Renesas RA Family

e² studio Integration - Getting Start Guide using RA6T2

Introduction

This document covers the development and testing of the Finger Friction demo using the Renesas RA6T2 Flexible Motor Control kit and Reality AI Tools portal. The portal is useful to generate AI models for non-visual sensing applications.

This document contains various sections - section 1 covers the prerequisites, section 2 describes Data Collection, section 3 describes Model Generation, section 4 describes Testing the Models, and section 5 is the Appendix. Sections should be followed sequentially.

Objectives

This document aims to help you gain hands-on experience on developing a simple AI demo using Renesas **RA6T2 motor kit** and **Reality AI Tools**.

Hardware and Software Required

- The following hardware and software is needed:
  - PC with Windows® 11.
  - Project files provided by Renesas - Reality AI team (see details in section 1).
  - Renesas e² studio IDE 2023-07 or newer with FSP 4.6 or newer.
  - Reality AI Tools account

Contents

1. Prerequisites

1.1 Overview

This section covers the prerequisites for running the finger friction demo.

1.2 Procedural Steps

This lab requires the RA6T2 Flexible Motor Control Kit (MCK-RA6T2).

The MCK-RA6T2 kit comes with:

1. RA6T2 board
2. DC brushless motor by Moon's Industries
3. Communications board (not used in this demo)
4. Necessary USB cables

Link: https://www.renesas.com/us/en/products/microcontrollers-microprocessors/ra-cortex-m-mcus/rtk0ema270s00020bj-mck-ra6t2-renesas-flexible-motor-control-kit-ra6t2-mcu-group

Additional components:

1. 36 V power supply (recommended but not required Amazon link)
2. USB to UART cable (Amazon link) - required (or alternate)

Assemble the kit as shown in the next section.

1.3 Kit Assembly

Assemble the kit as shown below. Make sure that the toggle switch is in the OFF position and all other jumpers are in their default condition.

Note: See the documentation that came with the kit if there is concern that the board may not be in the default configuration.

To assemble the kit:

1. Connect the PC to USB-C on the board
2. Connect the external power supply (optional).
3. Connect the USB to UART (CN10) (see the following table for UART to USB pin connections)
4. Connect the motor to CN2.

Pin reference

Pin[4]-> VCC (no connection)

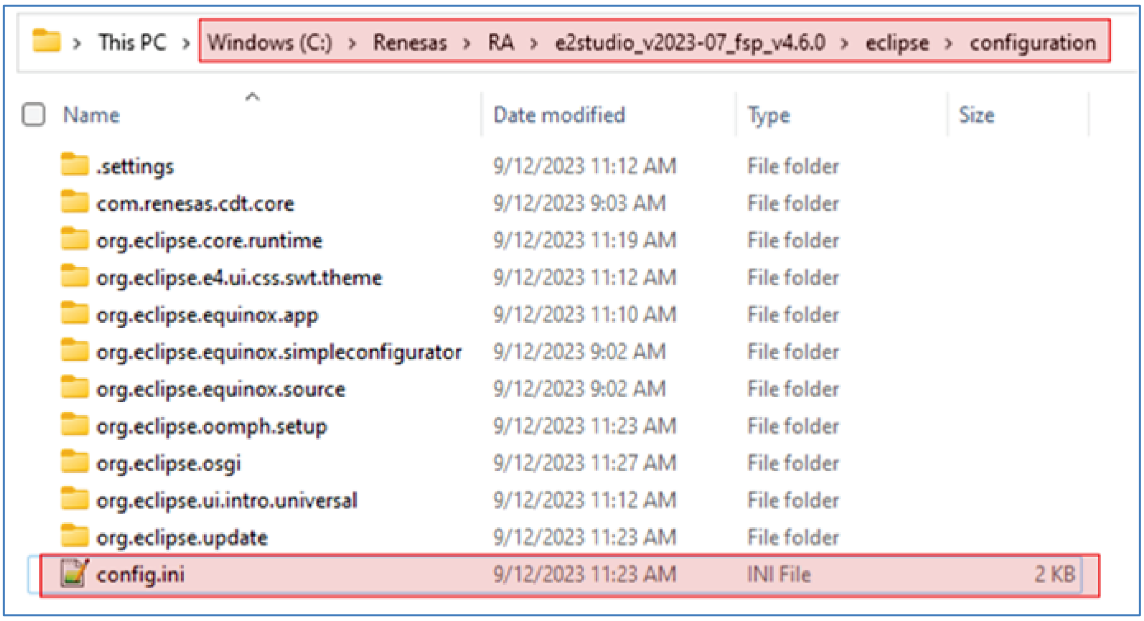
Pin[3]-> TX to RX (Green wire)

Pin[2]->RX to TX (White wire)

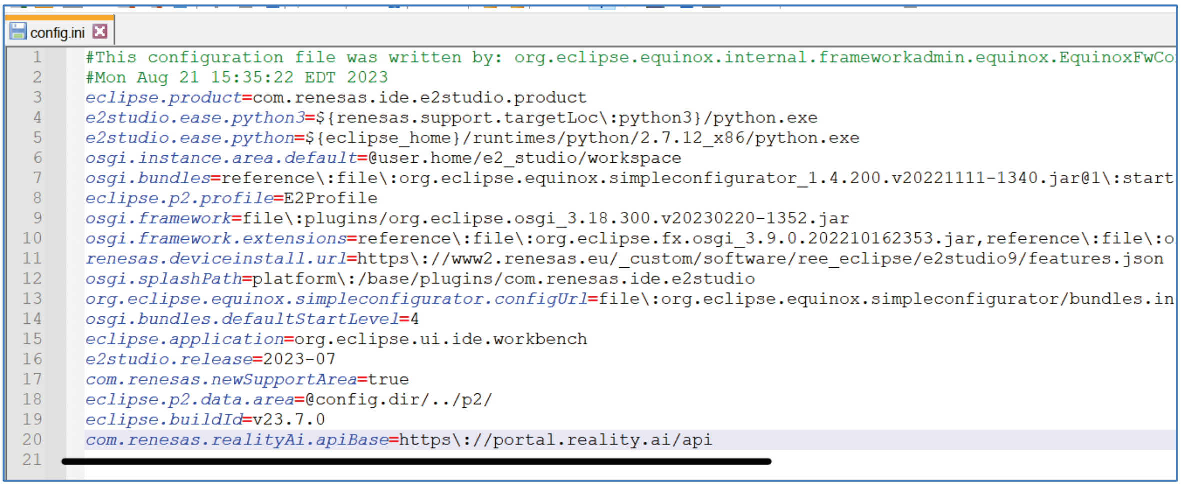
Pin[1]->GND to GND (Black wire)

1*3P Female Socket	Name	Colour	Description
Pin 1	GND	Black	Device ground supply
Pin 2	TXD	White	Transmit Asynchronous Data
Pin 3	RXD	Green	Receive Asynchronous Data

5. This lab requires Renesas e² studio IDE 2023-07 or newer with FSP 4.6 or newer.
   Platform installer is available here: Link Take note of where e² studio is installed.
6. This lab requires modification to the eclipse configuration file.
   Navigate to e² studio installation directory (install folder) > **e2_studio** > **eclipse** > **configuration** > open file: **config.ini** (installation folder may be different)

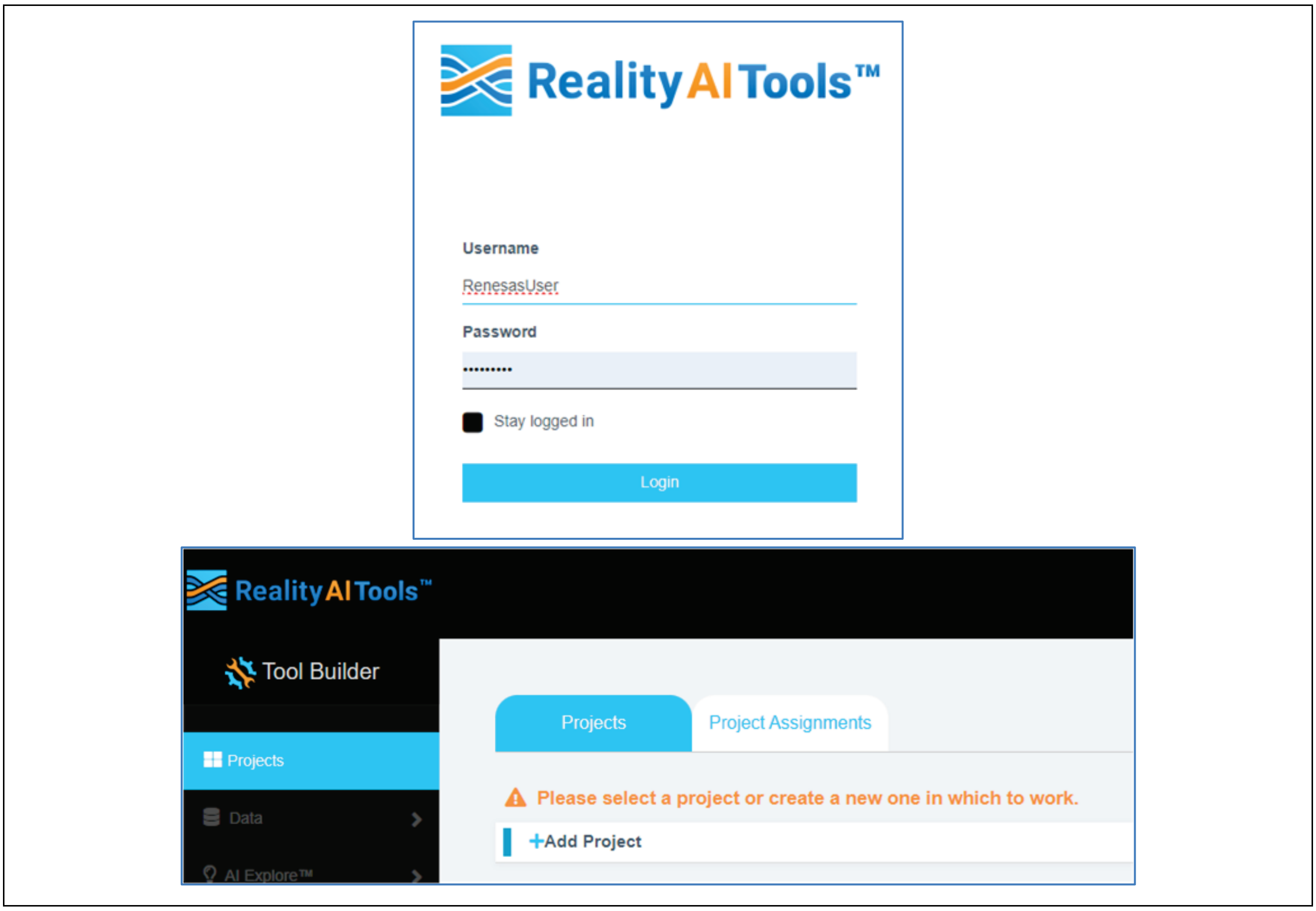


Copy the following line:
`com.renesas.realityAi.apiBase=https\://portal.reality.ai/api`
And paste it at the end of the file:



This line ensures that e² studio will connect to Reality AI server for data upload and model training. Be sure to save the config file before closing. **Restart e² studio if it is currently running**.

7. This lab requires an account on the Reality AI website.
   Login to Reality Tools (https://portal.reality.ai/login) using the username and password provided. Reach out to Renesas if you need credentials + instruction documents for the portal.



   Leave the browser open in the background.

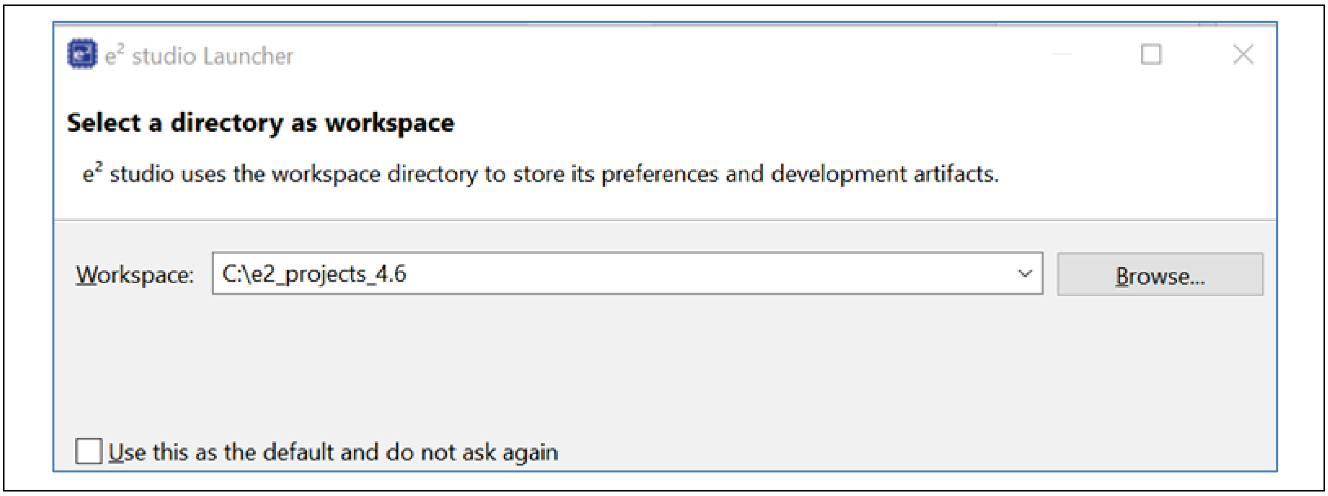
2. Data Collection

2.1 Overview

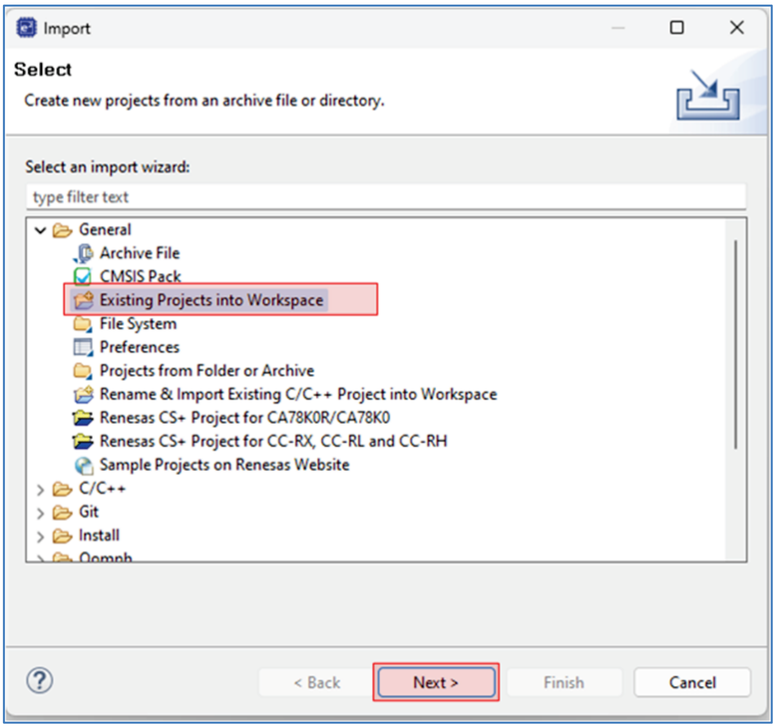
This section covers collecting data with the Reality AI eclipse plugin using e^2 studio.

2.2 Procedural Steps

1. Open e^2 studio IDE and select a workspace. Although the workspace can be in any folder, this lab assumes the workspace as shown in the following graphic, `c:\e2_projects_4.6`

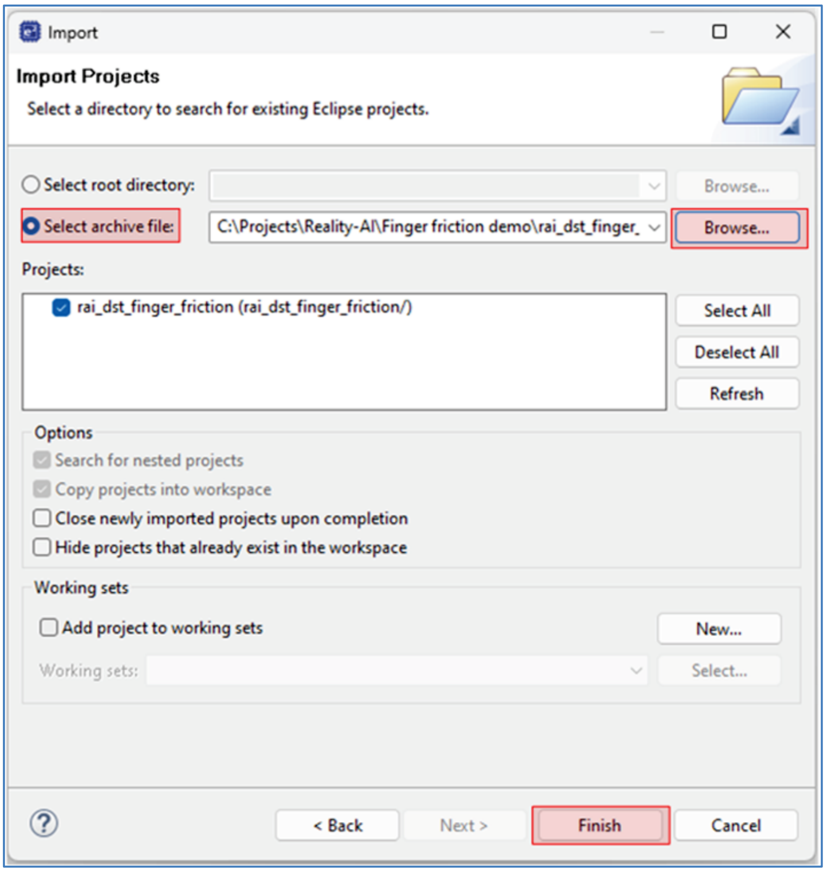


2. Go to **File** > **Import**... Then choose **General** > **Existing Projects into Workspace**



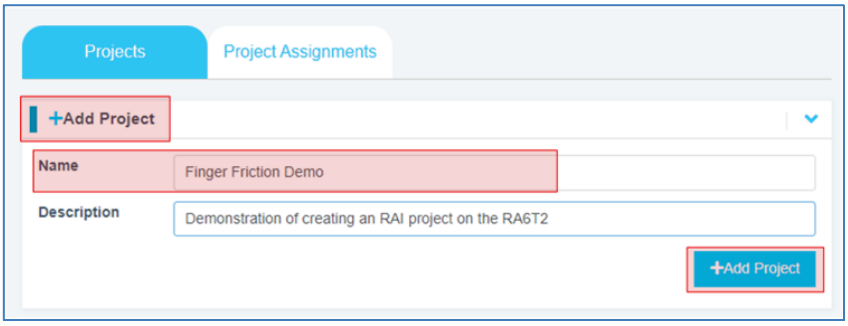
Click **Next** to continue.

3. Check **Select archive file** and navigate to the zip file for this lab. This lab contains one project. Make sure it is selected and then click **Finish**.

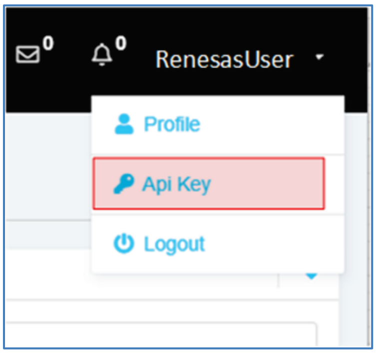


The demo project is imported into the workspace.

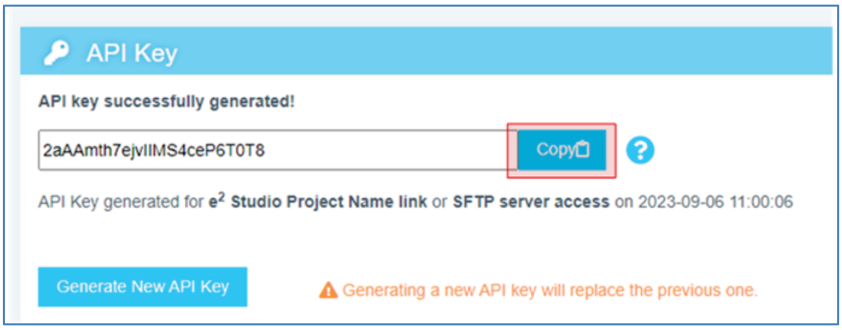
4. Return to the Reality AI tool and start a new project. Click on **+Add Project**, give the project a name and, optionally, a description. Then click the **+Add Project** button in the lower right.



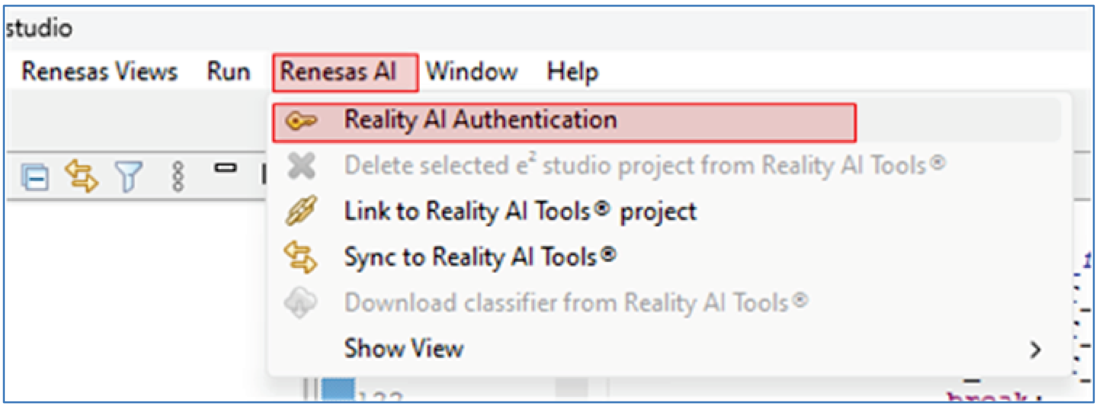
5. Navigate to the username on the top-right of the screen and select **API Key** option.



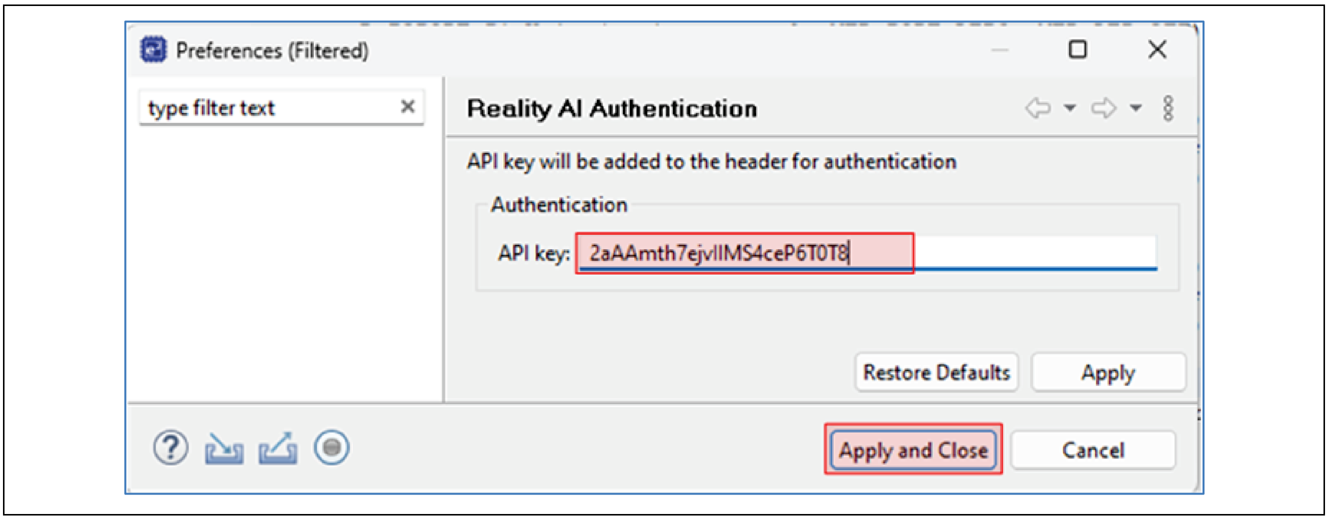
6. Copy this API key to the clipboard. This key will be used to connect e² studio IDE and Reality AI Tools.



7. Open e² studio and navigate to **Renesas AI** > **Reality AI Authentication**
   Note: If **Renesas AI** does not appear in the menu, consult the appendix.

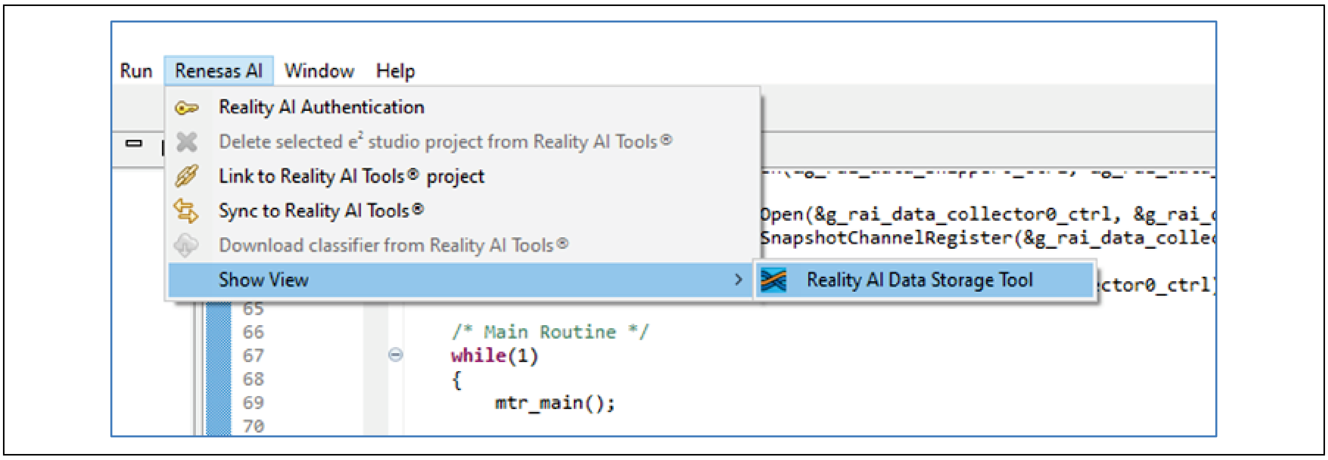


Paste the API key and click **Apply and Close.**

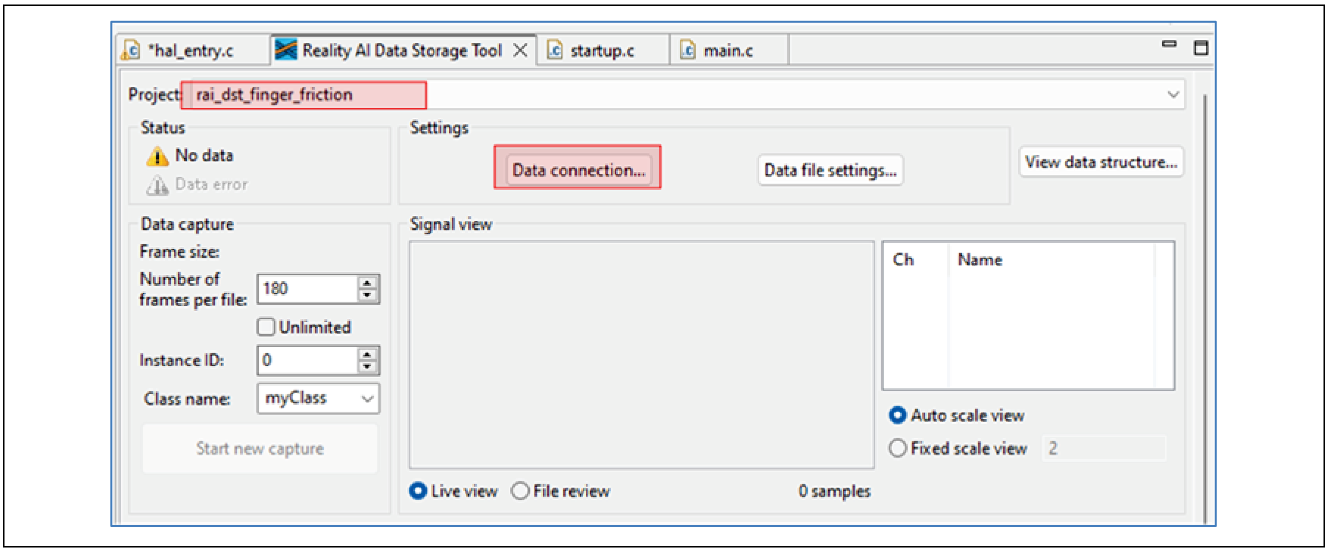


The IDE can now connect to the Reality AI site.

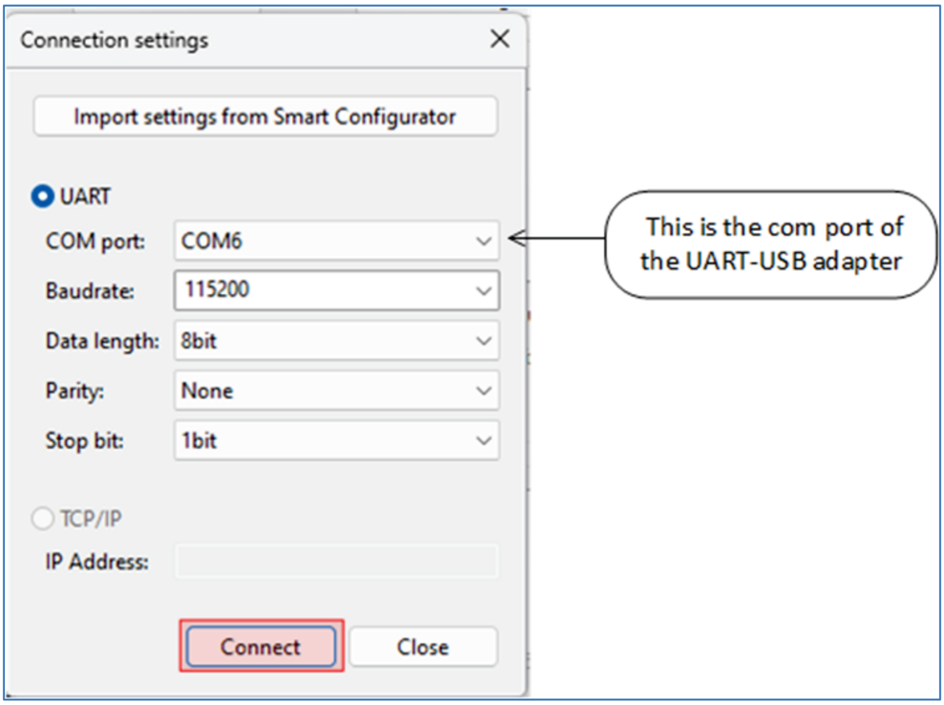
8. Navigate to **Renesas AI** > **Show View** > **Reality AI Data Storage Tool.**



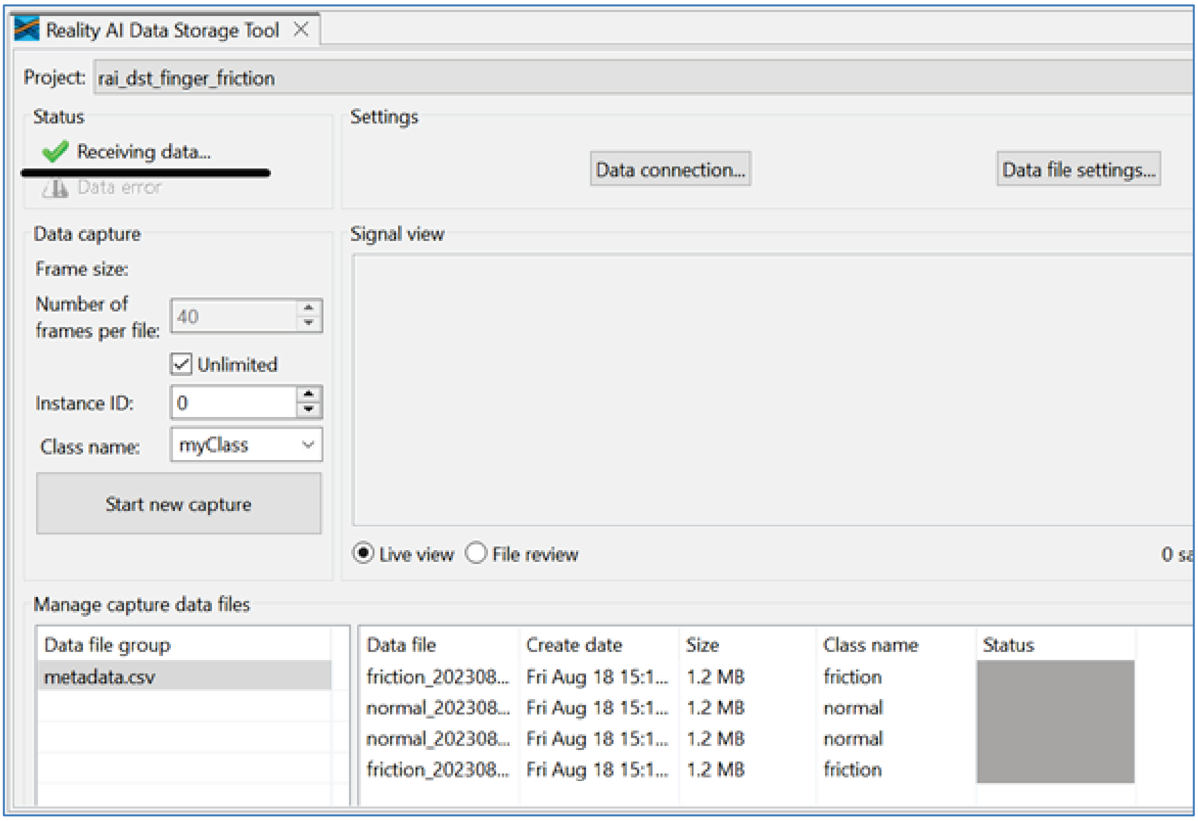
A new window will open in the bottom pane. Click-and-drag the view to the main view area by clicking on the tab and drag-drop in the tab area of the main view. Select the project in the pulldown, then click on **Data Connection** button.



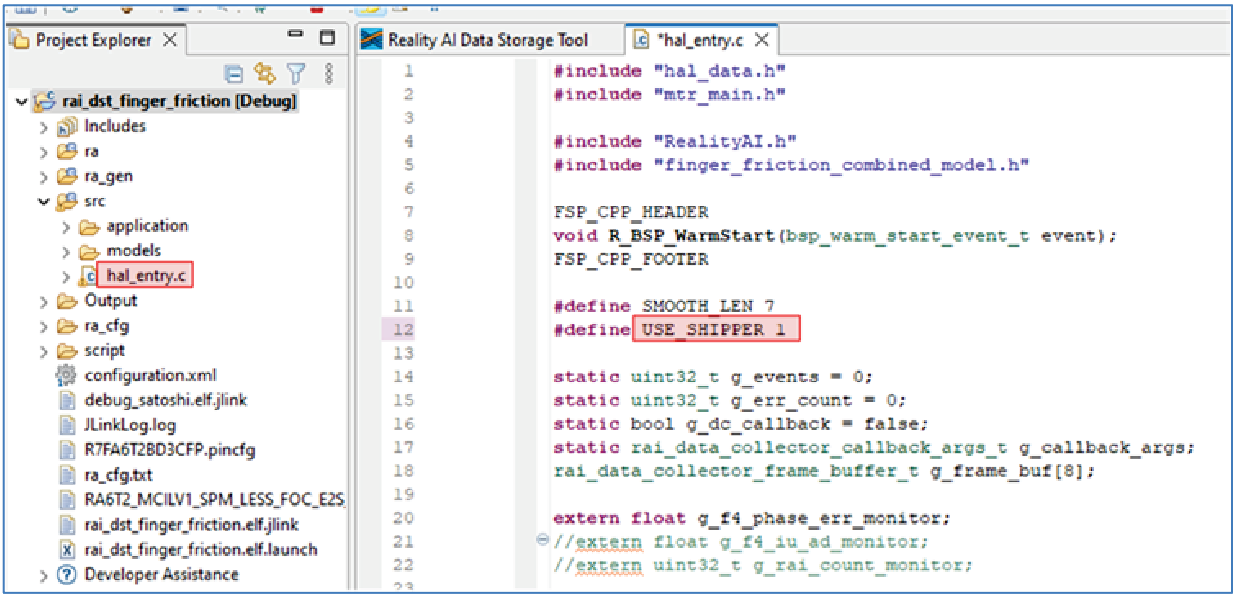
9. In the menu that pops up, select the COM port, baudrate (115200), and communication protocol parameters (8-N-1). Then press **Connect** > then press **Close**.



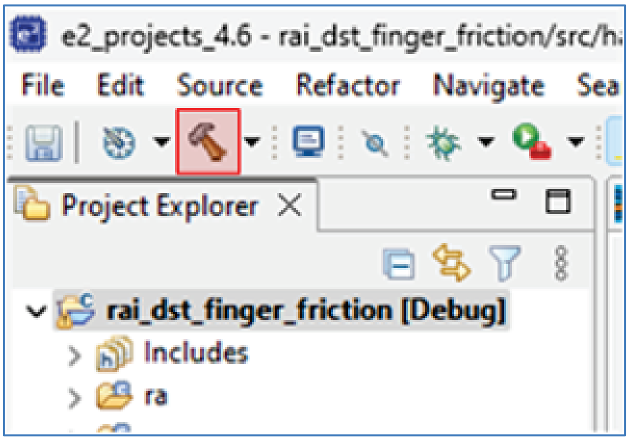
10. Once connected, the status (top left) should show “**Receiving data...**” with a check mark.
Note: The green check mark indicates that the connection is successful. Not that it is receiving data yet.



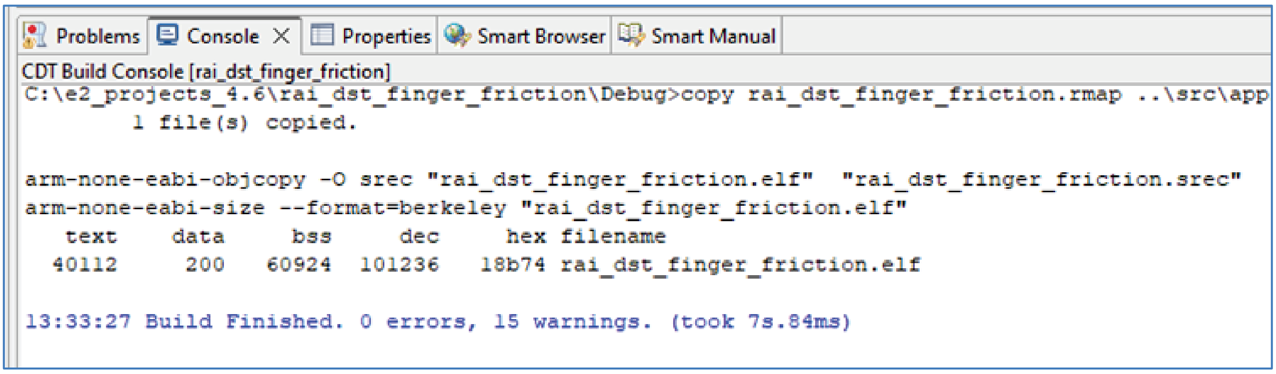
11. Expand the project in the Project Explorer view and open `hal_entry.c`. Verify/change USE_SHIPPER to be defined as 1. This will build the project in the data collection mode.



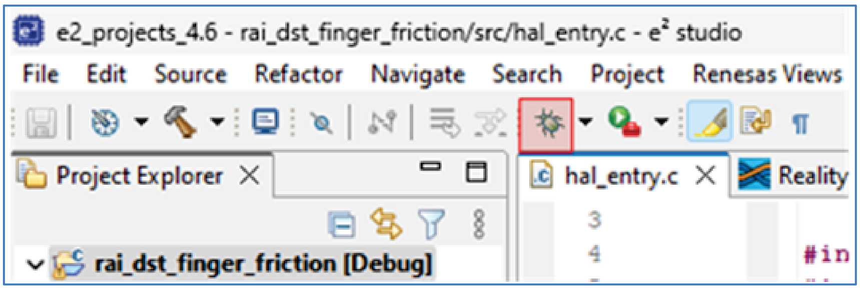
Save the file and build the project by clicking the hammer icon



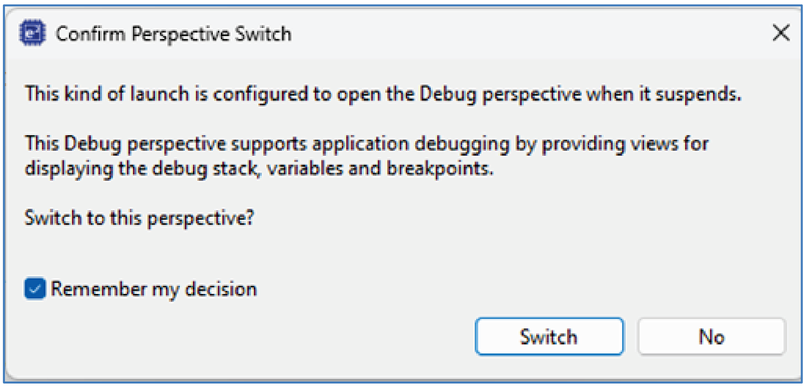
The project should build without error. There will be some warnings, these are expected.



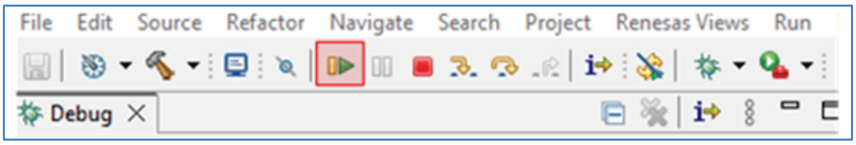
12. Make sure the toggle switch is in the OFF position. Start a debug session by clicking the **Debug** icon in the toolbar



Click **Allow access** if you get a warning from Windows Defender. Check the **Remember my decision** check box and click **Switch** if the following dialog appears.



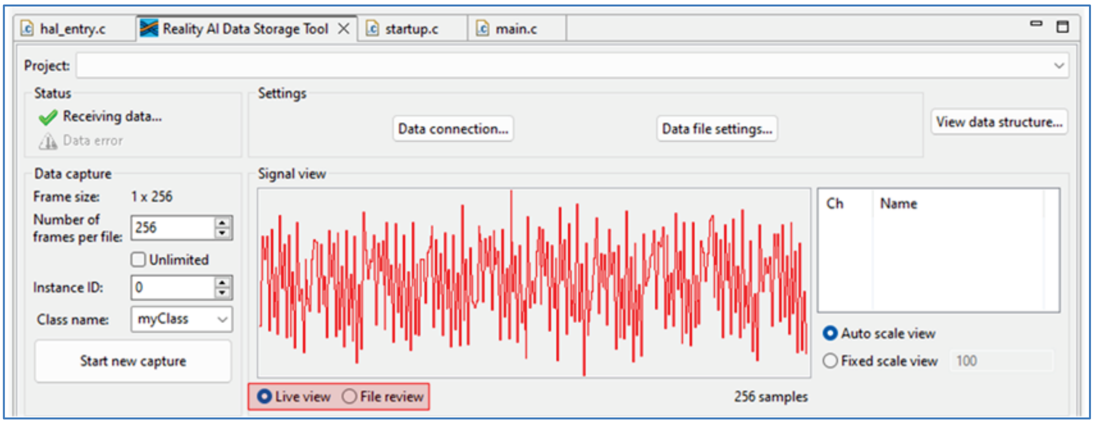
13. Click the **Resume** button twice. The program is now running.



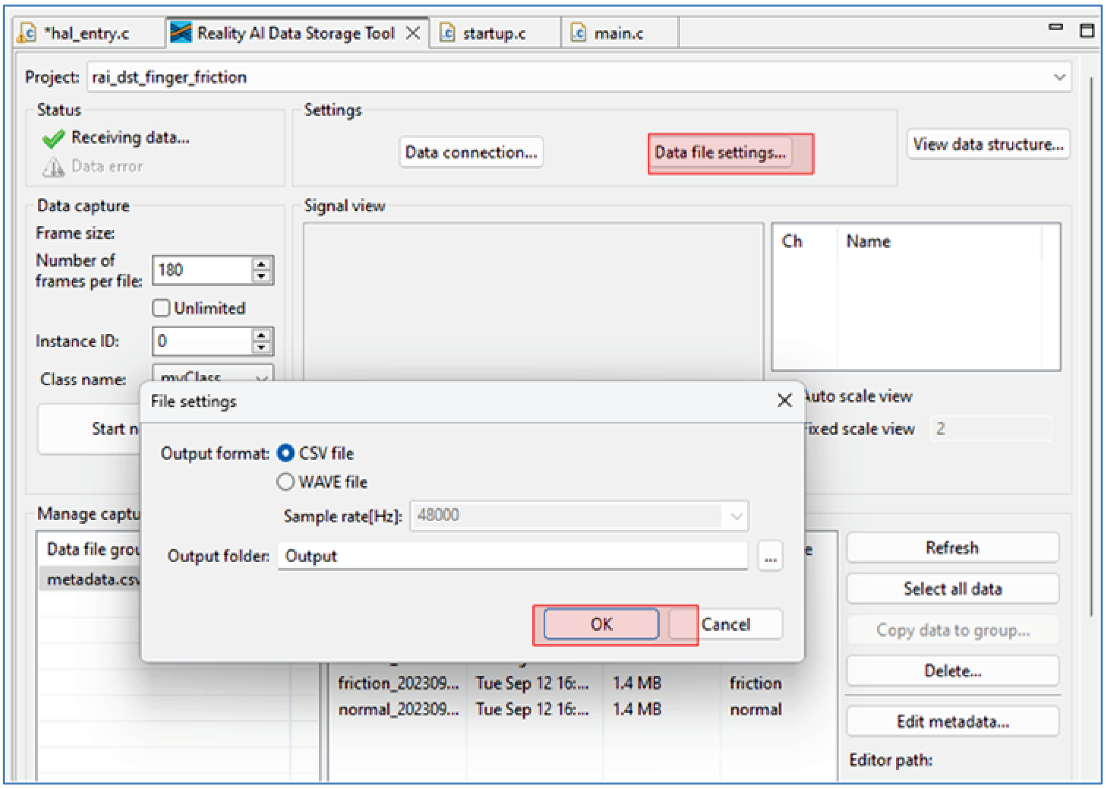
Switch the toggle switch to the ON position. The motor should now be running.

WARNING:	Never stop the debugger while the motor is running. This will cause commutation to stop, possibly leaving a coil energized. This will cause the motor to get very hot. If the motor is not spinning at this stage check the connections and review the material above.

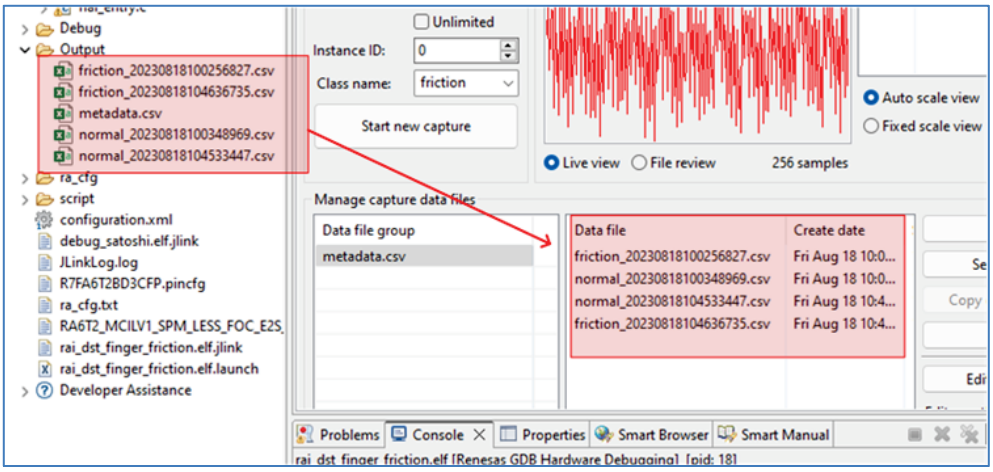
14. Click on the **Reality AI Data Storage Tool** view and verify the presence of a wave form in the Signal view. If there is nothing showing, click **File review** and then **Live view** (highlighted below). If there is still no waveform, recheck the prerequisite steps to make sure the connections are correct.



15. Click on **Data file settings…** and verify **CVS file** and **Output** folder. The Output folder is relative to the project. Click **OK** to close the dialog.

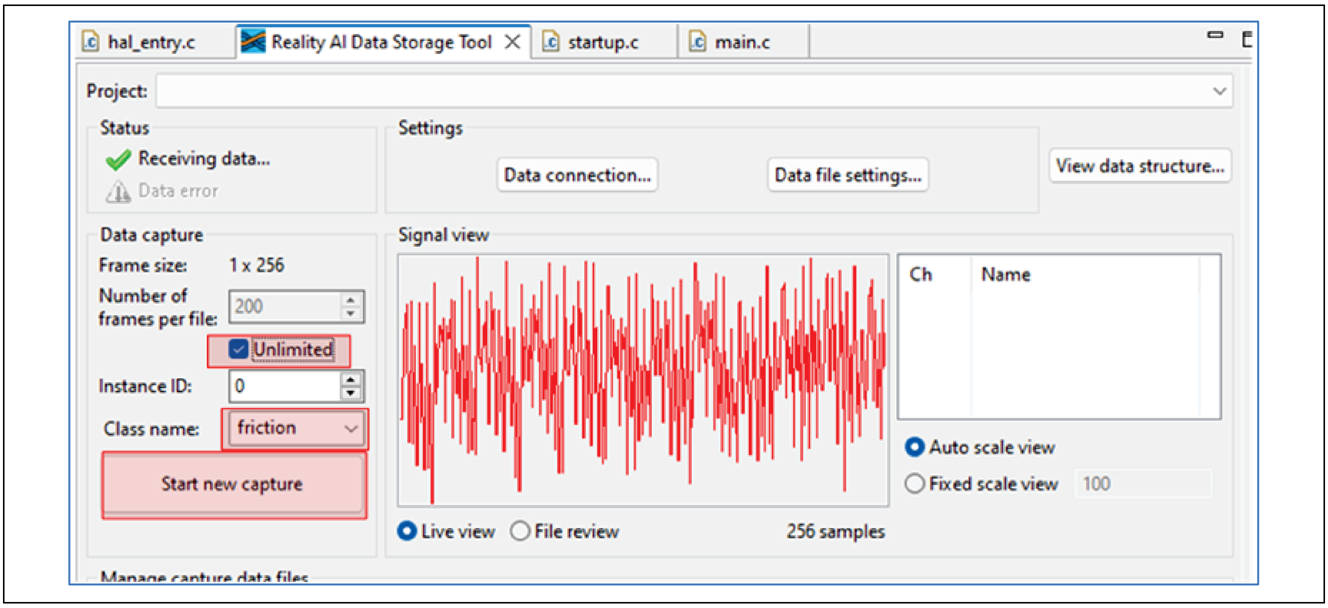


16. Note that the Data Storage tool reflects the data in the project's Output folder. The data currently there was collected during the lab development and includes date and time information. New data will be collected in the following sections.

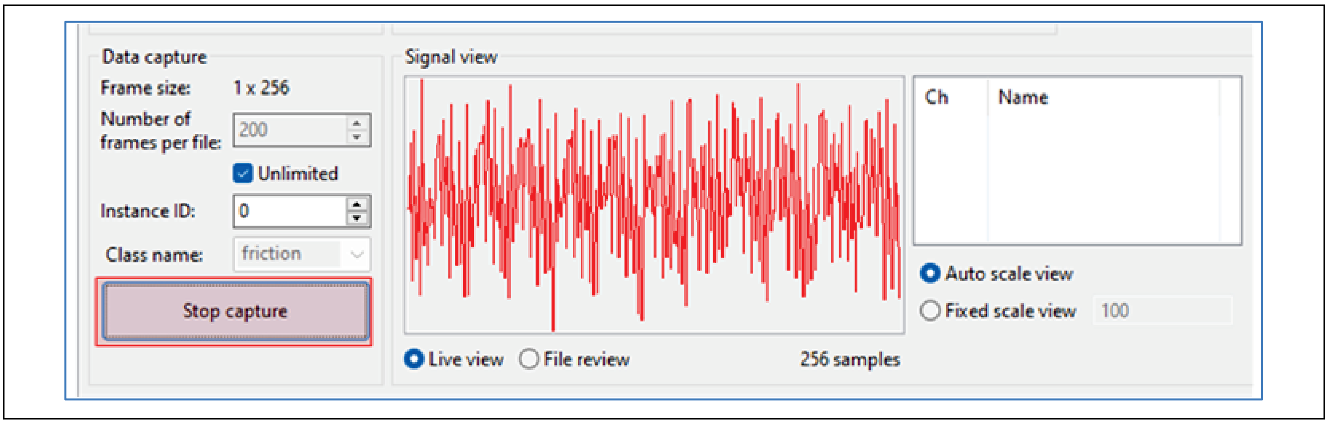


Note: There are two methods to collect data: **Number of frames per file** and **Unlimited**. If the user selects **Unlimited** (by checking the **Unlimited** checkbox) data collection will occur until the user manually stops the collection process. Otherwise (**Unlimited** box unchecked) the collection will automatically stop once the **Number of frames per file** is reached. For this project, setting the **Number of frames per file** to 200 results in approximately 52 seconds of data. The remainder of this lab assumes **Unlimited** capture.

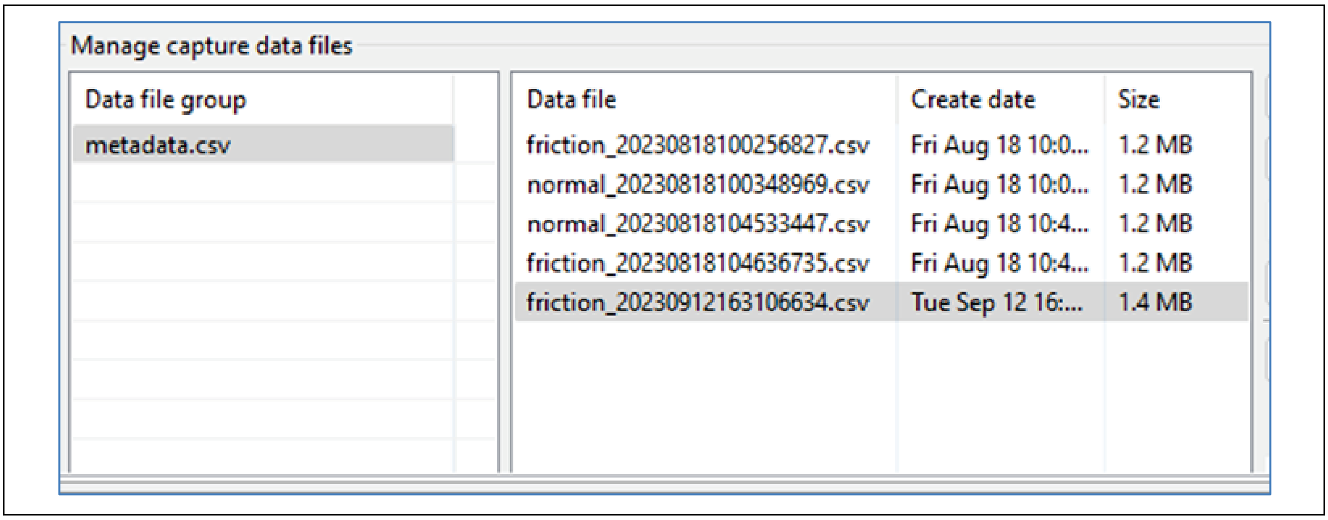
17. In the **Reality AI Data Storage Tool**, check the **Unlimited** checkbox. Enter the Class name **friction**. While applying and maintaining slight friction to the motor shaft, click the **Start new capture** button. The button text will change to **Stop Capture**. Maintain friction for 45 to 60 seconds and then click the **Stop Capture** button.



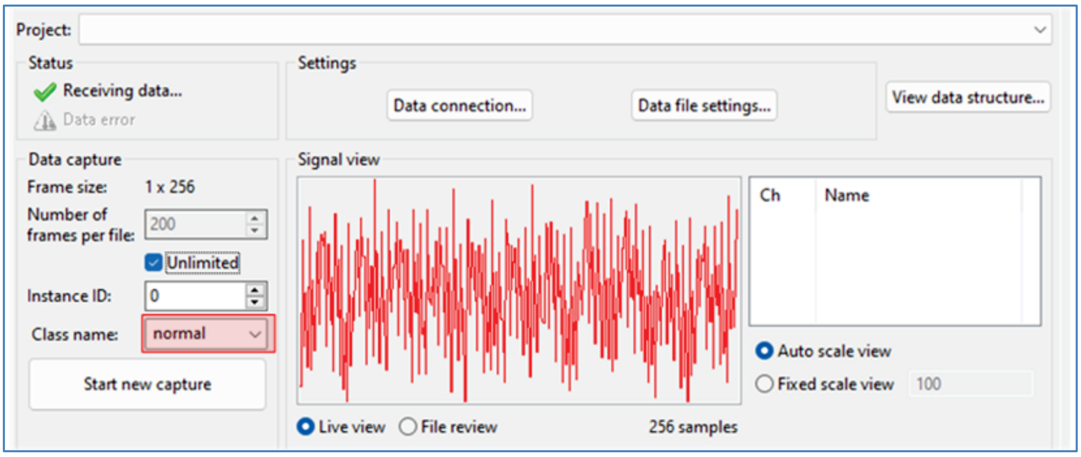
45 to 60 second's elapses...



A new data file appears in the output folder

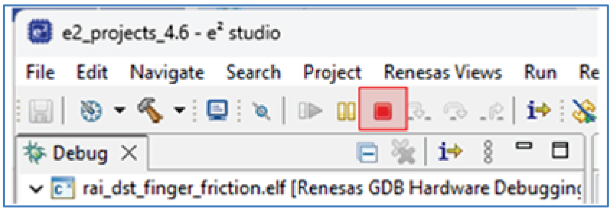


18. Change the Class name to **normal** and repeat the above steps **without applying friction**.



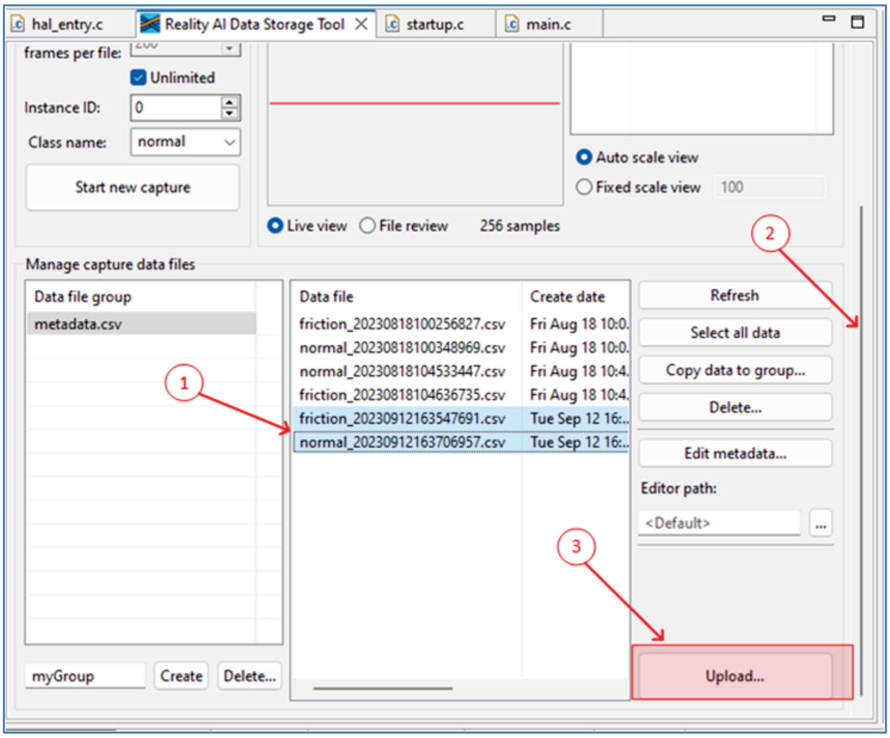
19. This concludes the data capture portion.
    Turn the toggle switch off to stop the motor.

Click the red square icon to terminate the debug session.

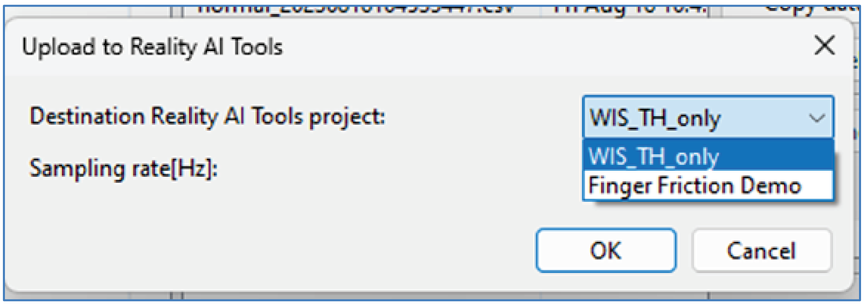


(If you do not see the terminate icon, select the Debug perspective in the upper right corner)

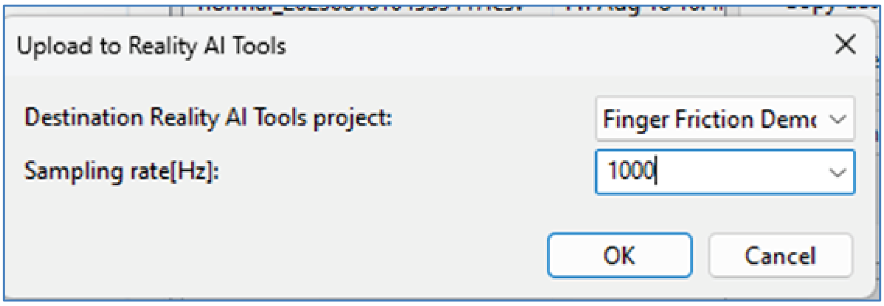
20. Follow these steps to upload the data files collected in the last section:
    1. Select the two files using click + ctrl-click
    2. Scroll the view to expose the **Upload…** button
    3. Click the **Upload** button



21. A dialog box should appear with the projects currently defined on the Reality AI website.
(if there are no projects in the pulldown, verify the changes made to the config.ini file, regenerate an API key and restart e² studios)



Select the Finger Friction Demo (or whatever the project was named in the prior step)
Set the Sampling rate to 1000 and press **OK**



22. The data files, along with the metadata.csv file will upload, and a confirmation dialog should appear.

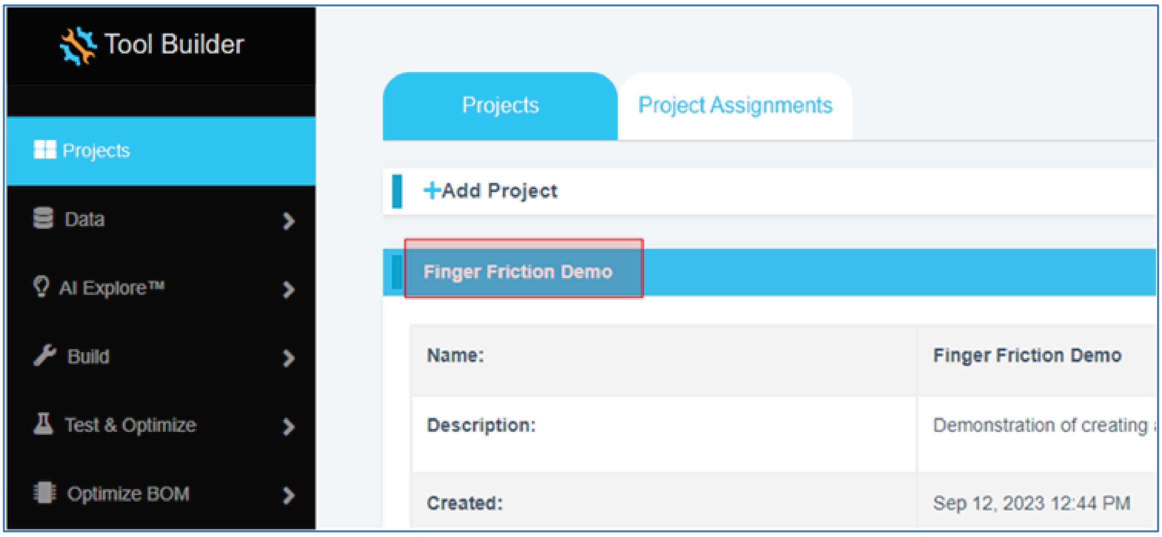
3. Creating Models

3.1 Overview

This section covers creating AI models on the collected dataset. A Reality AI Tools account is required.

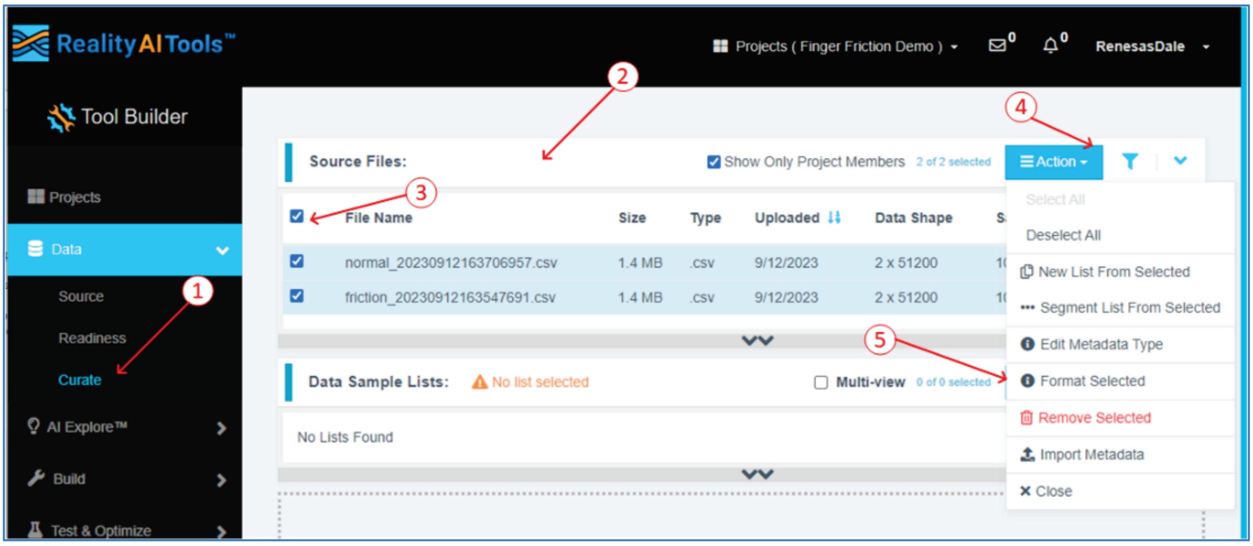
3.2 Procedural Steps

1. If not already logged in to Reaity AI Tools, use this link: https://portal.reality.ai/login with the provided password. Reach out to Renesas – Reality AI team members for credentials + instructions documents for the portal if needed.
2. Format the data. Click on the Finger Friction Demo project to set active.



Click on the **Data** tab on the left to expand and follow these steps:

1. Click on **Curate**.
2. Click on **Source Files** to expand.
3. Check box next to **File Name** to select all files.
4. Click on the **Action** button.
5. Click on **Format Selected**

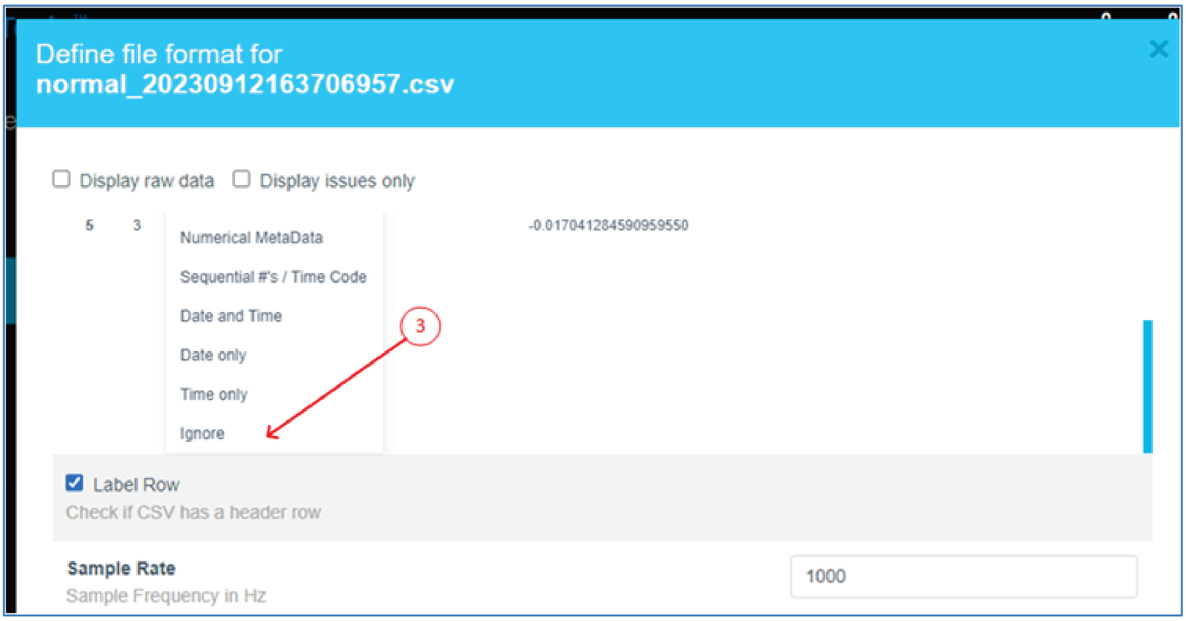


This opens a dialog box.

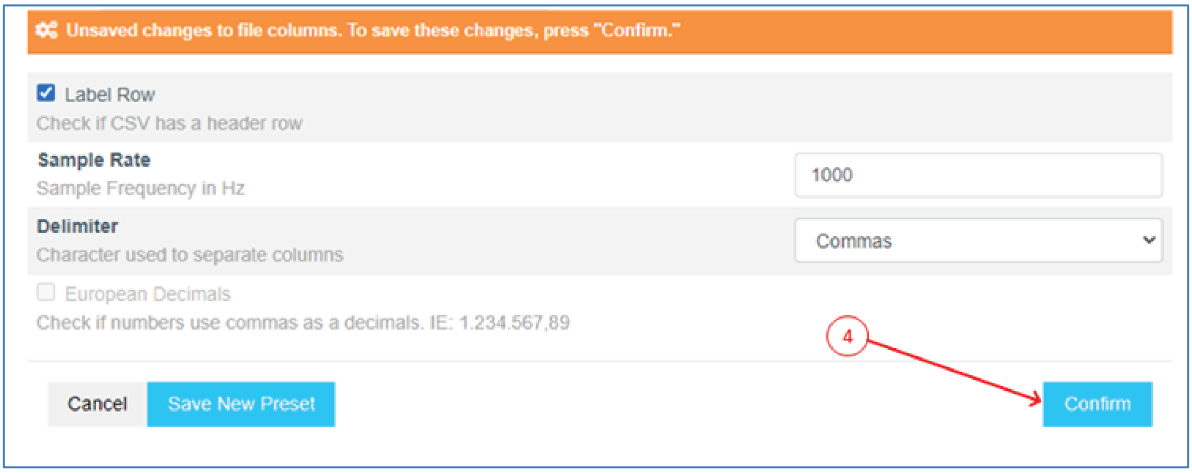
Follow these steps:

1. Click on **Data** to expand the drop down
2. Use the scroll bar to scroll down (or mouse wheel)

3. Select **Ignore** (this column is not part of the data)



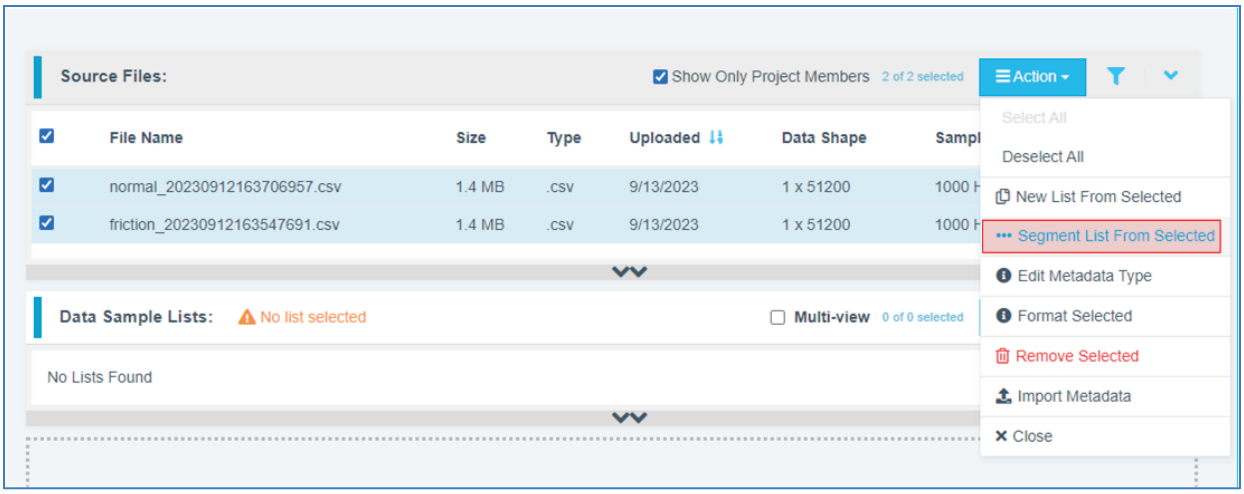
4. Click **confirm**.



3. Verify that the data shape is now 1 X {num of samples} and the Sample rate is 1000 Hz.

Data Shape	Sample Rate
1 x 51200	1000 Hz
1 x 51200	1000 Hz

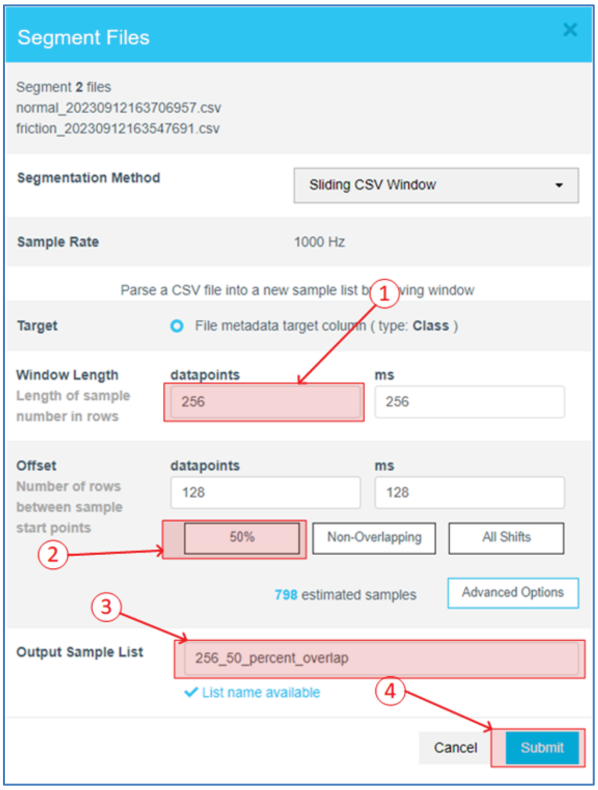
4. Preprocess/segment the data files. Click the Action button and select *** **Segment List from Selected**



Why Segmentation?

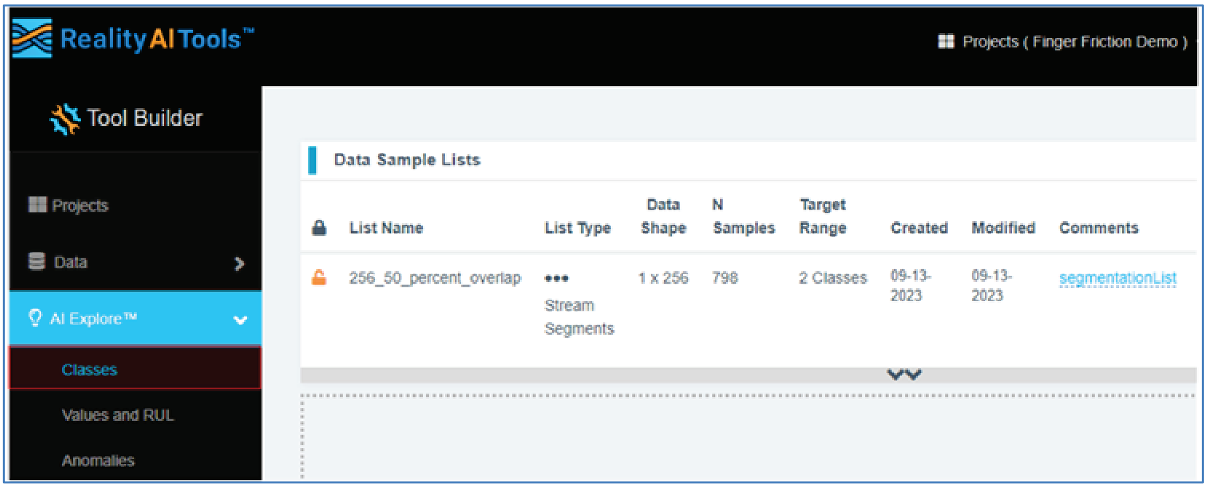
One of the main purposes of generating models from Tools is to deploy them to a variety of Renesas MCUs. To that effect, these models must process live data within a resource constrained environment. So, for practical purposes, a model might be looking at 1 second, 500 ms, or even a smaller window length to make quick predictions on Realtime data. As opposed to a few seconds (or minutes or hours) long data stream. To mimic that effect, we break down the raw training data and feed that to the model generation engine to start learning what it is going to see in a live (production) setting.

5. Assign window length, overlap, and provide name to the list using the following steps:
    1. Set **Window Length** to 256.
    2. Click the **50%** button to select a 50% offset
    3. Give the list a meaningful name
    4. Click **submit**



The number of estimated samples will be dependent on the amount of data collected. So, if your estimated samples do not match the following screenshot, it is still okay.
Window Length: The window length determines how much data will be considered by the AI to decide on a classification.
Offset: The offset determines how far ahead from the start of the last window the parser moves before creating a new window.
General Guide: 50% Overlap is usually a good compromise between covering starting-point variations in the data and too much redundancy.
Use **non-overlapping** windows when you have a great deal of data, with longer offsets. Typically, users will do initial exploration and training on a subset of the available data.
Use **All Shifts** (offset = 1) for testing after you have a suitable candidate classifier, and you want to simulate performance on a stream of new data arbitrarily sampled.

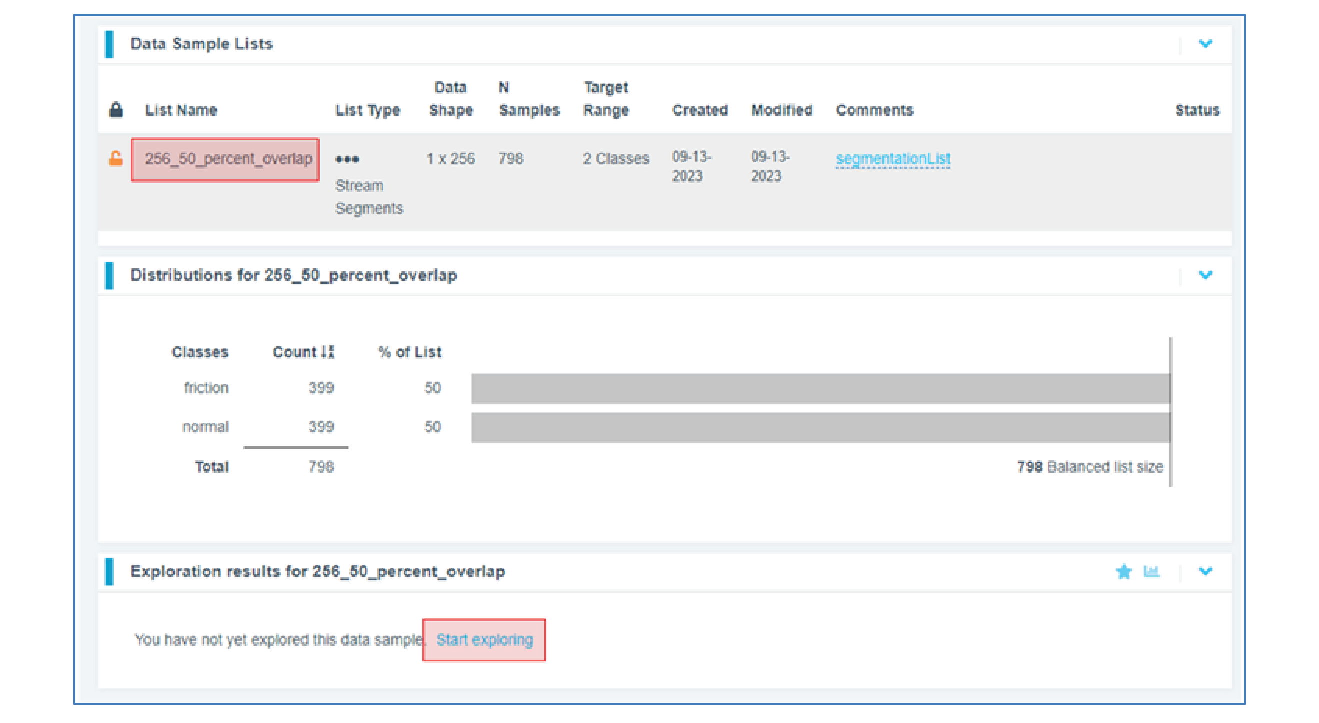
6. After successfully creating the segmented list, navigate to **AI Explore** > **Classes**.



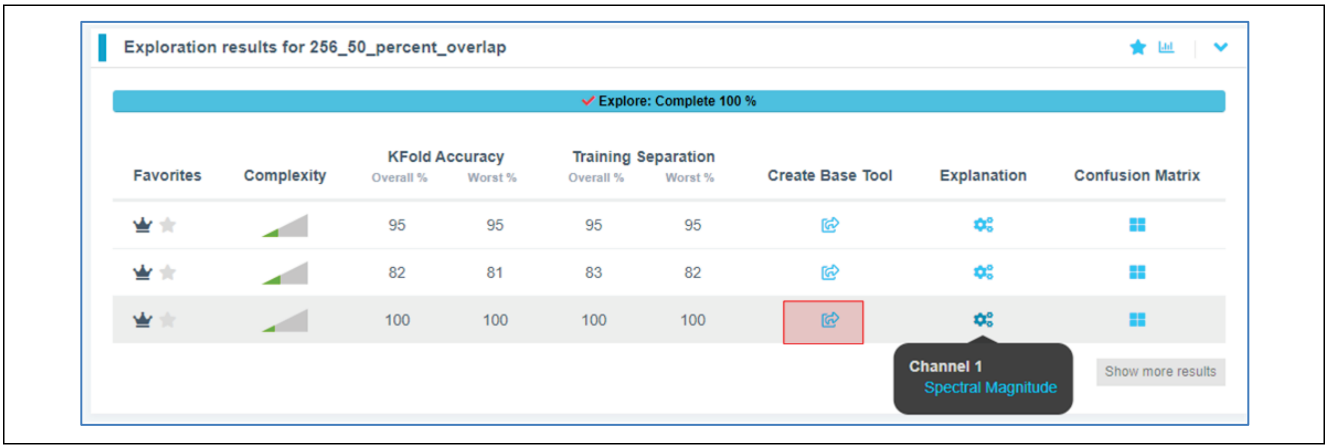
Reality AI Tools has options for creating 3 types of AI Models:

Classes: When there is labeled categorical data (**option used in the tutorial**). You might have noticed that we uploaded perfectly labeled data in section 1. This is needed for classification models as supervised learning is being performed.
Values: When discrete int or float values are used instead of categories. (Example: What is the exact temperature of a machine? Or What is the exact value of tire pressure of a car?). This is also supervised learning.
Anomalies: This is an Anomaly Detection module. It is a semi-supervised model where the user only needs examples of Normal data to create a baseline model.

7. Click on the newly created list and then click on **Start Exploring** to start the feature discovery and model training process.



8. Once the explore finishes, select the highest performing model (we recommend model with feature space: Spectral Magnitude) and click on the **Create Base Tool** button. Hover over each **Explanation** to get more information about the model.



What is happening in the background:

Clicking on the Start Explore button will turn the Reality AI algorithm loose on your data. It will create several optimized feature sets and machine learning models that best fit the classification problem and then summarize each model's accuracy and resource consumption.

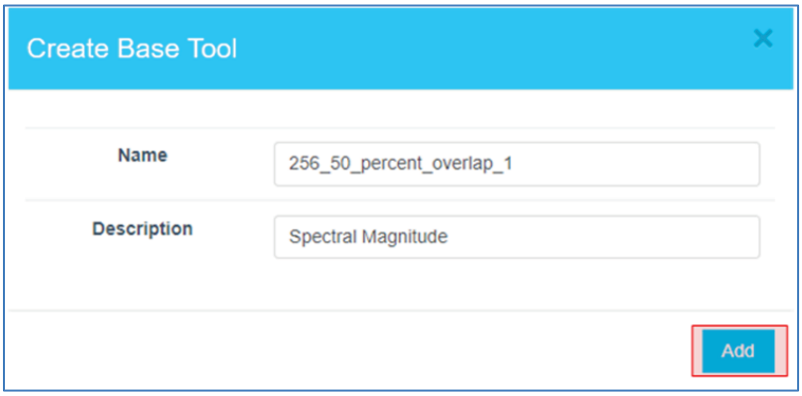
What is the algorithm doing: AI Explore first creates a balanced subset of the sample list (meaning a list in which each class has equal numbers). If the sample list is exceptionally large, it may also sub-sample for a shorter processing time. Explore then runs through a procedure in which the Reality AI algorithms attempt to discover the best possible set of features and machine learning parameters for separating the different training classes represented in the data. The feature sets that are most promising are then used to construct machine learning models, which are trained on the sub-sample and put through a K-Fold validation.

Only the best performing results are displayed. Several hundred different feature sets and machine learning models are compared in a typical AI Explore run.

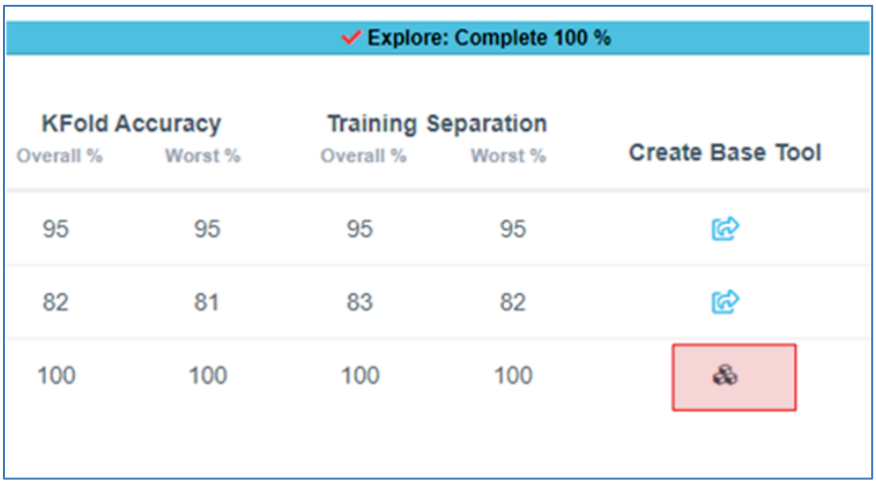
What is K-Fold validation:

K-Fold is where a given data set is split into a K number of sections/folds where each fold is used as a testing set at some point. Let us take the scenario of 10-Fold cross validation(K=10). Here, the data set is split into 10 folds. In the first iteration, the first fold is used to test the model and the rest are used to train the model. In the second iteration, the 2nd fold is used as the testing set while the rest serves as the training set. This process is repeated until each fold of the 10 folds has been used as the testing set. **We use K=10 in the AI Explore page**.

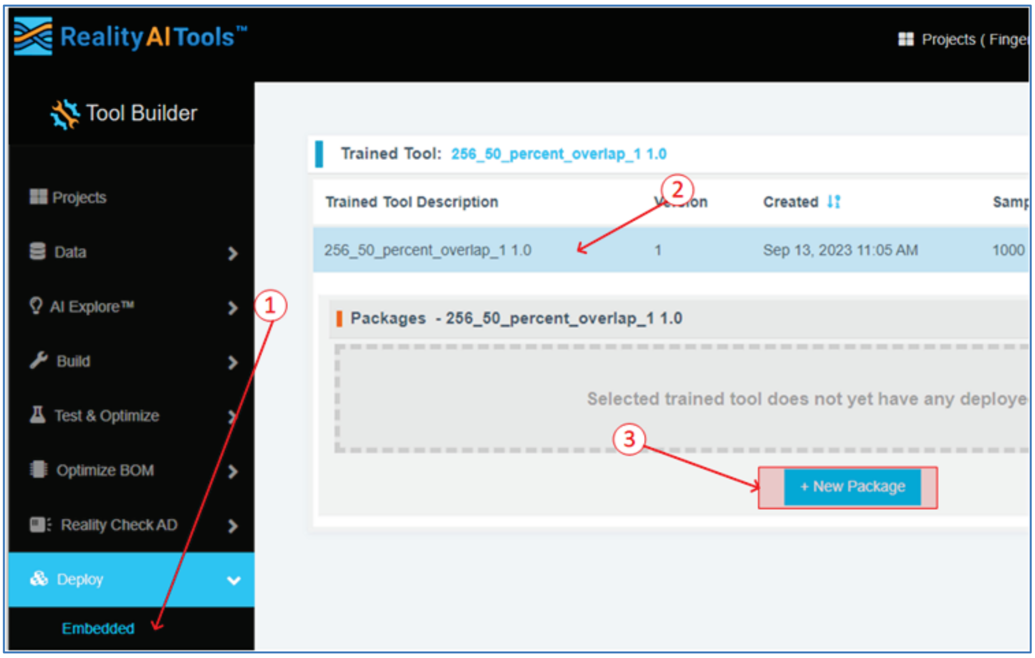
9. Provide a name or use the one suggested. Click **Add**.



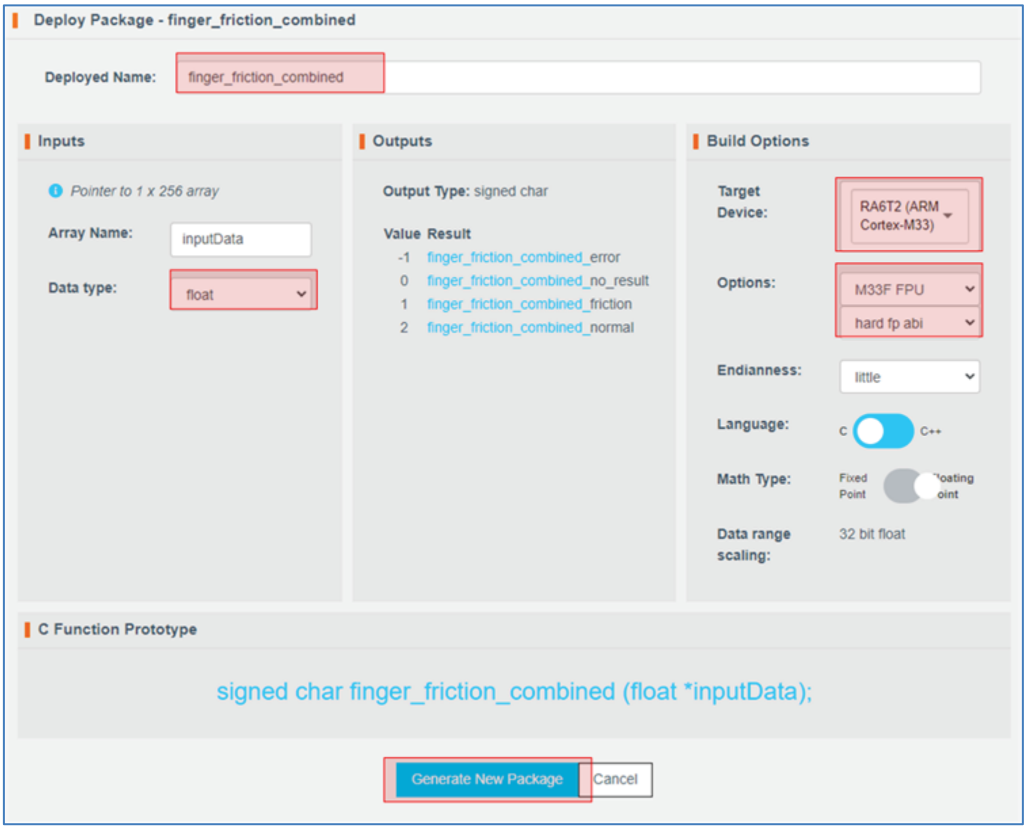
The icon will change, indicating the base tool has been created.



10. Now that the model is ready to be deployed, Follow these steps to produce a new package.
    1. Click on **Deploy** > **Embedded**.
    2. Click on the **Trained Tool Description** list
    3. Click on **+ New Package**

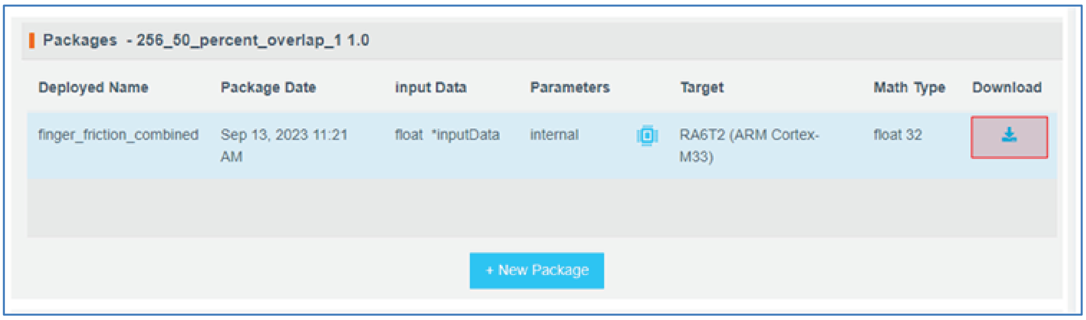


11. Set the options as indicated. The deployed name should be **finger_friction_combined** to match the code in e² studio. Choosing another name will require additional edits to the code.

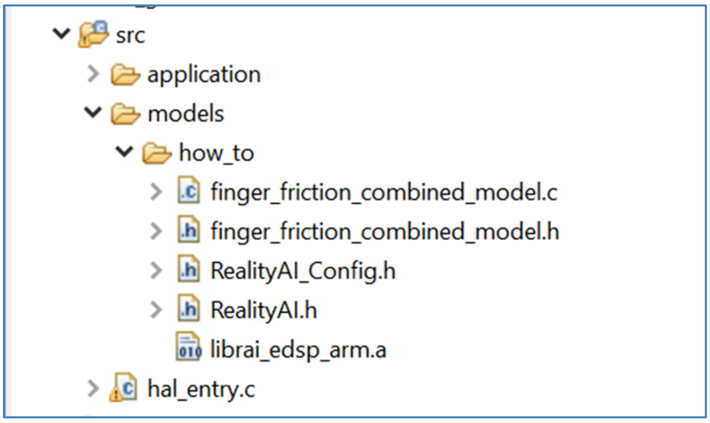


The package will take a few minutes to generate.

12. It will take ~10-15 minutes for the model to be available for download. Once ready, download the zip file using the highlighted button.



13. There are 7 files in the downloaded archive. Copy and paste all files except `example_main.c` and `readme.txt` to the src directory of the e² studio project workspace (**src** > **models** > **how_to**), overwriting the current files.
    Note: File names may be different depending on the project name assigned in tools.
    In the files below, the model function call is specified in `finger_friction_combined_model.h`.



4. Deploying the Model

4.1 Overview

This section covers testing the model on hardware.

4.2 Procedural Steps

1. Open `hal_entry.c` and import the header file. **If the deployed model name was finger_friction_combined then no changes are required**. Otherwise, edit the #include to the deployed model's header file.

```
#include "hal_data.h"
#include "mtr_main.h"

#include "RealityAI.h"
#include "finger_friction_combined_model.h"

FSP_CPP_HEADER
void R_BSP_WarmStart(bsp_warm_start_event_t event);
FSP_CPP_FOOTER

#define SMOOTH_LEN 7
#define USE_SHIPPER 1
```

2. Modify the following lines of code if the deployed model name was other than finger_friction_combined. Values can be found in the header file included in the last step.

```
#else
        pred = finger_friction_combined_predict(arg.p_sensor_data->p_frame_buf->p_buf);

        predBuff[SMOOTH_LEN - 1] = pred;

        for(int i = 0; i < SMOOTH_LEN - 1; i++)
            predBuff[i] = predBuff[i + 1];

        int cnt = 0, ucnt = 0;
        for(int i = 0; i < SMOOTH_LEN; i++){
            if(predBuff[i] == finger_friction_combined_friction)
                ucnt++;
            else if(predBuff[i] == finger_friction_combined_normal)
                cnt++;
        }

        if(ucnt > cnt)
            pred = finger_friction_combined_friction;
        else
            pred = finger_friction_combined_normal;

        switch(pred){
            case finger_friction_combined_friction:
                R_IOPORT_PinWrite(&g_ioport_ctrl, MTR_PORT_LED1, MTR_LED_ON);
                R_IOPORT_PinWrite(&g_ioport_ctrl, MTR_PORT_LED2, MTR_LED_ON);
                R_IOPORT_PinWrite(&g_ioport_ctrl, MTR_PORT_LED3, MTR_LED_ON);
                break;
            case finger_friction_combined_normal:
                R_IOPORT_PinWrite(&g_ioport_ctrl, MTR_PORT_LED1, MTR_LED_OFF);
                R_IOPORT_PinWrite(&g_ioport_ctrl, MTR_PORT_LED2, MTR_LED_OFF);
```

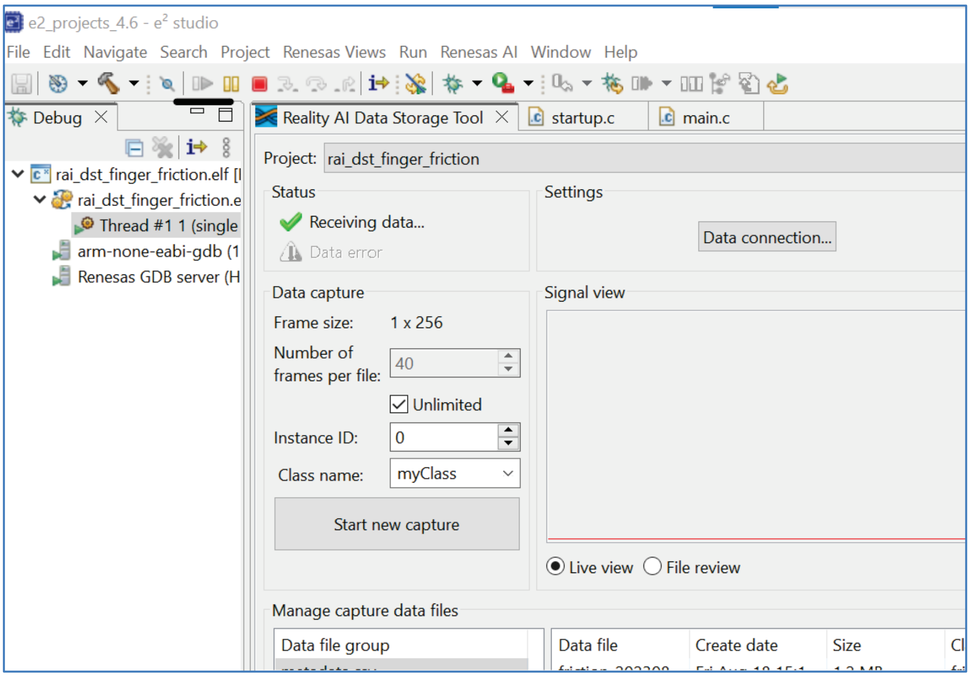
Edit lines above by copying the class names from header file included above. **Again, no edits are necessary if the deployed model name is finger_friction_combined**.

```
typedef enum finger_friction_combined_AICLASS{
    finger_friction_combined_no_results=0,
    finger_friction_combined_friction=1,
    finger_friction_combined_normal=2
} finger_friction_combined_AICLASS;

svm_classifier_struct* get_finger_friction_combined_model(void);
finger_friction_combined_AICLASS finger_friction_combined_predict(float* inputData);
```

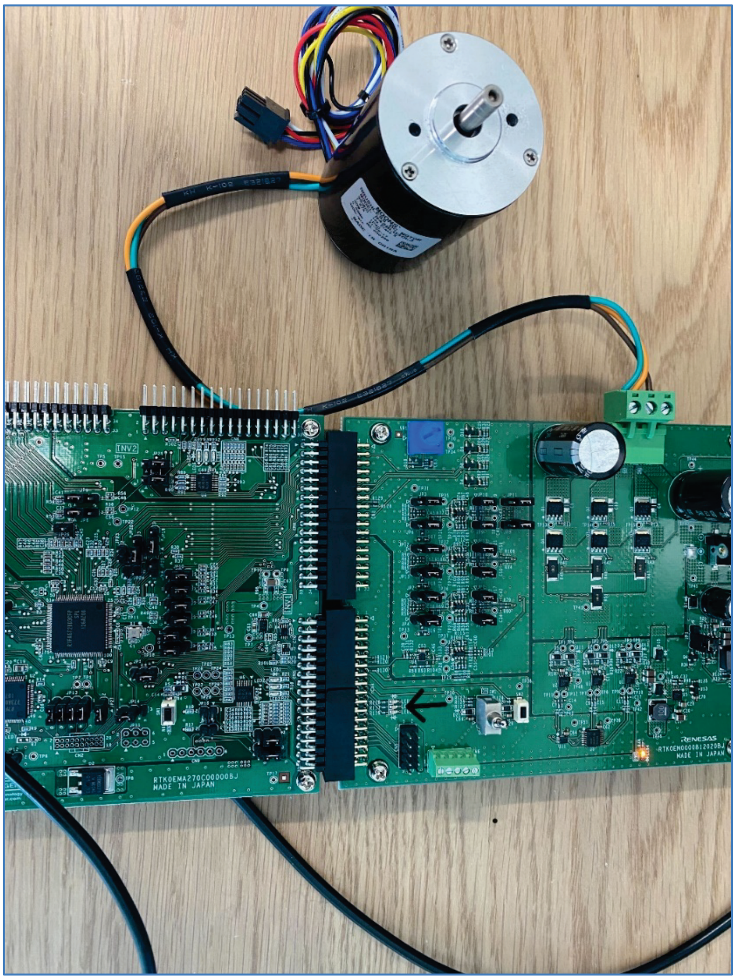
3. Change the USE_SHIPPER define to 0. This will cause the project to build in the test mode.

```
#include "finger_friction_combined_model.h"

FSP_CPP_HEADER
void R_BSP_WarmStart(bsp_warm_start_event_t
FSP_CPP_FOOTER

#define SMOOTH_LEN 7
#define USE_SHIPPER 0
```

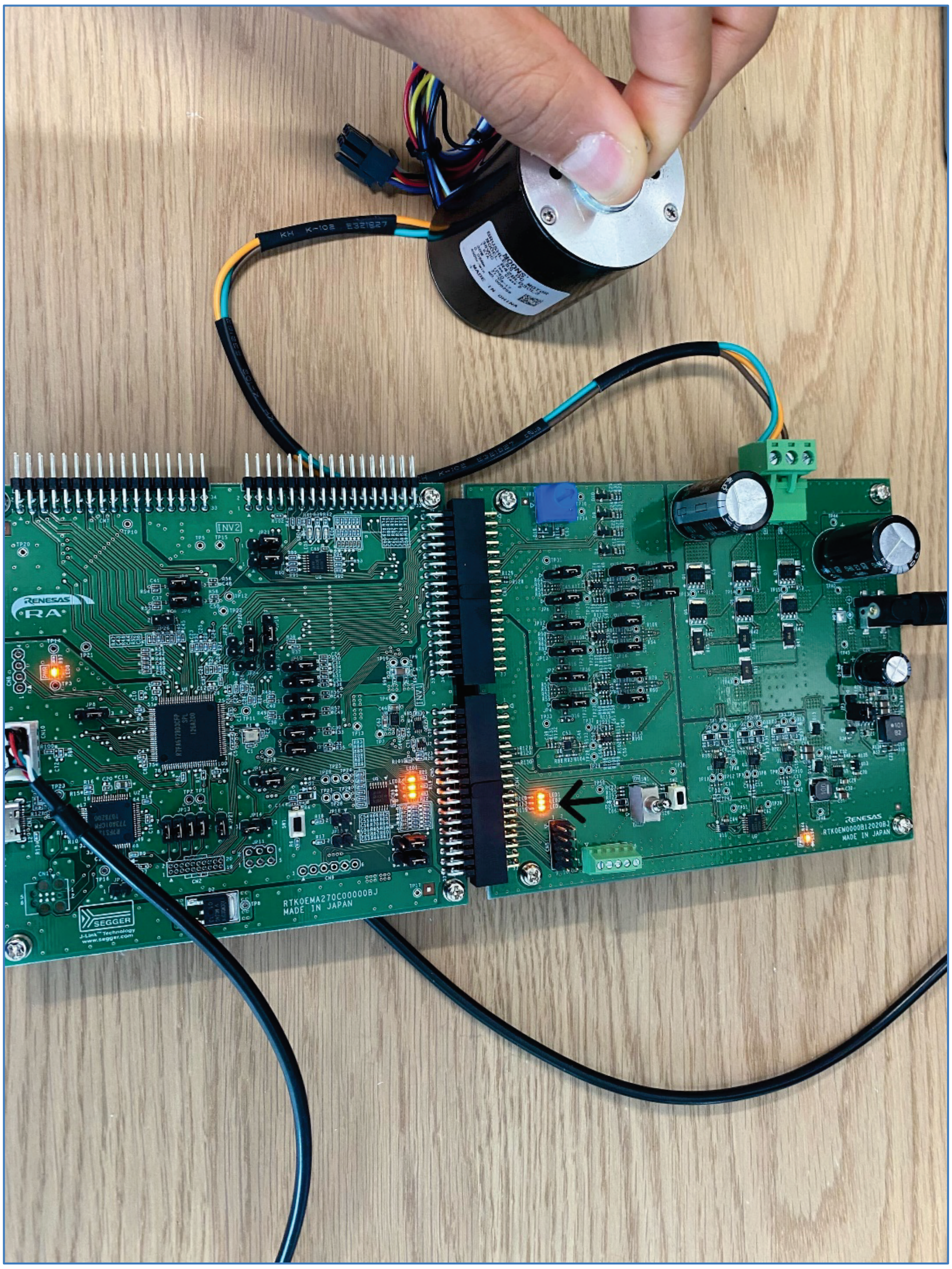
4. Build and debug the project as before. Click on the **Resume** button twice. Now the board is running on inference mode. Turn the motor switch ON and run the motor.



5. Let the motor run in normal/ balanced mode with no friction. The three LED's will be OFF (indicated by the arrow) during this time.



6. Apply minor/ light friction. The LED's should turn on (indicated by the arrow) to indicated friction/ unbalanced behavior.



What to do if the mode is not performing well?

Collect some more data and retrain the model. Usually additional data collection helps in creating better performing models across different conditions.

Double check if the data collection method and testing method are the same. For example: Are you collecting unbalanced data by applying high friction to the motor shaft by testing it by applying low friction? In that case, performance may be inconsistent. Collecting more data across these variations should help.

Warning: Be to turn the toggle switch off before stopping the debugger.

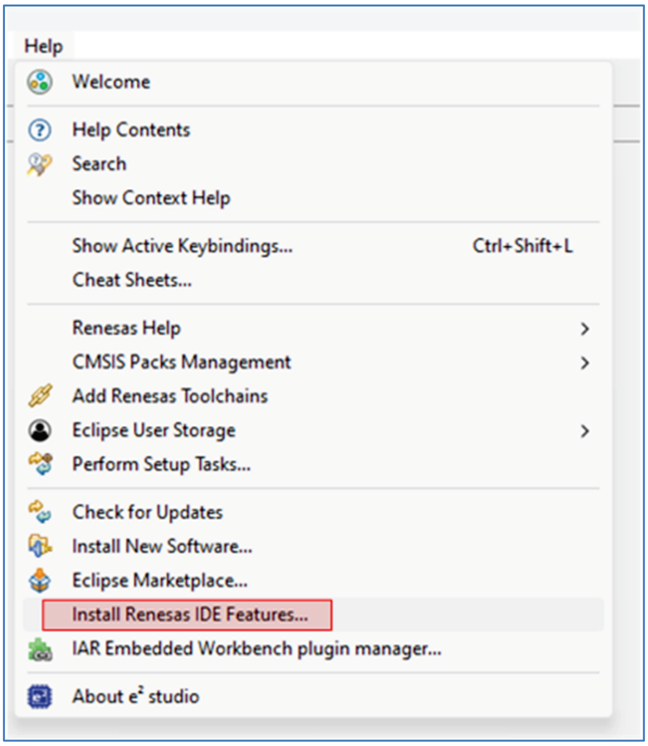
5. Appendix

5.1 Overview

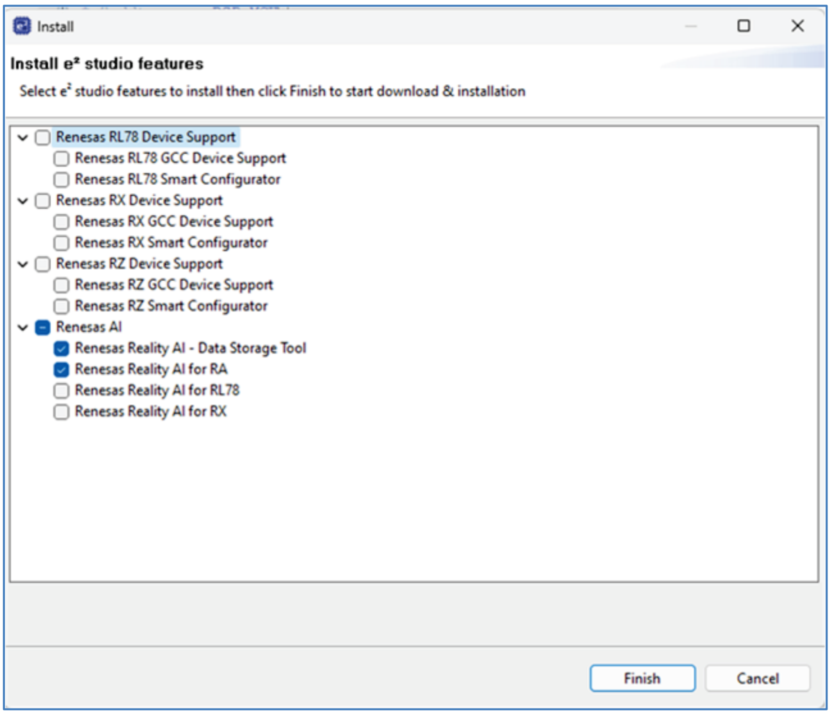
If the Reality AI items do not appear in the title bar, perform the following steps.

5.2 Procedural Steps

1. Click on **Help** > **Install Renesas IDE Features…**



2. Select the Reality AI features and click **Finish**



Restart e^2 studio when prompted.

6. Website and Support

Visit the following URLs to learn about key elements of the RA family, download components and related documentation, and get support:

RA Product Information	renesas.com/ra
RA Product Support Forum	renesas.com/ra/forum
RA Flexible Software Package	renesas.com/FSP
Renesas Support	renesas.com/support

Revision History

Rev.	Date	Description	
		Page	Summary
1.00	Sep.14.23	—	Initial release

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Precaution against Electrostatic Discharge (ESD)

   A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.
2. Processing at power-on

   The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.
3. Input of signal during power-off state

   Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.
4. Handling of unused pins

   Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.
5. Clock signals

   After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.
6. Voltage application waveform at input pin

   Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (Max.) and V_{IH} (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (Max.) and V_{IH} (Min.).
7. Prohibition of access to reserved addresses

   Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.
8. Differences between products

   Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
6. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.
   Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION ("Vulnerability Issues"). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
12. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit: www.renesas.com/contact/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.