Arm Virtual Hardware Examples

Version 1.0

Knowledge Base Article

Non-Confidential

Copyright © 2022 Arm Limited (or its affiliates).
All rights reserved.

Issue 01
107668_0100_01_en

Arm Virtual Hardware Examples

Knowledge Base Article

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.

Release information

Document history

Issue	Date	Confidentiality	Change
0100-01	21 September 2022	Non-Confidential	First release

Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws

and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm’s trademark usage guidelines at https://www.arm.com/company/policies/trademarks.

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Feedback

Arm® welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Contents

1. Capture Input from the Console....6
1.1 Retrieving Character Input from the User....6
1.2 Detecting When the User Presses Return....6
1.3 Modify How the Terminal Handles “newline”....6

2. iMX8m Arm Cortex Complex Examples....7
2.1 Blink the Led on the iMX8m Arm Cortex Complex device....7
2.1.1 Blink the LED....7

3. STM32U5 IoT Discovery Kit Examples....9
3.1 STM32CubeU5 TFM example....9
3.1.1 Creating your device....9
3.1.2 Running the example....14
3.1.3 Next steps....18
3.2 STM32CubeU5 WiFiBasics example....18

4. Raspberry Pi 4 Example....24
4.1 Raspberry Pi vMMIO example....24
4.1.1 Creating the VM....24
4.1.2 Writing to MMIO Without a vMMIO Handler Registered....28
4.1.3 Writing to MMIO With a vMMIO Handler Registered....29
4.1.4 Flow Diagram....30
4.1.5 Summary....31

5. API Examples....32
5.1 Python and Javascript API Examples....32
5.1.1 Python Examples....32
5.1.2 Javascript Examples....32
5.2 GitHub Actions Example - Building Firmware, Uploading, and Testing....33

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

1. Capture Input from the Console

Learn how to capture character and newline console input.

1.1 Retrieving Character Input from the User

It is possible to capture character input from the console using the scanf C library function.

For example, capture a character input using

```
scanf("%c", &input_char);
```

1.2 Detecting When the User Presses Return

Our UI console sends a new line (`\n`) when you hit return but doesn't send a carriage return (`\r`). When detecting a new line in code, use `\n` rather than `\r`.

For example the following will run, whereas using `\r` will hang:

```
while ( getchar() != '\n');
```

1.3 Modify How the Terminal Handles "newline"

Alternatively, you can adjust the terminal settings to not map CR to NL using the command `stty -icrnl`.

You can then send a `\r` by pressing Ctrl+M or `\n` by pressing Ctrl+J.

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

2. iMX8m Arm Cortex Complex Examples

iMX8m Arm Cortex Complex examples:

- Blink the Led on the iMX8m Arm Cortex Complex device

2.1 Blink the Led on the iMX8m Arm Cortex Complex device

This tutorial will demonstrate how to blink the LED in the iMX8m Arm Cortex Complex device. It follows from the Quickstart for iMX8m Arm Cortex Complex tutorial.

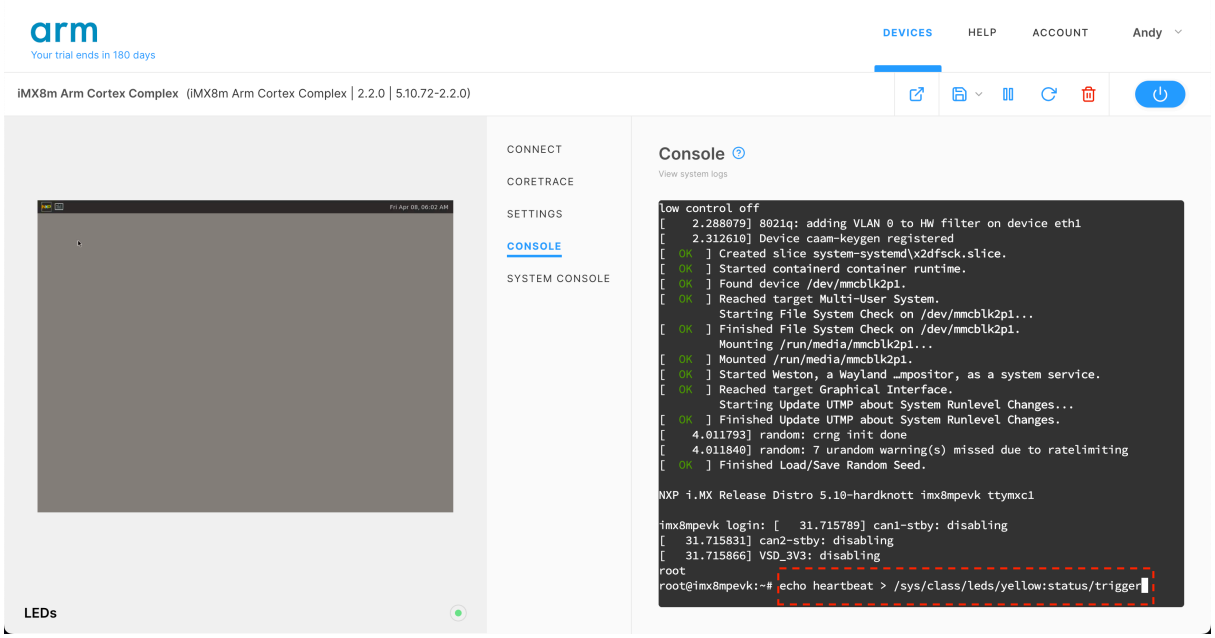
2.1.1 Blink the LED

To blink the LED, do the following:

1. Following on from the tutorial OR go to a iMX8m Arm Cortex Complex device that has the console logged in.
2. To blink the LED type `echo heartbeat > /sys/class/leds/yellow:status/trigger` into the console.

   The LED representation is at the bottom of the desktop. In the current state it is on and solid.

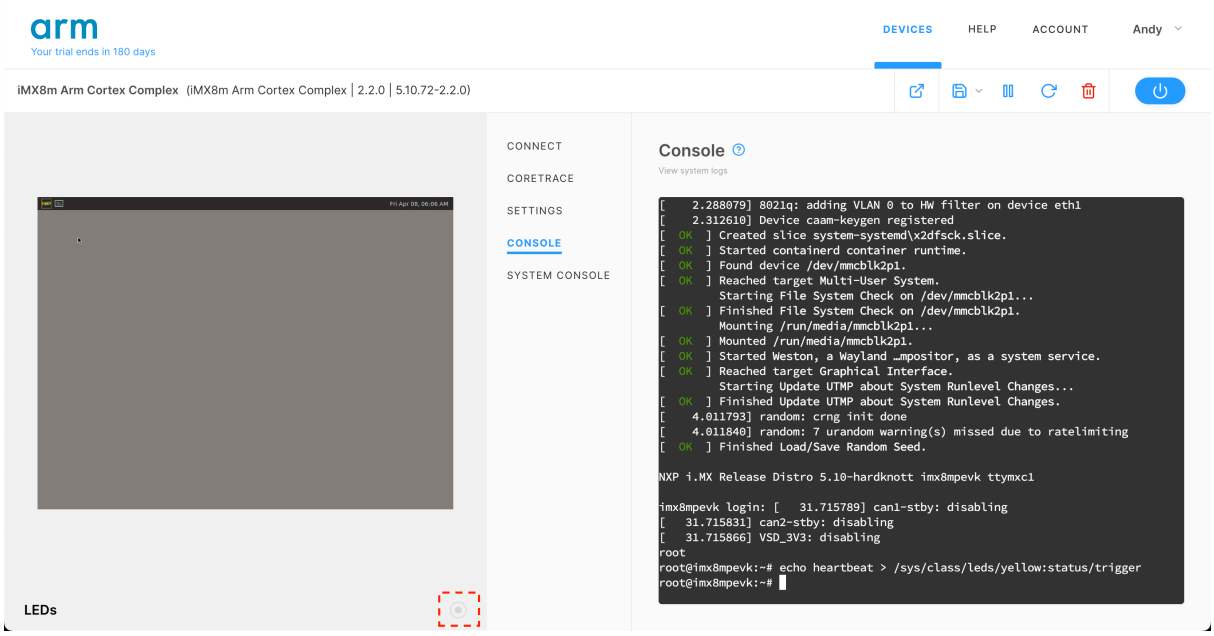
   Figure 2-1: LED on and solid



3. The LED will now be blinking.

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 2-2: LED blinking



Find out more

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

3. STM32U5 IoT Discovery Kit Examples

STM32U5 IoT Discovery Kit examples:

- STM32CubeU5 TFM example
- STM32CubeU5 WiFiBasics example

3.1 STM32CubeU5 TFM example

STM32U5 Trusted Firmware M tests firmware package example.

For this example the kernel component in the firmware package is itself a ZIP archive, containing various binaries and `load.txt` file describing their location in memory:

```
name:tfm_s_init.bin load:0xc024000
name:tfm_ns_init.bin load:0xc052000
name:loader.bin load:0xc1fa000
name:TFM_SBSFU_Boot.bin load:0xc002000
```

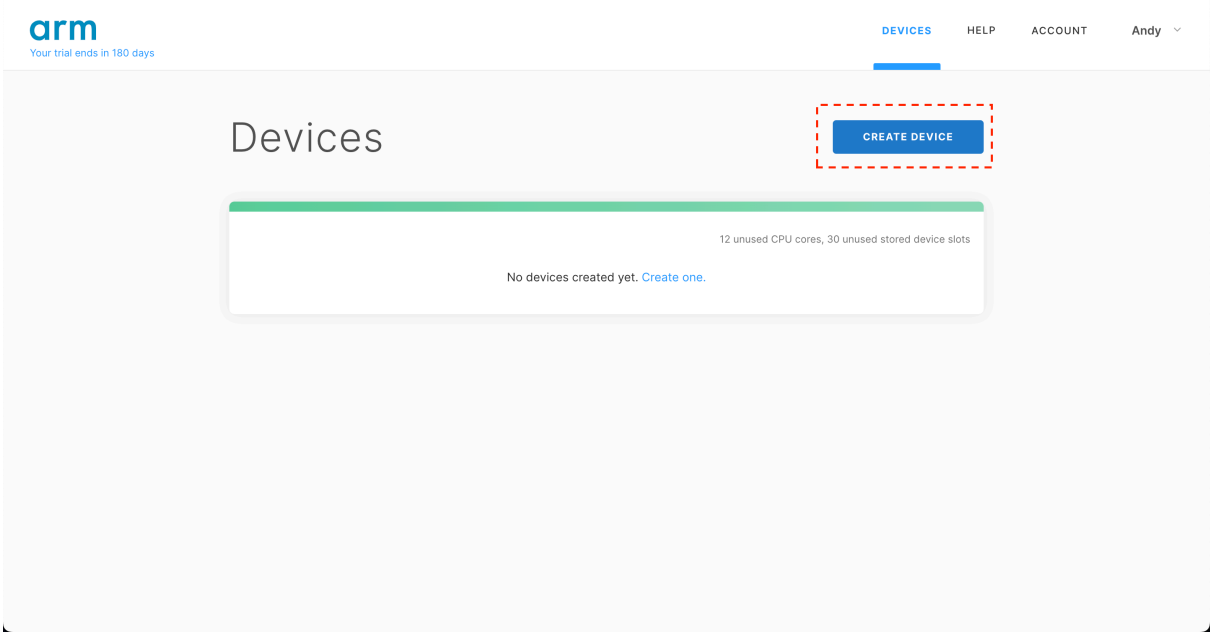
3.1.1 Creating your device

To create your device:

1. Log in to AVH with your Arm user account. If you do not have or need to create an Arm account, register at https://developer.arm.com/.
2. When logged in you land on the AVH Devices page. Click CREATE DEVICE.

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-1: Click Create Device



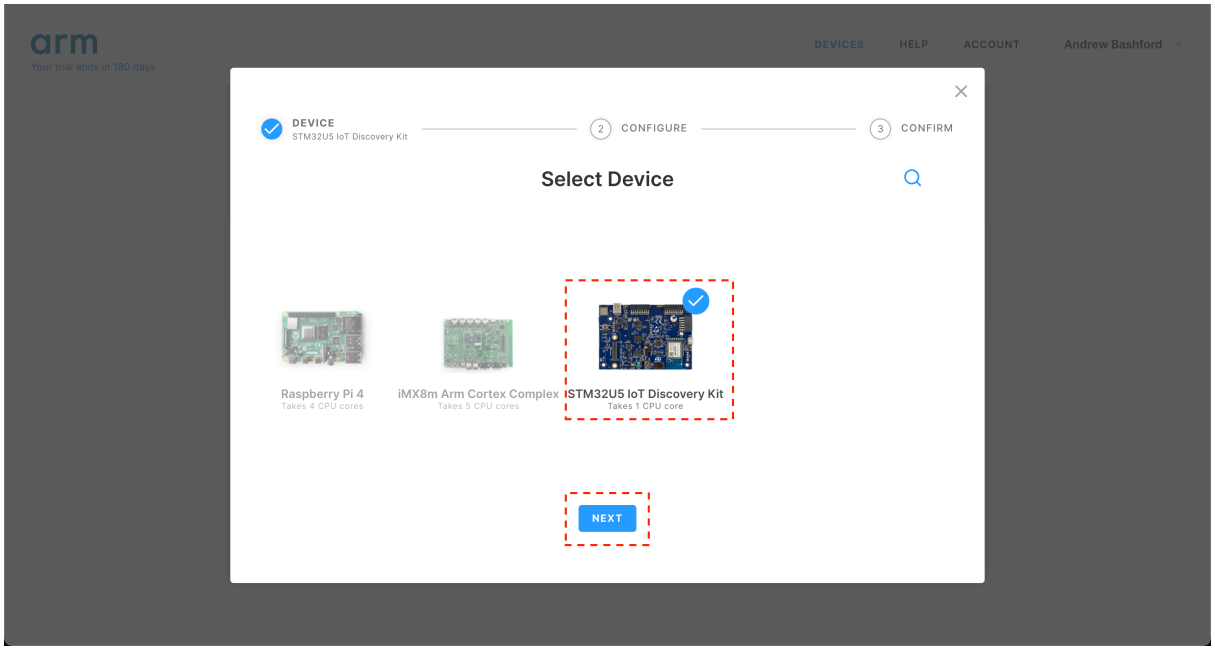
3. Click on the STM32U5 IoT Discovery Kit to select it. Then click NEXT.

Note

You require 1 free CPU core to build this device or it will be disabled.

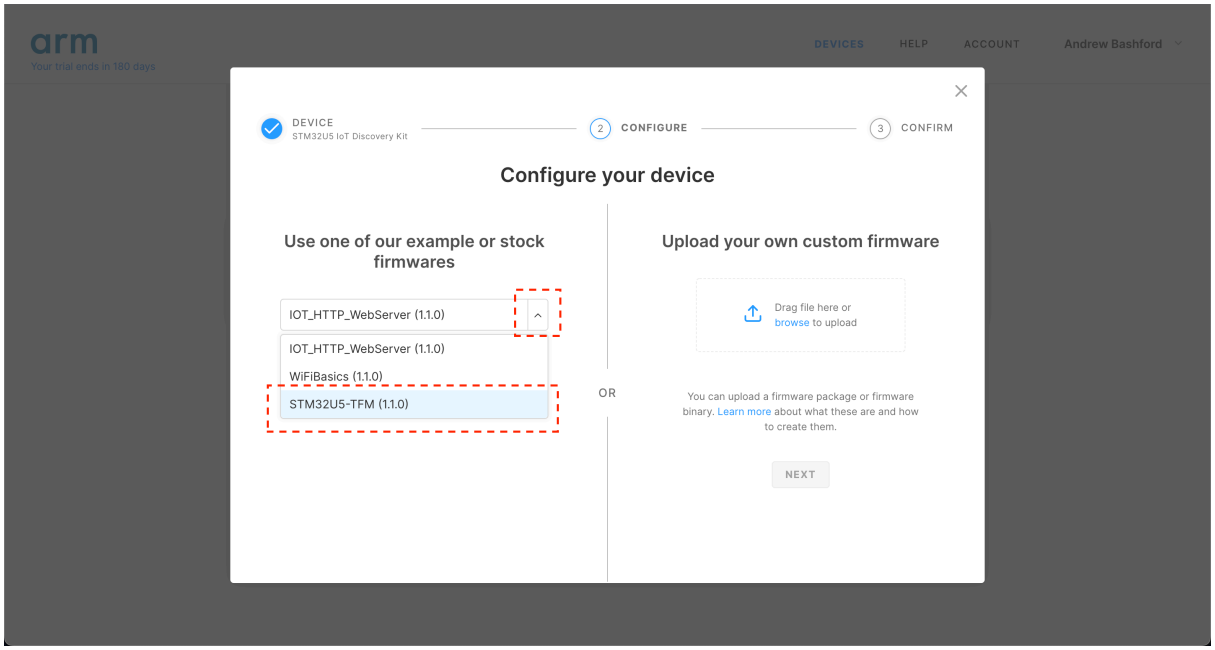
Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-2: Select device



4. In this next step, you'll see a list of pre-configured examples/stock firmware or you can upload your own custom firmware. In this example, we will use the selection `STM32U5-TFM (1.1.0)` in the stock firmware section.

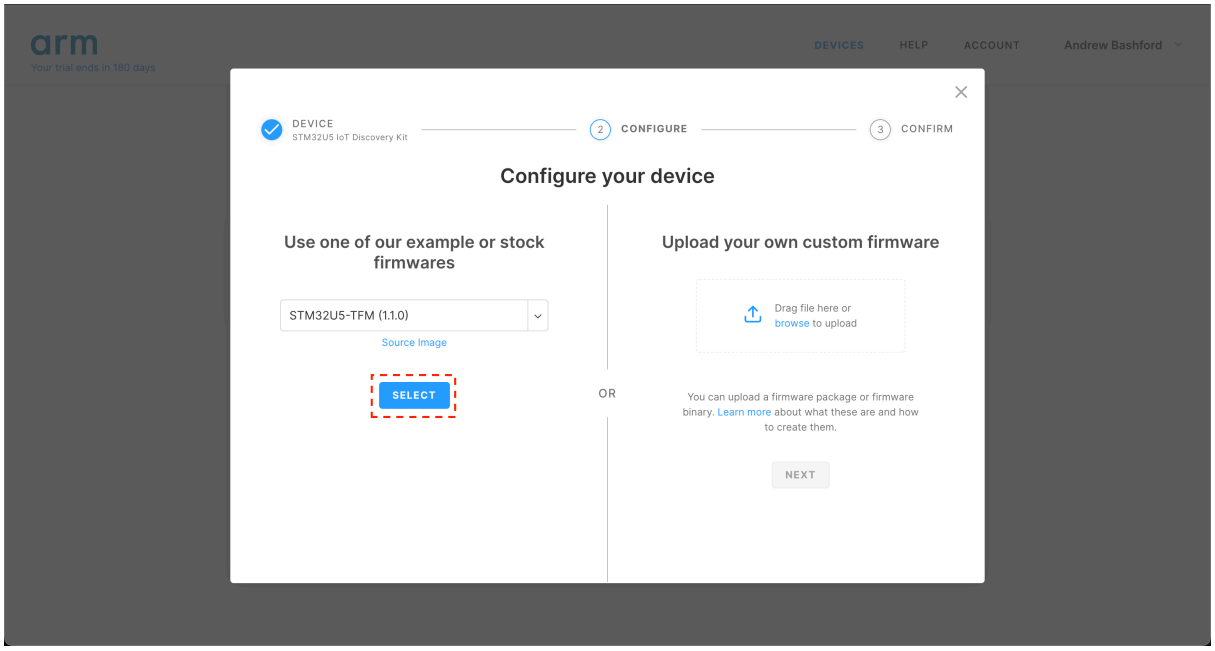
Figure 3-3: Configure your device



5. Click SELECT.

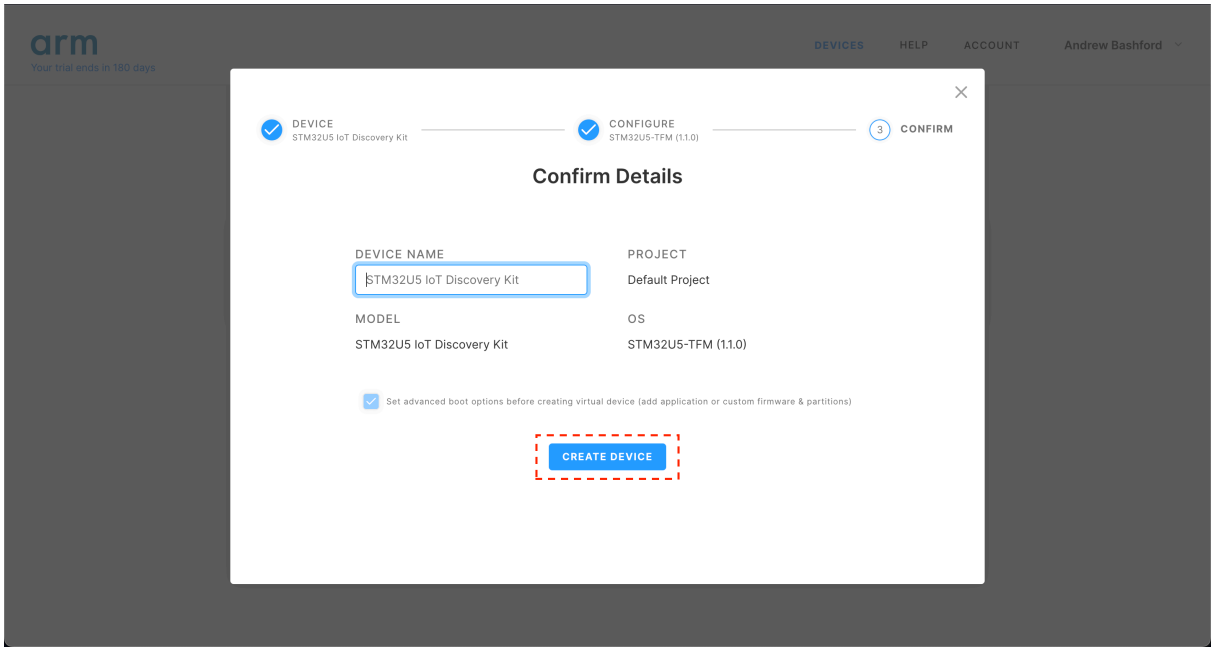
Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-4: Click Select



6. In the confirmation screen, click CREATE DEVICE using the default name and the advance boot options will be checked by default.

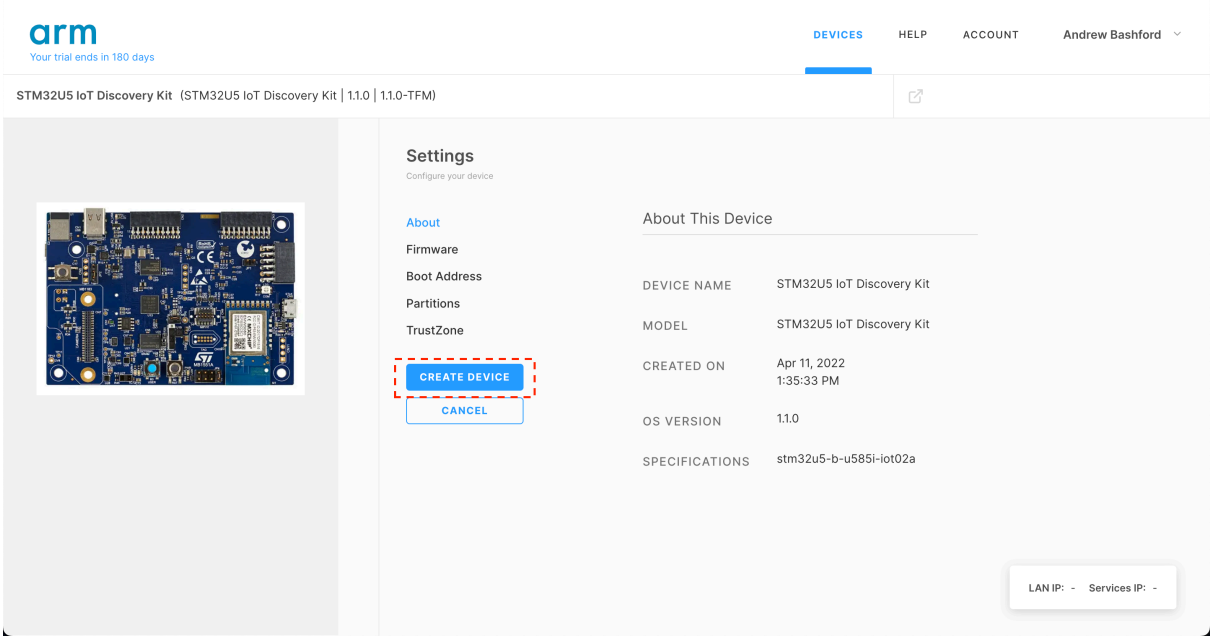
Figure 3-5: Confirm details



7. In the advanced boot options, we will just use the default. Click CREATE DEVICE.

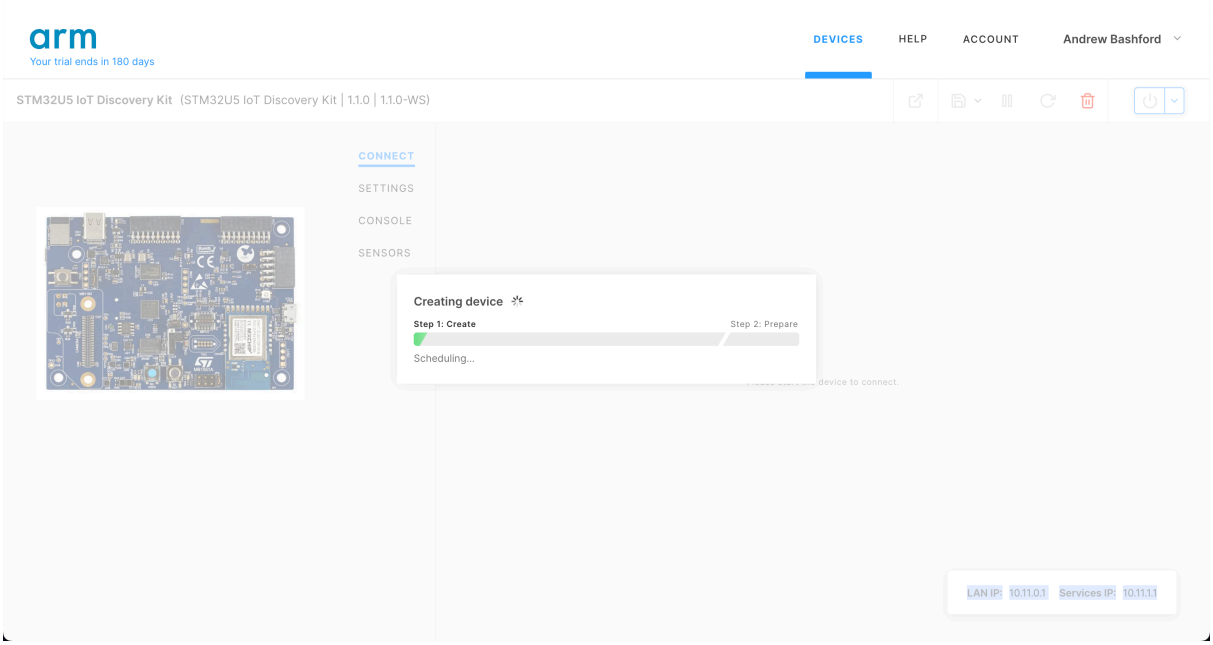
Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-6: Settings



8. The device will then be created. It takes a few moments…

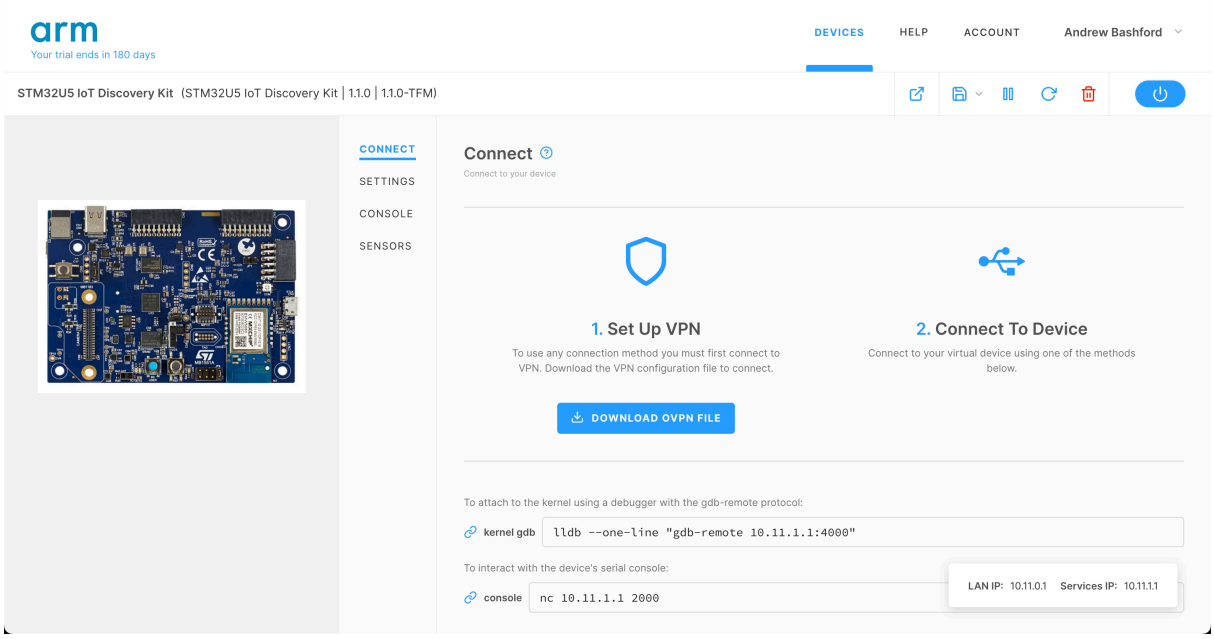
Figure 3-7: Creating device



9. Your STM32U5 IoT Discovery Kit is now ready to go!

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-8: Connect



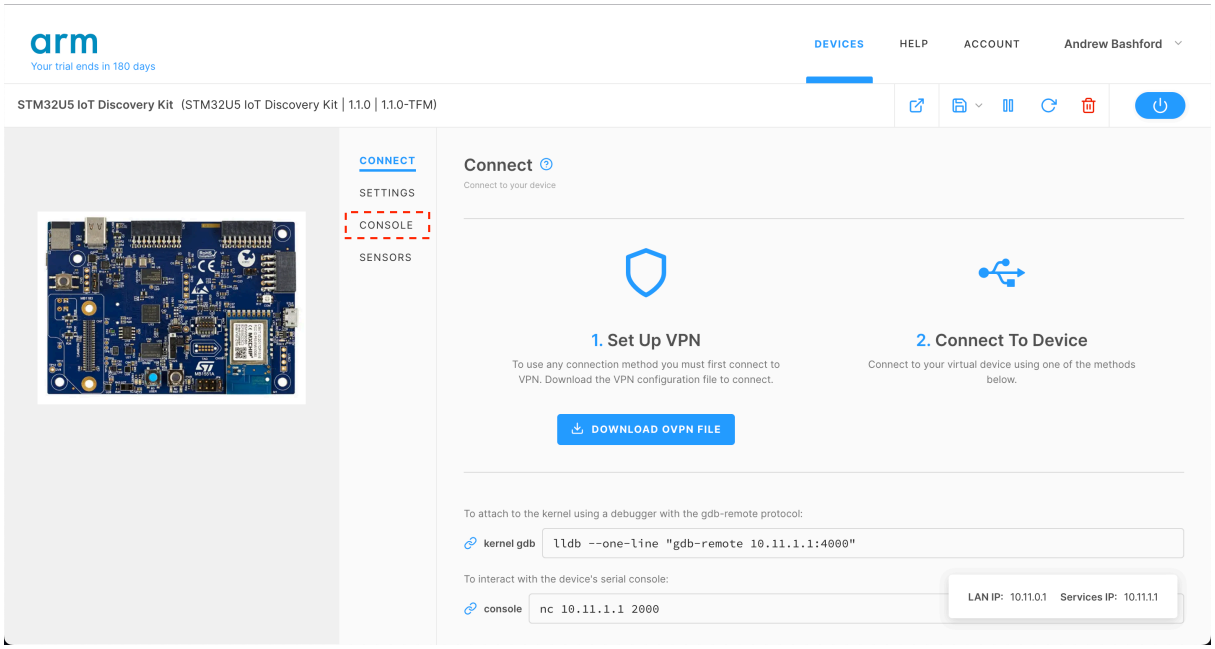
3.1.2 Running the example

To run the example:

1. Click on CONSOLE.

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-9: Click Console



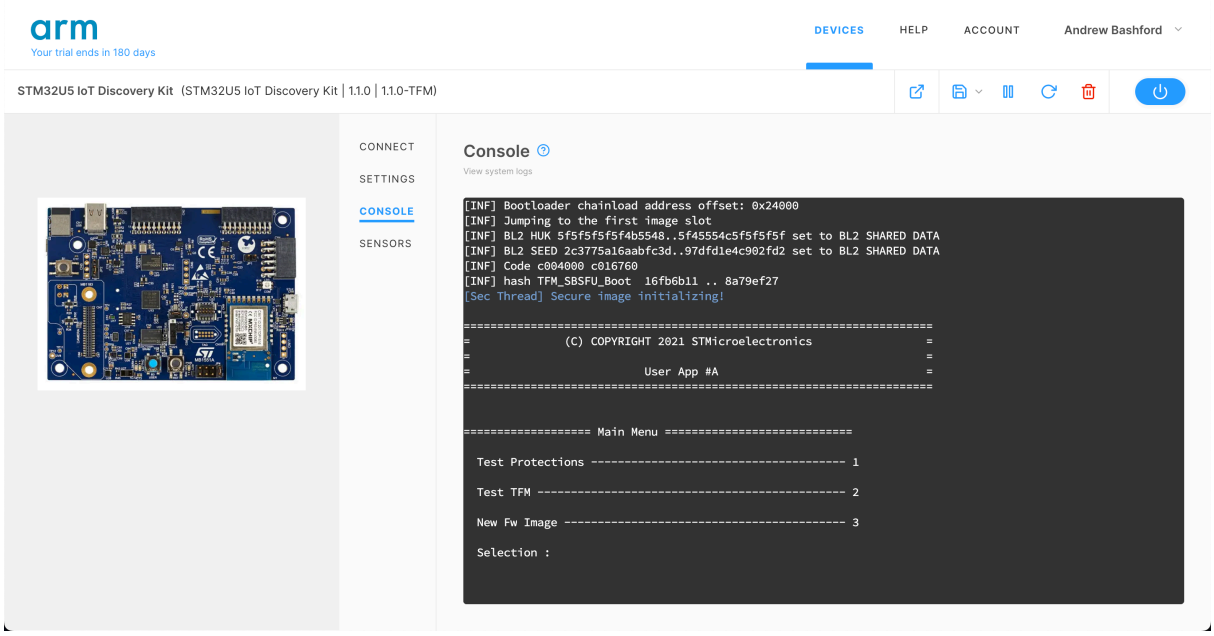
2. There is a console menu to run various tests using Trusted Firmware. For example type 2 (Test TFM) followed by return.

Note

You may see `Invalid Number !` also menus printed twice this is because the device is interpreting the return keypress as a number. Please ignore this.

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-10: Running tests on the console



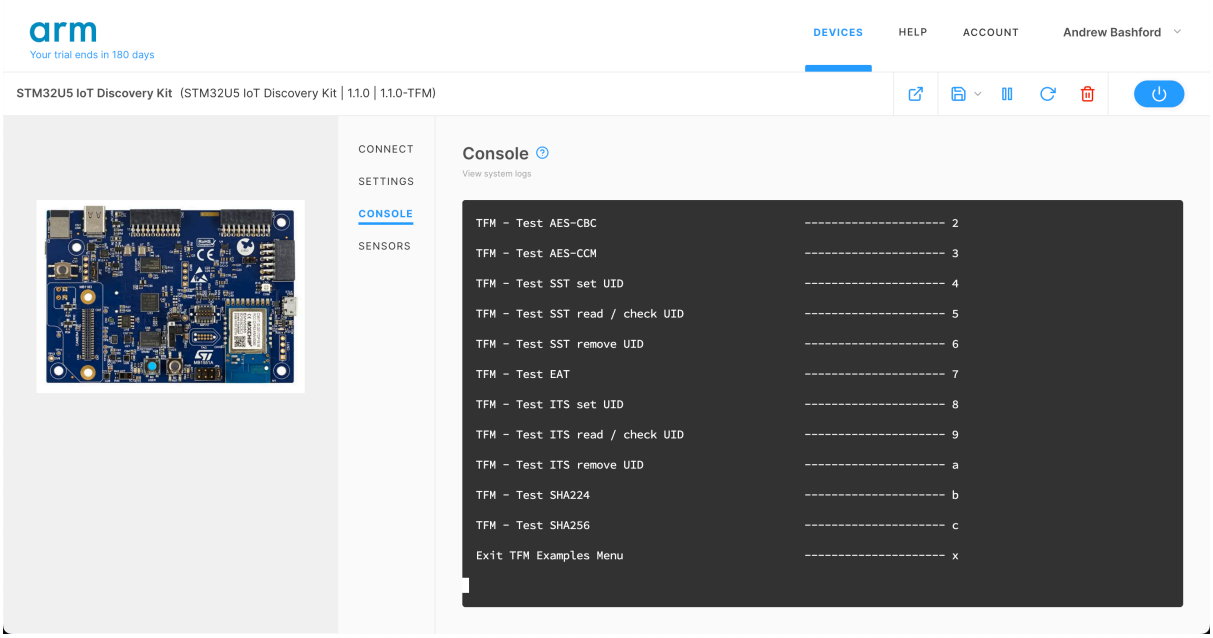
3. Then type `0` ("TFM - Test All") followed by `return`. to start tests.

Note

You may need to scroll up to see all of the menu.

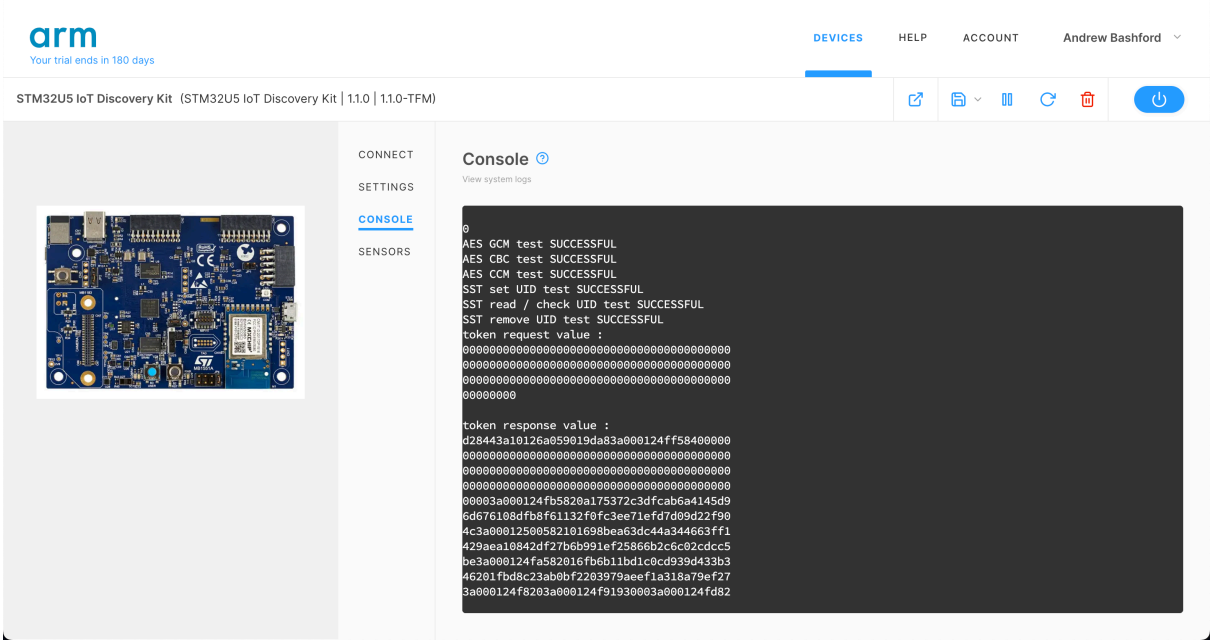
Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-11: TFM - Test All



4. You will be able to observe the result in the Console by scrolling up.

Figure 3-12: Test results



Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

3.1.3 Next steps

Other STM32U5 IoT Discovery Kit examples;

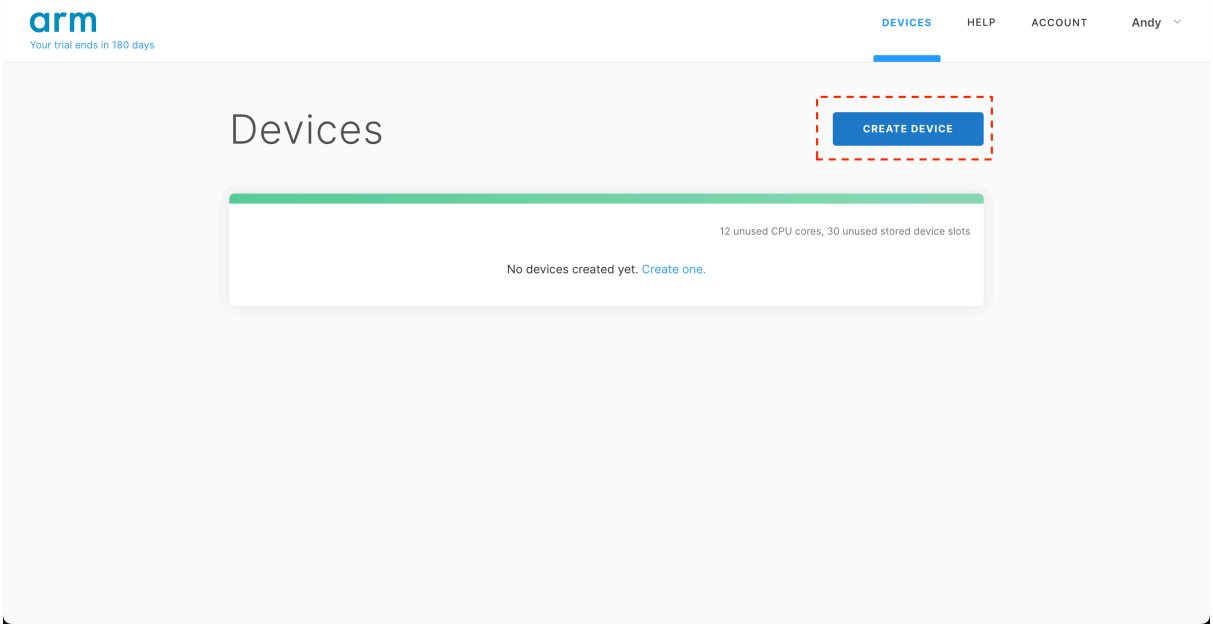
- STM32CubeU5 WiFiBasics example
- STM32CubeU5 IOT_HTTP_WebServer (This is the STM32U5 IoT Discovery Kit Quickstart guide)

3.2 STM32CubeU5 WiFiBasics example

This example will test basic WiFi networking functions:

1. Log in to AVH with your Arm user account. If you do not have or need to create an Arm account, register at https://developer.arm.com/.
2. When logged in you land on the AVH Devices page. Click CREATE DEVICE.

Figure 3-13: Create device



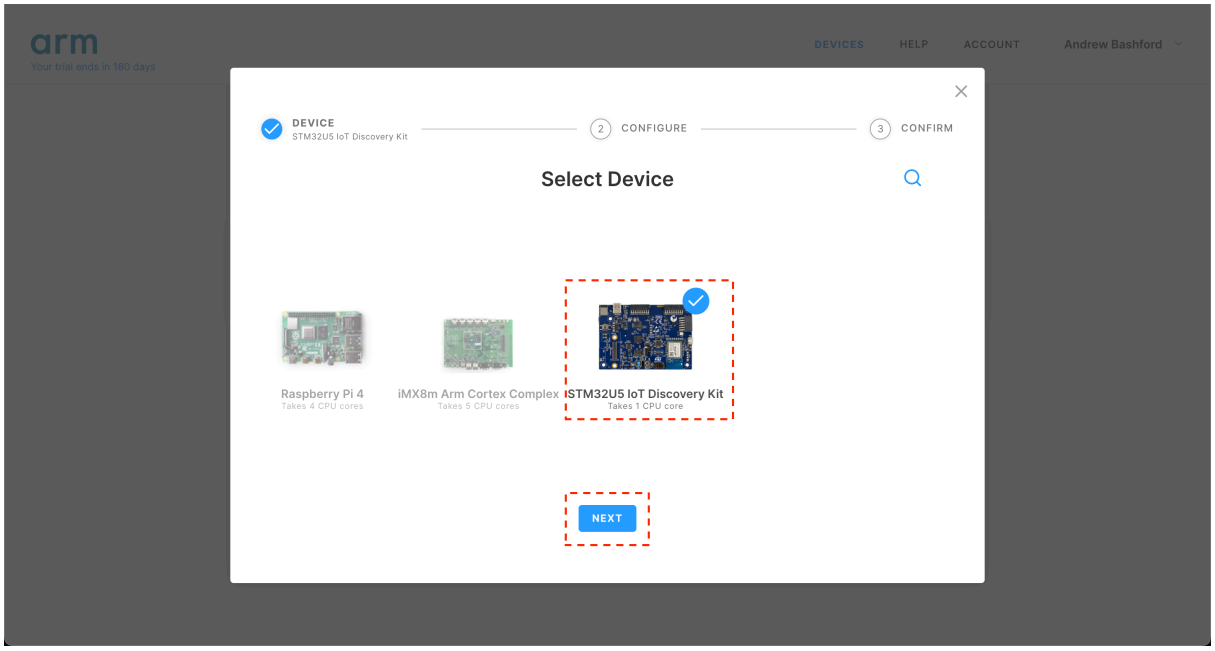
3. Click on the STM32U5 IoT Discovery Kit to select it. Then click NEXT.

Note

You require 1 free CPU core to build this device or it will be disabled.

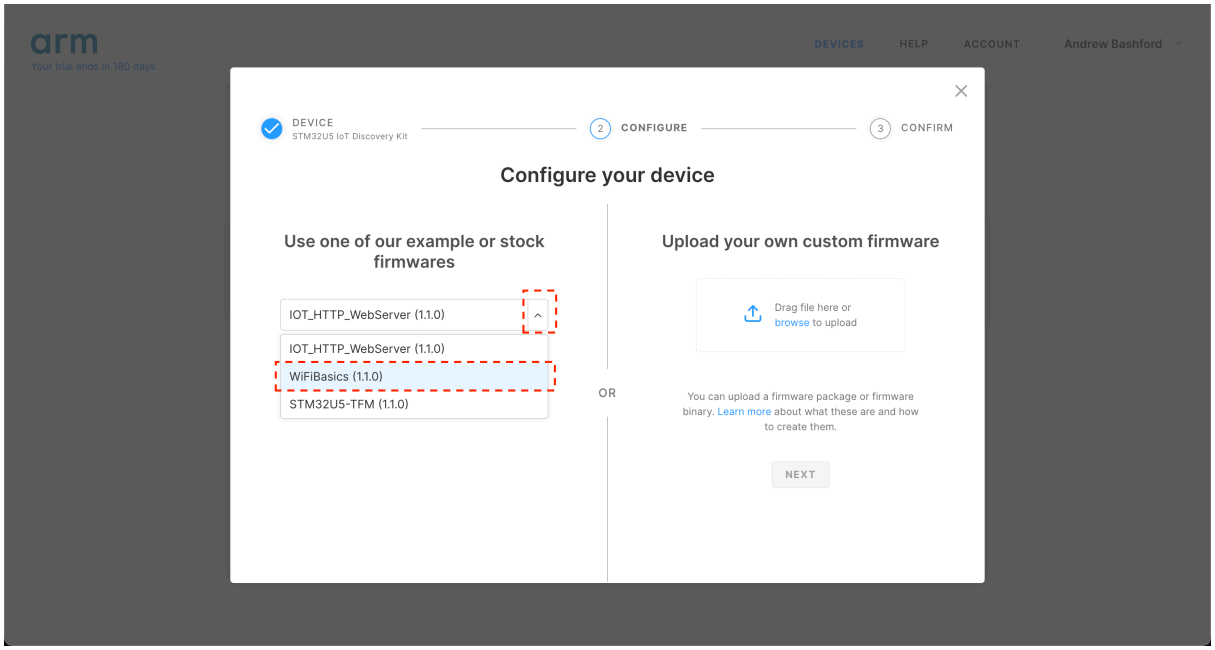
Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-14: Select device



4. In this next step, you will see a list of pre-configured examples/stock firmware or you can upload your own custom firmware. In this example, we will use the selection `WiFiBasics (1.1.0)` in the stock firmware section.

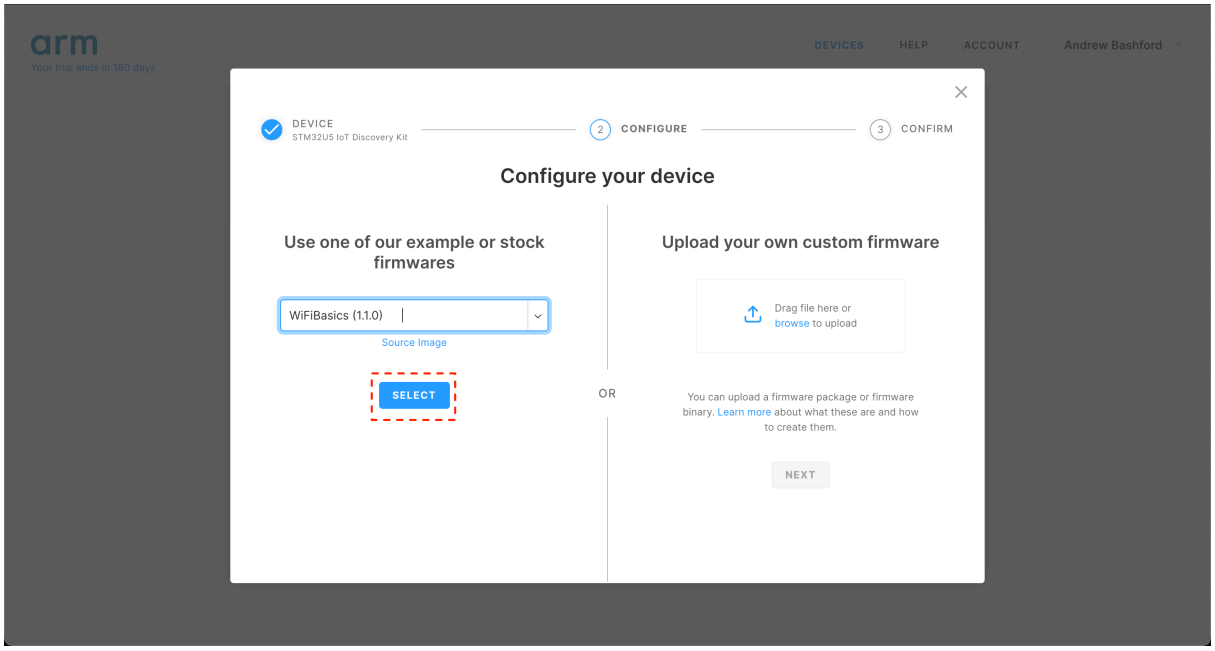
Figure 3-15: Configure your device



5. Click SELECT.

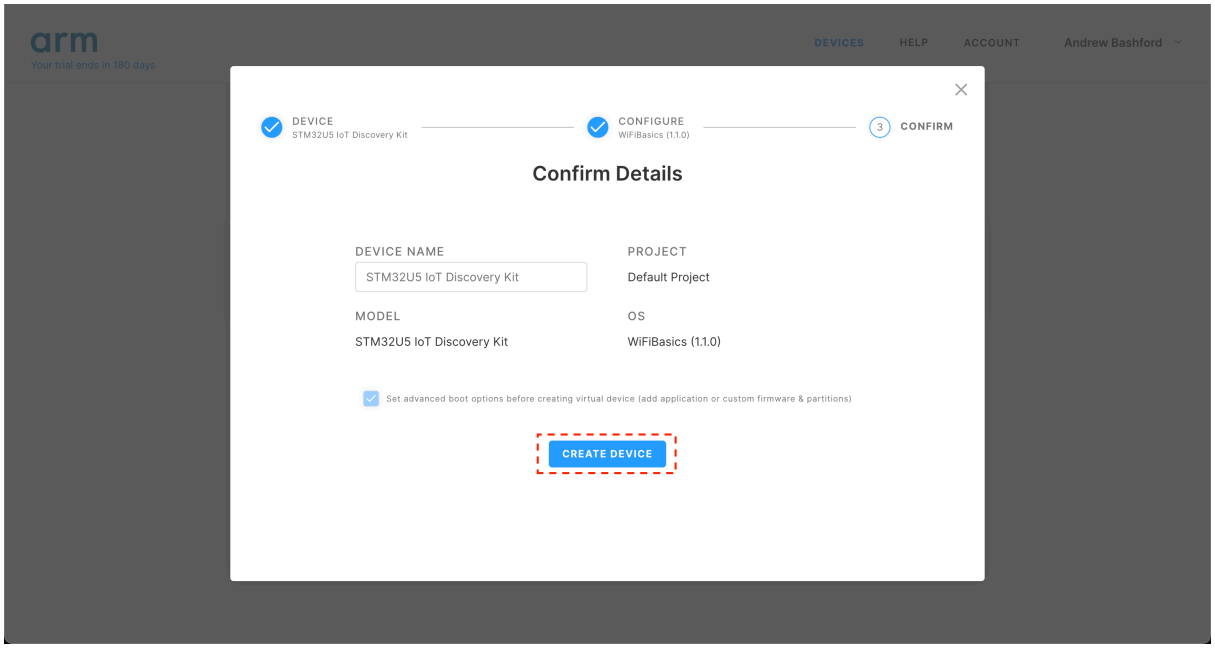
Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-16: Configure your device



6. In the confirmation screen, click CREATE DEVICE using the default name and the advance boot options will be checked by default.

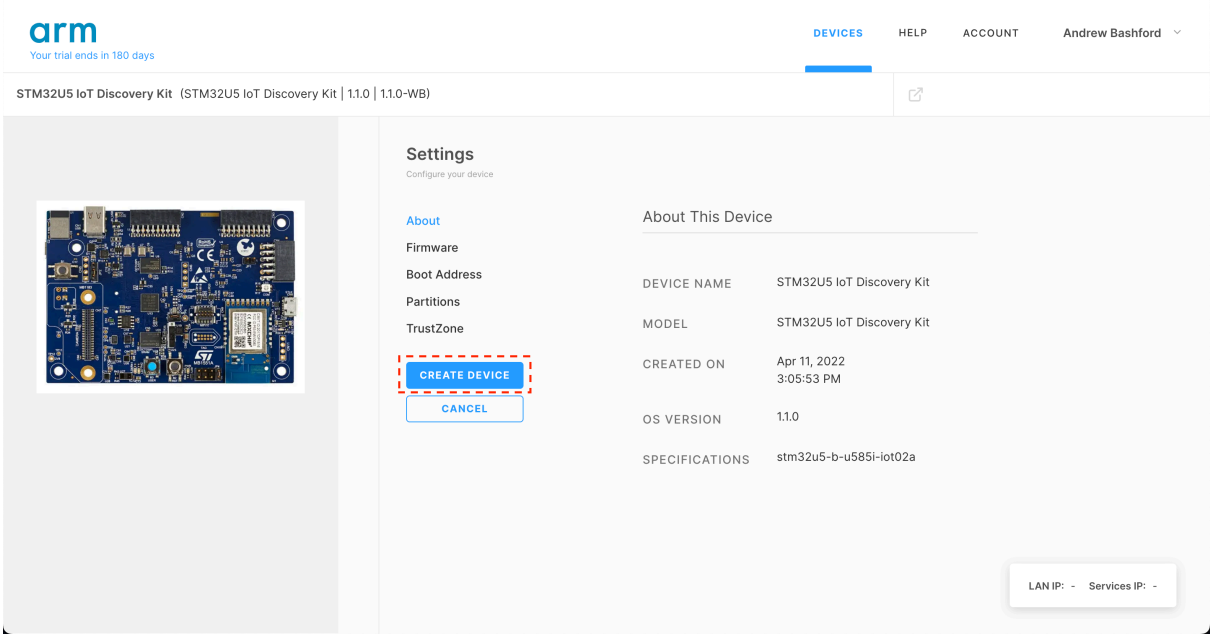
Figure 3-17: Confirm Details



7. In the advanced boot options, we will just use the default. Click CREATE DEVICE.

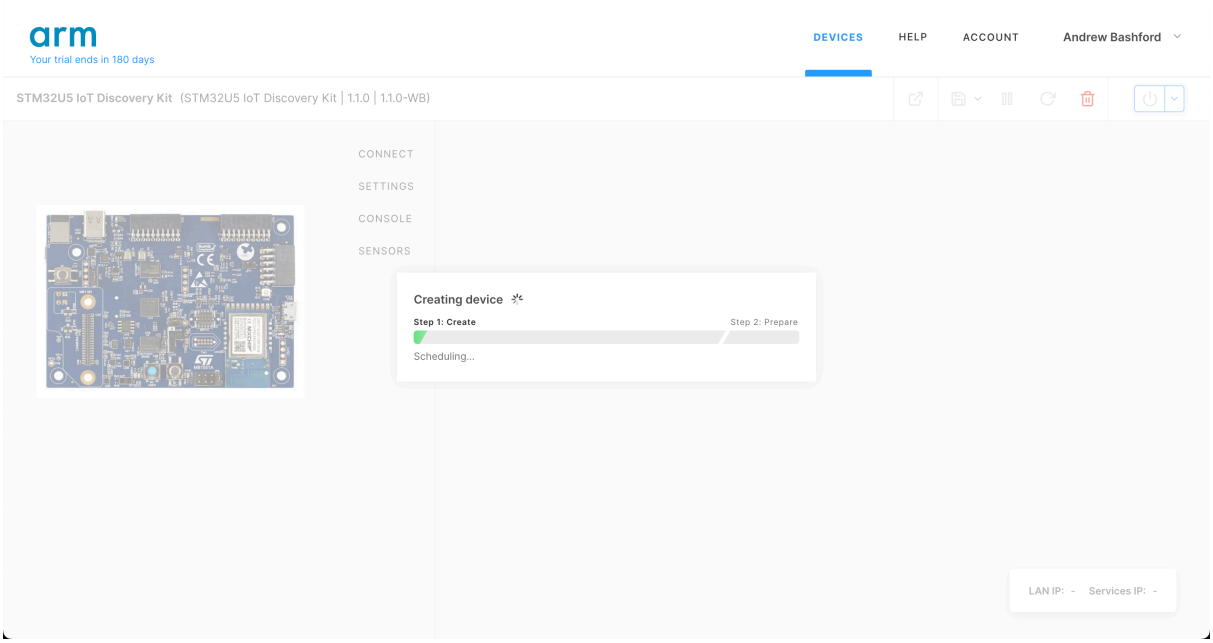
Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-18: Settings



8. The device will then be created. It takes a few moments…

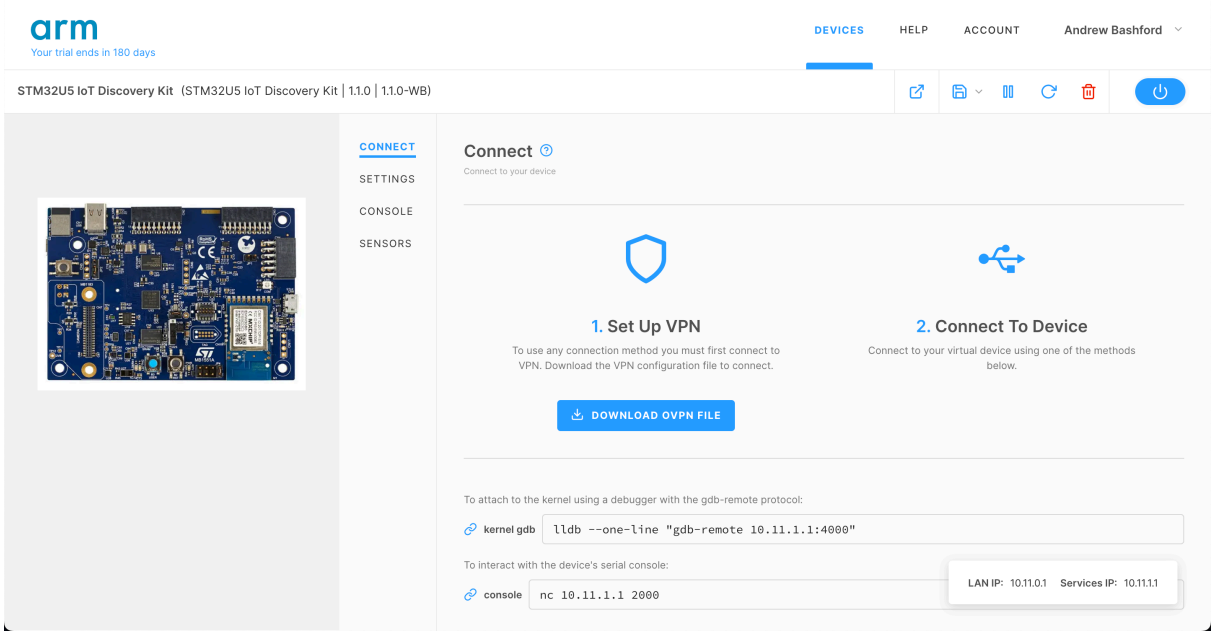
Figure 3-19: Creating device



9. Your STM32U5 IoT Discovery Kit is now ready to go!

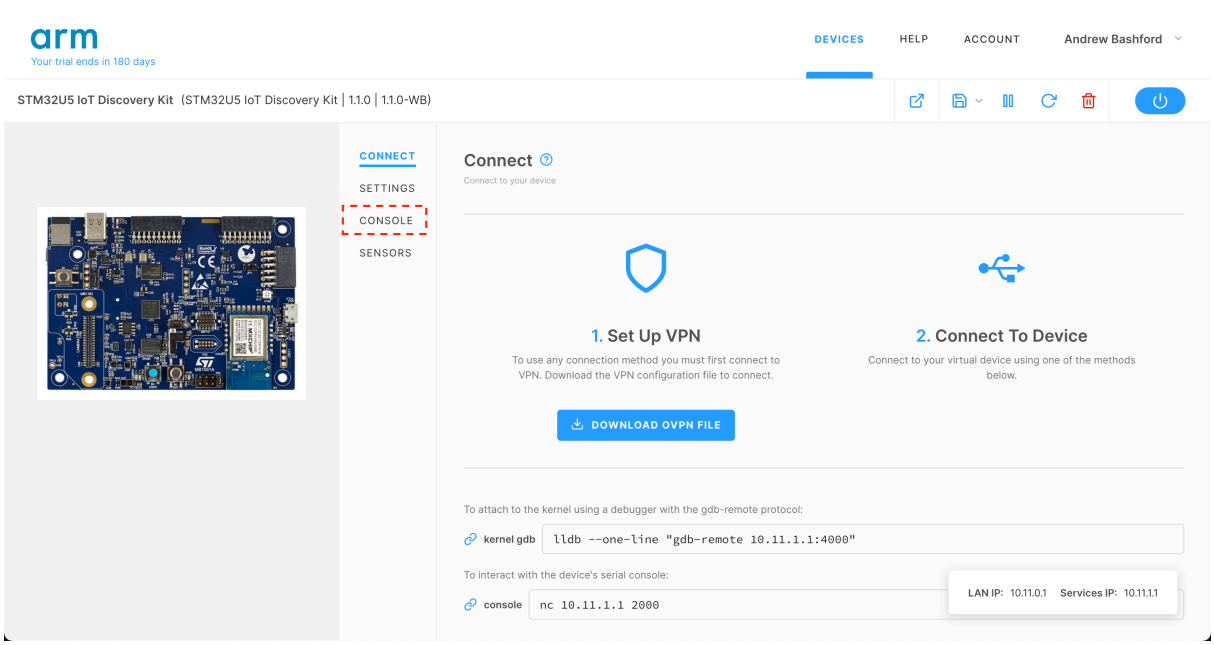
Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-20: Connect



10. Click CONSOLE to go to the Console.

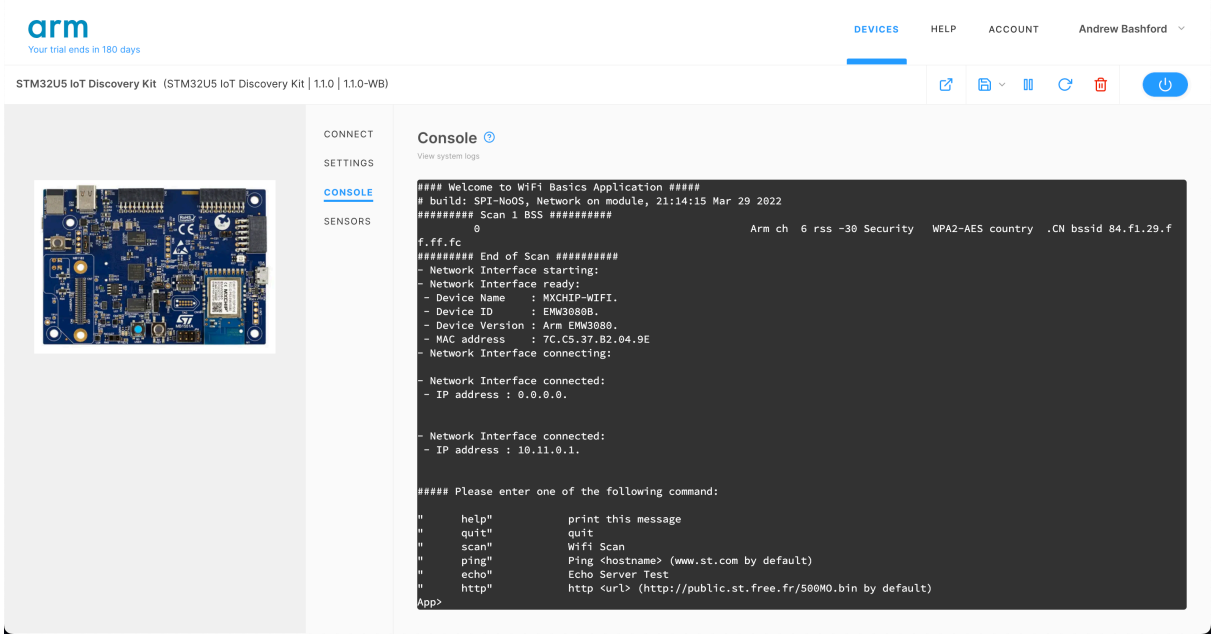
Figure 3-21: Click CONSOLE



11. Model scans for the available WiFi network and connects to the default one ("arm" in this case). There is a console menu to test various functions. For example, `http` command downloads binary from the ST website.

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 3-22: Console



Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

4. Raspberry Pi 4 Example

Raspberry Pi examples:

- Raspberry Pi vMMIO example

4.1 Raspberry Pi vMMIO example

This example will exercise vMMIO using the RaspberryPi model by demonstrating an external (over the network) MMIO handler and a local (RaspberryPi userspace) driver.

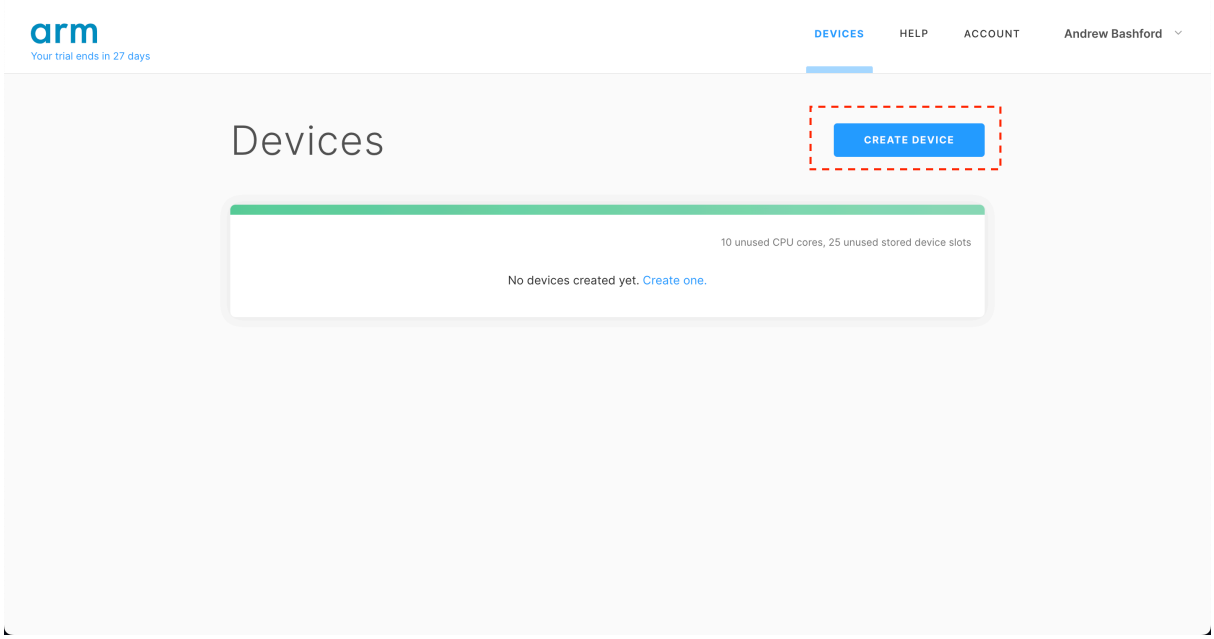
Read more in the AVH vMMIO Introduction article.

4.1.1 Creating the VM

To create the VM:

1. Create a Raspberry Pi 4 Virtual Machine. In DEVICES in the menu click on CREATE DEVICE.

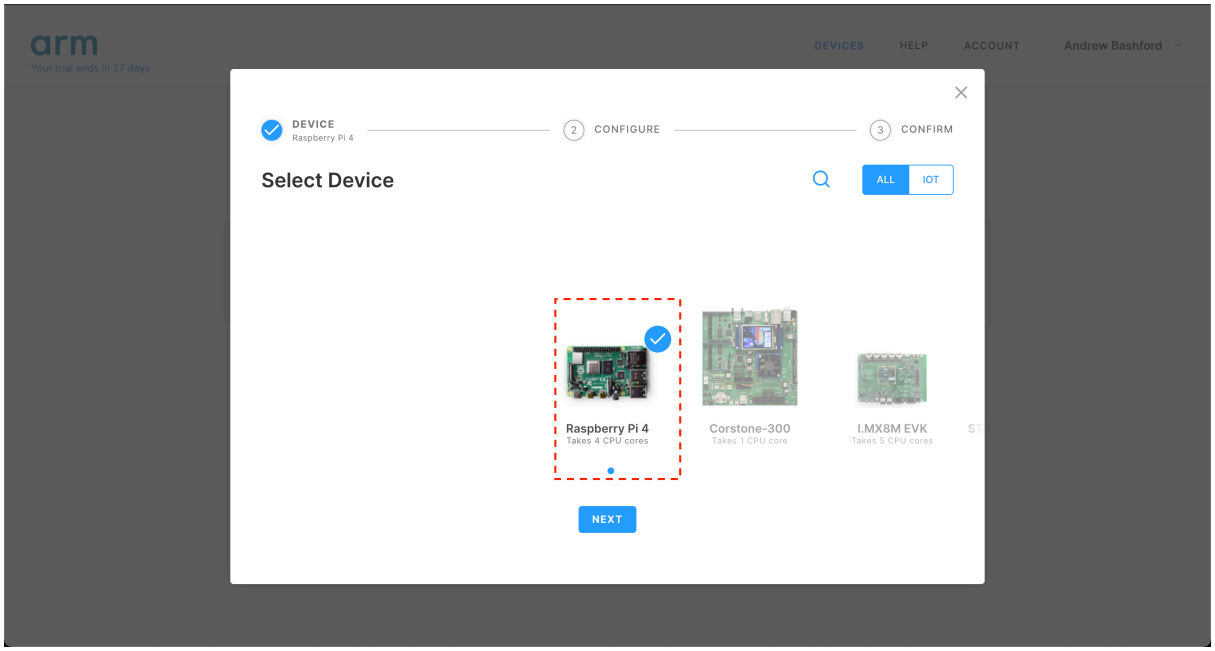
   Figure 4-1: Create device



2. Click on the RPi VM device then click NEXT.

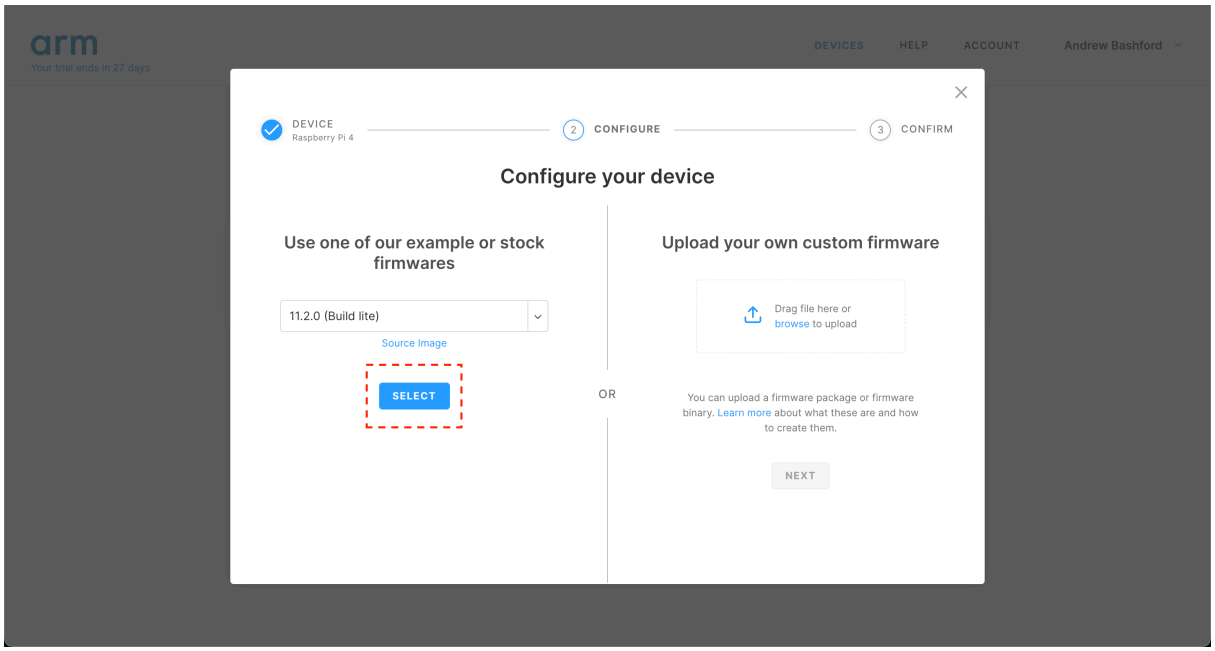
Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 4-2: Select device



3. Select the `lite` firmware option if not already selected. Then click SELECT.

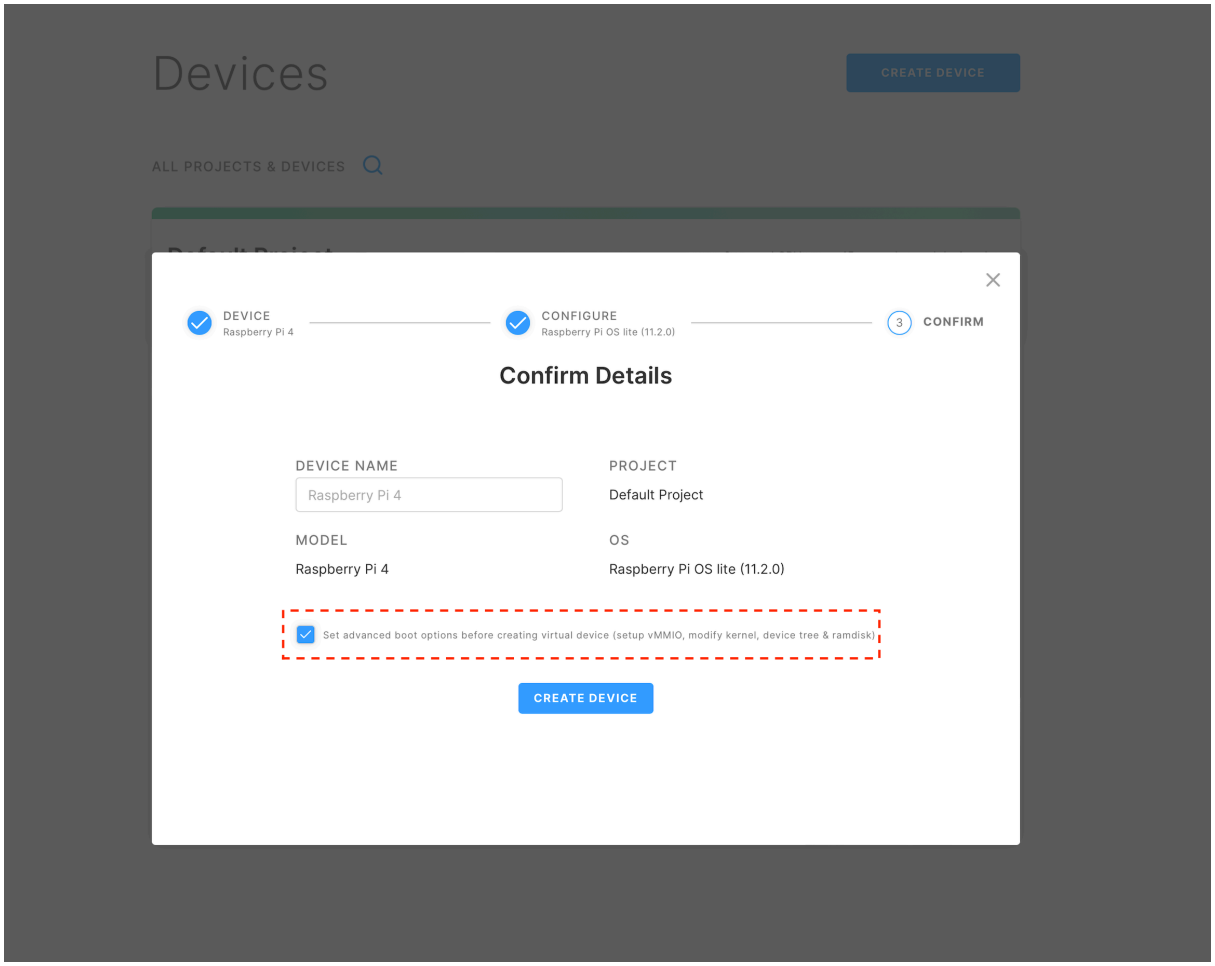
Figure 4-3: Configure your device



4. Select the `Set advanced boot options ...` checkbox and click CREATE DEVICE.

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

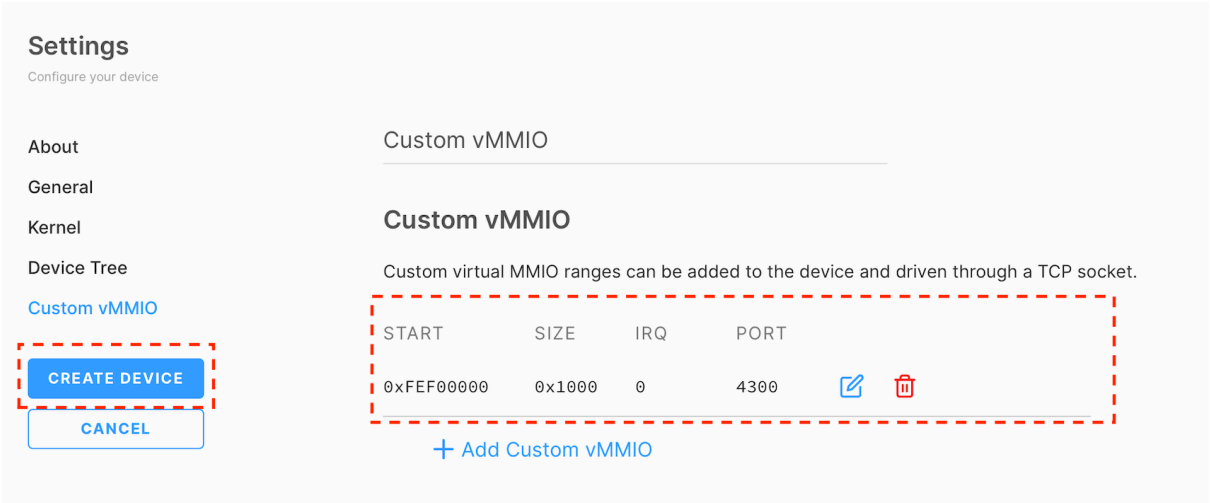
Figure 4-4: Confirm details



5. Select Custom vMMIO and enter `0xfef00000` for `start`, `0x1000` for `size`, 0 for `IRQ` (not used in this example), and 4300 for `port` (arbitrary). Then click CREATE DEVICE one final time.

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 4-5: Custom vMMIO

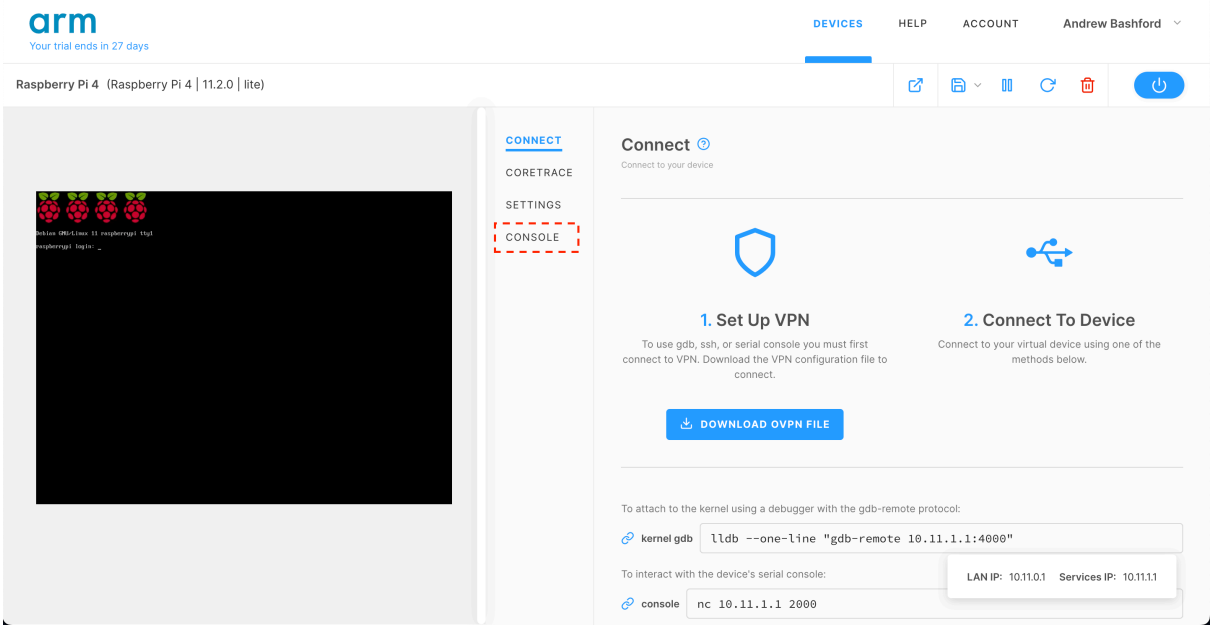


Note

`0xfef00000` was selected because the Raspberry Pi hardware does not have anything mapped at this address and as such it is safe to use for our purposes.

6. The device will take a few moments to build. When complete click on CONSOLE.

Figure 4-6: Connect



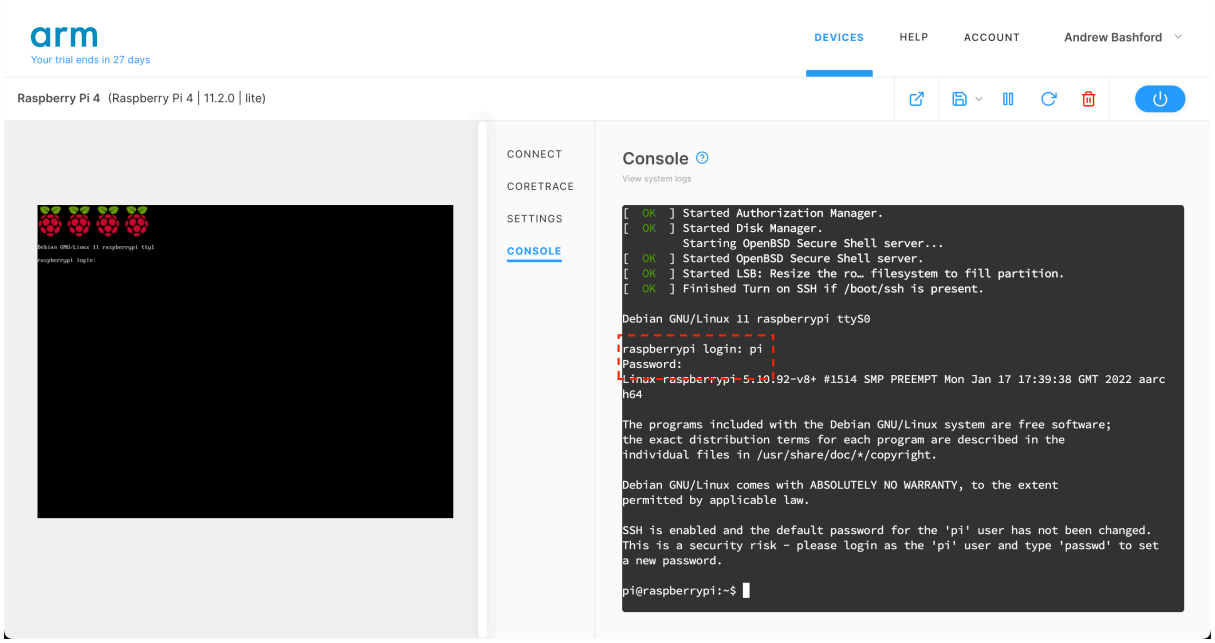
Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

4.1.2 Writing to MMIO Without a vMMIO Handler Registered

To write to MMIO without a vMMIO handler registered:

1. Log in using the default credentials of `pi/raspberry`.

   Figure 4-7: Log in



2. Download project sources to your Raspberry Pi, extract, and build it.

```
pi@raspberrypi:~ $ wget https://avh.arm.com/downloads/coreio.tar.gz
...

2022-04-12 14:28:55 (18.0 MB/s) - 'coreio.tar.gz' saved [23940/23940]

pi@raspberrypi:~ $ tar -xf coreio.tar.gz
pi@raspberrypi:~ $ cd coreio/examples/rpi-test/
pi@raspberrypi:~/coreio/examples/rpi-test $ make
gcc -g -Wall -I../../ -shared -fPIC -o libcoreio.so ../../corehdl-base.c ../../
coreio.c
gcc -g -Wall -I../../ -c -o corehdl-base.o ../../corehdl-base.c
gcc -g -Wall -I../../ -c -o coreio.o ../../coreio.c
ar cr libcoreio.a corehdl-base.o coreio.o
gcc -g -Wall -I../../ -ovmmio-test-inside vmmio-test-inside.c
gcc -g -Wall -I../../ -ovmmio-test-outside vmmio-test-outside.c libcoreio.a
```

3. Run the `vmmio-test-inside` build product

   This example source will open `/dev/mem` at offset `0xfef00000` (recall this is the address of the registered vMMIO) and attempt to read and write to it.

   However, without any registered vMMIO handlers running, AVH's default behavior, as mentioned earlier, is to return `0xaa` in the registers receiving the read contents.

```
$ sudo ./vmmio-test-inside
```

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
0: 0xaa
1: 0xaa
2: 0xaa
3: 0xaa
...
0: 0xaaaa
1: 0xaaaa
2: 0xaaaa
3: 0xaaaa
...
0: 0xaaaaaaaa
1: 0xaaaaaaaa
2: 0xaaaaaaaa
3: 0xaaaaaaaa
...
0: 0xaaaaaaaaaaaaaaaa
1: 0xaaaaaaaaaaaaaaaa
2: 0xaaaaaaaaaaaaaaaa
3: 0xaaaaaaaaaaaaaaaa
```

In order to implement something more meaningful we need to handle those read and write requests by registering a handler

4.1.3 Writing to MMIO With a vMMIO Handler Registered

To write to MMIO with a vMMIO handler registered:

1. Download project sources to your local machine, extract, and build it.

```
$ wget https://avh.arm.com/downloads/coreio.tar.gz
...

2022-04-12 14:28:55 (18.0 MB/s) - 'coreio.tar.gz' saved [23940/23940]

$ tar -xf coreio.tar.gz
$ cd coreio/examples/rpi-test/
$ make
gcc -g -Wall -I../../ -shared -fPIC -o libcoreio.so ../../corehdl-base.c ../../
coreio.c
gcc -g -Wall -I../../ -c -o corehdl-base.o ../../corehdl-base.c
gcc -g -Wall -I../../ -c -o coreio.o ../../coreio.c
ar cr libcoreio.a corehdl-base.o coreio.o
gcc -g -Wall -I../../ -ovmmio-test-inside vmmio-test-inside.c
gcc -g -Wall -I../../ -ovmmio-test-outside vmmio-test-outside.c libcoreio.a
```

2. Run the `vmmio-test-outside` build product, targeting the Services IP of the VM and vMMIO port (4300) as arguments

---

Note This step registers the vMMIO handler, overriding the default behavior just observed.

```
$ ./vmmio-test-outside 10.11.1.1:4300
tick
tick
```

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

```
tick
...
```

3. Re-run `vmmio-test-inside` within the RaspberryPi VM. Observe that the `vmmio-test-outside` binary will now emit state information for each read and write event happening inside the VM by `vmmio-test-inside`.

```
$ ./vmmio-test-outside 10.11.1.1:4300
tick
tick
tick
write fef00000 1 0000000000000000 0
write fef00001 1 0000000000000001 0
write fef00002 1 0000000000000002 0
write fef00003 1 0000000000000003 0
write fef00004 1 0000000000000004 0
write fef00005 1 0000000000000005 0
write fef00006 1 0000000000000006 0
tick
write fef00007 1 0000000000000007 0
write fef00008 1 0000000000000008 0
...
```

Meanwhile `vmmio-test-inside` will emit the contents it reads from the MMIO address after attempting to modify its contents with a write.

```
$ sudo ./vmmio-test-inside
0: 0x00
1: 0x01
2: 0x02
3: 0x03
...
0: 0x0100
1: 0x0201
2: 0x0302
3: 0x0403
...
0: 0x03020100
1: 0x04030201
2: 0x05040302
3: 0x06050403
...
0: 0x0706050403020100
1: 0x0807060504030201
2: 0x0908070605040302
3: 0x0a09080706050403
```

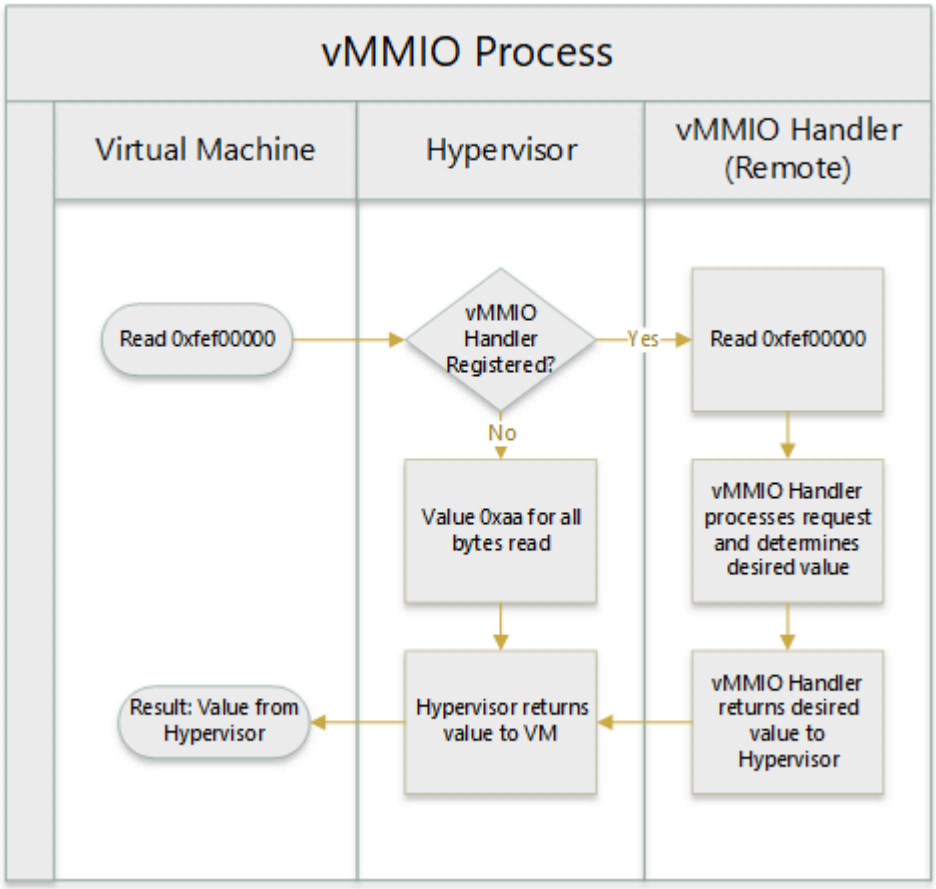
4.1.4 Flow Diagram

The flow followed in this example is illustrated below. The Writing to MMIO With a vMMIO Handler Registered example follows the “vMMIO Handler Registered” No path while the Writing to

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

MMIO With a vMMIO Handler Registered example has a remote handler registered and follows the Yes path.

Figure 4-8: Flow diagram



4.1.5 Summary

In summary, `vmmio-test-outside` is registering as an vMMIO read and write handler for a specified address range within the VM. When the physical address range specified is read from or written to, this handler receives a callback allowing it to handle the request. In this example, the `vmmio-test-inside` binary opened `/dev/mem` and simulates the OS or some other entity reading or writing to physical memory in order to exercise the callbacks in the registered vMMIO handler.

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

5. API Examples

API examples:

- Python and Javascript API Examples
- GitHub Actions Example - Building Firmware, Uploading, and Testing

5.1 Python and Javascript API Examples

Visiting the AVH GitHub where you will find a repo with examples for both Python and Javascript.

5.1.1 Python Examples

The Python examples are as follows:

- `instance_create_snapshot_delete.py`
  - This example demonstrates creating a STM32U5 device, taking a snapshot and then deleting it.
- `stm32_bsp_test.py`
  - This example demonstrates using GPIO input/output of a virtual machine
- `create_github_runner.py`
  - This example demonstrates using GPIO input/output of a virtual machine

5.1.2 Javascript Examples

The Javascript examples are as follows:

- `arm_api`
  - ArmApi - JavaScript client for arm_api REST API to manage your virtual devices.
- `sensordemo-docker`
  - This example will execute a build on your runner and create a new device on AVH to test the output of the build on.
- `sensordemo-generate-api`

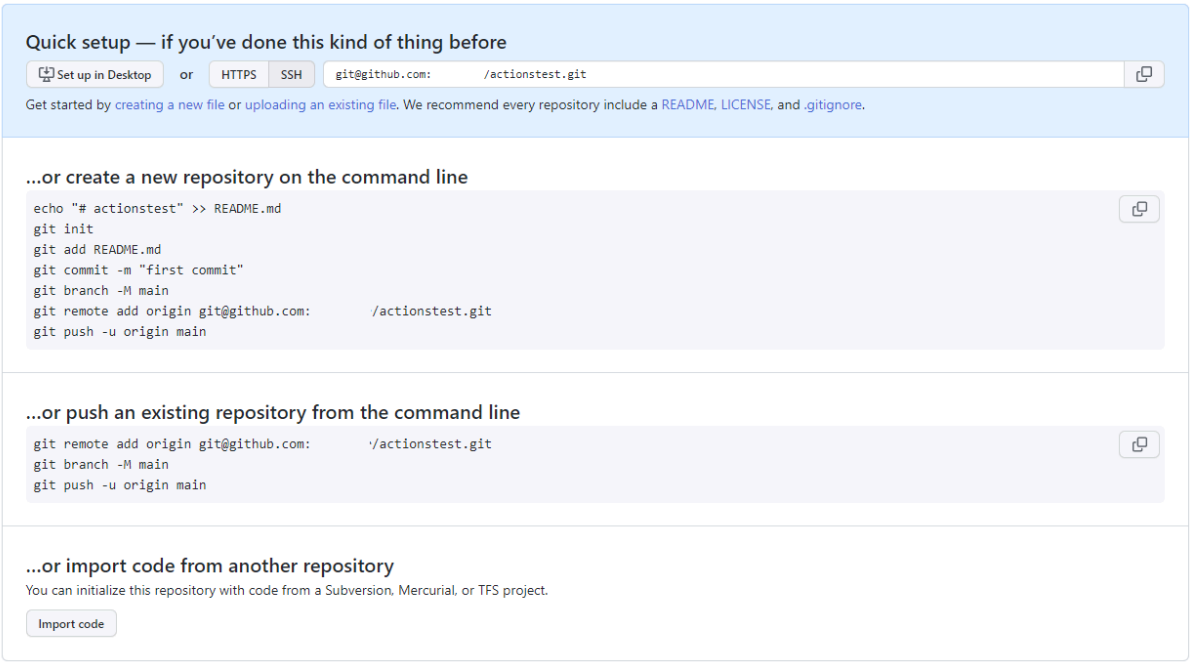
Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

5.2 GitHub Actions Example - Building Firmware, Uploading, and Testing

The following steps help you to set up GitHub Actions.

1. To get a VM ready for the GitHub runner, follow the steps in the Running Docker on AVH Pi article.
2. Ensure you have an AVH API Token by following the steps in the Obtaining AVH API Token article.
3. Create an empty GitHub repository for your runner to accept jobs from by following the steps in this GitHub Article.

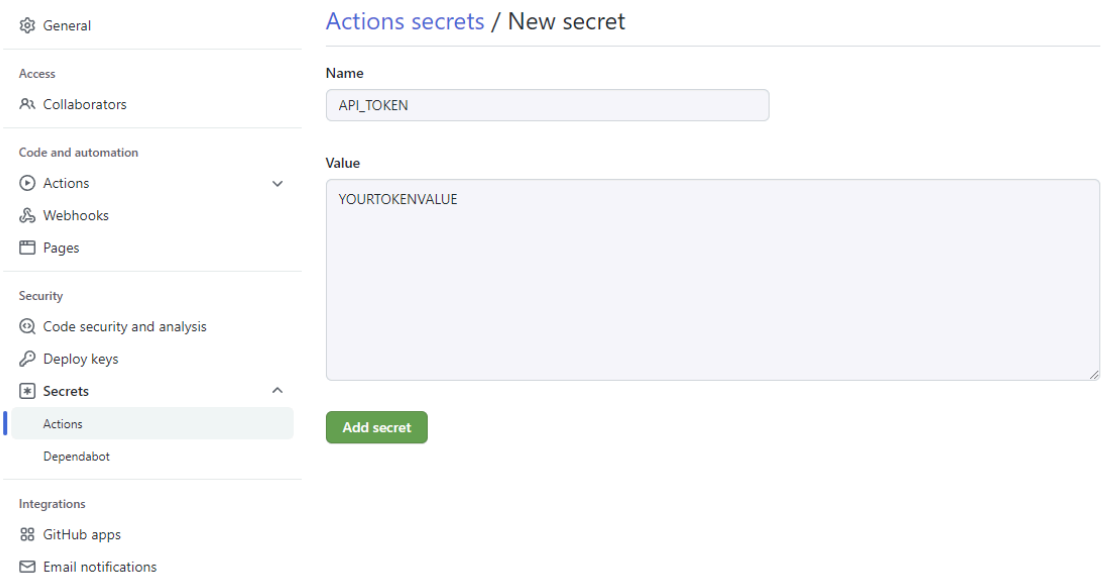
Figure 5-1: Create an empty GitHub repository



4. Add a "API_TOKEN" secret to your repository with a value of your API Token by following the steps in this GitHub Article.

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

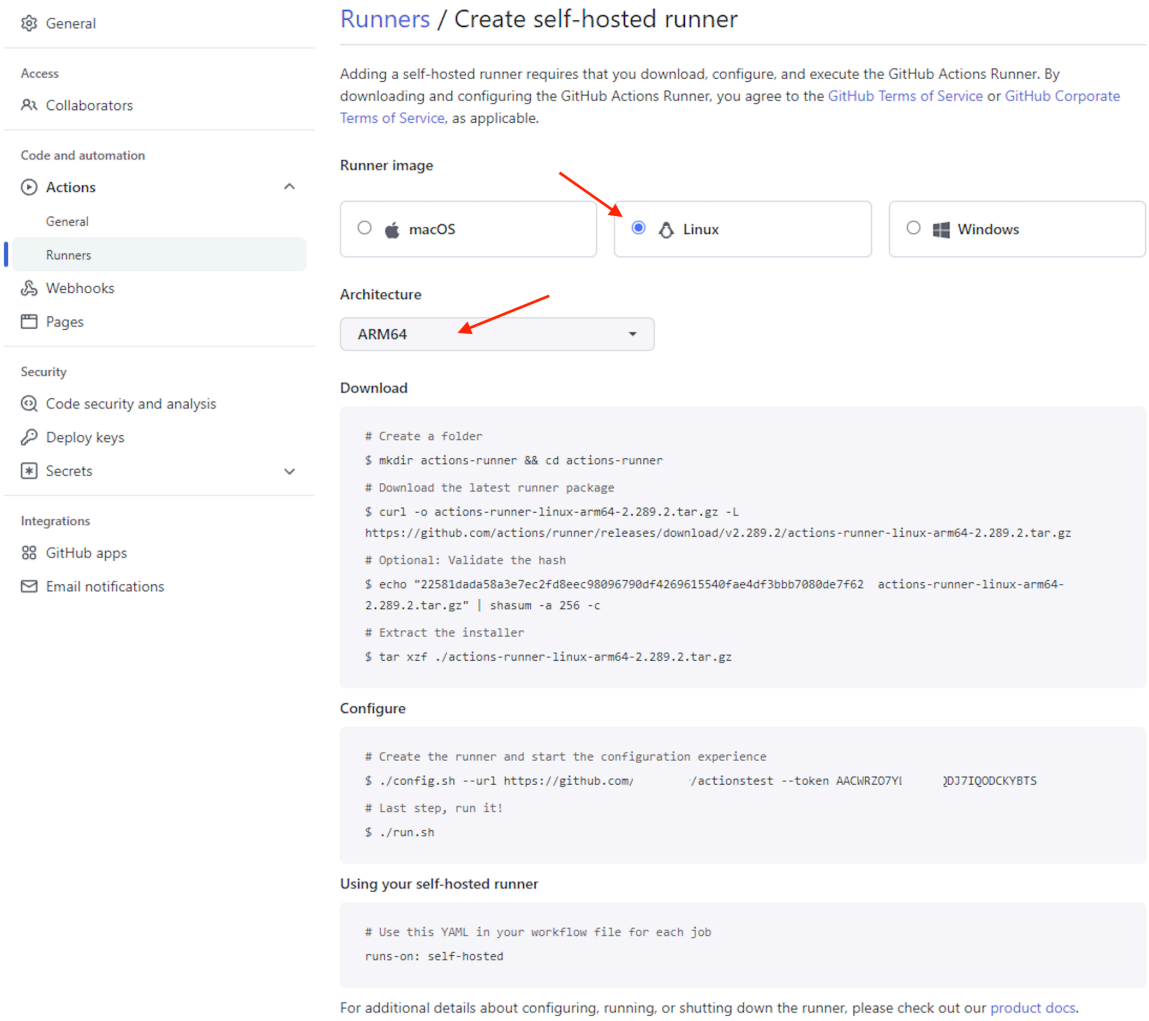
Figure 5-2: API_TOKEN



5. To configure and start the runner, follow the steps in this Adding Self Hosted Runners article. Select “Linux” as the OS and “ARM64” as the architecture.

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

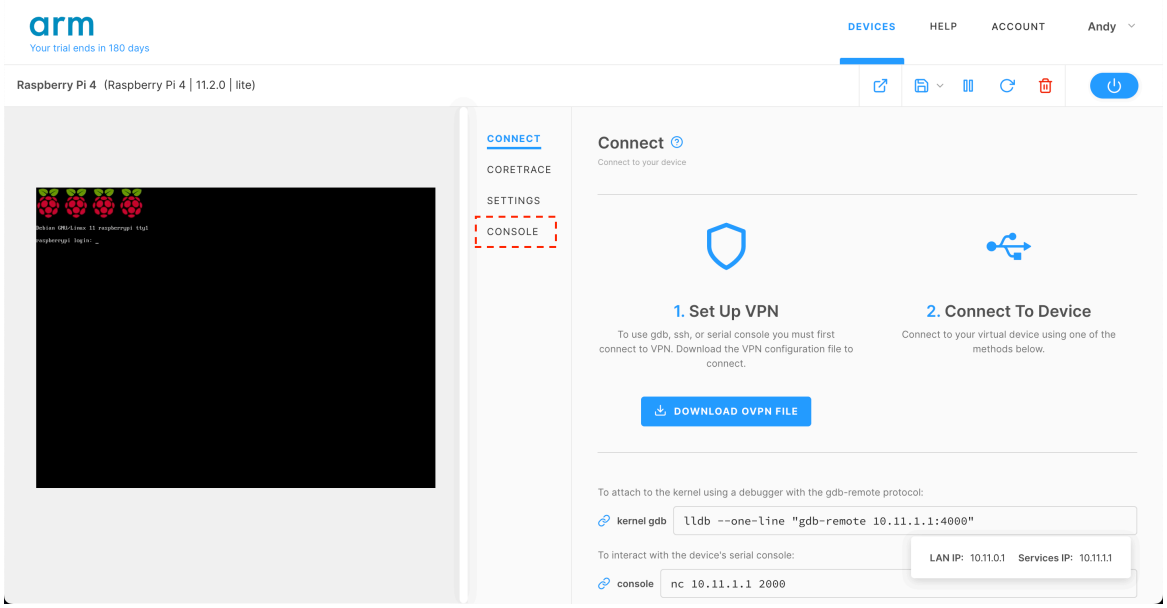
Figure 5-3: Adding self hosted runners



a. In AVH, select the Raspberry Pi VM created previously. Then select “CONSOLE” to access a shell to run the commands referenced above.

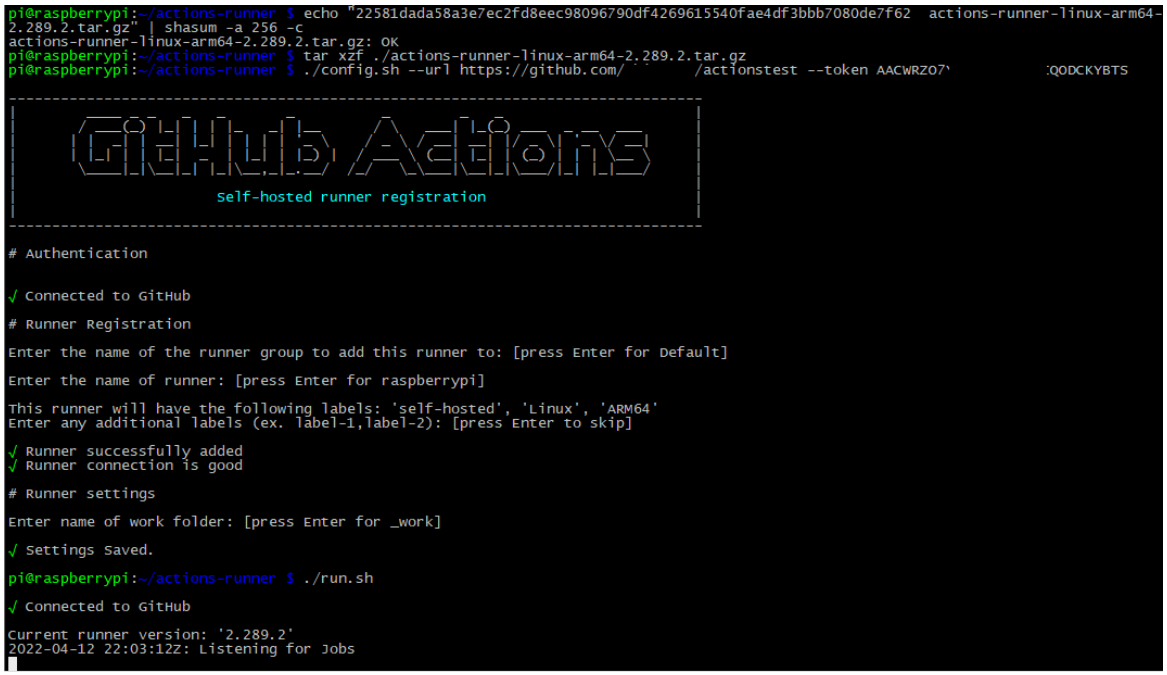
Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 5-4: Select Console



b. Upon completion of runner setup you should see something similar to:

Figure 5-5: Console output



c. Do not cancel the running process or your runner will be unavailable until it is run again. The runner can also be installed as a service, but that is not covered in this article.

6. Clone the avh-api repository on your usual local development system from GitHub with: `git clone https://github.com/ARM-software/avh-api.git`

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

Figure 5-6: Clone the avh-api repository

```
pi@raspberrypi:~$ git clone https://github.com/ARM-software/avh-api.git
Cloning into 'avh-api'...
remote: Enumerating objects: 1256, done.
remote: Counting objects: 100% (1256/1256), done.
remote: Compressing objects: 100% (638/638), done.
remote: Total 1256 (delta 600), reused 1227 (delta 573), pack-reused 0
Receiving objects: 100% (1256/1256), 25.91 MiB | 31.97 MiB/s, done.
Resolving deltas: 100% (600/600), done.
pi@raspberrypi:~$
```

Note

We used a raspberry pi for this in this example, this will likely be your laptop or desktop system.

7. Enter the `javascript/sensordemo-docker` directory

Figure 5-7: Enter the directory

```
pi@raspberrypi:~$ cd avh-api/javascript/sensordemo-docker/
pi@raspberrypi:~/avh-api/javascript/sensordemo-docker$
```

8. Run `make clean gitrepo GITREMOTE='<url to Github repo you created>'` in the console.

Note

This will overwrite any contents of the repo passed as GITREMOTE=.

Figure 5-8: Run GITREMOTE=

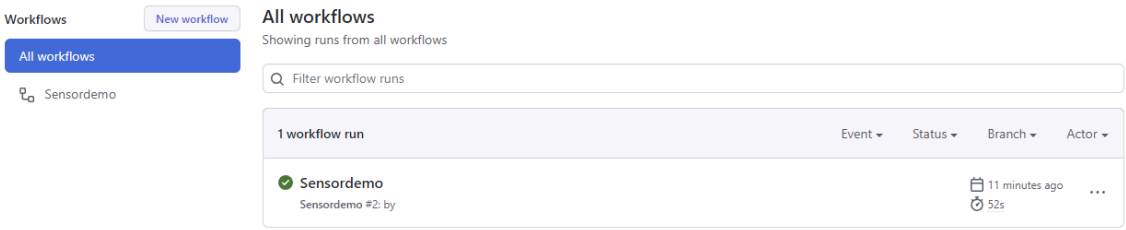
```
pi@raspberrypi:~/avh-api/javascript/sensordemo-docker $ make clean gitrepo GITREMOTE='https://github.com/        /actionstest.git'
rm -rf gitrepo
git archive HEAD --prefix="gitrepo/" | tar -xf -
cp -a ../api gitrepo/api
cd gitrepo; \
  git init; \
  mkdir -p .github/workflows; \
  mv github_workflow.yml .github/workflows/avh_build_and_test.yml; \
```

```
 create mode 100755 run_docker.sh
Example repository created in /home/pi/avh-api/javascript/sensordemo-docker/gitrepo
cd gitrepo; \
  git remote add origin https://github.com/        /actionstest.git; \
  git push -fu origin $(git rev-parse --abbrev-ref HEAD)
Username for 'https://github.com':
Password for 'https://        @github.com':
Enumerating objects: 750, done.
Counting objects: 100% (750/750), done.
Delta compression using up to 4 threads
Compressing objects: 100% (717/717), done.
Writing objects: 100% (750/750), 24.39 MiB | 8.82 MiB/s, done.
Total 750 (delta 215), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (215/215), done.
To https://github.com/        /actionstest.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
pi@raspberrypi:~/avh-api/javascript/sensordemo-docker $
```

9. You should see an action run on your repository executed from your AVH VM.

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential

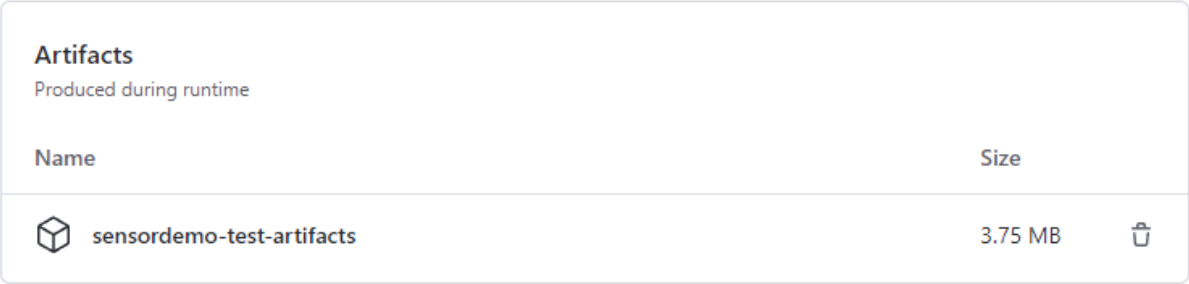
Figure 5-9: Executed action



This will compile a new firmware for STM32u5 on the pi VM, then upload it to a new STM32 VM in Arm Virtual Hardware, and run a test against it to ensure it behaved properly.

10. You will get a built firmware and the test results as the artifacts downloadable from GitHub when the action completes.

Figure 5-10: Artifacts



11. Any changes made to the repository, such as changing how the firmware behaves and pushes to GitHub, will trigger a new run.

Copyright © 2022 Arm Limited (or its affiliates). All rights reserved.
Non-Confidential