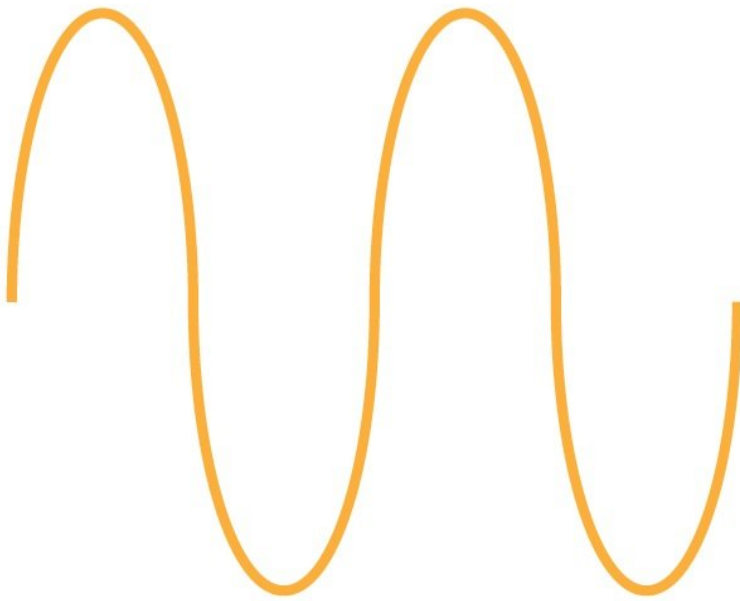


C - Smoother Line Following

Learn about analogue inputs to make an even more sophisticated line following robot, that will smoothly follow any path.



010110

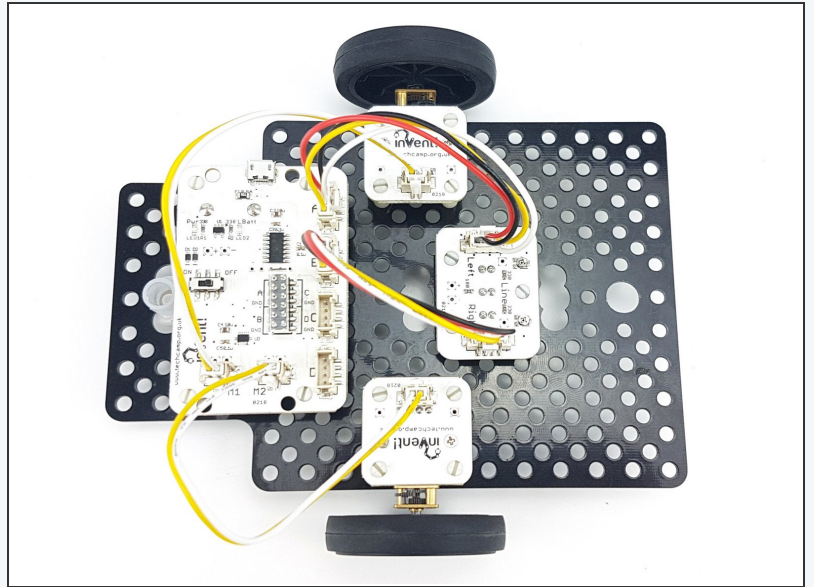
INTRODUCTION

Learn about analogue inputs to make an even more sophisticated line following robot, that will smoothly follow any path.

Step 1

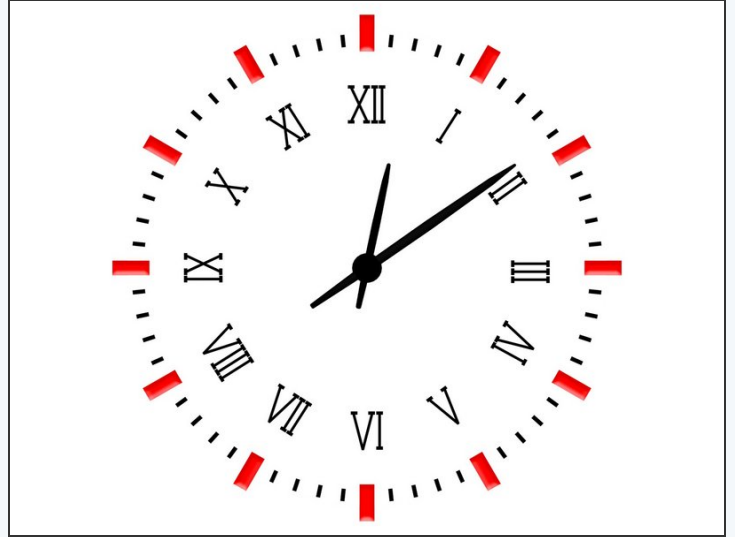
Setup Your Robot

- We just need the **line sensor** for now - make sure your robot is setup like the picture.



Step 2

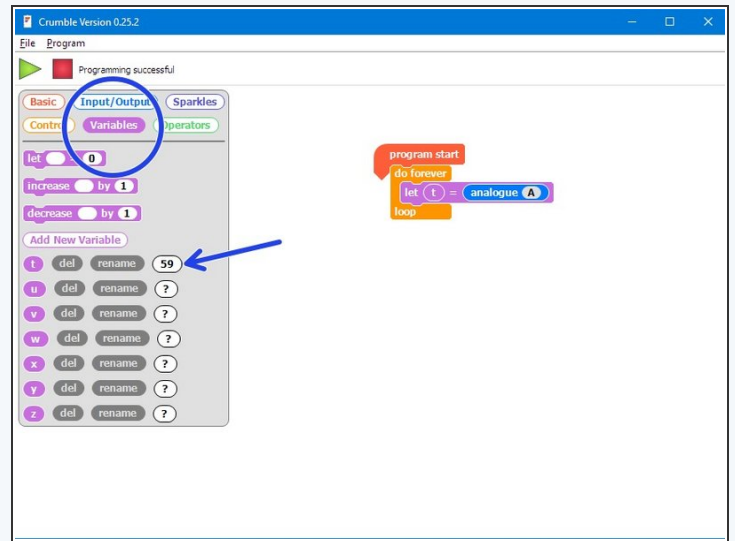
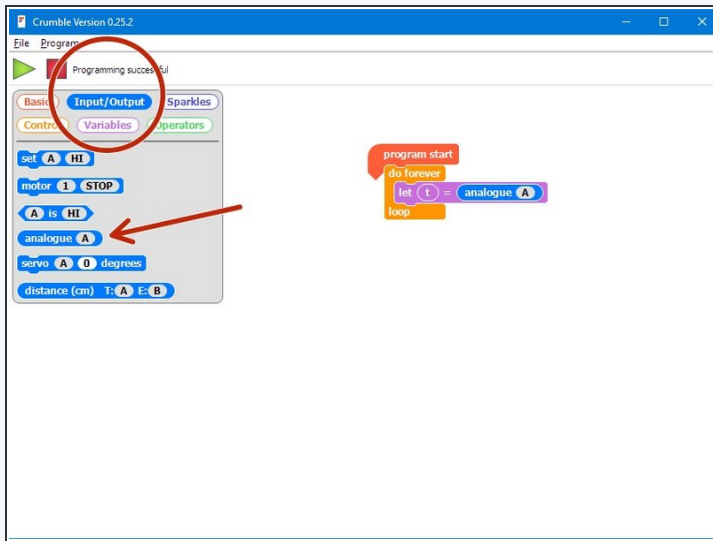
Analogue and Digital



- To make a **smoother**, better line follower, we need to use the line sensor in **analogue mode**.
- So far, we have been using it as a **digital** sensor - it can only be **ON or OFF (HI or LO)**.
- Analogue inputs (and outputs) can have **any value** - think about the difference between a digital and an analogue clock
- A digital clock must display a **whole number** of minutes
- But on an analogue clock, the minute hand can be **anywhere** - even halfway between two minutes!

Step 3

Analogue Line Sensor

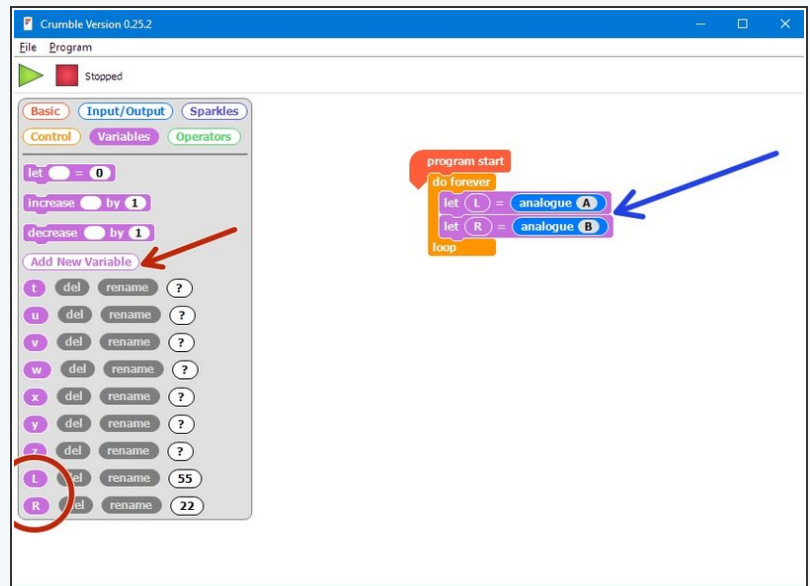


- **Build** the simple test program in the picture.
- You can find the **analogue** block in the **Input/Output** menu
- Program your robot, and **keep it plugged in**.
- Try moving the robot **slowly** from one side of the line to the other, whilst **watching the value of t** in the **variables menu**.
- See how it changes **gradually** as you approach the line?

Step 4

2 Analogue Sensors

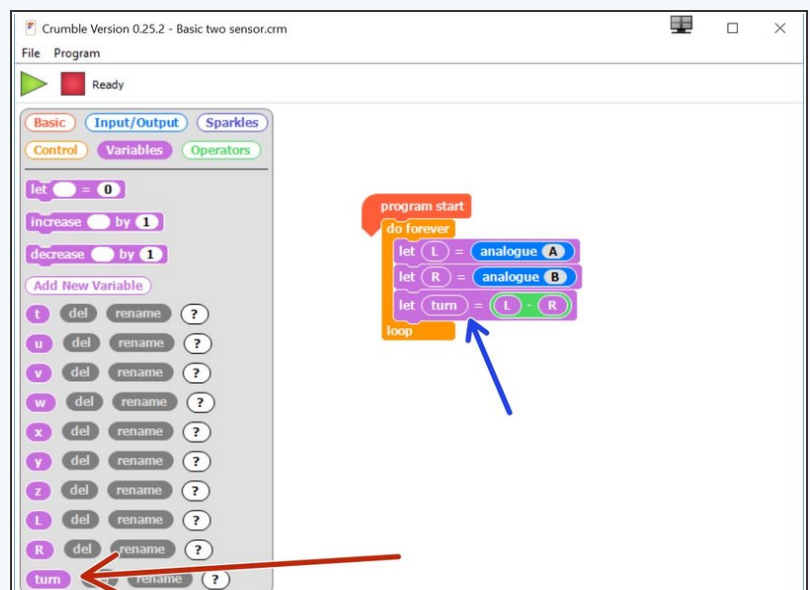
- We can use this **gradual** change to **smoothly** change the amount the robot turns as it gets **further from the line!**
- Add two new variables to the program called **L** and **R** (left and right).
- Let **L** = the analogue value of the **left sensor**, and **R** = the analogue value of the **right sensor**.
- **Program** the robot and try **moving it slowly** across the track again whilst still plugged in - make sure L and R behave **how you expect** them to!



Step 5

How much to turn?

- The larger the **difference** between L and R, the further the robot is from the line so the **more** we need to turn.
- For example, if both sensors are on the line, we **don't need to turn at all** and L and R will have the **same value**.
- Add a new variable called **turn**.
- **After** getting the values of L and R, set turn equal to the **difference** between **L and R**.



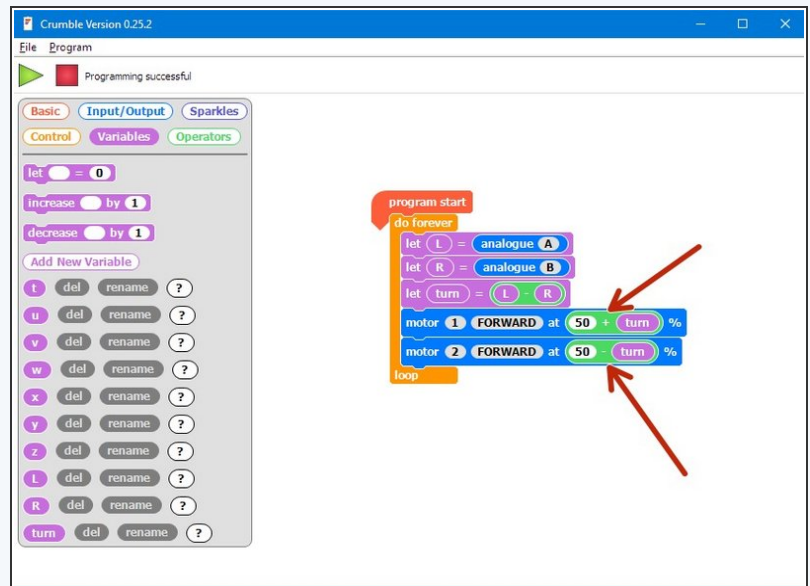
Step 6

Using the Turn

- **Program** your robot and watch what happens to the value of **turn** as you move it **across the line**. It should be **0** when the robot is exactly on the line!
- Let's use the turn variable to set the **speeds** of the motors.
- **Change** the motor blocks so they use the **turn variable** to set the speed like the picture.
- Test it out - this should follow the line **really smoothly**!
- Do you **understand** how the code works? (hint: turn is **positive** when we need to turn **right**, and **negative** when we need to turn **left**)



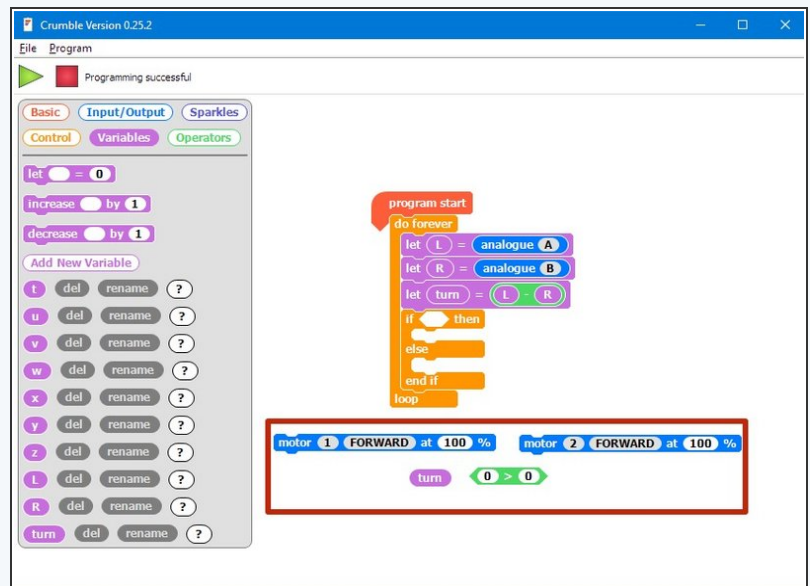
If your robot can't do the **really tight turns**, try multiplying turn by 2!



Step 7

Maximum Speed

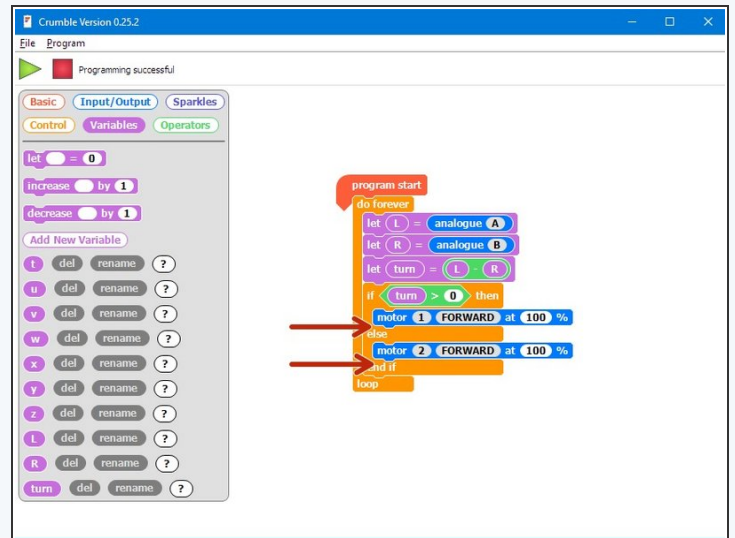
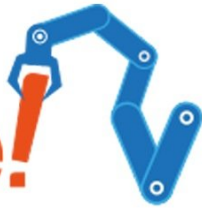
- You might have noticed that while the new program is **smooth**, it isn't as **fast** as the old two sensor **digital** program.
- To make it faster, we need to make sure 1 wheel is always going **100% forwards**, and then change the speed of the **other wheel only** based on how large **turn** is to follow the line.
- So, if turn is **positive**, we should be turning **right**, so motor **1** should be at 100%
- If turn is **negative**, we should be turning **left** so motor **2** should be at 100%
- Add an **IF/ELSE block** to check if **turn** is **positive** or not, and set the correct motor to **100%**.



Step 8

Proportional line follower

Challenge!



- Now its **over to you!**
- Add **two more motor blocks** into the IF/ELSE block to set the **other** motor's speed using the **turn variable**, for turning left and right
- **Test** the program really well - **experiment** with multiplying **turn** by different amounts to get a **reliable** line following program.
- ❗ Here's a **hint** if you're confused - for the first part where **motor 1 is at 100%**, motor 2 should be set to **forwards** at a speed of **100 - turn**.

Step 9

Proportional Sparkles

- If you're feeling really advanced, add the **Sparkle module** back in and set the colours of the LEDs **proportionally** based on how far away from the line the robot is!
- Your robot can also **get lost** and now has no way of finding the line again - try and **add the code you wrote previously** back in so the robot can't get lost, or at least **stops** if it loses the line completely.

