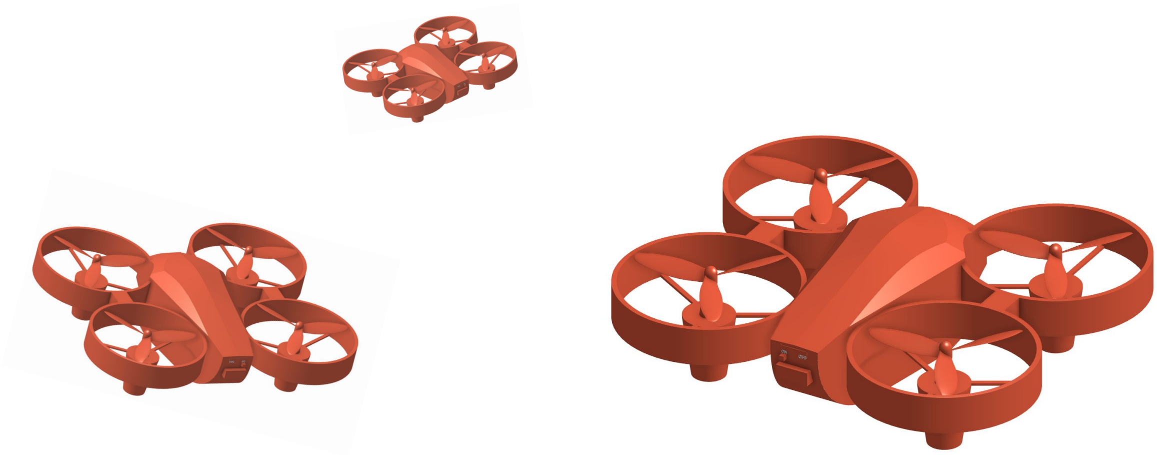
STAGE ONE EDUCATION

Hands-on Engineering Workshops

ROBOTICS WORKSHOP

ELECTRONICS & CODING



with

ARDUINO

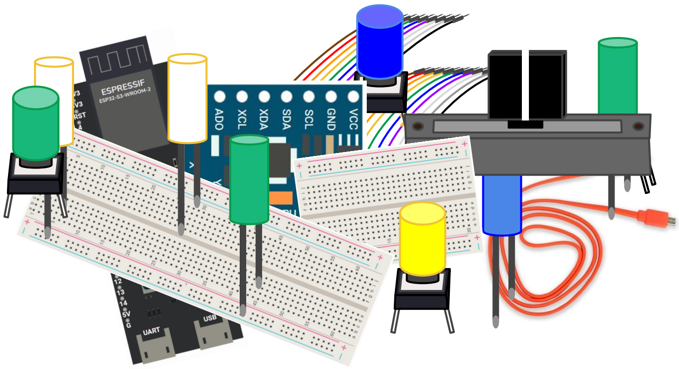
&

ESP32

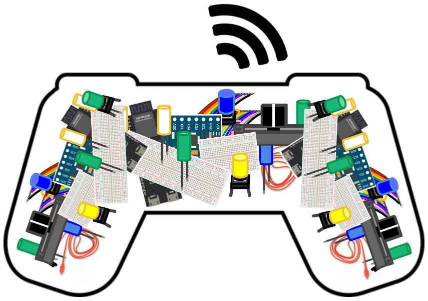
V 25.1

Background

Get ready to build a controller using simple electrical components!



You'll use this controller to send signals to your drone, allowing you to control its flight both manually and autonomously.

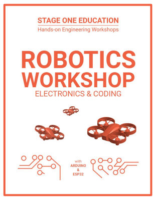


Along the way we will build, program, and optimize our circuit and drones flight.

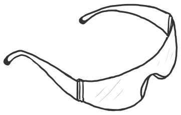
Parts that we'll use today

On your desk

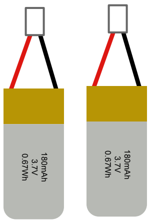
Laptop



Instructions



Safety Glasses



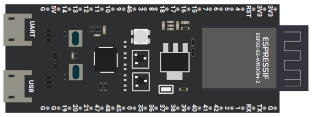
Drone Batteries

Open your laptop and connect to power

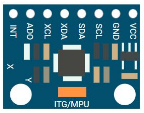
Check that you have all the parts we will use today

Electronics Box

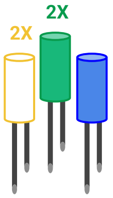
Drone



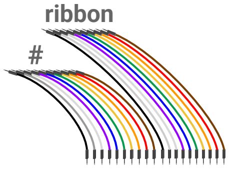
ESP32 Development Board



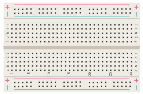
GY-521 Accelerometer



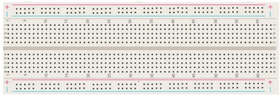
LED's



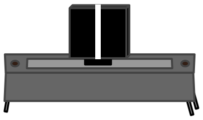
Wires



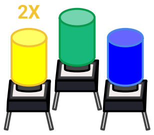
Small Breadboard



Large Breadboard



Slider Variable Resistors



Buttons

Drone Battery Charging Cable

USB to Micro USB

Start-Up

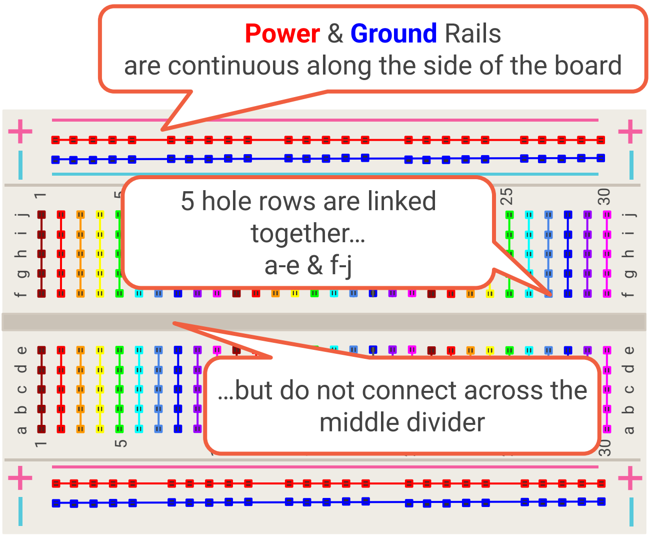
SAFETY GLASSES REQUIRED

Put on your safety glasses

BREADBOARDS

NEVER twist wires

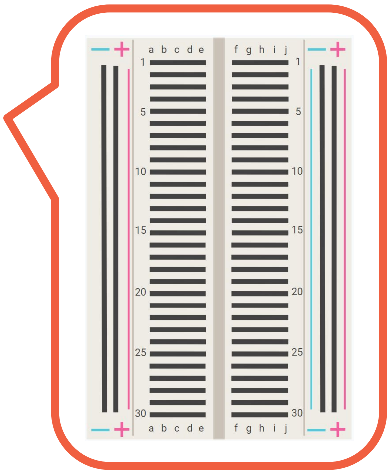
ALWAYS connect wires using the breadboard!



Power & Ground Rails are continuous along the side of the board

5 hole rows are linked together… a-e & f-j

…but do not connect across the middle divider



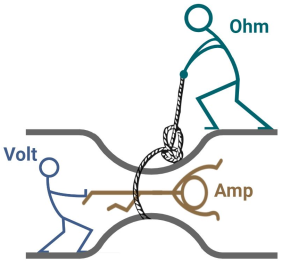
GOOD TO KNOW

Ohm's Law

Voltage = Current ✖ Resistance

Current (A)
measure the flow of electrical current in a circuit. It indicates how many electrons are passing a point in the circuit per second. It's measured in amperage (A)

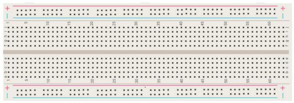
Voltage (V)
the electrical potential difference between two points in a circuit. It is measured in volts (V)

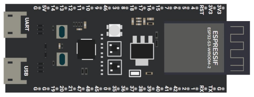


Resistance (Ω)
measures how much a component resists the flow of current. It's measured in ohms (Ω)

Drone Control Board Assembly

Parts we need

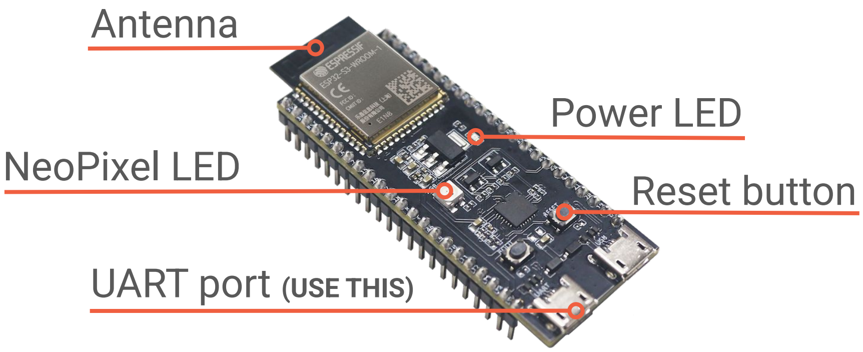

Large Breadboard


ESP32

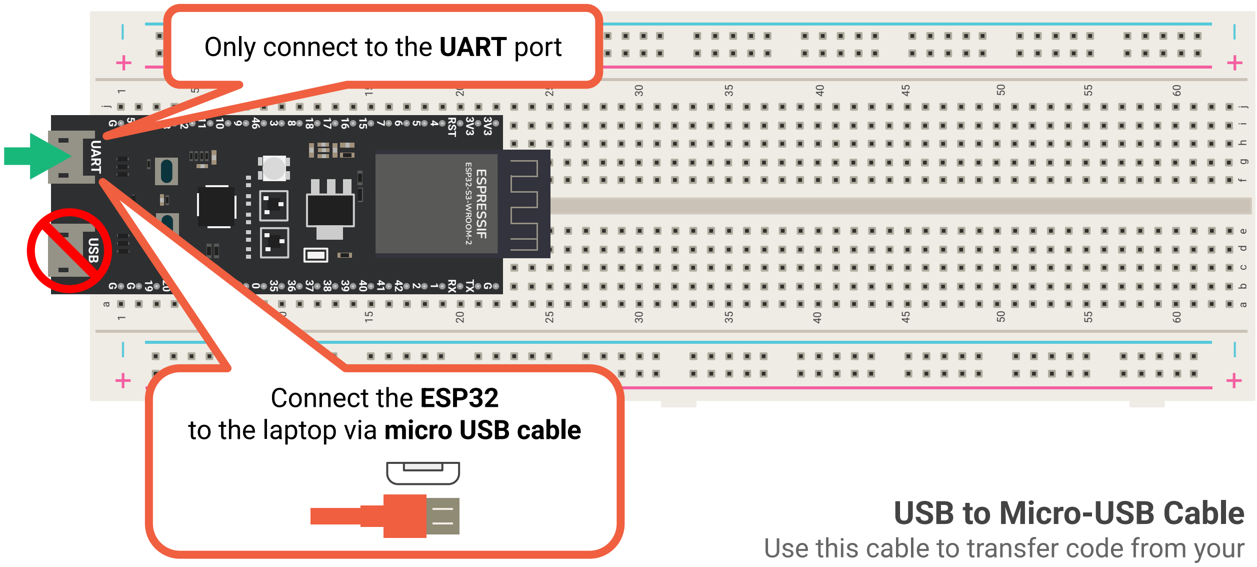
USB to Micro-USB

ESP32-S3-WROOM Development Module

This component is like a small computer that lets devices communicate. We'll use it to link to our drone's WiFi and create a controller to pilot the drone!



The **ESP32** will be installed on the **breadboard**



Only connect to the **UART** port

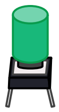
Connect the **ESP32**
to the laptop via **micro USB cable**

USB to Micro-USB Cable

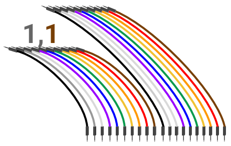
Use this cable to transfer code from your laptop to the ESP32. Only connect it when you're ready to upload to protect your circuit.

Takeoff Button Assembly

Parts we need



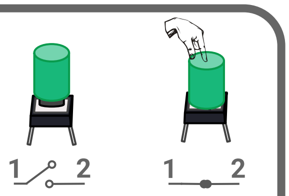
Green Button



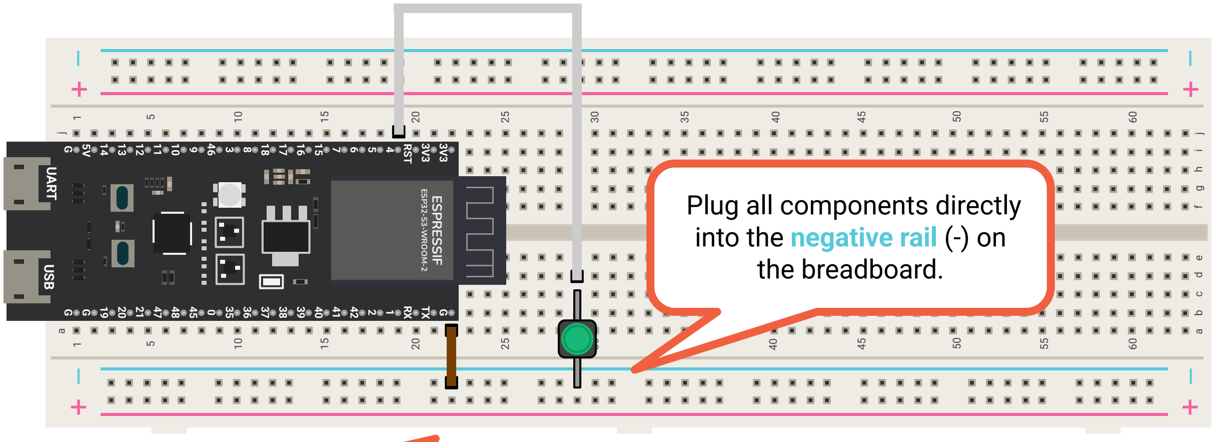
Wire

Two-Pin Button

- When the button is pressed, pins 1 and 2 connect
- Pressing the button can cause a bouncy connection, sending multiple signals.
- Two-pin buttons are used as on/off switches and triggers in various projects.



Schematic Symbol



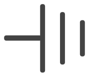
Let's use a **White** wire to link **pin 4** of the **ESP32** to the **Green** button.

Plug all components directly into the **negative rail** (-) on the breadboard.

Connect the **ESP32** to ground by using a **Brown** wire to link **pin G** to the **negative rail** (-) on the breadboard.

Grounding

- Ensures correct voltage levels by providing a common reference point.
- Reduces noise and interference for accurate readings and stable operation.
- Prevents shocks and protects components by safely redirecting excess electricity.



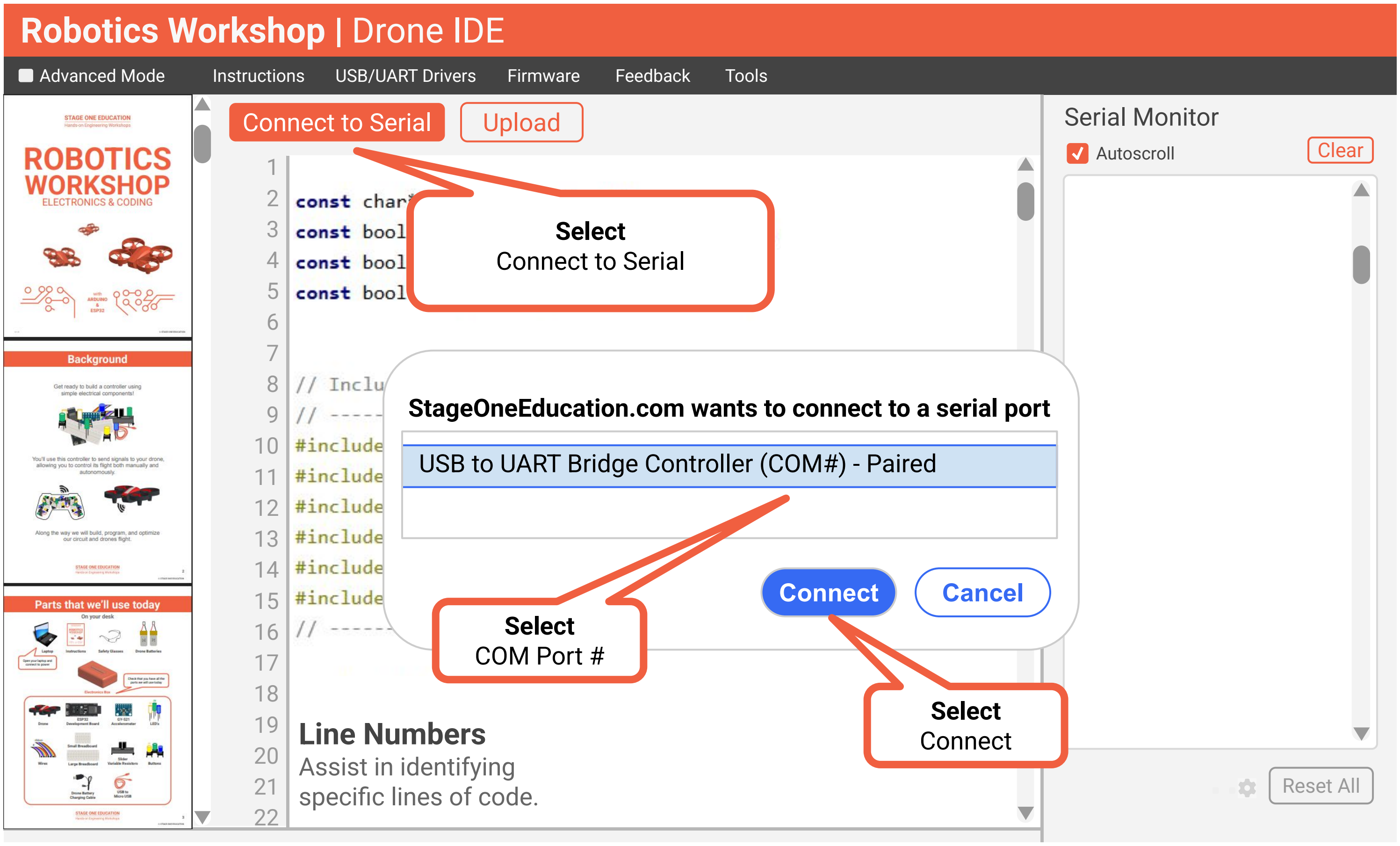
Schematic Symbol

Initiating Serial Communication

Connect the **Laptop** to the **USB cable**

Open Google Chrome

https://stageoneeducation.com/drone/



```
const char
const bool
const bool
const bool


// Inclu
// -----
#include
#include
#include
#include
#include
#include
// -----
```

Select Connect to Serial

StageOneEducation.com wants to connect to a serial port

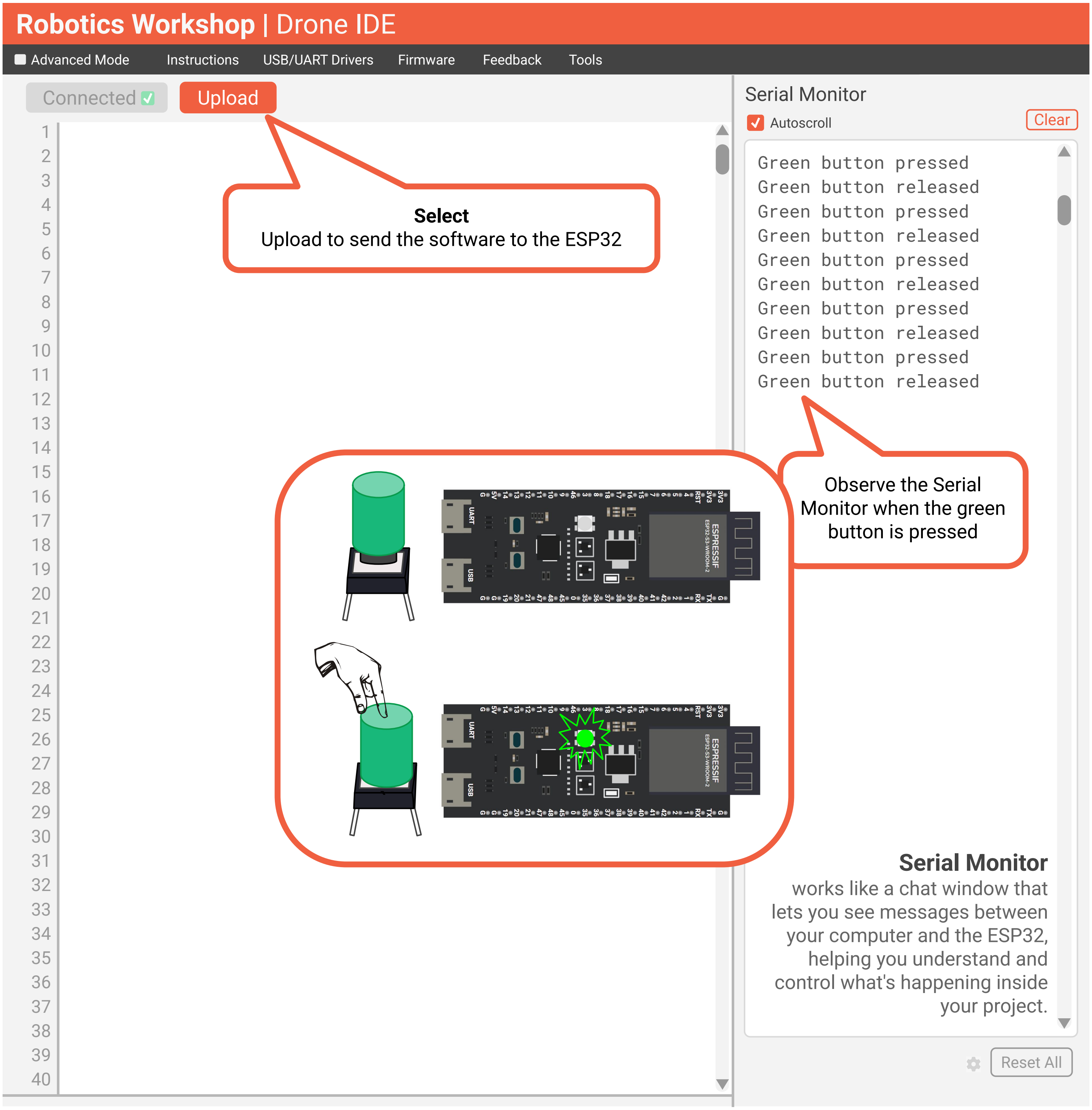
USB to UART Bridge Controller (COM#) - Paired

Select COM Port #

Select Connect

Line Numbers
Assist in identifying specific lines of code.

Upload



```
Green button pressed
Green button released
Green button pressed
Green button released
Green button pressed
Green button released
Green button pressed
Green button released
Green button pressed
Green button released
```

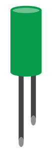
Select
Upload to send the software to the ESP32

Observe the Serial Monitor when the green button is pressed

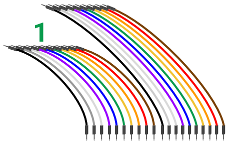
Serial Monitor works like a chat window that lets you see messages between your computer and the ESP32, helping you understand and control what's happening inside your project.

Takeoff LED

Parts we need



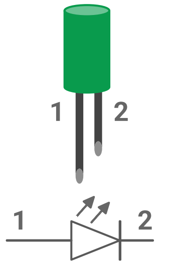
Green LED



Wire

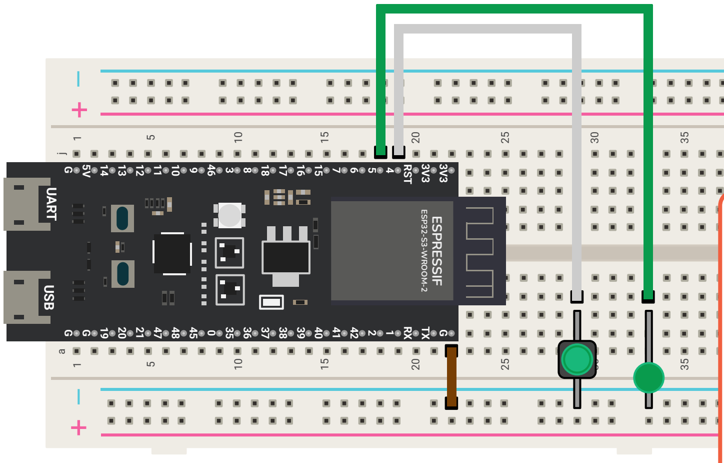
Light Emitting Diode - LED

- Long leg = Positive (anode)
- Short leg = Negative (cathode)
- LEDs require different voltages
  - Green, White, Blue: 2.8 - 3.6 volts
  - Red, Yellow: 1.8 - 2.3 volts
- The color of an LED depends on the materials it is made from.
  - EX: Green LEDs use indium gallium nitride (InGaN).



Schematic Symbol

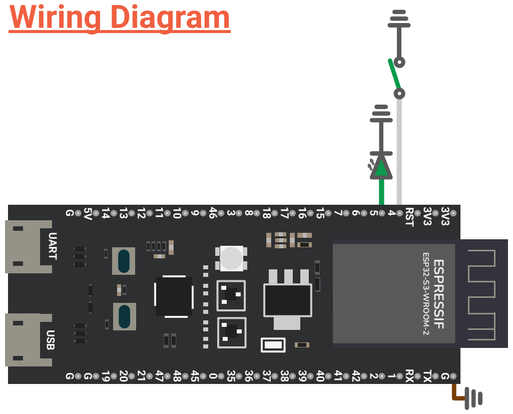
Let's use a **Green** wire to link **pin 5** of the **ESP32** to the **Green** LED.

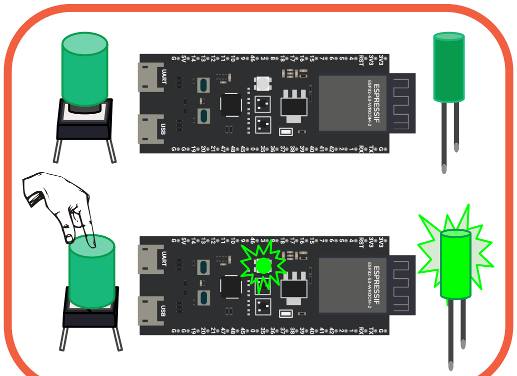


LED

short leg connects to GND

Wiring Diagram





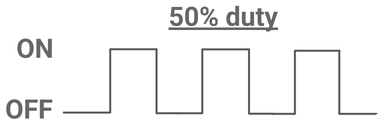
Common Ground

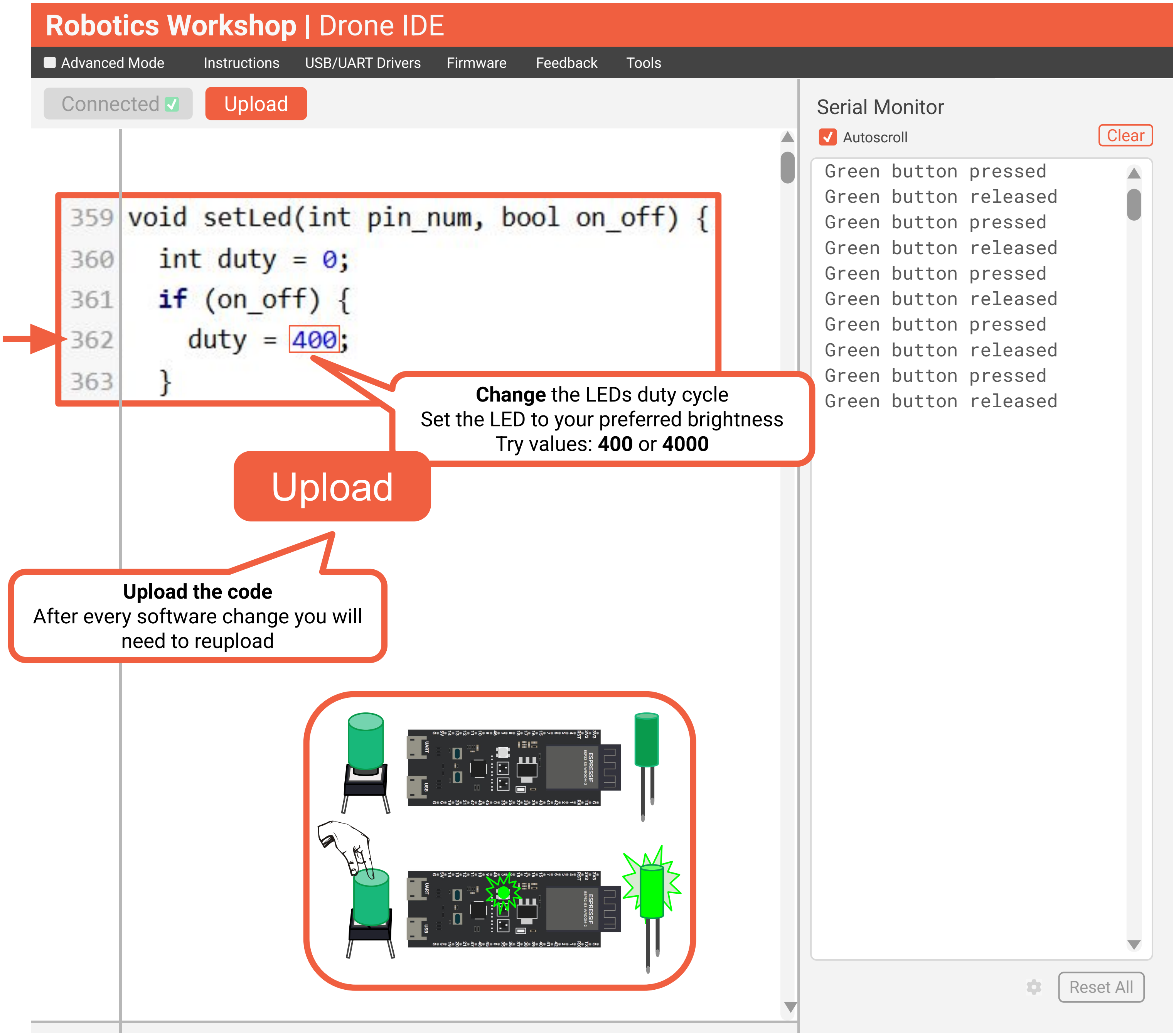
The three ground symbols are the same connection

LED Software Control

Pulse Width Modulation - PWM

- PWM turns a light on and off so quickly that the flickering is invisible to our eyes
- By adjusting how long the light is on compared to off, PWM controls brightness
- The duty cycle is the percentage of time the signal is on
  - higher means more power, lower means less.





```
void setLed(int pin_num, bool on_off) {
  int duty = 0;
  if (on_off) {
    duty = 400;
  }
```

```
Green button pressed
Green button released
Green button pressed
Green button released
Green button pressed
Green button released
Green button pressed
Green button released
Green button pressed
Green button released
```

Change the LEDs duty cycle
Set the LED to your preferred brightness
Try values: **400** or **4000**

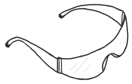
Upload the code
After every software change you will need to reupload

Cleared For Takeoff

Parts we need

Drone

Battery

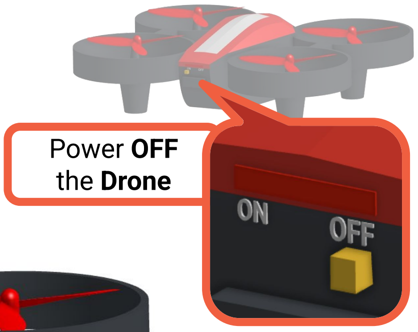


Safety Glasses

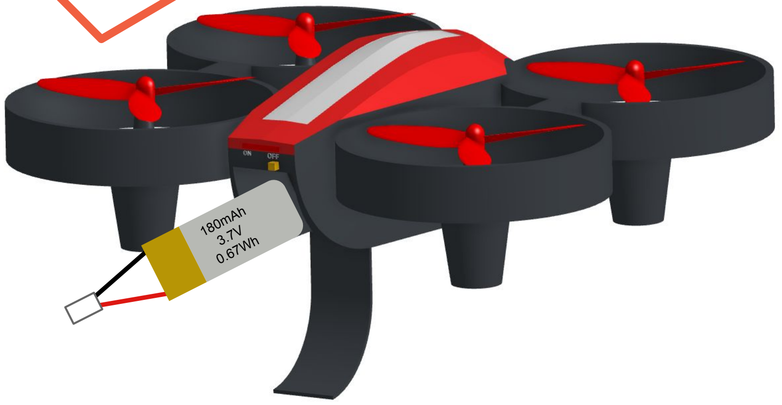
SAFETY GLASSES REQUIRED

Installing the Battery

1. Power OFF the drone
2. Open the battery compartment door
3. Insert the battery, silver side first
4. Connect the slotted battery plug
5. Close the battery compartment door



Power **OFF** the **Drone**



Airspace

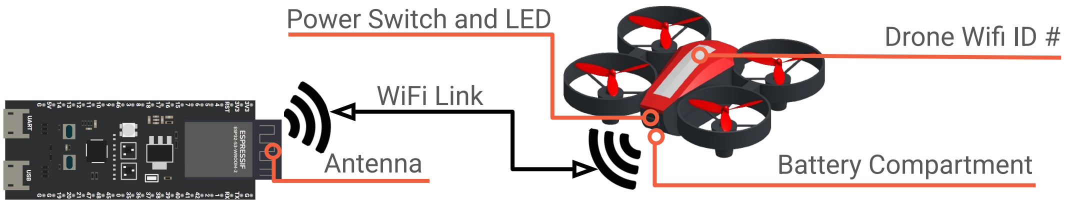
Stay in your airspace

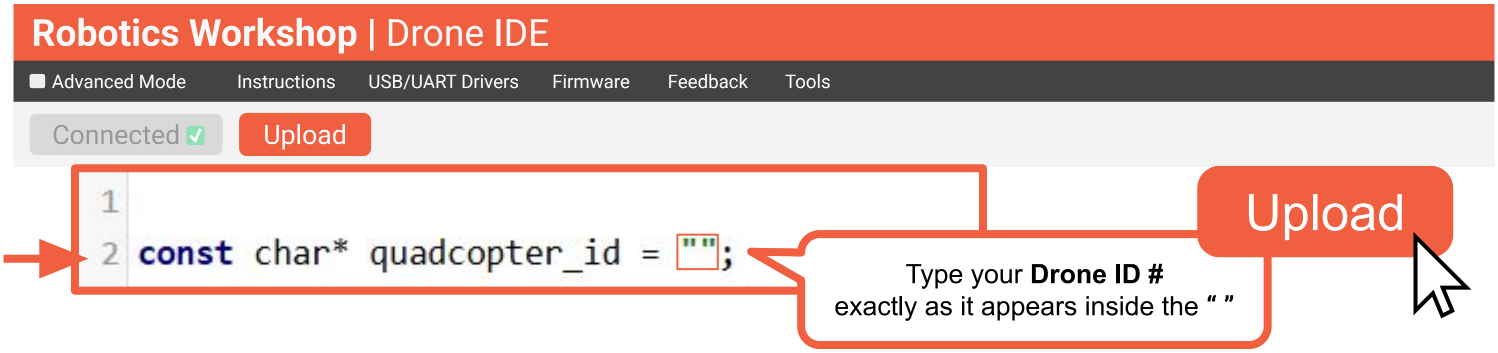
Your designated airspace is directly around your table. Keep all test flights within this space.

Drone Sync

Wireless Fidelity (WiFi) Drone

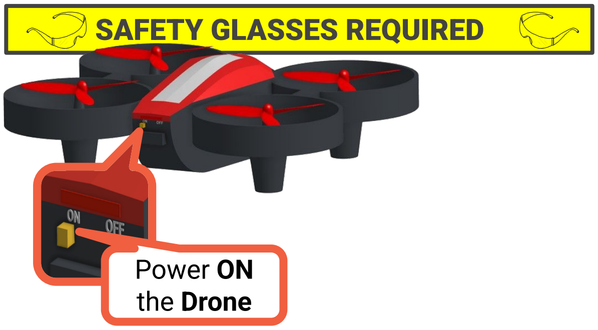
- The drone has a WiFi transmitter that creates a local network for the ESP32 to connect to.
- The ESP32 will send signals via WiFi to adjust propeller speeds controlling the drone.
- Powered by a LiPo battery, which offers high energy density in a lightweight package.



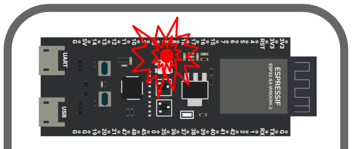


```
const char* quadcopter_id = "";
```

Type your **Drone ID #** exactly as it appears inside the “ ”

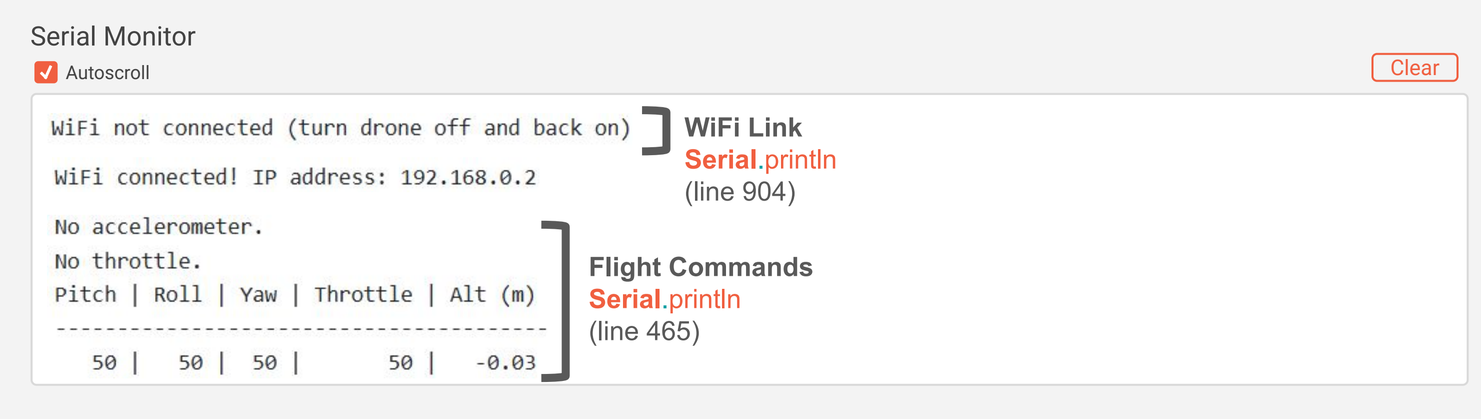


Power **ON** the **Drone**



WiFi Not Connected

Power cycle the drone by turning it OFF and ON



```
WiFi not connected (turn drone off and back on)

WiFi connected! IP address: 192.168.0.2

No accelerometer.
No throttle.
Pitch | Roll | Yaw | Throttle | Alt (m)
-----------------------------------------
   50 |   50 |  50 |       50 |   -0.03
```

WiFi Link
Serial.println
(line 904)

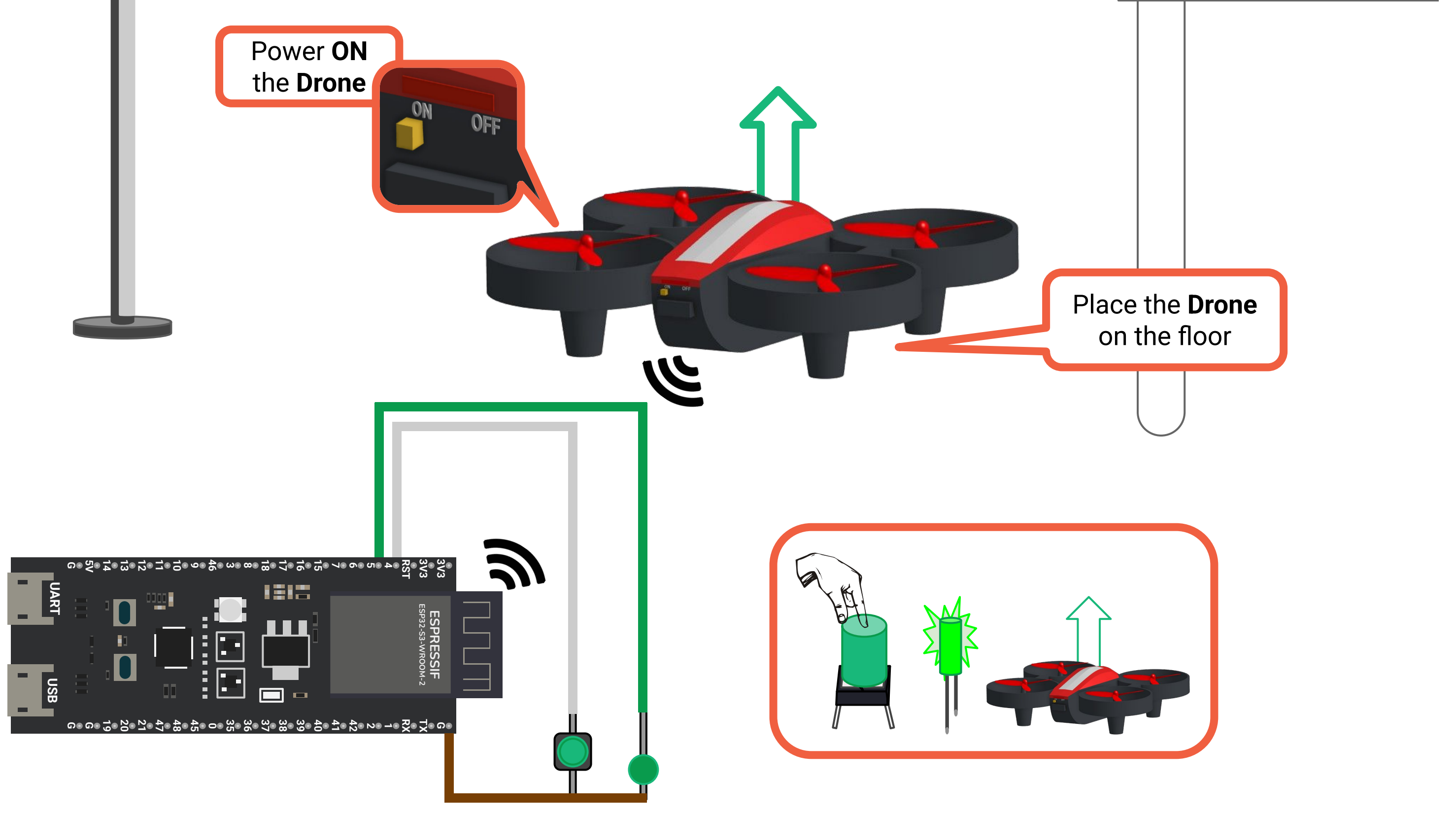
Flight Commands
Serial.println
(line 465)

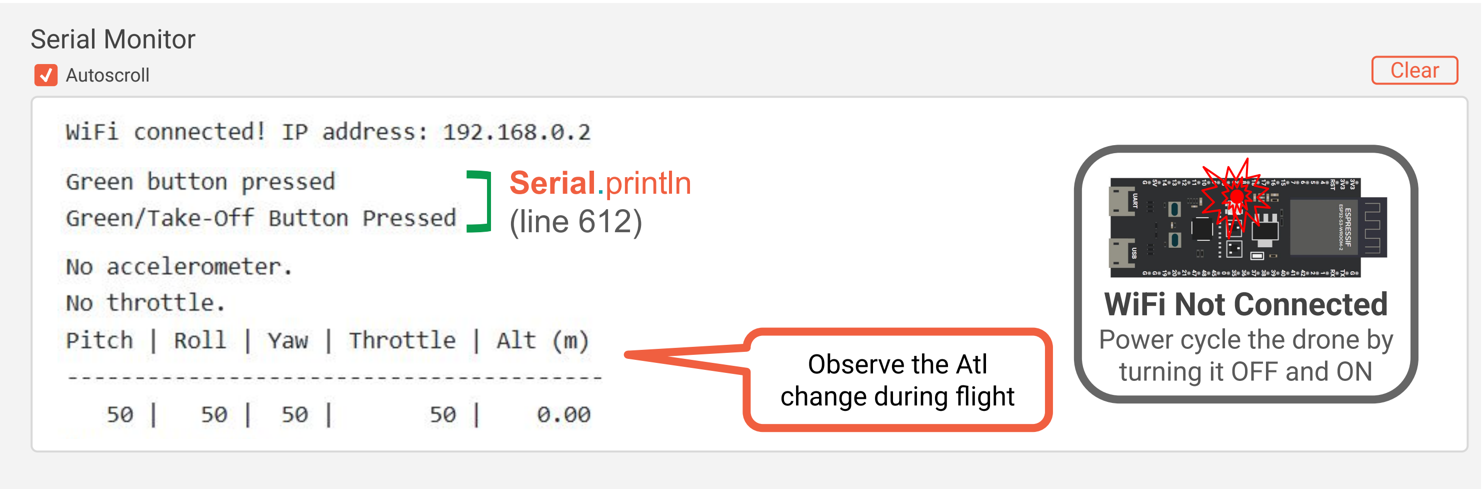
Takeoff Button Test Flight

SAFETY GLASSES REQUIRED

Flight Sequence

Takeoff is a crucial maneuver. Press the green button for a smooth takeoff; the drone will ascend for 1 second.





```
WiFi connected! IP address: 192.168.0.2

Green button pressed
Green/Take-Off Button Pressed

No accelerometer.
No throttle.
Pitch | Roll | Yaw | Throttle | Alt (m)
-----------------------------------------
   50 |   50 |  50 |       50 |    0.00
```

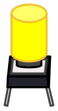
Serial.println (line 612)

Observe the Atl change during flight

WiFi Not Connected
Power cycle the drone by turning it OFF and ON

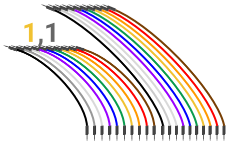
Stop Button Assembly

Parts we need



Yellow Button

White LED



Wire

Stop vs. Landing

To control our soon-to-be autonomous drone, we'll install a stop button that immediately ends the flight. This is different from landing, which is a controlled maneuver we'll program later.

Analyze the #define section of the code (lines 47 to 67) to see what **ESP32** pins to connect to the YELLOW_BUTTON_PIN_BASE and WHITE_LED_PIN_BASE

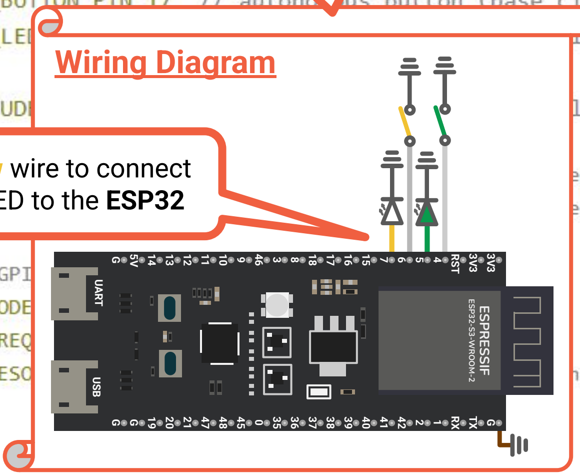
#define

- lets you give nicknames to numbers, making your code easier to read and change
- It doesn't use extra memory because it replaces the nickname with a number before your code runs

```
45  // IO pins
46
47  #define GREEN_BUTTON_PIN 4  // take-off/gyro-reset button (base circuit)
48  #define GREEN_LED_PIN_BASE 5  // take-off indicator (base circuit)
49  #define GREEN_LED_PIN_REMOTE 11  // take-off indicator (remote)
50
51  #define YELLOW_BUTTON_PIN_BA                                  it)
52  #define YELLOW_BUTTON_PIN_R
53  #define WHITE_LED_PIN_BASE                                    )
54  #define WHITE_LED_PIN_REMOT
55
56  #define BLUE_BUTTON_PIN 17  // autono    us button (base circuit)
57  #define BLUE_LE                                              rcuit)
58
59  #define ALTITUD                                     le wire (base circuit)
60
61                                                          emote)
62                                                          emote)
63
64  // Define the GPI
65  #define LEDC_MODE
66  #define LEDC_FREQ
67  #define LEDC_RESO
```

Use a **Gray** wire to connect the **Yellow** button to the **ESP32**

Use a **Yellow** wire to connect the **White** LED to the **ESP32**

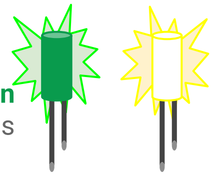

Wiring Diagram
G 5V 14 13 12 11 10 9 46 3 8 18 17 16 15 7 6 5 4 RST 3V3 3V3
UART
USB
ESPRESSIF
ESP32-S3-WROOM-2
G G 19 20 21 47 48 45 0 35 36 37 38 39 40 41 42 2 1 RX TX G

Macros
Predefined symbolic labels that replace specified values or code in a program before compilation.

Stop Button Assembly

Flight Mode

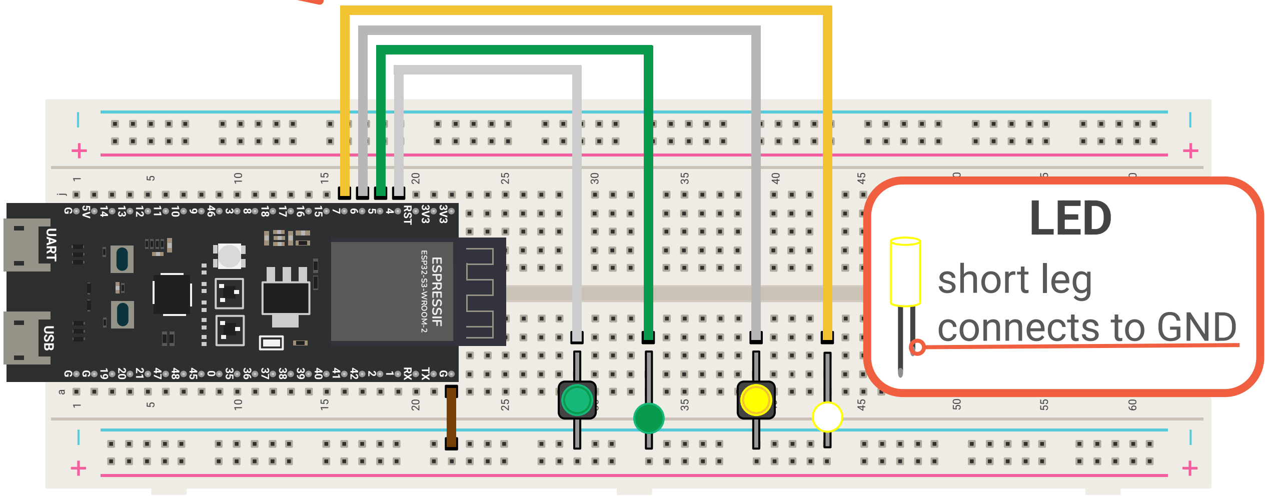
We will utilize LEDs to provide visual indicators of our flight controller's state



Green
Flight in progress

White
Disarmed and in standby

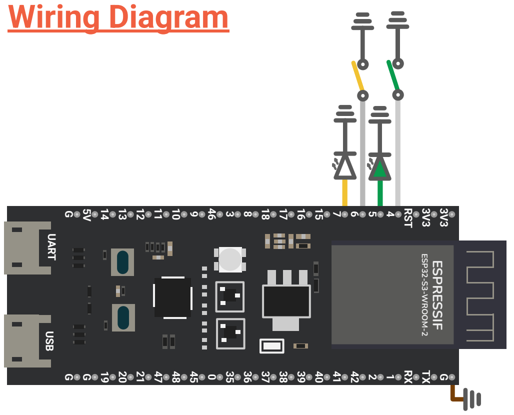
Check your circuit is correct



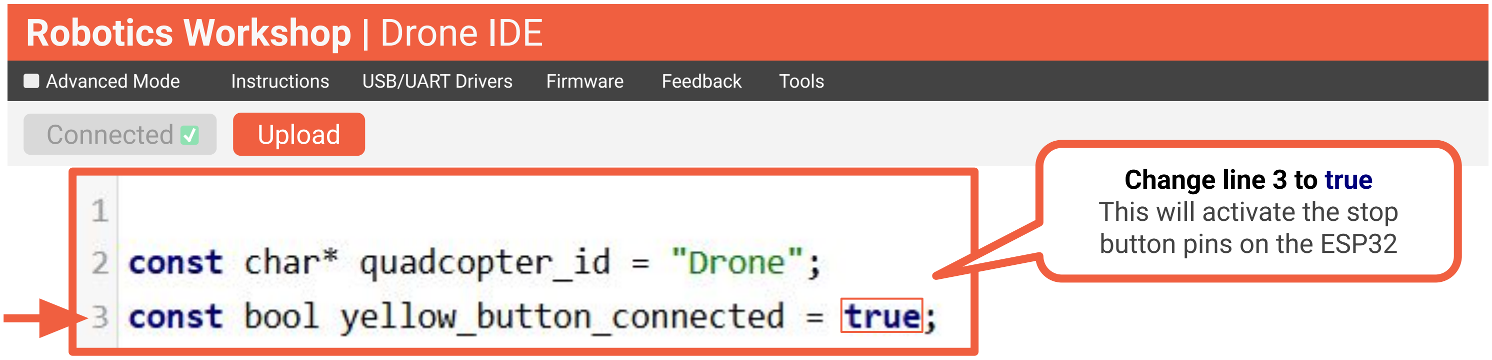
LED

short leg connects to GND

Wiring Diagram



Activate STOP

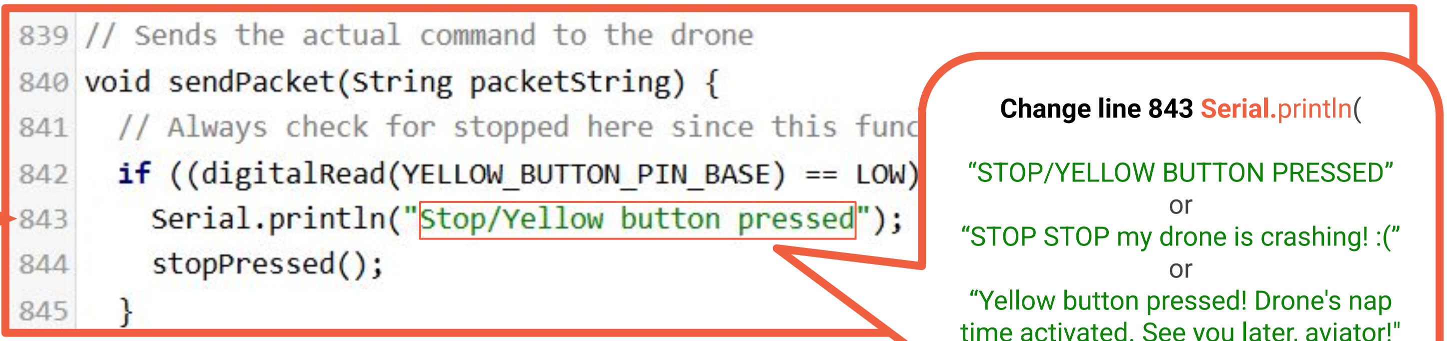


```
1
2 const char* quadcopter_id = "Drone";
3 const bool yellow_button_connected = true;
```

Change line 3 to true
This will activate the stop button pins on the ESP32

Serial.print("Hello, world!");

- Sends data from a microcontroller (e.g., ESP32) to a computer via a serial connection (USB)
- The text between the quotation marks (" ") will be sent and displayed on the serial monitor



```
839 // Sends the actual command to the drone
840 void sendPacket(String packetString) {
841   // Always check for stopped here since this func
842   if ((digitalRead(YELLOW_BUTTON_PIN_BASE) == LOW)
843     Serial.println("Stop/Yellow button pressed");
844     stopPressed();
845   }
```

Change line 843 Serial.println(

"STOP/YELLOW BUTTON PRESSED"
or
"STOP STOP my drone is crashing! :("
or
"Yellow button pressed! Drone's nap time activated. See you later, aviator!"
);

if();

- checks whether a condition is true or false. If the condition is true, the code inside the if block runs.
- If the condition is false, this code is skipped.

Upload

Upload the code

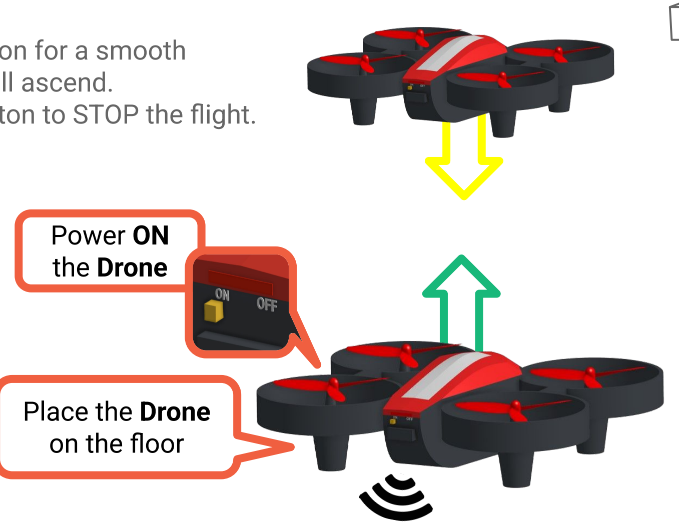
//
In Arduino, // is used to write a comment, which is like a note for humans to read that the computer ignores when running the code.

STOP Button Test Flight

SAFETY GLASSES REQUIRED

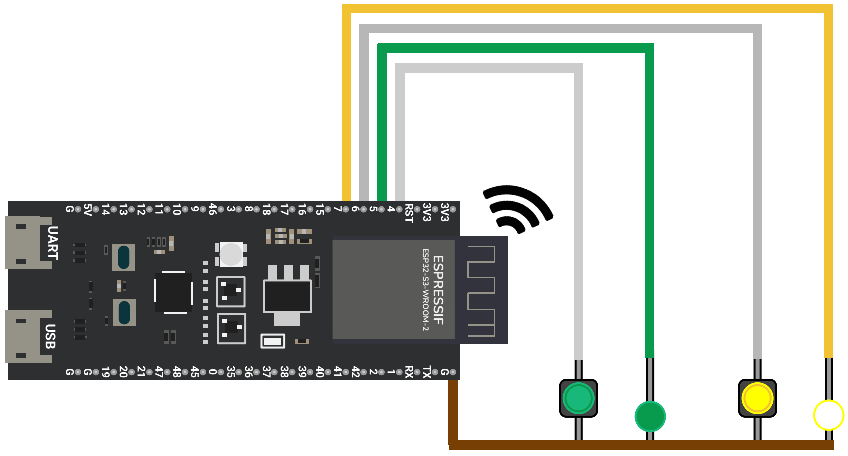
Flight Sequence

Press the green button for a smooth takeoff; the drone will ascend.
Press the yellow button to STOP the flight.

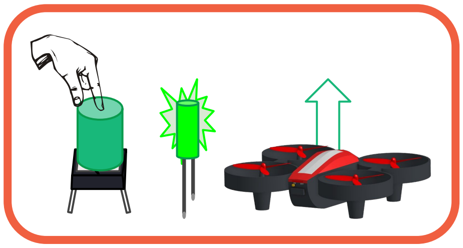


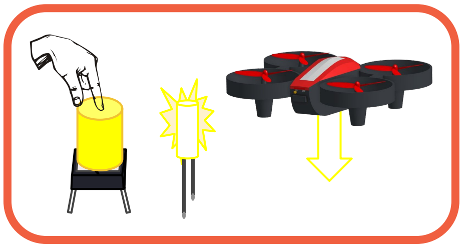
Power **ON** the **Drone**

Place the **Drone** on the floor



ESPRESSIF
ESP32-S3-WROOM-2
UART
USB
G 5V 14 13 12 11 10 9 46 3 8 18 17 16 15 7 6 5 4 RST 3V3 3V3
G G 19 20 21 47 48 45 0 35 36 37 38 39 40 41 42 2 1 RX TX G

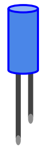




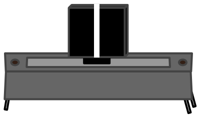
Not connecting…
Power cycle the drone by turning it OFF and ON

Altitude Control Slider

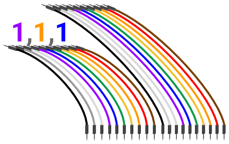
Parts we need



Blue LED



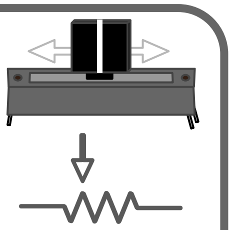
Slider



Wire

Variable Resistor - Slider

- Change circuit resistance by moving the slider, controlling electrical flow
- Sliding closer to one end decreases resistance there and increases it at the other end
- Common in volume controls, light dimmers, and sensor inputs, variable resistors are available in linear (straight slider) and rotary (knob) styles

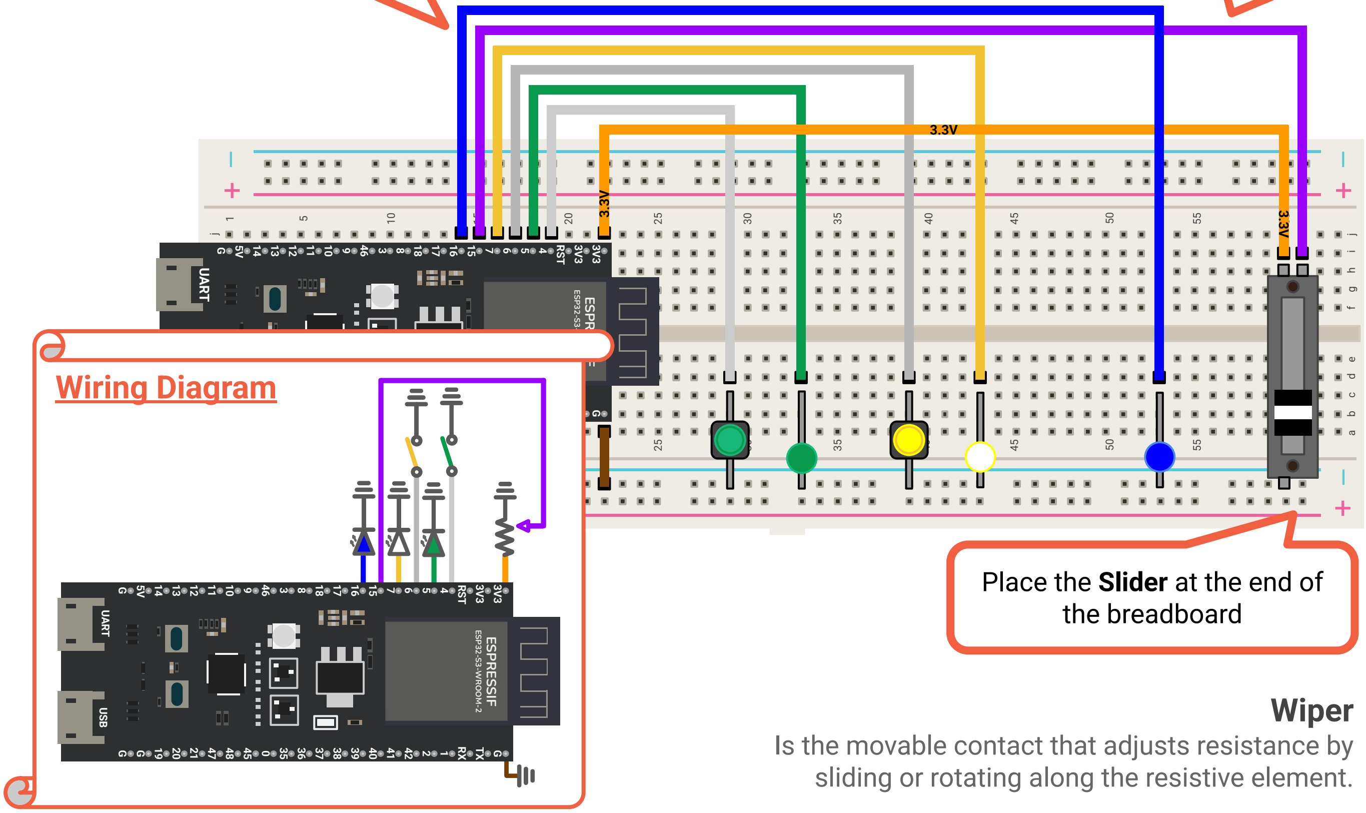


Schematic Symbol

Use a **blue** wire to connect **pin 16** to the **blue LED**

Use a **purple** wire to connect **pin 15** to the **slider - right pin**

Use a **orange** wire to connect **3.3V** to the **slider - left pin**

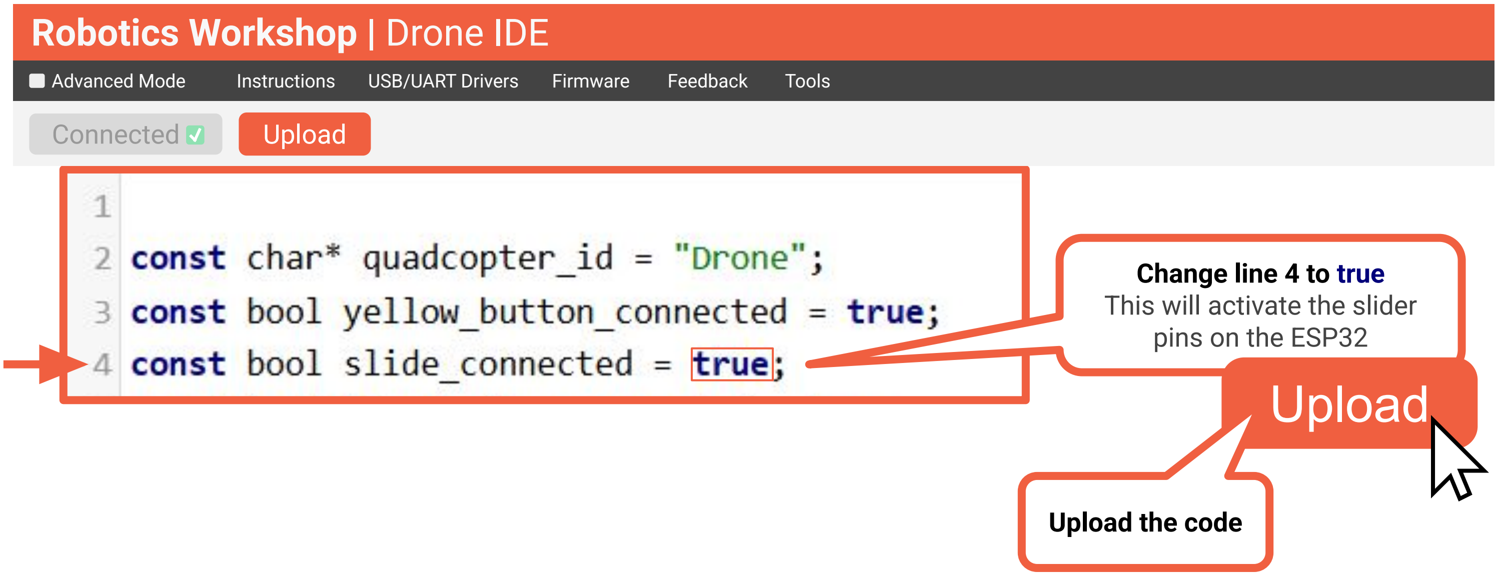


Place the **Slider** at the end of the breadboard

Wiper

Is the movable contact that adjusts resistance by sliding or rotating along the resistive element.

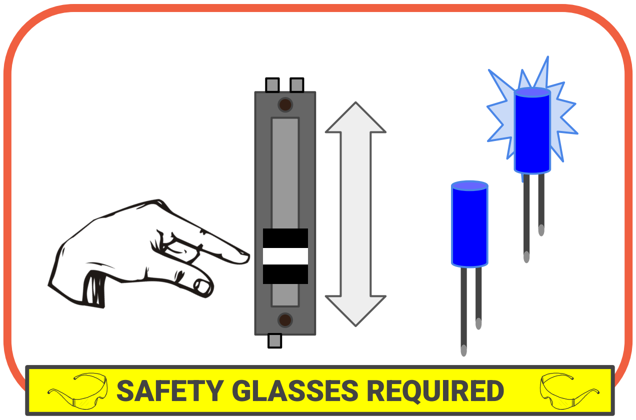
Activate Altitude Control



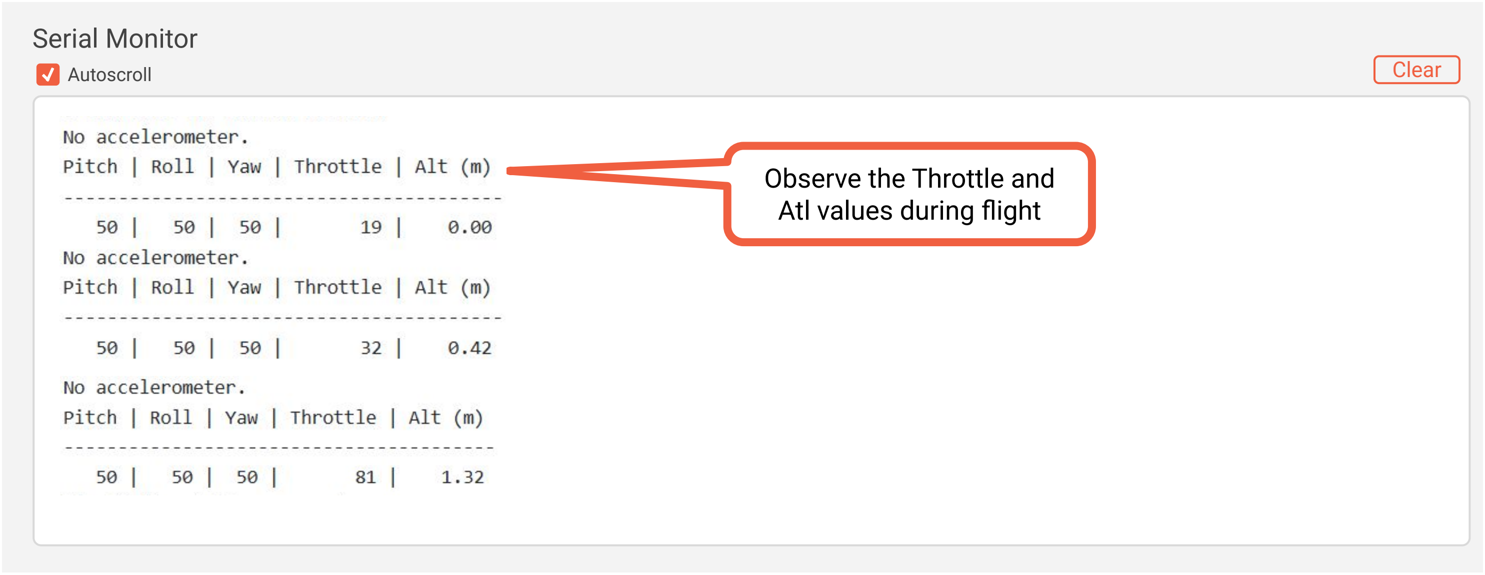
```
1
2 const char* quadcopter_id = "Drone";
3 const bool yellow_button_connected = true;
4 const bool slide_connected = true;
```

Change line 4 to true
This will activate the slider pins on the ESP32

Upload the code



SAFETY GLASSES REQUIRED



```
No accelerometer.
Pitch | Roll | Yaw | Throttle | Alt (m)
----------------------------------------
   50 |   50 |  50 |       19 |    0.00
No accelerometer.
Pitch | Roll | Yaw | Throttle | Alt (m)
----------------------------------------
   50 |   50 |  50 |       32 |    0.42
No accelerometer.
Pitch | Roll | Yaw | Throttle | Alt (m)
----------------------------------------
   50 |   50 |  50 |       81 |    1.32
```

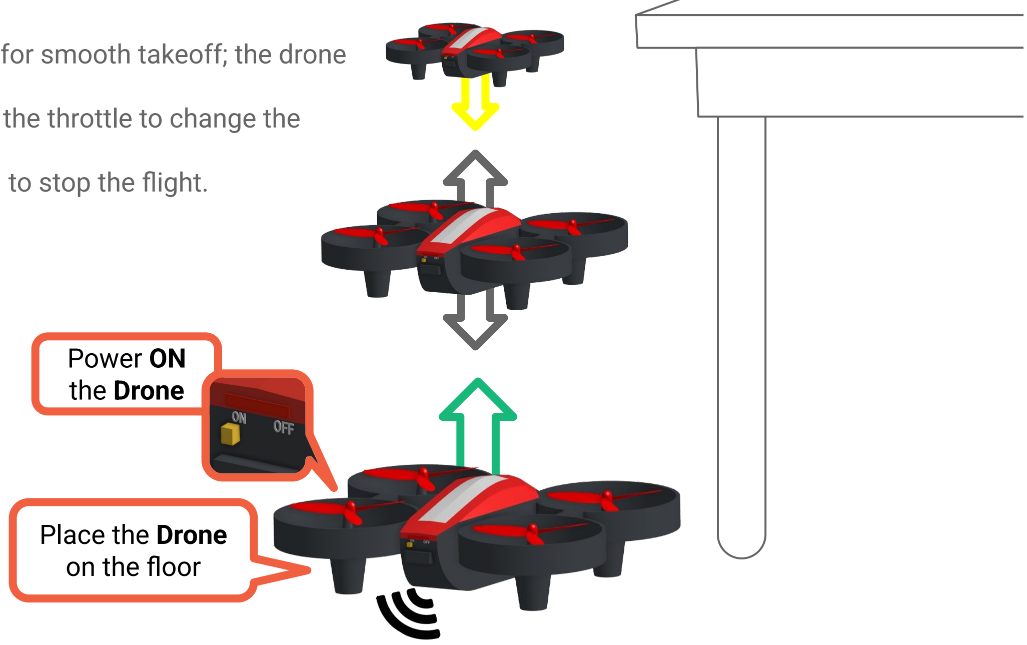
Observe the Throttle and Atl values during flight

Altitude Control Test Flight

SAFETY GLASSES REQUIRED

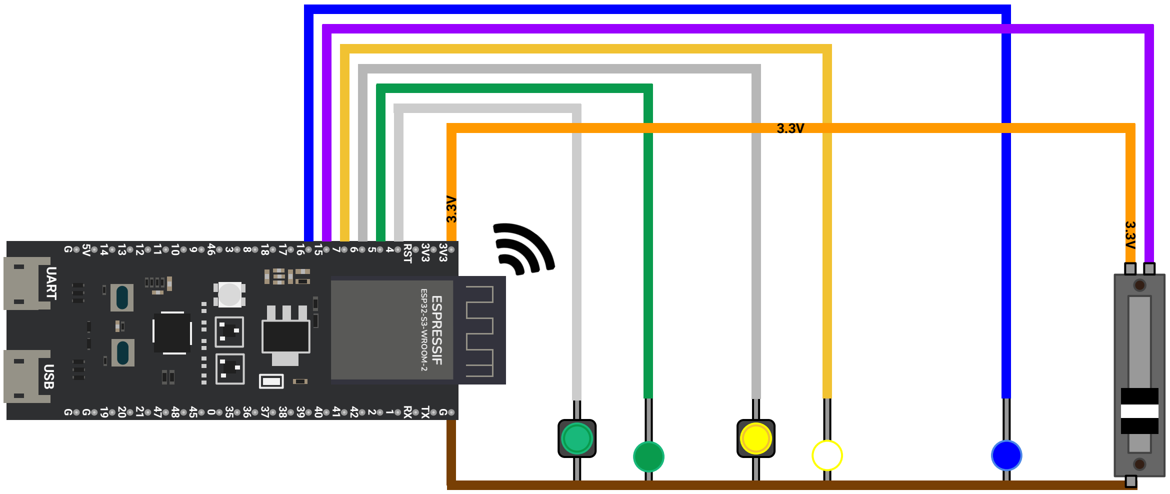
Flight Sequence

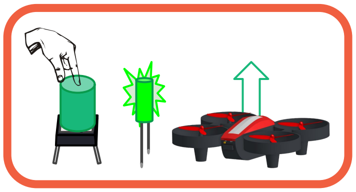
Press the green button for smooth takeoff; the drone will ascend.
Use the slider to adjust the throttle to change the drone's altitude.
Press the yellow button to stop the flight.

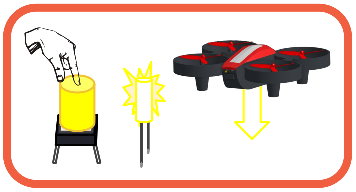


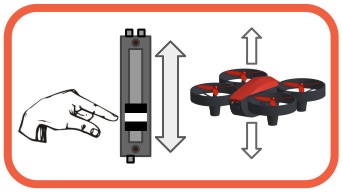
Power **ON** the **Drone**

Place the **Drone** on the floor





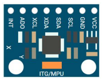




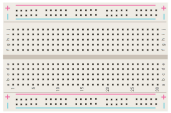
Not connecting…
Power cycle the drone by
turning it OFF and ON

Pilot Controller Assembly

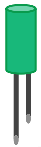
Parts we need



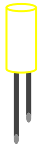
GY-521



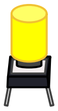
Small Breadboard



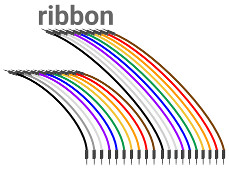
Green LED



White LED



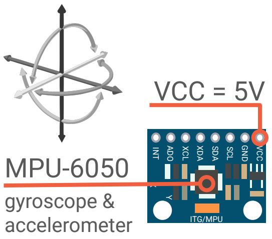
Yellow Button



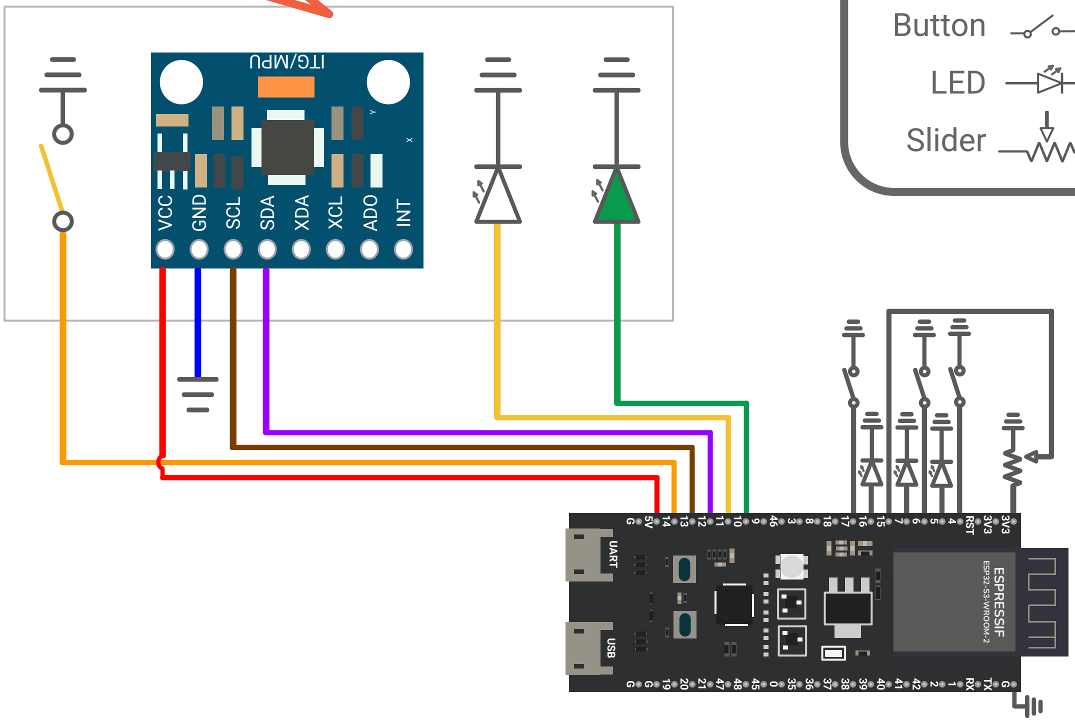
Wire

GY-521 Module

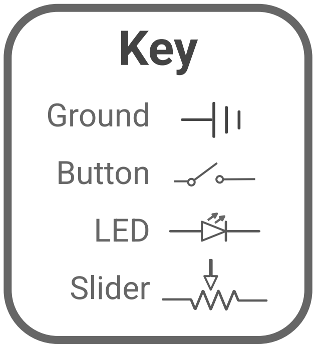
- Printed Circuit Board (PCB) with an MPU-6050 sensor
- Combines 3-axis gyroscope and accelerometer, measuring angular rates and linear accelerations
- Digital Motion Processor - DMP reduces main processor load
- Used in gaming, virtual reality, and robotics for motion tracking.



Use the **Wiring Diagram** to assemble the Pilot Controller on the small breadboard

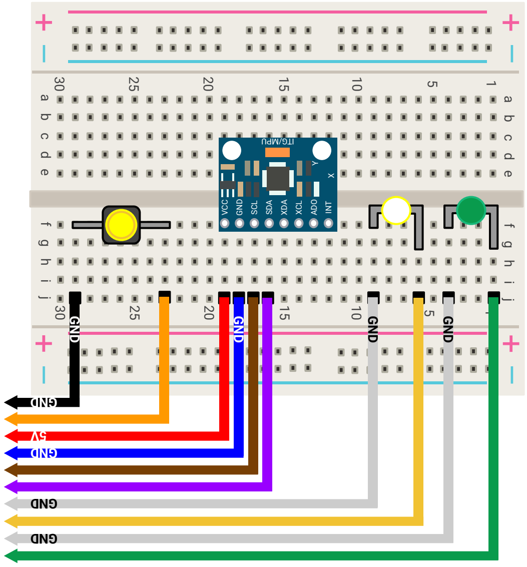


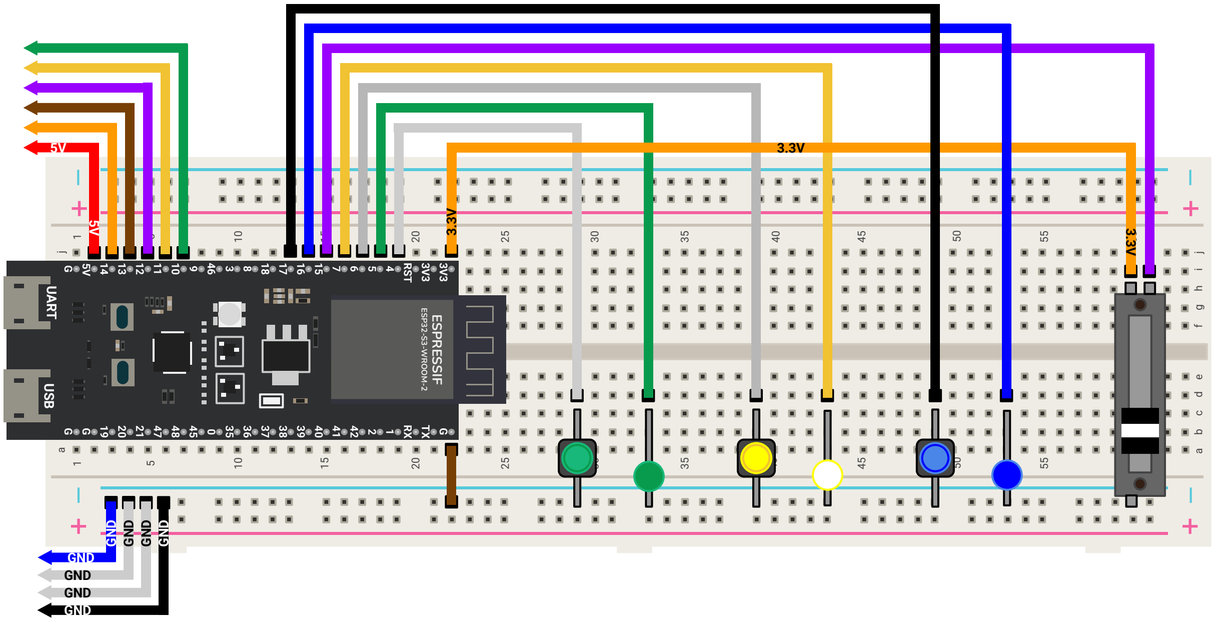
ITG/MPU
VCC
GND
SCL
SDA
XDA
XCL
ADO
INT
G 5V 14 13 12 11 10 9 46 3 8 18 17 16 15 7 6 5 4 RST 3V3 3V3
UART
USB
ESPRESSIF
ESP32-S3-WROOM-2
G G 19 20 21 47 48 45 0 35 36 37 38 39 40 41 42 2 1 RX TX G



Need a hint, go to the next page

Pilot Controller Assembly

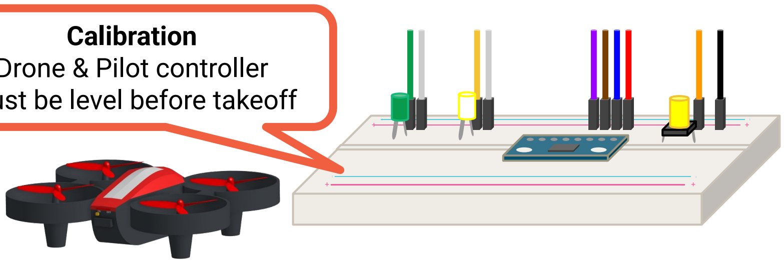




Pilot Controller

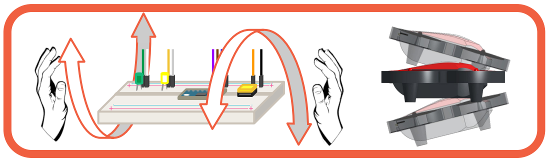
SAFETY GLASSES REQUIRED

Calibration
Drone & Pilot controller
must be level before takeoff



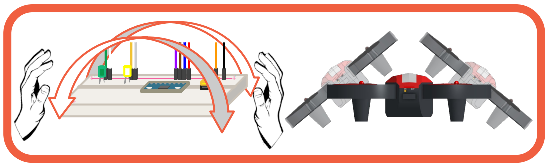
Pitch

Tilting up/down moves the drone forward/backward



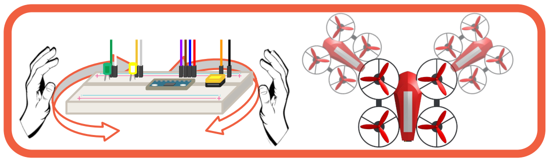
Roll

Tilting left/right moves the drone left/right



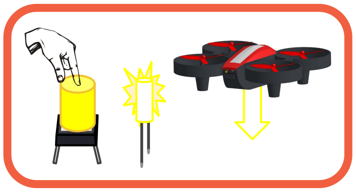
Yaw

Turning left/right changes the drone's direction



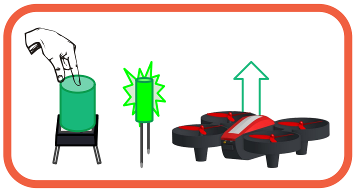
STOP

Immediately ends the flight

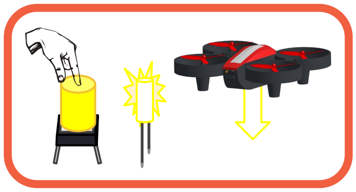


Ground Controller

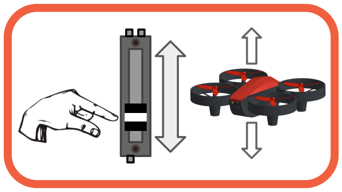
Takeoff



STOP



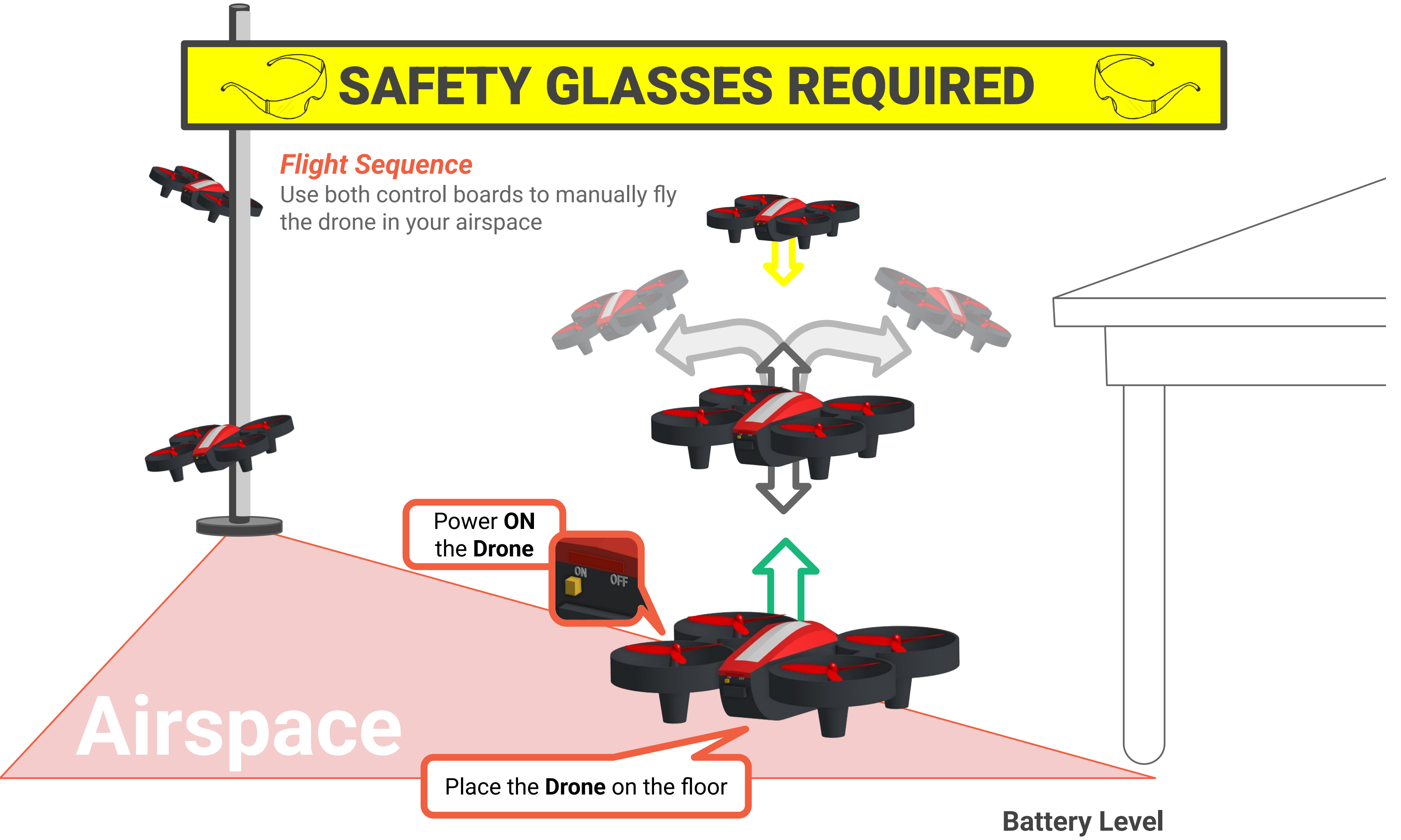
Altitude - Throttle



Teamwork

Both partners must collaborate for a controlled flight. One person operates the ground controller while the other operates the pilot controller.

Manual Control Flight



Flight Sequence

Use both control boards to manually fly the drone in your airspace

Power **ON** the **Drone**

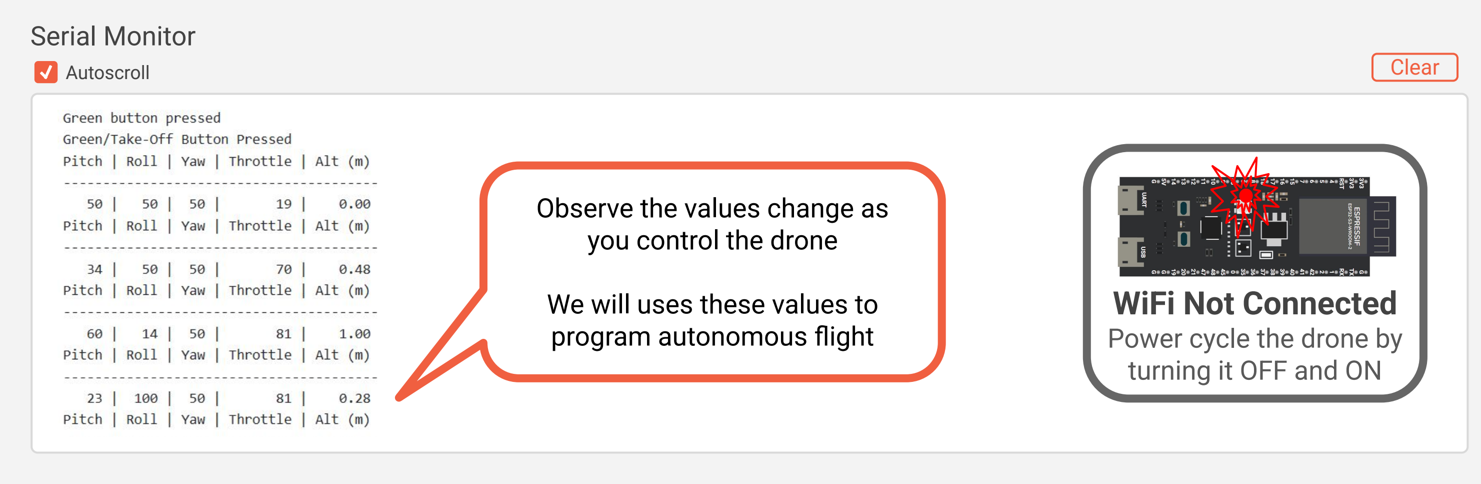
Place the **Drone** on the floor

Battery Level

Recharge your battery after a few minutes of flight time.

Challenge

Put your flying skills to the test by controlling your teams drone to fly from your table, fly out a predetermined distance and return to the table.



```
Green button pressed
Green/Take-Off Button Pressed
Pitch | Roll | Yaw | Throttle | Alt (m)
-----------------------------------------
   50 |   50 |  50 |       19 |    0.00
Pitch | Roll | Yaw | Throttle | Alt (m)
-----------------------------------------
   34 |   50 |  50 |       70 |    0.48
Pitch | Roll | Yaw | Throttle | Alt (m)
-----------------------------------------
   60 |   14 |  50 |       81 |    1.00
Pitch | Roll | Yaw | Throttle | Alt (m)
-----------------------------------------
   23 |  100 |  50 |       81 |    0.28
Pitch | Roll | Yaw | Throttle | Alt (m)
```

Observe the values change as you control the drone

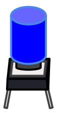
We will uses these values to program autonomous flight

WiFi Not Connected

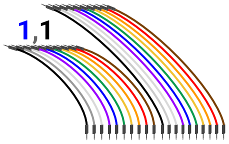
Power cycle the drone by turning it OFF and ON

Autonomous Button Assembly

Parts we need

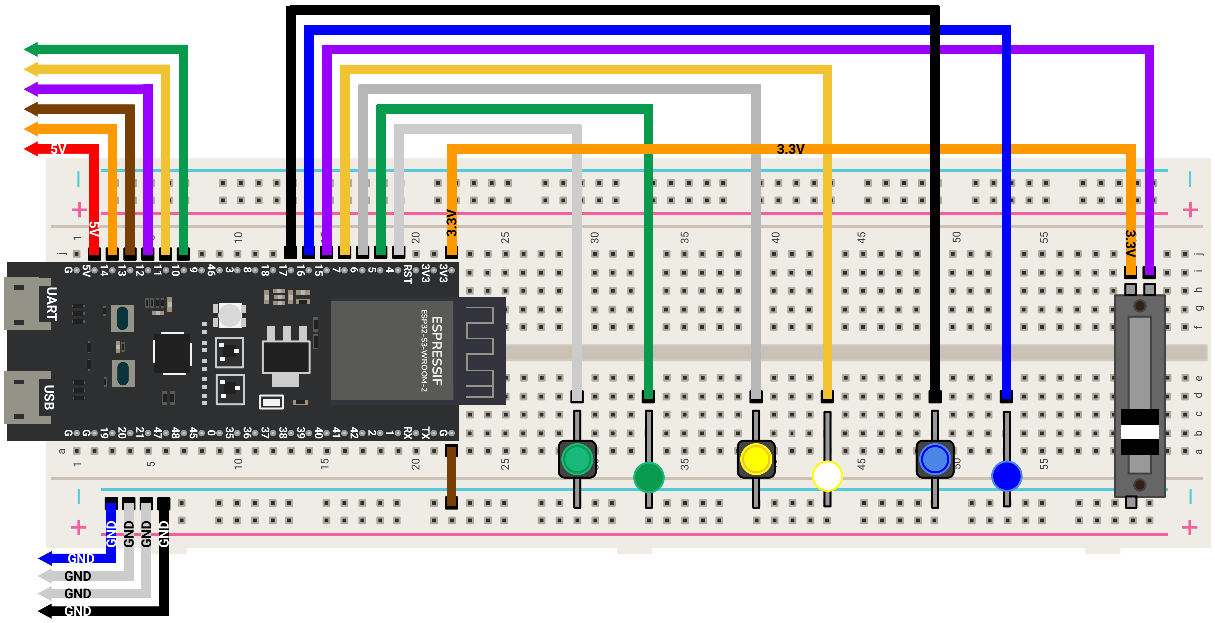


Blue Button

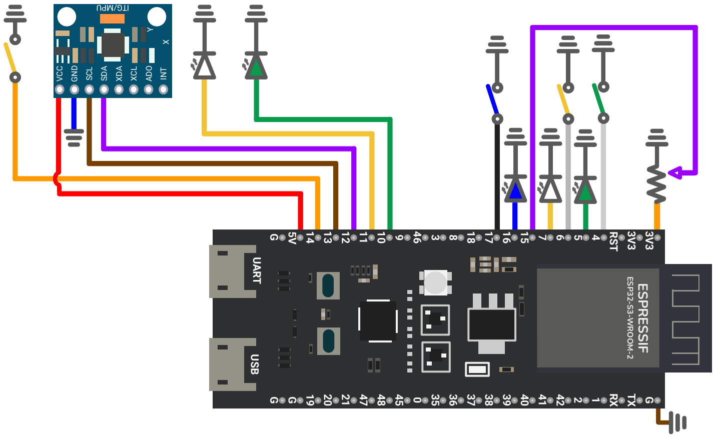


Wire

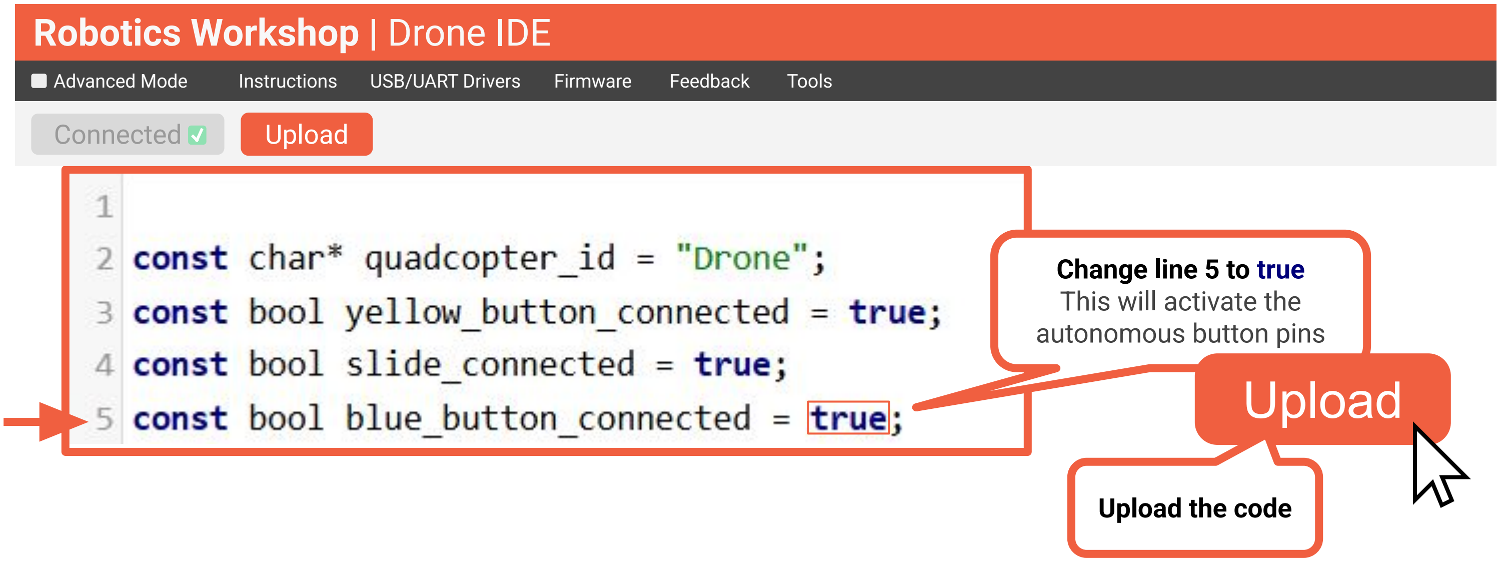
Use a **black** wire to connect **pin17** to the **blue** button



Wiring Diagram



Activate Autonomous Control



```
1
2 const char* quadcopter_id = "Drone";
3 const bool yellow_button_connected = true;
4 const bool slide_connected = true;
5 const bool blue_button_connected = true;
```

Change line 5 to true
This will activate the autonomous button pins

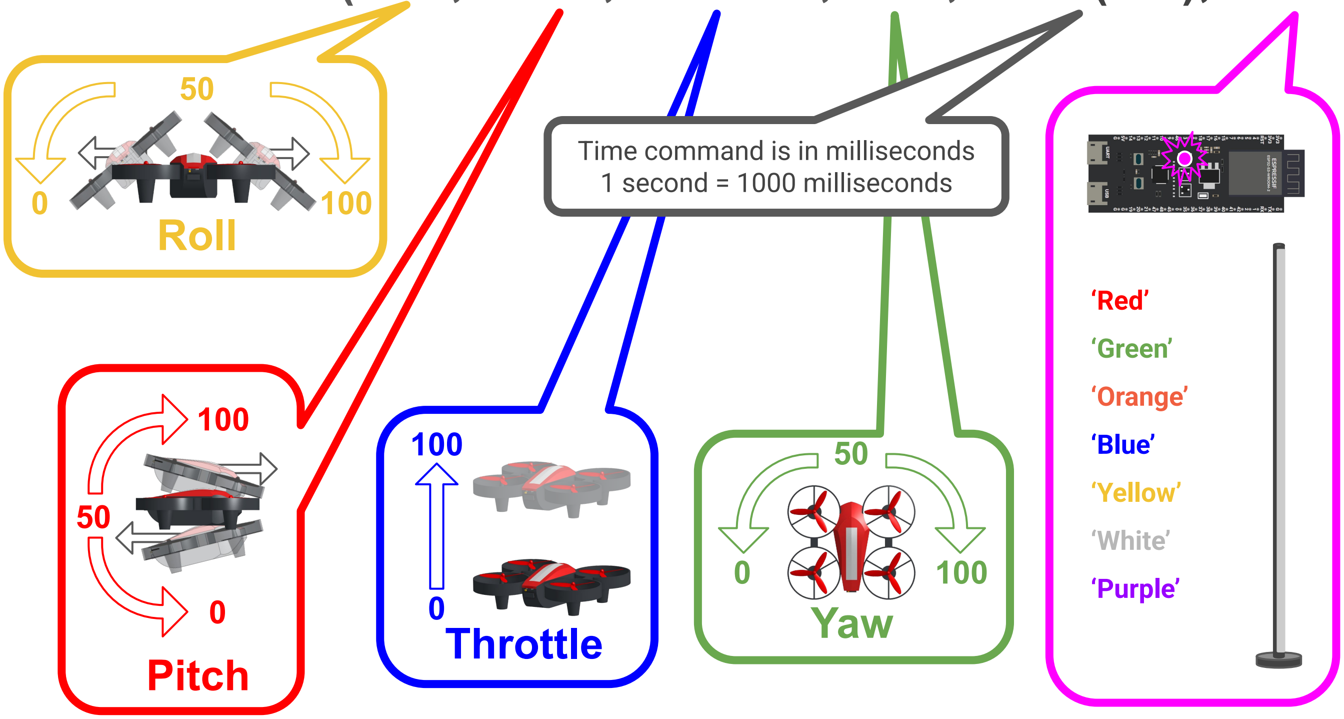
Upload

Upload the code

Autonomous Flight

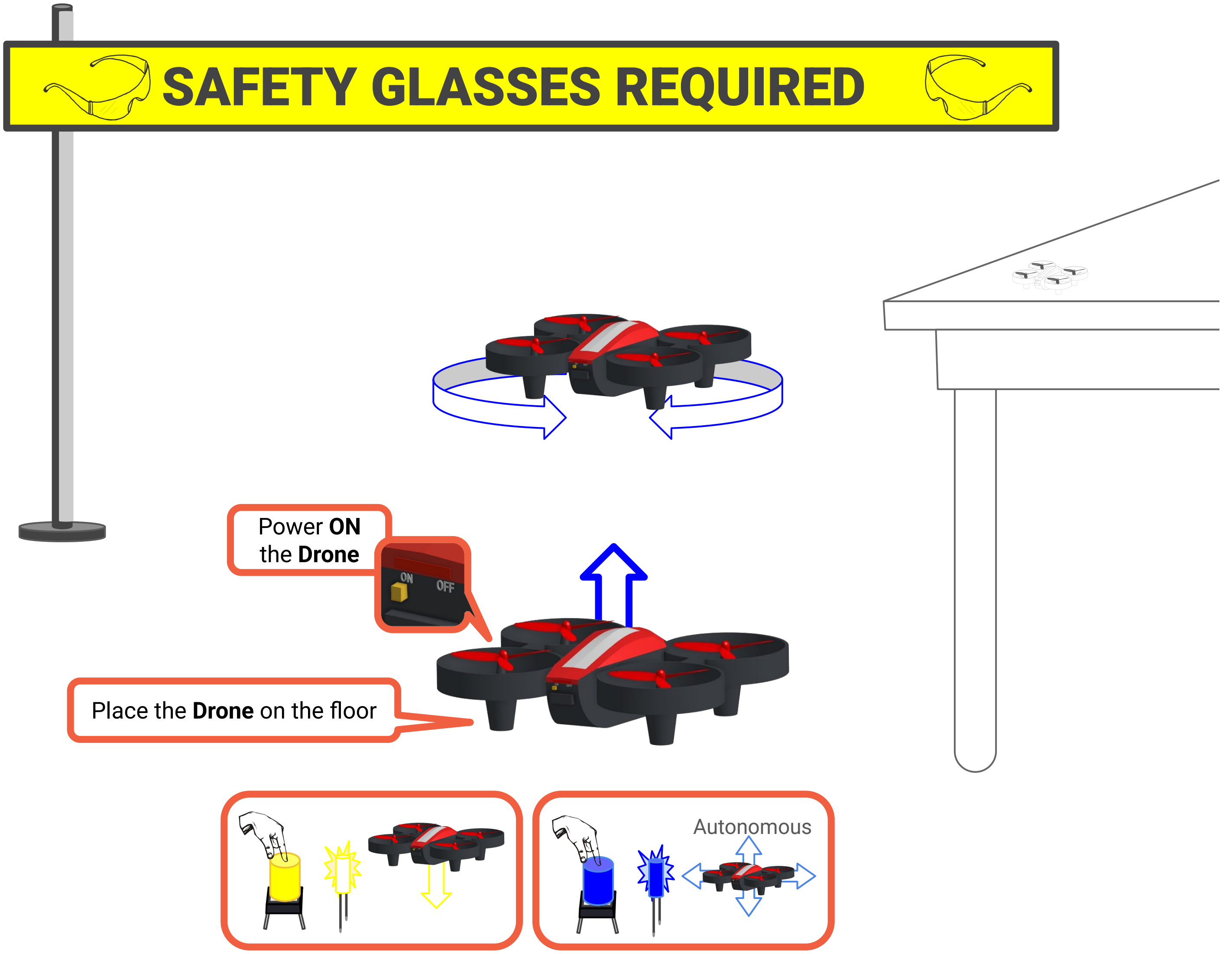
A hold command keeps the drone steady in one position for a set time. This can be a simple hover or a cool descending spiral. We'll use hold commands to create an autonomous flight, combining them to develop a complex flight pattern.

holdCommand(Roll, Pitch, Throttle, Yaw, Time(ms), Color);



Time command is in milliseconds
1 second = 1000 milliseconds

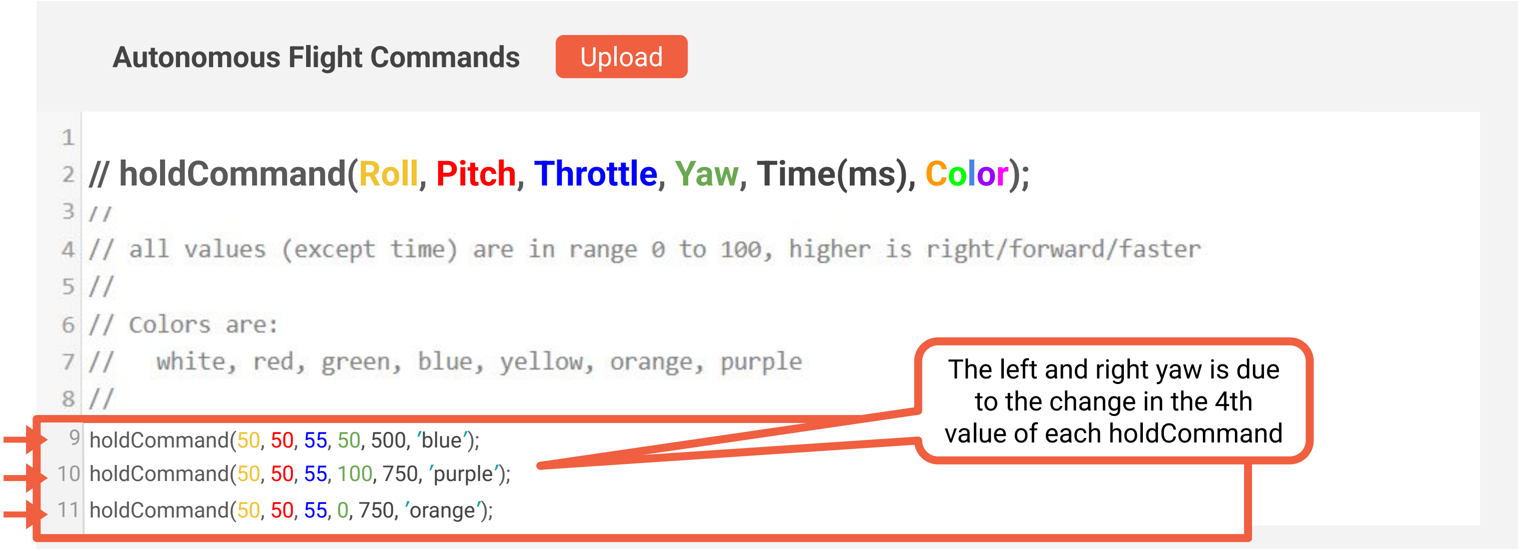
Autonomous Control Flight



Power **ON** the **Drone**

Place the **Drone** on the floor

Autonomous



```

// holdCommand(Roll, Pitch, Throttle, Yaw, Time(ms), Color);
//
// all values (except time) are in range 0 to 100, higher is right/forward/faster
//
// Colors are:
//   white, red, green, blue, yellow, orange, purple
//
holdCommand(50, 50, 55, 50, 500, ’blue’);
holdCommand(50, 50, 55, 100, 750, ’purple’);
holdCommand(50, 50, 55, 0, 750, ’orange’);
```

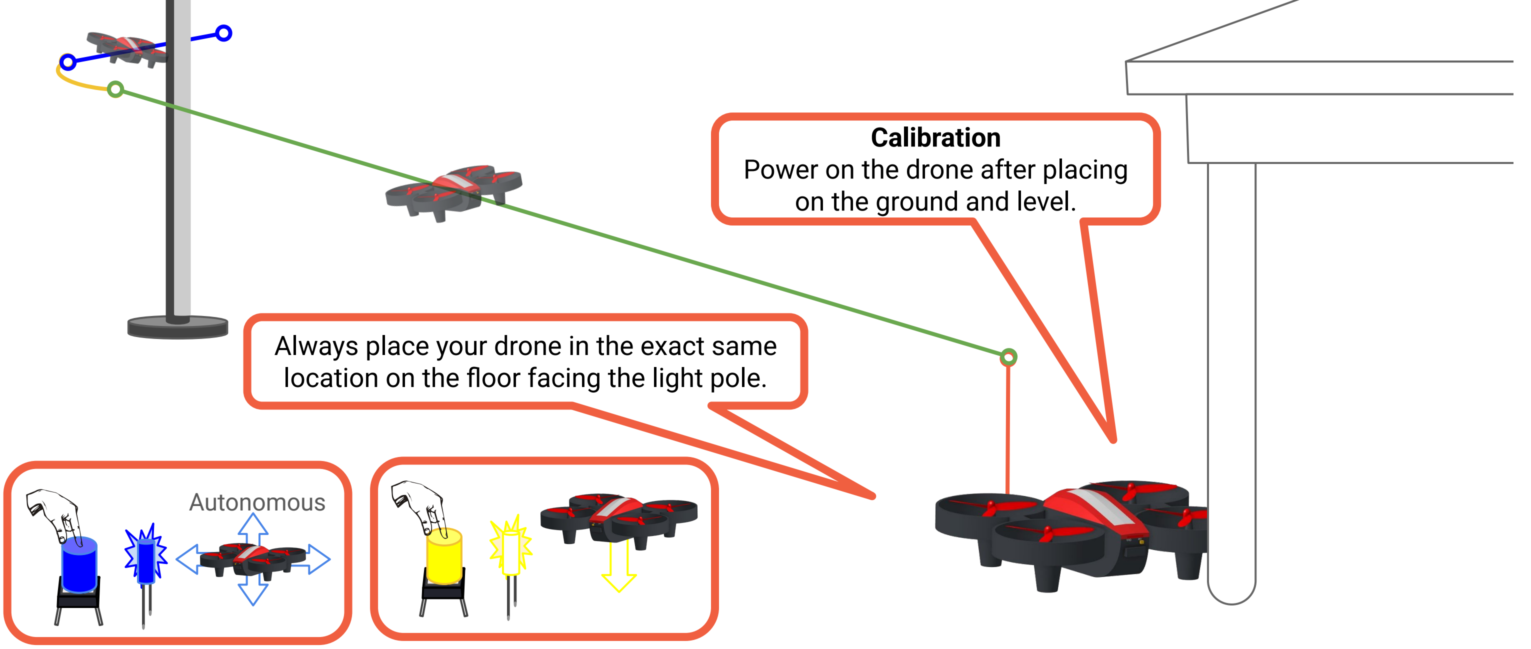
The left and right yaw is due to the change in the 4th value of each holdCommand

Autonomous Flight 1

SAFETY GLASSES REQUIRED

Autonomous Flight 1

Write a sequence of holdCommand() instructions to have your drone take off, fly outward, and make a right turn around a predetermined object.

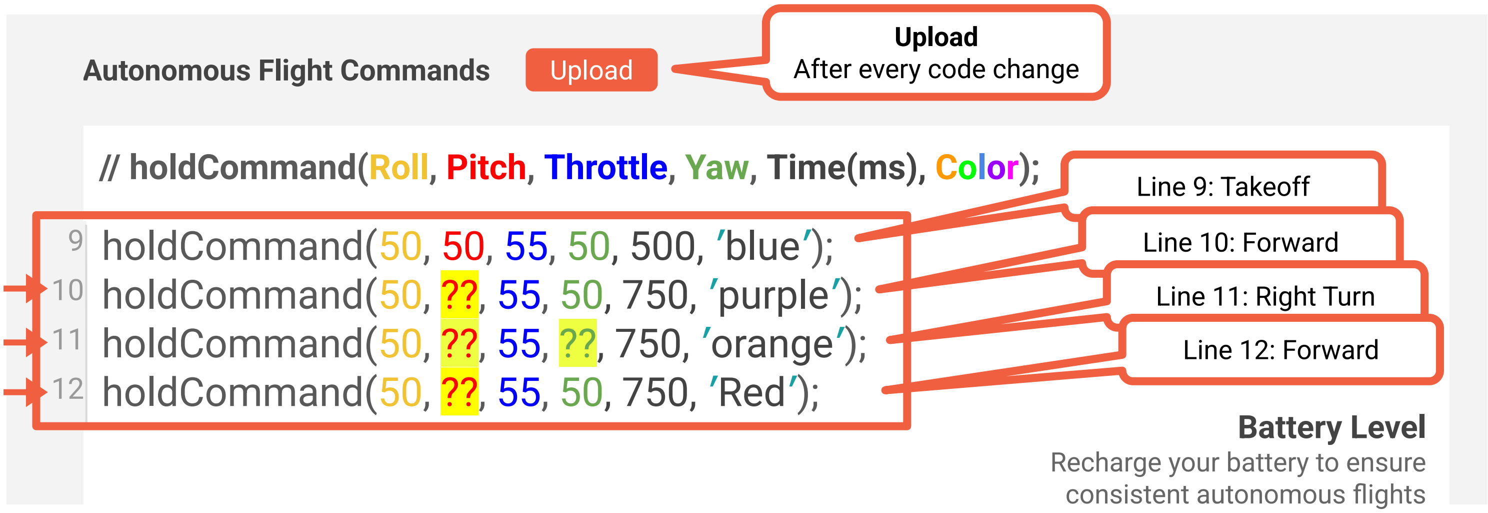


Calibration
Power on the drone after placing on the ground and level.

Always place your drone in the exact same location on the floor facing the light pole.

Tuning

- Begin with small adjustments, focusing on the first one or two hold command lines.
- Upload and test each change, expecting to crash many times before achieving a successful flight.
- Embrace the engineering process, which involves learning from each attempt, including inevitable crashes.



Autonomous Flight Commands

Upload
After every code change

```
// holdCommand(Roll, Pitch, Throttle, Yaw, Time(ms), Color);
9  holdCommand(50, 50, 55, 50, 500, 'blue');
10 holdCommand(50, ??, 55, 50, 750, 'purple');
11 holdCommand(50, ??, 55, ??, 750, 'orange');
12 holdCommand(50, ??, 55, 50, 750, 'Red');
```

Line 9: Takeoff

Line 10: Forward

Line 11: Right Turn

Line 12: Forward

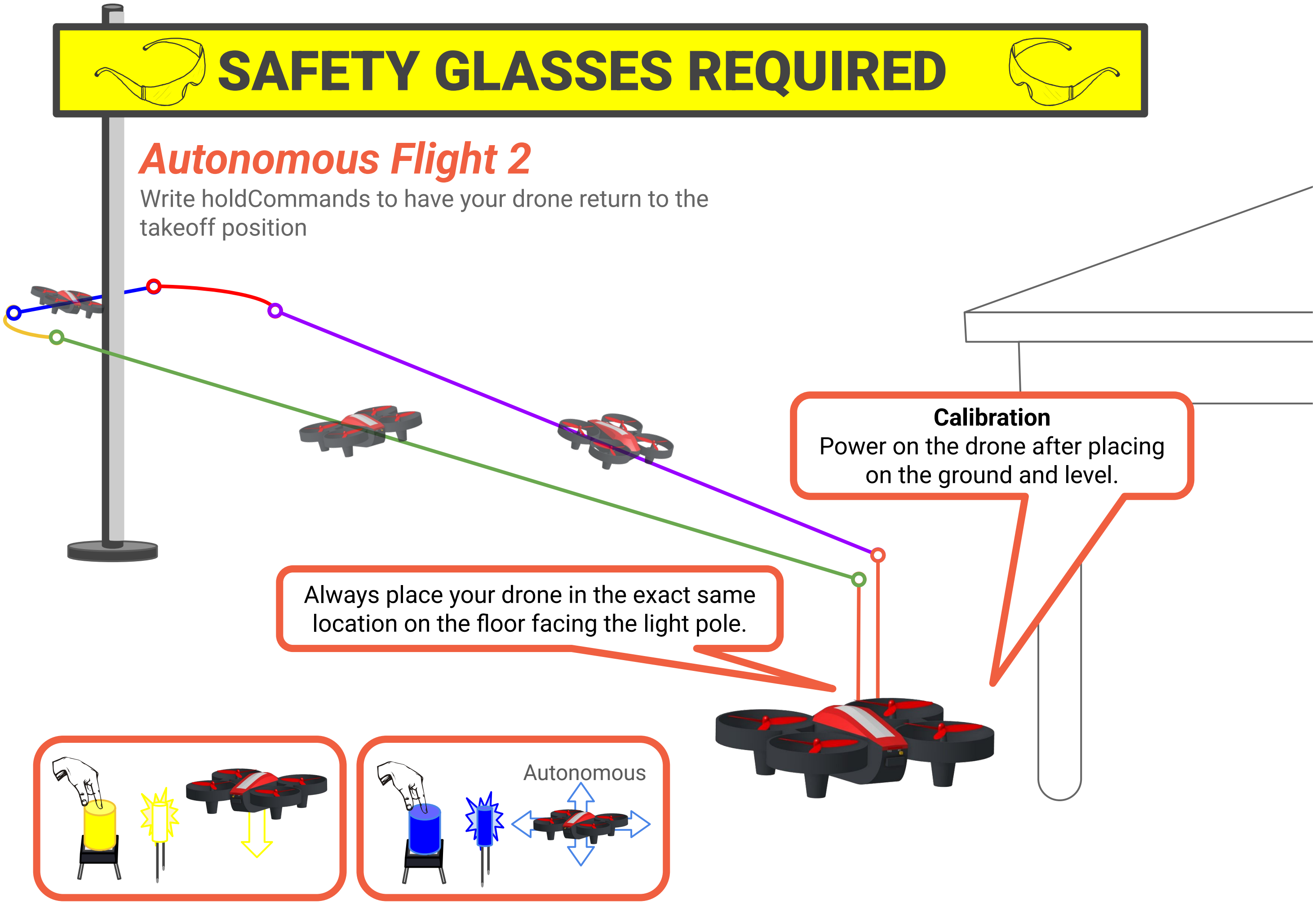
Battery Level
Recharge your battery to ensure consistent autonomous flights

Autonomous Flight 2

SAFETY GLASSES REQUIRED

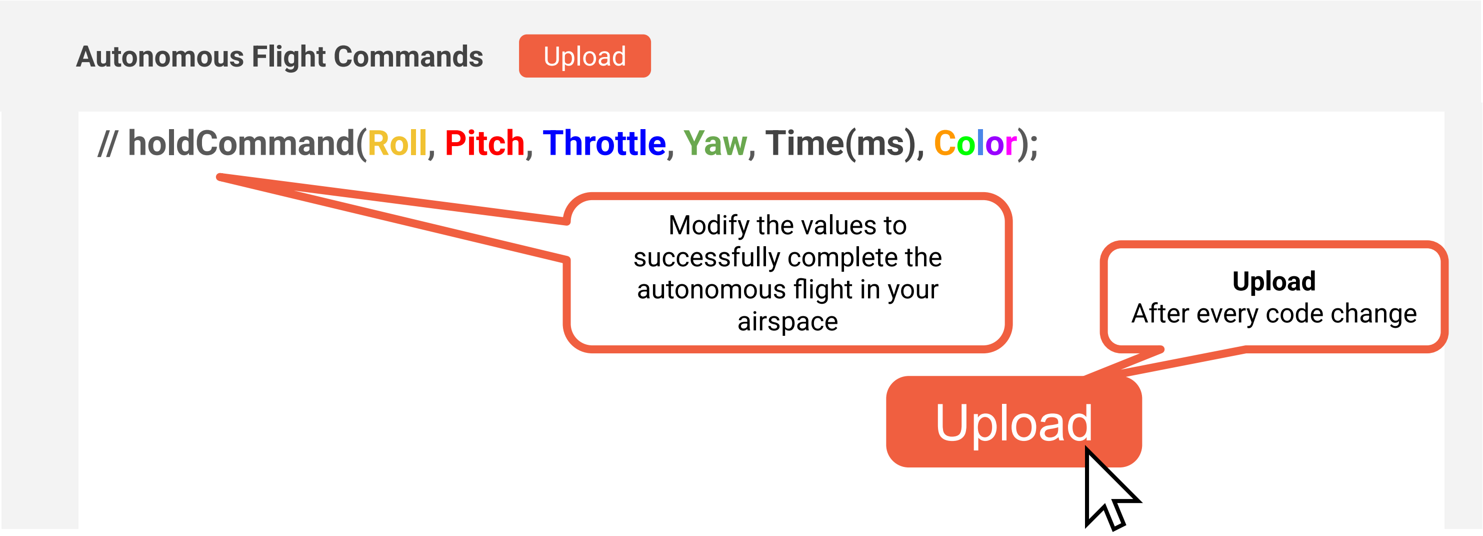
Autonomous Flight 2

Write holdCommands to have your drone return to the takeoff position



Calibration
Power on the drone after placing on the ground and level.

Always place your drone in the exact same location on the floor facing the light pole.



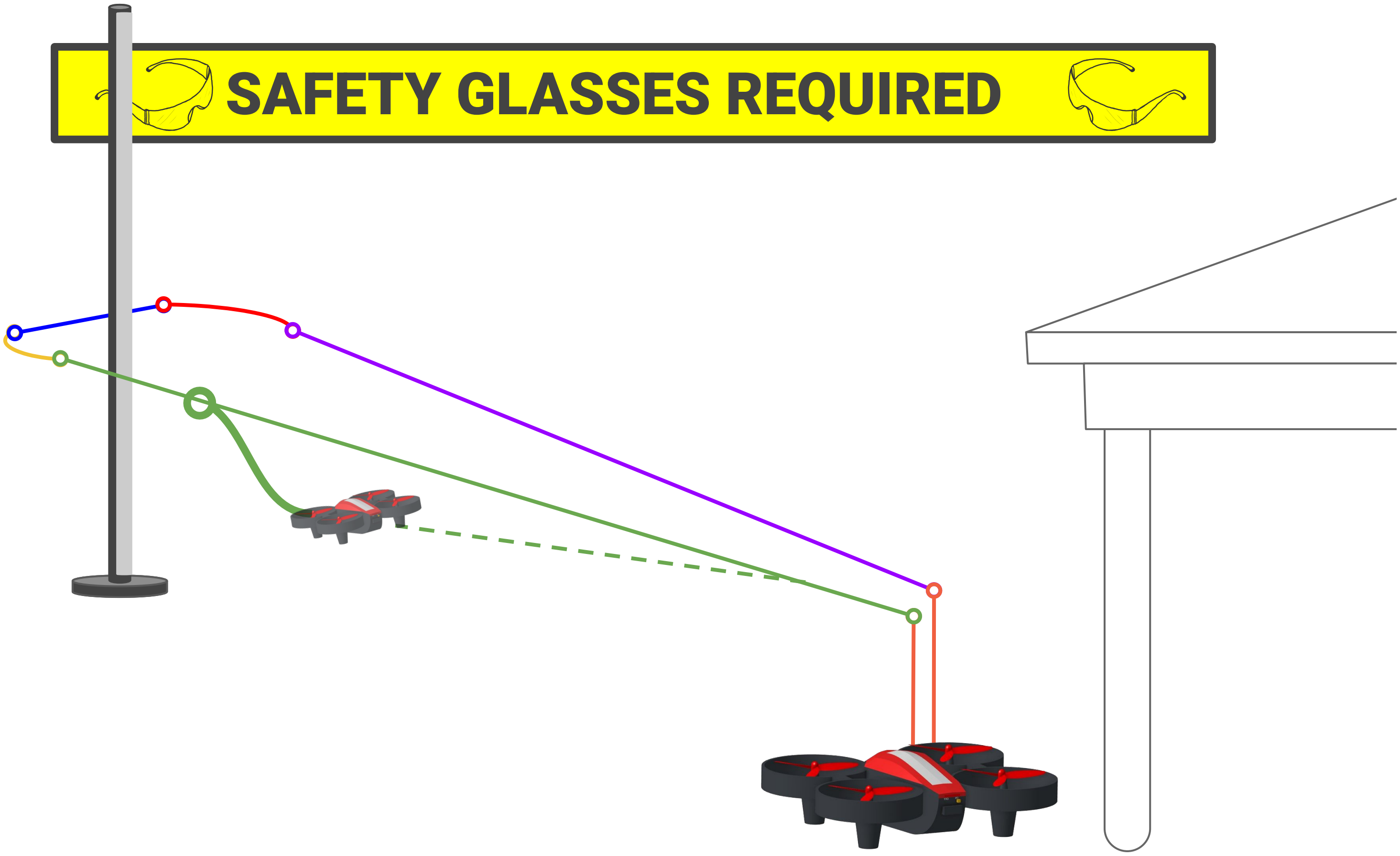
Autonomous Flight Commands Upload

```
// holdCommand(Roll, Pitch, Throttle, Yaw, Time(ms), Color);
```

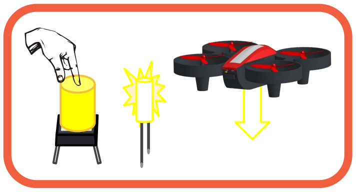
Modify the values to successfully complete the autonomous flight in your airspace

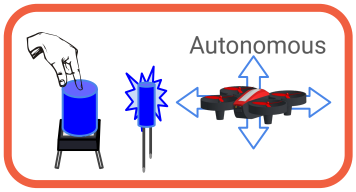
Upload
After every code change

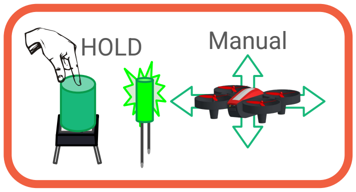
Pilot Override



Pause and Override Autonomous Flight
Hold down the green button to pause and override the autonomous flight.
Release the green button to resume the autonomous flight.
Be prepared to operate both the pilot and ground controller.



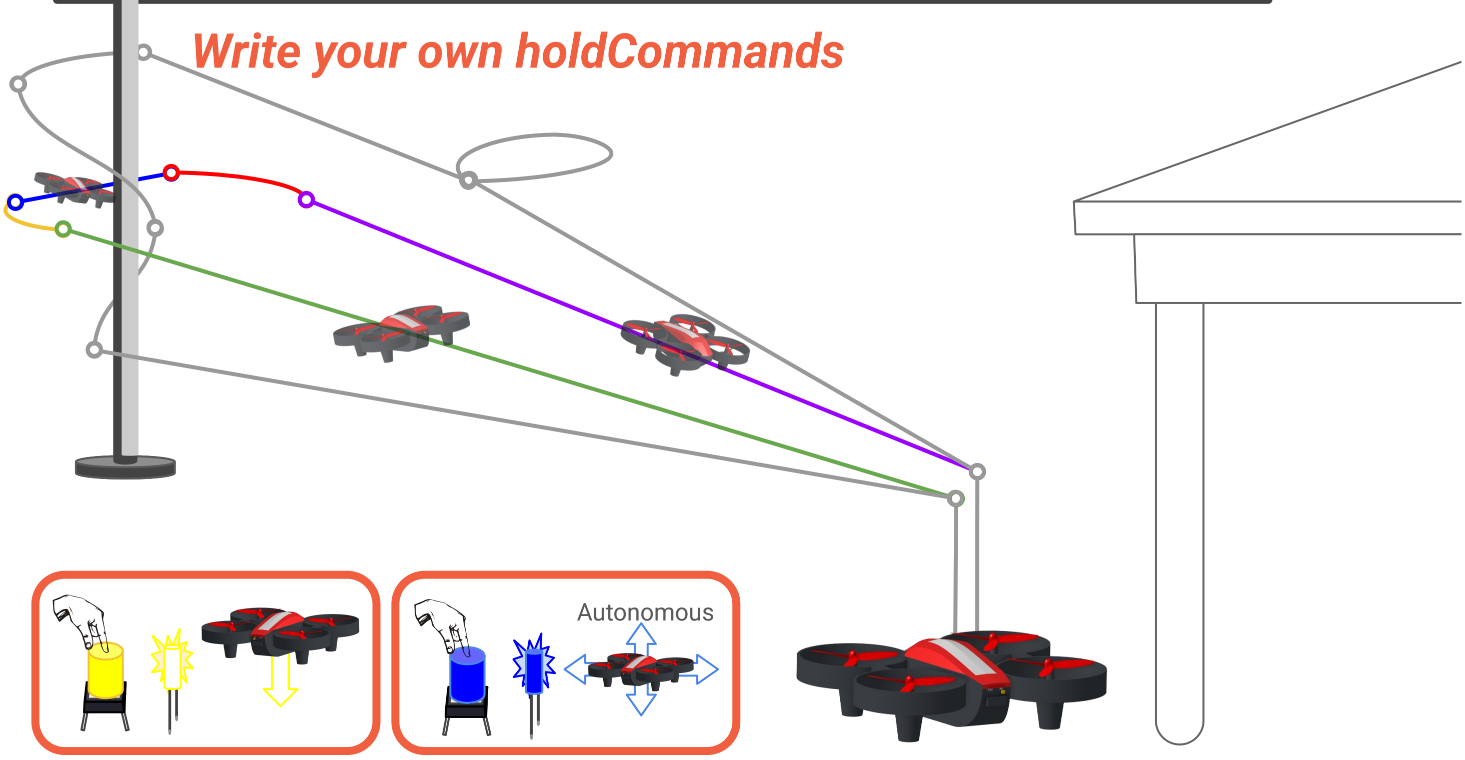


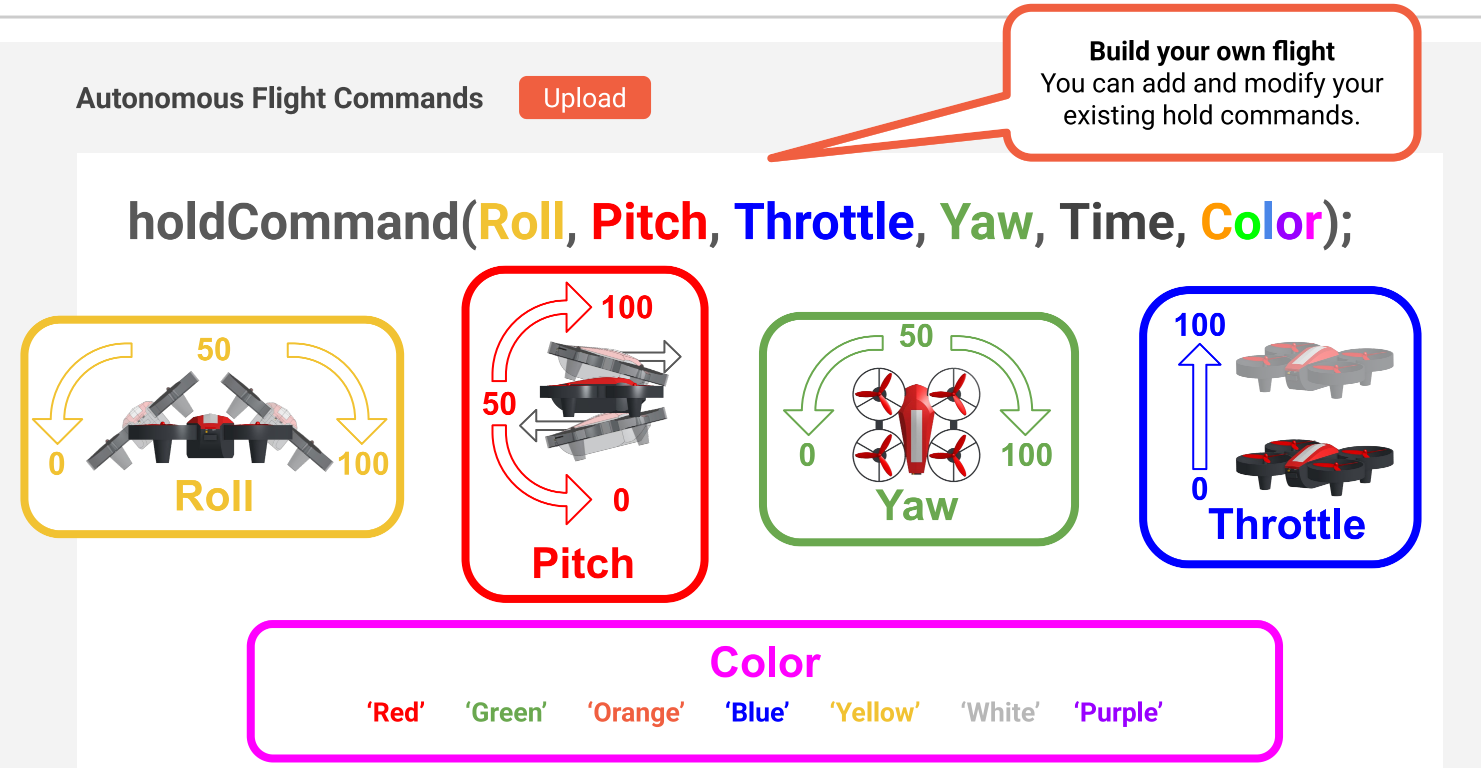


Custom Mission

SAFETY GLASSES REQUIRED

Write your own holdCommands





Autonomous Flight Commands

Build your own flight
You can add and modify your existing hold commands.

```
holdCommand(Roll, Pitch, Throttle, Yaw, Time, Color);
```

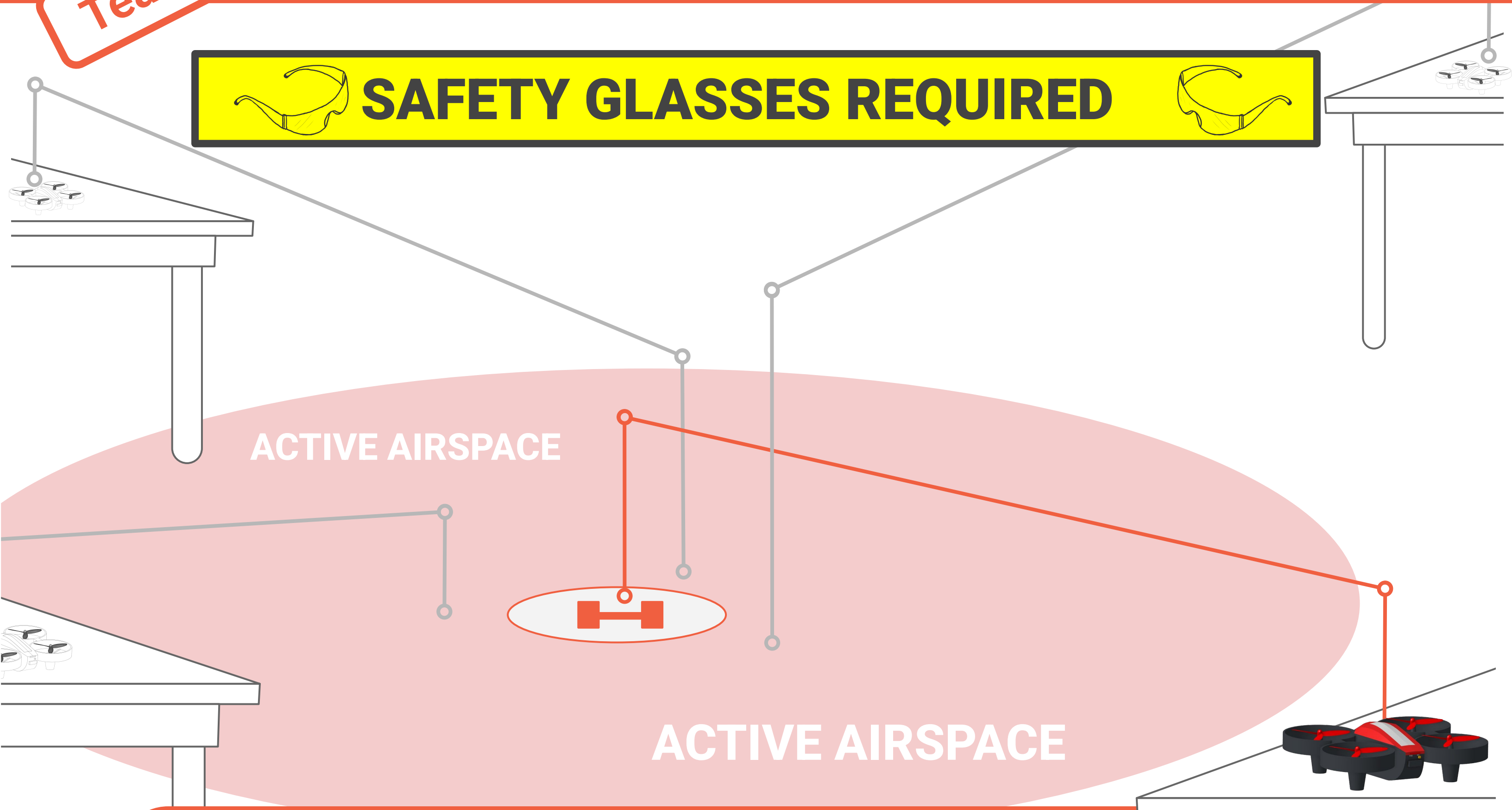
Color

'Red' 'Green' 'Orange' 'Blue' 'Yellow' 'White' 'Purple'

Team Challenge

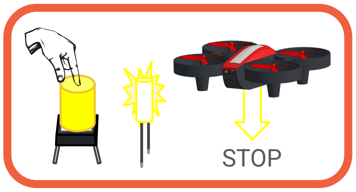
Final Approach

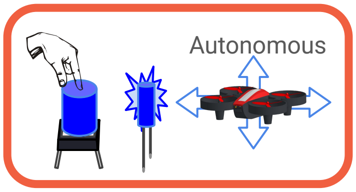
SAFETY GLASSES REQUIRED



Program your drone to take off from your group's starting point and land precisely on the designated landing zone.

- Always launch from the exact same spot, facing the same direction.
- Keep batteries topped-up. Low voltage changes how your drone responds
- Perform many short test flights, making only small adjustments to your hold-commands
- Record settings that work; build up to the full route step-by-step.



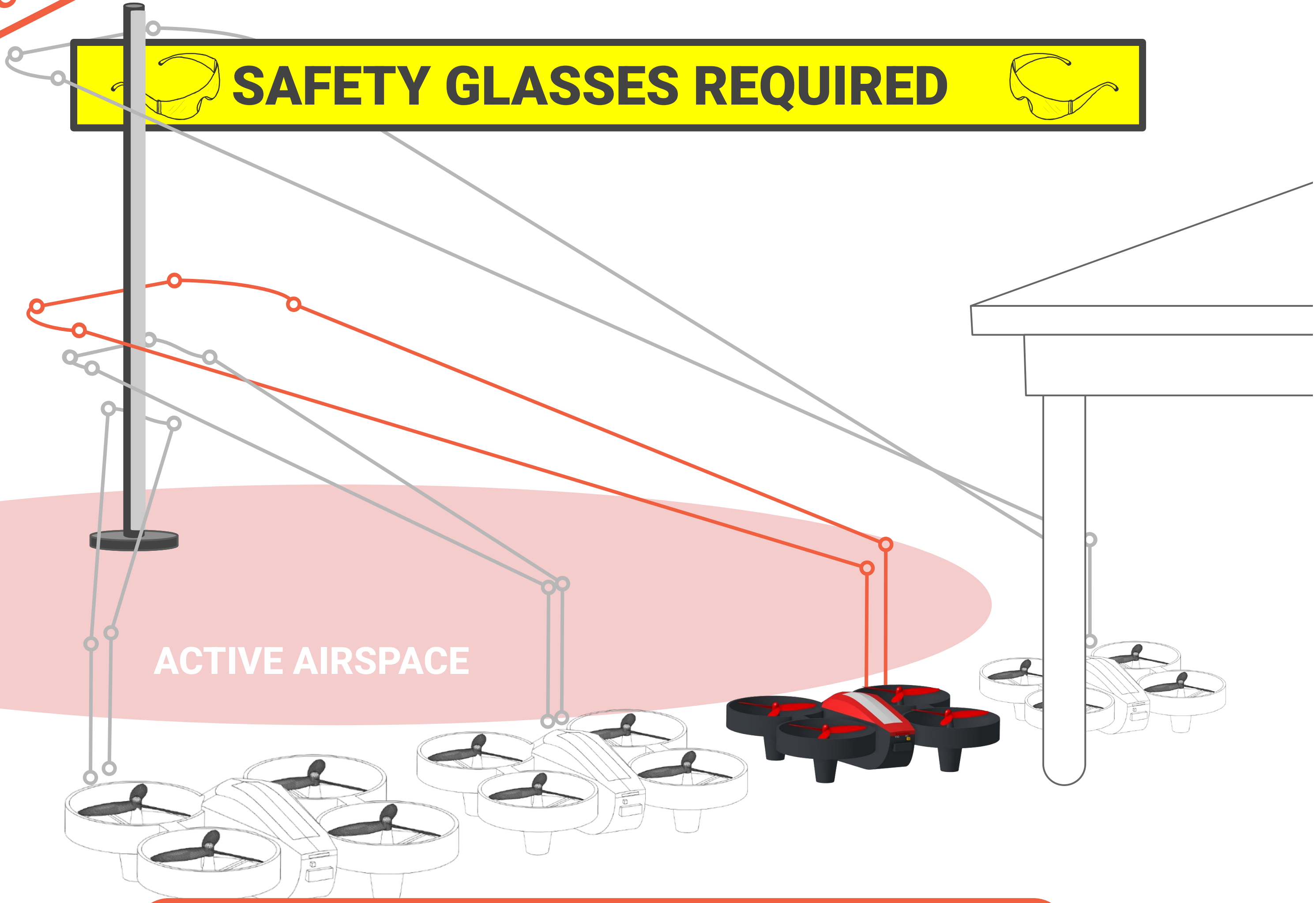


```
holdCommand(Roll, Pitch, Throttle, Yaw , Time, Color);
```

Team Challenge

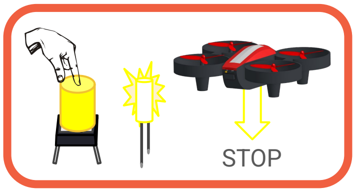
Grand Prix

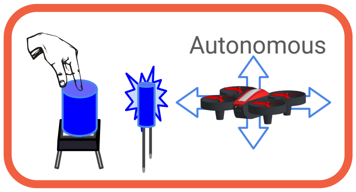
SAFETY GLASSES REQUIRED

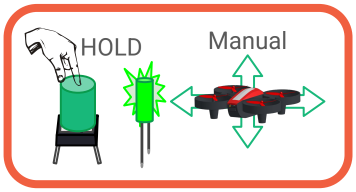


Program your drone to complete the Out-and-Back course

- Hold down the green button to pause and override the autonomous flight.
- Release the green button to resume the autonomous flight.
- Be prepared to operate both the pilot and ground controller.



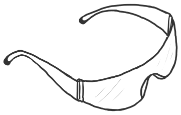




```
holdCommand(Roll, Pitch, Throttle, Yaw , Time, Color);
```

Checklist

1 SHUTDOWN



Autonomous Flight Control ……COMPLETE

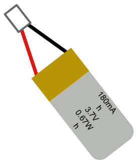
SAFETY GLASSES………………..…..OFF

2 COMPUTER

Applications……………………….……..CLOSE ALL

Laptop Power…….…………………..…..OFF

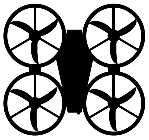
3 DRONE



Battery from Drone…..……………...REMOVE

Return Batteries to Instructor….YES

4 PARTS

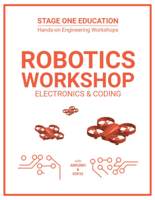


Return All Parts……………………….NEXT PAGE

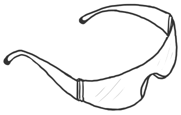
Parts Cleanup

On your desk

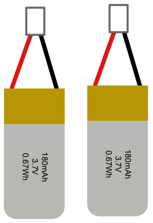
Laptop



Instructions



Safety Glasses



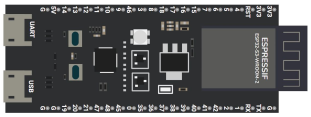
Drone Batteries

STAGE ONE EDUCATION
Hands-on Engineering Workshops

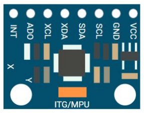
Check that all your parts are returned to the Electronic Box

Electronics Box

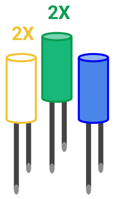
Drone



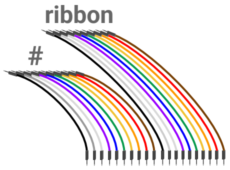
ESP32
Development Board



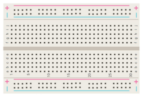
GY-521
Accelerometer



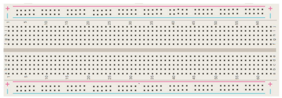
LED's



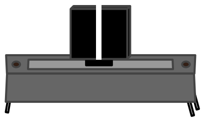
Wires



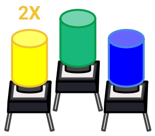
Small Breadboard



Large Breadboard



Slider
Variable Resistors



Buttons

Drone Battery
Charging Cable

USB to
Micro USB

Feedback Survey

YOU JUST COMPLETED THE

Robotics Workshop

Workshop Feedback Survey

feedback.stageoneeducation.com

Thank you for your participation

STAGE ONE EDUCATION

Hands-on Engineering Workshops