Robótica y automatización

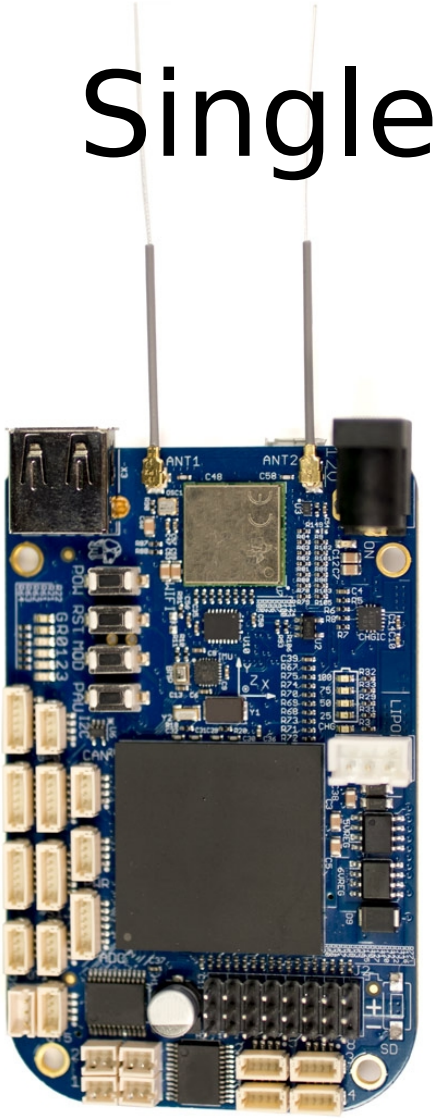
Presentación de la placa BBBl

Facultad de Ingeniería
Instituto de Computación

Contenido

- Generalidades de la placa Beaglebone Blue
- Módulos de hardware
- Pines
- Consideraciones de programación
- Biblioteca rcpy

Single Board Computer



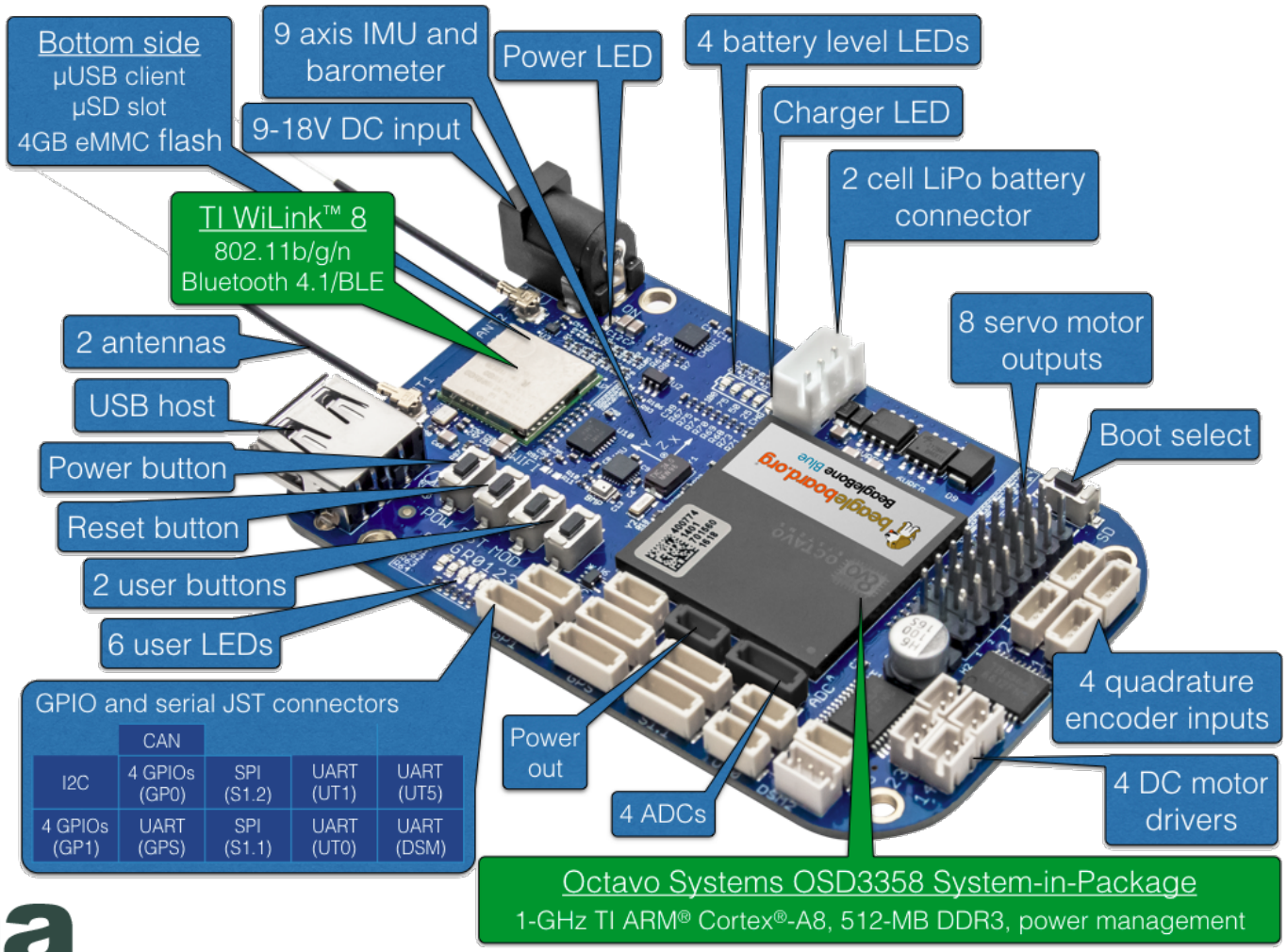
Processor: AM335x 1GHz ARM® Cortex-A8

- 512MB DDR3 RAM
- 4GB 8-bit eMMC on-board flash storage
- NEON floating-point accelerator
- 2x PRU 32-bit microcontrollers

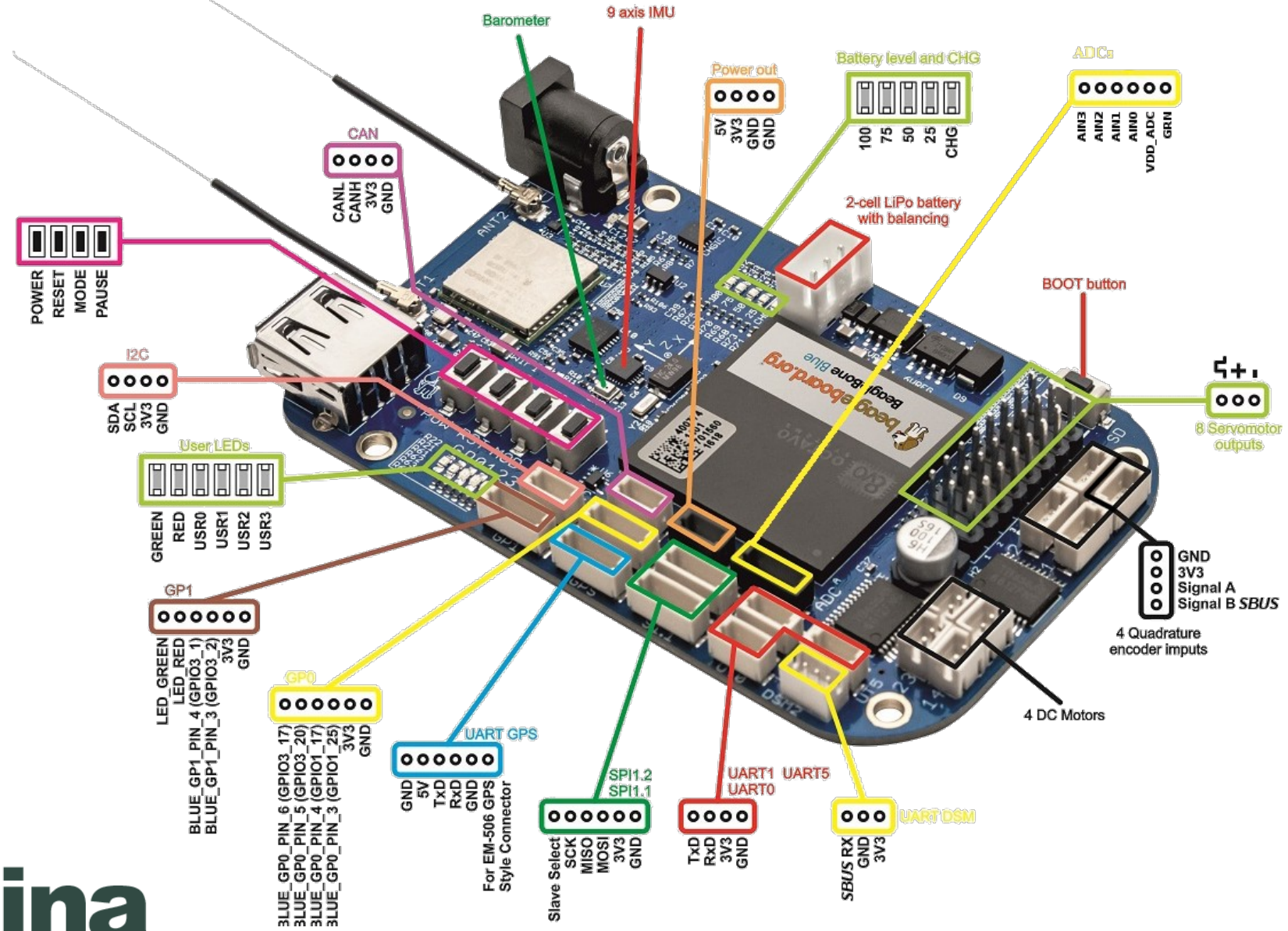
- E/S:
  - USB client for power & communications
  - USB host
  - Battery support
  - WiFi
  - Motor control
  - Sensores

mina
network management | artificial intelligence

Elementos de la SBC



Detalle de los conectores



GPIO

- GPIO general-purpose input/output.
- Pin cuyo comportamiento es controlable dinámicamente por el usuario.
- BBB dispone de 65 GPIO pines
- GPIO maneja niveles lógicos estándares (HIGH y LOW).
- Debe tenerse en cuenta:
  - Voltaje (1.8V)
  - Corriente máxima (4-6 mA).
- Los GPIO pueden ser usados como línea externa de interrupción al CPU.
- Los GPIO pueden agruparse para implementar un bus serial de comunicaciones (I2C, SPI, UART, entre otros).

GPIO

(consideraciones de programación)

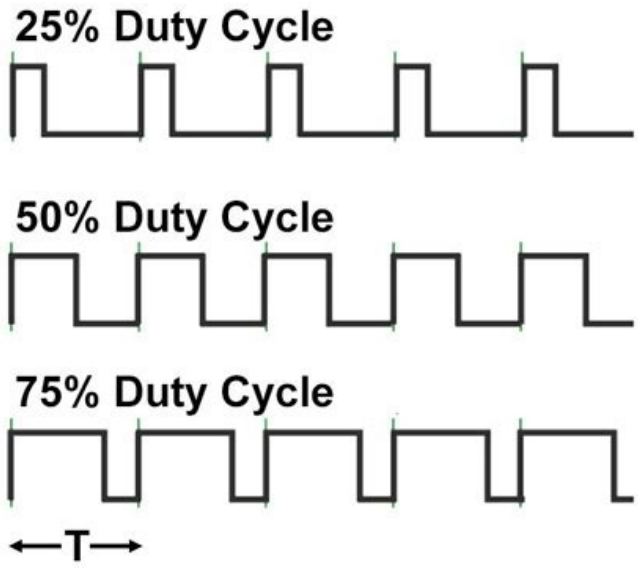
- Los GPIOs pueden:
  - Configurarse como entrada o salida.
  - Pueden habilitarse o deshabilitarse.
  - Pueden leerse.
  - Pueden escribirse (configurado como salida).
- Biblioteca
  - Definir dirección (INPUT o OUTPUT).
  - Leer.
  - Escribir (HIGH y LOW).
- Ejemplo
  - GPIO.setup("URS0", GPIO.OUT)
  - GPIO.output("URS0", GPIO.HIGH)

ADC

- Conversos analógico digital.
- Características en la BBBl
  - 7 canales.
  - 12 bits.
  - Tiempo de muestreo 125ns
- Consideraciones de programación:
  - ADC.setup()
  - analogReading = ADC.read("ADC0")

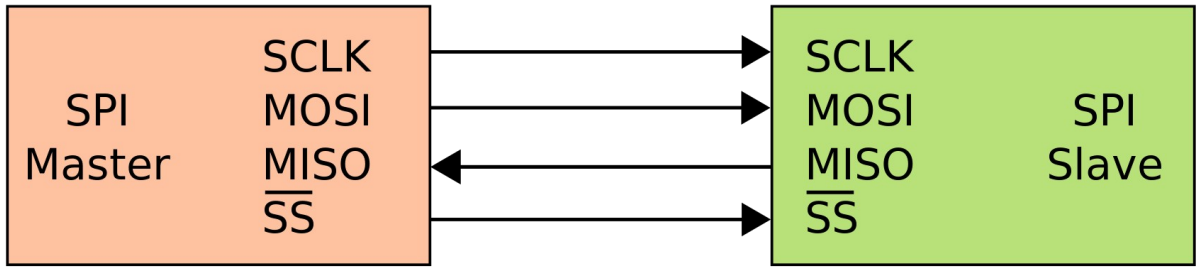
PWM

- Modulación por ancho de pulso.
- 8 canales PWM en la BBBl
- Consideraciones de programación:
  - PWM.start("PWM0")
  - PWM.stop("PWM0")
  - PWM.duty("PWM0", 50)
  - PWM.set_frequency("PWM0", 10)



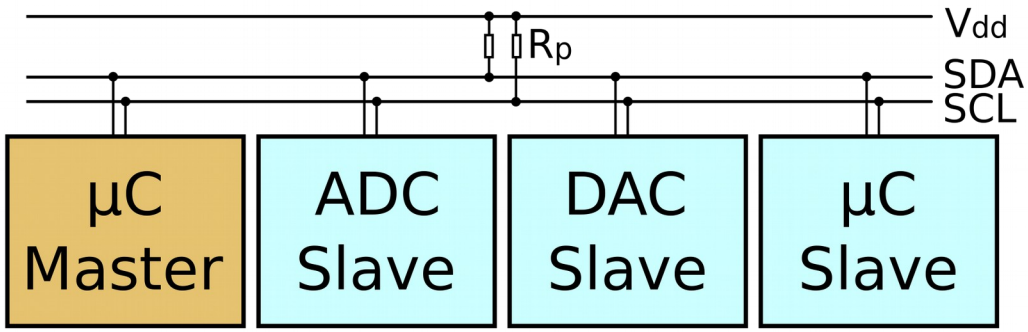
SPI

- Serial Peripheral Interface
- 2 puertos en la BBBl
- Comunicación serial síncrona.
- Maestro – esclavo. Maestro único.
- Full Duplex.
- Interfaz:
  - SCLK: Serial Clock (output from master)
  - MOSI: Master Output Slave Input (datos de salida desde el maestro)
  - MISO: Master Input Slave Output (datos de salida desde el esclavo)
  - SS: Slave Select



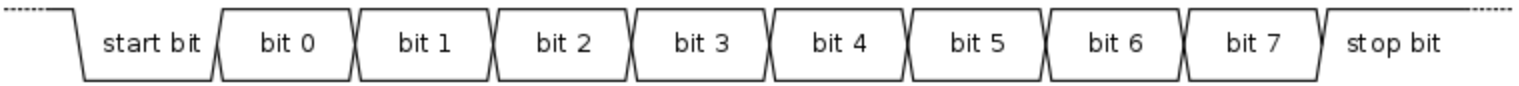
I2C

- Inter-Integrated Circuit.
- 2 puertos en la BBBl
- Comunicación serial síncrona.
- Maestro – esclavo. Soporta múltiples maestros.
- Half Duplex.
- Envio de mensajes con identificador de destinatario.



UART

- Universal asynchronous receiver-transmitter
- 4 puertos en la BBBl
- Comunicación serial asíncrona.
- Full Duplex.
- Consideraciones de programación:
  - UART.setup("UART1")
  - ser = serial.Serial(port = "/dev/ttyO1", baudrate=9600)
  - ser.open()
  - ser.write("Hello World!")
  - ser.close()

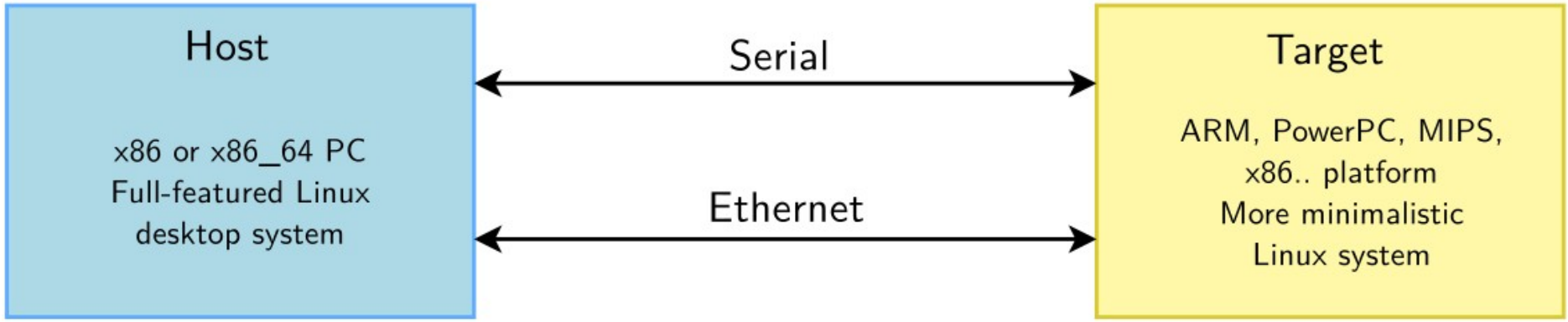


Biblioteca rcpy

- Biblioteca Python para robotic cape.
- Soporte para:
  - IMU
  - Motores DC
  - Servos
  - Encoders
  - LEDs
  - Botones
  - GPIO
  - ADC
- API https://guitar.ucsd.edu/rcpy/html/index.html

Host y target

- Comandos
  - ssh
  - scp
- Navegador de archivos (configurar servidor sftp)



Ejemplo 1

```
green = led.LED(2,3,ON)

while True:
    green.toggle()
    time.sleep(.5)
```

Ejemplo 2

```
from rcpy.button import mode, pause

while True:
    if mode.is_pressed():
        print('<Mode> pressed!')
        break
    time.sleep(1)
```

Ejemplo 3

```
import rcpy.adc as adc

for ch in range(adc.CHANNEL_COUNT):
    raw = adc.get_raw(ch)
    print("channel={} : raw={:4}".format(ch, raw))
```

Preguntas

¿?