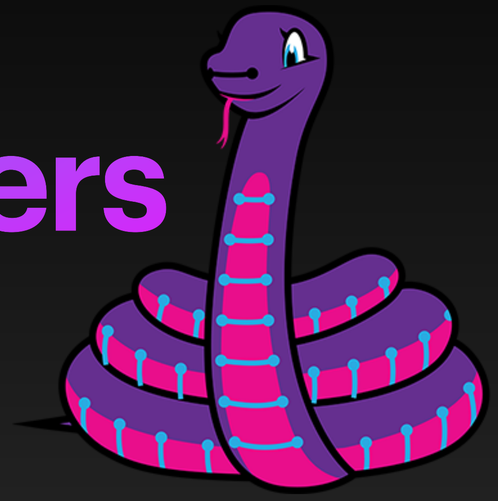
Programming Microcontrollers using CircuitPython



@iAyanPahwa

AP, HydPy-meetup , May 29th 2020

/iayanpahwa

ABOUT ME

- Automotive Embedded Software Engineer
- FPV Drones
- Maker
- Volunteer HHC-D and ILUG-D

codeNsolder.com

github.com/iAyanPahwa

instagram.com/iAyanPahwa

twitter.com/iAyanPahwa

youtube.com/iAyanPahwa

Motivation

The people who are really serious about software should make their own hardware

- Alan Kay

COOL! But what’s the problem?

You will need

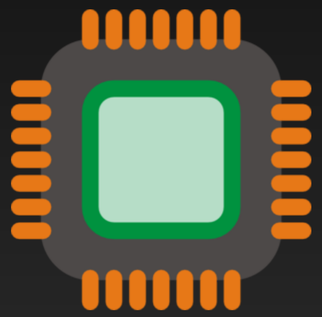
Flasher

Compiler

C Libray

Drivers

IDE



Linker

Serial Console

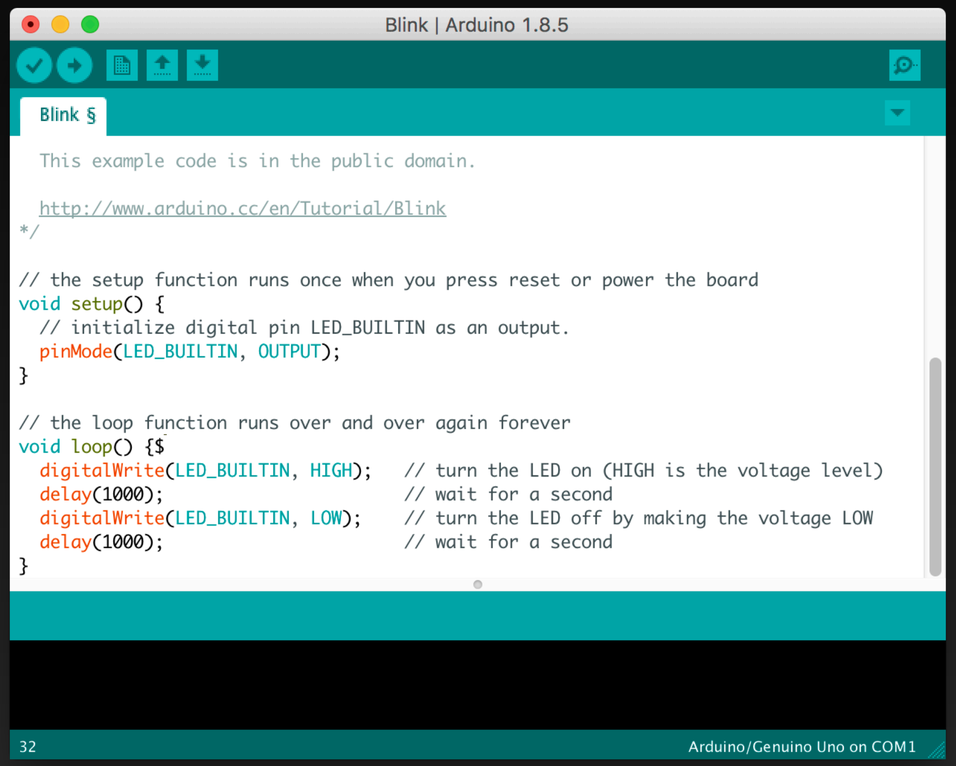
Debugger

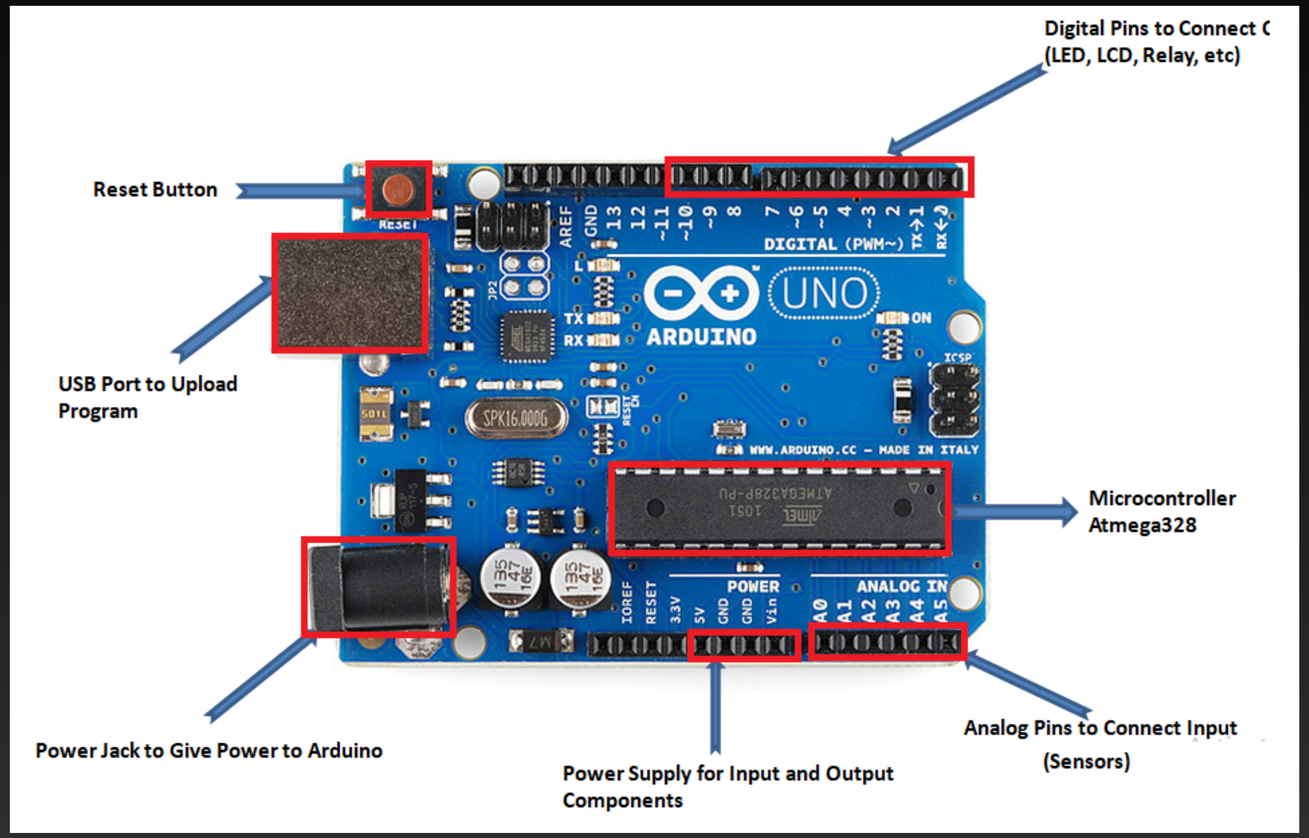
Bootloader

Programmer

BUT I JUST WANTED TO CONTROL LIGHTS OF MY ROOM :/

What about Arduino?

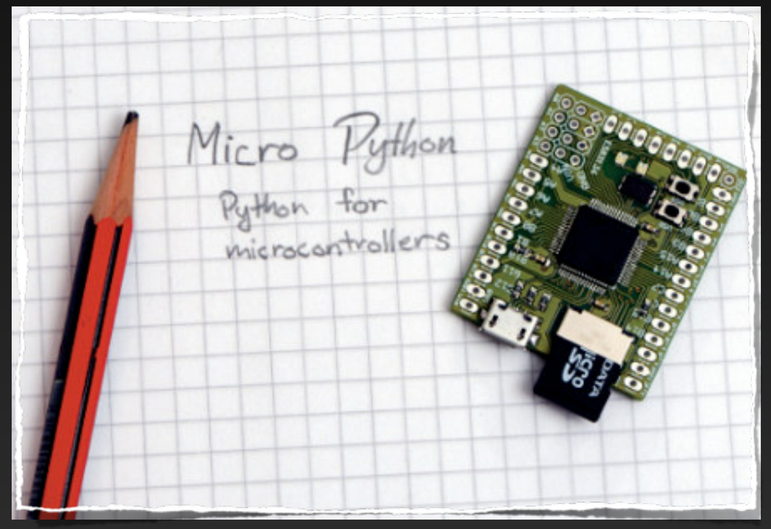




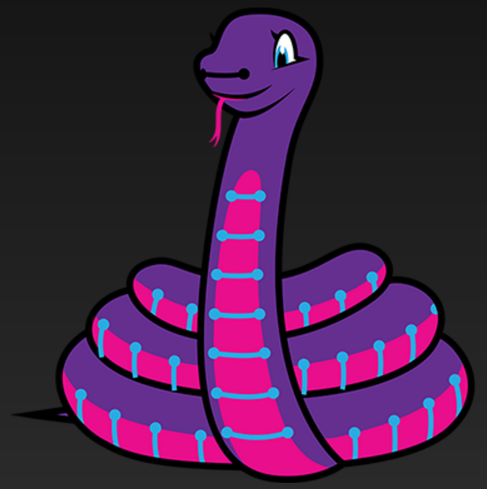
Well, it’s still compiled and use USB Serial (CDC class) drivers

Enter Python on Hardware

How it all started?



MicroPython

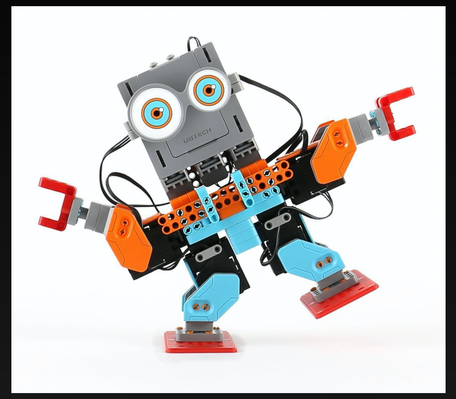


CircuitPython

Great, But why this is a huge deal?

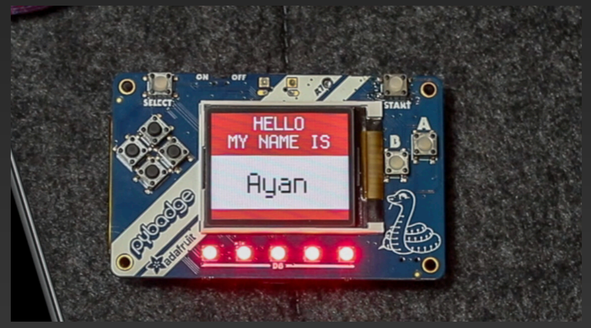
Opportunities

Smart Home



Robotics

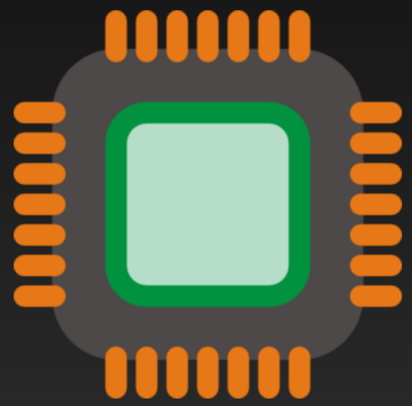
Cosplay



Conference Badge

Weather station

Microcontroller



- ~ 20-120 MHz System Clock
- ~ 32Kb RAM
- ~16MB ROM
- Single Core
- Register Level Access

OKAY, but why not CPython ?

- CPython is made for Desktops
- Extensive memory usage
  - ex: int object – 16 bytes on heap
- Not optimised for constraint devices

Difference - MicroPython & CircuitPython

Target:

MicroPython - General

CircuitPython - Education

API:

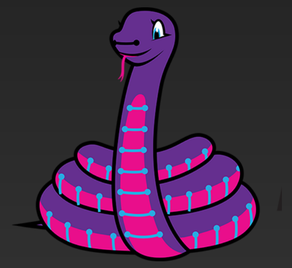
MicroPython - SoC dependent

CircuitPython - Uniform

Goals:

MicroPython - Efficiency

CircuitPython - Ease



Access:

MicroPython - SoC dependent

CircuitPython - Uniform (.uf2)

Learn More- https://github.com/adafruit/circuitpython

Components of CircuitPython

- A board with supported Microcontroller
  - Microchip ATSAMD21/52
  - Nordic NRF52840
  - STM32 (newly added)
  - ESP32Se (newly added)

Native USB support

Learn More- https://circuitpython.org/downloads

- UF2 Bootloader — Double click RESET button to enter bootloader moder

Learn More- https://github.com/microsoft/uf2-samdx1

- CircuitPython itself — The actual interpreter

Learn More- https://circuitpython.org/downloads

Behavior

- The order that files are run and the state that is shared between them. CircuitPython's goal is to clarify the role of each file and make each file independent from each other.
- `boot.py` (or `settings.py`) runs only once on start up before USB is initialized. This lays the ground work for configuring USB at startup rather than it being fixed. Since serial is not available, output is written to `boot_out.txt`.
- `code.py` (or `main.py`) is run after every reload until it finishes or is interrupted. After it is done running, the vm and hardware is reinitialized. **This means you cannot read state from `code.py` in the REPL anymore.** CircuitPython's goal for this change includes reduce confusion about pins and memory being used.
- After `code.py` the REPL can be entered by pressing any key. It no longer shares state with `code.py` so it is a fresh vm.
- Autoreload state will be maintained across reload.
- Adds a safe mode that does not run user code after a hard crash or brown out. The hope is that this will make it easier to fix code that causes nasty crashes by making it available through mass storage after the crash. A reset (the button) is needed after its fixed to get back into normal mode.
- RGB status LED indicating CircuitPython state, and errors through a sequence of colored flashes.
- Re-runs `code.py` or other main file after file system writes over USB mass storage. (Disable with `samd.disable_autoreload()`)
- Entering the REPL after the main code is finished requires a key press which enters the REPL and disables autoreload.
- Main is one of these: `code.txt`, `code.py`, `main.py`, `main.txt`
- Boot is one of these: `settings.txt`, `settings.py`, `boot.py`, `boot.txt`

Enough Talk, Let’s get some action going

So I need to buy a development board hah?

So what all can I do with CircuitPython?

What I can’t do with CircuitPython?

Show me more?

Thank You

Let's connect

codeNsolder.com

github.com/iAyanPahwa

instagram.com/iAyanPahwa

twitter.com/iAyanPahwa

youtube.com/iAyanPahwa