Table of Contents

1.- Presentación (breve)

Trataremos en este ‘blog’ de explicar como se ha desarrollado y puesto a punto una ‘aplicación’ (o ‘programa’, o ‘librería’ -que es lo que es en realidad-) para captura de parámetros físicos (conversión analógico digital) empleando python como lenguaje de programación. Como hardware se emplean -sucesivamente- A) un Raspberry sin hardware añadido para el bus oneWire, B) la raspberry más una conversor analógico digital tipo ADS1115 y C) la raspberry conectada a un arduino ProMini; en el primer caso los sensores son del tipo DS18B20, para los otros dos se emplean media docena de sensores convencionales -temperatura, tensión- que se describen en su momento. Como se irá explicando en el blog, se hace mucho énfasis en la justificación de la arquitectura de (clases que componen) la librería y del desarroloo de las clases, cuestiones que habitualmente resultan las más complejas en este tipo de aplicaciones.

El blog puede ser de interés:

- Como guía de prácticas de programación en cursos de grado medio (formación profesional); es posible que incluso en ciertas especialidades de grados universitarios sirva para idéntica finalidad.

- Para aficionados a (o profesionales de) la programación que deseen mejorar o comprobar sus conocimientos de OOP.

- Profesionales en el campo de la ‘adquisición de datos’; aunque el hardware a emplear no tiene grado industrial (no, al menos, el que especificamente se ha empleado para elaborar este blog), no es descartable que pueda ser de su interés.

Se procura que las explicaciones sean todo lo concisas que permite el asunto, que, como se comenta más adelante, no es un problema simple.

(Si solamente quiere familiarizarse con el uso de la librería y no con su diseño puede ir directamente al punto 8, “Uso de la librería”).

1.1.- Objetivo del blog.

El objetivo declarado del blog es, pués, la elaboración de una -pequeña- biblioteca escrita en python para facilitar el uso de varios tipos de adaptadores AD, en principio un ADS1115, el bus “oneWire” y un arduino Promini. El primero y el ultimo son añadidos de hardware al raspberry, mientras que el bus oneWire es nativo de las placas raspberry: cargado el correspondiente driver se pueden habilitar los pines del conector GPIO que se deseen para implementarlo.

Configuración del raspberry: Dependiendo de la versión del raspberry que tenga y de la “distribución” de raspbian, tendrá su raspberry configurado para el bus i2c (necesario para que funcione el ADS1115) y/o el oneWire (necesario para el bus del mismo nombre) y/o la comunicación serie (necesaria para el arduino). Consulte el anexo CC, “Configuración de Raspbian” para más detalles.

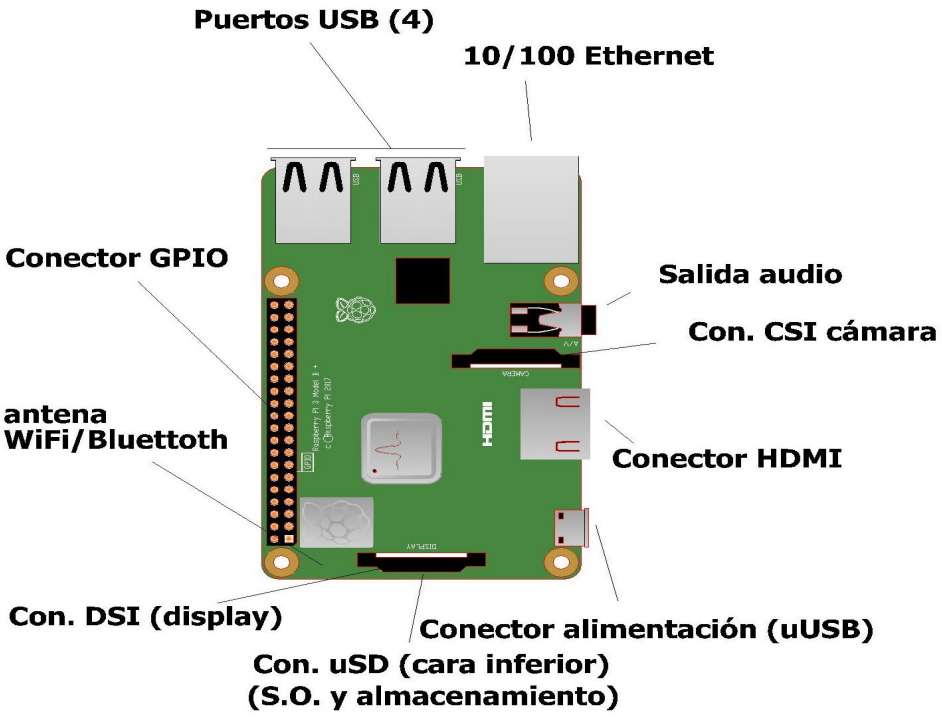

Figura 1. Raspberry 3B +

Y, sin embargo, el blog no se limita a esto: aunque efectivamente se desarrolle -se escriba- la biblioteca tal y como se ha descrito en el párrafo anterior, hay otro objetivo tan importante como el anterior (o más), al menos para quién desee aprender algo de análisis y programación OOP. Esperemos que el viaje, el análisis de necesidades, descomposición del problema en clases, el estudio de la arquitectura de la biblioteca, sea tan provechoso como el destino del mismo, el mero uso de la misma.

1.1.1.- Arquitectura de aplicaciones en OOP.

Para centrar la cuestión: al abordar el análisis de un desarrollo con OOP, uno de los problemas centrales es la arquitectura del sistema; i.e.: ¿Cómo encuentro las *clases* -que, después, se instanciarán para dar lugar a los *objetos*- que forman mi aplicación?. . ¿Cuánto debo desmenuzar -*granularity*- el problema?. ¿En cuántas clases?. ¿Qué relación tienen entre ellas?. ¿Y los objetos entre sí?. ¿Cómo se documenta?.

El problema específico -la librería propiamente dicha- que se aborda en este blog es relativamente simple: se resuelve con una docena mal contada de clases (como veremos en su momento); su mejor cualidad es que está desarrollado (que se dispone del código de la librería y que, además, funciona). El lector, al que en más de un momento es probable que le *tiemblen las piernas*, por lo menos tiene por seguro que llegará a buen puerto.

Es decir, que la mejor virtud de este blog es precisamente que se trata de un ejemplo completamente desarrollado, y no de describir las bases del análisis OOP, cuestión compleja y que requiere de bastante más que este blog. Si desea seguirlo -el blog- tenga en cuenta esto: se trata de un ejemplo desarrollado (programado, funcionando). Nada más (ni nada menos).

1.1.2.- Sistemas de adquisición de datos. Conversión A/D.

Si de lo que se trata es de hacer pedagogía de OOP podríamos haber escogido otra materia que no fuese la de conversión A/D. ¿Qué tiene este campo que no tengan otros para haberlo elegido como hilo conductor del blog?.

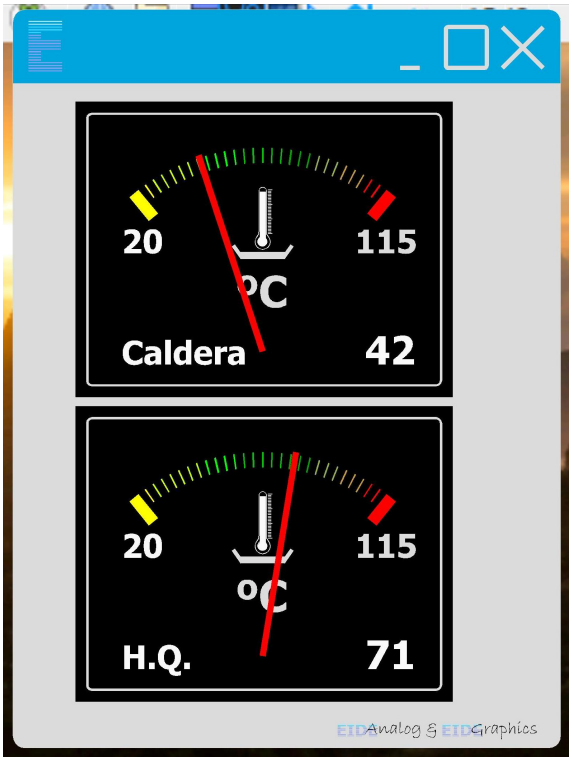

Figura 2. Sistema de adquisición de datos con Raspberry.

Que los autores -del blog- lo conocen bien (y son perfectamente conscientes de que la elección de una biblioteca para “para captura de parámetros físicos”, como se dice en la tercera línea del primer punto de este blog, le quitará una buena parte de seguidores que, de haber elegido otro tema, quizá se habrían animado a seguirlo).

Pero, por otra parte, el asunto tiene también sus atractivos, no siendo el menor de ellos que el resultado nos permitirá conectar sensores a un ordenador (cuestión que, en muchos sentidos, se puede considerar una excentricidad). Es este un asunto que interesa de forma transversal a un buen número de actividades, tanto en el ámbito de la industria como en el de la formación, y en este -el de la formación- a todas las ingenierías y grados en ciencias físicas o químicas. Al fin y al cabo, y salvo las interioridades del funcionamiento del dispositivo de conversión A/D propiemente dicho -que sí son específicas de esta disciplina-, el resto no es mucho más que álgebra y lógica binaria; una vez que se está conforme con la finalidad de lo que se está haciendo -la biblioteca-, la forma de hacerlo -el análisis- sí que es válido para problemas similares -de arquitectura similar.

1.1.3.- Desarrollo del blog. Pedagogía.

En lo que sí se ha sido muy cuidadoso es en el desarrollo del blog en sí mismo, adoptando un estilo fundamentalmente pedagógico. No se trata, solamente, de escribir unas clases para que con un “corta-pega” se incorporen a un proyecto (y, sin embargo, esto se puede hacer sin ningún problema -y funciona). El objetivo principal del blog es, como ya se ha dicho, ayudar a quien desee analizar un ejemplo de cómo encarar el análisis y desarrollo de una aplicación empleando las técnicas de la OOP.

A tal efecto, y en aras de no complicar demasiado la presentación no se incluye, de entrada, el código de captura de errores (ver 1.2.4, “[Conocimientos previos necesarios] Python. OOP y el apédice NN, ‘Código a prueba de errores. Bloques “Try-except-finally”’), que hace el código final bastante más confuso y puede desanimar al principiante. En el mismo anexo puede encontrar el código completo (“Pro”) que difiere del que se va a ir mostrando (“Estudiante”) sólamente en este aspecto (en que tiene tratamiento de errores); el código que se va a ir presentando, se insiste, es plenamente operativo.

1.1.4.- Conclusión. Objetivos.

Al acabar el blog (con aprovechamiento), usted habrá adquirido ciertas habilidades en lo referente a la OOP, vea 2.5.3, “OOP. Conclusión.”. Subsidiariamente, aprenderá:

- (Conceptos básicos) de adquisición de datos. ADC.
- Manejo de un Raspberry a nivel de usuario. Conceptos (muy) básicos de Linux.
- (Rudimentos de) funcionamiento de los buses oneWire, i2c y la comunicación serie (UART).
- Funcionamiento de algunos sensores comerciales (y *populares*).
- Termistores. NTC. Sensores tabulados.
- Tratamiento de errores con python (eso si se estudia el anexo correspondiente: “Anexo NN. Código a prueba de errores. Bloques ‘Try-except-finally’, si no, no).

1.2.- Conocimientos -previos- necesarios para seguir el blog.

Los autores del blog han sufrido en sus carnes, y en más de una ocasión, manuales que empezaban -entusiastamente- explicando que para usar el ratón de un ordenador hay que manejarlo con la mano, moverlo sobre la alfombrilla y pulsar el botón de la izquierda para seleccionar y el de la derecha para las opciones (y que se puede configurar al revés para zurdos); inevitablemente el manual -o lo que fuera- abandonaba este nivel de detalle al segundo párrafo para, lo que es peor, dar instrucciones indescifrables apenas tres párrafos más adelante (y que, cada una de ellas -de las instrucciones indescifrables-, habrían merecido por sí mismas un manual).

Procuraremos no caer en este error; sería de ilusos pensar que para seguir este blog no haga falta ningún conocimiento previo en las materias que se manejan. Quien quiera seguirlo deberá tener *conocimientos de usuario* (signifique esto lo que signifique; ustedes ya me entienden) de informática, de ordenadores personales. Como, además, usamos un raspberry como base, usted debería hacerse con el manejo de este (naturalmente, siempre que se haga referencia al raspberry estamos incluyendo también el sistema operativo -linux).

Figura 3. ADS1115 conectado al raspberry.

Tampoco le tiene que hacer ascos a mancharse de grasa: elija la opción que elija para poner en práctica el contenido del blog tendrá que conectar algo al raspberry -ver figura anterior- y apretar algún tornillo que otro (terminales). Tiene que tener el arrojo suficiente para conectar cosas entre sí: el raspberry con el conversor A/D, los sensores al conversor, la fuente de alimentación del raspberry (es como la de un teléfono móvil), … nada excesivamente complicado.

Vayamos por partes.

1.2.1.- Electrónica básica. Conversor A/D. Sistema de numeración en base 2.

Conviene que sepa manejar un polímetro (el *tester* de toda la vida); casi vale con que sepa medir tensiones en CC (corriente contínua). Repase la ley de Ohm, que no le vendrá mal (aunque para el blog no se necesite).

Figura 4. Polímetro (tester).

Lo de la conversión A/D es un poco más peliagudo.

Si se trata, como es muy habitual, de capturar valores *deprisa*, el asunto de la conversión A/D tiene esquinas por todos los lados. *Deprisa* es miles de veces por segundo -o más; o mucho más-, lo que es necesario, por ejemplo, en un estudio de audio para capturar la música sin que pierda calidad (44.100 veces por segundo); parecidas velocidades se precisan en muchísimos problemas en la industria de todo tipo, en investigación, telecomunicaciones, imagen (en este caso mucho más).

No es el caso de este blog; aquí basta con entender que se trata de convertir valores del mundo real a números con mucha más calma. Si bien habrá que profundizar en cómo se controla el conversor A/D, este es un problema más *admimistrativo* que de ingeniería, como vermos cuando toque.

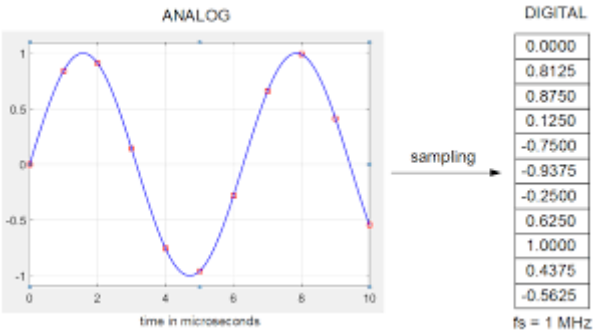

Figura 5. Conversión A/D.

(Como dice alguna página WEB, usted sentado al lado de un termometro de los de columna de mercurio tomando notas cada, digamos, 10 segundos, es un conversor analógico digital: está transformando -y registrando- valores analógicos a números)

El asunto es que el ordenador se maneja con el sistema de numeración en base 2 (los famosos 1010001010 …), que, en este blog, se convierte en el principal problema

*científico (*al menos diferente del de fondo: el análisis en OOP). Si usted ya se maneja con este asunto enhorabuena, pase al siguiente punto; si no es así se tiene que imponer en ello: las páginas de la wikipedia de las entradas “binario” y “bitwise” son un buen comienzo. Tiene que saber, también, lo que es el formato “en complemento a 2”.

Se usa el término *word* -la traducción literal al castellano, “*palabra*” se usa muy poco con el mismo sentido- para designar conjuntos de 16 bits. Aunque el término se puede usar para cualquier longitud -de bits; normalmente potencias de “2”- su uso sin especificar más suele referirse a 16 bits.

Aunque de forma tangencial, tambien se maneja el sistema de numeración hexadecimal; échele un vistazo en la wikipedia.

1.2.2.- Raspberry.

Prácticamente todo lo que necesita saber usted del hardware del raspberry, al menos para empezar, es 1) cómo conectarlo y 2) cómo configurar la µSD. Hay, literalmente, miles de WEBs en las que explican como conectarlo; incluso la documentación -en papel- que lo acompaña lo incluye.

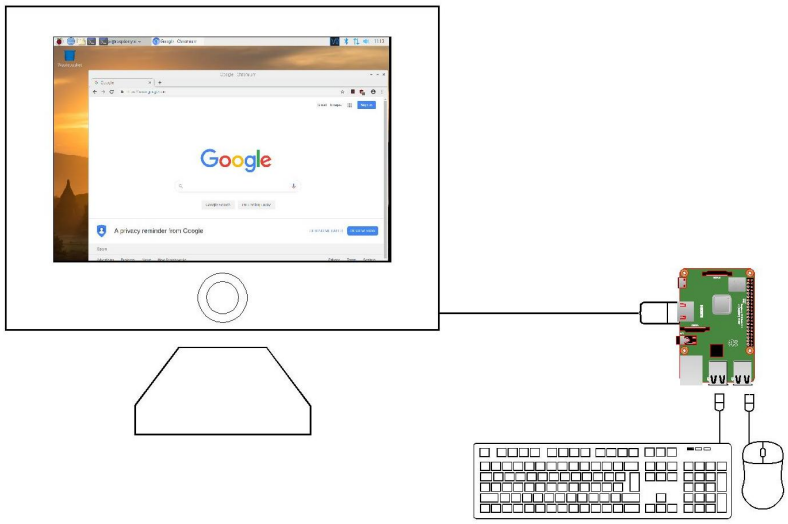

Figura 6. Raspberry conectado como ordenador de sobremesa.

Lo de la µSD tiene un poco más de miga: vaya a ver anexo CC, “Configuración de Raspbian” para más información.

Aunque a medida que vayamos presentando las diferentes opciones se irá explicando con cierto detalle, no viene mal que se vaya usted enterando de lo que es un *bus oneWire*, el *i2c* y la comunicación serie. Vaya echando un vistazo a las entradas de la wikipeidia para “onewire”, “i2c” y “Comunicación serie”: que le vayan sonando por lo menos.

1.2.3.- Linux.

El raspberry tiene la enorme ventaja de funcionar con linux … si usted no conoce linux el problema lo tiene usted.

(Perdón por la impertinencia: es fruto de los muchos disgustos que *el otro* sistema operativo le ha dado a los autores del blog).

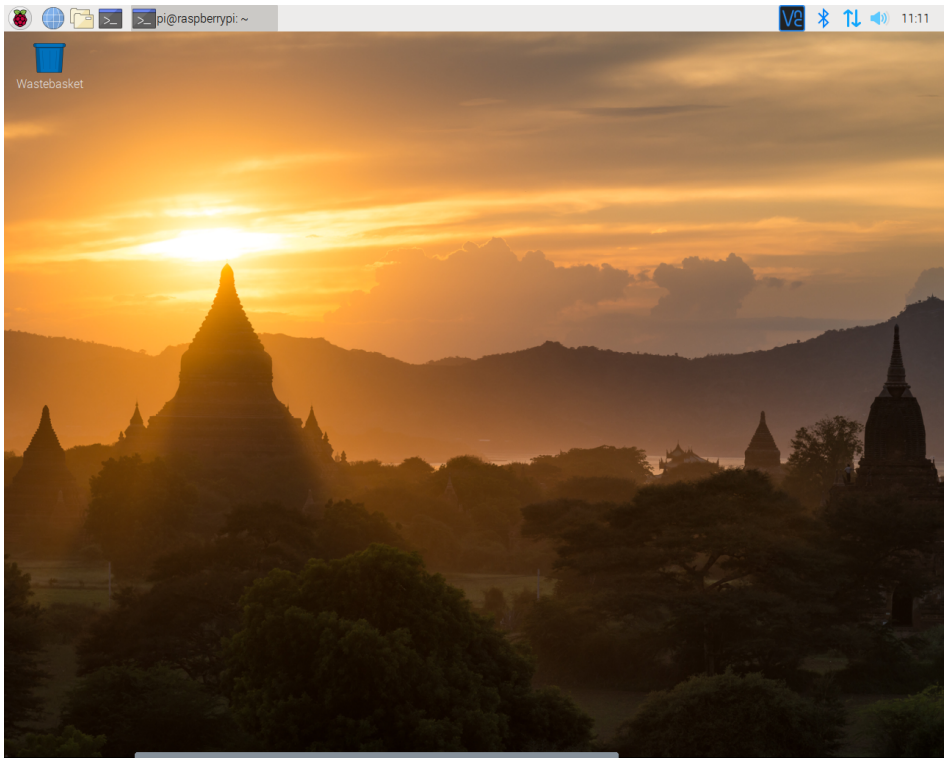

Figura 7. Interface de Linux (Raspbian).

Para seguir el blog no hay que saber mucho de linux: si usted dispone de una *distribución* medianamente bien configurada, los mentados *conocimientos básicos de usuario* deberían ser suficientes para empezar (ver anexo CC, "Configuración de Raspbian").

Como en el caso anterior, si usted ya se maneja con linux enhorabuena, pase al siguiente punto.

En caso contrario, le recomendaríamos que aprenda algo, digamos:

- Le conviene leer algo para, al menos, familiarizarse con los términos (*"Debian", "Jessie", "Raspbian", "Noobs", "Ubuntu"*, …; no se alarme, son, en gran medida, los mismos perros con diferentes collares). Lea el artículo -en castellano- de la wikipedia para "raspbian" (sea paciente: no entenderá muchas cosas al principio, pero hay que ir haciéndose con la música); la página WEB de raspberry ("www.raspberry.org") no es lo mejor del mundo, pero tampoco pasa nada por echarle un vistazo.
- "*Terminal*": el término hace referencia al uso del sistema operativo (linux, en este caso, aunque se puede aplicar a cualquier S.O.) mediante una "linea de comandos". Los más viejos del lugar recordarán la pantalla negra del MS-DOS anterior al windows (y del que -del MS/DOS- los autores del blog tienen la misma

opinión que del windows propiamente dicho, o sea, mala): eso es el “terminal” (y, por cierto, bastantes “comandos” del MS-DOS están copiados de linux; son los mismos).

Es conveniente que se maneje con un mínimo de comandos para configurar cosas del raspberry (de nuevo se remite al lector al anexo CC, “Configuración de Raspbian”). Consulte “https://www.linuxadictos.com/5-comandos-imprescindibles-en-linux”, “https://franciscomoya.gitbooks.io/taller-de-raspberry-pi/content/es/intro/gnu.html” y “recursostic.educacion.es/observatorio/web/ca/software/software-general/295-jose-ignacio-lopez” (*página WEB* que es buena y del ministerio de educación y ciencia … tiene su aquel). Es muy bueno el libro “The Linux Command Line: A Complete Introduction. William E. Shotts Jr”.

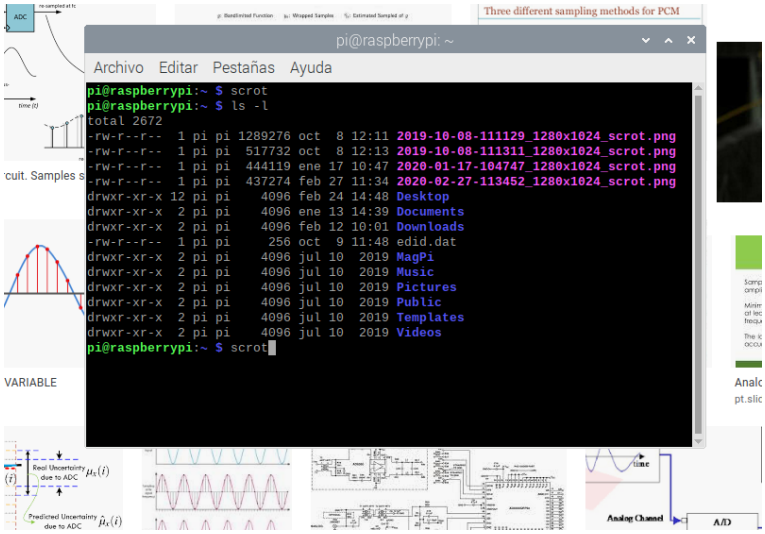

Figura 7. Terminal de Linux (Raspbian).

Copia de seguridad: Una de las ventajas fundamentales de linux, al menos a juicio de los autores del blog, es la posibilidad de recuperarse del desastre de la pérdida del S.O. propiamente dicho. Usando otro sistema operativo la avería del disco duro le deja a usted a la intemperie. En el caso del raspberry el equivalente sería la avería de la uSD (o del propio raspberry, puestos a imaginar desastres); la buena -excelente- noticia es que es posible tener una copia de la uSD simplemente con otra uSD (en la que, eso sí, se tiene usted que tomar la molestía de copiar la buena periodicamente). Si va a *cacharrear* con el raspberry, cosa que le recomendamos, es conveniente que tenga la copia de seguridad actualizada; o, simplemente, dos versiones del S.O., o sea, dos -o más- uSD con diferentes *situaciones* de su trabajo: son unos pocos euros a cambio de una tranquilidad impagable.

1.2.4.- Python. OOP.

Este blog *no* es un *tutorial* de python (ni siquiera de OOP). Para seguirlo con cierta continuidad -sin tener que volver con frecuencia sobre cuestiones más básicas que las que se requieren para comprender lo que se va desarrollando- es necesario tener unos conocimientos *medios* de python. Y tiene que haber escrito media docena de programas sencillos en python.

(El término *“medios”* del párrafo anterior está utilizado con toda la intención -*mala leche*, si se permite la expresión-, porque, ¿quién es capaz de decir qué son unos conocimientos

medios de python?. Uno de los autores del blog lleva cuatro años usando python, es un analista/programador con cierta experiencia, y, a día de hoy, no hay jornada de trabajo de más de cuatro o cinco horas en la que, desarrollando este blog o cualquier otra tarea que requiera del conocimiento previo de python, no descubra algo nuevo relacionado con el uso del lenguaje -o, al menos, de los fundamentos de la OOP).

Python es un lenguaje de los más fáciles (¿el más fácil?) de usar, y, al mismo tiempo, extraordinariamente potente. Y no se trata solo de que su sintaxis permita *goyerías* que resuman en una instrucción de 30 o 40 caracteres lo que con otros lenguajes requiere de varias líneas: python tiene implementadas como básicas todas las instrucciones y arquitectura de cualquier lenguaje moderno, y eso sin contar lo que añaden las librerías que incluye la descarga normal. Además incorpora estructuras -los *diccionarios*, por ejemplo- y/o innovaciones -los *atributos de clase*, por ejemplo- que en su momento, y todavía actualmente, lo diferencian del entorno.

Es, en resumen, un mundo. Así que a ver quién es el guapo que acota los conocimientos *medios* de python.

Procederemos, entonces, por extensión: tomemos como guía de la propia WEB de python el *tutorial* de referencia ("https://docs.python.org/3/tutorial"). Hay que saber:

- Definir y manejar variables numéricas, sus tipos -"*int*(eger)", "*float*", ..- y formatos -decimal, hexadecimal, binario, ...-. Operaciones aritméticas y lógicas -todas-. Entienda bien qué es "*None*".
- Definir y manejar *strings* (aquí sí, lo básico: qué es un string y poco más).
- "Control de flujo". El mentado tutorial de python no es de lo mejor de la *fundación* ("python Software Foundation"): la primera mención que se hace a este respecto -control de flujo de un programa- en el punto 3.2 es francamente desafortunada; no se acaba de entender por qué el -primer- ejemplo se implementa con un "*while*" sin ninguna anestesia previa. Para seguir el blog sin sobresaltos pase al punto "4": tiene que entender *todo* lo que contienen los puntos 4.1 a 4.6, *todo*.
- *Listas, tuples y diccionarios*: a fondo. Conviene estar familiarizado con todo el contenido de los puntos 5.1.1, 5.1.2. y 5.5 del tutorial. En la biblioteca se usan -mucho- para iterar: estudie a fondo el contenido del punto 5.6 y repase el 4.2.
- Módulos: es suficiente con entender para que sirve la instrucción "*import*" y lo que se entiende por "módulos estándar" (6.2)
- *Errores y excepciones*. Merecen un comentario aparte: la implementación de una librería que sea digna de tal nombre no puede ponerse a disposición de los posibles usuarios sin que tenga un mínimo de 'protección' frente a los errores que, con toda seguridad, se cometerán al utilizarla (y eso sin contar algunos que, sin duda, irán sibilinamente ocultos en el código de la propia librería). La contrapartida es que el uso de claúsulas de tratamiento de errores introduce una nueva complicación en el 'aspecto' del código, que, al menos en primera lectura, causa el rechazo de no pocos novatos.

  Ha habido, pués, que buscar un compromiso (que, por cierto, ha dado un buen trabajo): se han desarrollado simultáneamente dos versiones de la biblioteca que, no sin cierta sorna, hemos llamado "*versión estudiante*" y "*versión Pro*". Ambas funcionan perfectamente; la primera no tiene control de errores pero, a cambio, es muchísimo más fácil de seguir. Eso sí, una mala llamada a un método de la librería, una instanciación sin alguno de los argumentos requeridos, un literal en lugar de un número y se arriesga usted a los mensajes de error nativos de python (que, por

cierto, esos sí que son para *Pros* -como los de todos los lenguajes de progrmación, por otra parte).

La versión "*Pro*", a cambio, suministra textos de error *amigables* (si es que se puede usar el término: "5 no es un número de canal válido para el ADS1115" ó "el fichero 'NTC.txt' no existe", por ejemplo) amén de la propia gestión del error para tratarlo en la llamada (no se preocupe si no entiende esto, siga leyendo). Las contrapartidas son que 1) El código se vuelve extraordinariamente más complejo (abstruso; y esto es todavía más notable cuando se usa la OOP, ya que, por lo general, los tramos de código son más pequeños y las claúsulas -instrucciones- de gestión de error *se lo comen*) y 2) tiene usted que conocer los secretos del asunto, que no son *peccata minuta*: el capítulo 8 del tutorial de *cabo a rabo*.

- Clases. Objetos: Aquí viene cuando la matan. Ni que decir tiene que los términos clase y objeto no tienen que tener secretos para usted; y mal empezamos, no obstante, si usted frunce el cejo en cuanto aparece el primer "*self*" o sale despavorido cuando ve un "*__init__*" en el código. En el blog ni lo damos todo por sabido (en lo referente a los conceptos de clase, objeto, etcétera) ni podemos hacer pedagogía al respecto con los conceptos básicos de la cuestión (que, en cambio sí se hace -la pedagogía- cuando la cosa se complica: abstracción, encapsulamiento, herencia, polimorfismo, arquitectura). Por otra parte, la calidad didáctica del capítulo 9 del mentado tutorial ("https://docs.python.org/3/tutorial") deja, de nuevo, bastante que desear. Y no es este un asunto fácil (ni la cuestión básica de la técnica de OOP ni la de cómo empezar en ello); a continuación se recomiendan un par de sitios WEB que usted debe estudiar si quiere, como ya advertimos más arriba, seguir el blog sin sobresaltos.
    - https://www.programiz.com/python-programming
    - https://realpython.com

Instalación y configuración de python: Si usted no está familiarizado con linux es posible que la puesta en marcha de python en el raspberry entrañe cierta dificultad. Aunque los autores del blog son partidarios impenitentes de linux -y, por tanto, detractores confesos del "otro" sistema operativo-, sería un exceso sugerir al seguidor del blog que lo estudie a fondo -linux- como paso previo para empezar con el blog. La puesta en marcha de un raspberry -y su uso- no es complicada en absoluto si se dispone de una buena *distribución* (es la jerga para refererirse a la versión del S.O.). Hay multitud de vendedores en Internet que le pueden suministrar una uSD debidamente configurada; tiene un interface gráfico muy similar al de windows(R). Consulte el anexo CC, "Configuración de Raspbian" para más información.

1.2.5.- Otros conocimientos previos necesarios.

- Tipos de ficheros: debe ser usted capaz de identificar y modificar un fichero de texto (*notepad* de windows; *"leafpad"* en linux).
- Ni que decir tiene que tiene que saber lo que es el código ASCII.
- Échele un vistazo a la entrada de wikipedia para "termistor".