Girls' Programming Network

# *Tamagotchi with micro:bits!*

***Tutors Only***

# This project was created by GPN Australia for GPN sites all around Australia!

**This workbook and related materials were created by tutors at:**

Sydney, Melbourne and Perth

Girls' Programming Network

***If you see any of the following tutors, don't forget to thank them!!***

| Writers | Testers |
|---|---|
| Alison Wong | Emily Hadzajlic |
| Deanna Arora | Sheree Pudney |
| Janette Chiu | |
| Julia Wong | |
| Angela Sojan | |
| Emily Hadzajlic | |
| Maya Schildkraut | |
| Saanvi Telukoti | |
| Veronika Sevastyanova | |
| Karen Garcia | |
| Isabella Hogan | |

Intro to Micro:Bit

# Part 0: Setting Up

## Task 0.1: micro:bits and pieces

**Let's set up the micro:bit for programming today! You should have:**

- **1 micro:bit chip**
- **1 USB cable**

1. Connect the small end of the USB cord to the middle port of the micro:bit
2. Connect the big end of the cord to your computer
3. Go to **python.microbit.org**

## Task 0.2: Micro playground

First, we're going to play around with the displays on **python.microbit.org** and test them on our micro:bits.

1. Make sure `from microbit import *` is at the top of your code.
2. Change the code under the `while True:` loop to display a DUCK and scroll your name instead using `display.show(Image.DUCK)`, `sleep(1000)` and `display.scroll("Your name")`
3. Click the **'Send to micro:bit'** button, then follow the steps on the screen.
4. Try again with other words and pictures.

### *Hint: Cheat sheets*

Don't forget you have cheat sheets on the web page to help you code!
Remember to indent the code below the while loop!

## ☑ CHECKPOINT ☑

**If you can tick all of these off, you can go to Part 1:**

☐ You have connected your micro:bit to the computer

☐ You can display different pictures and words

## Today's Project Plan - Tamagotchi

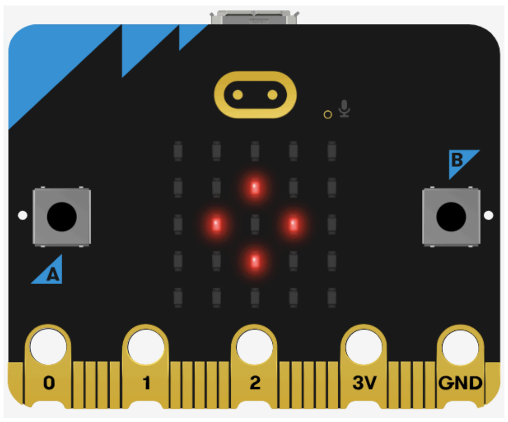
**We're going to make a Tamagotchi electronic pet! By pressing buttons and moving the micro:bit you can interact with the pet to keep it happy and alive.**

1. Start off by welcoming the user and showing an initial image of your pet
2. Setting up the main values related to your pet
3. Show the pet's name, hunger, boredom and sleepiness.
4. Add the ability to feed, play with, or let the pet to sleep.
5. Make it so time passes
6. Make the ability for the pet to die, but keep it alive!
7. Watch your pet grow up.

**+ more**

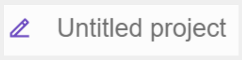
**Once your base pet works, add cool extensions!**

# Part 1: Welcome to Tamagotchi!



## Task 1.1: Name your file!

Now you've been introduced to your micro:bit, let's start working on the project!

1. At the top of the page, click on  Untitled project and edit your project name to be 'tamagotchi'.
2. At the top of your code, add a new line (above the import line) and use a comment to write your name

### *Hint: Comments*

Remember comments start with a **#**

```
# Comments don't actually do anything - they are just notes!
```

## Task 1.2: Welcoming the user

To show that the game is starting, let's welcome the user to your Tamagotchi game!

1. In a new line before the `while True`, use `display.scroll()` to show the text `'Welcome to Tamagotchi'`.

### *Hint: Display text*

Here is an example of how to `display` text that scrolls.
Make sure you put your text in quotes!

```
display.scroll('Hello World')
```

## Task 1.3: Baby Tamagotchi

Now it’s time to add our Tamagotchi pet image.

Let’s show a baby Tamagotchi image (small diamond) which looks like an egg!

1. Under our welcome message, before the `while True`, use `display.show()` to show a small diamond (called `Image.DIAMOND_SMALL)`.
2. Then we need to `sleep` for 1 second.
3. Put a # in front of the `while True` line and all the lines after it as you don’t need it at the moment (this is called ‘commenting out’ code)

We’ll display different images as our Tamagotchi grows up.

### *Hint: Show an image*

Here is an example of how to `show` an image:
`display.show(Image.SILLY)`

### *Hint: How to sleep?*

Remember our `sleep` function works in milliseconds (ms)
1 second = `1000` ms
10 seconds = `10000` ms
To wait for 5 seconds we’d use `sleep(5000)`

### *Hint: Commenting out code*

If you don’t need some of your code at the moment, but don’t want to delete it because you might need it in the future, **‘comment it out’** by putting a # in front

```
# while True:
#     display.show(Image.HEART)
#     sleep(1000)
#     display.scroll('Hello')
```

## Task 1.4: Try it out!

Let’s test your code:

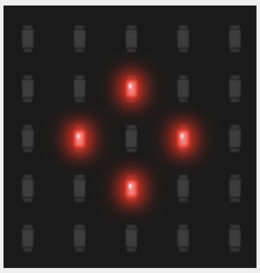
1. Click the play button on your simulator to check your welcome message and image are displaying correctly
2. Click the **'Send to micro:bit'** button to run it on your micro:bit

## CHECKPOINT ☑

**If you can tick all of these off you can go to Part 2:**

☐ Your program shows a scrolling welcome message.

☐ Your program shows a baby Tamagotchi (small diamond).



☐ You tried it on your real life micro:bit.

## TUTOR TIPS

The code should look like this (including bonuses):

```
# Isabella
from microbit import *

display.scroll("Welcome to Tamagotchi")
display.show(Image.DIAMOND_SMALL)
sleep(1000)

# while True:
#     display.show(Image.DUCK)
#     sleep(1000)
#     display.scroll("Izy")
```

# Part 2: Seeing our pet

Variables

## Task 2.1: Name your Tamagotchi!

**Let's give our Tamagotchi a name**

1. Create a variable called **`name`** underneath where you were showing the image
2. Choose a name for your Tamagotchi and store it in the variable **`name`**.

### *Hint: String variables*

Remember that a variable storing a string looks like this (don't forget the quotes!):

```
    dessert = "cake"
```

## Task 2.2: Store the state of your Tamagotchi

**We want to track the hunger, boredom and sleepiness of our Tamagotchi.** We will store these in variables and start them at 0.

1. Create 3 variables called **`hunger`**, **`boredom`** and **`sleepiness`** under where you created **`name`**
2. Set all the 3 variables to have a value of 0.

### *Hint: Integer variables*

Remember that a variable storing a number looks like this:

```
  fave_number = 3
```

## Task 2.3: Showing the name

Let's display the pet's name.

1. Under where we set the sleepiness variable to 0, use **`display.scroll()`** and the variable **`name`** to display the name of your pet.
2. Then we need to **`sleep`** for 1 second.

### *Hint: Show the value of a variable*

Here is an example of how to we use **`display.scroll`** to display the value of a variable called **`fav_icecream`**:

```
display.scroll(fav_icecream)
```

## Task 2.4: How hungry, bored, and sleepy is our pet?

We want to display each of the other variables after the name is displayed:

1. Use **display.scroll()** to scroll the text `"Hunger: "` followed by the **hunger** variable and then wait 50 milliseconds
2. Use **display.scroll()** to scroll the text `"Boredom: "` followed by the **boredom** variable then wait 50 milliseconds
3. Use **display.scroll()** to scroll the text `"Sleepiness:"` and then the contents of our **sleepiness** variable then wait 50 milliseconds
4. That's a lot of text to scroll across! Add a **`delay`** to each **display.scroll()** with a value of 100 to make it scroll faster.

### *Hint: Concatenating*

To display some text and the value of a variable, we can put them together (concatenate) like this.
The computer gets confused when we try to add a string and a number together, so we can turn a number into a string using **`str()`**.

```
display.scroll("Hunger: " + str(hunger))
```

### *Hint: Make it scroll faster*

You can set the speed of the text scrolling with **`delay`**.
The lower the number, the faster it goes. 150 is the speed if you don't use a delay.

```
display.scroll("Hello World” ,delay = 100)
```

## Task 2.5: Try it out!

Let's test your code:

1. Click the play button on your simulator to check your welcome message and image are displaying correctly

2. Click the **'Send to micro:bit'** button to run it on your micro:bit

## ☑ CHECKPOINT ☑

**If you can tick all of these off you can go to Part 3:**

☐ Your program stores the **`name`** of your Tamagotchi.

☐ Your program stores the **`hunger`**, **`boredom`** and **`sleepiness`** values of your Tamagotchi. They should all be **`0`**.

☐ Your program displays the name and an image for your pet

☐ It then scrolls to show the hunger, boredom, and sleepiness

## TUTOR TIPS

The code should look like this (with lines that are commented out):

```
# Isabella
from microbit import *

display.scroll("Welcome to Tamagotchi")
display.show(Image.DIAMOND_SMALL)
sleep(1000)

name = "Tammy"
hunger = 0
boredom = 0
sleepiness = 0

display.scroll(name)
sleep(1000)
display.scroll("Hunger: "+str(hunger), delay = 100)
sleep(50)
display.scroll("Boredom: "+str(boredom), delay = 100)
sleep(50)
display.scroll("Sleepiness: "+str(sleepiness), delay = 100)
sleep(50)

# while True:
```

```
#     display.show(Image.DUCK)
#     sleep(1000)
#     display.scroll("Izy")
```

Micro:bit Inputs

# Part 3: Shake it and see

## Task 3.1: Loop, loop, loop …

We're ready to start making a loop so the micro:bit will continually check for input and perform actions.

Let's activate our `while` loop

1. Remove the # in front of the `while True` line
2. Delete all the lines under the `while True` line

## Task 3.2: Shake it like a polaroid picture

Now we want to have the name, image, hunger, boredom and sleepiness information scroll across the display ONLY if we shake the micro:bit.

The micro:bit has an inbuilt accelerometer which can detect if it has been moved.

1. Under the line that says `while True`, write an if statement to check if the **`accelerometer`** was **`shaken`** (make sure you indent it under the `while)`
2. Move your lines of code that display the DIAMOND_SMALL image for one second under the `if` statement
3. Move your lines of code that display **`name, hunger, boredom and sleepiness`** (and their sleep lines) under the `if` statement so they display after the image
4. Indent all of the display lines you've just moved under the `if` statement so that they only happen if the `if` statement is `True`.

### *Hint: Using the Accelerometer*

This is how you can test if the accelerometer has been shaken like this:

```
if accelerometer.was_gesture("shake"):
  display.scroll("Shake!")
```

Check out the cheat sheets on the web page to find out what other movements the accelerometer can detect!

## *Hint: Moving Code & Indenting*

You can more multiple lines of code by using **CTL X** and **CTL V** to cut and paste them.
You can indent multiple lines of code by selecting them and using Tab ->
Ask your tutor for help if you haven't used the shortcuts before.

## Task 3.5: Try it out!

Let's test your code:

1. Click the play button on your simulator to check your welcome message and image are displaying correctly
2. Click the **'Send to micro:bit'** button to run it on your micro:bit

## ★ BONUS 3.6: Making your own images!

We're currently using an inbuilt image to represent our pet, maybe you want to make something more your style!

Use a different image OR create and display your own image

Find more inbuilt images and instructions for making your own image see:
https://microbit-micropython.readthedocs.io/en/latest/tutorials/images.html

Code for making your own image (pixel by pixel) looks like this:

```
gpn_heart = Image("09090:33903:90009:03030:00900")
display.show(gpn_heart)
```

## ☑ CHECKPOINT ☑

**If you can tick all of these off, you can go to Part 4:**

☐ When your Micro:Bit is shaken, it displays the image
☐ It then displays the name of your pet
☐ It then scrolls to show the hunger, boredom, and sleepiness

## TUTOR TIPS

The code should look like this:

Tutors - make sure students move the sleep() lines with the display lines

```
# Isabella
from microbit import *

display.scroll("Welcome to Tamagotchi")

name = "Tammy"
hunger = 0
boredom = 0
sleepiness = 0

while True:
    if accelerometer.was_gesture("shake"):
        display.show(Image.DIAMOND_SMALL)
        sleep(1000)
        display.scroll(name)
        sleep(1000)
        display.scroll("Hunger: "+str(hunger), delay = 100)
        sleep(50)
        display.scroll("Boredom: "+str(boredom), delay = 100)
        sleep(50)
        display.scroll("Sleepiness: "+str(sleepiness), delay = 100)
        sleep(50)
```

# Part 4: Interact with the Tamagotchi

Now that we are displaying and tracking our Tamagotchi's hunger, boredom and sleepiness, we want to keep it fed, entertained and rested!

## Task 4.1: Feed the Tamagotchi!

We're going to feed the Tamagotchi every time we press the **A button**.

1. Create a new `if` statement within our `while` loop after the `display.scroll` lines. Make sure it's indented at the same level as the `if` statement that checks for a shake.

2. Check if the **A** button was pressed.

3. Decrease the hunger variable by 3.

### *Hint: Check if a button has been pressed!*

We can check if micro:bit button a has been pressed like this:

```
if button_a.was_pressed():
  # do something here, don't forget the indent
```

### *Hint: Changing variables!*

We can decrease a variable by 2 like this:

```
my_number = my_number - 2
```

## Task 4.2:  Play with Tamagotchi!

Now that we can feed the Tamagotchi, we also want to keep it happy!

We want to play with the Tamagotchi using the **B button**.

1. Check if the **B** button was pressed

2. Decrease the `boredom` variable by 3

## Task 4.3: Getting some Zzzs!

Now that we can feed and play with the tamagotchi, let's keep it rested.

We want to make the tamagotchi sleep when we turn the microbit upside down.

1. Check the accelerometer to see if the microbit was `"face down"`
2. Decrease the **`sleepiness`** variable by 3

## *Hint: Using the Accelerometer*

Before you tested if the accelerometer detected a `"shake"`, now you have to test if it's been turned `"face down"`.

## Task 4.4 Try it out!

Test your code:

1. Click the **'Send to micro:bit'** button to run it on your micro:bit
2. Use the micro:bit to feed, play with and give your Tamagotchi some rest
3. Give your Micro:bit a shake to see its hunger, boredom and sleepiness scores - these should now be less than zero.

## ☑ CHECKPOINT ☑

**If you can tick all of these off, you can go to Part 5:**

☐ Your program decreases the hunger value by 3 when the A button is pressed.

☐ Your program decreases the boredom value by 3 when the B button is pressed.

☐ Your program decreases the sleepiness value by 3 when it's flipped over.

☐ You tried it on your real-life micro:bit

## TUTOR TIPS

The code should look like this:

```
# Isabella
from microbit import *

display.scroll("Welcome to Tamagotchi")

name = "Tammy"
hunger = 0
boredom = 0
sleepiness = 0
while True:
    if accelerometer.was_gesture("shake"):
        display.show(Image.DIAMOND_SMALL)
        sleep(1000)
        display.scroll(name)
        sleep(1000)
        display.scroll("Hunger: "+str(hunger, delay = 100))
        sleep(50)
        display.scroll("Boredom: "+str(boredom), delay = 100)
        sleep(50)
        display.scroll("Sleepiness: "+str(sleepiness), delay = 100)
        sleep(50)
    if button_a.was_pressed():
        hunger = hunger - 3
    if button_b.was_pressed():
        boredom = boredom - 3
    if accelerometer.was_gesture("face down"):
        sleepiness = sleepiness - 3
```

# Part 5: Time management

Let's make it so that our pet automatically gets more hungry, sleepy and bored as time goes by. Then it will be up to you to play with it, feed it and make it rest to keep it alive!

## Task 5.1: The Beginning of Time

Time to add ...time!

We can ask the micro:bit how long the game has been running using `running_time()`

1. Above your `while` loop, create a variable called `start_time` and set it to `running_time()`

## Task 5.2: How long has gone by?

Every 10 seconds, our Tamagotchi gets more hungry, sleepy, and bored. To do this, we can use `running_time()` which tells us how much time has passed since we started our code.

1. At the bottom of your code, inside the `while` loop, create a variable called `now_time` and set it to `running_time()`
2. Create a variable called `time_passed` and set it to `now_time` minus `start_time`.
3. Add an `if` statement to check if `time_passed` is more than 10 seconds (remember we're using milliseconds).
4. If `time_passed` is more than 10 seconds, increase `hunger`, `boredom`, and `sleepiness` by 1.

## Task 5.2: Starting again!

Your Tamagotchi just got a bit more hungry, bored and sleepy! Let's start a fresh countdown by resetting `start_time` so it can happen again—and again!

1. Inside your `if` statement (after you increase `hunger`, `boredom`, and `sleepiness` by 1) reset the `start_time` variable to `running_time()` the same way you did in Task 5.1

***Hint: Comparing time…***

- **`running_time()`** is the time in milliseconds since the game started
- We use **`running_time()`** to set **`start_time`** and **`now_time`** in different parts of the code.
- **`now_time`** minus **`start_time`** is the number of milliseconds that has passed between these two times

If **`start_time = 5000`** and **`now_time = 20000`**, then 20000 - 5000 = 15000 ( so 15 seconds have passed between the two times)

## Task 5.3 Try it out!

Test your code:

1. Click the **'Send to micro:bit'** button to run it on your micro:bit
2. Don't feed, play with or give your Tamagotchi any rest
3. After 10 seconds, give your Micro:bit a shake to see its hunger, boredom and sleepiness scores - these should all now have got higher.
4. Leave it for another 10 seconds, give your Micro:bit a shake to see its hunger, boredom and sleepiness scores - these should all now have got higher again!

## ★ BONUS 5.4: Bending Time!

How could you make your Tamagotchi get hungry, bored and sleepy faster (or slower)?

## ☑ CHECKPOINT ☑

**If you can tick all of these off, you can go to Part 6:**

☐ Your program keeps track of time!

☐ Your Tamagotchi gets hungrier, sleepier, and more bored every 10 seconds.

☐ You tried it on your real-life micro:bit.

## TUTOR TIPS

The code should look like this:

Tutors:

If pet dies quickly, make sure they reset start_time to running_time() at bottom of if statement that checks if 10 seconds has passed

```
# Isabella
from microbit import *

display.scroll("Welcome to Tamagotchi")
# display.show(Image.DIAMOND_SMALL)
name = "Tammy"
hunger = 0
boredom = 0
sleepiness = 0
start_time = running_time()
while True:
    if accelerometer.was_gesture("shake"):
        display.scroll(name)
        display.show(Image.DIAMOND_SMALL)
        sleep(1000)
        display.scroll("Hunger: "+str(hunger), delay = 100)
        sleep(50)
        display.scroll("Boredom: "+str(boredom), delay = 100)
        sleep(50)
        display.scroll("Sleepiness: "+str(sleepiness), delay = 100)
        sleep(50)
    if button_a.was_pressed():
        hunger = hunger - 3
    if button_b.was_pressed():
        boredom = boredom - 3
    if accelerometer.was_gesture("face down"):
        sleepiness = sleepiness - 3
    now_time = running_time()
    time_passed = now_time - start_time
    if time_passed > 10000:
        hunger = hunger + 1
        boredom = boredom + 1
        sleepiness = sleepiness + 1
        start_time = running_time()
```

# Part 6: Making the ability to die

## Task 6.1: Dying if the values are too high

If our Tamagotchi gets too hungry, sleepy or bored it will die!

1. Inside the `while` loop, at the bottom after the code you just wrote in Section 5, create an `if` statement to check if `hunger, boredom or sleepiness` is greater than 10.
2. If any of the variables are greater than 10, scroll the message '`Dead`', display the GHOST image and use `break` to get out of the `while` loop.

## *Hint: Checking for more than one condition*

You can check for more than one condition at the same time using `or` like this:

```
if weather == "Raining” or temperature < 10:
  display.scroll("Take an umbrella”)
```

## *Hint: Breakout!*

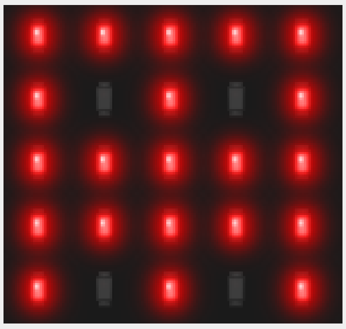
You can `break` out of `while` like this l

```
while True:
     if number > 20:
           display.scroll("Im breaking out”)
           break
```

## Task 6.2  Try it out!

Test your code:

1. Click the **'Send to micro:bit'** button to run it on your micro:bit
2. Check that your Tamagothchi dies if `hunger, boredom or sleepiness` goes above 10
3. Check that if your Tamagothchi dies, 'Dead' and the ghost image is displayed



## ☑ CHECKPOINT ☑

**If you can tick all of these off you can go to Part 7:**

- ☐ The pet dies if any of the tamagotchis variables get over 10
- ☐ You tried it on your real life micro:bit.

## TUTOR TIPS

The code should look like this:

```
# Isabella
from microbit import *

display.scroll("Welcome to Tamagotchi")

name = "Tammy"
hunger = 0
boredom = 0
sleepiness = 0

start_time = running_time()

while True:
    if accelerometer.was_gesture("shake"):
        display.show(Image.DIAMOND_SMALL)
        sleep(1000)
        display.scroll(name)
        sleep(1000)
        display.scroll("Hunger: "+str(hunger), delay = 100)
        sleep(50)
        display.scroll("Boredom: "+str(boredom), delay = 100)
        sleep(50)
        display.scroll("Sleepiness: "+str(sleepiness), delay = 100)
        sleep(50)
    if button_a.was_pressed():
        hunger = hunger - 3
    if button_b.was_pressed():
        boredom = boredom - 3
    if accelerometer.was_gesture("face down"):
        sleepiness = sleepiness - 3
    now_time = running_time()
    time_passed = now_time - start_time
    if time_passed > 10000:
        hunger = hunger + 1
        boredom = boredom + 1
        sleepiness = sleepiness + 1
        start_time = running_time()
        if hunger > 10 or boredom > 10 or sleepiness > 10:
              display.scroll("Dead")
              display.show(Image.GHOST)
              break
```

# Part 7: We're growing up, up, up!

## Task 7.1: How old is our pet?

We need to start storing the age of our pet so we can know how long we've kept it alive for!

1. In the section where you define all of the variables, add another variable called `age` and set it to 0.
2. Inside the `while` loop where you increase `hunger, boredom and sleepiness`, increase age by 1 as well (so our tamagotchi gets older!)

## Task 7.2 Display the age

Add the age to the information that is displayed when the micro:bit is shaken.

1. Find your code where you use `display.scroll` to print `hunger, boredom and sleepiness`
2. Add another `display.scroll` to print age.

## Task 7.3: Showing its age

Right now we only display a simple Tamagotchi image (a small diamond).

As our Tamagothchi gets older we want to change the image so it resembles the lifecycle of a butterfly (from egg to caterpillar to butterfly).

We'll use the **SMALL_DIAMOND** image for the egg, the **SNAKE** image for the caterpillar and the **BUTTERFLY** image for the butterfly.

1. Inside the `while` loop, where we're currently displaying a small diamond, make an `if` statement.
2. If `age` <= 3, it's a baby, so show the SMALL DIAMOND image (looks like an egg!)
3. Else if `age` <= 5, it's a teenager, so show a SNAKE image (looks like a caterpillar)
4. Else, it's an adult, so show a BUTTERFLY image

Don't forget to indent your `display.show` lines under your if, elif and else!

***Hint: if, elif and else statements***

Here is an example of `if`, `elif` and `else` statements:

```
if legs == 0:
      display.show(Image.SNAKE)
elif legs <= 2:
      display.show(Image.DUCK)
else:
      display.show(Image.GIRAFFE)
```

## Task 7.4 Try it out!

Test your code:

1. Click the **'Send to micro:bit'** button to run it on your micro:bit
2. Check you get the right image for your Tamagothchi's age when you shake the micro:bit

## ★ BONUS 7.5: Want to be different?

Select different images for the lifecycle of your pet

**Find more images on the *micro:bit Image Cheat Sheet***:
https://microbit-micropython.readthedocs.io/en/latest/tutorials/images.html

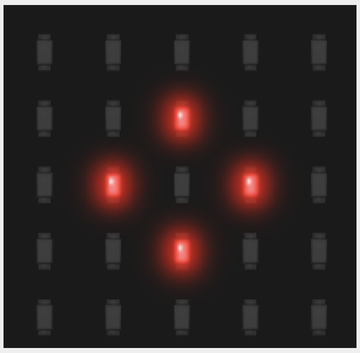
You can draw your own images if you want to - have a look back at **Bonus 3.6 Making your own images**

## ☑ CHECKPOINT ☑

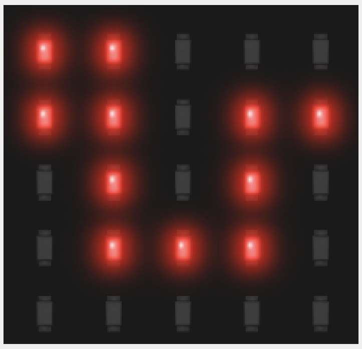
**If you can tick all of these off you can go to Part 8:**

When you shake the micro:bit:

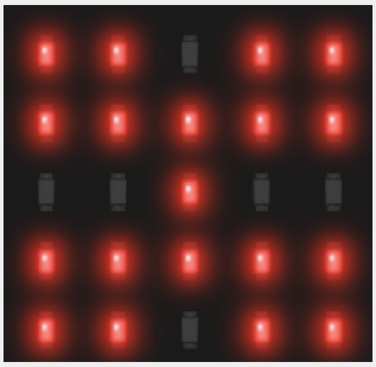
☐ If age is less than or equal to 3, our program shows a baby Tamagotchi (small diamond aka egg).



☐ Else if age is less than or equal to 5, our program shows a teenage Tamagotchi (snake aka caterpillar).



☐ Else our program shows an adult Tamagotchi! (butterfly).



☐ Age displays with the other information.

☐ You tried it on your real life micro:bit.

**TUTOR TIPS**

The code should look like this (with bonuses):

```
# Isabella
from microbit import *

display.scroll("Welcome to Tamagotchi")

name = "Tammy"
hunger = 0
boredom = 0
sleepiness = 0
age = 0

start_time = running_time()

while True:
    if accelerometer.was_gesture("shake"):
            if age <= 3:
            display.show(Image.DIAMOND_SMALL)
        elif age <=5:
            display.show(Image.SNAKE)
        else:
            display.show(Image.BUTTERFLY)
        sleep(1000)
        display.scroll(name)
        sleep(1000)
        display.scroll("Hunger: "+str(hunger), delay = 100)
        sleep(50)
        display.scroll("Boredom: "+str(boredom), delay = 100)
        sleep(50)
        display.scroll("Sleepiness: "+str(sleepiness), delay = 100)
        sleep(50)
        display.scroll("Age: "+str(age), delay = 100)
        sleep(50)
    if button_a.was_pressed():
        hunger = hunger - 3
    if button_b.was_pressed():
        boredom = boredom - 3
    if accelerometer.was_gesture("face down"):
        sleepiness = sleepiness - 3
    now_time = running_time()
    time_passed = now_time - start_time
    if time_passed > 10000:
        hunger = hunger + 1
        boredom = boredom + 1
        sleepiness = sleepiness + 1
         age = age + 1
         start_time = running_time()
         if hunger > 10 or boredom > 10 or sleepiness > 10:
            display.scroll("Dead")
            display.show(Image.GHOST)
            break
```

# Extension 8: More images

We can already see our Tamagotchi grow into a beautiful butterfly. Let's add more images to see if our Tamagotchi is sad, bored, angry or happy.

## Task 8.1: Don't make that face!

1. After we display an image showing if our Tamagochi is an egg, caterpillar or butterfly, we want to add an image to show its mood. We will start with hunger.
2. Check if `hunger >= 5`. If it is, then display the image: `Image.SAD`
3. Display the image for 1000 milliseconds using `sleep(1000)`.

## Task 8.2: Express yourself!

Our Tamagotchi would also be moody or angry if they are bored or sleepy. Now we will check and add images for these conditions!

1. If `boredom >= 5`, display image `Image:MEH.` Display the image for 1000 milliseconds using `sleep(1000)`.
2. If `sleepiness >= 5`, display image `Image:ANGRY.` Display the image for 1000 milliseconds using `sleep(1000)`.

### *Hint: Use sleep between images*

To see all these images one after the other, make sure to use `sleep()` after displaying a new image. If you don't the images will disappear super quick and you'll only be able to see the last image.

## Task 8.3: Doing fabulous!

We don't just want to see if our Tamagotchi is unhappy, we also want to see if our Tamagotchi is happy!

1. If our Tamagotchi is neither hungry, bored or sleepy, they must be happy. So we'll check if `hunger <= 5 and sleepiness <= 5 and boredom <= 5`
2. If this is `True`, we want to display a happy face. (`Image.HAPPY`)

## Task 8.4 Try it out!

Test your code:

1. Click the **'Send to micro:bit'** button to run it on your micro:bit
2. Do you get the right image depending on your pet's hunger, boredom and sleepiness values?

## ★ BONUS 8.5: Add your own images!

Instead of showing the images above, we can also choose our own images!

- Replace the code that shows the angry image with `Image.SWORD`
- Replace the code that shows the fabulous image with `Image.HEART`

What do you see when you run the program? What if you restart the program a few times?

**Find more images on the *micro:bit Image Cheat Sheet***:
https://microbit-micropython.readthedocs.io/en/latest/tutorials/images.html

## ☑ CHECKPOINT ☑

**If you can tick all of these off, you can go to Part 9:**

☐ Your Tamagotchi shows a sad face if their hunger is over 5.

☐ Your Tamagotchi also shows a meh face if their boredom is over 5.

☐ Your Tamagotchi also shows an angry face if their anger is over 5.

☐ Your Tamagotchi also shows a happy face if their hunger, anger ***and*** boredom is below 5.

☐ You tried it on your real-life micro:bit

# Extension 9: Time for a treat!

## Task 9.1: Scoring!

We're going to make it so Tamagotchi owners can get points to make the game more fun! With a high score, our Tamagotchi will become ✨*fabulous*✨.

Now we're going to add a score for keeping our Tamagotchi happy.

1. Add a variable called `score` at the top of your code. Set it to zero.

## Task 9.2: Happy Tamagochis get more points

1. We're going to add another if-else block to check how hungry, bored and sleepy your Tamagotchi is. The lower these values are, the happier your pet is and the more points you will score!
2. Add another if/elif statement after the ones you added in Part 8.
3. Increase the score by 1 point if the `hunger`, `sleepiness and boredom` are **all** less than 4.
4. Increase the score by 2 points if `hunger`, `sleepiness and boredom` are less than 2.

Make sure your code doesn't add both 1 and 2 to the score if `hunger`, `sleepiness and boredom` are less than 2.

## Task 9.3: Keeping score

We want to see what the score is too! Add some code to show it on the display.

After the `name` is displayed, and before `hunger, boredom,sleepiness` and `age` are displayed, display the score variable.

## Task 9.4: A fun show!

When our score gets to 10, we want to treat ourselves with more fabulous images!

1. If we have a high score, we should show a cool image after we display it. Near the code where we scroll our score, check if `score > 10`.

2. If the score is > 10, display `Image.FABULOUS`.

## Task 8.5 Try it out!

Test your code:

1. Click the **'Send to micro:bit'** button to run it on your micro:bit
2. Does score calculate and display correctly?
3. Do you get a celebratory image when score > 10.

## ☑ CHECKPOINT ☑

**If you can tick all of these off you are finished with the workbook!**

☐ Your Tamagotchi keeps score while it's alive and healthy

☐ Your score is displayed when the Tamagotchi is shaken

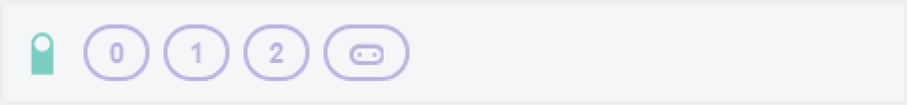
☐ Your Tamagotchi shows a fabulous image when your score is over 10.

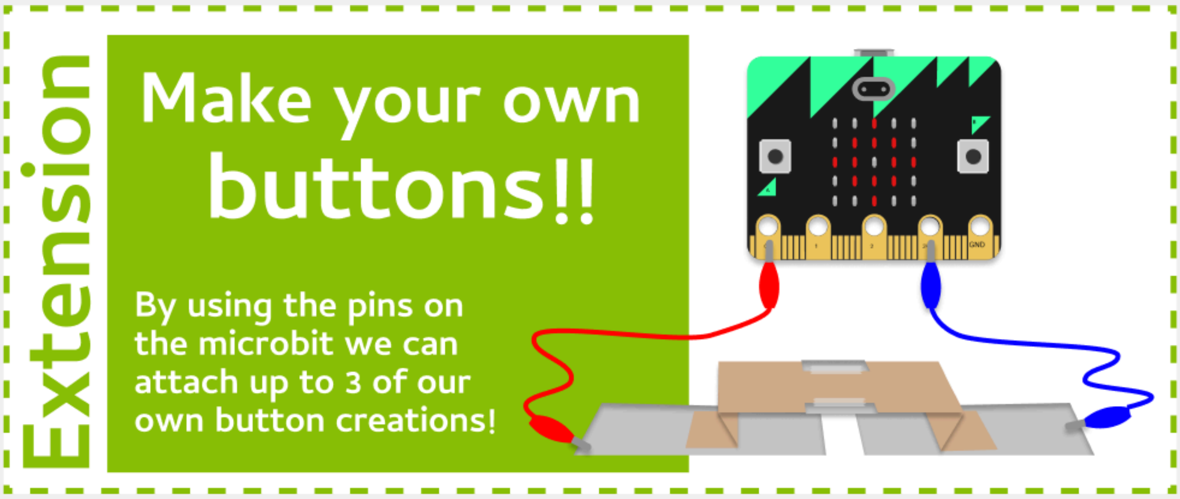
☐ You tried it on your real-life micro:bit

# Extension 10: Building Paper Buttons

## Task 10.0: Getting ready for pins

We're going to be using pins directly for this extension. You will need to watch out for the pins in the Microbit playground. They look like this:




Extension
Make your own buttons!!
By using the pins on the microbit we can attach up to 3 of our own button creations!
GND

## Task 10.1: Get the pins ready

**We need to start by resetting the pins so they are ready to read**

1. Go to your code and add a new line after the import section.

2. Add this line of code to prepare pin 0 by resetting it.
```
pin0.read_digital()
```

3. Repeat for pin1 (and pin2 if you want an extra button!)

## Task 10.2: Goodbye microbit buttons, hello my buttons!

**We'll edit our code to use handmade buttons instead of microbit buttons.**
*You can copy add to this later to use both the microbit and handmade buttons.*

1. Go to the line where you check if Button A is pressed.

2. We want to check if there is current in the circuit on pin0 (instead of checking if the button is pressed). **Replace** `button_a.is_pressed()` with `pin0.read_digital()`

3. Repeat by replacing **Replace** `button_b.is_pressed()` with `pin1.read_digital()`

4. Run your code in and test it out using the first pin button!

## Task 10.3: Build a button!

1. Follow the instructions on the next page to build your button

2. Learn how to make a basic button and connect it to your Micro:Bit to use your code in real life!

3. Extension 11 shows you how to make a button from cups that you can decorate to look like a pet.

## ★ Bonus 10.4: Want more actions?! ★

**★ Use third pin! ★**

Create another action and a button on pin2

**★ Use the Micro:Bit buttons again! ★**

With 2 buttons and 3 pins, you could have up to 5 actions!
So *add back in your original Micro:Bit button code, but make some changes.*
*Make sure you have different action names and pictures for each button/pin.*

## ☑ CHECKPOINT ☑

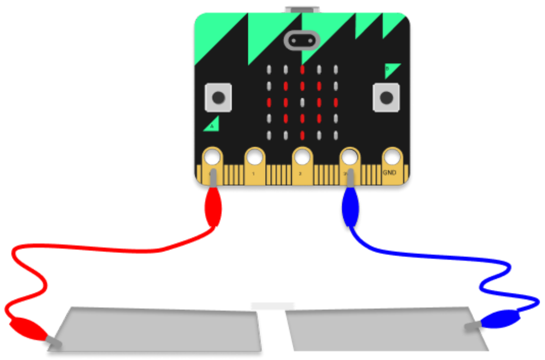
**If you can tick all of these off you have finished this Extension:**

☐ You made buttons/contraptions that complete circuits for you game

☐ Your game completes actions based on buttons connected to pins

# Build a Button

## Build the broken circuit

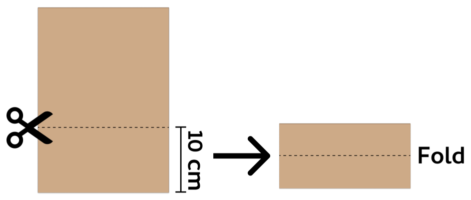


1. Get 2 alligator clips, connect one to **Pin 0** and the **3V Pin**.

2. **Connect** the other ends of the alligator clips to 2 pieces of aluminum foil.

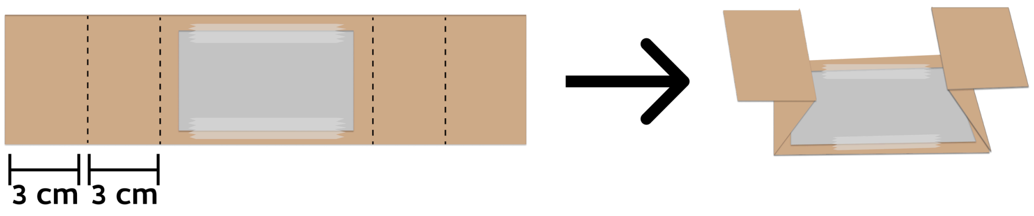
3. **Stick** those to the table so there is a gap in the middle.

## Build the button

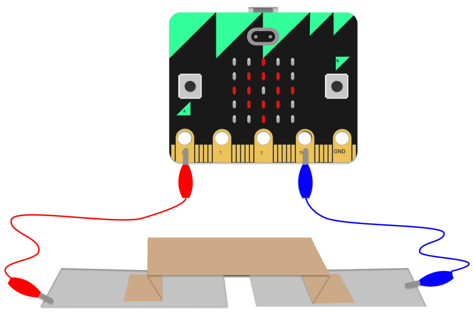


4. **Get** an A4 piece of paper.
   **Cut** a 10cm wide strip.
   **Fold** it in half long ways, for added strength.

5. **Take** the folded trip.
   **Stick** aluminium foil to the center.
   **Crease** the paper to make the button shape.



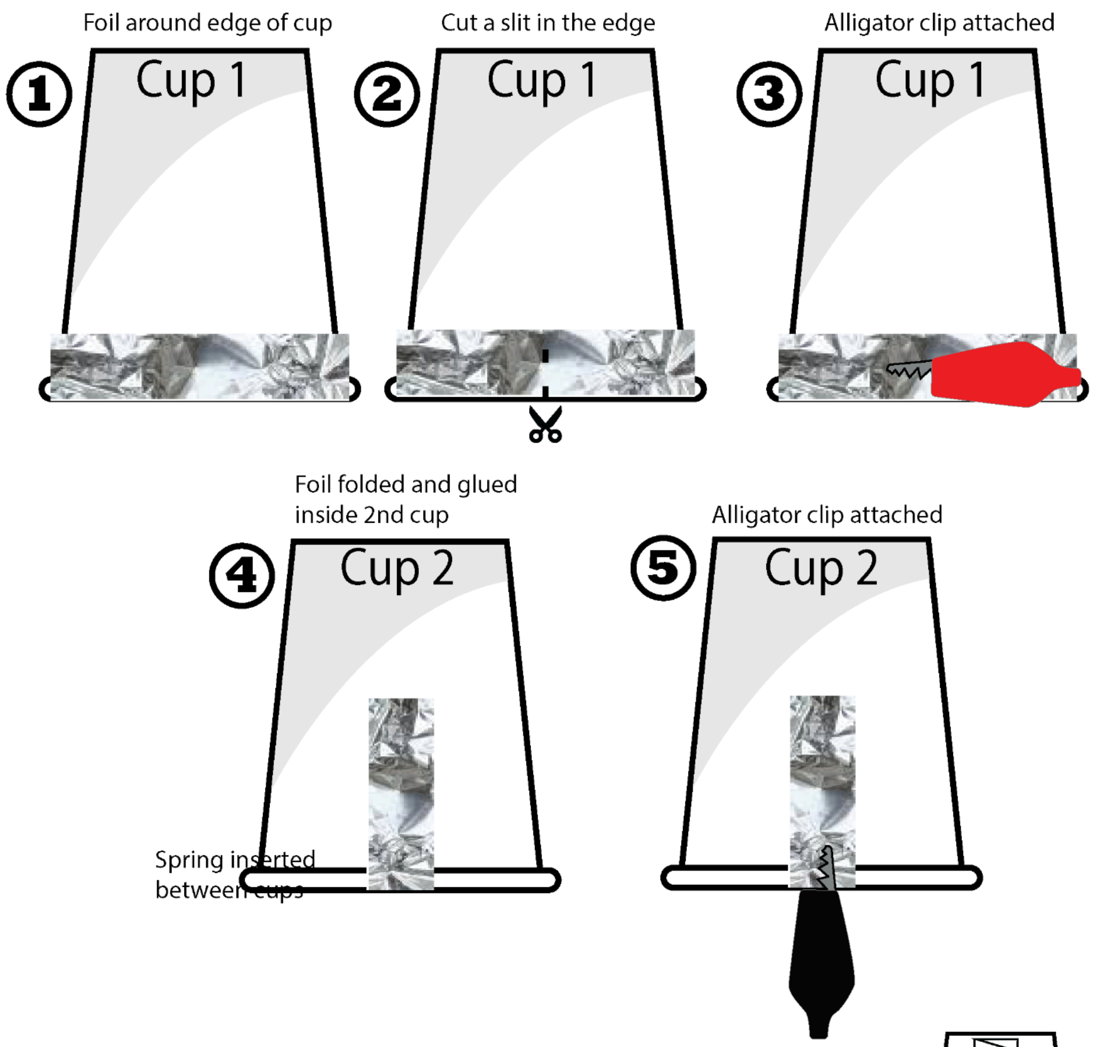
## Add the button



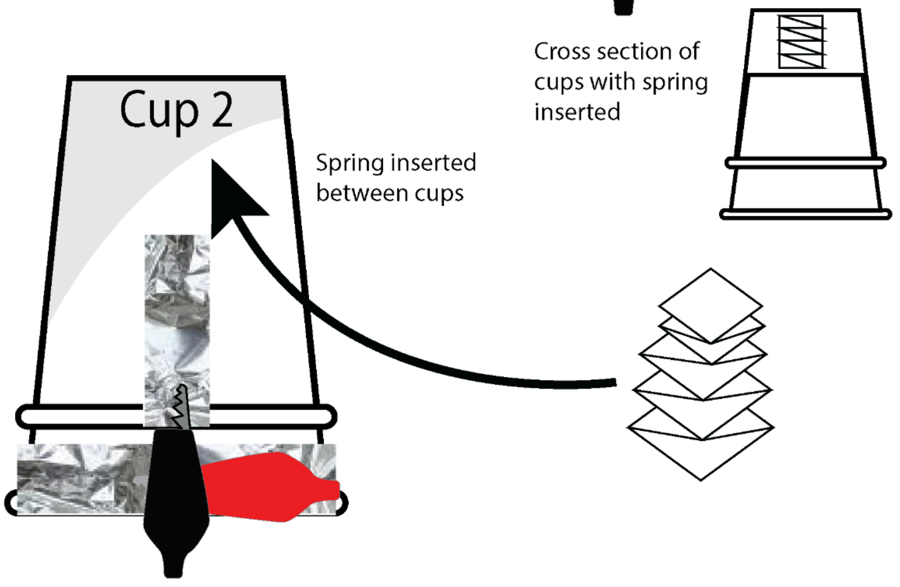
6. **Place** the button on the broken circuit.
   **Align it** so that when you squish the button the aluminum patch closes the circuit.
   **Stick** down the button

7. **You're ready to try your code on the Micro:Bit!**

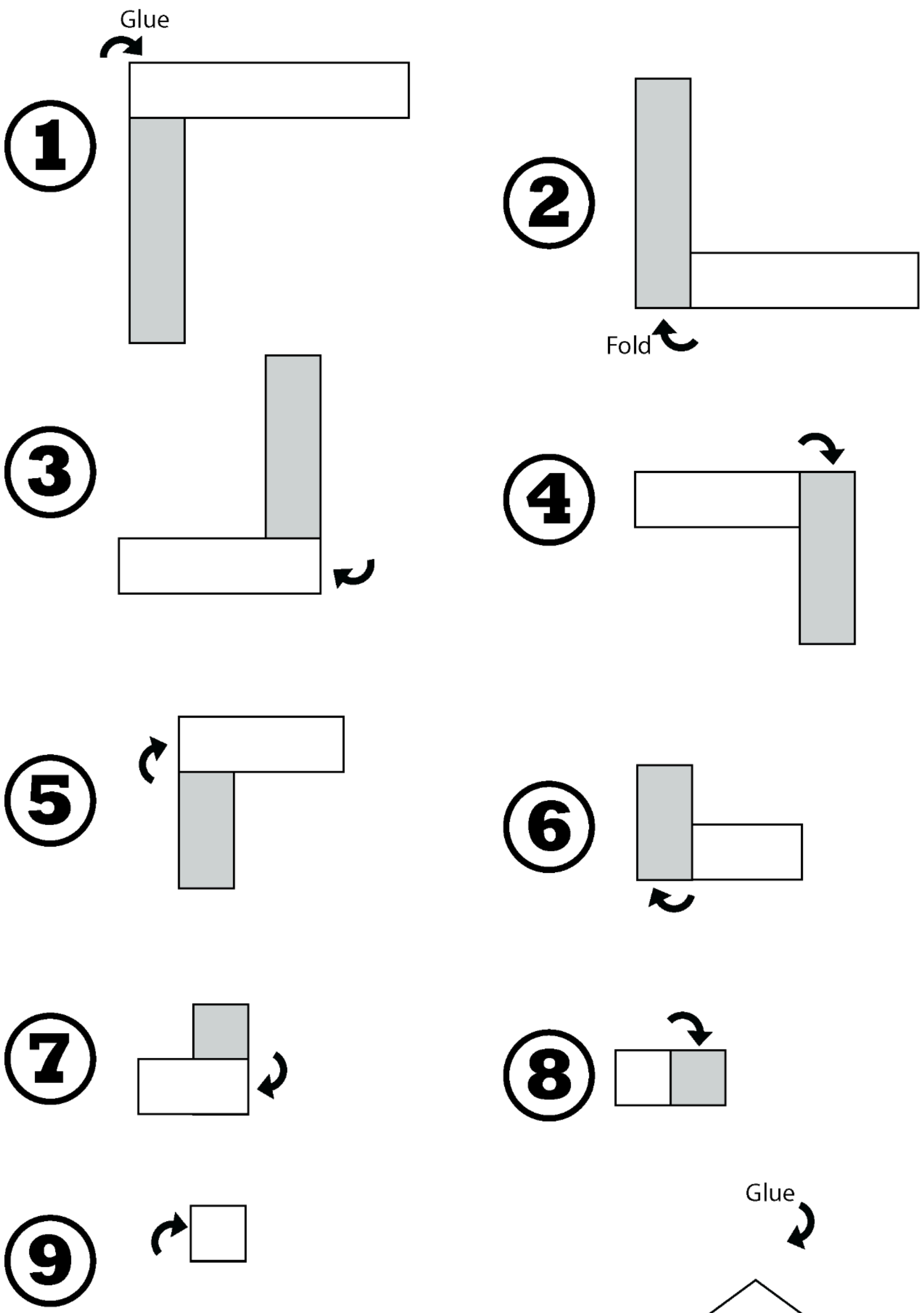
# Build a Cup Button



## Assemble



# Build a Paper Spring



Keep folding unti you run out of paper.