Australian
Computing
Academy

DT Mini Challenge

Blastoff with micro:bit (Blockly)

1. Launch the micro:bit
2. Repeating things
3. Making Decisions
4. Blast Off!

1

LAUNCH THE MICRO:BIT

1.1. Prepare for Launch

1.1.1. Go for Launch!

Welcome to our rocket mini challenge!

You can have fun with the challenges in this course and use the simulator to see your code come to life.

To make a physical rocket you'll need a real micro:bit and the following instructions.

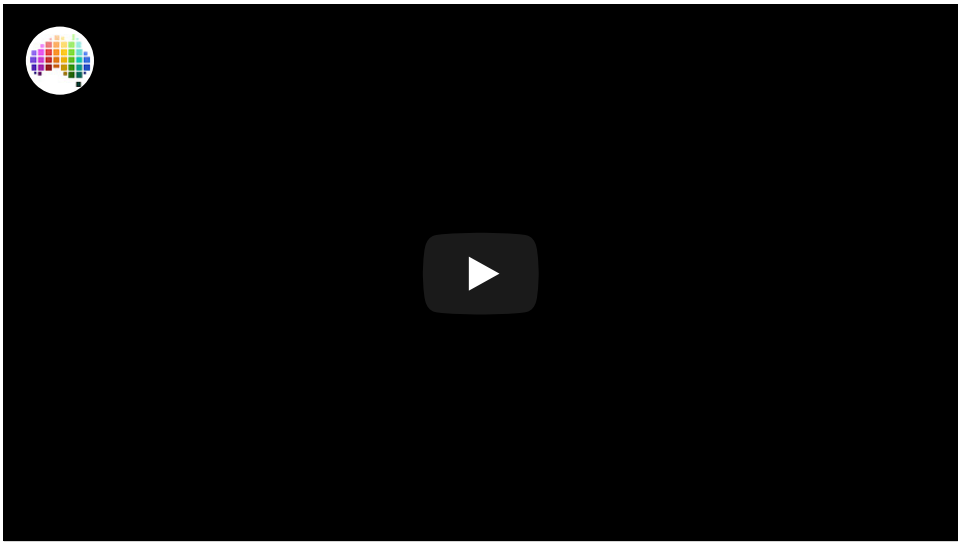
Here are the rocket instructions:

Build a Rocket Poster (https://aca.edu.au/public/posters/microbit_rocket_instructions.pdf)

Here are our designs for a rocket that you can print, or even draw your own!!

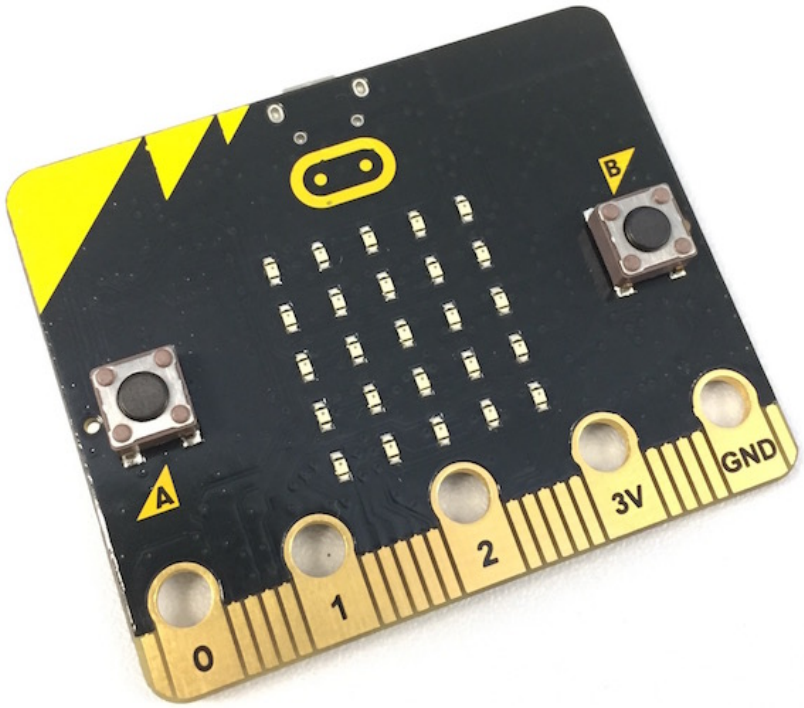
Printable Rockets (https://aca.edu.au/public/posters/spaceship_cutouts.pdf)

You too can make a rocket that runs like this...



1.1.2. Launch into micro:bit

The BBC micro:bit (https://www.microbit.co.uk/) is a tiny computer that runs the Python (https://microbit-micropython.readthedocs.io/) programming language.



The BBC micro:bit

The micro:bit has:

- **a 5 x 5 display of LEDs** (light emitting diodes)
- **two buttons** (A and B)
- **an accelerometer** (to know which way is up)
- **a magnetometer** (like a compass)
- **a temperature sensor**
- **Bluetooth** (to talk to other micro:bits and phones)
- **pins** (gold pads along the bottom) to connect to other devices like screens, motors, buttons, lights, robots and more!

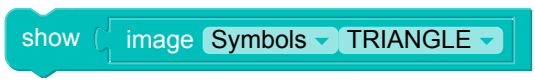
> **If you don't have a real micro:bit...**
>
> You can still do this course. It includes a full micro:bit simulator, so you'll be able to do everything you'd do on a real micro:bit!

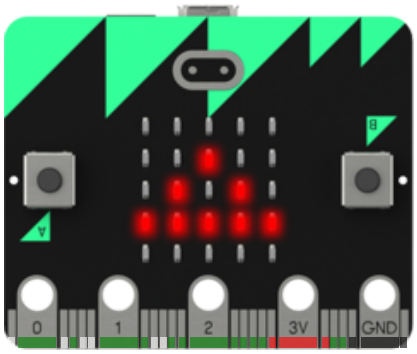
1.1.3. At first light

Let's get started by trying out a simple program!

This program turns on a pattern of *light emitting diodes* or LEDs. What does the pattern look like?

Click the run button ▶ in the example below





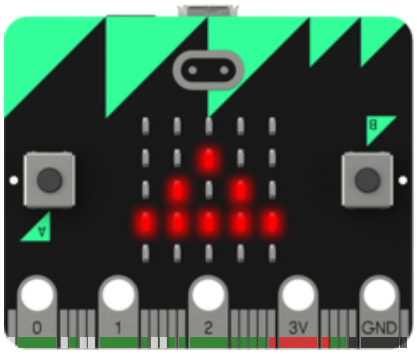
This is the closest we can get to a picture of a rocket with the images in micro:bit. But with a little more coding we can make a better picture.

Congratulations! You just programmed a micro:bit!

1.1.4. Problem: Shed light on the problem

Make your own micro:bit program to show a triangle.

1. Join the blocks together.
2. Click the Run button.
3. See a triangle like in the example below.
4. Click the Mark button.
5. Congratulations! You did it!



You'll need

program.blockly

Testing

- [ ] Testing that the display is showing a triangle.
- [ ] **Congratulations, you've written your first micro:bit program!**

1.1.5. Building blocks

You won't always have the blocks you need.

To get a new block:

1. Click the micro:bit drawer (this is the micro:bit button on the grey part of the screen).
2. Drag the block.
3. Make your program!

1.1.6. Problem: Like a diamond in the sky

This mini-challenge is all about space! One of the best known rhymes about space is "Twinkle twinkle little star."

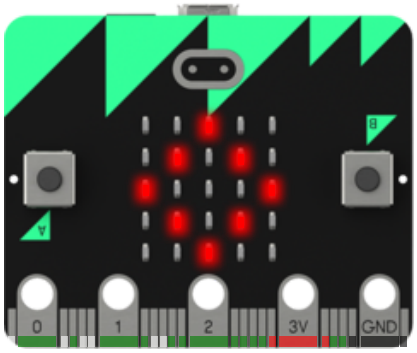
We can't easily show a star on our micro:bit, but we can show a diamond!

1. Click the micro:bit drawer.
2. Drag the show block.
3. Create the program to show a diamond.

Hint

A diamond is in the Symbols list.

4. Run and Mark your program.
5. Yay! You have twinkled your own diamond!



Testing

- [ ] Testing that the display is showing a diamond.
- [ ] **Congratulations! Look at that diamond twinkle!**

1.1.7. Downloading

If you have a real micro:bit you can download your code and run it in real life!

Click the Download button and you will get a `.hex` file.

Take the `.hex` and drag it into the micro:bit – just like it's a USB drive.

It will take a few seconds, but once it's done you'll see your program running on the micro:bit!

You can also read our blog post with more detailed instructions (https://medium.com/p/b89fbbac2552).

2

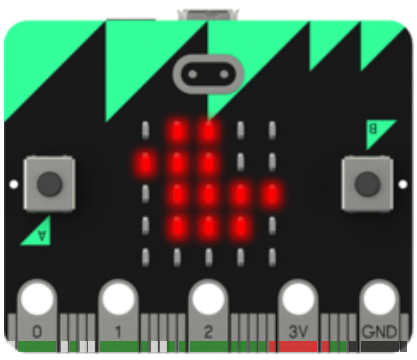
REPEATING THINGS

2.1. Off like a rocket

2.1.1. Ducks in a row

How do we show two images?

1 Click ▶ to run this example.



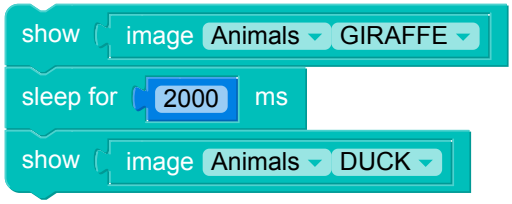
Uh oh! **We only see a duck!**

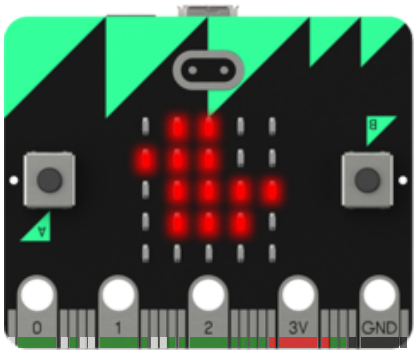
The micro:bit is *really* fast. It shows the giraffe too fast for us to see.

2.1.2. The sleep block

We can stop the micro:bit going too fast using sleep.

1. Click ▶ run the example below.
2. The giraffe appears for 2 seconds. Then the duck appears.



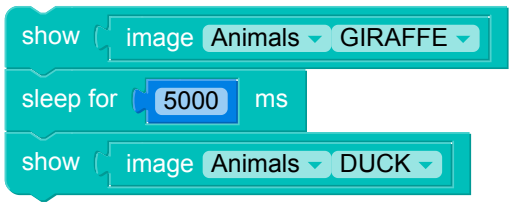


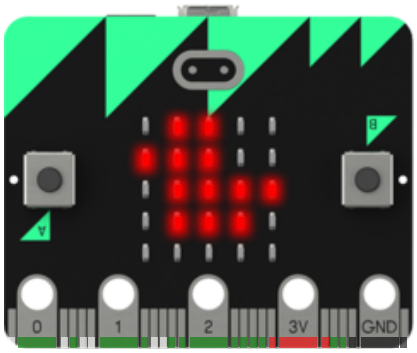
2.1.3. sleep 5000 ms

We can sleep for different lengths of time.

The example below shows the giraffe for **5 seconds**.

1. Change sleep 5000 ms to sleep 1000 ms.
2. Click ▶ run the example below. The giraffe appears for only **one second** now!


show image Animals GIRAFFE
sleep for 5000 ms
show image Animals DUCK


0
1
2
3V
GND

2.1.4. Problem: Hearts in space

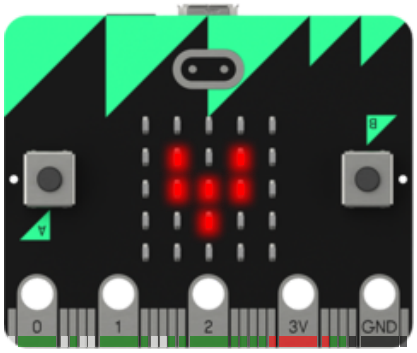
Did you know that the human heart changes size and shape in the small amount of gravity found in space? We're going to show two different sizes of heart one after the other.

Write a program that shows a heart for **one second**, followed by a small heart.

1. Use a show block to show the HEART
2. Use a sleep block to make it stay for 1 second
3. Use a show block to show the HEART_SMALL

Sleep for milliseconds

Remember: the `sleep` function takes the time in milliseconds. You can convert seconds to milliseconds by multiplying by 1000.



Here are the built-in images for you to use:

Name	Image
`HEART`	
`HEART_SMALL`	

You'll need

program.blockly

Testing

- [ ] Testing that the display starts with a big heart.
- [ ] Testing that the big heart face is still on the screen less than 1 second later.
- [ ] Testing that the display changes to a small heart after 1 second.
- [ ] **Congratulations!!**

2.1.5. Looping forever

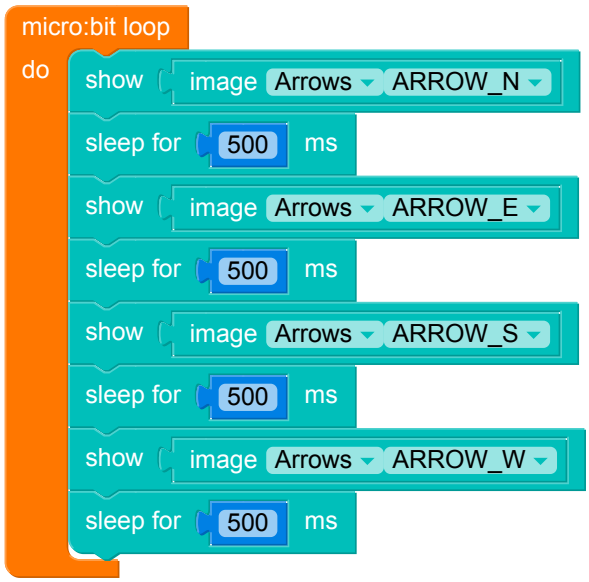
So far our programs only run each instruction once.

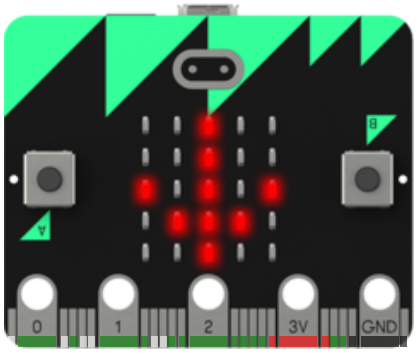
But we can use a micro:bit loop to repeat them!

1. Click ▶ to run the example below.

The arrows just keep changing forever!

2. Click the ■ button.
3. Try making the arrows change slower.





2.1.6. Problem: Twinkle twinkle little diamond

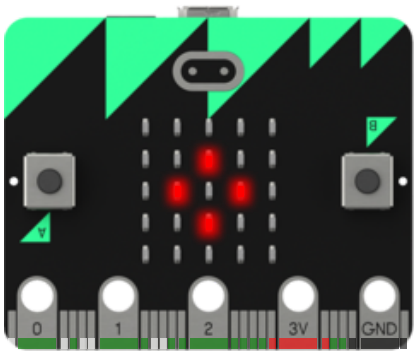
Stars twinkle because they are really really far away and the earth's atmosphere can make little changes in the path of the light.

We don't really have a star to work with so we're going to make the micro:bit DIAMOND shape twinkle.

Your program should:

1. Show the big diamond for **1 fifth of a second (200 milliseconds)**.
2. Show the small diamond for **1 fifth of a second (200 milliseconds)**.
3. Loop this forever!

It will look like this, but will *loop* forever!

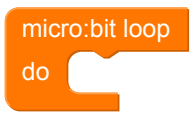


Here are the built-in images for you to use:

Name	Image
`DIAMOND`	
`DIAMOND_SMALL`	

You'll need

program.blockly



Testing

- [ ] Testing that the display starts with a big diamond.
- [ ] Testing that the display is still showing a big diamond after a fifth of a second.
- [ ] Testing that the display shows the small diamond for a fifth of a second.
- [ ] Testing that the display shows the small diamond for a fifth of a second.
- [ ] Checking that your code contains an infinite loop.
- [ ] Testing that the display goes back to the big diamond for a fifth of a second.
- [ ] Testing that the animation loops continuously.

2.1.7. Scrolling text

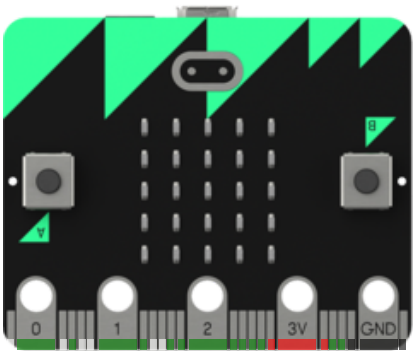
We can use scroll to show letters.

1. ▶ run the example.

Hello will scroll across the micro:bit.

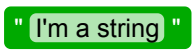
2. Click on " Hello ".
3. Type in your name.
4. ▶ run the example again.

The micro:bit scrolls your name!



A string of letters

The green block is called a **string**.

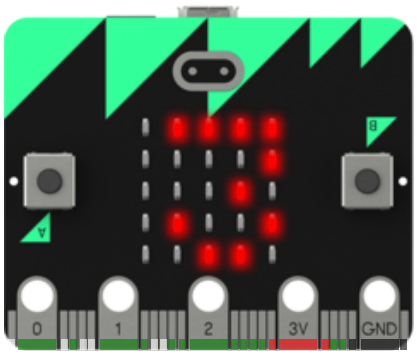


2.1.8. Problem: 3... 2... 1... GO!

Sometimes, when you're scared of doing something having a count down can make you feel better!

Make a count down timer to give you a little push when you need it. On your marks, get set, GO!... and GO!... and GO!

1. Get the micro:bit loop block.
2. Use the scroll block.
3. Use a " string " block.
4. Type the string `3 2 1 GO!`.



You'll need

program.blockly

Testing

- [ ] Checking that your code contains an infinite loop.
- [ ] Testing that the display counts down.
- [ ] Testing that it continues to work multiple times.

3

MAKING DECISIONS

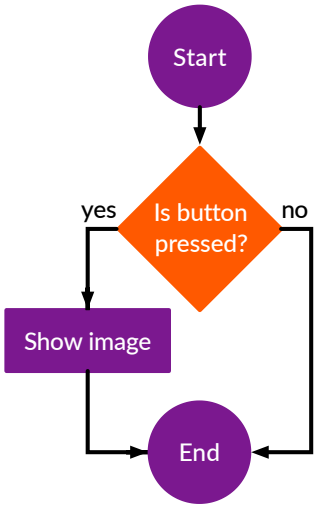
3.1. Ignition sequence

3.1.1. Making Decisions

So far our programs only *show* things. The *same program* always shows the *same thing*.

But a program can have *user input*. This means it can do *different things* in *different situations*.

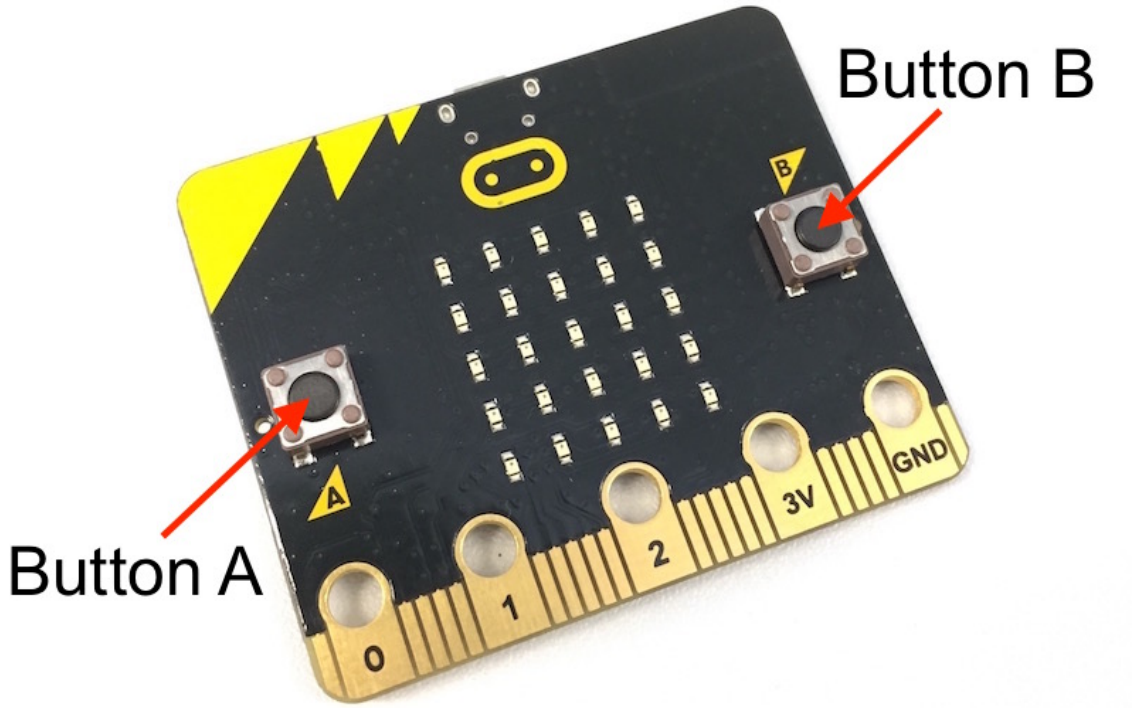
This flowchart is an example of making a decision. An image is shown *only* if the button is pressed.



3.1.2. Button A and Button B

The micro:bit has two buttons.

One is **A**. The other one is **B**.



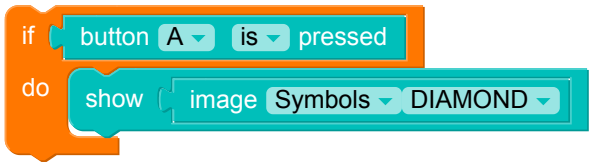
We can use button A is pressed and if to make decisions.

3.1.3. The if block

We can use the if block to make a decision.

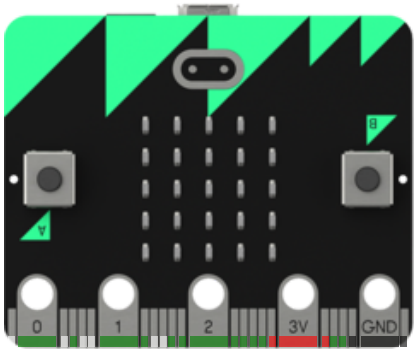
1. ▶ run the example below.
2. Press the A button on the micro:bit.

Uh oh! The program doesn't work!



Use your mouse or keyboard

Press the buttons in the examples by clicking with your mouse or pressing A or B on your keyboard.



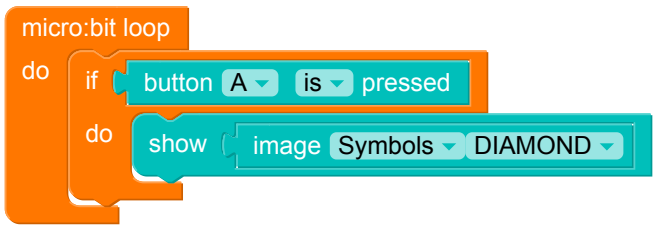
The program runs once and then stops.

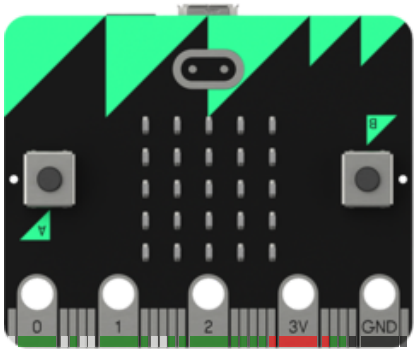
We need to run the decision many times...

3.1.4. Decisions inside a loop

We can fix this though! We put the if inside of the micro:bit loop.

1. ▶ run the example below.
2. Press the A button on the micro:bit.
3. What happens if you press A again?
4. You can click■ button.





3.1.5. Problem: Check the video feed

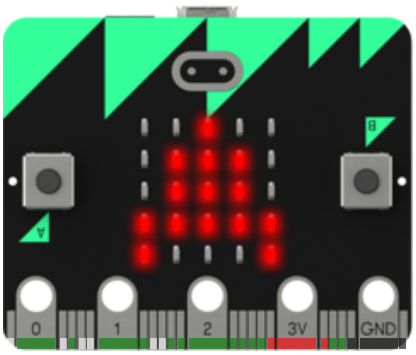
Spaceships have cameras to track what happens inside and outside of the ship. Make a program that shows the view of a rocket (using the special image `ROCKET`) or the view inside (a `HAPPY` astronaut).

Your program should:

1. If the `A` button was pressed **show a `HAPPY` face**
2. Otherwise, show a `ROCKET`.
3. Loop this forever!

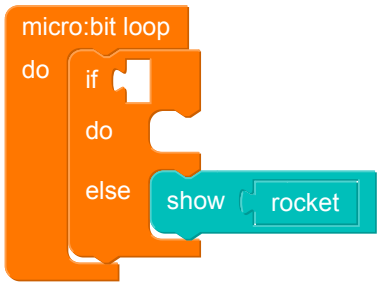
It will look like this, but will *loop* forever!

Try pressing the `A` button in this example.



You'll need

program.blockly

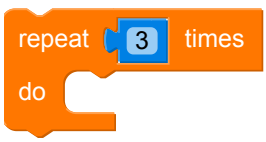


Testing

- [ ] Checking that your code contains an infinite loop.
- [ ] Testing that something happens when you press A.
- [ ] Testing that the display starts off showing a rocket.
- [ ] Testing that the display shows a happy face when the A button is pressed.
- [ ] Testing that it goes back to a rocket afterwards.
- [ ] Testing that it continues to work multiple times.

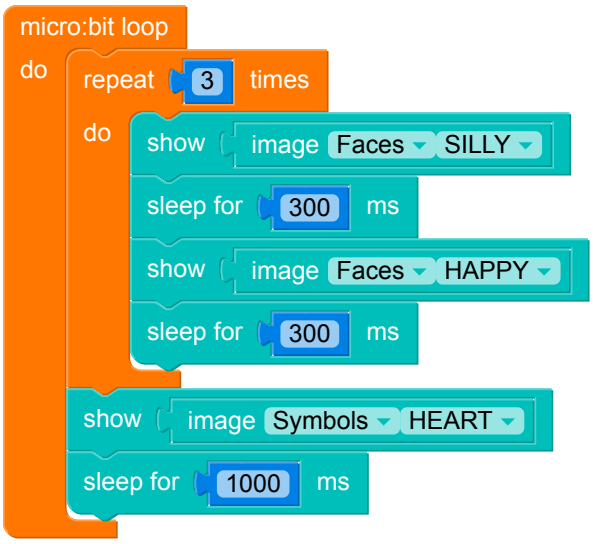
3.1.6. How (not) to repeat yourself

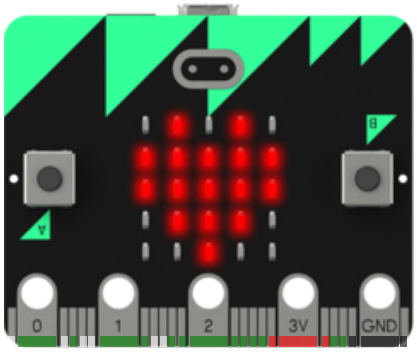
So far, we've learnt how to tell the computer to do specific things. Computers are great at doing a specific thing lots of times using the repeat block.



3.1.7. Repeated double takes!

We can use the repeat block to cycle through a pattern of images!





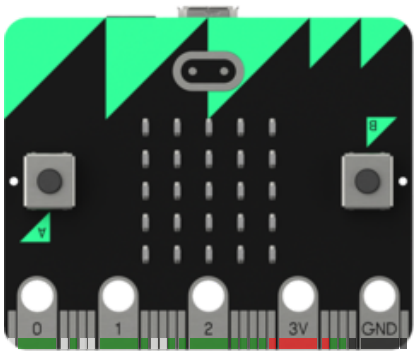
3.1.8. Problem: Keep the beat!

Did you know there's a guitar on the International Space Station, and lots of astronauts are musicians! Write a program to help keep the space-beat!

If the `A` button is pressed, your program should **repeat four times**:

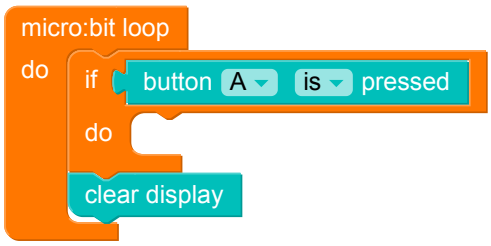
1. **show a `MUSIC_QUAVER` for 300 ms**, then
2. **show a `MUSIC_QUAVERS` for 300 ms**.
3. **After repeating that four times** the display should **clear**.

Try pressing the `A` button in this example.



You'll need

program.blockly



Testing

- [ ] Checking that your code contains an infinite loop.
- [ ] Testing that something happens when you press A.
- [ ] Testing that the display starts off blank.
- [ ] Testing that the display shows a `QUAVER` when the A button is pressed.
- [ ] Testing that the display changes to `QUAVERS` after 300ms.
- [ ] Testing that the display changes back to `QUAVER` after 300ms.
- [ ] Testing the repeated `QUAVER` and `QUAVERS`.
- [ ] Testing that it goes back to a blank screen afterwards.

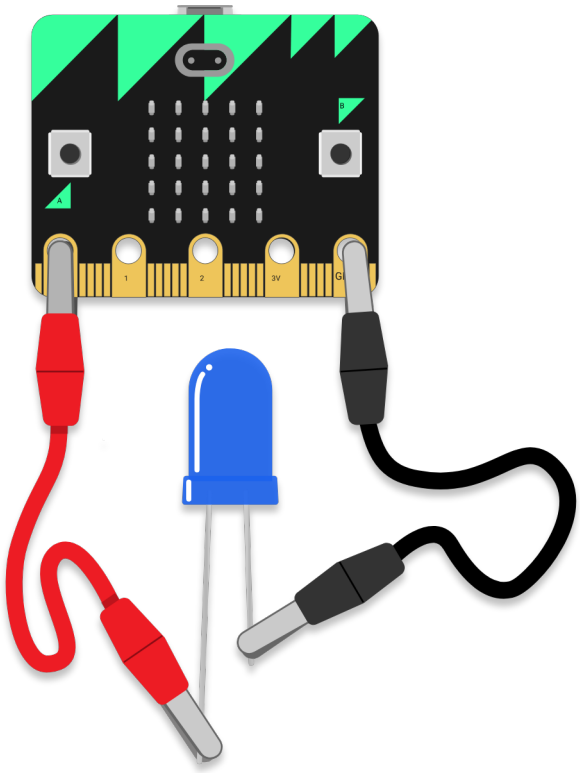
- [ ] Testing that it continues to work multiple times.

3.1.9. Hooking up extra stuff

You can use the gold holes on the micro:bit - which are referred to as external pins - to connect more LEDs and have colours other than red.

You can see in the picture below how to connect your micro:bit to an LED

The longer leg of the LED is the anode or positive leg and needs to be connected to the numbered pin. The shorter leg is the cathode or negative pin. It gets connected to the micro:bit pin labelled GND.

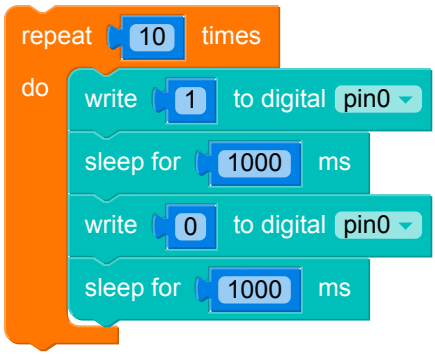


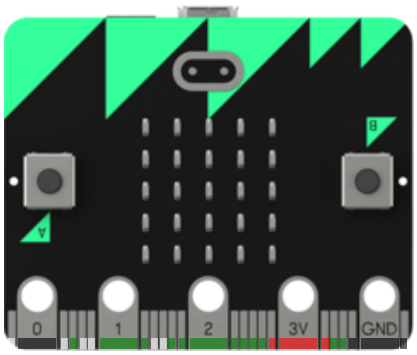
How to wire an extra LED

3.1.10. Turning your LED on and off

To turn the LED on and off you need to:

1. Connect your LED
2. Insert a `write to digital pin0` block and insert a `1` for **on**
3. Insert a `sleep` block (to make the LED stay on for some time)
4. Insert a `write to digital pin0` block and insert a `0` for **off**
5. Insert a `sleep` block (to make the LED stay off for some time)

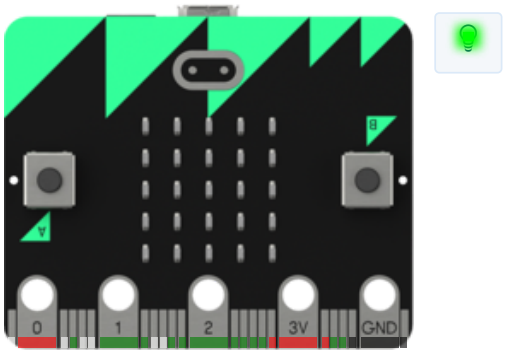

repeat 10 times
do
write 1 to digital pin0
sleep for 1000 ms
write 0 to digital pin0
sleep for 1000 ms


0
1
2
3V
GND

3.1.11. Problem: Life Detector

Your team wants a light to fit onto a life detector that blinks a green LED on and off every **half a second** and repeats forever.

1. Connect your LED
2. Insert a `write to digital pin0` block and insert a `1` for **on**
3. Insert a `sleep` block (to make the LED stay on for some time)
4. Insert a `write to digital pin0` block and insert a `0` for **off**
5. Insert a `sleep` block (to make the LED stay off for some time)



You'll need

program.blockly

components.json

Testing

- [ ] Checking that your code contains an infinite loop.
- [ ] Testing that the display starts with the green LED on.
- [ ] Testing that the green LED turns off.
- [ ] Testing that the green LED returns to on.
- [ ] **Congratulations!!**

4

BLAST OFF!

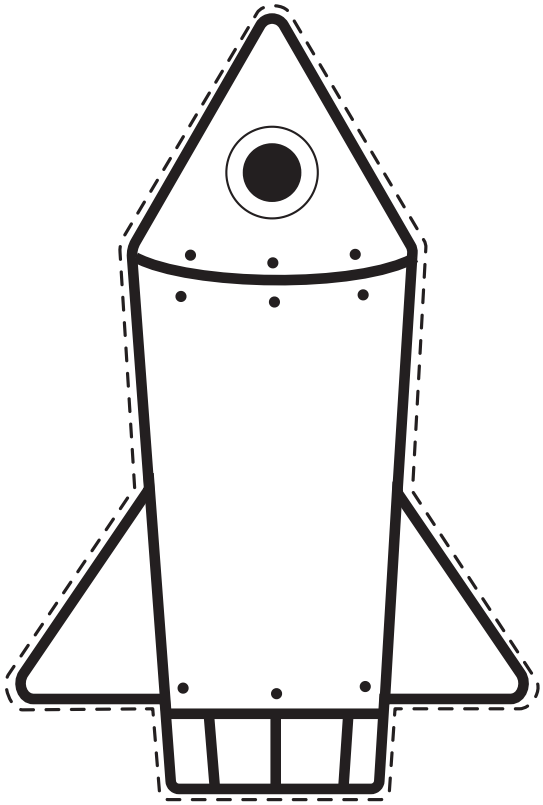
4.1. Blast Off!

4.1.1. Bring it all together.

We've now learnt how to do lots of things with the micro:bit! Nice work!

For our last project, we're going to put everything we've learnt together to make a rocket that blasts off and lights up. You can even make your own rocket out of cardboard and attach the micro:bit to it with tape to make it seem more real.

Download some rocket cut outs here (https://aca.edu.au/public/posters/spaceship_cutouts.pdf)!



An example rocket that we can light up

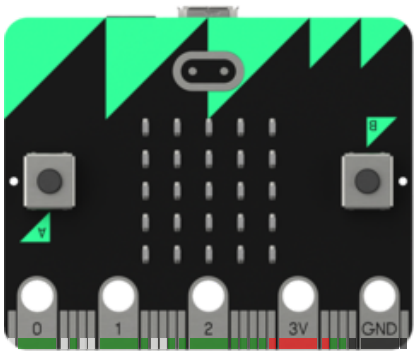
4.1.2. Problem: Countdown to launch!

On your 10... 9... 8... 7... 6... we are counting down to launch! Make the rocket Blast OFF!

Write a program to scroll `Blast OFF!` if the **A** button is pressed.

1. If the **A** button was pressed.
2. scroll `Blast OFF!`.
3. Loop forever!

Try pressing the **A** button in this example.



Testing

- [ ] Checking that your code contains an infinite loop.
- [ ] Testing that the display starts off being blank.
- [ ] Testing that the display counts down when the A button is pressed.
- [ ] Testing that it went back to a blank screen afterwards.
- [ ] Testing that it continues to work multiple times.

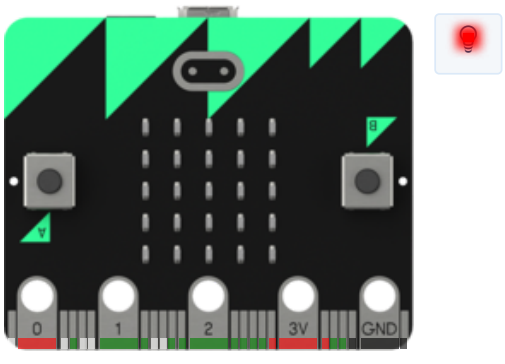
4.1.3. Problem: Fire the rockets!

We're going to need some jet engines to be lit up.

1. Turn the red LED on with write 1 to digital pin0 block
2. sleep for a very short time - **50 ms**
3. Turn the red LED off with write 0 to digital pin0 block
4. sleep for a very short time - **50 ms**

Repeat **100 times**

See our example.



You'll need

program.blockly

components.json

Testing

- [ ] Testing that the display starts with the red LED on.
- [ ] Testing that the red LED turns off.
- [ ] Testing that the red LED returns to on.
- [ ] Testing that the red LED flashes 100 times.
- [ ] Testing that the red LED stops flashing after 100 flashes.
- [ ] **Congratulations!**

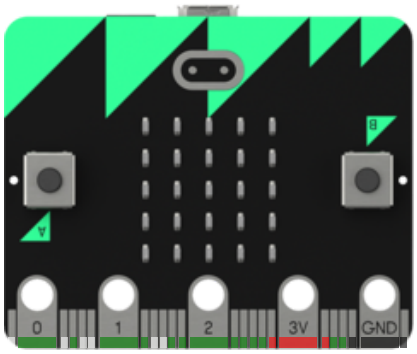
4.1.4. Problem: Final Project Blast Off!

Put everything together now. When the A button is pressed your rocket should scroll the words " Blast OFF! " on the screen and then flash the led on pin 0 on and off repeated 100 times.

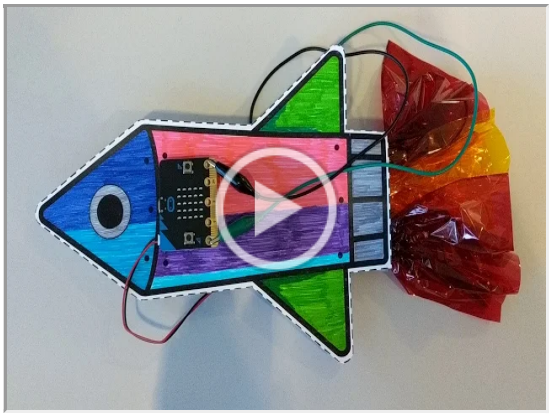
You can turn these on and off with the write 1 to digital pin0 block.

Your LED should flicker on and off repeating **100 times**, with a very small sleep of 50 ms in between.

Run our example. Don't forget to **press button A**!



Follow these instructions (https://aca.edu.au/public/posters/microbit_rocket_instructions.pdf) to make your own project like ours!



You'll need

program.blockly

Testing

- [ ] Checking your code contains an infinite loop.
- [ ] Testing that the display starts off being blank.
- [ ] Testing that the display counts down when the A button is pressed.
- [ ] Testing that the the red LED turns on after the scroll.
- [ ] Testing that the red LED flashes 100 times.
- [ ] **Congratulations!!**

4.1.5. Problem: Blockly micro:bit Playground

This is a micro:bit playground question! Use the blocks to build anything you like!

You'll need

components.json

Testing

- [ ] Nothing to mark!