# Report

# When I run the code all things goes well, the code create a connection between the shell and the simulator Udacity. But, an error appear, [GET /socket.io/?EIO=4&transport=websocket HTTP/1.1](https://github.com/llSourcell/How_to_simulate_a_self_driving_car/issues/34#top), so i had to fix this problem in the next session.

# I change the code drive, because the old don’t work i try everthing. So i change all the code and that appear well.

# The code :

import argparse

import base64

from datetime import datetime

import os

import shutil

import numpy as np

import socketio

import eventlet

import eventlet.wsgi

from PIL import Image

from flask import Flask

from io import BytesIO

import tensorflow as tf

import h5py

sio = socketio.Server()

app = Flask(\_\_name\_\_)

model = None

prev\_image\_array = None

class SimplePIController:

    def \_\_init\_\_(self, Kp, Ki):

        self.Kp = Kp

        self.Ki = Ki

        self.set\_point = 0.

        self.error = 0.

        self.integral = 0.

    def set\_desired(self, desired):

        self.set\_point = desired

    def update(self, measurement):

        # proportional error

        self.error = self.set\_point - measurement

        # integral error

        self.integral += self.error

        return self.Kp \* self.error + self.Ki \* self.integral

controller = SimplePIController(0.1, 0.002)

# 781 is good to 9

# 881

set\_speed = 10

controller.set\_desired(set\_speed)

@sio.on('telemetry')

def telemetry(sid, data):

    if data:

        # The current steering angle of the car

        steering\_angle = data["steering\_angle"]

        # The current throttle of the car

        throttle = data["throttle"]

        #print("throttle", throttle)

        # The current speed of the car

        speed = data["speed"]

        # The current image from the center camera of the car

        imgString = data["image"]

        image = Image.open(BytesIO(base64.b64decode(imgString)))

        image\_array = np.asarray(image)

        steering\_angle = float(model.predict(image\_array[None, :, :, :], batch\_size=1))

        throttle = controller.update(float(speed))

        print(steering\_angle, throttle)

        send\_control(steering\_angle, throttle)

        # save frame

        if args.image\_folder != '':

            timestamp = datetime.utcnow().strftime('%Y\_%m\_%d\_%H\_%M\_%S\_%f')[:-3]

            image\_filename = os.path.join(args.image\_folder, timestamp)

            image.save('{}.jpg'.format(image\_filename))

    else:

        # NOTE: DON'T EDIT THIS.

        sio.emit('manual', data={}, skip\_sid=True)

@sio.on('connect')

def connect(sid, environ):

    print("connect ", sid)

    send\_control(0, 0)

def send\_control(steering\_angle, throttle):

    sio.emit(

        "steer",

        data={

            'steering\_angle': steering\_angle.\_\_str\_\_(),

            'throttle': throttle.\_\_str\_\_()

        },

        skip\_sid=True)

if \_\_name\_\_ == '\_\_main\_\_':

    parser = argparse.ArgumentParser(description='Remote Driving')

    parser.add\_argument(

        'model',

        type=str,

        help='Path to model h5 file. Model should be on the same path.'

    )

    parser.add\_argument(

        'image\_folder',

        type=str,

        nargs='?',

        default='',

        help='Path to image folder. This is where the images from the run will be saved.'

    )

    args = parser.parse\_args()

    model = tf.keras.models.load\_model(args.model)

    if args.image\_folder != '':

        print("Creating image folder at {}".format(args.image\_folder))

        if not os.path.exists(args.image\_folder):

            os.makedirs(args.image\_folder)

        else:

            shutil.rmtree(args.image\_folder)

            os.makedirs(args.image\_folder)

        print("RECORDING THIS RUN ...")

    else:

        print("NOT RECORDING THIS RUN ...")

    # wrap Flask application with engineio's middleware

    app = socketio.Middleware(sio, app)

    # deploy as an eventlet WSGI server

    eventlet.wsgi.server(eventlet.listen(('', 4567)), app)