UCD School of Physics
Scoil na Fisice UCD

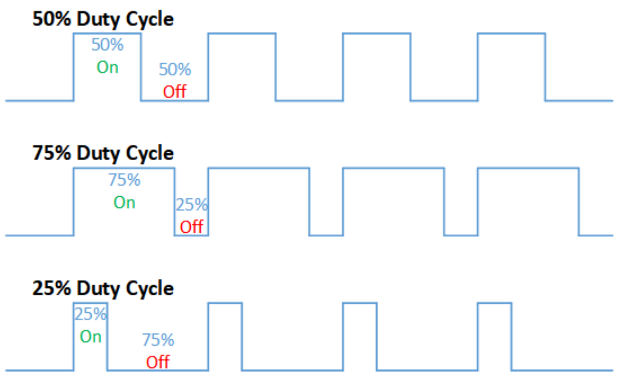
555 Timer & Raspberry Pi Pico Pulse-Width Modulation

Aim:

The aim of this project is to become familiar with the PWM techniques and their applications. The PWM output of a Raspberry Pi Pico and a general-purpose 555 IC will be investigated and then the will be used to build a PWM controller for a motor. If time permits, you are requested to investigate and prototype another 555 Timer application (not necessarily PWM) of your own choosing!

Definition:

Pulse-width modulation is a method of reducing the average power delivered by an electrical system, by effectively chopping it up into discrete parts. The average power supplied to the load is controlled by quickly switching on and off the voltage supplied to the load. This switching is typically done at high frequencies, so that there is no noticeable flicker in power delivery.

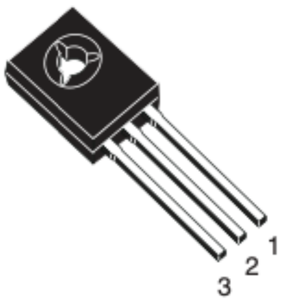


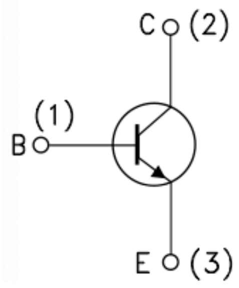
Components:

Raspberry Pi Pico Microcontroller:
A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip and usually runs C code. However, there are new microcontrollers which can run special versions of Python (MicroPython or CircuitPython) instead of C with the performance considerably slower in Python than in C.

555 Timer:
The 555 timer IC is a specialised integrated circuit used in a variety of timer, delay, pulse generation, and oscillator applications

BD139 NPN Transistor:
When a small current flows from the base (Pin 1) to the emitter (Pin 3), a large current flows from the collector (Pin 2) to the emitter (Pin 3). The collector-to-emitter "driving" current for the BD139 can be up to 1.5 amps. Make sure to use a 1 K resistor between the PICO/555 and the base to keep the "control" current small. CAUTION: transistors may become warm during operation, especially when switching.





Note pins on the BD139 are labelled from RHS, which is quite unconventional. Normally pin 1 is the leftmost pin.

Raspberry Pi Pico PWM:

The Raspberry PI Pico is a modern micro-controller, from the company that produces mini computers, which has multi-function digital I/O pins. including PWM. The Pico is programmed in a dialect of the Python language, called Micropython. The suggested development environment for programming the Pico is a lightweight program called "Thonny" (available at https://thonny.org/). Further information can be found on RaspberryPI.org

Some useful Micropython timing commands:

I. For delays you can use `sleep_ms()` or `sleep_us()` from the `time` library to specify the sleep time in milliseconds or microseconds (in integers).

II. For measuring time intervals you can use the `ticks_ms()` or `ticks_us()` and `ticks_diff()` functions to many how milliseconds or microseconds elapsed. One thing to be carefully of is that sometimes the numbers roll over and Usage:

```
start = ticks_ms()

<run some code>

end = ticks_ms()

print(ticks_diff(end, start))
```

For more info see: https://docs.micropython.org/en/latest/library/time.html

PI Pico ***PWM:***

The frequency command (`pwm.freq()`) tells Raspberry Pi Pico how often to switch the power between on and off. The Raspberry Pi Pico can output PWM frequencies of greater than a MHz but for this lab. the maximum you should use is 100 kHz.

The duty cycle (`pwm.duty_u16(duty)`) determines how long it should be on each time. For Raspberry Pi Pico in MicroPython, this can range from 0 to 65535. 65535 would be 100% of the time, 32767 would indicate that it should be on for half the time.

Task 1: PI Pico PWM Exercises:

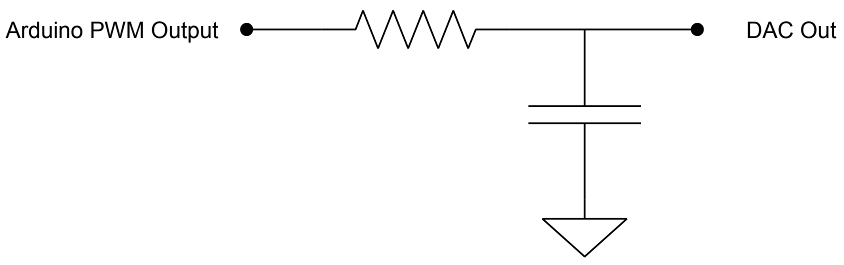
a) ***Basics:***

I. Verify that PWM output of the Pico looks as expected (view the output on an oscilloscope and look at a few duty cycles and frequencies).

II. Control the brightness of an LED from your computer by specifying a brightness between 0 (off) and 1 (full brightness). You can use the green LED on the Pico board which is on GPIO Pin 25 or an external LED (remember to use a 330 Ω current-limiting resistor as well). No visible flickering should be visible for a given brightness setting. Explain your choice of PWM frequency.

III. Make the LED brightness “linearly” brighten and fade with a frequency of approximately 0.25 Hz using 100 points over each cycle. Does the variation look linear to the eye? Note: the eye has a logarithmic response to changes in light intensity so consider using logarithmic steps for the brightness changes and visually compare to the linear steps.

b) **Simple DAC:** Create a simple DAC (Digital to Analog) converter by supplying the PWM signal to a low pass filter (i.e. **the analogue output is the averaged/smoothed PWM signal**).



I. How does the choice of the R and C values combined with the PWM frequency impact on the smoothness of the output of the DAC and what are the implications for producing high frequency signals with the DAC? Note: in this case it is useful to consider the charging and discharging of the capacitor.

II. Determine an R and C combination so that the signal has less than 5% variation for 50% duty cycle of a PWM frequency of 20 kHz. Make sure to include any calculations in your report. Verify the performance by measuring on an oscilloscope.

III. Explain why the output shouldn’t be used unbuffered and how unity gain op-amp (voltage follower) can be used to buffer the output signal (there is no need to build the voltage follower).

c) **Generating analogue output functions**: By using the low-pass filter and varying the PWM duty cycle, output signals, such as triangular and sine functions, of much lower frequency than the PWM can be generated. This can be achieved by changing the PWM duty cycle in discrete steps with a certain number of steps per cycle. Python is slow (~200 times slower has C/C++) and it might be more efficient to pre-compute the values and write out the values rather than performing calculations in the main loop. Also, the time Python takes to execute the commands in a loop is significant (may be 10s of microseconds) which may result in the output sine wave having a lower frequency than you expect - try to come up with ways to compensate for the slow Python command execution.

I. Generate a triangular signal and display it on an oscilloscope with the period of the signal set by a variable in you program. Verify with triangular waves of period 100 ms and 50 ms.

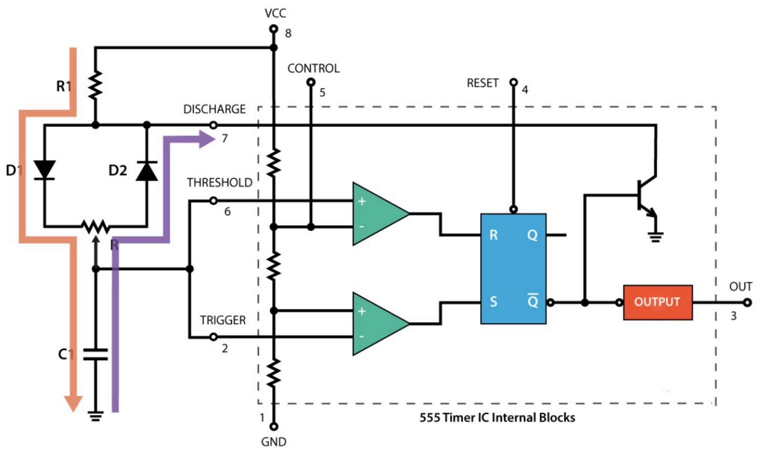
II. Generate a sine wave where you can set its frequency (in Hz) in your code and record a few different frequencies on an oscilloscope, verifying the correct frequency is produced. Look at frequencies in the range 10 to 100 Hz. Note: What happens if you try to generate higher frequency sine waves? (Note: you might compare to the 3 dB “cutoff” frequency of the filter you chose).

Task 2, 555 timer:

The 555 Timer is a popular timing IC that is capable of generating PWM signals when set up as an astable oscillator mode. There are many online tutorials in addition to the handout provided.

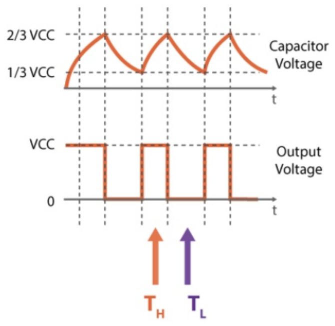
(A) Simulate the 555 Timer as an Astable Oscillator using TINA or on https://www.falstad.com/circuit/ to become familiar with its use as an oscillator. Use your simulations to understand what controls the frequency and duty cycle of the output.

(B) Simulate the 555 Timer as a PWM generator with duty cycle controlled by a potentiometer, as



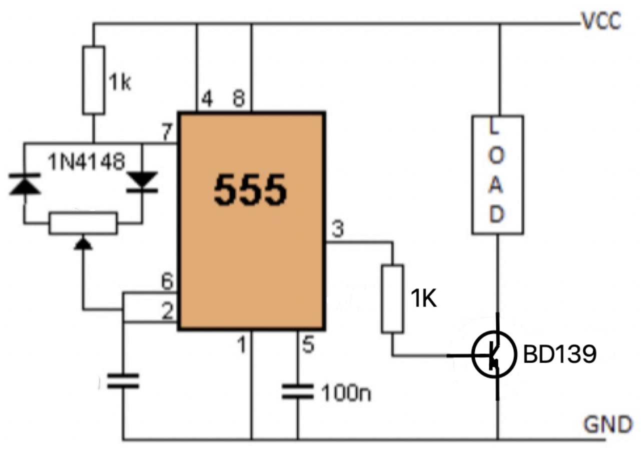
shown in the circuit shown in the figure below. Use an oscilloscope to view the output and investigate the effect of changing the potentiometers value. There is no need to include the circuit connected to pin 3 for now.

Explanation of 555 PWM circuit:
External to IC: An RC circuit acts as the timekeeping element in this circuit. The 555 chip essentially just controls the RC pair and generates a clean square wave from the charging and discharging capacitor C1. The potentiometer and R1 act as the resistors in the RC circuit, and determine the rate of charge and discharge. Diodes D1 and D2 route the current depending on whether charging or discharging (Orange is charging, purple, discharge). Sliding the potentiometer changes both times, but keeps total time the same. (i.e Frequency), which is perfect for PWM.



Inside the IC: There are three resistors inside the 555, splitting the supply voltage into 3. Comparators inside the device compare the voltage at the Threshold and Trigger pins to 1/3 VCC and 2/3 VCC respectively, in order to toggle the output accordingly. A high output pulls the discharge pin low, triggering the discharge of the capacitor and thus repeating the cycle.

(C) Build the circuit on a breadboard and use a DC motor as the load, by sending the PWM signal to the base of the transistor. Use 5 V from a bench power supply as VCC, source voltage, for the circuit.



Try different RC values, to get higher and lower PWM frequencies, note what you observe about the motor behaviour.

(D) (Time permitting) Come up with another application for a 555 Timer chip and prototype it. It does not have to be PWM and can include other components!

PWM Application Examples:

PWM is very common as a means of power control.

The LUAS tram uses PWM to regulate power to the drive motors. The switching is carried out at high frequencies, on the order of kHz, and can be heard as a high pitched whine.

Electric drills also use PWM for speed control, and PWM is used to control the brightness of LEDs such as laptop backlighting and daytime running lights in cars.

Feel free to make additions/improvements to the above project. For any questions or extra components, contact the Electronics Workshop, Room 332

Useful Pi Pico links:

https://www.tomshardware.com/how-to/raspberry-pi-pico-setup
Pin Diagram: https://datasheets.raspberrypi.com/pico/Pico-R3-A4-Pinout.pdf