Introduction and Theory

This document contains the instructions for using a micro bit to collect data in an environment to maximize a [indoor] plant’s health. We will pretend that we are inside a spaceship and want to grow some plants. The group of kids make up a team of scientists, each tasked with ensuring the health of a different plant assigned to them. We will be considering light, temperature, relative humidity, water, and soil health specific to a given plant. We will build each component step by step in chronological order.

We will begin by finding an optimal spot to place our plant:

1. We will find a location in or outside the house with the optimal temperature for this plant.
2. Next, we will check the light at these spots to see if any modifications need to be made to accommodate the plant. For example, we may place a fan nearby to cool down that section of the house.
3. We will also check the relative humidity of the air in this spot. Canada has lower humidity in the winter, so we may need to accommodate this as well. Again, we can look for a new location with higher humidity (like in a bathroom), or we can try to raise the humidity locally.
4. Now that we have found the perfect spot, we can use the micro bit to track the plant’s health. We will be concerned with checking the soil’s water level and how nutritious the soil is.

We can use the tool from our first step from time to time since all of those parameters are likely to change seasonally.

Each of these steps corresponds to one activity. Each activity will have a hardware part and a programming part.

This is a form to fill out about the chosen plant:

|  |  |
| --- | --- |
| Plant specifications | |
| Light |  |
| Temperature |  |
| Relative Humidity |  |
| Water (soil moisture) |  |
| Soil health | // maybe. Not sure how to track nutrients. Something to do with salt? |

* There will also be another form to track the results of an exercise as needed

# **Theory**

## **Factors for a [indoor] healthy plant.**

1. **Light**

All plants need light to survive. They use light received from the sun (sunlight) or artificial light to make food. This process is called photosynthesis. Different types of plants need different brightness and amounts of sunlight. Some plants may prefer to be in sunlight for most of the day, while others are the healthiest, mostly in the shade. It’s essential to find a spot in the house that gets the right amount of light for your plant!

1. **Temperature**

Another important part of plant health is temperature. What temperature a plant prefers depends on where it originated from. All places on Earth have plants that naturally grow there. When we choose to grow a plant that originally grew in a different part of the world, we must consider what temperature it will be the healthiest in.

1. **Relative Humidity**

Relative humidity tells us how much water is in our surroundings compared to the amount of air. Each plant requires a different range of relative humidity. For example, tropical plants generally need more humidity than cacti.

1. **Water**

All living things need water to survive. Plants drink water through their roots. They need water for photosynthesis and also to absorb and move nutrients from the soil. Too much or too little water is bad for a plant’s health.

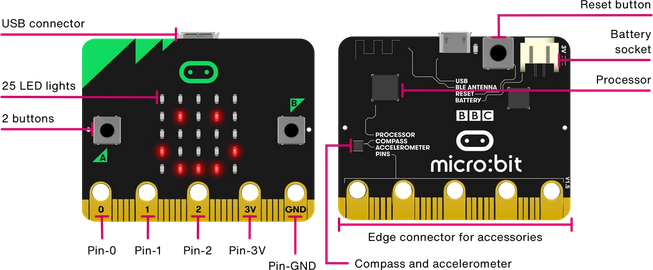
## **micro:bit**

### **Hardware**

A computer has inputs, a place to store and do something with information from the input, and a way to output things. A micro:bit is a tiny computer.

An input is a place where the micro:bit gets information from. We also call the information we get input.

An output a place where the micro:bit puts out information (like a screen or speaker). We call the information we want to share the output.

Let’s explore the different parts of a micro:bit [relevant to our activites].

* **Reset Button**

Once you put in the batteries, the micro:bit is always on. Sometimes we want to reset the micro:bit. This is like restarting your computer or phone. We can do this by pressing the black circle button on the back.

* **Processor**

This is the brain of the micro:bit. This is where all the code we write goes. The micro:bit reads our code as instructions, so it knows what to do.

* **LED Lights**

There are 25 LED lights on a micro:bit. We can control which lights to turn on or off to create patterns. In this way, we can use the lights to output information. These LED lights are also light sensors, which means they input the amount of light shined on the micro:bit.

* **Buttons**

There are two buttons on the micro:bit. It is always waiting to see if anyone will press a button. If a button is pressed, then it sends a signal as an input. If we receive this signal, then we can make something happen!

* **Touch Logo**

The touch logo is the green symbol right above the LED lights. It is just like the buttons. The only difference is that we don’t need to press it. All we have to do is touch it!

* **Temperature sensor**

There is a temperature sensor inside the processor of the micro:bit. This keeps track of the temperature around the micro:bit and sends the micro:bit this information as input.

* **Pin**

We can connect different things to the pins located at the bottom of the micro:bit to have more inputs and outputs.

### **Software [coding]**

* **Blocks**

In MakeCode, each component is called a block. Each concept that is discussed in this section will correspond to a block in MakeCode.

[ USE FROM CODEMAKERS - THANKS BEA ]

Errors:

Line x NameError. “someName” is not define

* Did you import microbit libraries?
* First line of code should be: from microbit import \*.

Line x attribute error ‘function’ input has no attributes temperature

Syntax error: Invalid syntax

* Make sure the tabs are in the proper place

ACTIVITIES

# Activity #1: Welcome to the spaceship

**Grades**: 3 - 5

**Length:** 30 minutes

**Description:** Define precision agriculture. Introduce the scenario for the following days.

**Materials**:

**Procedure:** After introducing yourself giving any auxiliary information about precision agriculture (e.g. women in computer science, CS at the University of Manitoba, etc.). Ask the class if they know what precision agriculture is.

What: precision agriculture is when we collect data about the environment to make decisions regarding a crop to maximize returns and preserve resources. (say this in your own words)

Why: precision agriculture helps us minimize waste and preserve resources, so it is good for the environment. It also helps us track a crop’s health to make sure it’s as healthy as possible.

Share this scenario:

“*For the next week, imagine that you are part of a team of scientists inside a spaceship. Imagine that your house or room is a spaceship. This team’s mission is to grow plants on the spaceship for food and to improve air quality. Each of you has a different type of plant that you have to make sure grows. You have limited supplies and space, so you want to make sure each plant is as healthy as possible. Together, we will be making and using precision agriculture tools to help us make the best decisions for our plants.*

*Let’s start by opening up our kits. Inside you will find a unique plant whose health you are in charge of! You will also find a sheet of paper that will give you information about the plant. This is important, so keep it safe. You will also find a microbit and some other things that I will explain later on in the week.”*

Ask them to ensure that the kit contains: one plant seed, a bag of soil, a little pot, an information sheet about the plant, a microbit, two nails, two alligator clip wires (one black, one red).

Now give the kids a chance to fill out an activity about their plant.

After they fill out this activity, they will introduce their plant to each other. Maybe explain your plant first. Something like: “Hi! I am Kajal, the sunflower expert. I will be making sure that we can grow sunflowers on our plant. [insert an interesting fact from the information sheet’ This is my sunflower, Sunny.”

Next, come up with a team name together. From now on, refer to this group by their team name. Let them decide where their spaceship is going.

# 

# Activity #2: The temperature sensor

**Grades**: 3 - 5

**Length:** 30 minutes

**Description:** Display the room temperature. As homework, keep track of how it changes throughout the day for each spot (if they want).

**Materials**: micro:bit

**Procedure:** After introducing yourself and the micro:bit giving any auxiliary information about precision agriculture (e.g. women in computer science, CS at the University of Manitoba, etc.) offer the class to play the following scenario:

“Welcome back, Team \_\_\_\_\_\_\_\_\_\_\_. *[Do you remember what your mission is? Does anyone want to share it?] Imagine that you are part of a team of Computer Scientists inside a spaceship. Your job is to grow plants on the spaceship so that your team has something to eat and improve air quality. Each of you has a different type of plant that you have to make sure grows. You have limited supplies and space, so you want to make sure each plant is as healthy as possible.*

*Yesterday, you selected some spaceship areas that get the right amount of sunlight for your plant. Another important part of plant health is temperature. What temperature a plant prefers depends on where it originated from. All places on Earth have plants that naturally grow there. When we choose to grow a plant that originally grew in a different part of the world, we must consider what temperature it will be the healthiest in.*

*Today you will create a device that will help you track the temperature in each part of the spaceship. Then you will use the device to narrow down your list of spots based on what kind of climate is most suitable for your plant.”*

Explain: micro:bits have a temperature sensor that checks how hot it’s CPU is. It inputs this temperature to the micro:bit. The micro:bit usually doesn’t get super hot so the temperature returned is fairly close to the real temperature. It is off by constant at a given time. We will be using the led matrix to display this input (temperature). We don’t want the temperature always to be showing, so we will use the shake input to decide when to show the temperature.

Instructions for making the temperature tracker (picture at the end):

1. Connect the micro:bit to your computer using the USB port.
2. Open https://python.microbit.org/v/2
3. Click on the connect button and select your microbit
4. Code introToConditionals.py. This code has comments to help you explain some basic coding concepts which will be helpful for the next step and to familiarize everyone with the editor and the microbit.
5. Tell them to name their temperature tracker something cool. This will be the name of their project.
6. Code basicTemperatureSensor.py
7. Compare the results to the real temperature in their house. Fill out the following:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Real temperature | Micro:bit’s temperature | Difference in real and micro:bit  (specify if the negative or positive) |
| House temperature (check thermostat) |  |  |  |
| Fridge temperature |  |  |  |
| Outside temperature |  |  |  |

1. Why is the temperature different?

Explanation:

The micro:bit's temperature sensor checks how hot the CPU is. The temperature sensor has high precision, but isn't accurate.

What this means is that the temperature given is very precise but not accurate. Precise: the micro:bit can sense changes in the temperature really well.

Inaccurate: It's not perfectly accurate because remember the micro:bit is returning the CPU’s temperature which might be a little hotter than the actual temperature of the room. To account for this we will find an approximate offset value.

So on my micro:bit it returns 28 degrees when it's actually 25 degrees, and it returns 12 degrees when it's actually 9 degrees.

1. Use the table you filled out to come up with an approximate offset value and then update your code to account for the offset value. Code: temperatureComplete
2. Shake the micro:bit to see if it works.
3. Ask them to make a file that will hold all the combined code over the weeks. The name of this file can be something creative. Whatever they want to call their gadget.
4. Ask them to copy and paste the code they have so far into this new file.They will alter this code based on step 13.
5. Use code from: lightSensorAsFunction.py to tell them what functions are and define the code from step 12 into a function. Ask them what they think the benefit of a function is (this will be more clear in the future activities, where we will ask them this question again).
6. Shake the micro:bit to see if it works.
7. Click flash to upload the program to the microbit. Safely disconnect the micro:bit from your computer.
8. Done!

Make sure that they do this part safely and with adult supervision.

Instruct them to unplug the micro:bit and explore the spaceship to find the warmest and the coolest spots (safely and with supervision). Remember to come back are share any interesting results with your teammates.

The last step is to see what temperature their plant requires. Go to the list of possible spots and make sure the temperature is healthy for the plant. If it is NOT, then cross off that spot from your list.

If none of the spots remain for a kid after the last step, brainstorm with the entire team to find solutions. Should they use artificial light (lamp), add shade to the spot, place a fan nearby to cool down the temperature etc. Don’t give them the answers. Prompt them with questions.

# 

# 

# 

# Activity #3: The light sensor

**Grades**: 3 - 5

**Length:** 30 minutes

**Description:** use the microbit’s light sensor to display information about the quality of light for a given plant. Talk about how we can store this data to see how light intensity changes in a spot depending on the time of the year.

Factors to consider: hours of light, light intensity

Source for available display images: https://microbit-micropython.readthedocs.io/en/v1.0.1/tutorials/images.html

**Materials**: micro:bit

**Procedure:** After introducing yourself and the micro:bit giving any auxiliary information about precision agriculture (e.g. women in computer science, CS at the University of Manitoba, etc.) offer the class to play the following scenario:

“Welcome back, Team \_\_\_\_\_\_\_\_\_\_\_. *[Do you remember what your mission is? Does anyone want to share it?] Imagine that you are part of a team of Computer Scientists inside a spaceship. Your job is to grow plants on the spaceship so that your team has something to eat and improve air quality. Each of you has a different type of plant that you have to make sure grows. You have limited supplies and space, so you want to make sure each plant is as healthy as possible.*

*Yesterday, we ensured that we have an area of our house with the optimal temperature for our plant’s health. Before we start planting, we need to find the correct place for our plant to grow. All plants need light to survive. They use light received from the sun (sunlight) or artificial light to make food. This process is called photosynthesis. Different types of plants need different brightness and amounts of sunlight. Some plants may prefer to be in sunlight for most of the day, while others are the healthiest, mostly in the shade. It’s essential to find a spot in the house that gets the right amount of light for your plant!*

*Today we will create a device that will help you track the amount of light in each part of the spaceship. Then you will use the device to find the right spot for your plant.”*

Use a diagram of a micro:bit or show a micro:bit to them. Point out the LED matrix and explain that each LED can be used as input and output. Each LED is a light sensor that returns a number ranging from 0 (dark) to 255 (really bright). We will use our micro:bit as a precise light sensor which will update the light value and show us the result in real time.

Instructions for making the LED light tracker (picture at the end):

1. Connect the micro:bit to your computer using the USB port.
2. Open https://python.microbit.org/v/2
3. Click on the connect button and select your microbit
4. Code introToConditionals.py. This code has comments to help you explain some basic coding concepts which will be helpful for the next step.
5. Code basicLightSensor.py
6. Now ask them to take a look at their plant’s information sheet at the section that talks about light. Using this information, play around with the amount of light shining on the microbit to find rough values to fill out the following chart:

|  |  |  |
| --- | --- | --- |
| Light quality | Microbit light level (0-255) | Plant preference  (yes/no) |
| Bright (direct) | What is the minimum light level to be considered bright? Ex: greater than 150 |  |
| Indirect |  |  |
| Dark | What is the maximum light level to be considered dark ? Ex: less than 100 |  |

1. Depending on the students and time you can skip the following step:

* Use the knowledge about conditionals and the chart above to display a happy face when the light is of the plant’s preference, meh face if it’s okay and sad face for if it’s bad for the plant.

1. A possible version of the correct code for step 7 can be found in lightSensorComplete.py
2. Modify their main file that they made last week (name of file is whatever they named their gadget) to make a new function for the light sensor.

Code: lightSensorAsFunction.py. Ask them what they think the benefit of a function is (this will be more clear in the future activities, where we will ask them this question again).

1. Click flash to upload the program to the microbit. Safely disconnect the micro:bit from your computer.

Make sure that they do this part safely and with adult supervision.

The last step is to use the information from your light sensor and find several possible right spots. In the next activity, we will narrow down our choices.

If someone wasn’t able to find a spot: encourage the team to brainstorm ideas to help create the light for this plant. Does it need more light? They can place a lamp nearby. Does it require less light? Can they shade the plant somehow (possibly using other plants)? Don’t give them the answers. Prompt them with questions.

If there is time: discuss how the micro:bit can be improved. Our micro:bit is giving us precise information about the light but how useful is it without storing and analyzing this information. In a real world situation, we could have multiple light sensors to track the quality of light in a certain place over a period of time. We could then analyse this information for all the possible spots and use this information to inform our decisions.

# Activity #4.5: Plant your seed

**NOTE: DO THIS ACTIVITY AT THE END OF ACTIVITY 4 OR THE START OF ACTIVITY 5. WHICHEVER WORKS OUT.**

**Grades**: 3 - 5

**Length:** 15 minutes

**Materials**: seed, soil, pot, stickers, markers/paint/crayon/glitter whatever they have

**Description:** Guide the students through putting soil in their pot and planting the seed. Give them some time to decorate their pot if you have time. Otherwise, let them know that they can decorate their plant (on their own time).

# 

# Activity #5: The water sensor

**https://www.youtube.com/watch?v=n0WRQf11Pzo**

**Grades**: 3 - 5

**Length:** 1 hour

**Description:** Check soil’s moisture level to make sure that the plant is not getting under/over watered.

**Materials**: micro:bit, two nails, two alligator clip wires, soil divided into three cups ,a plant - more specifically - soil, water

**Procedure:** After introducing yourself and the micro:bit giving any auxiliary information about precision agriculture (e.g. women in computer science, CS at the University of Manitoba, etc.) offer the class to play the following scenario:

“Welcome back, Team \_\_\_\_\_\_\_\_\_\_\_. *[Do you remember what your mission is? Does anyone want to share it?] Imagine that you are part of a team of Computer Scientists inside a spaceship. Your job is to grow plants on the spaceship so that your team has something to eat and improve air quality. Each of you has a different type of plant that you have to make sure grows. You have limited supplies and space, so you want to make sure each plant is as healthy as possible.*

*All living things need water to survive. Plants drink water through their roots. They need water for photosynthesis and also to absorb and move nutrients from the soil. Too much or too little water is bad for a plant’s health. Today you will create a device that will help you track the soil moisture. Then you will use the device to make sure your plant is getting the right amount of water..”*

Explain: We can connect different things to the pins located at the bottom of the micro:bit to have more inputs and outputs.

“The soil itself has some electrical resistance which depends on the amount of water and nutrients in it. It acts like a variable resistor in an electronic circuit. The water is not conductive but the nutrient content is. The combination of water and soil nutrients makes the soil have some conductivity. So, the more water there is, combined with the nutrients, the less the soil will have electrical resistance.

To measure this, we read the voltage on pin P0 using an analog read pin which returns a value between 0 (no current) and 1023 (maximum current).”

Setup:

1. Divide an equal amount of soil into three different containers (make sure the containers have some holes at the bottom for drainage).
2. Label the first container ‘dry’, the second container ‘moist’, and the third container ‘wet’
3. Do not put any water in the container labeled dry.
4. Put water into the third container (labeled wet) until it drains out from the bottom.
5. Put some water in the second container, just enough to wet the top of the soil.

Instructions for making the moisture sensor (pictures at the end):

1. Connect the red alligator clip to the 0v pin on one end and one of the nails on the other end.
   * This wire will input moisture and nutrient information from the soil to the microbit.
2. Connect the black alligator click to the 3v pin one end to the other nail on the other end.
3. Connect the micro:bit to your computer using the USB port.
4. Open https://python.microbit.org/v/2
5. Click on the connect button and select your microbit.
6. Code: basicSoilSensor.py.
7. Click flash to upload the program to the microbit. Safely disconnect the micro:bit from your computer.
8. Press button B the micro:bit to see if it works.
9. Put the nails inside each soil container and and press button B to see the moisture level of this plant. Record this information:

|  |  |
| --- | --- |
|  | Soil moisture level |
| dry |  |
| moist |  |
| wet |  |

1. Go over the first couple lines of the code from soilSensorComplete.py to explain what the difference between constants and variables are.
2. Also explain what breaks are.

while (1==1):

if (some\_condtition):

break

This would cause the while loop to stop if some\_condtition is met.

1. Ask them to use their knowledge from the previous activities and the chart from step 14 to modify the current code for soil sensors. The program should have the following functionalities:
   * display a skull when the moisture level of the soil is too dry or too wet. Image.SKULL
   * display the image of a target if the moisture level is moist.

image.TARGET

* + The display should only clear if on of the following conditions is met:
    1. If you shake the micro:bit
    2. the target image is shown

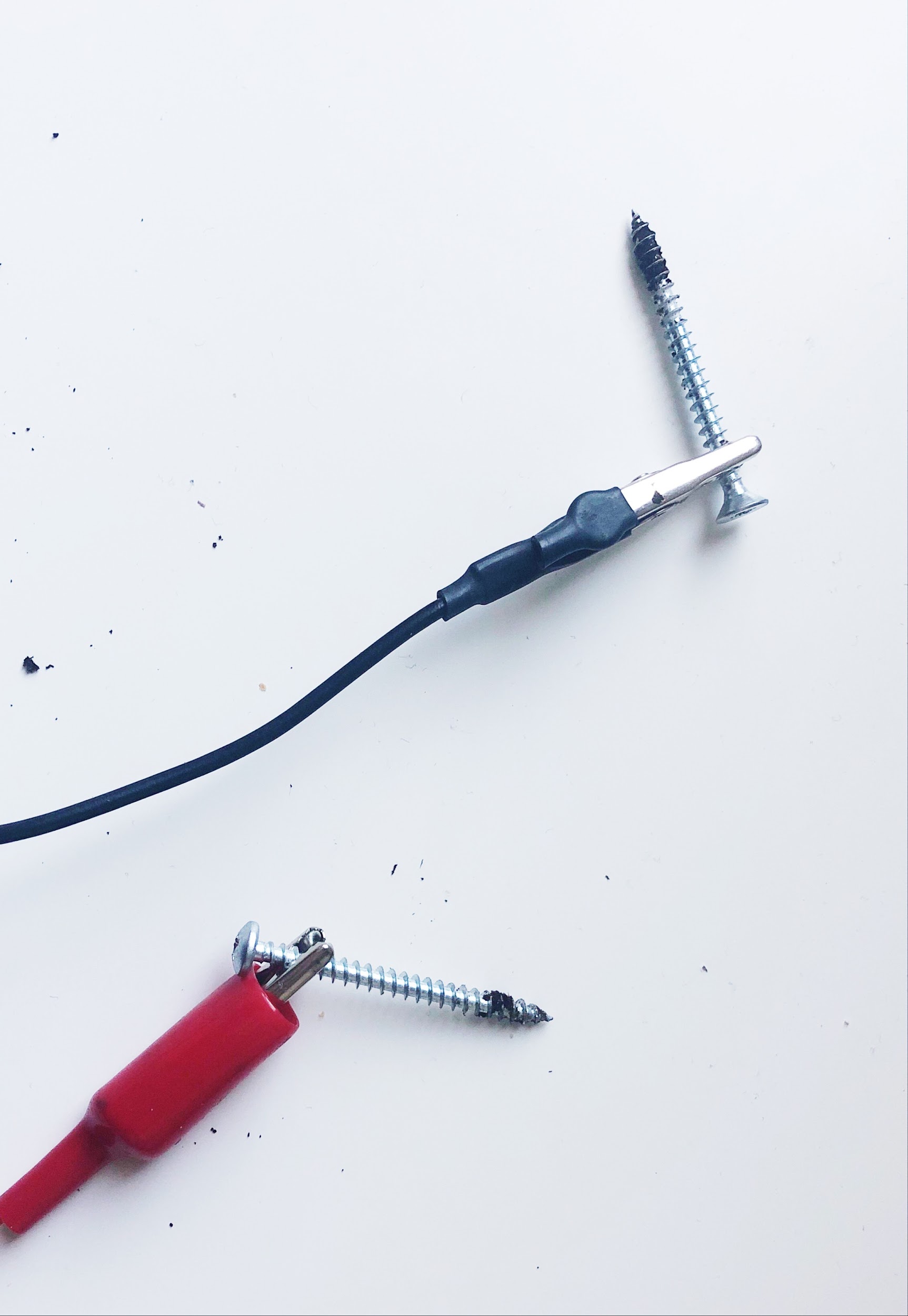
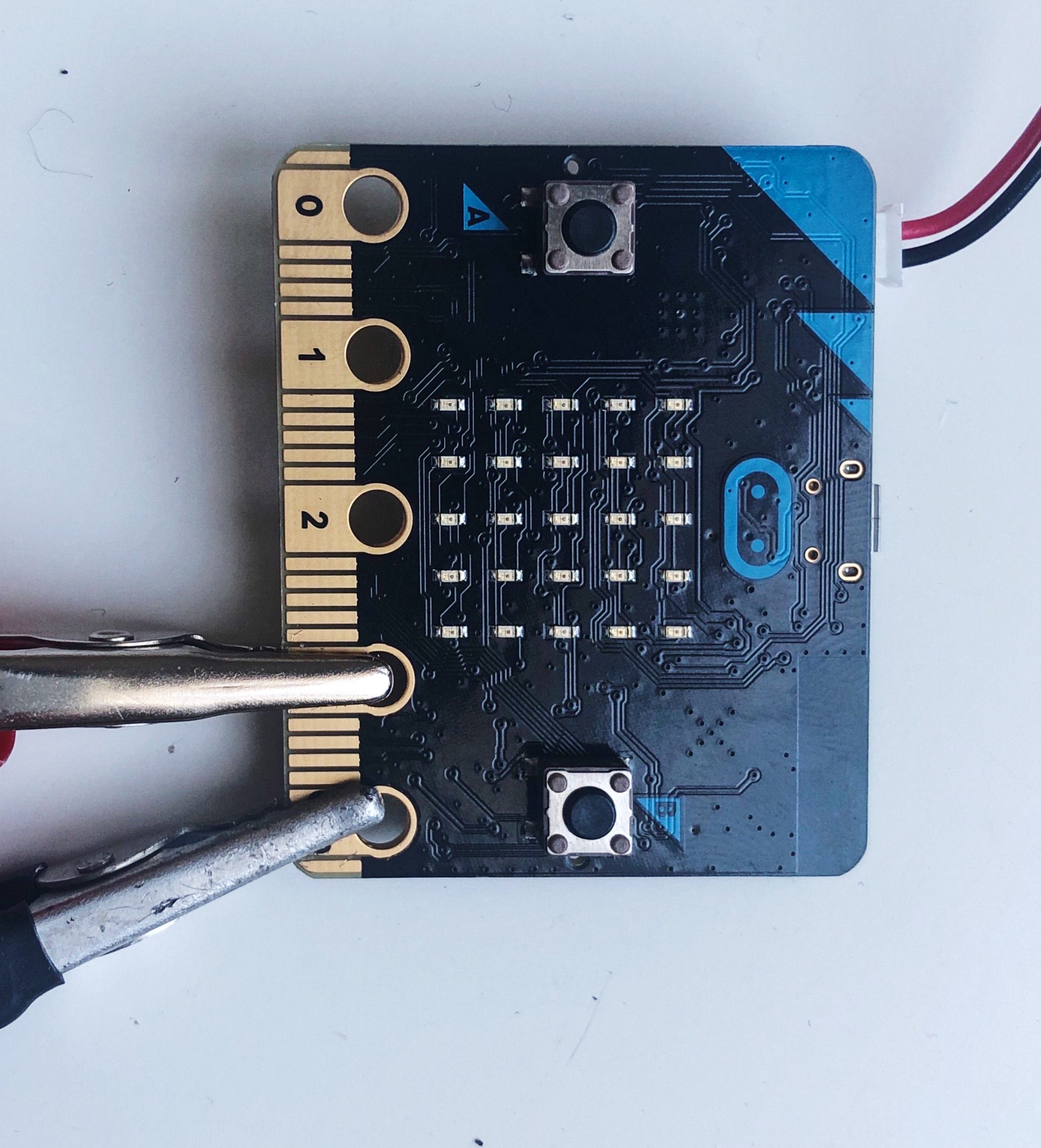
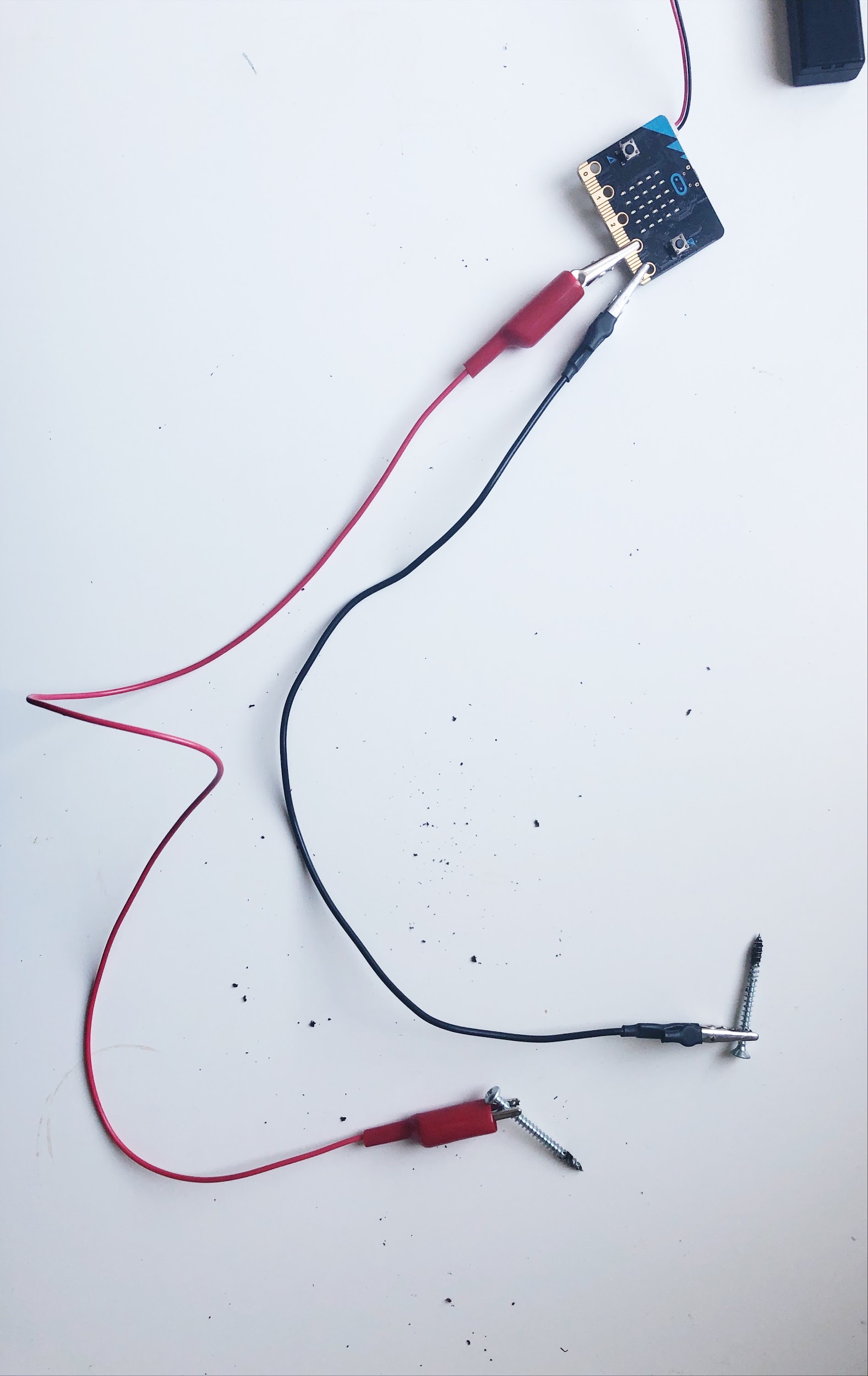
One solution for the specification in step 16 can be found in soilSensorComplete.py

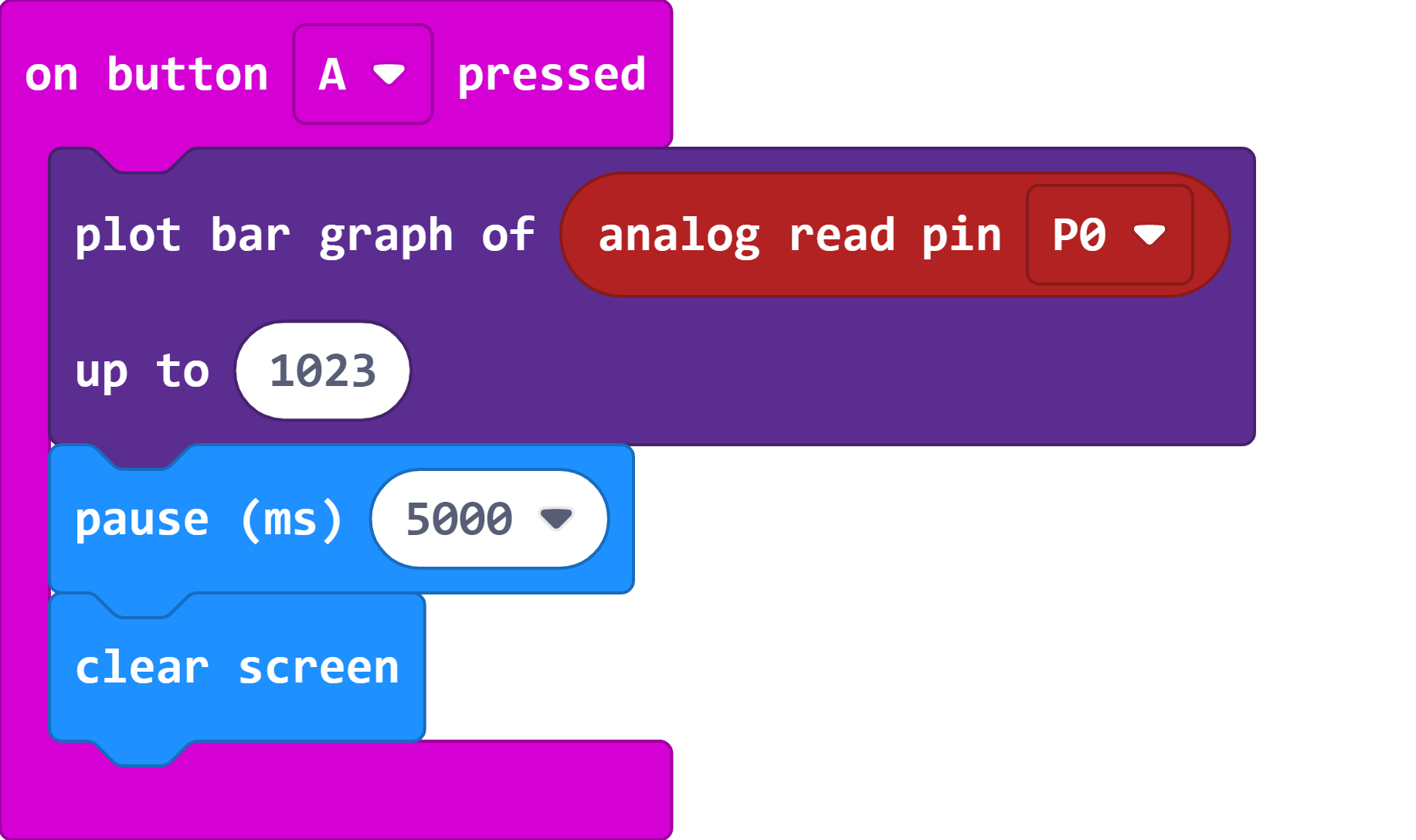
1. Ask them to flash the program onto the micro:bit and test with their three containers of soil to make sure that it works as expected.
2. Modify their main file that they made last week (name of file is whatever they named their gadget) to make a new function for the soil sensor.

Code: soilSensorAsFunction.py. Ask them what they think the benefit of a function is (this will be more clear in the future activities, where we will ask them this question again).

1. Click flash to upload the program to the microbit. Safely disconnect the micro:bit from your computer.
2. Press button B the micro:bit to see if it works.
3. Put the nails inside their own plant’s soil and press button B to see the moisture level of this plant).

[Photos on next page]







CODE

# IntroToWhileLoop.py

from microbit import \*

# everything in the while loop keeps happening over and over again

# as long as the while condition is true

# indentations in the right place are important. You always indent once for all the lines that are inside a function.

# What will this condition do?

# while (not(0==1))

# ask them to come up with their own statement that is always true

# so that we can run the loop forever (ex: while True, while 5==5, while 4!=6, while not false)

while (1==1):

display.scroll("Hello, world.")

# click on flash to upload this program to your microbit

# 

# basicTemperatureSensor.py

from microbit import \*

while (1==1):

if (accelerometer.was\_gesture('shake')):

temp = temperature()

display.scroll(temp)

display.clear()

# click on flash to upload this program to your microbit

# 

# temperatureComplete.py

from microbit import \*

# ask them to come up with their own statement that is always true

# so that we can run the loop forever (ex: while True, while 5==5, while 4!=6, while not false)

while (1==1):

if (accelerometer.was\_gesture('shake')):

microbitTemp = temperature()

offset = -3

actualTemp = microbitTemp + offset

display.scroll(actualTemp)

display.clear()

# click on flash to upload this program to your microbit

# tempSensorAsFunction.py

from microbit import \*

# in this level we take the code we have for the temperature sensor and move it into a method

# why? This allows our code to be more organized and readable.

# this will be really useful as we add more functionality to our microbit

# emphasize the importance of comments

# so you can come back to your code later and know what everything does

# the temp sensor function.

def temp\_sensor():

microbitTemp = temperature()

offset = -3

actualTemp = microbitTemp + offset

display.scroll(actualTemp)

display.clear()

# -------------------------------------------------------------------------------------------

# main loop

# main script. This is like the manager of our program. It will call the correct methods

# depending on how we interact with the microbit

# again, they can put any valid statement that will cause the while loop to run forever

while (1==1):

# if the microbit is shook then call the temp sensor method. It knows what to do.

if(accelerometer.was\_gesture('shake')):

temp\_sensor()

# click on flash to upload this program to your microbit

# introToConditionals.py

# from microbit import \*

# 

# 

# # indentations in the right place are important. You always indent once for all the lines that are inside a function.

# 

# # ask them to come up with their own statement that is always true

# # so that we can run the loop forever (ex: while True, while 5==5, while 4!=6, while not false)

# while (1==1):

# 

# # we have declared a variable called num and set it's value to 5 (for now)

# num = 5;

# 

# # ways to compare to numbers

# # (number 1) == (number 2) // numbers are the same

# # (number 1) <= (number 2) // number 1 is smaller than or equal to number 2

# # (number 1) >= (number 2) // number 1 is bigger than or equal to number 2

# 

# # they can choose which image they would like to display

# # the following link has a list of images that are included in the microbit library

# # https://microbit-micropython.readthedocs.io/en/v1.0.1/tutorials/images.html

# if(num == 5):

# display.show(Image.SILLY)

# elif(num >= 10):

# display.show(Image.PACMAN)

# else:

# display.show(Image.BUTTERFLY)

# 

# # click on flash to upload this program to your microbit

# # change the value of num and predict what will happen

# # change the condition statements

# 

# basicLightSensor.py

from microbit import \*

# ask them to come up with their own statement that is always true

# so that we can run the loop forever (ex: while True, while 5==5, while 4!=6, while not false)

while (1==1):

if(button\_a.is\_pressed()):

# display.read\_light\_level() returns a number from 0-255 (dark-bright)

lightLevel = display.read\_light\_level()

display.scroll(lightLevel)

display.clear()

# click on flash to upload this program to your microbit

# try covering the microbit with your hand and then press A,

# shine a light on the microbit

# lightSensorComplete.py

from microbit import \*

# this is a function

# this line means "I am defining this function and it's called light\_sensor"

# a function needs to have () after it. If light sensor needed information we would sent it in the brackets

def light\_sensor():

# do everything in this loop while microbit is steady

# stops if someone shakes the microbit

while(not(accelerometer.was\_gesture('shake'))):

## read the light level and store it in a variable

## called lightLevel

lightLevel = display.read\_light\_level()

# note: display.read\_light\_level() is a function that someone else wrote from microbit

# our program can use this function because we imported it from the microbit (first line of code)

# show the right display depending on how much light is recieved

if (lightLevel <= 40):

# range: 0 - 40

# too dark for my plant

# will show sad face

display.show(Image.MEH)

elif (lightLevel <= 100):

# range: 41 - 100

# good range of light for my plant

display.show(Image.HAPPY)

else:

# range: 101 - 255

# too bright for my plant

display.show(Image.SAD)

# clear the screen

display.clear()

while True:

# this will loop forever

# if button a is pressed then call the light\_sensor() function.

# it knows what to do

if(button\_a.is\_pressed()):

light\_sensor()

# click on flash to upload this program to your microbit

# lightSensorAsFunction.py

from microbit import \*

# emphasize the importance of comments

# so you can come back to your code later and know what everything does

# -------------------------------------------------------------------------------------------

# the temp sensor function.

def temp\_sensor():

microbitTemp = temperature()

offset = -3

actualTemp = microbitTemp + offset

display.scroll(actualTemp)

display.clear()

# -------------------------------------------------------------------------------------------

# the light sensor function.

def light\_sensor():

while(not(accelerometer.was\_gesture('shake'))):

lightLevel = display.read\_light\_level()

if (lightLevel <= 40):

# range: 0 - 40

# too dark for my plant

# will show sad face

display.show(Image.MEH)

elif (lightLevel <= 100):

# range: 41 - 100

# good range of light for my plant

display.show(Image.HAPPY)

else:

# range: 101 - 255

# too bright for my plant

display.show(Image.SAD)

display.clear()

# -------------------------------------------------------------------------------------------

# main loop

# main script. This is like the manager of our program. It will call the correct methods

# depending on how we interact with the microbit

# again, they can put any valid statement that will cause the while loop to run forever

while True:

# if the microbit is shook then call the temp sensor method. It knows what to do.

if(accelerometer.was\_gesture('shake')):

temp\_sensor()

# if button A is pressed then call the light sensor method. It knows what to do.

if(button\_a.is\_pressed()):

light\_sensor()

# click on flash to upload this program to your microbit

# basicSoilMoisture.py

from microbit import \*

# ask them to come up with their own statement that is always true

# so that we can run the loop forever (ex: while True, while 5==5, while 4!=6, while not false)

while (1==1):

if(button\_b.is\_pressed()):

# reads in the moisture level in the pin

moistureLvl = pin0.read\_analog()

display.scroll(moistureLvl)

# click on flash to upload this program to your microbit

# try covering the microbit with your hand and then press A,

# shine a light on the microbit

# soilSensorComplete.py

from microbit import \*

# fill out chart

#

# soil moisture level reading

# ------------------------------------

# dry 802

# moist 1000

# wet 1014

# use this chart to inform what the micro bit shows in this example

# again, they can put any valid statement that will cause the while loop to run forever

while True:

# constants are a type of variable whose value should never be changed

# in python, we name constants in all caps so we know which values

# should never be reassigned

# constant

DRY = 802

MOIST = 1000

WET = 1020

# variable

num = pin1.read\_analog()

# logic: explain and, or, and not

# why would (num<= DRY or num>=wet) be wrong?

while(not((num > DRY) and (num < WET))):

if(accelerometer.was\_gesture('shake')): # if we just wanted to check the

# moisture level and not water it right now

break

else:

num = pin1.read\_analog() # notice that num is changing (it's a variable)

display.show(Image.SKULL)

if((num > DRY) and (num < WET)):

display.show(Image.TARGET)

sleep(1000)

display.clear()

# click on flash to upload this program to your microbit

# soilSensorAsFunction.py

from microbit import \*

# emphasize the importance of comments

# so you can come back to your code later and know what everything does

# -------------------------------------------------------------------------------------------

# the temp sensor function.

def temp\_sensor():

microbitTemp = temperature()

offset = -3

actualTemp = microbitTemp + offset

display.scroll(actualTemp)

display.clear()

# -------------------------------------------------------------------------------------------

# the light sensor function.

def light\_sensor():

while(not(accelerometer.was\_gesture('shake'))):

lightLevel = display.read\_light\_level()

if (lightLevel <= 40):

# range: 0 - 40

# too dark for my plant

# will show sad face

display.show(Image.MEH)

elif (lightLevel <= 100):

# range: 41 - 100

# good range of light for my plant

display.show(Image.HAPPY)

else:

# range: 101 - 255

# too bright for my plant

display.show(Image.SAD)

display.clear()

# -------------------------------------------------------------------------------------------

# the soil sensor function

def soil\_sensor():

# constants are a type of variable whose value should never be changed

# in python, we name constants in all caps so we know which values

# should never be reassigned

# constant

DRY = 802

MOIST = 1000

WET = 1020

# variable

num = pin1.read\_analog()

# logic: explain and, or, and not

# why would (num<= DRY or num>=wet) be wrong?

while(not((num > DRY) and (num < WET))):

if(accelerometer.was\_gesture('shake')): # if we just wanted to check the

# moisture level and not water it right now

break

else:

num = pin1.read\_analog() # notice that num is changing (it's a variable)

display.show(Image.SKULL)

if((num > DRY) and (num < WET)):

display.show(Image.TARGET)

sleep(1000)

display.clear()

# -------------------------------------------------------------------------------------------

# main loop

# main script. This is like the manager of our program. It will call the correct methods

# depending on how we interact with the microbit

# again, they can put any valid statement that will cause the while loop to run forever

while True:

# if the microbit is shook then call the temp sensor method. It knows what to do.

if(accelerometer.was\_gesture('shake')):

temp\_sensor()

# if button A is pressed then call the light sensor method. It knows what to do.

if(button\_a.is\_pressed()):

light\_sensor()

# if button B is pressed then call the soil sensor method. It knows what to do.

if(button\_b.is\_pressed()):

soil\_sensor()

# click on flash to upload this program to your microbit