

**María Fernanda Mendoza Romero**

**José Francisco Morales**

**Francisco Gonzáles Morales**

**Proyecto Final**

**“Sensor BME280 con Python en Raspberry Pi”**

Arquitecturas Programables Avanzadas

Profesor: Rafael Pérez Aguirre

Universidad Iberoamericana Puebla

**27 de noviembre del 2019**

|  |  |
| --- | --- |
| **Proyecto final #1 “Sensor de BME280 con Python en Rasperry Pi”** | Resultado de imagen para dht11 python |

1. **INTRODUCCIÓN**

**Python es un lenguaje de programación interpretado.**

1. **OBJETIVOS**
   1. **General**

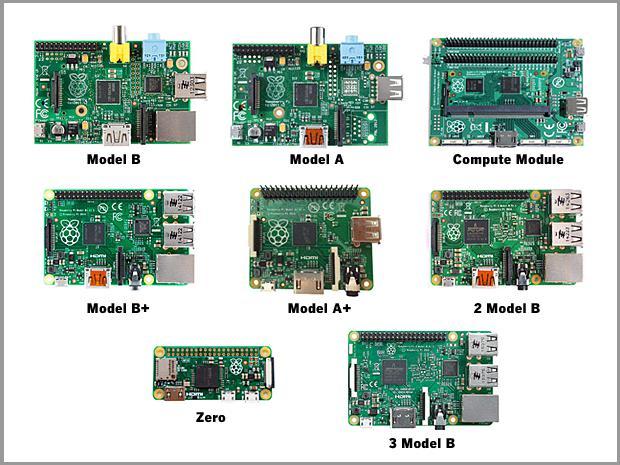
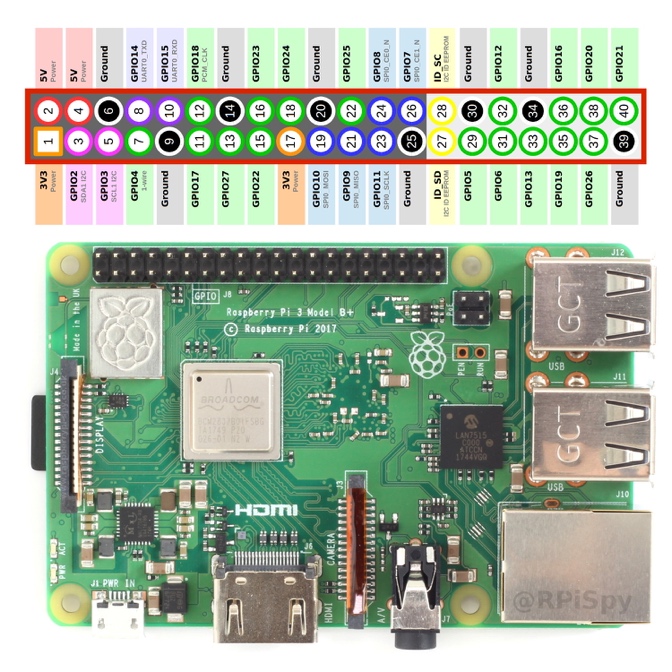
* Desarrollar un sistema embebido que obtenga variables ambientales como pesión, temperatura y humedad.
* Usar el sensor BME280.
  1. **Específicos**
* Conectar el sistema a una aplicación escrita en Python para la Raspberry Pi, dicho programa será una interfaz gráfica estilo dashboard
* Mostrar los datos actuales en tiempo real y una sección de historial, como tablas o gráficas.
* Almacentar los datos del sensor en una base de datos.
* Crear un reporte debidamente documentado, con portada, introducción, desarrollo y conclusiones.
* Subir el proyecto y reporte al repositorio de su proyecto en github, ademas del moodle.

1. **MARCO TEÓRICO**

* **Raspberry Pi**

Una Raspberry Pi es una mini computadora que funciona a 5VD/2.5A, es también un sistema digital de procesamiento y que funciona gracias a un sistema operativo. Dentro de sus principales características de este dispositivo una de ellas es el tamaño reducido y su gran capacidad de procesamiento. Es una de las tarjetas más usadas por desarrolladores de sistemas embebidos. Además tiene la capacidad de conexión con un monitor vía puerto HDMI como también, puede tener distintas conexiones por ejemplo puerto USB, Wifi, pines entrada y salida, puertos para cámara y vídeo(AV) , en esta imagen podemos apreciar mejor sus componentes y puertos:

Como antes mencionaba existen varios modelos en la siguiente imagen se pueden apreciar mejor las diferencias:



1. **Python**

Es un lenguaje de programación de alto nivel fácil de aprender, hoy en día es el más usado. Cuenta con estucturas de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programaci´no orientada a objetos. Funciona como interprete y consta de una amplia biblioteca de funciones. Es considerado como un lenguaje de extensiones para aplicaciones personalizables. Interpretado lo cual significa que no es necesario compliar el código para ejecutarlo y multipropósito, en los últimos años se ha utilizado constantemente y ha ido aumentando usabilidad en el desarrollo de software.Python puede ser utilizado en diversas pataformas y sistemas operatios: Windows, Mac OS y Linux.



1. **Sensor BME280**

Es un sensor que integra en un solo dipositivo el sensor de presión atmosférica, temperatura y humedad relativa con gran precisión, bajo consumo energético y formato ultra compacto. Esta basado en la tecnología BOSCH piezo -resistiva Es un método para facilirar la interacción del usuario con el ordenador o la computadora a tráves de la utilización de un conjunto de imágenes y objetos(íconos, ventanas, botones, etc…) ademas de texto. Surge como evolución de la línea de comandos de los primeros sistemas operativos y es la pieza fundamental en un entorno gráfico.

**ESPECIFICACIONES TÉCNICAS**

* 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. **Interfaz Gráfica**

Es un método para facilirar la interacción del usuario con el ordenador o la computadora a tráves de la utilización de un conjunto de imágenes y objetos(íconos, ventanas, botones, etc…) ademas de texto. Surge como evolución de la línea de comandos de los primeros sistemas operativos y es la pieza fundamental en un entorno gráfico.

1. **QT**

El marco multiplataforma Qt, por ejemplo, reduce la cantidad de trabajo en el desarrollo de interfaces de usuario y en la creación de programas completos, donde además resulta de especial importancia facilitar la colaboración entre programadores y desarrolladores. Todo ello queda garantizado por los componentes como el entorno de desarrollo integrado Qt Creator, un lenguaje declarativo propio y diversos módulos libremente seleccionables.

Debido a los problemas que les planteaban las herramientas para el desarrollo de software multiplataforma, Noweger Haavard Nord y Eirik Chambe-Eng comenzaron en1990a desarrollar Qt. Dos años más tarde, crearon la empresa Trolltech, adquirida posteriormente por la multinacional de comunicaciones Nokia y, en 2012, por la compañía de software finlandesa Digia. Así, desde 2014, la filial The Qt Company es la responsable del desarrollo de herramientas y bibliotecas.

A lo largo de los años, Qt ha pasado de ser una sencilla biblioteca de clases a un amplio framework, dejando así tras de sí numerosas actualizaciones en su modelo de licencia. El actual modelo de licencia dual, que brinda al usuario la oportunidad de elegir entre una licencia propietaria o licencias de código abierto GPL y LGPL, hace posible tanto el uso libre como la explotación comercial.



Qt Designer: Es un programa (parte del conjuntos de programas para el desarrollo de aplicaciones para el Framework Qt) para desarrollar interfaces gráficas de usuario (multilenguages debido a que genera un archivo XML cuyo contenido es el formato de dicho GUI, pudiéndolo convertir con los programas pertinentes a cada lenguage).



1. **Aplicación PostgressSQL**

Es un sistema que permite la gestión de base de datos relacionada a objetos. Esta aplicación es de código abierto y permite la creación de servidores para bases de datos.

Se utilizó esta aplicación para base de datos debido a que presenta un entorno mucho más simple, lo que implica menos tiempo para aprender a utilizar la aplicación.

1. **Base de datos**

Una base de datos consiste en un conjunto de datos que se encuentran relacionados entre si y que por medio de esta se pueden hacer consultas, actualizaciones, modificaciones e incluso eliminaciones de datos que se encuentren en dicha base. Se utilizan como una herramienta para la recopilación y administración de información. En una base de datos es posible almacenar datos de todo tipo tanto información de personas como de cosas.

Una base de datos consiste en un conjunto de tablas que se guardan en un solo archivo además de otro tipo de funciones, cada tabla consiste en la división de filas y columnas donde se almacenan los datos, este tipo de almacenamiento de datos hace que sea más flexible y evita que existan redundancias

1. **Github**

GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git.

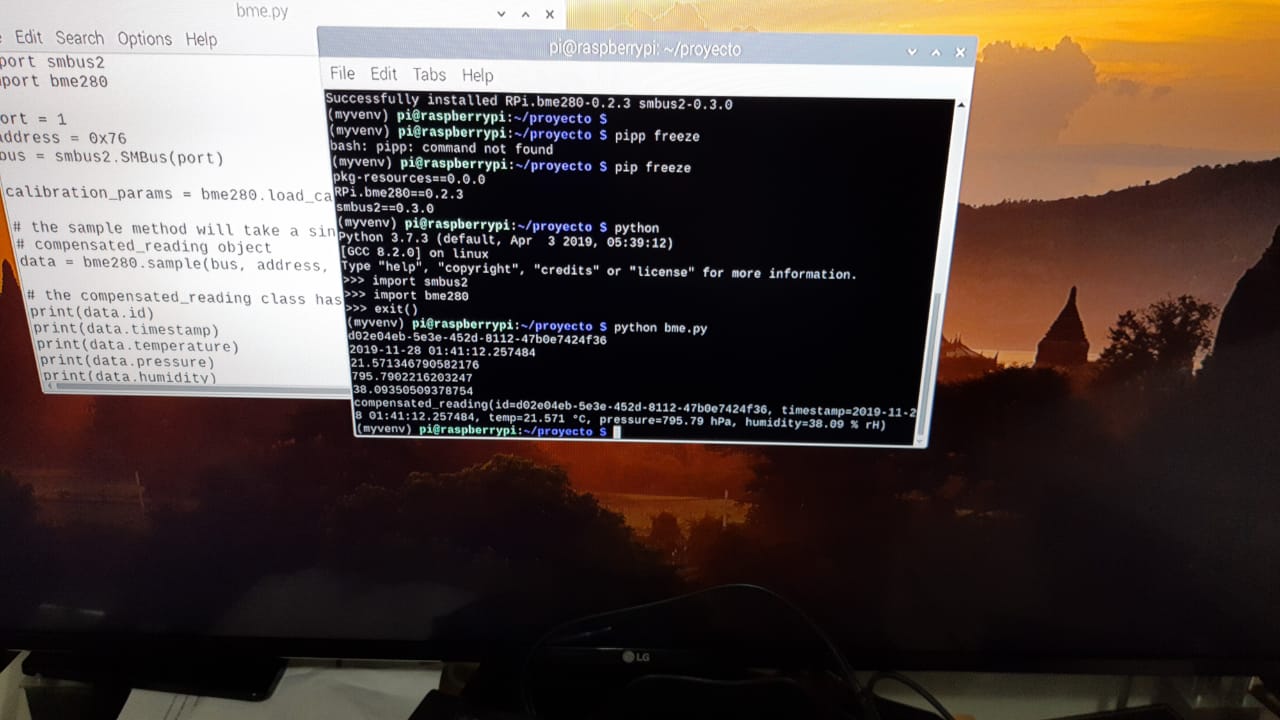
El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago.GitHub aloja tu repositorio de código y te brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto.Además de eso, puedes contribuir a mejorar el software de los demás. Para poder alcanzar esta meta, GitHub provee de funcionalidades para hacer un fork y solicitar pulls. Un fork es simplemente clonar un repositorio ajeno (genera una copia en tu cuenta), para eliminar algún bug o modificar cosas de él. Una vez realizadas tus modificaciones puedes enviar un pull al dueño del proyecto. Éste podrá analizar los cambios que has realizado fácilmente, y si considera interesante tu contribución, adjuntarlo con el repositorio original.



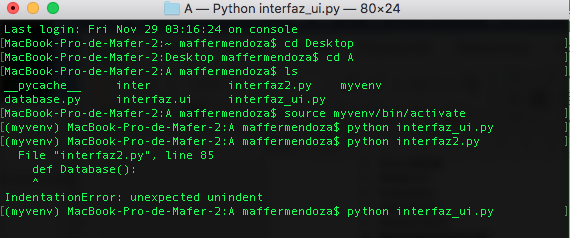
1. **DESARROLLO**
   1. **Materiales**

* Sensor BME280
* Raspberry Pi
* Computadora
* Cable Ethernet
* Programa Python
* Base de datos Postgres SQL
* QT Designer para la interfaz gráfica
  1. **Metodología**

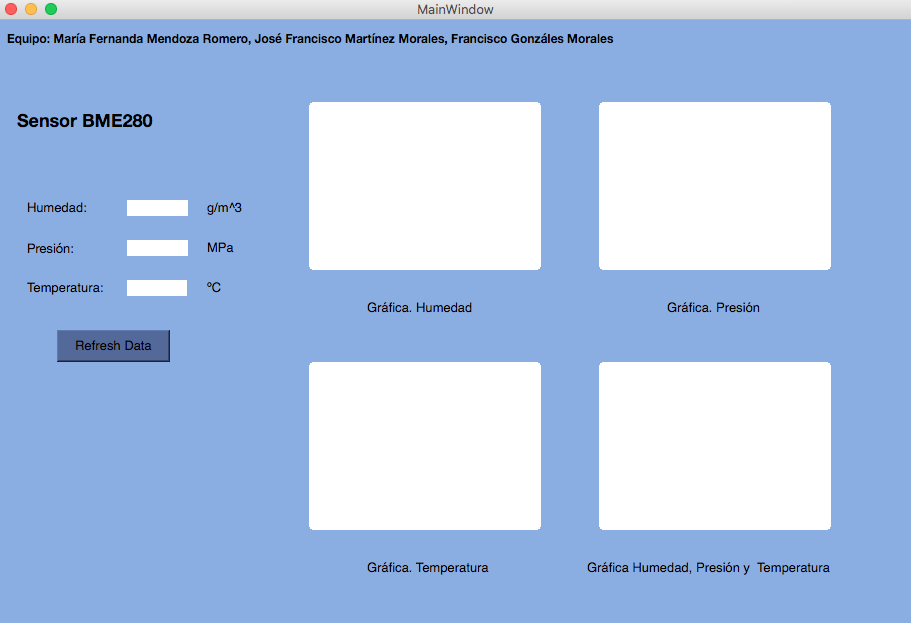
Para la elaboración de este proyecto el primer paso fue conseguir el sensor en Mercado Libre, esperamos a que llegara. Una vez teniendo el sensor desarrollamos pruebas para conocer y comprender su funcionamiento.

****

En esta imagen podemos observar como se programó el sensor en la raspberry con el fin de comprobar los valores que arrojaba el sensor.

****

Al mismo tiempo se realizaron pruebas con la interfaz gráfica, la parte bonita del sensor se creó en Qt Designer esta herramienta fue aprendida en clase.Se creó un entorno virtual dentro de la carpeta donde incluirá los archivos adjuntos para poder indicarle a la interfaz los movimientos o los datos que almacenara y mostrará en pantalla.



Esta es nuestra interfaz gráfica, donde el sensor mandará los datos y los mostrará en forma de gráficas en los rectángulos grandes y en los rectángulos pequeños mostrará el número flotante de la variables de la humedad, presión y temperatura.

Este fue el código de la interfaz, tuvimos que hacer unos ajustes debido a que no compilaba la biblioteca QtChart que es para las gráficas. Se cambió la clase de este archivo .ui a .py por medio de las diapositivas vistas en clase.

# -\*- coding: utf-8 -\*-

# Form implementation generated from reading ui file 'interfaz.ui'

#

# Created by: PyQt5 UI code generator 5.13.2

#

# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets,QtChart

#from QtCharts import QChartView

class Ui\_MainWindow(object):

def setupUi(self, MainWindow):

MainWindow.setObjectName("MainWindow")

MainWindow.resize(918, 653)

self.centralwidget = QtWidgets.QWidget(MainWindow)

self.centralwidget.setStyleSheet("background-color: rgb(138, 174, 225)")

self.centralwidget.setObjectName("centralwidget")

self.label = QtWidgets.QLabel(self.centralwidget)

self.label.setGeometry(QtCore.QRect(20, 60, 231, 81))

font = QtGui.QFont()

font.setFamily("Helvetica Neue")

font.setPointSize(18)

font.setBold(True)

font.setWeight(75)

self.label.setFont(font)

self.label.setObjectName("label")

self.label\_2 = QtWidgets.QLabel(self.centralwidget)

self.label\_2.setGeometry(QtCore.QRect(30, 180, 60, 16))

font = QtGui.QFont()

font.setFamily("Helvetica")

self.label\_2.setFont(font)

self.label\_2.setObjectName("label\_2")

self.label\_3 = QtWidgets.QLabel(self.centralwidget)

self.label\_3.setGeometry(QtCore.QRect(30, 220, 60, 16))

font = QtGui.QFont()

font.setFamily("Helvetica Neue")

self.label\_3.setFont(font)

self.label\_3.setObjectName("label\_3")

self.label\_4 = QtWidgets.QLabel(self.centralwidget)

self.label\_4.setGeometry(QtCore.QRect(30, 260, 101, 16))

font = QtGui.QFont()

font.setFamily("Helvetica")

self.label\_4.setFont(font)

self.label\_4.setObjectName("label\_4")

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

self.pushButton.setGeometry(QtCore.QRect(60, 310, 113, 32))

font = QtGui.QFont()

font.setFamily("Helvetica")

self.pushButton.setFont(font)

self.pushButton.setStyleSheet("background-color: rgb(83, 105, 154);")

self.pushButton.setObjectName("pushButton")

self.graphicsView = QtChart.QChartView(self.centralwidget)

self.graphicsView.setGeometry(QtCore.QRect(300, 70, 256, 192))

self.graphicsView.setObjectName("graphicsView")

self.label\_5 = QtWidgets.QLabel(self.centralwidget)

self.label\_5.setGeometry(QtCore.QRect(130, 180, 61, 16))

self.label\_5.setStyleSheet("background-color: rgb(255, 255, 255);")

self.label\_5.setText("")

self.label\_5.setObjectName("label\_5")

self.label\_6 = QtWidgets.QLabel(self.centralwidget)

self.label\_6.setGeometry(QtCore.QRect(130, 220, 61, 16))

self.label\_6.setStyleSheet("background-color: rgb(255, 255, 255);")

self.label\_6.setText("")

self.label\_6.setObjectName("label\_6")

self.label\_7 = QtWidgets.QLabel(self.centralwidget)

self.label\_7.setGeometry(QtCore.QRect(130, 260, 60, 16))

self.label\_7.setStyleSheet("background-color: rgb(254, 254, 254);")

self.label\_7.setText("")

self.label\_7.setObjectName("label\_7")

self.graphicsView\_2 = QtChart.QChartView(self.centralwidget)

self.graphicsView\_2.setGeometry(QtCore.QRect(590, 70, 256, 192))

self.graphicsView\_2.setObjectName("graphicsView\_2")

self.graphicsView\_3 = QtChart.QChartView(self.centralwidget)

self.graphicsView\_3.setGeometry(QtCore.QRect(300, 330, 256, 192))

self.graphicsView\_3.setObjectName("graphicsView\_3")

self.graphicsView\_4 = QtChart.QChartView(self.centralwidget)

self.graphicsView\_4.setGeometry(QtCore.QRect(590, 330, 256, 192))

self.graphicsView\_4.setObjectName("graphicsView\_4")

self.label\_8 = QtWidgets.QLabel(self.centralwidget)

self.label\_8.setGeometry(QtCore.QRect(210, 180, 60, 16))

font = QtGui.QFont()

font.setFamily("Helvetica")

self.label\_8.setFont(font)

self.label\_8.setObjectName("label\_8")

self.label\_9 = QtWidgets.QLabel(self.centralwidget)

self.label\_9.setGeometry(QtCore.QRect(210, 220, 60, 16))

font = QtGui.QFont()

font.setFamily("Helvetica")

self.label\_9.setFont(font)

self.label\_9.setObjectName("label\_9")

self.label\_10 = QtWidgets.QLabel(self.centralwidget)

self.label\_10.setGeometry(QtCore.QRect(210, 260, 60, 16))

font = QtGui.QFont()

font.setFamily("Helvetica")

self.label\_10.setFont(font)

self.label\_10.setObjectName("label\_10")

self.label\_11 = QtWidgets.QLabel(self.centralwidget)

self.label\_11.setGeometry(QtCore.QRect(10, 10, 791, 16))

font = QtGui.QFont()

font.setFamily("Helvetica Neue")

font.setPointSize(12)

font.setBold(True)

font.setWeight(75)

self.label\_11.setFont(font)

self.label\_11.setObjectName("label\_11")

self.label\_12 = QtWidgets.QLabel(self.centralwidget)

self.label\_12.setGeometry(QtCore.QRect(370, 280, 121, 16))

font = QtGui.QFont()

font.setFamily("Helvetica")

self.label\_12.setFont(font)

self.label\_12.setObjectName("label\_12")

self.label\_13 = QtWidgets.QLabel(self.centralwidget)

self.label\_13.setGeometry(QtCore.QRect(670, 280, 121, 16))

font = QtGui.QFont()

font.setFamily("Helvetica")

self.label\_13.setFont(font)

self.label\_13.setObjectName("label\_13")

self.label\_14 = QtWidgets.QLabel(self.centralwidget)

self.label\_14.setGeometry(QtCore.QRect(370, 540, 141, 16))

font = QtGui.QFont()

font.setFamily("Helvetica")

self.label\_14.setFont(font)

self.label\_14.setObjectName("label\_14")

self.label\_15 = QtWidgets.QLabel(self.centralwidget)

self.label\_15.setGeometry(QtCore.QRect(590, 540, 261, 16))

font = QtGui.QFont()

font.setFamily("Helvetica")

self.label\_15.setFont(font)

self.label\_15.setObjectName("label\_15")

MainWindow.setCentralWidget(self.centralwidget)

self.menubar = QtWidgets.QMenuBar(MainWindow)

self.menubar.setGeometry(QtCore.QRect(0, 0, 918, 22))

self.menubar.setObjectName("menubar")

MainWindow.setMenuBar(self.menubar)

self.statusbar = QtWidgets.QStatusBar(MainWindow)

self.statusbar.setObjectName("statusbar")

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.label.setText(\_translate("MainWindow", "Sensor BME280"))

self.label\_2.setText(\_translate("MainWindow", "Humedad:"))

self.label\_3.setText(\_translate("MainWindow", "Presión:"))

self.label\_4.setText(\_translate("MainWindow", "Temperatura:"))

self.pushButton.setText(\_translate("MainWindow", "Refresh Data"))

self.label\_8.setText(\_translate("MainWindow", "g/m^3"))

self.label\_9.setText(\_translate("MainWindow", "MPa"))

self.label\_10.setText(\_translate("MainWindow", "ºC"))

self.label\_11.setText(\_translate("MainWindow", "Equipo: María Fernanda Mendoza Romero, José Francisco Martínez Morales, Francisco Gonzáles Morales "))

self.label\_12.setText(\_translate("MainWindow", "Gráfica. Humedad"))

self.label\_13.setText(\_translate("MainWindow", "Gráfica. Presión"))

self.label\_14.setText(\_translate("MainWindow", "Gráfica. Temperatura"))

self.label\_15.setText(\_translate("MainWindow", "Gráfica Humedad, Presión y Temperatura"))

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

import sys

app = QtWidgets.QApplication(sys.argv)

MainWindow = QtWidgets.QMainWindow()

ui = Ui\_MainWindow()

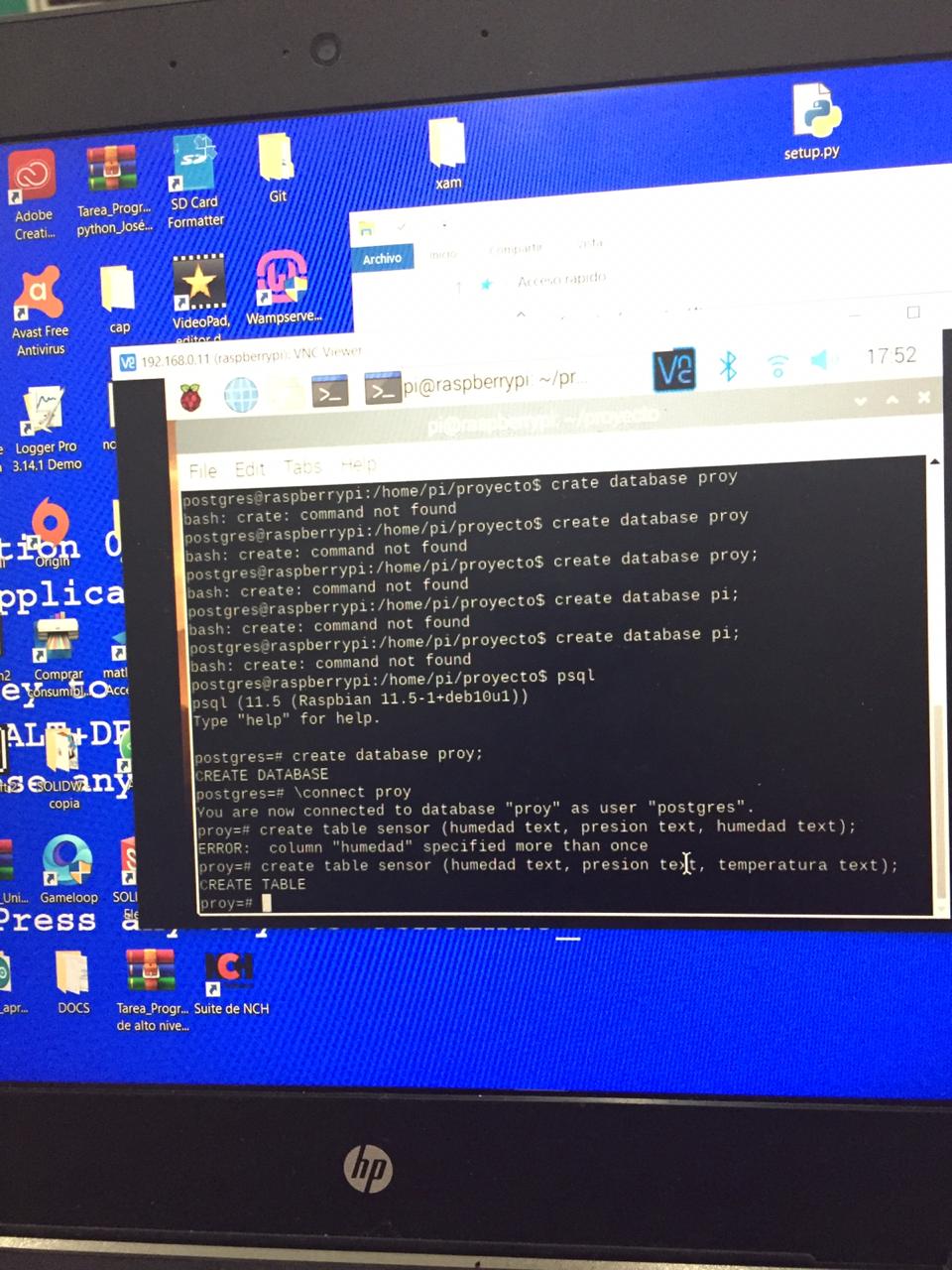
ui.setupUi(MainWindow)

MainWindow.show()

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

Este fue el código de la interfaz, tuvimos que hacer unos ajustes debido a que no compilaba la biblioteca QtChart que es para las gráficas. Se cambió la clase de este archivo .ui a .py por medio de las diapositivas vistas en clase.

Para la Tabla de datos, se descargó PostgreSQL y ahí incluida PgAdmin por lo mismo fue tema visto en clase, ya que era una manera muy sencilla de comprender el funcionamiento de una tabla de datos por medio de un servidor, ya que el proyecto es un sistema embebido con el fin de ser un proyecto en tiempo real. Se crearon otras dos tablas para ver como funcionaba esta aplicación en la Raspberry Pi.



En este código, se guió por la diapositiva de clase, con los parámetros de seguridad. Descargamos las bibliotecas psycopg2,  determinamos nuestro usuario y nuestra contraseña, el host y la database , se creó la tabla con los datos(Humedad,Presión y Temperatura) que necesitamos conocer para que sean mostrados en la interfaz gráfica.

import psycopg2

from psycopg2 import Error

            connection=psycopg2.connect(user="proy", password="proy",host="127.0.0.1",port="5432",database="proy")

            cursor=connection.cursor()

            try:

                createTableQuery='''Create TABLE IF NOT EXISTS SENSORBME280

                    (ID SERIAL  PRIMARY KEY,

                    HUMEDAD    REAL NOT NULL,

                    PRESION        REAL NOT NULL,

                    TEMPERATURA       REAL NOT NULL);'''

                cursor.execute(createTableQuery)

                connection.commit()

                print('Table created successfully in PostgresSQL')

except:

                (Exception,psycop2.DatabaseError) as error:

                print("Error while creating PostgreSQL table", error)print('Table already created')

            try:

                addTableQuerry="INSERT INTO SENSORBME280 (humedad,presion,temperatura) VALUES (%0.2f,%0.2f,%0.2f);" %(Humedad,Presion,Temperatura)

                print(addTableQuerry)

                cursor.execute(addTableQuerry)

                connection.commit()

            finally:

                if (connection):

                    cursor.close()

                    connection.close()

                    print("PostgreSQL connection is closed")

Una vez, que obtuvimos todos los datos por separado decidimos juntarlo y fue ahí donde tuvimos problemas técnicos. Nuestra Raspberry Pi presentó problemas al instalarle varios programas, por lo que nos vimos obligados a volverle a instalar el sistema operativo. Esto sucedió una vez más no abría la terminal y ninguna aplicación.

* 1. **Mediciones y resultados**

Como resultado quisimos presentar el siguiente código, tratando de probarlo en la Raspberry Pi , pero por cuestiones de tiempo, de conocimientos y problemas técnicos tuvimos bastantes fallas.

from interfaz\_ui import\*

from PyQt5 import QtCore, QtGui, QtWidgets,QtChart

import pyqtgraph as pg

#import RPi.GPIO as GPIO

#from time import sleep

#import board

#import busio

#import adafruit\_bme280

import psycopg2

#from psycopg2 import Error

#try:

#i2c = busio.I2C(board.SCL, board.SDA)

#BME280 = adafruit\_bme280.Adafruit\_BME280\_I2C(i2c)

#except:

#print("No se encuentra ningún sensor")

#GPIO.setmode(GPIO.BCM)

#GPIO.setup(6,GPIO.OUT)

Humedad=0

Presion=0

Temperatura=0

valorinicial=0

humedad=[]

presion=[]

temperatura=[]

tiempo=[]

pen=pg.mkPen(255,0,0)

pen1=pg.mkPen(0,255,0)

class MainWindow(QtWidgets.QMainWindow,Ui\_MainWindow):

global Temperatura, Humedad, Presion

def \_\_init\_\_(self,args,\*kwargs):

QtWidgets.QMainWindow.\_init\_(self,args,\*kwargs)

self.setupUi(self)

self.datos()

self.graphicsView.setBackground('w')

self.graphicsView\_2.setBackground('w')

self.graphicsView\_3.setBackground('w')

self.graphicsView\_4.setBackground('w')

self.graphicsView\_5.setBackground('w')

def actualizo(self):

print ("Actualizando")

self.label.setText("Actualizando Datos del Sensor")

self.pushButton.setText("lo actualizaste")

def valores(self):

global Temperatura, Humedad, Presion

self.label\_5.setText("value", Humedad)

self.label\_6.setText("value", Presion)

self.label\_7.setText("value", Temperatura)

def actualizar(self):

global Humedad,Presion,Temperatura,humedad,presion,temperatura

sensor()

self.graphicsView.QtChart.QChartView(tiempo,humedad)

self.graphicsView\_2.QtChart.QChartView(tiempo,presion, pen=pen)

self.graphicsView\_3.QtChart.QChartView(tiempo,temperatura,pen=pen1)

self.graphicsView\_4.QtChart.QChartView(tiempo,humedad)

self.graphicsView\_4.QtChart.QChartView(tiempo,presion, pen=pen)

self.graphicsView\_4.QtChart.QChartView(tiempo,temperatura,pen=pen1)

def sensor():

global Humedad,Presion,Temperatura,humedad,presion,temperatura,tiempo,valorinicial

try:

Humedad=bme280.humidity

Presion=bme280.pressure

Temperatura=bme280.temperature

humedad.append(Humedad)

presion.append(Presion)

temperatura.append(Temperatura)

tiempo.append(valorinicial)

valorinicial+=1

def Database():

global Humedad,Presion,Temperatura

try:

connection=psycopg2.connect(user="proy", password="proy",host="127.0.0.1",port="5432",database="proy")

            cursor=connection.cursor()

            try:

                createTableQuery='''Create TABLE IF NOT EXISTS SENSORBME280

                    (ID SERIAL  PRIMARY KEY,

                    HUMEDAD    REAL NOT NULL,

                    PRESION        REAL NOT NULL,

                    TEMPERATURA       REAL NOT NULL);'''

                cursor.execute(createTableQuery)

                connection.commit()

                print('Table created successfully in PostgresSQL')

except:

                (Exception,psycop2.DatabaseError) as error:

                print("Error while creating PostgreSQL table", error)print('Table already created')

            try:

                addTableQuerry="INSERT INTO SENSORBME280 (humedad,presion,temperatura) VALUES (%0.2f,%0.2f,%0.2f);" %(Humedad,Presion,Temperatura)

                print(addTableQuerry)

                cursor.execute(addTableQuerry)

                connection.commit()

            finally:

                if (connection):

                    cursor.close()

                    connection.close()

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

app=QtWidgets.QApplication([])

window = MainWindow()

window.show()

app.exec()

El proyecto esta en el repositorio <https://github.com/Maffer16/Proyecto-final-arquitecturas-programables-avanzadas.git>

1. **CONCLUSIONES**

Este proyecto nos ha servido para mostrar los conocimientos aprendidos durante el curso, lamentablemente hemos tenido bastantes fallas técnicas así como un poco de falta de concimientos sobre programación. Fue un reto para nosotros sacar adelante este proyecto, aunque no se probará o compilara en la Raspberry Pi , se lograron los objetivos separados , únicamente faltó enseñar el sistema embebido funcionando sin ningún error.se subió a Moodle y al repositorio GitHub. Se trató de realizar este proyecto con la iniciativa y las ganas sobre los conocimientos adquiridos en clase. Sin embargo, aún nos quedan varios temas que pulir para poder ser buenos compiladores en Python ,Postgress PgAdmin. Disfrutamos aprender de los errores pero en el futuro seremos capaces de manejar estos programas y poder realizar proyectos en base a este.

1. **REFERENCIAS**

* <http://issuu.com/rduinostar/docs/dht11_datasheet>
* <https://learn.adafruit.com/dht>
* <http://www.oddwires.com/using-a-dht11-to-measure-temperature-and-humidity/>
* <http://www.engineersgarage.com/electronic-components/16x2-lcd-module-datasheet>