UNIVERSIDAD DE GUADALAJARA

CENTRO UNIVERSITARIO DE LOS LAGOS

DIVISIÓN DE ESTUDIOS DE LA BIODIVERSIDAD E INNOVACIÓN TECNOLÓGICA

PRACTICA 6

Asignatura:

Sistemas Embebidos

Presenta:

Oscar Iván Moreno Gutiérrez #220942754

Arnold Jonathan Bradley Mercado Plascencia #220942835

Alejandro Orozco Ramirez #217490257

Profesor:

Dr. Afanador Delgado Samuel Mardoqueo

Fecha:

31 de octubre de 2024

Índice general

Palabras Clave

Aquí van las palabras clave de tu documento.

Objetivo

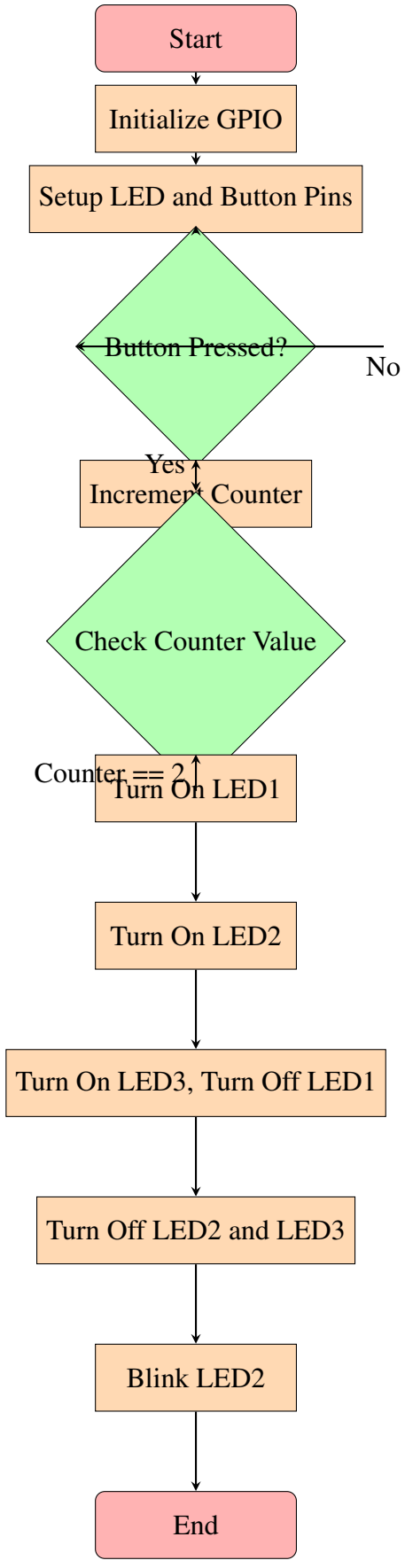
Desarrollar coodigo de configuracion y ejecucion de los puertos GPIO de sistema embebido mediante lenguaje de alto nivel.

Contenido

1.1 Material

- Raspberry Pi
- Protoboard
- Cables
- Resistencias
- 3 LEDs
- Push Button

1.2 Diagrama de Flujo del Programa



1.3 Programa en Python

```
import RPi.GPIO as GPIO
import time
## Modo
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

## Declaracion de Puertos
buttonpin = 18
led1,led2,led3 = 16,20,21

## Declaracion de direccion de puertos
## LEDS
GPIO.setup(led1,GPIO.OUT)
GPIO.setup(led2,GPIO.OUT)
GPIO.setup(led3,GPIO.OUT)

## Boton
GPIO.setup(buttonpin,GPIO.IN,pull_up_down=GPIO.PUD_UP)

##programa
cont = 0
res = False

def prenderLED(ledID):
  GPIO.output(ledID,True)
def apagarLED(ledID):
  GPIO.output(ledID,False)

#def setPWM():

try:
  while True:
    current_state = GPIO.input(buttonpin)
    if current_state == GPIO.LOW and not res:
```

```
      cont += 1
      print(f"Presion numero: {cont}")
      res = True
    if current_state == GPIO.HIGH:
      res = False

    ## PARTE DE PRENDER LEDS
    if cont == 2:
      prenderLED(led1)
    elif cont == 3:
      prenderLED(led2)
    elif cont == 4:
      prenderLED(led3)
      apagarLED(led1)
    elif cont == 5:
      apagarLED(led2)
      apagarLED(led3)
    elif cont == 6:
      time.sleep(1)
      prenderLED(led2)
      time.sleep(0.2)
      apagarLED(led2)

    elif cont == 7:
      print("FUgaaaaaa")
      break
    time.sleep(0.1)

except KeyboardInterrupt:
  print("Adios")
finally:
  GPIO.cleanup()
```

1.4 Codigo python con interrupciones

```
import RPi.GPIO as GPIO
import time
```

```
# Modo
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

# Declaracion de Puertos
buttonpin = 4
led1, led2, led3 = 16, 20, 21

# Declaracion de direccion de puertos
# LEDS
GPIO.setup(led1, GPIO.OUT)
GPIO.setup(led2, GPIO.OUT)
GPIO.setup(led3, GPIO.OUT)

# Boton
GPIO.setup(buttonpin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# Variables
cont = 0
res = False

def prenderLED(ledID):
    GPIO.output(ledID, True)

def apagarLED(ledID):
    GPIO.output(ledID, False)

def button_callback(channel):
    global cont, res
    if not res:
        cont += 1
        print(f"Presion numero: {cont}")
        res = True

    # PARTE DE PRENDER LEDS
```

```
    if cont == 2:
        prenderLED(led1)
    elif cont == 3:
        prenderLED(led2)
    elif cont == 4:
        prenderLED(led3)
        apagarLED(led1)
    elif cont == 5:
        apagarLED(led2)
        apagarLED(led3)
    elif cont == 6:
        time.sleep(1)
        prenderLED(led2)
        time.sleep(0.2)
        apagarLED(led2)
    elif cont == 7:
        print("FUgaaaaaa")
        GPIO.cleanup()
        exit()

# Setup interrupt
GPIO.add_event_detect(buttonpin, GPIO.FALLING, callback=button_callb

try:
    while True:
        current_state = GPIO.input(buttonpin)
        if current_state == GPIO.HIGH:
            res = False
        time.sleep(0.1)

except KeyboardInterrupt:
    print("Adios")
finally:
    GPIO.cleanup()
```

1.5 Fotos

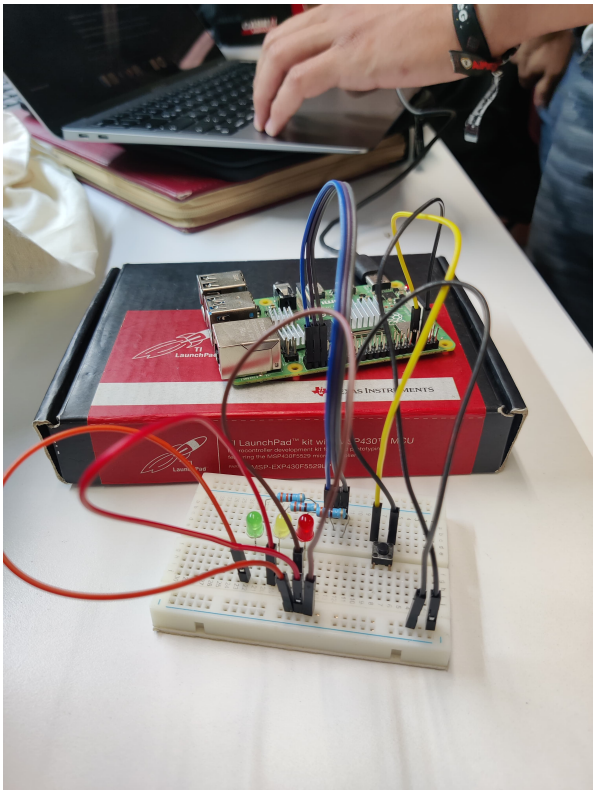


Figura 1.1: En 0s

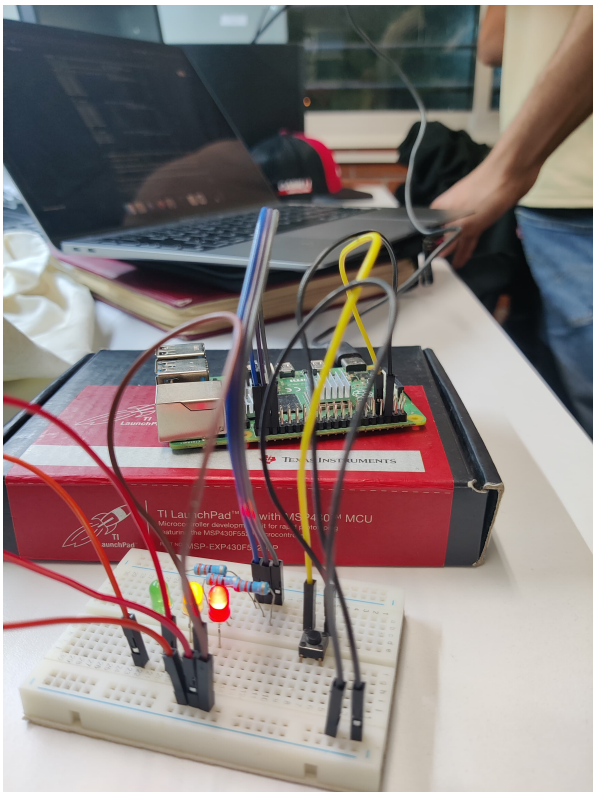


Figura 1.2: 3 pulsaciones

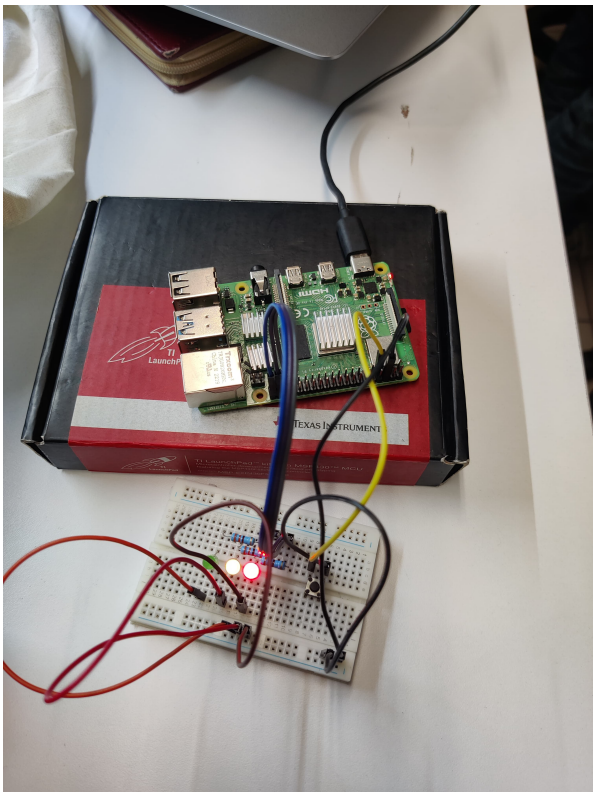


Figura 1.3: 3 pulsaciones tambien

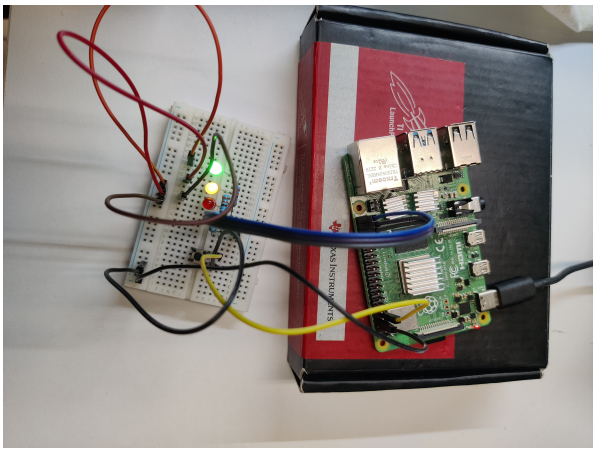


Figura 1.4: 4 pulsaciones

Conclusiones

Utilizando pollnig para hacer el program toma mas recursos del sistema, en cambio con interrupciones el sistema se mantiene en espera de una señal para ejecutar una accion, lo que hace que el sistema sea mas eficiente.