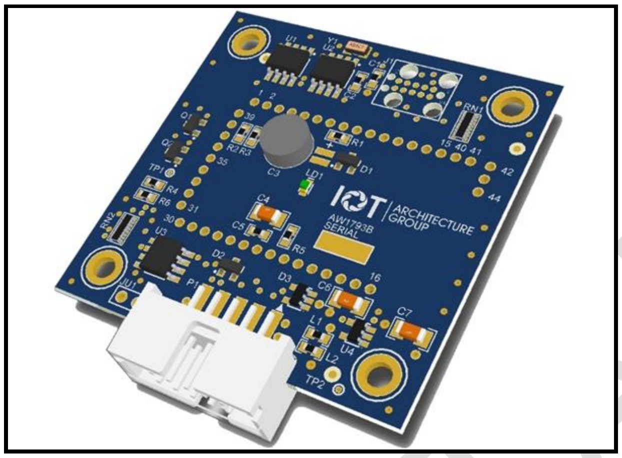
Over-The-Air Updates Guide for the Wisplet® S2W IOT Engine

WWW.IOTARCHITECTUREGROUP.COM
SILICON ENGINES
3550 W. SALT CREEK LANE, SUITE 105
ARLINGTON HEIGHTS, IL 60005 USA
847-637-1180

Contents

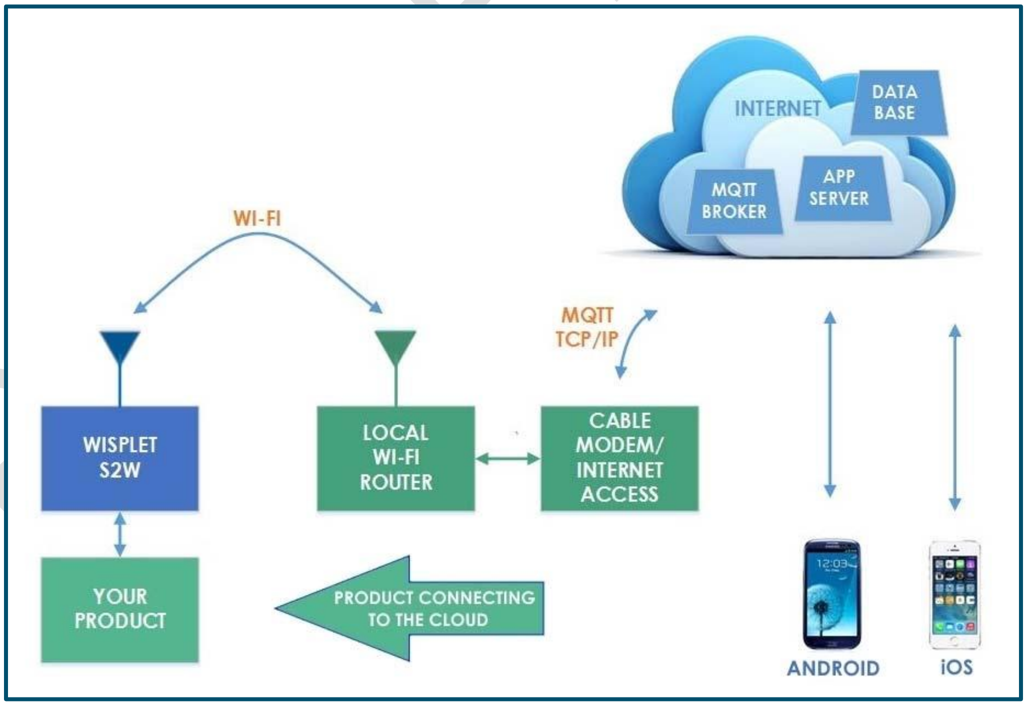
1. INTRODUCTION

The Wisplet S2W is a small, low-cost, internet connectivity hardware board specifically designed for Internet-of-Things applications.



WISPLET S2W HARDWARE MODULE
FIGURE 1.1

The Wisplet S2W provides internet connectivity on one side, and connectivity to a hardware product on the other side, for the purpose of providing a way for that hardware product to connect to a cloud application.



WISPLET S2W CONNECTS YOUR PRODUCT TO THE CLOUD
FIGURE 1.2

In addition to including the necessary radio circuitry for providing internet connectivity, the Wisplet provides an implementation of the MQTT protocol, which is fast becoming the industry standard for Internet-of-Things (IOT) applications.

The Wisplet also performs rules processing, so that it can be configured to understand the various sensors built into the hardware product to which it is attached. The Wisplet can be configured to know which sensor value to report to the cloud at a defined interval in status reports, and can also know which ranges of values for each sensor that should be considered to be a condition that should generate an immediate alert to the cloud.

In addition to being able to report data from the hardware product to the cloud, the Wisplet can also deliver control messages from the cloud to the hardware product. This could be something as simple as telling the hardware product to display a particular value on an LED display on the product, or something more complicated like telling the hardware product to unlock or open a door.

2. MQTT INTERFACE

There are three interfaces to the Wisplet S2W:

- its UART which is how it talks to your product,
- its configuration web pages, used to select your Wi-Fi access point, and
- via MQTT commands.

This document is directed towards the MQTT interface. There are 2 components to a MQTT message:

- the topic string (aka the address to send the payload), and
- the actual message data (payload).

2.1 Topic String Definition

Topic strings are case sensitive, hierarchal in their definition and different levels are separated via a fore slash (/) character. The convention adopted by the Wisplet is to use all lowercase characters in the topic string to avoid any issues of mixed case.

The convention used by the Wisplet requires that the first level of the topic string is always the client ID, the unique identifier that is used by the MQTT client when connecting to the MQTT broker, who is publishing the message. For the Wisplet, this is its MAC address, expressed via 12 hexadecimal characters (0-9 a-f) all lower case. For the cloud or other non-Wisplet device this could be anything. At the time this document was written, this portion of the topic string is ignored by the Wisplet for any messages it receives. For future compatibility, it may be useful to carefully define a unique name to be used here in case the Wisplet subscriptions are tightened up to only permit messages from a specific source. Whatever is used, be careful to keep that field 12 characters or less, no spaces or other special characters, recommend using just A-Z a-z 0-9 – and _ for now.

The second level of the topic depends on the direction of the message.

- For messages being sent by a Wisplet to the supervisor of the system, aka the north side, it is simply the string ***msg*** to keep it short and meaningful.
- For those messages being sent to a Wisplet, it is the MAC address of that Wisplet device that you want to receive that message. This address is in the same format as the first level that of the topic string.

There is a special variation of this second level MAC address that uses wildcards to permit targeting of multiple Wisplets via a single message. The wildcard character is defined as a lower case letter **x**, and it can be used to substitute for a specific hexadecimal character in the second level address. Wildcards are only valid when used in a specific order, left to right, starting with the most significant character of the MAC address.

For example, let's assume that we want to send a message to a Wisplet with MAC address abcdef123456 and our MAC address is 1a2b3c4d5e6f, then the topic string will begin like: 1a2b3c4d5e6f/abcdef123456 with the remainder of the topic string to be defined later. Sending a message to a topic string with this definition will only reach one and only one Wisplet, the Wisplet with MAC address abcdef123456. With one wildcard, xbcdef123456, we can make the same message affect up to 16 Wisplets, since the first hexadecimal character can be anything 0-9 a-f. If we use two wildcards, the address becomes xxcdef123456 and that address can affect up to 256 Wisplets. There are 13 possible addresses possible using this wild card scheme, ranging from a specific address with no wildcards at all, up to 12 wildcards which means all Wisplets. Using our above example we can create the following addresses:

`abcdef123456` – up to 16^0 = 1 device

`xbcdef123456` – up to 16^1 = 16 devices

`xxcdef123456` – up to 16^2 = 256 devices

`xxxdef123456` – up to 16^3 = 4,096 devices

`xxxxef123456` – up to 16^4 = 65,536 devices

`xxxxxf123456` – up to 16^5 = 1,048,576 devices

`xxxxxx123456` – up to 16^6 = 16,777,216 devices

`xxxxxxx23456` – up to 16^7 = 268,435,456 devices

`xxxxxxxx3456` – up to 16^8 = 4,294,967,296 devices

`xxxxxxxxx456` – up to 16^9 = 68,719,476,736 devices

`xxxxxxxxxx56` – up to 16^{10} = 1,099,511,627,776 devices

`xxxxxxxxxxx6` – up to 16^{11} = 17,592,186,044,416 devices

`xxxxxxxxxxxx` – all devices, up 16^{12} = 281,474,976,710,656 devices

Care must be taken when using this type of addressing because you could flood the network with a lot of traffic from Wisplets replying back. But this addressing scheme can be very useful for such tasks as firmware updates where you could stage the update process. For example by using 11 wildcards, that leaves the last character as defined, and with 16 possible choices for that character (0-9 a-f) you could address the entire device pool of Wisplets with 16 separate topic string definitions, giving you a way to stage your firmware updates and lessen the burden on the server providing the update. If you choose 2 wildcards, then you stage over 256 topic strings and messages. The less wildcards used, the more stages you can define and roll out a firmware update or query a group of Wisplets.

To take advantage of this wild card scheme, the Wisplet will subscribe to 13 topic strings, with each topic being a variation of its MAC address as documented above. In this manner, the Wisplet does not need to process every message received to see if it is for it, something which could be time consuming in large eco system. Instead, the Wisplet will only get a message if that message was addressed to it via an exact address or some version of the wild cards.

2.2 Message Payload Definition

The message payload varies with each message/reply. See the following message definitions for more details.

Legend

<from> = MACaddressOfSendingWisplet OR the device sending to a Wisplet

<to> = targetWispletMACaddress or wildcard address for that Wisplet

Wisplet firmware version: A.B.C.D where

A = UART interface to customer protocol rev

B = MQTT client version

C = OS Version

D = build number

While A,B and C may jump to reflect changes in the supporting libraries, you should use D (build number) when checking for new features, etc.

3. Wisplet Firmware Update

- Requires Wisplet firmware version = 2.144.352.21 or later (DCT formatting)

- Direction = to Wisplet (command)
- Topic string format = <from>/<to>/**updates/newwispfirmware**
- Payload format
  - CRC32 : <crc 32-bit checksum value for the file>
  - SHA256 : <special SHA-256 code to protect file, prevent bad firmware>
  - length : <# bytes of the file being sent>
  - port : <port # for server hosting the file>
  - server : <name or IP address of the server>
  - link : <link to file on that server>
  - secure : <0=no (clear), other positive value = yes (TLS)>
    - auth : <0=no username, password required, other positive value = yes> username : <if auth is yes, then this string must be defined, no spaces>
    - password : <if auth is yes, then this string must be defined, no spaces>

- Direction = from Wisplet (reply)
- Topic String format = <from>/msg/**updates/ackwispfirmware**
- There should be 3 replies to this command
  - 1st reply = if command was accepted or rejected, should be immediate
    - requeststatus : "true" OR
    - requeststatus : "false", "info" : <error msg>

  - 2nd reply = result for file download and store locally
    - downloadstatus : "true" OR
    - downloadstatus : "false", "info" : <error msg>

  - 3rd reply = result of bootloading the Wisplet board
    - bootloadstatus : "true" OR
    - bootloadstatus : "false", "info" : <error msg>, Val1 : <value1>, Val2 : <value2>, Val3 : <value3>, "WispletVersion" : <current fw version>

- 1st reply is result of error checking request, to make sure all required parameters exist, so it should return right away

- 2nd reply will happen if result of 1st was true. This will happen after the file has been downloaded, validated and stored locally. The Wisplet will go offline to complete this request and then come back online to communicate the results.

3rd reply will happen if 1st and 2nd are true. The timing for this reply depends on the size of the new firmware file, but it should be less than 5 minutes to consume the file and finish its bootload.

Below is an example of a message for a Wisplet firmware update. The CRC32, SHA256 and length should come from developer of firmware and should not be changed. The other settings can be adjusted as needed to fit the environment serving the file. Notice below that the server is using a non-standard port and the file is in a subfolder on the server.

```
{
"CRC32":"731643670",
"SHA256":"3CC3D1BB0F64D7A1A6B9B1733B23CEF9667A180E4556830072F40D69CB655796",
"auth":"1",
"length":"684740",
"link":"/wispfw/wisplet_v2.144.352.21_2016_Mar_31.bin",
"password":"mysecret",
"port":"4443",
"secure":"1",
"server":"www.myserver.com",
"username":"wispletupdate"
}
```

4. REVISION HISTORY

4.1 Rev. A

First release – 2016 June 3 – dmr SE