Università Politecnica delle Marche

FACOLTÀ DI INGEGNERIA
Corso di Laurea Triennale in Ingegneria Informatica e dell'Automazione

Tesi di laurea triennale

Analisi e integrazione di un sensore di prossimità innovativo per robot bipedi

Analysis and integration of an innovative proximity sensor for bipedal robots

Candidato:
Cecilia De Padova
Matricola 1087059

Relatore:
David Scaradozzi

Anno Accademico 2020-2021

Alla mia famiglia
che non ha mai smesso di credere in me

Indice

Elenco delle figure

Listato

1 Introduzione

L'inizio della quarta rivoluzione industriale, meglio conosciuta come industria 4.0, ha reso sempre più necessaria la collaborazione tra l'uomo e il mondo dei robot [1]. In ambito industriale i robot cominciano ad essere sempre più centrali nello sviluppo di un'azienda in quanto permettono la semplificazione e l'ottimizzazione del lavoro.

Questo progetto nasce dall'esigenza dell'uomo di relazionarsi con queste macchine in modo efficiente cercando, quanto più possibile, di realizzare un ambiente sicuro. La mancanza di sicurezza in un ambiente lavorativo ostacola in modo rilevante il processo di automatizzazione dell'industria. Negli anni sono state elaborate diverse normative che specificano i requisiti di sicurezza per l'integrazione di robot industriali, sistemi di robot industriali e celle di robot industriali.

Sviluppare un sistema di sicurezza permette di eliminare le barriere facendo sì che l'attività del robot non escluda quella dell'uomo. I robot industriali, attraverso l'implementazione di sensori per la sicurezza, riescono in autonomia a rilevare informazioni riguardo l'ambiente circostante così da poter modificare il proprio comportamento in base ai vari stimoli che ricevono.

Figura 1: Robot industriali

Il presente elaborato consiste in una ricerca per l'integrazione di un sistema di sicurezza su un robot bipede. Questo sistema viene posizionato sul retro del robot così da permettere a quest'ultimo, camminando all'indietro, di evitare urti potenzialmente pericolosi. Il fine è quello di favorire l'interazione tra uomo e robot in un ambiente condiviso sicuro.

La tesi è articolata in sei sezioni: nella prima sezione si va ad introdurre il problema che ha portato allo sviluppo di questo studio, nella seconda ci si concentra nell'analisi della strumentazione hardware e software utilizzata. Nella

terza sezione si espone in dettaglio il funzionamento del sensore mentre nella quarta si procede descrivendo tutto ciò che è stato necessario alla preparazione del sistema. Nella quinta sezione viene seguita l'implementazione del progetto. Infine si conclude con delle considerazioni generali sul lavoro svolto.

2 Materiali e ambienti di sviluppo

Questa sezione sarà dedicata alla presentazione della strumentazione hardware e software utilizzata per il progetto. In particolare per la parte software si analizzeranno gli ambienti di sviluppo necessari per l'implementazione del progetto che si è svolta su due fronti: uno riguardante Red Pitaya e il sensore, l'altro relativo al robot NAO.

In entrambi i casi è stato utilizzato come linguaggio di programmazione Python [2]. Quest'ultimo è un linguaggio di programmazione ad alto livello, performante e ricco di librerie che lo rendono adatto allo sviluppo del progetto.

2.1 Robot NAO

NAO [3] è un robot umanoide creato dall'azienda SoftBank Robotics progettato per interagire con le persone. È dotato di sensori e motori che gli permettono di camminare, ballare, parlare e riconoscere volti o oggetti.

Essendo completamente programmabile è molto versatile ed adatto a molteplici ambiti. Per esempio è utilizzato nella ricerca, nell'istruzione e nell'assistenza sanitaria in tutto il mondo.

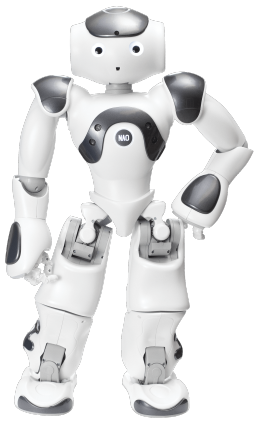

Figura 2: Il robot NAO

NAO è dotato di diverse interfacce tra cui:

- una videocamera;
- un altoparlante per emettere suoni e parlare;
- un microfono;
- una porta USB tramite cui ricaricarlo;
- una porta Ethernet per stabilire una connessione con il computer.

Il robot è ottimo per sviluppare il sistema di sicurezza di questo studio dato che, nella parte posteriore, è sprovvisto di sensori che gli permettano di riconoscere la presenza di un ostacolo.

2.2 Il sensore

Il sensore [4] ha una struttura molto semplice: è composto da un supporto di plastica con 60 avvolgimenti di cavo unipolare intorno ad esso.

Figura 3: Il sensore

Il sensore, alimentato ad una determinata frequenza, genera un campo magnetico variabile. Quando l'essere umano entra nel campo indotto dal sensore si generano su di esso delle correnti indotte. A loro volta, le correnti, variano nel tempo e generano un campo magnetico che si oppone al campo magnetico iniziale. Questa iterazione provoca l'alterazione dei parametri caratteristici del sensore come, per esempio, l'induttanza caratteristica. Sono proprio queste perturbazioni che, appositamente analizzate, permettono di riconoscere la presenza di un essere umano.

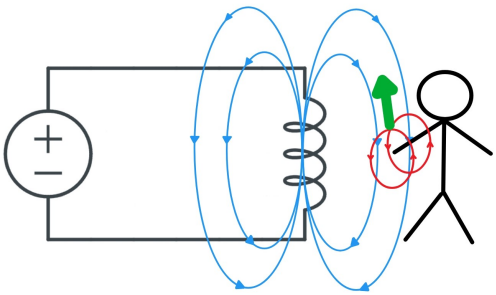

Figura 4: Rappresentazione funzionamento

2.3 Red Pitaya

Red Pitaya [5] è una piattaforma hardware e software di dimensioni contenute che sostituisce molti strumenti di misurazione e di controllo costosi. L'utilizzo di questa scheda ci permette di controllare ed elaborare le informazioni ricevute attraverso il sensore.

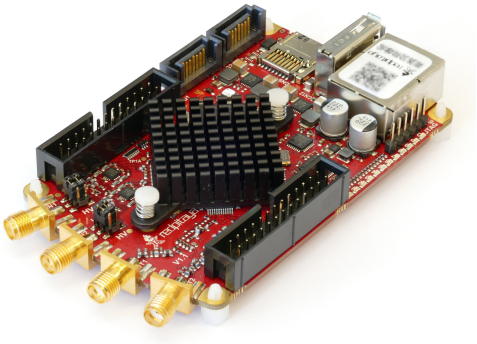

Figura 5: Red Pitaya

Il modello su cui si è lavorato è lo STEMlab 125-14 che monta un processore Dual-Core ARM Cortex-A9 MPCore ed è provvisto di una RAM di 4 Gb. Può essere programmata attraverso vari linguaggi di programmazione tra cui Python, Jupyter, MATLAB o LabVIEW, altrimenti molte delle sue applicazioni sono accessibili dall'interfaccia utente Web. Quest'ultima può essere raggiuta attraverso l'URL: rp-xxxxxx.local/ solo dopo aver connesso la Red Pitaya al computer, dove le "x" devono essere sostituite dal codice identificativo che si trova sulla scheda.

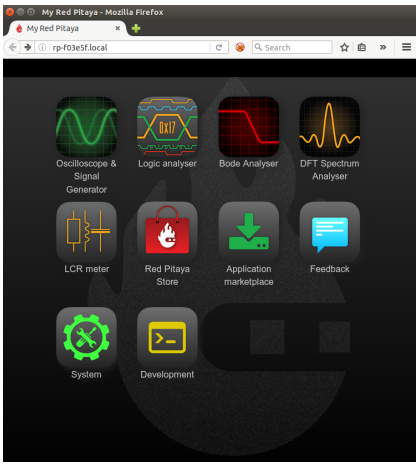

Figura 6: Interfaccia utente Web

Il modo sicuramente più veloce per connettere Red Pitaya al computer è attraverso LAN: sia con connessione con cavo ethernet al router sia sviluppando una connessione wireless. Nell'ultimo caso c'è bisogno di un dongle Wifi. Ci sono due altre alternative: connessione diretta tramite cavo ethernet al computer oppure attraverso la creazione di un Access Point. Per poter funzionare Red Pitaya ha bisogno di una scheda SD con, al suo interno, il sistema operativo installato.

2.4 Visual Studio Code

Per lavorare su Red Pitaya si è scelto di utilizzare Visual Studio Code [6], editor di codice leggero ma potente disponibile per Windows, macOS e Linux. É stato sviluppato da Microsoft e supporta vari linguaggi tra cui C++, C#, Java e Python.

Figura 7: Logo di Visual Studio Code

Visual Studio Code si sarebbe potuto usare anche per implementare il programma del robot NAO ma si è preferito utilizzare il suo software, presentato di seguito, per la possibilità di lavorare su un gemello virtuale e per la facilità di creazione di programmi in Python grazie all'uso di blocchi con una struttura e delle funzioni già definite.

2.5 Choregraphe

Choregraphe [7] è un ambiente di sviluppo dedicato alla programmazione dei prodotti della SoftBank Robotics. É un software di programmazione a blocchi dove si possono sfruttare i blocchi già esistenti oppure realizzarne di nuovi attraverso vari linguaggi di programmazione. Il punto di forza di Choregraphe è la possibilità di mandare in esecuzione i programmi su un robot virtuale così da poter fare dei test anche senza avere la disponibilità materiale del robot.

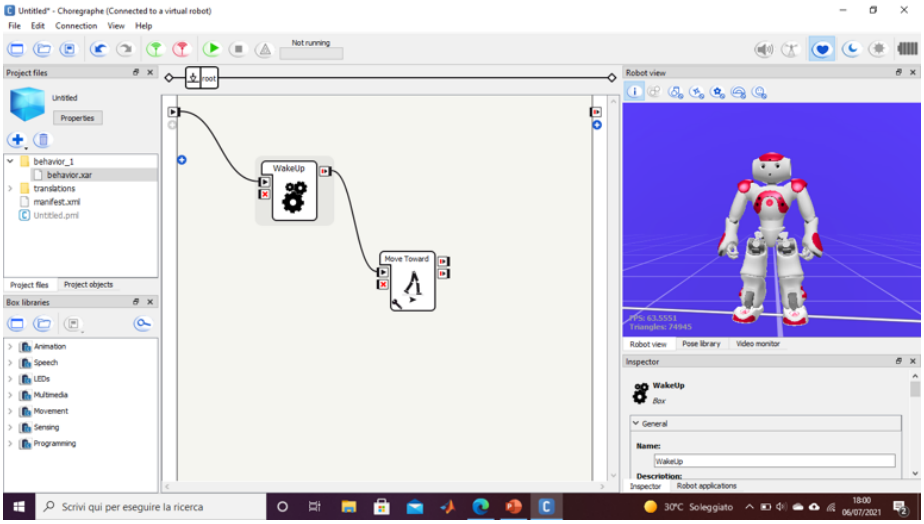

Figura 8: Schermata Choregraphe

La schermata del programma può dividersi in varie sezioni. Nella parte superiore si trovano i pulsanti per:

- creare un nuovo progetto;
- aprire un progetto esistente;
- salvare il progetto;
- connettere il robot alla sua forma virtuale o a quella reale;
- disconnettere il robot;
- visualizzare eventuali errori prodotti da un programma.

Sul lato sinistro si trovano tutti i blocchi utilizzabili mentre a destra è presente un monitor che permette di visualizzare il comportamento del robot.

Choregraphe consente agli utenti di programmare facilmente NAO [8]. Per creare un programma basta selezionare un blocco e inserirlo al centro della

schermata. Inoltre bisogna trascinare il mouse dall'input onStart della schermata all'input onStart del blocco per creare un collegamento come illustrato nella figura sottostante.

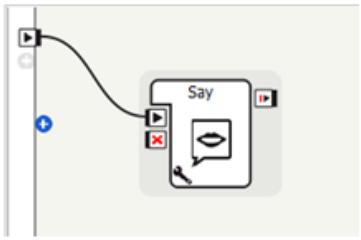

Figura 9: Struttura programma con blocco

Cliccando sul simbolo della chiave che si trova in basso a sinistra nel blocco si aprirà un pannello da cui l'utente può modificare vari parametri per personalizzare l'azione svolta dal robot. Infine basterà mandare in esecuzione il programma tramite l'apposito pulsante.

3 Il sensore in dettaglio

3.1 Funzionamento fisico

Il sensore permette di sfruttare le caratteristiche elettromagnetiche dell'essere umano al fine di perturbare alcuni parametri di un campo appositamente generato. Il ruolo del sensore è proprio quello di generare un campo che massimizzi la variazioni indotte dalla presenza dell'uomo. Questi risultati vengono elaborati attraverso la scheda Red Pitaya.

Sono state sviluppate due tipologie di sensori: i sensori di tipo capacitivo e i sensori induttivi.

3.1.1 Sensori di tipo capacitivo

Questa tipologia di sensore replica il comportamento di un condensatore. Si consideri un sensore con una superficie dei piani A e su ciascuna superficie una carica Q. Se alimentato si sviluppa all'interno del condensatore un campo elettrico costante E tra le due armature con valore:

$$\vec{E} = \frac{Q}{\varepsilon_0 \cdot A}$$

mentre risulta non costante alle estremità dei piani paralleli. Quest'ultimo fenomeno si chiama *effetto di bordo*. Sperimentalmente si è potuto evidenziare che è questa porzione del campo ad essere maggiormente sensibile alle perturbazioni dell'uomo.

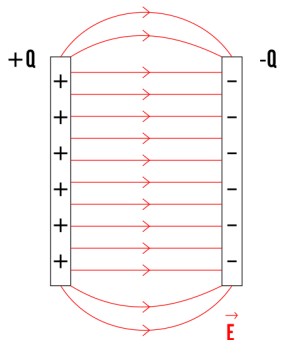

Figura 10: Campo elettrico condensatore

È stato dimostrato che con questa tipologia di sensori si possono rilevare esseri umani in un intorno di raggio 5-7 cm tale da fermare il braccio robotico quando viene percepita la presenza dell'essere umano ma non sufficiente a rendere l'ambiente sicuro.

3.1.2 Sensori induttivi

I sensori induttivi sono stati progettati per ampliare l'*effetto di bordo* ed espandere la zona sensibile. Si consideri un solenoide, strumento di solito usato come induttore, percorso da corrente che genera quindi un campo magnetico.

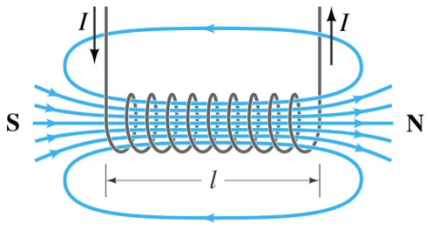

Figura 11: Campo magnetico solenoide

Questa soluzione, rispetto alla precedente, assicura una maggiore area di interazione con le macchine grazie all'estensione del campo nello spazio circostante. Inoltre, i sensori induttivi, permettono di sfruttare anche l'effetto capacitivo dato che tra due spire poste a una determinata distanza si va a formare un accoppiamento simile a quello dei condensatori analizzato in precedenza. Grazie a questo nuovo studio sono stati ottenuti risultati più soddisfacenti arrivando ad avere una sensibilità che supera i 20 cm.

Il sensore utilizzato per implementare il progetto è proprio un sensore induttivo.

4 Setup sperimentale

4.1 Preparazione Red Pitaya

Red Pitaya deve essere collegata sia al sensore sia al computer in modo da poter creare un campo magnetico ed elaborarne le perturbazioni. I collegamenti con il sensore sono riportati in figura.

Figura 12: Collegamenti Red Pitaya con sensore

Inoltre, la scheda, è stata collegata al computer attraverso una connessione diretta con cavo ethernet assegnandole un indirizzo IP statico. Questo tipo di collegamento ha permesso di avere una stabilità e velocità tale da consentire l'elaborazione continua dei dati in tempo reale e, allo stesso tempo, avere sempre lo stesso IP così da non doverlo modificare manualmente sul programma ogni volta che la scheda viene riconnessa.

4.2 Programma per rilevare vicinanza di un essere umano

Il programma da cui si è partiti permette di riconoscere la presenza di un essere umano ed è suddiviso in quattro classi analizzate di seguito [9]:

- **ButterWorth** implementa un filtro che permette di pulire il segnale e rimuovere i rumori.
- **SignalGeneration** permette, una volta stabilita la connessione con Red Pitaya, di generare un segnale, acquisirlo e salvare i risultati in un array.
- **Calibration** è la classe che viene eseguita per prima nel programma in quanto permette di calibrare il sensore attraverso il calcolo della frequenza di risonanza.
- **Main** è la parte centrale del programma. Inizialmente si crea un oggetto del tipo Calibration che serve a calibrare il sensore in modo da trovare la frequenza di risonanza, si lavorerà ad una frequenza molto vicina a quest'ultima. Dopodiché c'è un ciclo che permette di trovare 20 volte l'ampiezza relativa al segnale generato alla frequenza calcolata, questa serie di valori viene salvata in un array. Quindi verrà calcolata la media e la deviazione standard dei valori dell'array. Infine si ha un ciclo infinito che, attraverso il calcolo della media delle ampiezze confrontato con la deviazione standard, permette di verificare in tempo reale la presenza di un essere umano.

Per permettere il corretto funzionamento del programma sono state installate alcune librerie. La libreria principale che è stata utilizzata è PyRPL, software open source che fornisce molti strumenti per schede hardware FPGA. Inoltre sono state usate: PyQt5, PyQt5-sp, PySide2, PyYALM, QtPy e Quamash.

4.3 Connessione NAO

Per connettere il robot NAO al computer si è deciso di sviluppare una connessione wireless. In particolare sia NAO che il computer sono stati connessi ad un Access Point, quindi ad una stessa rete. La connessione è risultata abbastanza stabile da permettere lo scambio di dati tra Red Pitaya e il robot. La gestione della connessione di NAO è avvenuta tramite la sua Web Page nella sezione Network.

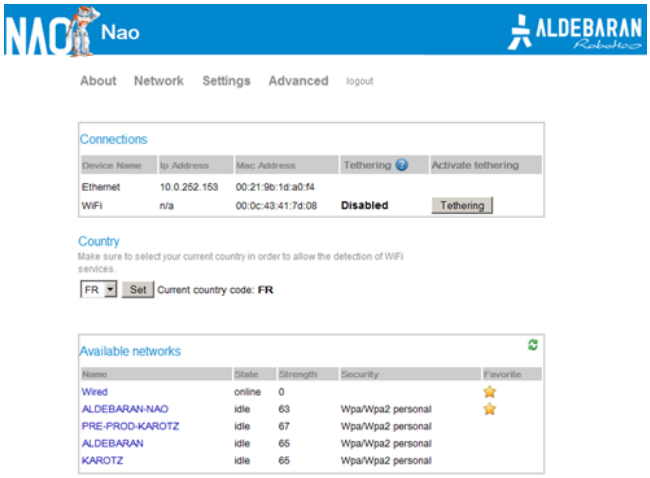

Figura 13: Sezione Network NAO robot

Per il progetto è stato assegnato al robot un IP statico che permette di avere un indirizzo sempre uguale su cui stabilire la comunicazione tra Red Pitaya e NAO.

5 Sviluppo

L'obiettivo di questo progetto è permettere al robot NAO di rilevare la presenza di un essere umano in modo da poterne evitare la collisione. Per far ciò è stato necessario creare una comunicazione tra Red Pitaya e NAO. In particolare le alternative analizzate sono state due: la creazione di un server che permetta lo scambio dei dati tra i due dispositivi e la comunicazione attraverso un protocollo IP. La prima soluzione è stata esclusa quasi immediatamente dopo aver capito che non sarebbe stato possibile importare la libreria per la creazione di un server in Choregraphe. Quindi si è deciso di utilizzare la seconda alternativa: una comunicazione tramite protocollo IP. In particolare per gli obiettivi di questo progetto è risultato adatto il protocollo UDP.

5.1 Protocollo UDP

UDP [10] è un protocollo di rete che permette il trasferimento di dati su più terminali connessi ad una stessa rete. Supporta sia il traffico multicasting che broadcasting. È adatto allo scopo di questo progetto in quanto predilige un invio tempestivo dei dati. Un'applicazione nota che sfrutta il protocollo UDP è lo streaming di contenuti video.

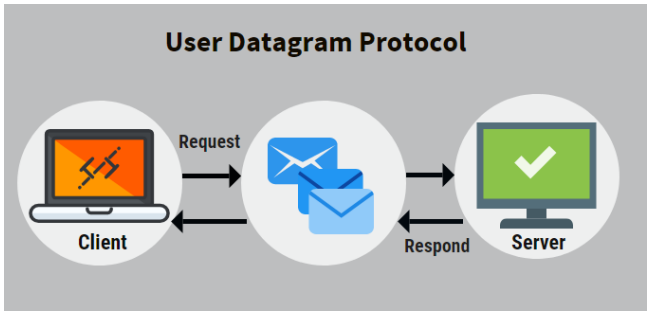

Figura 14: Schema UDP

I terminali tra cui avviene la comunicazione saranno identificati come lato server e lato client. Viene creato il server che rimarrà in ascolto su un indirizzo e una porta scelti. Il client contatterà il server quando dovrà comunicare. Quest'ultimo non seguirà una procedura di connessione al server per essere autenticato ma verrà conosciuto dal server solo quando invierà una richiesta.

Per implementare una connessione UDP tra NAO e Red Pitaya prima di tutto si è individuato quale dispositivo avrebbe dovuto funzionare come server e quale come client. Il robot NAO è risultato più adatto ad essere utilizzato come server. In particolare il robot una volta creato il server deve aspettare fermo che gli vengano inviati i dati dal client, cioè da Red Pitaya. Quando il

client comincia ad inviare le richieste, il robot comincia a muoversi all'indietro e deve rispondere in maniera diversa a seconda che venga rilevata la presenza di un essere umano o meno. Nel caso in cui non venga rilevata continuerà a muoversi mentre, se viene rilevata, dovrà fermarsi finché l'essere umano non sarà abbastanza lontano.

Si lavora quindi su due programmi in Python che permettono di comunicare e scambiarsi informazioni. In particolare per realizzare la connessione si fa uso della libreria socket e delle relative funzioni.

5.2 Programma robot NAO

Per implementare il programma di NAO è stato creato in Choregraphe un blocco programmabile in Python.

```
class MyClass(GeneratedClass):
    def __init__(self):
        GeneratedClass.__init__(self)

    def onLoad(self):
        pass

    def onUnload(self):
        pass

    def onInput_onStart(self):
        pass

    def onInput_onStop(self):
        self.onUnload()
```

Listato 1: Struttura blocco Python Choregraphe

Il blocco creato di default avrà cinque funzioni, nel nostro caso ne andremo ad utilizzare tre:

- **onUnload**: metodo che viene chiamato mentre viene scaricato il flow diagram del blocco. Dopo che è stato scaricato, il blocco non potrà ricevere più nessun evento in ingresso.
- **onInput_onStart**: metodo che viene eseguito quando il blocco va in esecuzione.
- **onInput_onStop**: metodo chiamato quando si ferma l'esecuzione del blocco.

Di seguito verrà analizzato il programma sviluppato per NAO. Fuori dalla classe viene importata la libreria socket e implementata la parte server della comunicazione. Vengono poi importate le classi di NAOqi così da poter utilizzare i metodi per permettere al robot di muoversi e cambiare posizione.

```
import socket

sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
udp_port = 12359                    # specified port to connect
udp_host = "10.10.10.3"                         # Host IP
sock.bind((udp_host,udp_port))

motionProxy = ALProxy("ALMotion")
```

```
posturePro xy = ALProxy("ALRobotPosture")
```

Listato 2: Creazione lato server

La prima funzione scritta è *onInput_ onStart* dato che viene eseguita all'avvio del blocco. Il ciclo infinito permette di ricevere continuamente da Red Pitaya un valore che verrà salvato nella variabile *a*, nel caso venga ricevuta la stringa contenente "1" il robot deve continuare a muoversi altrimenti deve fermarsi. Per far camminare e fermare il robot è stata usata la funzione *setWalkTargetVelocity*.

```
def onInput_onStart(self):
        postureProxy.goToPosture("StandInit", 0.5)
        while True:
            try:
                data,addr = sock.recvfrom(1024)
                a = data.decode()
            except socket.error:
                sock.close()
                X = 0.0
                Y = 0.0
                Theta = 0.0
                Frequency =0.0
                motionProxy.setWalkTargetVelocity(X, Y, Theta,
    Frequency)
                break

            if(a == "1") :
                motionProxy.setWalkArmsEnabled(True, True)
                motionProxy.setMotionConfig([["
    ENABLE_FOOT_CONTACT_PROTECTION", True]])
                X = -0.5
                Y = 0.0
                Theta = 0.0
                Frequency =0.0
                motionProxy.setWalkTargetVelocity(X, Y, Theta,
    Frequency)
            else:
                X = 0.0
                Y = 0.0
                Theta = 0.0
                Frequency =0.0 # low speed
                motionProxy.setWalkTargetVelocity(X, Y, Theta,
    Frequency)
```

Listato 3: onInput_onStart

Nel caso della funzione *onInput_ onStop* semplicemente viene passata la funzione *onUnload*.

```
def onInput_onStop(self):
        self.onUnload()
```

Listato 4: onInput_onStop

Per ultima viene implementata la funzione *onUnload* che, una volta terminato il programma, fa fermare il robot in posizione eretta.

```
def onUnload(self):
        sock.close()
        motionProxy.stopMove()
        X = 0.0
        Y = 0.0
        Theta = 0.0
        Frequency =0.0
        motionProxy.setWalkTargetVelocity(X, Y, Theta, Frequency)
        postureProxy.goToPosture("StandInit", 0.5)
```

Listato 5: onUnload

5.3 Programma Red Pitaya

Nel programma di Red Pitaya viene implementata la parte client che permette di inviare i dati al robot. La classe che viene modificata, rispetto al programma precedentemente analizzato per riconoscere la presenza di un individuo, è la classe Main.

Fuori dal ciclo infinito verrà importata la libreria socket e creato il lato client della comunicazione. L'host scelto è l'indirizzo IP del server, quindi del robot, mentre la porta è quella scelta per comunicare.

```
import socket

sock = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
udp_host = "10.10.10.3"
udp_port = 12359
```

Listato 6: Creazione client

L'obiettivo è quello di inviare al server una stringa a seconda se l'essere umano è stato rilevato o meno usando la funzione *sendto*. Vengono inviate tre diverse stringhe:

- **"0:1"**: se si è presentato un errore nel programma;
- **"0"**: se non è stato rilevato nessun essere umano;
- **"1"**: se è stata rilevata la presenza di un essere umano.

Alla fine è stato inserito anche un controllo attraverso eccezione che permette al programma di uscire dal ciclo while in caso di problemi con la comunicazione UDP.

```
while 1:
    try:
        try:
            amplitude1 = findAmplitude(frequency)
            amplitude2 = findAmplitude(frequency)
            amplitude = (amplitude1 + amplitude2)/2
            buffer = np.append(buffer, amplitude)
            if (j%5) == 0:
                deviation = np.std(buffer)
                buffer = np.ndarray(0)
            mean = abs(amplitude - media)
            dvs = (deviation + devStd)/2
            print("Media: ", mean, "; dev std: ", dvs)
        except Exception as e:
            print(e)
            msg = "0:1"
            sock.sendto(msg.encode(),(udp_host,udp_port))
        if mean > dvs * 1.5:
```

```
19              print("Ostacolo rilevato")
20              msg = "0"
21          else:
22              print("Ostacolo NON rilevato")
23              msg = "1"
24          sock.sendto(msg.encode(),(udp_host,udp_port))
25          j+=1
26      except socket.error:
27          break
```

Listato 7: Ciclo while con invio stringa

5.4 Sistema sul robot

La parte finale del progetto consiste nell'adattare il sistema di sicurezza al robot in modo da permettergli di trasportarlo. Nel particolare dovrà essere in grado di trasportare il sensore, la scheda Red Pitaya e i cavi per i vari collegamenti.

Innanzitutto è stata modificata la connessione tra Red Pitaya e il computer dato che il cavo ethernet, avendo una lunghezza limitata, non permetteva di allontanarsi abbastanza. Per far ciò inizialmente è stata implementata una connessione wireless ma con risultati non soddisfacenti in quanto non garantiva una velocità e stabilità tale da inviare dati in tempo reale. L'alternativa è stata collegare Red Pitaya attraverso cavo ethernet direttamente a un ripetitore a sua volta alimentato tramite powerbank. Questa soluzione è risultata migliore in quanto si è tornati ad avere un invio dei dati corretto senza perdita di informazioni.

Figura 15: Ripetitore per connessione Red Pitaya

Inserendo un powerbank nel sistema è stato risolto anche il problema dell'alimentazione in quanto si è potuto collegare Red Pitaya a quest'ultimo. Inoltre NAO è stato collegato, tramite wireless, al ripetitore in modo da essere connesso alla stessa rete di Red Pitaya e del computer così da permettere la comunicazione tramite UDP.

Per far trasportare il sistema a NAO si è deciso di utilizzare uno zainetto che possa contenere tutto il materiale facendo sì che il robot non si sbilanci durante i vari movimenti.

6 Conclusioni

In conclusione si è riusciti a realizzare un'applicazione del sistema di sicurezza sul robot NAO. Quest'ultimo riesce ad evitare il contatto di un essere umano quando ne viene rilevata la presenza nelle vicinanze. Si è riusciti a stabilire una connessione abbastanza stabile per lo scambio di informazioni tra NAO e Red Pitaya che ha permesso di creare un ambiente sicuro.

Questo risultato è un punto di partenza per le future implementazioni anche in ambiti di maggiore rilevanza quale quello industriale. NAO è stato sicuramente un'ottima scelta per lo sviluppo del progetto date le sue dimensioni contenute e la sua non pericolosità per l'uomo che hanno reso i test molto semplici. Tuttavia è un robot molto limitato che non permette piena libertà nella programmazione del software. Sarà sicuramente di maggiore interesse integrare il sensore a robot industriali potenzialmente pericolosi. Inoltre si potrà migliorare la precisione di scambio dei dati e la velocità con cui essa avviene per applicare questo sistema di sicurezza a macchinari in situazioni più realistiche.

Riferimenti bibliografici

[1] A. A Khalid, B. P Kirisci, C. Z Ghrairi, D. K-D Thoben, E. J Pannek and F. Editor *Towards Implementing Safety and Security Concepts for Human-Robot-Collaboration in the context of Industry 4.0*

[2] Python, *Python 3.9.7 documentation* https://docs.python.org/3/

[3] SoftBank Robotics, *NAO Documentation* http://doc.aldebaran.com/2-1/home_nao.html

[4] Genovesi Riccardo *Analisi e studio di un sensore E-Field per la cooperazione tra uomo e macchina*

[5] Red Pitaya, *Red Pitaya Documentation* https://redpitaya.readthedocs.io/en/latest/

[6] Visual Studio Code, *Documentation for Visual Studio Code* https://code.visualstudio.com/docs

[7] SoftBank Robotics, *Choregraphe Suite* http://doc.aldebaran.com/2-5/software/choregraphe/index.html

[8] SoftBank Robotics, *NAOqi Documentation* http://doc.aldebaran.com/2-5/index_dev_guide.html

[9] Conte Giuseppe, Scaradozzi David, Carnevale Pasquale, De Grandis Simone, Tesei Riccardo *Tesina approfondimento pratico laboratorio di meccatronica*

[10] Santosh Kumar, Sonam Rai *Survey on Transport Layer Protocols: TCP & UDP*