# Taller de Programación y Robótica en CMM BML **2023T2 – CL22 - 15 Noviembre 2023**


[R-HW] Usar **Displays #1 – 35'**

Presentación serie Displays

**->** solo texto: **LCD I2C 20x4**

**[R] – Sincronización reloj** PICO W **con NTP – 55'**

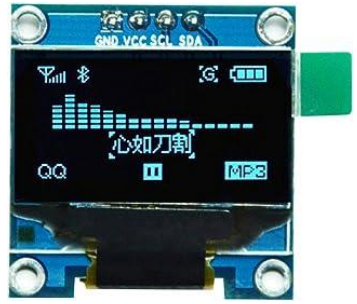
- Recordar problema CL12 temperatura con "timestamp"
- Añadir LCD a temperatura con "timestamp"
- **RTC interno** ¿qué es? y porque esta desincronizado
- NTP: Sincronización del RTC interno con NTC -> Reloj LCD
- Solución de sincronización #1 : Boot_wifi_sincro

Voluntario : J.C. Santamaria

# Clase 22.1.0 – [R-HW] Displays Presentación Serie

## Presentacion Serie Displays

Los displays son la forma más elaborada de mostrar información en robótica. Mas simple y menos versátil que los displays tenemos: LED 1 / 2 / 3 colores, neopixels, 10 bar leds, 7 Segments Display, 8x8 LED Matrix, 4 Digit LED, .. ver tutoriales de MicroPython for Kids para estos 4 últimos tipos (que no hemos visto y solo veremos si queréis).




## En esta serie vamos a ver

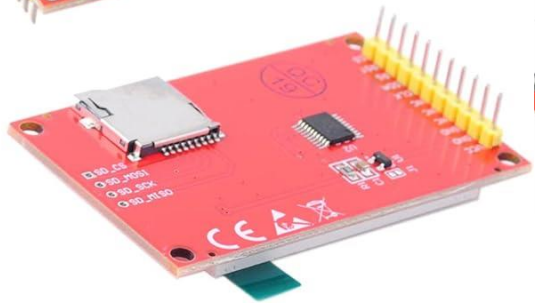
**#1 - LCD 16x2 o 20x4 i2C : solo caracteres**

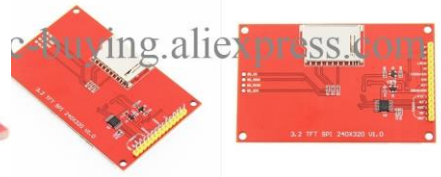
**#2 - SSD1306 I2C - Graphic : 0.96”  128 x64 1 color**

**#3 - TFT ST7735r SPI Graph:  1.8” 128x160 RGB 64K**

**#4- TFT SPI- ST7789VW Graph 1.14” 240 × 135, RGB 64K**

**#5 – TFT SPI ILI9341  Graph 2.8”/3.2· 320 x 240, RGB 64K + touch screen XPT2046 + SD card**

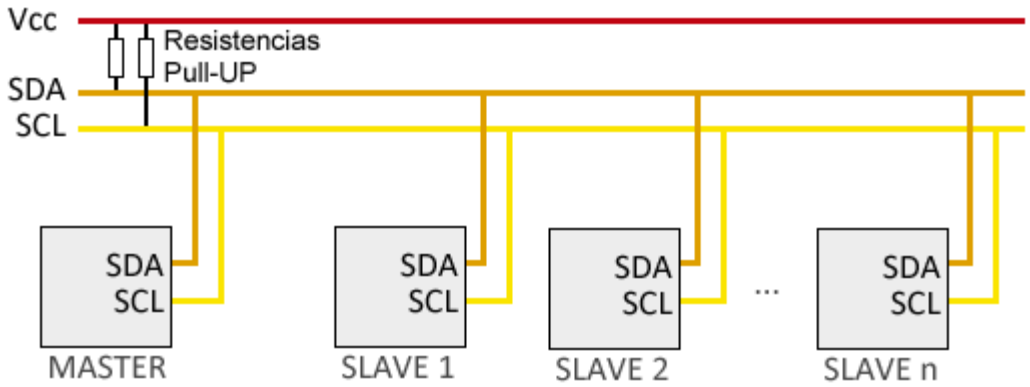


# Clase 222.1.1.1 – [R-HW] Display LCD I2C – Investigación de Librerías

## I2C – Información del protocolo (link)

El bus I2C requiere únicamente **dos cables Reloj-SCL y Datos-SDA**. Funciona en modo **maestro-esclavo sincróno**: el uC es el maestro y los sensores o display los esclavos: el dispositivo **maestro inicia la comunicación** con los esclavos, y puede mandar o recibir datos de los esclavos. Los esclavos no pueden iniciar la comunicación

Cada dispositivo conectado al bus debe tener una **dirección única(7bits),** que, junto a la **frecuencia** del bus, y los **pines SCL y SDA** es lo **único que configuramos en SW**



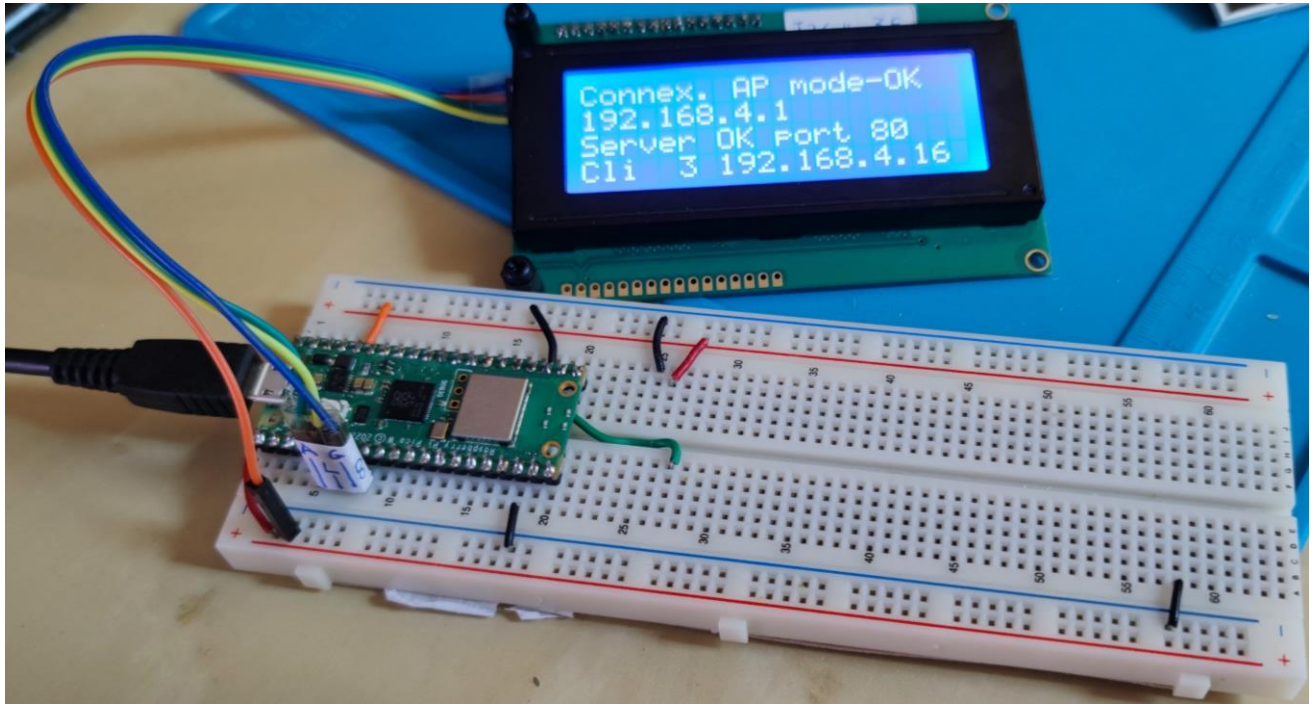

## Librerías de LCD I2c

- No hay una librería standard en micropython
- No usa una copia “pizarra” se escribe directamente al display
- Investigo 3 librerías
  1. **T622** -> Test ok tiene 3 años ==> OK con Pico
  2. **Brainelectronics** : 3 meses muchas funciones y custom char —> **MAL algunas funciones**
  3. **Sunfounder** : pocas funciones
- Selecciono librería T62

https://github.com/T-622/RPI-PICO-I2C-LCD

Va Ok a 400.000Hz, tiene función de crear emojis x8

# Clase 22.1.1.2 – [R-HW] Display LCD I2C – Montaje




**Pines: Se recomienda usar los I2C por defecto**

# I2C0 I2C(0, scl=Pin(5), sda=Pin(4), freq=400_000)

# I2C1 I2C(1, scl=Pin(7), sda=Pin(6), freq=400_000)

Ejemplo

```
i2c = machine.I2C(0, scl=machine.Pin(5), sda=machine.Pin(4), freq=400_000)
```

## Ventajas

- Muy adecuada para solo texto
- Es muy simple de usar y conectar
- Rápida
- Grande y facil de leer

## Contras

- Solo 2 x 16 caracteres o 4 x 20
- Consumo elevado 40 mA
- No se pueden hacer gráficos
- Mala visibilidad al sol

# Clase 22.1.1.3 – [R-HW] Display LCD I2C – Comandos

**Principales**

- **lcd.putstr("Texto aqui") o lcd.putstr(str(variable)) o lcd.putstr(f"Nombre {variable}")**
- **lcd.move_to(x,y) :** x 0 – 19 / y 0 – 3
- **lcd.clear()**

**Control iluminación y encendido / apagado**

- lcd.display_on() / lcd.display_off()
- lcd.show_cursor() / lcd.hide_cursor()
- lcd.blink_cursor_on() / lcd.blink_cursor_off()
- lcd.backlight_on() / lcd.backlight_off()

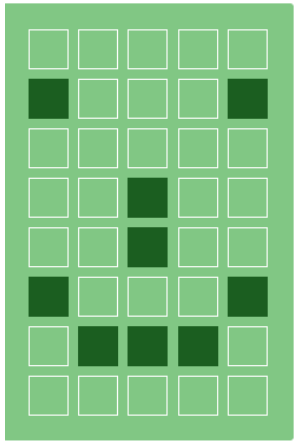
**Caracteres de diseño**

- lcd.custom_char(Num, bytearray([HEX chars])) - Num 0 - 8 HEX chars -> ver Crear Caracteres

## Crear caracteres

- Se pueden crear hasta 8 caracteres "de diseño" con la librería T62. Los caracteres son de 5x8 y se pueden codificar con una "tool" antigua para arduino
- Tool : https://maxpromer.github.io/LCD-Character-Creator/
- Ejemplo CH_SMILE =

```
[0x00,
0x00,
0x11,
0x04,
0x04,
0x11,
0x0E,
0x00]
```



lcd.custom_char(0, CH_SMILE) # guarda el nuevo carácter

Lcd.putstr("cara sonriente " + chr(0)) # uso de un carácter

- Veras que es facil generarlos sin usar la tool, sua binario o hex

# Clase 22.1.1.4 – [R-HW] Display LCD I2C – Basic HW Test

BMMR_bhwt_lcd20x4_I2C.py

- Test super Básico 1
  1. Muestra un mensaje básico si el display está OK y bien conectado
  2. Dispone de unas líneas de codigo para debug de I2c : **# I2C debug block**
  3. Puede también servir para ver cómo usar el display en otro programa copiando trozos de codigo


BMMR bhwt_lcd20x4_I2C_v2.py

- Test Básico v2
  1. Pantalla 1:

     Muestra en pantalla los primeros 80 caracteres dentro de todos los caracteres que puede mostrar la librería usada
  2. Pantalla2:

     Muestra un reloj con un carácter creado: una cara sonriente



# Clase 22.2.1 [R]- Recordar problema CL12 temperatura con “Timestamp”

BMMR_CL22_ADC_temp_file3_0.py

Este programa era un ejemplo de cómo leer datos de un sensor periódicamente y almacenarlos, para su analisis posterior. El sensor es el más simple posible con datos analógicos : Temperatura interna

La forma de guardado es como fichero en el “file system” de la PICO, en modo texto, csv sin cabeceras, que se podrá leer con un Excel.

El tiempo ahora se obtiene del RTC interno de la pico y PICO w con el método **.datetime()**

Nota : es el mismo programa de CL12 solo que con comentarios

```
# F.1 – F.1 - Stores in a file 1 sensor reading + a timestamp
# Usa contexto ( with ) para abrir el fichero que es la forma más adecuada
# 0.1- Crea el objeto sensor de temperatura asociado al ADC interno 4 = CORE_TEMP
# 0.2- Crea el objeto Contador en tiempo real con el RTC interno del Pico,
# Thonny lo inicializa con el reloj del PC , sino inicializado mal a 2021
# 0.3- Crea el objeto Led interno para hacer un breve parpadeo al escribir un registro


# 1- Bucle infinito de lectura y escritura en fichero
    # 1.1 obtiene la temperatura
    # 1.2 obtiene el tiempo en este momento y lo formatea
    # 1.3 Graba la temperatura con una marca de tiempo
    # 1.4 visualiza con un breve flash con la escritura de un registro en fichero
    # 1.5 espera 10 segundos hasta la siguiente lectura
```

# Clase 22.2.2 [R]- Añadir LCD a temperatura con “Timestamp”

BMMR_CL22_ADC_temp_LCD_file_4_0.py

Como **queremos ejecutar el programa SIN estar conectado a un PC vía el IDE Thony,** (por razones que luego veremos) , lo mejor para ver las lecturas y el timestamp, es mostrarlas en un display LDC

Usamos un **LCD de 20x4**:

- 1ra lin : nombre proyecto y version
- 2da lin: fecha
- 3ra y 4ta lin: lecturas de temperatura y hora:mi:seg, se van superponiendo con nuevos datos

**Comentar los TO DO´s, ver cabecera**

```
# 0.1- Crea el objeto sensor de temperatura asociado al ADC interno 4
# 0.2- Crea el objeto Contador en tiempo real con el RTC interno del Pico,
# 0.3- Crea el objeto Led interno para flash al escribir un registro
# 0.4- Crea el objeto LCD, configuraciones por defecto I2C0
# F.1 – F.1 - Stores in a file 1 sensor reading + a timestamp
# 1- (Programa principal) - Presentación en LCD en línea 0
try: # 2- Try - Except :
    # 3- Bucle infinito de lectura y escritura en fichero
        # -3.1 obtine la fecha + la escribe en LCD linea1
        # -3.2 bucle for para scroll de las 2 ultimas líneas LCD  2 y 3
            # -3.2.1 obtine hora en este momento + escribe en LCD
            # -3.2.2 Graba la temperatura con una marca de tiempo
            # -3.2.4 visualiza con un breve flash con escritura en fichero log
            # -3.2.5 espera 10 segundos hasta la siguiente lectura
except KeyboardInterrupt: # 2.2 - si CTRL+C se presiona - > limpiar display
```

# Clase 22.2.3 [R]-RTC interno ¿qué es? y porque esta desincronizado


Un **RTC** (real time clock) es un pequeño circuito diseñado para guardar traza del tiempo. Típicamente tiene un oscilador de cristal, una batería, una pequeña cantidad de memoria no-volatil y la circuitería necesaria para comunicarse.

Un módulo muy popular de RTC es el DS3231 (I2C)

El chip RP2040 de la PICO y la PICO W incorpora un RTC, pero no tiene una batería independiente de la alimentación del uC, por lo que al alimentar la Pico se **inicializará a 1-1-2021,** asi que de alguna forma **hay que re-inicializar** a la fecha y hora correcta si el programa necesita "timestamps" por ejemplo.

El IDE Thony al conectarse por USB a la PICO o PICO W, inicializa correctamente el RTC, pero en situaciones aisladas o con otro IDE , el RTC estará mal inicializado

Documentación de la librería RTC en uPython

```
from machine import RTC
rtc = RTC() rtc.datetime((2017, 8, 23, 2, 12, 48, 0, 0))
# fija una fecha y hora especifica ej. 2017/8/23 12:48 - X
rtc.datetime()
# Obtiene el dato de fecha y hora en este momento según el RTC
```

Para conversiones será necesario usar la librería **time**, que ya hemos usado varias veces.

Documentación de la librería utime/time en uPython

ATENCION : las Tuplas de Fecha y hora son distintas para las librerías RTC y Time

# Clase 22.2.4 [R]- NTP: Sincronización del RTC interno con NTC -> Reloj LCD

BMMR_CL22_lcd20x4_clock_4_0.py

1. Guardar el programa de reloj LCD en la pico W como “main.py”
2. Desconectar la PicoW del PC y
3. Conectar a alimentación por USB

Ver como la fecha/hora inicial es: **2021/01/01 00:00 ->** la **sincronizada**

NPT es un servicio sobre TCP/IP en el port **123,** que envia la hora precisa de relojes atómicos de cesio. En España este servicio NTP lo presta la Armada

El tuto de sunfounder sobre Openweather , explica rápido NTP

Tutorial detallado de la librería NTP en uPython

Codigo de la librería NTP en uPython – port PICO W, Está hecha con sockets, asi que ya podemos echarle un vistazo y entender el codigo

```
# 0- Constates y Variables globales + Objetos
TIMEZONE = 1 # España es GMT +1
ntptime.host = 'ntp.roa.es' # servidor NTP de España, si no usa el host x defecto
# 0.1 LCD, # Usa las configuraciones por defecto I2C0
def setLtime():
"Set RTC date & time synchronized with NTP server. Manages Time zone & DST”
# 1- Mensajes de inicio en LCD y consola
try: # 2- Conexion a internet
    do_connect()
except RuntimeError as et: # debajo codigo para tatar error

# 3- Mensajes de hora inicial
try: # 4- sincroniza RTC con hora GMT
    setLtime()
except:  # debajo codigo para tatar Error NTP
# 5- Mensajes de hora Sincronizada
```

# Clase 22.2.5 [R]- Solución de sincronización #1 : Boot_wifi_sincro

BMMR_CL22_boot_sincro_1_0.py

Ya sabemos que en uPython, el uControlador ejecutara el programa **main.py** , al conectarle a la alimentación.

En realidad, primero busca

**Boot.py**

que ejecutara antes que main.py

Solución de sincronización:

Hacer un boot.py con do_connect, seguido de una sincronización con ntp, y cerrar la conexión para liberar recursos y ahorrar consumo

```
# 0- Constates y Variables globales + Objetos
TIMEZONE = 1 # sPAIN IS gmt +1
ntptime.host = 'ntp.roa.es' # servidor NTP de España,
…. Resto de variables de do_connect
def setLtime():
   """ Set RTC date and time synchronized with NTP server. Manages Time zone and DST"""
 …. Resto de do_connect hasta el final
 # 5 - Sincronizacion con ntp
try:     # sincroniza RTC con hora Local
   setLtime()
except:     #Error NTP
   print("Error NTP")
# 6- Desconectar del router y 7- Desconectar el HW WIFI
wlan.disconnect()
wlan.active(False)
```

# Archivo de transparencias relevantes

Taller personalizado de Programación y Robótica en CMM BML 2023 -2024

# [Redes] Socket Bind : un ordenador en modo servidor puede tener 2 direcciones de red

```
C:\Users\josec>ipconfig

Configuración IP de Windows


Adaptador de Ethernet Ethernet 4:

   Sufijo DNS específico para la conexión. . :
   Vínculo: dirección IPv6 local. . . : fe80::7631:d510:cd2c:cbf3%11
   Dirección IPv4. . . . . . . . . . . . . . : 192.168.1.55
   Máscara de subred . . . . . . . . . . . . : 255.255.255.0
   Puerta de enlace predeterminada . . . . . : 192.168.1.1

Adaptador de LAN inalámbrica Conexión de área local* 1:

   Estado de los medios. . . . . . . . . . . : medios desconectados
   Sufijo DNS específico para la conexión. . :

Adaptador de LAN inalámbrica Conexión de área local* 2:

   Estado de los medios. . . . . . . . . . . : medios desconectados
   Sufijo DNS específico para la conexión. . :

Adaptador de LAN inalámbrica Wi-Fi:

   Sufijo DNS específico para la conexión. . :
   Vínculo: dirección IPv6 local. . . : fe80::aa1f:6c6d:27f6:7f29%5
   Dirección IPv4. . . . . . . . . . . . . . : 192.168.1.58
   Máscara de subred . . . . . . . . . . . . : 255.255.255.0
   Puerta de enlace predeterminada . . . . . : 192.168.1.1

Adaptador de Ethernet Conexión de red Bluetooth:

   Estado de los medios. . . . . . . . . . . : medios desconectados
   Sufijo DNS específico para la conexión. . :

C:\Users\josec>
```

Un PC con cable Ethernet y wifi ambos activados tiene 2 direcciones de red, en el ejemplo tiene

192.168.1.55 -> Ethernet 4

192.168.1.58 -> Wifi

Para escuchar por todas, se configura bind como

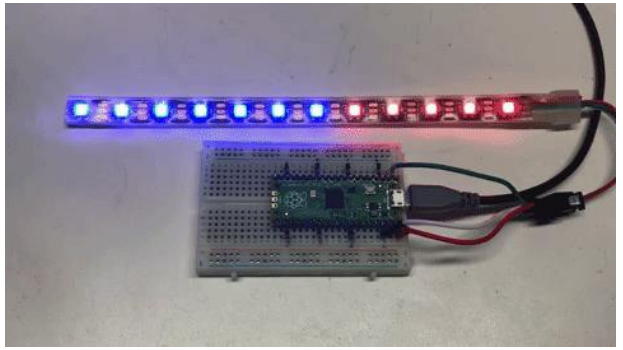
socket.bind((0.0.0.0), port ) or

socket.bind(‘’, port )

usa la dirección especial INADDR_ANY para escuchar en todas las direcciones al mismo tiempo

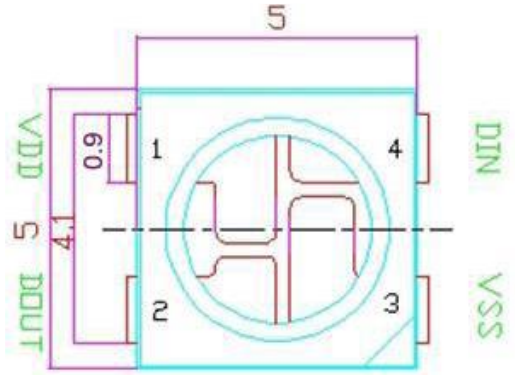
# [Robotica] **Neopixel** ¿Qué son y para que se usan? Seguiremos fuente # **3 - MicroPython for Kids**

https://www.coderdojotc.org/micropython/basics/05-neopixel/



Los **neopixeles** son

- Dispositivos de Salida,
- Tipo led que combinan los 3 colores básicos Red Green Blue, con intensidades programables en cada color = 8bits x3 => 16 millones de colores
- Se pueden conectar “encadenados” y se puede programar el color de cada pieza de la cadena
- Solo requieren 1 hilo para su control + 2 para alimentación
  - El hilo de control “entra” y “sale” de cada neopixel
- El mas común es el compuesto por unidades de **WS2812B**




| NO. | Symbol | Function description |
|---|---|---|
| 1 | VDD | Power supply LED |
| 2 | DOUT | Control data signal output |
| 3 | VSS | Ground |
| 4 | DIN | Control data signal input |

- Usaremos la librería integrada en MicroPython: desde 1.18 hay soporte integrado para NeoPixels para el microcontrolador Raspberry Pi Pico y Pico W (RP2040)
- Alimentación de módulos a 5volt con uControlador a 3.3volt -> Ver Uso Basico
- Cómo mezclar lógicas de 5volt y de 3.3 volt

# [Orden] #1 Que hemos hecho hasta ahora y que sugiero hacer en los próximos meses

| Clases | Fechas | Super Resumen |
|---|---|---|
| 0,1,2,3 | Feb-M | Primeros pasos , 1ros programas |
| 4 | Marzo | Entrega **PicoW**, **Led** |
| 5,6 y 7 | Ma-Ab | **Servos**, POO, **hacer librería Servo** |
| 8,9 | Abril | **Pulsadores**, rebotes, **interrupciones**<br>Prototipado progresivo : cifra cesar<br>Buscar librerías |
| 10-11<br>12 | Mayo | 3 en ralla – bloques juego, estrategias<br>**Neopixel**, volts-niveles lógicos / Trama<br>Bucles **for** / **Funciones / Ficheros->Log**<br>Entradas analógicas **ADC, Sensor Temp** |
| 13+14 | Mayo<br>Junio | **Multímetro**: uso básico<br>**Diccionarios , List comprehension**<br>Sonido **Altavoz** con **transistor NPN** |
| 15+16 | Junio | **Proyecto Completo Termómetro (NTC -> DHT ) con Display LCD I2C** |

## Objetivos sugeridos para 4T2023- 2024

1. 'Internet no es difícil' – haremos IoT
   - Vamos a usar las capacidades Wifi del PICO W para "cacharear" con Internet de las cosas (IoT)
   - En Python del PC haremos programas que lean API´s de internet
2. **Seguiremos profundizando en conceptos de Python para poder hacer mejores programas, tanto en PC como en PICOW**
   - Ejemplo: profundizar en diccionarios y datos JSON (que se usan intensivamente en API´s)
3. Opcional – Avanzar aún más en Python
   - Hacer juegos con interfaz gráfica **PyGame**, con MAKING GAMES WITH PYTHON & PYGAME
   - Hacer **programas prácticos** con AUTOMATE THE BORING STUFF WITH PYTHON

# [Orden y Para casa] #2 Internet para makers

Ejemplo **Hecho** / **Pendiente**

## ¿Qué es internet? Selección Tutos

**How does the Internet work? 1**

**How does the Internet work? 2**

**Guía Buena algo Antigua(ingles)**
**mucho detalle**

## Video

How the internet Works in 5 minutes: A 5-minute by Aaron Titus

¿Cómo funciona Internet? 10 minutos en castellano

## Tipos de proyectos maker – IoT y ejemplos

1. Leer/acceder a recursos a través de API´s
   - Ej. Cheerligth
   - Ej Datos meteo Openweather / AEMET
2. Publicar datos de sensores propios
   1. POST : Thinkspeak
   2. MQTT: Thinkspeak
   3. En modo AP : ej valor temperatura PICO W
3. Avisos de sensores propios a email, Telegram o WA : ej. IFTTT
4. Controlar HW propio
   1. Desde Dentro de TU red local
      - Web server LEDx3 + NTC / Control de relé
   2. Desde Fuera de la red local ( es más complejo)
      1. API´s o MQTT:
      2. APP Movil : ej Anvil
      3. Firebase (google) …..(abrir puertos en router NO RECOMENDABLE)