codrone EDU codrone EDU

codrone EDU

Sensor-Powered Missions

with Blockly and Python

Aerial Drone Program Summit 2025

BLOCKLY for ROBOLINK

python for ROBOLINK

robolink.com

ROBOLINK

Agenda

- Welcome and Introduction
- Walkthrough of each sensor
  - CoDrone EDU Applications
  - Functions
  - Troubleshooting
- Programming review
- Interactive coding challenges
- Closing

1 Sensors Overview

What is a Sensor?

- A sensor is a device that responds to an input and produces an output
- Converts external data to something we can interpret, analyze, and possibly input to another system
- Examples in everyday life:
  - Thermometer in a thermostat
  - Tire pressure sensors
  - Photoreceptor in a camera
  - Humidity sensor for a greenhouse
  - Motion infrared sensors in security devices
  - Accelerometer in a video game controller

Accelerometer

For sensing acceleration

Gyroscope

For sensing rotation

Barometer

For sensing height and pressure

Front range

For sensing obstacles ahead

Bottom range

For sensing distance to the ground

Color

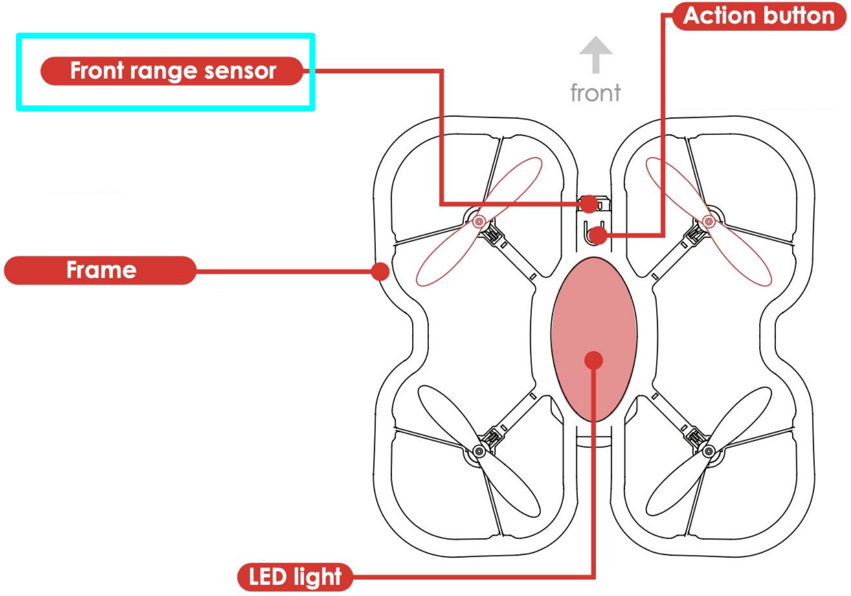
For sensing surface colors

Optical flow

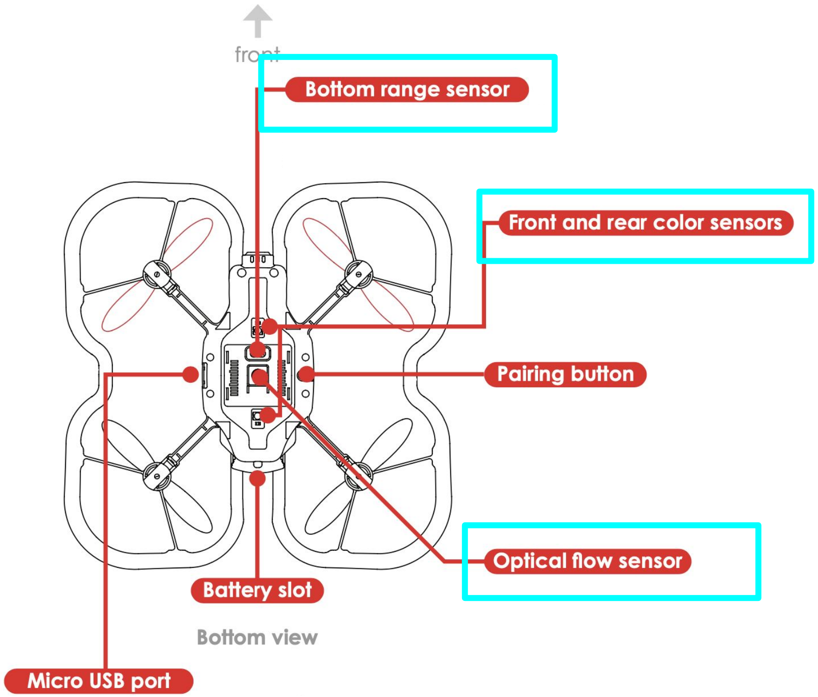
For sensing relative position

ROBOLINK

Where are the sensors on CoDrone EDU?



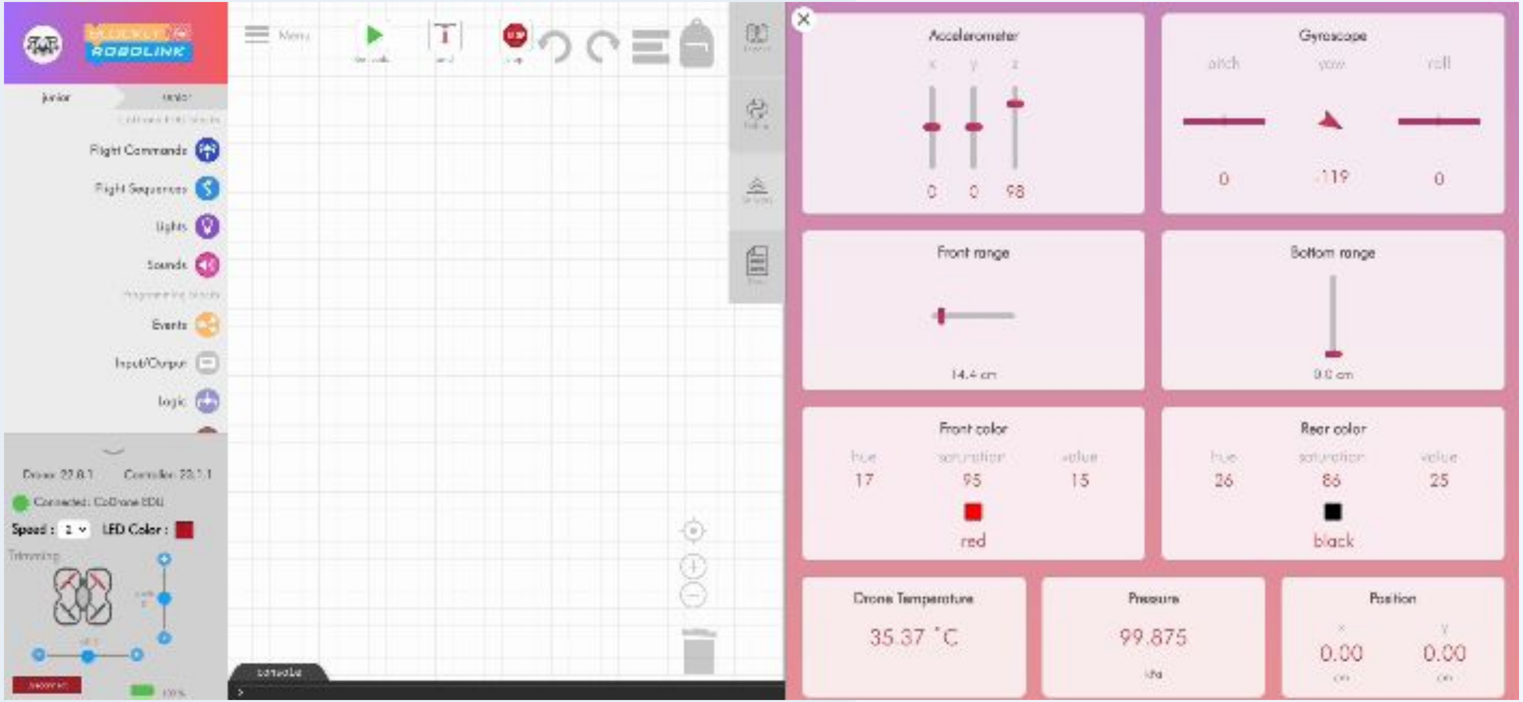
Top view



Bottom view

Sensor Dashboard

See live sensor feedback directly from the browser!



Drone: 22.8.1 Controller: 22.1.1

Connected: CoDrone EDU

Speed : 2 LED Color : 🟥

Accelerometer

x	y	z
0	0	98

Gyroscope

pitch	yaw	roll
0	119	0

Front range

14.4 cm

Bottom range

9.0 cm

Front color

hue	saturation	value
17	95	15

🟥 red

Rear color

hue	saturation	value
26	86	25

⬛ black

Drone Temperature

35.37 °C

Pressure

99.875

kPa

Position

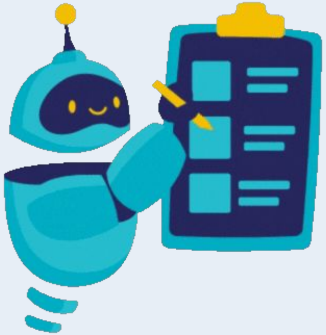
x	y
0.00	0.00
cm	cm

Available in both Blockly and Python!

2 Programming Platforms

Device Compatibility

If you are using a laptop from your school or organization, please check with IT that you have access to:



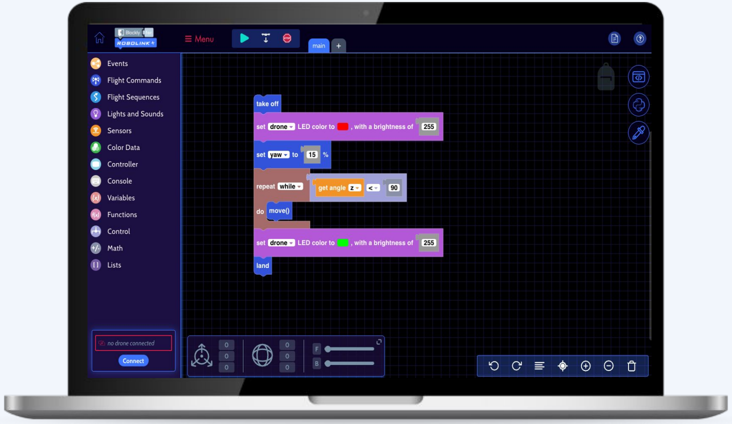
- Serial communication over USB ports
- USB-C adapter (if applicable)
- Robolink sites are whitelisted
- Optional: Ability to download and install Python/Pycharm (optional)

Not compatible with iPads, Tablets, or Cell Phones

Compatible with Chromebooks, Macs, and PCs

ROBOLINK

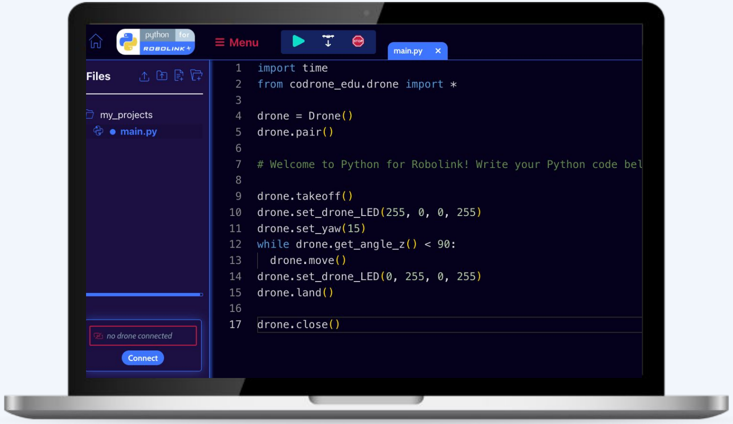
Languages



```
take off
set drone LED color to 🟥 , with a brightness of 255
set yaw to 15 %
repeat while get angle z < 90
do move()
set drone LED color to 🟩 , with a brightness of 255
land
```

Blockly for ROBOLINK

- **Mac, Windows, Chromebook**
- Visual programming in the browser
- Elementary/middle school or first-time coders
- No installation required



```
import time
from codrone_edu.drone import *

drone = Drone()
drone.pair()

# Welcome to Python for Robolink! Write your Python code bel

drone.takeoff()
drone.set_drone_LED(255, 0, 0, 255)
drone.set_yaw(15)
while drone.get_angle_z() < 90:
  drone.move()
drone.set_drone_LED(0, 255, 0, 255)
drone.land()

drone.close()
```

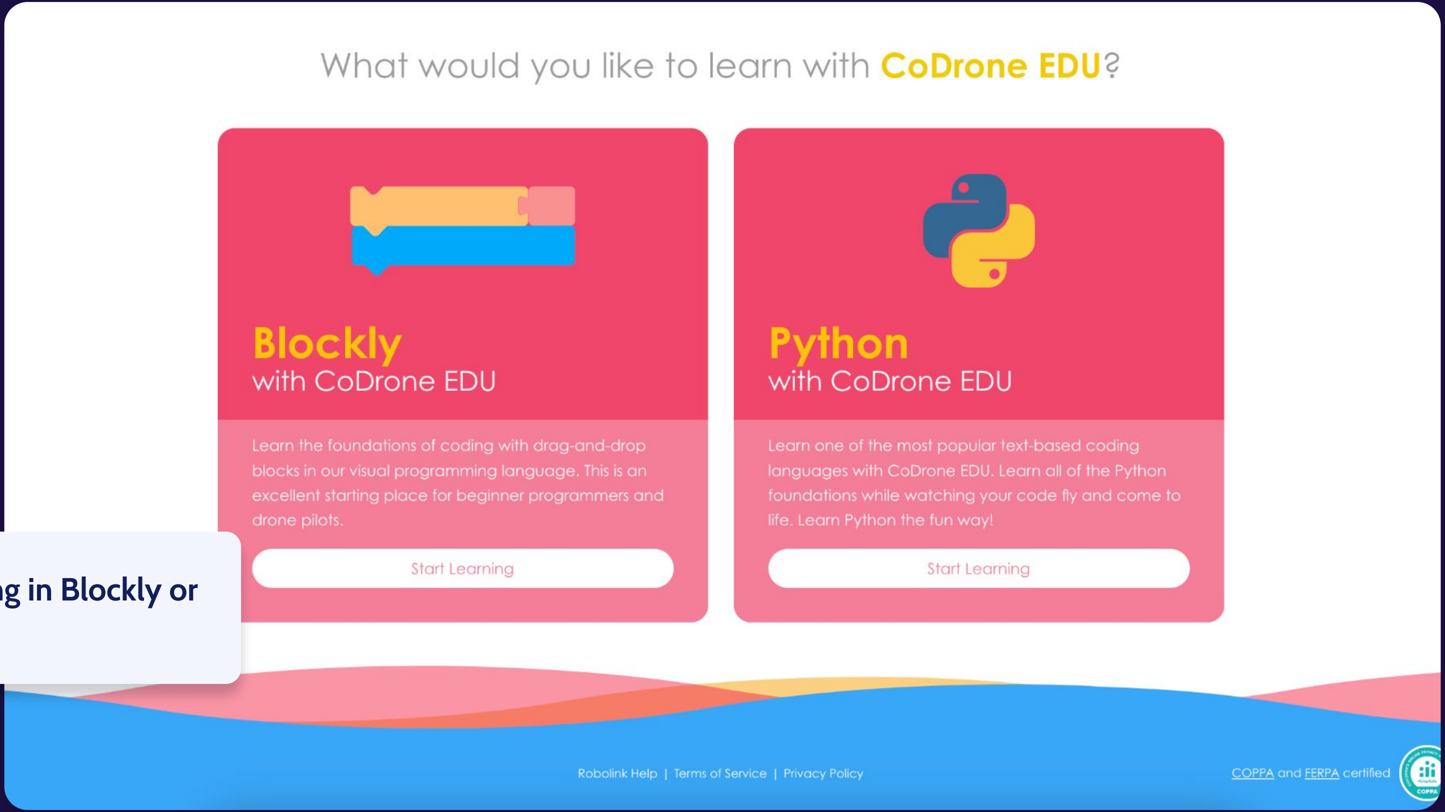
python for ROBOLINK

- **Mac, Windows, Chromebook**
- Text-based language in the browser
- Suitable for 6th grade and above
- No installation required or use your own IDE (Mac/Windows only)

ROBOLINK

Getting Started

learn.robolink.com/product/codrone-edu/



What would you like to learn with **CoDrone EDU**?

Blockly
with CoDrone EDU

Learn the foundations of coding with drag-and-drop blocks in our visual programming language. This is an excellent starting place for beginner programmers and drone pilots.

Start Learning

Python
with CoDrone EDU

Learn one of the most popular text-based coding languages with CoDrone EDU. Learn all of the Python foundations while watching your code fly and come to life. Learn Python the fun way!

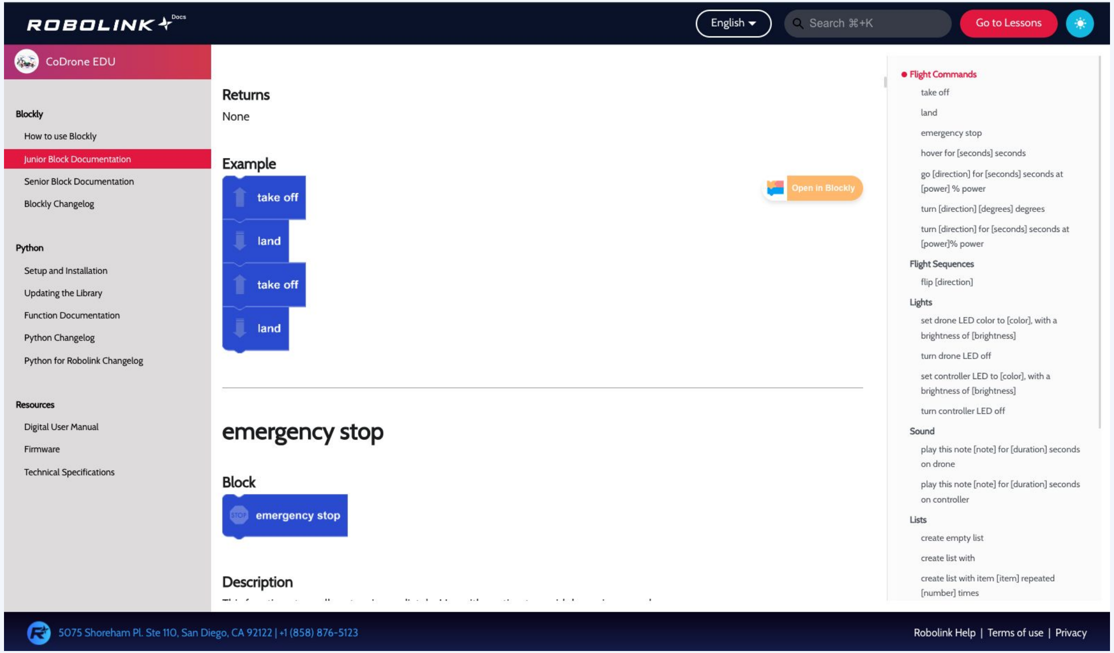
Start Learning

Robolink Help | Terms of Service | Privacy Policy
COPPA and FERPA certified

Start learning in Blockly or Python!

Documentation site

- Find resources on “How to use Blockly” or “How to use *Python for Robolink*”
- See function documentation on both Blockly and Python
- View version changelogs and release notes
- Find the user manuals, firmware information, and technical specifications
- Open examples directly from the documentation site!

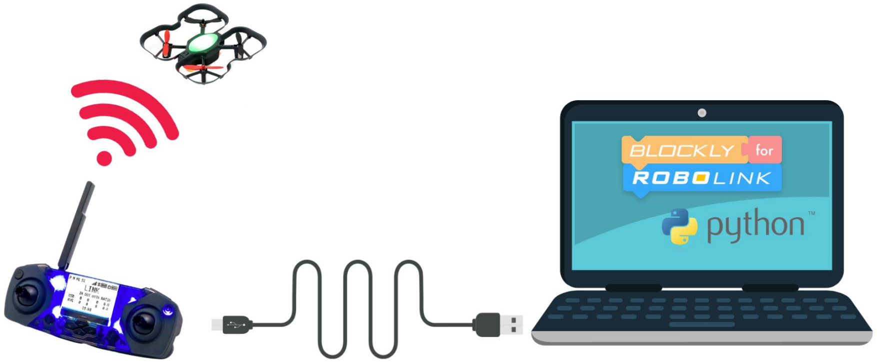

ROBOLINK Docs
English
Search ⌘+K
Go to Lessons
CoDrone EDU
Blockly
How to use Blockly
Junior Block Documentation
Senior Block Documentation
Blockly Changelog
Python
Setup and Installation
Updating the Library
Function Documentation
Python Changelog
Python for Robolink Changelog
Resources
Digital User Manual
Firmware
Technical Specifications
Returns
None
Example
Open in Blockly
take off
land
take off
land
emergency stop
Block
emergency stop
Description
Flight Commands
take off
land
emergency stop
hover for [seconds] seconds
go [direction] for [seconds] seconds at [power] % power
turn [direction] [degrees] degrees
turn [direction] for [seconds] seconds at [power]% power
Flight Sequences
flip [direction]
Lights
set drone LED color to [color], with a brightness of [brightness]
turn drone LED off
set controller LED to [color], with a brightness of [brightness]
turn controller LED off
Sound
play this note [note] for [duration] seconds on drone
play this note [note] for [duration] seconds on controller
Lists
create empty list
create list with
create list with item [item] repeated [number] times
5075 Shoreham Pl. Ste 110, San Diego, CA 92122 | +1 (858) 876-5123
Robolink Help | Terms of use | Privacy

Link: https://docs.robolink.com

Programming setup

Programming in Blockly or Python:

- The drone and controller must be paired together beforehand.
- **The controller isn't programmed.** The controller **transmits** the code.
- The remote controller must be connected to a computer for programming CoDrone EDU.
- The controller must be in the **LINK** state. **On the JROTC edition controller, you must do this manually by tapping the "Power" button after connecting and powering on.**



3 Range Sensors

Introduction

Range Sensors

- Range sensors in robotics are typically used to measure distances to obstacles
- Some common ways of detecting range:
  - infrared
  - ultrasonic
  - LiDAR
- Composed of emitters and receivers
- Time the duration it takes for the emitted signal to come back to the receiver

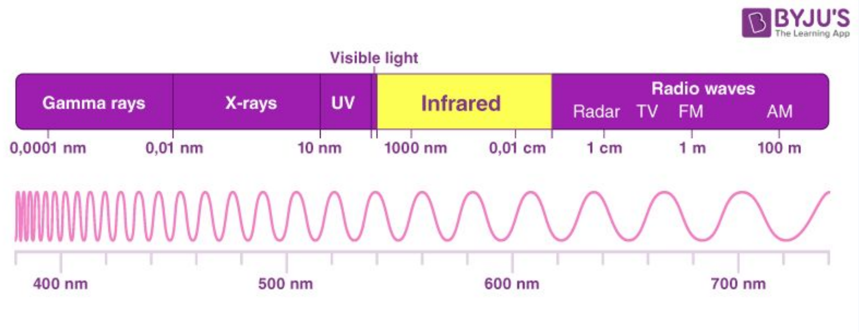
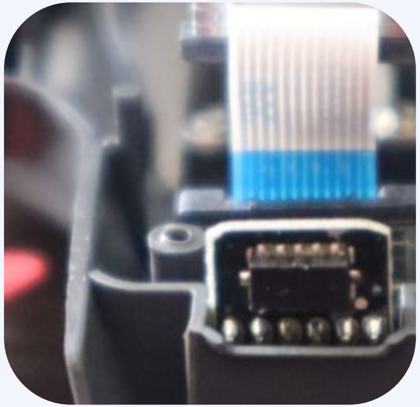


Image: BYJU

CoDrone EDU Usage

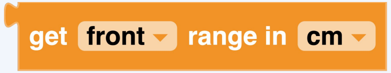
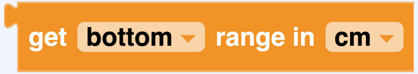
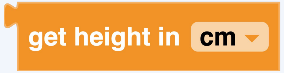
Range Sensors

- Bottom (and front) range sensors use **time-of-flight infrared laser sensors.**
- A transmitter emits an IR beam of light and times how long it takes for the light to travel back to the receiver. It uses the speed of light to estimate the distance between the sensor and an object.
- Sensor data is most accurate up to 1.5 meters

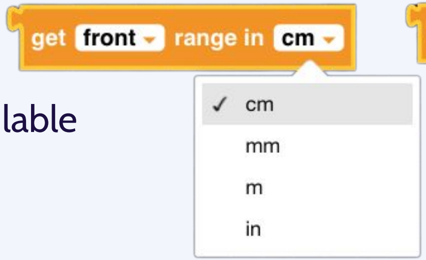


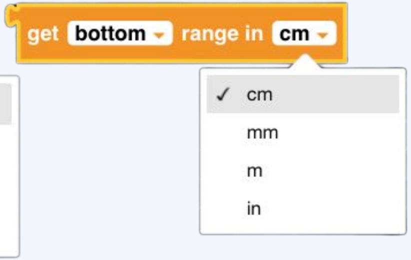
Function Description

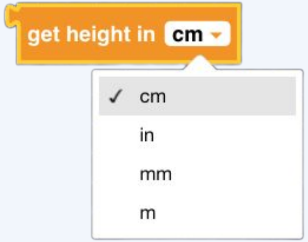
Range Sensors

Data	Range	Units	Block	Python
Front range	0~200	centimeters	 get front range in cm	`drone.get_front_range()`
Bottom range	0~200	centimeters	 get bottom range in cm	`drone.get_bottom_range()`
Height (same as bottom range)	0~200	centimeters	 get height in cm	`drone.get_height()`

! Other units are available







Troubleshooting

Surface materials

- Some materials do not reflect infrared light well
- Competition programming mat provides optimal flight performance

Infrared interference

- Sunlight or other infrared sources can interfere

Uneven surface

- Uneven surfaces may cause the laser to reflect in a way that it cannot be received by the sensor

Sensor timeout (missed measurement)

- A sensor timeout or error may return a value of over 999
- This appears as "out of range" in Sensor Dashboard

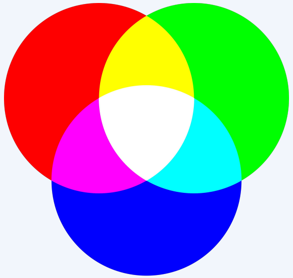
ROBOLINK

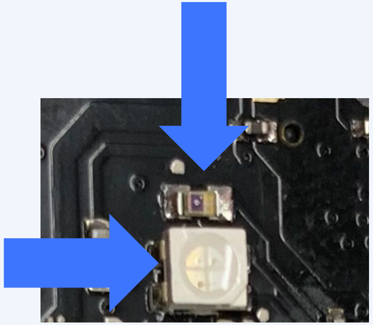
4 Color Sensors

Introduction

Color Sensors

- Drones typically detect colors with cameras
- Without cameras, you can use simpler light sensors to measure the amount of red, green, and blue (RGB) entering the sensor.
- Why RGB?
  - Mimics human vision
  - Cost effective
  - All colors can be represented by a combination of red, green, and blue.





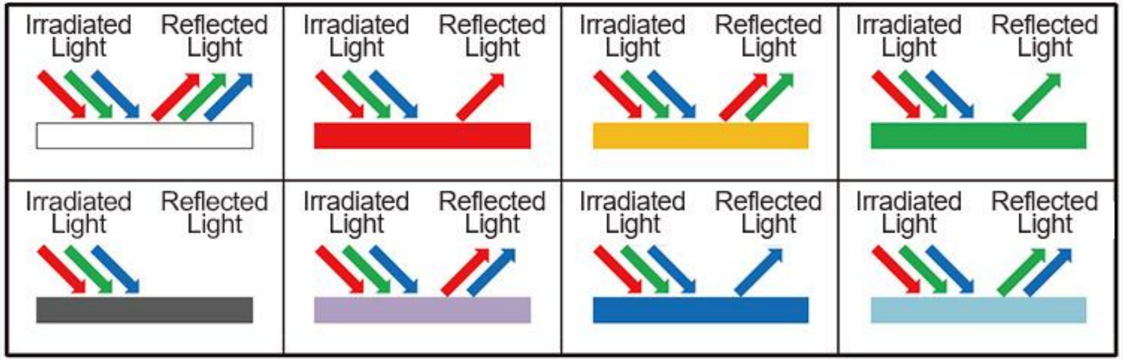
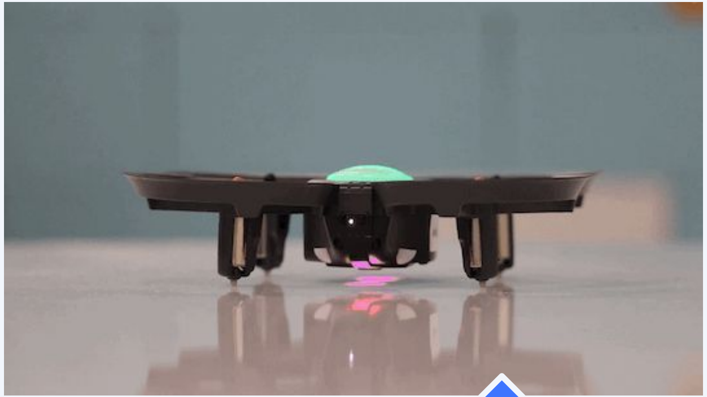


Image: ROHM Semiconductor

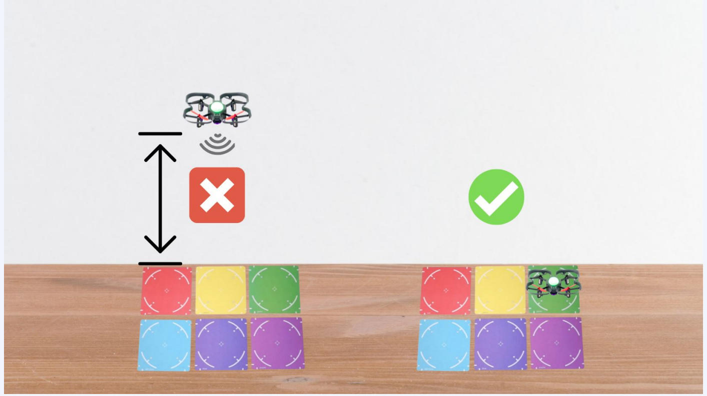
CoDrone EDU Usage

Color sensors

- CoDrone EDU has two (2) color sensors
- RGB (red, green, blue) light is emitted from an LED underneath the drone.
- The sensor detects the amount of red, green, and blue light reflected back from the surface.
- These RGB values correspond can be mapped to a particular color.
- The color sensor only works when the drone is on a surface.



The drone scans for red using the red LED, then blue, then green

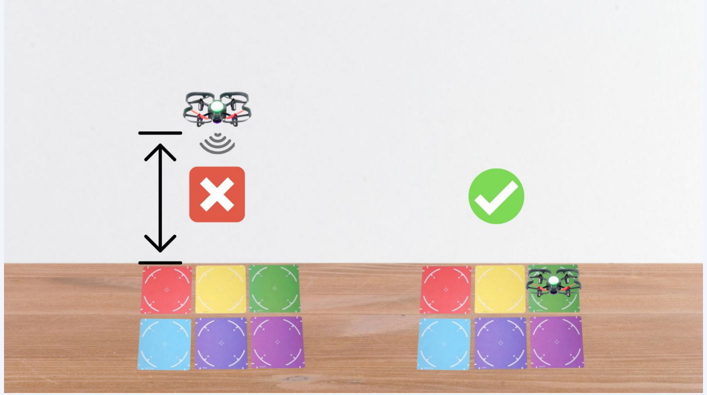


CoDrone EDU Usage

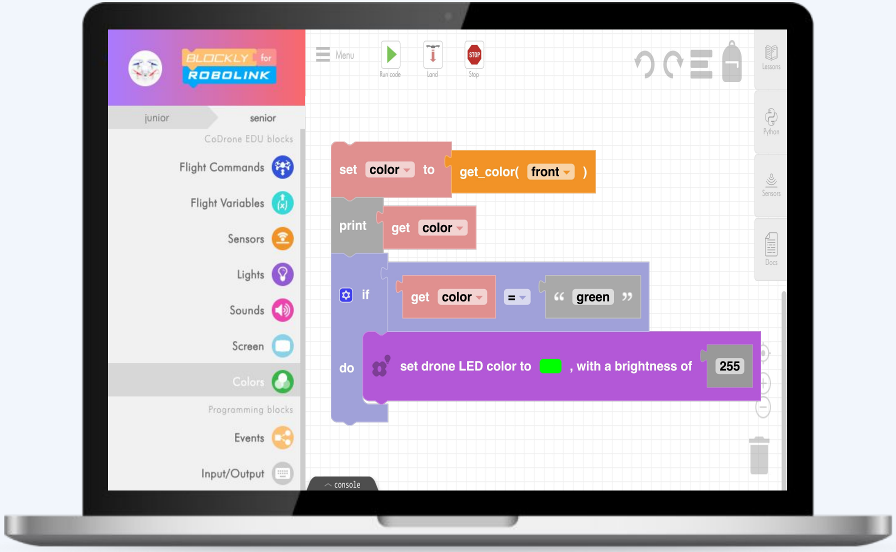
Color sensors

CoDrone EDU is pre-calibrated to the 8 color cards included in the box.

- white
- black
- red
- yellow
- green
- light blue
- blue
- purple



Lesson: 3.7 - Color Sensor



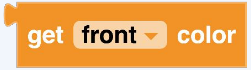
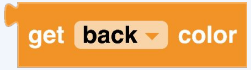
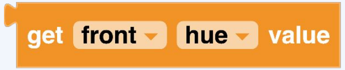
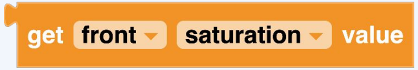
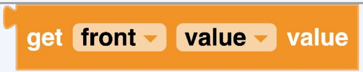
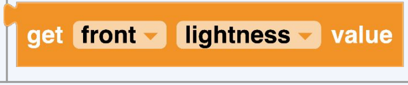
```
set color to get_color( front )
print get color
if get color = " green "
do set drone LED color to 🟩 , with a brightness of 255
```

Use the orange Sensor blocks to access the default values.

- This block will not detect “new colors”
- Designed to work with the **included** colors
- For **new colors**, add new color data (next slides)

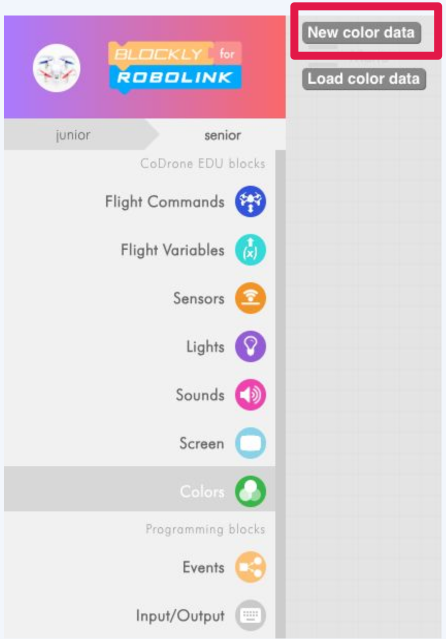
Function Description

Color Sensors

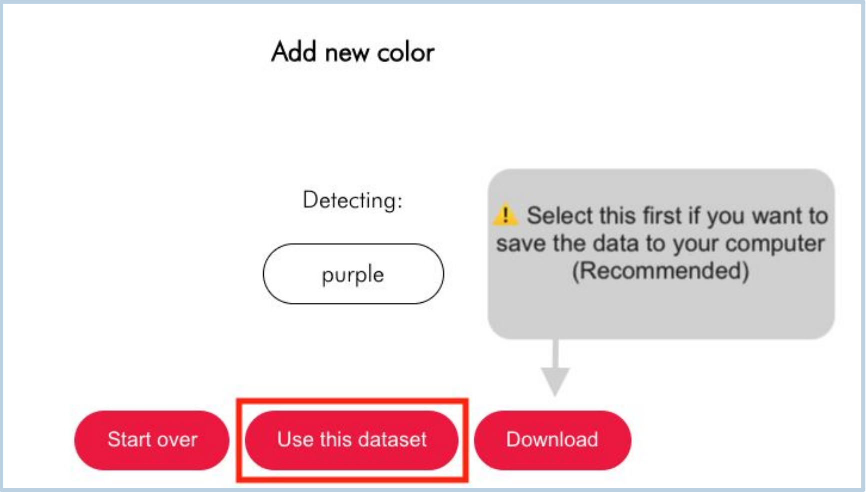
Data	Range	Units	Block	Python
Front color name	–	_	 get front color	`drone.get_front_color()`
Back color name	–	–	 get back color	`drone.get_back_color()`
Front color hue (H) *	0-360	_	 get front hue value	`drone.get_front_color(“hsv”)[0]`
Front color saturation (S) *	0-100	_	 get front saturation value	`drone.get_front_color(“hsv”)[1]`
Front color value (V) *	0-100	_	 get front value value	`drone.get_front_color(“hsv”)[2]`
Front color lightness (L) *	0-100	_	 get front lightness value	`drone.get_front_color(“hsl”)[2]`

* Also available for the back color

Adding a new color data set



(Left) Follow the steps to create a new dataset using the colors you want to add.


Add new color

Detecting:

purple

⚠️ Select this first if you want to save the data to your computer (Recommended)

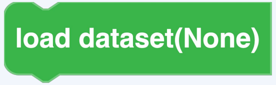
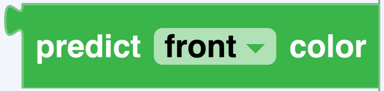
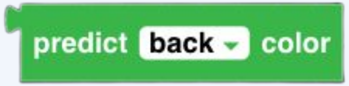
(Above) ⚠️Click the download button to save the data.

Lesson: 3.7 - Color Sensor

Note: This feature is currently unavailable for *Python for Robolink*

Function Description

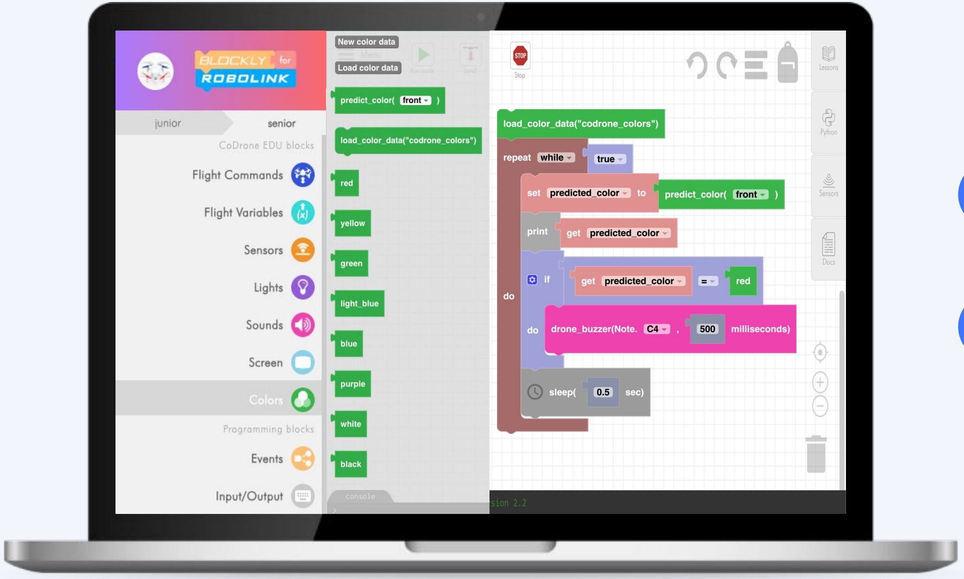
Color Sensors

Function Description	Parameters	Returns	Block	Python
Loads dataset for classifying customer colors	Folder of .txt files	–	 load dataset(None)	`drone.load_color_data(None)`
Predicts the front color based on the dataset. Requires user to load the dataset.	–	–	 predict front color	`drone.predict_color(drone.get_color_data())[0]`
Predicts the back color based on the dataset. Requires user to load the dataset.	-	–	 predict back color	`drone.predict_color(drone.get_color_data())[1]`

Use the green color blocks for custom color data

- These blocks will work with data you added
- To use the green "predict" block, make sure you have loaded in data

Loading color data



```
load_color_data("codrone_colors")
repeat while true
do
    set predicted_color to predict_color( front )
    print get predicted_color
    if get predicted_color = red
    do drone_buzzer(Note. C4 , 500 milliseconds)
    sleep( 0.5 sec)
```

To reuse the program you wrote previously (with data) you must do 2 things:

1. Save and load your XML file (the actual Blockly code

2. Save and load the color data! The Colors menu must be populated with blocks. **Match** the data set name with your code.

Lesson: 3.7 - Color Sensor

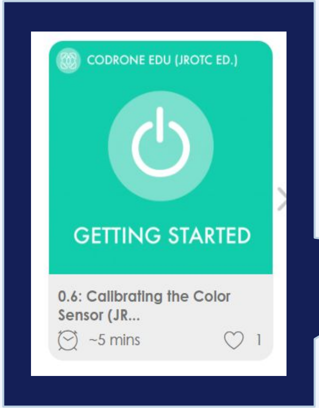
Note: This feature is currently unavailable for *Python for Robolink*

Troubleshooting

x1
white
black
red
yellow
green
light blue
blue
purple

Color calibration is the process of calibrating the color sensor to the default values (*Required for JROTC ed.)

- Also available for standard CoDrone EDU but not required. It may help improve the accuracy.
- Use the included color cards


CODRONE EDU (JROTC ED.)

GETTING STARTED

0.6: Calibrating the Color Sensor (JR...

~5 mins

1

Note, this lesson is specific only to the JROTC ed. and can be found in the JROTC getting started.

Lesson: 0.6 Calibrating the Color Sensor

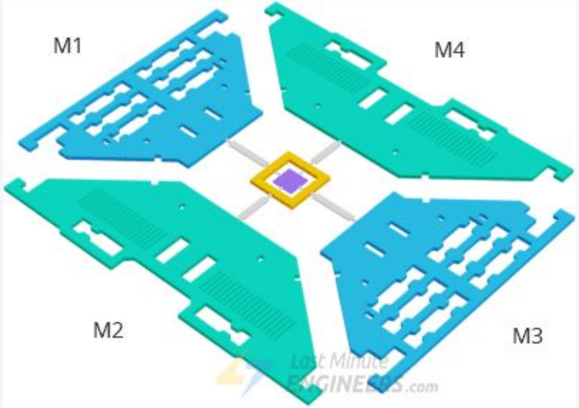
5

Gyroscope and Accelerometer

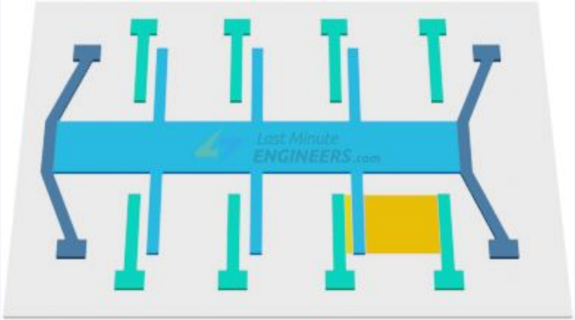
Introduction

Gyroscope and Accelerometer

- These sensors are typically found on the same chip
- Also known as an **inertial measurement unit (IMU)**
- If you looked at these components under a microscope you would find tiny moving parts that are measuring acceleration and rotation.
- They measure the forces when there is a change in motion.



Gyroscope



Accelerometer

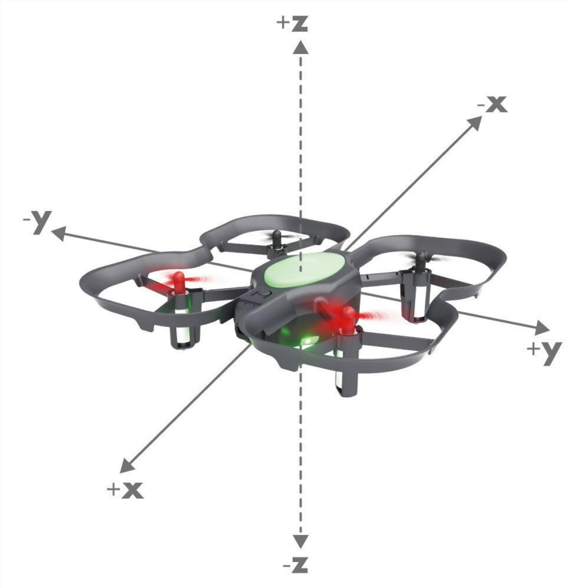
Images: Last Minute Engineers

ROBOLINK

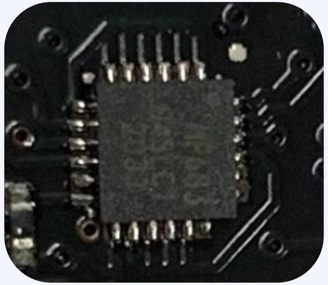
CoDrone EDU Usage

Gyroscope and Accelerometer

- The sensor is found on the drone PCB (not visible from the outside of the drone)
- The front of the drone points towards the positive X-Axis
- Used by the drone to hover and stabilize
- The drone also uses this information to make turns and detect other motion (such as collisions)

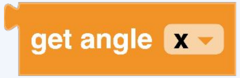
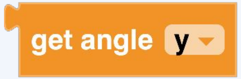
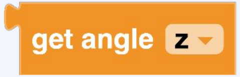
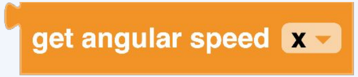
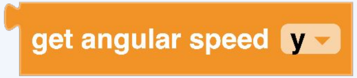
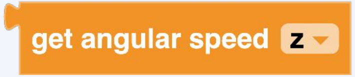


Integrated circuit (IC) chip



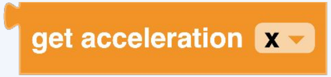
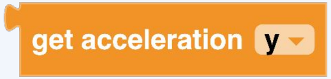
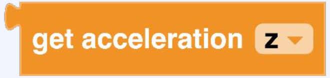
Function Description

Gyroscope and Accelerometer

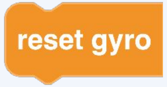
Data	Range	Units	Block	Python
X-Angle (Roll)	-180-180	degrees	 get angle x	`drone.get_angle_x()`
Y-Angle (Pitch)	-180-180	degrees	 get angle y	`drone.get_angle_y()`
Z-Angle (Yaw)	-180-180	degrees	 get angle z	`drone.get_angle_z()`
X-Angular Velocity	-2000-2000	degrees per second	 get angular speed x	`drone.get_angular_speed_x()`
Y-Angular Velocity	-2000-2000	degrees per second	 get angular speed y	`drone.get_angular_speed_y()`
Z-Angular Velocity	-2000-2000	degrees per second	 get angular speed z	`drone.get_angular_speed_z()`

Function Description

Gyroscope and Accelerometer

Data	Range	Units	Block	Python
X-Acceleration	-1568-1568	meters/second^2 x 10	 get acceleration x	`drone.get_accel_x()`
Y-Acceleration	-1568-1568	meters/second^2 x 10	 get acceleration y	`drone.get_accel_y()`
Z-Acceleration	-1568-1568	meters/second^2 x 10	 get acceleration z	`drone.get_accel_z()`

! Remember the IMU is resets when the drone powers on or takes off. You can reset it using this function.


reset gyro

```
drone.reset_gyro()
```

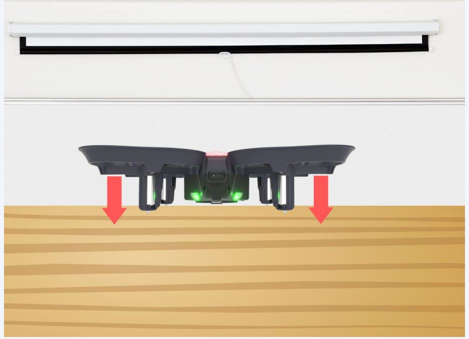
Troubleshooting

Gyroscope and Accelerometer

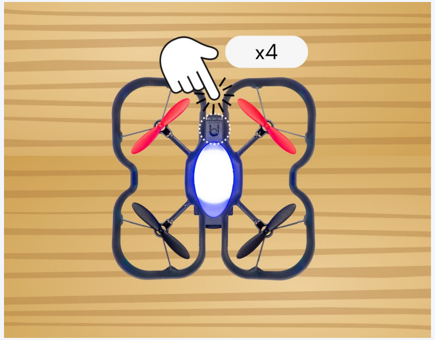
My controller screen says “attitude not stable” or “motion calibration error”.

Solution

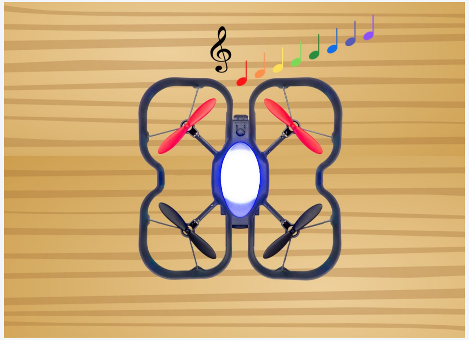
1. Try resetting the gyroscope and accelerometer on a flat surface.
2. If the controller continues or now says “motion calibration error”, the component is damaged.



Place the drone on a flat surface.



Press the button 4 times.



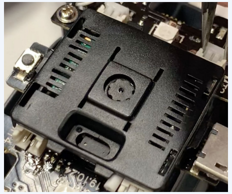
Wait for the sounds!

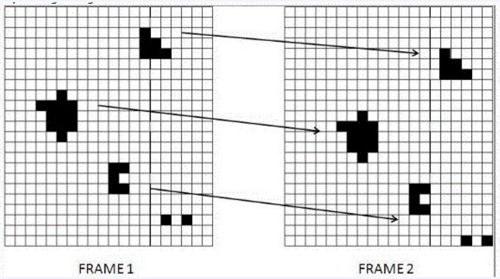
6 Optical Flow Sensor

Introduction

Optical Flow Sensor

- The optical flow sensors work like your mouse!
- It uses a tiny low-resolution camera to compare two images and see if the sensor has moved
- Works best when there is texture and non-repeating patterns
  - Have you tried to use a mouse on a glass table before?
- Works together with height to know how far you are off of the ground





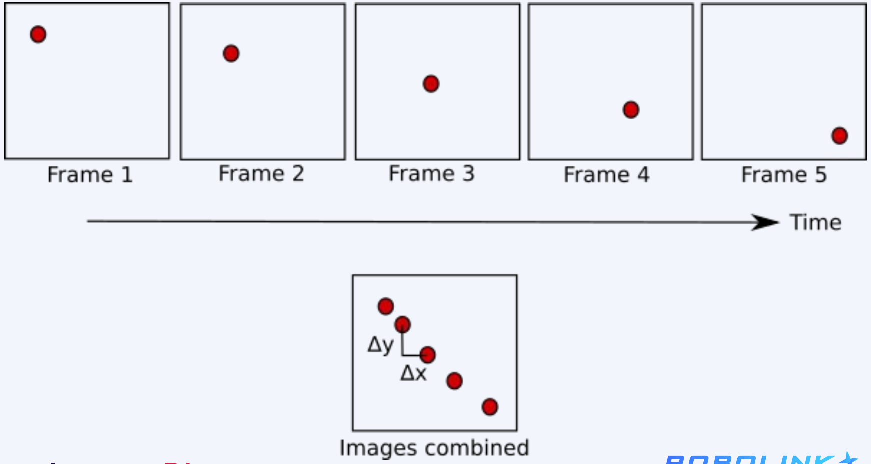


Image: Bitcraze

CoDrone EDU Usage

Position Sensor

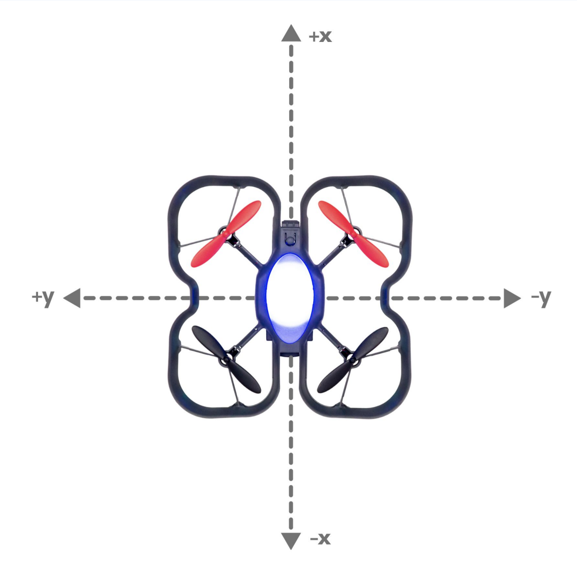
Understanding the position (optical flow) sensor

1. Drone initializes coordinates (0,0) at the takeoff location. It will reset to these values at every takeoff.

2. Optical flow sensor only updates when color sensor is inactive (while drone is moving or off the ground)

    - This also means while you're picking it up!

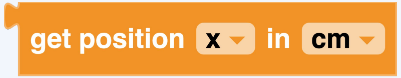
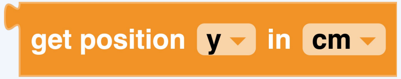
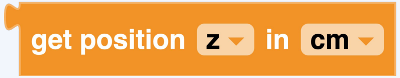
3. This sensor relies on accurate height sensor data. Works best at heights below 1.5 meters.



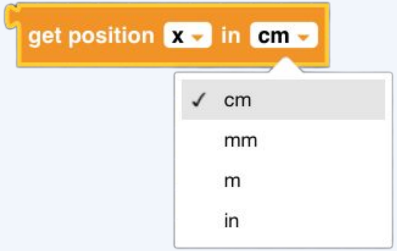
The optical flow sensor detects surface feature changes - similar to your computer mouse! 🐭

Function Description

Position Sensors

Data	Range	Units	Block	Python
X-Position	-1000 ~ 1000	centimeters	 get position x in cm	`drone.get_pos_x()`
Y-Position	-1000 ~ 1000	centimeters	 get position y in cm	`drone.get_pos_y()`
Z-position (Sensor fusion*)	-1000 ~ 1000	centimeters	 get position z in cm	`drone.get_pos_z()`

! Other units available



get position x in cm

Troubleshooting

Position Sensors

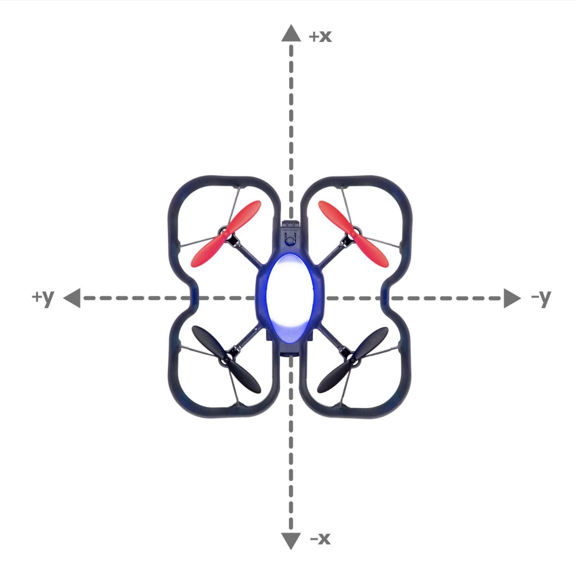
Environmental factors

Surface materials/pattern

- Optical flow needs a non-linear pattern (Competition programming mat provides optimal flight performance).
- Surface should be well-lit.

Uneven surface

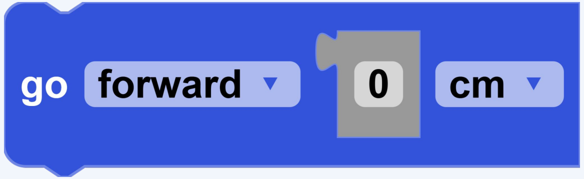
- For best performance stick to flying over a consistent flat surface.



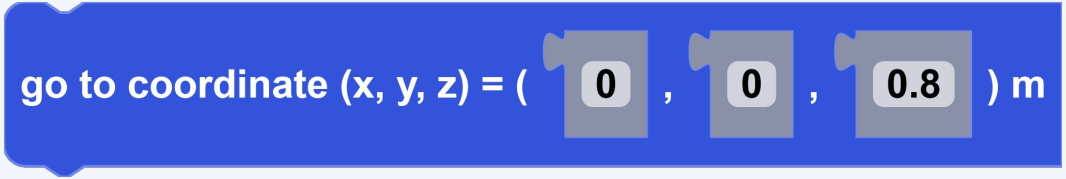
⚠️ **It is possible to disable this sensor using your controller settings (Position vs Attitude control). Watch the video to learn more.**

New functions!

Position Sensors



go forward ▼ 0 cm ▼



go to coordinate (x, y, z) = (0 , 0 , 0.8) m

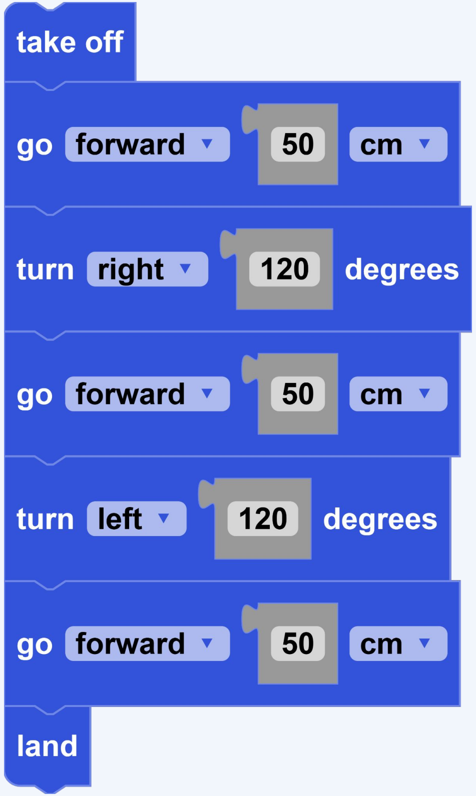
- Instructs the drone to fly a certain distance from its current location
- Works best when flying over the programming mat and at lower heights (a meter or below)

- Instructs the drone to fly to a certain coordinate
- The origin (0,0,0) is set at takeoff
- Works best when flying over the programming mat and at lower heights (a meter or below)

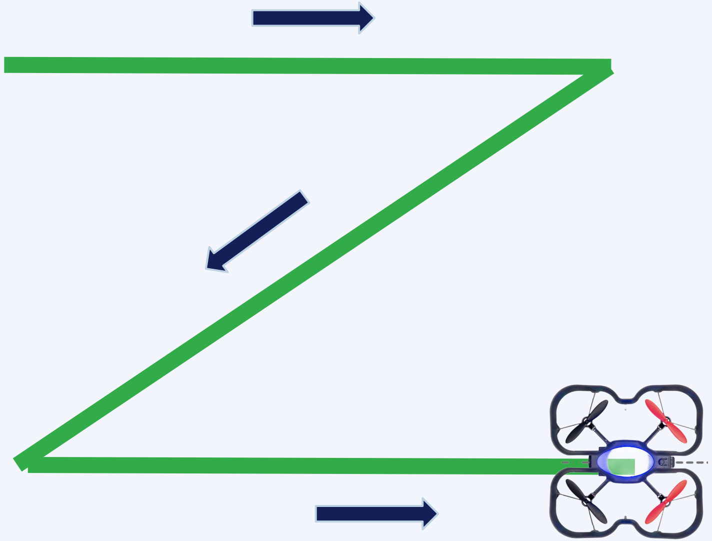
Because these functions rely on the same optical flow (position) sensor, it is subject to the same limitations as other sensor-based functions.

New function example

Position Sensors



```
take off
go forward 50 cm
turn right 120 degrees
go forward 50 cm
turn left 120 degrees
go forward 50 cm
land
```



7 Barometer

Introduction

Barometer

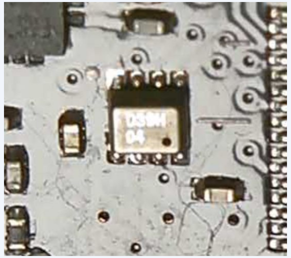
- A barometer measures air pressure
- Common units
  - pascals (Pa)
  - millibars
  - Atmospheres (atm)
- 100 pascals = 1 millibar (common unit for meteorology)
- Air pressure can also be used to estimate the elevation or altitude of aircraft
  - Affected by temperature, humidity, etc.

CoDrone EDU Usage

Barometer

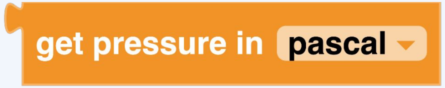
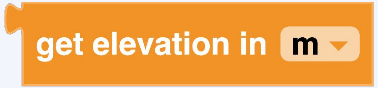
- The pressure sensor has a membrane very sensitive to changes in air pressure. As the membrane moves, the chip produces a current that is proportional to the pressure.
- CoDrone EDU returns pressure in pascals by default
- The drone combines range data with barometer data for more accurate height sensing
- Also used to estimate elevation

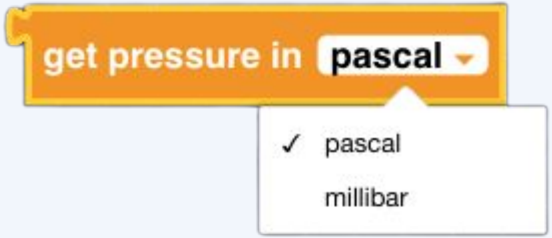
Integrated circuit (IC) chip

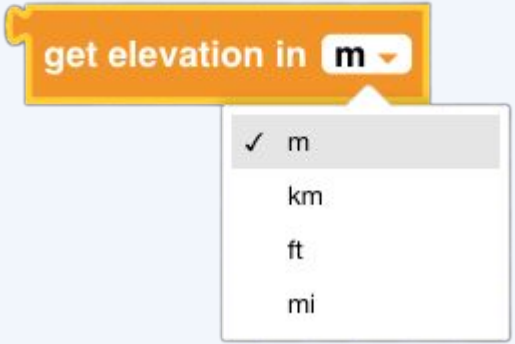


Function Description

Barometer

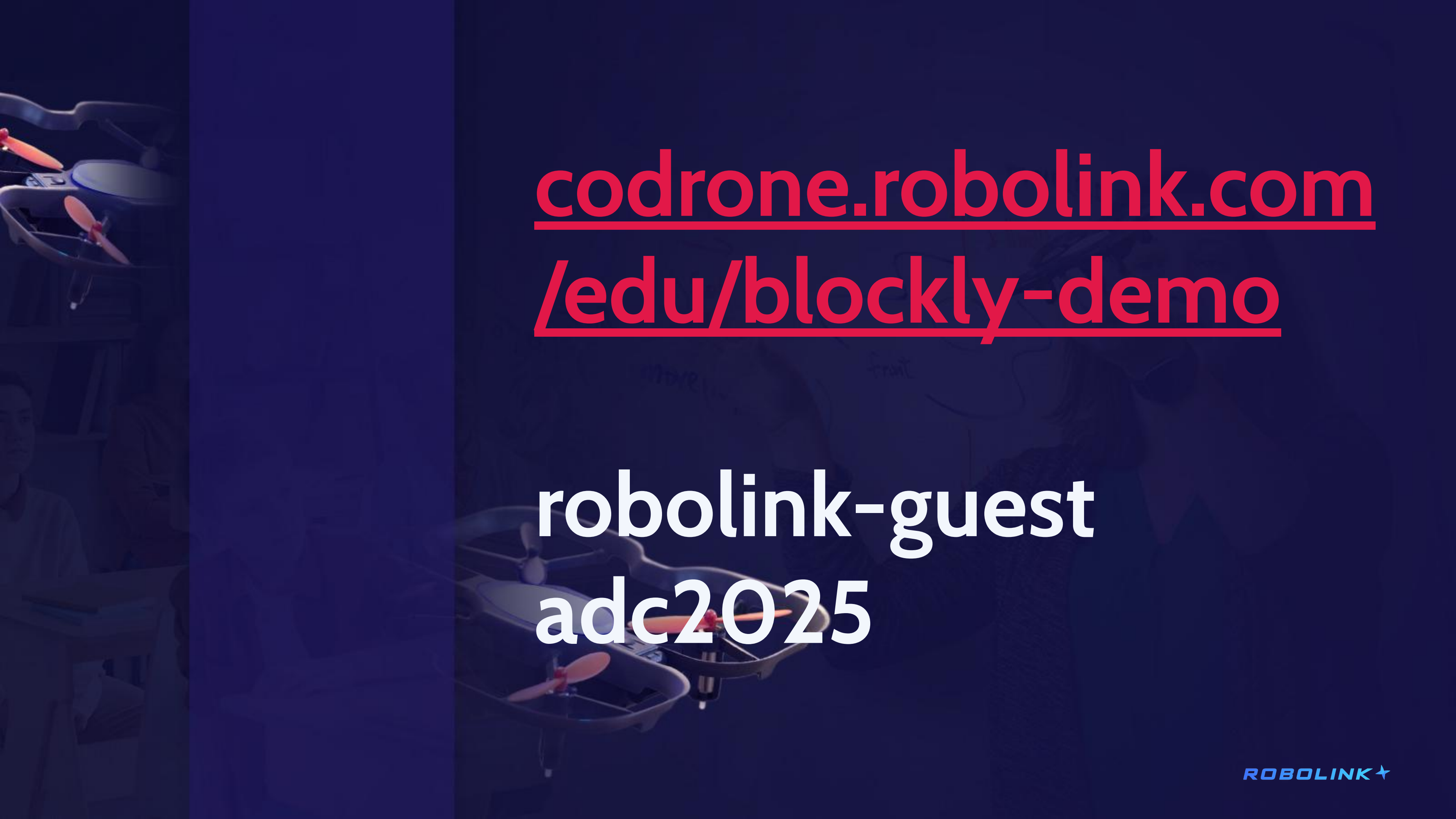
Data	Range	Units	Block	Python
Pressure	–	pascals	 get pressure in pascal	`drone.get_pressure()`
Altitude/Elevation	-1568-1568	meters	 get elevation in m	`drone.get_elevation()`





! Other units are available

8 Demos

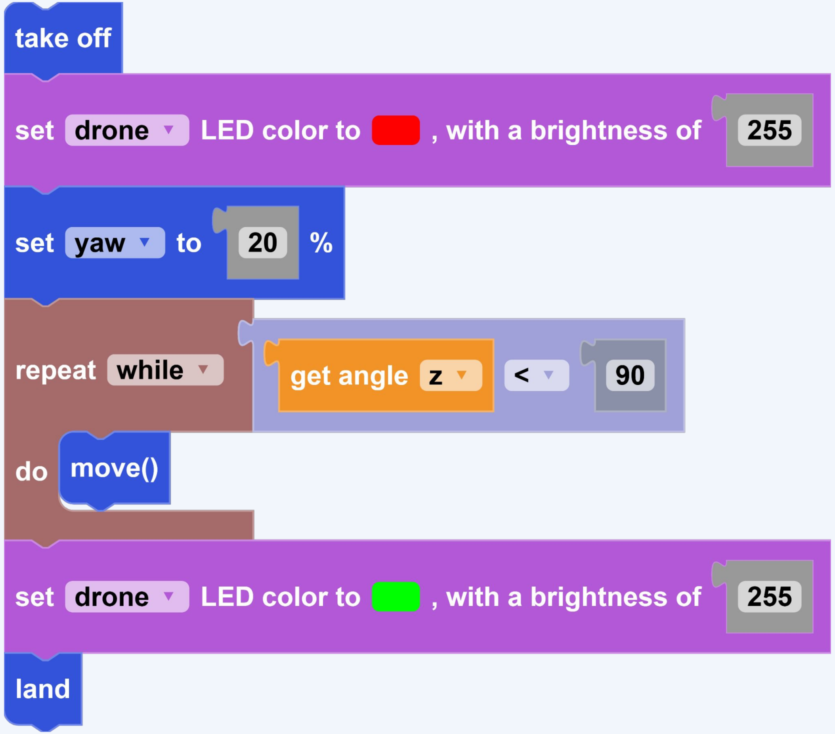


codrone.robolink.com/edu/blockly-demo

robolink-guest
adc2025

Example

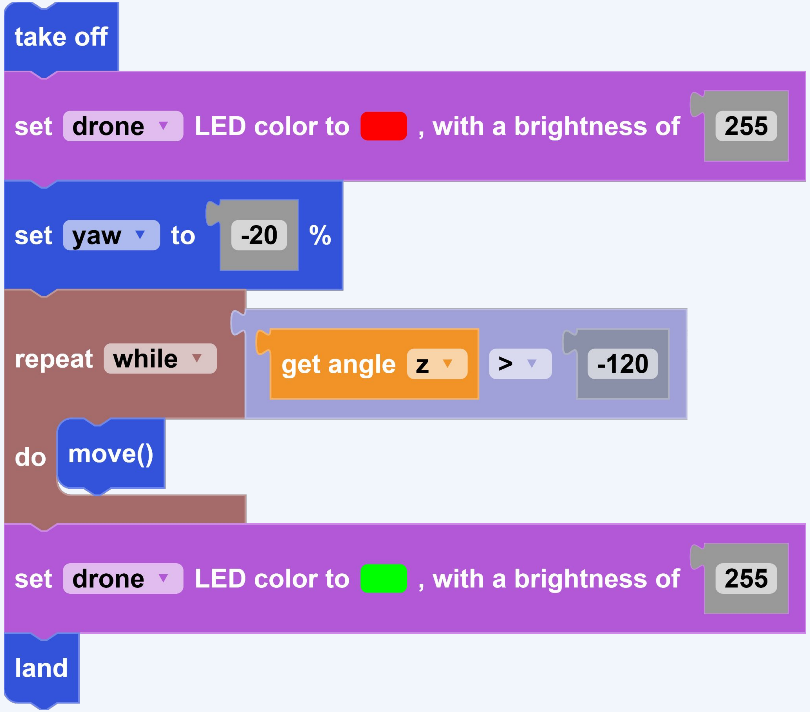
Gyroscope



```
take off
set drone ▾ LED color to 🟥 , with a brightness of 255
set yaw ▾ to 20 %
repeat while ▾ get angle z ▾ < ▾ 90
do  move()
set drone ▾ LED color to 🟩 , with a brightness of 255
land
```

Solution

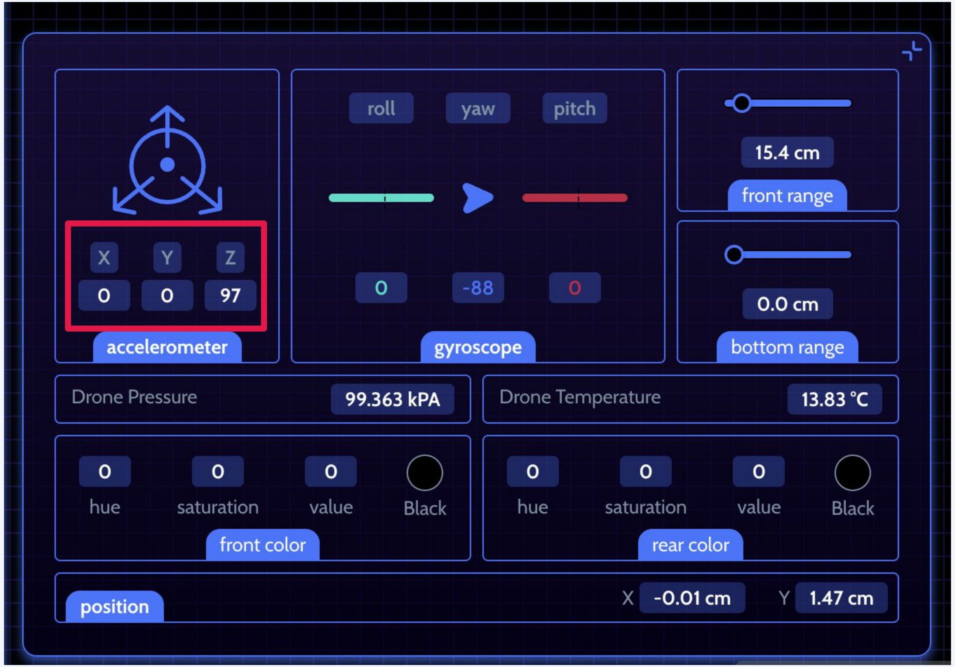
Gyroscope



```
take off
set drone ▾ LED color to 🟥 , with a brightness of 255
set yaw ▾ to -20 %
repeat while ▾ get angle z ▾ > ▾ -120
do  move()
set drone ▾ LED color to 🟩 , with a brightness of 255
land
```

Example

Accelerometer



accelerometer

X	Y	Z
0	0	97

gyroscope

roll	yaw	pitch
0	-88	0

front range 15.4 cm

bottom range 0.0 cm

Drone Pressure 99.363 kPA

Drone Temperature 13.83 °C

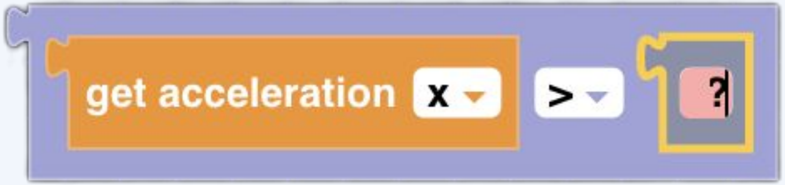
front color

hue	saturation	value	
0	0	0	Black

rear color

hue	saturation	value	
0	0	0	Black

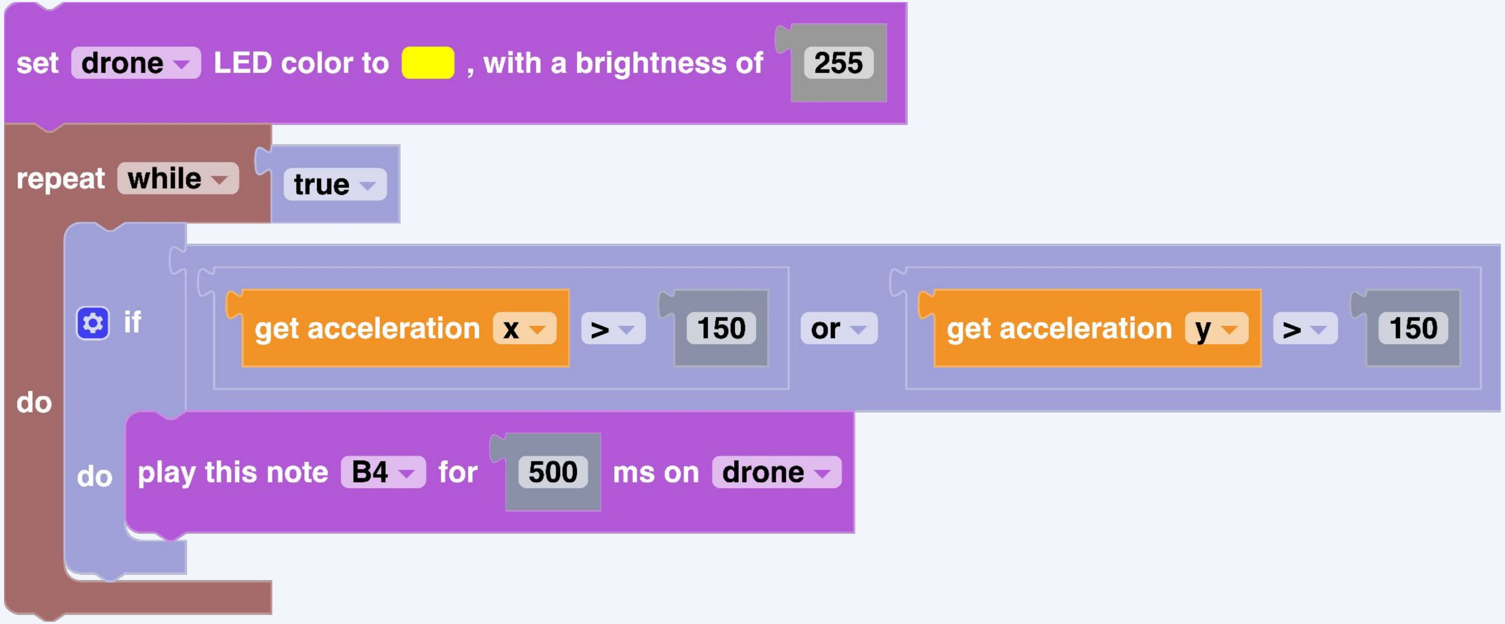
position X -0.01 cm Y 1.47 cm



```
get acceleration x > ?
```

Solution

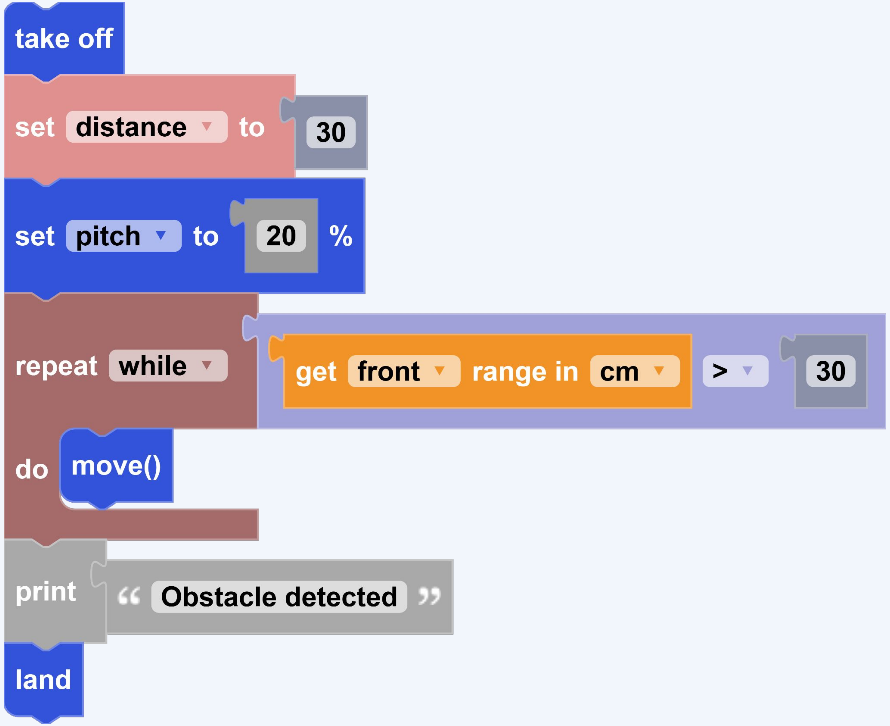
Accelerometer



```
set drone LED color to 🟨 , with a brightness of 255
repeat while true
do
    if get acceleration x > 150 or get acceleration y > 150
    do
        play this note B4 for 500 ms on drone
```

Example

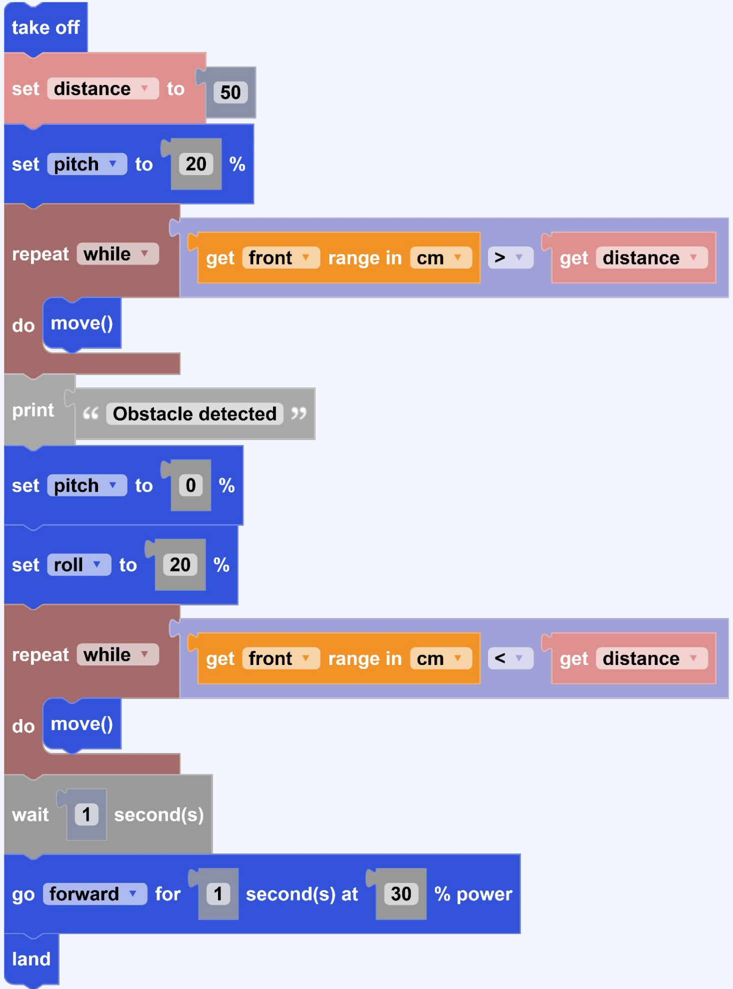
Front Range



```
take off
set distance to 30
set pitch to 20 %
repeat while get front range in cm > 30
do
    move()
print “ Obstacle detected ”
land
```

Solution

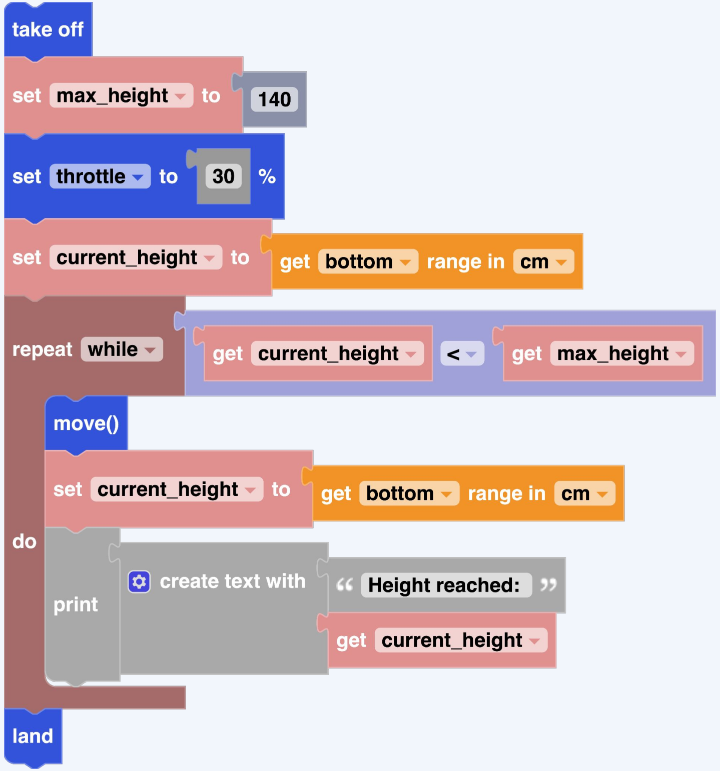
Front Range



```
take off
set distance to 50
set pitch to 20 %
repeat while get front range in cm > get distance
do  move()
print “ Obstacle detected ”
set pitch to 0 %
set roll to 20 %
repeat while get front range in cm < get distance
do  move()
wait 1 second(s)
go forward for 1 second(s) at 30 % power
land
```

Example

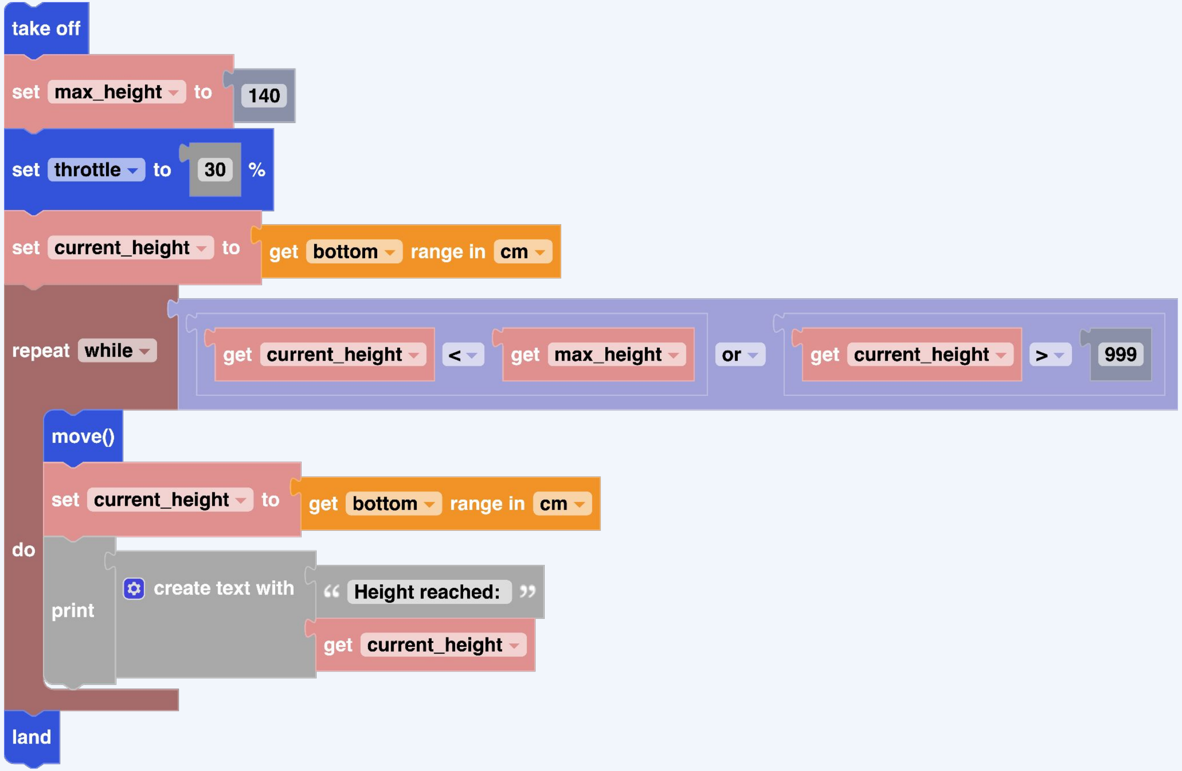
Bottom Range



```
take off
set max_height to 140
set throttle to 30 %
set current_height to get bottom range in cm
repeat while get current_height < get max_height
do
    move()
    set current_height to get bottom range in cm
    print create text with “ Height reached: ”
                           get current_height
land
```

Solution

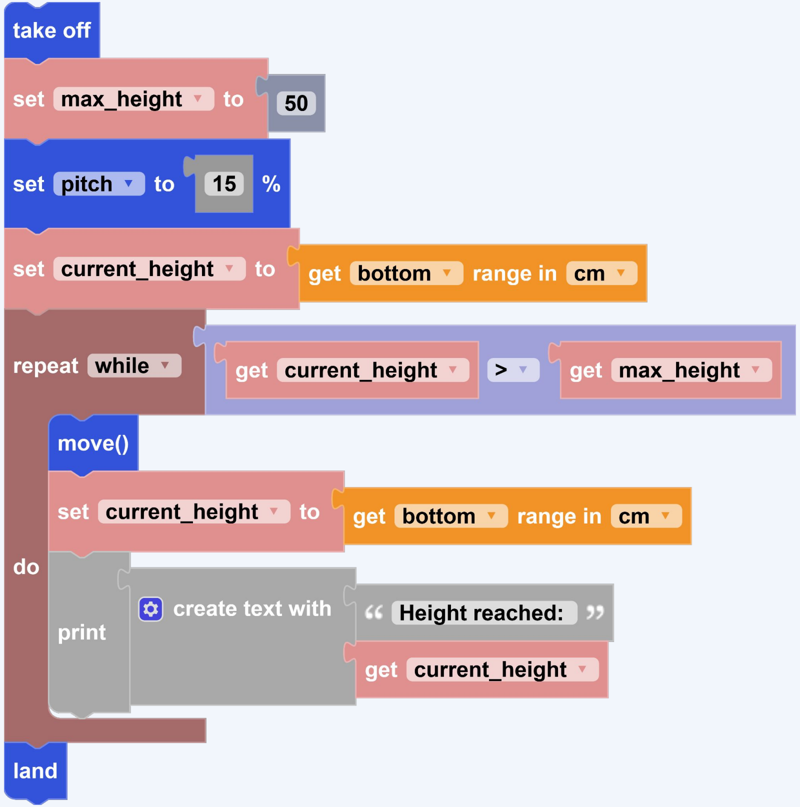
Bottom Range



```
take off
set max_height to 140
set throttle to 30 %
set current_height to get bottom range in cm
repeat while get current_height < get max_height or get current_height > 999
do
    move()
    set current_height to get bottom range in cm
    print create text with " Height reached: "
                           get current_height
land
```

Bonus

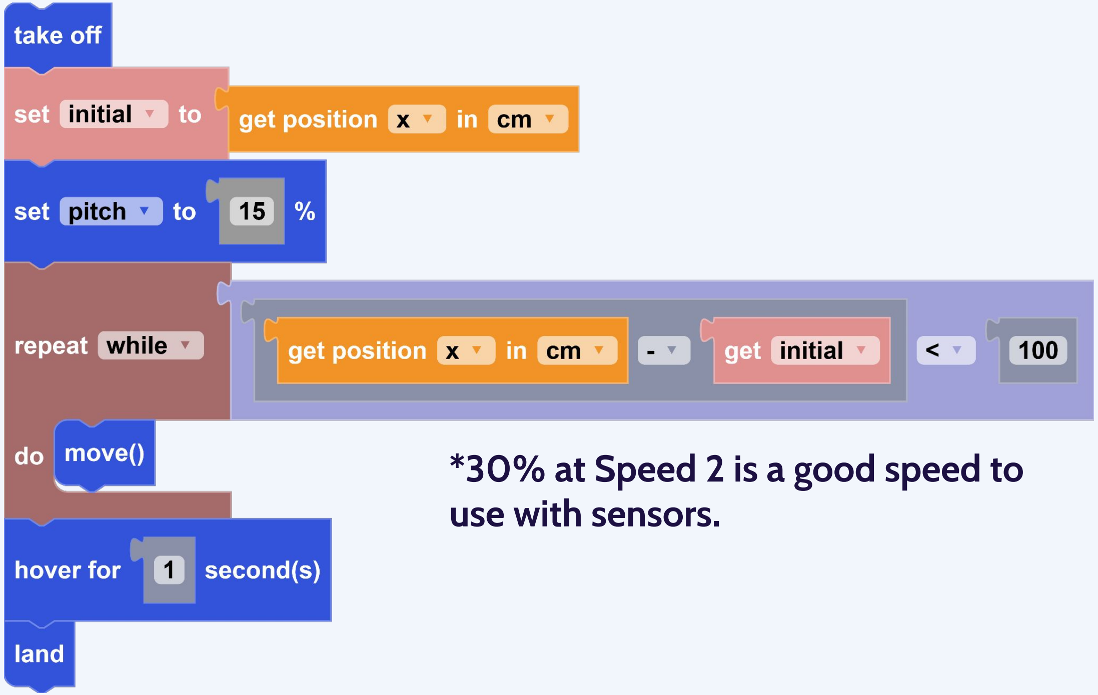
Bottom Range



```
take off
set max_height to 50
set pitch to 15 %
set current_height to get bottom range in cm
repeat while get current_height > get max_height
do
    move()
    set current_height to get bottom range in cm
    print create text with "Height reached:"
                           get current_height
land
```

Example

Position Sensor



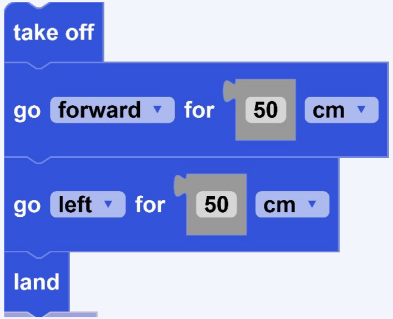
```
take off
set initial to get position x in cm
set pitch to 15 %
repeat while get position x in cm - get initial < 100
do move()
hover for 1 second(s)
land
```

*30% at Speed 2 is a good speed to use with sensors.

Solution

Position Sensor

*Note: Solution only shows 1 if statement



```
take off
go forward for 50 cm
go left for 50 cm
land
```

bit.ly/44IApTW

8 Resources

Resources

User Manual

Find getting started info and troubleshooting guides

User Manual

Basecamp

Free, online lessons for Blockly and Python with resources for teachers

https://learn.robolink.com/

Web Updater

Update your drone and controller using a web browser

https://codrone.robolink.com/edu/updater/

Blockly

Program using block-based programming

https://codrone.robolink.com/edu/blockly/

Python for Robolink

A web-based solution for programming in Python

https://codrone.robolink.com/edu/python/

Documentation

Functions guide for Python and Blockly

https://docs.robolink.com/

Robolink FAQs

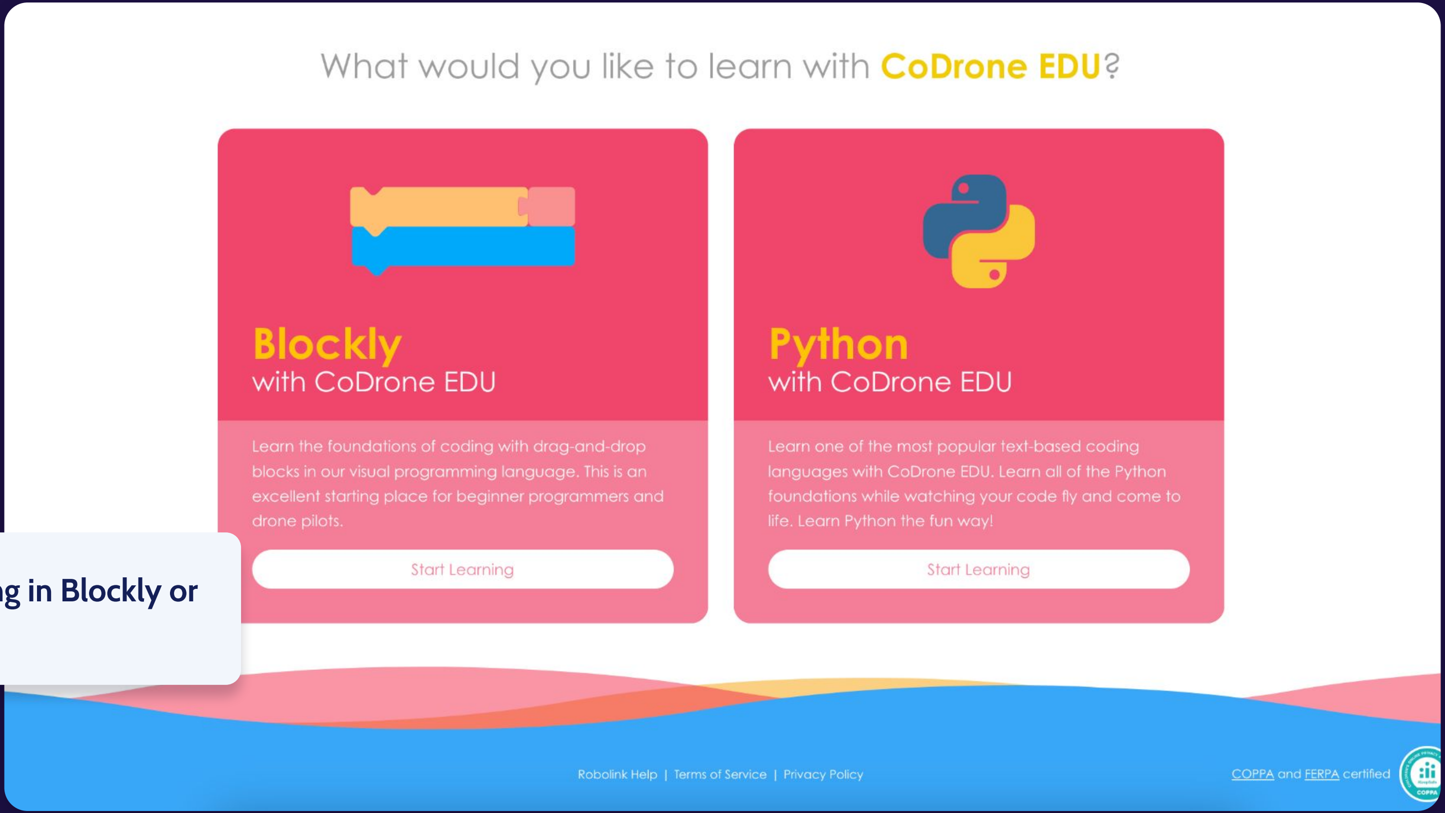
Visit https://help.robolink.com/

Need technical support?

Email us at support@robolink.com

Getting Started

learn.robolink.com/product/codrone-edu/



What would you like to learn with CoDrone EDU?

Blockly
with CoDrone EDU

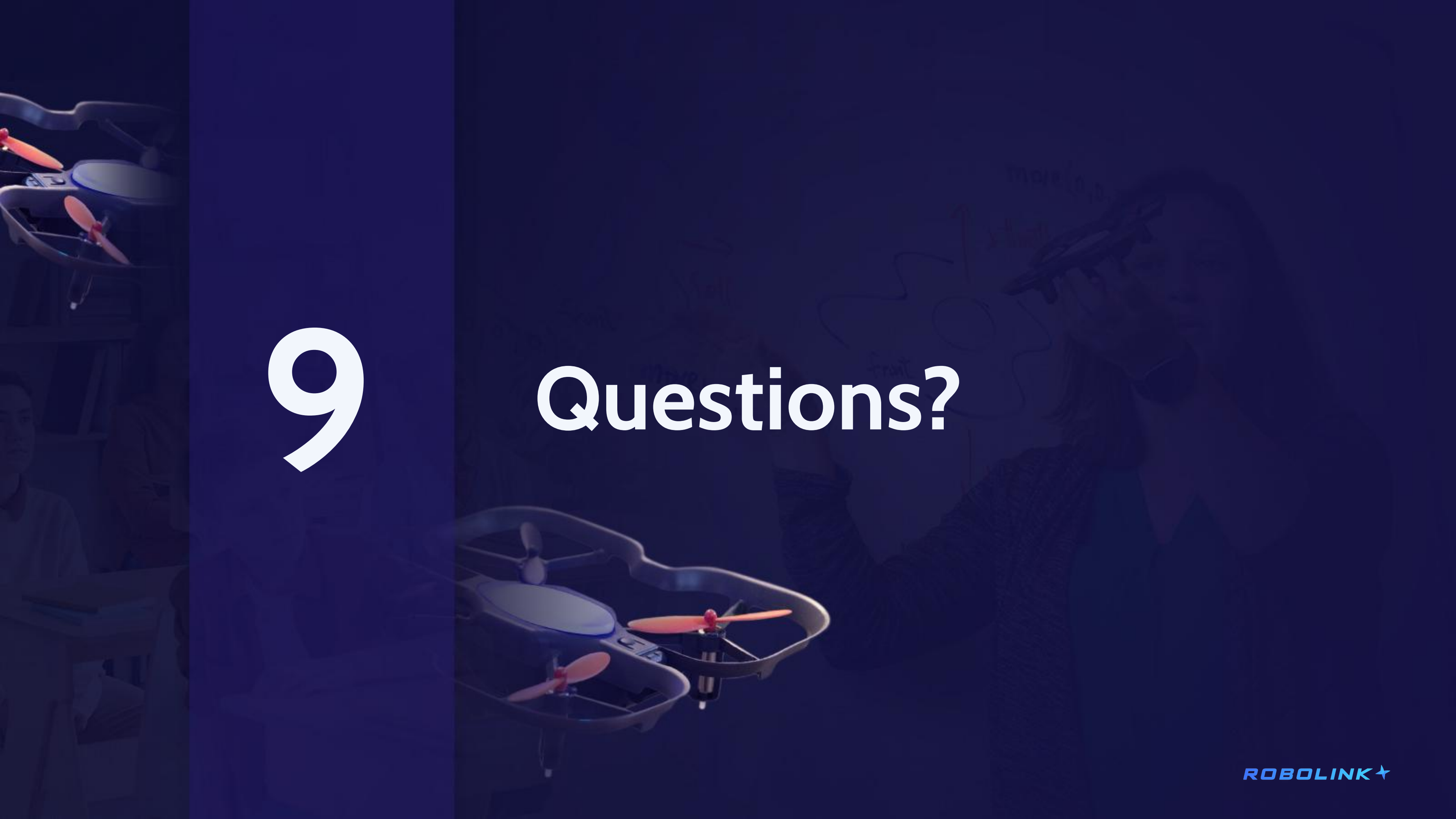
Learn the foundations of coding with drag-and-drop blocks in our visual programming language. This is an excellent starting place for beginner programmers and drone pilots.

Python
with CoDrone EDU

Learn one of the most popular text-based coding languages with CoDrone EDU. Learn all of the Python foundations while watching your code fly and come to life. Learn Python the fun way!

Start Learning
Robolink Help | Terms of Service | Privacy Policy
COPPA and FERPA certified

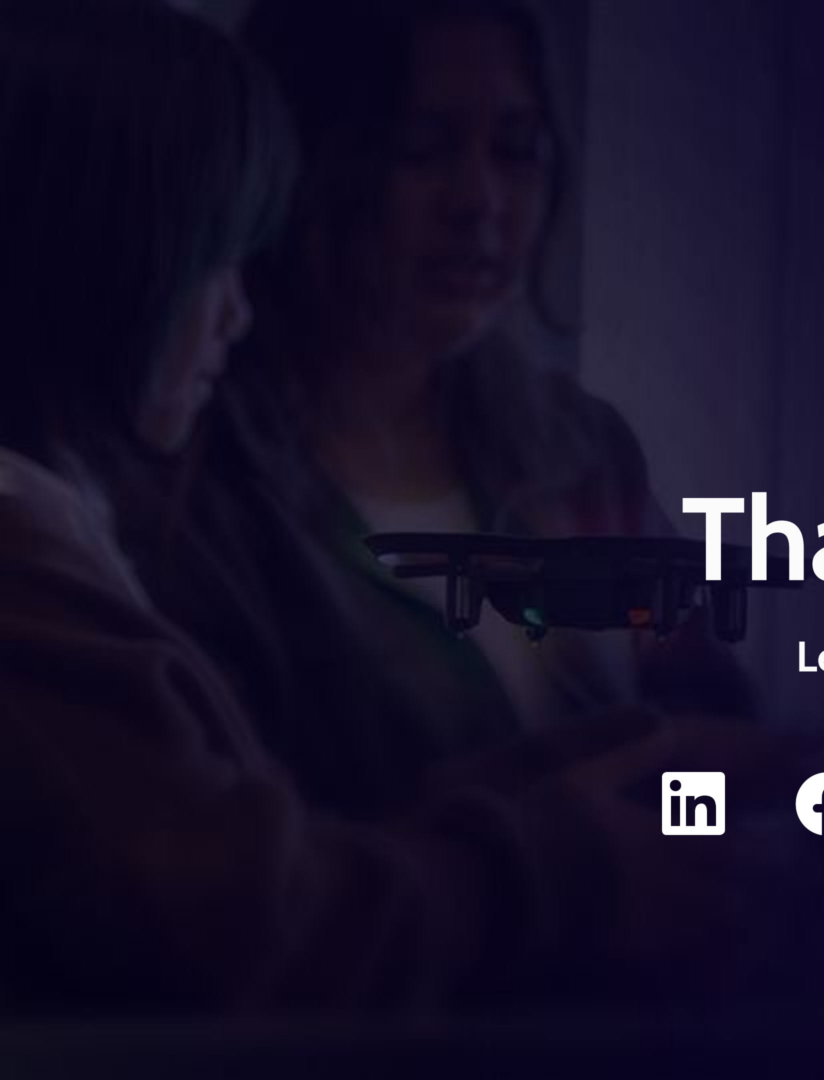
Start learning in Blockly or Python!



9

Questions?

Feedback:



Thank you!

Let’s stay in touch
@RobolinkInc