

PROYECTO INTERGRADOR I

DOCENTE: GONZALO VERA

ALUMNO/A: KARINA JAZMIN BARBERO

CARRERA: TÉCNICO SUPERIOR EN TELECOMUNICACIONES

Trabajo Práctico #2

Fundamentos de Programación IoT y Módulos de Desarrollo

Objetivos

1. Entender y aplicar los fundamentos de la programación en Arduino y Micro Python en el contexto de IoT, enfocándose en el manejo de entradas y salidas digitales y analógicas.
2. Desarrollar habilidades prácticas en el uso de módulos de desarrollo y shields para ESP32, explorando diversas aplicaciones de los módulos en proyectos IoT.
3. Implementar controladores básicos utilizando ESP32, que gestionen efectivamente entradas y salidas tanto digitales como analógicas.
4. Familiarizarse con las herramientas de desarrollo como PlatformIO y RT-Thread en Visual Studio Code y utilizar la simulación en Wokwi para validar los programas desarrollados.

Desarrollo

1. Investigación de las plataformas de desarrollo:

- Documentar las características principales de Arduino (PlatformIO) y Micro Python (RT-Thread), destacando sus diferencias y aplicaciones en IoT.
- Investigar sobre diferentes módulos shield disponibles para ESP32 y su aplicación en controladores IoT.

Micro Python

- Micro Python es una implementación eficiente y optimizada del lenguaje de programación Python diseñada específicamente para microcontroladores.
- Permite escribir código en Python y ejecutarlo directamente en placas de microcontroladores.

- Ventajas:

- Sintaxis Python: Micro Python utiliza la misma sintaxis que Python estándar, lo que facilita la transición para aquellos familiarizados con Python.
- Ecosistema rico: Aprovecha las bibliotecas y módulos disponibles en el ecosistema de Python para acceder a funcionalidades avanzadas.

- Interactividad: Su Shell interactivo permite probar fragmentos de código en tiempo real, ideal para prototipado rápido y depuración.

- Capacidades de depuración: Proporciona herramientas de depuración como puntos de interrupción y la inspección de variables.

- Ideal para proyectos de robótica y sistemas embebidos.

Arduino

- es una plataforma electrónica de código abierto conocida por su simplicidad y facilidad de uso.

- Ventajas:

- Creatividad y simplicidad: Permite a los usuarios crear proyectos sin necesidad de conocimientos profundos en programación o electrónica.

- Amplia comunidad y recursos de aprendizaje.

- Ideal para proyectos de automatización, control y monitoreo.

- Utiliza un lenguaje de programación propio basado en C/C++.

Módulos Shield para ESP32 en IoT

Los módulos shield son placas de expansión que extienden las capacidades del microcontrolador ESP32. Algunas aplicaciones comunes incluyen sistemas de automatización del hogar, dispositivos IoT personalizados, sistemas de control y robótica.

Algunas opciones:

1. ESP32 PLC: Ofrece la posibilidad de expandirse con hasta 127 módulos a través de I2C, lo que permite hasta 7100 entradas/salidas en conexiones maestro-esclavo, además de sensores.
2. ESP32 Industrial PLC: Similar al anterior, pero diseñado específicamente para aplicaciones industriales.
3. ESP32-DevKitC: Placa de desarrollo con Wi-Fi, Bluetooth LE y 38 pines GPIO. Incluye puente USB a UART, botones, LED y ranura para tarjeta MicroSD.
4. ESP32-WROVER: Versión con compatibilidad para flash SPI externo y mayor cantidad de memoria RAM y flash.

2. Ejercicios de Implementación:

- a- Controlador de Entradas Digitales: Crear un sketch en Arduino y un script en Micro Python que lea el estado de un botón y encienda un LED cuando el botón esté presionado.
- b- Controlador de Entradas Analógicas: Desarrollar un programa que lea valores de un sensor de temperatura y los muestre en el Serial Monitor/consola.
- c- Controlador de Salidas Digitales: Implementar un sistema que alterne el encendido de un conjunto de LEDs en intervalos regulares.
- d- Controlador de Salidas Analógicas: Escribir un código que controle la intensidad de un LED usando PWM basado en la lectura de un potenciómetro.

Ejercicio a:

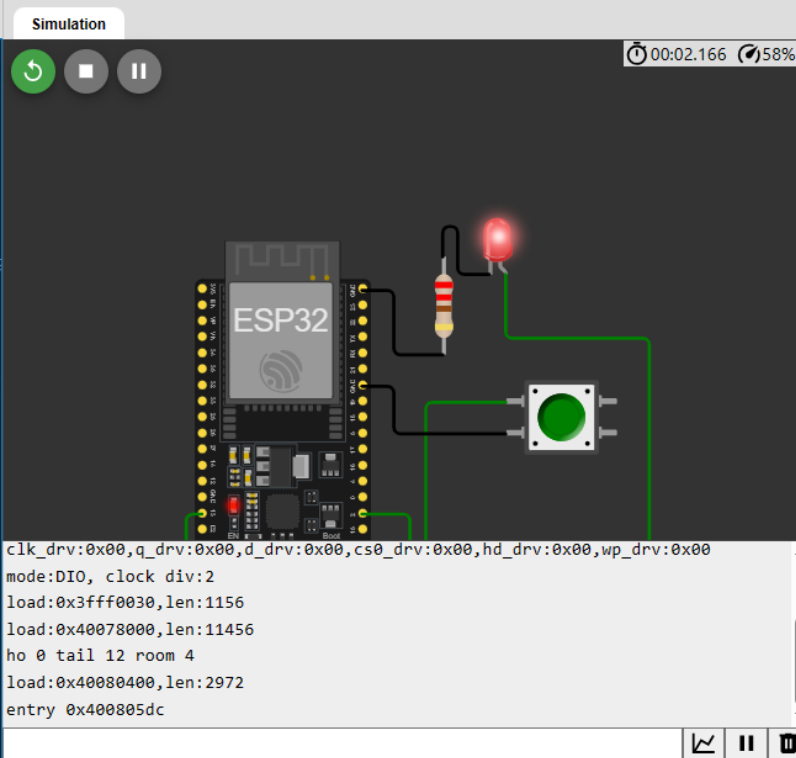
WOKWI SAVE SHARE TP2 A- Controlador de Entradas Digitales (ARDUINO) Docs

sketch.ino diagram.json Library Manager

```
1 const int BUTTON_PIN = 2; // Pin donde se conecta el botón
2 const int LED_PIN = 13; // Pin donde se conecta el LED
3
4 void setup() {
5   pinMode(BUTTON_PIN, INPUT_PULLUP); // Configura el pin del botón como entrada con
6   pinMode(LED_PIN, OUTPUT); // Configura el pin del LED como salida
7 }
8
9 void loop() {
10  int buttonState = digitalRead(BUTTON_PIN); // Lee el estado del botón
11  if (buttonState == LOW) { // Si el botón está presionado (LOW deb
12    digitalWrite(LED_PIN, HIGH); // Enciende el LED
13  } else { // Si no...
14    digitalWrite(LED_PIN, LOW); // Apaga el LED
15  }
16 }
17
```

Simulation






00:02.166 58%



clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1156
load:0x40078000,len:11456
ho 0 tail 12 room 4
load:0x40080400,len:2972
entry 0x400805dc

<https://wokwi.com/projects/398074577147374593>

Ejercicio a: Python

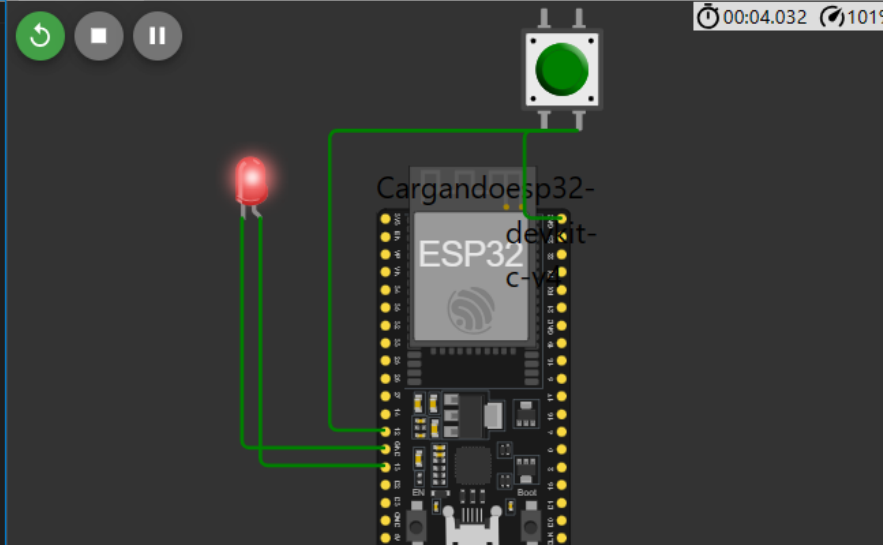
WOKWI  SALVAR  COMPARTIR  TP2 PI A- Controlador de Entradas Digitales  Docs 

main.py diagram.json

```
1 desde el pin de importación de la máquina
2 Tiempo de importación
3
4 led = Pin(13, Pin.OUT)
5 botón = Pin(12, Pin.IN, Pin.PULL_UP)
6
7 mientras que True:
8     if button.value() == 0: # Botón presionado
9         led.value(1) # Enciende el LED
10        De lo contrario:
11        led.value(0) # Apaga el LED
12        time.sleep(0.1)
13
14
15
16
17
18
19
20
21
```

Simulación

00:04.032 1019



Cargando esp32-
de kit-
c-y

ets Jul 29 2019 12:21:46

```
rst:0x1 (POWERON_RESET),boot:0x13 (SPY_FUAT_FLUSH_BOOT)
confYgsYp: 0, SPYWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DY0, clock dyw:2
load:0x3fff0030,len:4728
load:0x40078000,len:14888
load:0x40080400,len:3368
entry 0x400805cc
```

<https://wokwi.com/projects/396185768982348801>

Ejercicio b:

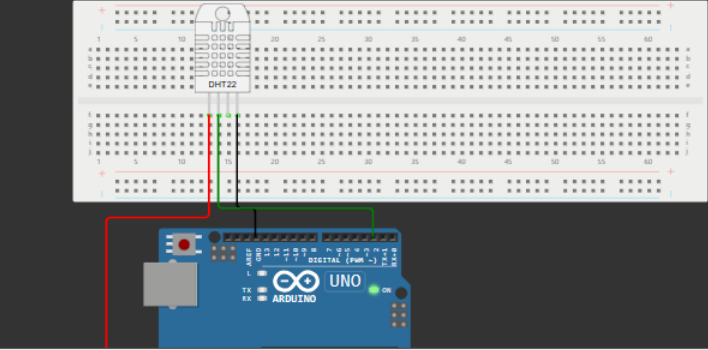
WOKWI TP2 B-Controlador de Entradas Analógicas

sketch.ino diagram.json libraries.txt Administrador de la biblioteca

```
1 #include <DHT.h>
2 #define PinSensor 2
3
4 DHT dht (PinSensor, DHT22);
5
6 Configuración nula() {
7   Serie.begin(9600);
8   dht.empezar();
9 }
10
11
12
13 Bucle vacío() {
14   Retraso(2000);
15   Flotación humedad = DHT.readHumedad();
16   Temperatura del flotador = DHT.readTemperature();
17   Serie.print("Humedad: ");
18   Serie.print(humedad);
19   Serie.Impresión("% - ");
20   Serie.print("Temperatura: ");
21   Serie.print(temperatura);
22   Serie.println(" °C");
23 }
24
```

Simulación

00:03.363 96%



Humedad: 40.00% - Temperatura: 24.00 °C

<https://wokwi.com/projects/396499626790118401>

Ejercicio c:

WOKWI SAVE SHARE TP2 C-Controlador de Salidas Digitales Docs

sketch.ino diagram.json Library Manager

```
1 // Definición de pines para los LEDs
2 const int ledPin1 = 12;
3 const int ledPin2 = 13;
4 const int ledPin3 = 11;
5
6 // Tiempo de encendido y apagado (en milisegundos)
7 const unsigned long tiempoEncendido = 1000; // 1 segundo
8 const unsigned long tiempoApagado = 1000; // 1 segundo
9
10 void setup() {
11   // Configura los pines como salidas
12   pinMode(ledPin1, OUTPUT);
13   pinMode(ledPin2, OUTPUT);
14   pinMode(ledPin3, OUTPUT);
15 }
16
17 void loop() {
18   // Enciende el primer LED
19   digitalWrite(ledPin1, HIGH);
20   delay(tiempoEncendido);
21   digitalWrite(ledPin1, LOW);
22
23   // Enciende el segundo LED
24   digitalWrite(ledPin2, HIGH);
25   delay(tiempoEncendido);
26   digitalWrite(ledPin2, LOW);
27
28   // Enciende el tercer LED
29   digitalWrite(ledPin3, HIGH);
30   delay(tiempoEncendido);
31   digitalWrite(ledPin3, LOW);
32
33   // Espera antes de repetir
34   delay(tiempoApagado);
35 }
```

Simulation 00:05.412 93%

<https://wokwi.com/projects/396546877915864065>

Ejercicio d:

WOKWI SAVE SHARE TP2 D-Controlador de Salidas Analógicas Docs

sketch.ino diagram.json Library Manager

```
1 const int LED = 3; // Indicando que el LED se encuentra en el pin 3
2 const int potenciometro = A0; // El potenciómetro está conectado al pin A0
3 int intensidad; // Variable para la intensidad de brillo
4
5 void setup() {
6   pinMode(LED, OUTPUT);
7 }
8
9 void loop() {
10  intensidad = analogRead(potenciometro); // Leemos el valor del potenciómetro
11  analogWrite(LED, intensidad); // Controlamos el brillo del LED con PWM
12 }
13
```

Simulation 00:07.246 101%

The simulation shows an Arduino Uno microcontroller board. A potentiometer is connected to the board with its wiper to pin A0 and its outer terminals to pins 5V and GND. An LED is connected with its anode to pin 3 and its cathode to GND. The potentiometer is represented by a slider on a scale from 0 to 100, with a red dot indicating the current position.

<https://wokwi.com/projects/396773522574255105>

3. Simulación en Wokwi:

- Realizar simulaciones de cada uno de los controladores implementados para validar su funcionamiento antes de la carga en el hardware real.

Entrega

- Formato de Entrega: El trabajo debe ser presentado en el monorepositorio creado para el TP#1 en formato Markdown (.md), documentos PDF y proyectos desarrollados.
- Contenido Requerido:
 - Documentos de investigación sobre Arduino y Micro Python.
 - Códigos fuente de los controladores desarrollados.
 - Capturas de pantalla o enlaces a las simulaciones en Wokwi demostrando el funcionamiento de los controladores con sus respectivos proyectos en VsCode.
- Fecha de Entrega: La fecha de entrega será el último día de la Semana 2 21/04.