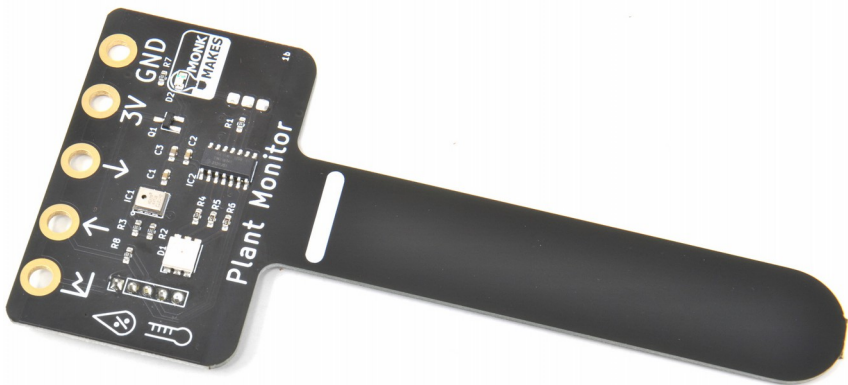


Instructions:



PLANT MONITOR



Instructions version 1h.

TABLE OF CONTENTS

Warning.....	2
Introduction	3
Using the Plant Monitor.....	4
Waterproofing.....	5
micro:bit.....	6
Raspberry Pi.....	10
Raspberry Pi Pico (1 or 2).....	15
ESP32.....	19
Arduino.....	23
Laptop with Serial Adapter.....	28
Troubleshooting.....	31
Support.....	31
MonkMakes.....	32

WARNING

Only the prong of the Plant Monitor below the white line should be allowed to get wet. If the top of the board gets wet, disconnect it from everything, dry it using a paper towel and then leave it the thoroughly dry out before trying to use it again.

INTRODUCTION

The MonkMakes Plant Monitor measures soil moisture, temperature and relative humidity. This board is compatible with the BBC micro:bit, Raspberry Pi and most microcontroller boards.

- Superior capacitive sensor (no electrical contact with soil)
- Alligator / crocodile clip rings (for use with BBC micro:bit and Adafruit Clue etc.)
- Ready soldered header pins for Arduino and other microcontroller boards.
- Easy to use UART serial interface
- Additional analog output for moisture only
- Built-in RGB LED (switchable)

In the following sections you will find instructions for using this board with:

- The BBC micro:bit (Makecode)
- The Raspberry Pi 4 (Python)
- The Raspberry Pi Pico (MicroPython)
- Arduino (Arduino C - SoftSerial)



USING THE PLANT MONITOR

The plant monitor should be placed as shown below.



The front side of the prong should be as close to the edge of the pot as possible. The sensing all takes place from the far side of the prong.

The electronics should be facing out of the pot and the prong of the Plant Monitor pushed into the dirt as far as the white line (but no deeper).

It's a good idea to attach the wires you are going to use to connect to the Plant Monitor before positioning it in the plant pot.

Once powered up, the plant monitor will immediately start displaying the level of wetness using the builtin LED. Red means dry, green means wet. Before you put the Plant Monitor in the pot, try gripping the prong in your hand and the moisture of your body should be enough to alter the LED's color.

WATERPROOFING

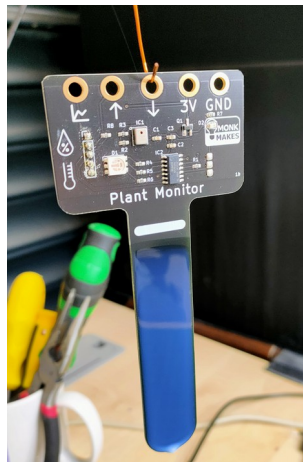
Your Plant Monitor is a Printed Circuit Board (PCB). These are made from layers of material, including copper layers used as a capacitive sensor. If you are leaving your Plant Monitor in wet soil, for long periods of time, the water may eventually seep between the layers of the PCB and stop it working properly.

Because the sensing does not rely on an electrical connection with the soil, you can take steps to waterproof your sensor. One way to do this is to coat it in outdoor wood varnish.

You can paint the varnish on, but its difficult to make sure that the edges (where the water is likely to get in) are well covered. So, we suggest dipping the prong of the plant Monitor in the varnish and then hanging it up to drip-dry. You can repeat this a few times, to give the Plant Monitor's prong a really good covering of varnish.

Be sure to follow the varnish's instructions for safely and drying times.

After dipping the Plant Monitor's prong, you can hang it up by the middle connector ring, using a piece of string or wire. Put something underneath to catch the drips. This could be the varnish can to start with but you'll want to put the lid back on after the main dripping is over, to stop the varnish in the can drying out.

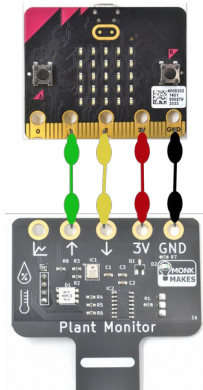


MICRO:BIT

You will need the following items to try out this example:

- A BBC micro:bit version 1 or 2
- 4 croc-clip leads

To make full use of wetness, temperature and humidity readings from the Plant Monitor, connect it up as shown below.



The red and black leads provide power from the micro:bit to the Plant Monitor and the yellow and green leads send serial data to and from the Plant Monitor.

There is a Makecode extension for the Plant Monitor that makes it easy to write code and the best way to get started is to load the example project from here:

https://makecode.microbit.org/_gCdhCRXqfHtY

When you download this project onto your micro:bit it will show the wetness using the bar graph block. Pressing button A will show the temperature in degrees C and pressing button B, the relative humidity as a percentage.

Here's the code:

```
on start
  Start Plant Monitor
  set show_wetness to true
```

```
forever
  if show_wetness = true then
    plot bar graph of Plant Wetness (0-100)
    up to 100
```

```
on button A pressed
  set show_wetness to false
  clear screen
  show number Plant Temperature (deg C)
  pause (ms) 1000
  set show_wetness to true
```

```
on button B pressed
  set show_wetness to false
  clear screen
  show number Plant Humidity
  pause (ms) 1000
  set show_wetness to true
```

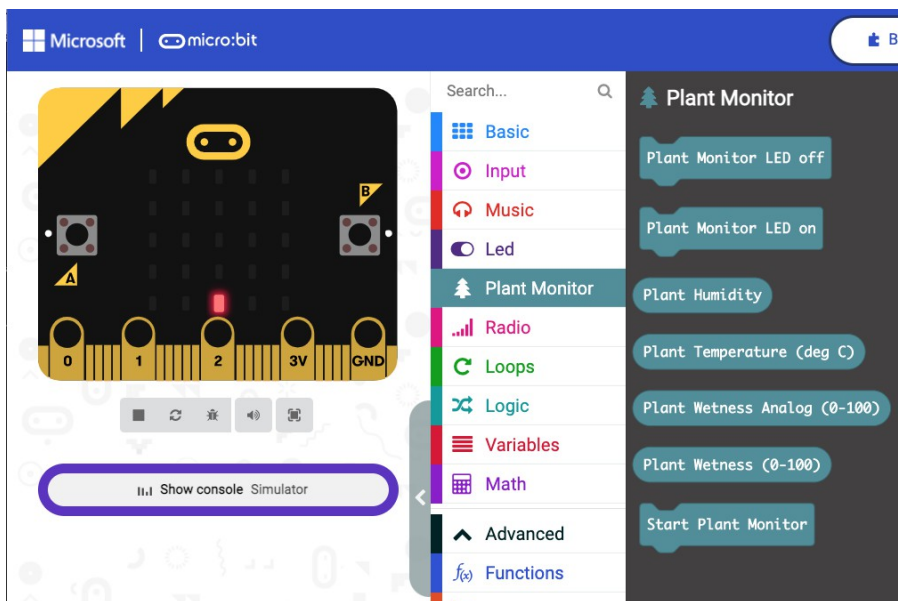
The on start block must contain the block **Start Plant Monitor**, that initiates communication between the micro:bit and the Plant Monitor. The variable **show_wetness** is used to switch the displaying of the bar graph on and off, otherwise it will try and draw the bar graph over the top of the temperature and humidity readings.

The **forever** block uses the block **Plant Wetness** and displays it as a bar graph as long as **show_wetness** is true.

When button A is pressed, the screen is cleared and **show_wetness** set to false and then the temperature is displayed. After a pause for you to read the temperature, **show_wetness** is set to true again so that the bar graph continues refreshing.

The button B code is much the same, except that the relative humidity is displayed rather than the temperature.

You can see the Plant Monitor extension in the Palette – it has an icon of a tree.



As well as the **Start Plant Monitor**, **Plant Humidity** and **Plant Temperature** blocks, there are also blocks that turn the LED on and off.

Extension

If you would prefer to install the extension, then select Extensions from the MakeCode editor and then paste the following URL into the Search/URL field at the

top.

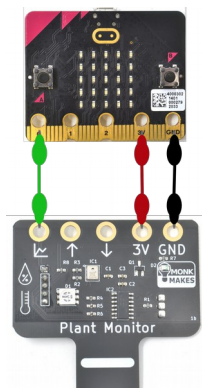
<https://github.com/monkmakes/plant-monitor-makecode>

Select the MonkMakes Plant Monitor extension and this will be added to your palette.

Analog Output

The **Plant Wetness Analog** block allows you to measure the wetness using the Plant Monitor's analog output pin. This is useful if you want to use the Makecode plotting facility, as this is not possible using the serial interface.

To use the analog interface, connect your micro:bit like this:



As before, power is supplied by the micro:bit, but this time, micro:Bit pin 0 is used to connect to the analog output of the Plant Monitor.

Connecting the Plant Monitor in this way only measures wetness, but it does mean that because the serial interface of the micro:bit is not used, it can be used to send serial data back to Makecode's plotting feature.

Try out the project here:

https://makecode.microbit.org/_cFq7vzhAXYTz

The code for this is really quite minimal.



If you have your micro:bit paired to the Makecode editor in your browser, then as soon as the program has uploaded, you should see an option Show Console Device appear. Clicking on this will show a real-time plot of the wetness reading.

makecode.microbit.org/#editor

Microsoft | micro:bit

Blocks JavaScript

Go back

Device

0 1 2 3V GND

wetness: 52

34.00

Show console Simulator

Show console Device

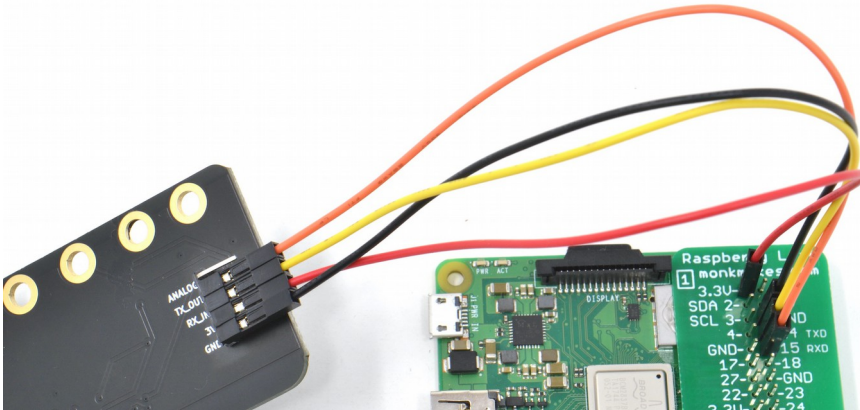
```
1 wetness:52
2 wetness:53
2 wetness:52
2 wetness:53
wetness:52
```

RASPBERRY PI

You will need the following items to try out this example:

- A Raspberry Pi version 2 or newer recommended
- 4 female to female jumper wires
- Optionally a GPIO template like the Raspberry Leaf to make pin identification easier. (<https://monkmakes.com/leaf.html>)

Connect your Raspberry Pi to the Plant Monitor using female to female jumper wires as shown below:

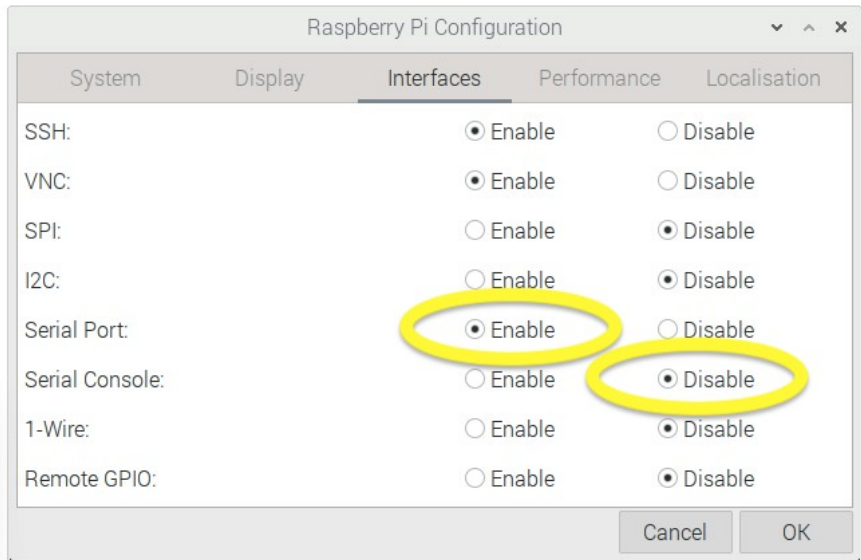


The connections are:

- GND to GND
- 3.3V on the Raspberry Pi to 3V on the Plant Monitor
- 14 TXD on the Raspberry Pi to RX_IN on the Plant Monitor
- 15 RXD on the Raspberry Pi to TX_OUT on the Plant Monitor

WARNING: Do not connect this board to the 5V pin of the Raspberry Pi. The board is designed to operate at 3.3V and a 5V supply is likely to destroy it.

Since the Raspberry Pi does not have analog inputs, then the only interface option is to use the serial UART interface. This interface has to be enabled on the Raspberry Pi from the Raspberry Pi Configuration tool that you will find in the Preferences section of the Start menu.



Enable the Serial Port and Disable the Serial Console. You may be prompted to restart your Pi for these changes to take effect.

To download the example programs for the Plant Monitor open a terminal window and enter the command:

```
$ git clone https://github.com/monkmakes/pmon.git
```

This will download all the example programs for various different platforms into a folder called pmon, so change to the right directory for the Raspberry Pi examples by entering the command:

```
$ cd pmon/raspberry_pi
```

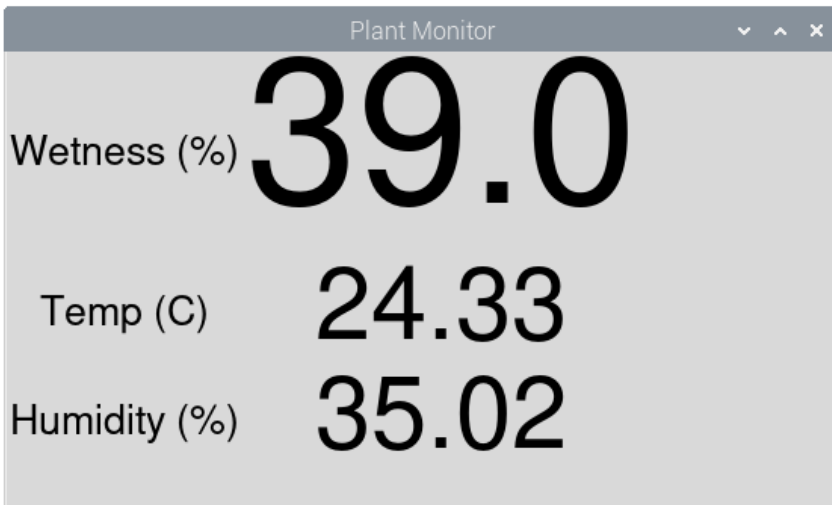
Before you can run the examples, you will need to install the GUIZero library with the command:

```
$ pip3 install guizero
```

You can now run the examples. The first example 01_meter.py displays the wetness level, the temperature and humidity level. Run it using the command below.

```
$ python3 01_meter.py
```

.. and the following window should appear.



Try holding the prong of the moisture meter and you should see the wetness % rise. Similarly putting your finger over the metal box on the Plant Monitor that is the temperature and humidity sensor will change both readings.

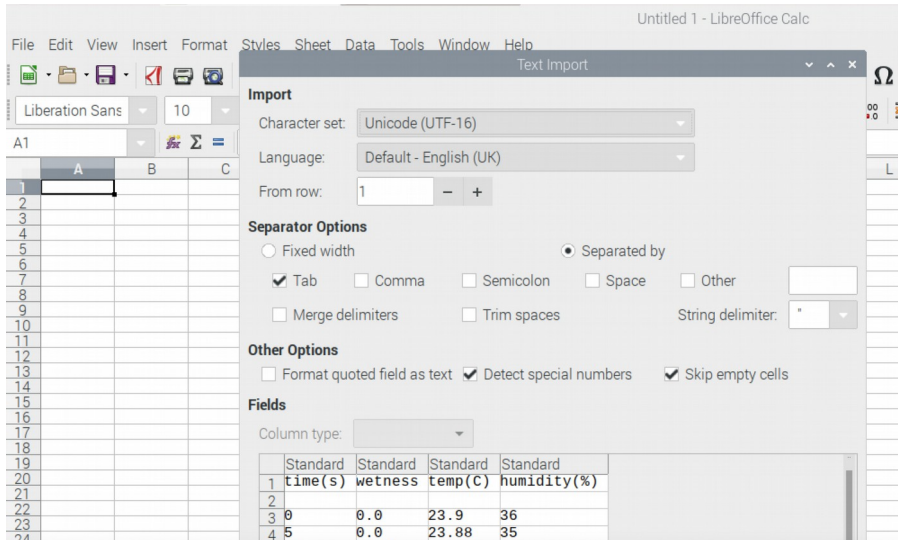
The second example (`02_data_logger.py`) is a data logger that records all three values periodically and put them into a file that you can then import into a spreadsheet.

Run the program as shown below and collect some data. You might like to put the Plant Monitor into a plant pot (see page 4) and record readings every minute for 24 hours (but perhaps start with a smaller set of data).

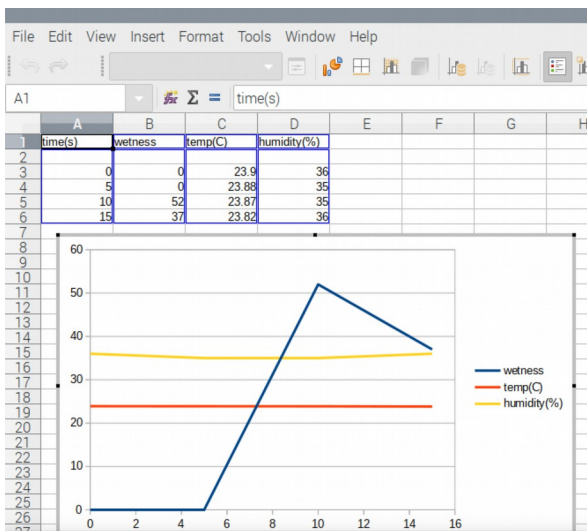
```
$ python3 02_data_logger.py
Enter interval between readings (seconds):60
Enter filename:test.txt
Logging started at: 2022-05-05 14:54:33
Press CTRL-c to end logging
time(s)      wetness      temp(C)      humidity(%)
0           0.0      23.9      36
5           0.0      23.88     35
10          52.0      23.87     35
15          37.0      23.82     36
20          36.0      24.58     45
25          93.0      25.43     36
^C
```

Logging to file test.txt complete

When you have got enough data, CTRL-c the program. If you don't have a spreadsheet installed on your Raspberry Pi, install LibreOffice using the Recommended Software tool in the Preferences section of the Start menu. Run LibreCalc and then import the file of data.



You can draw a chart of the data by selecting it (including the headings) and then inserting a Chart. Pick a chart type of XY.

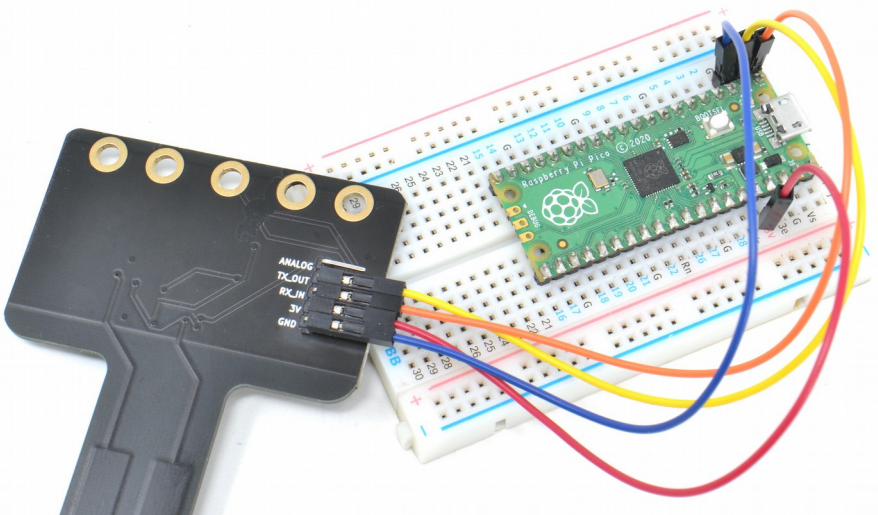


RASPBERRY PI PICO (1 OR 2)

You will need the following items to try out this example:

- A Raspberry Pi Pico or Pico W
- Solderless breadboard. The MonkMakes Breadboard for Pico is recommended as it labels the Pico's pins.
(http://www.monkmakes.com/pico_bb.html)
- 4 male to female jumper wires

Connect the Plant Monitor to your Raspberry Pi Pico using solderless breadboard and female to male jumper wires as shown below, or if you prefer female to female jumper wires, directly from board to board..



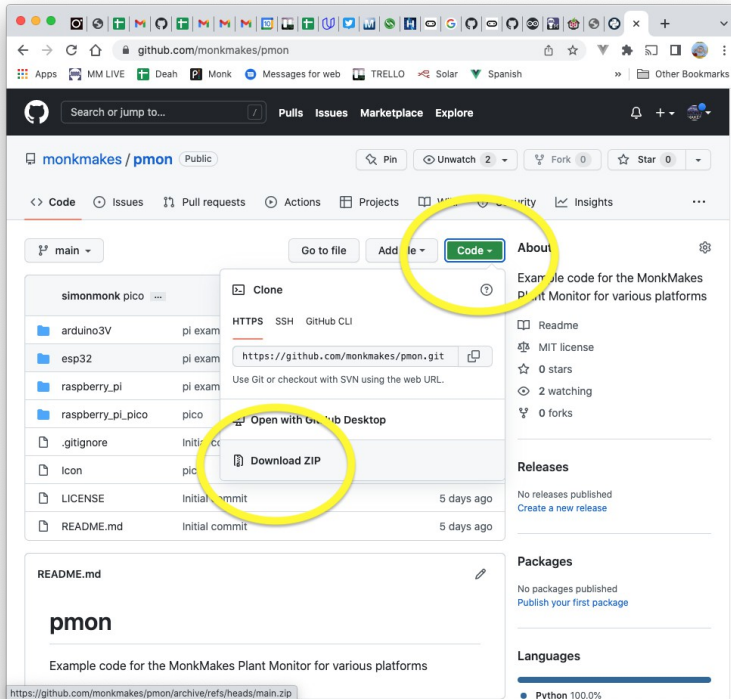
The connections are as follows:

- GND to GND
- 3V on the Pico to 3V on the Plant Monitor
- TX on the Pico to RX_IN on the Plant Monitor
- RX on the Pico to TX_OUT on the Plant Monitor

Using the MonkMakes Breadboard for Pico (https://www.monkmakes.com/pico_bb) will make it very much easier to identify which pin of the Pico is which.

To get you started, you will find a MicroPython library and test program in the examples for this board on its github page here:
<https://github.com/monkmakes/pmon>

If you are unfamiliar with git, the easiest way to download it is to go to the github page above and then use the download ZIP feature (see below).



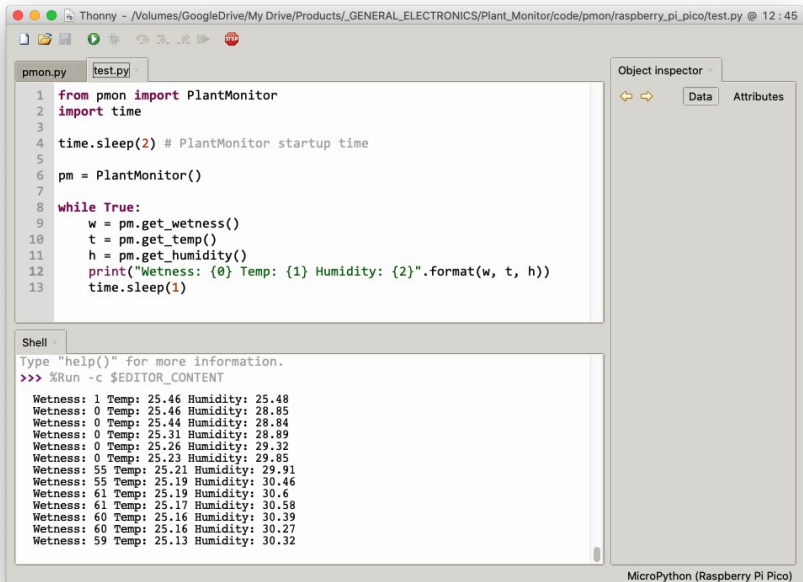
This download contains the example files for the Plant Monitor for lots of platforms, not just the Pico. So having downloaded the ZIP archive and extracted it, you will find it contains a folder called `raspberry_pi_pico`. Inside this folder you will find five files, that you should open using Thonny (<https://thonny.org/>).

- `pmon.py` – a MicroPython library for the Plant Monitor
- `test.py` – a test program using the `pmon` library
- `microdot.py` – the microdot light-weight web server
- `mm_wlan.py` – A Wireless LAN library
- `pico_w_server.py` – a web server reporting readings from the Plant Monitor

The last three of these files are only needed if you want to run the Pico W web server example.

Before you can run `test.py` copy the files `pmon.py`, `microdot.py` and `mm_wlan.py` onto your Pico, by using the *SaveAs* menu option in Thonny and then selecting *Raspberry Pi Pico* for the destination. When you run `test.py` you should see the wetness, temperature and humidity being printed out.

Use the program `test.py` as a template for your own programs for the Pico.



```
1 from pmon import PlantMonitor
2 import time
3
4 time.sleep(2) # PlantMonitor startup time
5
6 pm = PlantMonitor()
7
8 while True:
9     w = pm.get_wetness()
10    t = pm.get_temp()
11    h = pm.get_humidity()
12    print("Wetness: {0} Temp: {1} Humidity: {2}".format(w, t, h))
13    time.sleep(1)
```

```
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

Wetness: 1 Temp: 25.46 Humidity: 25.48
Wetness: 0 Temp: 25.46 Humidity: 28.85
Wetness: 0 Temp: 25.44 Humidity: 28.84
Wetness: 0 Temp: 25.31 Humidity: 28.89
Wetness: 0 Temp: 25.26 Humidity: 29.32
Wetness: 0 Temp: 25.23 Humidity: 29.85
Wetness: 55 Temp: 25.21 Humidity: 29.91
Wetness: 55 Temp: 25.19 Humidity: 30.46
Wetness: 61 Temp: 25.19 Humidity: 30.6
Wetness: 61 Temp: 25.17 Humidity: 30.58
Wetness: 60 Temp: 25.16 Humidity: 30.39
Wetness: 60 Temp: 25.16 Humidity: 30.27
Wetness: 59 Temp: 25.13 Humidity: 30.32
```

Web Interface (Pico W and Pico 2W only)

If you have a Pico W or Pico 2 W, then you can take advantage of its wireless capabilities and try out the example `pico_w_server.py`.

Pico W Plant Monitor

Water: 93

Temp (C): 31.43

Humidity: 26.94

Before running `pico_w_server.py`, find the following lines, and put your network name and password into the code.

```
ssid = 'network name'  
password = 'password'
```

When you run the program, you should see something like:

```
Connecting to Network...  
Connected IP Address = 192.168.1.132  
Setting up webserver...
```

Now open a browser window on a computer on the same network as the Pico W and navigate to the IP address shown when the program first ran (in this case, 192.168.1.132).

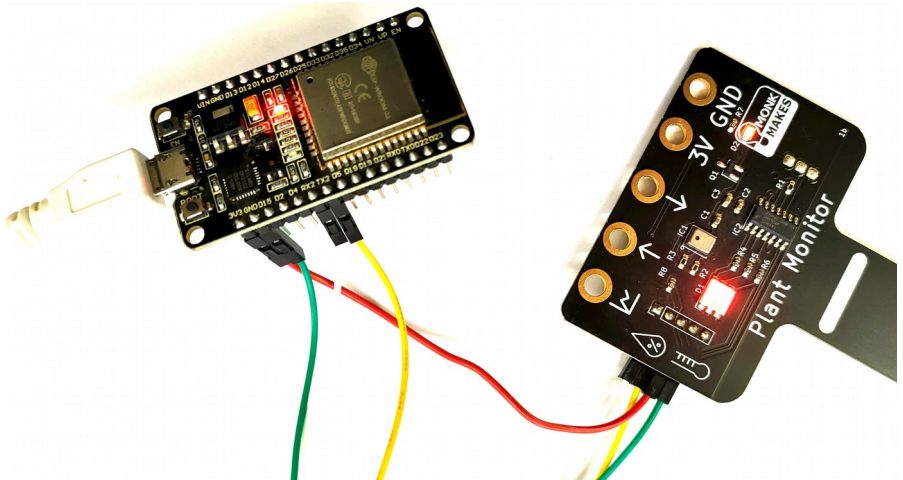
The readings will refresh every second.

ESP32

You will need the following items to try out this example:

- An ESP32 board such as the ESP32 Lite or ESP32 DevKit 1.
- 4 female to female jumper wires

Connect the Plant Monitor to your ESP32 board using solderless breadboard and female to female jumper wires as shown below, or if you prefer female to female jumper wires, directly from board to board..



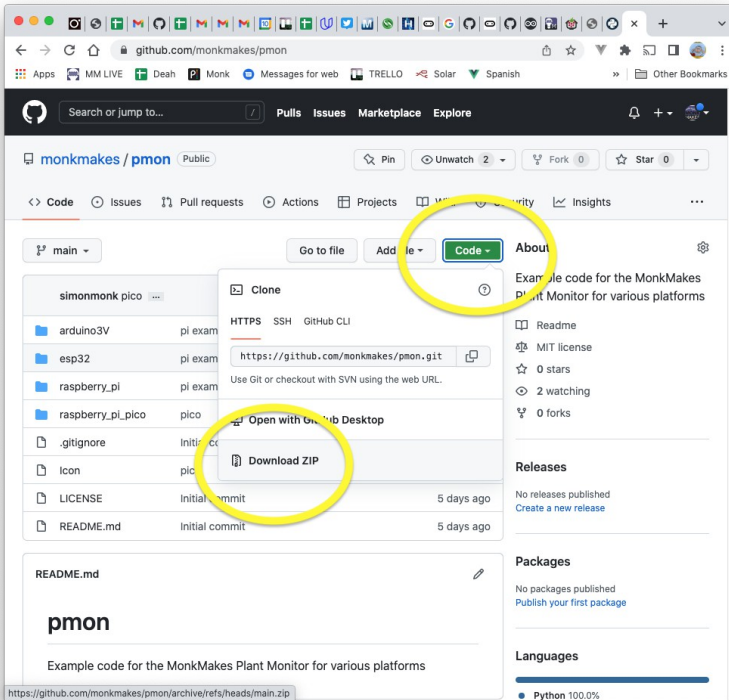
The connections are as follows:

- GND to GND
- 3V on the ESP32 to 3V on the Plant Monitor
- TX2 (17) on the ESP32 to RX_IN on the Plant Monitor
- RX2 (16) on the ESP32 to TX_OUT on the Plant Monitor

To get you started, you will find a MicroPython library and test program in the examples for this board on its github page here:

<https://github.com/monkmakes/pmon/esp32>

If you are unfamiliar with git, the easiest way to download it is to go to the github page above and then use the download ZIP feature (see below).



This download contains the example files for the Plant Monitor for lots of platforms, not just the ESP32. So having downloaded the ZIP archive and extracted it, you will find it contains a folder called `esp32`. Inside this folder you will find four files, that you should open using Thonny (<https://thonny.org/>).

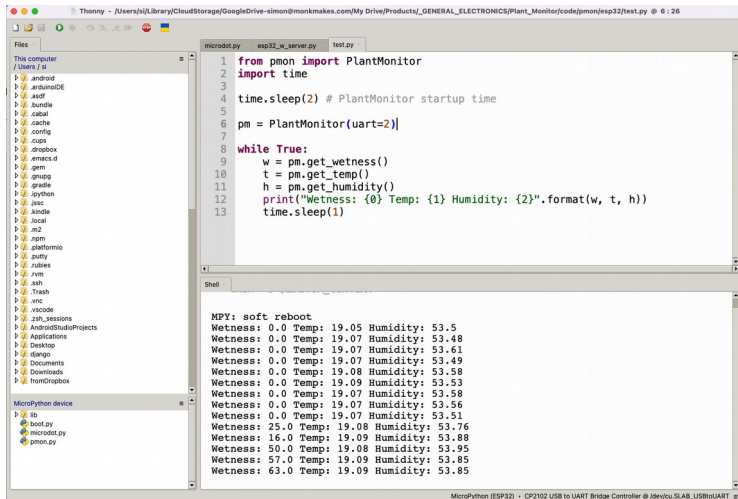
- `pmon.py` – a MicroPython library for the Plant Monitor
- `test.py` – a test program using the `pmon` library
- `microdot.py` – the microdot light-weight web server
- `esp32_w_server.py` – a web server reporting readings from the Plant Monitor

The last three of these files are only needed if you want to run the web server example.

Before you can run `test.py` copy the files `pmon.py` and `microdot.py` onto your ESP32, by using the *SaveAs* menu option in Thonny and then selecting *MicroPython Device* for the destination. When you run `test.py` you should see the

wetness, temperature and humidity being printed out.

Use the program *test.py* as a template for your own programs for the Pico.

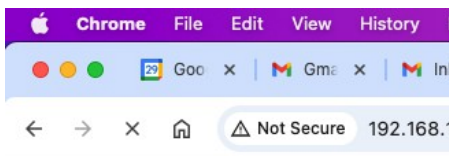


```
1 from pmon import PlantMonitor
2 import time
3
4 time.sleep(2) # PlantMonitor startup time
5
6 pm = PlantMonitor(uart=2)
7
8 while True:
9     w = pm.get_wetness()
10    t = pm.get_temp()
11    h = pm.get_humidity()
12    print("Wetness: {0} Temp: {1} Humidity: {2}".format(w, t, h))
13    time.sleep(1)
```

```
MPY: soft reboot
Wetness: 0.0 Temp: 19.05 Humidity: 53.5
Wetness: 0.0 Temp: 19.07 Humidity: 53.48
Wetness: 0.0 Temp: 19.07 Humidity: 53.61
Wetness: 0.0 Temp: 19.07 Humidity: 53.49
Wetness: 0.0 Temp: 19.08 Humidity: 53.58
Wetness: 0.0 Temp: 19.09 Humidity: 53.53
Wetness: 0.0 Temp: 19.07 Humidity: 53.58
Wetness: 0.0 Temp: 19.07 Humidity: 53.56
Wetness: 0.0 Temp: 19.07 Humidity: 53.51
Wetness: 25.0 Temp: 19.08 Humidity: 53.76
Wetness: 16.0 Temp: 19.09 Humidity: 53.88
Wetness: 50.0 Temp: 19.08 Humidity: 53.95
Wetness: 57.0 Temp: 19.09 Humidity: 53.85
Wetness: 63.0 Temp: 19.09 Humidity: 53.85
```

ESP32 Web Server Example

This example runs a simple web server on the ESP32 that you can connect to with your browser to see the readings from the Plant Monitor.



ESP32 Plant Monitor

Water: 0.0

Temp (C): 19.06

Humidity: 53.59

The program for this can be found in `esp32_w_server.py`. Before running it, find the following lines, and put your network name and password into the code.

```
ssid = 'network name'  
password = 'password'
```

When you run the program, you should see something like:

```
Connecting to Network...  
Connected IP Address = 192.168.1.132  
Setting up webserver...
```

Now open a browser window on a computer on the same network as the Pico W and navigate to the IP address shown when the program first ran (in this case, 192.168.1.132).

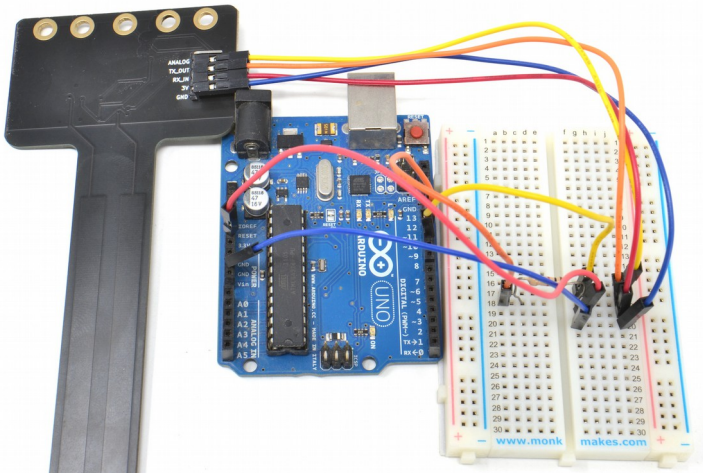
The readings will refresh every second.

ARDUINO

You will need the following items to try out this example:

- An Arduino Uno
- Solderless breadboard
- A 1kΩ resistor
- 4 male to female jumper wires
- 4 male to male jumper wires

If you have a 3V Arduino, your wiring should be very much simpler and you won't need the 1kΩ resistor.



Warning: The Plant Monitor is designed to operate at 3.3V, not the 5V that some Arduinos such as the Arduino Uno operate at. So, never power the Plant Monitor with 5V and make sure that none of its input pins receive more than 3.3V.

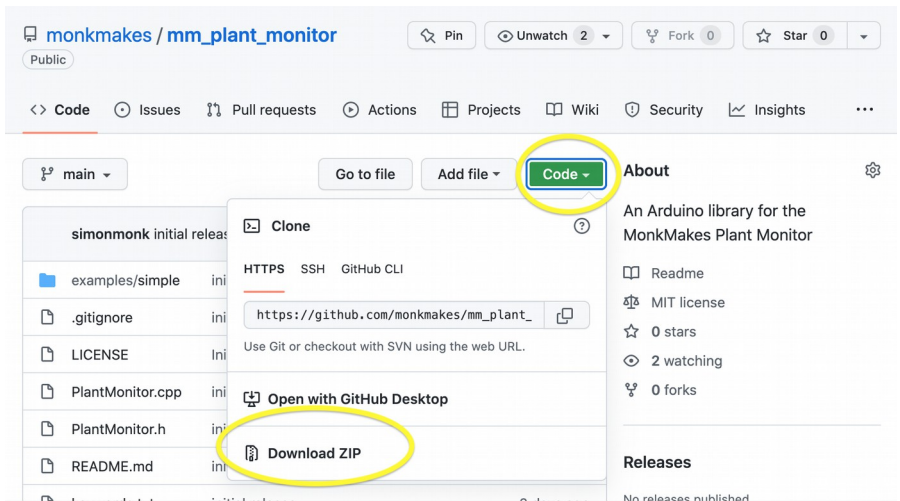
To connect a 5V Arduino, such as the Arduino Uno or Leonardo you will need to use a level converter or a (as we have here) a 1kΩ resistor to limit the current flowing from the 5V Soft Serial transmit pin of the Arduino (pin 11) to the 3.3V RX_IN pin of the Plant Monitor.

Here's what this looks like, solderless breadboard is used to hold the resistor (in the middle of the breadboard), male to male jumper wires to connect the the Arduino to

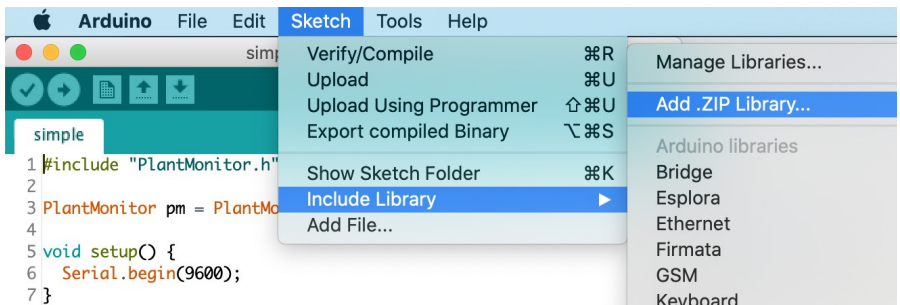
the breadboard and female to male jumper wires to connect the Plant Monitor to the breadboard. The connections are as follows:

- GND on the Arduino to GND on the Plant Monitor
- 3V on the Arduino to 3V on the Plant Monitor
- Pin 10 on the Arduino to TX_OUT on the Plant Monitor
- Pin 11 on the Arduino to RX_IN on the Plant Monitor **via a 1k Ω resistor**.
Note that the resistor is not needed for a 3V Arduino.

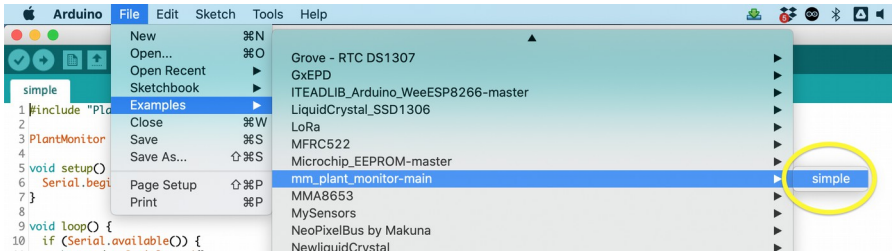
Once its all connected, you can install the Arduino library for the PlantMonitor by going to https://github.com/monkmake/mm_plant_monitor and the from the Code menu, select Download ZIP.



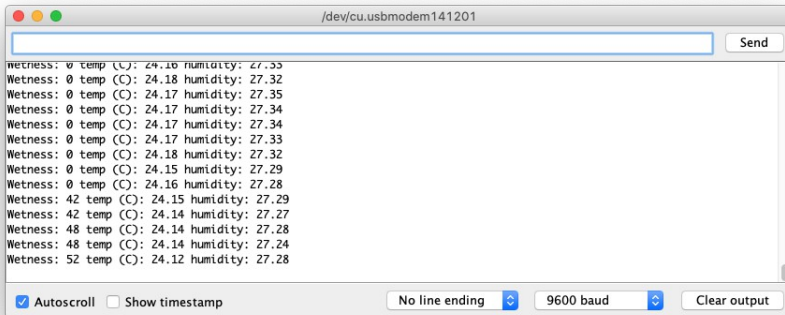
Now open the Arduino IDE and from the Sketch menu select the option to Add .ZIP Library and navigate to the ZIP file you just downloaded.



As well as installing the library, this will also fetch an example program that you will find in the Examples sub-menu of the File menu, under the category Examples from Custom Libraries.



Upload the example called *Simple* to your Arduino and then open the Serial Monitor. Here, you will see a series of readings. You can also turn the Plant Monitor's LED on and off from the Serial Monitor by sending serial commands. Type L in the send area of the Serial Monitor and then press the Send button to turn the LED on, and l (lower-case L) to turn the LED off.



Here is the code for this example:

```
#include "PlantMonitor.h"

PlantMonitor pm = PlantMonitor(10, 11); // RX, TX

void setup() {
  Serial.begin(9600);
}

void loop() {
  if (Serial.available()) {
    char cmd = Serial.read();
    if (cmd == 'l') {
```

```

        pm.ledOff();
    }
    else if (cmd == 'L') {
        pm.ledOn();
    }
}
report();
delay(1000);
}

void report() {
    Serial.print("Wetness: ");
    Serial.print(pm.getWater());
    Serial.print(" temp (C): ");
    Serial.print(pm.getTemp());
    Serial.print(" humidity: ");
    Serial.println(pm.getHumidity());
}

```

The library uses another Arduino library called `SoftSerial` to communicate with the Plant Monitor. This can carry out serial communication on any of the Arduino pins. So, when an instance of `PlantMonitor` called `pm` is created, the pins to be used to communicate to the Plant Monitor hardware are specified (in this case, 10 and 11). If you like, you can change 10 and 11 for other pins.

The main `loop` checks for incoming messages of `L` or `/` from you to turn the LED on or off respectively, using the `pm.ledOn` or `pm.ledOff` commands.

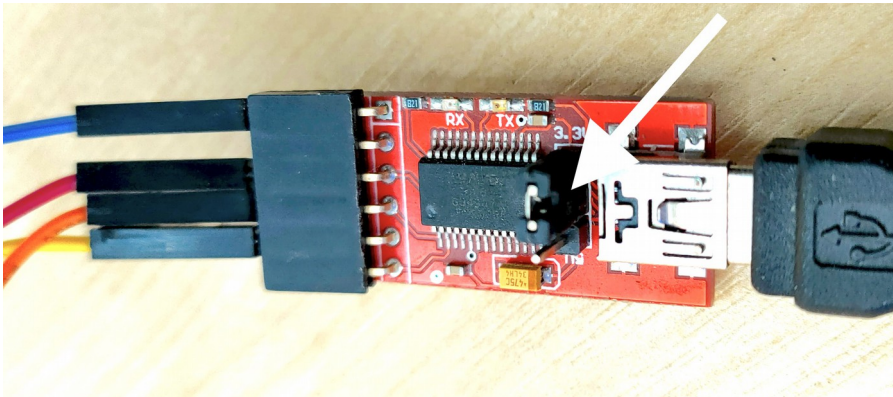
Getting readings from the `PlantMonitor` all takes place in the `report` function that writes out all the readings to the Arduino IDE's Serial Monitor.

LAPTOP WITH SERIAL ADAPTER

You can connect the Plant Monitor to your regular Windows, Mac or Linux computer using a USB to Serial converter.

You will need the following items to try out this example:

- A USB to Serial adaptor. For least trouble use an FTDI adaptor, like the one shown below. **Warning: The USB to Serial adaptor MUST either be the 3V type, or (like the one shown below) have selectable voltage, and be set to 3V. Setting it to 5V will probbably break your Plant Monitor.**
- 4 male to female jumper wires.

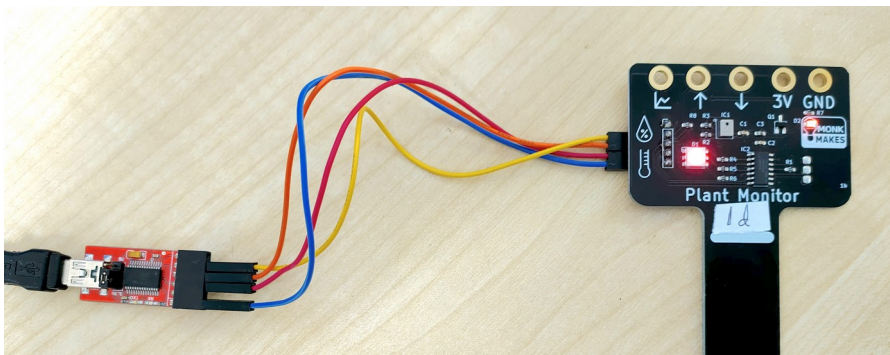


You can then, for example, use the sample Python program provided to log readings as a CSV file and then import them into a spreadsheet.

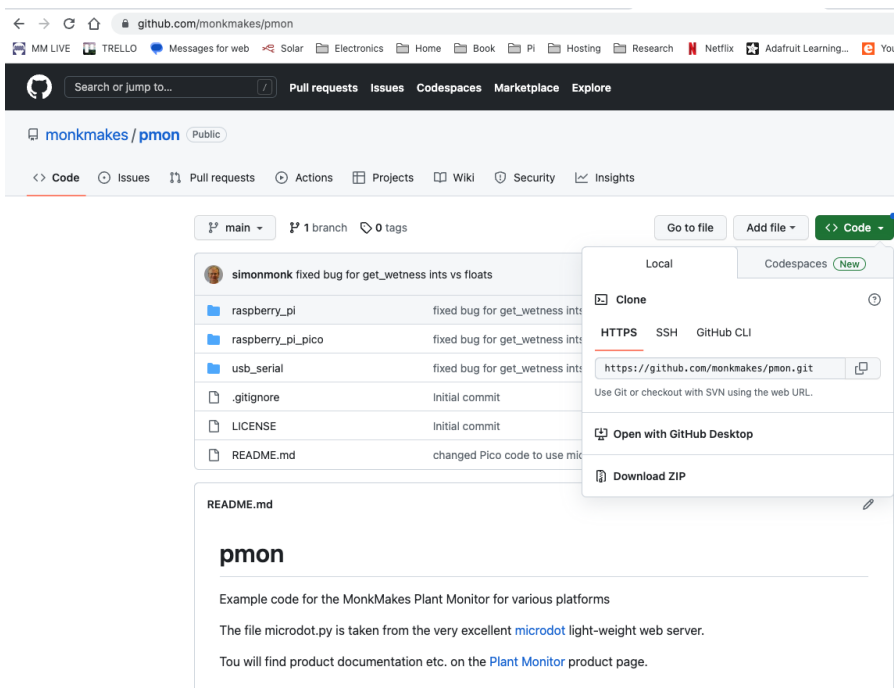
Make the following connections with the jumper wires from the USB adaptor to the Plant Monitor.

- GND to GND
- VCC on the USB to Serial adaptor to 3V on the Plant Monitor
- TXD on the USB to Serial adaptor to RX_IN on the Plant Monitor
- RXD on the USB to Serial adaptor to TX_OUT on the Plant Monitor

You can then plug the USB to Serial adaptor into your computer. The Plant Monitor's LEDs should light.



You can try out the Plant Monitor with an example datalogging example, which you will find on github. Visit <https://github.com/monkmakes/pmon> and from the Code button, select Download ZIP.



Unzip the downloaded archive, and within it you will find a folder called `usb_serial`, containing two Python files, `plant_monitor.py` and `datalogger.py`.

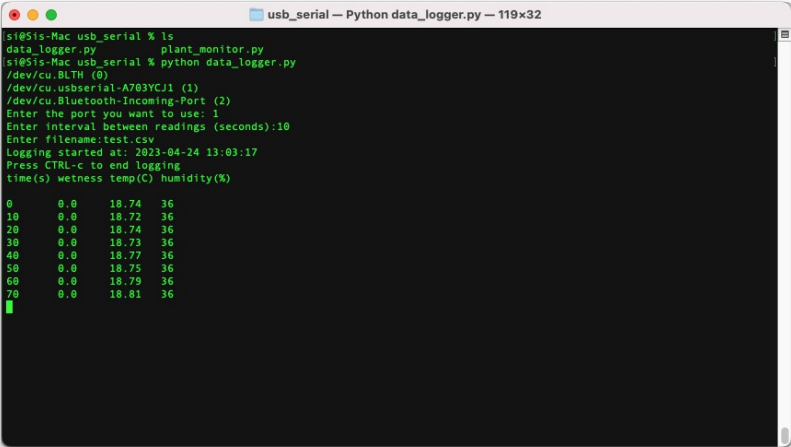
Open a terminal or PowerShell session in the same directory as the two files and run the command:

```
$ python data_logger.py
```

If you get an error message, then you may need to install the pyserial module using the following command:

```
$ pip install pyserial
```

You will be prompted to identify the serial port to use, followed by the interval between readings and the file name to use. The program will then start logging data into the file.



```
usb_serial — Python data_logger.py — 119x32
si@Si5-Mac usb_serial % ls
data_logger.py      plant_monitor.py
si@Si5-Mac usb_serial % python data_logger.py
/dev/cu.Bluetooth-Incoming-Port (0)
/dev/cu.usbserial-A703YCJ1 (1)
Enter the port you want to use: 1
Enter interval between readings (seconds):10
Enter filename:test.csv
Logging started at: 2023-04-24 13:03:17
Press CTRL-C to end logging
time(s) wetness temp(C) humidity(%)
0      0.0  18.74  36
10     0.0  18.72  36
20     0.0  18.74  36
30     0.0  18.73  36
40     0.0  18.77  36
50     0.0  18.75  36
60     0.0  18.79  36
70     0.0  18.81  36
```

When you have enough data use CTRL-c to stop the program.

You can then import the file into any spreadsheet program. (See the Raspberry Pi section).

TROUBLESHOOTING

Problem: When I first connect power to the PlantMonitor, the LED cycles through colors. Is this normal?

Solution: Yes, this is the Plant Monitor doing a self-test as it starts up.

Problem: The LED on the Plant Monitor does not light at all.

Solution: Check the power connections to the Plant Monitor. Alligator leads and jumper wires can become faulty. Try changing the leads.

Problem: I am connecting using the serial interface, and I get wetness readings, but the humidity and temperature readings are wrong and not changing.

Solution: You may have inadvertently powered your Plant Monitor from 5V rather than 3V. This may have destroyed the temperature and humidity sensor.

SUPPORT

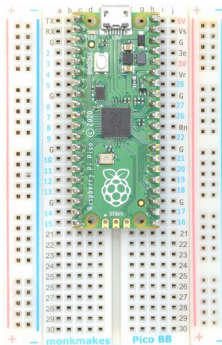
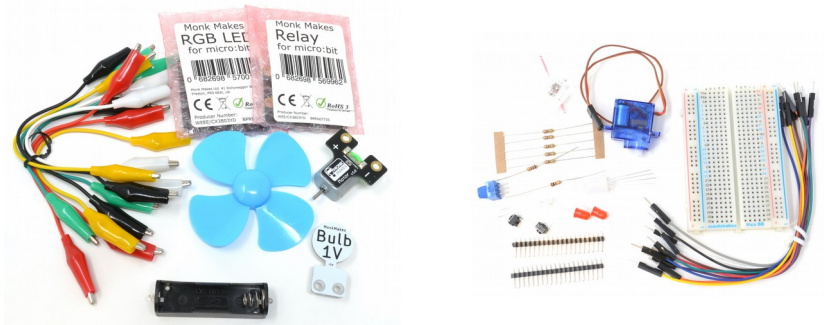
You can find the Product's information page here: <https://monkmakes.com/pmon> including a datasheet for the product.

If you need further support, please email support@monkmakes.com.

MONKMAKES

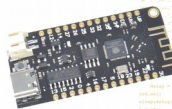
As well as this kit, MonkMakes makes all sorts of kits and gadgets to help with your electronics projects. Find out more, as well as where to buy here:

<https://monkmakes.com> you can also follow MonkMakes on Twitter @monkmakes.



Programming ESP32

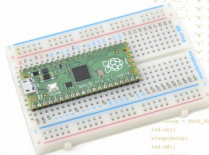
Learn MicroPython Coding and Electronics



Simon Monk

Programming the Pico

Learn Coding and Electronics with the Raspberry Pi Pico



Simon Monk

