ROBOCUP JUNIOR VICTORIA

Programming the EV3

EV3 Classroom app

(Scratch-based)

ROBOT PROGRAMMING

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

Rescue Robot Programming

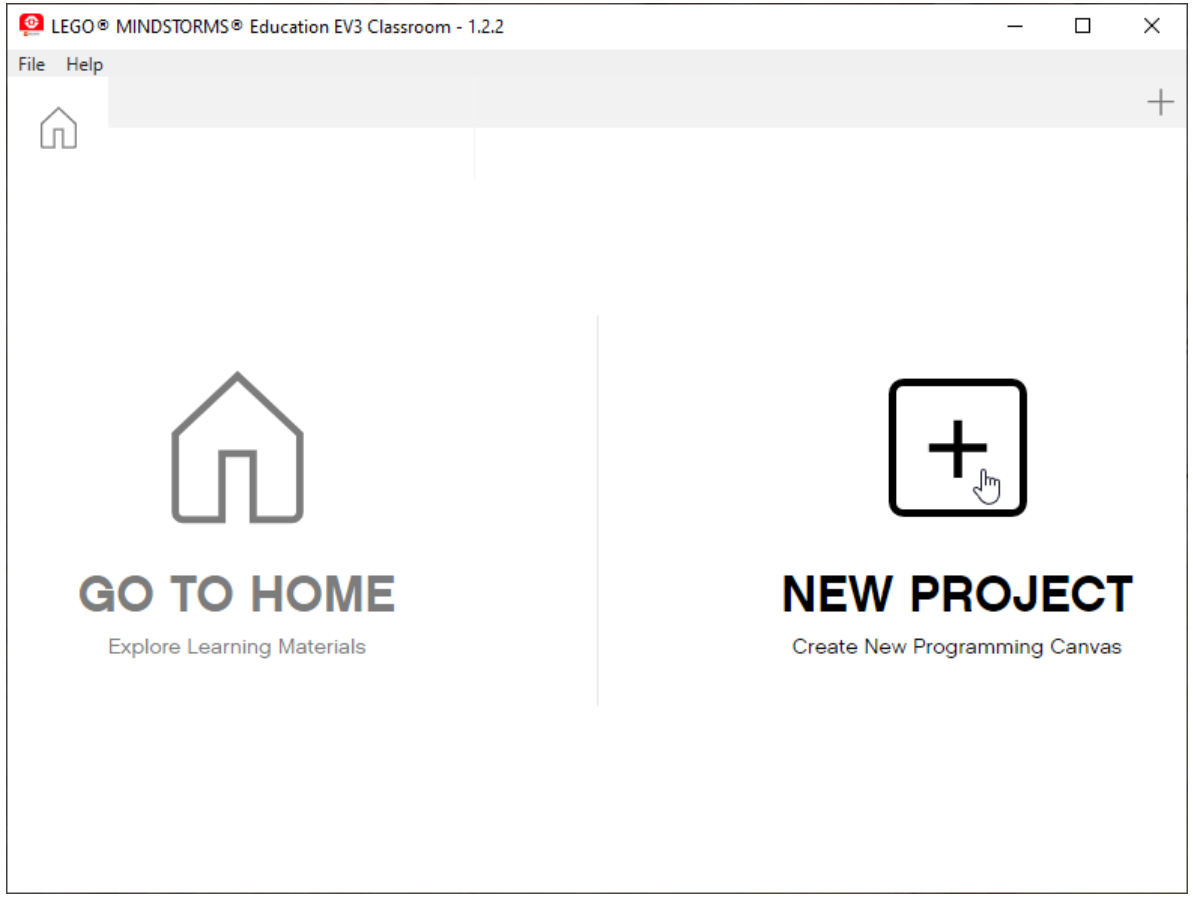
- Rule #1 – Construction impacts programming
- Rule #2 – Programming impacts construction

- Most Rescue robots are programmed using Lego Mindstorms graphical programming languages for EV3

- 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 (EV3 Classroom)

Opening a new project



GO TO HOME

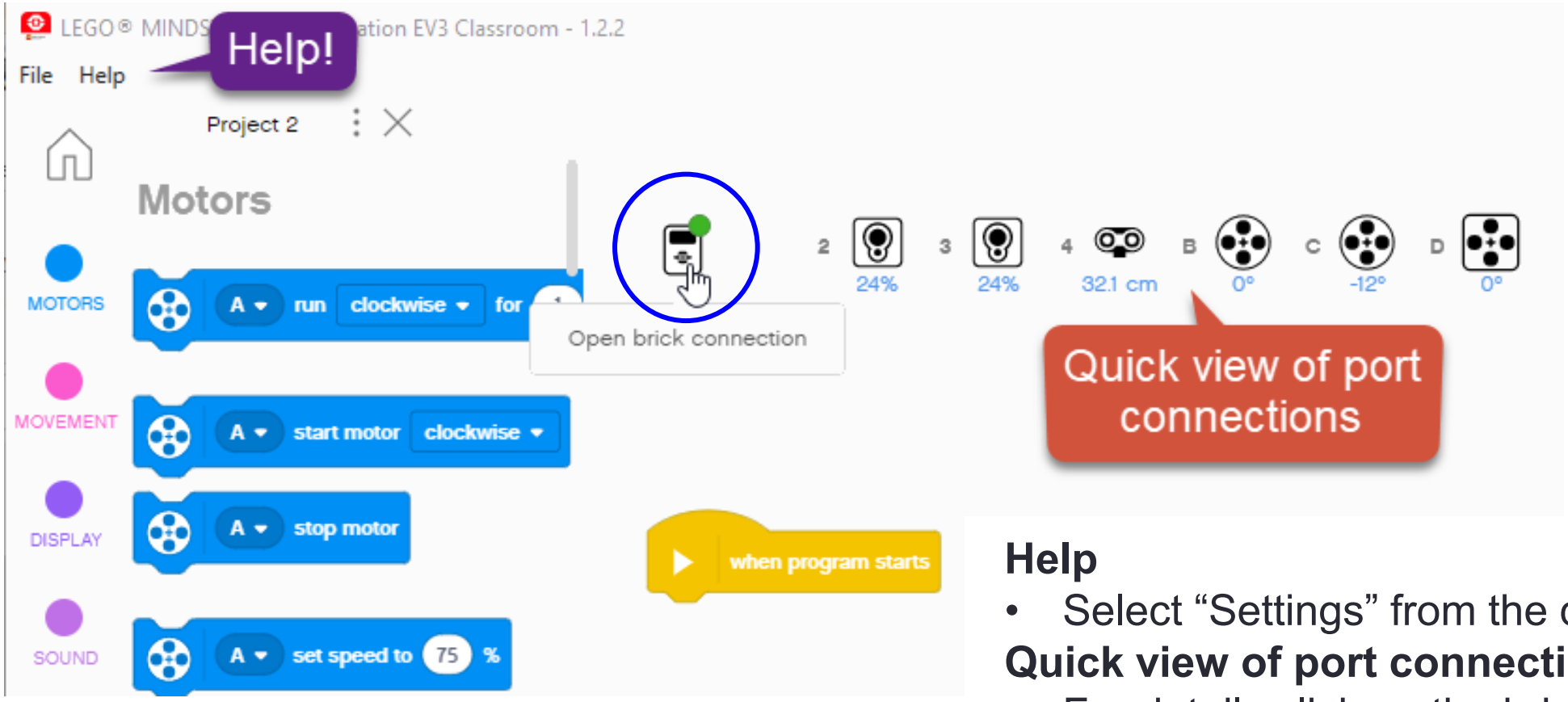
- Tutorials
- Teaching unit plans
- Links to your recent projects

NEW PROJECT

- Open a new coding window

Introduction to the coding interface

Getting familiar with the “programming canvas”



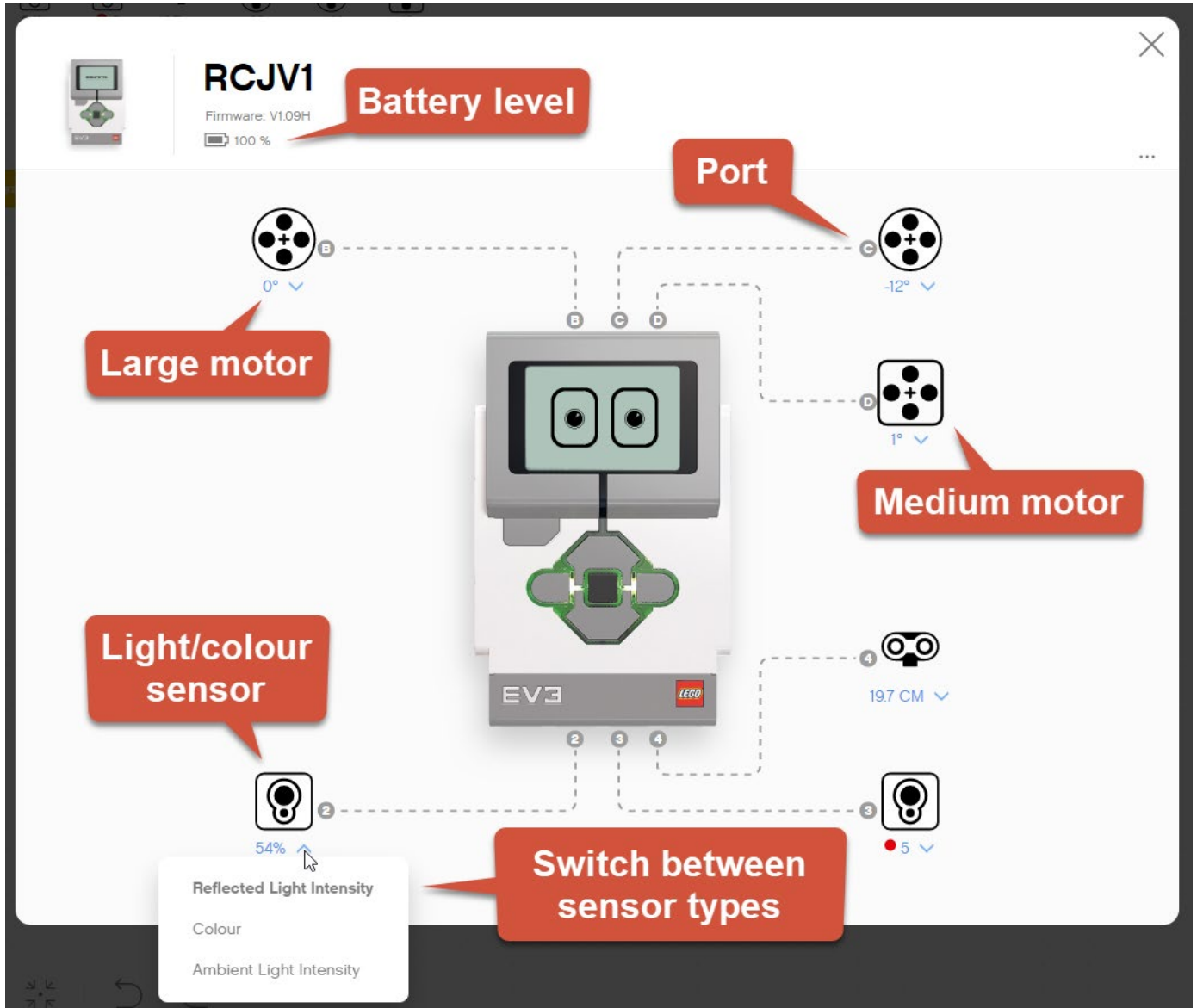
Help

- 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
- Position of all connected inputs and outputs
- Readings from each port
  - What is viewed can be set using the dropdown menu

Values can also be read directly on the brick

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

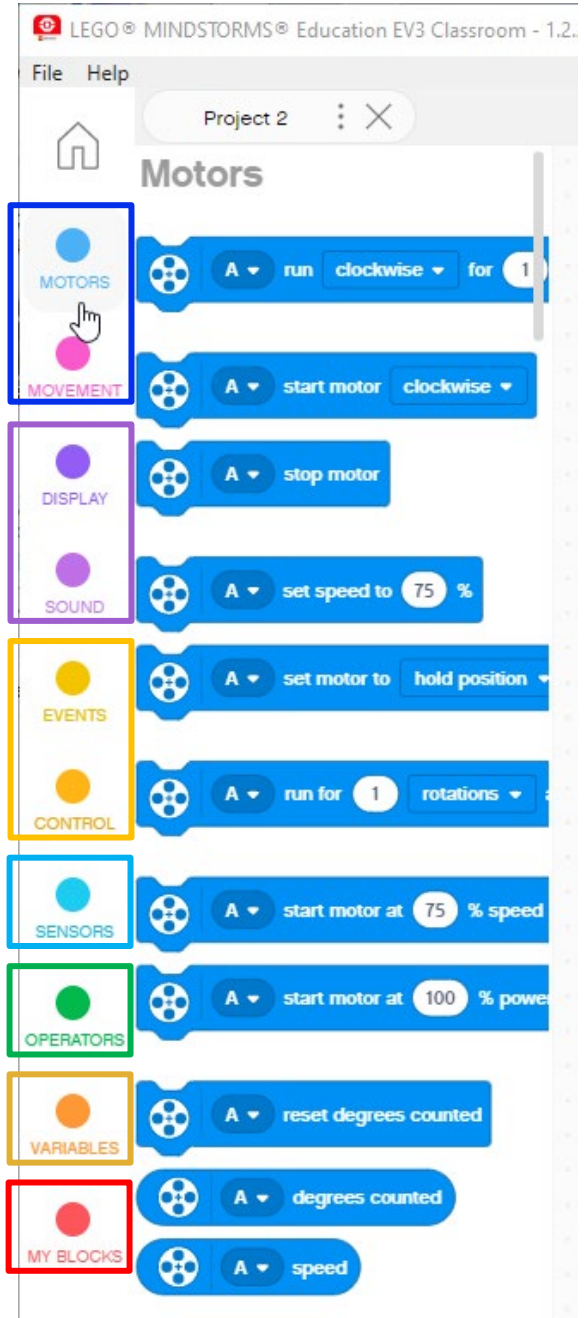
Flow control – Program control elements

Sensors – Inputs

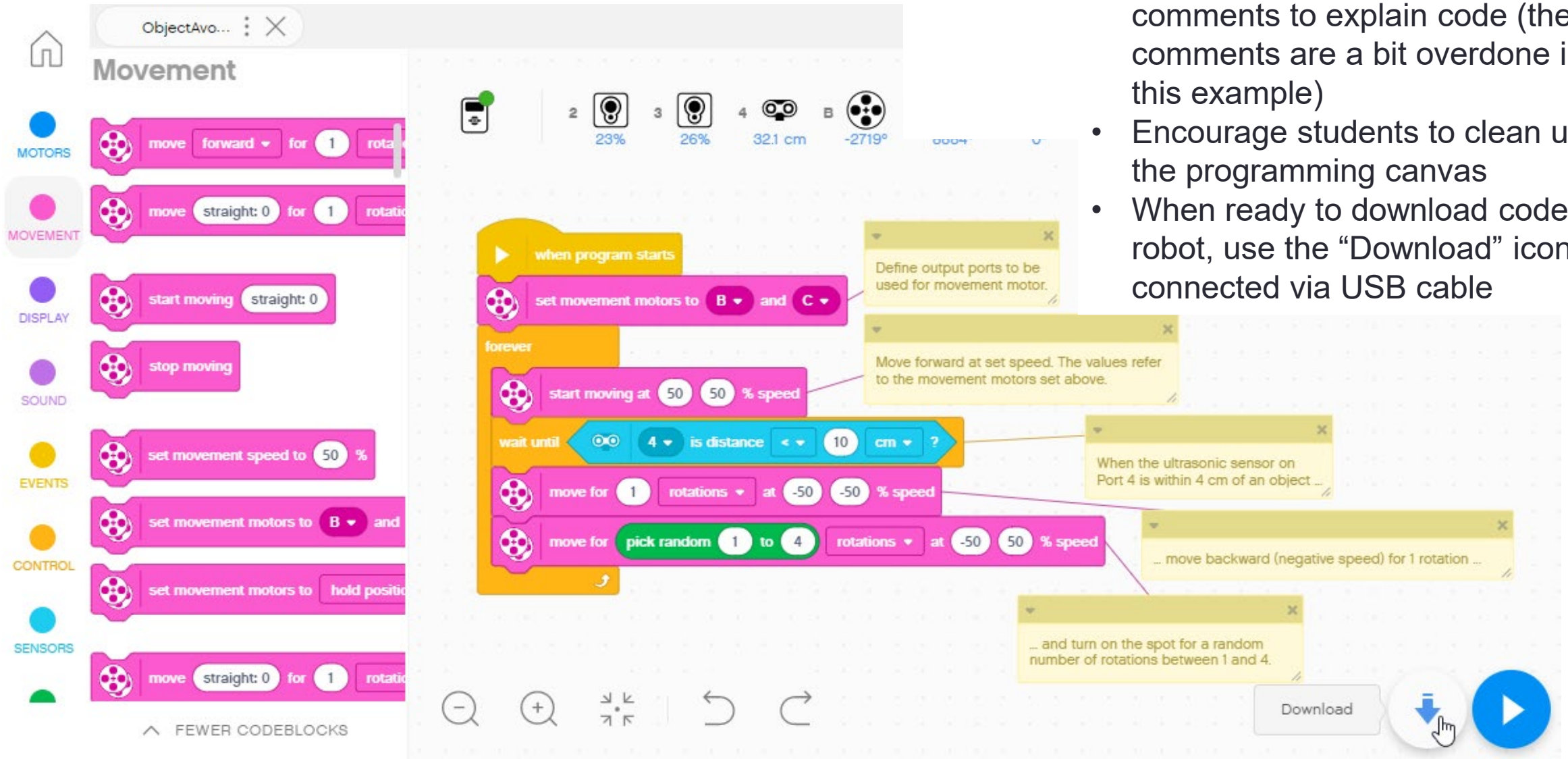
Operators – Mathematics and comparisons

Variables – data containers

Functions – collecting blocks of code



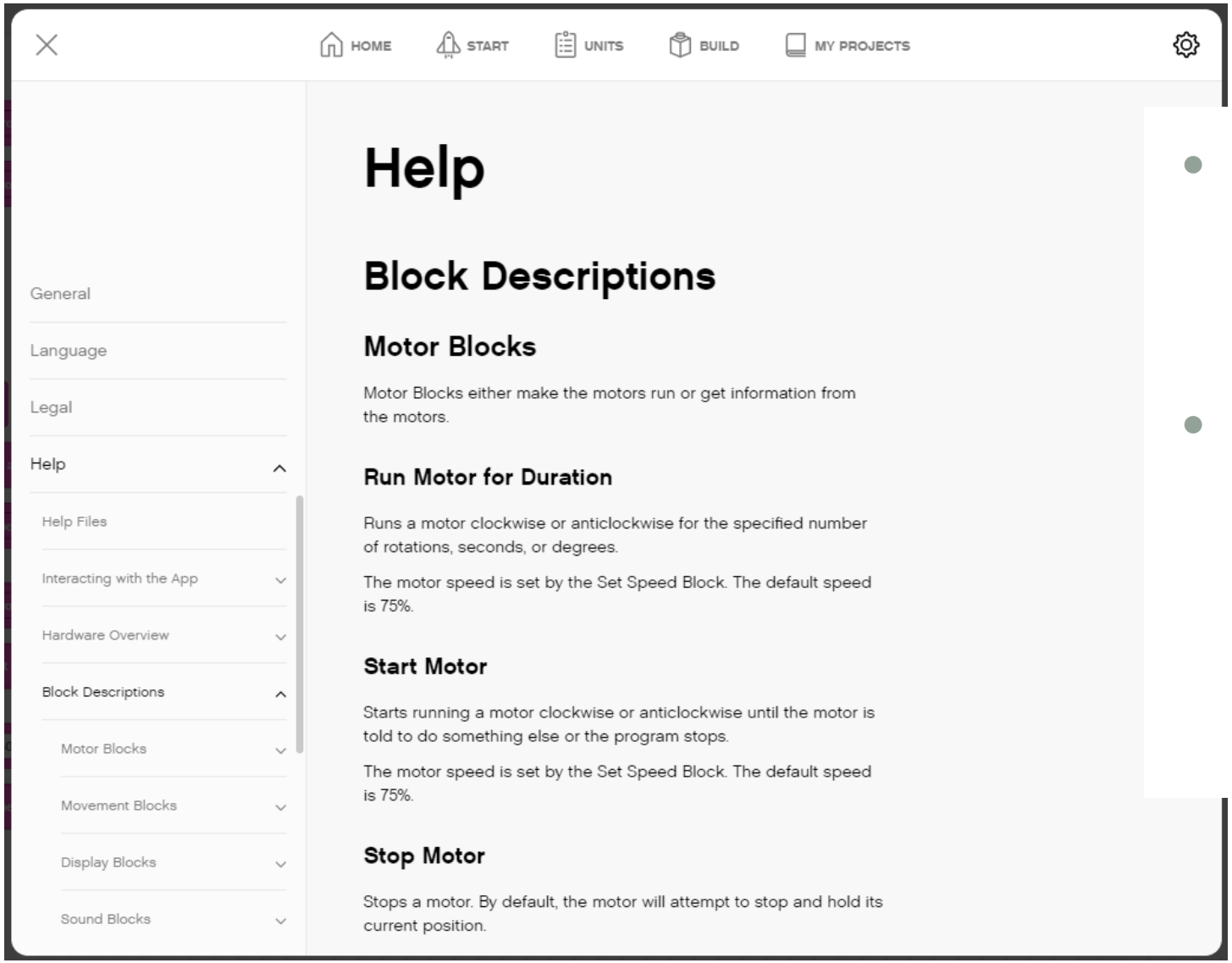
Example of simple program



Things to note:

- All blocks are ‘drag and drop’
- 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, use the “Download” icon if connected via USB cable

Getting help



- "Help" isn't as extensive as it was in the previous version of EV3 Mindstorms
- "Help" includes short descriptions of each coding block and how to use them

LINE FOLLOWING

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

Rescue Robot Programming

Where to start?

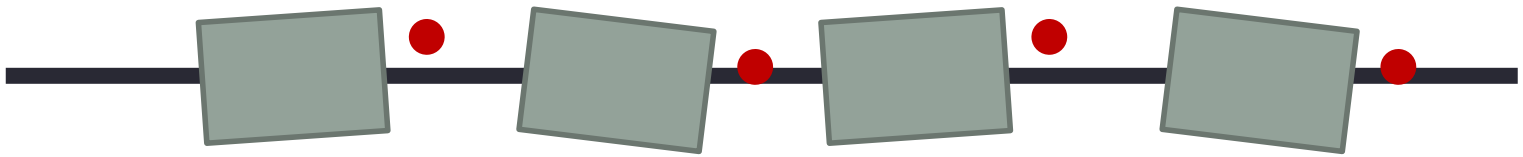
All Rescue divisions require the robot to:

- Follow a line
- Locate victim

One light sensor or two?

- Riley Rover Rescue 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
```

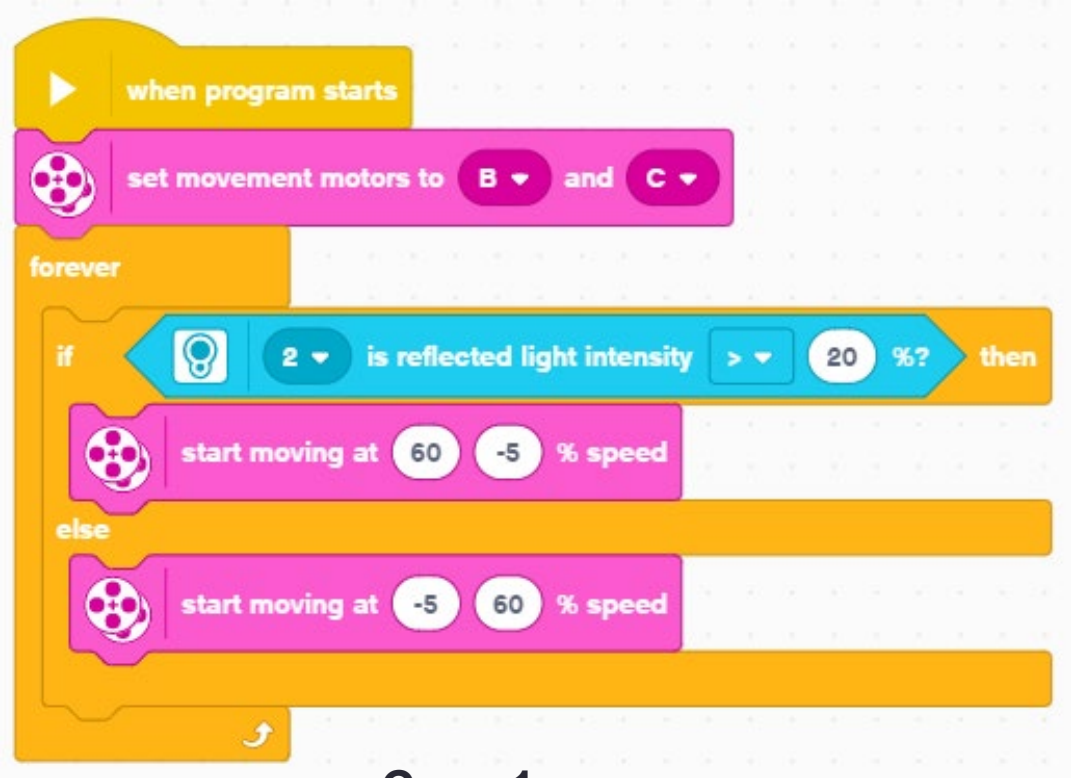
Case 2

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

What does this look like in code?

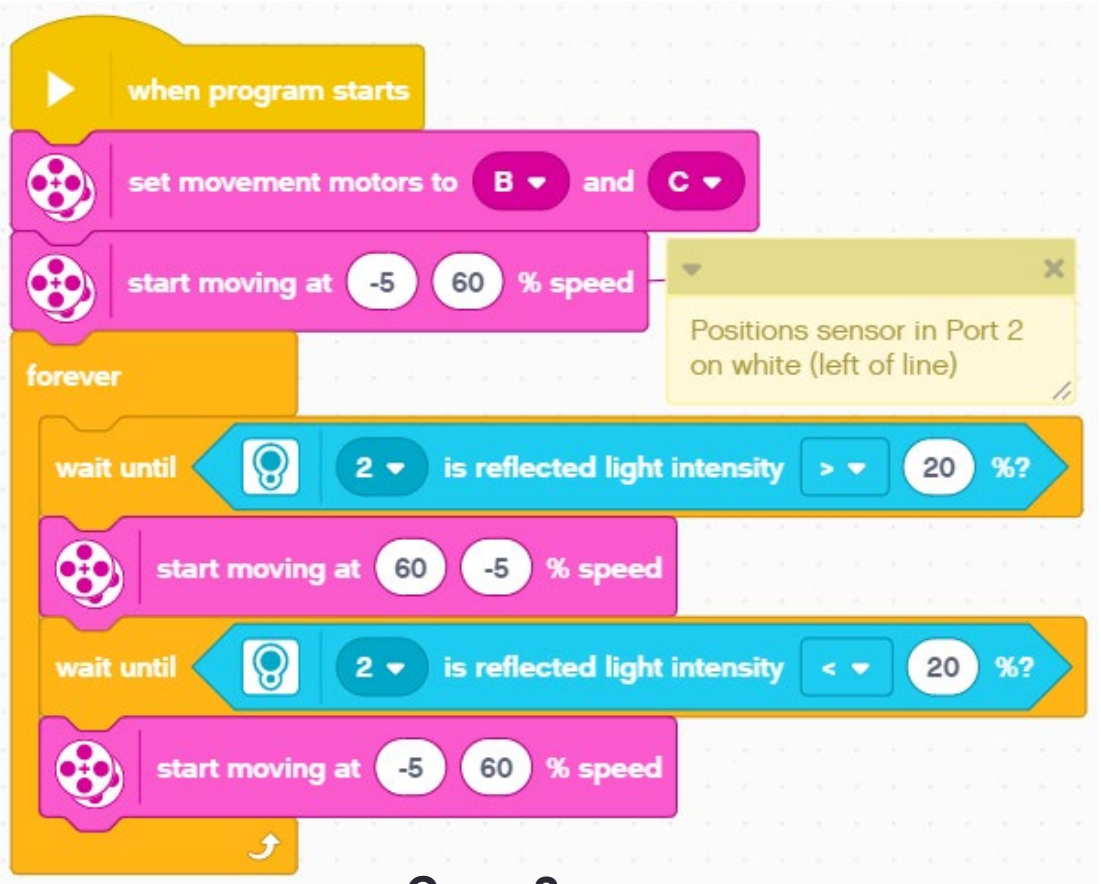
Please note: I have not included comments with all example codes. A good exercise for the students is for them to add comments. It forces them to think through the code and also helps them to see the important role that comments play.

Single Sensor



Case 1

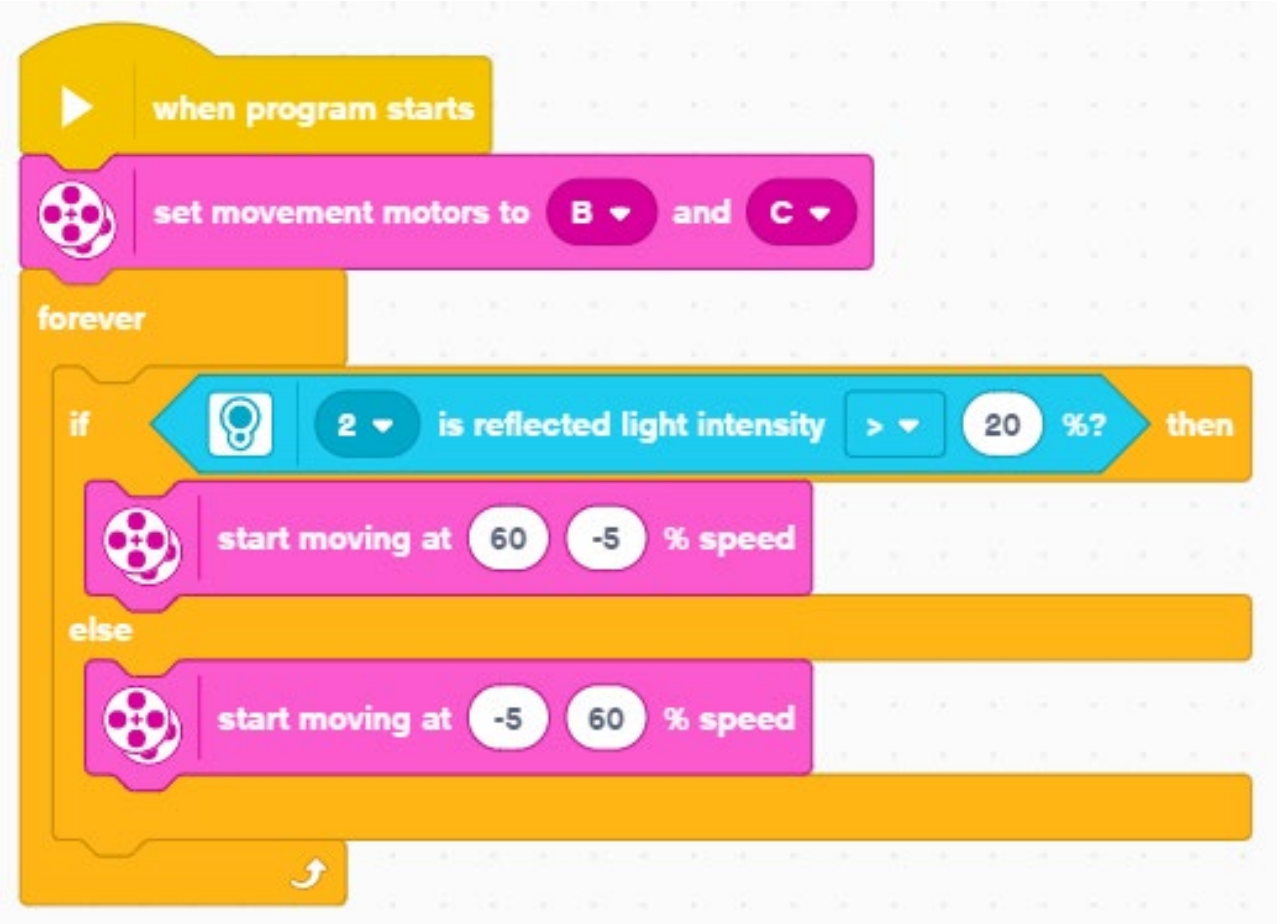
While True: # robot on
    If sensor sees white:
        Turn right
    Else:
        Turn left



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

- Sensor/motor ports in program don't correspond to ports used on robot
- Motors mounted in reverse orientation

➢ **NOTE:** This code may not work for your build and your light conditions

Ports B and C are the default motor ports. If they are used, the "set movement motors" block is not required but it is good practice so it is not forgotten if different ports are used.

➢ Students will use different ports!

Adding a second sensor

Case 1

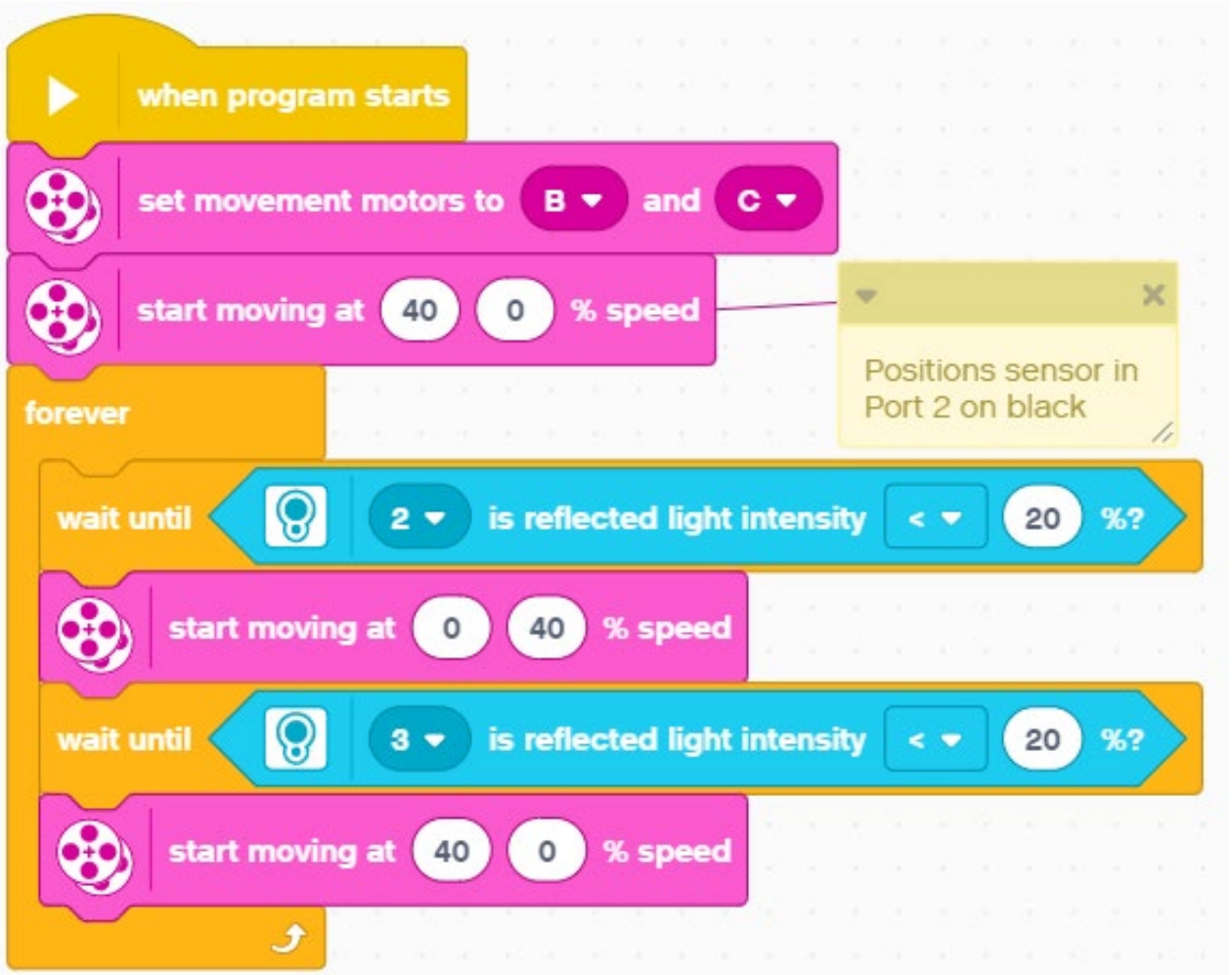
```
While True: # robot on
    While sensorLeft sees black:
        Turn left
    While sensorRight sees black:
        Turn right
```

What does this look like in code?

Case 2

```
While True: #robot on
    If sensorLeft sees white AND sensor Right sees white:
        go straight
    If sensorLeft sees white AND sensorRight sees black:
        turn right
    If sensorLeft sees black AND sensorRight sees white:
        turn left
    If sensorLeft sees black AND sensorRight sees black:
        # Will this ever occur??? What should happen???
```

Double sensor – Case 1



Case 1

While True: # robot on

    While sensorLeft sees black:

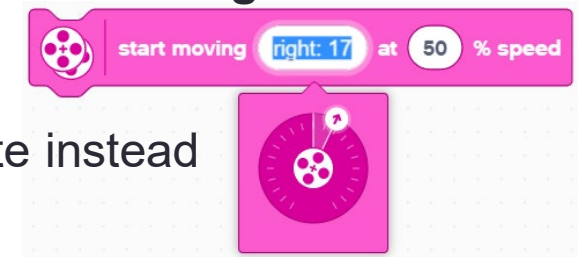
        Turn right

    While sensorRight sees black:

        Turn left

I have used the "**start moving at**" block

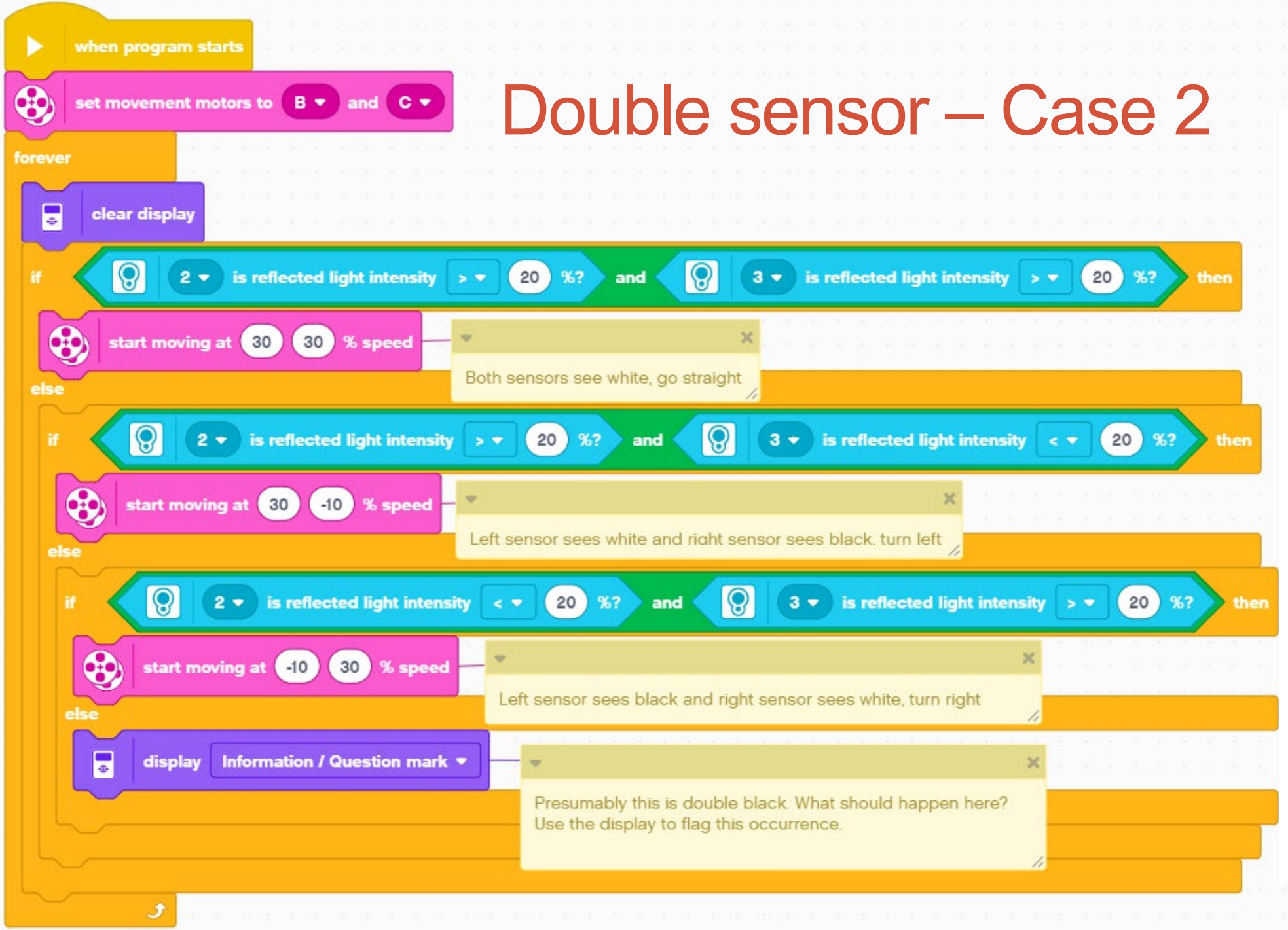
- Could I have used the **"start moving with steering"** block?



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

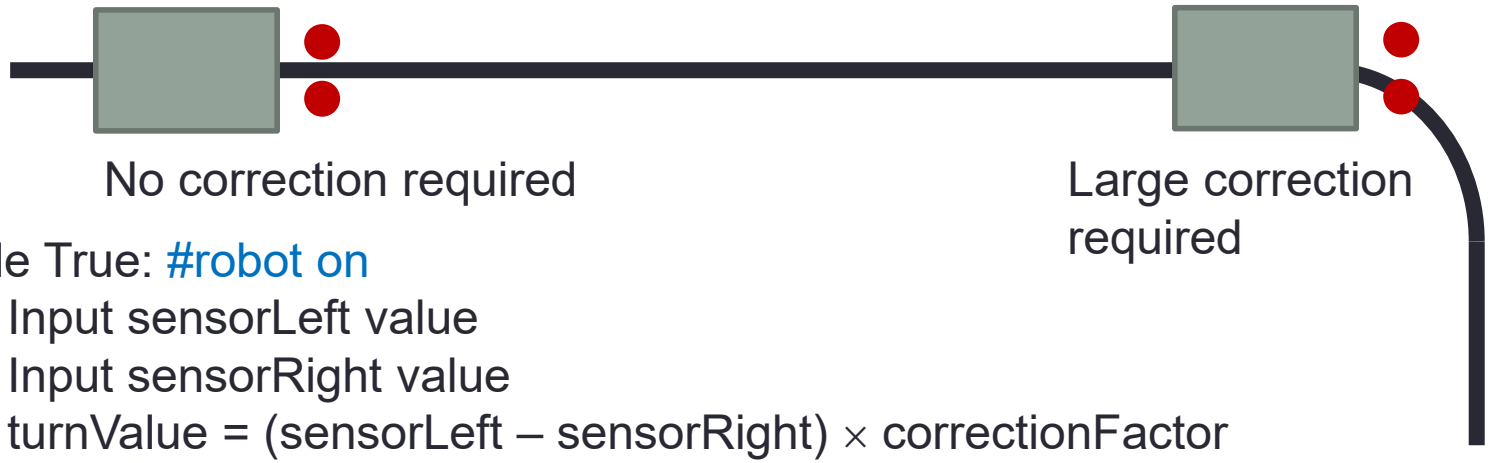


At the very bottom of the code is the “black-black” situation

- 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?

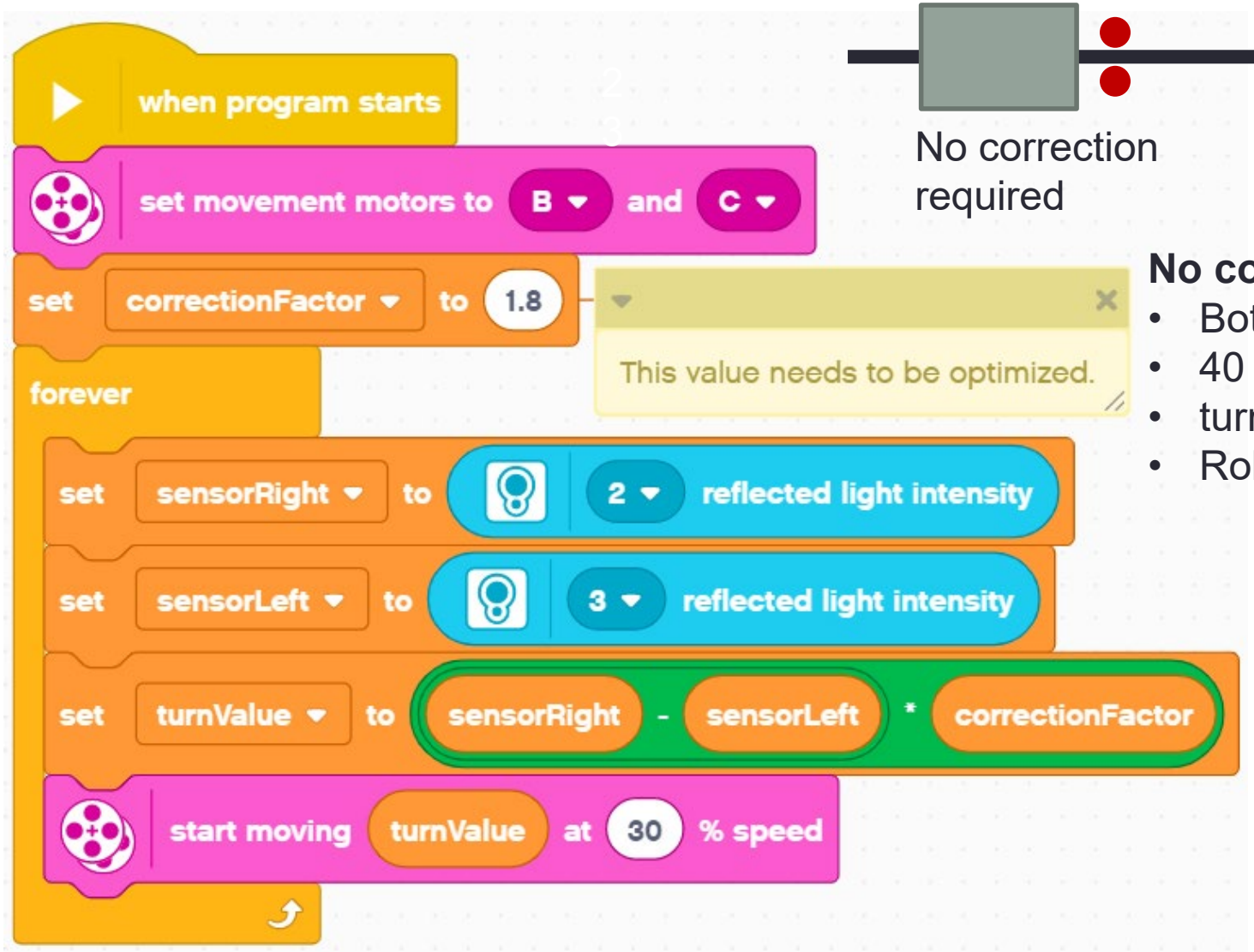
Proportional line follower

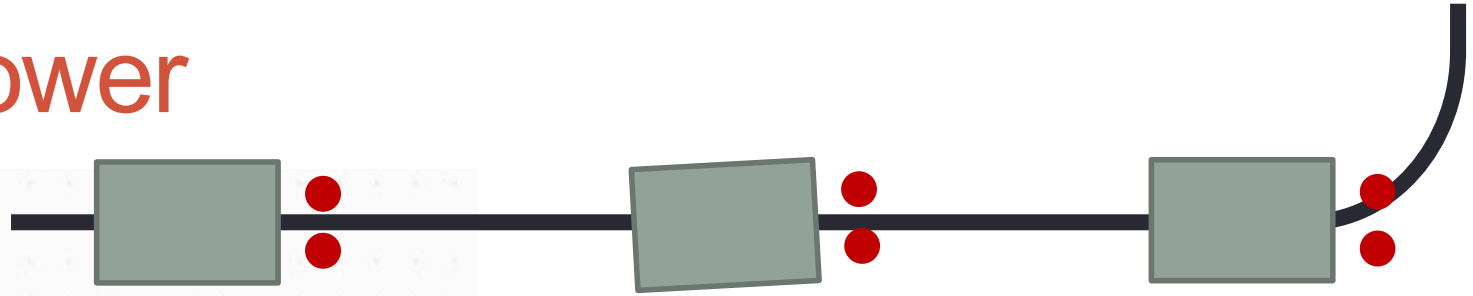
- What if I could adjust the amount the robot turns by how far off the line it is?
- Would this make for a much smoother run along the line?



While True: #robot on
    Input sensorLeft value
    Input sensorRight value
    turnValue = (sensorLeft – sensorRight) × correctionFactor
    Input turnValue to Move Steering block
    # if sensorLeft > sensorRight, turnValue > 0 and robot turns right
    # if sensorLeft < sensorRight, turnValue < 0 and robot turns left

Proportional line follower





No correction required

Small correction required

Larger correction required

No correction

- Both sensors on white
- 40 – 40 = 0
- turnValue = 0
- Robot goes straight

Small correction

- Right sensor on edge (35)
- Left sensor on white (40)
- 35 – 40 = -5
- turnValue = -5 × 1.8 = -9
- Robot turns slight right

Large correction

- Right sensor on white (40)
- Left sensor on black (20)
- 40 – 20 = 20
- turnValue = 20 × 1.8 = 36
- 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

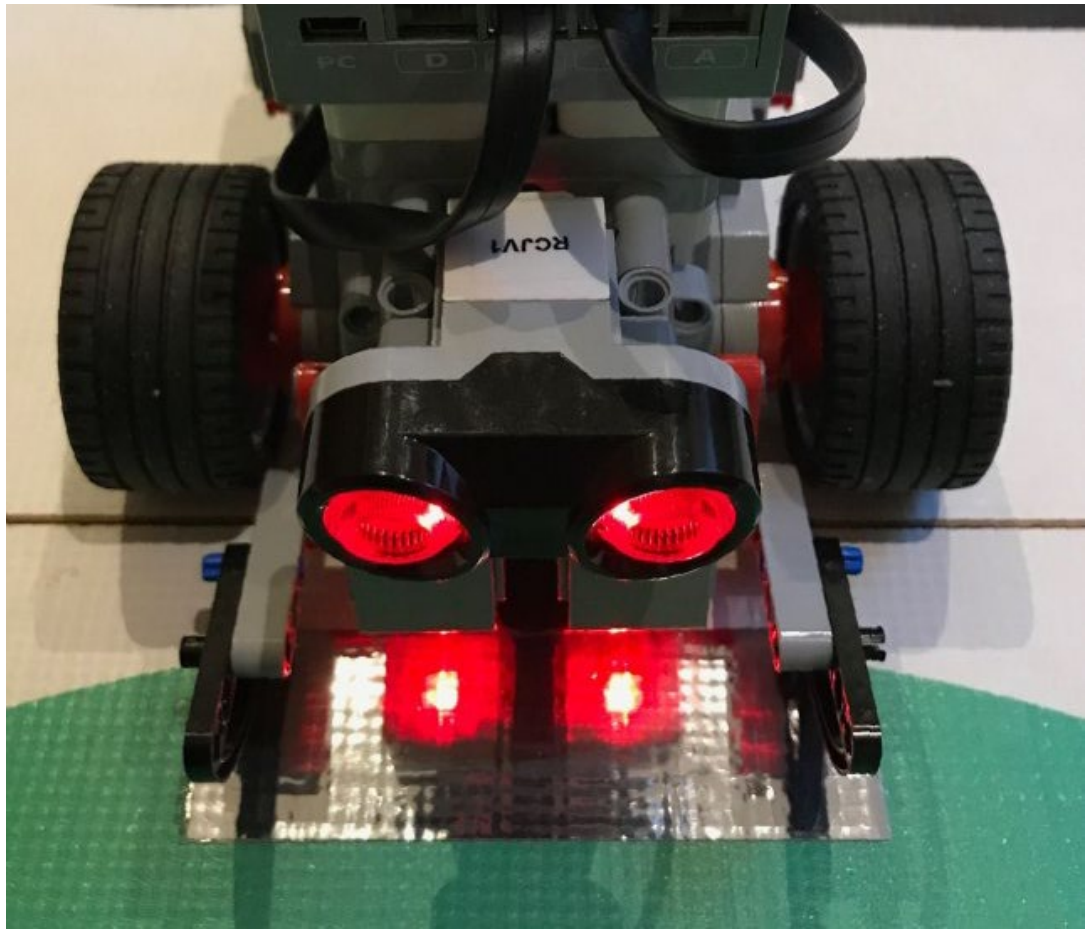
How can the robot control the victim?

- 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



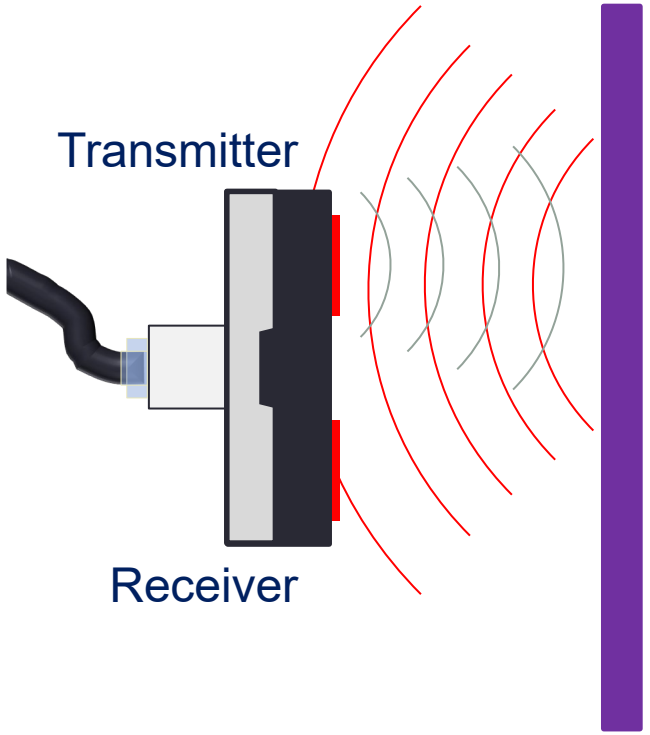
Detecting chemical spill

- 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?

Think through problem – Finding the victim

Finding victim

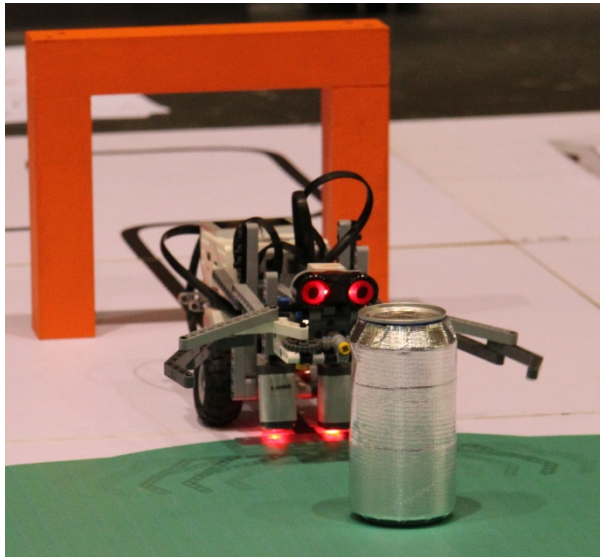
- Ultrasonic sensor measures distance from object
- Where is the best place to position the robot to begin checking?
- What is the maximum distance the victim could be from the robot?
- What happens if the sensor is too close to the victim?
- Is the ultrasonic sensor able to detect curved surfaces as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



Thinking through problem – Rescuing the victim

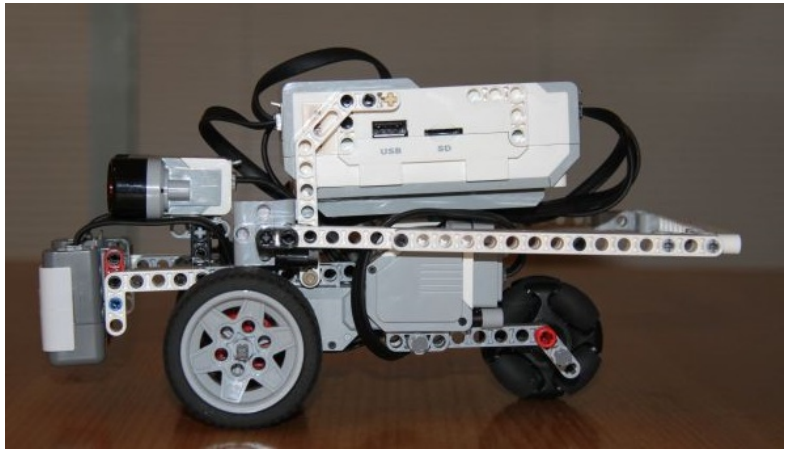
Controlling the victim (secondary)

- 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

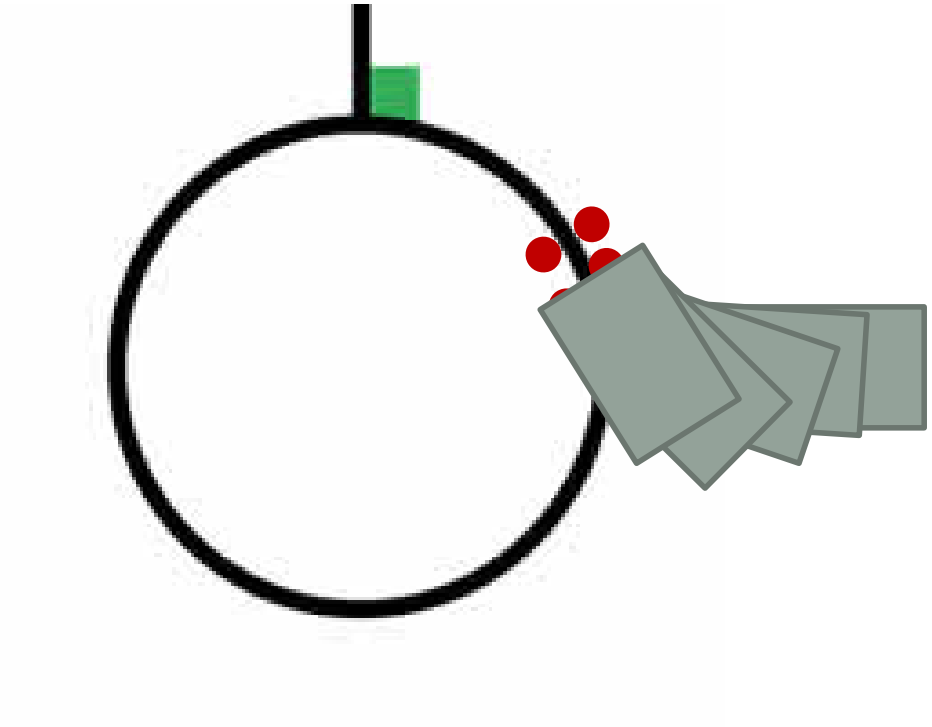
ADDITIONAL CHALLENGES

Detecting intersections

Navigating around water tower

Detecting green at intersections

Robots should turn in the direction of the green marker



Detecting green at intersections

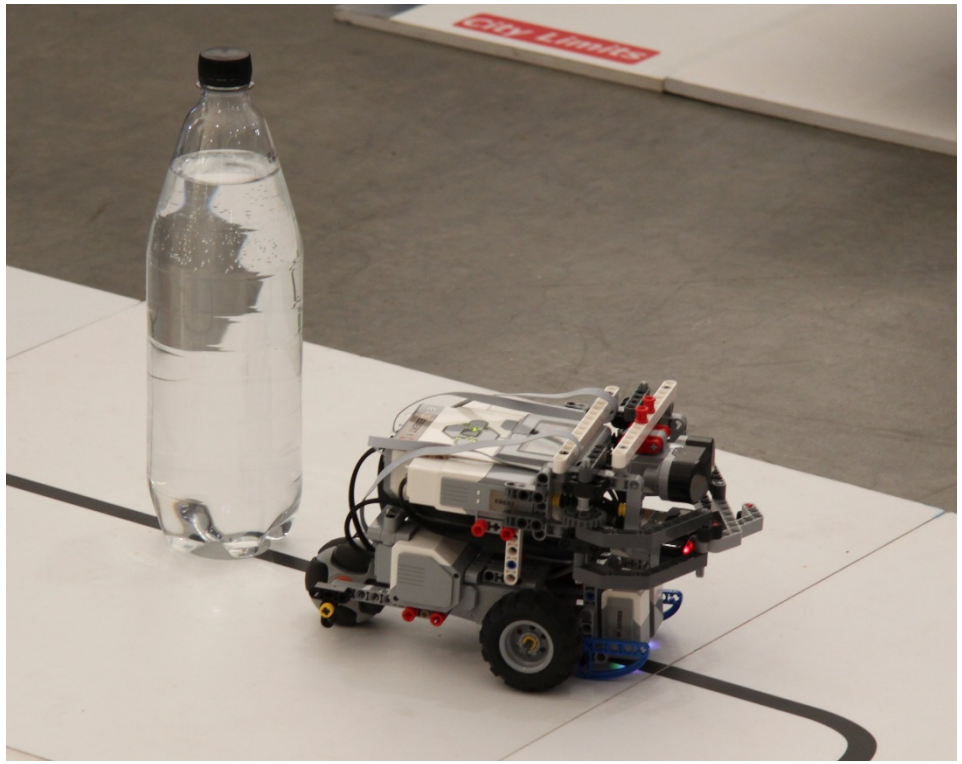
NOTE: The green on Rescue challenge mats used in the Victorian competitions are detected as green by Lego EV3 colour sensors. This is not necessarily the case for all Rescue mats and may not be true if sensors change. There is nothing in the national rules that specifies the shade of green.

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”?
- 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