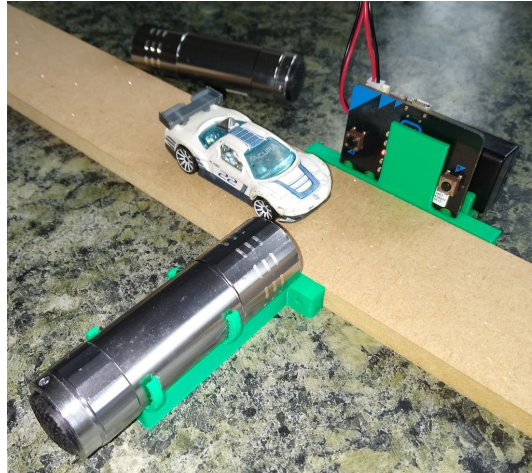# Toy Cars Timing Gate

The aim of this exercise is to find the fastest toy car by timing it over a set distance.

This will utilise:

3 Microbit
Sloping ramp
Two torches
Sticky tape/tack
Radio
While loops
If logic

How It Works

Two micro bits are used as timing traps. The torch shines a light across the track onto a Microbit. The LED on the front are also sensitive to light and when the car's shadow passes in front of the LED the drop in signal can be used to trigger the timer. One to start a second to stop. A third Microbit receives the timing signals, reads the times and calculates and displays the speed for each run.

In the Python language **spelling and punctuation are important**. Capital letters When needed are Important. Also indented code after a colon : is also important. Missing the : will result in an error. You have been warned!

Testing the Timing Gate and Car Shadow

Take one Microbit and connect to the computer. Open the Mu editor and a new program page. Save this as your 'receiver'. Start with this basic code:

```
from microbit import *
import radio
radio.on()
print("Ready")
```

The first two lines of code bring into your program information about the Microbit and how it works and also the radio to send and receive messages. The next just turns the radio on and then prints a line of text saying 'ready'. Just to prove the program is running and waiting.

Copy this code to the Microbit by pressing Flash. Then when the code has been succesfully copied across click the REPL button. A new editor window will open with >>>. This shows Python is running on the Microbit. Now press the **reset button on the back** of the Microbit. The ready message should show up and then the program stops. That is all the program does to start. You will use the REPL for the reading the light values and speeds.

Close the REPL by clicking the button again. Now add **below** your code the following:

```python
while True:
    message = radio.receive()
    if message:
        print(message)
```

Note the : and indented code after the True.

This is a forever loop. Continually checking if any 'message' is received. 'Message' is just a variable name this code gives to any information received by radio. Message is just a name though it could just as easily be 'car_passed_the_gate' or 'Josephine'. Just as long as it makes sense and you use the same expression throughout the whole code. Then if a message is received it is printed.

Now flash this code across. Disconnect the Microbit and replace it with another.

Copy this code into a new editor page and flash it onto the Microbit. Some lines are the same as before. Save time and **copy/paste the code** across from the other page.
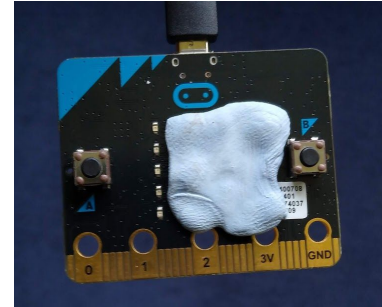
```python
from microbit import *
import radio

radio.on()

while True:
    light_level = display.read_light_level()
    radio.send(str(light_level))
    sleep(1000)
```

Here in the forever loop the code continually reads the light level falling onto the Microbit LED and sends it out by radio then waits a second before doing it again and again. The light_level variable name (sensibly named note) is sent as a string of text not a number. The Microbit radio only sends letters (text) not mathematical numbers. The value sent over radio is not a number you can do maths with, unless it is reconverted back into a number (integer).

Disconnect the the Microbit. Stick a piece of tape or tac over the LED on the front leaving just the left side clear. This makes for a nice sharp timing gate. Tape the Microbit to the side of the track and a torch across from it shining onto the LED. Plug in the battery pack.

Reattach the original Microbit to the computer and start the REPL. Remember to restart the Microbit.

Slowly place a car in front of the sensor and watch how the values drop. Note down a value that could be used to determine the point that a car crosses the gate. (Our tests with our torches and width of track gave 280 for no car and around 100 for the cross point).

Starting the Start, Stopping the Stop

Stop the REPL, remove the Microbit and reinsert the one that is the timing gate. **Change** the timing gate code **utilising your value recorded** above.

```
from microbit import *
import radio

radio.on()

while True:
    light_level = display.read_light_level()
    if light_level < 100:
        radio.send("Start")
        sleep(1000)
    sleep(10)
```

This new code now only sends the message, which is now the word 'Start", when the value drops to below your start value.

Use the same code for the third Microbit to be the stopping gate. Change the radio.send from 'Start' to 'Stop'.

The Timer

The final coding part is to calculate the speed from the arrival times of the 'Start' and 'Stop' signals.
The Microbits count, in milliseconds, the amount of time that has passed since they are switched on. By recording the time at the point of receiving the two messages and subtracting the start from the stop the elapsed time is known. If you then know the distance between the two points on the track the speed can be calculated.

Set up the Track

Place the Start and Stop timing gates on the side of the track with their torches. Place them a measured distance apart from LED to LED. We used 250mm or 0.25m for our example.

The final piece of the code is then this for the receiver. **Change** the code you already have to match this but with your distance in metres. So half a metre is 0.5. The new code is in blue.

The ticks_ms is the millisecond timing information to use in the code. One millisecond is a tick of the clock.

```
import radio
from time import ticks_ms
```

The start_time, and stop, are variables that are used to briefly hold the start and stop values for the calculation. Set to zero just to start with.

```
radio.on()
print("Ready")
start_time = 0
stop_time = 0
```

This is the big bit of code. The if and elif check to see which message is received when a message is received. The start_time (or stop) are then set to the value of the ticks as described above. These times are then printed onto the screen under the REPL.

```python
while True:
    message = radio.receive()

    if message == 'Start':
        start_time = ticks_ms()
        print("Start at " + str(start_time))

    elif message == 'Stop':
        stop_time = ticks_ms()
        print("Stop at " + str(stop_time))
```
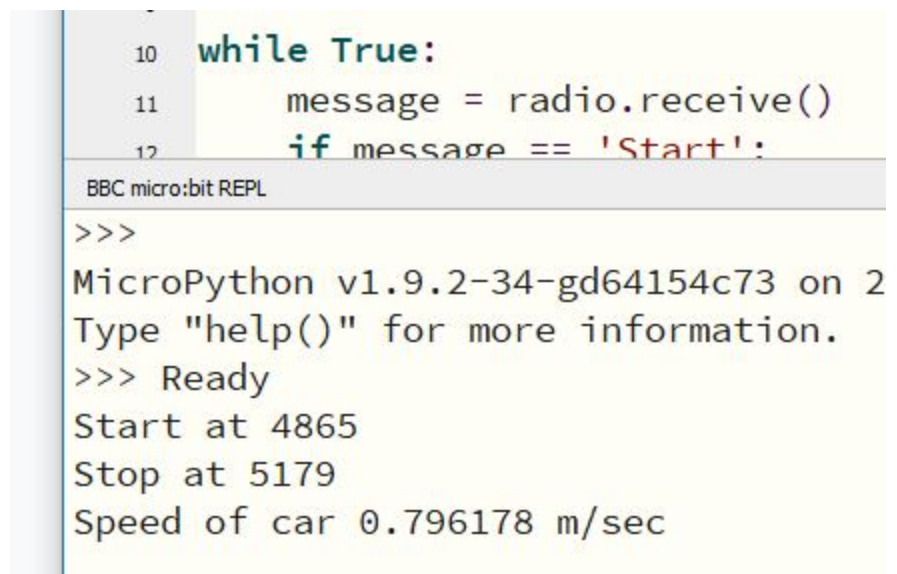
Once the stop message has been received the calculation is made. The car's speed is the distance travelled between the two gates divided by the difference between the two times and then all multiplied by 1000 to convert milliseconds to seconds. And the result is printed.

```python
        car_speed = (0.25/(stop_time - start_time))*1000
        print("Speed of car " + str(car_speed) + " m/sec")
```

Here is the display showing two times and the speed.

So which of your cars is the fastest, or slowest!

```
10  while True:
11      message = radio.receive()
12      if message == 'Start':
```

BBC micro:bit REPL

```
>>>
MicroPython v1.9.2-34-gd64154c73 on 2
Type "help()" for more information.
>>> Ready
Start at 4865
Stop at 5179
Speed of car 0.796178 m/sec
```