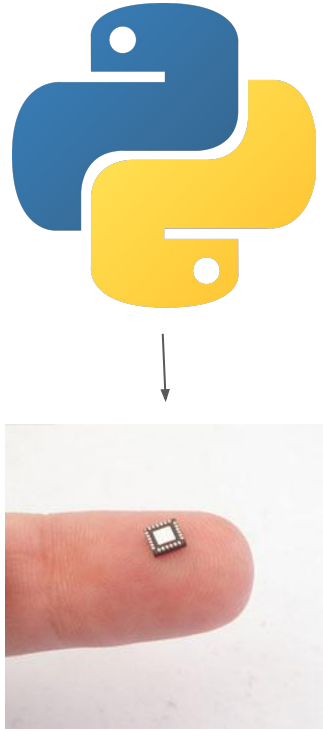
Python for Microcontrollers

Ronald Pulliam
Flexware Innovation, Inc

Outline:

1. Why Python for Embedded
2. What is MicroPython
3. MicroPython History
4. CircuitPython
5. Supported Boards
6. How to use
7. Projects



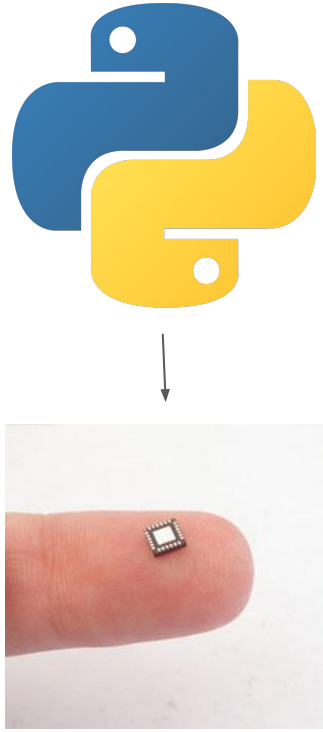
Why Python for Embedded:

Why Python:

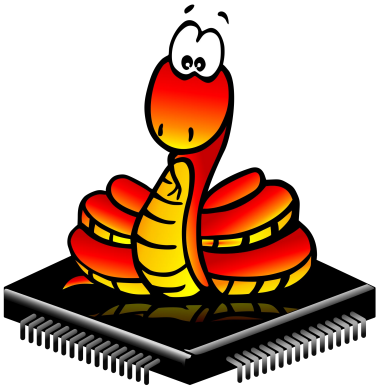
- High level language
- Large existing community
- Easy to learn, but powerful

Why microcontrollers:

- Power management
- Fast execution
- Small (size, cost, complexity)



What is MicroPython:



MicroPython is

- A complete reimplementation of Python (written in C)
- Designed to be efficient with resources
- Designed to run on bare metal

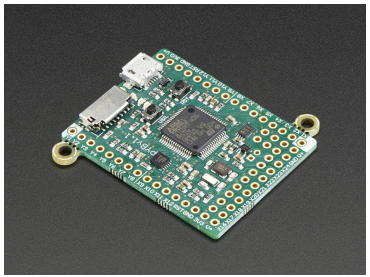
MicroPython includes:

- A compiler, runtime, and familiar REPL
- Support for basic libraries (modules), most begin with ‘u’
- Extra modules to control hardware

MicroPython

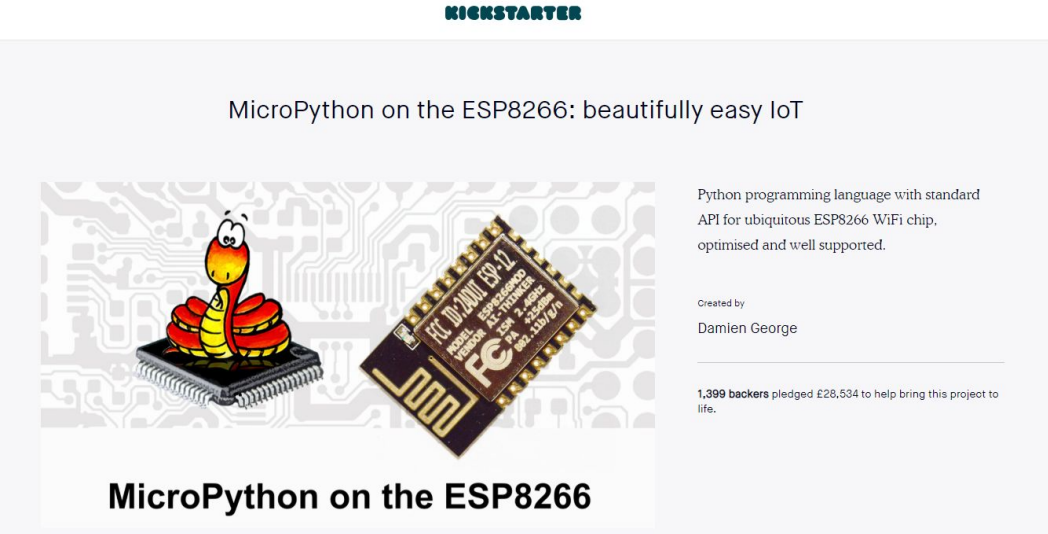
MicroPython History:

- By Australian physicist Damian George while postdoc at Cambridge
- First line of code: April 30, 2013
- Kickstarter launched: November 13, 2013
  - Ran for 30 days
  - Total backers: 1,931
  - Total raised: $128,000
  - Included pyboard hardware
- Officially finished April 12, 2015



MicroPython History cont.:

- Second Kickstarter launched: January 27, 2016
  - Software only
  - Port to ESP8266
  - Total backers: 1,399
  - Total raised: $37,500



MicroPython Good & Bad:

Benefits:

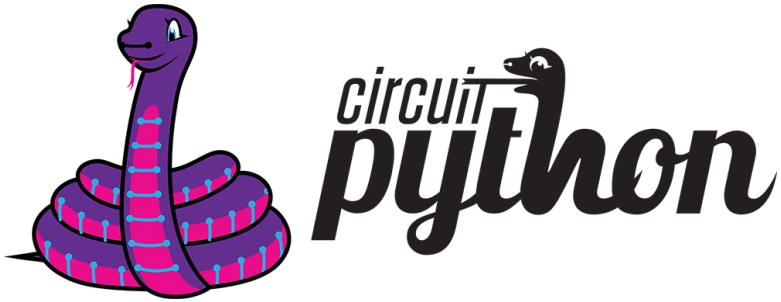
- Scripting Language
- Easy to learn
- Rapid prototype
- Time to market
- Easy extensibility
- Compiles on chip
- Portable
- MIT license
- Faster, small, more efficient than py

Drawbacks

- Not good for small microcontrollers (e.g. AVR)
- Increased resources over C, Assembly
- Lack of devs in embedded
- Dynamically typed language (garbage collection, fragmentation)
- Not good for large projects (embedded linux)

MicroPython Derivative: CircuitPython

- Created by Adafruit Industries
- Education friendly - Easier for Noobs
- Developed for Atmel SAMD21 processors
  - Onboard USB support
- Separation from runtime and REPL

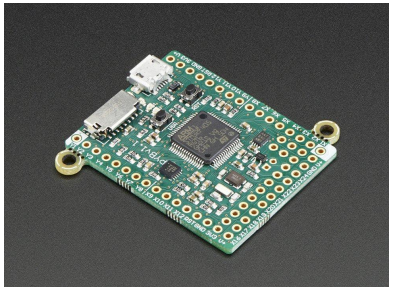


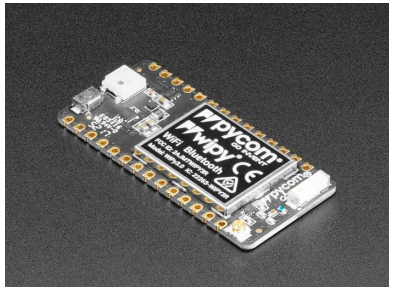
MicroPython for those coming up from C/C++

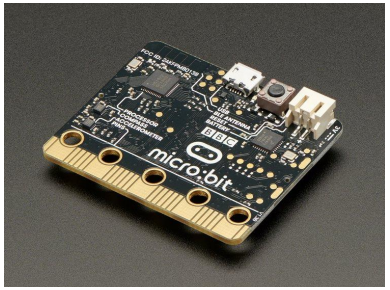
CircuitPython for those coming down from Python

Zerynth - Also Python on Microcontrollers, different implementation, closed

Supported Boards:







- **pyboard**: STM32 Cortex M4F, 168MHz, 192k RAM, 1Mb flash
- **pyboard2**: STM32F7 M7F, 216Mhz, 512k RAM, 2Mb flash
- **WiPy**: Cortex M4, 80 MHz, 256k RAM, 2Mb flash
  - 3.0 - ESP32
- **Micro:bit**: Nordic nRF51822 Cortex M0, 16MHz, 256k flash, 16k RAM
- **ESP8266**: Xtensa LX106, 80-160 MHz, 96k RAM, up to 16Mb flash
- **ESP32**: Xtensa LX6, 160-240 MHz, 520k RAM, up to 16Mb flash

CircuitPython: Big collection of Adafruit boards, some SparkFun, Particle, Arduino

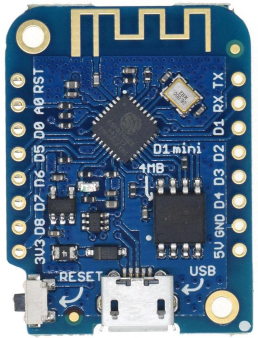
As of Release 4.0 no WiFi support (can add as Co-Processor)

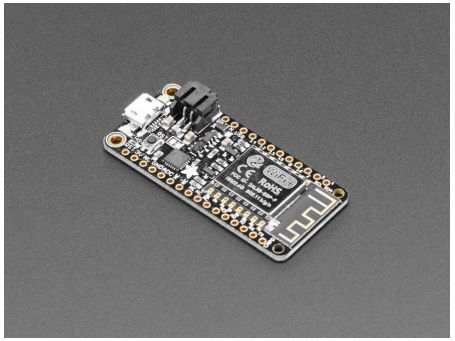
How to use: Espressif

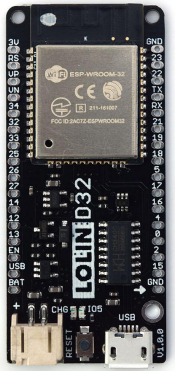
Wemos D1 mini, Lolin D32
    23+ Shields

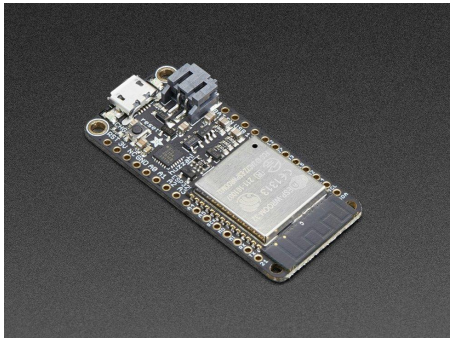
Tools

- esptool.py
- SSH, SCP, Putty
- WebREPL



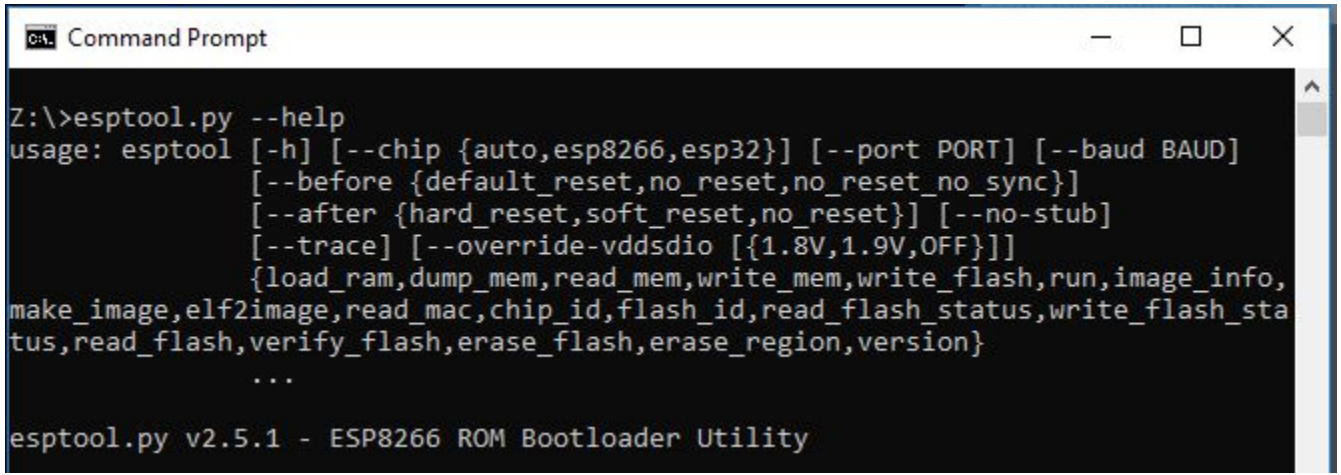






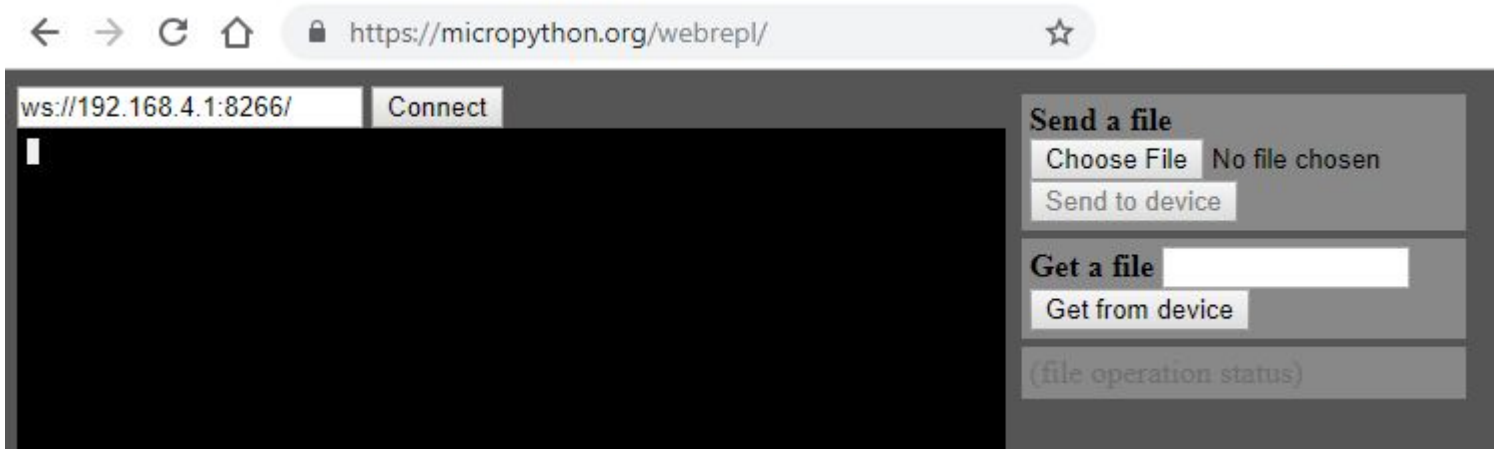
How to use: esptool.py

esptool.py -p COM3 -b 115200 {command}



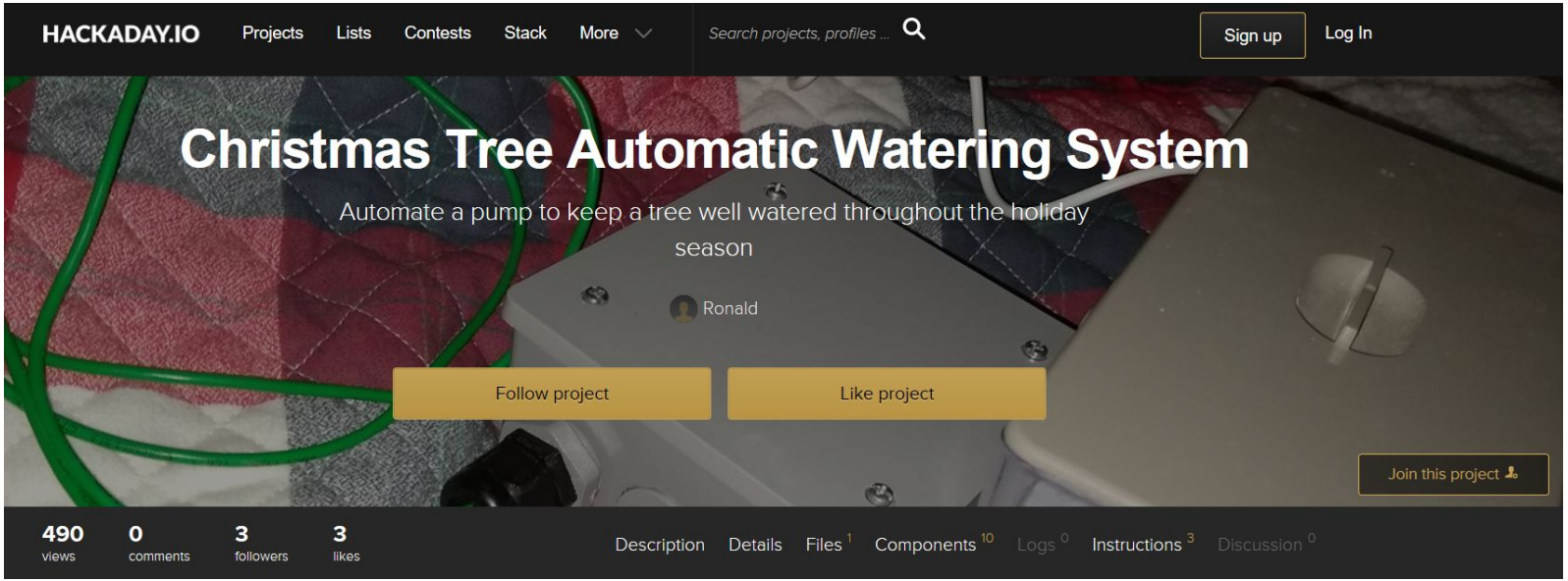
How to use: WebREPL

import webrepl_setup

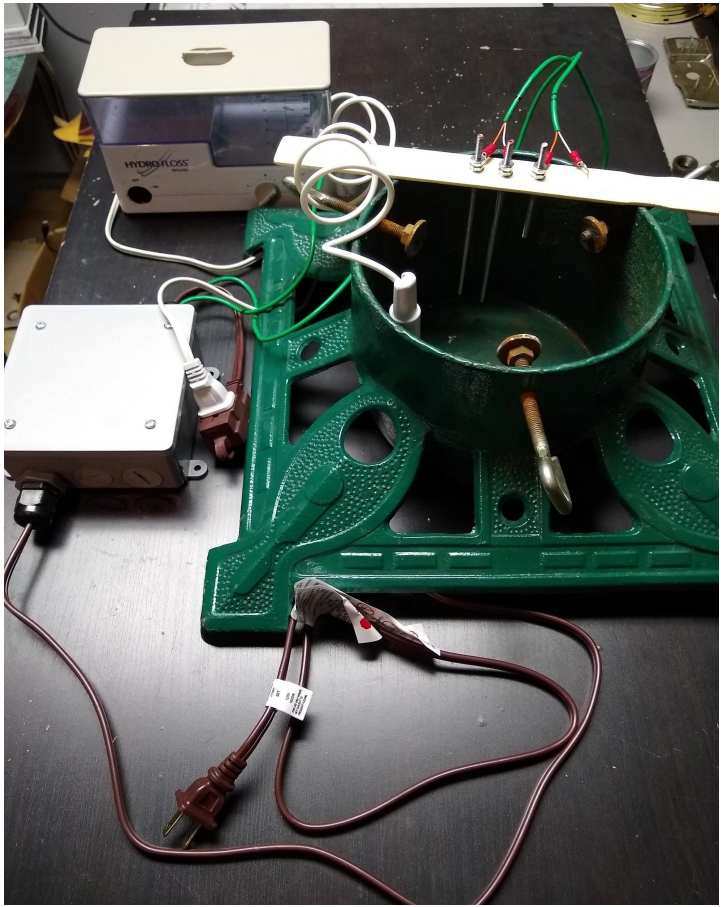


Project: Auto Xmas Tree Watering

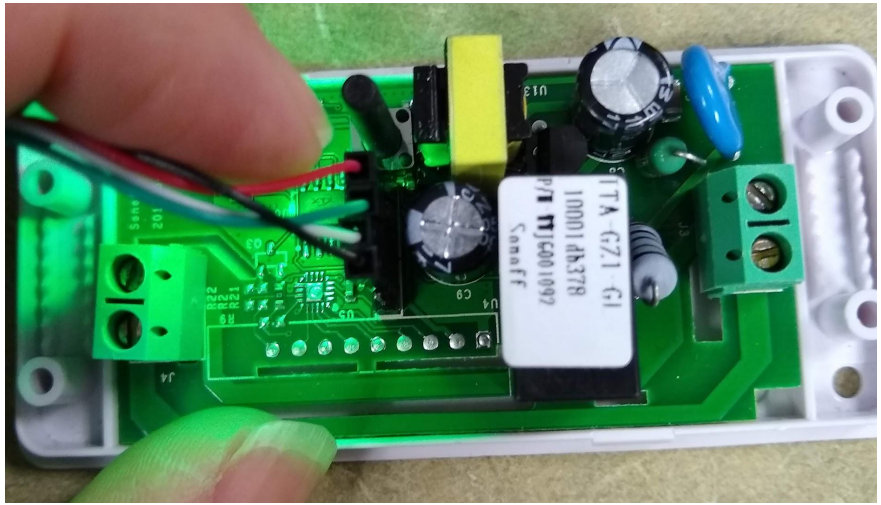
https://hackaday.io/project/162538-christmas-tree-automatic-watering-system



Project: Auto Xmas Tree



Project: Auto Xmas Tree



Project: Auto Xmas Tree Watering

```
##############################################################################
# Setup code goes below, this is called once at the start of the program: #
##############################################################################
import time
import machine

button = machine.Pin(0) # 0 = pressed
relay = machine.Pin(12, machine.Pin.OUT) #relay.on(), relay.off()
led = machine.Pin(13, machine.Pin.OUT) #led.value(1) = off, led.value(0) = on
pin = machine.Pin(14, machine.Pin.IN, machine.Pin.PULL_UP) #extra pin on header
tx = machine.Pin(1, machine.Pin.IN, machine.Pin.PULL_UP) #tx pin on header
#rx = machine.Pin(3, machine.Pin.IN, machine.Pin.PULL_UP) #rx pin on header

status = 0
timer = 0
timeout = 30
```

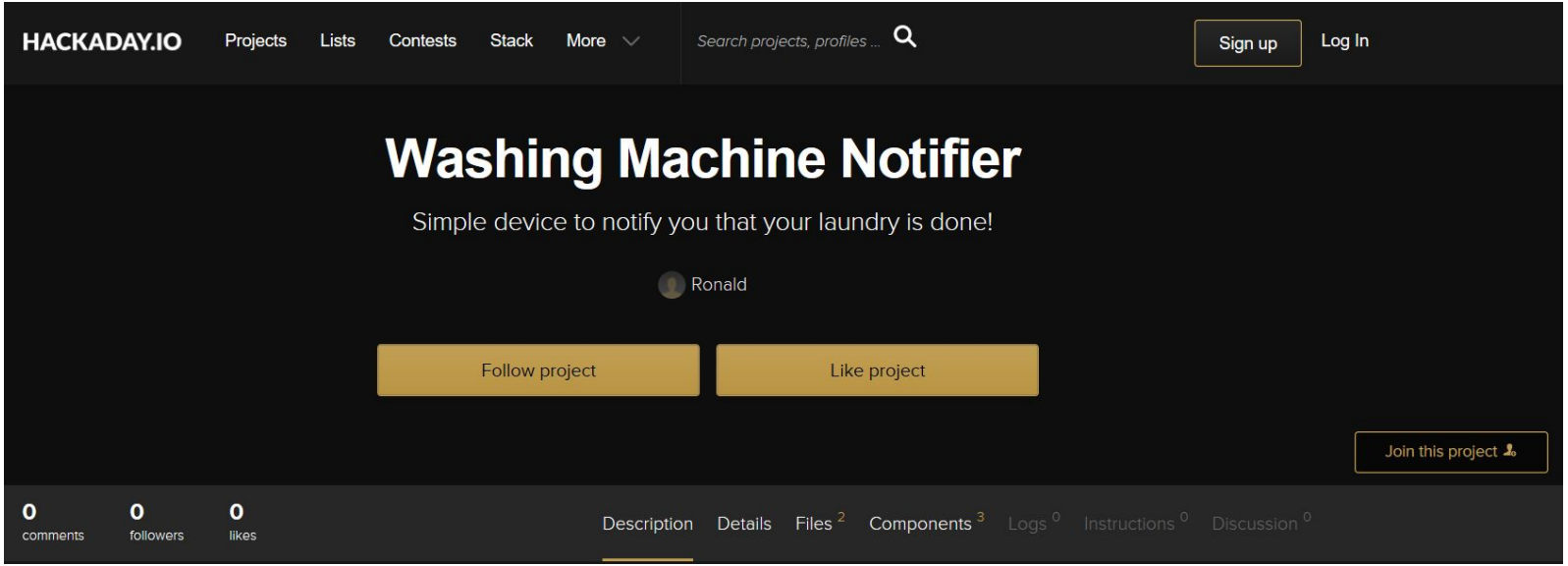
Project: Auto Xmas Tree Watering

```
#########################################################################
# Loop code goes inside the loop here, this is called repeatedly: #
#########################################################################
while True:
  time.sleep(1.0)

  #Check Inputs
  if pin.value():
        if tx.value():
                status = 0 #Empty
        else:
                status = 1
  else:
        if tx.value():
                status = 2
        else:
                status = 3 #Full
```

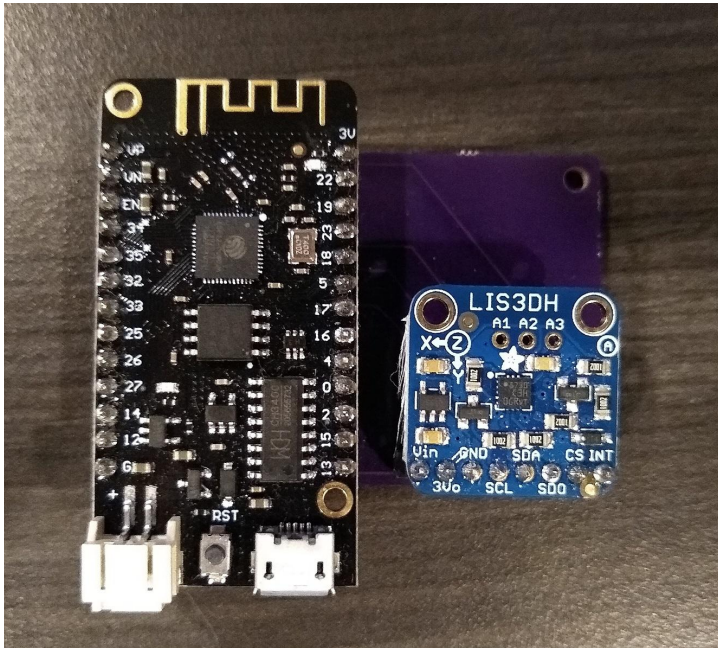
```
  #Set Outputs
  if status == 0 and timer == 0:
        print('Turning relay on')
        relay.on()
  elif status == 1:
        print('status = 1')
  elif status == 2:
        print('status = 2')
  elif status == 3:
        print('Turning relay off')
        relay.off()
        timer = 0
  else:
        print('Something when wrong')

#Timeout
if relay.value():
      timer += 1
      if timer > timeout: relay.off()
```

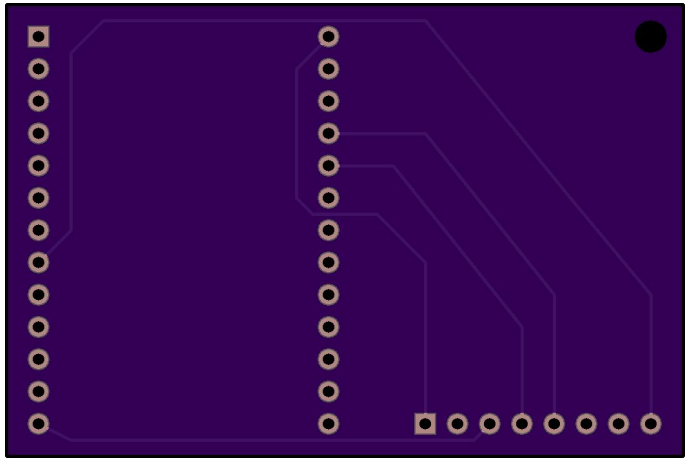
Project: Washing Machine Notifier

https://hackaday.io/project/164356-washing-machine-notifier



Project: Washing Machine Notifier





Project: Washing Machine Notifier

```
1  import utime
2  import lis3dh
3  import machine
4  import network
5  import socket
6
7  sta_if = network.WLAN(network.STA_IF); sta_if.active(True)
8  sta_if.connect("3GWiFi_557A34","[redacted]")
9  #sta_if.connect("NETGEAR61","[redacted]")
10
11 led = machine.Pin(22, machine.Pin.OUT)
12 led.value(1)#off
13 accint = machine.Pin(25, machine.Pin.IN)
14
15 i2c=machine.I2C(scl=machine.Pin(18), sda=machine.Pin(23))
16 accelerometer=lis3dh.LIS3DH_I2C(i2c, address=0x18)
17
18 MY_IFTTT_KEY = '[redacted]'
```

Project: Washing Machine Notifier

```
20  #Simple function that handles sending a get request and printing the response
21  def http_get(site, port, reference, val1, val2, val3):
22      address = socket.getaddrinfo(site, port)[0][-1]
23      print('Connecting to ', site, '(', address, '):', port)
24      s = socket.socket()
25      s.connect(address)
26      message = 'GET ' + reference + '?value1=' + str(val1) + '&amp;amp;amp;amp;amp;amp;value2=' + s
27      print('Sending: ', message)
28      s.send(message)
29      print(s.recv(500))
30      s.close()
31
32  #Simple function to handle sending the IFTTT get request to trigger an event
33  def ifttt_message(event, val1, val2, val3):
34      http_get('maker.ifttt.com', 80, '/trigger/' + event + '/with/key/' + MY_IFTTT_KEY, val1, val2,
35
```

Project: Washing Machine Notifier

```
36  sleep_time = .1 #seconds
37  trigger_time = 10 #seconds
38  count = 0
39
40  while(True):
41      #print(accelerometer.acceleration)
42      if accint.value():
43          led.value(0)#on
44          accelerometer._read_register_byte(0x31) #clear interupt
45          count += 1
46          print(count)
47      else:
48          led.value(1)#off
49          if (count * sleep_time) > trigger_time:
50              #send message
51              ifttt_message('wmnotify', '0', '0', '0')
52          count = 0
53      utime.sleep(sleep_time)
```

Questions

me@ronaldpulliam.com

Try it out:

http://micropython.org/live

http://micropython.org/unicorn