

ROBOCUP JUNIOR VICTORIA

Programming Spike Prime

- Scratch-based coding

ROBOT PROGRAMMING

What are the programming challenges and how are they best approached?

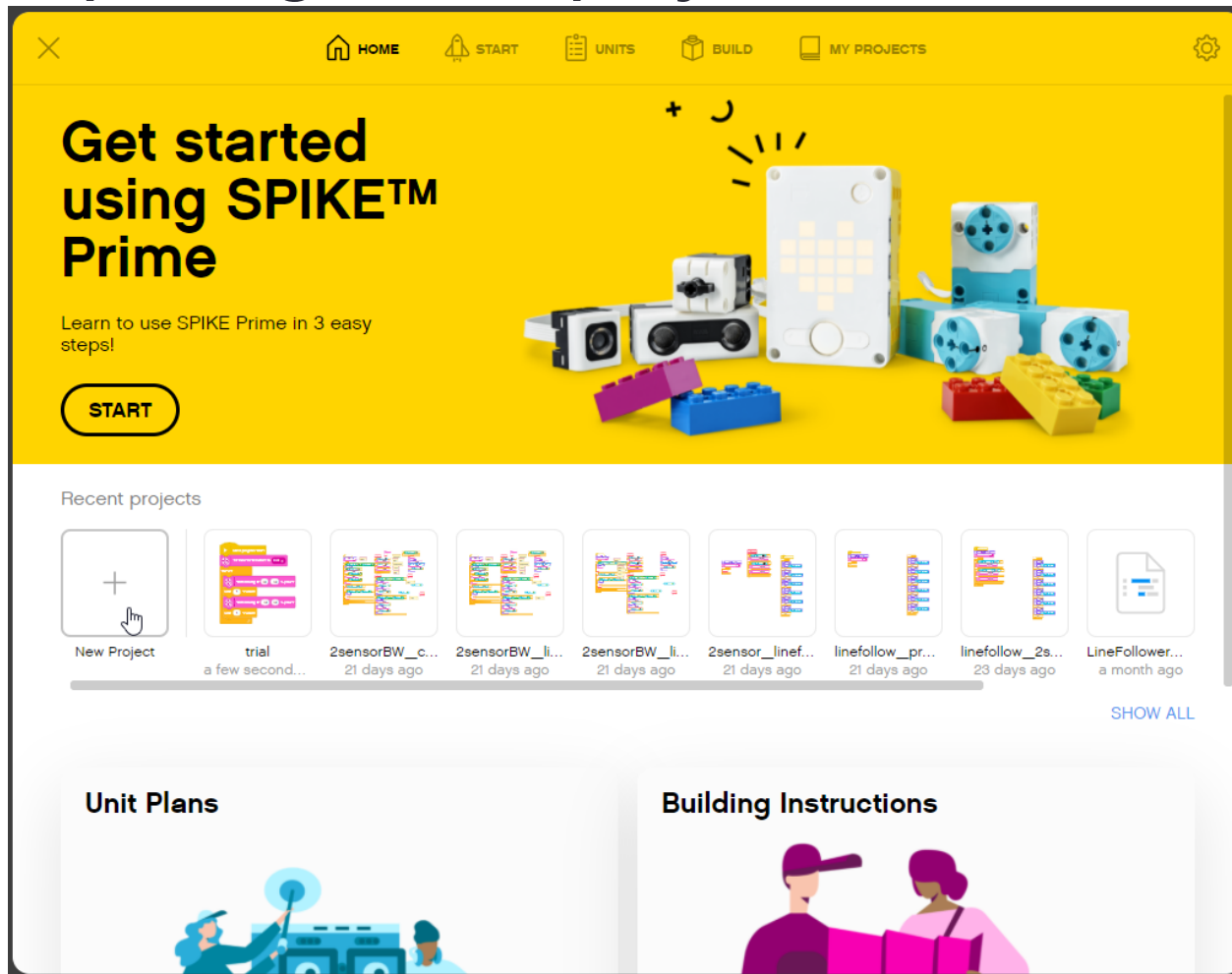
Rescue Robot Programming

- Rule #1 – Construction impacts programming
- Rule #2 – Programming impacts construction

- Simplicity is the key to successful programming, especially for beginners. If it looks more complicated than necessary, it probably is.

Introduction to the coding interface

Opening a new project



From the HOME page

- Tutorials
- Teaching unit plans
- Build instructions
- Links to your recent projects

New Project

- Open a new coding window
- Choose to code in either
 - WORD BLOCKS (Scratch)
 - PYTHON
- This presentation will focus on coding with word blocks

Introduction to the coding interface

Getting familiar with the “programming canvas”

The screenshot shows the LEGO Education SPIKE Prime - 1.3.4 software interface. The top menu bar includes 'File' and 'Help'. Below the menu is a toolbar with a home icon and a 'Project 1' tab. The main workspace is titled 'Motors' and features a sidebar with categories: MOTORS, MOVEMENT, LIGHT, and SOUND. A 'when program starts' block is connected to a 'run for 1' block, which is connected to a 'go shortest path to position' block, which is connected to a 'start motor' block, which is connected to a 'stop motor' block. The 'Motors' section displays a row of port connection settings for ports A through F. Port A is selected, showing a green dot and the number '5'. A tooltip 'Open Hub connection' is visible over the port A icon. A yellow box highlights the 'Help' menu item, and another yellow box highlights the port connection settings area.

LEGO Education SPIKE Prime - 1.3.4

File Help

Project 1

Motors

MOTORS

MOVEMENT

LIGHT

SOUND

when program starts

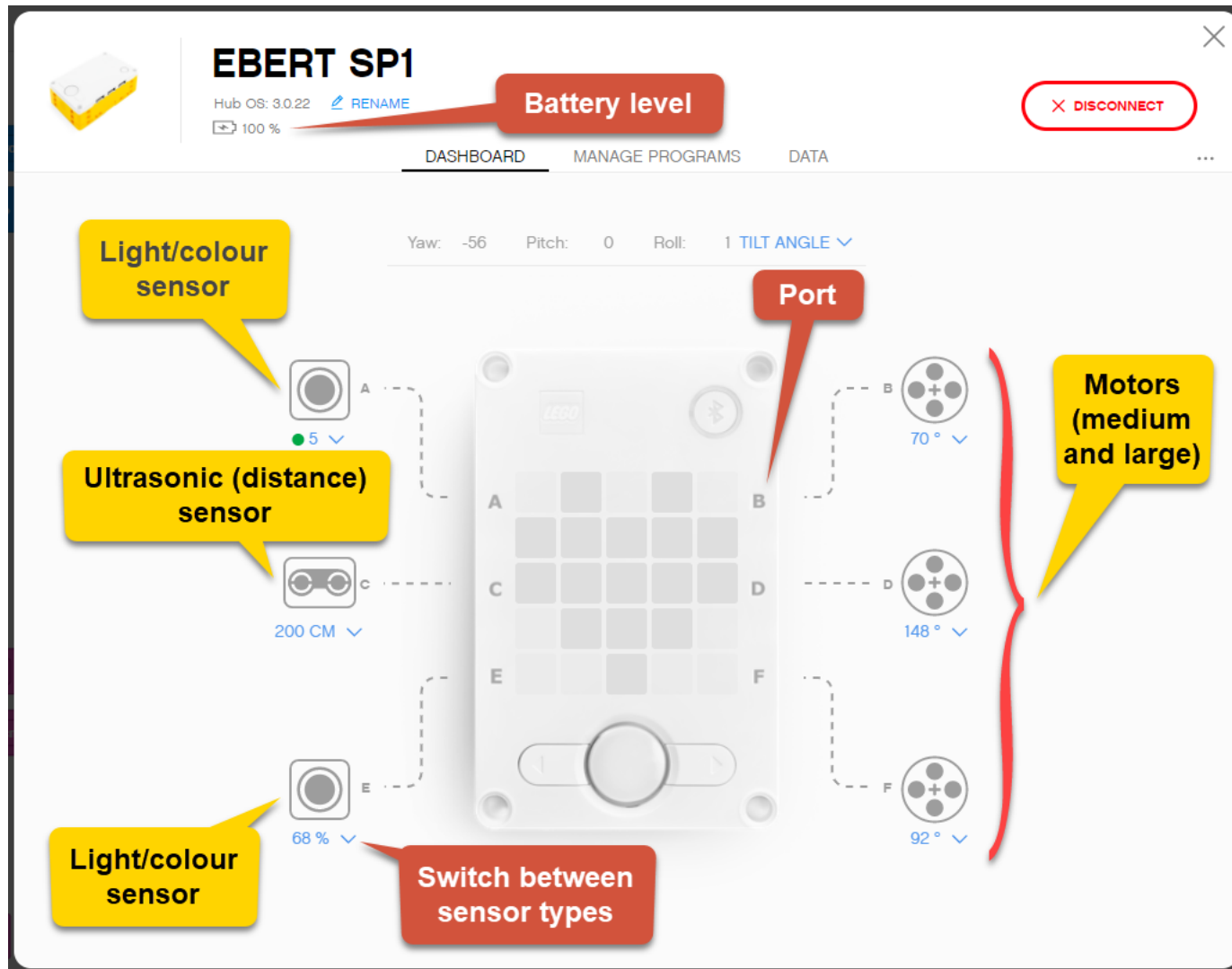
Open Hub connection

A 5 B 70° C 28 cm D 148° E 68% F 92°

Help

- Select “Settings” from the dropdown menu
- **Quick view of port connections**
- For details click on the brick icon to open the “Dashboard”

Introduction to the coding interface



DASHBOARD (if connected)

- Battery level indicator
- Port position for all connected inputs and outputs
 - All ports can act as either input or output ports
- Readings from each port
 - What is viewed can be set using the dropdown menu

MANAGE PROGRAMS

- Hub can hold up to 20 programs
- Options for deleting programs held on hub

Introduction to the coding interface

Palette of programming blocks

- Details about each programming block can be found under “Help”

Action - Motor control

Outputs – Sound and display

Flow control – Program control elements

Sensors – Inputs

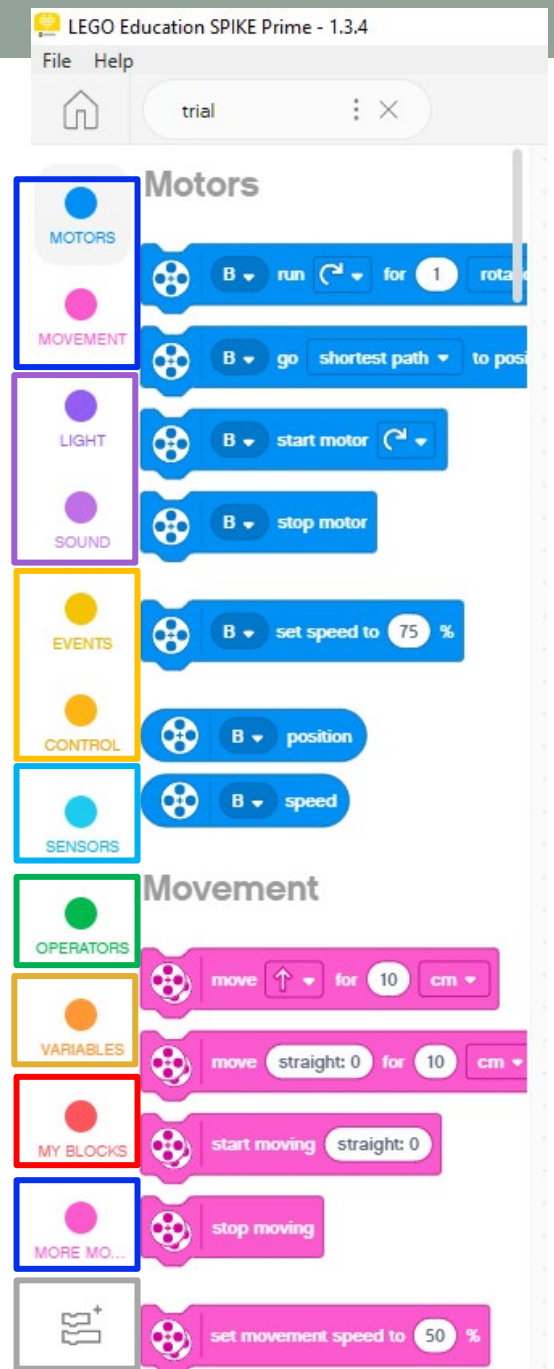
Operators – Mathematics and comparisons

Variables – data containers

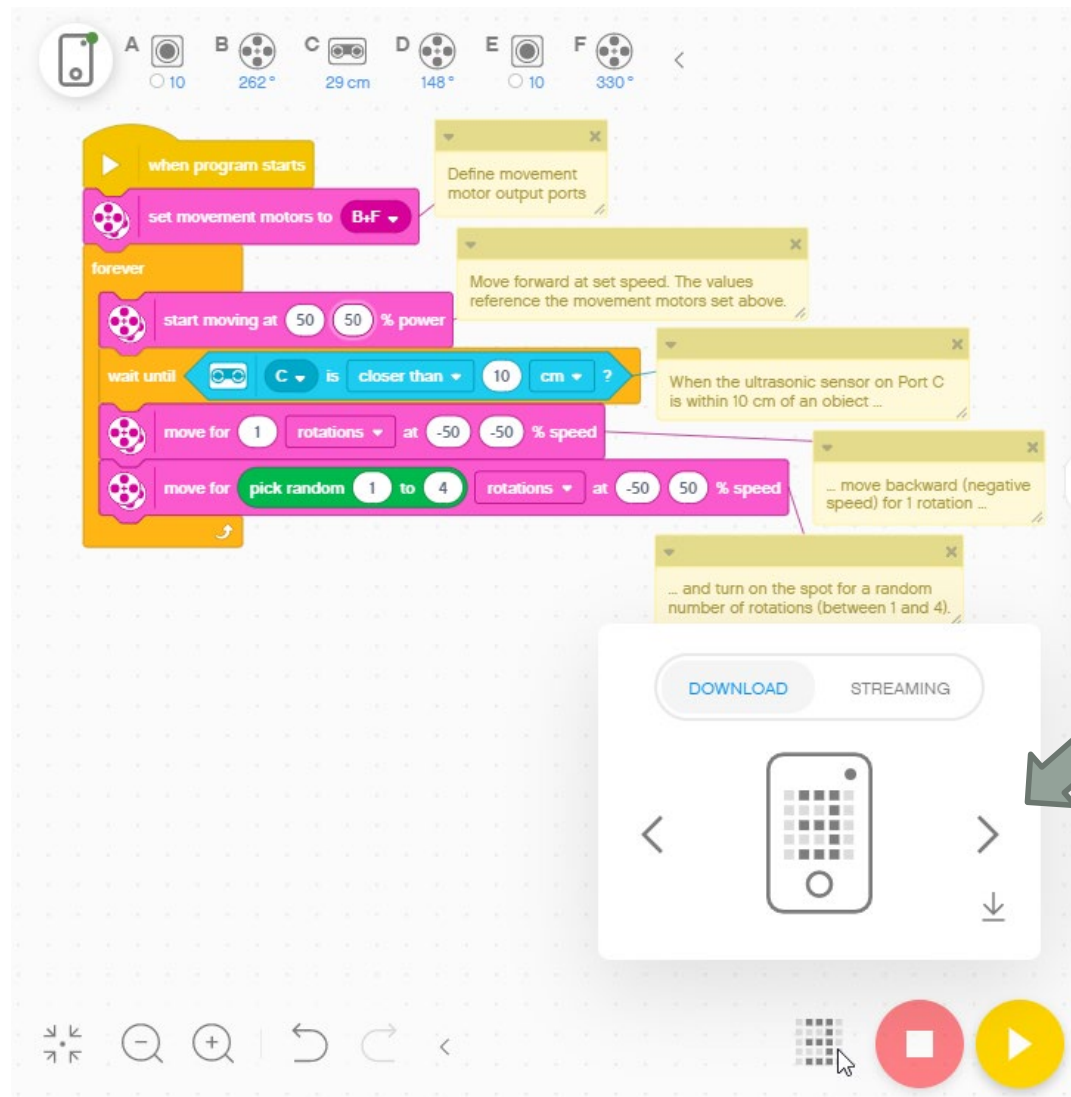
Functions – collecting blocks of code

Action – More movement blocks (from extensions)

Extensions – Additional code blocks



Example of simple program



Things to note:

- Encourage students to add comments to explain code (the comments are a bit overdone in this example)
- Encourage students to clean up the programming canvas
- When ready to download code to robot, click on the 5 x 5 grid at the bottom to select the program storage position (0 – 19) on the hub
- Click on the down arrow to download to hub

Getting help

The screenshot shows the LEGO Mindstorms EV3 software interface. At the top, there is a navigation bar with a close button, a 'Close lobby' button, and icons for HOME, START, UNITS, BUILD, and MY PROJECTS. A settings gear icon is on the far right. On the left, a sidebar menu is visible with categories: General, Language, Legal, Help (selected), Help Files, Interacting with the App, Hardware Overview, Types of Word Blocks, Word Block Description (expanded), Motor Blocks Category, and Light Blocks Category. The main content area displays the 'Help' page for 'Word Block Description'. It features a sub-section 'Motor Blocks Category' with a description: 'Motor Blocks either make the motors run or retrieve information from the motors. The *Motor Blocks* category contains the most common Motor Blocks.' Below this is the 'Run Motor for Duration' section, which states: 'This block will run one or more motors clockwise or anticlockwise for a specified number of rotations, seconds or degrees. The motor speed is set by the Set Speed Block. The default speed is 75%. 'Stall detection' is enabled by default. See the *Turn Stall Detection On/Off Block* for more information.' The 'Motor Go to Position' section is partially visible at the bottom, starting with 'This block sets one or more motors to a specified position. The motor can be set to run clockwise, anticlockwise or to take the shortest path to the specified position. The position ranges from 0 to 359 degrees.' The version number '1.3.4 (1.3.4)' is shown at the bottom left of the interface.

- “Help” isn’t as extensive as it was in the previous version of EV3 Mindstorms
- “Help” includes short descriptions of types of code blocks how to use individual code blocks

LINE FOLLOWING

What are the programming challenges and how are they best approached?

Rescue Robot Programming

Where to start?

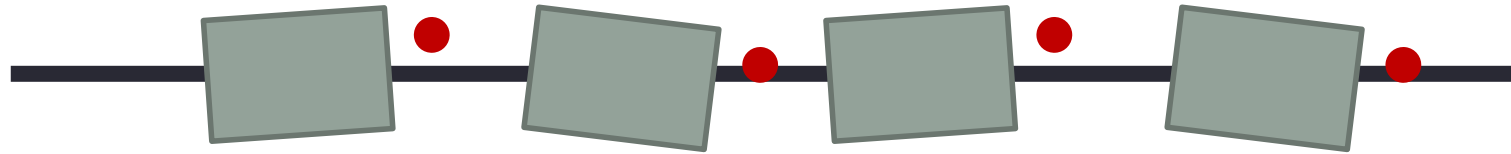
All Rescue divisions require the robot to:

- Follow a line
- Locate victim

One light sensor or two?

- Riley Rover Rescue (Victoria only) only needs one
- All other divisions require two

Principles of Line Following



Case 1

While True: # robot on

 If sensor sees white:
 turn right

 If sensor sees NOT white:
 turn left

Case 2

While True: # robot on

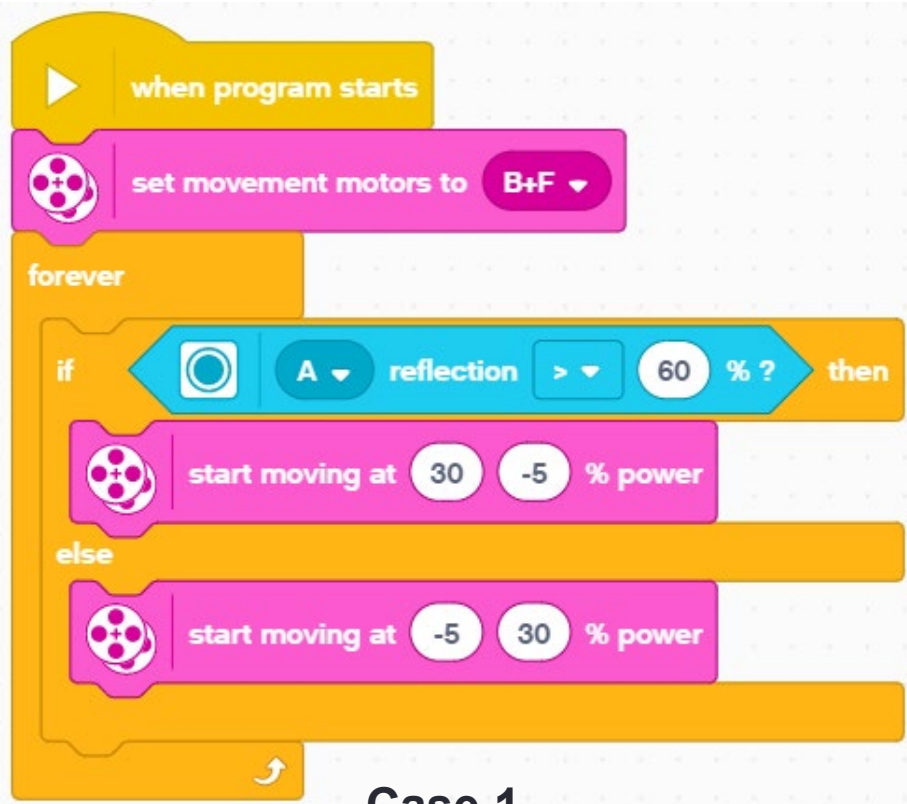
 While sensor sees white:
 Turn right

 While sensor sees black:
 Turn left

What does this look like in code?

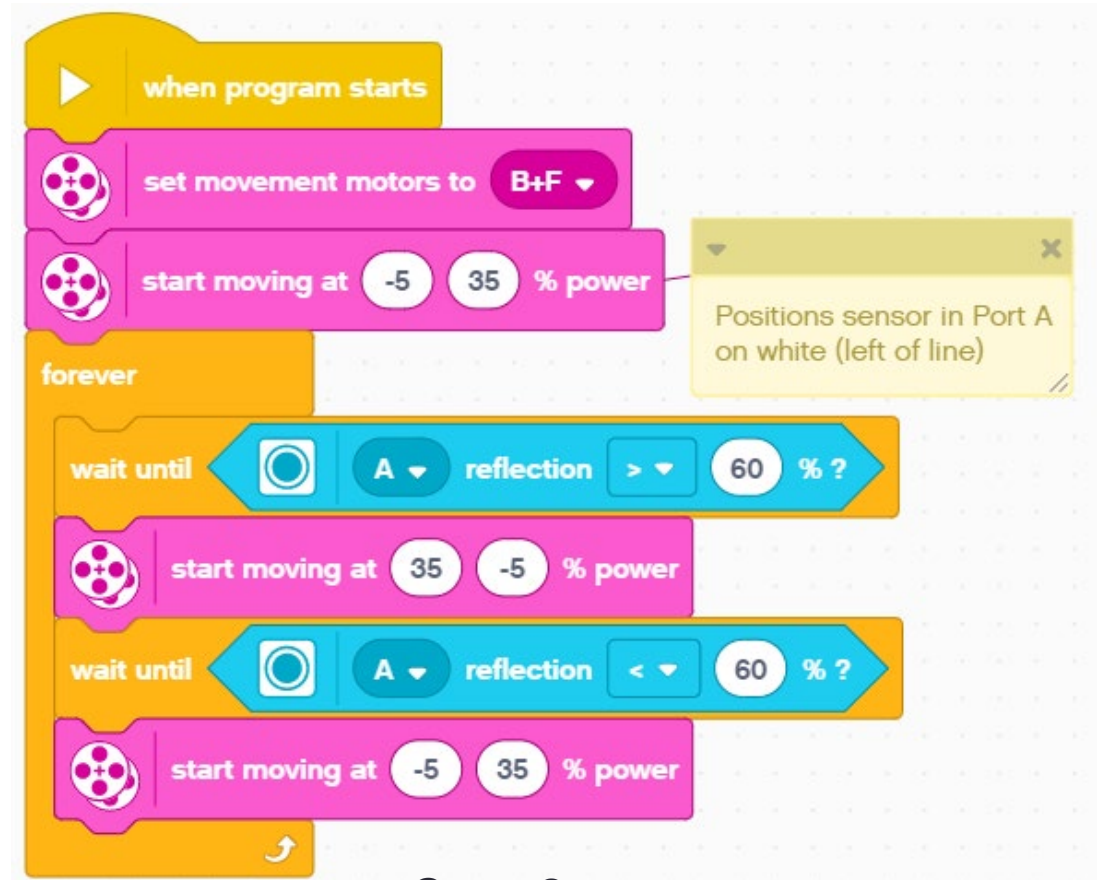
Please note: I have not included comments with all example codes. A good exercise for the students is for them to add comments. It forces them to think through the code and also helps them to see the important role that comments play.

Single Sensor



Case 1

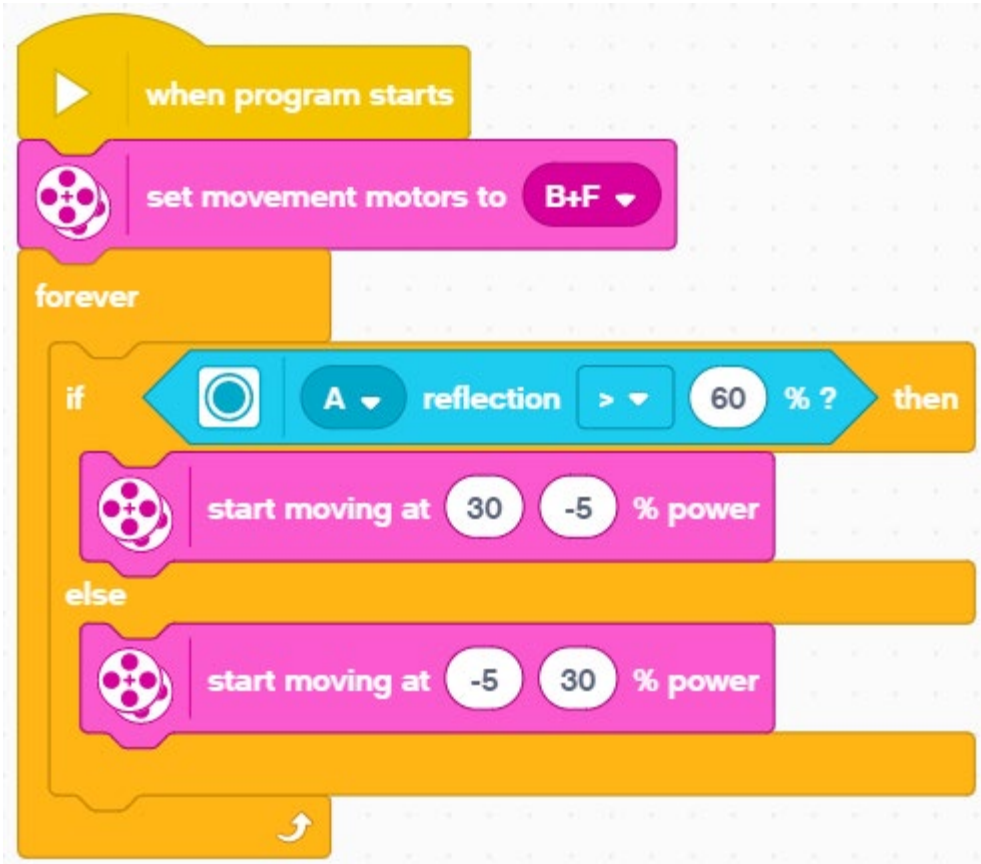
While True: # robot on
If sensor sees white:
Turn right
Else:
Turn left



Case 2

While True: # robot on
While sensor sees white:
Turn right
While sensor sees black:
Turn left

Single Sensor



I have used reflected light

- How do you decide what to set the reflected light intensity to?
- Could this be done using colour instead of reflected light?
- Why would you choose one over the other?
- Do motor speed settings matter?

Common pitfalls

- Movement motor ports not set
- Sensor/motor ports in program don't correspond to ports used on robot
- Motors mounted in reverse orientation
- This code may not work for your build and your light conditions

Adding a second sensor



Case 1

While True: `# robot on`

 While sensor_left sees black:

 Turn left

 While sensor_right sees black:

 Turn right

What does this look
like in code?

Case 2

While True: `#robot on`

 If sensor_left sees white AND sensor_right sees white:
 go straight

 If sensor_left sees white AND sensor_right sees black:
 turn right

 If sensor_left sees black AND sensor_right sees white:
 turn left

 If sensor_left sees black AND sensor_right sees black:
 `# Will this ever occur??? What should happen???`

Double sensor – Case 1

The image shows a Scratch script for a robot. It starts with a 'when program starts' block, followed by 'set movement motors to B+F' and 'start moving at 40 0 % power'. A yellow tooltip points to the 'start moving at' block with the text 'Positions sensor in Port A on black'. Below this is a 'forever' loop containing four blocks: 'wait until A reflection < 60 %?', 'start moving at 0 40 % power', 'wait until E reflection < 60 %?', and 'start moving at 40 0 % power'.

Case 1

While True: # robot on

While sensor_left sees black:

Turn right

While sensor_right sees black:

Turn left

I have used the “**start moving at**” block

- Could I have used the “**start moving with steering**” block?

A close-up of the 'start moving with steering' block in Scratch. The block is pink and contains the text 'start moving right: 20 at 50 % power'. Below the block is a circular icon representing a robot's steering mechanism, with a dial set to 'right: 20' and '+' and '-' buttons.

Could the robot look for white instead of black?

Also consider colour vs reflected light, cut-off values, motor speed settings

Double sensor – Case 2

The code is written in Scratch and is contained within a 'forever' loop. It starts with a 'when program starts' block that sets the movement motors to 'B+F'. The main logic is as follows:

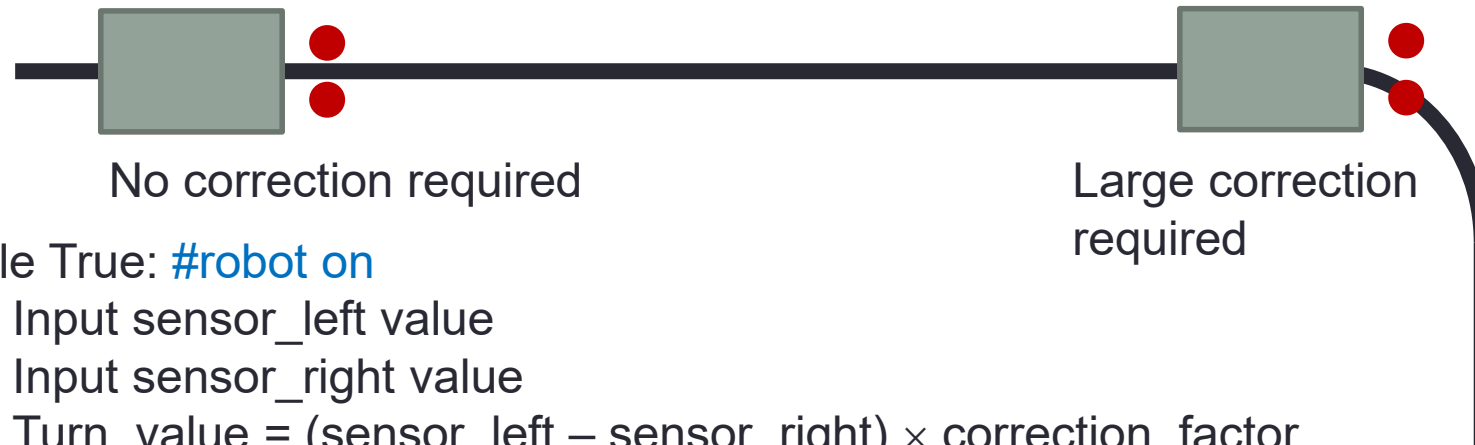
- Initial State:** A 'turn on' block is active.
- First Sensor Check:** An 'if' block checks if sensor A's reflection is greater than 60% and sensor E's reflection is greater than 60%.
 - Then:** A 'start moving at 30 30 % power' block is executed. A display block shows: "Both sensors see white, go straight".
 - Else:** An 'if' block checks if sensor A's reflection is greater than 60% and sensor E's reflection is less than 60%.
 - Then:** A 'start moving at 30 -15 % power' block is executed. A display block shows: "Left sensor sees white and right sensor sees black, turn left".
 - Else:** An 'if' block checks if sensor A's reflection is less than 60% and sensor E's reflection is greater than 60%.
 - Then:** A 'start moving at -15 30 % power' block is executed. A display block shows: "Left sensor sees black and right sensor sees white, turn right".
 - Else:** A 'turn on' block is active. A display block shows: "Presumably this is double black. What should happen here? Use the display to flag this occurrence."

At the very bottom of the code is the “black-black” situation

- Does your robot ever encounter this?
- Could this be used to detect green?
- What if a sensor value equals 20?
- Note the use of the display block to flag when the “else” situation occurs.
- Could you use the “stop moving” block to check the actual sensor values at the moment it occurs?

Proportional line follower

- What if I could adjust the amount the robot turns by how far off the line it is?
- Would this make for a much smoother run along the line?



While True: `#robot on`

Input sensor_left value

Input sensor_right value

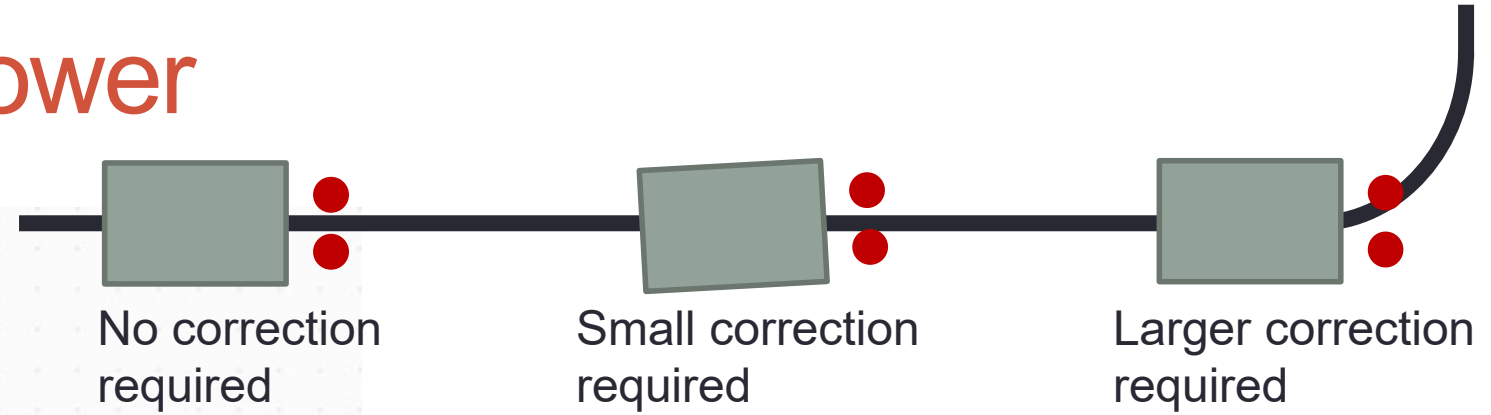
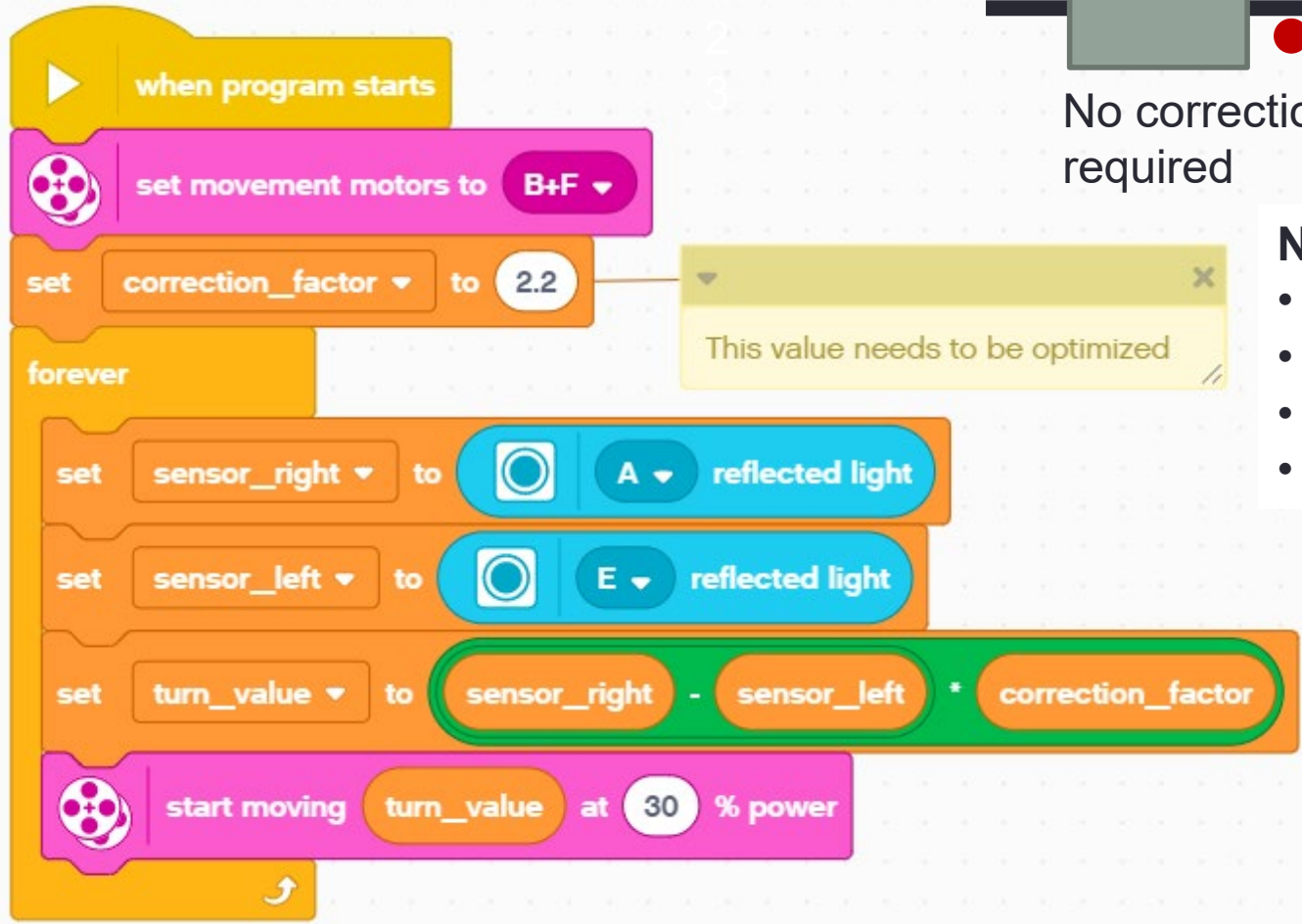
$\text{Turn_value} = (\text{sensor_left} - \text{sensor_right}) \times \text{correction_factor}$

Input turn_value to Move Steering block

`# if sensor_left > sensor_right, turn_value > 0 and robot turns right`

`# if sensor_left < sensor_right, turn_value < 0 and robot turns left`

Proportional line follower



No correction

- Both sensors on white
- $98 - 98 = 0$
- $\text{turn_value} = 0$
- Robot goes straight

Small correction

- Right sensor on edge (83)
- Left sensor on white (98)
- $83 - 98 = -15$
- $\text{turn_value} = -15 \times 2.2 = -33$
- Robot turns slight right

Large correction

- Right sensor on white (98)
- Left sensor on black (28)
- $98 - 28 = 70$
- $\text{turn_value} = 70 \times 2.2 = 154$ (NB: max is 100)
- Robot turns hard left

RESCUING THE VICTIM

What are the programming challenges and how are they best approached?

Rescue

Riley Rover (Victoria only)

- Push victim completely out of chemical spill

Primary Rescue

- Push victim completely out of chemical spill
- Exit chemical spill and recapture line (new in 2021)

Secondary Rescue

- Control and release victim in an upright position outside the swamp
- Exit chemical spill and recapture line

Open Rescue

- Lift victim onto rescue platform in upright position
- Exit chemical spill and recapture line

Entering chemical spill and detecting the victim

How can the robot detect chemical spill tile?

- Highly reflective tape at entrance

How can the robot detect the victim?

- For Riley Rover, it doesn't need to but more efficient if it does

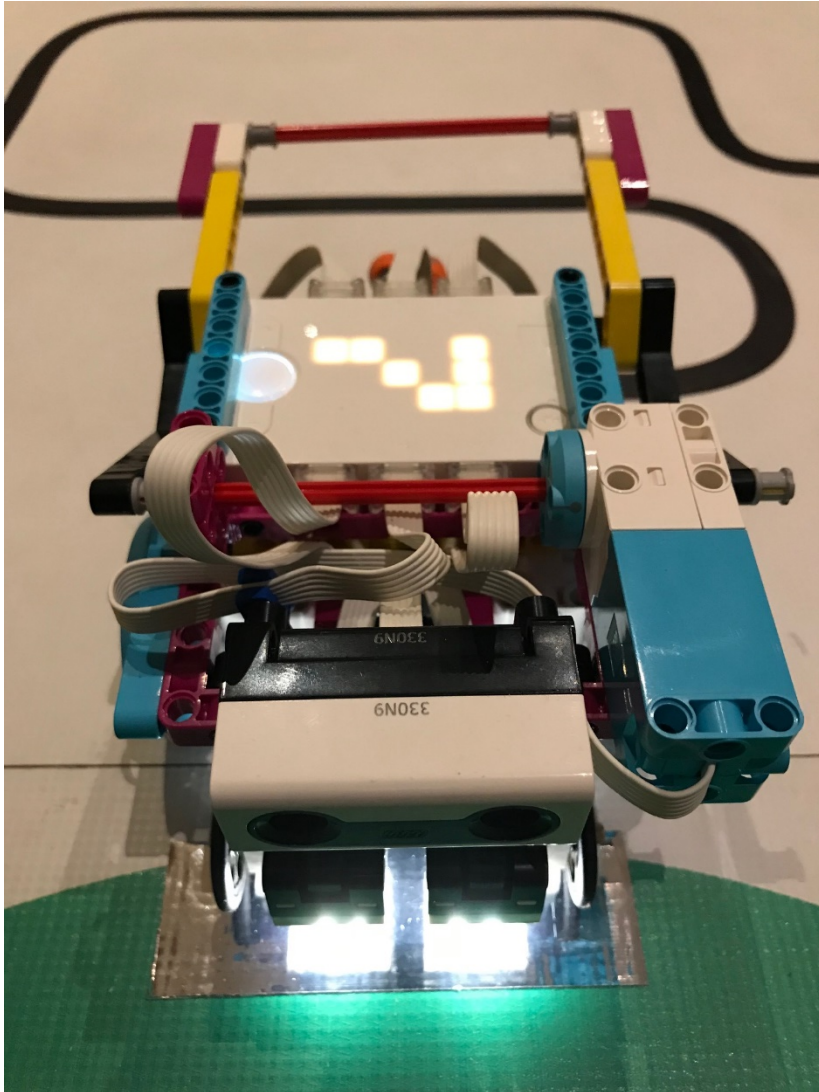
How can the robot control the victim?

- What level of control is needed for each division?

How can the robot exit the spill and regain the line?

- Not required for Riley Rover

Think through problem – Detecting the spill tile



Detecting chemical spill

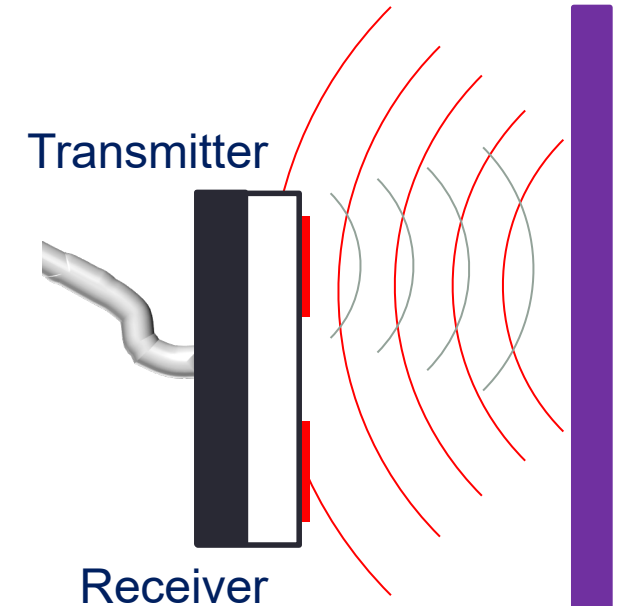
- Does the reflected value for the foil tape differ from white?
- Does the tape have a colour value?
- Are measured values for reflected light and colour consistent?
- With Spike Prime it is possible to measure the raw red, green and blue values individually. Can this help distinguish the silver foil from white?



Think through problem – Finding the victim

Finding victim

- Ultrasonic sensor measures distance from object
- Where is the best place to position the robot to begin checking?
- What is the maximum distance the victim could be from the robot?
- What happens if the sensor is too close to the victim?
- Is the ultrasonic sensor able to detect curved surfaces as easily as flat surfaces?
- What if alignment is not perfect?



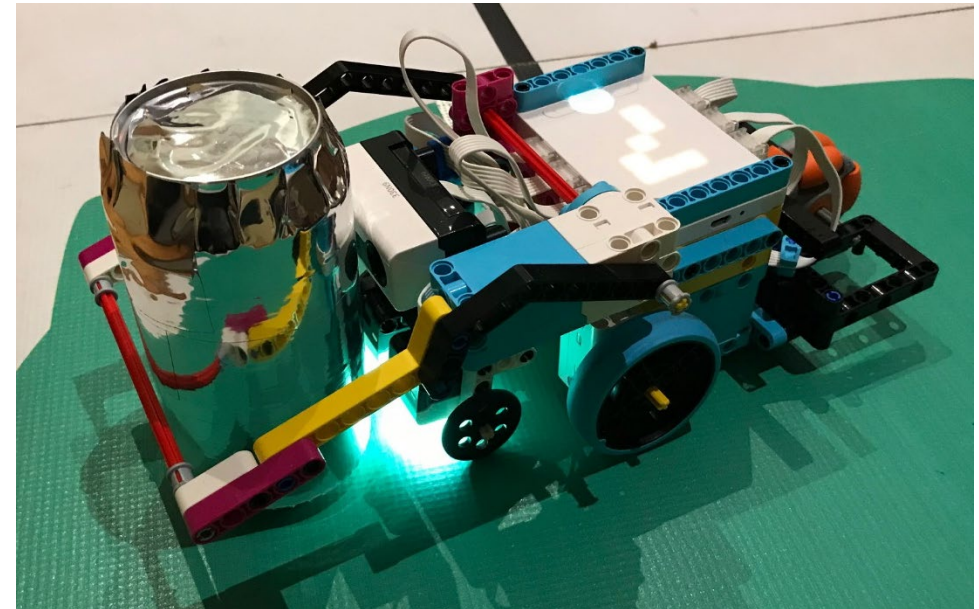
Thinking through problem – Rescuing the victim

Controlling the victim (secondary)

- Grabber mechanism
 - **Positive:** Can get away with not being perfectly lined up since grabber will gather victim in
 - **Negative:** Can be bulky and add to length of robot resulting in course navigation problems
- Cage mechanism
 - **Positive:** Much more compact
 - **Negative:** Need precise alignment

Rescuing the victim

- Push/drag rescue capsule to white and release
- What happens if the robot has missed or lost control of the rescue capsule?



Think through problem – Finding exit and regaining the line

Finding exit (3 options – are there others?)

1. Use single light sensor line follower algorithm to follow edge of green until reflective tape is reached
 - Where would you position the robot relative to the edge?
2. Random or systematic “walk” until reflective tape is detected
3. Record and retrace steps

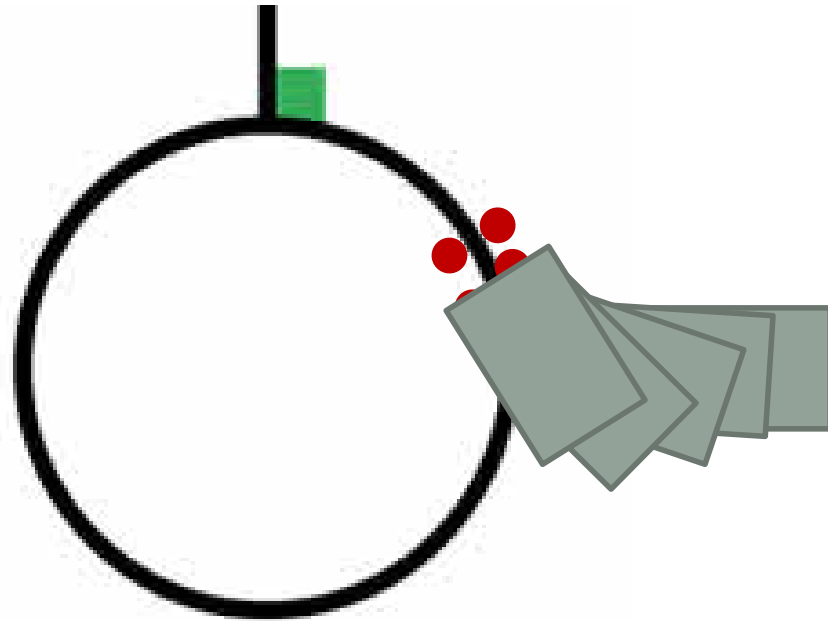
ADDITIONAL CHALLENGES

Detecting intersections

Navigating around water tower

Detecting green at intersections

Robots should turn in the direction of the green marker



Detecting green at intersections

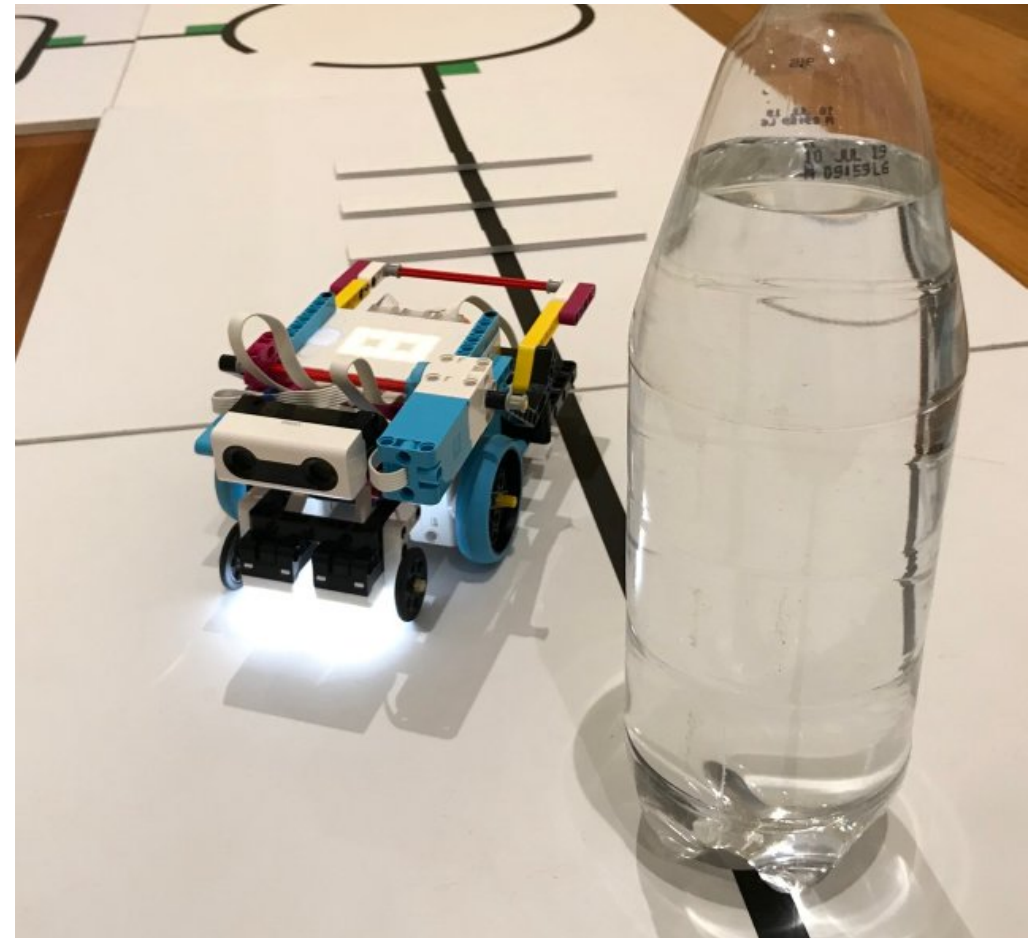
NOTE: The green on Rescue challenge mats used in the Victorian competitions are detected as green by Lego EV3 colour sensors. This is not necessarily the case for all Rescue mats and may not be true if sensors change. There is nothing in the national rules that specifies the shade of green.

Thinking through the problem:

- Does the robot turn correctly using a basic line following program? Always? Most of the time? Rarely? Never?
- Do the colour sensors detect the “green” as “green”?
 - Spike Prime colour sensors seem to be more reliable than EV3 sensors
- What are the reflected light values when over the green squares? Are they unique individually? As a sum? As a difference? Can you use any of this to reliably detect green?

Navigating around water tower

- What should be used to detect the water tower (Ultrasonic? Touch?)?
- It is relatively easy to pre-program a route around tower, but ...
 - What happens if robot isn't perfectly aligned with water tower?
 - How does the robot know when it has found the line again?
- Must recapture the line on the same tile to get points



Troubleshooting

Check that:

- Port settings in program **match ports** on robot
- Movement motors have been defined correctly
- **Actual reflected light/colour readings** correspond to values set in program
- Motor power values are reversed in program if **motors are in reverse** orientation
- **All conditional statements** are set correctly
- **Sounds/displays** aren't affecting program flow

And a few tips

- Program in small increments
- Use the brick displays or sounds to help identify if a particular part of the program is being
- Use My Blocks to help organise more complex programs
- **Save programs with significant changes as a new version, so stable older versions are not lost**
- On competition days make sure that programs loaded on the brick are functional programs (don't leave rubbish programs on the brick that could be run by accident)
- Make sure that battery is charged