

D - Robot break time - make a game!

Even robots need to rest sometimes - let's build a reaction timer game to play when we have some time off from the mission.



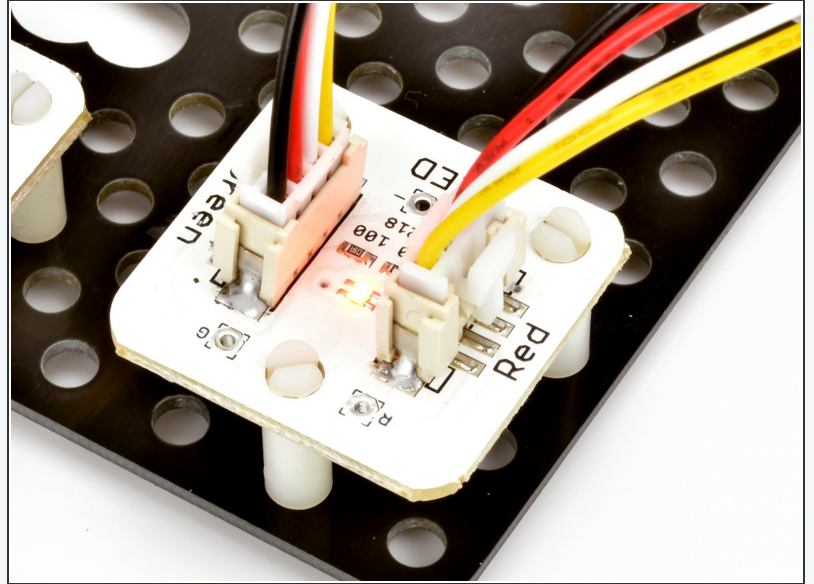
INTRODUCTION

Even robots need to rest sometimes - let's build a reaction timer game to play when we have some time off from the mission.

Step 1

Orange Lights

- So far we have only used our green and red LEDs separately - however you actually have an **orange LED** as well!
- Because both LEDs are in the same component, if we turn on the green and red LEDs **at the same time**, it will look orange.
- Plug your green and red LEDs into your board, and write a simple program to turn them **both** on and check it looks orange.



Step 2

What is a 'Reaction Timer'?

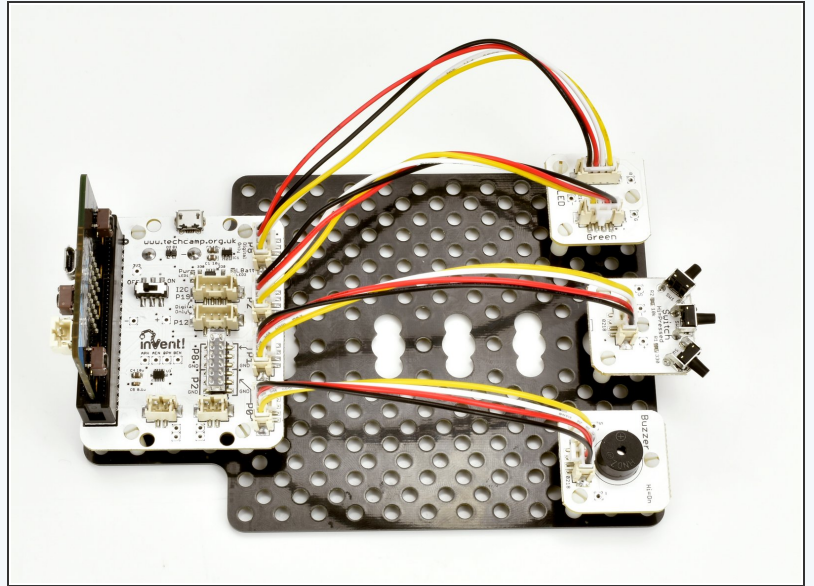
- A Reaction Timer is a game that tells you how **quickly** you managed to react to something.
- For our timer, we're going to see how long it takes the player to **press a button** when a **red LED** comes on.
- After they press the button, it will tell the player how well they did:
 - A **green** light for a really fast reaction (less than 0.1 seconds) - good job!
 - An **orange** light for an average reaction (less than 0.2 seconds)
 - A **red** light for a slow reaction (more than 0.3 seconds)
- After telling them how well they did, the game should **automatically restart** so they can play again.



Step 3

Game Plan

- Our program for the reaction timer game is going to be quite **complicated!**
- Let's **think through the plan first** so we know what we're doing:
 - Wait **5 seconds**, turn on **red** LED
 - Wait **100 milliseconds** - **if** the player has pressed the button already, turn on green LED **only** (really fast reaction!)
 - Wait **another** 100 milliseconds - **if** player has pressed the button now, turn on **green and red** LEDs (average reaction time)
 - Wait a final **100 milliseconds** - **if** player has pressed button, turn on **red LED** (slow reaction)
 - Wait a few seconds for the player to see their result, then **go back to the start** and repeat everything **forever!**
- Your game device should be assembled the same way as for the previous section - check the picture if you can't remember.



Step 4

Starting the Game

- Now we know the plan, start with a **blank program**. Don't forget the **invent code at the top** we always need!
- Add some lines of code to the **loop** to make the program **sleep** for 5 seconds, then **turn on the red LED**.

```
9      p[6]=(p[5]+p[3])/2;p[7]=(p[4]+p[2])
10     while(p0()>p[7]and p1()>p[6]):d(0 i
11     def digital_read_line(s): return 1 if (
12     def analog_read_line(s): v=p0()if s==0
13     # Invent! Code End
14     # Start your code below here!
15
16     sleep(5000)
17     pin8.write_digital(1)
18
19
20
21
22
23
24
```

Step 5

Waiting for Switches

- Now we need to put in three similar sections of code, to see if the player has **fast, average or slow reactions**.
- Remember the plan? Each section should:
 - **Sleep** 100 milliseconds
 - Check **if** the switch is pressed
 - If it is, **turn on** the correct colour LED for fast, average or slow
- You should end up with **three sections** which all look quite similar, which should go in the loop **after** the code that turns on the LED after 5 seconds.
- Check the picture for a hint of how to do the **first** one - you'll have to do the next two **on your own!** This is only one of the sections by itself, and you'll have to add it to the program you made in the last step.

```
sleep(100)
if pin1.read_digital() == 1:
    pin2.write_digital(1)
    pin8.write_digital(0)
```

Step 6

Finish the game

- At the end of the game, we need to **wait** for a few seconds so the player can see how they did, and then make sure **both LEDs are turned off** before the game starts again.
- **Add** these finishing touches - check the picture for a **hint** on the lines of code you might need!

```
sleep(5000)
pin2.write_digital(0)
pin8.write_digital(0)
```

Step 7

Test your game!

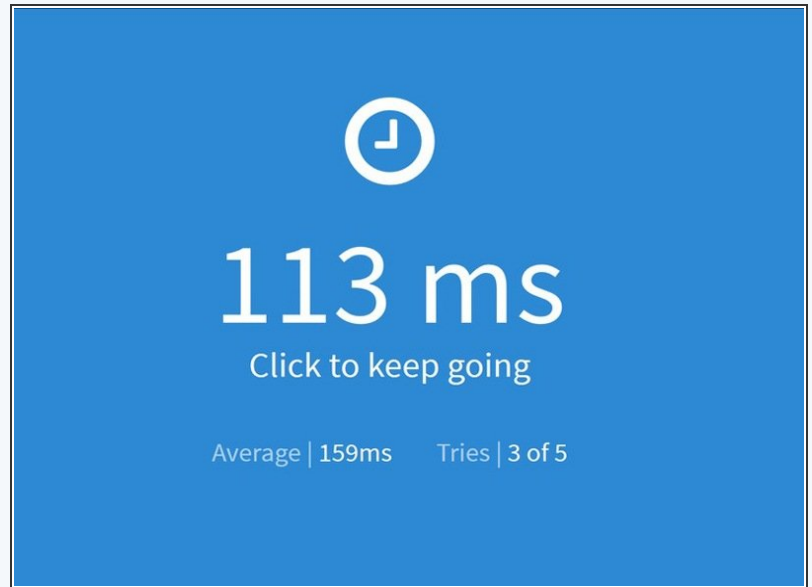
- An important part of writing programs is to make sure you **test them properly**, so you can **fix** any problems!
- You don't want to give a program to someone to use, only for them to have loads of problems which you could have found by **testing it first**.
- **Test your program** with your neighbour - see if either of you can get a green reaction time!
- If you still can't get your program to work, have a look at the example in the picture. This **isn't the only way** to program the game, and some of the code is not required - **can you work out which parts?**

```
13 # Invent! Code End
14 # Start your code below here!
15
16 while True:
17     sleep(5000)
18     pin8.write_digital(1)
19     sleep(100)
20     if pin1.read_digital() == 1:
21         pin2.write_digital(1)
22         pin8.write_digital(0)
23     sleep(100)
24     if pin1.read_digital() == 1:
25         pin2.write_digital(1)
26         pin8.write_digital(1)
27     sleep(100)
28     if pin1.read_digital() == 1:
29         pin2.write_digital(0)
30         pin8.write_digital(1)
31     sleep(5000)
32     pin2.write_digital(0)
33     pin8.write_digital(0)
34
```

Step 8

Fast Reactions?

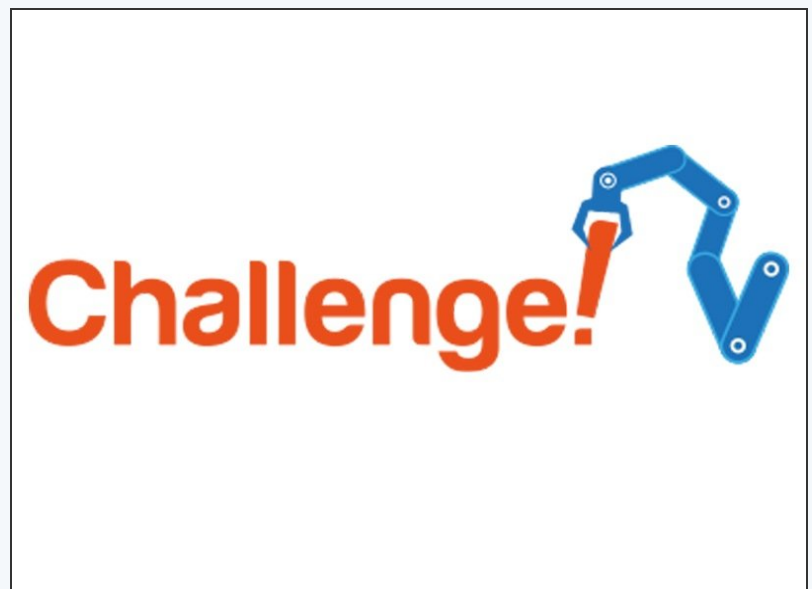
- You will find it **very difficult** to get a green result (without cheating by holding down the switch!).
- This is because the average human reaction time is **over 200 milliseconds**.
- You will probably only be able to do it by **counting down** the 5 second delay in your head.
- Even the fastest recorded reaction time is **over 100 milliseconds** - this is because it **takes time** for the signal from your eyes to reach your brain, then the signal from your brain to get sent down your nerves to move your muscle to press the switch.
- If you want to measure your reaction time accurately, head to this site:
<http://www.humanbenchmark.com/tests/reac...>
<http://www.humanbenchmark.com/tests/reactiontime>



Step 9

Reaction Timer

- Let's make some changes to the game using the **buzzer**, to make it easier to use. You need to:
 - Make the LED turn **green** when the player needs to press the switch, and **red** during the 5 second delay when they shouldn't.
 - Instead of the LED colour showing how well you did, make the buzzer buzz **once for each 100 milliseconds** it took the player to press the switch.
 - For example, the 'fast' reaction time is 100 milliseconds, so the buzzer should buzz **once**. The 'average' reaction time is 200 milliseconds, so it should buzz **twice** (and so on).
- Make sure to **test** your program afterwards!



Step 10

Reaction timer with buzz count

- Let's improve our reaction timer game even more.
- Add some more IF statements and sleeps so your reaction timer can count up to **500 milliseconds**.
- Inside your IF statements, **use for loops** to make your buzzer buzz the correct number of times, so you can reduce the number of lines in your program.
- Don't forget to **save your program**, you might need it later!

Extension Challenge! 