AlphaBot a comando wireless

Nome: Arianna Dutto

Classe: 5^A ROB

Descrizione degli obiettivi da raggiungere:

1. flashare una sim e configurarla con i dati necessari al progetto
2. connettere l'alphabot al proprio PC
3. testare e personalizzare implementando un client/server la libreria base fornita dal professore
4. imparare la gestione dei sensori ad infrarossi e ad ostacoli
5. imparare la gestione della libreria pyinput
6. gestire il movimento dell'alphabot tramite l'implementazione di un database

Prerequisiti	- balenaEtcher - python - SQLite - PuTTY - WinSCP - Socket - libreria time - Accesso all'interfaccia web di amministrazione del router
Conoscenze	- cosa sono, come funzionano e a cosa servono i sensori ad infrarossi - cosa sono, come funzionano e a cosa servono i sensori ad ostacoli - utilitá e contenuto della libreria pynput - a cosa serve la libreria SQLite - come funzionano i motori - come si calcola l'avanzamento dell'alphabot (a giri di ruote e non a tempo) - funzionamento della libreria alphabot
Abilità	- gestione dei sensori ad infrarossi - gestione dei sensori ad ostacoli - gestione della libreria pyinput - gestione della libreria SQLite - gestione dei motori - applicazione della libreria alphabot
Competenze	- fare una relazione - comandare un alphabot attraverso applicazione client/server - gestire un database in python - gestione dei comandi alphabot in modalitá controller senza necessitá di un continuo invio

Abstract

Questa relazione descrive il percorso svolto a scuola per l'acquisizione di competenze avanzate nella gestione di un alphabot. L'attività è partita dalle basi, come la

configurazione di una scheda SIM e l'utilizzo della libreria alphabot, per arrivare allo sviluppo di competenze complete orientate alla realizzazione di:

- una comunicazione client/server per il controllo dell'alphabot in python;
- un sistema di gestione dell'alphabot tramite database SQLite.

L'intero progetto è stato svolto in 11 ore di lezione, durante le quali la classe, organizzata in coppie, ha lavorato per sperimentare e applicare tutte le fasi del percorso.

Dettaglio delle fasi operative: come è stato organizzato il lavoro nelle diverse lezioni. Riportare anche frammenti di codice commentati

lezione 1:

- dopo esserci procurate una sim, un alphabot e un computer, abbiamo fleshato la sim tramite l'ausilio dell'applicazione balenaEtcher
- grazie all'utilizzo di una tastiera esterna raspberry, collegata ad un monitor abbiamo configurato la sim con le caratteristiche necessarie alla inizializzazione dell'alphabot:
  1. nome dell'alphabot
  2. connessione al wifi
  3. inizzializzazioni interfacce
- successivamente abbiamo proceduto all'accensione dell'alphabot e del router
- grazie all'applicazione web 'archer' abbiamo potuto verificare che la sim fosse configurata correttamente

lezione 2:

- abbiamo iniziato adattando il codice fornito dal professore, il quale presentava già le funzioni avanti, indietro, destra e sinistra, aggiungendo le funzioni di rotazione destra di 90 gradi e sinistra di 90 gradi (Il tempo di rotazione è stato deciso dopo diversi tentativi di sperimentazione).
  Esempio della funzione 90 gradi a sinistra:

```
def l90(self):
        GPIO.output(self.IN1,GPIO.LOW)
        GPIO.output(self.IN2,GPIO.LOW)
        GPIO.output(self.IN3,GPIO.LOW)
        GPIO.output(self.IN4,GPIO.HIGH)
        time.sleep(0.45)
        self.stop()
```

- successivamente abbiamo implementato una gestione per il controllo di tipo client/server
- terminato il codice, abbiamo aperto il sito web 'archer' sul pc per visualizzare l'ip dell'alphabot, riconoscibile dal nome dell'alphabot deciso inizialmente
- in seguito abbiamo aperto l'applicazione PuTTY ed avviato la connessione computer-alphabot scrivendo "pi" e inserendo la password "raspberry"
- fatto ció abbiamo fatto l'accesso all'applicazione 'WinSCP', inserendo l'ip e il nome dell'alphabot, fatto ció abbiamo stabilito una connessione e abbiamo proceduto con lo spostamento dei file creati in precedenza (client, server, libreria alphabot) dal pc all'alphabot
- Infine abbiamo testato il lavoro svolto comandando il client dal pc e avviando il server sull'alphabot tramite l'applicazione PuTTY

lezione 3 e 4:

- Abbiamo iniziato regolando i sensori in modo tale da poter rilevare un'ostacolo alla distanza da noi preferita
- successivamente abbiamo implementato il vecchio codice aggiungendo la gestione dei sensori permettendo cosí l'arresto del robot in caso di rilevamento

di un ostacolo mentre l'alphabot va avanti. Esempio di codice:

```
def forward(self):
        self.controllo = True
        //settaggio motori
        while self.controllo:
            self.DR_status = GPIO.input(self.DR)
            time.sleep(0.1)
            self.DL_status = GPIO.input(self.DL)
            time.sleep(0.1)
            if self.DR_status == 0 or self.DL_status == 0:
                self.controllo = False
                self.stop()
                time.sleep(500)
```

- Infine abbiamo testato il lavoro svolto comandando il client dal pc e avviando il server sull'alphabot tramite l'applicazione PuTTY. Abbiamo riscontrato alcuni problemi che peró sono stati prontamente risolti.

lezione 5:

- Abbiamo iniziato implementando il codice sviluppato precedentemente, aggiornando la gestione del movimento del robot, sostituendola con il controllo WASD. Da qui in avanti l'alphabot si muoverà in base al tasto premuto e si fermerà quando il tasto non è più premuto a differenza di prima che si muoveva all'infinito una volta che si dava invio ad un comando a meno che non si inviava il comando stop
- dato che il nostro srver é questo:

```
        if comando == "av":
            Ab.forward()
        elif comando == "in":
            Ab.backward()
        elif comando == "si":
            Ab.left()
        elif comando == "de":
            Ab.right()
        elif comando == "s":
            Ab.stop()
        elif comando == "90r":
            Ab.r90()
        elif comando == "90l":
            Ab.l90()
        elif comando == "esci":
            print("Chiusura server...")
            break
        else:
             print("Comando non riconosciuto.")
```

- allora abbiamo creato questo:

```
def on_press(key):
```

```
        print(key)
        if key.char == 'w':
            invia('av')
        elif key.char == 'd':
            invia('de')
        elif key.char == 'a':
            invia('si')
        elif key.char == 's':
            invia('in')
    def on_release(key):
        print(f"rel = {key}")
        if key.char == 'a' or key.char == 's' or key.char == 'd' or
key.char == 'w':
            invia('s')
```

- Infine abbiamo testato il lavoro svolto comandando il client dal pc e avviando il server sull'alphabot tramite l'applicazione PuTTY.

lezione 6:

- Abbiamo iniziato sviluppando un database in SQLite3 contenente tutti i comandi necessari per il movimento dell'alphabot
- successivamente abbiamo sviluppato un codice che gestisca il database e permetta il movimento del robot in base all'input inserito dall'utente
- Infine abbiamo testato il lavoro svolto comandando il client dal pc e avviando il server sull'alphabot tramite l'applicazione PuTTY.

Risultati raggiunti e autovalutazione

il progetto è stato completato nella sua pienezza, senza tralasciare nessun punto di quelli forniti dal professore. Nel complesso siamo molto soddisfatte del risultato raggiunto e della nostra capacitá nel risolvere le criticitá

Criticità riscontrate e modalità di gestione delle criticità

sensori: inizialmente i sensori rilevavano costantemente un ostacolo, per questo motivo, sotto consiglio del professore, è stato necessario regolare l'altezza del sensore, cosí che non rilevasse come ostacolo la base stessa. Un secondo problema è che quanto l'alphabot era esposto ai raggi solari, rilevava sempre un ostacolo per via della luce. Per questo motivo è stato necessario svolgere tutti i test all'ombra.
collegamento: nella seconda lezione abbiamo avuto dei problemi a collegare l'alphabot al pc, in quanto non riuscivamo a individuarlo nel sito web archer. Dopo diversi tentativi e controlli sulla sim, è stato rilevato che il problema era il robot in sè. Per questo motivo è stato necessario cambiare alphabot.

Prospettive di sviluppo future

Durante questo progetto abbiamo raggiunto un grande obiettivo, ma il lavoro non è ancora terminato. Nelle prossime lezioni si dovrá finire l'implementazione di un database ricco di comandi in modo tale da poter comandare il nostro alphabot con funzioni anche piú complesse e terminato ció grazie all'aiuto dei professori proveremo a implementare il framework Flask cosí da poter implementare un'applicazione web in grado di comandare l'alphabot come un vero e proprio controller.