Universidad Nacional Autónoma de México
Facultad de Ingeniería
Laboratorio de Fundamentos Embebidos

Práctica 7:
Lighshowpi

Integrantes:
Cárdenas Cárdenas Jorge
Garrido Sánchez Samuel Arturo
Murrieta Villegas Alfonso
Reza Chavarría Sergio Gabriel
Valdespino Mendieta Joaquín

Práctica 7: Lighshowpi

Objetivo:

Implementar el uso de alguna biblioteca como Lighshowpi para el uso y manipulación de tiras leds.

Introducción:

Leds o en español “diodo emisor de luz”, son una tecnología destinada a la iluminación, pero ¿Por qué ha cobrado tanta importancia en las últimas 2 décadas?:

La primera razón es que los leds producen luz aproximadamente un 90% más eficientemente que las bombillas incandescentes y eso se debe a que la corriente eléctrica que pasa a través de un “microchip” ilumina las diminutas fuentes de luz que llamamos leds y el resultado es luz visible.

La segunda razón es la relacionada con el rendimiento respecto al manejo de calor, los leds utilizan disipadores de calor para absorber el calor producido y disipan en el entorno circundante. Esto evita que estos se sobrecalienten y se quemen. La gestión térmica es generalmente el factor más importante en el desempeño exitoso de un led durante su vida útil, cuanto mayor sea la temperatura a la que se operan, más rápidamente se degradará la luz y menor será la vida útil.

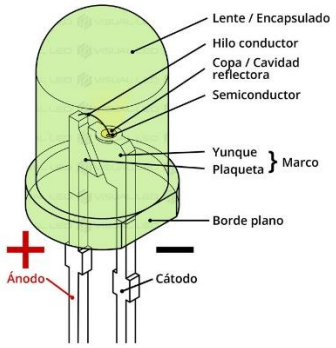

Imagen 1: Estructura de los leds para emitir los leds

En la presente práctica abordaremos una alternativa a Lighshowpi para manejar tiras leds o led strips con el objetivo de controlar a estos mediante una Raspberry Pi.

NOTA:
La razón principal de por qué emplearemos la biblioteca Neopixel es debido a la comodidad que nos ofrece respecto a la manipulación de datos de los diferentes colores y frecuencias en los leds de las tiras.

Desarrollo y resultados:

1. Biblioteca

A continuación, se muestra paso a paso como instalar la biblioteca de neopixel para hacer funcionar una tira led de tipo WS2812:

1. Instalar la biblioteca encargada de traer todos los objetos necesarios como es el caso de neopixel

```
sudo pip3 install rpi_ws281x adafruit-circuitpython-neopixel
```

2. Instalar la biblioteca para el manejo de los leds

```
sudo python3 -m pip install --force-reinstall adafruit-blinka
```

Es importante destacar que, para el funcionamiento correcto de este código, se requiere una tira led de tipo WS2812, pues no todas las tiras leds tienen la misma cantidad de pins además de la forma en que estos funcionan:

A continuación, se muestran los esquemáticos descriptivos de los leds y de las conexiones de nuestra tira led:

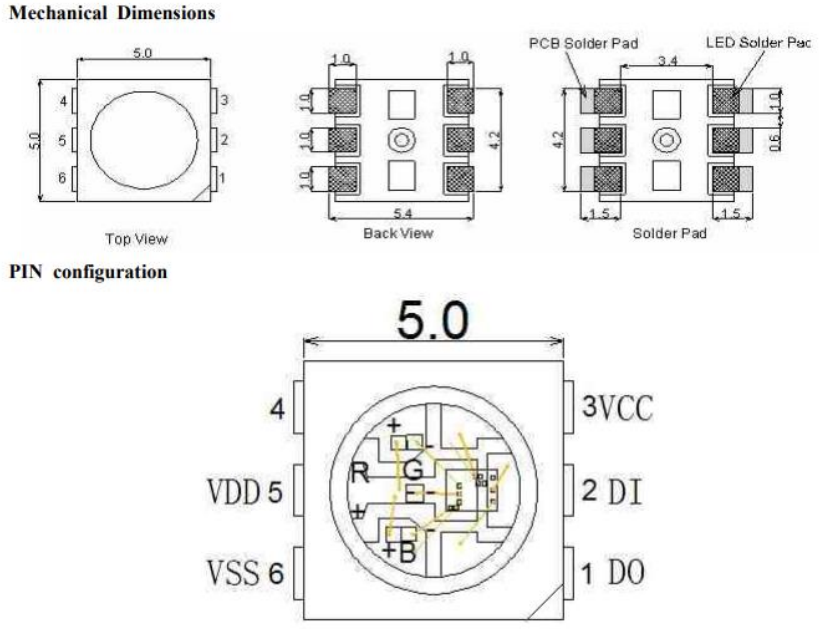

Imagen 2: Estructura de los leds en una tira WS2812

Podemos observar que a diferencia de las tiras tradicionales que emplean 4 puertos (Tierra, y R-G-B), las tiras de este tipo emplean solamente 3 pins, donde uno es la alimentación, otro es tierra (GND) y el de en medio es DIN o Data Input, es decir, es el pin encargado de meter la señal PWM dada por nuestro futuro Raspberry.

La ventaja de este tipo de tira es que dentro de ella tiene un circuito integrado “WS2812” destinado a la interpretación de ese PWM para poder definir la frecuencia y los colores de cada uno de los leds de nuestra tira

2. Conexiones

A continuación, se muestra el diagrama de conexión de nuestra Raspberry Pi respecto a la tira led y la alimentación a la misma:

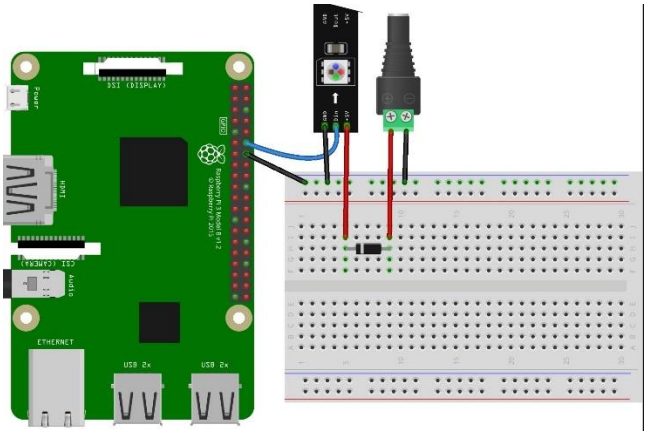

Imagen 3: Diagrama de conexiones para la implementación de nuestro sensor ultrasónico

3. Código

A continuación, se muestra parte a parte el código encargado del funcionamiento de nuestra tira led:

1. Importación de bibliotecas e instancias encargadas de los puertos y configuración de los leds de nuestra tira

```
import time
import board
import neopixel

pixel_pin = board.D18
num_pixels = 30# The number of NeoPixels
ORDER = neopixel.GRB

pixels = neopixel.NeoPixel(
    pixel_pin, num_pixels, brightness=0.2, auto_write=False, pixel_order=ORDER
)
```

2. Funciones para hacer un ciclo respecto a los colores de nuestra tira (Es para dar un efecto arcoíris en nuestra tira)

```
def wheel(pos):
    if pos < 0 or pos > 255:
        r = g = b = 0
    elif pos < 85:
        r = int(pos * 3)
        g = int(255 - pos * 3)
        b = 0
    elif pos < 170:
        pos -= 85
```

```
        r = int(255 - pos * 3)
        g = 0
        b = int(pos * 3)
    else:
        pos -= 170
        r = 0
        g = int(pos * 3)
        b = int(255 - pos * 3)
    return (r, g, b) if ORDER in (neopixel.RGB, neopixel.GRB) else (r, g, b, 0)
```

La función específica de hacer el efecto arcoíris es la siguiente, metemos el arreglo de los leds dentro de un ciclo que simplemente será “infito” para que siempre vaya cambiando el color mediante la función anterior

```
def rainbow_cycle(wait):
    for j in range(255):
        for i in range(num_pixels):
            pixel_index = (i * 256 // num_pixels) + j
            pixels[i] = wheel(pixel_index & 255)
        pixels.show()
        time.sleep(wait)
```

3. Apartado principal encargado de instanciar las funciones anteriores

```
while True:
    rainbow_cycle(0.001)  # rainbow cycle with 1ms delay per step
```

4. Resultados

A continuación, se muestran algunos colores dados por nuestra tira led conectada a nuestra Rapberry, sin embargo, para poder apreciar el efecto visual del arcoíris se recomienda ver el video-evidencia:

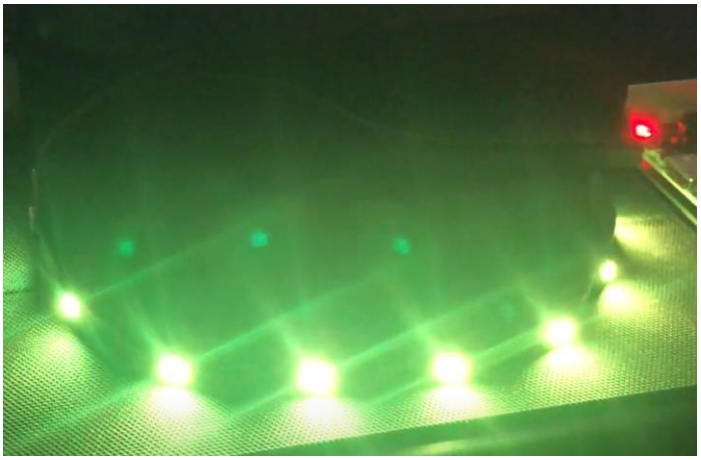

Imagen 4: Tira led ofreciendo un color verde

Conclusiones:

Cárdenas Cárdenas Jorge

En esta práctica realizamos la animación de tiras de leds con los distintos colores que componen el modelo de color RGB desde una Raspberry. Se trató de una práctica muy didáctica y entretenida que nos permitió interactuar con otro tipo de bibliotecas diferentes a las que hasta ahora habíamos venido usando, así como con elementos de hardware un tanto diferente al que incluso en otras materias habíamos usado como los típicos motores, cuyos controles los habíamos implementado usando desde una FPGA hasta un PIC.

El manejo de las tiras de led mediante el uso de señales PWM me pareció realmente interesante, sobre todo porque no pensé que fuese así como funcionaran. Igual cabe destacar que la sencillez con que desde Python se pueden generar este tipo de señales es un factor que facilita enormemente la programación, así como la interacción con estos elementos.

Garrido Sánchez Samuel Arturo

Mediante el uso de pwms presentes en algunos puertos gpio de la raspberry podremos controlar y mandar señales de información a elementos de hardware que se puedan comportar de formas distintas dependiendo de la información que se envíe. La biblioteca de neopixel nos permitió controlar las luces mediante un tercer canal donde podemos enviar un tren de pulsos con diferente ciclo útil, ya sea 25%, 50%, 75%, del cuál se encarga de gestionar internamente la biblioteca. Ahora, las luces necesitan que esta entrada sea de 3.3V ya que es lo máximo que puede mandar el Raspberry, por lo que se dificulta conseguir un juego de luces con tales características. Este control de pwm también es utilizado para llevar direcciones y velocidades en motores, ya sean pservos o simples de DC, el controlador recibe esta señal en EN para ajustar las velocidades o direcciones. Finalmente estos elementos nos servirán en gran medida para los sensores de nuestro proyecto al ajustar divesos parámetros de salida.

Murrieta Villegas Alfonso

En la presente práctica aprendimos a emplear una de las bibliotecas más conocidas dentro del mundo de los leds en Raspberry, sin duda, fue más que sorprendente que hoy en día puedan existir circuitos integrados dentro de las mismas tiras que hagan posible la traducción de PWM dado por nuestra biblioteca-Raspberry.

Por otro lado, aprendimos que siempre existen alternativas de código y hardware, lo cual es sobre todo bueno para no depender de un solo recurso incluso nos da una gran versatilidad al momento de poder comparar precios, eficiencia energética e incluso comodidad en el aspecto de desarrollo de software.

Por último, recordamos aspectos importantes en el manejo de señales a través de PWM que es sobre todo un recurso más que necesario para poder interactuar con una capa de hardware.

Reza Chavarria Sergio Gabriel

Mediante la practica comprendimos el manejo de salidas complejas en el ámbito de los sistemas embebidos. Nos permiten tener mayor versatilidad en la implementación del uso de Leds.

El manejo de Python para obtener la implementación facilito en el manejo de esto. El comportamiento de tiras Led puede ser implementado en la personalización de los sistemas dependiendo del usuario, emisión de múltiples alertas o simplemente un atractivo visual para llamar la atención de personas interesadas en los productos.

Valdespino Mendieta Joaquín

En la presente practica aprendimos a usar una biblioteca con un uso especifico, pero en un campo amplio actualmente, que es para el manejo de luces LED, que prácticamente existen en muchos productos de moda, más en artículos gaming, con las famosas tiras LED RGB, que son circuitos integrados, y modulares que nos permiten realizar diferentes aplicaciones, más allá de ser decorativos, esto aclarando mediante la biblioteca de rapsberry y el uso de señales PWM.

Referencias:

1. Worldsemi. *Intelligent Control LED*. Recuperado el 20 de octubre de 2021, de www.i-enet.com (adafruit.com)

2. Adafruit. *Welcome To CircuitPython.* Recuperado el 20 de octubre de 2021, Welcome To CircuitPython | Welcome to CircuitPython! | Adafruit Learning System