Serdar Arıtan

serdar.aritan@hacettepe.edu.tr

Hacettepe Üniversitesi

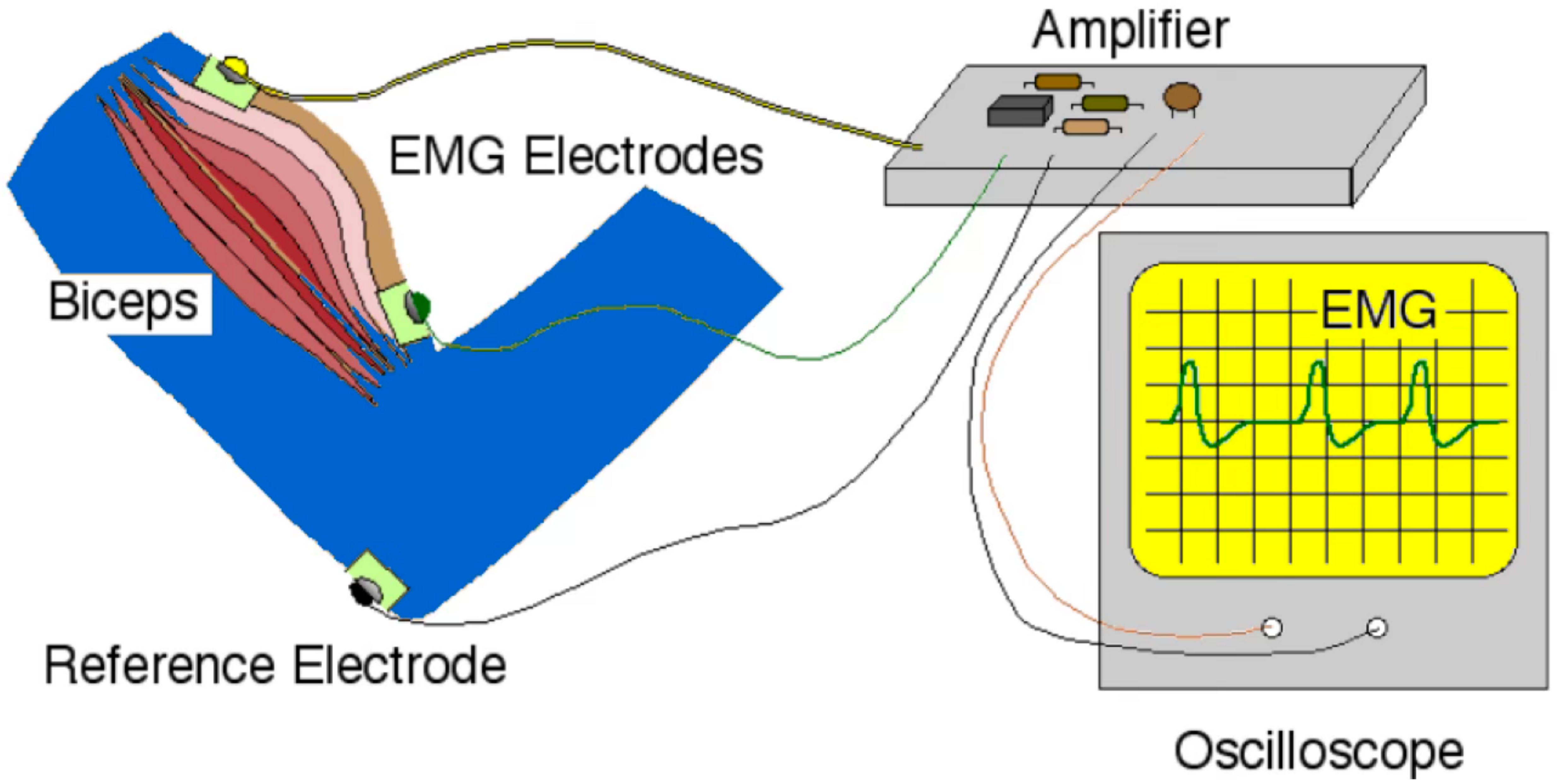
www.hacettepe.edu.tr

Biyomekanik Araştırma Grubu

www.biomech.hacettepe.edu.tr

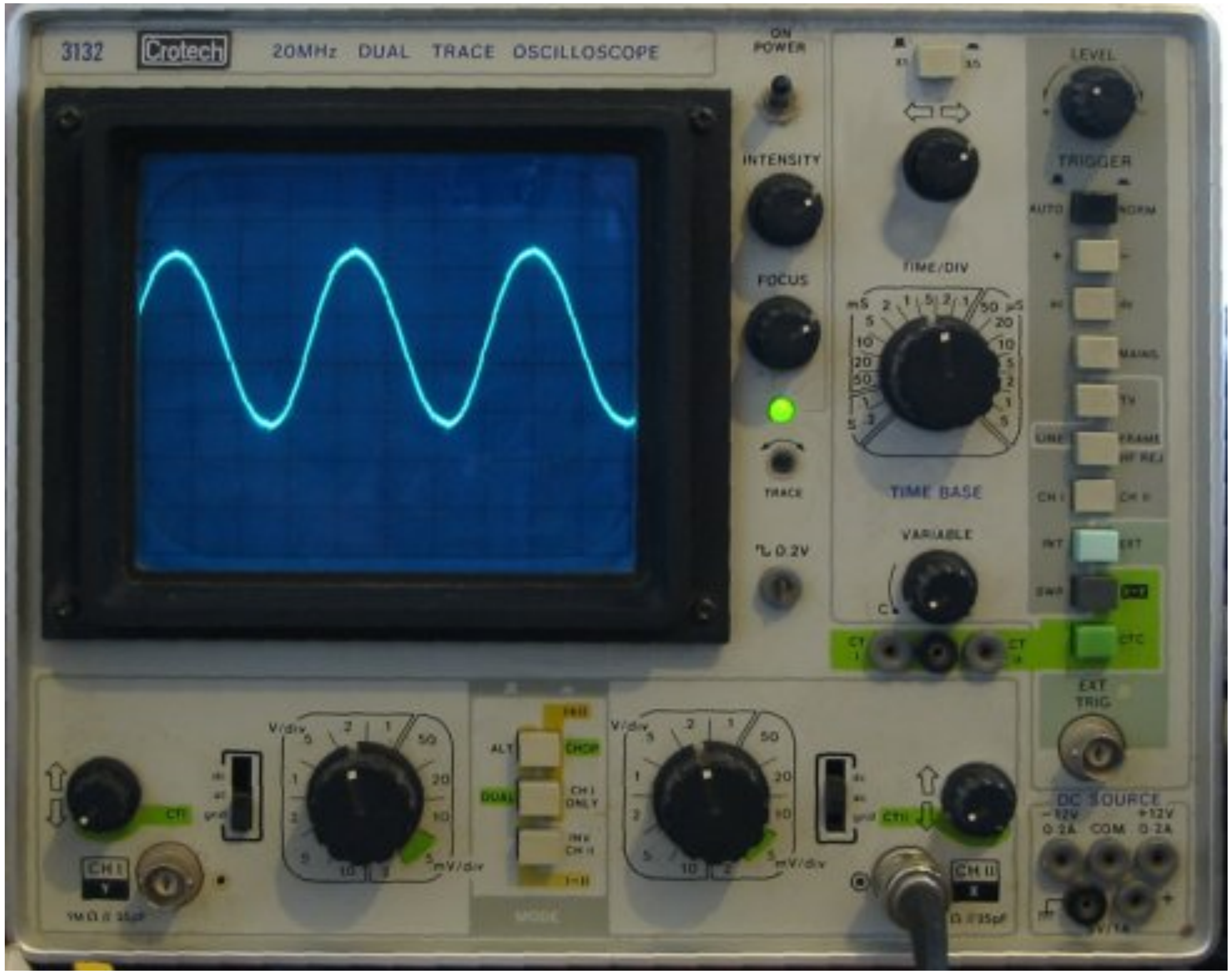
#3

Oscillography

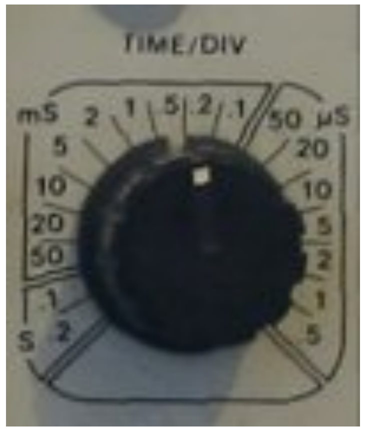


Oscillography

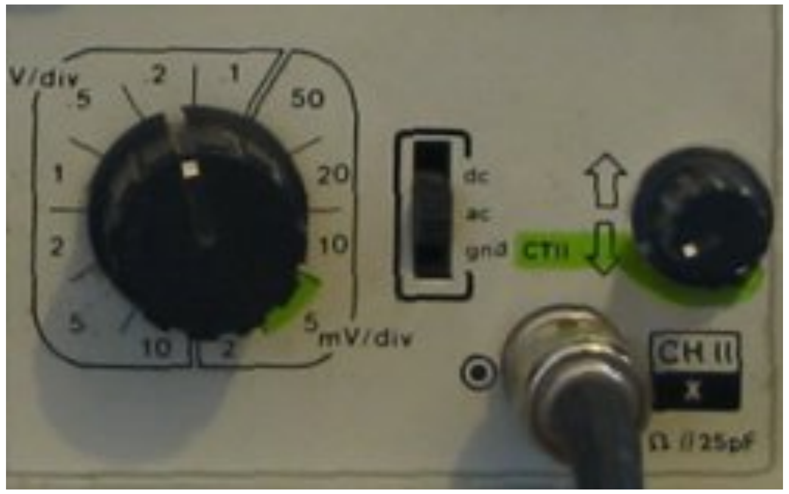
Oscilloscope: -cathode ray tube (CRT) with controls for displaying analog waveforms (continuously varying voltages)



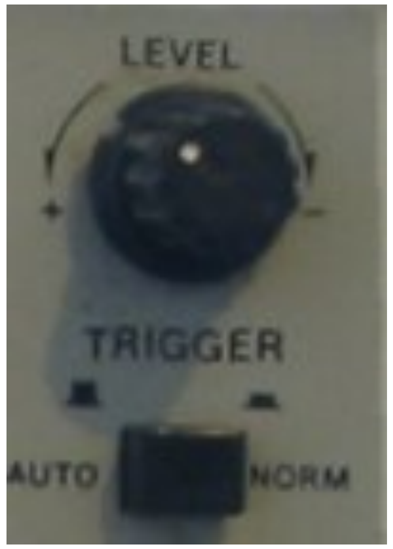
Oscillography



The timebase sets the time that the beam is scanned from left to right on the screen and it's calibrated in horizontal divisions (the black grid on the front of the screen).

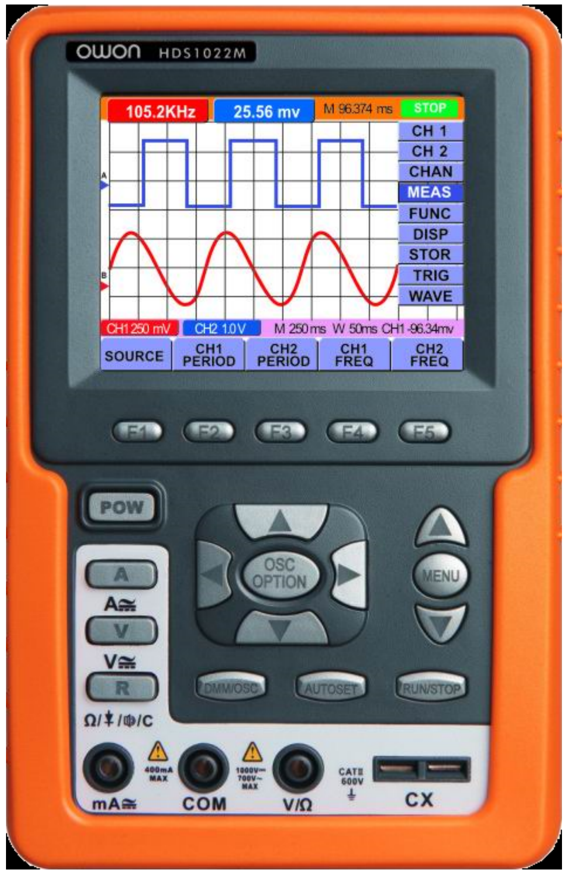


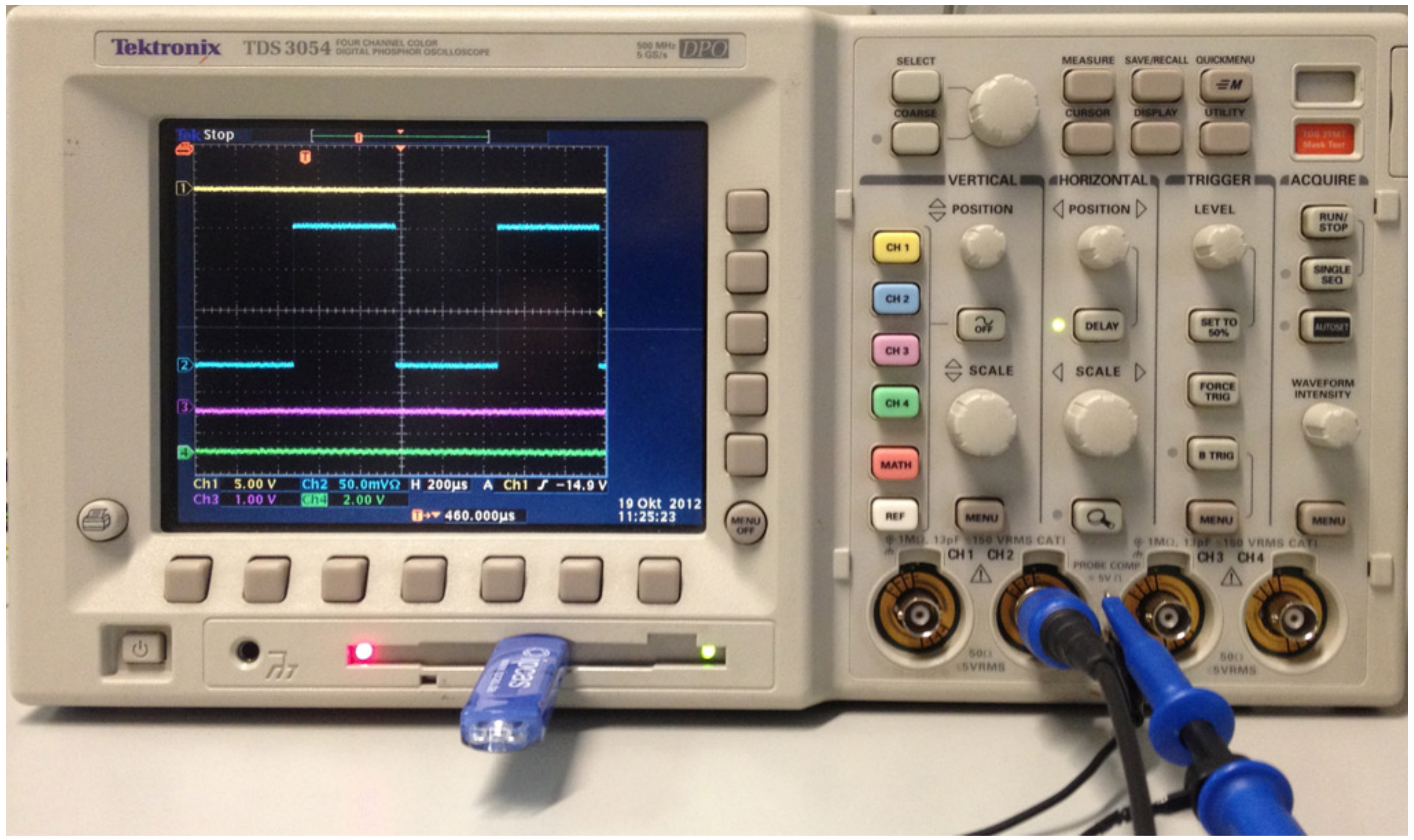
Each channel on the oscilloscope is really just a high quality amplifier with low noise, high bandwidth and selectable gain which connects to the vertical deflector in the oscilloscope.



The trigger detects when to start moving the beam to the right across the display. Setting this to Auto makes the beam trigger continuously.

Oscillography





Oscillography

Vertical section

- controls for amplitude of input(s),
- coupling (AC or DC), ground,
- single-ended or differential,
- invert signal and some allow multiple inputs (2 or 4)

Horizontal section

- time base or X-Y mode,
- horizontal start position,
- sweep speed,
- For multiple waveforms select alternate or chopped sweep depending upon sweep speed

Trigger section

- determines when beam starts its sweep across CRT or LCD.
- Source may be from an input of the vertical section, internal triggering, external triggering or line frequency.

Oscillography

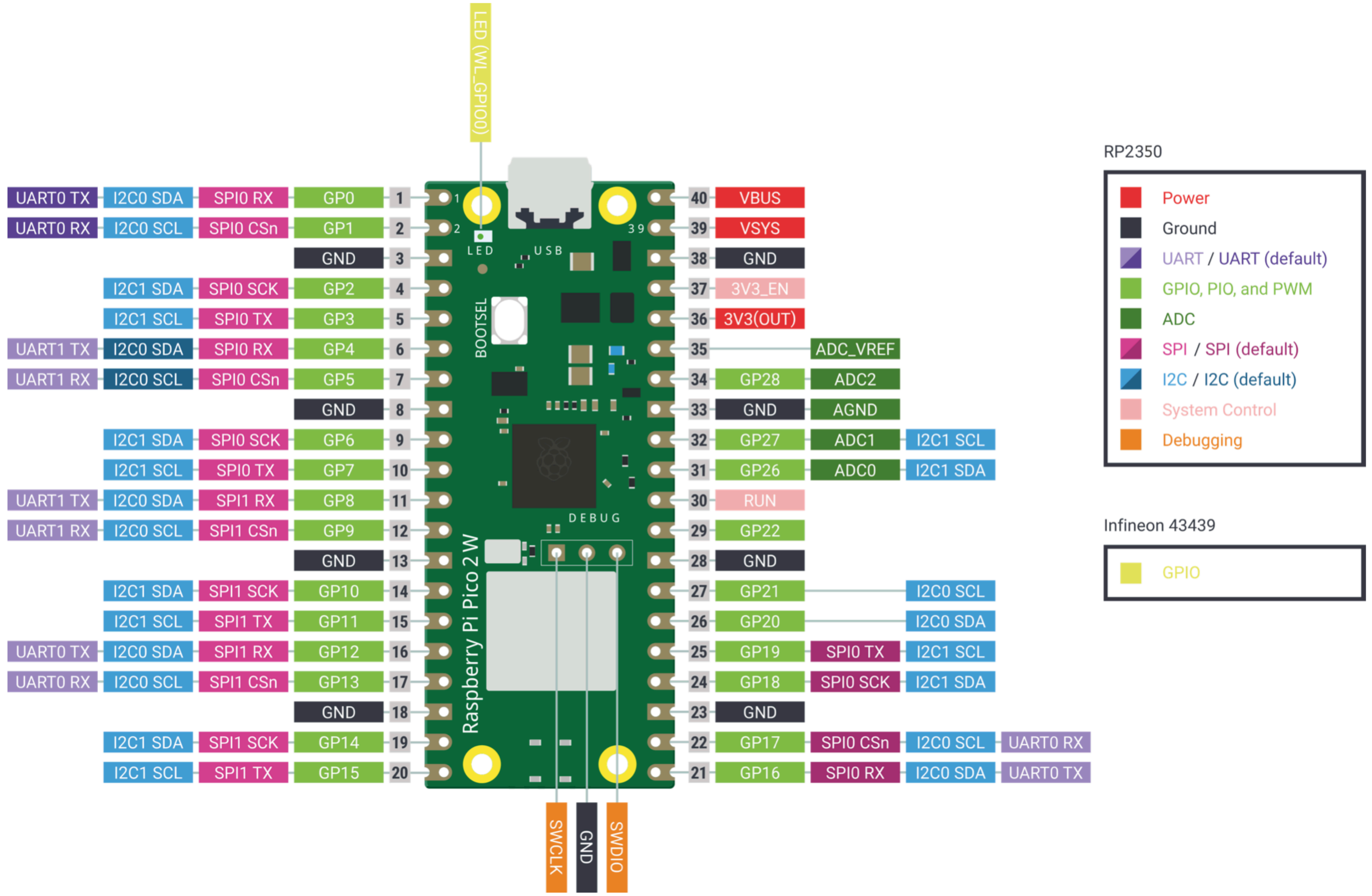
Coupling

Ground:
- shorts input (0 volts) to enable vertical positioning of beam

AC
- removes **DC component** of waveform. **Useful for EMG waveforms** because it keeps waveform centered around 0 volts, like a high-pass filter.

DC
- on some scopes labelled AC/DC. Waveform unaffected before amplification. Must be used for force and displacement waveforms.

Try: Use a signal generator to show how AC-coupling influences a sine wave that has a variable DC offset (or bias).


LED (WL_GPIO0)
UART0 TX I2C0 SDA SPI0 RX GP0 1
UART0 RX I2C0 SCL SPI0 CSn GP1 2
GND 3
I2C1 SDA SPI0 SCK GP2 4
I2C1 SCL SPI0 TX GP3 5
UART1 TX I2C0 SDA SPI0 RX GP4 6
UART1 RX I2C0 SCL SPI0 CSn GP5 7
GND 8
I2C1 SDA SPI0 SCK GP6 9
I2C1 SCL SPI0 TX GP7 10
UART1 TX I2C0 SDA SPI1 RX GP8 11
UART1 RX I2C0 SCL SPI1 CSn GP9 12
GND 13
I2C1 SDA SPI1 SCK GP10 14
I2C1 SCL SPI1 TX GP11 15
UART0 TX I2C0 SDA SPI1 RX GP12 16
UART0 RX I2C0 SCL SPI1 CSn GP13 17
GND 18
I2C1 SDA SPI1 SCK GP14 19
I2C1 SCL SPI1 TX GP15 20
40 VBUS
39 VSYS
38 GND
37 3V3_EN
36 3V3(OUT)
35 ADC_VREF
34 GP28 ADC2
33 GND AGND
32 GP27 ADC1 I2C1 SCL
31 GP26 ADC0 I2C1 SDA
30 RUN
29 GP22
28 GND
27 GP21 I2C0 SCL
26 GP20 I2C0 SDA
25 GP19 SPI0 TX I2C1 SCL
24 GP18 SPI0 SCK I2C1 SDA
23 GND
22 GP17 SPI0 CSn I2C0 SCL UART0 RX
21 GP16 SPI0 RX I2C0 SDA UART0 TX
LED
USB
BOOTSEL
DEBUG
Raspberry Pi Pico 2 W
1
2
39
SWCLK
GND
SWDIO
RP2350
Power
Ground
UART / UART (default)
GPIO, PIO, and PWM
ADC
SPI / SPI (default)
I2C / I2C (default)
System Control
Debugging
Infineon 43439
GPIO

Raspberry Pi Pico – PWM Channels

The Raspberry Pi Pico has **8 independent PWM generators** called slices. Each slice has two channels, which makes a total of **16 PWM channels**.

The frequency of the PWM signal can range between 8Hz and 62.5MHz, while the microcontroller is running at a frequency of 125MHz.

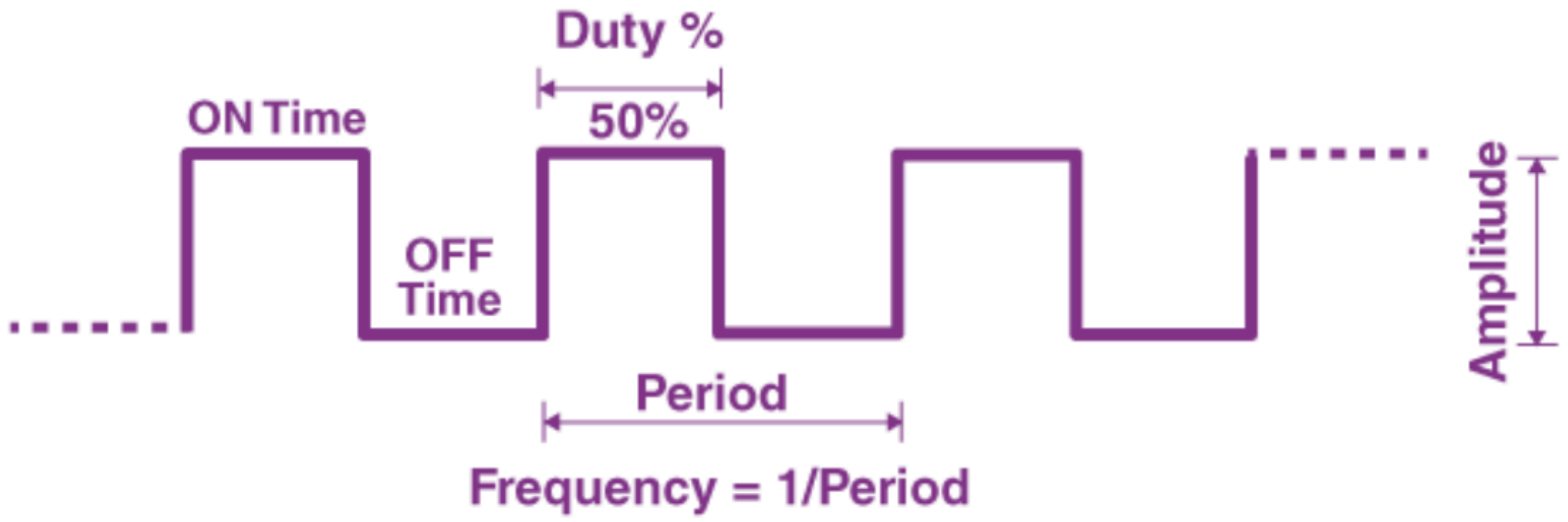
```
from machine import PWM

pwm = PWM(pin, freq=50, duty_u16=8192)   # create a PWM object on a pin
                                          # and set freq 50 Hz and duty 12.5%
pwm.duty_u16(32768)                       # set duty to 50%

# reinitialise with a period of 200us, duty of 5us
pwm.init(freq=5000, duty_ns=5000)

pwm.duty_ns(3000)                         # set pulse width to 3us

pwm.deinit()
```

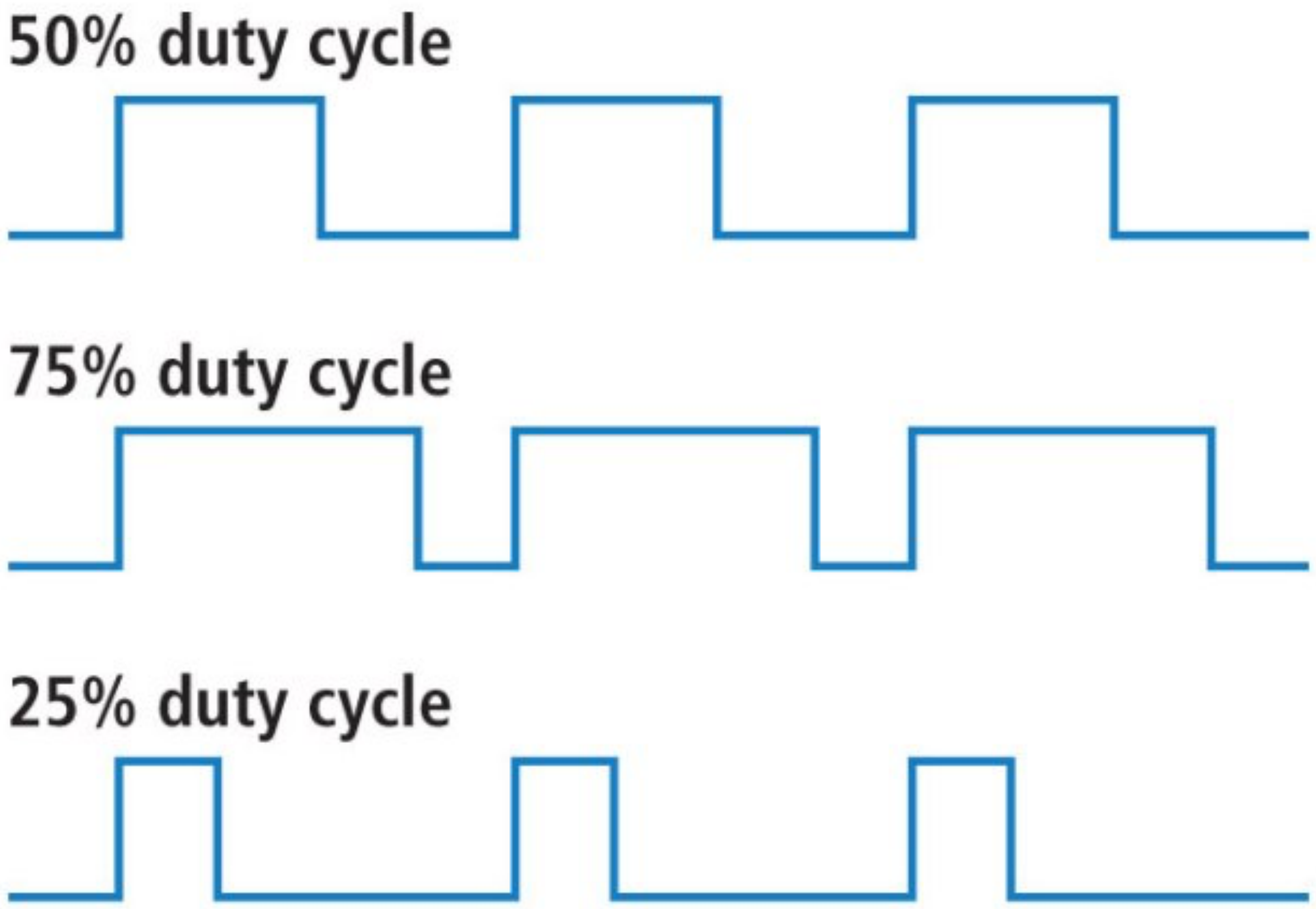


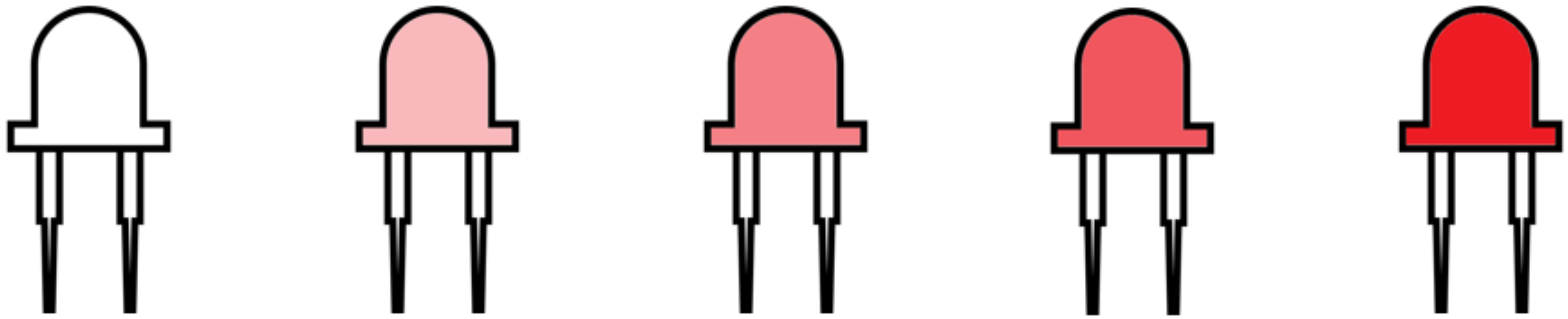
The frequency of PWM determines how fast a PWM completes a period. The frequency of a pulse
shown in the figure above.

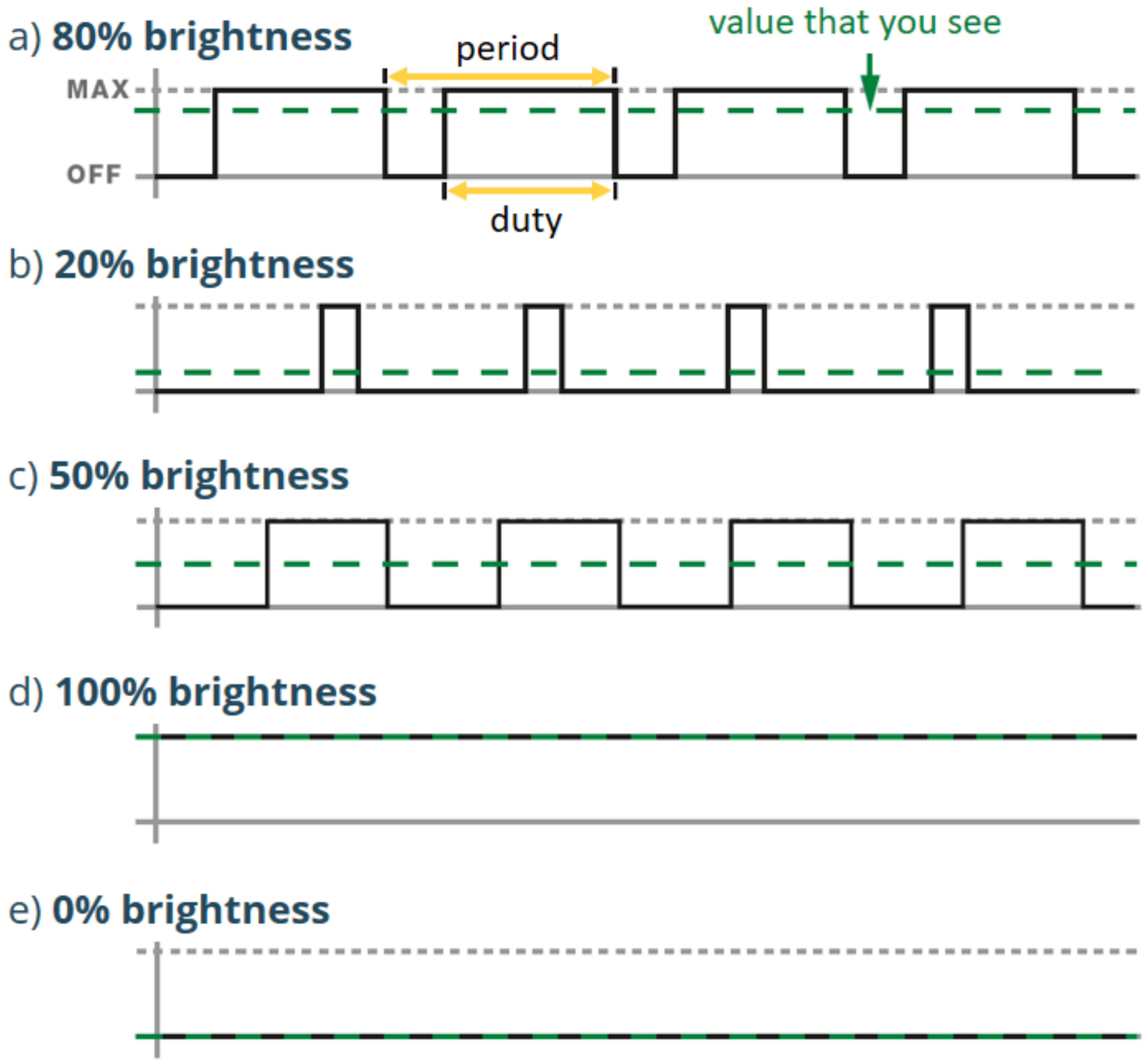
The frequency of PWM can be calculated as follows:

Frequency = 1/Time Period

Time Period = On Time + OFF time


50% duty cycle
75% duty cycle
25% duty cycle




a) 80% brightness
period
value that you see
MAX
OFF
duty
b) 20% brightness
c) 50% brightness
d) 100% brightness
e) 0% brightness

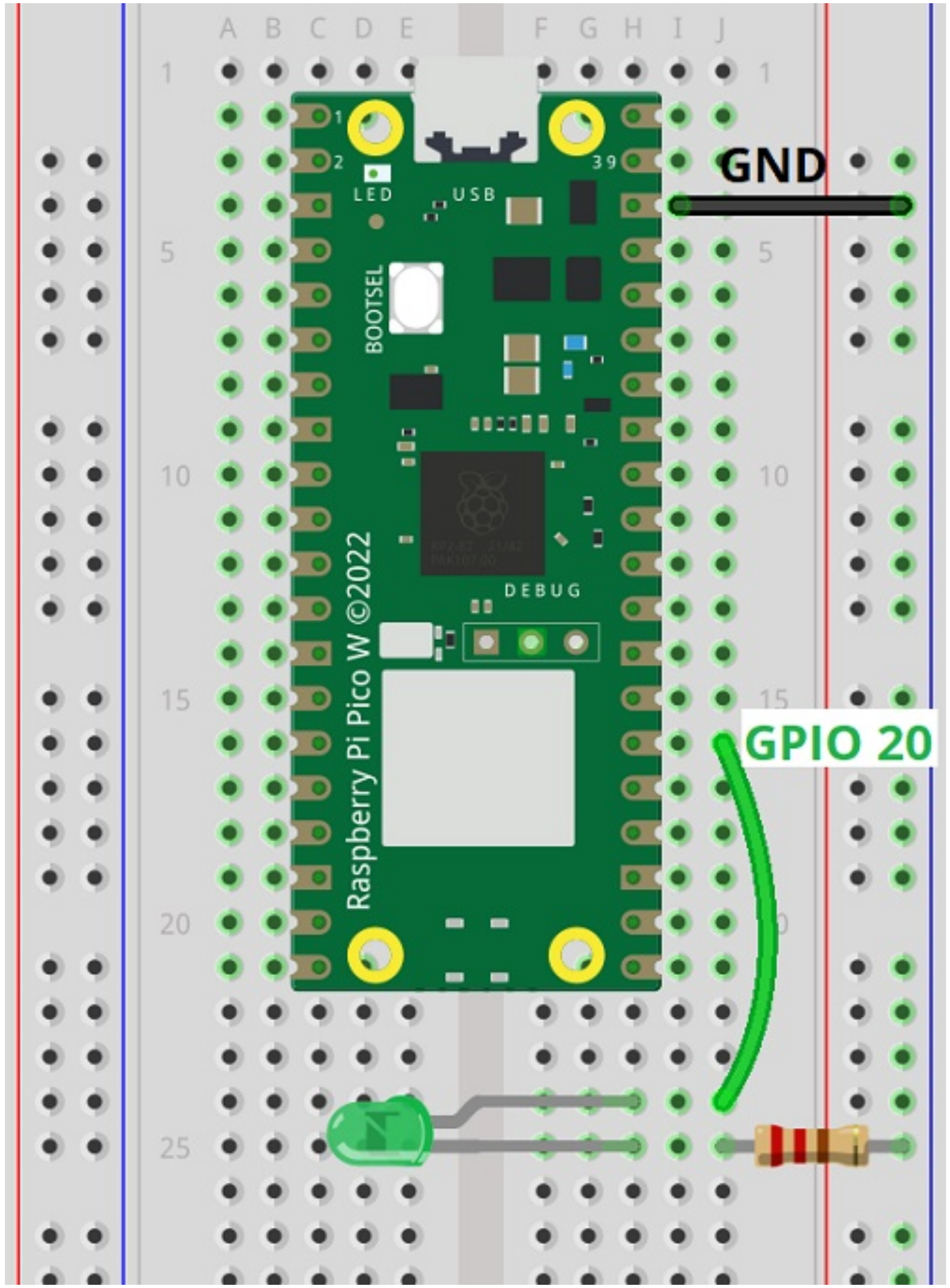
To handle PWM signals with the Raspberry Pi Pico using MicroPython, use the `machine.PWM` class. After creating a `PWM` object, called `out_pwm` for example:
`out_pwm = PWM(Pin(20))`

You can use the following methods:

`led_pwm.freq(FREQUENCY)` to set the frequency of the PWM signal.

`led_pwm.duty_u16(DUTY_CYCLE)` to set the duty cycle. It should be a value between 0 and 65535 (16-bit)

`led_pwm.deinit()` to turn off PWM on the PWM slice used by the OUT (in this example, it's GPIO 20, so it will also stop PWM on GPIO 21 because they belong to the same PWM slice).


GND
GPIO 20
Raspberry Pi Pico W ©2022
BOOTSEL
LED
USB
DEBUG
A B C D E F G H I J

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

# Set up PWM Pin
led = machine.Pin(20)
led_pwm = PWM(led)
duty_step = 129  # Step size for changing the duty cycle

#Set PWM frequency
frequency = 5000
led_pwm.freq (frequency)

try:
    while True:
      # Increase the duty cycle gradually
      for duty_cycle in range(0, 65536, duty_step):
        led_pwm.duty_u16(duty_cycle)
        sleep(0.005)

      # Decrease the duty cycle gradually
      for duty_cycle in range(65536, 0, -duty_step):
        led_pwm.duty_u16(duty_cycle)
        sleep(0.005)

except KeyboardInterrupt:
    print("Keyboard interrupt")
    led_pwm.duty_u16(0)
    print(led_pwm)
    led_pwm.deinit()
```