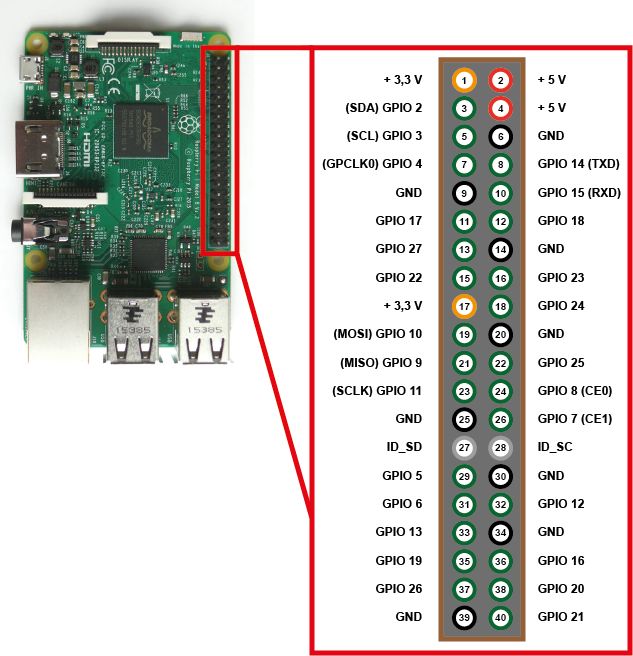
Laboratorio 6 - Puerto Serial y GUI Con Qt PyQt

En este laboratorio empezaremos a utilizar dos partes adicionales. Una relacionada a hardware que es el puerto serial del raspberry pi, útil para equipos que son “legacy” o algunos equipos que tienen interfaz nativa para comunicación.

La siguiente parte tiene que ver con PyQt y QtCreator que es un framework de software muy utilizado en la industria para realizar no solo interfaces gráficas sino también para capa de aplicación, pues posee librerías para cualquier uso de código, incluso otras empresas como V-Play lo utiliza en su motor gráfico de juegos.



Header de Rasberry Pi, para referencia rápida.

**Parte 1 – Puerto Serial**

UART – Puerto serial

El receptor / transmisor asíncrono universal (UART), o puerto serie, es una interfaz de comunicación donde los datos se transmiten en serie en bits desde un sensor a la computadora host. El uso de un puerto serie es una de las formas más antiguas de protocolo de comunicación. Se utiliza en el registro de datos donde los microcontroladores recopilan datos de los sensores y transmiten los datos a través de dicha interfaz. También hay sensores que transmiten datos a través de comunicación en serie como respuestas a los comandos entrantes.

No entraremos en la teoría detrás de las comunicaciones de puerto serie (hay mucha teoría disponible en la Web en <https://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter>).

Discutiremos el uso del puerto serie para conectar diferentes sensores con la Raspberry Pi.

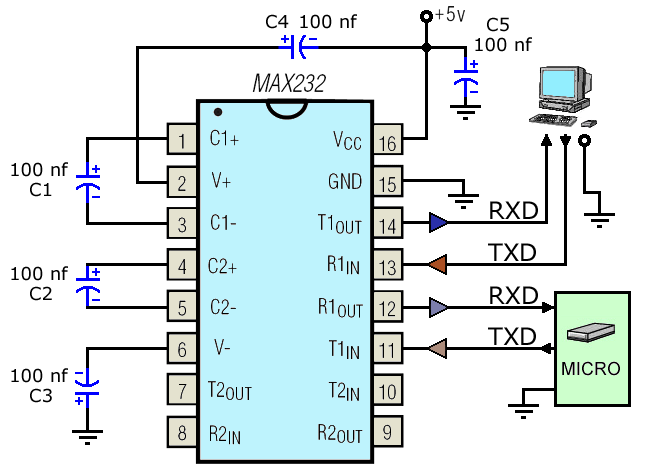
Puerto UART de Raspberry Pi

Normalmente, los puertos UART consisten en un receptor (Rx) y un pin transmisor (Tx) que reciben y transmiten datos. El encabezado GPIO de Raspberry Pi viene con un puerto UART. Los pines GPIO 14 (el pin Tx) y 15 (es el pin Rx) sirven como puerto UART para la Raspberry Pi:

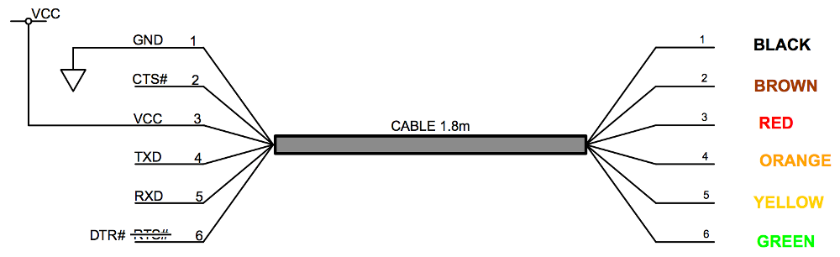
Los pines GPIO 14 y 15 son los pines UART (fuente de la imagen: https://www.rs-online.com/designspark/introducing-the-raspberry-pi-b-plus)

Hardware para comunicar el Raspberry Pi

Usualmente se utiliza el MAX232 que es un IC para realizar “level shifting”, es decir, como el raspberry pi es de señal de 3.3V para las líneas (header), en serial hay picos y valles de -12V a +12V que simbolizan un “1” y un “0”



Otra opción es utilizar un cable FTDI para realizar comunicación, esta sería la opción más cómoda pues se crearía un puerto virtual en la PC como un COM port y se cablearían solamente 3 cables (mínimo) para realizar la comunicación.



Configurando el puerto serial de Raspberry Pi Zero

Para poder utilizar el puerto serie para hablar con los sensores, el inicio de sesión / consola del puerto serie debe estar deshabilitado. En la imagen del sistema operativo Raspbian, esto está habilitado de forma predeterminada, ya que permite una fácil depuración.

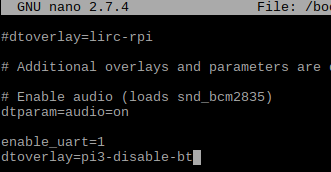
El inicio de sesión del puerto serie se puede desactivar a través del archivo config.txt …

Habilitar solo terminal serial por ttyAMA0

sudo apt-get install python-serial

pip install pyserial

sudo nano /boot/config.txt



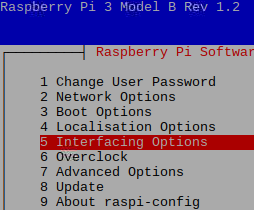
La línea dtoverlay desabilita Bluetooth de la línea serial

sudo reboot –h now

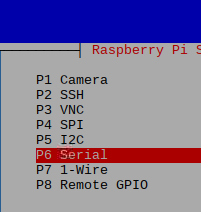
… o por modificación usando la terminal.

sudo raspi-config

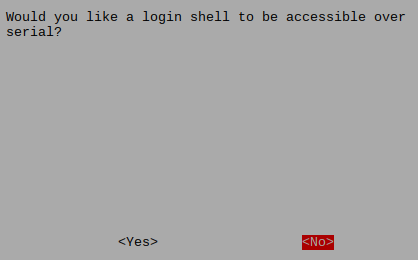
Seleccione opciones de interfaz en el menú principal de raspi-config:



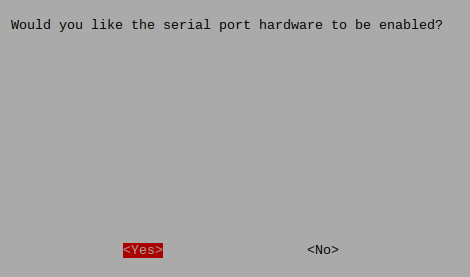
Seleccione Serial en el menú de configuración de raspi



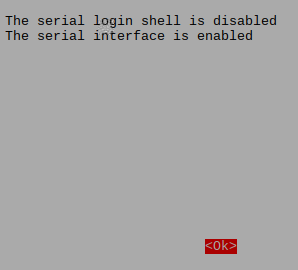
Seleccione deshabilitar el “login Shell” por serial:



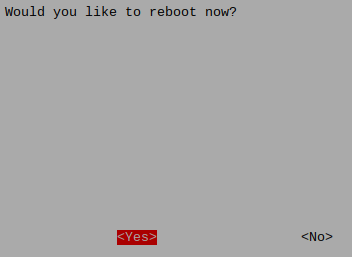
Seleccione mantener habilitado por hardware el puerto serie



Se confirmará lo que está habilitado o no:



Pedirá realizar el reinicio, ejecútelo



Código Python para la comunicación del puerto serie.

Programa de Prueba de eco de byte. Nombrar el archive SerialPort.py, no llamar serial.py, dará error.

Esperar unos segundos luego de arrancar a que se estabilice el serial

import time

import serial

print "Starting program"

ser = serial.Serial('/dev/ttyAMA0', baudrate=9600,

parity=serial.PARITY\_NONE,

stopbits=serial.STOPBITS\_ONE,

bytesize=serial.EIGHTBITS

)

time.sleep(1)

try:

ser.write('Hello World\r\n')

ser.write('Serial Communication Using Raspberry Pi\r\n')

ser.write('By: Embedded Laboratory\r\n')

print 'Data Echo Mode Enabled'

while True:

inbuffer = ser.readline()

print(inbuffer)

ser.write(inbuffer)

except KeyboardInterrupt:

print "Exiting Program"

except:

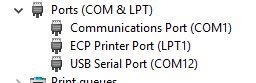
print "Error Occurs, Exiting Program"

finally:

ser.close()

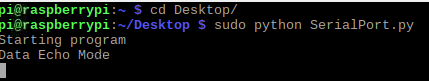
pass

Verificar en el “Device Manager” el puerto COM abierto, en mi caso el COM 12

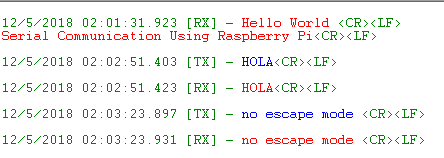


Utilice un programa de terminal serial como CoolTerm, Teraterm, Putty, Docklight Scripting para ver la salida por serial

Correr el programa serial de raspberry pi



Ver y enviar datos de la terminal



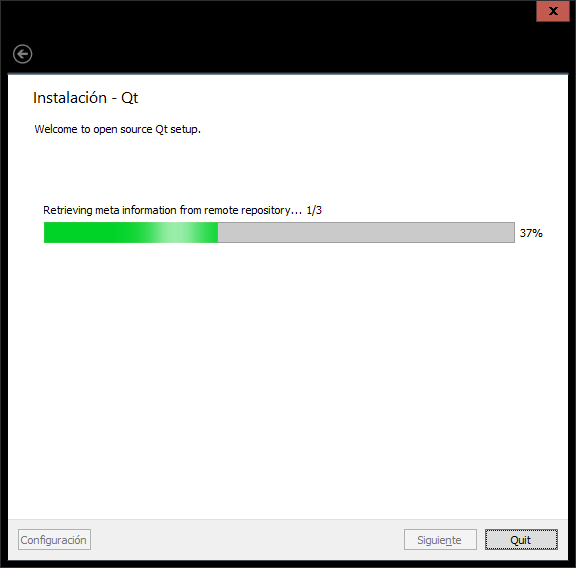
**Qt Creator**

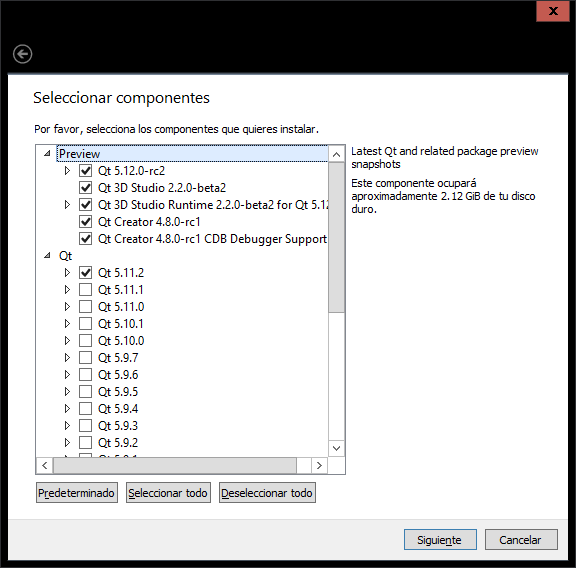
Qt es framework gigantesco para entregar aplicaciones sofisticadas en una multitud de plataformas. Se usa ampliamente en dispositivos integrados, como televisores, decodificadores satelitales, equipos médicos, tableros de instrumentos para automóviles y mucho más. También tiene una rica historia en el mundo de Linux, con KDE y Sailfish OS que la usan extensivamente y muchas aplicaciones en las tiendas que se desarrollan usando Qt. También ha hecho grandes avances en el campo móvil en los últimos años. Sin embargo, para Microsoft Windows y Apple Mac OS X, el predominio de C # / .NET y Objective-C / Cocoa significa que Qt a menudo se pasa por alto.

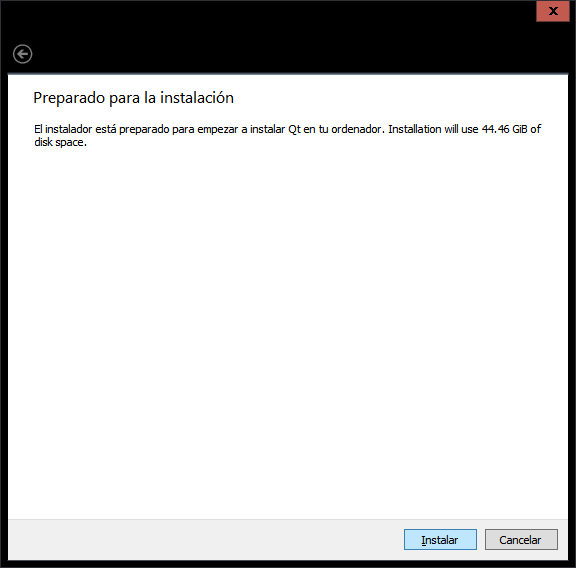
El objetivo de esta sección es demostrar el poder y la flexibilidad del marco Qt y mostrar cómo puede escribir su aplicación una vez e implementarla en varios escritorios de sistemas operativos.

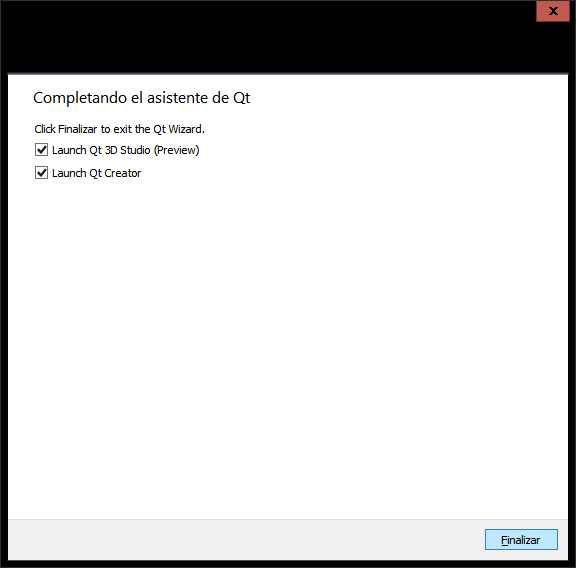
En esta sección, instalaremos y configuraremos el Qt y el QtCreator asociado del entorno de desarrollo integrado (IDE). Crearemos una sencilla aplicación de bloc de notas que utilizaremos durante el resto del libro para demostrar varias técnicas.

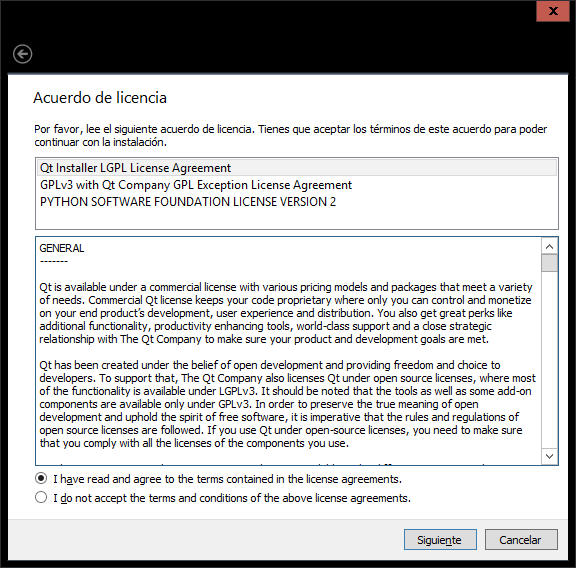
Descargue e instale qt como se muestra en las siguientes figuras

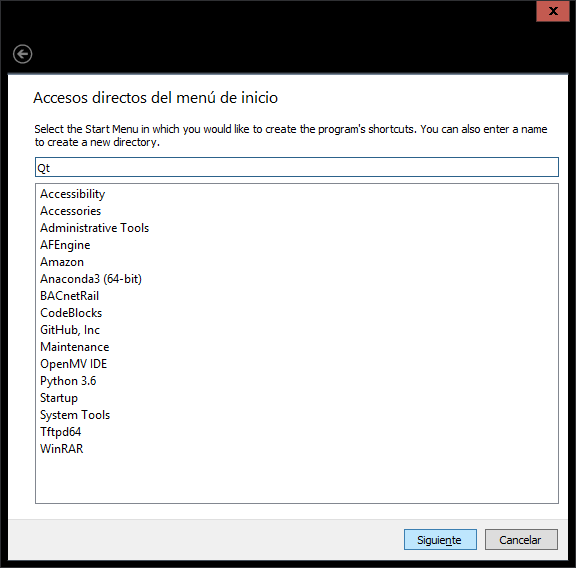










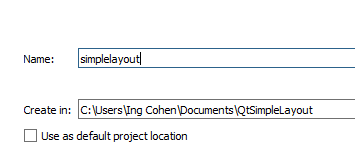


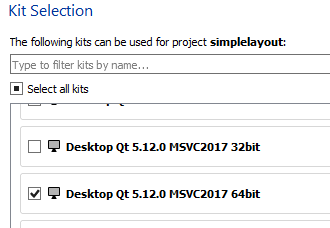
Realizar un proyecto como sigue:

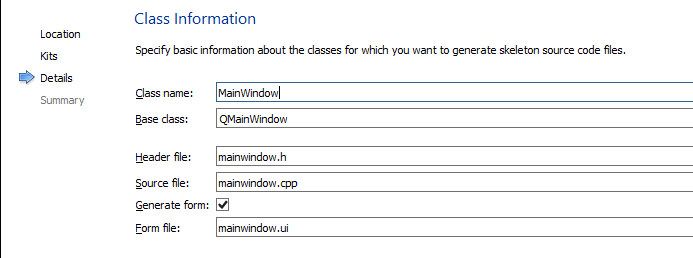


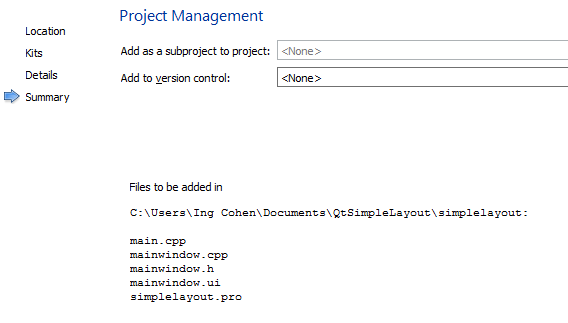


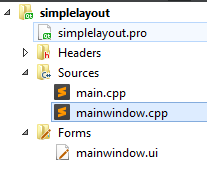
Crear la carpeta QtSimpleLayout

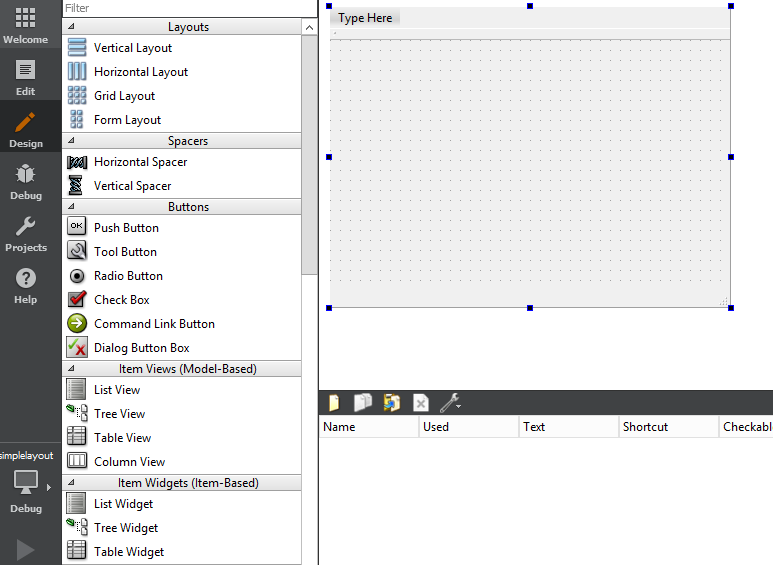


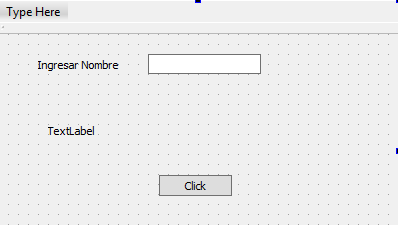




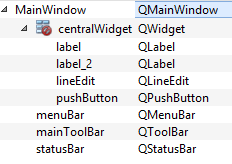








La ventana de nombres de objetos debe quedar como sigue:



Ejecutar el HMI par observar como queda presentado



Tanto para la PC como el raspberry pi instale pyqt5 si no lo ha realizado

pip installa pyqt5



Para la ejecución en Qt no debe tener espacios la carpeta, poner en una ruta + carpeta sin espacios



Copie el siguiente código para llamar a la ventana y realizar la aplicación, grábelo como main.py

import sys

from PyQt5.QtWidgets import \*

from mainwindow import \*

class MyForm(QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.ui = Ui\_MainWindow()

self.ui.setupUi(self)

self.ui.pushButton.clicked.connect(self.dispmessage)

self.show()

def dispmessage(self):

self.ui.label\_2.setText("Hello "

+self.ui.lineEdit.text())

if \_\_name\_\_=="\_\_main\_\_":

app = QApplication(sys.argv)

w = MyForm()

w.show()

sys.exit(app.exec\_())

Llame la aplicación

sudo python main.py

**Utilitarios**

**Realizar una salida de audio de un tono.**

import subprocess

WAV\_FOLDER = "/home/pi/tone.wav"

class TonePlayer(object):

"""Se puede usar este código si quiere que emita un tono cuando booteo el sistema"""

def \_\_init\_\_(self, file\_name):

self.file\_name = file\_name

def set\_volume(self, value):

"""set tone sound volume"""

subprocess.Popen(["amixer", "set", "'PCM'", str(value)], shell=False)

def play(self):

"""play the wav file"""

subprocess.Popen(["aplay", self.file\_name], shell=False)

if \_\_name\_\_ == "\_\_main\_\_":

tone\_player = TonePlayer(WAV\_FOLDER)

tone\_player.set\_volume(75)

tone\_player.play()

Conecte unas bocinas y ejecute

python /home/pi/player.py

**Realizar append a archivos (el archivo tiene un contenido pero queremos agregar algo)**

if \_\_name\_\_ == "\_\_main\_\_":

# abre un archivo para lectura

file = open('write\_file.txt', 'a')

# lee una linea de un archivo

file.write('This is a line appended to the file\n')

file.close()

file = open('write\_file.txt', 'r')

data = file.read()

print(data)

file.close()

**Realizar un archivo de configuración que luego será leído o modificado**

**import** **configparser**

**if** \_\_name\_\_ == "\_\_main\_\_":

*# Inicializa el config parser*

config\_parser = configparser.ConfigParser()

*# Creamos el archivo de configuración*

**with** open('raspi.cfg', 'w') **as** config\_file:

*#Añadimos una sección llamada AppInfo*

config\_parser.add\_section('AppInfo')

*#Añadimos datos de archivo de configuración*

config\_parser.set('AppInfo', 'id', '123')

config\_parser.set('AppInfo', 'gpio', '2')

config\_parser.set('AppInfo', 'debug\_switch', 'True')

config\_parser.set('AppInfo', 'sensor\_address', '0x62')

*#Añadimos una sección llamada credenciales*

config\_parser.add\_section('Credentials')

config\_parser.set('Credentials', 'token', 'abcxyz123')

config\_parser.write(config\_file)

**print**("Config File Creation Complete")

**Realizar un programa que cambie nuestro archivo de configuración**

**import** **configparser**

**if** \_\_name\_\_ == "\_\_main\_\_":

*# Inicializa el ConfigParser*

config\_parser = configparser.ConfigParser()

*# Lee el archivo raspi.cfg*

config\_parser.read('raspi.cfg')

*# Escribe el appinfo*

config\_parser.set('AppInfo', 'fw\_version', 'A3')

*# al final grabamos en el archivo de configuración*

**with** open('raspi.cfg', 'w') **as** config\_file:

config\_parser.write(config\_file)

**Realizar un programa que haga lectura de un archivo de configuración**

**if** \_\_name\_\_ == "\_\_main\_\_":

*# abre el archivo a leer*

file = open('write\_file.txt', 'a')

*# lee una línea del archivo*

file.write('This is a line appended to the file**\n**')

file.close()

file = open('write\_file.txt', 'r')

data = file.read()

**print**(data)

file.close()

**Leer un archivo csv (también se puede utilizar la librería pandas)**

**import** **csv**

**if** \_\_name\_\_ == "\_\_main\_\_":

*# inicializa la lectura*

**with** open("csv\_example.csv", 'r') **as** csv\_file:

csv\_reader = csv.reader(csv\_file)

**for** row **in** csv\_reader:

**print**(row)

**Escribir un archivo csv**

**import** **csv**

**if** \_\_name\_\_ == "\_\_main\_\_":

*# inicializa la escritura*

**with** open("csv\_example.csv", 'w') **as** csv\_file:

csv\_writer = csv.writer(csv\_file)

csv\_writer.writerow([123, 456, 789])

csv\_writer.writerow(["Red", "Green", "Blue"])

**Identificamos una lista de archivos de diferentes patrones**

**import** **glob**

**if** \_\_name\_\_ == "\_\_main\_\_":

*# lista los archivos*

**for** file **in** glob.glob('\*.py'):

**print**(file)

**for** file **in** glob.glob('txt\_files/file1[0-9][0-9].txt'):

**print**(file)

**Manipulando existencia de archivos, carpetas, borra y mata un proceso. El ID del proceso se puede averiguar de la ventana de comandos con “ps aux” en una terminal**

**import** **os**

**import** **signal**

**if** \_\_name\_\_ == "\_\_main\_\_":

*# Verifica si existe un archivo*

**if** os.path.isfile('/home/pi/Desktop/code\_samples/write\_file.txt'):

**print**('The file exists!')

**else**:

**print**('The file does not exist!')

*# confirma la existencia de una carpeta*

**if** os.path.isdir('/home/pi/Desktop/folder\_in\_desktop):

**print**('The directory exists!')

**else**:

**print**('The directory does not exist!')

*#Elimina archivo del filesystem*

os.remove('/home/pi/Desktop/code\_samples/read\_file.txt')

*# Nota: Debe cambiar el PID del proceso que quiere matar, para saber el id del proceso*

*# ejecutar en la terminal el comando ps aux*

**try**:

os.kill(1815, signal.SIGKILL)

**except** **OSError** **as** error:

**print**("OS Error " + str(error))

**Llamando un subproceso llamado “programa” desde un archivo de python (llamar un programa)**

**import** **subprocess**

**if** \_\_name\_\_ == "\_\_main\_\_":

subprocess.Popen(['programa', 'tone.wav'])

**Leyendo argumentos de entrada. Tambien se puede realizar con la librería ArgParser.**

**import** **sys**

**if** \_\_name\_\_ == "\_\_main\_\_":

**with** open(sys.argv[1], 'r') **as** read\_file:

**print**(read\_file.read())

**Leyendo argumentos de entrada con argparser**

import argparse

if \_\_name\_\_ == "\_\_main\_\_":

# construct the argument parser and parse the arguments

ap = argparse.ArgumentParser()

ap.add\_argument("-p", "--plot", type=str, default="plot.png", \

help="path to output loss/accuracy plot")

args = vars(ap.parse\_args())

print(args)

**Web Frameworks**

**Tomando la estampa de tiempo desde una página de internet**



import requests

if \_\_name\_\_ == "\_\_main\_\_":

# Fuente: http://stackoverflow.com/a/30635751/822170

try:

response = requests.get('http://nist.time.gov/actualtime.cgi')

print(response.text)

except requests.exceptions.ConnectionError as error:

print("Something went wrong. Try again")

**Conectándose a una página de internet para tomar el dato de temperatura**

**Nota: Favor sacar su propio API\_KEY logueandose a OpenWeatherMap**



**import** **requests**

*# generate your own API key*

APP\_ID = '7bf1a7ed318cef61f32bdd193d33fd2b'

ZIP = 94103

URL = """http://api.openweathermap.org/data/2.5/weather?zip={}&appid={}**\**

&units=imperial""".format(ZIP, APP\_ID)

**if** \_\_name\_\_ == "\_\_main\_\_":

*# Documentación de funciones: http://openweathermap.org/current#current\_JSON*

**try**:

*# Data codificada y decodificada en format JSON (Java Script Object Notation)*

response = requests.get(URL)

json\_data = response.json()

**print**("Temperature is **%s** degrees Fahrenheit" % json\_data['main']['temp'])

**except** requests.exceptions.ConnectionError **as** error:

**print**("The error is **%s**" % error)

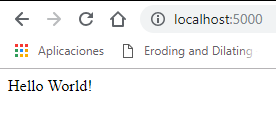
Abriendo en flask un servidor

Nota: Primero instalar flask

pip intall flask

Luego correr el código

Finalmente abrir el navegador y poner el ip del raspberry pi al puerto 5000, por ejemplo: 192.168.2.38:5000. Puede encontrar su ip con ifconfig



**from** **flask** **import** Flask

app = Flask(\_\_name\_\_)

@app.route("/")

**def** hello():

**return** "Hello World!"

**if** \_\_name\_\_ == "\_\_main\_\_":

app.run('0.0.0.0')

**Abriendo en flask un servidor**

Ejecutando controles desde una página web, por ejemplo ingresar



**from** **flask** **import** Flask

**import** **RPi.GPIO** **as GPIO**

GPIO.setmode(GPIO.BOARD)

GPIO.setup(12, GPIO.OUT)

app = Flask(\_\_name\_\_)

@app.route("/lamp/<bombillo>")

**def** control(control):

**if** control == "on":

GPIO.output(12, GPIO.HIGH)

**elif** control == "off":

GPIO.output(12, GPIO.LOW)

**return** "La lampara esta **%s**" % control

**if** \_\_name\_\_ == "\_\_main\_\_":

app.run('0.0.0.0')

**Camara Raspberry Pi**

Realizando una captura con raspberry pi

**import** **cv2**

**from** **picamera** **import** PiCamera

**from** **time** **import** sleep

**if** \_\_name\_\_ == "\_\_main\_\_":

**with** PiCamera() **as** camera:

camera.vflip = True

camera.start\_preview()

sleep(10)

camera.capture("/home/pi/Desktop/desktop\_shot.jpg")

camera.stop\_preview()

img = cv2.imread("/home/pi/Desktop/desktop\_shot.jpg", cv2.IMREAD\_GRAYSCALE)

cv2.imshow("image", img)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Multithreads en Python**

Ejecuta tareas en paralelo

import threading

import time

def sensor\_processing(string):

for num in range(5):

time.sleep(5)

print("%s: Iteration: %d" %(string, num))

if \_\_name\_\_ == '\_\_main\_\_':

thread\_1 = threading.Thread(target=sensor\_processing, args=("Sensor 1",))

thread\_1.start()

thread\_2 = threading.Thread(target=sensor\_processing, args=("Sensor 2",))

thread\_2.start()

thread\_3 = threading.Thread(target=sensor\_processing, args=("Sensor 3",))

thread\_3.start()

**Flask Web Control de un “Robot diferencial” (Carro)**

Nota: Se necesita el archivo posterior a este código para ejecutar esta sección. Si lo quiere ejecutar desde una página web en específico copiar el archivo index.html e invocarlo en el navegador.

from flask import Flask, render\_template, request, redirect

from robot import Robot

app = Flask(\_\_name\_\_)

my\_robot = Robot(1,2)

@app.route("/")

def hello():

return render\_template('index.html')

@app.route('/forward', methods = ['POST'])

def forward():

my\_robot.forward(0.25)

return redirect('/')

@app.route('/reverse', methods = ['POST'])

def reverse():

my\_robot.reverse(0.25)

return redirect('/')

@app.route('/left', methods = ['POST'])

def left():

my\_robot.left(0.25)

return redirect('/')

@app.route('/right', methods = ['POST'])

def right():

my\_robot.right(0.25)

return redirect('/')

if \_\_name\_\_ == "\_\_main\_\_":

app.run('0.0.0.0')

Archivo de robot diferencial Robot.py utilizando el adafruit motor hat

import time

from Adafruit\_MotorHAT import Adafruit\_MotorHAT

class Robot(object):

def \_\_init\_\_(self, left\_channel, right\_channel):

self.motor = Adafruit\_MotorHAT(0x60)

self.left\_motor = self.motor.getMotor(left\_channel)

self.right\_motor = self.motor.getMotor(right\_channel)

def set\_speed(self):

self.left\_motor.setSpeed(200)

self.right\_motor.setSpeed(200)

def stop(self):

self.left\_motor.run(Adafruit\_MotorHAT.RELEASE)

self.right\_motor.run(Adafruit\_MotorHAT.RELEASE)

def forward(self, duration):

self.set\_speed()

self.left\_motor.run(Adafruit\_MotorHAT.FORWARD)

self.right\_motor.run(Adafruit\_MotorHAT.FORWARD)

time.sleep(duration)

self.stop()

def reverse(self, duration):

self.set\_speed()

self.left\_motor.run(Adafruit\_MotorHAT.BACKWARD)

self.right\_motor.run(Adafruit\_MotorHAT.BACKWARD)

time.sleep(duration)

self.stop()

def left(self, duration):

self.set\_speed()

self.right\_motor.run(Adafruit\_MotorHAT.FORWARD)

time.sleep(duration)

self.stop()

def right(self, duration):

self.set\_speed()

self.left\_motor.run(Adafruit\_MotorHAT.FORWARD)

time.sleep(duration)

self.stop()

if \_\_name\_\_ == "\_\_main\_\_":

# Crea una instancia de la clase Robot con dos canals de motor

robot = Robot(1,2)

print("Moving forward...")

robot.forward(5)

print("Moving backward...")

robot.reverse(5)

robot.stop()

Página para inicio en flask (index.html)

<!DOCTYPE html>

<html>

<head>

<title>Raspberry Pi</title>

</head>

<body>

<iframe id="stream" src="http://192.168.86.111:8081/?action=stream" width="320" height="240"></iframe>

<table style="width:100%; max-width: 500px; height:300px;">

<tr>

<td>

<form action="/forward" method="POST">

<input type="submit" value="forward" style="float:left; width:80% ;">

</br>

</form>

</td>

<td>

<form action="/reverse" method="POST">

<input type="submit" value="reverse" style="float:left; width:80% ;">

</br>

</form>

</td>

<td>

<form action="/left" method="POST">

<input type="submit" value="left" style="float:left; width:80% ;">

</br>

</form>

</td>

<td>

<form action="/right" method="POST">

<input type="submit" value="right" style="float:left; width:80% ;">

</br>

</form>

</td>

</tr>

</table>

</body>

</html>

I - Haga un chat serial entre dos raspberry pi usando minicom

1 – Instalar minicom: sudo apt-get install minicom

2 – Arrancar minicom: minicom –b 9600 –o –D /dev/ttyAMA0

Nota: La otra RPi debe tener la misma velocidad en baudios GND debe ser conectados entre sí, RPi1.TX a RPi2.RX y RPi1.RX a RPi2.TX

3 – Conecte el raspberry Pi a Matlab usando el puerto Serial

4 – Pruebe encender un LED con el Raspberry Pi utilizando QtCreator cuando se hace un click