Universidad Iberoamericana Puebla

Departamento de ingenierías



**Proyecto Final: Sistema embebido usando sensor BME280**

Arellano Jacinto Diego Alexis

Campos Rosales Abraham

Enciso Torres Raúl

Arquitecturas Programables Avanzadas

Prof. Rafael Pérez Aguirre

1. de noviembre de 2019

**Proyecto Final: Sistema embebido usando sensor BME280**

1. **Introducción**

El sensor BME280 integra en un solo dispositivo sensores de presión atmosférica, temperatura y humedad relativa, con gran precisión, bajo consumo energético y un formato ultra compacto. Basado en tecnología BOSCH piezo-resistiva con gran robustez EMC, alta precisión y linealidad, así como con estabilidad a largo plazo. Se conecta directamente a un microcontrolador a través de I2C o SPI.La etapa de sensado de presión es similar al sensor BMP280, y por lo tanto es superior al BMP180. En cuando al sensor de humedad relativa presenta un desempeño sobresaliente comparado a los sensores DHT22 o DHT21.

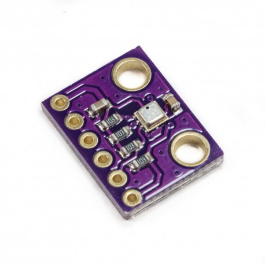
1. **Objetivo general**

Desarrollar sistema embebido que obtenga variables ambientales como presión, temperatura y humedad usando el sensor BME280.

1. **Objetivos específicos**

* Hacer la programación en Python tomando en cuenta las especificaciones del sensor BME 280.
* Conectar el circuito siguiendo los pines de sensor ya elegido, usando las librerías correspondientes y en Python hacer la programación usando la Raspberry.
* Diseñar una interfaz gráfica en QtDesginer , el cual actuará como dashboard en donde se moestrarán los datos en tiempo real.
* Generar una base de datos para que las variables se almacenen en la misma.
* Reportar datos y conclusiones.

1. **Material**



Sensor BME280

Cable de Red

Raspberry



Monitor o laptop

VNC Viewer instalado

1. **Marco teórico**

**5.1 Raspberry**

Raspberry Pi es un miniordenador de pequeñas dimensiones y precio destinado principalmente al desarrollo de pequeños prototipos y a estimular la enseñanza de las ciencias de la computación en los centros educativos. Desarrollado en hardware libre cuenta con sistemas operativos GNU/Linux como Raspbian, aunque podemos encontrar otros sistemas operativos optimizados para el hardware de la Raspberry Pi.

Raspberry Pi utiliza una arquitectura para el procesador ARM distinta a la que estamos acostumbrados a utilizar en nuestros ordenadores de sobremesa o portátiles. Esta arquitectura es de tipo RISC (Reduced Instruction Set Computer), es decir, utiliza un sistema de instrucciones realmente simple lo que le permite ejecutar tareas con un mínimo consumo de energía.

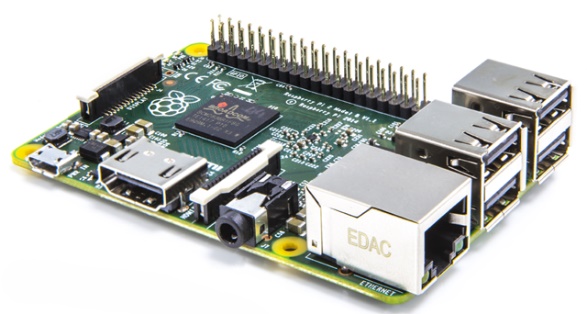


Figura 1. Raspberry

**5.3 Servidores Web**

Es un programa que gestiona cualquier aplicación en el lado del servidor realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente generando una respuesta en cualquier lenguaje o aplicación en el lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un Navegador Web. Para la transmisión de todos estos datos se utiliza algún protocolo. Generalmente se utiliza el protocolo HTTP para estas comunicaciones, perteneciente a la capa de aplicación del Modelo OSI. El término también se emplea para referirse al ordenador que ejecuta el programa.

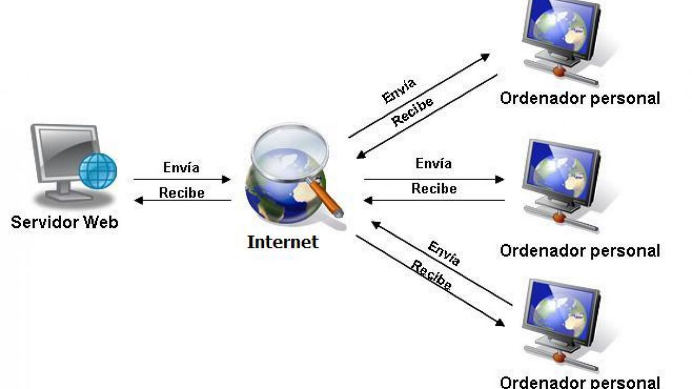


Figura 4. Esquema de comunicación Web

Figura 3. Servidor Web de gran tamaño

**5.4 Sensor BME280**

El BME280 es un pequeño gran sensor si se puede decir así. En un solo encapsulado es capaz de medir temperatura, humedad y presión atmosférica. Se puede comunicar mediante el bus I2C o SPI, así que es compatible con prácticamente cualquier plataforma. También es muy preciso ya que tiene muy poca desviación en cada medición además puede alimentarse tanto con 5V como 3.3V.

El módulo viene con un regulador LM6206 3.3V incorporado y un traductor de nivel de voltaje I2C, por lo que puede usarlo sin problemas con un microcontrolador lógico de 3.3V o 5V como Arduino.

El BME280 consume menos de 1 mA durante las mediciones y solo 5 μA durante inactividad o estado idle. Este bajo consumo de energía permite la implementación en dispositivos que funcionan con baterías, como teléfonos, módulos GPS, drones o relojes.

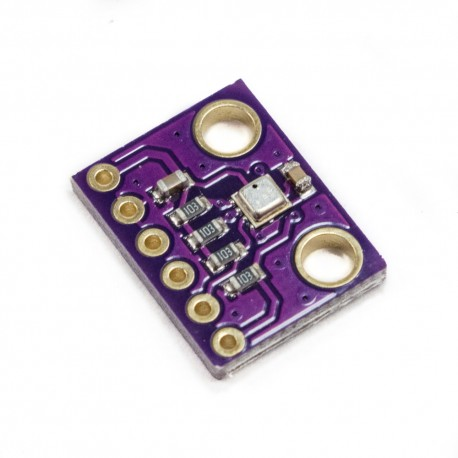


Figura 5. Sensor BME280

**5.5 Plataforma GitHub**

GitHub aloja tu repositorio de código y te brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto. Además de eso, puedes contribuir a mejorar el software de los demás. Para poder alcanzar esta meta, GitHub provee de funcionalidades para hacer un fork y solicitar pulls.

GitHub es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador. El software que opera GitHub fue escrito en Ruby on Rails. Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc. Anteriormente era conocida como Logical Awesome LLC. El código de los proyectos alojados en GitHub se almacena típicamente de forma pública, aunque utilizando una cuenta de pago, también permite hospedar repositorios privados.



Figura 6. Logo GitHub

1. **Desarrollo**

**6.1 Raspberry**

from PyQt5 import QtCore, QtGui, QtWidgets

import smbus2

import bme280

import psycopg2

from matplotlib import pyplot as plt

time from import

conn = psycopg2.connect ( ' dbname = Proyecto\_Raul\_Abraham\_DiegoA ' )

puerto = 1

direccion = 0x 76

bus = smbus2.SMBus (puerto)

parametros\_calibracion = bme280.load\_calibration\_params (bus, direccion)

def sensarTemperatura ():

datos = bme280.sample (bus, direccion, parametros\_calibracion)

temperatura = datos.temperature

temperatura = redonda (temperatura, 3 )

print ( " Temperatura: " + str (temperatura) + " ºC " )

cur = conn.cursor ()

tiempo = int (time.time ())

comando = " insertar en valores de temperatura ( " + str (tiempo) + " , " + str (temperatura) + " ) "

cur.execute (comando)

conn.commit ()

cur.close ()

def sensarPresion ():

datos = bme280.sample (bus, direccion, parametros\_calibracion)

presion = datos.pressure

presionando = redondo (presionando, 3 )

print ( " Presion: " + str (presion) + " hPa " )

cur = conn.cursor ()

tiempo = int (time.time ())

comando = " insertar en valores de presiones ( " + str (tiempo) + " , " + str (presion) + " ) "

cur.execute (comando)

conn.commit ()

cur.close ()

def sensarHumedad ():

datos = bme280.sample (bus, direccion, parametros\_calibracion)

humedad = datos.humedad

humedad = redonda (humedad, 3 )

print ( " Humedad: " + str (humedad) + " % r H " )

cur = conn.cursor ()

tiempo = int (time.time ())

comando = " insertar en valores de humedades ( " + str (tiempo) + " , " + str (humedad) + " ) "

cur.execute (comando)

conn.commit ()

cur.close ()

def medirTodo ():

sensarTemperatura ()

sensarHumedad ()

sensarPresion ()

imprimir ( " " )

def mostrarTemperatura ():

print ( " Mostrando temperatura ... " )

cur = conn.cursor ()

cur.execute ( ' select \* from temperaturas ' )

datos = cur.fetchall ()

tiempo = []

temperatura = []

para dato en datos:

tiempo.append (dato [ 0 ])

temperatura.append (dato [ 1 ])

plt.plot (tiempo, temperatura)

plt.show ()

def mostrarPresion ():

print ( " Mostrando presion ... " )

cur = conn.cursor ()

cur.execute ( ' select \* from presiones ' )

datos = cur.fetchall ()

tiempo = []

presionando = []

para dato en datos:

tiempo.append (dato [ 0 ])

presion.append (dato [ 1 ])

plt.plot (tiempo, presionar)

plt.show ()

def mostrarHumedad ():

print ( " Mostrando humedad ... " )

cur = conn.cursor ()

cur.execute ( ' select \* from humedades ' )

datos = cur.fetchall ()

tiempo = []

humedad = []

para dato en datos:

tiempo.append (dato [ 0 ])

humedad.append (dato [ 1 ])

plt.plot (tiempo, humedad)

plt.show ()

class Ui\_MainWindow ( objeto ):

def setupUi ( self , MainWindow ):

MainWindow.setObjectName ( " MainWindow " )

MainWindow.resize ( 882 , 702 )

self .centralwidget = QtWidgets.QWidget (MainWindow)

self .centralwidget.setObjectName ( " centralwidget " )

self .botonTemperatura = QtWidgets.QPushButton ( self .centralwidget)

self .botonTemperatura.setGeometry (QtCore.QRect ( 60 , 90 , 191 , 91 ))

self .botonTemperatura.setObjectName ( " botonTemperatura " )

self .botonTemperatura.clicked.connect (sensarTemperatura)

self .botonPresion = QtWidgets.QPushButton ( self .centralwidget)

self .botonPresion.setGeometry (QtCore.QRect ( 570 , 90 , 191 , 91 ))

self .botonPresion.setObjectName ( " botonPresion " )

self .botonPresion.clicked.connect (sensarPresion)

self .botonHumedad = QtWidgets.QPushButton ( self .centralwidget)

self .botonHumedad.setGeometry (QtCore.QRect ( 310 , 90 , 191 , 91 ))

self .botonHumedad.setObjectName ( " botonHumedad " )

self .botonHumedad.clicked.connect (sensarHumedad)

self .botonVerTemperatura = QtWidgets.QPushButton ( self .centralwidget)

self .botonVerTemperatura.setGeometry (QtCore.QRect ( 60 , 240 , 191 , 81 ))

self .botonVerTemperatura.setObjectName ( " botonVerTemperatura " )

self .botonVerTemperatura.clicked.connect (mostrarTemperatura)

self .botonVerHumedad = QtWidgets.QPushButton ( self .centralwidget)

self .botonVerHumedad.setGeometry (QtCore.QRect ( 310 , 250 , 191 , 81 ))

self .botonVerHumedad.setObjectName ( " botonVerHumedad " )

self .botonVerHumedad.clicked.connect (mostrarHumedad)

self .botonVerPresion = QtWidgets.QPushButton ( self .centralwidget)

self .botonVerPresion.setGeometry (QtCore.QRect ( 570 , 250 , 191 , 81 ))

self .botonVerPresion.setObjectName ( " botonVerPresion " )

self .botonVerPresion.clicked.connect (mostrarPresion)

self .timer = QtCore.QTimer ( self .centralwidget)

self .timer.setInterval ( 5000 )

self .timer.timeout.connect (medirTodo)

self .timer.start ()

MainWindow.setCentralWidget ( self .centralwidget)

self .menubar = QtWidgets.QMenuBar (MainWindow)

self .menubar.setGeometry (QtCore.QRect ( 0 , 0 , 882 , 28 ))

self .menubar.setObjectName ( " barra de menú " )

MainWindow.setMenuBar ( self .menubar)

self .statusbar = QtWidgets.QStatusBar (MainWindow)

self .statusbar.setObjectName ( " barra de estado " )

MainWindow.setStatusBar ( self .statusbar)

self .retranslateUi (MainWindow)

QtCore.QMetaObject.connectSlotsByName (MainWindow)

def retranslateUi ( self , MainWindow ):

\_translate = QtCore.QCoreApplication.translate

MainWindow.setWindowTitle (\_translate ( " MainWindow " , " MainWindow " ))

self .botonTemperatura.setText (\_translate ( " MainWindow " , " Temperatura " ))

self .botonPresion.setText (\_translate ( " MainWindow " , " Presion " ))

self .botonHumedad.setText (\_translate ( " MainWindow " , " Humedad " ))

self .botonVerTemperatura.setText (\_translate ( " MainWindow " , " Ver temperatura " ))

self .botonVerHumedad.setText (\_translate ( " MainWindow " , " Ver Humedad " ))

self .botonVerPresion.setText (\_translate ( " MainWindow " , " Ver presión " ))

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

import sys

apply = QtWidgets.QApplication (sys.argv)

MainWindow = QtWidgets.QMainWindow ()

ui = Ui\_MainWindow ()

ui.setupUi (MainWindow)

MainWindow.show ()

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

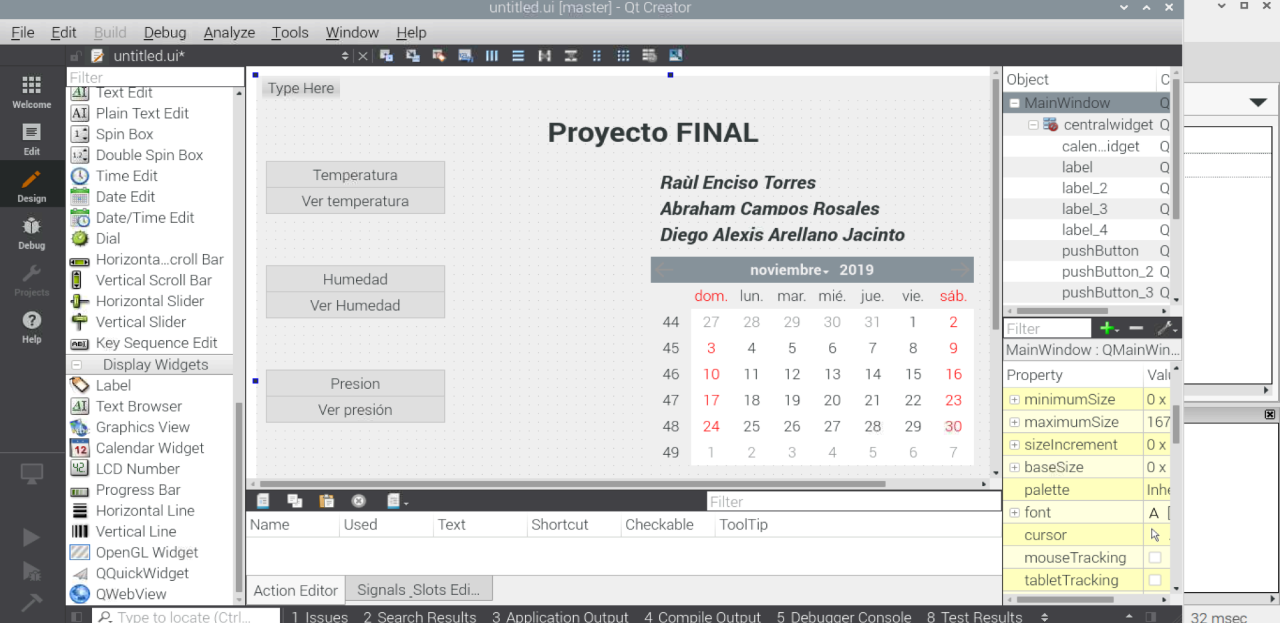


Figura 6. Interfaz en QtDesigner

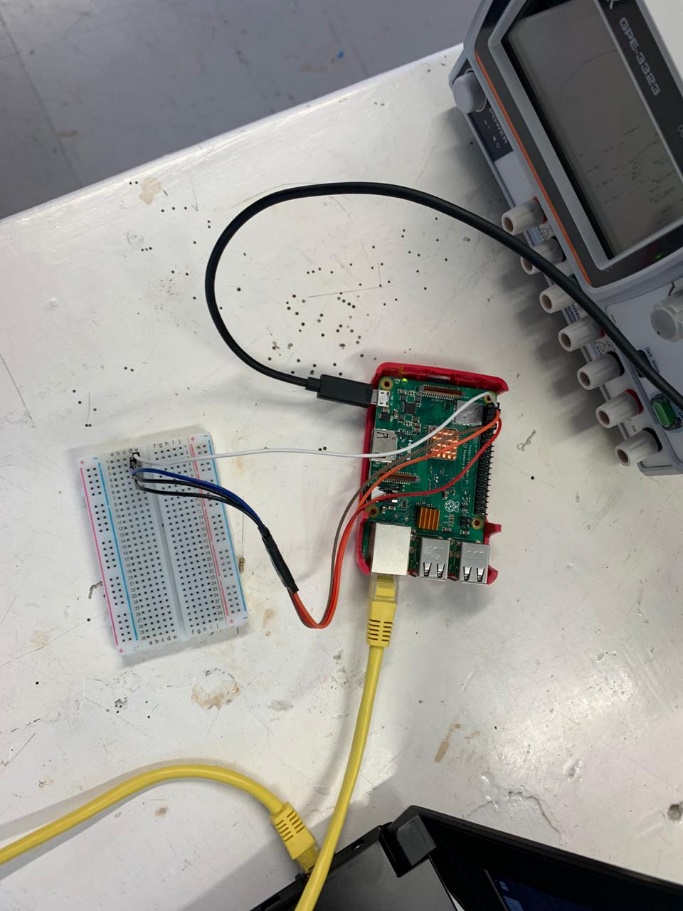
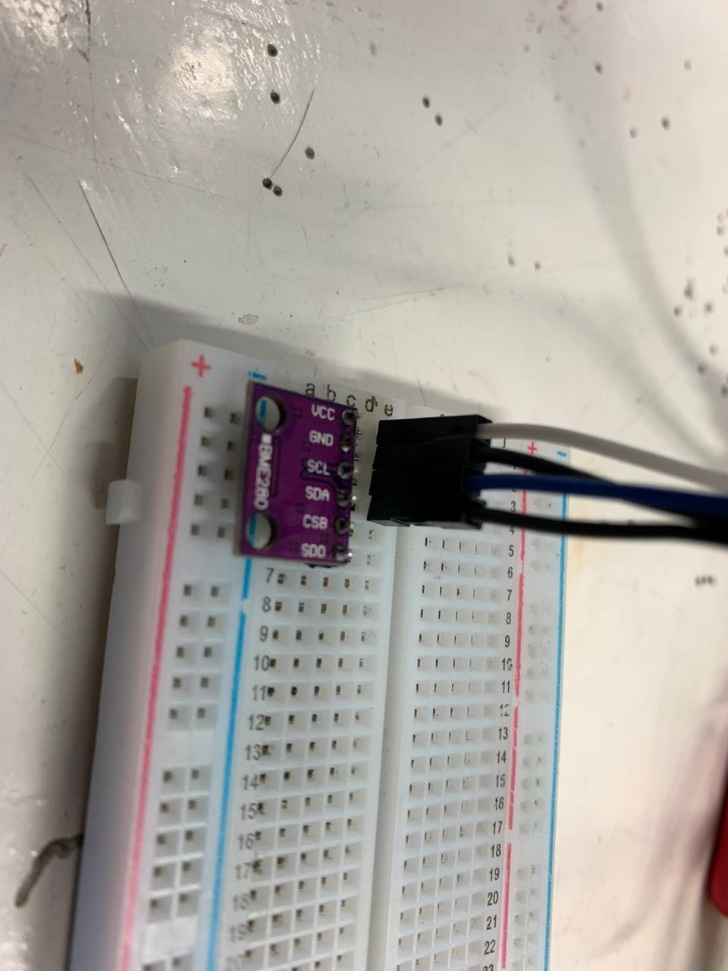


Figura 8. Sensor BME280

Figura 7. Circuito en funcionamiento

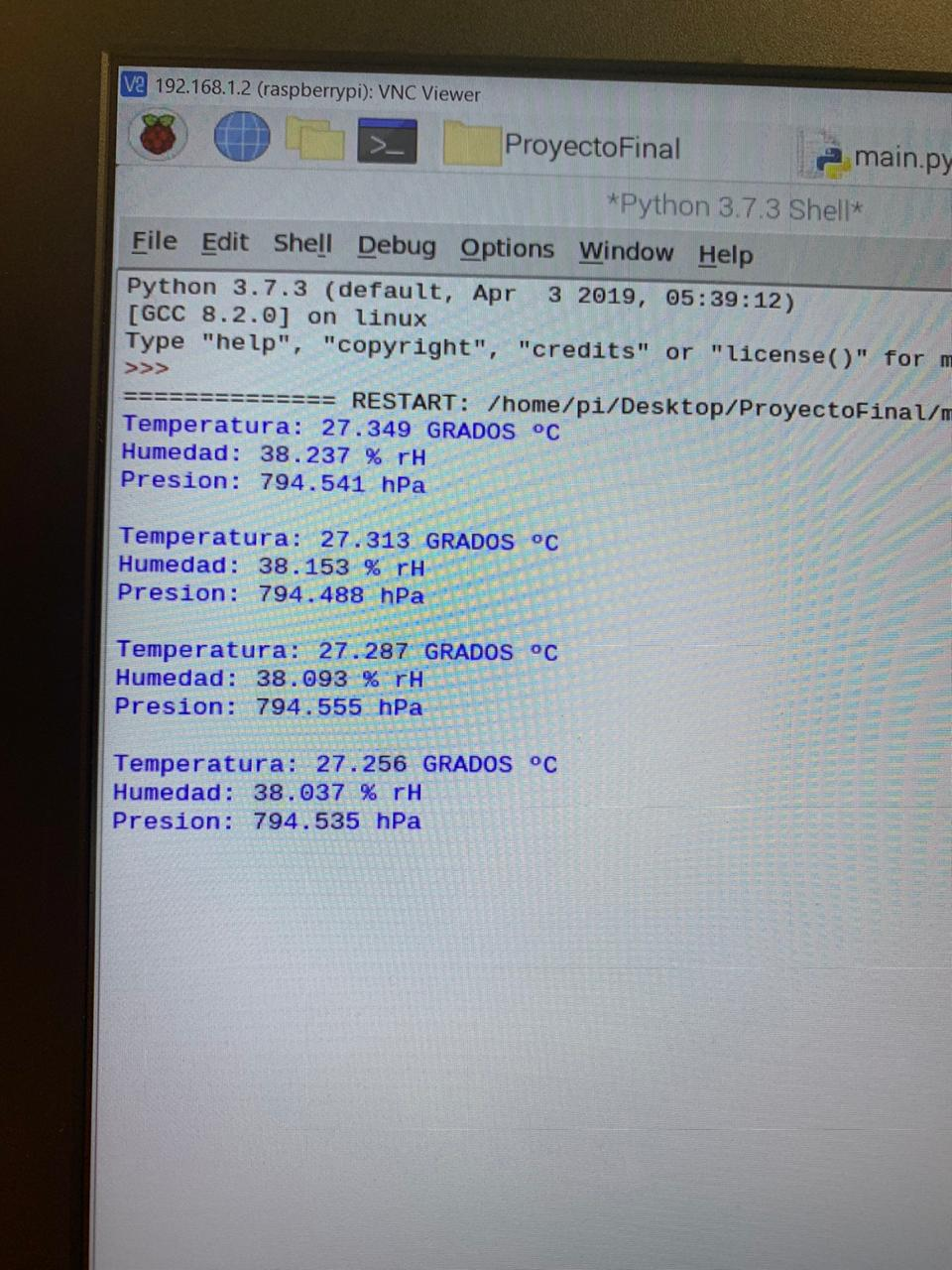


Figura 9. Resultados en el IDLE

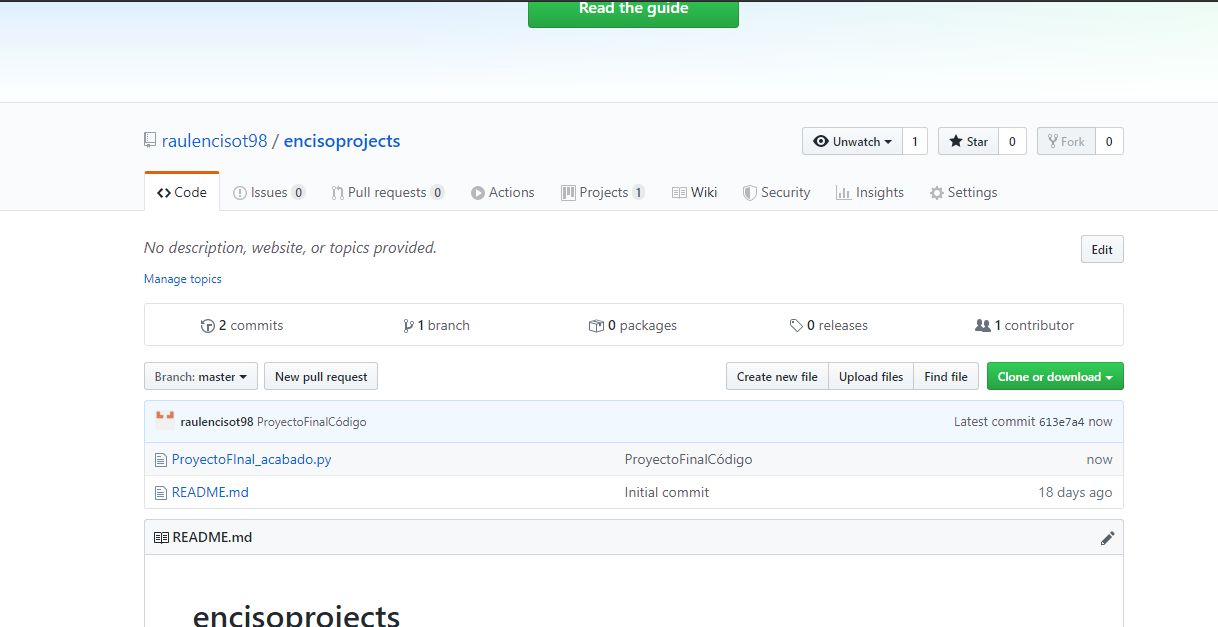


Figura 10. Proyecto subido a Github

[**https://github.com/raulencisot98/encisoprojects.git**](https://github.com/raulencisot98/encisoprojects.git)

[**https://github.com/raulencisot98/encisoprojects.git**](https://github.com/raulencisot98/encisoprojects.git)

[**https://github.com/raulencisot98/encisoprojects.git**](https://github.com/raulencisot98/encisoprojects.git)

[**https://github.com/raulencisot98/encisoprojects.git**](https://github.com/raulencisot98/encisoprojects.git)

[**https://github.com/raulencisot98/encisoprojects.git**](https://github.com/raulencisot98/encisoprojects.git)

[**https://github.com/raulencisot98/encisoprojects.git**](https://github.com/raulencisot98/encisoprojects.git)

**Referencias**

1. Gregorio Fernández Fernández, Curso de Ordenadores. Conceptos básicos de arquitectura y sistemas operativos, (4ª Edición). Servicio de Publicaciones de la E.T.S.I. Telecomunicación de Madrid, 2003. ISBN 84-7402-304-1. [Fernández, 2003]
2. William Stallings, Organización y Arquitectura de Computadores, (5ª Edición). Ed. Prentice-Hall. 2000. ISBN 84-205-2993-1. [Stallings, 2000]
3. A. S. Tanenbaum, Organización de Computadoras. Un enfoque estructurado, (4ª Edición). Pearson Educación. 2000. ISBN 970-17-0399-5. [Tanenbaum, 2000]Logenecker, Justin G. (2007). Administración de Pequeñas Empresas. (13ª Ed.). México Distrito Federal, México: Thomson Editores.
4. Moyer. Mc Guigan, Kretlow. (2003). Administración Financiera Contemporánea. (9ª Ed.). Ed. Thomson. Sounth-Western.