ECE 476/676 - Homework #4

Timing, Analog I/O, Motors with Binary Inputs - Due Monday, February 10th

Motor Speed Control

1) Hardware: Connect your DC motor to your Pi-Pico. Verify that the Pico can make the motor spin CW and CCW

Step 1) Check your motor works: Connect your DC motor to 3.3V and ground. The motor should spin

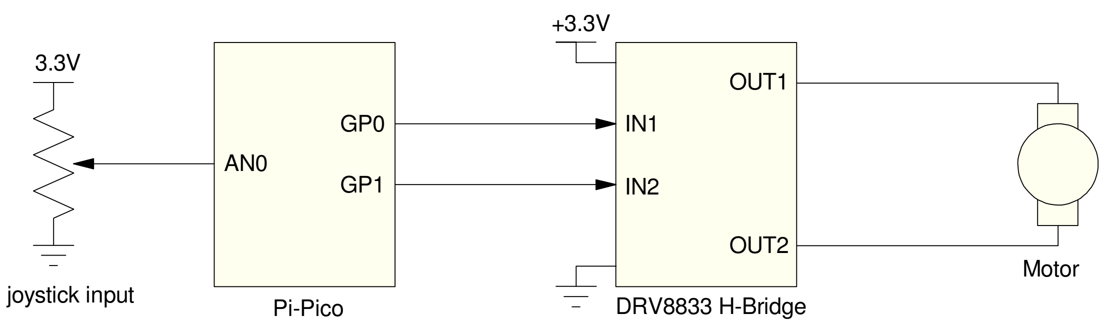
Step 2) Check your H-bridge works:

- Connect the H-bridge to the motor along with 3.3V and ground.
- Connects IN1 and IN2 directly to 3.3V and ground. You should see

IN1.	IN2	Motor
0V	0V	stopped (0V)
3.3V	0V	CW (+3.3V)
0V	3.3V	CCW (-3.3V)

Step 3) Connect the H-bridge to your Pi-Pico. You should be able to set the direction of the motor in software

```
from machine import Pin
from time import sleep

IN1 = Pin(0, Pin.OUT)
IN2 = Pin(1, Pin.OUT)

IN1.value(1)
IN2.value(0)
print('CW')
sleep(2)
IN1.value(0)
IN2.value(0)
print('Stop')
sleep(2)
IN1.value(0)
IN2.value(1)
print('CCW')
sleep(2)
IN1.value(0)
IN2.value(0)
```



2) Software: Write a Python program which

- Reads the analog input on AN0 (the joystick input) and
- Drives a DC motor via an H-bridge

The analog input controls the PWM driving the motor

- When the joystick is left in it's rest state (middle position), the PWM to the motor remains constant
- When the joystick is pushed forward (towards 3.3V), the motor speeds up (PWM slowly increases to +100%)
- When the joystick is pulled back (towards 0V), the motor slows down (PWM slowly decreases to -100%)

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

def Analog_Out(Pct):
    if(Pct < -100):
        Pct = -100
    if(Pct > 100):
        Pct = 100
    PW = int( abs(Pct) * 655.35)
    if(Pct > 0):
        IN1.duty_u16(PW)
        IN2.duty_u16(0)
    else:
        IN1.duty_u16(0)
        IN2.duty_u16(PW)

a2d1 = ADC(1)
k = 3.3 / 65520

IN1 = Pin(16, Pin.OUT)
IN1 = PWM(Pin(16))
IN1.freq(1000)
IN1.duty_u16(0)

IN2 = Pin(17, Pin.OUT)
IN2 = PWM(Pin(17))
IN2.freq(1000)
IN2.duty_u16(0)

center = a2d1.read_u16()
Pct = 0
dt = 0.1

while(1):
    a1 = a2d1.read_u16()
    V1 = k * (a1 - center)

    Pct += V1 * dt * 10

    Analog_Out(Pct)

    print(V1, Pct)
    sleep(dt)
```

3) Test and verify your Python program works

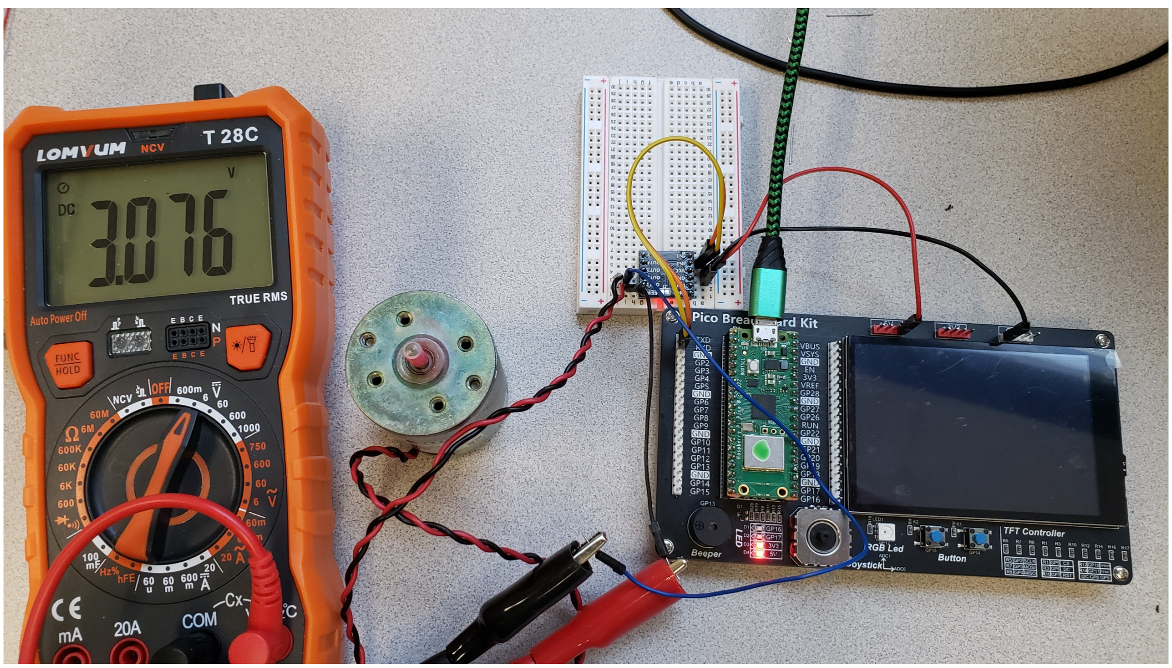
Adjusting the duty cycle and measuring the DC voltage at OUT1 - OUT2

PWM	Votlage	Motor
+100%	+4.467V	CW
+50%	+3.989V	CW
+25%	+2.879V	CW
0	0V	stop
-25%	-2.820V	CCW
-50%	-3.382V	CCW
-100%	-4.447V	CCW

It's not quite linear (not sure why), but the voltage changes with the PWM input

4) Demo (in-person or with a video)

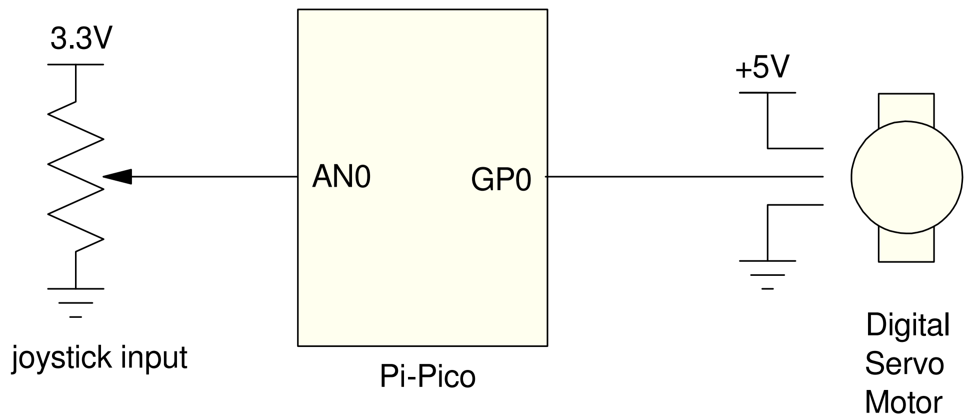
- Looks better in the video



H-Bridge driven from GP0 - GP1 along with 5V input
Motor & Volt Meter connected to OUT1 - OUT2

Motor Angle Control

5) Hardware: Connect your digital servo motor to your Pi-Pico.



6) Software: Write a Python program which

- Reads the analog input on AN0 (the joystick input) and
- Drives a digital servo motor from 0% to 100% of it's angle output

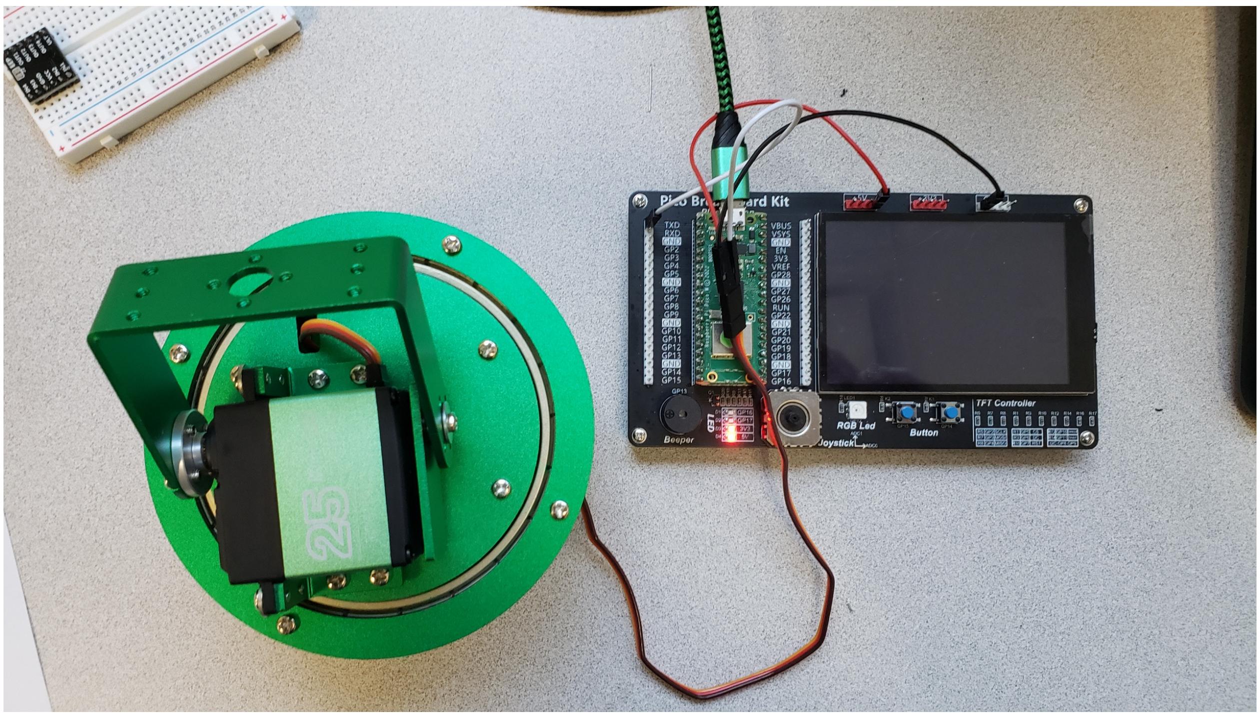
```
from machine import ADC, PWM, Pin
from time import sleep

def Analog_Out(Pct):
    if(Pct < 0):
        Pct = 0
    if(Pct > 100):
        Pct = 100
    ns = int( 500_000 + Pct*20_000)
    Control.duty_ns(ns)


a2d1 = ADC(1)
k = 3.3 / 65520

Control = Pin(16, Pin.OUT)
Control = PWM(Pin(16))
Control.freq(50)

center = a2d1.read_u16()
Pct = 50
dt = 0.1

while(1):
    a1 = a2d1.read_u16()
    V1 = k * (a1 - center)

    Pct += V1 * dt * 10

    Analog_Out(Pct)

    print(V1, Pct)
    sleep(dt)
```

7) Test and verify your Python program works

- Pushing the pot up moves the motor CW
- Pulling the pot back moves the motor CCW
- Center holds the motor stationary (slow drift)

Duty Cycle	Angle
0%	0 degrees
35%	90 degrees
66%	180 degrees
100%	270 degrees

8) Demo (in-person or with a video)



Pico connected to a digital servo motor (alt-azimuth assembly - $50 version of what's in your kit)