LAWRENCE TECHNOLOGICAL UNIVERSITY ROBOFEST 2020

BOTTLESumo

Vex IQ Workshop
Using Robot Mesh

This file can be found under the **TechResources → Workshops** page on the website
Coaches are responsible for communicating rules updates to contestants

www.robofest.net robofest@ltu.edu 248-204-3568

Room J233 Taubman Complex, LTU
21000 West 10 Mile Road, Southfield, MI 48075, USA

LAWRENCE TECHNOLOGICAL UNIVERSITY ROBOFEST

2020 Workshops

Sponsored by

DENSO

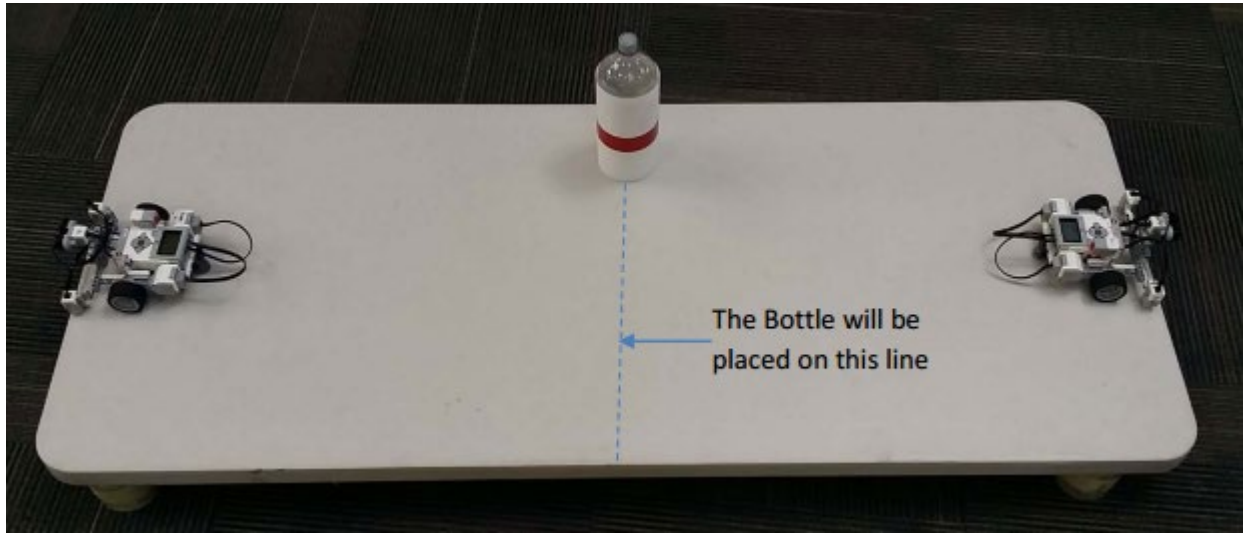
Lawrence Technological University

Computer Science

Course Overview

- 2019-20 RoboFest BottleSumo Rule Highlights
- Vex IQ Robot Configuration
- RobotMesh Software introduction
- Using the Vex IQ for BottleSumo
  - Sense the table edge
  - Finding the bottle/opponent
  - Push bottle/opponent off
- Physics and Strategy

Rules Overview



- Objective: Intentionally push the bottle or opponent off **AND** remain on the table for > 3 second after
- How to start the robot – unveiled 30 min before impound
- Starting location, orientation, and exact location of bottle zone unveiled after impounding
- Time trials – push 2 bottles off the table (3 bottle for Sr. teams)
- Please remember to read the rules!

2020 BottleSumo Robot Rules

- Teams must bring a fully-constructed robot to the competition with labels clearly indicating their team ID number and the “front” of their robot.
- Teams will need to bring laptop computers to modify their programs to solve the unknown starting task as well as to adjust their programs for the lighting conditions, floor color, and table color, etc. that are unknown until the competition day

2020 Robot Requirements (1/2)

- Robot must be fully-constructed upon arrival to the competition
- Robot must be fully autonomous. No human control, signal, or remote computer control (tele-op)
- One robot per team (same robot must be use entire tournament)
- Robots must have labels clearly indicating their team ID number and FRONT of robot (side with sensors)
- Teams will need to bring laptop computers to modify their programs for unknown starting task and to adjust for conditions that are unknown until the competition day

	Junior Division	Senior Classic	Senior Unlimited
Maximum robot weight	0.9 Kg	1.5 Kg	3 Kg
Robot Brain	LEGO NXT, LEGO EV3, Lego Spike Prime or Vex IQ		Any
Maximum robot width, length, and height	Must fit in 20x20x20cm box. Robots may **NOT** expand their dimensions during the game.	Must fit in 30x30x30cm box. Robots **may expand** their dimensions, but the maximum dimensions allowable is 35x35x35cm.	
# of robot brains per robot	One brain only	Any	
Traditional sensor types	Any unless it can be harmful to humans.		

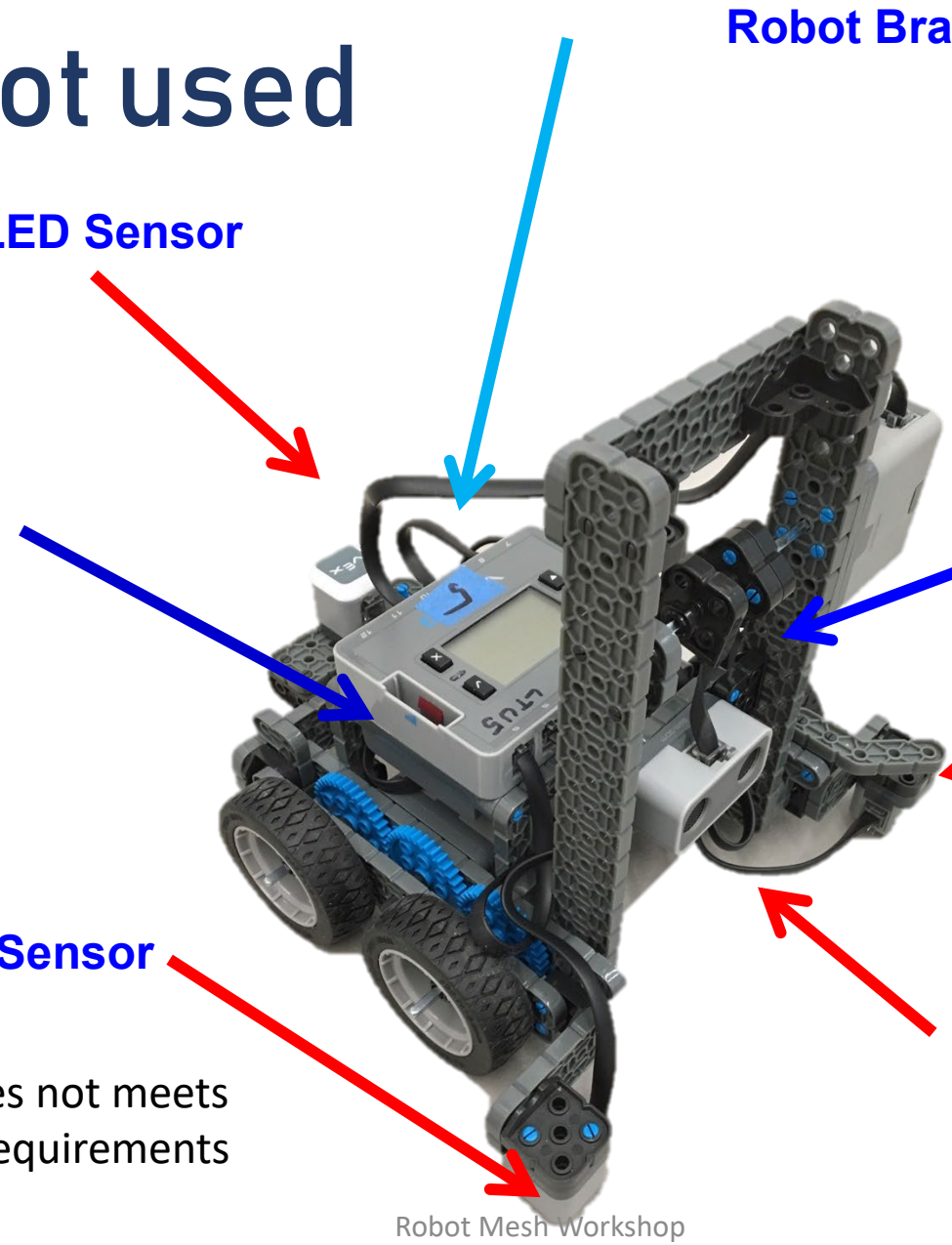
2020 Robot Requirements (2/2)

	Junior Division	Senior Classic	Senior Unlimited
On-board vision sensor system	**NOT** allowed	Examples of allowed vision sensors: • NXTcam: • Pixicam: • Others such as smart phone vision:	
Number of sensors	At least one sensor that can detect dark/light contrast on the plane of the table AND at least one sensor that can detect objects in front . These may be needed for unknown start		
	Maximum 4 sensors (Sensor Multiplexer NOT allowed)		Unlimited (Sensor Multiplexers ALLOWED)
Number of motors	Maximum 3		Unlimited
Motor types	LEGO NXT(9842), LEGO EV3 (455202), LEGO Spike Prime (45602,45603) or Vex IQ (228-2560) only. Voltage altering over default voltage is **NOT** allowed. Other motors such as Lego Power Function and EV3 medium motors **NOT** allowed		Any
Wheels, tread, or legs (the parts driven by motors which touch the ground)	Must be standard Lego or Vex IQ parts that are completely unmodified. Vacuum or sticky material **NOT** allowed		Vacuum or sticky material **NOT** allowed
Other Material	Any. You may use tape, glue, rubber bands, etc. to construct the robot		
Programming language	Any		

What does your robot need to do to win?

- Not fall off the table
- Find objects: bottle or opponent
- Push objects off of table

Vex IQ robot used



Note: Workshop Robot does not meets
Jr Div Robot Size/Weight Requirements

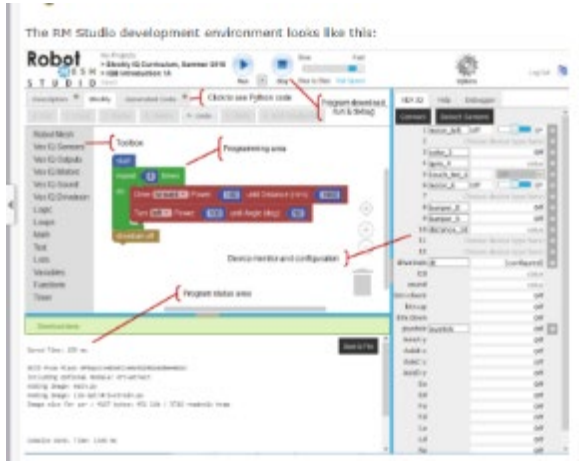
Remember the connections!

- Left Motor connects to **1**
- Right Motor connects to **6**
  - If your motors are upside down forward will be backwards in your program
- LH Color sensor connects to port no. **7**
- RH Color sensor connects to port no. **5**
- Touch LED sensor connects to port no. **4**
- Ultrasonic sensor connects to port no. **2**
- Arm Motor connects to **10**

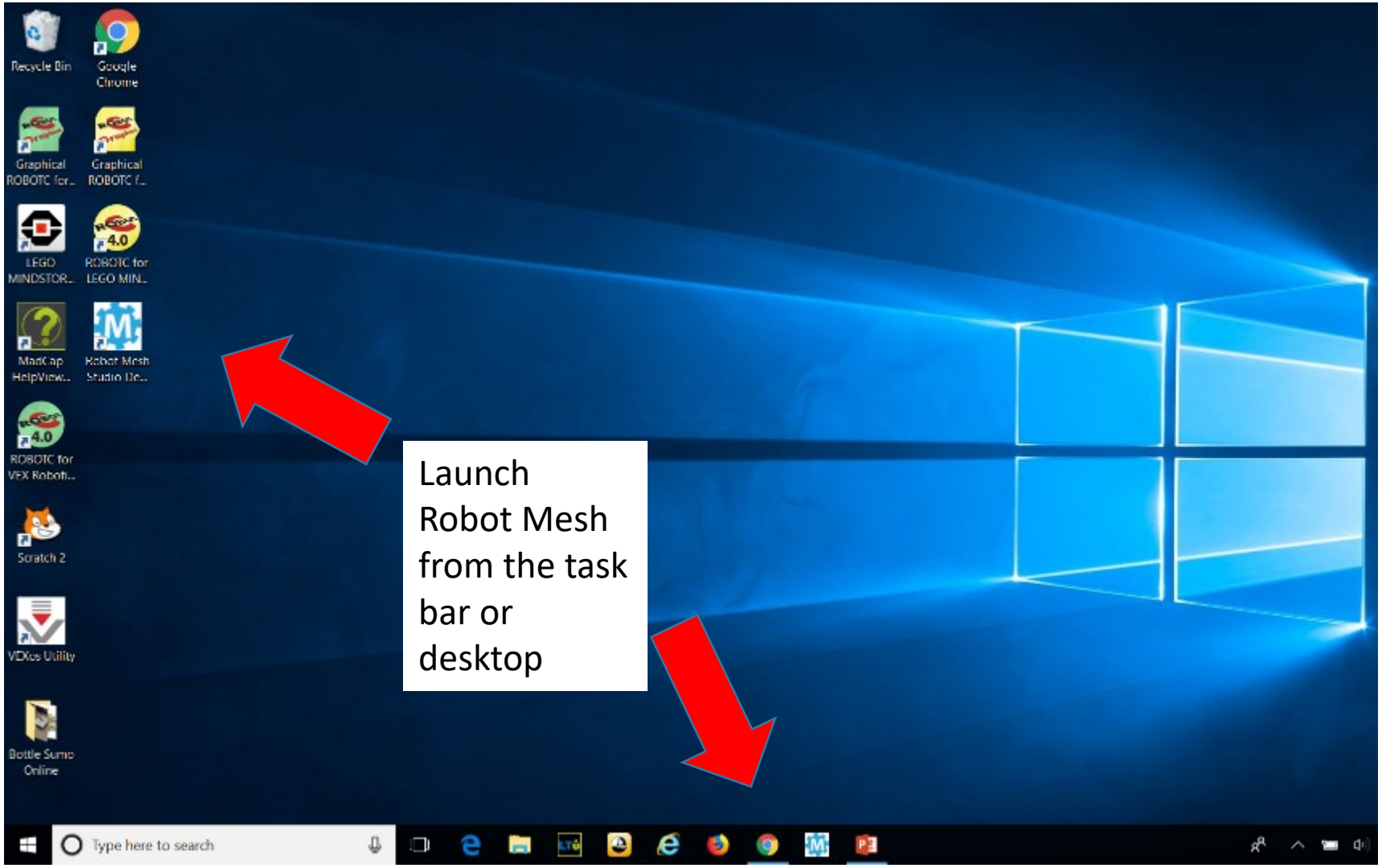
What is Robot Mesh Studio?

- Programming software for VexIQ
- Uses a drag and drop interface (Blockly, same as Scratch)
- Based on Python programming language

Robot MESH STUDIO
Education Edition

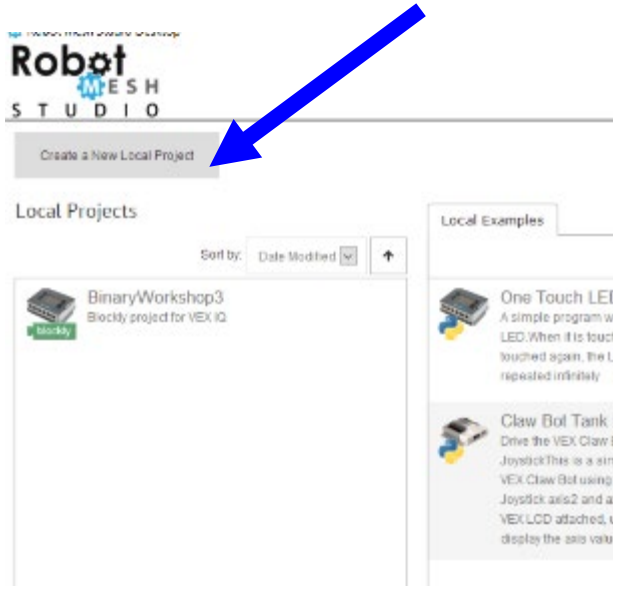


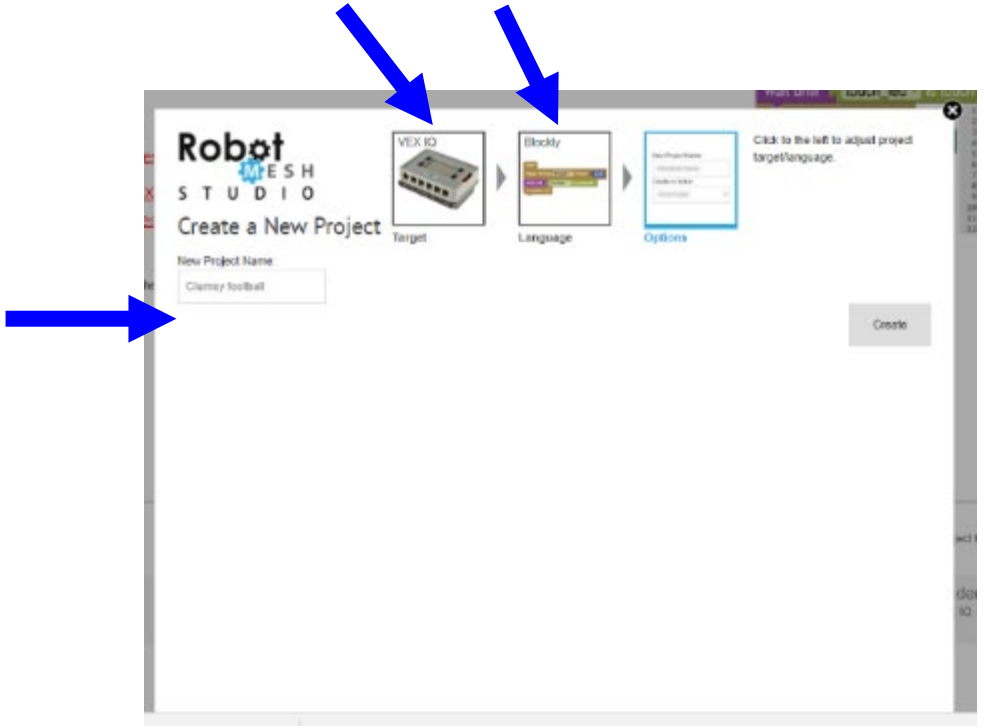
Getting Started



Create a New Program

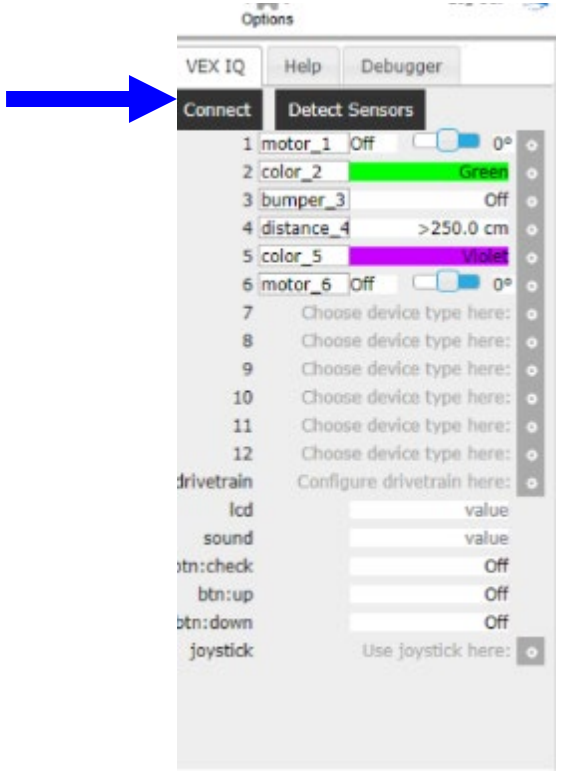
- Click on “Create a new program”
- Make sure these are selected
  1. Target: “Vex IQ”
  2. Language: “Blocky”
- Name the program “Workshop”



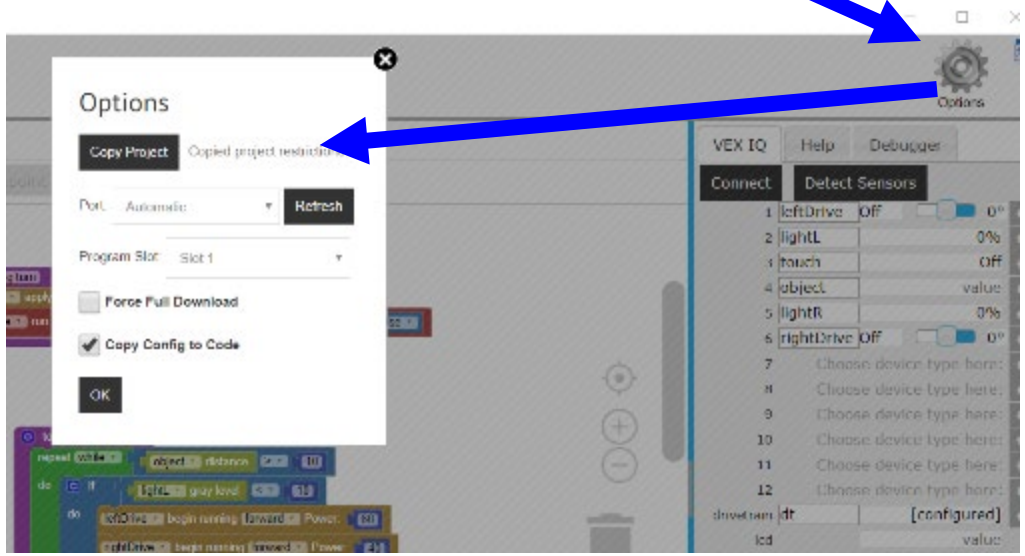


Connect to Robot

- Click on “Connect”



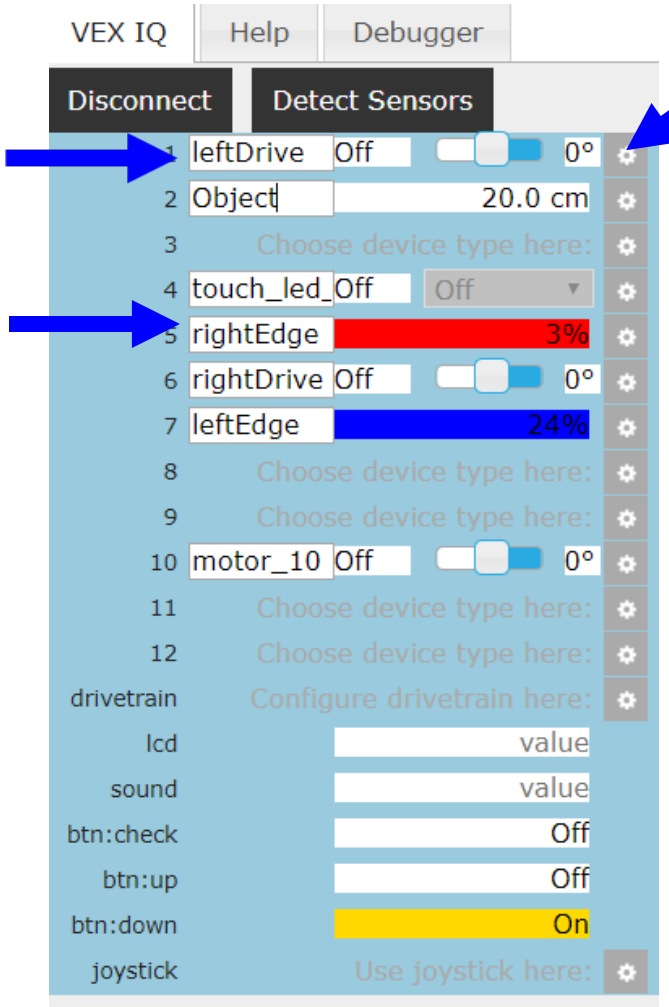
If there are problems connecting, use “options” to manually select port



Motor and Sensor Setup

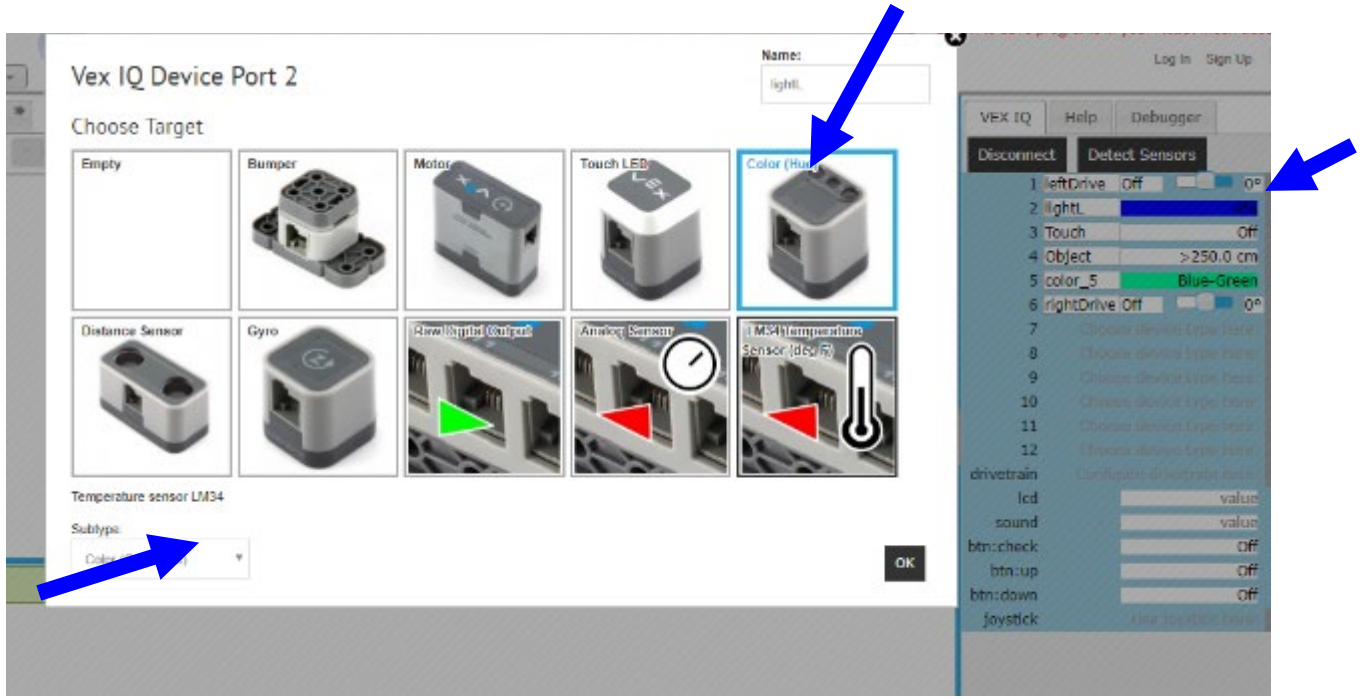
1. Click Here

2. Change to names that have meaning



Motor and Sensor Setup

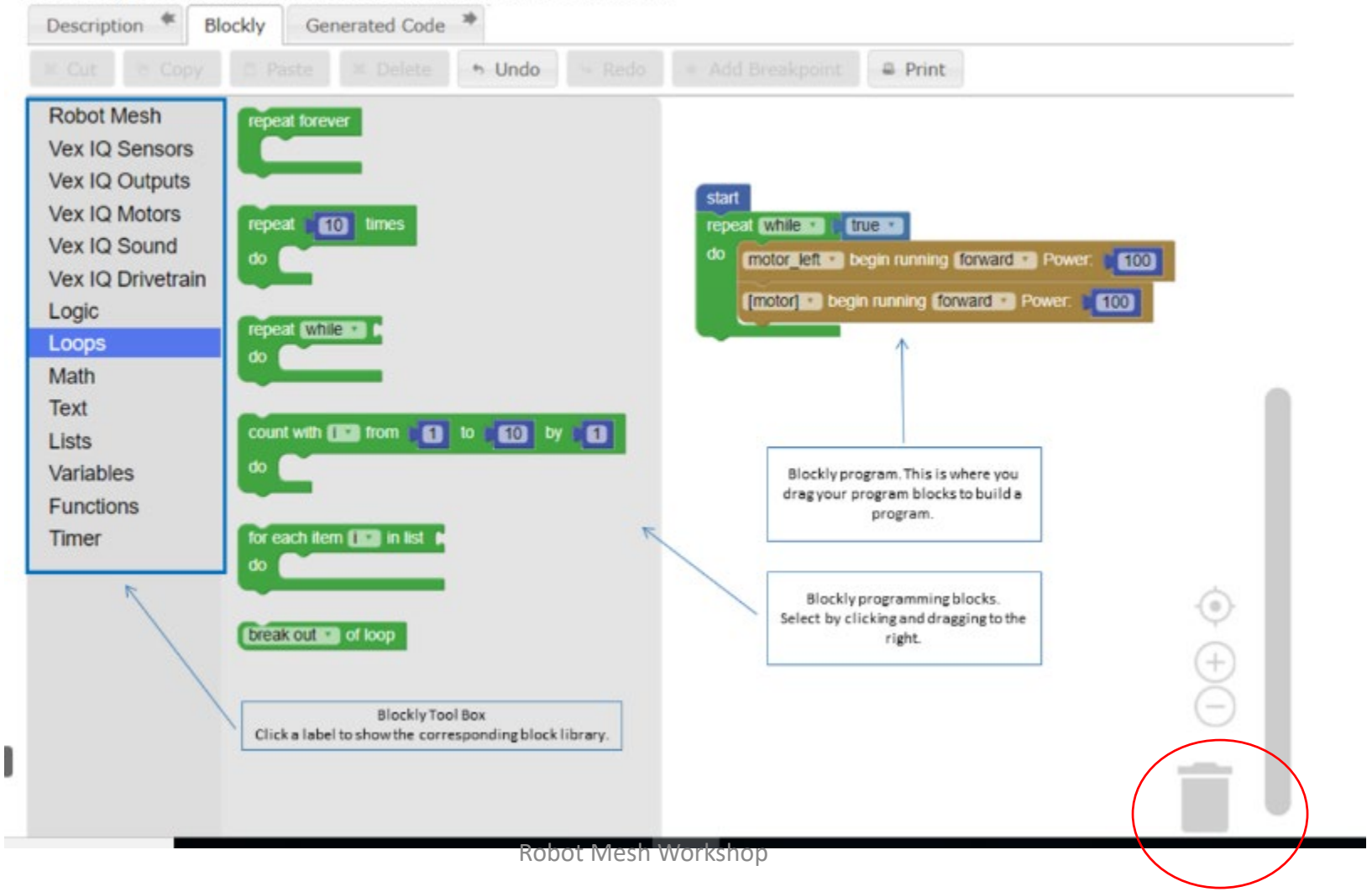
Change Color Sensors to gray scale mode



Blockly Programming

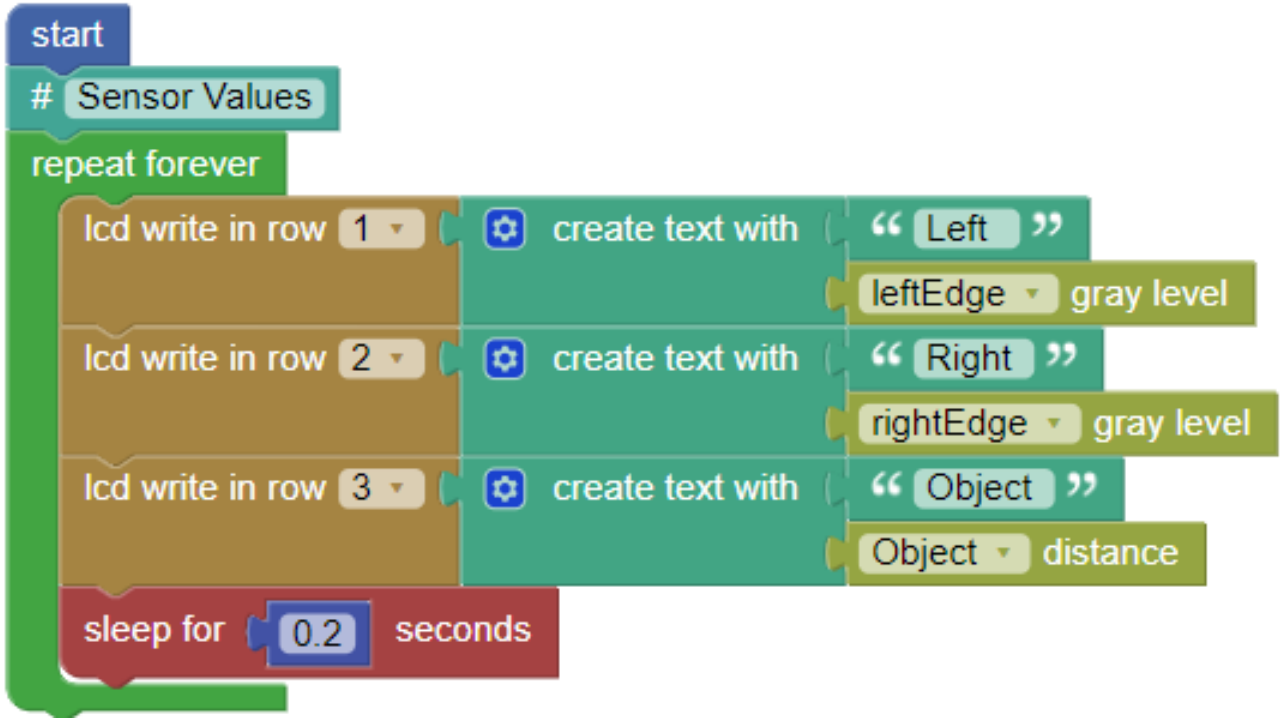
Writing Your Blockly Code

The Blockly programming area in the Robot Mesh Studio IDE looks like this:



Display Sensor Values on LCD

Use the LCD Display to Display the Value of a Sensor

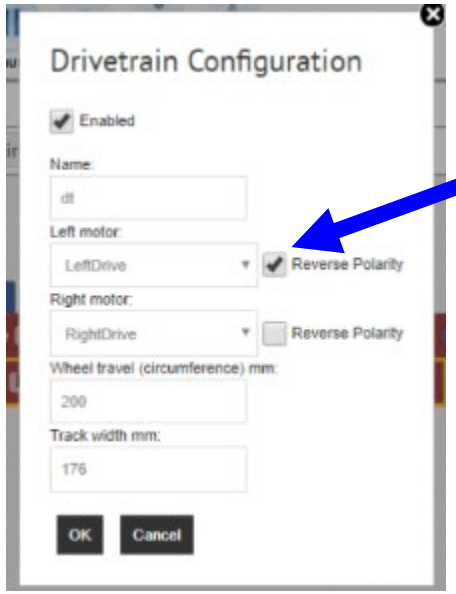


Drivetrain functions

- Robot Mesh has “Drivetrain” programming blocks
- Blocks enable shorter programs and faster coding
- You need to configure the drivetrain in the right side Interface Panel
- Click on the gear symbol

Drivetrain Configuration

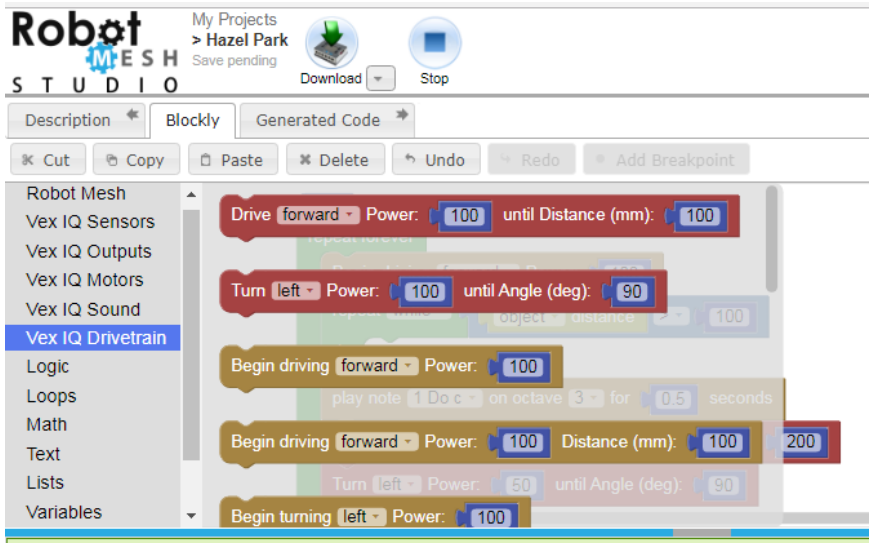
- Configuration dialogue window
- Configuration below should work
- Click “ok” when done



Make sure Left Motor is REVERSED and Right motor is NOT reversed

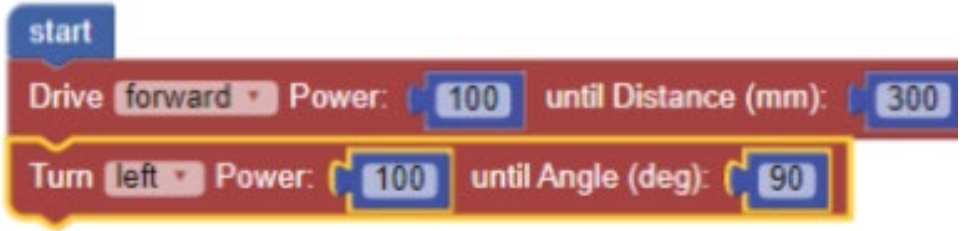
Drivetrain blocks

- Found on left menu in “Vex IQ Drivetrain”
- Enables you to program both drive motors with one block

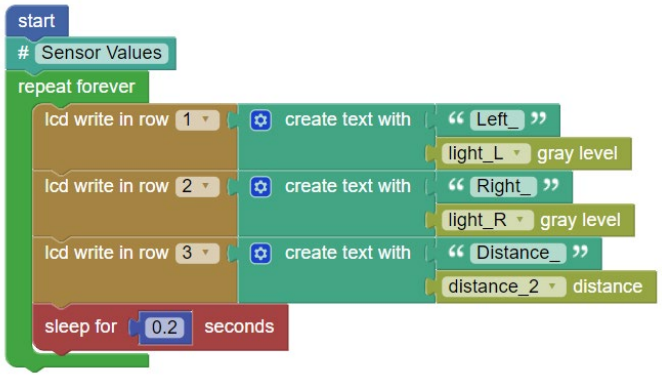


Drive Forward and Turn

- Add this code to our program
- Use a new start block

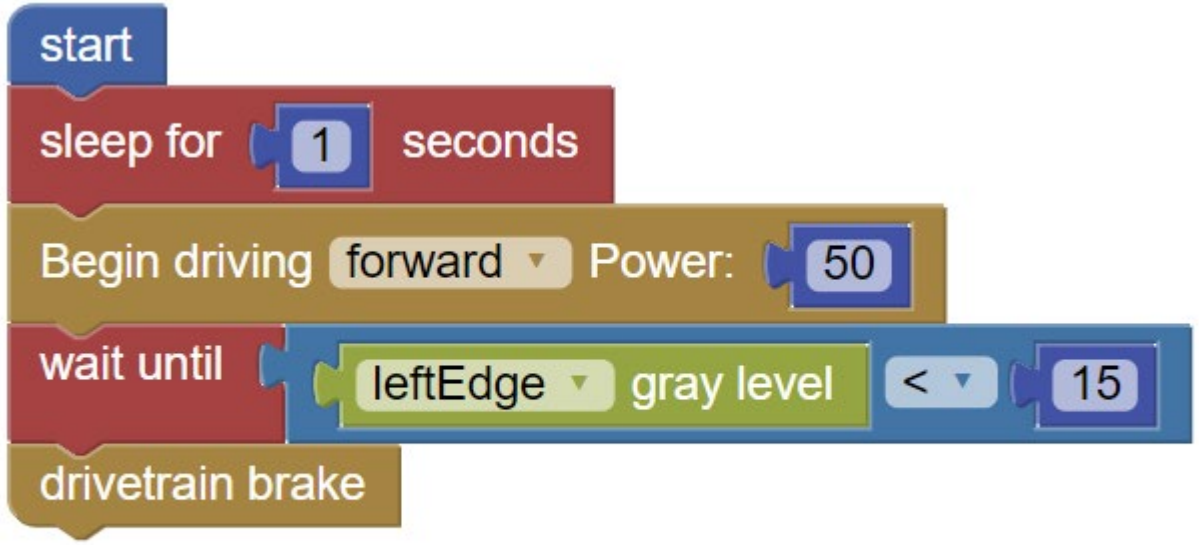


- Both the new code and the display code can be run at the same time



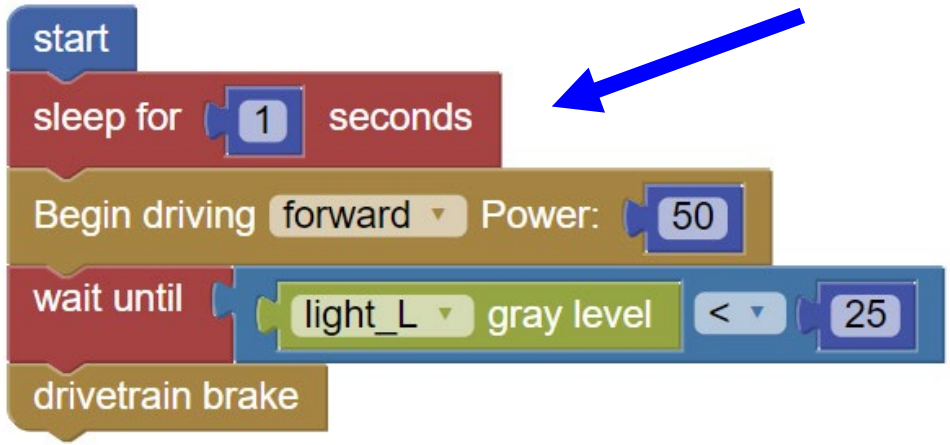
Find the edge of the playing field

- Delete the “forward” and “turn” blocks and replace with the code below



TIP

- Put a small delay at the start of the program
- Allows for initialization of sensors
- Without delay, initial sensor readings may be incorrect



Detect The Edge Of The Table

- Light sensor settings example
  - Off table = 5
  - On table = 45
  - Median threshold = (5+45)/2 = 25

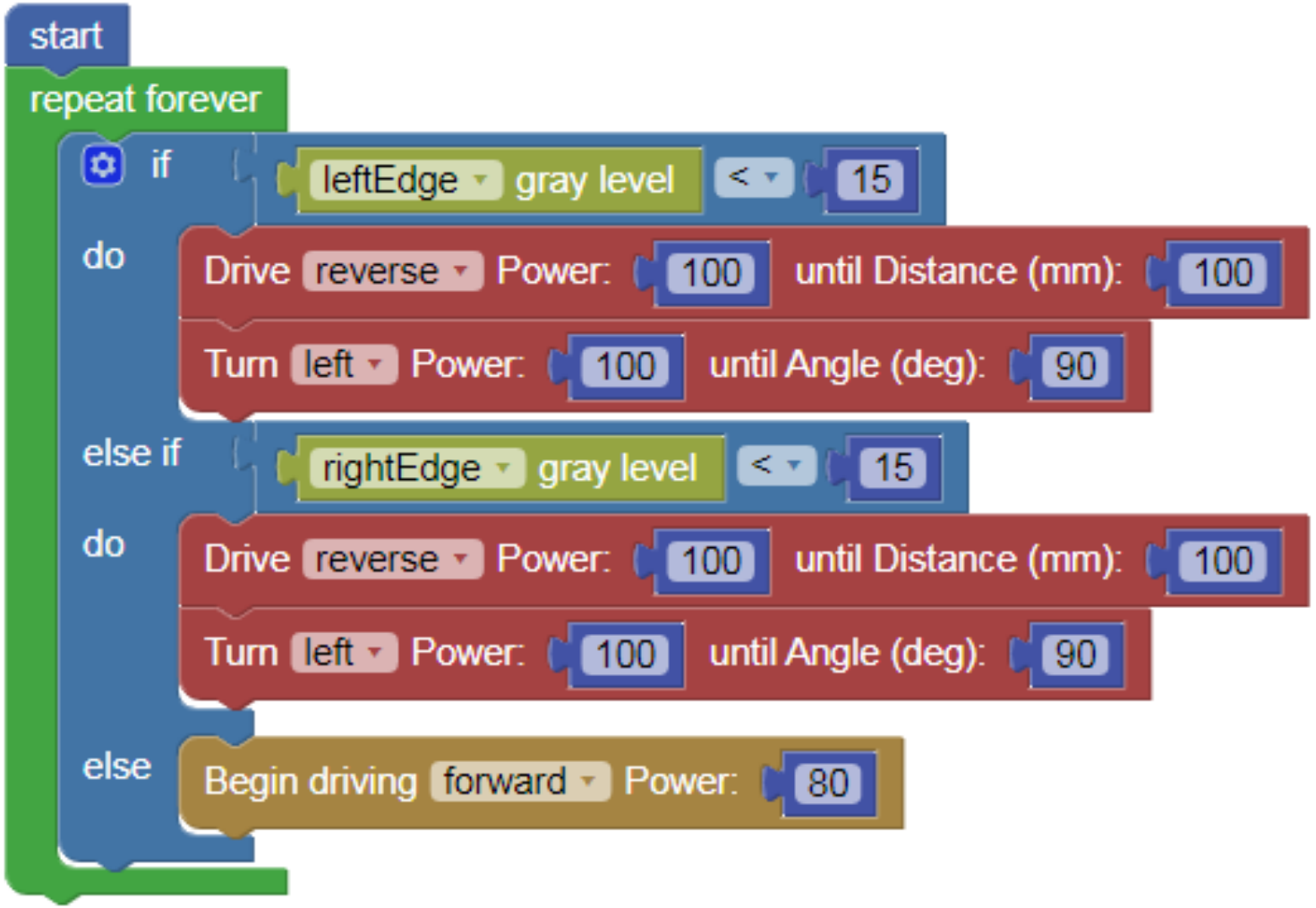
- Two cases
  - Light sensor reading > 25. On table.
  - Light sensor reading < 25. Off table.

Edge Problems

- If your robot stops too late or not at all:
  - Check that the move block is “Begin” not “Until for Seconds/degrees/rotations”
  - Check color sensor is shining a light
  - Check color sensor values on/off the table, the threshold average, and the “less than” inequality
  - Check light sensor port number (1 for left sensor)
- We only are looking at one light sensor – what if the other one goes off the table first?

Staying on the Table

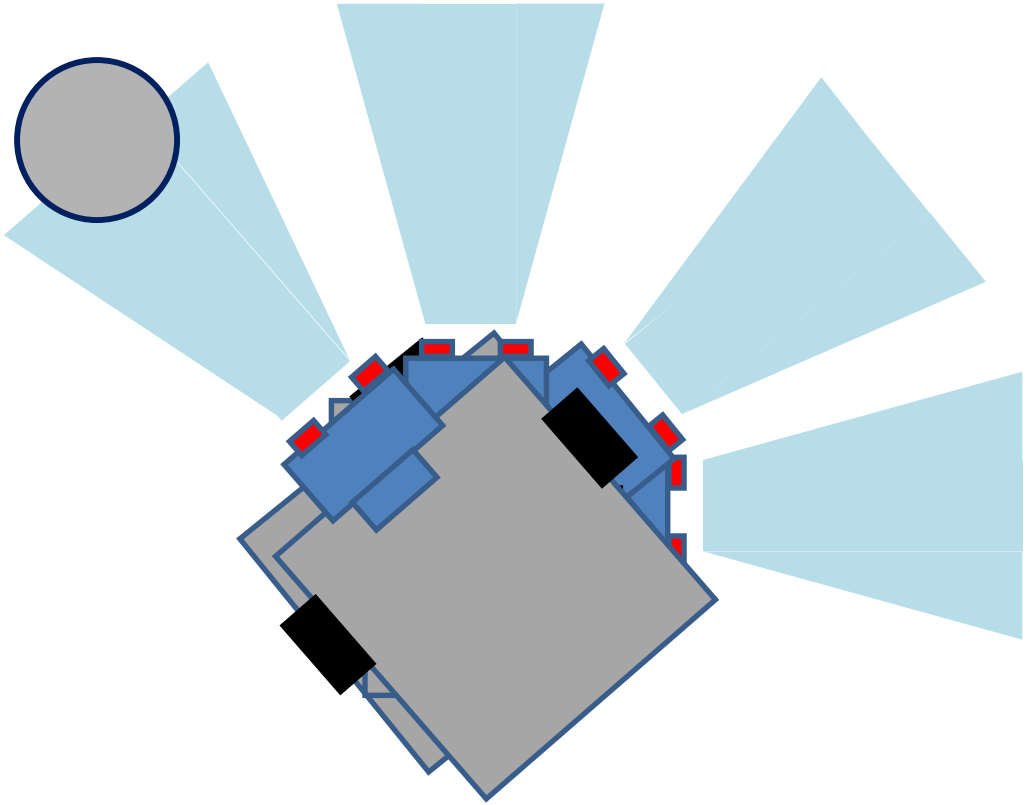
Back up if edge detected, else go forward



Detecting Objects

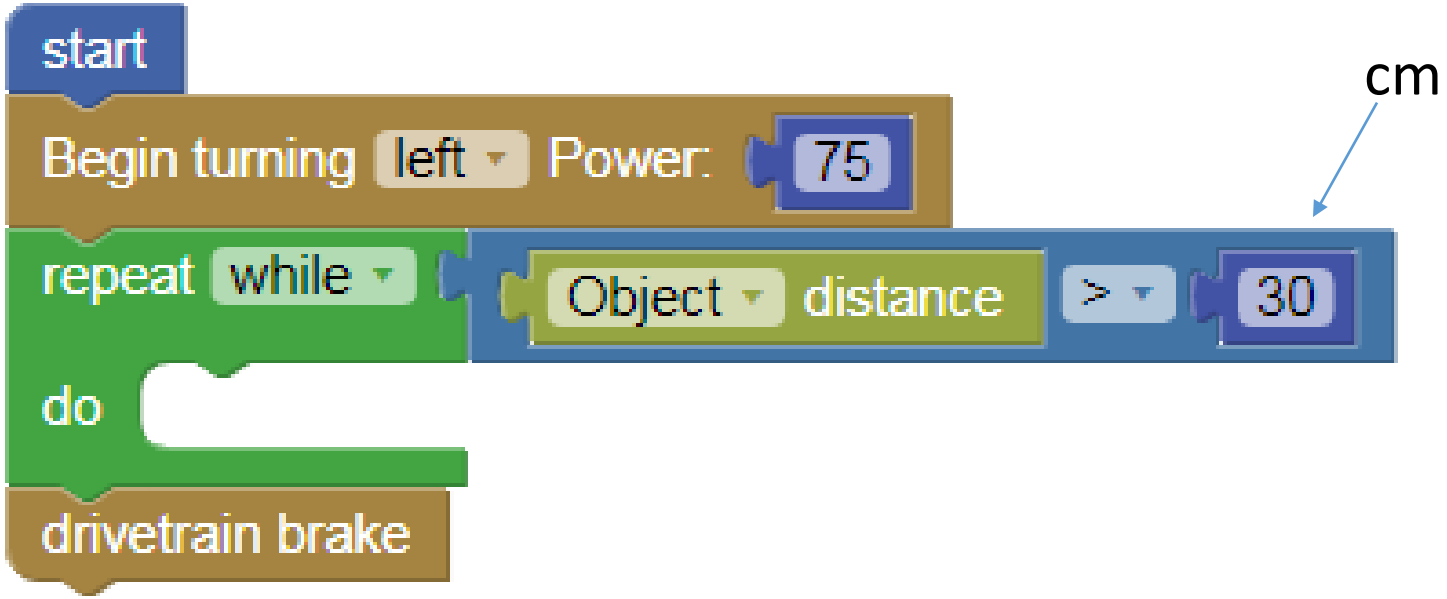
Finding the Bottle/Opponent

- Point Turn (Spin) robot
- Stop when object found



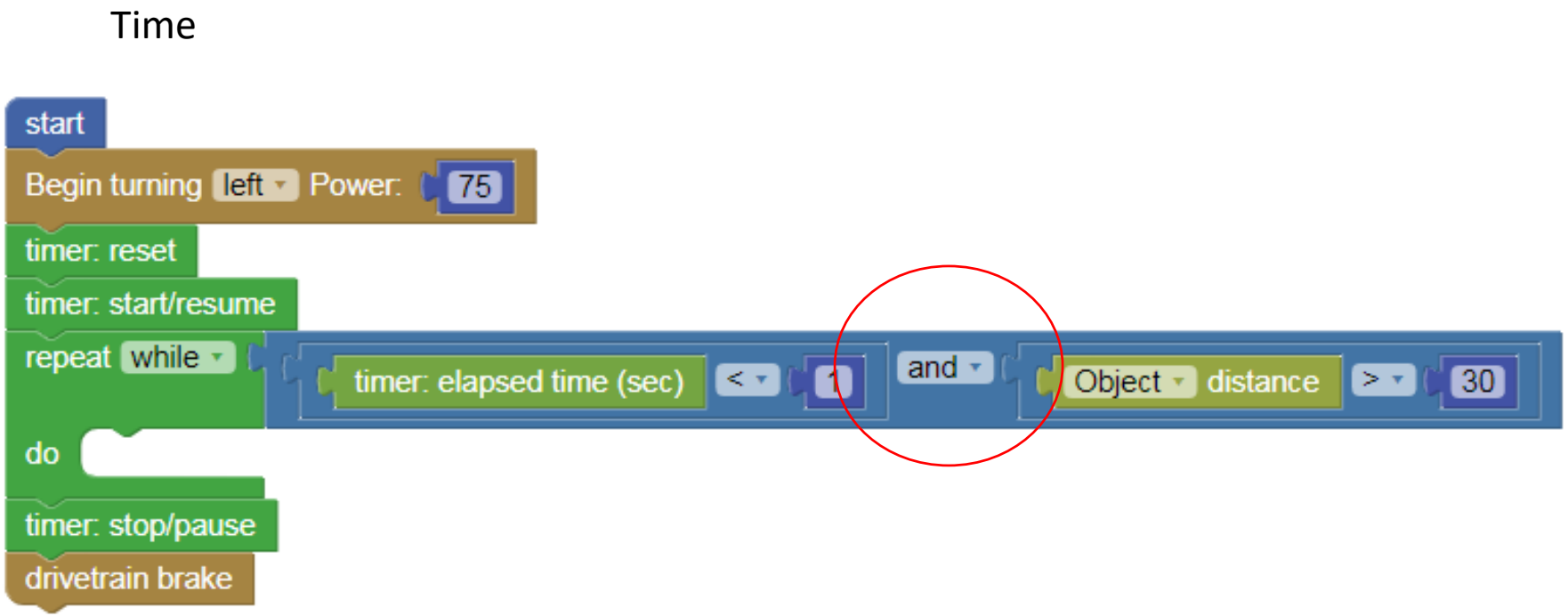
Finding Objects by Scanning

- Program the Robot to Scan (turn) until it detects an object



Limit the Scan

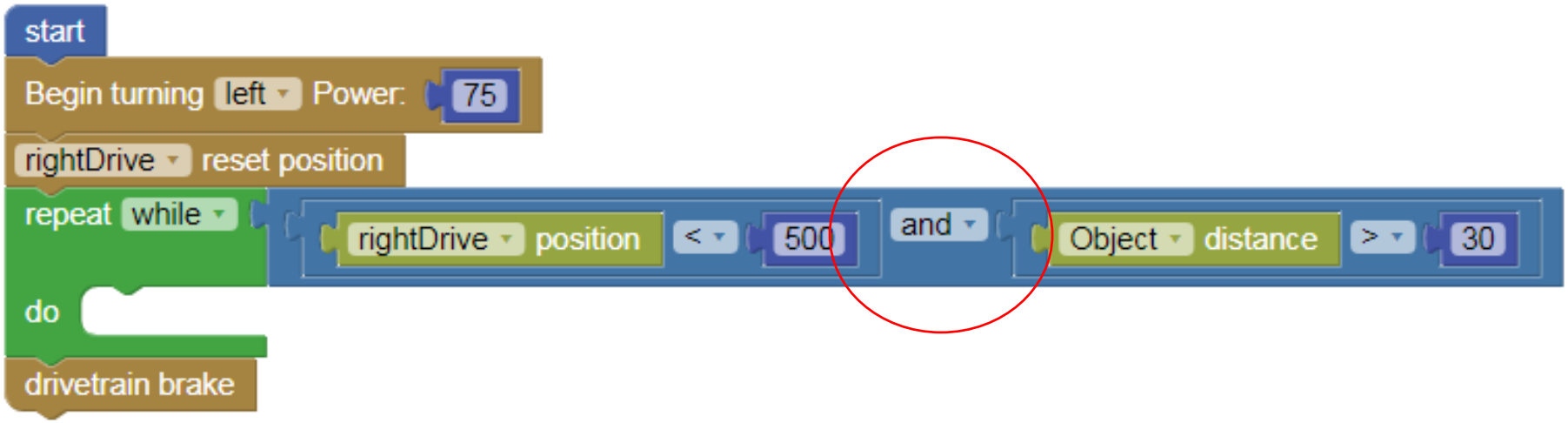
- Program the Robot to Scan (turn) until it detects an object or reaches a limit



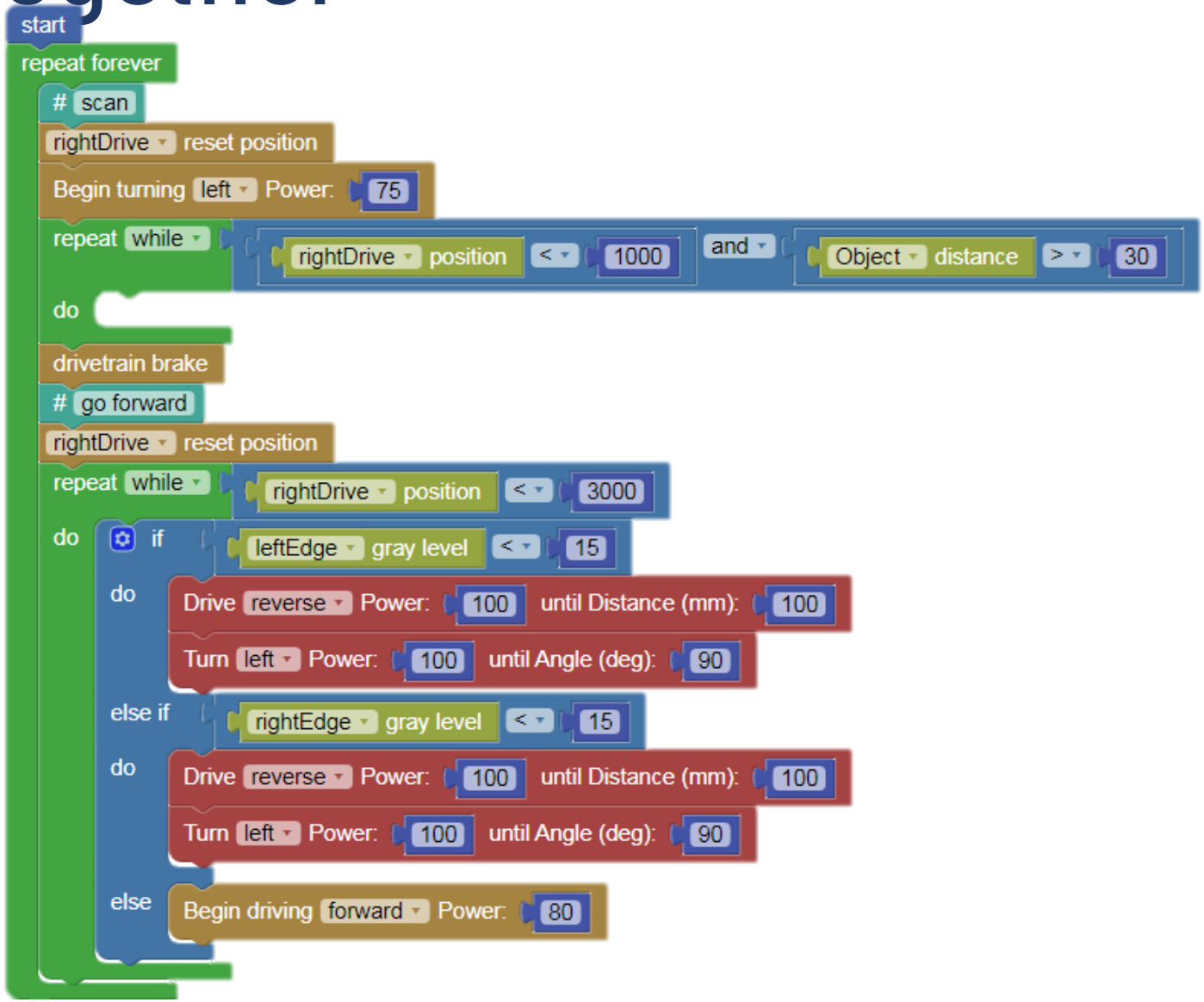
Limit the Scan

- Program the Robot to Scan (turn) until it detects an object or reaches a limit

Rotation

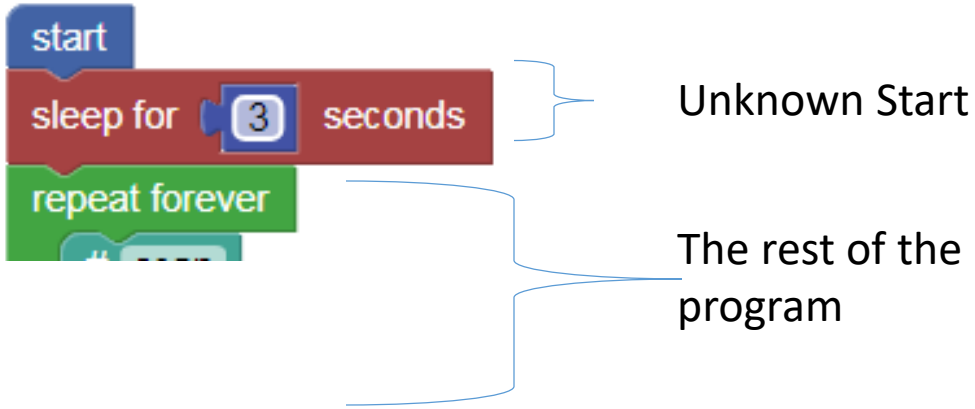


Putting It Together



Unknown Starts

- The way to start the robot moving is an Unknown Task that is unveiled 30 minutes prior to impounding robots.
- Example- a robot must wait 5 seconds after the game is started during which a judge will place a bottle on the table approximately equidistant from each robot.



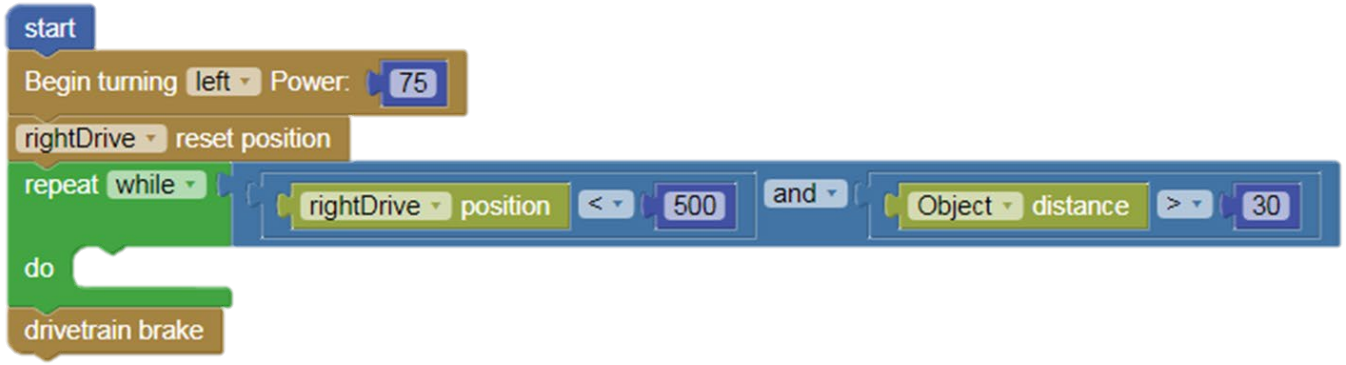
Building Functions

Functions

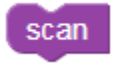
- Very large programs can be difficult to understand, navigate and use
- To alleviate this issue, the Robot Mesh Studio software has “Functions” to create custom blocks that can replace sections of your program
- Variables can be used to make the function more flexible

Functions

- For example, let's assume you have a section code that may look like this

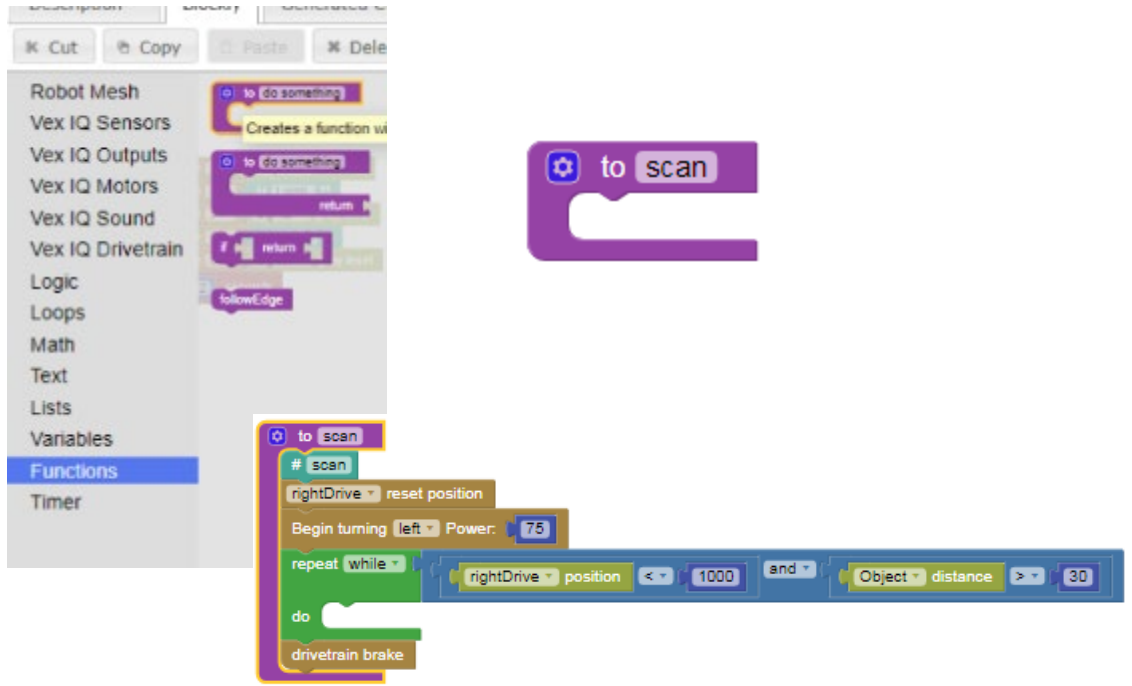


- Functions will allow us to convert this to a single block

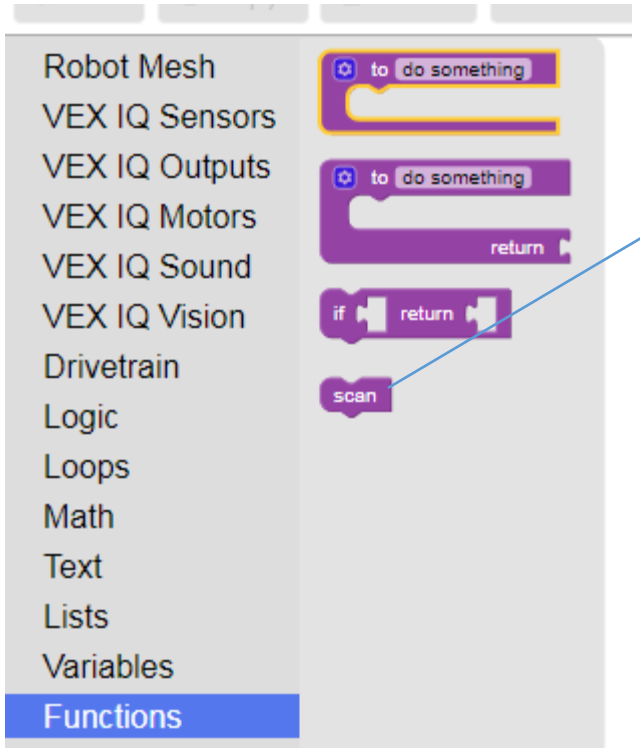


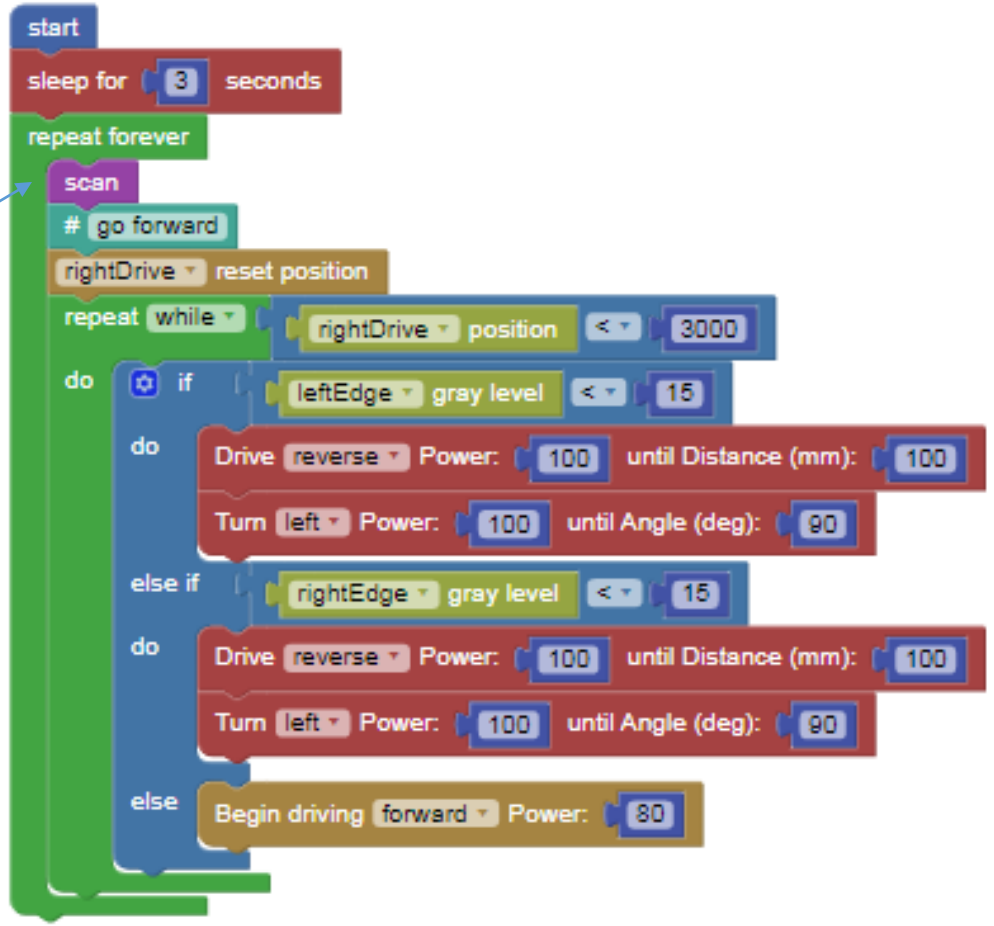
Create a Function

- Go to Functions Menu
- Get a “do something” block
- Name it “scan”
- Drag the blocks into the function block



Use the Function in the Program



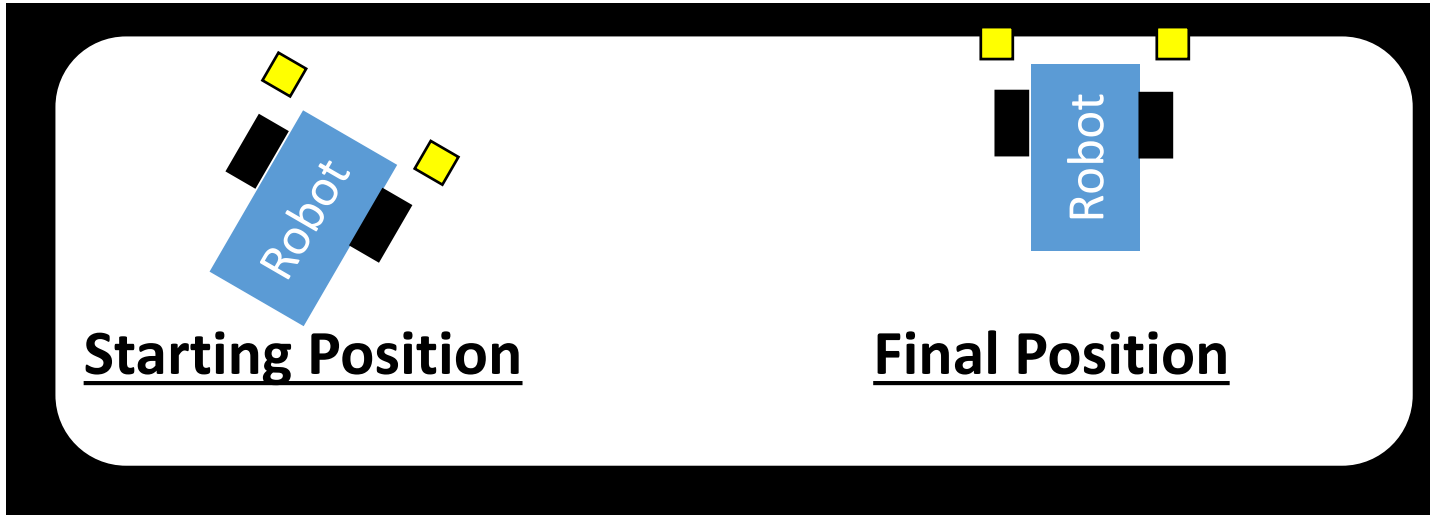


Improving Your Robot

- “Lock on” to an object- push until edge detected
- Align to an edge for a more complete push
- Turn different directions depending on which sensor detects the edge
- Vary the amount of turn (randomize)
- Look for objects in additional places in the program
- Go forward first, then scan for objects

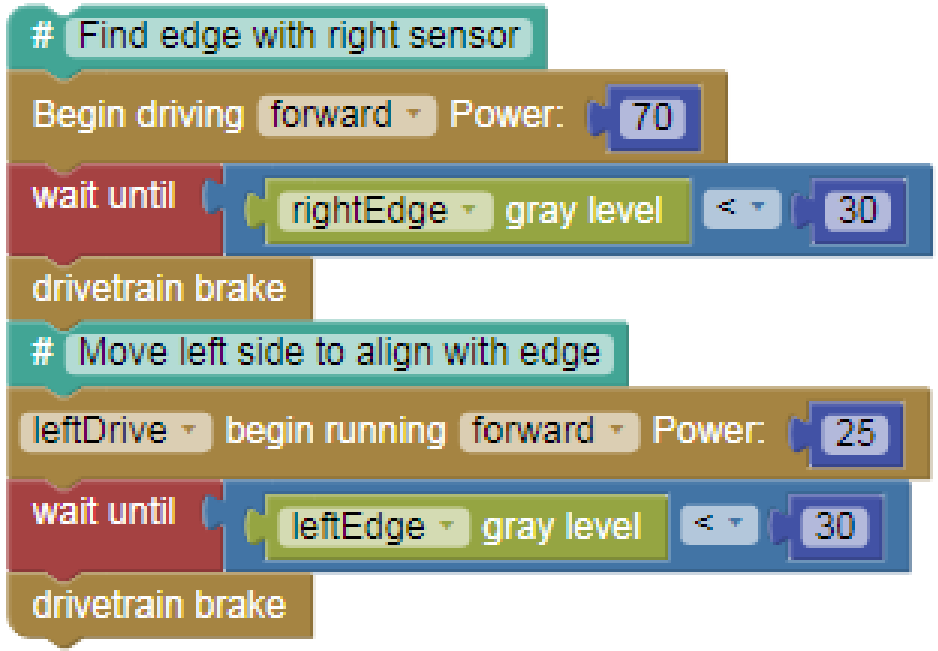
Aligning the robot to an edge

- In some situations we desire align with robot to an edge of the table as shown below
- Assuming the starting position below, how can we program the robot to reach the final position that is aligned with the edge of the table?



Aligning the robot to an edge

- Travel until color sensor #2 reaches the edge, swing robot until it is aligned with the edge



https://youtu.be/1xpdJ4JoV-Q

Physic of Sumo

Your robot will be more effective at pushing other robots off the table if you consider

- Mass
- Velocity
- Force
- Power

Mass

- Quantity of matter [kilograms]
- m=mass, ν=velocity, p=(linear) momentum

  $p=m*\nu$
- F=force, a=acceleration

  $F=m*a$

  •With a larger the mass, more Force is required for same acceleration

  •Where to add mass? Consider the center of gravity (should be low and inside wheel base)

Velocity

- Velocity = Speed and Direction [meter/second]
- m=mass, ν=velocity, p=(linear) momentum

  $p=m*\nu$
- If we increase the velocity, we increase the momentum
  - The larger the momentum, the greater the impact force will be applied to the opponent
  - How do we increase the velocity of the robot?

    Higher motor speed setting, larger wheels, gear-up

Force

- Pushing (Pulling) or Twisting
- Linear Force [Newton] or Torque [N-m]
- T=torque, r=wheel radius
- Force=Torque/radius
- How do we increase the Force applied by the wheels?
  - Increase the torque (lower gear ratio)
  - Decrease the radius (smaller wheels)

Power

- Power is Force *Distance / time [W=N*m/s]
- P=Power, F =Force, T=Torque, ν=velocity, ω=angular velocity

  $P=F*\nu$

  $P=T*\omega$
- How do we increase the Force and/or velocity?
  - Increase the power (Motor power)
  - Make sure batteries are fully charged!

Build a better Robot (later)

- Sturdy construction
  - At least 2 attachment points for each part so robot stays together
  - Compact design
  - Triple pegs on multiple beams
- Wheel base
  - Wide base for slow turns
  - Narrow base for fast turns
  - Which is best for stability of robot?

Robot Design cont.

- Wheels
  - How many tires? 2 or more?
  - Tires- rubber or Omni? (for traction?)
  - Size – large or small?
  - Placement of wheels

Robot Design cont

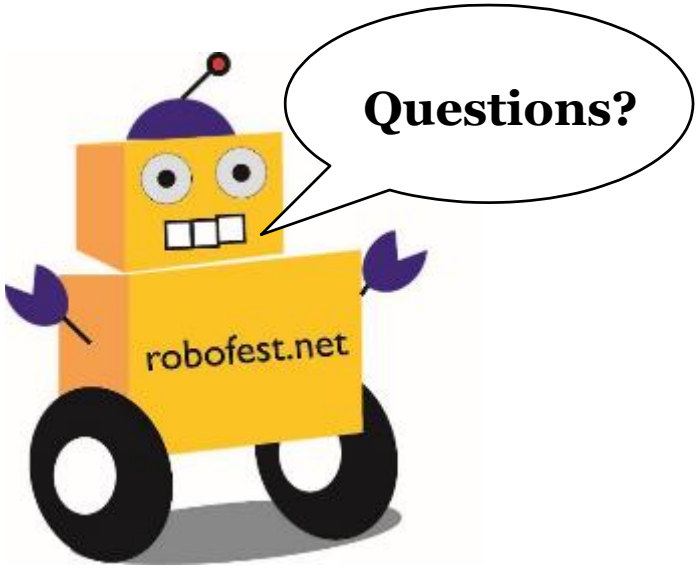
- Sensors
  - Placement of light sensors (height above table? How far in front or to side of wheels?)
  - Placement of sonar sensor (low or high?)
  - Use additional sensor(s)?
    - Touch sensor(s) to detect if an opponent is pushing your robot?
    - Touch sensor(s) to detect if you are pushing an opponent or the bottle?

Strategy (for the future)

- Try to lift an opponent off the table?
- Try to hide from an opponent (cloaking)?
- Bounce off an opponent if attacked?
- Add a motor to power a mechanical device to attack your opponent (try to flip your opponent over)?
- Better search for opponent? Find again if opponent moves before you get to him.

LAWRENCE TECHNOLOGICAL UNIVERSITY ROBOFEST

Little Robots, Big Missions



robofest@LTU.edu
LTU Computer Science

LAWRENCE TECHNOLOGICAL UNIVERSITY ROBOFEST

2020 Workshops

Sponsored by

DENSO

Lawrence Technological University

Computer Science