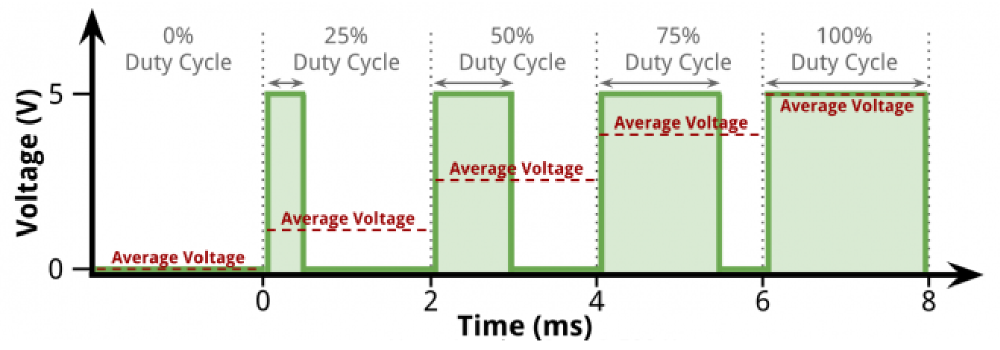
Introduction. Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave - a signal switched between on(HIGH = 3.3V) and off(LOW = 0V). This on-off pattern can simulate voltages by changing the portion of the time the signal spends on versus the time that the signal spends off. The duty cycle is proportional to the average *(time_high/(time_high + time_low) * 100%)* voltage on the selected PWM pin.



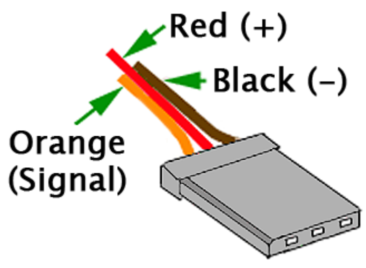
**robotic-controls.com*

Raspberry Pi Pico has PWM capability on all GPIO pins. When a GPIO pin is set to PWM mode, the output of each pin varies with the frequency cycle. The lowest PWM frequency output is 10[Hz].

All the codes are available in the GitHub repository - `ex04_interfaces.zip`.

Task 1. Create a MicroPython script to control the SG90 servomechanism.

Pico board	Servo
Gnd	brown
VSYS	red
GP15	orange



```
from machine import Pin,PWM
from utime import sleep
```

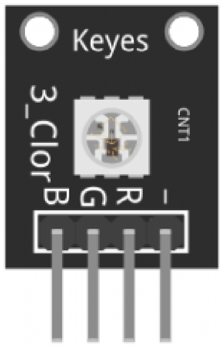
```
MIN_POS = 2500
MAX_POS = 9000

servo = PWM(Pin(15))
servo.freq(50)

#angle from 0 to 180
def set_servo_pos(shaft_position):
    servo.duty_u16(shaft_position)
    sleep(0.2)

while True:
    for position in range(MIN_POS,MAX_POS,50):
        set_servo_pos(position)
    for position in range(MAX_POS,MIN_POS,-50):
        set_servo_pos(position)
```

Task 2. Connect an RGB module to the Raspberry Pi Pico board. You need to use 100R resistors to prevent LED burnout. Those resistors should be connected between Pico board GPIOs and the KY-009 board R, G, and B pins.



Arduino board	KY-009 board	
GP2	B	blue component
GP3	G	green component
GP4	R	red component
Gnd	-	cathode

```
from machine import Pin,PWM
from utime import sleep

#Pins
R_pin = PWM(Pin(2))
G_pin = PWM(Pin(3))
B_pin = PWM(Pin(4))
```

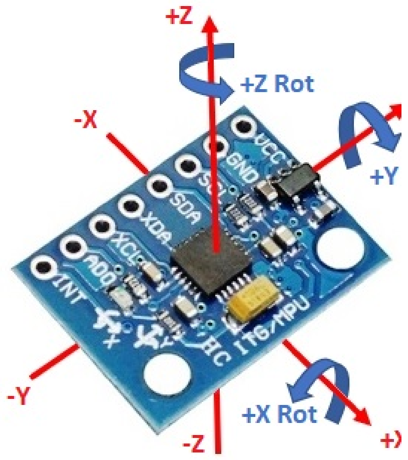
```
while True:
    for intensity in range(0, 65000,5000):
        R_pin.duty_u16(intensity)
        sleep(1)
```

Task 3. I2C interface.

Download `imu.py` and `vector3d.py` form:

https://github.com/micropython-IMU/micropython-mpu9x50

Save the `py` files in the Raspberry Pi Pico memory.



Pico Expander	MPU/ITG board
3V3	VCC
GND	GND
GP14	SDA
GP15	SCL

```
from imu import MPU6050
from machine import I2C, Pin
import time
import ujson

# Initialize I2C
i2c = I2C(1, sda=Pin(14), scl=Pin(15), freq=400000)

mpu = MPU6050(i2c)
#acceleration dictionary
#acc = {"aX" : 0.0,"aY" : 0.0,"aZ" : 0.0}
acc ={}
```

```
while True:
# Accelerometer data (x, y, z)
    print("-" * 100)
    acc['aX'] = mpu.accel.x
    acc['aY'] = mpu.accel.y
    acc['aZ'] = mpu.accel.z
    json_acc = ujson.dumps(acc)
    print(f'x: {acc["aX"]} y: {acc["aY"]} z: {acc["aZ"]}')
    print(json_acc)
    time.sleep(0.5)
# Gyroscope data (x, y, z)
    print('X: {:.2f} Y: {:.2f}'.format(mpu.gyro.x,mpu.gyro.y))
    print('Z: {:.5f}'.format(mpu.gyro.z))
    time.sleep(0.5)
```

Task 4. Connect *Pico-LCD-x.xx* module.

Test *Pico-LCD-x.xx* module. Use examples from the Waveshare web page - section *For those interested* - link number 3. Display a text message and draw a circle on the display.

Task 5. (1 point to the final score) Use the circuit from Task 1. Connect *Pico-LCD-x.xx* module. The user should be able to set the desired servo shaft position on the LCD with the joystick. Pressing the joystick button should move the servo shaft.

Task 6. (0.5 points to the final score) Create a simple HMI consisting of a joystick and 3 buttons for controlling the color of light emitted from the

RGB module. The current R, G, and B values should be presented on the *Pico-LCD-x.xx* module.

Task 7. (1 point to the final score) Use the A and B buttons to control the shaft position of 2 servos independently. The current shaft position should be presented on the LCD (*Pico-LCD-x.xx*).

Task 8. (1 point to the final score). Use an accelerometer to control the servo shaft position. Create and display on the *Pico-LCD-x.xx* module a visualization of the T-bar connected to the servo shaft. The orientation of the T-bar image on the screen should correspond to the current position of the servo shaft.

For those interested:

1. MicroPython web page:
   micropython.org/download/rp2-pico/

2. Control a Servo Motor with Raspberry Pi Pico Using PWM in MicroPython tutorial:
   circuitdigest.com/microcontroller-projects/control-a-servo-motor-with-raspberry-pi-pico-using-pwm-in-micropython

3. Waveshare Pico LCD
   www.waveshare.com/wiki/Pico-LCD-1.8