circuitpython_kernel

December 1, 2018

1 Using CircuitPython with Jupyter

This project is a fork of Adafruit's Jupyter kernel with a few enhancements (like reconnecting to the CP VM if the connection is lost). Until these changes have been incorportated in Adafruit's distribution use the instructions below instead of those on Adafruit's website.

1.1 Installation

Requires Python 3, then run the following commands from the command line:

```
pip3 install cp_kernel
```

Then run start jupyter lab with

```
jupyter lab
```

A browser window opens with the Jupyer Lab GUI. The console windows displays verbose information that you can ignore except when you experience problems.

1.2 CircuitPython Notebook

Choose `File->New->Notebook` from the menu and select the `CircuitPython` kernel. Make sure a microcontroller board is connected to the computer via USB. The kernel finds the board automatically; check the console output in case of problems.

Use the notebook as usual, except that now code in cells is uploaded to the board for evaluation with CircuitPython.

Few differences:

- No output is printed except that from explicit `print` statements
- The results from evaluation are displayed all at once, not incrementally during code execution
- Magic commands do not work, except those described below

Occasionally you may get syntax error messages for code that is clearly correct. This is likely due to the microcontroller not being able to keep up with the code upload and occurs in particular with large cells.

The solution is to add a delay between each line of code as it is being sent to the controller. Executing the cell below will set this delay to 0.1 seconds.

```
In [1]: %upload_delay 0.1
```

Execute Python code as usual:

```
In [2]: for i in range(3):
            print(i)
```

```
0
1
2
```

Error messages are marked in the notebook output by coloring:

```
In [3]: for i in range(5):
            print(i)
            if i>3:
                raise IndexError("error demo")

        print("never gets here")
```

```
0
1
2
3
4
```

```
Traceback (most recent call last):
  File "<stdin>", line 4, in <module>
IndexError: error demo
```

Longer calculations output results as they become available:

```
In [4]: from time import sleep

        for i in range(5):
            print(i)
            sleep(1)
```

```
0
1
2
3
4
```

1.3 Working with GPIO

The code below blinks an led (change `D7` to match your hardware). If you run this once, all works as expected, but if you execute the cell a second time you get an error:

```
Traceback (most recent call last):
  File "<stdin>", line 7, in <module>
ValueError: D7 in use
```

To fix it, you can either deinit the led (and other resources your code is using) before running the cell, or, simpler, perform a soft reset `%softreset` as shown below:

```
In [6]: %softreset

        from board import D7
        from time import sleep
        import digitalio

        led = digitalio.DigitalInOut(D7)
        led.direction = digitalio.Direction.OUTPUT

        led.switch_to_output(value=False, drive_mode=digitalio.DriveMode.PUSH_PULL)

        for i in range(10):
            led.value = not led.value
            sleep(0.3)
```