

**Gabriel Maldonado Cervantes**

**Felipe Gómez Gonzales**

**José Emiliano Cortina Cruzado**

**Proyecto Final**

**Arquitecturas Programables Avanzadas**

**Profesor: Rafael Pérez Aguirre**

**Universidad Iberoamericana Puebla**

**28 de noviembre de 2019**

1. **INTRODUCCIÓN**

En el presente trabajo se implementó los conocimientos obtenidos en el curso para poder hacer un sistema embebido. El cual medirá presión, humedad y temperatura con ayuda del sensor BME280. Con ayuda de Python para la Raspberry se mostrarán los datos actuales del tiempo real y una sección de historial, con gráficas y tablas. Los datos también serán almacenados en una base de datos.

1. **OBJETIVOS**

* Desarrollar un sistema embebido que obtenga variables ambientales como presión,

temperatura y humedad. Usa el sensor BME280.

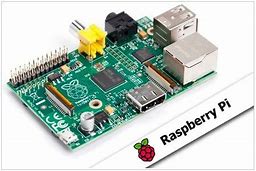
* Conéctalo a una aplicación escrita en Python para la Raspberry Pi, dicho programa será

una interfaz gráfica estilo dashboard, donde se mostrarán los datos actuales en tiempo

real y una sección de historial, como tablas o gráficas.

1. **Material**

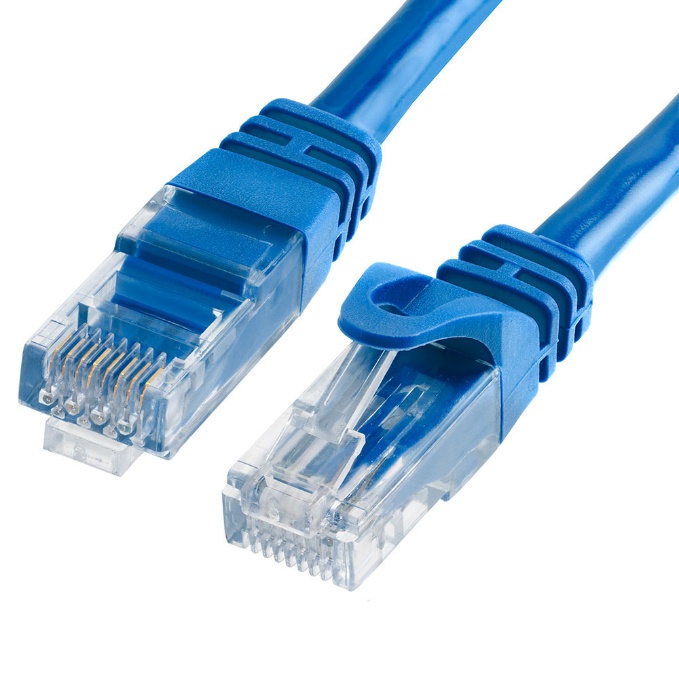
* Raspberry Pi
* Cable de Red
* Sensor BME280

1. **MARCO TEÓRICO**
2. **Raspberry pi**
   1. Raspberry Pi es un ordenador de placa reducida, ordenador de placa única u ordenador de placa simple (SBC) de bajo coste desarrollado en el Reino Unido por la Raspberry Pi Foundation, con el objetivo de estimular la enseñanza de informática en las escuelas.

El software es de código abierto, siendo su sistema operativo oficial una versión adaptada de Debian, denominada Raspbian, aunque permite usar otros sistemas operativos, incluido una versión de Windows 10. En todas sus versiones, incluye un procesador Broadcom, memoria RAM, GPU, puertos USB, HDMI, Ethernet (el primer modelo no lo tenía), 40 pines GPIO (desde la Raspberry Pi 2) y un conector para cámara. Ninguna de sus ediciones incluye memoria, siendo esta en su primera versión una tarjeta SD y en ediciones posteriores una tarjeta MicroSD.

La fundación da soporte para las descargas de las distribuciones para arquitectura ARM, Raspbian (derivada de Debian), RISC OS , Arch Linux ARM (derivado de Arch Linux) y Pidora (derivado de Fedora) y promueve principalmente el aprendizaje del lenguaje de programación Python. Otros lenguajes también soportados son Tiny BASIC, C, Perl y Ruby.

1. **Cable de Red**

Es un estándar de redes de área local para computadores, por sus siglas en español Acceso Múltiple con Escucha de Portadora y Detección de Colisiones (CSMA/CD). Su nombre procede del concepto físico de éter. Ethernet define las características de cableado y señalización; de nivel físico y los formatos de tramas de datos del nivel de enlace de datos del modelo OSI. Ethernet se tomó como base para la redacción del estándar internacional IEEE 802.3, siendo usualmente tomados como sinónimos. Se diferencian en uno de los campos de la trama de datos. Sin embargo, las tramas Ethernet e IEEE 802.3 pueden coexistir en la misma zona. Es el tipo de cable utilizado habitualmente para interconectar todos los dispositivos que conforman una LAN, incluyendo impresoras, discos externos, routers, escaners, switches y por supuesto las propias computadoras.

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 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.

Este tipo de sensores pueden ser utilizados para calcular la altitud con gran precisión (barómetro), 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.

**Especificaciones:**

* Voltaje de Operación: 1.8V - 3.3V DC
* Interfaz de comunicación: I2C o SPI (3.3V)
* Rango de Presión: 300 a 1100 hPa (0.3-1.1bar)
* Resolución: 0.16 Pa
* Precisión absoluta: 1 hPa
* Rango de Temperatura: -40°C a 85°C
* Resolución de temperatura: 0.01°C
* Precisión Temperatura: 1°C
* Rango de Humedad Relativa: 0-100% RH
* Precisión de HR: +-3%
* Rango de altura medible: 0-9100 metros
* Ultra-bajo consumo de energía
* Completamente calibrado
* Frecuencia de Muestreo: 157 Hz (máx.)

1. **DESARROLLO**
   1. **Metodología**
2. Se hizo un sistema embebido que a partir del Sensor **BME280** que permite la medición de temperatura, presión y humedad. Primero se desarrollo el programa a partir de lo visto en clase.

Se uso la biblioteca PyQt5 para la interfaz gráfica, para la comunicación I2C y la lectura del sensor si importaron dos bibliotecas smbus2 y bme280. Para la implementación de la base de datos se usó Postgres y para la realización de las gráficas se ocupó matplot.

Luego se desarrollo la interfaz gráfica con ayuda con el programa de QTDesigner. La interfaz fue la siguiente:

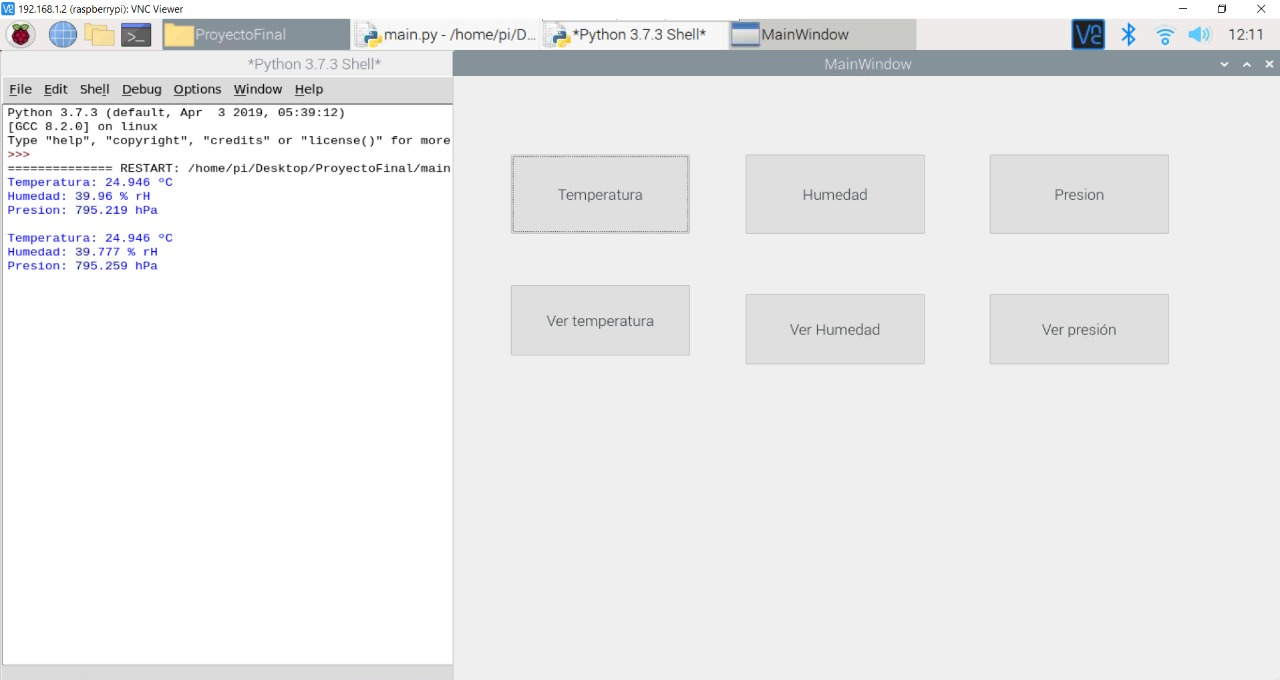


Figura 1. Interfaz Gráfica

El código del desarrollo de sistema embebido en base a una estación meteorológica, el código fue el siguiente:

|  |
| --- |
| # Importar bibliotecas para la interfaz gráfica  desde PyQt5 importar QtCore, QtGui, QtWidgets  # Importar bibliotecas para la conexión I2C y lectura del sensor  importar smbus2  importar bme280  # Biblioteca para interactuar con la base de datos Postgres  importar psycopg2  # Biblioteca para realizar gráficas  desde matplotlib importa pyplot como plt  # Abrir conexión con la BD  conn = psycopg2.connect ( ' dbname = proyectofinal ' )  # Abrir conexión I2C con sensor BME280 de temperatura, presión, y humedad  puerto = 1  direccion = 0x 76 # Dirección predeterminada del sensor  bus = smbus2.SMBus (puerto)  parametros\_calibracion = bme280.load\_calibration\_params (bus, direccion)  def sensarTemperatura ():  '' '  Función que obtiene la temperatura del sensor bme280 y la almacena  en la base de datos junto con el tiempo de medición  '' '  # Obtener todos los datos del sensor  datos = bme280.sample (bus, direccion, parametros\_calibracion)  # Obtener temperatura  temperatura = datos.temperature  # Redondear temperatura a 3 decimales  temperatura = redonda (temperatura, 3 )  # Imprimir resultado  imprimir ( " Temperatura: " + str (temperatura) + " ºC " )  # Agregar datos a la base de datos  # Abrir el cursor para escribir en la BD  cur = conn.cursor ()  # Obtener tiempo real  tiempo = int (time.time ())  # Crear comando SQL  comando = " insertar en valores de temperatura ( " + str (tiempo) + " , " + str (temperatura) + " ) "  # Ejecutar el comando  cur.execute (comando)  # Guardar los cambios  conn.commit ()  # Cerrar el cursor  cur.close ()  def sensarPresion ():  '' '  Función que obtiene la presión del sensor bme280 y la almacena  en la base de datos junto con el tiempo de medición  '' '  # Obtener todos los datos del sensor  datos = bme280.sample (bus, direccion, parametros\_calibracion)  # Obtener presionando  presion = datos.pressure  # Redondear presionando a 3 decimales  presionando = redondo (presionando, 3 )  # Imprimir resultado  print ( " Presion: " + str (presion) + " hPa " )  # Agregar datos a la base de datos  # Abrir el cursor para escribir en la BD  cur = conn.cursor ()  # Obtener tiempo real  tiempo = int (time.time ())  # Crear comando SQL  comando = " insertar en valores de presiones ( " + str (tiempo) + " , " + str (presion) + " ) "  # Ejecutar el comando  cur.execute (comando)  # Guardar los cambios  conn.commit ()  # Cerrar el cursor  cur.close ()  def sensarHumedad ():  '' '  Función que obtiene la presión del sensor bme280 y la almacena  en la base de datos junto con el tiempo de medición  '' '  # Obtener todos los datos del sensor  datos = bme280.sample (bus, direccion, parametros\_calibracion)  # Obtener humedad  humedad = datos.humedad  # Redondear humedad a 3 decimales  humedad = redonda (humedad, 3 )  # Imprimir resultado  print ( " Humedad: " + str (humedad) + " % r H " )  # Agregar datos a la base de datos  # Abrir el cursor para escribir en la BD  cur = conn.cursor ()  # Obtener tiempo real  tiempo = int (time.time ())  # Crear comando SQL  comando = " insertar en valores de humedades ( " + str (tiempo) + " , " + str (humedad) + " ) "  # Ejecutar el comando    cur.execute (comando)  # Guardar los cambios  conn.commit ()  # Cerrar el cursor  cur.close ()  def medirTodo ():  '' '  Función que mide todo (temperatura, presión, humedad) y lo  guarda en la BD  '' '  sensarTemperatura ()  sensarHumedad ()  sensarPresion ()  imprimir ( " " )    def mostrarTemperatura ():  '' '  Función que muestra el registro histórico de la temperatura usando matplotlib  '' '  print ( " Mostrando temperatura ... " )  # Abrir el cursor para obtener datos de la base de datos  cur = conn.cursor ()  # Ejecutar comando para obtener toda la información de la tabla de temperaturas  cur.execute ( ' select \* from temperaturas ' )  # Recuperar los resultados del SQL  datos = cur.fetchall ()  # Crear listas para tener el tiempo y temperatura en arreglos individuales  tiempo = []  temperatura = []  para dato en datos:  tiempo.append (dato [ 0 ])  temperatura.append (dato [ 1 ])  # Armar gráfica  plt.plot (tiempo, temperatura)  # Mostrar gráfica  plt.show ()  def mostrarPresion ():  '' '  Función que muestra el registro histórico de la presión usando matplotlib  '' '  print ( " Mostrando presion ... " )  # Abrir el cursor para obtener datos de la base de datos  cur = conn.cursor ()  # Ejecutar comando para obtener toda la información de la tabla de presiones  cur.execute ( ' select \* from presiones ' )  # Recuperar los resultados del SQL  datos = cur.fetchall ()  # Crear listas para tener el tiempo y presionar en arreglos individuales  tiempo = []  presionando = []  para dato en datos:  tiempo.append (dato [ 0 ])  presion.append (dato [ 1 ])  # Armar gráfica  plt.plot (tiempo, presionar)  # Mostrar gráfica  plt.show ()  def mostrarHumedad ():  '' '  Función que muestra el registro histórico de la humedad usando matplotlib  '' '  print ( " Mostrando humedad ... " )  # Abrir el cursor para obtener datos de la base de datos  cur = conn.cursor ()  # Ejecutar comando para obtener toda la información de la tabla de humedad  cur.execute ( ' select \* from humedades ' )  # Recuperar los resultados del SQL  datos = cur.fetchall ()  # Crear listas para tener el tiempo y la humedad en arreglos individuales  tiempo = []  humedad = []  para dato en datos:  tiempo.append (dato [ 0 ])  humedad.append (dato [ 1 ])  # Armar gráfica  plt.plot (tiempo, humedad)  # Mostrar gráfica  plt.show ()  # Definir interfaz gráfica QT  clase Ui\_MainWindow ( objeto ):  def setupUi ( self , MainWindow ):  MainWindow.setObjectName ( " MainWindow " )  MainWindow.resize ( 882 , 702 )  # Ventana principal  self .centralwidget = QtWidgets.QWidget (MainWindow)  self .centralwidget.setObjectName ( " centralwidget " )  # Botón de medir temperatura  self .botonTemperatura = QtWidgets.QPushButton ( self .centralwidget)  self .botonTemperatura.setGeometry (QtCore.QRect ( 60 , 90 , 191 , 91 ))  self .botonTemperatura.setObjectName ( " botonTemperatura " )  # Mapear ese botón a la función de medición  self .botonTemperatura.clicked.connect (sensarTemperatura)  # Botón de medir presionando  self .botonPresion = QtWidgets.QPushButton ( self .centralwidget)  self .botonPresion.setGeometry (QtCore.QRect ( 570 , 90 , 191 , 91 ))  self .botonPresion.setObjectName ( " botonPresion " )  # Mapear ese botón a la función de medición  self .botonPresion.clicked.connect (sensarPresion)  # Botón de medir humedad  self .botonHumedad = QtWidgets.QPushButton ( self .centralwidget)  self .botonHumedad.setGeometry (QtCore.QRect ( 310 , 90 , 191 , 91 ))  self .botonHumedad.setObjectName ( " botonHumedad " )  # Mapear ese botón a la función de medición  self .botonHumedad.clicked.connect (sensarHumedad)  # botón para ver gráfica de temperatura  self .botonVerTemperatura = QtWidgets.QPushButton ( self .centralwidget)  self .botonVerTemperatura.setGeometry (QtCore.QRect ( 60 , 240 , 191 , 81 ))  self .botonVerTemperatura.setObjectName ( " botonVerTemperatura " )  # Mapear el botón a la función de visualización  self .botonVerTemperatura.clicked.connect (mostrarTemperatura)  # botón para ver gráfica de humedad  self .botonVerHumedad = QtWidgets.QPushButton ( self .centralwidget)  self .botonVerHumedad.setGeometry (QtCore.QRect ( 310 , 250 , 191 , 81 ))  self .botonVerHumedad.setObjectName ( " botonVerHumedad " )  # Mapear el botón a la función de visualización  self .botonVerHumedad.clicked.connect (mostrarHumedad)  # botón para ver gráfica de presión  self .botonVerPresion = QtWidgets.QPushButton ( self .centralwidget)  self .botonVerPresion.setGeometry (QtCore.QRect ( 570 , 250 , 191 , 81 ))  self .botonVerPresion.setObjectName ( " botonVerPresion " )  # Mapear el botón a la función de visualización  self .botonVerPresion.clicked.connect (mostrarPresion)  # Crear timer para sensar cada 1 minuto  self .timer = QtCore.QTimer ( self .centralwidget)  # Darle un intervalo de 5 segundos  self .timer.setInterval ( 5000 )  # Conectar con función de sensado  self .timer.timeout.connect (medirTodo)  # Temporizador Empezar  self .timer.start ()  # Preparar la ventana (código generador por pyuic5)  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)  # Función que aplica textos a los botones  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\_ " :  importación sys  aplicación = QtWidgets.QApplication (sys.argv)  MainWindow = QtWidgets.QMainWindow ()  ui = Ui\_MainWindow ()  ui.setupUi (MainWindow)  MainWindow.show ()  sys.exit (app.exec\_ ()) |

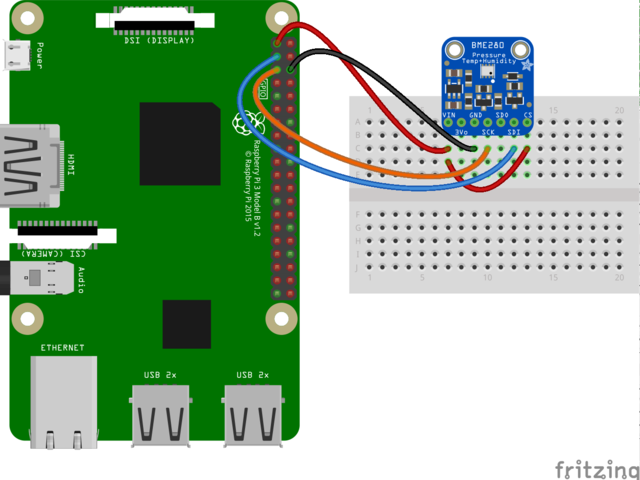
Se continuo con la conexión del sensor con la Raspberry pi.

Figura 2. Conexión del sensor BME280 a la Raspberry Pi

1. **Resultados**

A partir de la interfaz gráfica mostrada anteriormente, al presionar el botón de temperatura, presión o humedad te muestra el valor en tiempo real en el Shell de Python, como se muestra a continuación:

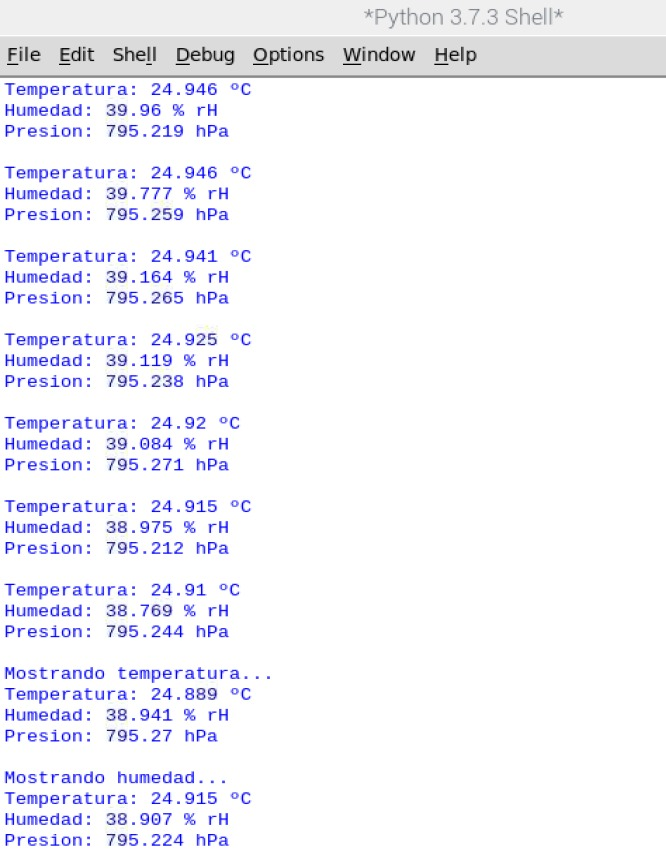
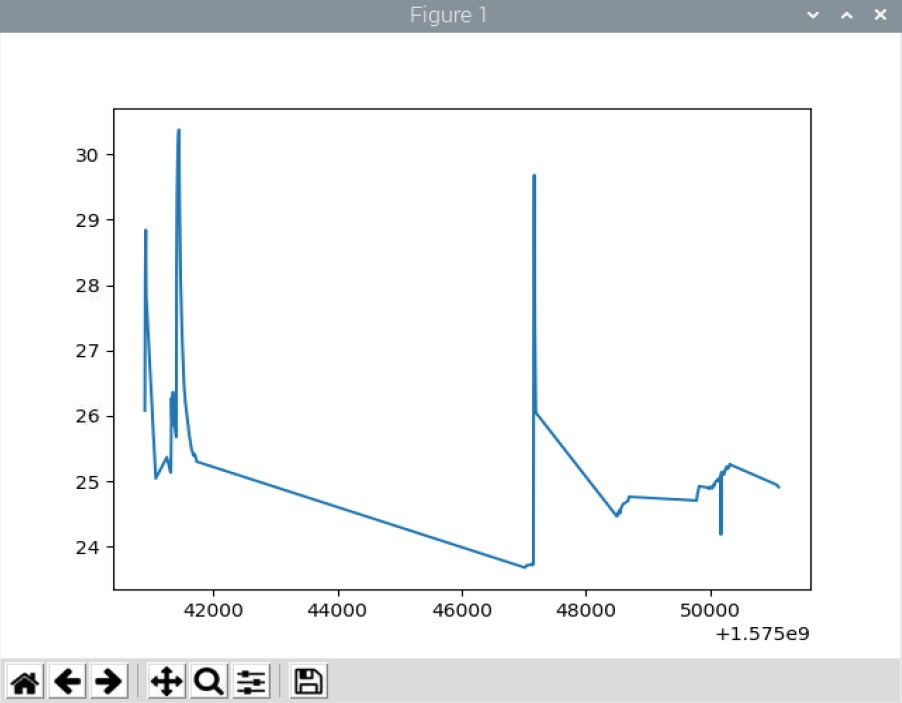


Figura 3. Valores del sensor en tiempo real

Al presionar el botón de “Ver humedad” o el deseado, se te mostraba la grafica de cada uno de valores del sensor. Por cada sensor se muestra una gráfica.



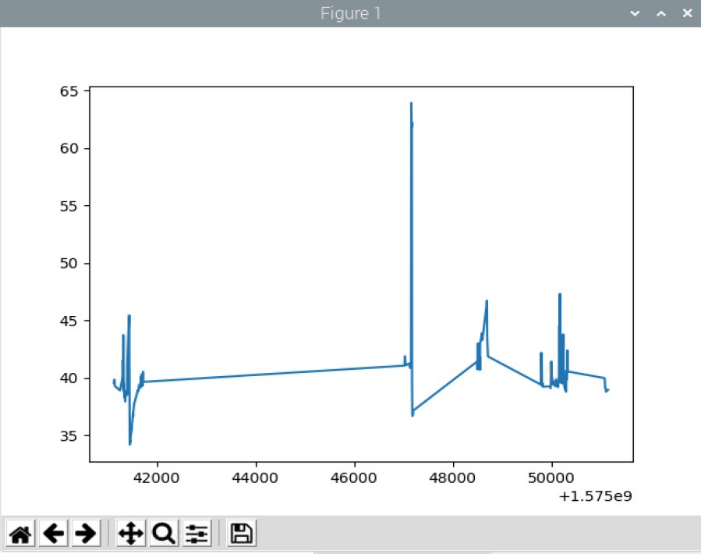
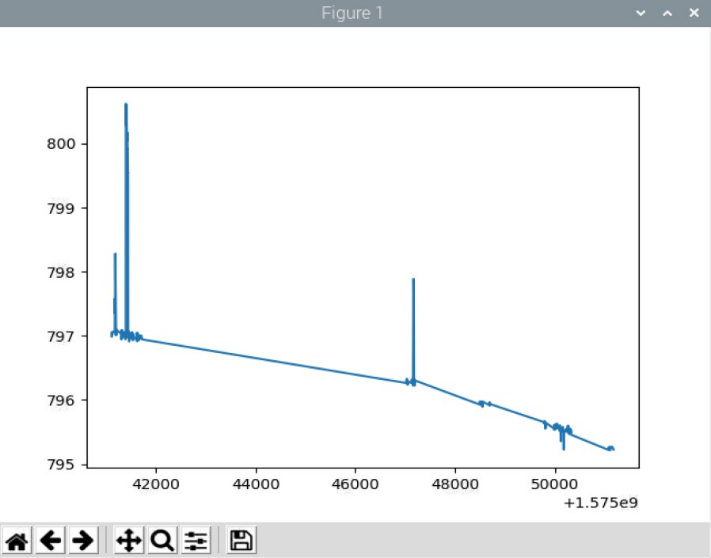
Figura 4. Gráfica de Temperatura

Figura 5. Gráfica de Humedad

Figura 6. Gráfica de Presión

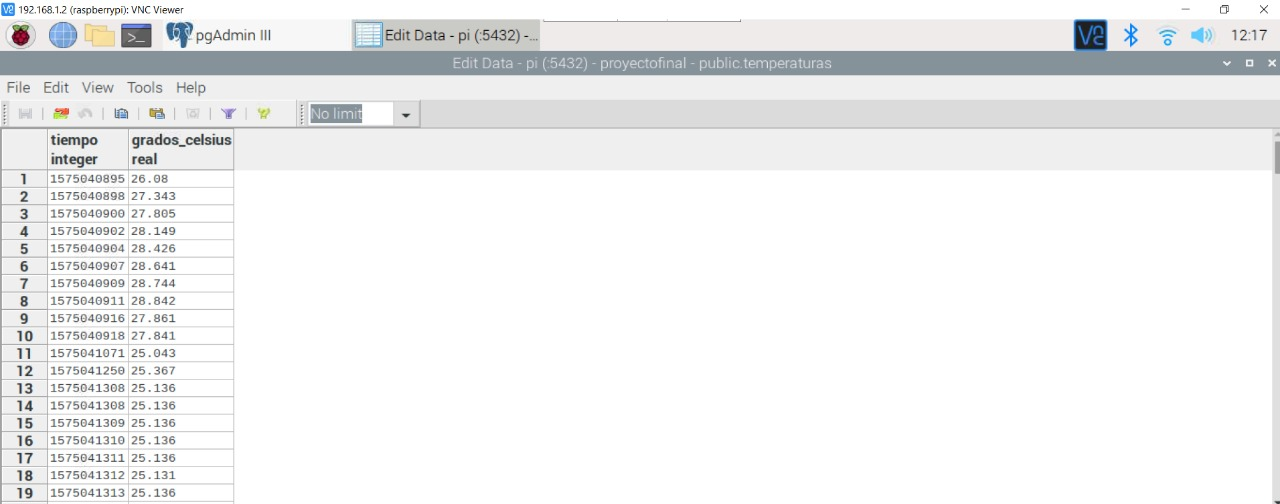
La obtención de datos es continua por lo cual, todo dato almacenado se iba a nuestra base de datos hecha a partir de Postgress. Los datos guardados son los siguientes:

Figura 7. Datos de temperatura

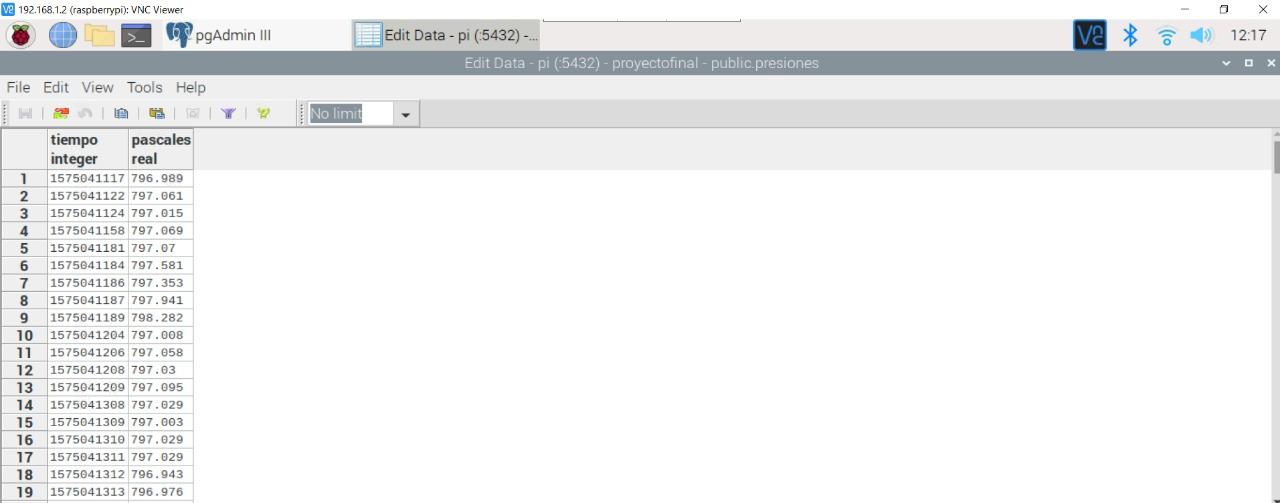


Figura 8. Datos de la Presión

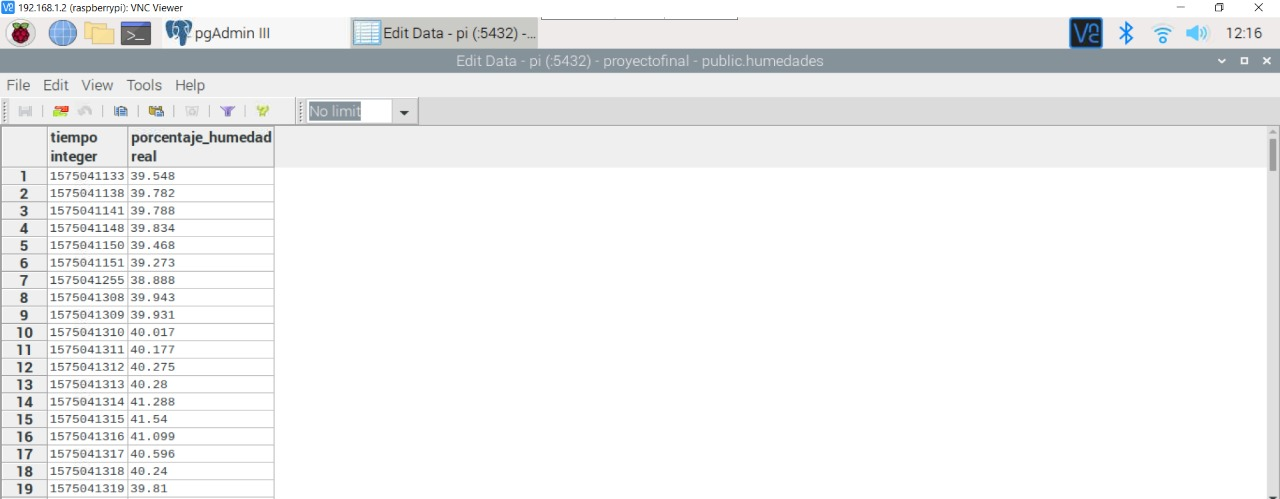


Figura 9. Datos de la Humedad

**Repositorio GitHub**

<https://github.com/Emiliano0911/ProyectoFinal>

1. **Conclusiones**

A través de la realización del siguiente proyecto se llegó a poder poner en práctica los conocimientos obtenidos alrededor se semestre y con ello poder hacer con cualquier sensor, la lectura de datos a partir de un interfaz y con ello poder también almacenados en un base de datos para poder obtenerla en cualquier momento deseado. También se pudo ver la gran diferencia de poder trabajarlo a través de la Rasberry ya que es mucho más cómodo e intuitivo poder trabajar con ella y de la misma manera poder implementarla en muchos sistemas industriales sin tener la necesidad de comprar un ordenador que podría salir más caro y el espacio ocupado es mucho mayor.

1. **REFERENCIAS**

**Electronics, Naylamp.** Naylamp Electronics. *Naylamp Electronics.* [En línea] [Citado el: 2019 de Noviembre de 2019.] https://naylampmechatronics.com/sensores-posicion-inerciales-gps/357-sensor-de-presiontemperatura-humedad-bme280.html.

**N.A. 2019.** Comunicación serie. [En línea] 16 de 08 de 2015. [Citado el: 29 de 11 de 2019.] https://es.wikipedia.org/wiki/Comunicaci%C3%B3n\_serie.

**—. 2019.** Raspberry Pi. [En línea] 20 de 09 de 2014. [Citado el: 29 de 11 de 2019.] https://es.wikipedia.org/wiki/Raspberry\_Pi.

**N:A:. 2019.** Arduino. [En línea] 20 de 09 de 2015. [Citado el: 29 de 11 de 2019.] https://es.wikipedia.org/wiki/Arduino.