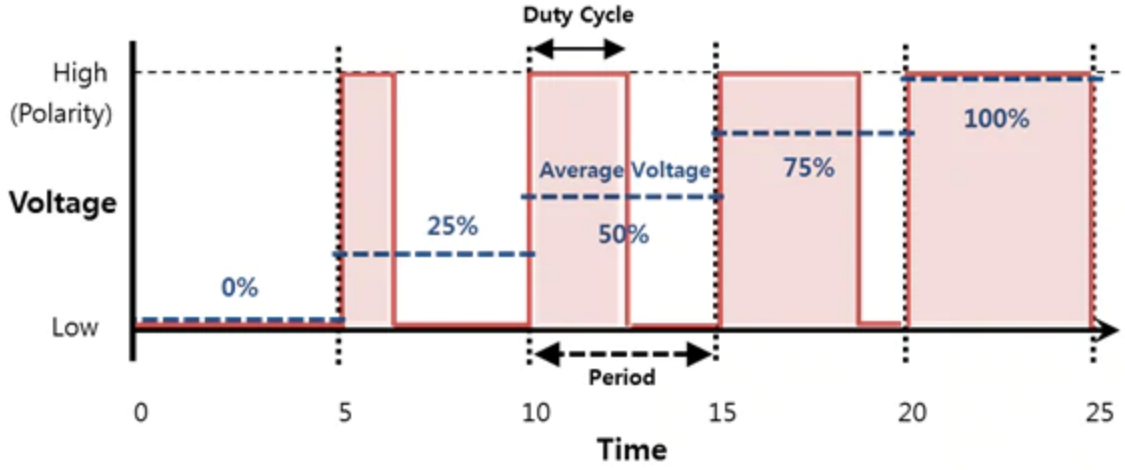
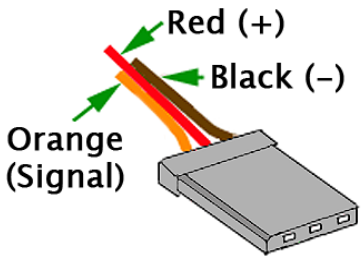
Wprowadzenie. PWM jest techniką generowania sygnału pseudo analogowego za pomocą linii cyfrowych. Na wybranym wyjściu cyfrowym generowany jest przebieg prostokątny o wartości maksymalnej (on) równej 3.3V i wartości minimalnej (off) 0V. W przypadku odpowiednio dużej częstotliwości przełączeń taki sygnał jest odbierany przez urządzenie zewnętrzne tak jak sygnał stały. Współczynnik wypełnienia równy (czas_on)/(czas_on + czas_off) * 100%) jest proporcjonalny do średniej wartości napięcia.



Moduł Raspberry Pi Pico jest zdolny do generowania sygnałów PWM na dowolnym GPIO. Najniższa, dostępna, częstotliwość sygnału prostokątnego wynosi 10[Hz].

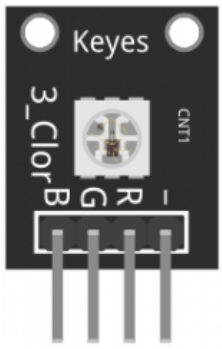
Zad 1. Napisz skrypt w MicroPython do sterowania serwomechanizmem SG-90.

moduł Pico	**serwomechanizm SG-90**
Gnd	brązowy
VSYS	czerwony
GP15	pomarańczowy/żółty



```
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)
```

Zad 2. Podłącz moduł RGB KY-009 do płytki Raspberry Pi Pico. W celu uniknięcia przegrzania LED wchodzących w skład modułu należy zastosować rezystory ograniczające prąd 100R. Rezystory powinny być włączone (szeregowo) pomiędzy wyprowadzenia GPIO na płytce Pico a piny R,G,B modułu Keyes KY-009.



RPi Pico	KY-009	
GP2	B	składowa niebieska
GP3	G	składowa zielona
GP4	R	składowa czerwona
Gnd	-	katoda

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

#Pins
R_pin = PWM(Pin(2))
```

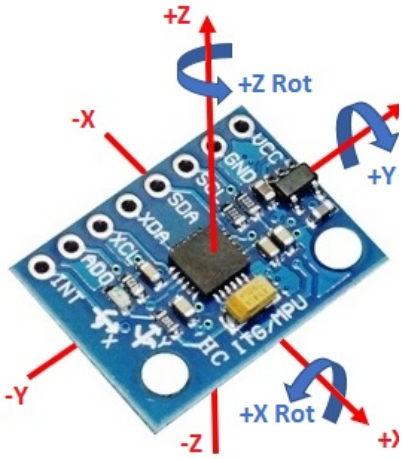
```
G_pin = PWM(Pin(3))
B_pin = PWM(Pin(4))

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

Task 3. Interfejs I2C.

Pobierz `imu.py` oraz `vector3d.py`:

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

Zapisz pliku, z rozszerzeniem `py`, na płytce Raspberry Pi Pico.



Pico Expander	Moduł MPU/ITG
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)
```

Zad 4. (**1 punkt**) Podłącz wyświetlacz Pico-LCD-0.96 lub Pico-LCD-1.8 do płytki Raspberry Pi Pico.

Pobierz *Demo code* z sekcji *Resources* na odpowiedniej stronie:

- www.waveshare.com/wiki/Pico-LCD-0.96
- www.waveshare.com/wiki/Pico-LCD-1.8

Użytkownik, za pomocą joystick'a, może ustawić żądaną pozycję serwomechnizmu na wyświetlaczu. Naciśnięcie przycisku, wbudowanego w joystick, pozwala na obrót wału serwomechanizmu do ustawionej pozycji.

Zad 5. (**1 punkt**) Zbuduj prosty interfejs użytkownika (HMI), składający się z wywietlacza, joysticka i 3 przycisków do kontroli koloru koloru i natężenia światła emitowanego przez moduł RGB.

Zad 6. (**1.5 punktu**). Wykorzystaj akcelerometr do sterowania położenia wału serwomechanizmu. Wizualizacja graficzna położenia orczyka (T-bar) serwomechanizmu powinna być prezentowana w czasie rzeczywistym na wyświetlaczu, wymienionym w zadaniu 4.

Dla zainteresowanych:

1. MicroPython - strona:
   micropython.org/download/rp2-pico/
2. Sterowanie serwomechanizmem przy zastosowaniu techniki PWM w MicroPython:
   circuitdigest.com/microcontroller-projects/control-a-servo-motor-with-raspberry-pi-pico-using-pwm-in-micropython