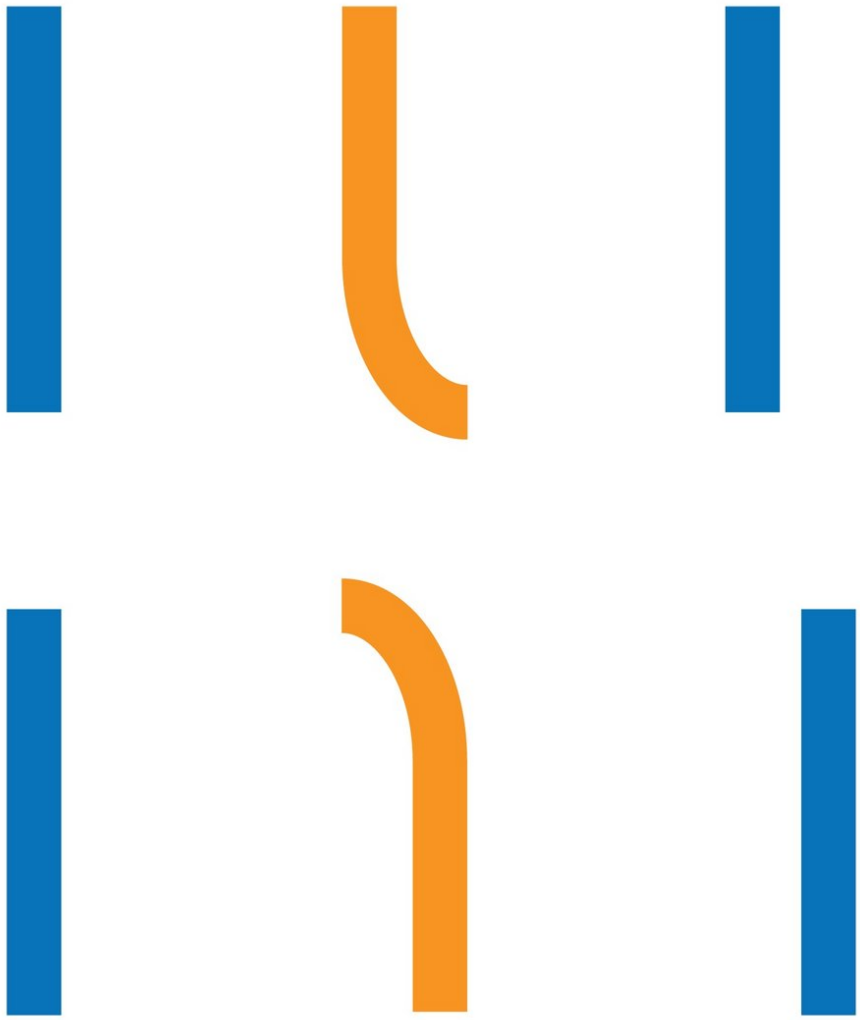
B - Broken Track

There's a gap in the track! We need to make our robot even more intelligent so it won't get stuck, and can find the track again on its own.



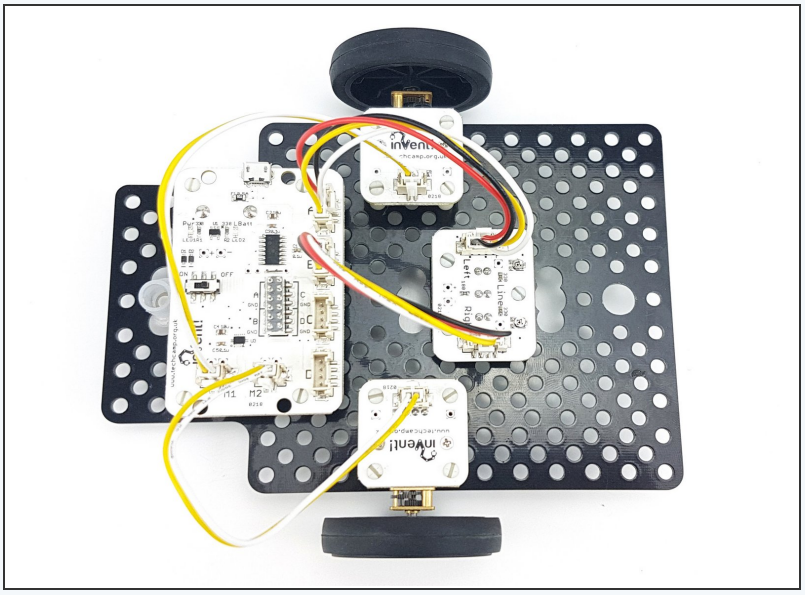
INTRODUCTION

There's a gap in the track! We need to make our robot even more intelligent so it won't get stuck, and can find the track again on its own.

Step 1

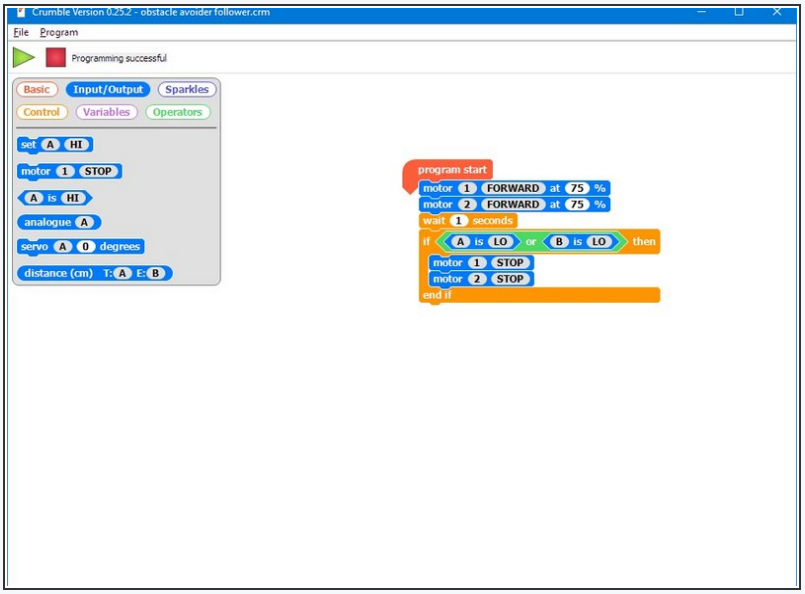
Assemble the Robot

- We only need the **line follower** module for this lesson - **assemble your robot** like the picture!



Step 2

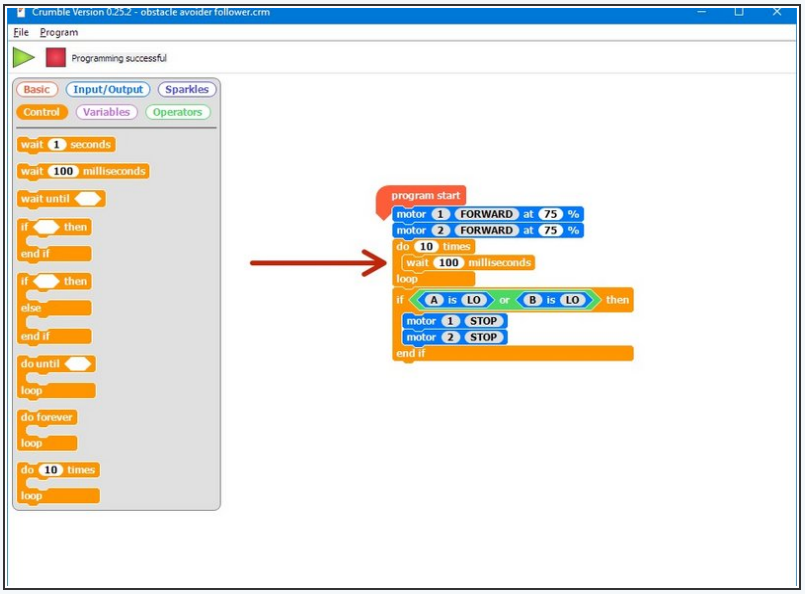
Waiting Causes Problems



- Dealing with breaks in the track is difficult. We can't just drive forward for 1 second using a **wait** block, as we won't be able to sense the line **at the same time.**
- **Build** the test program in the picture.
- It would be great if this program **drove forwards** and then **stopped** on the line - try it, see what happens.
- It only works if one of the sensors is **exactly on the line after 1 second** - this is not very likely!
- This is because the wait block **stops anything else from happening** whilst it is waiting for 1 second - so all the time we are driving forwards, we **can't check the sensors** - that's no good!
- Things that stop other things from happening are called **'blocking'** - they **block** everything else until they are finished.

Step 3

Wait differently

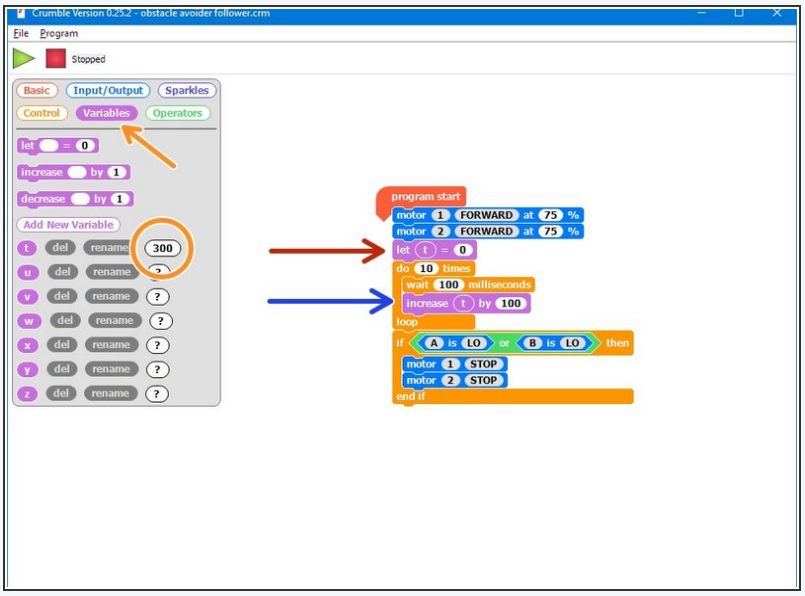


- We need to come up with a way to wait whilst **still being able to do other things.**
- Replace the **wait 1 second** block with a **do 10 times loop**, and a **100 millisecond** wait block inside the loop.
- This code will wait for **1 second**, just like before.

Step 4

Using a Counter

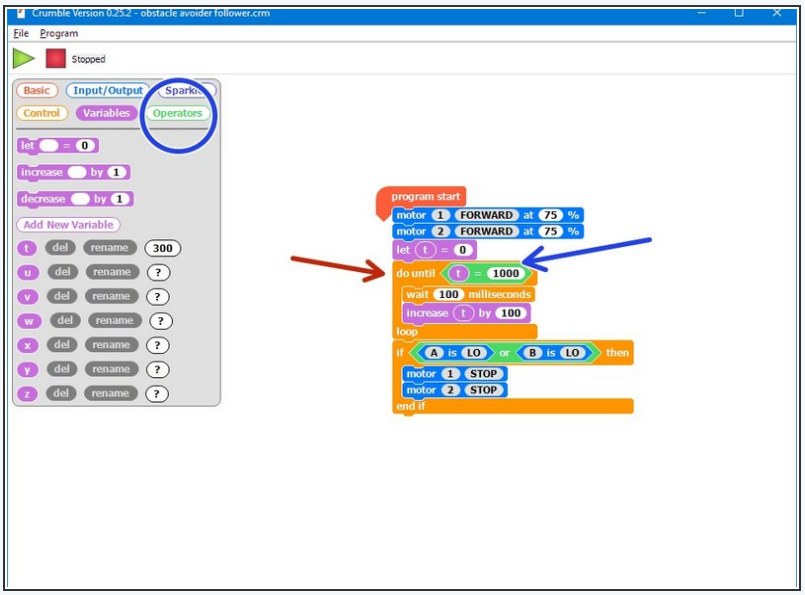
- We now need to add a **variable** that acts as a **counter** - it will count how many milliseconds of delay have happened.
- **Above** the do 10 times loop, add a block to **let t = 0.**
- Inside the loop, **increase t by 100.**
- **Run the program** and **watch the value of t in the variables menu** - it will now **count** the number of milliseconds of delay!



Step 5

Counter in the loop

- Now let's use the counter to control the **number of times** the loop runs.
- Instead of using a **do 10 times** loop, replace it with a **do until** loop.
- For the condition, use an **= block** from the **operators** menu so the loop runs **until t = 1000.**
- We can now **change** this number to decide how long the wait is! For example, changing to **t=2000** would run the loop for a total of **2000 milliseconds** (2 seconds).

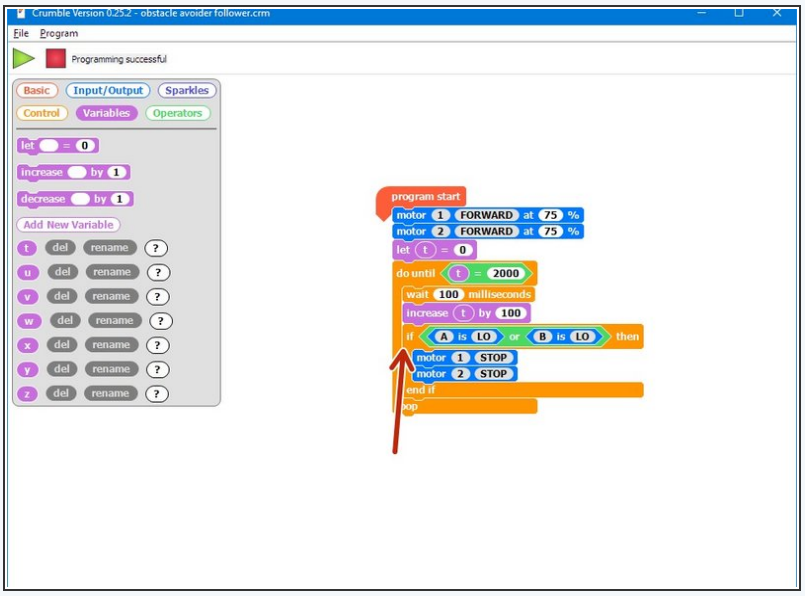


This document was generated on 2021-12-30 11:37:19 PM (MST).

Step 6

Sensors in the Loop

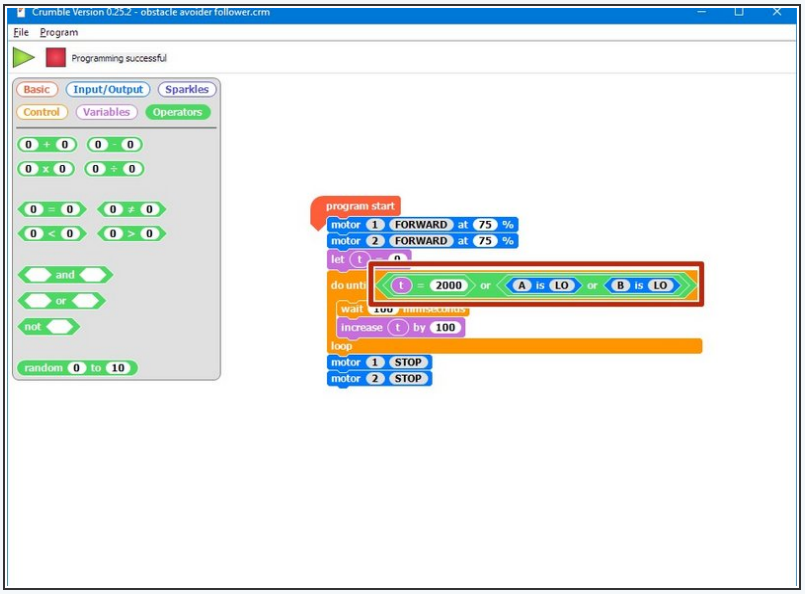
- **What's the point** of making a really **complicated wait block?**
- Anything we put in the loop will be run **as the wait is happening** - so we can check the sensors **whilst we are driving forwards!**
- Move the **IF block** checking the sensors **inside the loop**.
- **Try it out** - the robot should now drive forward and stop exactly on the line, **every time!**
- **Experiment** with changing the length **of the wait loop, so the robot can start further away** from the line and still reach it.



Step 7

Stop Waiting Sooner

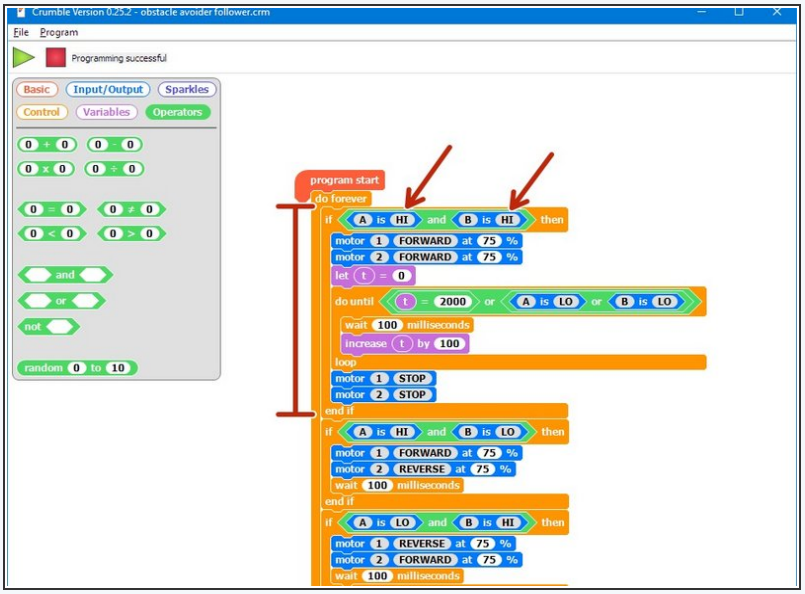
- We actually don't need the IF block in the loop - we can **merge** the conditions of the loop and the IF block together!
- Let's think about this - we want to **stop** the loop (and then stop the motors) if:
  - t=2000, **OR**
  - Sensor A is LO, **OR**
  - Sensor B is LO
- Luckily, we can use OR blocks **inside each other** to do this! Change your code to look like the picture, and **test it out.**



Step 8

Merge with Line Follower

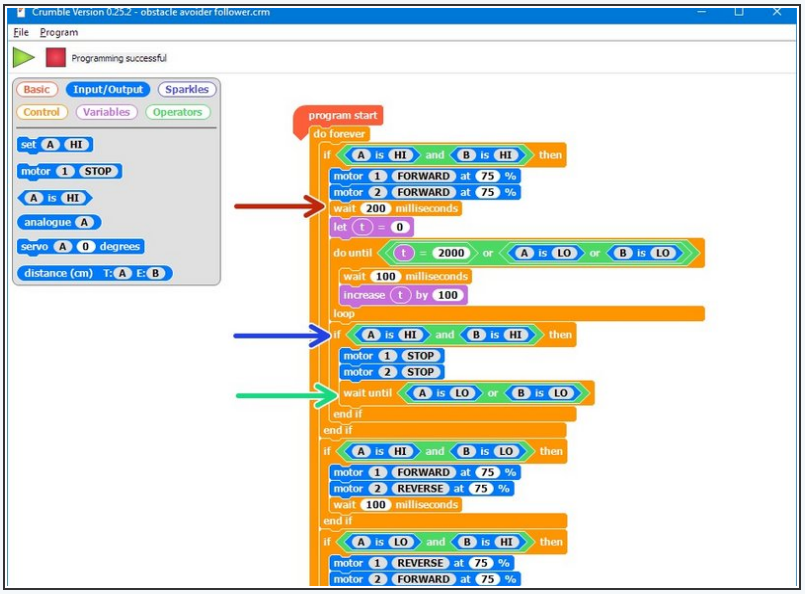
- Let's **merge our code** with the 2 sensor line follower program to deal with simple **breaks in the track.**
- **Load** up your code and add the line finder code you just wrote to the **IF** block where both sensors are **off the track (HI).**
- It should **look like the picture!**



Step 9

A Few Changes

- We need to make a few **changes** to make our code work with the line follower:
- Add a **wait block** of **200** milliseconds before the wait loop (this makes sure both sensors are not on the line)
- We only want to **stop** the motors if **both** sensors are **still off the track** after the wait loop.
- Put the motor stop blocks in an **IF block**, that checks if both sensors are **HI**
- After we have stopped the motors, we then want to **wait until 1 of the sensors is LO** before we continue
- Add a **wait until** block to do this!



Step 10

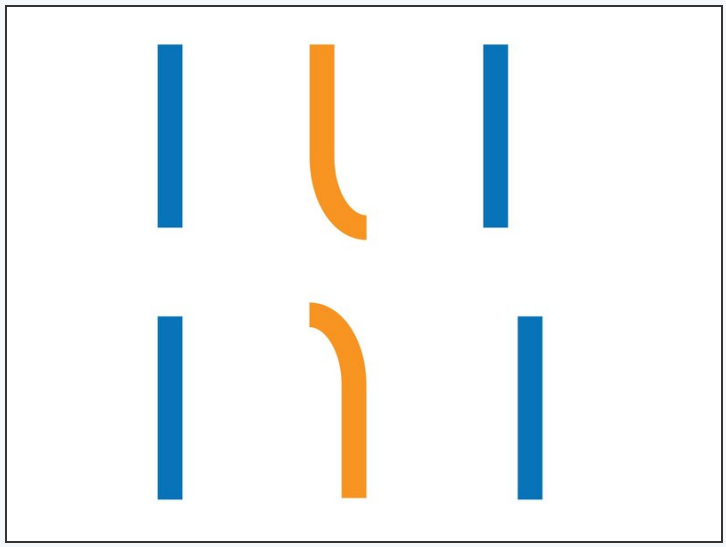
Line follower with breaks in track

- Cover a small section of **straight track** (about 5cm) with a **piece of paper** and tape it down to test the program.
- You will probably need to make **adjustments** to speeds and timings to make it work **reliably!**
- **Keep experimenting** until it works well.

⚠ Make sure the gap is on a straight section of track - this code won't work on gaps in curves! **Can you work out why?**

Step 11

Curved Breaks



- Once you can cross a gap in straight track, try a gap in **curved track!**
- To do this, you will need to make the robot **move side to side** in the wait loop, instead of just moving forwards.
- This can be done by making the robot **turn** to start with instead of going forwards, and then **changing the direction of turn** inside the loop every so often.
- It works best if the robot goes left and right **several times** , in a kind of sweeping motion.
- You can also experiment with **other types of break** like in the picture - offset lines and breaks that point the robot in the **wrong direction** like the middle example are particularly difficult to get right!