THE AMERICAN UNIVERSITY IN CAIRO

School of Sciences and Engineering
Mechanical Engineering Department

RCSS 5930

Industrial IoT and Digital Twins of Cyber Physical Systems

Fall 2022

Final Report

Submitted To:

Dr. Mohamed Abdelsalam

By:

Mohamed Hamdy
Marwan Shehata
Mahmoud Ayman Kamal
Aya Fouad
Nada Ali
Alaa Elfiqi
Eman Khaled

Due Date:

23 / 12 / 2022

Submitted Date:

23 / 12 / 2022

Table of Contents

Table of Contents 1

List of Figures 2

INTRODUCTION & BACKGROUND 3

1.1 General Understanding 3
1.2 Applications 3
1.2.1 Smart Homes 3
1.2.2 Healthcare 3
1.2.3 Smart Cities 3
1.3 Problem Statement 4

PROJECT PHYSICAL SYSTEM 5

2.1 Physical System Components 5
2.1.1 Ultrasonic Sensor 5
2.1.2 Motors and Motor Controller 5
2.1.3 Arduino 5
2.1.4 Raspberry pi 6
2.2 Physical Connections 6
2.3 Arduino Code 7

PROJECT CYBER SYSTEM 10

3.1 Robotics Operating System (ROS) 10
3.2 Amazon Web Services (AWS) 11
3.3 CARLA Simulator 15
3.4 Functional Mockup Interface (FMI) 22
3.4.1 Trial 1: Using Ego Vehicle FMU 22
3.4.2 Trial 2: Using Braking System FMU 23
3.4.2.1 Comment on results: 24
3.4.3 Trial 3: Creating an FMU 25
3.4.4 Trial 4: Simulating an FMU 25

INTEGRATING PHYSICAL AND CYBER SYSTEMS 27

4.1 Data Flow 27

REFERENCES 54

List of Figures

Figure 1: Ultrasonic Sensor [4] ... 5
Figure 2: DC Gain Motor [5] ... 5
Figure 3: Motor Controller [6] ... 5
Figure 4: Arduino Mega 2560 [7] ... 6
Figure 5: Raspberry Pi [8] ... 6
Figure 6: Project Schematic Design ... 6
Figure 7: Arduino and Raspberry Pi Connection [9] ... 7
Figure 8: Arduino Code (1) ... 7
Figure 9: Arduino Code (2) ... 8
Figure 10: Arduino Code (3) ... 8
Figure 11: Arduino Code (4) ... 8
Figure 12: Arduino Code (5) ... 9
Figure 13: ROS (1) ... 10
Figure 14: ROS (2) ... 10
Figure 15: Creating AWS Account ... 11
Figure 16: Creating a "Thing" ... 11
Figure 17: Creating Certificate (1) ... 12
Figure 18: Creating Certificate (2) ... 12
Figure 19: Creating Policy (1) ... 13
Figure 20: Creating Policy (2) ... 13
Figure 21: Creating Policy (3) ... 14
Figure 22: Connect Policy to IoT Thing (1) ... 14
Figure 23: Connect Policy to IoT Thing (2) ... 15
Figure 24: CARLA Starter and ROS Subscriber ... 15
Figure 25: Docker CARLA Server Initializer ... 16
Figure 26: CARLA Main Script (1) ... 16
Figure 27: CARLA Main Script (2) ... 16
Figure 28: CARLA Main Script (3) ... 17
Figure 29: CARLA Main Script (4) ... 17
Figure 30: CARLA Main Script (5) ... 17
Figure 31: CARLA Main Script (6) ... 18
Figure 32: CARLA Main Script (7) ... 18
Figure 33: CARLA Main Script (8) ... 18
Figure 34: CARLA Main Script (9) ... 19
Figure 35: CARLA Main Script (10) ... 19
Figure 36: CARLA Main Script (11) ... 19
Figure 37: CARLA Main Script (12) ... 20
Figure 38: CARLA Main Script (13) ... 20
Figure 39: CARLA Main Script (14) ... 20
Figure 40: CARLA Main Script (15) ... 21
Figure 41: CARLA Main Script (16) ... 21
Figure 42: CARLA Main Script (17) ... 21
Figure 43: CARLA Simulation ... 22
Figure 44: Error message for trial 1 (part 1) ... 23
Figure 45: Error message for trial 1 (part 2) ... 23
Figure 46: Error in installing PyFMI library ... 23
Figure 47: Error in trial 2 (part 1) ... 24
Figure 48: Error in trial 2 (part 2) ... 24
Figure 49: Graphical User Interface of FMPy Library ... 25
Figure 50: Simulation results of a sample FMU to show the system response ... 26

INTRODUCTION & BACKGROUND

1.1 General Understanding

The Internet of Things (IoT) refers to any device that has an embedded technology that can connect and exchange data with other devices and humans through the internet. These devices are often referred to as smart objects. They can collect information about their environment and send them for analysis. Based on the gained results, the device can take an action [1][2].

IoT systems must be able to:

- Sense and collect information
- Process the collected data and send control signals accordingly.
- Convert the electrical control signals to mechanical movements to perform the needed actions
- Connect different devices to each other and the cloud.
- Protect sensitive data

1.2 Applications

1.2.1 Smart Homes

Some of the common uses of IoT in smart homes includes having a smart lightning system. If a room is empty of people, the lights are switched off automatically. The heating system is another common use. The rooms have temperature sensors that take readings after a set amount of time. If the room temperature is measured to be less than the required set temperature, the heater is automatically turned on [3].

1.2.2 Healthcare

This is one of the most important advancements done. Doctors can monitor their patient's health at all times. A patient can be at home using a smart sphygmomanometer while the doctor is at the clinic. The results of the blood pressure get analyzed and sent to the cloud where the authorized doctors can access it [3].

1.2.3 Smart Cities

IoT has multiple implementations in smart cities. The traffic lights can be controlled depending on priority. This means that in case a firefighting truck is on its way to an emergency, it won't have to wait until the traffic lights are green. The traffic lights will turn green as soon as the truck gets closer [2].

1.3 Problem Statement

For this course project, we built an IoT obstacle avoidance car that is connected to a virtual one on CARLA simulator. An ultrasonic sensor will measure the distance between the car and the obstacle in front of it. If the car gets too close to the obstacle, it will automatically stop. Simultaneously, the CARLA simulator should at the same time as the physical car.

PROJECT PHYSICAL SYSTEM

2.1 Physical System Components

2.1.1 Ultrasonic Sensor

The ultrasonic sensor is a proximity sensor that is made of two main components: the transmitter and the receiver. The transmitter emits ultrasonic sound waves, and the receiver waits for the reflection. The distance is calculated by measuring the time it took to transmit and receive the sound wave [4].

In this project, we used the ultrasonic sensor in figure 1 to be able to detect if there is an obstacle blocking the car's pathway.

Figure 1: Ultrasonic Sensor [4]

2.1.2 Motors and Motor Controller

We used four DC gain motors to drive the obstacle avoidance car. One motor was used to drive each of the wheels. For the purpose of the project, the motors will either keep rotating in the same direction (car moving forward) or it will stop (in case an obstacle is detected). The motors will be controlled using the Adafruit motor shield shown in figure 3.

Figure 2: DC Gain Motor [5]

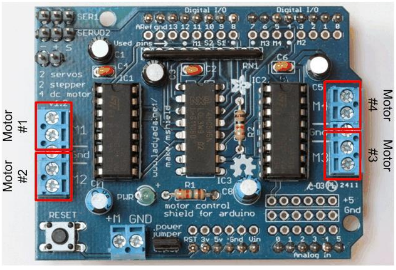


Figure 3: Motor Controller [6]

2.1.3 Arduino

The Arduino Mega 2560 microcontroller was used in this project to be able to receive readings from the ultrasonic sensors and transmit them to the raspberry pi (which acts as the main computer). Simultaneously, the microcontroller will make decisions based on the sensor readings to either keep the car moving or stop it.

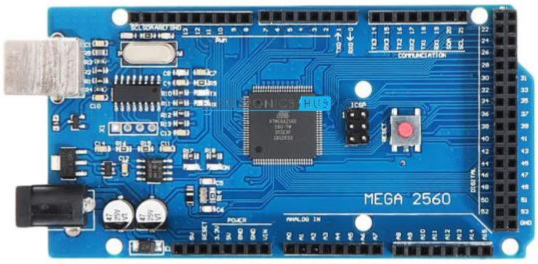


Figure 4: Arduino Mega 2560 [7]

2.1.4 Raspberry pi

Raspberry pi is a single board computer that can be used for various purposes. It can connect to the internet which makes it a perfect solution for IoT projects. Raspberry pi is the main controller in this assignment. It receives readings from the ultrasonic sensor and publishes it to AWS through the MQTT protocol.

Figure 5: Raspberry Pi [8]

2.2 Physical Connections

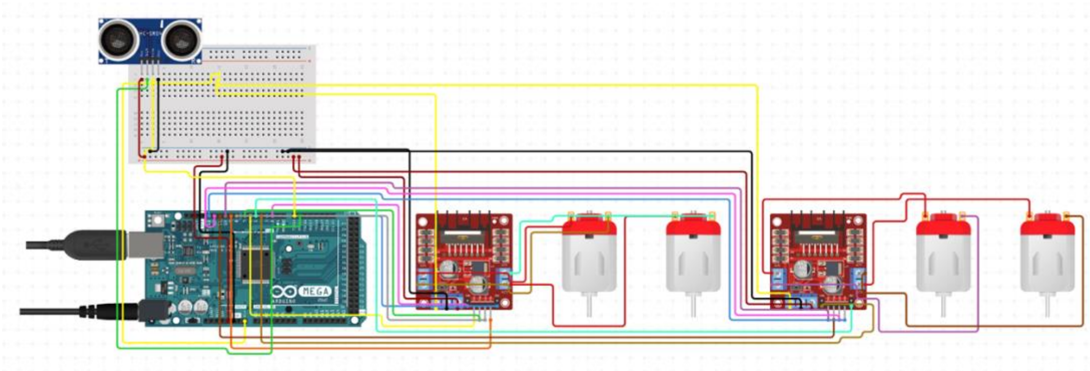


Figure 6: Project Schematic Design

A circuit design application (circuito.io) was used to create the schematic design shown in figure 6 above. Therefore, we were unable to locate the same components used in our physical design. However, the schematic design provides a good overview of the original connection.

The motor driver and ultrasonic sensor are both directly connected to the Arduino microcontroller, and the four gear motors are linked together via jumper wires to the motor controller.

Before the final connection was made, each of these components was tested separately to ensure that they are functioning properly. Fortunately, all the components were behaving normally. Therefore, we moved on with combining the elements as shown above. However, the motors were not rotating due to low power supply, so we decided to power the design prototype using a wall adaptor (at least while for the testing period). Finally, the Arduino board was connected to the raspberry pi via the USB port as shown in figure 7 below to begin integrating the physical and cyber sections of the project.

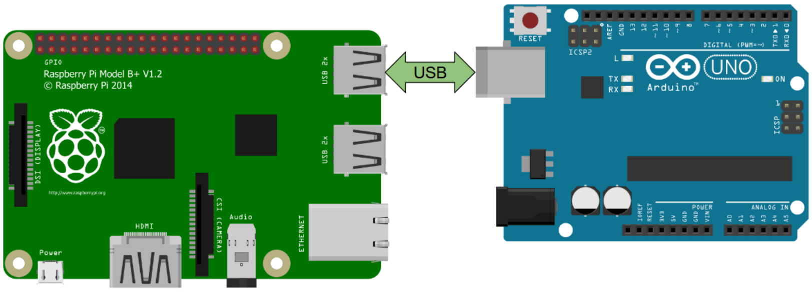


Figure 7: Arduino and Raspberry Pi Connection [9]

2.3 Arduino Code

The Arduino code presented in this section is used to control the movements of the physical car.

```
#include <AFMotor.h>


AF_DCMotor motor1(1);
AF_DCMotor motor2(2);
AF_DCMotor motor3(3);
AF_DCMotor motor4(4);

#define echoPin 16
#define trigPin 17


int distance;
int min_distance;
int x= 0;

long duration;
int distanceU;

```

Figure 8: Arduino Code (1)

```
int ultrasonic()
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distanceU = duration * 0.034 / 2;

  return distanceU;

}
```

Figure 9: Arduino Code (2)

```
void moveForward() {
  motor1.setSpeed(200);
  motor2.setSpeed(200);
  motor3.setSpeed(200);
  motor4.setSpeed(200);
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}

void Stop() {
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
}
```

Figure 10: Arduino Code (3)

```
void setup() {
  Serial.begin(9600);
  min_distance = 5;

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);

}
```

Figure 11: Arduino Code (4)

```
void loop() {
distance = ultrasonic();


  if (distance <= min_distance)
  {
    Stop();
    Serial.println(distance);
  }
  else
  {
    moveForward();
    Serial.println(distance);
  }
delay(1900);
}
```

Figure 12: Arduino Code (5)

To summarize the code in Figures 8-12:

- Figure 8: Including the motor controller library and defining the 4 motors used, the trig and echo pins of the ultrasonic sensor and some important variables that will be used at a later stage in this code.
- Figure 9: Shows the ultrasonic sensor code finding the time between the emission of the sound wave and receiving it as well as calculating the distance.
- Figure 10: Instructions of what the motors speed and direction when it moves forward and when it stops.
- Figure 11: Is initializing the minimum distance, defined ‘trigPin’ of the ultrasonic sensor as input and ‘echoPin’ as the output.
- Figure 12: This is the main system loop that will keep measuring the distance and take actions accordingly.

PROJECT CYBER SYSTEM

3.1 Robotics Operating System (ROS)

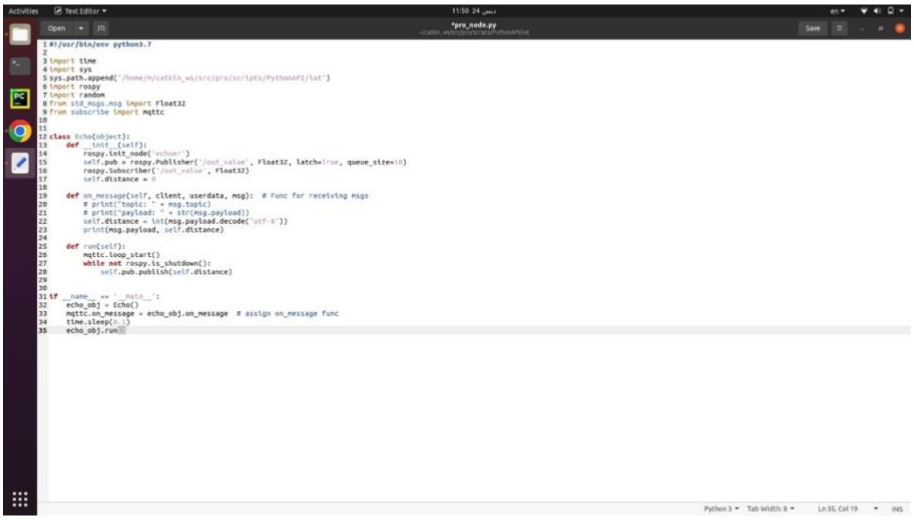


Figure 13: ROS (1)

The script file in figure 13 shows a ROS publisher on a topic named '/out_value' and runs an AWS IoT MQTT subscriber

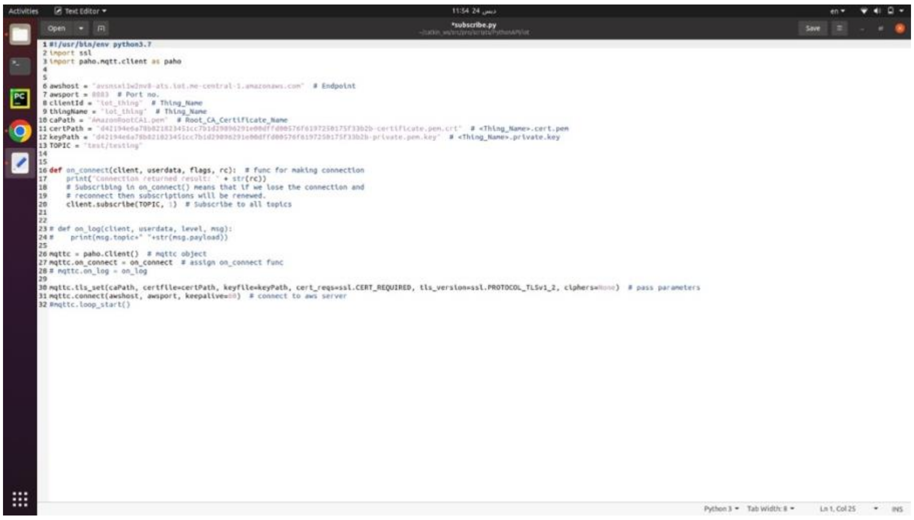


Figure 14: ROS (2)

Figure 14 is the AWS IoT MQTT subscriber. It subscribes to the AWS IoT topic called 'test/testing'. However, there is no need for us to run the AWS IoT MQTT subscriber individually as it is already imported in the ROS publisher file in Figure 13.

3.2 Amazon Web Services (AWS)

Amazon Web Services (AWS) offers more than 200 fully featured services from data centers across the world, making it the most complete and commonly used cloud platform in the world. Millions of customers use AWS to reduce costs, improve agility, and innovate more quickly, including huge corporations, leading government agencies, and startups with rapid growth [10]. The following are the steps taken to create the AWS cloud:

1. Creating AWS account

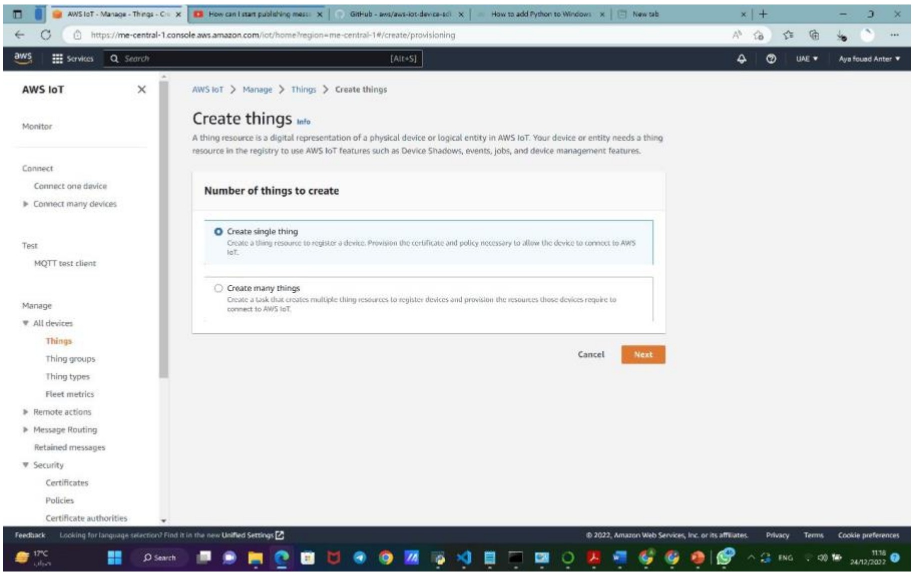


Figure 15: Creating AWS Account

2. Create a thing and give it a name (IOT-thing)

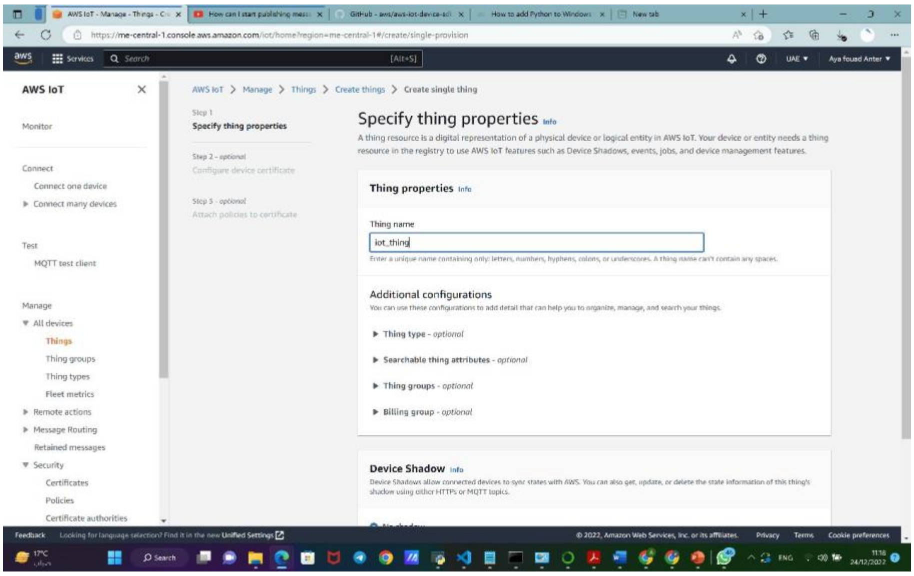


Figure 16: Creating a "Thing"

3. Create a certificate and download the files

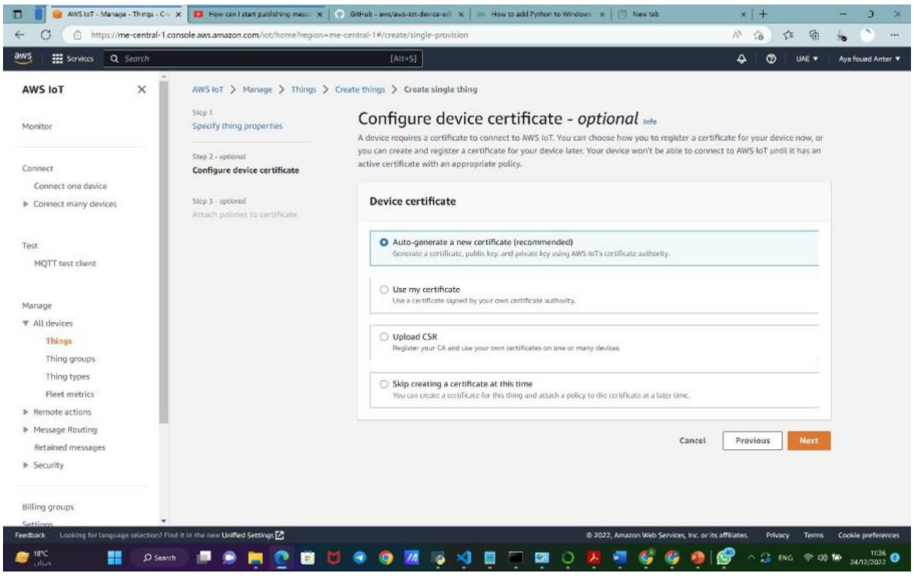


Figure 17: Creating Certificate (1)

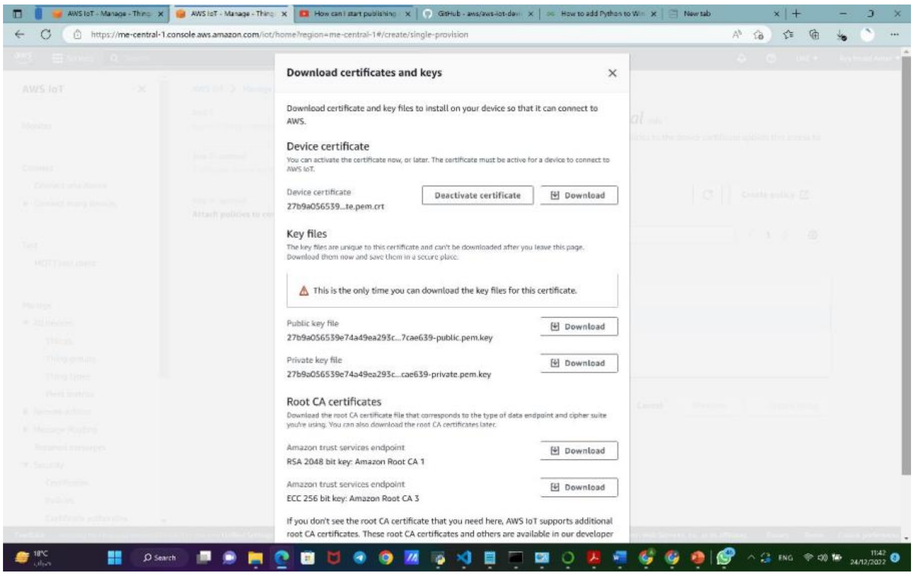


Figure 18: Creating Certificate (2)

4. Creating a policy and give it a name (IOT-policy):

Policy effect : allow

Policy action: IOTconnect*

Policy resource (*)

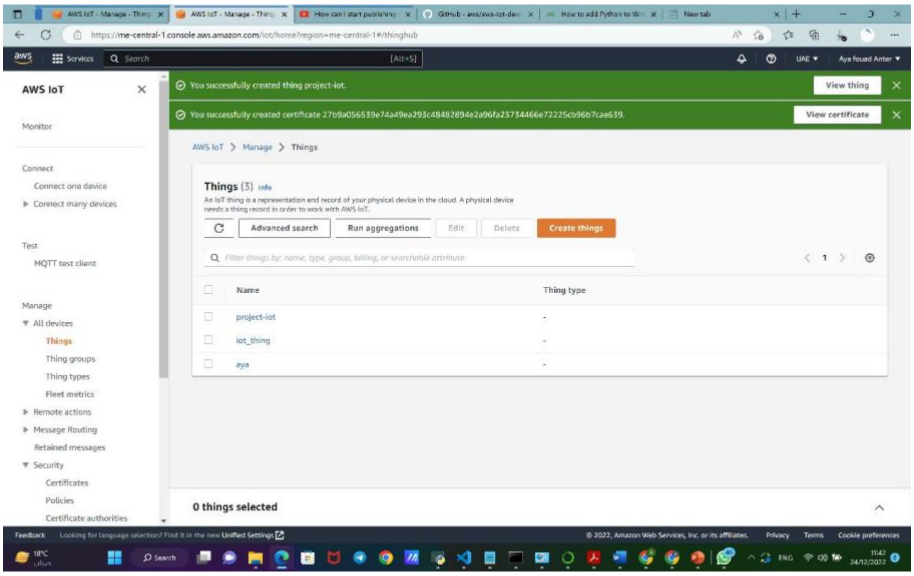


Figure 19: Creating Policy (1)

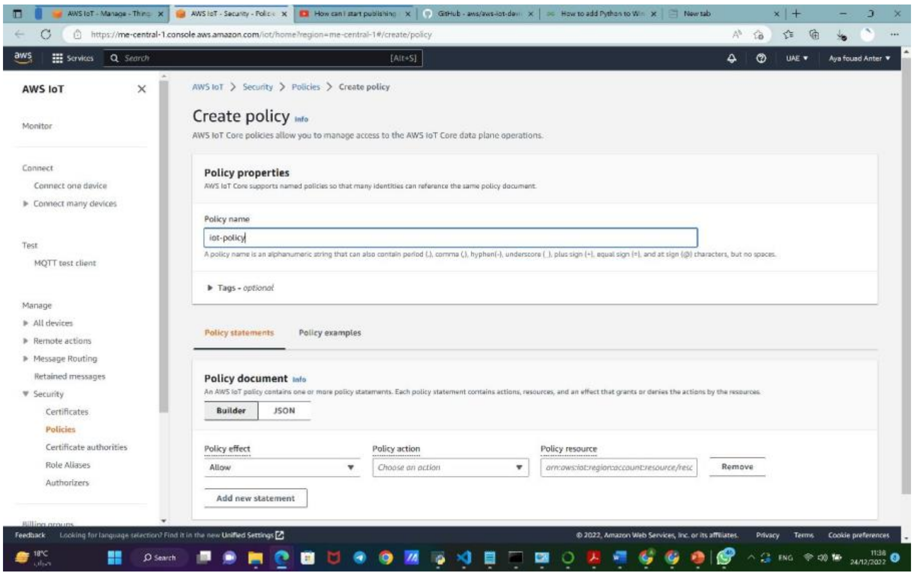


Figure 20: Creating Policy (2)

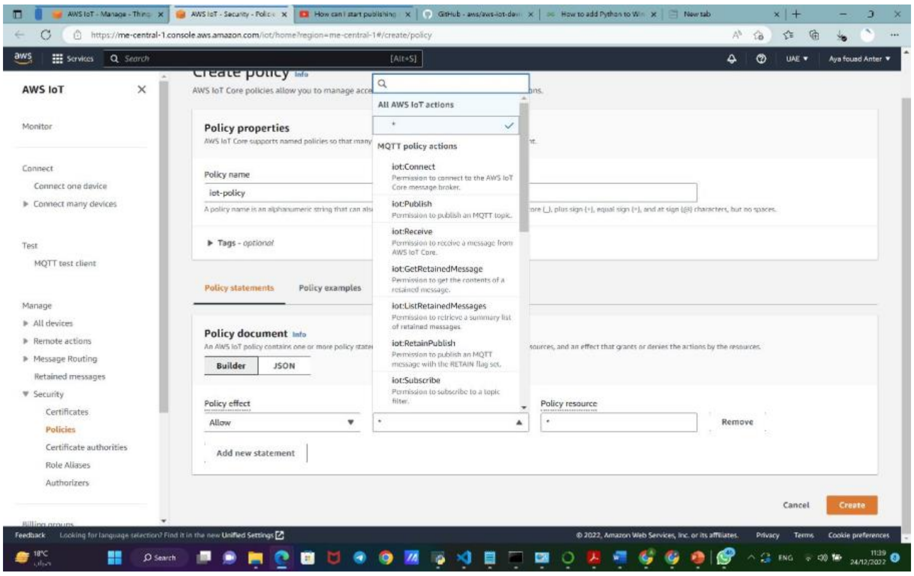


Figure 21: Creating Policy (3)

5. Attach the policy to IOT-thing created.

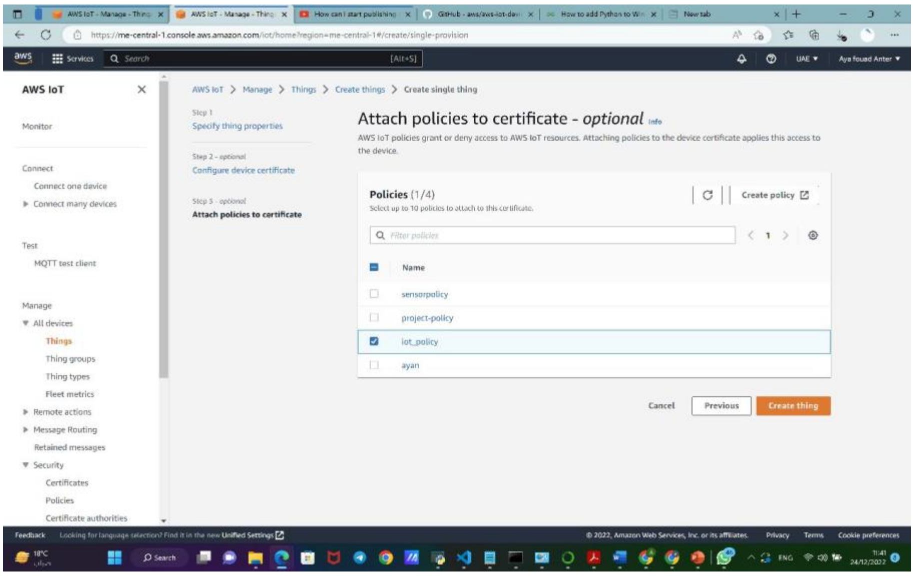


Figure 22: Connect Policy to IoT Thing (1)

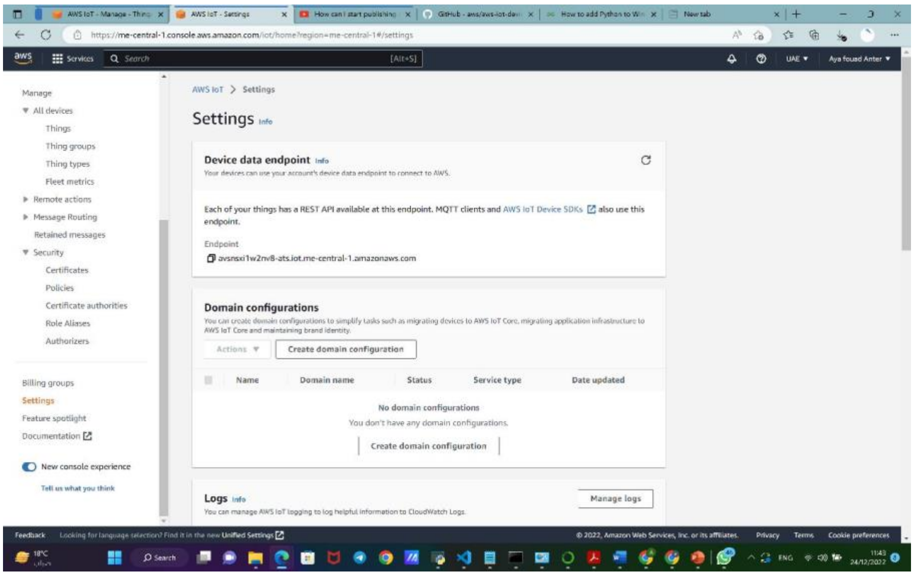


Figure 23: Connect Policy to IoT Thing (2)

3.3 CARLA Simulator

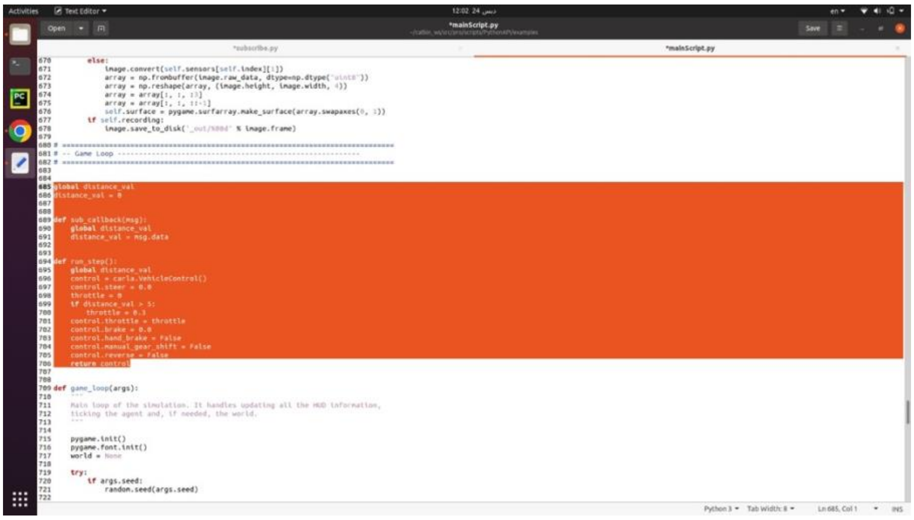


Figure 24: CARLA Starter and ROS Subscriber

CARLA script includes a CARLA starter and ROS subscriber.

The highlighted part in Figure 24 is the most important part in the CARLA script. The distance is retrieved by subscribing to ROS topic '/out_value' which we had previously published to. After that the distance gets processed. By default the throttle value of the vehicle in CARLA is 0 unless the CARLA script receives distance greater than 5. In that case, vehicle will start to move with constant throttle value 0.3.

CARLA simulation is a server client-based connection, and the CARLA starter is a client which connects to a server. The Docker will first start a CARLA server to which the CARLA client can connect

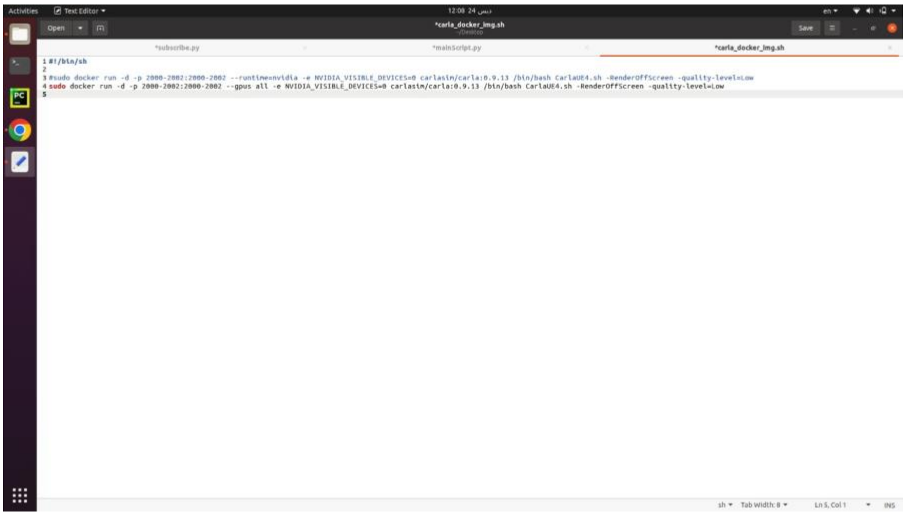


Figure 25: Docker CARLA Server Initializer

The figures below show CARLA's main script file. The code I very long so it was divided to multiple screenshots.

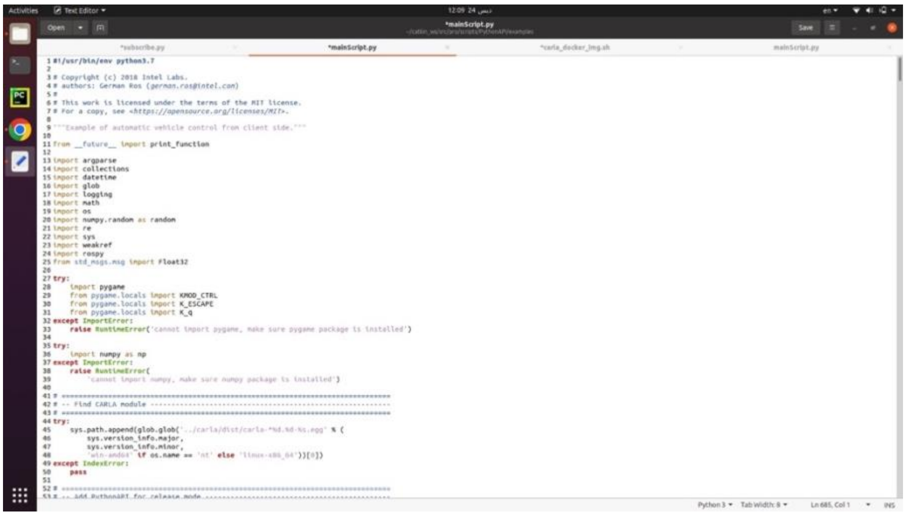


Figure 26: CARLA Main Script (1)

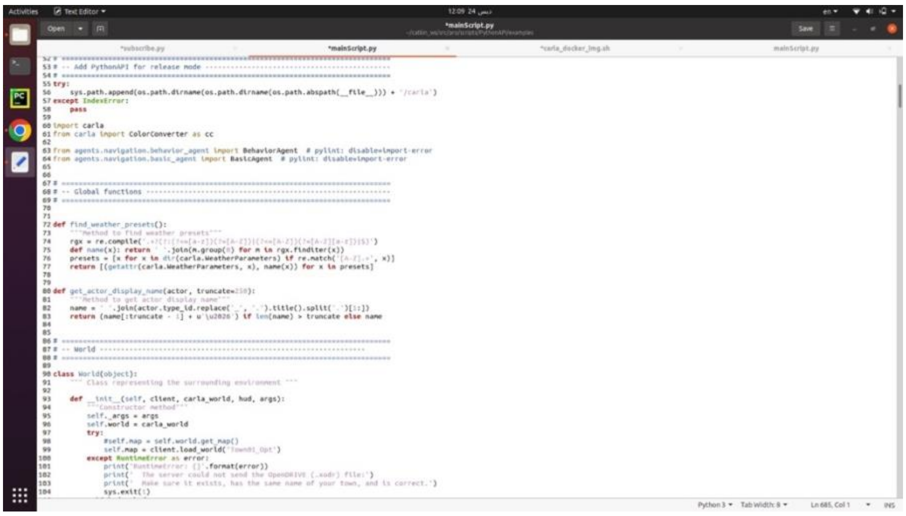


Figure 27: CARLA Main Script (2)

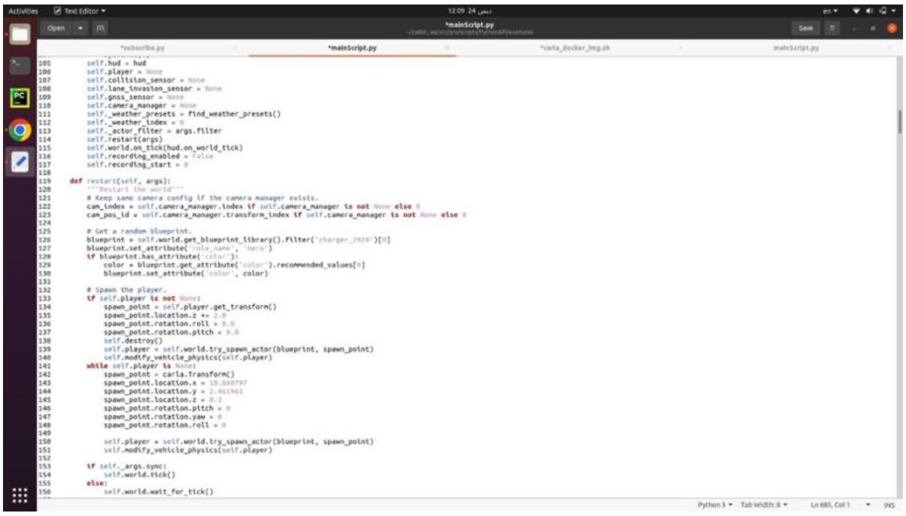


Figure 28: CARLA Main Script (3)

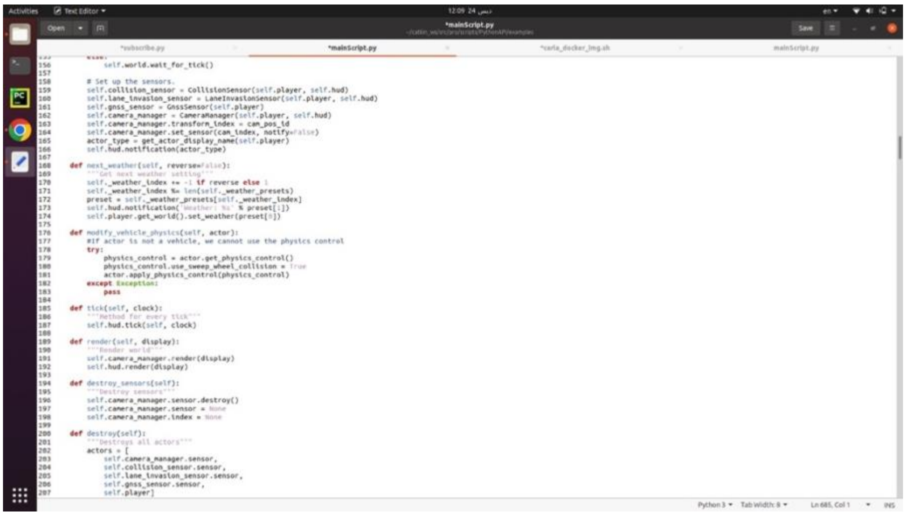


Figure 29: CARLA Main Script (4)

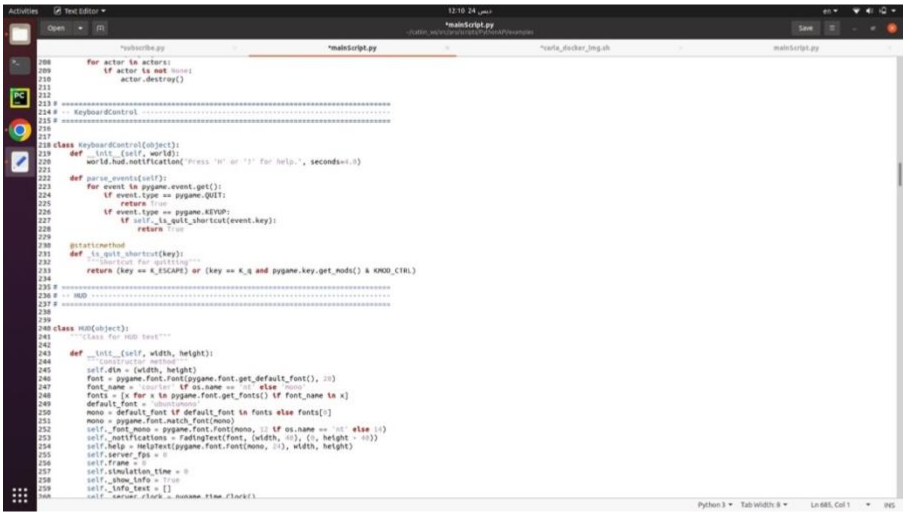


Figure 30: CARLA Main Script (5)

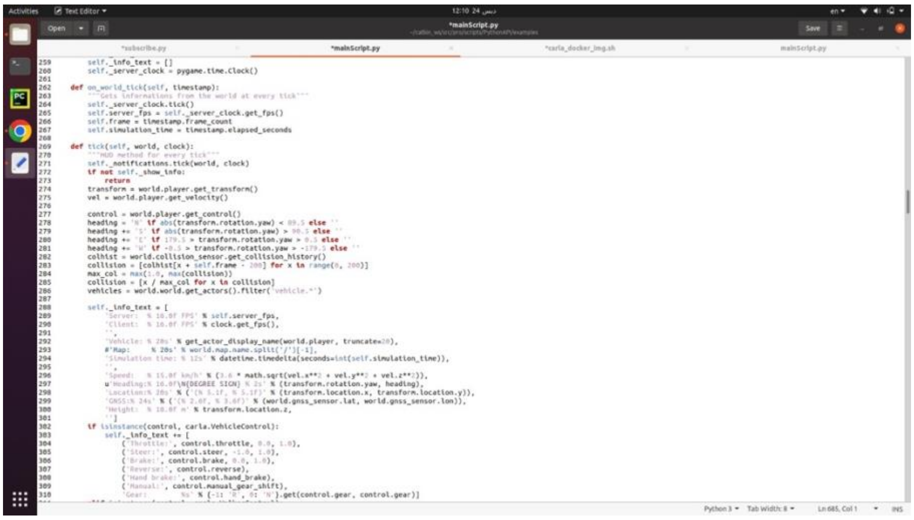


Figure 31: CARLA Main Script (6)

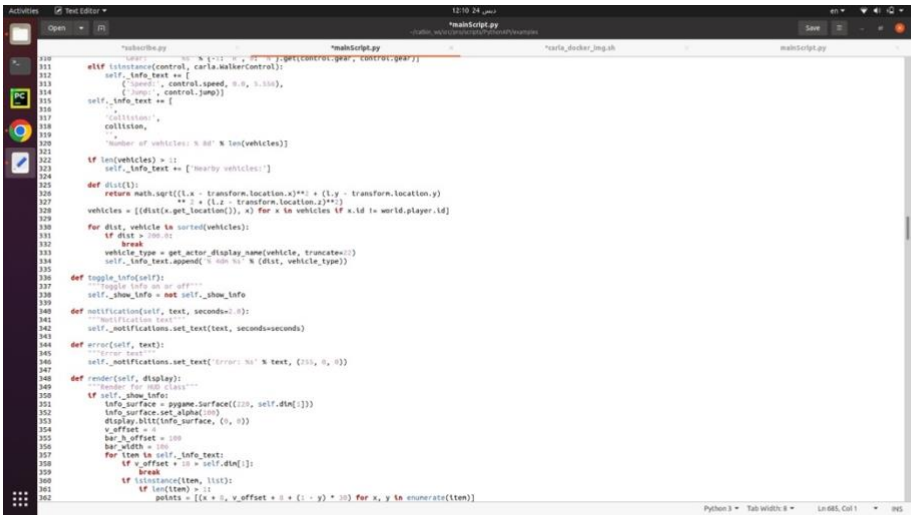


Figure 32: CARLA Main Script (7)

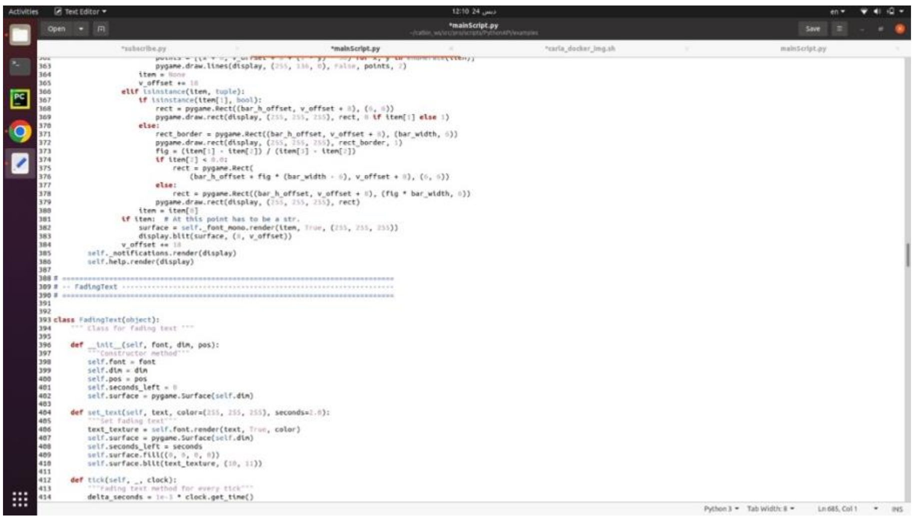


Figure 33: CARLA Main Script (8)

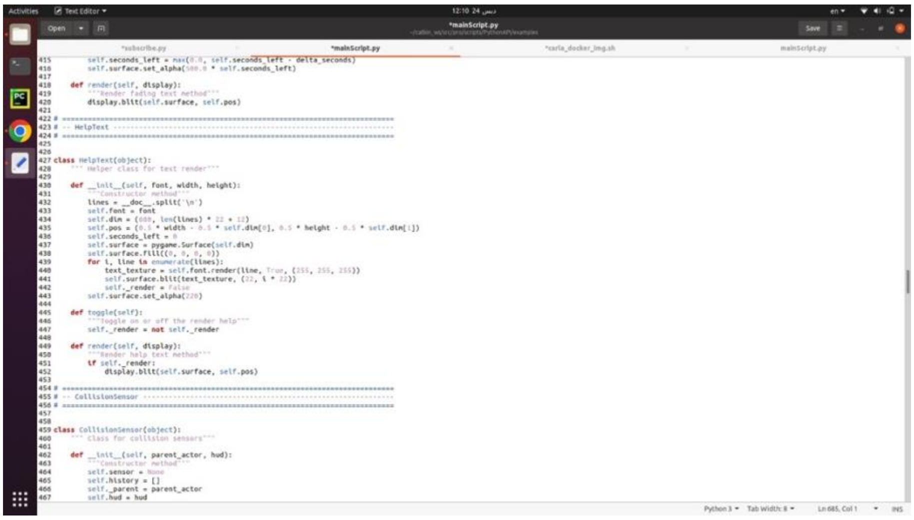


Figure 34: CARLA Main Script (9)

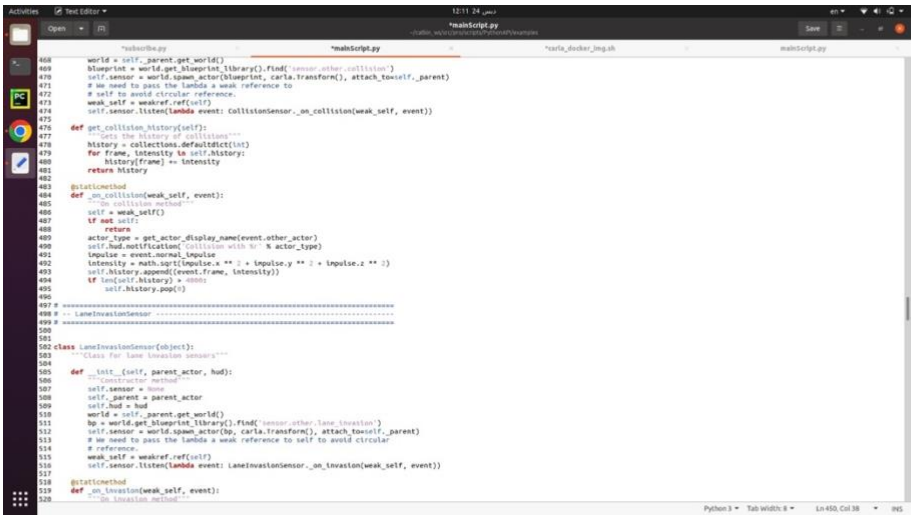


Figure 35: CARLA Main Script (10)

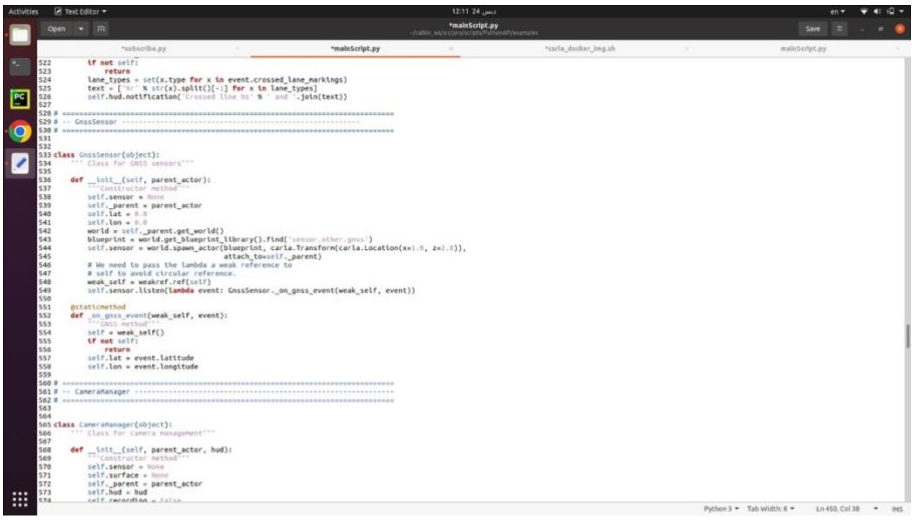


Figure 36: CARLA Main Script (11)

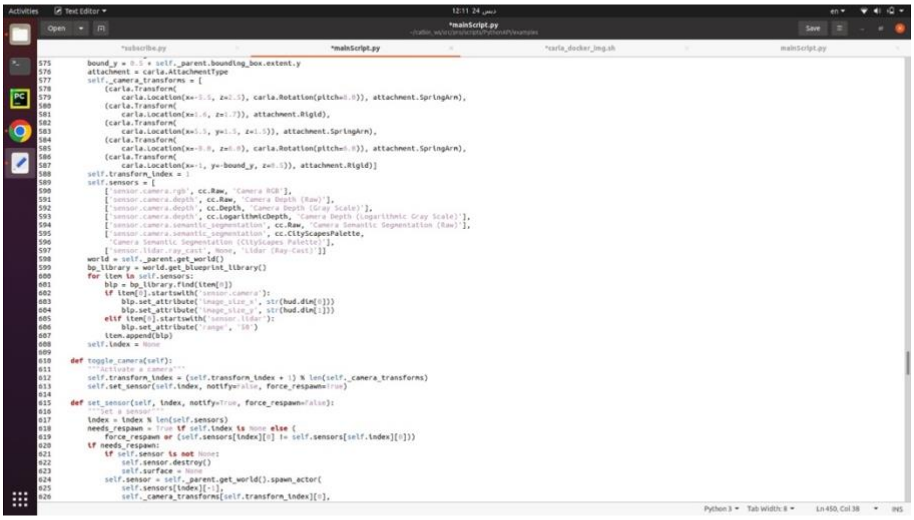


Figure 37: CARLA Main Script (12)

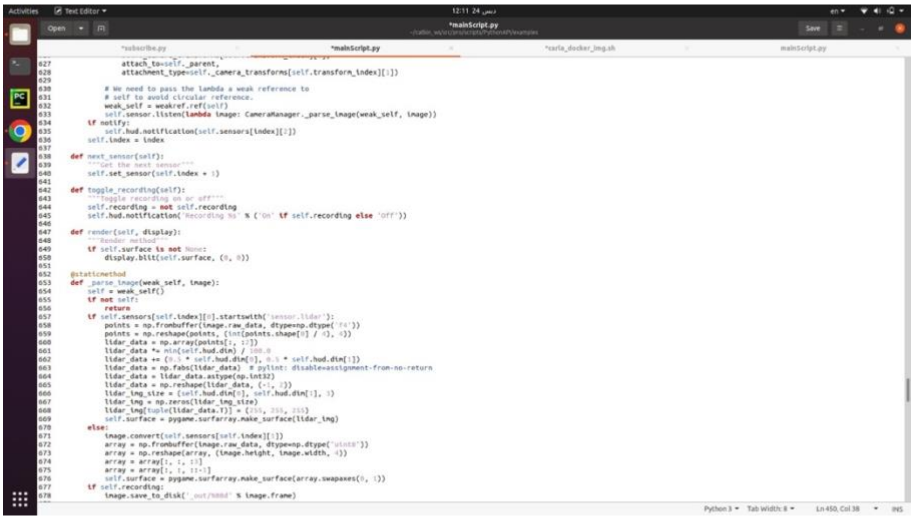


Figure 38: CARLA Main Script (13)

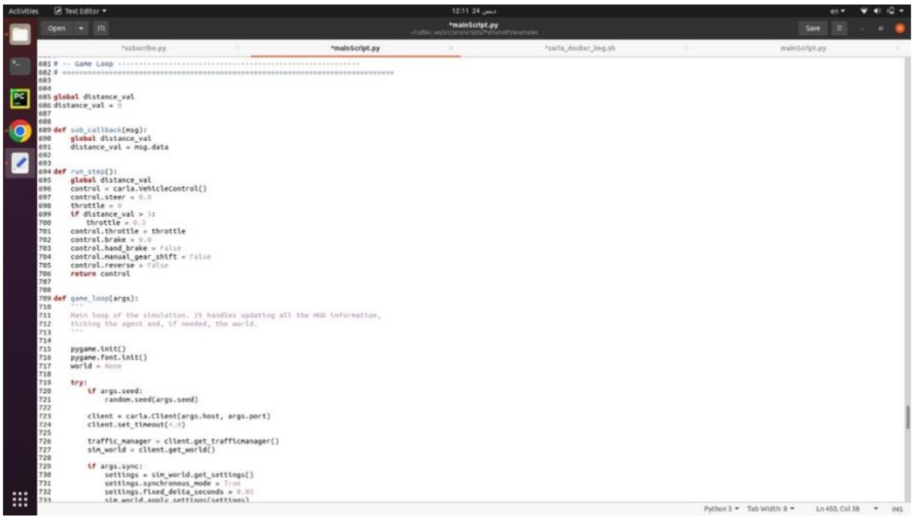


Figure 39: CARLA Main Script (14)

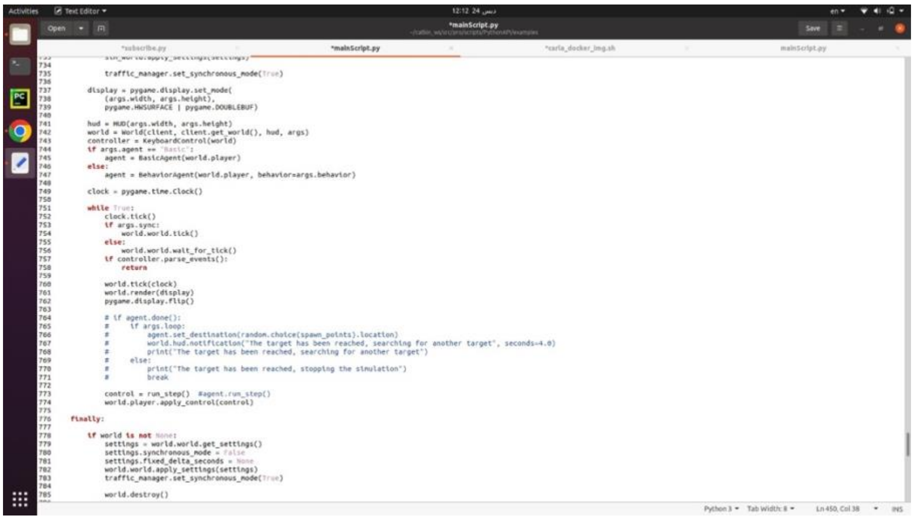


Figure 40: CARLA Main Script (15)

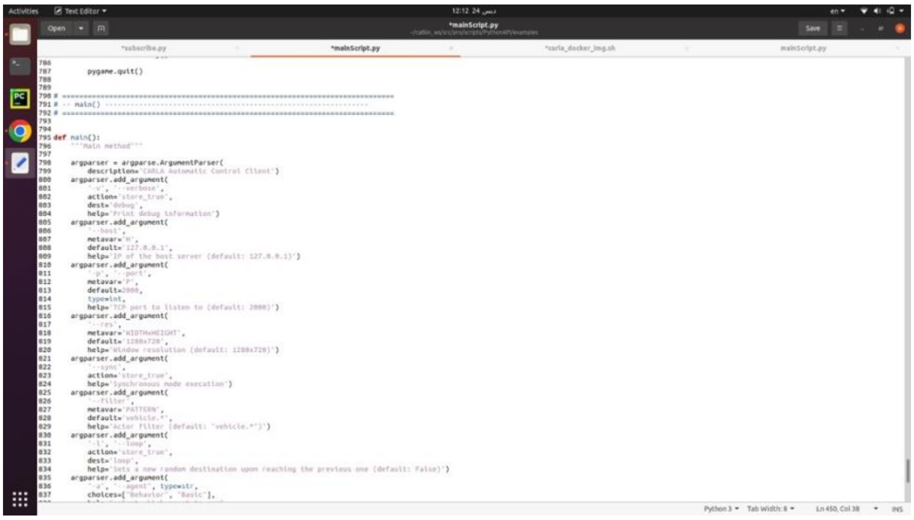


Figure 41: CARLA Main Script (16)

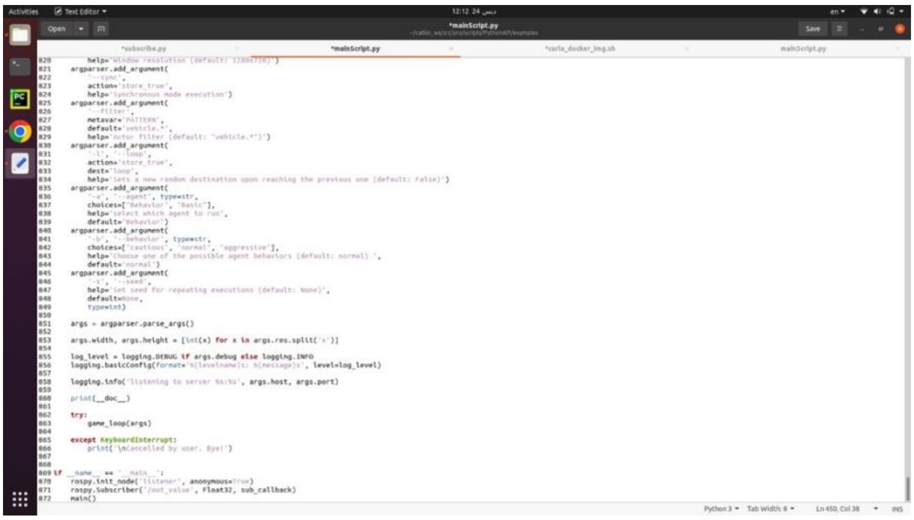


Figure 42: CARLA Main Script (17)

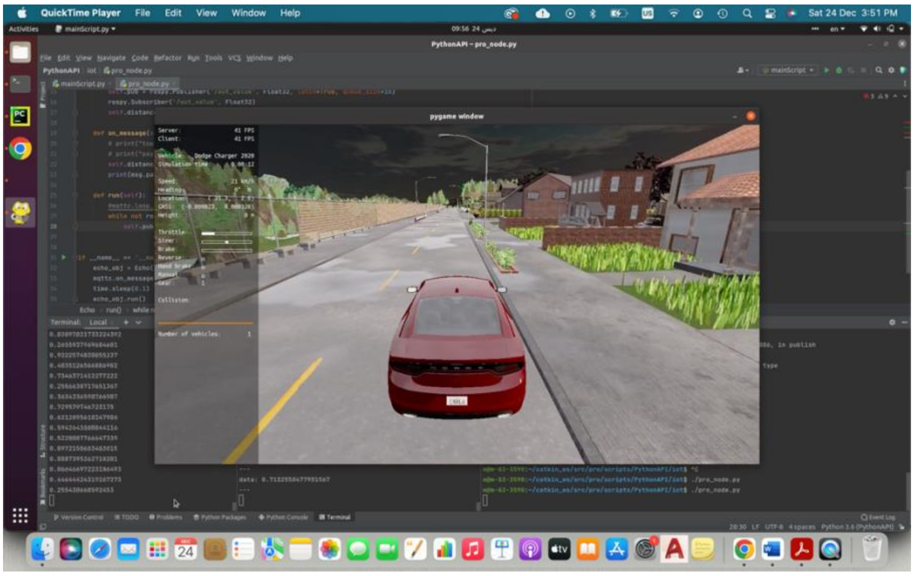


Figure 43: CARLA Simulation

3.4 Functional Mockup Interface (FMI)

A functional mockup interface (FMI) is a model that represents the dynamics of a system, and it can be used to simulate the response of the system to a given input. The FMI is a combination of Extensible Markup Language (XML) files, binaries and code [11]. In this project, an FMI representing the dynamics of the vehicle is needed to simulate the vehicle behavior. There are two main libraries in Python for working with FMIs: FMPy and PyFMI. The FMPy library is implemented mainly in Python, but the PyFMI library is a wrapper for the FMIL library implemented in C language. PyFMI has more advanced functionality that FMPy [12]. The term functional mockup unit (FMU) is used to describe the instance of the simulation object that is created using the functional mockup interface (FMI) [13]. There are two FMIs provided for vehicle dynamics: the complete ego vehicle FMU and the simplified braking system FMU. Following are the trials made to simulate the FMU in this project.

3.4.1 Trial 1: Using Ego Vehicle FMU

In the first trial, the aim was to use the ego vehicle dynamics to have an accurate representation of the vehicle dynamics. The following system specifications were used.

- Ubuntu 20.04.4 in VMware Workstation
- Python 3.9.13

Installation of the FMPy was done, and the code for simulating the ego vehicle FMU was run. After adding all the missing libraries to the path and adding the path to the bashrc file, a fatal error in simulation itself happens. The error is mainly caused by:

- The AME environment variable is undefined.

- There is no server list for AME_LIC_INIT

These two errors caused the simulation to fail as shown in Figure 44 and Figure 45.

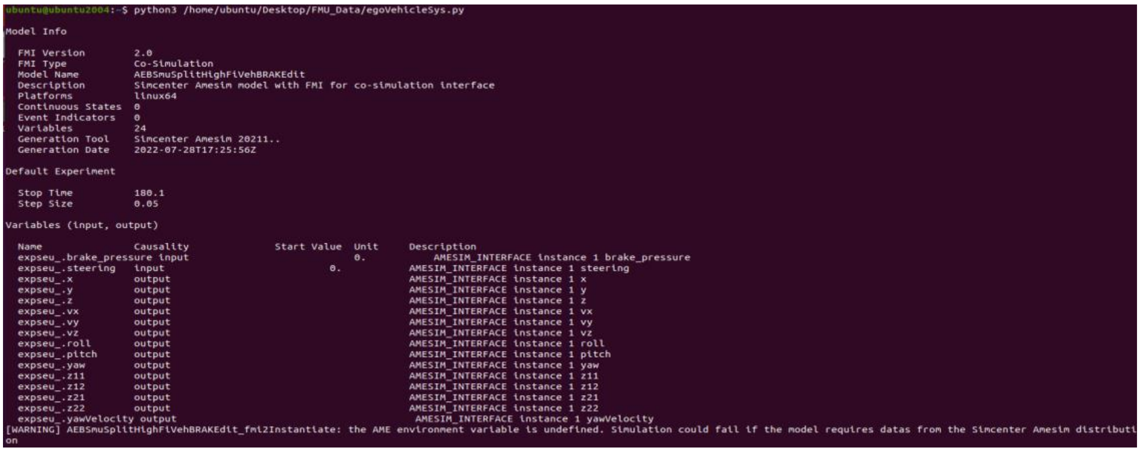


Figure 44: Error message for trial 1 (part 1)

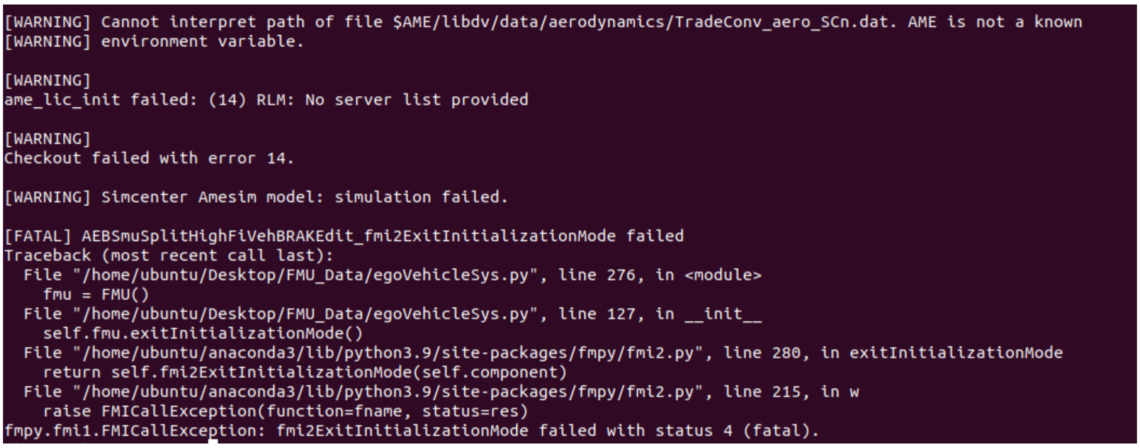


Figure 45: Error message for trial 1 (part 2)

3.4.2 Trial 2: Using Braking System FMU

In the next trial, the aim was to use the simplified model to avoid the complexities of the ego vehicle dynamics. First, there was multiple trials to install the PyFMI library. The trials fails due to a problem in the package as shown in Figure 46.

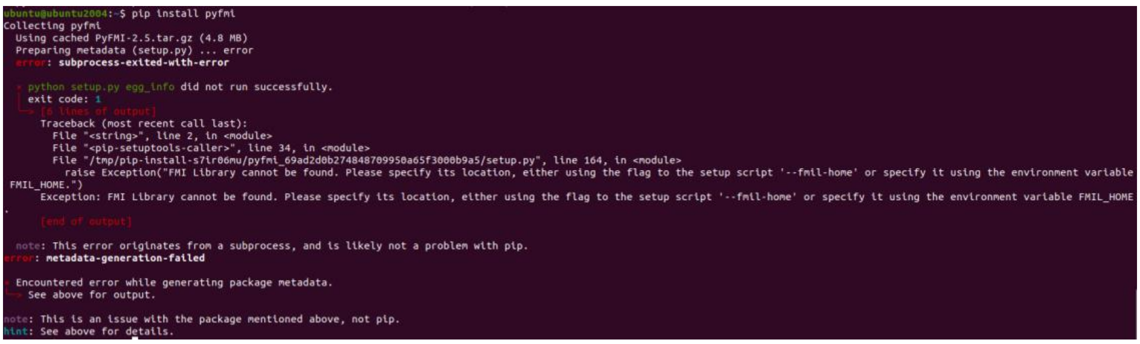


Figure 46: Error in installing PyFMI library

Accordingly, the code of the braking system simulation is studied and the use of PyFMI library was eliminated. The FMPy was used instead to simulate the FMU. However, the same error of trial 1 appears with trial 2, although the FMU is changed as shown in Figure 47 and Figure 48.

```
ubuntu@ubuntu2004:~$ python3 /home/ubuntu/Desktop/FMU_Data/brakingSys.py

Model Info

  FMI Version        2.0
  FMI Type           Co-Simulation
  Model Name         BrakingSystem
  Description        Simcenter Amesim model with FMI for co-simulation interface
  Platforms          linux64
  Continuous States  0
  Event Indicators   0
  Variables          13
  Generation Tool    Simcenter Amesim 20211..
  Generation Date    2022-04-07T17:24:53Z

Default Experiment

  Stop Time          1.0
  Step Size          0.001

Variables (input, output)

  Name               Causality              Start Value  Unit     Description
  expseu_.Force      input                           0.           expseu instance 1 Force
  expseu_.Force_Front_Right output                                       expseu instance 1 Force_Front_Right
  expseu_.Force_Rear_Right output                                       expseu instance 1 Force_Rear_Right
  expseu_.Force_Front_Left output                                       expseu instance 1 Force_Front_Left
  expseu_.Force_Rear_Left output                                       expseu instance 1 Force_Rear_Left
{'expseu_.Force': 134217728, 'expseu_.Force_Front_Right': 268435456, 'expseu_.Force_Rear_Right': 268435457, 'expseu_.Force_Fron
'run_parameters.maxTimeStep': 536870912, 'run_parameters.tolerance': 536870913, 'run_parameters.fixedStep': 536870914, 'run_par
ype': 536870913, 'run_parameters.solverType': 536870914, 'run_parameters.integrationMethod': 536870915, 'run_parameters.fixedOr
/home/ubuntu/Desktop/FMU_Data/brakingSys.py:20: DeprecationWarning: `np.float` is a deprecated alias for the builtin `float`. T
will not modify any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  dtype = [('expseu_.Force', np.float)]
[WARNING] BrakingSystem_fmi2Instantiate: the AME environment variable is undefined. Simulation could fail if the model requires
[WARNING]
ame_lic_init failed: (14) RLM: No server list provided

[WARNING]
Checkout failed with error 14.
```

Figure 47: Error in trial 2 (part 1)

```
ame_lic_init failed: (14) RLM: No server list provided

[WARNING]
Checkout failed with error 14.

[WARNING] Simcenter Amesim model: initialization failed.

[FATAL] BrakingSystem_fmi2ExitInitializationMode failed
Traceback (most recent call last):
  File "/home/ubuntu/Desktop/FMU_Data/brakingSys.py", line 60, in <module>
    pushing_into_file(fmu("path_of_fmu","force or list of forces"))
  File "/home/ubuntu/Desktop/FMU_Data/brakingSys.py", line 22, in fmu
    result = simulate_fmu(fmu, stop_time=0.5, input=signals)
  File "/home/ubuntu/anaconda3/lib/python3.9/site-packages/fmpy/simulation.py", line 758, in simulate_fmu
    result = simulateCS(model_description, fmu, start_time, stop_time, relative_tolerance, start_values, apply_default_st
ed, set_input_derivatives, use_event_mode, early_return_allowed, validate, initialize, terminate)
  File "/home/ubuntu/anaconda3/lib/python3.9/site-packages/fmpy/simulation.py", line 1199, in simulateCS
    fmu.exitInitializationMode()
  File "/home/ubuntu/anaconda3/lib/python3.9/site-packages/fmpy/fmi2.py", line 280, in exitInitializationMode
    return self.fmi2ExitInitializationMode(self.component)
  File "/home/ubuntu/anaconda3/lib/python3.9/site-packages/fmpy/fmi2.py", line 215, in w
    raise FMICallException(function=fname, status=res)
fmpy.fmi1.FMICallException: fmi2ExitInitializationMode failed with status 4 (fatal).
```

Figure 48: Error in trial 2 (part 2)

3.4.2.1 Comment on results:

At that point, it was apparent that the problem in both FMUs is related to the license. Both FMUs are Simcenter Amesim models that cannot be simulated without having all the licenses and related datasets. Therefore, a shift in the procedure was done. The aim was to create a simple FMU from scratch to simulate it later on as shown in Trial 3.

3.4.3 Trial 3: Creating an FMU

Based on internet research, it was found that Simulink can be used to export standalone FMUs [14]. However, this option is not possible in the version of MATLAB installed and that capability is not just a simple add-on that can be directly installed. Therefore, although this option is possible, it was not suitable in our case.

3.4.4 Trial 4: Simulating an FMU

Finally, to gain insights about simulating FMUs, an FMU was found online to help in testing [15]. Simulation of such FMU was done using the graphical user interface (GUI) of the FMPy library. To run the GUI, the following command is written in the command window, and some python libraries had to be installed using the pip command.

```
python -m fmpy.gui
```

Figure 49 shows the GUI of FMPy library. Simulating the model using the default settings resulted in the behavior shown in Figure 50. Therefore, the idea behind using FMUs and Python libraries becomes apparent.

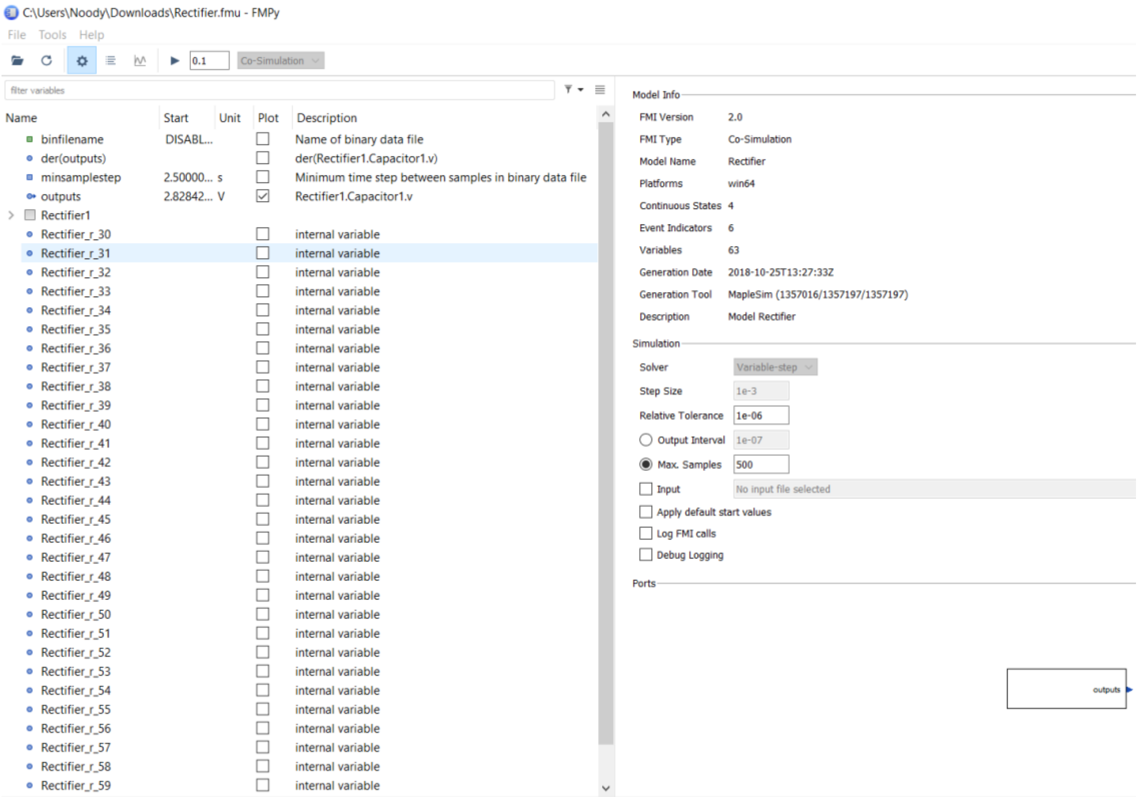


Figure 49: Graphical User Interface of FMPy Library

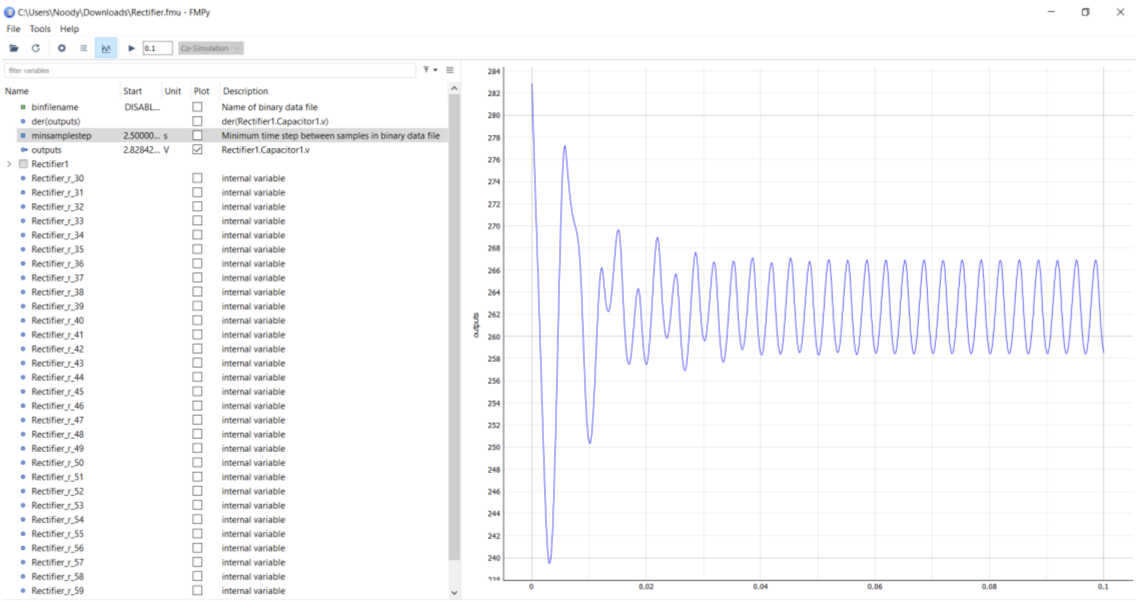
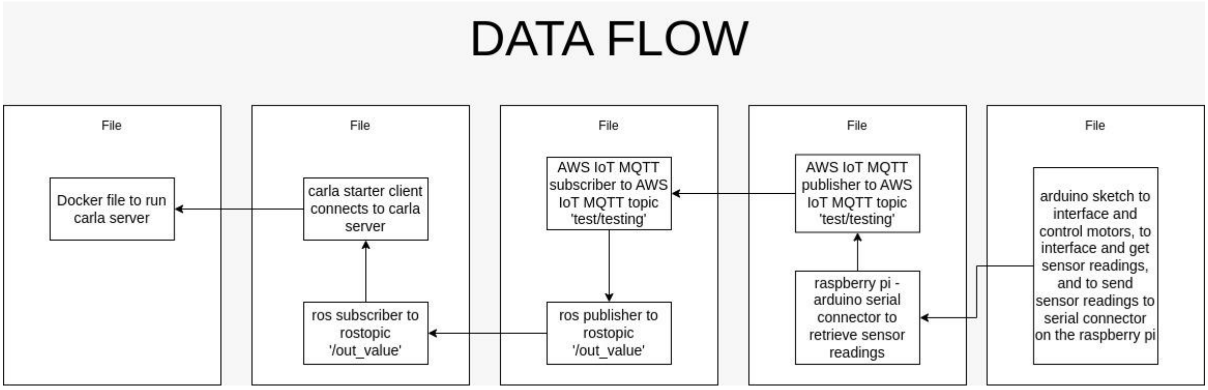


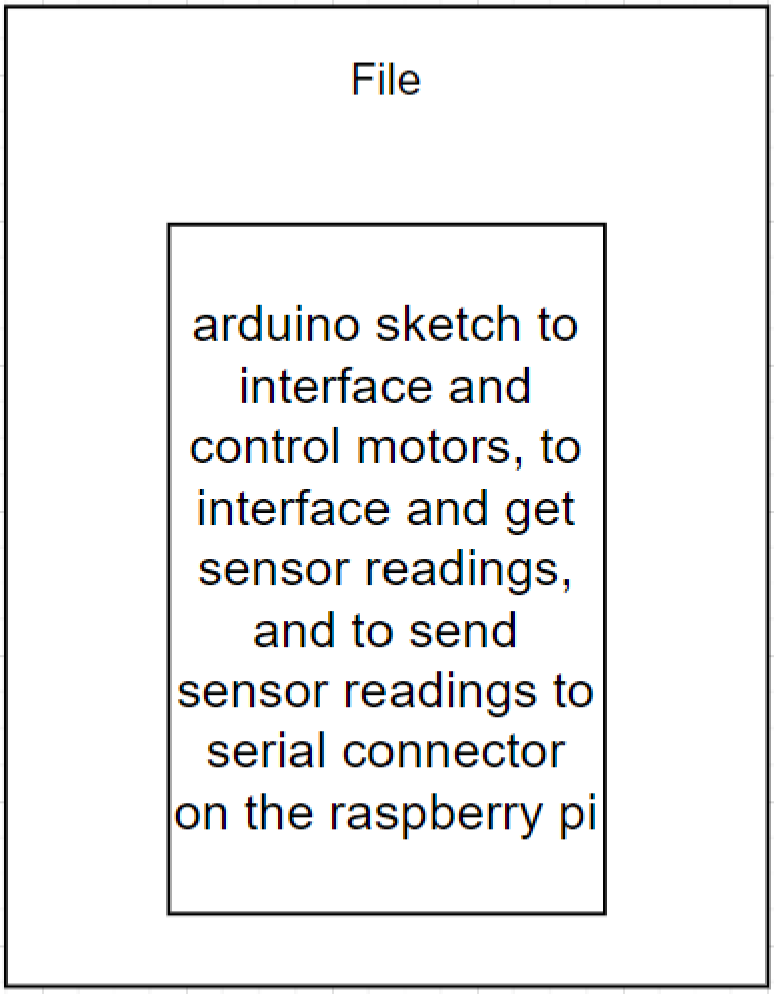
Figure 50: Simulation results of a sample FMU to show the system response

INTEGRATING PHYSICAL AND CYBER SYSTEMS

4.1 Data Flow



1.



(file)

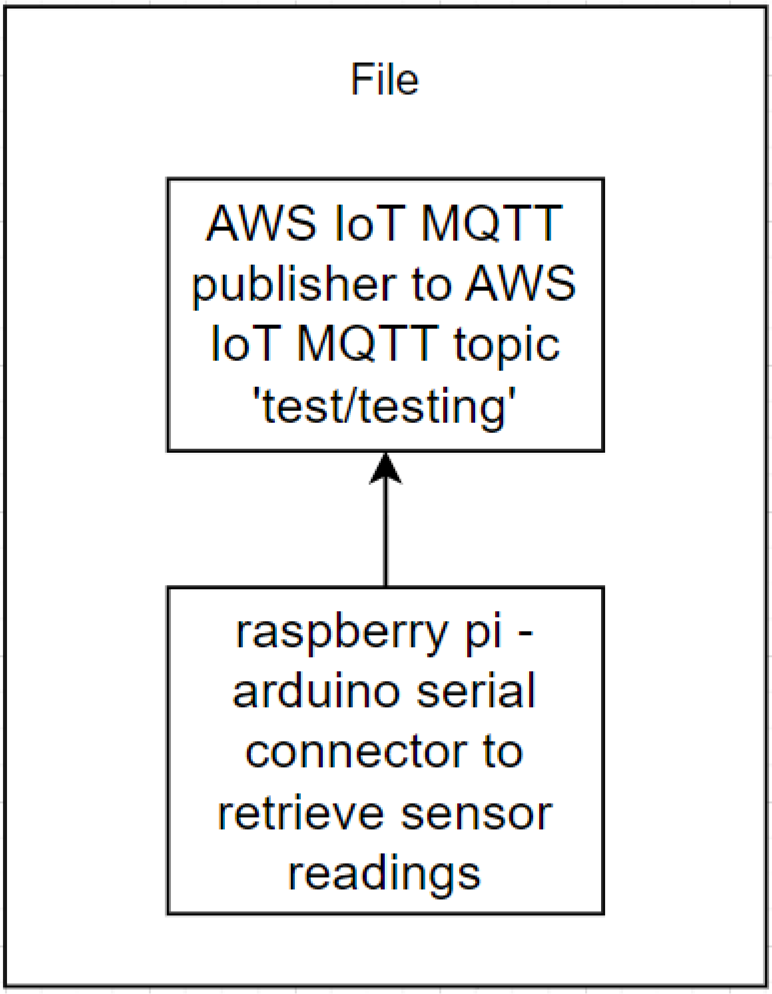
```
#include <AFMotor.h> // Motor Controller Library


AF_DCMotor motor1(1);
AF_DCMotor motor2(2);
AF_DCMotor motor3(3);
```

```
AF_DCMotor motor4(4);

#define echoPin 16 // attach pin D2 Arduino to pin Echo of HC-SR04
#define trigPin 17 //attach pin D3 Arduino to pin Trig of HC-SR04


int distance;
int min_distance;
int x= 0;
long t1;
long t2;
long duration; // variable for the duration of sound wave travel
int distanceU; // variable for the distance measurement


int ultrasonic()
{
  // Clears the trigPin condition
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distanceU = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and
back)
  // Displays the distance on the Serial Monitor
  //Serial.print("Distance: ");
  //Serial.print(distanceU);
  //Serial.println(" cm");

  return distanceU;

}



void moveForward() {
  motor1.setSpeed(200);
  motor2.setSpeed(200);
  motor3.setSpeed(200);
  motor4.setSpeed(200);
  motor1.run(FORWARD);
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
}
void moveBackward() {
```

```
  motor1.setSpeed(200);
  motor2.setSpeed(200);
  motor3.setSpeed(200);
  motor4.setSpeed(200);
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
}

void Stop() {
  motor1.run(RELEASE);
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
}

void setup() {
  Serial.begin(9600);
  min_distance = 5;

  // Ultrasonic setup
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT
  Serial.begin(9600); // // Serial Communication is starting with 9600 of baudrate
speed
  //Serial.println("Ultrasonic Sensor HC-SR04 Test"); // print some text in Serial
Monitor
  //Serial.println("with Arduino UNO R3");

}

void loop() {
//t1 = millis();
distance = ultrasonic();


  if (distance <= min_distance)
  {
    Stop();

    Serial.println(distance);
    //Serial.write('\n');
    //Serial.write("stop");
    //Serial.write('\n');
    //delay ();

  }
  else
  {
```

```
    moveForward();
    Serial.println(distance);
    //Serial.write('\n');
    //Serial.write("moving");
    //Serial.write('\n');
    //delay (3000);

  }
delay(1500);
//t2 = millis();


}
```

2.

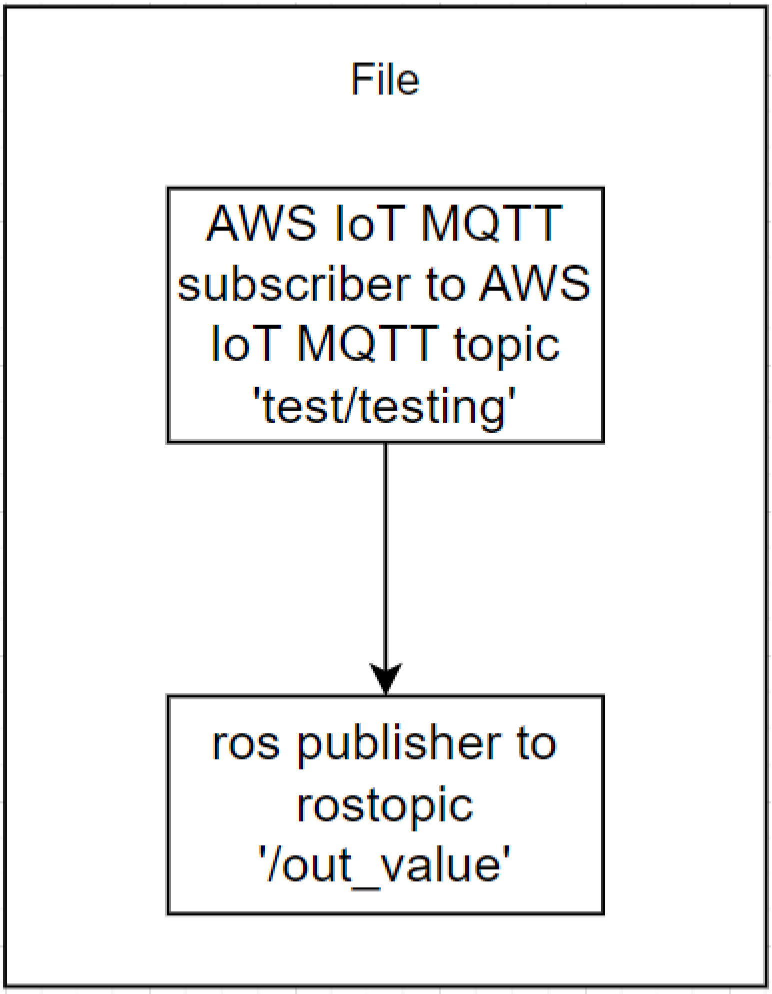


(file)

```
#!/usr/bin/env python3
import serial
import json
import time
import AWSIoTPythonSDK.MQTTLib as AWSIoTPyMQTT
```

```python
# Define ENDPOINT, CLIENT_ID, PATH_TO_CERTIFICATE, PATH_TO_PRIVATE_KEY,
PATH_TO_AMAZON_ROOT_CA_1, MESSAGE, TOPIC, and RANGE
ENDPOINT = "avsnsxi1w2nv8-ats.iot.me-central-1.amazonaws.com"
CLIENT_ID = "iot_thing"
PATH_TO_CERTIFICATE =
"d42194e6a78b821823451cc7b1d29896291e00dffd00576f6197250175f33b2b-
certificate.pem.crt"
PATH_TO_PRIVATE_KEY =
"d42194e6a78b821823451cc7b1d29896291e00dffd00576f6197250175f33b2b-private.pem.key"
PATH_TO_AMAZON_ROOT_CA_1 = "AmazonRootCA1.pem"
TOPIC = "test/testing"

myAWSIoTMQTTClient = AWSIoTPyMQTT.AWSIoTMQTTClient(CLIENT_ID)
myAWSIoTMQTTClient.configureEndpoint(ENDPOINT, 8883)
myAWSIoTMQTTClient.configureCredentials(PATH_TO_AMAZON_ROOT_CA_1,
PATH_TO_PRIVATE_KEY, PATH_TO_CERTIFICATE)
myAWSIoTMQTTClient.configureMQTTOperationTimeout(1000)
myAWSIoTMQTTClient.connect()

ser = serial.Serial('/dev/ttyACM0', 9600, timeout=2)
ser.reset_input_buffer()


def sr():
    reading = ser.readline().decode()
    reading = reading.strip()
    distance = int(reading) if len(reading) > 0 else 0
    return distance


def publish_to_cloud(msg):
    myAWSIoTMQTTClient.publish(TOPIC, json.dumps(msg), 1)


if __name__ == '__main__':
    counter = 0
    while True:
        distance = sr()
        publish_to_cloud(distance)
        #time.sleep(1)
        print(distance)
```

3.



(file – part1)

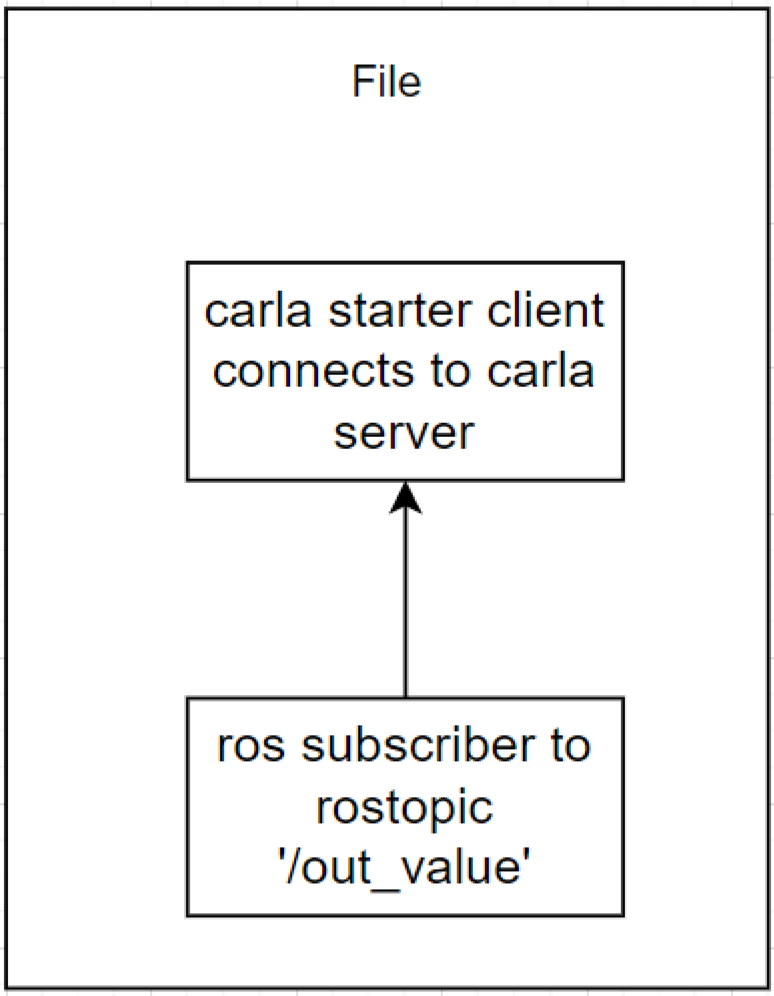
```
#!/usr/bin/env python3.7

import time
import sys
sys.path.append("/home/m/catkin_ws/src/pro/scripts/PythonAPI/iot")
import rospy
from std_msgs.msg import Float32
from subscribe import mqttc


class Echo(object):
    def __init__(self):
        rospy.init_node('echoer')
        self.pub = rospy.Publisher('/out_value', Float32, latch=True,
queue_size=10)
        rospy.Subscriber('/out_value', Float32)
        self.distance = 0.2

    def on_message(self, client, userdata, msg):  # Func for receiving msgs
        # print("topic: " + msg.topic)
        # print("payload: " + str(msg.payload))
        self.distance = int(msg.payload.decode("utf-8"))
```

```
        print(msg.payload, self.distance)

    def run(self):
        mqttc.loop_start()
        while not rospy.is_shutdown():
            self.pub.publish(self.distance)


if __name__ == '__main__':
    echo_obj = Echo()
    mqttc.on_message = echo_obj.on_message  # assign on_message func
    time.sleep(0.1)
    echo_obj.run()
```

(file – part2)

```
#!/usr/bin/env python3.7
import ssl
import paho.mqtt.client as paho


awshost = "avsnsxi1w2nv8-ats.iot.me-central-1.amazonaws.com"  # Endpoint
awsport = 8883  # Port no.
clientId = "iot_thing"  # Thing_Name
thingName = "iot_thing"  # Thing_Name
caPath = "AmazonRootCA1.pem"  # Root_CA_Certificate_Name
certPath = "d42194e6a78b821823451cc7b1d29896291e00dffd00576f6197250175f33b2b-
certificate.pem.crt"  # <Thing_Name>.cert.pem
keyPath = "d42194e6a78b821823451cc7b1d29896291e00dffd00576f6197250175f33b2b-
private.pem.key"  # <Thing_Name>.private.key
TOPIC = "test/testing"


def on_connect(client, userdata, flags, rc):  # func for making connection
    print("Connection returned result: " + str(rc))
    # Subscribing in on_connect() means that if we lose the connection and
    # reconnect then subscriptions will be renewed.
    client.subscribe(TOPIC, 1)  # Subscribe to all topics


# def on_log(client, userdata, level, msg):
#    print(msg.topic+" "+str(msg.payload))

mqttc = paho.Client()  # mqttc object
mqttc.on_connect = on_connect  # assign on_connect func
# mqttc.on_log = on_log

mqttc.tls_set(caPath, certfile=certPath, keyfile=keyPath,
cert_reqs=ssl.CERT_REQUIRED, tls_version=ssl.PROTOCOL_TLSv1_2, ciphers=None)  #
pass parameters
mqttc.connect(awshost, awsport, keepalive=60)  # connect to aws server
#mqttc.loop_start()
```

4.



(file)

```
#!/usr/bin/env python3.7

# Copyright (c) 2018 Intel Labs.
# authors: German Ros (german.ros@intel.com)
#
# This work is licensed under the terms of the MIT license.
# For a copy, see <https://opensource.org/licenses/MIT>.

"""Example of automatic vehicle control from client side."""

from __future__ import print_function

import argparse
import collections
import datetime
import glob
import logging
import math
import os
import numpy.random as random
import re
```

```
import sys
import weakref
import rospy
from std_msgs.msg import Float32

try:
    import pygame
    from pygame.locals import KMOD_CTRL
    from pygame.locals import K_ESCAPE
    from pygame.locals import K_q
except ImportError:
    raise RuntimeError('cannot import pygame, make sure pygame package is
installed')

try:
    import numpy as np
except ImportError:
    raise RuntimeError(
        'cannot import numpy, make sure numpy package is installed')

# ==============================================================================
# -- Find CARLA module ---------------------------------------------------------
# ==============================================================================
try:
    sys.path.append(glob.glob('../carla/dist/carla-*%d.%d-%s.egg' % (
        sys.version_info.major,
        sys.version_info.minor,
        'win-amd64' if os.name == 'nt' else 'linux-x86_64'))[0])
except IndexError:
    pass

# ==============================================================================
# -- Add PythonAPI for release mode --------------------------------------------
# ==============================================================================
try:
    sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))) +
'/carla')
except IndexError:
    pass

import carla
from carla import ColorConverter as cc

from agents.navigation.behavior_agent import BehaviorAgent  # pylint:
disable=import-error
from agents.navigation.basic_agent import BasicAgent  # pylint: disable=import-
error


# ==============================================================================
```

```python
# -- Global functions ----------------------------------------------------------
# ==============================================================================


def find_weather_presets():
    """Method to find weather presets"""
    rgx = re.compile('.+?(?:(?<=[a-z])(?=[A-Z])|(?<=[A-Z])(?=[A-Z][a-z])|$)')
    def name(x): return ' '.join(m.group(0) for m in rgx.finditer(x))
    presets = [x for x in dir(carla.WeatherParameters) if re.match('[A-Z].+', x)]
    return [(getattr(carla.WeatherParameters, x), name(x)) for x in presets]


def get_actor_display_name(actor, truncate=250):
    """Method to get actor display name"""
    name = ' '.join(actor.type_id.replace('_', '.').title().split('.')[1:])
    return (name[:truncate - 1] + u'\u2026') if len(name) > truncate else name


# ==============================================================================
# -- World ---------------------------------------------------------------------
# ==============================================================================

class World(object):
    """ Class representing the surrounding environment """

    def __init__(self, client, carla_world, hud, args):
        """Constructor method"""
        self._args = args
        self.world = carla_world
        try:
            #self.map = self.world.get_map()
            self.map = client.load_world('Town01_Opt')
        except RuntimeError as error:
            print('RuntimeError: {}'.format(error))
            print('  The server could not send the OpenDRIVE (.xodr) file:')
            print('  Make sure it exists, has the same name of your town, and is 
correct.')
            sys.exit(1)
        self.hud = hud
        self.player = None
        self.collision_sensor = None
        self.lane_invasion_sensor = None
        self.gnss_sensor = None
        self.camera_manager = None
        self._weather_presets = find_weather_presets()
        self._weather_index = 0
        self._actor_filter = args.filter
        self.restart(args)
        self.world.on_tick(hud.on_world_tick)
        self.recording_enabled = False
        self.recording_start = 0
```

```
    def restart(self, args):
        """Restart the world"""
        # Keep same camera config if the camera manager exists.
        cam_index = self.camera_manager.index if self.camera_manager is not None
else 0
        cam_pos_id = self.camera_manager.transform_index if self.camera_manager is
not None else 0

        # Get a random blueprint.
        blueprint = self.world.get_blueprint_library().filter('charger_2020')[0]
        blueprint.set_attribute('role_name', 'hero')
        if blueprint.has_attribute('color'):
            color = blueprint.get_attribute('color').recommended_values[0]
            blueprint.set_attribute('color', color)

        # Spawn the player.
        if self.player is not None:
            spawn_point = self.player.get_transform()
            spawn_point.location.z += 2.0
            spawn_point.rotation.roll = 0.0
            spawn_point.rotation.pitch = 0.0
            self.destroy()
            self.player = self.world.try_spawn_actor(blueprint, spawn_point)
            self.modify_vehicle_physics(self.player)
        while self.player is None:
            spawn_point = carla.Transform()
            spawn_point.location.x = 10.868797
            spawn_point.location.y = 2.461965
            spawn_point.location.z = 0.3
            spawn_point.rotation.pitch = 0
            spawn_point.rotation.yaw = 0
            spawn_point.rotation.roll = 0

            self.player = self.world.try_spawn_actor(blueprint, spawn_point)
            self.modify_vehicle_physics(self.player)

        if self._args.sync:
            self.world.tick()
        else:
            self.world.wait_for_tick()

        # Set up the sensors.
        self.collision_sensor = CollisionSensor(self.player, self.hud)
        self.lane_invasion_sensor = LaneInvasionSensor(self.player, self.hud)
        self.gnss_sensor = GnssSensor(self.player)
        self.camera_manager = CameraManager(self.player, self.hud)
        self.camera_manager.transform_index = cam_pos_id
        self.camera_manager.set_sensor(cam_index, notify=False)
        actor_type = get_actor_display_name(self.player)
```

```python
        self.hud.notification(actor_type)

    def next_weather(self, reverse=False):
        """Get next weather setting"""
        self._weather_index += -1 if reverse else 1
        self._weather_index %= len(self._weather_presets)
        preset = self._weather_presets[self._weather_index]
        self.hud.notification('Weather: %s' % preset[1])
        self.player.get_world().set_weather(preset[0])

    def modify_vehicle_physics(self, actor):
        #If actor is not a vehicle, we cannot use the physics control
        try:
            physics_control = actor.get_physics_control()
            physics_control.use_sweep_wheel_collision = True
            actor.apply_physics_control(physics_control)
        except Exception:
            pass

    def tick(self, clock):
        """Method for every tick"""
        self.hud.tick(self, clock)

    def render(self, display):
        """Render world"""
        self.camera_manager.render(display)
        self.hud.render(display)

    def destroy_sensors(self):
        """Destroy sensors"""
        self.camera_manager.sensor.destroy()
        self.camera_manager.sensor = None
        self.camera_manager.index = None

    def destroy(self):
        """Destroys all actors"""
        actors = [
            self.camera_manager.sensor,
            self.collision_sensor.sensor,
            self.lane_invasion_sensor.sensor,
            self.gnss_sensor.sensor,
            self.player]
        for actor in actors:
            if actor is not None:
                actor.destroy()


# ==============================================================================
# -- KeyboardControl -----------------------------------------------------------
# ==============================================================================
```

```
class KeyboardControl(object):
    def __init__(self, world):
        world.hud.notification("Press 'H' or '?' for help.", seconds=4.0)

    def parse_events(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                return True
            if event.type == pygame.KEYUP:
                if self._is_quit_shortcut(event.key):
                    return True

    @staticmethod
    def _is_quit_shortcut(key):
        """Shortcut for quitting"""
        return (key == K_ESCAPE) or (key == K_q and pygame.key.get_mods() &
KMOD_CTRL)

# ==============================================================================
# -- HUD -----------------------------------------------------------------------
# ==============================================================================


class HUD(object):
    """Class for HUD text"""

    def __init__(self, width, height):
        """Constructor method"""
        self.dim = (width, height)
        font = pygame.font.Font(pygame.font.get_default_font(), 20)
        font_name = 'courier' if os.name == 'nt' else 'mono'
        fonts = [x for x in pygame.font.get_fonts() if font_name in x]
        default_font = 'ubuntumono'
        mono = default_font if default_font in fonts else fonts[0]
        mono = pygame.font.match_font(mono)
        self._font_mono = pygame.font.Font(mono, 12 if os.name == 'nt' else 14)
        self._notifications = FadingText(font, (width, 40), (0, height - 40))
        self.help = HelpText(pygame.font.Font(mono, 24), width, height)
        self.server_fps = 0
        self.frame = 0
        self.simulation_time = 0
        self._show_info = True
        self._info_text = []
        self._server_clock = pygame.time.Clock()

    def on_world_tick(self, timestamp):
        """Gets informations from the world at every tick"""
        self._server_clock.tick()
```

```
        self.server_fps = self._server_clock.get_fps()
        self.frame = timestamp.frame_count
        self.simulation_time = timestamp.elapsed_seconds

    def tick(self, world, clock):
        """HUD method for every tick"""
        self._notifications.tick(world, clock)
        if not self._show_info:
            return
        transform = world.player.get_transform()
        vel = world.player.get_velocity()

        control = world.player.get_control()
        heading = 'N' if abs(transform.rotation.yaw) < 89.5 else ''
        heading += 'S' if abs(transform.rotation.yaw) > 90.5 else ''
        heading += 'E' if 179.5 > transform.rotation.yaw > 0.5 else ''
        heading += 'W' if -0.5 > transform.rotation.yaw > -179.5 else ''
        colhist = world.collision_sensor.get_collision_history()
        collision = [colhist[x + self.frame - 200] for x in range(0, 200)]
        max_col = max(1.0, max(collision))
        collision = [x / max_col for x in collision]
        vehicles = world.world.get_actors().filter('vehicle.*')

        self._info_text = [
            'Server:  % 16.0f FPS' % self.server_fps,
            'Client:  % 16.0f FPS' % clock.get_fps(),
            '',
            'Vehicle: % 20s' % get_actor_display_name(world.player, truncate=20),
            #'Map:     % 20s' % world.map.name.split('/')[-1],
            'Simulation time: % 12s' %
datetime.timedelta(seconds=int(self.simulation_time)),
            '',
            'Speed:   % 15.0f km/h' % (3.6 * math.sqrt(vel.x**2 + vel.y**2 +
vel.z**2)),
            u'Heading:% 16.0f\N{DEGREE SIGN} % 2s' % (transform.rotation.yaw,
heading),
            'Location:% 20s' % ('(% 5.1f, % 5.1f)' % (transform.location.x,
transform.location.y)),
            'GNSS:% 24s' % ('(% 2.6f, % 3.6f)' % (world.gnss_sensor.lat,
world.gnss_sensor.lon)),
            'Height:  % 18.0f m' % transform.location.z,
            '']
        if isinstance(control, carla.VehicleControl):
            self._info_text += [
                ('Throttle:', control.throttle, 0.0, 1.0),
                ('Steer:', control.steer, -1.0, 1.0),
                ('Brake:', control.brake, 0.0, 1.0),
                ('Reverse:', control.reverse),
                ('Hand brake:', control.hand_brake),
                ('Manual:', control.manual_gear_shift),
```

```
                'Gear:        %s' % {-1: 'R', 0: 'N'}.get(control.gear,
control.gear)]
        elif isinstance(control, carla.WalkerControl):
            self._info_text += [
                ('Speed:', control.speed, 0.0, 5.556),
                ('Jump:', control.jump)]
        self._info_text += [
            '',
            'Collision:',
            collision,
            '',
            'Number of vehicles: % 8d' % len(vehicles)]

        if len(vehicles) > 1:
            self._info_text += ['Nearby vehicles:']

        def dist(l):
            return math.sqrt((l.x - transform.location.x)**2 + (l.y -
transform.location.y)
                             ** 2 + (l.z - transform.location.z)**2)
        vehicles = [(dist(x.get_location()), x) for x in vehicles if x.id !=
world.player.id]

        for dist, vehicle in sorted(vehicles):
            if dist > 200.0:
                break
            vehicle_type = get_actor_display_name(vehicle, truncate=22)
            self._info_text.append('% 4dm %s' % (dist, vehicle_type))

    def toggle_info(self):
        """Toggle info on or off"""
        self._show_info = not self._show_info

    def notification(self, text, seconds=2.0):
        """Notification text"""
        self._notifications.set_text(text, seconds=seconds)

    def error(self, text):
        """Error text"""
        self._notifications.set_text('Error: %s' % text, (255, 0, 0))

    def render(self, display):
        """Render for HUD class"""
        if self._show_info:
            info_surface = pygame.Surface((220, self.dim[1]))
            info_surface.set_alpha(100)
            display.blit(info_surface, (0, 0))
            v_offset = 4
            bar_h_offset = 100
            bar_width = 106
```

```
            for item in self._info_text:
                if v_offset + 18 > self.dim[1]:
                    break
                if isinstance(item, list):
                    if len(item) > 1:
                        points = [(x + 8, v_offset + 8 + (1 - y) * 30) for x, y in
enumerate(item)]
                        pygame.draw.lines(display, (255, 136, 0), False, points, 2)
                    item = None
                    v_offset += 18
                elif isinstance(item, tuple):
                    if isinstance(item[1], bool):
                        rect = pygame.Rect((bar_h_offset, v_offset + 8), (6, 6))
                        pygame.draw.rect(display, (255, 255, 255), rect, 0 if
item[1] else 1)
                    else:
                        rect_border = pygame.Rect((bar_h_offset, v_offset + 8),
(bar_width, 6))
                        pygame.draw.rect(display, (255, 255, 255), rect_border, 1)
                        fig = (item[1] - item[2]) / (item[3] - item[2])
                        if item[2] < 0.0:
                            rect = pygame.Rect(
                                (bar_h_offset + fig * (bar_width - 6), v_offset +
8), (6, 6))
                        else:
                            rect = pygame.Rect((bar_h_offset, v_offset + 8), (fig *
bar_width, 6))
                        pygame.draw.rect(display, (255, 255, 255), rect)
                    item = item[0]
                if item:  # At this point has to be a str.
                    surface = self._font_mono.render(item, True, (255, 255, 255))
                    display.blit(surface, (8, v_offset))
                v_offset += 18
        self._notifications.render(display)
        self.help.render(display)

# ==============================================================================
# -- FadingText ----------------------------------------------------------------
# ==============================================================================


class FadingText(object):
    """ Class for fading text """

    def __init__(self, font, dim, pos):
        """Constructor method"""
        self.font = font
        self.dim = dim
        self.pos = pos
        self.seconds_left = 0
```

```python
        self.surface = pygame.Surface(self.dim)

    def set_text(self, text, color=(255, 255, 255), seconds=2.0):
        """Set fading text"""
        text_texture = self.font.render(text, True, color)
        self.surface = pygame.Surface(self.dim)
        self.seconds_left = seconds
        self.surface.fill((0, 0, 0, 0))
        self.surface.blit(text_texture, (10, 11))

    def tick(self, _, clock):
        """Fading text method for every tick"""
        delta_seconds = 1e-3 * clock.get_time()
        self.seconds_left = max(0.0, self.seconds_left - delta_seconds)
        self.surface.set_alpha(500.0 * self.seconds_left)

    def render(self, display):
        """Render fading text method"""
        display.blit(self.surface, self.pos)

# ==============================================================================
# -- HelpText ------------------------------------------------------------------
# ==============================================================================


class HelpText(object):
    """ Helper class for text render"""

    def __init__(self, font, width, height):
        """Constructor method"""
        lines = __doc__.split('\n')
        self.font = font
        self.dim = (680, len(lines) * 22 + 12)
        self.pos = (0.5 * width - 0.5 * self.dim[0], 0.5 * height - 0.5 *
self.dim[1])
        self.seconds_left = 0
        self.surface = pygame.Surface(self.dim)
        self.surface.fill((0, 0, 0, 0))
        for i, line in enumerate(lines):
            text_texture = self.font.render(line, True, (255, 255, 255))
            self.surface.blit(text_texture, (22, i * 22))
            self._render = False
        self.surface.set_alpha(220)

    def toggle(self):
        """Toggle on or off the render help"""
        self._render = not self._render

    def render(self, display):
        """Render help text method"""
```

```
        if self._render:
            display.blit(self.surface, self.pos)

# ==============================================================================
# -- CollisionSensor -----------------------------------------------------------
# ==============================================================================


class CollisionSensor(object):
    """ Class for collision sensors"""

    def __init__(self, parent_actor, hud):
        """Constructor method"""
        self.sensor = None
        self.history = []
        self._parent = parent_actor
        self.hud = hud
        world = self._parent.get_world()
        blueprint = world.get_blueprint_library().find('sensor.other.collision')
        self.sensor = world.spawn_actor(blueprint, carla.Transform(),
attach_to=self._parent)
        # We need to pass the lambda a weak reference to
        # self to avoid circular reference.
        weak_self = weakref.ref(self)
        self.sensor.listen(lambda event: CollisionSensor._on_collision(weak_self,
event))

    def get_collision_history(self):
        """Gets the history of collisions"""
        history = collections.defaultdict(int)
        for frame, intensity in self.history:
            history[frame] += intensity
        return history

    @staticmethod
    def _on_collision(weak_self, event):
        """On collision method"""
        self = weak_self()
        if not self:
            return
        actor_type = get_actor_display_name(event.other_actor)
        self.hud.notification('Collision with %r' % actor_type)
        impulse = event.normal_impulse
        intensity = math.sqrt(impulse.x ** 2 + impulse.y ** 2 + impulse.z ** 2)
        self.history.append((event.frame, intensity))
        if len(self.history) > 4000:
            self.history.pop(0)

# ==============================================================================
# -- LaneInvasionSensor --------------------------------------------------------
```

```
# ==============================================================================


class LaneInvasionSensor(object):
    """Class for lane invasion sensors"""

    def __init__(self, parent_actor, hud):
        """Constructor method"""
        self.sensor = None
        self._parent = parent_actor
        self.hud = hud
        world = self._parent.get_world()
        bp = world.get_blueprint_library().find('sensor.other.lane_invasion')
        self.sensor = world.spawn_actor(bp, carla.Transform(),
attach_to=self._parent)
        # We need to pass the lambda a weak reference to self to avoid circular
        # reference.
        weak_self = weakref.ref(self)
        self.sensor.listen(lambda event: LaneInvasionSensor._on_invasion(weak_self,
event))

    @staticmethod
    def _on_invasion(weak_self, event):
        """On invasion method"""
        self = weak_self()
        if not self:
            return
        lane_types = set(x.type for x in event.crossed_lane_markings)
        text = ['%r' % str(x).split()[-1] for x in lane_types]
        self.hud.notification('Crossed line %s' % ' and '.join(text))

# ==============================================================================
# -- GnssSensor ----------------------------------------------------------------
# ==============================================================================


class GnssSensor(object):
    """ Class for GNSS sensors"""

    def __init__(self, parent_actor):
        """Constructor method"""
        self.sensor = None
        self._parent = parent_actor
        self.lat = 0.0
        self.lon = 0.0
        world = self._parent.get_world()
        blueprint = world.get_blueprint_library().find('sensor.other.gnss')
        self.sensor = world.spawn_actor(blueprint,
carla.Transform(carla.Location(x=1.0, z=2.8)),
                                        attach_to=self._parent)
```

```
        # We need to pass the lambda a weak reference to
        # self to avoid circular reference.
        weak_self = weakref.ref(self)
        self.sensor.listen(lambda event: GnssSensor._on_gnss_event(weak_self,
event))

    @staticmethod
    def _on_gnss_event(weak_self, event):
        """GNSS method"""
        self = weak_self()
        if not self:
            return
        self.lat = event.latitude
        self.lon = event.longitude

# ==============================================================================
# -- CameraManager -------------------------------------------------------------
# ==============================================================================


class CameraManager(object):
    """ Class for camera management"""

    def __init__(self, parent_actor, hud):
        """Constructor method"""
        self.sensor = None
        self.surface = None
        self._parent = parent_actor
        self.hud = hud
        self.recording = False
        bound_y = 0.5 + self._parent.bounding_box.extent.y
        attachment = carla.AttachmentType
        self._camera_transforms = [
            (carla.Transform(
                carla.Location(x=-5.5, z=2.5), carla.Rotation(pitch=8.0)),
attachment.SpringArm),
            (carla.Transform(
                carla.Location(x=1.6, z=1.7)), attachment.Rigid),
            (carla.Transform(
                carla.Location(x=5.5, y=1.5, z=1.5)), attachment.SpringArm),
            (carla.Transform(
                carla.Location(x=-8.0, z=6.0), carla.Rotation(pitch=6.0)),
attachment.SpringArm),
            (carla.Transform(
                carla.Location(x=-1, y=-bound_y, z=0.5)), attachment.Rigid)]
        self.transform_index = 1
        self.sensors = [
            ['sensor.camera.rgb', cc.Raw, 'Camera RGB'],
            ['sensor.camera.depth', cc.Raw, 'Camera Depth (Raw)'],
            ['sensor.camera.depth', cc.Depth, 'Camera Depth (Gray Scale)'],
```

```
            ['sensor.camera.depth', cc.LogarithmicDepth, 'Camera Depth (Logarithmic
Gray Scale)'],
            ['sensor.camera.semantic_segmentation', cc.Raw, 'Camera Semantic
Segmentation (Raw)'],
            ['sensor.camera.semantic_segmentation', cc.CityScapesPalette,
             'Camera Semantic Segmentation (CityScapes Palette)'],
            ['sensor.lidar.ray_cast', None, 'Lidar (Ray-Cast)']]
        world = self._parent.get_world()
        bp_library = world.get_blueprint_library()
        for item in self.sensors:
            blp = bp_library.find(item[0])
            if item[0].startswith('sensor.camera'):
                blp.set_attribute('image_size_x', str(hud.dim[0]))
                blp.set_attribute('image_size_y', str(hud.dim[1]))
            elif item[0].startswith('sensor.lidar'):
                blp.set_attribute('range', '50')
            item.append(blp)
        self.index = None

    def toggle_camera(self):
        """Activate a camera"""
        self.transform_index = (self.transform_index + 1) %
len(self._camera_transforms)
        self.set_sensor(self.index, notify=False, force_respawn=True)

    def set_sensor(self, index, notify=True, force_respawn=False):
        """Set a sensor"""
        index = index % len(self.sensors)
        needs_respawn = True if self.index is None else (
            force_respawn or (self.sensors[index][0] !=
self.sensors[self.index][0]))
        if needs_respawn:
            if self.sensor is not None:
                self.sensor.destroy()
                self.surface = None
            self.sensor = self._parent.get_world().spawn_actor(
                self.sensors[index][-1],
                self._camera_transforms[self.transform_index][0],
                attach_to=self._parent,
                attachment_type=self._camera_transforms[self.transform_index][1])

            # We need to pass the lambda a weak reference to
            # self to avoid circular reference.
            weak_self = weakref.ref(self)
            self.sensor.listen(lambda image: CameraManager._parse_image(weak_self,
image))
        if notify:
            self.hud.notification(self.sensors[index][2])
        self.index = index
```

```python
    def next_sensor(self):
        """Get the next sensor"""
        self.set_sensor(self.index + 1)

    def toggle_recording(self):
        """Toggle recording on or off"""
        self.recording = not self.recording
        self.hud.notification('Recording %s' % ('On' if self.recording else 'Off'))

    def render(self, display):
        """Render method"""
        if self.surface is not None:
            display.blit(self.surface, (0, 0))

    @staticmethod
    def _parse_image(weak_self, image):
        self = weak_self()
        if not self:
            return
        if self.sensors[self.index][0].startswith('sensor.lidar'):
            points = np.frombuffer(image.raw_data, dtype=np.dtype('f4'))
            points = np.reshape(points, (int(points.shape[0] / 4), 4))
            lidar_data = np.array(points[:, :2])
            lidar_data *= min(self.hud.dim) / 100.0
            lidar_data += (0.5 * self.hud.dim[0], 0.5 * self.hud.dim[1])
            lidar_data = np.fabs(lidar_data)  # pylint: disable=assignment-from-no-
return
            lidar_data = lidar_data.astype(np.int32)
            lidar_data = np.reshape(lidar_data, (-1, 2))
            lidar_img_size = (self.hud.dim[0], self.hud.dim[1], 3)
            lidar_img = np.zeros(lidar_img_size)
            lidar_img[tuple(lidar_data.T)] = (255, 255, 255)
            self.surface = pygame.surfarray.make_surface(lidar_img)
        else:
            image.convert(self.sensors[self.index][1])
            array = np.frombuffer(image.raw_data, dtype=np.dtype("uint8"))
            array = np.reshape(array, (image.height, image.width, 4))
            array = array[:, :, :3]
            array = array[:, :, ::-1]
            self.surface = pygame.surfarray.make_surface(array.swapaxes(0, 1))
        if self.recording:
            image.save_to_disk('_out/%08d' % image.frame)

# ==============================================================================
# -- Game Loop ---------------------------------------------------------
# ==============================================================================


global xscr
xscr = 0
```

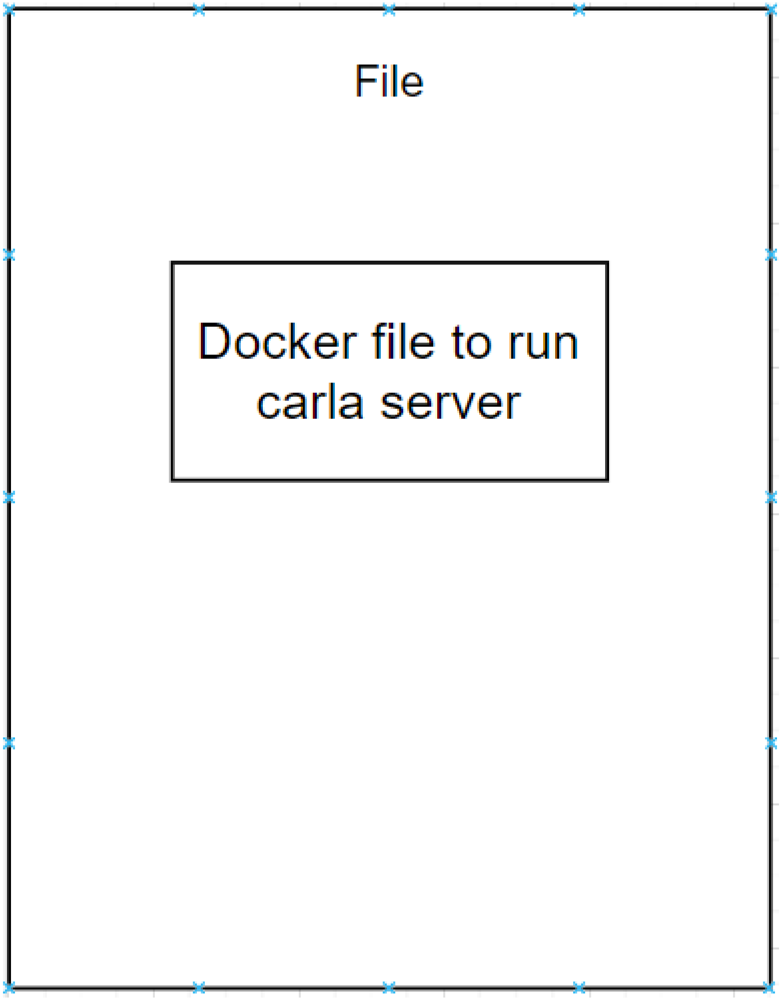
```
def sub_callback(msg):
    global xscr
    xscr = msg.data
    #print(xscr)


def run_step():
    global xscr
    control = carla.VehicleControl()
    control.steer = 0.0
    print(xscr)
    control.throttle = xscr
    control.brake = 0.0
    control.hand_brake = False
    control.manual_gear_shift = False
    control.reverse = False
    return control


def game_loop(args):
    """
    Main loop of the simulation. It handles updating all the HUD information,
    ticking the agent and, if needed, the world.
    """

    pygame.init()
    pygame.font.init()
    world = None

    try:
        if args.seed:
            random.seed(args.seed)

        client = carla.Client(args.host, args.port)
        client.set_timeout(4.0)

        traffic_manager = client.get_trafficmanager()
        sim_world = client.get_world()

        if args.sync:
            settings = sim_world.get_settings()
            settings.synchronous_mode = True
            settings.fixed_delta_seconds = 0.05
            sim_world.apply_settings(settings)

            traffic_manager.set_synchronous_mode(True)

        display = pygame.display.set_mode(
            (args.width, args.height),
```

```
            pygame.HWSURFACE | pygame.DOUBLEBUF)

        hud = HUD(args.width, args.height)
        world = World(client, client.get_world(), hud, args)
        controller = KeyboardControl(world)
        if args.agent == "Basic":
            agent = BasicAgent(world.player)
        else:
            agent = BehaviorAgent(world.player, behavior=args.behavior)

        clock = pygame.time.Clock()

        while True:
            clock.tick()
            if args.sync:
                world.world.tick()
            else:
                world.world.wait_for_tick()
            if controller.parse_events():
                return

            world.tick(clock)
            world.render(display)
            pygame.display.flip()

            # if agent.done():
            #     if args.loop:
            #         agent.set_destination(random.choice(spawn_points).location)
            #         world.hud.notification("The target has been reached,
searching for another target", seconds=4.0)
            #         print("The target has been reached, searching for another
target")
            #     else:
            #         print("The target has been reached, stopping the simulation")
            #         break

            control = run_step()  #agent.run_step()
            world.player.apply_control(control)

    finally:

        if world is not None:
            settings = world.world.get_settings()
            settings.synchronous_mode = False
            settings.fixed_delta_seconds = None
            world.world.apply_settings(settings)
            traffic_manager.set_synchronous_mode(True)

            world.destroy()
```

```
        pygame.quit()


# ==============================================================================
# -- main() --------------------------------------------------------------
# ==============================================================================


def main():
    """Main method"""

    argparser = argparse.ArgumentParser(
        description='CARLA Automatic Control Client')
    argparser.add_argument(
        '-v', '--verbose',
        action='store_true',
        dest='debug',
        help='Print debug information')
    argparser.add_argument(
        '--host',
        metavar='H',
        default='127.0.0.1',
        help='IP of the host server (default: 127.0.0.1)')
    argparser.add_argument(
        '-p', '--port',
        metavar='P',
        default=2000,
        type=int,
        help='TCP port to listen to (default: 2000)')
    argparser.add_argument(
        '--res',
        metavar='WIDTHxHEIGHT',
        default='1280x720',
        help='Window resolution (default: 1280x720)')
    argparser.add_argument(
        '--sync',
        action='store_true',
        help='Synchronous mode execution')
    argparser.add_argument(
        '--filter',
        metavar='PATTERN',
        default='vehicle.*',
        help='Actor filter (default: "vehicle.*")')
    argparser.add_argument(
        '-l', '--loop',
        action='store_true',
        dest='loop',
        help='Sets a new random destination upon reaching the previous one
(default: False)')
    argparser.add_argument(
```

```
        "-a", "--agent", type=str,
        choices=["Behavior", "Basic"],
        help="select which agent to run",
        default="Behavior")
    argparser.add_argument(
        '-b', '--behavior', type=str,
        choices=["cautious", "normal", "aggressive"],
        help='Choose one of the possible agent behaviors (default: normal) ',
        default='normal')
    argparser.add_argument(
        '-s', '--seed',
        help='Set seed for repeating executions (default: None)',
        default=None,
        type=int)

    args = argparser.parse_args()

    args.width, args.height = [int(x) for x in args.res.split('x')]

    log_level = logging.DEBUG if args.debug else logging.INFO
    logging.basicConfig(format='%(levelname)s: %(message)s', level=log_level)

    logging.info('listening to server %s:%s', args.host, args.port)

    print(__doc__)

    try:
        game_loop(args)

    except KeyboardInterrupt:
        print('\nCancelled by user. Bye!')


if __name__ == '__main__':
    rospy.init_node('listener', anonymous=True)
    rospy.Subscriber('/out_value', Float32, sub_callback)
    main()
```

5.




```
#!/bin/sh

sudo docker run -d -p 2000-2002:2000-2002 --gpus all -e NVIDIA_VISIBLE_DEVICES=0
carlasim/carla:0.9.13 /bin/bash CarlaUE4.sh -RenderOffScreen -quality-level=Low
```

REFERENCES

[1] *What is the internet of things (IOT)?* What Is the Internet of Things (IoT)? | Oracle Egypt. (n.d.). Retrieved December 24, 2022, from https://www.oracle.com/eg/internet-of-things/what-is-iot/#:~:text=What%20is%20IoT%3F,and%20systems%20over%20the%20internet.

[2] AG, I. T. (n.d.). *Internet of things: Definition, basics, use - infineon technologies*. Internet of Things: Definition, Basics, Use - Infineon Technologies. Retrieved December 24, 2022, from https://www.infineon.com/cms/en/discoveries/internet-of-things-basics/

[3] Mohanakrishnan, R. (2022, May 17). *Top 10 applications of IOT in 2022*. Top 10 Applications of IoT. Retrieved December 24, 2022, from https://www.spiceworks.com/tech/iot/articles/top-applications-internet-of-things/#_006

[4] Jost, D. (2019, October 7). *What is an ultrasonic sensor?* Fierce Electronics. Retrieved December 24, 2022, from https://www.fierceelectronics.com/sensors/what-ultrasonic-sensor

[5] *Gear Motor Dual Shaft for smart car robot arduino, DC geared motors for robots straight shaft*. Buy Online at Best Price in Egypt - Souq is now Amazon.eg. (n.d.). Retrieved December 24, 2022, from https://www.amazon.eg/-/en/Arduino-Geared-Motors-Robots-Straight/dp/B091C61G94

[6] Ada, L. (n.d.). *Adafruit Motor Shield*. Adafruit Learning System. Retrieved December 24, 2022, from https://learn.adafruit.com/adafruit-motor-shield/af-dcmotor-class

[7] *Arduino Mega 2560 R3 Microcontroller Board*. Buy Online at Best Price in Egypt - Souq is now Amazon.eg. (n.d.). Retrieved December 24, 2022, from https://www.amazon.eg/-/en/Arduino-Mega-2560-Microcontroller-Board/dp/B0922Z7HRP

[8] Ed. (2021, November 15). *Raspberry pi arduino serial communication - everything you need to know - the robotics back*. End. Retrieved December 24, 2022, from https://roboticsbackend.com/raspberry-pi-arduino-serial-communication/#:~:text=The%20easiest%20way%20is%20to,Arduino%20IDE)%20to%20your%20board.

[9] *What is a Raspberry Pi?* Opensource.com. (n.d.). Retrieved December 24, 2022, from https://opensource.com/resources/raspberry-pi

[10] Baptista, J. V. (1991). *AR*. Amazon. Retrieved December 24, 2022, from https://aws.amazon.com/ar/what-is-aws/

[11] "FMI Functional Mockup Interface", FMI Standard Website. Retrieved from: https://fmi-standard.org/

[12] Hatledal, L. I., Zhang, H., Styve, A., & Hovland, G. (2018). "Fmi4j: A software package for working with functional mock-up units on the java virtual machine". In The 59th Conference on Simulation and Modelling (SIMS 59). Linköping University Electronic Press, Linköpings universitet.

[13] “Functional Mock-up Interface (FMI)”, WOLFRAM Website. Retrieved from: https://reference.wolfram.com/system-modeler/UserGuide/ModelCenterFunctionalMockupInterface.html

[14] “Export Simulink Model to Standalone FMU”, Mathworks website. Retrieved from: https://www.mathworks.com/help/slcompiler/ug/simulinkfmuexample.html

[15] “Graphical User Interface”, FMPy Website. Retrieved from: https://fmpy.readthedocs.io/en/latest/tutorial/