Acknowledgment

I would like to take the time to express my profound gratitude to my professor Mr. Mondher Mlaouah, for his guidance, advices and help. Also, special thanks goes to the jurors who gave us some of their time and accepted to judge our modest work.

Table of Contents

Figures

Introduction

Electrical engineering and information technology are major fields in the current world, and combining them results in great technical innovations. Whether its Cell Phones, Microchips, Batteries for Electric Cars, or Internet of Things systems, all are equipment, parts, or process, that were invented with the help of these two fields.

Large numbers of Internet of Things(IOT) systems are joining the world network every year. In each second, they transmit or receive sensors data over the internet, and it gets processed by the main board or computer or it is used to control mechanical or electrical parts. These IOTs are very useful and used in enormous fields all over the world.

Hence, in this project we will develop an IOT system, which include a Raspberry Pi board that receives data from an android app through a socket server, or a graphical user interface installed inside the board. Then the data is used to control a micro servo and a dc motor.

Chapter 1: Specifications

1. Introduction

In this chapter, you will find the project specification and the reason of chosen each component.

2. Hardware

a. The Board

There are multiple boards that can be used in this project:

- The NanoPi Neo Plus 2 uses a 1.5GHz quad-core processor, a 1GB of RAM, built-in Wi-Fi, Bluetooth, and Ethernet, 2 USB2.0 ports, and support for MicroSD cards to augment its 8GB of onboard storage. This board lacks the HDMI, audio port and unavailable in the market.
- The ASUS Tinker includes a 1.8GHz quad-core processor with 2GB of RAM. It has the standard Ethernet/Wi-Fi/Bluetooth combo, plus an included MicroSD card slot and 4 USB2.0 ports. The board is expensive and unavailable in the market.
- The Raspberry Pi 3 model B comes with 1.2GHz processor, 1GB of RAM, 400MHz GPU, built-in Ethernet/Wi-Fi/Bluetooth, HDMI, Audio port, 4 USB2.0 ports, plus a MicroSD card slot. The Raspberry is available in the market, cheaper than the Asus Tinker, same price as the NanoPi, well documented, and has a large online community.

In this project, I will be using the Raspberry Pi 3 model B.

b. Motor Driver

Two reasons make us use a motor driver. 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 the board. 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 Half-H motor driver L293D is designed for. Plus, it has an affordable price. That's why in this project I have used it.

c. Micro Servo

Servos differ from one another, but they still do the same task, rotate 180 or 360 degrees. In this project I choose to use the SG92R, it's small, lightweight, cheap, available in the market and can handle the task required from it.

d. DC Motor

In this project, I am using a 6v DC motor which is the same as the ones used in a small toy car.

3. Software

a. Raspberry Pi

There are several programming languages that you can program the raspberry pi with:

- Python is widely used, well documented, has a large community, and it's the official programming language for Raspberry Pi.
- Java is widely used, well documented for working on desktops, but lacks the documentation for embedded systems.
- C is used in a wide range of platform, but there's no documentation on how to use it in the RPI.

Python is the programming language that was used in the RPI.

b. Android App

Multiply platforms allows the development of android apps, such as:

- App Inventor for Android is an open-source web application originally provided by Google, and now maintained by the Massachusetts Institute of Technology (MIT). It's a graphical programming language (drag and drop) same as Scratch.
- HTML/CSS/JS There are some framework that allows you to create an app with web tools such as NativeScript, Tabris, PhoneGap …
- Android Studio which is the official development environment for android apps.

Android Studio was used to develop the Android App.

4. Conclusion

Hardware and software component were chosen based on availability, price, documentation and official support from the organization.

Chapter 2: Material

1. Introduction

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

2. Hardware

a. Raspberry pi 3 Model B

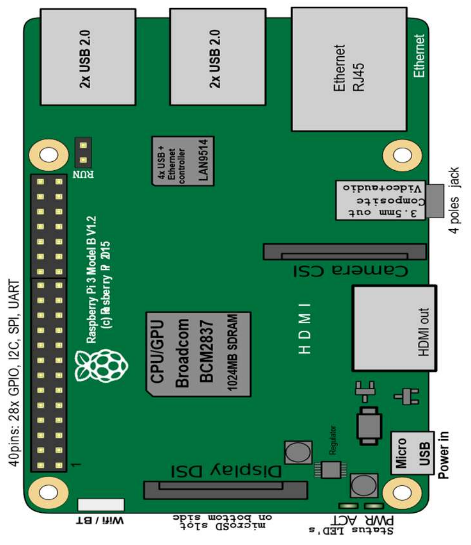
The raspberry pi is a series of a small single-board computer developed in the U.K. by the Raspberry Pi Foundation to promote teaching of basic computer science in schools and in developing countries.

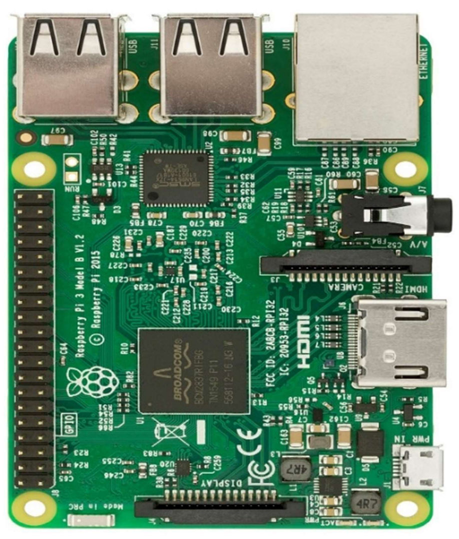
The foundation is a charity organization founded in 2009. In 2014 the first generation of RPIs was released and since then 12.5 million board have been sold, Making it the best-selling general-purpose computer.

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


2x USB 2.0
2x USB 2.0
Ethernet RJ45
Ethernet
RUN
4x USB + Ethernet controller LAN9514
Videoaudio Composite 3.5mm out
4 poles jack
Camera CSI
Raspberry Pi 3 Model B V1.2 (c) Rasberry Pi 2015
CPU/GPU Broadcom BCM2837 1024MB SDRAM
H D M I
HDMI out
40pins: 28x GPIO, I2C, SPI, UART
1
Regulator
Micro USB
Power in
Display DSI
Wifi / BT
microSD slot on bottom side
Status LED's PWR ACT



- 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 is a Debian-based Linux system.

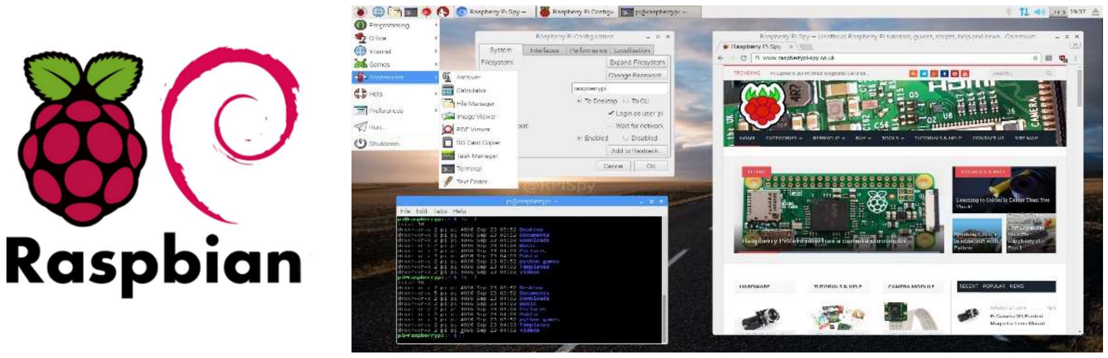
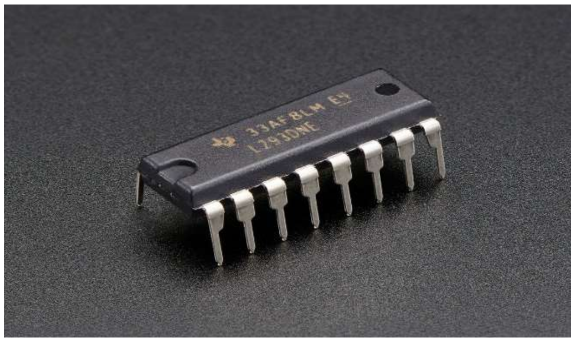


Figure 2 Raspbian

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.



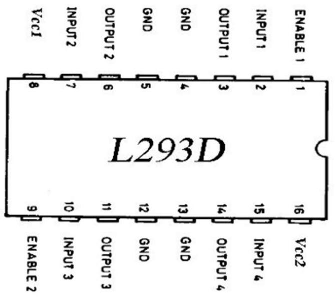


Figure 3 L293D

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.

Figure 4 Micro Servo

Connection to the raspberry pi:

Red wire into GPIO #2 (5v)

Brown into GPIO #6 (GND)

Yellow wire into GPIO #3 (PWM input).

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.

3. Software

a. Python PyQt

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 3.6.3 for python3.

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.

Figure 6 Python

Figure 7 Qt Icon

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

4. Conclusion

Various hardware and software component were put together to develop this project.

Chapter 3: Project Description

1. Introduction

In This chapter, you will find detailed description of the project, and how it works.

2. Hardware

We are using the raspberry pi, the L293D motor driver, DC motor, and the micro servo. Connecting these parts to the raspberry is done by jumper wires: male to male and female to male.

You can find in Chapter 2 how to connect each component to the raspberry. The final assembly is as shown in this image:

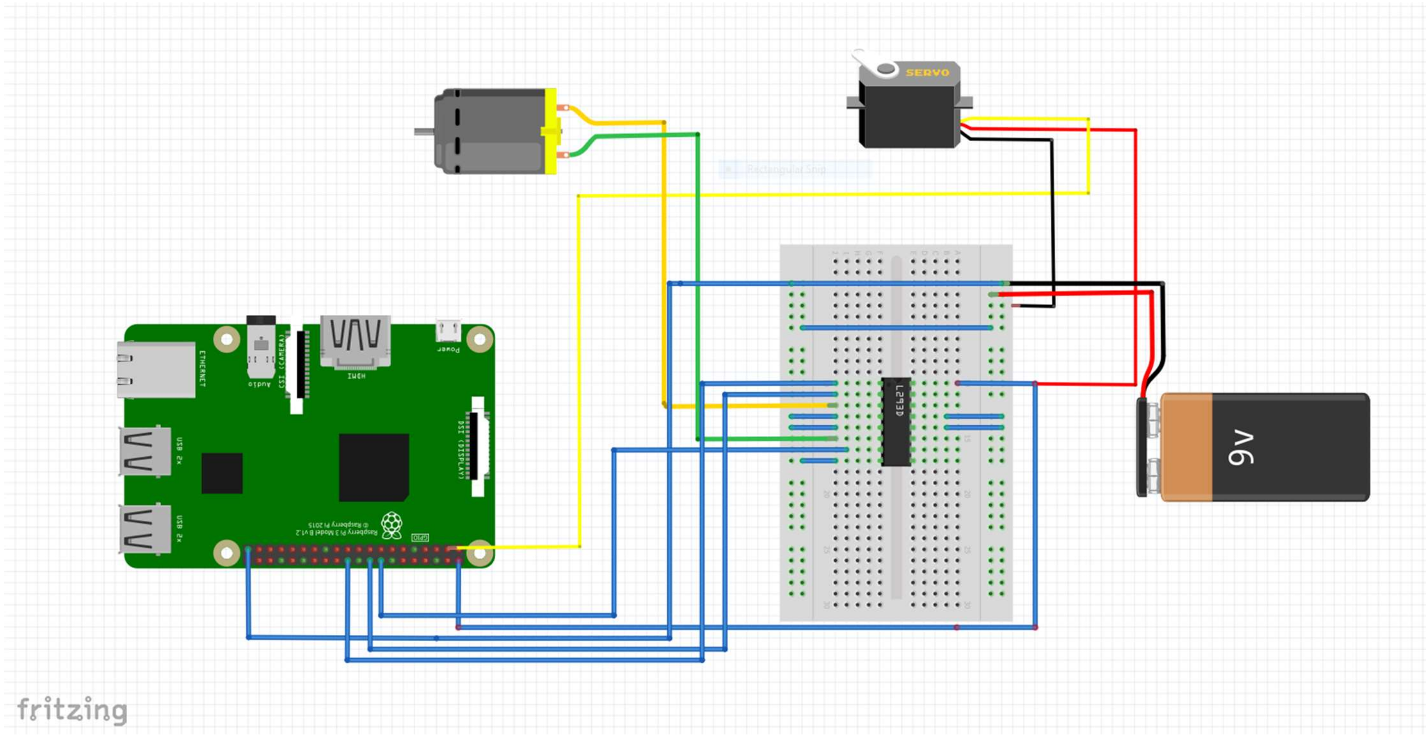


Figure 10 Connection

After connecting the component, you only need to edit the code and add the pins that you connected the micro servo and the DC motor to.

3. Software

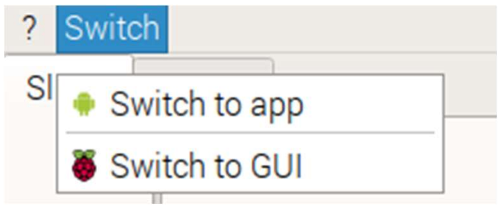
a. Graphical User Interface GUI

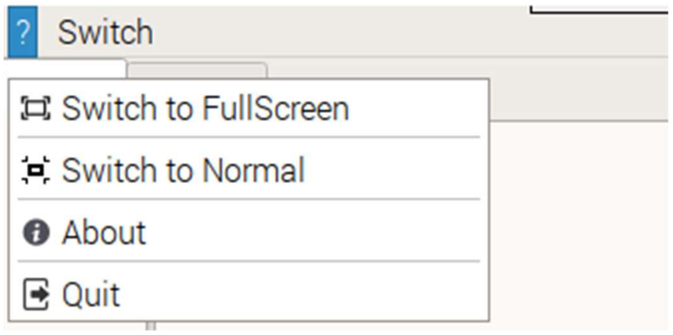
There are two ways you can control the component with, either by the Button control or the Slider control. Both are the same, they only differ in UIs.

You can switch between the GUI and the android app by clicking switch to app in the switch menu.

Figure 11 GUI Switch Menu

Figure 12 GUI ? Menu





i. Button Control

This tab allows you to control the Raspberry with buttons.

The forward and backward buttons will increase or decrease the current speed by 20%.

The left and right buttons will increase or decrease the angles by10°.

The stop button will set the speed to 0% and the angle to 90°.

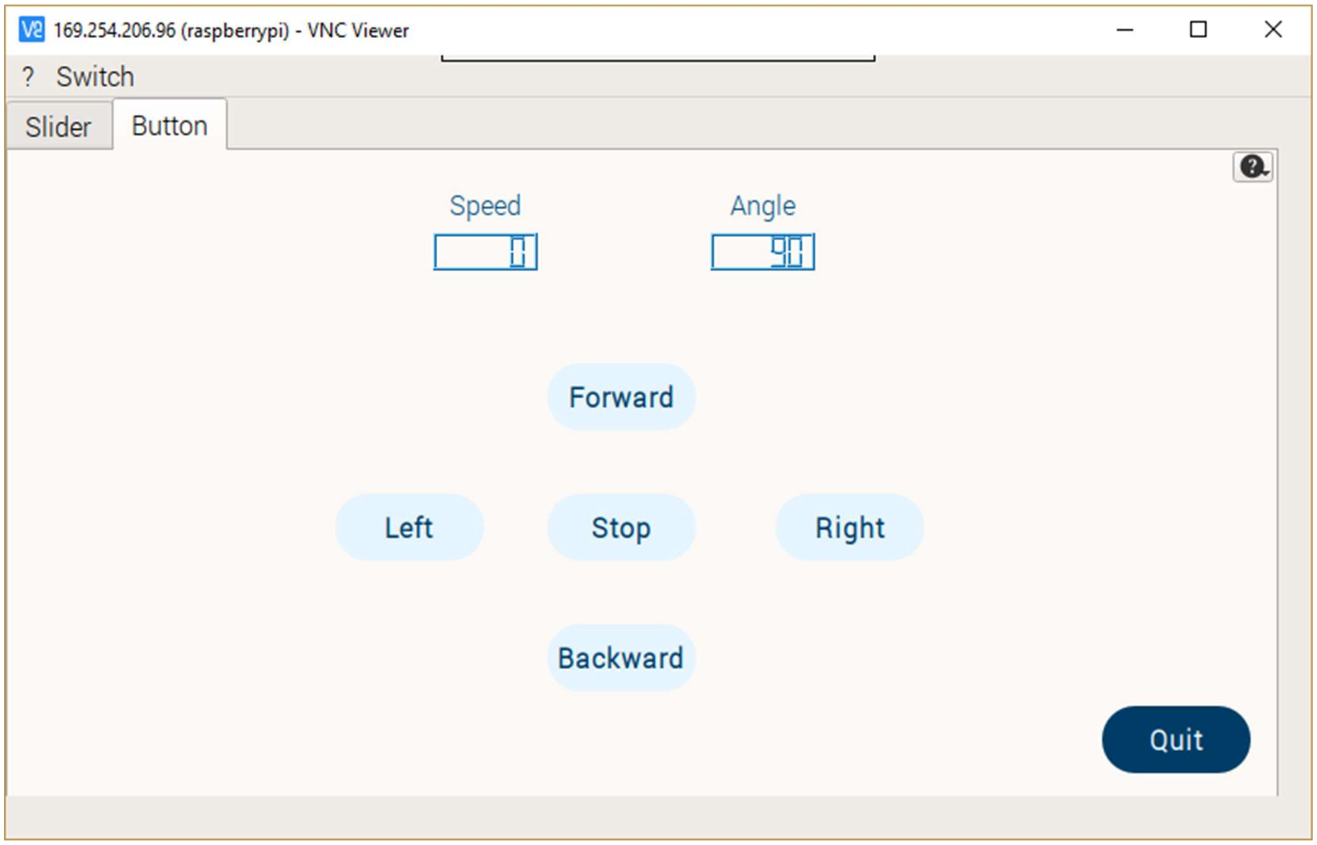


Figure 13 Button Control

ii. Slider Control

This tab allows you to control the Raspberry with sliders. You can click or move each slider and based on that, the angle and speed will be set.

If you click on the speed slider, based on the direction of the click the speed will be increased or decreased by 20%.

If you click on the angle slider, based on the direction of the click the angle will be increased or decreased by 10°.

The stop button will set the speed to 0% and the angle to 90°.

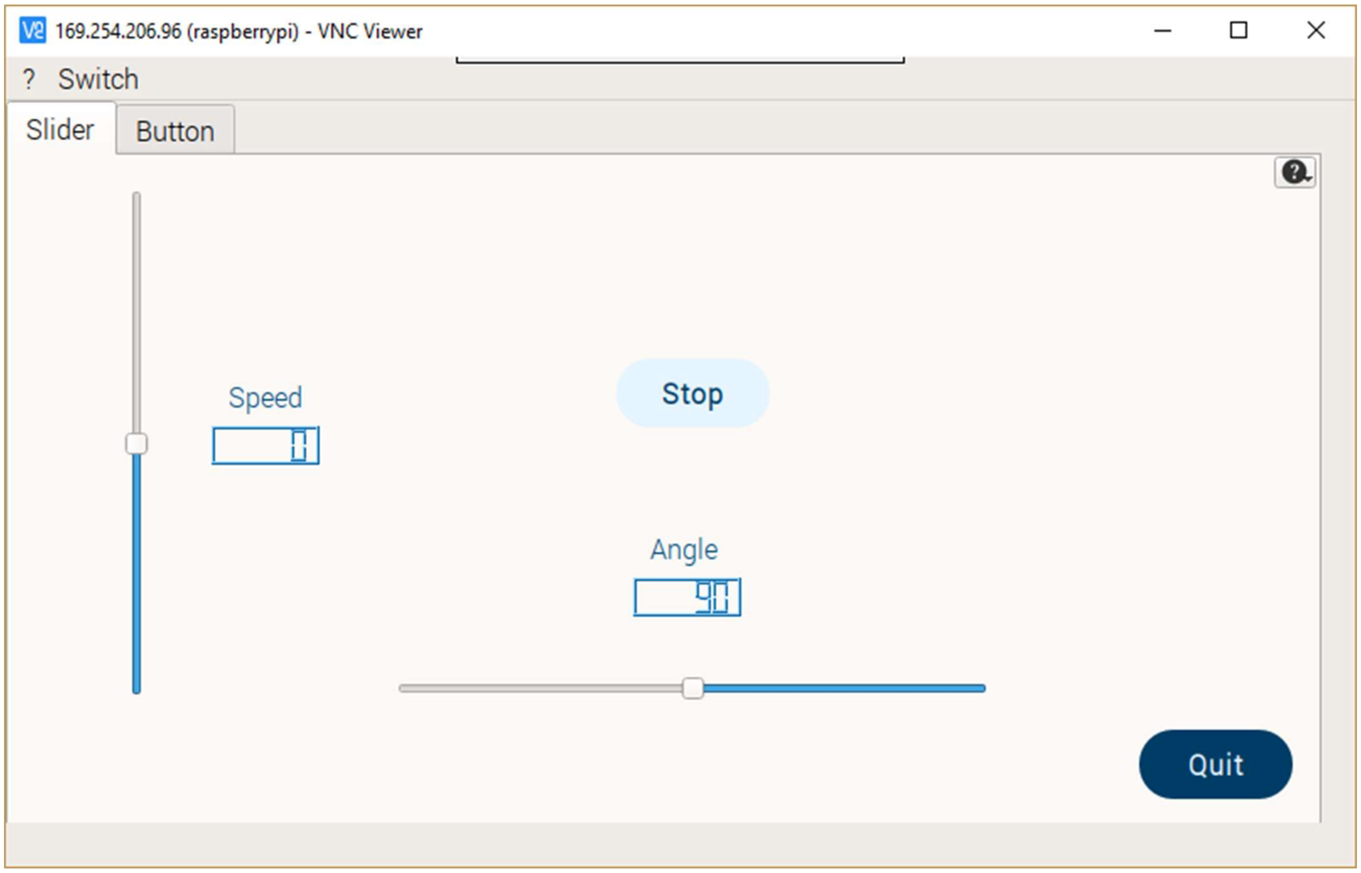


Figure 14 Slider Control

b. Android App

The control in the app is based on the joystick principle, so all you do is move the small circle around and the data will be calculated and sent to the raspberry pi which will send it the motor and servo.

Firstly, you will need to set up a connection to the server. In the setting menu enter the server ip and connect to it.

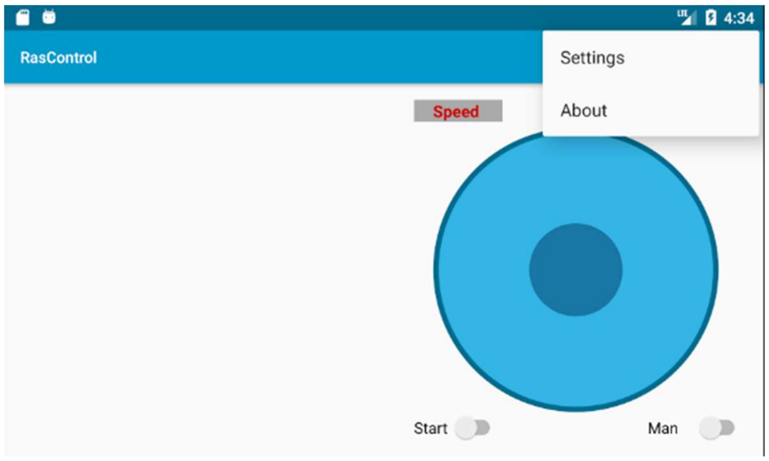


Figure 15 App Menu

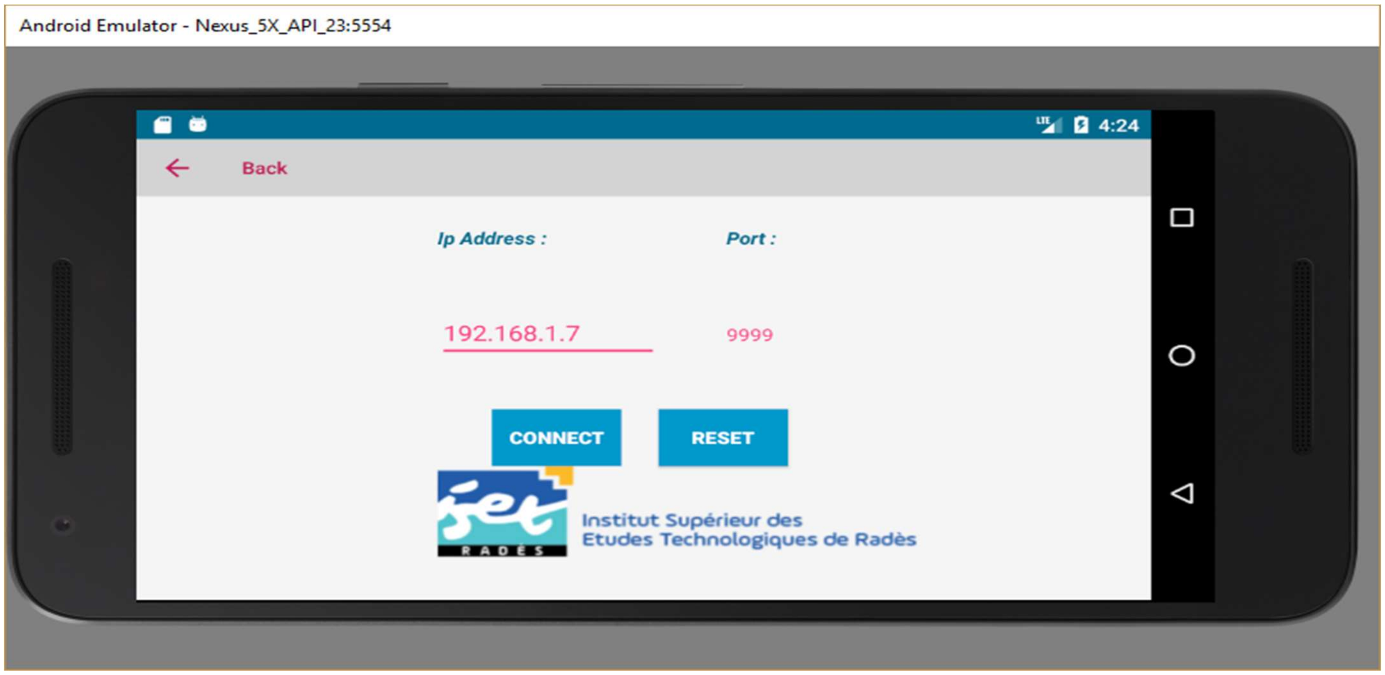


Figure 16 App Setting

Secondly, click the start switch in the main activity which will cause the app to try to connect to the server. If it succeeds you will get a notification confirming the connection and you can start moving the joystick. Else you will get an error notification and you can reattempt to connect.

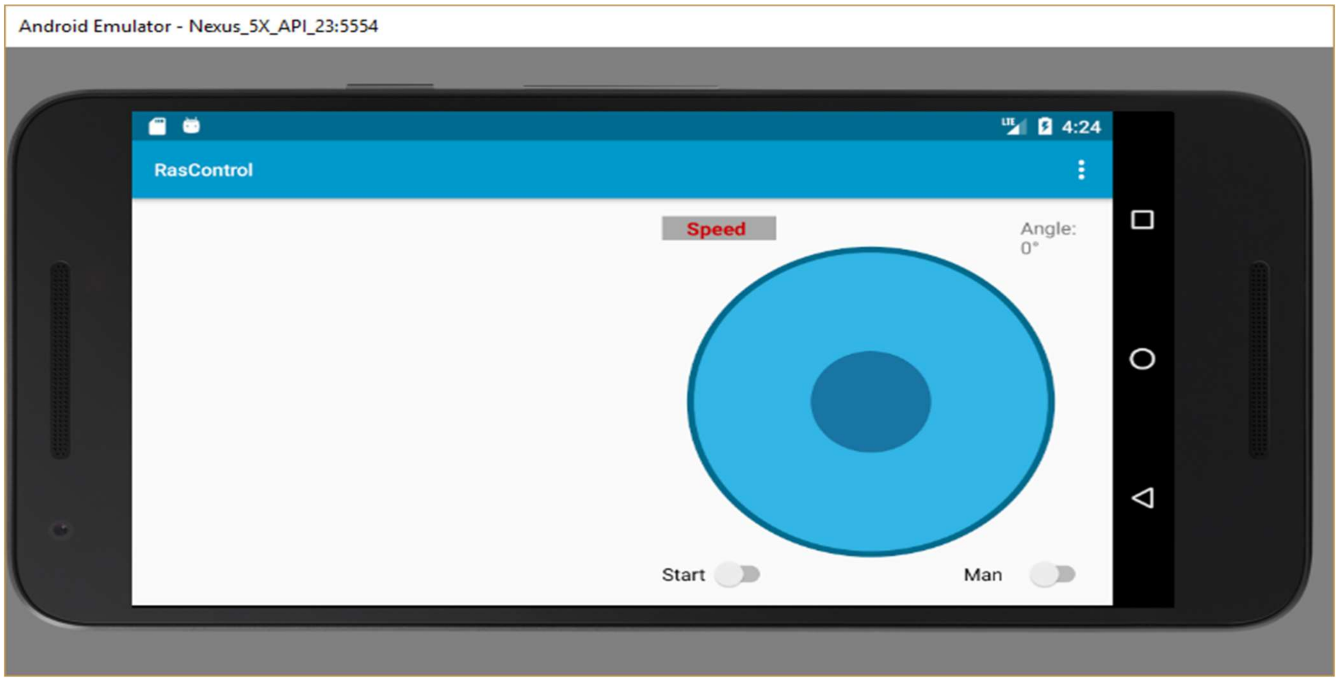


Figure 17 App Main

4. Conclusion

The project is easy to comprehend, use, and to be modified based on your needs.

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 data can be transmitted back and forward.

```
js.setOnMoveListener(new JoystickView.OnMoveListener() {
        @Override
        public void onMove(int angle, int strength, int posx, int posy) {
          // do whatever you want
          if(startControl) {
            textangle.setText("Angle : "+angle+"°");

            double s = (((int) (js.getW()/2.0) - posy))/(js.getB()/100.0);
            textspeed.setText(df2.format(s)+" ");

connection.SendDataToNetwork(Integer.toString(angle)+":"+Integer.toString(strength)+":"+Double.toString(s)+"\n
");
          }
          else {
            Log.d("stop mode", "Stop mode is on!");
          }

        }
      }, 100);
```

2. Graphical User Interface GUI

The MyTCPHandler class is the socket server that waits for a client to connect to send/receive data. Its then captured and sent to Motor/Servo function and to the UI if the App mode is enabled.

```
# variable init
a = 0
angle = 90
preAngle = 0
```

```
preSpeed = 0
speed = 0
strength = 0
add = ""
connected = ""
enabled = True

# Servo config
serPin = 37
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(serPin, GPIO.OUT)
servopwm = GPIO.PWM(serPin, 50)
servopwm.start(7)

# DC motor config
fPin = 36
bPin = 38
ePin = 40
GPIO.setup(fPin,GPIO.OUT)
GPIO.setup(bPin,GPIO.OUT)
GPIO.setup(ePin,GPIO.OUT)

motorpwm = GPIO.PWM(ePin, 50)
motorpwm.start(0)

# Socket server class
class MyTCPHandler(socketserver.StreamRequestHandler):

    def handle(self):
        while True:
            global angle, strength ,speed, a, connected, appui, ui, enabled
            if(not enabled):
                self.data = self.request.recv(1024).strip()
                if(a==0):
                    connected = "Connected ip: "+self.client_address[0]
                    appui.setCon(connected)
```

```
          a+=1
        data = self.data.decode("utf-8").rstrip().strip("\n")
        if(data!= ""):
          if(data == "disconnect"):
            print("restarting")
            a = 0
            self.close()
            sleep(1)
            mainServer()
          elif("gui" in data):
            appui.hide()
            ui.setGui()
            enabled = True
          else:
            temp = data.split(":")
            angle = int(temp[0])
            strength = int(temp[1])
            speed = float(temp[2])
            appui.update(speed, angle)
            Servo()
            Motor()

# ********************************************* #
def mainServer():

  p = subprocess.Popen("ifconfig wlan0 | grep 'inet ' | cut -d' ' -f10", shell=True, stdout=subprocess.PIPE,
stderr=subprocess.STDOUT)
  ip = p.stdout.readlines()[0].decode("utf-8").rstrip().strip("\n")
  port = 9999
  global add
  add = ip+":"+str(port)

  try:
    # Create the server, binding to localhost on port 9999
    server = socketserver.TCPServer((ip, port), MyTCPHandler)
    server.serve_forever()
    except(OSError):
        print("Something happend!")

# ************************************************ #
# Servo motor control
def Servo():
    b = 0
    global preAngle, angle, strength, a
    if(not enabled):
        if(a!=0):
            if(angle==0 and b!=1 and strength==0):
                setCenter()
                b = 1
            elif(preAngle+3 >= angle and preAngle-3 <= angle):
                pass
            else:
                SetAngle(angle-180 if angle>=180 else angle)
                preAngle = angle
                b = 0
    else:
        SetAngle(angle)

# ************************************************ #
# set the servo to center (90degree)
def setCenter():
    GPIO.output(serPin, True)
    servopwm.ChangeDutyCycle(7)
    sleep(0.0001)
# ************************************************ #
# set the servo to angle
def SetAngle(angle):
    duty = angle / 18.0 + 2
    GPIO.output(serPin, True)
    servopwm.ChangeDutyCycle(duty)
    sleep(0.0001)

# ************************************************ #
# Motor control
def Motor():
    global a, speed, preSpeed
    b = 0
    if(not enabled):
        if(a!=0):
            if(speed == 0 and b == 0):
                b = 1
                stop()
            elif(preSpeed+1 >= speed and preSpeed-1 <= speed):
                pass
            elif(speed != 0):
                preSpeed = speed
                b = 0
                forward(speed) if speed>0 else backward(speed)
    else:
        if(speed == 0):
            stop()
        else:
            forward(speed) if speed>0 else backward(speed)

# ************************************************ #
def stop():
    motorpwm.ChangeDutyCycle(0)
# ************************************************ #
def forward(speed):
    motorpwm.ChangeDutyCycle(speed)
    GPIO.output(fPin,GPIO.HIGH)
    GPIO.output(bPin,GPIO.LOW)
    GPIO.output(ePin,GPIO.HIGH)
    sleep(0.0001)

# ************************************************ #
def backward(speed):
    motorpwm.ChangeDutyCycle(-speed)
    GPIO.output(fPin,GPIO.LOW)
    GPIO.output(bPin,GPIO.HIGH)
    GPIO.output(ePin,GPIO.HIGH)
    sleep(0.0001)

# ************************************************ #
def main(f):
    start_new_thread(mainServer, ())

    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()

    global appui, ui

    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)

    appui = Ui_AppWindow()

    if(f == 1):
        MainWindow.showFullScreen()
    else:
        MainWindow.showNormal()
    sys.exit(app.exec_())

# ------------------------------------------------------------------ #

if __name__ == "__main__":

    main(1)
```

The full code can be found at GitHub:

RasControl App: https://github.com/medyas/RasControl

RasControl GUI: https://github.com/medyas/RasControlGui

Conclusion

Developing an IOT system, allowed me to gain experience and enhance my knowledge both in electrical and information field.

The project helped me get comfortable working with raspberry pi and the Linux command line (Terminal), get familiar with Python and PyQt programming language to develop the GUI and control the servo/motor, also develop an android app using XML and Java from scratch.

Since our world is moving into an information era, having such knowledge is useful, it can land you a job, help solve an industry problem, or help launch startup.

References

https://www.raspberrypi.org/

https://en.wikipedia.org/wiki/Raspberry_Pi

https://en.wikipedia.org/wiki/Python_(programming_language)

https://en.wikipedia.org/wiki/PyQt

https://en.wikipedia.org/wiki/Android_(operating_system)

https://en.wikipedia.org/wiki/Android_Studio

https://en.wikipedia.org/wiki/Fritzing

https://stackoverflow.com/

https://github.com/controlwear/virtual-joystick-android