DTiF
Digital Technologies in focus
Initiative of and funded by the Australian Government Department of Education, Skills and Employment

acara AUSTRALIAN CURRICULUM, ASSESSMENT AND REPORTING AUTHORITY

CLASSROOM IDEAS: YEARS 7–8

CREATING A DIGITAL START LINE AND FINISH LINE WITH MICRO:BITS

Designing and producing vehicles such as rubber band racers, electric vehicles, dragsters (Figure 1) or CO_2-bulb-powered rocket racers are all popular student projects. The following activity suggests one way Digital Technologies could be integrated into a unit where vehicles are being designed and produced.

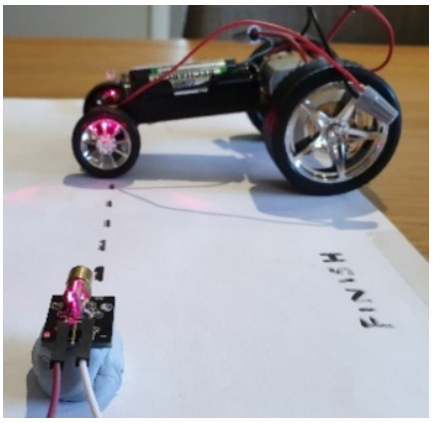

Figure 1: A dragster racer positioned at the digital finish line

Adding a Digital Technologies perspective to student racing vehicles

Let's look at this authentic context for Digital Technologies. Instead of getting students to use a stopwatch and formula to work out their vehicle's speed and velocity at the finish line, ask them to produce a digital timing system to measure time taken from the start to the finish line. They could also apply other formulas.

An example of this project is provided as a professional learning tutorial on the following pages. The coding involved to create the digital start/finish line could be a unit in its own right; however, in this project students will simply think about the instructions needed, learn how to code those instructions and then refocus on the engineering part of the unit. We recommend that teachers direct students to collect and interpret data on race times as part of the activity. This will maximise the opportunity to learn about multiple aspects of Digital Technologies learning in context.

Teachers could lead students through this unit or ask them to come up with the answers more independently. The approach depends on confidence and the time available to teach the unit.

Safety considerations

Always follow appropriate risk assessment procedures. Students will be using a laser in this activity. Lasers can do damage to eyes if not treated carefully. While this laser draws only 30 mA, it would still do damage so students need to be made aware of the safe/correct way to use the lasers.

An alternative method to using a laser transmitter and receiver is to use an infrared emitter and receiver. For instructions on this method see Appendix A on page 10.

TUTORIAL

This tutorial shows how the coding needed for the digital start/finish line can be created using both visual programming and general-purpose programming language.

Integration context: student-engineered vehicles (Design and Technologies)

The challenge: Create a digital start/finish line that captures the movement of a vehicle.

Materials list (Figure 2):

2 x micro:bits

2 x micro:bit power supply

1 x micro:bit USB connector

2 x laser transmitter (KY-008)*

https://tinyurl.com/swwg2qe

2 x laser receiver sensor (ICstation 5V) receiver*

https://tinyurl.com/wjjr7dy

4 x AA batteries

1 x AA battery pack/holder (4-battery capacity)

10 x alligator leads

Blu Tack

a computer to code the micro:bits

* See Appendix A for an infrared option.

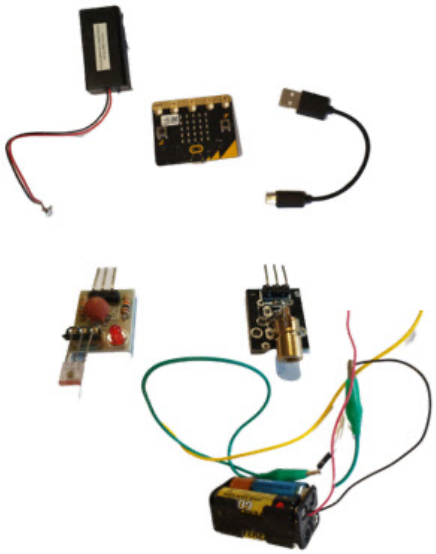

Figure 2: L t R: micro:bit power supply, micro:bit, micro:bit to USB connector , laser transmitter, laser receiver sensor (ICstation 5V), 4-battery capacity AA battery holder with alligator leads attached.

Part A: Algorithms

Suggested introductory activity

Use the ACARA computational thinking poster as a stimulus to identify the aspects of computational thinking involved in this activity. See https://www.australiancurriculum.edu.au/media/7393/computational_thinking_poster.pdf

Algorithms: Expressed as a simple sequence of steps

What is the sequence of steps needed to achieve the digital solution?

- The start line micro:bit must sense a laser/infrared beam.
- When a change in the laser beam signal is detected (a vehicle passes through it), the micro:bit must know that too.
- When change is detected, the micro:bit must start a timer.
- When change is again detected at the finish line (a vehicle passes through it) by a second micro:bit and laser, the timer must stop.
- The user must be able to see the time taken/displayed. Optionally, we can get the micro:bit to apply a formula to work out the average speed and finishing speed. These extra data, average speed and finishing speed, also need to be displayed to the user.

Algorithms: Expressed in English/pseudocode

How could these steps be expressed in pseudocode?

Start line micro:bit

```
START
   SET communications channel to 10
   SET transmission power to full (7)
   SHOW a LED to indicate the program is running
   FOREVER
      SET a variable called level to analog input of pin 0
      IF value of level reduces to less than 30
         Send a message to listening devices
         Change LED to a tick
         Wait 20 seconds (long enough for the race to finish)
   LOOP
END
```

Finish line micro:bit

```
START
   SET communications channel to 10
   SET transmission power to full (7)
   SHOW a LED to indicate the program is running
   IF a message is received
      SET a variable called CountingTime to Zero
      SET a variable called Finished to False
      Change LED display to a square to show message has been received
      WHILE Finished = False
         PAUSE 100ms
         Change CountingTime by 0.1
         IF analog input of pin 0 is less than 30
            SET a variable called FinalTime to the current value of CountingTime
            SET Finished to true
      Endwhile
   IF button B pressed
      SHOW number FinalTime
END
```

The algorithm for the start line and finish line micro:bits is used to help align the laser and the laser receiver. It should constantly display a number around (or above) 60 so it can detect change when the beam is broken. Since each micro:bit is triggered by a change in light detected, each needs to be accurately aligned. It takes practice to align the beam and keep it still. Blu Tack can help with this. If a more accurate timer is required, adjust the code so that the pause in the finish line micro:bit is reduced to 10 ms and the ***CountingTime*** is changed by 0.01.

Algorithm for start line and finish line micro:bits

```
START
   IF button A is pressed
   Display analog input value of pin 0
END
```

Part B: Implementing the solution

Step 1: Setting up the timers

Set up the equipment and then follow the instructions to check that everything is in place. First set up two laser detection systems – one at the start (start gate – Figure 3) and one at the finish (finish gate – Figure 4).

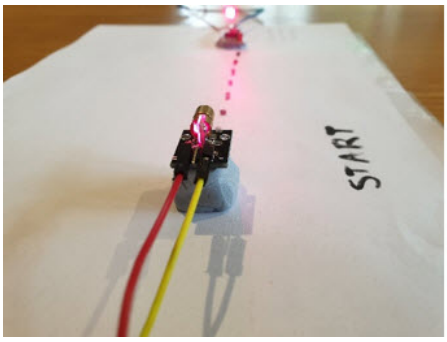

Figure 3

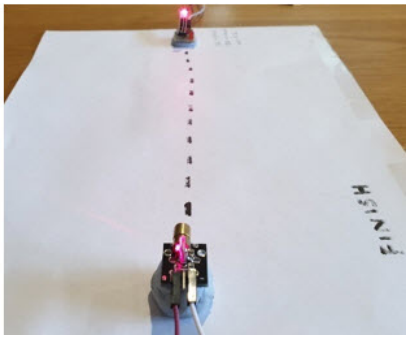

Figure 4

Each detection system needs the laser to be aligned as accurately as possible.
When the racer breaks the beam it will: start (begin the timer – Figure 5) and then finish: (end the timer – Figure 6). Note: The timing code is only on the finish line micro:bit.

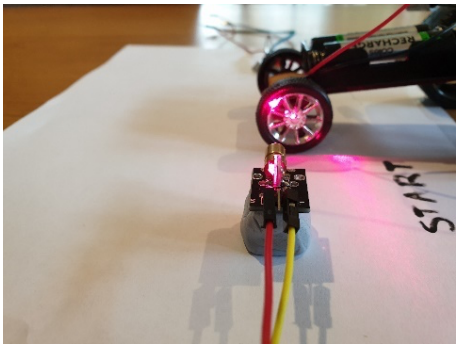

Figure 5

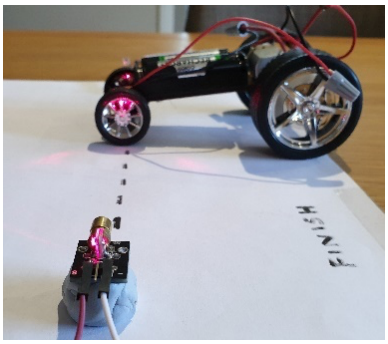

Figure 6

Step 2: Wiring the laser transmitter

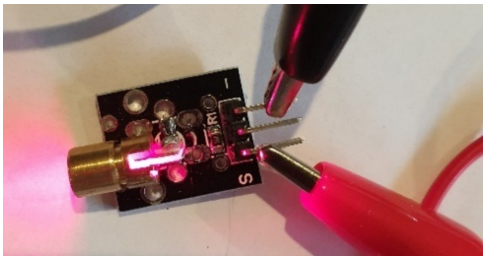

Figure 7 shows the labels '–' and 'S' on the laser transmitter. (The middle pin is not needed.)

1. Connect the negative to the pin closest to – (Figure 7).
2. Connect the positive to the pin closest to S (Figure 7).
3. Connect the other end to the battery pack as shown in Figure 8.

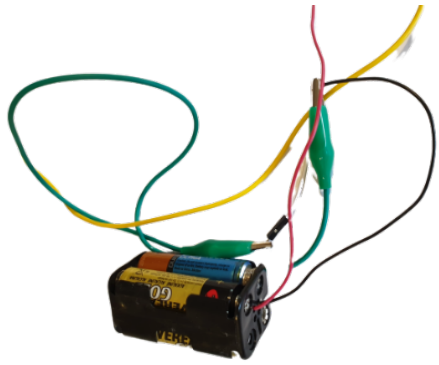

Figure 8: Step 3 is to connect to the battery pack.

Step 3: Wiring up the micro:bit to the laser receiver sensor

Figure 9 shows the sensor as it looks, taken out of the packaging. Facing you, at the top of this image is the back of the sensor (the square shape on the three upright metal legs) – it doesn't sense anything on this side so we will need to carefully bend it over so that the other side of the sensor is facing the finish line (Figure 10). That way all the connecting wires will be out of the way and off the road.

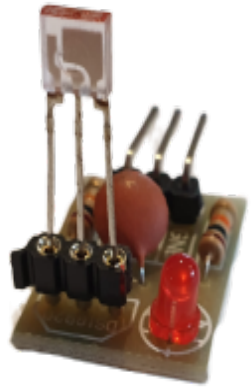

Figure 9

Next, connect the wires using what is written on the sensor board to guide you:

- **GND** to the **GND** on the micro:bit
- **VCC** to the **3V** pin
- **OUT** to the micro:bit pin you choose to use: **0** (as shown in Figure 11), 1 or 2.

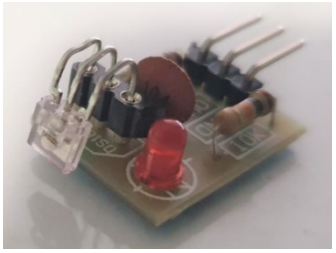

Figure 10

When you connect the alligator clips/wires to the micro:bit, the set-up should look like Figures 11 and 12.

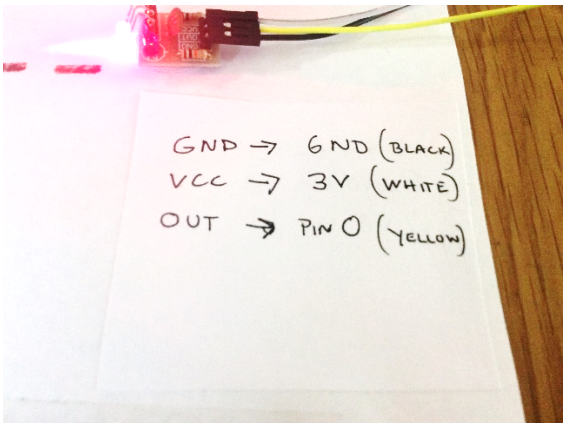

Figure 11: Laser receiver sensor

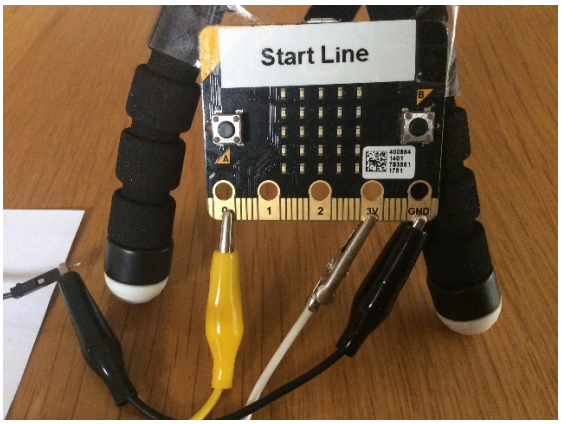

Figure 12: micro:bit

Part C: Coding (implementing)

Coding using visual programming (MakeCode)

Now we know how to connect the micro:bit to the laser receiver sensor we can write the code using www.makecode.microbit.org.

Coding the start line micro:bit using visual programming

Writing the code and loading it onto the start line micro:bit can be done one step at a time but it is shown here all at once (Figure 13).

Load this code to the micro:bit first.

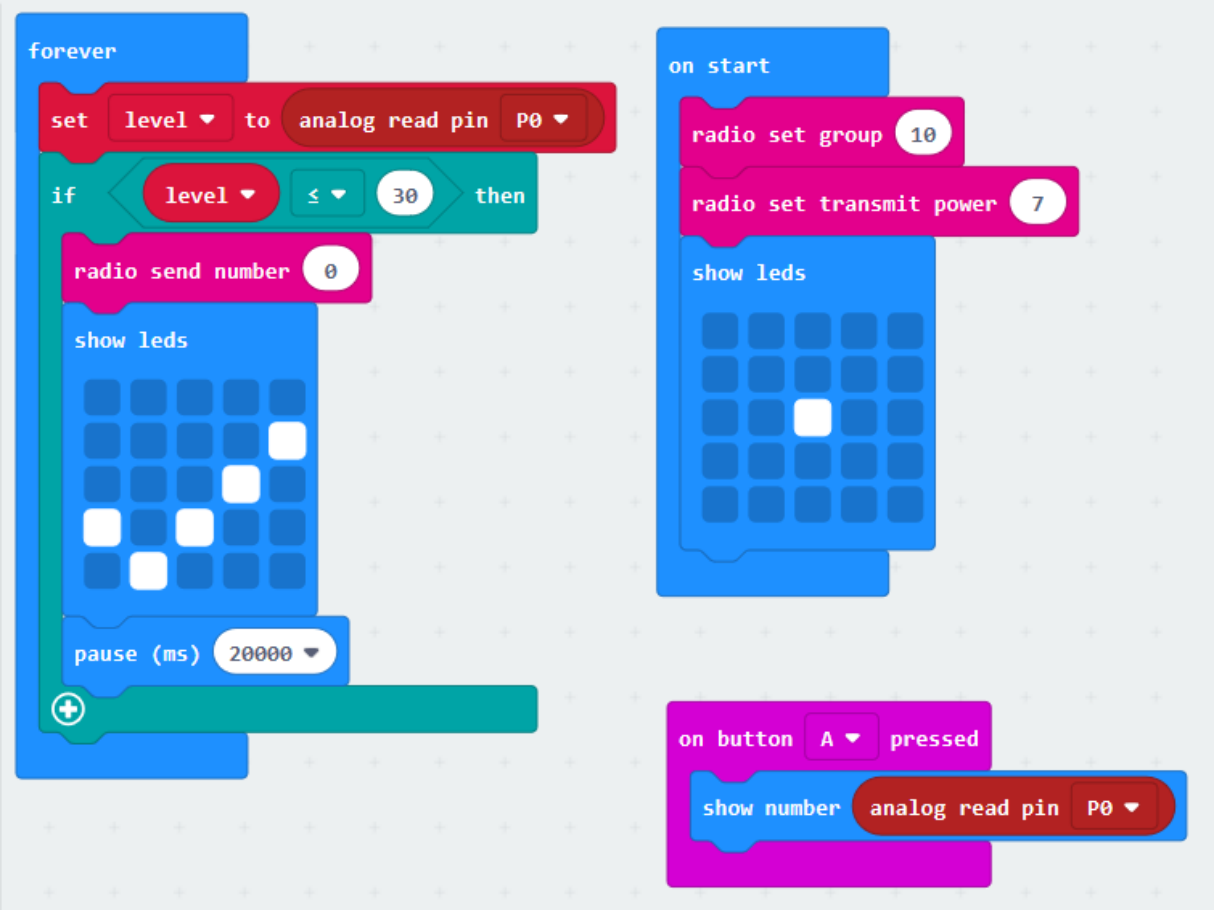

Figure 13

To understand this fully, compare it with the sequence of steps and pseudocode in Part A. Make sure you load this code into the start line micro:bit, then connect the micro:bit as shown previously.

Coding the finish line micro:bit using visual programming

The code for the finish line micro:bit is shown in Figure 14.

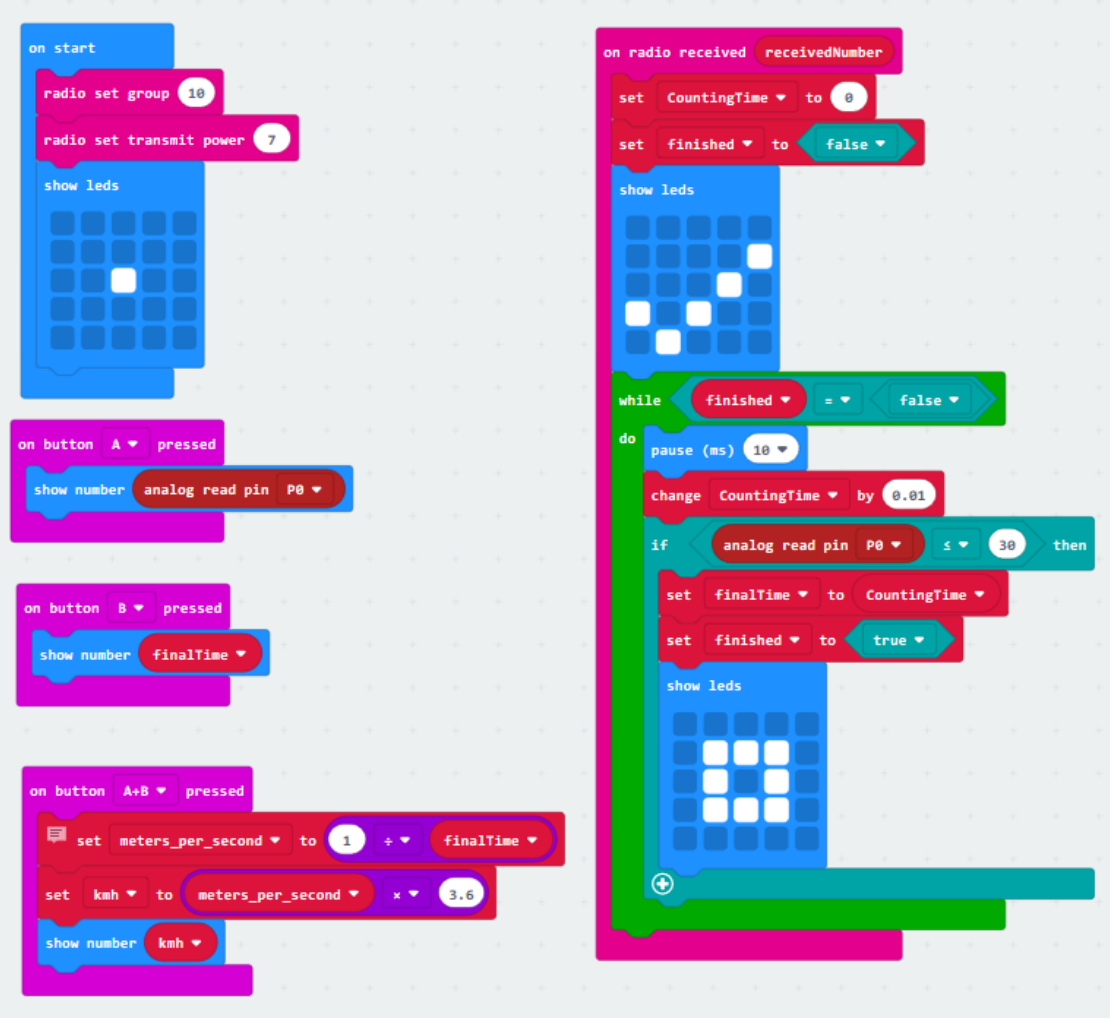

Figure 14

Again, you can compare the code in Figure 14 with the pseudocode algorithm in Part A. Load the code onto the finish line micro:bit and power it up.

Aligning the lasers

Connect the programmed micro:bits and sensors and battery pack together as shown in Figures 5, 6, 7, 8, 11 and 12 and then align each laser transmitter and receiver sensor as shown in Figures 3 and 4. (It can be tricky so be patient.)

Press button A to get the analog sensor reading. If the analog value is above 30 with the laser on and pointing directly at the sensor, then that will be enough to detect a signal is being received from the transmitter. A higher value such as 200 shows a stronger signal is being received. Note: If the table is bumped, the laser beam can move fractionally. This will upset the readings and mean the lasers will need to be realigned, so be careful.

When readings are above the threshold (30) and students are ready to go, let them test their cars and see how fast they travel. You could get them to do all the mathematics to work out the speed each time. Certainly, just looking at the time will tell the fastest cars. You could also program the micro:bit to work out the speed for you. The code shown in Figure 15 will work out the average speed in kilometres/hour.

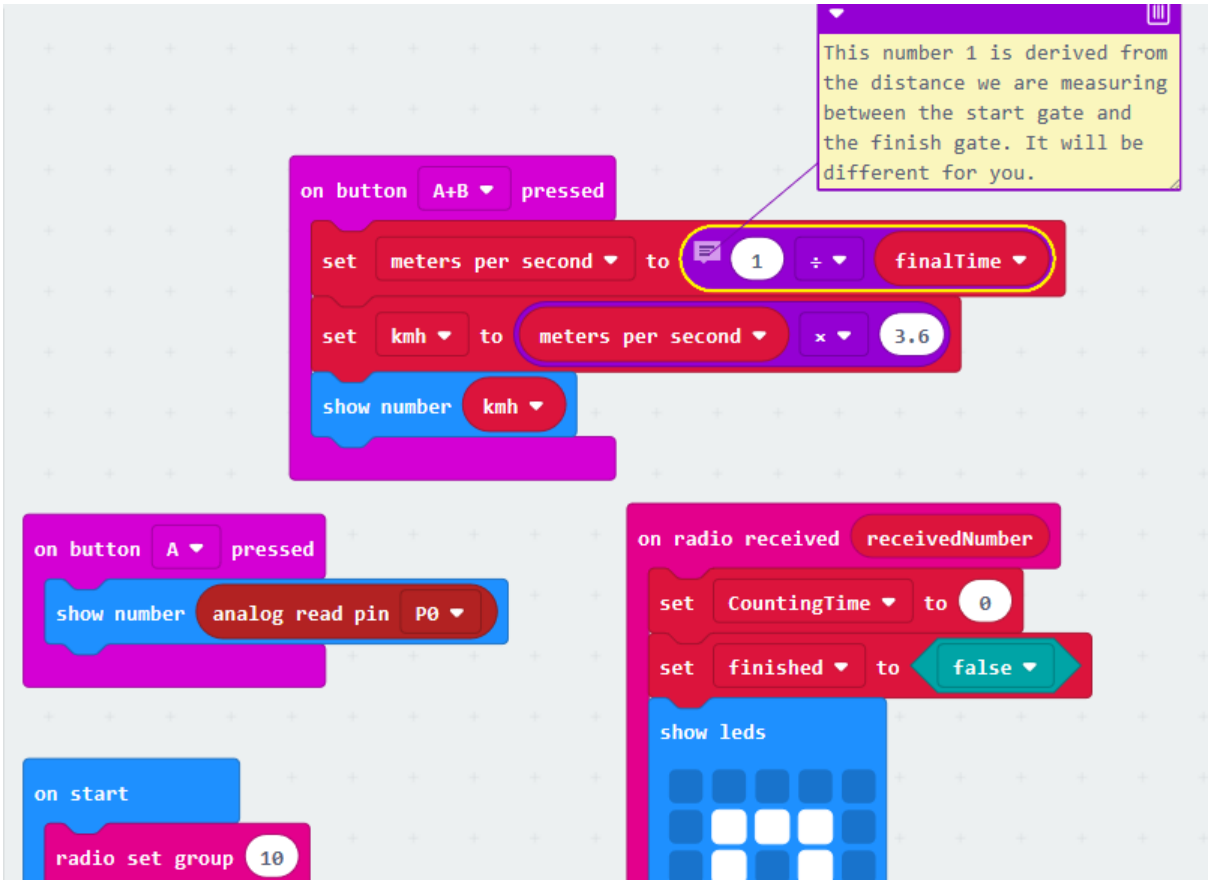

Figure 15

The ***on button A+B pressed*** code gives us the kilometres/hour data. Students could try applying the end velocity algorithm to work out the final speed as an extension. This code is added to the finish line micro:bit.

Your algorithm will vary compared with the example shown as you will probably be measuring over a greater distance than one metre. Check the coder comments in Figure 15. The video at https://youtu.be/jl4JmfPRsK4 (Figure 16) explains the whole process.

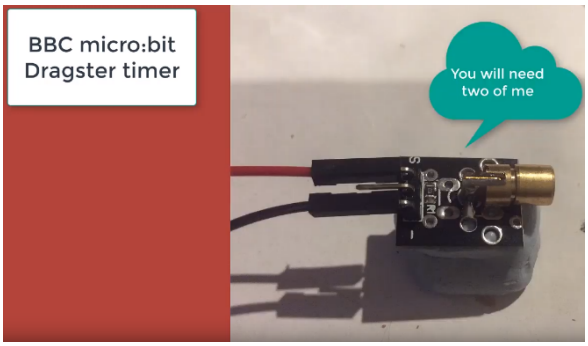

Figure 16: Screen shot from ACARA's video: Digital start finish line tutorial

Coding using general-purpose programming language (Python/MicroPython)

The wiring and set-up for the Python/MicroPython version is exactly the same as the block code version. The computational thinking and algorithms are also the same. There would be many solutions to this timing problem. Just one is presented here, which is quite similar to the block code. The code for the Python/MicroPython version is shown at Figures 17 and 18 (start and finish, respectively). Students can code in Python inside the MakeCode website or in MicroPython using an offline editor called Mu. The following code was created in MicroPython.

Note: There are an unusually large number of coder comments in this code. This is to help students understand what is going on. Students don't need to write all the comments in – as proficiency increases, the coder comments can be reduced. Comments always start with a hashtag (#).

Coding the start line micro:bit using Python/MicroPython

```
from microbit import *  # get the necessary libraries
import radio

radio.on()              # set up the communication stuff
radio.config(channel=10)
radio.config(power=7)
display.show(Image.YES)       # alert user program is running

while True:                     # loop forever
    lightVal = pin0.read_analog()  # get the pin0 value
    if button_a.was_pressed():    # used to align the laser and receiver
        display.scroll(lightVal)
    if lightVal < 30:               # if beam is broken
        radio.send(str(lightVal))   # send message out
        display.show(Image.CONFUSED)    # alert user beam has been cut
    sleep(100)
```

Figure 17

Coding the finish line micro:bit using Python/MicroPython

```
from microbit import *      # get the necessary libraries
import radio

radio.on()                  # set up the communication
radio.config(channel=10)
radio.config(power=7)

CountingTime = 0  # counts in increments of hundredths of seconds
FinalTime = 0       # used to store the time taken between gates
display.show(Image.YES)   # just lets the user know program is running

while True:                         # do this forever
    lightVal = pin0.read_analog()   # get the pin0 value
    if button_a.was_pressed():      # if button a was pressed
        display.scroll(lightVal)                        # this must come first
    incoming = radio.receive()      # check if an incoming message has arrived
    if incoming is not None:        # if it has
        display.show(Image.HAPPY)   # alert user timer has started
        finished = False
        while finished is False:    # loop continues until beam broken
            sleep(10)               # every 100th second
            CountingTime = CountingTime + 0.01  # increment by .01
            lightVal = pin0.read_analog()   # get the pin0 value
            if lightVal < 40:       # indicates beam has been broken
                FinalTime = CountingTime    # store the current CountingTime
                finished = True     # sets up the loop break
                display.clear()     # hide the smiley face
                display.show(Image.CHESSBOARD)   # alerts FinalTime has been stored
    if button_b.was_pressed():      # show the time taken
        display.scroll(FinalTime)
    if accelerometer.was_gesture('shake'): # works out km/h
        mps = 1.35/FinalTime           # get metres per second first
        kmh = mps * 3.6
        display.scroll(kmh)
```

Figure 18

Appendix A: An alternative emitter/receiver sensor method: infrared

Using lasers may be of concern in some classroom situations. An alternative approach is to use a different type of sensor. Infrared (IR) sensors, like the one shown in Figure 19, work just as well. There are 5 mm and 3 mm versions available. The 5 mm will have a slightly longer range; however, if you are placing the emitter (transmitter) and receiver close together, the 3 mm version will work just as well. A set of the 3 mm version should be easy to find for under $5.00.

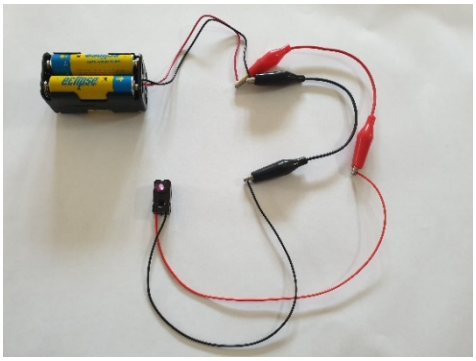

Figure 19

Wiring the IR emitter

The wiring up for this alternative sensor is very similar to the laser wiring. Connect the negative lead from a 3 V or 5 V battery pack to the negative (black) lead of the emitter. Connect the positive lead from the battery pack to the positive (red) lead of the emitter. The IR emitter is the one with only two leads. It also has a clear glass bulb to emit the infrared light.

Is it working?

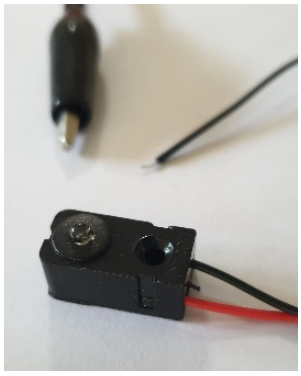

Figure 20

Figure 21

While humans cannot easily see infrared, most mobile phone cameras do detect infrared and can display it. Figures 20 and 21 depict images taken with a mobile phone with the emitter circuit broken and then complete. Figure 21 shows a red glow. This is the light being emitted. Looking directly into the top of the clear bulb, you should see a slight reddish glow when the circuit is completed. This is not harmful to your eyes, unlike looking directly into a laser.

Wiring up the receiver sensor

The receiver (Figure 22) has three leads running from it: red, black and white.

1. Connect the black wire to the **GND** of a micro:bit microcontroller.

2. Connect the red wire to the **3V** pin of the micro:bit.

3. Connect the white lead to the pin you want the input to communicate with (**pin 0** is used in Figure 22).

Make sure that the micro:bit has a power supply and the correct software downloaded to it (whether it is the start gate or the finish gate). Now you just need to align the sensor and emitter and you have a start or finish gate.

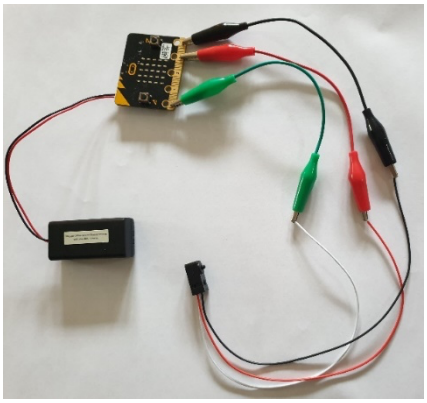

Figure 22

The coding is the same as the laser example discussed earlier. Both the block code and MicroPython versions of the code will work just as effectively for the IR start/finish lines as they do for the laser start/finish lines.

Links to the Australian Curriculum

Table 1: Aspects of the Australian Curriculum: Digital Technologies (V9) Years 7–8 which may be addressed depending on the task.

<table>
<tr><td>Digital Technologies
Achievement standard</td><td colspan="3">By the end of Year 8 students develop and modify creative digital solutions, decompose real-world problems, and evaluate alternative solutions against user stories and design criteria. Students acquire, interpret and model data with spreadsheets and represent data with integers and binary. They design and trace algorithms and implement them in a general-purpose programming language. Students select appropriate hardware for particular tasks, explain how data is transmitted and secured in networks, and identify cyber security threats. They select and use a range of digital tools efficiently and responsibly to create, locate and share content; and to plan, collaborate on and manage projects. Students manage their digital footprint.</td></tr>
<tr><td>Strand
Sub-strand</td><td colspan="3">Digital Technologies knowledge and understanding
Digital systems
Data representation
Digital Technologies processes and production skills
Investigating and defining
Generating and designing
Producing and implementing
Evaluating</td></tr>
<tr><td>Content descriptions</td><td colspan="3">explain how hardware specifications affect performance and select appropriate hardware for particular tasks and workloads AC9TDI8K01
investigate how data is transmitted and secured in wired and wireless networks including the internet AC9TDI8K02
define and decompose real-world problems with design criteria and by creating user stories AC9TDI8P04
design algorithms involving nested control structures and represent them using flowcharts and pseudocode AC9TDI8P05
trace algorithms to predict output for a given input and to identify errors AC9TDI8P06
implement, modify and debug programs involving control structures and functions in a general-purpose programming language AC9TDI8P09
evaluate existing and student solutions against the design criteria, user stories and possible future impact AC9TDI8P10</td></tr>
<tr><td rowspan="2">Technologies Core concepts</td><td rowspan="2">Systems
Systems thinking
Computational thinking
Data
Technologies processes and production skills
Interactions and impact</td><td>Digital Technologies Core concepts</td><td>Digital systems
Data representation
Data acquisition†
Data interpretation†
Abstraction
Specification
Algorithms
Implementation</td></tr>
<tr><td>General capabilities</td><td>Digital Literacy
Literacy
Numeracy</td></tr>
</table>

Cross-curriculum priorities	• Sustainability	***Learning area or subject connections***	• Science • Health and Physical Education

† *When data is a focus of the activity these additional content description(s) including Mathematics and core concepts may apply.*

Table 2: Aspects of the Australian Curriculum: Digital Technologies (V8.4) Years 7–8 which may be addressed depending on the task.

Digital Technologies ***Achievement standard***	By the end of Year 8, students distinguish between different types of networks and defined purposes. They explain how text, image and audio data can be represented, secured and presented in digital systems. Students plan and manage digital projects to create interactive information. They define and decompose problems in terms of functional requirements and constraints. Students design user experiences and algorithms incorporating branching and iterations, and test, modify and implement digital solutions. They evaluate information systems and their solutions in terms of meeting needs, innovation and sustainability. They analyse and evaluate data from a range of sources to model and create solutions. They use appropriate protocols when communicating and collaborating online.		
Strands	Digital Technologies knowledge and understanding • Digital systems Digital Technologies processes and production skills • Creating digital solutions by: – Investigating and defining – Producing and implementing – Evaluating		
Content descriptions	• Investigate how data is transmitted and secured in wired, wireless and mobile networks, and how the specifications affect performance (ACTDIK023) • Define and decompose real-world problems taking into account functional requirements and economic, environmental, social, technical and usability constraints (ACTDIP027) • Implement and modify programs with user interfaces involving branching, iteration and functions in a general-purpose programming language (ACTDIP030) • Evaluate how student solutions and existing information systems meet needs, are innovative, and take account of future risks and sustainability (ACTDIP031)		
Key concepts	• abstraction • data collection† • data interpretation† • specification • implementation • digital systems • impact	***Key ideas***	Thinking in Technologies • computational thinking • systems thinking
Cross-curriculum priorities		***General capabilities***	• Information and Communication Technology (ICT) Capability • Literacy

† *When data is a focus of the activity these additional content description(s) and key concepts may apply.*

Useful links

Coding

- Find out more about the micro:bit www.microbit.org
- Code the micro:bit at www.makecode.org
  - Code in Python inside MakeCode: https://python.microbit.org/v/1.1
  - Blockcode within MakeCode: https://makecode.microbit.org/
  - Python for beginners https://www.python.org/about/gettingstarted/
- Code in MicroPython with Mu editor. Download site: https://codewith.mu/en/download

Racing vehicles

- Sample work STEM Stage 4 – STEM Racers (NSW) https://educationstandards.nsw.edu.au/wps/portal/nesa/resource-finder/sample-work/stem/sample-work-stem-stage4-stem-racers
- STEM racer construction links (NSW) including student work samples https://educationstandards.nsw.edu.au/wps/portal/nesa/resource-finder/sample-work/stem
- Toys from trash – powered bottle car http://www.arvindguptatoys.com/toys/Poweredbottlecar.html

Competitions and challenges

- NRMA Future of transport challenge http://nrmafuturetransport.com.au/about/
- The F1 in Schools STEM Challenge https://rea.org.au/f1-in-schools/
- Let's Race (Queensland)
  The activities in this module are designed to continue the development of students' understandings of the basic principles of 'working technologically' within detailed design specifications, as they design and develop a dragster to enter the Queensland CO_2 Dragster Competition.
  https://www.qcaa.qld.edu.au/downloads/p_10/kla_tech_sbm_502.pdf
- Model solar vehicle challenge (Victoria) https://sites.google.com/view/modelsolar/home
- Synergy Schools Solar Challenge (WA) https://www.solarchallenge.net.au/

Other resources

- STARportal is a collection of STEM activities and providers. Search the STARportal to find local STEM education activities for your school. https://starportal.edu.au/
- The Girls in STEM (GiST) provides resources to inspire and inform girls, schools and families in science, technology, engineering and mathematics (STEM). Explore activities, resources, case studies, lessons, study pathways and careers.
  https://www.thegist.edu.au