ECE 350: Embedded Systems and Hardware Control

Lab 3 Servo Actuator Control and Proximity Detection

Name: Huy Tran

Submission date: March 20, 2022

1. **Object:**

In this lab, Python is used to demonstrate hardware control using an analog servo, the Raspberry Pi 3, and how to detect an object using an ultrasonic proximity sensor. By the end of this lab, we will be able to continue using the GPIO (General Purpose Input/Output) Python library, as well as how to interface and control a PWM (pulse width modulation) actuator using Raspberry Pi.

1. **Background**

Given the PWM=20ms and frequency is 50Hz. Thus, to calculate the angle equivalent to the PWM period at 0.5ms to 2.5 ms:

0.5ms/20ms\*100%= 2.5% at zero degress

1ms/20ms\*100%=5% at 45 degrees

1.5ms/20ms\*100%=7.5% at 90 degrees

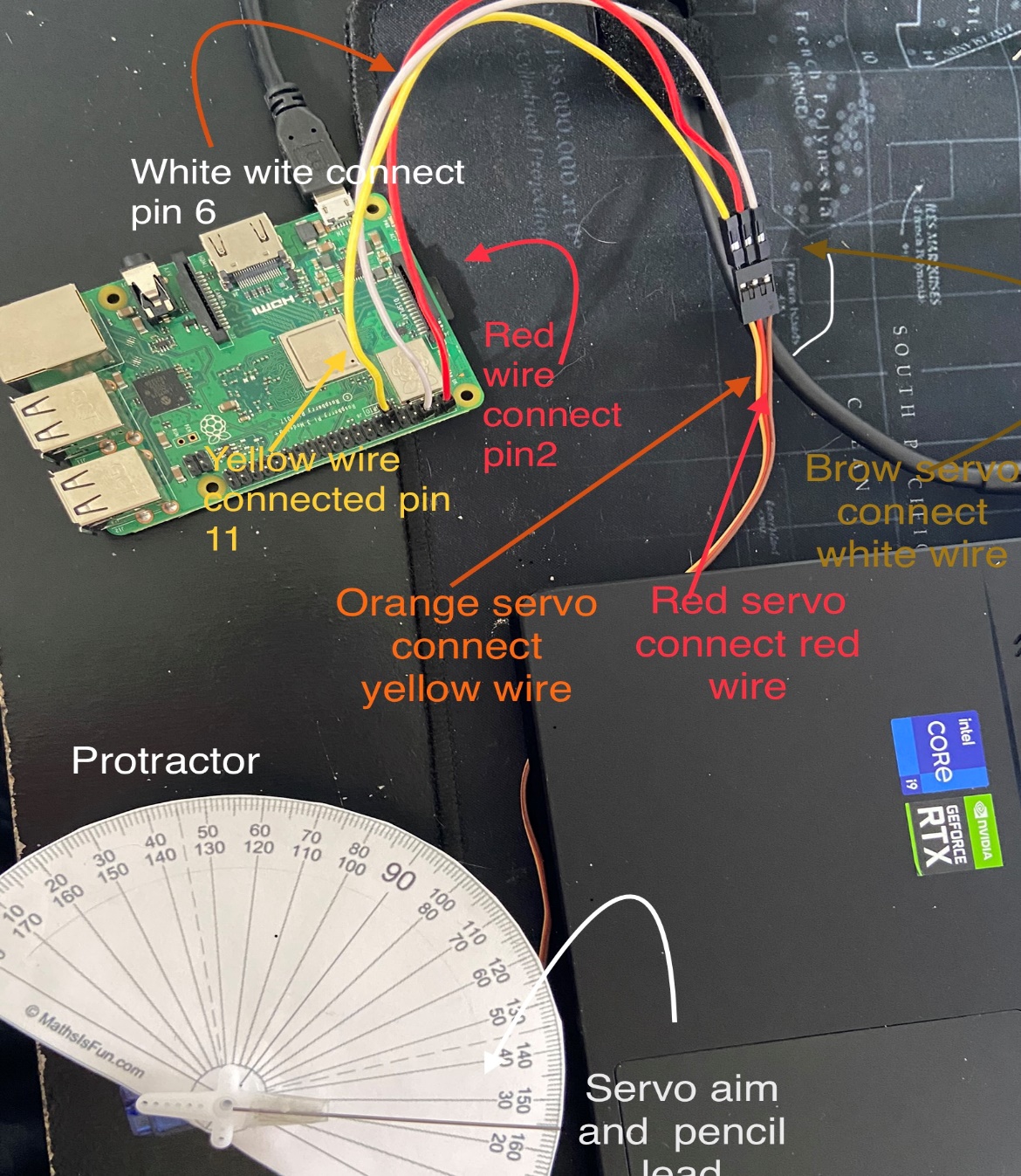
2ms/20ms\*100%=10% at 130 degrees

2.5ms/20ms\*100%=12.5% at 180 degrees

1. **Material and equipment**

* Raspberry Pi 3
* Tower Pro SG90 micro-servo in the ECE350 electronics kit
* Ultrasonic Distance Sensor (HC-SR04)
* Jumper wires (male-female)
* Breadboard
* Paper/printer
* Ruler/measuring tape (at least 1m long)
* Scissors, scotch tape/glue, pencil lead (or another object to attach to servo arm)

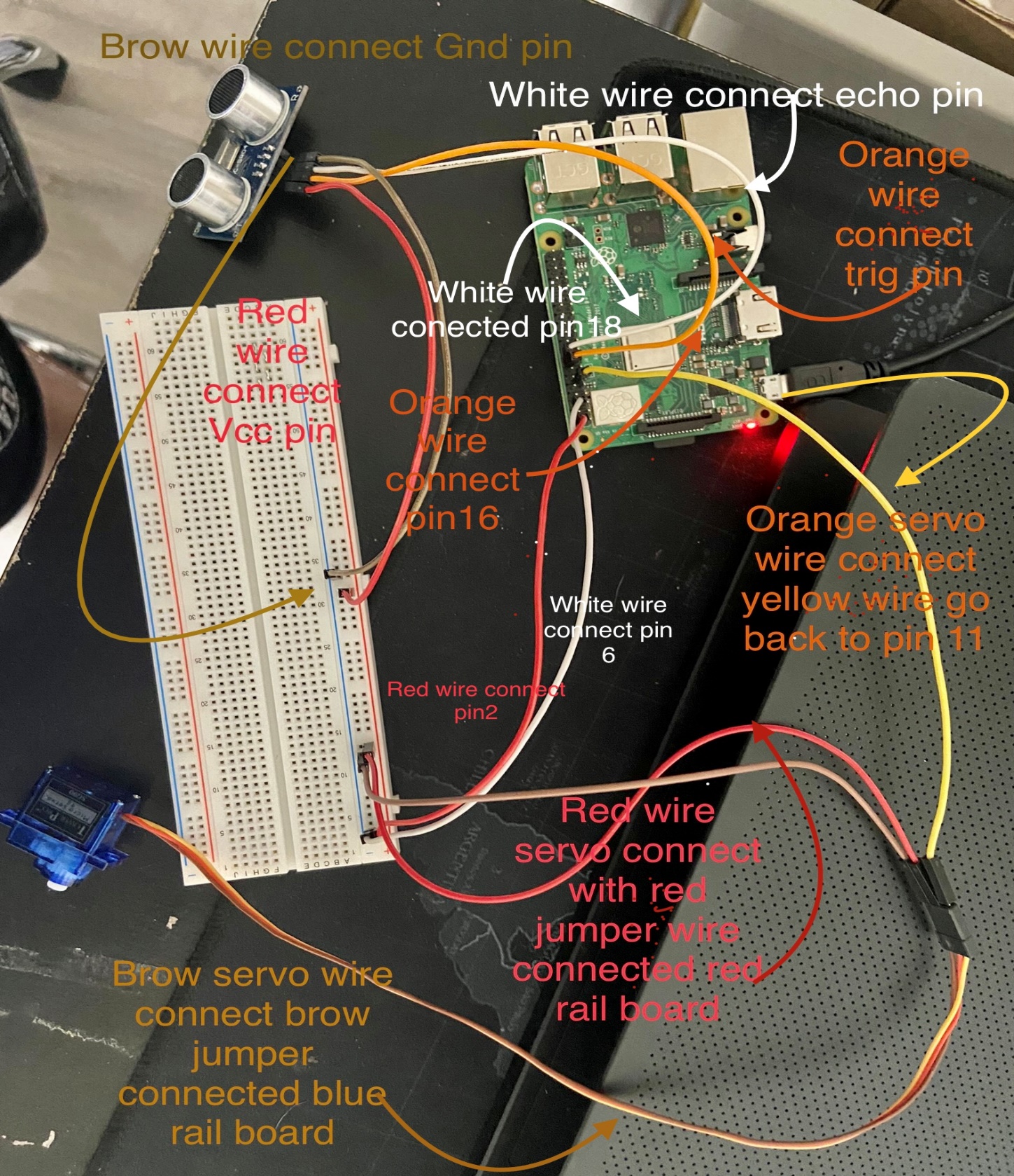
1. **Experiment set up circuit:**



**Picture1: set up a circuit for part1 and 2**

So, for picture1, we will have the same circuit build for part2 and part3. we just connect the servo motor with raspberry pi3. The servo motor has 3 wires (orange servo wire connected with the yellow jumper wire which another side is connected with pin 11 GPIO), (red servo wire connected with the red jumper wire which another side is connected with pin 2 (5v)), (brow servo wire connected with the brow jumper wire which another side is connected with pin 6 (3.3V)).

We add the protractor paper between the servo motor and servo aim which is attached pencil lead.



**Picture2: set up a circuit for part3**

In picture 2: the Servo motor has 3 wires: red wire connected with red (+) rail on board, the yellow wire connected with pin11 of RPi, and the grow wire connected with blue (-)rail on board. Ultrasonic sensors have 4 wires: VCC red wire connected with red (+) rail on board. TRIG (orange wire) connected with pin16 of RPi. Next, the Echo (white wire) connected with pin18 of RPi. Last, GND (brow wire) connected with blue (-) rail on board.

1. **Python script**

**Lab3\_part2 script:**

import RPi.GPIO as GPIO

import time

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT)

p = GPIO.PWM(12,50)

p.start(2.5)

time.sleep(3)

try:

    while True:

        p.ChangeDutyCycle(7.5)  # turn 90 degree

        time.sleep(3)           #sleep 1 second

        p.ChangeDutyCycle(2.5)  # turn 45 degree

        time.sleep(3)

        p.ChangeDutyCycle(12.5)  # turn 180 degree

        time.sleep(3)           #sleep 1 second

        p.ChangeDutyCycle(2.5)  # turn 45 degree

        time.sleep(1)

except KeyboardInterrupt:

    p.stop()

    Gpio.cleanup()

**Lab3\_part2 script:**

import RPi.GPIO as GPIO

import time

import atexit

atexit.register(GPIO.cleanup)

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT)

GPIO.setwarnings(False)

p = GPIO.PWM(12,50)

p.start(0)

time.sleep(2)

while(True):

    for i in range(0,180,10):

        p.ChangeDutyCycle(2.5 + 10 \* i / 180)

        time.sleep(1)

        p.ChangeDutyCycle(0)

        time.sleep(1)

    for i in range(180,0,-10):

        p.ChangeDutyCycle(2.5 + 10 \* i / 180)

        time.sleep(1)

        p.ChangeDutyCycle(0)

        time.sleep(1)

servo1.stop()

GPIO.cleanup()

**Lab3\_part3 script:**

import RPi.GPIO as GPIO

import time

def checkdist():

    GPIO.output(16, GPIO.HIGH)

    time.sleep(0.00015)

    GPIO.output(16, GPIO.LOW)

    while not GPIO.input(18):

        pass

    t1= time.time()

    while GPIO.input(18):

        pass

    t2=time.time()

    return (t2-t1)\*340/2

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.OUT)

GPIO.setup(16,GPIO.OUT, initial = GPIO.LOW)

GPIO.setup(18,GPIO.IN)

time.sleep(2)

p=GPIO.PWM(11,50)

p.start(2.5)

time.sleep(2)

try:

    while True:

        print (checkdist())

        time.sleep(0.5)

        if checkdist() <= 0.25:

            p.start(5.0)

            time.sleep(3)

        if checkdist() >0.25 and checkdist() <=0.5:

            p.start(7.5)

            time.sleep(3)

        if checkdist() >0.5 and checkdist() <=0.75:

            p.start(10)

            time.sleep(3)

        if checkdist() >0.75 and checkdist() <= 1:

            p.start(12.5)

            time.sleep(3)

except KeyboardInterrupt:

    p.stop()

    Gpio.cleanup()

1. **Screen captures/photos of the oscilloscope traces showing the four PWM signal in part2 section(7)**

Chart, scatter chart

Description automatically generated

**a table of these measurements from 0 to 180 degrees in increments of 10 degrees.**



**Picture 3: Pulse Width = 0.5 ms** (0.5ms = 500μs)

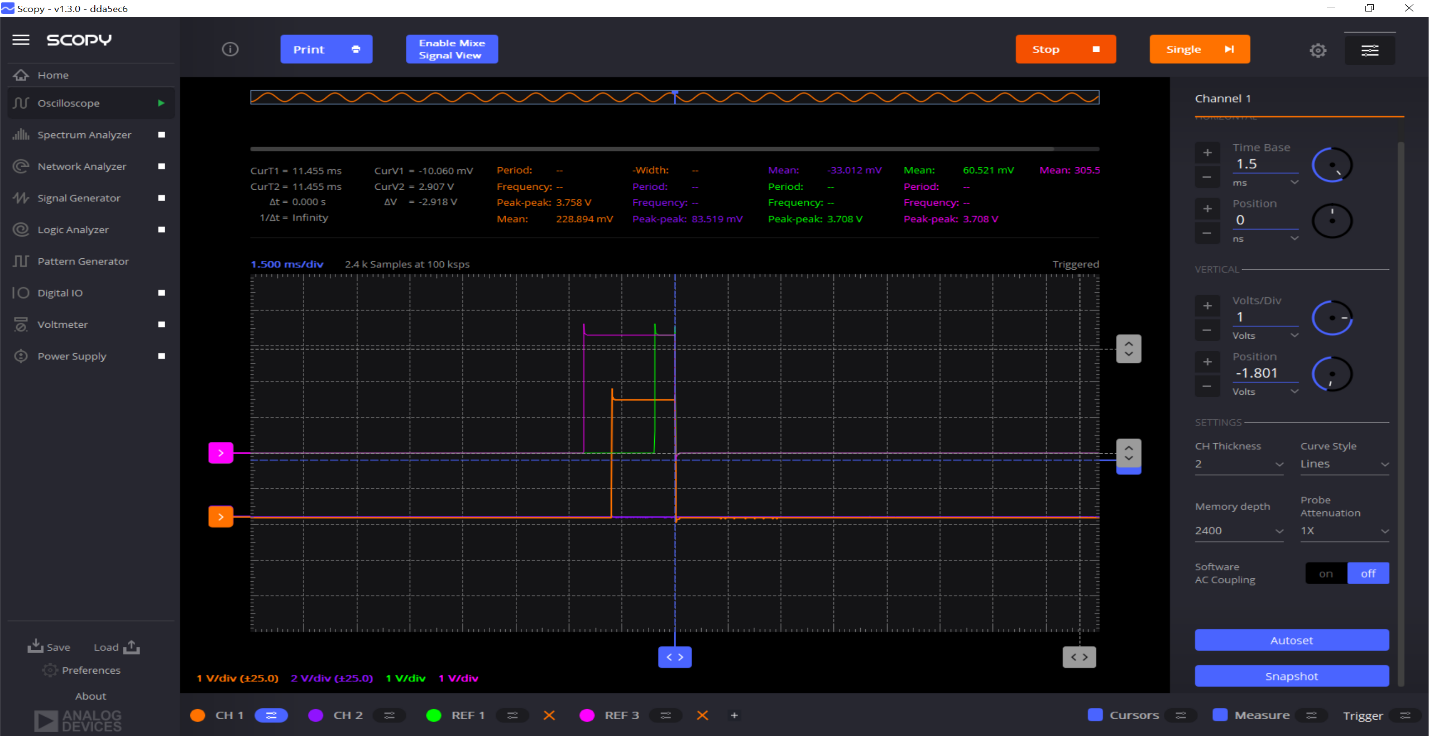
The PWM pulse train started from 0 degrees to 180 degrees ( the wide range of chart increased from -0.58 to-2.58 ms). When it comes back from 180 to 0 degrees(the wide range of chart decrease from -2.58 to -0.5 ms). See the picture below ( blue light is increasing from -0.58 ms and green light is decreasing from -2.58 ms). The time base is 0.5ms and has a duty cycle for a 4.2 square box.

A screenshot of a computer

Description automatically generated with medium confidence

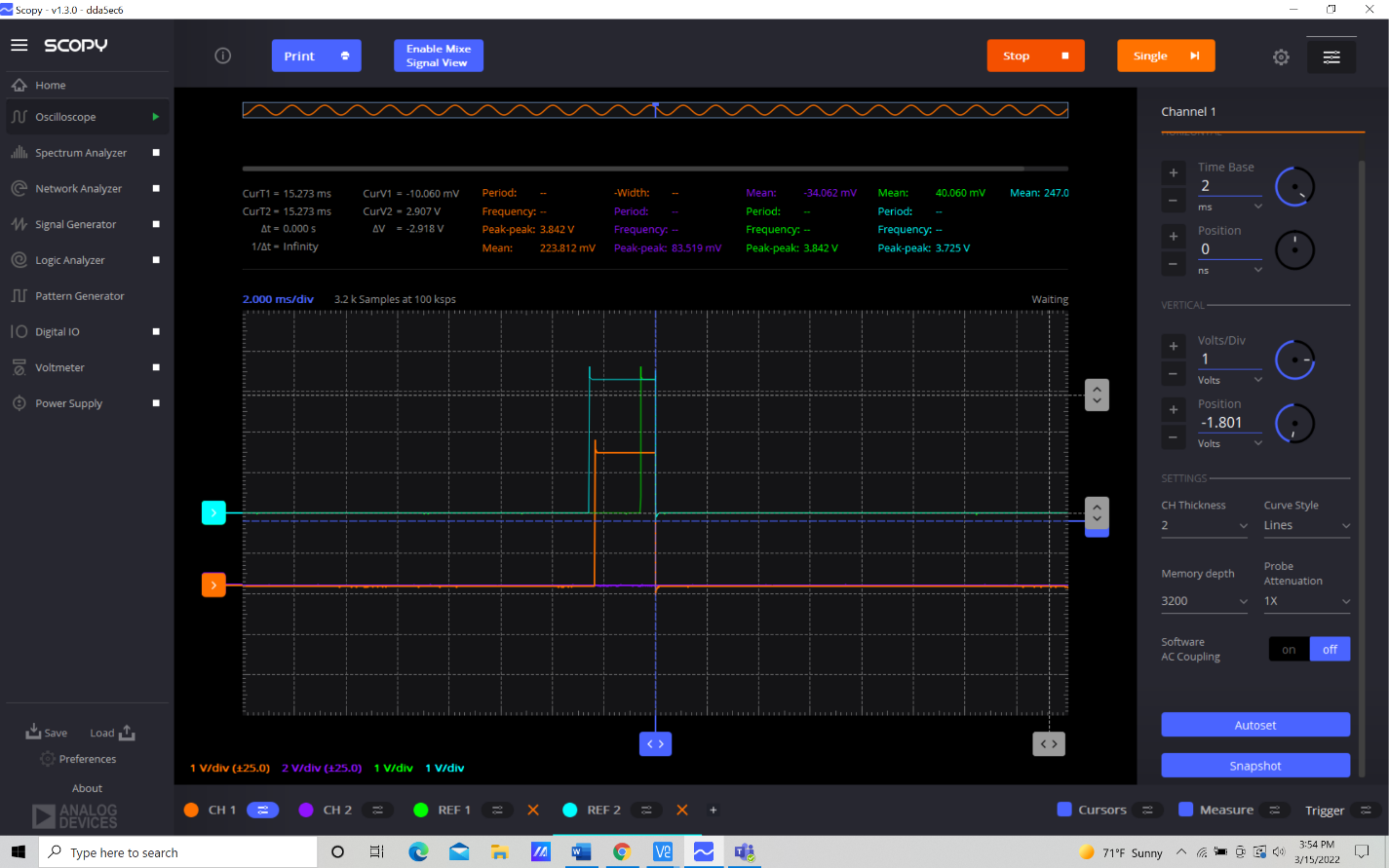
**Picture 4: Pulse Width = 1 ms**

The time base was changed to 1ms. The PWM pulse train started from -0.57ms (zero degrees) to -2.7ms (180 degrees) and then it goes back to -0.55ms (zero degrees) like figure 2 above. But compared with figure 1 we saw that the width of figure 2 (2.3 square) is shorter than the width of figure 1(around 5 square). The time base is 1ms and has a duty cycle for a 2 square box.



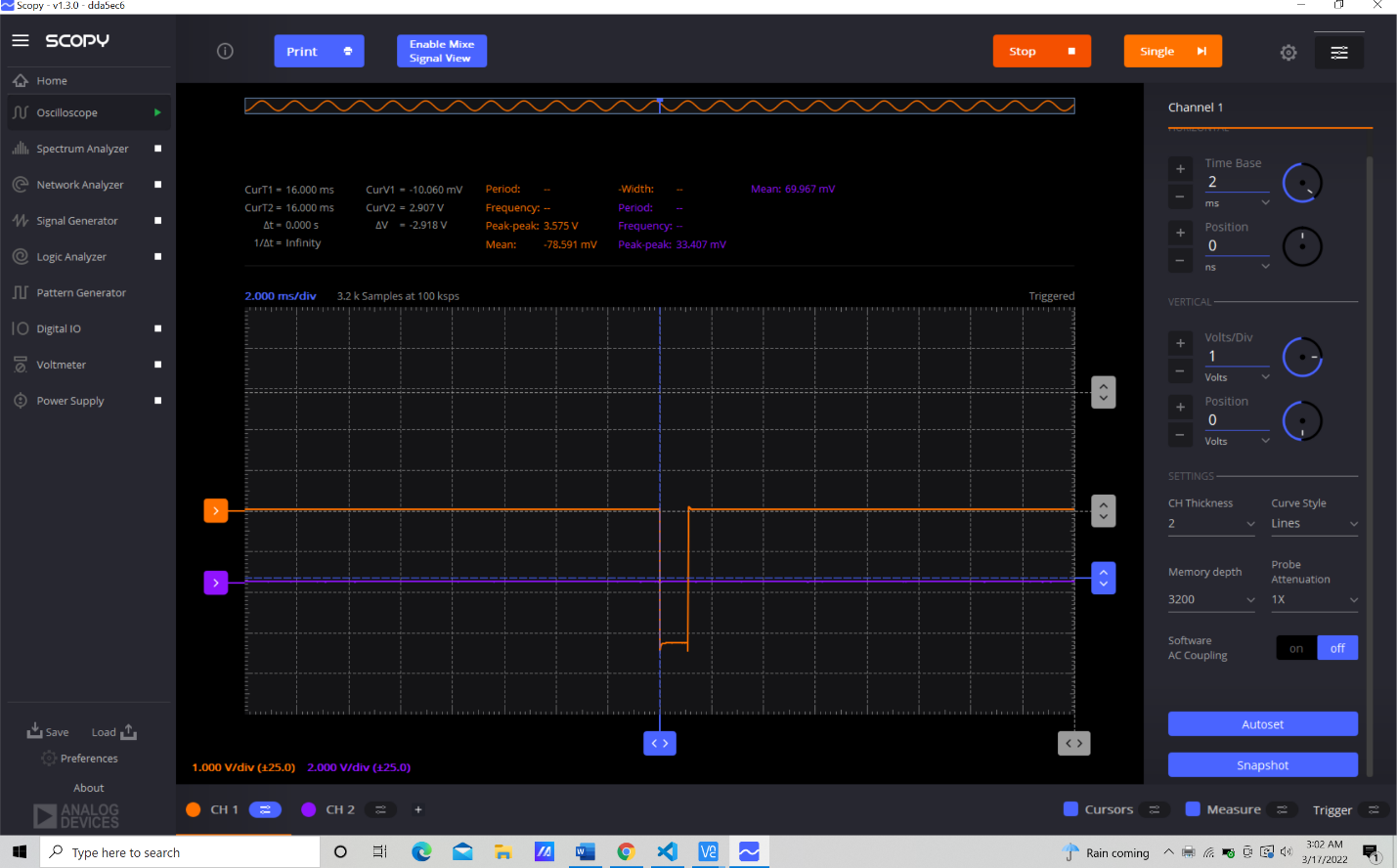
**Picture 5: Pulse Width = 1.5 ms**

Now the time base will be 1.5ms. The PWM pulse train is the same as the figure1&2. However, the duty cycle is shorter than figure1&2. The time base is 1.5 ms and has a duty cycle for a 1.5 square box.



**Picture 6: Pulse Width = 2 ms**

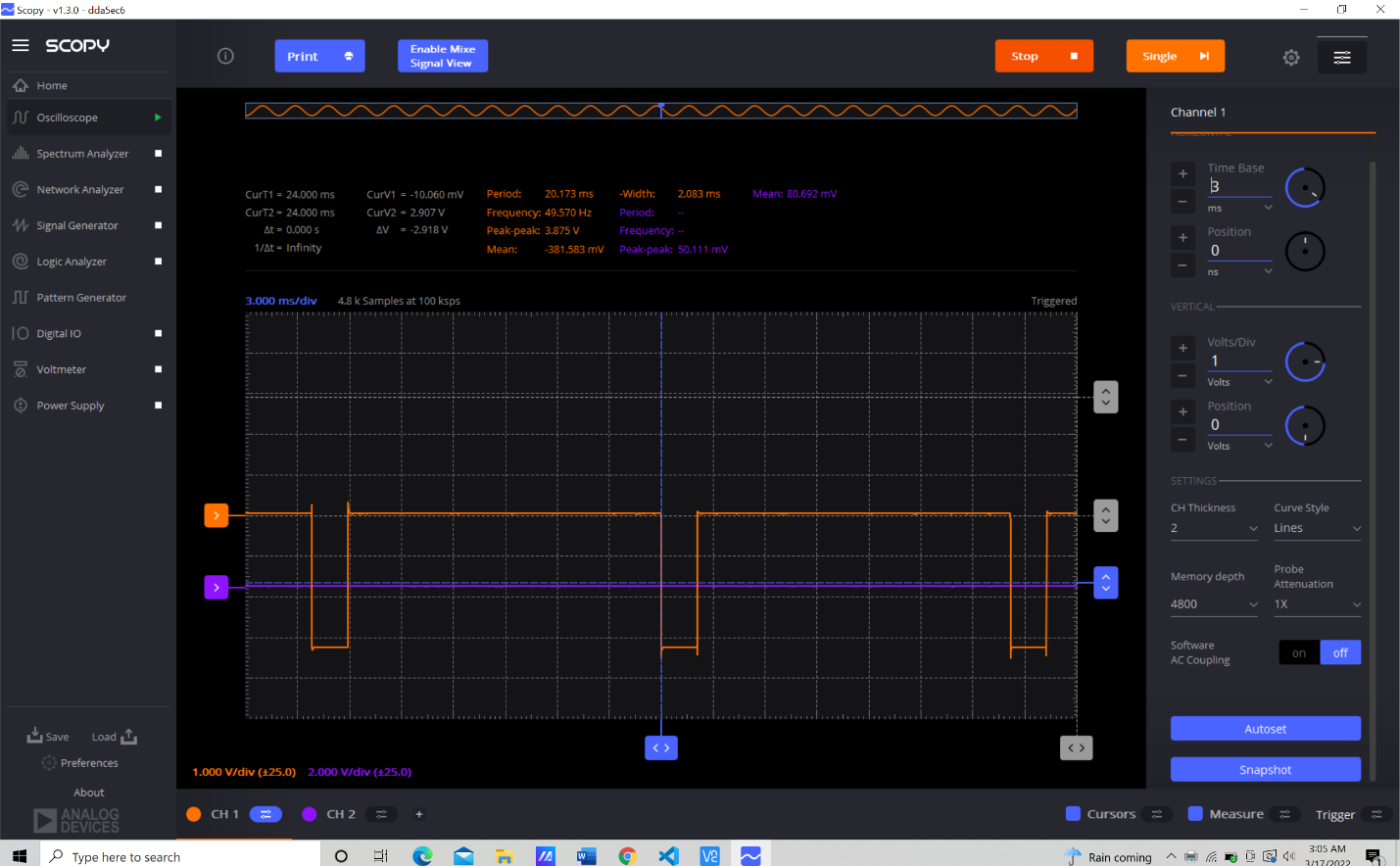
The time base changed to 2ms. PWM pulse train is the same as the figure1&2. However, the duty cycle is shorter than figure1&2&3. The time base is 2ms and has a duty cycle for a 1 square box.

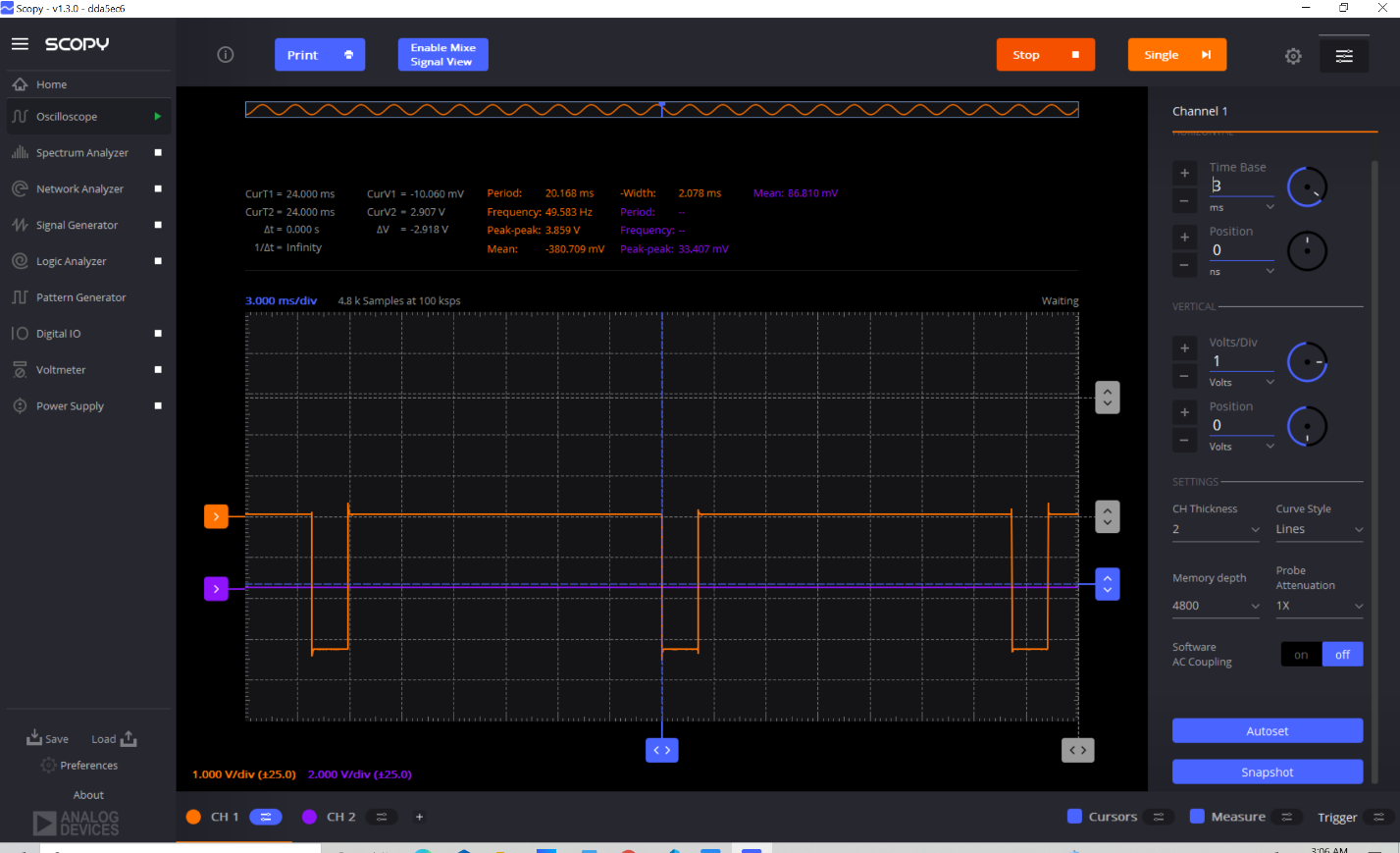
1. **Screen captures/photos of the oscilloscope traces showing the four PWM signal in part3 section9.**

**Picture 7: at a distance ≤0.25m (width is 1.12ms)A screenshot of a computer

Description automatically generated**

**Picture 8: at a distance >0.25m and ≤0.50m( width 1.6ms)**

****

**Picture 9: at a distance between >0.5m and ≤0.75m (width is 2.1ms)**

**Picture 10: at a distance >0.75m (width is 2.6ms)**

1. **Conclusion**

The data was linear, and the differences were very slight due to noise or tape holding it in place. However, this, and other errors would have a noticeable effect on any application in which the angle of the servo needs to be accurate.