EECS 192: Mechatronics Design Lab

Discussion 2: Hardware, Equipment, and More!

GSI: Andrew Barkan

January 29, 2020 (Week 2)

- Hardware
- Connecting and Control
- Version Control

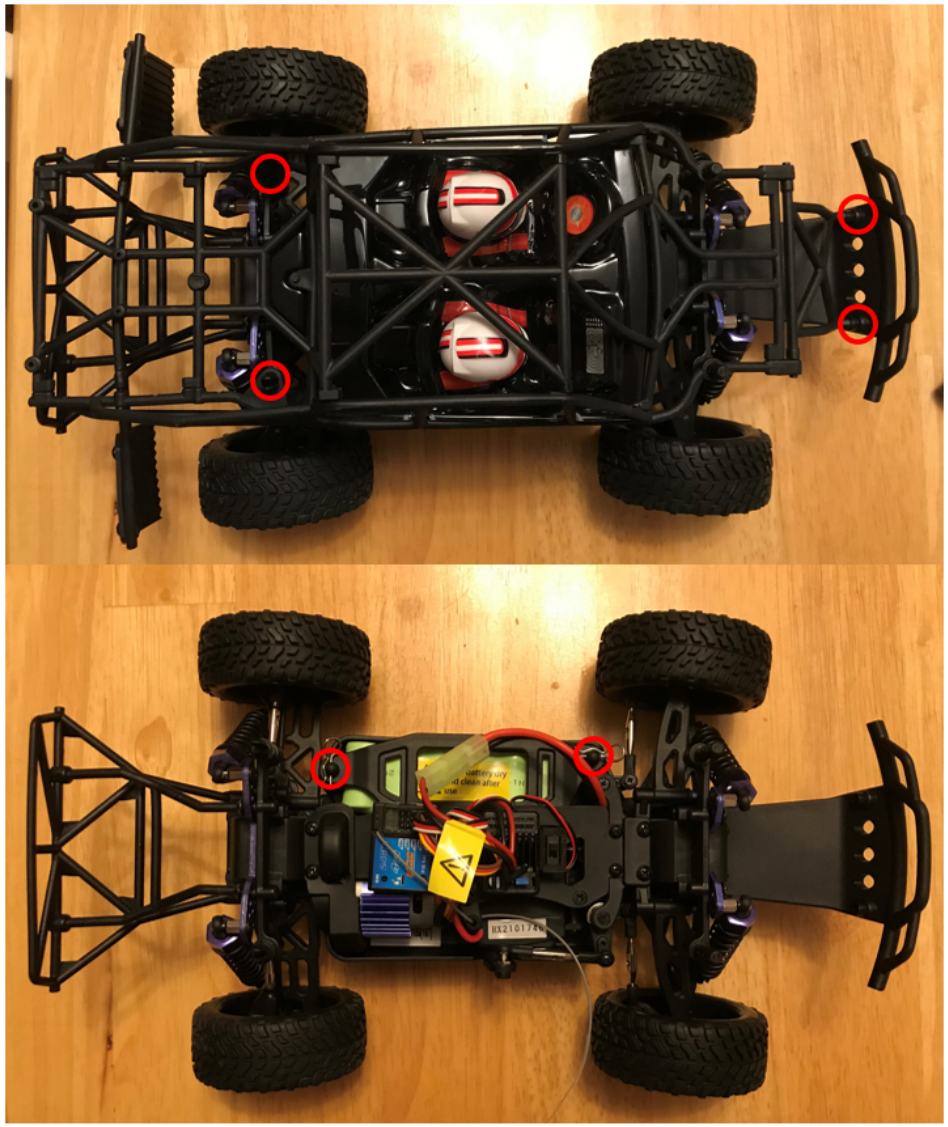
Before starting on any hardware, **TEST YOUR CAR!**
Follow the included instructions to see if everything is functional
Much more difficult to get a replacement once you've torn it apart!

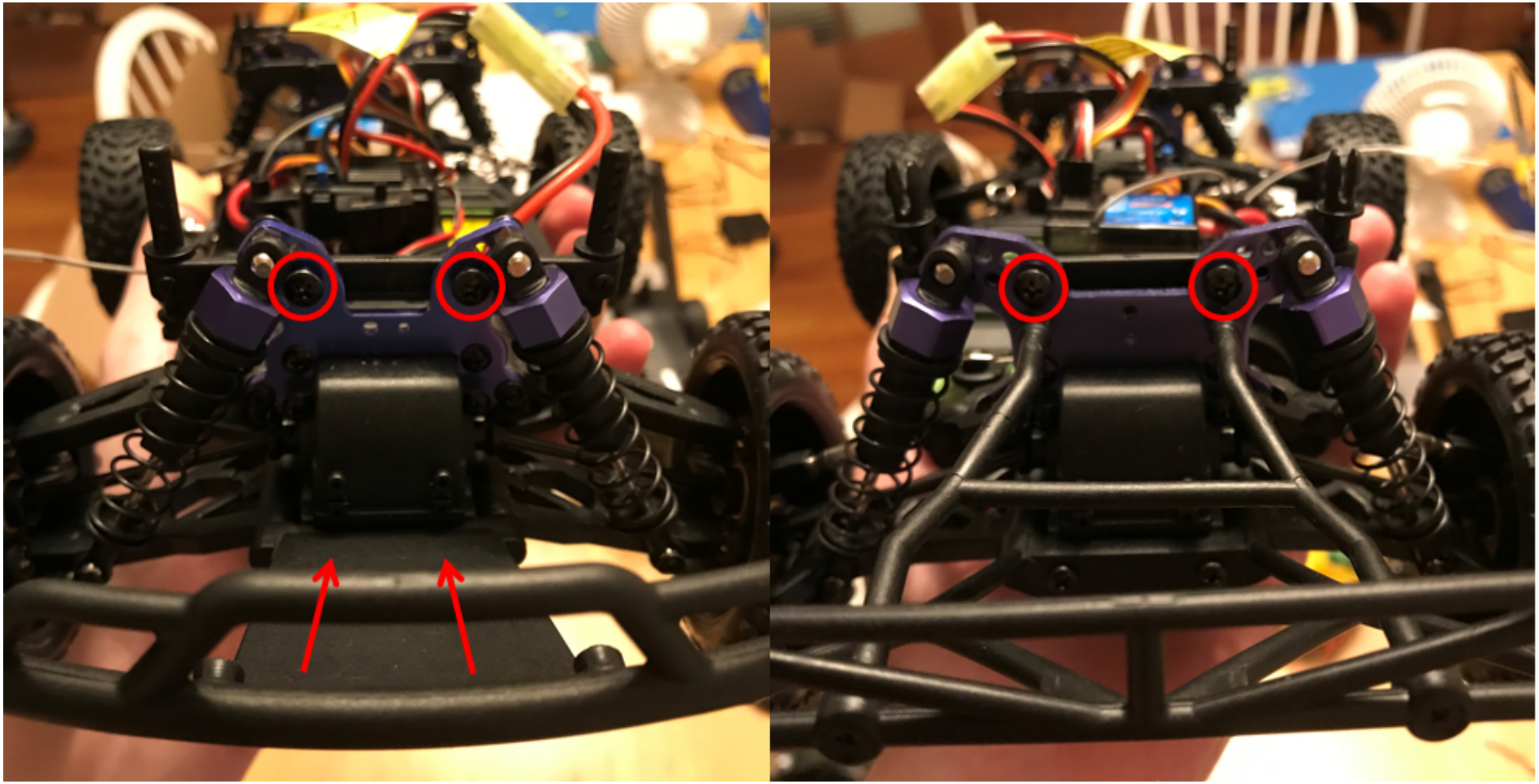
Some tips:

- Prop up your vehicle beneath the chassis, or turn upside down
- Battery may or may not be charged
  - NiMH batteries (like most batteries) are dangerous! Follow included charging instructions to ensure safe charging and operation
- Turn on remote controller first, then turn on ESC
- Test forward/reverse throttle, steering range

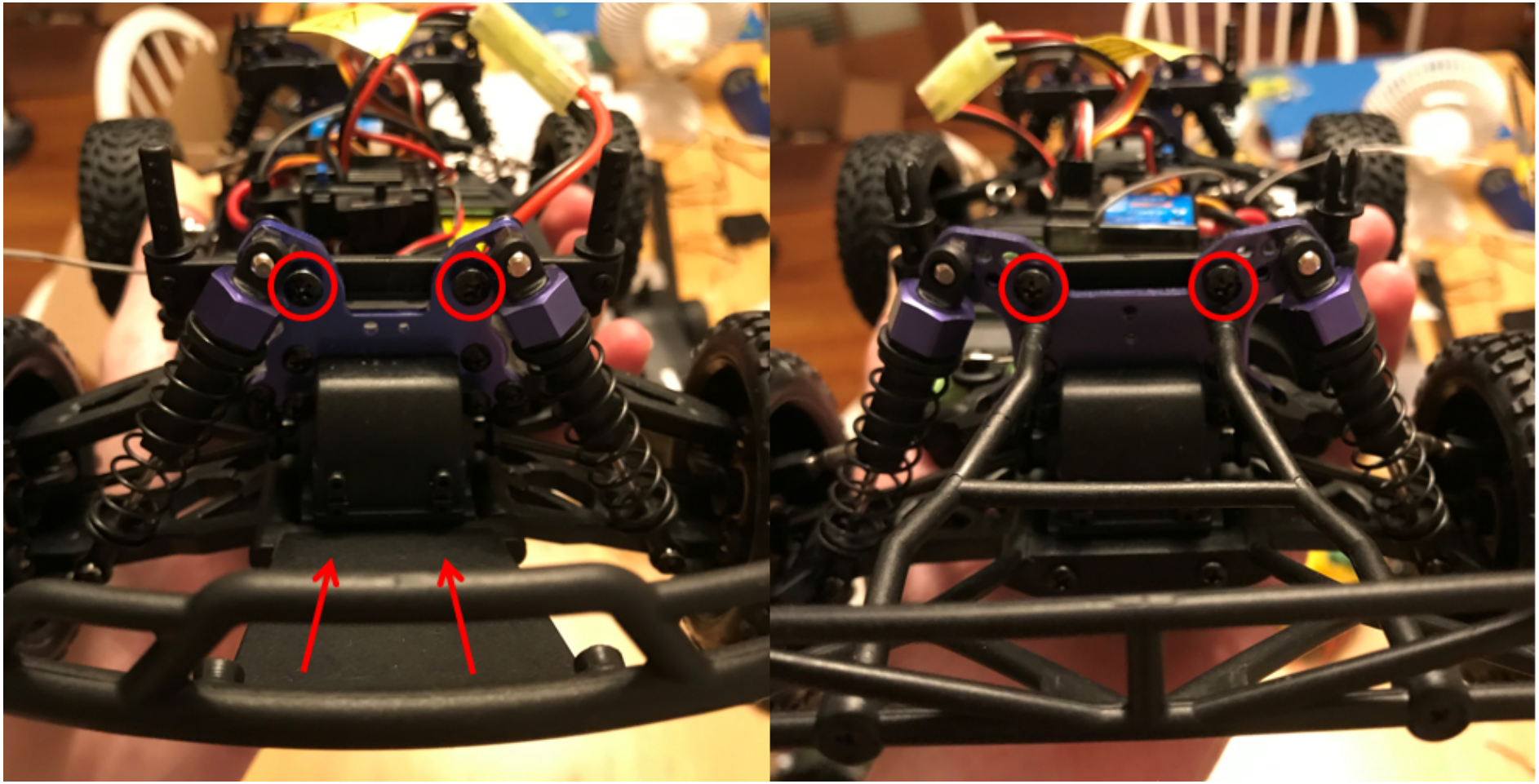
THE CAR

- Next comes the roll cage (and passengers...)
- Clip-on posts in rear, two screws in front
- Once off, can now access internal components!
- Note also the removable battery cover

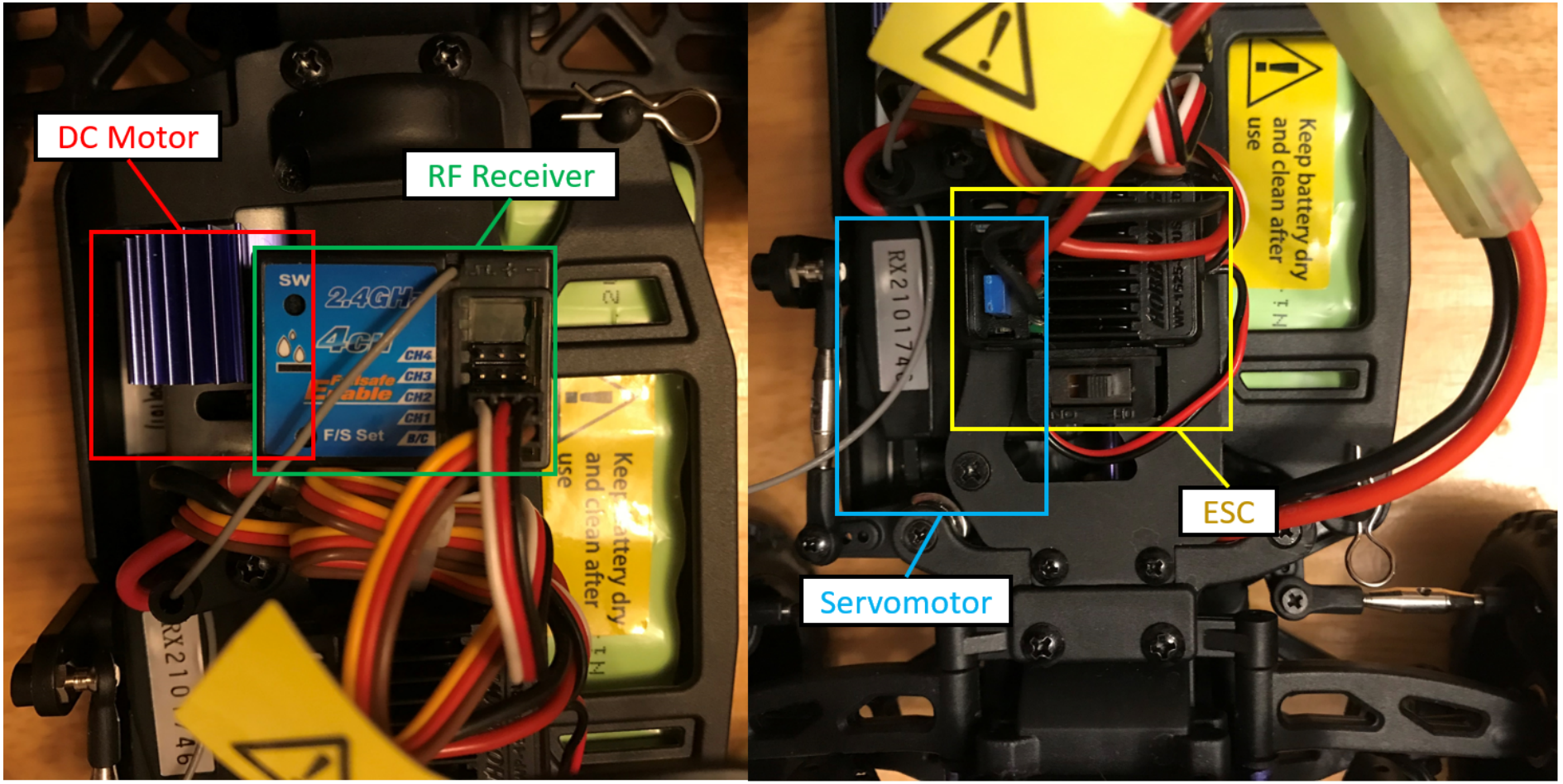




- Front and rear posts need to be removed
- Keep track of these fasteners for attaching adapters!

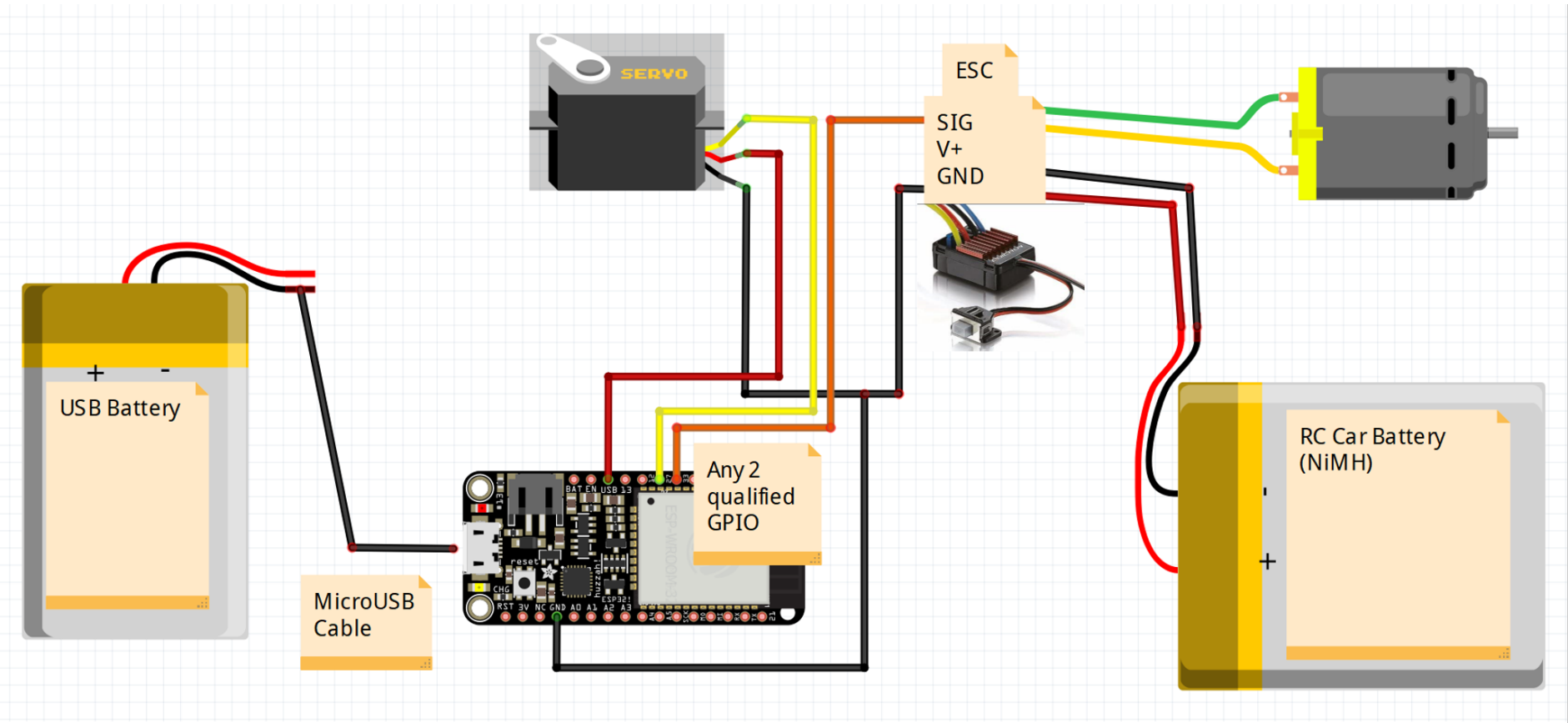


- May need to remove front bumper for accessing screws
- Be careful; front bumper is also used to secure wheel wings!



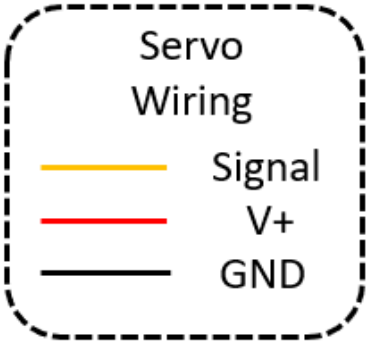
- RF Receiver; Note the array of channel connectors (Ch1: Servo, Ch2: ESC)
- ESC: Electronic Speed Controller; Note the ON/OFF switch for later

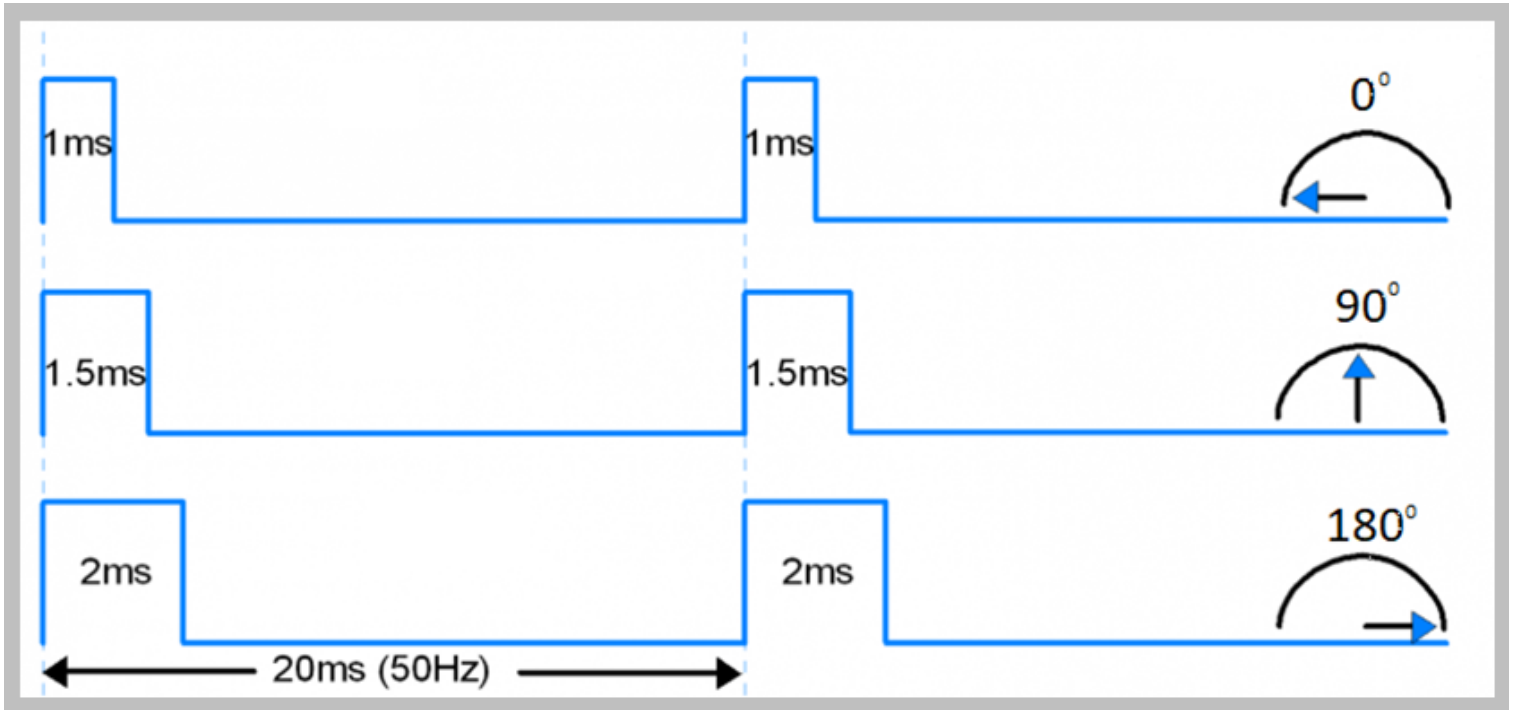
Wiring Diagram



- Use whichever qualified GPIO pins that suit your build
- IMPORTANT: Be careful not to swap power cables on NiMH battery
  - Should have Mini-Tamiya connector will be keyed, but it's possible that you may have color mismatch (red/black)

- Servo wiring has standard convention, 3-wire interface
- PWM control, 50Hz, (typically) between 1-2ms

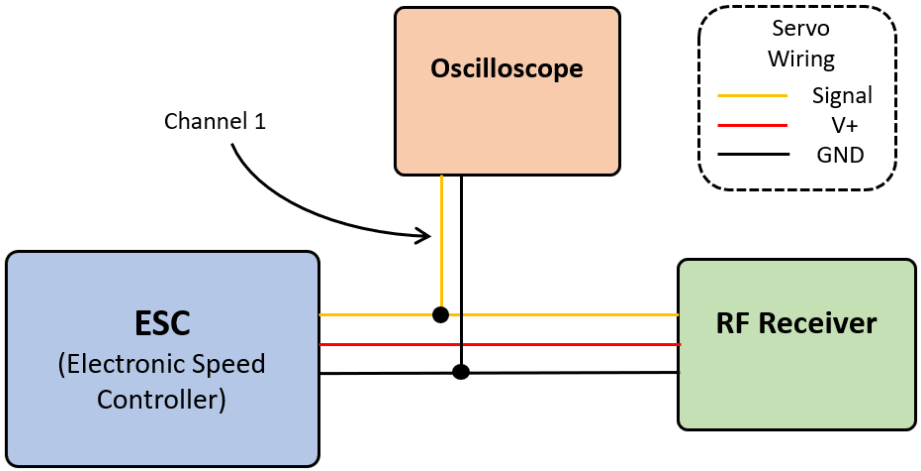


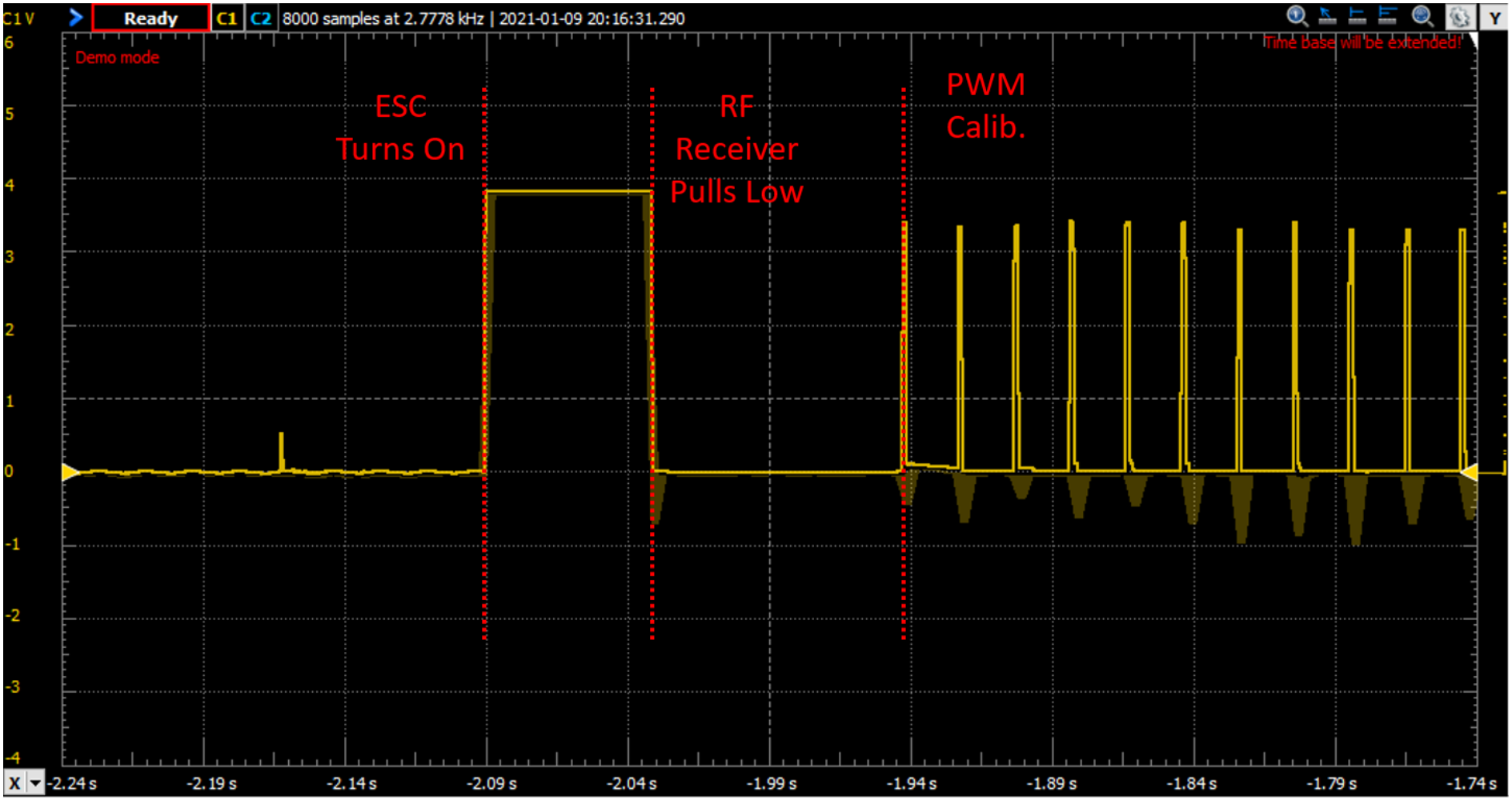


https://www.instructables.com/PANTILT-Camera-With-ESP32/

How to safely start your ESC!

- ESC has auto-callibration feature that can set throttle range vs. PWM
- Also, the ESC cleverly powers the RF receiver under normal operation
- Turns out that is not so useful for our purposes...
- We are going to investigate what happens before attempting to control our ESC!
  - Enter: the Digilent Analog Discovery 2 USB Oscilloscope

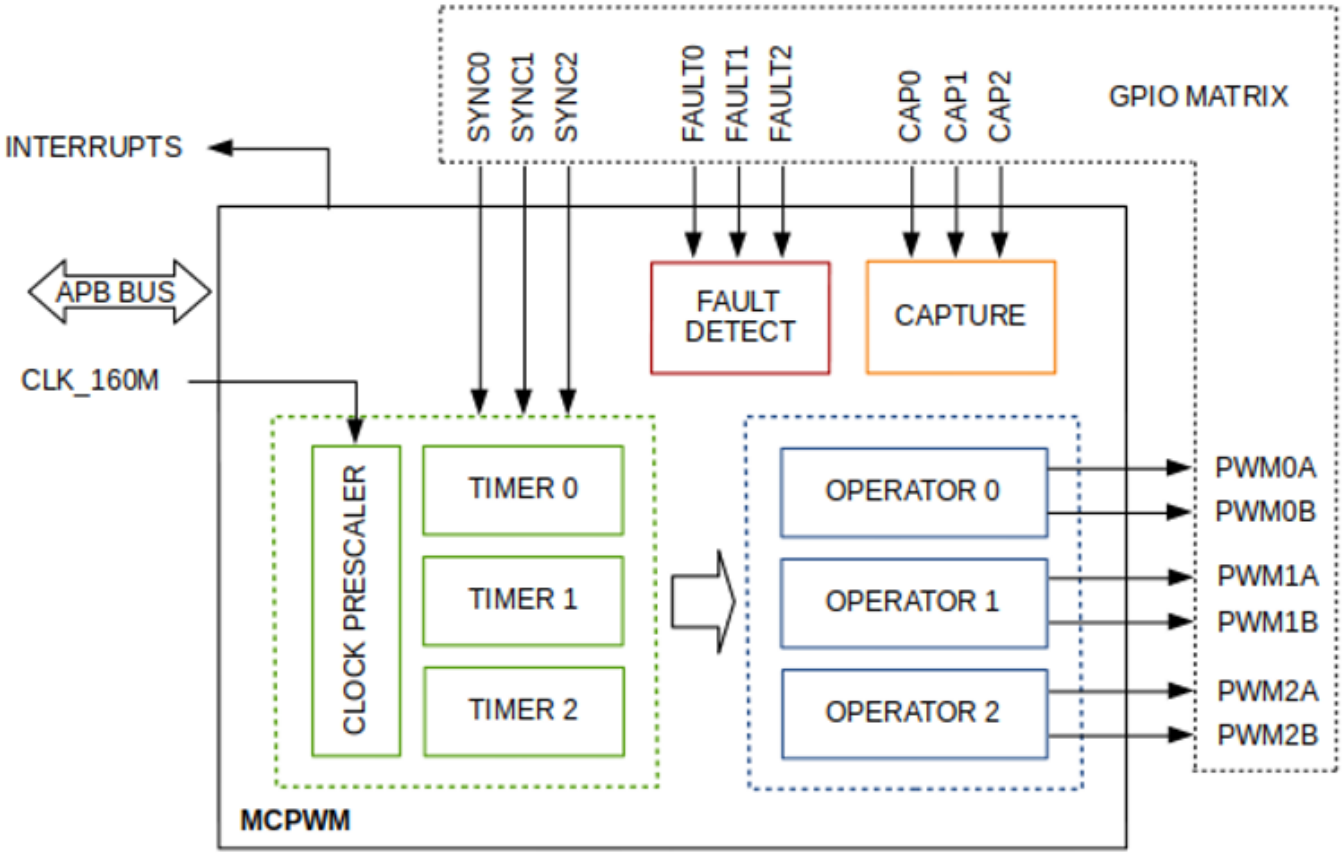


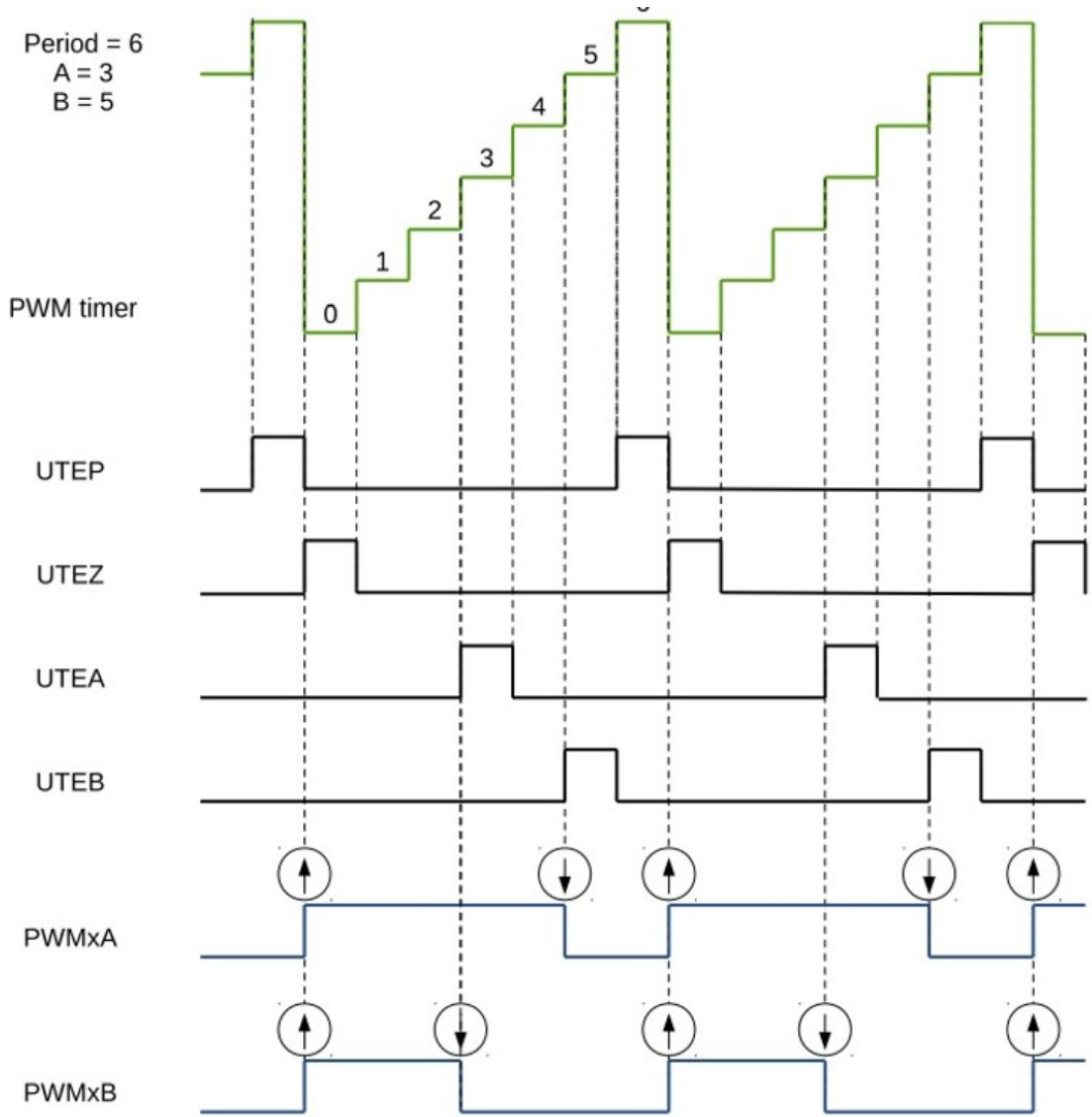


- Using Channel 1, trigger on rising edge
- PWM sent by RF receiver callibrates ESC for "zero" throttle

- Initial signal from ESC pulled to a high 5-6V depending on controller's signal pin impedance
  - **BE CAREFUL! This will potentially overload our 3.3V tolerant GPIO**
- Make sure you always initialize your pin before powering ESC manually
- Process should resemble the following:
  - Initialize PWM on throttle pin (so that signal is pulled low)
  - Turn on ESC
  - Send callibration PWM signal for 3 seconds
  - Should hear a confirmation beep from ESC

MCPWM Example




Period = 6
A = 3
B = 5
PWM timer
0
1
2
3
4
5
UTEP
UTEZ
UTEA
UTEB
PWMxA
PWMxB

```
void mcpwm_example_gpio_initialize(void)
{
    mcpwm_gpio_init(MCPWM_UNIT_0, MCPWM0A, 18);
        //Set GPIO 18 as PWM0A

    mcpwm_config_t pwm_config;
    pwm_config.frequency = 50;     //frequency = 50
        Hz
    pwm_config.cmpr_a = 0;     //duty cycle of
        PWMxA = 0
    pwm_config.cmpr_b = 0;     //duty cycle of
        PWMxb = 0
    pwm_config.counter_mode = MCPWM_UP_COUNTER;
    pwm_config.duty_mode = MCPWM_DUTY_MODE_0;
    mcpwm_init(MCPWM_UNIT_0, MCPWM_TIMER_0, &
        pwm_config);
}
```

```
void set_mcpwm_throttle(uint32_t *throttle)
{
    mcpwm_set_duty_in_us(MCPWM_UNIT_0,
       MCPWM_TIMER_0, MCPWM_OPR_A, throttle);
}
```

Let's try this on some actual hardware!

MCPWM controller demo for throttle control

... please work...

Git Collaboration Best Practices

We just wrote some code; we want to keep track of changes, versions, diffs, etc.; Time to commit!
However, we want to ensure that we are practicing safe collaboration...

- Check the status of your repository
  - See what changes were made; good for understanding your commits
  - Perfect time to catch unintended changes!
- **Pull/merge before commit**!
- One option: git stash your changes
  - Allows you to pull/merge from remote without risking conflicts
  - Can then git apply changes back on top
- Add (or stage) changes for commit
  - This is when you decide what to commit to your repository
- Commit! And add a descriptive message PLEASE.

What if we want to do something experimental that takes more than incremental commits to main?

- You can (and should) **create a branch**!
- Allows you to protect a clean and functional main branch
- When you're finished experimenting, you can either discard or merge into main!
- Maybe even a pull request?