CUADERNO DE PRÁCTICAS

JORGE ALMINGOL ESTRADA

NIA: 684039

REDES DE SENSORES

MASTER EN INGENIERÍA ELECTRÓNICA

CURSO 2020-2021

Índice

[1. Practica 1 3](#_Toc62402379)

[1.1. Configuración y control de módulos hardware 3](#_Toc62402380)

[1.2. Lectura del valor análogico cada 10s mediante el uso de la interrupción de timer 3](#_Toc62402381)

[1.3. Genera una salida pwm a 5kHz proporcional a la lectura del ADC. 5](#_Toc62402382)

[1.4. Comanda por la UART los periféricos 5](#_Toc62402383)

[1.5. Conecta un sensor inercial por I2C (o SPI), muestrea la aceleración cada 100 ms y manda los datos cada segundo vía UART 7](#_Toc62402384)

[2. Practica 2: Implementar el modelo anterior mediante tareas con el sistema operativo FreeRTOS. 10](#_Toc62402385)

[3. Practica 3. Comunicación, gestión y representación de datos de sensores con Python. 12](#_Toc62402386)

[3.1. Programa en Python un programa que acceda al puerto serie y muestre por pantalla los datos en tiempo real 12](#_Toc62402387)

[3.2. Modifica el programa para que almacene los datos en fichero .txt 12](#_Toc62402388)

[3.3. Modifica el programa para que acumule los datos durante 5 segundos, calcule el promedio y desviación estándar los represente en tiempo real. 14](#_Toc62402389)

[4. Comunicaciones WIFI y stack IP 17](#_Toc62402390)

[4.1. Conéctate a la red wifi del laboratorio o a una creada por el móvil como punto de acceso. Extrae tu IP. 17](#_Toc62402391)

[4.2. Pon en hora el módulo mediante un servidor NTP (Network Time Protocol) 19](#_Toc62402392)

[4.3. Monta un chat una aplicación software PC 20](#_Toc62402393)

[4.4. Después sustituye uno de los extremos por el módulo hardware siendo cliente y envía cada segundo tu hora local. 20](#_Toc62402394)

[4.5. Añadir una capa de control de tal modo que cuando se le mande “start” empiece a mandar la hora hasta que se le mande “stop”. 20](#_Toc62402395)

[4.6. Basándote en el estándar SENML crea un fichero JSON 22](#_Toc62402396)

[4.7. Sube datos a la nube, en concreto al servicio gratuito proporcionado por Adafruit. 24](#_Toc62402397)

[5. Conexiones BLE y Bluethoot 27](#_Toc62402398)

[5.1. Extrae temperatura, humedad y presión de los mensajes recibidos utilizando Python del sensor “RuuviTagSensor” 27](#_Toc62402399)

[5.2. Haz un advertising con tu módulo siguiendo la identificación iBeacon 27](#_Toc62402400)

[5.3. Establece un “chat” Bluetooth con perfil SPP con una APP en el móvil 28](#_Toc62402401)

[6. Comunicaciones Lora y LoraWAN 30](#_Toc62402402)

[6.1. Establece comunicaciones ping-pong con otros compañeros basada en broadcasts. 30](#_Toc62402403)

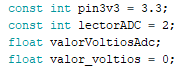
[6.2. Envía cada 30 segundos un número incremental de 2 bytes con protocolo Lorawan a un servidor en la nube. 31](#_Toc62402404)

# Practica 1

## Configuración y control de módulos hardware

Lectura analógica de una tensión entre 0 y 3.3V

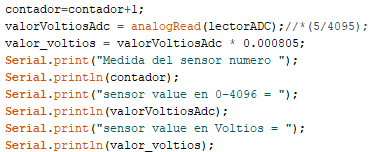
Introducimos las constantes necesarias



El setup se inicia la configuración del puerto serie

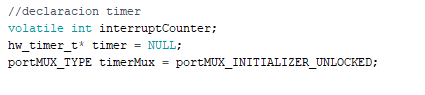
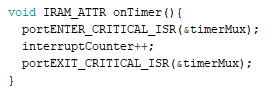


Por último se lee el valor del sensor del ADC y se adapta a voltios, mostrándolo por pantalla ya dentro del loop

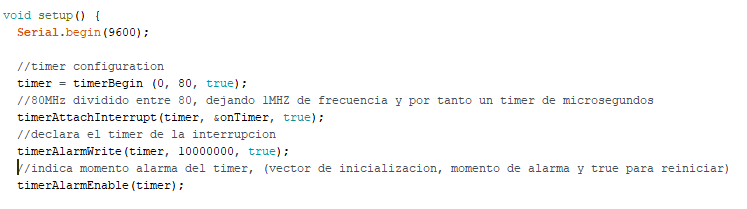


## Lectura del valor analógico cada 10s mediante el uso de la interrupción de timer

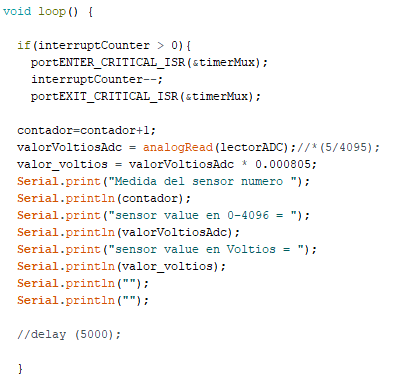
En primer lugar declaramos el timer y su función correspondiente. Esta función pone un flag a 1 cada vez que se ejecuta

Dentro del setup declaramos las características del timer

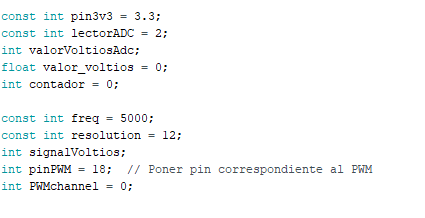


Y por último en el loop, cuando detecte que la interrupción ha activado el flag, lo pondrá a 0 y realizara la lectura del sensor



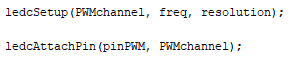
## Genera una salida pwm a 5kHz proporcional a la lectura del ADC.

Declaramos las variables necesarias para el funcionamiento del programa

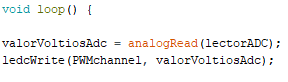


Después ajustamos dentro del setup la frecuencia del PWM a los 5khz requeridos y la resolución, puesto que la del adc es de 12, le pondremos 12.

Una vez declarado le asignamos un pin de salida. PWMchannel sería el pwm configurado y el pin asignado seria pinPWM.



Ya dentro del loop, asignamos el valor de entrada del PWM el de salida del ADC. Y para terminar se mostrará por pantalla como hemos hecho en programas anteriores

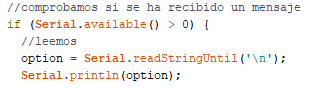


## Comanda por la UART los periféricos

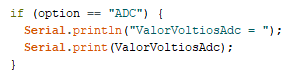
* 1. ADC: Envíe la lectura del ADC actual
  2. ADC(x): envíe la lectura del ADC cada x segundos. Si x=0, deja de mandar datos
  3. PWM(x): comanda el duty cycle del módulo PWM con números del 0 al 9.

La adquisición de los datos y el control de la salida del PWM se realizan igual que en el programa anterior. En este se desarrolla el control por UART.

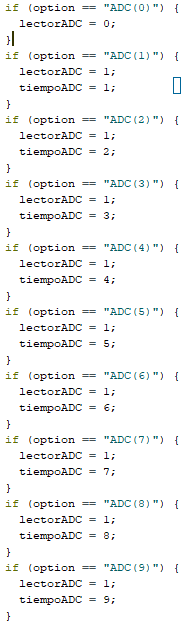
Ya dentro del loop, en primer lugar comprobamos si hemos recibido un mensaje



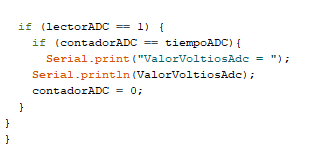
Si el mensaje es ADC envía el valor del ADC



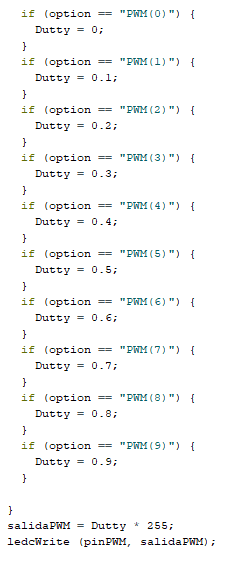
Si es ADC(x) envía el valor cada x segundos



Contamos los ciclos correspondientes y enviamos el mensaje (Si ADC(0) dejamos de enviar)

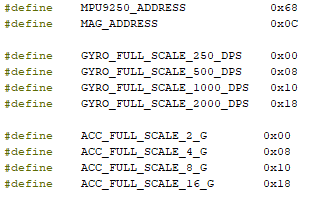


Para el PWM hacemos algo similar

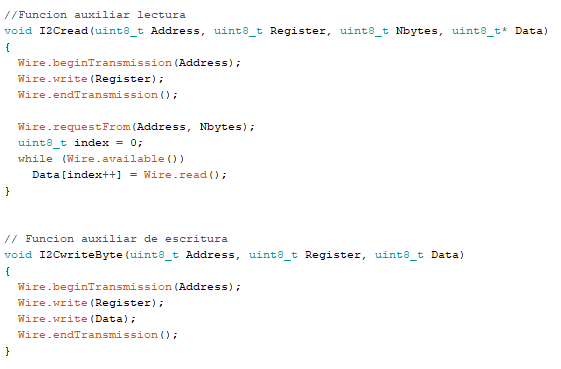


## Conecta un sensor inercial por I2C (o SPI), muestrea la aceleración cada 100 ms y manda los datos cada segundo vía UART

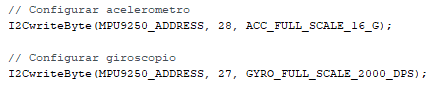
Definimos los datos de configuración



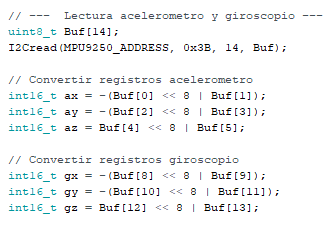
Las funciones de lectura y escritura de I2C



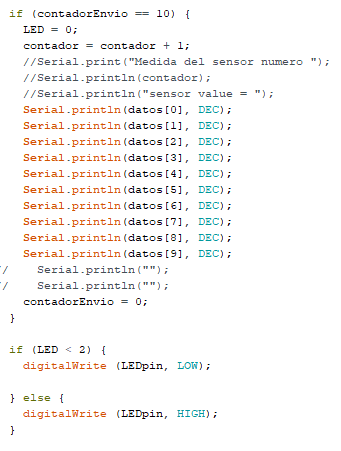
Dentro del setup configuramos el sensor de aceleración y de giro y sus rangos de trabajo



Y ya dentro del loop, controlado por la interrupción de timer leemos los datos



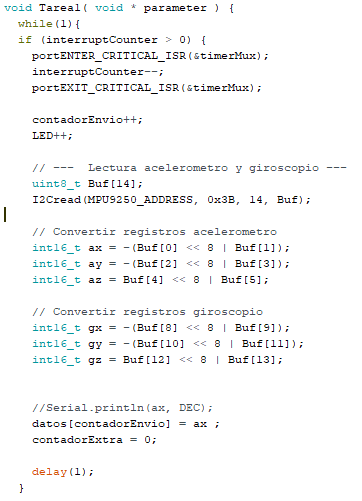
Para capturar en tiempo tenemos la interrupción temporal configurada en 100ms. Captura los datos y los guarda en un vector de tamaño 10, una vez lleno lo muestra. Los dos primeros ciclos de llenado encienden un led.



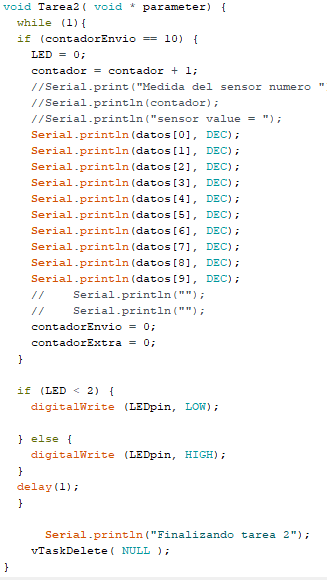
# Practica 2: Implementar el modelo anterior mediante tareas con el sistema operativo FreeRTOS.

Dentro del Setup creamos las tareas, en este caso se asigna a ambas la misma prioridad, pero en caso contrario la que tenga el número más alto será la más prioritaria, (la prioridad es el 5 elemento de la declaración, en ambas es 1)



Ya luego introducimos el código que queremos que realice cada tarea. La tarea 1 se encargara de recoger datos del sensor y la tarea 2 los mostrara por pantalla y encenderá el led 

Esta tarea usa la interrupción aunque realmente no es del todo necesario. En FreeRTOS lo interesante es utilizar los delay y confiar en la gestión temporal del propio sistema operativo. La tarea dos encargada del envío de los datos:



# Practica 3. Comunicación, gestión y representación de datos de sensores con Python.

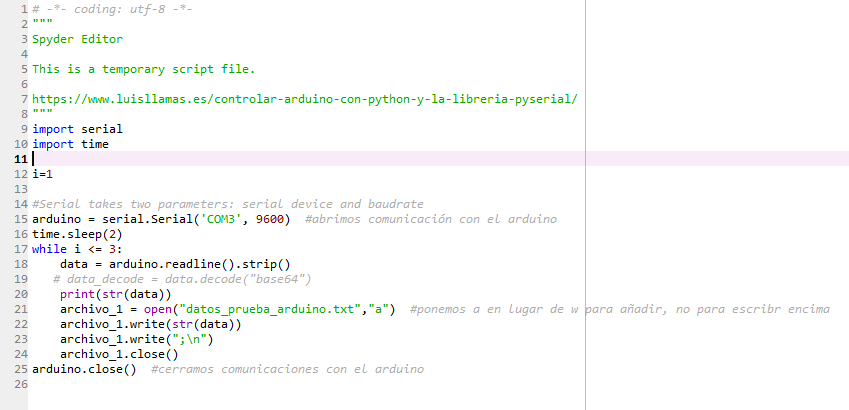
## Programa en Python un programa que acceda al puerto serie y muestre por pantalla los datos en tiempo real

## Modifica el programa para que almacene los datos en fichero .txt

Se van a realizar ambas a la vez

Como indica el enunciado instalamos pyserial y procedemos a desarrollar el código. Para ello hacemos del entorno anaconda y el programa spyder 3, que trabaja con python 3.5.

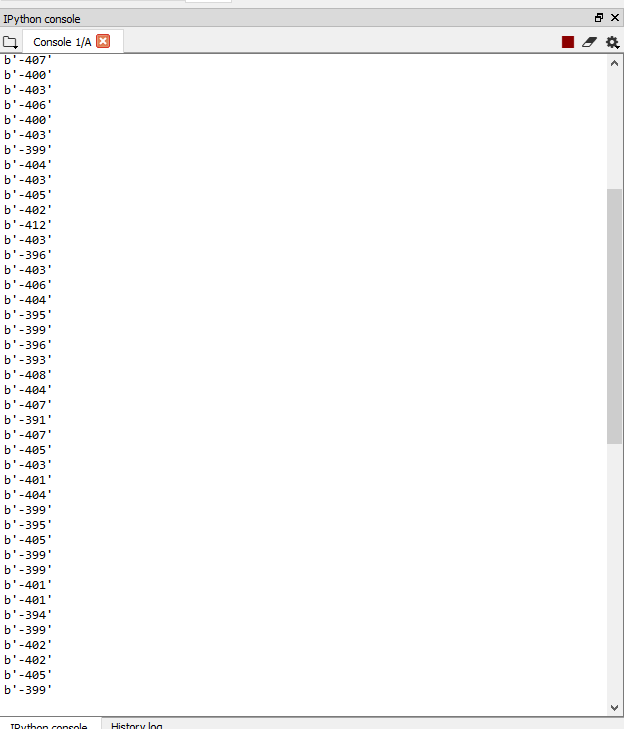
El código es el siguiente:



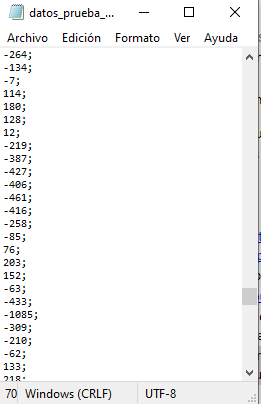
Creamos una comunicación serie llamada arduino con las características de comunicación del programa de arduino.

Una vez ya en el bucle leemos los datos del puerto serie, abrimos un documento de texto los guardamos incluyendo las terminaciones de punto y coma y retorno de carro para leerlas desde Excel.

Además la consola muestra los datos por pantalla.



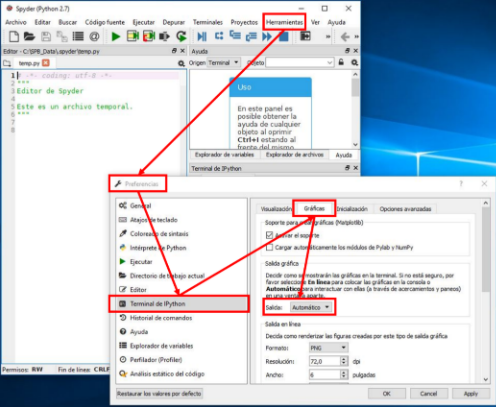
El txt. queda también bien realizado con los datos correctamente guardados.



Solo se realizó con un dato, pero en caso de usar más la solución sería añadir solo punto y coma y cuando se reciban los 6 datos incluir el final de línea

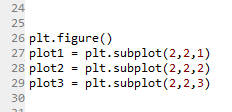
## Modifica el programa para que acumule los datos durante 5 segundos, calcule el promedio y desviación estándar los represente en tiempo real.

Realizamos la configuración recomendada en el enunciado

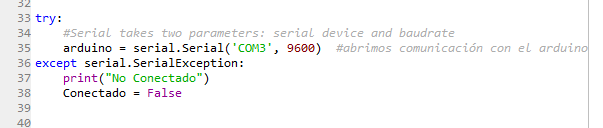


Y modificamos el código anterior para que genere una gráfica.

Declaramos las gráficas



Iniciamos comunicación con el micro

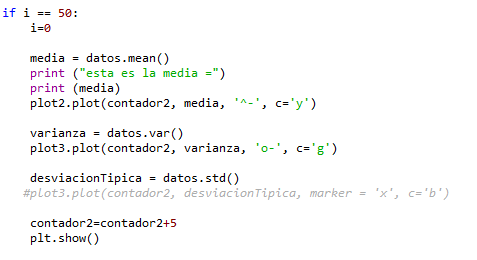


Y posteriormente, comunicamos con él, leemos el dato, ponemos un modo de saltarnos un posible error de decodificación y por ultimo modificamos los datos para que cumplan los requerimientos necesarios.

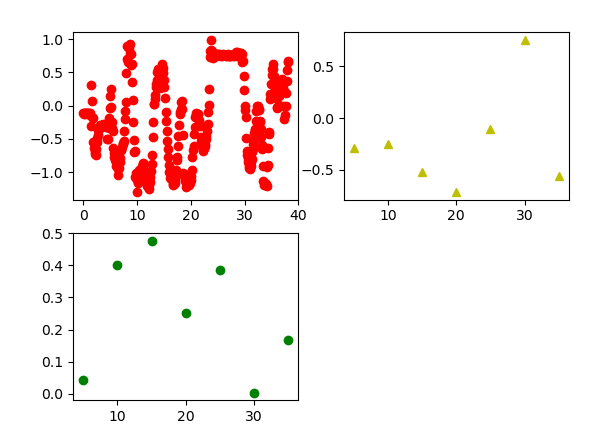


Una vez hecho esto guarda los datos en un array de tamaño 50 y, individualmente los guarda en un fichero de texto y los muestra en ploot1.

Después calcula la media y la varianza de los datos recogidos en el array y los muestra en otras 2 gráficas



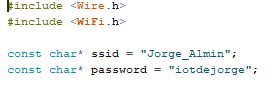
LA grafica queda de la siguiente manera mostrando las aceleraciones del eje x medidas cada 100ms y la media y varianza tomadas cada 50 datos.



# Comunicaciones WIFI y stack IP

## Conéctate a la red WIFI del laboratorio o a una creada por el móvil como punto de acceso. Extrae tu IP.

Incluimos las librerías correspondientes y añadimos la SSID y la PASS de la red WiFi a la que nos queremos conectar



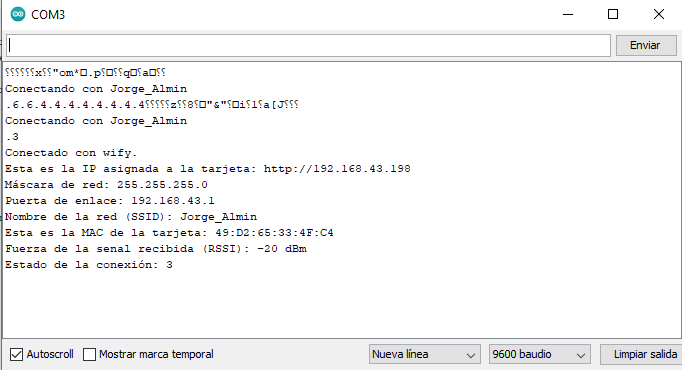
Dentro del Setup indicamos los datos de la red a la que nos queremos comentar y esperamos a que se conecte



Posteriormente mostramos por pantalla los datos requeridos mediante las siguientes peticiones entre las que se incluye mostrar la conexión IP, la máscara de red, la puerta de enlace, los datos de la red, la mac de la tarjeta e incluso os datos de la potencia de la señal.

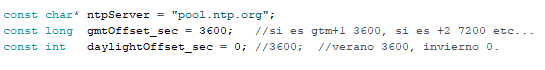


El resultado:

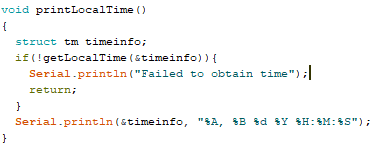


## Pon en hora el módulo mediante un servidor NTP (Network Time Protocol)

Para la conexión a la red WIFI usamos lo visto en el modelo anterior. En este caso añadimos la conexión a un servidor que proporciona la hora y lo configuramos a la hora de nuestro país teniendo en cuenta el cambio horario



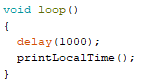
También añadimos una función encargada de mostrar correctamente la hora:



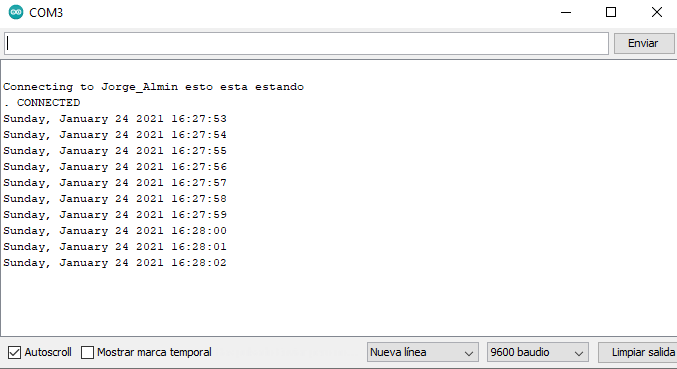
Dentro del setup, además de iniciar la comunicación con la red WIFI iniciamos la comunicación con el servidor ntp.



Y ya dentro del loop demandamos la hora cada segundo

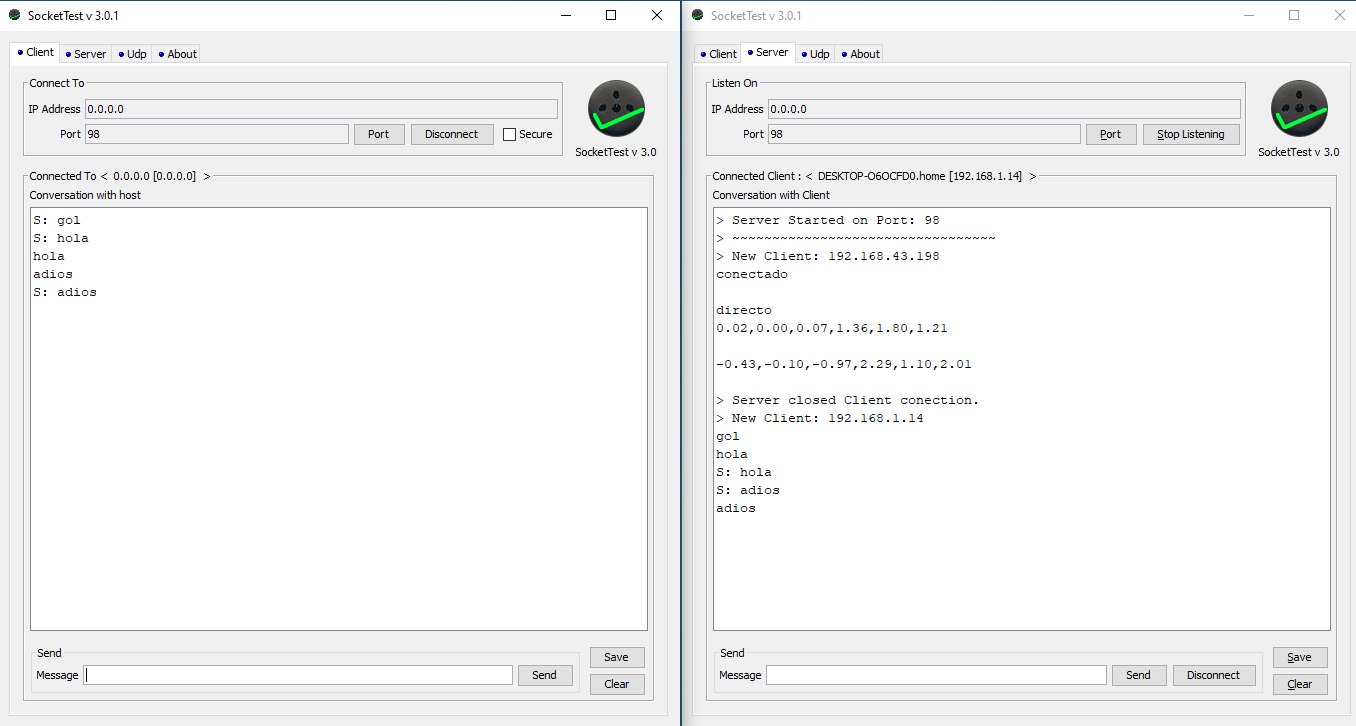


Con el siguiente resultado:



## Monta un chat una aplicación software PC

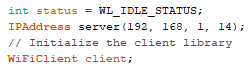
Mediante la aplicación SocketTest creamos un chat, para ello ejecutamos 2 ventanas, una será cliente y el otro servidor. Para ello, puesto que están dentro del pc solo es necesario configurar los dos con la misma dirección IP y en el mismo puerto.



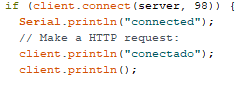
## Después sustituye uno de los extremos por el módulo hardware siendo cliente y envía cada segundo tu hora local.

## Añadir una capa de control de tal modo que cuando se le mande “start” empiece a mandar la hora hasta que se le mande “stop”.

Realizamos estas dos funciones en un mismo código. La configuración de la puesta en hora y la conexión WIFI la omitimos debido a que ya se ha explicado anteriormente. En primer lugar para conectarnos a un servidor de SocketTest introducimos la ip estática v4 del ordenador en el programa y creamos un cliente.

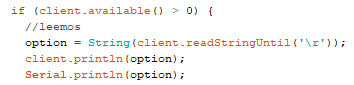


Una vez declarados estos datos nos conectamos con el servidor añadiendo la ip del servidor y el puerto de conexión dentro del setup:

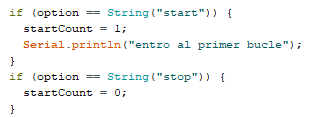


De esta forma ya podríamos enviar los datos al servidor pero además vamos a añadir el código completo con las peticiones de start stop.

Ya en el loop mediante una interrupción gestionamos el tiempo. Además en cada iteración evaluamos si hay algún mensaje pendiente que provenga del servidor. Se realiza igual que con las comunicaciones de puerto Serial.

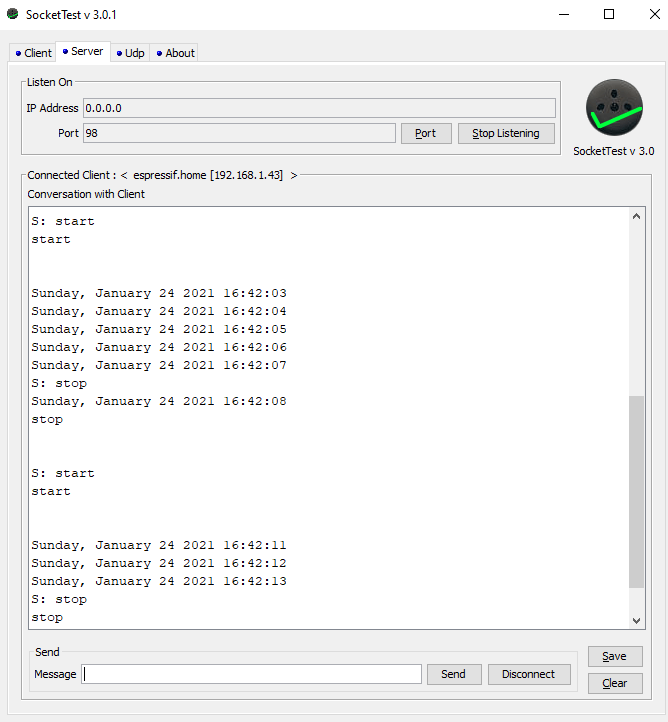


En caso de haber mensaje se comprueba si coincide con los comandos start o stop.



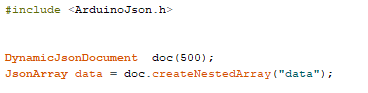
En caso de que el flag valga uno el programa mostrara la hora y, en caso de valer 0 no lo hará





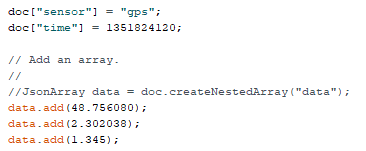
## Basándote en el estándar SENML crea un fichero JSON

Incluimos la biblioteca de JSON y creamos un documento y un array asignando memoria al documento. En este caso 500.

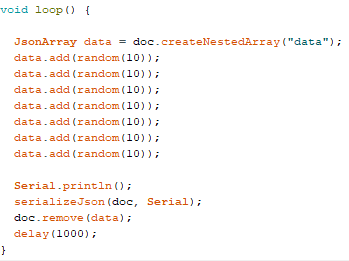


JSON permite añadir datos etiquetados. En este caso haremos uso de 3 etiquetas: Sensor, time y data (el array)

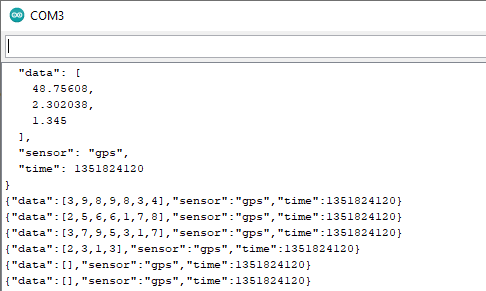
En el setup inicializamos estos grupos y añadimos los primeros datos



Ya en el loop añadimos más datos hasta completar la memoria RAM dedicada al archivo.

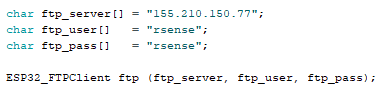


Aquí se comprueba cómo una vez lleno el documento deja de añadir datos

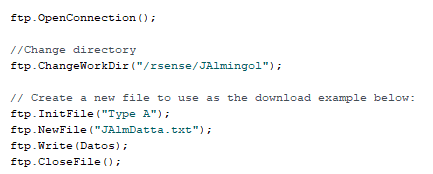


Continuamos añadiendo la comunicación ftp para enviar el documento al servidor.

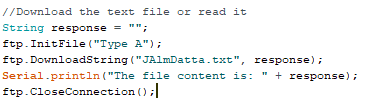
Nos conectamos a la red WIFI como en anteriores programas e incluimos esta vez la ip del servidor ftp más los datos de usuario y contraseña



En el setup abrimos la comunicación con el servidor, abrimos el directorio de trabajo, creamos el documento y escribimos en el los datos.



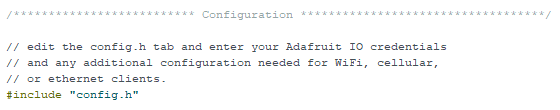
También podemos leer los datos del servidor



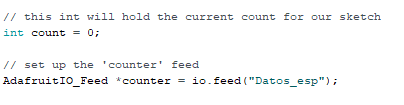
## Sube datos a la nube, en concreto al servicio gratuito proporcionado por Adafruit.

Creamos una cuenta en Adafruit y generamos una aplicación obteniendo un “AIO Key”. Y Usando MQTT enviamos datos al servidor.

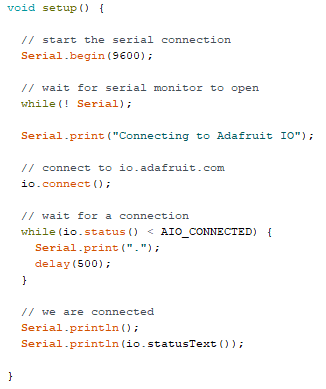
Para ello necesitamos adjuntar un archivo de configuración config.h con nuestras credenciales y el resto de datos necesarios como la red WIFI.



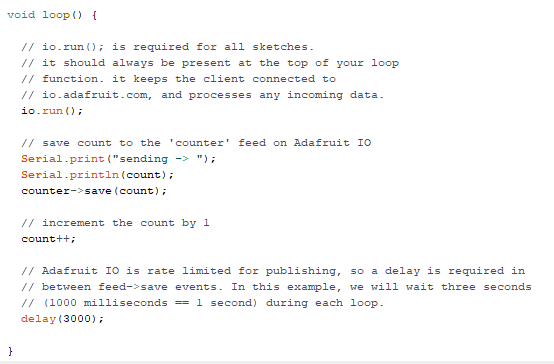
Se va a realizar un contador que se incremente constantemente y mande el dato a Adafruit



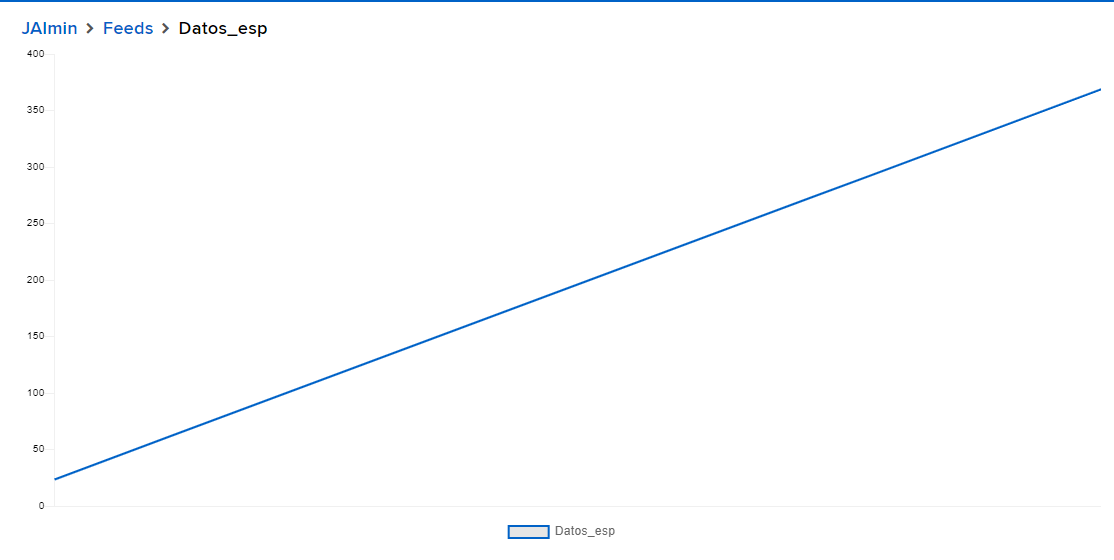
Dentro del Setup nos conectamos a Adafruit



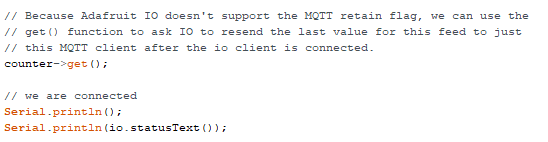
Y ya en el loop enviamos los datos incrementales cada 3 segundos. Para ello iniciamos la comunicación mediante run y guardamos el dato en cada iteración



Los datos en la web de AdaFruit



Para subir los códigos el mecanismo es el mismo cambiando en el setup el código para que lea datos

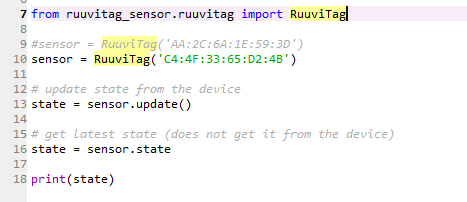


Y leyendo mediante el loop cualquier dato que se actualice en el servidor

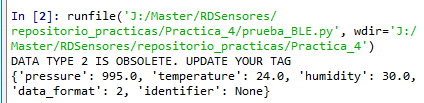


# Conexiones BLE y Bluethoot

## Extrae temperatura, humedad y presión de los mensajes recibidos utilizando Python del sensor “RuuviTagSensor”

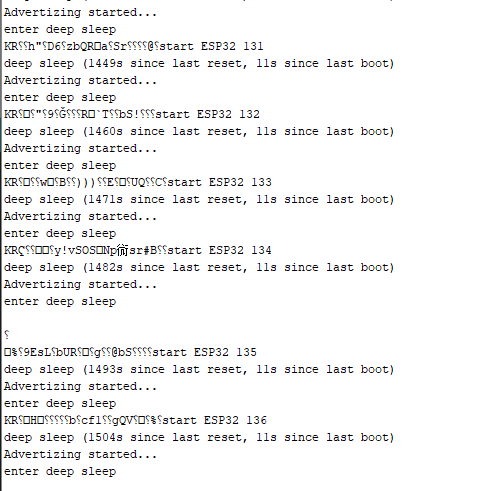


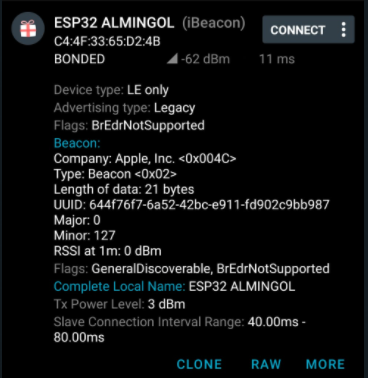
Obtenemos como resultado lo siguiente:



## Haz un advertising con tu módulo siguiendo la identificación iBeacon

Mediante los ejemplos de esp32BLE obtenemos el código que nos permite generar un anuncio. Con el siguiente resultado.





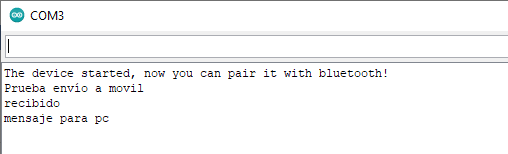
## Establece un “chat” Bluetooth con perfil SPP con una APP en el móvil

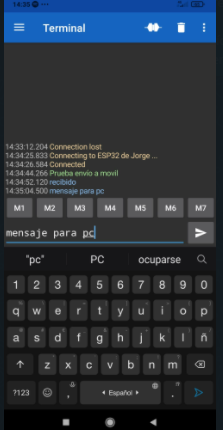
Conexión serie mediante bluethoot.

Creamos un servidor bluethoot en el esp32 que muestra por pantalla los mensajes que recibe y que envíe los mensajes que escribamos por pantalla a los clientes



Con el siguiente resultado:





# Comunicaciones Lora y LoraWAN

Para estas prácticas es necesario trabajar con micropython por tanto instalamos el entorno Visual Studio Code y en este instalamos el paquete Pymakr en la última versión 1.1.7. Además es necesario instalar NodeJS para que nos detecte la tarjeta y nos permita trabajar con ella. Utilizaremos la franja de 868 MHz para la comunicación. El modulo hardware queda de la siguiente forma:



## Establece comunicaciones ping-pong con otros compañeros basada en broadcasts.

Para esta conexión utilizamos el protocolo Lora ya que enviamos mensajes a cualquier destinatario, es decir, no están cifrados por tanto son legibles por cualquier receptor.

El código es el siguiente, encargado de crear un socket y enviar por el los mensajes escritos por línea de comandos, además muestra por pantalla los mensajes que pueda recibir de otros dispositivos.

#!/usr/bin/env python

#

# Copyright (c) 2019, Pycom Limited.

#

# This software is licensed under the GNU GPL version 3 or any

# later version, with permitted additional terms. For more information

# see the Pycom Licence v1.0 document supplied with this file, or

# available at https://www.pycom.io/opensource/licensing

#

from network import LoRa

import socket

# Initialize LoRa in LORA mode.

# More params can be given, like frequency, tx power and spreading factor.

# Please pick the region that matches where you are using the device:

# Asia = LoRa.AS923

# Australia = LoRa.AU915

# Europe = LoRa.EU868

# United States = LoRa.US915

lora = LoRa(mode=LoRa.LORA, region=LoRa.EU868)

# create a raw LoRa socket

s = socket.socket(socket.AF\_LORA, socket.SOCK\_RAW)

while True:

# send some data

s.setblocking(True)

mensaje = imput()

s.send(mensaje)

# get any data received...

s.setblocking(False)

data = s.recv(64)

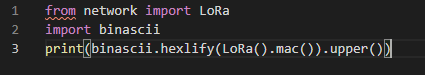
print(data)

time.sleep(1)

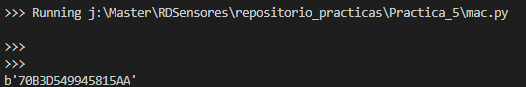
## Envía cada 30 segundos un número incremental de 2 bytes con protocolo Lorawan a un servidor en la nube.

Para este apartado nos registramos en la web oficial de The Things Network (TTN) y creamos una aplicación. Posteriormente registramos la dirección de nuestro dispositivo y ya tenemos la aplicación asignada a nuestro dispositivo y las credenciales correspondientes necesarias para hacer la comunicación punto a punto.

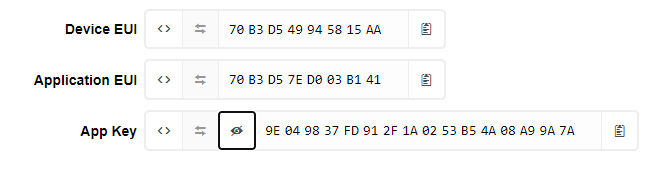
Código de identificación del dispositivo



Nos remite la siguiente dirección



Y con ella identificamos el dispositivo, obteniendo las credenciales en la web de TTN:



Una vez tenemos estos datos podemos crear la aplicación que se encarga del envió de información al servidor mediante el protocolo OTAA:

from network import LoRa

import socket

import time

import ubinascii

lora = LoRa(mode=LoRa.LORAWAN, region=LoRa.EU868)

# create an OTAA authentication parameters, change them to the provided credentials

app\_eui = ubinascii.unhexlify('70B3D57ED003B141')

app\_key = ubinascii.unhexlify('9E049837FD912F1A0253B54A08A99A7A')

#uncomment to use LoRaWAN application provided dev\_eui

#dev\_eui = ubinascii.unhexlify('70B3D549938EA1EE')

# join a network using OTAA (Over the Air Activation)

#uncomment below to use LoRaWAN application provided dev\_eui

lora.join(activation=LoRa.OTAA, auth=(app\_eui, app\_key), timeout=0)

#lora.join(activation=LoRa.OTAA, auth=(dev\_eui, app\_eui, app\_key), timeout=0)

# wait until the module has joined the network

while not lora.has\_joined():

    time.sleep(2.5)

    print('Not yet joined...')

print('Joined')

# create a LoRa socket

s = socket.socket(socket.AF\_LORA, socket.SOCK\_RAW)

# set the LoRaWAN data rate

s.setsockopt(socket.SOL\_LORA, socket.SO\_DR, 5)

# make the socket blocking

# (waits for the data to be sent and for the 2 receive windows to expire)

s.setblocking(True)

# send some data

s.send(bytes([0x01, 0x02, 0x03]))

# make the socket non-blocking

# (because if there's no data received it will block forever...)

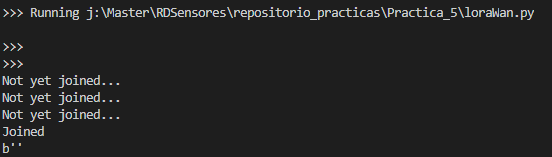
s.setblocking(False)

# get any data received (if any...)

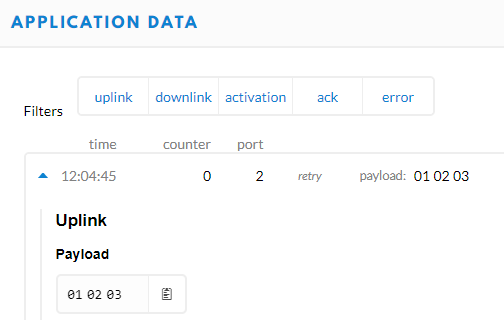
data = s.recv(64)

print(data)

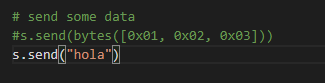
Ejecutamos el programa



Y se comprueba en la web TTN que se ha recibido la información:



En caso de enviar un comando de texto:



El servidor lo recibe en hexadecimal, no en ASCII:

