

E2-Studio Integration – Getting Started Guide

Using Renesas RA6T2 Flexible Motor Control Kit

Description

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

This document contains the following sections:

Lab Section	Demo Steps
Section 1	Prerequisites
Section 2	Data Collection
Section 3	Model Generation
Section 4	Testing the Models
Section 5	Appendix

Sections should be followed sequentially.

Lab Objectives <ul style="list-style-type: none"> Gain hands-on experience on developing a simple AI demo using Renesas RA6T2 motor kit + Reality AI Tools. 	Lab Materials <ul style="list-style-type: none"> Please verify you have the following materials at your lab station. <ul style="list-style-type: none"> PC with Windows 11. Project files provided by Renesas - Reality AI team (link in section 1 of the document). E2-Studio IDE v4.6 (link in section 1 of the document). Reality AI Tools Account
---	--

Lab Sections

1	Prerequisites.....	3
2	Data Collection	8

3	Creating Models	18
4	Deploying the model	29
5	Appendix.....	33

1 Prerequisites

Overview

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

Procedural Steps

This lab requires the RA6T2 Flexible Motor Control Kit.

Overview

Description Features Applications Related Products

The MCK-RA6T2 is a development kit that enables easy evaluation of motor control using permanent magnet synchronous motors (brushless DC motors). With this product and the sample code and QE for Motor that can be downloaded from the website, you can start evaluating motor control using the RA6T2 right away.



The MCK-RA6T2 kit comes with:

1. RA6T2 board
2. DC brushless motor by moon 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.

Assemble the kit as shown below. Make sure the toggle switch is in the OFF position. All other jumpers in their default condition. Reference the documentation that came with the kit if there is concern that the board may not be in the default configuration.

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 diagram below for pinning of the UART to USB)
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)

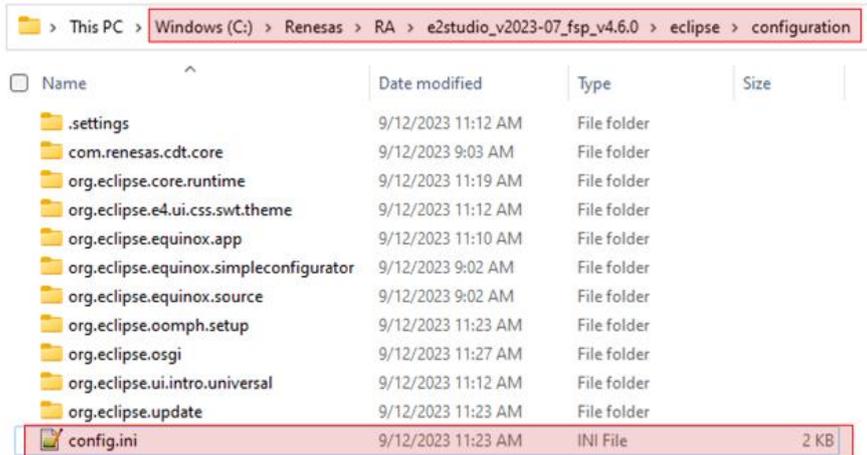
1*3P Female Socket	Name	Colour	Description
Pin 1	GND	Black	Device ground supply
Pin 2	TXD	White	Transmit Asynchronous Data
Pin 3	RXD	Green	Receive Asynchronous Data

This lab requires Renesas E2-Studio IDE 2023-07 or newer with FSP 4.6 or newer (till FSP 5.2).

Platform installer available here: [Link](#) Take note of where e2studio is installed.

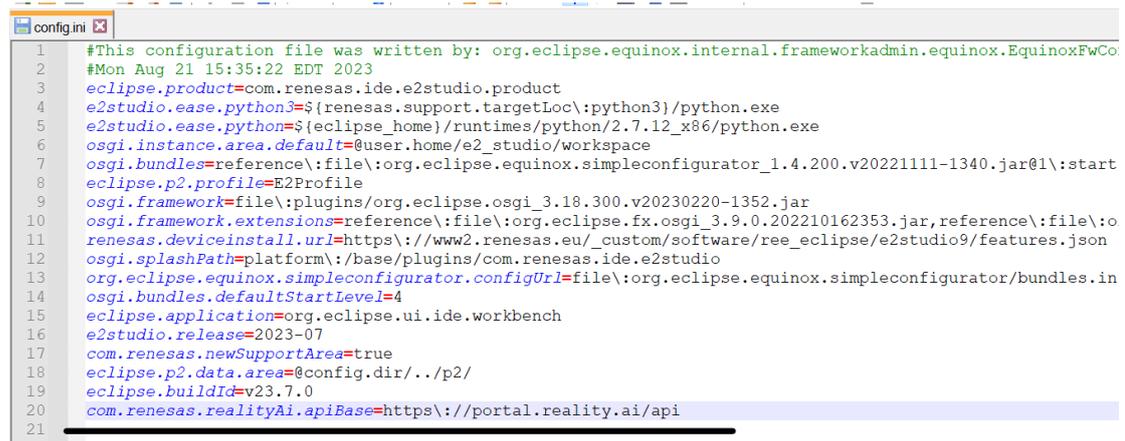
This lab requires modification to the eclipse configuration file if FSP 4.6 is used. For later versions, skip this step.

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 e2studio if it is currently running.**

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 – Reality AI team members if you need credentials + instruction documents for the portal.

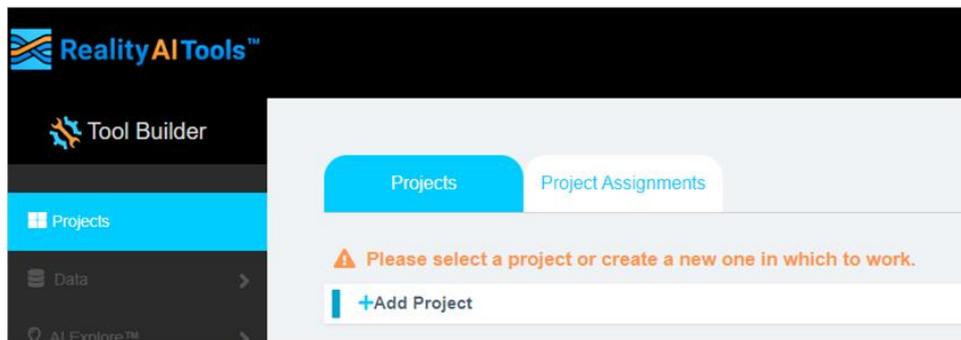


Username
RenesasUser

Password
.....

Stay logged in

Login



Leave the browser open in the background.

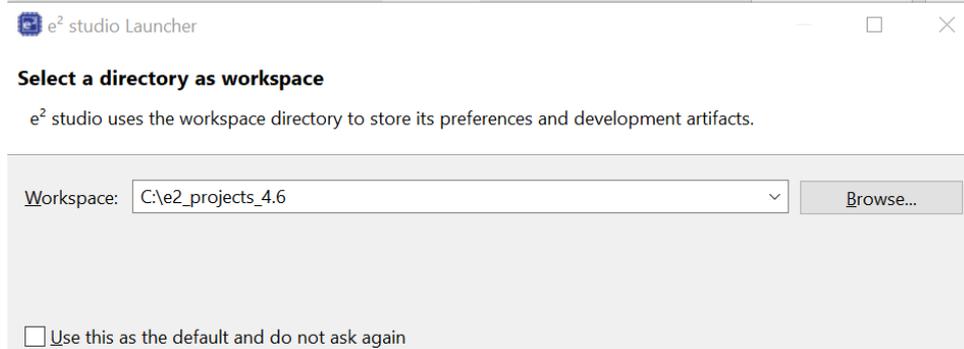
2 Data Collection

Overview

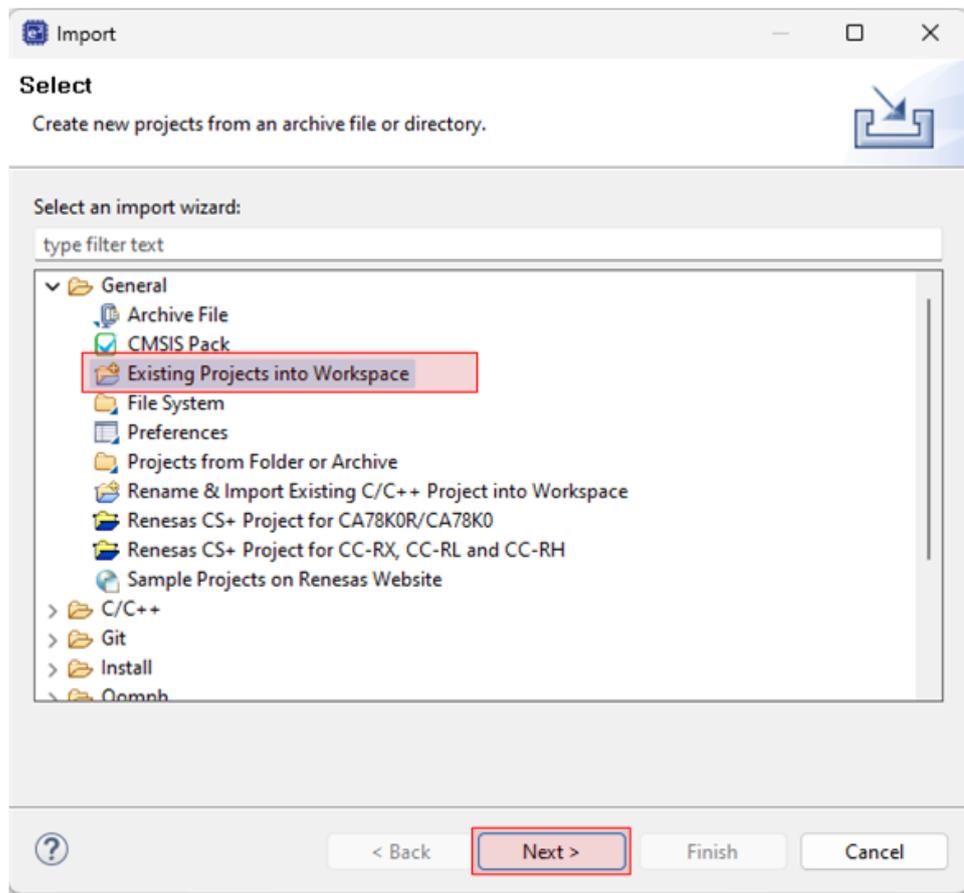
This section covers collecting data with the Reality-AI eclipse plugin using e2studios.

Procedural Steps

Open E2 Studio IDE and select a workspace. Although the workspace can be in any folder, this lab assumes the workspace shown below. c:\e2_projects_4.6

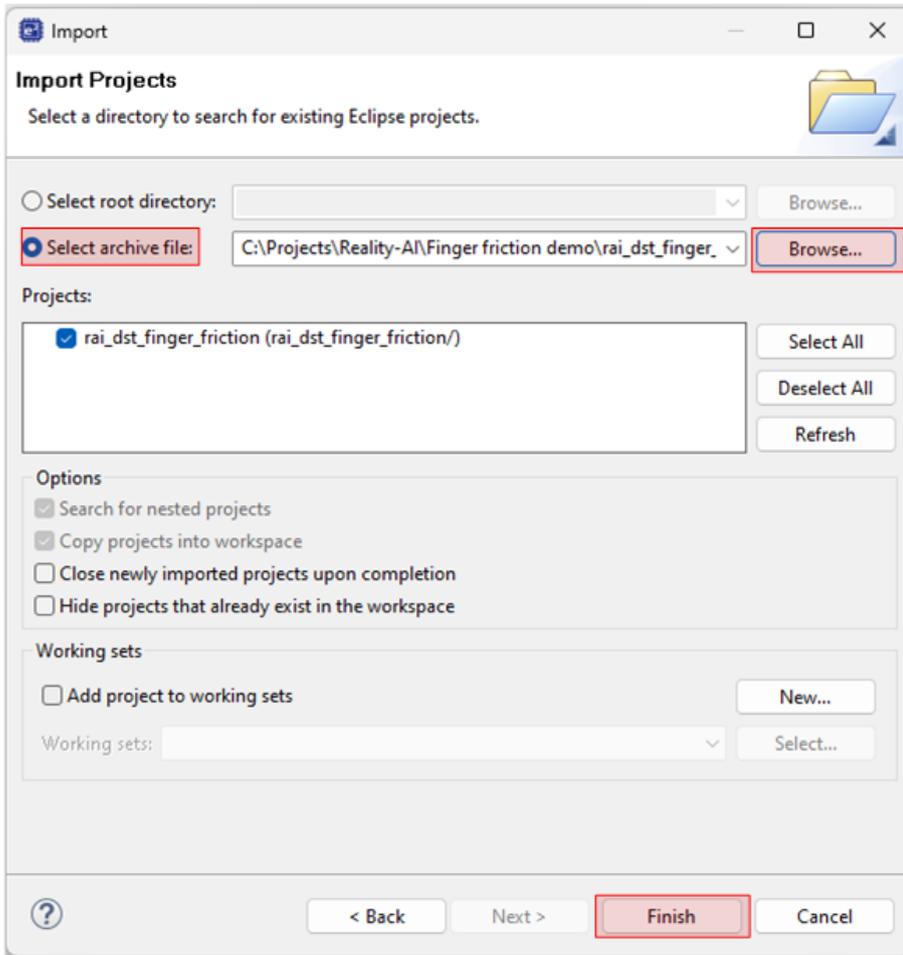


Go to File → Import... Then choose General->Existing Projects into Workspace



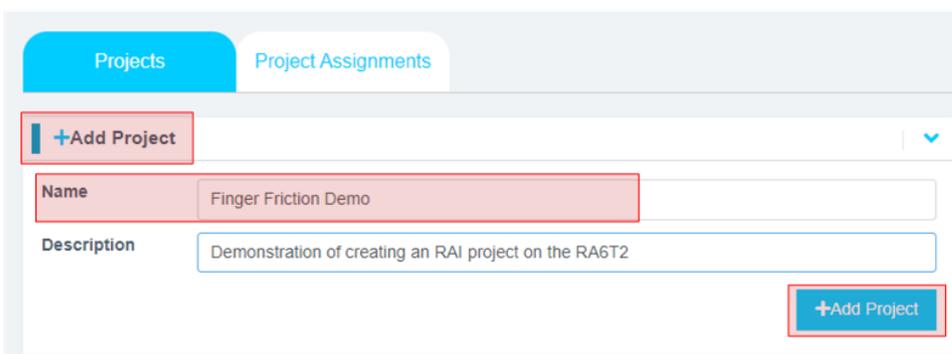
Click Next to continue.

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.

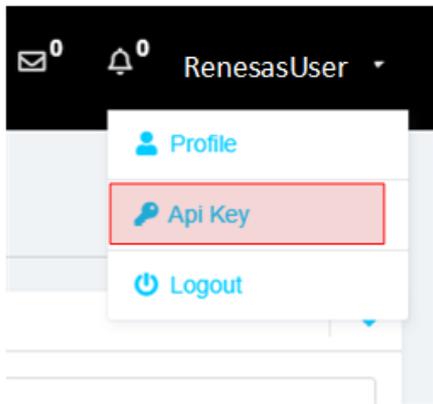


The demo project is imported into the workspace

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.



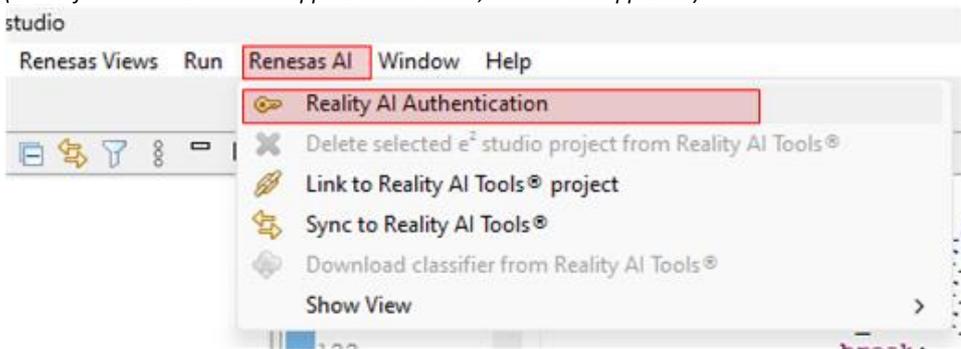
Navigate to the username on the top-right of the screen and select API Key Option.



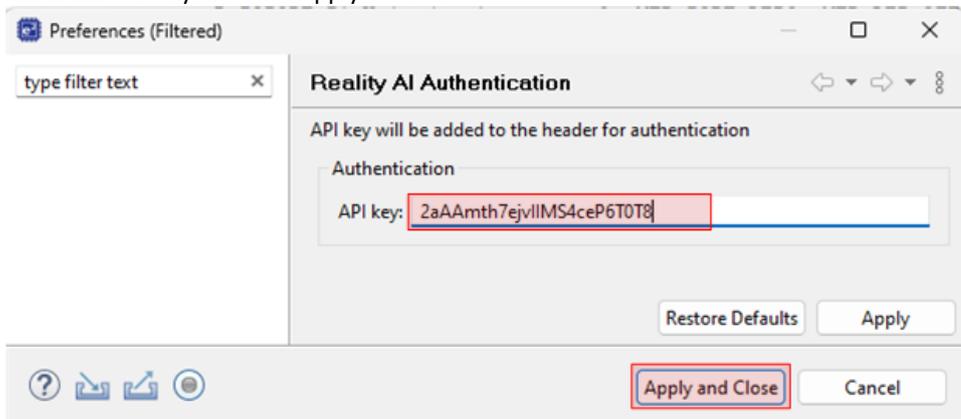
Copy this API key to the clipboard. This key will be used to connect e2 studio IDE and Reality AI Tools.



Open e2 studio and navigate to Renesas AI --> Reality AI Authentication
(note: If Renesas AI does not appear in the menu, consult the appendix)

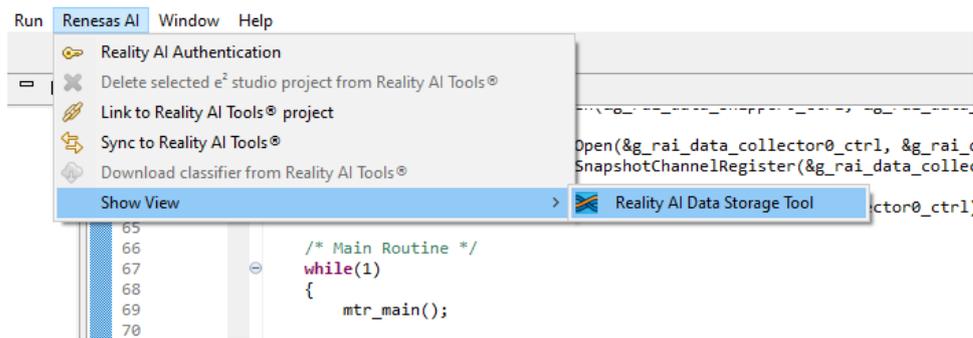


Paste the API key and click Apply and Close

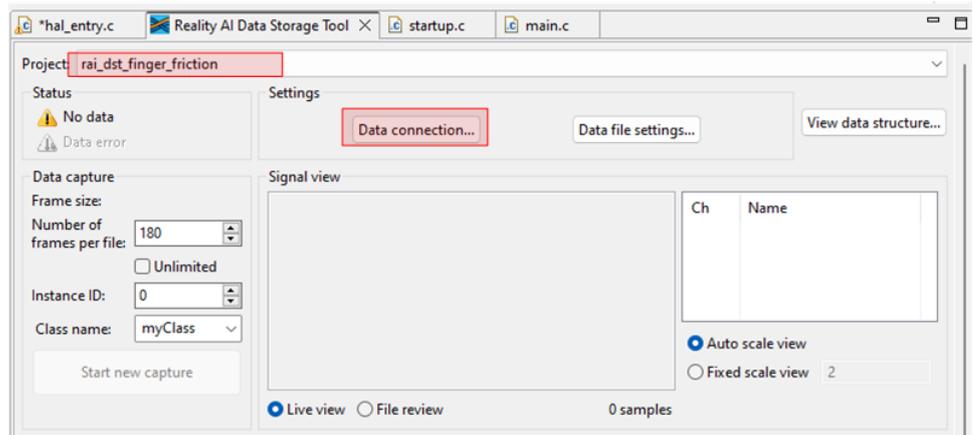


The IDE can now connect to the Reality AI site

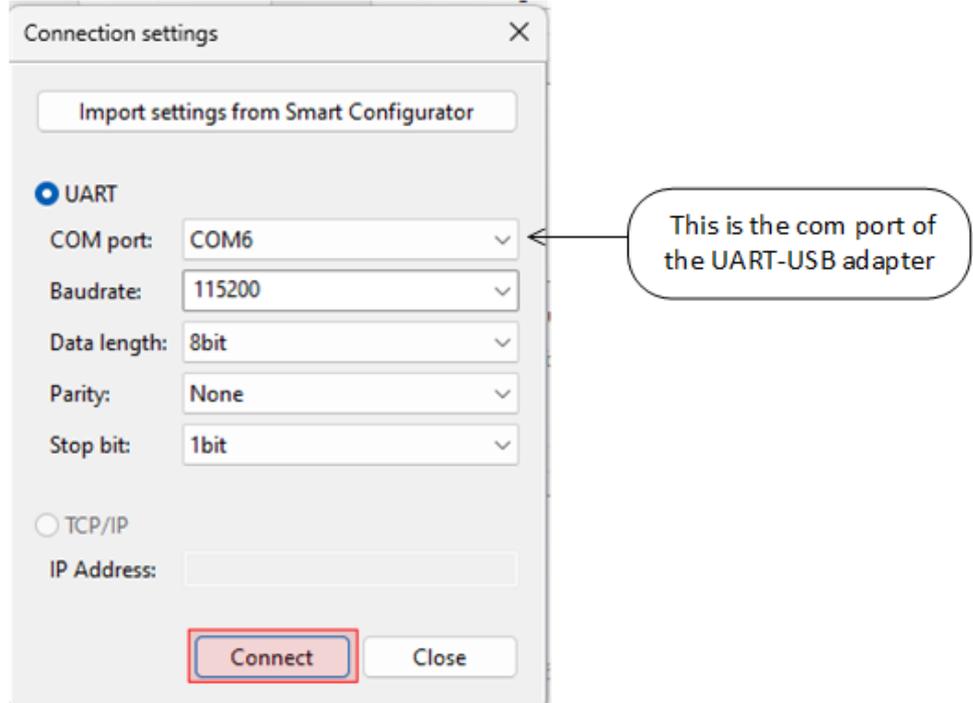
Navigate to Renesas AI --> Show View --> Reality AI Data Storage Tool



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.

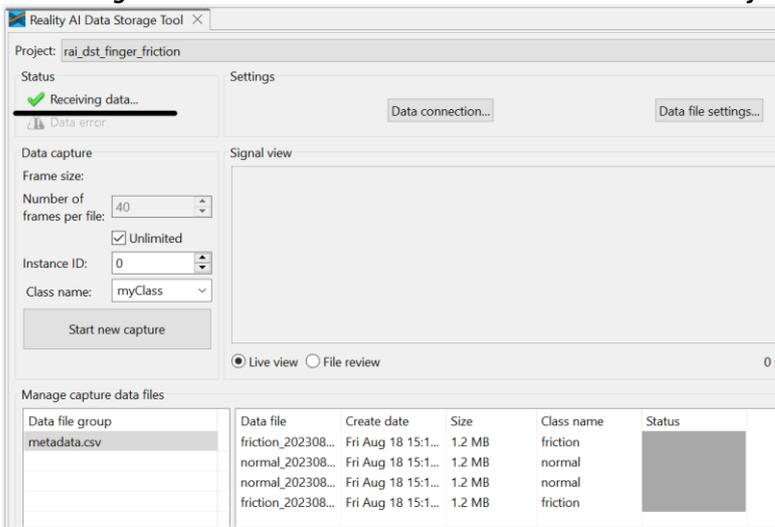


In the menu that pops up, select the COM port, baudrate (115200), and communication protocol parameters (8-N-1). Then press **Connect** --> then press **Close**.

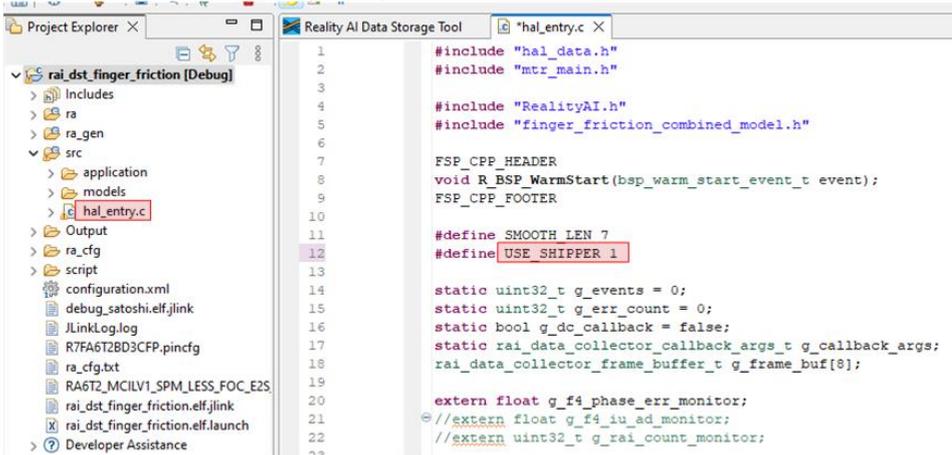


Once connected, the status (top left) should show "Receiving data..." with a check mark.

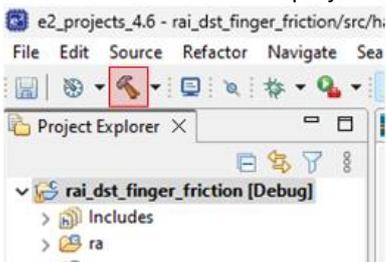
Note: the green check mark indicates that the connection is successful. Not that it is receiving data yet.



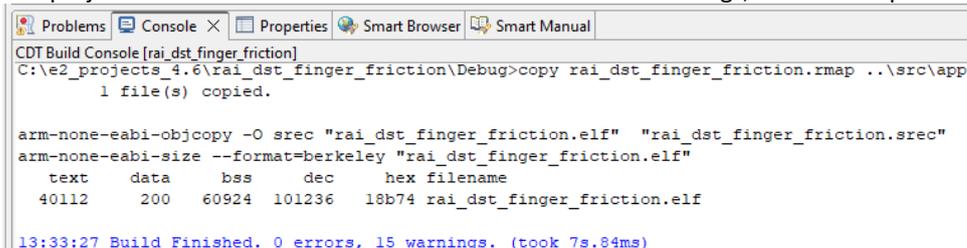
Expand the project in the Project Explorer view and open hal_entry.c. Verify/change USE_SHIPPER to be defined as 1. This will build the project in the data collection mode.



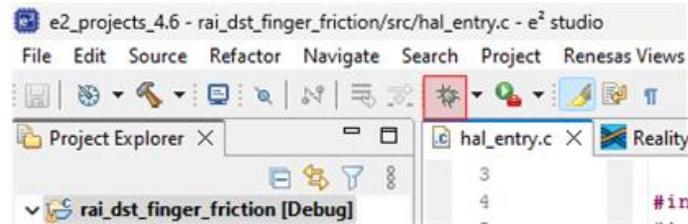
Save the file and build the project by clicking the hammer icon



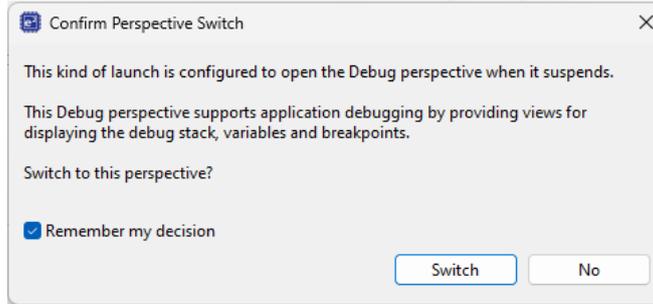
The project should build without error. There will be some warnings, these are expected.



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” checkbox and click Switch if the following dialog appears.



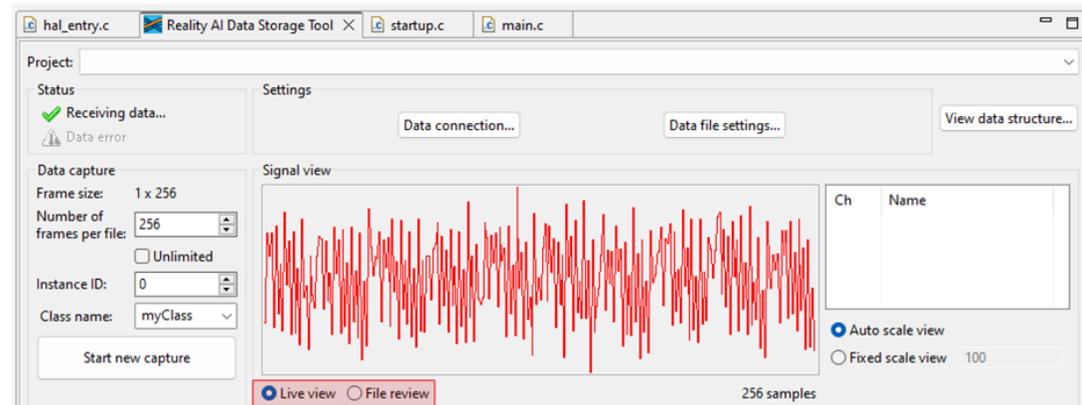
Click the Resume button twice. The program is now running



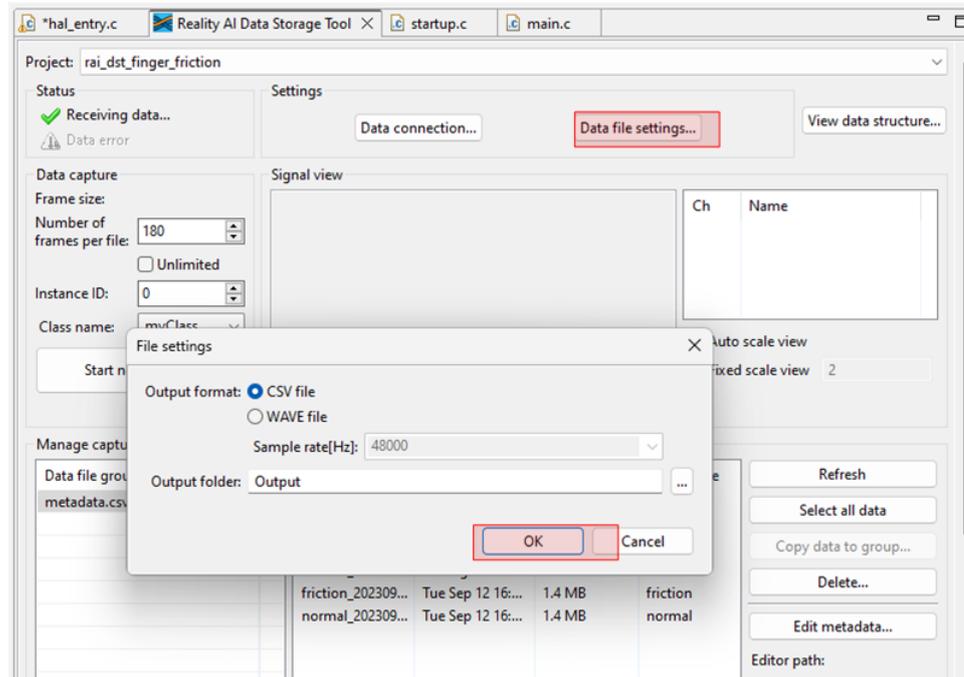
Switch the toggle switch to the ON position. The motor should now be running.

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.

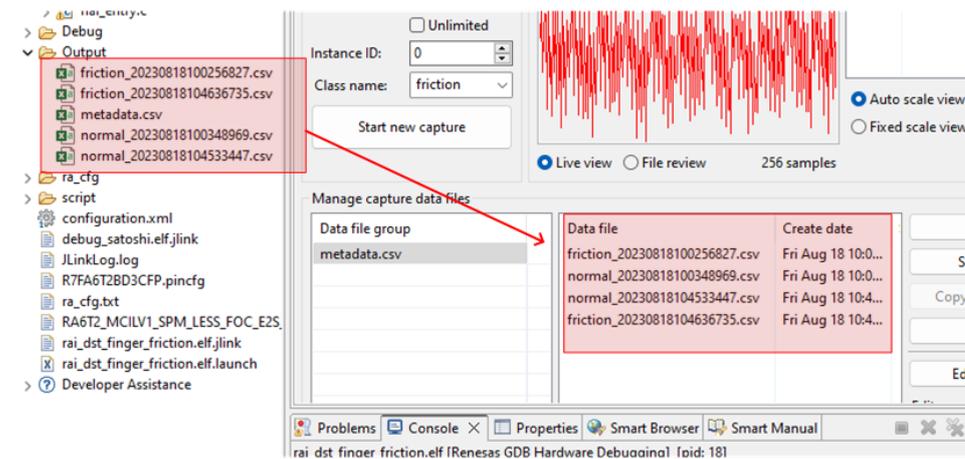
Click on the “Reality AI Data Storage Tool” view and verify the presence of a wave form in the Signal view. If there is nothing showing, click File review and then Live view (highlighted below). If there is still no waveform, recheck the prerequisite steps to make sure the connections are correct.



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

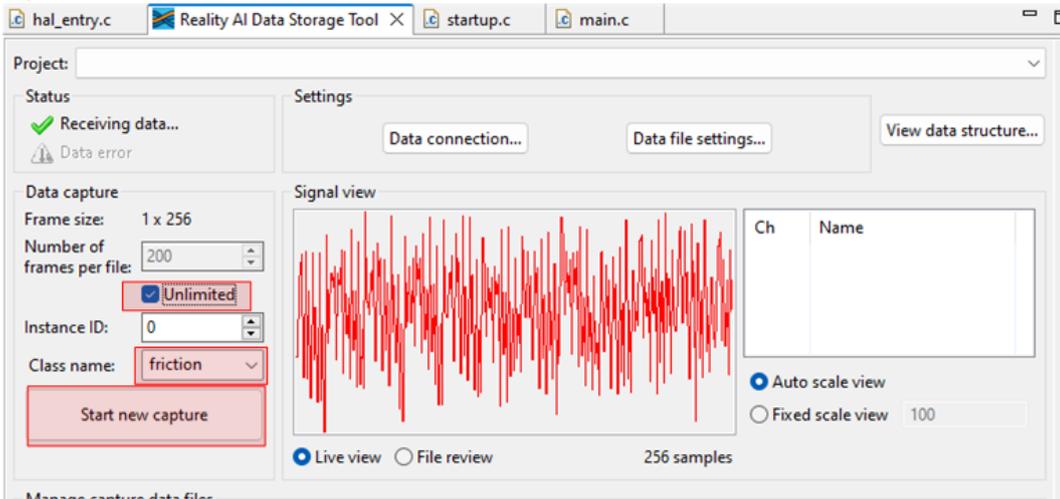


Note that the Data Storage tool reflects the data in the project's Output folder. The data currently there was collected during the lab development and includes date and time information. New data will be collected in the following sections.



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

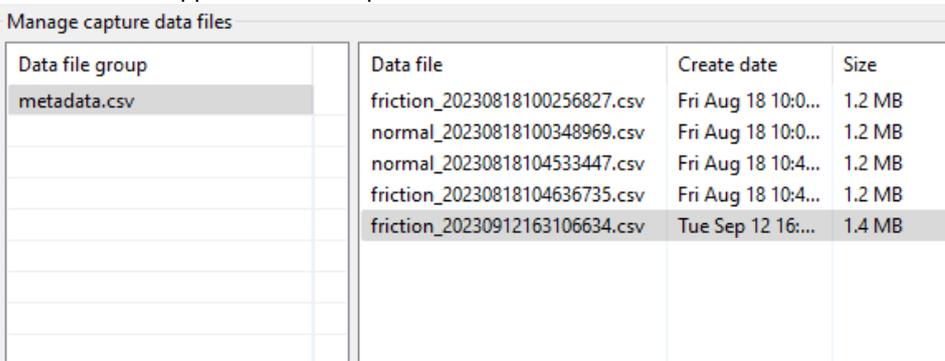
In the Reality AI Data Storage Tool, check the Unlimited checkbox. Enter the Class name “friction.” While applying and maintaining slight friction to the motor shaft, Click the “Start new Capture” button. The button text will change to “Stop Capture.” **Maintain friction for 45 to 60 seconds** and then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



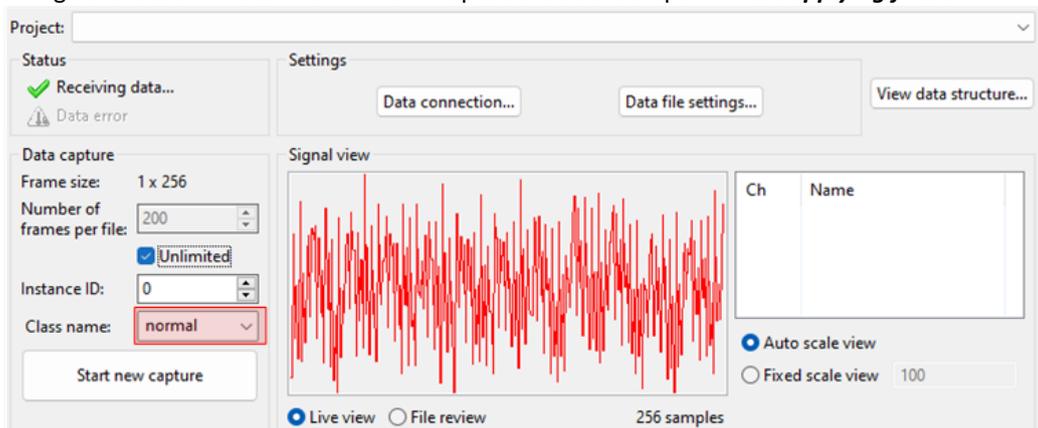
45 to 60 second's elapses...



A new data file appears in the output folder



Change the Class name to “normal” and repeat the above steps **without applying friction**



This concludes the data capture portion.

Turn the toggle switch off to stop the motor!

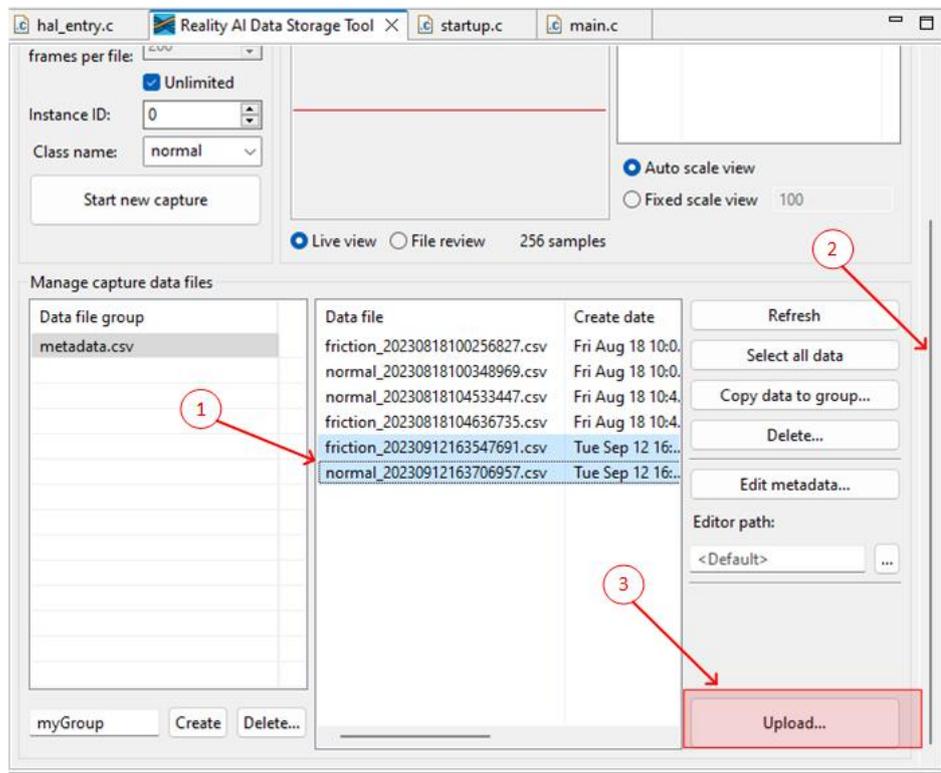
Click the red square icon to terminate the debug session.



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

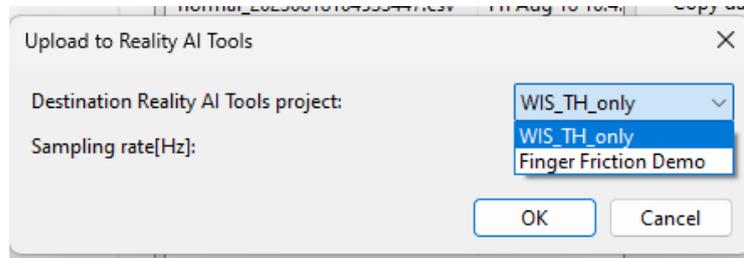
Follow these steps to upload the data files collected in the last section:

1. Select the two files using click + ctrl-click
2. Scroll the view to expose the Upload... button
3. Click the Upload... button



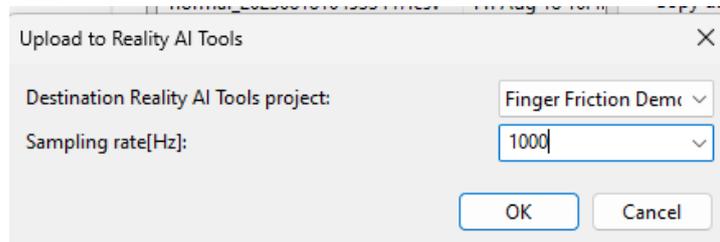
A dialog box should appear with the projects currently defined on the Reality AI website.

(if there are no projects in the pulldown, verify the changes made to the config.ini file, regenerate an API key and restart e2studios)



Select the Finger Friction Demo (or whatever the project was named in the prior step)

Set the Sampling rate to 1000 and press OK



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

You have finished this section.

3 Creating Models

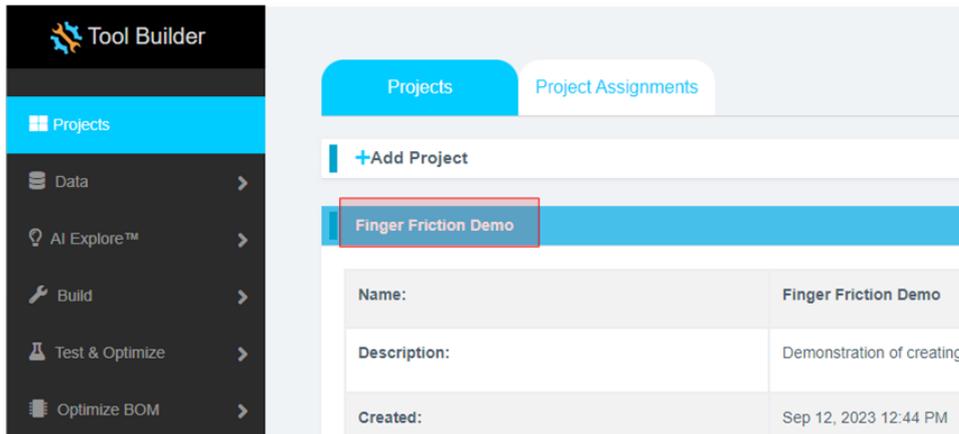
Overview

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

Procedural Steps

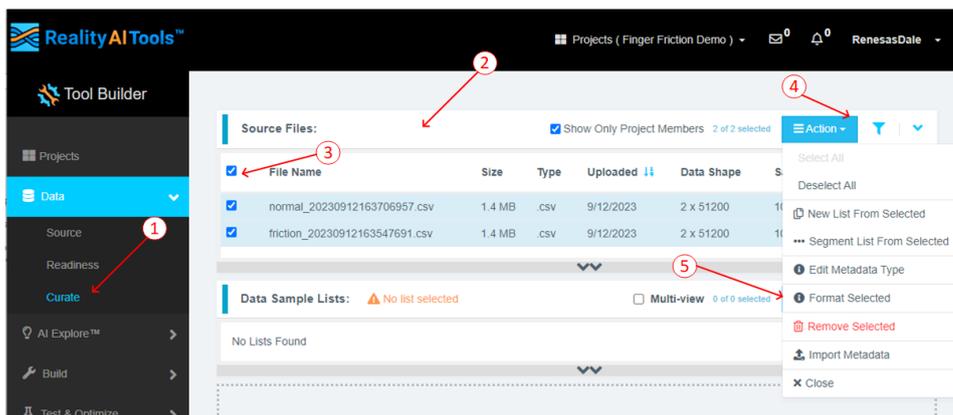
If not already logged in to Reaity AI Tools, use this link: <https://portal.reality.ai/login> with the provided password. Reach out to Renesas – Reality AI team members for credentials + instructions documents for the portal if needed.

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)

Define file format for normal_20230912163706957.csv

Display raw data Display issues only

#1	Data -	#2	Data -
1	Inde: Data	1	ch
2	0 Target Class	0.002558606443926692	
3	1 Target Value	-0.018888363614678383	
4	2 Categorical MetaData	0.019179338589310646	
5	3 Numerical MetaData	-0.017041284590959550	

Label Row
Check if CSV has a header row

Sample Rate
Sample Frequency in Hz:

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

Define file format for normal_20230912163706957.csv

Display raw data Display issues only

5	3 Numerical MetaData	-0.017041284590959550
---	----------------------	-----------------------

Sequential #'s / Time Code

Date and Time

Date only

Time only

Ignore

Label Row
Check if CSV has a header row

Sample Rate
Sample Frequency in Hz:

4. Click confirm

Unsaved changes to file columns. To save these changes, press "Confirm."

Label Row
Check if CSV has a header row

Sample Rate
Sample Frequency in Hz:

Delimiter
Character used to separate columns:

European Decimals
Check if numbers use commas as a decimals. IE: 1.234.567,89

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

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

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

The screenshot shows the 'Source Files' section of the Reality AI Tools interface. It contains a table with the following data:

File Name	Size	Type	Uploaded	Data Shape	Sample Rate
normal_20230912163706957.csv	1.4 MB	.csv	9/13/2023	1 x 51200	1000 Hz
friction_20230912163547691.csv	1.4 MB	.csv	9/13/2023	1 x 51200	1000 Hz

An 'Action' dropdown menu is open, showing options such as 'Select All', 'Deselect All', 'New List From Selected', and 'Segment List From Selected' (which is highlighted with a red box). Other options include 'Edit Metadata Type', 'Format Selected', 'Remove Selected', 'Import Metadata', and 'Close'.

Why Segmentation?

One of the main purposes of generating models from Tools is to deploy them to a variety of Renesas MCU's. 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.

Assign window length, overlap, and provide name to the list. Follow these 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 screenshot shows the 'Segment Files' configuration window. It includes the following elements:

- Segment 2 files:** normal_20230912163706957.csv, friction_20230912163547691.csv
- Segmentation Method:** Sliding CSV Window
- Sample Rate:** 1000 Hz
- Target:** File metadata target column (type: Class)
- Window Length:** 256 (highlighted with a red box and circled '1')
- Offset:** 128 (Number of rows between sample start points), 50% (highlighted with a red box and circled '2')
- Output Sample List:** 256_50_percent_overlap (highlighted with a red box and circled '3')
- Buttons:** Cancel, Submit (highlighted with a red box and circled '4')

The number of estimated samples will be dependent on the amount of data collected. So if your estimated samples don't match the below screenshot, its still okay.

Window Length: The window length determines how much data will be considered by the AI to decide on a classification.

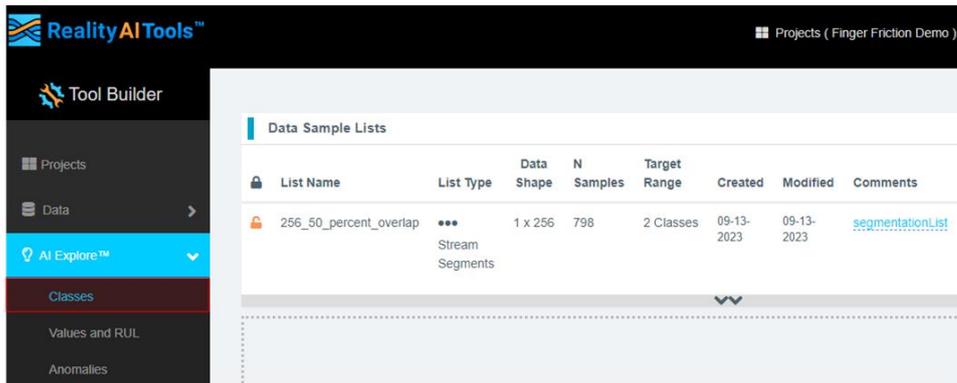
Offset: The offset determines how far ahead from the start of the last window the parser moves before creating a new window.

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.

After successfully creating the segmented list, navigate to AI Explore → Classes



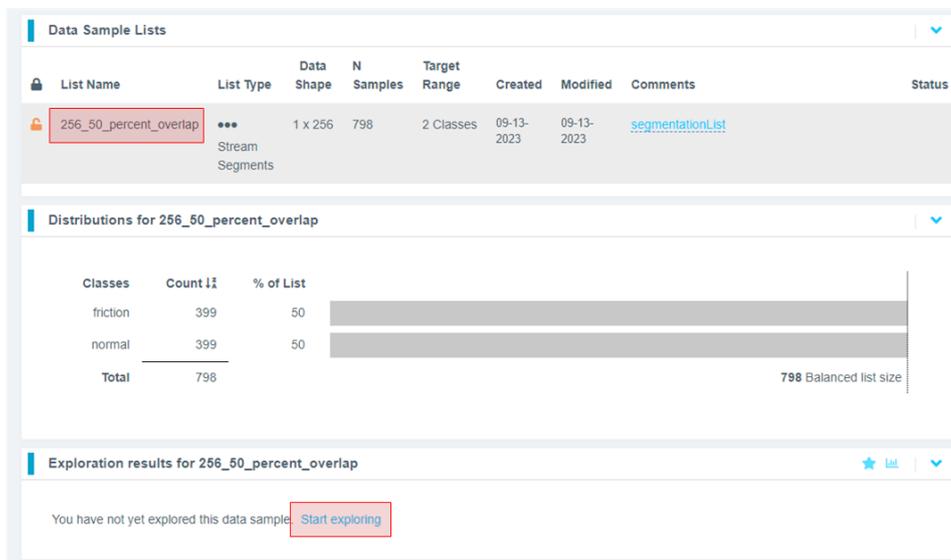
Reality AI Tools has options for creating 3 types of AI Models:

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.

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



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.

Exploration results for 256_50_percent_overlap

Explore: Complete 100 %

Favorites	Complexity	KFold Accuracy		Training Separation		Create Base Tool	Explanation	Confusion Matrix
		Overall %	Worst %	Overall %	Worst %			
★		95	95	95	95			
★		82	81	83	82			
★		100	100	100	100			

Channel 1
Spectral Magnitude

Show more results

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.**

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

Create Base Tool
✕

Name

Description

Add

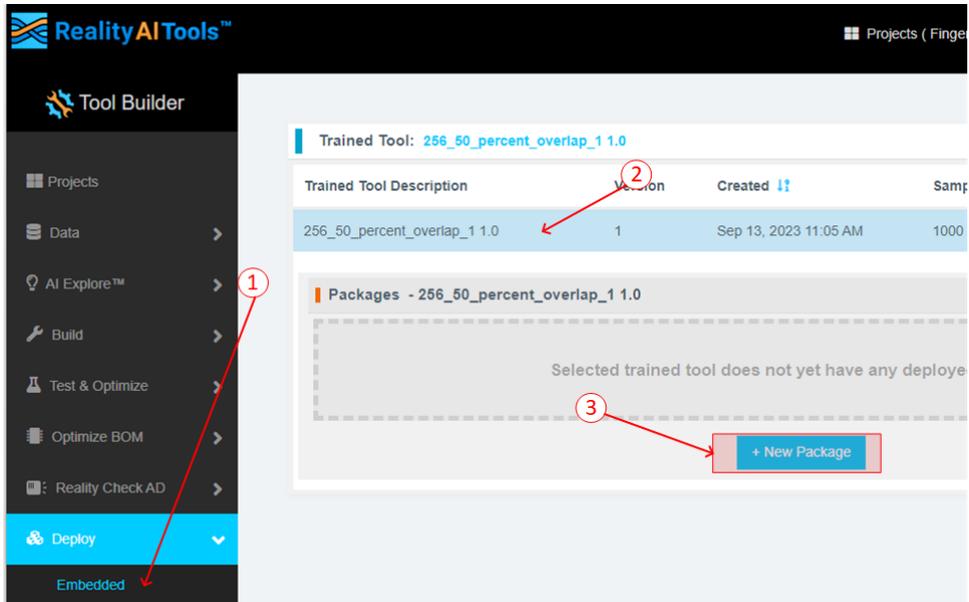
The icon will change, indicating the base tool has been created.

✓ Explore: Complete 100 %

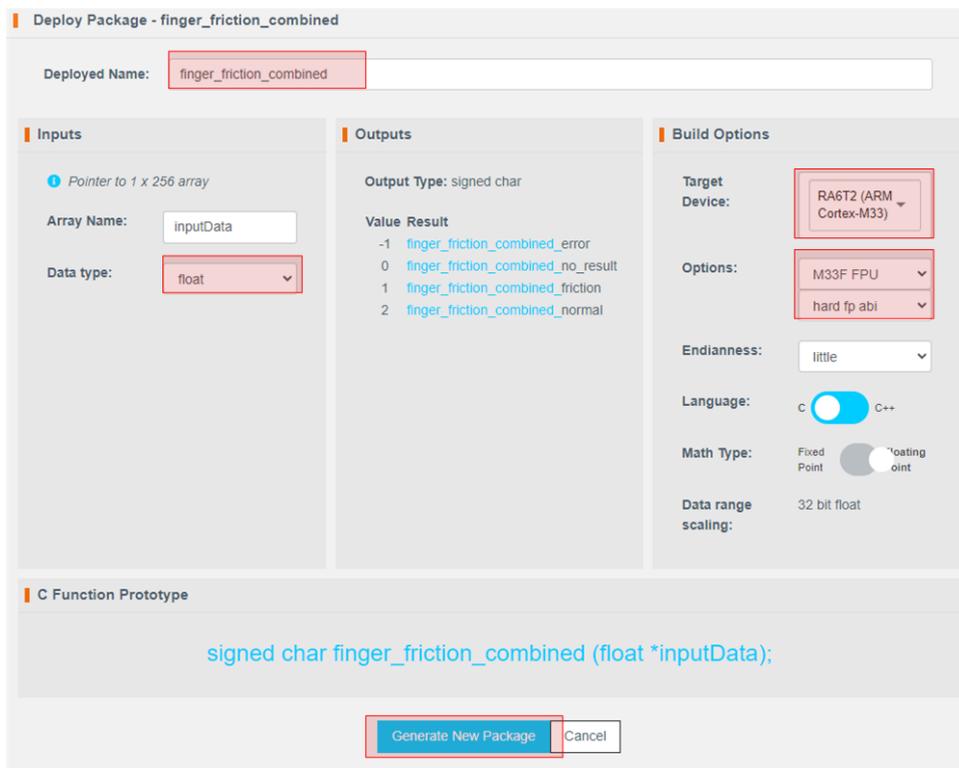
KFold Accuracy		Training Separation		Create Base Tool
Overall %	Worst %	Overall %	Worst %	
95	95	95	95	
82	81	83	82	
100	100	100	100	

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



Set the options as indicated. The deployed name should be finger_friction_combined to match the code in e2studios. Choosing another name will require additional edits to the code.



The package will take a few minutes to generate.

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

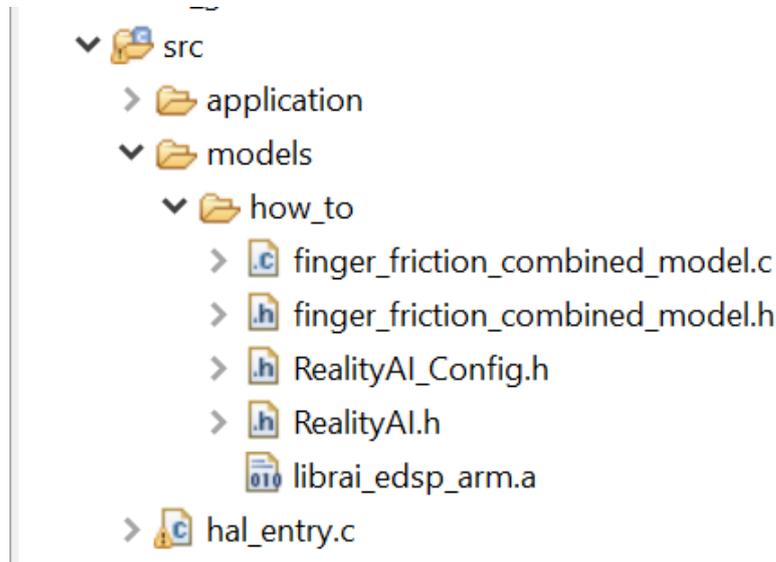
Deployed Name	Package Date	input Data	Parameters	Target	Math Type	Download
finger_friction_combined	Sep 13, 2023 11:21 AM	float *inputData	internal	RA6T2 (ARM Cortex-M33)	float 32	

[+ New Package](#)

There are 7 files in the downloaded archive. Copy and paste all files except example_main.c & 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`.



You have finished this section.

4 Deploying the model

Overview

This section covers testing the model on hardware.

Procedural Steps

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.

```

1      #include "hal_data.h"
2      #include "mtr_main.h"
3
4      #include "RealityAI.h"
5      #include "finger_friction_combined_model.h"
6
7      FSP_CPP_HEADER
8      void R_BSP_WarmStart(bsp_warm_start_event_t event);
9      FSP_CPP_FOOTER
10
11     #define SMOOTH_LEN 7
12     #define USE_SHIPPER 1
--

```

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.

```

91     #else
92         pred = finger_friction_combined_predict(arg.p_sensor_data->p_frame_buf->p_buf);
93
94         predBuff[SMOOTH_LEN - 1] = pred;
95
96         for(int i = 0; i < SMOOTH_LEN - 1; i++)
97             predBuff[i] = predBuff[i + 1];
98
99         int cnt = 0, ucnt = 0;
100        for(int i = 0; i < SMOOTH_LEN; i++){
101            if(predBuff[i] == finger_friction_combined_friction)
102                ucnt++;
103            else if(predBuff[i] == finger_friction_combined_normal)
104                cnt++;
105        }
106
107        if(ucnt > cnt)
108            pred = finger_friction_combined_friction;
109        else
110            pred = finger_friction_combined_normal;
111
112        switch(pred){
113            case finger_friction_combined_friction:
114                R_IOPORT_PinWrite(&_ioport_ctrl1, MTR_PORT_LED1, MTR_LED_ON);
115                R_IOPORT_PinWrite(&_ioport_ctrl1, MTR_PORT_LED2, MTR_LED_ON);
116                R_IOPORT_PinWrite(&_ioport_ctrl1, MTR_PORT_LED3, MTR_LED_ON);
117                break;
118            case finger_friction_combined_normal:
119                R_IOPORT_PinWrite(&_ioport_ctrl1, MTR_PORT_LED1, MTR_LED_OFF);
120                R_IOPORT_PinWrite(&_ioport_ctrl1, MTR_PORT_LED2, MTR_LED_OFF);
121                R_IOPORT_PinWrite(&_ioport_ctrl1, MTR_PORT_LED3, MTR_LED_OFF);
122        }
--

```

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

```

51     #define FINGER_FRICION_COMBINED_AI_CLASS_NAME "finger_friction_combined_AI_CLASS"
52     #typedef enum finger_friction_combined_AI_CLASS{
53         finger_friction_combined_no_results=0,
54         finger_friction_combined_friction=1,
55         finger_friction_combined_normal=2
56     } finger_friction_combined_AI_CLASS;
57
58     svm_classifier_struct* get_finger_friction_combined_model(void);
59     finger_friction_combined_AI_CLASS finger_friction_combined_predict(float* inputData);
60
--

```

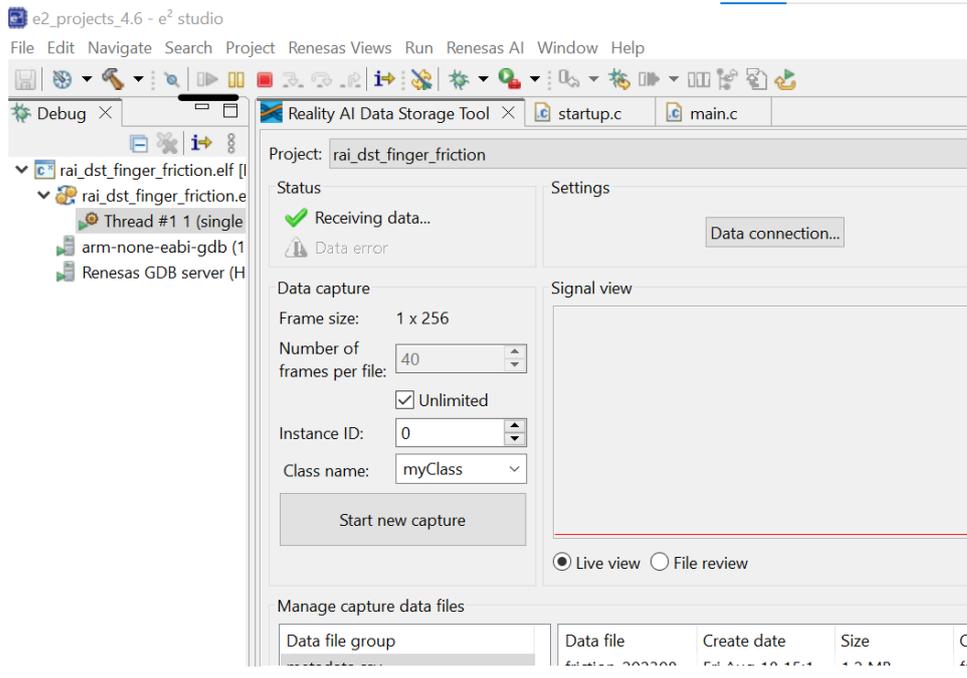
Change the USE_SHIPPER define to 0. This will cause the project to build in the test mode.

```

4      #include "realityAI.n"
5      #include "finger_friction_combined_model.h"
6
7      FSP_CPP_HEADER
8      void R_BSP_WarmStart(bsp_warm_start_event_t
9      FSP_CPP_FOOTER
10
11     #define SMOOTH_LEN 7
12     #define USE_SHIPPER 0
13

```

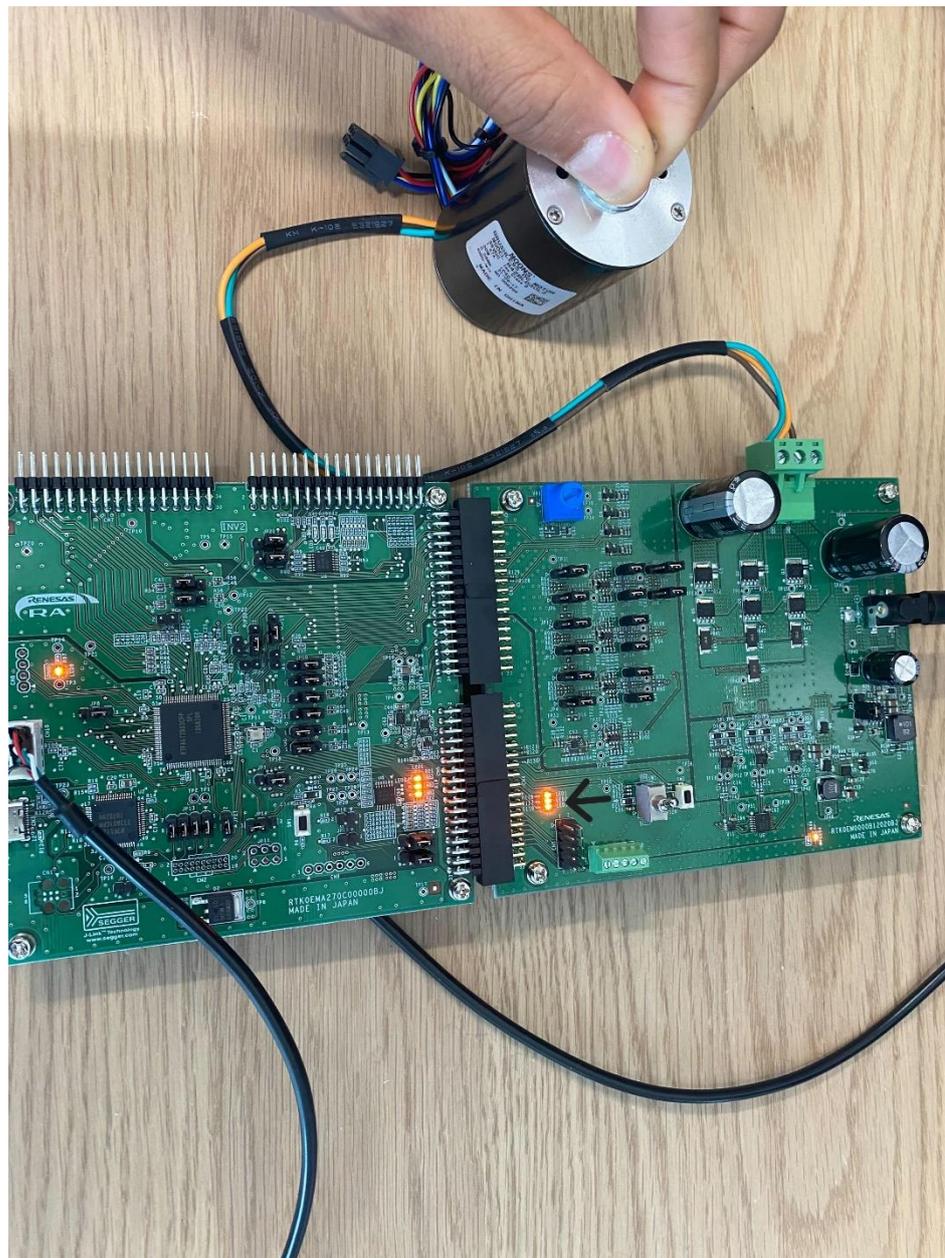
Build & debug the project as before. Click on the Resume button twice. Now the board is running on inference mode. Turn the motor switch ON and run the motor.



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.



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



What should I do if model 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.

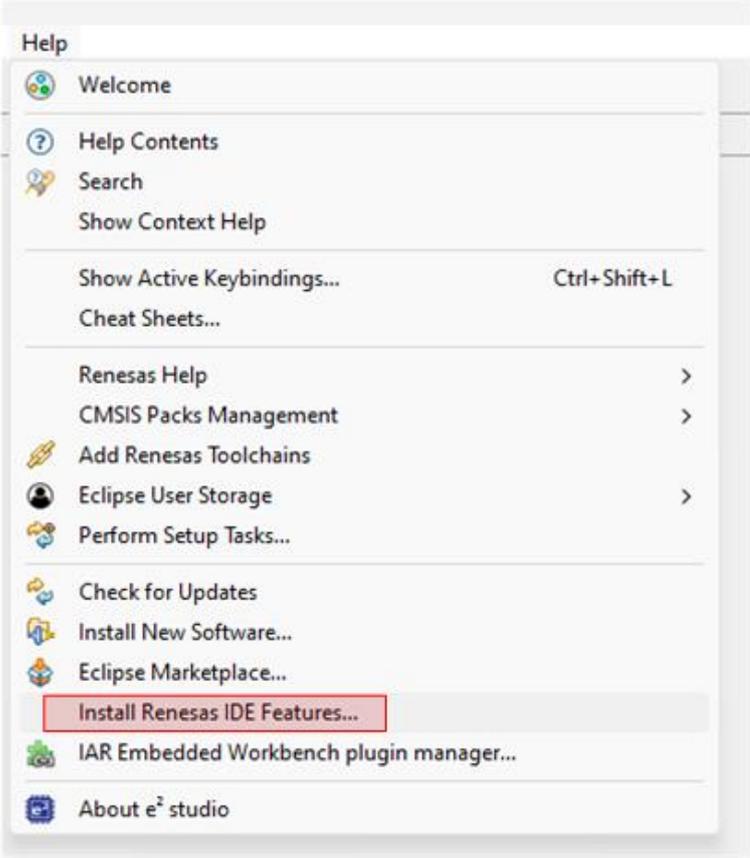
You have finished this section.

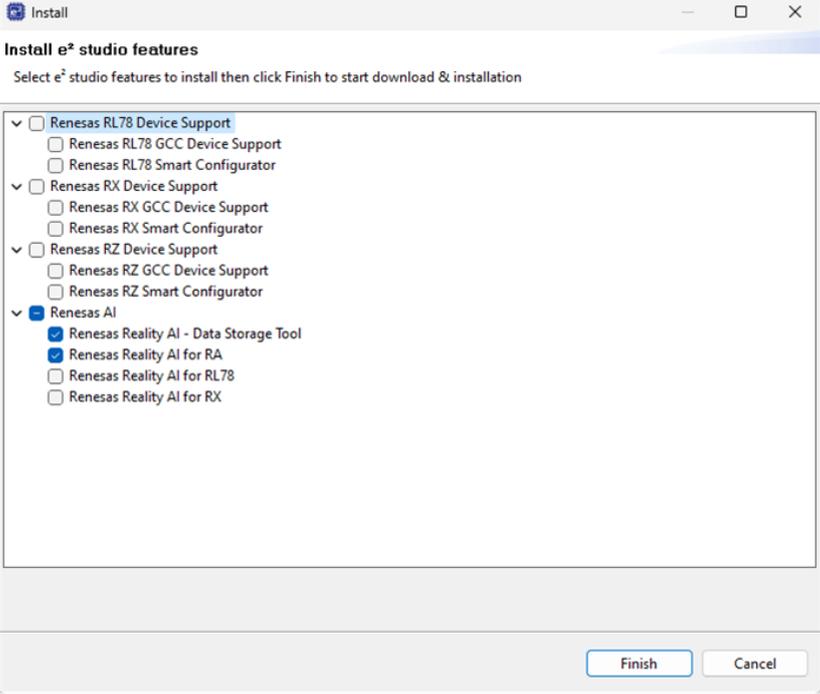
5 Appendix

Overview

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

Procedural Steps

	<p>Click on Help -> Install Renesas IDE Features...</p>  <p>The screenshot shows the 'Help' menu in the Eclipse IDE. The menu items are: Welcome, Help Contents, Search, Show Context Help, Show Active Keybindings... (Ctrl+Shift+L), Cheat Sheets..., Renesas Help, CMSIS Packs Management, Add Renesas Toolchains, Eclipse User Storage, Perform Setup Tasks..., Check for Updates, Install New Software..., Eclipse Marketplace..., Install Renesas IDE Features... (highlighted with a red box), IAR Embedded Workbench plugin manager..., and About e² studio.</p>
	<p>Select the Reality-AI features and click Finish.</p>

	 <p>Install e² studio features Select e² studio features to install then click Finish to start download & installation</p> <ul style="list-style-type: none">▼ <input type="checkbox"/> Renesas RL78 Device Support<ul style="list-style-type: none"><input type="checkbox"/> Renesas RL78 GCC Device Support<input type="checkbox"/> Renesas RL78 Smart Configurator▼ <input type="checkbox"/> Renesas RX Device Support<ul style="list-style-type: none"><input type="checkbox"/> Renesas RX GCC Device Support<input type="checkbox"/> Renesas RX Smart Configurator▼ <input type="checkbox"/> Renesas RZ Device Support<ul style="list-style-type: none"><input type="checkbox"/> Renesas RZ GCC Device Support<input type="checkbox"/> Renesas RZ Smart Configurator▼ <input checked="" type="checkbox"/> Renesas AI<ul style="list-style-type: none"><input checked="" type="checkbox"/> Renesas Reality AI - Data Storage Tool<input checked="" type="checkbox"/> Renesas Reality AI for RA<input type="checkbox"/> Renesas Reality AI for RL78<input type="checkbox"/> Renesas Reality AI for RX <p>Finish Cancel</p>
	<p>Restart e2studios when prompted.</p>
<p>You have finished this section.</p>	

**END OF LAB
THANK YOU**