

Gabriela Michelle Martínez Jiménez

María Angélica Medel Jarillas

Miguel Ángel Muñoz Hernández

Proyecto Final

Arquitecturas programables

Profesor: Rafael Pérez Aguirre

Universidad Iberoamericana Puebla

28 de noviembre de 2019

Introducción

El presente proyecto se implementó una estación meteorológica con el sensor BME280, el cual nos permite obtener los valores de temperatura, húmedad, altitud y presión en tiempo real. Los datos obtenidos por el sensor son obtenidos mediante el protocolo de comunicación i2c, la información es procesada y analizada con un script de Python y los datos son graficados por medio de una interfaz gráfica en Qt dentro de la Raspberry; estos datos previo a ser graficados son enviados a una base de datos de nombre Postgres en la cual se pueden observar los datos capturados por el programa

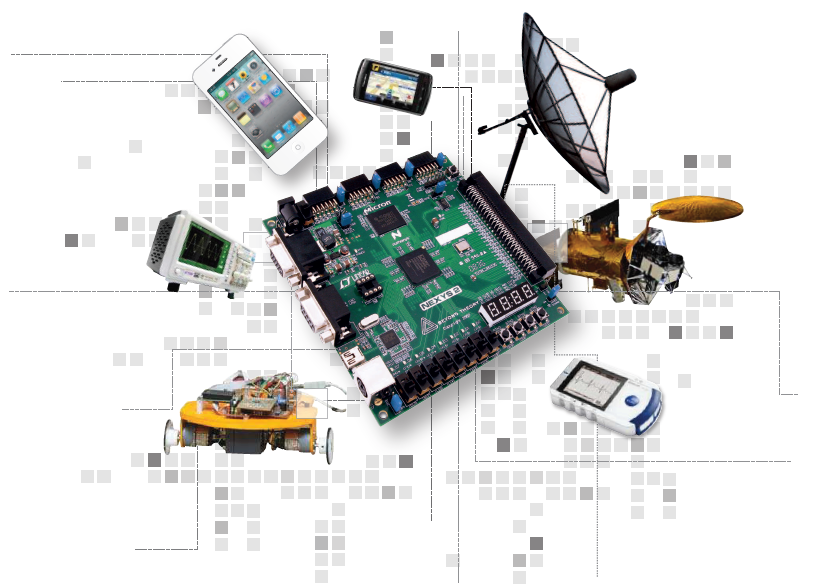
Objetivos

* Desarrollar un sistema embebido que obtenga variables ambientales como presión, temperatura y humedad. Usa el sensor BME280
* Conéctalo a una aplicación escrita en Python para la Raspberry Pi, dicho programa será una interfaz gráfica estilo dashboard, donde se mostrarán los datos actuales en tiempo real y una sección de historial, como tablas o gráficas. (El diseño de la interfaz es a tu consideración, intenta hacerlo lo más agradable posible.)
* Los datos del sensor deberán ser almacenados en una base de datos.

Marco Teórico

1. Sistema embebido

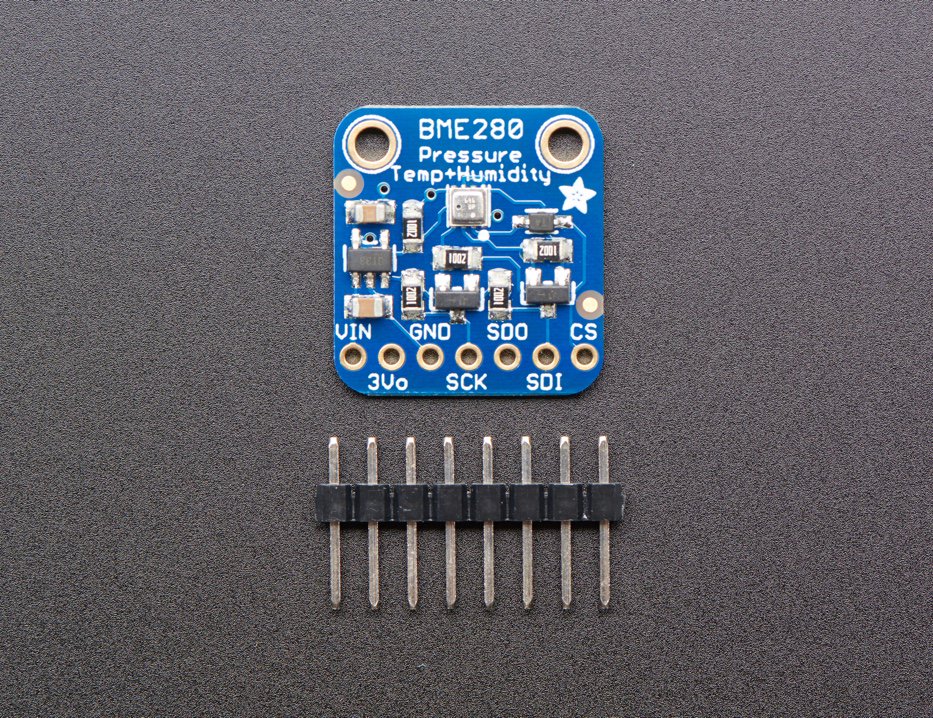
Un sistema embebido es un sistema informático de uso específico que está encapsulado totalmente por el dispositivo que controla. Los sistemas embebidos constituyen un sistema computacional fruto de la combinación de hardware y software. Esta combinación tiene como misión llevar a cabo una funcionalidad o un conjunto de funcionalidades determinadas. Se denominan embebidos o empotrados porque normalmente forman parte de un sistema completo o con funcionalidades más generales.



1. Sensor BME280

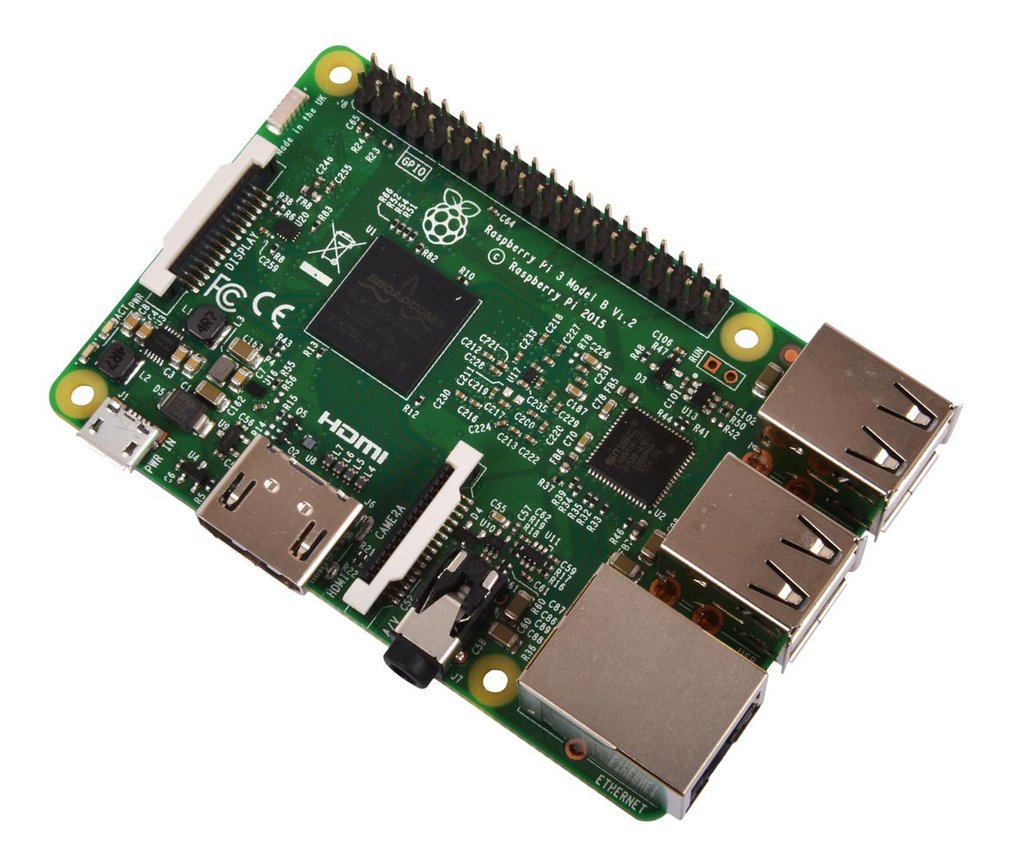
El sensor BME280 integra en un solo dispositivo sensores de presión atmosférica, temperatura y humedad relativa, con gran precisión, bajo consumo energético y un formato compacto. Basado en tecnología BOSCH piezo-resistiva con gran robustez EMC, alta precisión y linealidad, así como con estabilidad a largo plazo. Se conecta directamente a un microcontrolador a través de I2C o SPI.

Este tipo de sensores pueden ser utilizados para calcular la altitud con gran precisión, por lo que es un sensor muy utilizado en sistemas de Autopiloto para Drones (UAVs) entregando medidas de altitud con una precisión de hasta 1m. Otras aplicaciones son: Monitoreo de clima, Internet de las Cosas, Monitor de salud/fitness, Automatización del hogar o Domótica y Aire acondicionado.



1. Raspberry

Raspberry es un computador de placa reducida de bajo costo desarrollado en Reino Unido por la fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas. Aunque es un sistema que no se expresa directamente como un hardware, es un sistema con software open source, trabajando un sistema completamente adaptado a Debian, con una línea independiente llamada Raspbian, es un sistema que opera bajo todo el concepto de Linux.



1. Interfaz Gráfica

Interfaz gráfica de usuario (GUI), acrónimo en inglés de Graphical User Interfase. La interfaz gráfica de usuario es un programa o entorno que gestiona la interacción con el usuario basándose en relaciones visuales como iconos, menús o un puntero.

1. Base de Datos

Es un conjunto de [información](https://concepto.de/informacion/) perteneciente a un mismo contexto, ordenada de modo sistemático para su posterior recuperación, [análisis](https://concepto.de/analisis-3/) y/o transmisión.

1. GitHub

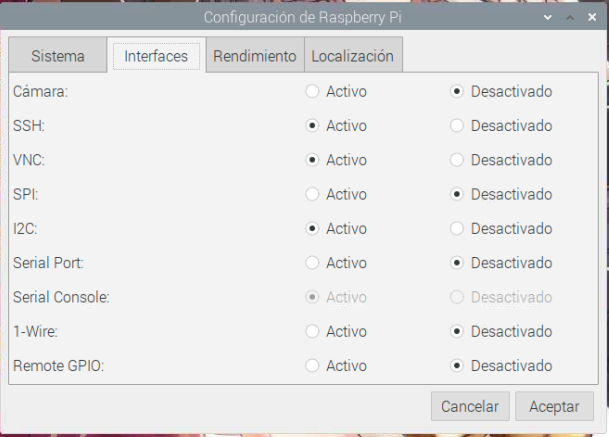
GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones [Git](http://es.wikipedia.org/wiki/Git). El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago.

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.



Desarrollo

Se activo en la raspberry el protocolo de comunicación de i2c, el cual permite utilizar dispositivos que se comuniquen en direcciones específicas, además nos permite conocer los dispositivos que estén conectados a la Raspberry vía pines SDA y SCL



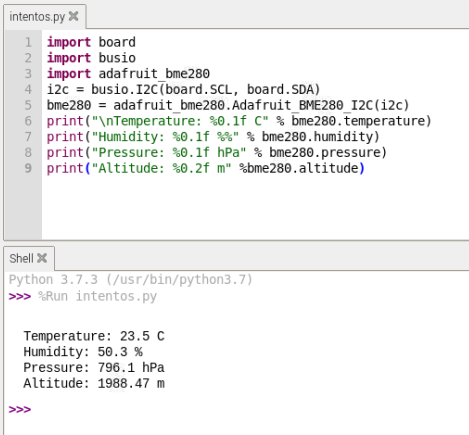
Se realizaron pruebas al sensor BME-280 debido a que existen 2 versiones de este modelo. El módulo de Adafruit y el módulo genérico.

|  |  |
| --- | --- |
| Resultado de imagen para bme280 adafruit | Resultado de imagen para bme280 |
| Sensor BME280 de Adafruit | Sensor BME 280 genérico |

Para el cual Adafruit desarrolló librerías y configuración especial para su modulo, que no es compatible con la versión genérica. Ya que no existen como tal librerías para el módulo de sensor genérico y se deben modificar las otras para el correcto funcionamiento del sensor



El modulo de Adafruit trabaja en la dirección 0x77 y el módulo genérico en la dirección 0x76, por lo que, al ejecutar el código, se debe considerar la dirección correcta.



Una vez operando el sensor se hizo el diseño de la interfaz grafica en QTdesigner donde se configuraron los bloques “Graphics View” como “Plot Widget”

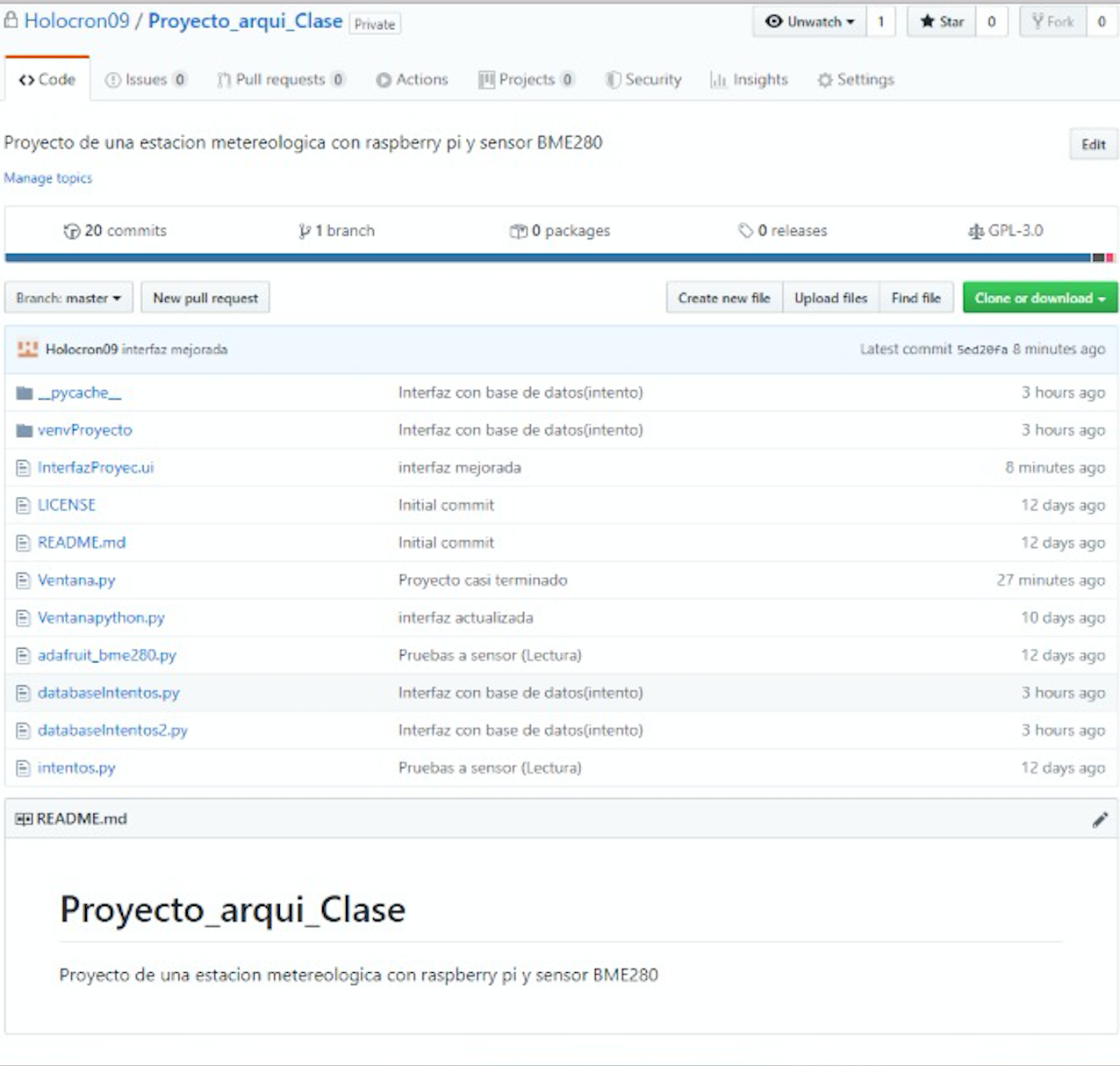
|  |  |
| --- | --- |
|  |  |

Dicha interfaz fue convertida de un tipo de archivo .UI a .py

Mediante la instrucción “ pyuic5 -x InterfazProyec.ui -o Ventanapython.py “, lo que nos permitió trabajar con la interfaz mediante PyQt en Python.

Desde este punto cada modificación y cambio fue actualizado directamente con la plataforma GitHub

<https://github.com/Holocron09/Proyecto_arqui_Clase.git>



El código se puede dividir en distintas secciones

|  |  |
| --- | --- |
| from Ventanapython import \*  from PyQt5.QtCore import QTimer  from pyqtgraph import PlotWidget, plot  import pyqtgraph as pg  import board  import busio  import adafruit\_bme280  import psycopg2  from psycopg2 import Error | En el bloque de importaciones, agregamos las librerias que utilizaremos durante el proyecto. Ventanapython es la interfaz grafica, QTimer de PyQt5.QtCore nos permite generar temporizadores de actualización de contenido de interfaz  Pyqtgraph es una librería para graficar en Qt  Board, Busio y adafruit\_bme280 son las librerias que permiten la comunicación via i2c con el sensor BME280  La librería pssycopg2 nos ayuda a establecer, crear agregar y modificar bases de datos |
| try:  i2c = busio.I2C(board.SCL, board.SDA)  bme280 = adafruit\_bme280.Adafruit\_BME280\_I2C(i2c)  except:  print("no sensor attatched") | Mediante estas líneas nos intentamos conectar con el sensor y establecer la forma en la que la raspberry se intentará comunicar con el dispositivo i2c |
| Temperatura=0  Humedad=0  Presion=0  Altitud=0  lock=0  val=0  hour=[]  temp=[]  hum=[]  pres=[]  Alt=[] | Aquí se declaran las variables para trabajarlas de modo global |
| pen=pg.mkPen(255,0,0)  pen1=pg.mkPen(0,255,0)  pen2=pg.mkPen(0,0,255) | Para personalizar las graficas establecemos distintos colores de linea |
| class MainWindow(QtWidgets.QMainWindow,Ui\_MainWindow):  global Temperatura, Humedad, Presion, Altitud  def \_init\_(self,args,\*kwargs):  QtWidgets.QMainWindow.\_init\_(self,args,\*kwargs)  self.setupUi(self)  self.valores()  self.graphicsView.setBackground('w')  self.graphicsView\_2.setBackground('w')  self.graphicsView\_3.setBackground('w')  self.graphicsView\_4.setBackground('w')  self.graphicsView\_5.setBackground('w')  #self.actualizar() | Se establece la clase MainWindow y se configuran algunas vistas predeterminadas para la interfaz grafica |
| def valores(self):  global Temperatura, Humedad, Presion, Altitud  self.label\_5.setText("%0.1f" %Temperatura)  self.label\_6.setText("%0.1f" %Humedad)  self.label\_7.setText("%0.1f" %Presion)  self.label\_8.setText("%0.1f" %Altitud)  self.label\_9.setText("ºC")  self.label\_10.setText("%" )  self.label\_11.setText("MPA")  self.label\_12.setText("m")  self.lcdNumber.setProperty("value",Temperatura)  self.lcdNumber\_2.setProperty("value",Humedad)  self.lcdNumber\_3.setProperty("value",Presion)  self.lcdNumber\_4.setProperty("value",Altitud) | En esta función se determina como se va a mostrar la información en la interfaz gráfica, ya que esta no solo esta conformada por etiquetas, también por pequeños lcd’s |
| def actualizar(self):  global Temperatura, Humedad, Presion, Altitud,val,hour,temp, hum, pres ,Alt  sensor()  self.graphicsView.plot(hour,temp)  self.graphicsView\_2.plot(hour,hum, pen=pen)  self.graphicsView\_3.plot(hour,Alt,pen=pen1)  self.graphicsView\_4.plot(hour,pres,pen=pen2)  self.graphicsView\_5.plot(hour,temp)  self.graphicsView\_5.plot(hour,hum, pen=pen)  self.graphicsView\_5.plot(hour,Alt,pen=pen1)  self.graphicsView\_5.plot(hour,pres,pen=pen2)  valores=self.spinBox.value()  #print(valores)  if (val)>=valores:  val=0  hour=[]  temp=[]  hum=[]  pres=[]  Alt=[]  self.graphicsView.clear()  self.graphicsView\_2.clear()  self.graphicsView\_3.clear()  self.graphicsView\_4.clear()  self.graphicsView\_5.clear()  self.valores() | Esta función permite que se grafiquen los datos obtenidos por el sensor, de manera personalizada, para que mediante distintos colores se pueda diferenciar los datos graficados.  El número de muestras a graficar se determina mediante un spinBox, el cual establece el limite de datos, una vez superado, se limpian las variables y nuevamente se vuelve a comenzar a graficar |
| def sensor():  global Temperatura, Humedad, Presion, Altitud,temp, hum, pres ,Alt,hour,val  try:  Temperatura=bme280.temperature  Humedad=bme280.humidity  Presion=bme280.pressure  Altitud=bme280.altitude  temp.append(Temperatura)  hum.append(Humedad)  pres.append(Presion)  Alt.append(Altitud)  hour.append(val)  val+=1  Database()  except:  print ("Sensor Data not Available") | Esta función se encarga de recolectar la información del sensor, además de añadir series de datos a la lista que nos permitirá graficar el eje x, ya que en caso de perderse la comunicación el programa no encontrará un error por igualdad en el numero de datos por eje.  Una vez preparadas las listas, se ejecuta la función que agrega la información recabada por el sensor a una base de datos en Postgres. |
| def Database():  global Temperatura, Humedad, Presion, Altitud  try: connection=psycopg2.connect(user="pi", password="raspberry",host="127.0.0.1",port="5432",database="pi")  cursor=connection.cursor()  try:  createTableQuery='''Create TABLE IF NOT EXISTS EstatMeteo  (ID SERIAL PRIMARY KEY,  TEMPERATURA REAL NOT NULL,  HUMEDAD REAL NOT NULL,  PRESION REAL NOT NULL,  ALTITUD REAL NOT NULL);'''  cursor.execute(createTableQuery)  connection.commit()  print('Table created successfully in PostgresSQL')  except:  print('Table already created')  try:  addTableQuerry="INSERT INTO EstatMeteo (temperatura,humedad,presion,altitud) VALUES (%0.2f,%0.2f,%0.2f,%0.2f);" %(Temperatura, Humedad, Presion, Altitud)  print(addTableQuerry)  cursor.execute(addTableQuerry)  connection.commit()  print("Data added successfully")  except:  print("Data could not be added")  except(Exception,psycopg2.DatabaseError)as error:  print('Error while creating PostgresSQL table', error)  finally:  #closing database connection  if (connection):  cursor.close()  connection.close()  print("PostgresSQL connection is closed") | Esta función crea y añade los datos recabados por el sensor a la base de datos en Postgres, la cual esta previamente configurada y solo espera añadirle información, para la cual se establece una query para añadir los valores del sensor directamente al campo adecuado. |
| if \_name=="main\_":  app=QtWidgets.QApplication([])  window = MainWindow()  timer=QTimer()  timer.timeout.connect(window.actualizar)  timer.start(500)  window.show()  app.exec() | Esta es la condición fundamental que permite que nuestra interfaz grafica funcione plenamente, pues aquí se establece las variables que se tienen que cargar de las distintas clases importadas y se establece la actualización de la interfaz a cada 500ms |

|  |  |
| --- | --- |
|  |  |
| Interfaz funcionando en raspbery pi. | Datos recabados por el sensor añadidos a base de datos. |

Conclusiones

Para la realización de este proyecto se aplicaron y se reforzaron los conocimientos vistos en clase, al igual que se obtuvieron nuevos para la medición y manipulación del sensor BME280. Se logró cumplir por completo con los objetivos requeridos del proyecto, de igual manera se realizó de una forma agradable y fácil de comprender.

Referencias

¿Qué es Raspberry PI y para que sirve?. Consultado el 24 de septiembre de 2019. Disponible en: (https://www.abc.es/tecnologia/informatica-hardware/20130716/abci-raspberry-como-201307151936.html)

Introducción - Conociendo GitHub. (2012). Consultado el día 26 de noviembre de 2019. Disponible en: (<https://conociendogithub.readthedocs.io/en/latest/data/introduccion/>)

Base de datos – Concepto, Tipo y Elementos. Consultado el día 26 de noviembre de 2019. Disponible en: (<https://concepto.de/base-de-datos/#ixzz66PI2Hflx>)