Presentation of the introductory workshop to the

RPI Pico W

Gerardo Muñoz

gmunoz@udistrital.edu.co

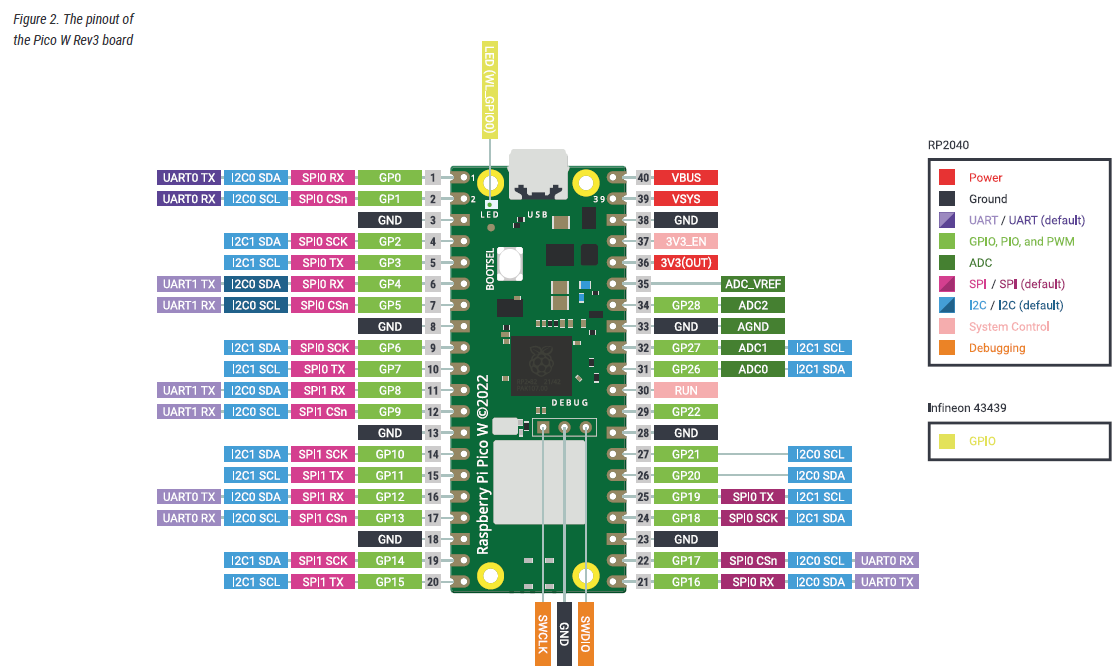
UDFJC

Bogotá, Colombia

Feb. 2024

Raspberry Pico W

<https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf>



Raspberry Pi Pico W has been designed to be a low cost yet flexible development platform for RP2040, with a 2.4GHz

wireless interface and the following key features:

**• Dual-core cortex M0+ at up to 133MHz**

**• RP2040 microcontroller with 2MB of flash memory**

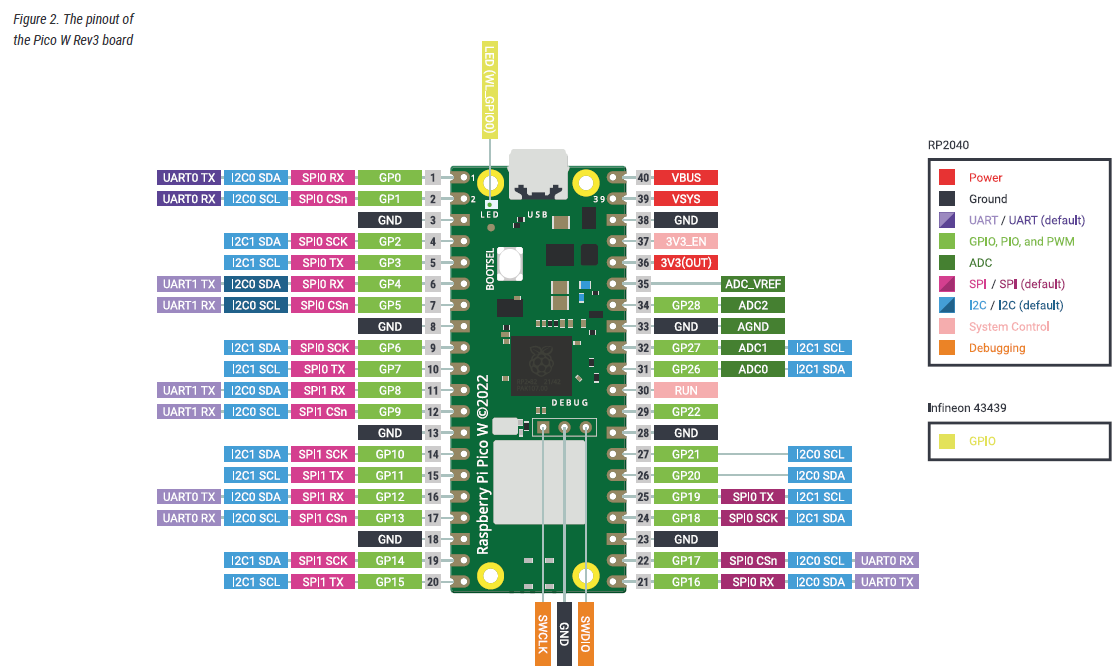
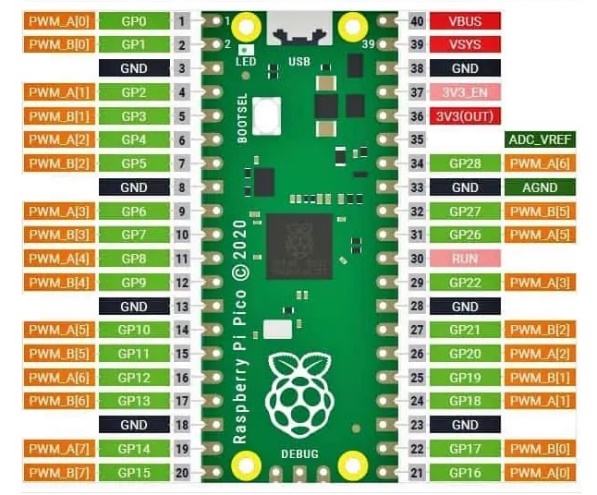
**• On-board single-band 2.4GHz wireless interfaces (802.11n, Bluetooth 5.2)**

**◦ Exposes 26 multi-function 3.3V general purpose I/O (GPIO)**

**◦ 23 GPIO are digital-only, with three also being ADC capable**

**◦12-bit 500ksps analogue to digital converter (ADC)**

**◦ 2 × UART, 2 × I2C, 2 × SPI, 16 × PWM channels**



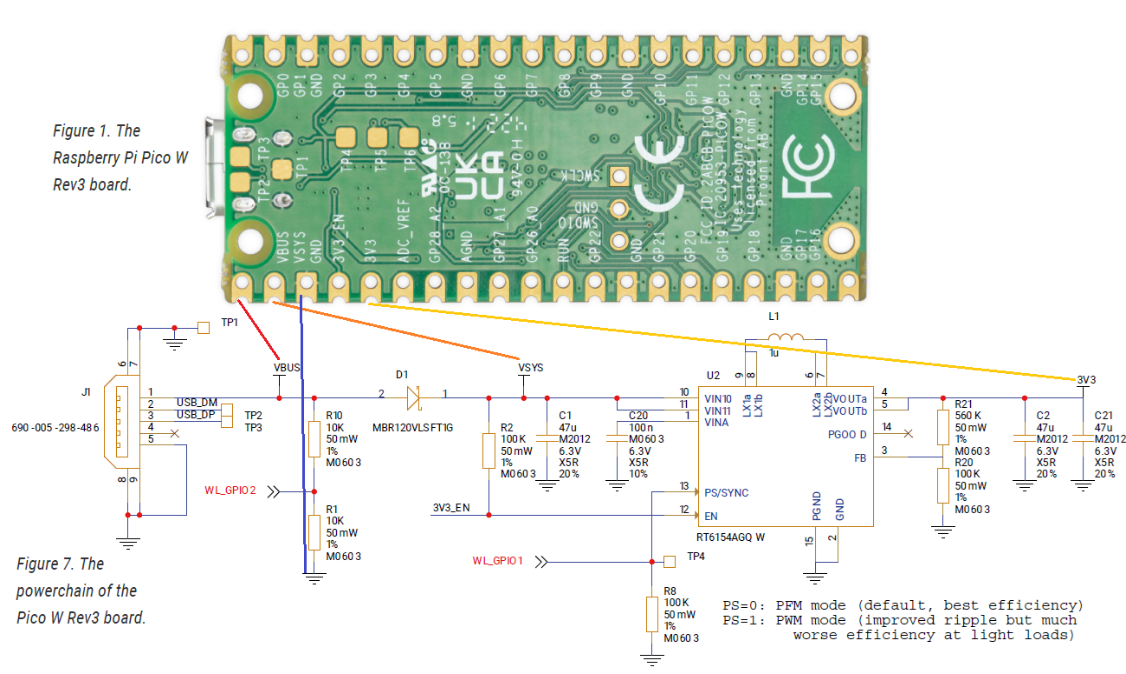
**◦ 1 × timer with 4 alarms, 1 × real time clock**

**• 2 × programmable I/O (PIO) blocks, 8 state machines in total**

◦ Flexible, user-programmable high-speed I/O

◦ Can emulate interfaces such as SD card and VGA

<https://docs.micropython.org/en/latest/rp2/quickref.html>

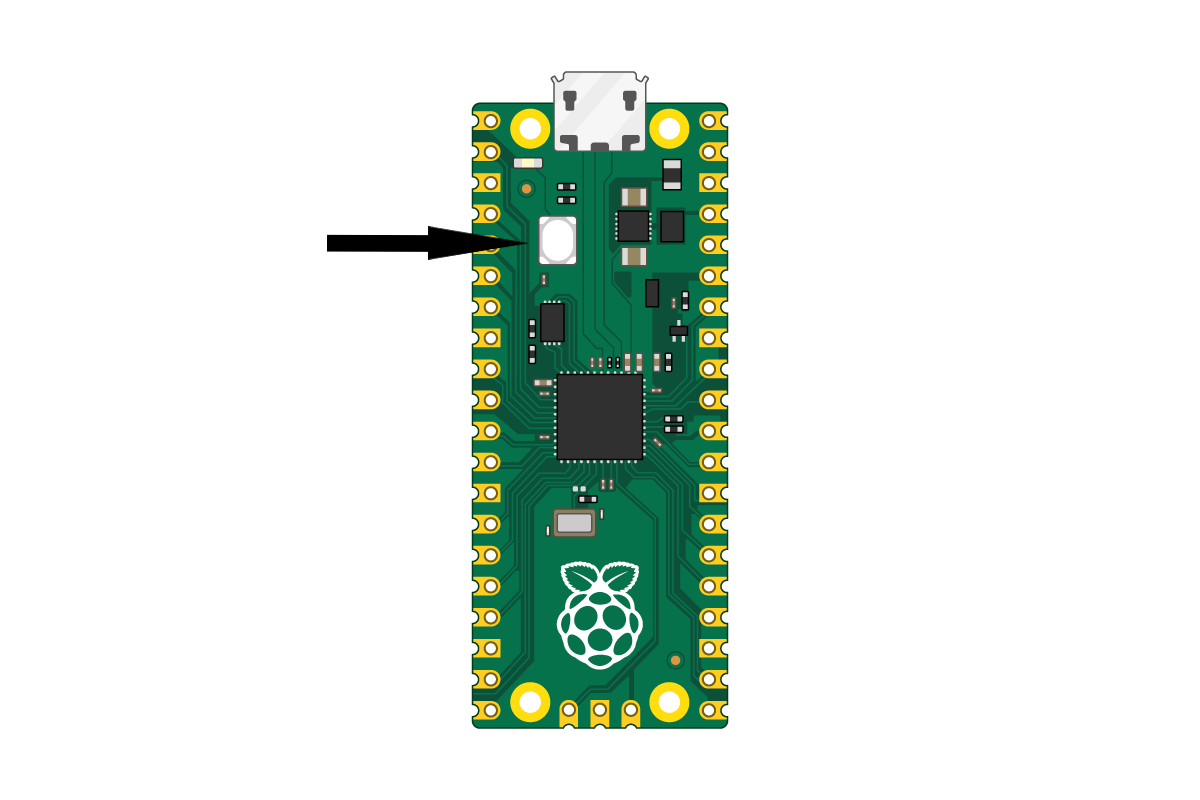


Operating conditions for the Pico W are largely a function of the operating conditions specified by its components.

* **Operating Temp Max** 70°C (including self-heating)
* **Operating Temp Min** -20°C
* **VBUS** 5V ± 10%.
* **VSYS Min** 1.8V
* **VSYS Max** 5.5V

Add the MicroPython firmware

<https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/3>



The onboard LED

https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/5

from machine import Pin

led = Pin(“LED”, Pin.OUT)

led(1)

from machine import Pin, Timer

led = Pin(“LED”, Pin.OUT)

timer = Timer()

def blink(timer):

led.toggle()

timer.init(freq=2.5, mode=Timer.PERIODIC, callback=blink)

Use digital inputs and outputs

https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/6

from machine import Pin

pin\_a = Pin(16, Pin.IN, Pin.PULL\_UP)

print(pin\_a())

from machine import Pin

pin\_a = Pin(16, Pin.IN, Pin.PULL\_UP)

pin\_b = Pin(17, Pin.IN, Pin.PULL\_UP)

led = Pin("LED", Pin.OUT)

def callback(pin):

    # led = a xor b

    led(pin\_a() ^ pin\_b())

pin\_a.irq( handler=callback)

pin\_b.irq( handler=callback)

callback(None)

print("Ok")

from machine import Pin

import time

led = Pin(15, Pin.OUT)

button = Pin(14, Pin.IN, Pin.PULL\_DOWN)

while True:

if button.value():

led.toggle()

time.sleep(0.5)

PIO (advanced, optional)

<https://colab.research.google.com/github/GerardoMunoz/Curso_Python/blob/main/Comparaci%C3%B3n_while_IRQ_PIO.ipynb>

File

import math

# Open a file for writing

file\_name = "duty\_cycle\_data\_sin.txt"

file = open(file\_name, "w")

try:

    # Define the number of data points

    num\_points = 100

    # Generate duty cycle data for a sine waveform

    for i in range(num\_points):

        angle = 2 \* math.pi \* i / num\_points

        duty\_cycle = (math.sin(angle) + 1) \* 50  # Scales sine values to 0-100%

        file.write(f"{duty\_cycle:.2f}\n")

    print(f"Data written to {file\_name}")

finally:

    # Close the file

    file.close()

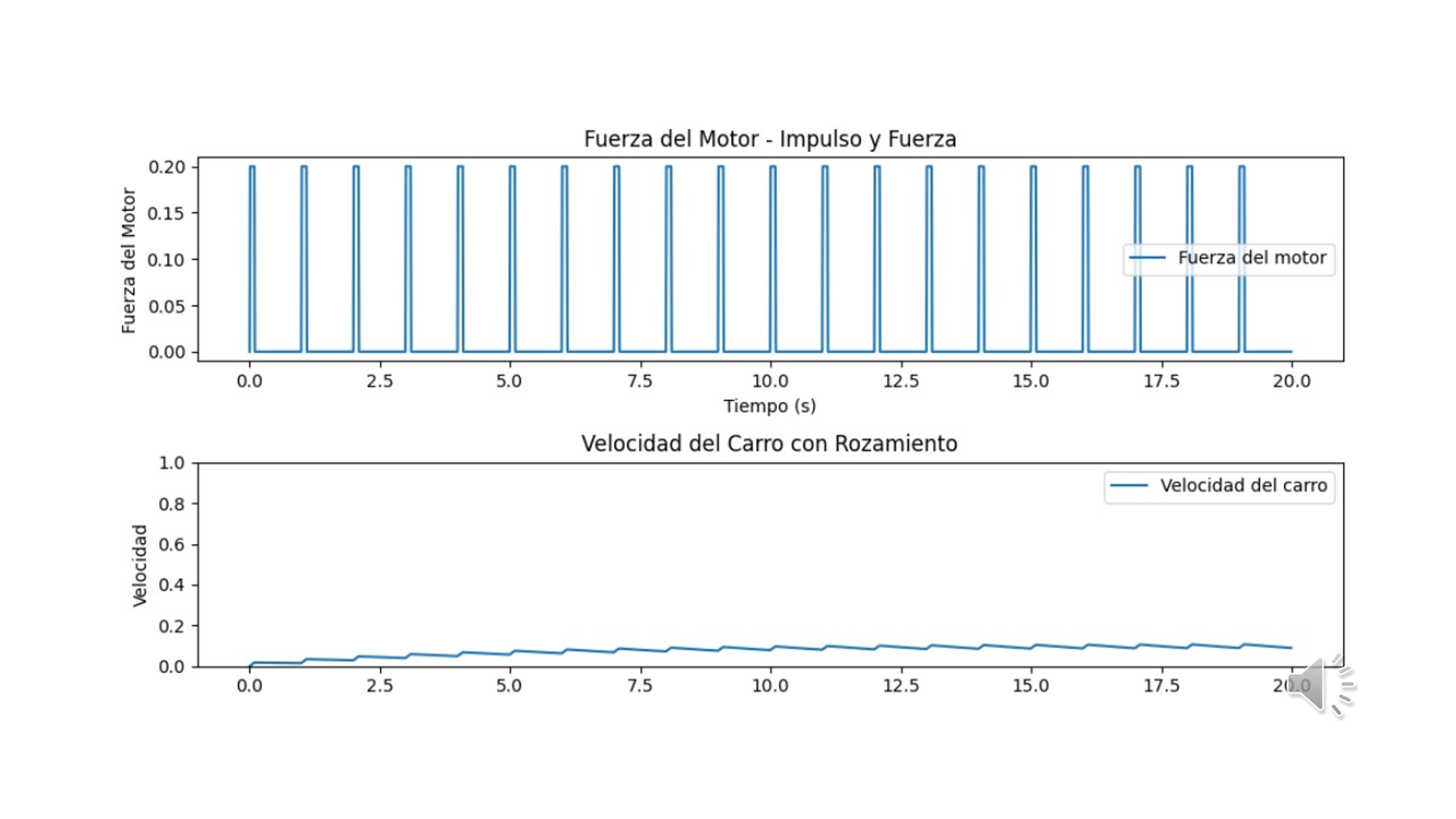
PWM

A graph of a graph

Description automatically generated with medium confidence

A graph of a graph of a motor

Description automatically generated with medium confidence



A graph of a motor

Description automatically generated with medium confidence

A graph of a motor vehicle

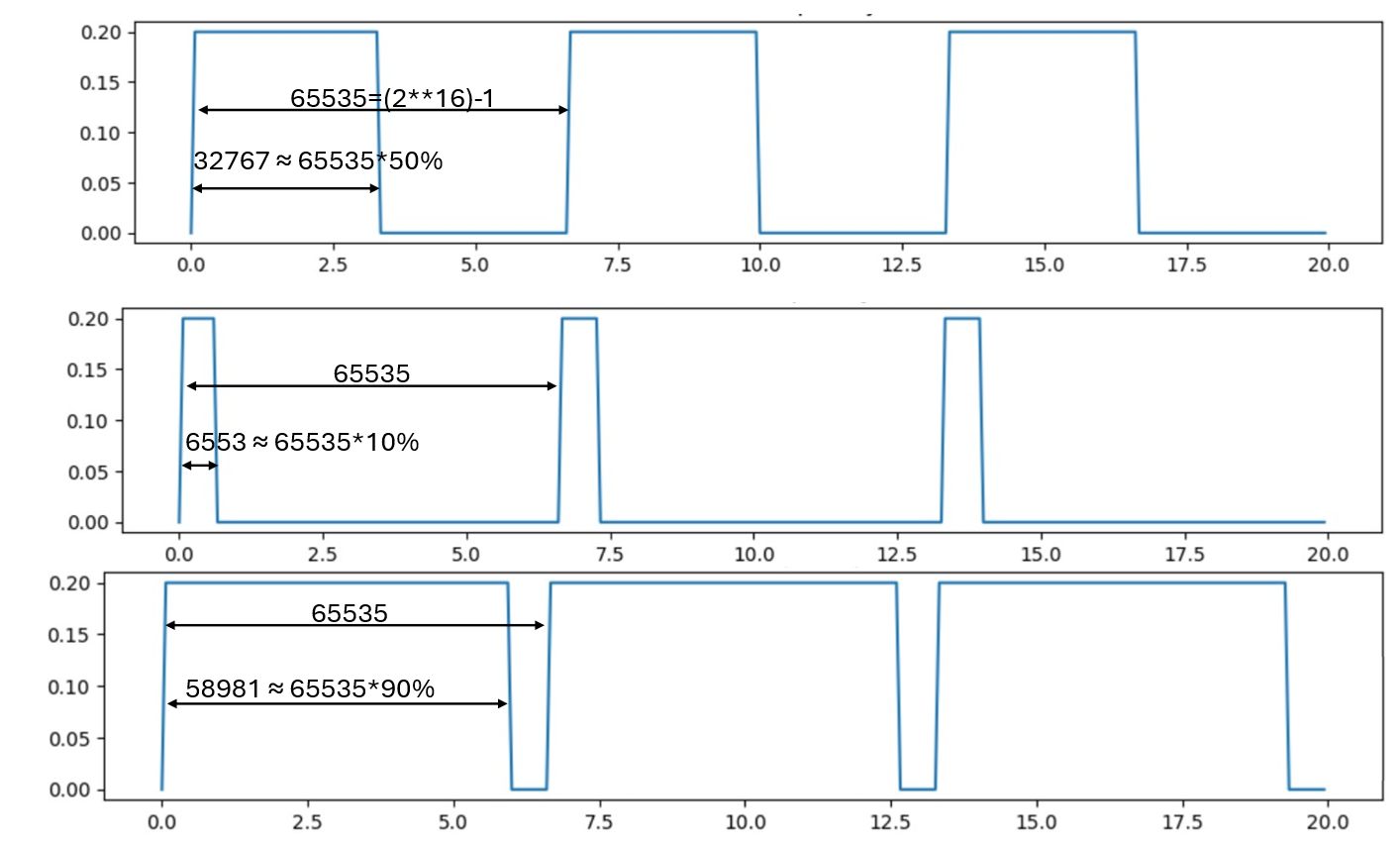
Description automatically generated with medium confidence

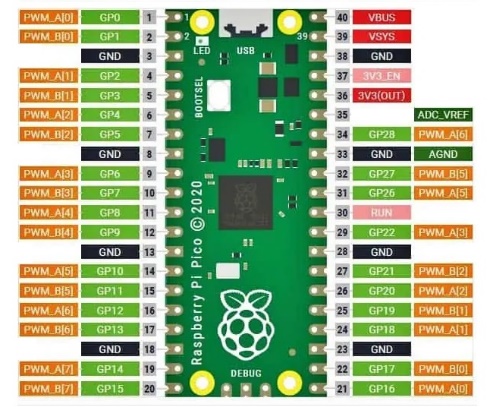
A graph of a car

Description automatically generated with medium confidence

A graph of a graph of a motor

Description automatically generated with medium confidence





from machine import Pin, PWM

# Define PWM pin

pwm\_pin = Pin(18)

# Configure PWM

pwm = PWM(pwm\_pin)

pwm.freq(1000)  # Set the PWM frequency (Hz)

duty\_cycle = 10

pwm.duty\_u16(int(65535 \* duty\_cycle / 100))

# Turn off PWM when done

#pwm.deinit()

from machine import Pin, PWM

import utime

# Define PWM pin

pwm\_pin = Pin(18)

# Open the file for reading

file\_name = "duty\_cycle\_data\_sin.txt"

file = open(file\_name, "r")

# Configure PWM

pwm = PWM(pwm\_pin)

pwm.freq(1000)  # Set the PWM frequency (Hz)

try:

    # Read duty cycles from the file

    duty\_cycles = [float(line.strip()) for line in file.readlines()]

finally:

    # Close the file

    file.close()

# Generate waveform based on duty cycles

for duty\_cycle in duty\_cycles:

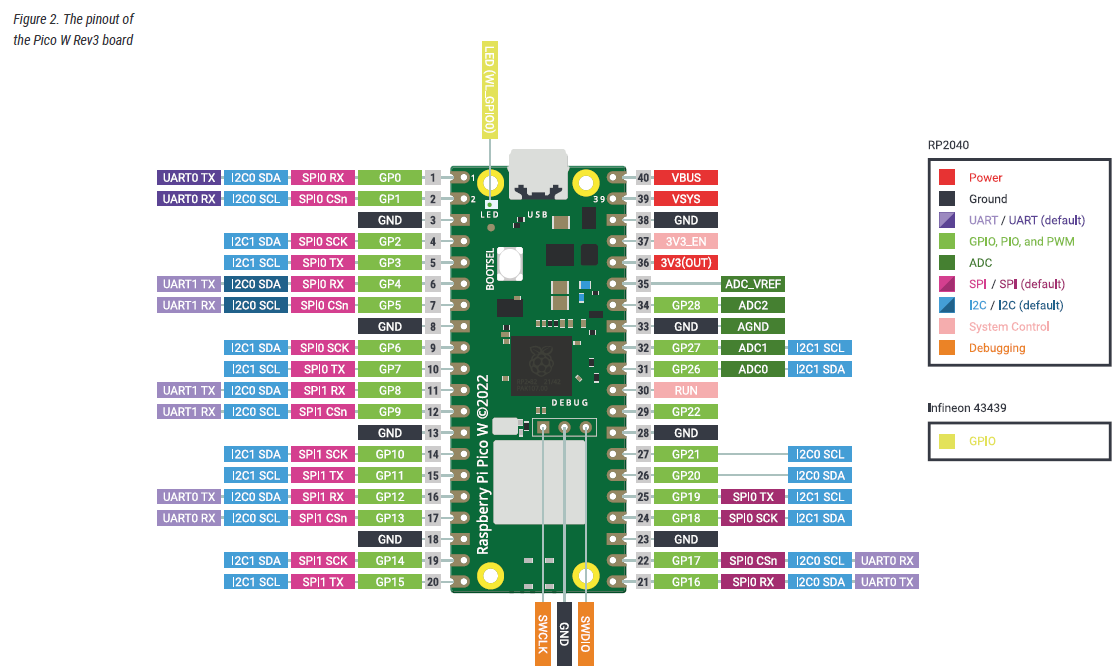
    pwm.duty\_u16(int(65535 \* duty\_cycle / 100))

    utime.sleep\_ms(100)  # Adjust sleep time as needed

# Turn off PWM when done

pwm.deinit()

ADC



from machine import Pin, ADC

# Set up ADC on Pin 26

adc = ADC(0)

print(adc.read\_u16())

from machine import Pin, Timer, ADC

import utime

# Set up ADC on Pin 26

adc = ADC(0)

# Open a file for writing

file\_name = "adc\_data\_timer.txt"

file = open(file\_name, "a") #a append the data

led = Pin("LED", Pin.OUT)

timer = Timer()

left\_readings = 10

def blink(timer):

    global left\_readings

    led.toggle()

    adc\_value = adc.read\_u16()

    file.write(str(adc\_value) + "\n")

    left\_readings -= 1

    if left\_readings < 1:

        timer.deinit()

        file.close()

        print("The END")

timer.init(freq=2.5, mode=Timer.PERIODIC, callback=blink)

ADC with DMA

<https://github.com/GerardoMunoz/Curso_Python/blob/main/ADC_upy_DMA.ipynb>