Plantagotchi: PyBadge Plant Monitor

Created by John Park

https://learn.adafruit.com/plantagotchi-pybadge-plant-monitor

Last updated on 2024-03-08 03:28:21 PM EST

Table of Contents

Overview 3

- Optional

Build the Plantagotchi 5

- Plug in the Leads and Nails

Code the Plantagotchi in MakeCode Arcade 7

- MakeCode Arcade
- Platagotchi
- Load the Code
- Plantagotchi Application Design
- Startup
- Extensions
- on start Continued
- A Touch of JavaScript
- Sprite Change Functions
- sunLevel
- waterLevel
- Do All the Things!
- Buttons for Testing

Update the PyBadge/PyGamer Bootloader 20

- PyBadge/PyBadge LC Bootloader
- PyGamer Bootloader
- Hardware Checks

Load a MakeCode Game on PyGamer/PyBadge 22

- Board Definition
- Change Board screen
- Bootloader Mode
- Drag and Drop
- Play!

Troubleshooting MakeCode Arcade 26

Overview

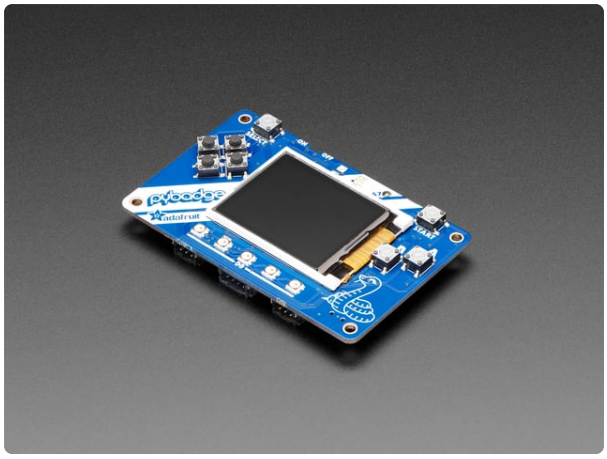
You can build and code your own Plantagotchi -- an adorable virtual plant friend to help you take care of your real potted plants and flowers!

We'll use the PyBadge's analog read capabilities to measure the moisture level of our potted plant's soil. In fact, the dirt will act like one big variable resistor. When soil is dry, it's not very conductive, so its resistance is high. Add water to the minerals in the soil and it will become more conductive, lowering the overall resistance.

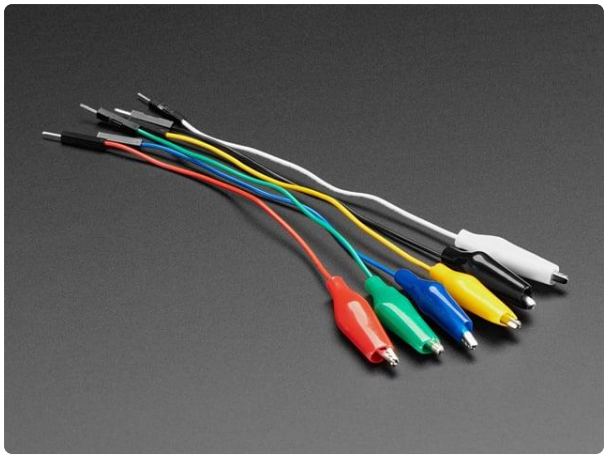
We'll provide a 3.3V reference voltage from the PyBadge over one alligator connected to a nail plunged into the soil. Then, another alligator clip will be plugged into analog pin A3 on the PyBadge and connected to a second nail in the soil. By reading the voltage level on pin A3 (from a possible 0 to 1023) we'll be able tell how moist the soil is. When it's dry, the voltage level read will be low, when it's wet, the reading will be high.

Determining light levels is even easier -- we'll simply use the PyBadge's built-in light sensor!

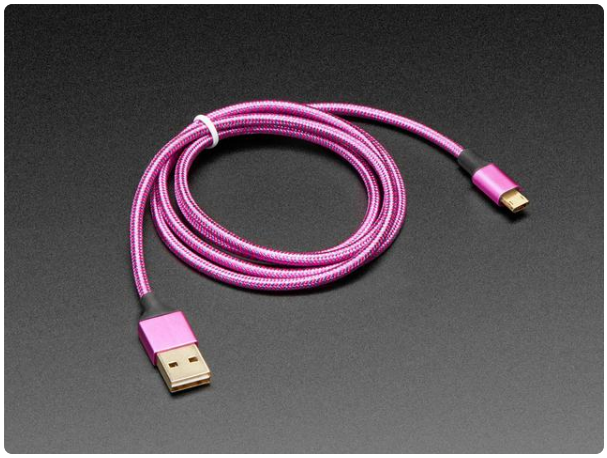
Parts



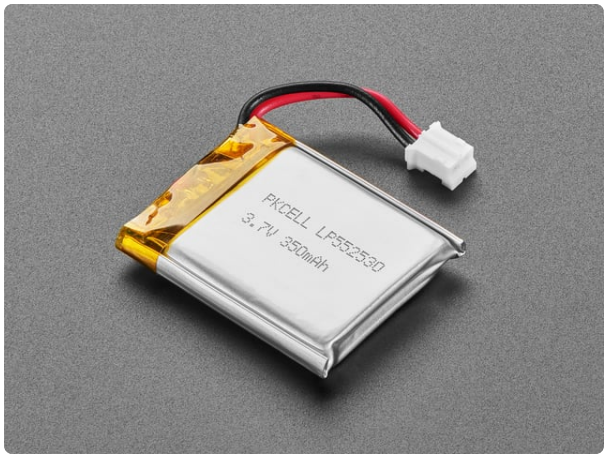
[Adafruit PyBadge for MakeCode Arcade, CircuitPython, or Arduino]
What's the size of a credit card and can run CircuitPython, MakeCode Arcade or Arduino? That's right, its the Adafruit PyBadge! We wanted to see how much we...
https://www.adafruit.com/product/4200



[Small Alligator Clip to Male Jumper Wire Bundle - 6 Pieces]
When working with unusual non-header-friendly surfaces, these handy cables will be your best friends! No longer will you have long, cumbersome strands of alligator clips. These...
https://www.adafruit.com/product/3448



[Fully Reversible Pink/Purple USB A to micro B Cable - 1m long]
This cable is not only super-fashionable, with a woven pink and purple Blinka-like pattern, it's also fully reversible! That's right, you will save seconds a day by...
https://www.adafruit.com/product/4111

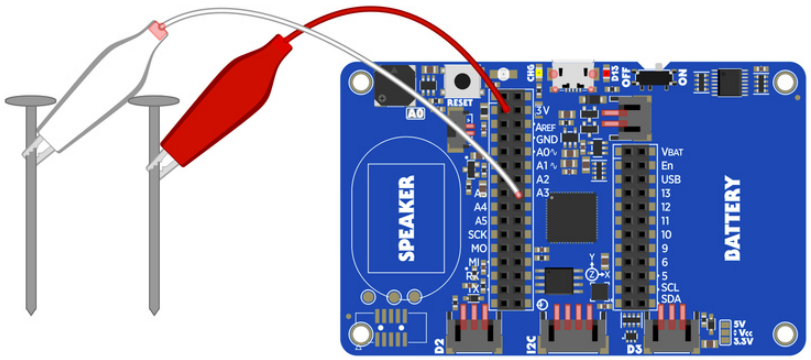


[Lithium Ion Polymer Battery with Short Cable - 3.7V 350mAh]
Lithium-ion polymer (also known as 'lipo' or 'lipoly') batteries are thin, light, and powerful. The output ranges from 4.2V when completely charged to 3.7V. This...
https://www.adafruit.com/product/4237

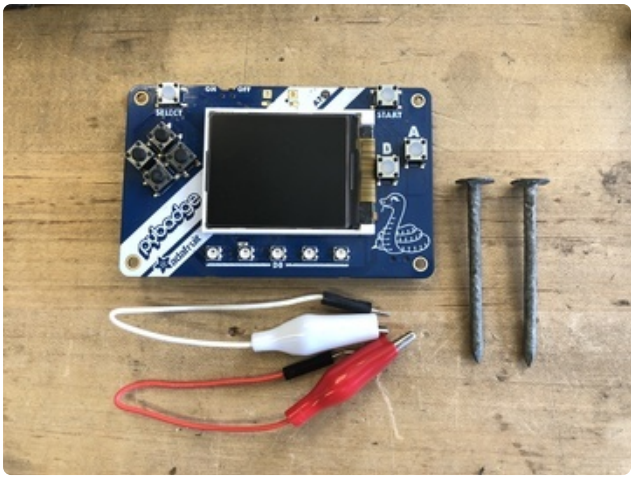
Optional

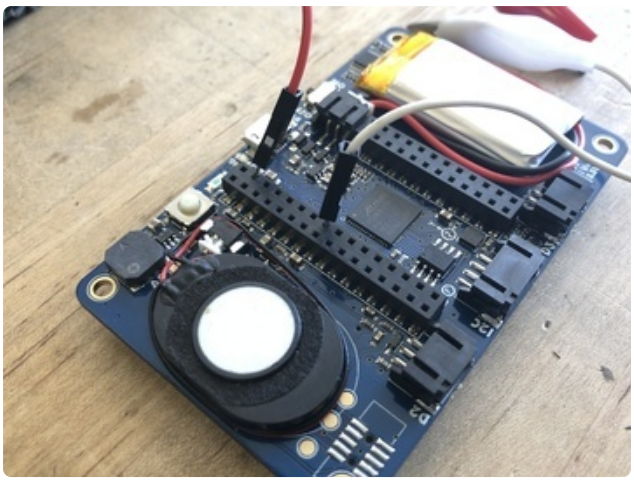
[4-H Grow Your Own Clovers Kit]
We are super pleased to enjoy a partnership with an engaging, empowering foundation like 4-H. Here at Adafruit we drive ourselves to "be...
https://www.adafruit.com/product/4223

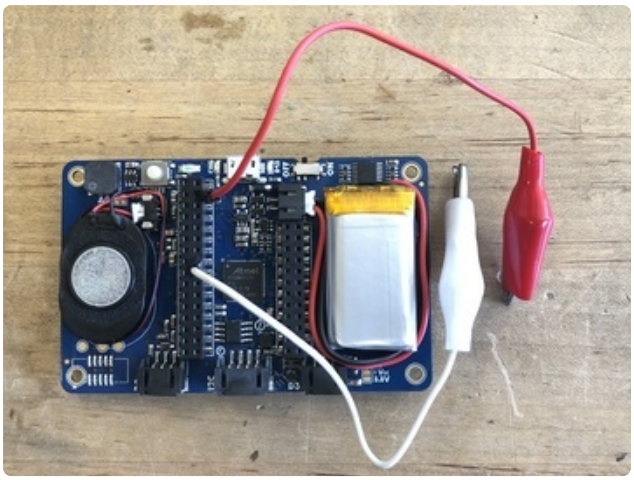
Build the Plantagotchi

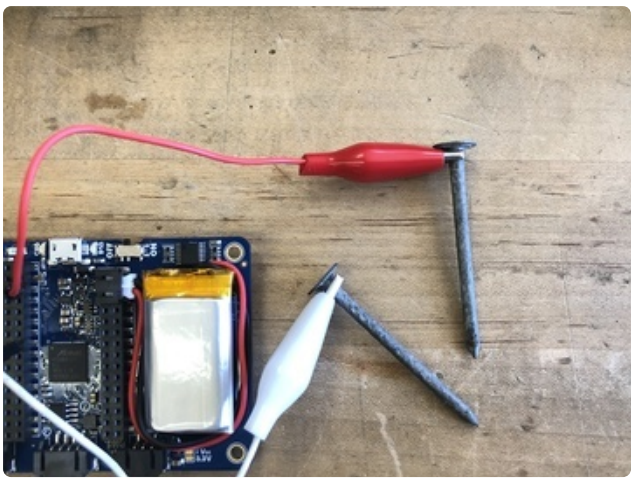


All you'll need to build the Plantagotchi are two nails and a way to plug them into the PyBadge to read the soil moisture levels. The simplest way is to use a couple of alligator clip to male header pin leads as shown here.









Plug in the Leads and Nails

Plug the red lead into the **3V** Feather header pin as shown.

Plug the white lead into the **A3** Feather header pin -- you may want to count it out -- it's the eighth pin from the top as oriented in the diagram above.

Note: the Feather headers are double row headers, and you can use either of the two holes that align with each of the pins.

You can also plug in your battery at this time and secure it with some double-sided sticky tape.

There is an optional speaker plugged into the PyBadge in these photos, this is optional as the PyBadge has a built-in speaker.

Next, let's code the PyBadge Plantagotchi in MakeCode Arcade!

Code the Plantagotchi in MakeCode Arcade

MakeCode Arcade

MakeCode Arcade is a free Microsoft block programming environment designed specifically to make games, but we can also use it for non-game application development. Learning to use MakeCode is easy & fun.

If you're not already familiar with the basics of MakeCode Arcade, check out this guide (https://adafru.it/Elc) on creating a character sprite and moving it with controls.

To start, open a new Chrome browser window (Chrome works best) and go to MakeCode Arcade (https://adafru.it/DCY).

These MakeCode Arcade guides are designed to take you through the fundamentals before tackling more complex games -- even though this NeoPixel Strip Control isn't a game, most of the techniques apply:

- Pixel Art (https://adafru.it/EOl)
- Animation (https://adafru.it/EOk)
- Level Design (https://adafru.it/EOj)
- Sparky Invaders (https://adafru.it/EYf)
- Next Level Game Techniques (https://adafru.it/EYg)

For intermediate-level techniques, check out:

- Re-MakeCode the Classics: Arkanoid (https://adafru.it/E-o)
- Re-MakeCode the Classics: Py Hunter (https://adafru.it/F2T)

Only use the Google Chrome browser with MakeCode!

Platagotchi

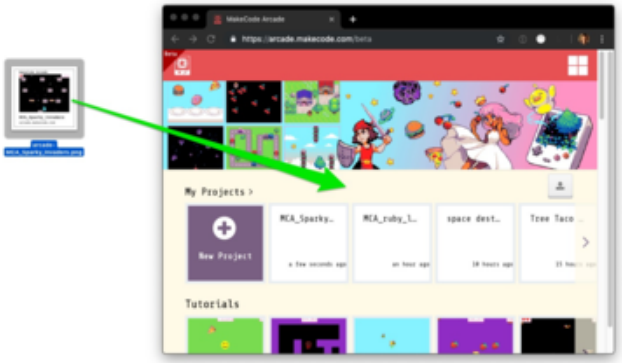
To start, let's load the code and take a look at how it works.

Start by launching MakeCode Arcade (https://adafru.it/DCY) using the Google Chrome web browser. Then, download the **led_strips_arcade-PyGamer-NeoPixel-Controller.png** image cartridge file above by right-clicking on the image and saving it to your computer.

Load the Code

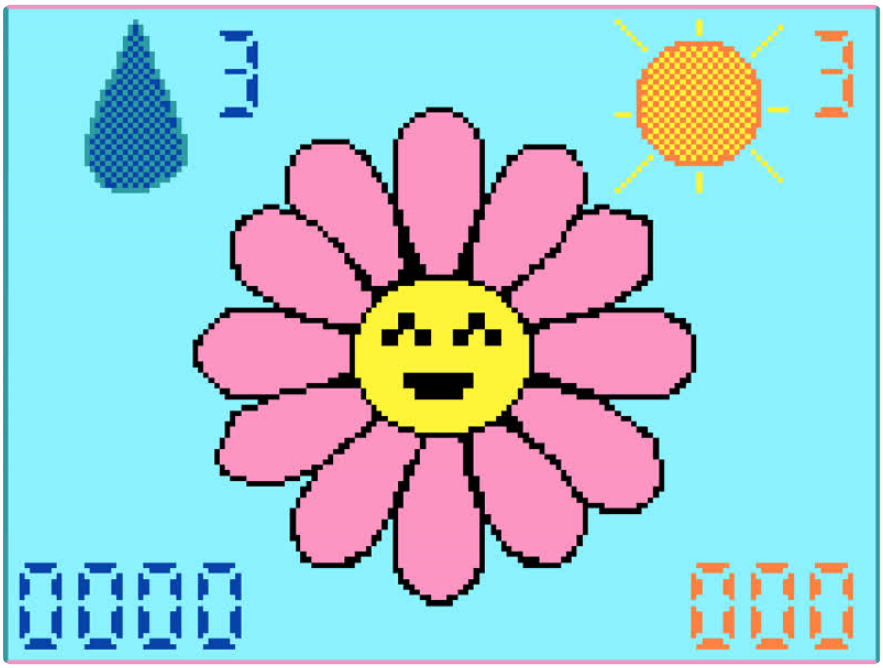
This is a special .png file that contains not only an image, but the entire game is embedded in it as well!

Simply drag it from the location to which you saved the image on your computer (such as the desktop as shown here) onto the Chrome browser window that is already running MakeCode Arcade (MCA). Note that the image in this graphic is of a different game, but you'll be dragging the **Plantagotchi.png** file.



This will open the code into the MCA editor.

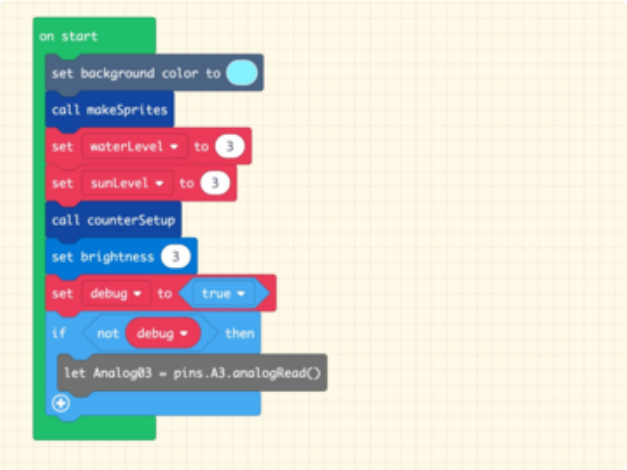
If you're ever unsure where a MakeCode block comes from, you can often find it by matching the block's color to a category on the left side of the editor. You can also use the handy search function!

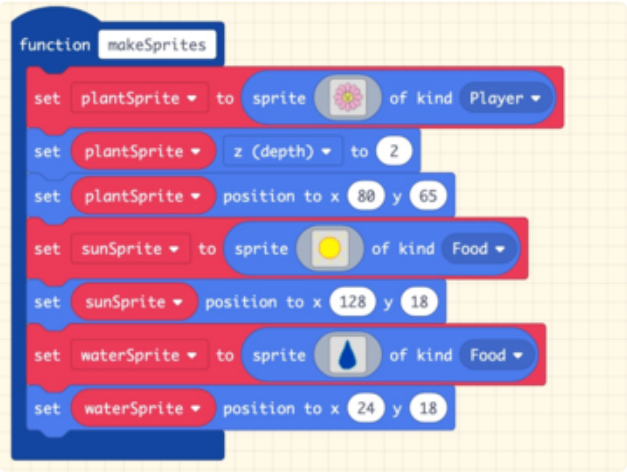


Plantagotchi Application Design

We'll take a look now at how the application works. When you load it into MakeCode Arcade, the first thing you want to do is find the **on start** block and turn the **debug** variable to **true.** This allows us to run the program in the simulator window. Since the program connects to real, physical sensors, the simulator can't run the program until we disable some of the features with the **debug** switch.

NOTE: You can learn more about the idea of using a debug control switch here:





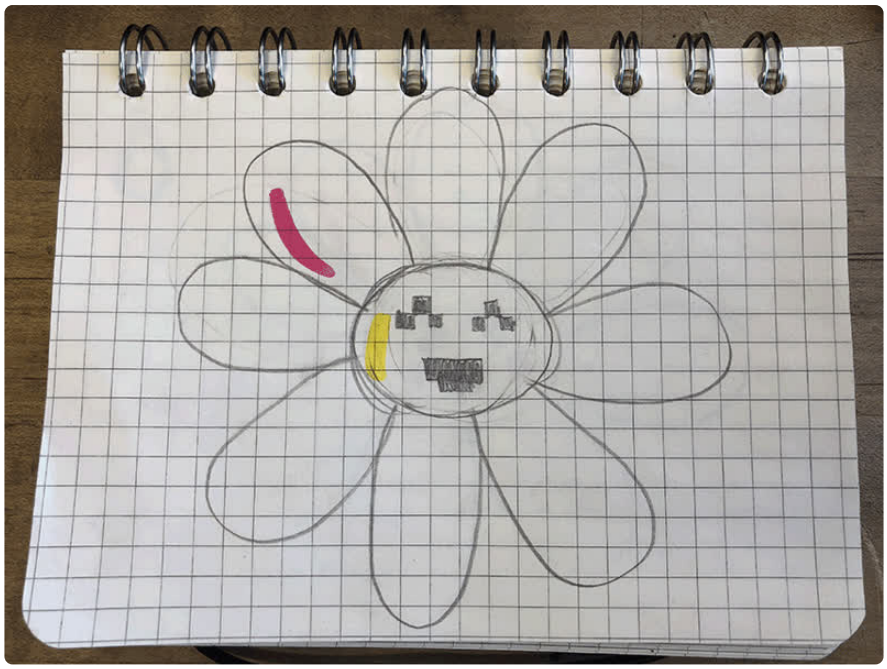
Startup

Here's what happens when we start up the program. First, we **set background color** to light blue. Then, we **call** the **makeSprites** function where we set up our **plantSprite** and the two icons for the corners of the screen, **sunSprite** and **waterSprite.**

My daughter designed the super-cute graphics. She drew them first on paper and then we worked together to translate them into pixel art!

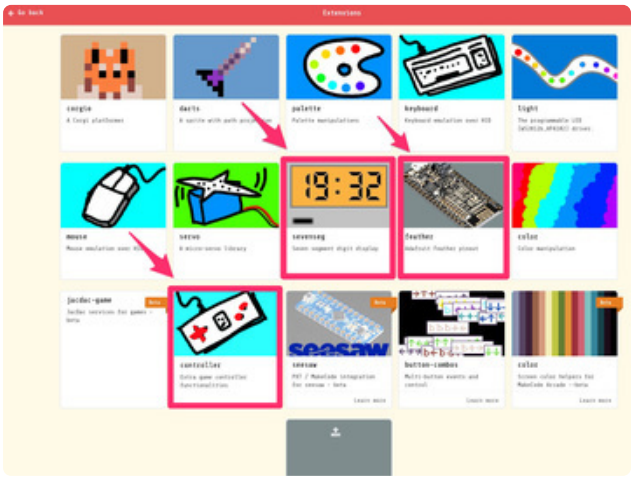
We'll set up a couple of variables to use later, **waterLevel** and **sunLevel.**

The **call counterSetup** block runs the **counterSetup** function which uses the Seven Seg extension to place number readouts in the four corners of the screen.



Extensions

This program makes use of some extensions to MakeCode Arcade: **Feather**, **Controller**, and **SevenSeg.**

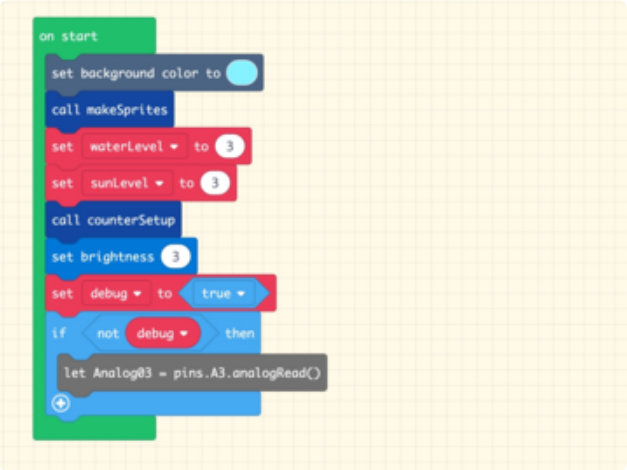


Feather allows us to use the many digital and analog IO pins available on the PyBadge Feather header -- this is how we'll read the analog values of the soil moisture sensor.

Controller will allow us to use the on-board photocell light sensor.

SevenSeg gives us the ability to display numeric values on screen.

If you're starting from scratch, Head to **Advanced** > **Extensions** + and then click on those two extensions to add them to your session. If you're opening the provided **.png** image cartridge there's no need, they're already added to the session.



on start Continued

Continuing on with the **on start** block, we'll set the **brightness** level for the onboard NeoPixels to something pretty low, in this case **3.**

Next is our **debug** variable, which we have set to a boolean **false** value. This can be flipped to **true** to use debug mode as mentioned above.

In fact, here's the first case of it that we'll see -- the **if not debug** conditional loop will only run the code contained within it if **debug** is **true.**

This is so that we can run the program in the simulator without making it throw an error message that it can't access the **Feather** analog **A3** pin.

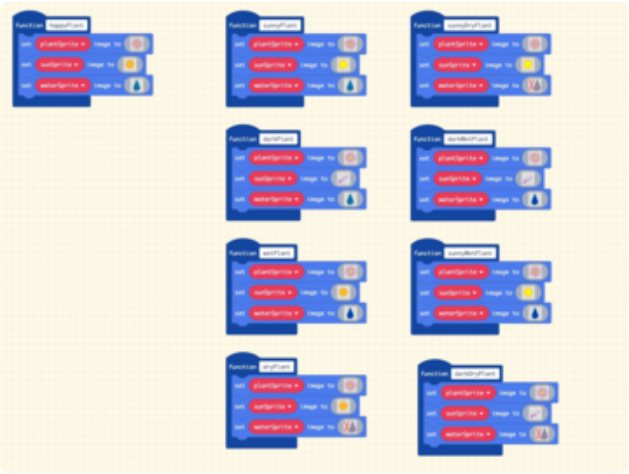
A Touch of JavaScript

The Feather extension gives us access to the PyBadge's Feather header pins, however the **Pins** blocks category is currently in beta testing. So, in the meantime, we can access those pins via the **JavaScript** tab. By clicking on the JavaScript tab at the top of the MakeCode Arcade browser window, we'll switch modes where all of the blocks are now represented by typed code.

This line gives us the functionality we need to instantiate the **A3** pin for analog reads:

```
let Analog03 = pins.A3.analogRead()
```

You can then switch back to Blocks mode by clicking the button at the top of the browser window.



Sprite Change Functions

You can think of the Plantagotchi as a machine that reads inputs for water levels and lights level, and then displays an appropriate set of graphics based on those readings. This set of functions is what we'll use to display the proper graphic.

For example, when the light level is 3 and the water level is 3, the default happy face will be displayed. By calling on the **happyPlant** function, the **plantSprite** image will change to the happy face, and the **sunSprite** and **waterSprite** icons will switch to their neutral state graphics.

The full set is:

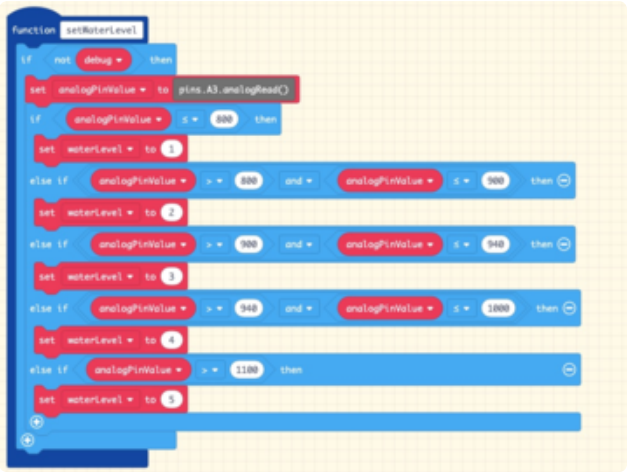
happyPlant
wetPlant
dryPlant
sunnyPlant
darkPlant
sunnyWetPlant
sunnyDryPlant
darkWetPlant
darkDryPlant

sunLevel

We'll use this **setSunLevel** function to remap the raw light readings into five bands. The **light level** block is the raw reading from the on-board light sensor, 0-255 values.

I put the PyBadge in different lighting scenarios to determine the different ranges for five **sunLevel** settings. So, anything less than 16 is total darkness (since there's some light leak from the screen's backlight it doesn't go down to 0), 16-25 is moderate dark, 25-50 is normal daylight in the middle of a room, 50-245 is on a window sill, and higher than 245 is direct sunlight. You may need to tune these values yourself.

In a moment we'll look at how we use both the raw values and the remapped ranges, but first, let's look at the **waterLevel** function.



waterLevel

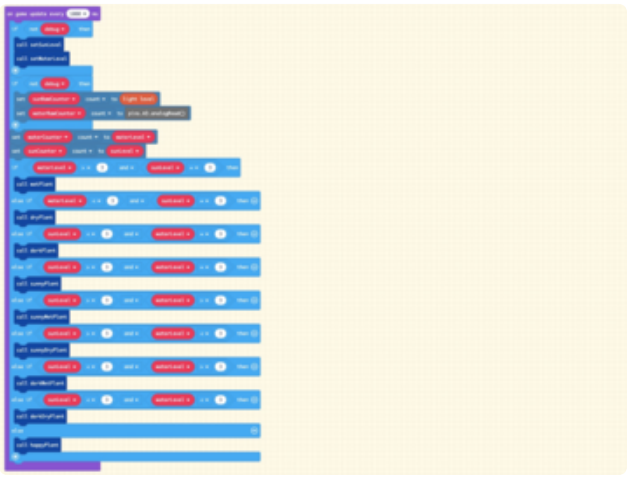
This works pretty much the same as the sunLevel function, except for reading the moisture sensor. One exception is that currently there isn't a block for `analogRead` of a Feather header pin (it should be coming soon!) so I went to JavaScript to type the line:

```
pins.A3.analogRead()
```

Again, I used some trial and error with very dry soil, moderately dry soil, average moisture soil, freshly watered soil, and overly wet soil to come up with practical value bands. The analog read values range from 0-1023, however my particular soil must have a lot of minerals in it, because even very dry soil was giving me a reading of nearly 800. Which is to say, your numbers may vary greatly depending on the dirt you use!

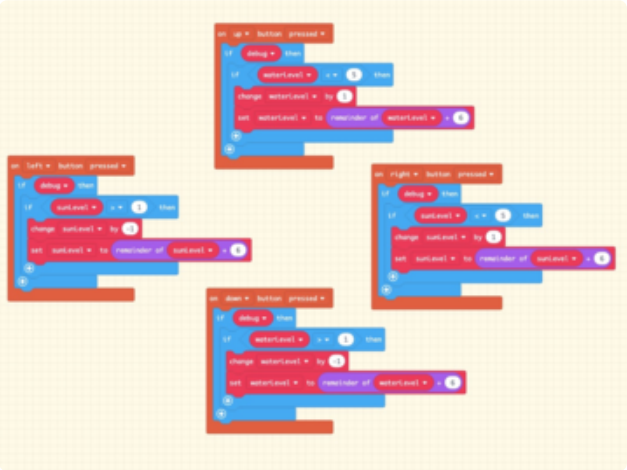
Do All the Things!

This is the block that does all the things! **on game update every 1000ms** will run every second and run through all of these steps:



check the **debug** variable state to determine if the **setSunLevel** and **setWaterLevel** functions will be called, thus getting the current sensor values, as well as updating the SevenSeg counters with those raw values
set the **waterCounter** and **sunCounter** to their respective remapped (1-5) values
Then, it will run through the big **if...else if** conditional block to determine which graphic set to display for the face and the two icons. For example, if **sunLevel** is greater than **3** and **waterLevel** is less than **3** the **sunnyDryPlant** function will be called, and we'll see those graphics appear. Probably time to water the plant and think about getting it out of the sun!

Buttons for Testing



When working with a project that uses sensors and displays graphics based on multiple possible conditions, it's useful to have a way to pretend each condition is being met to make sure the proper graphics show up (or whatever result you're expecting from each condition).

This is another way we'll use the **debug** mode. We've set up the d-pad buttons to increase and decrease the **waterLevel** and **sunLevel** values artificially! So, when **debug** is true, you can test the graphics in either the simulator or on the PyBadge after uploading the code just by pressing the buttons.

These have no function when you're out of debug mode and using the real sensor values.

NeoPixel Level Indicator

Since the PyBadge has a five NeoPixel strip built right onto it, it'd be a shame not to use it! Here, we'll use an **on game update every 5000ms** block to light up the NeoPixels with either the water level or light level every five seconds.

You can see how the **ledMode** variable is being used to flip flop back and forth between the two states every other five seconds.

Now, you can upload the code, connect up your probes to your potted plant, and try it out!

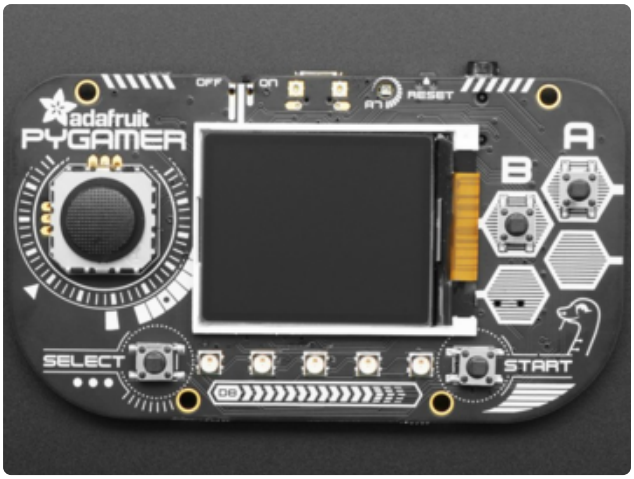
Update the PyBadge/PyGamer Bootloader

You are at the bleeding edge of handheld, open source, game playing hardware and software, what with your PyBadge/PyBadge LC or PyGamer! Congratulations! It's fun and exciting! It is also changing and improving all the time, so please update your bootloaders before proceeding to put your MakeCode Arcade games on the board!!

Among lots of other reasons, update the bootloader to prevent a problem with MacOS 10.14.4, to fix button problems, and get the thumbstick to work!

PyBadge/PyBadge LC Bootloader

If you have a **PyBadge** or **PyBadge LC**, please go to this page for instructions on updating the bootloader. (https://adafru.it/EWI)



PyGamer Bootloader

If you have a **PyGamer**, please go to this page for instructions on updating the bootloader. (https://adafru.it/EWJ)

A HUUUUUUGE number of people have problems because they pick a 'charge only' USB cable rather than a "Data/Sync" cable. Make 100% sure you have a good quality syncing cable. Srsly, I can't even express how many times people have nearly given up due to a flakey USB cable! Enter Alert Text...

Hardware Checks

If, after updating your board's bootloader, you still think you may have a hardware problem, here's a great way to test out all of the functions. From buttons, to the light sensor, thumbstick (PyGamer only), accelerometer (PyGamer and PyBadge only, not the LC), and more, we've got a super nifty set of hardware test .UF2 files you can use.

Click on the link for your board below for more info and a link to the appropriate UF2 file.

PyBadge/PyBadge LC Hardware Check

https://adafru.it/EWK

PyGamer Hardware Check

https://adafru.it/EWL

Another way to do a hardware check is with the handy, dandy MakeCode Arcade Basic Hardware Test. This was created with MakeCode Arcade and you can use it to check that your d-pad buttons or thumb joystick can move the yellow face around the screen, and that the A and B buttons work to play a sound (just make sure you have a speaker plugged in to the PyGamer first).

You can open this link (https://adafru.it/EWP) to get to it, or download the UF2 file below and drag it onto your board's USB drive in bootloader mode.

arcade-Basic-Hardware-Test.UF2

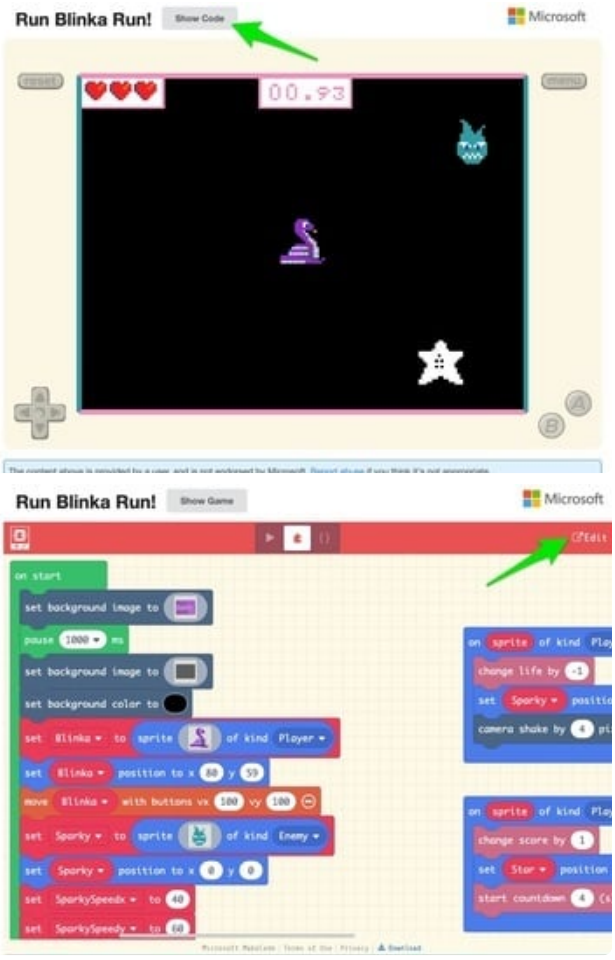
https://adafru.it/EWQ

Load a MakeCode Game on PyGamer/ PyBadge

Let's load a game! For example, here's a link to **Run, Blinka, Run!** To open the game in the MakeCode Arcade editor, first, click the share link below. This will allow you to play the game in the browser right away.

Makecode Arcade Game: Run, Blinka, Run!

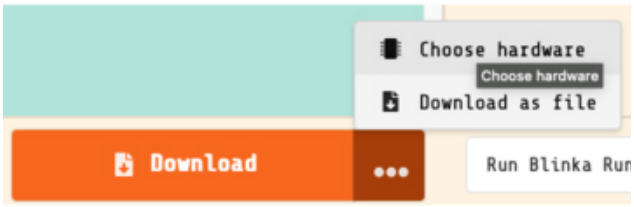
https://adafru.it/Fqf



Then, click on the Show Code button in the upper left corner. The shows the code for the game, and by clicking the Edit button in the upper right corner, it'll open into the editor where you can upload it to your PyGamer/PyBadge.

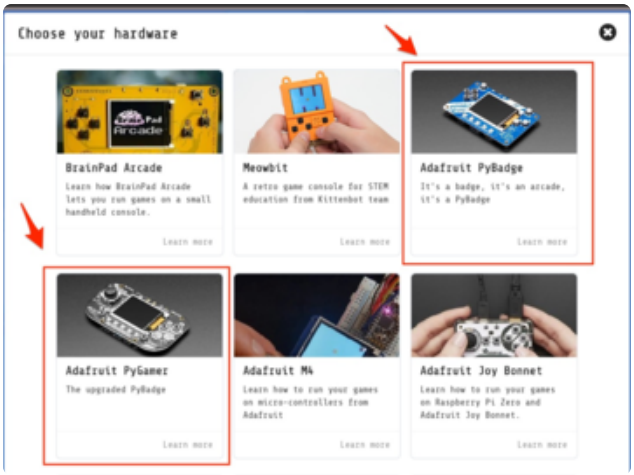
Once you have a game working on the MakeCode Arcade web editor, it's time to download it and flash it onto your board.

Please only use the Google Chrome browser with MakeCode! It has WebUSB support and seems to work best



Board Definition

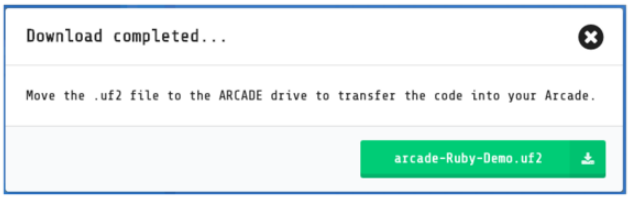
In order to load a game made in MakeCode Arcade onto the PyBadge, first choose the proper board definition inside of MakeCode. Click the ellipsis (...) next to DOWNLOAD and then the **Choose Hardware** item.



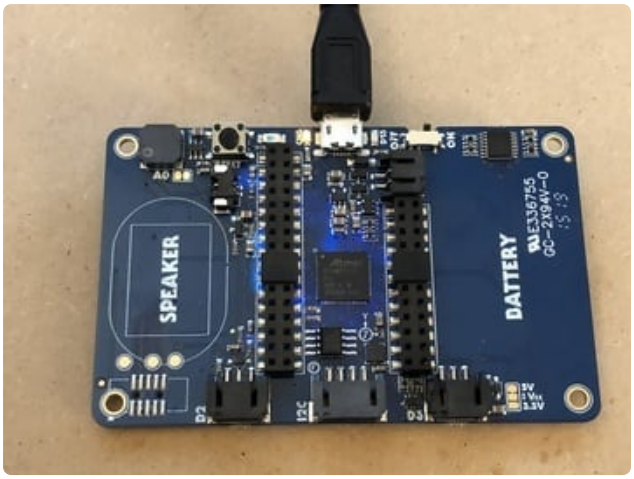
Change Board screen

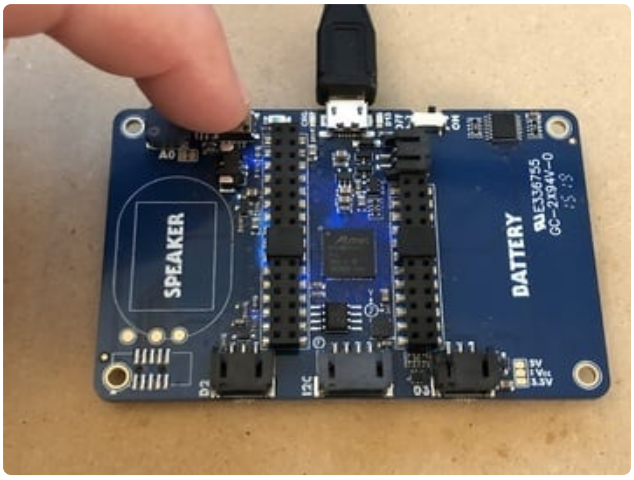
Click on the image of your board, either the PyBadge/PyBadge LC or the PyGamer

This will cause the game .uf2 file for your particular board to be saved to your hard drive. You only need to do this the first time you use a new board. Thereafter you can simply click the **Download** button on the MakeCode Arcade editor page.



A HUUUUUUGE number of people have problems because they pick a 'charge only' USB cable rather than a "Data/Sync" cable. Make 100% sure you have a good quality syncing cable. Srsly, I can't even express how many times people have nearly given up due to a flakey USB cable!



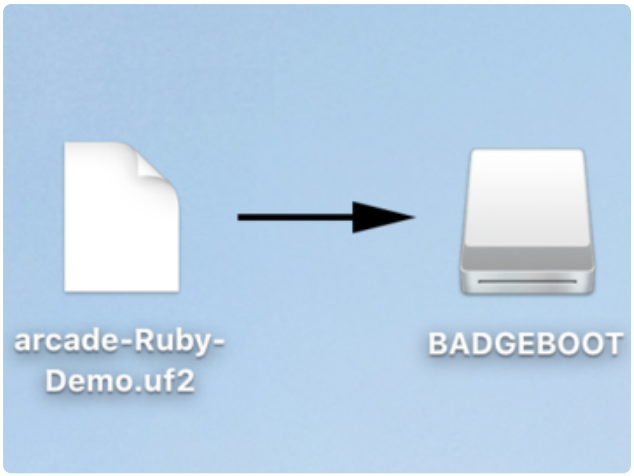


Bootloader Mode

Now, we'll put the board into bootloader mode so we can drag on the saved .uf2 file. On the back side of the board you'll see a reset button at the top. Make sure the board is plugged into your computer via USB with a USB micro B to A data cable. Also, be sure the board is turned on.

Then, press the reset button. This will initiate bootloader mode.

When the board is in bootloader mode you'll see a screen similar to this one show up.

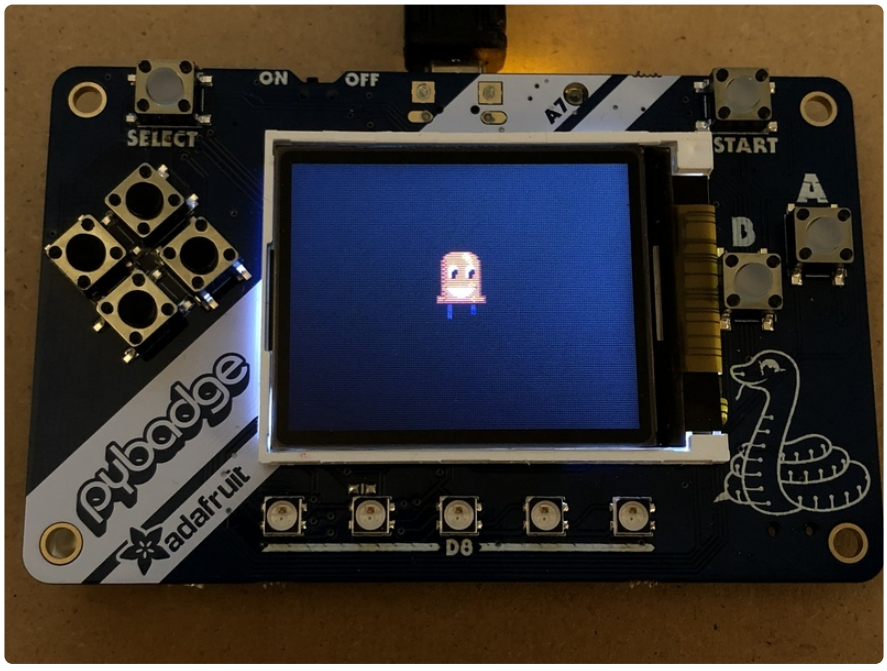


Drag and Drop

Now that the board is in bootloader mode, you should see a **BADGEBOOT** drive show up on your computer as a USB flash drive. Simply drag the arcade game .uf2 file onto the drive.

Play!

That's all there is to it! Once the file is copied over the board will restart and launch the game!



Keep an eye on Adafruit.com for additional game related content.

Troubleshooting MakeCode Arcade

If you run into trouble with MakeCode Arcade, here are some resources for getting help:

- Microsoft MakeCode Arcade Forum (https://adafru.it/EXI)
- Adafruit MakeCode Forum (https://adafru.it/EXJ)

- Microsoft MakeCode Arcade Discord (https://adafru.it/EXK) -- look for the #arcade channel
- Adafruit MakeCode Discord () -- look for the #makecode channel

Only use the Google Chrome browser with MakeCode!