CQ Cordelia-I STM32 AT Command Demo Application Note

Version V1.00

Table of Contents

1 Scope

This application note outlines the STM32 AT UART command example, which demonstrates how to control a Würth Electronics Cordelia-I module using a low-end microcontroller. It shows how this setup enables connectivity to a cloud service in collaboration with the Crypto Quantique security platform (QuarkLink).

2 Introduction

QuarkLink is Crypto Quantiques' universal IoT security platform that uses advanced cryptographic techniques to integrate with a hardware root of trust to provide provisioning, onboarding and monitoring for easy scalability and reliable security.

The STM32F401 AT UART command application example implements minimal AT commands needed to quickly get you started using the Würth Electronics Cordelia-I module. Both Quarklink local MQTT and AWS brokers are supported.

2.1 Required Resources

2.1.1 QuarkLink

A pre-requisite of the reader is that access to a Crypto Quantique QuarkLink Ignite instance is required. QuarkLink Ignite access is defined as a URL for the QuarkLink instance with an associated Username and Password to allow access. Users will need to provision the Cordelia-I module with their QuarkLink Ignite instance. A QuarkLink Instance can be created by visiting https://signup.quarklink.io.

2.1.2 AWS Account

Optionally, the reader is required to have access to an Amazon AWS account if they wish to make use of AWS services. To create an AWS account go to https://aws.amazon.com.

2.1.3 Documentation

- Crypto Quantique QuarkLink User Guide (https://docs.quarklink.io/)
- Crypto Quantique GitHub (https://github.com/cryptoquantique)
- Amazon AWS documentation (see Amazon AWS website)
- Cordelia Wurth website : (https://www.we-online.com/en/components/products/CORDELIA-I)

2.2 Intended Audience

The intended audience is users who want to develop secure MQTT or Amazon AWS cloud-based applications compatible with the Crypto Quantique universal IoT security platform (QuarkLink) for connected (IoT) devices using the Würth Electronics Cordelia-I module.

3 General Overview

The goal of this example application is to show how simple it can be to securely connect your IoT device to the cloud using the Würth Electronics Cordelia-I WiFi module and Crypto Quantique's Quarklink.

4 Hardware

4.1 Requirements

This example project requires the following hardware:

- STM32F4 NUCLEO Evaluation Board (Part number : ST - NUCLEO-F401RE)
- Cordelia-I WiFi Module EV-Kit (Part number : Wurth - 2610019225011)
- Header connection wires

4.2 Connections

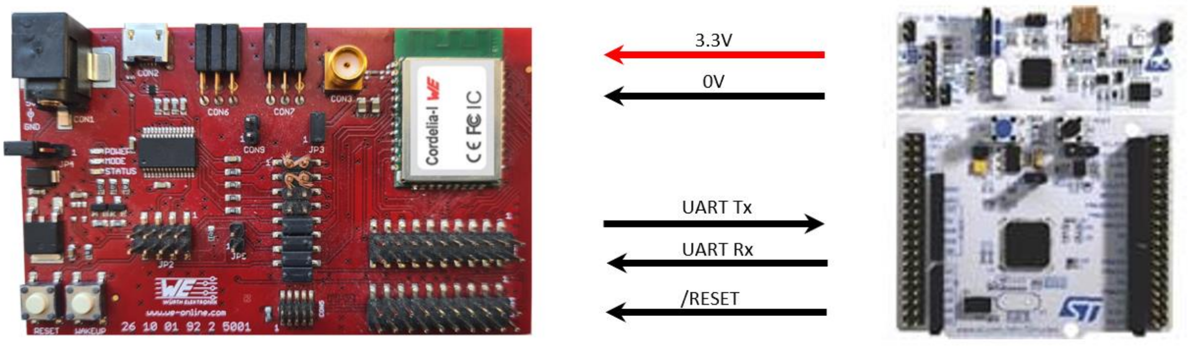
To reliably control the Cordelia-I module we need 5 connections between the STM32F4 board and the Cordelia-I WiFi module (see schematic diagram section 4.3).

Power – 3v3
Ground
RS232 Tx
RS232 Rx
/Reset

Additionally for this example we have enabled some status LED's for convenience and testing.

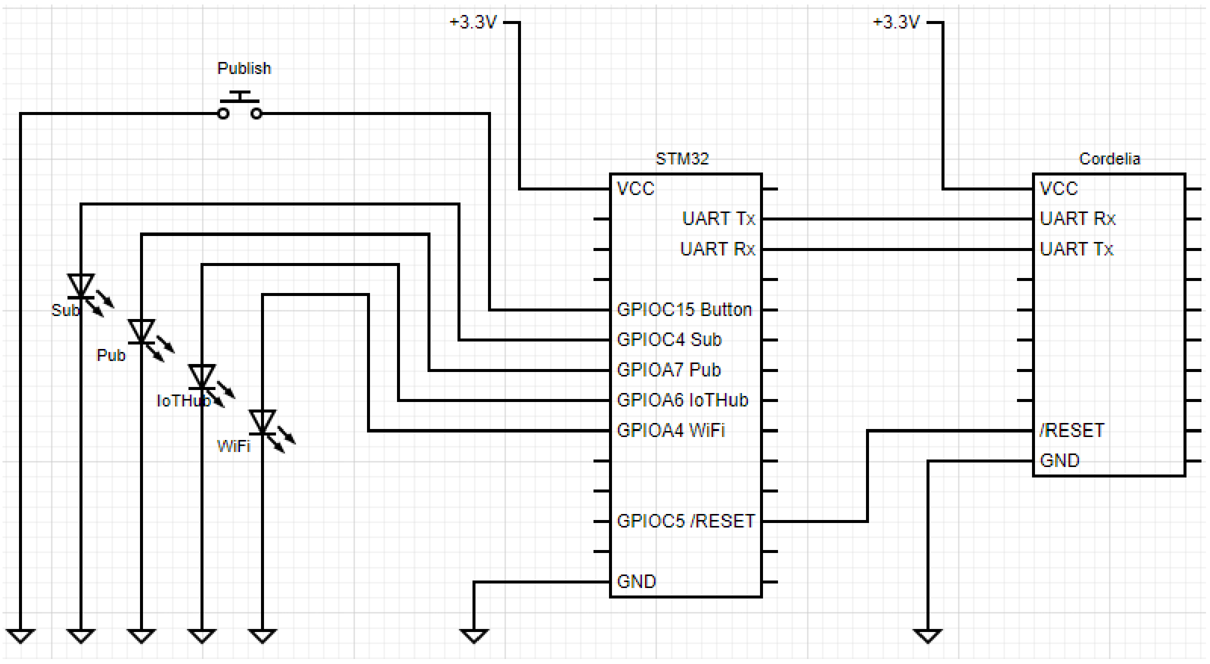
WiFi connected
IoT Hub enrolled
Flashes when successfully published a message
Flashes when successfully received a subscribed message

Function	**STM32**	**Cordelia-I**
Power	3v3	CON9 pin 2 - VCC 3v3
Ground	GND	CON9 pin 1 – GND
Rx	UART6 Rx GPIOC7	JP1 pin 4 UART0 Tx
Tx	UART6 Tx GPIOC6	JP1 pin 2 UART0 Rx
/RESET	GPIOC5	JP2 pin 2 nRESET
Button – push to publish	GPIOC13	
WiFi connected	GPIOA4	
IoTHub connected	GPIOA6	
Published message sent	GPIOA7	
Subscribed message received	GPIOC4	



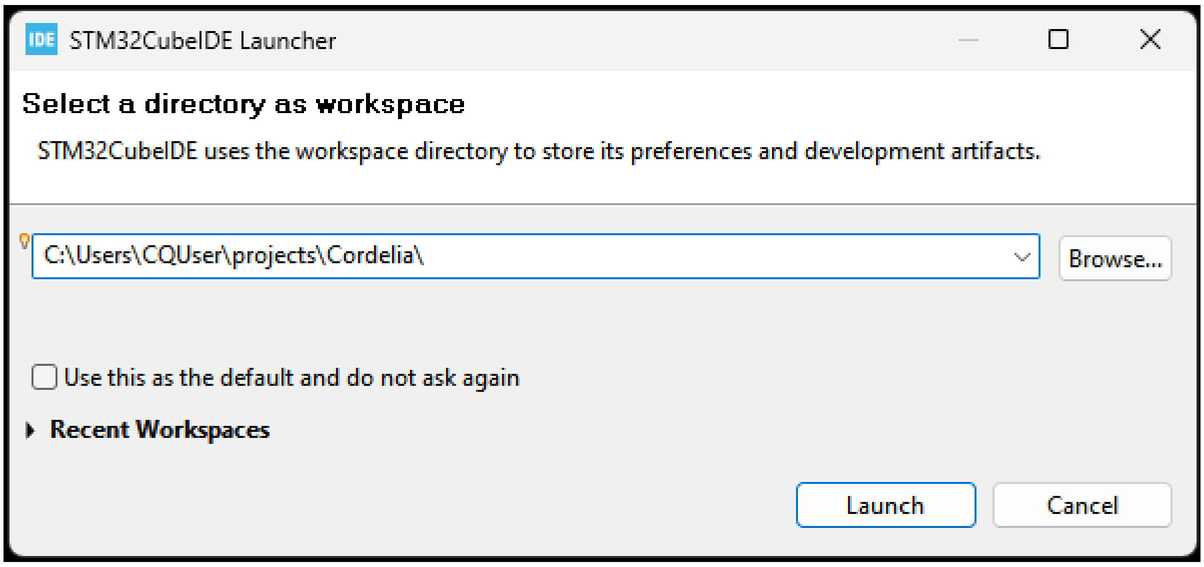
Evaluation Board Interconnections

4.3 Schematic Diagram



5 Build environment setup

The STM32 example application is built using ST Microelectronics STMCubeIDE. The user should clone the example project from the Crypto Quantique GitHub (see section 2.1.3) into a suitable local path, launch the IDE and create a new workspace.


STM32CubeIDE Launcher

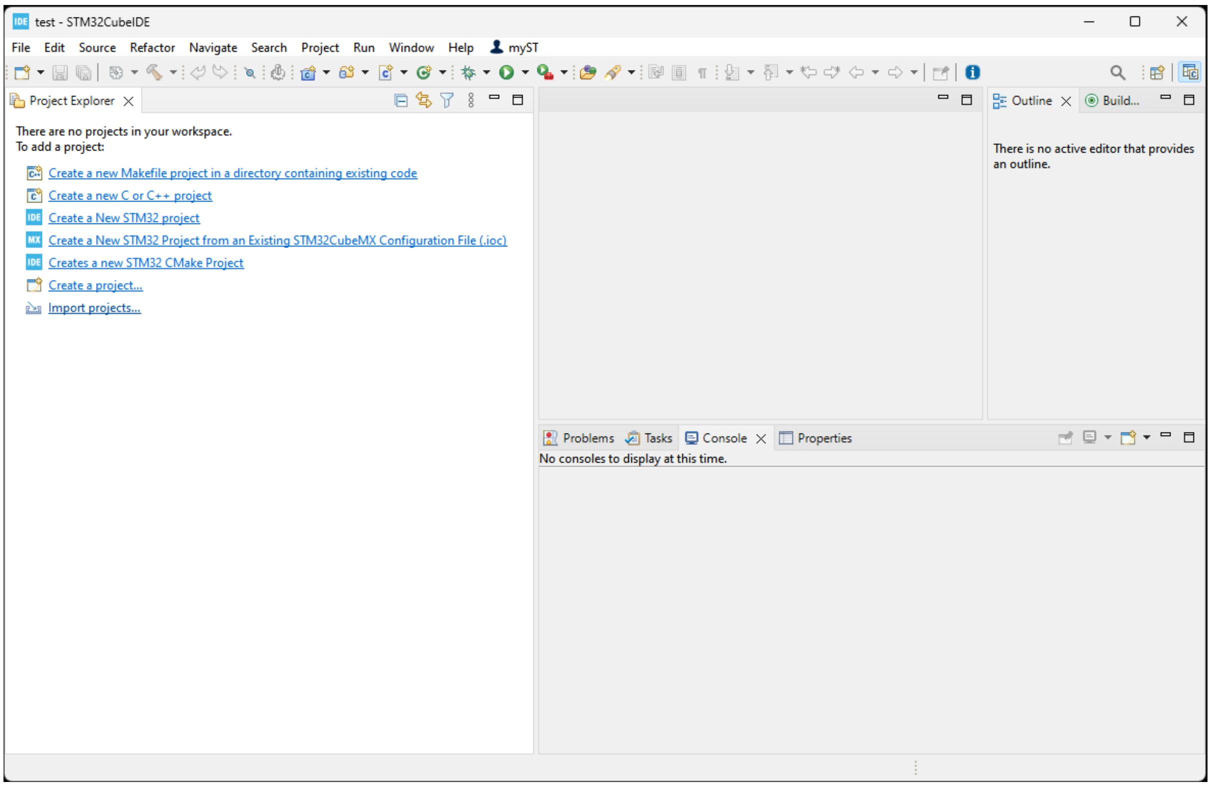
Select a directory as workspace

STM32CubeIDE uses the workspace directory to store its preferences and development artifacts.

C:\Users\CQUser\projects\Cordelia\

☐ Use this as the default and do not ask again

Next you will need to import the project – select the ‘import projects’ hyperlink

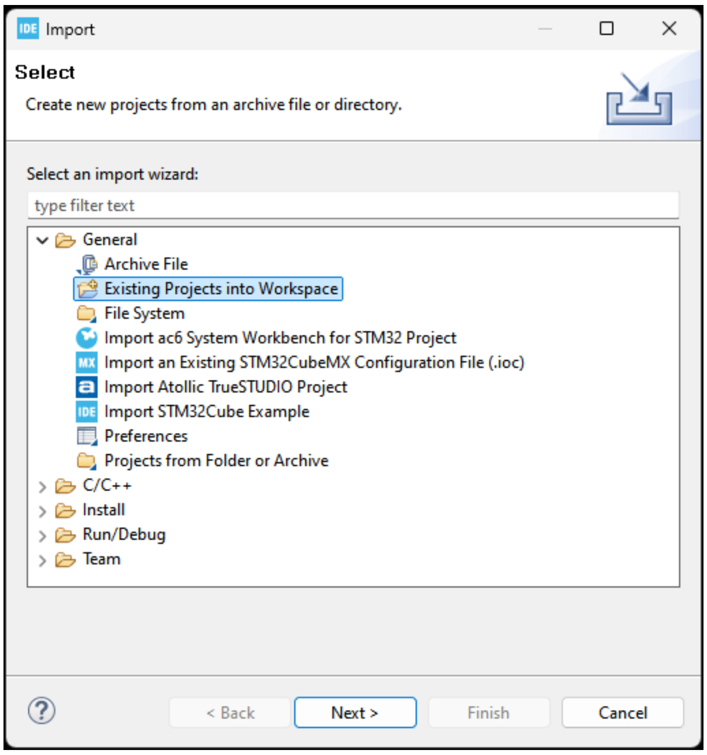


There are no projects in your workspace.
To add a project:
- Create a new Makefile project in a directory containing existing code
- Create a new C or C++ project
- Create a New STM32 project
- Create a New STM32 Project from an Existing STM32CubeMX Configuration File (.ioc)
- Creates a new STM32 CMake Project
- Create a project...
- Import projects...

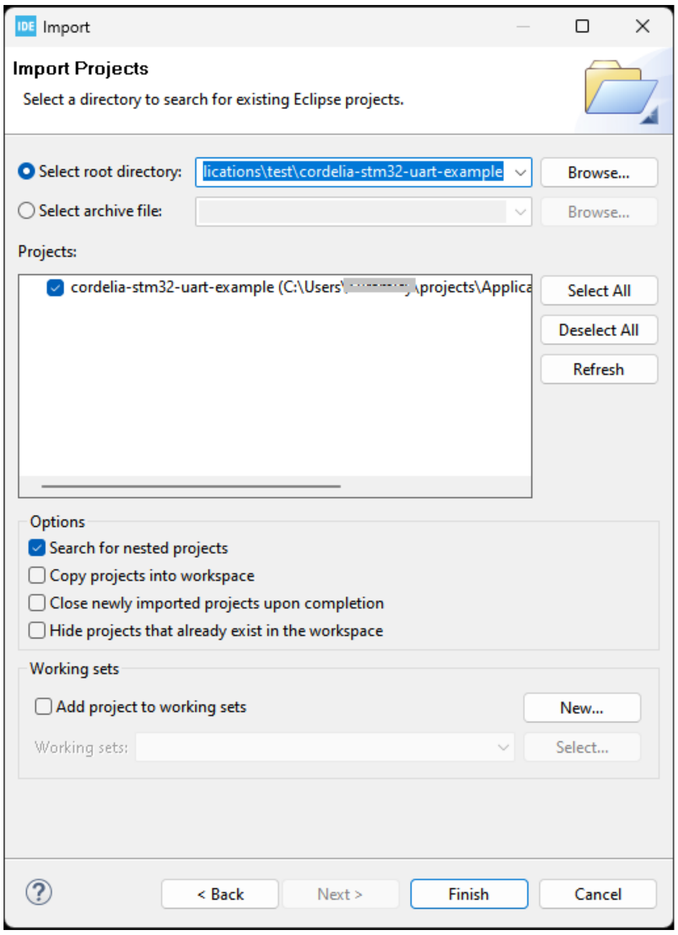
There is no active editor that provides an outline.

No consoles to display at this time.

Then expand General -> Existing Projects into Workspace


Import
Select
Create new projects from an archive file or directory.
Select an import wizard:
type filter text
General
Archive File
Existing Projects into Workspace
File System
Import ac6 System Workbench for STM32 Project
Import an Existing STM32CubeMX Configuration File (.ioc)
Import Atollic TrueSTUDIO Project
Import STM32Cube Example
Preferences
Projects from Folder or Archive
C/C++
Install
Run/Debug
Team
< Back
Next >
Finish
Cancel

Hit ‘Browse’ next to ‘Select root directory’ and navigate to your recently cloned GitHub project. You Should see the project name – ‘cordelia-stm32-uart-example’ project automatically detected.


Import

Import Projects

Select a directory to search for existing Eclipse projects.

(●) Select root directory: lications\test\cordelia-stm32-uart-example
( ) Select archive file:

Projects:

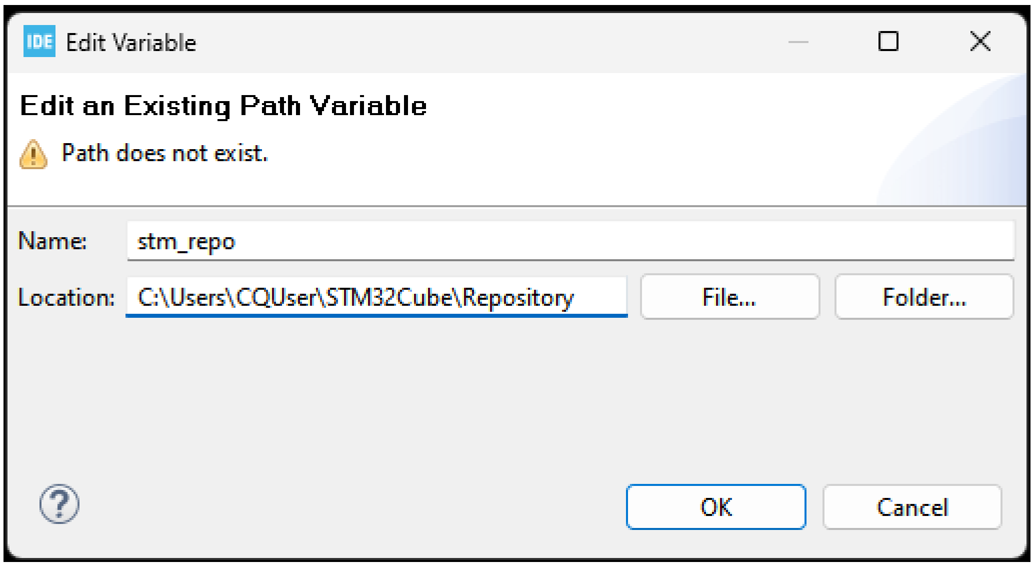
[✓] cordelia-stm32-uart-example (C:\Users\[redacted]\projects\Applica

Options
[✓] Search for nested projects
[ ] Copy projects into workspace
[ ] Close newly imported projects upon completion
[ ] Hide projects that already exist in the workspace

Working sets
[ ] Add project to working sets
Working sets:

Now before we can successfully build the project, we need to check some environment variables to ensure they are set correctly for your STMCubeIDE installation.

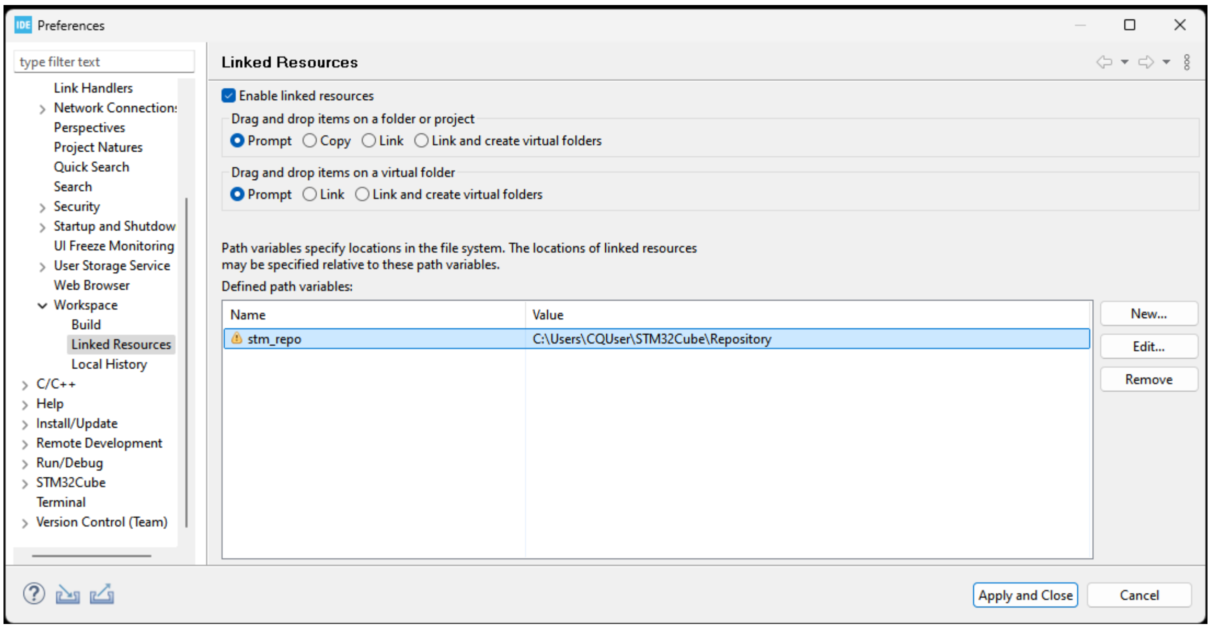
Navigate to the main menu bar – Window -> Preferences
Expand 'General' -> 'Workspace` -> 'Linked Resources'
Hit 'new' to add a new path variable.
You should name the new variable 'stm_repo'
Hit folder and navigate to your STMCube Repository folder (for a new installation you may need to create a folder).


Edit Variable

Edit an Existing Path Variable

⚠ Path does not exist.

Name: stm_repo
Location: C:\Users\CQUser\STM32Cube\Repository


Preferences

Linked Resources

☑ Enable linked resources

Drag and drop items on a folder or project: [✓ Prompt] Copy Link Link and create virtual folders

Drag and drop items on a virtual folder: [✓ Prompt] Link Link and create virtual folders

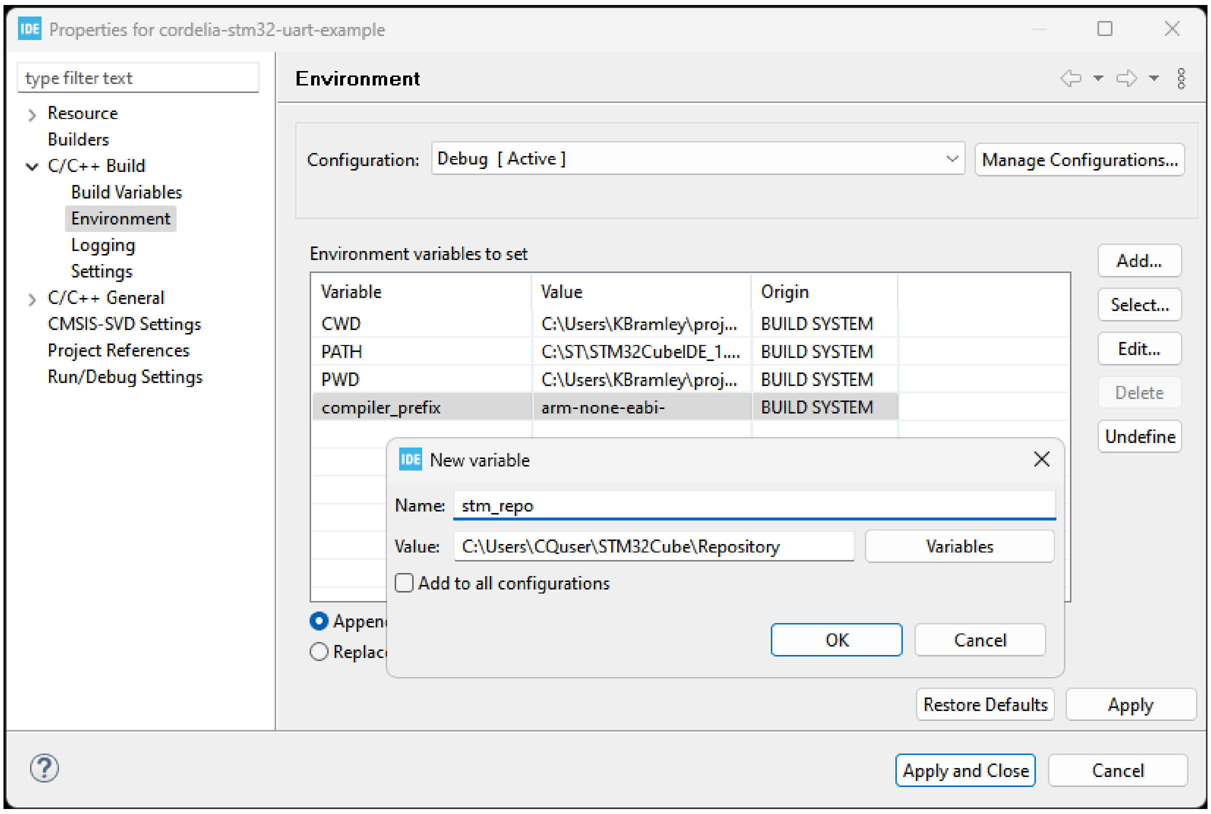
Path variables specify locations in the file system. The locations of linked resources may be specified relative to these path variables.

Defined path variables:

Name	Value
stm_repo	C:\Users\CQUser\STM32Cube\Repository

Click ‘Apply and Close’

Ensure the project is selected.
Right click and select ‘properties’
Expand C/C++ Build - > Environment
If stm_repo is NOT already added, Select Add


Properties for cordelia-stm32-uart-example

Environment

Configuration: Debug [Active]

Environment variables to set

Variable	Value	Origin
CWD	C:\Users\KBramley\proj...	BUILD SYSTEM
PATH	C:\ST\STM32CubeIDE_1....	BUILD SYSTEM
PWD	C:\Users\KBramley\proj...	BUILD SYSTEM
compiler_prefix	arm-none-eabi-	BUILD SYSTEM

New variable

Name: stm_repo

Value: C:\Users\CQuser\STM32Cube\Repository

☐ Add to all configurations

You should name the new variable ‘stm_repo’ as we did before.

Within the ‘Value’ field, you should enter the path to your STMCubeIDE Repository folder again.

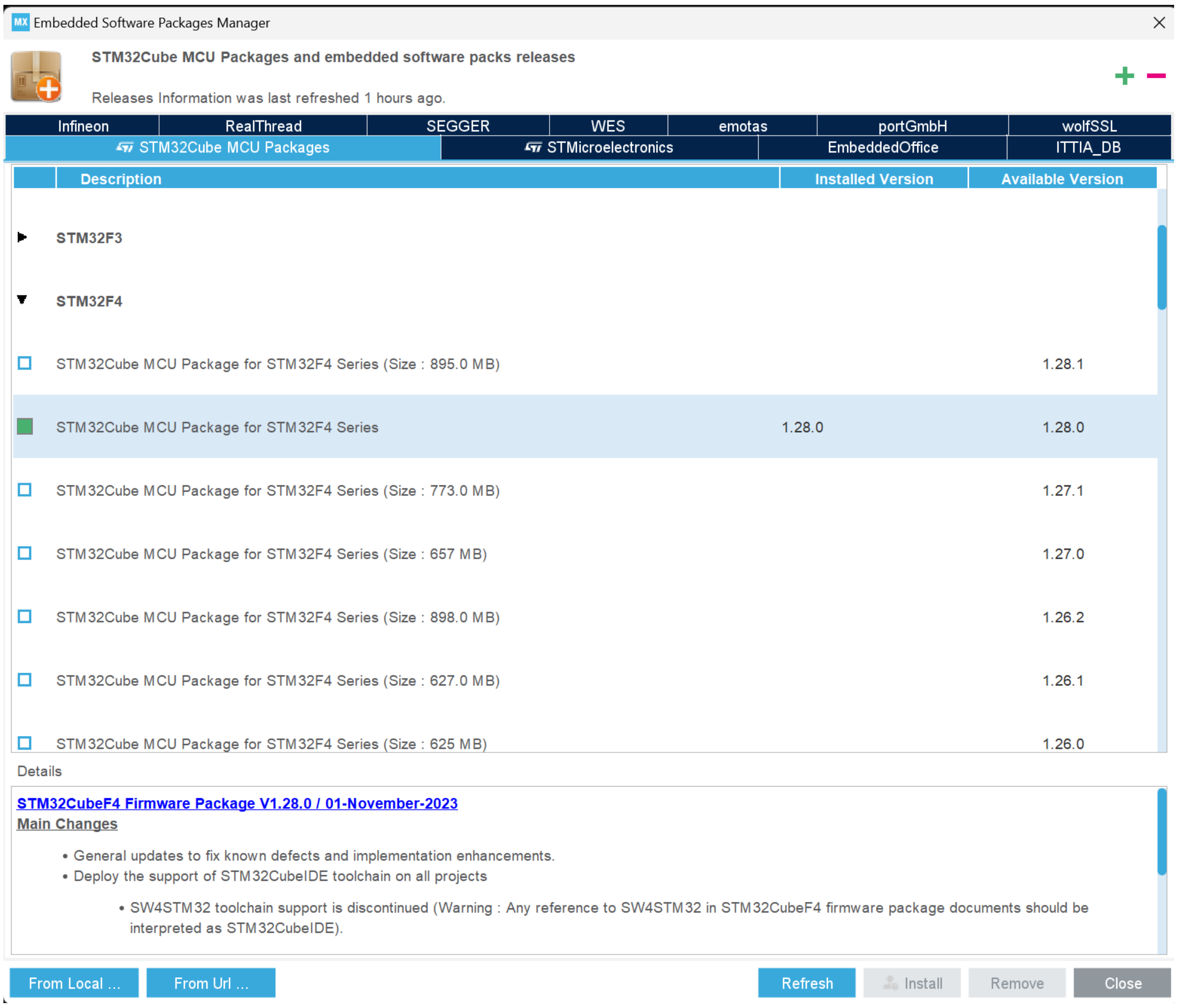
If, stm_repo is shown you should check the path is correct for your installation.

Click OK, then ‘Apply and Close’

Then we need to ensure that the BSP is available within your STM32CubeIDE installation. You should double click on the STM43F401RE_UART.ioc file to open the HW configuration panel.

Click “Software Packs” and select “Manage Software Packs”.

Navigate down to STM32F4 and select “STMCube MCU Package for STM32F4 Series” (this project uses V1.28.0) then hit install, you may need to log in to “myST” and accept the ST license agreement before the package is downloaded, unzipped and installed.


Embedded Software Packages Manager
STM32Cube MCU Packages and embedded software packs releases
Releases Information was last refreshed 1 hours ago.
Infineon
RealThread
SEGGER
WES
emotas
portGmbH
wolfSSL
STM32Cube MCU Packages
STMicroelectronics
EmbeddedOffice
ITTIA_DB

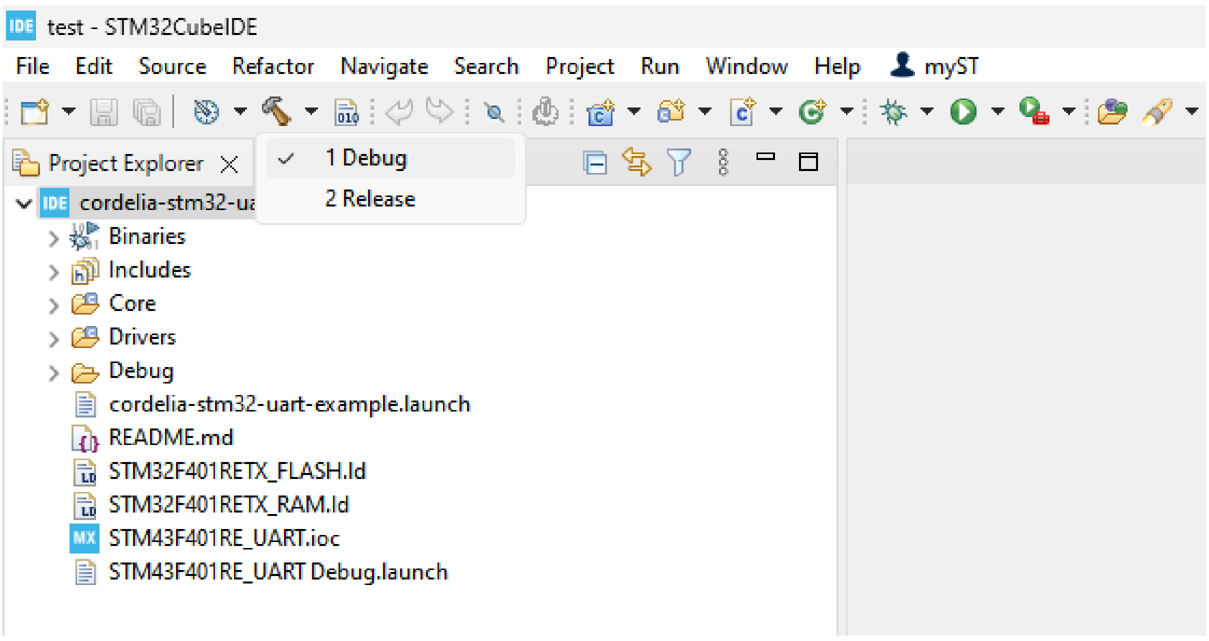
	Description	Installed Version	Available Version
▶	STM32F3		
▼	STM32F4		
☐	STM32Cube MCU Package for STM32F4 Series (Size : 895.0 MB)		1.28.1
🟩	STM32Cube MCU Package for STM32F4 Series	1.28.0	1.28.0
☐	STM32Cube MCU Package for STM32F4 Series (Size : 773.0 MB)		1.27.1
☐	STM32Cube MCU Package for STM32F4 Series (Size : 657 MB)		1.27.0
☐	STM32Cube MCU Package for STM32F4 Series (Size : 898.0 MB)		1.26.2
☐	STM32Cube MCU Package for STM32F4 Series (Size : 627.0 MB)		1.26.1
☐	STM32Cube MCU Package for STM32F4 Series (Size : 625 MB)		1.26.0

Details

STM32CubeF4 Firmware Package V1.28.0 / 01-November-2023
Main Changes

- General updates to fix known defects and implementation enhancements.
- Deploy the support of STM32CubeIDE toolchain on all projects
    - SW4STM32 toolchain support is discontinued (Warning : Any reference to SW4STM32 in STM32CubeF4 firmware package documents should be interpreted as STM32CubeIDE).

Now you should be able to successfully build the project.
Select the ‘hammer’ icon on the tool bar or navigate using the menu bar – Project -> Build Project

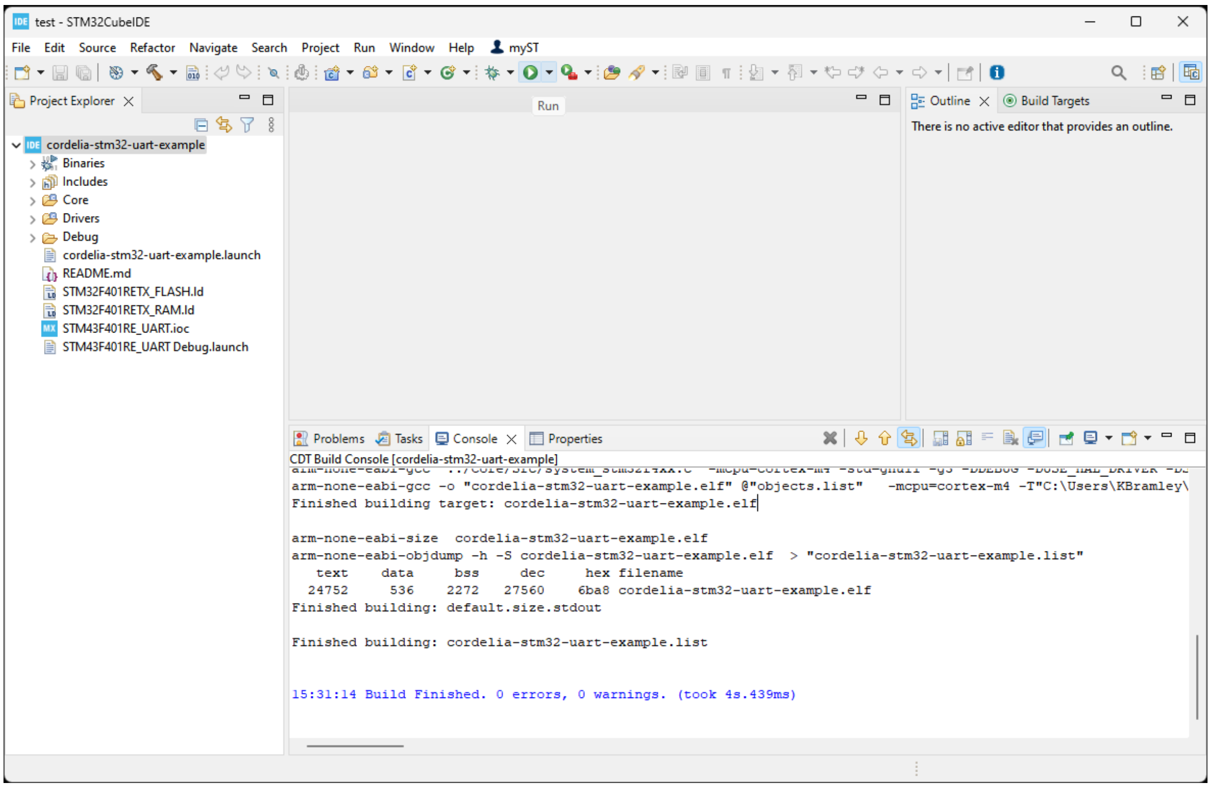


All being well, you should see the Console window build progress, ending with a completed build.

```
Finished building target: cordelia-stm32-uart-example.elf

arm-none-eabi-size  cordelia-stm32-uart-example.elf
arm-none-eabi-objdump -h -S cordelia-stm32-uart-example.elf  > "cordelia-
stm32-uart-example.list"
   text         data          bss          dec          hex      filename
  24752          536         2272        27560         6ba8      cordelia-stm32-
uart-example.elf
Finished building: default.size.stdout

Finished building: cordelia-stm32-uart-example.list

14:50:59 Build Finished. 0 errors, 0 warnings. (took 10s.547ms)
```

If you received any errors double check the new environment variable names and paths we just created.

You should ensure the board is connected the PC, you can now hit the ‘play’ icon to run the code – this will flash the project to your device.


test - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help myST
Project Explorer
cordelia-stm32-uart-example
Binaries
Includes
Core
Drivers
Debug
cordelia-stm32-uart-example.launch
README.md
STM32F401RETX_FLASH.ld
STM32F401RETX_RAM.ld
STM43F401RE_UART.ioc
STM43F401RE_UART Debug.launch
Run
Outline
Build Targets
There is no active editor that provides an outline.
Problems Tasks Console Properties

CDT Build Console [cordelia-stm32-uart-example]

```
arm-none-eabi-gcc  ../Core/Src/system_stm32f4xx.c  -mcpu=cortex-m4 -std=gnu11 -g3 -DDEBUG -DUSE_HAL_DRIVER -D
arm-none-eabi-gcc -o "cordelia-stm32-uart-example.elf" @"objects.list"   -mcpu=cortex-m4 -T"C:\Users\KBramley\
Finished building target: cordelia-stm32-uart-example.elf

arm-none-eabi-size  cordelia-stm32-uart-example.elf
arm-none-eabi-objdump -h -S cordelia-stm32-uart-example.elf  > "cordelia-stm32-uart-example.list"
   text    data     bss     dec     hex filename
  24752     536    2272   27560    6ba8 cordelia-stm32-uart-example.elf
Finished building: default.size.stdout

Finished building: cordelia-stm32-uart-example.list


15:31:14 Build Finished. 0 errors, 0 warnings. (took 4s.439ms)
```

For further guidance on using STM32CubeIDE please consult with STM and their documentation.

6 Cordelia-I Pre-Requisites

Prior to running this example application it is necessary to provision the Cordelia-I board. Provisioning the board is the process whereby the QuarkLink Instance, that the user has created (see section 2.1.1), connects directly to the Cordelia-I board, through a USB connection, and programs the Cordelia-I board with the credentials of the QuarkLink Instance.

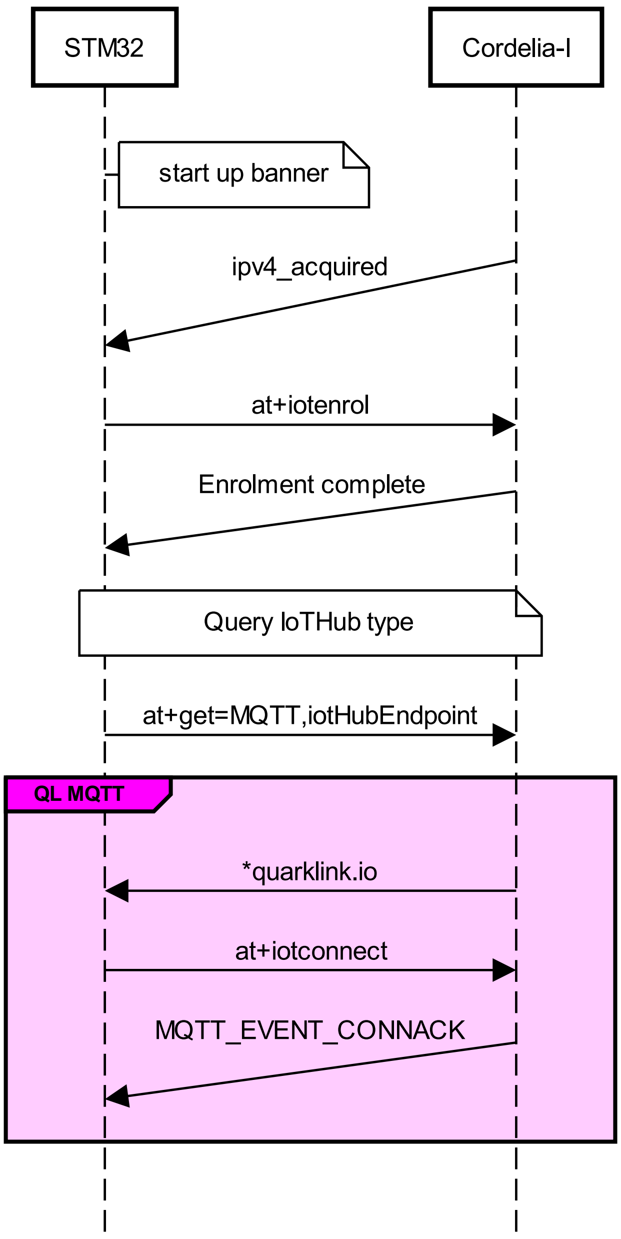
Full details of the provisioning process can be found in the ***QuarkLink Ignite Würth Cordelia Getting Started Guide*** (see section 2.1.3).

7 STM32F401 and Cordelia-I Interactions

This section shows how the STM32 Nucleo board interacts with the Cordelia-I WiFi module once the software is running on the STM32 Nucleo.

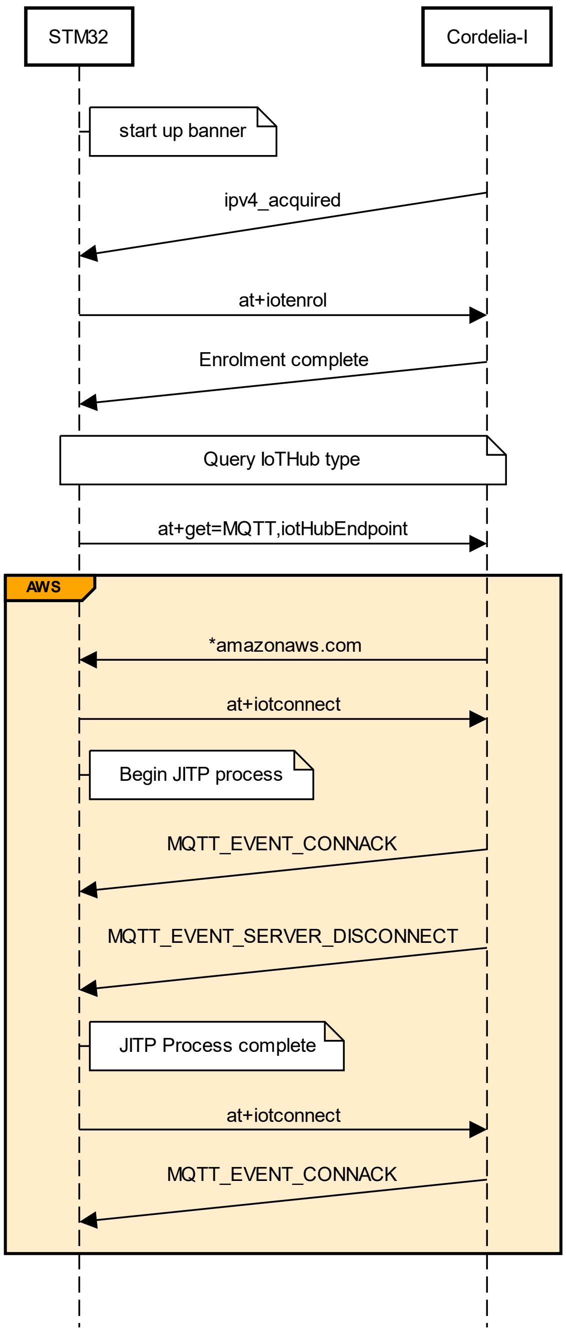
7.1 QL MQTT IoTHub

This section shows the flow of the STM32 Nucleo connecting to the MQTT hub that is available with the QuarkLink instance. The Cordelia-I first enrols onto the QuarkLink to receive the MQTT Hub credentials. Once they have been received the STM32 Nucleo connects directly to the MQTT broker of QuarkLink and can publish/subscribe.



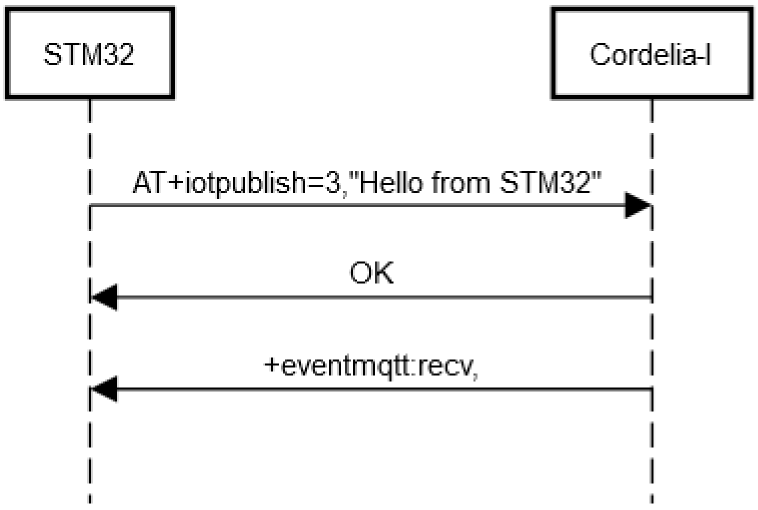
7.2 AWS IoTHub

This section shows the flow of the STM32 Nucleo connecting to an Amazon AWS MQTT hub, the credentials of which, have been configured, by the user, in the QuarkLink instance. The Cordelia-I first enrols onto the QuarkLink to receive the AWS MQTT Hub credentials and account information. Once they have been received the STM32 Nucleo connects directly to the AWS MQTT broker and can publish/subscribe.



7.3 Button press message sequence chart

When the user presses the button on the STM32 Nucleo, a publish message is sent and then subsequently received as we are also subscribed to the same topic.



8 Software description

The STM32 AT UART command example application utilises the UART, looking for command phrase and responses from Cordelia-I and contains a simple flag structure to track the status.

```
// Array of supported AT commands
AT_Responses commands[] = {
	{ "OK", handleATOK },
	{ "ERROR:", handleATError },
	{ "ipv4_acquired", handleATipv4 },
	// WiFi connected
	{ "eventwlan:disconnect",handleATWiFidisconnected },
	// WiFi disconnected
	{ "HTTP Client error:", handleATWiFidisconnected },
	// HTTP Error - reconnect
	{ "MQTT_EVENT_CONNACK", handleATmqtt },
	// MQTT connected
	{ "MQTT_EVENT_SERVER_DISCONNECT", handleATmqttDisconnect },
	// MQTT Disconnected
	{ "Device is revoked", handleATrevoked },
	// Device Revoked
	{ "+eventmqtt:recv,", handleATsub },
	// MQTT message received from a publisher
	{ "Enrolment complete", handleEnrolment },
	// IoTHubEnrolment
	{ "amazonaws.com", handleAWSEnrolment },
	// AWS IoTHub type
	{ "quarklink.io", handleMQTTEnrolment },
	// MQTT IoTHub type
};

// state flags
typedef struct {
	int wlan;			// WiFi Connected
	int enroled;			// Device enrolled
	int connected;			// IoTHub Connected
	int wlan_in_progress;		// WiFi details sent, awaiting IP address
	int recv;			// flag to catch the birth MQTT message
	int aws;			// AWS IoTHub type
	int mqtt;			// QL MQTT IoTHub type
} CordeliaStatus;
```

Once a command phrase is received the corresponding handler function is called. Within the handler functions we check the status and issue the next command.

The message flow is shown in Section 7.

The code contains some INFO_xxx and DEBUG_xxx macros that the user can simply enable or disable to aid debugging, see CQ_Debug.h. Care should be taken not to print too long or many messages as this can cause the STM32 to miss some response messages. Putting the messages before the AT command is issued to Cordelia-I can help.

At power up the Cordelia-I module will send a "`ipv4_acquired`" message, if successfully provisioned via QuarkLink and connected to WiFi. Cordelia-I can be considered connected to the WiFi router once it receives an IP address. Cordelia-I will retry the connection if it's dropped.

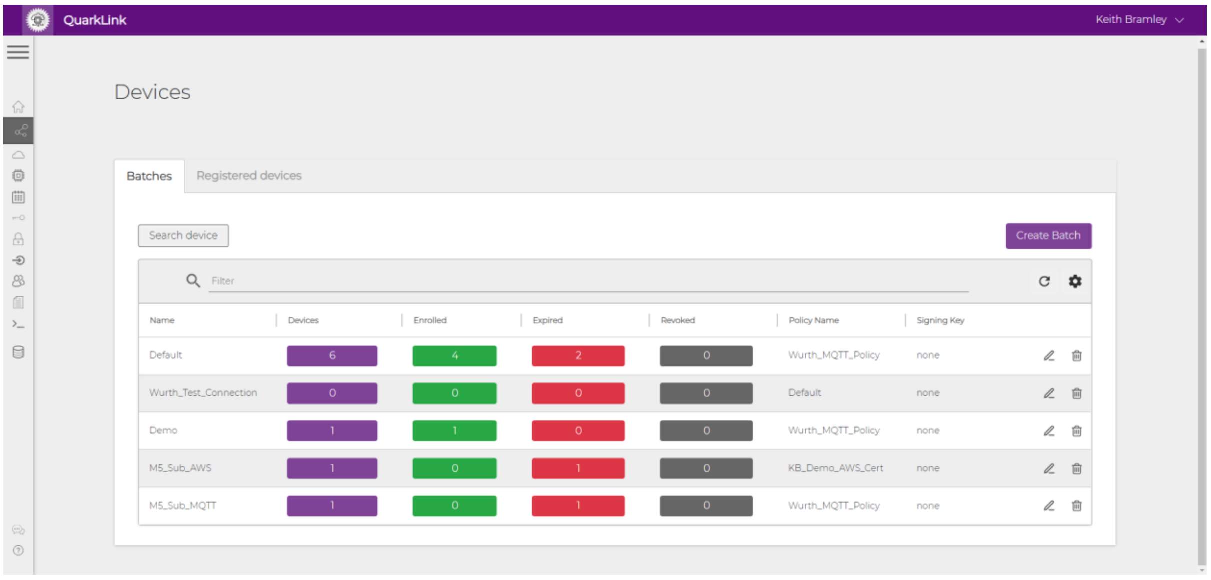
The `handleATipv4` function sets the `status.wlan` status flag, the WiFi LED indicator to true and sends an enrol request "`AT+iot=enrol`"

Assuming the enrolment was successful, we will receive an "`Enrolment complete`" message. This message triggers the `handleEnrolment` function which sets the `status.enroled` status bit.

Now we need to query the IoTHub type as the connection process differs for QL MQTT and AWS.

We issue an "at+get=MQTT,iotHubEndpoint" command from the QueryHub function to obtain the IoTHub details, we then set the AWS or MQTT status flag based on the response.

For QL MQTT based IoThub a single "`AT+iot=mqttConnect`" is sufficient, but for AWS the first "`AT+iot=mqttConnect`" is needed to begin the JITP process. Consult the AWS documentation for further details on the JITP process. We then issue a second "`AT+iot=mqttConnect`" which will allow Cordelia to connect successfully.



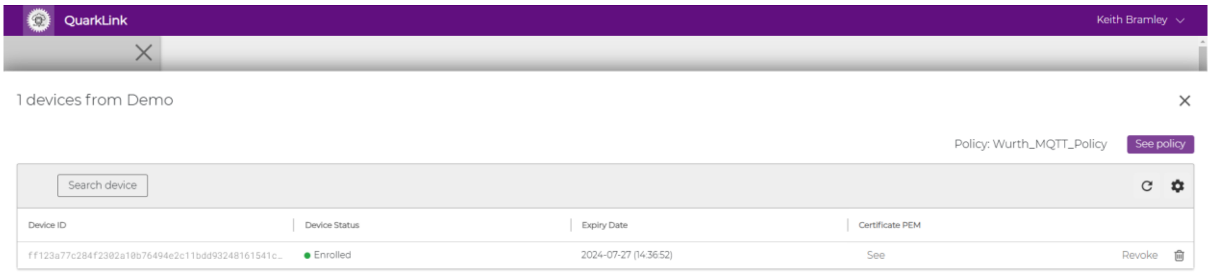
Name	Devices	Enrolled	Expired	Revoked	Policy Name	Signing Key
Default	6	4	2	0	Wurth_MQTT_Policy	none
Wurth_Test_Connection	0	0	0	0	Default	none
Demo	1	1	0	0	Wurth_MQTT_Policy	none
M5_Sub_AWS	1	0	1	0	KB_Demo_AWS_Cert	none
M5_Sub_MQTT	1	0	1	0	Wurth_MQTT_Policy	none

Cordelia-I will issue an "`MQTT_EVENT_CONNACK`" message once successfully connected, triggering the `handleATmqtt` function.
`handleATmqtt` sets the `status.connected` status bit and sets the LED indicator to true.

Upon successful connection to the MQTT broker the device will publish a birth type message so you will receive a subscribed message – "`Hello from STM32`"

8.1 Device Revocation

It is possible to revoke a device in QuarkLink and this is supported behaviour in this example project – in the QuarkLink GUI head to the Devices tab – then hit the purple devices button on the associated Batch – here you will see a list of deviceID's – on the right of each deviceID is the Revoke button.



1 devices from Demo

Policy: Wurth_MQTT_Policy

Device ID	Device Status	Expiry Date	Certificate PEM	
ff123a77c284f2302a10b76494e2c11bdd93248161541c…	● Enrolled	2024-07-27 (14:36:52)	See	Revoke

QuarkLink will force a disconnect from the MQTT broker – whether the local QL MQTT or AWS broker is used.

Cordelia-I will issue a "`MQTT_EVENT_SERVER_DISCONNECT`" message that triggers the `handleATmqttDisconnect` function call.

In the **`handleATmqttDisconnect`** function we clear the `status.enroled` status bit and the IoTHub indicator LED then trigger a enrol request "`AT+iot=enrol\r\n`"

When the device attempts to enrol again we will receive the "`Device is revoked`" message. `handleATrevoked` is called and we simply wait for the user to press the button to try to enrol again.

Once the user has set the re-enrol request within QuarkLink the enrolment will succeed.

Note : We need to try to re-enrol otherwise we just get disconnected again without a reason.

8.2 WiFi or IoTHub Disconnection

WiFi or IoTHub disconnection is handled within the same `handleATWiFidisconnected` function call following a "`eventwlan:disconnect`" or "`HTTP Client error:`" message.

handleATWiFidisconnected will clear the WiFi status flag. Cordelia-I will attempt to reconnect so once WiFi is available again we will receive the "ipv4_acquired" message.

9 Summary

As you can see, using Würth Cordelia-I with QuarkLink enables the user to quickly and easily add truly secure cloud based connectivity, only requiring 2 AT commands – AT+iotenrol and AT+iotconnect once the Cordelia-I board has been securely provisioned. A simple "AT+iotpublish=1,\"Hello from button press\"\r\n" is then needed to push your messages to the cloud.

10 Revision History

CQ Cordelia-I STM32 AT Command Demo Application Note

Rev.	Date	Owner	Description
1.00	Nov 2024	KB	Original document

CRYPTO
QUANTIQUE

United Kingdom

Unit 304-5,
164-180 Union Street,
London
SE1 0LH

General contact email:
info@cryptoquantique.com

QUANTUM
DRIVEN
CYBERSECURITY

The most advanced security product for the Internet of Things in the world