F1/10

TeleOp

Madhur Behl
CS4501/SYS4852

madhur.behl@virginia.edu

Welcome back !

Assignment 3 is out – Keyboard control of the F1/10 car

- Team assignment
- Due Wednesday April 3 – 2:00pm – Demo and Code
- Share the link to your team/blog

F1/10

NVIDIA

GTC GPU TECHNOLOGY CONFERENCE

autox
NVIDIA

On delivery

SELF-DRIVING TRUCK
tu
tu simple
NVIDIA
NVIDIA
kinetica ACTIVE ANALYTICS
9TB452

POWERED BY NVIDIA DRIVE

Enable ROS over network

The F1/10 has ROS over network preconfigured

Your remote computer/VM must also be configured in the same manner

To do this, open the .bashrc file using an editor

Eg: nano ~/.bashrc

Scroll down to the bottom and add: export ROS_MASTER_URI=http://192.168.1.1:11311

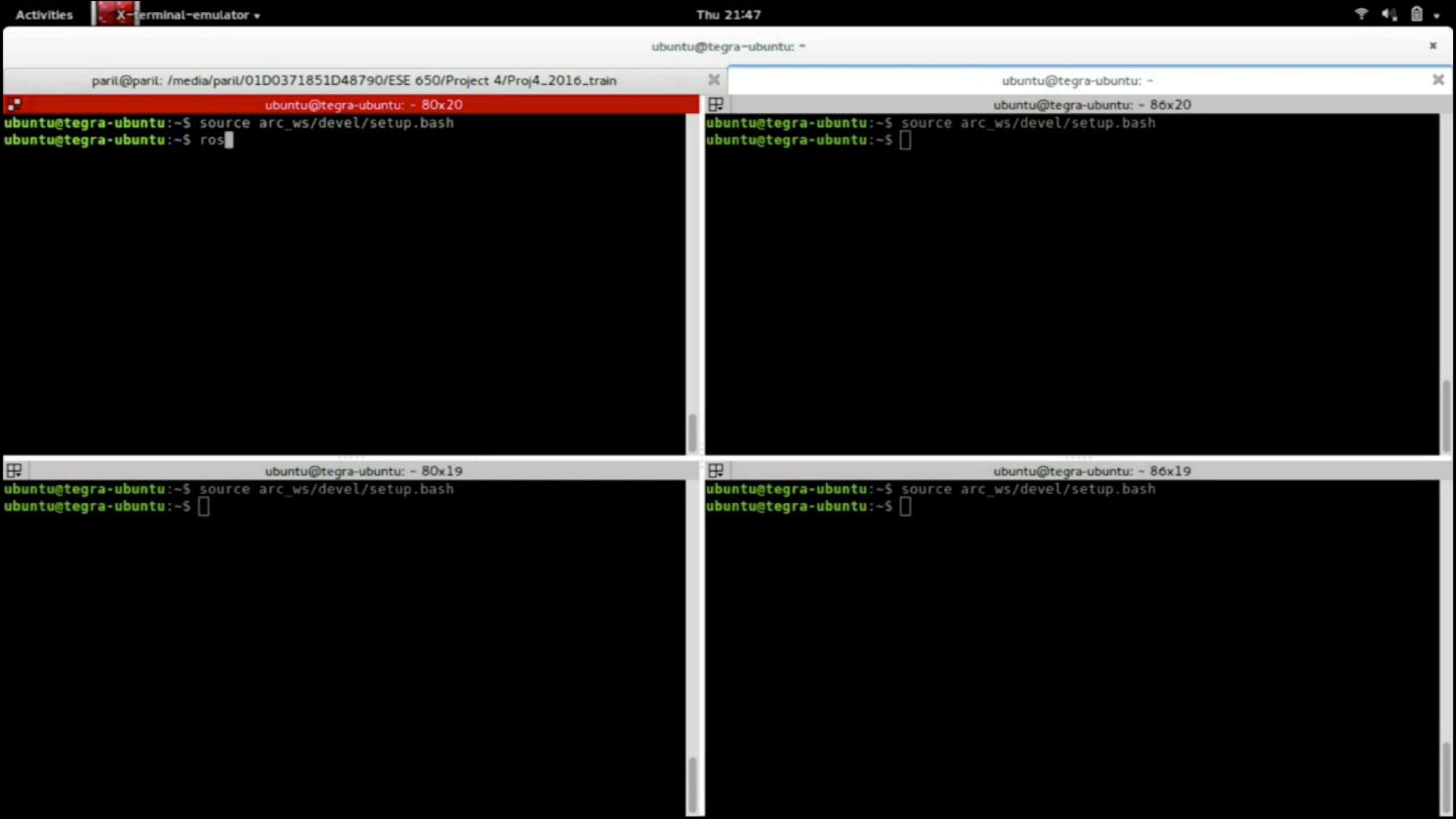
Remote login (SSH)

Do this only after you have configured the network as instructed

```
ssh ubuntu@192.168.1.1
password: ubuntu
```

Do not change passwords for TK1 or the network

Assignment 3 : Keyboard control for the car



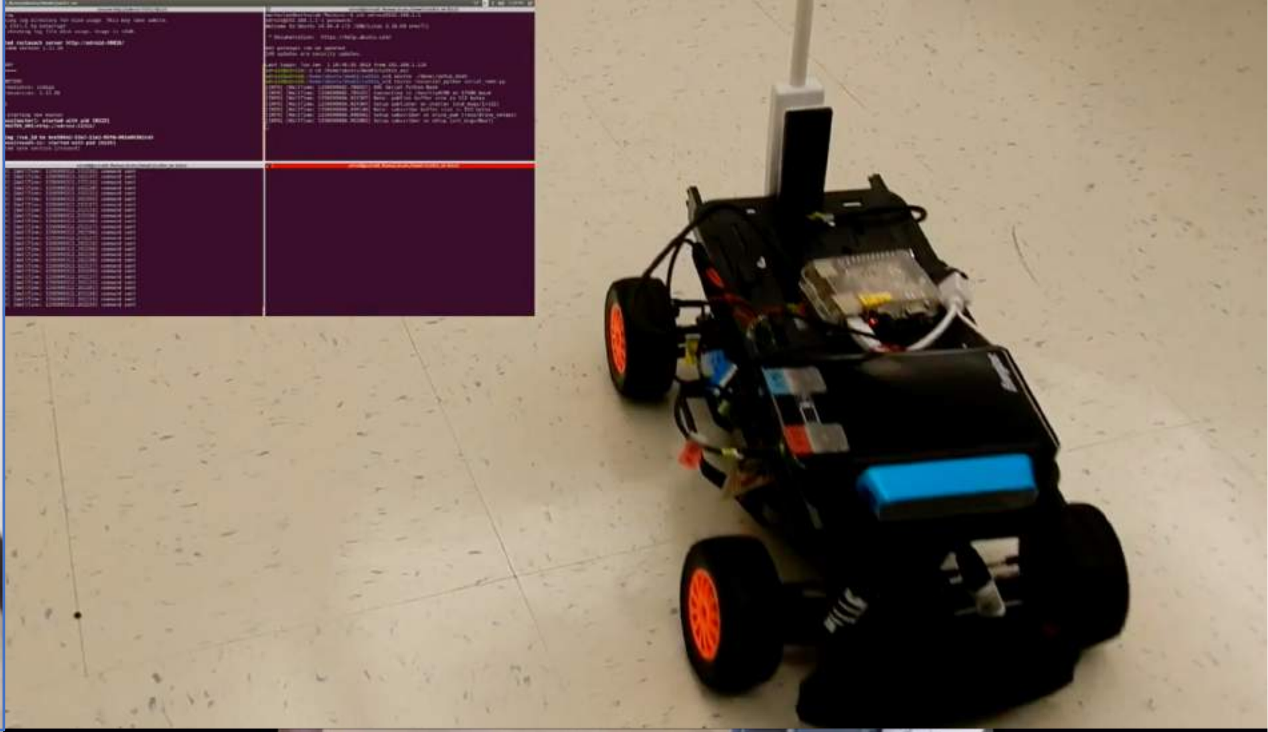
Activities
X-terminal-emulator
Thu 21:47
ubuntu@tegra-ubuntu: ~
paril@paril: /media/paril/01D0371851D48790/ESE 650/Project 4/Proj4_2016_train
ubuntu@tegra-ubuntu: ~
ubuntu@tegra-ubuntu: ~ 80x20
ubuntu@tegra-ubuntu: ~ 86x20
ubuntu@tegra-ubuntu: ~ 80x19
ubuntu@tegra-ubuntu: ~ 86x19

```
ubuntu@tegra-ubuntu:~$ source arc_ws/devel/setup.bash
ubuntu@tegra-ubuntu:~$ ros
```

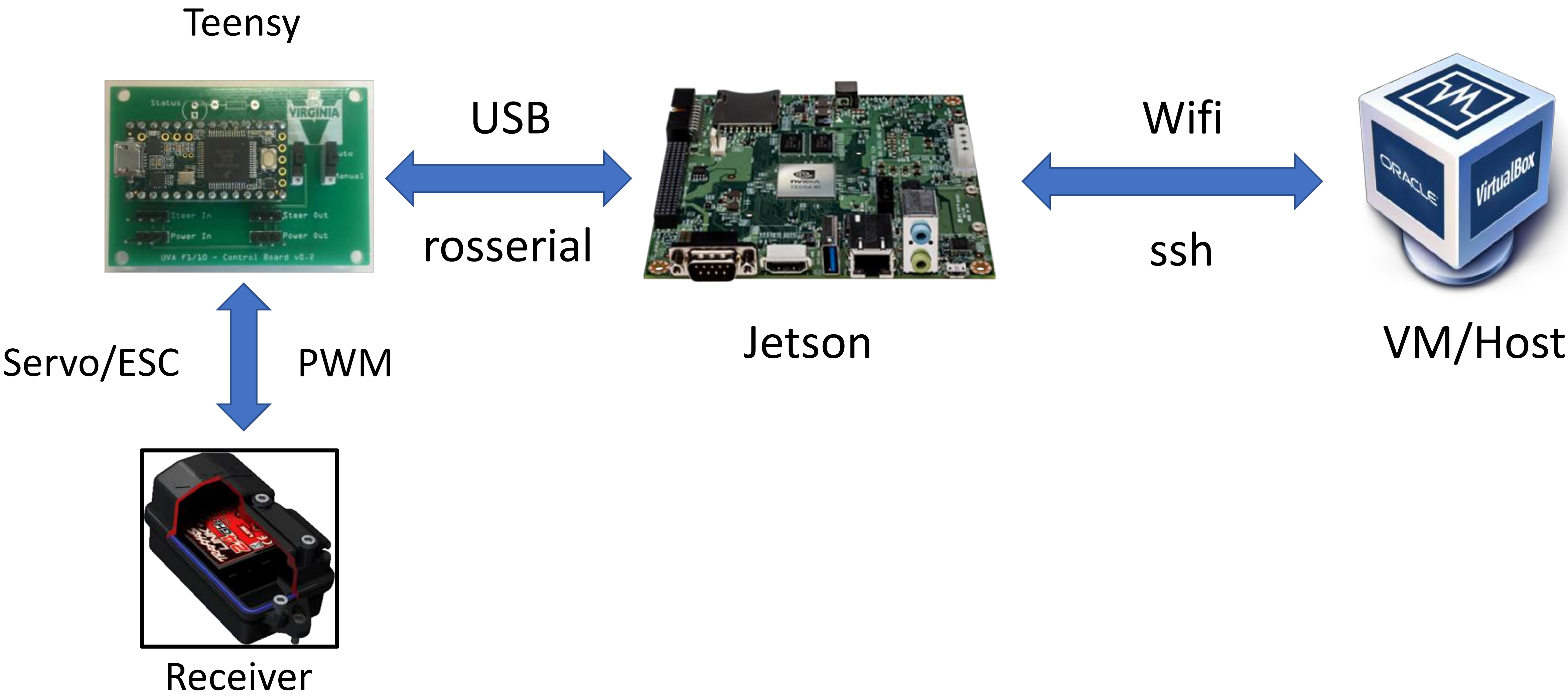
```
ubuntu@tegra-ubuntu:~$ source arc_ws/devel/setup.bash
ubuntu@tegra-ubuntu:~$
```

```
ubuntu@tegra-ubuntu:~$ source arc_ws/devel/setup.bash
ubuntu@tegra-ubuntu:~$
```

```
ubuntu@tegra-ubuntu:~$ source arc_ws/devel/setup.bash
ubuntu@tegra-ubuntu:~$
```



Connection diagram



Teensy

USB

rosserial

Wifi

ssh

Jetson

VM/Host

Servo/ESC

PWM

Receiver

Teensy Setup

- **Generates Pulse Width Modulation Signals for 2 Channels**

- **10%** duty cycle input on ESC channel – **Full throttle**
- **10%** duty cycle input on Servo Channel – **Steer max left**

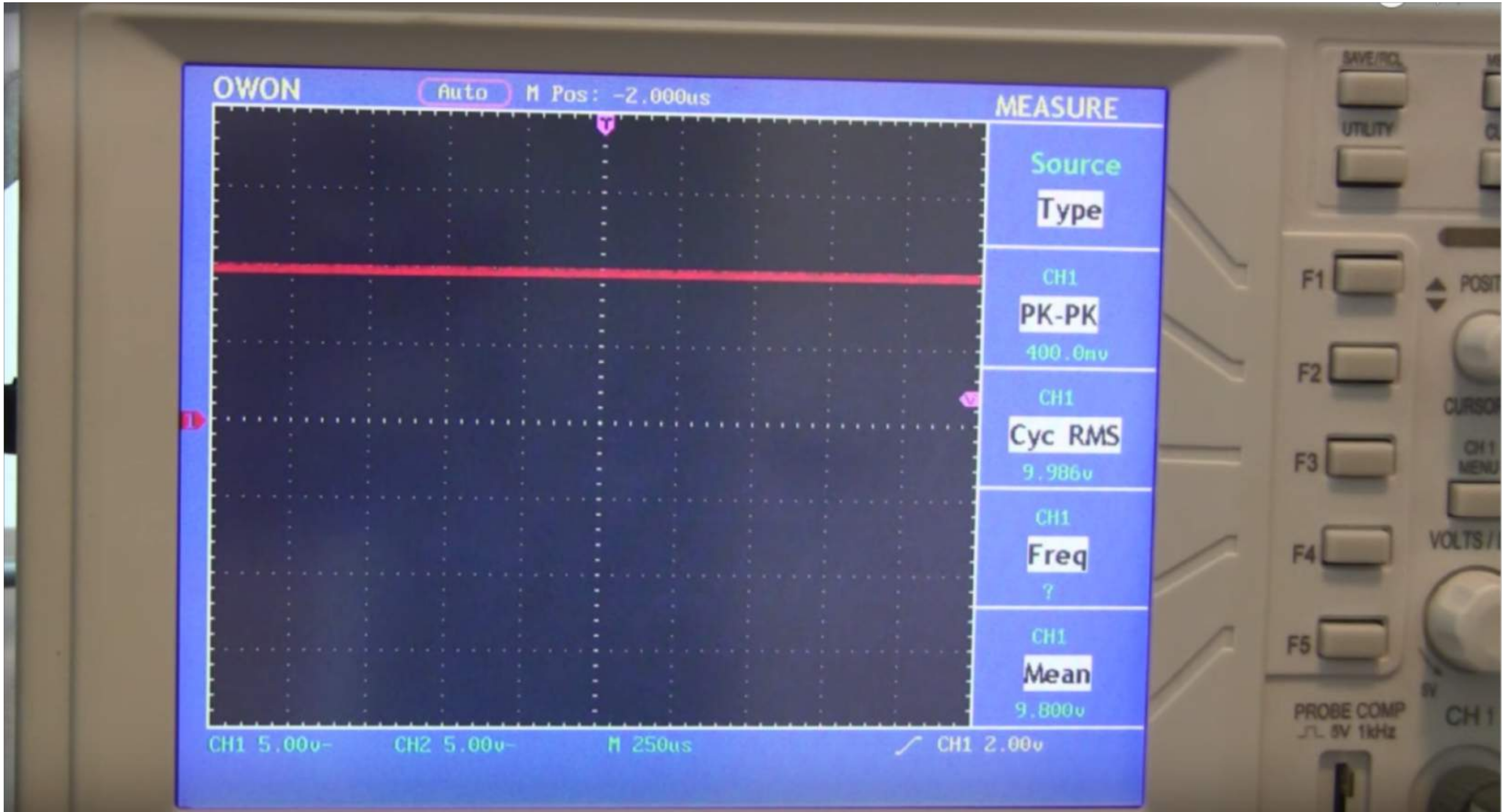
- **20%** duty cycle input on ESC channel – **Full reverse**
- **20%** duty cycle input on Servo channel – **Steer max right**

- **15 %** duty cycle input on ESC channel – **Zero throttle**
- **15 %** duty cycle input on Servo channel – **Center Steering**

What is PWM – Pulse Width Modulation

- Output signal alternates between on and off within specified period
- Controls power received by a device
- The voltage seen by the load is directly proportional to the source voltage

Here is what 10V DC looks like..



OWON Auto M Pos: -2.000us

MEASURE

Source

Type

CH1
PK-PK
400.0mv

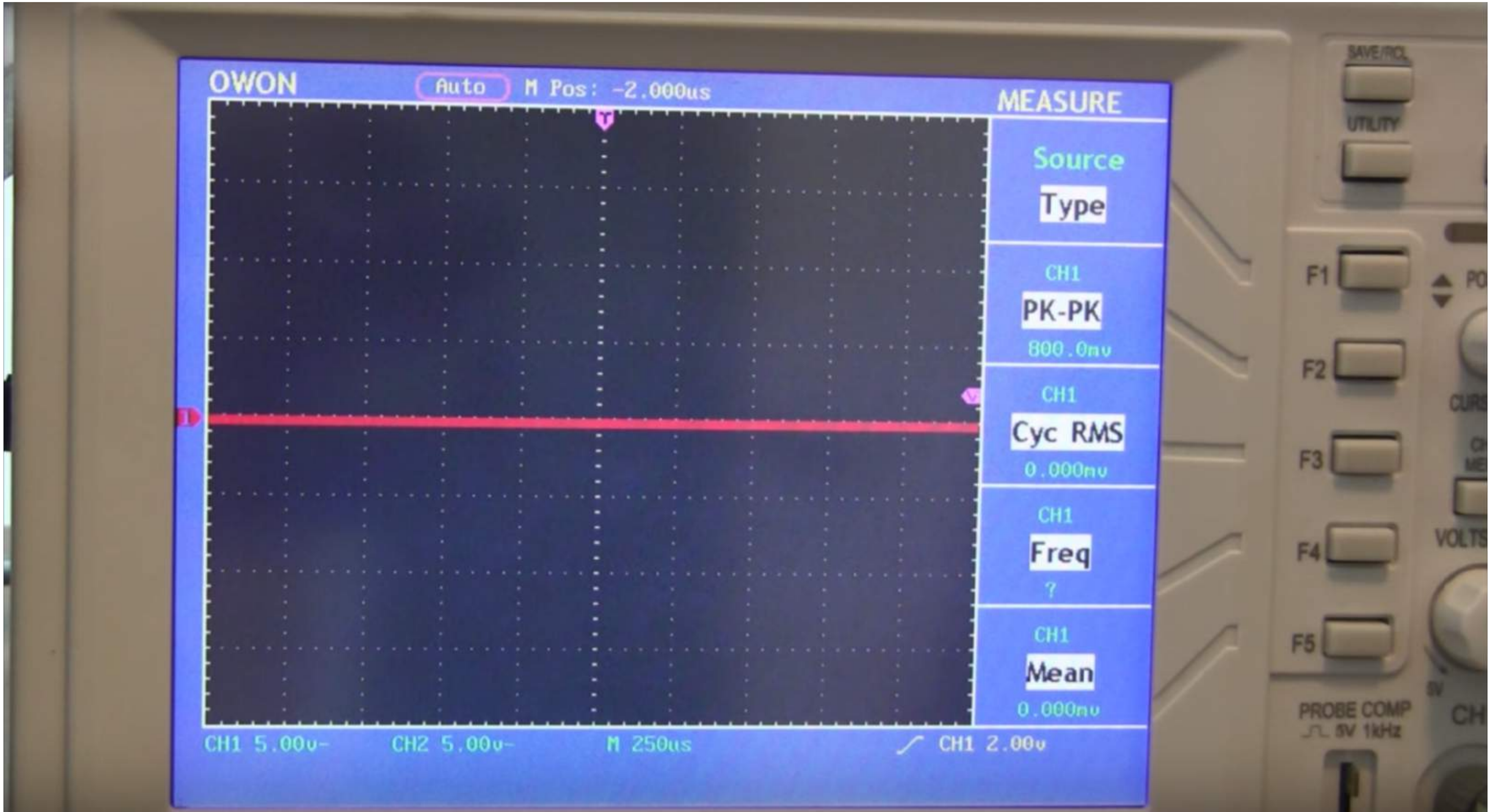
CH1
Cyc RMS
9.986v

CH1
Freq
?

CH1
Mean
9.800v

CH1 5.00v- CH2 5.00v- M 250us CH1 2.00v

Here is what 0V DC looks like..



OWON Auto M Pos: -2.000us

MEASURE

Source Type

CH1 PK-PK 800.0mv

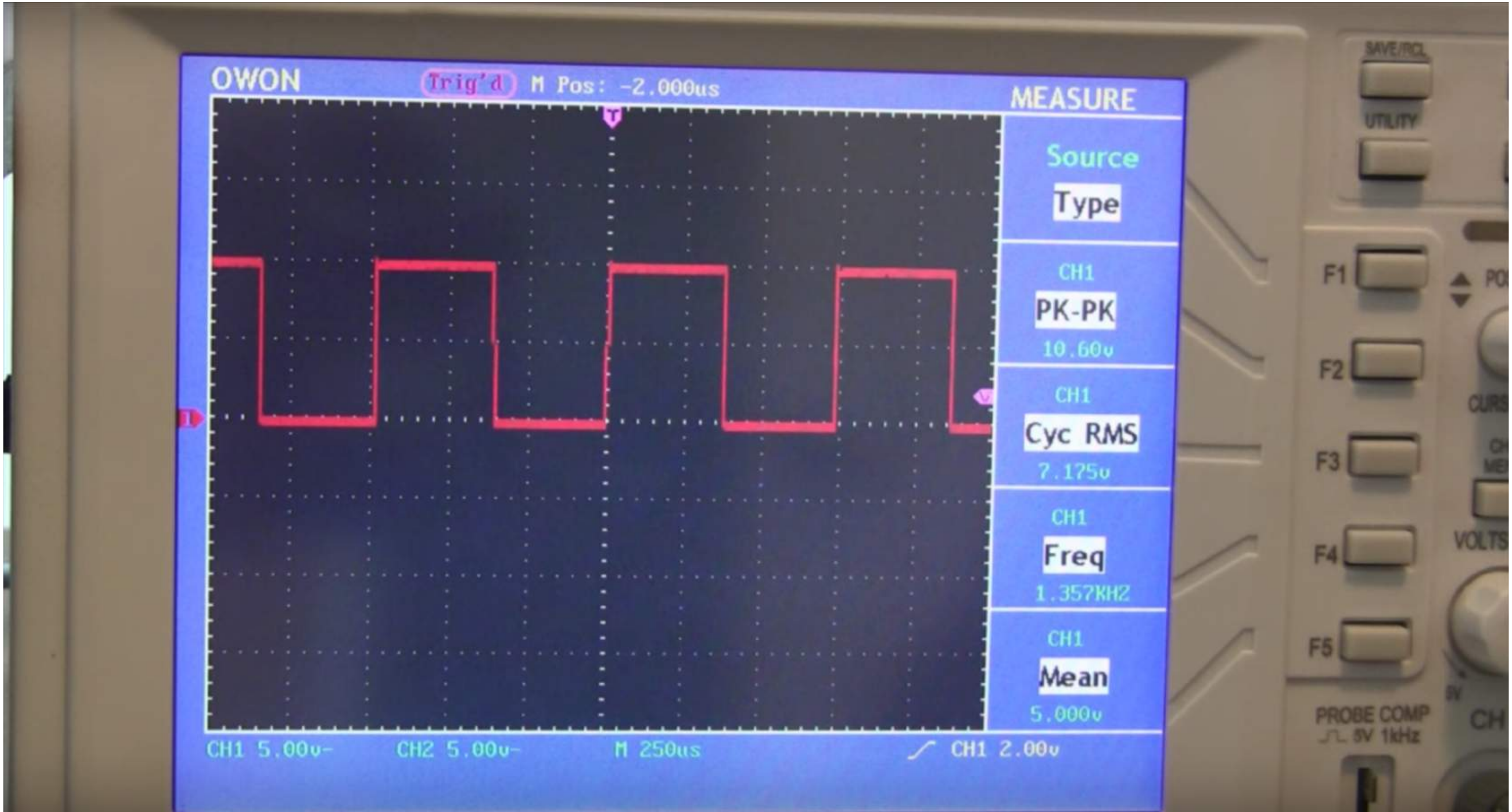
CH1 Cyc RMS 0.000mv

CH1 Freq ?

CH1 Mean 0.000mv

CH1 5.00v- CH2 5.00v- M 250us CH1 2.00v

Switch 10V ON half the time : 50% Duty Cycle



Source
Type

CH1
PK-PK
10.60v

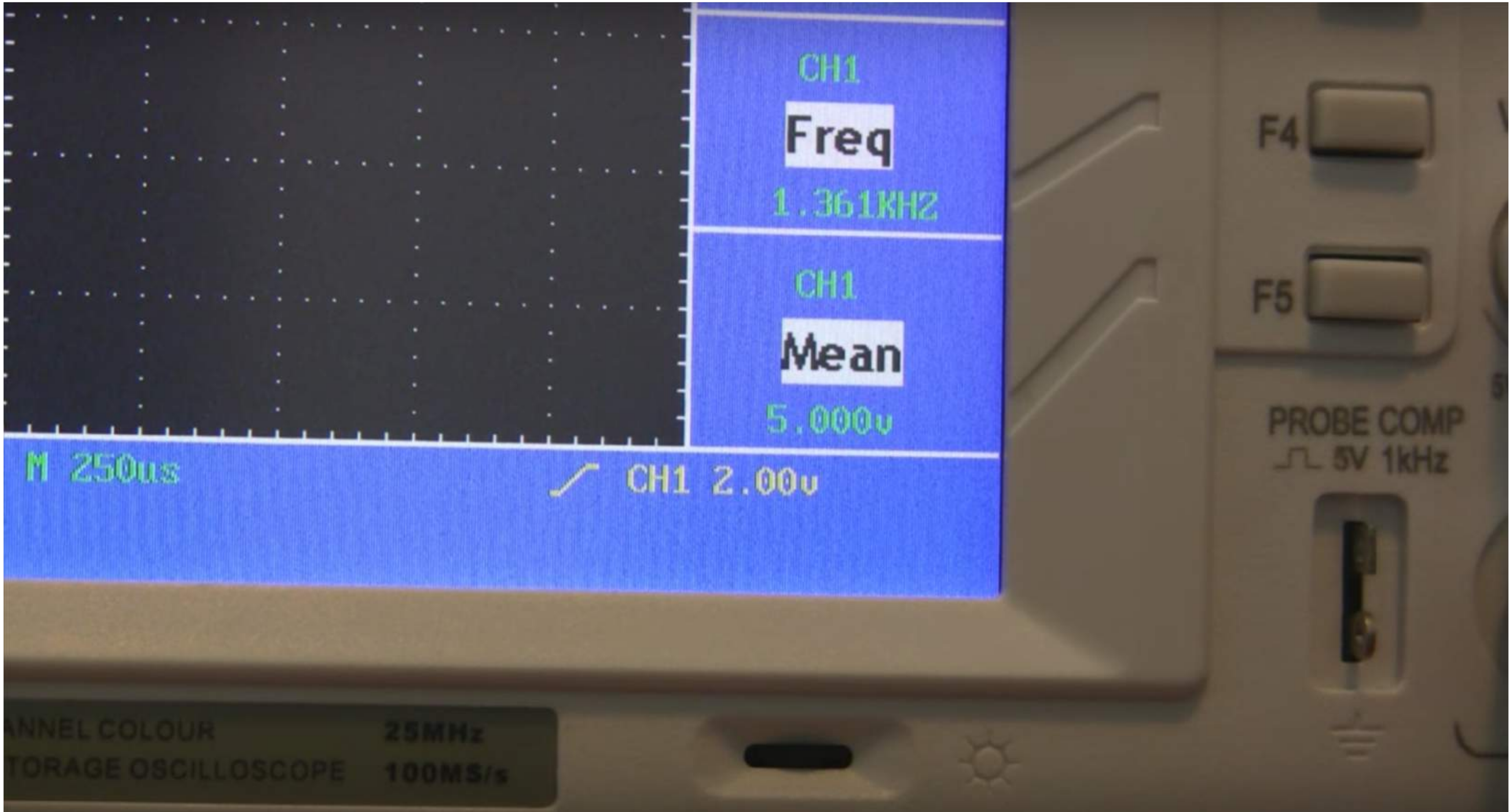
CH1
Cyc RMS
7.175v

CH1
Freq
1.357KHz

CH1
Mean
5.000v

CH1 5.00v- CH2 5.00v- M 250us CH1 2.00v

10V DC: 50% duty cycle → average of 5V DC



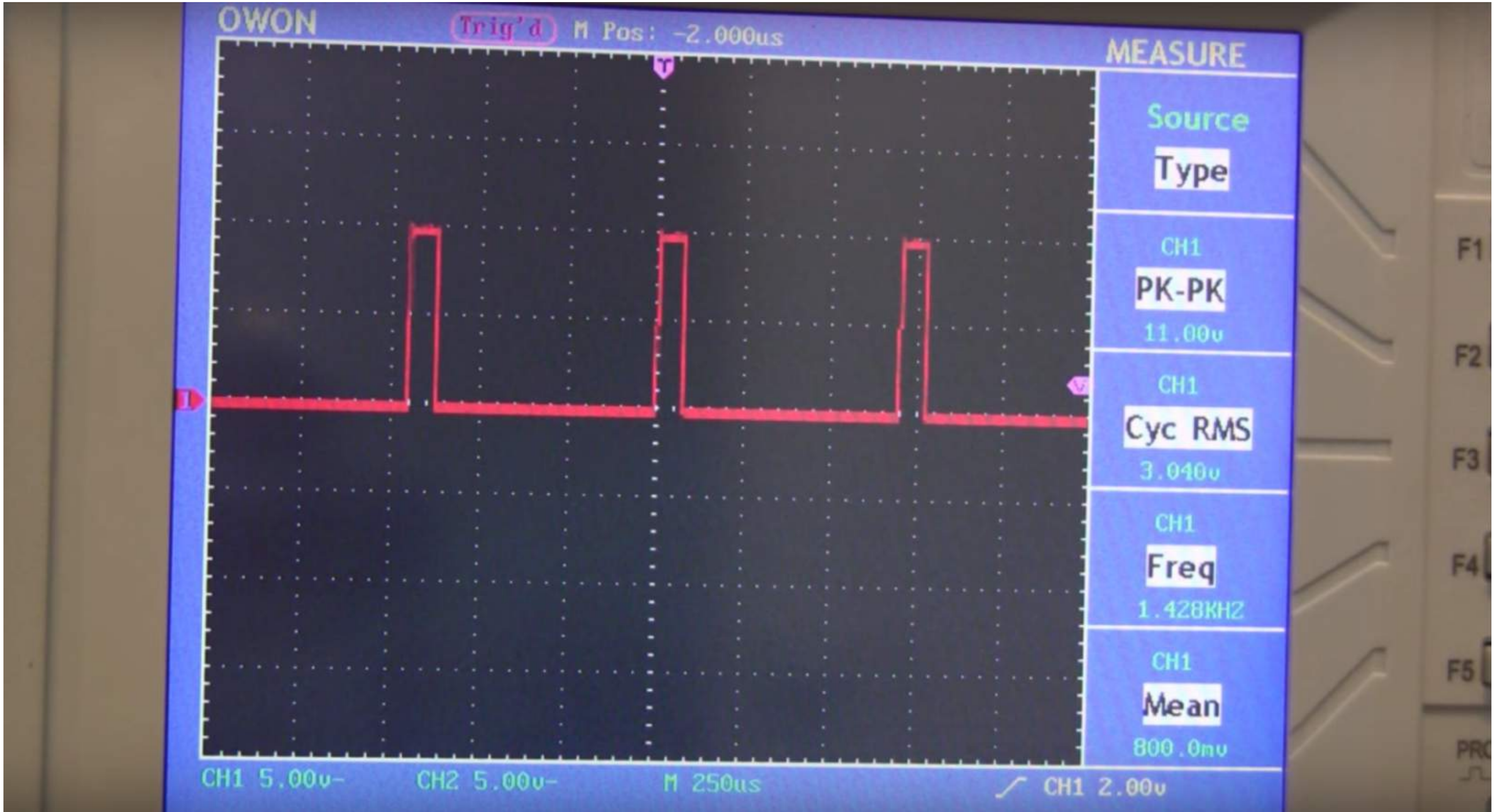
CH1
Freq
1.361KHZ

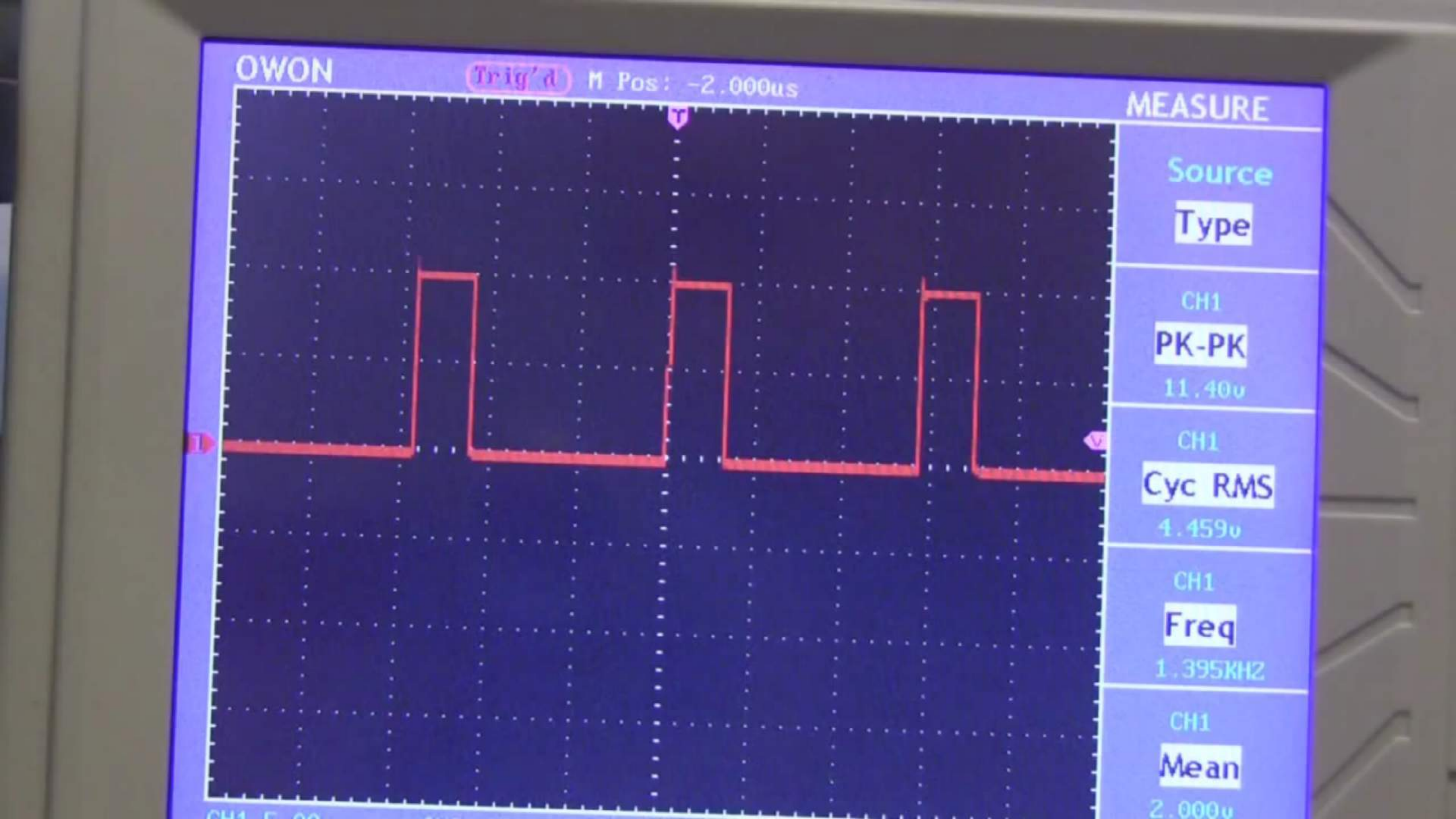
CH1
Mean
5.000v

M 250us

CH1 2.00v

Here is a case with 10% duty cycle..





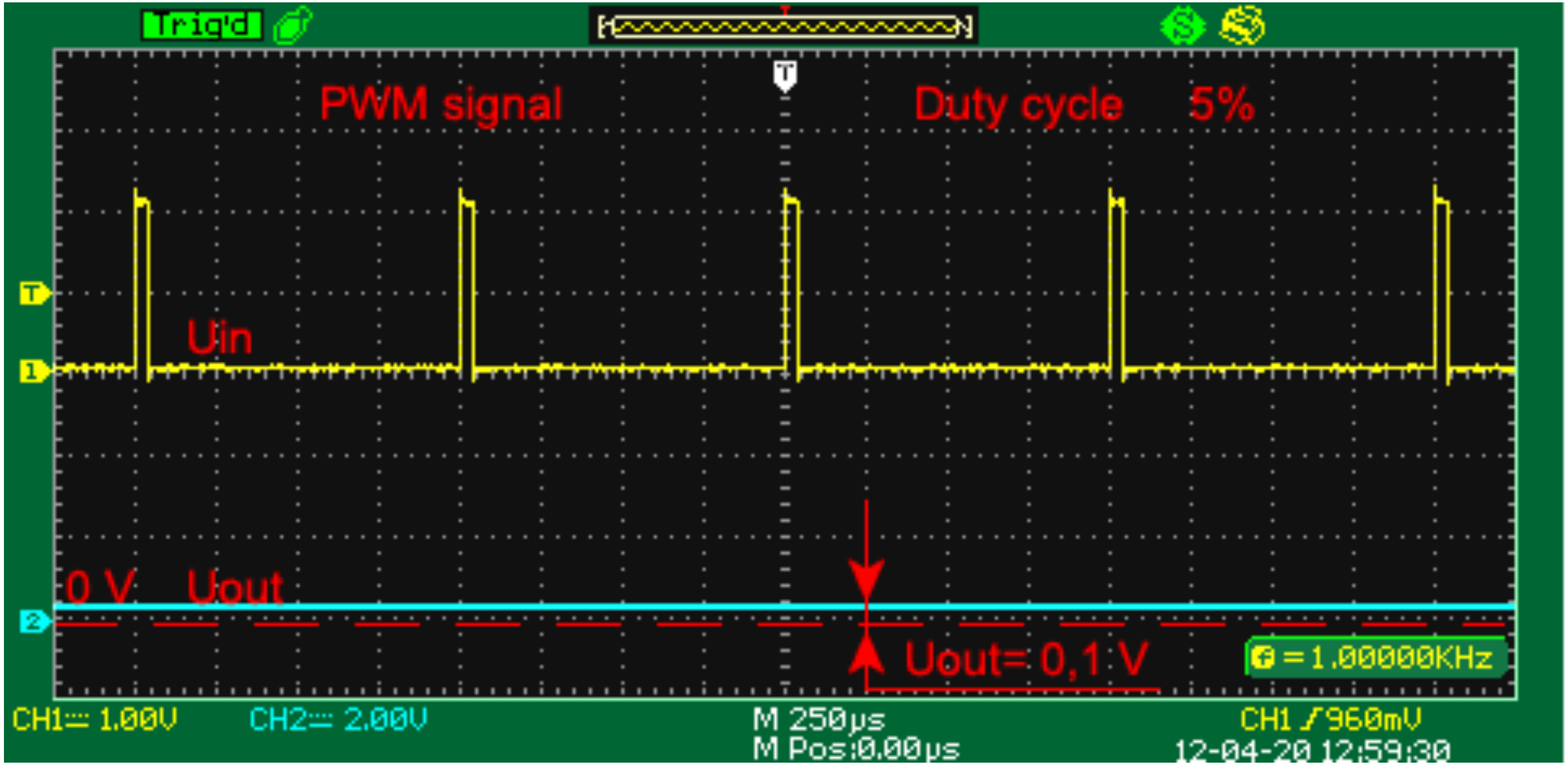
M Pos: -2.000us

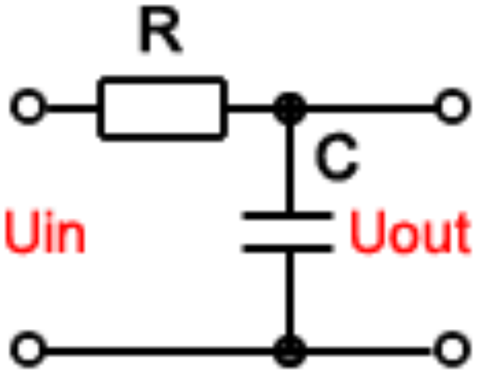
CH1 PK-PK 11.40v

CH1 Cyc RMS 4.459v

CH1 Freq 1.395KHz

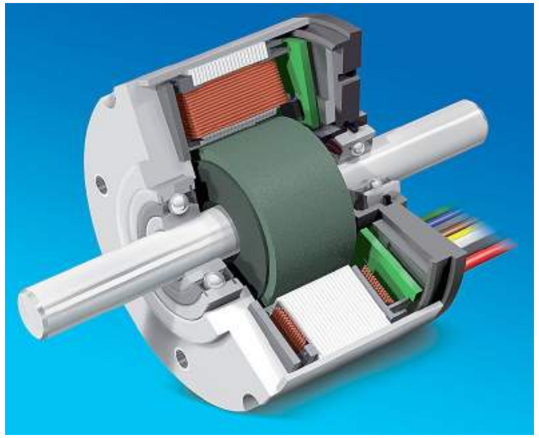
CH1 Mean 2.000v

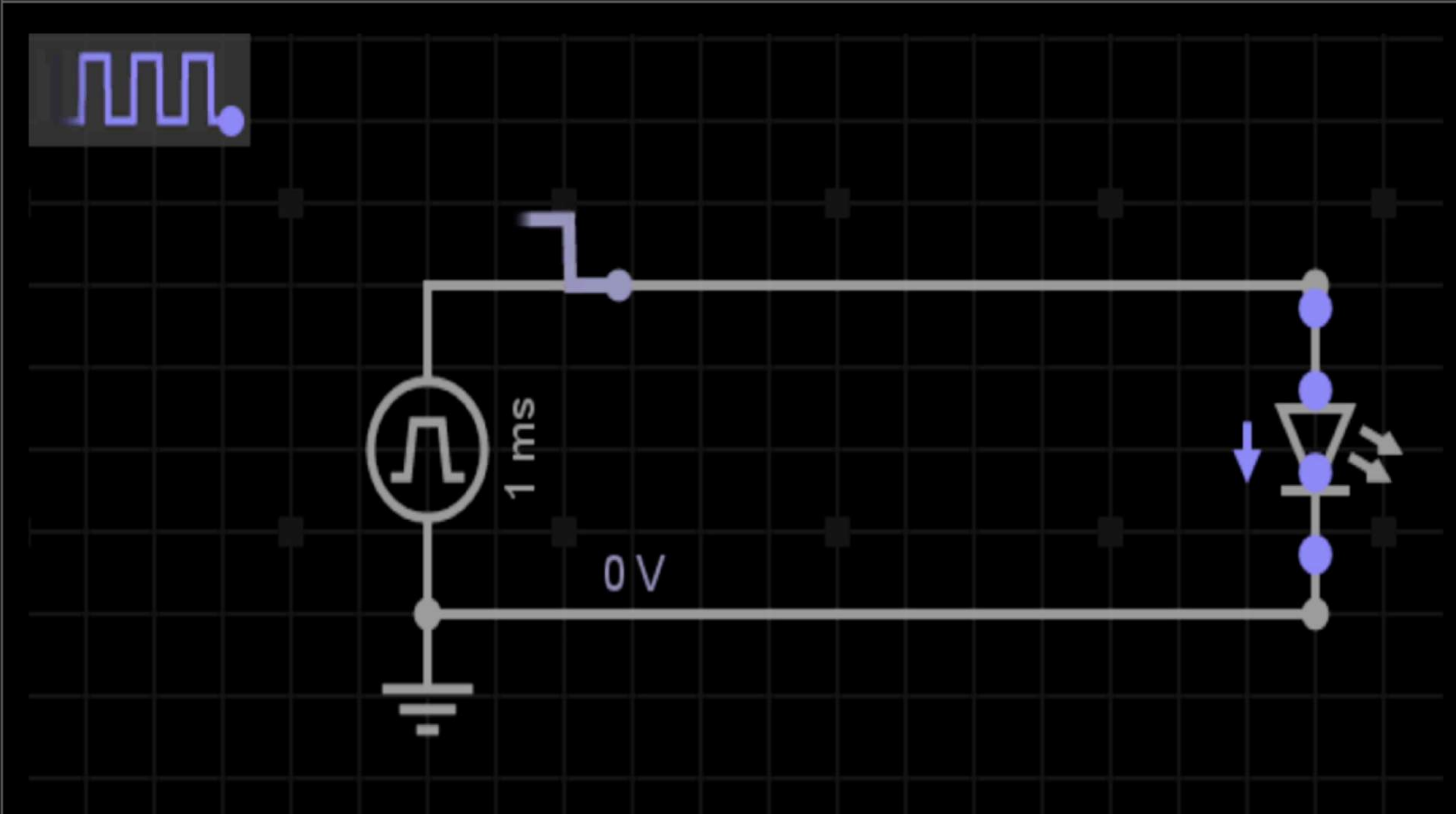

Trig'd
PWM signal
Duty cycle 5%
Uin
0 V Uout
Uout= 0,1 V
f = 1.00000KHz
CH1 1.00V
CH2 2.00V
M 250μs
M Pos:0.00μs
CH1 960mV
12-04-20 12:59:30


R
C
Uin
Uout

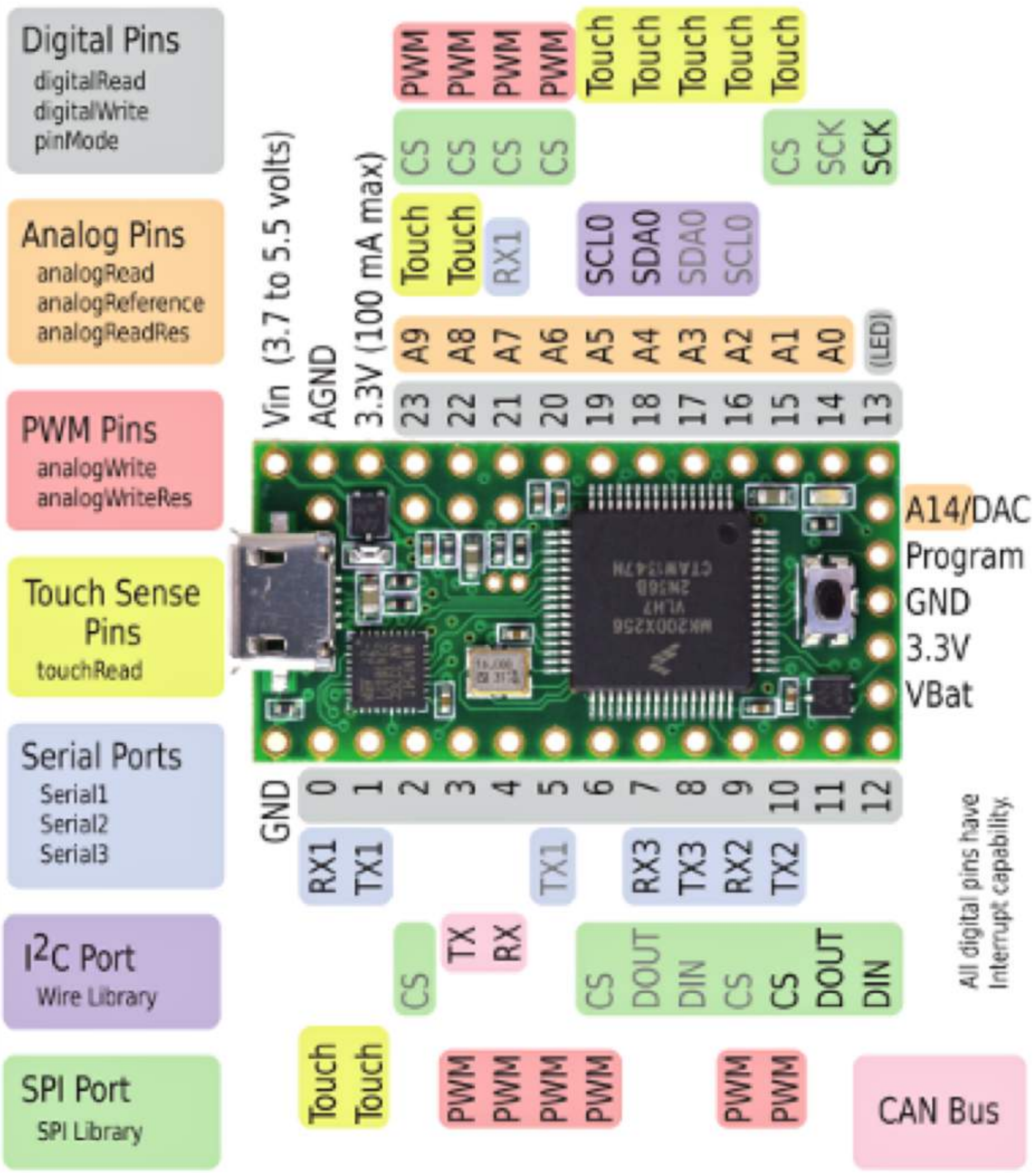
Application to DC motors

- The voltage supplied to a DC motor is proportional to the duty cycle
- Both brushed and brushless motors can be used with PWM
- Both analog and digital control techniques and components are available

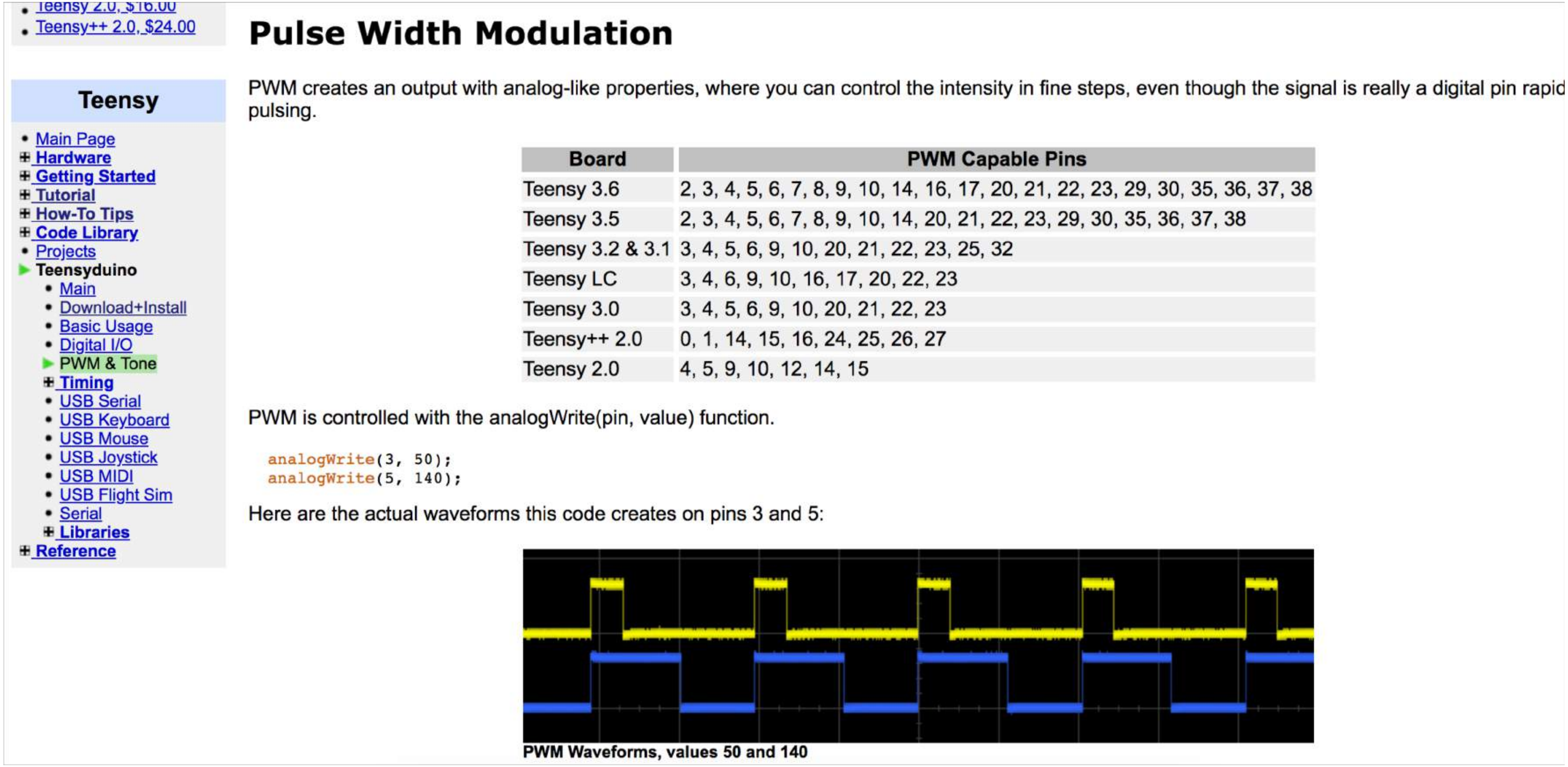



1 ms
0 V

Teensy can generate PWM signals



Teensy PWM



- Teensy 2.0, $16.00
- Teensy++ 2.0, $24.00

Teensy

- Main Page
- ⊞ Hardware
- ⊞ Getting Started
- ⊞ Tutorial
- ⊞ How-To Tips
- ⊞ Code Library
- Projects
- ▶ Teensyduino
  - Main
  - Download+Install
  - Basic Usage
  - Digital I/O
  - ▶ PWM & Tone
  - ⊞ Timing
  - USB Serial
  - USB Keyboard
  - USB Mouse
  - USB Joystick
  - USB MIDI
  - USB Flight Sim
  - Serial
  - ⊞ Libraries
- ⊞ Reference

Pulse Width Modulation

PWM creates an output with analog-like properties, where you can control the intensity in fine steps, even though the signal is really a digital pin rapid pulsing.

Board	PWM Capable Pins
Teensy 3.6	2, 3, 4, 5, 6, 7, 8, 9, 10, 14, 16, 17, 20, 21, 22, 23, 29, 30, 35, 36, 37, 38
Teensy 3.5	2, 3, 4, 5, 6, 7, 8, 9, 10, 14, 20, 21, 22, 23, 29, 30, 35, 36, 37, 38
Teensy 3.2 & 3.1	3, 4, 5, 6, 9, 10, 20, 21, 22, 23, 25, 32
Teensy LC	3, 4, 6, 9, 10, 16, 17, 20, 22, 23
Teensy 3.0	3, 4, 5, 6, 9, 10, 20, 21, 22, 23
Teensy++ 2.0	0, 1, 14, 15, 16, 24, 25, 26, 27
Teensy 2.0	4, 5, 9, 10, 12, 14, 15

PWM is controlled with the analogWrite(pin, value) function.

```
analogWrite(3, 50);
analogWrite(5, 140);
```

Here are the actual waveforms this code creates on pins 3 and 5:

PWM Waveforms, values 50 and 140

Teensy Setup

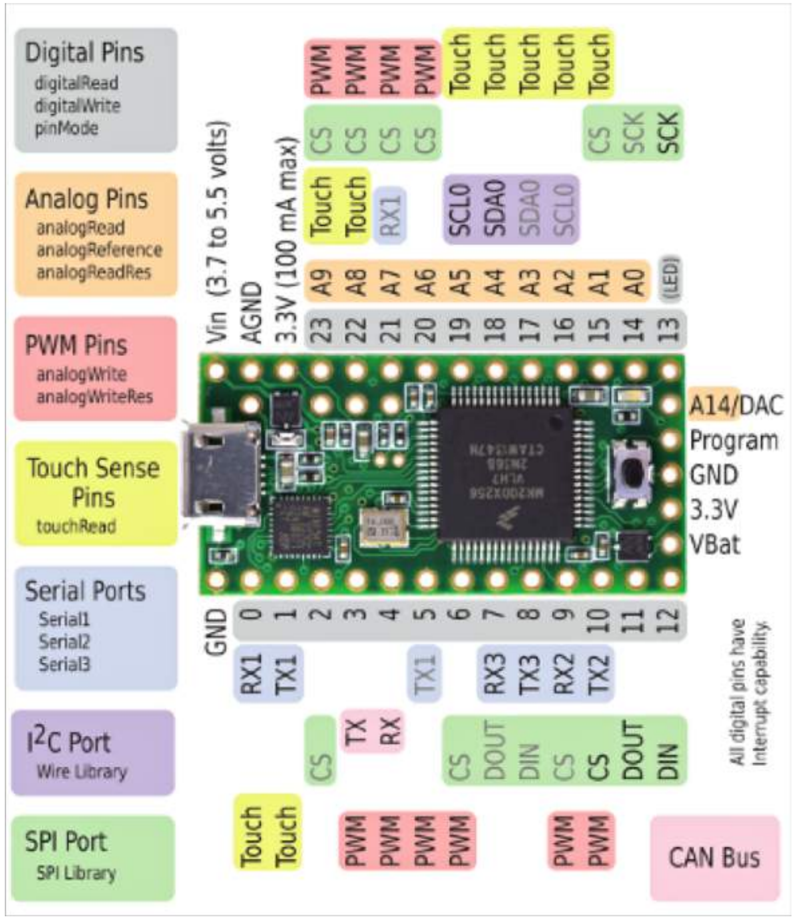
- **Generates Pulse Width Modulation Signals for 2 Channels**

- **10%** duty cycle input on ESC channel – **Full throttle**
- **10%** duty cycle input on Servo Channel – **Steer max left**

- **20%** duty cycle input on ESC channel – **Full reverse**
- **20%** duty cycle input on Servo channel – **Steer max right**

- **15 %** duty cycle input on ESC channel – **Zero throttle**
- **15 %** duty cycle input on Servo channel – **Center Steering**

What you need to know for Assignment 3



10% duty cycle → 6554 PWM value

20% duty cycle → 13108 PWM value

15% duty cycle → 9381 PWM value

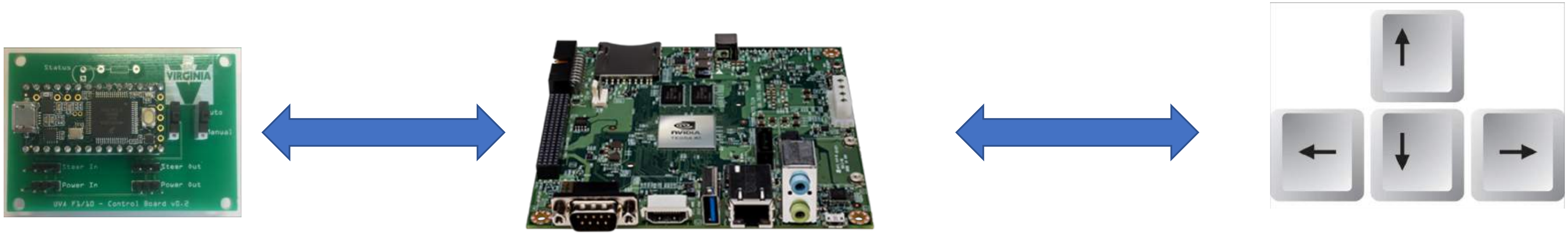
Jetson Setup

- The Jetson runs two ROS nodes:
  - **keyboard.py**
    - Obtain keyboard input from user (arrow keys)
  - **talker.py**
    - Convert user input into correct PWM values that will be sent to the Teensy

keyboard.py

Publishes topic – **drive_parameters**

Custom message type: **drive_param**



Subscribes to– **drive_pwm**

Remote connection
SSH

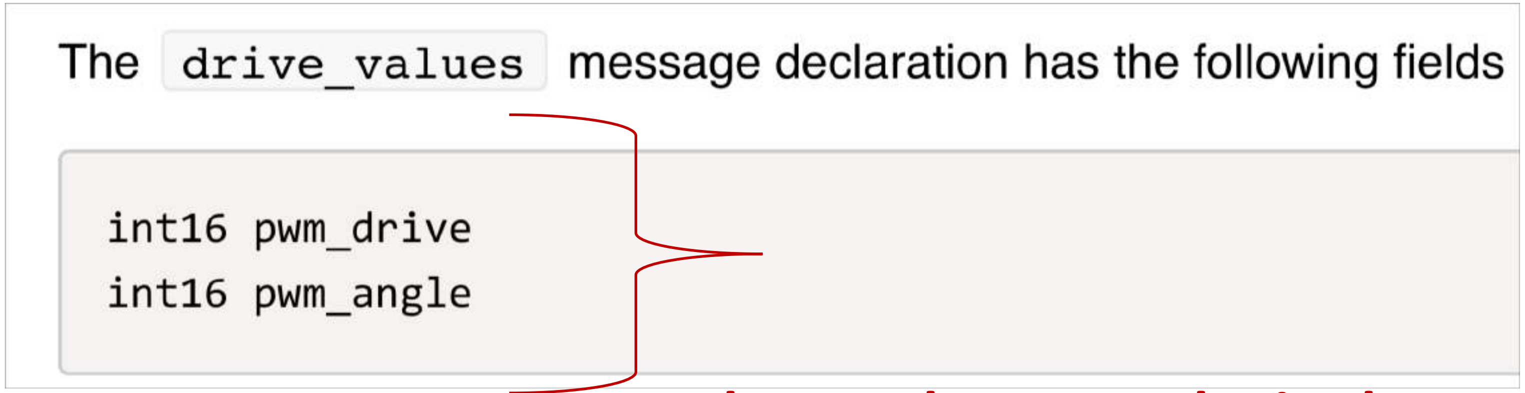
talker.py

Subscribes to– **drive_parameters**

Publishes topic – **drive_pwm**

Teensy runs a ROS node

Subscribes to a topic : **drive_pwm** with message type **drive_values**



The `drive_values` message declaration has the following fields

```
int16 pwm_drive
int16 pwm_angle
```

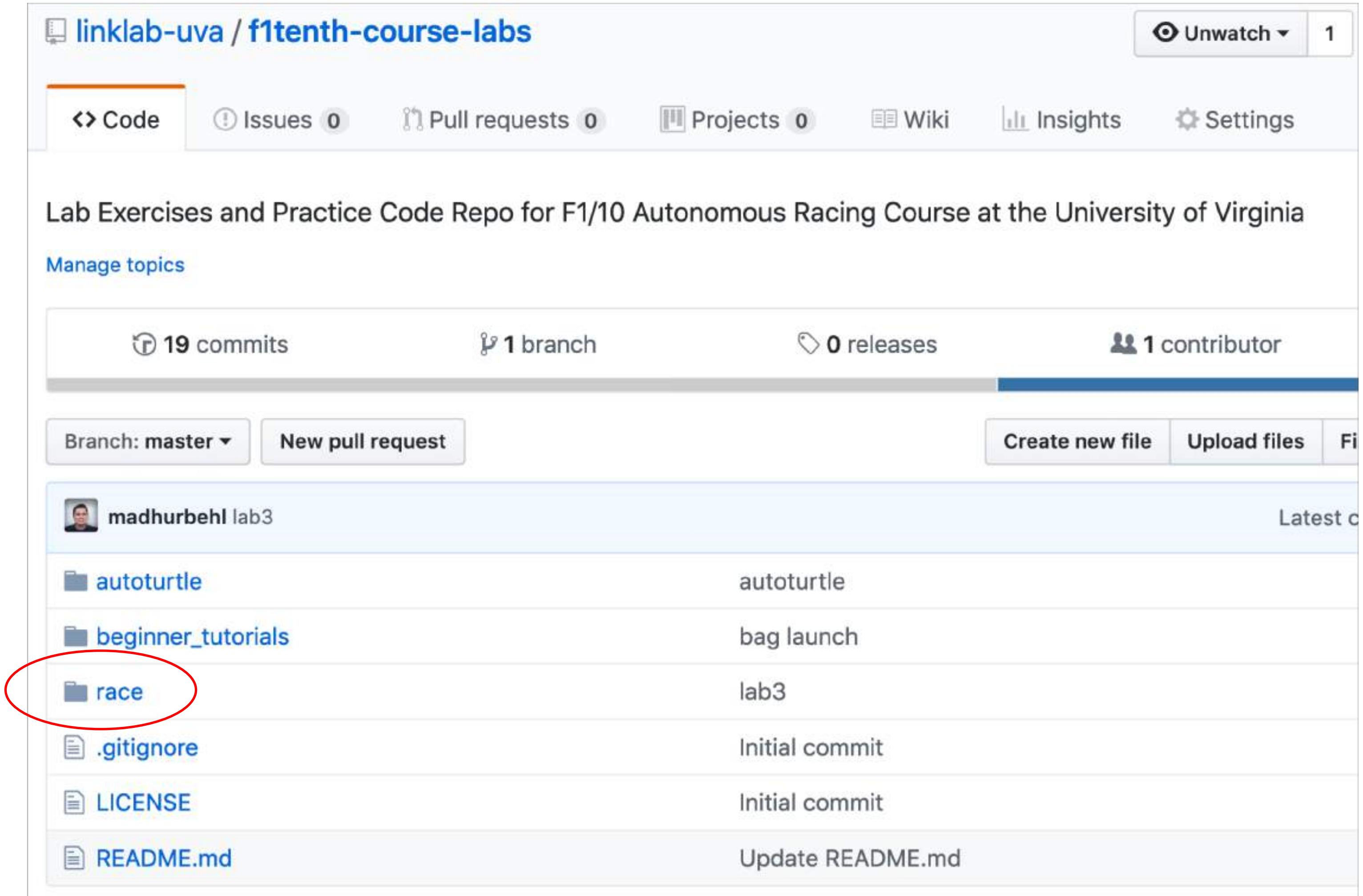
These values must be in the range (6554, 13108) corresponding to a duty cycle between (10%,20%)

Receiver

Jetson

Update Git repo

https://github.com/linklab-uva/f1tenth-course-labs



Lab Exercises and Practice Code Repo for F1/10 Autonomous Racing Course at the University of Virginia

madhurbehl lab3

autoturtle	autoturtle
beginner_tutorials	bag launch
race	lab3
.gitignore	Initial commit
LICENSE	Initial commit
README.md	Update README.md

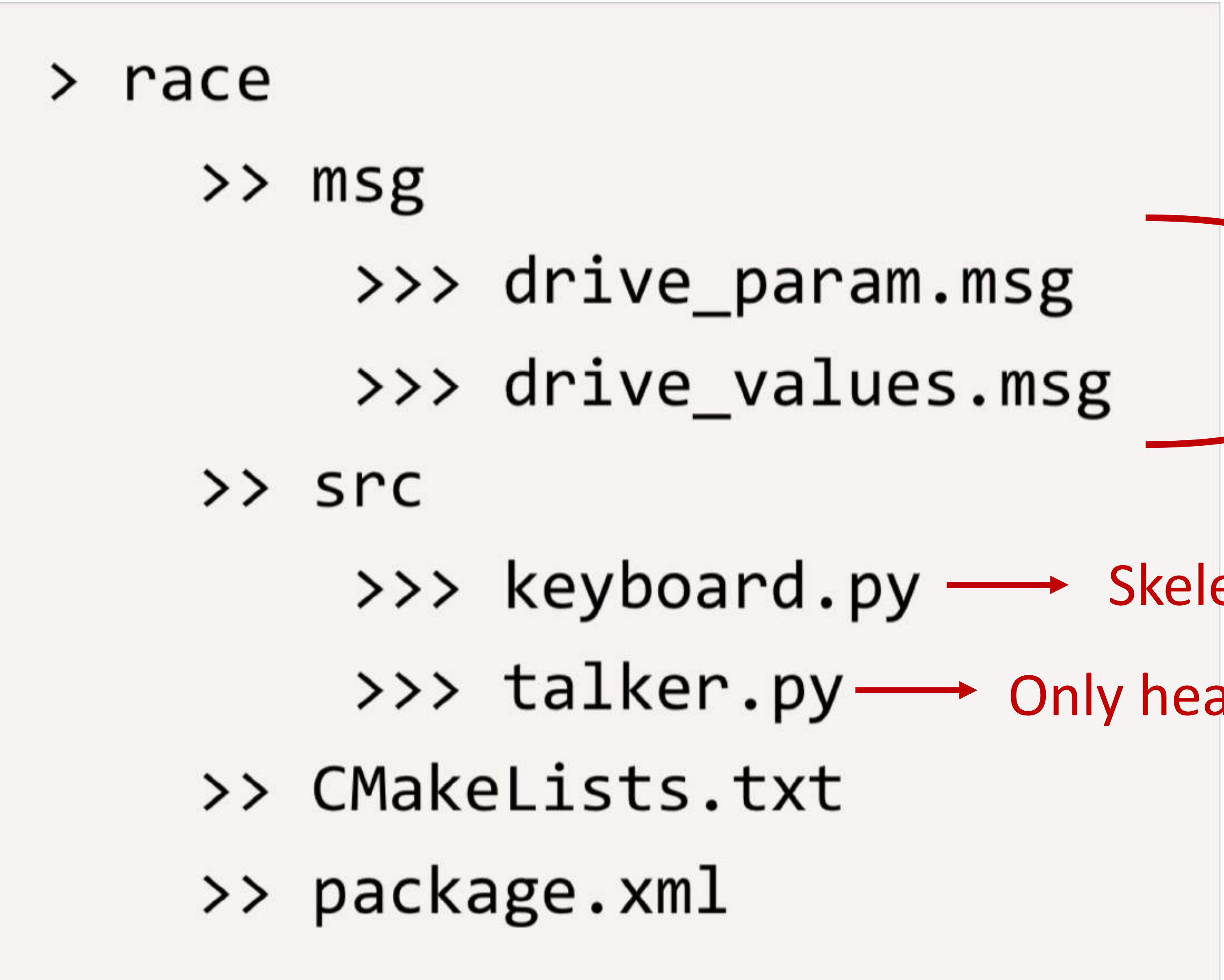
Move the race package to the Jetson

```
scp -r  ~/github/f1tenth_labs ubuntu@192.168.X.1:home/ubuntu/catkin_ws/src
```

You github path to the folder where the race package resides…

scp copies files between hosts on a network.
-r Recursively copy entire directories.

New package: ‘race’



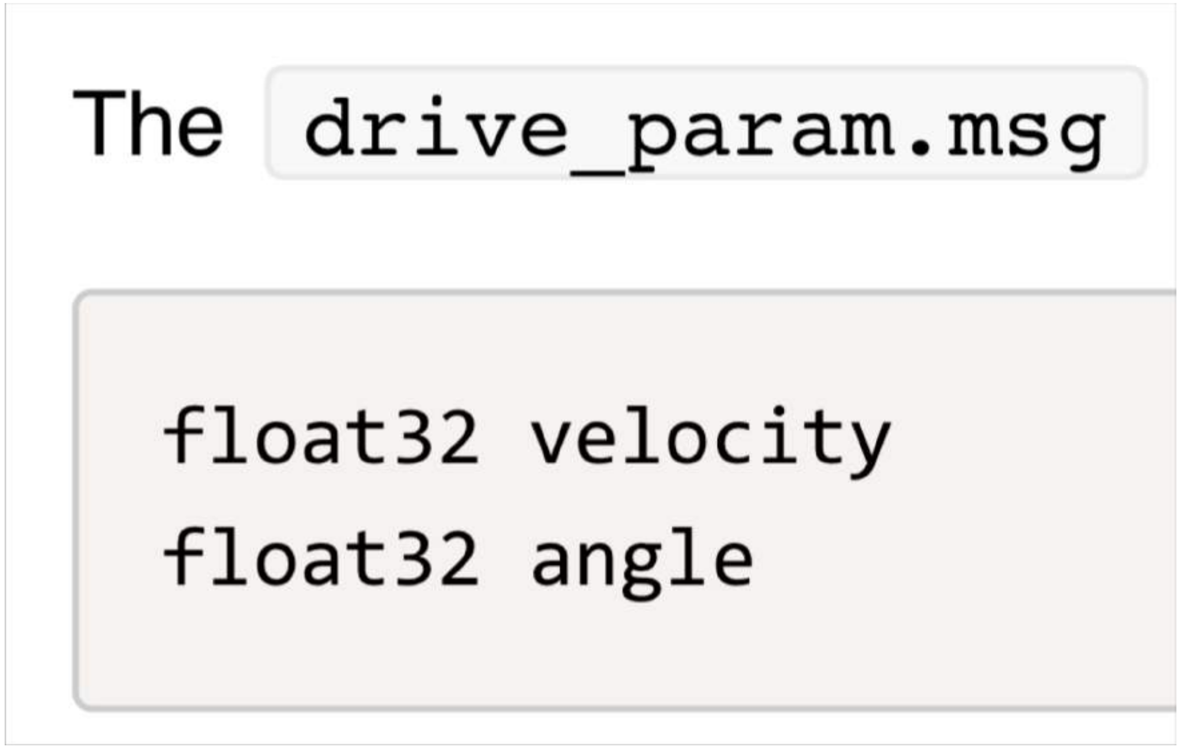
```
> race
    >> msg
        >>> drive_param.msg
        >>> drive_values.msg
    >> src
        >>> keyboard.py
        >>> talker.py
    >> CMakeLists.txt
    >> package.xml
```

Custom messages provided

keyboard.py → Skeleton code provided – You complete

talker.py → Only headers provided – You write the entire ROS node

Custom messages



The `drive_param.msg`

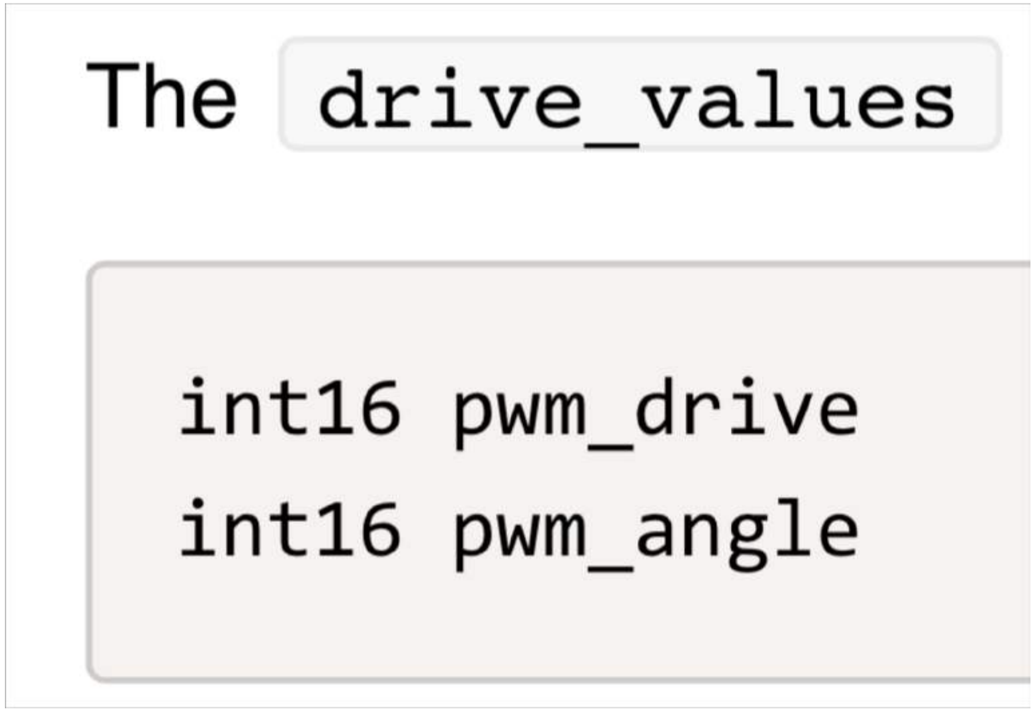
```
float32 velocity
float32 angle
```

Used by topic **drive_parameters**

- Published by keyboard.py
- Subscriber – talker.py

Both velocity and angle are a floating point number between **(-100,100)**

Custom messages



The `drive_values`

```
int16 pwm_drive
int16 pwm_angle
```

Used by topic **drive_pwm**

- Published by talker.py
- Subscriber – Teensy Node

Both pwm_drive and pwm_angle are integers number between **(6554,13108)**

Mappings: (part 1)

1. User presses an arrow key (Up, Down, Left, or Right)
2. The press is recognized by **keyboard.py**
3. The duration of the key press or the number of taps is mapped to a floating point value in the range **(-100,100)** . Use increments of 0.1
4. This is done for both velocity and angle.
5. A custom message is created of the type **drive_param** is created with the two values : velocity, and angle.
6. This message is published on the topic **drive_parameters** by the keyboard.py node.

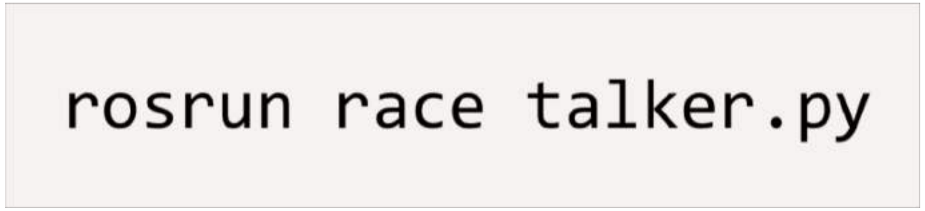
Mappings: (part 2)

1. The **talker.py** subscribes to the topic **drive_parameters**
2. It parses the velocity and angle floating values received , in a callback function.
3. In the callback function, it maps the received value to the correct PWM value in the integer range **(6554, 13108)**
4. Therefore, you are mapping some **floating** number in the range **(-100,100)** to an **integer** in the range **(6554, 13108)**
5. The two calculated values (one corresponding to velocity, and other to angle) are assigned to the fields **pwm_drive**, and **pwm_angle** of the custom message **drive_values**
6. This message is published on the topic **drive_pwm**

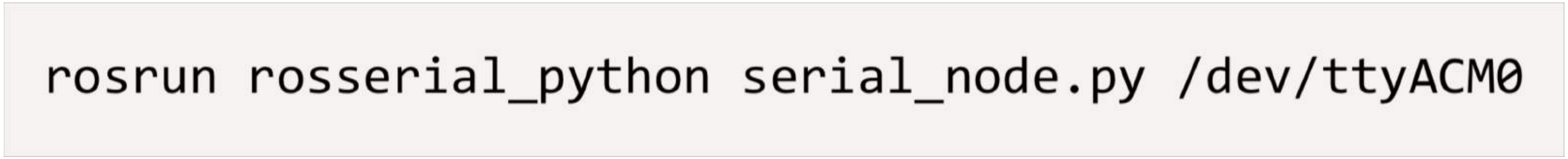
Mappings: (part 3)

1. The Teensy node is already flashed with the ros node code to listen for messages on the **drive_pwm** topic being published by **talker.py**
2. Ensure that the messages being sent to the Teensy, are always within the range (6554,13108) which corresponds to the 10% and 20% duty cycle values.
3. The 10%-20% duty cycle values are enough to fully throttle and steer the car in either direction.

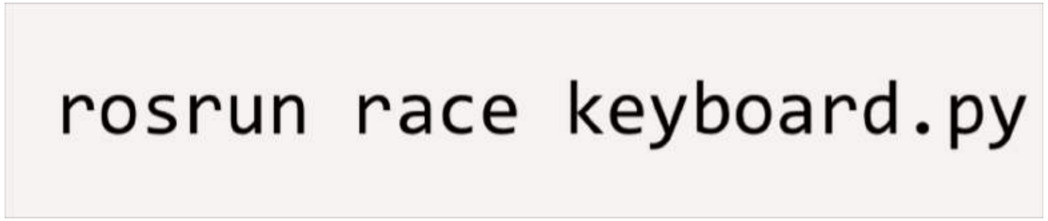
During the demo



```
rosrun race talker.py
```



```
rosrun rosserial_python serial_node.py /dev/ttyACM0
```



```
rosrun race keyboard.py
```

Remember !

- **Green** Energizer cable → **Jetson**
- **Blue** Energizer cabe → **Wifi [PoE connector]**
- Use the labels provided…