# Mozilla Project Things: SAMW25 Tutorial

Turn a Microchip Xplained Pro SAMW25 into a “Web Thing”

by Kathy Giori
August 15, 2018

Before this tutorial, you should first learn about and set up a Mozilla Things Gateway -- a hub for managing web-of-things ready IoT devices. This particular document focuses on how to set up a programming environment using PlatformIO, then how to program the SAMW25, using example C code (that leverages the Arduino/Wiring framework and libraries). The example will first configure the Wi-Fi module (as a client) and then enable dynamic control of the onboard LED (as an “On/Off” capable Light) using your web browser.

## Links and Prerequisites

Project web site: https://iot.mozilla.org
Github: https://github.com/mozilla-iot
Wiki: https://github.com/mozilla-iot/wiki/wiki
Slides: Overview of Mozilla Things Framework
Online access to a Mozilla Things Gateway test instance (data wiped daily):

  https://w3c-interop.mozilla-iot.org
  u: demo@mozilla.com
  p: …

SAMW25 programming prerequisites:

  Install VS Code, then install the PlatformIO extension to VS Code

Things Gateway user guides:

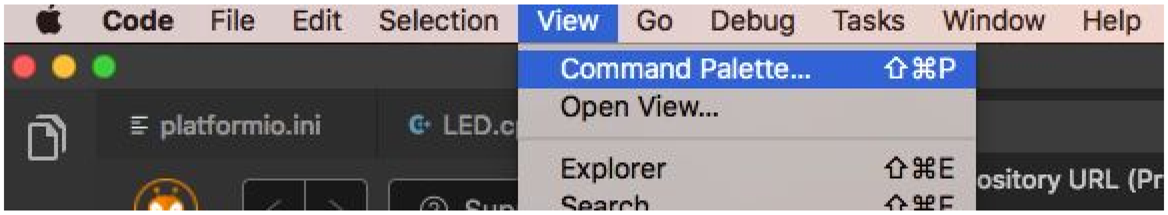
  Mozilla Things Gateway Quick Start Guide
  Mozilla Things Gateway Setup and User Guide

# Open "Webthing-Arduino-LED" Example Using PlatformIO as Programming Tool

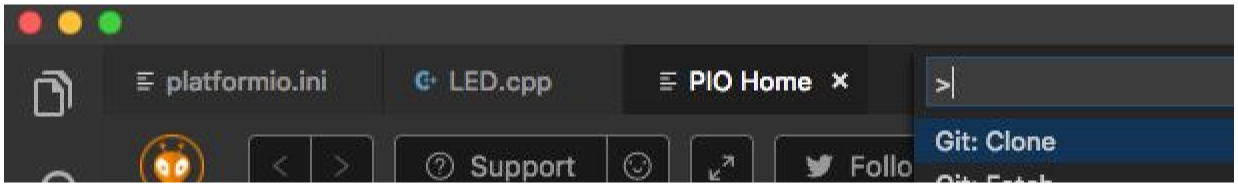
From VS Code, you can either git clone the webthing-arduino-led repository, or you can download it as a .zip file, then open it. Both methods are briefly described below.

## Git Clone Method

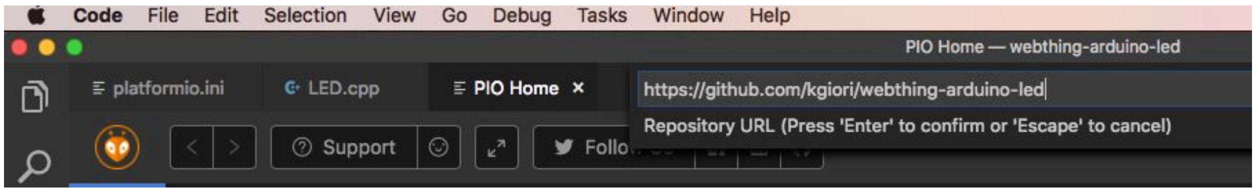
From the VS Code main menu, select **View > Command Palette...**



Start typing "Git" then select **Git: Clone**



Enter the example repository url: **https://github.com/kgiori/webthing-arduino-led.git**

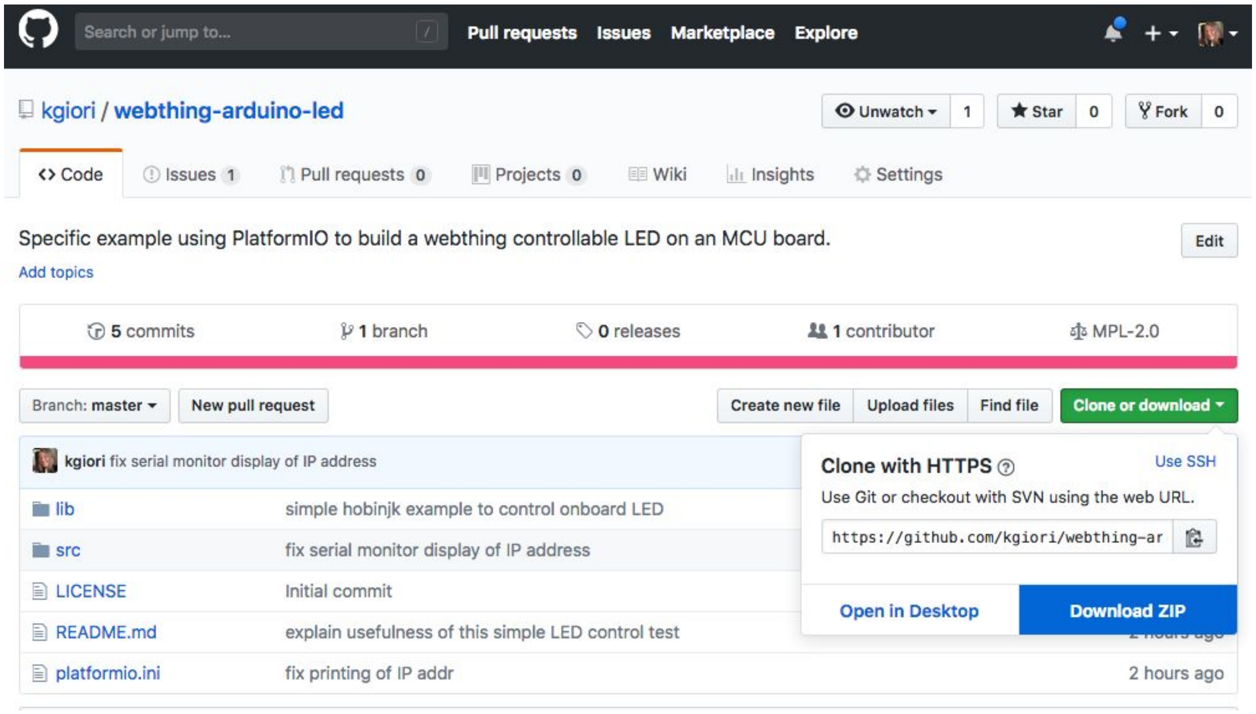


https://github.com/kgiori/webthing-arduino-led

Repository URL (Press 'Enter' to confirm or 'Escape' to cancel)

## Download Zip File Method

Alternatively, you can browse to https://github.com/kgiori/webthing-arduino-led and click on the green button "Clone or download" to fetch a zip file of the example software.
https://github.com/kgiori/webthing-arduino-led.git



kgiori / webthing-arduino-led

Specific example using PlatformIO to build a webthing controllable LED on an MCU board.

5 commits | 1 branch | 0 releases | 1 contributor | MPL-2.0

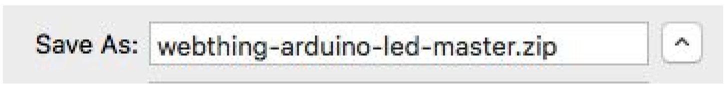
kgiori fix serial monitor display of IP address

| | | |
|---|---|---|
| lib | simple hobinjk example to control onboard LED | |
| src | fix serial monitor display of IP address | |
| LICENSE | Initial commit | |
| README.md | explain usefulness of this simple LED control test | 2 hours ago |
| platformio.ini | fix printing of IP addr | 2 hours ago |

Clone with HTTPS

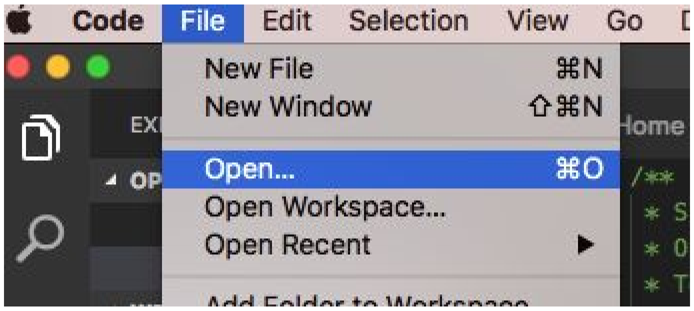
Use Git or checkout with SVN using the web URL.

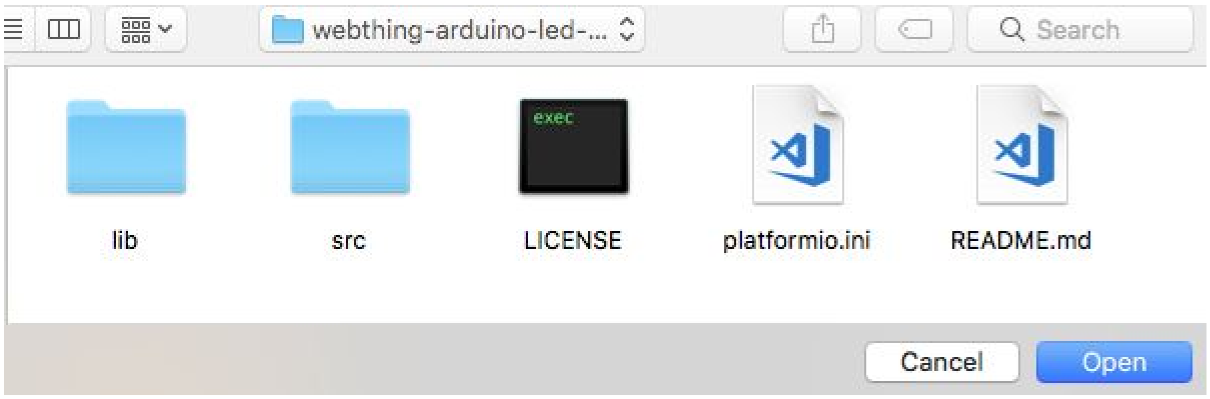
`https://github.com/kgiori/webthing-ar`



Save As: webthing-arduino-led-master.zip

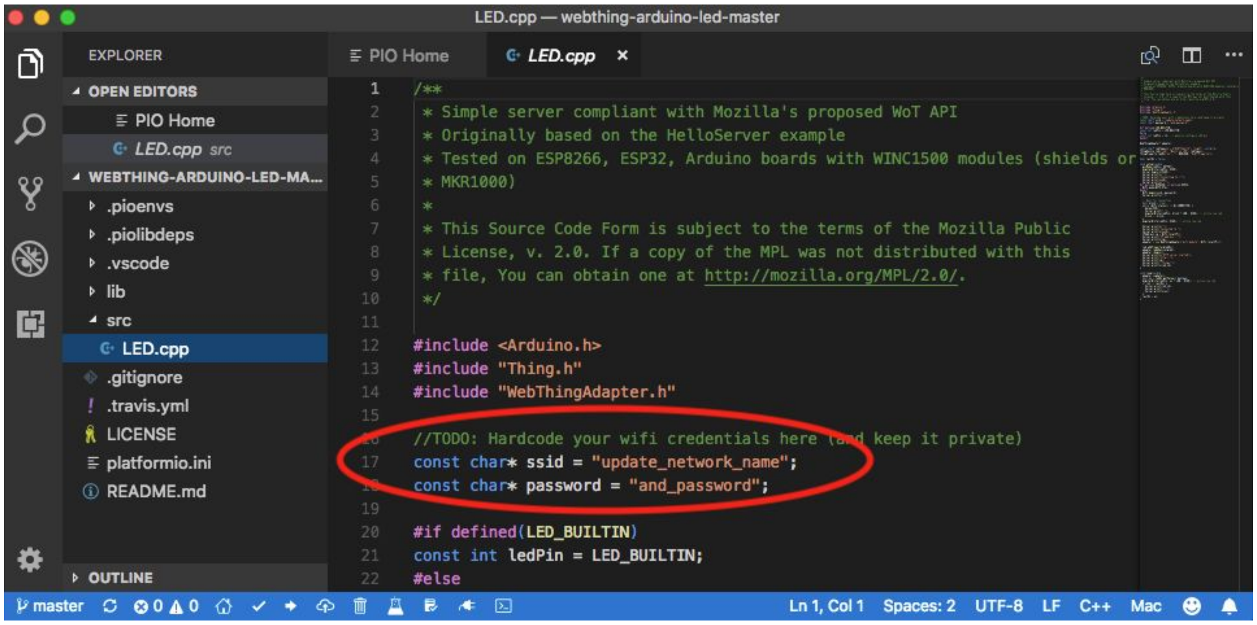
Double click the saved file to unzip it, then open the unzipped folder from VS Code.





## Edit the main program file LED.cpp

Using either method of the previous section, the example project should now be loaded into PlatformIO. In the VS Code editor, the upper left icon toggles visibility of the “File Explorer” -- make sure it is open to show the project files. Click on the arrow next to the “src” folder to see the main file “LED.cpp”. Click the file name “LED.cpp” to see the source code in the pane to the right. Note the “//TODO:...” comment on line 16. The following two lines (17-18) need to be updated. Change the ssid and password to match the local Wi-Fi network that the Mozilla Things Gateway is attached to.



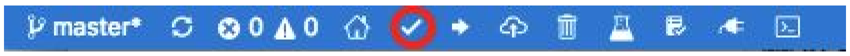
```
1  /**
2   * Simple server compliant with Mozilla's proposed WoT API
3   * Originally based on the HelloServer example
4   * Tested on ESP8266, ESP32, Arduino boards with WINC1500 modules (shields or
5   * MKR1000)
6   *
7   * This Source Code Form is subject to the terms of the Mozilla Public
8   * License, v. 2.0. If a copy of the MPL was not distributed with this
9   * file, You can obtain one at http://mozilla.org/MPL/2.0/.
10  */
11
12 #include <Arduino.h>
13 #include "Thing.h"
14 #include "WebThingAdapter.h"
15
16 //TODO: Hardcode your wifi credentials here (and keep it private)
17 const char* ssid = "update_network_name";
18 const char* password = "and_password";
19
20 #if defined(LED_BUILTIN)
21 const int ledPin = LED_BUILTIN;
22 #else
```

## Build the “Webthing-Arduino-LED” Example

After the Wi-Fi parameters have been modified, save the LED.cpp file. Note: Be careful not commit or submit changes to github that include your private Wi-Fi credentials.

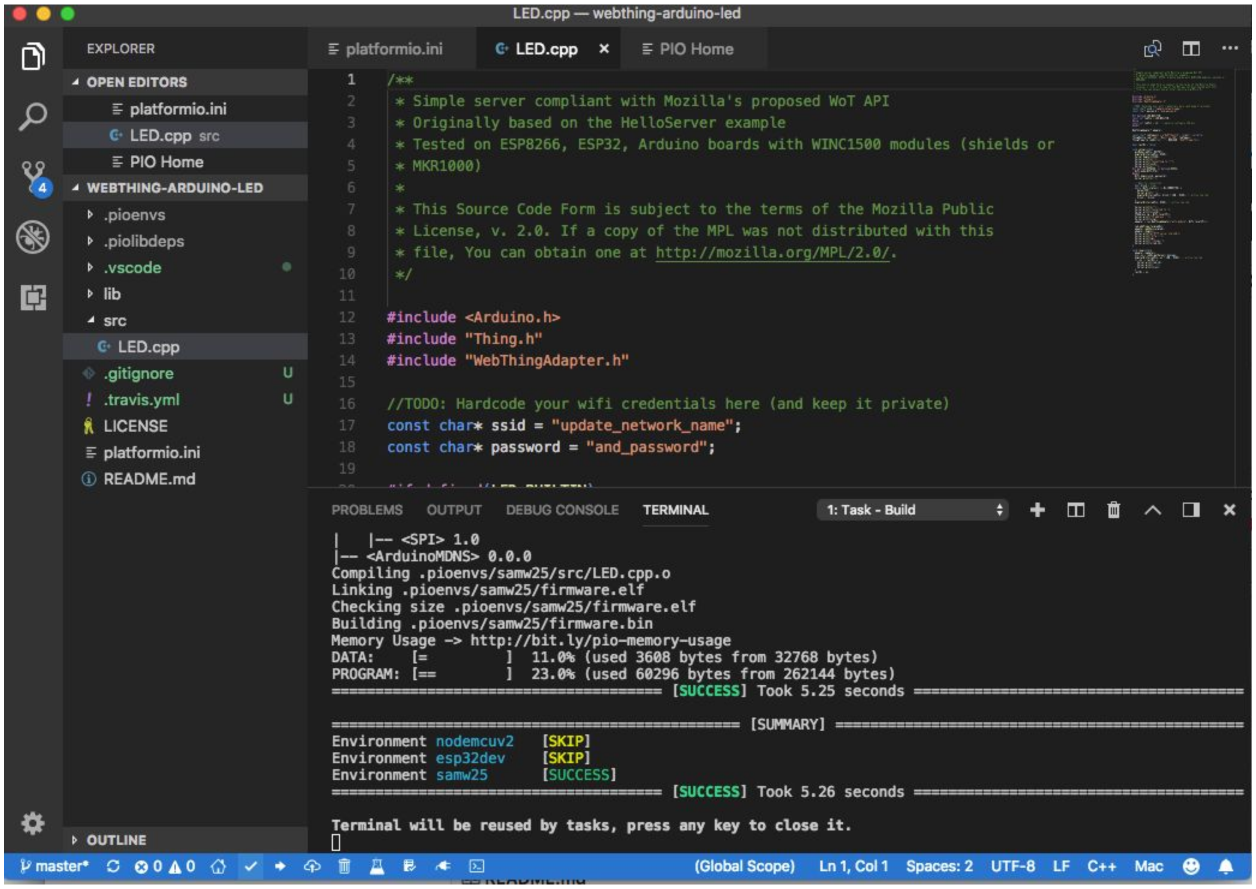
To build the project, click on the check mark of the taskbar (circled in red), or if you have enabled pio commands, you can type the following command in the terminal window.

```
$ pio run
```



The build results should resemble the screen below, showing “SUCCESS” for samw25. If you run into any errors, it is worthwhile to make sure all your software packages are up to date. To do so, at the command line run (or run equivalent in “Command Palette…”):

```
$ pio update
```

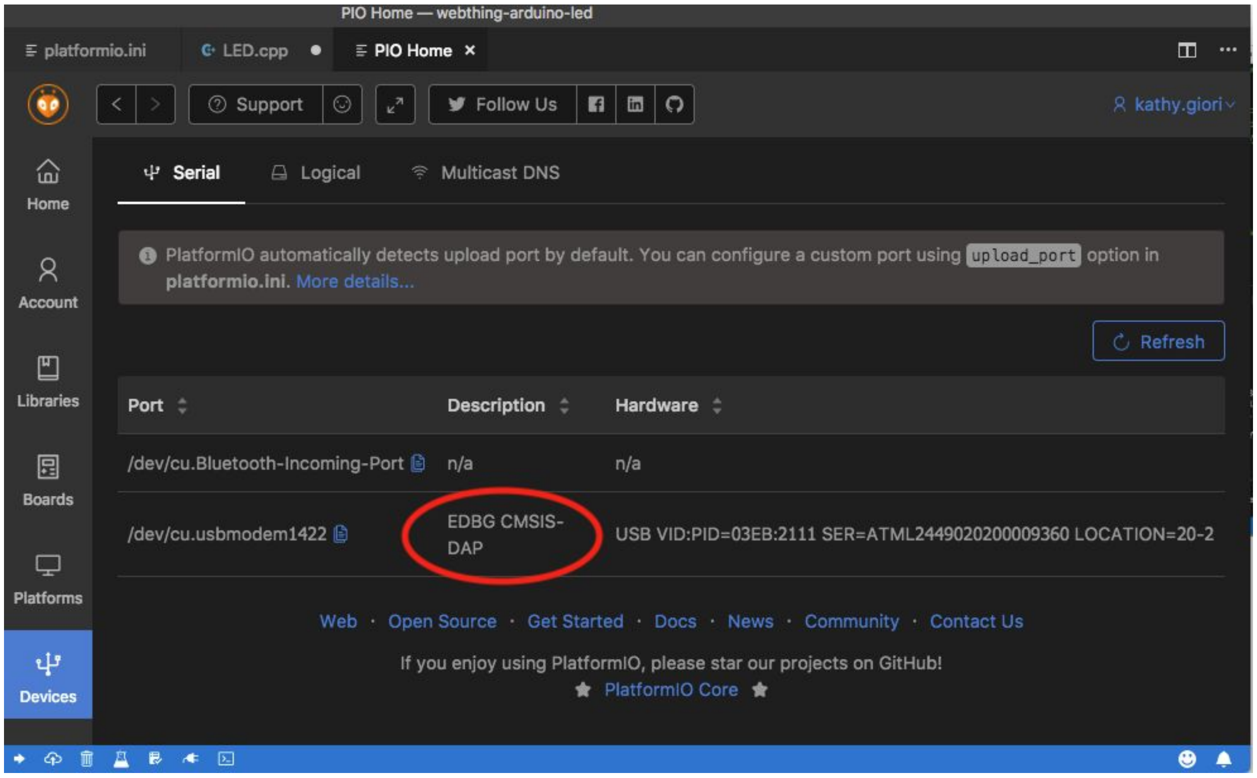


```
/**
 * Simple server compliant with Mozilla's proposed WoT API
 * Originally based on the HelloServer example
 * Tested on ESP8266, ESP32, Arduino boards with WINC1500 modules (shields or
 * MKR1000)
 *
 * This Source Code Form is subject to the terms of the Mozilla Public
 * License, v. 2.0. If a copy of the MPL was not distributed with this
 * file, You can obtain one at http://mozilla.org/MPL/2.0/.
 */

#include <Arduino.h>
#include "Thing.h"
#include "WebThingAdapter.h"

//TODO: Hardcode your wifi credentials here (and keep it private)
const char* ssid = "update_network_name";
const char* password = "and_password";

```

```
|   |-- <SPI> 1.0
|-- <ArduinoMDNS> 0.0.0
Compiling .pioenvs/samw25/src/LED.cpp.o
Linking .pioenvs/samw25/firmware.elf
Checking size .pioenvs/samw25/firmware.elf
Building .pioenvs/samw25/firmware.bin
Memory Usage -> http://bit.ly/pio-memory-usage
DATA:    [=         ]  11.0% (used 3608 bytes from 32768 bytes)
PROGRAM: [==        ]  23.0% (used 60296 bytes from 262144 bytes)
====================================== [SUCCESS] Took 5.25 seconds ======================================

============================================== [SUMMARY] ===============================================
Environment nodemcuv2   [SKIP]
Environment esp32dev    [SKIP]
Environment samw25      [SUCCESS]
====================================== [SUCCESS] Took 5.26 seconds ======================================

Terminal will be reused by tasks, press any key to close it.
```

## Upload Example to SAMW25 Developer Board

First check to make sure that the SAMW25 is recognized by the computer you are using to program it. From the PIO Home page, click on “Devices” along the left navigation column. You should see a device with description similar to that circled in red below.


PIO Home — webthing-arduino-led

PlatformIO automatically detects upload port by default. You can configure a custom port using `upload_port` option in **platformio.ini**. More details...

| Port | Description | Hardware |
|---|---|---|
| /dev/cu.Bluetooth-Incoming-Port | n/a | n/a |
| /dev/cu.usbmodem1422 | EDBG CMSIS-DAP | USB VID:PID=03EB:2111 SER=ATML2449020200009360 LOCATION=20-2 |

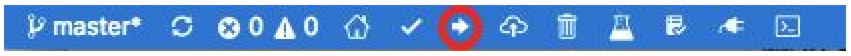
Web · Open Source · Get Started · Docs · News · Community · Contact Us

If you enjoy using PlatformIO, please star our projects on GitHub!

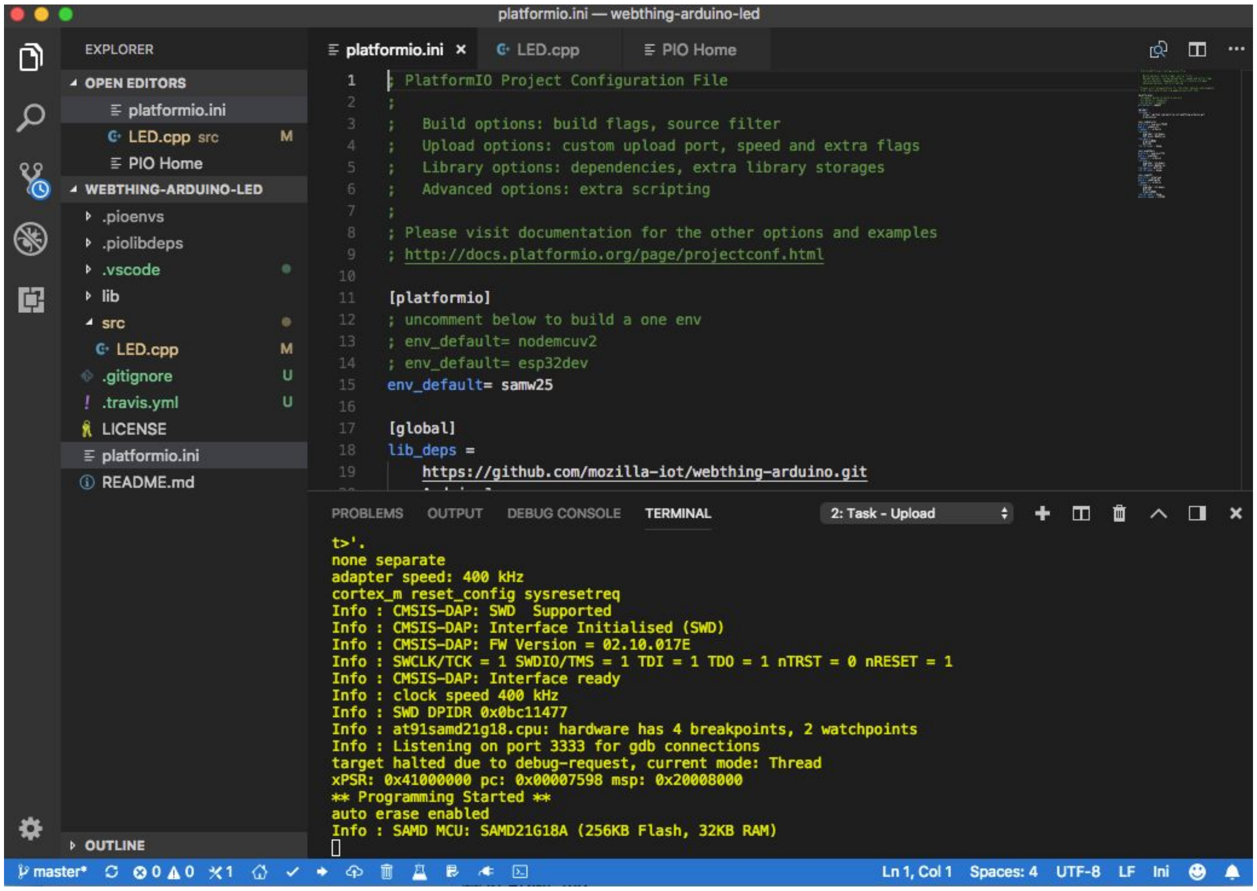
PlatformIO Core

Now you are ready to upload the compiled image to the SAMW25 board. Click the right arrow on the taskbar, or type the following in the terminal.

```
$ pio run -t upload
```



The upload process results should resemble the screen below.

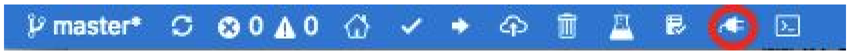


```
; PlatformIO Project Configuration File
;
;   Build options: build flags, source filter
;   Upload options: custom upload port, speed and extra flags
;   Library options: dependencies, extra library storages
;   Advanced options: extra scripting
;
; Please visit documentation for the other options and examples
; http://docs.platformio.org/page/projectconf.html

[platformio]
; uncomment below to build a one env
; env_default= nodemcuv2
; env_default= esp32dev
env_default= samw25

[global]
lib_deps =
    https://github.com/mozilla-iot/webthing-arduino.git
```

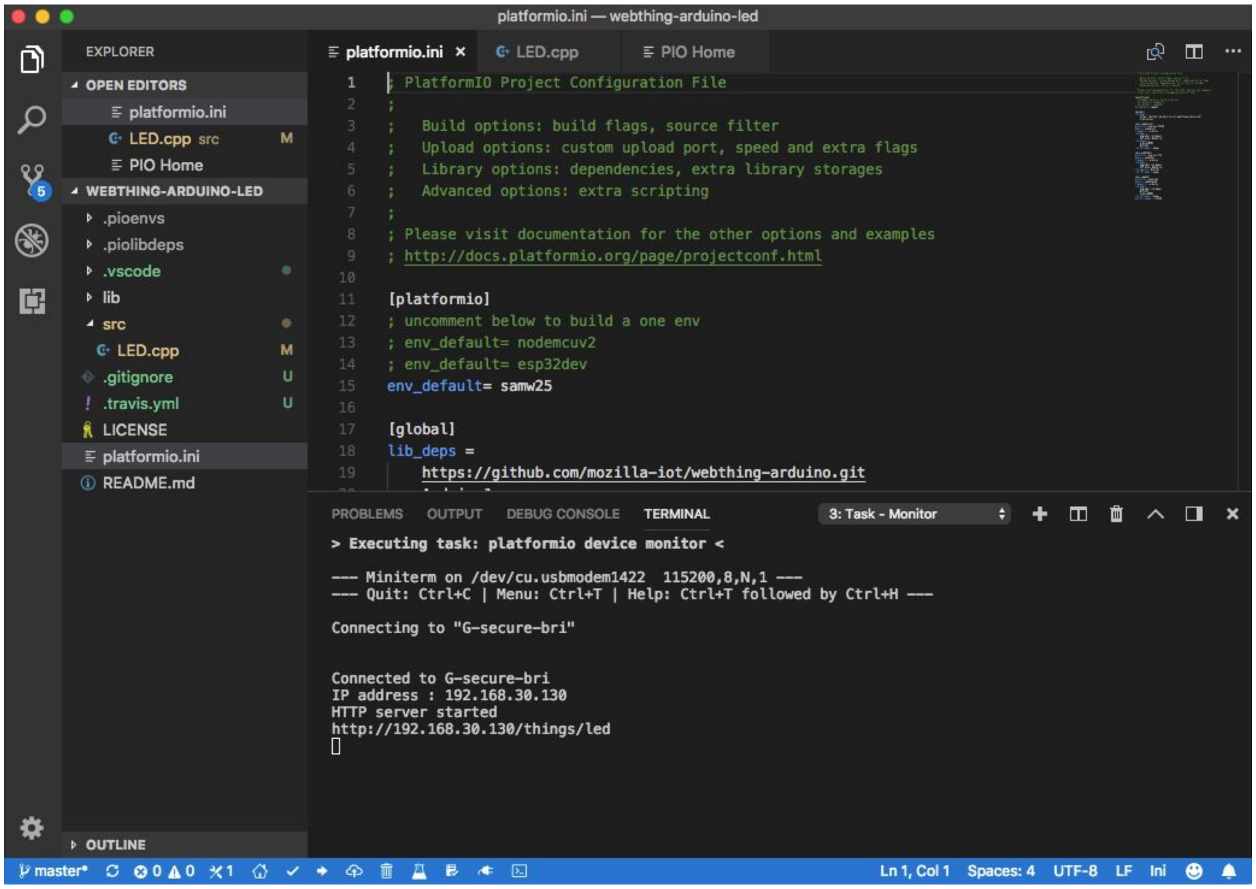
```
t>'.
none separate
adapter speed: 400 kHz
cortex_m reset_config sysresetreq
Info : CMSIS-DAP: SWD  Supported
Info : CMSIS-DAP: Interface Initialised (SWD)
Info : CMSIS-DAP: FW Version = 02.10.017E
Info : SWCLK/TCK = 1 SWDIO/TMS = 1 TDI = 1 TDO = 1 nTRST = 0 nRESET = 1
Info : CMSIS-DAP: Interface ready
Info : clock speed 400 kHz
Info : SWD DPIDR 0x0bc11477
Info : at91samd21g18.cpu: hardware has 4 breakpoints, 2 watchpoints
Info : Listening on port 3333 for gdb connections
target halted due to debug-request, current mode: Thread
xPSR: 0x41000000 pc: 0x00007598 msp: 0x20008000
** Programming Started **
auto erase enabled
Info : SAMD MCU: SAMD21G18A (256KB Flash, 32KB RAM)
```

In order to monitor the status of the uploaded software, click on the serial monitor icon (circled in red) to see the status of the Wi-Fi connection. Or type the corresponding pio command.

```
$ pio device monitor
```



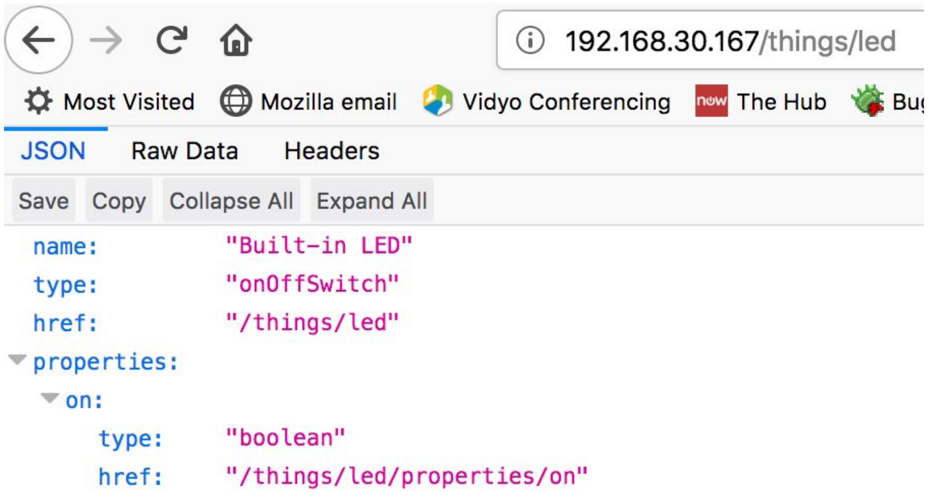
You should see the serial monitor output with results similar to the screen below, except specific to your local WI-Fi network..



```
; PlatformIO Project Configuration File
;
;   Build options: build flags, source filter
;   Upload options: custom upload port, speed and extra flags
;   Library options: dependencies, extra library storages
;   Advanced options: extra scripting
;
; Please visit documentation for the other options and examples
; http://docs.platformio.org/page/projectconf.html

[platformio]
; uncomment below to build a one env
; env_default= nodemcuv2
; env_default= esp32dev
env_default= samw25

[global]
lib_deps =
    https://github.com/mozilla-iot/webthing-arduino.git
```

```
> Executing task: platformio device monitor <

--- Miniterm on /dev/cu.usbmodem1422  115200,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---

Connecting to "G-secure-bri"


Connected to G-secure-bri
IP address : 192.168.30.130
HTTP server started
http://192.168.30.130/things/led
```

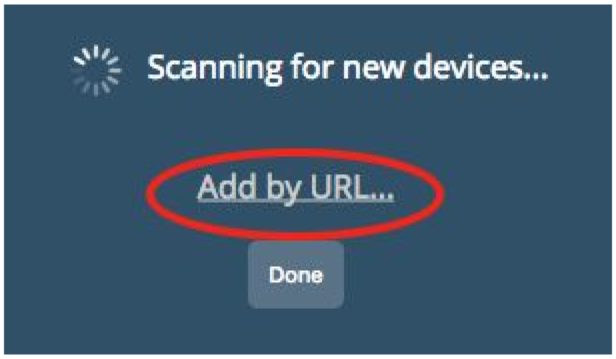
You can check that the SAMW25 is correctly responding by typing its thing url in a browser. You will see the web thing description.

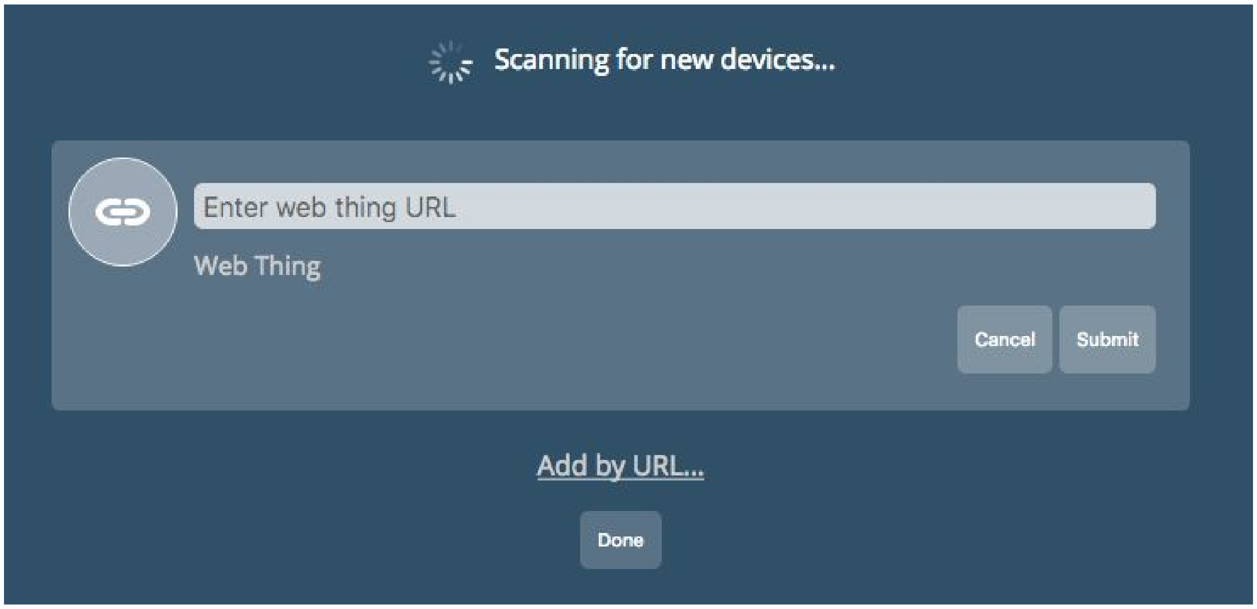


```
name:        "Built-in LED"
type:        "onOffSwitch"
href:        "/things/led"
properties:
  on:
    type:    "boolean"
    href:    "/things/led/properties/on"
```

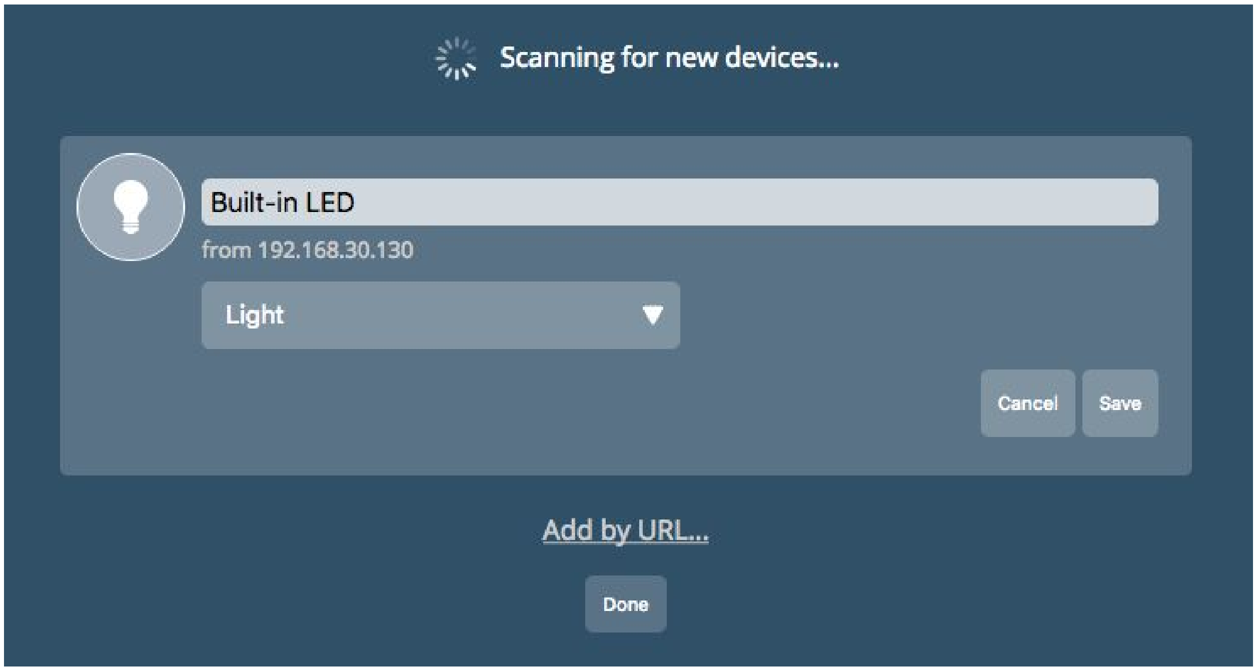
## Discover and Add SAMW25 as a "Web Thing" to the Things Gateway

From the "Things" page of your Mozilla Things Gateway, click the "+" icon in the lower right corner to scan for web things. Click the link "Add by URL…"

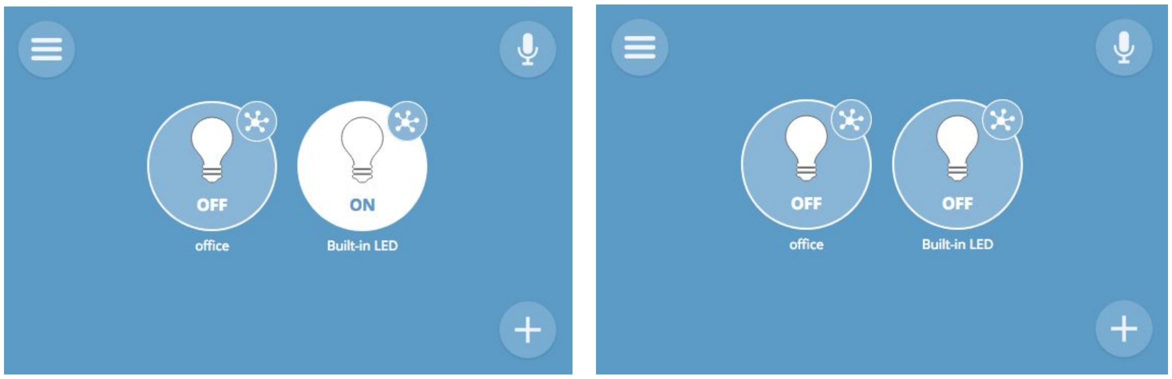




Enter the URL (such as `http://192.168.30.130/things/led`) as shown in the PlatformIO Serial Monitor pane, and click "Submit". If the SAMW25 is accessible, the gateway will load the thing description as shown below.



Change the name if you would like, then click “Save” and “Done”. The onboard LED of the SAMW25 should now be controllable via the Things page of the gateway. Click the icon to turn the LED on and off.



Watch the LED control in action in the embedded video clip.
https://drive.google.com/open?id=1cJPZx10R5DtV7aS0wXwDiF_Yj5-1FVu8

## Example PIO Commands (Terminal Commands)

HELP: `pio -h`
BUILD: `pio run`
(for specific board “environment”, use “`pio run -e samw25`”)
UPLOAD: `pio run -t upload`
(or for only samw25, “`pio run -e samw25 -t upload`”)
SERIAL MONITOR: `pio device monitor`