INSTITUTO TECNOLÓGICO DE SONORA
Educar para Trascender

Portafolio de evidencias

Integrantes:

Fiol Becheret Jassiel Uldarico

Manríquez Valenzuela Maribel

Navarro Villa Rey David

Rodríguez Nápoles Saul Antonio

Optativa I
Carlo Soto

Guaymas, Sonora

ISW
Ingeniero en Software

Índice:

I. Semáforo sin botón

Descripción:

En esta práctica debíamos simular la tarea que realiza un semáforo, es decir; definir tiempos para el cambio de luces e indicar cuando puede cruzar el peaton.

Componentes:

- Raspberry
- Breadboard
- 4 LEDS
- Cables

Principios:

En esta práctica utilizamos la librería GPIO y utilizamos métodos para cada acción (peatones cruzando, encender luces y apagarlas). Los LEDS son manejados como salidas.

Procedimiento:

Paso 1: Unimos la raspberry de la siguiente manera, para que coincida con la programación hecha.

Se presenta la evidencia y muestra en el siguiente video

Paso 2: Programar el código (Adjunto en el siguiente punto)

Paso 3: Compilar el código.

Código:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

#pins
ledVerde = 14
ledAmarillo = 15
ledRojo = 18
boton = 11

ledPeaton = 23
activado = False

#entradas y salidas
GPIO.setup(ledVerde, GPIO.OUT)
GPIO.setup(ledAmarillo, GPIO.OUT)
GPIO.setup(ledRojo, GPIO.OUT)
GPIO.setup(boton, GPIO.IN)
GPIO.setup(ledPeaton, GPIO.OUT)

#metodos
def pasandoPeatones():
    parpadear(led     Verde)
```

```
    encender(ledAmarillo)
    GPIO.output(ledRojo, True)
    time.sleep(0.1)
    print("PASANDO PEATONES")
    parpadear(ledPeaton)
    GPIO.output(ledRojo, False)

def encender(led):
    GPIO.output(led, True)
    time.sleep(2)
    GPIO.output(led, False)

def parpadear(led):
    for i in range(5):
        GPIO.output(led, True)
        time.sleep(0.4)
        GPIO.output(led, False)
        time.sleep(0.4)

#loop
while True:
    GPIO.output(ledVerde, True)

    for i in range(40):  #esperar 4 segundos a leer un botón
        if GPIO.input(boton) == True:
            pasandoPeatones()
            activado = True
            break
```

```
            time.sleep(0.1)
        if activado == False:
            pasandoPeatones()


        activado = False


    GPIO.cleanup() #limpiar GPIO
```

Conclusión:

Al igual que trabajar con arduino, raspberry es muy intuitivo y debido a que la práctica es igual sólo en diferente lenguaje, fue muy fácil realizarla, pero estuvo interesante como varía de arduino a raspberry.

II. Led Secuencial

Descripción:

En la siguiente práctica controlaremos la intensidad con la que se encienden y apagan los leds para que se vea de forma consecutiva, haciendo parecer que la luz viaja de un led a otro.

Componentes:

- Raspberry
- Breadboard
- 6 LEDS
- Cables

Principios:

Utilizaremos la modulación por ancho o de pulso (PWM) la cual nos servirá para poder modificar la cantidad de energía que se envía.

Se debe crear un bucle que cambie los ciclos de trabajo de los LEDs, de forma que cuando uno está al 100% ,el otro estará al 0%.

Procedimiento:

Paso 1: Unimos la raspberry de la siguiente manera, para que coincida con la programación hecha.

Se presenta la evidencia y muestra en el siguiente video

Paso 2: Programar el código (Adjunto en el siguiente punto)

Paso 3: Compilar el código.

Código:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

#pin leds
pinArray = [14,15,18,23,24,25]

#salidas
for pin in pinArray:
    GPIO.setup(pin, GPIO.OUT)
```

```
while True:

    for iteracion in range(2):
        for contador in range(5):
            GPIO.output(pinArray[contador], True)
            time.sleep(0.01)
            GPIO.output(pinArray[contador+1], True)
            time.sleep(0.01)
            GPIO.output(pinArray[contador], False)
            time.sleep(0.02)

        pinArray.reverse()



GPIO.cleanup() #limpiar GPIO
```

Conclusión:

Lo que pudimos observar es que se puede jugar con el ojo humano haciéndole creer que la luz está pasando de un led a otro, ya que no se puede percibir lo que en realidad ocurre a simple vista.

III. Semáforo con botón

Descripción:

En la siguiente práctica simularemos un semáforo en el cual, un peaton cruza la calle al ponerse el led en rojo, este semáforo se activa al pulsar el botón.

Componentes:

- Raspberry
- Breadboard
- 4 LEDS
- Cables
- Botón

Principios:

En esta práctica utilizamos la librería GPIO, utilizando distintos métodos para cada acción todo esto controlado mediante la pulsación de un botón.

Procedimiento:

Paso 1: Unimos la raspberry de la siguiente manera, para que coincida con la programación hecha.

Se presenta la evidencia y muestra en el siguiente video

Paso 2: Programar el código (Adjunto en el siguiente punto)

Paso 3: Compilar el código.

Código:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

#pins
```

```
ledVerde = 14
ledAmarillo = 15
ledRojo = 18
boton = 11

ledPeaton = 23
activado = False

#entradas y salidas
GPIO.setup(ledVerde, GPIO.OUT)
GPIO.setup(ledAmarillo, GPIO.OUT)
GPIO.setup(ledRojo, GPIO.OUT)
GPIO.setup(boton, GPIO.IN)
GPIO.setup(ledPeaton, GPIO.OUT)

#metodos
def pasandoPeatones():
    parpadear(ledVerde)
    encender(ledAmarillo)
    GPIO.output(ledRojo, True)
    time.sleep(0.1)
    print("PASANDO PEATONES")
    parpadear(ledPeaton)
    GPIO.output(ledRojo, False)


def encender(led):
    GPIO.output(led, True)
```

```
    time.sleep(2)
    GPIO.output(led, False)


def parpadear(led):
    for i in range(5):
        GPIO.output(led, True)
        time.sleep(0.4)
        GPIO.output(led, False)
        time.sleep(0.4)


#loop
while True:

    GPIO.output(ledVerde, True)

    for i in range(40):  #esperar 4 segundos a ler un boton
        if GPIO.input(boton) == True:
            pasandoPeatones()
            activado = True
            break
        time.sleep(0.1)

    if activado == False:
        pasandoPeatones()

    activado = False
```

```
GPIO.cleanup() #limpiar GPIO
```

Conclusión:

Realizar esta actividad es muy similar a realizarla con Arduino, se utiliza la misma lógica cambiando el lenguaje de programación. Anteriormente se había realizado esta práctica, así que no hubo muchos problemas para realizarse.

IV. Sensor PIR

Descripción:

En esta práctica se simulo con el sensor de movimiento una alarma para detectar si hay intrusos en nuestra área.

Componentes:

- Raspberry
- Breadboard
- 1 LEDS
- Cables
- Sensor PIR

Principios:

Utilizando la librería GPIO y time, además del sensor de movimiento se hizo una alarma que se activa con el movimiento, esto gracias al sensor PIR, si se detecta movimiento nos dirá que hay intrusos, en caso de que no, nos arrojará que no se detecta nada.

Procedimiento:

Paso 1: Unimos la raspberry de la siguiente manera, para que coincida con la programación hecha.

Se presenta la evidencia y muestra en el siguiente video

Paso 2: Programar el código (Adjunto en el siguiente punto)

Paso 3: Compilar el código.

Código:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

#pins
pinIR = 18
pinLed = 23

#entradas y salidas
GPIO.setup(pinLed, GPIO.OUT)
GPIO.setup(pinIR, GPIO.IN)
```

```
while True:
    leer = GPIO.input(pinIR)

    if leer == False:
        print("Intrusos")
        GPIO.output(pinLed, True)
        time.sleep(0.1)
    else:
        print("No hay instrusos")
        GPIO.output(pinLed, False)
        time.sleep(0.1)

    time.sleep(0.1)

GPIO.cleanup() #limpiar GPIO
```

Conclusión:

Esta actividad puede ser muy útil en la vida real y realizarla fue algo sencillo, la detección con el sensor PIR funciono correctamente, y si nos arrojaba datos precisos.

V. Domótica

Descripción:

En este proyecto se unieron varias prácticas y usamos los distintos sensores de manera que interactuaran entre sí.

Componentes:

- Raspberry PI
- Breadboard
- 3 LED
- Cables
- Foto Resistencia
- Sensor de proximidad
- Botón
- Buzzer

Principios:

En este proyecto, todo interactúa entre sí, en donde dependiendo de las condiciones, se activa la alarma o no, a cierta hora del día se activa junto con el buzzer para hacer sonido, etc.

Procedimiento:

Paso 1: Unir el arduino de la siguiente manera, conectando todos los cables, sensores, etc.

Ejemplo en el siguiente video

Paso 2: Escribir el código necesario para que los componentes funcionen

Paso 3: Ejecutar el código.

Código:

```
import RPi.GPIO as GPIO

import time

import random


GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)


#pins planta

pinRegado = 3 #led azul


#pins sensor de luz

pinLDR = 23 #sensor de luz


#pins sensor de movimiento

pinPIR = 4 #sensor de movimiento


#pins alarma

pinVerde = 8

pinRojo = 7

pinBuzzer = 10

pinBoton = 9
```

```
#entradas y salidas
GPIO.setup(pinRegado, GPIO.OUT)
GPIO.setup(pinPIR, GPIO.IN)

GPIO.setup(pinVerde, GPIO.OUT)
GPIO.setup(pinRojo, GPIO.OUT)
GPIO.setup(pinBuzzer, GPIO.OUT)
GPIO.setup(pinBoton, GPIO.IN)

#metodos sensor de luz
def RCtime(RCpin):
    leer = 0
    GPIO.setup(RCpin, GPIO.OUT)
    GPIO.output(RCpin, False)
    time.sleep(0.1)
    GPIO.setup(RCpin, GPIO.IN)
    while (GPIO.input(RCpin) == False):
        leer += 1
    return leer

#metodos alarma
def sonarAlarma():
        for i in range(10):
```

```
                GPIO.output(pinBuzzer, True)
                time.sleep(0.1)
                GPIO.output(pinBuzzer, False)
                time.sleep(0.1)
def activarAlarma():
        for i in range(50):
                print(i)

                if GPIO.input(pinBoton) == True:
                        GPIO.output(pinVerde, True)
                        print("Alarma desactivada")
                        time.sleep(0.1)
                        break
                elif i >= 49:
                        GPIO.output(pinRojo, True)
                        print("Alarma activada")
                        sonarAlarma()
                time.sleep(0.1)
        GPIO.output(pinVerde, False)
        GPIO.output(pinRojo, False)
while True:
        #detectar humedad
        humedad = random.randint(0, 1000)
```

```
        print("Humedad: ", humedad)


        if humedad < 10: #si la humedad es baja se riega la planta
                print("Regando planta")
                GPIO.output(pinRegado, True)
                time.sleep(5)
                GPIO.output(pinRegado, False)
        #detectar estado del dia
        estado = RCtime(pinLDR)
        if estado < 1000: #identificar si es de dia o de noche
                print("Es de dia ", estado)
        else:
                print("Es de noche", estado)
                #si es de noche se activa el sensor de movimiento
                if GPIO.input(pinPIR) == True: #identificar si existe movimiento
                        print("Se detecto movimiento")


                        #si se detecta movimiento se activa la alarma
                        activarAlarma()
                else:
                        print("No hay movimiento")
        time.sleep(0.5)
GPIO.cleanup() #limpiar GPIO
```

Conclusión:

Esta práctica nos resultó un poco más sencillo que la vez que lo hicimos utilizando Arduino, quizás sea por la versatilidad de la Raspberry o sencillamente por la experiencia que adquirimos en la ocasión pasada.

VI. Sensor Ultrasónico

Descripción:

Durante esta práctica se realizó un sensor que detecta el movimiento de cualquier tipo de objeto, al momento de detectarlo enciende un led.

Componentes:

- Raspberry PI
- Sensor sónico HC-SR04
- 1 leds
- Cables

Principios:

Se utilizó un sensor sónico HC-SR04 para detectar movimientos e imprimir en pantalla cual es la distancia del objeto en cuestión.

Procedimiento:

Paso 1: Unimos la Raspberry y los componentes necesarios para realizar la práctica de la siguiente manera:

Ejemplo en el siguiente video

Paso 2: Se escribe el código necesario(Mismo que se adjuntará en el siguiente punto) para hacer funcionar los componentes.

Paso 3: Compilar el código

Código:

```
import RPi.GPIO as GPIO
import time
from decimal import Decimal

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

#pins
trig = 23
echo = 24
```

```
#entradas y salidas
GPIO.setup(trig, GPIO.OUT)
GPIO.setup(echo, GPIO.IN)

#leer distancia del sensor
def detectar():
    GPIO.output(trig, False)
    time.sleep(2*10**-6) #2 microsegundos
    GPIO.output(trig, True)
    time.sleep(10*10**-6) #10 microsegundos
    GPIO.output(trig, False)

    #contar tiempo
    while GPIO.input(echo) == 0:
        start = time.time()
    while GPIO.input(echo) == 1:
        end = time.time()

    #duracion del pulso
    duracion = end-start

    #convertir a microsegundos
    #duracion = duracion*10**6

    #distacia en cm
    #medida = duracion/58

    distancia = (duracion * 34300)/2
    return Decimal(distancia)

while True:
    distancia = detectar()
    #print(distancia)

    if distancia <= 30:
        print("Objeto cercano", distancia)
    else:
        print("Objeto lejano", distancia)

    time.sleep(0.5)

GPIO.cleanup() #limpiar GPIO
```

Conclusión:
Realizar esta práctica fue una experiencia que consideramos nueva, puesto que después de venir de encender leds en las prácticas anteriores, en esta se interactúa con algo más a doc a lo que es el IoT.

VII. Motor DC

Descripción:
En este proyecto se utilizó una Raspberry para hacer que un motor se moviera presionando un botón.

Componentes:
- Raspberry PI
- Motor DC

Principios:
Conectamos y programamos una Raspberry de manera que el Motor DC se moviera y en consola nos imprimiera la palabra “Prendiendo Motor” una vez que se enciende y “Parando motor” al detenerlo.

Paso 1: Conectamos la Raspberry y los componentes necesarios para realizar la práctica de la siguiente manera:

Ejemplo en el siguiente video

Paso 2: Se escribe el código necesario para hacer funcionar los componentes.

Paso 3: Compilar el código

Código:
```
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BOARD)

Motor = 16


GPIO.setup(Motor,GPIO.OUT)

print "Prendiendo motor"
GPIO.output(Motor,GPIO.HIGH)

sleep(5)
```

```
print "Parando motor"
GPIO.output(Motor,GPIO.LOW)

GPIO.cleanup()
```

Conclusión:

La realización de esta práctica no estuvo tan accidentada como con la placa de Arduino.