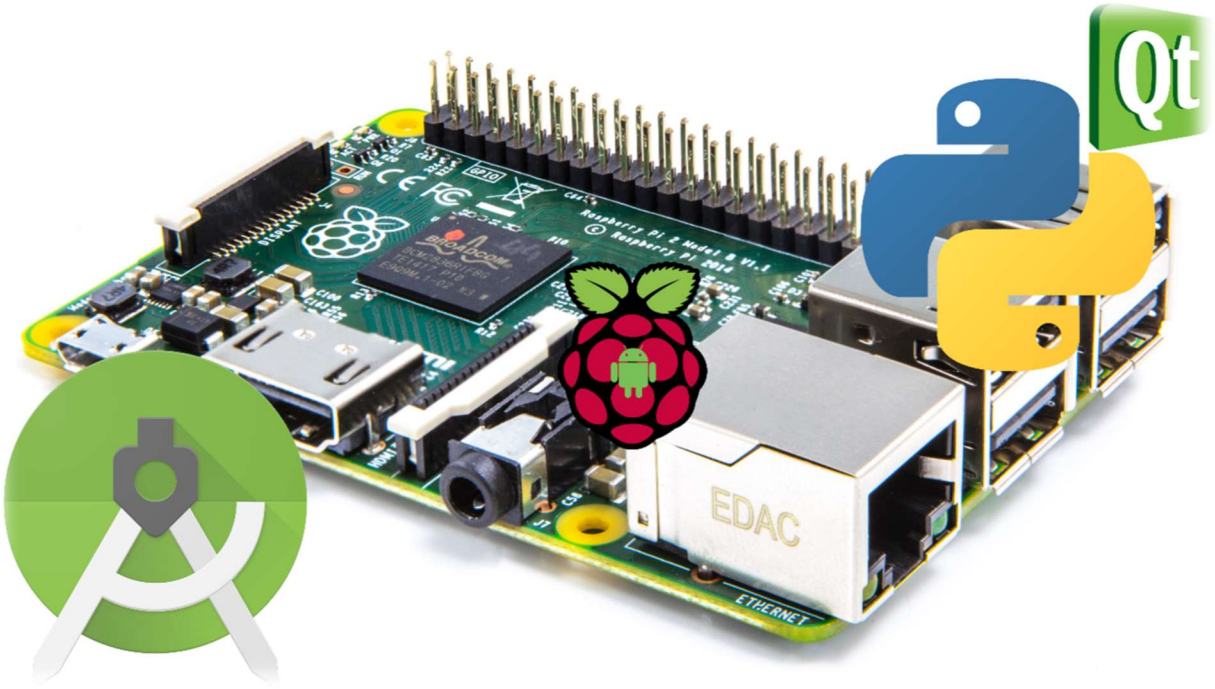
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Direction Générale des Etudes Technologiques
Institut Supérieur des Etudes Technologiques de Rades

Mini-Project

RasControl



Elaborated by:

Mohamed Yassine Sabri

Class L3AII1

Framed by:

Mr. Mondher Mlaouah

College year 2017-2018

Summary

Introduction

Chapter 1: Specifications

Chapter 2: Project Description

1. Introduction
2. Hardware
    a. Installation
3. Software
    a. Graphical User Interface GUI
    b. Android App

Chapter 3: Material

4. Introduction

In this chapter, you will find a full description of the software and hardware material that were used to help create the project.

5. Hardware

a. Raspberry pi 3 Model B

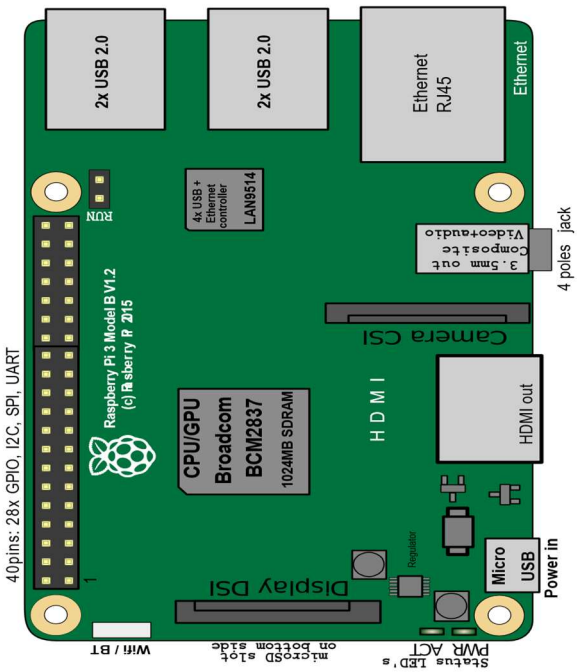
The raspberry pi is a series of a small single-board computer developed in the U.K. by the Raspberry Pi Foundation to promote teaching of basic computer science in schools and in developing countries.

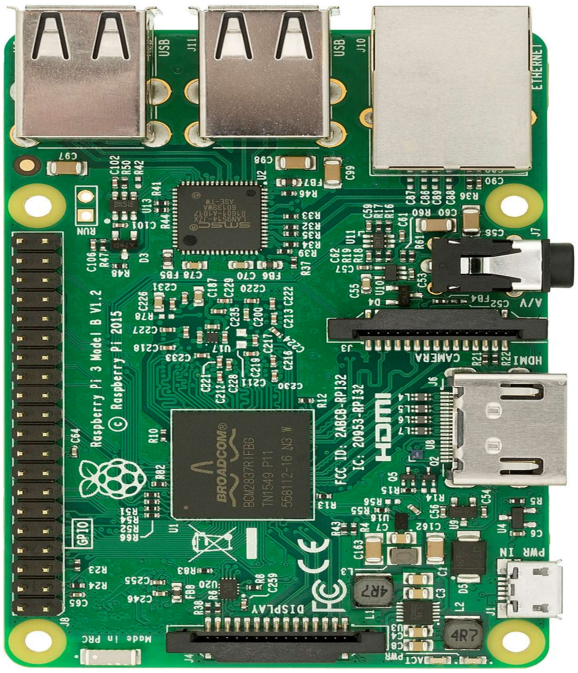
The foundation is a charity organization founded in 2009. In 2014 the first generation of RPIs was released and since then 12.5 million board have been sold, Making it the best-selling general-purpose computer.

There are various models A, B, Compute Module and Zero with various generations. In this project we use RPI Third Generation Model B.

Raspberry pi 3 model B was released in February 2016, and this board has:

Figure 1 Raspberry pi





- 1.2 GHz 64-bit quad-core ARM CPU
- Broadcom Video Core IV 400 MHz GPU
- 1GB (900 MHz) of SDRAM
- 40 GPIO pins
- Networking: 10/100Mb Ethernet, 2.4GHz 802.11n wireless
- Bluetooth: Bluetooth 4.1 Classic
- Ports**:** HDMI, 3.5mm analogue audio jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)

The official supported OS by the foundation is Raspbian which Debian-based Linux system.

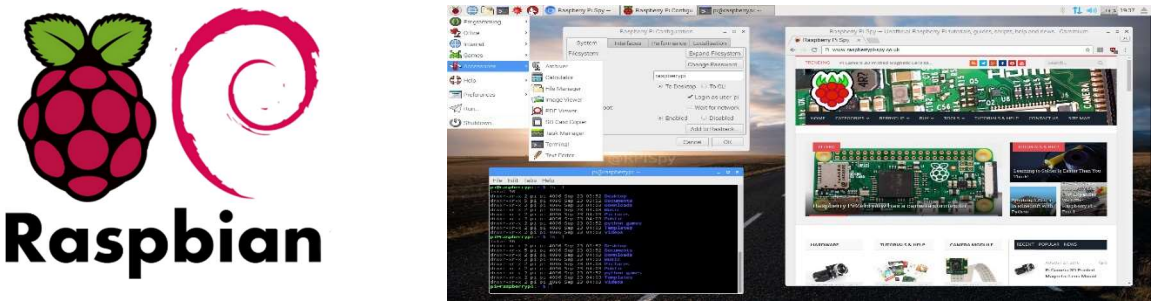


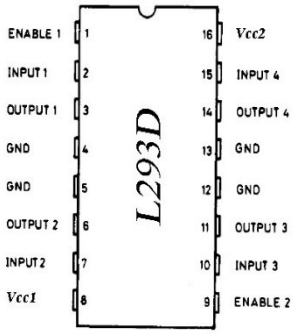
Figure 2 Raspbian

There are other third-party OS images like: Ubuntu Mate, windows IOT 10 Core, Android Things, Kali Linux …

b. Half-H Motor Driver L293D

The L293D device is quadruple high-current half-H driver. It is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V.

There are two reasons why we need to use a L293D chip in this project. The first is that the output of the Raspberry Pi is nowhere near strong enough to drive a motor directly and to try this may damage your Raspberry Pi. Secondly, we want to control the direction of the motor as well as its speed. This is only possible by reversing the direction of the current through the motor, something that the L293D is designed.



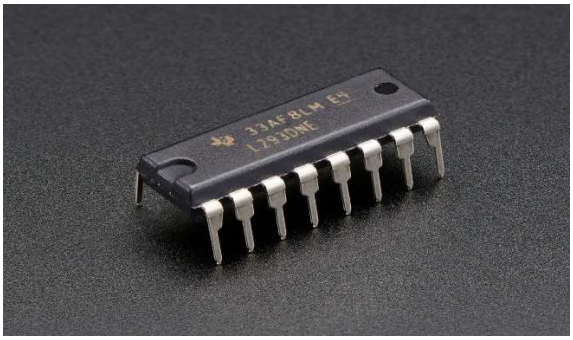


Figure 3 L293D

Connecting the L293D to the raspberry pi:

L293D Pin	RPI GPIO	L293D Pin	RPI GPIO
Enable 1	22	VCC2 16	2 (5V)
Input1 2	16	Input4 15	N/A
Output1 3	Motor output	Output4 14	N/A
GND 4	6 (GND)	GND 13	6 (GND)
GND 5	6 (GND)	GND 12	6 (GND)
Output2 6	Motor output	Output3 11	N/A
Input2 7	18	Input3 10	N/A
VCC1 8	Battery input (6v)	Enable2 9	N/A

c. Micro Servo SG92R

It is a tiny and lightweight servo that can rotate approximately 180 degrees.

Connection to the raspberry pi:

Red wire into GPIO #2 (5v)

Brown into GPIO #6 (GND)

Yellow wire into GPIO #3 (PWM input).

Figure 4 Micro Servo

d. DC Motor

With operating voltage of 6v, this motor can run at 11500rpm.

Figure 5 DC Motor

Connection to the raspberry pi:

It is connected to the L293D, through pin 3 and 6.

6. Software

a. Python PyQt

There are multiple programming languages that you can program the raspberry pi with, but Python is most documented, and it comes preinstalled.

Python is a widely used high-level programming language for general-purpose programming, cross platform and with its large libraries you can use it to develop your project easily. Python was created in 1991 and the latest stable release: Python 2.7.14 and Python 3.6.3 for python3.

Figure 6 Python

Figure 7 Qt Icon

Several libraries in python allows the developing of GUIs such as: Tkinter, WxWidgets, Gtk+, Qt….
In this project I am using PyQt library which is a python binding of the cross-platform GUI toolkit Qt to develop the front-end of the project.

b. Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains IntelliJ IDEA software and designed specifically for Android development. It was announced in May 2013, and the current stable release is 3.0.

Android is a mobile operating system developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets.

Android Studio supports multiple programming languages including Java, C++, and lately Kotlin. In creating the android app, I have used Java.

Figure 8 Android studio

Figure 9 Android

Chapter 4: Program

1. Android App

The angle, speed, and strength are sent from the app to raspberry pi through a socket server, The server is running on the raspberry and the android app connects to it and is able to send data.

```
2. js.setOnMoveListener(new JoystickView.OnMoveListener() {
3.     @Override
4.     public void onMove(int angle, int strength, int posx, int posy) {
5.         // do whatever you want
6.         if(startControl) {
7.             textangle.setText("Angle : "+angle+"°");
8.
9.             double s = (((int) (js.getW()/2.0) - posy))/(js.getB()/100.0);
10.              textspeed.setText(df2.format(s)+" rpm");
11.
12.
   connection.SendDataToNetwork(Integer.toString(angle)+":"+Integer.toString(strength)
   +":"+Double.toString(s)+"\n");
13.          }
14.          else {
15.              Log.d("stop mode", "Stop mode is on!");
16.          }
17.
18.      }
19.  });
```

2. Graphical User Interface GUI

The MyTCPHandler class is the socket server that waits for a client to connect and send data. Its then captured and send to Motor/Servo functions and to the UI.

```
3. class MyTCPHandler(socketserver.StreamRequestHandler):
4.
5.     def handle(self):
6.         while True:
7.             global angle, strength ,speed, a, connected, appui, ui
8.             if(not enabled):
9.                 # self.rfile is a file-like object created by the handler;
10.                  # we can now use e.g. readline() instead of raw recv() calls
11.                  self.data = self.rfile.readline().strip()
12.                  if(a==0):
13.                      connected = "Connected ip: "+self.client_address[0]
14.                      appui.setCon(connected)
15.                      a+=1
16.                  data = self.data.decode("utf-8").rstrip().strip("\n")
```

```
17.                 if(data!= ""):
18.                     if(data == "disconnect"):
19.                         print("restarting")
20.                         a = 0
21.                         self.close()
22.                         sleep(1)
23.                         mainServer()
24.                     elif(data == "gui"):
25.                         appui.hide()
26.                         enabled = True
27.                         ui.setGui()
28.                     else:
29.                         temp = data.split(":")
30.                         angle = int(temp[0])
31.                         strength = int(temp[1])
32.                         speed = float(temp[2])
33.                         appui.update(speed, angle)
34.                         Servo()
35.                         Motor()
36.
37. # ********************************************** #
38. def mainServer():
39.
40.     p = subprocess.Popen("ifconfig wlan0 | grep 'inet ' | cut -d' ' -f10",
    shell=True, stdout=subprocess.PIPE, stderr=subprocess.STDOUT)
41.     ip = p.stdout.readlines()[0].decode("utf-8").rstrip().strip("\n")
42.     port = 9999
43.     global add
44.     add = ip+":"+str(port)
45.
46.     try:
47.         # Create the server, binding to localhost on port 9999
48.         server = socketserver.TCPServer((ip, port), MyTCPHandler)
49.         server.serve_forever()
50.
51.     except(OSError):
52.         print("Something happend!")
53. # ********************************************** #
54. # Servo motor control
55. def Servo():
56.     b = 0
57.     global preAngle, angle, strength, a
58.     if(not enabled):
59.         if(a!=0):
60.             if(angle==0 and b!=1 and strength==0):
61.                 setCenter()
62.                 b = 1
63.             elif(preAngle+3 >= angle and preAngle-3 <= angle):
64.                 pass
```

```
65.             else:
66.                 SetAngle(angle-180 if angle>=180 else angle)
67.                 preAngle = angle
68.                 b = 0
69.     else:
70.         SetAngle(angle)
71.
72. # ********************************************** #
73. # set the servo to center (90degree)
74. def setCenter():
75.     GPIO.output(3, True)
76.     servopwm.ChangeDutyCycle(7)
77.     sleep(0.0001)
78. # ********************************************** #
79. # set the servo to angle
80. def SetAngle(angle):
81.     duty = angle / 18.0 + 2
82.     GPIO.output(3, True)
83.     servopwm.ChangeDutyCycle(duty)
84.     sleep(0.0001)
85.
86. # ********************************************** #
87. # Motor control
88. def Motor():
89.     global a, speed, preSpeed
90.     b = 0
91.     if(not enabled):
92.         if(a!=0):
93.             if(speed == 0 and b == 0):
94.                 b = 1
95.                 stop()
96.             elif(preSpeed+1 >= speed and preSpeed-1 <= speed):
97.                 pass
98.             elif(speed != 0):
99.                 preSpeed = speed
100.                b = 0
101.                forward(speed) if speed>0 else backward(speed)
102.    else:
103.        if(speed == 0):
104.            stop()
105.        else:
106.            forward(speed) if speed>0 else backward(speed)
107. # ********************************************** #
108. def stop():
109.     motorpwm.ChangeDutyCycle(0)
110. # ********************************************** #
111. def forward(speed):
112.     motorpwm.ChangeDutyCycle(speed)
113.     GPIO.output(16,GPIO.HIGH)
```

```
114.     GPIO.output(18,GPIO.LOW)
115.     GPIO.output(22,GPIO.HIGH)
116.     sleep(0.0001)
117.
118. # ********************************************** #
119. def backward(speed):
120.     motorpwm.ChangeDutyCycle(-speed)
121.     GPIO.output(16,GPIO.LOW)
122.     GPIO.output(18,GPIO.HIGH)
123.     GPIO.output(22,GPIO.HIGH)
124.     sleep(0.0001)
125.
126. # ********************************************** #
127. def main(f):
128.     start_new_thread(mainServer, ())
129.
130.     app = QtWidgets.QApplication(sys.argv)
131.     MainWindow = QtWidgets.QMainWindow()
132.
133.     global appui, ui
134.
135.     ui = Ui_MainWindow()
136.     ui.setupUi(MainWindow)
137.
138.     appui = Ui_AppWindow()
139.
140.     if(f == 1):
141.         MainWindow.showFullScreen()
142.     else:
143.         MainWindow.showNormal()
144.     sys.exit(app.exec_())
145.
146. # ------------------------------------------------------------------- #
147. if __name__ == "__main__":
148.
149.     main(1)
```

The full code can be found at Github:

RasControl App:

RasControl GUI:

Conclusion

Figures