eLearning

Wi-Fi® SoC Module *WFI32E01*

參考範例應用篇

A Leading Provider of Smart, Connected and Secure Embedded Control Solutions

King Chou, Pr. ESE

2021

Microchip - Chinese [Traditional]

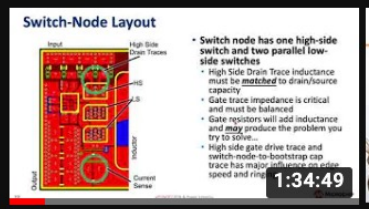
199 位訂閱者

已訂閱

首頁 影片 播放清單 頻道 討論 簡介

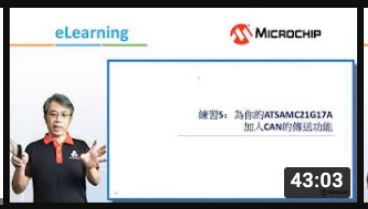
上傳的影片

▶ 全部播放



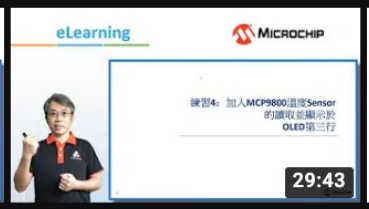
ePOW007——電源分佈網路與電源完整性

觀看次數：29次・3 週前



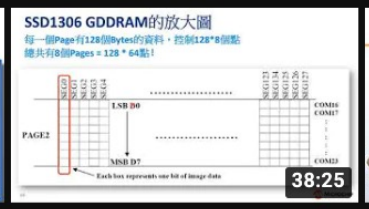
ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：97次・1 個月前



ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：28次・1 個月前



ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：31次・1 個月前

ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：39次・1 個月前

ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：27次・1 個月前

Featured Channels

Microchip Technology
5.38萬 位訂閱者

Microchip - Chinese [Simplified]
540 位訂閱者

Microchip - Japanese
343 位訂閱者

Microchip - Korean
1310 位訂閱者

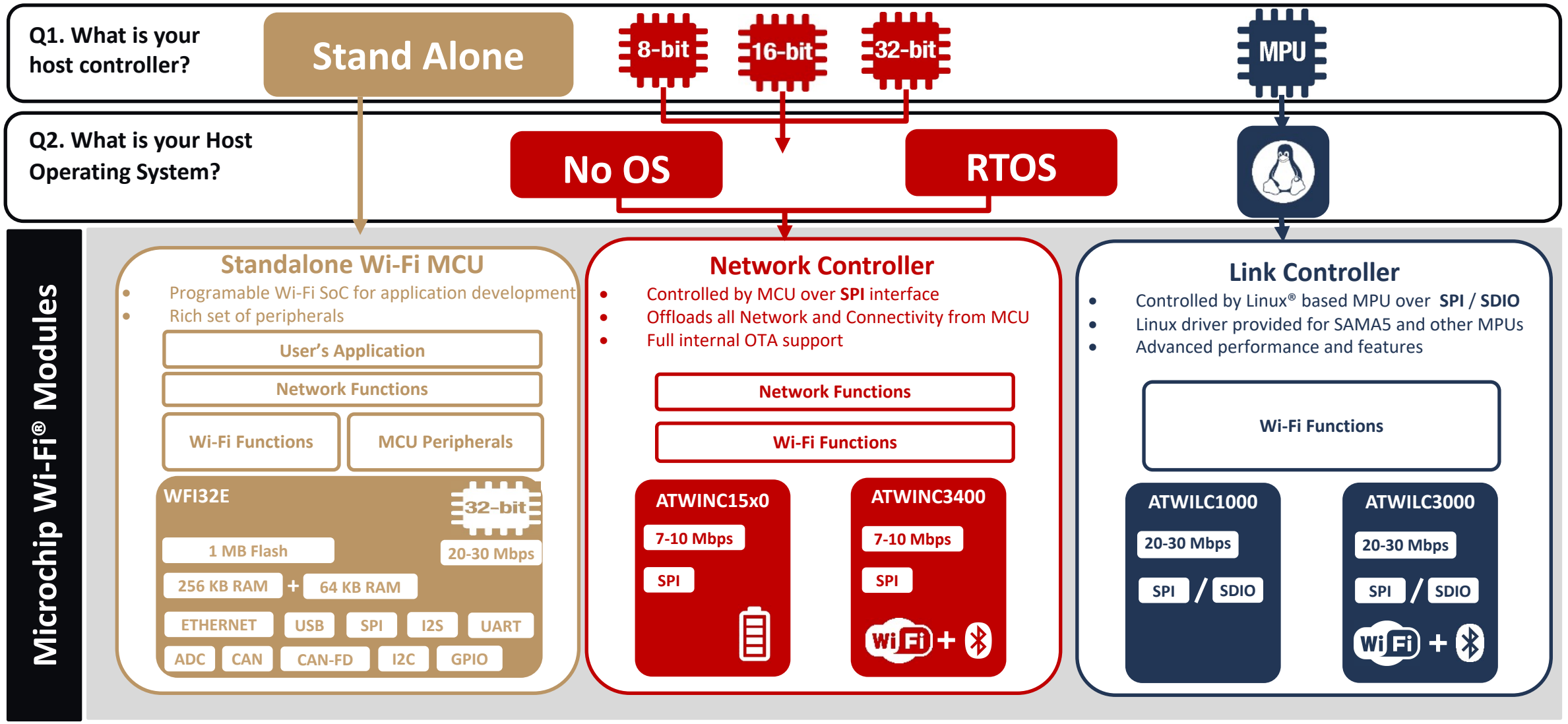
Microchip Makes
1.87萬 位訂閱者

eLearning

Wi-Fi® SoC Module *WFI32E01*
介紹及軟體應用篇

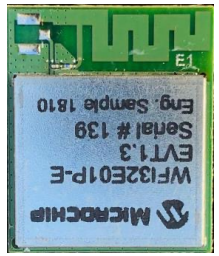
A Leading Provider of Smart, Connected and Secure Embedded Control Solutions

Microchip Wi-Fi® Overview

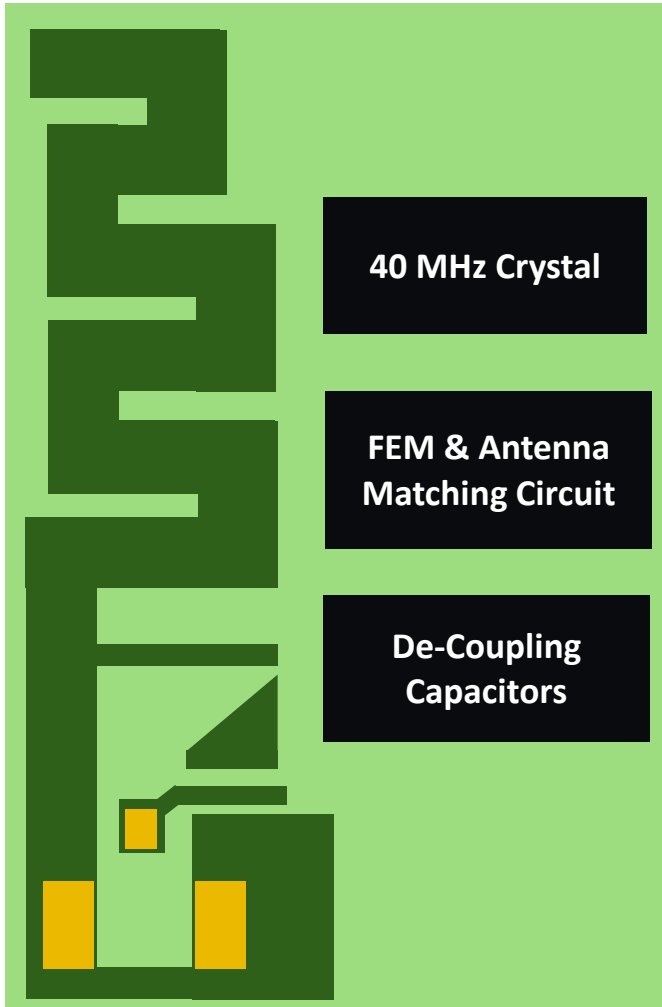


WFI32E01UE

Block Diagram for 54-Pad Module



- **Module Package:**
  - 54 pins SMD
  - 24.5 x 20.5 x 2.5 mm
- **Single Supply Voltage**
  - Operating Voltage: 3.0V to 3.6V
- **Operating Temperature:** -40°C to +85 °C



Encryption Engine
Data Crypto – AES, TDES
Public Key – ECC, ECHD, ECDSA, Curve25519, Ed25519

MIPS uAptiv UP Core
- 200 MHz
- FPU / DSP / DMA
- ICSP / Debug

Wi-Fi® 802.11 b/g/n
w/ Low Power Modes

1 MB Embedded Flash

320 KB RAM
- 256 KB SRAM
- 64 KB Wi-Fi Data Buffer

Die Temp Sensor

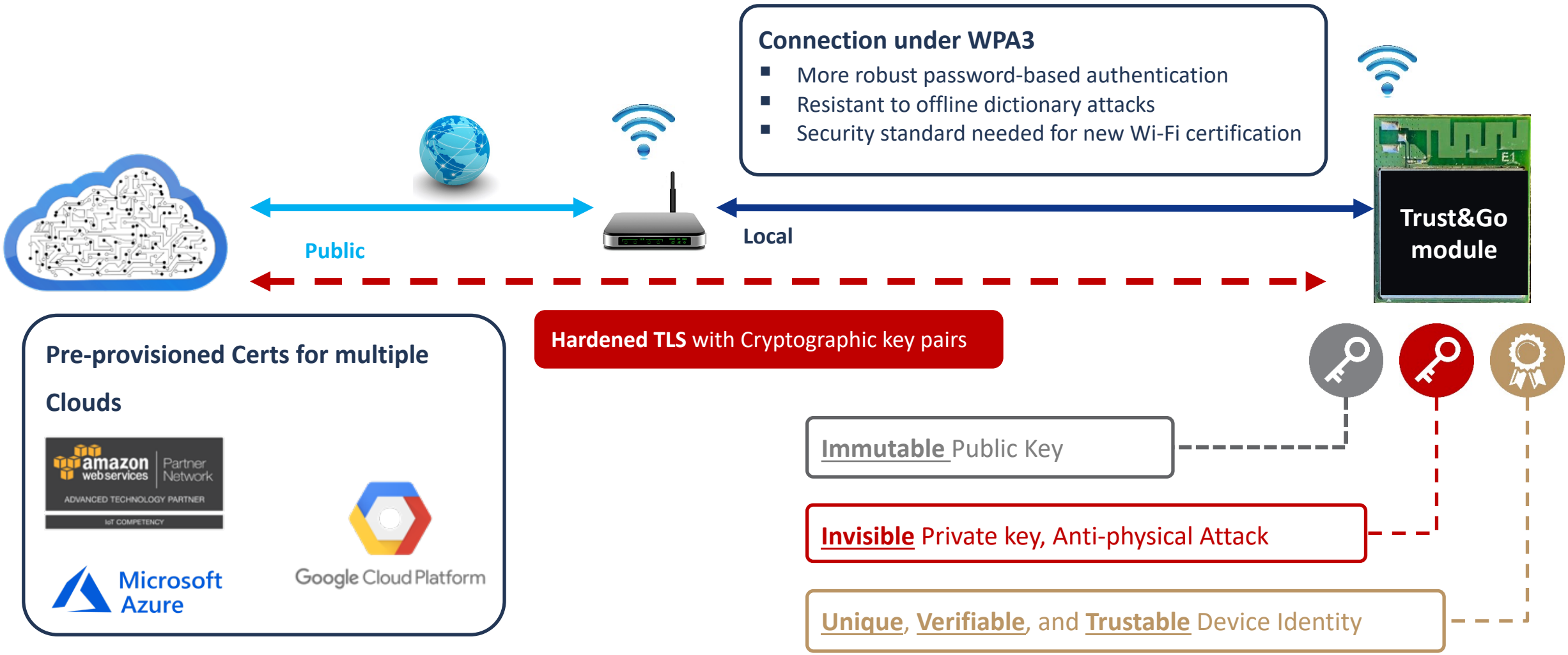
WWDT / DMT

Peripherals & Interfaces

- 8 MHz Internal Osc.
- LPRC, REFO
- 2x12-bit ADC, 1 MSPS, 12 ch.
  - Supports CVD Touch
- 6x32-bit / 16-bit Timers
- 4xCapture/Compare/PWM
- Peripheral Trigger Generator
- USB 2.0 Full Speed
- 10/100 Ethernet (RMII, IEEE1588)
- 3xUART/2xSPI/I²C/SQI/2xI²S
- CAN & CAN-FD
- 37 GPIO

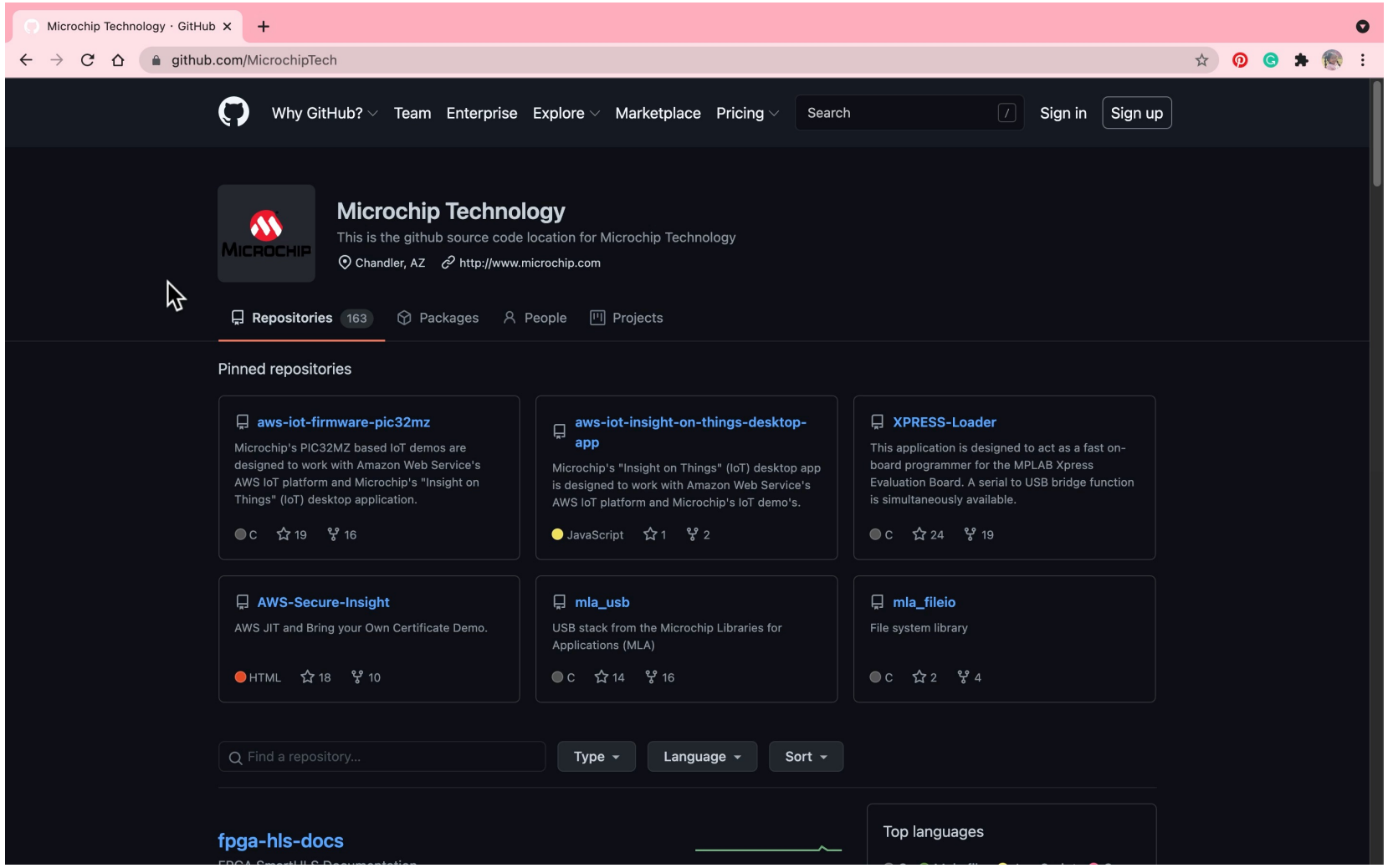
Build a Secure and Trustful IoT Connection

Trust&Go Wi-Fi® module: WFI32E01PC-I



PIC32MZW1_Workshop

- https://github.com/MicrochipTech/PIC32MZW1_Workshop



Agenda

eLearning

- **Reference Example Codes**
  - Develop a Secured Smart Home Application from scratch
  - Wi-Fi ® Touch and OLED Display
  - Wi-Fi Provisioning over BLE
  - Wi-Fi to CAN Bus Bridge

MICROCHIP

Agenda

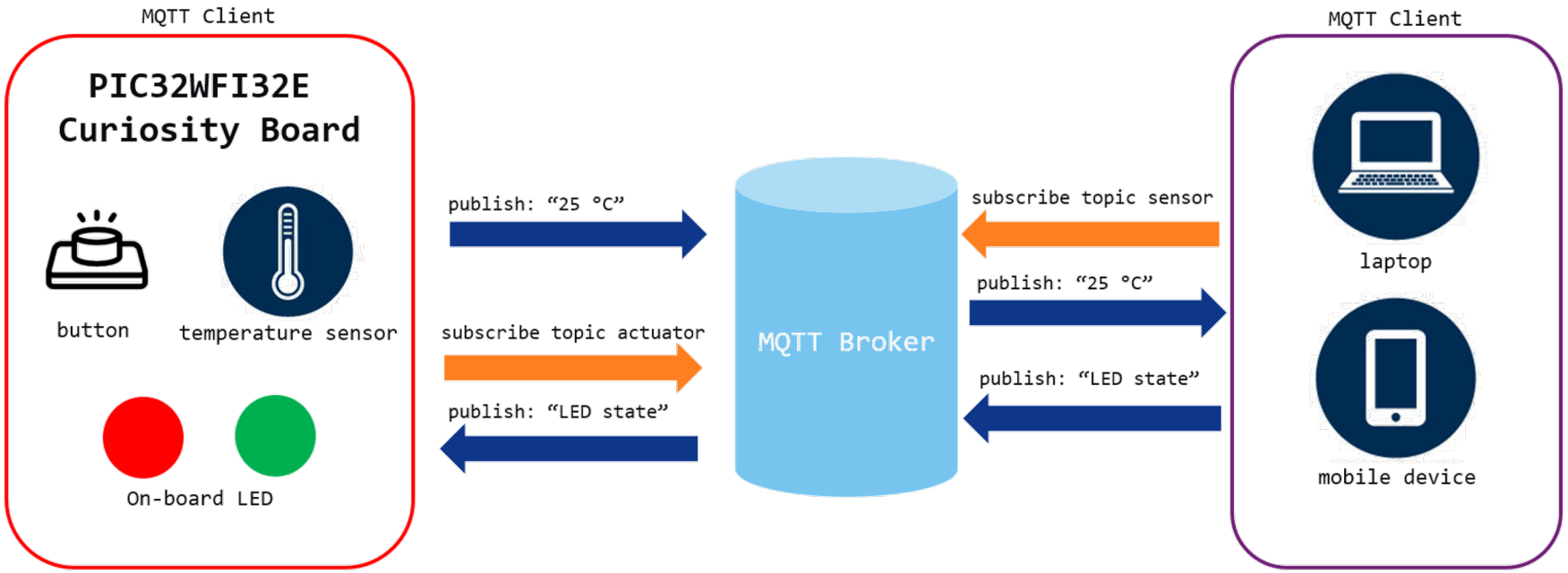
eLearning

- **Reference Example Codes**
  - Develop a Secured Smart Home Application from scratch
  - Wi-Fi ® Touch and OLED Display
  - Wi-Fi Provisioning over BLE
  - Wi-Fi to CAN Bus Bridge

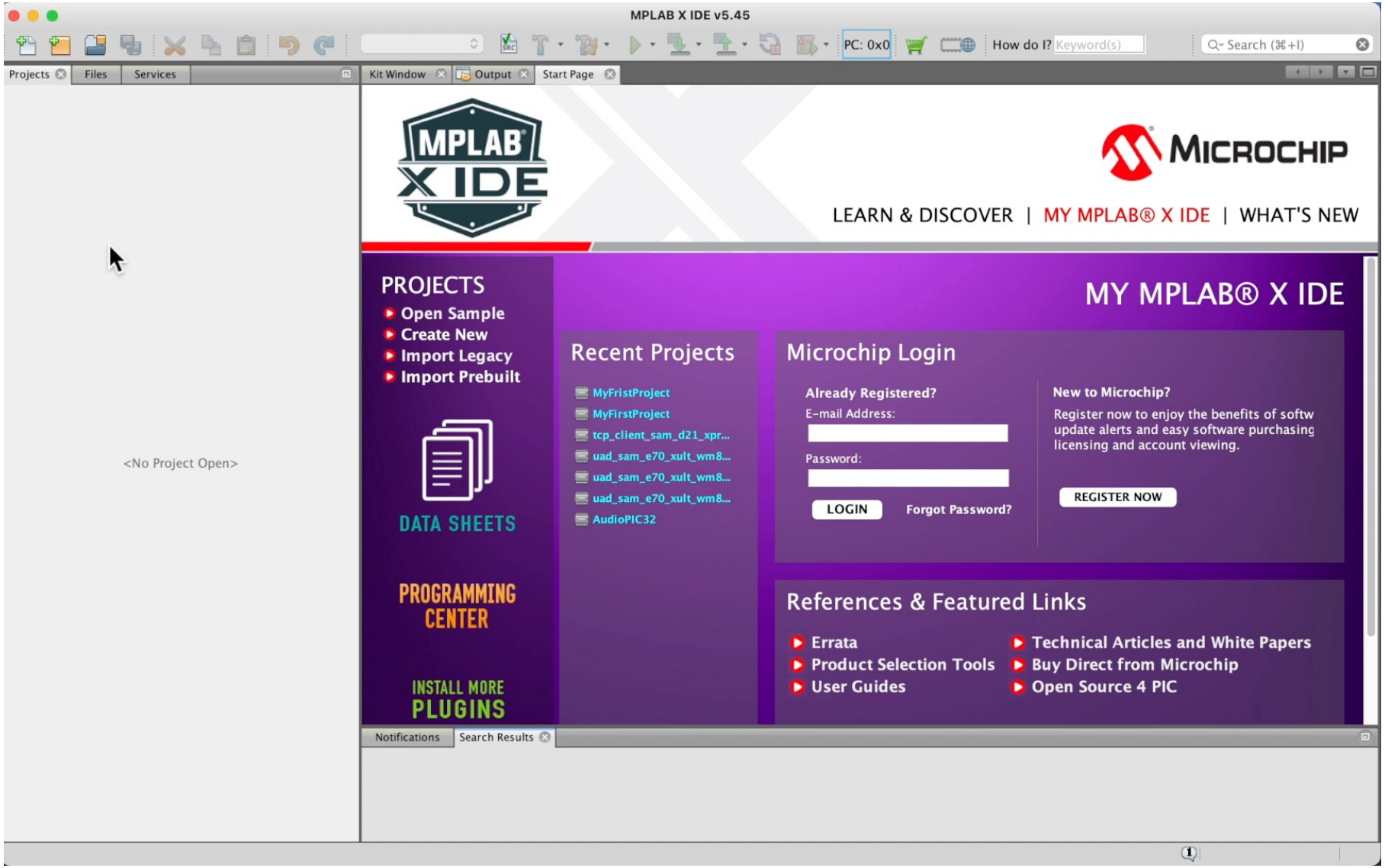
MICROCHIP

Develop a Secured Smart Home Application from scratch

- Make a simple Smart Home device based on WFI32E Curiosity Board connected to an MQTT Broker.
- The object will be monitored and controlled remotely thru a laptop or a mobile device. https://github.com/MicrochipTech/PIC32MZW1_Workshop/tree/master/06_develop



Develop a Secured Smart Home Application from scratch



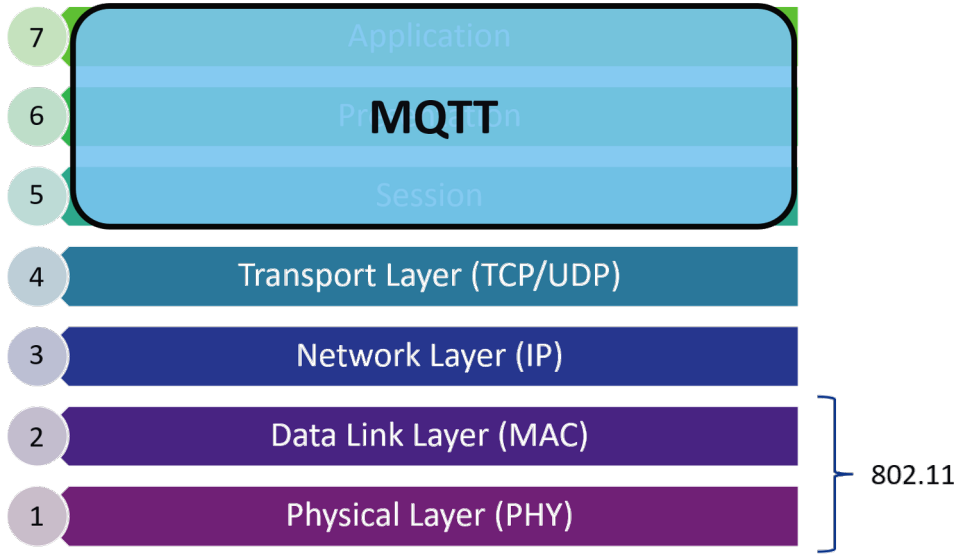
Develop a Secured Smart Home Application from scratch

MQTT Service

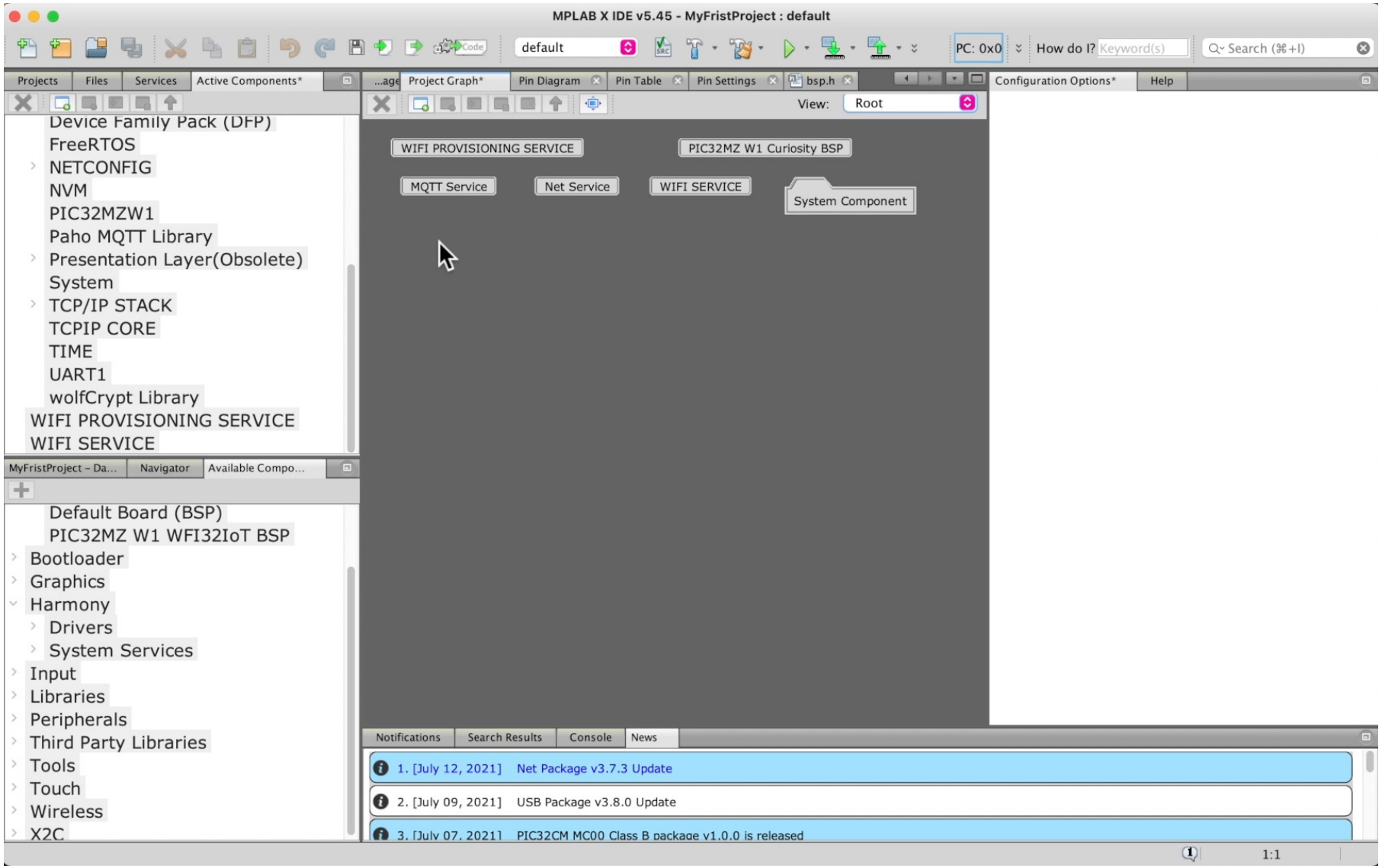
- **MQTT Service is an Harmony Networking layer which include components such as:** ***Net Service, Wi-Fi Service, NVM, Core Timer, UART1, Cryptographic, wolfCrypt Library, Time, Core, FreeRTOS, Console, Command, Debug, PIC32MZW1, Netconfig, TCPIP Core, BA414E, TCP/IP Application Layer Configuration, Netconfig, IPv4, ARP, UDP, ICMPv4, Wi-Fi Provisioning Service, Presentation Layer, Paho MQTT Library,...***

- **Notice that FreeRTOS component has been added and the Application is configured for FreeRTOS support.**

- **The MQTT protocol is based on TCP/IP. Both the client and the broker need to have a TCP/IP stack.**

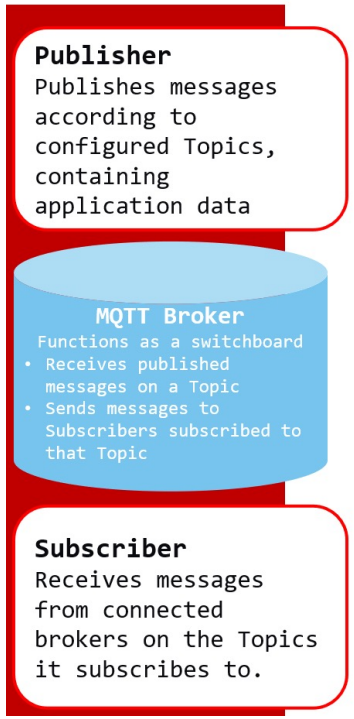


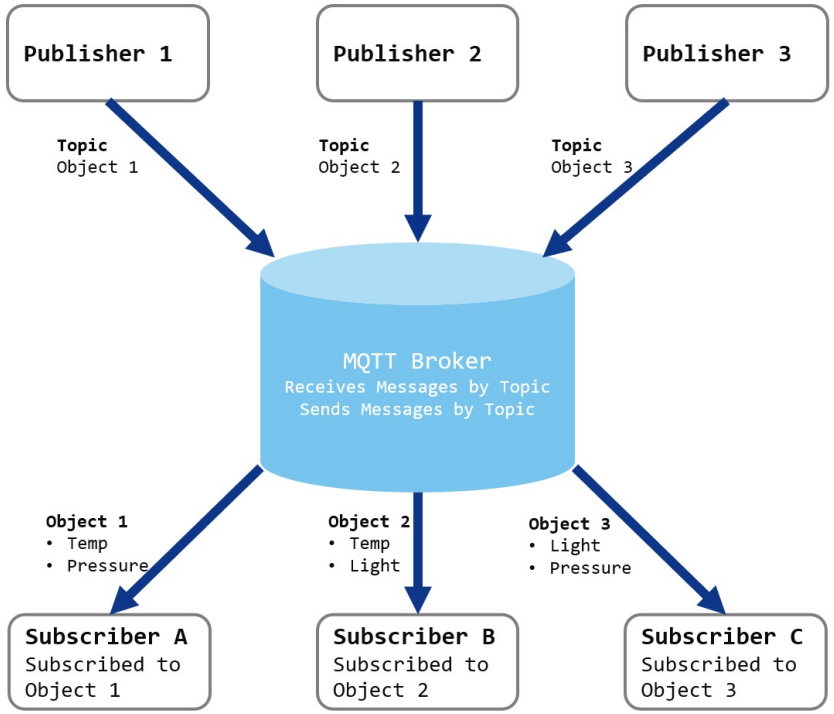
Develop a Secured Smart Home Application from scratch



Develop a Secured Smart Home Application from scratch

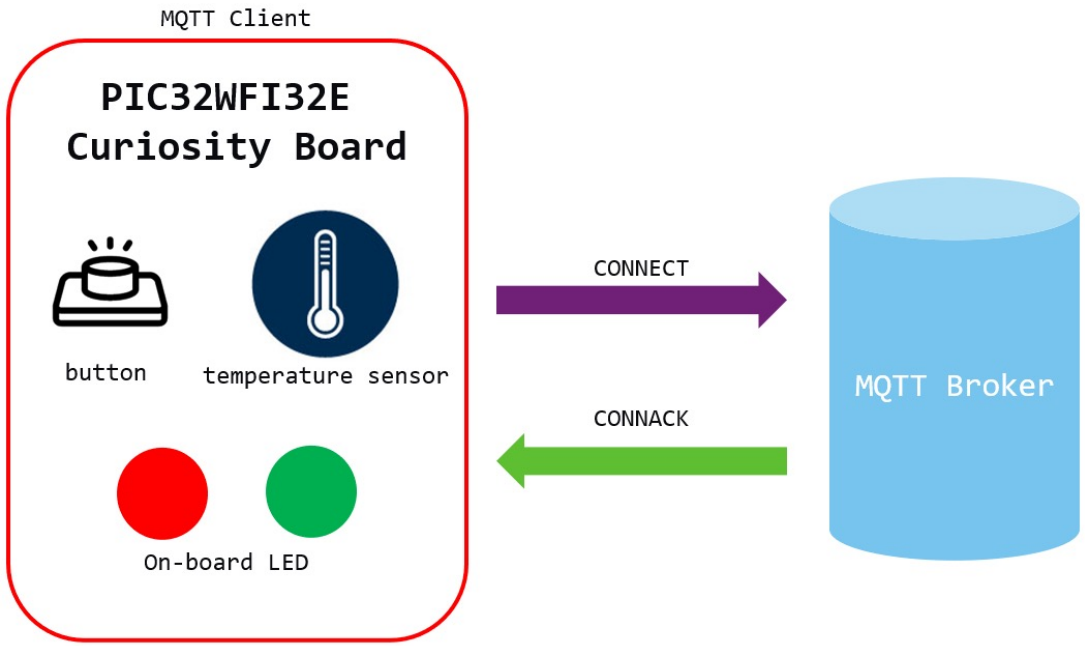
- To share information between MQTT clients over MQTT broker, you need to provide MQTT topics which is nothing more than a form of addressing. The MQTT connection is always between one client and the broker. Clients never connect to each other directly.





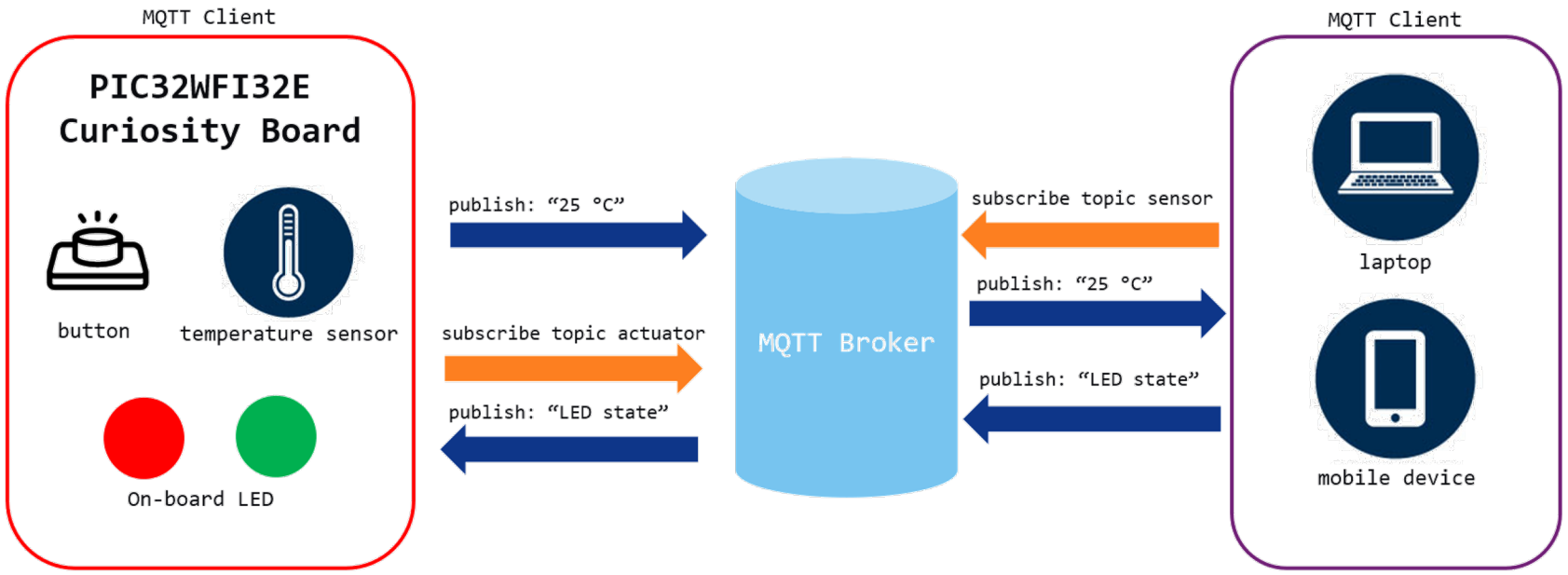
Develop a Secured Smart Home Application from scratch

- First, the client (PIC32MZ W1) sends a **CONNECT** message to the broker to initiate a connection
- The broker responds with a **CONNACK** message and a status code.
- Once the connection is established, the broker keeps it open until the client sends a disconnect command or the connection breaks.

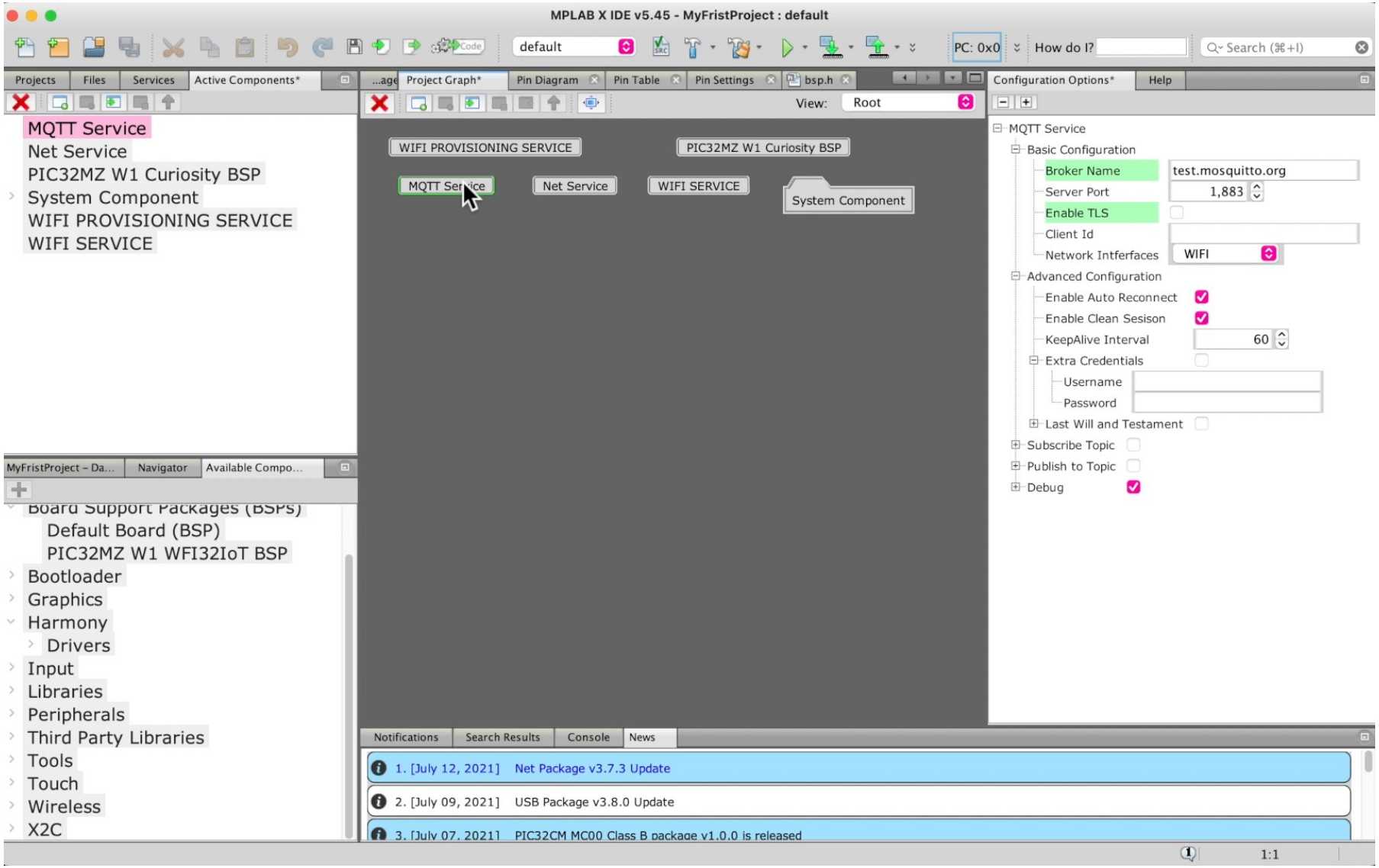


Develop a Secured Smart Home Application from scratch

- The client (PIC32MZ W1) should subscribe to a topic to receive the command messages that another client sends for changing the state of the actuator (PIC32MZW1's on-board LED).
- The client (PIC32MZ W1) should publish sensor data to Broker over another topic. Thus, another client will be able to monitor the data.



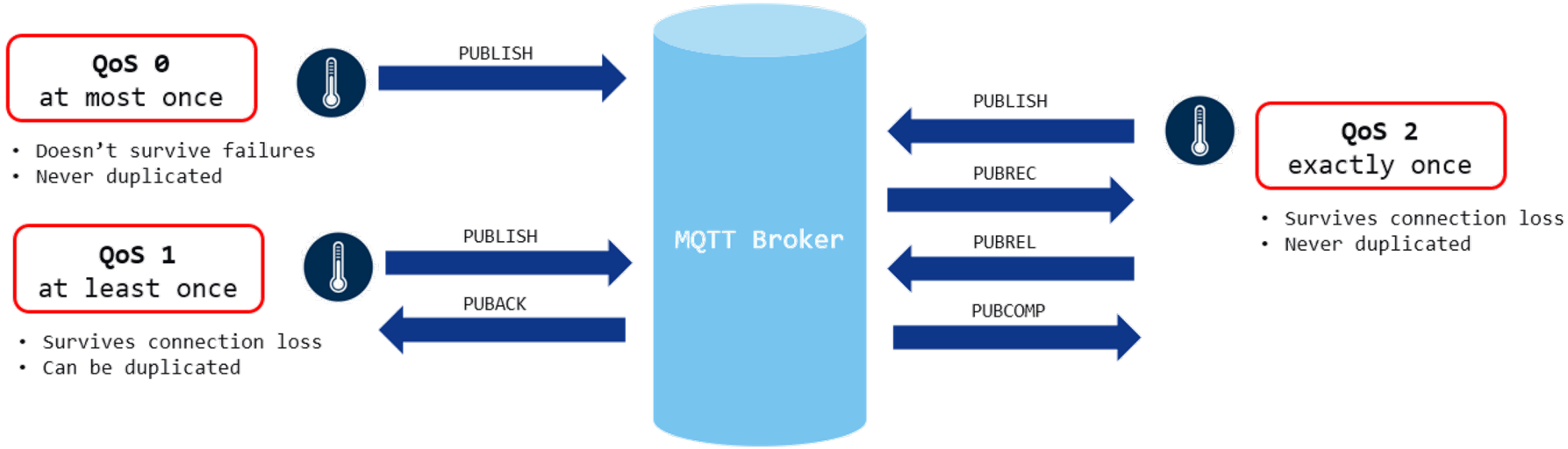
Develop a Secured Smart Home Application from scratch



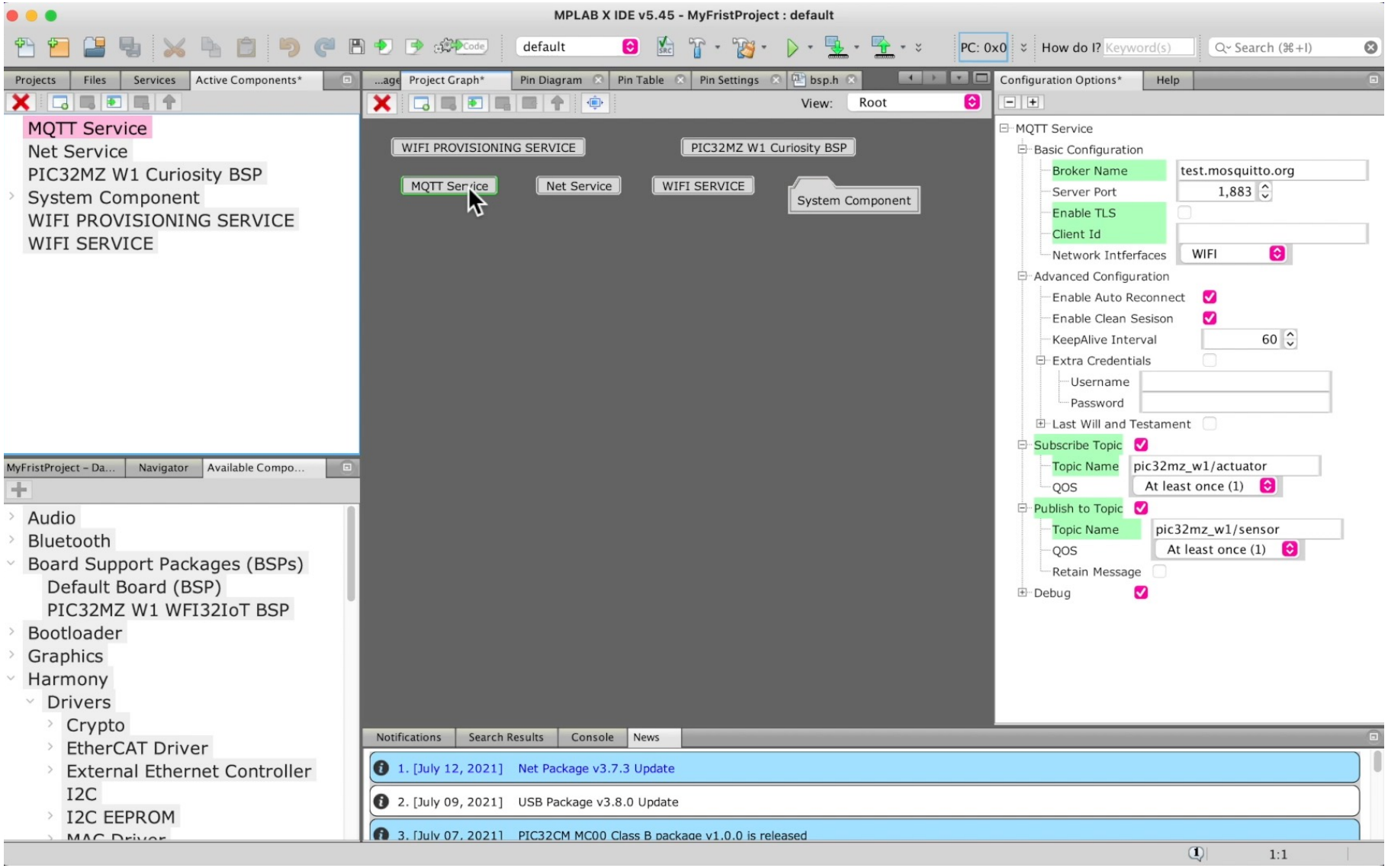
Develop a Secured Smart Home Application from scratch

QoS (Quality of Service) contains 3 levels for the messages emitted:

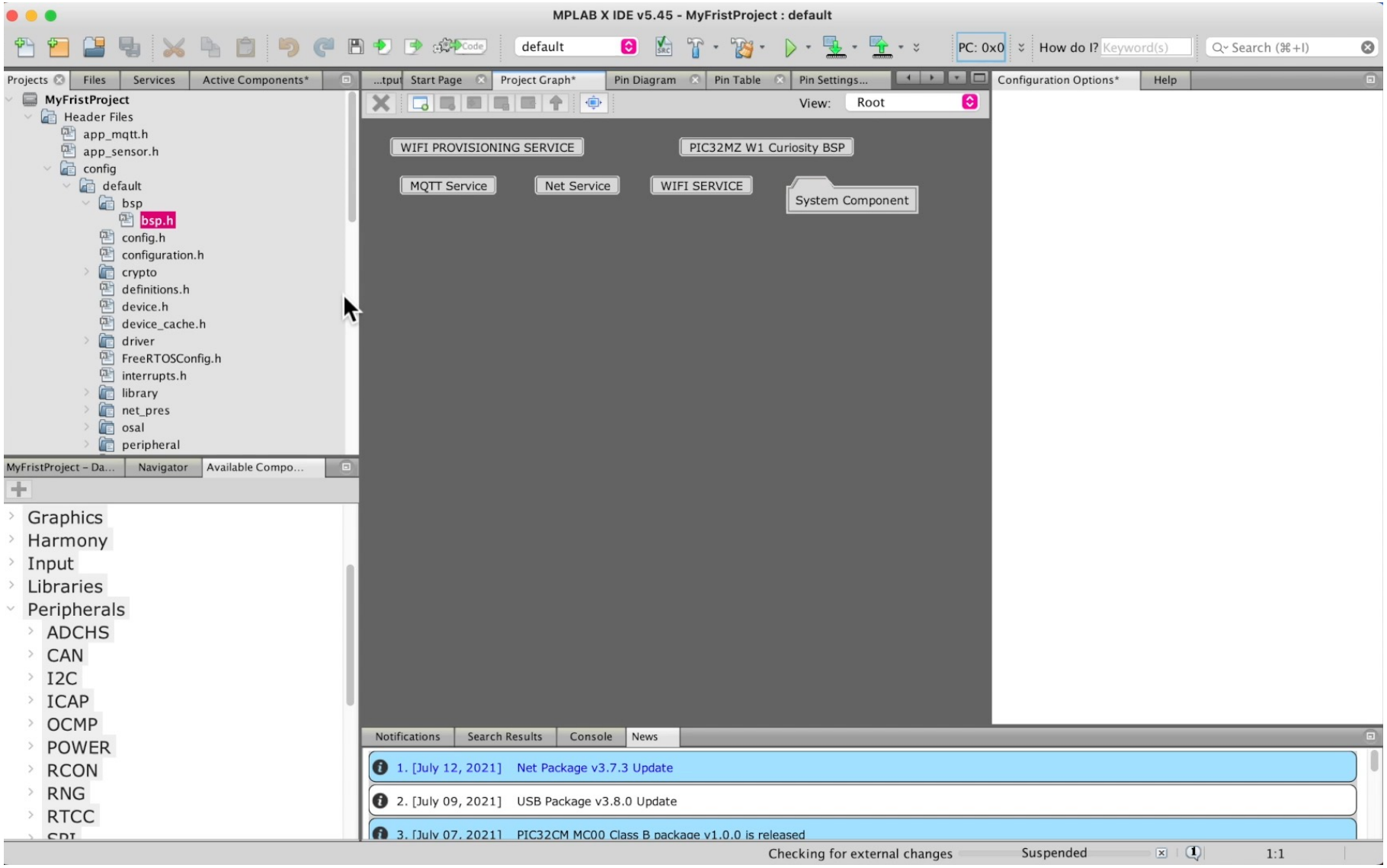
- ***QoS 0*** guarantees that the message is delivered ***at most once***, or it is not delivered at all. The sender does not care about the message after broadcasting (fire and forget).
- ***QoS 1*** guarantees that the message is always delivered ***at least once***. If the sender does not receive an acknowledgement, the message can be sent multiple times.
- ***QoS 2*** guarantees that the message is always delivered ***exactly once*** and no messages are lost. This is ensured by a two-step confirmation with additional messages.



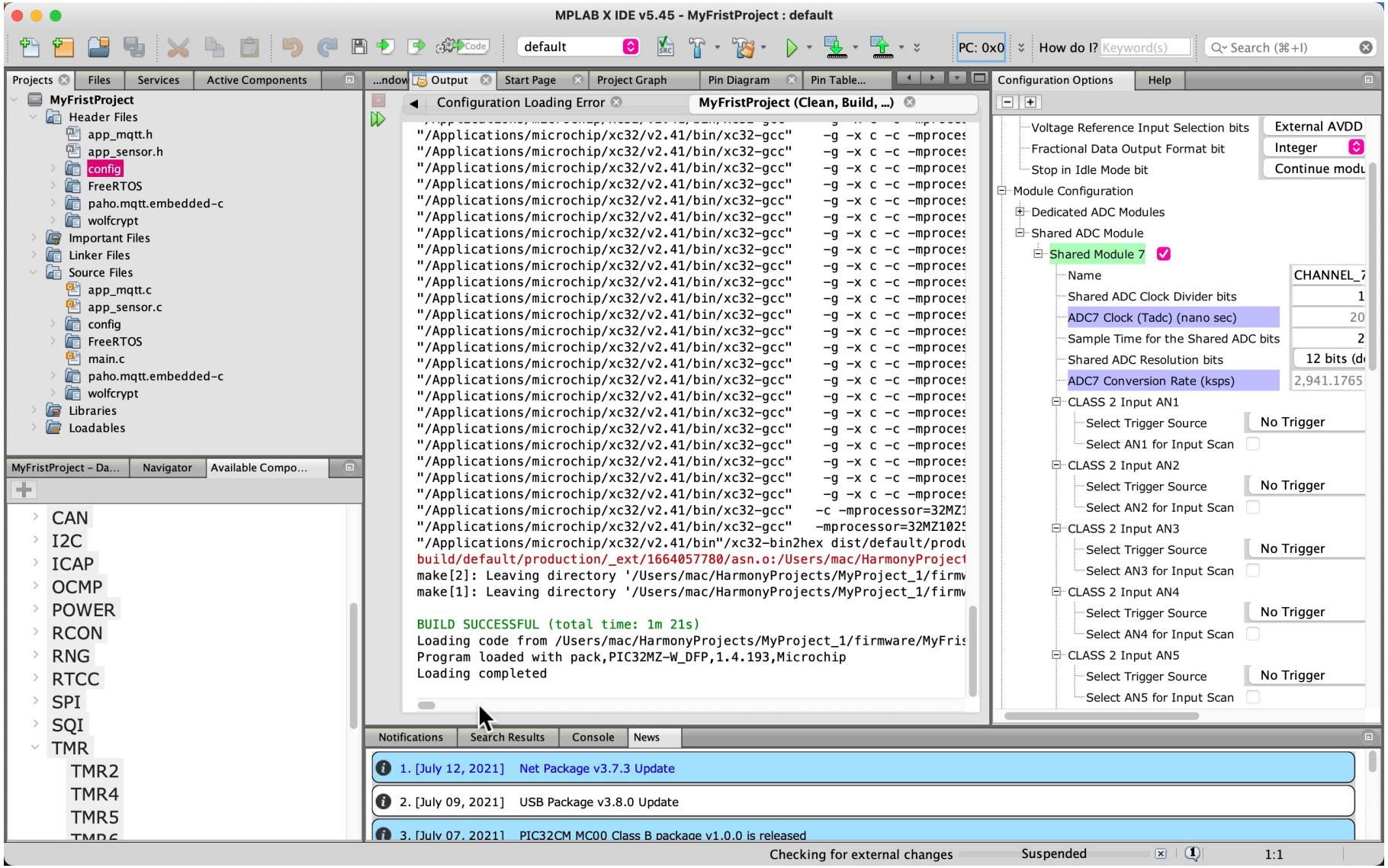
Develop a Secured Smart Home Application from scratch



Develop a Secured Smart Home Application from scratch

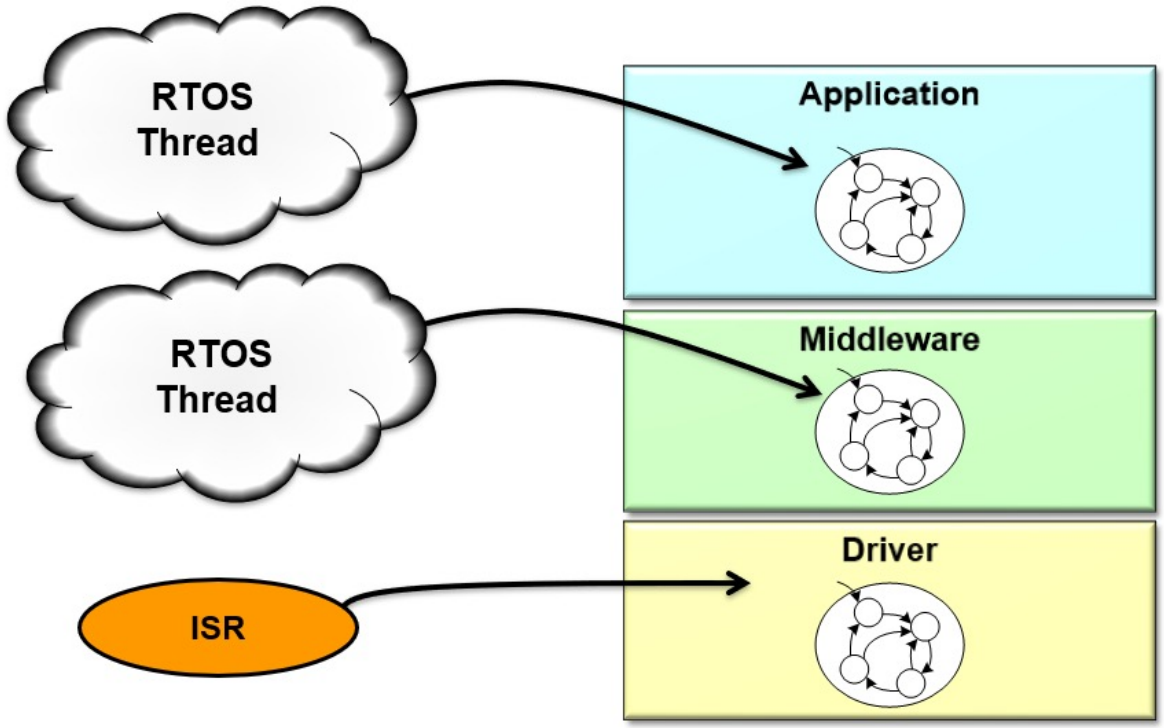


Develop a Secured Smart Home Application from scratch



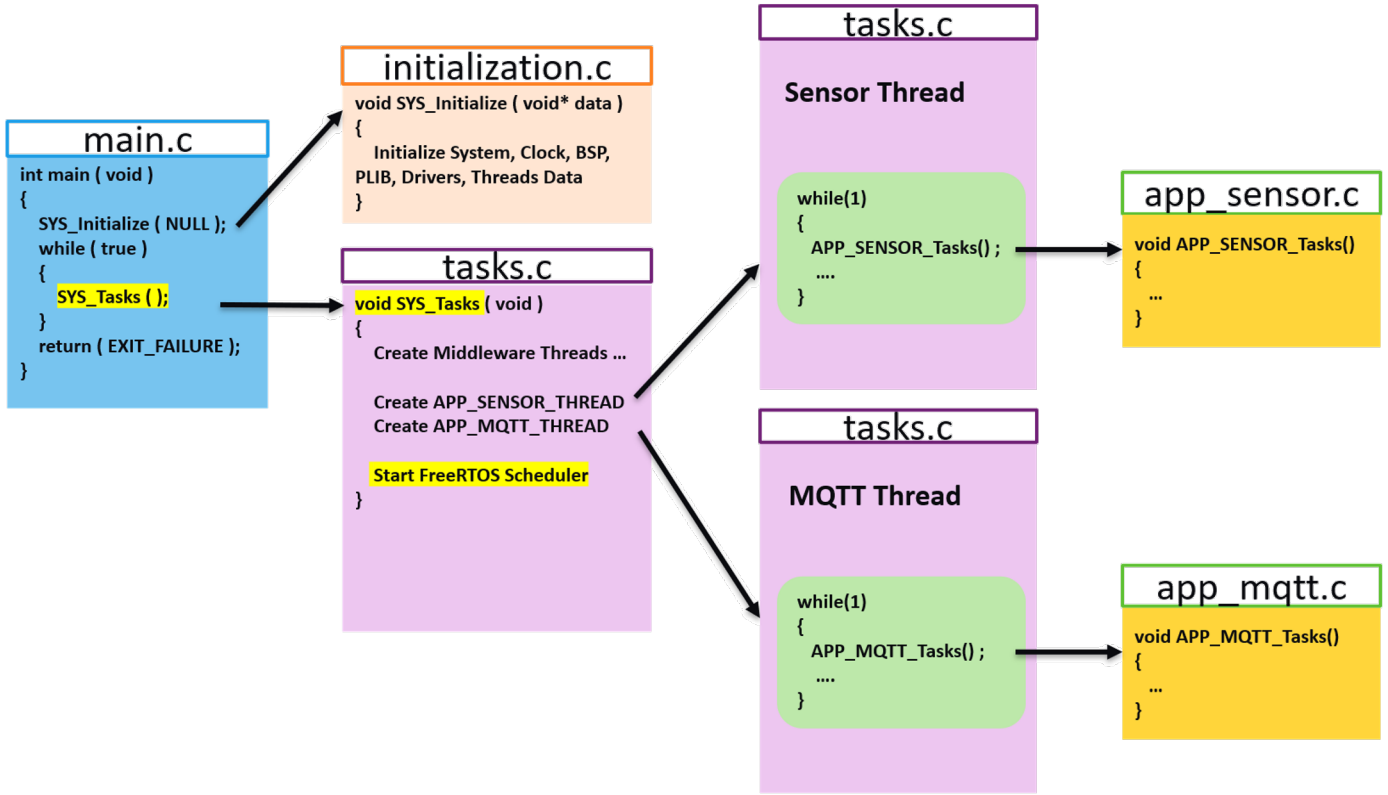
Develop a Secured Smart Home Application from scratch

- The project is using FreeRTOS library to create application threads & intercommunicate between application threads.
- Allows the individual task to be assigned a priority
- Interrupts support the best real-time response latency



Develop a Secured Smart Home Application from scratch

- The System task routine then calls the RTOS scheduler, thereafter all RTOS threads will be managed by the scheduler and the **SYS_Tasks** function will not return.
- MHC also generates the individual daemon threads that will run the MPLAB Harmony libraries and the application tasks (**_APP_SENSOR_Tasks** and **_APP_MQTT_Tasks**).



Develop a Secured Smart Home Application from scratch

- The **main.c** file calls the **SYS_Tasks()** routine, which create the sensor and MQTT threads and makes a call to FreeRTOS API **vTaskStartScheduler()** to start the scheduler.

tasks.c

```
void _APP_SENSOR_Tasks(  void *pvParameters  )
{
    while(1)
    {
        APP_SENSOR_Tasks();
        vTaskDelay(50 / portTICK_PERIOD_MS);
    }
}

void _APP_MQTT_Tasks(  void *pvParameters  )
{
    while(1)
    {
        APP_MQTT_Tasks();
        vTaskDelay(50 / portTICK_PERIOD_MS);
    }
}

void SYS_Tasks ( void )
{
    xTaskCreate((TaskFunction_t) _APP_SENSOR_Tasks, "APP_SENSOR_Tasks", 1024, NULL, 1, &xAPP_SENSOR_Tasks);
    xTaskCreate((TaskFunction_t) _APP_MQTT_Tasks,   "APP_MQTT_Tasks", 1024, NULL, 1, &xAPP_MQTT_Tasks);

    vTaskStartScheduler(); /* This function never returns. */
}
```

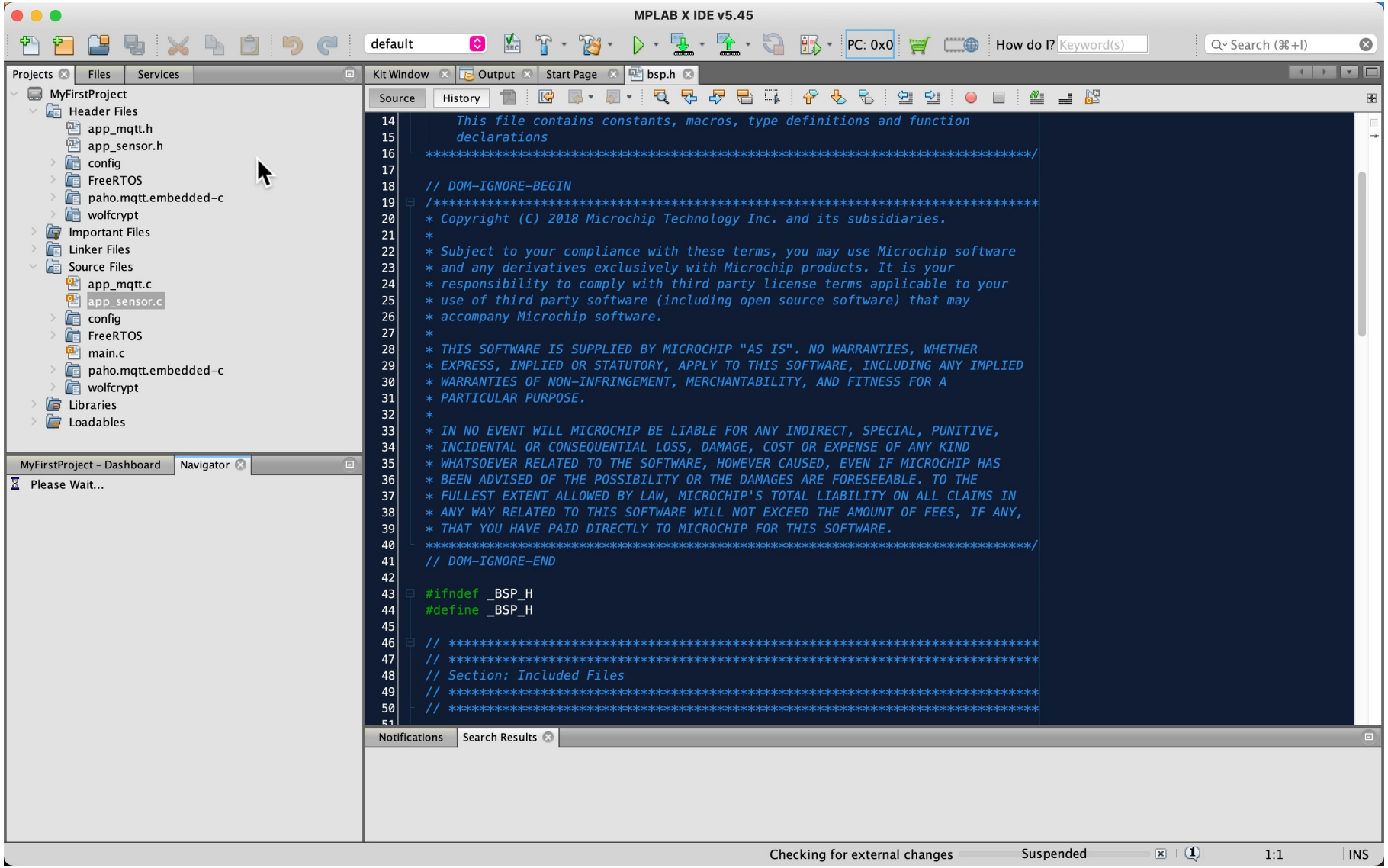
Thread Functions

FreeRTOS Specific

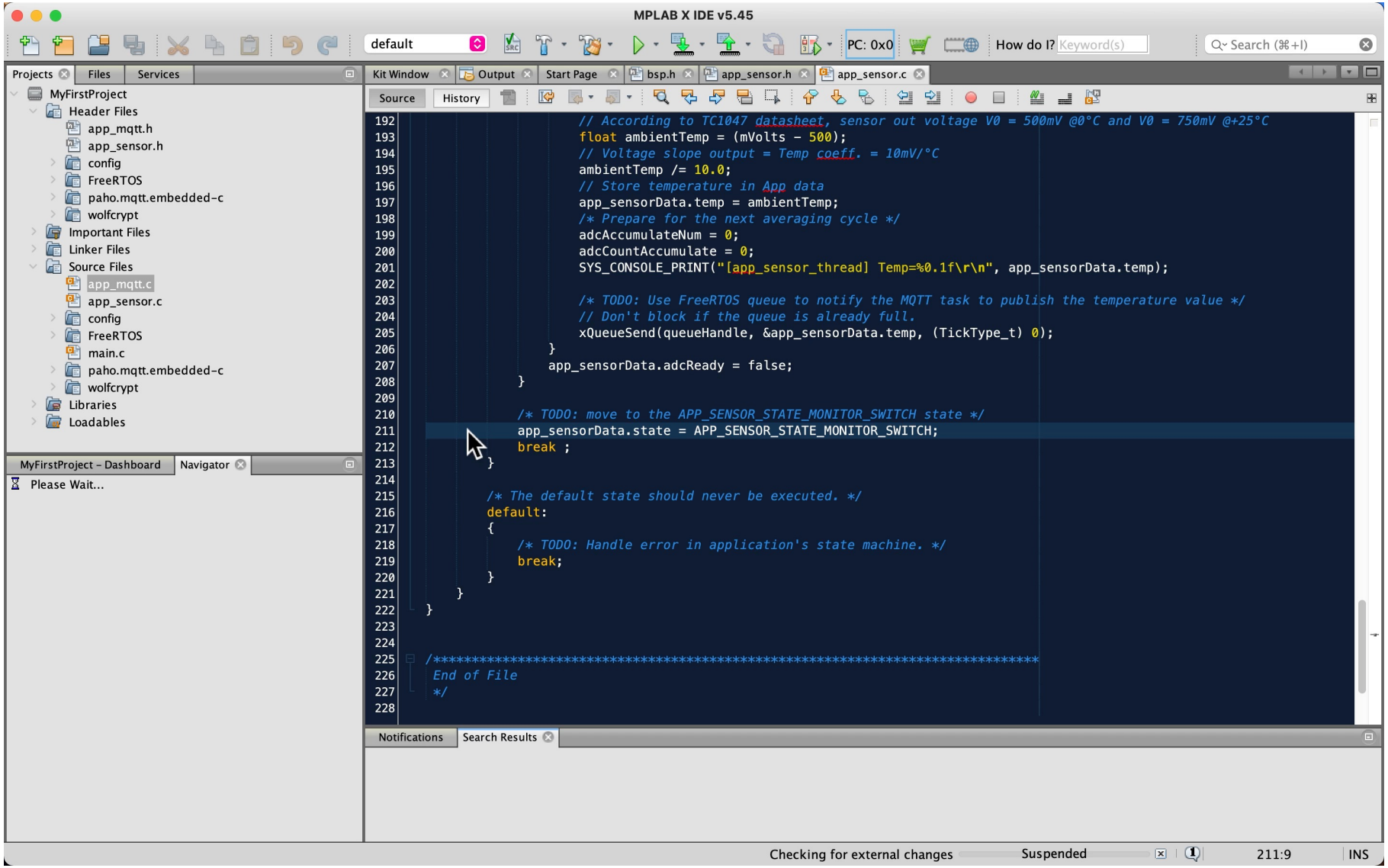
Develop a Secured Smart Home Application from scratch

- The MPLAB® Harmony components configured for MQTT and RTOS based application and you are ready to begin developing your application logic of two independent Application Tasks.
  - **_APP_SENSOR_Tasks**
  - **_APP_MQTT_Tasks**
    - pusblishing data to temperature topic
    - subscribing to actuator topic and control the on-board LEDs
  - The application is already developed and is available in the files:
    - app_sensor.c,
    - app_sensor.h,
    - app_mqtt.c,
    - app_mqtt.h

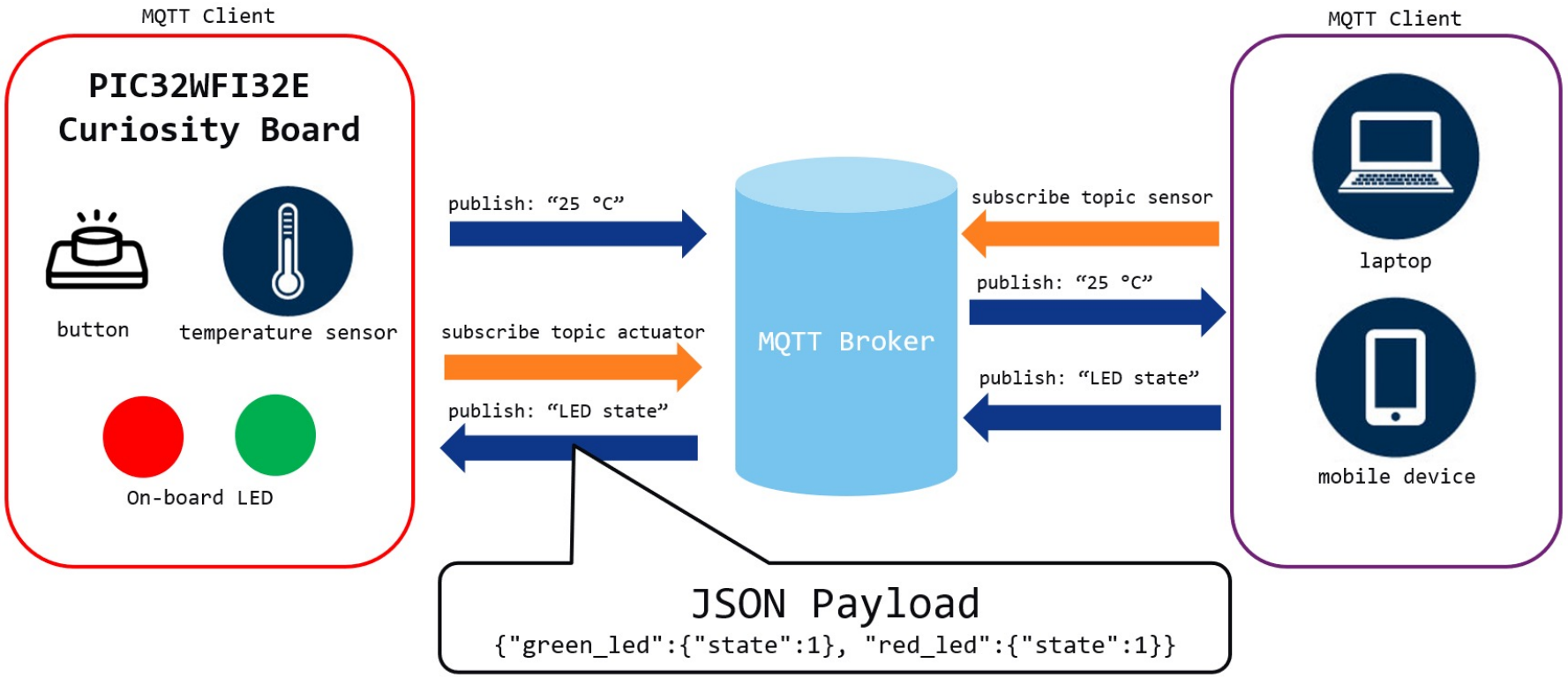
Develop a Secured Smart Home Application from scratch



Develop a Secured Smart Home Application from scratch

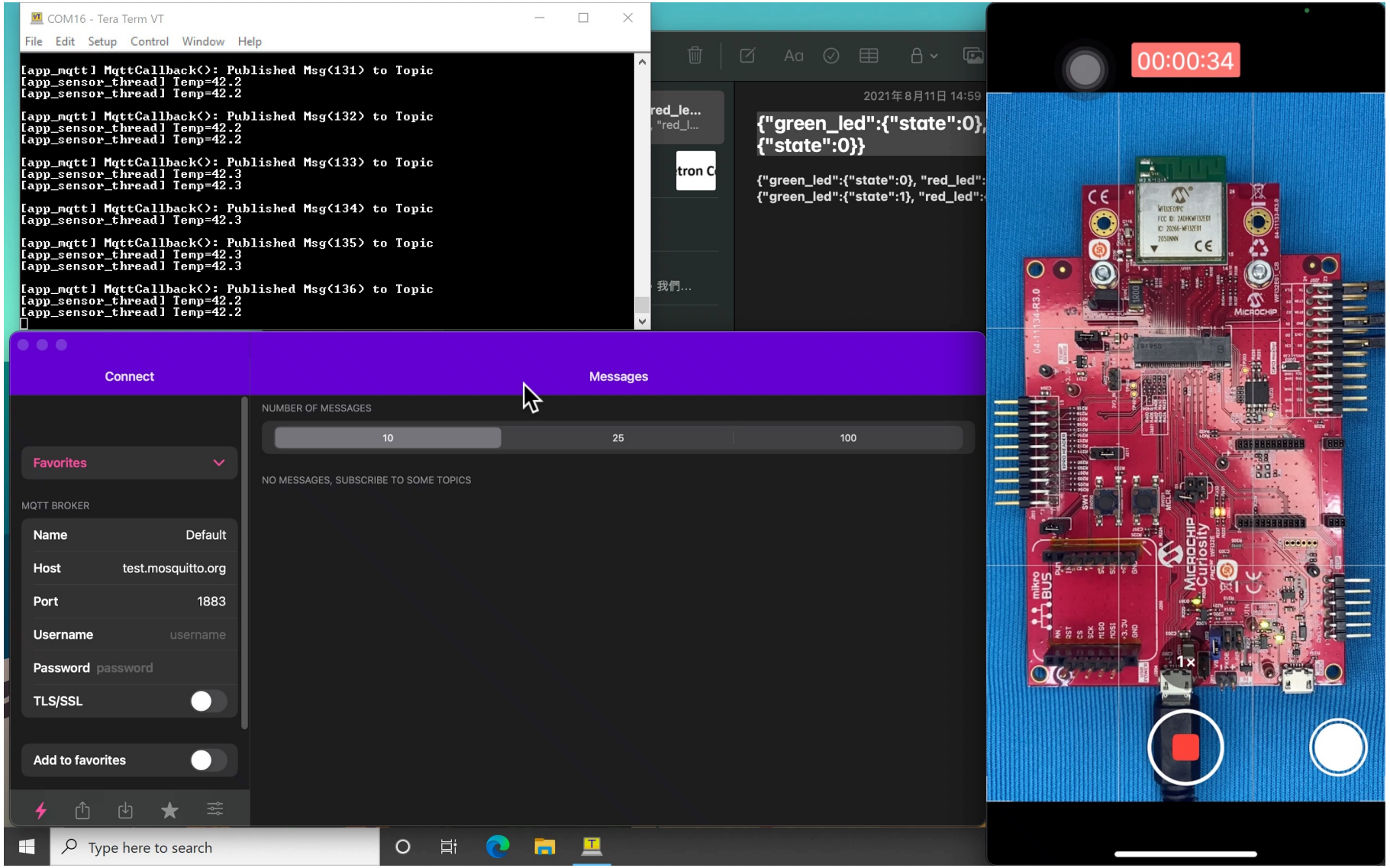


Develop a Secured Smart Home Application from scratch



Develop a Secured Smart Home Application from scratch

Demo



Agenda

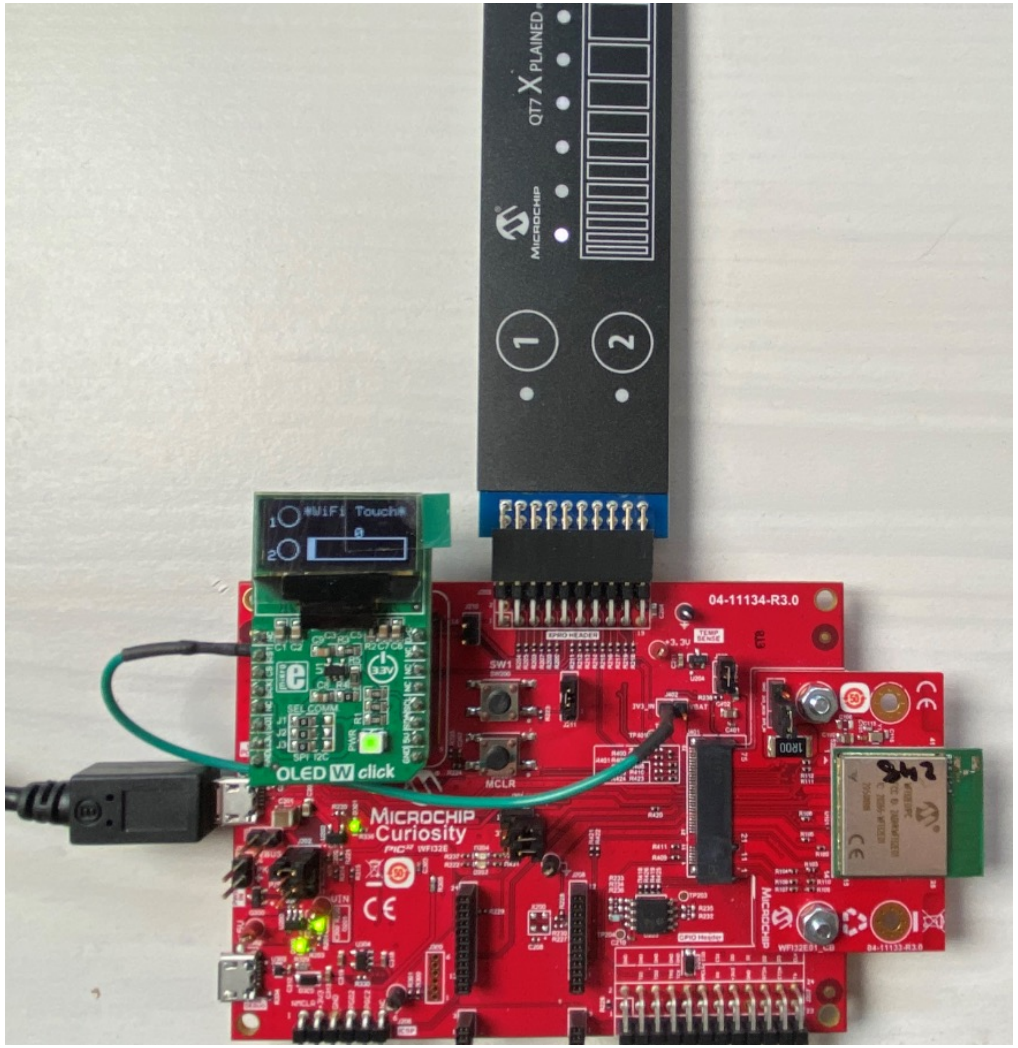
eLearning

- **Reference Example Codes**
  - Develop a Secured Smart Home Application from scratch
  - Wi-Fi ® Touch and OLED Display
  - Wi-Fi Provisioning over BLE
  - Wi-Fi to CAN Bus Bridge

MICROCHIP

Wi-Fi Touch and OLED Display

- Sample application showcasing Wi-Fi connectivity, Capacitive Touch and OLED Display control.
- The application acts as a TCP Server to which a TCP Client can connect and visualize QT7 Touch Xpro data.
- The Touch data are also printed on an OLED Display.

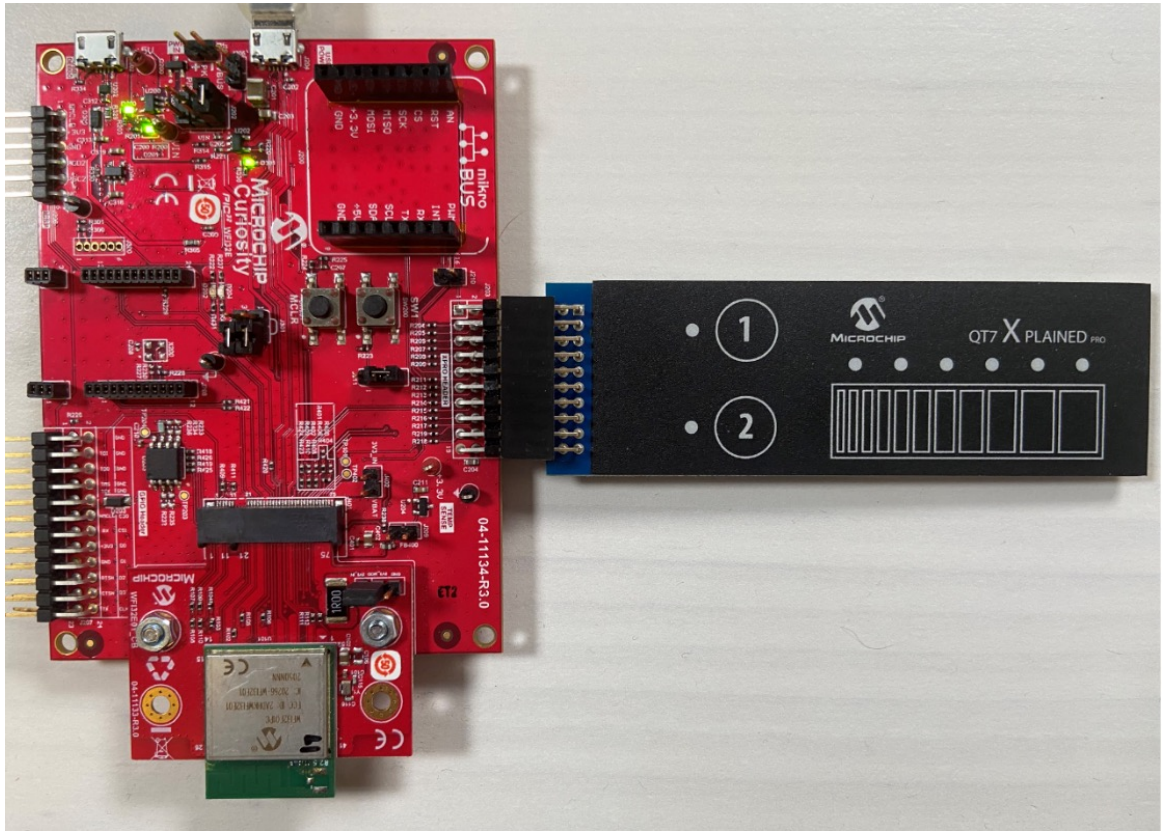


https://github.com/MicrochipTech/PIC32MZW1_Workshop/tree/master/07_projects#wi-fi-touch-and-oled-display

Wi-Fi Touch and OLED Display

Touch example with CVD and QT7 Xpro

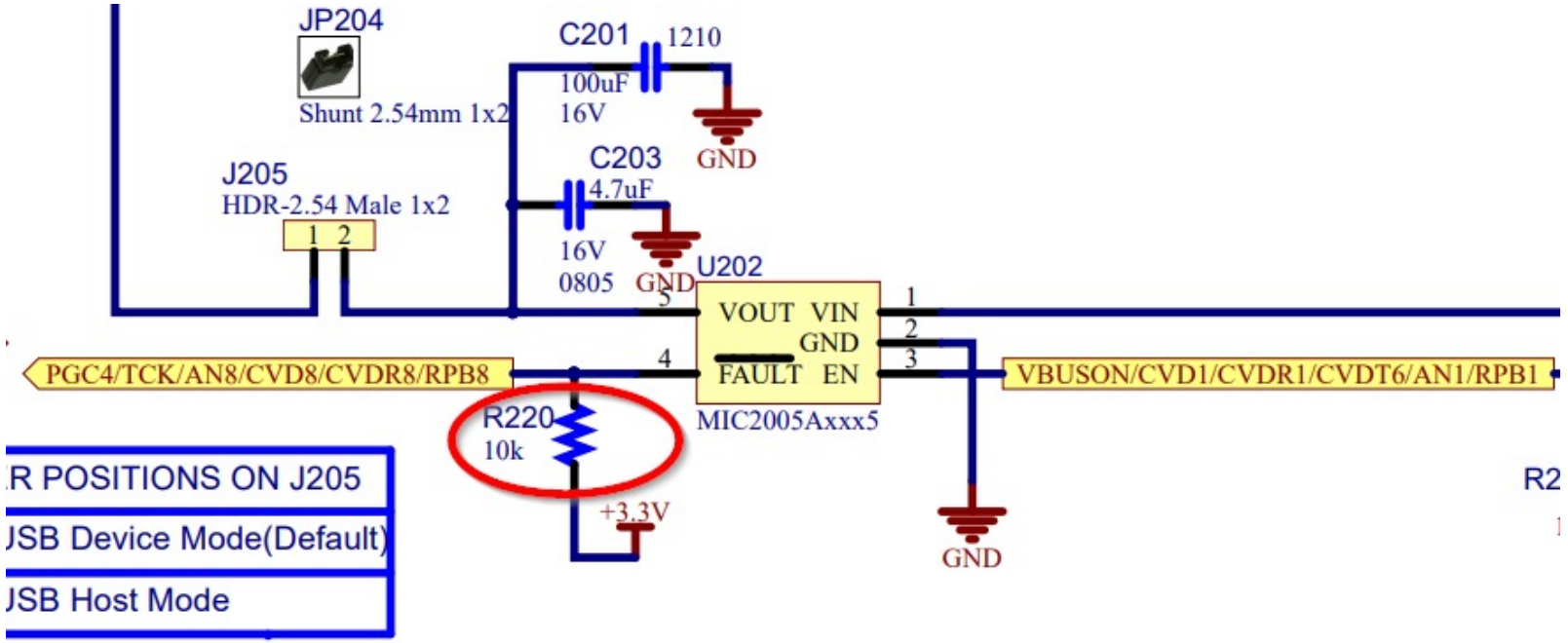
- Basic CVD provides a touch interface based on self capacitance touch sensing.
- The ADC module embedded in the PIC32MZ-W1 device supports CVD feature by using the shared ADC core to perform a modified scan of all 2nd and 3rd class channels.



Wi-Fi Touch and OLED Display

Touch example with CVD and QT7 Xpro

- One of the slider sensor line (Y-line2) is connected to U202 and the line is loaded with 10k (R220) resistor. You could remove R220 to not interfere the measurement of the slider sensor.



Wi-Fi Touch and OLED Display

Touch example with CVD and QT7 Xpro

- QT7 uses one slider & 2 touch buttons.

QT7 Xpro Header Pin	Function	Description	WFI32E01 Module Pin
1	ID	-	-
2	GND	-	-
3	Y-LINE-5	Driven shield	RB6 (X1 in MHC)
4	Y-LINE-1	Button 1 Sensor	RA14 (button 0/Y14 in MHC)
5	LED0	LED for Slider	RB12 (LED_SLIDER_6 in MHC)
6	LED6	LED for Button 1	RK6 (LED_BUTTON_1 in MHC)
7	Y-LINE-2	Slider Sensor	RB8 (slider 0/channel 4/Y8 in MHC)
8	Y-LINE-3	Slider Sensor	RA13 (slider 0/channel 3/Y15 in MHC), shared with Temp sensor)
9	Y-LINE-4	Slider Sensor	RA10 (slider 0/channel 2/Y17 in MHC)
10	Y-LINE-0	Button 2 Sensor	RB2 (button 1/Y2 in MHC)

Wi-Fi Touch and OLED Display

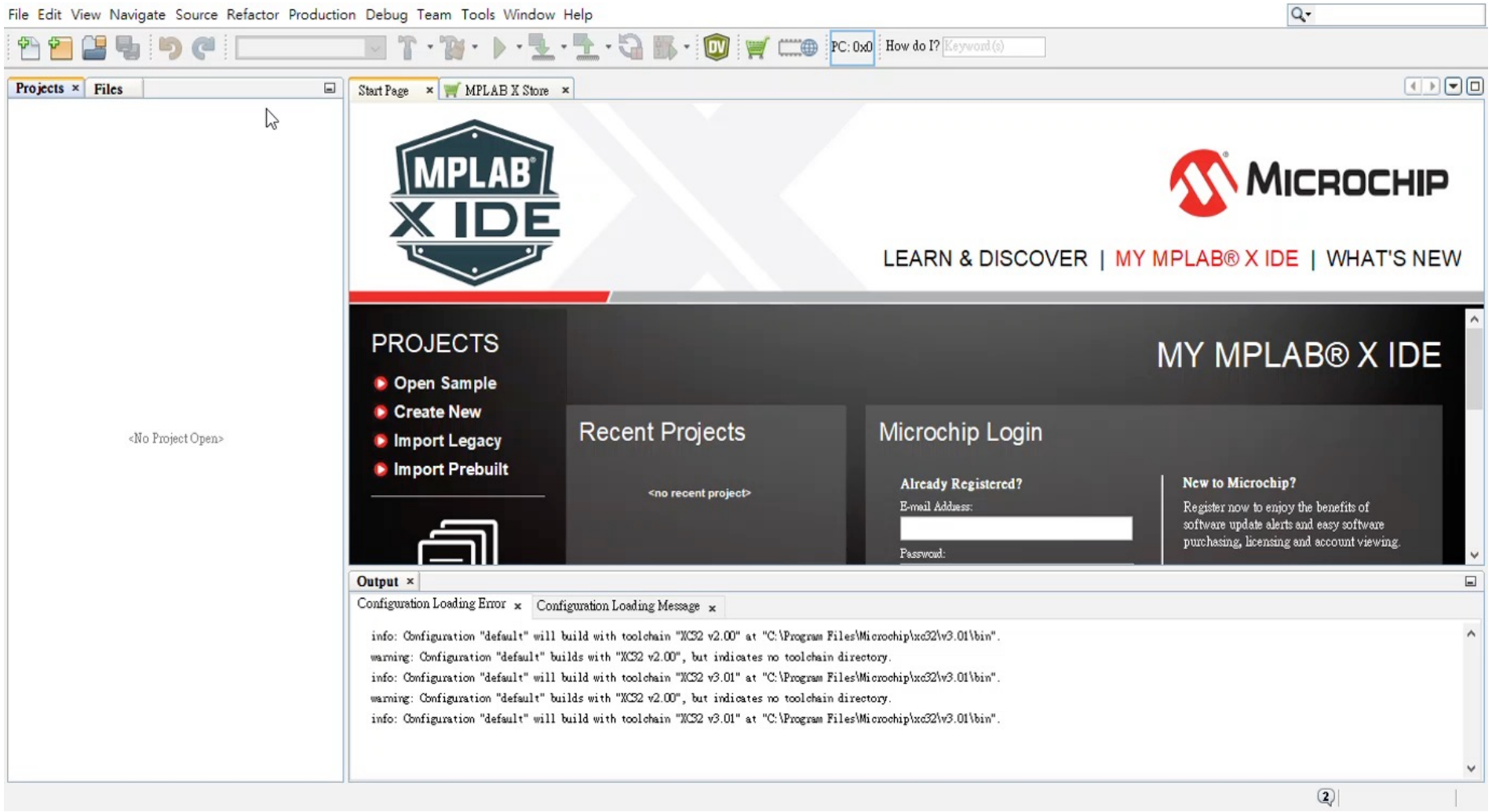
Touch example with CVD and QT7 Xpro

- QT7 uses one slider & 2 touch buttons.

QT7 Xpro Header Pin	Function	Description	WFI32E01 Module Pin
11	LED7	LED for Button 2	RA5 (LED_BUTTON_2)
12	LED1	LED for Slider	RA4 (LED_SLIDER_5)
13	NC	-	-
14	NC	-	-
15	LED2	LED for Slider	RB7 (LED_SLIDER_4)
16	LED3	LED for Slider	RK5 (LED_SLIDER_2)
17	LED4	LED for Slider	RK4 (LED_SLIDER_3)
18	LED5	LED for Slider	RA11 (LED_SLIDER_1)
19	GND	-	-
20	VCC	-	-

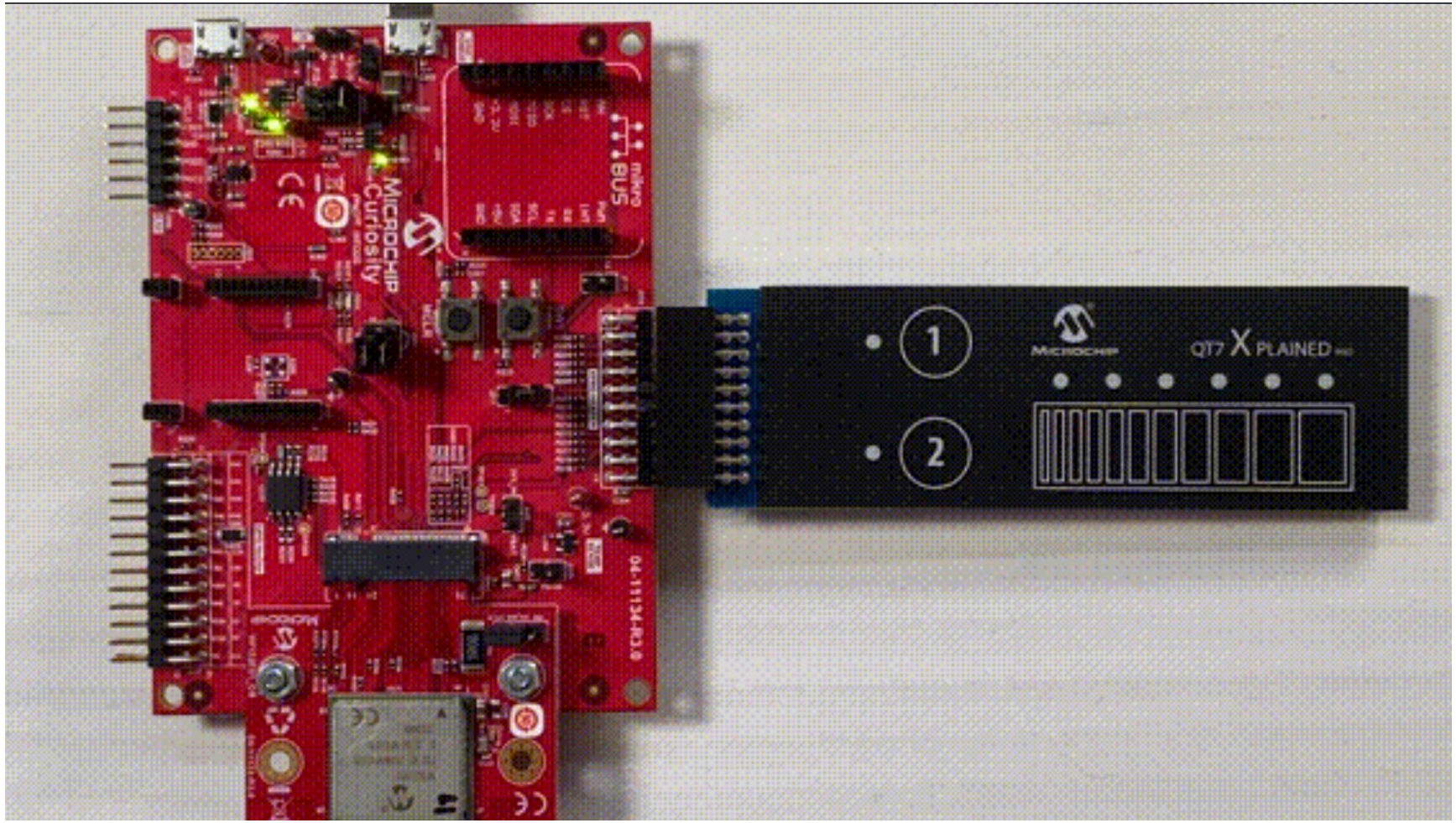
Wi-Fi Touch and OLED Display

Touch example with CVD and QT7 Xpro



Wi-Fi Touch and OLED Display

Touch example with CVD and QT7 Xpro

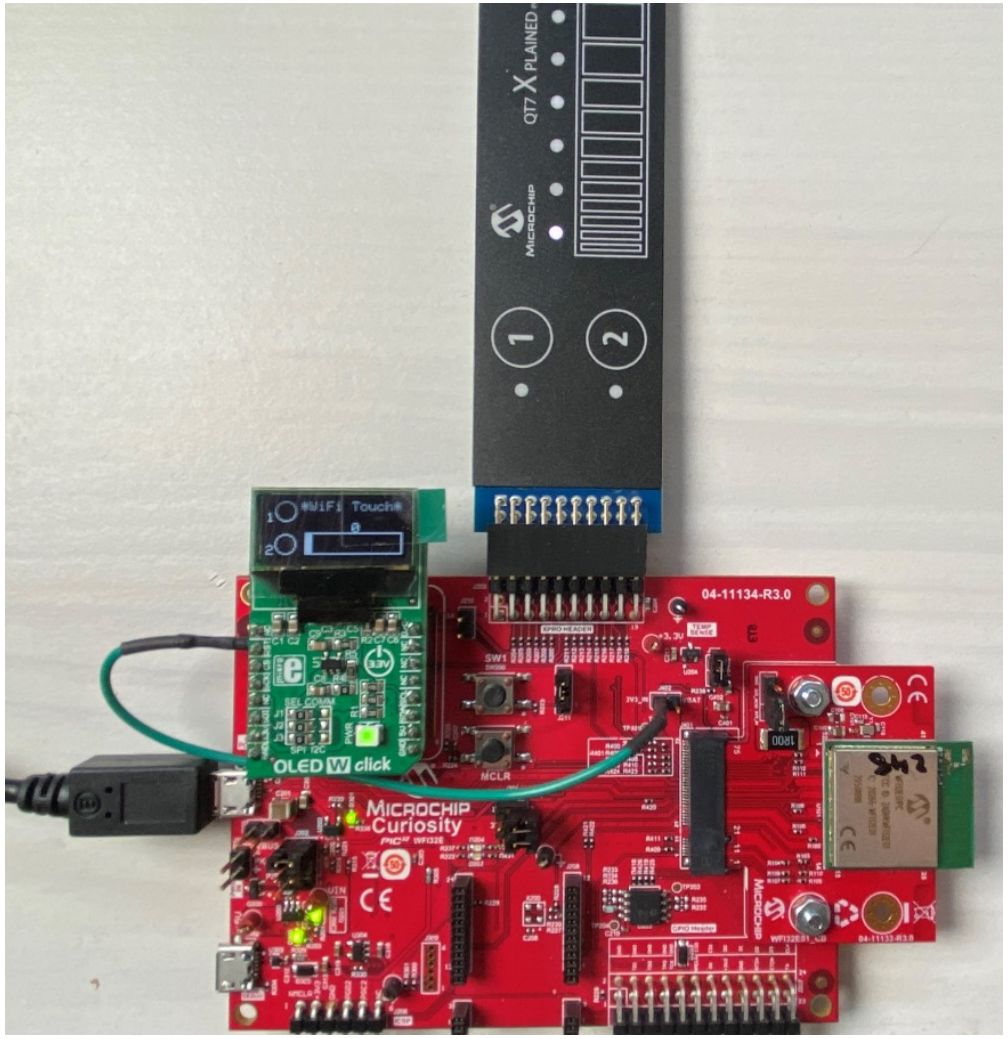


MICROCHIP

Wi-Fi Touch and OLED Display

Touch example with QT7 Xpro & OLED W Click

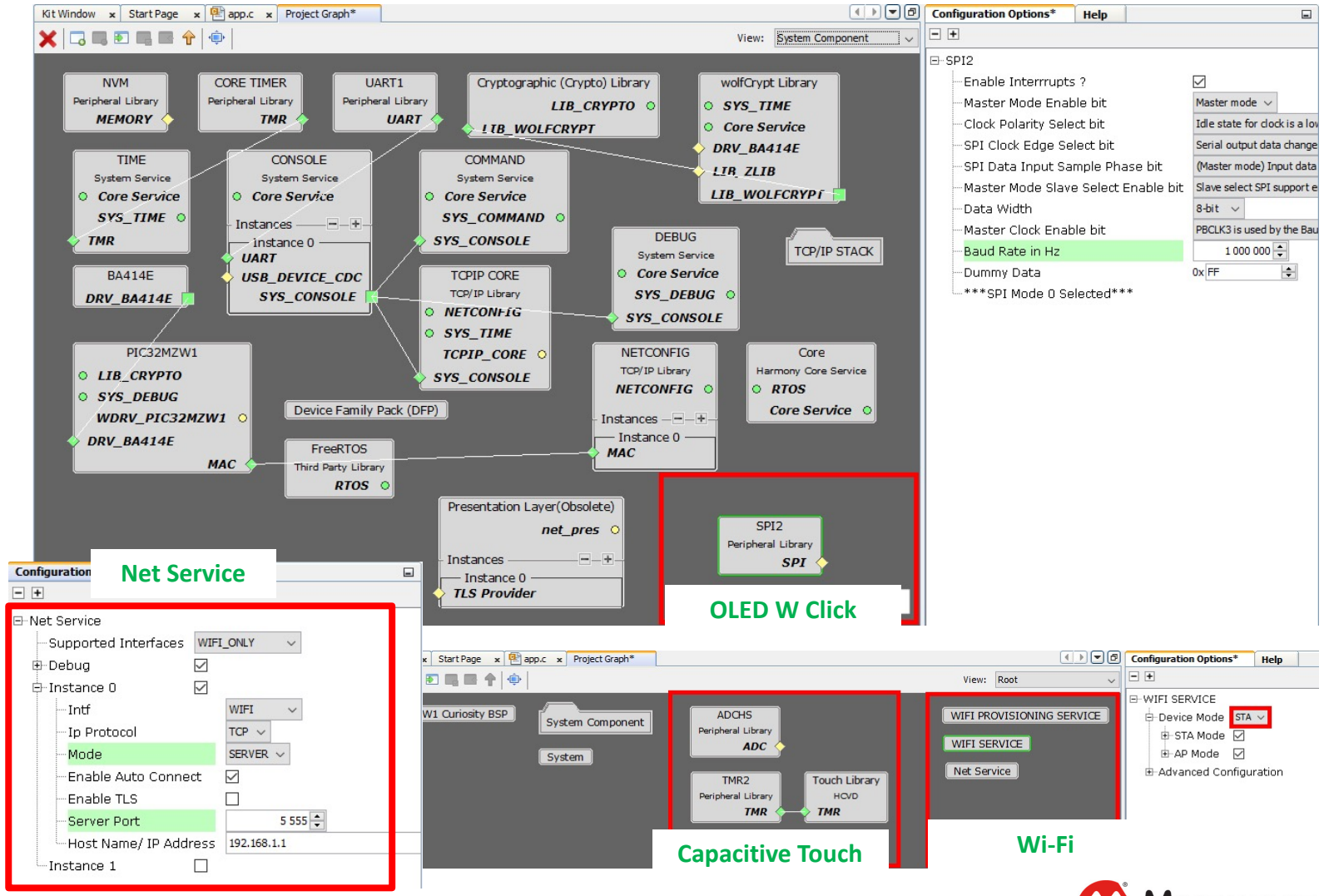
- The OLED W click carries a 96 x 39px white monochrome passive matrix OLED display. To drive the display, OLED W click features an SSD1306 controller.
- In this demo, the OLED W click communicates with the WFI32 MCU through SPI lines.



Wi-Fi Touch and OLED Display

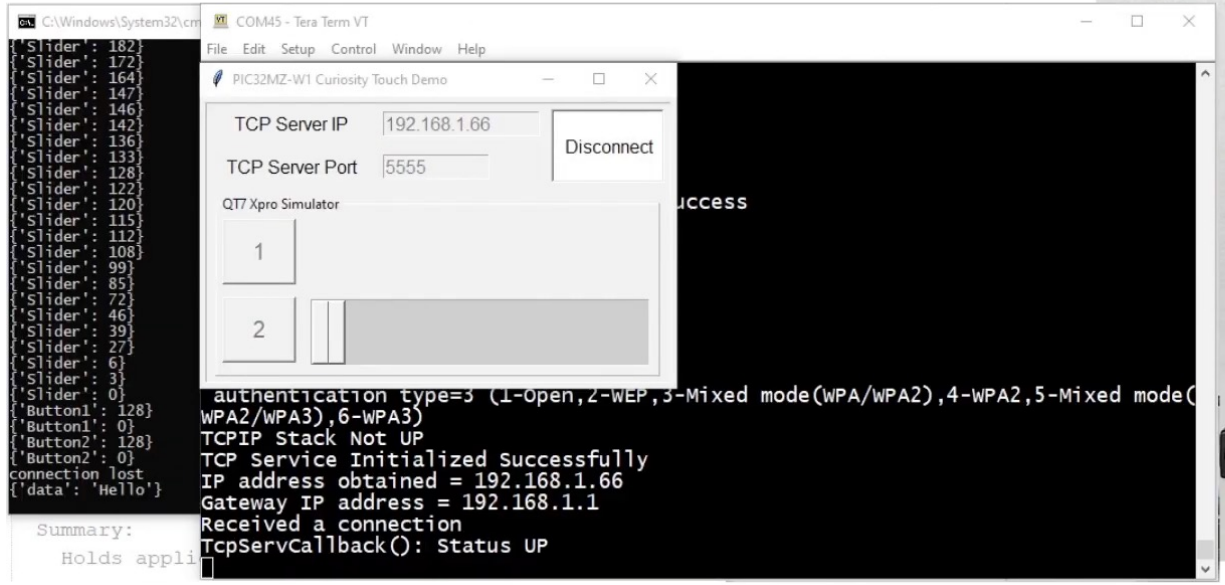
Touch example with QT7 Xpro & OLED W Click

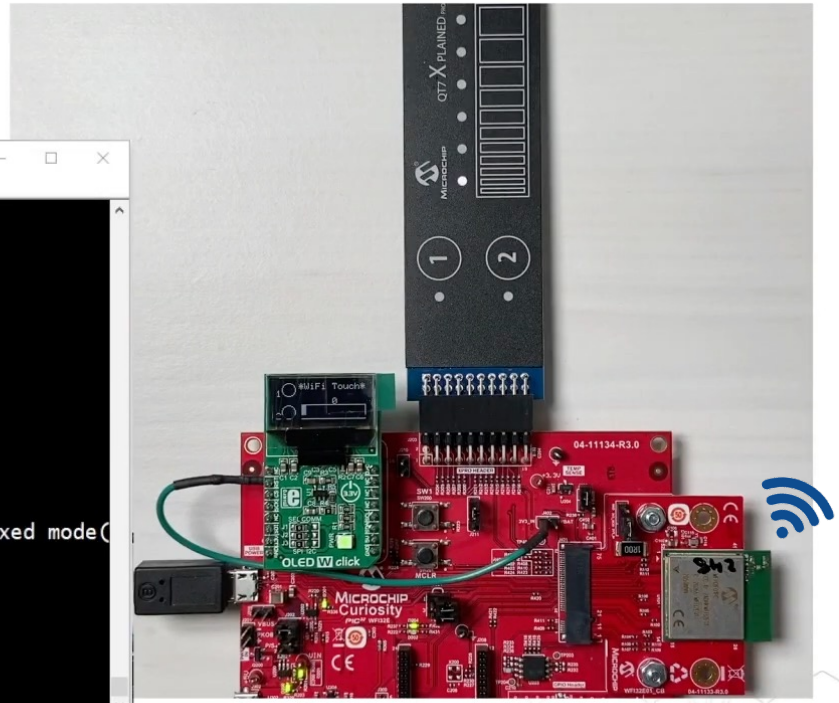
- SPI2 for OLED W click interface
- Touch Library using ADCHS and TMR2 components
- Wi-Fi Service, Wi-Fi provisioning and Net services



Wi-Fi Touch and OLED Display

Touch example with QT7 Xpro & OLED W Click





Agenda

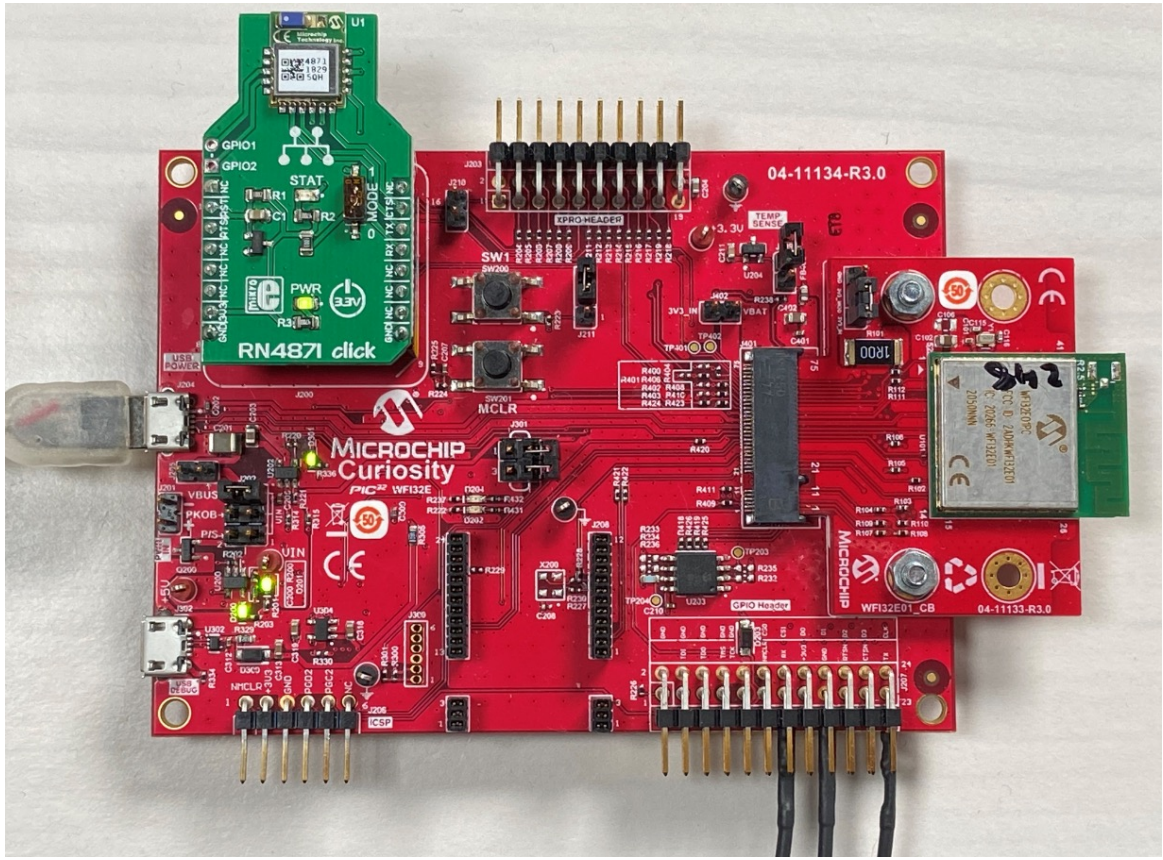
eLearning

- **Reference Example Codes**
  - Develop a Secured Smart Home Application from scratch
  - Wi-Fi ® Touch and OLED Display
  - Wi-Fi Provisioning over BLE
  - Wi-Fi to CAN Bus Bridge

MICROCHIP

Wi-Fi Provisioning over BLE

- Attach external BLE device to WFI32E Curiosity board and enable Wi-Fi communication and configuration over BLE.
- Watch the video and see how to enable Wi-Fi provisioning over BLE with WFI32E Curiosity board



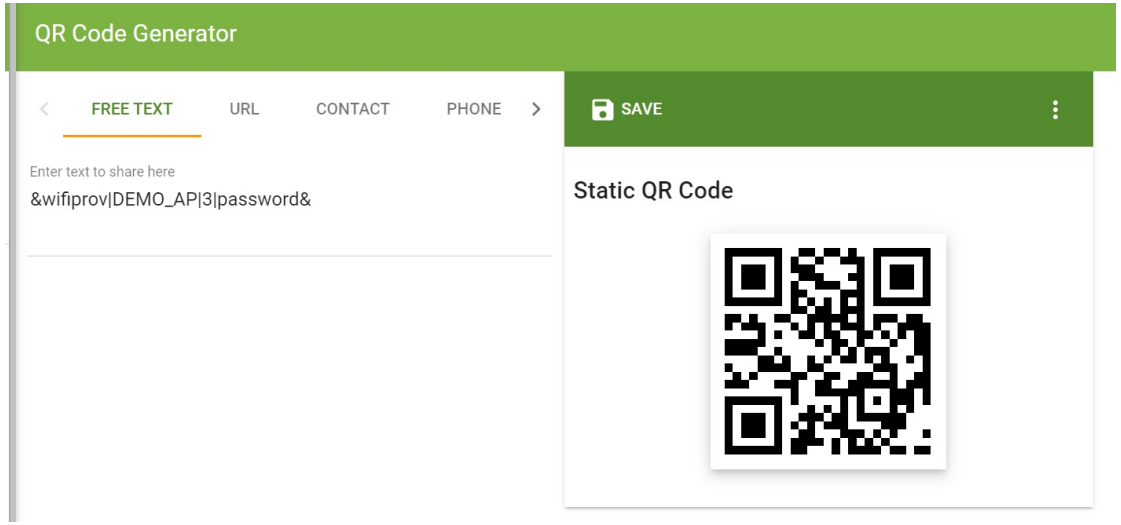
https://github.com/MicrochipTech/PIC32MZW1_Workshop/tree/master/07_projects#wi-fi-provisioning-over-ble

Wi-Fi Provisioning over BLE

Generate QR code for Wi-Fi provisioning

- A QR code is used in the demo to provision the Wi-Fi configuration over BLE.
- **Frame Format: &wifiprov|<ssid>|<authtype>|<password>&**
  - Where **&** is used to indicate the start and the end of the frame.
  - **wifiprov** is required and used as a command keyword.
  - **|** is required and used as a separator.
  - **<ssid>** is the name of the router / network.
  - Auth type represents the security type:
    - OPEN mode; WPAWPA2 (Mixed) mode; WPA2 mode
    - WPA2WPA3 (Mixed) mode; WPA3 mode
  - password is not required in Open mode
- **e.g.: &wifiprov|DEMO_AP|3|password&**
- Create your own QR code from:

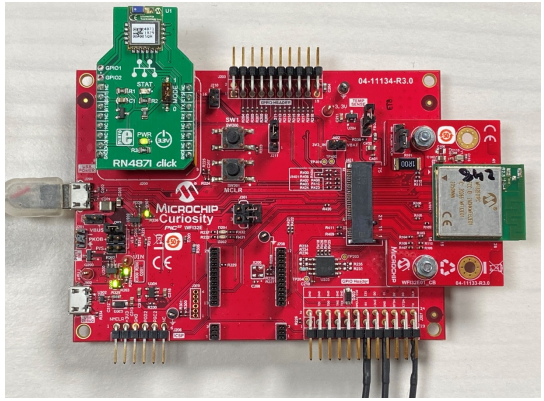
  https://www.the-qrcode-generator.com/



MICROCHIP

Wi-Fi Provisioning over BLE

Download from PIC32MZW1_Workshop

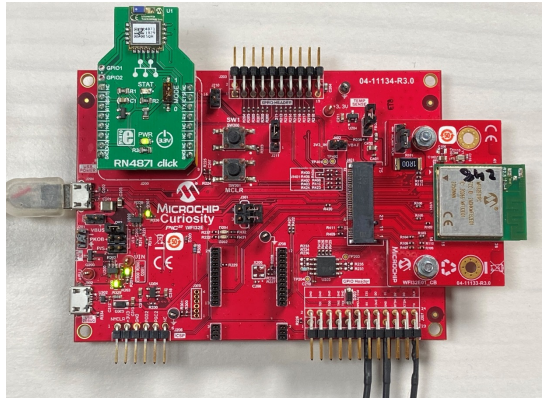


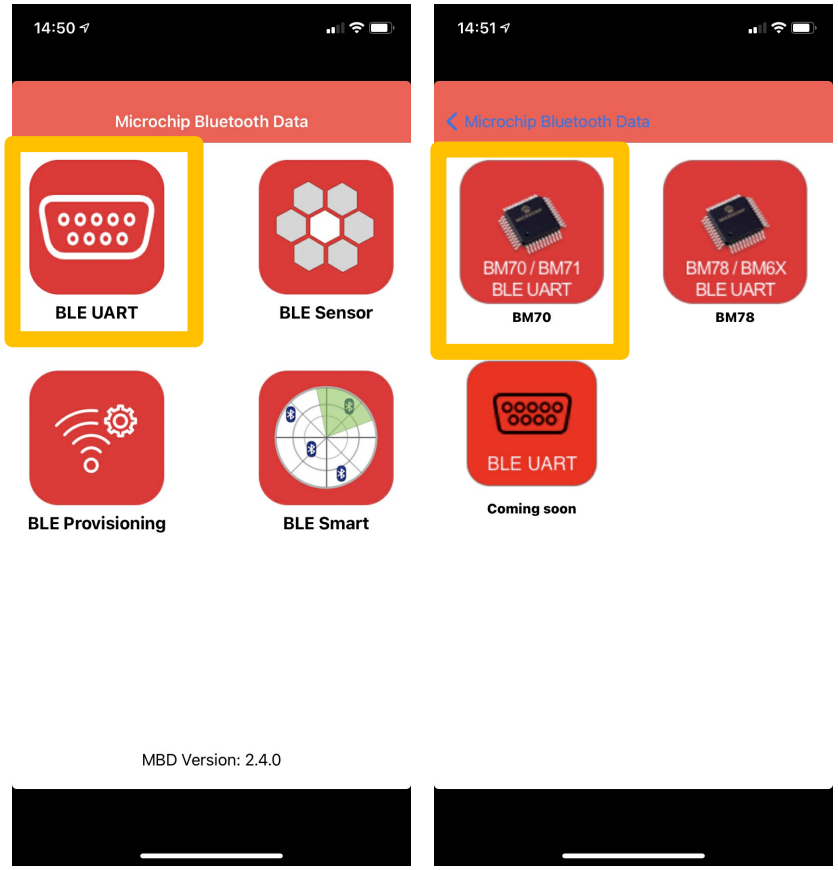
- Go to **PIC32MZW1_Workshop/07_projects/resources/software/**
- Download **ble_provisioning.zip**
- Build and program the code
- Open Tera Term to observe console logs
- Application starts in AP mode

```
TCP/IP Stack: Initialization Started
TCP/IP Stack: Initialization Ended - success

 mode=1 (0-STA,1-AP) saveConfig=1

 AP Configuration :
 channel=1
 ssidVisibility=1
 ssid=DEMO_AP_SOFTAP
 passphase=password
 authentication type=4 (1-Open,2-WEP,3-Mixed mode(WPA/WPA2),4-WPA2,5-Mixed mode(WPA2/WPA3),6-WPA3)
PIC32MZW1 AP Mode IP Address: 192.168.1.1
[APP_BLE] Init.
[APP_BLE] Configuration done.
Open Microchip Bluetooth Data App
- Select BLE UART and BM70
- Connect to your device WFI32_xxxx
- Select Transparent
- Frame format for Wi-Fi provisioning over BLE:
&wifiprov|<ssid>|<authtype>|<password>&
1: Open, 3: WPAWPA2, 4: WPA2, 5: WPA2WPA3, 6: WPA3
e.g. &wifiprov|DEMO_AP|3|password&
```

Wi-Fi Provisioning over BLE

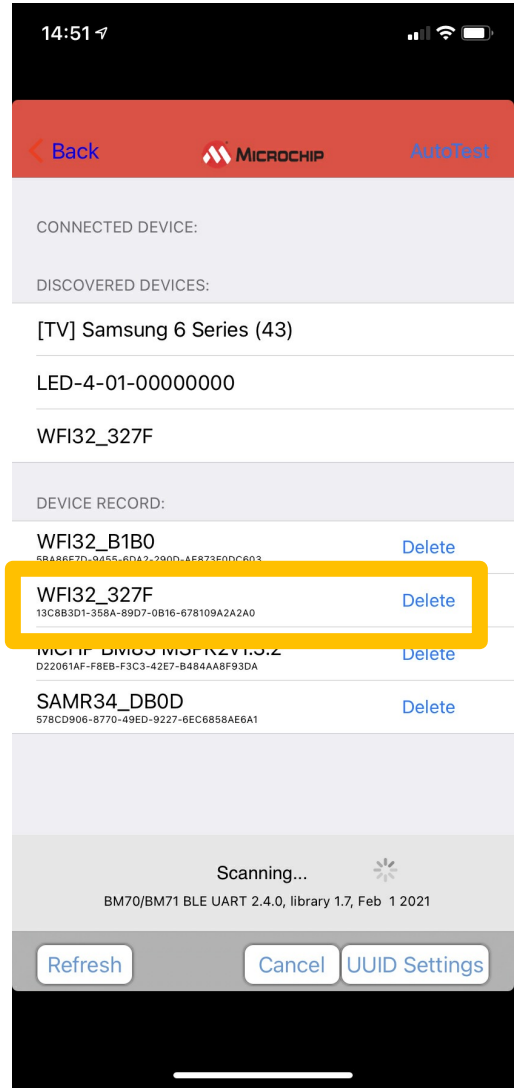
- Scan the QR code from the smartphone
- Copy your own Wi-Fi provisioning frame
- Open Microchip Bluetooth Data App
- Select BLE UART then BM70

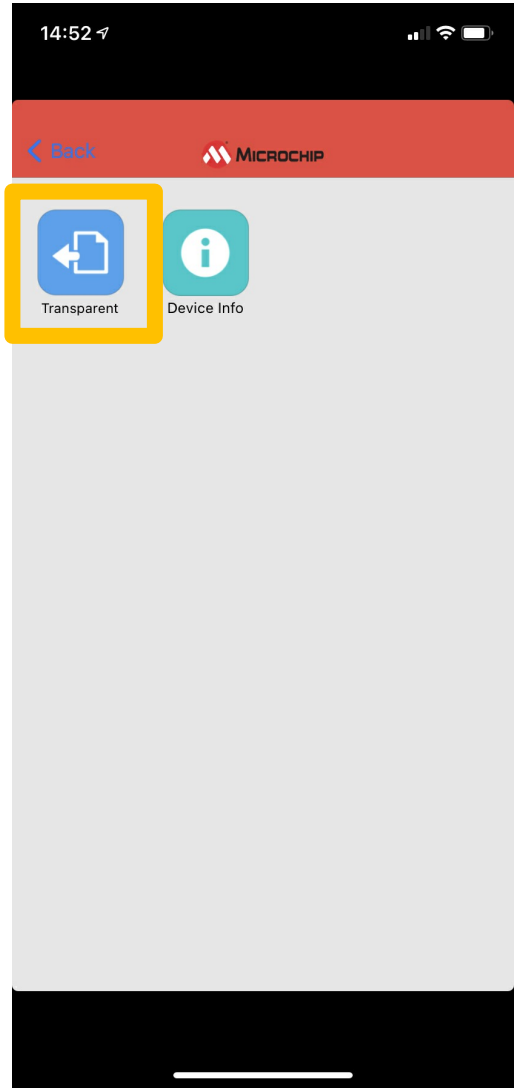




Wi-Fi Provisioning over BLE

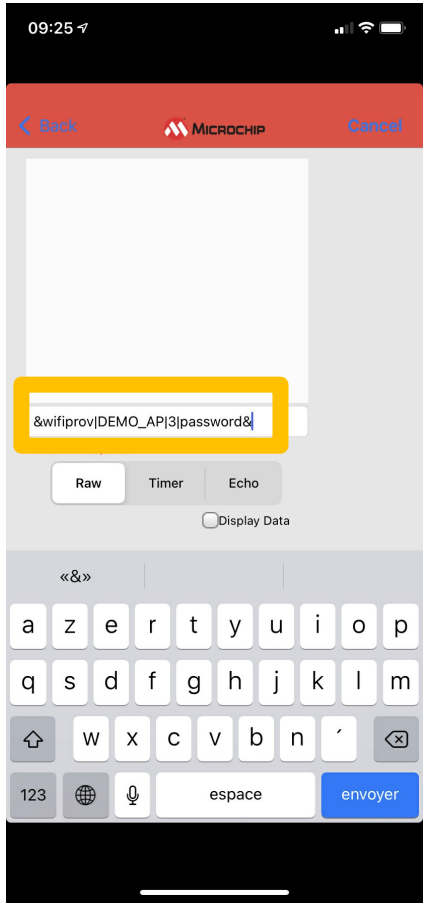
- Connect and select your WFI32_xxxx device
  - [APP_BLE] Connected
- Select Transparent option
  - [APP_BLE] Transparent stream opened





Wi-Fi Provisioning over BLE

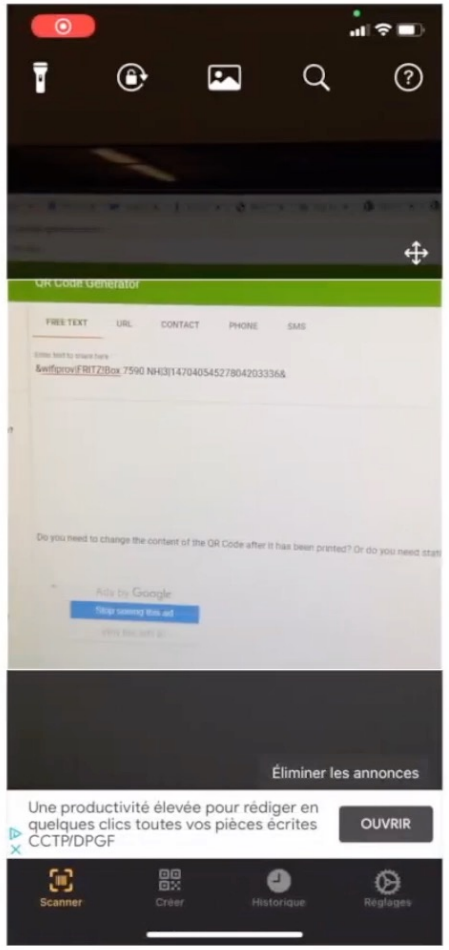
- Paste (or enter the data manually) and send the Wi-Fi provisioning frame
- Application restarts in STA mode using the new Wi-Fi configuration
- Application gets and IP address from the network




```
[APP_BLE] Frame received
SSID: DEMO_AP - AUTH: 3 - PASS: password
Wi-Fi Configuration done.ðTCP/IP Stack: Initialization Started
TCP/IP Stack: Initialization Ended - success

 mode=0 (0-STA,1-AP) saveConfig=1

 STA Configuration :
 channel=0
 autoConnect=1
 ssid=DEMO_AP
 passphase=password
 authentication type=3 (1-Open,2-WEP,3-Mixed mode(WPA/WPA2),4-WPA2,5-Mixed mode(WPA2/WPA3),6-WPA3)
[APP_BLE] Init.
[APP_BLE] Configuration done.
Open Microchip Bluetooth Data App
- Select BLE UART and BM70
- Connect to your device WFI32_xxxx
- Select Transparent
- Frame format for Wi-Fi provisioning over BLE:
&wifiprov|<ssid>|<authtype>|<password>&
1: Open, 3: WPAWPA2, 4: WPA2, 5: WPA2WPA3, 6: WPA3
e.g. &wifiprov|DEMO_AP|3|password&
 Trying to connect to SSID : DEMO_AP
 STA Connection failed.

 Trying to connect to SSID : DEMO_AP
 STA Connection failed.

IP address obtained = 192.168.1.149
Gateway IP address = 192.168.1.1
```

Wi-Fi Provisioning over BLE

Demo



Agenda

eLearning

- **Reference Example Codes**
  - Develop a Secured Smart Home Application from scratch
  - Wi-Fi ® Touch and OLED Display
  - Wi-Fi Provisioning over BLE
  - Wi-Fi to CAN Bus Bridge

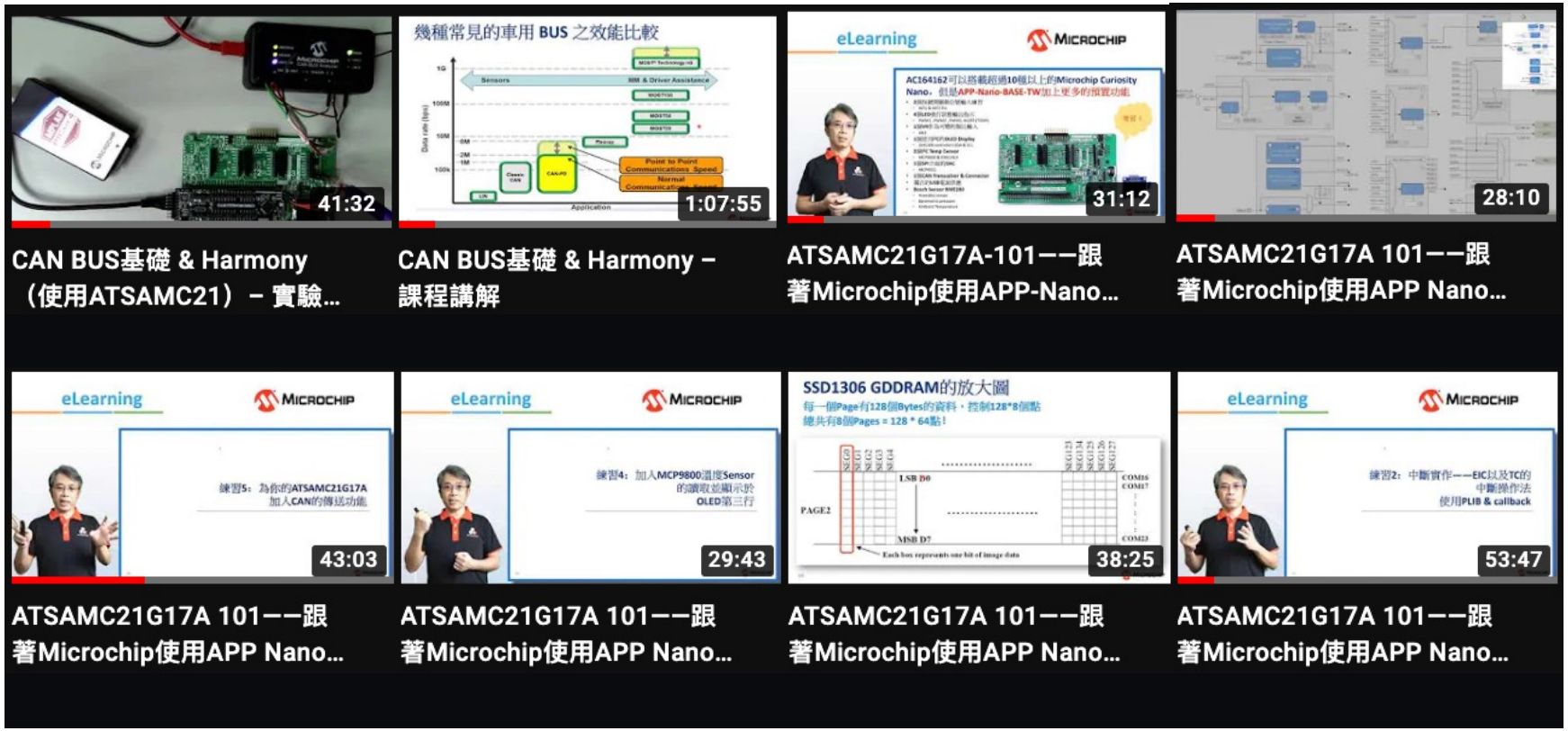
MICROCHIP

Wi-Fi to CAN Bus Bridge

Review APP-Nano-C21-D21-TW of eLearning Classes

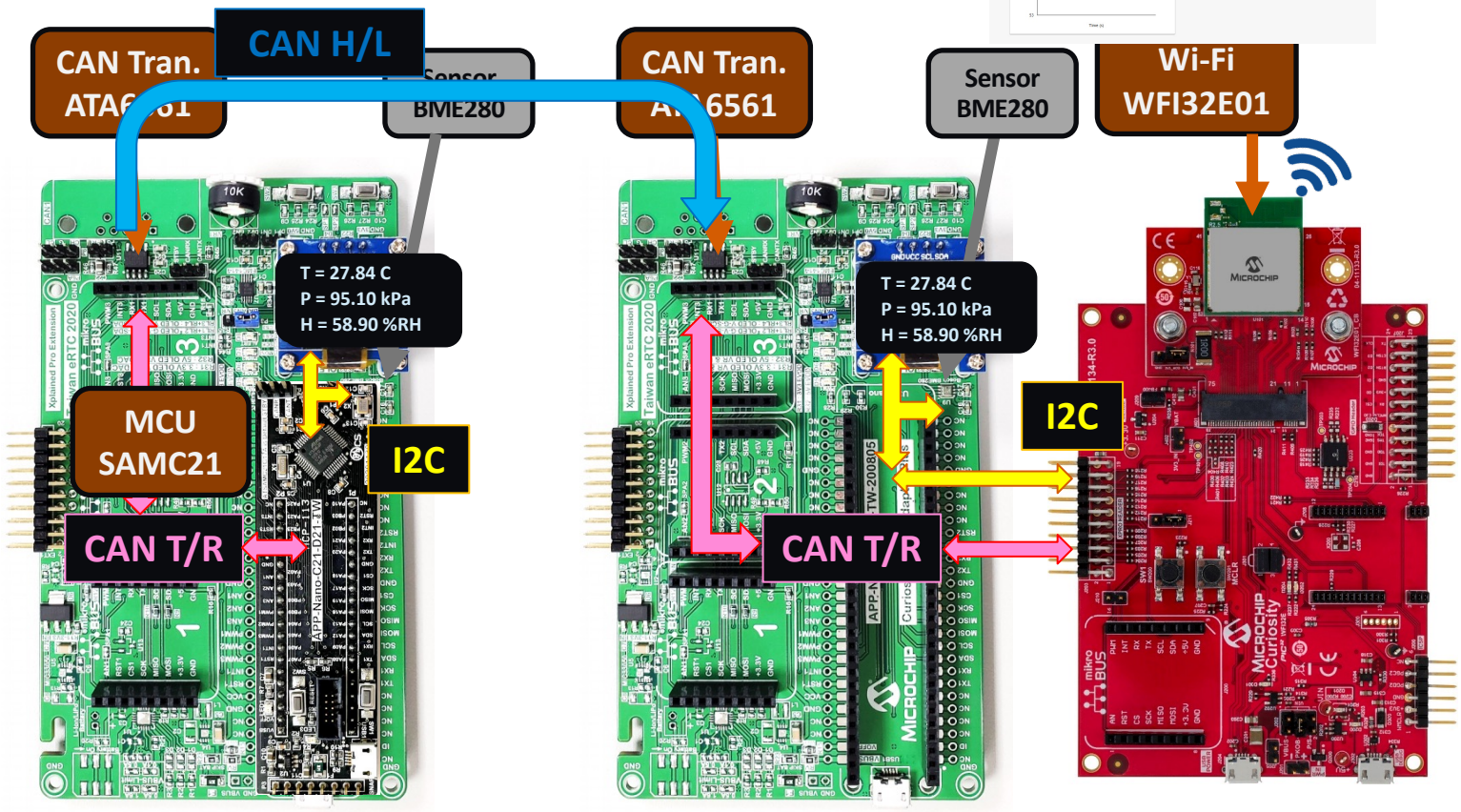
- CAN BUS基礎 & Harmony – 課程講解
- CAN BUS基礎 & Harmony (使用ATSAMC21) – 實驗講解
- ATSAMC21G17A-101 跟著Microchip使用APP-Nano-C21-D21-TW 為這個世界帶來更多的樂趣**1 ~ 6**

https://youtu.be/z3wg4OjW0Vs



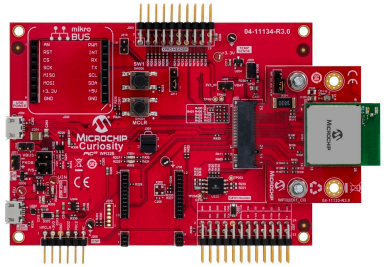
Wi-Fi to CAN Bus Bridge

Application & System Block Diagram

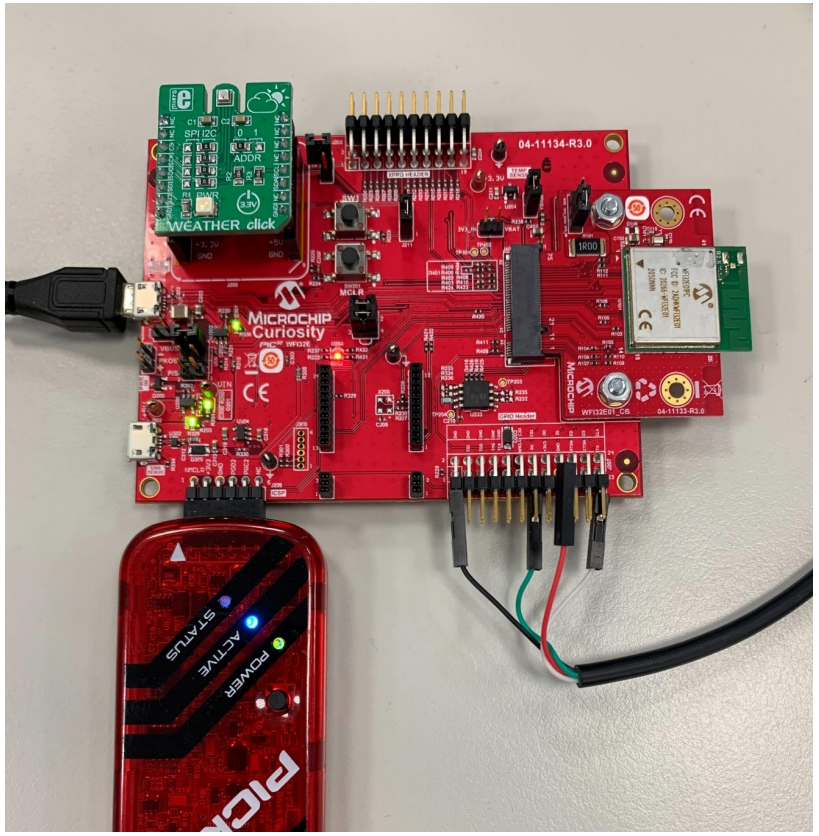


Wi-Fi to CAN Bus Bridge

PIC32WFI32E Curiosity Board - Environment Setup

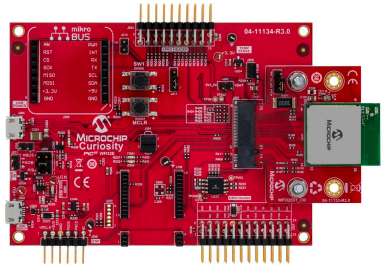


- **Visit https://github.com/MicrochipTech/PIC32MZW1_Workshop/tree/master/03_setup**
- **Follow the instructions to setup the PIC32MZW1 curiosity board and setup your development environment.**
- **Please install MPLAB® X IDE v5.45 or above, otherwise you might not able to open MPLAB® Harmony 3 Configurator in PIC32MZW1_Curiosity_OOB.**



Wi-Fi to CAN Bus Bridge

PIC32WFI32E Curiosity Board - Environment Setup



- **Download PIC32MZW1_Curiosity_OOB Example**
  - Visit https://github.com/MicrochipTech/PIC32MZW1_Curiosity_OOB

master · 2 branches · 7 tags · Go to file · Code

Vysakh Pillai Upgrade to latest H3 version ...

doc	Create .dummy	
resources/media	Upgrade to latest H3 version	
src/firmware	Upgrade to latest H3 version	
.gitignore	update U2TX PPS for v2.3 boards	
README.md	Upgrade to latest H3 version	
faq.md	Update faq.md	4 months ago

Clone

HTTPS GitHub CLI

https://github.com/MicrochipTech/PIC32

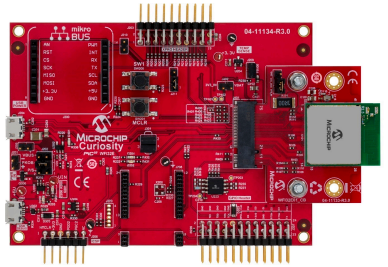
Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

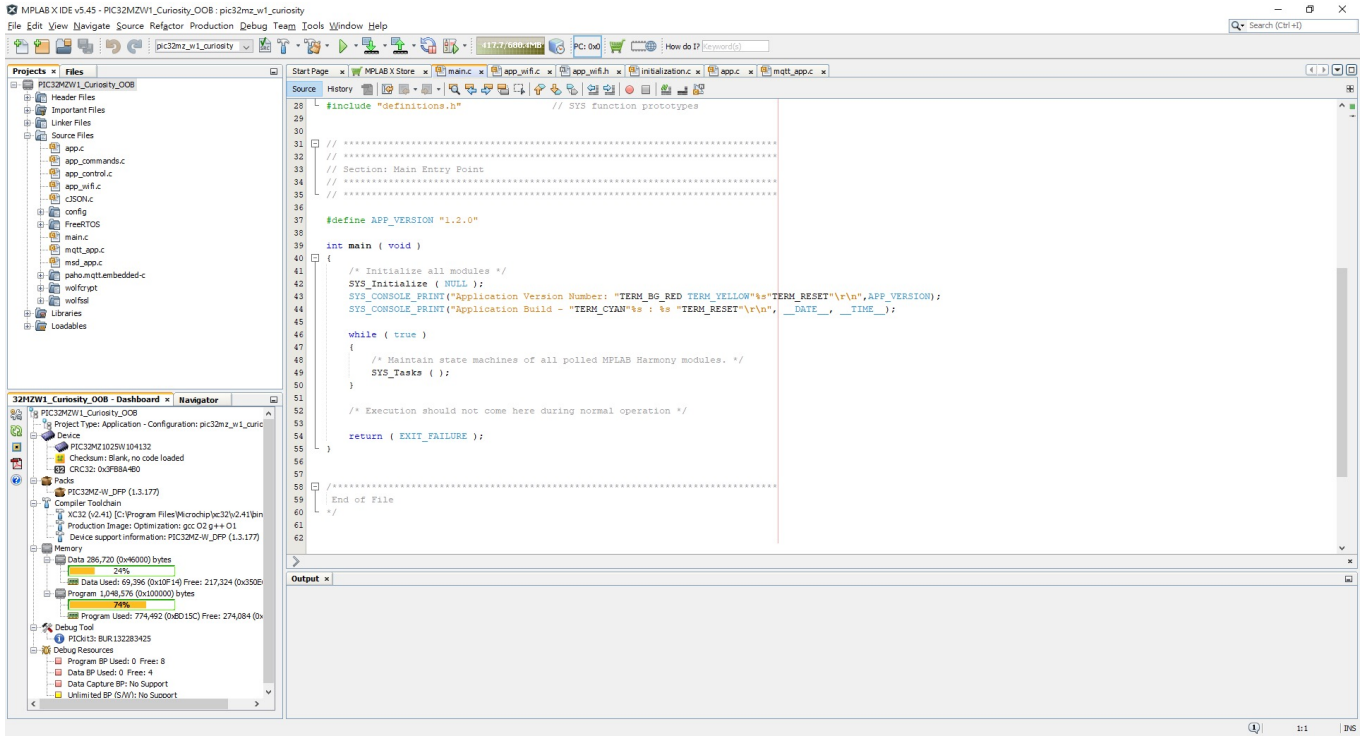
Download ZIP

Wi-Fi to CAN Bus Bridge

PIC32WFI32E Curiosity Board - Environment Setup

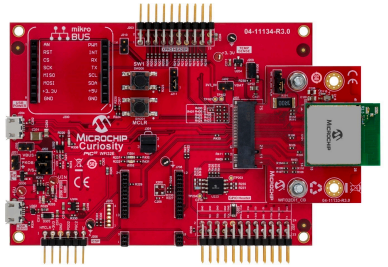


- **Open the project by MPLAB X IDE v5.45 or above**
  - The project is located at …\PIC32MZW1_Curiosity_OOB-master\src\firmware\PIC32MZW1_Curiosity_OOB.X

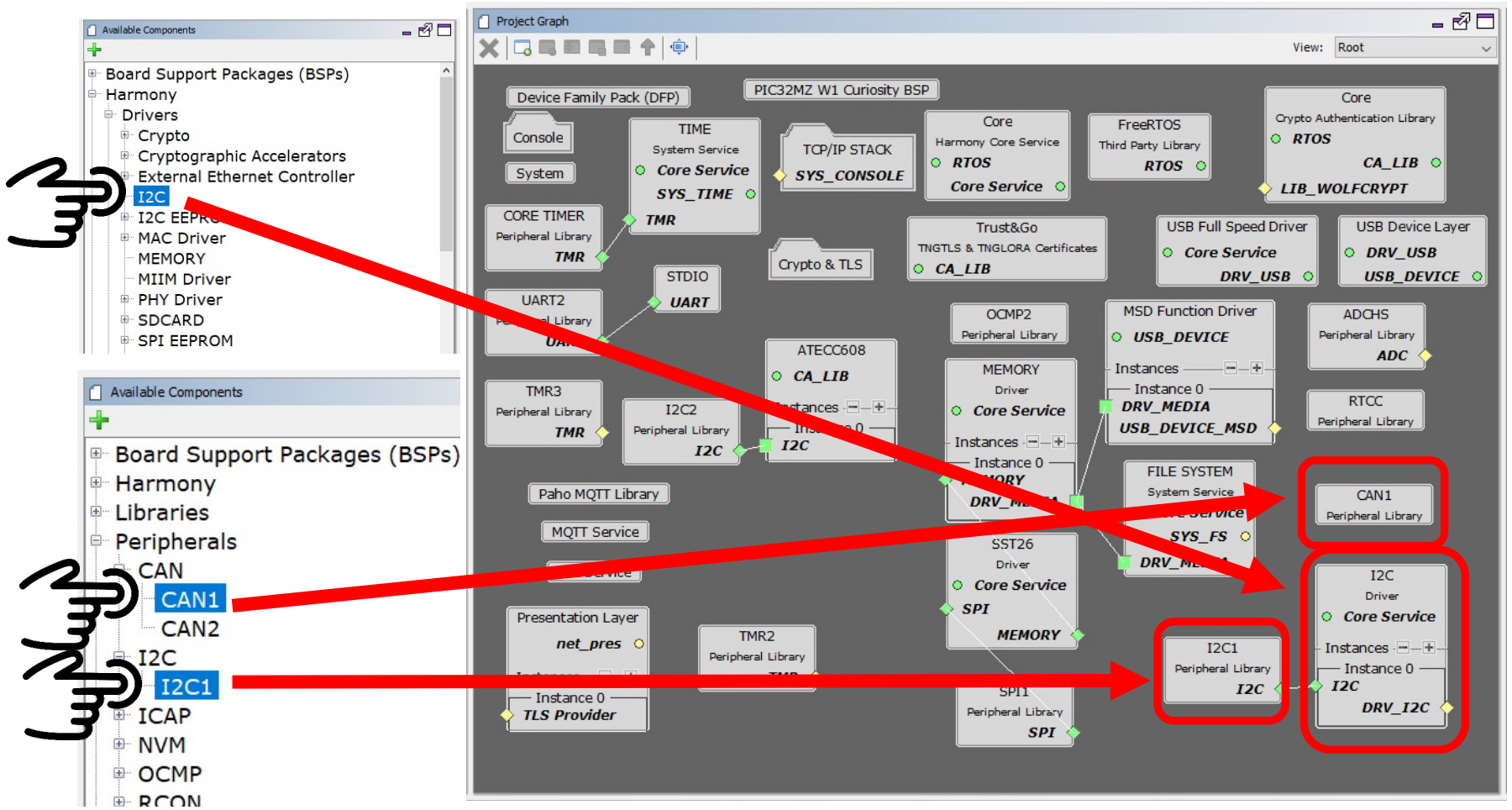


Wi-Fi to CAN Bus Bridge

PIC32WFI32E Curiosity Board - Harmony 3 Configurator

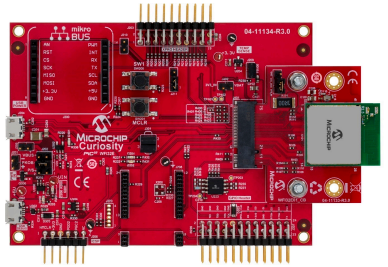


- **Add I2C and CAN interfaces**

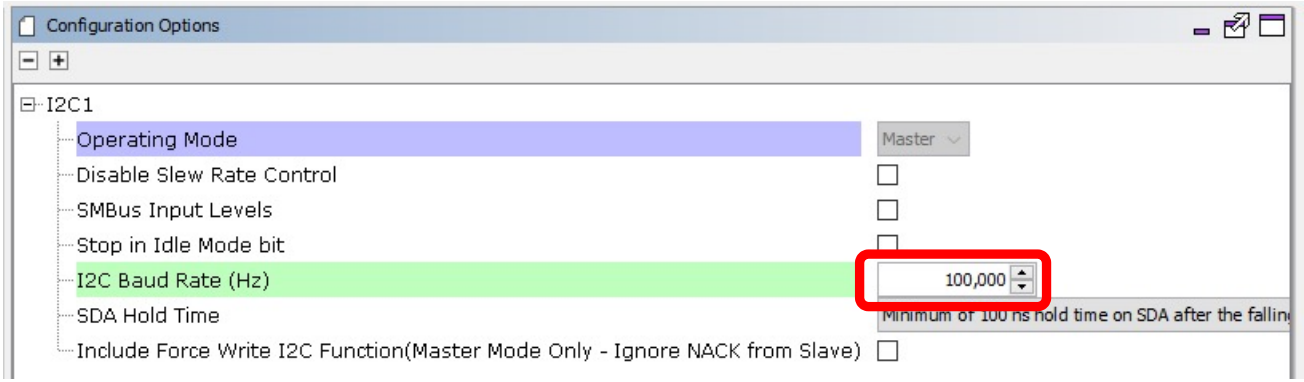


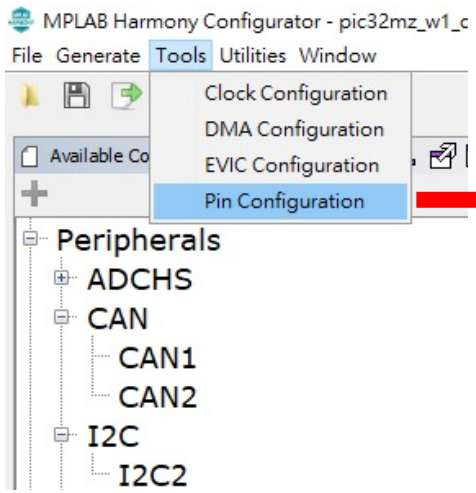
Wi-Fi to CAN Bus Bridge

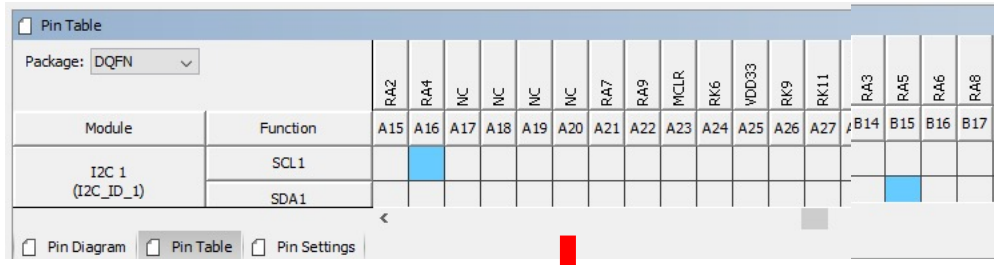
PIC32WFI32E Curiosity Board - Harmony 3 Configurator

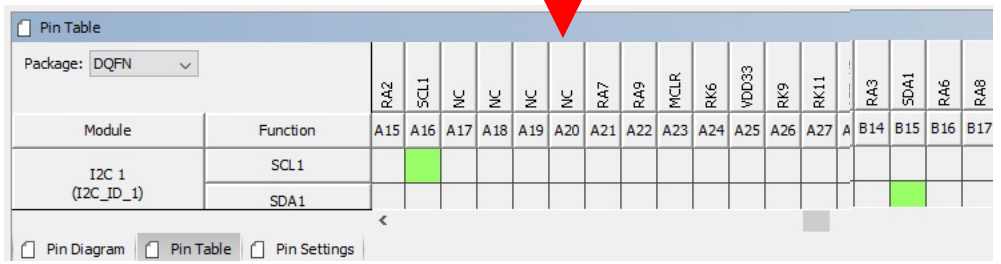


- **Change I2C1 Baud Rate (Hz) to 100,000 and set I2C1 pins**



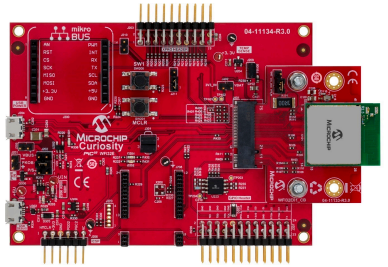




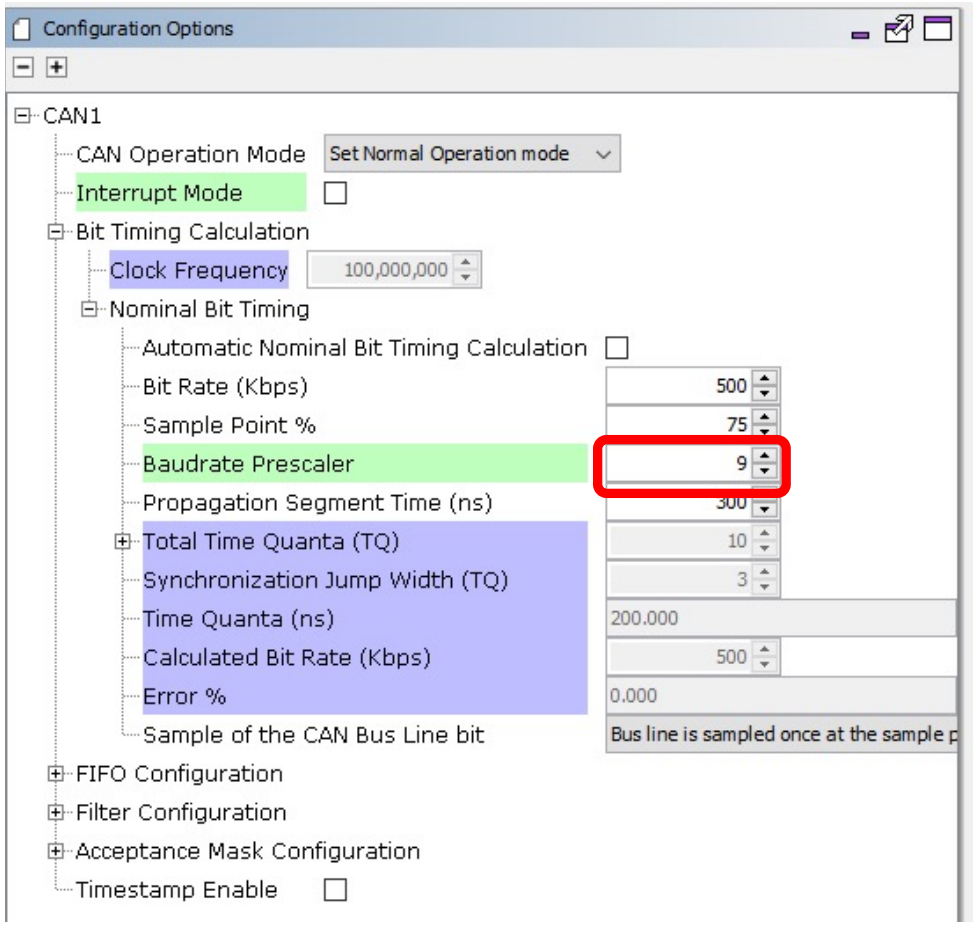


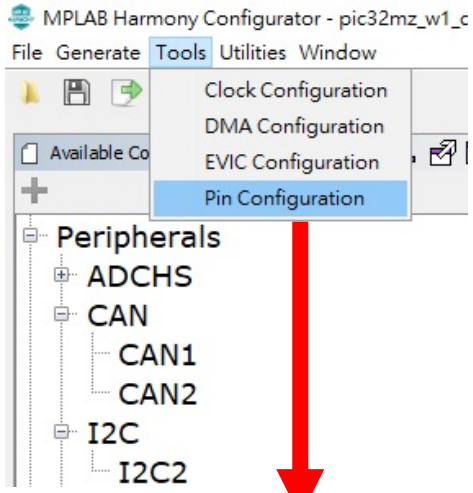
Wi-Fi to CAN Bus Bridge

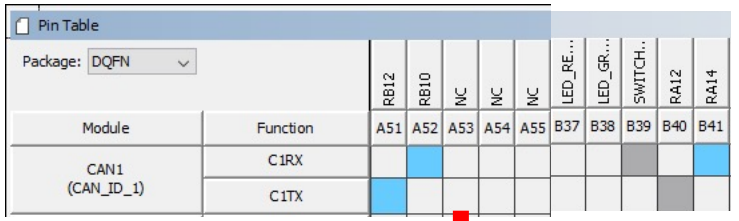
PIC32WFI32E Curiosity Board - Harmony 3 Configurator

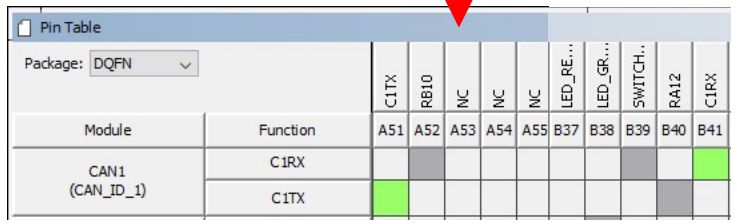


- **Change CAN1 Baudrate Prescaler to 9 and set CAN1 pins**



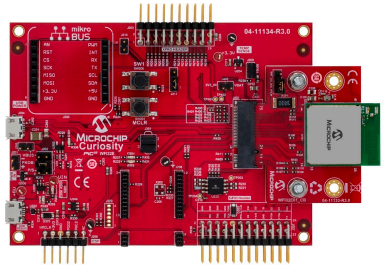




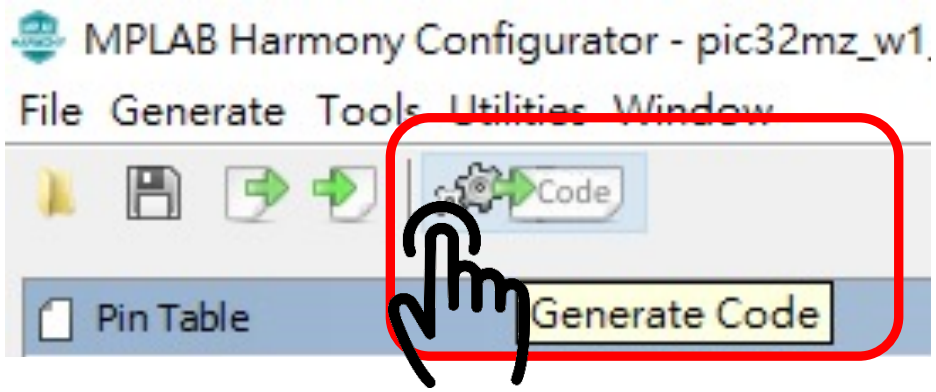


Wi-Fi to CAN Bus Bridge

PIC32WFI32E Curiosity Board - Harmony 3 Configurator

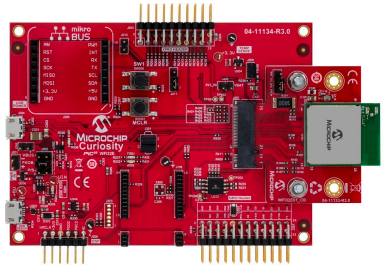


- **Generate Code**
  - Please make sure to Generate Code whenever you do any modification, otherwise, the code will not be generated!



Wi-Fi to CAN Bus Bridge

PIC32WFI32E Curiosity Board - Add Application Code



MPLAB X IDE v5.45 - PIC32MZW1_Curiosity_OOB : pic32mz_w1_curiosity

File Edit View Navigate Source Refactor Production Debug Team Tools Window Help

Projects × Files

PIC32MZW1_Curiosity_OOB
- Header Files
- Important Files
- Linker Files
- Source Files
  - app.c
  - app_commands.c
  - app_control.c
  - app_wifi.c
  - cJSON.c
  - config
  - FreeRTOS
  - main.c
  - mqtt_app.c
  - msd_app.c
  - paho.mqtt.embedded-c
  - wolfcrypt
  - wolfssl
- Libraries
- Loadables

app_control.c ×

Source History

```
447  void APP_CONTROL_Tasks(void) {
448      uint32_t status = 0;
449      uint16_t tempValue = 0;
450      uint8_t rx_message[8];
451      uint32_t rx_messageID = 0;
452      uint8_t rx_messageLength = 0;
453      //uint8_t length = 0;
454      CAN_MSG_RX_ATTRIBUTE msgAttr = CAN_MSG_RX_DATA_FRAME;
455      static int counter = 0;
456
457      WDT_Clear();
458      switch (app_controlData.state) {
459          case APP_CONTROL_STATE_INIT:
460          {
461              ADCHS_CallbackRegister(ADCHS_CH15, ADC_ResultHandler, (uintptr_t) NULL);
462              TMR3_Start(); /*TMR3 is used for ADC trigger*/
463
464              RTCC_CallbackRegister(RTCC_Callback, (uintptr_t) NULL);
465              setup_rtcc();
466              indicator_on();
467
468              /* Open I2C driver instance */
469              appData.drvI2CHandle = DRV_I2C_Open( DRV_I2C_INDEX_0, DRV_IO_INTENT_READWRITE);
470
471              OLED_Init();
472              OLED_CLS();
473
474              app_controlData.state = APP_CONTROL_STATE_MONITOR_CONNECTION;
475              break;
476          }
477          case APP_CONTROL_STATE_MONITOR_CONNECTION:
478          {
```

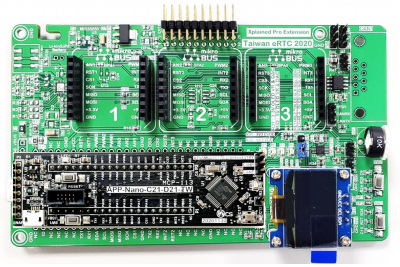
Output × Call Stack

Configuration Loading Error × Atmel-ICE × PIC32MZW1_Curiosity_OOB (Clean, Build, ...) ×

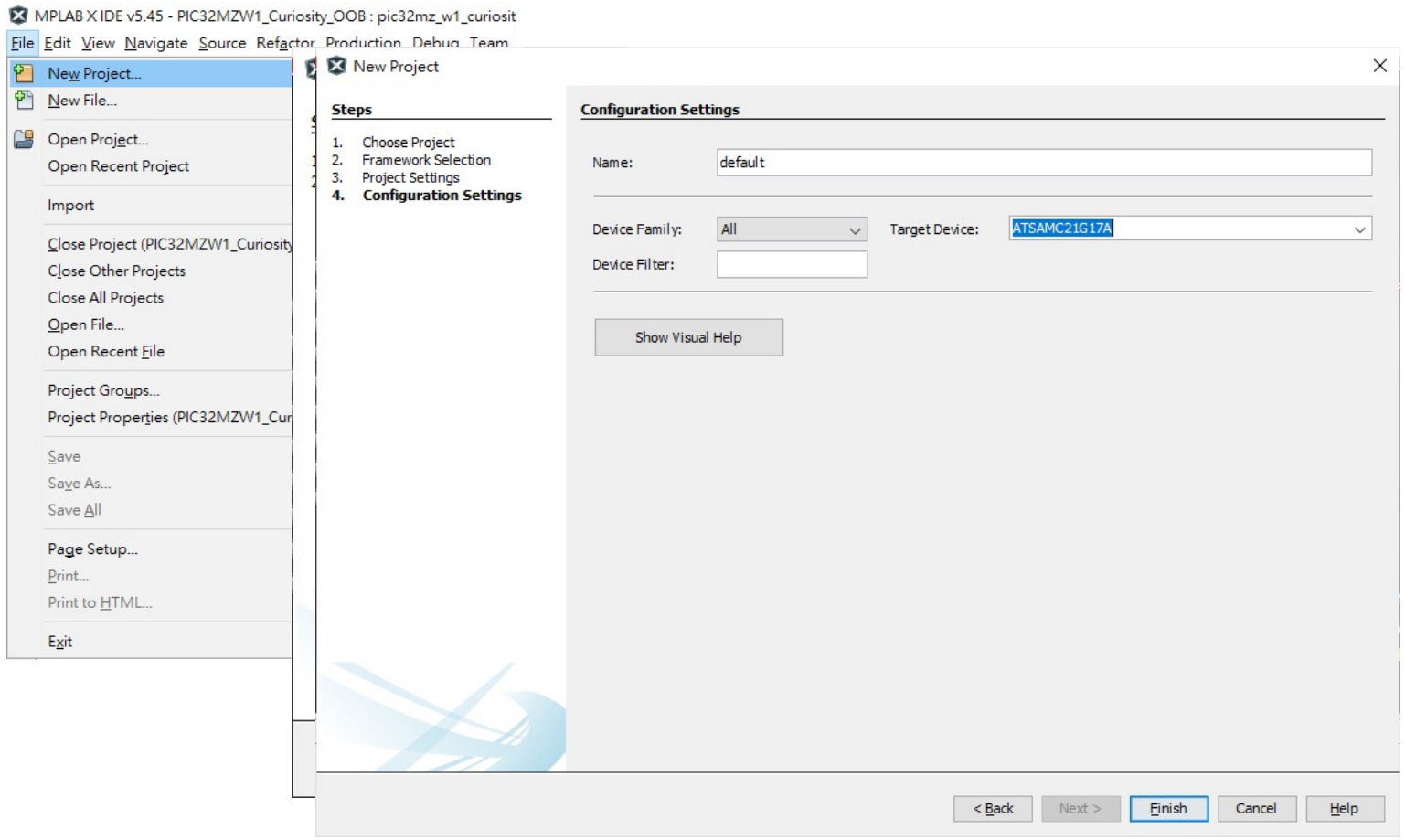
```
BUILD SUCCESSFUL (total time: 1m 19s)
Loading code from C:/Charley/Microchip/Trainings/20210531_WFI32_SAMC21/Demo_OOB_CAN/WFI32_CAN_Rx_OLED/PIC32MZW1_Curiosity_OOB/src/firmware/PIC32MZW1_Curiosity_OOB.X/dist/pic32mz_w1_curiosity/production/PIC32MZW1_Curiosity_OOB.X.production.hex...
Program loaded with pack,PIC32MZ-W_DFP,1.3.177,Microchip
Loading completed
```

Wi-Fi to CAN Bus Bridge

SAMC21 Nano Board - Environment Setup

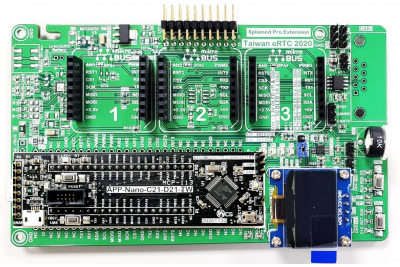


- **Select: ATSAMC21G17A**

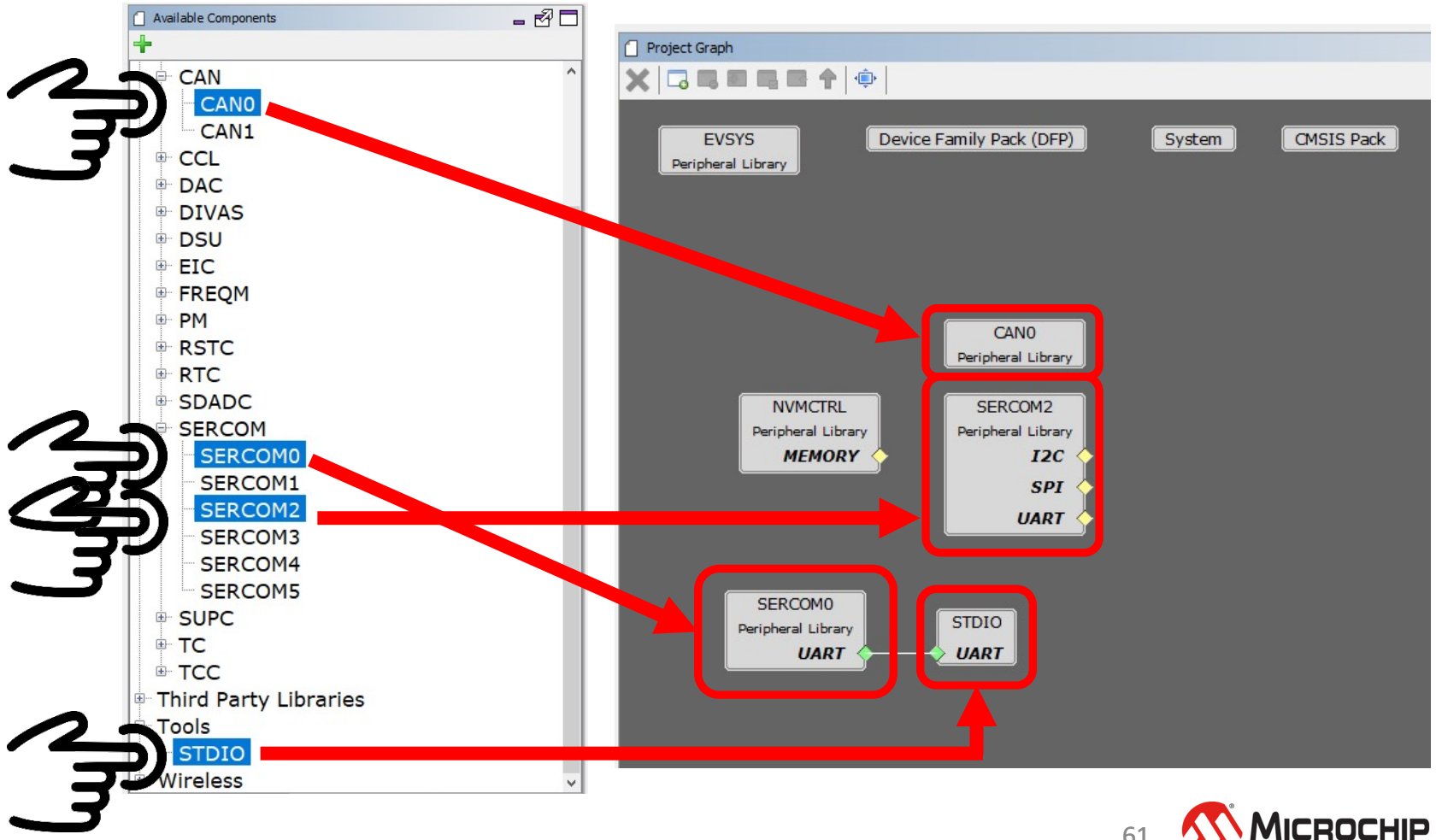


Wi-Fi to CAN Bus Bridge

SAMC21 Nano Board - Harmony 3 Configurator



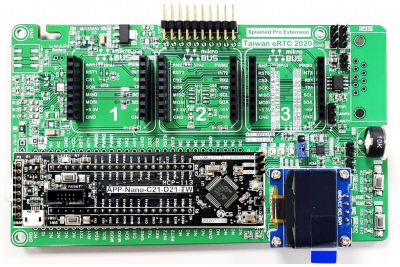
- **Add I2C , UART, and CAN interfaces**



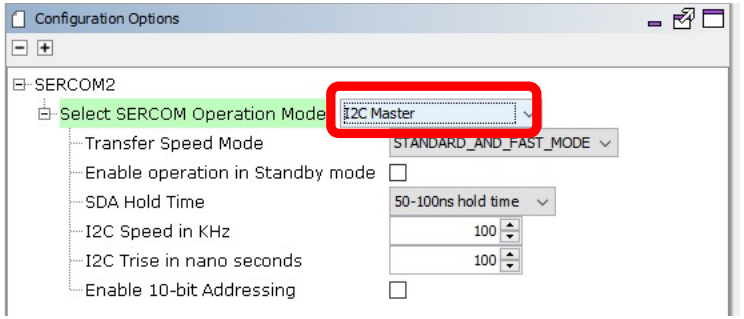
MICROCHIP

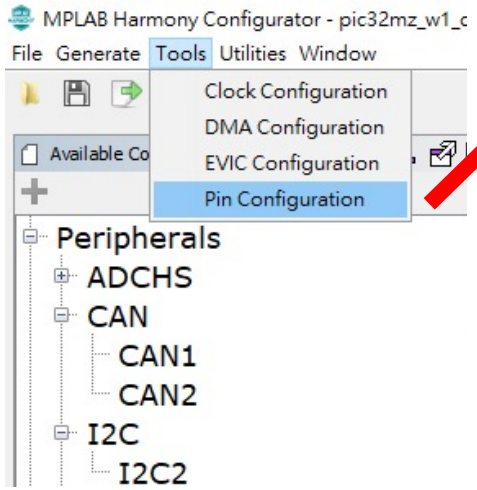
Wi-Fi to CAN Bus Bridge

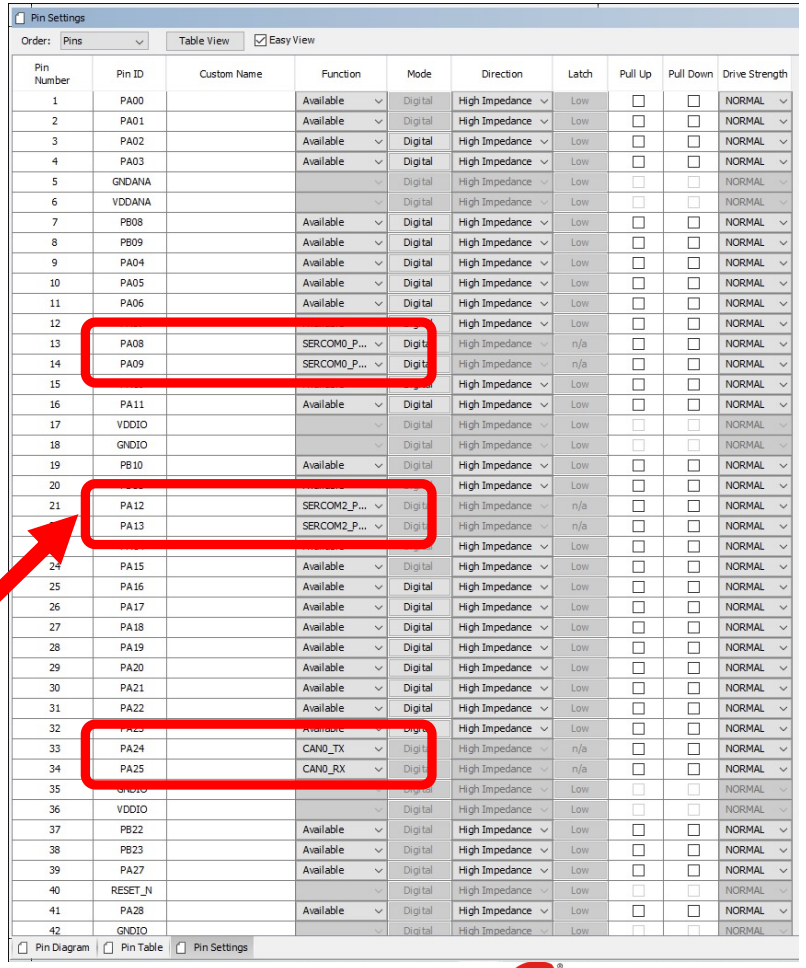
SAMC21 Nano Board - Harmony 3 Configurator



- **Change SERCOM2 to I2C Master, and set SERCOM0, SERCOM2, and CAN0 pins**
  - PA8 & PA9 are SERCOM0
  - PA12 & PA13 are SERCOM2
  - PA24 & PA25 are CAN0

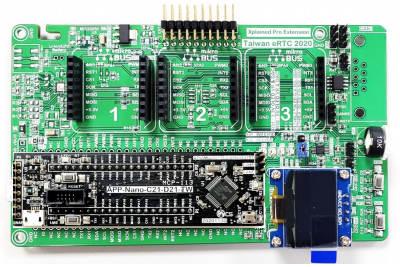




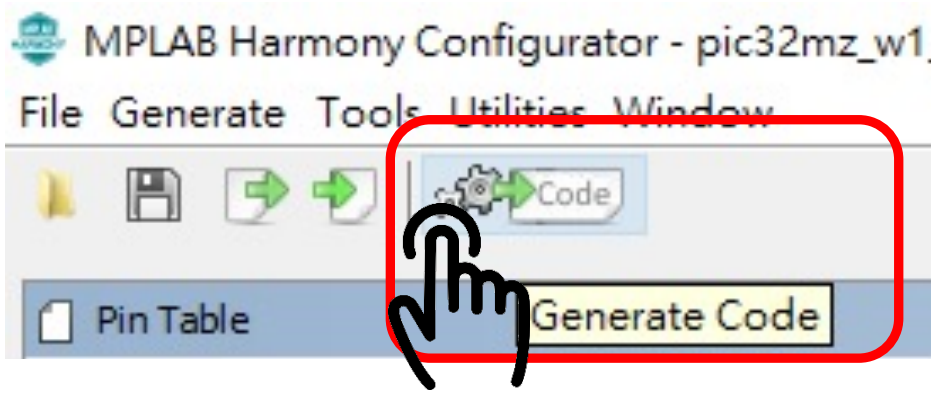


Wi-Fi to CAN Bus Bridge

SAMC21 Nano Board - Harmony 3 Configurator



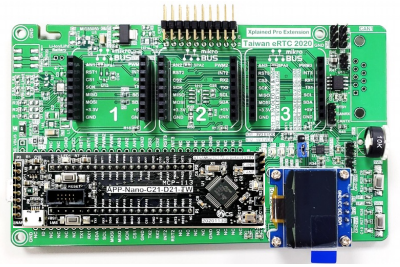
- **Generate Code**
  - Please make sure to Generate Code whenever you do any modification, otherwise, the code will not be generated!



MICROCHIP

Wi-Fi to CAN Bus Bridge

SAMC21 Nano Board - Add Application Code



MPLAB X IDE v5.45 - can_demo_sam_c21n_xpro : sam_c21n_xpro

File Edit View Navigate Source Refactor Production Debug Team Tools Window Help

Projects × Files

- can_demo_sam_c21n_xpro
  - Header Files
  - Important Files
  - Linker Files
  - Source Files
    - config
    - main.c
  - Libraries
  - Loadables
- PIC32MZW1_Curiosity_OOB

main.c ×

Source History

```
200
201  int main ( void )
202  {
203      uint32_t status = 0;
204      uint16_t bitMask = 0x00FF;
205      uint8_t rx_message[8], tx_message[8];
206      uint32_t rx_messageID = 0, tx_messageID = 123;
207      uint8_t rx_messageLength = 0;
208      CAN_MSG_RX_FRAME_ATTRIBUTE msgFrameAttr = CAN_MSG_RX_DATA_FRAME;
209      uint32_t getDataDelay = 0, resetDelay = 0;
210      CALIBR_BME280_STATES calibStage = CALIBR_BME280_T_P_WRITE;
211      uint8_t tempWrite = 0, writeStage = 0;
212      int tempTemp = 0, tempPress = 0, tempHumid = 0;
213
214      appData.state = APP_BME280_STATE_INIT;
215
216      /* Initialize all modules */
217      SYS_Initialize ( NULL );
218
219      SERCOM2_I2C_CallbackRegister(i2cEventHandler, 0);
220
221      /* Set Message RAM Configuration */
222      CAN0_MessageRAMConfigSet(Can0MessageRAM);
223
224      while ( true )
225      {
226          /* Check the application's current state. */
227          switch(appData.state)
228          {
229              case APP_BME280_STATE_INIT:
230              {
231                  /* Open I2C driver instance */
232                  appData.state = APP_BME280_STATE_READY_WAIT;
```

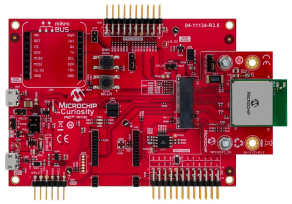
Output × Call Stack

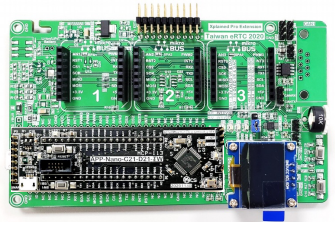
Configuration Loading Error × Atmel-ICE ×

Loading script file C:\Program Files\Microchip\MPLABX\v5.45\packs\Microchip\SAMC21_DFP\3.4.78\samc21\..\scripts\dap_cortex-mOplus.py

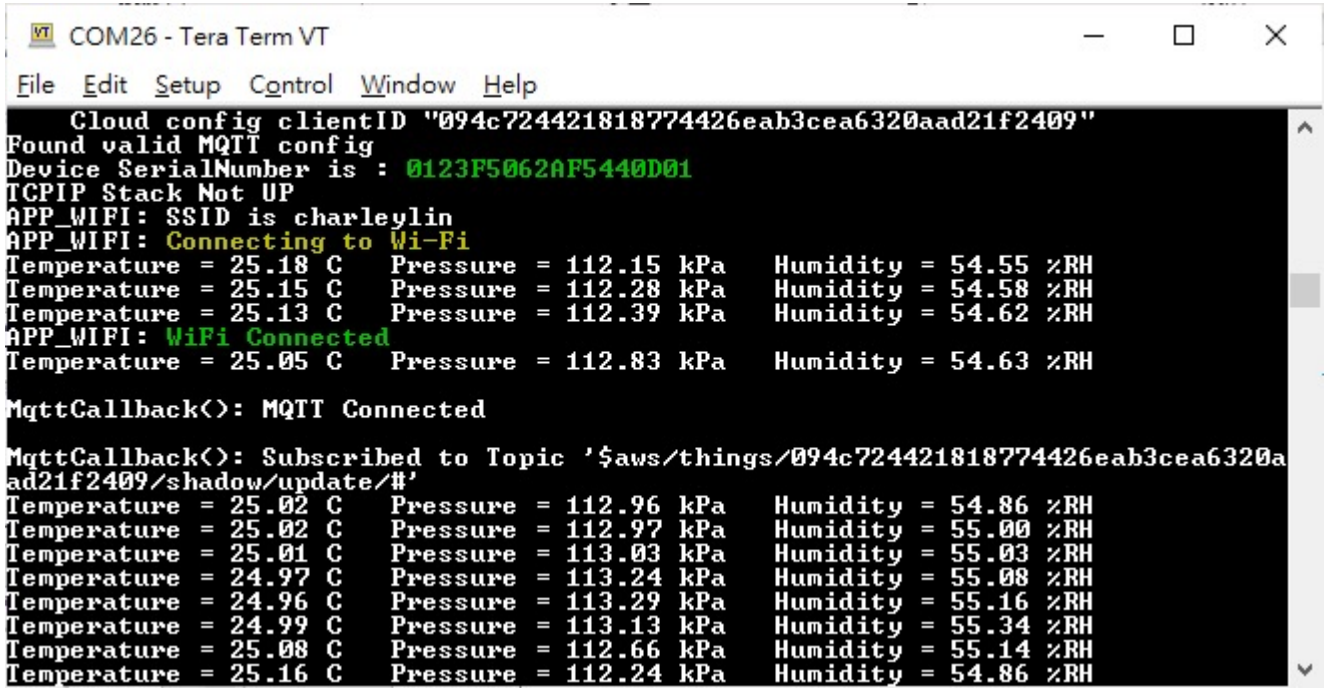
Wi-Fi to CAN Bus Bridge

Demo



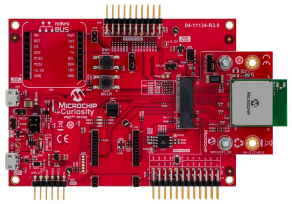


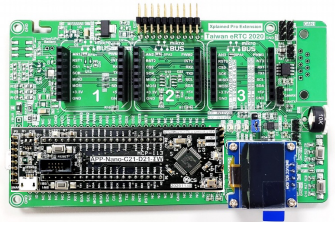
- **Follow the instructions in**
  https://github.com/MicrochipTech/PIC32MZW1_Curiosity_OOB
  You can get the results below:

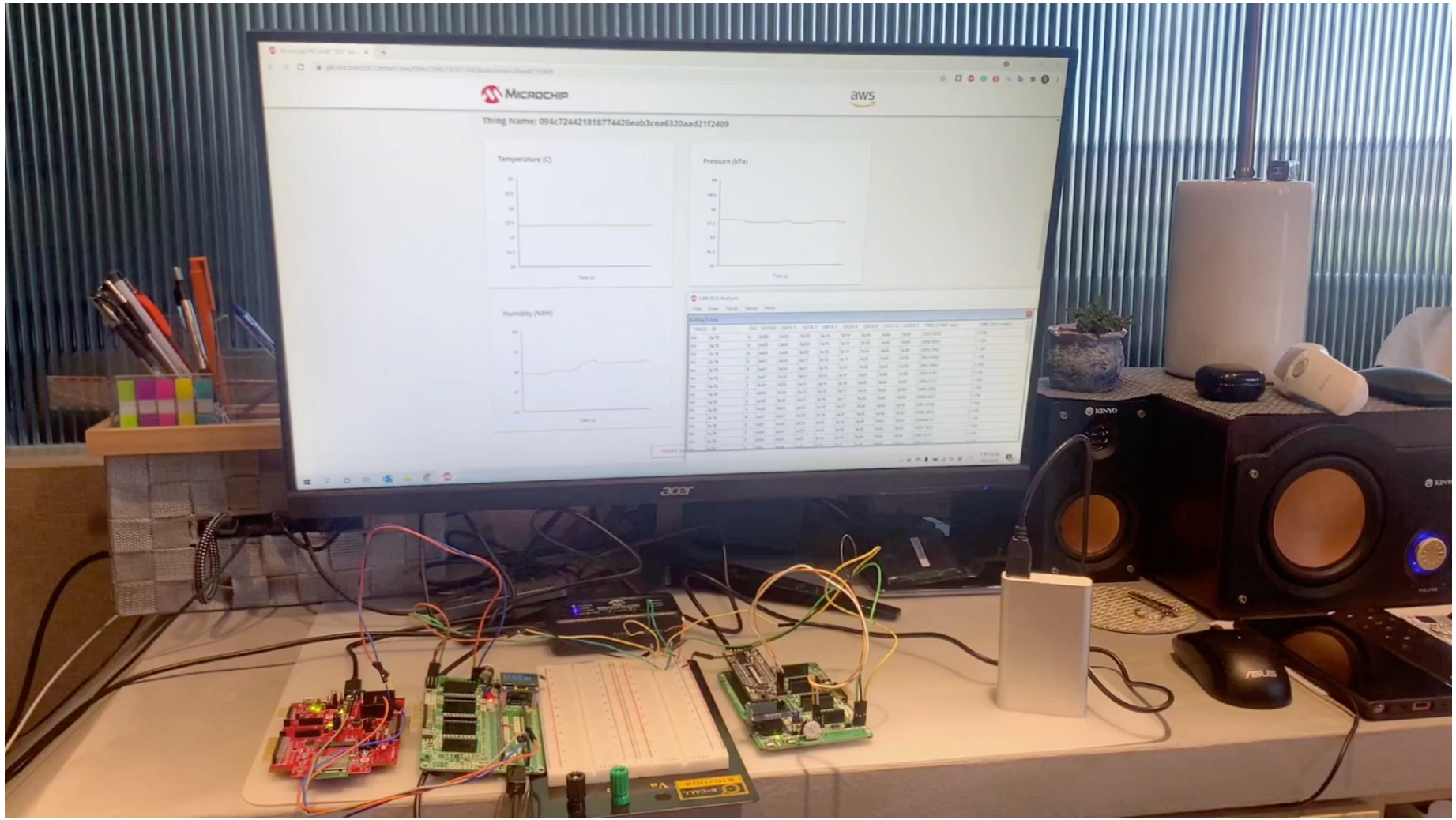


Wi-Fi to CAN Bus Bridge

Demo







MICROCHIP UNIVERSITY

Microchip has been delivering superior learning for our clients over the last 23 years at our annual users' conference. Our goal for the new online Microchip University Program is to provide you with all the same information you need to design robust embedded control systems with Microchip solutions.

All Microchip University courses are being offered for free at this time.

Microchip - Chinese [Traditional]

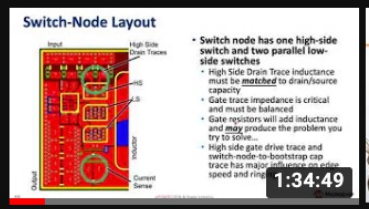
199 位訂閱者

已訂閱

首頁 影片 播放清單 頻道 討論 簡介

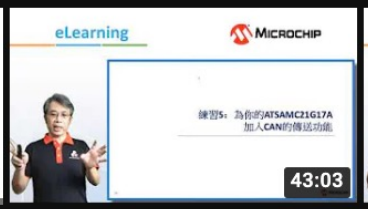
上傳的影片

▶ 全部播放



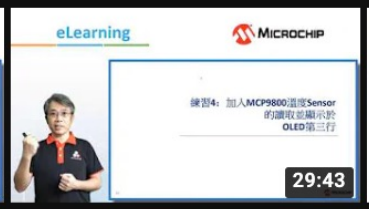
ePOW007——電源分佈網路與電源完整性

觀看次數：29次・3 週前



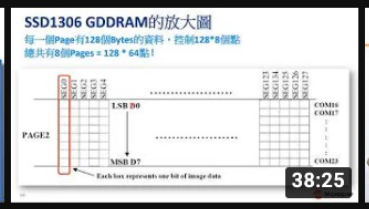
ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：97次・1 個月前



ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：28次・1 個月前



ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：31次・1 個月前

ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：39次・1 個月前

ATSAMC21G17A 101——跟著Microchip使用APP Nano...

觀看次數：27次・1 個月前

Featured Channels

Microchip Technology
5.38萬 位訂閱者

Microchip - Chinese [Simplified]
540 位訂閱者

Microchip - Japanese
343 位訂閱者

Microchip - Korean
1310 位訂閱者

Microchip Makes
1.87萬 位訂閱者

eLearning

Wi-Fi® SoC Module *WFI32E01*

參考範例應用篇

Thank you!