AN1389: Running Zigbee Host Applications in a Docker Container

The updated application structure in Zigbee EmberZNet 7.0 and higher no longer supports compiling host applications in MinGW for Windows. This document offers an alternative solution by using a Docker container to run the NCP Host Application.

KEY POINTS

- Example Configuration for NCP Host Application in Docker
- Host configuration solution for Windows users.

1 Introduction

Beginning with Zigbee EmberZNet 7.0, Silicon Labs introduced a new callback framework that automatically supplies *weak* callback functions to the stack, such as global-callback.c and zap-event.c. These *weak* functions ensure that the project builds when the callbacks are not explicitly implemented. This new framework replaces the 'callback-stub.c' file generated in EmberZNet 6.0 by AppBuilder. Overall, the new callback structure simplifies the user experience in application development but has been seen to cause errors in the linking stage in environments that do not support *weak* functions. More information on the new EmberZNet 7.0 callback framework can be found in *UG491: Zigbee Application Framework Developer's Guide for SDK 7.x and Higher.*

Specifically, the *weak* functions in EmberZNet 7.0 cause issues for Windows users when running applications in MinGW on Cygwin. MinGW treats *weak* function as "Null", causing undefined reference errors in the linking stages of the project build. This application note offers an alternative solution for Windows users that uses Docker to configure the host application. This solution is not required for Linux and MacOS operating systems, as Clang and GCC support *weak* functions. However, if you want to follow a similar procedure for Linux and MacOS, configuring a Docker host application will differ slightly from the procedure below. For more information contact Silicon Labs support through the support portal https://www.silabs.com/support.

This application note will walk through the configuration of the host application in a Docker container on a local Windows PC connected to an NCP-configured radio board via USB. To follow the procedure, you should have both Docker Desktop and Silink installed on your local PC. Silink is a Simplicity Studio tool that maps USB Ports to IP addresses, which allows Docker Containers running on Windows PCs to connect to the radio board.

Currently, this configuration does not allow Network Analyzer's network capture to run while the host and NCP applications are running. This is due to conflicts with Silink's port mapping. It is recommended to configure another radio board as a sniffer node and complete the network capture that way.

Requirements:

Docker (Version 20.10.0 or Higher) installed from https://www.docker.com/products/docker-desktop. This will install all dependencies needed for this procedure.

Simplicity Studio 5 installed from https://www.silabs.com/developers/simplicity-studio

Silink installed as part of Simplicity Studio 5 adapter pack tools (see section 2.3 Step 3: Install and Config Silink for instructions)

EmberZNet SDK 7.x or higher, installed as part of the Gecko SDK Suite (GSDK). See the Simplicity Studio v5 User's Guide for more information about installing the GSDK.

Radio Board installed on a development mainboard (BRD4162A is used in this example)

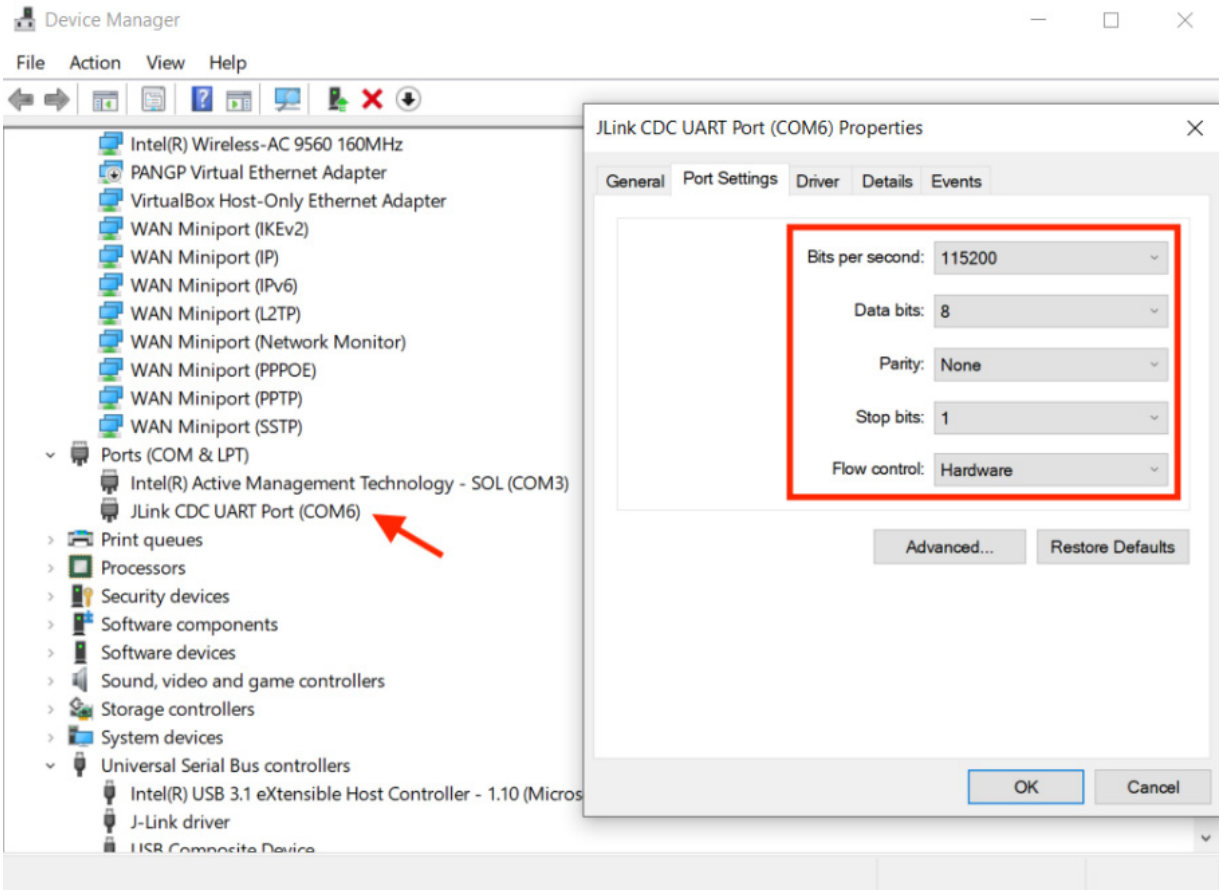
2 Running a Host Application in a Docker Container

2.1 Step 1: Create and Flash NCP-UART-HW Example Project onto BRD4162A

Create a new Simplicity Studio project for the target radio board (BRD4162A). Select the **ncp-uart-hw** example project. Build and flash the project onto BRD4162A. This will be the NCP-configured device. For more information on how to create projects in Simplicity Studio, refer to *QSG180: Zigbee EmberZNet Quick-Start Guide for SDK v7.0 and Higher*.

2.2 Step 2: Config COM Port

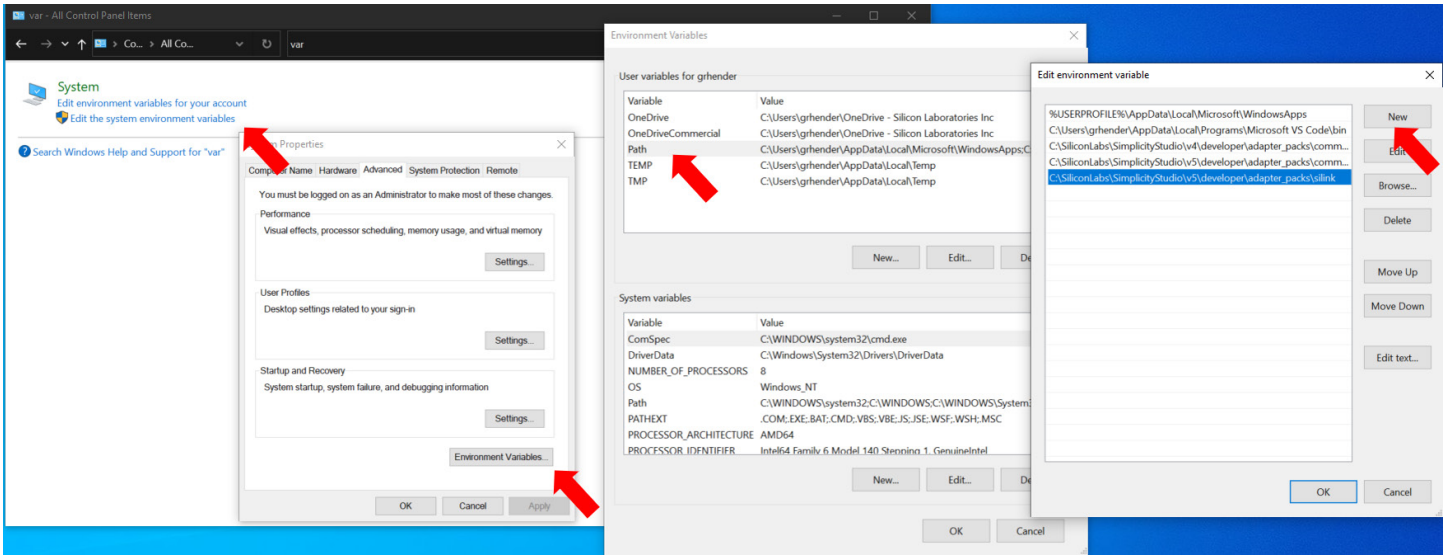
Make sure the radio board (BRD4162A) is connected to the host PC via USB. Open the Windows Device Manager and configure the "Port Settings" to match those used by the NCP application (115200 Baud rate with hardware flow control), as shown in the following figure.



2.3 Step 3: Install and Config Silink

To find Silink's installation path, navigate through the installation path of Simplicity Studios until you reach the adapter pack directory. If you used the default installation path, the file path should look like: C:/SiliconLabs/SimplicityStudio/v5/developer/adapter_packs/Silink.

Add the Silink file path to the Windows "PATH" environment variables. The following figure shows the steps required.



Run "Windows PowerShell" as Administrator and execute the following commands to map the USB ports to IP address ports.

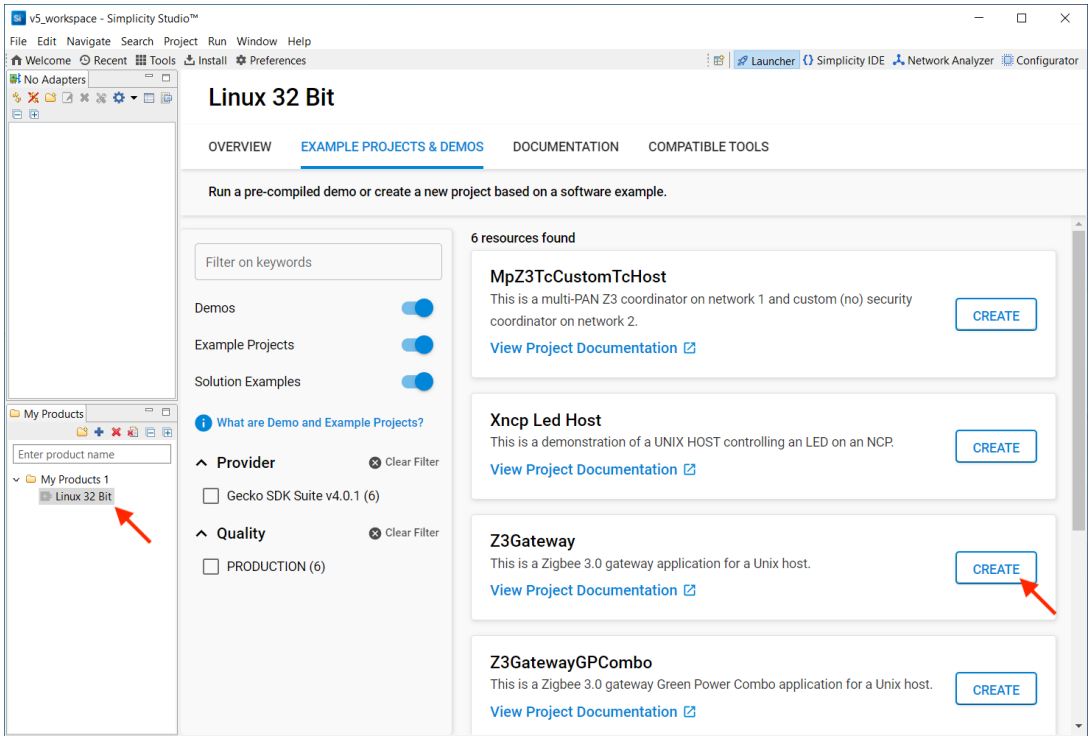
```
silink.exe -automap 4900
```

The following figure shows the desired output from the console.

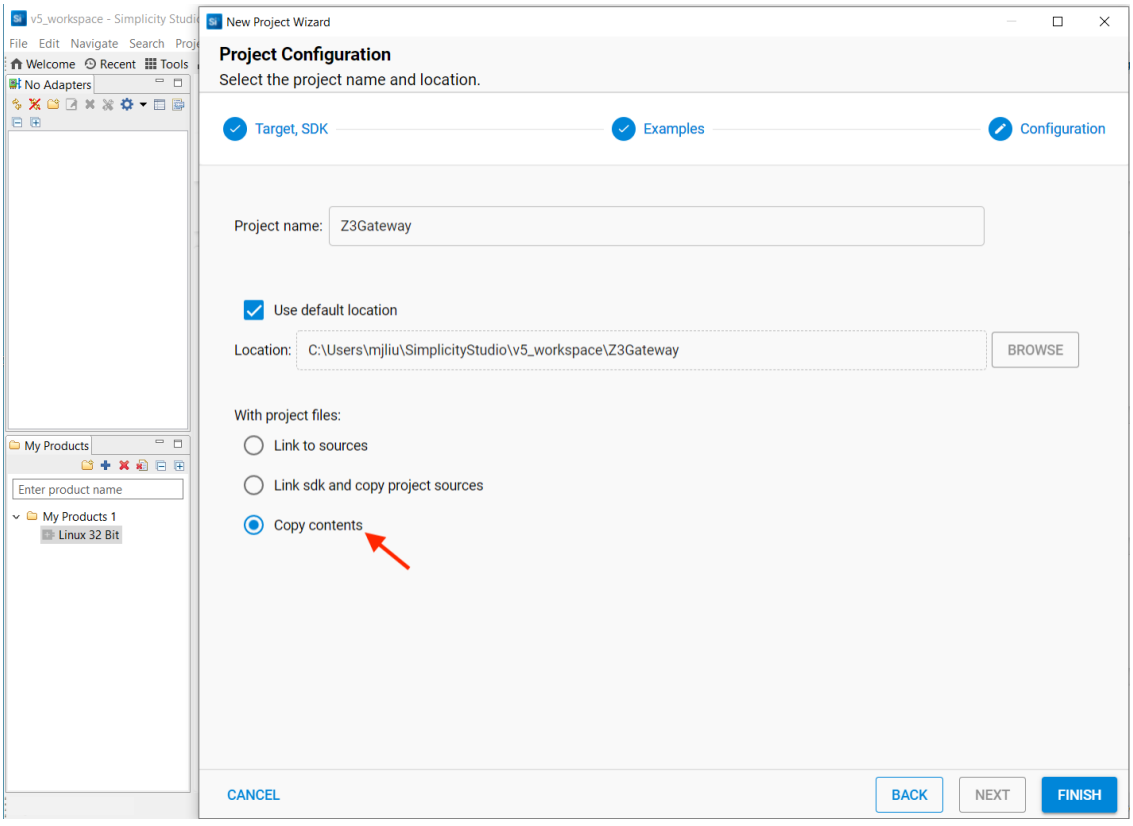
```
PS C:\WINDOWS\system32> silink.exe -automap 4900
Silink version: 0.5.2b238
VUART Target connection allowed.
USB Protocol: USBT
Mapping channel VUART0 to port 4900
Mapping channel VCOM0 to port 4901
Mapping channel ADM_CONSOLE to port 4902
Mapping channel DCH to port 4905
OEM:
Device family: Unknown
Serial number: 440187227
silink>
```

2.4 Step 4: Create a Host App for Linux 32 Bit

In the Simplicity Studio My Products view, select **Linux 32 Bit** as the target part and select the Z3 Gateway project, as shown in the following figure.



In the Project Configuration dialog, select **Copy Contents**, as shown in the following figure. This copies necessary files from the GSDK to the project folder, so the Docker Container does not need a complete GSDK.

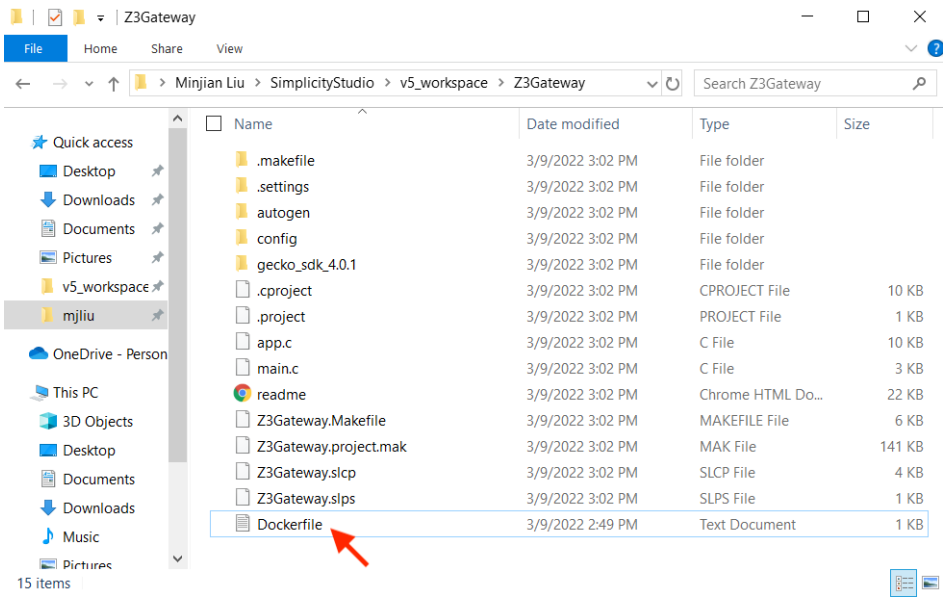


2.5 Step 5: Create a Dockerfile

Create a new file in the root project directory and name it "Dockerfile" without any file extension or suffix. Add the following commands:

```
FROM gcc
RUN apt-get update && apt-get -y install gcc-multilib socat
RUN echo "socat -d TCP:host.docker.internal:4901 pty,raw,echo=0,link=/dev/ttySilink &" >> ~/.bashrc
COPY . /usr/src/Z3Gateway
WORKDIR /usr/src/Z3Gateway
RUN make -f Z3Gateway.Makefile
```

This file configures the Docker container with a GCC environment and downloads the required packages to run the application. The following figure shows the Dockerfile location in the project structure of Z3 Gateway.

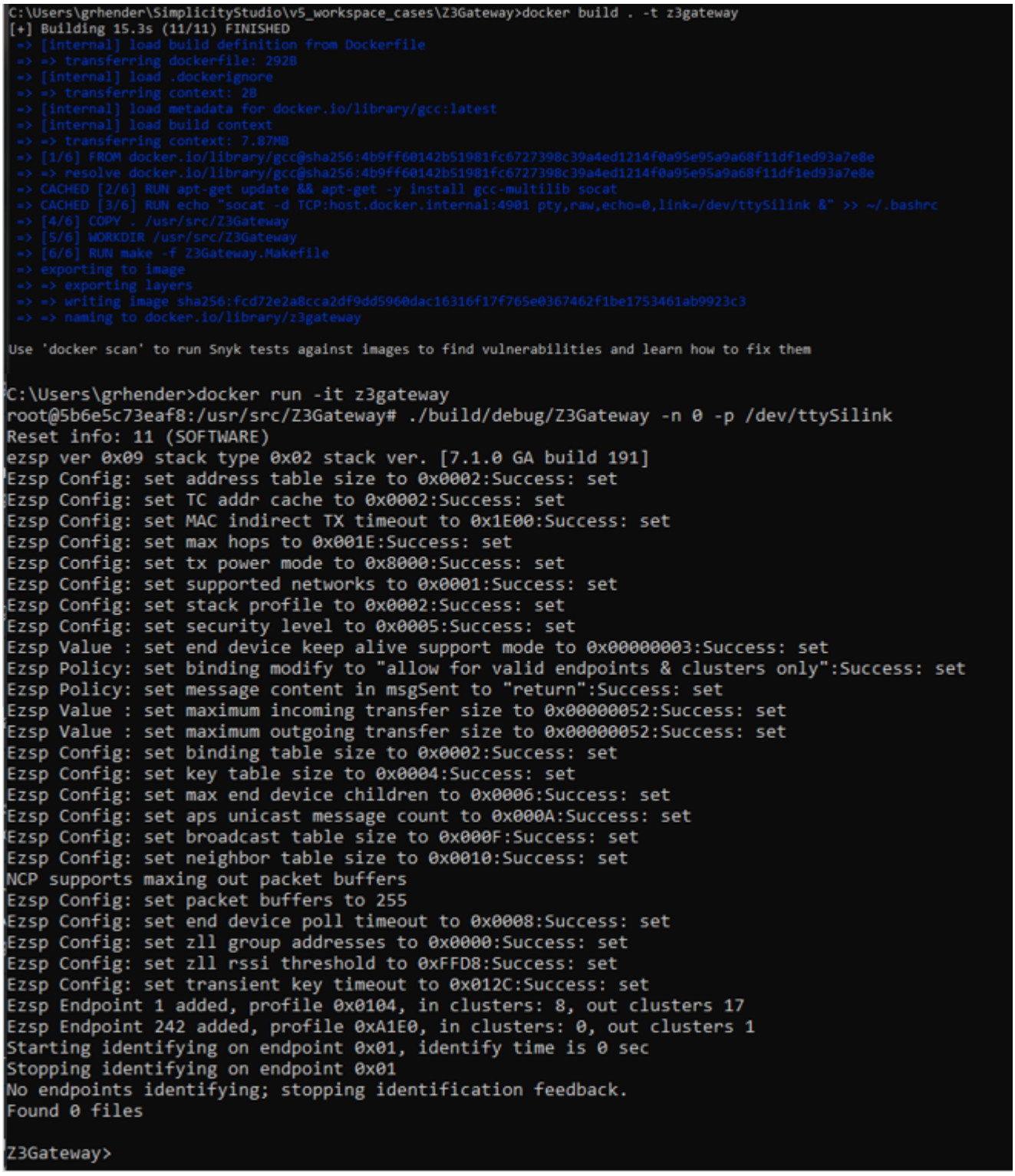


2.6 Step 6: Build and Run the Host App in the Container

Run the following commands on the console inside the project folder to build and start the container.

```
docker build . -t z3gateway
docker run -it z3gateway
./build/debug/Z3Gateway -n 0 -p /dev/ttySilink
```

The following figure shows the desired output of these commands.



After this step, the host application is up and running.

In order to stop the Docker container, enter the following commands into a new console. `Docker ps` lists all running containers on the local PC. `Docker stop` then stops whatever container is explicitly specified.

```
Docker ps
Docker stop <container_id>
```

```
C:\Users\grhender>docker ps
CONTAINER ID   IMAGE              COMMAND   CREATED              STATUS              PORTS     NAMES
e520d1a02776   z3gateway:latest   "bash"    About a minute ago   Up About a minute             flamboyant_satoshi

C:\Users\grhender>docker stop e520d1a02776
e520d1a02776

C:\Users\grhender>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

Alternatively, the Docker desktop application can run, stop, and pause the Docker containers on the local machine.

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com