Red Light Green Light

This is the classic “Red Light, Green Light” game where one person is a virtual stoplight and gives commands to the other players to either stop or go.

Rules of play

The player chosen as the current the stoplight says “Green Light!” and turns away from the other players. The other players move toward the stoplight player, from a distance set at the beginning of the game, and try to touch them. The stoplight player can at any time say “Red light!” and then turn around to face the other players. If the stop light player sees anyone still moving, they call them out and they are finished playing until a new game is started. The stoplight player repeats the red light, green light cycle. If one of the other players happens to touch the stop light player before they can turn around when saying “Red Light!”, then the current stoplight player moves to the beginning of the course and the other player becomes the stoplight. The game continues until only the the stoplight player remains.

In this remake of the game, we will use a micro:bit, its radio, and the accelerometer to enforce these rules!

Multi editor

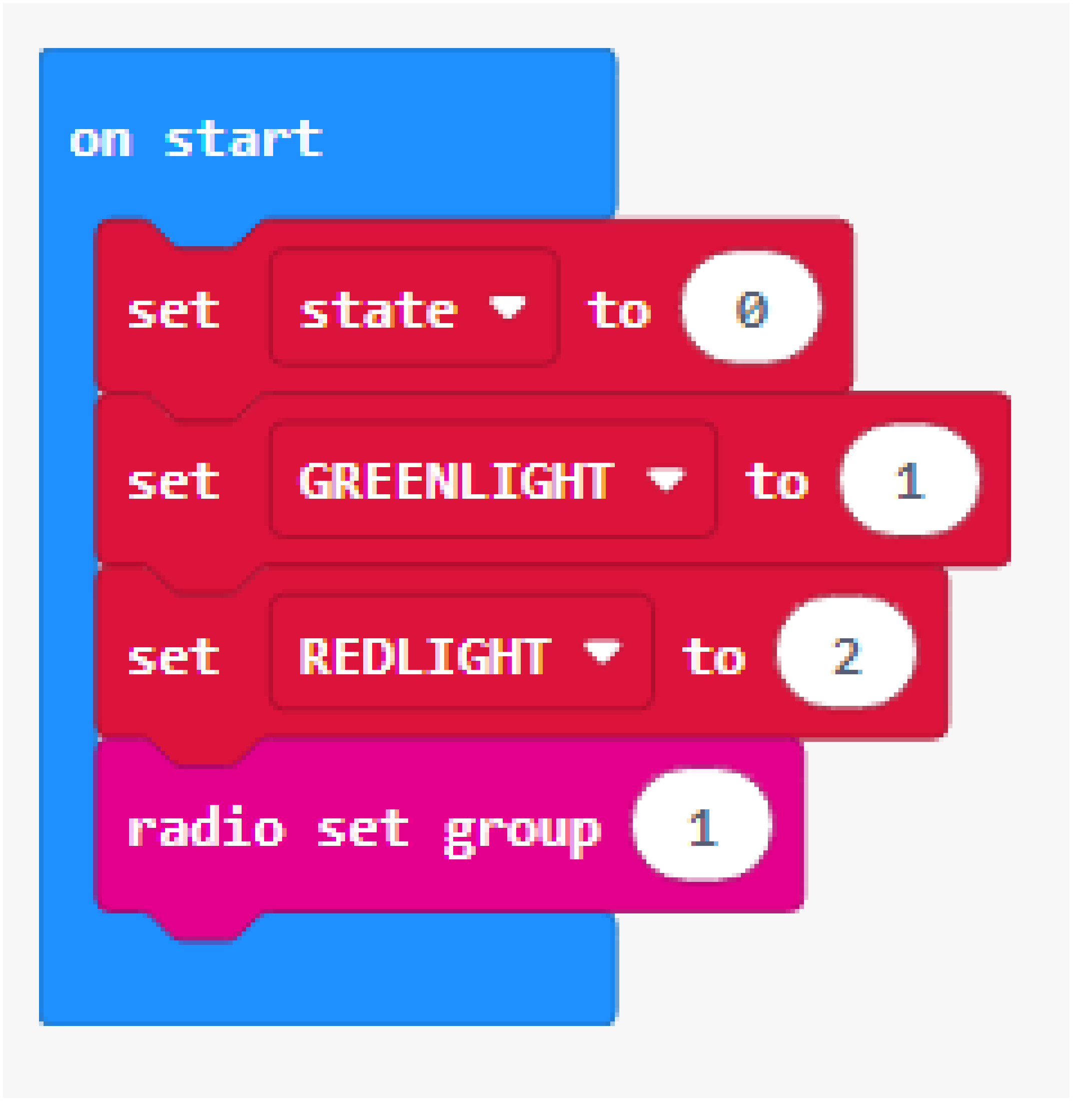
This project uses radio to communicate status to other micro:bits. It’s helpful to know how a second micro:bit will respond when it’s sent a radio message. You can code and test two radio programs using the multi-editor feature.
Open **https://makecode.com/multi** to launch 2 side-by-side micro:bit editors.

Creating the stoplight

Let’s start with the code running on the stoplight’s micro:bit. Don’t use this code for the other players!

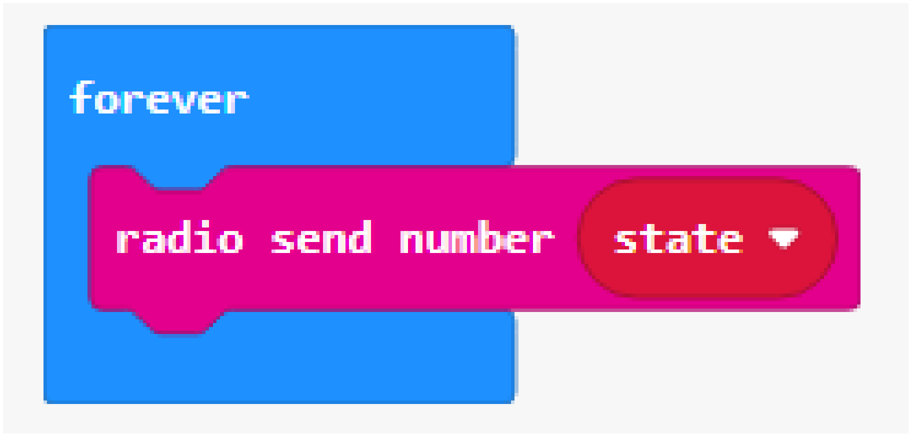
States

We define two *states*, or game conditions, called `GREENLIGHT` and `REDLIGHT`. A variable named `state` will store the current game state. When the stoplight player presses `A`, the game goes into “green light” mode. When they press `B`, the state goes into “red light” mode. The radio group for all game players is set to `1`. We will set the same group in the player’s code too.


```
on start
    set state to 0
    set GREENLIGHT to 1
    set REDLIGHT to 2
    radio set group 1
```

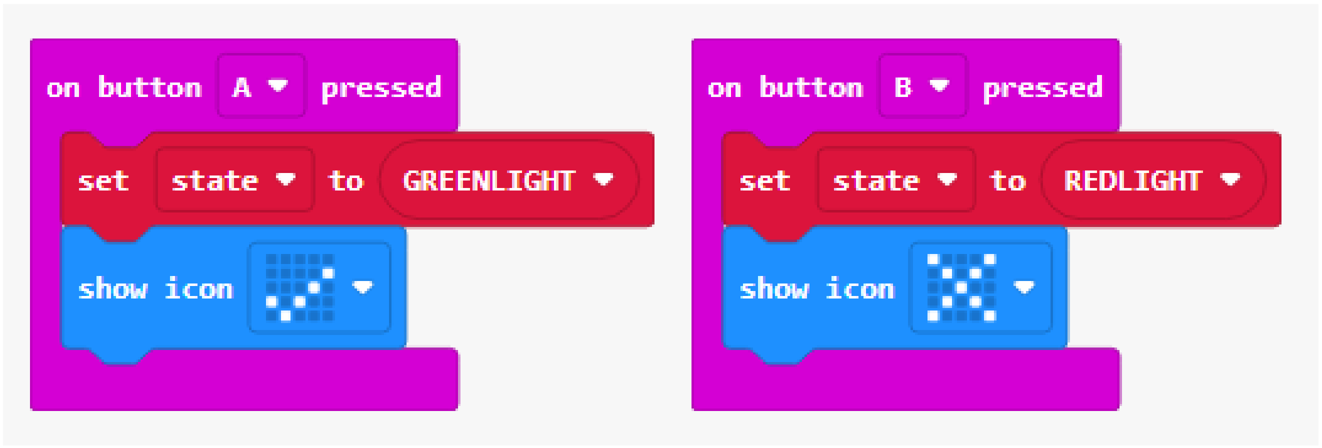
Communication

A `forever` loop will broadcast the game state so that players continuously receive it.


```
forever
    radio send number state
```

Red light, green light

Use the `on button pressed` block to run code when button `A` and `B` are pressed. When `A` is pressed, the game goes into `GREENLIGHT` mode. When `B` is pressed, the game goes into `REDLIGHT` mode. We also use `show icon` to display the current game state.

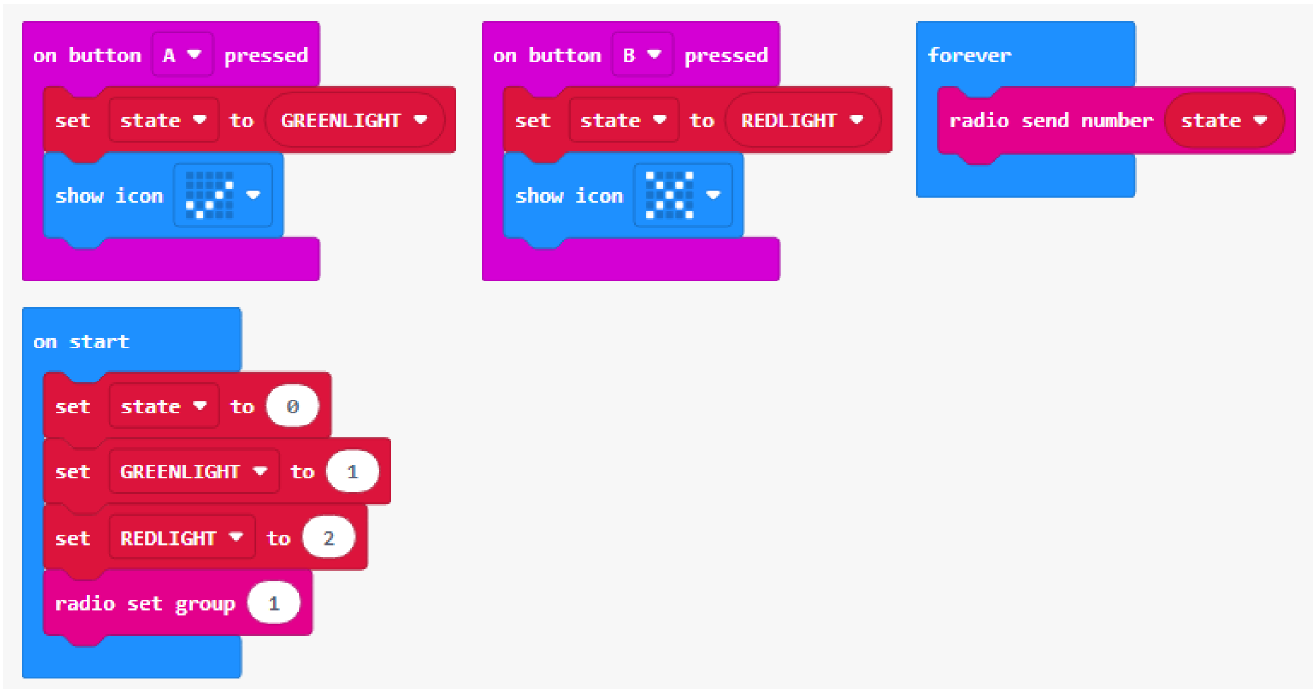


```
on button A pressed
    set state to GREENLIGHT
    show icon

on button B pressed
    set state to REDLIGHT
    show icon
```

Stoplight code

All together the stoplight code looks like this:



```
on button A pressed
    set state to GREENLIGHT
    show icon

on button B pressed
    set state to REDLIGHT
    show icon

forever
    radio send number state

on start
    set state to 0
    set GREENLIGHT to 1
    set REDLIGHT to 2
    radio set group 1
```

Improve the game

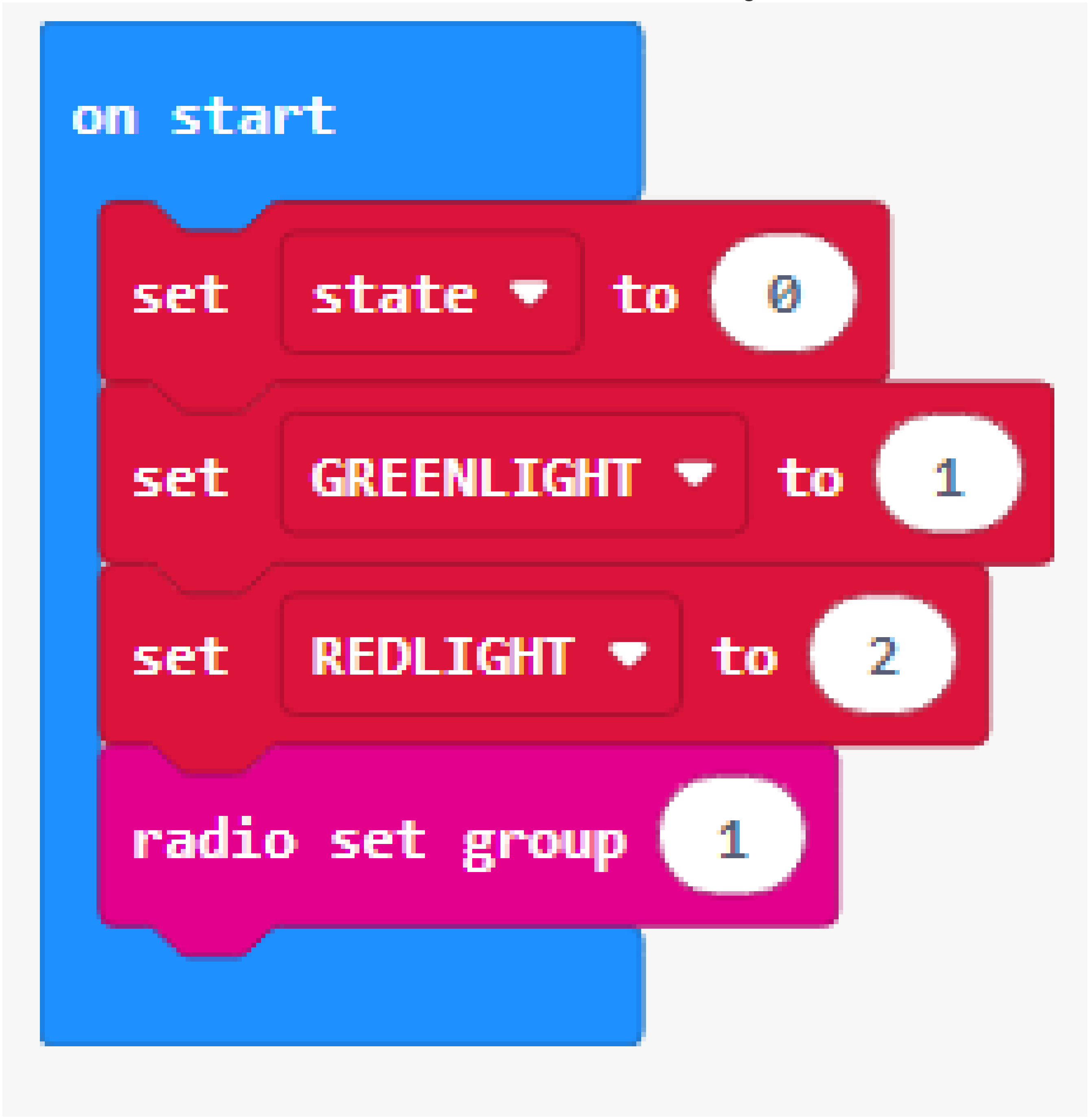
- Use `ring tone` to play a sound while the game is in `GREENLIGHT` mode.
- Attach a servo and move the arm based on the game state.

The players

The code for the other players needs to listen for the stoplight's state.

States

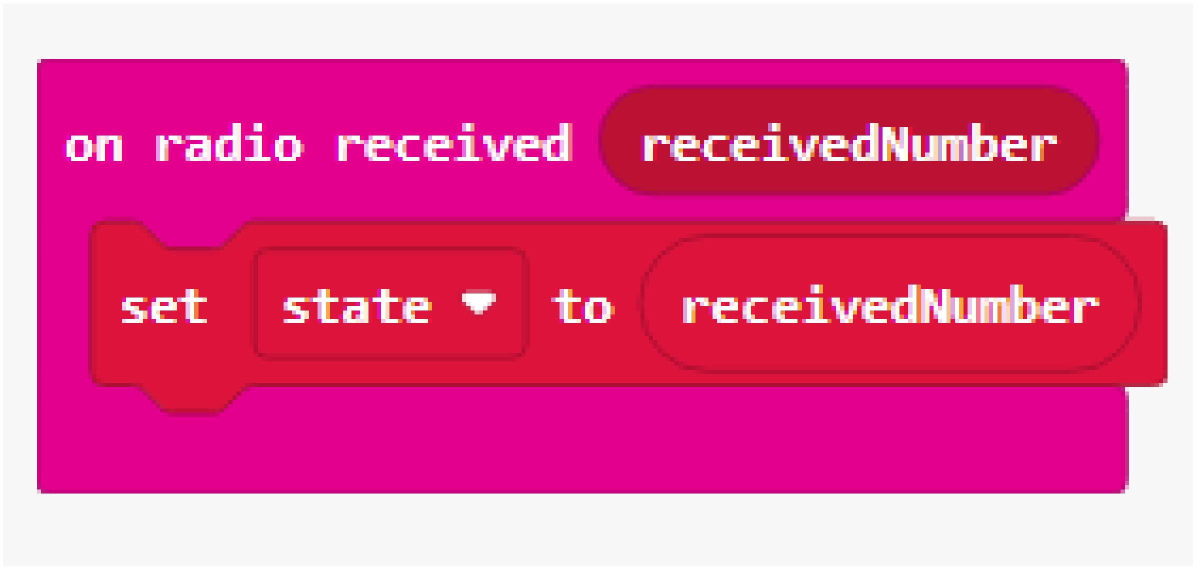
First, we again define the state constants `GREENLIGHT`, `REDLIGHT`, and set the radio group to `1`. We also add a `state` variable that will store the state of the game.



```
on start
    set state to 0
    set GREENLIGHT to 1
    set REDLIGHT to 2
    radio set group 1
```

Communication

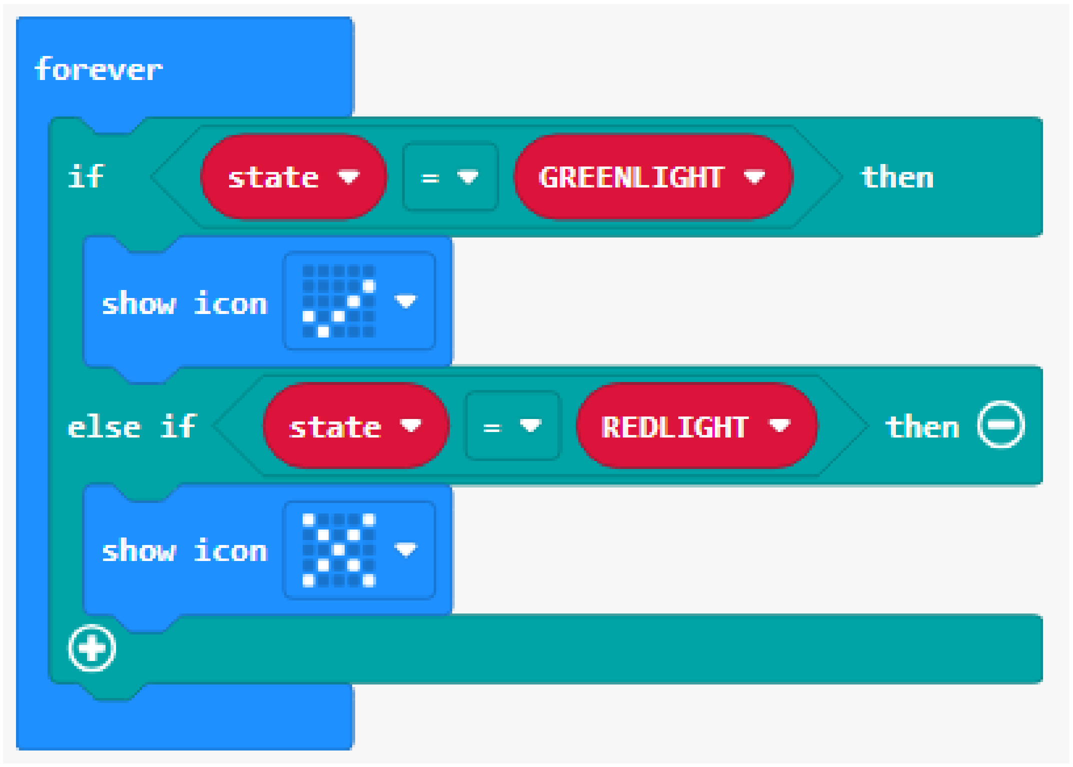
We use the `on received number` block to store the stoplight state into the `state` variable.



```
on radio received  receivedNumber
    set  state ▾  to  receivedNumber
```

Display

In a `forever` loop, we display different icons based on the game state. Use a `if` and `show icon` blocks to display the game state.



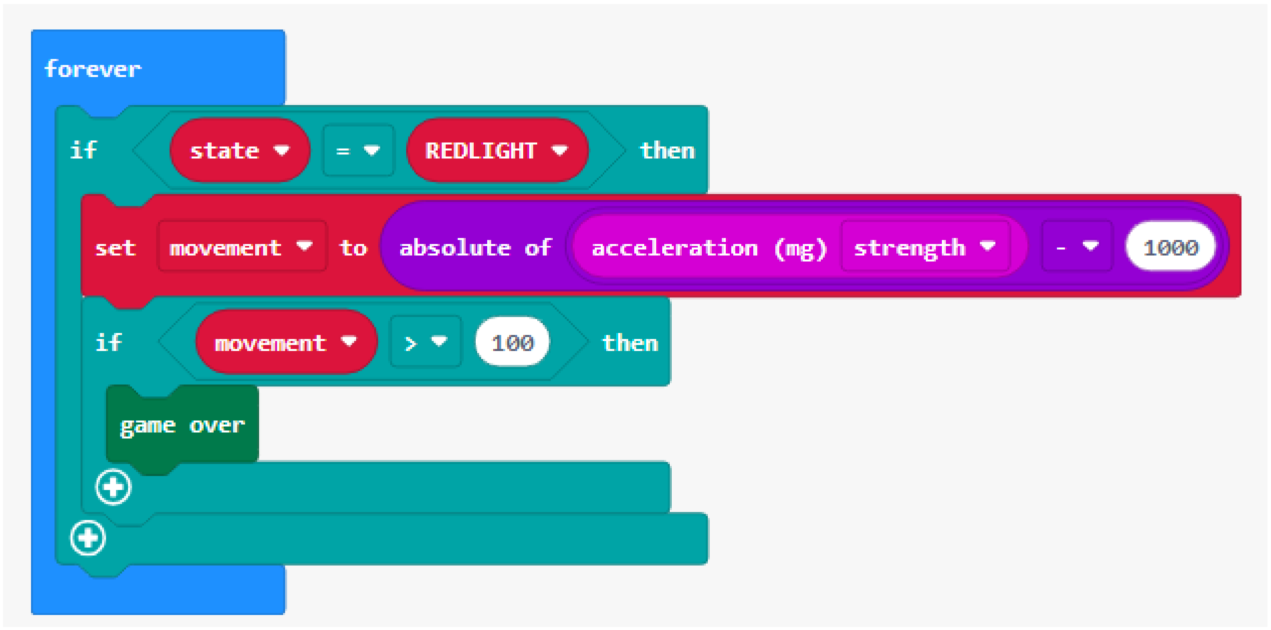
```
forever
    if  state ▾  = ▾  GREENLIGHT ▾  then
        show icon ▾
    else if  state ▾  = ▾  REDLIGHT ▾  then ⊖
        show icon ▾
    ⊕
```

Movement check

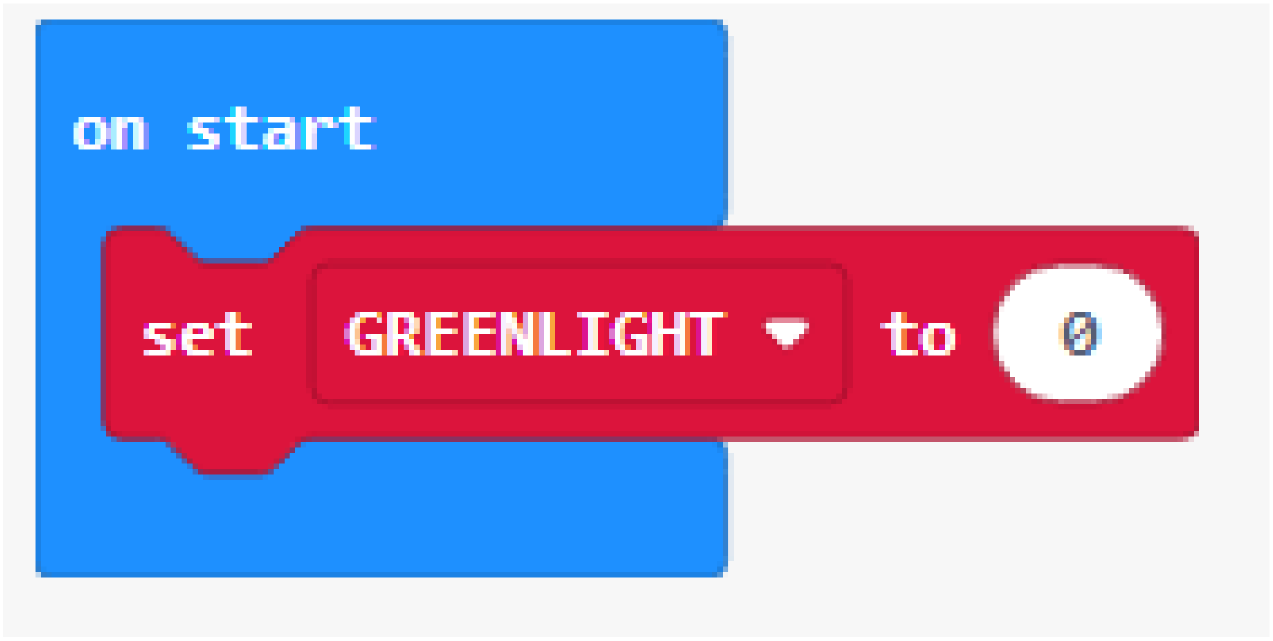
If the `state` is equal to `REDLIGHT`, we need to check that the player is not moving. This is where the accelerometer comes into play. The accelerometer measures forces applied to the micro:bit. If the player moves, it's likely that the accelerometer will detect any small forces applied to the micro:bit. At all times, gravity is applied to the micro:bit, so the acceleration strength at rest is always near `1000` mg. If the acceleration strength is far from that value, say `1100` or `900`, we can assume that the player is moving. To compute this we use the formula:

```
movement = | acc strength - 1000 |
```

Now that we know the math for it, we can turn this into code.



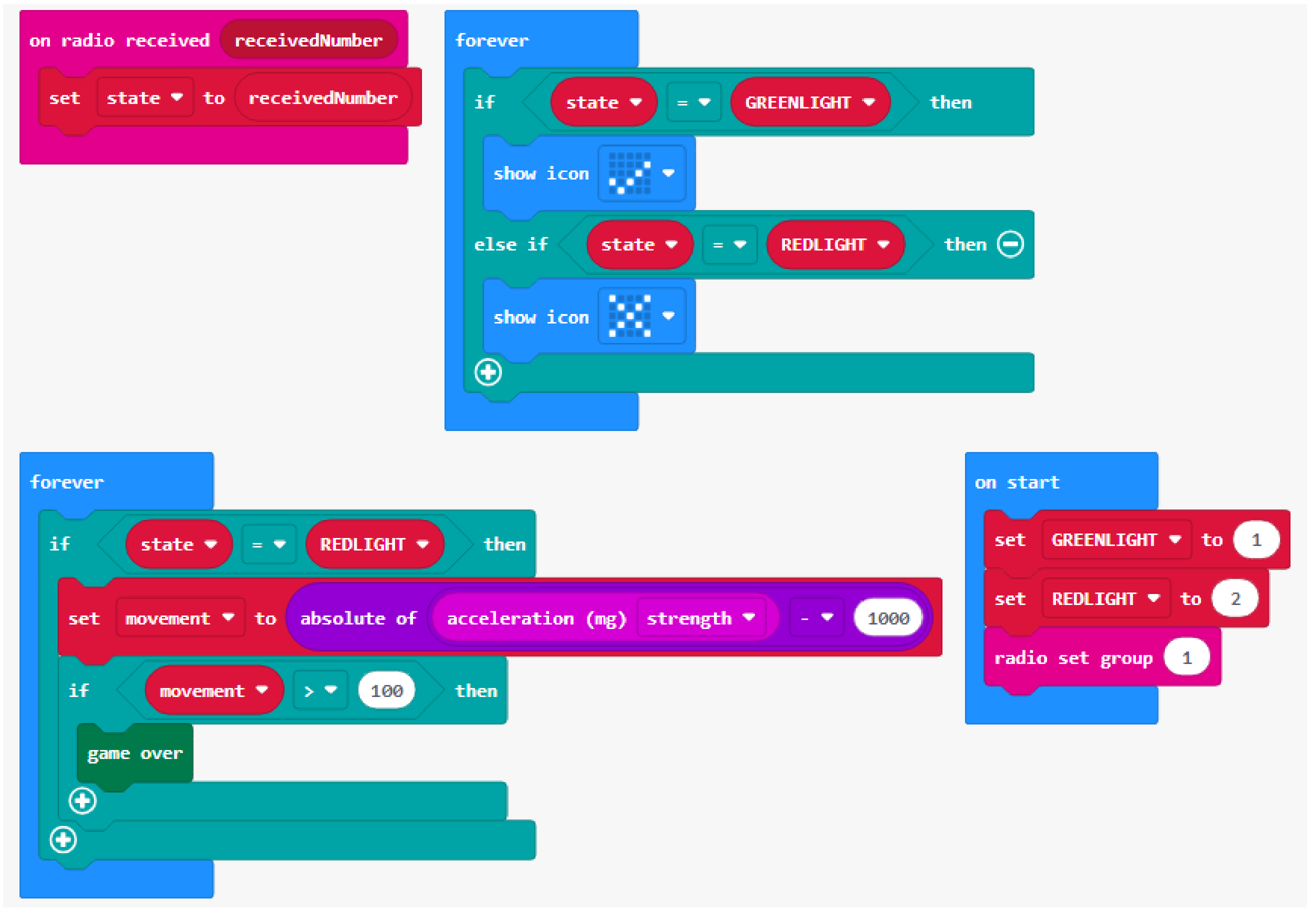
```
forever
  if state = REDLIGHT then
    set movement to absolute of acceleration (mg) strength - 1000
    if movement > 100 then
      game over
```



```
on start
  set GREENLIGHT to 0
```

Player code

All together, the code for the players is:



```
on radio received receivedNumber
    set state to receivedNumber

forever
    if state = GREENLIGHT then
        show icon
    else if state = REDLIGHT then
        show icon

forever
    if state = REDLIGHT then
        set movement to absolute of acceleration (mg) strength - 1000
        if movement > 100 then
            game over

on start
    set GREENLIGHT to 1
    set REDLIGHT to 2
    radio set group 1
```

Tuning

Does the movement check work? Try changing the `100` value to tune the detection sensitivity. Try `64` maybe.

Improve the game

- Use `ring tone` to play a tone while in green mode.
- Use the packet signal strength to detect that you've reached the stoplight.