Lenguajes de Alto Nivel en Microcontroladores

Taller de Nuevas Tecnologías (IF017)

Nahuel Defossé

Microcontroladores

- Dispositivos compuestos por CPU, RAM, ROM e interfase de comunicaciones orientadas al control directo o comunicación con transductores/actuadores.
- GPIO (General Purpose Input Output), los pines pueden cambiar de función según los requerimientos
- No son lo mismo que un *single board computer* (SBC), como Raspberry Pi o alternativas
- Generalmente gestionados por una comunicación serial, que puede proveer desde un simple puerto serial, hasta un In Circuit Debugger (ICD).

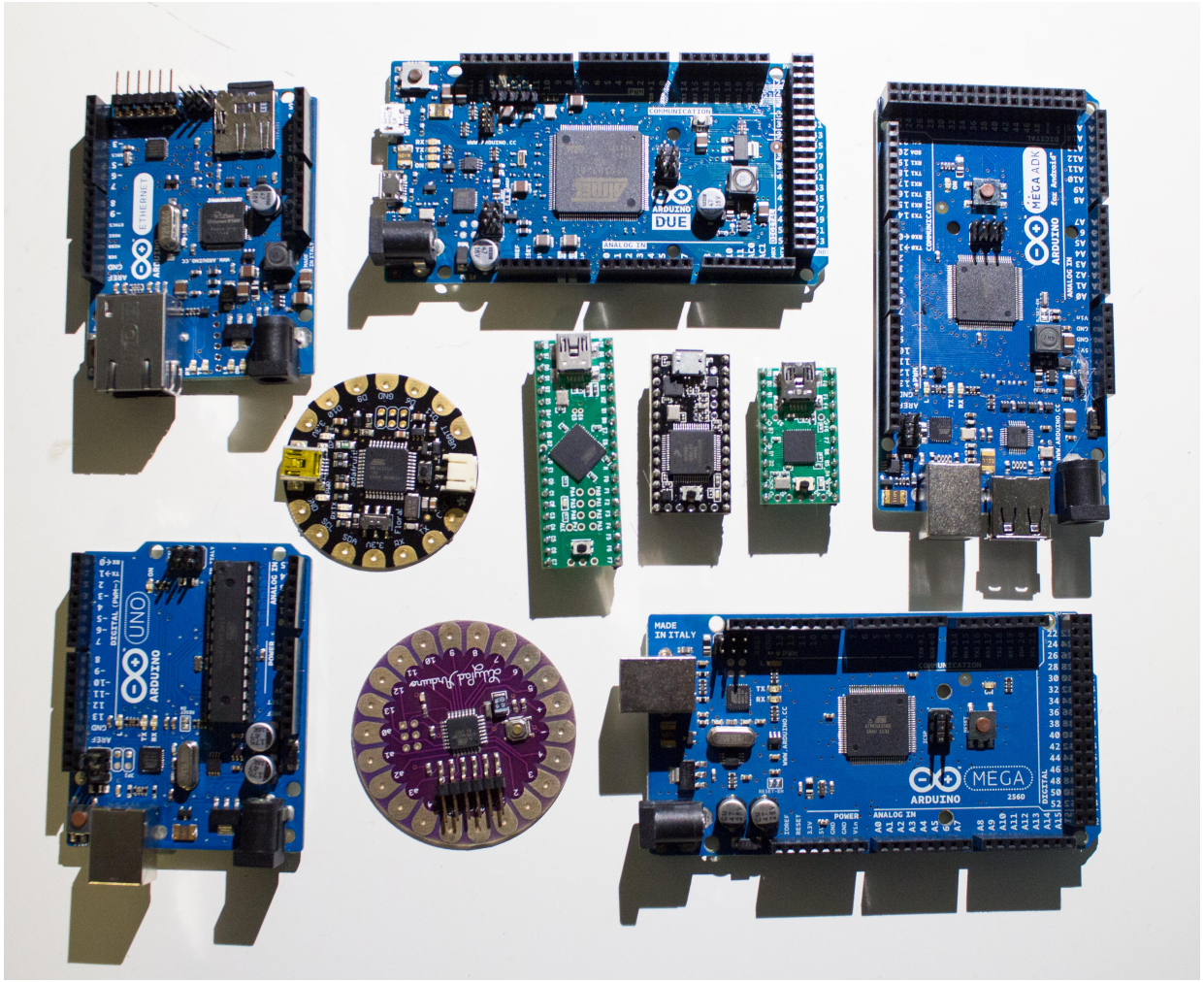

Figura1: Microcontroladores de Familia Arduino

Características

- Frecuencias de trabajo más bajas que un CPU convencional (desde kHz hasta 180MHz)
- Menor consumo energético.
- RAM inferior al megabyte.
- Almacenamiento Flash del orden de los KiB a pocos MiB.
- Bus de datos de 8 bits (Arduino, PIC) o 32 bits (ARM Cortex)

Comunicaciones

- Serial (UART, RS232, RS485)
- I2C (*inter integraded circuit* communcation)
- 1 Wire
- SPI (Serial Peripherical Interfase)
- PWM (*pulse width modulation*)
- SDIO (*secure digital input output*)

Ejemplos de uC

- Atmel [1]
  - Atmega 328p utilizado en Arduino UNO, Nano, etc.
  - SAM D21 Cortex M0 utilizado en
- Microchip
  - PIC18F4550 utilizado en Pinguino
- ESP
  - ESP8266, Sillica Xtensa + Wifi
  - ESP32, evolución *dual core* de 8266 con Bluetooth.
- ARM Cortex [2]
  - LPC 4337 utilizado en Edu CIAA

[1] Atmel es parte de Microchip, pero conserva la marca.

[2] ARM no fabrica los circuitos integrados, sino que licencia la propiedad intelectual a otras compañías para su fabricación.

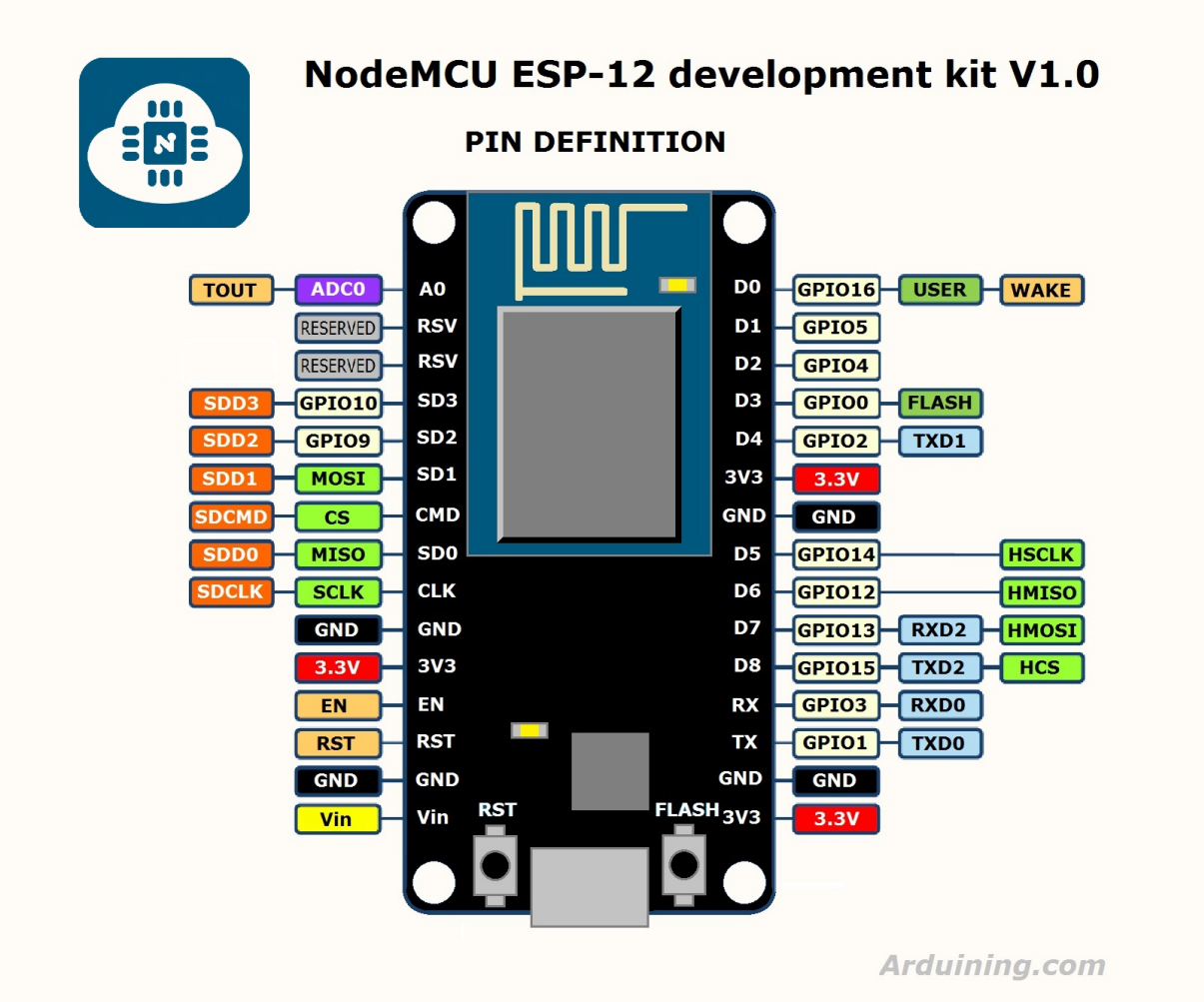

Figura2: NodeMCU (ESP8266)

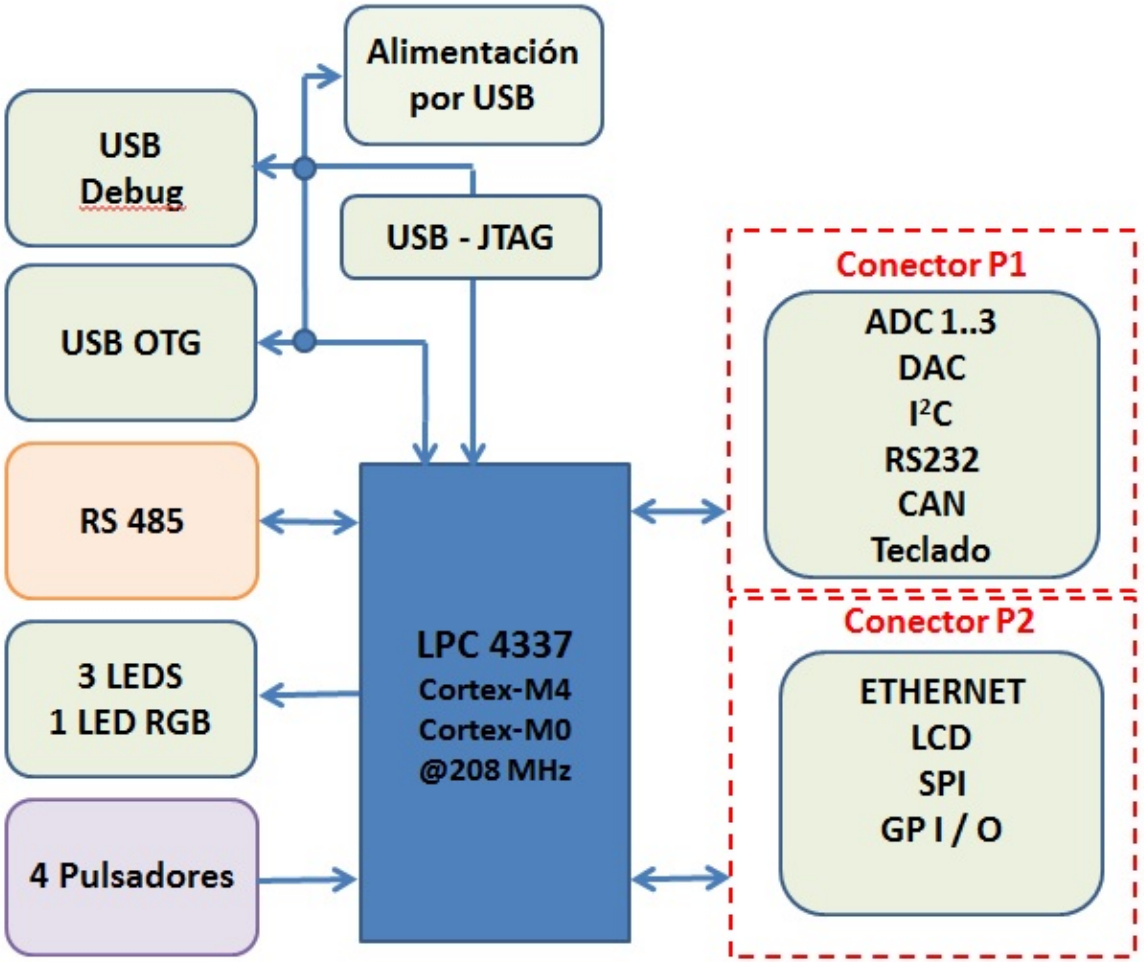

Figura3: EduCIAA NXP

Programación de microcontroladores

Frameworks / (RT)OSs

- La programación en lenguaje ensamblador es compleja, por lo tanto los fabricantes de hardware ofrecen no solo bibliotecas, sino servicios de mayor nivel de abstracción. En ARM por ejemplo, para los microcontroladores Cortex, existe CMSIS
- Algunos Frameworks son incluso de mayor nivel de abstracción como Arduino
- FreeRTOS (Amazon) puede ser considerado también un Framework de programación, aunque no define por ejemplo, como ser realizan las operaciones de I/O.
- Arduino es el más popular, ofreciendo APIs en C++.

Herramientas/GUIs/IDEs

- Muchos *toolchains* están basados en herramientas libres o al menos gratuitas. Arduino utiliza gcc y avrdude por detrás.
- Si bien es posible utilizar *Makefiles*, en algunos casos resulta poco práctico
- Eclipse, NetBeans, Visual Studio, Arduino IDE son algunos ejemplos de IDEs genéricas o específicas.
- PlatformIO es una herramienta de línea de comandos que ofrece:
  - Gestión de plataformas, librerías, proyectos.ß
  - Plugin para VSCode/Atom
  - CI/CD
  - Compilación en la nube
  - Debugging remoto

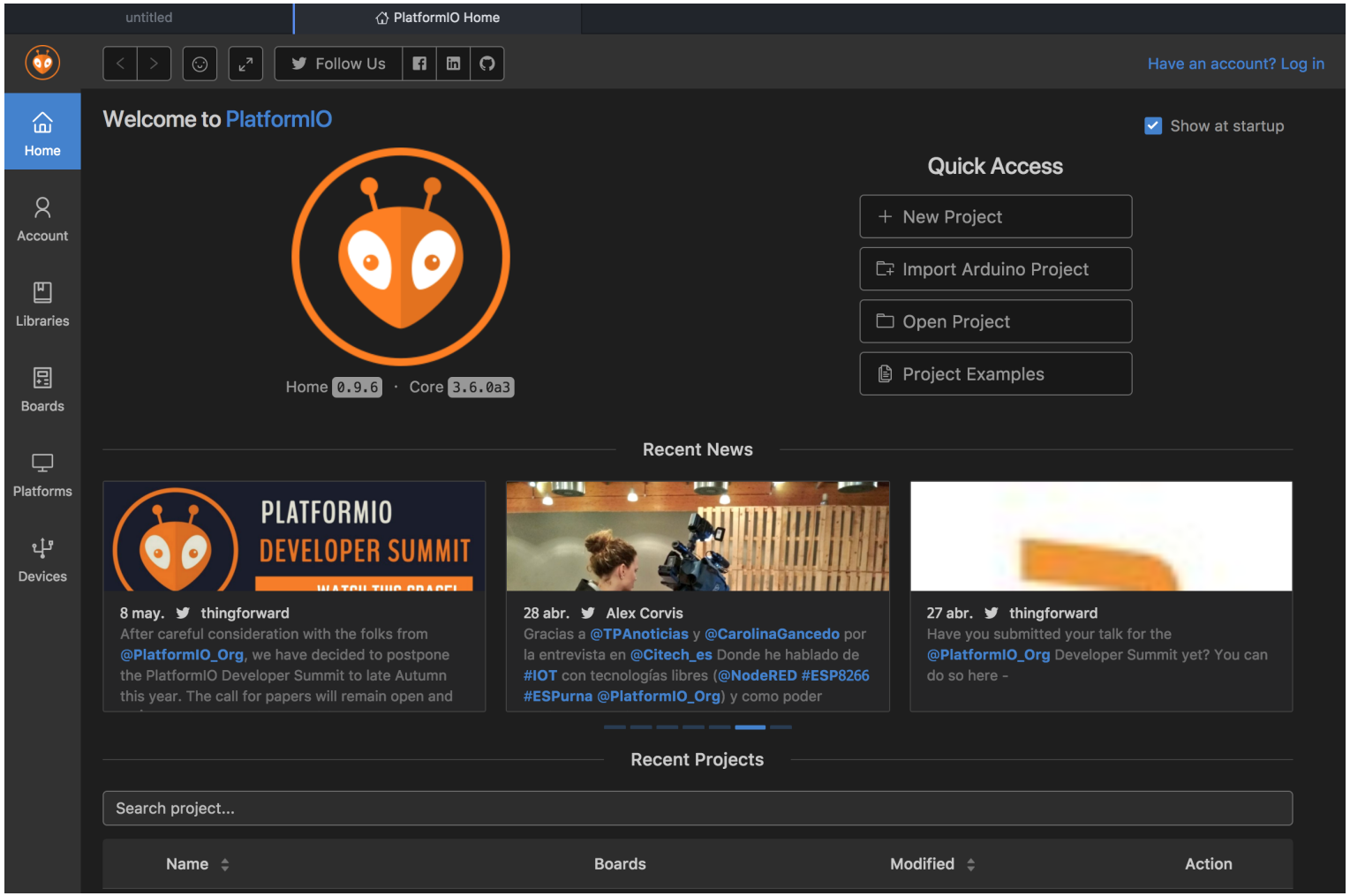

Figura4: Atom + PlatformIO

Lenguajes de lato nivel

- Crecimiento de prestaciones posibilita la escritura de intérpretes para uC.
  - Alta velocidad en MHz
  - 64KiB de RAM
  - 256KiB de Flash
  - Conectividad 802.11
- Aparición de lenguajes interpretados
  - Ciclo `fetch/dispatch`

Ejemplos de lenguajes

- NodeJS
  - Placa Tessel 2
  - Placa Photon
  - Placa Espurino
- eLUA
  - Soporte bastante amplio
- MicroPython
  - STM32, ESP8266/32,
  - EduCIAA
  - Micro:Bit

Ejemplo de integración con MQTT

```
from network import WLAN
from umqtt import MQTTClient
import machine
import time

def settimeout(duration):
    pass

wlan = WLAN(mode=WLAN.STA)
wlan.antenna(WLAN.EXT_ANT)
wlan.connect("red", auth=(WLAN.WPA2, "clave"), timeout=500

while not wlan.isconnected():
    machine.idle()
```

```
print("Conectado to Wifi\n")
client = MQTTClient("joe", "192.168.0.100", port=1883)
client.settimeout = settimeout
client.connect()

while True:
      print("Sending ON")
      client.publish("/lights", "ON")
      time.sleep(1)
      print("Sending OFF")
      client.publish("/lights", "OFF")
      time.sleep(1)
```