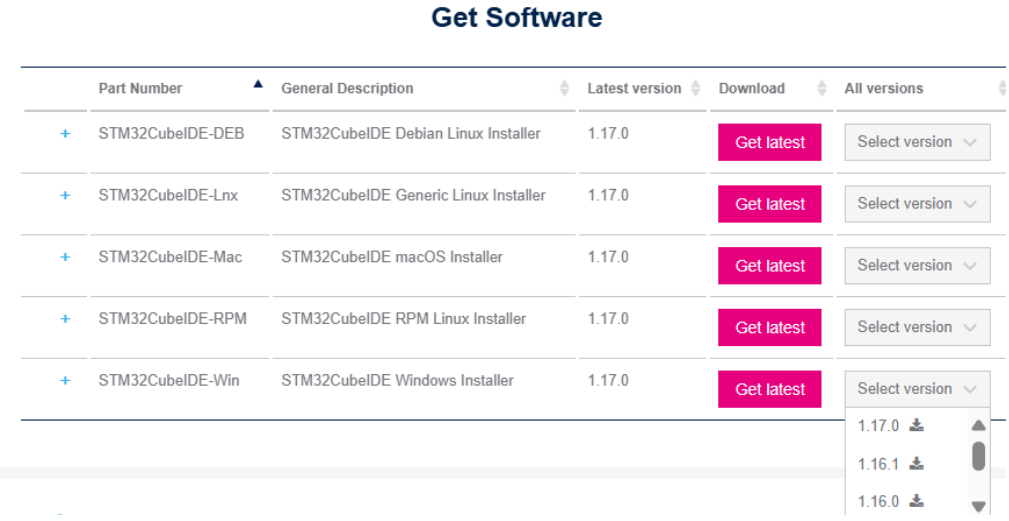
Deploying Tiny Deep Learning on ARM Cortex microcontroller

This user manual demonstrates how to deploy a tiny deep learning model on an ARM Cortex microcontroller, specifically the STM32F746G Discovery Board. The model used in this implementation is from the STM32AI Model Zoo, trained on a custom dataset consisting of "robot" and "non-robot". By the end of this manual, users will be able to train pre-trained models, test the model, perform JPEG decoding, deploy models on the STM32F746G Discovery Board, and run inference.

Pre - requisites

Software

1. **STM32CubeIDE v1.16.1.**
   STM32CubeIDE is an integrated development environment for STM32 microcontrollers.
   a. Use this link : [STM32CubeIDE - Integrated Development Environment for STM32 - STMicroelectronics]
   b. Locate STM32CubeIDE v1.16.1
   c. Choose the appropriate version for your operating system



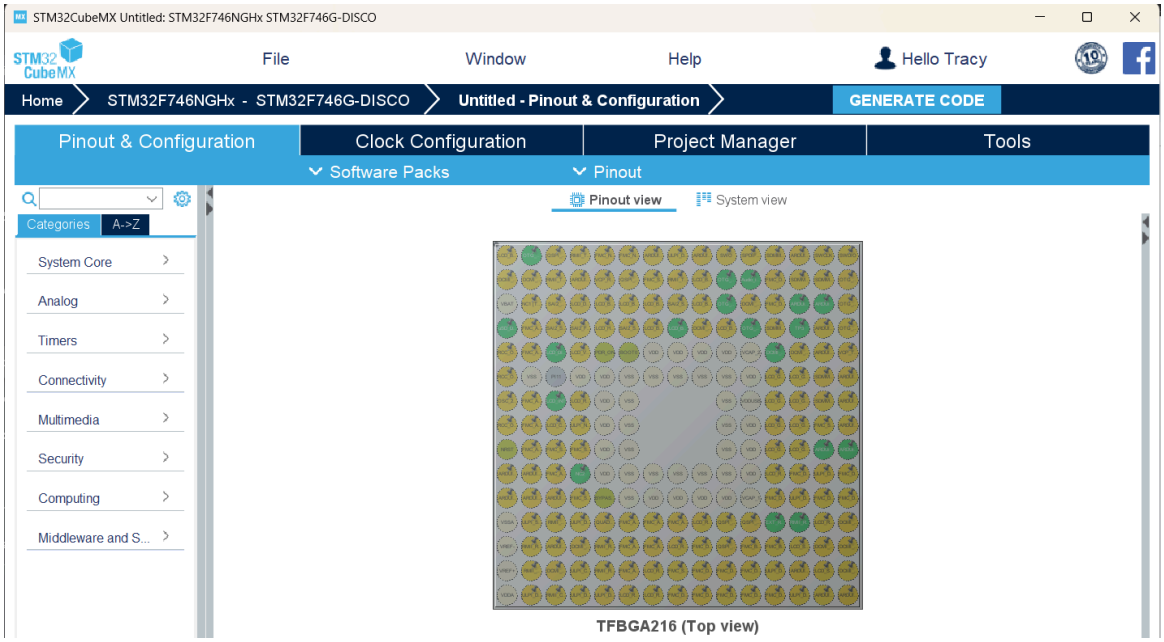
2. **STM32CubeMX**
   STM32CubeMX is a graphical configuration tool for STM32 projects.

a. Use this link : [STM32CubeMX - STM32Cube initialization code generator - STMicroelectronics]
b. Locate STM32CubeMX 6.10.0 or get the latest version
c. Choose the appropriate version for your operating system

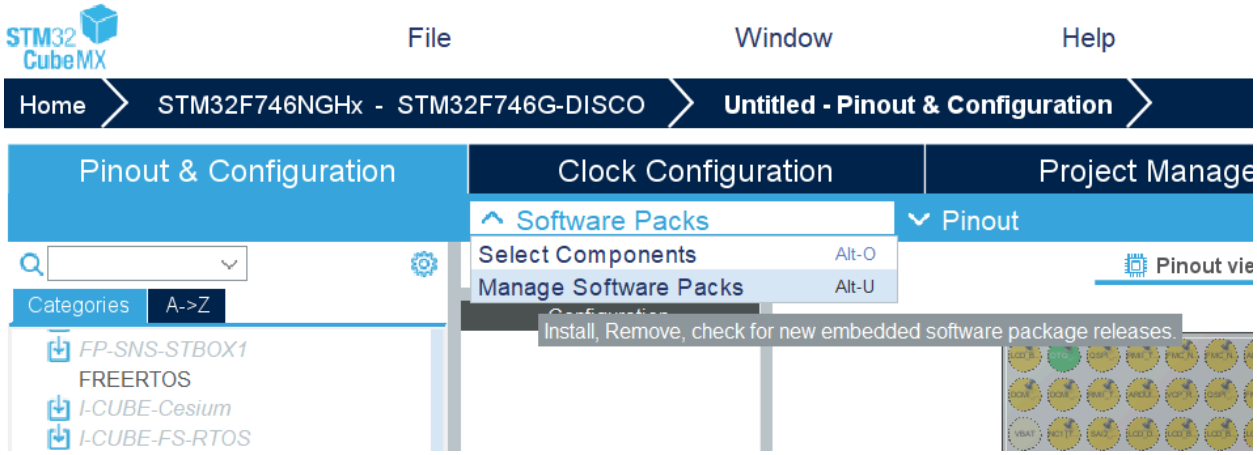
3. **X-Cube AI v7.0.0**

X-Cube AI is an AI extension for STM32CubeMX that allows converting pre-trained models into C code for STM32 microcontrollers.

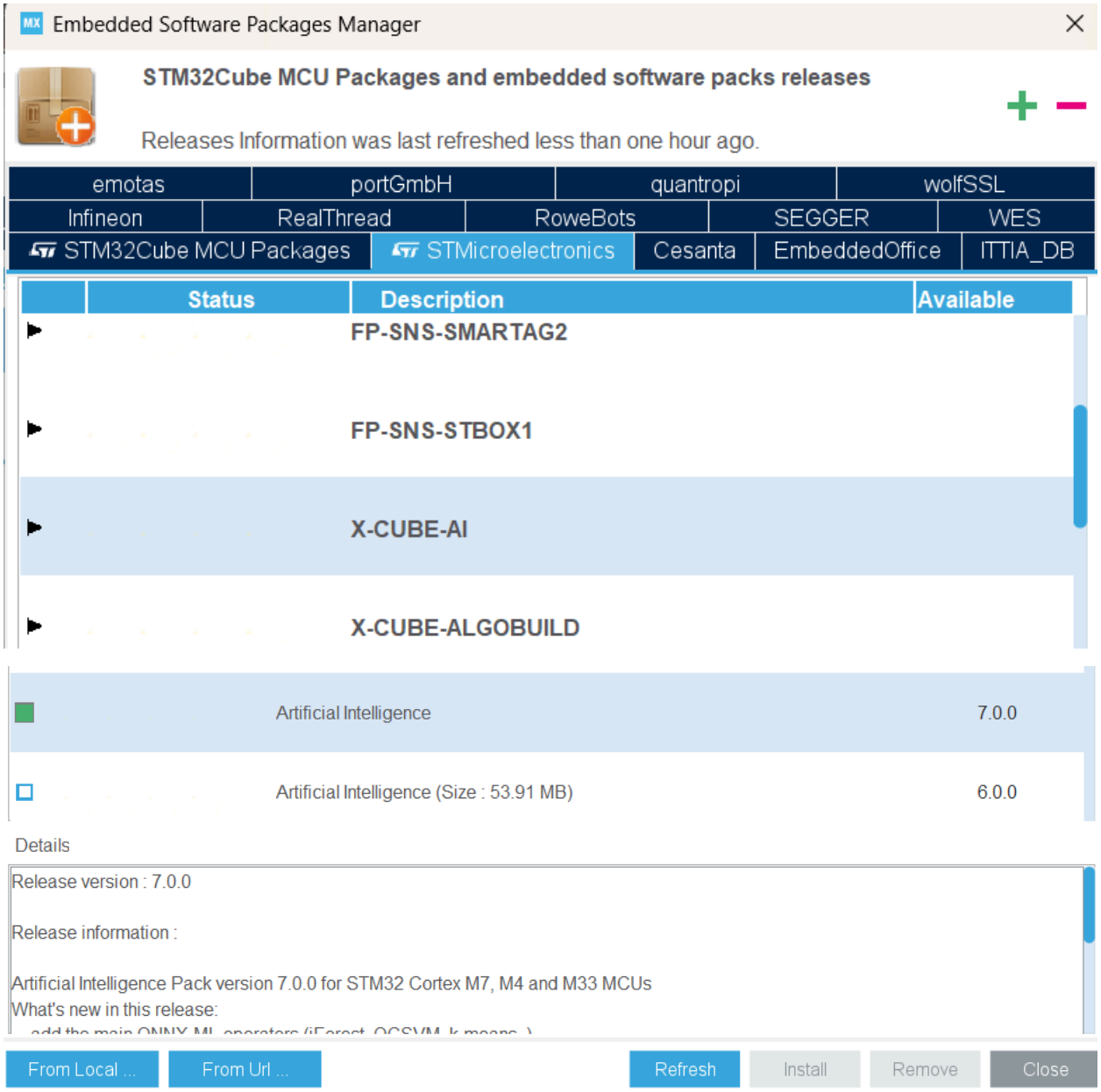
a. Ensure that STM32CubeMX or STM32CubeIDE is already installed
b. You can opt to download this later when you want to integrate AI
c. Create a STM32 project inside STM32CubeMX or STM32CubeIDE, specify STM32F746G Discovery board to be used
d. After creating project, you will be directed here



e. Click Software Packs > Manage Software Packs



f. Locate X-CUBE-AI > Select Artificial Intelligence version 7.0.0 > Install



4. **BSP Drivers**

   The Board Support Package (BSP) Drivers provide low-level drivers for the STM32F746G Discovery Board.

   a. Download it from this repository: STMicroelectronics/32f746gdiscovery-bsp at 64c20f758a907cd873e368502cf16e498934d0ec

Note: When using BSP Drivers only include the necessary drivers for your implementation and remove everything else to avoid errors

5. **HAL Drivers**

HAL (Hardware Abstraction Layer) Drivers simplify hardware control for STM32 microcontrollers.

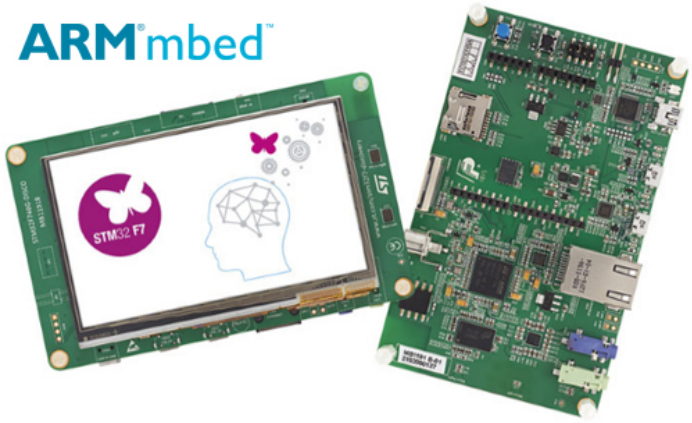
a. Download it from this repository: [stm32f7xx-hal-driver/Src at da6f1efa878f545dc45278fdae6dab85a680cb80 · STMicroelectronics/stm32f7xx-hal-driver]

Note: When creating a new STM32 project in STM32CubeMX or STM32CubeIDE, the required HAL Drivers are typically included automatically. However, when using BSP drivers, the necessary HAL drivers may not be added automatically. Ensure that all required HAL drivers are included manually to avoid errors.

Hardware

1. **STM32F746G DISCO Board**

   The microcontroller used for this implementation



2. **USB Type A to Mini-B Cable**

   For uploading STM32 programs in the STM32F746G DISCO

3. **SD Card**

   For storing test images and binary files of robot and non robot

Model Preparation

This section demonstrates how to train the mobilenet_v2_0.35_128 model on a custom dataset, test the model's predictions, and perform model quantization using the STM32AI Model Zoo.

Training the model

1. Download the stm32ai model zoo repository from this link : [STMicroelectronics/stm32ai-modelzoo: AI Model Zoo for STM32 devices]
2. Create a folder [model-training-stm32] in your local PC
3. Copy the contents of the repository into the folder [model-training-stm32]
4. Setup your virtual environment

```
PS C:\Users\USER\OneDrive\Documents\model-training-stm32> python --version
Python 3.9.13
PS C:\Users\USER\OneDrive\Documents\model-training-stm32> pyenv --version
pyenv 3.1.1
PS C:\Users\USER\OneDrive\Documents\model-training-stm32> python -m venv st_zoo
PS C:\Users\USER\OneDrive\Documents\model-training-stm32> ls


    Directory: C:\Users\USER\OneDrive\Documents\model-training-stm32


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        22/11/2024  11:12 am                audio_event_detection
d-----        22/11/2024  11:12 am                common
d-----        22/11/2024  11:12 am                hand_posture
d-----        22/11/2024  11:12 am                human_activity_recognition
d-----        22/11/2024  11:12 am                image_classification
d-----        22/11/2024  11:12 am                object_detection
d-----        22/11/2024  11:12 am                pose_estimation
d-----        22/11/2024  11:12 am                semantic_segmentation
d-----        22/11/2024  11:12 am                stm32ai_application_code
d-----        22/11/2024  12:14 pm                st_zoo
d-----        22/11/2024  11:12 am                tutorials
d-----        22/11/2024  11:12 am                X-LINUX-AI_application_code
-a----        16/09/2024   3:12 pm           3411 CODE_OF_CONDUCT.md
-a----        16/09/2024   3:12 pm           1178 CONTRIBUTING.md
-a----        16/09/2024   3:12 pm           1315 LICENSE.md
-a----        16/09/2024   3:12 pm          20485 manifest.json
-a----        16/09/2024   3:12 pm          20931 README.md
-a----        16/09/2024   3:12 pm            477 requirements.txt
-a----        16/09/2024   3:12 pm           1698 SECURITY.md


PS C:\Users\USER\OneDrive\Documents\model-training-stm32> .\st_zoo\Scripts\activate
(st_zoo) PS C:\Users\USER\OneDrive\Documents\model-training-stm32> python --version
Python 3.9.13
(st_zoo) PS C:\Users\USER\OneDrive\Documents\model-training-stm32> pip install -r requirements.txt
```

5. Prepare your dataset, nonrobot and robot folders inside the robotdatasets then store it inside dataset folder
6. To train the image classification model, refer to this manual: [stm32ai-modelzoo/image_classification/src/training at main · STMicroelectronics/stm32ai-modelzoo]
7. For our implementation, specify these parameters in the configuration file
    a. name: robot_presence_detection
    b. class_names: [robot, nonrobot]

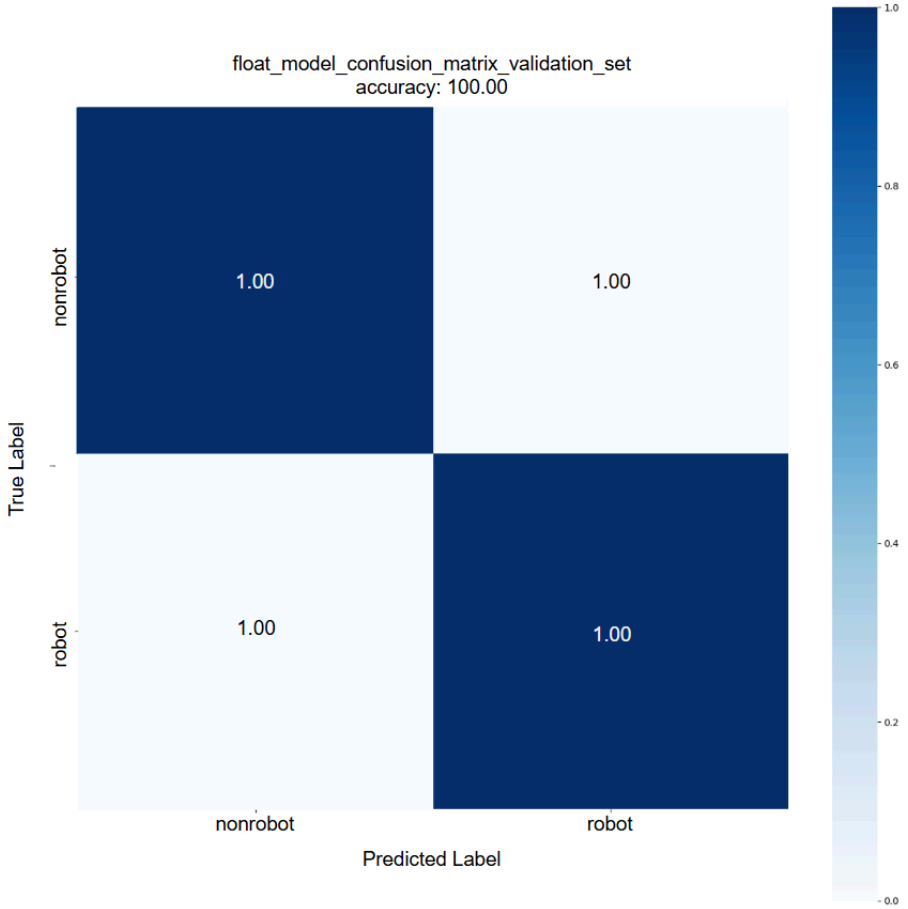
c. training_path: ../datasets/robotdatasets
d. pretrained_weights: imagenet
e. input_shape: (128, 128, 3)

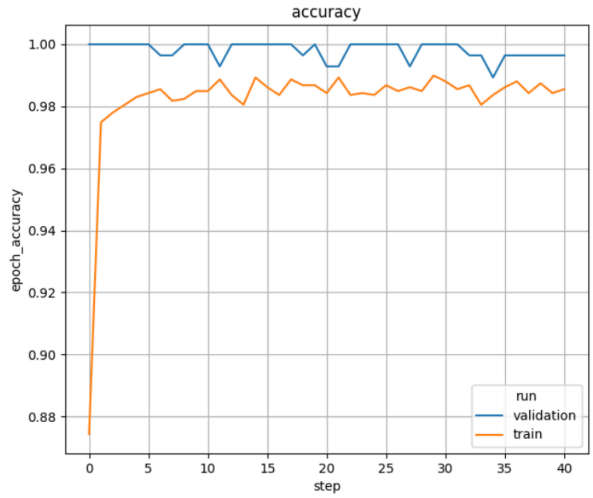
8. Run the training script

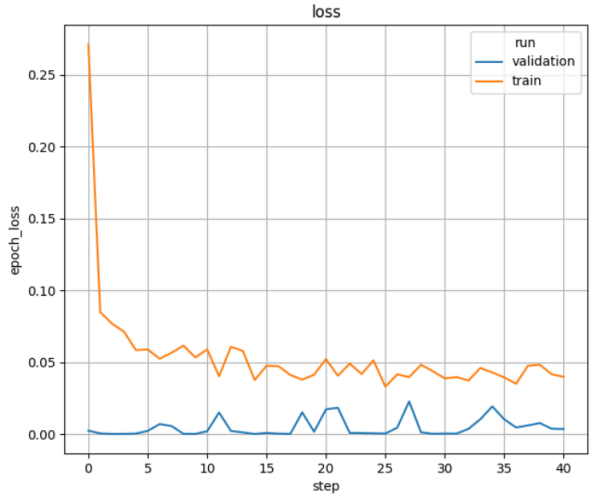
```
(st_zoo) PS C:\Users\USER\OneDrive\Documents\model-training-stm32> cd C:\Users\USER\OneDrive\Documents\model-training-stm32\image_classification\src
(st_zoo) PS C:\Users\USER\OneDrive\Documents\model-training-stm32\image_classification\src> python stm32ai_main.py --config-path ./config_file_examples/ --c
onfig-name training_config.yaml
[INFO] : Setting upper limit of usable GPU memory to 3GBytes.
[INFO] : Running `training` operation mode
2024/11/22 12:22:11 INFO mlflow.tracking.fluent: Experiment with name 'tf_robot_presence_detection' does not exist. Creating a new experiment.
[INFO] : The random seed for this simulation is 127
Dataset stats:
  classes: 2
  training set size: 1592
  validation set size: 281
  no test set
[INFO] : Using `mobilenet` model
[INFO] : Initialized model with 'imagenet' pretrained weights
[INFO] : Set dropout rate to 0.4
_________________________________________________________________________________________________
  Model: mobilenet_v2_alpha_0.35
_________________________________________________________________________________________________
  Layer index  Trainable    Name                      Type                      Params#  Output shape
-------------  -----------  ------------------------  ----------------------  ---------  -------------------
            0  True         input_1                   InputLayer                      0  (None, 128, 128, 3)
            1  True         conv2d                    Conv2D                        432  (None, 64, 64, 16)
            2  True         batch_normalization       BatchNormalization             64  (None, 64, 64, 16)
            3  True         re_lu                     ReLU                            0  (None, 64, 64, 16)
            4  True         depthwise_conv2d          DepthwiseConv2D               144  (None, 64, 64, 16)
            5  True         batch_normalization_1     BatchNormalization             64  (None, 64, 64, 16)
            6  True         re_lu_1                   ReLU                            0  (None, 64, 64, 16)
            7  True         conv2d_1                  Conv2D                        128  (None, 64, 64, 8)
            8  True         batch_normalization_2     BatchNormalization             32  (None, 64, 64, 8)
            9  True         conv2d_2                  Conv2D                        384  (None, 64, 64, 48)
           10  True         batch_normalization_3     BatchNormalization            192  (None, 64, 64, 48)
           11  True         re_lu_2                   ReLU                            0  (None, 64, 64, 48)
           12  True         zero_padding2d            ZeroPadding2D                   0  (None, 65, 65, 48)
           13  True         depthwise_conv2d_1        DepthwiseConv2D               432  (None, 32, 32, 48)
           14  True         batch_normalization_4     BatchNormalization            192  (None, 32, 32, 48)
```

```
_________________________________________________________________________________________________
Starting training...
Epoch 1/200
13/13 [==============================] - 44s 2s/step - loss: 0.2712 - accuracy: 0.8744 - val_loss: 0.0024 - val_accuracy: 1.0000 - lr: 0.0010
Epoch 2/200
13/13 [==============================] - 30s 2s/step - loss: 0.0848 - accuracy: 0.9749 - val_loss: 4.0759e-04 - val_accuracy: 1.0000 - lr: 0.0010
Epoch 3/200
13/13 [==============================] - 30s 2s/step - loss: 0.0767 - accuracy: 0.9780 - val_loss: 3.5208e-05 - val_accuracy: 1.0000 - lr: 0.0010
Epoch 4/200
13/13 [==============================] - 31s 2s/step - loss: 0.0712 - accuracy: 0.9805 - val_loss: 1.0757e-04 - val_accuracy: 1.0000 - lr: 0.0010
Epoch 5/200
13/13 [==============================] - 31s 2s/step - loss: 0.0584 - accuracy: 0.9830 - val_loss: 2.8695e-04 - val_accuracy: 1.0000 - lr: 0.0010
Epoch 6/200
13/13 [==============================] - 30s 2s/step - loss: 0.0588 - accuracy: 0.9843 - val_loss: 0.0022 - val_accuracy: 1.0000 - lr: 0.0010
Epoch 7/200
13/13 [==============================] - 30s 2s/step - loss: 0.0523 - accuracy: 0.9856 - val_loss: 0.0069 - val_accuracy: 0.9964 - lr: 0.0010
Epoch 8/200
13/13 [==============================] - 31s 2s/step - loss: 0.0565 - accuracy: 0.9818 - val_loss: 0.0056 - val_accuracy: 0.9964 - lr: 0.0010
Epoch 9/200
13/13 [==============================] - 32s 2s/step - loss: 0.0615 - accuracy: 0.9824 - val_loss: 1.3707e-04 - val_accuracy: 1.0000 - lr: 0.0010
Epoch 10/200
13/13 [==============================] - 32s 2s/step - loss: 0.0533 - accuracy: 0.9849 - val_loss: 5.9751e-05 - val_accuracy: 1.0000 - lr: 0.0010
Epoch 11/200
13/13 [==============================] - 30s 2s/step - loss: 0.0589 - accuracy: 0.9849 - val_loss: 0.0019 - val_accuracy: 1.0000 - lr: 0.0010
Epoch 12/200
13/13 [==============================] - 30s 2s/step - loss: 0.0402 - accuracy: 0.9887 - val_loss: 0.0149 - val_accuracy: 0.9929 - lr: 0.0010
Epoch 13/200
13/13 [==============================] - 30s 2s/step - loss: 0.0607 - accuracy: 0.9837 - val_loss: 0.0021 - val_accuracy: 1.0000 - lr: 0.0010
Epoch 14/200
13/13 [==============================] - 29s 2s/step - loss: 0.0578 - accuracy: 0.9805 - val_loss: 0.0011 - val_accuracy: 1.0000 - lr: 0.0010
Epoch 15/200
13/13 [==============================] - 29s 2s/step - loss: 0.0376 - accuracy: 0.9893 - val_loss: 3.6510e-05 - val_accuracy: 1.0000 - lr: 0.0010
Epoch 16/200
13/13 [==============================] - 30s 2s/step - loss: 0.0475 - accuracy: 0.9862 - val_loss: 7.3276e-04 - val_accuracy: 1.0000 - lr: 0.0010
Epoch 17/200
13/13 [==============================] - 30s 2s/step - loss: 0.0472 - accuracy: 0.9837 - val_loss: 2.5688e-04 - val_accuracy: 1.0000 - lr: 0.0010
Epoch 18/200
13/13 [==============================] - 29s 2s/step - loss: 0.0411 - accuracy: 0.9887 - val_loss: 8.4811e-05 - val_accuracy: 1.0000 - lr: 0.0010
Epoch 19/200
13/13 [==============================] - 30s 2s/step - loss: 0.0378 - accuracy: 0.9868 - val_loss: 0.0151 - val_accuracy: 0.9964 - lr: 0.0010
```

9. Model is then saved into the *C:\Users\USER\OneDrive\Documents\model-training-stm32\image_classification\src\experiments_outputs\2024_11_22_12_22_07\saved_models*
10. To verify training results visualizations, go to /experiments_outputs/mlruns inside the src folder of image_classification.


float_model_confusion_matrix_validation_set
accuracy: 100.00
1.00
1.00
1.00
1.00
nonrobot
robot
True Label
nonrobot
robot
Predicted Label
1.0
0.8
0.6
0.4
0.2
0.0


accuracy
epoch_accuracy
step
run
validation
train


loss
epoch_loss
step
run
validation
train

Testing the model

1. To test the model on unseen data, refer on this manual : [stm32ai-modelzoo/image_classification/src/prediction at main · STMicroelectronics/stm32ai-modelzoo]
2. For our implementation, specify these parameters in the configuration file
    a. Specify best model path
    b. test_files_path: "C:/Users/USER/OneDrive/Documents/model-training-stm32/image_classification/datasets/test/robot"
3. Since the implementation is a binary classification with sigmoid activation on the final layer, in line 96 update the predict.py to interpret the prediction score correctly.
4. Run the prediction script:

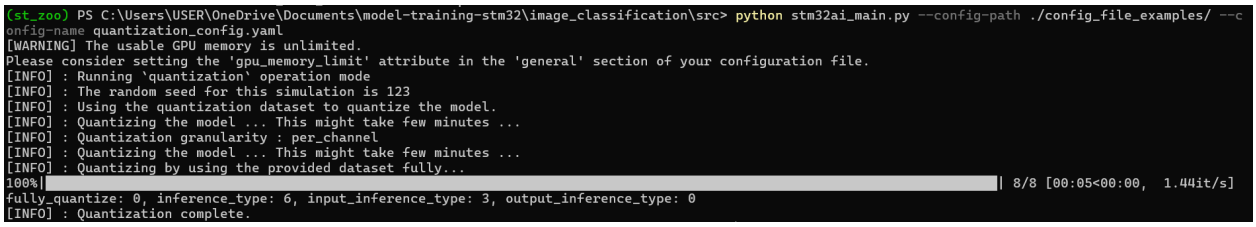
```
(st_zoo) PS C:\Users\USER\OneDrive\Documents\model-training-stm32\image_classification\src> python stm32ai_main.py --config-path ./config_file_examples/ --config-name prediction_config.yaml
[WARNING] The usable GPU memory is unlimited.
Please consider setting the 'gpu_memory_limit' attribute in the 'general' section of your configuration file.
[INFO] : Running 'prediction' operation mode
[INFO] : The random seed for this simulation is 123
[INFO] : Making predictions using:
  model: C:/Users/USER/OneDrive/Documents/model-training-stm32/image_classification/src/experiments_outputs/2024_11_22_12_22_07/saved_models/best_model.h5
  images directory: C:/Users/USER/OneDrive/Documents/model-training-stm32/image_classification/datasets/test/robot
Prediction    Score    Image file
------------  -------  -------------
robot         100      robot1000.png
robot         100      robot1012.png
robot         100      robot1017.png
robot         100      robot1018.png
robot         100      robot1034.png
robot         100      robot1035.png
robot         100      robot1049.png
robot         100      robot105.png
robot         100      robot1052.png
robot         100      robot107.png
robot         100      robot1072.png
robot         100      robot1073.png
robot         100      robot1081.png
robot         100      robot1083.png
robot         100      robot1087.png
robot         100      robot110.png
robot         100      robot1114.png
robot         100      robot1124.png
robot         100      robot1132.png
robot         100      robot1139.png
robot         100      robot1144.png
robot         100      robot1147.png
robot         100      robot1149.png
robot         100      robot116.png
robot         100      robot1169.png
robot         100      robot1181.png
robot         100      robot1195.png
robot         100      robot1196.png
```

```
(st_zoo) PS C:\Users\USER\OneDrive\Documents\model-training-stm32\image_classification\src> python stm32ai_main.py --config-path ./config_file_examples/ --config-name prediction_config.yaml
[WARNING] The usable GPU memory is unlimited.
Please consider setting the 'gpu_memory_limit' attribute in the 'general' section of your configuration file.
[INFO] : Running 'prediction' operation mode
[INFO] : The random seed for this simulation is 123
[INFO] : Making predictions using:
  model: C:/Users/USER/OneDrive/Documents/model-training-stm32/image_classification/src/experiments_outputs/2024_11_22_12_22_07/saved_models/best_model.h5
  images directory: C:/Users/USER/OneDrive/Documents/model-training-stm32/image_classification/datasets/test/nonrobot
Prediction    Score    Image file
------------  -------  ----------------
nonrobot      99.8     nonrobot101.png
nonrobot      100      nonrobot1014.png
nonrobot      100      nonrobot1017.png
nonrobot      100      nonrobot1025.png
nonrobot      100      nonrobot1028.png
nonrobot      100      nonrobot103.png
nonrobot      99.9     nonrobot1035.png
nonrobot      100      nonrobot1041.png
nonrobot      100      nonrobot1042.png
nonrobot      100      nonrobot1046.png
nonrobot      100      nonrobot1049.png
nonrobot      100      nonrobot1051.png
nonrobot      100      nonrobot1057.png
nonrobot      100      nonrobot1058.png
nonrobot      99.8     nonrobot1063.png
nonrobot      100      nonrobot1069.png
nonrobot      100      nonrobot1070.png
nonrobot      100      nonrobot1072.png
nonrobot      100      nonrobot1081.png
nonrobot      100      nonrobot1095.png
nonrobot      100      nonrobot1100.png
nonrobot      100      nonrobot1101.png
nonrobot      100      nonrobot1102.png
nonrobot      100      nonrobot1110.png
nonrobot      100      nonrobot1111.png
nonrobot      100      nonrobot1118.png
nonrobot      100      nonrobot1123.png
nonrobot      100      nonrobot1124.png
nonrobot      100      nonrobot1127.png
nonrobot      96.3     nonrobot1128.png
```

Quantize the model

Post-Training Quantization (PTQ) is applied using TensorFlow Lite's TFLite Converter to optimize the trained model.

1. To reduce the size of your best model, refer on this manual : [stm32ai-modelzoo/image_classification/src/prediction at main · STMicroelectronics/stm32ai-modelzoo]
2. For our implementation, specify these parameters in the configuration file
    a. Specify best model path
    b. Quantization_path: "C:/Users/USER/OneDrive/Documents/model-training-stm32/image_classification/datasets/robot_photos"
    c. quantizer: TFlite_converter
    d. quantization_type: PTQ
    e. quantization_input_type: uint8
    f. quantization_output_type: float
3. Run the quantization script:



4. The quantized model will be stored in the experiments_outputs directory, e.g C:\Users\USER\OneDrive\Documents\model-training-stm32\image_classification\src\experiments_outputs\2024_11_22_15_24_55\quantized_models
5. Check the quantized model size, it should be less than ~ 800 KB

Testing the quantized model

1. Specify these parameters in the configuration file
    a. Specify quantized model path
    b. test_files_path: "C:/Users/USER/OneDrive/Documents/model-training-stm32/image_classification/datasets/test/robot"
2. Run the prediction script:

```
(st_zoo) PS C:\Users\USER\OneDrive\Documents\model-training-stm32\image_classification\src> python stm32ai_main.py --config-path ./config_file_examples/ --c
onfig-name prediction_config.yaml
[WARNING] The usable GPU memory is unlimited.
Please consider setting the 'gpu_memory_limit' attribute in the 'general' section of your configuration file.
[INFO] : Running `prediction` operation mode
[INFO] : The random seed for this simulation is 123
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
[INFO] : Making predictions using:
  model: C:/Users/USER/OneDrive/Documents/model-training-stm32/image_classification/src/experiments_outputs/2024_11_22_15_24_55/quantized_models/quantized_m
odel.tflite
  images directory: C:/Users/USER/OneDrive/Documents/model-training-stm32/image_classification/datasets/test/robot
Prediction    Score    Image file
------------  -------  -------------
robot         99.6     robot1000.png
robot         99.6     robot1012.png
robot         99.6     robot1017.png
robot         99.6     robot1018.png
robot         99.6     robot1034.png
robot         99.6     robot1035.png
robot         99.6     robot1049.png
robot         99.6     robot105.png
robot         99.6     robot1052.png
robot         99.6     robot107.png
robot         99.6     robot1072.png
robot         99.6     robot1073.png
robot         99.6     robot1081.png
robot         99.6     robot1083.png
robot         99.6     robot1087.png
robot         99.6     robot110.png
robot         99.6     robot1114.png
robot         99.6     robot1124.png
robot         99.6     robot1132.png
robot         99.6     robot1139.png
robot         99.6     robot1144.png
robot         99.6     robot1147.png
robot         99.6     robot1149.png
robot         99.6     robot116.png
robot         99.6     robot1169.png
robot         99.6     robot1181.png
robot         99.6     robot1195.png
robot         99.6     robot1196.png
```

```
(st_zoo) PS C:\Users\USER\OneDrive\Documents\model-training-stm32\image_classification\src> python stm32ai_main.py --config-path ./config_file_examples/ --c
onfig-name prediction_config.yaml
[WARNING] The usable GPU memory is unlimited.
Please consider setting the 'gpu_memory_limit' attribute in the 'general' section of your configuration file.
[INFO] : Running `prediction` operation mode
[INFO] : The random seed for this simulation is 123
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
[INFO] : Making predictions using:
  model: C:/Users/USER/OneDrive/Documents/model-training-stm32/image_classification/src/experiments_outputs/2024_11_22_15_24_55/quantized_models/quantized_m
odel.tflite
  images directory: C:/Users/USER/OneDrive/Documents/model-training-stm32/image_classification/datasets/test/nonrobot
Prediction    Score    Image file
------------  -------  ----------------
nonrobot      99.6     nonrobot101.png
nonrobot      100      nonrobot1014.png
nonrobot      100      nonrobot1017.png
nonrobot      100      nonrobot1025.png
nonrobot      100      nonrobot1028.png
nonrobot      100      nonrobot103.png
nonrobot      100      nonrobot1035.png
nonrobot      100      nonrobot1041.png
nonrobot      100      nonrobot1042.png
nonrobot      100      nonrobot1046.png
nonrobot      100      nonrobot1049.png
nonrobot      100      nonrobot1051.png
nonrobot      100      nonrobot1057.png
nonrobot      100      nonrobot1058.png
nonrobot      100      nonrobot1063.png
nonrobot      100      nonrobot1069.png
nonrobot      100      nonrobot1070.png
nonrobot      100      nonrobot1072.png
nonrobot      100      nonrobot1081.png
nonrobot      100      nonrobot1095.png
nonrobot      100      nonrobot1100.png
nonrobot      100      nonrobot1101.png
nonrobot      100      nonrobot1102.png
nonrobot      100      nonrobot1110.png
nonrobot      100      nonrobot1111.png
nonrobot      100      nonrobot1118.png
nonrobot      100      nonrobot1123.png
```


robot1


robot2


robot3


robot5


testimage2


testrobot (1)


testrobot (5)


testrobot (6)


testrobot (7)


testrobot (12)

```
(st_zoo) PS C:\Users\USER\OneDrive\Documents\model-training-stm32\image_classification\src> python stm32ai_main.py --config-path ./config_file_e
xamples/ --config-name prediction_config.yaml
[WARNING] The usable GPU memory is unlimited.
Please consider setting the 'gpu_memory_limit' attribute in the 'general' section of your configuration file.
[INFO] : Running `prediction` operation mode
[INFO] : The random seed for this simulation is 123
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
[INFO] : Making predictions using:
  model: C:/Users/USER/OneDrive/Documents/model-training-stm32/image_classification/src/experiments_outputs/2024_11_22_15_24_55/quantized_models
/quantized_model.tflite
  images directory: C:/Users/USER/OneDrive/Documents/model-training-stm32/image_classification/datasets/random
Prediction    Score    Image file
------------  -------  ------------------
robot         99.6     robot1.jpg
robot         99.6     robot2.jpg
robot         99.6     robot3.jpg
robot         99.6     robot5.jpg
robot         99.6     testimage2.jpg
nonrobot      100      testrobot (1).jpg
nonrobot      100      testrobot (12).jpg
nonrobot      100      testrobot (5).jpg
nonrobot      100      testrobot (6).jpg
nonrobot      100      testrobot (7).jpg
[INFO] : Prediction complete.
```

Data Preparation

Preprocessing

Refer to this notebook: [thesis_preprocessing_final.ipynb - Colab]

1. Upload your nonrobot and robot folder inside the test folder
2. The model expects an input shape of 128 x 128 x 3 and uint8 data type

```
[ ]  def preprocess_image_for_stm32(img_path, target_size=(128, 128)):
         img = Image.open(img_path).convert('RGB')   # Load image and convert to RGB
         img = img.resize(target_size)               # Resize to 128x128
         img_array = np.array(img, dtype=np.uint8)   # Convert to uint8 array
         img_array = img_array.flatten()             # Flatten to 1D array
         return img_array
```

3. Save the model as binary file

input directory then process it

+ Code + Text

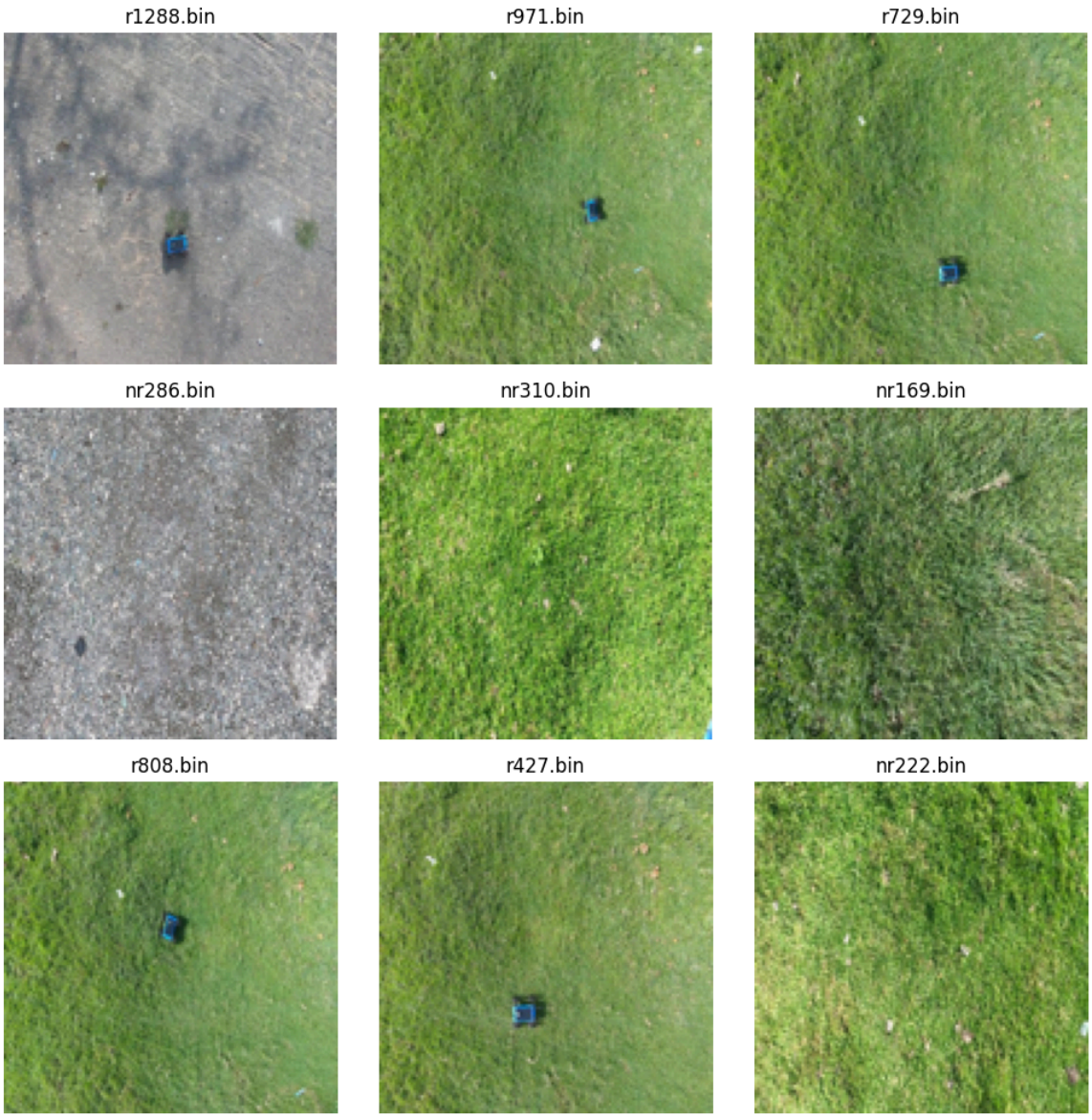
```
robot_directory = '/content/robot'

# Process images in the robot directory
process_robot_images(robot_directory)
```

```
Binary file saved as /content/robot/robot_bin/r289.bin
Binary file saved as /content/robot/robot_bin/r946.bin
Binary file saved as /content/robot/robot_bin/r153.bin
Binary file saved as /content/robot/robot_bin/r1196.bin
Binary file saved as /content/robot/robot_bin/r1139.bin
```

4. You can opt to verify the binary file

```
binary_directory = '/content/test'
verify_and_display_multiple_images(binary_directory)
```



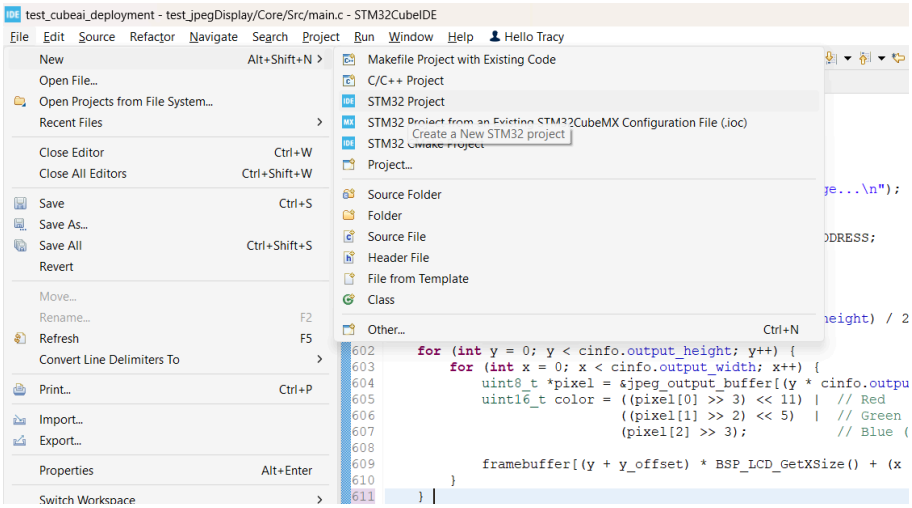
5. Download the robot and non robot binary files and save it to the sd card or choose less than 20 binary files to test it on the microcontroller for inference.

Model Deployment

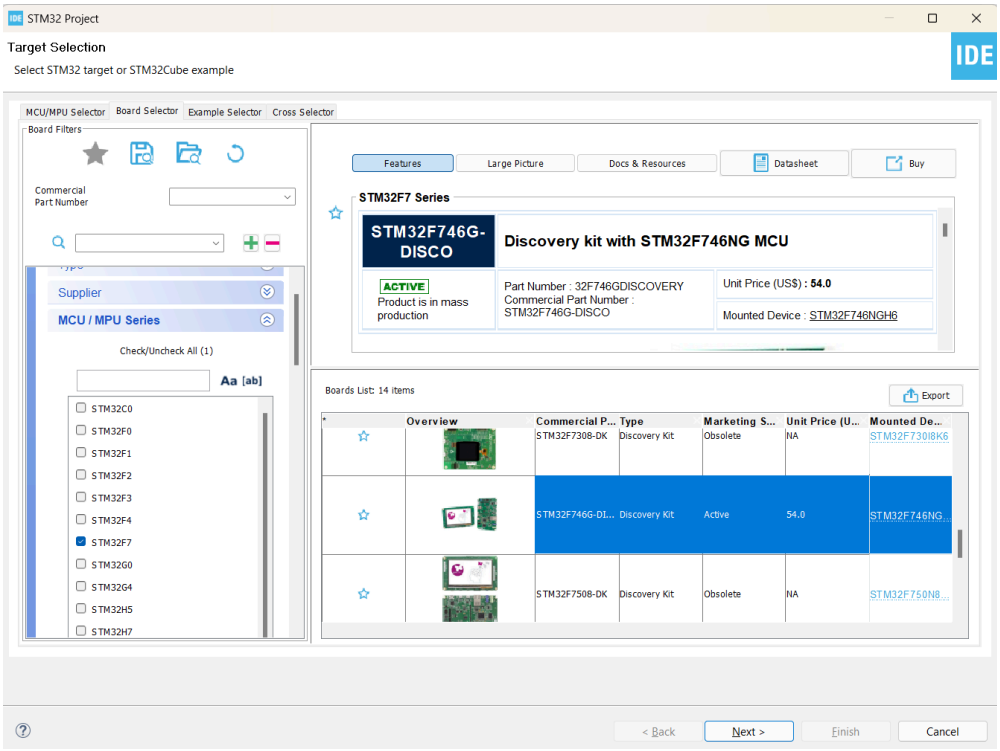
Creating an STM32 Project

This section explains how to create a new STM32 project using STM32CubeIDE

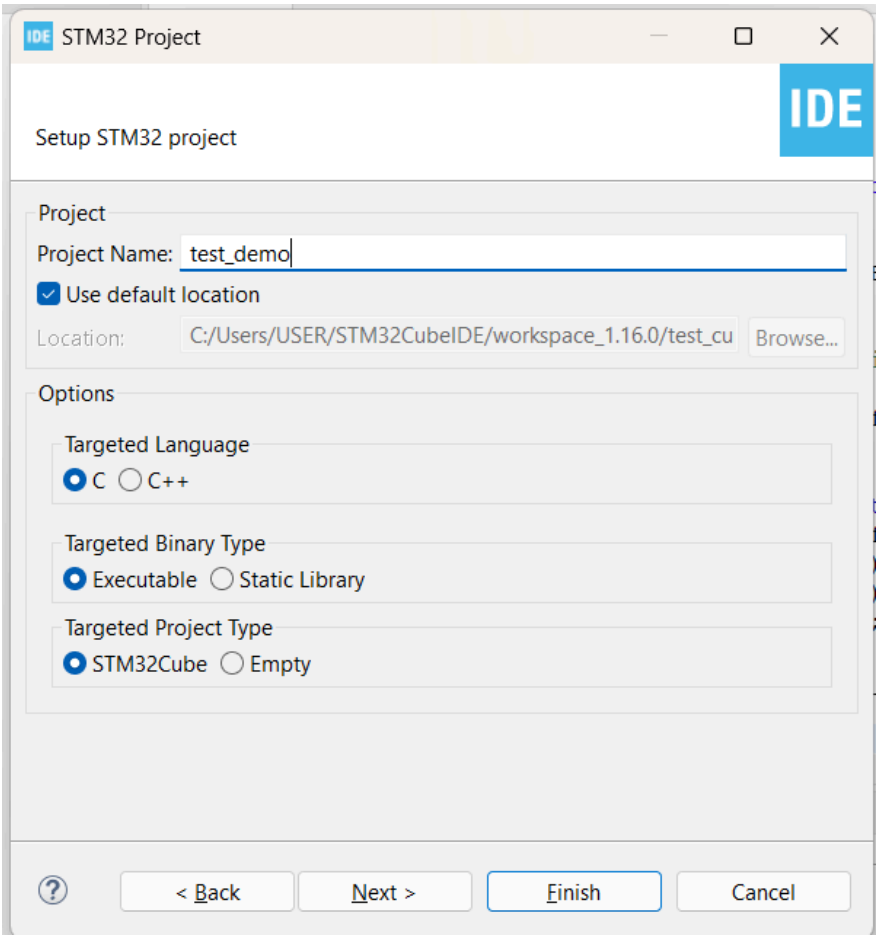
1. Go to STM32 Cube IDE > File > New > STM32 Project



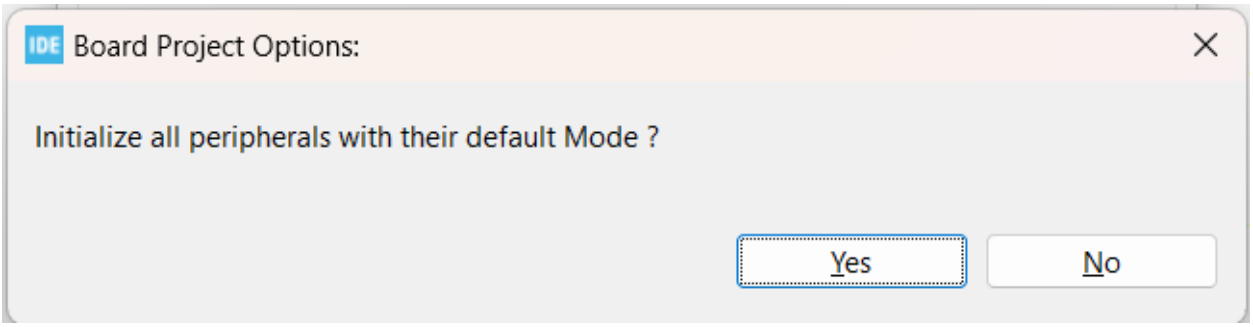
2. You will be redirected here, click on Board Selector > MCU / MPU Series > STM32F7 > STM32F746G DISCOVERY KIT > Next



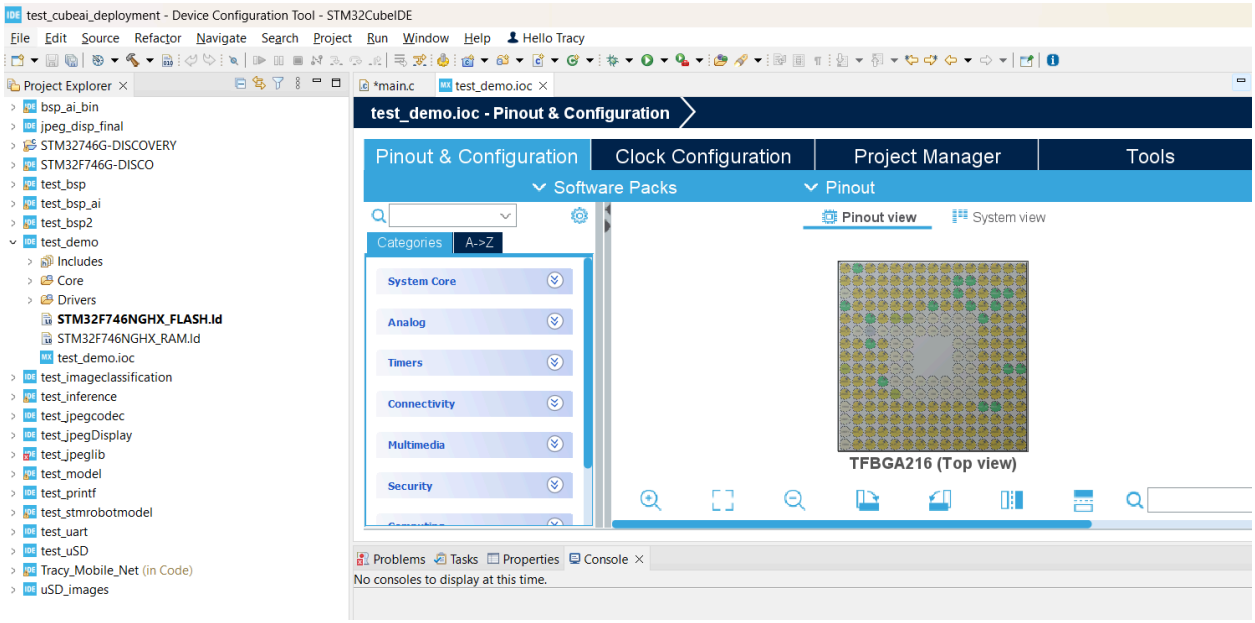
3. Name your project as test_demo or any then click on finish



4. When asked whether to initialize all peripherals with their default modes, click NO. Disabling default initialization ensures only the peripherals you manually configure are included, giving you full control over the configuration. This avoids unnecessary memory usage and simplifies debugging.



After the project is created, you will see an .ioc file automatically displayed in the workspace.

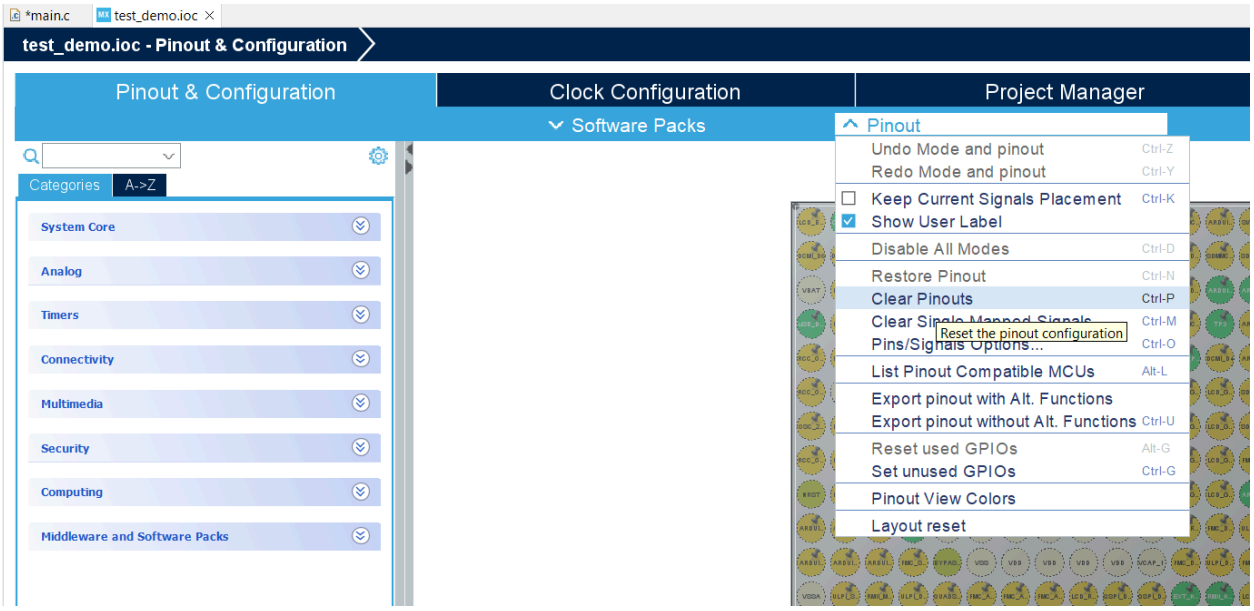


The IOC file (Initialization and Configuration file) contains the project's configuration, including peripheral settings, clock configuration, and middleware options. It allows you to modify and reconfigure the project using STM32CubeMX.

Clock configuration

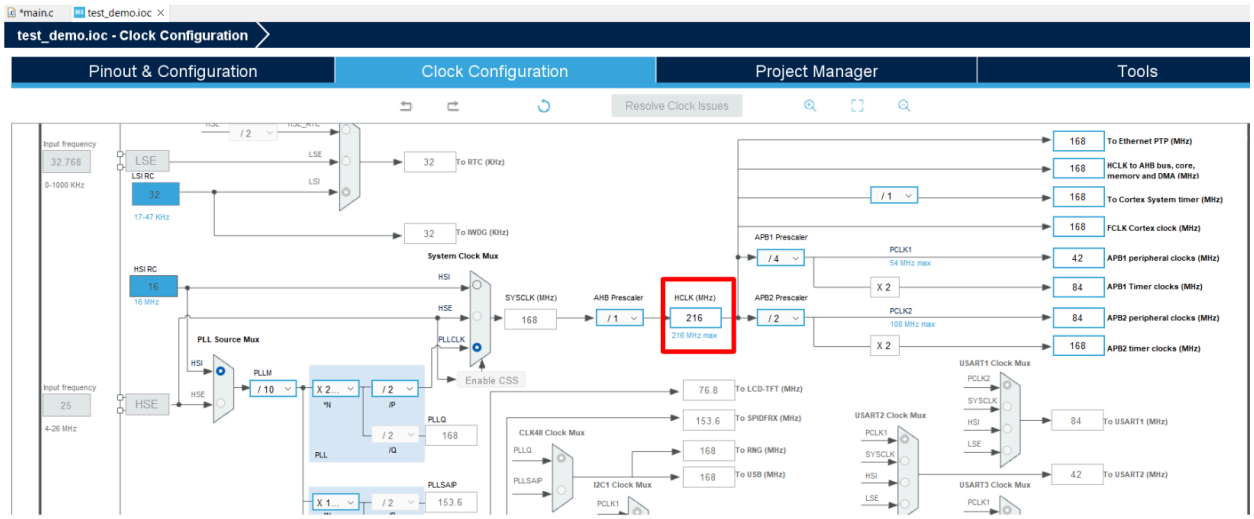
This section explains how to configure the clock for your STM32 project, specifically setting the clock frequency to its maximum value of 216 MHz. Proper clock configuration is essential to ensure the microcontroller operates efficiently

Ensure that all pins are not initialized to their default mode, click Pinout > Clear pinouts > Yes



Clearing default pinouts ensures that only the necessary pins and peripherals are configured, minimizing conflicts and reducing memory usage. When the pinouts are cleared almost all of the yellow circles will be turned gray

1. Now, set the clock frequency to maximum, go to Clock Configuration > set HCLK value as 216 then enter



Setting the clock to its maximum frequency (216 MHz for STM32F746G) ensures that the microcontroller operates at its full potential

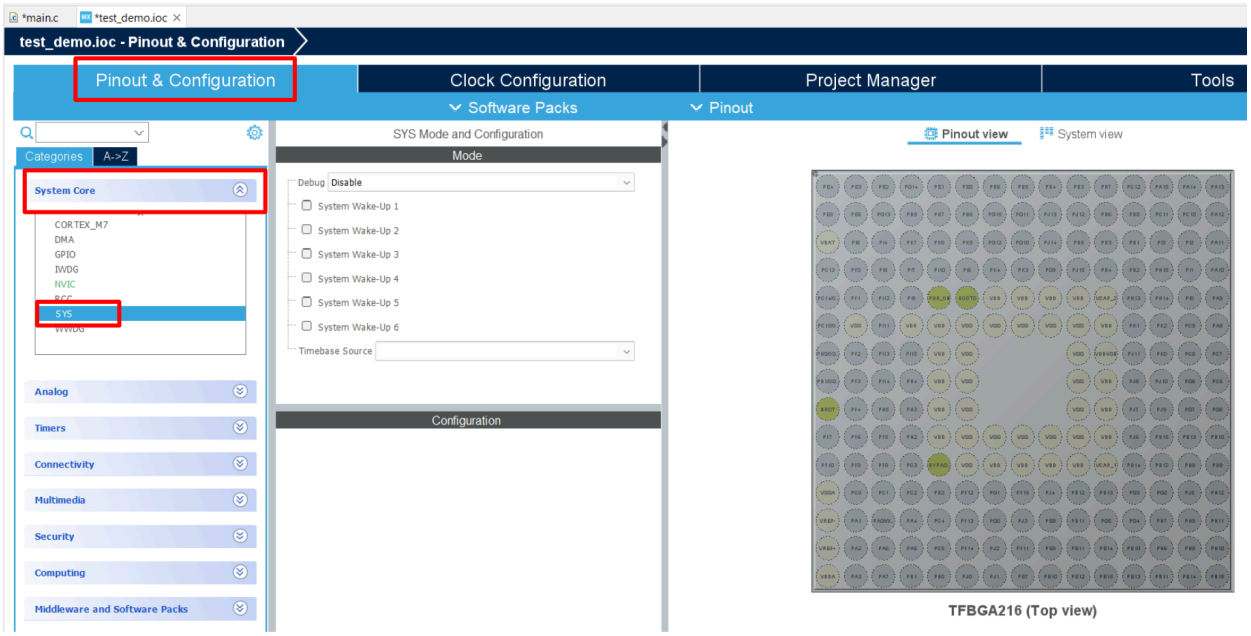
Debug Configuration

This section explains how to set up the debugging configuration for your STM32 project, specifically to view the output of printf statements in the SWV ITM Data

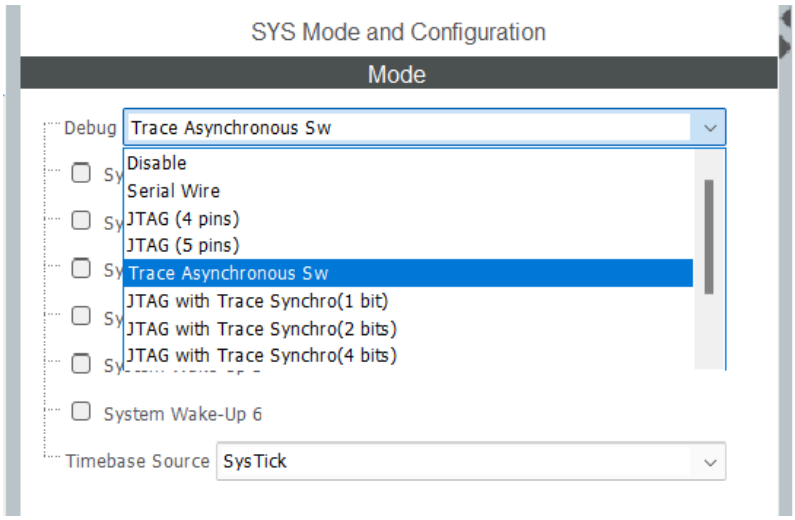
Console during a debug session. Debugging is a critical aspect of embedded development, and configuring this feature provides a convenient way to monitor application behavior.

For a step-by-step video guide on configuring the debug mode and SWV ITM Data Console, refer to this tutorial https://www.youtube.com/watch?v=WLqUImiV5Gs

1. Go to Pinout & Configuration > System Core > Sys



2. Set Mode to Trace Asynchronous SW.



The Trace Asynchronous SW is a Serial Wire Debug (SWD) mode that enables real-time data tracing and debugging features, such as the SWV ITM Data Console. It supports efficient debugging over a single wire.

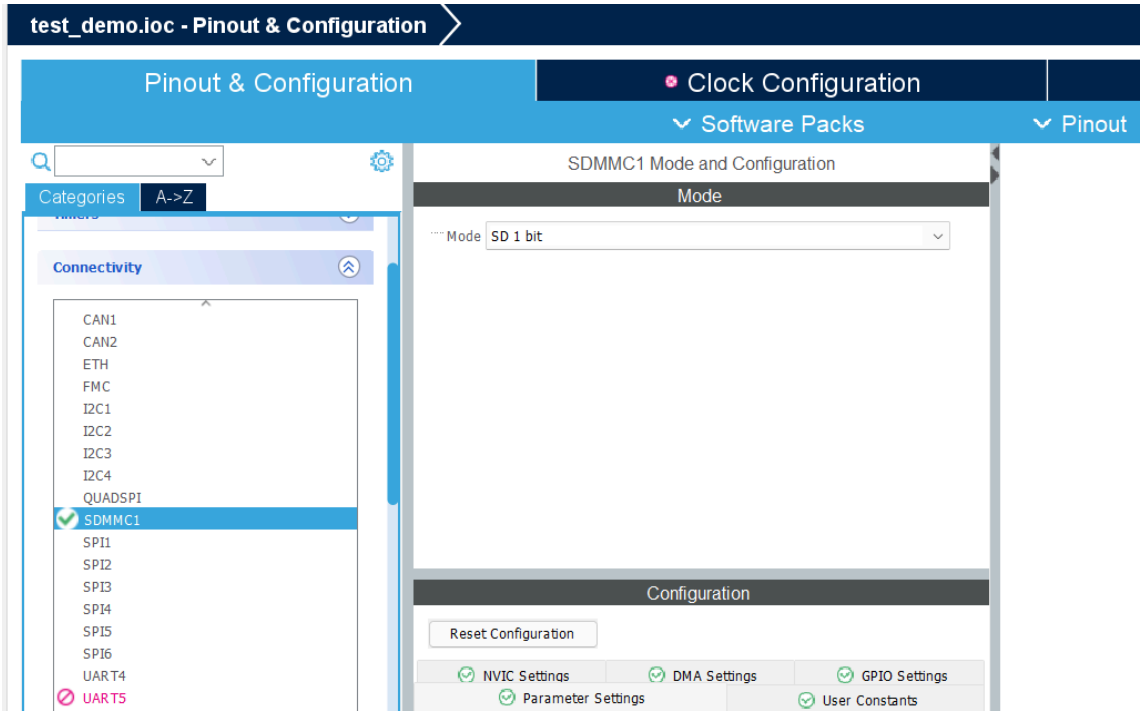
Creating a File System

This section guides you through the process of configuring an SD card as a file system for your STM32F746G Discovery board. STM32 microcontrollers can use FATFS (a lightweight file system) to manage files on SD cards.
Refer the tutorial here: [STM32 – Creating a File System on a SD card]

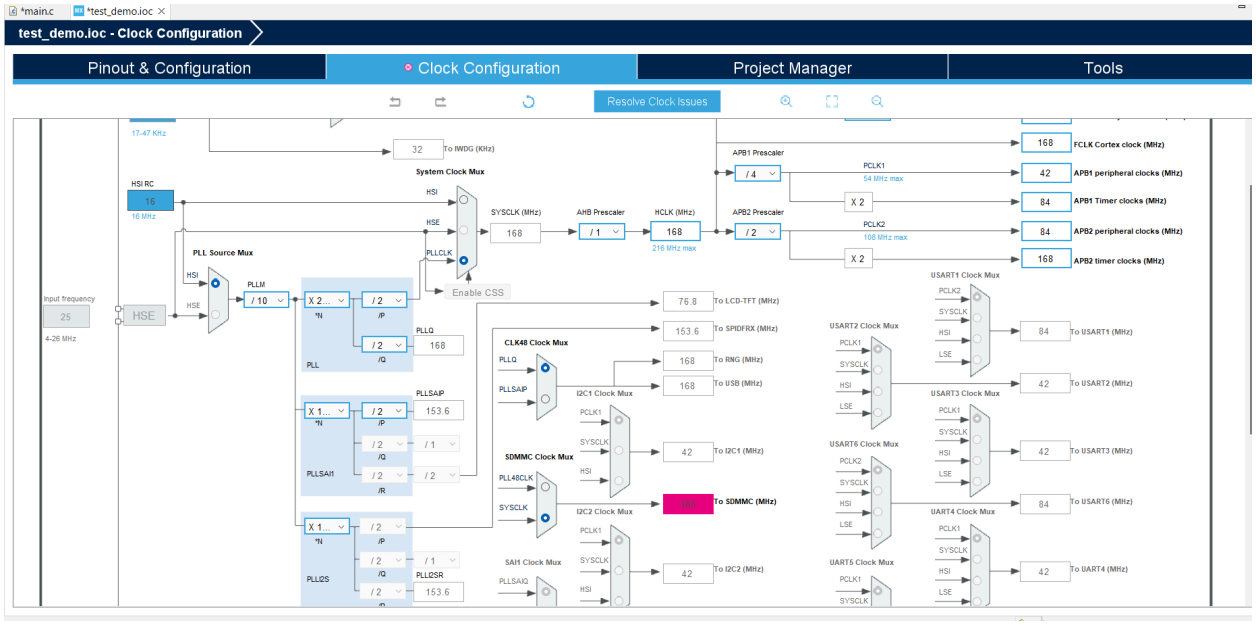
SD Card configuration

Refer to the user Manual of the microcontroller, The 32F746GDISCOVERY board supports the 2-Gbyte (or more) microSD™ card connected to the SDMMC1 port of STM32F746NGH6.
[Discovery kit for STM32F7 Series with STM32F746NG MCU - User manual]

1. Go to Connectivity > SDMMC1 > Set mode to SD 1 Bit

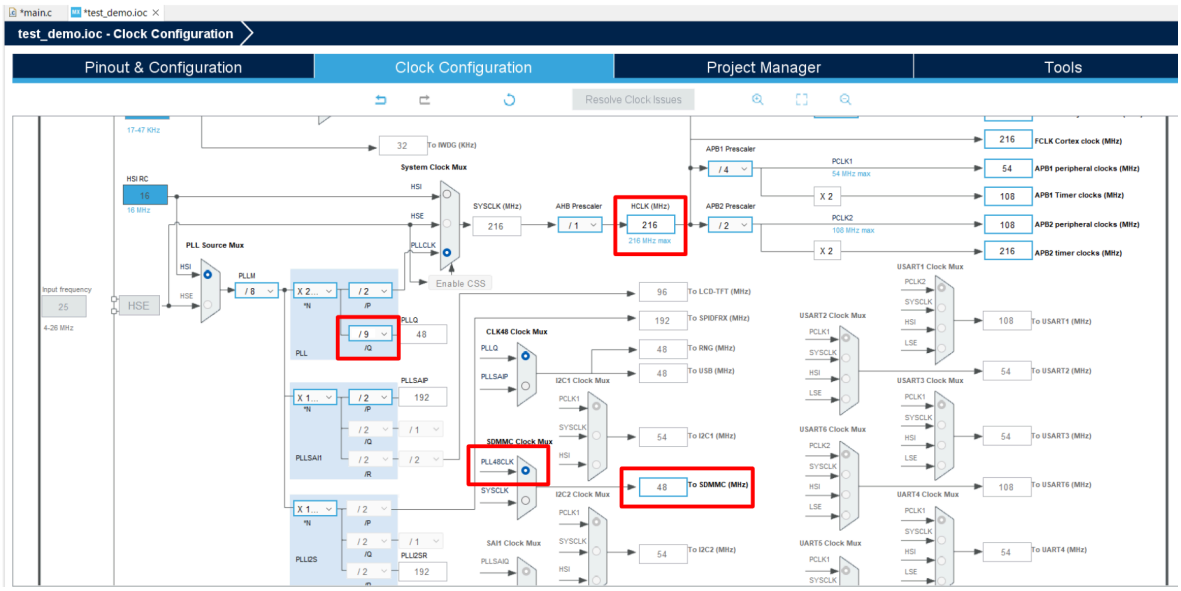


2. As you can see, the clock configuration has some problems, go to Clock Configuration

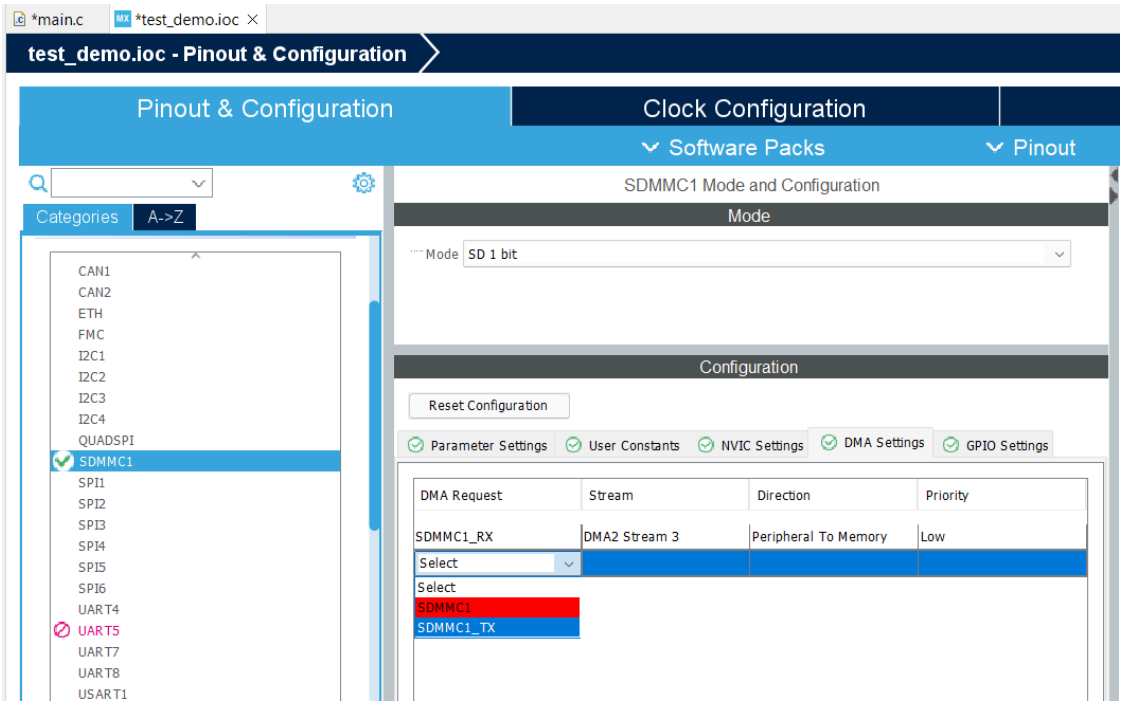


The HCLK value has a value 168 even though we have set it as 216 before. This is because we were configuring the SDMMC1 without setting its appropriate value for its clock

3. Refer to the user manual, the data transfer of SDMCC1 is only up to 50 MHz. Set the HCLK back to 216, choose PLL48CLK as SDMMC1 Clock Mux, set SDMCC to 48 and set the PLLM Q to 9 then enter. You can manually do it too as long as the SDMMC1 clock does not exceeds 50

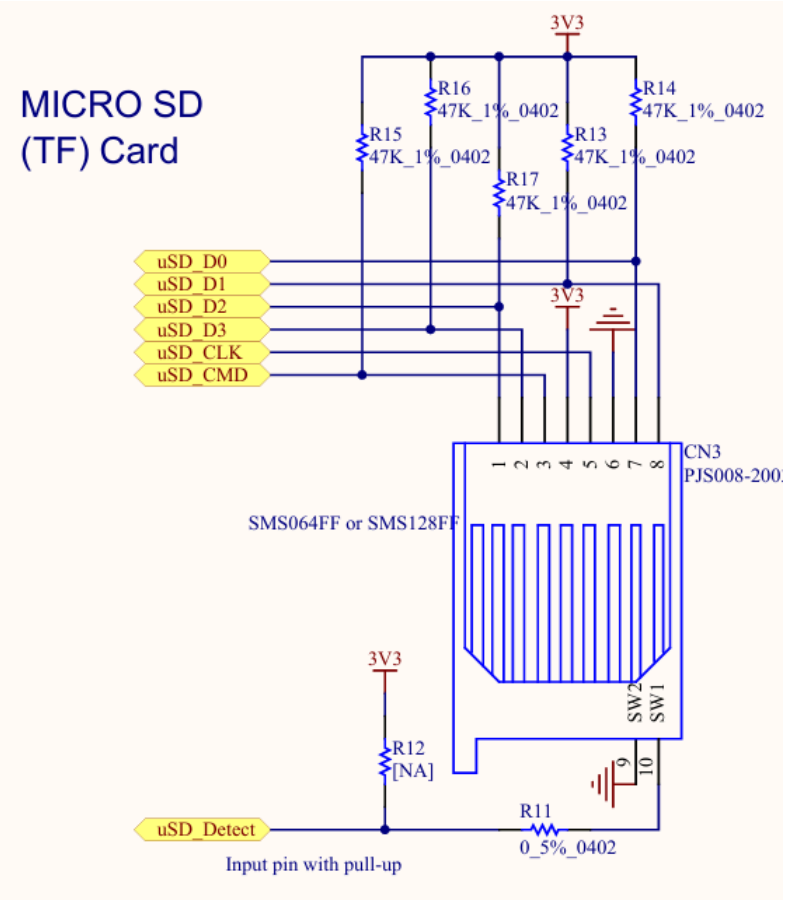


4. Enable and configure the DMA for SDMMC1 to improve performance during read/write operations. Go back to Pinout & Configuration > SDMMC1 > DMA Settings > Select Add > Select SDMMC1_RX > Add Again > Select SDMMC1_TX

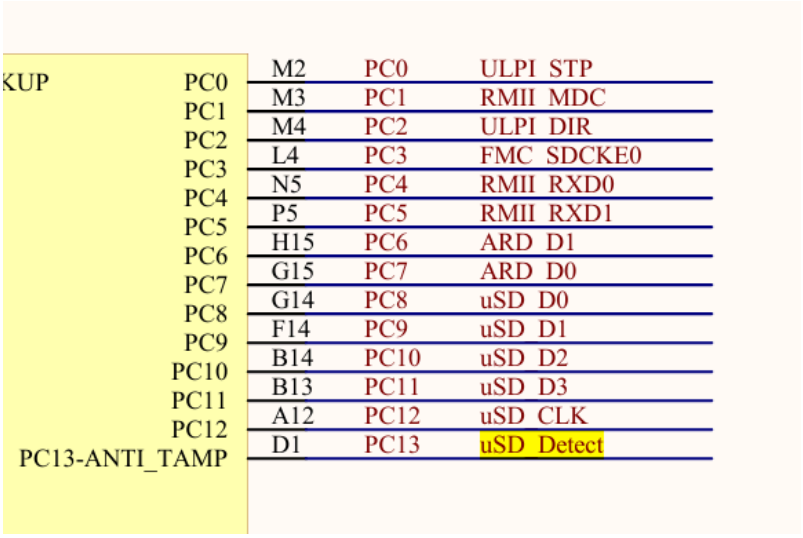


Adding DMA can help improve performance during read/write operations by reducing CPU involvement.

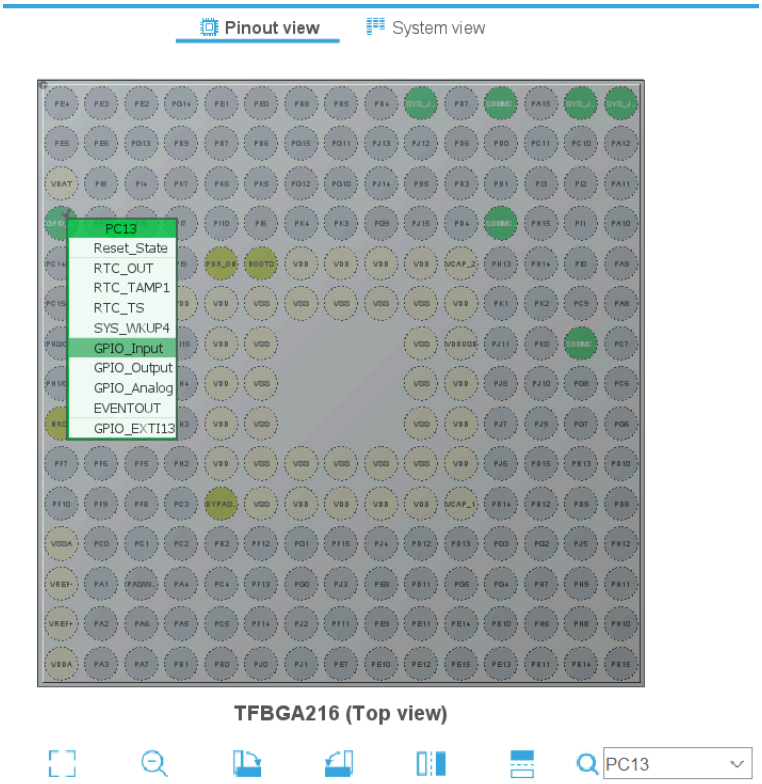
5. Locate uSD_Detect Pin. Check the schematic for the STM32F746G Discovery Board. The uSD_Detect pin is connected to PC13. The schematic pack are available under the 32F746GDISCOVERY page under CAD Resources or refer to this link, https://www.st.com/resource/en/schematic_pack/mb1191-f746ngh6-b02_schematic.pdf



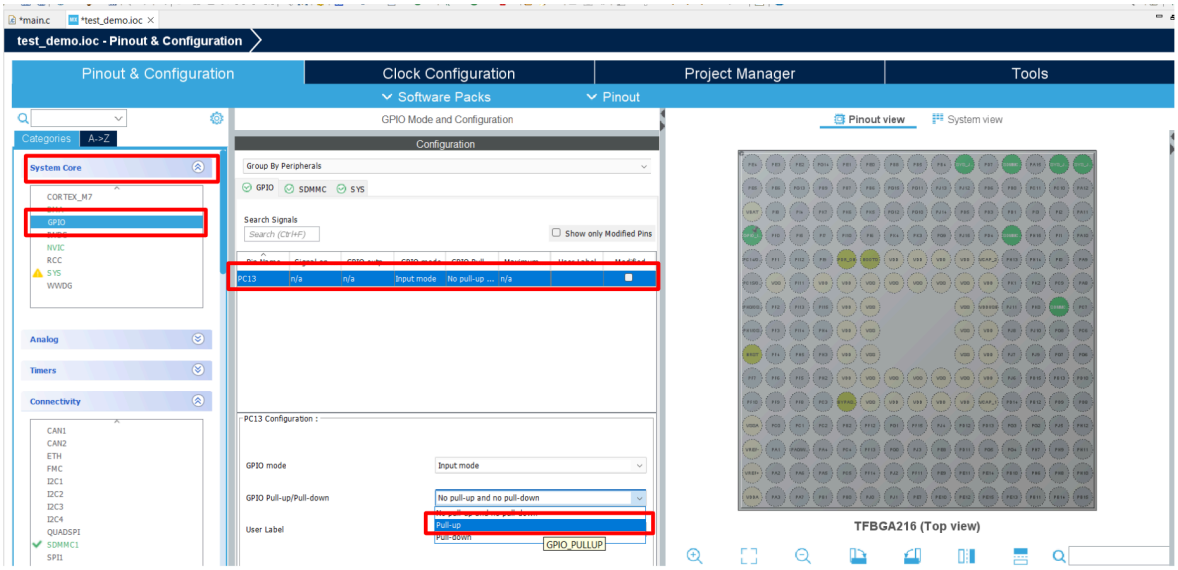
6. Notice that uSD_Detect is connected to PC13, so we need to configure a GPIO pin **PC13** as an **Input** for SD card presence detection and enable the pull up



7. Search for PC13 then set it as GPIO_Input

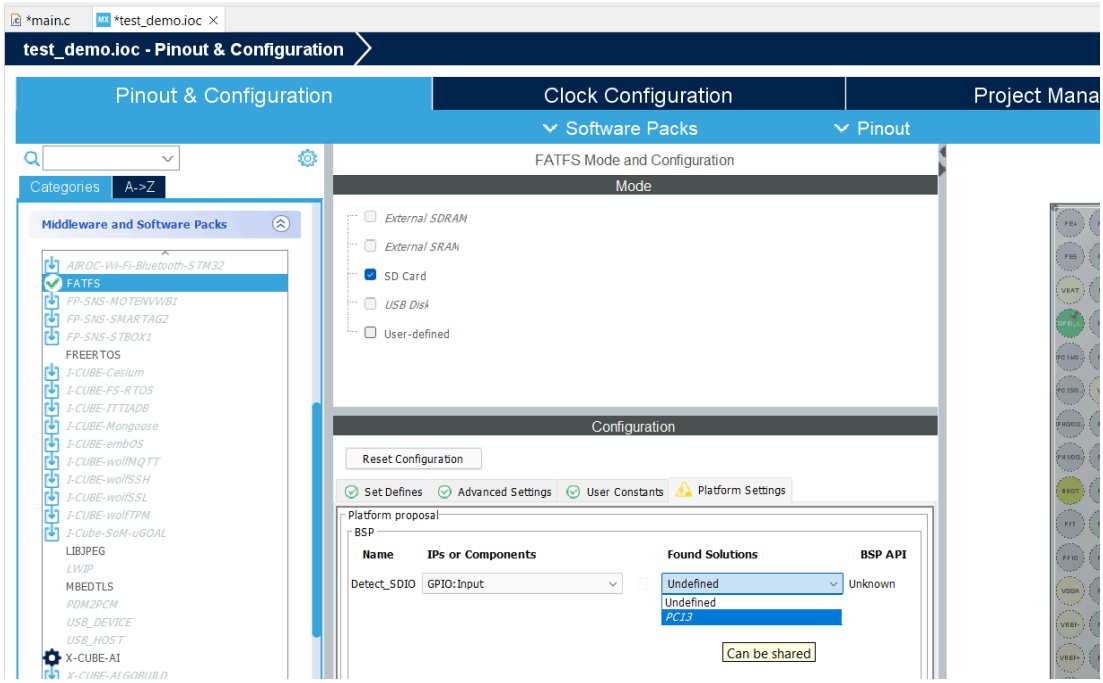


8. To enable pull up, go to pinout & configuration > System Core > GPIO > PC13 > in PC13 configuration panel, select Pull up in the GPIO Pull-up/Pull-down



Enable FATFS

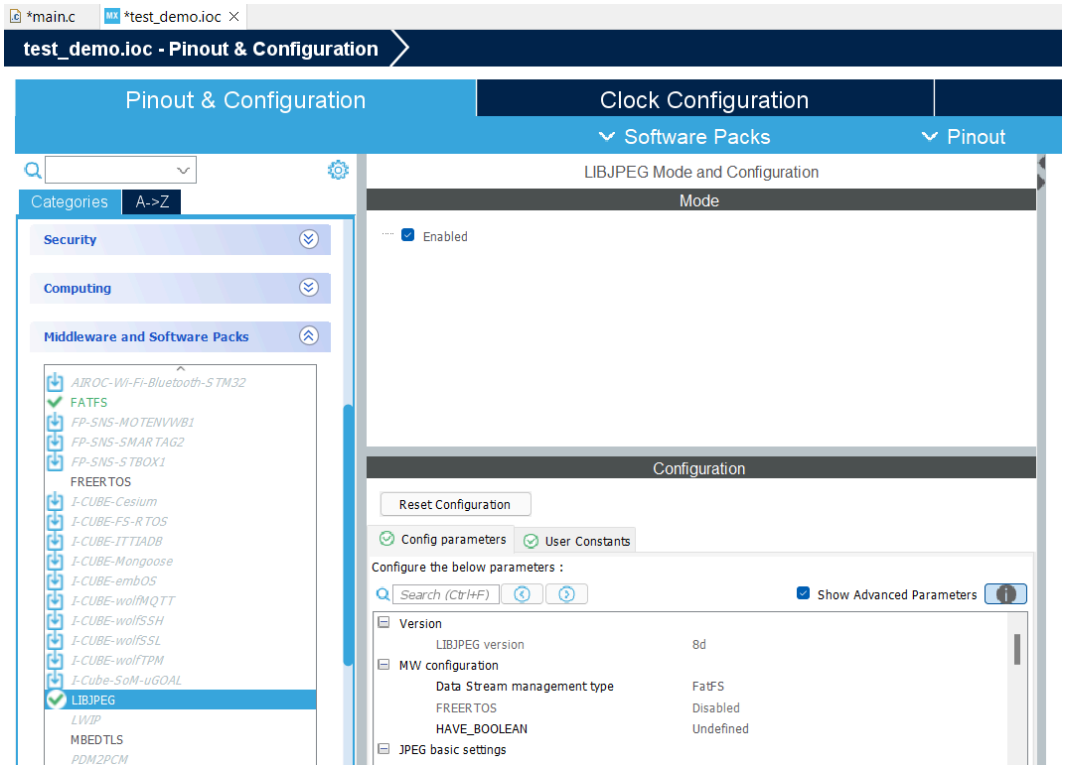
To add the middleware, go to Middleware and Software Packs > Set FATFS Mode to SD Card > In platform settings, select PC13



Enable LIBJPEG

LIBJPEG is a middleware library that allows STM32 microcontrollers to handle JPEG image decompression. It converts compressed JPEG files into raw image data that can be processed or displayed by the microcontroller

1. Enable LIBJPEG, go to > Middleware and Software Packs > LIBJPEG > Enable

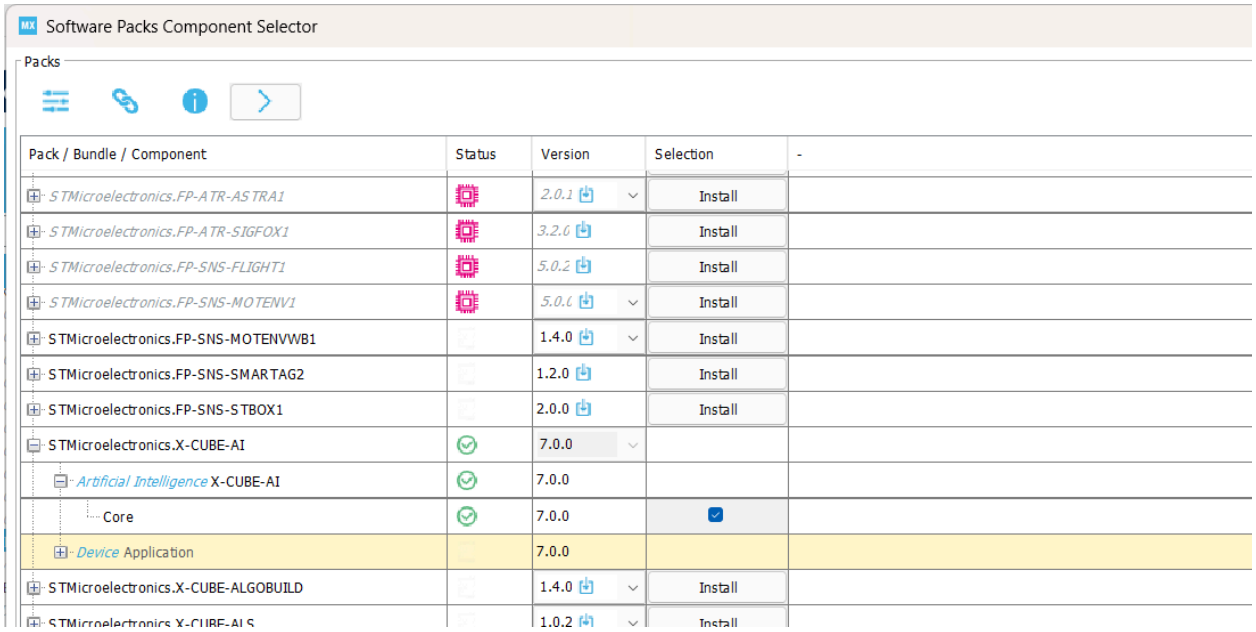


Using the X-CUBE-AI Configuration Wizard

To configure X-CUBE-AI and convert your pre-trained model into a C file.

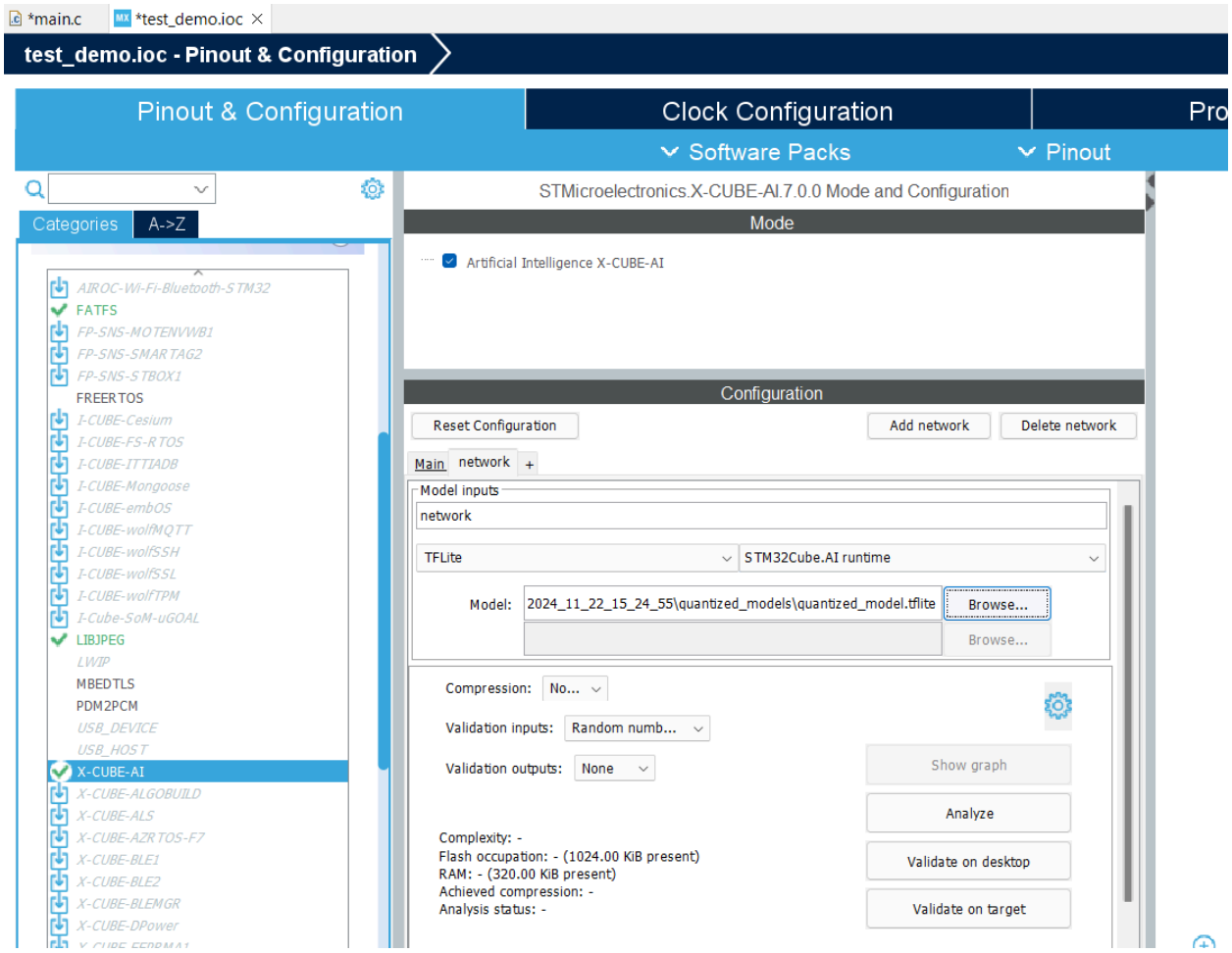
For detailed guidance on using the X-CUBE-AI Configuration Wizard, refer to the X-CUBE-AI User Manual:
[Getting started with X-CUBE-AI Expansion Package for Artificial Intelligence (AI) - User manual]

1. Go to Middleware and Software Packs > X-CUBE-AI > Select X-CUBE-AI version 7.0.0 > Enable AI X-CUBE-AI Core > Click OK



   a. For device application, you can explore to use other templates like validation, system performance and application
      i. **System performance** : standalone AI test application for performance purpose
      ii. **Validation** : AI test application for validation purpose
      iii. **Template application** : basic application template for AI application

2. In the X-CUBE-AI Configuration panel, upload your model, click Add Network > then browse > then Analyze

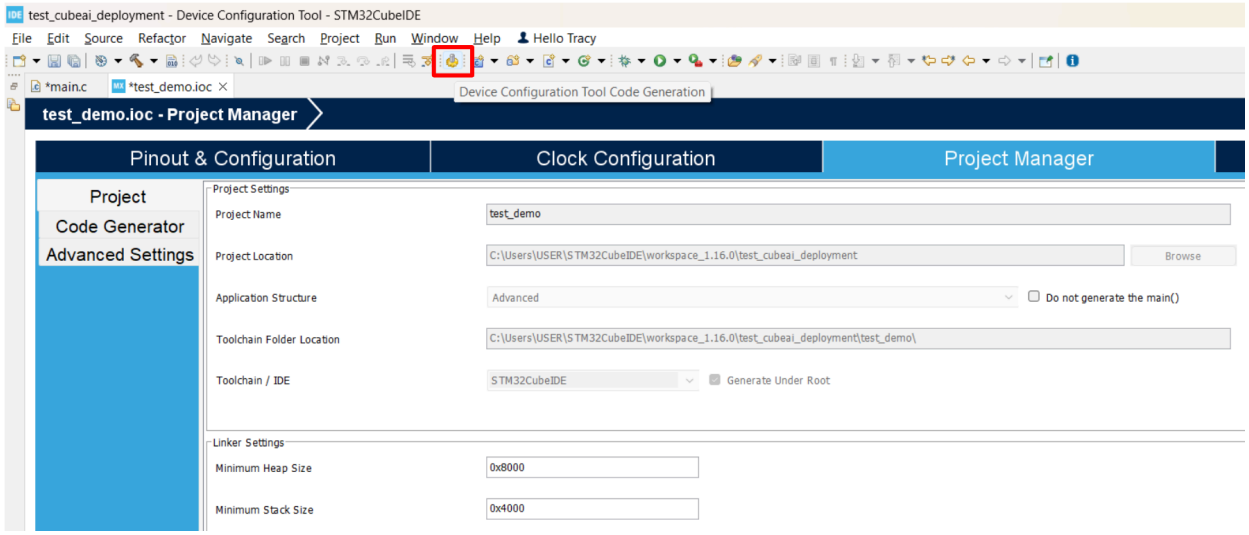


3. X-CUBE-AI allows you to validate your model in two ways:
    a. **Validate on desktop** runs inference on random input data and analyzes model size, ensuring compatibility with your microcontroller.
    b. Validate on target Executes inference directly on the microcontroller using random input data.
    Note: Ensure your STM32F746G Discovery Board is connected to your PC during this step.

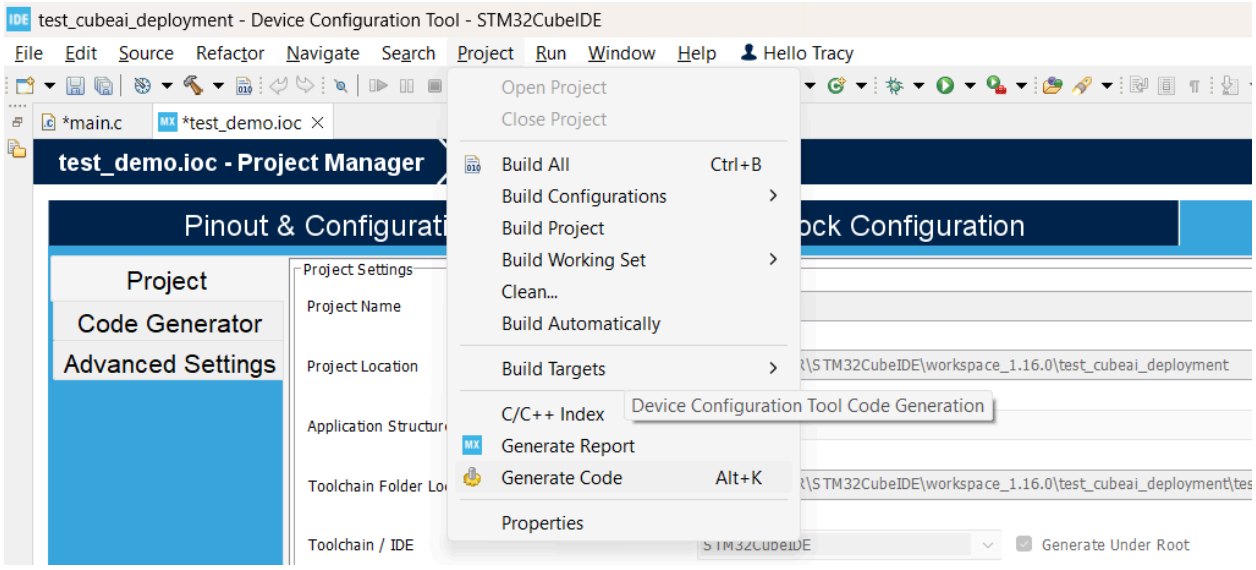
Code Generation

Now that the clock, debug, file system, JPEG, and AI configurations are complete, and all peripherals and pins are set, follow these steps to generate the code:

1. Save Your Configuration, navigate to File > Save to save the current setup.
2. Go to File > then go to Project Manager > Set the heap and stack size to 0x8000 and 0x4000 > click the Generate Code icon.



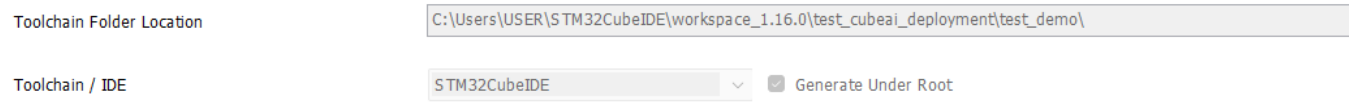
3. Alternatively, you can go to Project > Generate Code to start the process.



Exploring STM32CubeMX for Configuration

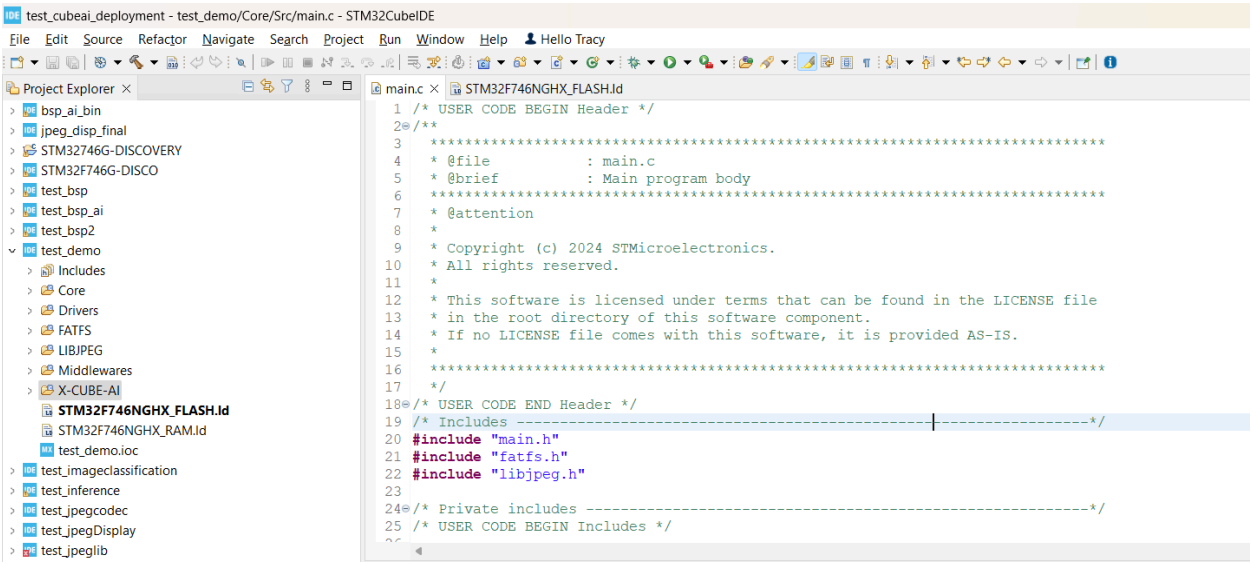
You can explore STM32CubeMX to configure the clock, debug, file system, JPEG, and AI settings. The process remains the same, but ensure the following:

- Specify your toolchain as STM32CubeIDE in the Project Manager.



Post Code Generation

After generating the code, STM32CubeMX will automatically create the configuration files for the settings you defined.Insert your custom code into the User Code Blocks in the generated files. This ensures that your code is preserved when reconfiguring the project through STM32CubeMX.



Board Specific Drivers

This section provides a detailed guide for integrating BSP (Board Support Package) drivers into your STM32 project. BSP drivers are hardware-specific drivers provided by STM32CubeF7. They simplify interactions with the peripherals on the STM32F746G Discovery board, such as the LCD and SDRAM, by offering high-level APIs.

For displaying strings and image, we will use BSP Drivers.

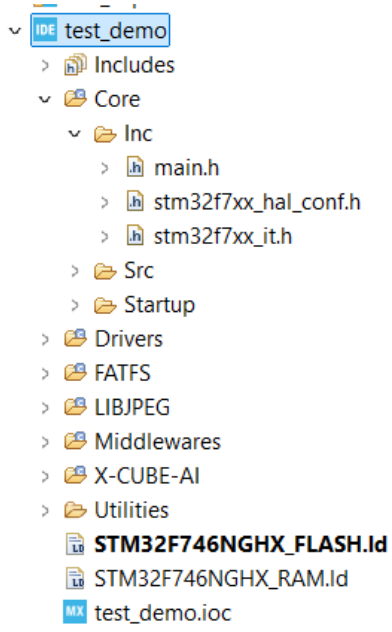
1. Add the following header files in your main file to access BSP functionalities

```
#include "stm32746g_discovery.h"
#include "stm32746g_discovery_lcd.h"
#include "stm32746g_discovery_sdram.h"
```

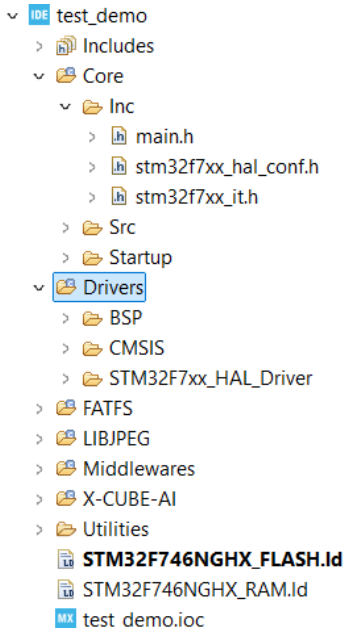
2. Enable required HAL modules. Go to Core folder > Inc > stm32f7xx_hal_conf.h and make sure to enable these following modules

```
#define HAL_MODULE_ENABLED
#define HAL_DMA2D_MODULE_ENABLED
#define HAL_SDRAM_MODULE_ENABLED
#define HAL_LTDC_MODULE_ENABLED
#define HAL_SD_MODULE_ENABLED
#define HAL_UART_MODULE_ENABLED
#define HAL_DSI_MODULE_ENABLED
#define HAL_JPEG_MODULE_ENABLED
#define HAL_GPIO_MODULE_ENABLED
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
#define HAL_PWR_MODULE_ENABLED
#define HAL_I2C_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED
```

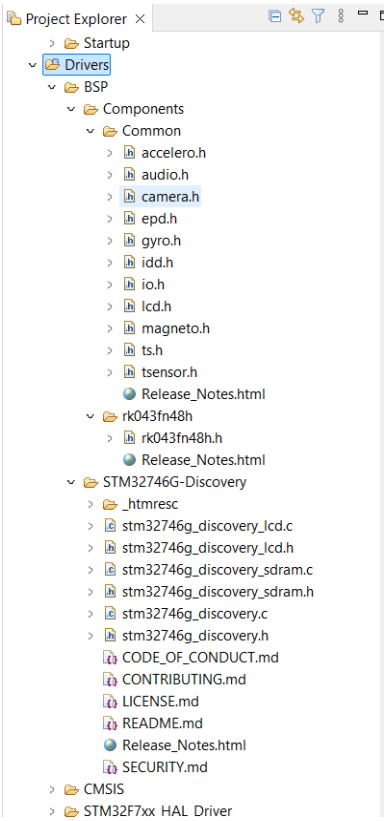
3. Add Fonts for Display. From the STM32CubeF7 repository, copy the Utilities folder from here https://github.com/STMicroelectronics/STM32CubeF7/tree/master/Utilities/Fonts , remove unnecessary folders from Utilities and keep only the Fonts folder
4. Paste it into your project directory under test_demo/Utilities



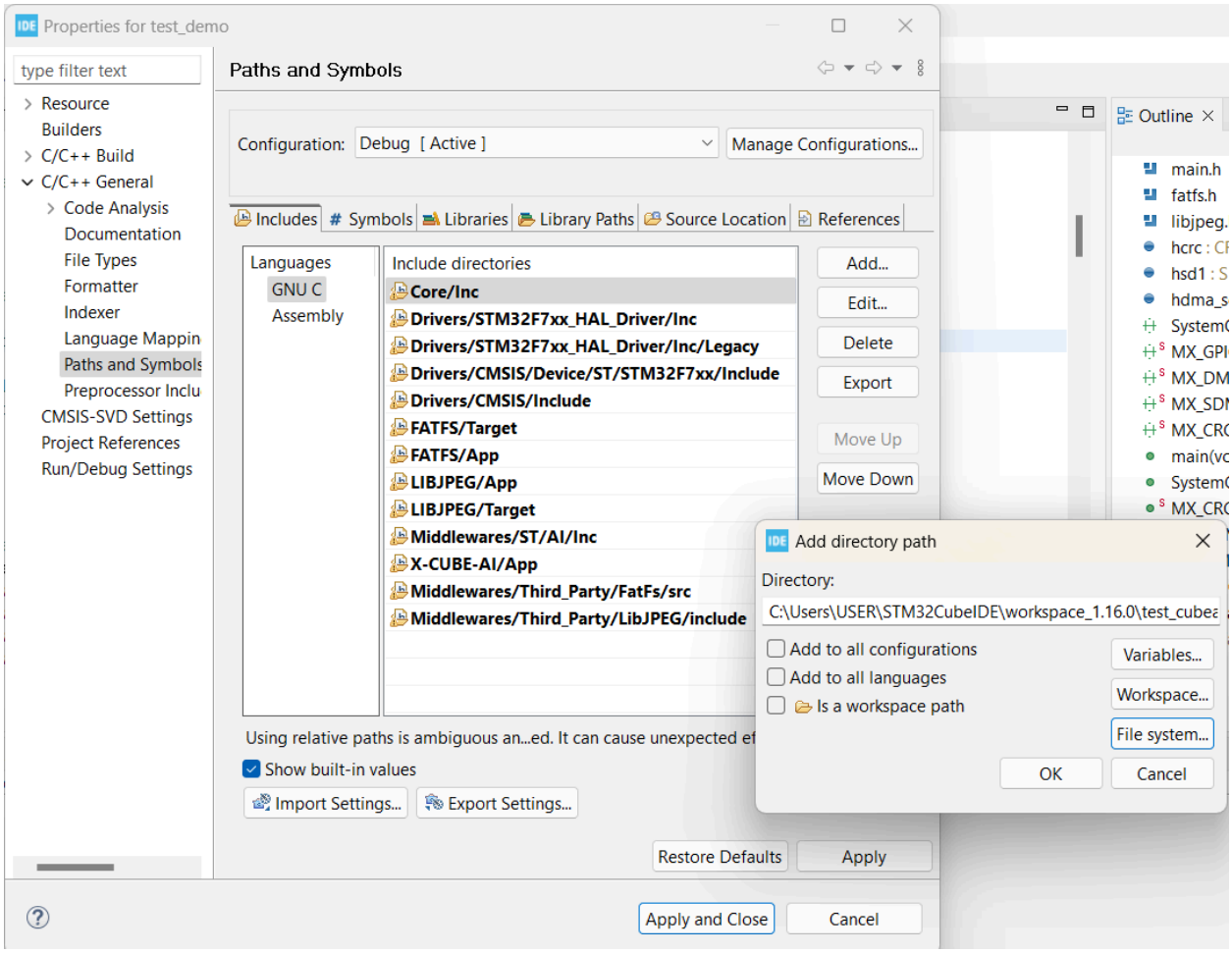
5. Create a BSP folder inside your project's Drivers directory, create a folder named BSP.



6. Go to the STM32CubeF7 repository and copy the following subfolders.
    a. `BSP/Components/Common`
    b. `BSP/Components/rk043fn48h`
    c. `BSP/STM32746G-Discovery`

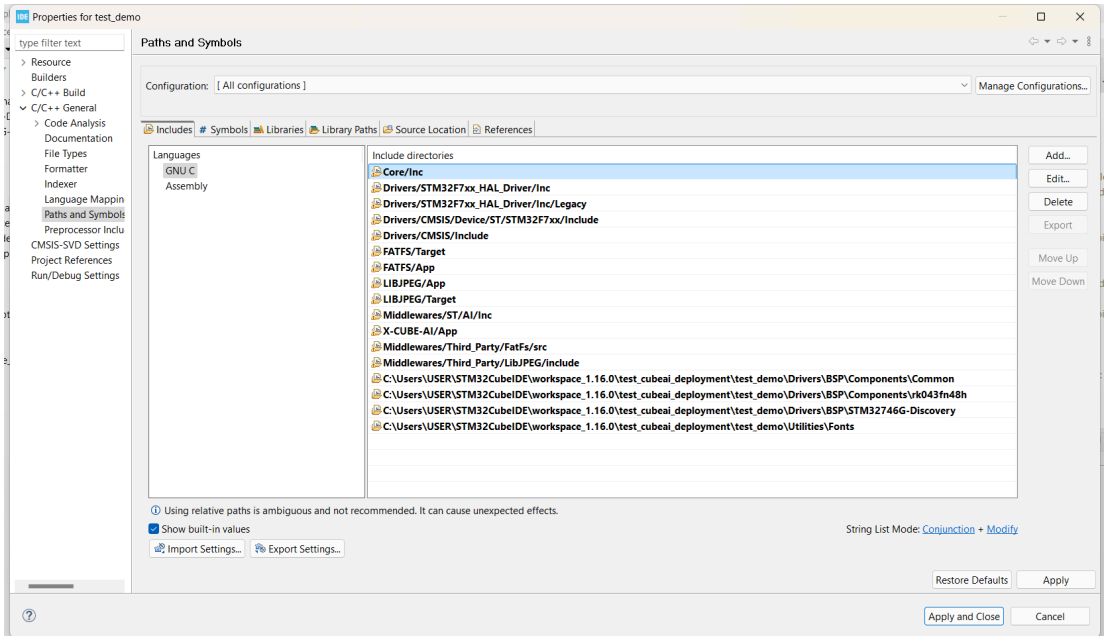


7. Add BSP Paths to Project. Go to Project > Properties > C/C++ General > Paths and Symbols > Set Configuration to All Configuration > In the Includes Panel, click ADD > File System > Navigate to your projects drivers directory.

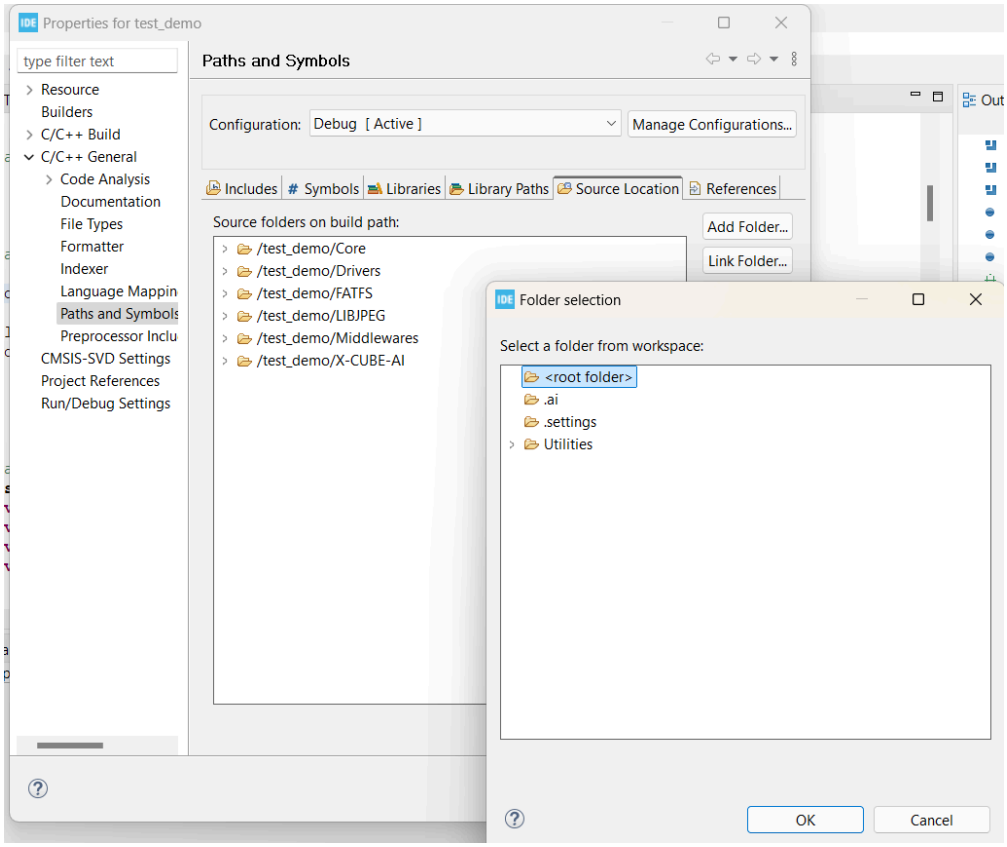


Add the following paths:

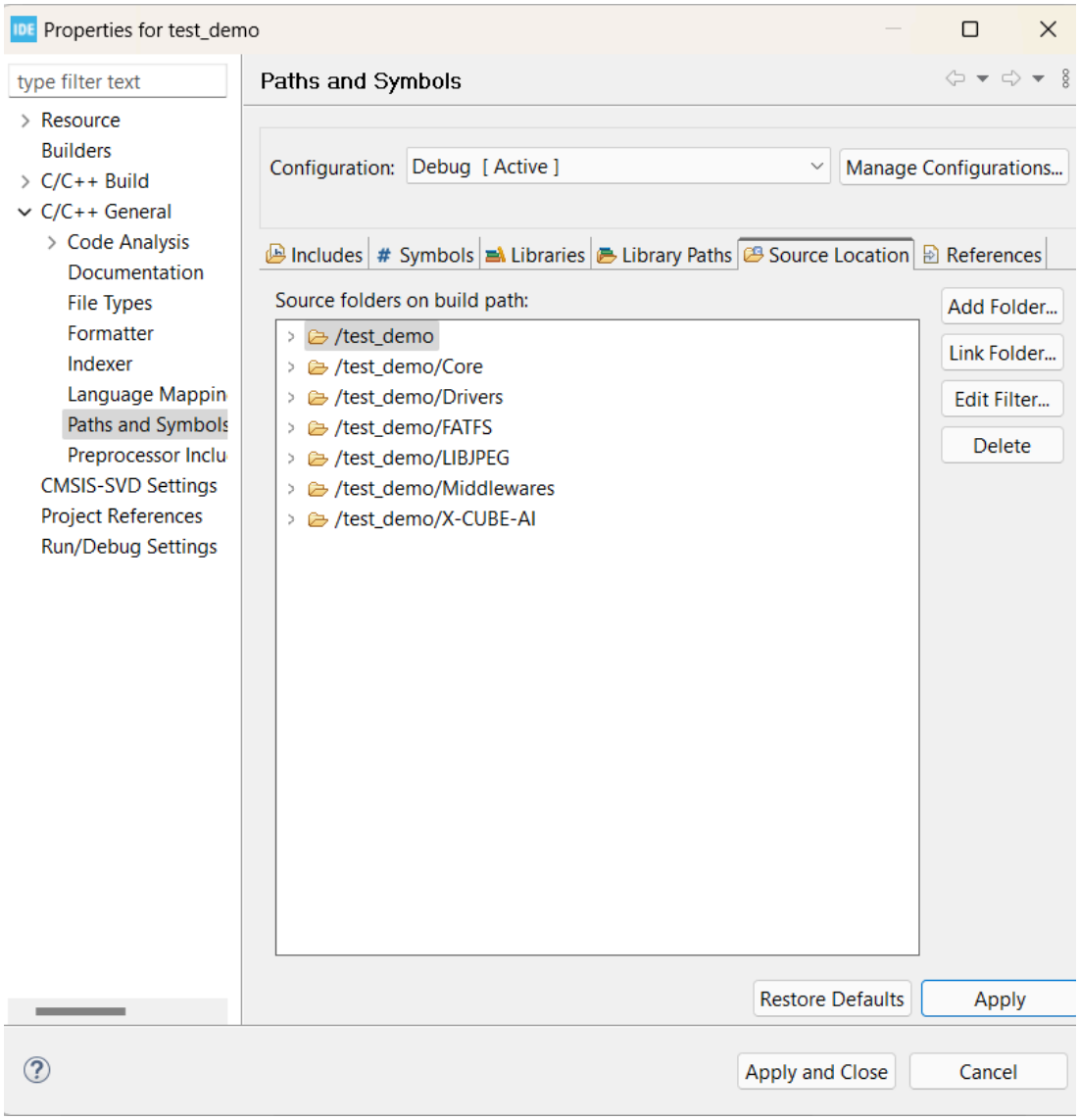
a. BSP\Components\Common
b. BSP\Components\rk043fn48h
c. BSP\Components\STM32746G-Discovery
d. Utilities\Fonts



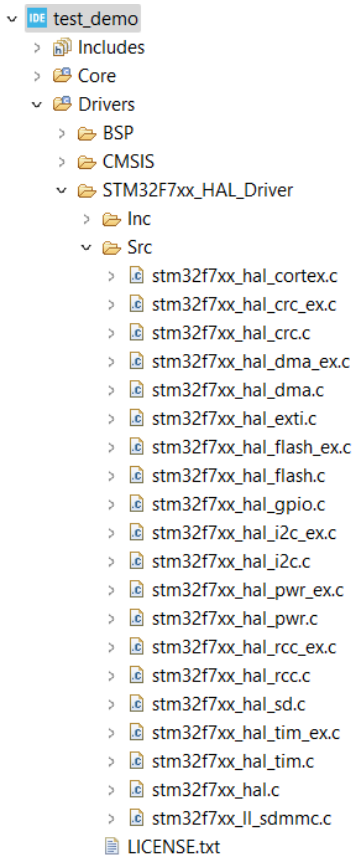
8. Add Root Folder. Select configuration as Debug then go to Source Location > Add folder > <root folder>



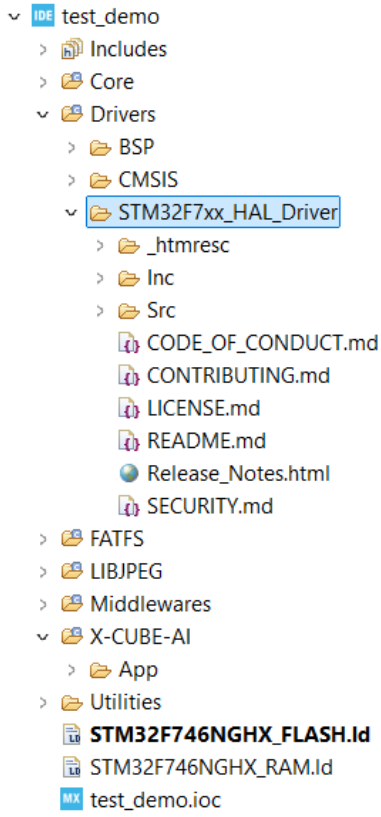
9. Click Apply, then Rebuild Index, and finally Apply and Close



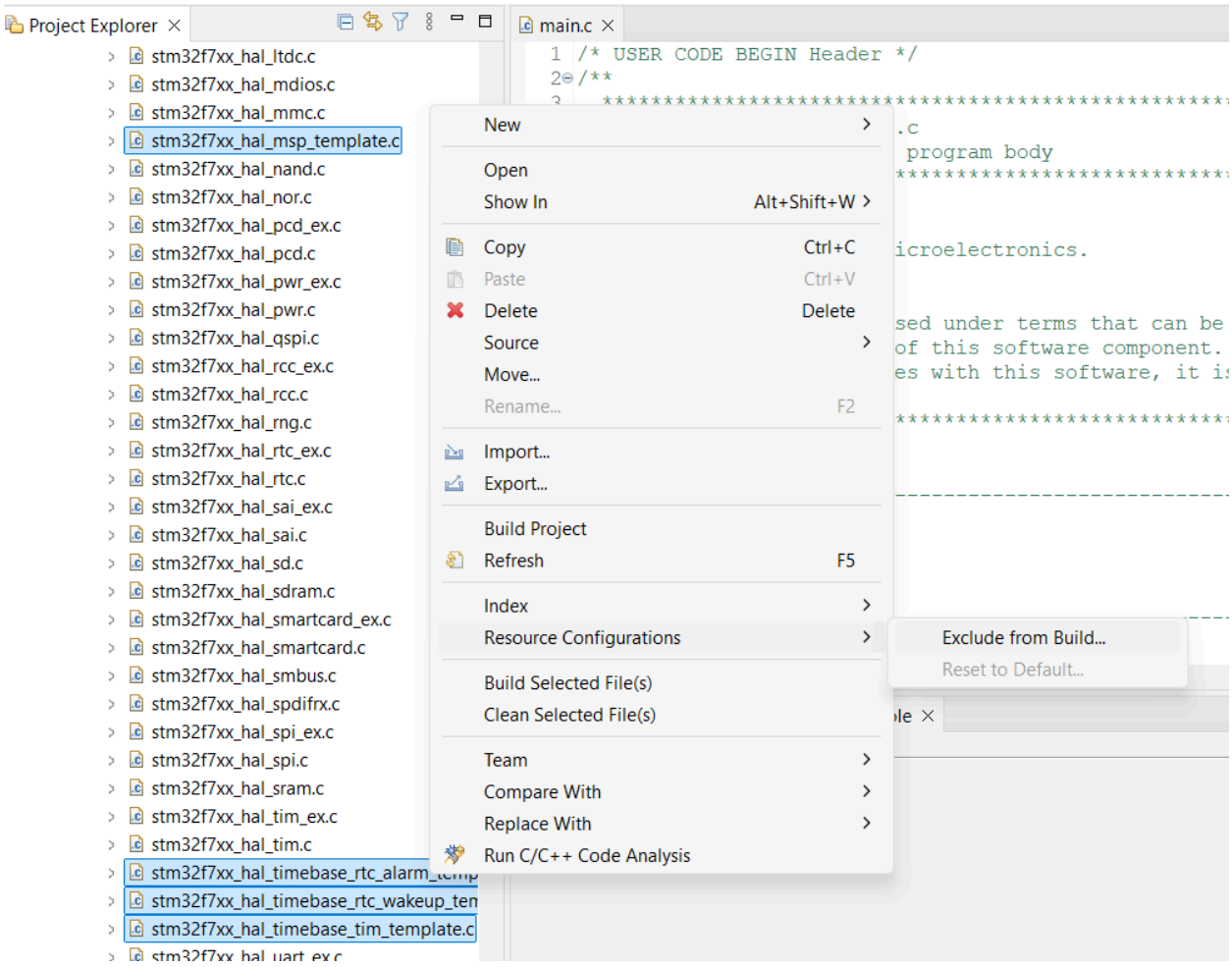
10. Update STM32F7xx_HAL_Driver, this is the original contents of STM32F7xx_HAL_Driver that is included during code generation,



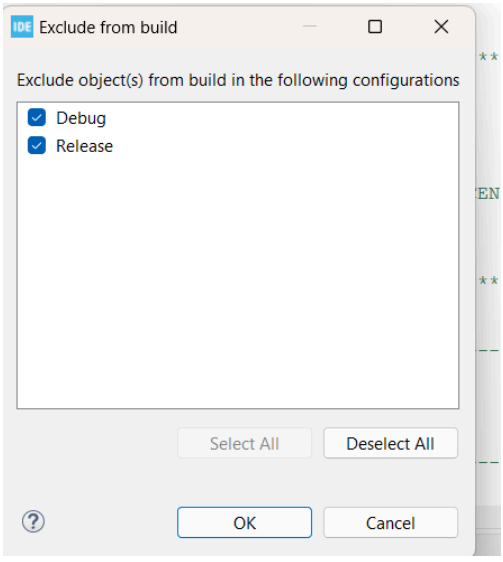
11. You can either delete this folder or add missing source and header files. If you want to delete the contents of your STM32F7xx_HAL_Driver folder in your project, make sure you download the STM32F7xx_HAL_Driver from this repo: STMicroelectronics/stm32f7xx-hal-driver at da6f1efa878f545dc45278fdae6dab85a680cb80. Then copy the contents of it and paste it on STM32F7xx_HAL_Driver folder in your project


test_demo
Includes
Core
Drivers
BSP
CMSIS
STM32F7xx_HAL_Driver
_htmresc
Inc
Src
CODE_OF_CONDUCT.md
CONTRIBUTING.md
LICENSE.md
README.md
Release_Notes.html
SECURITY.md
FATFS
LIBJPEG
Middlewares
X-CUBE-AI
App
Utilities
STM32F746NGHX_FLASH.ld
STM32F746NGHX_RAM.ld
test_demo.ioc

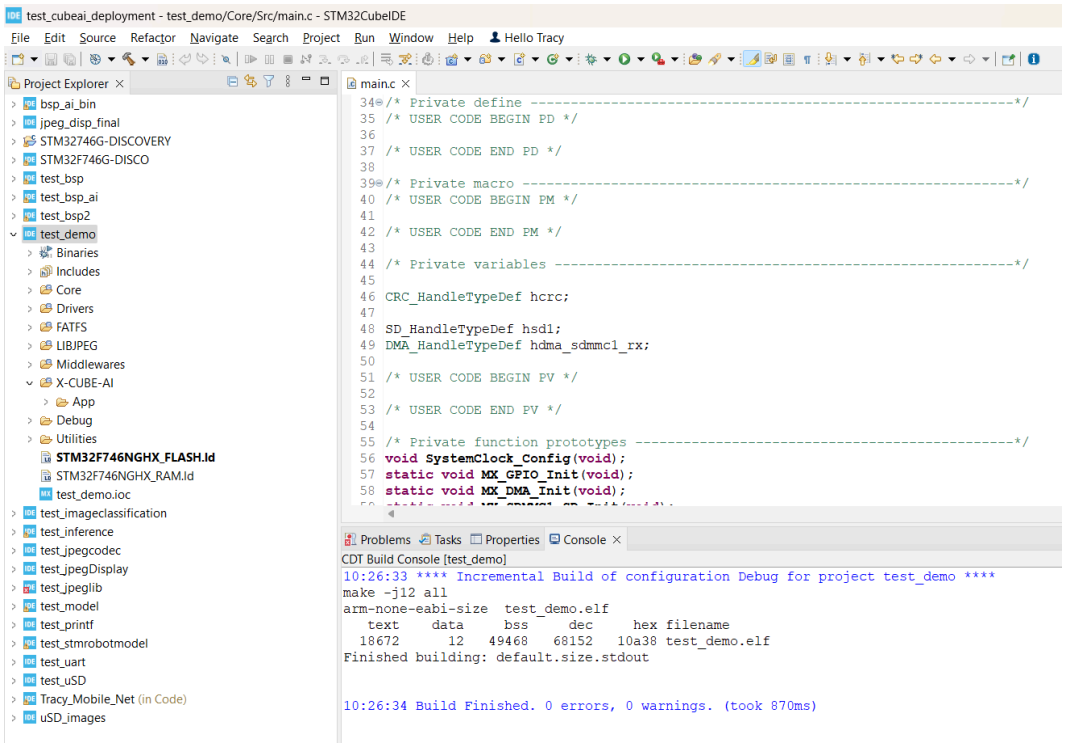
12. In your STM32F7xx_HAL_Driver folder exclude the following unnecessary source files during build.



13. Select All then OK



14. Build the project, click on Project . Build Project



The drivers and paths are now fully set up, enabling you to display strings and images or manage peripherals effectively.

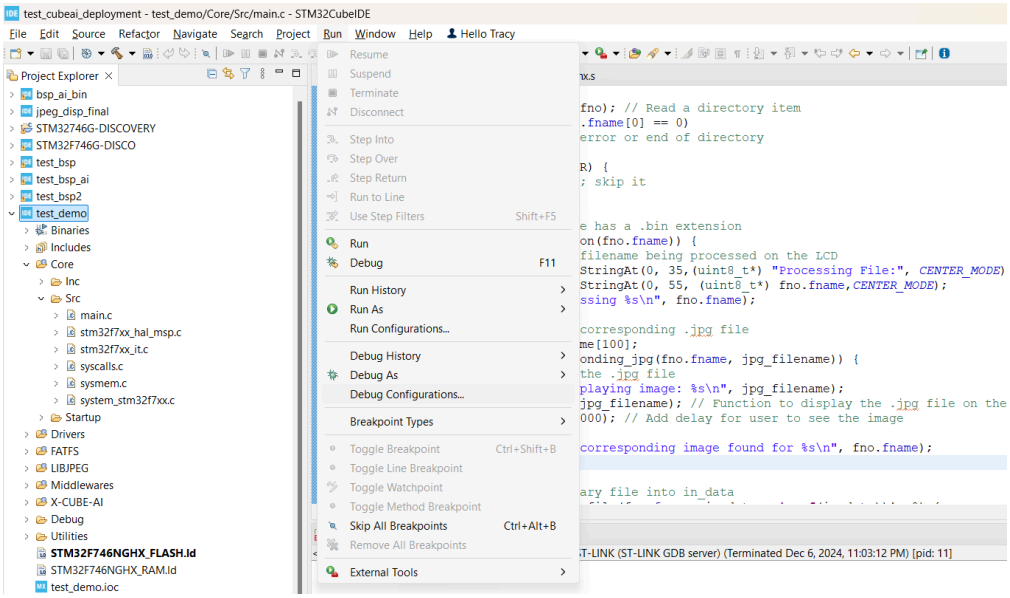
Application Code

After configuring the pins, peripherals, middlewares, and paths, you will copy the application code, build the project, and run it.

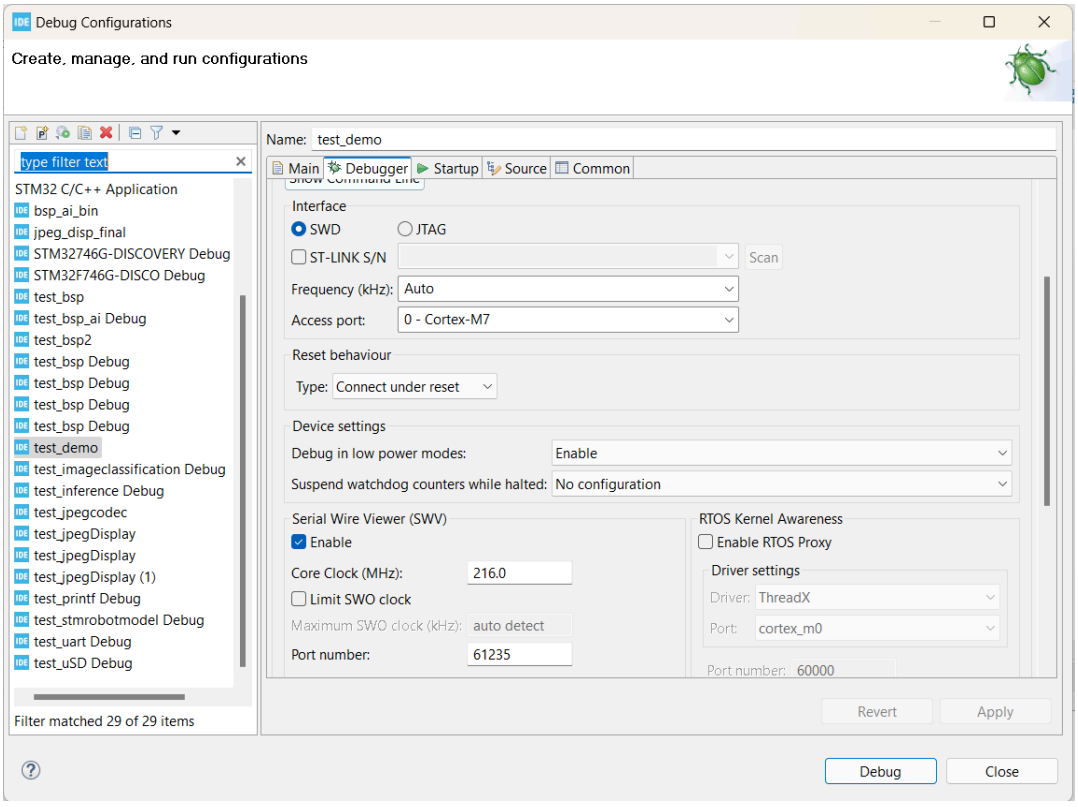
1. Access the repository: [tracyyy19/stm32_nn: binary classfication on stm32f746g discovery board]
2. Copy the main application code to your test_demo project
    a. Navigate to the main.c file in the repository.
    b. Download or copy the contents of the main.c file into your project
    c. Verify if syscall.c is included in the Core > Src folder
        i. If not, copy the syscall.c and paste it in your project Core > Src
3. Build the Project
    a. Click **Project > Build Project**
4. Run the application

a. Connect the STM32F746G Discovery Board to your PC using the USB Type-A to Mini-B cable.

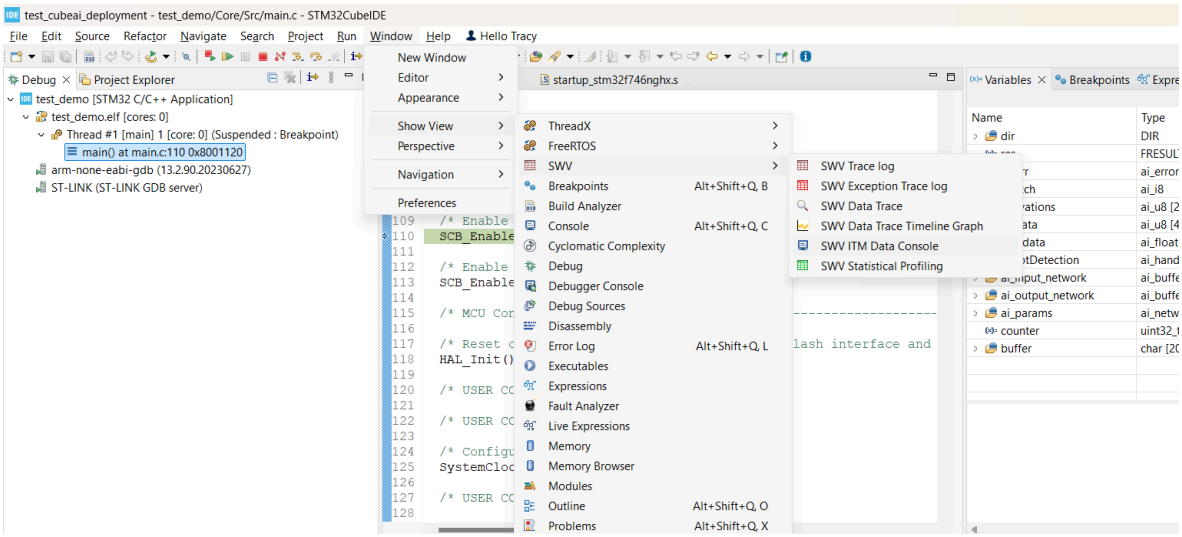
b. Start a debug session, go to Run > Debug Configurations



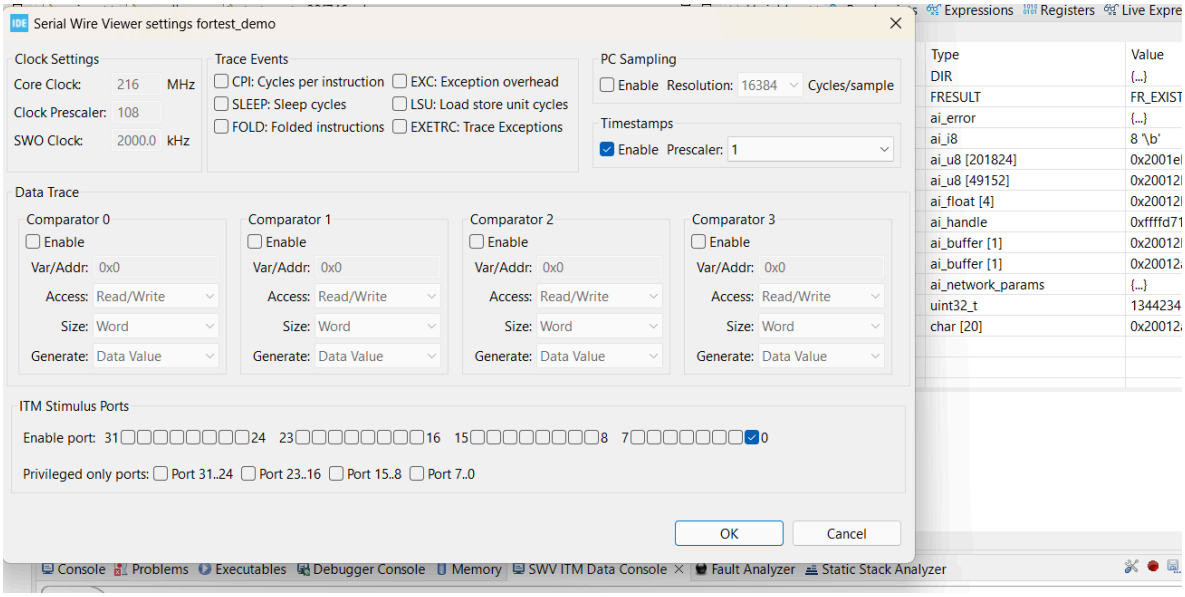
c. To monitor the output, enable the Serial Wire Viewer (SWV) then Debug



d. Show the SWV ITM Data Console. Click Window > Show View > SWV > SWV ITM Data Console



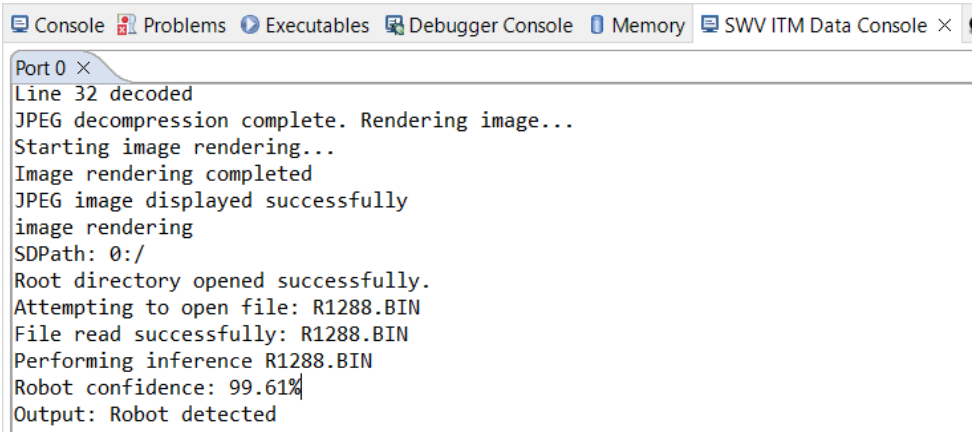
e. Configure Trace by clicking on the  icon and Select Port 0. Then Start Trace by clicking on the  icon.



Results

Test Image : r1288.jpg

Below is a sample output in the SWV ITM Data Console with the test image r1288.jpg



Below is a sample output visible in the microcontroller LCD