Nome: Gustavo Silva e Luiz Fernando
RA: 2084040

Relatório de Projeto de Arquitetura de Computadores

O projeto realizado na matéria de **Arquitetura e Organização de Computadores** foi desenvolvido como projeto final da disciplina cujo tinha como objetivo montar algo utilizando o ***raspberry pi 3*** onde se pode emular a execução de um sistema operacional como **Linux, Windows.**

Durante o decorrer do projeto se obteve algumas dificuldades como a instalação e execução de alguns métodos do openCV3, por conta que a versão do sistema não estava reconhecendo a instalação, ou seja, estava dando erro ao executar o comando **pip install opencv=3.10.0-numpy.**

Ferramentas utilizadas

- 1 raspberry pi 3;
- 1 ponte h;
- Cabos Macho Fêmea;
- 1 Câmera própria do raspberry pi
- 1 Notebook
- 1 Teclado;
- 1 Mouse;
- 1 Cabo HDMI, USB;

Montagem do projeto

De inicio foi realizado a montagem do carrinho utilizando as conexões GPIO do raspberry, para cada porta de conexão da ponte H, ou seja, para que se pudesse controlar as rodas do carrinho que posteriormente foram ativadas ou desativadas conforme a escolha do comando.

Para conseguir montar o código dos motores foi importada a biblioteca ***gpiozero*** e então dessa forma importamos o método *PWMOutputDevice* que foi onde cada pino foi setado para que se pudesse utilizá-lo, logo após realizar a configuração das portas foram criados as funções **esquerda(), direita(), frente() e a parada()**.

→ Código

```
#! /usr/bin/env python3

# Importando as Bibliotecas

from gpiozero import PWMOutputDevice
import time
```

```
# from opencv import cv2

#-------------------------------------------------
# ->> Nome do grupo: Gustavo S./Luis R.
# ->> Matéria: Arquitetura de Computadores
# ->> Professor/Orientador: Itamar Iliuk
# ->> Periódo: 3° Semestre
#-------------------------------------------------

# Configurando as portas GPIO
IN1 = 26 # Motor 1°
IN2 = 19 # Motor 2°
IN3 = 13 # Motor 1°
IN4 = 6 # Motor  2°

# Inicializando os pinos para que possam ser ativados/desativados

# Motor Primáio
Frente_Esq = PWMOutputDevice(IN1, True, 0, 1000)
Frente_Dir = PWMOutputDevice(IN3, True, 0, 1000)

# Motor Secundário
Re_Dir = PWMOutputDevice(IN2, True, 0, 1000)
Re_Esq = PWMOutputDevice(IN4, True, 0, 1000)

# Criando as funções responsáveis por controlar os motores

def frente():
      Frente_Esq.value = 1.0
      Frente_Dir.value = 1.0
      Re_Esq.value = 0
      Re_Dir.value = 0

def direita():
      Frente_Dir.value = 1.0
      Frente_Esq.value = 0
      Re_Dir.value = 1.0
      Re_Esq.value = 0

def esquerda():
      Frente_Dir.value = 0
      Re_Esq.value = 1.0
      Frente_Esq.value = 1.0
      Re_Dir.value = 0

def parada():
      Frente_Dir.value = 0
      Re_Esq.value = 0
      Frente_Esq.value = 0
      Re_Dir.value = 0
```

Para cada função pino foi atribuído um valor de 0 onde seria identificado como desligado e o valor máximo 1 que seria a potencia, ou seja, ligado.

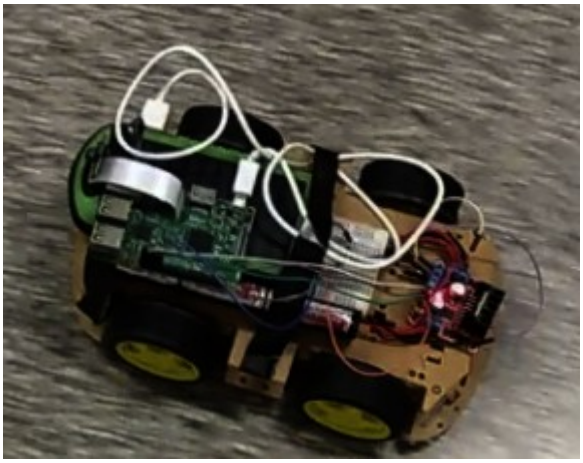

Figura 1
Carrinho Montado

Após realizar a montagem do carrinho, foi realizado a montagem da câmera cujo mostrou alguns empecilhos como realizar a instalação da biblioteca do **opencv** que necessitava de uma versão mais atual para que pudesse reconhecer o objeto, e pesquisando mais sobre a biblioteca e os métodos utilizados para realizar instalação foi possível chegar a um web site cujo tinha realizado uma instalação mais completa através de repositórios no **GitHub.**

→ Código da Câmera.

```
import cv2
import numpy as np
import biblioteca as b
import carrinho

cap = cv2.VideoCapture(0)
_, frame = cap.read()
tela = b.Tela(frame)

while True:
    _, frame = cap.read()
    hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    low_green = np.array([45, 100, 80])
    high_green = np.array([85, 255, 255])
    green_mask = cv2.inRange(hsv_frame, low_green, high_green)

    l = cv2.findContours(green_mask,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

    if (len(l[1]) == 0):
        print("parado")
```

```
else:
    v = l[1][0][0][0][0]
    if(v < tela.D1):
        print("esquerda")
        esquerda()
    elif(v > tela.D2):
        print("direita")
        direita()
    elif(v < tela.D2):
        print("frente")
        frente()
    else:
        print("Parado")
        parado()


green = cv2.bitwise_and(frame, frame, mask=green_mask)
```