

*Hello everyone!*

*Turn your microphones off, and you  
may leave your camera on if you  
want.*

*Zoom Names should be written  
(Name - Team Number)*

*If you have no team, write "Student"*


*If you'd like to support,  
connect with me on LinkedIn! =>*






*Sandpiper Training Session*  
*Creating an Autonomous!*

**10/8/22**



**Presented by Rien Gupta, Kevin Li, and Rick Taylor**



# *If you have not already...*

*Last meeting, we took a look at arm mechanisms and VEX and some small tips you can use to build a better robot.*

Before trying out coding, consider creating

- an arm and drive base
- the main mechanism you use to score

If you need any help, with these, take a look at last week's meeting!



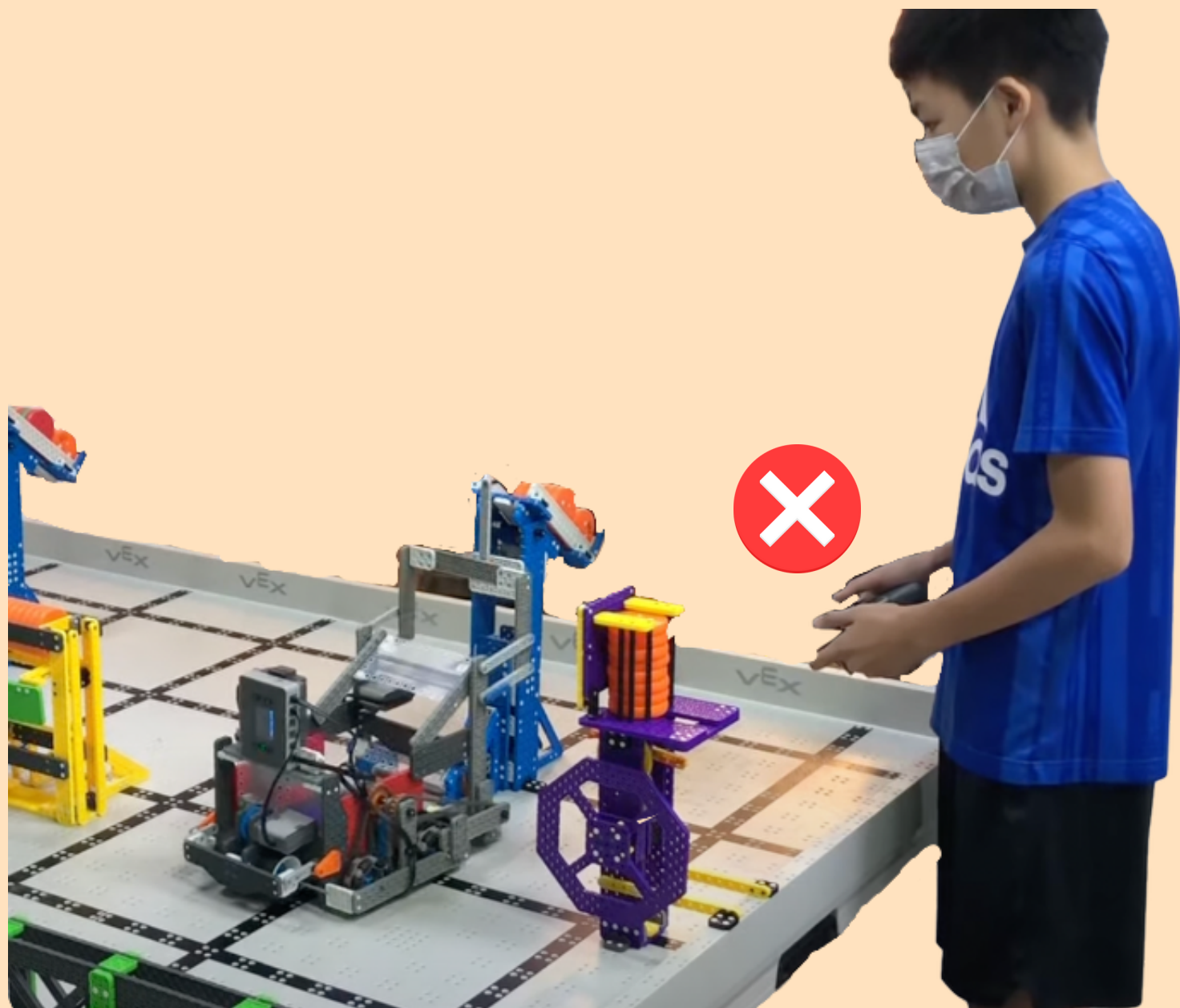
Presentation =>



# *What are Autonomous programs?*

**Autonomous Programs** are the true show of skill in coding for Vex Robotics. During a tournament, teams will have 1 minute skills matches where a robot will have to score as many points as they can **without a driver**.

Autonomous in Vex is complicated to understand, so we will be going over the basics.



# *Getting Started*

**VEXcode** allows students to get started coding quickly and easily. VEXcode is consistent across Blocks, Python, and C++. VEXcode can be used on a variety of platforms, even on a browser! Look at the QR Code below, or type [codeiq.vex.com](http://codeiq.vex.com)



# Basic Overview

## What is VEX code?

- Sequential instructions to the robot.
  - Tells the motors when to spin and how fast.
- Uses blocks, with clear, readable language

## What can you do in VEX code?

- If/Else statements
  - Sensor Inputs
- Loops
- Direct instructions

*Last but not Least...*

## Code is all about reliability!

The robot has little sense of its surroundings, can't fix it's mistakes and cannot adapt. Your job is to make a series of **rigid** instructions which can still work in many environments.



# What you need to program

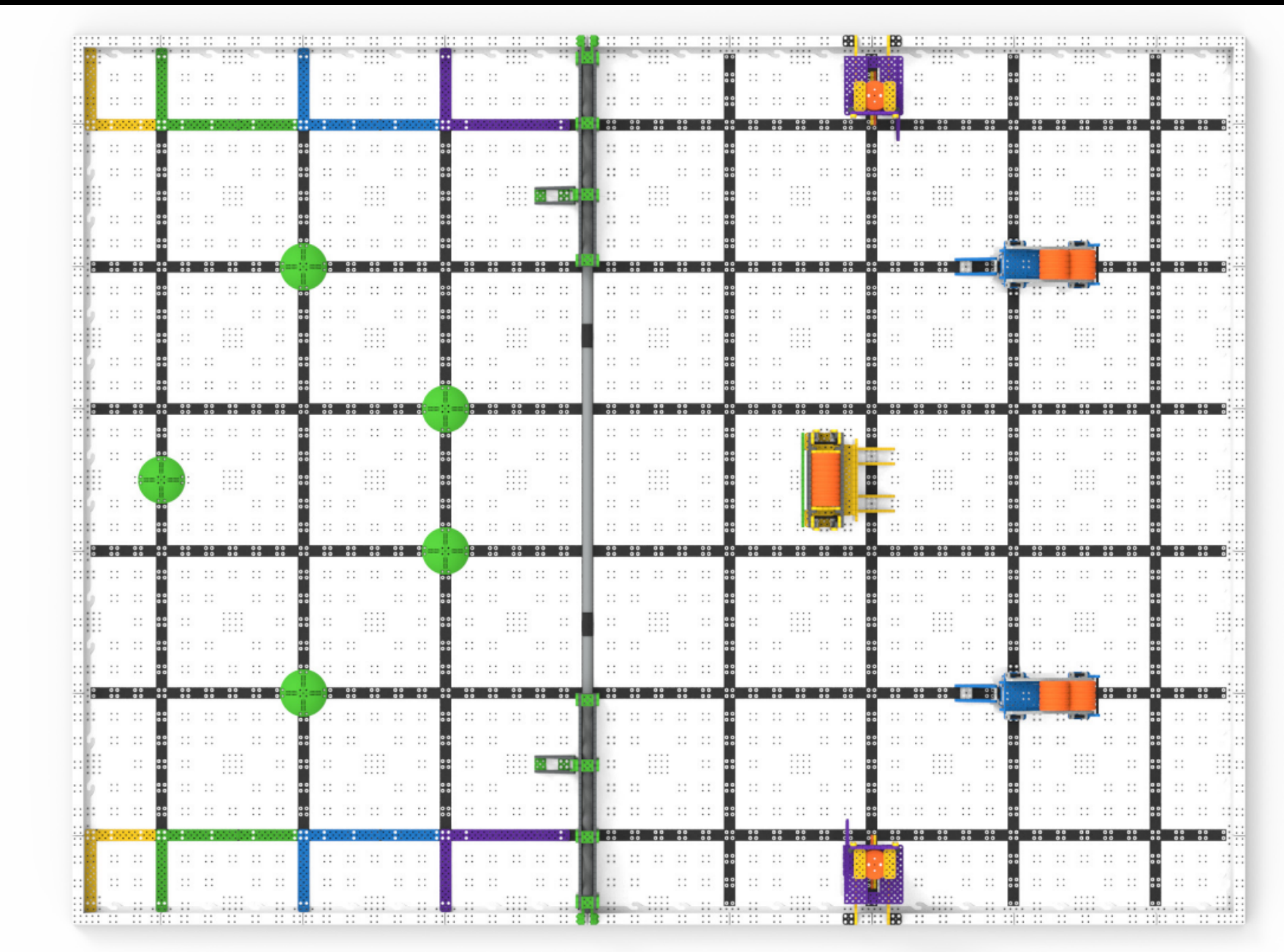
- 1 VEXCode
- 2 A Computer
- 3 Your Robot with a Vex Brain & controller
- 4 The right wires (USB-C for 2nd generation brains, Micro USB for older generation brains)



2nd generation brain



1st generation brain



# *Plan First!*

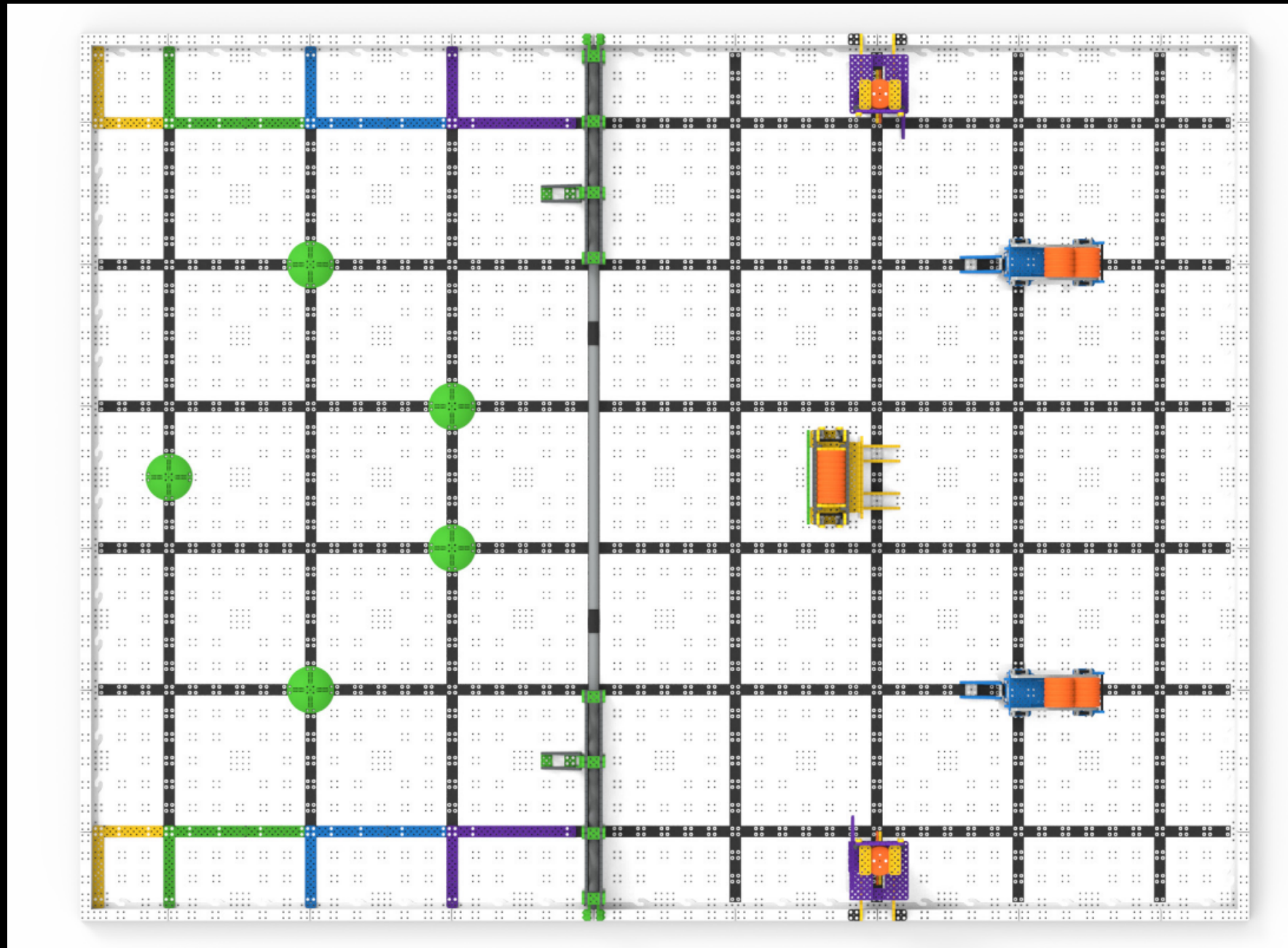
**Before starting, come up with a plan for your autonomous route. Do not free hand your autonomous program.**



# *Make it Reliable!*

**Before rushing to program collecting or scoring the disks, make sure **the route** the robot travels is **reliable!****

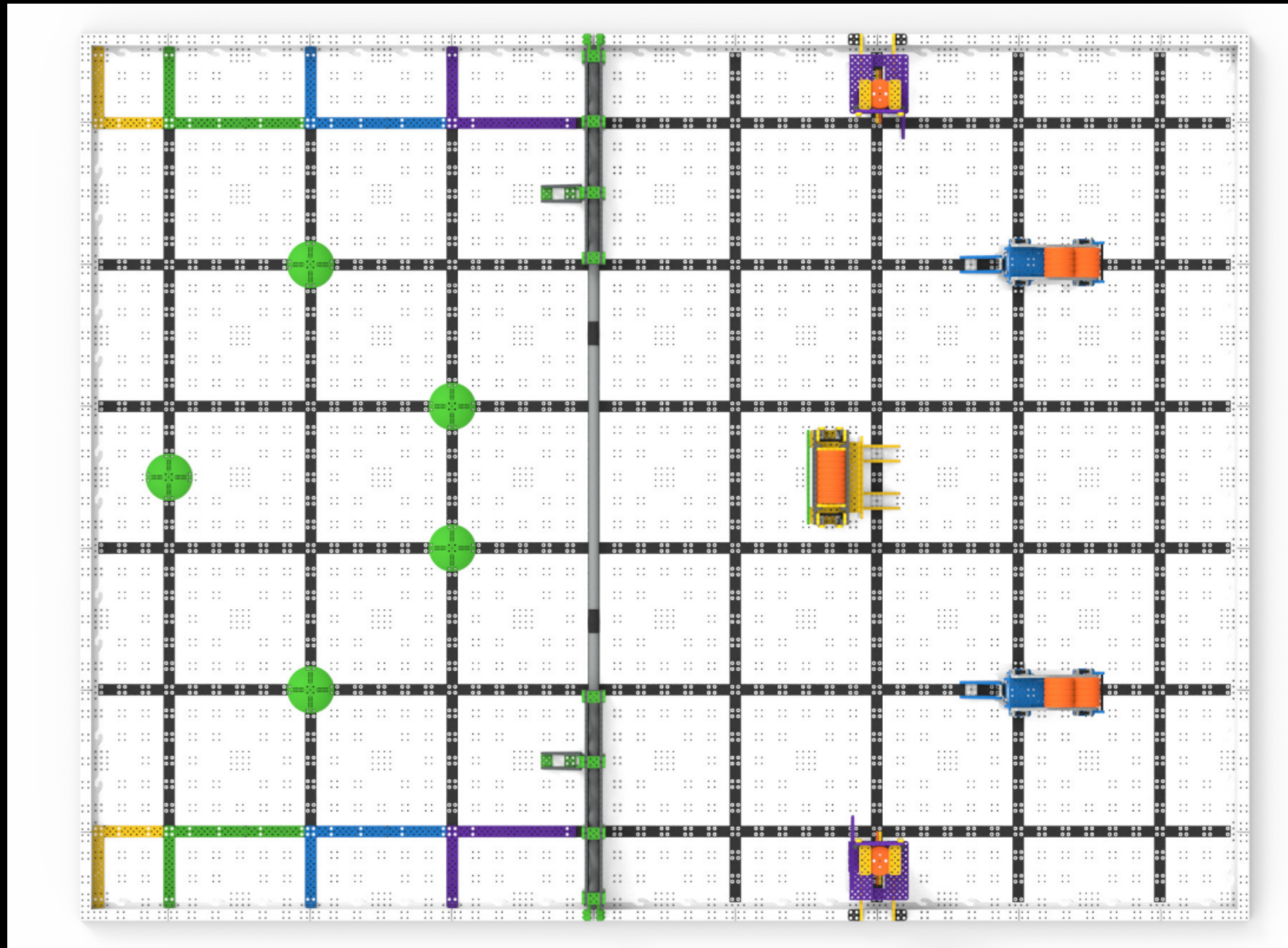
- **Move in directions parallel to the black lines.**
- **Align yourself using the field elements or the walls.**
- **Adjust the center of balance of the robot so it is more stable.**



# *Work in Modules!*

**Don't try and do everything in one go!**

- Make sure you have a few dispensers reliably working before moving on.
- Use comments!
- You can move the robot if there's no field elements in it!
  - Consider moving the robot back to the start after finishing scoring for reliability.



# Configure your devices

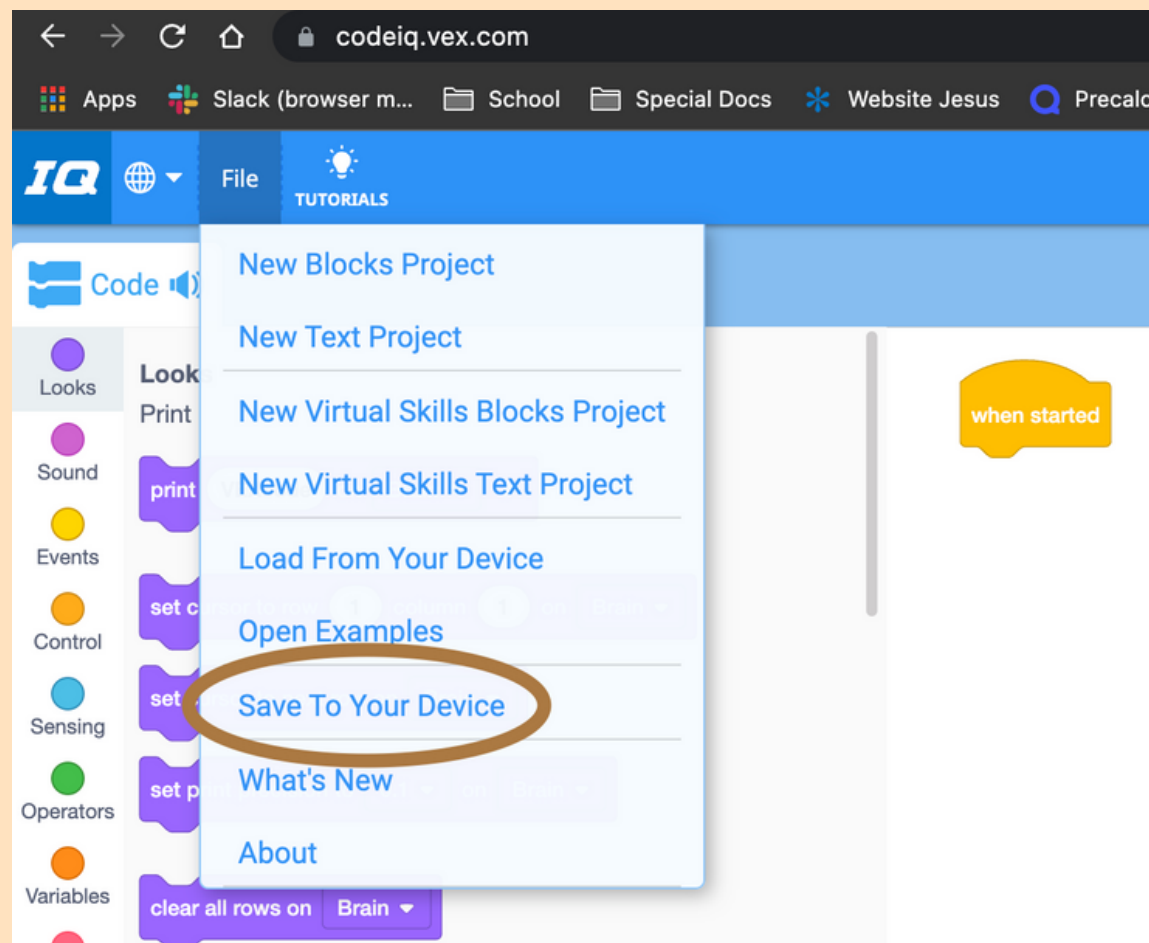
The screenshot displays the VEXcode IDE interface. The top menu bar includes 'IQ', 'File', 'TUTORIALS', 'UNDO', 'REDO', 'SLOT', 'VEXcode Project', 'CONTROLLER', 'BRAIN', 'DOWNLOAD', 'START', 'STOP', 'SHARE', and 'FEEDBACK'. The main workspace is divided into two sections: a block-based programming area on the left and a 'Devices' configuration panel on the right.

**Block-based Programming:**

- Motion:** 'spin ClawMotor close', 'spin ClawMotor close for 90 degrees', 'spin ClawMotor to position 90 degrees', 'stop ClawMotor'.
- Control:** 'set ClawMotor position to 0 degrees', 'set ClawMotor velocity to 50 %', 'set ClawMotor stopping to brake', 'set ClawMotor max torque to 50 %', 'set ClawMotor timeout to 1 seconds'.
- Drivetrain:** 'drive forward', 'drive forward for 200 mm', 'turn right', 'turn right for 90 degrees', 'turn to heading 90 degrees'.

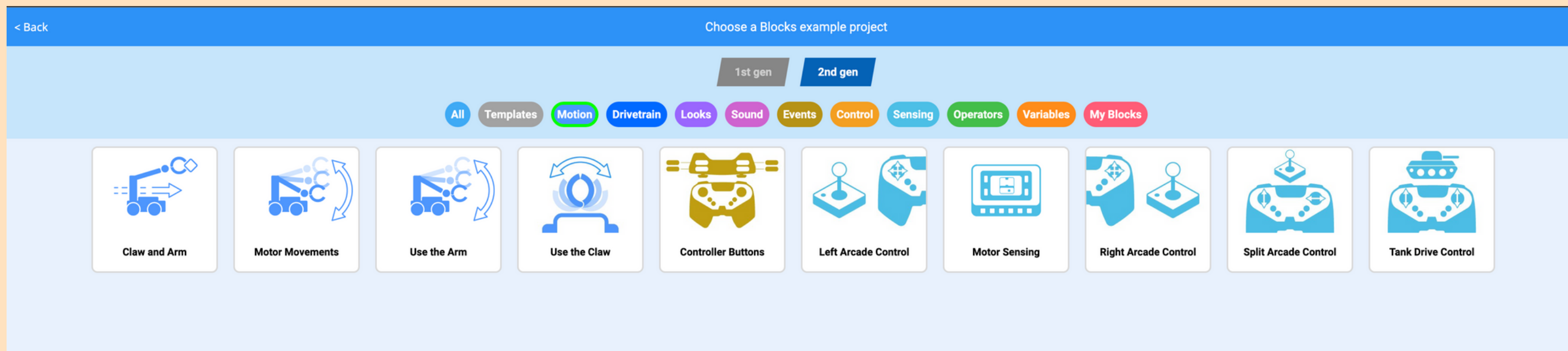
**Devices Panel:**

- IQ Robot Brain:** 1st gen, 2nd gen
- ClawMotor:** 4
- ArmMotor:** 10
- Drivetrain:** 1, 6, 3
- Controller:**
- SidewaysMotor:** 2
- Distance5:** 5
- Gyro12:** 12
- Color9:** 9
- Add a device**

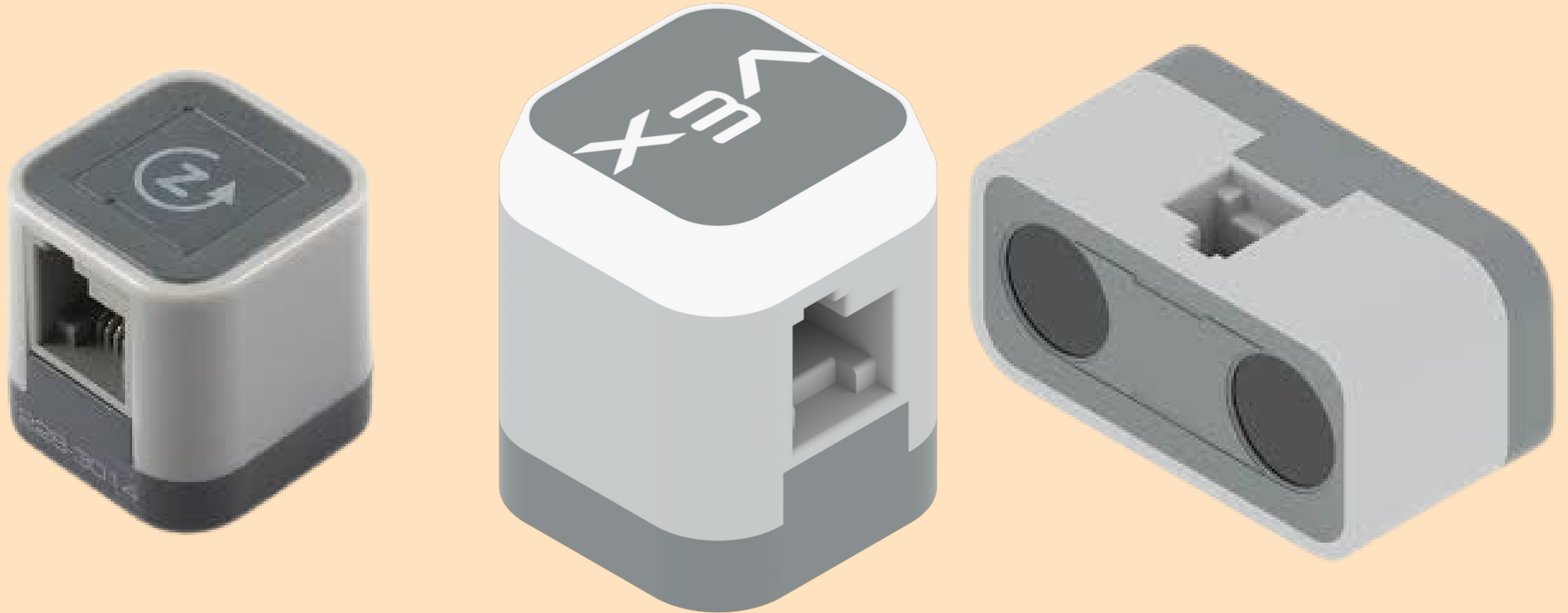


# Create a basic drive route!

Create a basic drive route for the robot to follow, and find the functions you will use to drive a motor.



*Looking for Precision  
... with Sensors!*



# Key Sensors

Gyro

Button/Bumper

Color

Proximity

# What do we use sensors for?

- ① Use sensors for alignment
- ② Precise tracking
- ③ Fixes position when something wrong happens

# Gyro Sensor!

## What does it do?

- Precisely tracks rotation and alignment of robot.

## How is it used?

- Place the gyro face up on the robot and use the gyro sensor to make sure your turns are reliable.
  - If there's some changes to the field, simply timing the turn won't work. Use a gyro!

*Make sure to calibrate the gyro before running the program.*  
The sensor needs to know the orientation your robot at the start!



(If you have a second generation brain, use inertial!)



# Color Sensor!

## What does it do?

- Senses the color and brightness of the object directly in front of them.

## How is it used?

- Use the color sensor face down on your robot to track the lines on the bottom of the field!
  - For example, a program could stop moving the robot after passing some number of lines. At that point, you'd know exactly where the robot is!

*This sensor can only sense **directly in front of it!** Make sure it is positioned well!*

Gen 1 Version



Gen 2 Version

# Proximity Sensor

## What does it do?

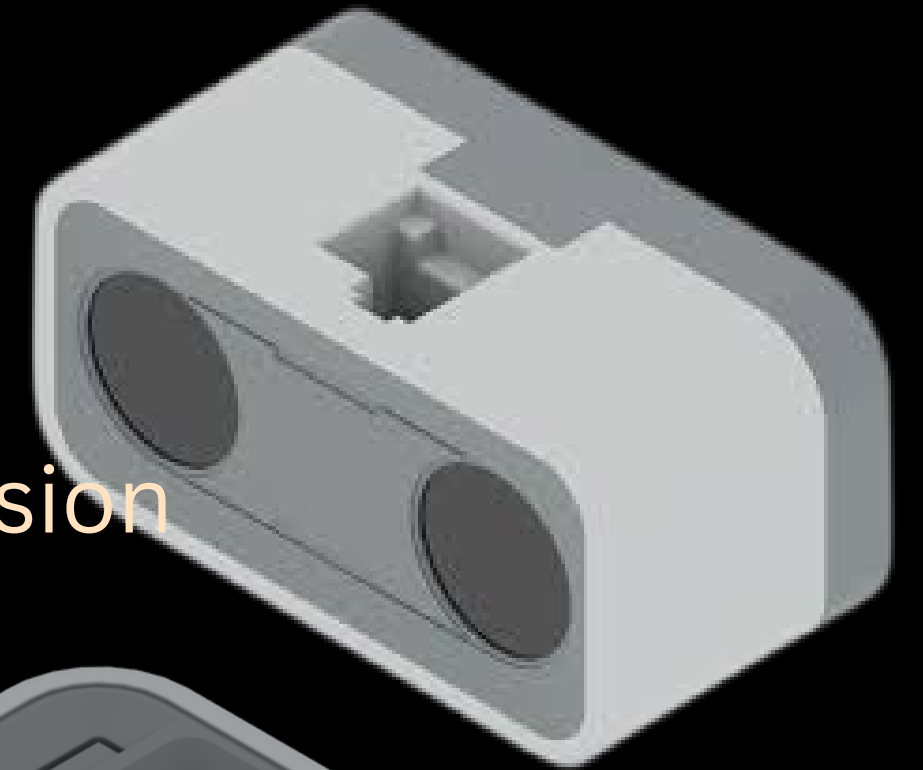
- Senses how far the nearest object is **directly in front of it.**

## How is it used?

- Use this sensor at the front of your robot or assembly to sense how far away a wall is!
  - The wall is always there, so when the robot is near the wall, use this sensor to precisely adjust the position of the robot.

*This sensor isn't very good at sensing field elements!*  
Sometimes, the field element is too small, or the sensor is badly positioned.

Gen 1 Version



Gen 2 Version



# Touch Sensor

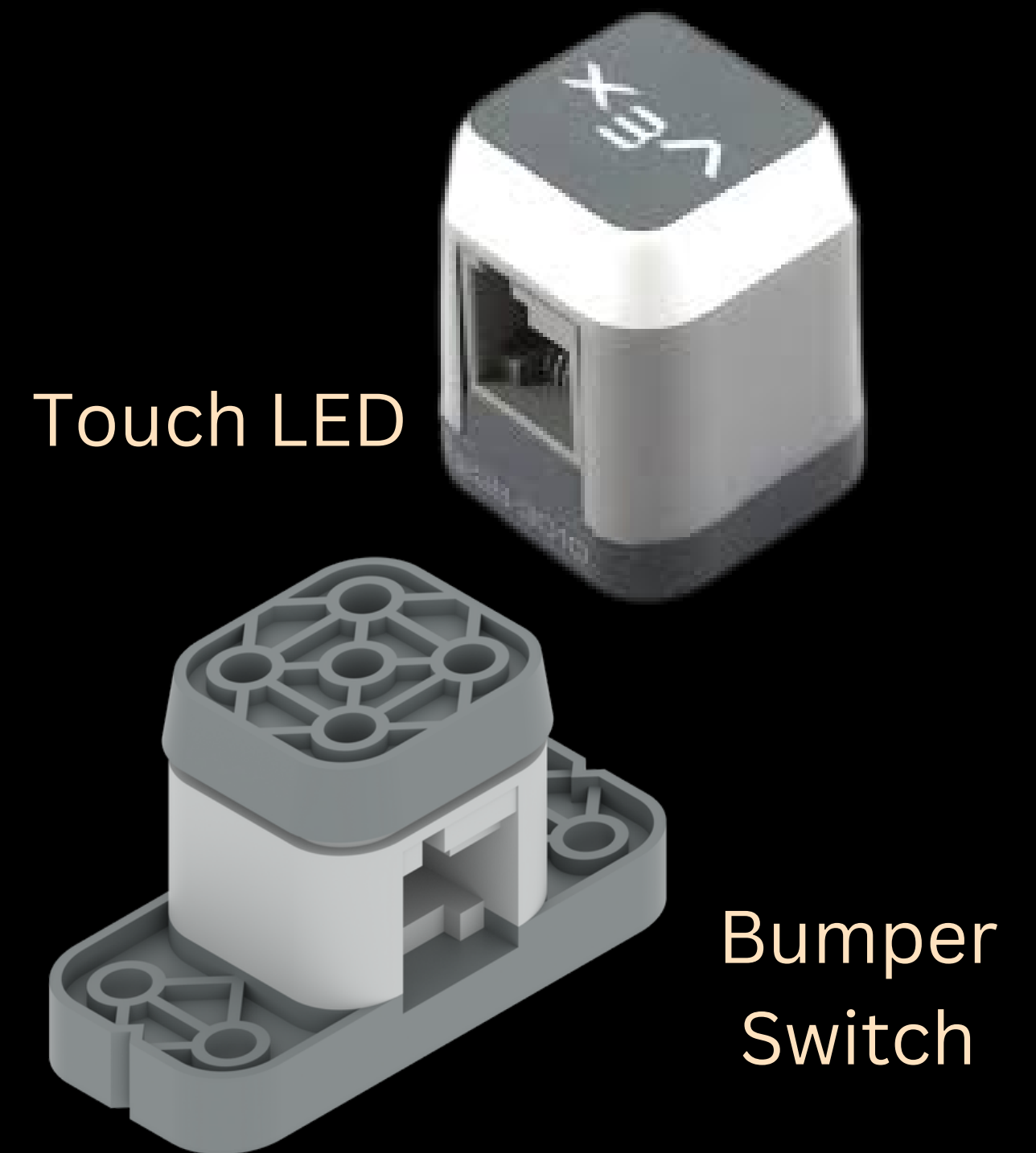
## What does it do?

- Senses touch, or collision.
- Press to activate functions

## How is it used?

- Primarily used to separate modules in a program.
  - When programs are separated into different modules, the LED and bumper switch are used to signify the start of the next module. (As opposed to simply waiting a period of time)

These sensors aren't very good at sensing collisions with field elements, since most elements are not perfectly flat.



# Gyro Sensor!

The image shows a Scratch script for a Gyro Sensor. It starts with a yellow 'when started' block. Below it is a grey comment block that says 'Calibrate and reset the values for the Gyro for EACH USAGE'. The script then consists of three blue blocks: 'calibrate Gyro5 for 2 seconds', 'set Gyro5 heading to 0 degrees', and 'set Gyro5 rotation to 0 degrees'.

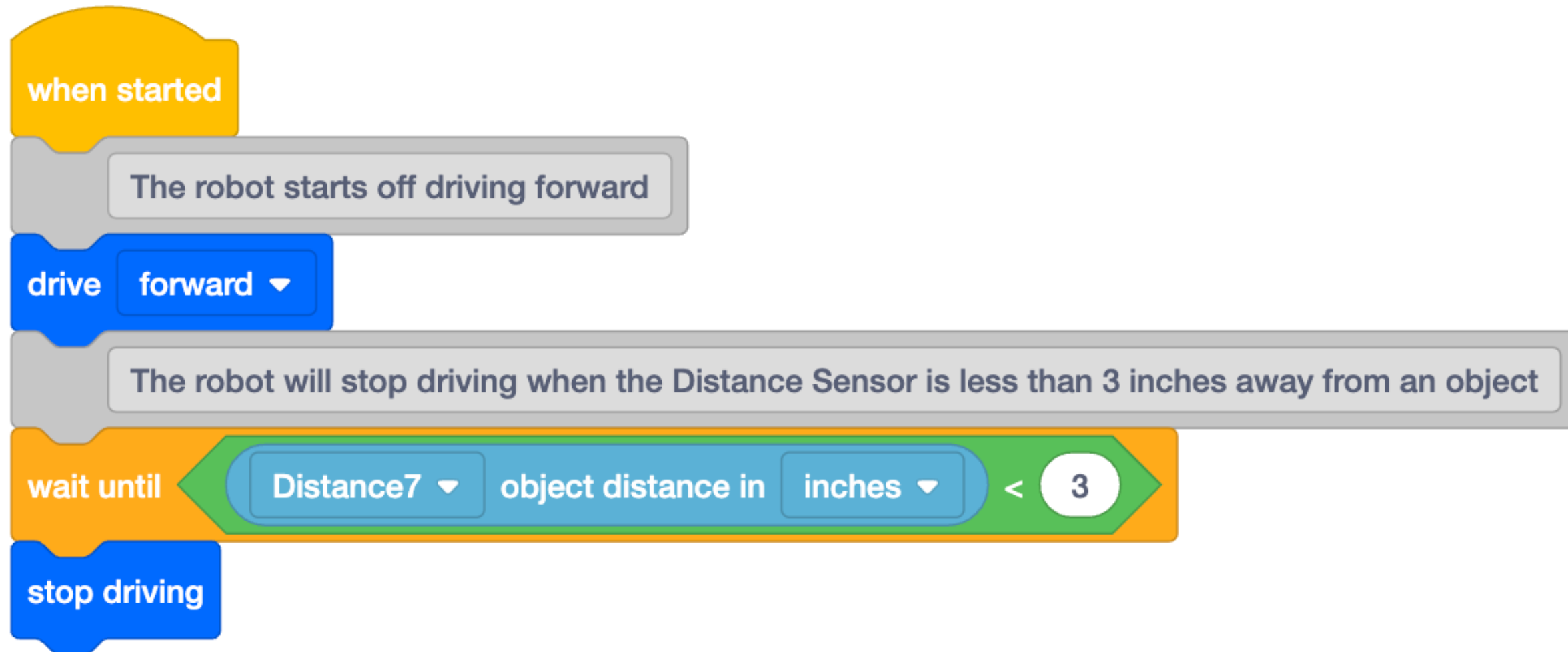
```
when started  
  Calibrate and reset the values for the Gyro for EACH USAGE  
  calibrate Gyro5 for 2 seconds  
  set Gyro5 heading to 0 degrees  
  set Gyro5 rotation to 0 degrees
```

# Color Sensor!

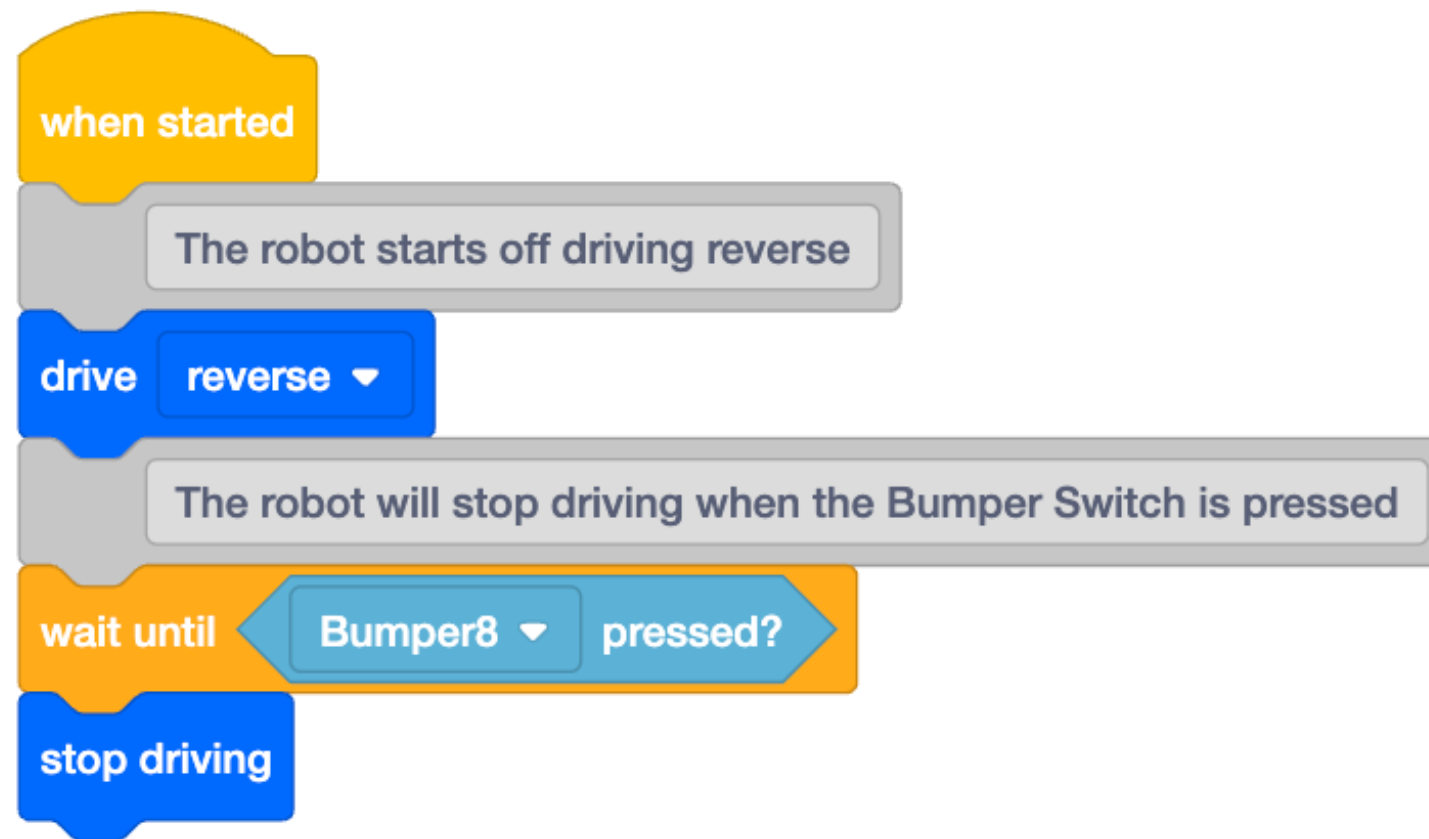
```
when started
  calibrate BrainInertial
  Change the font size to fit on the IQ (2nd generation) Brain's screen
  set font to Prop Medium on Brain
  forever
    if the Color Sensor detects red, the robot will drive forward three inches
      if Color4 detects red ? then
        drive forward for 3 inches
      else
    if the Color Sensor detects blue, the robot will drive backward three inches
      if Color4 detects blue ? then
        drive reverse for 3 inches
      else
    if the Color Sensor detects green, the robot will turn 90 degrees to the right
      if Color4 detects green ? then
        turn right for 90 degrees
      else
    If none of the above colors are detected, the robot will do nothing
      Print if no color is detected
      print No Color Detected on Brain
      wait 0.5 seconds
      Clears the screen if no color is detected
      clear all rows on Brain
      set cursor to row 1 column 1 on Brain
```

```
If the Color Sensor detects yellow, the robot will turn 90 degrees to the left
if Color4 detects yellow ? then
  turn left for 90 degrees
else
```

# Proximity Sensor



# Touch Sensor



Project Name: Bumper Switch

Description:  
This program will drive reverse until the Bumper Switch comes into contact with an object.

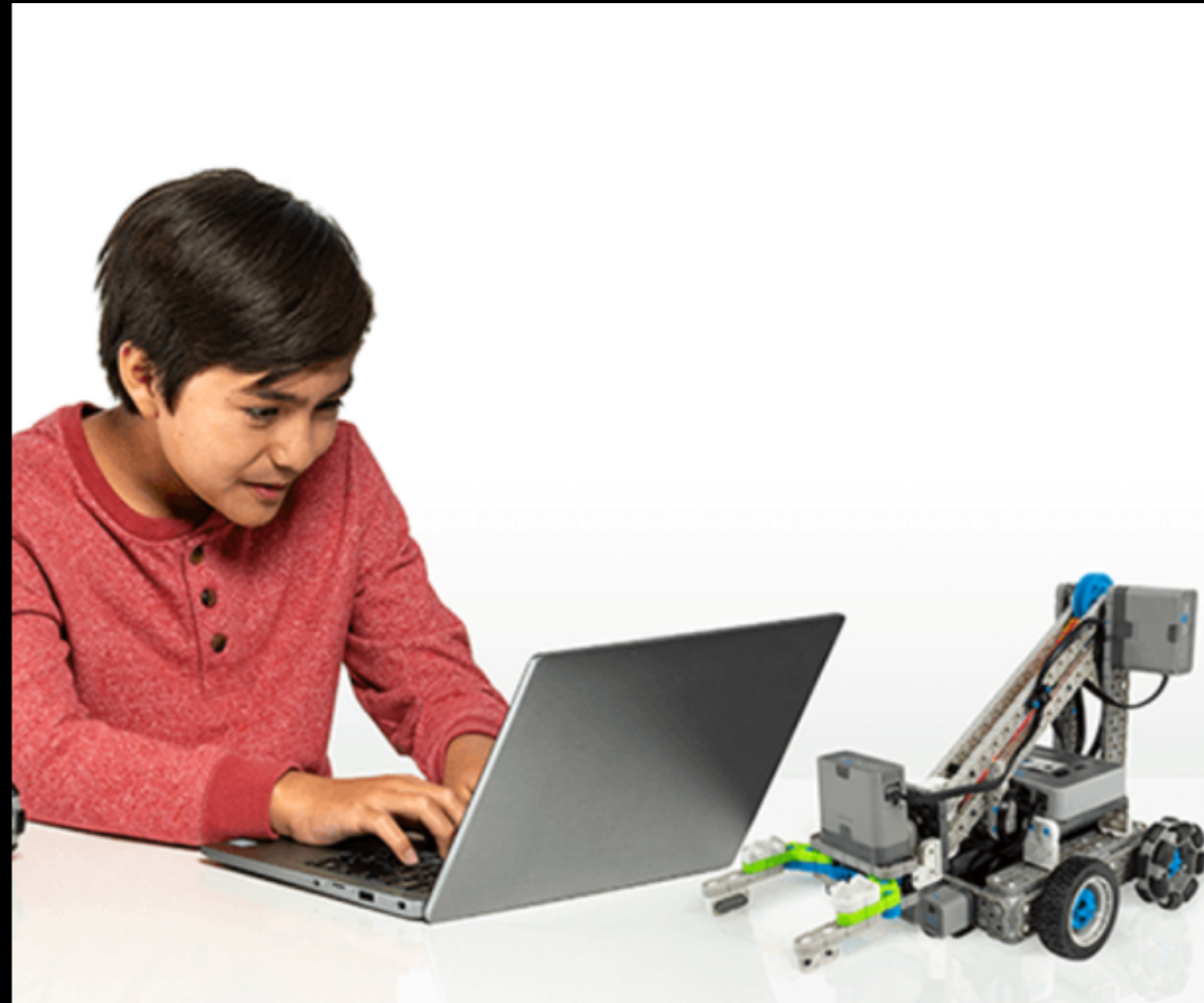
Brain Supported: 2nd generation

Configuration:  
BaseBot (Drivetrain 2-motor, Inertial)  
Bumper in Port 8

# What to do now?

- 1 Write on a piece of paper your plan for autonomous
- 2 Prototype the drive code using the driving functions
- 3 Slowly develop a code (Action by Action) that preforms a singular task reliably





# Before we end!

- Coding is not difficult, it is time-consuming. Do not give up!
- Autonomous can help get you the skills award, which will qualify teams for states, so there is a lot of value!

*Thank  
You*

If you'd like to support,  
connect with me on LinkedIn! =>

