ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

SCHOOL OF ENGINEERING

Department of
Computer Science and Engineering
DISI

Master's Degree in Computer Science Engineering

Project Work
on
Digital Systems

Wheelly The Robot Coyote

Authors:
Caterina Leonelli
Anna Vandi

Professor:
prof. Stefano Mattoccia
prof. Matteo Poggi

Academic Year
2022–2023

Contents

Abstract

Lo sviluppo di robot mobili rappresenta oggigiorno un settore multidisciplinare e con molte applicazioni in campo industriale e di ricerca scientifica. Questo progetto ha lo scopo di presentare Wheelly, un robot mobile a quattro ruote controllato da Raspberry Pi 4B. Wheelly unisce la potenza dell'object detection all'efficacia dei dispositivi embedded. Grazie ad una rete neurale basata su Tensorflow Lite ed una camera, Wheelly è in grado di individuare una figura umana e di seguirla, evitando gli ostacoli con l'ausilio di un sensore ad ultrasuoni.

Introduzione

Motivazioni

Le motivazioni che ci hanno spinto alla realizzazione del progetto *"Wheelly The Robot Coyote"* sono state principalmente legate alla volontà di acquisire conoscenze e competenze circa l'utilizzo del Raspberry Pi e dei dispositivi embedded. Realizzare un progetto di questo tipo ha permesso di approfondire le conoscenze in questi ambiti e di acquisire una maggiore dimestichezza nell'utilizzo di strumenti software e hardware dedicati. In particolare, l'utilizzo di Tensorflow Lite e OpenCV per la realizzazione dell'object detection ha rappresentato un'occasione per approfondire le conoscenze in ambito di machine learning e computer vision, imparando ad utilizzare le reti neurali anche su piattaforme embedded.

Object Detection

L'object detection consiste nella capacità di individuare la presenza e la posizione di oggetti specifici all'interno di un'immagine di un video, anche più di uno ed anche se sovrapposti.
Grazie all'object detection, spesso viene segnalata la posizione dell'oggetto tramite una *bounding box*, ovvero un rettangolo che circonda l'oggetto. Le tecnologie di object detection utilizzano solitamente tecniche di deep learning come le reti neurali convoluzionali (CNN).

Software

Tensorflow Lite

Ci sono molti vantaggi nell'utilizzo di TensorFlow Lite, un insieme di strumenti che consente l'utilizzo dell'apprendimento automatico su dispositivi mobili ed embedded. Grazie a questi strumenti, si possono eseguire modelli di machine learning su dispositivi con limitazioni di memoria e consumo di energia.

TensorFlow Lite supporta anche molte piattaforme tra cui Android e iOS, il che lo rende una scelta popolare per gli sviluppatori di applicazioni mobili.

TensorFlow Lite offre una serie di strumenti per convertire modelli di apprendimento automatico di TensorFlow in una versione compatibile con TensorFlow Lite. Ciò significa che gli sviluppatori possono importare
facilmente modelli già addestrati sui propri dispositivi mobili senza doverli creare da zero.

OpenCV

OpenCV (Open Source Computer Vision) è una libreria open source di elaborazione di immagini e computer vision, disponibile per diversi sistemi operativi tra cui Linux, Windows, macOS, Android e iOS.

OpenCV è scritta principalmente in C++, ma ha anche interfacce per Python, Java e MATLAB, tra gli altri. La libreria offre molte funzionalità per l'elaborazione delle immagini, come il rilevamento dei bordi, la correzione della distorsione, la segmentazione e la ricostruzione 3D. Inoltre, OpenCV include anche algoritmi di computer vision per il rilevamento dei volti, il tracciamento e la classificazione degli oggetti.

OpenCV può essere utilizzato per l'object detection, ovvero l'identificazione e la localizzazione degli oggetti all'interno di un'immagine o di un video. L'object detection viene spesso utilizzato in applicazioni di sorveglianza, monitoraggio del traffico, rilevamento dei volti e molto altro.

Per utilizzare OpenCV per l'object detection, è necessario prima definire il modello che deve essere utilizzato per il rilevamento degli oggetti. Questo può essere fatto addestrando un classificatore di immagini con un dataset di immagini. Ci sono anche modelli pre-addestrati disponibili online, che possono essere scaricati e utilizzati direttamente con OpenCV.

Una volta che il modello è pronto, OpenCV viene utilizzato per caricare un'immagine o un video e passare ogni frame attraverso il modello di rilevamento degli oggetti. Il modello identificherà gli oggetti nell'immagine o nel video, restituendo le coordinate della bounding box che circonda l'oggetto.

Queste coordinate possono quindi essere utilizzate per disegnare la bounding box sull'immagine o sul video in modo da evidenziare gli oggetti identificati. È anche possibile utilizzare queste coordinate per eseguire ulteriori analisi sugli oggetti identificati, ad esempio per determinare la loro posizione o per contare il numero di oggetti identificati in un determinato intervallo di tempo.

Hardware

Raspberry Pi 4B

Il Raspberry Pi 4B è l'ultima versione corrente del mini-computer Raspberry Pi prodotto dalla Raspberry Pi Foundation. Questo modello è stato rilasciato nel giugno del 2019 ed è stato introdotto per sostituire il precedente Raspberry Pi 3B+.

Il Raspberry Pi 4B presenta numerose caratteristiche e miglioramenti rispetto al modello precedente, tra cui:

1. Processore: il Raspberry Pi 4B è dotato di un processore quad-core ARM Cortex-A72 a 1,5 GHz, che offre prestazioni significativamente migliori rispetto al processore quad-core ARM Cortex-A53 a 1,4 GHz del Raspberry Pi 3B+.
2. RAM: il Raspberry Pi 4B è disponibile con diverse opzioni di RAM, tra cui 2 GB, 4 GB o 8 GB di memoria. Per questo progetto è stato untilizzato un Raspberry Pi con 4GB di RAM.
3. Interfacce USB: il Raspberry Pi 4B presenta due porte USB 2.0 e due porte USB 3.0, che consentono di collegare una vasta gamma di dispositivi periferici, come tastiere, mouse, dischi rigidi esterni e altro ancora.
4. Interfacce video: il Raspberry Pi 4B dispone di due porte micro HDMI, che supportano la riproduzione video.
5. Connettività: il Raspberry Pi 4B è dotato di connettività Ethernet Gigabit, Wi-Fi e Bluetooth, che consentono di connettersi a reti locali e ad altri dispositivi.
6. GPIO: il Raspberry Pi 4B presenta 40 pin GPIO (General Purpose Input/Output), che possono essere utilizzati per collegare sensori, motori e altri dispositivi elettronici.
7. Camera: il Raspberry Pi 4B supporta la connessione di una vasta gamma di fotocamere compatibili attraverso la porta camera, che consente di acquisire immagini e video di alta qualità.

Figure 1: Raspberry Pi4B

Hardware Interfaces

Il Raspberry Pi dispone di diverse interfacce hardware per supportare fotocamere e sensori. In generale, per l'object detection, le interfacce più comuni sono GPIO, CSI e USB, mentre per l'utilizzo di sensori, le interfacce più comuni sono GPIO, I2C e UART.

1. **GPIO (General Purpose Input/Output):** è una delle interfacce hardware principali per il Raspberry Pi. Si tratta di un'interfaccia a basso livello che consente al processore del Raspberry Pi di comunicare con il mondo esterno attraverso segnali elettrici. Le porte GPIO possono essere configurate come input o output e utilizzate per leggere i dati dei sensori, come ad esempio un sensore ad infrarossi, o per controllare attuatori come ad esempio un LED. La Raspberry Pi dispone di diversi pin GPIO, che possono essere utilizzati per vari scopi a seconda delle esigenze. Per programmare i GPIO, è necessario utilizzare un linguaggio di programmazione come Python o C, che consente di accedere ai registri del processore e di controllare il valore dei pin GPIO. Esistono anche librerie e framework come RPi.GPIO per Python che semplificano la programmazione dei GPIO al fine di poter interagire con i pin in modo più semplice.

2. **Interfaccia I2C (Inter-Integrated Circuit):** è un'interfaccia seriale a due fili utilizzata per la comunicazione con i dispositivi di sensoristica, come ad esempio i sensori di distanza ad ultrasuoni.

3. **UART (Universal Asynchronous Receiver/Transmitter):** è un'interfaccia seriale asincrona utilizzata per la comunicazione con altri dispositivi come moduli GPS o sensori di movimento.

4. **USB (Universal Serial Bus):** è un'interfaccia comune utilizzata per collegare dispositivi esterni, come ad esempio fotocamere USB, microfoni o dispositivi di acquisizione video.

5. **Camera Serial Interface (CSI):** è un'interfaccia hardware presente sulla scheda Raspberry Pi che consente di collegare direttamente una fotocamera compatibile senza la necessità di utilizzare un adattatore o un connettore esterno. Questa interfaccia presenta un ingresso per un cavo flat e supporta diverse fotocamere ad alta risoluzione. L'utilizzo della porta camera consente di trasferire i dati immagine dal sensore della fotocamera direttamente al processore del Raspberry Pi, semplificando notevolmente il collegamento e l'uso della camera.

Camera

La *Raspberry Pi Camera* è una fotocamera a basso costo progettata appositamente per essere utilizzata con il Raspberry Pi.

Ci sono anche altre opzioni per utilizzare una fotocamera con il Raspberry Pi. Ad esempio, *DroidCam* è un'applicazione che consente di utilizzare la fotocamera di uno smartphone come fotocamera USB per il Raspberry Pi. In alternativa, è possibile utilizzare una *fotocamera USB standard*.

1. **Raspberry Pi Camera v1.3**: attualmente esistono 3 versioni della Raspberry Pi Camera: la v1.3 e la v2.1 e la v3, tutte compatibili con il Raspberry Pi 4B, anche se a partire dal 2023 la versione 1.3 non è più supportata.
   La fotocamera v1.3 ha una risoluzione di 5 megapixel e può catturare immagini fisse e video a 1080p, la fotocamera v2.1 ha una risoluzione di 8 megapixel e può catturare immagini fisse e video a 1080p o 720p, mentre la fotocamera v3 ha una risoluzione di

megapixel e può catturare immagini fisse e video a 1080p50, 720p100 o 480p120.
La *Raspberry Pi Camera v1.3* è dotata di un obiettivo fisso con una lunghezza focale di 3,6 mm e un angolo di campo di circa 54 gradi.
Tutte le versioni utilizzano un cavo flessibile come connettore al Raspberry Pi. In questo modo, è possibile utilizzare le librerie di programmazione disponibili, come OpenCV, per acquisire e elaborare le immagini catturate dalla fotocamera.

2. **DroidCam**: DroidCam è un'applicazione che permette di collegare la fotocamera di un dispositivo mobile (Android/iOS) al computer (Windows/Linux) sia tramite il cavo USB che in modalità wireless. Ciò consente di utilizzare lo smartphone come webcam per il Raspberry Pi.

3. **Usb camera**: In genere, le webcam USB offrono una qualità inferiore rispetto ai moduli della fotocamera che si collegano all'interfaccia CSI e non possono essere controllate tramite i comandi da terminale (raspistill e raspvid) o tramite il pacchetto Python picamera (controllo webcam in linguaggio Python). Può essere vantaggioso collegare una fotocamera USB al Raspberry Pi quando si desidera configurare più fotocamere con un singolo Raspberry Pi, a condizione che siano alimentate correttamente in base ai loro requisiti di alimentazione.

Figure 2: Camera Pi v1.3

Servo Motor

Un servomotore è un piccolo motore dotato di circuiti di controllo incorporati, in grado di ruotare fino a 180 o 360 gradi.

Esso viene controllato accendendo e spegnendo uno dei pin GPIO. La durata degli impulsi (ampiezza) controlla la direzione indicata dal servomotore.

Questi segnali vengono chiamati PWM (*Pulse Width Modulation*). Il Raspberry Pi non supporta la generazione di tali segnali PWM di default, poichè non dispone di un sistema di clock dedicato per farlo.
In questo progetto, verranno utilizzati segnali PWM generati dal software.
In pratica, il processore del Raspberry Pi genera impulsi elettrici attraverso i pin GPIO che alternano il loro stato tra ON e OFF a frequenze molto elevate, al fine di simulare l'effetto dei segnali PWM. Questo metodo ha il vantaggio di utilizzare solo i pin GPIO del Raspberry Pi, senza dover aggiungere componenti hardware esterni. Tuttavia, i segnali generati non saranno perfetti e questo potrebbe portare a piccole ed indesiderate oscillazioni del servomotore (effetto *jitter*). Per limitare questo effetto, è stata utilizzata la libreria `pigpio`.
Un servomotore viene collegato tramite 3 cavi. Nel caso specifico di un Micro Servo:

- cavo di terra o ground (di solito di colore marrone/nero)
- cavo di alimentazione positiva Vcc (di solito di colore rosso)
- cavo pilot usato per inviare segnali di controllo, ad esempio il segnale di controllo della modulazione dell'ampiezza dell'impulso (di solito di colore giallo/arancione).

Tramite l'ausilio di jumper wire maschio-femmina è possibile collegare i pin femmina del servomotore ai pin GPIO del Raspberry Pi.

Figure 3: Servo Motor

Sonar

Il sonar (sensore ad ultrasuoni) è un dispositivo ad ultrasuoni che consente di misurare la distanza tra un oggetto che ha davanti a sè e il sensore stesso, utilizzando segnali ad ultrasuoni. Esso è composto da un trasmettitore ad ultrasuoni, un ricevitore ad ultrasuoni ed un circuito di controllo.

Il funzionamento del sonar HC-SR04 si basa sulla trasmissione di un impulso ad ultrasuoni dal trasmettitore, che viene poi riflesso sulla superficie dell'oggetto e ricevuto dal ricevitore. La distanza tra il dispositivo e l'oggetto viene quindi calcolata misurando il tempo di volo dell'impulso ad ultrasuoni.

Un sonar può essere collegato a un Raspberry Pi utilizzando uno dei suoi pin GPIO.

Per collegare il sonar, è necessario connettere i suoi 4 pin ai pin GPIO del Raspberry Pi:

- Vcc: collegare a 5V
- Gnd: collegare a Gnd
- Trig: collegare a un pin GPIO
- Echo: collegare a un altro pin GPIO

Una volta collegato il sonar, è possibile utilizzare un programma Python per controllarlo e leggere le misurazioni di distanza.

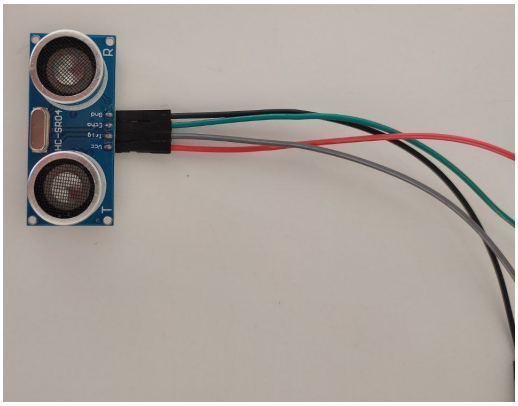

Figure 4: Sonar HC-SR04

Il segnale di uscita del sensore (ECHO) sull'HC-SR04 è valutato a 5V. Tuttavia, il pin di ingresso sul GPIO Raspberry Pi è valutato a 3,3 V. L'invio di un segnale da 5 V a quella porta di ingresso da 3,3 V non protetta potrebbe danneggiare i pin GPIO, dunque avremo bisogno di

utilizzare un piccolo circuito divisore di tensione, costituito da due resistori, per abbassare la tensione di uscita del sensore a qualcosa che il nostro Raspberry Pi può gestire.

H-Bridge

Per pilotare dei motori elettrici in corrente continua, è possibile utilizzare un **L298N Dual H-Bridge Motor Controller**.
Al suo interno troviamo due ponti H integrati, che possono essere controllati da segnali in logica TTL (*Transistor-Transistor Logic*). Questo tipo di segnali si basa su una tensione di alimentazione di 5 V e presenta due livelli di tensione: uno alto, di circa 5 V, e uno basso, di circa 0 V. Infatti, il Raspberry Pi invia i segnali in logica TTL ai piedini del ponte H al fine di controllare la *direzione* e la *velocità* dei motori.
In particolare:

- Il segnale alto inviato al piedino INx mette l'uscita OUTx ad alto.
- Per controllare la direzione, vengono utilizzati due dei quattro ingressi del ponte H:
  1. Il segnale inviato al piedino IN1 e controlla la direzione del primo motore (se è 1 il motore gira in una direzione, se è 0 nell'altra),
  2. Il segnale inviato al piedino IN2 controlla la direzione del secondo motore.
- Per controllare la velocità, invece, si utilizzano i due piedini di enable del ponte H (EN1 e EN2):
  - Il segnale TTL inviato su questi piedini modula l'ampiezza della corrente che fluisce nel motore, e quindi la sua velocità.

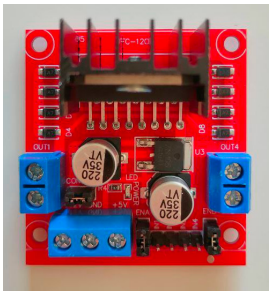

Figure 5: L298N Dual H-Bridge Motor Controller

In questo progetto, ci occuperemo soltanto del controllo della direzione e non della velocità.

Motors

Un motoriduttore DC è un tipo di motore elettrico in corrente continua che viene utilizzato per generare un movimento rotatorio.

È composto da un motore elettrico e da un riduttore meccanico. Quest'ultimo riduce la velocità di rotazione del motore, aumentando al contempo la coppia disponibile. Grazie all'installazione di un riduttore di velocità, il motore sarà in grado di fornire una maggiore forza rotazionale, a discapito di una riduzione della velocità.

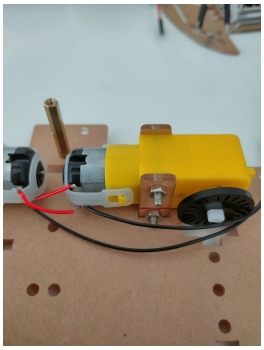

Figure 6: Motoriduttore

Wiring

La figura proposta di seguito rappresenta il cablaggio, la disposizione delle periferiche e le connessioni tra esse.

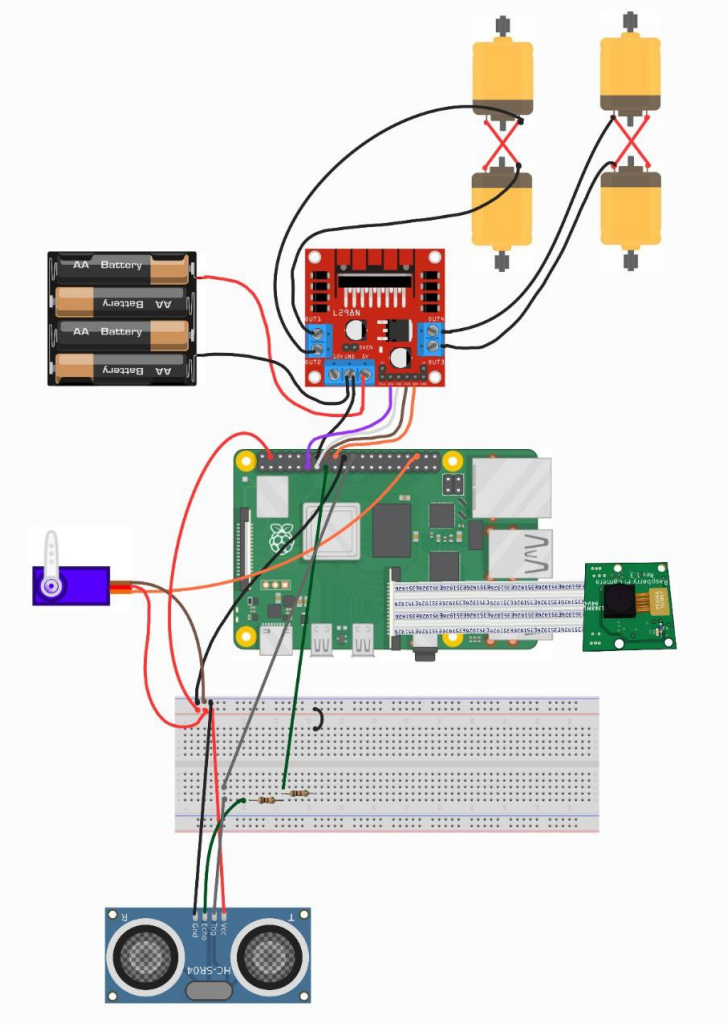

Figure 7: Circuito

Ponte H - Motoriduttori

È necessario collegare il ponte H ai 4 motori, al fine di garantire il controllo da parte del Raspberry Pi.

Entrambi i motori del lato sinistro (e analogamente del destro) dovranno ricevere lo stesso segnale (o alto o basso) nello stesso momento, al fine di permettere alle ruote di girare nello stesso verso. Essendo i motori posizionati "al contrario", è necessario collegare il pin superiore del motore anteriore con il pin inferiore del motore posteriore tra loro, e fare lo stesso con il pin inferiore del motore anteriore e quello superiore del motore posteriore. I motori sul lato destro saranno collegati ai pin Out3 e Out4 del Ponte-H. Analogamente, i motori di sinistra (sia anteriore che posteriore) verranno collegati ai pin Out1 e Out2 del Ponte-H attraverso la medesima tecnica.

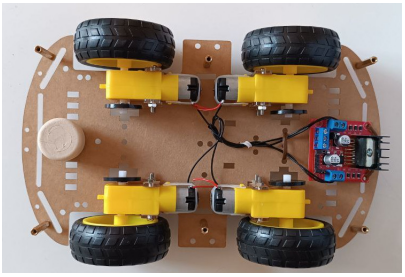

Figure 8: Wiring tra Ponte-H e Motoriduttori

Raspberry PI - Ponte H

È necessario collegare il ponte H al Raspberry Pi per consentire il controllo dei 4 motori. Utilizzando opportuni cavi, i pin del Raspberry Pi sono stati collegati agli opportuni pin del ponte-H come descritto in seguito. È necessario collegare i cavi ai pin del ponte H come segue. *GPIO*:

- GPIO-17 (cavo viola) → pin IN1 del ponte-H
- GPIO-27 (cavo bianco) → pin IN2 del ponte-H
- GPIO-23 (cavo marrone)→ pin IN3 del ponte-H
- GPIO-24 (cavo arancione) → pin IN4 del ponte-H
- Ground (cavo nero) → gnd del ponte-H

Batterie:

- Batteria Vcc (cavo rosso) → +5 Volt Ponte-H
- Batteria Gnd (cavo nero) → Gnd Ponte-H

Sonar - Raspberry PI

Il sensore a ultrasuoni HC-SR04 ha quattro pin: *terra* (**GND**), *Echo Pulse Output* (**ECHO**), *Trigger Pulse Input* (**TRIG**) e 5V *Supply* (**Vcc**). L'alimentazione avviene per mezzo di Vcc, il collegamento a terra utilizzando GND e verrà sfruttato il Raspberry Pi per inviare un segnale di ingresso a TRIG. Grazie a tale segnale di input, il sensore potrà inviare un impulso. Le onde prodotte, vengono in seguito riflesse al sensore e rilevate grazie alla ricezione di un segnale a 5 V sul pin ECHO. Tuttavia, il pin di ingresso sul GPIO Raspberry Pi è valutato a 3,3 V, pertanto,

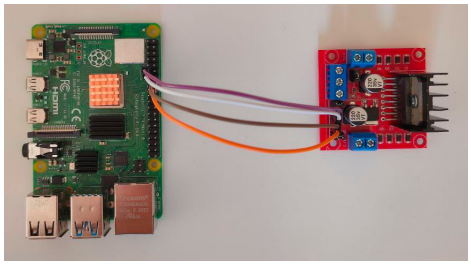

Figure 9: Wiring tra Ponte-H e Raspberry PI

l'invio di un segnale da 5 V ad una porta di ingresso da 3,3 V non protetta adeguatamente potrebbe danneggiare i pin GPIO.
Questo rischio può essere evitato introducendo un piccolo circuito divisore di tensione, costituito da due resistori, al fine di abbassare la tensione di uscita del sensore per renderla gestibile da parte del Raspberry Pi. È diventato quindi necessario introdurre non solo una resistenza, ma anche una Breadboard. La Breadboard srebbe necessaria soltanto per il pin di echo, ma per comodità l'abbiamo utilizzata per tutti i pin.

Gnd (cavo nero):

1. Gnd → Guida Negativa Breadbord
2. Guida Negativa → Gnd RPI

Echo (cavo Verde):

1. Echo → Guida Vuota x Breadbord
2. *Resistenza R1 (1k):*
   - Guida Vuota x ⟷ Guida Vuota y
3. *Resistenza R2 (2k):*
   - Guida Vuota y ⟷ Guida Vuota z
4. Guida Vuota z ⟷ Guida Gnd
5. Guida Vuota y → GPIO-22 RPI

Trig (cavo Grigio):

1. Trig → Guida Vuota Breadbord
2. Guida Vuota → GPIO-25 RPI

Vcc (cavo rosso):

1. Vcc → Guida Positiva Breadboard
2. Guida Positiva → +5V RPI

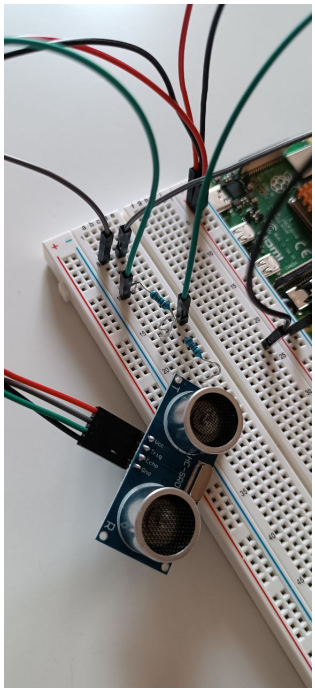

Figure 10: Disposizione dei resistori e del Pin Echo del Sonar sulla Breadboard

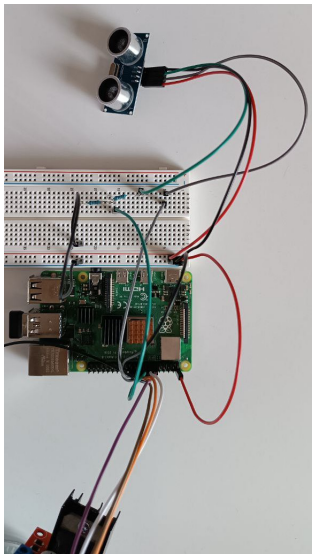

Figure 11: Wiring tra Sonar e Raspberry PI

Servo - Raspberry PI

Come accennato in precedenza, il servomotore possiede un cavo arancione che viene utilizzato per trasmettere il segnale PWM. Il cavo rosso verrà collegato a 5 volt e il nero a Gnd. Nel progetto in questone abbiamo deciso di alimentare il nostro servomotore utilizzando la breadboard e quindi usando il voltaggio dato dal Raspberry Pi. Questo funziona grazie alle piccole dimensioni del servomotore; ma su un modello più grande sarebbe necessario un apposito alimentatore.

- Pilot (cavo arancione) → Guida Vuota → GPIO-16 RPI
- Ground (cavo marrone) → Guida Negativa (Già collegata a Gnd RPI)
- Vcc (cavo rosso) → Guida Positiva (Già collegata a 5V RPI)

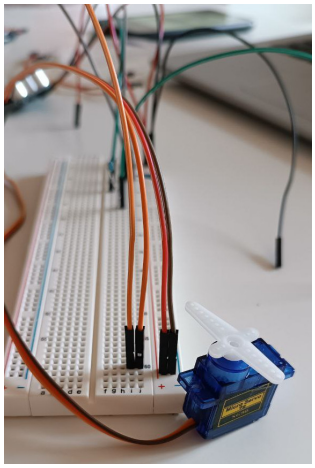

Figure 12: Wiring tra Servo e Raspberry PI

Struttura

In questo capitolo si descrive come è stato costruito il robot Wheelly, con un focus sulle sue componenti fisiche relative al movimento e all'alimentazione. Un robot viene definito come sistema autonomo in grado di percepire l'ambiente circostante ed influenzarlo per raggiungere i suoi obiettivi. Wheelly è composto da un chassis, sensori (sonar e fotocamera) e attuatori (ruote, servo e motori), oltre a un powerbank per l'autonomia. La parte elettronica include il Raspberry PI 4B, i 4 motori in corrente continua, un Ponte-H L298N e quattro batterie AA da 1,5V ciascuna.

Forma e Struttura

Il Raspberry PI permette di controllare i motori in base ai segnali provenienti dai sensori, con il supporto del servo motore. Durante la costruzione, si sono riscontrati problemi legati al peso eccessivo della struttura, il che ha evidenziato la necessità di sostituire la fonte di alimentazione con una più leggera. Inoltre, è stato necessario dotare il Raspberry PI di una *ventola* per evitare il surriscaldamento causato dai programmi di Machine Learning che impegano una computazione onerosa.

Figure 13: Struttura esterna del robot, con Case del Raspberry PI dotato di ventola

La struttura del robot utilizza una tecnica a piani, che consente di aggiungere nuovi componenti in modo semplice, mantenendo le parti già costruite. L'aggiunta di nuovi componenti avviene mediante l'aggiunta di strati che seguono la forma degli strati precedenti, mantenendo la larghezza del robot invariata. L'altezza può invece variare senza compromettere le funzionalità del robot, che è progettato per operare su superfici piane. Come piattaforme sono stati utilizzati due trolley chassis in vetro temperato.

Assemblaggio

Il primo livello contiene lo spazio per la powerbank del Raspberry Pi, posta al centro per evitare che il peso sbilanci il robot, e gli slot necessari per i motori e le ruote. Questo livello contiene

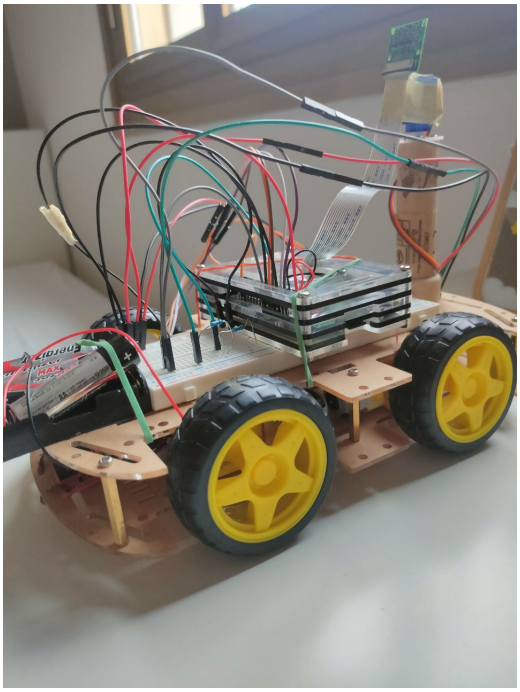

Figure 14: Struttura verticale del robot con chassis

anche il dispositivo sonar, i cui cavi passano attraverso opportuni fori nello chassis e il ponte H.

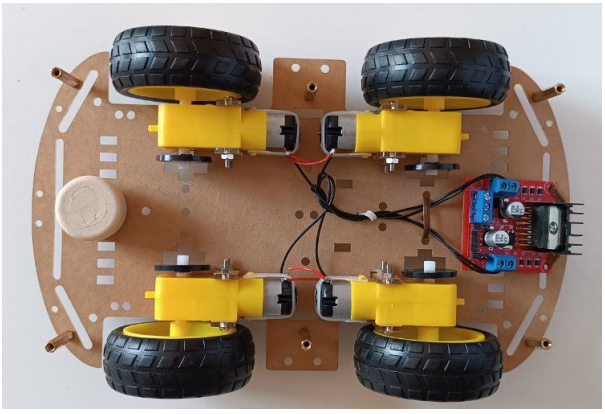

Figure 15: Primo livello della struttura

Il secondo livello ospita le batterie del motore, di modo che siano facilmente accessibili nel caso si dovesse cambiare una batteria, il Raspberry PI all'interno di un case con ventola e la breadboard. I motori sono collegati al Ponte-H tramite passaggi nello chassis. Questo livello contiene anche la fotocamera e il servo motore, opportunamente sollevati di modo da poter far inquadrare meglio la figura umana.

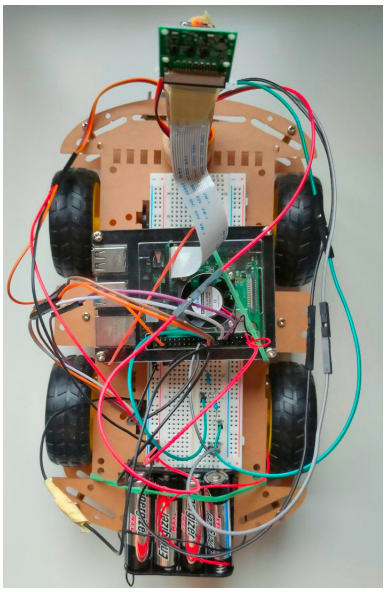

Figure 16: Secondo livello della struttura

Bill of Materials

I componenti necessari per la realizzazione del progetto sono:

- Raspberry PI 4B
- 2 Trolley Chassis
- 4 Motoriduttori DC
- 4 ruote
- L298N Dual H-Bridge Motor Controller
- Sonar (L298N Dual H-Bridge Motor Controller)
- Jumper wires maschio-machio / maschio-femmina / femmina-femmina
- Breadboard
- Cavo USB-C
- Ventola di raffreddamento
- Alimentatore 5V/3A per Raspberry Pi 4B
- Power Bank leggera
- 4 batterie AA di 1,5V
- Camera Pi module v1.3 da 5 Megapixel con cavo flat
- Servo Motor Control

Implementazione

Repository

Il progetto è disponibile per la consultazione:
`https://github.com/RootLeo00/robot-wheelly-object-detection`

Source Files

Esaminiamo di seguito i dettagli principali del codice.

Movimento: direzione

Nella classe `MotorWheels()` vengono definiti i metodi necessari per interagire con i pin GPIO del Raspberry Pi, al fine di controllare la direzione di rotazione. Nel metodo *init*, vengono inizializzati i pin GPIO come output.

- Il metodo `forward` fa ruotare entrambi i motori in avanti per un certo numero di secondi.
- I metodi `forwardleft` e `forwardright` fanno ruotare solo uno dei due motori in avanti, in modo da girare la macchina verso sinistra o destra rispettivamente.
- Analogamente per i metodi `backwardleft` e `backwardright`che fanno ruotare solo uno dei due motori all'indietro.
- `backward` fa ruotare entrambi i motori all'indietro.
- `stop` ferma entrambi i motori

```
RPi.GPIO as gpio
import time

class MotorWheels():
    def __init__(self):
        gpio.setmode(gpio.BCM)
        gpio.setup(17, gpio.OUT)
        gpio.setup(27, gpio.OUT)
        gpio.setup(23, gpio.OUT)
        gpio.setup(24, gpio.OUT)

    def forward(self, sec):
        gpio.output(17, True)
        gpio.output(27, False)
        gpio.output(23, True)
        gpio.output(24, False)
        time.sleep(sec)

    ...
```

Listing 1: wheels.py

Sonar: distanza

La classe `Sonar()` ha lo scopo di musurare la distanza da un oggetto usando un sensore ad ultrasuoni. Il metodo *init*, inizializza i pin GPIO come output per il trigger e input per l'echo. Il trigger viene impostato su `True` e poi subito dopo su `False` per generare un impulso di durata di 10 microsecondi.
Il tempo di volo viene quindi misurato dall'eco del segnale emesso. L'istruzione `while` viene utilizzata per salvare il tempo di inizio del segnale riflettuto (l'eco viene rilevato) e il tempo di arrivo del segnale (il segnale riflesso termina).
La differenza temporale viene convertita in una distanza in centimetri.

```
import RPi.GPIO as GPIO
import time

#set GPIO Pins
GPIO_TRIGGER = 25
GPIO_ECHO = 22

class Sonar():

    def __init__(self):
        GPIO.setmode(GPIO.BCM)

        GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
        GPIO.setup(GPIO_ECHO, GPIO.IN)

    def distance(self):
        GPIO.output(GPIO_TRIGGER, True)

        # set Trigger after 0.01ms to LOW
        time.sleep(0.00001)
        GPIO.output(GPIO_TRIGGER, False)

        StartTime = time.time()
        StopTime = time.time()

        # save StartTime
        while GPIO.input(GPIO_ECHO) == 0:
            StartTime = time.time()

        # save time of arrival
        while GPIO.input(GPIO_ECHO) == 1:
            StopTime = time.time()

        # time difference between start and arrival
        TimeElapsed = StopTime - StartTime
        # multiply with the sonic speed (34300 cm/s)
        # and divide by 2, because there and back
        distance = (TimeElapsed * 34300) / 2
        #GPIO.cleanup()
        return distance
```

Listing 2: sonar.py

Servo motore

La classe `Servo()` controlla il servo motore con l'ausilio della libreria `gpiozero`. Il GPIO corrispondente viene passato come parametro al costruttore, il quale inizializza il servo nella posizione `mid()`.

- I metodi `rotate_to_left()`, `rotate_to_right()` e `rotate_to_middle()` chiamano rispettivamente

i metodi `max()`, `min()` e `mid()` dell'oggetto servo della libreria gpiozero, utilizzati per impostare il valore del PWM per muovere il servo.

- I metodi `is_rotated_left`, `is_rotated_left` e `is_rotated_middle` restituiscono un valore booleano a seconda della posizione del servo.

È stato utilizzato il modulo `gpiozero` di modo da ridurre al minimo gli effetti di jitter.

```
import time
from gpiozero import Servo as GPIOZeroServo
from gpiozero.pins.pigpio import PiGPIOFactory

#set GPIO Pins
GPIO_PILOT = 16


class Servo():

    #place servo in left position
    def rotate_to_left(self):
        self.servo.max()
        time.sleep(0.5)

    #place servo in right position
    def rotate_to_right(self):
        self.servo.min()
        time.sleep(0.5)

    #place servo in middle position
    def rotate_to_middle(self):
        self.servo.mid()
        time.sleep(0.5)

    def is_rotated_left(self):
        return self.servo.value == 1.0

    def is_rotated_right(self):
        return self.servo.value == -1.0

    def is_rotated_middle(self):
        return self.servo.value == 0.0 or self.servo.value==-0.0

    def __init__(self, gpio_number):
        # initialize servo
        factory = PiGPIOFactory()
        self.servo = GPIOZeroServo(gpio_number, pin_factory=factory)
        self.servo.mid()
```

Listing 3: servo.py

Object Dection: OpenCV e tflite

Il codice proposto utilizza `TensorFlow Lite` per rilevare la presenza di una persona in un'immagine acquisita dalla webcam per poi seguirla.

- La funzione `run()` rappresenta il cuore dell'object detection.
  - ◇ Inizializza la camera, il modello di object detection, i motori delle ruote, il sensore ad ultrasuoni e il servo per il controllo della camera.
  - ◇ Avvia un loop `while` che acquisisce continuamente immagini dalla telecamera ed esegue l'object detection su di esse, utilizzando l'opportuno modello.

- La funzione `do_detection` viene invocata dalla `run()`; l'immagine acquisita viene prima convertita in formato RGB e passata al modello. Il risultato dell'object detection è restituito come oggetto `DetectionResult`, che contiene le informazioni sulle bounding box, le categorie e i punteggi di confidenza delle classi rilevate.
- La funzione `follow_detected_object` viene invocata qualora venga rilevata una persona nell'immagine.
  - ◇ Il robot viene fatto avanzare verso la persona fino ad un massimo di 30 cm dal sonar.
  - ◇ Se il robot si avvicina troppo alla persona, si ferma per un secondo e riprende la ricerca.
  - ◇ Il servo viene utilizzato per orientare la telecamera verso la persona.

```
import
...

from tflite_support.task import core
from tflite_support.task import processor
from tflite_support.task import vision

def run(model: str, camera_id: int, width: int, height: int, num_threads: int,
    enable_edgetpu: bool) -> None:

    try:
        # start capturing video input from the camera
        cap = cv2.VideoCapture(camera_id)
        cap.set(cv2.CAP_PROP_FRAME_WIDTH, width)
        cap.set(cv2.CAP_PROP_FRAME_HEIGHT, height)
        cap_xcenter=cap.get(cv2.CAP_PROP_FRAME_WIDTH) /2


        # initialize the object detection model
        base_options = core.BaseOptions( ...)
        ...
        servo = Servo(16)  #initialize servo
        wheels = MotorWheels()  #initialize wheels
        sonar = Sonar() #initialize sonar

        #capture images from the camera and run inference
        while cap.isOpened():
            if(sonar.distance()<30): # check sonar
                wheels.stop()
                time.sleep(1)
            else:
                # move servo-webcam for random search
                if servo.is_rotated_left():
                    servo.rotate_to_right()
                elif servo.is_rotated_right():
                    servo.rotate_to_middle()
                elif servo.is_rotated_middle():
                    servo.rotate_to_left()

                # perform object detection for 5 seconds
                t_end = time.time() + 5
                already_followed=False
                while not
                already_followed and time.time() < t_end:
                detection_result = do_detection(cap, detector)

                    # move the robot if target is detected
                    for detection in detection_result.detections:
```

```
                        category = detection.categories[0]
                        if not already_followed
                        and category.category_name == "person"
                        and category.score>0.70:
                            print("OBJECT DETECTED")
                            follow_detected_object(...)
                            ...
...

def follow_detected_object(servo, cap, sonar, wheels):
    try:
        # check sonar distance
        while sonar.distance()>30:
            if servo.is_rotated_left():
                wheels.forwardleft(1)
                servo.rotate_to_middle()
            elif servo.is_rotated_right():
                wheels.forwardright(1)
                servo.rotate_to_middle()
            elif servo.is_rotated_middle():
                wheels.forward(0.05)

        # stop robot if distance is less than 30 cm
        wheels.stop()
        time.sleep(1)
        return
        ...

def do_detection(cap, detector):
       ...
    image = cv2.flip(image, 1)
    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    # Create a TensorImage object
    input_tensor = vision.TensorImage.create_from_array(rgb_image)

    # Run object detection estimation using the model.
    detection_result = detector.detect(input_tensor)
    return detection_result
...
```

Listing 4: detect.py

EfficientDet

Nel main di `detect.py` , il file `efficientdet_lite0.tflite` viene utilizzato per creare un'istanza del modello `EfficientDet Lite0` di `TensorFlow Lite`.
`EfficientDet Lite0` è un modello di Deep Learning pre-addestrato per la rilevazione di oggetti. Tale modello è stato addestrato su un grande dataset di immagini e può essere utilizzato per rilevare oggetti in nuove immagini. La versione *lite* della rete neurale `EfficientDet` è stata sviluppata appositamente per dispositivi mobili, in cui vi sono risorse limitate. Ciò consente di eseguire l'inferenza (elaborazione delle predizioni) direttamente sul dispositivo, senza la necessità di una connessione internet o di un server remoto per l'elaborazione dei dati.

Risorse e Documentazioni

Di seguito le risorse consultate:

- https://github.com/tensorflow/tpu/blob/master/models/official/efficientnet/lite/
- https://www.tensorflow.org/lite/api_docs/python/tflite_support/task
- https://pypi.org/project/RPi.GPIO/
- https://abyz.me.uk/rpi/pigpio/pigpiod.html
- https://docs.opencv.org/3.4/

Conclusioni

In conclusione, Wheelly rappresenta un esempio di come i sistemi digitali possano essere applicati al fine di progettare soluzioni che coinvolgono un approccio interdisciplinare in grado di unire meccanica, elettronica e programmazione. Grazie all'utilizzo di componenti digitali avanzati, come Raspberry Pi, Camera, Servo Motor e Sonar, è stato possibile implementare una soluzione di object detection basata su rete neurale. Possibili sviluppi futuri comprendono la possibilità di rendere più veloce la fase di object detection, e per questo si possono usare acceleratori grafici esterni (e.g. USB Coral Accelerator), board con GPU (e.g NVIDIA Jetson), board più sofisticate (FPGA). La progettazione e lo sviluppo di *"Wheelly The Robot Coyote"* è stata inoltre opportunità di apprendimento e di sperimentazione per acquisire nuove competenze in un settore in continua evoluzione come quello dei sistemi embedded e della robotica.