mediaLab Documentation

Publicación 1

Johnny Cubides, Carolina Pulido

dic. 31, 2016

Índice general

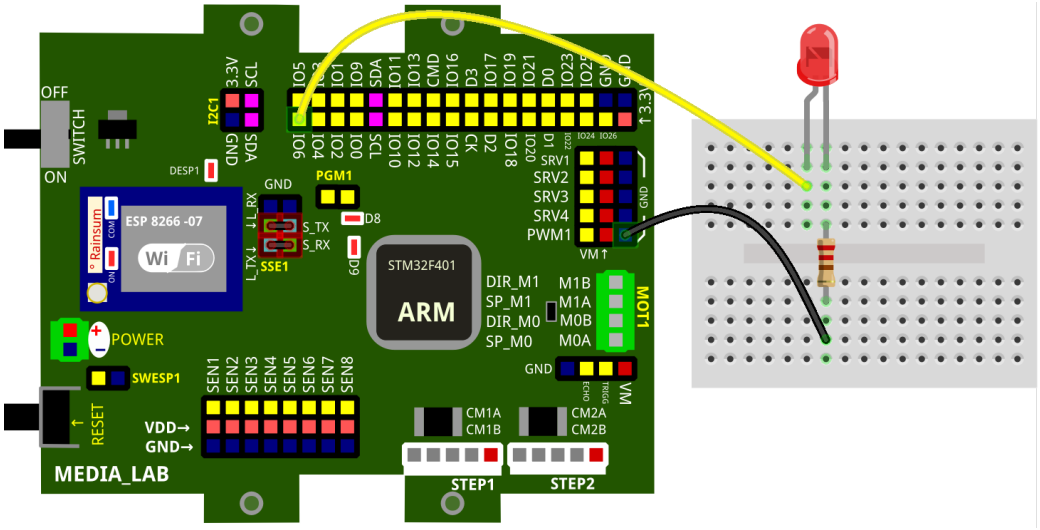
Contents:

Referencia Rápida

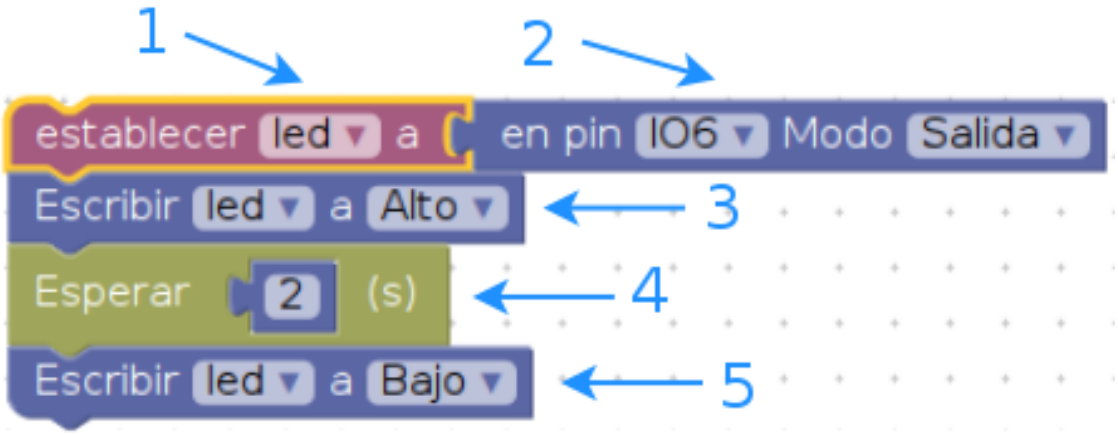
1.1 SALIDAS DIGITALES

En éste ejemplo un led será encendido durante un par de segundos para luego ser apagado.

1.1.1 Conexión física



1.1.2 Programa en blockly



Explicación de programa

En la imagen se numeran los bloques para luego ser explicados como sigue:

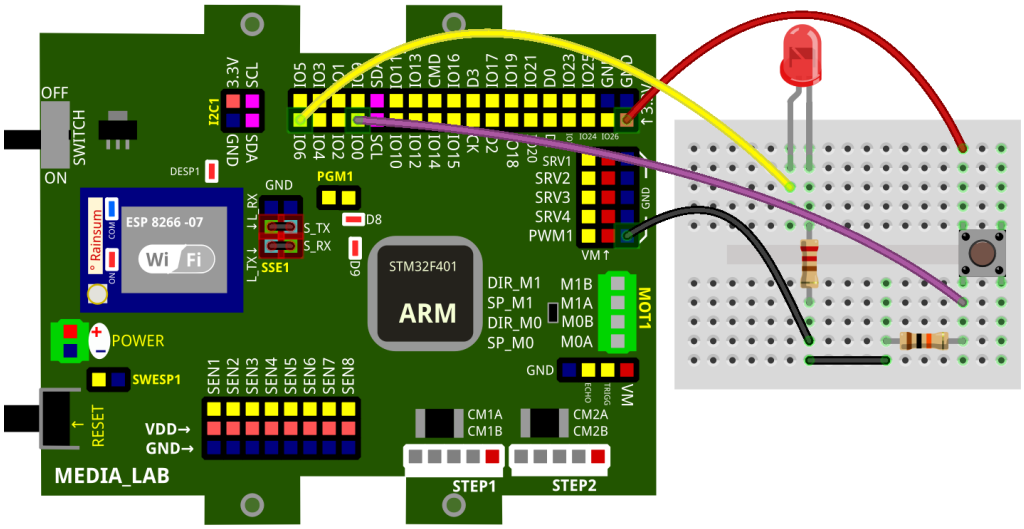
1. Los pines digitales son manejados como objetos, así que debe ser iniciado como una variable con un identificador (nombre), se sugiere un nombre explícito como en éste caso (led).
2. Aquí se ubica el nombre del pin correspondiente a la conexión física en la placa, además, permite definir el pin como una entrada o como una salida digital. (Compare la imagen de la conexión física con la del código blockly, notará que ambas se refieren al pin IO6).
3. Ya definido el pin (led) es hora de darle un estado, en éste caso es alto. Cuando se refiere a ALTO quiere decir que envía un 1 lógico a el pin (led), en lógica positiva indica que la presencia de energía estimula a el led a encenderse.
4. La permanencia en un estado depende de las transiciones, en éste caso pasar de un estado a otro estado depende del tiempo, éste bloque permite que se permanezca en un estado por determinados segundos.
5. Siendo éste el mismo bloque que está en **3** se ha indicado que pondrá el led en bajo, es decir, el led será apagado, terminando así con el programa.

1.2 ENTRADAS DIGITALES

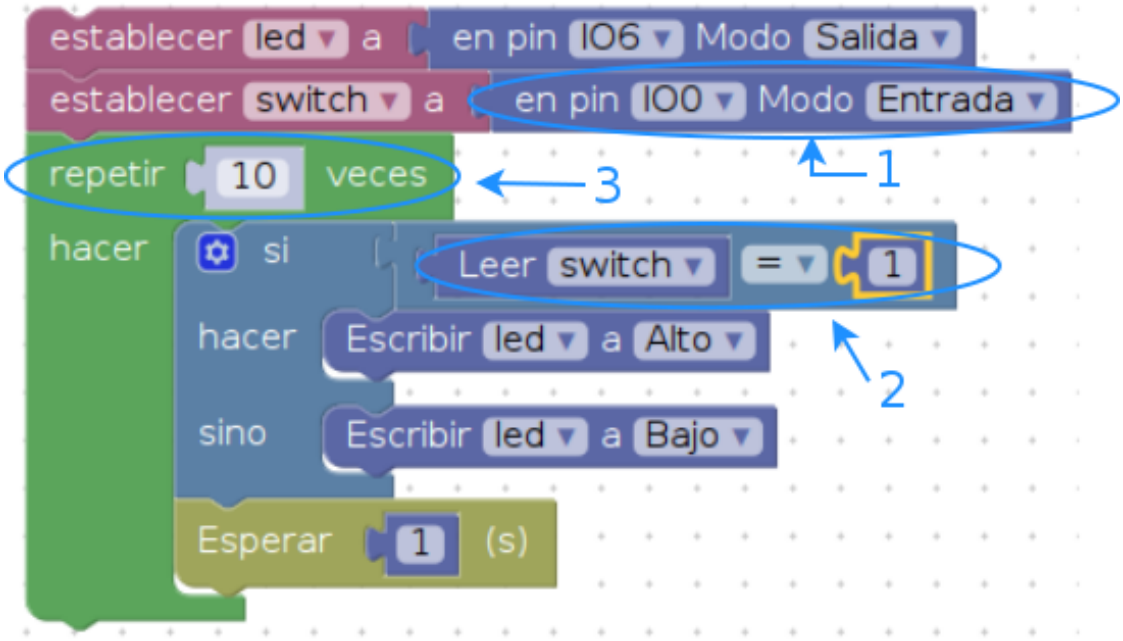
En éste ejemplo un switch (como entrada digital) controlará el estado de un led. Cuando el switch es oprimido el led es encendido, en caso contrario el led permanece apagado. Éste algoritmo será ejecutado únicamente diez veces, para luego terminar con el programa.

Nota: Se recomienda hacer ésta prueba con un bloque de secuencia finita, ya que con uno infinito, µpython no podrá ejecutar otra tarea hasta no recibir un reset.

1.2.1 Conexión física



1.2.2 Programa en blockly



Explicación de programa

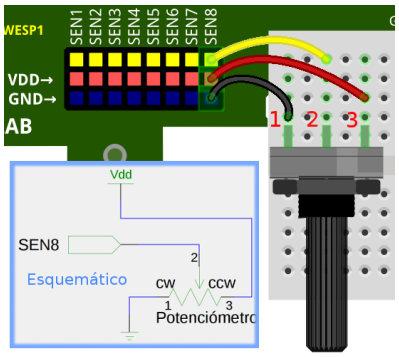
En la imagen se numeran los bloques relevantes, los demás fueron explicados en la sección SALIDAS DIGITALES; el bloque **si... hacer... sino** (en azul) será explicado de forma detallada en otra sección.

1. Este bloque denominado modo del pin permite construir el objeto **switch** en el pin **IO0** (ver también imagen *conexión física*) con características de entrada. Por tratarse de un pin digital solo tendrá dos posibles estados de lectura, **1** o **0**.

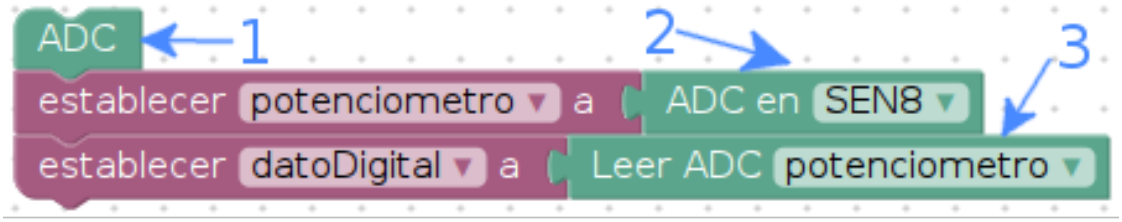
2. Este par de bloques permiten leer, retornar el estado del pin (switch) y comparar su estado. El valor retornado será 1 si se detecta presencia de tensión con un umbral de nivel positivo, de lo contrario, el valor será 0. Con la ayuda del bloque **si... hacer... sino**, se determina el estado que tendrá el led. El estado que tenga el led durará el valor asignado en el bloque de espera.
3. El bloque repetir, asigna un límite finito de repeticiones del algoritmo antes de terminar con el programa.

1.3 ENTRADAS ANÁLOGAS

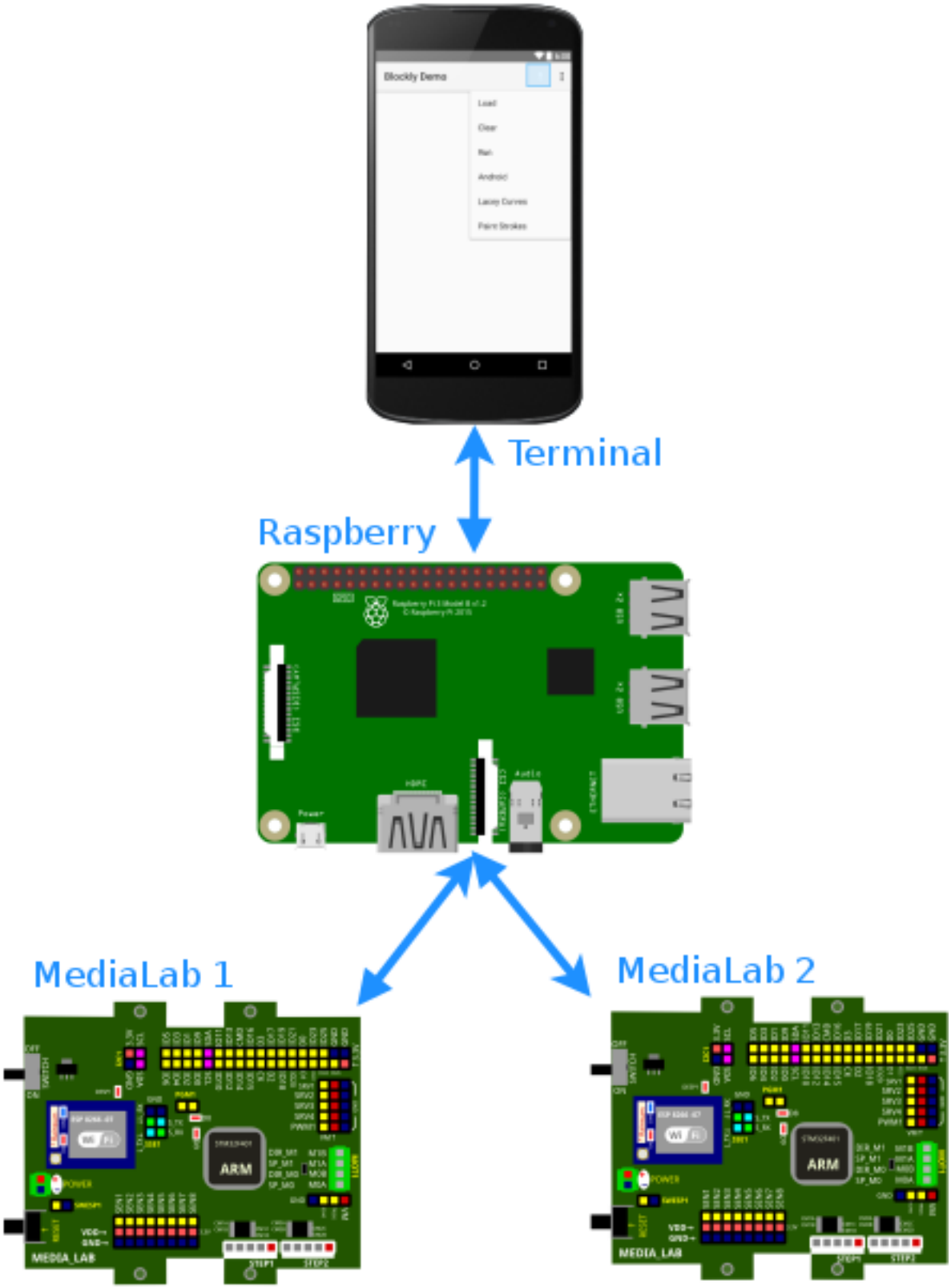
1.3.1 Conexión física

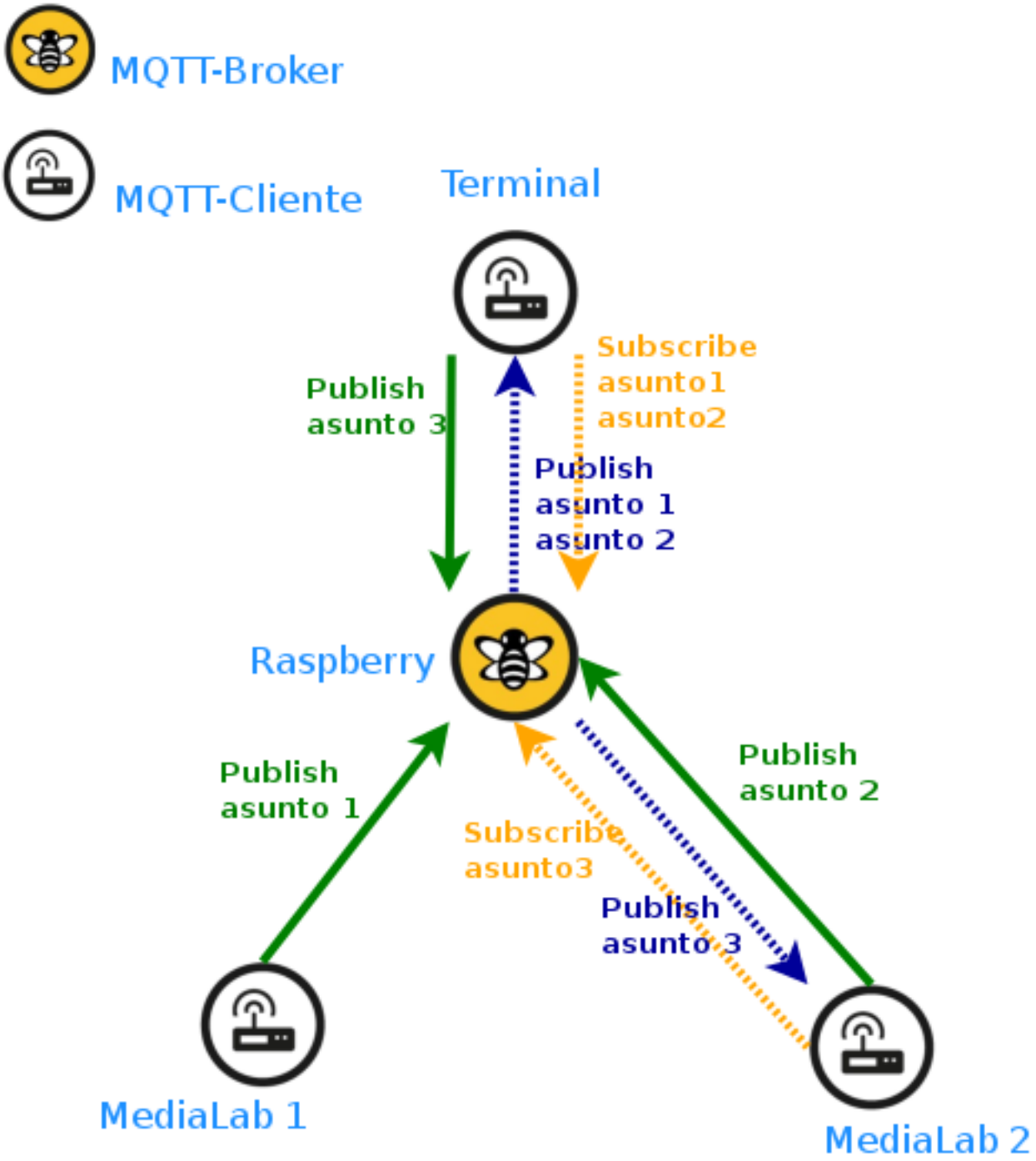


1.3.2 Programa en blockly



1.4 MQTT


Blockly Demo
Load
Clear
Run
Android
Lacey Curves
Faire Strokes
Terminal
Raspberry
MediaLab 1
MediaLab 2
ARM
MEDIA_LAB


MQTT-Broker
MQTT-Cliente
Terminal
Subscribe
asunto1
asunto2
Publish
asunto 3
Publish
asunto 1
asunto 2
Raspberry
Publish
asunto 1
Publish
asunto 2
Subscribe
asunto3
Publish
asunto 3
MediaLab 1
MediaLab 2

Guia de Uso

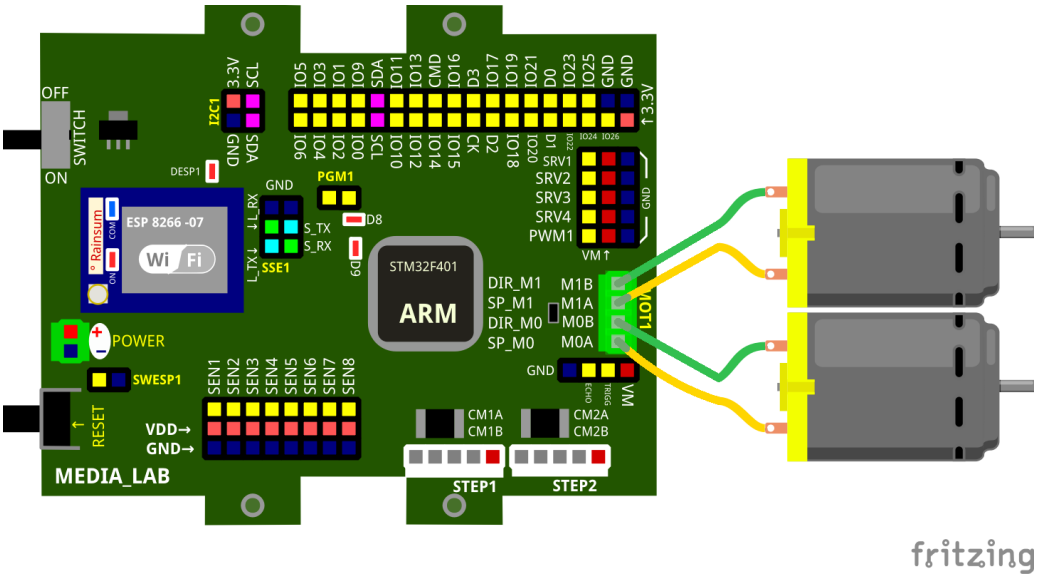
2.1 Hola Mundo

```
import pyb

def blink():
    LED = pyb.LED(1)
    LED.toggle()
    return

blink()
```

2.2 Manejo de Motores

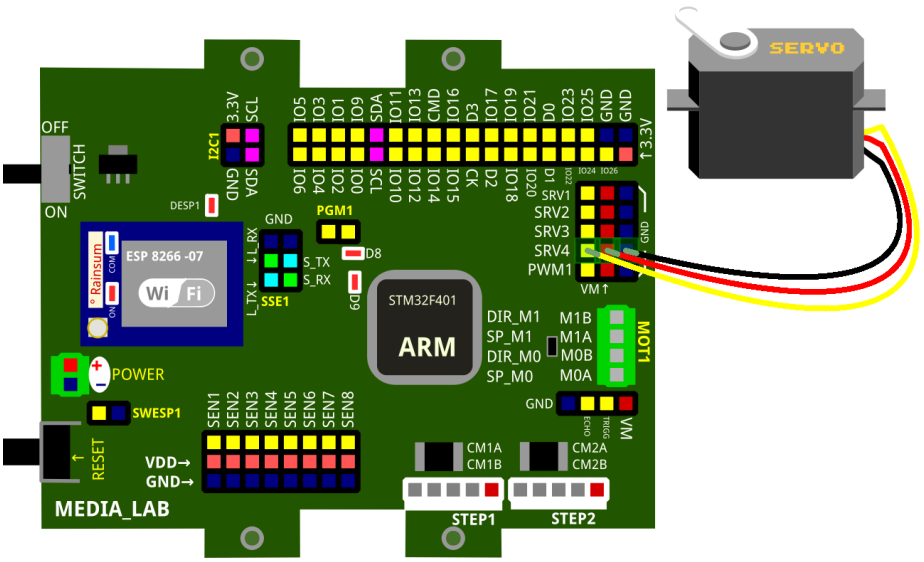



```
from pyb import Timer, Pin

class mobile:
    pl = 100
    pr = 100
```

```
    def __init__(self):
        timerMotors = Timer(1, freq=1000)
        self.lPWM = timerMotors.channel(2, Timer.PWM, pin=Pin('SP_M0'))
        self.rPWM = timerMotors.channel(4, Timer.PWM, pin=Pin('SP_M1'))
        self.ld = Pin('DIR_M0', mode=Pin.OUT)
        self.rd = Pin('DIR_M1', mode=Pin.OUT)
    def calibration(self, l, r):
        self.pl = l
        self.pd = r
    def run(self, l, r):
        self.lPWM.pulse_width_percent(int(abs(l)*self.pl/100))
        self.rPWM.pulse_width_percent(int(abs(r)*self.pr/100))
        if l < 0:
            self.ld.low()
        else:
            self.ld.high()
        if r < 0:
            self.rd.low()
        else:
            self.rd.high()
```

2.3 Manejo de ServoMotores




```
import pyb
class servomotor:
def __init__(self, pin):
    timer = pyb.Timer(4, freq=200)
    self.ch = timer.channel(5-pin, pyb.Timer.PWM, pin=pyb.Pin('SRV'+str(pin)))
def angulo(self,x):
    if 0 <= x and x <= 180:
        self.ch.pulse_width(4800+int(x*15600/180))
```

CAPÍTULO 3

Indices and tables

- genindex
- modindex
- search