ATuino

In this tutorial we are going to remotely control an Arduino from a python-script

The hardware-components will be modeled in an object-oriented manner to provide a nice and reusable interface

Enable native scrolling

atuino.ino

In order to remotely control the Arduino it has to be flashed with a firmware that accepts and executes commands

```
at_commands[] = {
  {.name= "AT",            .cb= atcb_at},
  {.name= "AT+SET_OUT",    .cb= atcb_setout},
  {.name= "AT+SET_IN",     .cb= atcb_setin},
  …
  {.name= NULL, .cb=NULL}
};
```

…

The sketch above accepts AT-style commands that can be used to read and write digital pins

atuino.ino

Upload the sketch from the previous slide to your Arduino and configure the serial monitor to send a carriage return ␍ and a newline ␊ character at the end of each line, and to use a speed of `115200 Baud`

```
AT+SET_OUT=13
AT+WRITE_HIGH=13
```

Enter the commands above into the serial monitor to turn on the Arduino on-board LED

The Arduino should acknowledge both commands with an `OK` and turn on the LED

atuino.ino

Study the commands in the `at_commands` array

find the commands that let you read out the `HIGH` or `LOW` state of pin `10`

Use a piece of wire to change the state of pin `10`

atduino.py

```
duino= Arduino('/dev/ttyACM0')

key1= InputPin(duino, 10, True)
#led1= OutputPin(duino, 13)
```

...

The code above contains class definitions that model the connection to an Arduino and its input pins

Change `'/dev/ttyACM0'` to the name of your serial port, run the program and observe its output

Hint: You can use the Arduino IDE to find the name of your serial port

class OutputPin

Write a class `OutputPin` that, analogous to the class `InputPin`, models an output pin

The class should provide a method `set_state` that turns the output on or off

When you are done, uncomment the commented-out lines, to verify that your implementation works as expected

Hint: you can use the interactive commandline to test your implementation `python3 -i 16_atuino.py`

Complete Example

```
class OutputPin(object):
    def set_state(self, state):
        self.arduino.exec_cmd(
            'WRITE_HIGH' if state else 'WRITE_LOW',
            self.pin_no
        )
```

...