Lab 0x01 - Driving DC Motors

You will have one week to complete this exercise in your lab groups. You will be introduced to DC motor driving.

1 Preemptive Caution

Some of the hardware you will be using for this assignment is sensitive. Please regard the following suggestions to protect your hardware:

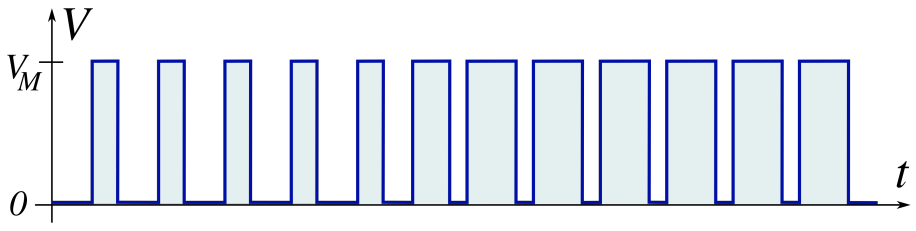
- Do not touch any conductive parts on any printed circuit boards. Human skin can have voltages in the range of 20kV to 50kV due to static charge which can cause immediate damage to electronics.
- Be careful with polarity and orientation as you are adjusting your circuit. You should likely not need to adjust any wiring after your initial setup.
- Do not change any wiring while a device is running. Instead, power off the device completely before changing your circuit.

2 Spinning a Motor

DC motors are simple reliable actuators that are ubiquitous in the world around us. As part of this exercise you will be spinning a small DC motor by applying pulse width modulation or PWM. In common practice, most motors are controlled using switching circuitry and PWM because it is cheaper, more efficient, and simpler than using analog circuitry.

PWM motor control techniques take advantage of the fact that DC Motors act like low-pass filters. Therefore, high frequency input signals are filtered out but the low frequency average value of the signal passes through. If the frequency of this PWM signal is large enough, then the motor will only be affected by the average value.

Consider the PWM waveform shown below. About halfway through the waveform, the duty-cycle[1] changes from about 40% to about 80%. From the motor's 'perspective', this is equivalent to the applied voltage across the motor leads changing from $V = 0.4V_M$ to $V = 0.8V_M$, doubling the effort requested from the motor.



[1] Duty-cycle is defined as the ratio of on-time (or high-time) to the period of the waveform. A duty cycle of 30% would be have an on-time equal to 30% of the total PWM period.

The Nucleo L476 development board that you have been using in ME405 comes equipped with several internal timer modules that can generate ultra-precise PWM. Your first task is to become familiar generating PWM using the hardware drivers available in MicroPython.

Begin by surveying the documentation available at https://docs.micropython.org/en/latest/library/pyb.html; pay special attention to the class list at the bottom of the page which shows the available hardware drivers for our Nucleo L476.

3 The Project

Your core task for this assignment is to write a driver class that will allow you to independently command two DC motors connected to the L6206 motor driver that is part of the IHM04A1 expansion board for the Nucleo. The expansion board carries the L6206 motor driver from ST Microelectronics, along with required supporting circuitry, so that you may drive two DC motors bidirectionally. Driving two motors independently will be accomplished by writing a simple motor driver class so you may instantiate multiple objects of the class, one for each motor.

Consider the following code snippet:

```
PC0 = Pin(Pin.cpu.C0, mode=Pin.OUT_PP)
PC1 = Pin(Pin.cpu.C1, mode=Pin.Analog)
```

In the preceding code snippet a pair of pin objects are instantiated. That is, from a single `Pin` class definition multiple pin objects can be instantiated so that you may interact with each physical pin through its own variables and functions.

For this lab you will write an `L6206` class that is in many ways analogous to the `Pin` class in the example above. That is, you will write a class representing *a single channel* of the L6206 so that you may create as many objects as needed for as many motors as needed.

```
mot_A = L6206(...)
mot_B = L6206(...)
```

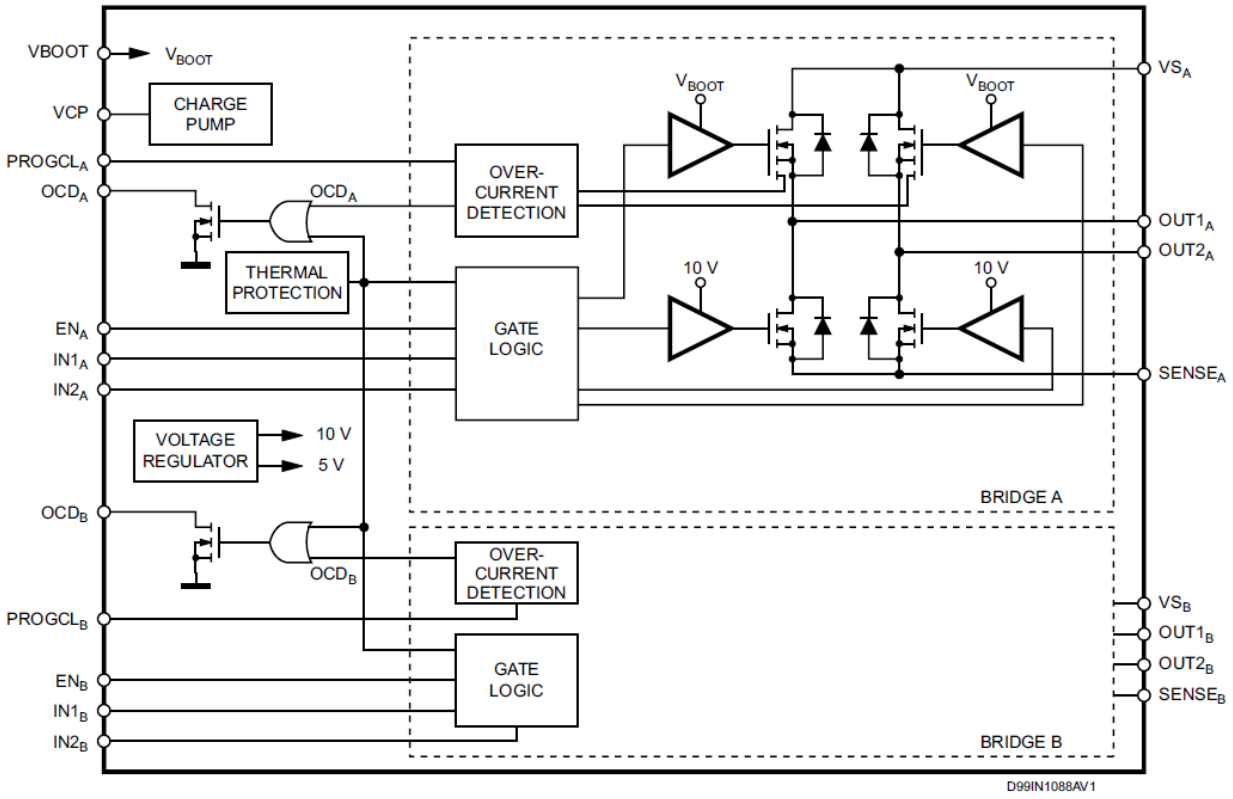
3.1 Familiarization

Before beginning to write your driver class for this assignment, first familiarize yourself with the hardware and get things spinning.

An excerpt from the L6206 datasheet, also available online, is shown below; it should be noted that the behavior of the motor driver is similar to many other DC motor drivers built around H-bridges. Each one of the chip's two H-bridges consists of four N-channel MOSFETs, but only the details associated with channel A are shown in the right half of the diagram.

The first motor would be connected between pins OUT1_A and OUT2_A which are commanded by the microcontroller through the control pins IN1_A and IN2_A using pulse width modulation. An additional pin, EN_A, is an active-high enable pin. The EN_A pin is pulled low by the driver internally[2] so that the motor driver will be in sleep mode by default and the motors will not spin up without your intervention. To enable, or "wakeup", the motor you can set the EN_A pin high with your microcontroller.

[2]Some particular IHM04A1 expansion boards do not effectively keep the enable pin low on startup, so you *may* see some spurious motor activity when you first power on your system.



A **truth table** in the L6206 datasheet, reproduced below, specifies the logic between the input pins connected to the microcontroller and what the chip does to the motor.

Inputs			Outputs	
EN	**IN1**	**IN2**	**OUT1**	**OUT2**
L	X(1)	X(1)	High Z(2)	High Z(2)
H	L	L	GND	GND
H	H	L	Vs	GND
H	L	H	GND	Vs
H	H	H	Vs	Vs

1. X = don't care.
2. High Z = high impedance output.

A common way of using these pins is to first enable the motor driver by setting the EN pin high, and to then set IN1 high while sending a PWM signal to IN2; this will power the power the motor in the forward direction. Applying PWM in this way has the effect of alternating between row 3 (forward) and row 5 (brake) in the truth table shown below; the PWM duty cycle therefore represents the percentage of time spent in row 3 or row 5[3]. This type of switching is called slow decay mode. Exchange the signals applied to IN1 and IN2 to drive the motor in the other direction.

The table below outlines the connections between the IHM04A1 expansion board and the Nucleo. Your team will need to fill in the remaining columns of the table to determine what CPU pins to use for each signal on the L6206.

L6206 Signal	Morpho Pin Number	CPU Pin	Timer Number	Timer Channel
EN_A	CN10-33		N.A.	N.A.
$IN1_A$	CN10-27			
$IN2_A$	CN10-29			
EN_B	CN7-36		N.A.	N.A.
$IN1_B$	CN7-28			
$IN2_B$	CN7-30			

Before writing your class, attempt to spin one of the DC motors using commands in the REPL. Don't forget to enable the motor driver with the EN pin! Try to make your motor spin both directions at different speeds before moving any of your code to a file.

3.2 L6206 Motor Driver Class

Part of your assignment for this week is to write and test a motor driver class that encapsulates all of the features associated with driving a single DC motor using pulse width modulation. Use the following steps to begin your motor driver class.

(a) Use the starter file found below to begin writing a driver encapsulating all of the functionality that will be useful for interacting with the motor and to write test code evaluating the functionality of your motor driver class.

In addition to filling in the code where appropriate, make sure to adjust all of the comments to improve the level of detail and reflect the code that you write.

(b) You must be able to independently control two separate motors so your motor driver must be parameterized properly to allow the timer, timer channels, and pins associated with a given motor to be selected at the time of instantiation.

(c) You do *not* need to use a task-based system to initially test your motor driver code, but your motor driver code must work cooperatively if the methods are called from within a task as you will eventually need a dedicated motor task to complete future assignments.

(d) Your motor driver code should not print anything to the serial port during normal operation. If you use print messages for debugging while writing your class make sure to remove those print statements before you finish your driver.

[3]The precise interpretation of the duty cycle depends on whether the PWM is inverted or not. For non-inverted PWM the duty cycle represents the percentage of time in the motor brake row of the truth table. Only for inverted PWM does the duty cycle represent the percentage of time in the forward direction row of the table.

Listing 1: Starter Template for `L6206.py`

```
'''!@file L6206.py
'''
from pyb import Pin, Timer

class L6206:
    '''!@brief      A driver class for one channel of the L6206.
        @details    Objects of this class can be used to apply PWM to a given
                    DC motor on one channel of the L6206 from ST Microelectronics.
    '''

    def __init__ (self, PWM_tim, IN1_pin, IN2_pin, ... ):
        '''!@brief      Initializes and returns an object associated with a DC motor.
        '''
        pass

    def set_duty (self, duty):
        '''!@brief      Set the PWM duty cycle for the DC motor.
            @details    This method sets the duty cycle to be sent
                        to the L6206 to a given level. Positive values
                        cause effort in one direction, negative values
                        in the opposite direction.
            @param      duty    A signed number holding the duty
                                cycle of the PWM signal sent to the L6206
        '''
        pass

    def enable (self):
        '''!@brief      Enable one channel of the L6206.
            @details    This method sets the enable pin associated with one
                        channel of the L6206 high in order to enable that
                        channel of the motor driver.
        '''
        pass

if __name__ == '__main__':
    # Adjust the following code to write a test program for your L6206 class. Any
    # code within the if __name__ == '__main__' block will only run when the
    # script is executed as a standalone program. If the script is imported as
    # a module the code block will not run.

    # Create a timer object to use for motor control
    tim_A = Timer(n, freq = 20_000)

    # Create an L6206 driver object. You will need to modify the code to facilitate
    # passing in the pins and timer objects needed to run the motors.
    mot_A       = L6206(tim_A, Pin.cpu.xx, Pin.cpu.yy, ...)

    # Enable the L6206 driver
    mot_A.enable()

    # Set the duty cycle of the first L6206 channel to 40 percent
    mot_A.set_duty(40)
```

4 Deliverables

There will be no online submission for this lab assignment.

To receive credit for this lab you must demonstrate for your instructor at the beginning of your next lab period that your motor driver is working correctly. That is, during the first 30 minutes of lab, one week after this lab is assigned, you must demonstrate through commands in PuTTY that your code is capable of:

- Running a motor connected to Channel A forward and backward at arbitrary speeds.
- Running a motor connected to Channel B forward and backward at arbitrary speeds.
- Running both motors simultaneously without one motor affecting the other.

Late demonstrations will not be accepted for this assignment. Reach out to your instructor preemptively if you think you may not have a functional setup ready in time to demonstrate.