Gateway to the IoT



Creating a Connected Thing Using a SAM W25 Xplained Pro Evaluation Kit and Mozilla Things Gateway

Contributed by Jorge Amodio

Among the many challenges of implementing an Internet of Things (IoT) application in an optimal and cost-effective way, we can list connectivity, power and security, as well as data aggregation. I've been doing R&D on IoT applications for many years, even before it had a name. At the time, we used to talk about connected smart objects. Now we just refer to them as things. These objects or devices could be a simple sensor, a light or a more complex device. But the common feature is that they are all connected to the global Internet, feeding data to various cloud applications or taking control commands from anywhere on the net and performing some action.

Given the advances in semiconductor technologies to enable more complex and powerful microcontrollers (MCUs) and application-specific integrated circuits, with a substantial reduction in size and power requirements, it is possible today to implement many things at an affordable and scalable cost. But there are still some tradeoffs when putting together a complete IoT application.

One question that often arises is, "Does it make sense for every single thing to connect directly to the Internet?" The simple and short answer is, "No." There are applications where there is no need to have every device connected directly to the open Internet; hence the device can have a much simpler design, reducing cost, power requirements and firmware complexity. But we still want it to be an Internet thing. This introduces the concept of a gateway to the architecture of the IoT.

A gateway is a device of medium complexity that is connected to the public Internet on one side and includes various interfaces and/or adapters for connecting to local things. In this scenario, data is sent and received from the things through the gateway. This approach significantly eases the requirements for implementing things for a more robust, secure and cost-effective application.

For example, we can run a decently featured firewall and secure tunneling protocol from the gateway to various cloud applications, aggregate and normalize data generated by many sensors on our things and have more control of what information is being sent to the open Internet. This reduces what functionality needs to be implemented on the thing.

I recently had the opportunity to attend one of the workshops Microchip's **Get Launched** event in Austin, Texas, where I learned how to connect the **Mozilla Things Gateway** and a simple thing via Wi-Fi® using Microchip's **SAM W25** module. A detailed tutorial will require more space than what we have available for this article, but I'll share the most important pointers about how to put together this project.

(continued on page 28)

Let's start with the Mozilla Things Gateway. This is a prototype of an open-source IoT gateway implemented in JavaScript with a Node.js web server running on a Raspberry Pi® 3. By taking advantage of the Raspberry Pi's on-board Wi-Fi interface, you can connect things locally to the gateway, and then connect the gateway to the open Internet either via Wi-Fi or the on-board wired Ethernet interface. For this project I am using the Ethernet option. The Raspberry Pi is connected to one of my internal Wi-Fi networks along with the things, and it is connected to the Internet via a standard router on my lab's LAN.

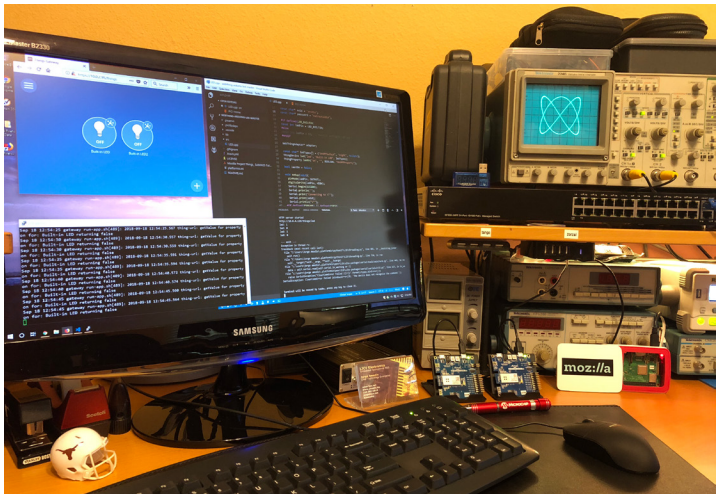

Figure 1: Mozilla Things Gateway Project Setup in the Lab

While the Mozilla Gateway can be built using the source code that is available on Mozilla's repository, the easiest and fastest way to get started is to just download the pre-built Raspbian **microSD™ card image**, unzip the file and write it onto a fresh microSD card with at least 4 GB of space. I recommend that you use a Class 10 high-speed microSD card. On a PC running Windows®, you can use Etcher, Disk Image Writer or a similar tool to write the image on the card. After this is done, insert the microSD card into the Raspberry Pi and power the board on.

The very first time the Raspberry Pi boots, it will automatically start the Mozilla Gateway. For the initial setup, you can use the wired Ethernet interface to connect to http://gateway.local/, or you can join the ad-hoc network via Wi-Fi using "Mozilla IoT Gateway" for the SSID. The first step is to configure the Wi-Fi network to which you will connect, and that will be shared with the local things.

The next step is to create the Mozilla IoT account and subdomain for secure access to the gateway. The gateway user interface will then show a page where you can start adding your things. Next, it will be time to make a thing.

As I mentioned earlier, during the Microchip workshop we learned how to make a thing using the SAM W25 module. To do this, we were provided with a **SAM W25 Xplained Pro Evaluation Kit**. The board includes the SAM W25 module, a pair of user LEDs, a user pushbutton, a RESET button and a pair of micro USB connectors. One of these connectors is used to program and debug the on-board MCU running the Atmel Embedded Debugger (EDBG) that will let us connect to the SAM W25 module to program the thing's firmware.

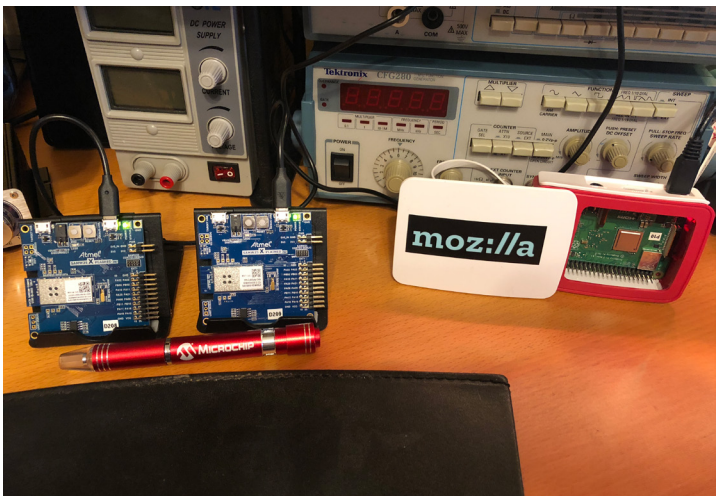

Figure 2: SAM W25 Xplained Pro Boards and Raspberry Pi® 3 Gateway

Talking about firmware, this another area where Mozilla has contributed with open-source tools and code. First, you will need to download and install **Microsoft® Visual Studio Code**. After you have VSCode running, install the **PlatformIO IDE Extension**. VSCode will download and install the IDE and all required dependencies. You will also need another tool: **GIT**, the popular, open-source source-code management system.

Now you need the source code for the thing. The Mozilla IoT Project put together a very simple example to show how this works. You can clone the repository with GIT or just simply download a zip file. Unzip the files onto a local directory, then run VSCode, launch PlatformIO, and load the "webthing-arduino-led" project. Since this is just a simple example, you will have to hard code the SSID and password for your local Wi-Fi network into the existing LED.cpp file. Enter the correct SSID and password, save the file and you will be ready to build the image.

Using the micro USB connector labeled "DEBUG USB," connect the SAM W25 Xplained Pro board to your PC. Windows will recognize the USB device, and it will be listed on PlatformIO as one of the available devices on a virtual serial port. In my project, it appeared as COM5.

To build the image and program the SAM W25 module in PlatformIO IDE, go down to the toolbar in the bottom left corner

(continued on page 29)

and first click on the “Build” icon (the check mark) and then on the Upload icon (an arrow pointing right), indicated by the green and red arrows in Figure 3 below:

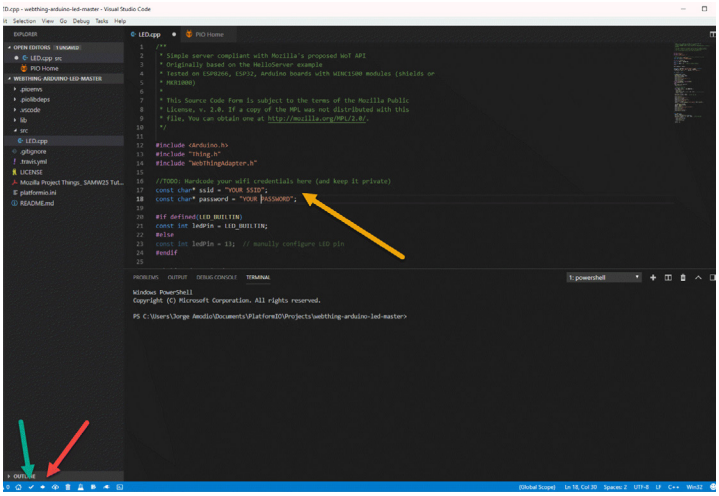

Figure 3 – PlatformIO Code Changes and Programming for SAM W25 Module

If everything works as expected, the SAM W25 module will connect to the local Wi-Fi network and will be ready to receive commands to turn on or off the on-board LED0. You can use the Serial Port Monitor (the icon that looks like a power plug) to check the status of the board. When more than one device is present, it will ask you which one to connect.

After the board is programmed and the SAM W25 module comes out of reset, you should see something like:

Connected to YOURSSID
IP address: 10.0.4.136
HTTP server started
http://10.0.4.136/things/led

The last line will be the URL defined for your new thing. Copy this URL for the next step.

Adding your new thing to the gateway is very easy. Go to the web page served by the gateway and click on the + sign in the lower right corner. The gateway will start scanning for devices and will also present the “Add by URL” option. Click on this option and paste or type in the URL for the thing that you created earlier. The thing will be added to the gateway. When you click on it, the status of the LED on the SAM W25 Xplained Pro board will change.

When you add the thing, in the background the gateway will connect to it and retrieve the JSON web thing description. This is the schema Mozilla is proposing as a standard, developed from the knowledge and experiences gathered in the evolution of the World Wide Web.

```
[
  - {
      name: "Built-in LED",
      href: "/things/led",
      @context: "https://iot.mozilla.org/schemas",
    - @type: [
        "OnOffSwitch",
        "Light"
      ],
    - properties: {
      - on: {
          type: "boolean",
          @type: "OnOffProperty",
          href: "/things/led/properties/on"
        }
      }
  }
]
```

Figure 4 – JSON Description of Connected Thing

Just a few words about the Mozilla IoT project. While there is a lot of buzz and activity on the IoT front, we are just scratching the surface of the potential applications. More than 50 billion connected things are projected to be in use in the very near future. Today there is no standard framework that will guarantee interoperability between different implementations that are still proprietary, without a common data model and highly centralized. The primary mission at Mozilla is to, “Ensure the Internet is a global public resource, open and accessible for all.” The IoT is an area of great importance, and Mozilla is making a major contribution in the development of open standards. In May 2017, Mozilla Corporation submitted a draft Web Thing API to the World Wide Web Consortium (W3C) and is an active participant in the Web of Things Interest Group.

As you can see from this very simple example, it is not that hard to get started on developing applications for what, in my opinion, will become the next technological revolution of our time. If you have any questions or feedback about this project, feel free to contact me at **jorge@ljcv.net**.