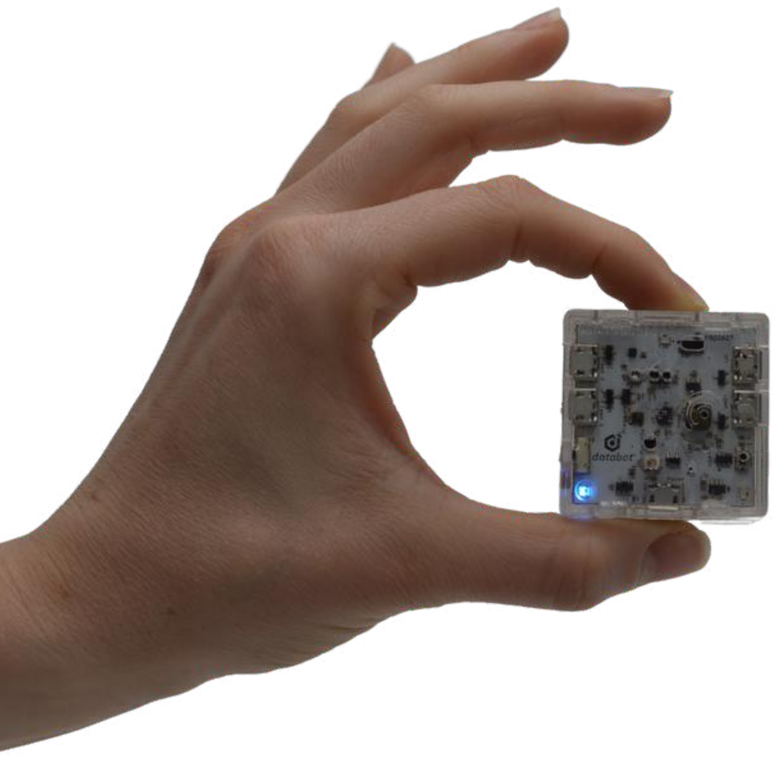
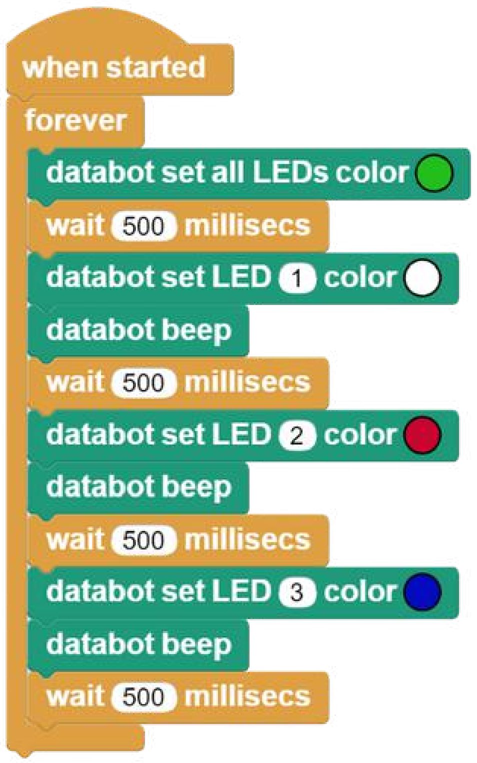
Getting Started with databot™ & MicroBlocks®





```
when started
forever
    databot set all LEDs color 🟢
    wait 500 millisecs
    databot set LED 1 color ⚪
    databot beep
    wait 500 millisecs
    databot set LED 2 color 🔴
    databot beep
    wait 500 millisecs
    databot set LED 3 color 🔵
    databot beep
    wait 500 millisecs
```

Blink & Beep if you love physical computing!

Draft Version 04.21.2023

Table of Contents

databot™ & MicroBlocks®

Background

MicroBlocks is a blocks programming language for physical computing inspired by Scratch. It runs on microcontrollers such as the micro:bit, Calliope mini, AdaFruit Circuit Playground Express, and many others including databot™!

MicroBlocks has some outstanding features for physical computing educators as the code is extremely portable, multi-platform, easy-to-use, and best of all, super fast! Long compile times and download fatigue present significant challenges to educators trying to maintain student engagement. MicroBlocks is a live environment and solves this challenge beautifully. Click on a block and it runs immediately, right on the board. Try out commands. See and graph sensor values in real time. No more waiting for code to compile and download. This document introduces basic MicroBlocks use in the context of databot. Visit the MicroBlocks website for complete documentation here: https://learn.microblocks.fun/en/other-resources

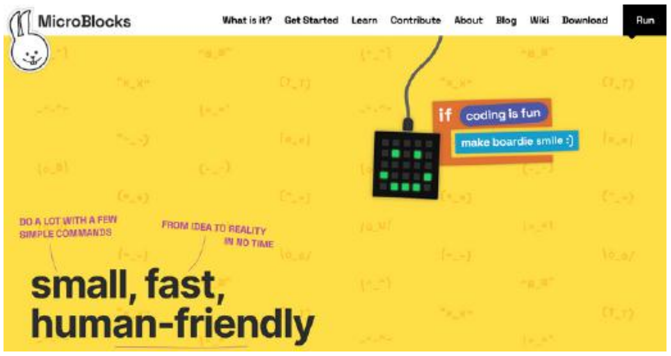
What You Will Need/Prep

It's not going to get much simpler than this for physical computing - databot has lights, sound, a multitude of sensors, wi-fi, and internal storage all of which can be controlled by MicroBlocks!

- databot 2.0
- MicroUSB Cable
- PC, Mac, or Chromebook
- Internet Connected Browser
- MicroBlocks - runs in the browser!

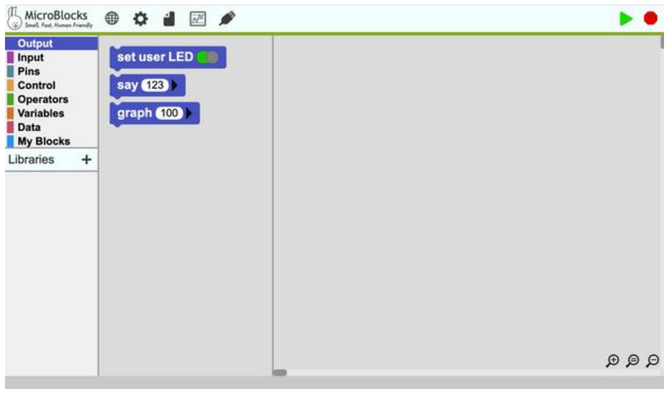
Setup & Installation

Three easy steps and you will be up and coding with databot and MicroBlocks in no time! The following instructions are for the browser based version (recommended for simplicity).


MicroBlocks
What is it? Get Started Learn Contribute About Blog Wiki Download Run
if coding is fun
make boardie smile :)
DO A LOT WITH A FEW SIMPLE COMMANDS
FROM IDEA TO REALITY IN NO TIME
small, fast, human-friendly

<u>Launch MicroBlocks</u>

- Open an Internet connected Chrome or Edge Browser.
- Go to the MicroBlocks Pilot Library (databot™ libraries will be found in the full release version sometime in Q2 2023).
    - https://microblocks.fun/run-pilot/microblocks.html
- Your MicroBlocks coding environment will load as shown to the right.
- Notice the color-coded block coding categories in the left hand menu, similar to Scratch.

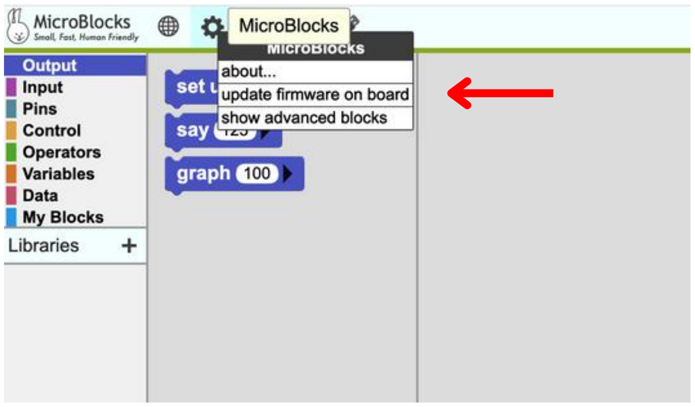


Load the Firmware!

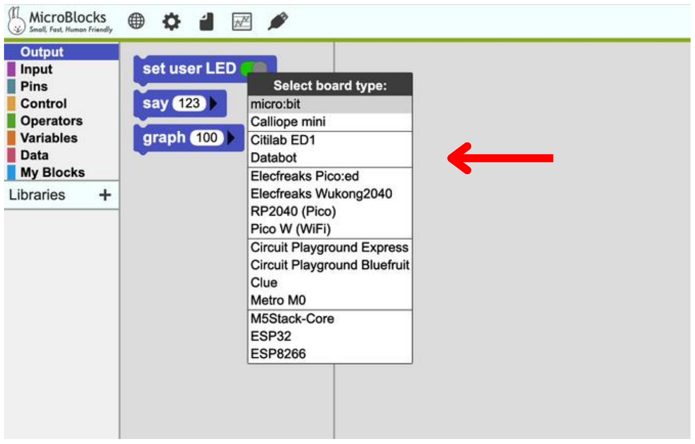
databot comes with a standard firmware installed that enables connection to Vizeey™, server mode, and other standard operations. To enable coding with MicroBlocks install the MicroBlocks™ firmware as follows.

Note: To return to your normal Vizeey mode you will need to restore the databot™ firmware. You can do this here: https://databot.us.com/firmware

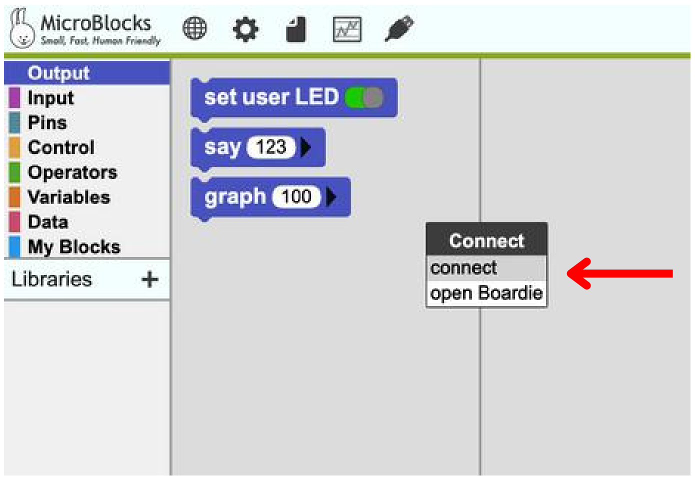
- Connect your databot™ to an open USB port.
- Turn it on.
- Go to the "gearwheel" MicroBlocks™ menu.
- Select "update firmware on board"



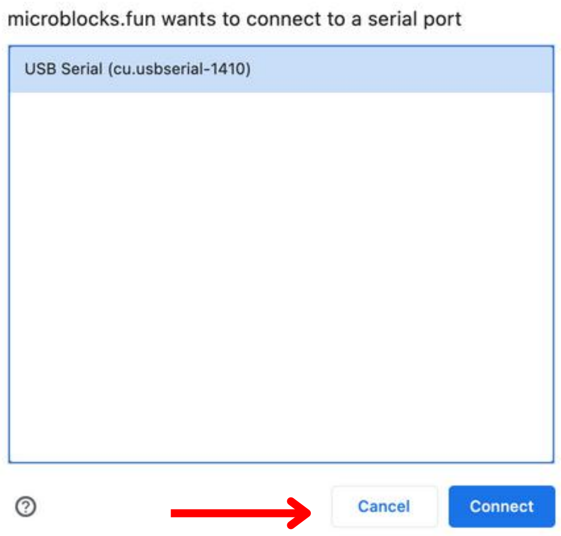
- Select databot



- Select "connect"

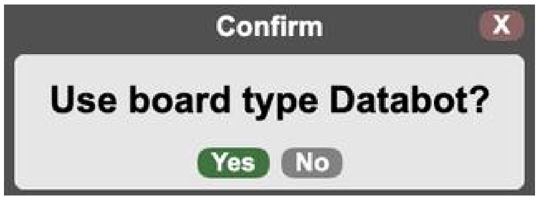


- MicroBlocks will prompt you to connect. You should see a USB serial device displayed. Select it and select connect.
- Troubleshooting: If no serial ports appear in the Connect dialog, make sure that the databot is connected to your laptop with a good USB cable (not a power-only one) and is turned on.


microblocks.fun wants to connect to a serial port

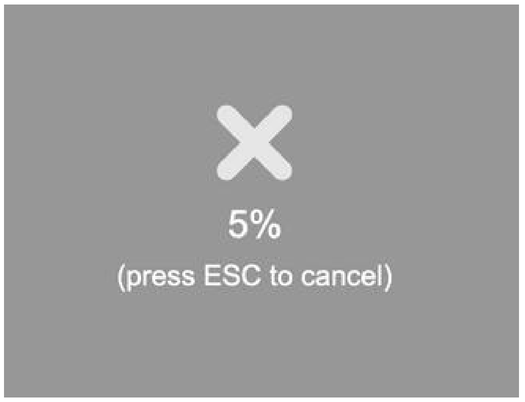
USB Serial (cu.usbserial-1410)

- Confirm the board type as databot

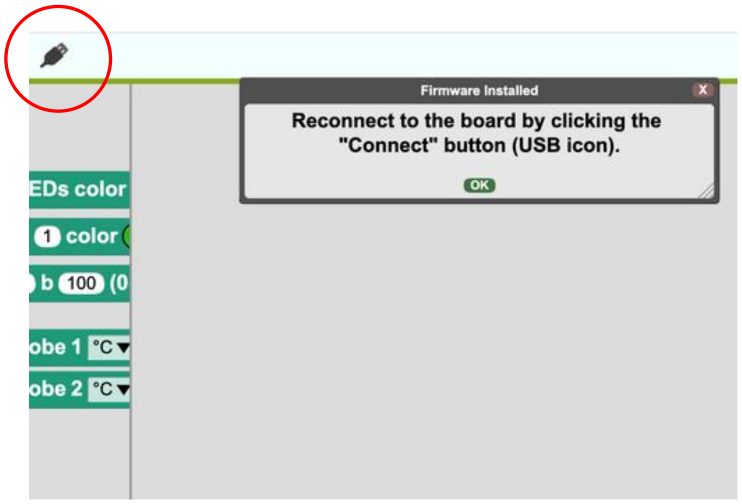

Confirm

Use board type Databot?

- Wait for the installation process to complete.

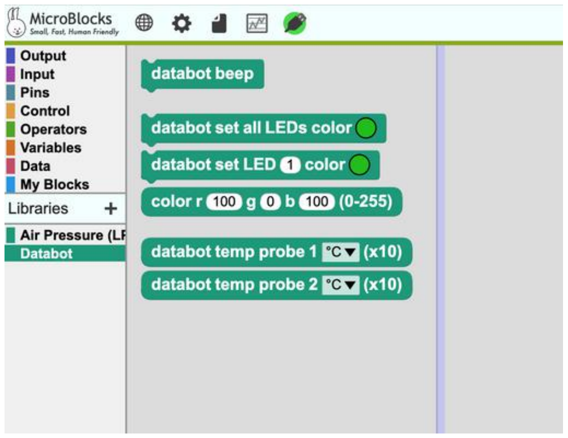

5%
(press ESC to cancel)

Connect and Prepare to Code!


Firmware Installed

Reconnect to the board by clicking the "Connect" button (USB icon).

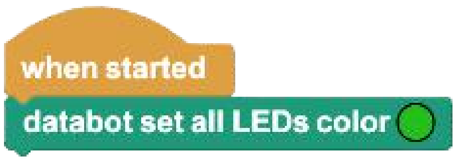
- Select OK
- Reconnect using the USB icon.
- Note: Now that the firmware is installed simply click the USB icon to connect.


MicroBlocks
Small, Fast, Human Friendly
Output
Input
Pins
Control
Operators
Variables
Data
My Blocks
Libraries
Air Pressure (LP
Databot
databot beep
databot set all LEDs color
databot set LED 1 color
color r 100 g 0 b 100 (0-255)
databot temp probe 1 °C (x10)
databot temp probe 2 °C (x10)

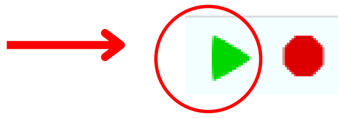
- When connected the USB icon will turn green and the default libraries and commands for databot™ will load and display.
- You are connected and ready to code!

First Program - Blink

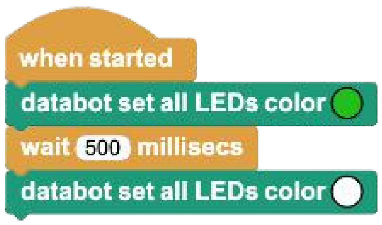
- Begin with the orange **Control** category and browse the various blocks that can start, stop, and control your program.
- Drag the "when started" block into the scripts pane as shown.
- Select the **databot** library and add the databot "set all LEDs color" block.
- Start the program using the green play button in the upper right hand corner. Your databot™ will light up in green! Use the start arrow anytime to start your program.



```
when started
databot set all LEDs color 🟢
```

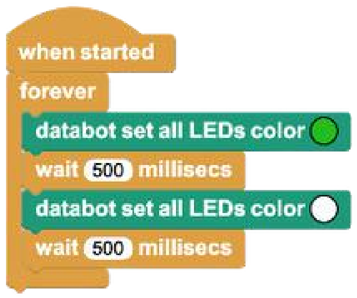


- Let's add a second color. Since your program will execute instantly, you need to add a pause between blocks, otherwise it will flash your initial color so quickly you probably won't see it. Add a wait block from the **Control** blocks as shown.
- Add a second color. In this example databot displays green, waits a half a second, turns white, then the program ends.



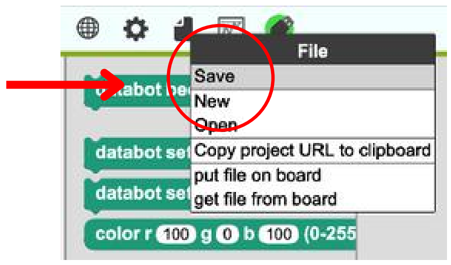
```
when started
databot set all LEDs color 🟢
wait 500 millisecs
databot set all LEDs color ⚪
```

- Time to blink! Since we don't want our program to stop, rather, we want it to repeat and blink, we need to add a "forever" block from the **Control** category that will cause all the blocks inside it to repeat indefinitely.
- Note the additional wait block has been added. Why?
- Start your program, experiment by changing the wait times and the colors until you have the perfect blinking databot.
- Congratulations - you have taught databot to blink!



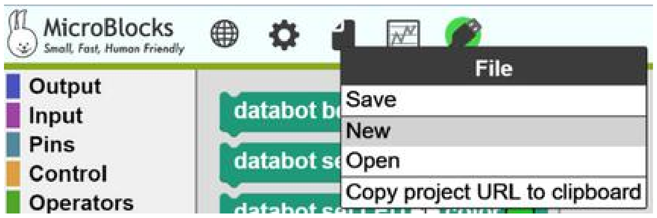
```
when started
forever
    databot set all LEDs color 🟢
    wait 500 millisecs
    databot set all LEDs color ⚪
    wait 500 millisecs
```

- Save your masterpiece! Now is a good time to plan for organizing your coding projects. From the File menu, select Save to save Blink to your device. MicroBlocks does not "autosave" so it is up to you to keep your code safe. Good coding habits include keeping your files safe and organized!

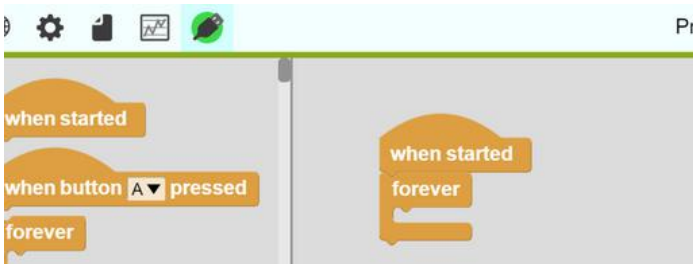


Program 2 - Blink and Beep!

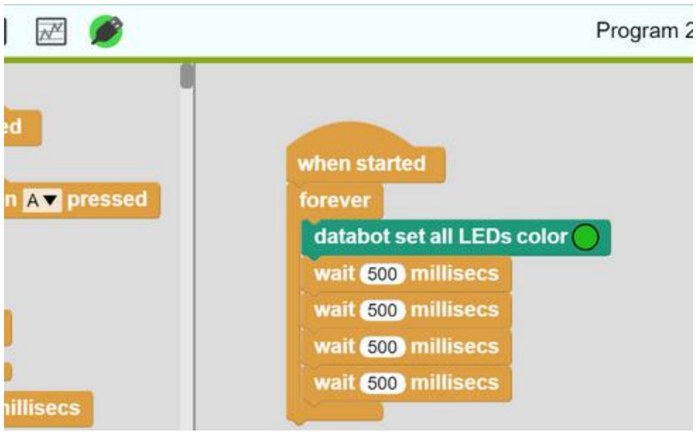
- Go to the “File” menu and select “New” to start a new project.



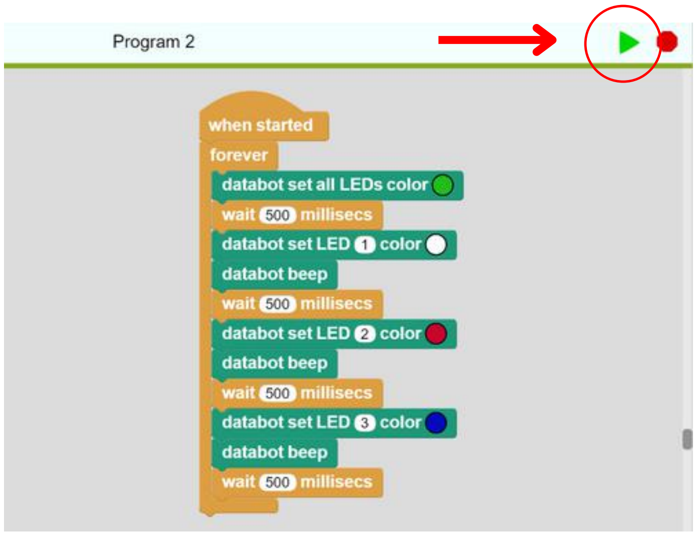
- Begin with the orange "control" menu and drag the "when started" block into the scripts pane.
- Add a forever block so that your program will run indefinitely.



- Select the databot library and add the databot™ set all LED color command. The default is green but you can select any color you want.
- Time to blink and beep! Return to the orange control menu and drag four wait blocks onto your stack. The default 500 milliseconds is half a second. These wait blocks will be the time each blink color is held before moving to the next color in the program.



- From the databot library, drag set LED (1) block color and beep blocks and place them between the wait commands.
- Modify the default number to 1, 2, 3, and the default green color to any color you want.
- Start the program using the green play button in the upper right-hand corner. Your databot™ will light up green, and change the color of LEDs one by one with a beep sound!



Program 3 - Meet Roy G Biv!

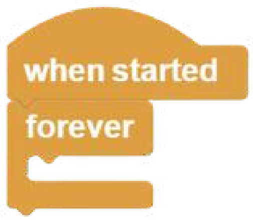
Background

Roy G Biv is a well know mnemonic, a memory aid, to recall the colors of the visible light spectrum: **R**ed, **O**range, **Y**ellow, **G**reen, **B**lue, **I**ndigo, **V**iolet! If you remember your friend Roy G. Biv, you will always remember the colors of the visible light spectrum. In this activity you will learn how to use the RGB color block to set databot™ LED colors. When using RGB values in coding, the following information will be very useful:

- RGB stands for **R**ed, **G**reen, and **B**lue. RGB colors are added together to form a new color.
- Each parameter (red, green, and blue) defines the intensity of the color with a value between 0 and 255.
- For example, r(255)g(0)b(0) is displayed as red, because red is set to its highest value (255), and the other two (green and blue) are set to 0.
- To display black, set all color parameters to 0 and to display white, set all color parameters to 255.

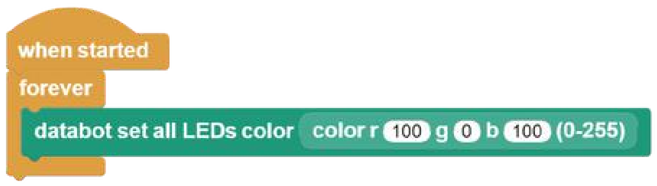
The Code

- Go to the "File" menu and select "New" to start a new project.
- Begin with the orange **Control** category and drag the "when started" block into the scripts pane as shown.
- Add a forever block so that your program will run indefinitely.



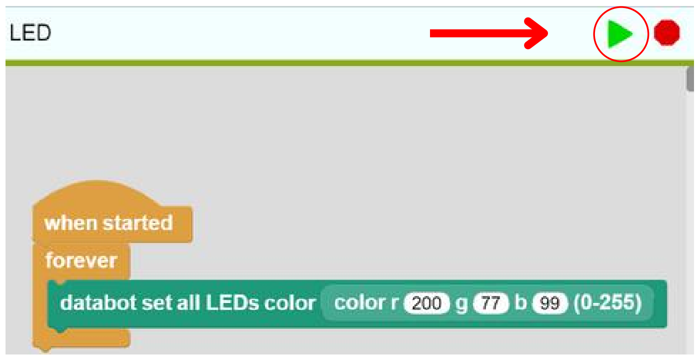
```
when started
forever
```

- Select the **databot** library and add the databot set all LED color command. The default is green but let's create a new color by using the color rgb block.
- Drag the color rgb block and place it in the place of default green color as shown.



```
when started
forever
  databot set all LEDs color  color r 100 g 0 b 100 (0-255)
```

- Time to create new color! Modify the rgb values as you wish but remember the intensity range from 0 to 255.
- Start the program using the green play button in the upper right-hand corner. Your databot™ will light up in your new color choice!
- Tinker with the colors while the program is running by editing the color values and hitting return or clicking outside the block. You will see an instant response as the color updates to the new value.



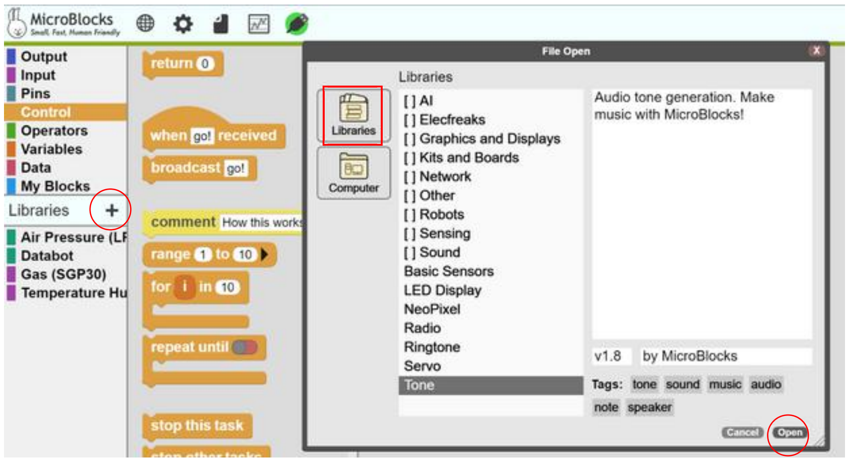
```
when started
forever
  databot set all LEDs color  color r 200 g 77 b 99 (0-255)
```

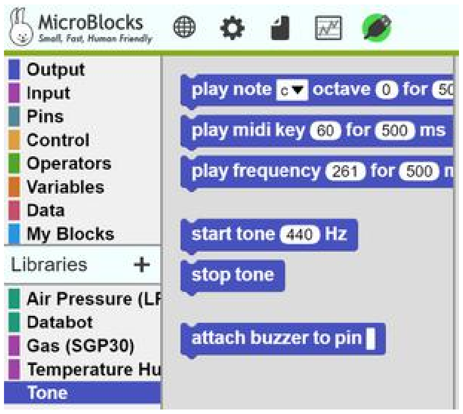
Program 4 - Tone up!

Is it possible to add tones and play music with your databot? Yes! A tone library can be used to play tones using the piezo buzzer that is built in to databot. By varying the voltage to the buzzer you can play notes and specific tones. Let's try it!

Step 1: Add the Tone Library

- MicroBlocks has a built in Tone Library that works with databot™ and many other microcontrollers. Go to the **Libraries +** section, select the **Libraries** icon, and choose the **Tone** library as shown to the right. Open the library!



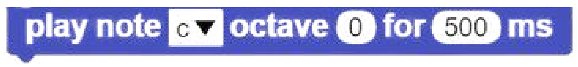


- Once the library is added you can see Tone blocks as shown in your library collection.

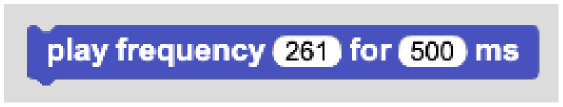
Step 2: New Tone Blocks!

You will use two new blocks in this code. One plays sounds using notes and the second plays sounds using frequencies. Let's take a look at them both and use them in a program.

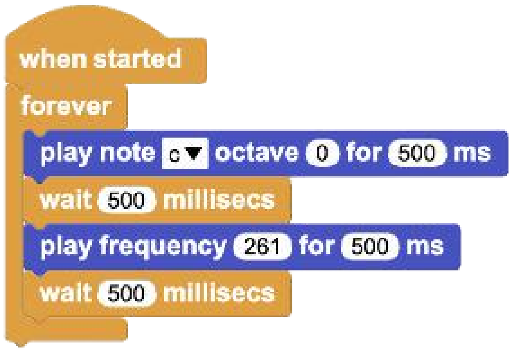
- The play note block enables you to play tones using traditional notes and octaves. You can actually take notes from sheet music and build a song using MicroBlocks and databot's tone generator!



- The play frequency block enables you to play tones defined by frequency in Hertz (Hz). Middle C is about 261 Hz so both these blocks will play the same tone!



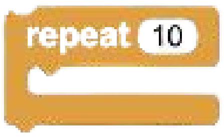
- Begin with a when started block from the **Control** category.
- Add the play note and play frequency blocks to a forever loop so you can experiment.
- There are three fields to set for play note block: the given note in the given octave for the given number of milliseconds.
- There are only two fields for the play frequency: the frequency (in Hz) and the time in milliseconds.
- Start your program and listen.
- Challenge: Since the two commands sound identical, how could you change your code so that you can identify which is which?
- Save this program if you'd like to keep it!



```
when started
forever
    play note c octave 0 for 500 ms
    wait 500 millisecs
    play frequency 261 for 500 ms
    wait 500 millisecs
```

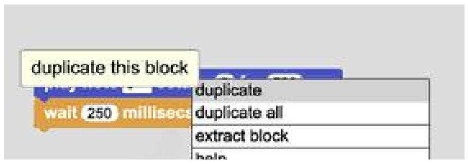
Step 3: Play It Again Sam

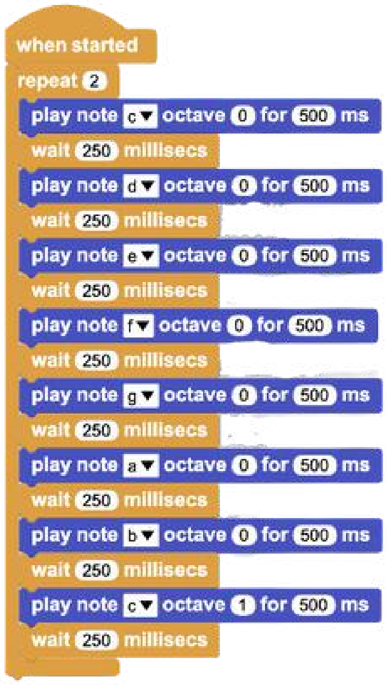
- Start a new program and begin with the when started block from the **Control** category.
- This time, instead of a forever loop we will use a "repeat" block. This block, instead of looping forever gives you the control to repeat an action an exact number of times.
- Drag in a play note block and a wait block and connect them together.
- Right click on the two and duplicate four times to create a code stack like the one shown.



```
repeat 10
```

Meet "Repeat"



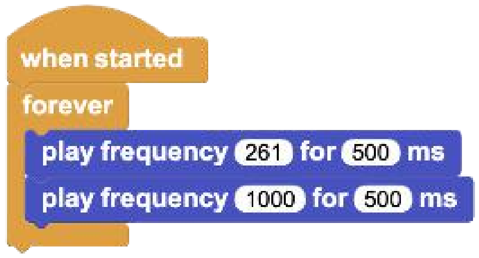


```
when started
repeat 2
    play note c octave 0 for 500 ms
    wait 250 millisecs
    play note d octave 0 for 500 ms
    wait 250 millisecs
    play note e octave 0 for 500 ms
    wait 250 millisecs
    play note f octave 0 for 500 ms
    wait 250 millisecs
    play note g octave 0 for 500 ms
    wait 250 millisecs
    play note a octave 0 for 500 ms
    wait 250 millisecs
    play note b octave 0 for 500 ms
    wait 250 millisecs
    play note c octave 1 for 500 ms
    wait 250 millisecs
```

- Play with your new code stack by changing the notes and timing.
  - Change the delays
  - Change the play time
  - Change the notes
- Can you build a song with these eight notes?

Go Further - Sound the Alarm!

Using the play frequency block, experiment with this simple program and try to create a sound pattern as close to an ambulance or police siren possible.



```
when started
forever
  play frequency 261 for 500 ms
  play frequency 1000 for 500 ms
```

Program 5 - Let's Talk Variables!

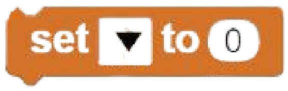
What is a Variable?

A Variable is a container for storing values in computer memory. Once the variable is created, it can be used and updated multiple times.

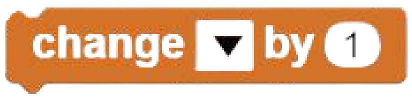
Imagine a football game, where the **score** is a variable that is updated multiple times when the player places a goal. Setting an initial value for the variable and updating it are the two important concepts. For example, when the game starts the score should be 0, whenever the player places a goal the score should change by 1 (increment), and whenever the player fouls the score should change by -1 (decrement).

Step 1: New Blocks!

- You will use two new blocks in this code from the Variables section. One is to set an initial value for the variable when the program begins and the second is to increment or decrement the value of the variable.
- The first block is used to set the value of the variable. This block is often used when the program starts, but it can also be used to change the value of the variable at any time while a program is running.
- The second block is used to change the value of the variable either by increasing or decreasing it.



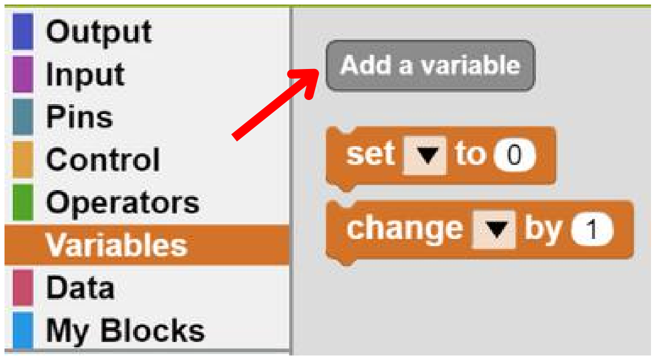
```
set ▼ to 0
```



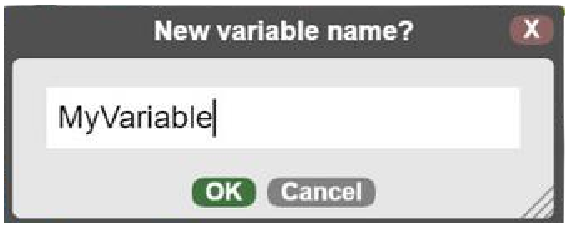
```
change ▼ by 1
```

step 2: Create a Variable!

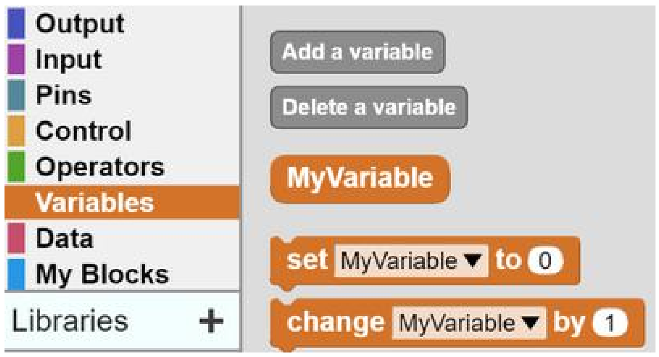
- When naming variables:
    - Make it unique
    - Give it a name that clearly states its purpose
- Go to the “File” menu and select “New” to start a new project.
- Go to the **Variables** menu and click the “Add a variable” button to create a new variable.



- Type the name of your variable and select “ok”.

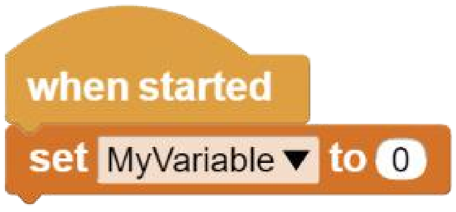


- Once the variable is created you can see it in the **Variables** blocks. And you can also delete the variable if required using the ‘Delete a variable’ button.

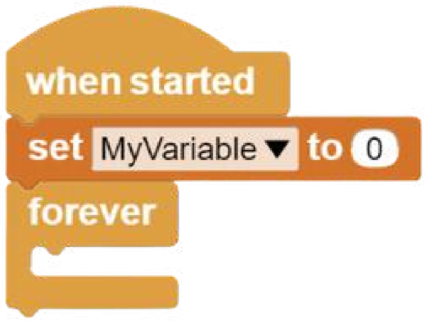


Step 3: Let's Code!

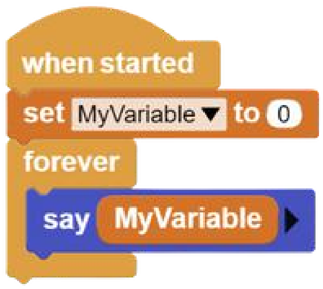
- Begin with the orange **Control** menu and drag the "when started" block into the scripts pane.
- Go to the “variables” menu and set the initial value of the variable to 0.



- Switch to the Control menu and add a forever block below the set block so that your program will run indefinitely.

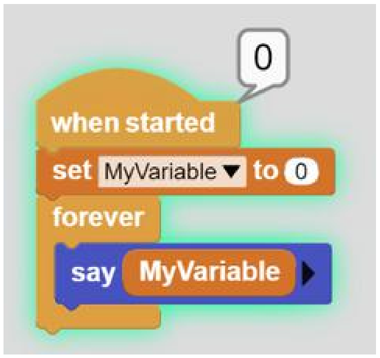


- To see our variable value, we need some kind of output that will display it. Let's do that next.
- Select the **Output** menu and drag the "say" block into the "forever" loop. The default value is 123 but you can change it to the variable name by dragging and dropping the named variable block in the **Variables** tab into the say window.



```
when started
set MyVariable to 0
forever
    say MyVariable
```

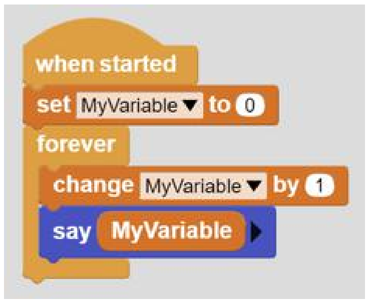
- Start the program using the green play button in the upper right-hand corner. You will see the value of the variable as the output.



```
0
when started
set MyVariable to 0
forever
    say MyVariable
```

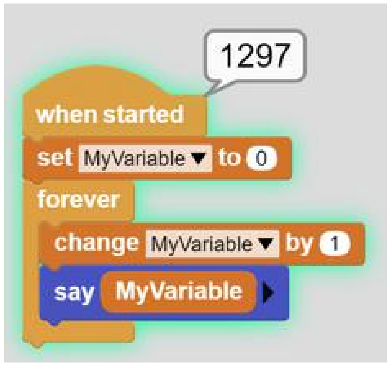
Step 4: Increment!

- Let's increase the value of the variable using the change block and display its live value in the output.
- Go to the "variables" menu and drag the change block and place it before the say block.



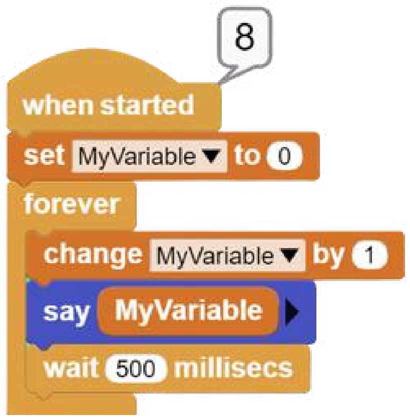
```
when started
set MyVariable to 0
forever
    change MyVariable by 1
    say MyVariable
```

- Start the program using the green play button in the upper right-hand corner. You will see the live incrementing value of the variable increasing rapidly.



```
1297
when started
set MyVariable to 0
forever
    change MyVariable by 1
    say MyVariable
```

- Slow down the increment by adding a wait block inside the loop from the **Control** menu.
- Start the program using the green play button in the upper right-hand corner. You will see the live incrementing value of the variable now increasing gradually.

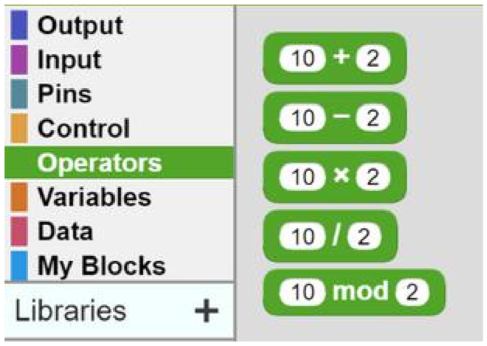


Program 6 - Operators!

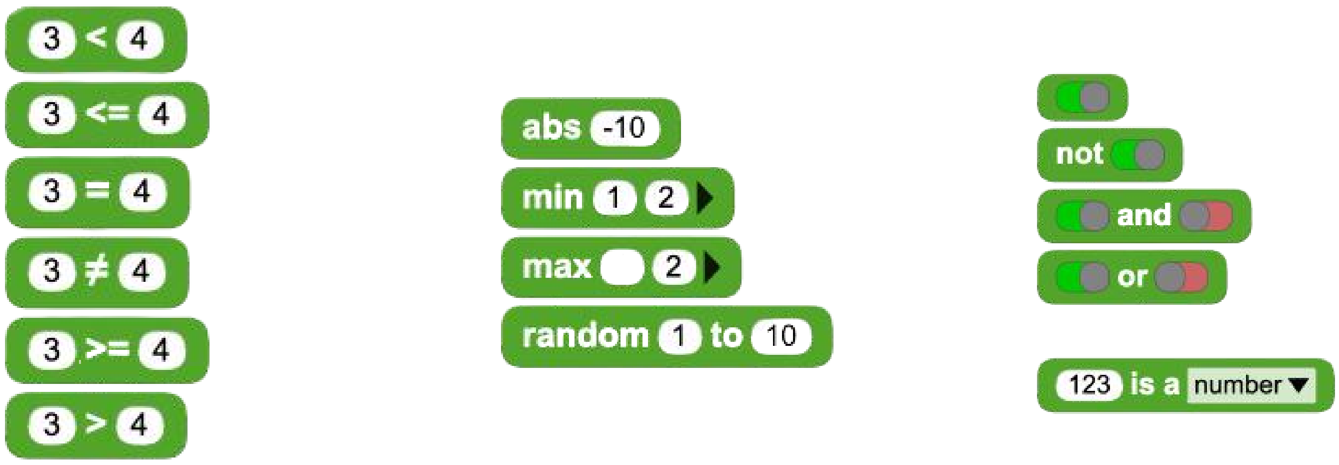
What is an Operator?

Operators are an important part of your coding as this is where you bring in the math and the decision making capabilities required for cool physical computing to take place.

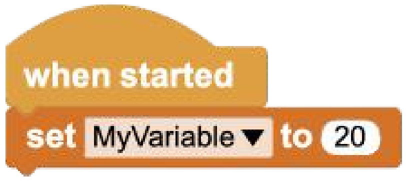
- You can perform basic arithmetic operations like addition, subtraction, multiplication, division, and modulo operations using the blocks available in the **Operators** menu.



- In addition, you will notice a number of options for decision making and other numerical operations in this area. Let's try it out.

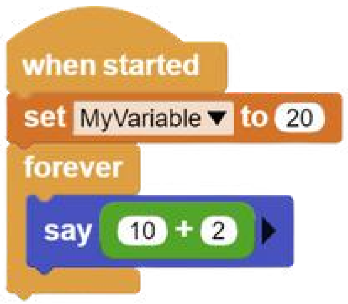


- Go to the “File” menu and select “New” to start a new project.
- From the **Control** blocks drag the "when started" block into the scripts pane.
- Go to the “variables” menu and create a variable again.
- Set the variable to 20 as shown here.



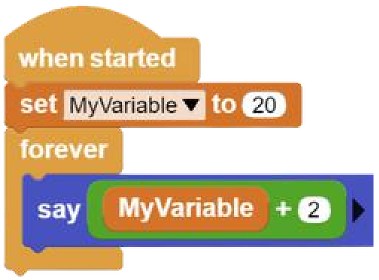
```
when started
set MyVariable ▼ to 20
```

- Switch to the **Control** menu and add a forever block below the "set" block so that your program will run indefinitely.
- From the **Output** blocks drag the "say" block into the “forever” loop. The default value is 123 but replace it with the addition operator as shown here.



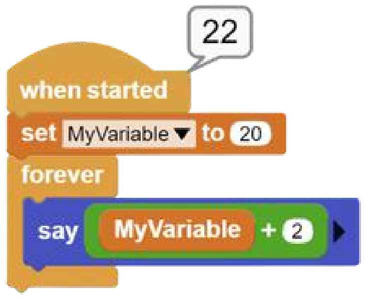
```
when started
set MyVariable ▼ to 20
forever
    say 10 + 2
```

- The operator block has two operands, replace anyone with the variable name.



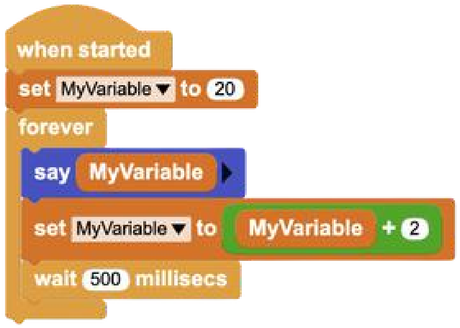
```
when started
set MyVariable ▼ to 20
forever
    say MyVariable + 2
```

- Start the program using the green play button in the upper right-hand corner. You will see the result of the calculation.



```
22
when started
set MyVariable ▼ to 20
forever
    say MyVariable + 2
```

- Let's conclude with something a little more interesting! Reconfigure your code as shown and now you are incrementing by 2.
- Play with other **Operators** items and see what you can come up with to alter your output in interesting ways.



```
when started
set MyVariable ▼ to 20
forever
    say MyVariable
    set MyVariable ▼ to MyVariable + 2
    wait 500 millisecs
```

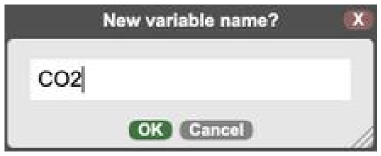
Program 7 - CO2 Detector

Now that you are familiar with some of the basics of MicroBlocks® let's create a useful program using a sensor. In this program you will code databot™ to act as a CO2 detector that acts as an alarm device if excessive CO2 levels are detected. Cool!

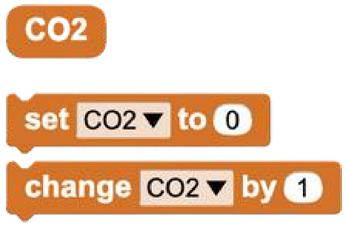
Begin by connecting your databot™ to a USB port and connect using the USB icon on MicroBlocks®.

- Create a Variable
  - Go to the **Variables** blocks and select "add a variable."
  - Write CO2 in the form to name your variable.

This will be the variable you can check at any time to see the current CO2 level.



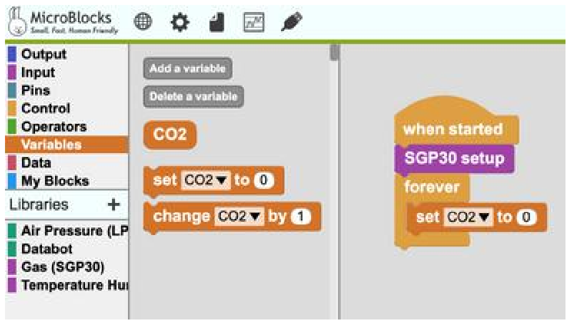
- As you have seen before, your new variable will now be displayed in the **Variables** blocks menu and easily accessed for your new program. Variables are awesome tools as they can hold a value that is constantly "varying" based on mathematical operators or, in this case, because it is reading live, changing data on the databot™ CO2 sensor.



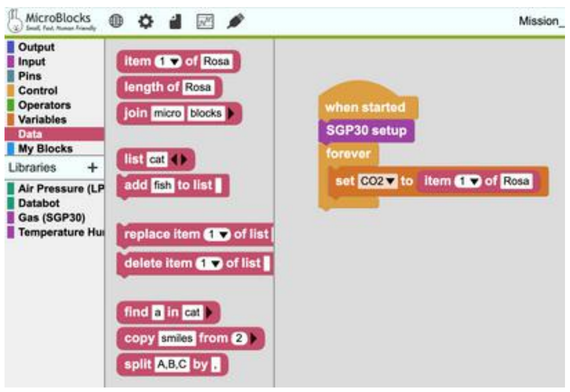
- From **Control** Blocks select "when started" .
- Select **Gas (SGP30)** library, you should see a variety of commands.
  - Add the **SGP30 setup**.

This tells MicroBlocks® where to find the sensor information.

- From **Control** Blocks add "forever" .
- From the **Variables Blocks** add the "set block".

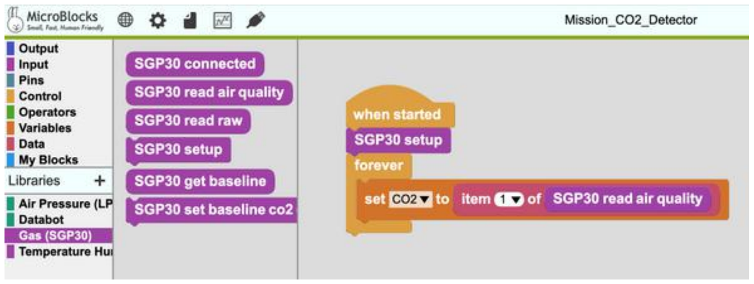


- From the **Data** Blocks, select the item 1 block as shown and add to the end of the set block.



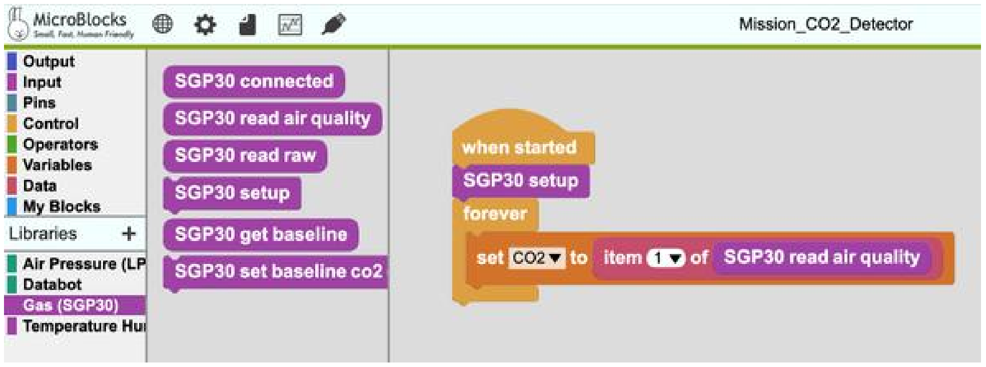
- This block enables you to select individual "items" from a list of values. The databot™ SGP30 gas sensor actually has two values it is reading:
  - 1 - CO2
  - 2 - Volatile Organic Compounds (VOCs)

- Add the SGP30 read air quality block and select Item 1 in the SGP 30 list to set your CO2 variable.



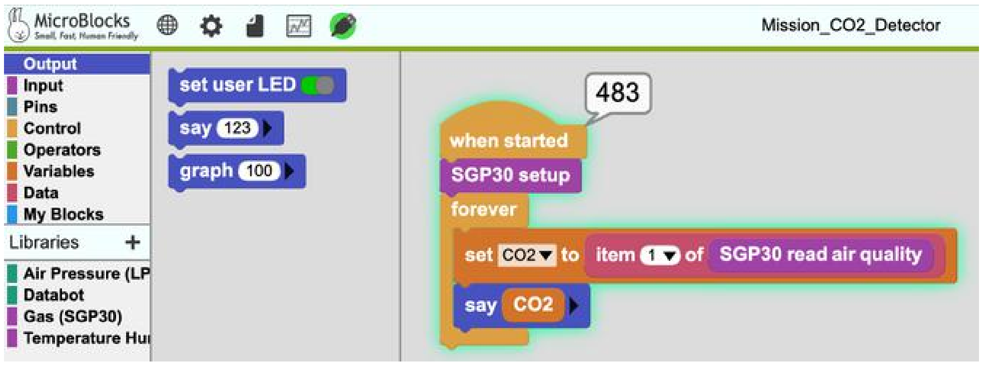
- From the **Gas** blocks select the SGP read air quality block.
- Place on the end of the item block as shown.

You are now set to read live data into your CO2 variable! Let's see if it works.



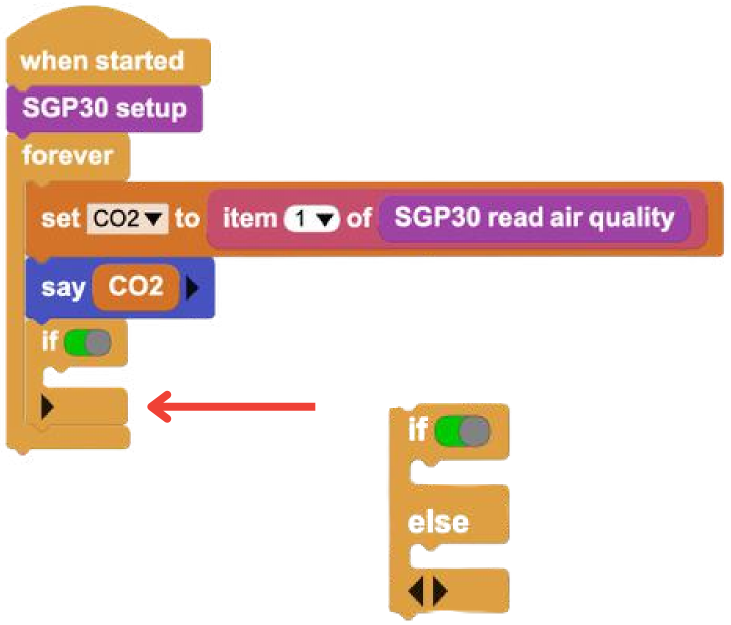
- From the **Output** blocks, drag a say block and place it into your forever loop.
- Add the CO2 variable from the **Variables** blocks and start your program.

You should see a changing value displayed as shown here. Breathe on databot™ and watch the number increase!



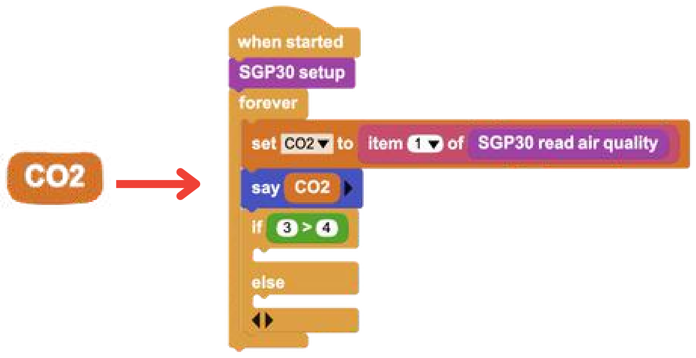
Make sure you are connected to your databot™, your USB icon should be green as shown in the menu bar.

Now that we are reading CO2 levels, it is time to create our alarm system. To do this we must create an "if" statement. These logic statements are used in coding to make a decision. For example, "if" the CO2 level is above 1,000 parts per million turn on a warning light.



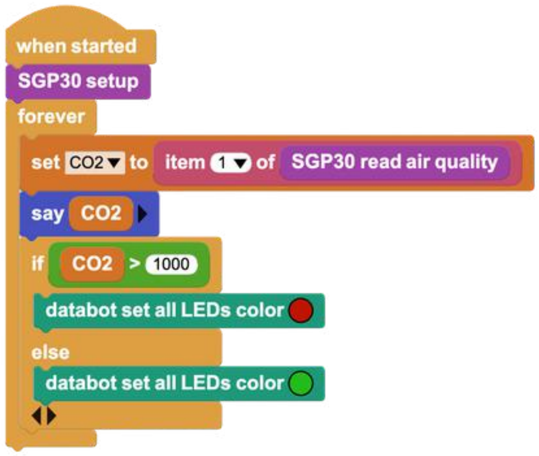
- From the **Control** blocks add in an if statement as shown.
- Expand it to be an "if-else" statement by clicking on the arrow.

- Now we add our decision! From the **Operators** blocks select the > block as shown.
- Replace the 3 with your CO2 variable.



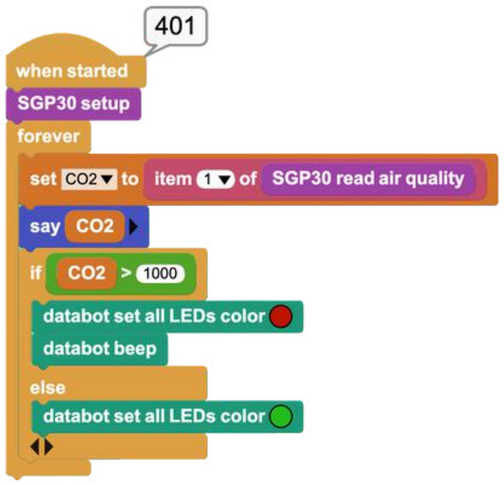
```
CO2 →
when started
SGP30 setup
forever
  set CO2 to item 1 of SGP30 read air quality
  say CO2
  if 3 > 4
  else
```

- Set your CO2 threshold to 1,000 parts per million (ppm) in the if statement as shown.
- From the **databot** blocks, add in two set all LEDs blocks as shown with red selected for your alarm color.
- Start your program and experiment with the detector. Breathe on databot™ and watch the CO2 levels increase. If you can get the number above 1000 ppm you should see your red alert LED.



```
when started
SGP30 setup
forever
  set CO2 to item 1 of SGP30 read air quality
  say CO2
  if CO2 > 1000
    databot set all LEDs color 🔴
  else
    databot set all LEDs color 🟢
```

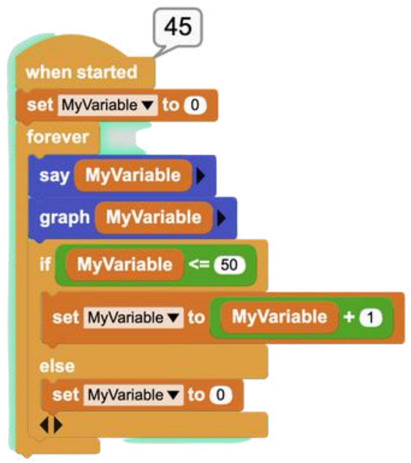
- Go further with your alarm and add some sound effects with the beep block.
- Test this new alarm effect.



```
401
when started
SGP30 setup
forever
  set CO2 to item 1 of SGP30 read air quality
  say CO2
  if CO2 > 1000
    databot set all LEDs color 🔴
    databot beep
  else
    databot set all LEDs color 🟢
```

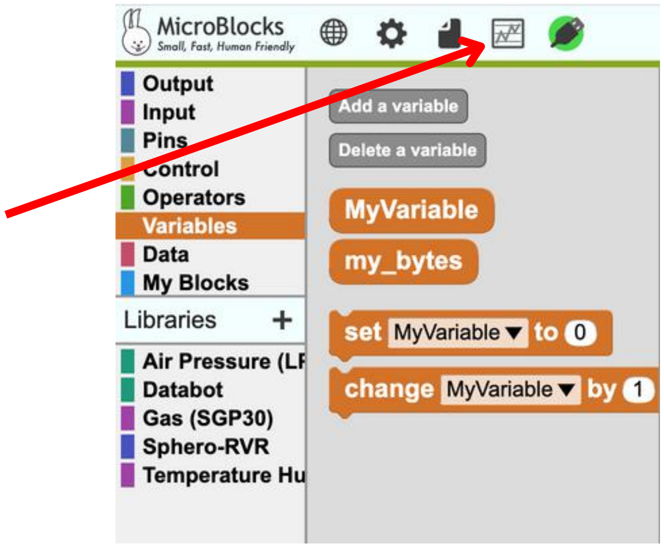
Program 8 - Graphing

MicroBlocks® has a built-in graphing function that is very useful for visualizing your data. Test it out by building this simple code stack. This creates a variable, sets it to zero, then increments it by one until it reaches 50 then it resets to zero. Notice the graph block has been added from the **Output** menu.

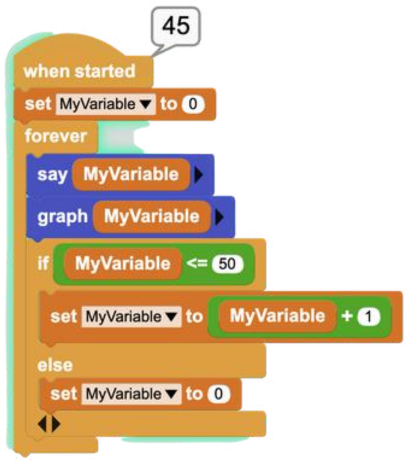


```
when started
set MyVariable to 0
forever
    say MyVariable
    graph MyVariable
    if MyVariable <= 50
        set MyVariable to MyVariable + 1
    else
        set MyVariable to 0
```

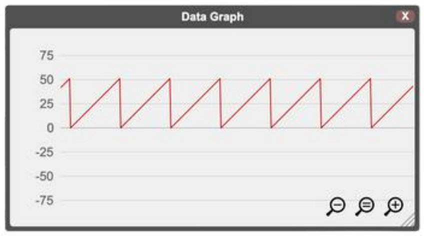
- Display the graph by selecting it from the main menu.



- Drag it on to your workspace and adjust the size to your preference.
- Start your program and observe the data.
- Play with your operators and see what other kinds of visualizations you can come up with.

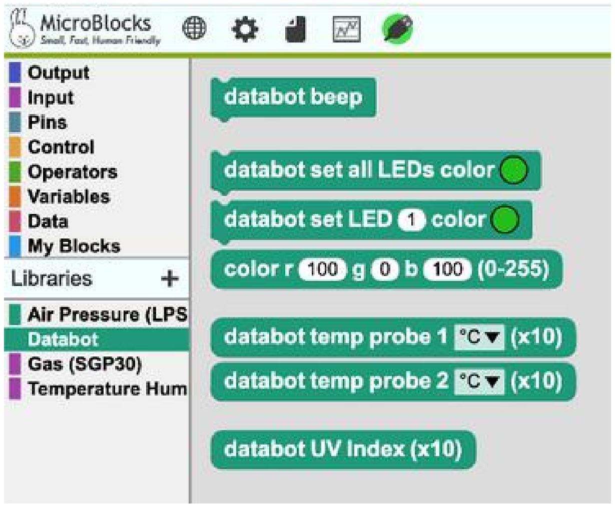


```
when started
set MyVariable to 0
forever
    say MyVariable
    graph MyVariable
    if MyVariable <= 50
        set MyVariable to MyVariable + 1
    else
        set MyVariable to 0
```



Program 9 - Temperature Probes

Your **databot** blocks category contains the sensor blocks for a number of the databot™ sensors as shown here.



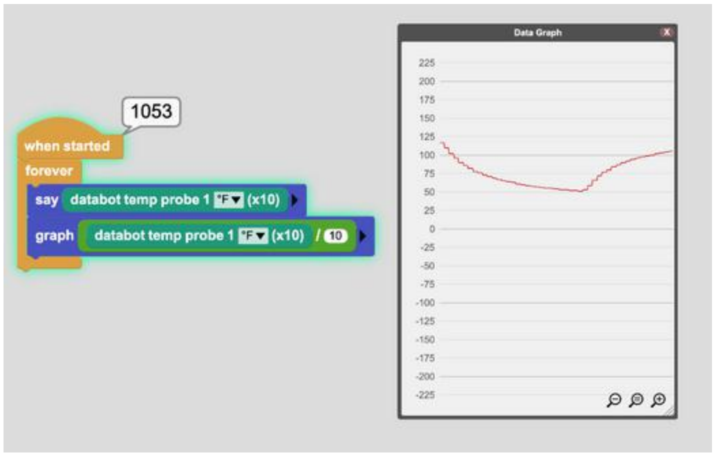
```
databot beep
databot set all LEDs color ●
databot set LED 1 color ●
color r 100 g 0 b 100 (0-255)
databot temp probe 1 °C▾ (x10)
databot temp probe 2 °C▾ (x10)
databot UV Index (x10)
```

This is an example of using the temperature probe being placed back and forth into a hot and cold water bath.

- Note the temperature output is multiplied by 10. This is because MicroBlocks does not support floating point decimal places. So the number you are seeing in the "say" block output is ten times the temperature in F°. For example, 1053 means 105.3° F.

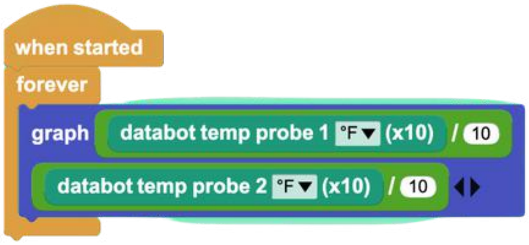
- The graph output is dividing the value by 10 to output the corrected value.

- You can use the drop down to select either Celsius or Fahrenheit units.

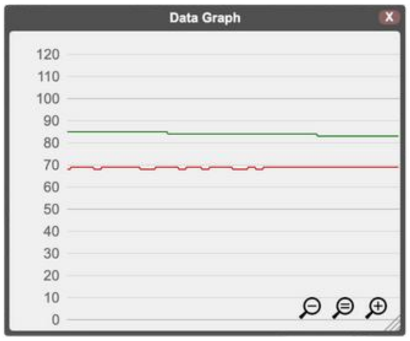


```
1053
when started
forever
  say databot temp probe 1 °F▾ (x10)
  graph databot temp probe 1 °F▾ (x10) / 10
```

MicroBlocks graphing can support up to six values and will display them in different colors. Here we display both temperature probes.



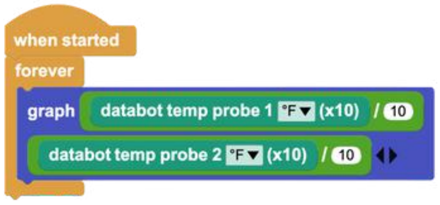
```
when started
forever
  graph databot temp probe 1 °F▾ (x10) / 10
        databot temp probe 2 °F▾ (x10) / 10
```

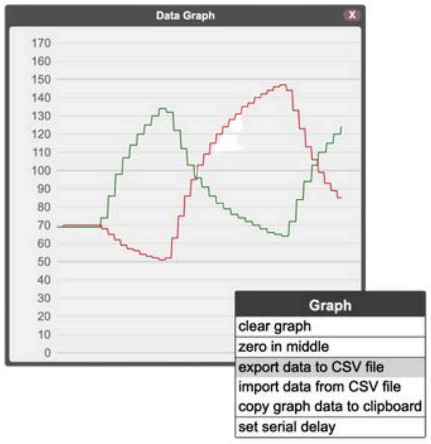


Program 10 - Exporting Graph Data

Exporting your graph data can be done by:

- Right clicking on your graph

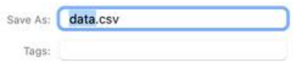

when started
forever
graph databot temp probe 1 °F (x10) / 10
databot temp probe 2 °F (x10) / 10



- Selecting export data to .CSV file.

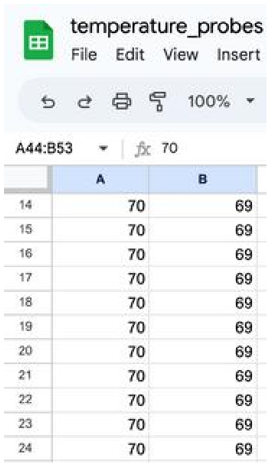
.

- Save to your local drive.



.

- Open in your desired software.



	A	B
14	70	69
15	70	69
16	70	69
17	70	69
18	70	69
19	70	69
20	70	69
21	70	69
22	70	69
23	70	69
24	70	69

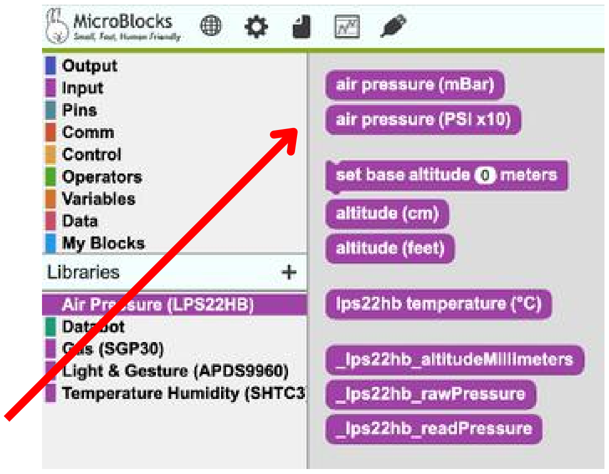
Program 11 - Under Pressure!

The air pressure sensor measures atmospheric pressure also called barometric pressure. The atmosphere is a layer of air wrapped around the earth. This air has mass and gravity exerts an attractive force on it creating weight. Air pressure then is the weight of air pressing against everything it touches.

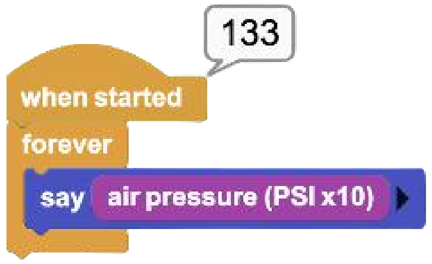
Using MicroBlocks you can read the atmospheric pressure using the sensor on board databot. There are two different units of measurement available in MicroBlocks:

- Millibars - Commonly used by meteorologists and scientists)
- Pounds Per Square Inch (PSI) - Commonly used to describe air pressure in tires, rafts, etc.

Let's try it out!



- Create this simple program and activate it to see what the air pressure is where you live.
- The units expressed here are Pounds Per Square Inch (PSI) multiplied by 10. This is because MicroBlocks does not support floating point decimals, so the actual number in PSI shown here will be 13.3.

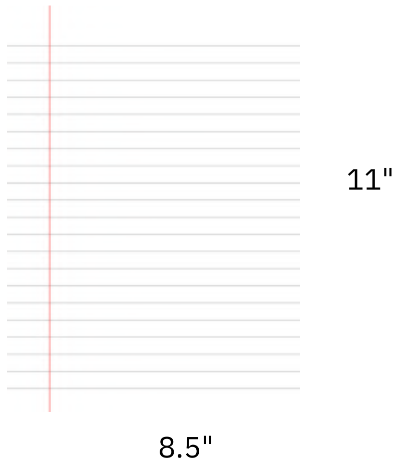


```
when started
forever
  say air pressure (PSI x10)
```

Air pressure in Pounds Per Square Inch (PSI) is 13.3 after dividing by 10 and adding the decimal.

- Let's apply this data we have just acquired and calculate the weight of air pressing down on a piece of paper on your desk.
- Lay a piece of paper out in front of you and measure its length by width and multiply them together to get the area of the paper.

8.5"	x	11"	=	93.5 sq in
Length		Width		Area



- Now multiply this area by your air pressure in PSI and see how much the air weighs that is pressing down on your paper.
- Are you surprised by this number?

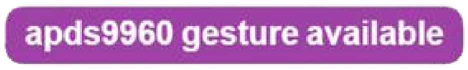
93.5 sq in	x	13.3	=	1,243.55 lbs
Area		Air Pressure in PSI		Weight of Air on Your Paper!

Program 12 - A Grand Gesture

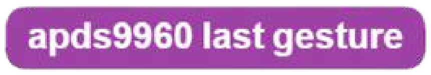
What does a gesture sensor do?

The gesture sensor detects hand motion. The directions of hand motion like UP, DOWN, LEFT, and RIGHT. When the hand/finger is swiped infront of the sensor the direction of the movement will be detected by it.

- You will use two new blocks in this code from the **Light & Gesture (APDS9960)** library. One is to detect the availability of the gesture and the other has the detected direction of the last gesture.
- The first block is used to detect the gesture so that we can begin the code only when the gesture is available. This block should be used along with the **When** block in the **Control** menu.
- The second block has the detected direction of hand motion like UP, DOWN, LEFT, and RIGHT.



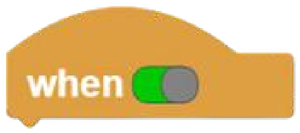
```
apds9960 gesture available
```



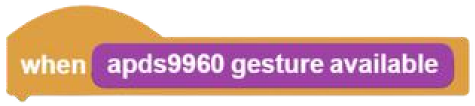
```
apds9960 last gesture
```

Let's Try It

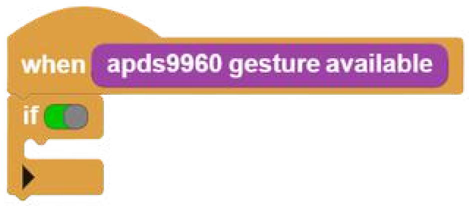
- Begin with the orange Control menu and drag the when block into the scripts pane.



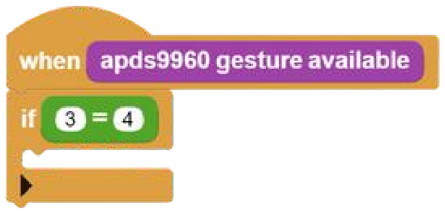
- Select Light & Gesture (APDS9960) library and add the **gesture available** block to the **when** block as shown, so that the blocks connected below this block will run only when the gesture is available.



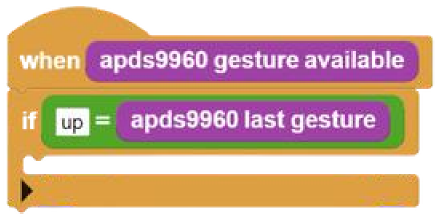
- It's time to check the direction motion of the hand/finger using the **If** block. Switch to the Control menu and add an **If** block below the **when** block.



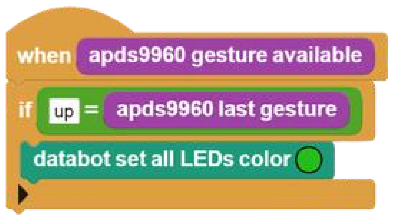
- To check the direction motion we need a compare operator (**=**). Go to the Operators menu and drag compare operator (**=**) to the **If** block.



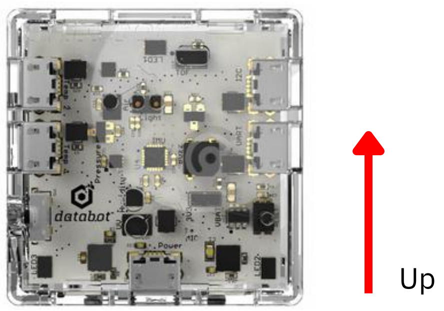
- Let’s first check the upward motion of the gesture by comparing the **last gesture** movement with the up text as shown.

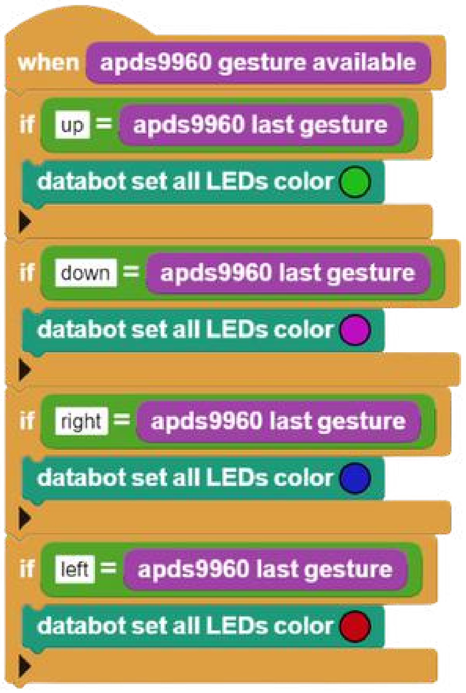

when apds9960 gesture available
if up = apds9960 last gesture

- If the **last gesture** movement is up then **set all LED’s color** to green. Select the databot library and add the databot™ **set all LED color** command and place it inside the **If** condition.


when apds9960 gesture available
if up = apds9960 last gesture
databot set all LEDs color

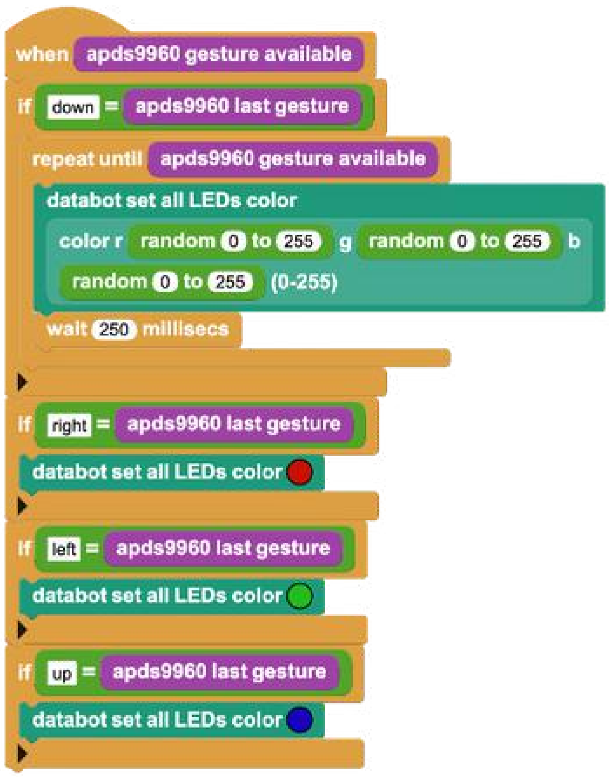
- Test your code! Start the program using the green play button in the upper right-hand corner. Swipe your hand upwards and you should see your databot LEDS light up green!





```
when apds9960 gesture available
  if up = apds9960 last gesture
    databot set all LEDs color 🟢
  if down = apds9960 last gesture
    databot set all LEDs color 🟣
  if right = apds9960 last gesture
    databot set all LEDs color 🔵
  if left = apds9960 last gesture
    databot set all LEDs color 🔴
```

- In a similar fashion, add blocks to check down, up, right, and left movements and **set** different colors for the LEDs.
- Once complete, start the program and swipe your hand in different motions: RIGHT, LEFT, UP, and DOWN and observe the results.
- Experiment by changing colors and watching the results update as you tinker.
- What other applications for the gesture sensor can you think of?

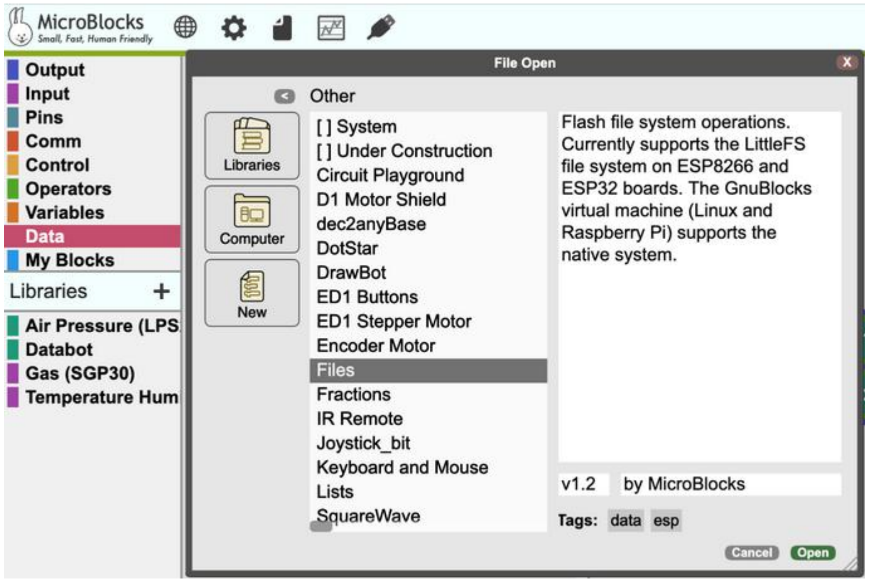


```
when apds9960 gesture available
  if down = apds9960 last gesture
    repeat until apds9960 gesture available
      databot set all LEDs color
        color r random 0 to 255 g random 0 to 255 b
        random 0 to 255 (0-255)
      wait 250 millisecs
  if right = apds9960 last gesture
    databot set all LEDs color 🔴
  if left = apds9960 last gesture
    databot set all LEDs color 🟢
  if up = apds9960 last gesture
    databot set all LEDs color 🔵
```

Try this fun extension! In this version of Grand Gesture we add in blinking light show that flashes random colored lights when you do a down gesture.

Program 13 - Recording Data

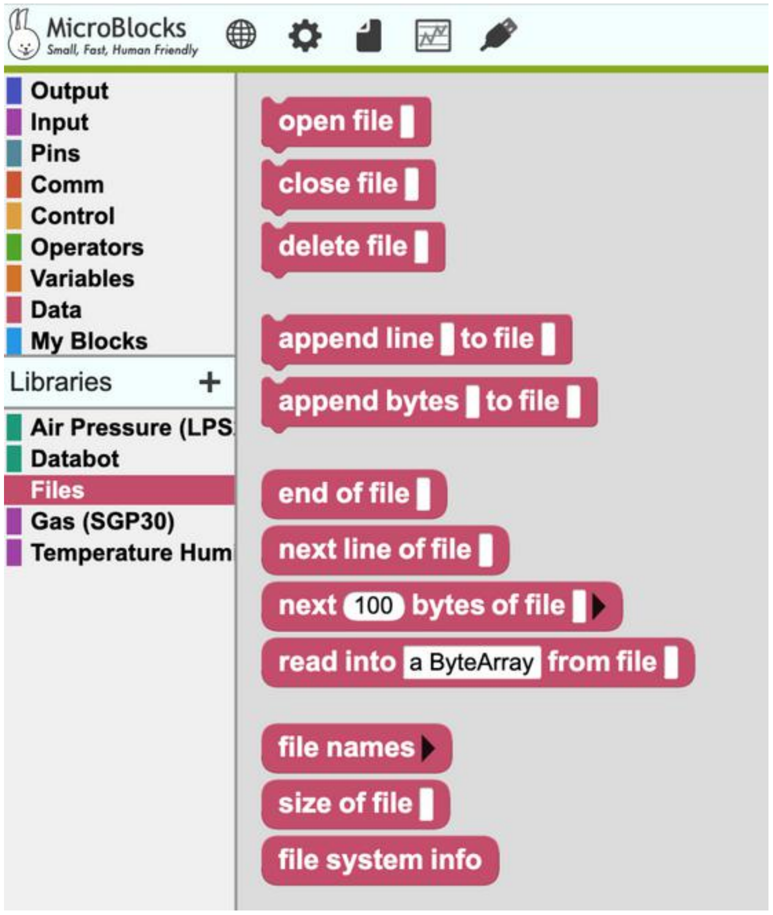
Adding the File library enables data logging and storing data to databot™s local memory. This feature empowers you to create any number of data logging experiments such as sending databot™ up in a high altitude weather balloon, on a remote drone mission, or gathering environmental data in a remote location over time. Go to your add libraries function and you will find "Files" in the "Other" library folder.



Flash file system operations. Currently supports the LittleFS file system on ESP8266 and ESP32 boards. The GnuBlocks virtual machine (Linux and Raspberry Pi) supports the native system.

Look at your Files blocks as shown here. Important background information:

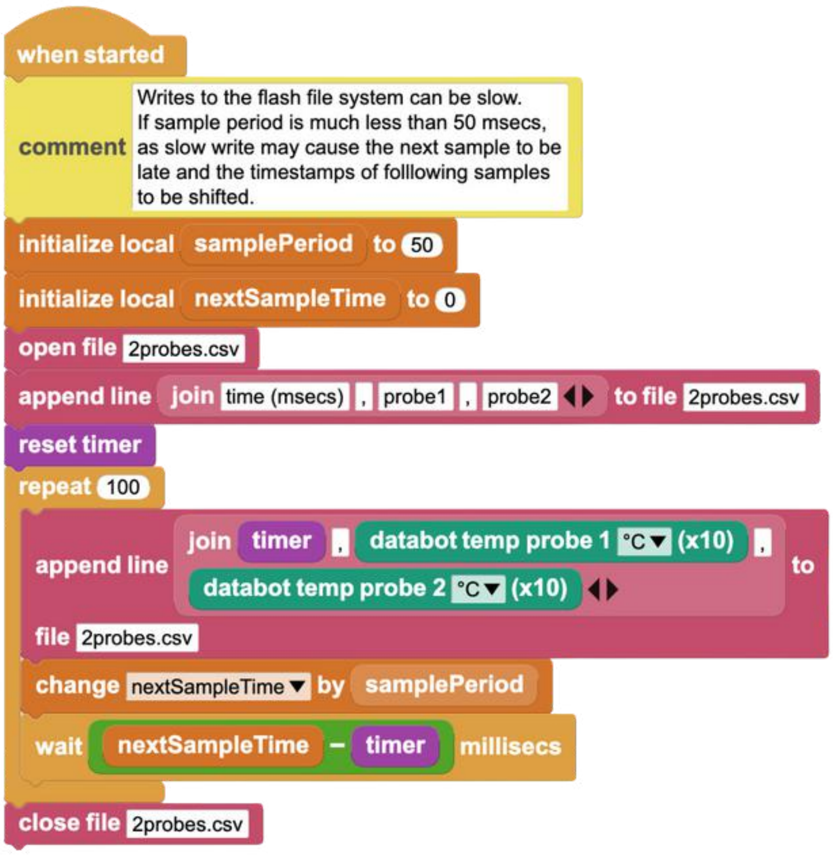
- Most file operations take a file name as the argument.
- The databot ESP32 file system does not support directories.
- File names are limited in length to 31 characters.



To create and append to a file, you do the following sequence:

- Open file
- Repeat as needed:
  - append line to file
- Close file

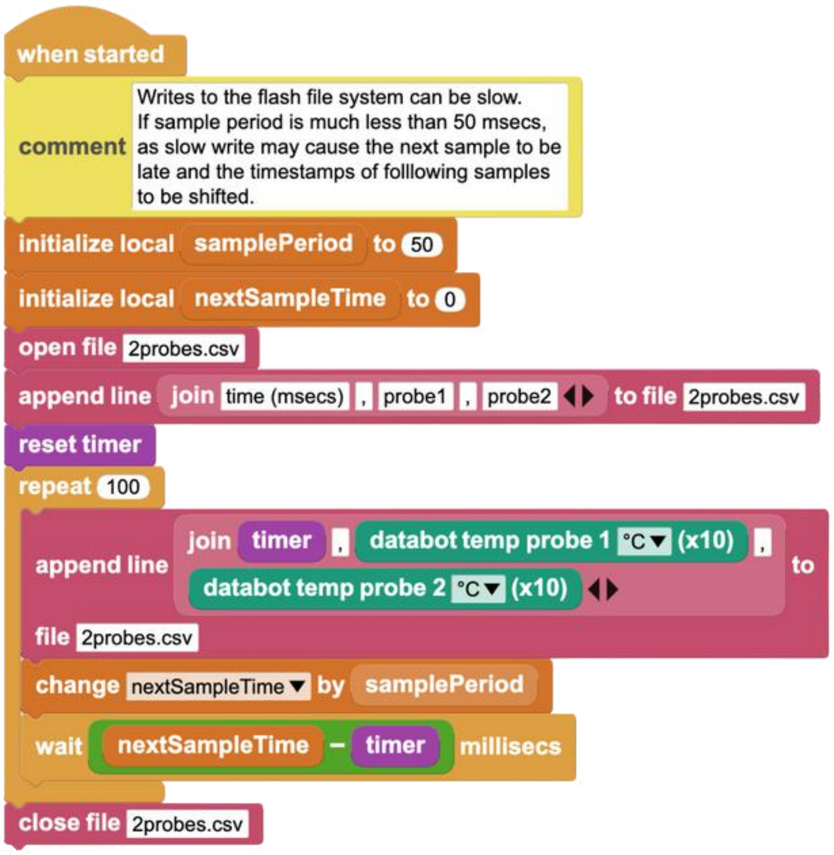
Here is a working example of a program that will create a file called 2probes.csv, insert a row that labels the data columns, then grabs a timestamp and temperature probe data appending this information to the file 100 times.



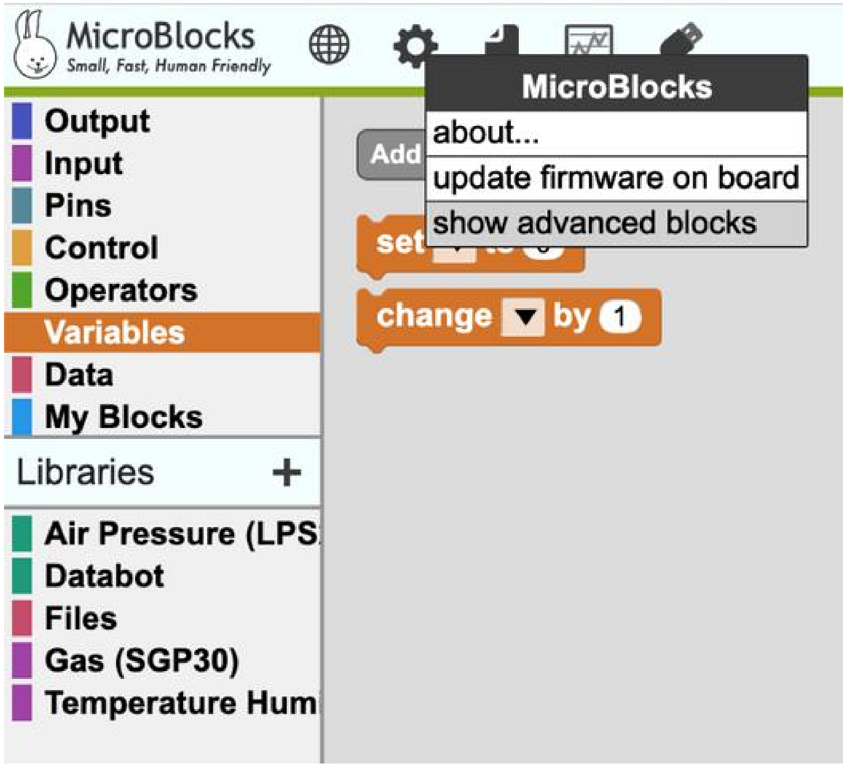
```
when started
comment  Writes to the flash file system can be slow.
         If sample period is much less than 50 msecs,
         as slow write may cause the next sample to be
         late and the timestamps of folllowing samples
         to be shifted.
initialize local samplePeriod to 50
initialize local nextSampleTime to 0
open file 2probes.csv
append line join time (msecs) , probe1 , probe2 ◀▶ to file 2probes.csv
reset timer
repeat 100
    append line join timer , databot temp probe 1 °C▼ (x10) ,
                     databot temp probe 2 °C▼ (x10) ◀▶          to
    file 2probes.csv
    change nextSampleTime▼ by samplePeriod
    wait nextSampleTime − timer millisecs
close file 2probes.csv
```

Some notes of further explanation.

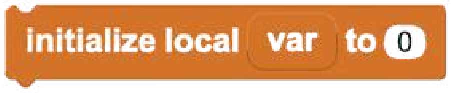
- If the file doesn't exist, it will be created.
- If it does exist, you'll be appending to it.
- If you want to clear the file before appending data to it, add a "delete file" before the other steps.
- Once this program is written and on your databot™ it will run each time you turn on your databot™. In this fashion you can develop programs that will run and collect data in the field, not connected to your programming device.

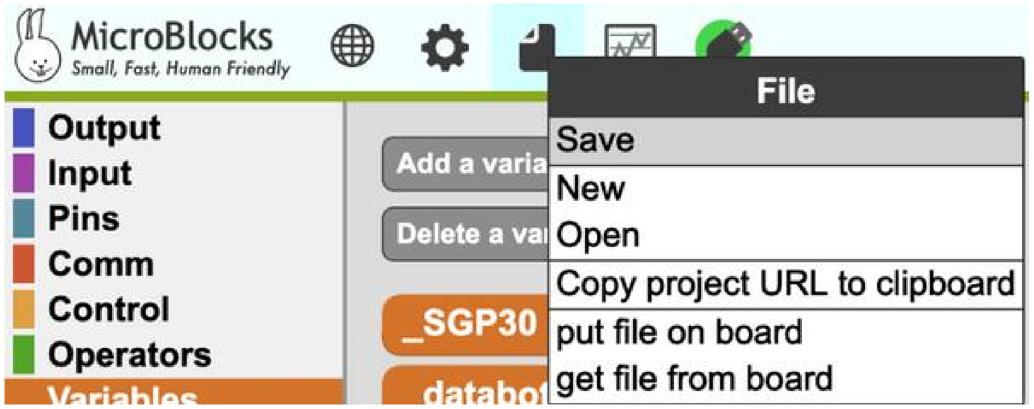


```
when started
comment  Writes to the flash file system can be slow.
         If sample period is much less than 50 msecs,
         as slow write may cause the next sample to be
         late and the timestamps of folllowing samples
         to be shifted.
initialize local samplePeriod to 50
initialize local nextSampleTime to 0
open file 2probes.csv
append line join time (msecs) , probe1 , probe2 ◀▶ to file 2probes.csv
reset timer
repeat 100
    append line join timer , databot temp probe 1 °C▼ (x10) ,
                     databot temp probe 2 °C▼ (x10) ◀▶          to
    file 2probes.csv
    change nextSampleTime▼ by samplePeriod
    wait nextSampleTime − timer millisecs
close file 2probes.csv
```



Note that this program is using local variables, not global. You can initialize local variables as shown by enabling show advanced blocks

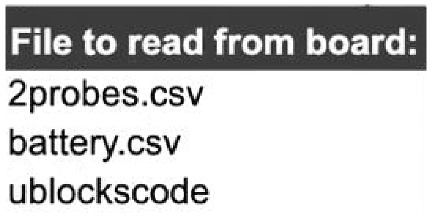




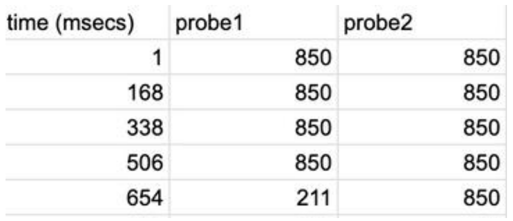
To retrieve your file from the databot™ internal memory follow these steps.

- Make sure you are connected to your databot™.
- Go to the File menu.
- Select get file from board.

- You will see a display showing you the available files.
- Select and download your file.



- This is an example of the output file created by this program.

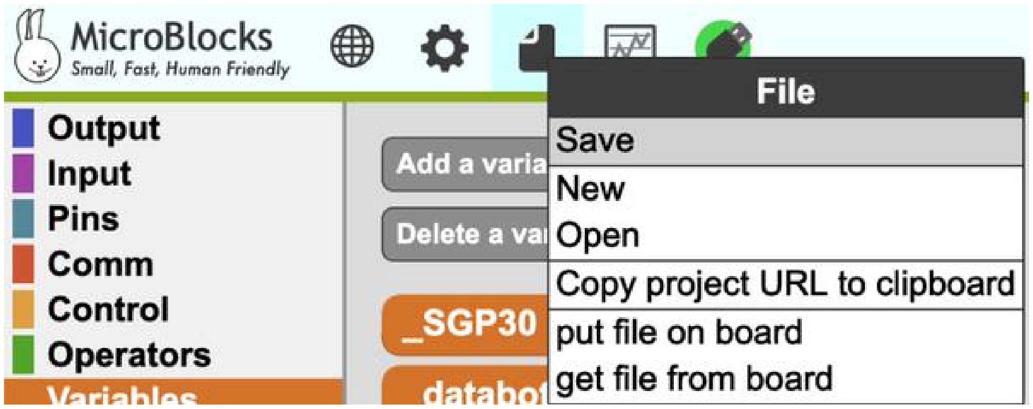


time (msecs)	probe1	probe2
1	850	850
168	850	850
338	850	850
506	850	850
654	211	850

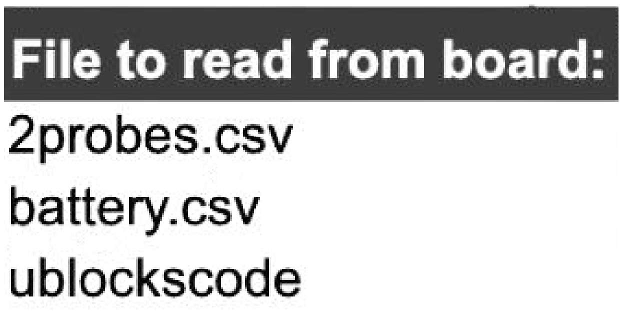
Congratulations! You are now able to store sensor data on your databot™. Go further with this program by tinkering with the file management functions, collecting data from various sensors, and importing your files into other programs for analysis.

To retrieve your file from the databot™ internal memory follow these steps.

- Make sure you are connected to your databot™.
- Go to the File menu.
- Select get file from board.



- You will see a display showing you the available files.

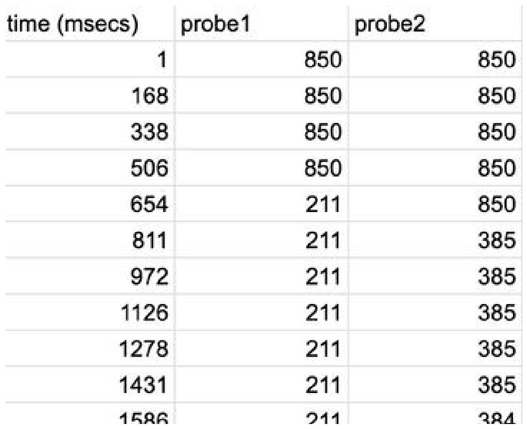


File to read from board:

2probes.csv
battery.csv
ublockscode

- Select and download your file.

- This is an example of the output file created by this program.



time (msecs)	probe1	probe2
1	850	850
168	850	850
338	850	850
506	850	850
654	211	850
811	211	385
972	211	385
1126	211	385
1278	211	385
1431	211	385
1586	211	384

Congratulations! You are now able to store sensor data on your databot™. Go further with this program by tinkering with the file management functions, collecting data from various sensors, and importing your files into other programs for analysis.