Università Politecnica delle Marche

FACOLTÀ DI INGEGNERIA

Corso di Laurea Triennale in Ingegneria Informatica e dell'Automazione

TESI DI LAUREA TRIENNALE

Studio e implementazione di strumenti di robotica educativa per l'identificazione e modellazione dell'apprendimento

Study and implementation of educational robotics tools for the identification and modeling of learning

Candidato:
Marco Cervigni
Matricola 1089480

Relatore:
David Scaradozzi

Correlatore:
Laura Screpanti
Lorenzo Cesaretti

Anno Accademico 2020-2021

Ringraziamenti

Sono giunto al termine di un lungo percorso e sono certo che non sarebbe stato possibile senza il contributo di chi mi è stato accanto nei momenti più duri, ritengo quindi doverosi dei ringraziamenti. Comincio con il ringraziare la mia famiglia, per avere reso possibile questo viaggio e per essere sempre al mio fianco, vi sarò per sempre grato. Ci tengo inoltre a ringraziare il mio relatore Prof. David Scaradozzi e i ragazzi che ci hanno assistito nel corso del tirocinio, Laura Screpanti e Lorenzo Cesaretti .

Sommario

Lo scopo di questa tesi è lo sviluppo di un sistema che supporti l'acquisizione di dati da esperienze di robotica educativa. Durante una tipica attività in classe di robotica educativa gli studenti sono divisi in gruppi ed esplorano i fondamenti della robotica attraverso la costruzione e la programmazione di kit educativi, come ad esempio il Lego Mindstorms EV3 o prodotti similari. In genere, viene loro chiesto di risolvere piccole sfide che portano alla luce tutte le loro capacità di problem solving attraverso il processo di costruzione e programmazione del kit robotico. Obiettivo della presente tesi è costruire un sistema che tenga traccia delle interazioni degli studenti con il robot e con il suo ambiente di sviluppo, in modo tale da poter implementare in modo efficiente una raccolta dati sul campo in maniera automatizzata. Questa raccolta dati consentirà di comprendere il modo in cui gli studenti e le studentesse affrontano una sfida robotica. In questa tesi si descrive il sistema di raccolta dati sviluppato a partire dall'ambiente di programmazione UIFlow, in cui gli studenti, usando un linguaggio di programmazione grafica, programmano robot perché eseguano determinate azioni. Si descriverà il file di log, contenente le interazioni degli studenti con il robot e il suo ambiente di programmazione, e il meccanismo automatico di traduzione delle azioni degli studenti in stringhe. Il sistema illustrato consentirà interessanti sviluppi futuri; esso, infatti, è il primo passo per la creazione di un' infrastruttura con la capacità di analizzare i dati raccolti attraverso tecniche di machine learning e intelligenza artificiale, consentendo così la creazione di tecnologie in grado di supportare gli insegnanti nella valutazione delle attività di robotica educativa e anche nella loro attività didattica.

Indice

Elenco delle figure

Listato

1 Introduzione

La Robotica nell'istruzione (Robotics In Education, RIE) è un'ampia area di applicazioni della robotica nel campo dell'istruzione. All'interno della RIE la Robotica educativa (ER) aiuta gli studenti a esplorare idee potenti e un apprendi- mento autentico in ambienti senza limiti (open-ended learning environments, OELEs), dove gli strumenti hardware e software consentono di esplorare e creare soluzioni a un determinato compito. Nonostante la valutazione pedagogica e l'identificazione di tale apprendimento siano cruciali, non è ancora chiaro come misurare i risultati dell'apprendimento degli studenti e come certificarli in un regolare curriculum . [1]

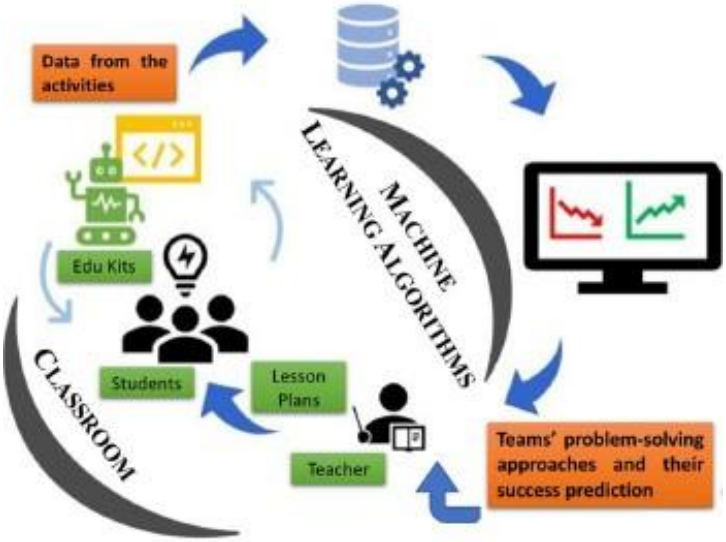

Figura 1: Approccio Robotica Educativa

Come illustrato nella figura 1, durante una comune attività di robotica educativa gli studenti interagiscono con kit robotico, come ad esempio il kit Lego Mindstorm EV3. Questi kit consentono agli studenti di programmare e costruire un robot. L'ambiente di programmazione può essere grafico o testuale: nella prima gli studenti programmano il robot tramite il drag and drop di icone (definite "blocchi") le quali rappresentano visivamente un insieme di funzioni che il robot eseguirà. Il codice che risulterà da queste operazioni verrà inviato a un dispositivo in grado di interpretarle ed eseguirle. Nella presente tesi si prenderà in considerazione il dispositivo M5StickC (figura 2), una scheda di sviluppo estremamente versatile programmabile in diverse modalità.

Figura 2: M5StickC

L'obiettivo della presente tesi è tenere traccia delle operazioni eseguite dagli studenti con l'M5StickC. A tal fine, il sistema proposto si propone di raccogliere dati grezzi sull'interazione degli studenti con il toolkit basato su M5StickC e di archiviarli all'interno di un server, in modo tale da renderli disponibili per una successiva analisi.

Per realizzare questo, verranno apportate delle modifiche all'ambiente di programmazione grafica in modo tale che l'informazione riguardante le serie temporali delle interazioni degli studenti con i robot possa essere memorizzata con una stringa. A tal fine, verranno registrati anche i blocchi utilizzati durante la risoluzione della sfida di robotica educativa. Quando l'M5StickC esegue il programma ideato dagli studenti, contemporaneamente esegue una routine di scrittura all'interno del server. In un futuro lavoro, i dati raccolti grazie a questo sistema saranno analizzati per trovare evidenze del processo di apprendimento degli studenti. Queste evidenze consentiranno agli insegnanti di comprendere gli approcci di risoluzione dei problemi degli studenti; inoltre, ad un livello più elevato potranno mostrare ilprogresso degli studenti nel corso del tempo.

Nella sezione 2 di questa tesi si illustreranno i materiali e i metodi utilizzati in questo progetto. Nella terza sezione si descriveranno i risultati ottenuti e nella quarta le limitazioni che sono state riscontrate. Infine nell'ultima sezione saranno presenti le conclusioni e i possibili sviluppi futuri che potranno integrare il lavoro svolto.

2 Materiali e Metodi

In questa unità verranno descritti gli strumenti hardware utilizzati e i software necessari allo sviluppo del sistema oggetto di questa tesi. Nello specifico, verranno descritti in modo approfondito l'M5StickC e l'M5StickC PLUS (figura 3). In seguito, verranno analizzati UIFlow, una piattaforma di programmazione appositamente progettata per i dispositivi M5Stick, e Visual Studio Code, un editor di codice che sarà usato per eseguire codice Python. Infine, verrà descritta la piattaforma di comunicazione Discord, usata in questa fase come server e preferita ad altre piattaforme, come ad esempio Telegram.

2.1 Il kit robotico

2.1.1 Hardware

L'M5StickC [2] è una scheda di sviluppo estremamente versatile programmabile in diverse modalità e permette di realizzare facilmente e rapidamente progetti in ambito Internet of Things (IoT). Il piccolo "mattoncino" ha dimensioni esigue, 5x2.5 cm. Al suo interno contiene:

- microprocessore ESP 32
- display da 0,96 pollici (risoluzione 160×80)
- microfono
- trasmettitore IR
- wi-fi
- bluetooth
- accelerometro
- giroscopio (6 gradi di libertà)
- LED di segnalazione integrato
- due pulsanti (A e B) programmabili
- batteria da 80 mAh
- memoria flash da 4MB
- modalità di programmazione via USB e WiFi
- Linguaggi di programmazione: UIFlow (Blockly), C (pronto per essere programmato con IDE Arduino), MicroPython

Figura 3: M5stickC e M5stickC PLUS

L'M5StickC PLUS presenta alcune differenze rispetto al M5StickC; infatti, ha uno schermo leggermente più ampio, una batteria più grande e un buzzer incluso. Entrambi i dispositivi sono compatibili con la piattaforma di programmazione UIFlow.

2.1.2 Software

UIFlow è una piattaforma di programmazione appositamente progettata per i dispositivi prodotti dall'azienda cinese M5Stack. L'IDE di programmazione grafico a blocchi di questa piattaforma è basato su Blockly, un linguaggio grafico di programmazione ben conosciuto in campo didattico.

UIFlow consente inoltre di programmare qualsiasi oggetto M5StickC anche in Micro- Python, implementazione di Python 3 per microcontrollori e sistemi embedded. UIFlow fornisce le funzionalità necessarie per la realizzazione, in maniera estremamente semplice, di progetti con forte interazione con il mondo reale, ideale quindi per chi si avvicina al mondo dell'automazione e della programmazione.[3]

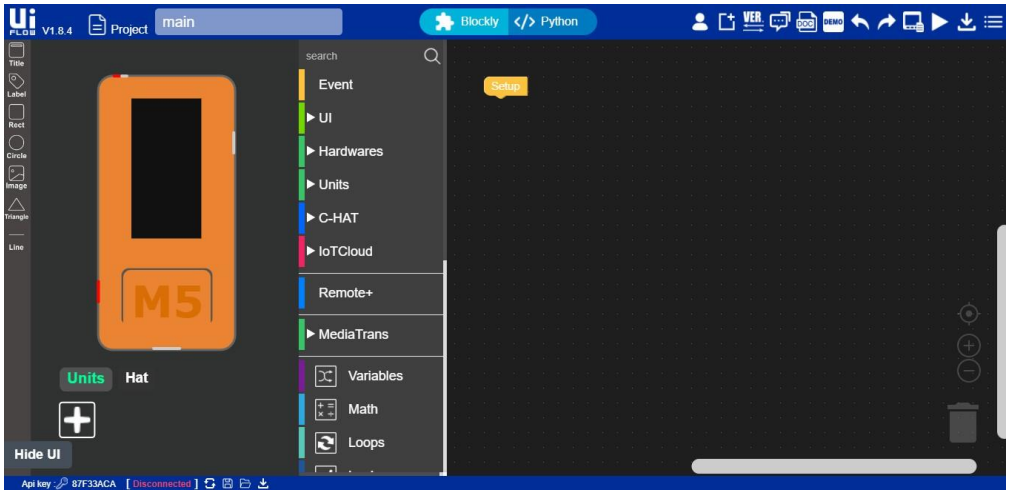

Figura 4: Interfaccia utente dell'ambiente di programmazione UIFlow nella versione 1.8.4

L'IDE UIFlow può essere utilizzato on-line oppure localmente su pc, scaricandolo. Nell'interfaccia dell'IDE UIFlow (figura 4) si può immediatamente notare come il codice è implementabile sia in Blockly (quindi utilizzando blocchi precostruiti che eseguono determinate funzioni come ad esempio accendere o spegnere il led dell'M5StickC) oppure in python.

In basso sono presenti i comandi per connettere l'M5StickC, selezionare la corretta API key, e salvare o caricare contenuti al suo interno. Si può connettere l'M5StickC sia via USB con l'apposito cavo sia mediante Wi-Fi, per questa seconda opzione basta far sì che sia connesso allo stessa rete del pc dal quale si usa UIFlow.

Una volta connesso il dispositivo è possibile programmare il comportamento del dispositivo M5StickC e di ogni sua parte; ad esempio, è possibile modificarne il display aggiungendo un titolo, etichette, immagini e molto altro.

Nonostate le numerose funzioni disponibili nell'interfaccia UIFlow, nella presente tesi verranno illustrati gli sviluppi apportati a un ristretto numero di funzioni, cioè quelle funzioni che verranno usate dagli studenti nella risoluzione di esercizi di introduzione alla robotica.

La differenza fondamentale che sussiste fra UIFlow online e la versione locale, è il fatto che solo nella versione online è possibile utilizzare UIFlow Block Maker (figura 5), ovvero l'editor dei blocchi di UIFlow che consente di creare nuovi blocchi utilizzando codice python. [4][5]

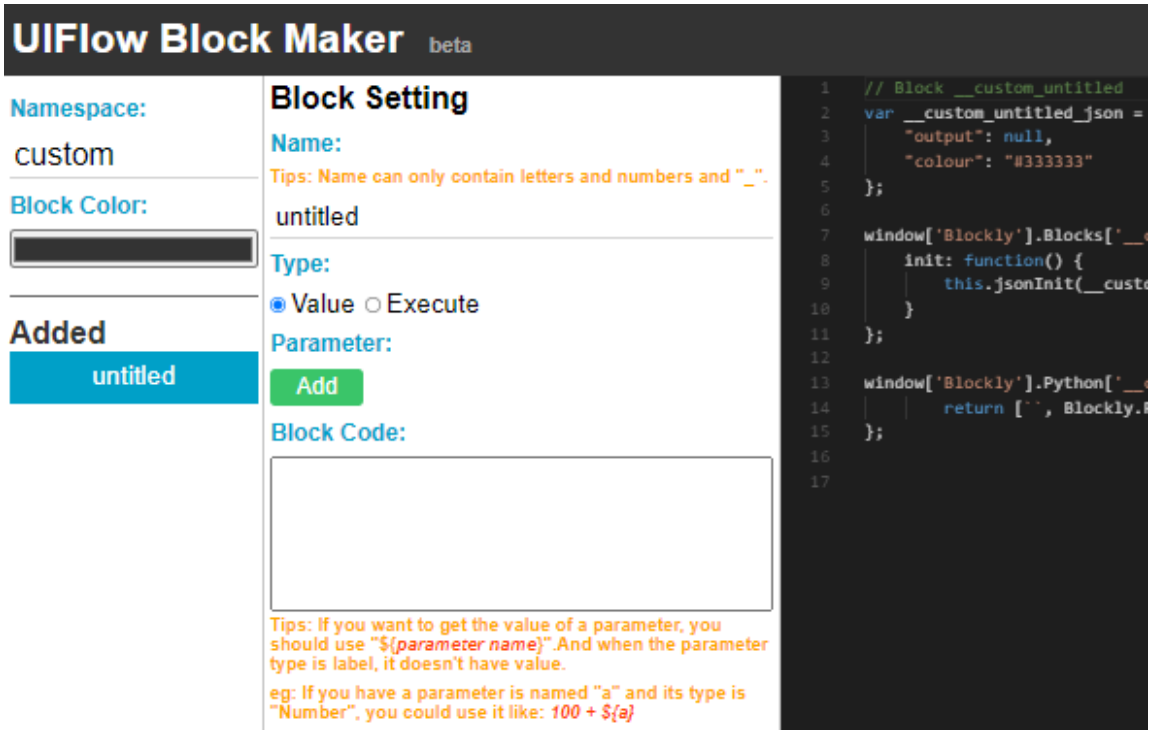

Figura 5: UIFlow Block Maker

UIFlow Block Maker consente di implementare funzioni in blockly mediante codice python, che possono essere scaricati e in seguito utilizzati con l'M5StickC. L'editor permette di definire:

- nome del blocco
- colore
- tipo di blocco: *'value'* se il blocco ritorna un valore, *'execute'* se esegue un comando senza ritornare nulla
- i vari parametri, per ognuno dei quali vanno delineati nome e tipo (*label* se è una semplice etichetta, *string* se è una stringa, *number* se è numero e *variable* per le variabili)
- va inoltre inserito il codice del blocco, in python; per richiamare i parametri si usa la dicitura *"${NomeParametro}"*
- è possibile infine caricare blocchi già ultimati, e scaricare i blocchi che vengono implementati

2.2 Il server web

Per tenere traccia dei blocchi eseguiti su UIFlow e salvarli in un file di log è stato utilizzato un server di Discord (figura 6), una piattaforma di messaggistica e distribuzione digitale ideata per la comunicazione tra videogiocatori. Gli utenti comunicanocon chiamate vocali, messaggi di testo, media e file in chat private o come membri di un server (figura 7), i quali altro non sono che una raccolta di canali di tipo vocale o testuale. [6]

Figura 6: Il logo di Discord

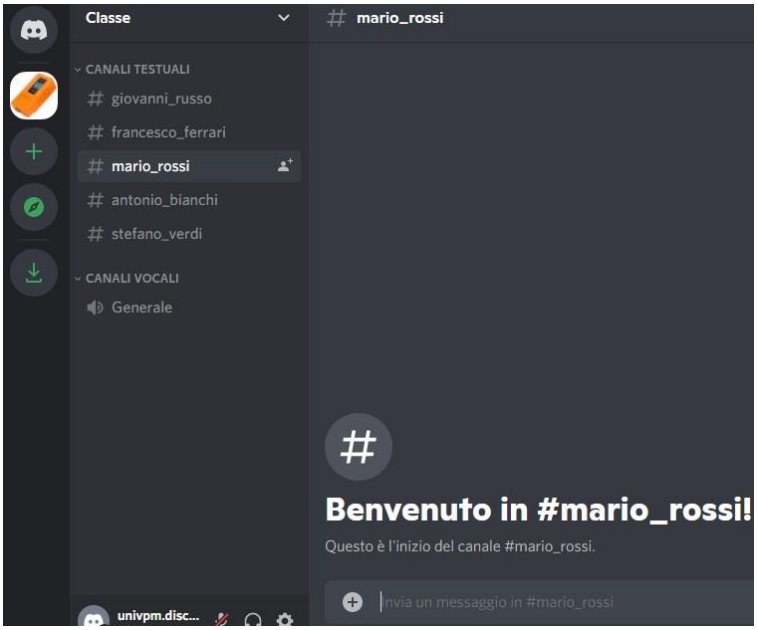

Figura 7: Un esempio di canale testuale

Discord consente, una volta effettuata la registrazione mediante email e password, di creare server e invitare persone al suo interno. È inoltre possibile creare canali testuali e vocali, e stabilire chi all'interno del canale ha la possibilità di inviare messaggi e chi invece può solo leggere.

Discord verrà usato creando un canale testuale per ogni gruppo di studenti (e quindi uno per ogni M5StickC) all'interno del quale vengano salvati i dati delle interazioni di quel gruppo, quindi tutte le operazioni da loro eseguite.

Anche Telegram (figura 8) è un servizio di messaggistica e broadcasting; e come Discord, anche Telegram consente lo scambio di messaggi di testo, chiamate vocali su IP e la creazione di grandi gruppi di utenti. I gruppi in Telegram sono progettati per contenere fino a 200 membri in cui ogni membro può aggiungere nuovi membri. Una volta che il gruppo supera il limite di 200 membri, si può trasformare in un supergruppo che contiene fino a 100.000 membri. Su questo aspetto, Discord funziona in modo diverso ed è altamente scalabile. Un suo server può contenere un massimo di 500 canali senza limiti al numero di utenti che i canali o i server possono integrare. Discord inoltre consente un maggiore controllo dei propri server, in quanto si basa sull'assegnazione di ruoli agli utenti. I membri possono essere nominati per ruoli diversi come abbonato, amministratore e moderatori. In conclusione, Discord sembra dare maggiori garanzie di sicurezza e prestazioni coerenti con l'obiettivo di salvare dati dalle interazioni di studenti durante attività di robotica educativa inviandoli direttamente dal dispositivo prescelto, ovvero l'M5stickC [7].

Figura 8: Logo di Telegram

2.3 Ambiente di sviluppo

Una volta forniti i dispositivi M5StickC agli studenti bisogna far si che ogni M5StickC invii i messaggi nel canale testuale associato al gruppo (uno per ogni 3-4 studenti). Per ottenere ciò è stato creato un software python capace di modificare l'URL di destinazione dei messaggi inviati dai blocchi di UIFlow; per

programmare in python abbiamo utilizzato Visual Studio Code.[8]

Figura 9: Logo di Visual Studio Code

Visual Studio Code (figura 9) è un editor di codice sorgente che può essere utilizza-to con diversi linguaggi di programmazione, come la famiglia di linguaggi C, HTML, Java e molti altri. Incorpora un insieme di funzioni che variano a seconda del linguaggio che si sta usando.

Il suo uso si è reso necessario per definire un software capace di modificare l'URL di destinazione dei messaggi inviati dai blocchi di UIFlow. Per fare questo si è scelto il linguaggio di programmazione Python 3.10 .

Grazie ad esso è stato possibile definire un software che prende in input l'URL del canale testuale di Discord, nel quale si intende inviare i messaggi che descrivono le operazioni effettuate, e sostituirlo in ognuno dei blocchi creati, e ripetere il procedimento per ogni gruppo di studenti così da avere i blocchi pronti per ogni gruppo.

2.4 Il sistema di raccolta dati

In questa sezione verrà descritto la logica del sistema di raccolta dati. In un ambiente aperto gli studenti interagiranno con il kit di robotica educativa e risolveranno le sfide proposte dall'insegnante, collaborando e seguendo un approccio laboratoriale allo sviluppo di competenze e conoscenze. Il sistema intende essere minimamente invasivo in questo processo. Per tale motivo i ragazzi non saranno chiamati a compiere alcuna azione ulteriore rispetto a quelle proposte dall'insegnate o dall'attività esplorativa connaturata con la metodologia della robotica educativa.

I ragazzi interagiranno con l'M5StickC che precedentemente saranno stati predisposti dall'insegnante o educatore secondo la seguente procedura:

- Utilizzare 1 M5stickC per ogni gruppo di studenti (in genere gruppi di 3 o 4 alunni); l'M5StickC dovrà essere configurato per connettersi alla stessa rete wifi del pc usato dagli studenti

- Creare un canale testuale su Discord per ogni gruppo, all'interno dello stesso server chiamato 'classe'; ogni canale sarà identificato da un URL reperibile tramite "Impostazioni→Altri strumenti->Strumenti per Sviluppatori". A questo punto basta inviare un messaggio qualunque all'interno del canale testuale scelto, e selezionare "messages", dove è presente l'URL alla sezione "Request URL"

- Copiare l'URL di ogni canale testuale Discord

- Mandare in esecuzione il programma python *SwitchID.py* (vedi Sezione 3.2) inserendo come parametro l'URL di un canale Discord e ripetere l'esecuzione per ogni URL (uno per ogni gruppo di studenti), in modo da avere tutti i blocchi aggiornati per ogni M5StickC

- Per ogni dispositivo M5StickC associare un pc in cui sia stata salvata la libreria con i blocchi di codice modificati e inizializzati con l'URL del canale Discord associato; questo consentirà di fare l'upload della libreria tramite UIFlow

- Sul piano didattico, durante l'attività spiegare agli studenti che al termine del codice che hanno assemblato per controllare il robot va sempre aggiunto il blocco 'end' per segnalare al robot il termine del programma; sul piano tecnico questa istruzione servirà per segnalare nel file di log l'inizio di una nuova esecuzione del programma.

- Dopo che gli studenti hanno svolto gli esercizi che sono stati loro assegnati, raccogliere i dati salvati su Discord, dove è già indicata la data e l'orario preciso di arrivo dei messaggi contenenti le operazioni utilizzate.

3 Risultati

In questa sezione verranno descritti tutti i blocchi implementati mediante UIFlow Block Maker con i relativi codici. Non è stato possibile implementare tutti quanti i blocchi predefiniti di UIFlow a causa di diverse problematiche riscontrate che verranno analizzate in seguito nella sezione “Limitazioni”. Nelle figure il token presente all’interno dei codici è stato oscurato per questioni di privacy, in quanto è associato a un email.

3.1 Implementazione dei blocchi di programmazione grafica

3.1.1 LED

La sezione ’Hardwares’ di UIFlow contiene operazioni che fanno funzionare determinate componenti hardware. In questo caso sono stato modificati i blocchi che consentono di accendere e spegnere il led presente nell’M5StickC.

```
  import urequests

M5Led . on ()

wait_ms (500)

try:
  req = urequests.request(method=' POST ', url=' server_url ', json={'
    content ' :' #led_on;end '}, headers={' authorization ' :' token '})
  print(' Successo ')
except:
  print(' Fallimento ')

wait_ms (500)

```

Listato 1: LED_ON

Il codice che provvede allo spegnimento del led è quasi identico, fatta eccezione per il comando ’M5Led.on()’ (che diventa ’M5Led.off()’) e il messaggio inviato mediante chiamata post (’#led_on’ diventa ’#led_off’)

3.1.2 Math

All’interno della sezione Math sono presenti tutti i blocchi che consentono di effettuare operazioni matematiche, come somme e elevazioni a potenza, ma anche conversione in *’float’* o in *’int’* di un numero.

```

${ a}+${ b}

wait_ms (500)

import urequests
```

```
7
8  try:
9    req = urequests.request(method=' POST', url=' server_url', json={'
       content' :' # sum ; var1 : ${ a}; var2 : ${ b}; end' }, headers ={ '
       authorization' :' token' })
10   print(' Successo')
11 except:
12   print(' Fallimento')
13
14 wait_ms(500)
15
16
```

Listato 2: Somma

I blocchi sottrazione, prodotto, divisione, remainder (calcola il resto della divisione) ed elevazione a potenza sono simili al blocco somma, variano solo l'operazione e nome del blocco inviato per messaggio su discord.

```
1
2  ${  a}  >0
3
4  wait_ms (500)
5
6  import urequests
7
8  try:
9    req = urequests.request(method=' POST', url=' server_url', json={'
       content' :' #is_positive;var1:${a};end' }, headers={' authorization
       ' :' token' })
10   print(' Successo')
11 except:
12   print(' Fallimento')
13
14 wait_ms(500)
15
16
```

Listato 3: is positive

Il blocco *'is positive'* restituisce *true* se il numero inserito come parametro è maggiore di 0, altrimenti *false*. Discorso analogo ma opposto per il blocco 'is negative'.

```
1
2  (${a}%2) ==0
3
4  wait_ms (500)
5
6  import urequests
7
8  try:
9    req = urequests.request(method=' POST', url=' server_url', json={'
       content':'#is_even;var1:${a};end'}, headers={'authorization':'
       token'})
10   print('Successo')
11 except:
12   print('Fallimento')
13
14 wait_ms(500)
15
```

Listato 4: is even

Il blocco *'is even'* verifica se il numero inserito in ingresso sia pari, controllando se il resto della divisione tra il parametro e 2 sia pari a 0, in tal caso restituisce true, altrimenti false. Discorso analogo per il blocco *'is odd'*, ma controlla che il resto della divisione sia pari a 1.

```
1
2  int (${a})
3
4  wait_ms (500)
5
6  import urequests
7
8  try:
9    req = urequests.request(method='POST', url='server_url', json={'
       content':'#convert_to_int;var1:${a};end'}, headers={'
       authorization':'token'})
10   print('Successo')
11 except:
12   print('Fallimento')
13
14 wait_ms(500)
15
16
```

Listato 5: Converti_in_int

Il blocco converti_in_int effettua la conversione del numero inserito come parametro, in intero. *'converti_in_float'* opera allo stesso modo, ma converte in float(numero in virgola mobile).

3.13 Timer

Nella sezione timer su UIFlow sono presenti i blocchetti che consentono di inserire un tempo di attesa fra le varie istruzioni.

```
1
2  import urequests
3
4  wait(${seconds})
5
6  wait_ms  (500)
7
8  try:
9    req = urequests.request(method='POST', url='server_url', json={'
       content':'#wait_seconds;var1:${seconds};end'}, headers={'
       authorization':'token'})
10   print('Successo')
11 except:
12   print('Fallimento')
13
14 wait_ms(500)
15
16
17
```

Listato 6: wait_s

Il blocco *'wait_s'* impone una pausa di un numero di secondi pari al numero inserito come parametro. Lo stesso per il blocco *'wait_ms'*, solo che l'attesa viene definita in millisecondi.

3.1.4 Blocco End

Il blocco *'End'* invia semplicemente un messaggio su Discord contenente la dicitura "##". Viene utilizzato alla fine di ogni successione di operazioni, per indicare che l'esecuzione è terminata.

```
1  wait_ms(500)
2
3  import urequests
4  try:
5    req = urequests.request(method='POST', url='server_url', json={'
       content':'##'}, headers={'authorization':'token'})
6    print('Successo')
7  except:
8    print('Fallimento')
9
10 wait_ms(500)
11
```

Listato 7: End

3.2 Setup del server web

Come già detto l'obiettivo di questo progetto è di salvare i dati che contengono le operazioni effettuate dagli M5StickC in un file di log. Un log file contiene l'intera "storia" delle operazioni effettuate da un utente o da una macchina. In un log file, infatti, vengono registrate tutte le operazioni, in ordine cronologico, svolte nel normale utilizzo di un software, di un applicativo o più semplicemente di un computer. Quindi nel nostro caso, per andare a salvare i dati, sono stati modificati i blocchi predefiniti di UIFlow, utilizzando UIFlow BlockMaker, in modo tale che inviassero su Discord i messaggi contenenti informazioni sull'operazione effettuata (figura 10). Ad esempio per modificare il blocco "somma", che prende due parametri e ne effettua l'addizione, bisogna innanzitutto ricreare il blocco identico, poi aggiungere un protocollo per lo scambio dei messaggi.

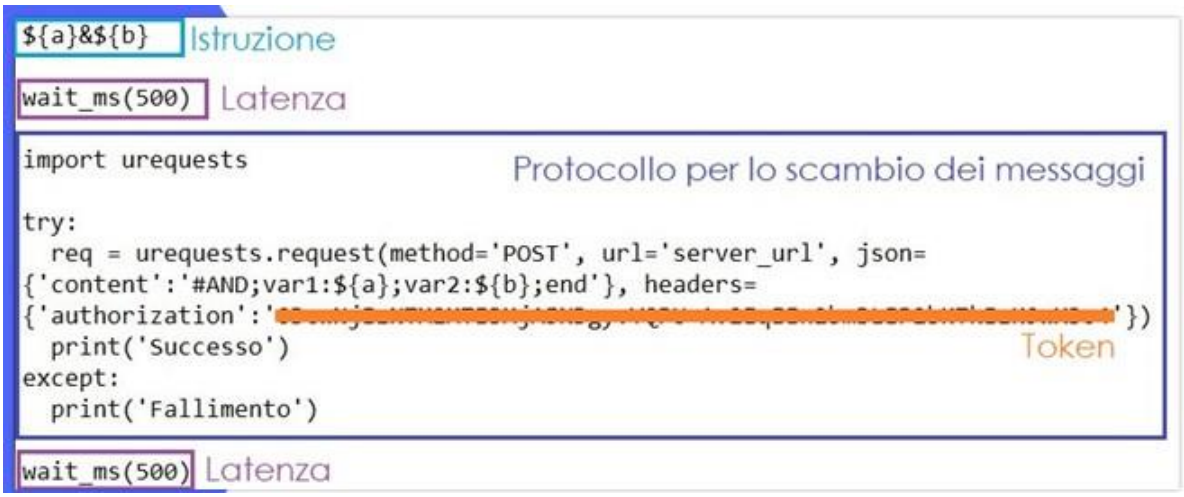

Figura 10: Esempio di blocco somma ricreato in UIFlow BlockMaker

La prima riga di codice ricrea il blocco "somma" tale e quale. La latenza è stata inserita per risolvere un problema riscontrato nell'invio dei messaggi, di cui si parlerà in seguito. Abbiamo poi il protocollo per lo scambio di messaggi, che consiste in una richiesta HTTP, nello specifico una chiamata 'POST', che invia direttamente nel canale testuale Discord un messaggio contenente le informazioni sul blocco utilizzato.

La parte di codice che invia il messaggio include:

- L'import della libreria '*Urequests*', che mette a disposizione le principali funzionalità di HTTP
- Un '*try...except*' necessario per gestire le eventuali eccezioni

- La Richiesta'POST' che invia il messaggio: la dicitura '*server_url* ' viene sostituita con l'URL del canale testuale Discord nel quale va inviato il messaggio, grazie al software python
- '*content*' indica il nome dell'operazione effettuata dall'M5StickC
- '*var1*, *var2* ' sono i vari parametri che vengono forniti al metodo utilizzato
- '*authorization*' è il token, un codice identificativo che permette all'M5StickC di postare messaggi sui canali testuali discord, sottoforma di un utente, al quale è stato assegnato il nome di UnivPM.TalkingBlocks.M5. Il token identifica quindi un utente fittizio che invia i messaggi
- una stampa (*print*) utile per avere un ulteriore conferma in caso di successo

Grazie al comando di UIFlow Block Maker *"${parametro}*" i parametri utilizzati nell'istruzione eseguita dall'M5StickC vengono inviati sul canale Discord insieme al nome dell'operazione, come nella figura 11.

Figura 11: Esempio di messaggi ricevuti su discord

Discord consente inoltre di tenere traccia del momento in cui viene inviato il messaggio, il timestamp è quindi già incluso.

Come già descritto in , è stato utilizzato un programma in codice python per definire nei blocchi l'URL di destinazione dei messaggi. Il software si occupa di sostituire la dicitura *'server_url'* all'interno del protocollo per lo scambio dei messaggi con l'URL del canale discord desiderato all'interno del quale inviare i messaggi. Il programma *SwitchID.py* va eseguito da chi fornisce gli M5StickC agli studenti, una volta aver diviso questi ultimi in gruppi, in modo da avere un canale testuale per ogni gruppo.

```
import os

text_channel_id=input("Scrivere l'ID del canale testuale che si intende associare ai blocchi di Blockly: ")

texttofind = 'server_url'
texttoreplace = "https://discord.com/api/v9/channels/" + text_channel_id + "/messages"
sourcepath = os.listdir('Blocchetti_M5/')

for file in sourcepath:
    inputfile = 'Blocchetti_M5/' + file
    print('Conversion is ongoing for:' +inputfile)
    with open(inputfile, 'r') as inputfile:
        filedata = inputfile.read()
        freq = 0
        freq = filedata.count(texttofind)
        destinationpath = 'OutputFile/' + file #modificare il nome della cartella di destinazione all'occorrenza
        filedata = filedata.replace(texttofind,texttoreplace)
        with open(destinationpath,'w') as file:
            file.write(filedata)
            print('Total %d Record Replace' %freq)
            print("\n")
            print ("L'ID inserito e':" + text_channel_id + "\n" + "I blocchi personalizzati si trovano nella cartella OutputFile")
```

Listato 8: SwitchID.py

Una volta mandato in esecuzione, il programma chiede all'utente (in questo caso chi gestisce gli M5) di inserire l'URL del canale testuale discord dove si vuole tenere traccia delle operazioni eseguite da UIFlow. In seguito sostituisce la dicitura *'server_url'* con l'URL inserito, in ogni file all'interno della cartella *Blocchetti_M5* , che contiene tutte le operazioni implementate in blockly. Infine, salva tutti questi file (di estensione *'.m5b'*) nella cartella *OutputFile*, dove possono venire prelevati e caricati su UIFlow (come illustrato in sezione 2.4).

4 Limitazioni

Lo sviluppo del sistema di raccolta dati ha raggiunto una fase di sviluppo del prototipo funzionante in laboratorio. Tuttavia già in questa fase sono state evidenziate delle criticità che potranno essere risolte in una fase successiva di raffinamento del sistema e comunque prima di poter procedere all'implementazione in ambienti rilevanti.

Una limitazione del sistema è legata alla ricezione dei messaggi sul canale Discord. Una volta testato il sistema in laboratorio, il funzionamento desiderato del sistema prevedeva che, man mano che il dispositivo M5StickC eseguiva le istruzioni contenute nei blocchi software, le stringhe associate venivano trasmesse al server Discord e subito venivano rappresentate nel canale testuale associato. Questo funzionamento è stato messo in crisi dall'esecuzione di più istruzioni in successione su UIFlow (in un'unica esecuzione); in questo caso, non tutti i messaggi venivano inviati su Discord e alcuni andavano perduti. Una prima soluzione al problema è stata proposta e verificata inserendo blocchi *'wait'* di 0.5 secondi all'inizio e alla fine di ogni blocco di istruzioni, i quali vanno a costituire quella che è stata chiamata 'latenza', in modo tale che l'M5StickC avesse più tempo per eseguire il comando e inviare il messaggio. La scelta di inserire mezzo secondo di attesa è stata presa dopo aver effettuato diversi test, tra i quali quello con maggiore successo (figura 12) mostra come, su 10 esecuzioni di 10 operazioni ciascuna, 8 esecuzioni inviano tutti e 10 i messaggi, in 2 di esse vengono perduti 4 messaggi in totale, è stata quindi data per buona questa soluzione.

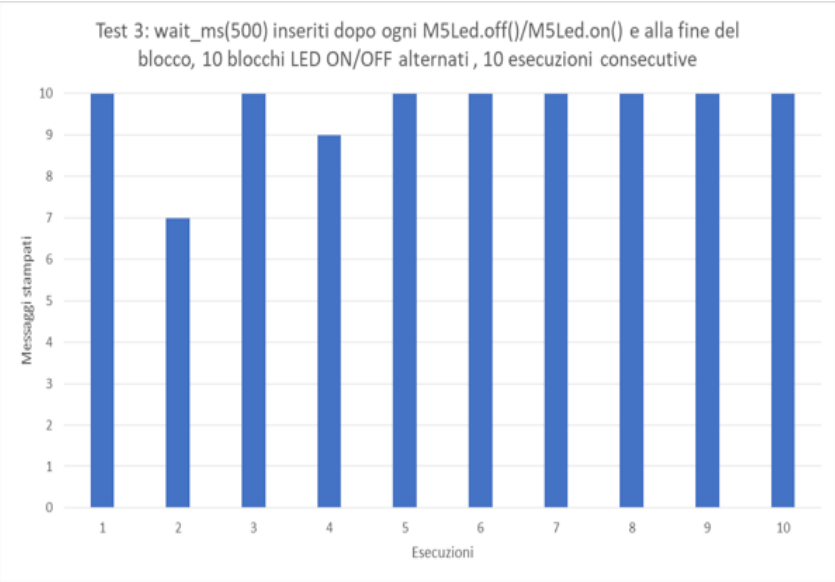

Figura 12: 10 esecuzioni identiche, con 10 operazioni ciascuna

Il secondo problema riscontrato nell'implementazione dei blocchi software riguarda la possibilità di rappresentare i blocchi stessi in modo grafico ed accurato, in modo tale da non far accorgere l'utente di star usando un blocco modificato. Alcuni blocchi presenti nella libreria standard di UIFlow sono risultati impossibili da realizzare,per diverse motivazioni.

Alcuni blocchi della sezione *'Math'*, affinché possano essere eseguiti richiedono di dover importare una libreria, come *'random fraction'*, che necessita della libreria "*random*", la radice quadrata, logaritmo, seno, coseno e tangente, che necessitano della libreria "*Math*". Il problema è dato dal fatto che, essendo dei blocchi di tipo *'value'*, che quindi restituiscono un valore, il primo comandodel blocco deve essere necessariamente quello che importa la libreria (*import random*); quindi qualora ad esempio volessimo assegnare a una variabile un valore randomico, o risultato di una radice, UIFlow assegna immediatamente alla variabile il valore *'import random'* , risultando quindi ovviamente in un errore (figura 13) .

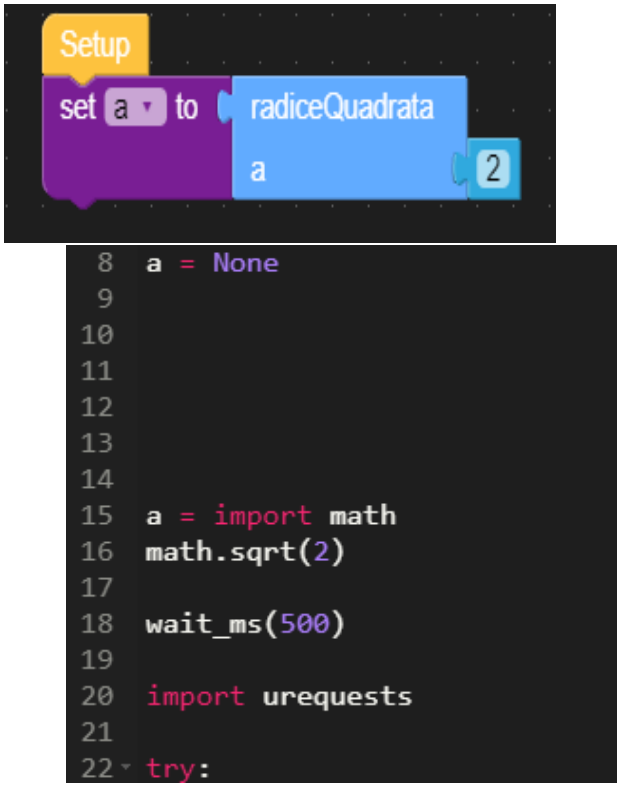

Figura 13: blocco radice quadrata

Sono stati riscontrati diversi problemi anche con i blocchi del gruppo 'Logic', come *'if...do'* o *'try...except'*. In questo caso il problema ha natura differente, in quanto anche solo ricreare i blocchi identici ai predefiniti presenti in UIFlow non risulta possibile: i blocchi della famiglia *'Logic'* sono di tipo *'execute'* (eseguono un'operazione e non restituiscono un valore), e necessitano di avere dei blocchi a loro volta di tipo *'execute'* come parametro (si pensi all' *'if...do'*). Purtroppo, UIFlow Block Maker consente di inserire parametri che siano etichette, numeri, stringhe o variabili. Risulta quindi impossibile inserire un comando da eseguire ('do') se la variabile booleana è vera ('if').

5 Conclusioni e sviluppi futuri

Per concludere in questo progetto è stato possibile realizzare un sistema che permettesse di salvare su Discord tutte le informazioni relative ai blocchi UIFlow utilizzati da studenti durante attività di robotica educativa con kit basati su M5StickC. In questo modo quando gli studenti utilizzano gli M5StickC per risolvere esercizi che vengono loro assegnati, si tiene traccia del modo in cui hanno interagito con il robot e con l'ambiente di programmazione; in questo modo si pone la base per un sistema di analisi delle interazioni che potrebbe condurre all'identificazione di pattern definiti di problem solving.

I risultati mostrano interessanti applicazioni per l'automatizzazione della raccolta dati; tuttavia, il sistema sviluppato presenta delle limitazioni che possono essere superate. Infatti, si potrà sicuramente diminuire la frequenza di messaggi persi ogni esecuzione, così come si potrà trovare una soluzione che permetta di implementare i blocchi che non è stato possibile creare.

Riferimenti bibliografici

[1] Scaradozzi D., Screpanti L., Cesaretti L. (2019) Towards a Definition of Educational Robotics: A Classification of Tools, Experiences and Assessments. In: Daniela L. (eds) Smart Learning with Educational Robotics. Springer, Cham. https://doi.org/10.1007/978-3-030-19913-5_3

[2] M5StickC, *M5 stick C* https://m5stack.hackster.io/products/m5stickc-esp32-pico-mini-iot-development-board

[3] UIFlow, *info e download UIFlow* https://m5stack.github.io/UIFlow_doc/en/

[4] UIFlow block maker,*M5BlockMaker* http://block-maker.m5stack.com/

[5] Blockly,*Pagina Wikipedia su Blockly* https://en.wikipedia.org/wiki/Blockly

[6] Discord, *Discord website* https://discord.com/

[7] Joe comp *Telegram vs discord: quale è meglio per te* https://it.joecomp.com/telegram-vs-discord-which-is-better

[8] Visual Studio Code, *Visual Studio Code download website* https://code.visualstudio.com/

[9]