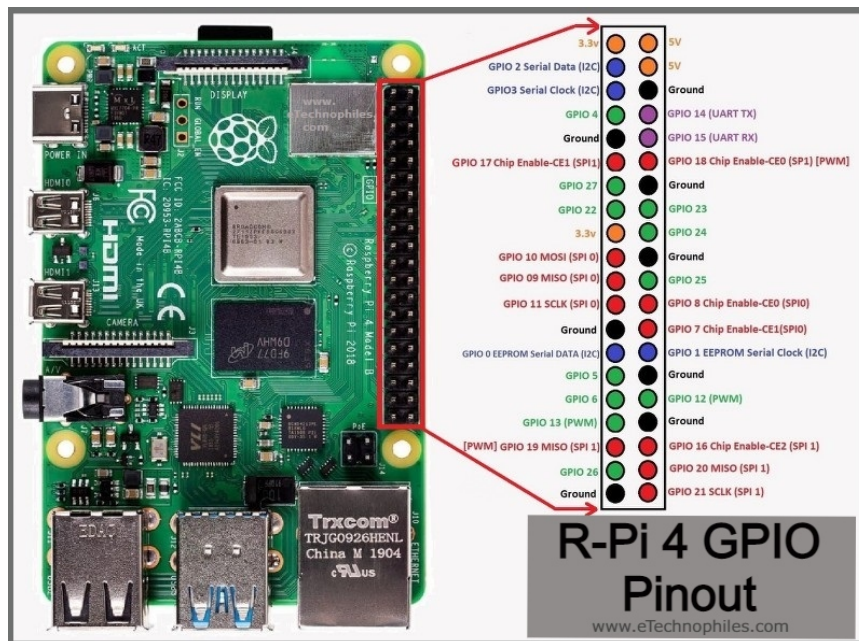


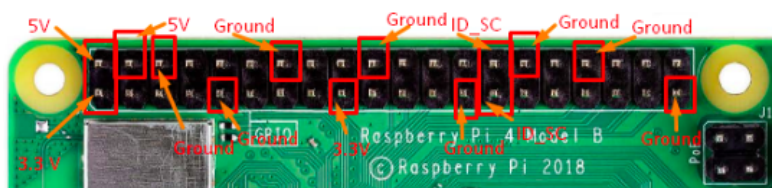
Fecha: 12.11.2022.17:17 UTC -3

## Introducción al scripting en SBC/MCU

Se realizará la programación de una tarjeta SBC (single-board computer, también conocido como Raspberry Pi, mientras que las MCU son unidades micro controladoras como las placas Arduino, ESP32, etc..) por medio del lenguaje de programación **Python** / **JavaScript** [En el simulador PT]. Denominamos sus pines GPIO: pines de entrada/salida de propósito general) para conectar físicamente al Raspberry (SBC) a un sensor o actuador.

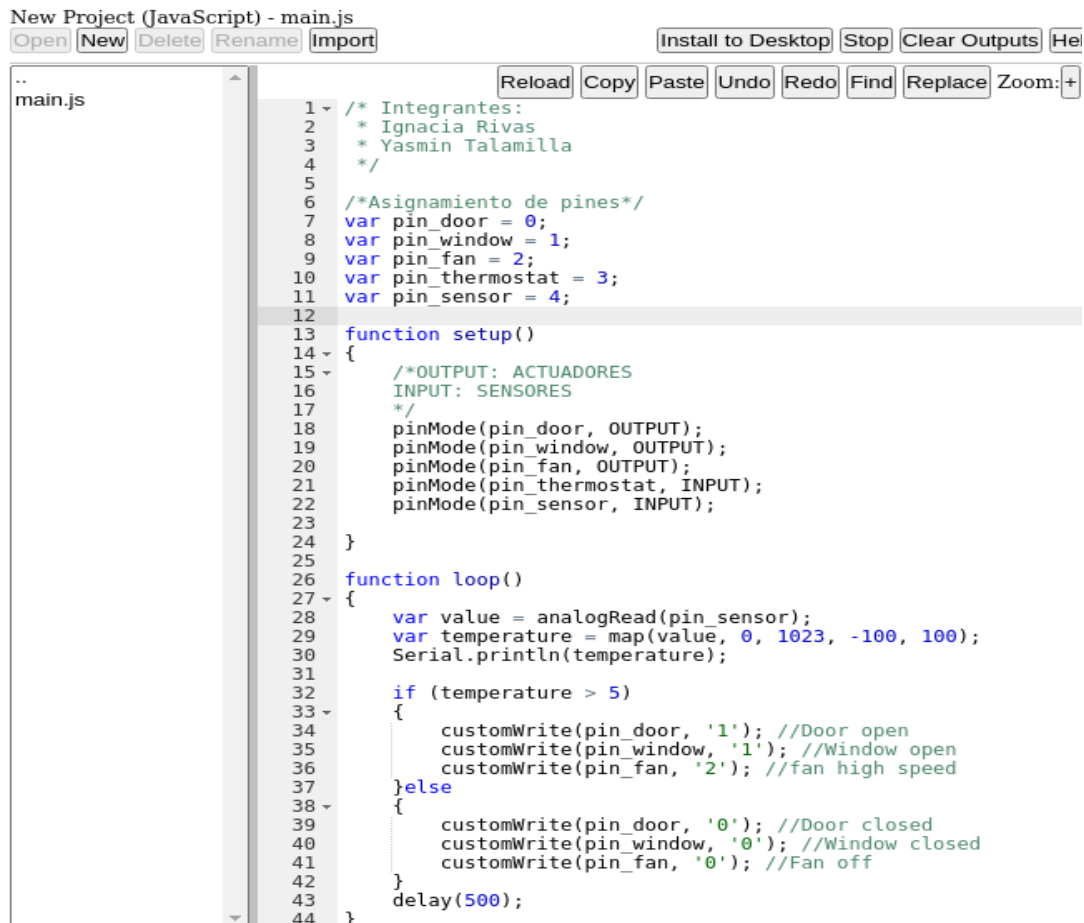


Les recomiendo tener el PinOut del dispositivo siempre a la mano.



## Ejemplo 1: JS Scripting [Packet Tracer]

He tomado como ejemplo el código T1 del grupo que logró el objetivo del bonus track sobre programar microcontroladores voluntariamente para el taller. A continuación se explica detalladamente.



```
New Project (JavaScript) - main.js
Open New Delete Rename Import Install to Desktop Stop Clear Outputs Hei
Reload Copy Paste Undo Redo Find Replace Zoom: +

.. main.js
1  /* Integrantes:
2   * Ignacia Rivas
3   * Yasmin Talamilla
4   */
5
6  /*Asignamiento de pines*/
7  var pin_door = 0;
8  var pin_window = 1;
9  var pin_fan = 2;
10 var pin_thermostat = 3;
11 var pin_sensor = 4;
12
13 function setup()
14 {
15   /*OUTPUT: ACTUADORES
16   INPUT: SENSORES
17   */
18   pinMode(pin_door, OUTPUT);
19   pinMode(pin_window, OUTPUT);
20   pinMode(pin_fan, OUTPUT);
21   pinMode(pin_thermostat, INPUT);
22   pinMode(pin_sensor, INPUT);
23 }
24
25
26 function loop()
27 {
28   var value = analogRead(pin_sensor);
29   var temperature = map(value, 0, 1023, -100, 100);
30   Serial.println(temperature);
31
32   if (temperature > 5)
33   {
34     customWrite(pin_door, '1'); //Door open
35     customWrite(pin_window, '1'); //Window open
36     customWrite(pin_fan, '2'); //fan high speed
37   }else
38   {
39     customWrite(pin_door, '0'); //Door closed
40     customWrite(pin_window, '0'); //Window closed
41     customWrite(pin_fan, '0'); //Fan off
42   }
43   delay(500);
44 }
```

**Paso 1:** Asignar variables a pines físicos de la placa.

**Paso 2:** definir la función setup() con el modo que operarán los GPIO

INPUT : Sensores | OUTPUT: Actuadores

**Paso 3:** definir la función loop() , donde realizaremos la lectura mientras se ejecuta nuestro programa.

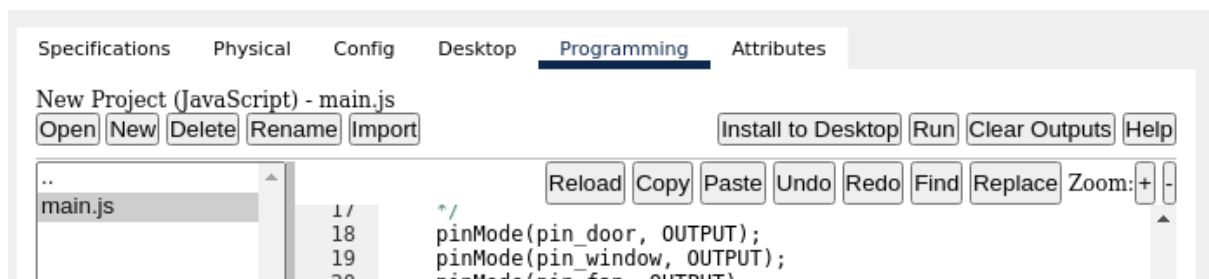
- [map\(\)](#) nos permite aplicar una función a un valor de un rango de valores.
- [analogRead\(\)](#) lee la señal analógica del sensor enlazado a un pin.

- [analogWrite\(\)](#) escribe una señal analógica de 0 (siempre apagado) a 255 (siempre prendido).
- [customWrite\(\)](#) escribe una señal solamente 1 byte.

**Paso 4:** Declarar las variables a utilizar.

- sensorValue almacena el valor leído del sensor.
- ledValue almacena el valor a enviar al led.
- ledPin almacena el valor del pin del led.

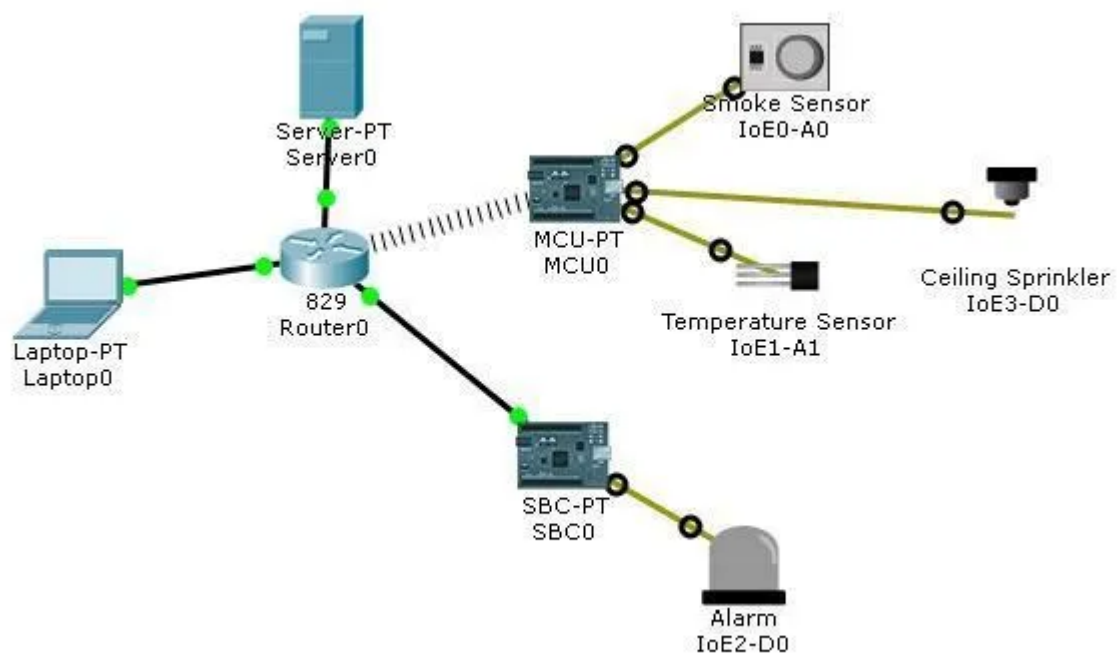
**paso 5:** Hacer click en “Run” para que el programa se ejecute.



## Ejemplo 2: Control de elementos usando un server IoE (Internet of Everything)

**[Esto no será evaluado en el taller 2]**

Por ejemplo, tenemos una topología del siguiente estilo:



El objetivo es configurar el servidor IoE desde consola desde el Router

Al igual que en el T1, para configurar este servicio, debemos tener conectividad en nuestro dispositivo, acceso a nuestra red wifi.

1. Activamos el AP integrado que posee este modelo de Router.
2. Realizaremos la configuración de credenciales wpa-psk mediante consola
3. Activamos nuestro AP con el ssid TEST y la clave AZERTYUIOP

```
dot11 ssid TEST
 authentication open
 authentication key-management wpa
 wpa-psk ascii 0 AZERTYUIOP
 guest-mode
```

#### 4. Configuramos la interface Dot11Radio0.

```
interface Dot11Radio0
 no ip address
 bridge-group 1
 encryption mode ciphers aes-ccm
 ssid TEST
```

#### 5. Ahora el flujo de paquetes del AP del router estará enlazado a la interfaz Wlan-GigabitEthernet0, en modo access.

```
interface Wlan-GigabitEthernet0
 switchport mode access
```

6. Configuramos el dhcp, excluimos los servidores y asignamos una dhcp pool llamada IoE\_POOL.
7. Anunciamos la red a la cual estaremos conectados con el server IoE.
8. Entramos a la interfaz Vlan 1 y le damos su SVI (default gateway virtual.. btw).

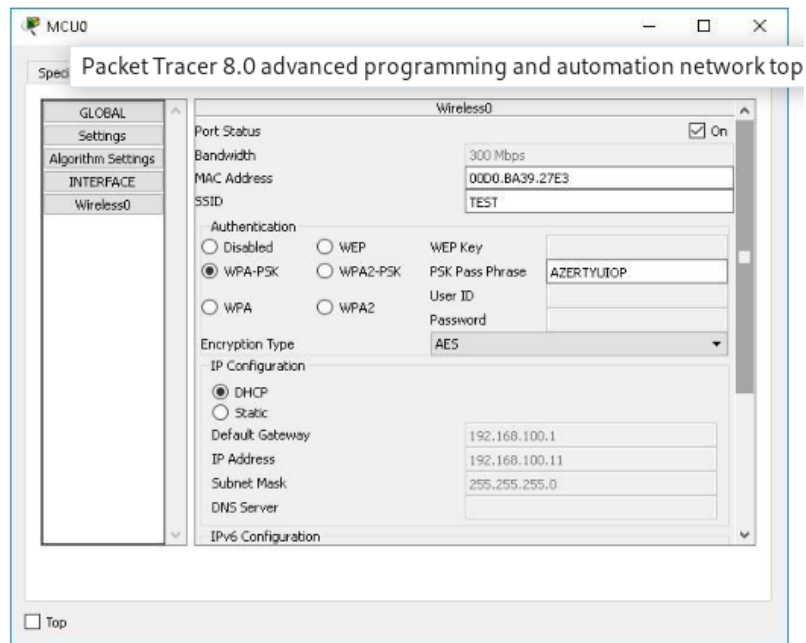
### Registrando una MCU en un server IoE

```
ip dhcp excluded-address 192.168.100.1 192.168.100.9
!
ip dhcp pool IoE_POOL
 network 192.168.100.0 255.255.255.0
 default-router 192.168.100.1

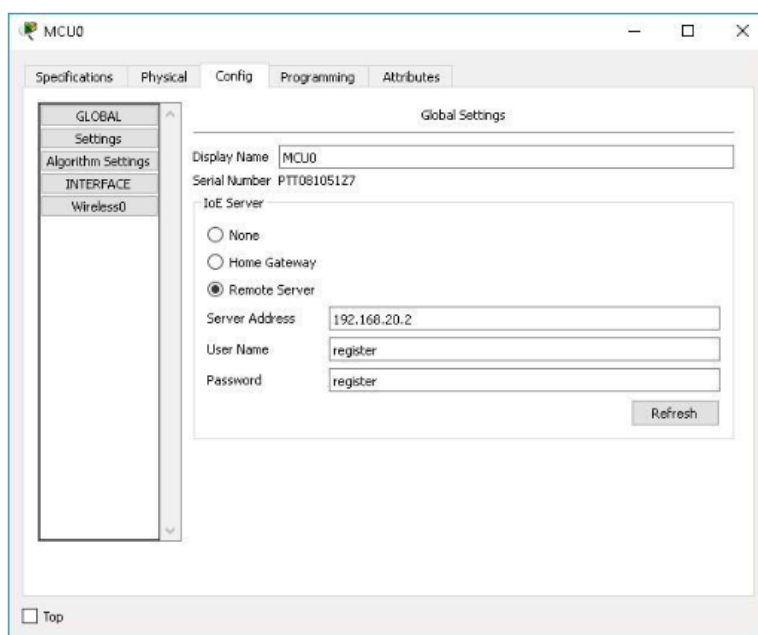
interface Vlan1
 ip address 192.168.100.1 255.255.255.0
```

## Acceso al server IoE desde una MCU

1. Colocamos las credenciales que le asignamos al Router.
2. Seleccionar asignamiento DHCP



3. Nos conectamos al servidor de manera remota como lo hicimos en el T1 anteriormente.



## código en JS para controlar sensores mediante una MCU utilizando un server IoT

```
1  function setup() {  
2    pinMode(1, OUTPUT);  
3    IoEClient.setup({  
4      type: "Fire Detection",  
5      states: [{  
6        name: "Fire",  
7        type: "bool"  
8      }],  
9      {  
10     name: "Temperature",  
11     type: "number",  
12     unit: "°C",  
13     imperialUnit: "°F",  
14     toImperialConversion: "x*1.8+32",  
15     toMetricConversion: "(x-32)/1.8",  
16     decimalDigits: 1  
17   }],  
18   {  
19     name: "Smoke Level",  
20     type: "number",  
21     unit: "ppm",  
22     decimalDigits: 1  
23   }]  
24   });  
25 }  
26  
27 function loop() {  
28   var SmokeLevel = analogRead(0);  
29   var Temperature = analogRead(1);  
30   var FireDetected=false;  
31  
32   if (SmokeLevel>50 || Temperature>580) {  
33     digitalWrite(0, HIGH);  
34     FireDetected=true;  
35   }  
36   else {  
37     digitalWrite(0, LOW);  
38     FireDetected=false;  
39   }  
40   IoEClient.reportStates([FireDetected, Temperature, SmokeLevel  
41   delay(500);  
42 }
```

Código para el SCB que se encuentra conectado a la alarma.

```
var state = 0;

function setup() {

    IoEClient.setup({
        type: "FireAlarm",
        states: [{
            name: "On",
            type: "bool",
            controllable: true
        }]
    });

    IoEClient.onInputReceive = function(input) {
        processData(input, true);
    };

    attachInterrupt(0, function() {
        processData(customRead(0), false);
    });

    setState(state);
}

function processData(data, bIsRemote) {
    if ( data.length <= 0 )
        return;
    setState(parseInt(data));
}

function setState(newState) {
    state = newState;

    if ( state === 0 )
        digitalWrite(0, LOW);
    else
        digitalWrite(0, HIGH);

    customWrite(0, state);
    IoEClient.reportStates(state);
    setDeviceProperty(getName(), "state", state);
}
```