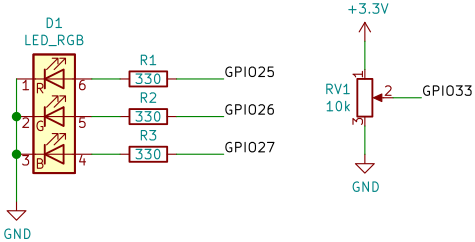
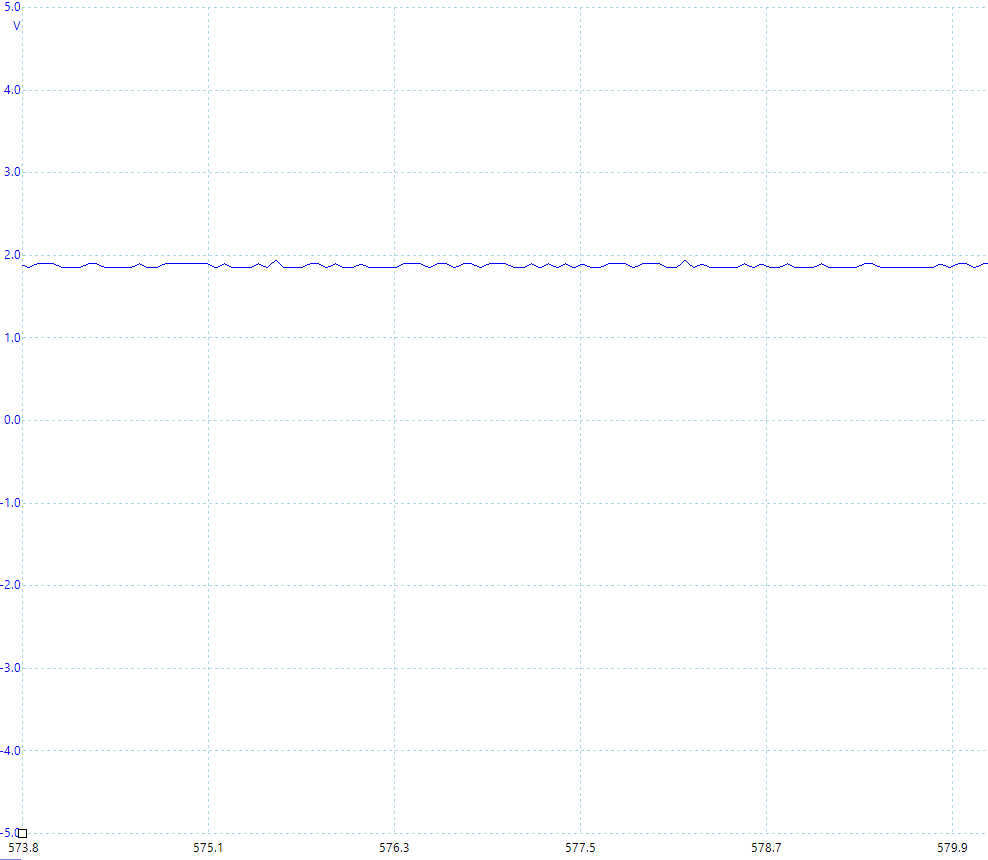
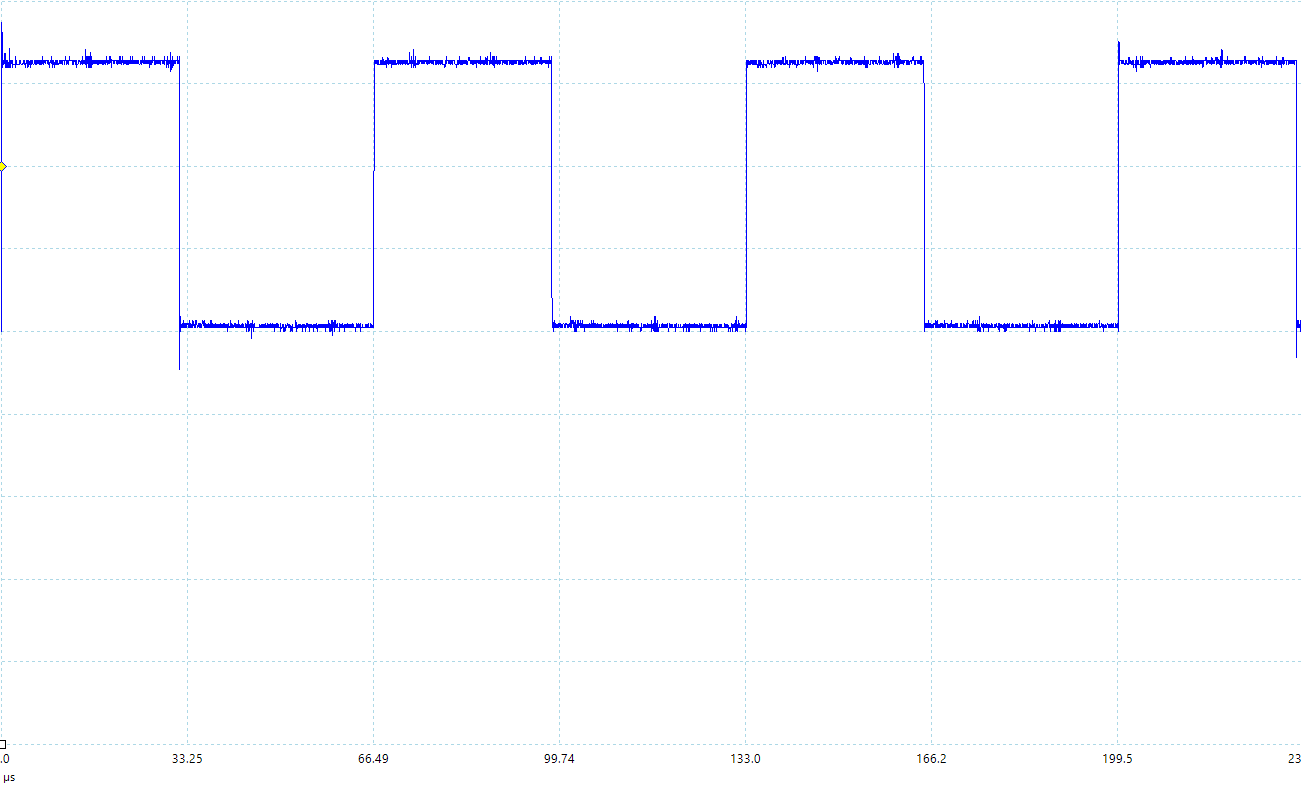
**Part A. RGB LED with PWM**

**A1. Wire up the follow circuit and upload** RGB\_PWM.ino**. For RGB LED, the longest leg is the common cathode (negative). Other legs, you may test which one is which color.**

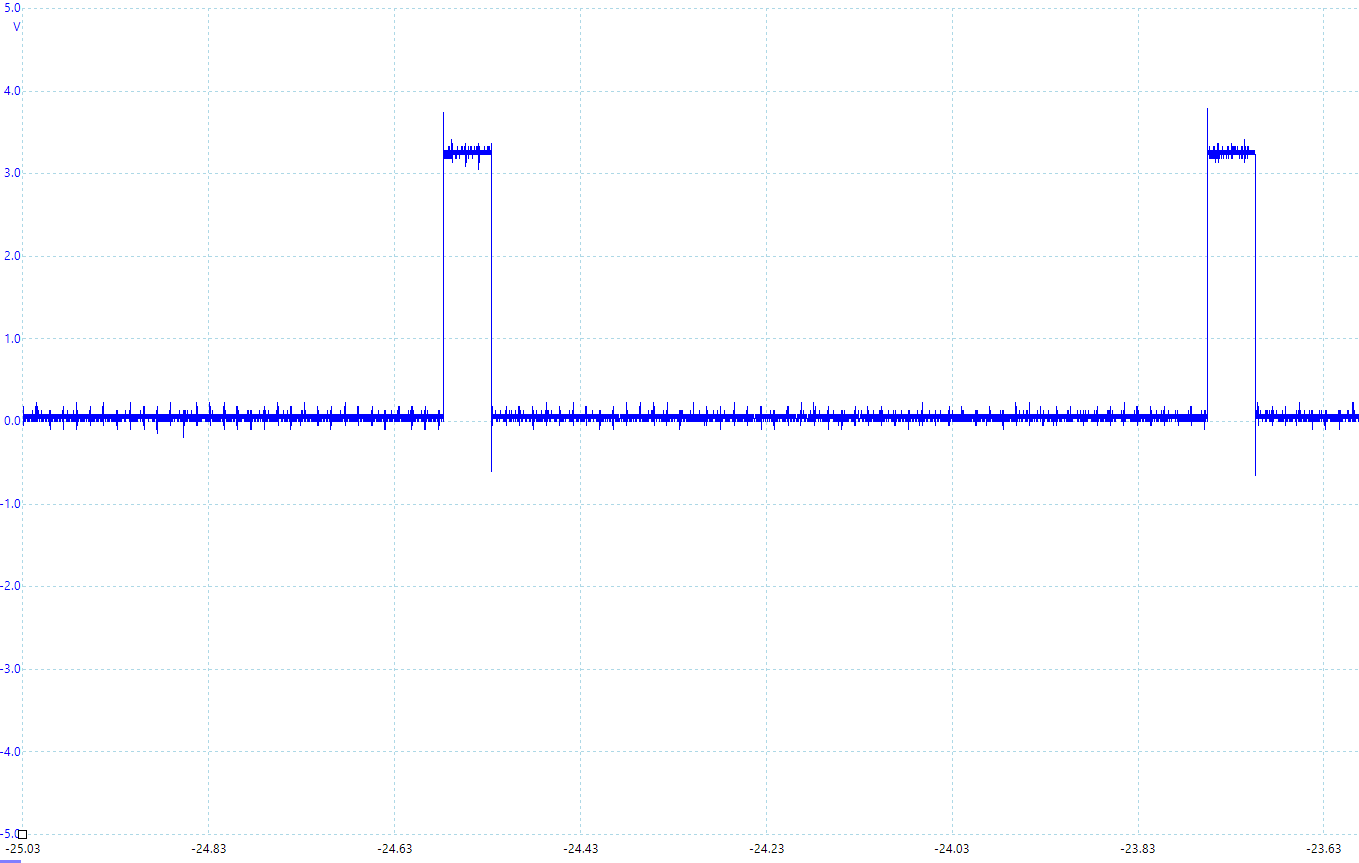


**Turn the potentiometer to around middle and use Picoscope to measure GPIO25 and GPIO33. Capture the waveforms and paste the 2 pictures below. Adjust the voltage range and time range so that you see proper waveform.**



**---**

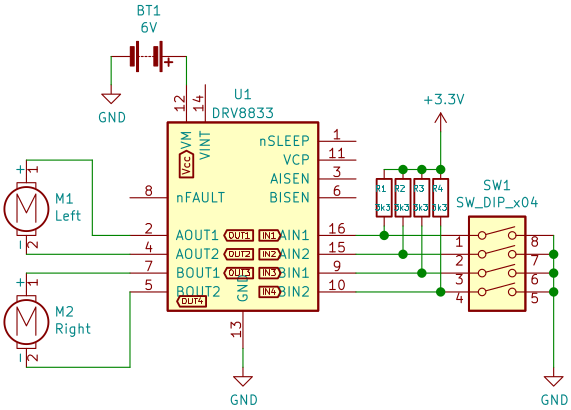
**What happens if you change PWM resolution of PWM in the source code to 16 bits? Try sweeping the potentiometer from minimum to maximum while capturing the waveform at GPIO25. Compare this to the previous configuration (12-bit PWM).**



**Part B. DC Motor**

**B1. Wire up the circuit below. SW1 is a DIP switch. The Vcc 3.3V comes from your ESP32. You have to also connect the ground of 6V battery to the ground of ESP32.**

**Answer the switch configurations in the following questions in** [AIN1, AIN2, BIN1, BIN2] **format. For example,** [close, open, open, close] **means that you turn on AIN1, turn off AIN2, turn off BIN1, turn on BIN2. Close means close circuit. Open means open circuit.**



**What switch configuration makes M1 and M2 turn forward?**

Open close open close

**What switch configuration makes M1 turn forward and M2 turn backward?**

Open close close open

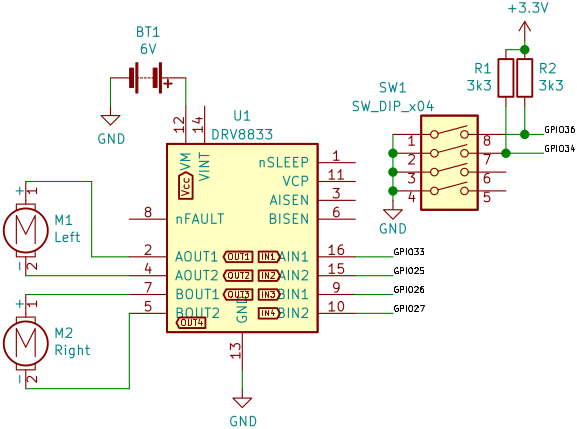
**What switch configuration makes M1 turn backward and M2 stop?**

Close open open open

**What switch configurations makes M1 and M2 stop? There are more than one configuration.**

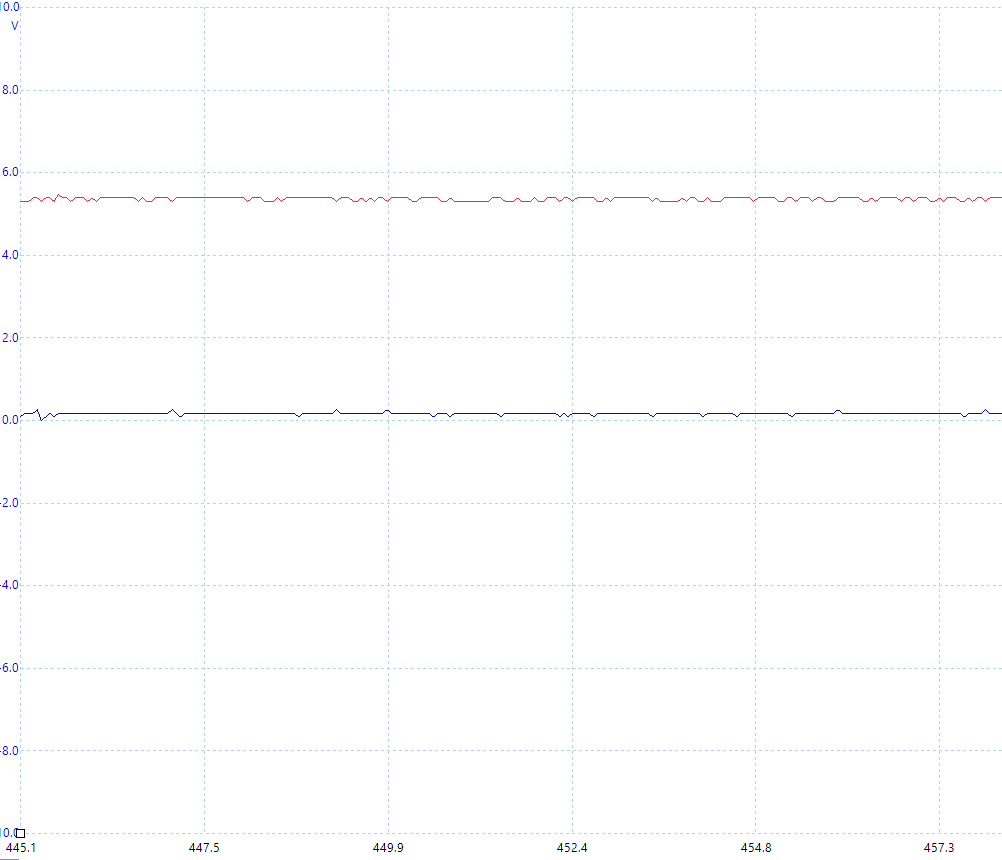
Open open open open open or close close close close close

**B2. Wire up the following circuit. Upload** DC\_Motor\_1.ino**.**

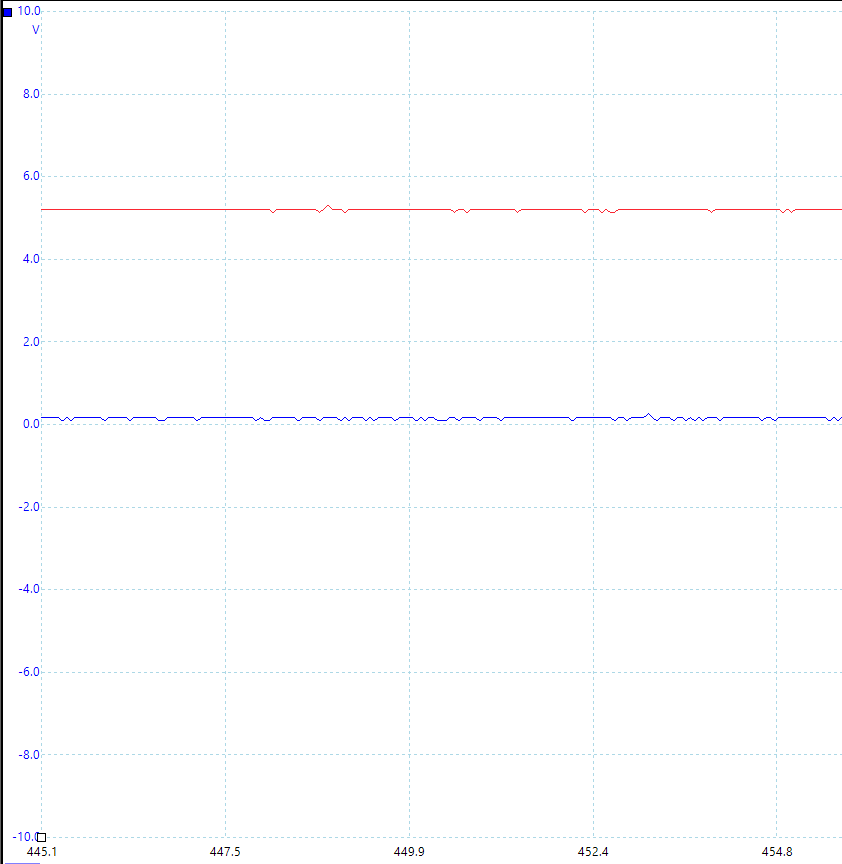


**When SW1 is all open, use the Picoscope to measure GPIO33, GPIO25, GPIO26, and GPIO27. Adjust the voltage range and time range so that you see proper waveform. Capture the waveform of both pins and paste them below.**

**GPIO33, GPIO25**



**GPIO26, and GPIO27**



**From the above configuration, what direction does the M1 and M2 turn (forward or backward)?**

**If the motor turns in the opposite direction to the configuration in source code, you can change your source code so that it corresponds to the real physical direction.**

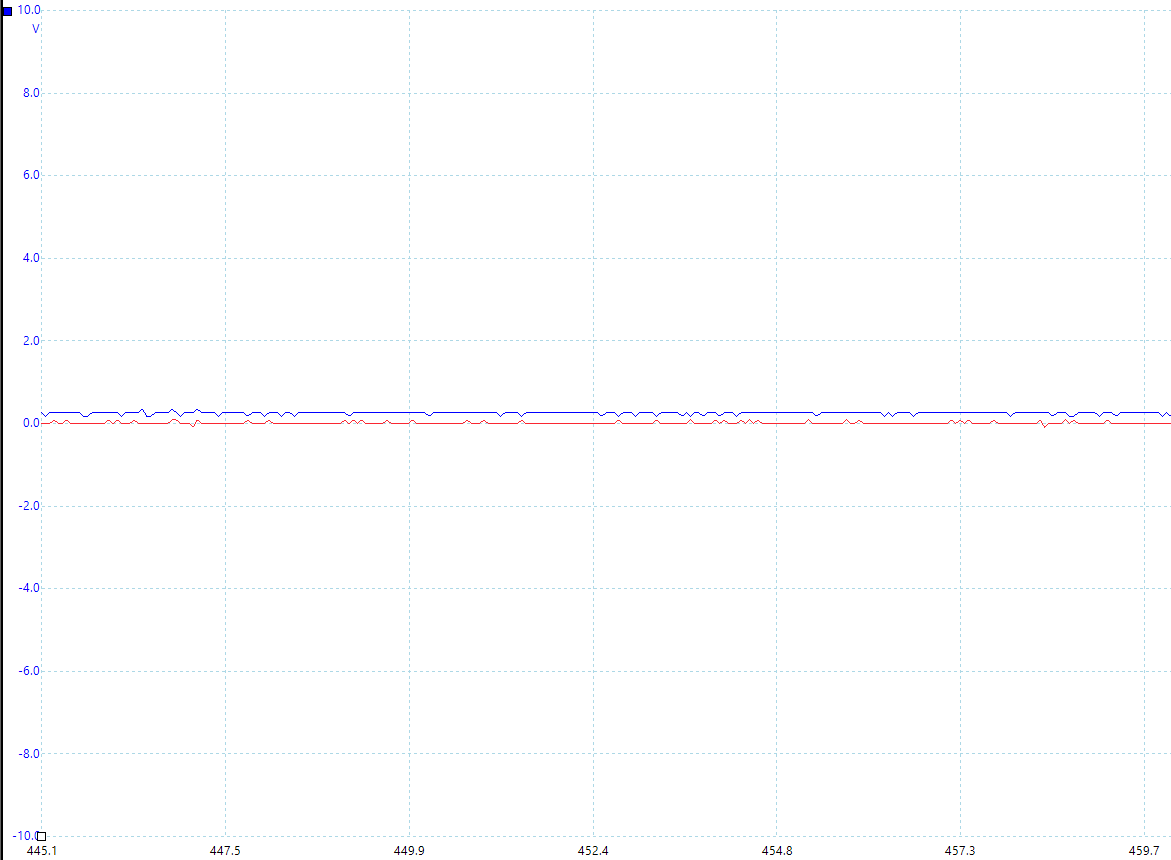
**--- Forward**

**Try changing the duty cycle to some number between 0 and 100 (in percents), and observe the speed of the motors. What is the relationship between the duty cycle and the speed of the motor?**

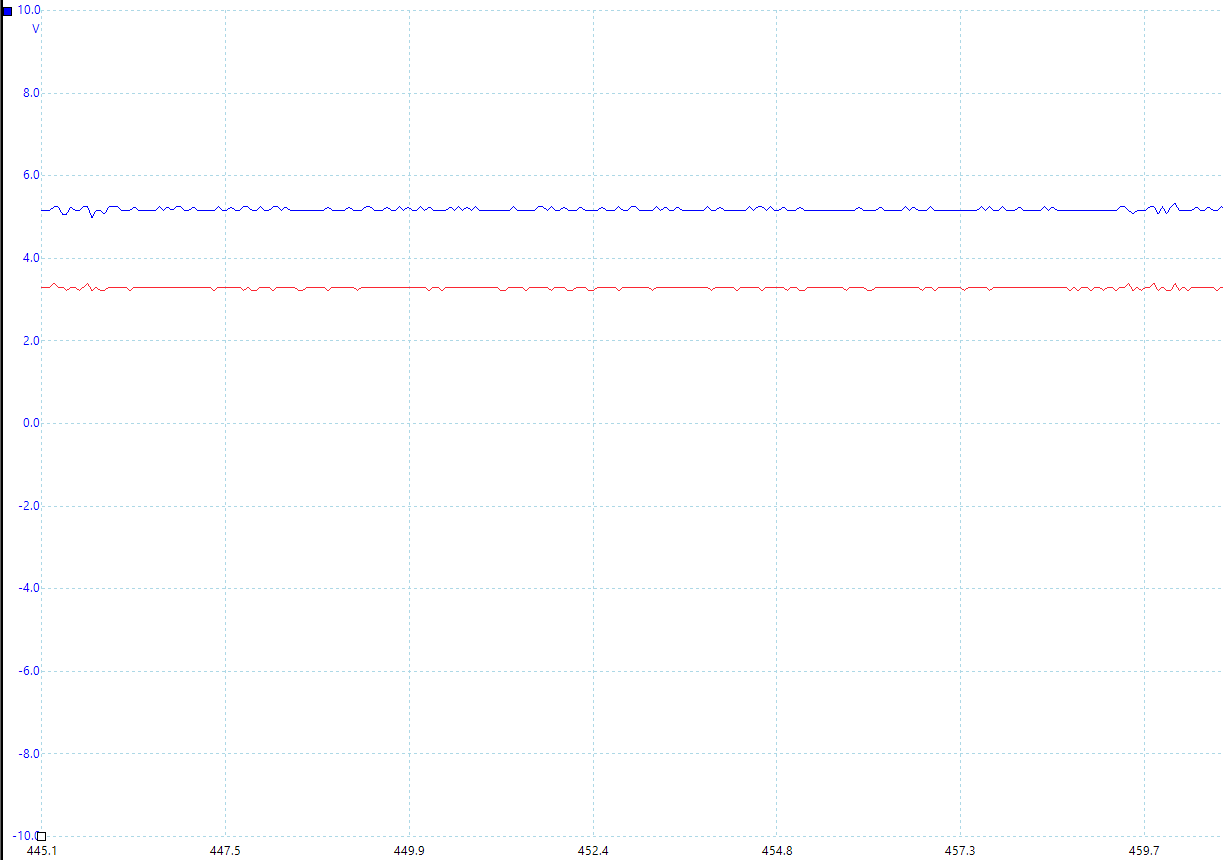
**--- Higher duty cycle, higher speed of moter Lower duty cycle, lower speed of motor**

**Use Picoscope to measure AIN1, AIN2, AOUT1, and AOUT2. Paste the 4 captured pictures below.**

**--- AIN1(RED), AOUT1(BLUE)**



**AIN2(RED), AOUT2(BLUE)**



**What is an obvious difference between waveforms at inputs and outputs that you can observe from Picoscope?**

When inputs waveform rise, output waveform also rise.

Input waveform voltage lower than output waveform voltage when rising.**B3. Using the circuit in B2 and also add a potentiometer, write a program so that you can control the speed of the motor via the potentiometer. You can use some knowledge from A1. Copy and paste your finished program below.**

#define *IN\_1* 33 // Pin attached to IN1 of DRV8833 module

#define *IN\_2* 25 // Pin attached to IN2 of DRV8833 module

#define *IN\_3* 26 // Pin attached to IN3 of DRV8833 module

#define *IN\_4* 27 // Pin attached to IN4 of DRV8833 module

#define *START\_STOP* 36

#define *FWD\_BWD* 34

#define *FWD* 0

#define *BWD* 1

#define *PWM\_CH\_1* 0

#define *PWM\_CH\_2* 1

#define *PWM\_CH\_3* 2

#define *PWM\_CH\_4* 3

#define *PWM\_FREQ* 15000

#define *PWM\_RES* 12 // Resolution in bits

#define *ADC* 32

#define *MAX\_ADC* 4095

#define *MIN\_CYCLE* 60

const int PWM\_MAX = *pow*(2, *PWM\_RES*) - 1;

int dutyCycle = 100; // percents

int adc = 0;

void *setup*()

{

*ledcSetup*(*PWM\_CH\_1*, *PWM\_FREQ*, *PWM\_RES*);

*ledcSetup*(*PWM\_CH\_2*, *PWM\_FREQ*, *PWM\_RES*);

*ledcSetup*(*PWM\_CH\_3*, *PWM\_FREQ*, *PWM\_RES*);

*ledcSetup*(*PWM\_CH\_4*, *PWM\_FREQ*, *PWM\_RES*);

*ledcAttachPin*(*IN\_1*, *PWM\_CH\_1*);

*ledcAttachPin*(*IN\_2*, *PWM\_CH\_2*);

*ledcAttachPin*(*IN\_3*, *PWM\_CH\_3*);

*ledcAttachPin*(*IN\_4*, *PWM\_CH\_4*);

*pinMode*(*START\_STOP*, INPUT);

*pinMode*(*FWD\_BWD*, INPUT);

}

void *loop*()

{

    adc = *analogRead*(*ADC*);

    dutyCycle = 60 + ((40 \* adc) / 4095);

    if (*digitalRead*(*START\_STOP*))

    {

        if (*digitalRead*(*FWD\_BWD*) == *FWD*)

        {

*moveForward*(*PWM\_CH\_1*, *PWM\_CH\_2*, dutyCycle);

*moveForward*(*PWM\_CH\_3*, *PWM\_CH\_4*, dutyCycle);

        }

        else

        {

*moveBackward*(*PWM\_CH\_1*, *PWM\_CH\_2*, dutyCycle);

*moveBackward*(*PWM\_CH\_3*, *PWM\_CH\_4*, dutyCycle);

        }

    }

    else

    {

*stopMotor*(*PWM\_CH\_1*, *PWM\_CH\_2*);

*stopMotor*(*PWM\_CH\_3*, *PWM\_CH\_4*);

    }

}

void *moveForward*(int motorPositive, int motorNegative, int dutyCycle)

{

    int pwmValue = dutyCycle / 100.0 \* PWM\_MAX;

    /\* The current flows from positive to negative. \*/

*ledcWrite*(motorPositive, pwmValue);

*ledcWrite*(motorNegative, 0);

}

void *moveBackward*(int motorPositive, int motorNegative, int dutyCycle)

{

    int pwmValue = dutyCycle / 100.0 \* PWM\_MAX;

    /\* The current flows from negative to positive. \*/

*ledcWrite*(motorPositive, 0);

*ledcWrite*(motorNegative, pwmValue);

}

void *stopMotor*(int motorPositive, int motorNegative)

{

*ledcWrite*(motorPositive, 0);

*ledcWrite*(motorNegative, 0);

}