The MonkMakes Plant Monitor and CircuitPython

Created by Simon Monk

https://learn.adafruit.com/monkmakes-plant-monitor-and-circuitpython

Last updated on 2023-08-29 03:27:42 PM EDT
## Table of Contents

**Overview**  
• Parts

**Pinout**  

**Wiring**  

**CircuitPython**  
• Install or update CircuitPython!  
• Further Information

**Code**  

**Plant Pots**  

**Plant Monitor Protocol**  

© Adafruit Industries
Overview

The MonkMakes Plant Monitor is a capacitive moisture meter that also measures temperature and relative humidity. Because it uses a capacitive moisture sensor, there is no electrical contact with the soil, so the probe doesn't corrode in the way that resistive sensors do.

In this guide, you will learn how to use the Plant Monitor's serial (UART) interface with CircuitPython. As an example, you'll use the ten Neopixel LEDs on a Circuit Playground Express to indicate the moisture level.

Parts

Monk Makes Plant Monitor - Capacitive Moisture Meter
If you're like me and have trouble keeping your calatheas and pepperomias happy and not overwatering them to root rot ruin, why not take a gander at the...
https://www.adafruit.com/product/5587

Circuit Playground Express
Circuit Playground Express is the next step towards a perfect introduction to electronics and programming. We've taken the original Circuit Playground Classic and...
https://www.adafruit.com/product/3333
Small Alligator Clip Test Lead (set of 12)
Connect this to that without soldering using these handy mini alligator clip test leads. 15” cables with alligator clip on each end, color coded. You get 12 pieces in 6 colors....
https://www.adafruit.com/product/1008

USB cable - USB A to Micro-B
This here is your standard A to micro-B USB cable, for USB 1.1 or 2.0. Perfect for connecting a PC to your Metro, Feather, Raspberry Pi or other dev-board or...
https://www.adafruit.com/product/592

---

**Pinout**

The MonkMakes Plant Monitor can be connected with alligator clip leads (great for the Circuit Playground Express or micro:bit) or header pins, which are easier for boards designed to be breadboard friendly like our Feather range of boards or the Raspberry Pi Pico.

The MonkMakes Plant Monitor is 3.3V only. Do not connect it to a 5V supply. Doing so will probably break it.
On the front side of the board, the rings have the following functions (from left to right):

- ANALOG output: This can be connected to the analog input of a microcontroller and outputs a voltage of between 0 (bone dry) and 2.5V (very wet). If you don't care about using the Plant Monitor's temperature or relative humidity (of the air) features, then this can be a simple way to interface to the board
- TX_OUT: UART serial output from the Plant Monitor. If you are using this to communicate with your microcontroller board, then this should be connected to the RX (receive) pin of the microcontroller
- RX_IN: UART serial input. This should be connected to the TX (transmit) pin of the microcontroller
- 3V: Connect to a 3.3V microcontroller power supply.
- GND: Ground connection
The header pin connections on the back of the Plant Monitor mirror the connections provided by the alligator clip ring connectors. You can use female to male header leads to connect these to a solderless breadboard.

Wiring

Use alligator clip leads to connect the Plant Monitor to your Circuit Playground Express. Note that the choice of lead color is just to match up with the photo below.

- Black - GND on the Plant Monitor to GND on the Circuit Playground Express. I suggest the GND near the battery connector of the Circuit Playground Express
- Red - 3V on the Plant Monitor to 3.3V on the Circuit Playground Express
- Yellow - RX_IN of the Plant Monitor (downwards arrow) to the TX A7 ring of the Circuit Playground Express
- Green - TX_OUT of the Plant Monitor to the RX A6 ring of the Circuit Playground Express
As we continue to develop CircuitPython and create new releases, we will stop supporting older releases. If you are running an older version of CircuitPython, you need to update. Click the button below to download the latest!

Install or update CircuitPython!

Follow this quick step-by-step for super-fast Python power :) 

Download the latest version of CircuitPython for this board via CircuitPython.org

Click the link above and download the latest UF2 file

Download and save it to your Desktop (or wherever is handy)
Plug your Circuit Playground Express into your computer using a known-good USB cable.

A lot of people end up using charge-only USB cables and it is very frustrating! So make sure you have a USB cable you know is good for data sync.

Double-click the small Reset button in the middle of the CPX, you will see all of the LEDs turn green. If they turn all red, check the USB cable, try another USB port, etc.

(If double-clicking doesn't do it, try a single-click!)

You will see a new disk drive appear called CPLAYBOOT.

Drag the adafruit-circuitpython-etc...uf2 file onto it.
The CPLAYBOOT drive will disappear and a new disk drive will appear called CIRCUITPY

That's it! You're done :)

Further Information

For more detailed info on installing CircuitPython, check out Installing CircuitPython().

Code

The Plant Monitor receives single command letters over the serial connection and responds to the command letter. So, for example, sending the letter w causes the Plant Monitor to read the wetness level and respond with a message w=67 where the number after the = is the wetness as a percentage. Other commands are t for temperature, h for humidity and j which returns a JSON representation of wetness, temperature and humidity in one message.

This human readable message format, means that we can experiment by sending commands to the Plant Monitor using Mu's REPL().

Connect your Circuit Playground Express to your computer with a known good data+power USB cable. Run Mu and click the Serial button in the list of buttons at the top of the editor window. You should see the Mu window split to show the serial window towards the bottom. If you do not get the >>> prompt, press the control key and then the c key at the same time.
This is a CircuitPython REPL session. Type in the following commands after the `>>>` prompts:

```python
import board
import busio
uart = busio.UART(board.TX, board.RX, baudrate=9600, timeout=0.1)
uart.write(b"h")
uart.read()
```

After we have made the necessary imports of `board` and `busio`, an instance of `busio.UART` is created. This specifies the pins to use (in this case the default RX and TX pins for the board) and the baud rate of 9600 (expected by the Plant Monitor). The `timeout` parameter shortens the 2 second default timeout that would otherwise occur.

The line `uart.write(b"h")` writes the byte array string of `h` (for humidity) to the Plant Monitor.

If you are not getting responses, check your wiring.
To read the Plant Monitor’s response use `uart.read()`. In this case, you can see that the response message contains the humidity value of **33.24** followed by the end of line characters.

The wetness can be read by sending the `w` command. But before running the first of the commands below, grip the prong of the Plant Monitor in the palm of your hand (see above). This will register the moisture of your hand so that the reading isn’t **0**.

```python
>>> uart.write(b"w")
1
>>> uart.read()
b'w=74\r\n'
>>> uart.write(b"w")
1
```

Here you can see that the wetness level is reported as **74%**.

If you wanted to read the temperature, you would send the message `t`. Note that this is in degrees Celsius. To convert this into Fahrenheit, multiply the number by 9, divide the result by 5 and then add 32.

Here’s how use serial communication in a program that prints out the wetness reading roughly once a second.

```python
import board
import busio
from time import sleep

uart = busio.UART(board.TX, board.RX, baudrate=9600, timeout=0.1)

while True:
    uart.write(b"w")
    wetness_str = uart.read()
    # b'w=53\r\n'
    wetness = float(wetness_str[2:-2]) # extract number from response
```
Inside the main while loop, the `w` message is sent and the result read. We are only interested in the number part of the message, so we can chop off the first two characters (`w=`) and the last two end of line characters with the command `wetness_str[2:-2]` before converting the string into a float.

The following example program makes use of the Circuit Playground Express built-in NeoPixels to display the wetness level.

```python
import board
import busio
import neopixel
from time import sleep

num_leds = 10
uart = busio.UART(board.TX, board.RX, baudrate=9600, timeout=0.1)
pixels = neopixel.NeoPixel(board.NEOPIXEL, num_leds, brightness=0.2, auto_write=False)

RED = (255, 0, 0)
YELLOW = (255, 150, 0)
GREEN = (0, 255, 0)
OFF = (0, 0, 0)

def show_wetness(percent):
    led_index = int((percent - 1) / 10)
    color = RED
    if led_index &gt; 6:
        color = GREEN
    elif led_index &gt; 3:
        color = YELLOW
    print(led_index)
    for i in range(0, num_leds):
        if i &lt;= led_index:
            pixels[i] = color
        else:
            pixels[i] = OFF
    pixels.show()

while True:
    uart.write(b"w")
    wetness_str = uart.read()
    # b'w=53\r\n'
    wetness = int(wetness_str[2:-2]) # extract number from response
    show_wetness(wetness)
```

The code for reading the wetness is the same as the previous example. The code that displays the wetness is in the `show_wetness` function. This calculates the number of LEDs to light (``led_index``) as the percentage wetness minus 1 divided by 10 to give an range of 0 to 9, corresponding to the Circuit Playground Express' 10 NeoPixel LEDs.

The function then chooses a color of `RED`, `YELLOW` or `GREEN` using the value of `led_index`.
The \texttt{for} loop goes through all 10 LED positions, either lighting the LED in the chosen color (if its index position is less than or equal to \texttt{led\_index}) or otherwise turning that LED position off.

---

**Plant Pots**

Before you put the Plant Monitor in the pot, you may try gripping the prong in your hand and the moisture of your body should be enough to alter the LED's color as a test.

When placing the Plant Monitor in a plant pot, 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 built-in LED. Red on the large LED to the left means it is dry and needs water.
Adding some water to a dry plant provides moisture. The left multicolor LED turns green as it detects the added moisture.

Plant Monitor Protocol

Although we have just used the w, t and h commands, the Plant Monitor can respond to some other commands too.

Here's a summary of the commands.

<table>
<thead>
<tr>
<th>Command</th>
<th>Response</th>
<th>Notes</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w=55 \n returns the moisture level 0 to 100 %</td>
</tr>
<tr>
<td>t</td>
<td>t=20.5 \n returns the temperature in degrees C</td>
</tr>
<tr>
<td>h</td>
<td>h=50.5 \n returns the relative humidity %</td>
</tr>
<tr>
<td>j</td>
<td>{&quot;wetness&quot;:57, &quot;humidity&quot;:50, &quot;temp&quot;:20} \n returns all three readings in JSON format</td>
</tr>
<tr>
<td>L</td>
<td>no response turn on the Plant Monitor's LED to display the current moisture. Red dry (0%), green wet (100%), a mix of red and green in between.</td>
</tr>
<tr>
<td>l (lower case L)</td>
<td>no response turn off the Plant Monitor's built-in LED</td>
</tr>
<tr>
<td>v</td>
<td>1d          This reports the Plant Monitor's firmware version.</td>
</tr>
</tbody>
</table>

The Plant Monitor's firmware is open source and you can find it here: [https://github.com/monkmakes/plant_monitor_firmware]()