Application Note

PubNub Thermostat example using SAM W25 Xplained Pro

AN-XXXXX

Prerequisites

- **Hardware Prerequisites**
  - Atmel SAM W25 Xplained Pro Evaluation Kit
  - Atmel IO1 extension
  - USB Micro Cable (TypeA / MicroB)

- **Software Prerequisites**
  - Atmel Studio 6.2

Introduction

This application note describes how to use a state-of-the-art Internet of Things (IoT) application with the SAM W25 in the PubNub cloud.

The following topics will be covered:

- How to build and execute the example.
- How to set SAM W25 and PubNub console to communicate each other.

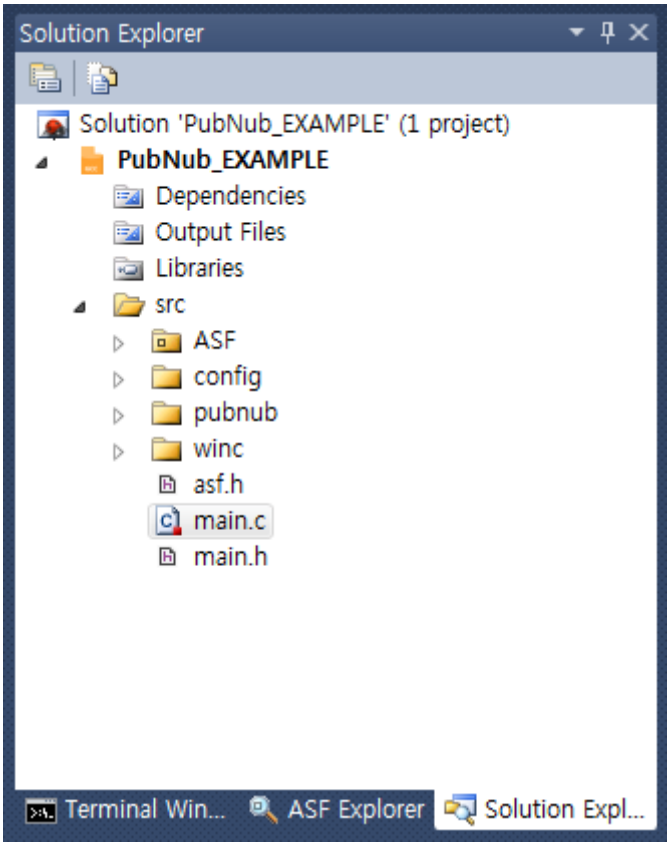
Table of Contents

Icon Key Identifiers

INFO	Delivers Contextual Information About a Specific Topic
TIP	Highlights Useful Tips and Techniques
TO DO	Highlights Objectives to be Completed
RESULT	Highlights the Expected Result of an Assignment Step
WARNING	Indicates Important Information
EXECUTE	Highlights Actions to be Executed Out of the Target

1. Example Source Tree

The example can be shown as below. The main topic sources are ‘main.h’ file and ‘main.c’ file.

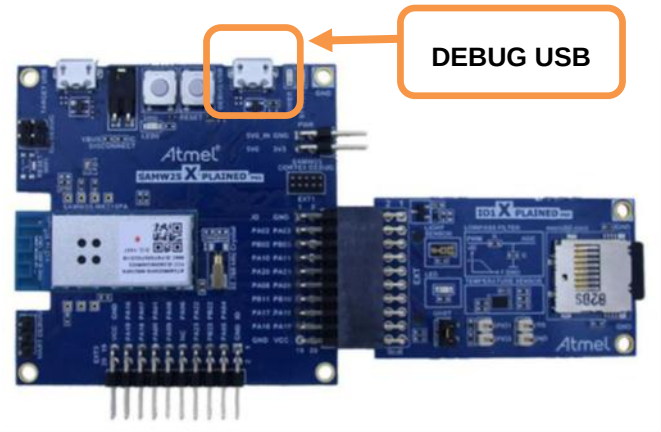


2. How to Build & Execute

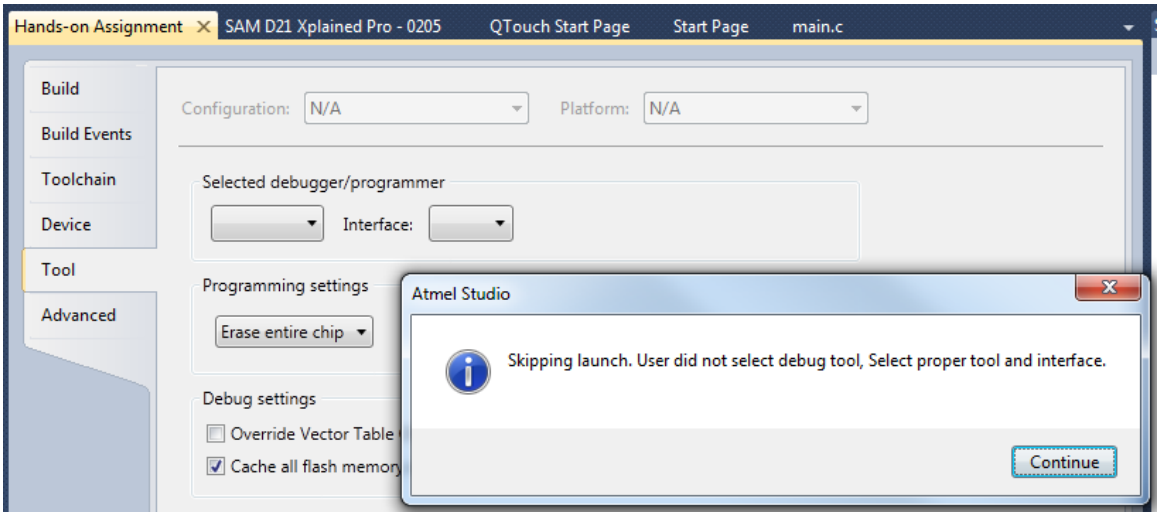
EXECUTE Build the solution (F7) and ensure you get no error: 

EXECUTE Program the SAM W25 Xplained Pro.

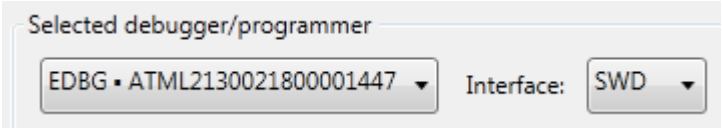
- Connect the IO1 extension to the SAM Xplained Pro as displayed below:



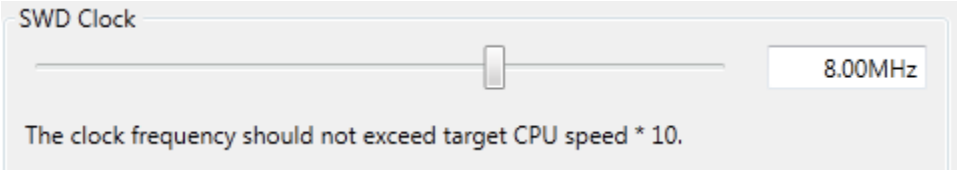
- Connect the SAM W25 Xplained Pro board to your PC by using DEBUG USB connector.
- Program the application by clicking on the Start Debugging and Break icon: 
- You will be asked to select your debug tool as below:



- Select EDBG and SWD (Serial Wire Debug) as Interface:

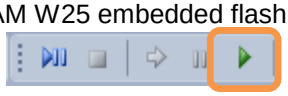


- Set SWD clock to 8 MHz to speed up programming:

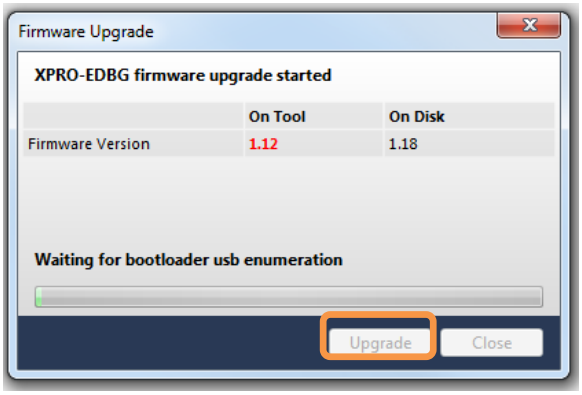


- Click again on the Start Debugging and Break icon: 

- The application will be programmed in the SAM W25 embedded flash and breaks at main function.

  Click on Continue to execute the application: 

INFO You may be asked to upgrade your EDBG firmware. If so, click on Upgrade:



WARNING Upgrade operation may take a few minutes, please **wait** for the operation to complete.

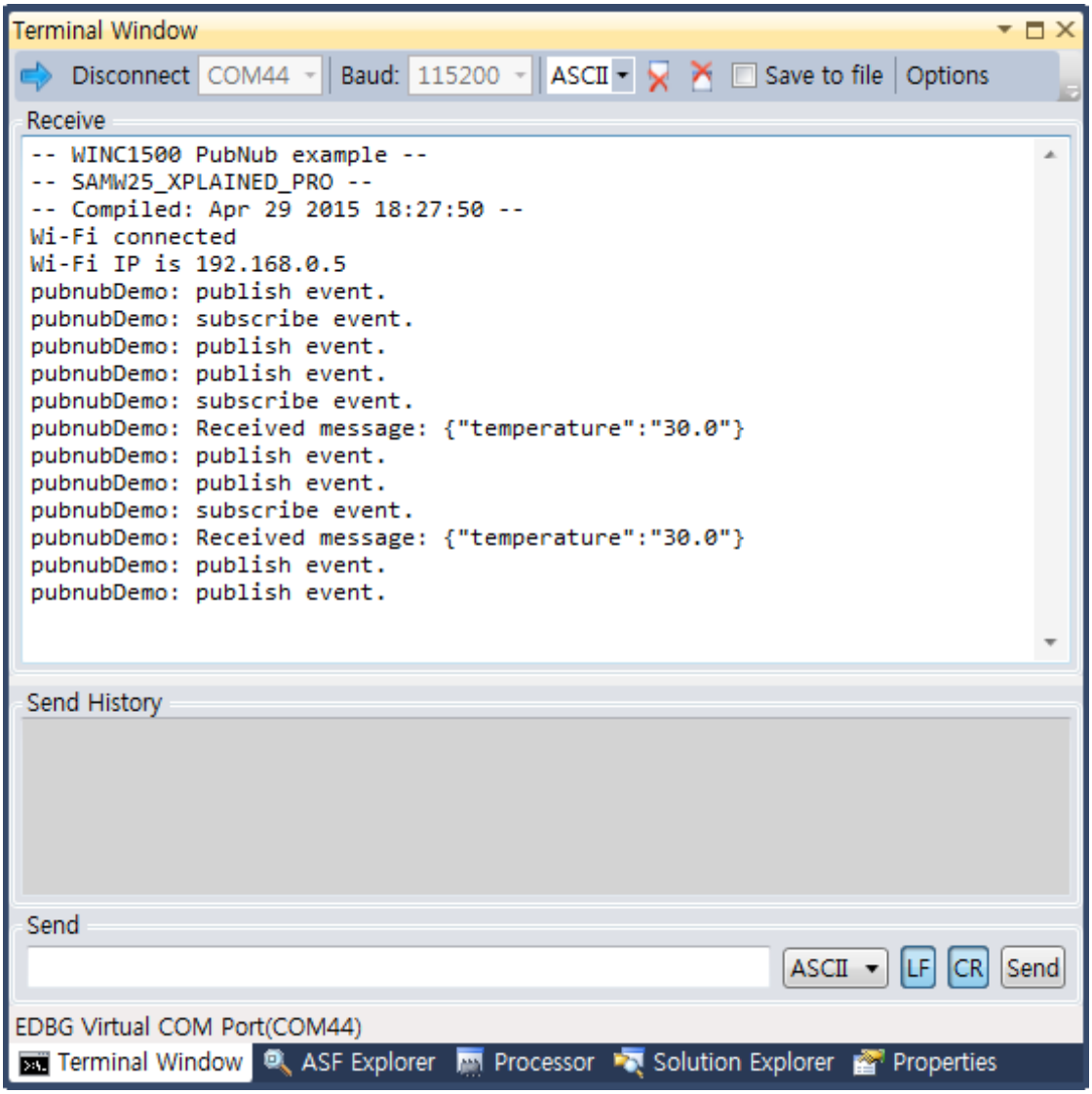
RESULT The IOT application is now programmed and running.

EXECUTE Open the serial terminal with the following settings:

- 115200 bauds
- 8 bit data
- no parity
- one stop bit
- no flow control

TIPS

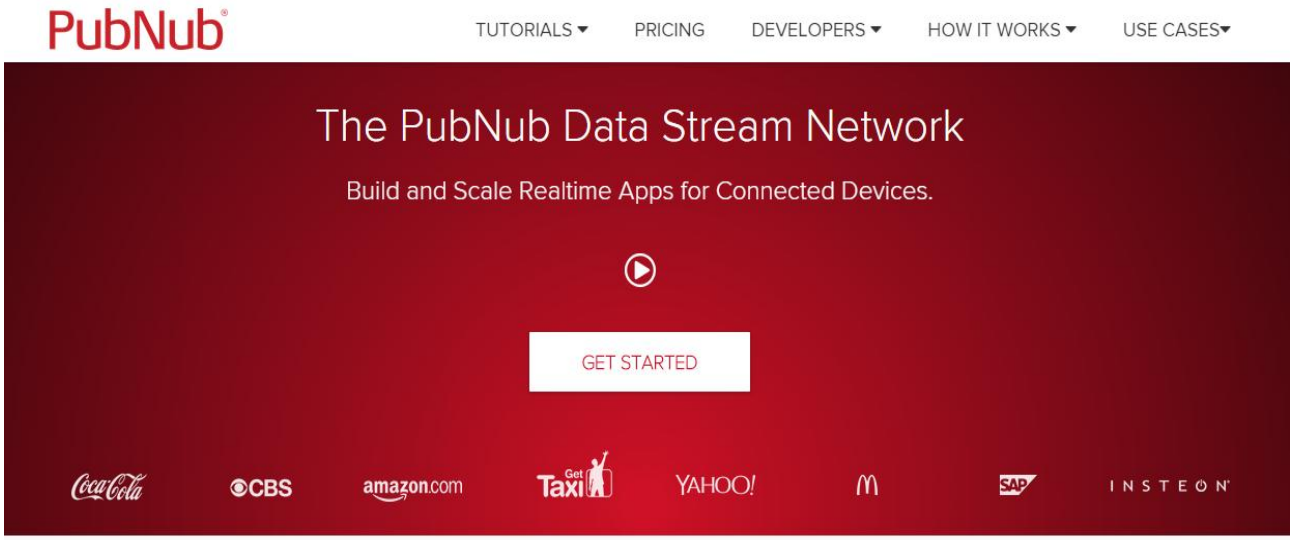
You can use your preferred serial terminal, such as PuTTY, Tera Term or etc. You can also use terminal window plug-in in Atmel studio. You can install it through the menu (Menu > Tools > Extension Manager).



3. What is PubNub

PubNub is a secure global Data Stream Network (DSN) and easy to use API that enable developers to connect, scale, and manage realtime applications and IoT devices. PubNub's sophisticated data streaming features are available to developers via over 70 SDKs for a wide variety of web, mobile, embedded, desktop and server platforms. PubNub's infrastructure provides ¼-second worldwide data transfer times, reliable message delivery and advanced data protection and privacy features. PubNub has proven ability to scale to hundreds of millions of devices over standard Internet protocols and a variety of transmission transports including Ethernet, Wi-Fi and Cellular networks.

For more information, please refer to http://www.pubnub.com. It looks like below at this moment.



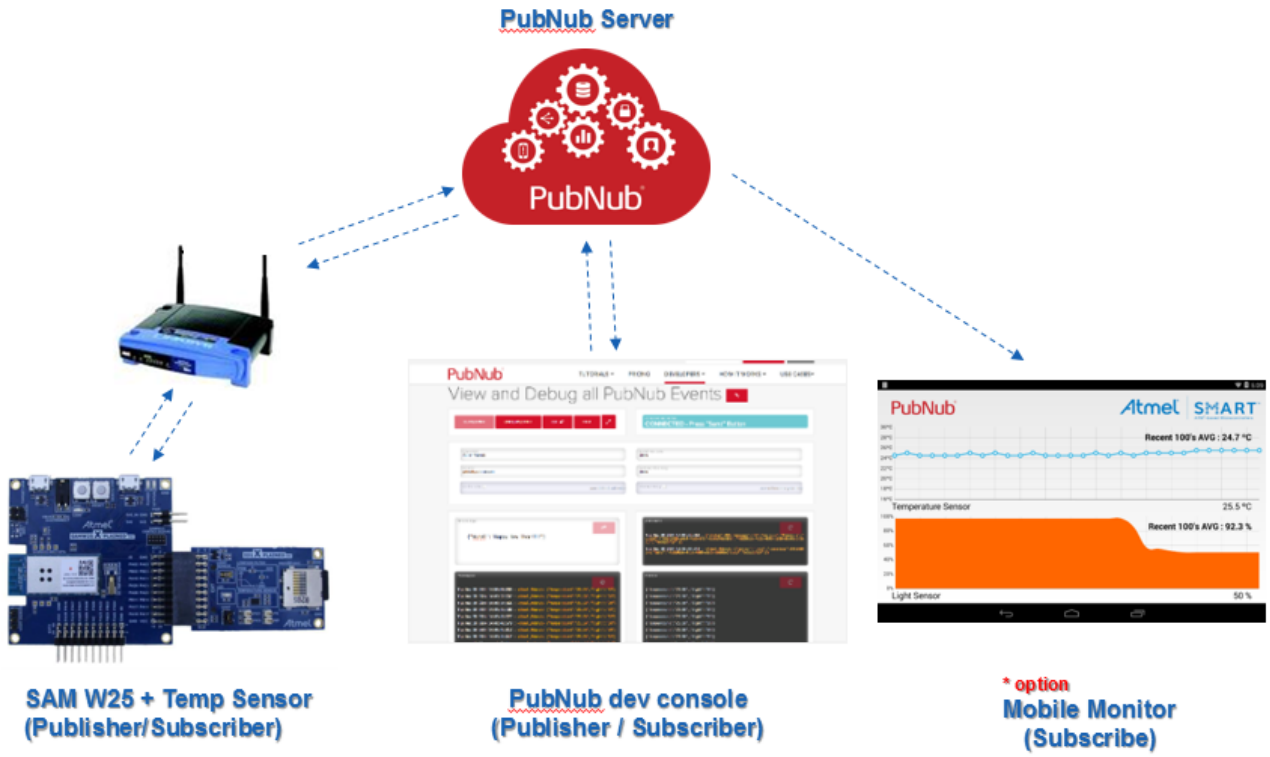
Developers can use this example without creating PubNub account. Try to choose 'DEVELOPERS -> DEV CONSOLE' in the above web site.

4. Ecosystem of Atmel SAM W25 + PubNub Data Stream Network

PubNub offers various SDKs, but C SDK was integrated to the example for ATMEL SMART IoT Device.

ATMEL SMART IoT Device is connected to the PubNub server and publishes/subscribes messages to communicate each other.

The PubNub console shows the received data from SAM W25 and you can send user messages to SAM W25 through the PubNub console.



5. Configuration of Example

This example demonstrates how to use the SAM W25 Xplained Pro board for PubNub Data Stream Network. It is to publish and subscribe messages between Atmel SAM W25 and PubNub. It uses the following hardware and server:

5.1 How to set SAM W25 Device

5.1.1 PubNub Key setting

Modify the following 3 string values in 'main.c' file. Those must be set in the PubNub console.

```
/** PubNub Settings*/
static const char pubkey[] = "demo";
static const char subkey[] = "demo";
static const char channel[] = "AtmelGallery_PubNub";
```

5.1.2 Start Device with AP provisioning

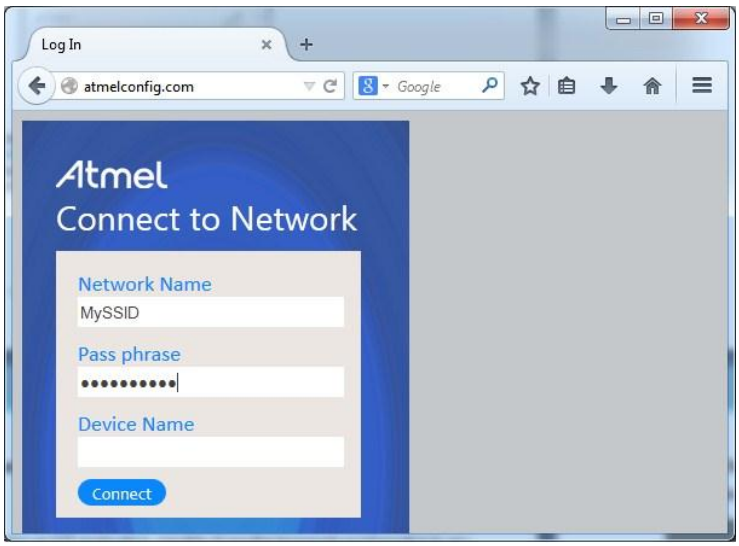
1) Build the project through "Menu > Build > Build Solution" (Shortcut: F7).
2) Run through "Menu > Debug > Start Without Debugging" (Shortcut: Ctrl+Alt+F5).

RESULT The application is now programmed and running. The following information will be displayed in the serial terminal.

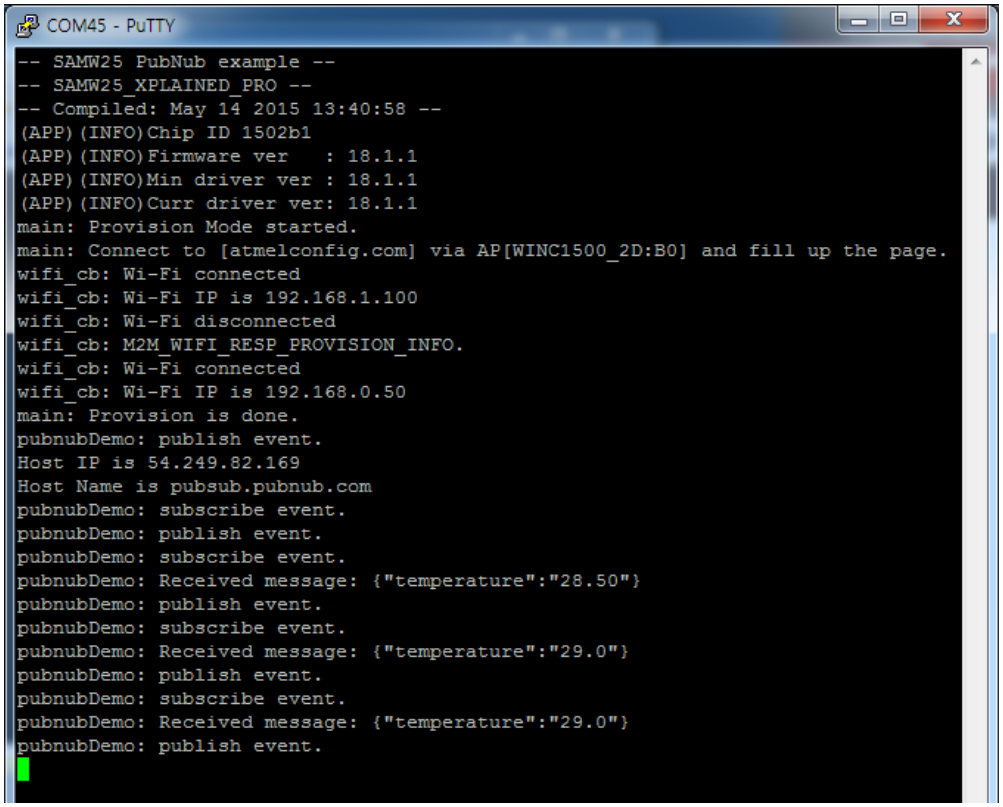
```
-- SAMW25 PubNub example –
-- SAMW25_XPLAINED_PRO –
-- Compiled: May 14 2015 13:40:58 –
main: Provision Mode started.
main: Connect to [atmelconfig.com] via AP[WINC1500_2D:B0] and fill up the
page.
```

3) The SAM W25 device now should be in AP mode and be listed as a Wi-Fi network with the same name as shown in the below picture (Atmel_SAMW25_XX:XX). Notice that the last two bytes of the MAC address are appended to the SSID. Simply connect your PC/Smartphone to the device in AP mode.

TO DO Once connected, just open your favorite web browser at the following address 'http://atmelconfig.com' and provide the required Network Name (SSID) and Pass phrase (Device Name can be blank) fields of the WiFi AP the SAM W25 is supposed to connect.



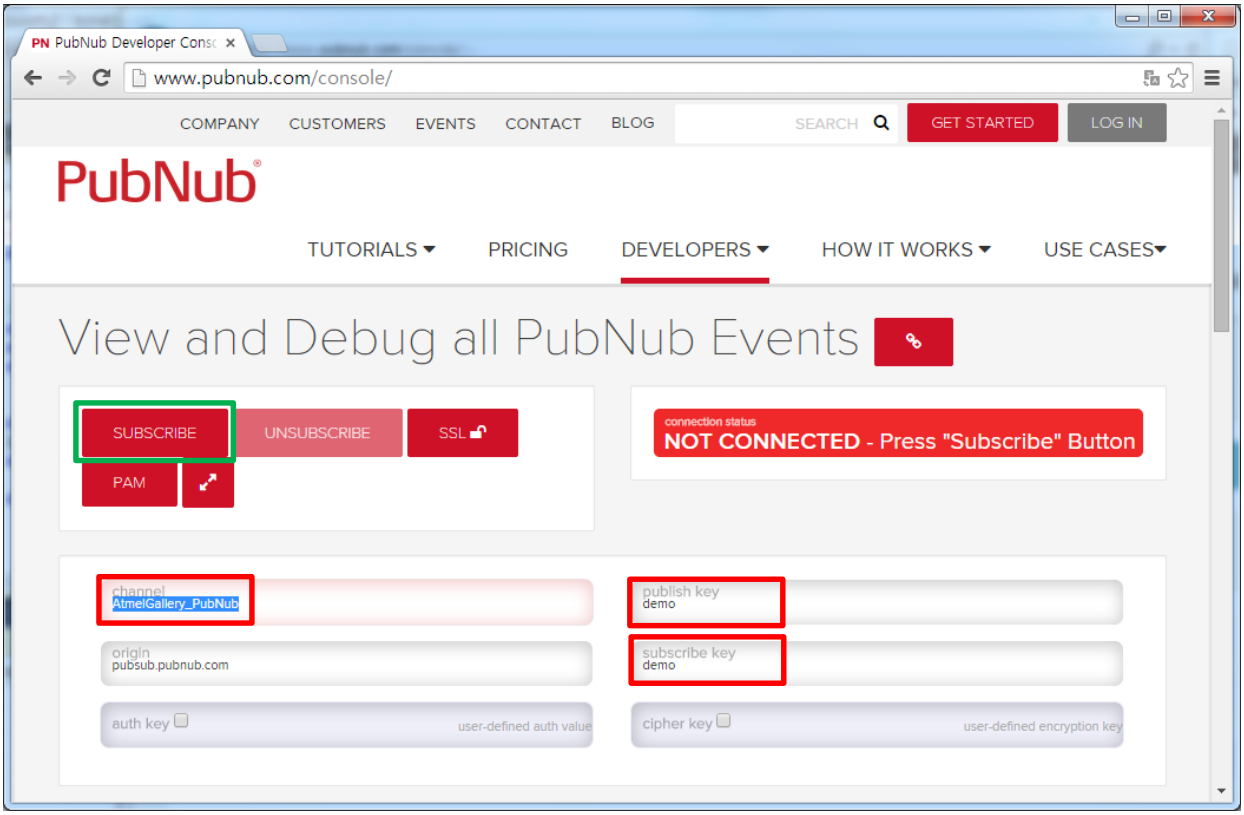
RESULT Now your SAM W25 device is provisioned and connected to the provided Wi-Fi AP.



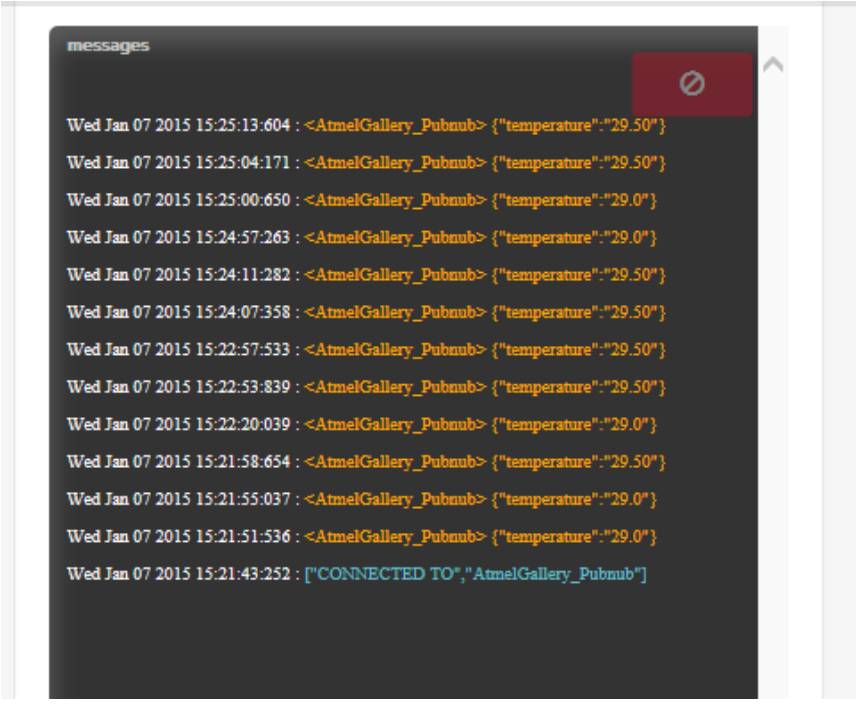
Now that the SAM W25 is connected to the AP with internet connectivity, it will immediately connect to the PubNub server and will start sending the temperature data at regular intervals.

5.2 How to set PubNub dev console

1) Open PubNub console site (http://www.pubnub.com/console/).
2) Set ‘channel’, ‘publish key’ and ‘subscribe key’ with your previous values at the section 5.1.1.
3) Click SUBSCRIBE button on the site.



4) After a while, the following messages should be shown.



5.3 How to set Android Mobile Monitor application

1) Open the project by using the 'Android Studio' IDE.

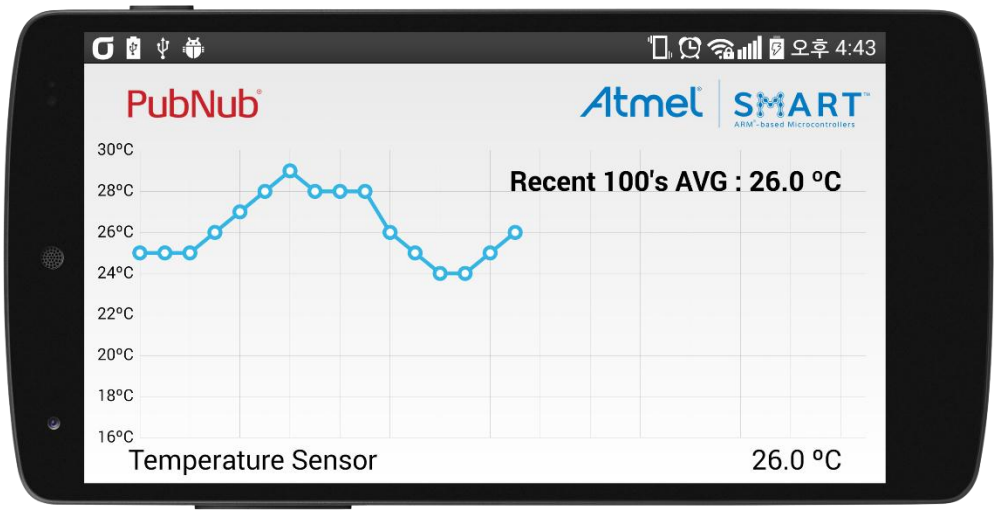
2) Modify the following 3 string values in 'MainActivity.java' file with your previous values at the section 5.1.1.

```
/** PubNub Settings*/
private final String PublishKey = "demo";
private final String SubscribeKey = "demo";
private final String Channel = "AtmelGallery_PubNub";
```

3) Connect usb cable from PC to android target.

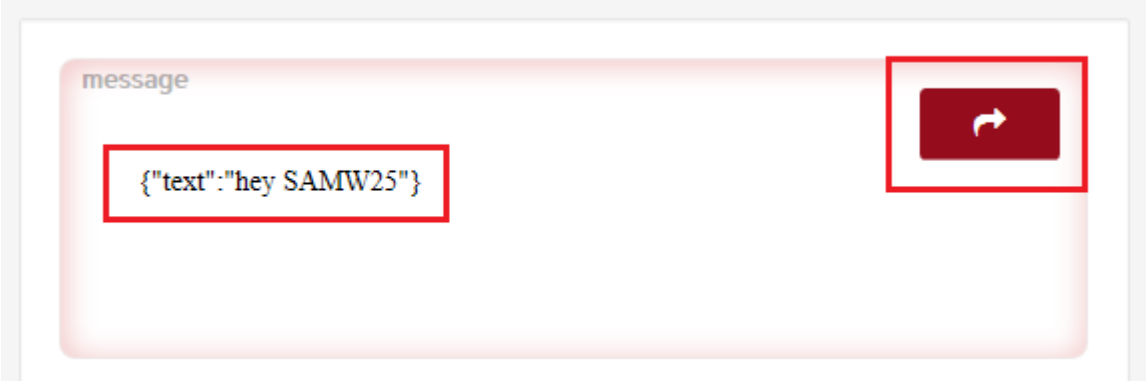
4) Start Android application at android target.

5) After a while, the chart will be shown such as below:

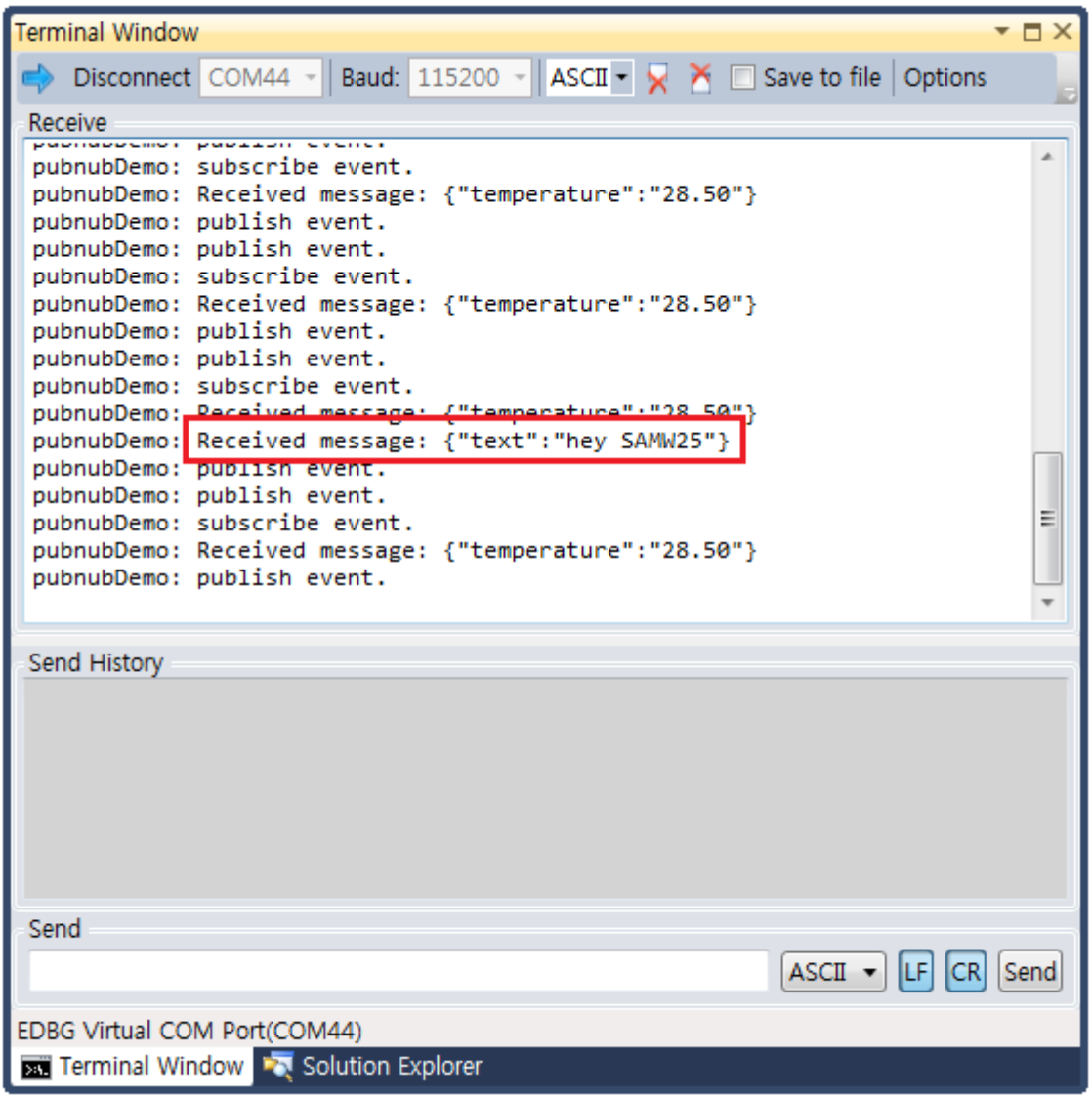


5.4 How to send a message from PubNub console to SAM W25 device.

1) Input any message into the message box below and click the send button.



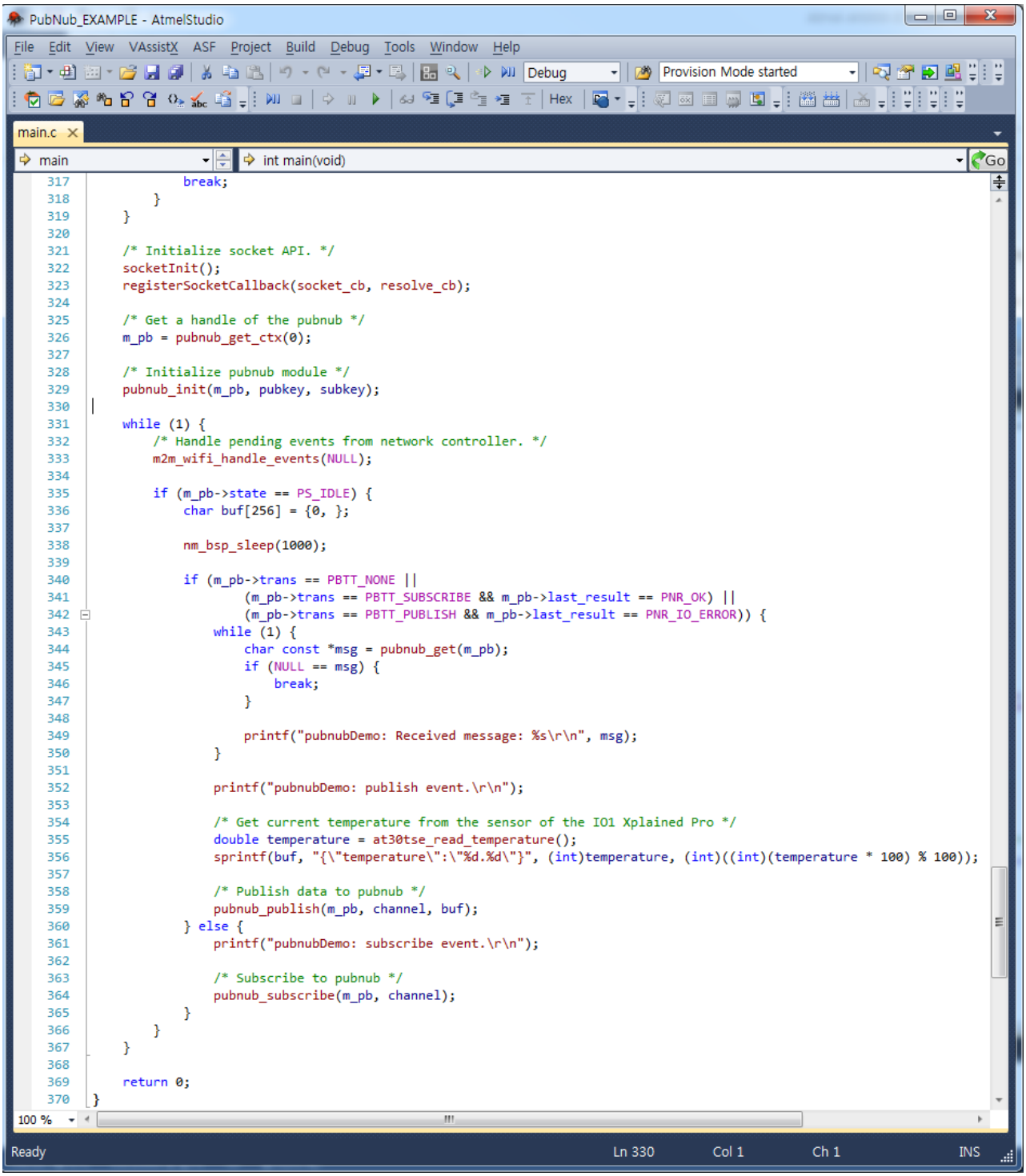
2) After a while, SAM W25 will get a message as below.


Terminal Window
Disconnect
COM44
Baud:
115200
ASCII
Save to file
Options
Receive
pubnubDemo: subscribe event.
pubnubDemo: Received message: {"temperature":"28.50"}
pubnubDemo: publish event.
pubnubDemo: publish event.
pubnubDemo: subscribe event.
pubnubDemo: Received message: {"temperature":"28.50"}
pubnubDemo: publish event.
pubnubDemo: publish event.
pubnubDemo: subscribe event.
pubnubDemo: Received message: {"temperature":"28.50"}
pubnubDemo: Received message: {"text":"hey SAMW25"}
pubnubDemo: publish event.
pubnubDemo: publish event.
pubnubDemo: subscribe event.
pubnubDemo: Received message: {"temperature":"28.50"}
pubnubDemo: publish event.
Send History
Send
ASCII
LF
CR
Send
EDBG Virtual COM Port(COM44)
Terminal Window
Solution Explorer

6. Programming Guide for SAM W25

For this application, you can start with "*main.c" file.*

The above file has all the codes needed to do the system initialization by controlling the components such as Uart, Temperature sensor, Wi-Fi driver and socket.




```c
            break;
        }
    }

    /* Initialize socket API. */
    socketInit();
    registerSocketCallback(socket_cb, resolve_cb);

    /* Get a handle of the pubnub */
    m_pb = pubnub_get_ctx(0);

    /* Initialize pubnub module */
    pubnub_init(m_pb, pubkey, subkey);

    while (1) {
        /* Handle pending events from network controller. */
        m2m_wifi_handle_events(NULL);

        if (m_pb->state == PS_IDLE) {
            char buf[256] = {0, };

            nm_bsp_sleep(1000);

            if (m_pb->trans == PBTT_NONE ||
                    (m_pb->trans == PBTT_SUBSCRIBE && m_pb->last_result == PNR_OK) ||
                    (m_pb->trans == PBTT_PUBLISH && m_pb->last_result == PNR_IO_ERROR)) {
                while (1) {
                    char const *msg = pubnub_get(m_pb);
                    if (NULL == msg) {
                        break;
                    }

                    printf("pubnubDemo: Received message: %s\r\n", msg);
                }

                printf("pubnubDemo: publish event.\r\n");

                /* Get current temperature from the sensor of the IO1 Xplained Pro */
                double temperature = at30tse_read_temperature();
                sprintf(buf, "{\"temperature\":\"%d.%d\"}", (int)temperature, (int)((int)(temperature * 100) % 100));

                /* Publish data to pubnub */
                pubnub_publish(m_pb, channel, buf);
            } else {
                printf("pubnubDemo: subscribe event.\r\n");

                /* Subscribe to pubnub */
                pubnub_subscribe(m_pb, channel);
            }
        }
    }

    return 0;
}
```

6.1 Initialize PubNub

The first step is to initialize the PubNub client module by calling the function “pubnub_init”. This function receives 3 parameters which are “publish key”, “subscribe key” and pointer variable from “pubnub_get_ctx” function. We need to add the following code just before the “for(;;system_sleep())” inside *main()* function

```
    /* Get a handle of the pubnub */
    m_pb = pubnub_get_ctx(0);

    /* Initialize pubnub module */
    pubnub_init(m_pb, pubkey, subkey);
```

6.2 Publish

In order to send the data from the temperature sensor on the I/O1 to the PubNub server, the function “*pubnub_publish*” should be used as below:

```
    while (1) {
        /* Handle pending events from network controller. */
        m2m_wifi_handle_events(NULL);

        if (m_pb->state == PS_IDLE) {
            char buf[256] = {0, };

            nm_bsp_sleep(1000);

            if (m_pb->trans == PBTT_NONE ||
                    (m_pb->trans == PBTT_SUBSCRIBE && m_pb->last_result == PNR_OK) ||
                    (m_pb->trans == PBTT_PUBLISH && m_pb->last_result == PNR_IO_ERROR)) {
                while (1) {
                    char const *msg = pubnub_get(m_pb);
                    if (NULL == msg) {
                        break;
                    }

                    printf("pubnubDemo: Received message: %s\r\n", msg);
                }

                printf("pubnubDemo: publish event.\r\n");

                /* Get current temperature from the sensor of the IO1 Xplained Pro */
                double temperature = at30tse_read_temperature();
                sprintf(buf, "{\"temperature\":\"%d.%d\"}", (int)temperature, (int)((int)(temperature * 100) % 100));

                /* Publish data to pubnub */
                pubnub_publish(m_pb, channel, buf);
            } else {
                printf("pubnubDemo: subscribe event.\r\n");

                /* Subscribe to pubnub */
                pubnub_subscribe(m_pb, channel);
            }
        }
    }
```

6.3 Subscribe

In order to subscribe to a Pubnub channel, we need to use *“pubnub_subscribe”* and “*pubnub_get”* function as below:

```
330
331     while (1) {
332         /* Handle pending events from network controller. */
333         m2m_wifi_handle_events(NULL);
334
335         if (m_pb->state == PS_IDLE) {
336             char buf[256] = {0, };
337
338             nm_bsp_sleep(1000);
339
340             if (m_pb->trans == PBTT_NONE ||
341                     (m_pb->trans == PBTT_SUBSCRIBE && m_pb->last_result == PNR_OK) ||
342                     (m_pb->trans == PBTT_PUBLISH && m_pb->last_result == PNR_IO_ERROR)) {
343                 while (1) {
344                     char const *msg = pubnub_get(m_pb);
345                     if (NULL == msg) {
346                         break;
347                     }
348
349                     printf("pubnubDemo: Received message: %s\r\n", msg);
350                 }
351
352                 printf("pubnubDemo: publish event.\r\n");
353
354                 /* Get current temperature from the sensor of the IO1 Xplained Pro */
355                 double temperature = at30tse_read_temperature();
356                 sprintf(buf, "{\"temperature\":\"%d.%d\"}", (int)temperature, (int)((int)(temperature * 100) % 100));
357
358                 /* Publish data to pubnub */
359                 pubnub_publish(m_pb, channel, buf);
360             } else {
361                 printf("pubnubDemo: subscribe event.\r\n");
362
363                 /* Subscribe to pubnub */
364                 pubnub_subscribe(m_pb, channel);
365             }
366         }
367     }
368
```

7. Conclusion

This application note explained how to use the SAM W25 Xplained Pro board for PubNub Data Stream Network.

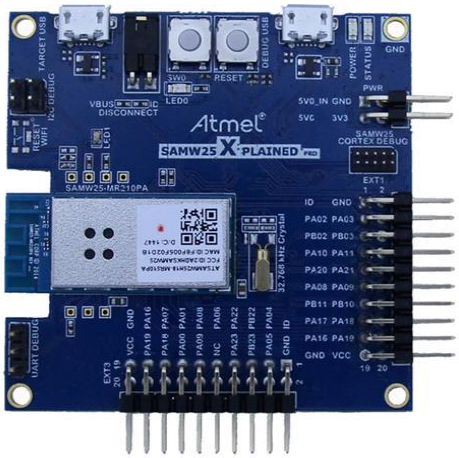
The following topics have been covered:

- Example source tree
- Ecosystem of SAM W25 + PubNub
- Configuration of the Example

8. Appendix – Device Information

Atmel SMART : SAMW25-XPRO Evaluation kit.

- Turnkey system with integrated software that includes TLS 1.0 and a TCP/IP stack WPA2 personal and enterprise security
- Single-band 2.4GHz IEEE 802.11 b/g/n Wi-Fi ATWINC1500
- Atmel | SMART ARM Cortex M0+-based SAM D21; 256KB Flash; 32KB SRAM
- Serial Peripheral Interface (SPI)
- Over-the-air updates
- ATECC108A CryptoAuthentication™ engine with ultra-secure hardware-based key storage for secure connectivity
- Extreme low power
- Compact footprint: 33.8 x 14.9mm
- Operating Voltage: 2.7 to 3.6V
- Worldwide acceptance: FCC (USA), CE (Europe) and TELEC
- RoHS compliant
- Network services – DHCP, DNS, TCP/IP (IPv4), UDP, HTTP, HTTPS
- Here is the link for more detailed information : Atmel SAMW25 Xplained Pro



Atmel ATIO1-XPRO Sensor extension board.

- microSD card connector
  - 2GB microSD card included
  - Accessed with SPI interface
- PWM
  - LED control
  - PWM → Low pass filter → ADC
- ADC
  - PWM → Low pass filter → ADC
  - Light sensor
- UART
  - Loopback interface via pin header
- TWI
  - AT30TSE758 Temperature sensor with EEPROM
- Xplained Pro hardware identification system
- Here is the link for more detailed information : Atmel IO1 Xplained Pro

9. Revision History

Doc. Rev.	Date	Comments
XXXXXA	May 1st / 2015	Initial document release

Enabling Unlimited Possibilities®

Atmel Corporation
1600 Technology Drive
San Jose, CA 95110
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan G.K.
16F Shin-Osaki Kangyo Bldg.
1-6-4 Osaki, Shinagawa-ku
Tokyo 141-0032
JAPAN
Tel: (+81)(3) 6417-0300
Fax: (+81)(3) 6417-0370