

Voice Control Light Tutorial

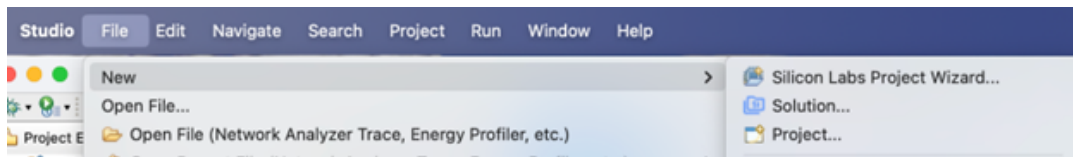
This guide details the process of creating a voice-controlled light application using TensorFlow Lite Micro (TFLM) on an EFR32xG24 Development Kit. This example uses the `keyword_spotting_on_off_v3.tflite` model (recommended) for "on" and "off" keyword detection. For more information on model creation, see the [MLTK tutorial](#).

Hardware: EFR32xG24 Dev Kit Board (BRD2601B Rev A01)

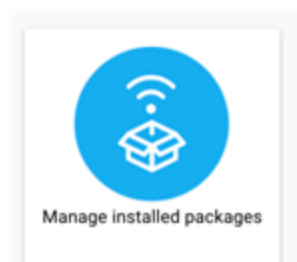
Software: Simplicity Studio (SiSDK 2024.12 or later)

1. Install AI/ML Extension

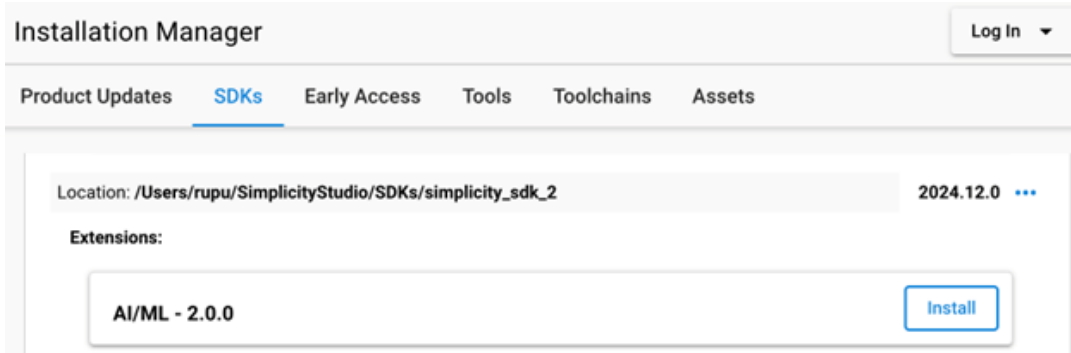
1. Click "Install" on the top bar.



2. Click "Manage Installed Packages".

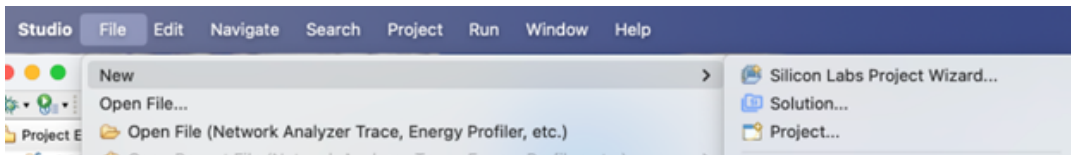


3. Under "SDKs", install the latest version of the AI/ML extension (available from SiSDK 2024.12 onwards).



2. Start a New Simplicity Project

1. From the "File" menu, select "New" >> "Silicon Labs Project Wizard".



2. Select the target board (EFR32xG24 Development Kit), SDK (Simplicity SDK v2024.12.0 or later), and IDE/Toolchain (e.g., GNU ARM v12.2.1). Click "Next".

New Project Wizard

Target, SDK, and Toolchain Selection
Select the target board, device, SDK, and IDE/toolchain to use for the project.

Target, SDK Examples Configuration

Target Boards:
Search or Select
EFR32xG24 Explorer Kit (BRD2703A Rev A01)

Target Device:
Search or Select
EFR32MG24B210F1536IM48

SDK:
Select SDK
Simplicity SDK Suite v2024.6.2: Amazon 202012.00, Bluetooth 8.2.0, Bluetooth Mesh 7.0.2, EmberZNet 8.0.2.0, Flex 3.8.2.0, Micrium O

[Manage SDKs](#)

IDE / Toolchain:
Select IDE / Toolchain
Simplicity IDE / GNU ARM v12.2.1

New Project Wizard

Target, SDK, and Toolchain Selection
Select the target board, device, SDK, and IDE/toolchain to use for the project.

Target, SDK Examples Configuration

Target Boards:
Search or Select
EFR32xG24 Dev Kit Board (BRD2601B Rev A01)

Target Device:
Search or Select
EFR32MG24B310F1536IM48

SDK:
Select SDK
Simplicity SDK Suite v2024.12.0: AI/ML 2.0.0, Amazon 202012.00, Bluetooth 9.0.0, Bluetooth Mesh 8.0.0, CCC 0.9.0.0, Connect 4.0.0.0

[Manage SDKs](#)

IDE / Toolchain:
Select IDE / Toolchain
Simplicity IDE / GNU ARM v12.2.1

3. Choose "Empty C++ Project". Click "Next".
4. Give your project a name and click "Finish".

Project Configuration
Select the project name and location.

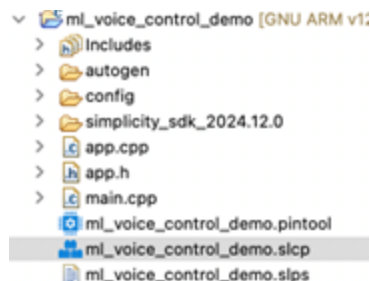
☒ Target, SDK
 ☒ Examples

Project name:

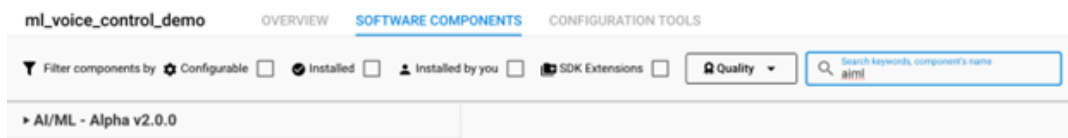
☒ Use default location
 Location:

3. Add Machine Learning Software Component

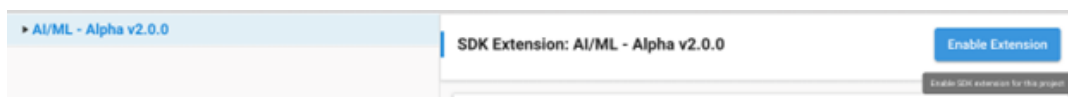
1. Open your project file (the one with the `.slcp` extension).



2. Under "Software Components", search for "aiml".



3. Enable the AI/ML extension by clicking "Enable Extension".



4. Expand: AI/ML >> Machine Learning >> TensorFlow. Select "TensorFlow Lite Micro" and click "Install".

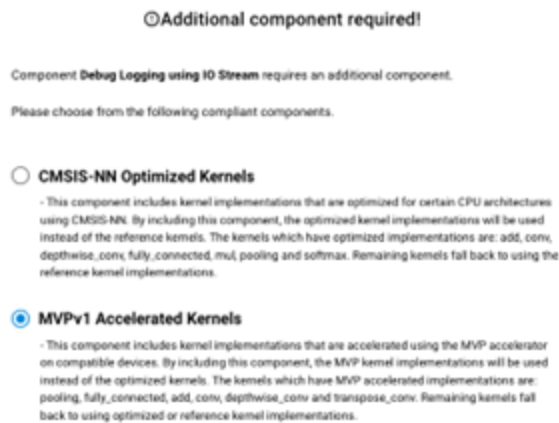


5. You'll be prompted to select additional components:

- **Debug Logging:** Choose "Debug Logging using IO Stream" (if needed) or "Debug Logging Disabled". Click "Install".

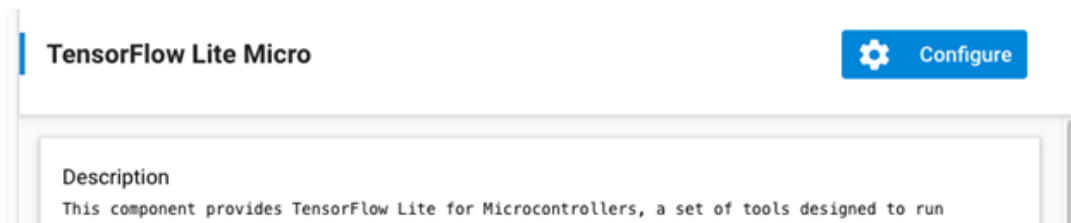


- **Kernels:** Select "MVPv1 Accelerated Kernels". Click "Install".

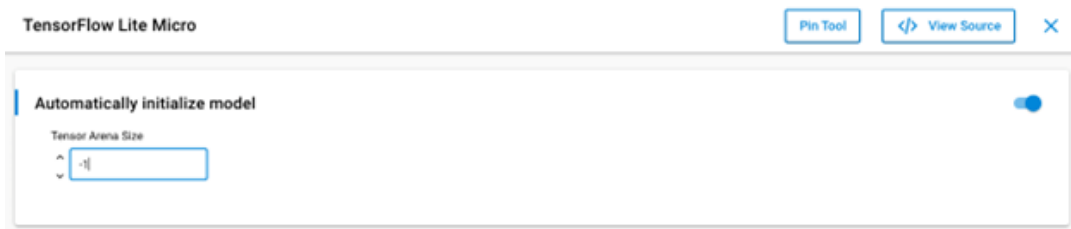


4. Configure the TFLM Component

1. Click "Configure" in the TensorFlow Lite Micro Software Component.



2. Set the "Arena Size". For this example, enter `1`. This tells the system to dynamically determine the optimal arena size at runtime.



5. Include and Convert the Model

1. Create a `tfliite` directory inside your project's `config` directory (optional but recommended).
2. Drag and drop the `keyword_spotting_on_off_v2.tflite` file into the `config/tfliite` directory (or directly into `config` if you skipped creating the subdirectory).
3. The framework will automatically convert the `.tflite` file into a C array (`sl_tflite_micro_model.c` in the `autogen` directory). The TFLM interpreter is also initialized automatically.

6. Profile the Model (Optional)

Model profiling can be helpful for optimization. For advanced users who wish to analyze model performance, the [MLTK Model Profiler Utility](#) can be used. This is not strictly required for this basic example.

7. Run the Model

1. **Include TensorFlow Init API:** Add the necessary code to initialize the TFLM interpreter.
2. **Provide Input Data:**

- Get a pointer to the input tensor: `TfLiteTensor* input = sl_tflite_micro_get_input_tensor();`
- Load your input data (microphone audio quantized to int8) into the input tensor: `input->data.int8f[0] = <input array from microphone quantized to int8>;` (See the [example code](#) for audio feature generation).

3. Run Inference:

- Invoke the interpreter: `TfLiteStatus invoke_status = sl_tflite_micro_get_interpreter()->Invoke();`
- Check for errors: `if (invoke_status != kTfLiteOk) {
TF_LITE_REPORT_ERROR(sl_tflite_micro_get_error_reporter(), "bad input tensor parameters in model"); }`

4. Read Output:

- Get a pointer to the output tensor: `TfLiteTensor* output = sl_tflite_micro_get_output_tensor();`
- Access the output data: `int8_t value = output->data.int8_tf[0];`

8. Implement Post-Processing

1. **Develop an Algorithm:** Create an algorithm to interpret the model's output (e.g., the `int8_t value`) and determine whether "on" or "off" was spoken.
2. **Trigger Events:** Based on the post-processed output, trigger actions like controlling the LED. Refer to the `voice_control_light.cc` , `recognize_commands.cc` , and `recognize_commands.h` files in the [aiml-extension examples](#) for guidance on implementing this logic, including LED control and command recognition. You will need to add components for the microphone, audio processing, and LED control to your project.