eIQ: Transfer Learning Using IMX RT1060 / IMX RT1050 EVK – Without Camera

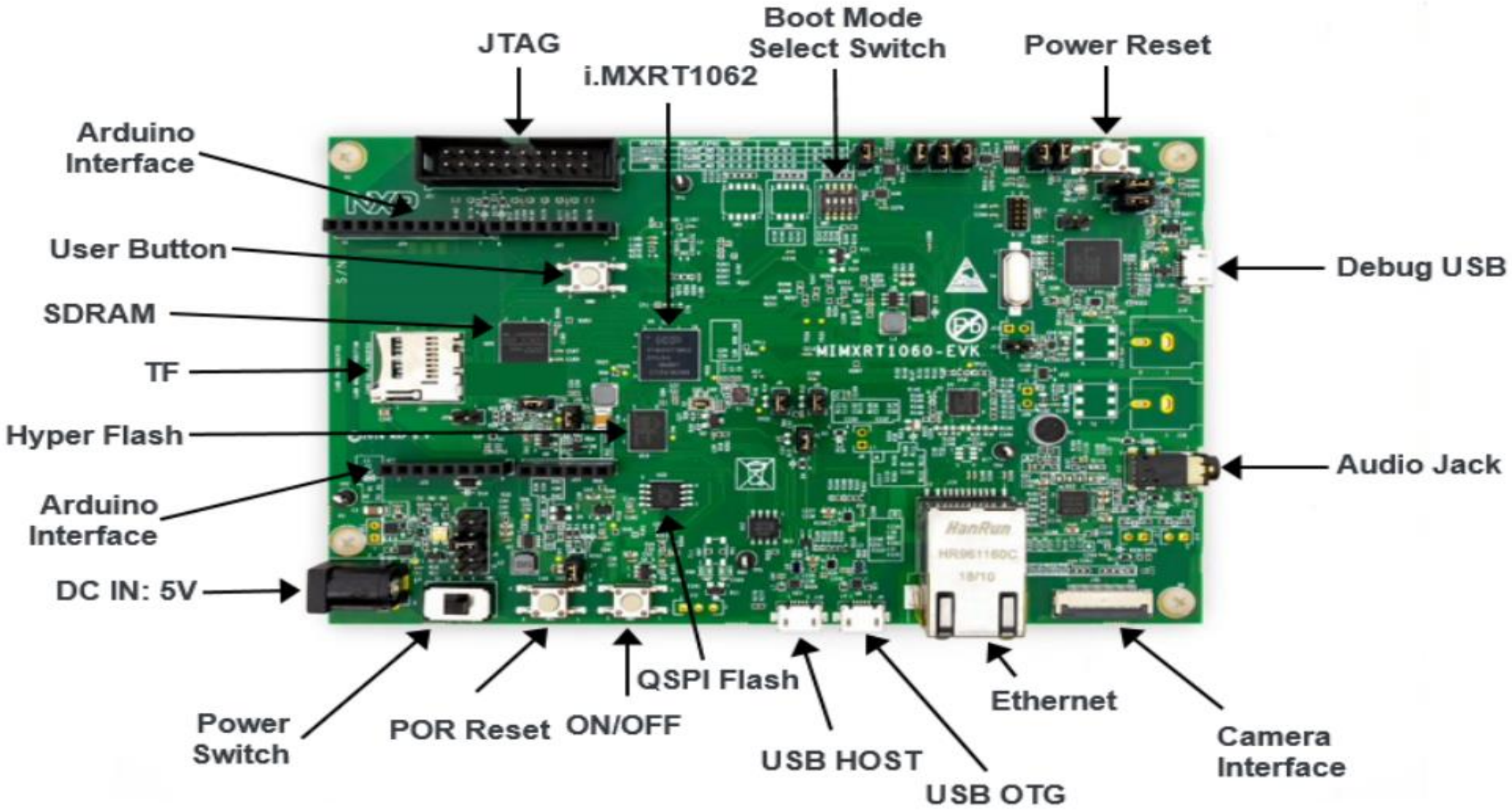
Contents:

Lab Overview:

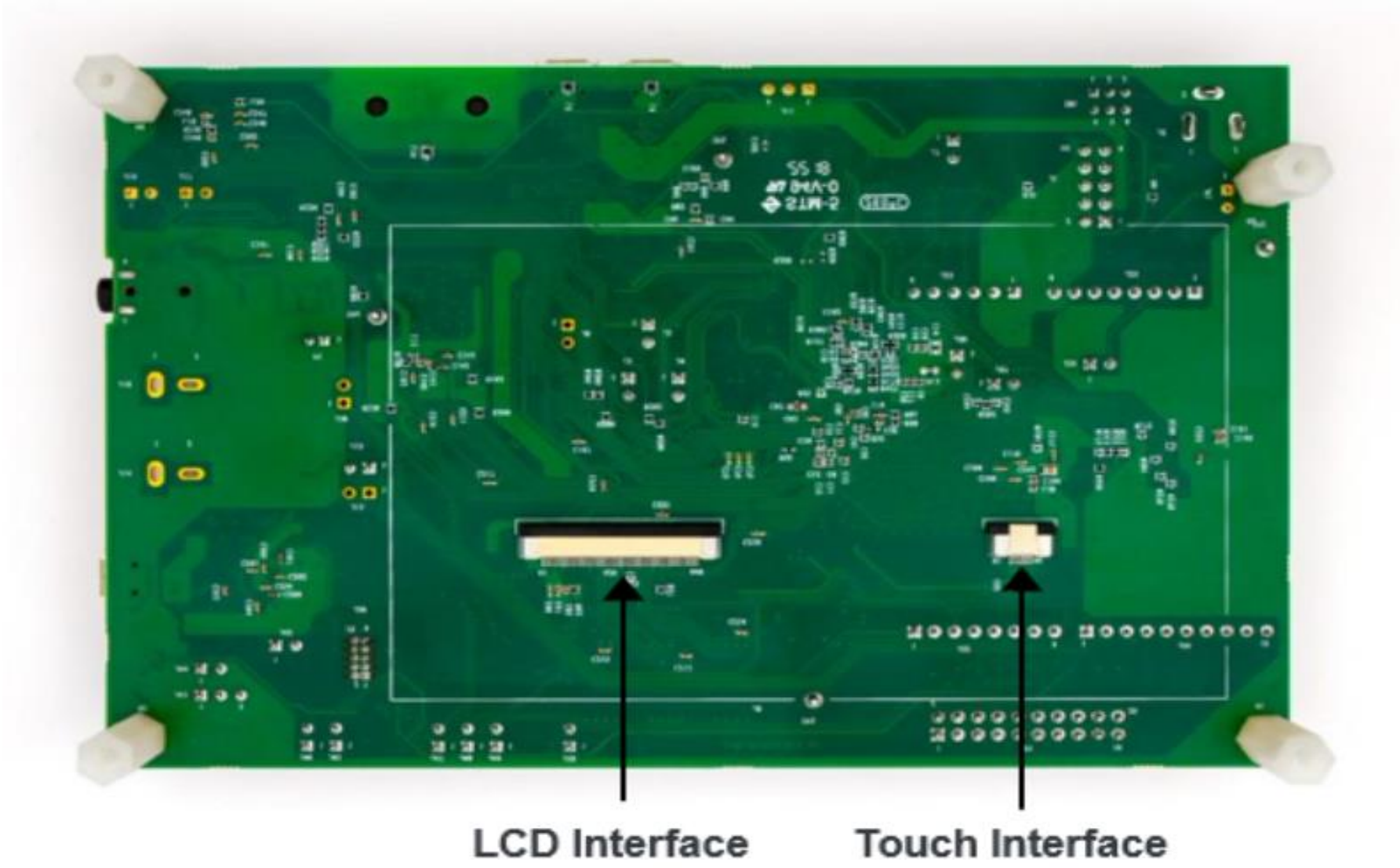
- This lab will cover how to take an existing TensorFlow image classification model, and re-train it to categorize images of flowers. This is known as transfer learning. This updated model will then be converted into a TensorFlow Lite file. By using that file with the TensorFlow Lite inference engine that is part of NXPs eIQ package, the model can be run on an i.MX RT embedded device.
- This lab is used without a camera + LCD, but the flower image will need to be converted to a C array and loaded at compile time.
- This lab is written for the RT1060-EVK. It can also be used with the RT1050-EVKB and RT1064-EVK with minor modifications. Also note that the RT1060-EVK and RT1064-EVK come with a camera sensor. The RT1050-EVKB does not come with a camera sensor. In all cases the LCD must be purchased separately.

IMX RT1060-EVK

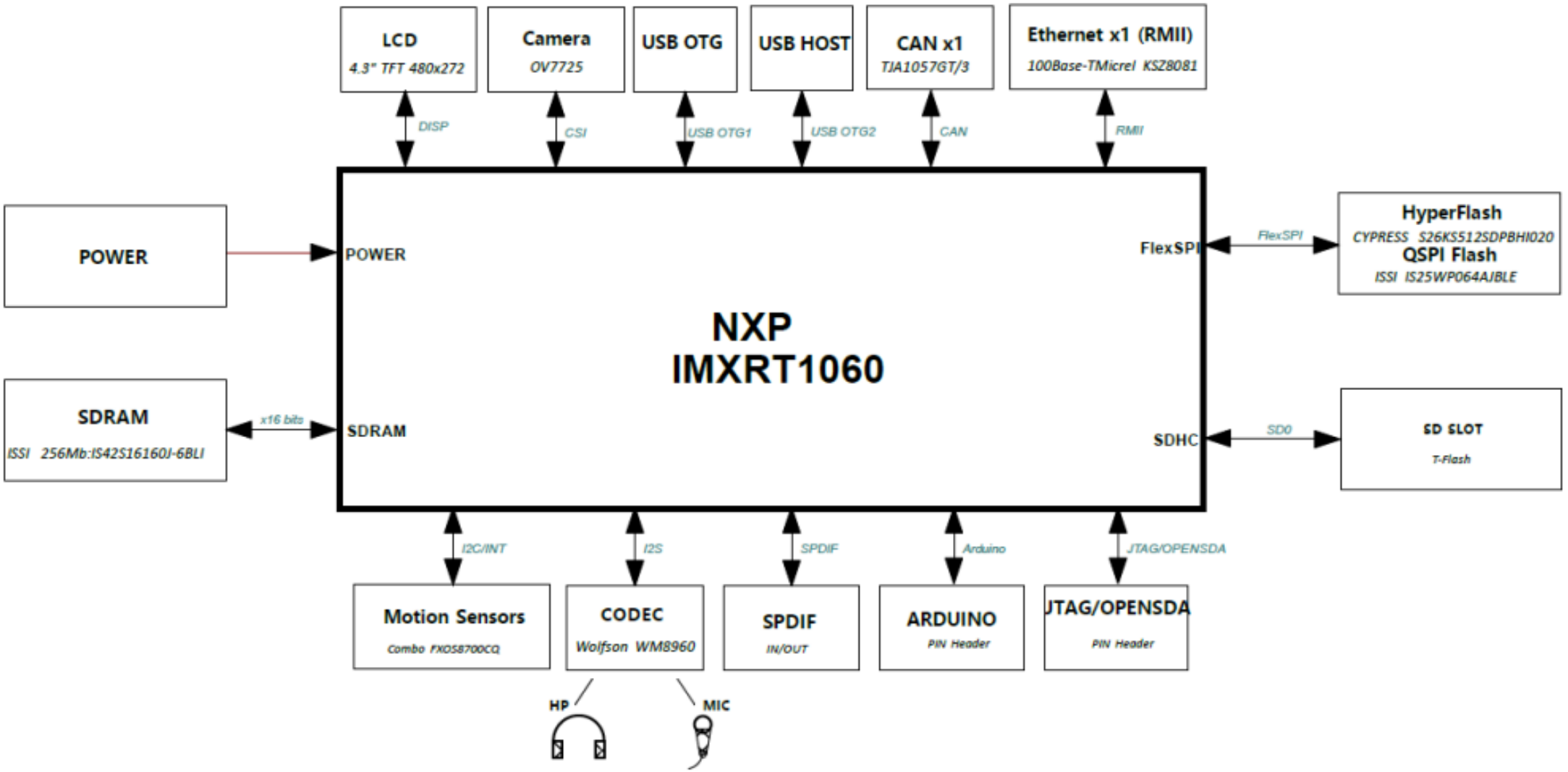
Overview of the MIMXRT1060 EVK board (Front side):



Overview of the MIMXRT1060 EVK board (Back side):



Block diagram:



Board Features:

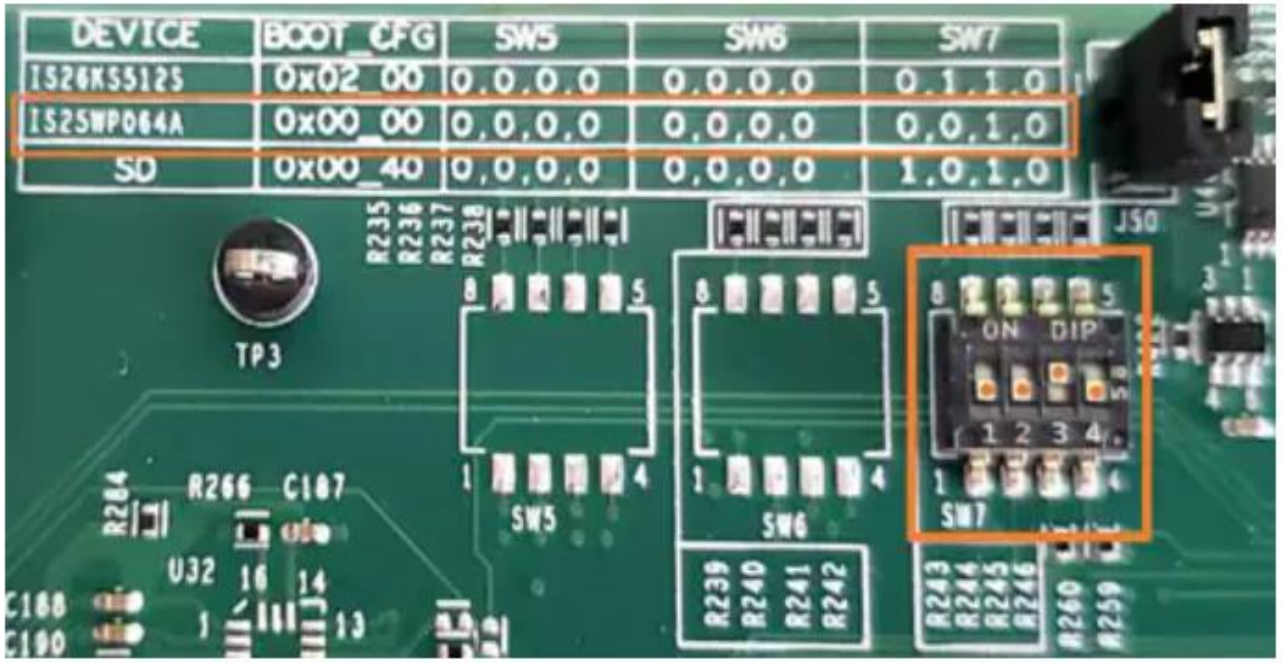
Processor	NXP Processor	MIMXRT10 6 2DVL6A
DRAM Memory	SDRAM 256 Mbit, 166 MHz	IS42S16160J-6BLI
DCDC	MPS	MP2144GJ
LDO	UNION	UM1550S-18 UM1750S-00
Mass Storage	TF Card Slot	
	64 Mbit Quad SPI flash	
	512 Mbit Hyper flash	
Display Interface	LCD connector	
Ethernet	10/100 Mbit/s Ethernet connector. PHY chip: KSZ8081RNB	
USB	USB 2.0 OTG connector	
	USB 2.0 host connector	

Audio Connector	3.5 mm audio stereo headphone jack
	Board-mounted microphone
	Left and right speaker out connectors
	S/PDIF interface (unpopulated)
Power Connector	5 V DC-jack
Debug Connector	JTAG 20-pin connector (SWD by default)
	OpenSDA with DAP-Link
Sensor	FXOS8700CQ: 6-Axis Ecompass (3-Axis Mag, 3-Axis Accel) (Some boards are unpopulated)
Camera	CMOS sensor interface
CAN	CAN bus connector
User Interface Button	ON/OFF, POR Reset, Reset, USER button
LED Indicator	Power Status, Reset, OpenSDA, USER LED
Expansion Port	Arduino interface
PCB	3.937 inch x 5.9055 inch (10 cm x 15 cm), 4-layer board

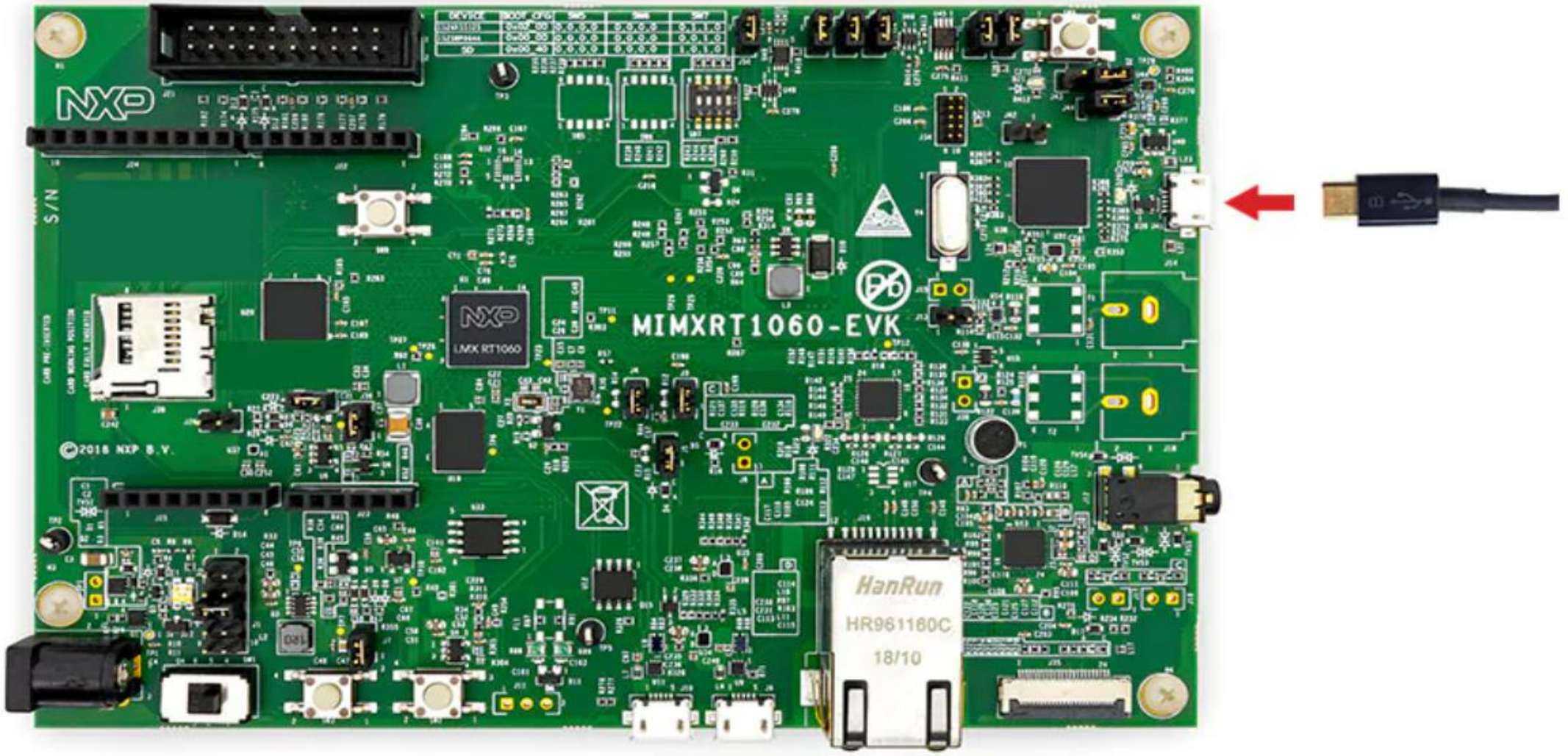
Configure Boot Mode:

The device has four boot modes (one is reserved for NXP use). The boot mode is selected based on the binary value stored in the internal BOOT_MODE register. Switch SW7 is used to select the boot mode on the MIMXRT1060-EVK / EVKB / MIMXRT1064-EVK board.

To boot from the QSPI flash, make sure SW7 is set to 0010.

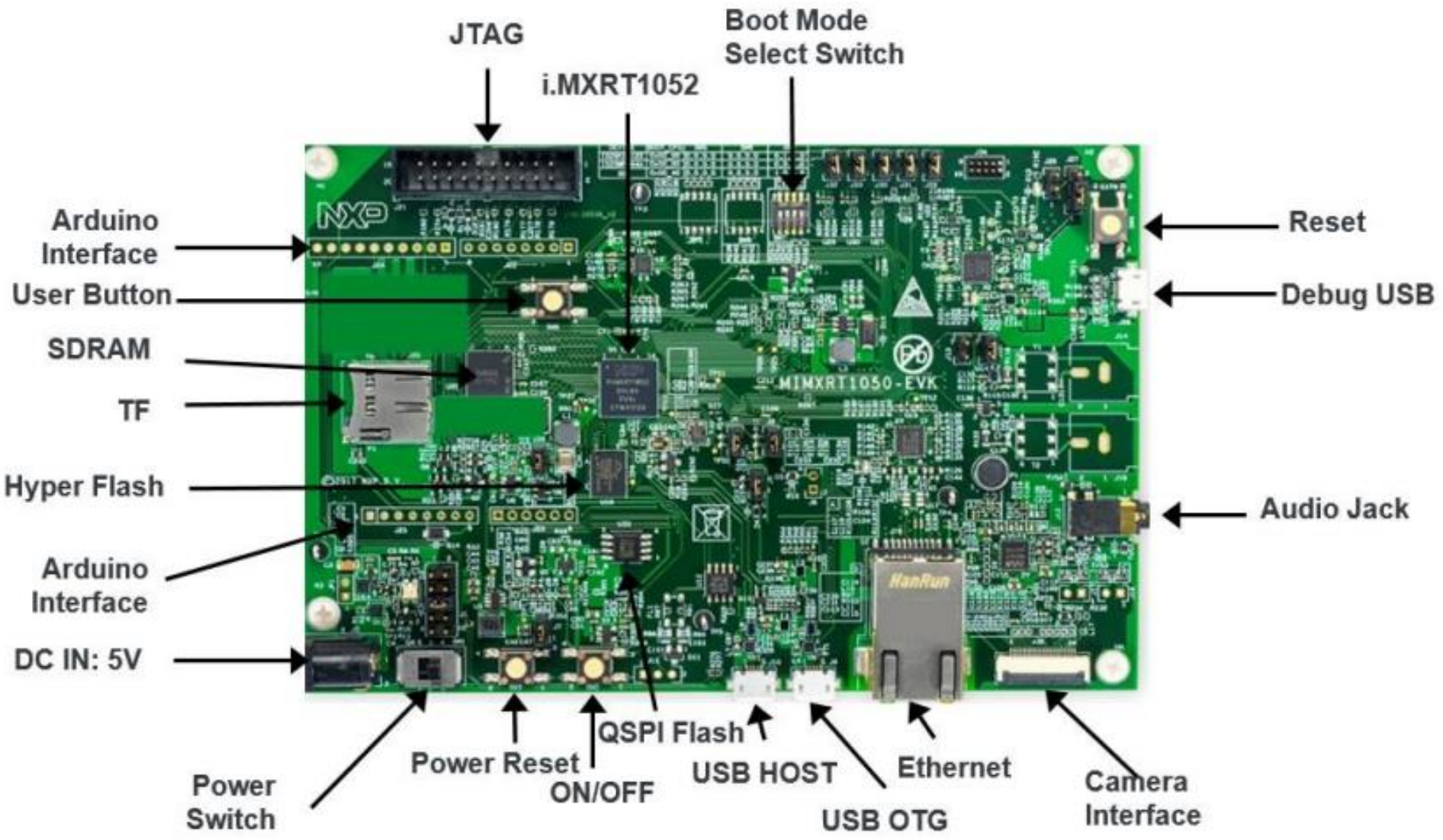


Attach USB Cable:

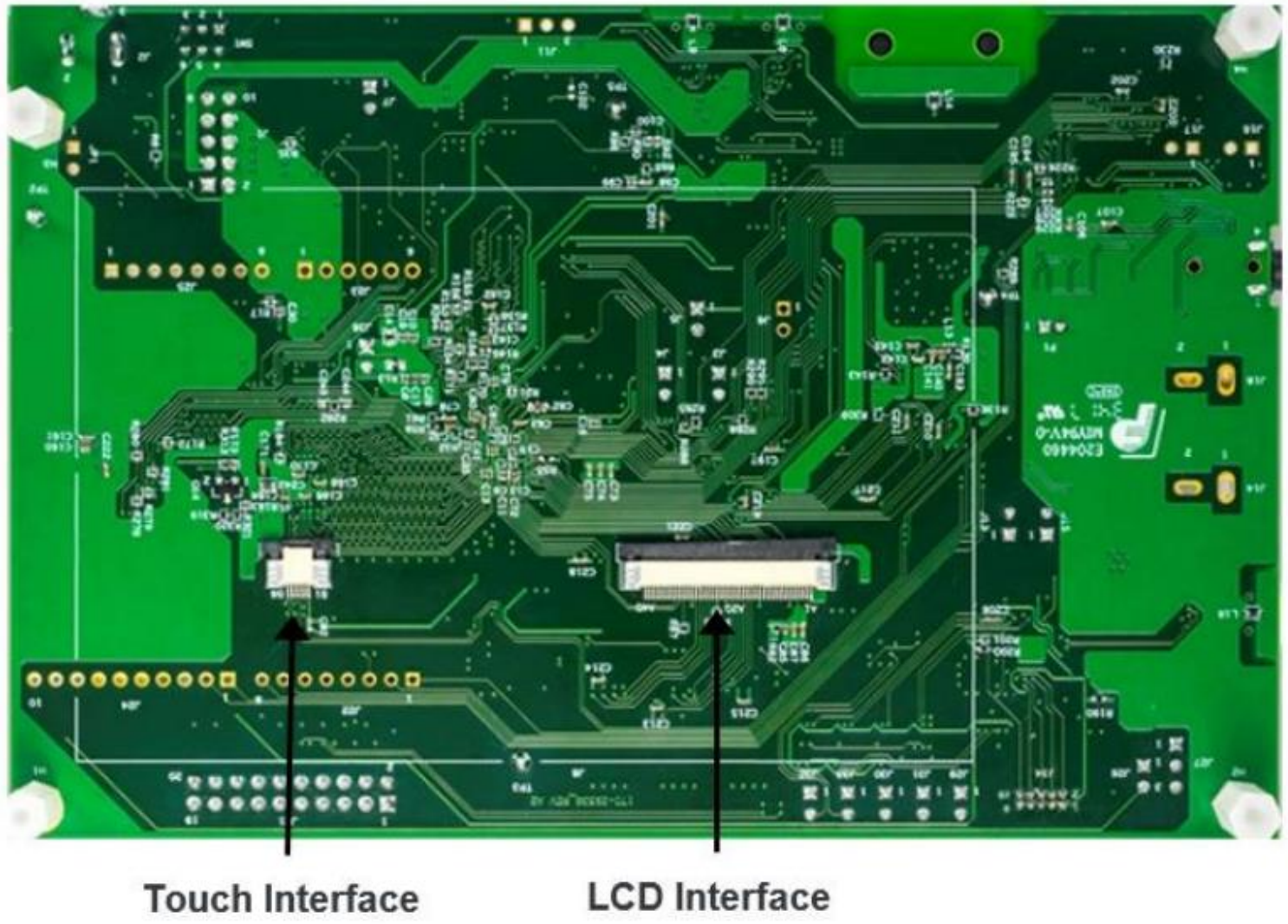


IMX RT1050-EVK

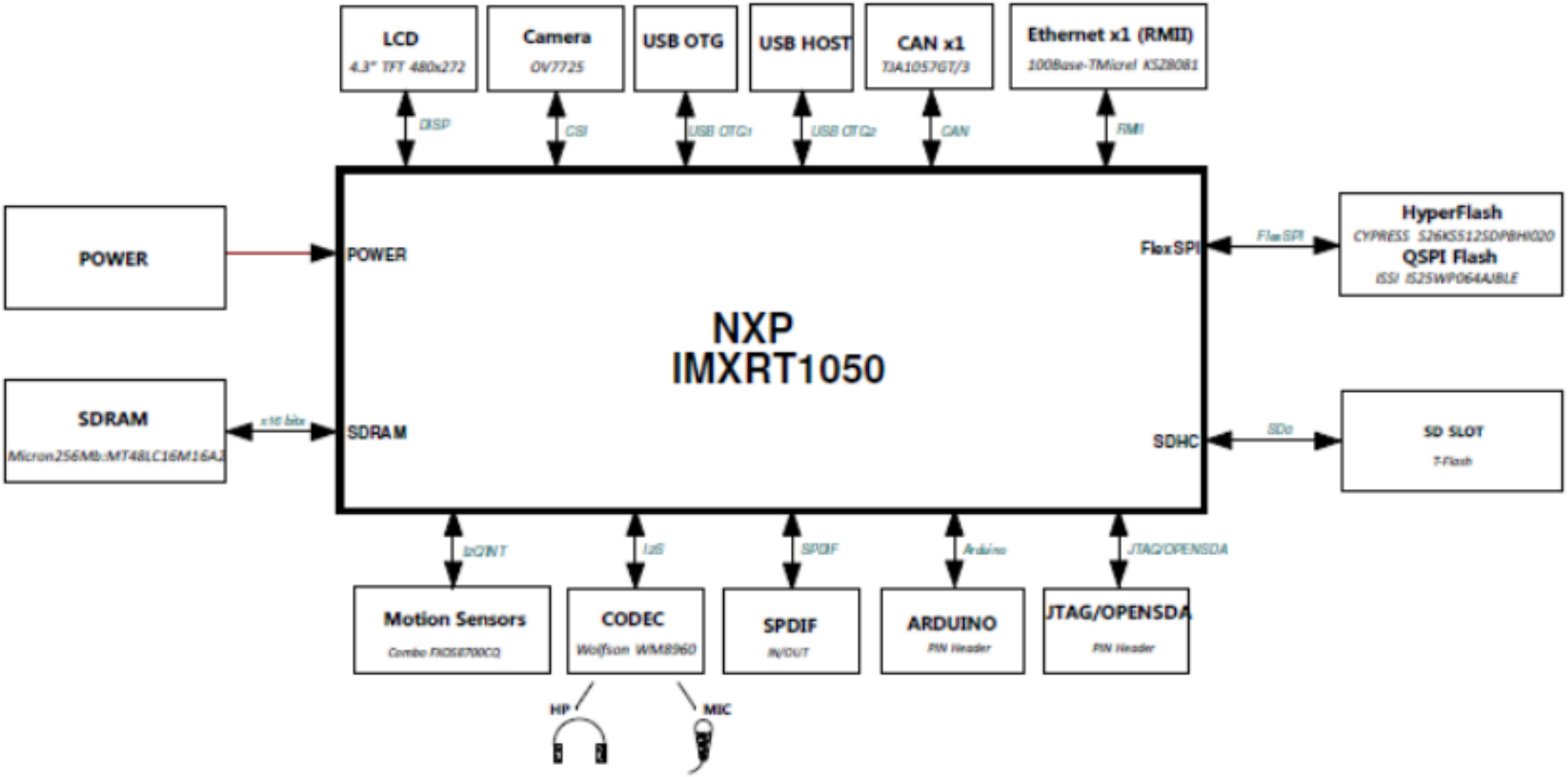
Overview of the MIMXRT1050 EVK board (Front side):



Overview of the MIMXRT1050 EVK board (Back side):



Block diagram:



Board Features:

Processor	NXP Processor	MIMXRT1052DVL6A(A0 silicon) MIMXRT1052DVL6B(A1 silicon)
DRAM Memory	SDRAM 256 Mb, 166MHz	MT48LC16M16A2B4-6AIT: G
DCDC	MPS	MP2144GJ
LDO	UNION	UM1550S-18 UM1750S-00
Mass Storage	TF Card Slot	
	64 Mbit Quad SPI Flash	
	512 Mbit Hyper Flash	
Display Interface	LCD Connector	
Ethernet	10/100 Mbit/s Ethernet Connector. PHY Chip: KSZ8081RNB	
USB	USB 2.0 OTG Connector	
	USB 2.0 Host Connector	

Audio Connector	3.5 mm Audio Stereo Headphone Jack
	Board-Mounted Microphone
	Left & Right Speaker Out Connectors
	SPDIF Interface(unpopulated)
Power Connector	5V DC-Jack
Debug Connector	JTAG 20-pin Connector (SWD by default)
	OpenSDA with DAP-Link
Sensor	FXOS8700CQ: 6-Axis Ecompass (3-Axis Mag, 3-Axis Accel)
Camera	CMOS Sensor Interface
CAN	CAN Bus Connector
User Interface Button	ON/OFF, POR Reset, Reset, USER Button
Led Indicator	Power Status, Reset, OpenSDA, USER LED
Expansion Port	Arduino Interface
PCB	3.937-inch x 5.9055-inch (10cm x 15cm), 4-layer board

Configure Boot Mode:

- The device has four boot modes (one is reserved for NXP use). The boot mode is selected based on the binary value stored in the internal BOOT_MODE register. Switch (SW7-3 & SW7-4) is used to select the boot mode on the MIMXRT1050 EVK Board.
- Enable Hyper Flash Boot mode to see the output in TeraTerm.

SW7-1	SW7-2	SW7-3	SW7-4	Boot Device
OFF	ON	ON	OFF	Hyper Flash
OFF	OFF	ON	OFF	QSPI Flash
ON	OFF	ON	OFF	SD Card

Software and Hardware Installation

MCUXpresso

Install the MCUXpresso IDE:

- Install the latest version of MCUXpresso IDE
- Use the link:

https://www.nxp.com/design/software/development-software/mcuxpresso-software-and-tools-/mcuxpresso-integrated-development-environment-ide:MCUXpresso-IDE?tab=Design_Tools_Tab

- It will ask you to log in to NXP. If you don’t have an account in NXP, create one and proceed.

MCUXpresso Integrated Development Environment (IDE)

FOLLOW

Overview

The MCUXpresso IDE brings developers an easy-to-use Eclipse-based development environment for NXP® MCUs based on Arm® Cortex®-M cores, including its general purpose crossover and wireless - enabled MCUs. The MCUXpresso IDE offers advanced editing, compiling, and debugging features with the addition of MCU-specific debugging views, code trace and profiling, multicore debugging, and integrated configuration tools. The MCUXpresso IDE debug connections support

More

USER GUIDE DOWNLOAD

Features

- A complimentary, unlimited code size, easy-to-use IDE
- Advanced editing, compiling and editing with syntax coloring, MCU-specific debugging views, code trace and profiling
- Use built-in SDK selection tool, or drag and drop pre-built packages made with SDK Builder
- Ubuntu 18.04 LTS / 20.04.2 LTS, Github project development support

Tera Term

Install Tera Term:

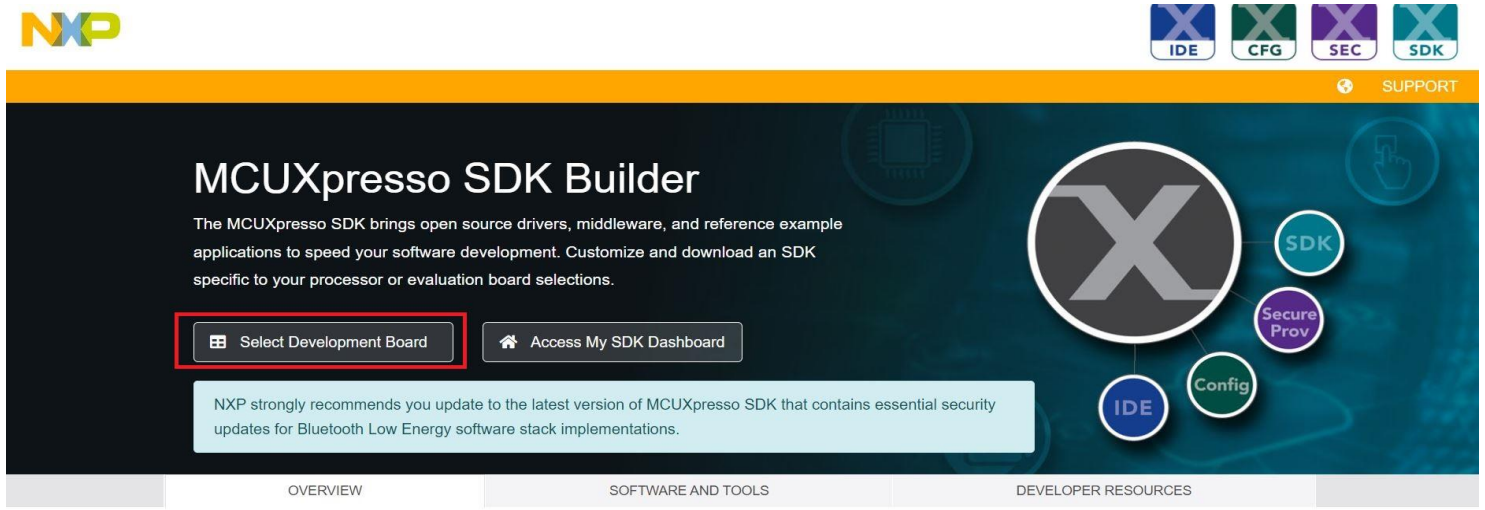
- Tera Term is the terminal emulator for Microsoft Windows, that supports serial port, telnet and SSH connections.
- Among many other features it also has built-in Macro scripting language.
- Tera Term is often used to automate tasks related to remote connections initiated from PC.
- Install the latest version of Tera Term
- Use the link:

https://ttssh2.osdn.jp/index.html.en

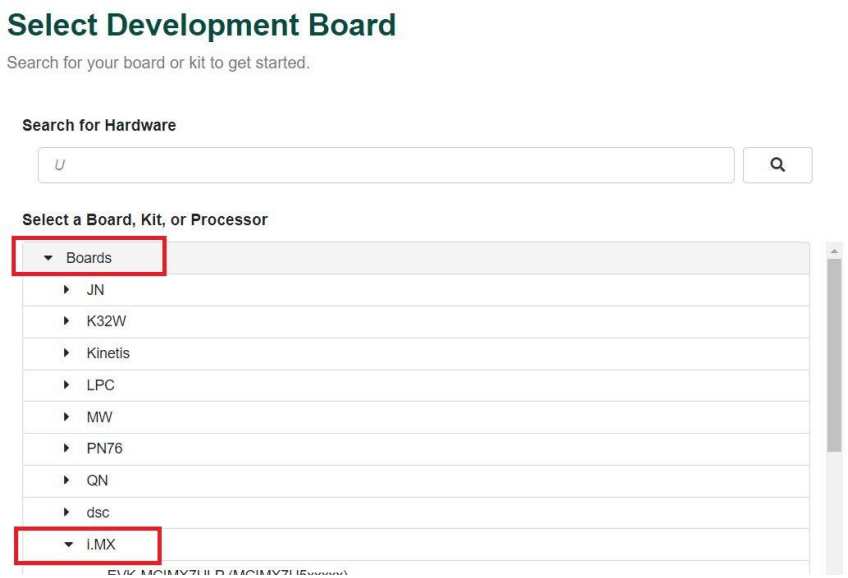
MCUXpresso SDK

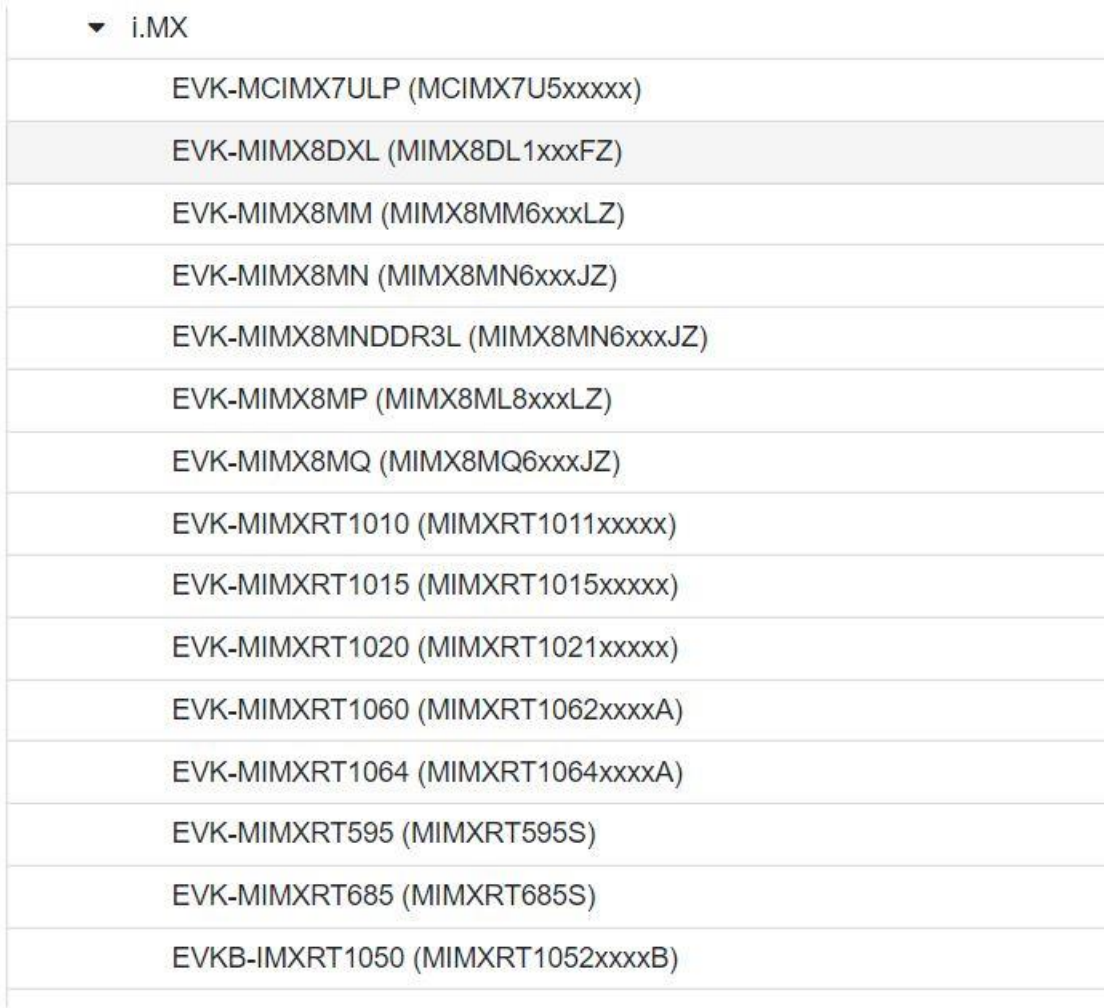
Install the MCUXpresso SDK:

- The MCUXpresso SDK is complimentary and includes full source code under a permissive open-source license for all hardware abstraction and peripheral driver software.
- Download MCUXpresso SDK 2.6.2 version for i.MXRT1060 (or) SDK 2.6.0 version for i.MXRT1050 . It includes the eIQ software platform and demos:
- Use the link: https://mcuxpresso.nxp.com/en/welcome
- Click on select development board

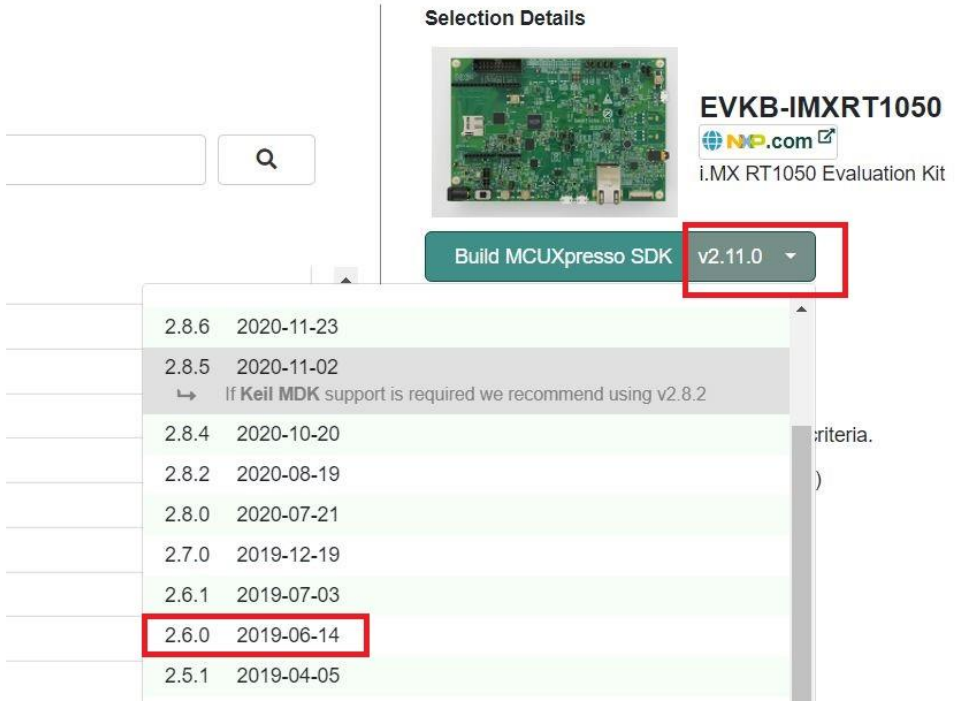


- Select boards>i.MX> and your board name(EVK-MIMXRT1060 or EVK-MIMXRT1050)

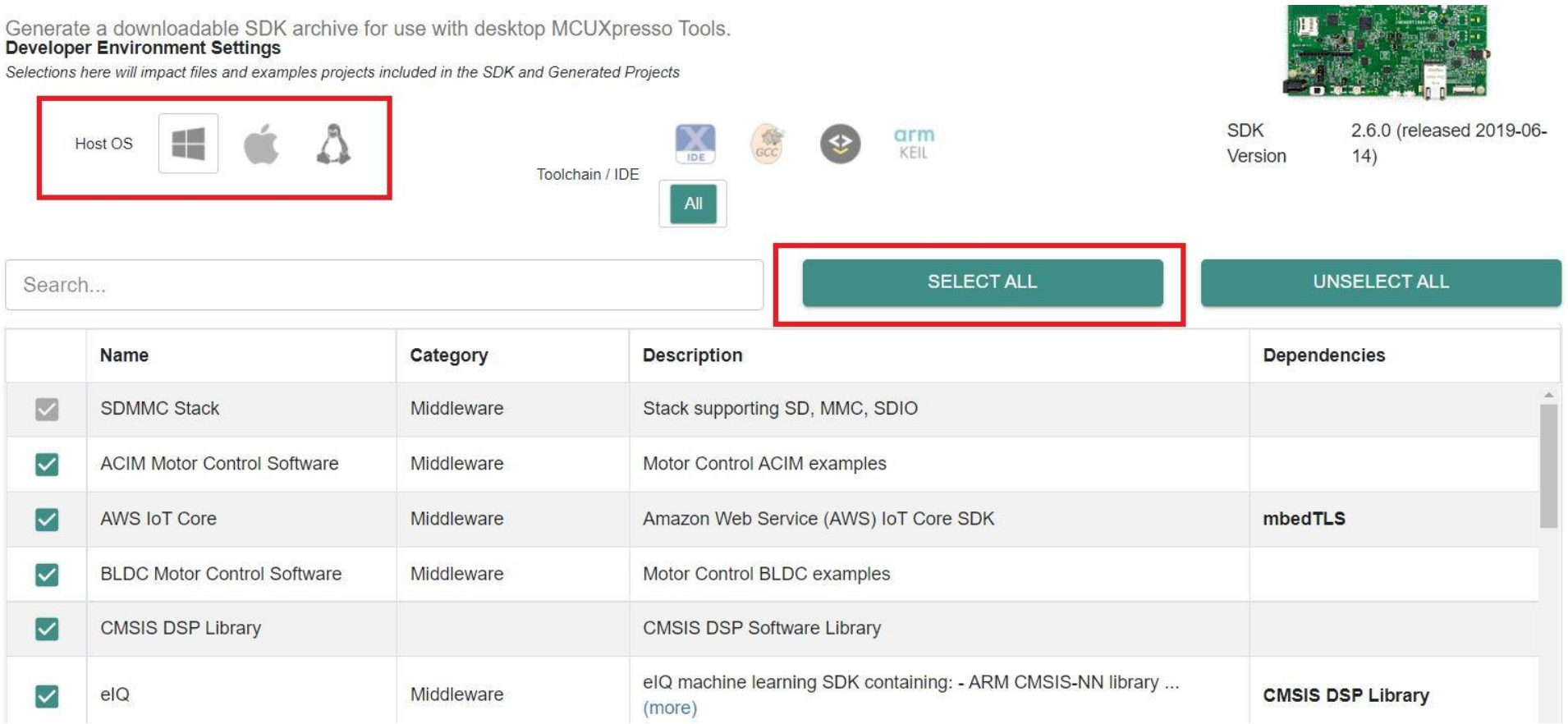




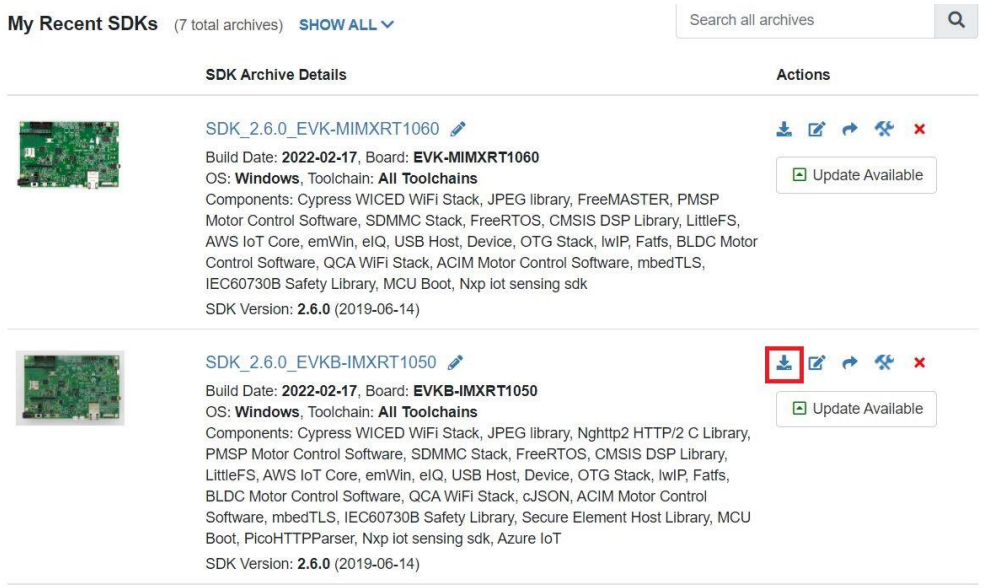
- It leads to select board details in right most corner of the page. There click on version tab
- For IMXRT1060: Select 2.6.2 version
- For IMXRT1050: Select 2.6.0 version
- After that, click on build MCUXpresso SDK.



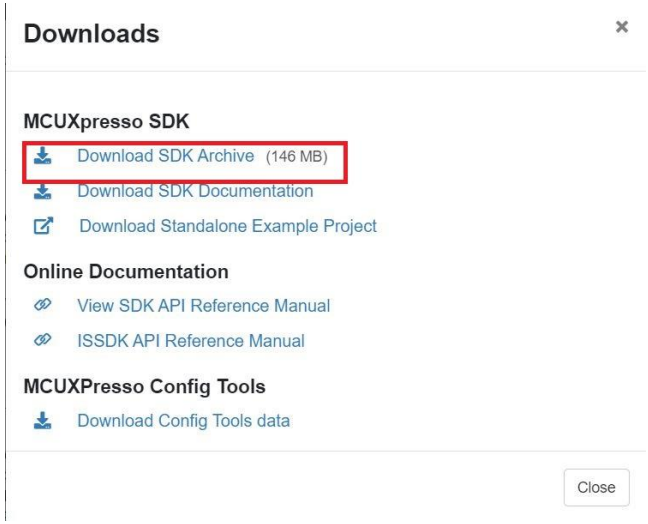
- Then select you Host OS and click on select all icon which will select all the software components related to that board.
- After that go to bottom of the page and click on download SDK



- This will build the required SDK and after building we will be able to download it.



- Then select download SDK archive. Agree the terms and condition. Now SDK archive will be downloaded.



TensorFlow

Install TensorFlow:

- TensorFlow makes it easy to create machine learning models for desktop, mobile, web, and cloud
- TensorFlow provides a collection of workflows to develop and train models using Python or JavaScript, and to easily deploy in the cloud, on-prem, in the browser, or on-device no matter what language you use.
- Download and install Python latest version. The 64-bit edition is required.
- Use the link:

https://www.python.org/downloads/

Python | PSF | Docs | PyPI | Jobs | Community

python™

Donate | Search | GO | Socialize

About | Downloads | Documentation | Community | Success Stories | News | Events

Download the latest version for Windows

Download Python 3.10.2

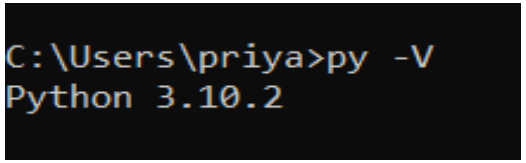
Looking for Python with a different OS? Python for Windows, Linux/UNIX, macOS, Other

Want to help test development versions of Python? Prereleases, Docker images

Looking for Python 2.7? See below for specific releases

- Verify that the python command corresponds to Python 3.x.Type the Command in command prompt :

  py –V



- Update the python installer tools. Type the command in command prompt:

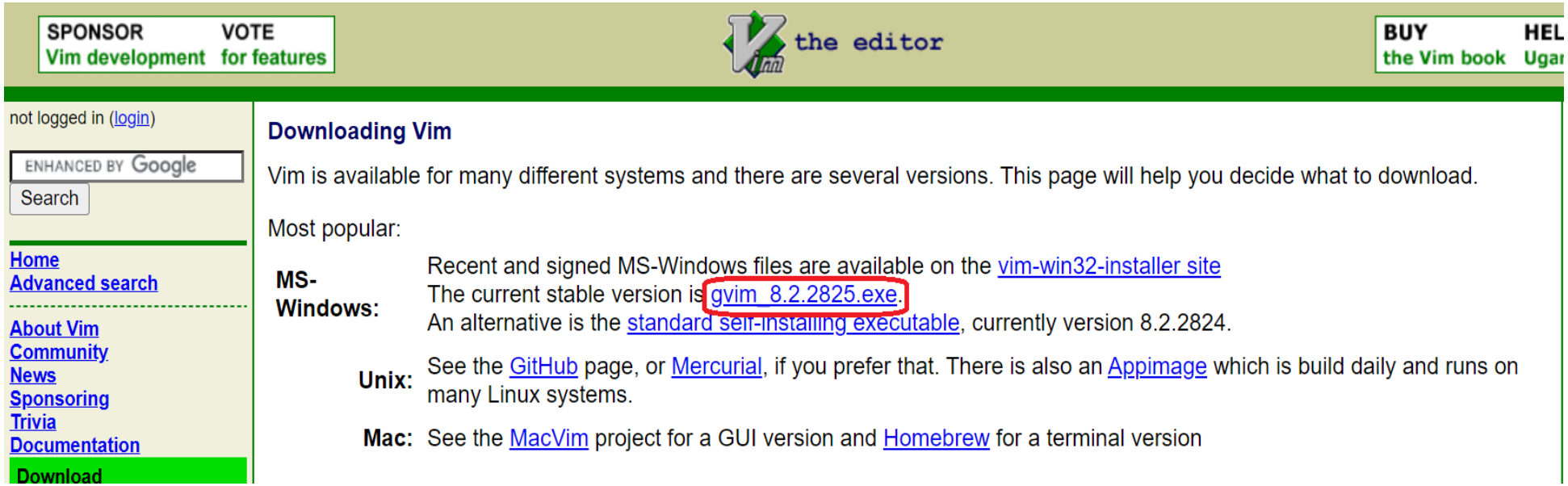
  py -m pip install -U pip

  py -m pip install -U setuptools

- Install the latest version Tensorflow libraries and support for python. Type the command in command prompt:

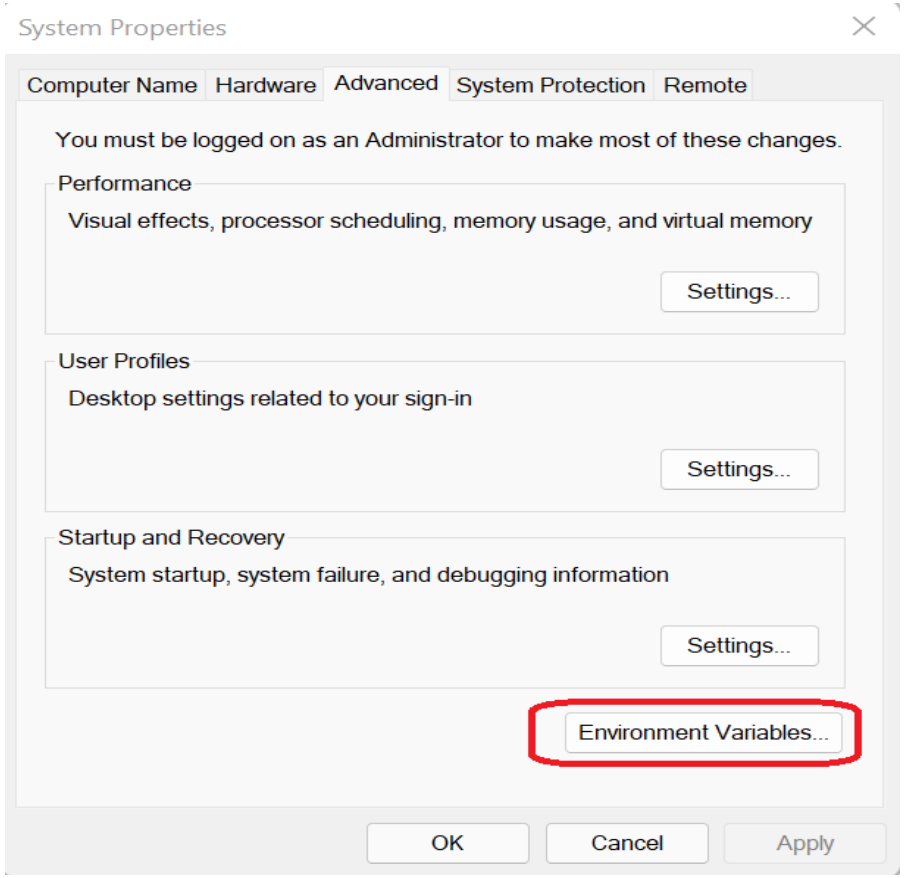
  py -m pip install tensorflow

- Install other useful python packages. Type the command in command prompt:
  py -m pip install tensorflow-datasets
  py -m pip install numpy scipy matplotlib ipython jupyter pandas sympy nose
  py -m pip install opencv-python
  py -m pip install PILLOW
  py -m pip install netron
- If on Windows, install latest Vim using the link: https://www.vim.org/download.php#pc. There is a binary convertor programmed named xxd.exe located inside that package that will be needed.

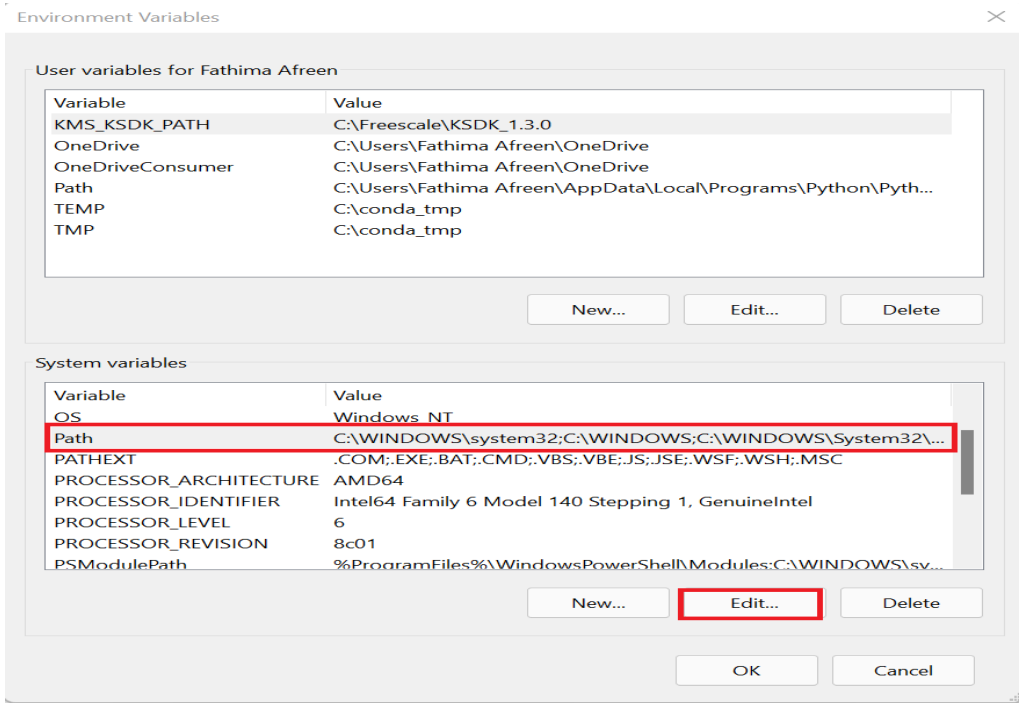


- If on Windows, add the following directories to your executable PATH if they are not already. Steps to add path is:

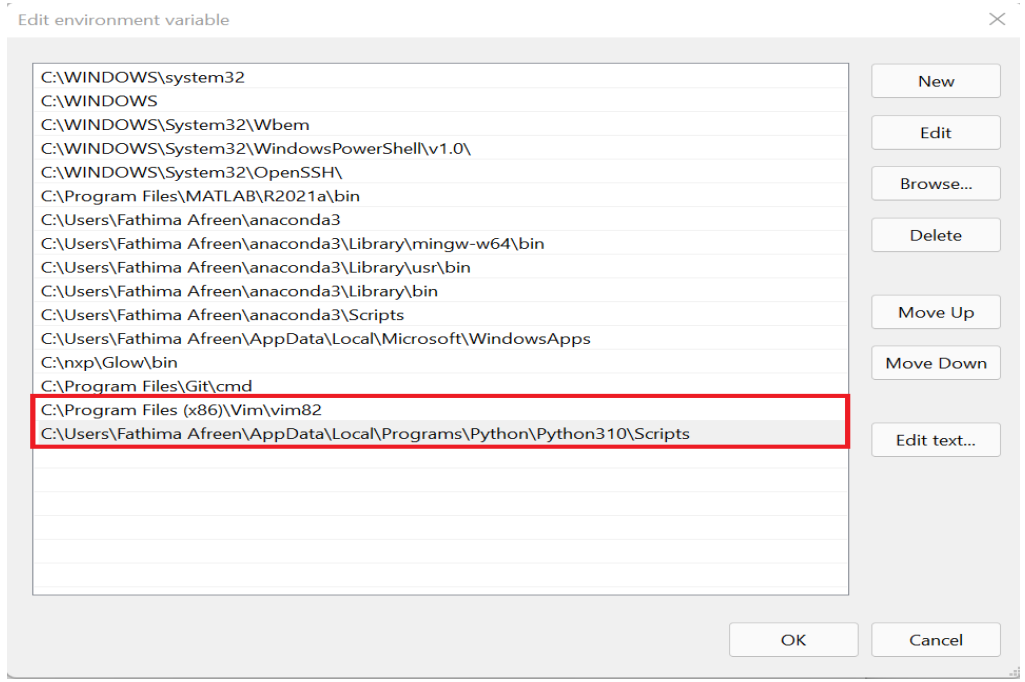
  Go to system properties>Environment Variables



Under System Variables, select path and edit.



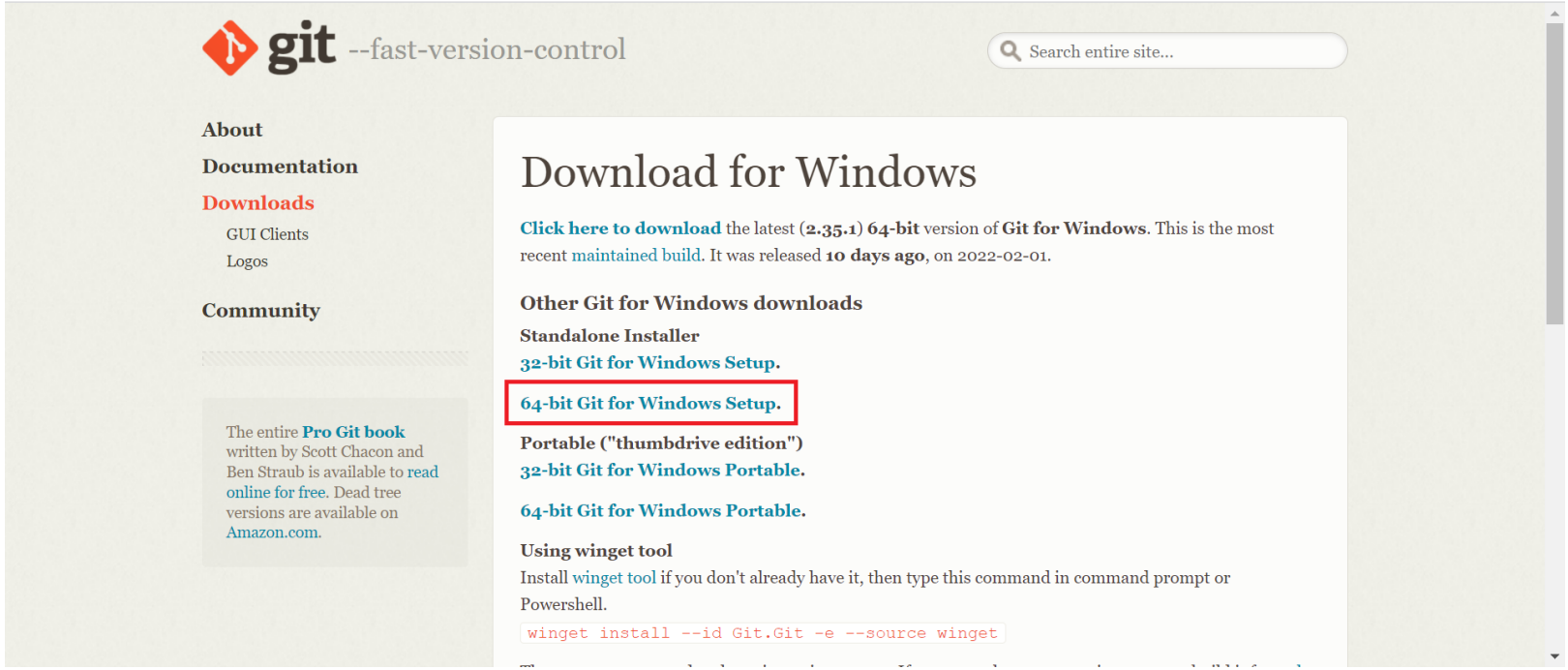
Add path as:
<python_install_directory>/scripts
<vim_install_directory>



Lab Scripts Installation

- The python scripts that will be used to retrain an already existing model will be downloaded via Git.
- Experiment: We'll be retraining the model to recognize photos of flowers and categorize them into different types(5 category)- daisy, dandelion, roses, sunflowers and tulips
- The new flower data that the model will be retrained on also will be download.
- Steps are:

1. Install Git: https://git-scm.com/downloads



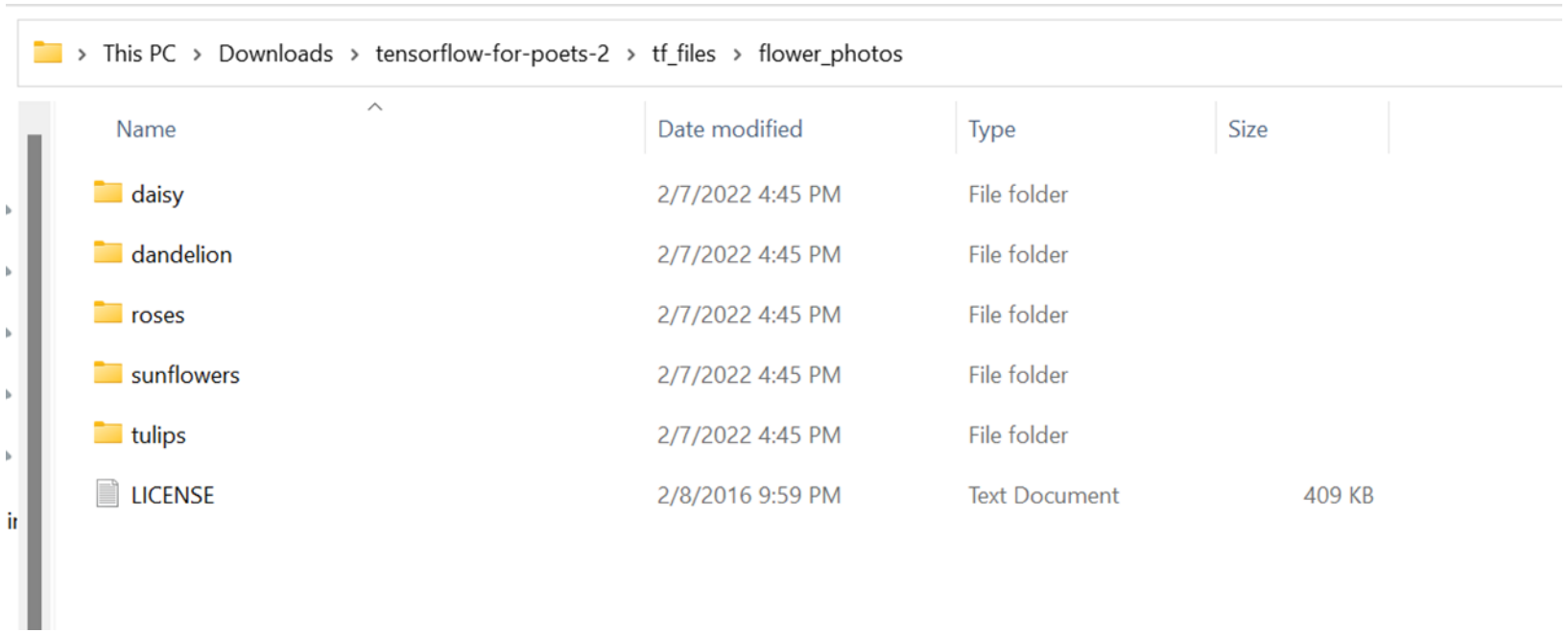
2. Open a command prompt, and in a directory of your choosing, download the tutorial repository with git:
git clone https://github.com/googlecodelabs/tensorflow-for-poets-2

3. Download a set of Creative Commons licensed flowers images that have already been categorized into 5 different classes:
http://download.tensorflow.org/example_images/flower_photos.tgz

4. Unzip that file which will create a “flower_photos” directory:

a. If on Windows, you may need to install 7-zip or Winzip to unzip the .tgz file.
b. If on Linux, use: tar -xvzf flower_photos.tgz

5. Place the “flower_photos” directory inside the “tf_files” directory that is in the tensorflow-for-poets-2 repo downloaded in the first step. It should look like the following when done:



Retrain Existing Model

- For this lab we will retrain an already existing model with new data. This is called transfer learning. The structure of the model has already been setup for image classification, so the goal is to retrain one layer to classify new images with new custom labels. This greatly shortens the amount of time it will take to train the model. Once retrained, the model can be converted to TensorFlow Lite format and ran on the i.MXRT device. The following steps are based on this Google CodeLabs tutorial: TensorFlow for Poets

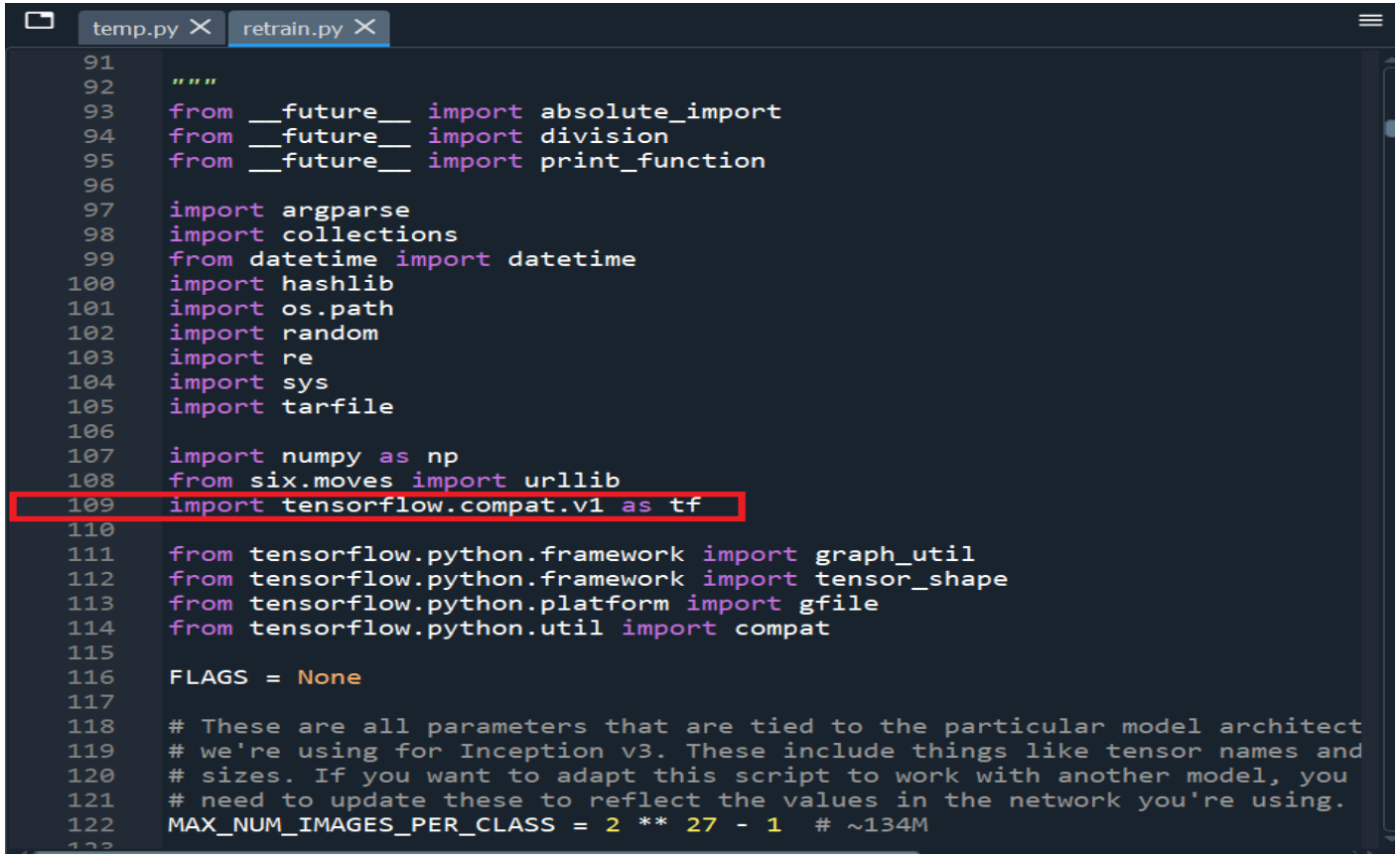
https://codelabs.developers.google.com/codelabs/tensorflow-forpoets/index.html/

  - 1. Open a command prompt and go to the directory that was created by cloning the git repository in the last section. It should be something like this:

    cd C:\eiq\tensorflow-for-poets-2

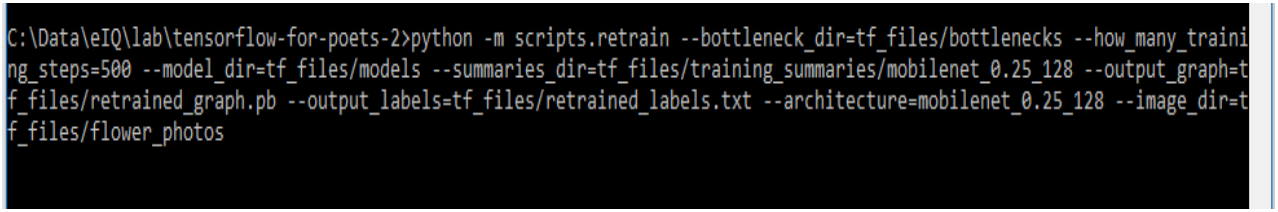
2. The model being used is called MobileNet.

Open retrain.py file changed 109 line i.e, earlier it was import tensorflow as tf we changed it to import tensorflow.compat.v1 as tf. The image after changing is shown below.



We will retrain this model for 128x128 pixel images using a python script found inside the tutorial folder. Navigate to the main root directory and run the following command.

This should be one long continuous line like in the image below:



python -m scripts.retrain
--bottleneck_dir=tf_files/bottlenecks
--how_many_training_steps=500
--model_dir=tf_files/models/
--summaries_dir=tf_files/training_summaries/mobilenet_0.25_128
--output_graph=tf_files/retrained_graph.pb
--output_labels=tf_files/retrained_labels.txt
--architecture=mobilenet_0.25_128
--image_dir=tf_files/flower_photos

3. This will take several minutes to run. While waiting, here's an explanation for the arguments in the command you just ran:

--bottleneck_dir=tf_files/bottlenecks

Directory to store cached data information

--how_many_training_steps=500

# of iterations. The more iterations the longer the training takes, but the more accurate the model will likely be

--model_dir=tf_files/models/

Directory location to download the pre-existing model

--summaries_dir=tf_files/training_summaries/mobilenet_0.25_128

Output directory for data files used by an optional analysis program called TensorBoard

--output_graph=tf_files/retrained_graph.pb

Name of the retrained model in Protocol Buffer (pb) format.

--output_labels=tf_files/retrained_labels.txt

Text file with the labels for the model (determined by the names of the sub-directories of the training data)

--architecture=mobilenet_0.25_128

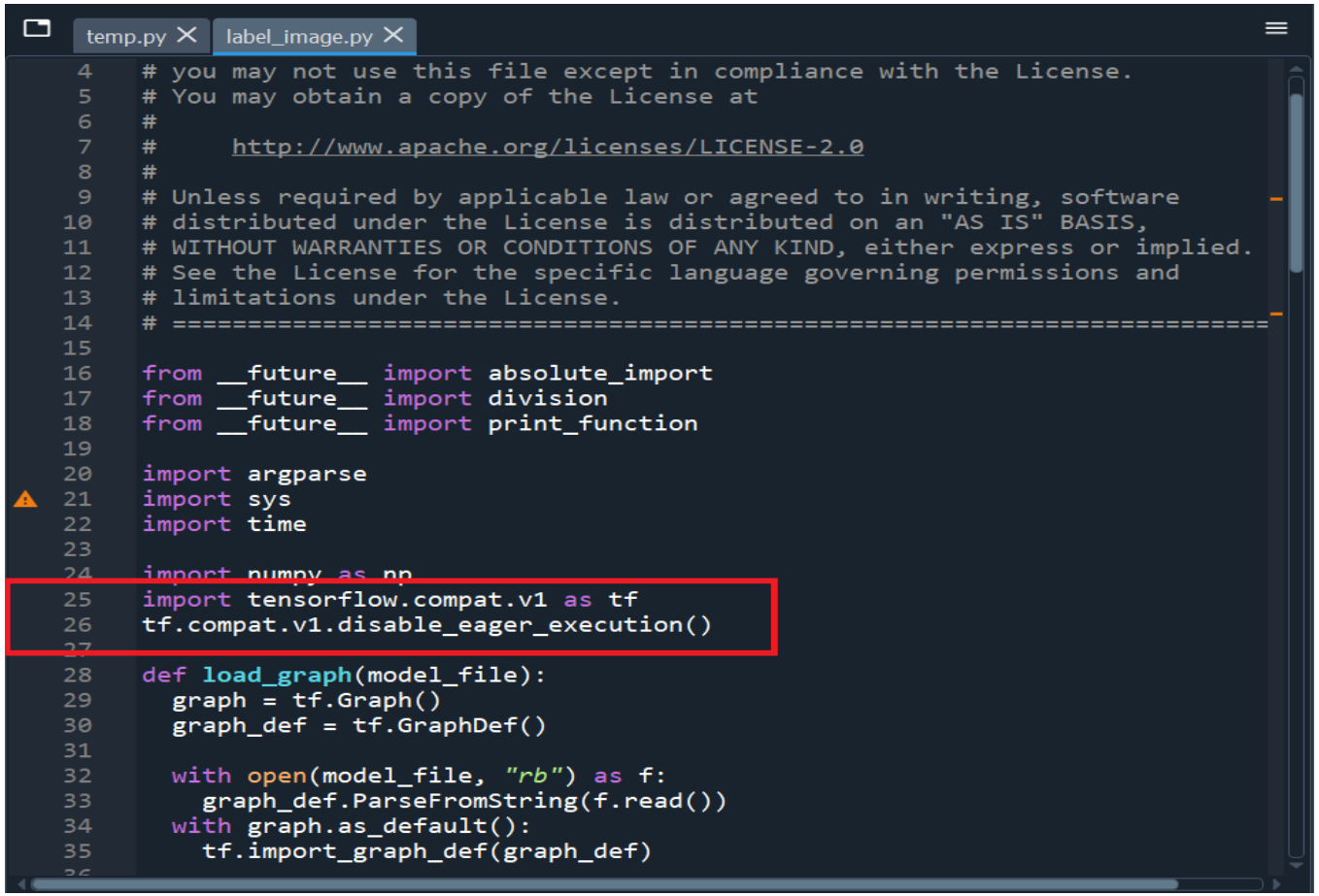
The particular type of Mobilenet model to use as a starting point

--image_dir=tf_files/flower_photos

Location of the data to train the model on. Each sub-directory is a label classifying those images

4. Once finished, look inside the tf_files directory. You should have a model file named retrained_graph.pb. This is the newly trained model.

5. You can test this model file against flower images using the label_image python script. Open label_image python script and change import tensorflow as tf to import tensorflow.compat.v1 as tf, tf.compat.v1.disable_eager_execution () at line 25 and 26. See below figures for reference.



- From the main directory, call a script to use the new graph and have it analyze a daisy image with the following command as one long continuous line:

python -m scripts.label_image

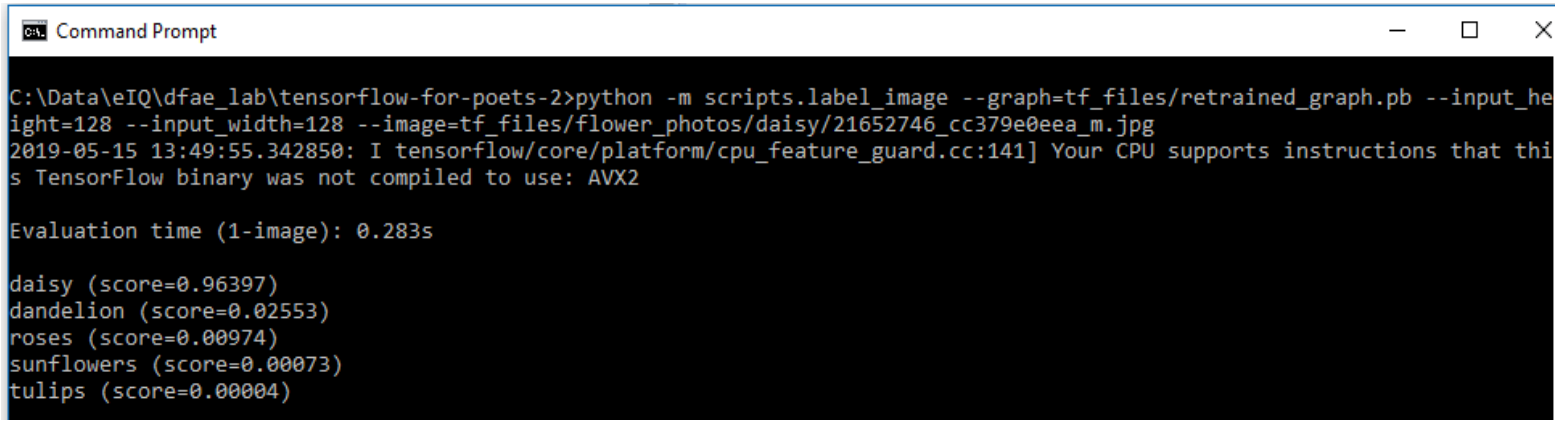
--graph=tf_files/retrained_graph.pb

--input_height=128

--input_width=128

--image=tf_files/flower_photos/daisy/21652746_cc379e0eea_m.jpg

6. It should respond back that that photo is of a daisy. The confidence level may vary slightly as the model training will be slightly different each time. If you see a low confidence level (<.80) for identifying that image as a daisy, try running the retraining script again.



Convert Model and Data

Now that the retrained model is running on your laptop, the next step is to use a TensorFlow utility named tflite_convert to convert that model into a file that can be loaded onto an embedded device like the i.MXRT1060/1050.

Convert TensorFlow model:

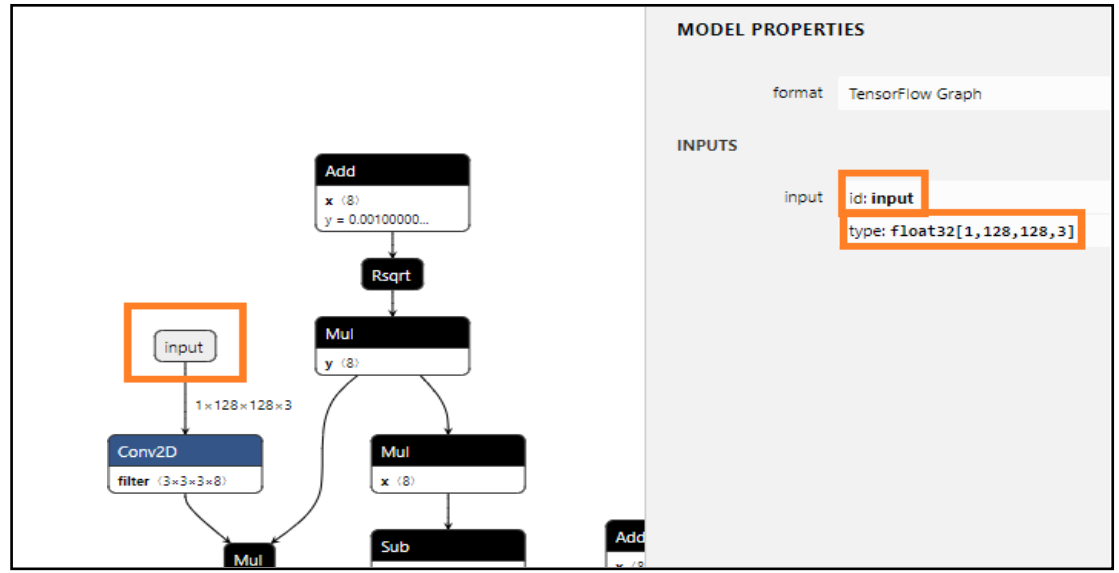
Convert the .pb model file into a format that can be imported into the RT1060 project with **tflite_convert.**

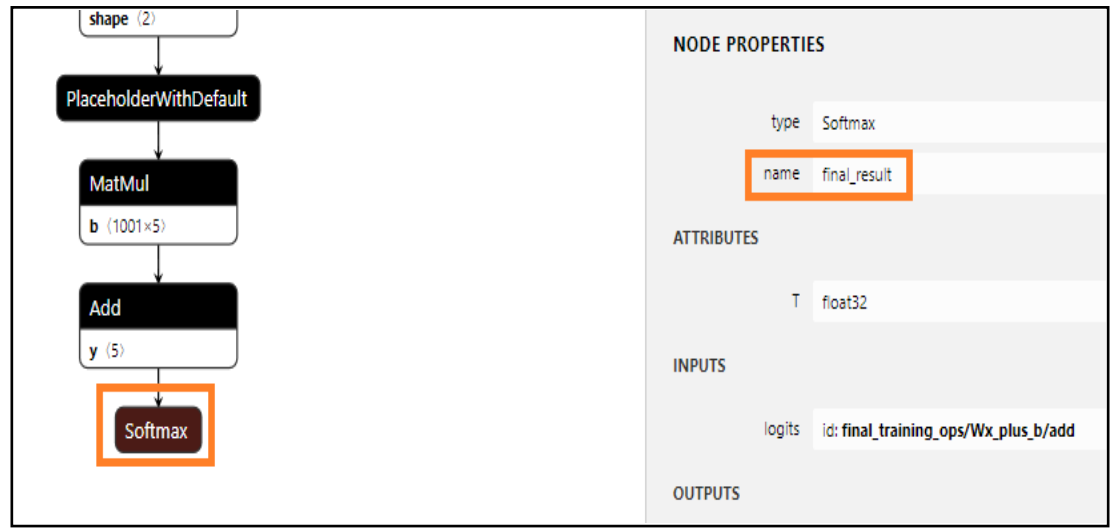
1. The first and last layer names need to be read from the TensorFlow .pb model file, which will be used later in the conversion process. This can be done with a neural network visualization tool called Netron.

a. Use Netron by using the following command on a .pb file:

netron tf_files\retrained_graph.pb

b. While the command is running, open up a web browser and navigate to http://localhost:8080 and click on the first and last nodes to get the layer names and the input shape.


Add
x (8)
y = 0.00100000...
Rsqrt
input
Mul
y (8)
1×128×128×3
Conv2D
filter (3×3×3×8)
Mul
x (8)
Mul
Sub
Add
MODEL PROPERTIES
format
TensorFlow Graph
INPUTS
input
id: input
type: float32[1,128,128,3]


shape (2)
PlaceholderWithDefault
MatMul
b (1001×5)
Add
y (5)
Softmax
NODE PROPERTIES
type
Softmax
name
final_result
ATTRIBUTES
T
float32
INPUTS
logits
id: final_training_ops/Wx_plus_b/add
OUTPUTS

c. After reading the labels, go back to the terminal window and hit Ctrl+c to stop the server and return to the command prompt

2. Use tflite_convert to transform the model file into a .tflite file. There are options available to quantize and optimize the model during this step, but they seem to significantly decrease the accuracy of the converted model. The following options keep the accuracy about the same as the full model. Type in the command as one long continuous line like below:

tflite_convert
--graph_def_file=tf_files/retrained_graph.pb
--output_file=tf_files/retrained_graph.tflite
--input_shape=1,128,128,3
--input_array=input
--output_array=final_result
--inference_type=FLOAT
--input_data_type=FLOAT
--enable_v1_converter

Here is what each of those arguments determine:

--graph_def_file=tf_files/retrained_graph.pb

Name of the TensorFlow model to convert

--output_file=tf_files/retrained_graph.tflite

Name of the converted tflite file

--input_shape=1,128,128,3

This model takes in a 128 x 128 image that has 3 color channels

--input_array=input

Name of the first layer of the model

--output_array=final_result

Name of the last layer of the model

--inference_type=FLOAT

Model uses floating (instead of 8-bit quantized) inference

--input_data_type=FLOAT

Model uses floating (instead of 8-bit quantized) input

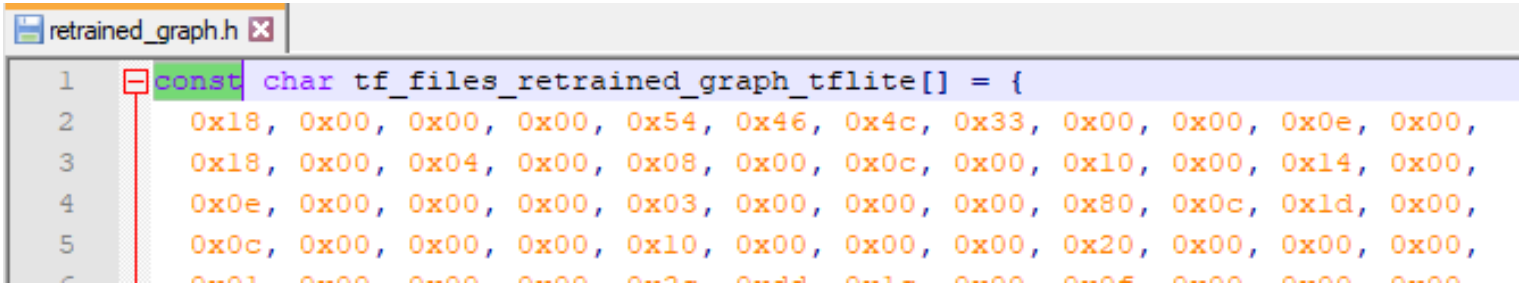
3. You may get a warning about AXV2 instructions not being supported. You can ignore this warning.

4. Inside the tf_files directory you should see a new file named **retrained_graph.tflite**

5. Use the **xxd** utility to convert the .tflite binary file into a C array that can be imported into an embedded project. Do not run this command in Powershell as it will cause compilation issues later. Run it in a standard command prompt:

xxd -i tf_files/retrained_graph.tflite > tf_files/retrained_graph.h

6. The generated header file may need to be modified slightly to define it as a const. Open up the file and, if necessary, change “unsigned char” to “const char”. Also make note of the array name as it will be used in the next section.



7. The following files should now be in the tf_files directory:

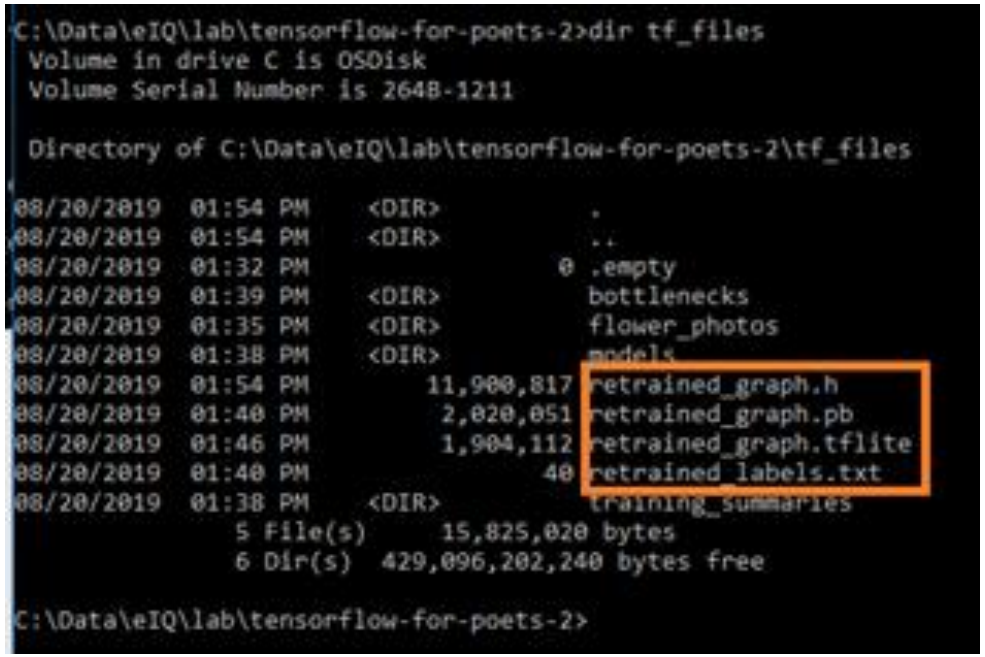


Image Data Conversion – No Camera Only:

Now that the model has been converted, an image also needs to be converted into a C array. The Label Image example in eIQ takes in 24-bit BMP image files, so will convert one of the images in the dataset to a BMP file and then convert that into a C array.

8. Convert one of the JPEG flower image files into a 24-bit BMP file. Preferably pick an image that was labeled already using the label_image python script so that result can be compared to the result on the RT1060. In this case we'll pick the file

tf_files/flower_photos/daisy/21652746_cc379e0eea_m.jpg.

Save the new .bmp file as daisy.bmp

• In Linux use:

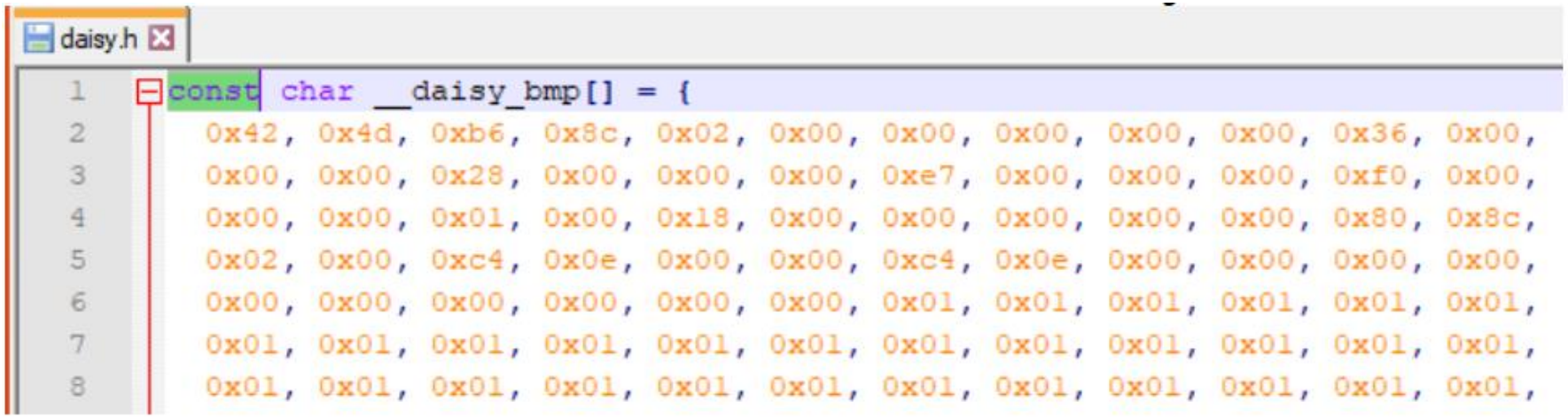
convert tf_files/flower_photos/daisy/21652746_cc379e0eea_m.jpg -type truecolor tf_files/daisy.bmp

• In Windows use a paint program like MS Paint.

9. After converting the .jpg file to a .BMP file, use the xxd program to convert it to a C array. Do not run this command in Powershell as it will cause compilation issues later. Run it in a standard command prompt:

xxd -i daisy.bmp > daisy.h

10. Open the generated header file and, if necessary, change the array type from “unsigned char” to “const char” and make note of the array name, as that name will be used in the next section:



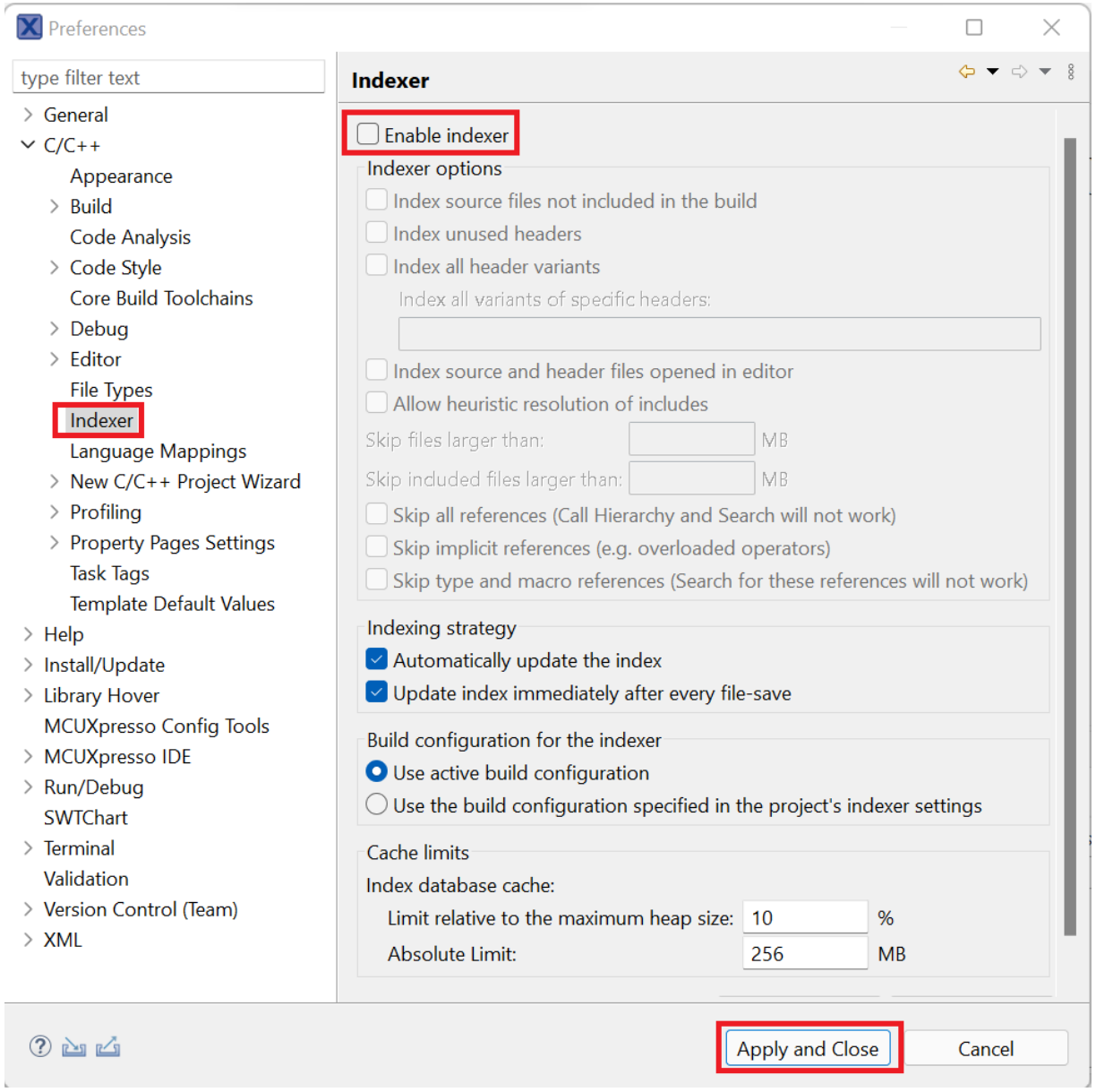
Run Demo

- The final step is to take the Label Image example and modify it to use the retrained model with the new image.

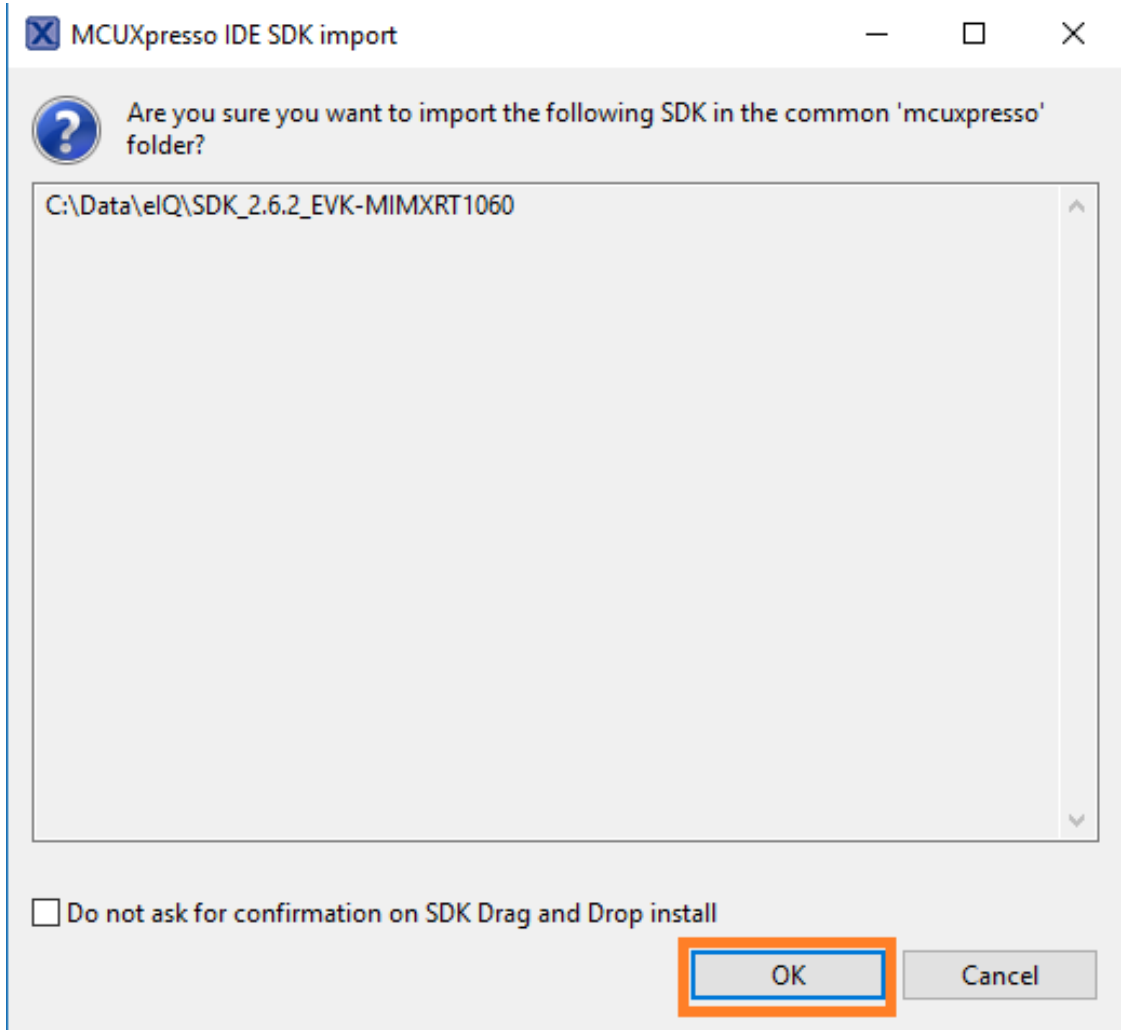
Copy and Create Files:

1. Open MCUXpresso IDE and select a workspace location in an empty directory.
2. Because of the large size of the model file, the indexer settings need to be changed by going to **Window->Preferences** from the menu bar. In the dialog box that comes up, go to the **C/C++->Indexer** category and uncheck **Enable Indexer**. Then click on **Apply and Close.**

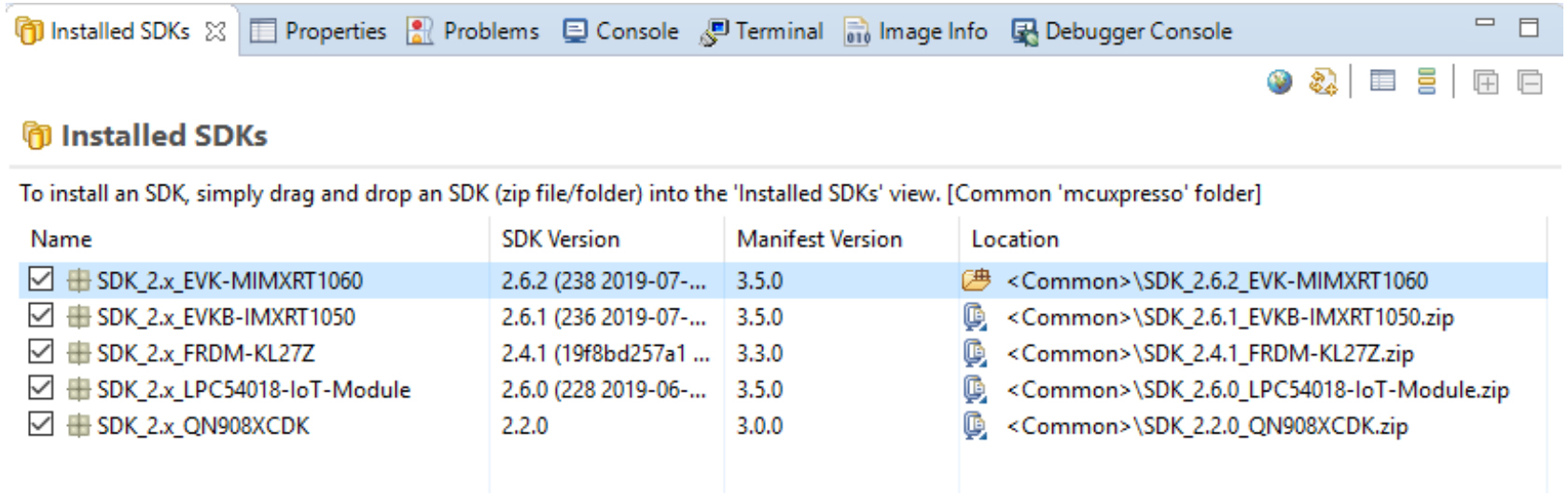
See image in next page for reference.


Preferences
type filter text
General
C/C++
Appearance
Build
Code Analysis
Code Style
Core Build Toolchains
Debug
Editor
File Types
Indexer
Language Mappings
New C/C++ Project Wizard
Profiling
Property Pages Settings
Task Tags
Template Default Values
Help
Install/Update
Library Hover
MCUXpresso Config Tools
MCUXpresso IDE
Run/Debug
SWTChart
Terminal
Validation
Version Control (Team)
XML
Indexer
Enable indexer
Indexer options
Index source files not included in the build
Index unused headers
Index all header variants
Index all variants of specific headers:
Index source and header files opened in editor
Allow heuristic resolution of includes
Skip files larger than:
MB
Skip included files larger than:
MB
Skip all references (Call Hierarchy and Search will not work)
Skip implicit references (e.g. overloaded operators)
Skip type and macro references (Search for these references will not work)
Indexing strategy
Automatically update the index
Update index immediately after every file-save
Build configuration for the indexer
Use active build configuration
Use the build configuration specified in the project's indexer settings
Cache limits
Index database cache:
Limit relative to the maximum heap size:
10
%
Absolute Limit:
256
MB
Apply and Close
Cancel

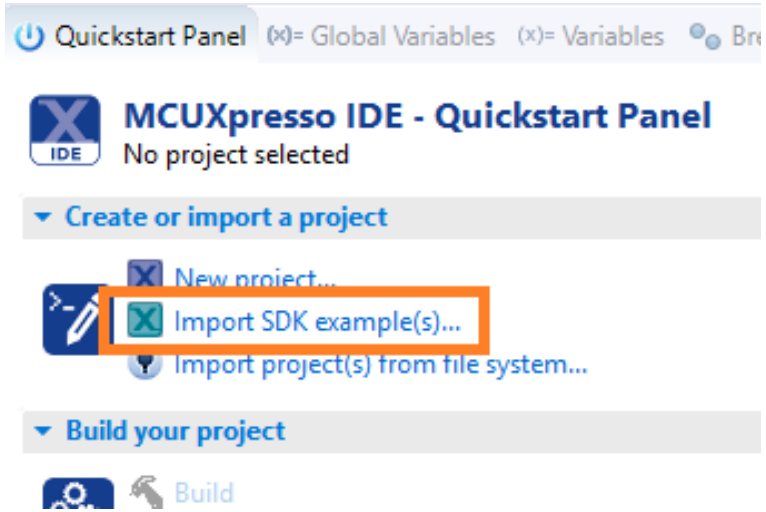
3. Drag-and-drop the unzipped SDK folder into the Installed SDKs window. It should have updated files as described in the first section. You will get the following pop-up, so hit OK.



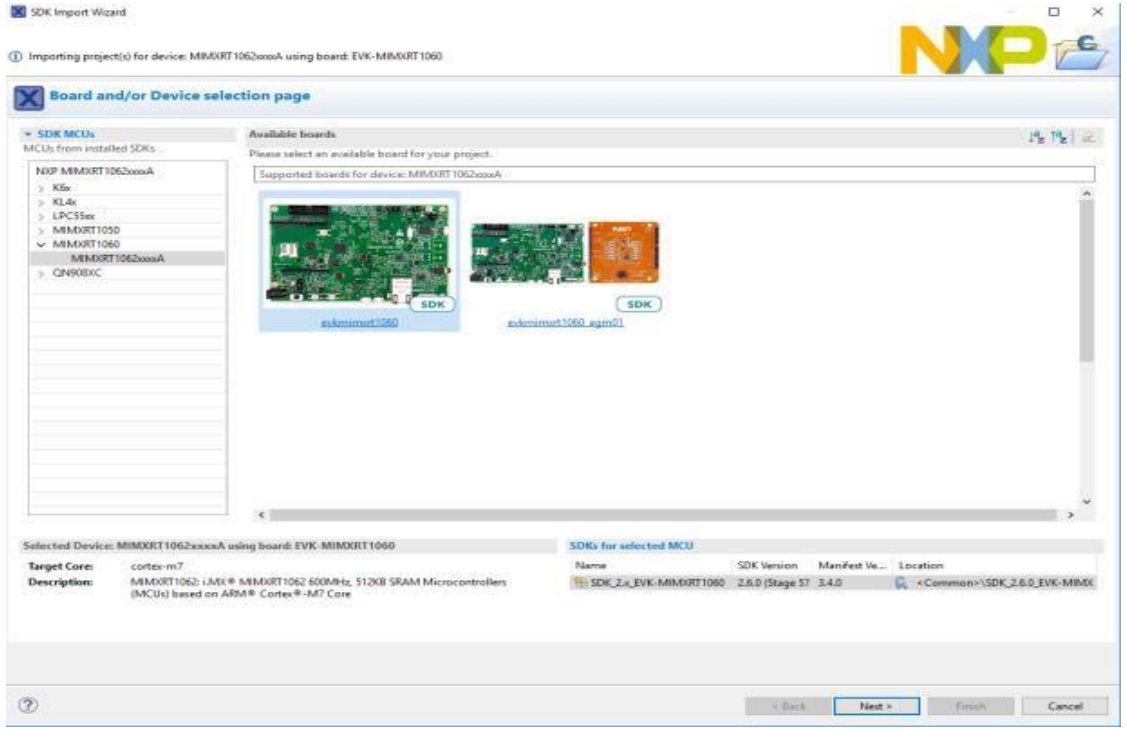
4. Once imported, the Installed SDK window will look something like this



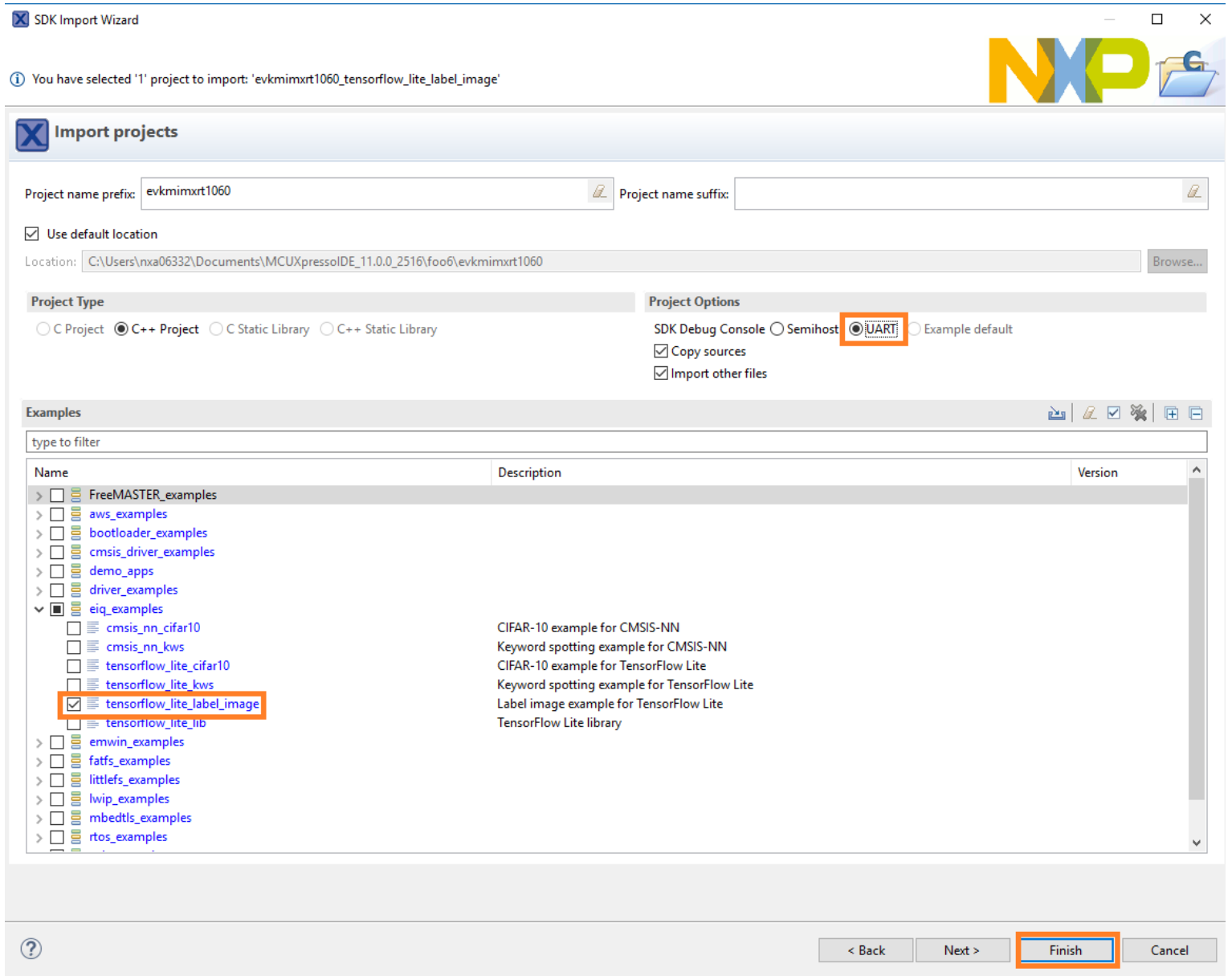
5. Next import the desired project. In the Quickstart Panel, select **Import SDK examples(s)…**



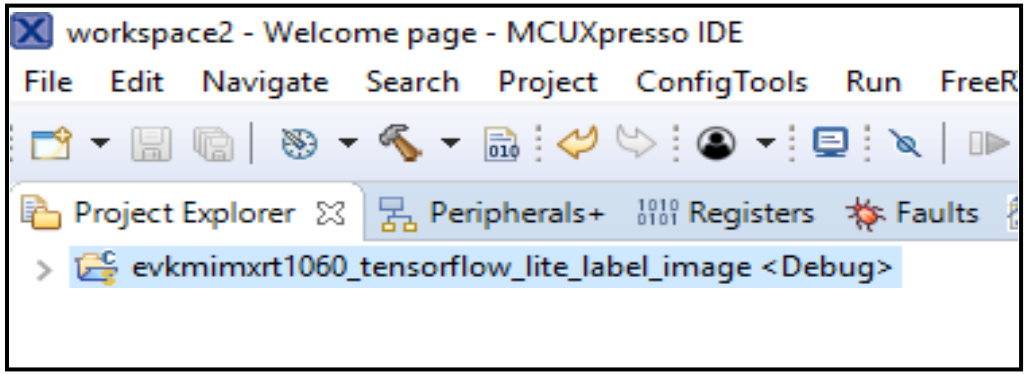
6. Select the evkmimxrt1060 board and click on Next



7. Then expand the **eiq_examples** folder and select **tensorflow_lite_label_image**. Also select **UART** for the SDK Debug Console. Then click on Finish to select that project.

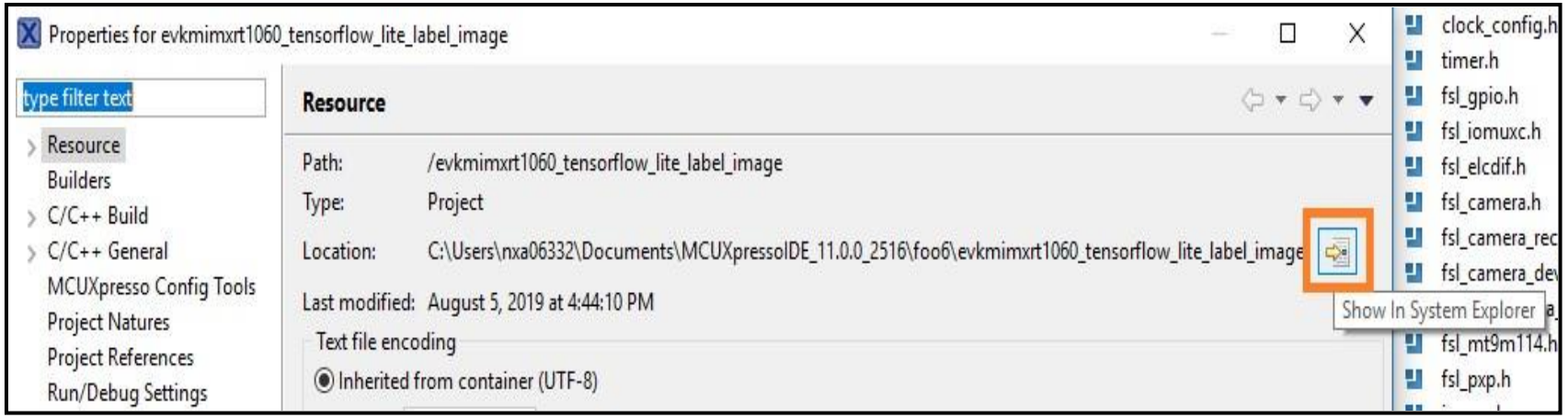

SDK Import Wizard
You have selected '1' project to import: 'evkmimxrt1060_tensorflow_lite_label_image'
Import projects
Project name prefix: evkmimxrt1060
Project name suffix:
Use default location
Location: C:\Users\nxa06332\Documents\MCUXpressoIDE_11.0.0_2516\foo6\evkmimxrt1060
Browse...
Project Type
C Project
C++ Project
C Static Library
C++ Static Library
Project Options
SDK Debug Console
Semihost
UART
Example default
Copy sources
Import other files
Examples
type to filter
Name
Description
Version
FreeMASTER_examples
aws_examples
bootloader_examples
cmsis_driver_examples
demo_apps
driver_examples
eiq_examples
cmsis_nn_cifar10
CIFAR-10 example for CMSIS-NN
cmsis_nn_kws
Keyword spotting example for CMSIS-NN
tensorflow_lite_cifar10
CIFAR-10 example for TensorFlow Lite
tensorflow_lite_kws
Keyword spotting example for TensorFlow Lite
tensorflow_lite_label_image
Label image example for TensorFlow Lite
tensorflow_lite_lib
TensorFlow Lite library
emwin_examples
fatfs_examples
littlefs_examples
lwip_examples
mbedtls_examples
rtos_examples
< Back
Next >
Finish
Cancel

8. It will look like the following when imported into the Project Explorer window:



9. Now we need to import the retrained model file that was generated in the last section into this project.

10. Find the directory location that this example was copied to by right clicking on the project name and select Properties. In the dialog box that comes up, click on the icon to open up that directory inside Windows explorer:



11. Go to the "**source**" directory inside the **evkmimxrt1060_tensorflow_lite_label_image** folder that you just opened. It should be something like: **C:\Users\nxp_training\Documents\MCUXpressoIDE_11.0.0_2516\workspace\evkmimxrt10 60_tensorflow_lite_label_image\source**

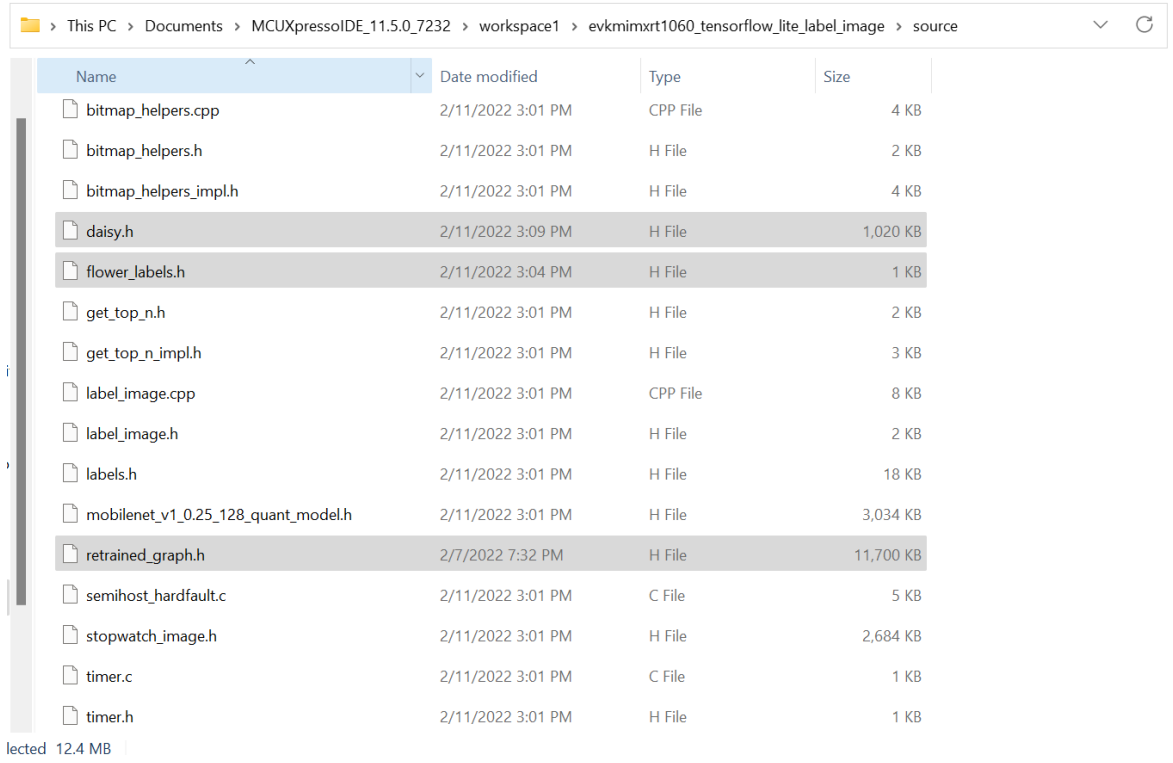
12. Inside that source directory, copy the retrained_graph.h file generated in the previous section.

13. If not using the camera, also copy the daisy.h file generated from previous section.

14. In that same directory, create a new header file named flower_labels.h and put in the following text, which will define the labels used to classify the flower images. This new file will be used to provide the classification labels instead of the labels.h file that was used by the default example. The file should look exactly like the following:

```
std::string labels_txt = R"(daisy
dandelion
roses
sunflowers
tulips
)";
```

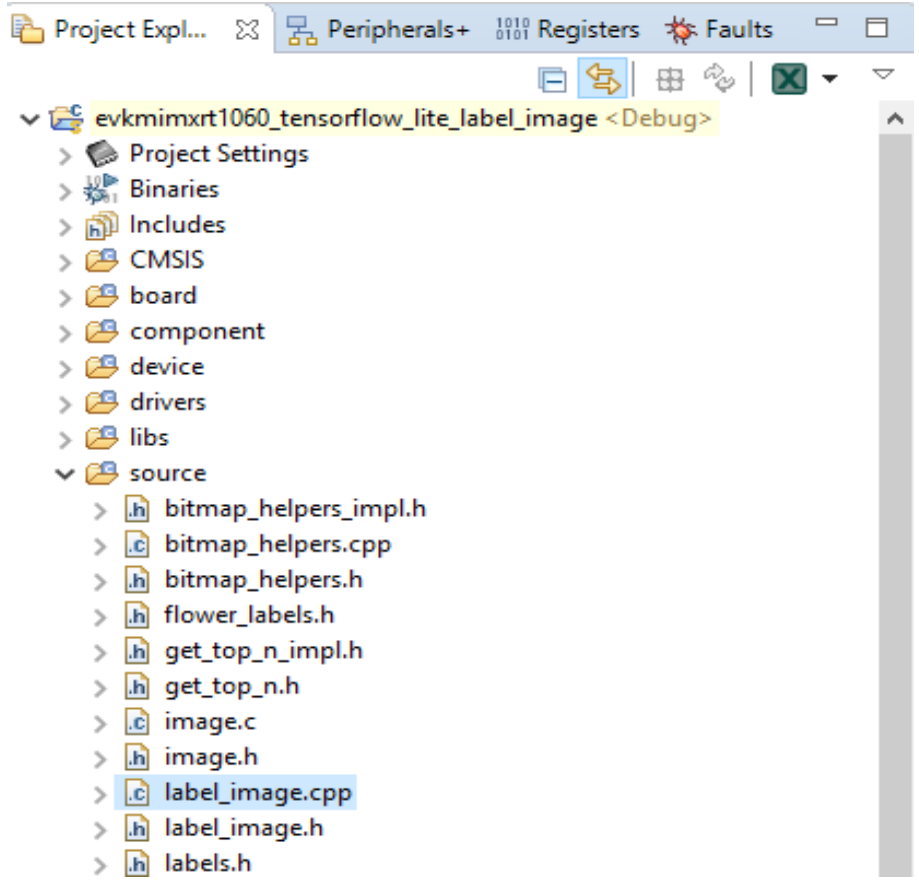
15. Directory should look like the beside image when finished:



Modify Source Code:

Now edit the source files to include these new files

16. Double click on the label_image.cpp file under the “source” folder in the Project View to open it.



17. Starting on line 34, comment out original #includes for the image, model, and label files. Then add new #includes to bring in the new image, model, and label files that were copied in. It should look like the following when finished:

label_image.cpp

```
26 #include "tensorflow/contrib/lite/model.h"
27 #include "tensorflow/contrib/lite/optional_debug_tools.h"
28 #include "tensorflow/contrib/lite/string_util.h"
29
30 #include "label_image.h"
31 #include "bitmap_helpers.h"
32 #include "get_top_n.h"
33
34 //#include "stopwatch_image.h"
35 //#include "mobilenet_v1_0.25_128_quant_model.h"
36 //#include "labels.h"
37
38 #include "daisy.h"
39 #include "retrained_graph.h"
40 #include "flower_labels.h"
41
42 #define LOG(x) std::cout
43
44 namespace tflite {
```

18. At around line 70, comment out the API call to load the default model, and replace it with the new model name and model length from the header file. It may be a slightly different name than the one listed below:

```
62      result->emplace_back();
63    }
64    return kTfLiteOk;
65  }
66
67  void RunInference(Settings* s) {
68    std::unique_ptr<tflite::FlatBufferModel> model;
69    std::unique_ptr<tflite::Interpreter> interpreter;
70    //model = tflite::FlatBufferModel::BuildFromBuffer(mobilenet_model, mobilenet_model_len);
71    model = tflite::FlatBufferModel::BuildFromBuffer(tf_files_retrained_graph_tflite, tf_files_retrained_graph_tflite_len);
72    if (!model) {
73      LOG(FATAL) << "Failed to load model\r\n";
74      return;
75    }
```

19. At around line 103, change the image height and width to 128 if they are not already, and then update the image buffer name and image length with the new image. The names may be a slightly different name than the one listed below and should match the array name and array length names in the daisy.h file:

label_image.cpp

```
92    int t_size = interpreter->tensors_size();
93    for (int i = 0; i < t_size; i++) {
94      if (interpreter->tensor(i)->name)
95        LOG(INFO) << i << ": " << interpreter->tensor(i)->name << ", "
96                  << interpreter->tensor(i)->bytes << ", "
97                  << interpreter->tensor(i)->type << ", "
98                  << interpreter->tensor(i)->params.scale << ", "
99                  << interpreter->tensor(i)->params.zero_point << "\r\n";
100   }
101 }
102
103 int image_width = 128;
104 int image_height = 128;
105 int image_channels = 3;
106 uint8_t* in = read_bmp(daisy_bmp, daisy_bmp_len, &image_width, &image_height,
107                        &image_channels, s);
108
109 int input = interpreter->inputs()[0];
110 if (s->verbose) LOG(INFO) << "input: " << input << "\r\n";
111
```

Run Example:

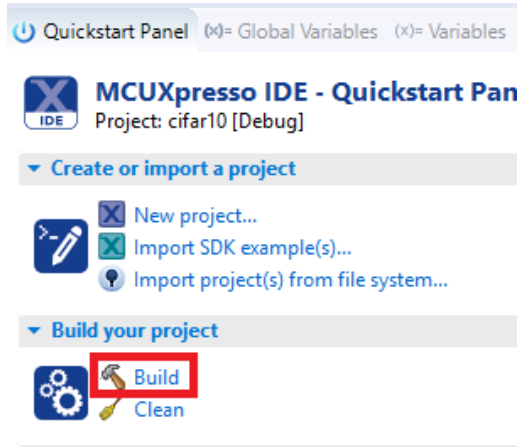
20. If using the i.MXRT1064 board, make the changes outlined in the following document: https://community.nxp.com/docs/DOC-344225. If using the i.MXRT1050 or i.MXRT1060 boards, this step is not needed.

21. On the i.MXRT1060 board, change SW7 to configure the board to boot from the flash. SW7 should be OFF-OFF-ON-OFF(as shown below). For i.MXRT1050 board SW7 should be OFF-ON-ON-OFF.

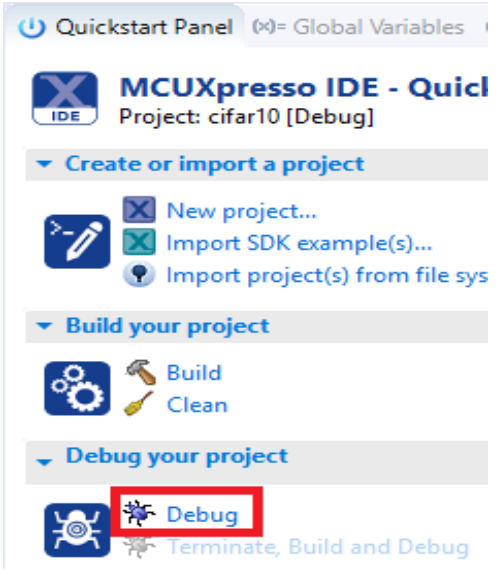
22. Plug the micro-B USB cable into the board at J41.

23. Open TeraTerm or other terminal program, and connect to the COM port that the board enumerated as. Use 115200 baud, 1 stop bit, no parity.

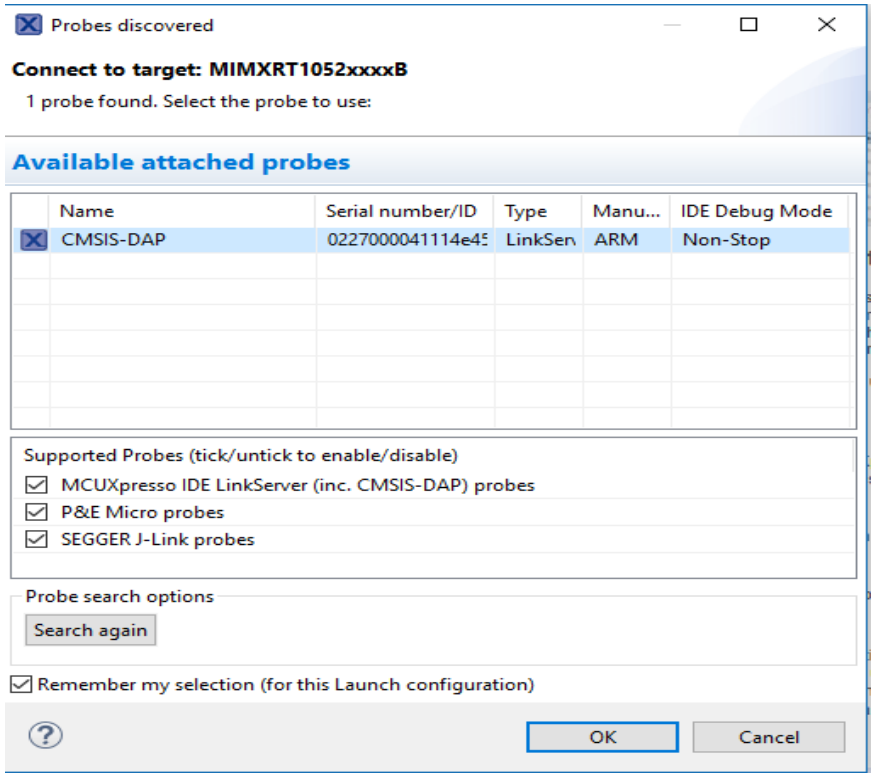
24. Build the project by clicking on “Build” in the Quickstart Panel.



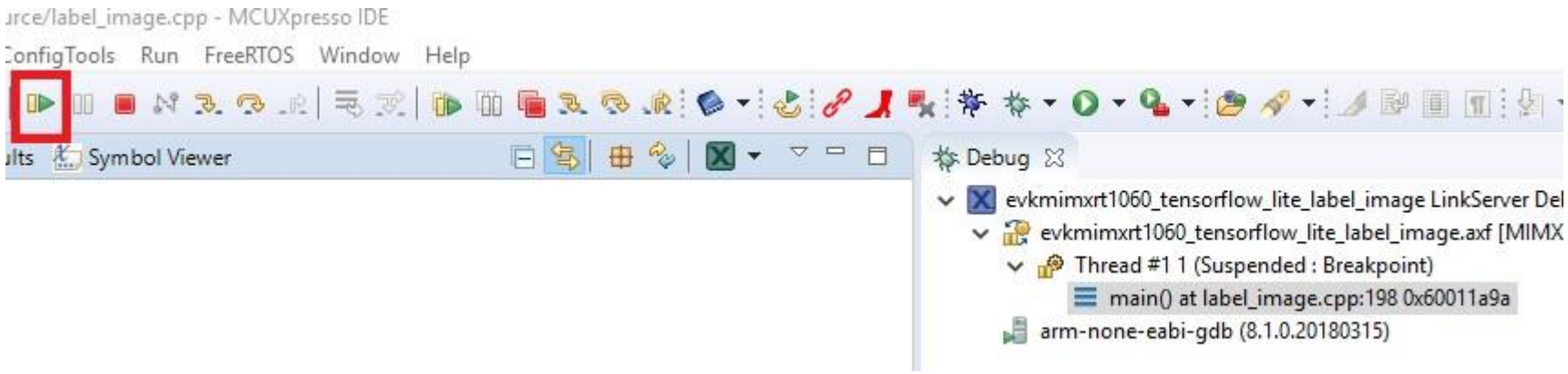
25. Debug the project by clicking on “Debug” in the Quickstart Panel.



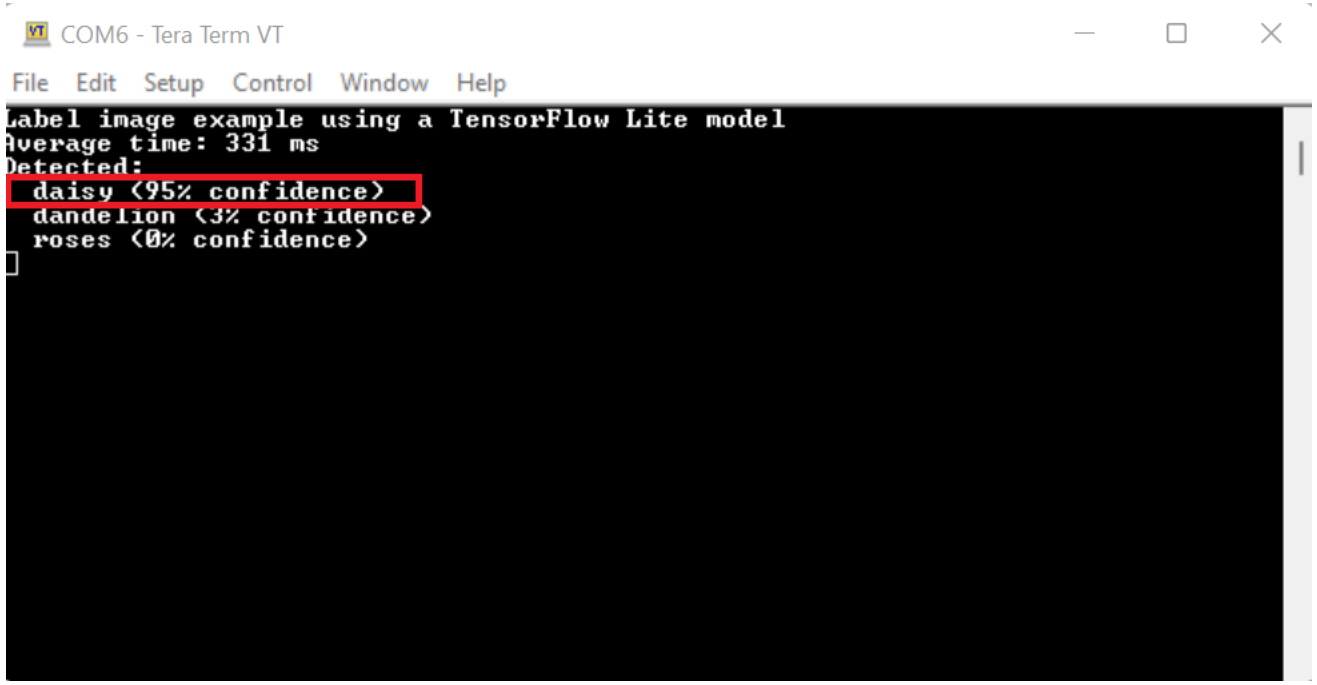
26. It will ask what interface to use. Select CMSIS-DAP.



27.The debugger will download the firmware and open up the debug view. Click on the Resume button to start running.



28. You should see the output on the console:



Conclusion:

- This lab demonstrated how to use the **tflite_convert** utility convert a TensorFlow model into a format that can be imported and ran on an embedded system using the eIQ software platform.

- The particular model was used to classify flower images. However, the model can also be trained on new types of images by retraining it. Just add a new directory name and example images of that classification to the flower_photos directory, and new images can be recognized by this model.

- Other types of TensorFlow models can be converted with this same process as well. By enabling machine learning in embedded systems, there’s a wide world of opportunity for new applications.