interrogare il server implementato sull'Arduino Yun.

```
def on_chat_message(msg):
    content_type, chat_type, chat_id = telepot.glance(msg)

    keyboard = InlineKeyboardMarkup(inline_keyboard=[
                    [InlineKeyboardButton(text='Temperatura', callback_data='temp')],
                    [InlineKeyboardButton(text='Led', callback_data='led')],
                    [InlineKeyboardButton(text='Movement', callback_data='move')],
                    [InlineKeyboardButton(text='Air Conditioning ON', callback_data='dcmotor1'
                    [InlineKeyboardButton(text='Air Conditioning OFF', callback_data='dcmotor2
                ])

    bot.sendMessage(chat_id, 'Usa il menu per mostrare/settare i valori della tua SmartHome',

    content_type, chat_type, chat_id = telepot.glance(msg)
```



I metodi implementati sono due:

- *on_chat_message()* permette di ricevere informazioni dall'utente con cui il bot sta chattando e, tramite l'oggetto *InlineKeyboardMarkup()*, siamo in grado di creare il "menu a tendina" mostrato nella figura a fianco. Inoltre, ogni InlineKeyboardButton ha un valore *text* che è la label che apparirà nel pulsante e *callback_data* che è il valore che utilizzeremo per capire, nella funzione on_callback_query, quale dei pulsanti è stato premuto.

- La funzione *on_callback_query*, sulla base del pulsante precedentemente premuto, effettua una richiesta GET, tramite la funzione *requests.get(<url>)*, al server implementato sull'Arduino e attraverso il metodo *sendMessage()* viene inviato all'utente, e quindi stampato sulla schermata del bot, il valore ritornato dalla richiesta GET stessa.

```
def on_callback_query(msg):

    query_id, from_id, query_data = telepot.glance(msg, flavor='callback_query')
    print('Callback Query:', query_id, from_id, query_data)
    if(query_data == 'temp'):
        response = requests.get("http://seba.local/arduino/temperature")
        data_dict = response.json()
        bot.sendMessage(from_id, text='La Temperatura è di: ' + str(data_dict['e'][0]['v'])
```

Le funzionalità fornite dal Bot Telegram sono diverse:

- permette di leggere il valore di temperatura corrente;
- permette di settare lo stato del led;
- permette di contare il numero di movimenti che si sono rilevati nella stanza negli ultimi 60 secondi;
- permette di attivare o disattivare il dcMotor.

2. **Lato Arduino**: come nell'esercizio 1 del Laib hardware 3, abbiamo realizzato un server HTTP sull'Arduino Yun. Il server può soddisfare diverse richieste:

- ricevere comandi di attuazione per alterare lo stato del led o per attivare/disattivare il dcMotor (inteso come aria condizionata in un ambito Smart Home);
- inviare dati riguardanti la temperatura o il numero di movimenti rilevati, sulla base dei valori letti dai sensori.

```
  server.listenOnLocalhost();
  server.begin();
  attachInterrupt(digitalPinToInterrupt(PIR_PIN), checkPresence, CHANGE);
}

void loop() { //gestire le richieste provenienti al server
   BridgeClient client = server.accept(); //accetta nuovi client
   if(client){
    process(client);
    client.stop();
   }
   delay(50);
}
```

Sketch: nella funzione *setup()*, oltre ai comandi per settare i pin, abbiamo configurato un http server sulla Yun e abbiamo definito la funzione *attachInterrupt()*, nella quale, oltre a specificare il pin che vogliamo monitorare, dobbiamo specificare il nome della funzione da eseguire quando si scatena l'interruzione. In questo caso al verificarsi di un evento sul PIR_PIN viene invocata la funzione *checkPresence()* che conta il numero di movimenti rilevati.

Questo server risponde a tutti i client che inviano delle richieste e per ogni client viene avviato un processo tramite la funzione *process()*. L'obiettivo della funzione *process()* è quello di "scomporre" l'URL in due elementi e

sulla base di questi elementi soddisfare l'una o l'altra richiesta. Ad esempio, se il comando fosse "led", verrebbe modificato lo stato attuale del led; se il comando fosse "dcmotor1", verrebbe avviato il dcMotor sulla base del valore di temperatura corrente e così via. All'interno di essa grande rilevanza assume la funzione *printResponse()*, che permette di stampare il body della risposta http. Inoltre, quest'ultimo viene codificato come file con formato SenML JSON tramite la funzione *senMlEncode()*, nella quale abbiamo creato manualmente un file "stile JSON" che poi, attraverso la funzione *serializeJson()*, viene trasformato in stringa JSON e usato nel body della risposta http.

Esercizio 2:

Obiettivo: Il nostro obiettivo di questo secondo esercizio è stato quello di ricevere valori dai sensori montati sull'Arduino Yun e fornire anche la possibilità di inviare comandi di attuazione, il tutto tramite un'interfaccia grafica. Per realizzarla abbiamo usufruito della libreria Tkinter.

```
window = tk.Tk()
window.geometry("480x500")
window.title("MANAGE ARDUINO")
window.configure(background = "gold")
window.grid_columnconfigure(0, weight=1)
```

In un primo momento abbiamo creato il nostro widget principale e gli abbiamo assegnato una dimensione e un titolo.

```
string = """Premi:
1. Temperature = per sapere il valore di temperatura;
2. Led = per cambiare lo stato attuale del led
3. Movement = per sapere il numero di movimenti rilevati negli ultimi 60 sec
4. DcMotorON = per avviare la ventola
5. DcMotorOFF = per stoppare la ventola\n"""
textwidget = tk.Text(height=8)
textwidget.configure(background = "pale green", fg="orangered")
textwidget.insert(tk.END, string)
textwidget.grid(row=8, column =0, sticky="WE", padx = 10)
```

Abbiamo anche definito un widget testuale che informa l'utente sulle funzionalità fornite da ogni Button.
Ad ognuno di essi è associata l'esecuzione di un comando e, per esempio, nel caso mostrato in figura, nel momento in cui viene premuto il tasto "Temperature"

```
download_button = tk.Button(text="Temperature", command=temperature, background = "white smo
download_button.grid(row=3, column=0, sticky="WE", pady=5, padx=50)
```

viene eseguita la funzione *temperature()*. Questa funzione permette di sfruttare i metodi forniti dal web server implementato nell'Arduino (come nell'esercizio precedente) e, in questo caso, alla ricezione del valore di temperatura viene creato un nuovo elemento widget testuale nel quale viene mostrato il valore di temperatura stesso.

```
def temperature():
    output = requests.get("http://seba.local/arduino/temperature").json()
    text_response = f"Il valore di temperatura è: {output['e'][0]['v']}"
    textwidget = tk.Text(height=2)
    textwidget.configure(background = "deep sky blue")
    textwidget.insert(tk.END, text_response)
    textwidget.grid(row=1, column =0, sticky="WE", padx = 10)
```

Questo codice viene ripetuto per riuscire a sfruttare tutte le funzionalità che vengono offerte dal server Arduino, e quindi possiamo:

- leggere il valore di temperatura corrente;
- settare lo stato del led;
- contare il numero di movimenti che si sono rilevati nella stanza negli ultimi 60 secondi;
- attivare o disattivare il dcMotor.