

Ri Masterclasses at Warwick

Robotics – Moving Intelligently

Teacher Pack

Outline	1
Follow up	1
Promotion and Social Media	2
Challenges	2
Timings	4
Cheat Sheet	6

Outline

This series of 5 × 50 minute sessions will focus on building and calibrating a cardboard computer-controlled robot car. It can function entirely without computers, if needed, though a good deal of extension work is available if the students have access to computers.

For a ‘tech-free’ approach students will interact with code that has already been installed onto the micro:bit that they are provided with. This code has a set of ‘modes’ the students can cycle through. These modes are:

- Drive forwards
- Drive backwards
- Turn in a circle (left)
- Turn in a circle (right)
- Follow a line

However, no two motors are ever quite the same and no construction is ever perfect so there will be issues with these codes ‘out of the box’. The challenges for the students are to try to calibrate the robot in those modes to achieve peak performance.

Follow up

Our mission is to provide the best enrichment to schools that we can. If there is anything that you think we could do better, please tell us. You can get in touch at wmgoutreach@warwick.ac.uk and I promise we take even negative feedback well. Your comments could make the difference so please do let us know how we’ve done.

Promotion and Social Media

We would also love to see any photos or comments you or your students have about the events on social media! You can find WMG @wmgwarwick, the High Value Manufacturing Catapult @hvm_catapult and the Royal Institution @Ri_science on Twitter and other major social media platforms. We're all active in promoting the good work that schools do to enrich the curriculum so if you are able to, please do share on social media what you think of this activity and what the students have created.

Challenges

The vehicle kits will have some issues – they are being built under time pressure out of cardboard with minimal fixings holding motors in place and so on – but this is a valuable lesson to the students that an initial prototype always has bugs and issues. The skill of a good engineer is to iron out the creases and resolve these bugs to get a functional robot. The 'Make, test, evaluate, improve' cycle is key to this series of tasks.

The challenges are:

- Drive in a straight line for 10 seconds (forwards and backwards)
A programmatic 'bias' variable can be set by the students without having to edit any code. They will therefore be able to adjust the motors on either side of the vehicle to get it to drive in a straight line.
- Turn in a circle (left and right)
The vehicle drives with one motor turning faster than the other for a length of time. The students can then measure how far around a circle their vehicle goes using a protractor. They can then calculate the ideal amount of time this function should run for to turn the vehicle exactly 360 degrees.
- Follow a line
Vehicles often struggle with line following based on how fast they are moving and trying to over-correct for inputs from the sensors. Getting the speed the vehicle moves at while following a line will balance speed and performance.

These challenges are all about testing out variables and trying to find the optimum balance. This is a major theme of all STEM practices. In order to reflect and internalise what they have learned the final session will be used to prepare a poster that is templated as a 'Calibration Sheet'. Again, a calibration sheet is standard in industrial STEM positions. The students will have the opportunity to record the best settings they found for their vehicle, reflect on how well the vehicle performed, suggest improvements for a future version of this kit, and suggest additional kit that could be added to make the vehicle more useful.

Throughout the booklet there are references to real-world scenarios to contextualise the content the students are exploring. The full code and digital

Dr Phil Jemmett
WMG Outreach Team

Foundation Fellow, Warwick Institute of Engagement

WMG
International Manufacturing Centre
University of Warwick
Coventry CV4 7AL

versions of the booklet are available here:

https://warwick.ac.uk/wvgoutreach/resources/microbit/ri_masterclass2021

Timings

5 minutes have been left spare from each 50 minute session to allow any administrative tasks you may need to complete in the session e.g. registers.

Session 1

Introduction (10 minutes)

Phil will talk to the students and provide some context for why we might want to make vehicles more intelligent and how WMG are working to support this. He will discuss how engineers are more than just the people who fix cars but also the people who design, test and improve all kinds of products. This can then be linked to how students might progress through to a career in STEM.

Main task (30 minutes)

Build and test the vehicle following the instructions provided.

Wrap-up (5 minutes)

Students will be asked to share, peer-to-peer, which parts of the main task were challenging or fiddly, how they might be improved, and what they think went well.

Session 2

Introduction (5 minutes)

The students by this point should have an assembled robot that is able to use the code that was pre-programmed onto the microbit. Phil will give a brief overview of how this code has been made and how it is designed to work, before introducing the first challenge the students will be working on.

Main task (35 minutes)

Challenge #1: Test how the vehicle drives in a straight line forwards and backwards and find the perfect bias setting to get the vehicle to drive as straight as possible. (35 minutes)

Wrap-up (5 minutes)

Again, peer-to-peer sharing will be encouraged. If the students are quiet Phil will cover the STEM skills the students are learning and bring in some more context for intelligent vehicles.

Session 3

Introduction (10 minutes)

Since this will be on a separate day, Phil will briefly recap where we got to in the previous sessions and cover what STEM skills the students will already have picked up. The second set of challenges and the scope of what can be achieved in the remaining 3 sessions will be explored.

Main task (30 minutes)

Challenge #2: Test how the vehicle drives in circles and calculate the calibrated pause time required to complete a perfect circle.

Wrap up (5 minutes)

Peer-to-peer sharing

Session 4

Introduction (5 minutes)

Main task (35 minutes)

Challenge #3: Test how the vehicle performs following a line on a track. Choose the perfect speed to balance performance and the time taken to navigate the town map.

Wrap up (5 minutes)

Peer-to-peer sharing

Session 5

Introduction (5 minutes)

As a final introduction, this short section will cover the idea of a calibration sheet and how they are used in industry throughout STEM.

Main task (30 minutes)

Calibration sheet: Fill in a templated poster calibration sheet with the settings they have found through the challenges and suggest improvements for a future version of this robot.

Wrap up (10 minutes)

Phil will cover what the students have achieved by calibrating this vehicle and how this exact process is done in almost every STEM field. The scope of what else could be done with these vehicles if students continue to program them will also be covered. Finally, Phil will give a small outlook on the future of transport and suggest what roles and jobs students might be able to take if they went into a STEM field.

Cheat Sheet

There are a few questions in the booklet for students to work out. This section of the teacher pack aims to help you support students who are stuck on those sections.

Making the Perfect Circle (p.33-34)

The students are aiming to work out how long the 'pause' command should be in the code to enable their vehicle to drive a perfect 360 degree circle. The current code drives the motors for 12 seconds (12000 ms in the code). They should measure the circle their vehicle drives in to get an angle that it is currently completing – for example 330°.

The first part of the question asks the students to work out the percentage of the circle they are actually getting around. For our example of 330° this is:

$$\% \text{ of circle} = 100 \times \frac{\text{current angle of circle completed}}{\text{degrees in a full circle}} = 100 \times \frac{330}{360} = 91.\dot{6} = 92\%$$

The second part of the question asks the students to work out the length of pause that would be needed to get to 360°. For our example, this is:

$$\text{Pause needed} = \frac{\text{current pause length}}{\text{current percentage of circle completed}} = \frac{12000}{92\%} = \frac{12000}{0.92} = 13043 \text{ ms}$$

On the Right Track (p.38-46)

Line following sensors are notorious for being temperamental. If the lighting in the room is really bright or changes a lot (a sunny day with clouds) there can be challenges. The lighting can interfere with the sensors.

The code has been designed to try and calibrate the sensors each time the vehicle is switched on, in an attempt to account for any changes in light. When the switch on the side of the vehicle is switched on the sensor takes a reading. The vehicle should be placed on white paper with the sensor bar in place at this moment. Then, move the vehicle onto the black line the students have drawn or on the town map. Switch the vehicle into line following mode by pressing B until the line symbol appears. When the students press A a second value will be read from the sensor – this time over a black line. Now the sensor has a value for white paper and for black lines and should hopefully be able to tell the difference between the two.

Extension question: Going Wheely Far

The first extension question asks the students to work out how far their vehicle travels in the 10 seconds that the motors are switched on in either Drive Forwards or Drive Backwards mode. To do this, mark the position of the front of the vehicle before pressing A and then mark the front of the vehicle when it stops after 10 seconds. A speed unit of cm/s is probably most sensible here. My vehicle travelled 105 cm in 10 seconds. It's 'real world' speed is therefore:

$$\text{Real world speed} = \frac{\text{distance travelled (cm)}}{\text{time spent travelling (s)}} = \frac{105}{10} = 10.5 \text{ cm/s}$$

If I wanted my vehicle to travel exactly 1 metre before stopping, I would need to use this real world speed to calculate how long the 'pause' command should be:

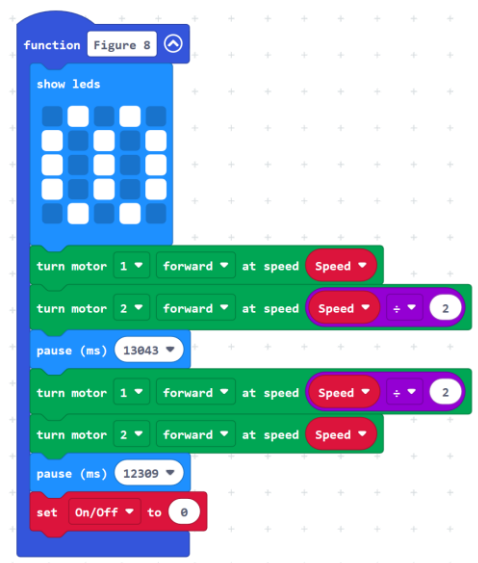
$$\text{Time to travel 1 metre} = \frac{\text{target distance (cm)}}{\text{real world speed (cm/s)}} = \frac{100}{10.5} = 9.524 \text{ s}$$

But since the code runs in milliseconds we would need to put 9524 ms into the pause command.

Extension question: Round and Round

This extension question asks the students to think about what a Figure of 8 function might look like. It will need the students to have worked out how long the pause command should be to get a full 360° circle to the left and to the right.

The code should perform one circle and then the other, so will effectively look like the two separate functions one on top of the other. Here's an example:



```

function Figure 8
  show leds
  turn motor 1 forward at speed Speed
  turn motor 2 forward at speed Speed + 2
  pause (ms) 13043
  turn motor 1 forward at speed Speed + 2
  turn motor 2 forward at speed Speed
  pause (ms) 12309
  set On/Off to 0
  
```

Extension Question: Adding a New Mode

The mode switching code:

```

on button B pressed
  if CurrentMode < 4 then
    change CurrentMode by 1
  else if CurrentMode = 4 then
    set CurrentMode to 0
  else
    show string "Error 2"
  
```

Could easily be edited to add an extra mode. At the moment it has room for 5 modes (0,1,2,3, and 4). When you press B you increase the mode number until it is 4, where pressing B changes the mode back to 0. To add a 6th mode with an index of 5, you would change the number '4' to '5'. That's it! Pressing B will now cycle through 6 modes.

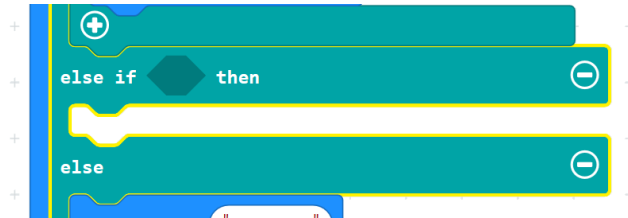
To get that sixth mode to match on to your new function, you would need to go to the bottom of the 'forever' loop:

```

else if CurrentMode = 4 then
  if On/Off = 0 then
    call LineFollow_OFF
  else if On/Off = 1 then
    call LineFollow_ON
  else
    show string "Error 8"
  +
  else
    show string "Error 9"
  +

```

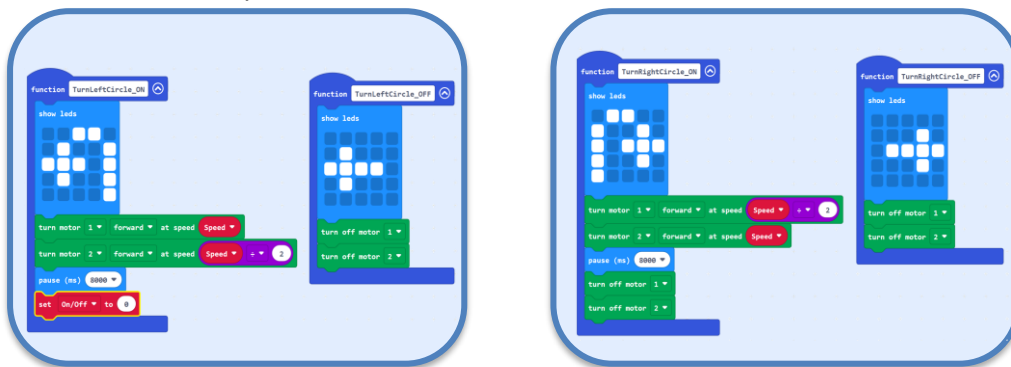
And add an extra 'else if' statement to it. To do this, you should click the '+' symbol right at the bottom of the blue/green if statement.



You can then use the ‘logic’ tab to get the ‘Comparison’ statement to make ‘else if CurrentMode = 5 then’. The code now has somewhere to go when the 6th mode (which has an index of 5 since the count starts at 0) is selected. Into this gap should be another ‘if’ statement to account for whether the motors are on or off, which should follow the format of all the other modes. It could be built without an on/off switch by just putting the ‘call function’ command in that gap though!

Extension Question: Turn in a Circle

This extension question asks the students to think about why we might write code in one way rather than another. Once the function has performed a task, the motors should be switched off. There are several ways to do this:

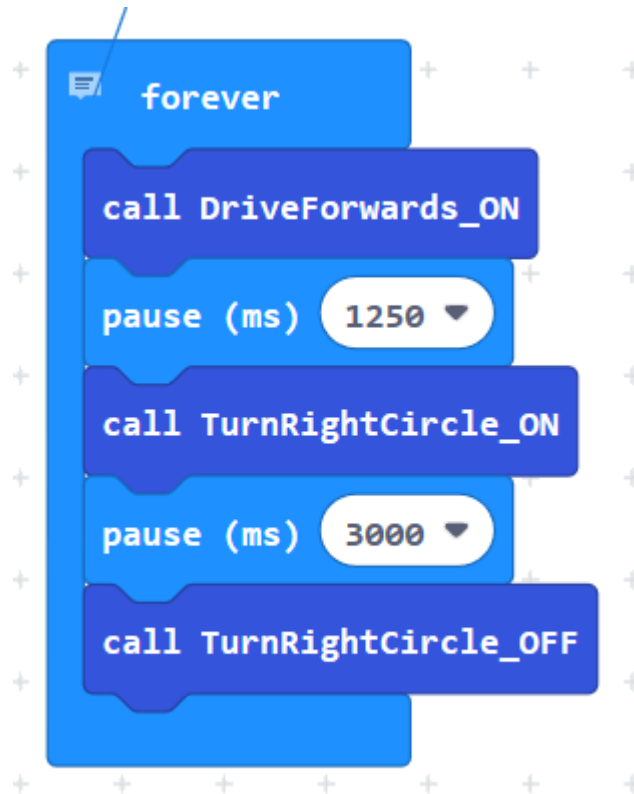


The first example sets the on/off variable to 0 while the second just switches the motors off directly. The students are asked to compare these two.

The advantage of the first option is that changing the on/off variable actually changes the state that the program is in. It will leave the ‘TurnLeftCircle_ON’ state and enter the ‘TurnLeftCircle_OFF’ state. This will change the image shown on the LED screen and to turn the motors back on all you would need to do is press ‘A’ again. The second option keeps the car in the ‘ON’ state but with the motors turned off – we don’t want the ‘ON’ state to have motors off! The symbol on the front of the microbit would not change and to start the motors again you would need to press ‘A’ twice – once to leave the ‘ON’ state to get into ‘OFF’ and then once more to get back into ‘ON’. The first option here is much preferable.

Extension Question: Obstacle Course

The students here have the choice to program quite a complicated set of instructions for the vehicle to follow in order to navigate an obstacle course. They could delete everything in the ‘forever’ loop and just start building a series of functions and pauses, e.g.:



```
forever
  call DriveForwards_ON
  pause (ms) 1250
  call TurnRightCircle_ON
  pause (ms) 3000
  call TurnRightCircle_OFF
```

The image shows a Scratch script block titled 'forever' containing five sub-blocks: 'call DriveForwards_ON', 'pause (ms) 1250', 'call TurnRightCircle_ON', 'pause (ms) 3000', and 'call TurnRightCircle_OFF'. The script is set against a grid background with plus signs at the corners of each block.

The exact sequence and pauses will obviously depend on the course the students wish to navigate.