PRÁCTICA IR-RASPBERRY PI PICO W
MICROCONTROLADORES

Universidad Distrital Francisco José de Caldas
Proyecto Curricular de Ingeniería electrónica
14 de febrero de 2024
Bogotá DC, Colombia

Juan Camilo Rodriguez Garcia - 20202005070

Juan Sebastian Casas Barbosa - 20211005031

Haider Santiago Calderón Rodríguez - 20211005075

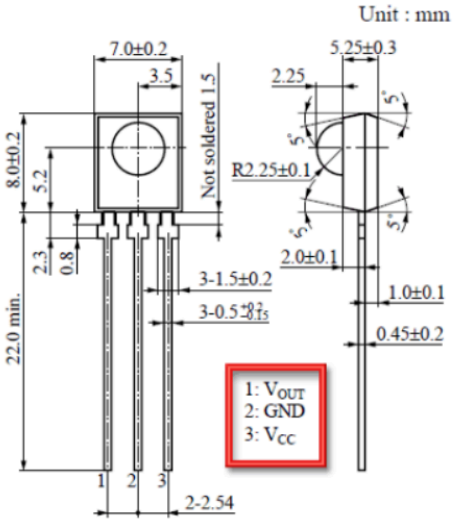
I. INTRODUCCIÓN

En esta práctica, se utilizó un circuito receptor infrarrojo (IR) junto con una Raspberry Pi Pico W para capturar señales infrarrojas y decodificarlas en un número binario. El circuito receptor IR captura las señales de un control remoto IR y las convierte en pulsos eléctricos. Estos pulsos son luego interpretados por la Raspberry Pi Pico utilizando una máquina de estados programable (PIO) para determinar el número asociado a la señal recibida.

El código utilizado define un programa en ensamblador para la PIO que controla la interpretación de las señales recibidas. El programa utiliza lógica para distinguir entre pulsos de encendido y apagado, y luego convierte estos patrones en un valor binario. Este valor binario se compara con un diccionario predefinido para determinar el número asociado a la señal recibida, que se imprime como salida del programa.

II. MARCO TEÓRICO

Los sensores infrarrojos (IR) son dispositivos que pueden detectar la radiación infrarroja emitida por objetos cercanos. Estos sensores son comúnmente utilizados en aplicaciones de control remoto, detección de movimiento y sistemas de seguridad. En el caso del sensor IR TSSP4038, su salida se activa (se pone a nivel alto) cuando detecta una señal de luz infrarroja que parpadea a una frecuencia de 38 kHz. Esta frecuencia de parpadeo es común en muchos dispositivos de control remoto, como los mandos a distancia de televisores y equipos de audio.



En un mando a distancia por infrarrojos, cada tecla está asociada con una secuencia única de pulsos infrarrojos. Estos pulsos representan los bits de información que se transmiten al dispositivo receptor. Por ejemplo, el cero lógico y el uno lógico pueden estar representados por diferentes duraciones de pulsos infrarrojos. La secuencia de pulsos enviada por una tecla específica es conocida como su "código de control remoto" y es interpretada por el receptor IR para realizar una acción específica, como cambiar de canal en un televisor o ajustar el volumen de un equipo de audio.

La Raspberry Pi Pico se puede utilizar como receptor de señales IR al conectar un sensor IR adecuado a sus pines GPIO. El programa en Python que se ejecuta en la Raspberry Pi Pico, junto con la configuración de la máquina de estados programable (PIO), permite interpretar los pulsos infrarrojos recibidos y decodificarlos en los comandos correspondientes. Esto facilita la integración de la Raspberry Pi Pico en sistemas de control remoto y otros proyectos que requieran la recepción y decodificación de señales IR.

III. DESARROLLO

Para la realizacion de la practica se partio del siguiente codigo en ensamblador el cual corresponde a una ASM mediante la cual se programa el sensor VS1838 con uno de los pines de salida de la Raspberry Pi Pico W la cual hace uso del microcontrolador RP2040, asi entonces el codigo es el siguiente:

```
import time
import rp2
from machine import Pin

# Lista para almacenar los valores
data_list = []

@rp2.asm_pio(set_init=rp2.PIO.OUT_LOW)
def blink():
    label('inicio')
    mov(x,invert(null))
    jmp(pin,'uno')
    label('cero')
    set(pins, 0)
    jmp(x_dec,'cero_bis')
    jmp('fin')
    label('cero_bis')
    jmp(pin,'fin')
    jmp('cero')
    label('uno')
    set(pins, 1)
    jmp(x_dec,'uno_bis')
    jmp('fin')
    label('uno_bis')
    nop()
    jmp(pin,'uno')
    label('fin')
    mov(isr,x)
    push(noblock)
    jmp('inicio')

# Inicializa el StateMachine y lo activa
sm = rp2.StateMachine(
    0,
    blink,
    freq=38000*100,
    set_base=Pin(0),
    jmp_pin= Pin(17, Pin.IN, Pin.PULL_UP)
)
sm.active(1)
```

En el código se importan las librerías

- **time:** proporciona varias funciones relacionadas con el tiempo

- **rp2:** contiene funciones y clases específicas para el microcontrolador RP2040, como las que se usan en la Raspberry Pi Pico, este módulo cuenta con funciones para ensamblar programas PIO

y se importa **Pin** de **machine** el cual sirve para poder usar los pines del microcontrolador.
Así mismo se crea una lista llamada **data_list** en la cual se van a almacenar los datos de la siguente manera:

```
import time
import rp2
from machine import Pin
```

```
# Lista para almacenar los valores
data_list = []
```

Por otro lado el codigo en micropython utiliza de E/S (entradas y salidas) programable (PIO) del RP2040 de modo que:

```
@rp2.asm_pio(set_init=rp2.PIO.OUT_LOW)
```

Esta linea corresponde a un decorador especial para la funcion de ensamblador en micropython donde **set_init=rp2.PIO.OUT_LOW** configura el estado inicial de salida del pin de salida como bajo.

Asi entonces en las siguientes lineas de codigo

```
sm = rp2.StateMachine(
    0,
    blink,
    freq=38000*100,
    set_base=Pin(0),
    jmp_pin= Pin(17, Pin.IN, Pin.PULL_UP)
)
sm.active(1)
```

Se inicializa y activa el StateMachine (sm) del RP2040. Se le asigna el programa definido **blink**, se configura una frecuencia de 38KHz que corresponde a la señal que sale del emisor (control IR) que manda la informacion con una portadora de 38KHz y que por consiguiente corresponde a la frecuencia a del recepción del sensor VS1838. Se especifica el pin de salida (set_base) y el pin de entrada (jmp_pin = GPI17)

Asi entonces antes de inicializar la maquina de estados da la descripcion del programa que se va a asignar a la misma que corresponde a un codigo en ensamblador el cual hace uso de las siguientes funciones del PIO

- **label(label)**: Define una etiqueta en la ubicación actual, puede ser un entero o un string

- **jmp(...)**: una instruccion con dos formas
    - **jmp(label)**: salta a la etiqueta label incondicionalmente
    - **jmp(cond,label)**: salta a label, dependiendo de la condicion cond, que hay de varios tipos pero las utilizadas dentro de esta máquina de estados son:
        - **x_dec**: verdad si el registro no es cero y hace un decremento
        - **pin**: verdadera si el pin está cargado
- **mov(dest,src)**: mueve a dest el valor de src
- **set(dest,data)**: carga dest con el valor de data donde data puede ser un valor entre (0-31)

De este modo la máquina de estados se puede explicar mejor y ver su funcionamiento a partir de un diagrama ASM teniendo clara cada una de las funciones usadas del PIO de la siguiente manera:

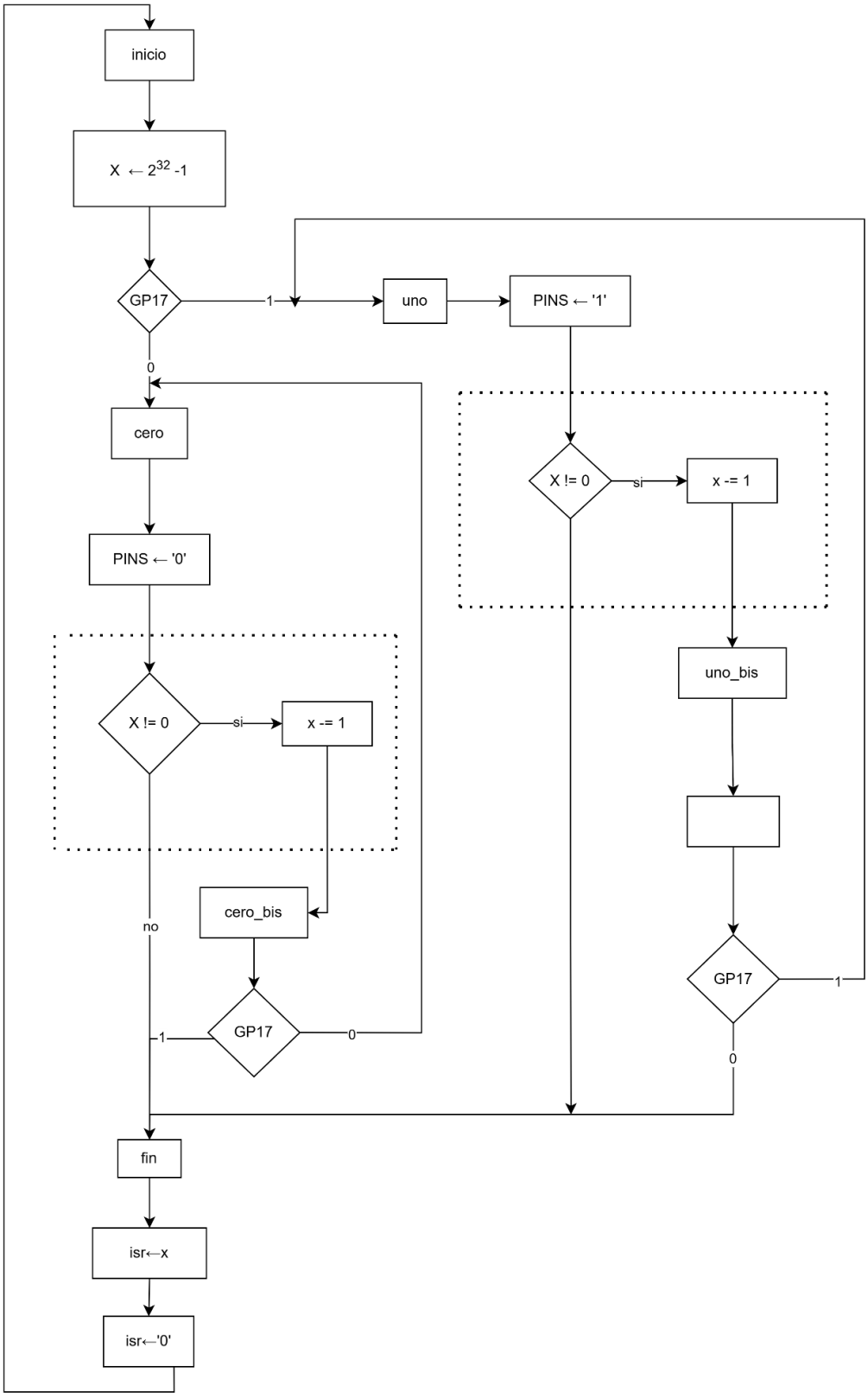


Fig #. Diagrama ASM Recepción IR

Teniendo el código para la recepción de los datos haciendo uso del sensor se creó un código mediante el cual se guarda en la lista ya mencionada los datos recogidos y estos se manipulan para poder obtener un

código hexadecimal y posteriormente llevar esa codificación a el reconocimiento de cada uno de los números del control IR para imprimirlos en la consola, así entonces se partió para la práctica con el número 3, haciendo uso del código anterior se recopilaron los datos de oprimir 10 veces el botón número 3 del control en intervalos de aproximadamente 2 segundos obteniendo así los siguiente:

1378155, 9328, 4306, 603, 517, 630, 490, 605, 516, 633, 487, 635, 485, 629, 491, 631, 489, 629, 491, 635, 1547, 635, 1546, 634, 1547, 631, 1550, 633, 1547, 634, 1548, 632, 1549, 605, 1576, 629, 491, 637, 1545, 632, 1549, 631, 1549, 627, 1552, 605, 515, 632, 1548, 636, 486, 610, 1573, 606, 518, 607, 518, 625, 500, 643, 484, 610, 1576, 633, 486, 611, 1570, 636, 38134, 9333, 2097, 640, 1293415, 9304, 4281, 637, 488, 634, 491, 633, 491, 607, 518, 636, 489, 633, 491, 634, 491, 631, 494, 632, 1554, 634, 1551, 631, 1555, 606, 1579, 606, 1580, 629, 1556, 631, 1555, 635, 1551, 629, 495, 635, 1551, 606, 1579, 607, 1579, 606, 1579, 630, 494, 606, 1580, 605, 519, 632, 1552, 636, 488, 630, 493, 634, 490, 629, 494, 630, 1555, 636, 487, 635, 1550, 607, 38173, 9273, 2134, 629, 1280626, 9252, 4275, 634, 487, 634, 487, 630, 490, 639, 483, 637, 485, 635, 487, 638, 485, 639, 484, 640, 1544, 640, 1545, 614, 1571, 643, 1543, 643, 1543, 636, 1548, 642, 1543, 639, 1545, 639, 485, 637, 1547, 642, 1543, 641, 1544, 639, 1546, 619, 505, 615, 1570, 637, 488, 640, 1547, 639, 485, 639, 486, 641, 484, 616, 508, 637, 1547, 640, 486, 640, 1546, 639, 38148, 9235, 2133, 638, 1295017, 9284, 4306, 644, 482, 640, 484, 640, 485, 616, 508, 638, 487, 637, 487, 640, 484, 616, 509, 638, 1547, 619, 1567, 638, 1547, 617, 1568, 614, 1570, 640, 1545, 637, 1548, 636, 1549, 613, 510, 639, 1545, 640, 1544, 637, 1548, 635, 1549, 635, 488, 639, 1544, 639, 485, 639, 1544, 637, 486, 638, 485, 638, 485, 638, 484, 639, 1545, 637, 486, 637, 1547, 637, 38152, 9286, 2105, 634, 1254348, 9353, 4282, 611, 516, 640, 487, 639, 488, 636, 490, 608, 519, 636, 490, 639, 488, 639, 487, 634, 1553, 633, 1555, 610, 1578, 636, 1551, 611, 1576, 609, 1579, 634, 1553, 608, 1580, 636, 489, 615, 1573, 639, 1548, 639, 1549, 608, 1578, 636, 490, 612, 1574, 641, 486, 636, 1550, 635, 491, 616, 509, 640, 485, 639, 486, 641, 1546, 640, 485, 643, 1543, 638, 38152, 9279, 2109, 634, 1309297, 9298, 4303, 646, 481, 644, 482, 641, 486, 638, 489, 643, 483, 645, 482, 613, 516, 639, 488, 643, 1544, 645, 1543, 642, 1546, 642, 1545, 644, 1543, 641, 1546, 624, 1563, 640, 1548, 619, 507, 638, 1550, 647, 1540, 640, 1546, 616, 1571, 642, 483, 637, 1549, 641, 484, 617, 1569, 638, 487, 640, 483, 640, 485, 638, 486, 638, 1546, 639, 485, 643, 1542, 615, 38167, 9290, 2105, 640, 1309543, 9296, 4278, 639, 486, 646, 479, 634, 492, 643, 482, 642, 485, 640, 486, 644, 482, 644, 482, 639, 1548, 640, 1546, 640, 1545, 644, 1543, 639, 1547, 638, 1548, 640, 1546, 639, 1546, 638, 486, 636, 1548, 635, 1550, 611, 1574, 636, 1548, 635, 489, 639, 1545, 637, 487, 635, 1549, 634, 489, 616, 507, 636, 487, 608, 515, 614, 1570, 614, 510, 641, 1543, 613, 38175, 9229, 2133, 613, 1242392, 9324, 4273, 639, 486, 633, 491, 638, 487, 638, 487, 637, 487, 635, 489, 634, 490, 637, 486, 639, 1546, 611, 1574, 638, 1547, 637, 1548, 638, 1548, 638, 1547, 637, 1549, 638, 1547, 640, 485, 611, 1575, 638, 1547, 617, 1569, 641, 1544, 637, 488, 639, 1546, 613, 512, 611, 1574, 633, 490, 635, 489, 639, 485, 639, 484, 615, 1569, 609, 514, 633, 1550, 607, 38172, 9246, 2131, 606, 1323732, 9318, 4271, 638, 486, 638, 486, 640, 485, 638, 486, 641, 483, 638, 487, 641, 483, 639, 486, 637, 1548, 644, 1542, 640, 1546, 644, 1542, 641, 1544, 618, 1567, 638, 1547, 633, 1551, 639, 484, 641, 1544, 637, 1546, 614, 1570, 637, 1547, 614, 509, 637, 1546, 616, 507, 637, 1547, 637, 485, 634, 489, 638, 484, 637, 485, 636, 1548, 612, 511, 637, 1547, 635, 38108, 9270, 2133, 640, 1365060, 9295, 4274, 637, 489, 641, 483, 620, 504, 636, 488, 638, 484, 614, 509, 637, 485, 633, 489, 633, 1550, 637, 1547, 611, 1572, 612, 1571, 634, 1550, 638, 1545, 635, 1547, 635, 1548, 636, 486, 610, 1573, 608, 1576, 612, 1571, 608, 1576, 615, 508, 639, 1545, 642, 484, 638, 1548, 636, 487, 636, 486, 638, 484, 637, 485, 639, 1545, 642, 483, 641, 1545, 640, 38068, 9315, 2104, 640,

A partir de estos datos se realizó el siguiente código

```
valores = [Numeros pagina anterior]
listas_separadas = [[]]
for valor in valores:
    if valor > 8000:
        listas_separadas.append([])
    listas_separadas[-1].append(valor)

print("Mostramos las listas generadas",listas_separadas)
resultado = [lista for lista in listas_separadas if len(lista) == 67]
print("Mostramos la lista de listas depurada en tamaño",resultado)
tamanos = [len(lista) for lista in resultado]
print("Mostramos el tamaño de listas que estamos trabajando",tamanos)
```

En este código lo que se hace es separar los valores en los datos de cada pulsacion del boton numero 3, donde la separación se da cuando hay valores mayores a 8000 que corresponde al tiempo de espera mientras se realizaba la otra pulsación del botón, así entonces estos valores se guardan en una lista de listas y por último se imprime el tamaño de cada una de las listas.

Posteriormente se graficaron estos resultados para la corrección de errores haciendo uso del siguiente código:

```
import matplotlib.pyplot as plt
def plot(datos):
  plt.rcParams['figure.figsize'] = [20, 3]
  x_val=list(range(len(datos[0])))
  fig, ax = plt.subplots()
  for lista in datos:
      ax.plot(lista)
  plt.xticks(x_val)
  plt.show()

plot(resultado)
plot([lista[2:66] for lista in resultado])
```

Obteniendo lo siguiente:

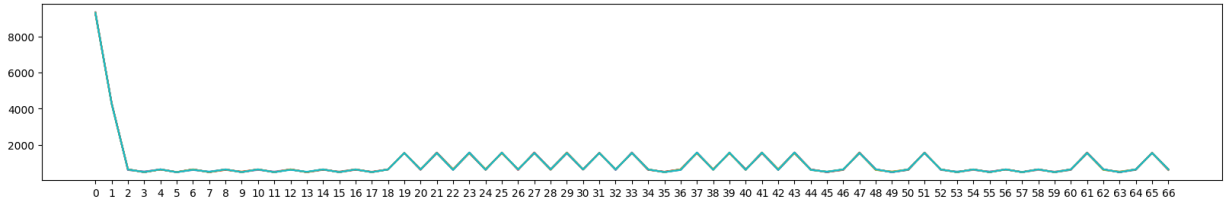


Fig #. Grafica datos IR

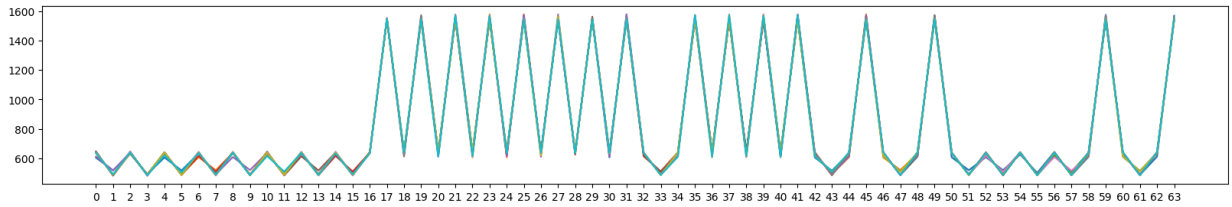


Fig #. Grafica datos IR ZOOM

A partir de estos datos se realizó el siguiente código mediante el cual se hace el procesamiento de datos mediante arreglos de listas

---

```
# Obtener el número total de listas en la lista
num_listas = len(resultado)

# Obtener el número total de elementos en cada lista
num_elementos = len(resultado[0])

# Inicializar una lista para almacenar los promedios
promedios = [0] * num_elementos

# Calcular la suma de cada elemento en todas las listas
for lista in resultado:
    for i in range(num_elementos):
        promedios[i] += lista[i]

# Dividir la suma de cada elemento por el número de listas para obtener el
promedio
for i in range(num_elementos):
    promedios[i] /= num_listas

print("Mostramos el promedio de cada posición de las listas",promedios)

valores_posiciones_pares = [promedios[i] for i in range(len(promedios)) if i %
2 != 0][1:]

print("Los valores de interés son",valores_posiciones_pares)
print("El tamaño del arreglo es de",len(valores_posiciones_pares))

binario=[('1' if valores_posiciones_pares[i] > 600 else '0')  for i in
range(32) ]
binario=''.join(binario)
x_5=binario
print("El código en binario es",binario)

cod=[int(binario[i*8 : (i+1)*8],2)  for i in range(4)]
```

```
print("El código en entero es",cod)
hexad=[hex(val) for val in cod]
print("El código en hexadecimal es",hexad)
```

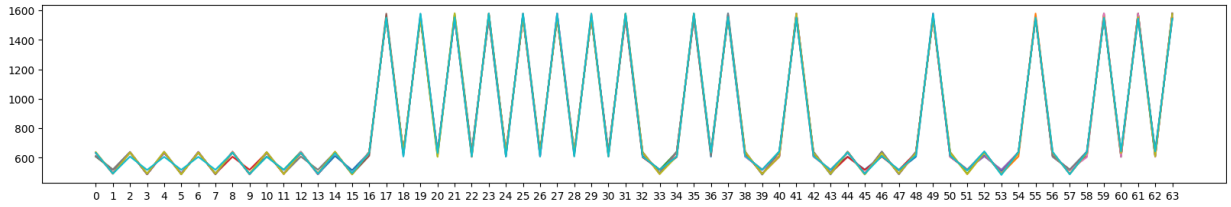
En la consola los códigos en Binario, Decimal y Hexadecimal son los siguientes

```
El código en binario es 00000000111111110011100011000111
El código en entero es [0, 255, 56, 199]
El código en hexadecimal es ['0x0', '0xff', '0x38', '0xc7']
```

Obteniendo así el código en hexadecimal para el 3

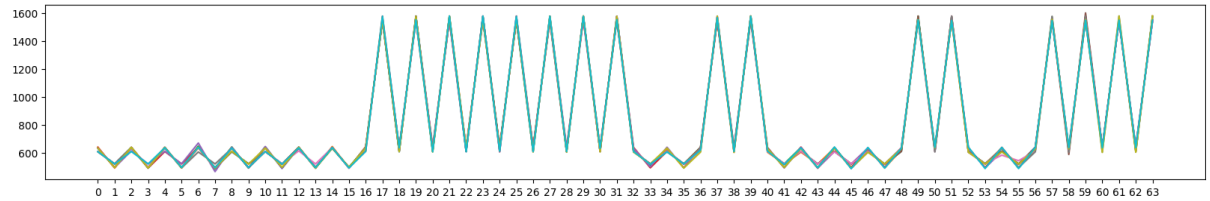
De este modo tendiendo el método para poder obtener una codificación de los botones del control IR y poder mostrar en consola el valor del botón que se oprime se procedió a hacer este procedimiento para cada uno de los botones del control obteniendo las gráficas y los códigos para cada uno de los botones como se muestra a continuación:

Botón 0



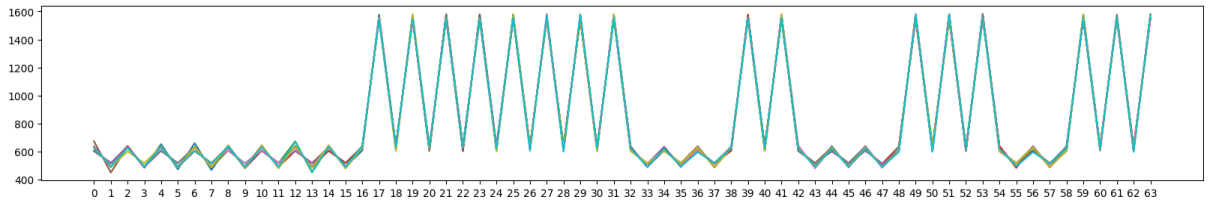

```
El código en binario es 00000000111111110110100010010111
El código en entero es [0, 255, 104, 151]
El codigo en hexadecimal es ['0x0', '0xff', '0x68', '0x97']
```

Botón 1



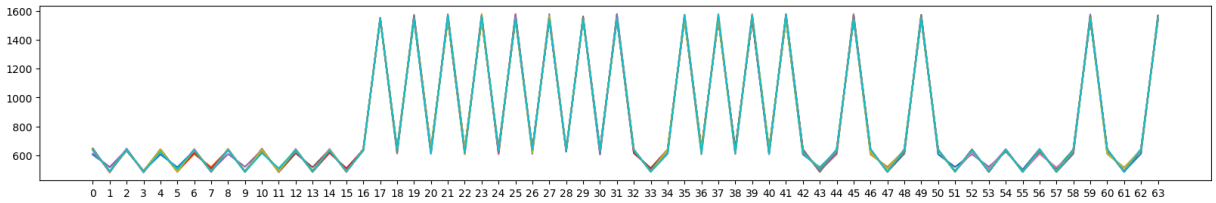

```
El código en binario es 00000000111111110011000011001111
El código en entero es [0, 255, 48, 207]
El código en hexadecimal es ['0x0', '0xff', '0x30', '0xcf']
```

Botón 2



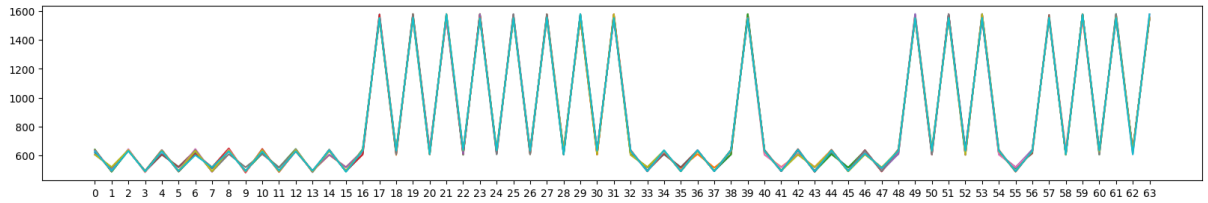
```
El código en binario es 00000000111111110001100011100111
El código en entero es [0, 255, 24, 231]
El código en hexadecimal es ['0x0', '0xff', '0x18', '0xe7']
```

Botón 3



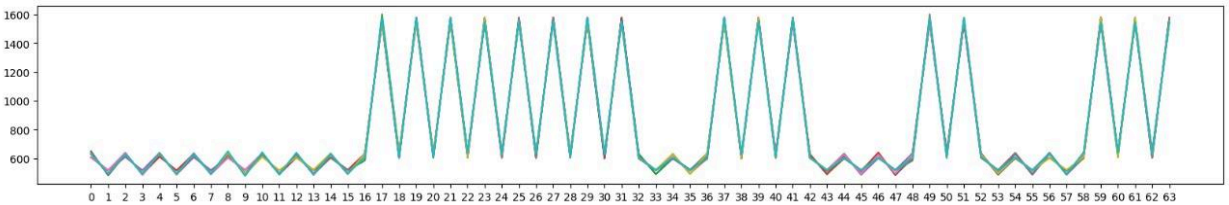
```
El código en binario es 00000000111111110111101010000101
El código en entero es [0, 255, 122, 133]
El código en hexadecimal es ['0x0', '0xff', '0x7a', '0x85']
```

Botón 4



```
El código en binario es 00000000111111110001000011101111
El código en entero es [0, 255, 16, 239]
El código en hexadecimal es ['0x0', '0xff', '0x10', '0xef']
```

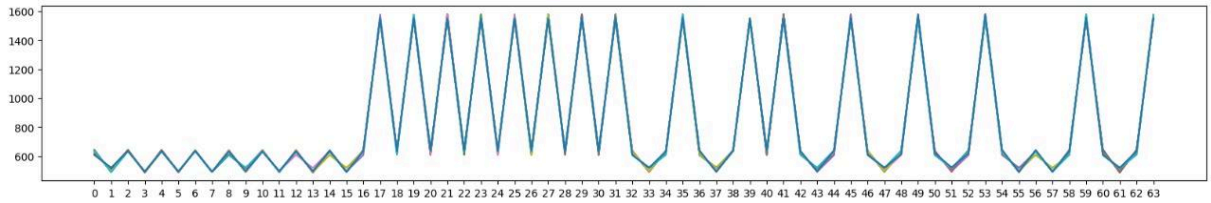
Botón 5



```
El código en binario es 00000000111111110011100011000111
El código en entero es [0, 255, 56, 199]
```

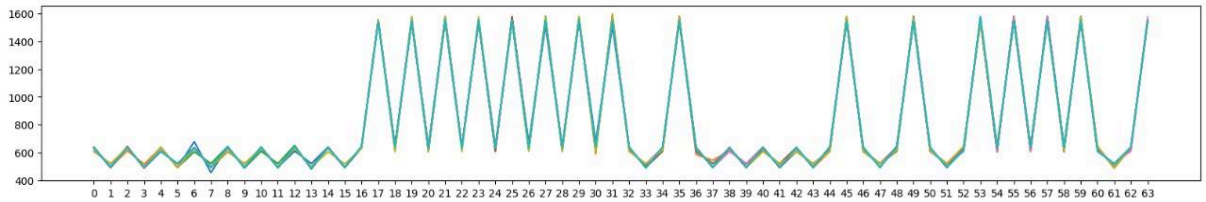
El código en hexadecimal es ['0x0', '0xff', '0x38', '0xc7']

Botón 6



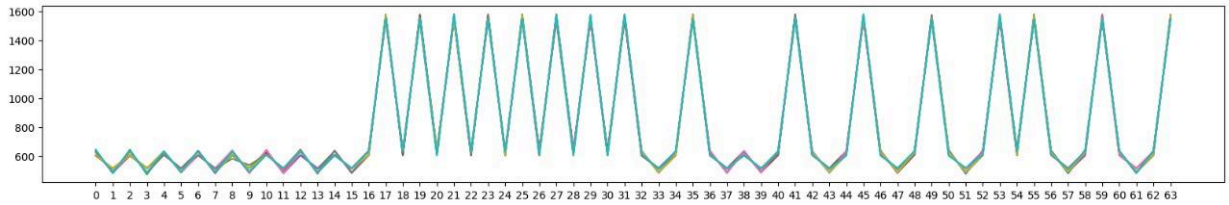
El código en binario es 00000000111111110101101010100101
El código en entero es [0, 255, 90, 165]
El código en hexadecimal es ['0x0', '0xff', '0x5a', '0xa5']

Botón 7



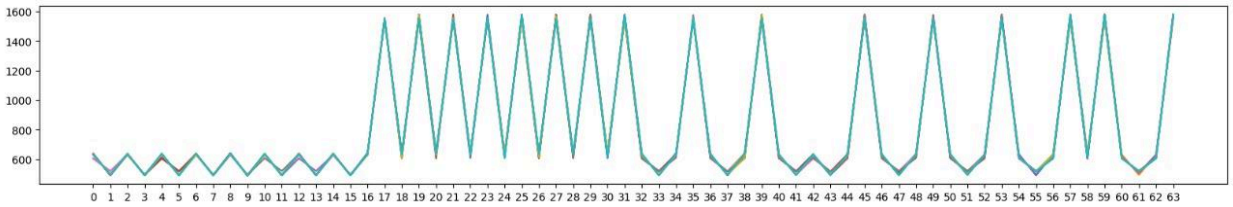
El código en binario es 00000000111111110100001010111101
El código en entero es [0, 255, 66, 189]
El código en hexadecimal es ['0x0', '0xff', '0x42', '0xbd']

Botón 8



El código en binario es 00000000111111110100101010110101
El código en entero es [0, 255, 74, 181]
El código en hexadecimal es ['0x0', '0xff', '0x4a', '0xb5']

Botón 9



```
El código en binario es 00000000111111110101001010101101
El código en entero es [0, 255, 82, 173]
El código en hexadecimal es ['0x0', '0xff', '0x52', '0xad']
```

Con estos resultados se optó por utilizar los códigos binarios para codificar lo que enviaba el control IR y así mediante un código en python mostrar el valor numérico correspondiente al botón oprimido, de la siguiente manera:

```
# Bucle para obtener datos y almacenarlos en la lista
while True:
    while True:
        dat = sm.get()
        data_list.append(2**32 - dat)
        # Condición de salida del bucle
        if len(data_list) >= 72:
            break

    data_list.pop(0)
    #print(data_list)

    valores_posiciones_pares = [data_list[i] for i in range(len(data_list)) if i % 2 != 0][1:]

    binario = [('1' if valores_posiciones_pares[i] > 800 else '0')  for i in range(32) ]
    binario = ''.join(binario)

    mapeo = {
    "00000000111111110110100010010111": 0,
    "00000000111111110011000011001111": 1,
    "00000000111111110001100011100111": 2,
    "00000000111111110111101010000101": 3,
    "00000000111111110001000011101111": 4,
    "00000000111111110011100011000111": 5,
    "00000000111111110101101010100101": 6,
    "00000000111111110100001010111101": 7,
    "00000000111111110100101010110101": 8,
    "00000000111111110101001010101101": 9
    }
```

```
# Obtener el número correspondiente al valor binario utilizando el método
get
    numero = mapeo.get(binario,"Sobrecarga")

# Imprimir el resultado
    print(numero)
    valores_posiciones_pares.clear()
    data_list.clear()
```

Este código recibe los datos y los guarda en una lista, posteriormente elimina el primer valor de esta, ya que corresponde al tiempo muerto en el que no se pulsa el botón del control IR, ya con esto se obtienen los valores de las posiciones pares y el código binario y luego se compara este valor obtenido con los valores binarios obtenidos con el método ya mencionado, si los códigos binarios coinciden se imprime en la consola el número presionado, de lo contrario se imprimirá sobrecarga.

IV. CONCLUSIONES

La codificación de las señales de los botones del control remoto fue exitosa, permitiendo interpretar las señales infrarrojas recibidas. Mediante un código en Python, se logró decodificar estas señales y mostrar en pantalla los valores correspondientes a las teclas presionadas.

V. REFERENCIAS

rp2 — functionality specific to the RP2040 — MicroPython latest documentation