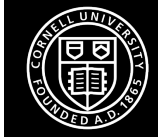
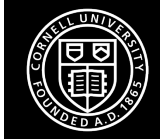


ECE 5725
Principles of Embedded Operating Systems
Lecture 16

Prof. Joseph F. Skovira



Items

Lab 2 Report

Lab3 week 1

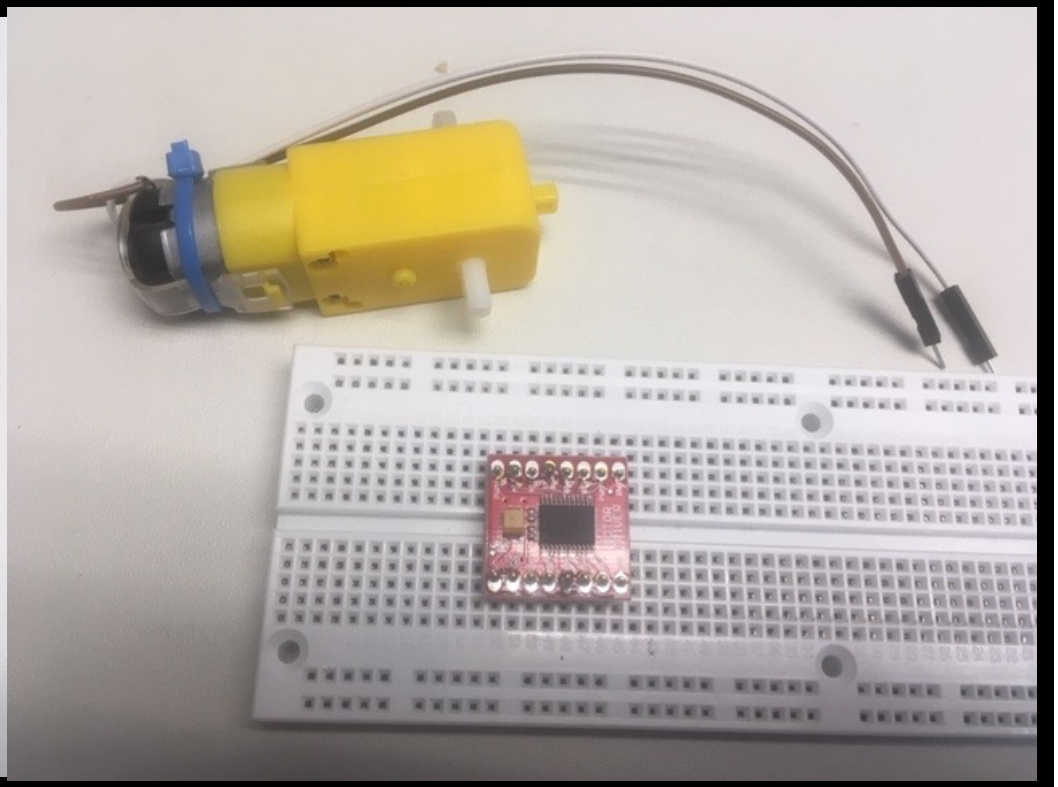
Homework 3 Part 1

Canvas and Cornell Box

Lab 2 Items

Lab 3 items

Adding GPIO Outputs

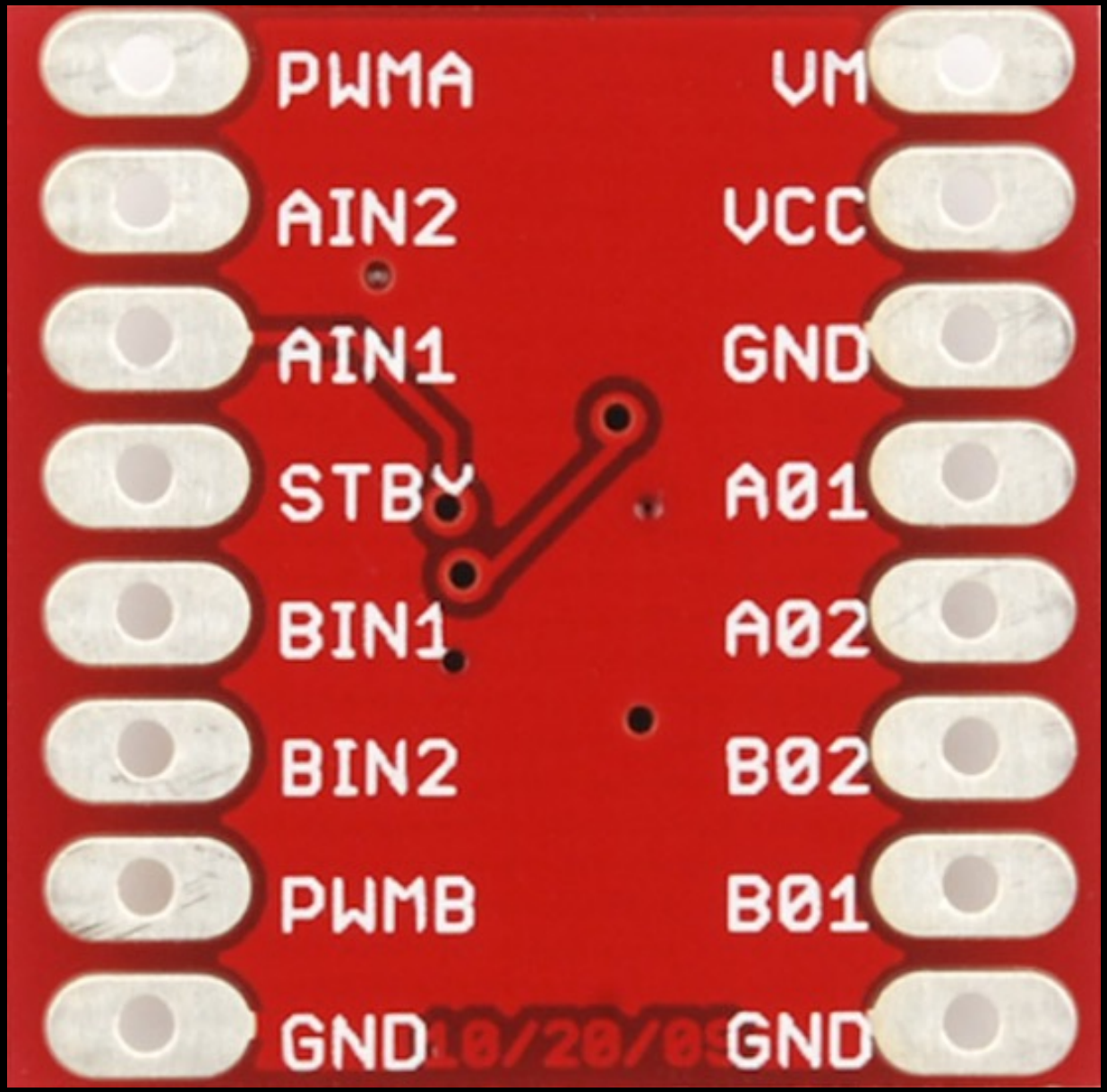


Connections

Input From Rpi: PWM A speed control

Input From Rpi: Motor A direction

Standby!



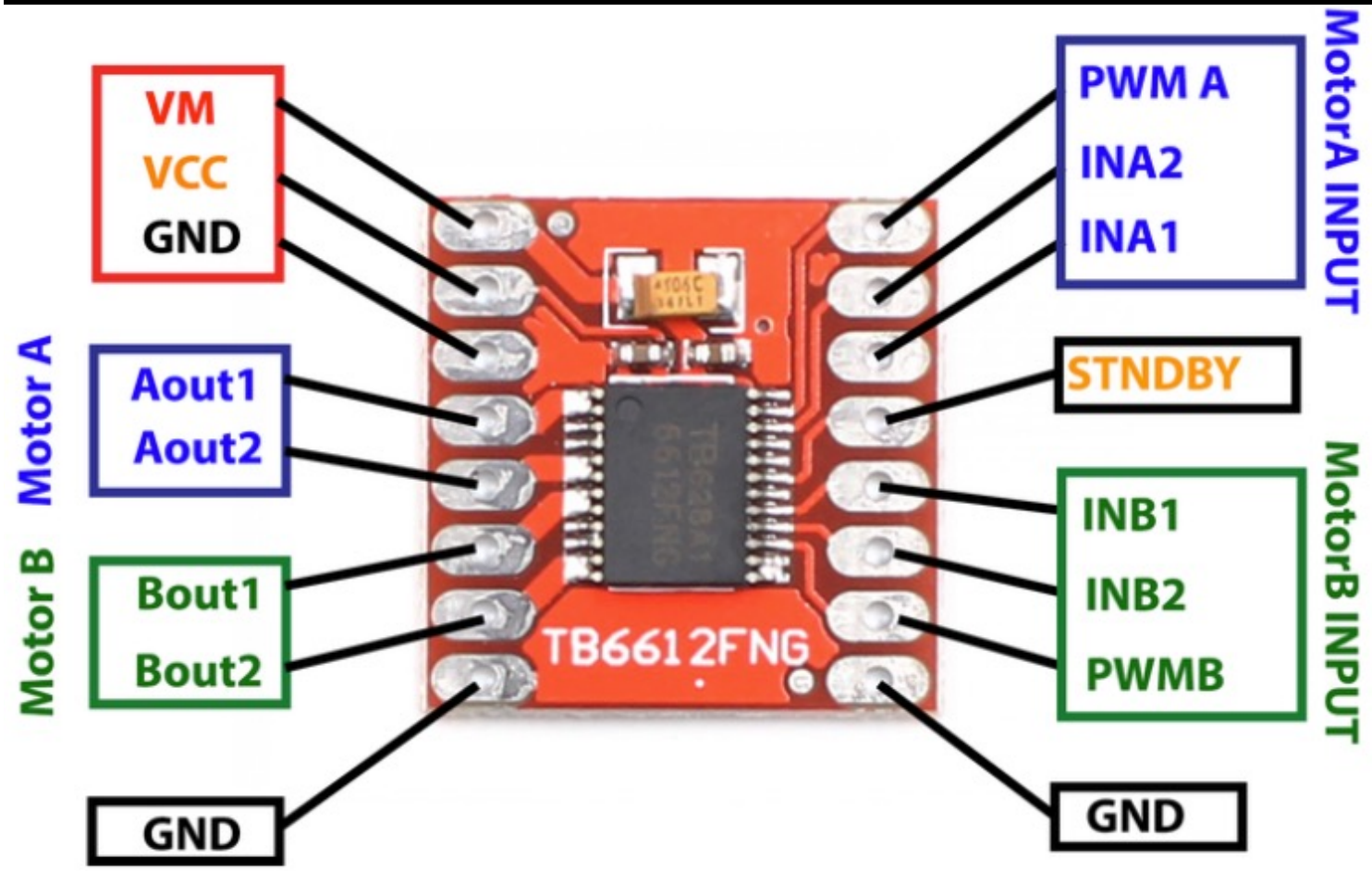
Input: Motor Power

Input: Controller Power

Ground!

Output: To Motor A

Connections



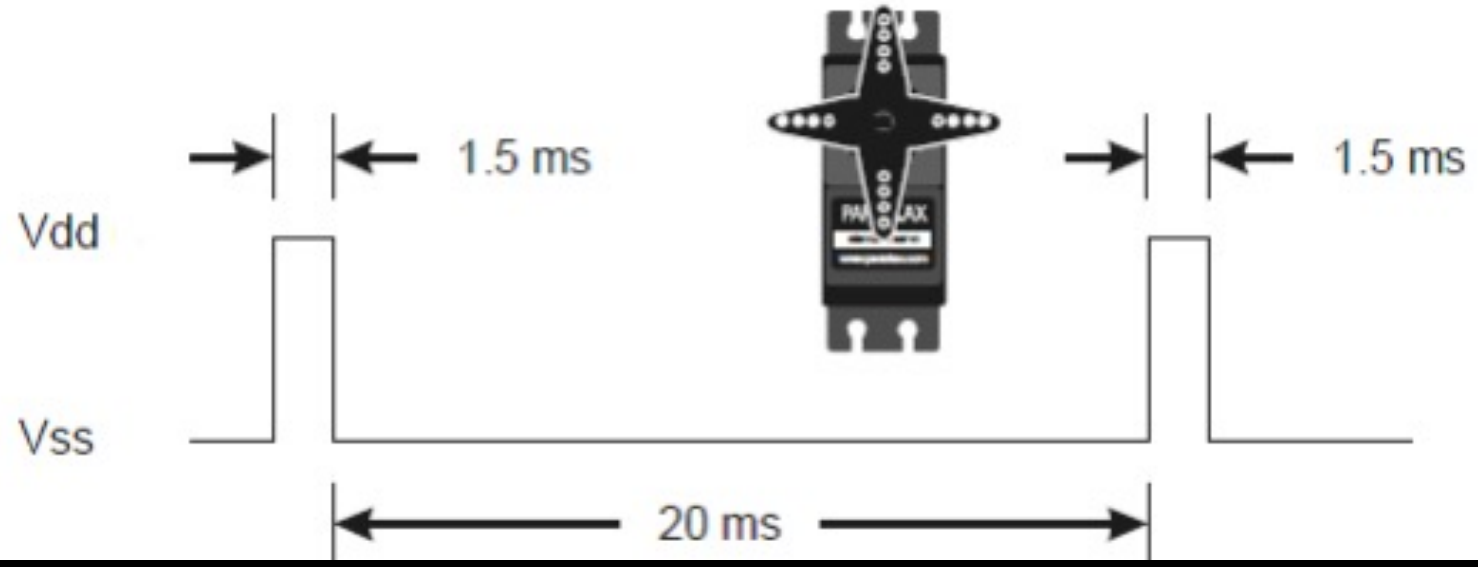
VM	To Power source(15v Max)
VCC	To Logic supply (3.3v/5v)
Aout1 Aout2	To Motor A
Bout1 Bout2	To Motor B
PWMA	Input from MCU(PWM)
INA1	Input from MCU
INA2	Input from MCU
STNDBY	Input from MCU
INB1	Input from MCU
INB2	Input from MCU
PWMB	Input from MCU(PWM)

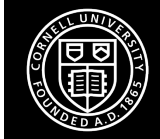
ArtisTech

Lab 3 items - PWM

Communication Protocol

The Parallax Continuous Rotation Servo is controlled through pulse width modulation. Rotational speed and direction are determi ned by the d uration of a high pulse, i n the 1.3--1.7 ms range. In order for smooth rotation, the servo needs a 20 ms pause between pulses. Below is a sample timing diagram for a centered servo:





Output: not PWM

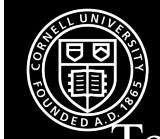
```
#  v1 - blink an LED
#
import time
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)   # Set for broadcom numbering not board numbers...

GPIO.setup(13, GPIO.OUT) # set GPIO 13 as output to blink LED

i = 0
while (i < 10):
    GPIO.output(13, GPIO.HIGH)
    time.sleep(0.5)
    GPIO.output(13, GPIO.LOW)
    time.sleep(0.5)
    i = i + 1

GPIO.cleanup()
```



Output using PWM

To create a PWM instance:

```
p = GPIO.PWM(GPIO_pin, frequency)
```

To start PWM:

```
p.start(dc)   # where dc is the duty cycle (0.0 <= dc <= 100.0)
```

To change the frequency:

```
p.ChangeFrequency(freq)   # where freq is the new frequency in Hz
```

To change the duty cycle:

```
p.ChangeDutyCycle(dc)  # where 0.0 <= dc <= 100.0
```

To stop PWM:

```
p.stop()
```

Note that PWM will also stop if the instance variable 'p' goes out of scope

Modular Code

blink.py

motor_control.py

two_wheel.py

rolling_control.py

Comparisons

Embedded systems

Real Time Systems