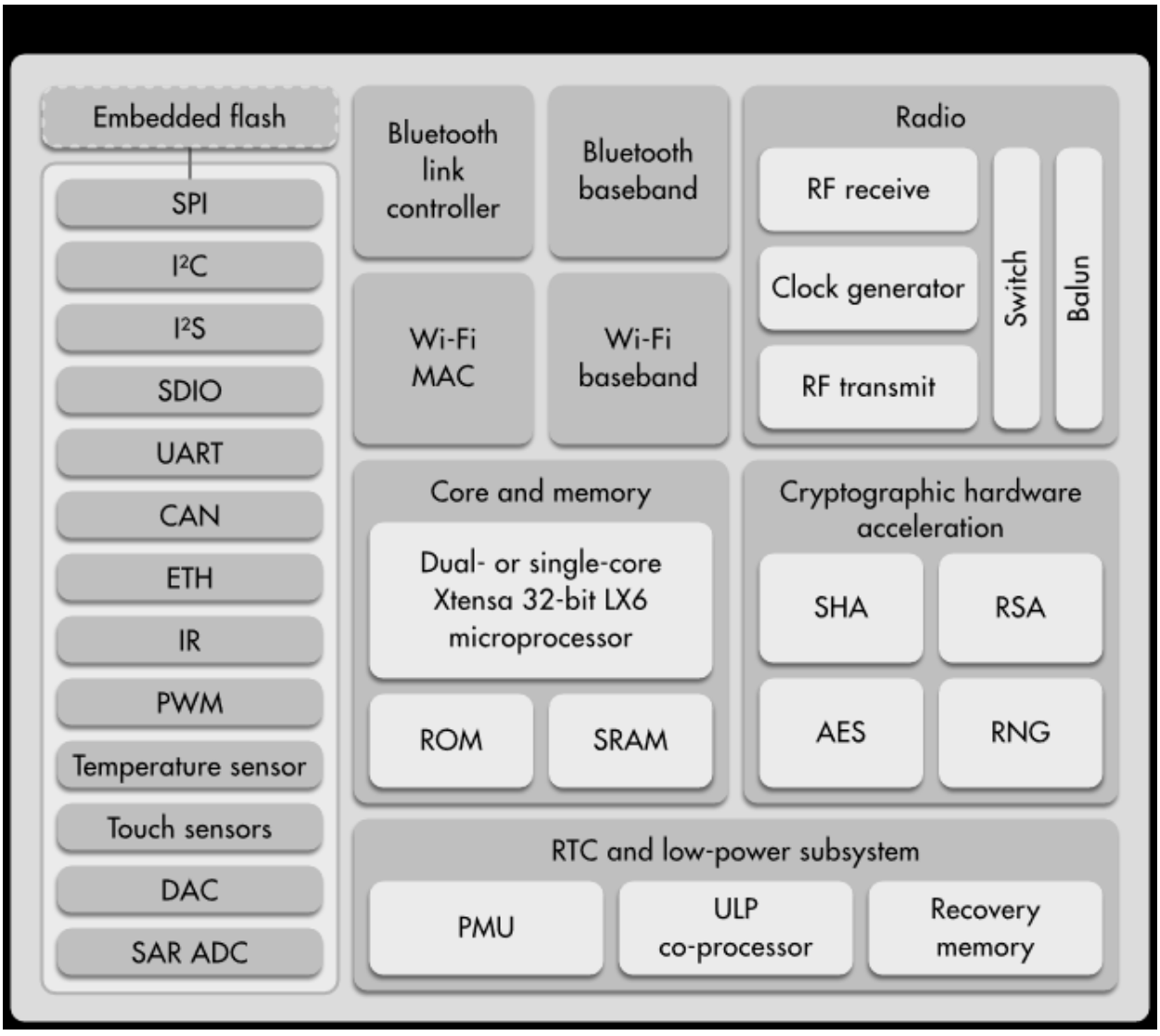
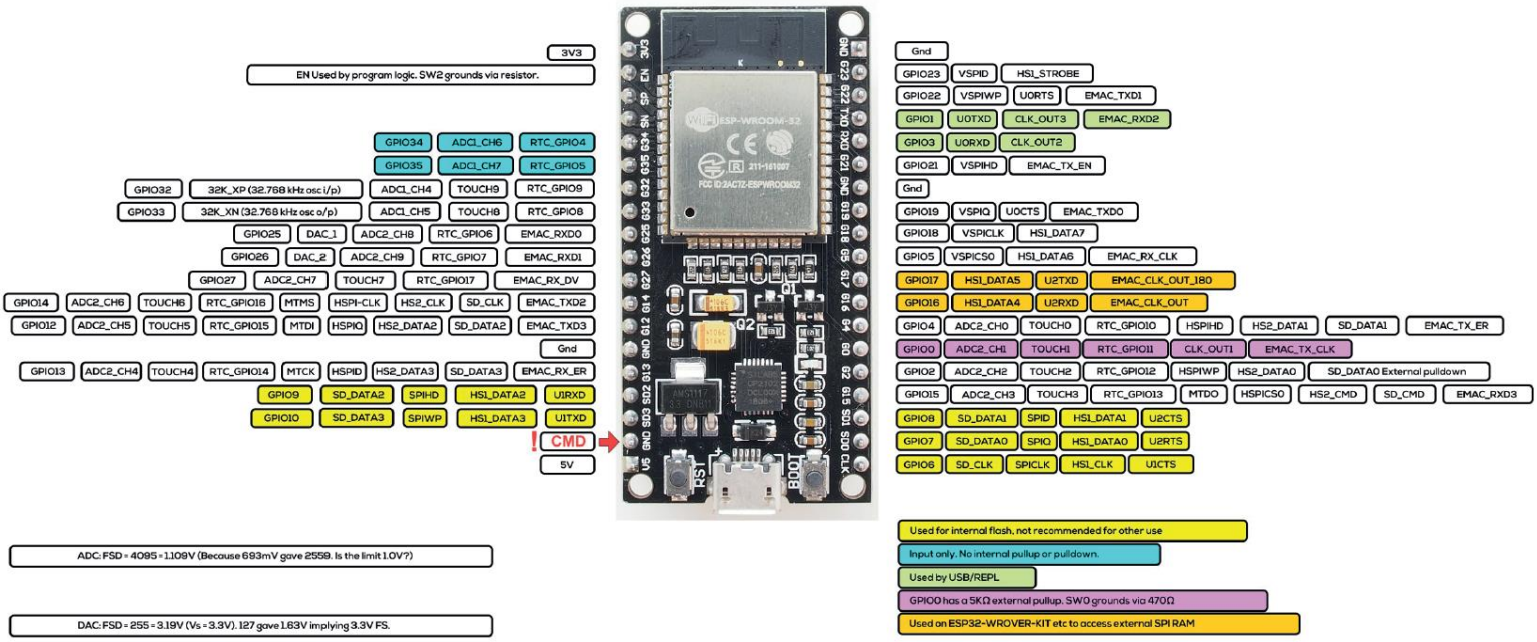
Arquitectura ESP32

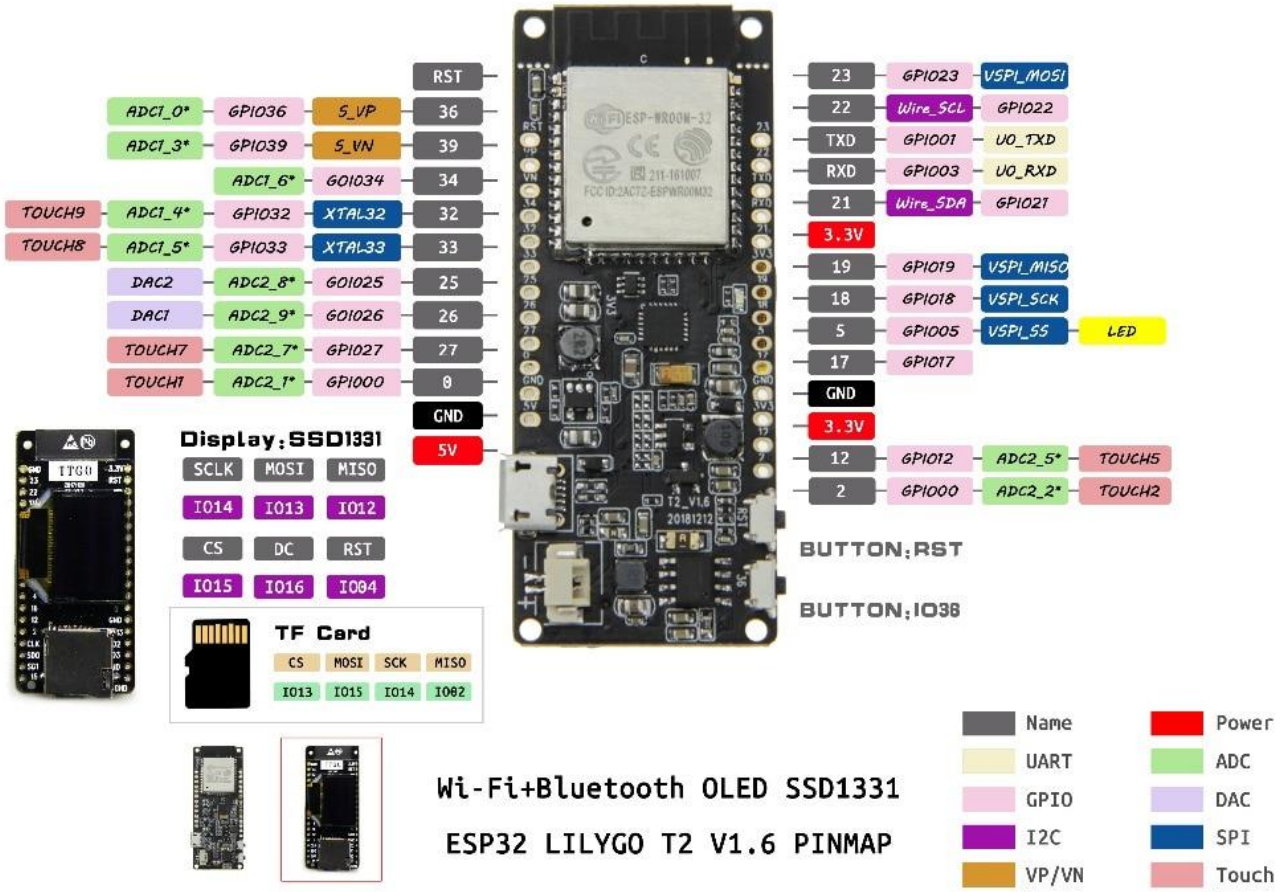
Los objetivos son:

- Familiarizarse con el sistema de desarrollo ESP32
- Manejar los periféricos típicos de cualquier microcontrolador: ADC, PWM, Timer, etc.

Trabajaremos con la placa de desarrollo basada en el ESP32:
https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf
O con un kit con pantalla y memoria SD: https://es.aliexpress.com/item/32843038592.html







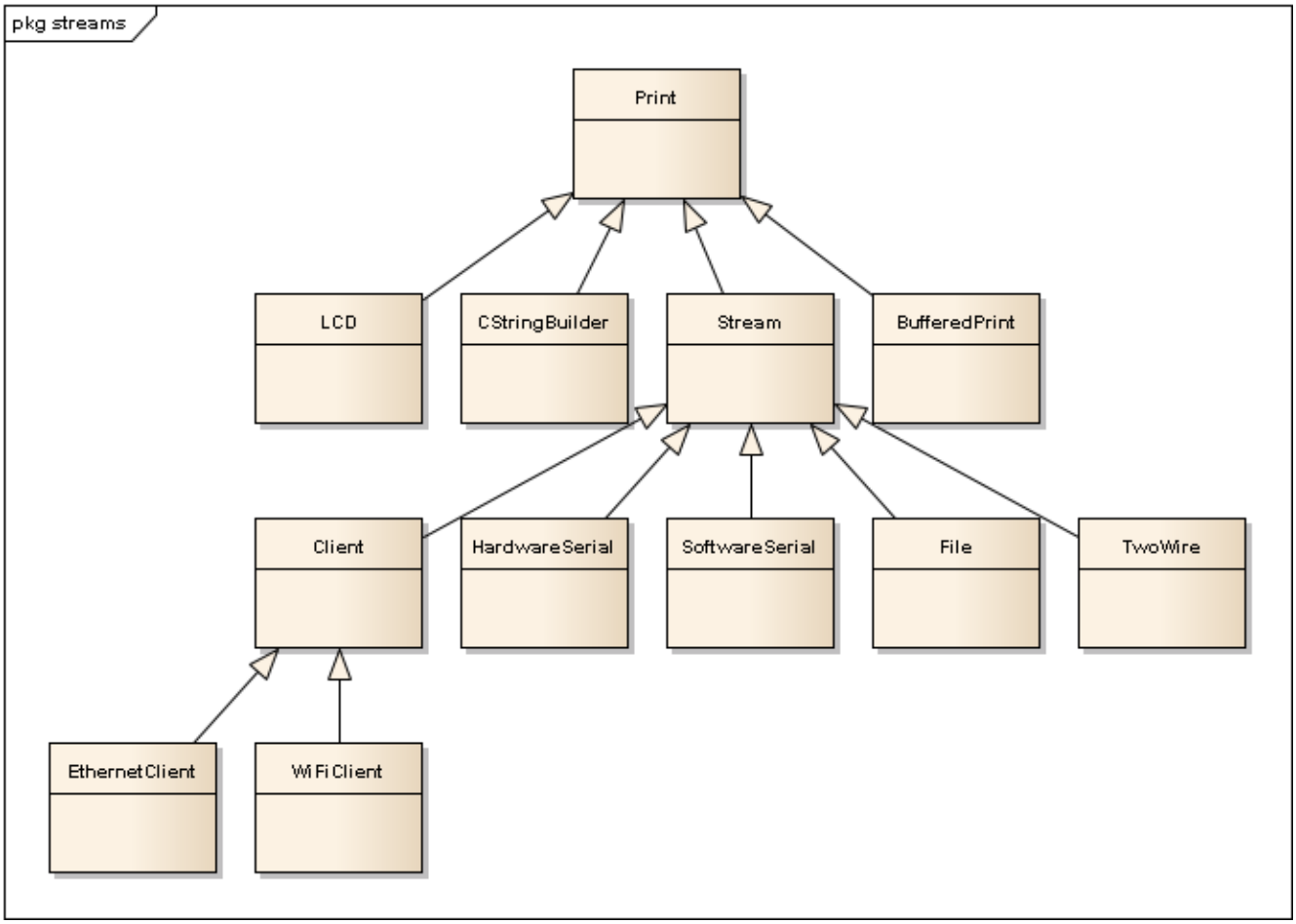
Instalación del entorno en: https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/

Documentación de librerías: https://github.com/espressif/arduino-esp32/tree/master/libraries

Se plantea la configuración y control de diferentes módulos de hardware:

1. Con una periodicidad de un segundo, lee el valor del ADC y muéstralo por consola. **Utiliza la salida de 3V3 de la placa con un potenciómetro para generar la tensión analógica. Antes de conectarla al módulo comprueba que no superas 3V3 con un multímetro.**
   https://arduinobasics.blogspot.com/2019/05/sprintf-function.html
2. Utilizando los timer hardware genera una interrupción cada 10s que lea el ADC y lo muestre por pantalla (https://techtutorialsx.com/2017/10/07/esp32-arduino-timer-interrupts/). **[ENSEÑAR]**
3. Genera una salida pwm a 5kHz proporcional a la lectura del ADC. Comprueba en el osciloscopio tanto la medida analógica como la salida PWM.
4. Comanda por la UART los periféricos mediante un protocolo de tal modo que si le mandas **[ENSEÑAR]**:
   a. ADC: Envíe la lectura del ADC actual
   b. ADC(x): envíe la lectura del ADC cada x segundos. Si x=0, deja de mandar datos
   c. PWM(x): comanda el duty cycle del módulo PWM con números del 0 al 9.

   Utiliza el objeto string y sus métodos
   https://www.arduino.cc/reference/en/language/variables/data-types/string/
   (https://www.arduino.cc/reference/en/language/functions/communication/serial/)
   https://www.arduino.cc/reference/en/language/functions/communication/stream/
5. Conecta un sensor inercial por I2C (o SPI), muestrea la aceleración cada 100 ms y manda los datos cada segundo vía UART (cada vez que envíes activa un LED durante 200ms).
   Utilizaremos el sensor https://es.aliexpress.com/item/32843038592.html y
   https://es.aliexpress.com/item/1904029297.html u otro similar **[ENSEÑAR]**



Diseño de firmware basado en sistema operativo de tiempo real (RTOS)

Los objetivos son:

1. Comprender el uso y los conceptos asociados a un sistema operativo en tiempo real (RTOS)
2. Diseñar un firmware basado en RTOS

Trabajaremos con el sistema FreeRTOS: https://www.freertos.org/Documentation/RTOS_book.html

1. Crea un programa que cree dos tareas, una que parpadee un LED cada 200 ms y otra que envíe un "hola mundo" por la UART cada segundo. https://github.com/uagaviria/ESP32_FreeRtos
2. Implementa la tarea 5 del punto anterior con FREETOS (Conecta un sensor inercial por I2C (o SPI), muestrea la aceleración cada 100 ms y manda los datos cada segundo vía UART (cada vez que envíes activa un LED durante 200ms)) **[ENSEÑAR]**

Comunicación, gestión y representación de datos de sensores con Python.

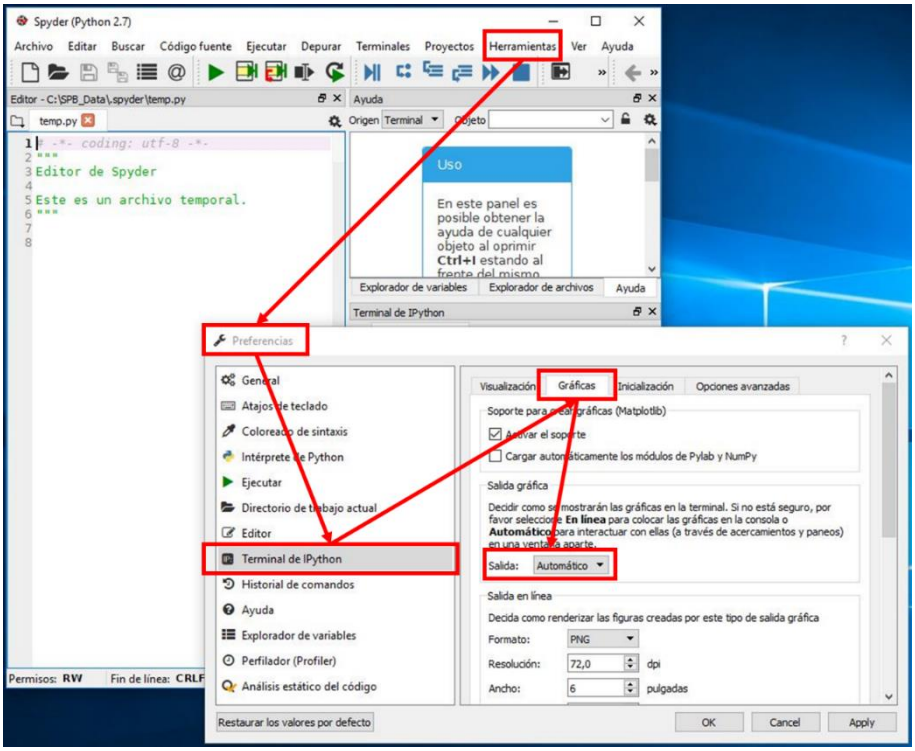
Los objetivos son:

- Familiarizarse con Python como lenguaje de programación
- Gestionar datos procedentes de comunicaciones serie
- Trabajar con estos datos en formato csv y json para almacenarlos en ficheros y redireccionarlos vía internet
- Operar con los datos y representarlos gráficamente

Partimos de la anterior práctica en la que se conecta un sensor inercial por I2C (o SPI), muestrea la aceleración y manda los datos vía UART cada 50 ms al PC.

Las tareas a realizar:

0. Si no sabes Python, primero necesitas un tutorial como por ejemplo https://uniwebsidad.com/libros/python (aquí tienes prácticamente todo lo necesario para esta práctica). La referencia oficial la tienes en https://docs.python.org/3/
1. Conecta el SMART IMU y comprueba la recepción de datos en el puerto serie con cualquier programa como TERMITE (https://www.compuphase.com/software_termite.htm).
2. Programa en Python un programa que acceda al puerto serie y muestre por pantalla los datos en tiempo real (igual que el terminal). Necesitarás instalar el objeto pyserial como administrador.
3. Modifica el programa para que almacene los datos en fichero .txt separando cada variable con ";" y con retorno de carro al final de cada muestra para abrirlo desde Excel. **[ENSEÑAR]**
   Previamente guarda un fichero con texto fijo para asegurarte que generas el fichero adecuadamente.
4. Modifica el programa para que acumule los datos durante 5 segundos, calcule el promedio y desviación estándar los represente en tiempo real. Necesitarás instalar la librería matplotlib (u otra de representación de gráficas) y configurar spyder para que no genere una gráfica detrás de otra, sino que permita la actualización automática de la gráfica tras cada muestra ploteada. **[ENSEÑAR]**



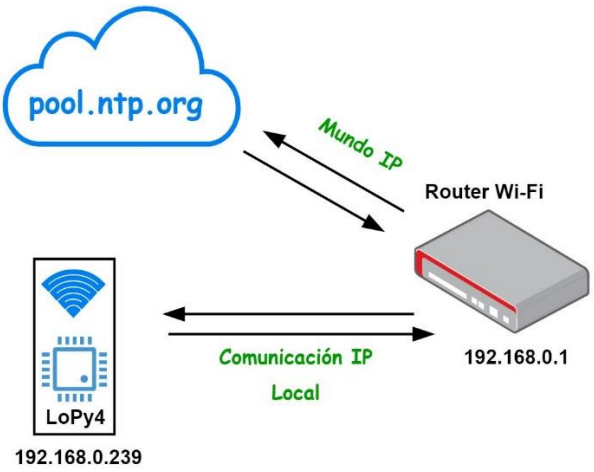
Comunicaciones WIFI y stack IP

Los objetivos son:

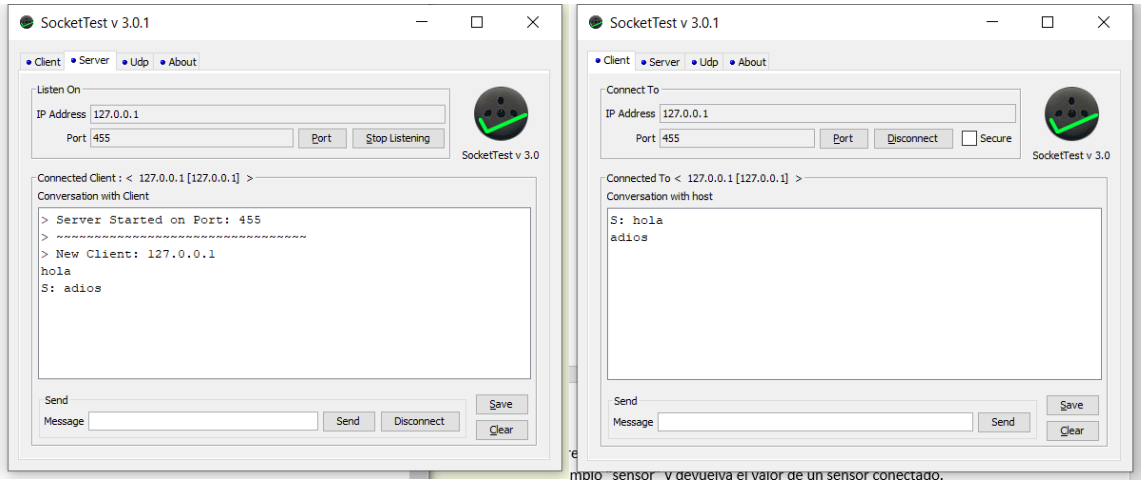
- Poner en práctica los conocimientos de redes inalámbricas WIFI
- Manejar comunicaciones IP a bajo nivel mediante sockets y entre diversas plataformas: PC, Móvil, CLOUD, Sensor
- Manejar protocolos de alto nivel: HTTP, FTP, NTP, MQTT y estándares de interoperabilidad: SENML

Las tareas a realizar son:

1. Conéctate a la red wifi del laboratorio o a una creada por el móvil como punto de acceso. Extrae tu IP. Comprueba la conectividad con Google mediante un ping.
2. Pon en hora el módulo mediante un servidor NTP (Network Time Protocol)
   https://www.esp32.com/viewtopic.php?t=5978
   https://lastminuteengineers.com/esp32-ntp-server-date-time-tutorial/



3. Monta un chat una aplicación software PC (http://sockettest.sourceforge.net/) o con una aplicación móvil (simple TCP socket tester). A veces el firewall del ordenador no permite las conexiones externas, y es necesario configurarlo correctamente.



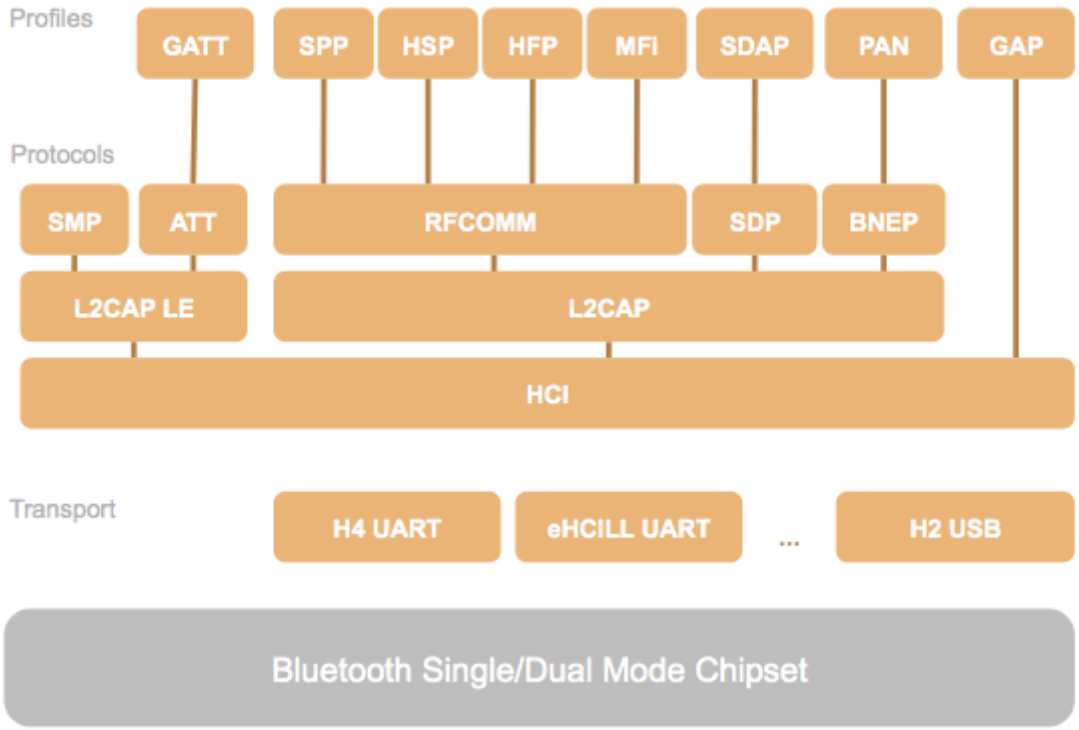
4. Después sustituye uno de los extremos por el módulo hardware siendo cliente y envía cada segundo tu hora local.

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”. **[ENSEÑAR]**
6. Monta un servidor WEB (https://randomnerdtutorials.com/esp32-web-server-spiffs-spi-flash-file-system/) que muestre la hora y tenga un botón para resetear la hora a las 0:00 **[ENSEÑAR]**
7. Basándote en el estándar SENML (https://tools.ietf.org/html/rfc8428) crea un fichero json (https://arduinojson.org/) que se genere cada 10 segundos, que contenga datos de temperatura inventados, las unidades y la marca temporal. Súbelos a un servidor ftp con un nombre que sea grupoXX_ddmmss.json. (https://platformio.org/lib/show/6543/esp32_ftpclient) Usa el del laboratorio (IP: xxx) o móntate uno con https://filezilla-project.org/download.php?type=server (asegúrate que el firewall permite conexiones entrantes). **[ENSEÑAR]**
8. Sube datos a la nube, en concreto al servicio gratuito proporcionado por Adafruit.
    a. En primer lugar se debe de crear un usuario en https://io.adafruit.com/, generar una aplicación y obtener un “AIO Key” y crear un feed al que subir datos. Leer info en https://learn.adafruit.com/adafruit-io-basics-feeds
    b. Sube datos desde navegador (https://www.apirequest.io/) utilizando POST HTTP según documentación (https://io.adafruit.com/api/docs/?shell#create-data)
    c. Usa librería de Adafruit IO (https://github.com/adafruit/Adafruit_IO_Arduino) para subir datos al feed usando MQTT.
    d. Usa la librería para suscribirte al feed y comprueba que recibes actualización al escribir desde el navegador. **[ENSEÑAR]**
    e. https://github.com/plapointe6/EspMQTTClient

Comunicaciones BLE y Bluetooth

Los objetivos son:

- Leer mensajes BLE emitidos por un sensor
- Emitir mensajes de advertising vía BLE
- Establecer una comunicación bidireccional basada en perfil SPP de Bluetooth



Para familiarizarse con la implementación Bluetooth:

- https://randomnerdtutorials.com/esp32-bluetooth-low-energy-ble-arduino-ide/

Las tareas a realizar son:

1. En el laboratorio habrá un sensor "RuuviTagSensor" que emite mensajes según el formato https://github.com/ruuvi/ruuvi-sensor-protocols/blob/master/dataformat_04.md (formato 2). Descarga la aplicación "nRF Connect" y escanea dispositivos BLE en el entorno. Extrae temperatura, humedad y presión de los mensajes recibidos utilizando Python (https://pypi.org/project/ruuvitag_sensor/).
   Para adivinar cual de todos los dispositivos descubiertos es el sensor, lo acercaremos al móvil y observar con que potencia se recibe el broadcasting que hacen los dispositivos. De todos los dispositivos escogemos el que mayor potencia de transmisión tenga y apuntaremos esa MAC.
2. Haz un advertising con tu módulo siguiendo la identificación iBeacon (https://en.wikipedia.org/wiki/IBeacon) incluyendo el número de grupo dentro del campo UUID **[ENSEÑAR]**.
3. Escanea con el módulo hardware y reporta los valores decodificados por el terminal vía terminal serie en formato JSON-SENML **[ENSEÑAR]**.
4. Establece un "chat" Bluetooth con perfil SPP con una APP en el móvil: https://github.com/espressif/arduino-esp32/tree/master/libraries/BluetoothSerial

Comunicaciones Lora y LoraWAN

Los objetivos son:

- Establecer una comunicación bidireccional basada en Lora
- Emitir mensajes LoraWAN a servidor en la nube

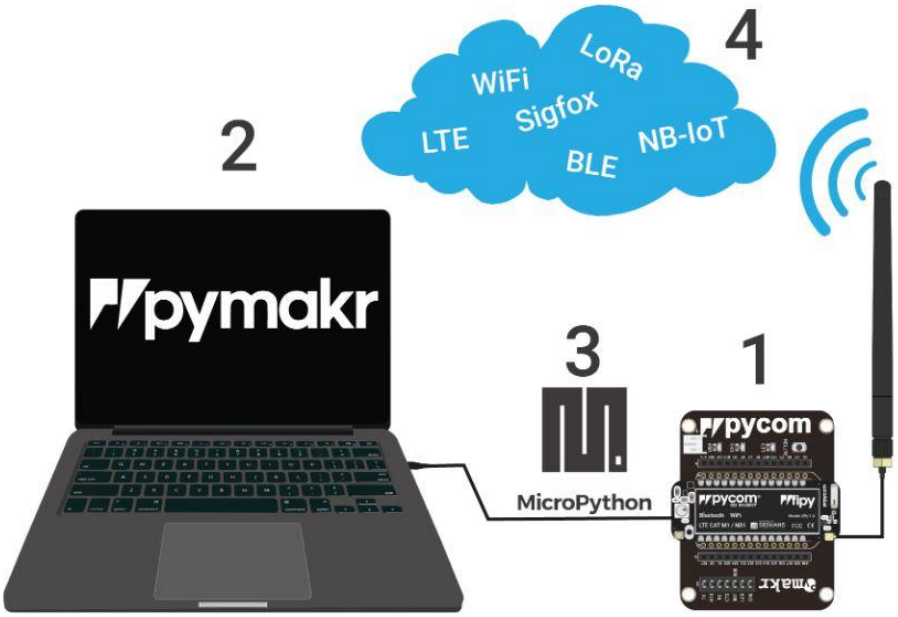
Para familiarizarse con Lora y Lopy4 se utilizarán los tutoriales de la página oficial:

- https://docs.pycom.io/firmwareapi/pycom/network/lora/
- https://docs.pycom.io/tutorials/lora/lorawan-abp/

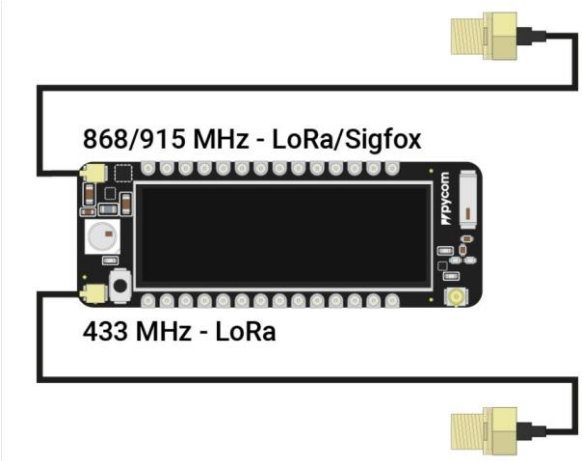
Y la documentación que se aporta en la pagina oficial de TTN “The Things Network”:

- https://www.thethingsnetwork.org/docs/lorawan/architecture.html

Para comenzar, a parte del HW del módulo LoPy4 necesitaremos una Antena para poder realizar las comunicaciones a cierta distancia y la suficiente potencia sin dañar al módulo.

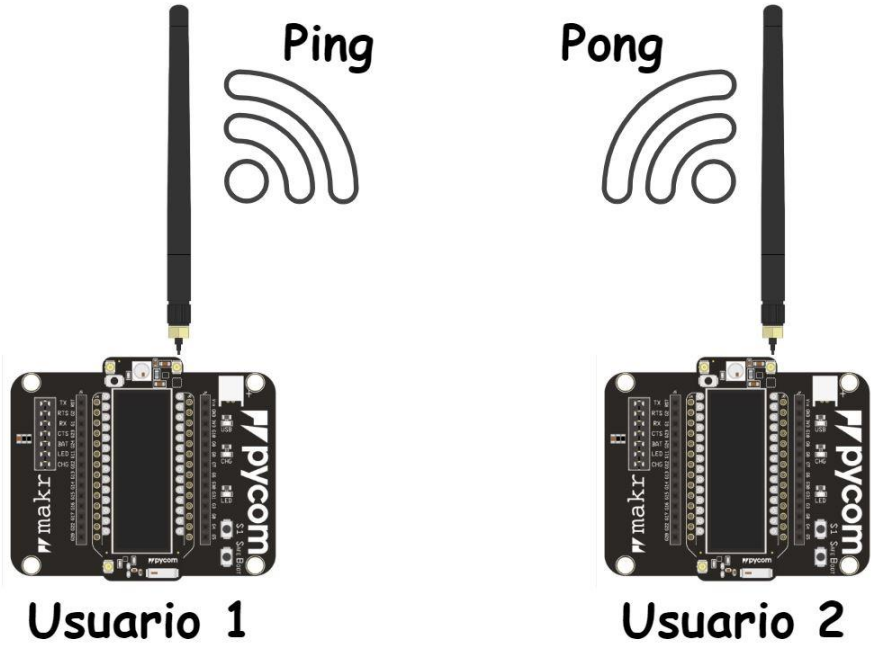


La antena deberá de conectarse al conector de 868MHz del módulo LoPy4, ya que es la franja de frecuencia permitida en Europa para protocolo LoRa.

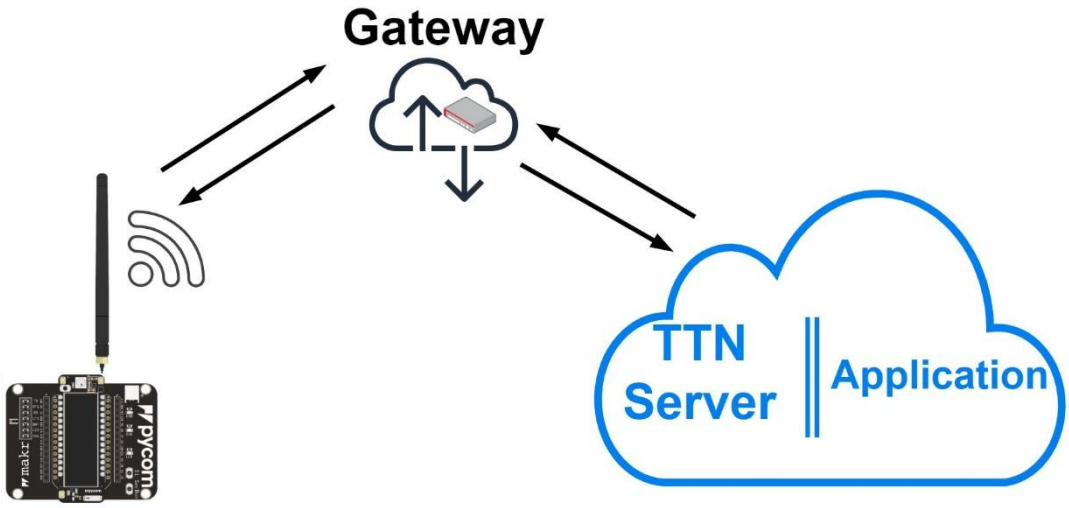


Las tareas a realizar son:

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



2. Envía cada 30 segundos un número incremental de 2 bytes con protocolo Lorawan a un servidor en la nube. Para poder realizar una comunicación con el servicio Cloud de TTN hay que registrarse dentro de la web oficial (https://www.thethingsnetwork.org/docs/devices/lopy/).



Recursos interesantes:

- https://www.arduino.cc/en/Hacking/LibraryTutorial
- https://cpp4arduino.com/2018/11/06/what-is-heap-fragmentation.html
- https://cpp4arduino.com/2018/11/21/eight-tips-to-use-the-string-class-efficiently.html
- https://cpp4arduino.com/2018/10/23/what-is-string-interning.html
- https://www.arduino.cc/reference/en/language/variables/utilities/progmem/

Para el trabajo:

https://techtutorialsx.com/2018/05/17/esp32-arduino-sending-data-with-socket-client/

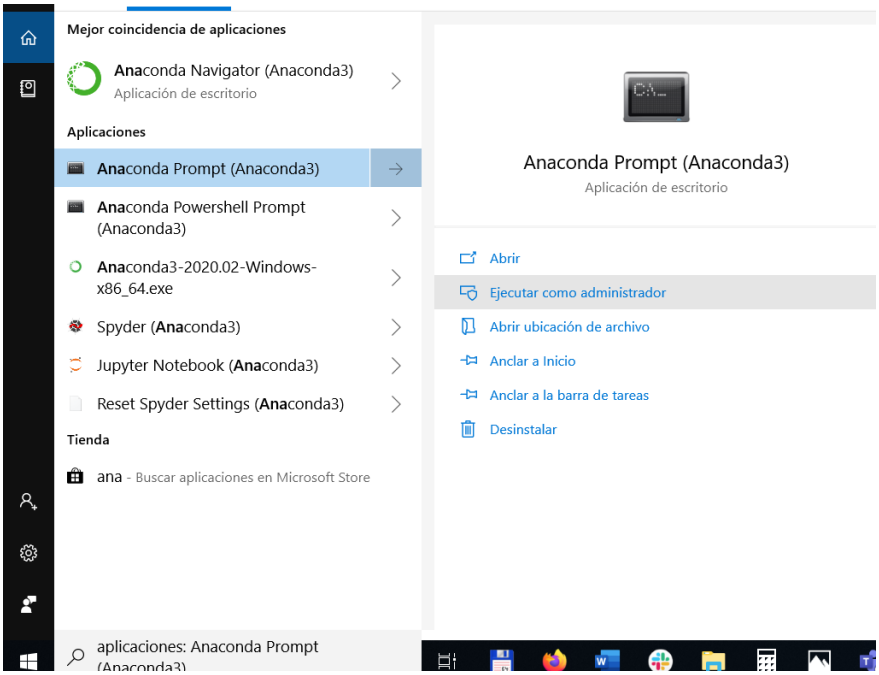
Terminal serie: https://www.compuphase.com/software_termite.htm

Arduino:

- https://www.arduino.cc/en/Main/Software
- Instalación del entorno ESP32 en: https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/

Python:

- https://www.anaconda.com/distribution/#download-section (versión 3.7)
- Instalar librerías:



- Pyserial: pip install pyserial
- Matplotlib: pip install matplotlib