

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

Prerequisites	1
Introduction	1
Icon Key Identifiers	3
1. Example Source Tree	4
2. How to Build & Execute.....	4
3. What is PubNub	8
4. Ecosystem of Atmel SAM W25 + PubNub Data Stream Network	9
5. Configuration of Example.....	10
5.1 How to set SAM W25 Device (AP provisioning)	10
5.2 How to set PubNub server	12
5.3 How to set Android application	14
5.4 How to send a message from PubNub console to SAM W25 device.	14
6. Programming Guide for SAM W25.....	16
6.1 Initialize PubNub	17
6.2 Publish	17
6.3 Subscribe	17
7. Conclusion	19
8. Appendix – Device Information	20
9. Revision History	21

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

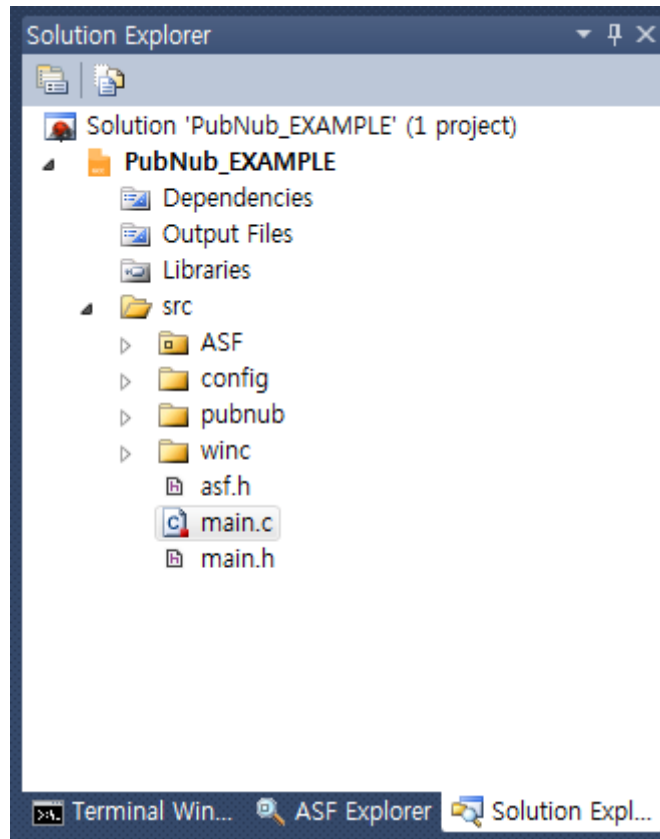


EXECUTE

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

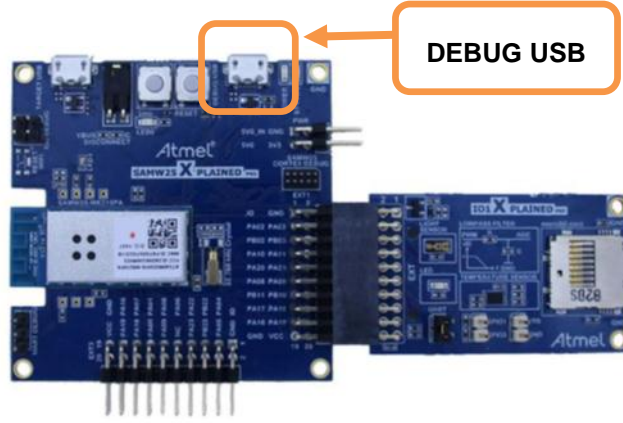



EXECUTE Build the solution (F7) and ensure you get no error:

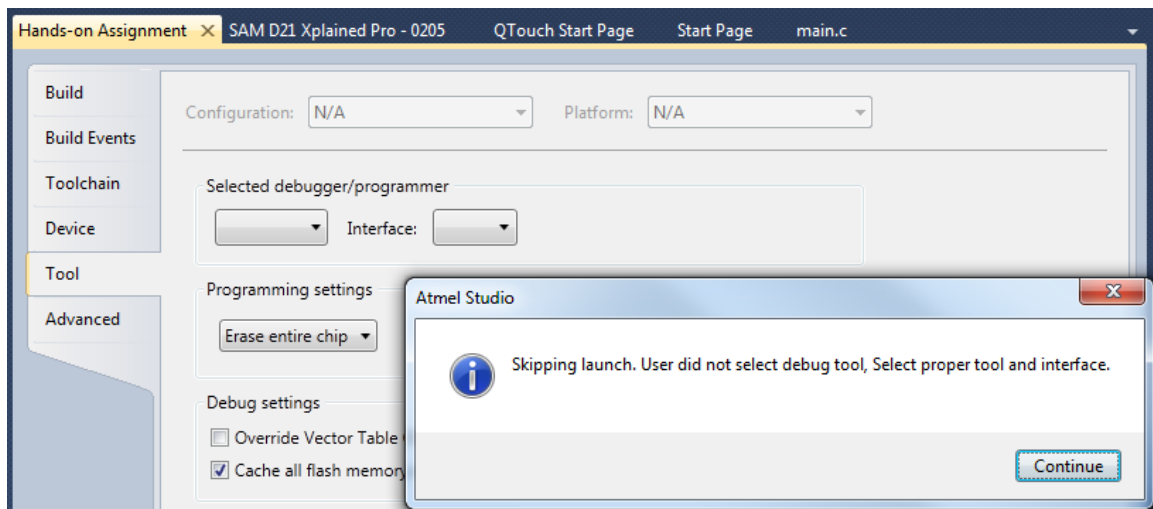


EXECUTE Program the SAM W25 Xplained Pro.

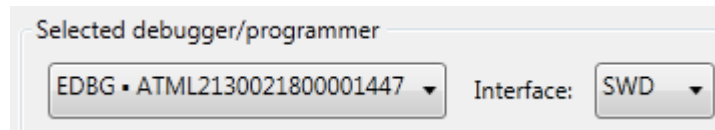
- Connect the IO1 extension to the SAM Xplained Pro as displayed below:



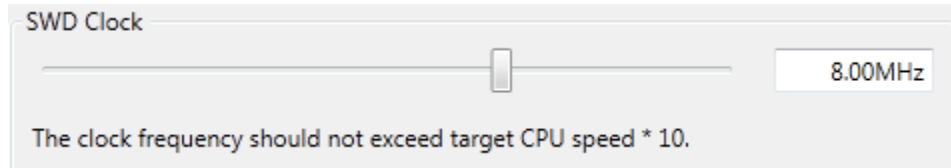
- Connect the SAM W25 Xplained Pro board to your PC by using DEBUG USB connector.
- Program the application by clicking on the Start Debugging and Break icon: 
- You will be asked to select your debug tool as below:





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



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

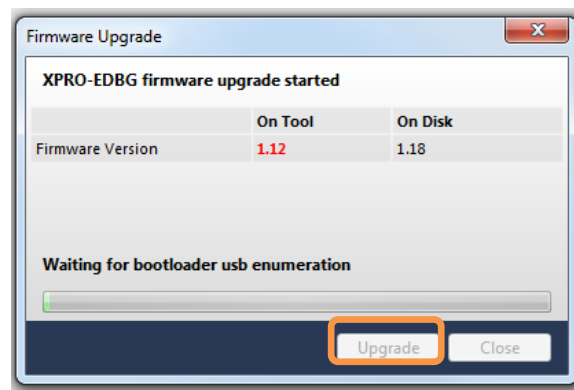


- Click again on the Start Debugging and Break icon: 
- The application will be programmed in the SAM W25 embedded flash and breaks at main function.
Click on Continue to execute the application: 



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).

The screenshot shows the 'Terminal Window' in Atmel Studio. The window title is 'Terminal Window'. The top bar contains a 'Disconnect' button, a dropdown menu set to 'COM44', a 'Baud:' dropdown set to '115200', an 'ASCII' dropdown, and buttons for 'Save to file' and 'Options'. The main area is divided into 'Receive' and 'Send' sections. The 'Receive' section displays the following output:

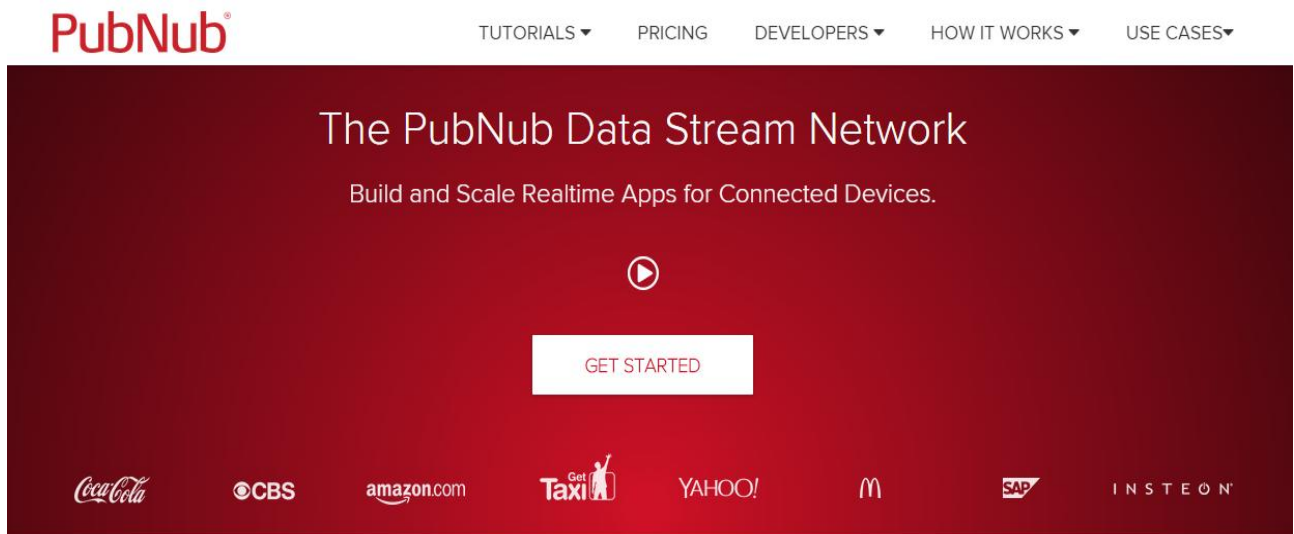
```
-- WINC1500 PubNub example --  
-- SAMW25_XPLAINED_PRO --  
-- Compiled: Apr 29 2015 18:27:50 --  
Wi-Fi connected  
Wi-Fi IP is 192.168.0.5  
pubnubDemo: publish event.  
pubnubDemo: subscribe event.  
pubnubDemo: publish event.  
pubnubDemo: publish event.  
pubnubDemo: subscribe event.  
pubnubDemo: Received message: {"temperature":"30.0"}  
pubnubDemo: publish event.  
pubnubDemo: publish event.  
pubnubDemo: subscribe event.  
pubnubDemo: Received message: {"temperature":"30.0"}  
pubnubDemo: publish event.  
pubnubDemo: publish event.
```

The 'Send' section has a text input field and buttons for 'ASCII', 'LF', 'CR', and 'Send'. Below the terminal window, the status bar shows 'EDBG Virtual COM Port(COM44)' and a tabbed interface with 'Terminal Window', 'ASF Explorer', 'Processor', 'Solution Explorer', and 'Properties'.

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 1/4-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.



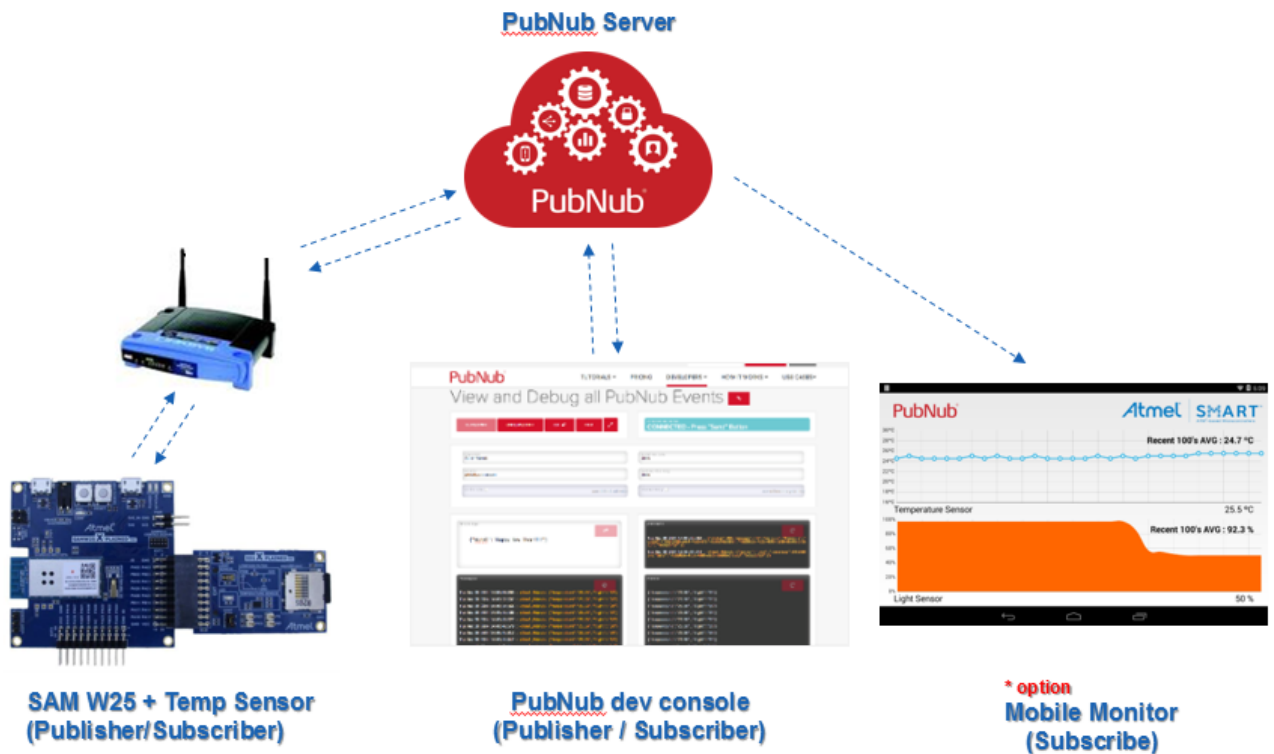
Developers can use this example without creating PubNub account. Try to choose 'DEVELOPERS -> DEV CONSOLE' in the above web site.

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.

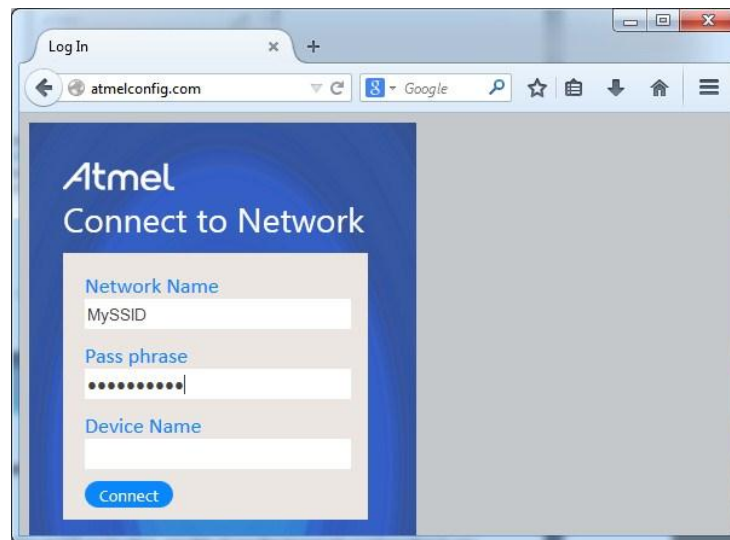
```
-- SAMW25 PubNub example -  
-- SAMW25_XPLAINED_PRO -  
-- Compiled: May 14 2015 13:40:58 -  
main: Provision Mode started.  
main: Connect to [atmelconfig.com] via AP[WINC1500_2D:B0] and fill up the  
page.
```

- 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.

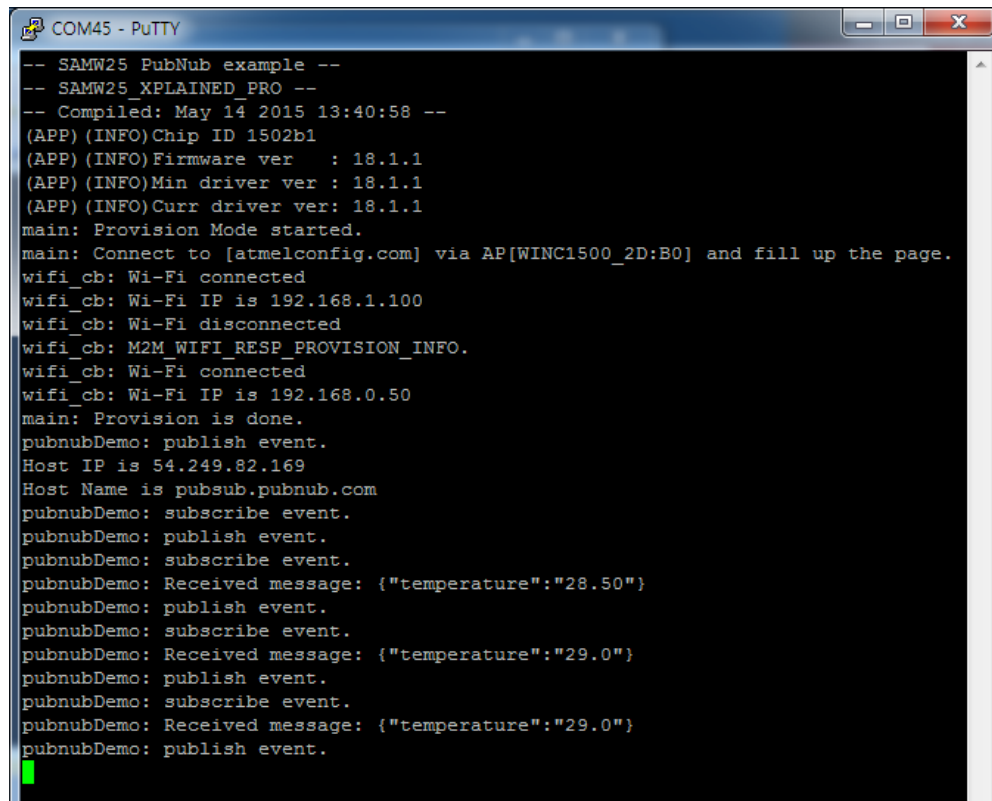


TO DO

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.

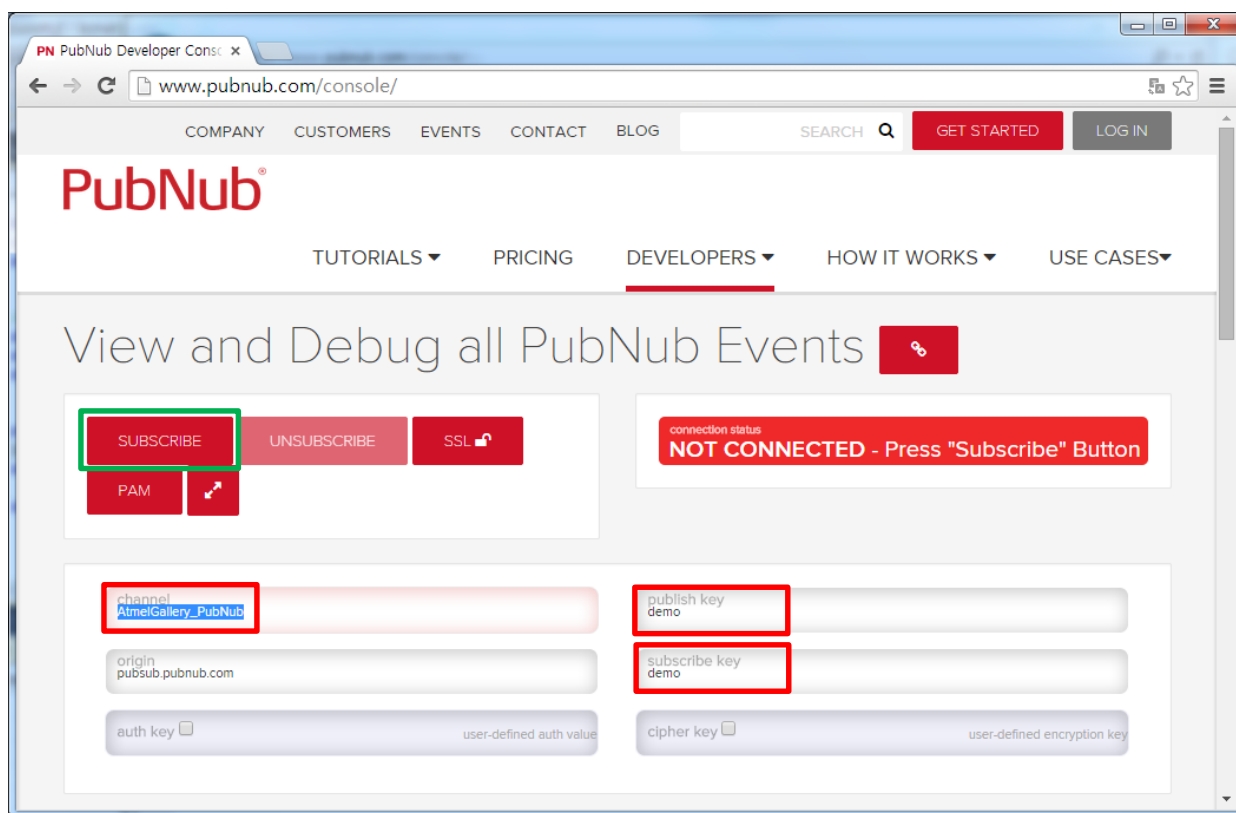


```
-- SAMW25 PubNub example --
-- SAMW25_XPLAINED_PRO --
-- Compiled: May 14 2015 13:40:58 --
(APP) (INFO) Chip ID 1502b1
(APP) (INFO) Firmware ver : 18.1.1
(APP) (INFO) Min driver ver : 18.1.1
(APP) (INFO) Curr driver ver: 18.1.1
main: Provision Mode started.
main: Connect to [atmelconfig.com] via AP[WINC1500_2D:B0] and fill up the page.
wifi_cb: Wi-Fi connected
wifi_cb: Wi-Fi IP is 192.168.1.100
wifi_cb: Wi-Fi disconnected
wifi_cb: M2M_WIFI_RESP_PROVISION_INFO.
wifi_cb: Wi-Fi connected
wifi_cb: Wi-Fi IP is 192.168.0.50
main: Provision is done.
pubnubDemo: publish event.
Host IP is 54.249.82.169
Host Name is pubsub.pubnub.com
pubnubDemo: subscribe event.
pubnubDemo: publish event.
pubnubDemo: subscribe event.
pubnubDemo: Received message: {"temperature":"28.50"}
pubnubDemo: publish event.
pubnubDemo: subscribe event.
pubnubDemo: Received message: {"temperature":"29.0"}
pubnubDemo: publish event.
pubnubDemo: subscribe event.
pubnubDemo: Received message: {"temperature":"29.0"}
pubnubDemo: publish event.
```

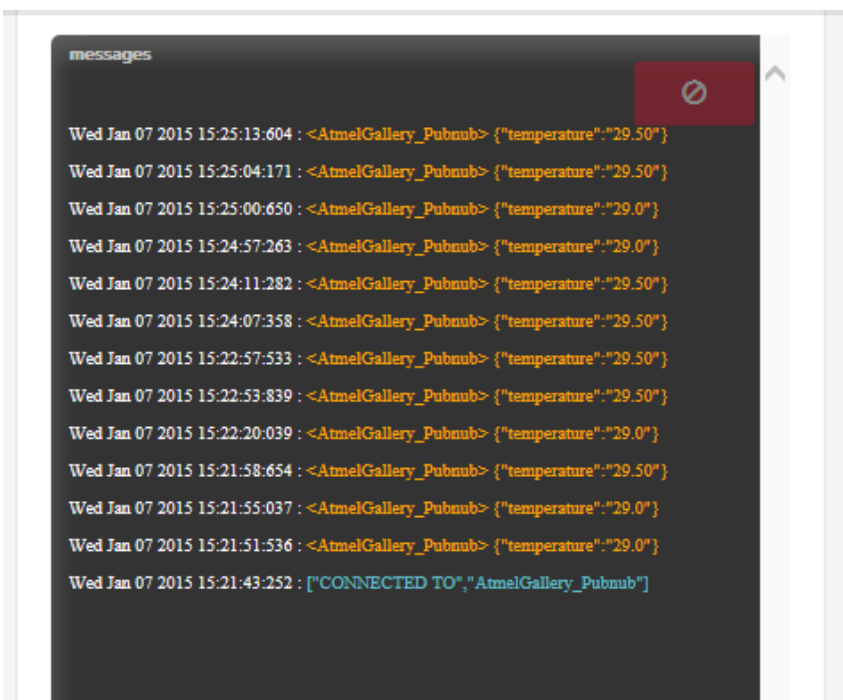
Now that the SAM W25 is connected to the AP with internet connectivity, it will immediately connect to the PubNub server and will start sending the temperature data at regular intervals.

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.

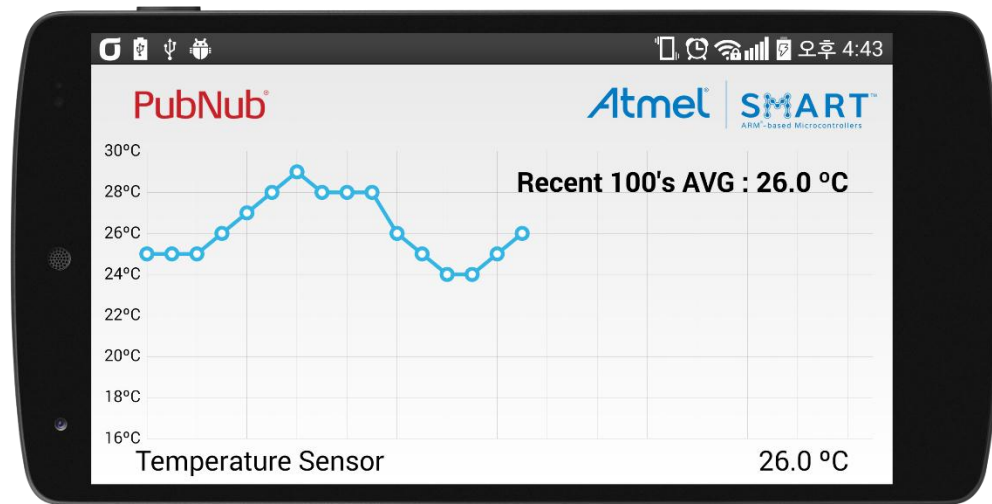


5.3 How to set Android Mobile Monitor application

- 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:

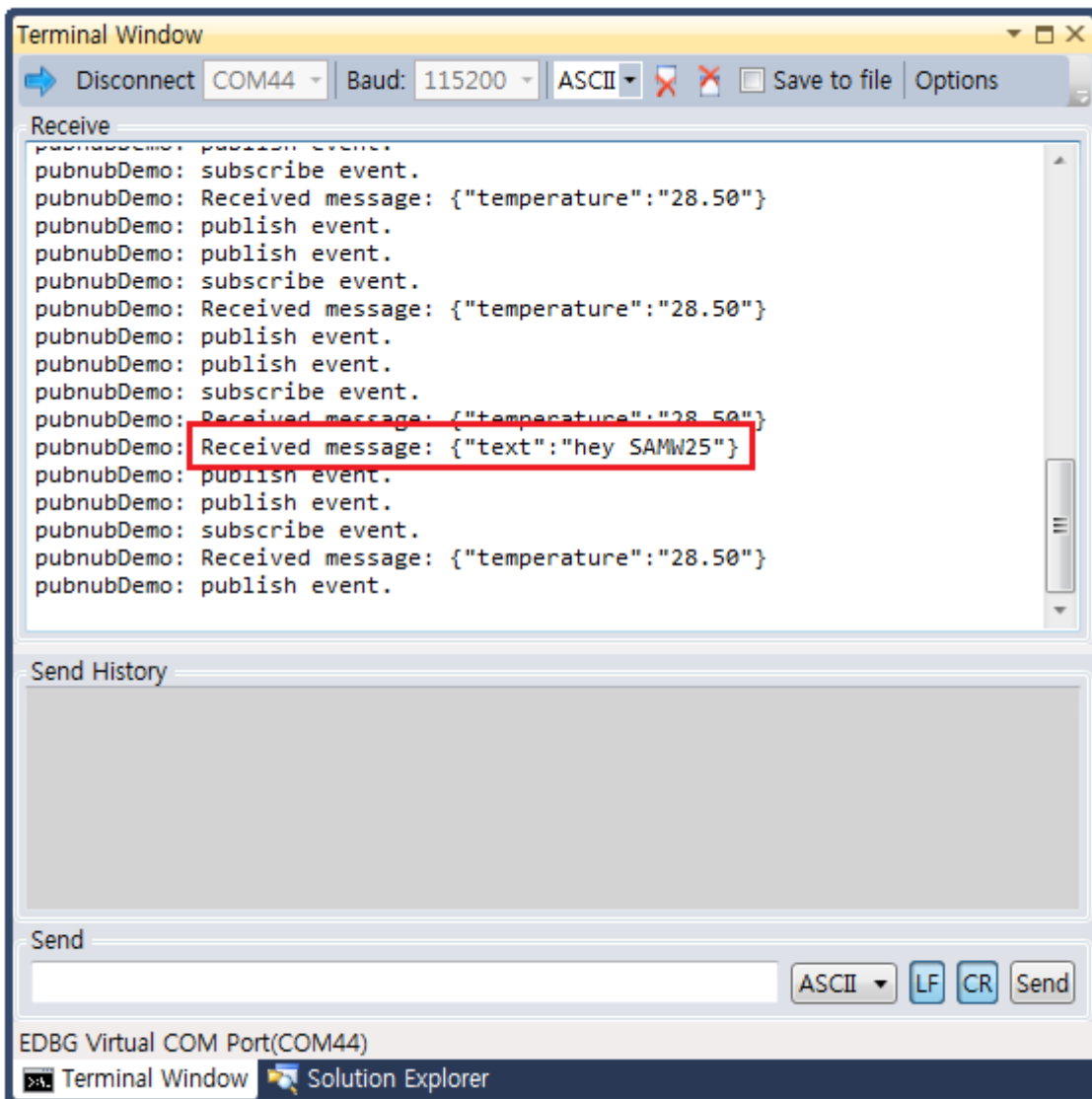


5.4 How to send a message from PubNub console to SAM W25 device.

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

The screenshot shows a web interface for sending a message. It features a large text input area with the placeholder text 'message'. Inside this area, a red box highlights the JSON message payload: `{"text": "hey SAMW25"}`. To the right of the input area, there is a red button with a white right-pointing arrow, which is also highlighted by a red box. The entire interface is set against a light gray background.

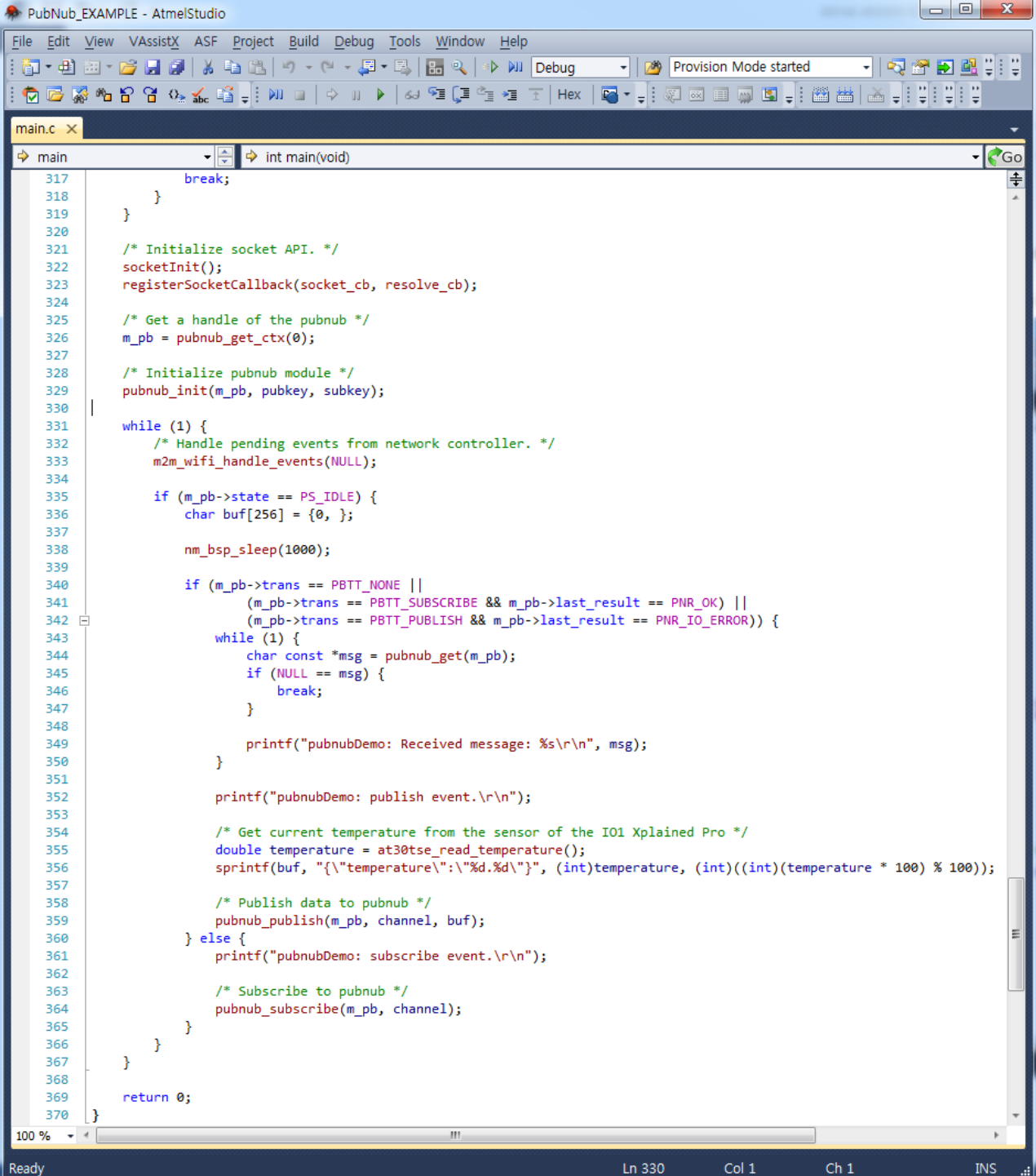
- 2) After a while, SAM W25 will get a message as below.



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.



```
main.c
main
int main(void)
317     break;
318 }
319 }
320
321 /* Initialize socket API. */
322 socketInit();
323 registerSocketCallback(socket_cb, resolve_cb);
324
325 /* Get a handle of the pubnub */
326 m_pb = pubnub_get_ctx(0);
327
328 /* Initialize pubnub module */
329 pubnub_init(m_pb, pubkey, subkey);
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                 printf("pubnubDemo: Received message: %s\r\n", msg);
349             }
350             printf("pubnubDemo: publish event.\r\n");
351
352             /* Get current temperature from the sensor of the I01 Xplained Pro */
353             double temperature = at30tse_read_temperature();
354             sprintf(buf, "{\"temperature\": \"%d.%d\"}", (int)temperature, (int)((int)(temperature * 100) % 100));
355
356             /* Publish data to pubnub */
357             pubnub_publish(m_pb, channel, buf);
358         } else {
359             printf("pubnubDemo: subscribe event.\r\n");
360
361             /* Subscribe to pubnub */
362             pubnub_subscribe(m_pb, channel);
363         }
364     }
365 }
366
367 return 0;
368
369 }
370 }
```


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

```
324
325      /* Get a handle of the pubnub */
326      m_pb = pubnub_get_ctx(0);
327
328      /* Initialize pubnub module */
329      pubnub_init(m_pb, pubkey, subkey);
330
```

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:

```
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 I01 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         }
361         else {
362             printf("pubnubDemo: subscribe event.\r\n");
363
364             /* Subscribe to pubnub */
365             pubnub_subscribe(m_pb, channel);
366         }
367     }
368 }
```

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 == PBT_NONE ||
341             (m_pb->trans == PBT_SUBSCRIBE && m_pb->last_result == PNR_OK) ||
342             (m_pb->trans == PBT_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                 printf("pubnubDemo: Received message: %s\r\n", msg);
349             }
350
351             printf("pubnubDemo: publish event.\r\n");
352
353             /* Get current temperature from the sensor of the IOT Xplained Pro */
354             double temperature = at30tse_read_temperature();
355             sprintf(buf, "{\"temperature\": \"%d.%d\"}", (int)temperature, (int)((int)(temperature * 100) % 100));
356
357             /* Publish data to pubnub */
358             pubnub_publish(m_pb, channel, buf);
359         } else {
360             printf("pubnubDemo: subscribe event.\r\n");
361
362             /* Subscribe to pubnub */
363             pubnub_subscribe(m_pb, channel);
364         }
365     }
366 }
367
368

```

7. Conclusion

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

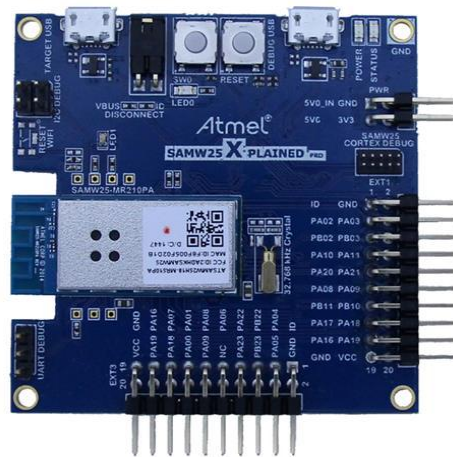
The following topics have been covered:

- 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

© 2012 Atmel Corporation. All rights reserved. / Rev.: XXXA-10

Atmel®, Atmel logo and combinations thereof, Enabling Unlimited Possibilities®, and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and products descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.

