ROBOTIKIDS TECHNICAL MANUAL

RobotiKids

For Association Anoual

Erin Gowaski, Hannah Jaworski,
Nathan Kumar, Josh Palmer

WPI

ROBOTIKIDS - ROBOTICS CURRICULUM FOR MIDDLE SCHOOL STUDENTS
TECHNICAL MANUAL
2022

Technical Manual Purpose:
This is a supplementary manual that documents the use of VEX IQ robots in the RobotiKids program. This technical manual was referenced in the LESSON MANUAL submitted to Association Anoual.

The program of RobotiKids was developed, created and tested with the use and assistance of the various VEX resources. This technical manual offers insight into the models of robots used for each lesson, the code used for each lesson and advice that relates accordingly. Note that there are various ways to accomplish a task with the code and robot, so answers are not limited to this manual's answers. The intent of this manual is to offer guidance and support to VEX code and hardware in relation to the RobotiKids program. While the program can be supplemented with other robotic kits we highly recommend VEX for its ease of use and its relationship with RobotiKids.

Additional Resources
These resources aided the development of RobotiKids and can offer further help around issues that may arise. We recommend becoming acquainted with VEX before facilitating the program.
- VEX IQ Robotics Education Guide
- VEX IQ Teacher Supplement
- VEX IQ Video Demos
- VEXcode IQ Program Download

Two Manuals included in the VEX IQ Kits

The Control System User Guide - Explains the connecting and software components of the kit, for the program the most important components are:

- Batteries and charging (blue)
- Connecting the smart devices (green)
    - Sensors
- The robot brain (red)
- Built in Programs (optional) (purple)
    - Search Patterns

From VEX IQ Kit Control System User Guide

Build Instructions -Detailed step-by-step construction of the hardware pieces of the kit, for the program the most important components are:

- Building the basebot
- Construction of arm
- Construction of claw
- Attachment of smart devices

VEX IQ
Build Instructions
Instructions de Construction
Includes plans for all four robots
Inclus les plans pour les quatre robots

Hardware Information

We recommend using the basebot that is outlined and explained in the hard copy of the VEX IQ Kit that provides step-by-step instruction for construction.

The kit includes many parts to construct the basebot or build a new robot by adding supplemental parts to the basebot.
The pieces included in the VEX IQ kit are illustrated in a poster that outlines their real size, dimensions, product number and count in the kit.
The parts can be organized into four categories:

Brain and Smart Devices

- The robot brain
- Battery and charging
- Sensors and radios
- Cables

Movement pieces

- Rubber bands
- Pulleys
- Gears
- Shafts and shaft accessories

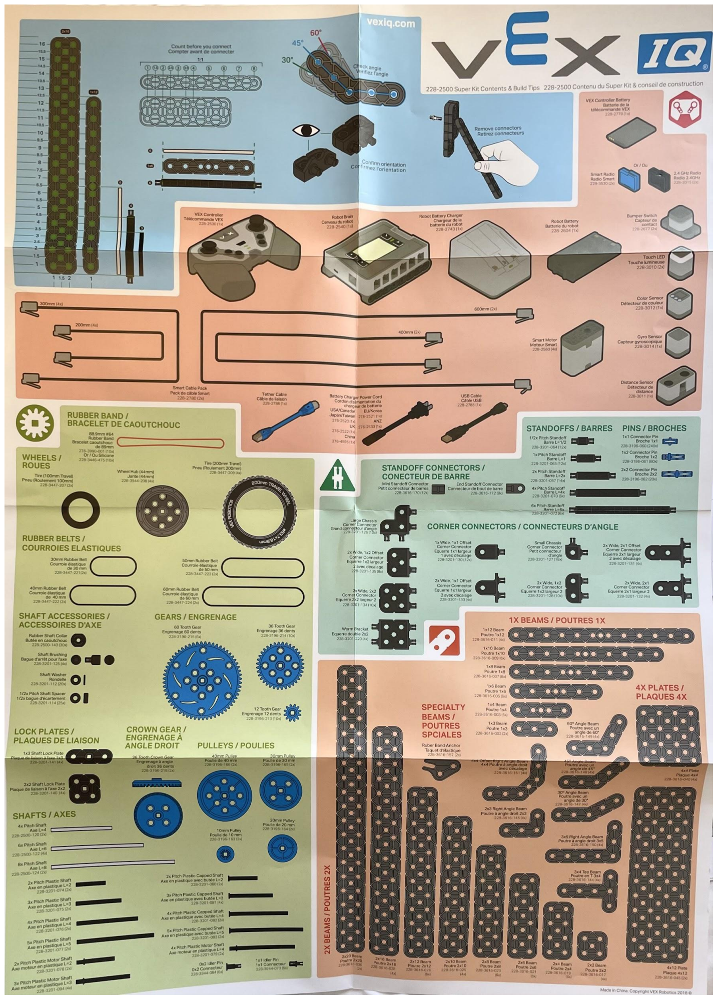
Connectors

- Standoffs and standoff connectors
- Corner connectors
- Pins

Bases

- Various 1X beams
- Various 2X beams
- Plates
- Specialty beams

The pieces snap together and have connection points within them that offer security but also ease of release.



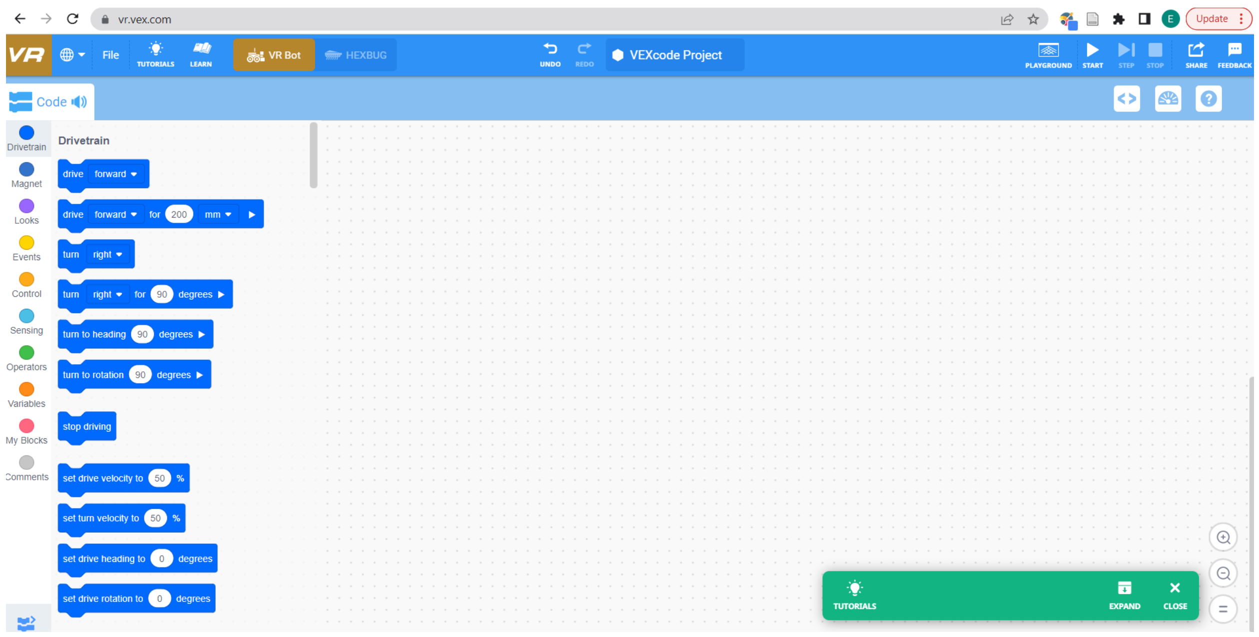
vexiq.com
VEX IQ
RUBBER BAND / BRACELET DE CAOUTCHOUC
WHEELS / ROUES
RUBBER BELTS / COURROIES ELASTIQUES
SHAFT ACCESSORIES / ACCESSOIRES D'AXE
GEARS / ENGRENAGE
LOCK PLATES / PLAQUES DE LIAISON
CROWN GEAR / ENGRENAGE À ANGLE DROIT
PULLEYS / POULIES
SHAFTS / AXES
STANDOFFS / BARRES
PINS / BROCHES
STANDOFF CONNECTORS / CONNECTEUR DE BARRE
CORNER CONNECTORS / CONNECTEURS D'ANGLE
1X BEAMS / POUTRES 1X
SPECIALTY BEAMS / POUTRES SPCIALES
4X PLATES / PLAQUES 4X
2X BEAMS / POUTRES 2X

From VEX IQ Kit Hardware Illustration

VEXcode VR Simulator

The VEXcode Virtual Robot (VR) simulator is an online resource that allows the user to use block coding to simulate actions of a VEX robot.
The following link will bring you to the VEXcode VR online platform: https://vr.vex.com/

Having followed the attached link you will be welcomed to the following page:



At the top, is a toolbar.

This toolbar includes a tab called ***tutorials***. These tutorials will help you get set up with the VEXcode VR platform.

The code functions are organized on the left hand portion of the screen and can be dragged and dropped into an open grid.

The playground button in the top corner opens a virtual robot to mimic the code that was constructed. The playground has a variety of areas and grounds to experiment on.

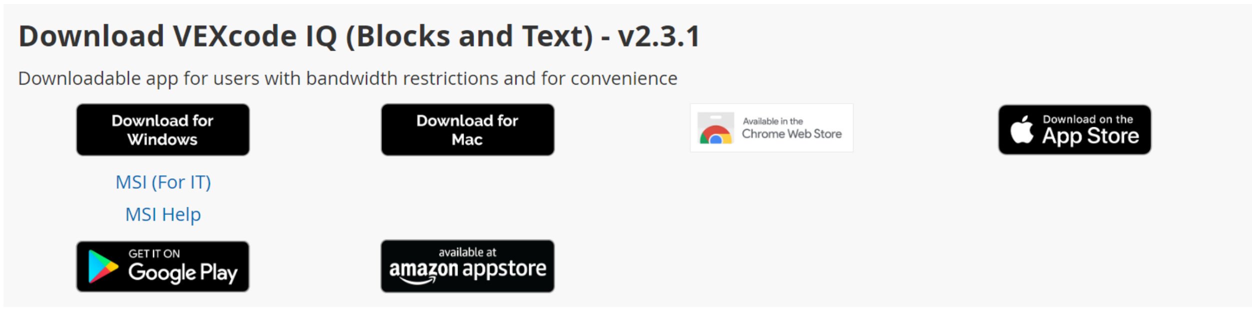
TIPS:

- VEXcode VR automatically saves the progress.
- There is an undo, a play and a stop button in the top banner strip.
- The code must always be started with a start function.

VEXcode IQ

VEXcode IQ is a program of block code that can be downloaded directly to the VEX IQ robot brain. This software is similar to the VEX VR simulator. Follow the Steps listed to download software

1. Visit: **https://www.vexrobotics.com/vexcode-download**
2. Scroll to **VEXcode Blocks**
3. Select **IQ**
4. Select the format that is applicable for your device



Download VEXcode IQ (Blocks and Text) - v2.3.1

Downloadable app for users with bandwidth restrictions and for convenience

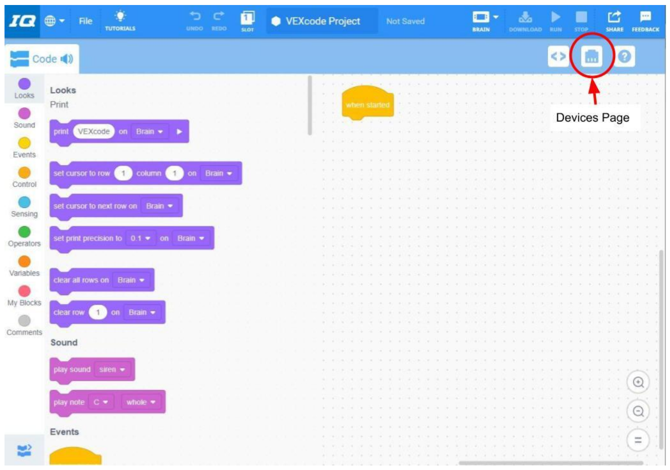
Download for Windows
Download for Mac
Available in the Chrome Web Store
Download on the App Store
MSI (For IT)
MSI Help
GET IT ON Google Play
available at amazon appstore

Setting up the software:

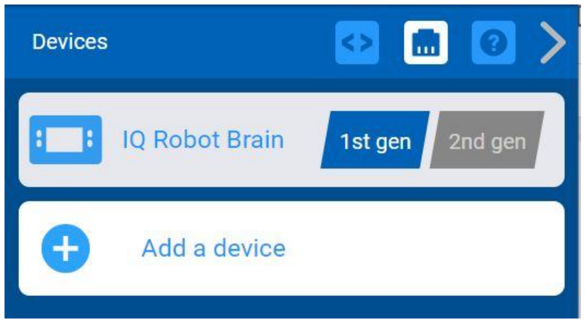
1. Open the downloaded software
2. Turn on your robot/robot brain
3. Plug in the programming cable from the robot to the computer

How to add a device to a port in VEXcode IQ

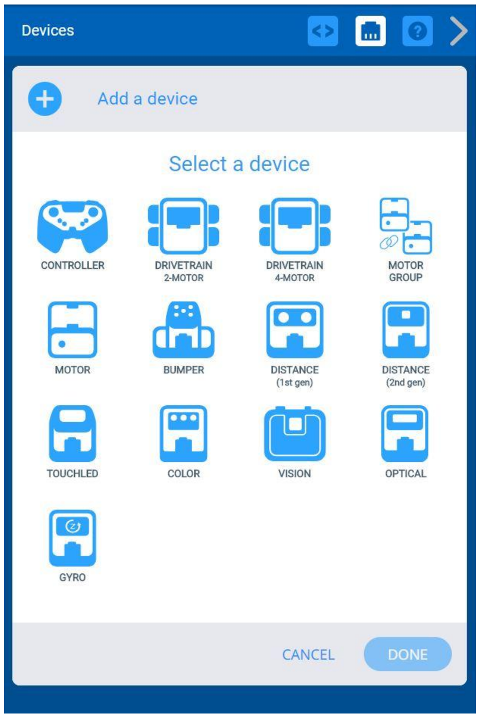
1. Once you have opened VEXcode IQ, select the devices page



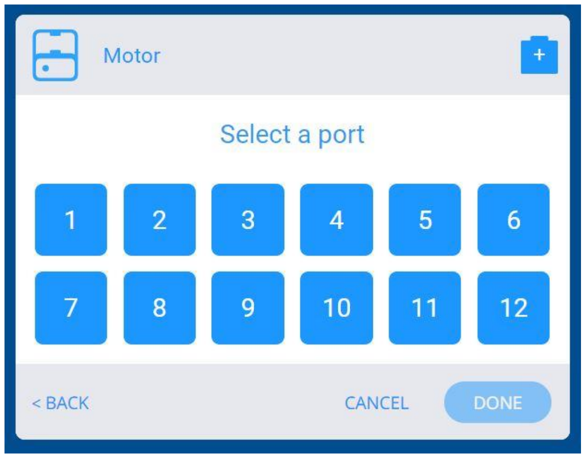
2. Add a device



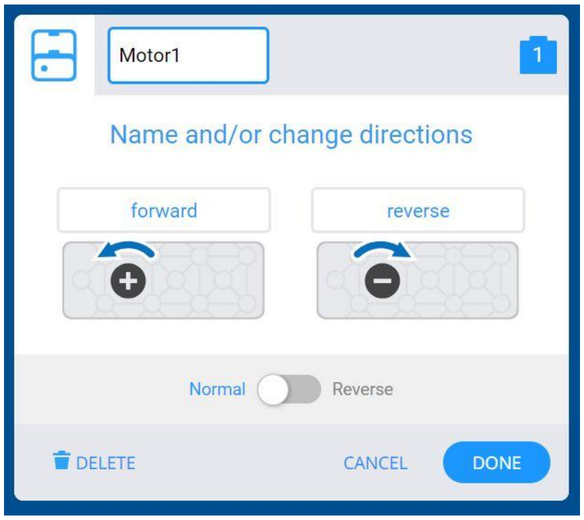
3. You will need to add: a two motor drive train, a motor for both the arm and claw, a bumper, a distance (1st gen) sensor, a color sensor, and the gyro



4. Click the port number that the motor is plugged into on the physical robot.

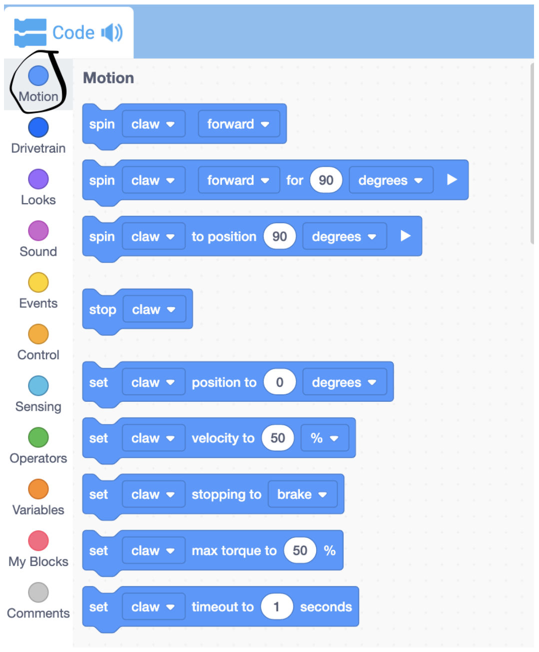


5. You can rename the motor. Then press "DONE"



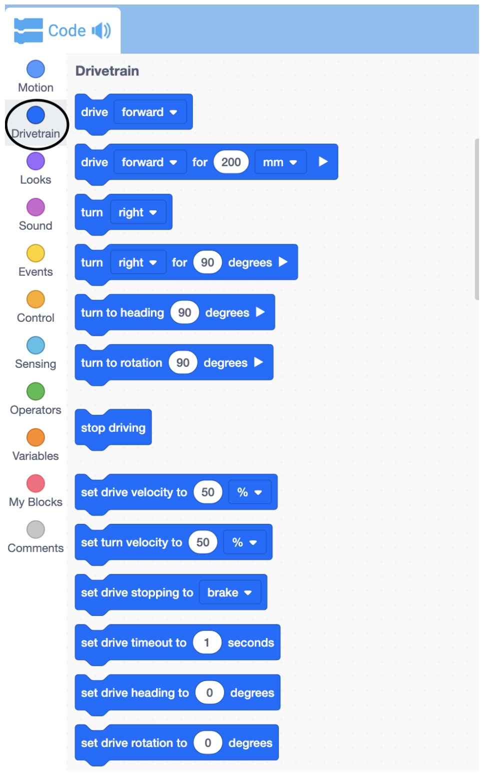
Reviewing Code Sections:

Motion:
The motion section of the VEXcode IQ includes block code that deals with the <u>movements of the arm and claw</u>. You can lift/lower the arm as well as open/close the claw of the robot. *Note that the name of the dropdown 'claw' may be different depending on what you name the motor.*


Code
Motion
Drivetrain
Looks
Sound
Events
Control
Sensing
Operators
Variables
My Blocks
Comments
Motion
spin claw forward
spin claw forward for 90 degrees
spin claw to position 90 degrees
stop claw
set claw position to 0 degrees
set claw velocity to 50 %
set claw stopping to brake
set claw max torque to 50 %
set claw timeout to 1 seconds

DriveTrain:

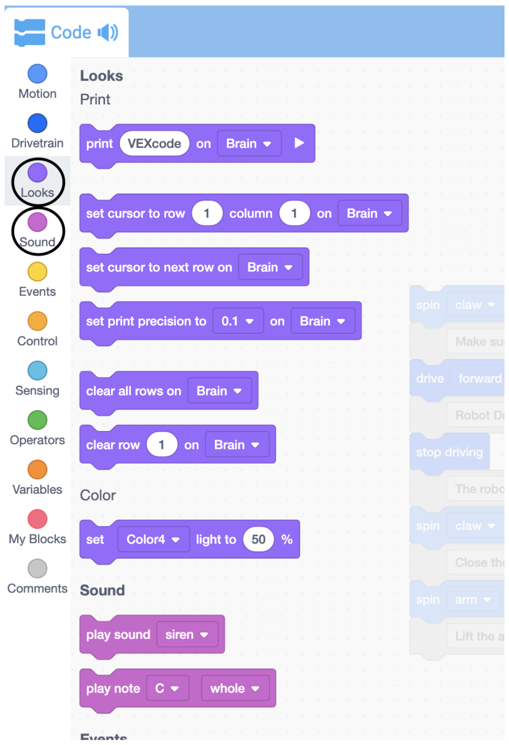
The drivetrain section of the VEXcode IQ includes code that will *drive* the robot. This includes setting the speed and distance the robot can travel.


Code
Motion
Drivetrain
Looks
Sound
Events
Control
Sensing
Operators
Variables
My Blocks
Comments
Drivetrain
drive forward
drive forward for 200 mm
turn right
turn right for 90 degrees
turn to heading 90 degrees
turn to rotation 90 degrees
stop driving
set drive velocity to 50 %
set turn velocity to 50 %
set drive stopping to brake
set drive timeout to 1 seconds
set drive heading to 0 degrees
set drive rotation to 0 degrees

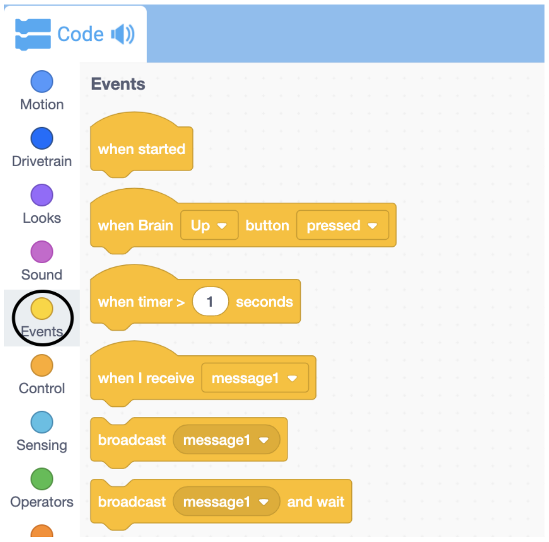
Looks & Sound:

The looks section on VEXcode IQ will allow for text to be *displayed* on the brain of the robot.

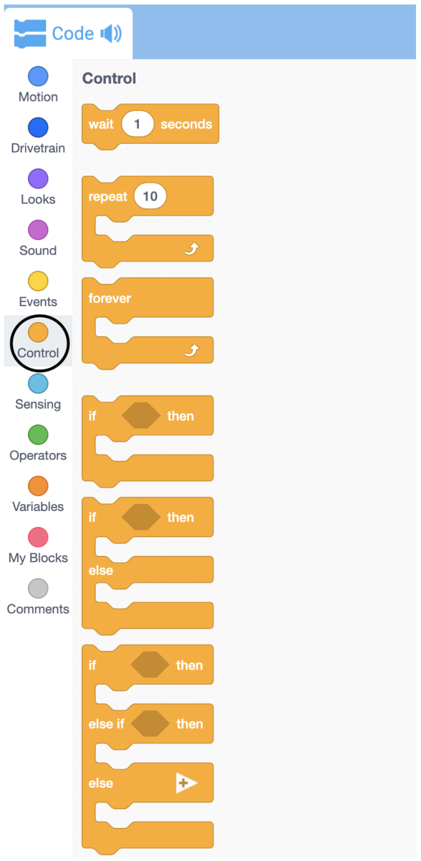
The sound section allows for audio to be played.



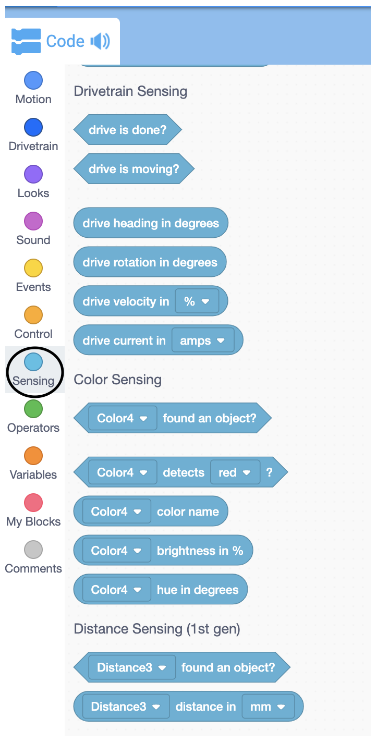
Events:
The events section of the VEXcode IQ included blocks that will begin code and broadcast events. The most important block in the event section is “*when started*.” It indicates the start of a program and proceeds to execute the code in the program.



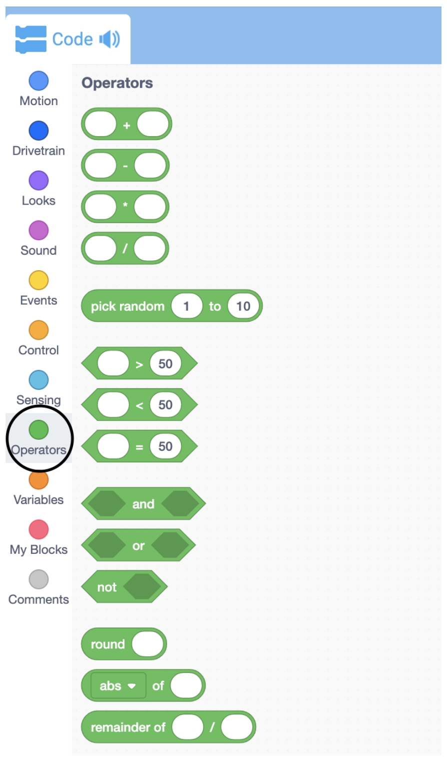
Control:
The control section is where conditional statements are found.



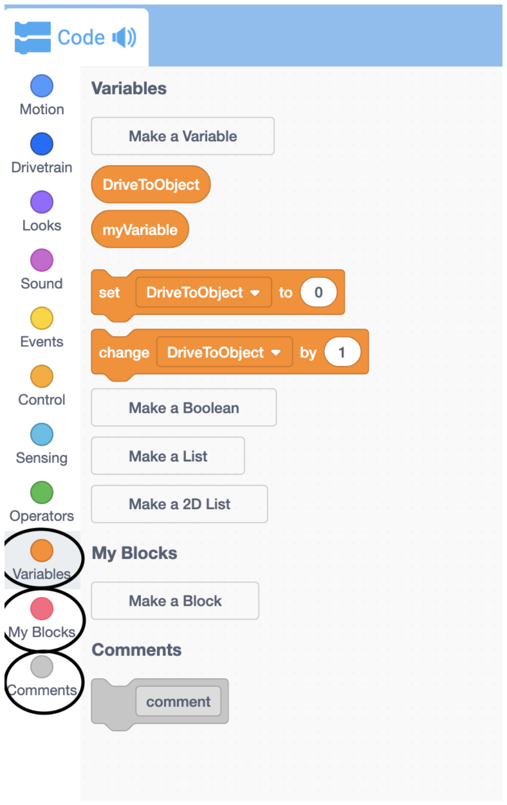
Sensors:
The sensor section is where blocks that let the robot detect objects and their color, as well as many other functions. *Note the names in the picture, "color 4" and "distance 3" will be different depending on what you name the motor.*



Operations:
The operations section is useful with conditionals, especially, *and, or* and *not*. These are often used within an *if statement* which are found in the control section.



Variables & My Blocks & Comments:
The variable section can be used to create a custom variable.
My blocks can be used to store a custom block of code.
Comments are used to describe the contents of code.



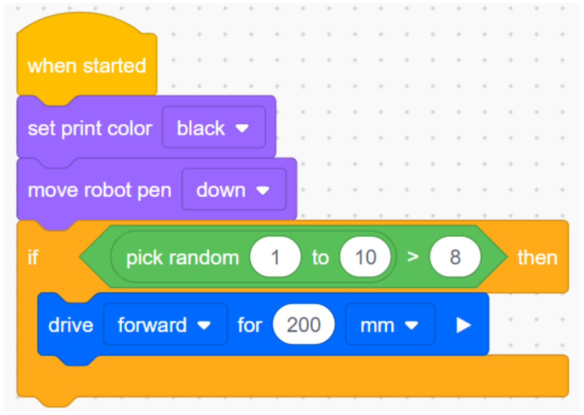
Lesson 2: Introduction to Coding

VEX offers two platforms for block coding. The first, VEXcode VR (Virtual Robot) operates solely as a virtual robot and does not coordinate with the VEX kit. It allows students to drag and drop code into the grid and then see the motion of the virtual robot on the platform. This translates into VEXcode IQ which is a parallel format, except the code can be delivered to the constructed robot. **Note:** that the following code will only work in the VR simulator. However, the concepts and logic can be used in VEXcode IQ.

The following examples use basic movement forwards and backwards and turning, this can be used as an introduction to movement seen in lesson 3.
If you would like to implement the following code open the "Art Canvas" playground in VEXcode VR to see the code work to full potential.

If/Then Statements:
If/Then states are explained in the Lesson Manual.

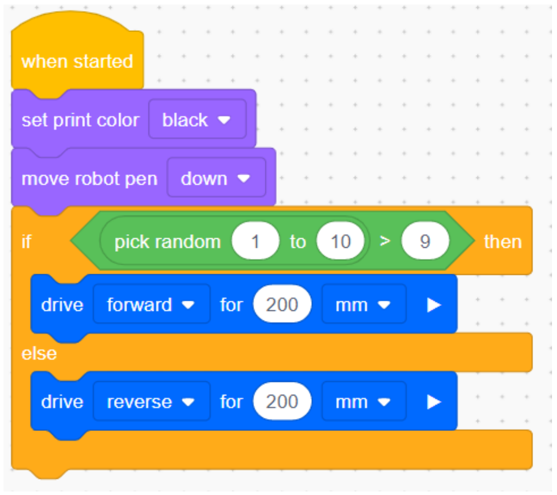
When the play button is pressed the pen on the robot will be set to black and placed on the canvas. The computer will randomly select a number 1 to 10. If the number is greater than 8 the robot will drive forward 200 mm. If the computer does not select a number greater than 8, the robot will not move and the loop will terminate.



```
when started
set print color black
move robot pen down
if pick random 1 to 10 > 8 then
    drive forward for 200 mm
```

If/Then…Else Statements:
If/Then…else statements are explained in the Lesson Manual

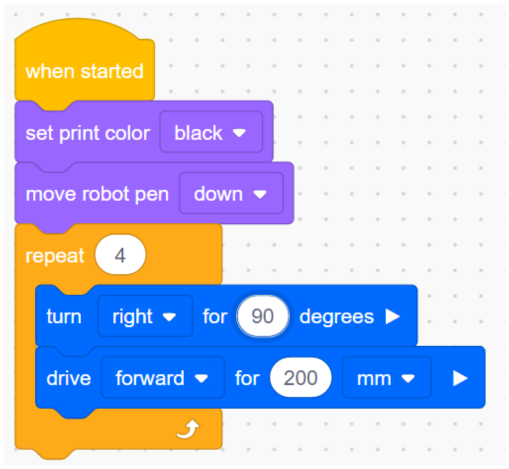
When the play button is pressed the pen on the robot will be set to black and placed on the canvas. The computer will randomly select a number 1 to 10. If that number is greater than 9, the robot will drive forward 200 mm leaving a black line drawn. If that condition is not met, the robot will drive backwards for 200 mm leaving a black line drawn.



```
when started
set print color black
move robot pen down
if pick random 1 to 10 > 9 then
    drive forward for 200 mm
else
    drive reverse for 200 mm
```

For Loop:
For loops are explained in the Lesson Manual. For loops are referred to as "repeat" in VEXcode software.

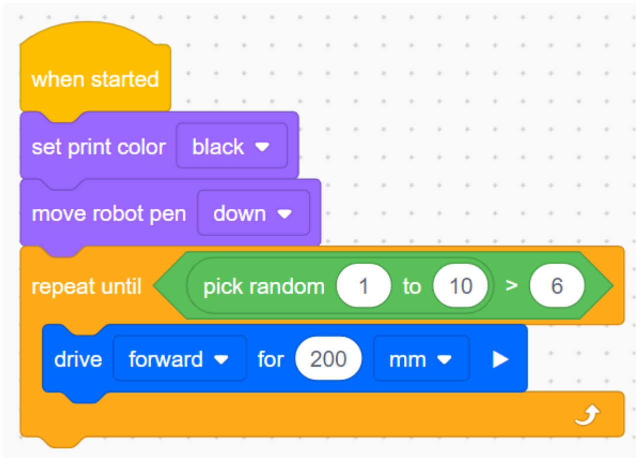
When the play button is pressed the pen on the robot will be set to black and placed on the canvas. The robot will make a 90 degree right turn and move forward 200 mm. That sequence will repeat 4 times.
*this will draw a square!



```
when started
set print color black
move robot pen down
repeat 4
    turn right for 90 degrees
    drive forward for 200 mm
```

Repeat Until Loop:
Repeat until loops are explained in the Lesson Manual.

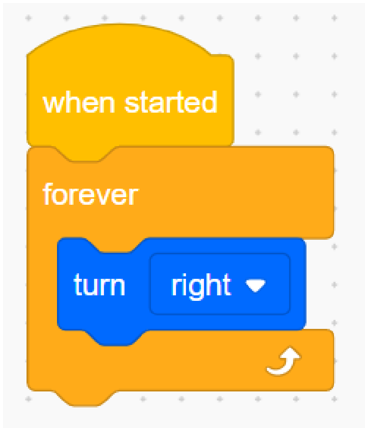
When the play button is pressed the pen on the robot will be set to black and placed on the canvas. The robot is then instructed to move forward in increments of 200 mm until the computer randomly selects a number larger than 6, from the range of 1-10. Once a number larger than 6 has been selected the "repeat until" loop terminates and the robot stops moving.



```
when started
set print color black
move robot pen down
repeat until pick random 1 to 10 > 6
    drive forward for 200 mm
```

Forever Loop:
Forever loops are explained in the Lesson Manual.

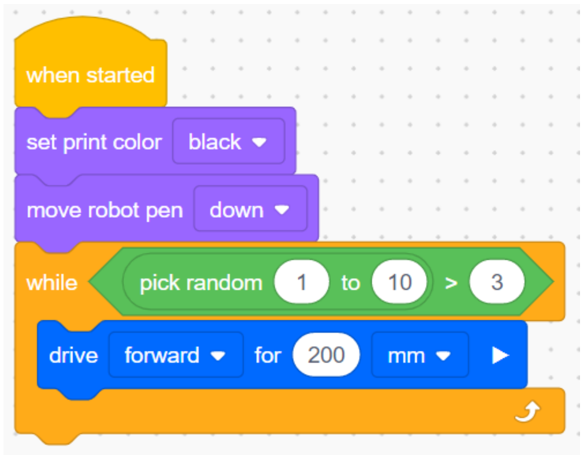
When the play button is pressed the robot will turn right (varying in degrees) forever. This loop will not end, until instructed to do so by a *break* or end project



```
when started
forever
    turn right
```

While Loop:
While loops are explained in the Lesson Manual.

When the play button is pressed the pen on the robot will be set to black and placed on the canvas. The robot will then move forward until the computer selects a number that does not fit the condition of being greater than 3.



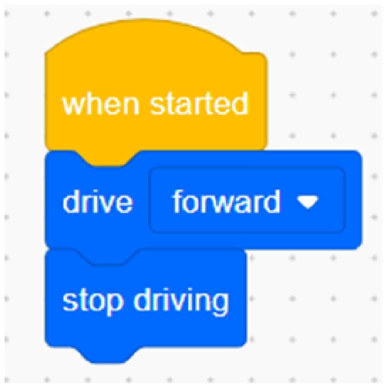
```
when started
set print color black
move robot pen down
while pick random 1 to 10 > 3
    drive forward for 200 mm
```

Lesson 3: Movement and Turning

This lesson is applicable for both the VEXcode VR and the VEXcode IQ platforms.
The following lesson outlines the code and necessary steps it takes to code for the movement of forwards, backwards, and turning left and right.
With each lesson students, and yourself, should feel welcome to play around with other values and formats.

Movement Forwards:

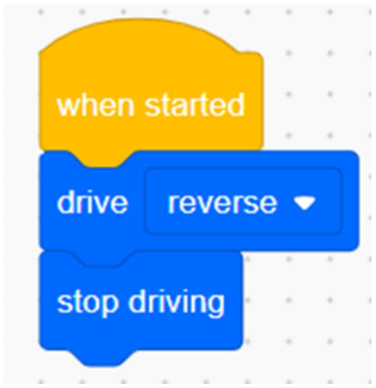
Basic movement forwards with no defined parameters. Once the play button is pressed the robot will move forward infinitely.



```
when started
drive forward ▾
stop driving
```

Movement Backwards/reverse:

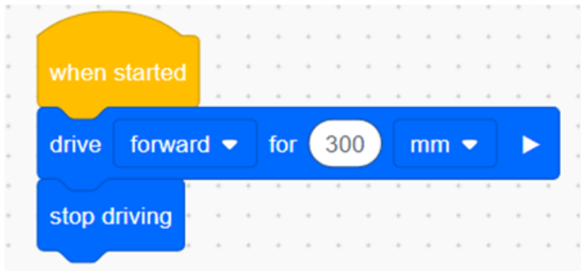
Basic Movement backwards with no defined parameters. Once the play button is pressed the robot will move backwards indefinitely.



```
when started
drive reverse ▾
stop driving
```

Movement forward/backwards with defined distance parameter:

Movement forwards for 300 mm. The robot will stop driving once 300 mm have been traveled.



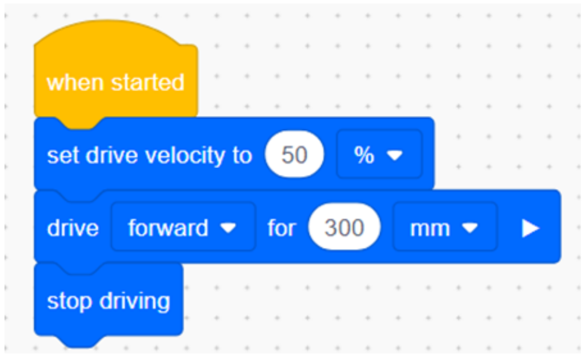
```
when started
drive forward ▾ for 300 mm ▾ ▶
stop driving
```

Movement forward/backwards at velocity measured in percentage:

***NOTE**: Throughout this technical manual, velocity and speed will be interchangeably.*

Movement forward for a defined distance parameter (300mm) at 50% velocity.

The velocity of this robot is undefined, however students should feel welcome to see the robot move at 100% percent speed and a slower rate.

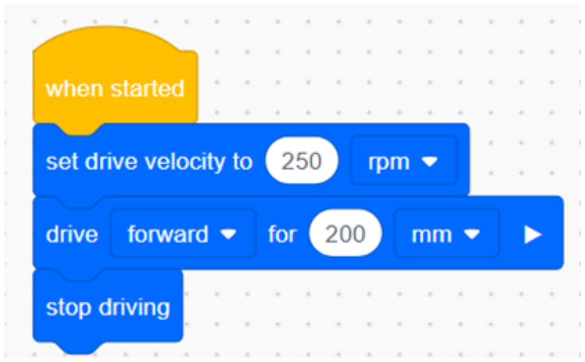


```
when started
set drive velocity to 50 %
drive forward for 300 mm ▶
stop driving
```

Movement forward/backwards at rpms:

Movement forward for a defined distance parameter (200 mm).

Rpms are explained in the lesson manual. Students should feel welcome to see how fast 100 rpms is and how fast 500 rpms is.

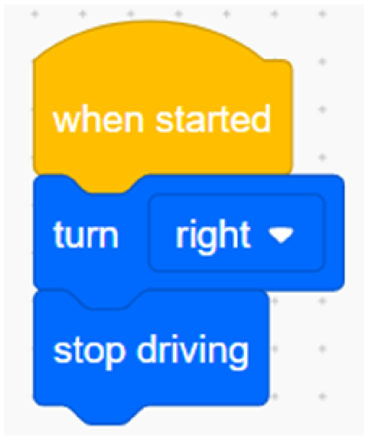


```
when started
set drive velocity to 250 rpm
drive forward for 200 mm ▶
stop driving
```

Turning Right/Left:

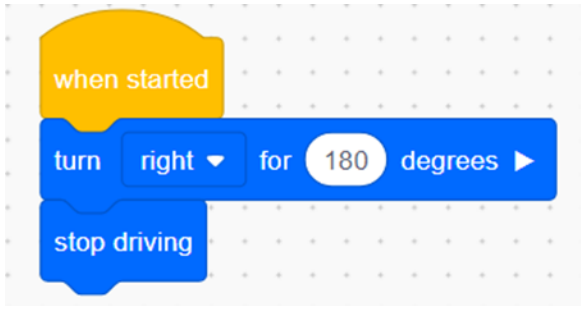
This movement to turn right uses the gyroscope. Use the lesson manual to describe what the gyroscope does. This movement will automatically turn the robot 90 degrees right.

By selecting the drop-down arrow next to "right" students can select "right" or "left" directions. This code can be used for either right or left turning



```
when started
turn right
stop driving
```

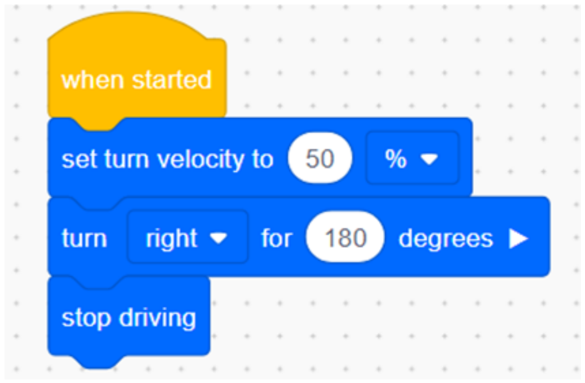
Turning using Degrees:



This movement allows the robot to turn in increments other than 90 degrees. Lessons on degrees are detailed in the Lesson Manual. The gyroscope is used for this function: see Lesson Manual for explanation.

Turning right 180 degrees.

Turning at Velocity:



NOTE*: Throughout this technical manual, velocity and speed will be interchangeably*

This movement allows the robot to turn at a given speed. The turn velocity is only defined by percentage.

The velocity of this robot is undefined, however students should feel welcome to see the robot move at 100% percent velocity and a slower rate.

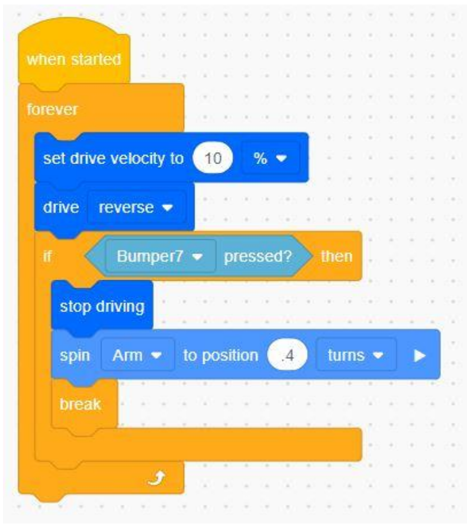
Lesson 4: Sensors

This lesson is applicable for both the VEXcode VR and the VEXcode IQ platforms.
The following lesson outlines the code and necessary steps it takes to use a variety of sensors for robot movement and automation driving.
With each lesson students, and yourself, should feel welcome to play around with other values and formats.

Bumper Sensor:

The VEX Instruction booklet details how to configure the bumper sensor. This is intended to be a *demonstration* of how the sensor works on a basic level.

The bumper sensor is explained in the Lesson Manual.



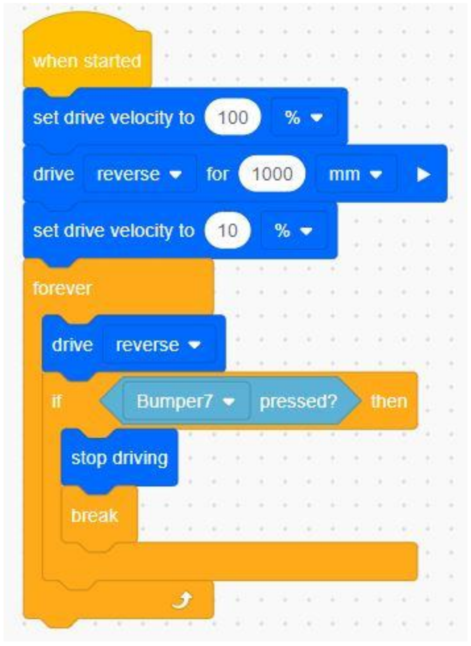
```
when started
forever
    set drive velocity to 10 %
    drive reverse
    if Bumper7 pressed? then
        stop driving
        spin Arm to position .4 turns ▶
        break
```

Water Bottle Activity:

The water bottle activity is outlined in the Lesson Manual.

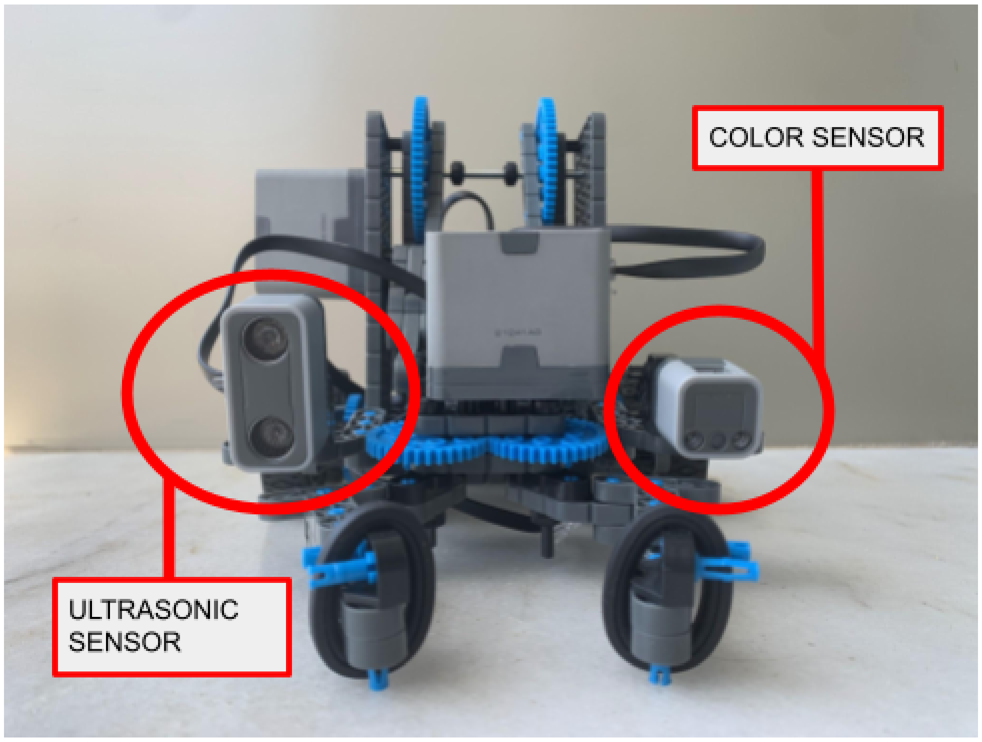
The robot will set its velocity to 100% and move backwards 1000 mm. Once that distance is reached, it will lower its velocity to 10%. The robot will continue to move backwards until the bumper sensor is pressed. When the sensor is pressed the robot will stop.

Students should feel welcome to manipulate the code to make it more efficient or complicated. This may be necessary in some cases depending on how far the bottle is away from the robot.

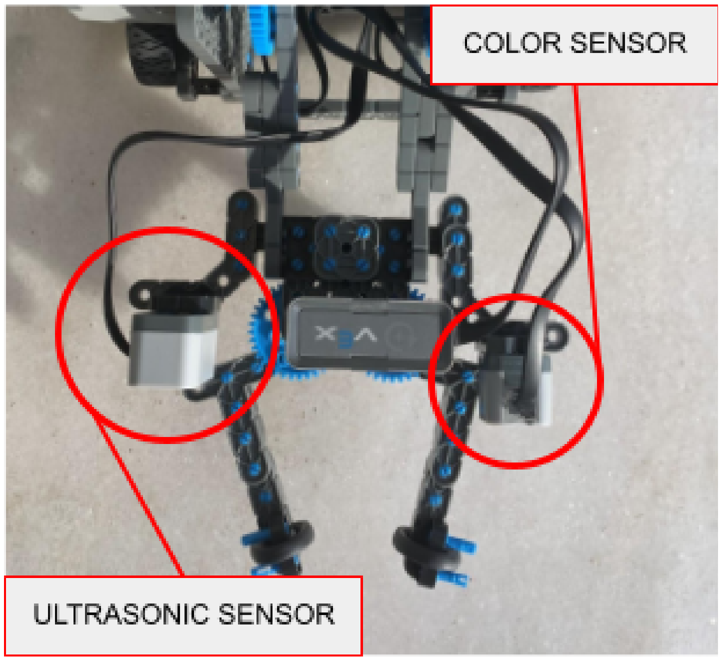


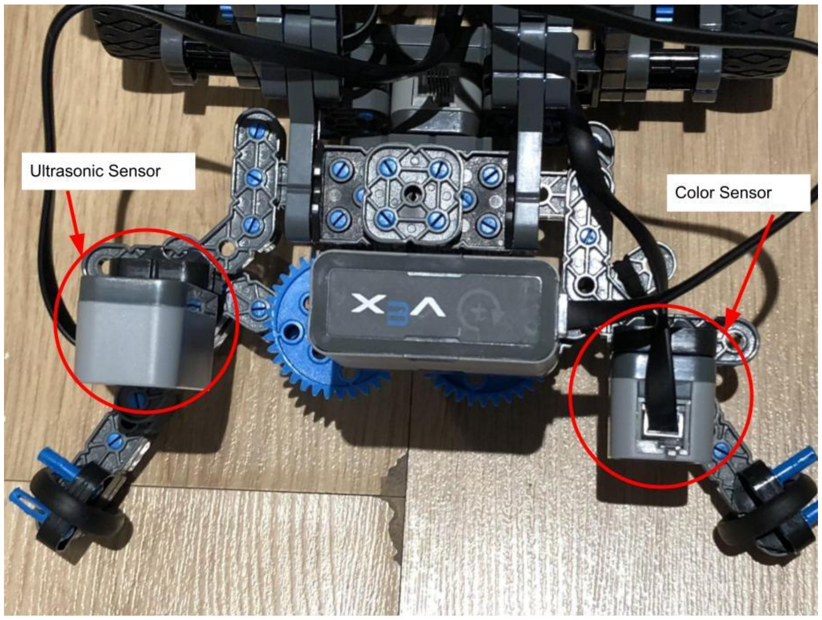
```
when started
set drive velocity to 100 %
drive reverse for 1000 mm ▶
set drive velocity to 10 %
forever
    drive reverse
    if Bumper7 pressed? then
        stop driving
        break
```

The following shows how to set up the color and ultrasonic (distance) sensors. Note that both are attached by the same bracer piece (4x4 offset right angle beam)

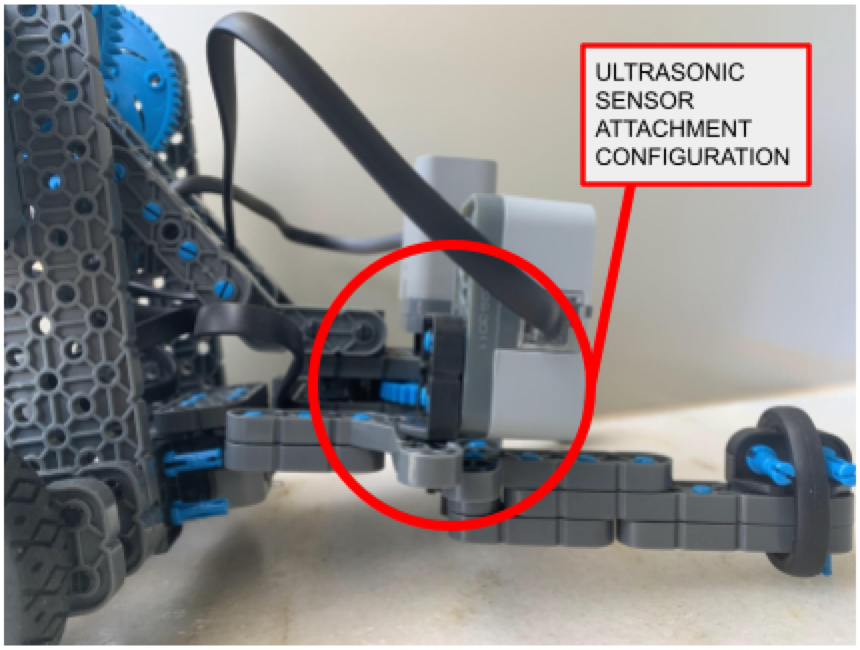


Location of Ultrasonic and Color Sensors

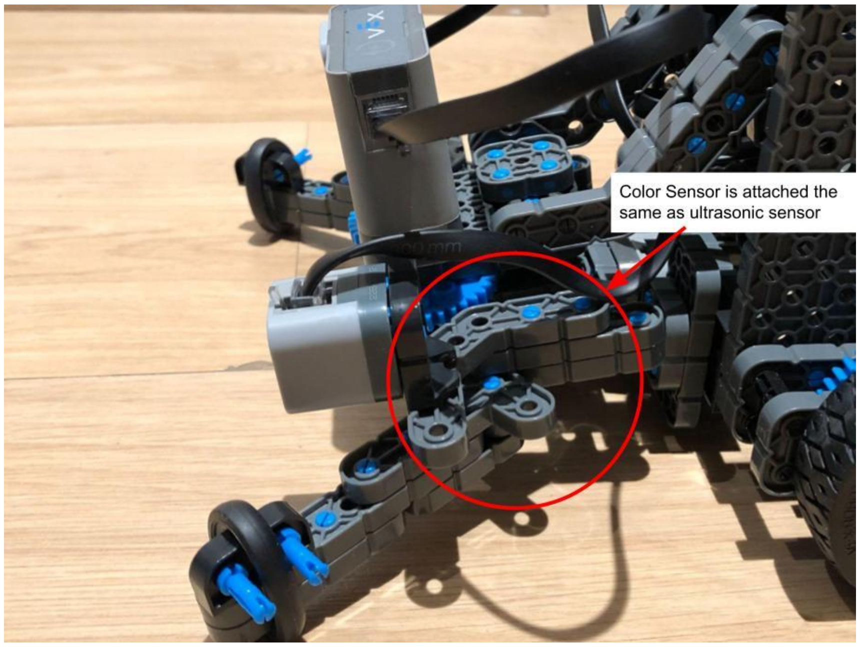




Overhead view of Ultrasonic and Color sensors



Attachment Configuration of Ultrasonic sensor

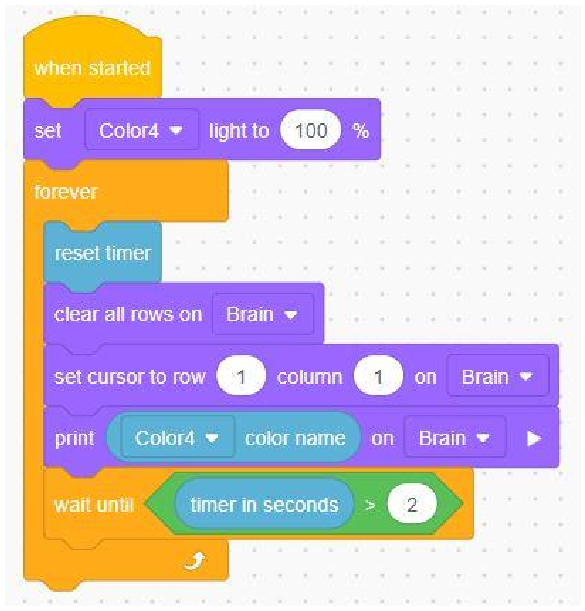


Attachment Configuration of Color Sensor

Color Sensor:

The color sensor is explained in the Lesson Manual.

When the play button is pressed the robot will turn on its light to 100%. The robot will follow the next sequence forever: it will reset its internal clock, clear all the rows on the robot brain, set the cursor to row 1 column 1 on the brain, and print the color that the color sensor sees on the display of the brain. Then it will wait until the timer is greater than 2 in seconds. It will continue to do this process. This is to help determine if what we see is the same color as the robot sees.

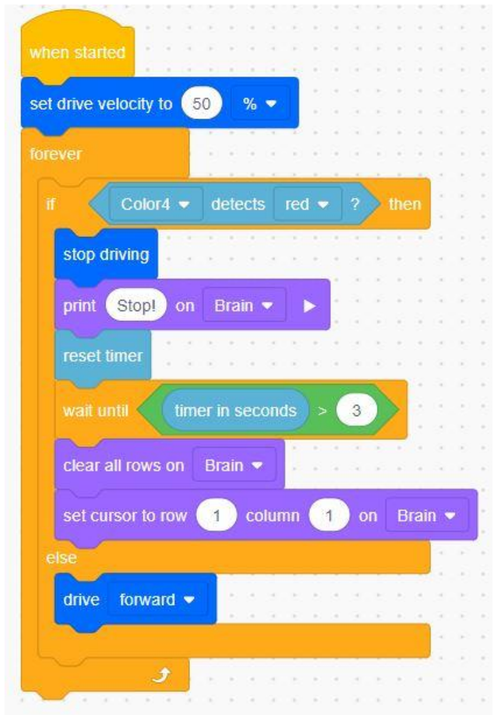


```
when started
set Color4 light to 100 %
forever
    reset timer
    clear all rows on Brain
    set cursor to row 1 column 1 on Brain
    print Color4 color name on Brain ▶
    wait until timer in seconds > 2
```

Red Light Demo:

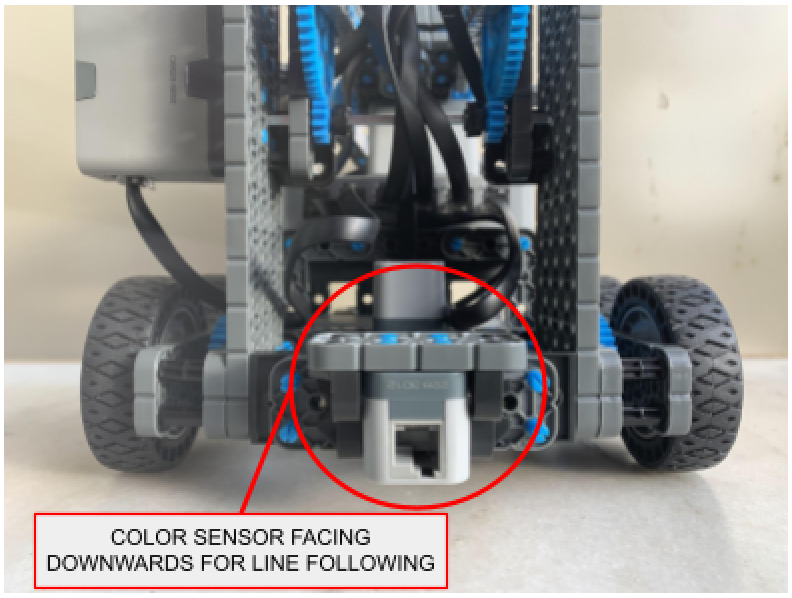
The red light demo is detailed in the Lesson Manual

When the play button is pressed the robot will set its velocity to 50 %. If the robot's color sensor detects red then the robot will stop driving and print "stop!" on the brain display and reset the timer. The robot once it has seen red and stopped driving will wait until the timer has passed 3 seconds then clear the display and check again. This is meant to be fun and interactive for the students but also to teach them a bit more about the sensor and how it works.

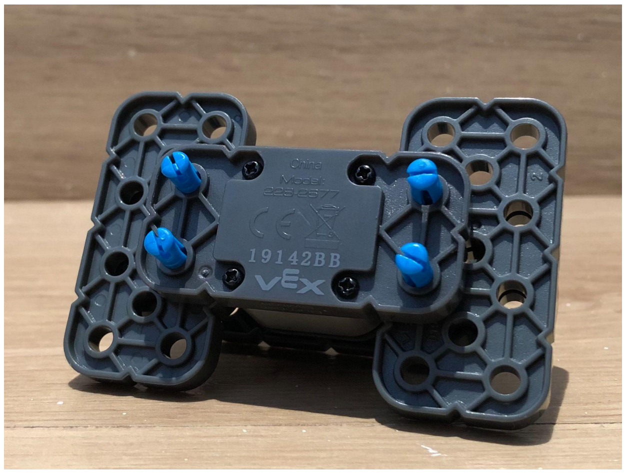


```
when started
set drive velocity to 50 %
forever
    if Color4 detects red ? then
        stop driving
        print Stop! on Brain ▶
        reset timer
        wait until timer in seconds > 3
        clear all rows on Brain
        set cursor to row 1 column 1 on Brain
    else
        drive forward
```

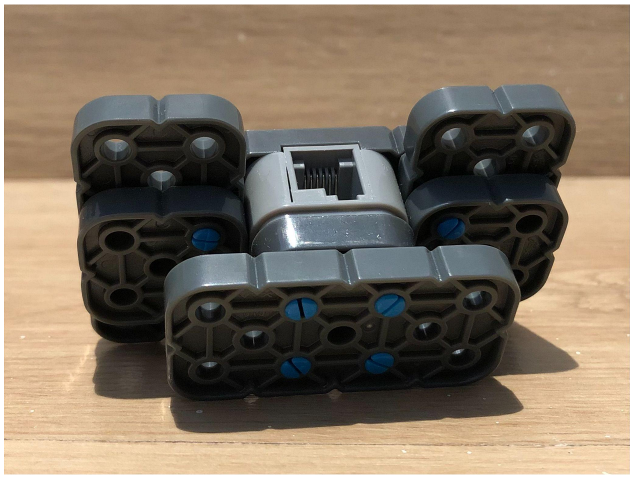
Line Following:
The following images are the robot configuration for line following using the color sensor. This configuration is needed to have the robot perform line following and decision making at the same time. The bumper piece that needs to come off and the color sensor piece that needs to be attached for the line following configuration are shown below. The motors also need to be put into the VEX IQ as individual motors for this part of the lesson, so when you see "RightMotor" that is the right motor set up as an individual motor, and same for the left motor.

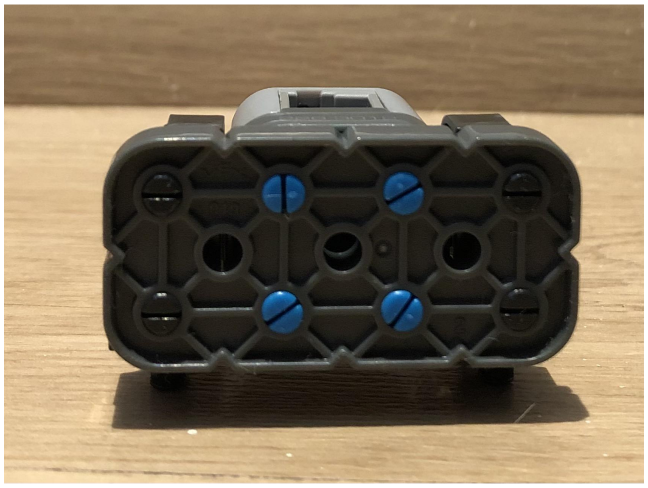


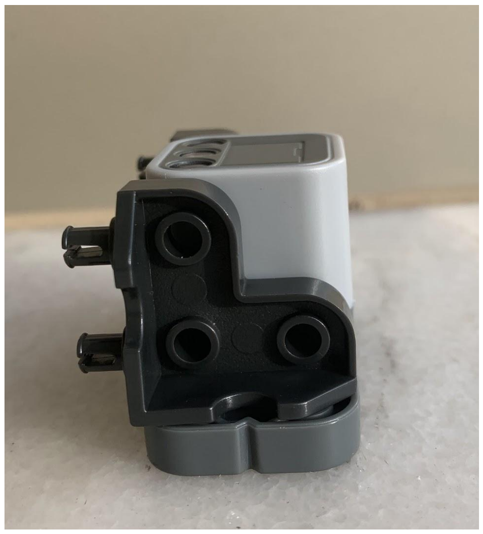
Color Sensor location for line following



The back of the detached bumper sensor assembly

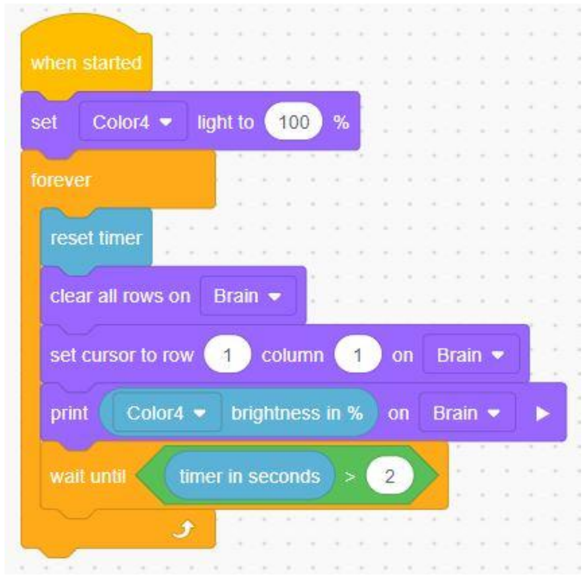

The front of the detached bumper sensor assembly


The back of the detached color sensor assembly


The front of the detached color sensor assembly

Color Sensor Determining Brightness Values:
This helps to determine the brightness values used to make the robot turn right or left to stay on the line

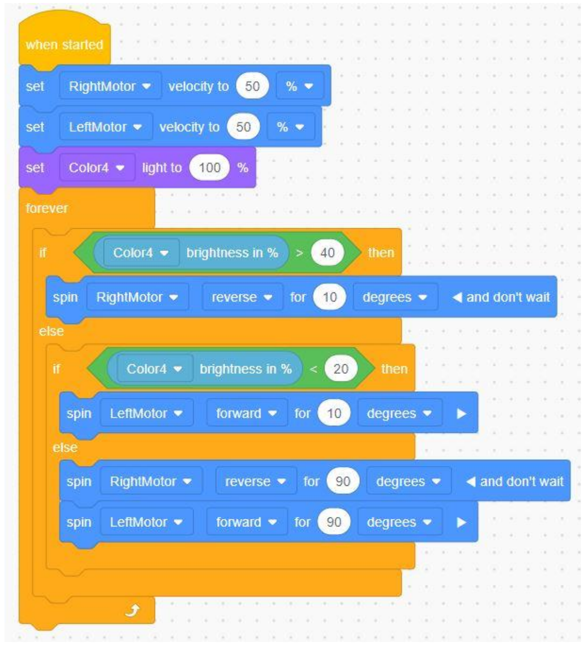
When the play button is pressed the color sensor will adjust its light to 100%. The robot will reset the timer, clear all the rows in the brain, set the cursor to row 1 column 1 on the Brain. The robot will then print the brightness level in percentage on the brain. The robot will wait until the timer is greater than 2 seconds and repeat the process forever. This helps to determine the 40 and 20 down below in the follower code.



```
when started
set Color4 light to 100 %
forever
    reset timer
    clear all rows on Brain
    set cursor to row 1 column 1 on Brain
    print Color4 brightness in % on Brain ▶
    wait until timer in seconds > 2
```

Line Following:
This is the line follower code, it is just a baseline and would need to be adjusted to be more accurate and make the robot go faster.

When the play button is pressed the right and left motor's velocity will be set to 50%. The robot will set its light to 100%. Then if the robot detects light with a brightness over 40% then the right motor will spin in reverse (forward because the motor is upside down) for 10 degrees. Else, if the robot detects light with a brightness less than 20% then the left motor will spin forward 10 degrees. If that condition is also not met then both motors will spin in the robot forward direction and it will drive forward.

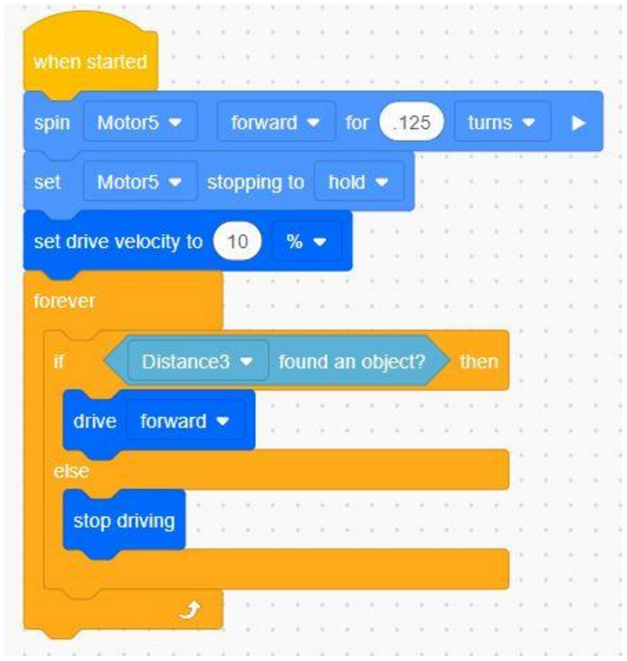


```
when started
set RightMotor velocity to 50 %
set LeftMotor velocity to 50 %
set Color4 light to 100 %
forever
    if Color4 brightness in % > 40 then
        spin RightMotor reverse for 10 degrees ◀ and don't wait
    else
        if Color4 brightness in % < 20 then
            spin LeftMotor forward for 10 degrees ▶
        else
            spin RightMotor reverse for 90 degrees ◀ and don't wait
            spin LeftMotor forward for 90 degrees ▶
```

Ensuring the Ultrasonic Sensor Works:
The ultrasonic sensor is also referred to as the "distance sensor".

The ultrasonic sensor is explained in the Lesson Manual.

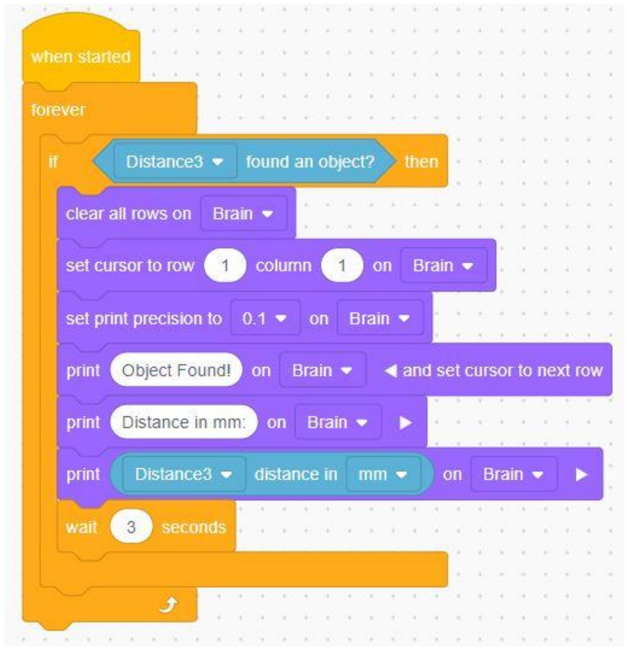
When the play button is pressed the motor in the 5th port will spin forward .125 turns. The motor will then hold at that position for the remainder of the code. The drive velocity will be set to 10%. The robot will then detect if an object is seen. If an object is detected the robot will move forward. Else the robot will stop moving.



```
when started
spin Motor5 forward for .125 turns
set Motor5 stopping to hold
set drive velocity to 10 %
forever
    if Distance3 found an object? then
        drive forward
    else
        stop driving
```

Robot as a Ruler:
This activity is detailed in the Lesson Manual.

When the play button is pressed the robot will search to detect an object, if the robot does detect an object then all rows on the brain will be cleared, the robot will set the cursor to row 1 column 1 on the brain. The robot will set print precision to 0.1 on the brain. The robot will print "object found!" on the brain and set the cursor to the next row. The brain will then print "distance in mm:" on the display. The robot will print the distance detected on the display. The robot will wait 3 seconds and repeat the actions.



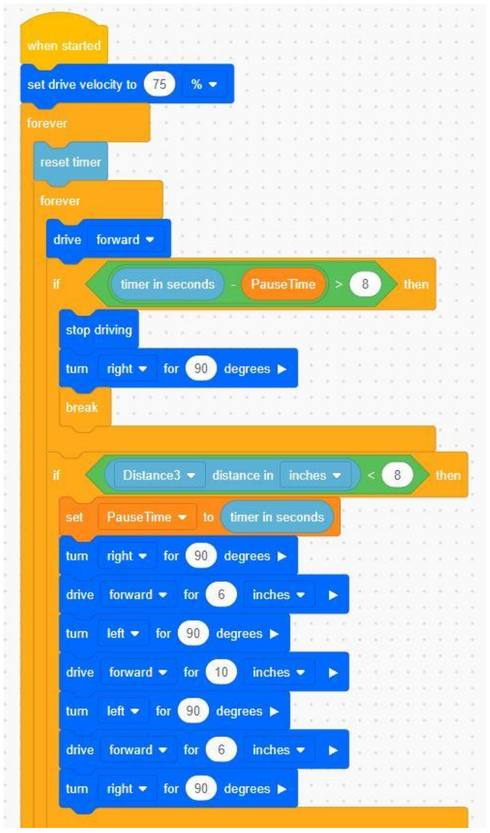
```
when started
forever
    if Distance3 found an object? then
        clear all rows on Brain
        set cursor to row 1 column 1 on Brain
        set print precision to 0.1 on Brain
        print Object Found! on Brain and set cursor to next row
        print Distance in mm: on Brain
        print Distance3 distance in mm on Brain
        wait 3 seconds
```

Path Finding and Object Avoidance:

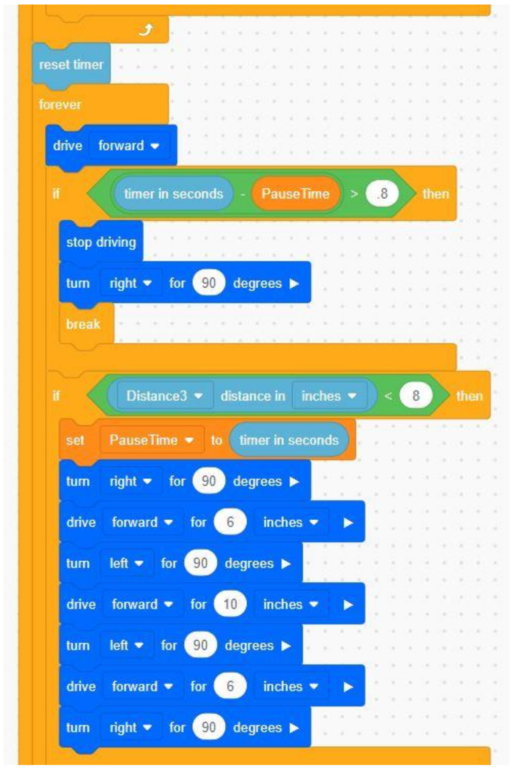
challenging activity* or *optional

In this activity the robot is supposed to drive in a lawnmower path (forward and back), but there are obstacles it needs to avoid so it has to path around them. The code for this activity is somewhat inconsistent, but it would be beneficial to show students the possibilities that come from a program like this. The values can all be changed to make it work better or differently if needed.

When the play button is pressed the robot sets a time and begins to drive forward. The robot will move forward and run through a series of if/then statements to constantly check the conditions set forth in the if/then statements. It constantly checks if it has traveled far enough by using the first if statement. It also checks if there is an object in the way by using the second if statement. If there is an object in the way then the robot records that time and stores it until the object is avoided. Then the robot keeps checking if it has passed the required time. Once it passes the required time it goes through the if statement and breaks out of that forever loop and moves to the next forever loop. It does this for all the steps and keeps repeating the search pattern until it is told to stop.

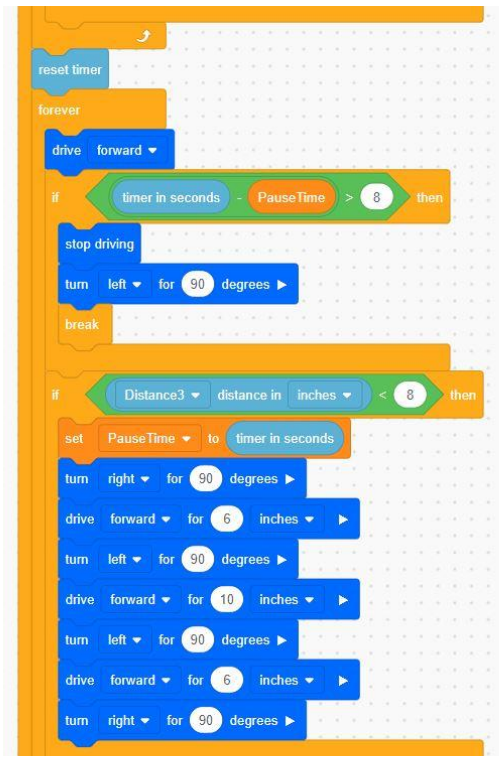


```
when started
set drive velocity to 75 %
forever
    reset timer
    forever
        drive forward
        if < timer in seconds - PauseTime > 8 > then
            stop driving
            turn right for 90 degrees
            break
        if < Distance3 distance in inches < 8 > then
            set PauseTime to timer in seconds
            turn right for 90 degrees
            drive forward for 6 inches
            turn left for 90 degrees
            drive forward for 10 inches
            turn left for 90 degrees
            drive forward for 6 inches
            turn right for 90 degrees
```

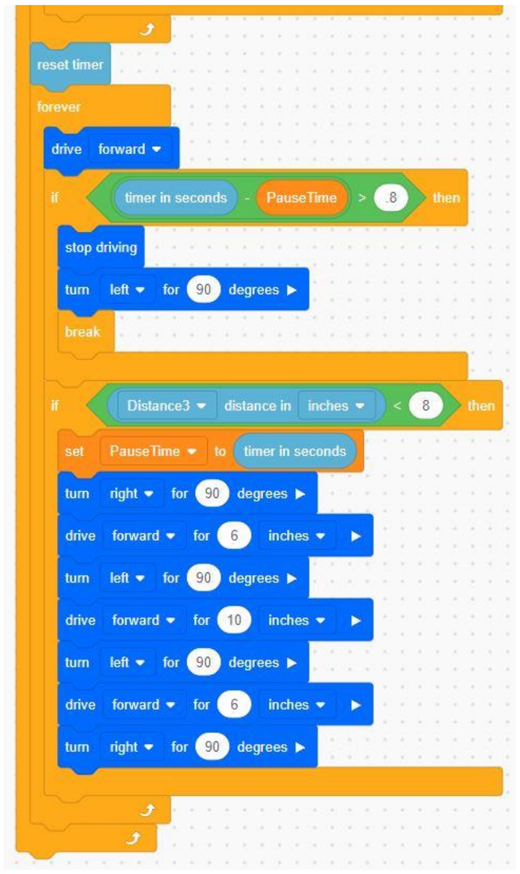


```
    reset timer
    forever
        drive forward
        if < timer in seconds - PauseTime > .8 > then
            stop driving
            turn right for 90 degrees
            break
        if < Distance3 distance in inches < 8 > then
            set PauseTime to timer in seconds
            turn right for 90 degrees
            drive forward for 6 inches
            turn left for 90 degrees
            drive forward for 10 inches
            turn left for 90 degrees
            drive forward for 6 inches
            turn right for 90 degrees
```

Path following and objective avoidance (cont.)

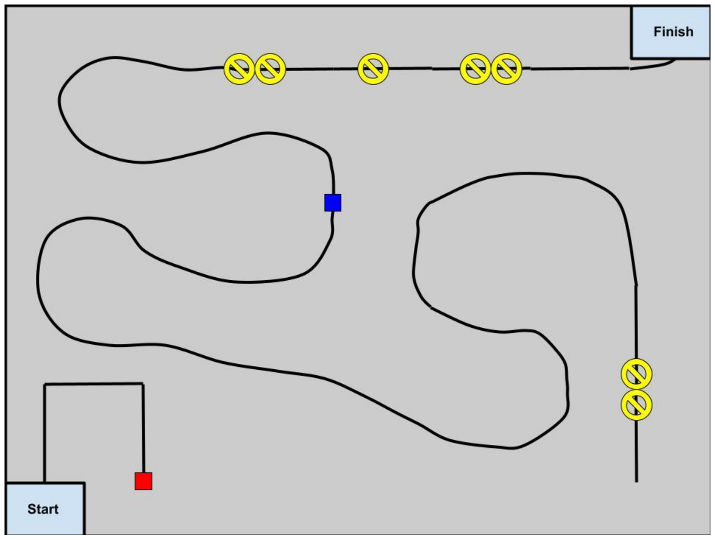


```
reset timer
forever
  drive forward
  if timer in seconds - PauseTime > 8 then
    stop driving
    turn left for 90 degrees
    break
  if Distance3 distance in inches < 8 then
    set PauseTime to timer in seconds
    turn right for 90 degrees
    drive forward for 6 inches
    turn left for 90 degrees
    drive forward for 10 inches
    turn left for 90 degrees
    drive forward for 6 inches
    turn right for 90 degrees
```

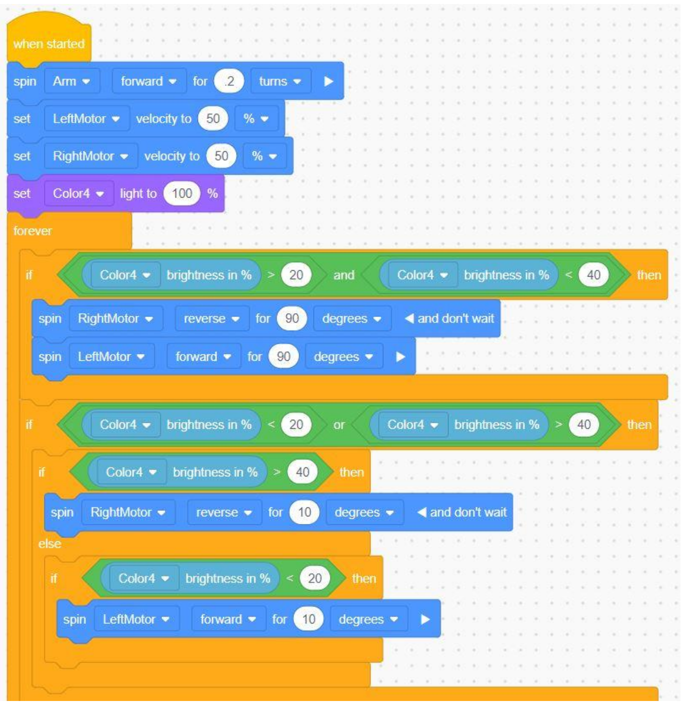


```
reset timer
forever
  drive forward
  if timer in seconds - PauseTime > .8 then
    stop driving
    turn left for 90 degrees
    break
  if Distance3 distance in inches < 8 then
    set PauseTime to timer in seconds
    turn right for 90 degrees
    drive forward for 6 inches
    turn left for 90 degrees
    drive forward for 10 inches
    turn left for 90 degrees
    drive forward for 6 inches
    turn right for 90 degrees
```

Big Maze Activity
The image to the right shows a possible playing field for the maze activity. The red and blue squares are decisions where the robot will have to do something. The yellow objects are pieces the robot will pathfind around and avoid. The reason “solo” motors are being used is because it makes it easier to control moving forward while also paying attention to pathing. See “How to add a device to a port in VEXcode IQ” on how to add a motor.



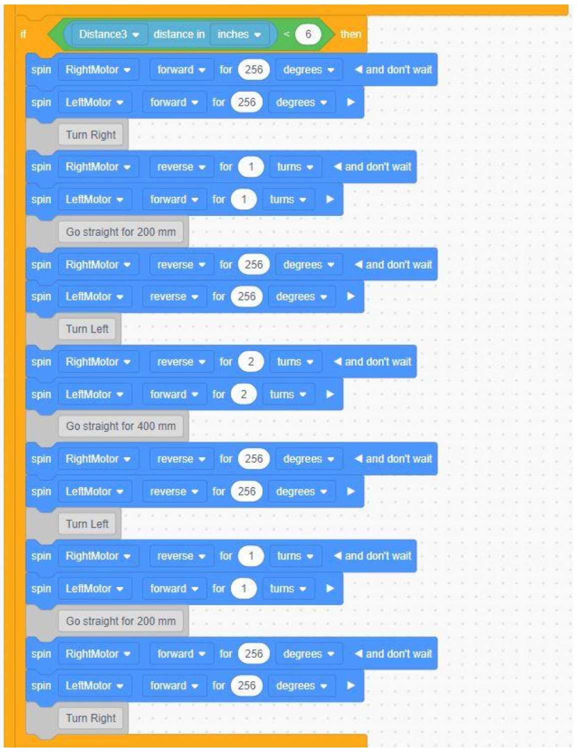
The code in this part of the lesson works by constantly running through a series of if/then statements. The main function that the robot should be doing is line following, which is accomplished in the first two if/then statements. The other two processes the robot is doing are object avoidance (3rd if statement) and decision making (4th and 5th if statements). The decisions the robot makes at the different colors is arbitrary and can be changed.



```
when started
spin Arm forward for .2 turns ▶
set LeftMotor velocity to 50 %
set RightMotor velocity to 50 %
set Color4 light to 100 %
forever
    if Color4 brightness in % > 20 and Color4 brightness in % < 40 then
        spin RightMotor reverse for 90 degrees ◀ and don't wait
        spin LeftMotor forward for 90 degrees ▶
    if Color4 brightness in % < 20 or Color4 brightness in % > 40 then
        if Color4 brightness in % > 40 then
            spin RightMotor reverse for 10 degrees ◀ and don't wait
        else
            if Color4 brightness in % < 20 then
                spin LeftMotor forward for 10 degrees ▶
```

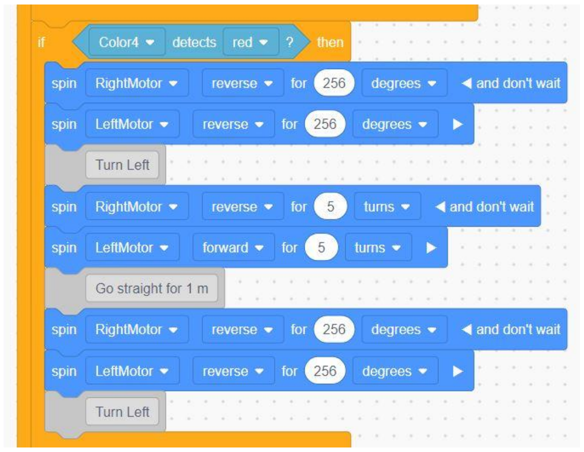
Big Maze Activity (cont.)

When the sensor detects an object less than six inches away, then the left and right motor will turn ,as shown in the picture.



```
if  Distance3 distance in inches < 6  then
    spin RightMotor forward for 256 degrees and don't wait
    spin LeftMotor forward for 256 degrees
    Turn Right
    spin RightMotor reverse for 1 turns and don't wait
    spin LeftMotor forward for 1 turns
    Go straight for 200 mm
    spin RightMotor reverse for 256 degrees and don't wait
    spin LeftMotor reverse for 256 degrees
    Turn Left
    spin RightMotor reverse for 2 turns and don't wait
    spin LeftMotor forward for 2 turns
    Go straight for 400 mm
    spin RightMotor reverse for 256 degrees and don't wait
    spin LeftMotor reverse for 256 degrees
    Turn Left
    spin RightMotor reverse for 1 turns and don't wait
    spin LeftMotor forward for 1 turns
    Go straight for 200 mm
    spin RightMotor forward for 256 degrees and don't wait
    spin LeftMotor forward for 256 degrees
    Turn Right
```

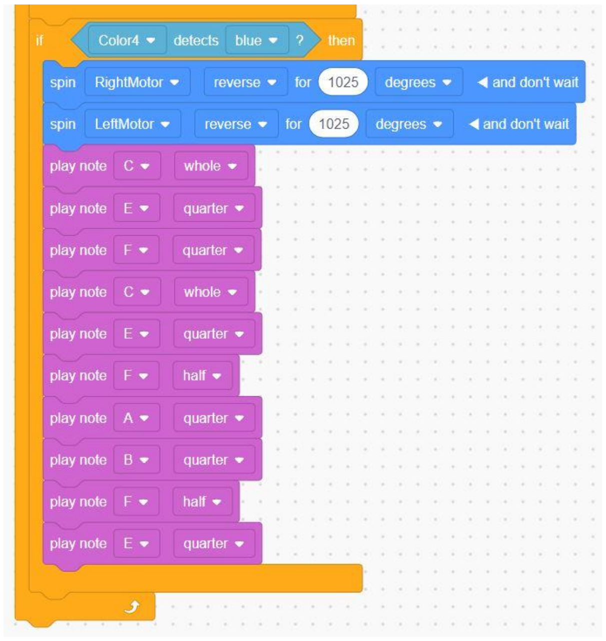
If the sensor detects red, the left and right motors will move in the direction as shown in the picture. It will make the robot turn left, go straight for one meter and then turn left again.



```
if  Color4 detects red ?  then
    spin RightMotor reverse for 256 degrees and don't wait
    spin LeftMotor reverse for 256 degrees
    Turn Left
    spin RightMotor reverse for 5 turns and don't wait
    spin LeftMotor forward for 5 turns
    Go straight for 1 m
    spin RightMotor reverse for 256 degrees and don't wait
    spin LeftMotor reverse for 256 degrees
    Turn Left
```

Big Maze Activity (cont.)

If the sensor detects a blue object then the right motor will spin for 1025 degrees followed by the left motor spinning 1025 degrees. After, the notes displayed in the picture will be played.



```
if < Color4 detects blue ? > then
    spin RightMotor reverse for 1025 degrees ◀ and don't wait
    spin LeftMotor reverse for 1025 degrees ◀ and don't wait
    play note C whole
    play note E quarter
    play note F quarter
    play note C whole
    play note E quarter
    play note F half
    play note A quarter
    play note B quarter
    play note F half
    play note E quarter
```

Lesson 5: Joints and Arms

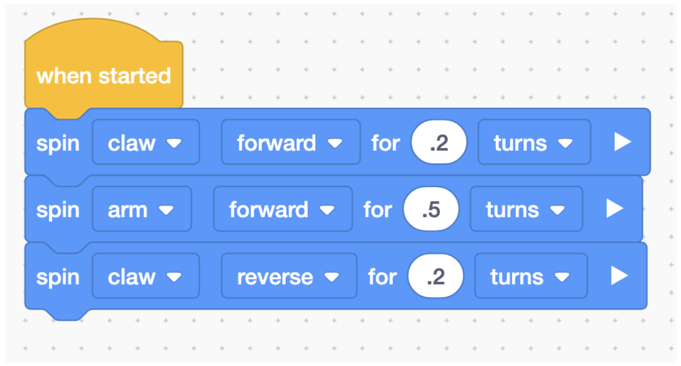
This lesson is applicable for both the VEXcode VR and the VEXcode IQ platforms.
The following lesson outlines the code and necessary steps it takes to code for arm and claw movement.
With each lesson students, and yourself, should feel welcome to play around with other values and formats to understand logic.
Ensure that arm motors are attached in the devices of VEXcode IQ platform

Picking up an Object located in Between Claw:

Note that the students will have to test various objects to pick up as depending on the shape and weight of the object, it could be difficult to complete some of the tasks.

When the play button is pressed:
1. The claw spins forward .2 turns
2. The arm spins forward .5 turns
3. The claw spins in reverse .2 turns.

Note: The example code is using turns, but it is also possible to use degrees. If you close the claw with more than .2 turns, it will not work. You will also have to test different objects to determine what works best.

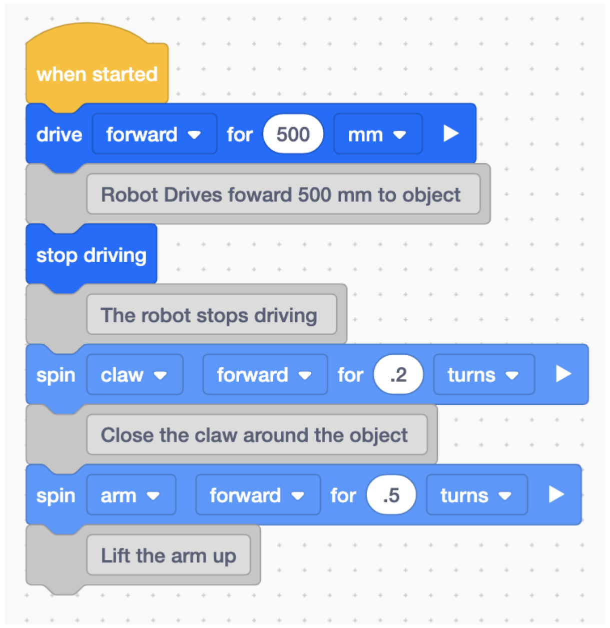


```
when started
spin claw forward for .2 turns
spin arm forward for .5 turns
spin claw reverse for .2 turns
```

Forward Movement to Pick up an Object:

Movement forward for a defined distance parameter (500mm). The motor to the claw spins forward .2 turns to close the claw. The arm motor will then spin forward .5 turns to lift up the arm.

Students should feel welcome to see the robot move different values in distance and turns.



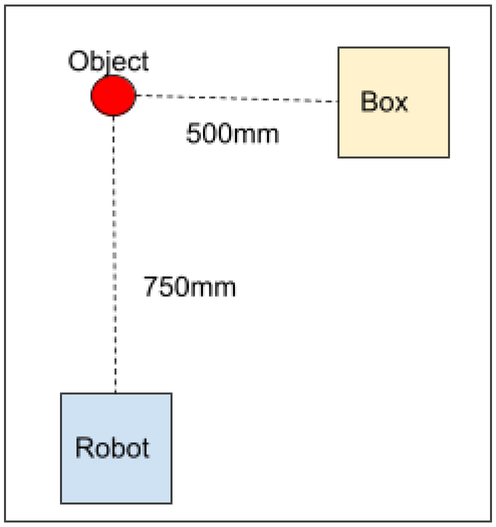
```
when started
drive forward for 500 mm
Robot Drives foward 500 mm to object
stop driving
The robot stops driving
spin claw forward for .2 turns
Close the claw around the object
spin arm forward for .5 turns
Lift the arm up
```

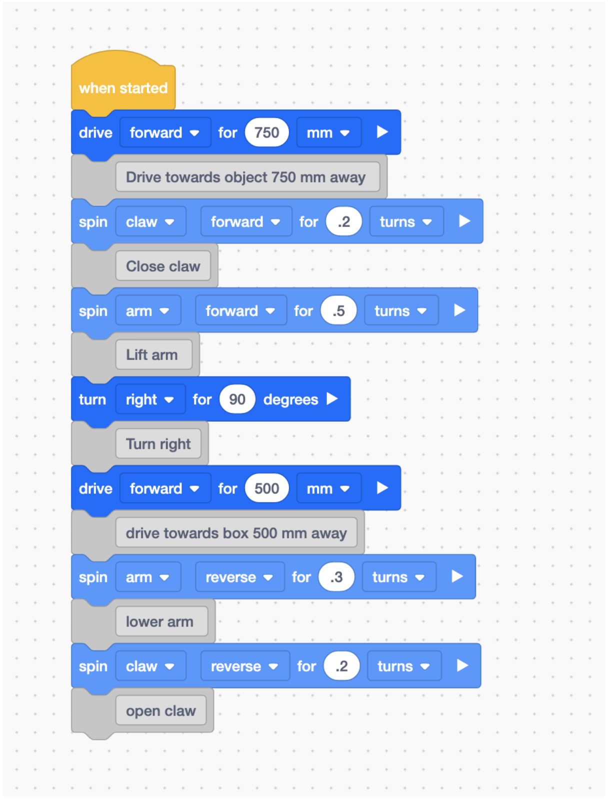
Picking up an object and placing the object on a box:

Set up activity based on the picture seen right. The values are adjustable

When the play button is pressed the robot will drive forward 750 mm. The robot will then spin the claw motor forward .2 turns and the arm motor forward .5 turns, lifting the arm. The robot will then turn right 90 degrees and move forward 500mm. Once the robot has moved forward 500 mm the arm motor will reverse .3 turns lowering the arm. Then the claw motor will reverse .2 turns opening the claw and placing the object.

The distance and direction of the box is subject to change based on the student preference, this is just one example. The lowering of the arm will depend on the height of the box.

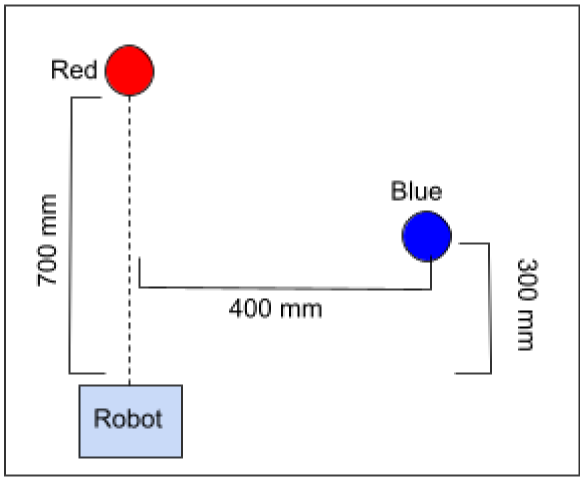




```
when started
drive forward for 750 mm
    Drive towards object 750 mm away
spin claw forward for .2 turns
    Close claw
spin arm forward for .5 turns
    Lift arm
turn right for 90 degrees
    Turn right
drive forward for 500 mm
    drive towards box 500 mm away
spin arm reverse for .3 turns
    lower arm
spin claw reverse for .2 turns
    open claw
```

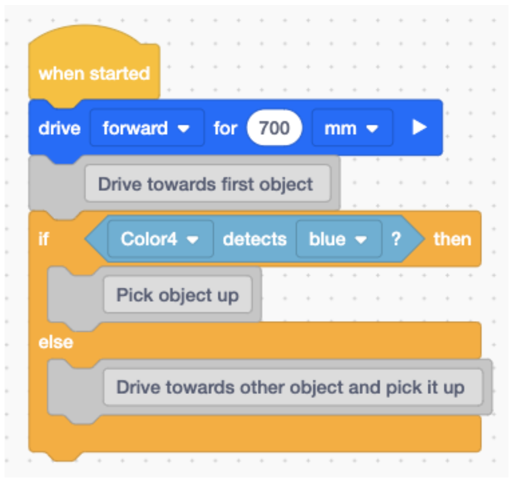
**Picking up an Object Based on Color:
*this is a difficult task***

This is a difficult task as the color sensor on the robot is inconsistent. Students should test different objects to determine if the robot can detect that color. This can be done by designing code that will play a sound when the robot detects a specific color. The student should hold the object in front of the robot and move it back and forth in front of the color sensor until a sound is played.



Set up activity based on the picture seen right. Users can select values and colors as they see fit. Adjust the following code according to the user's selected distance and color selection.

The code demonstrates **the logic** of detecting the blue object and picking it up. Note that the code is designed for the diagram shown on the right. The students should test their code with trial and error and use the logic provided if needed.

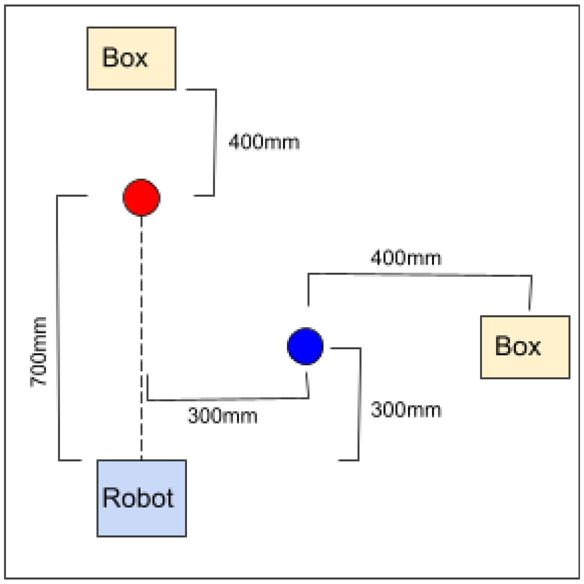


```
when started
drive forward for 700 mm
    Drive towards first object
if Color4 detects blue ? then
    Pick object up
else
    Drive towards other object and pick it up
```

When the play button is pressed the robot will drive 700 mm forward. The robot will check if that object is blue. If the object is blue then the robot will pick the object up. Else, drive towards the other object, detect if it is blue and if so, pick it up.

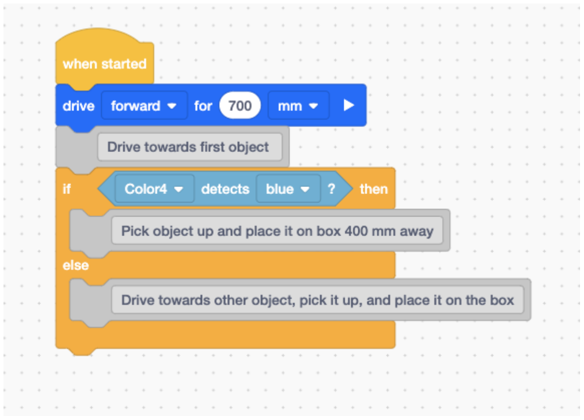
Picking up an object based on color and placing it on a box:
this is a difficult task

This is a difficult task as the color sensor on the robot is inconsistent. Students should test different objects to determine if the robot can detect that color. This can be done by designing code that will play a sound when the robot detects a specific color. The student should hold the object in front of the robot and move it in front of the color sensor until a sound is played.



Set up activity based on the picture seen right. Users can select values and colors as they see fit. Adjust the following code according to the user's selected distance and color selection.

The logic for this code is explained in the example above. The only difference is that this example also includes placing the object on a box. The best way to tackle this problem is by testing the code a little at a time.



```
when started
drive forward for 700 mm
Drive towards first object
if Color4 detects blue ? then
    Pick object up and place it on box 400 mm away
else
    Drive towards other object, pick it up, and place it on the box
```

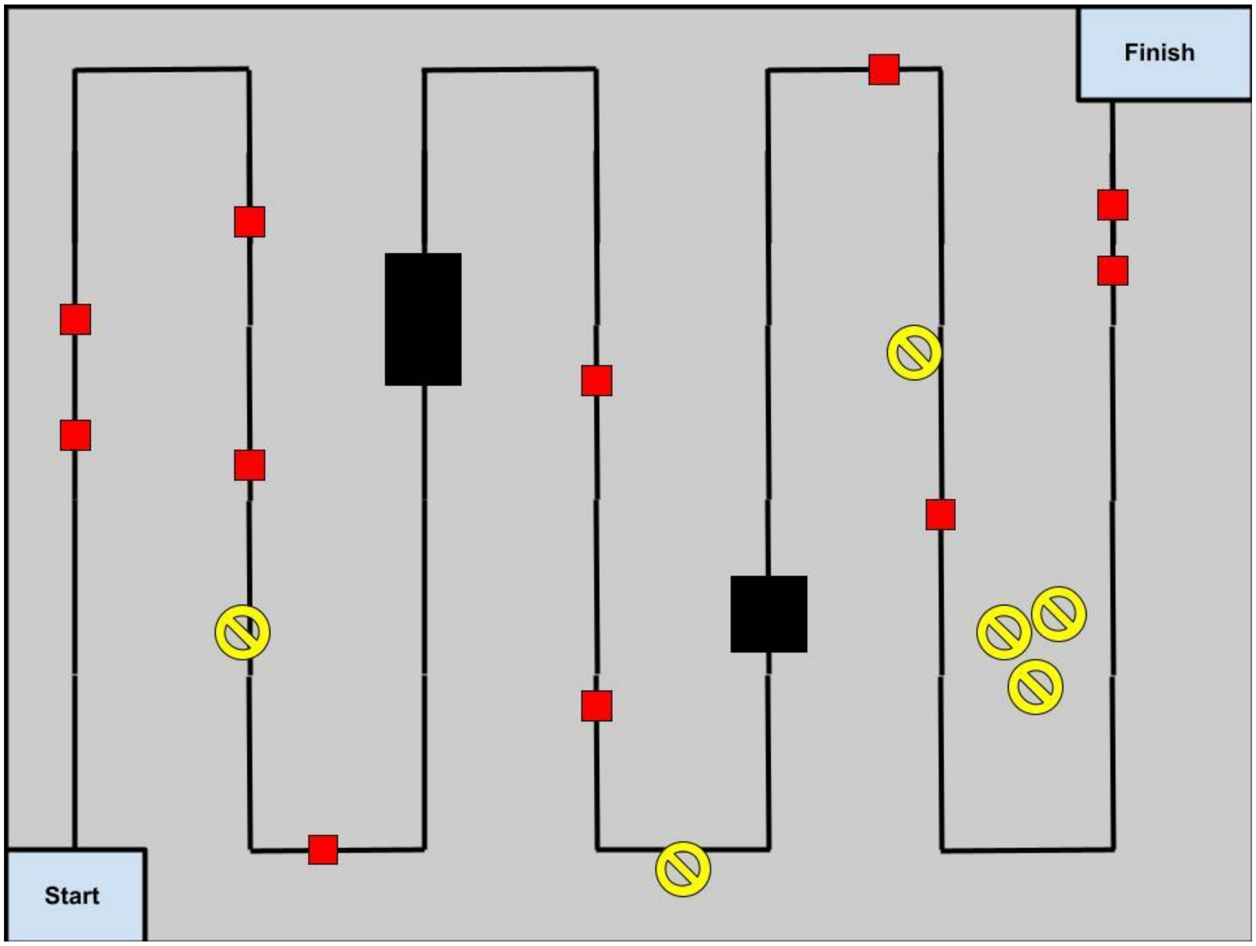
When the play button is pressed the robot will drive 700 mm forward. The robot will check if that object is blue. If the object is blue then the robot will pick the object up. Else, drive towards the other object, detect if it is blue and if so, pick it up and move the correct distance to place the object on the box. Reference "*Picking up an object and placing the object on a box*" for guidance on how to place an object on a box.

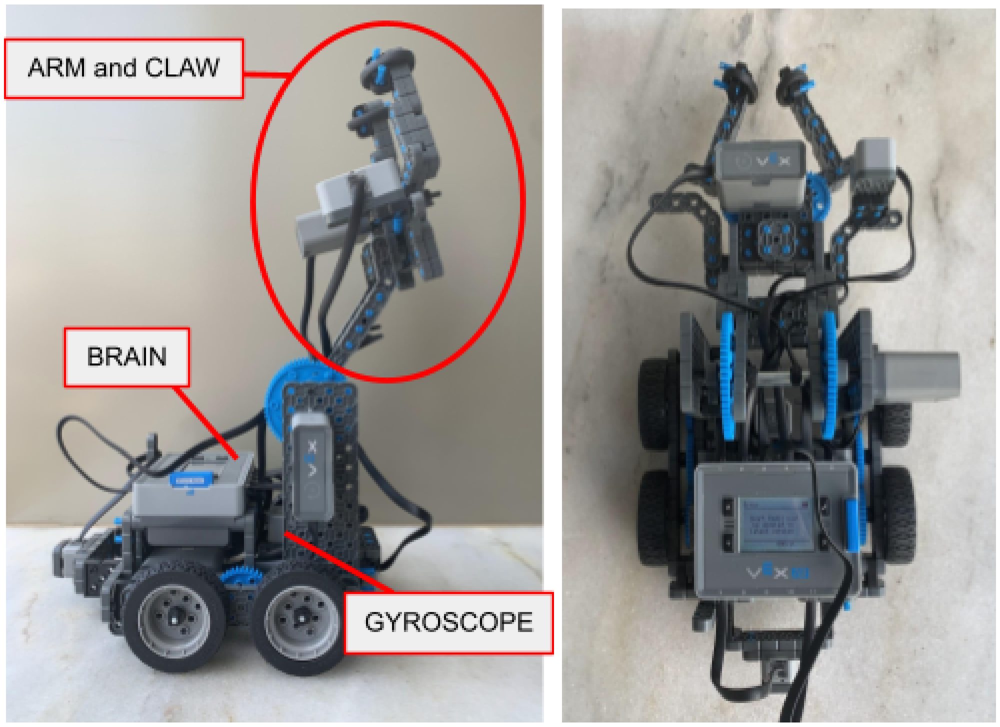
Lesson 6 & 7: Project Introduction and Presentation

The project introduction, prompt, and presentation details are outlined in the lesson manual.

For the final project and presentation, we recommend creating a standard arena in a defined grid. Amongst the grid, it is recommended to have gathered objects of light weight and at least 3 by 3 inch dimensions with varied colors. We recommend having obstacles or walls set up amongst the area and as the border. The project goal can be altered or adjusted as needed. Below is a visual of a possible final robot, it is the basebot, equipped with a gyroscope, arms and claws, and a bumper sensor.
This is a possible playing field for the final project, where the black lines are the path the robot will travel using path finding. The red objects are trash it will pick up while traveling, and the yellow objects are objects it needs to either pick up and move or play a noise to get them to move. The black objects are rocks that cannot be moved and need to be avoided by the robot. There are a variety of ways to code the final robot that are introduced in the above lessons and should help to guide completion of the tasks

Potential Playing Field:





Picture of potential final robot