



# Entorno de programación educativo en lenguaje Python para la EDU-CIAA-NXP

- **Autor:** Ing. Ernesto Gigliotti (UTN-FRA)
- **Director:** Esp. Ing. Eric Pernia (UNQ,UBA)
- **Jurados:**
  - Dr. Ing. Pablo Gomez (UBA)
  - Ing. Gerardo Sager (UNLP)
  - Esp. Ing. Pablo Ridolfi (UTN-FRBA/FRH,UBA)



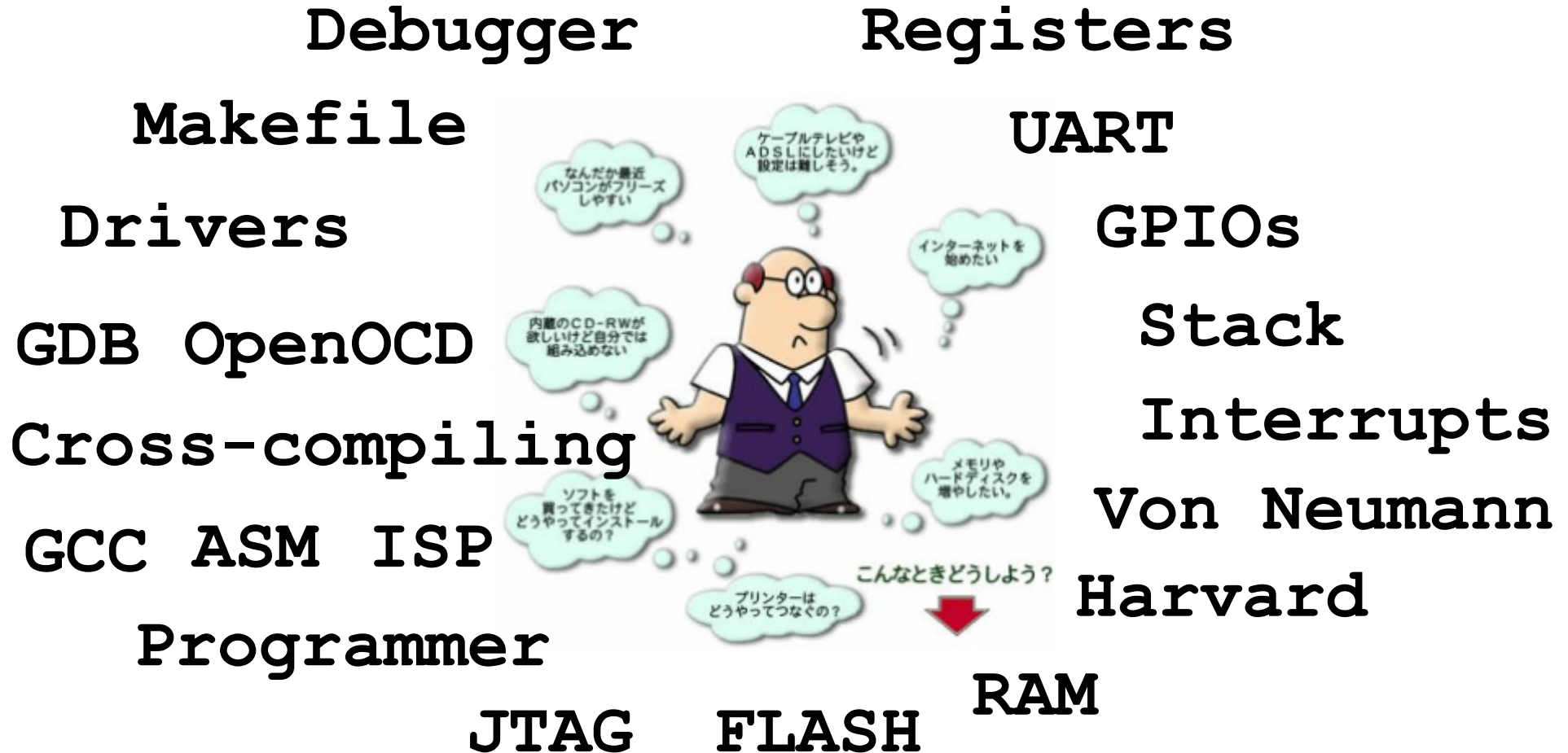
# 1. INTRODUCCIÓN

# **Dificultades en la enseñanza de programación**

- **Lenguaje elegido**
- **Sintaxis**
- **Tipos de datos**
- **Sentencias condicionales**
- **Bucles**
- **Referencias, punteros**
- **IDE**

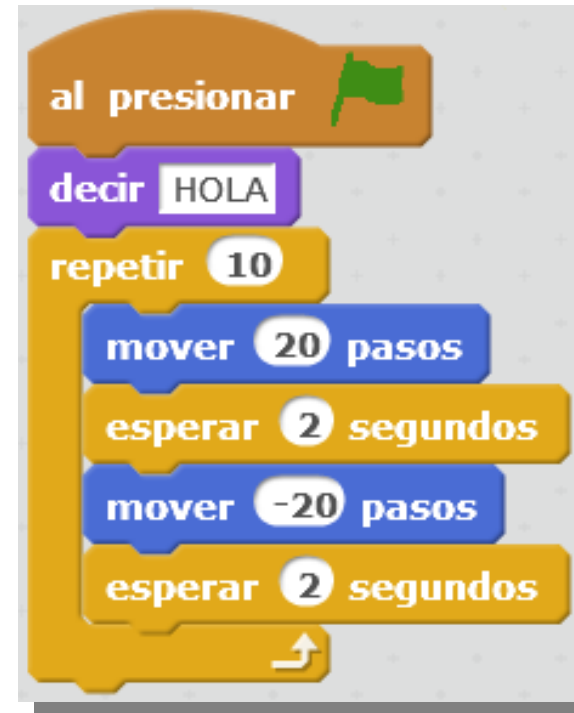
# **Dificultades en la enseñanza de sistemas embebidos**

# Dificultades en la enseñanza de sistemas embebidos



# Herramientas

# Herramientas



Scratch

# Herramientas

LEGO  
WeDo



MakeBlock



Scratch



## Herramientas

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by makin
  delay(1000);                      // wait for a second
}
```



C/C++ simplificado

LEGO  
WeDo



Scratch

MakeBlock

# **Plataforma educativa propuesta en este trabajo**

# Lenguaje: Python

- Sintaxis simple y clara
- Ideal como primer lenguaje
- Adoptado por muchas universidades



# Lenguaje: Python

- Sintaxis simple y clara
- Ideal como primer lenguaje
- Adoptado por muchas universidades



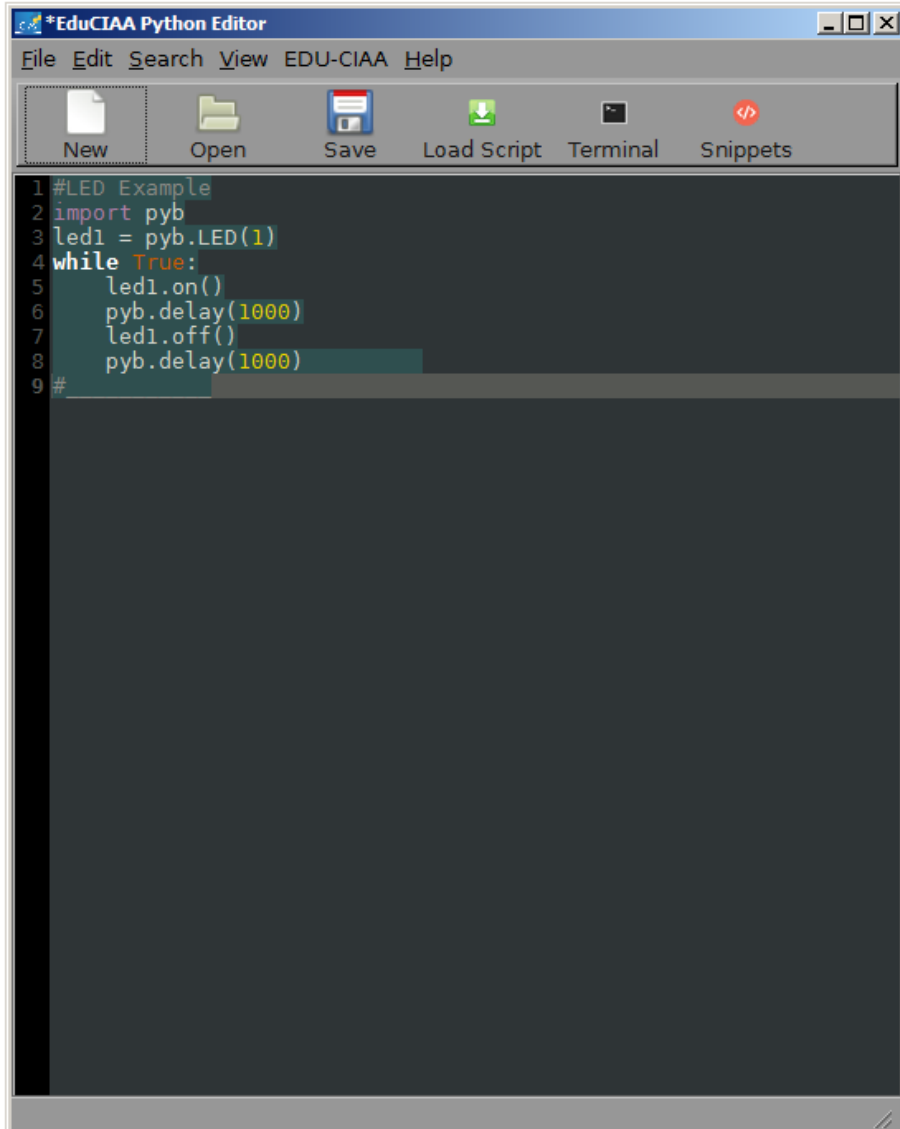
```
while True:  
    led.on()  
    pyb.delay(500)  
    led.off()  
    pyb.delay(500)
```

# Hardware: EDU-CIAA-NXP

- Bajo costo
- Ideal como primer hardware
- Adoptado por muchas universidades
- Comunidad Proyecto CIAA



## Entorno de desarrollo



- Fácil de instalar
- Fácil de configurar
- Graba en la placa el código Python
- Snippets de código
- Terminal integrada

# Documentación



- Documentación de bibliotecas
- Ejemplos

# Plataforma educativa



**Placa y  
firmware**

+



**IDE**

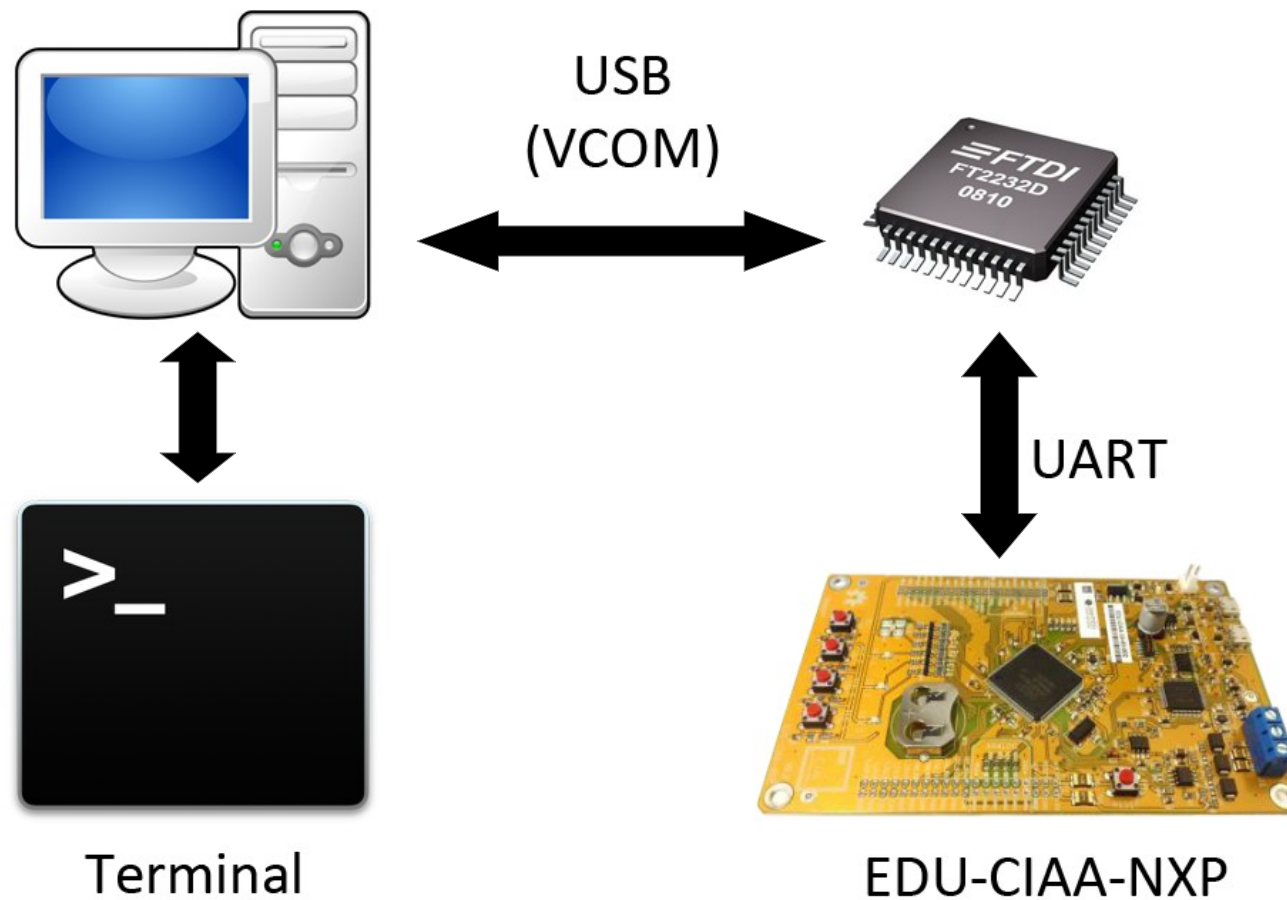
+



**Ejemplos**



# Conexión



## 2. DEMO

# Punto de partida

- Port de Micropython<sup>[1]</sup> para la EDU-CIAA-NXP:
  - Intérprete.
  - Garbage Collector.
  - Filesystem FAT12.
  - Sin soporte de periféricos.

# Punto de partida

- Proyecto EDILE:
  - Open Source.
  - Procesador de texto.
  - Syntax highlight.
  - Python.

# Requerimientos

- Manejo de hardware desde Python:
  - Leds que dispone la placa.
  - Pulsadores.
  - GPIO.
  - UART.
  - Interface RS485.
  - Entradas ADC.
  - Salida DAC.
  - La EEPROM interna.
  - Timers.

# Requerimientos

- Entorno de desarrollo:
  - Multiplataforma.
  - Instalación simple.
  - No cambiar firmware de la placa.
  - Comunicación por USB.
  - Terminal serie.
  - Snippets.
  - Syntax highlight.
  - 1 archivo con script de python.

# Requerimientos

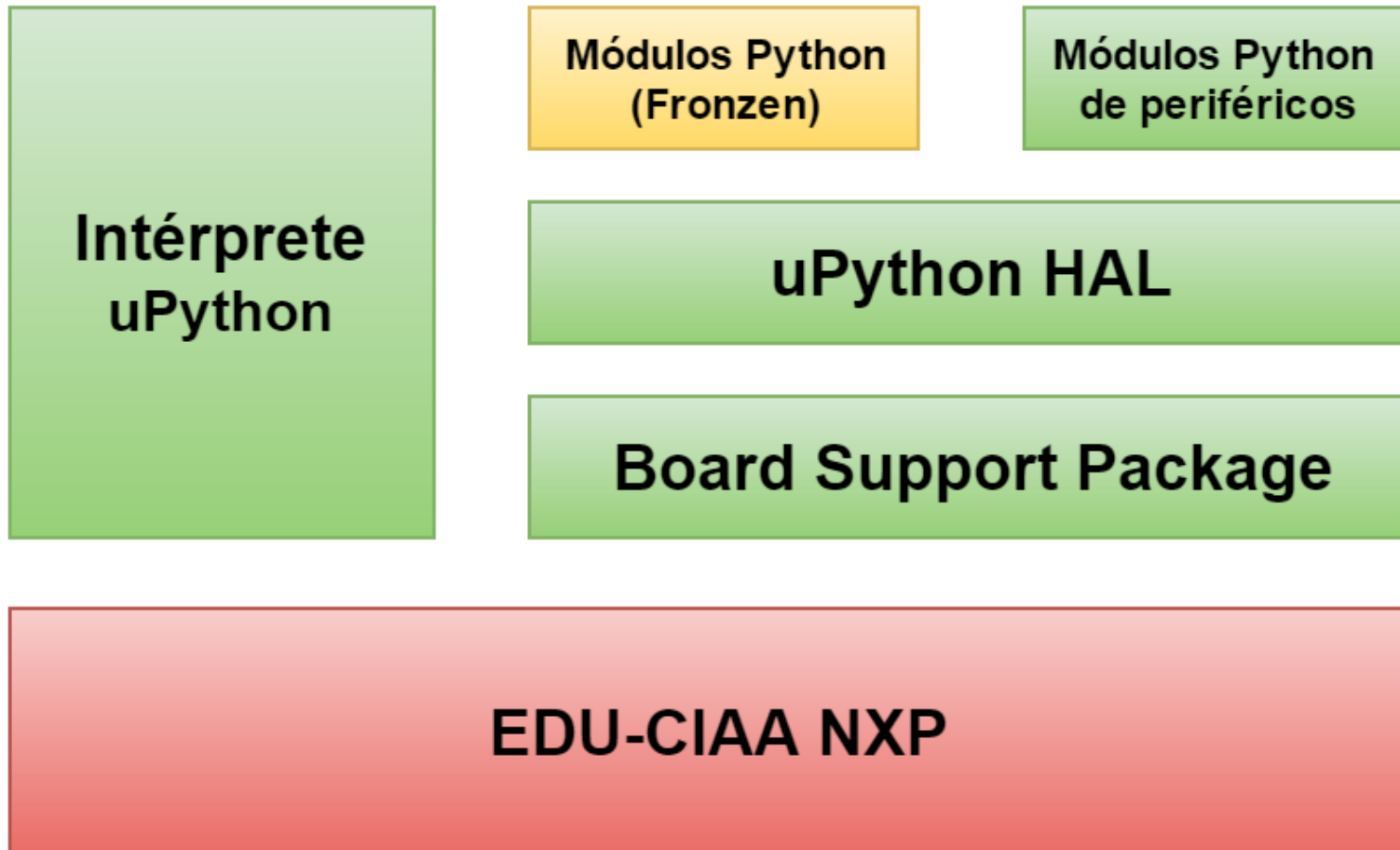
- **Proyectos de ejemplo:**
  - **Inicial.**
  - **Intermedio.**
  - **Avanzado.**
- **Explicaciones detalladas.**
- **Documentación de las bibliotecas.**

# **3. DISEÑO E IMPLEMENTACIÓN**



# Arquitectura Firmware

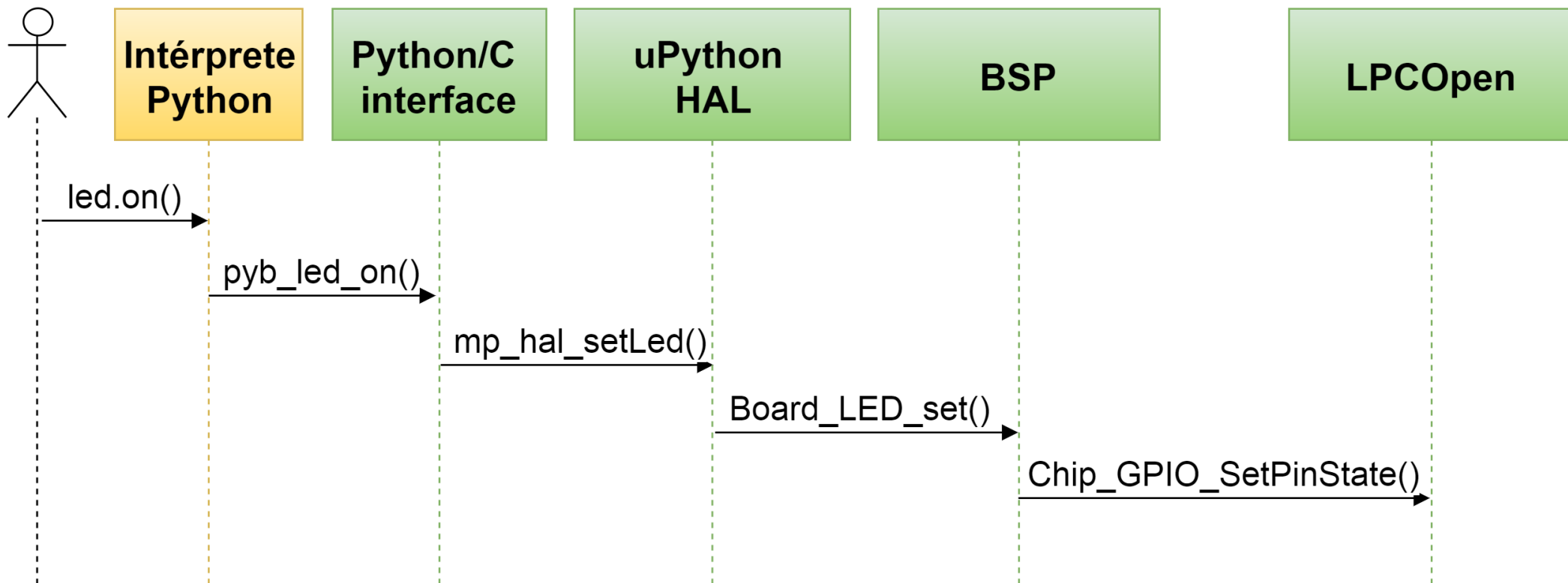
Script Python programado por el usuario

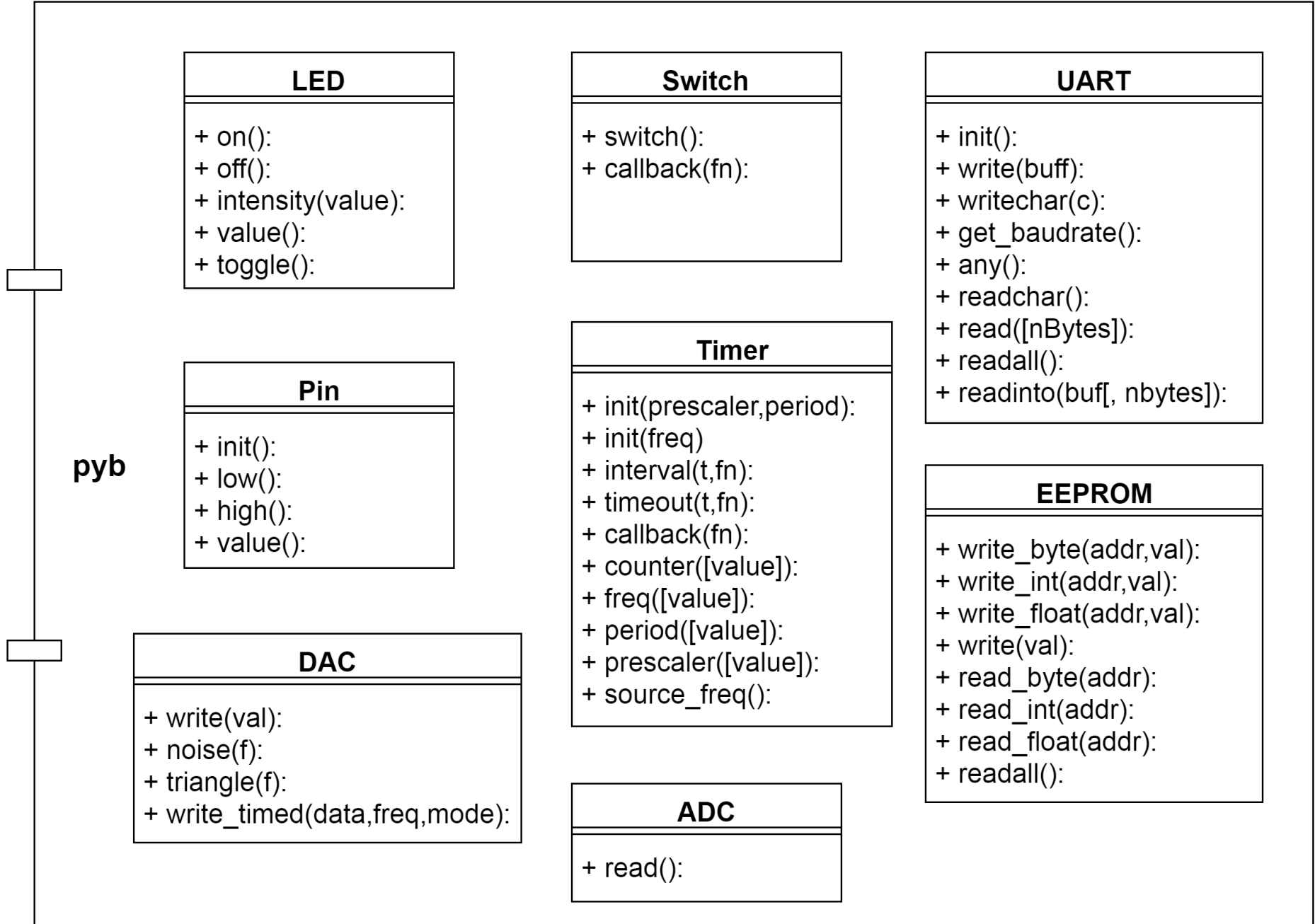


```
import pyb
```

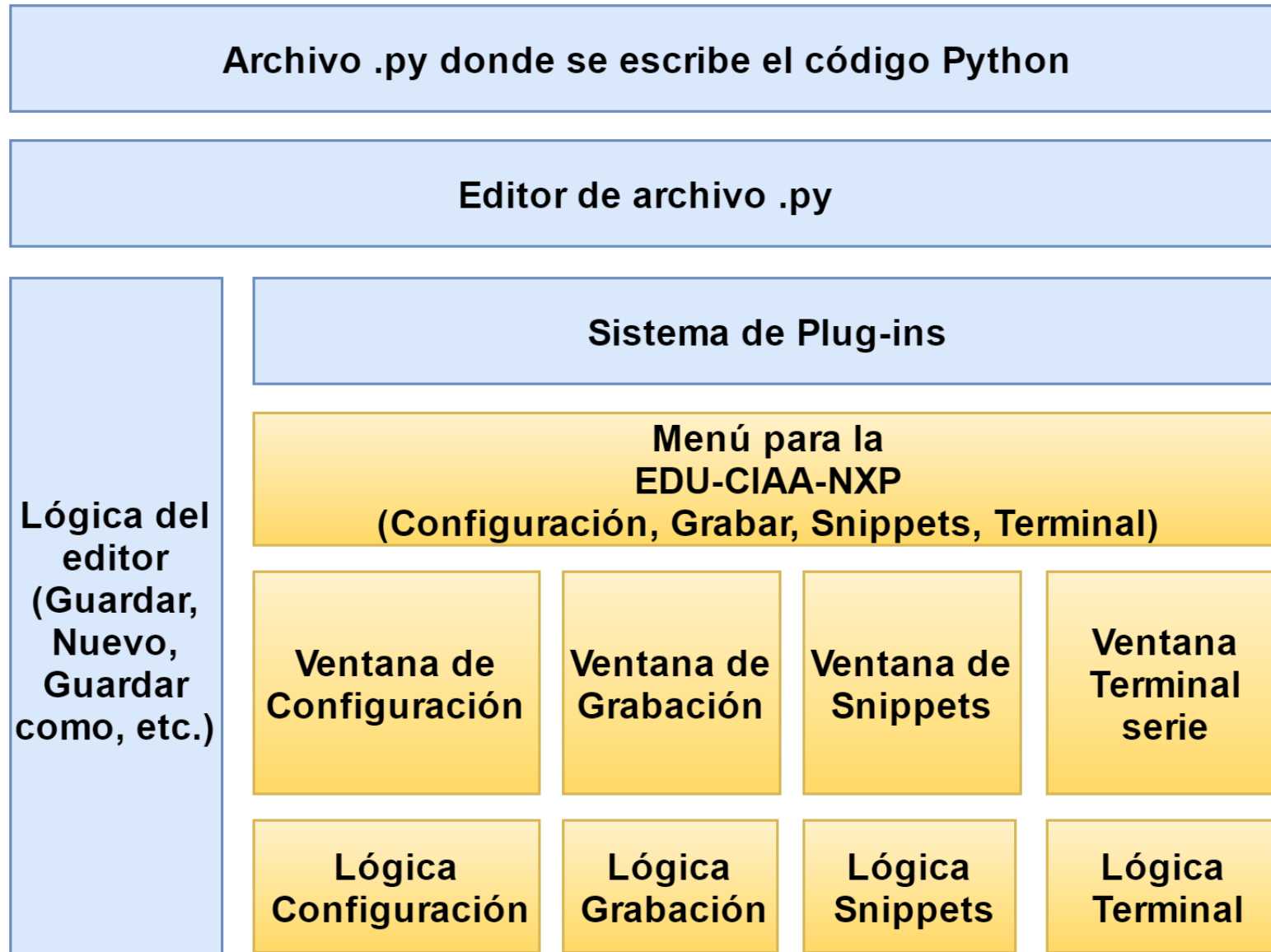
```
led = pyb.LED(1)
```

```
led.on()
```





# Arquitectura IDE



## **4. ENSAYOS Y RESULTADOS**

# Tests Unitarios uPython HAL

mainTest.c

testsLeds.c

testsDAC.c

testsTimers.c

testsSwitches.c

testsADC.c

tests485.c

testsUart.c

testsGPIO.c

testsEEPROM.c

utest.c

utest.h

# Tests Unitarios uPython HAL

mainTest.c

testsLeds.c

testsDAC.c

testsTimers.c

testsSwitches.c

testsADC.c

tests485.c

testsUart.c

testsGPIO.c

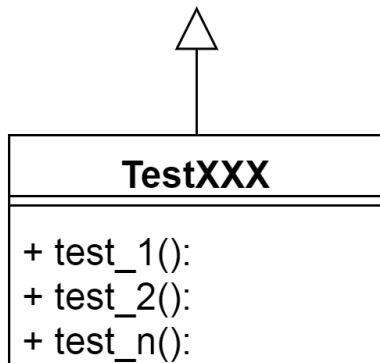
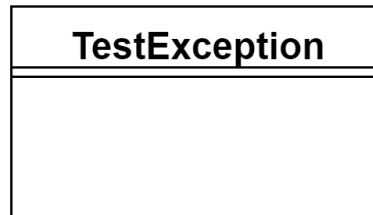
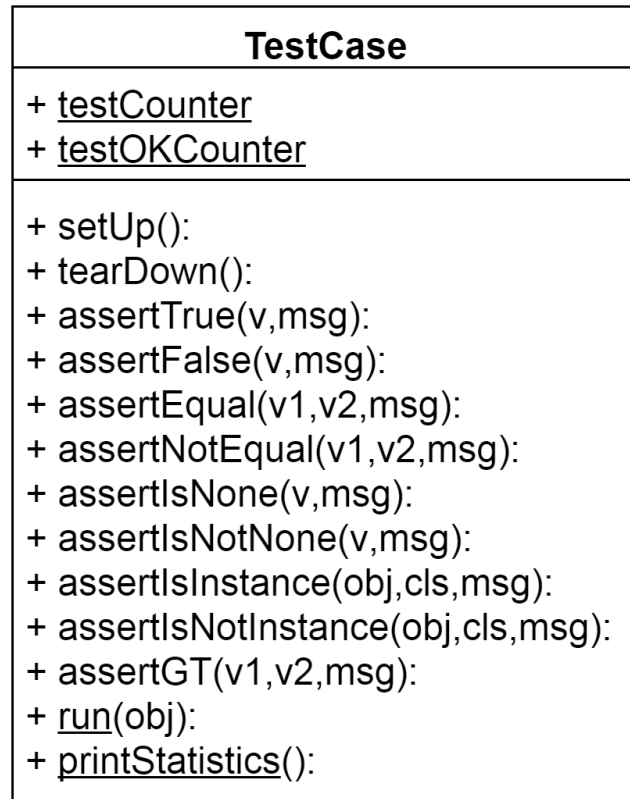
testsEEPROM.c

utest.c

utest.h

58

# Tests Unitarios clases Python



- **TestLeds**
- **TestSwitches**
- **TestUart**
- **TestEEPROM**
- **TestDAC**
- **TestADC**
- **TestGPIO**
- **TestRS485**
- **TestTimers**



# Tests Unitarios clases Python

| TestCase  |
|---|
| + <u>testCounter</u><br>+ <u>testOKCounter</u>  |
| + setUp():<br>+ tearDown():<br>+ assertTrue(v,msg):<br>+ assertFalse(v,msg):<br>+ assertEquals(v1,v2,msg):<br>+ assertNotEqual(v1,v2,msg):<br>+ assertIsNone(v,msg):<br>+ assertIsNotNone(v,msg):<br>+ assertIsInstance(obj,cls,msg):<br>+ assertIsNotInstance(obj,cls,msg):<br>+ assertGT(v1,v2,msg):<br>+ <u>run</u> (obj):<br>+ <u>printStatistics</u> (): |

| TestException |
|---------------|
|               |

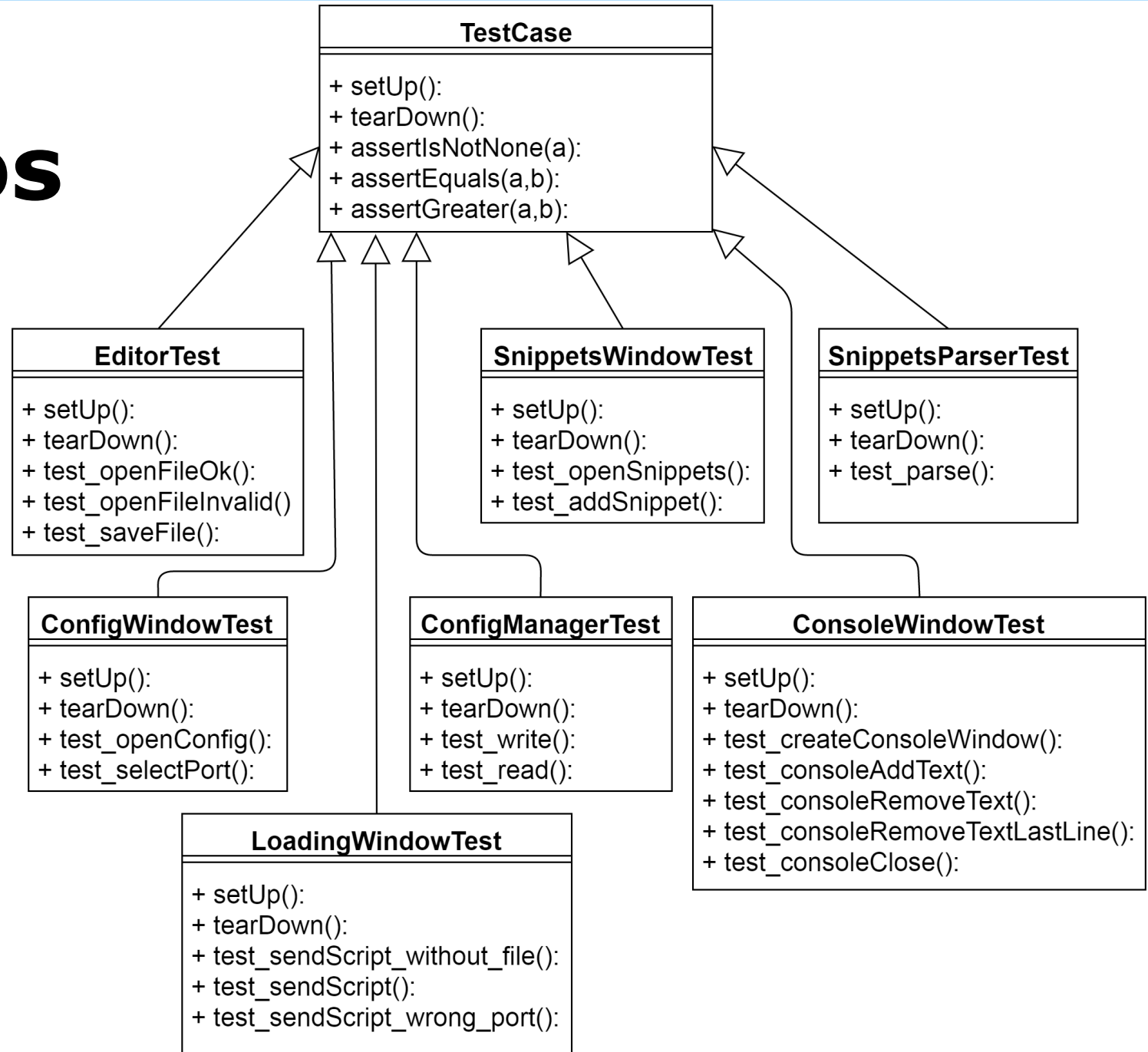
69

| TestXXX                                   |
|---|
| + test_1():<br>+ test_2():<br>+ test_n(): |



- TestLeds
- TestSwitches
- TestUart
- TestEEPROM
- TestDAC
- TestADC
- TestGPIO
- TestRS485
- TestTimers

# Tests Unitarios IDE



# Tests funcionales

# Tests funcionales

- Clases Python

# Tests funcionales

- Clases Python
- Uso del IDE

# Tests funcionales

- Clases Python
- Uso del IDE
- Matriz trazabilidad

## 5. CONCLUSIONES

# CONCLUSIONES

- Dictado de una clase de micropython
- Dictado de una clase de P00
- Programación de microprocesadores
- Ingeniería de software en sistemas embebidos
- Gestión de proyectos
- Taller de trabajo final



# **Pasos a seguir**

# Pasos a seguir

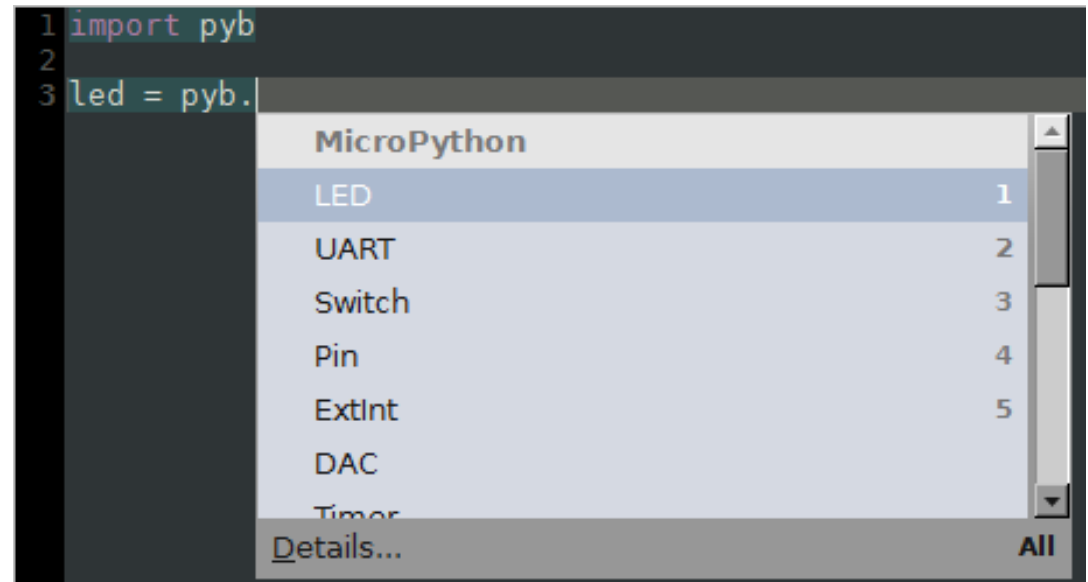
- Interrupciones
- PWM
- Keyboard y LCD
- SPI
- I2C
- RTC

# Pasos a seguir

- Interrupciones
- PWM
- Keyboard y LCD
- SPI
- I2C
- RTC
  
- Modbus
- time
- Core M0
- CAN
- Ethernet

## Pasos a seguir

- Interrupciones
- PWM
- Keyboard y LCD
- SPI
- I2C
- RTC

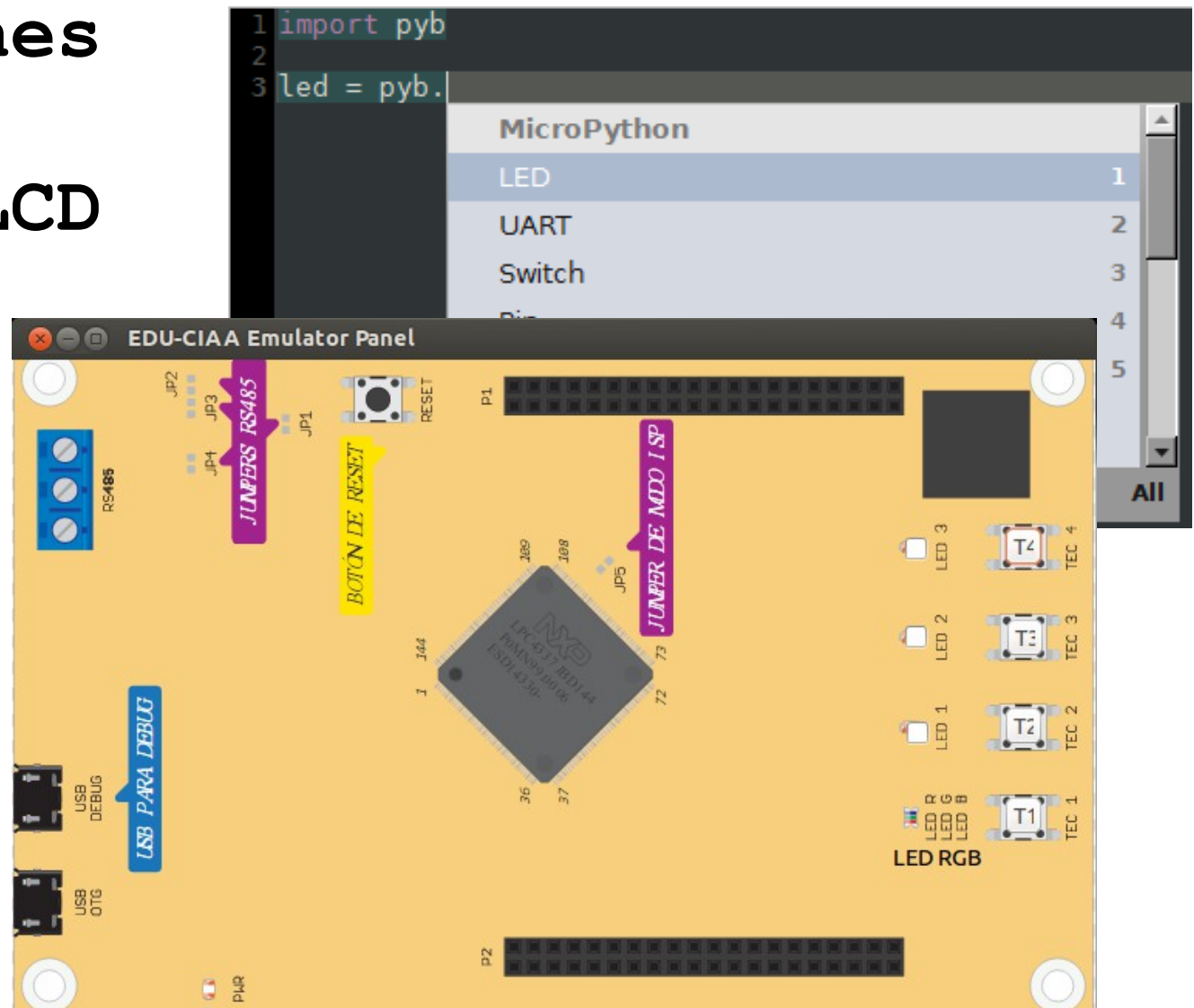


- Modbus
- time
- Core M0
- CAN
- Ethernet

## Pasos a seguir

- Interrupciones
- PWM
- Keyboard y LCD
- SPI
- I2C
- RTC

- Modbus
- time
- Core M0
- CAN
- Ethernet



# PREGUNTAS