基於 Zephyr 平台的 M55M1 NPU 關鍵字偵測

NuMicro® 32 位元系列微控制器範例代碼介紹

文件資訊

應用簡述	本範例程式碼使用 M55M1 NPU，在 Zephyr 平台上示範機器學習的關鍵字偵測。
BSP 版本	M55M1_Series_BSP_CMSIS_V3.01.003 Zephyr v4.3.0
開發平台	NuMaker-X-M55M1D

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

1. 概述

本範例展示一個關鍵字偵測（KWS）推論應用，可在錄音中辨識特定關鍵字，並運行於新唐的 Ethos-U NPU 平台。其原始來源為 Arm 的 ML Embedded Evaluation Kit / Keyword Spotting 範例，該範例提供更多關於關鍵字偵測應用的資訊。

本範例主要包含：

- 透過數位麥克風（DMIC）錄製音訊片段
- 使用 TensorFlow Lite Micro 框架與 Ethos-U NPU 執行關鍵字偵測推論。

  此流程包含三個步驟：

  1. 將錄製的音訊片段前處理為推論所需的指定輸入張量格式。
  2. 執行關鍵字偵測推論。
  3. 對輸出張量進行後處理，轉換為外部可理解 / 可使用的格式。

本範例執行一個來自 Arm Model Zoo 的 DS-CNN Medium INT8 模型。接著透過 Vela 編譯器進行優化，使其可在新唐的 Ethos-U NPU 平台上執行並加速。

Vela 以 *.tflite* 檔作為輸入，並輸出另一個 *.tflite* 檔，其中由 Ethos-U 支援的運算元會被替換為 Ethos-U 的自訂運算元。在理想情況下，整個模型網路將被替換為單一的 Ethos-U 自訂運算元。

1.1 硬體功能

NuMicro® M55M1 系列是新一代 AI 微控制器，提供高達 110 GOPS 的 AI 運算效能，專為端點 AI 應用（如資料辨識與智慧音訊）所設計。M55M1 是一款 32 位元微控制器，採用 220 MHz 的 Arm® Cortex®-M55 核心，並搭載 Arm® Ethos™-U55 神經網路處理單元（NPU）作為協同處理器，以加速神經網路運算。除了卓越的 AI 效能之外，M55M1 亦提供豐富的通訊與周邊介面，包括用於音訊錄製的數位麥克風（DMIC）。Ethos-U55 NPU 與 DMIC 的結合，使其非常適合用於展示關鍵字喚醒（Keyword Spotting）推論應用。

1.2 使用 Vela 優化模型

本節說明如何使用 Vela 將 DS-CNN Medium INT8 模型優化，目標為 M55M1 的 Ethos-U NPU。

- 下載名為 *ds_cnn_m_quantized.tflite* 的 DS-CNN Medium INT8 模型。
- 使用 Vela 編譯器優化模型 *ds_cnn_m_quantized.tflite*，並取得優化後的模型 *ds_cnn_m_quantized_vela.tflite*。

```
$ vela ds_cnn_m_quantized.tflite \
--output-dir . \
--accelerator-config ethos-u55-256 \
--optimise Performance
```

- 將其重新命名為 *ds_cnn_m_quantized_ethos-u55-256_opt-speed.tflite*，以利清楚區分。

```
$ mv ds_cnn_m_quantized_vela.tflite \
ds_cnn_m_quantized_ethos-u55-256_opt-speed.tflite
```

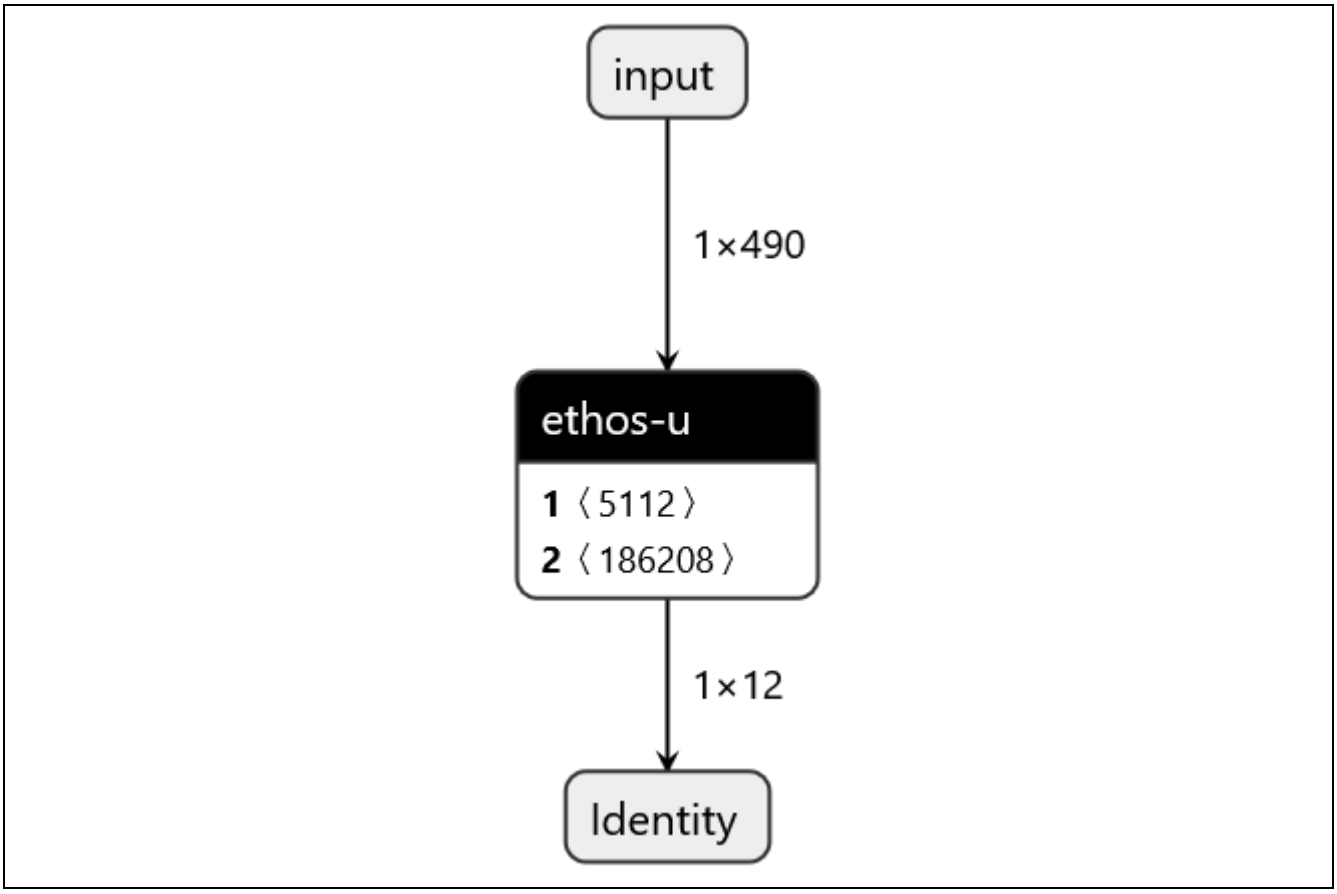

圖 1-1 優化模型

1.3 程式碼、模型與 Ethos-U 之間的綁定

範例程式、經 Vela 編譯的模型，以及 Ethos-U NPU 之間存在綁定關係。這代表除非使用者對三者都有充分了解，否則不可任意更動其中任一項。

由上述內容可知，Vela 編譯後的模型與 Ethos-U NPU 之間具有綁定：M55M1 的 Ethos-U NPU 為 Ethos-U55、每個週期 256 MAC，故 **--config** 參數必須設定為 **ethos-u55-256** 才能匹配。

範例程式必須負責推論所需的前處理與後處理，且前處理與後處理的邏輯必須符合模型網路的輸入與輸出需求。

1.4 示範結果

要示範關鍵字偵測推論，請依照「範例程式執行」章節的指示，並將主機端終端機程式設定為 **115200/8-N-1**（115200 波特率，8 個資料位元，無同位檢查，1 個停止位元）。完成 **west flash** 後，您將看到類似以下的訊息輸出：

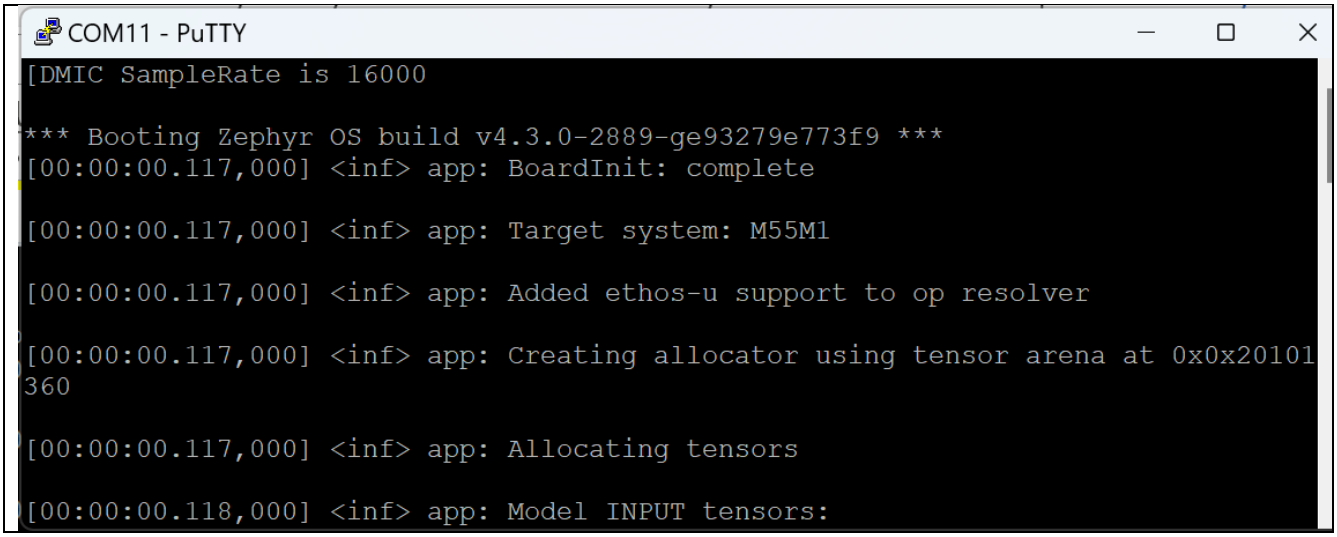

圖 1-2 終端機：初始化關鍵字偵測推論

在訊息輸出末端，系統會提示您輸入以繼續：

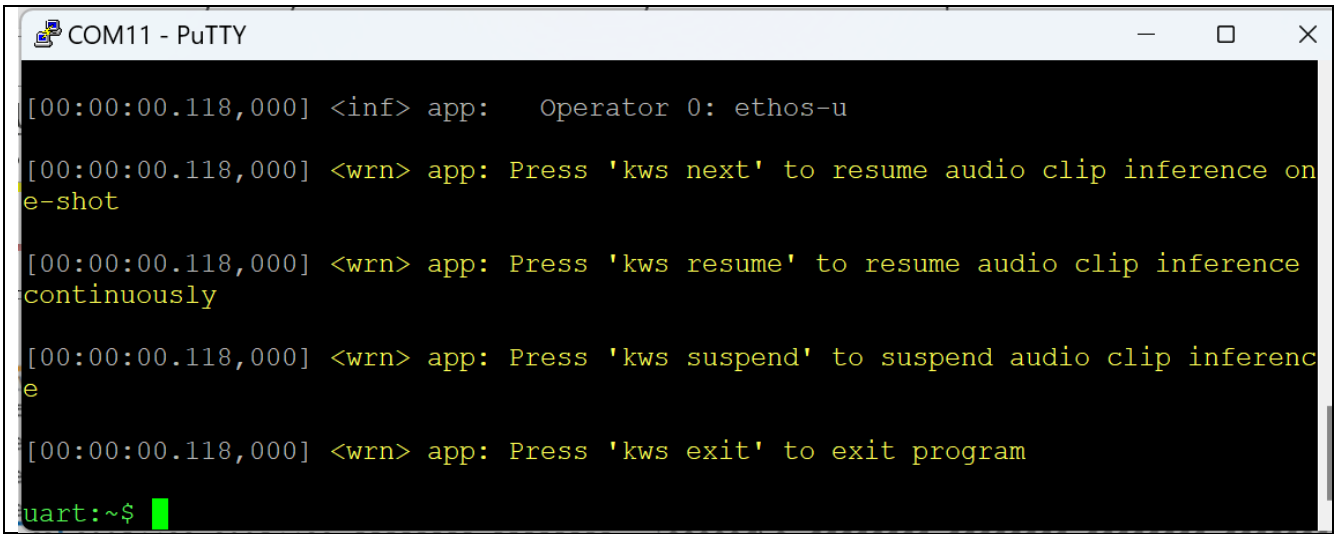

圖 1-3 終端機：關鍵字偵測 Shell 指令

請輸入：**kws resume**

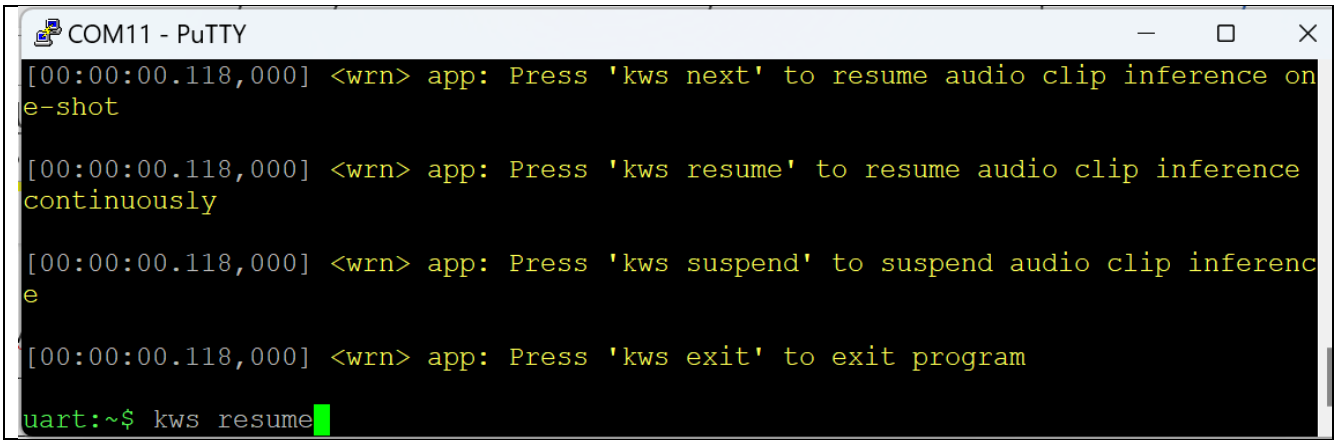

圖 1-4 終端機：開始關鍵字偵測推論

您會看到「未偵測」的訊息輸出：

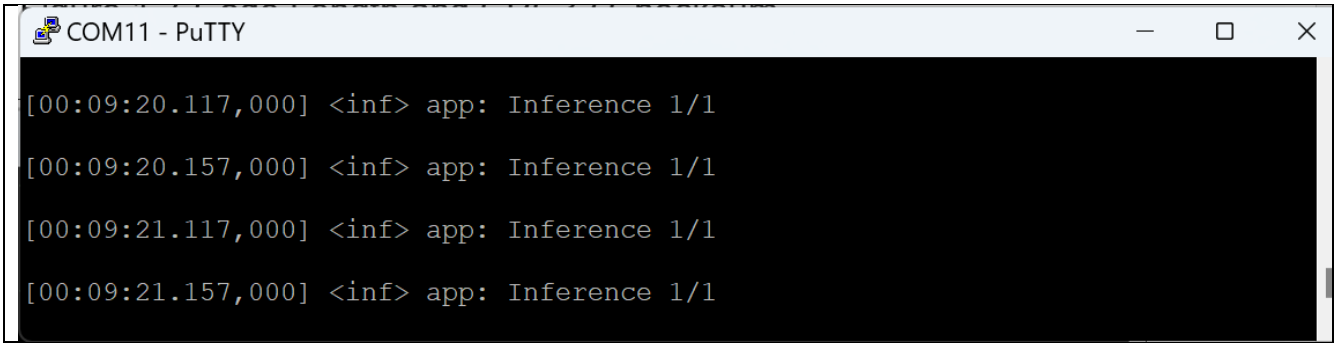

圖 1-5 終端機：未偵測到關鍵字

靠近目標板，說出已訓練的關鍵字，例如 “Go”，您將看到「已偵測」的訊息輸出：

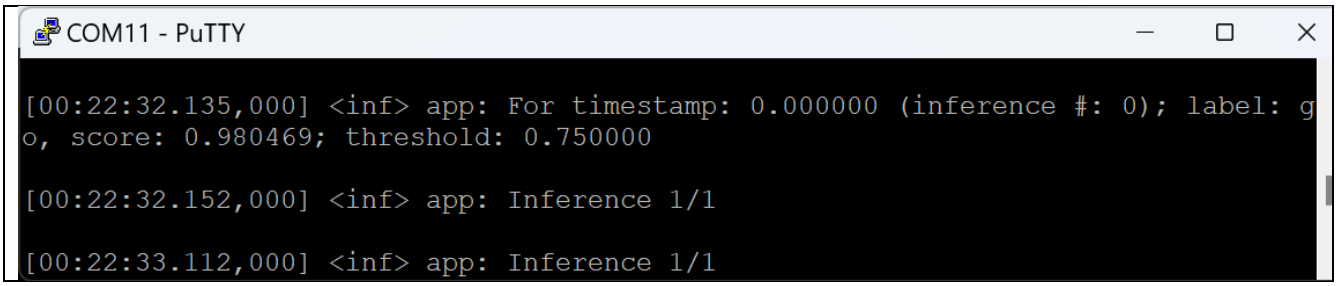

圖 1-6 終端機：偵測到關鍵字「Go」

關鍵字偵測模型所訓練的關鍵字列於檔案 *ModelLabels.cpp*。

2. 代碼介紹

在 Zephyr 上，使用者不需要明確撰寫常見的初始化程式碼（例如系統時鐘與 UART），因為 Zephyr 的初始化流程會在 **main** 函式之前先完成這些工作。本範例中的 Ethos-U 也會在該流程中初始化。不過，對於 Zephyr 未內建支援的其他周邊，使用者仍需手動初始化，並在 **BoardInit** 中完成。

```
int main()
{
    /* Board specific initialization */
    BoardInit();
```

在 **BoardInit** 中，為本範例初始化數位麥克風（DMIC）。

```
    // DMIC_APLL1_FREQ_196608KHZ for DMIC 8000/16000/48000 Hz sample-rate with 64/128/256
down-sample
    CLK_EnableAPLL(CLK_APLLCTL_APLLSRC_HIRC, DMIC_APLL1_FREQ_196608KHZ, CLK_APLL1_SELECT);
    // Select DMIC CLK source from APLL1_DIV2.
    CLK_SetModuleClock(DMIC0_MODULE, CLK_DMICSEL_DMIC0SEL_APLL1_DIV2, MODULE_NoMsk);
    // Enable DMIC clock.
    CLK_EnableModuleClock(DMIC0_MODULE);
    // DPWM IPReset.
    SYS_ResetModule(SYS_DMIC0RST);

    // LPPDMA Initial.
    CLK_EnableModuleClock(LPPDMA0_MODULE);
    CLK_EnableModuleClock(LPSRAM0_MODULE);
    SYS_ResetModule(SYS_LPPDMA0RST);

    /* Set multi-function pins for DMIC */
    SET_DMIC0_CLK_PB4();
    SET_DMIC0_DAT_PB5();
```

在 **main** 中，於 **BoardInit** 之後，執行 TensorFlow Lite Micro 的初始化流程，包括準備關鍵字偵測模型。

```
    /* Model object creation and initialization. */
    arm::app::KwsModel model;

    if (!model.Init(arm::app::tensorArena,
                    sizeof(arm::app::tensorArena),
                    arm::app::kws::GetModelPointer(),
                    arm::app::kws::GetModelLen()))
    {
        printf_err("Failed to initialise model\n");
        return EXIT_FAILURE;
    }

    constexpr int minTensorDims = static_cast<int>(
```

```
                                        (arm::app::KwsModel::ms_inputRowsIdx >
arm::app::KwsModel::ms_inputColsIdx)
                                        ? arm::app::KwsModel::ms_inputRowsIdx
                                        : arm::app::KwsModel::ms_inputColsIdx);

    const auto mfccFrameLength = 640;
    const auto mfccFrameStride = 320;
    const auto scoreThreshold  = 0.75;

    /* Get Input and Output tensors for pre/post processing. */
    TfLiteTensor *inputTensor  = model.GetInputTensor(0);
    TfLiteTensor *outputTensor = model.GetOutputTensor(0);

    if (!inputTensor->dims)
    {
        printf_err("Invalid input tensor dims\n");
        return EXIT_FAILURE;
    }
    else if (inputTensor->dims->size < minTensorDims)
    {
        printf_err("Input tensor dimension should be >= %d\n", minTensorDims);
        return EXIT_FAILURE;
    }

    /* Get input shape for feature extraction. */
    __maybe_unused TfLiteIntArray *inputShape     = model.GetInputShape(0);
#if defined (MODEL_DS_CNN)
    const uint32_t numMfccFeatures = 10;
    const uint32_t numMfccFrames = 49;
#else
#error "No model specified"
#endif

    /* We expect to be sampling 1 second worth of data at a time.
     * NOTE: This is only used for time stamp calculation. */
    const float secondsPerSample = 1.0 /
arm::app::audio::MicroNetKwsMFCC::ms_defaultSamplingFreq;

    /* Classifier object for results */
    arm::app::KwsClassifier classifier;

    /* Object to hold label strings. */
    std::vector<std::string> labels;

    /* Populate the labels here. */
    GetLabelsVector(labels);
```

記憶體保護單元配置（MPU）設定已在 Zephyr 的初始化流程中於 **main** 之前完成。

```
    /* Set up MPU configuration
     *
     * On zephyr, configure mpu region for tensor arena in zephyr way
     * rather than direct control
     */
```

開始 DMIC 錄音。

```
    ret = DMICRecord_Init(AUDIO_SAMPLE_RATE, AUDIO_CHANNEL, audioSlidingSamples,
AUDIO_SAMPLE_BLOCK);
    if (ret)
    {
        printf_err("Unable init DMIC record error(%d)\n", ret);
        delete []audioSlidingBuf;
        return EXIT_FAILURE;
    }
    DMICRecord_StartRec();
```

在 Zephyr 上，使用自訂的 shell 指令來控制關鍵字偵測的錄音。

```
    warn("Press 'kws next' to resume audio clip inference one-shot\n");
    warn("Press 'kws resume' to resume audio clip inference continuously\n");
    warn("Press 'kws suspend' to suspend audio clip inference\n");
    warn("Press 'kws exit' to exit program\n");
```

接著進入 super while 迴圈。

```
    while (1)
    {
        /* … */
    }
```

執行訊息迴圈。這將擷取如同上述的「kws next」等 Shell 自訂指令，並加以處理。

```
        k_mutex_lock(&arm::app::kws::mutex_record_ctrl, K_FOREVER);
        if (arm::app::kws::record_ctrl_end)
        {
            k_mutex_unlock(&arm::app::kws::mutex_record_ctrl);
            warn("Bye!\n");
            break;
        }
        else if (!arm::app::kws::record_ctrl_cont && !arm::app::kws::record_ctrl_oneshot)
        {
            k_condvar_wait(&arm::app::kws::condvar_record_ctrl,
&arm::app::kws::mutex_record_ctrl, K_FOREVER);
            k_mutex_unlock(&arm::app::kws::mutex_record_ctrl);
            continue;
        }
        else if (arm::app::kws::record_ctrl_cont)
        {
            __ASSERT_NO_MSG(!arm::app::kws::record_ctrl_oneshot);
        }
        else if (arm::app::kws::record_ctrl_oneshot)
        {
            __ASSERT_NO_MSG(!arm::app::kws::record_ctrl_cont);
            arm::app::kws::record_ctrl_oneshot = false;
        }
        k_mutex_unlock(&arm::app::kws::mutex_record_ctrl);
```

執行 DMIC 錄音。

```
        while (DMICRecord_AvailSamples() < audioSlidingSamples)
        {
            /*
             * On zephyr, other threads will starve with busy-wait here,
             * for example, logging thread won't work with lower priority.
             * Change to non-busy-wait.
             */
            DMICRecord_SleepUntilUpdate(200);
        }

        DMICRecord_ReadSamples(audioSlidingBuf, audioSlidingSamples);

        DMICRecord_UpdateReadSampleIndex(audioStrideSamples);
```

以及前處理、推論與後處理。

```
            if (!preProcess.DoPreProcess(inferenceWindow, audioDataSlider.Index()))
            {
                printf_err("Pre-processing failed.");
                break;
            }

            if (!model.RunInference())
            {
                printf_err("Inference failed.");
                break;
            }

            if (!postProcess.DoPostProcess())
            {
                printf_err("Post-processing failed.");
                break;
            }
```

最後，顯示推論結果。

```
                for (uint32_t j = 0; j < result.m_resultVec.size(); ++j)
                {
                    info("For timestamp: %f (inference #: %" PRIu32
                         "); label: %s, score: %f; threshold: %f\n",
                         (double)result.m_timeStamp,
                         result.m_inferenceNumber,
                         result.m_resultVec[j].m_label.c_str(),
                         result.m_resultVec[j].m_normalisedVal,
                         (double)result.m_threshold);
                }
```

3. 軟體與硬體需求

3.1 軟體需求

- 主機作業系統：Windows 10 64 位元或更新版本。

  多數新唐 Cortex-M 系列 MCU 的使用者在 Windows 上開發，因此本文件以該環境為主。

  本文件中的指令列已在 Windows 的 Git Bash 環境驗證過。若使用其他 Shell，請確認各 Shell 在行續接、引號與跳脫字元的用法差異。對於 Bash，行續接符號為「\」。

- Python（3.10 或之後版本）

  請單獨安裝此工具，或依照《Developing with Zephyr Getting-Started Guide》進行安裝。

- Git

  請單獨安裝此工具，或依照《Developing with Zephyr Getting-Started Guide》進行安裝。

  本文件偏好使用 Git Bash 作為命令列環境。

- CMake（3.20.5 或之後版本）與 Ninja

  請分別安裝這些工具，或依照《Developing with Zephyr Getting-Started Guide》進行安裝。

- Zephyr development environment

  在上述工具安裝完成後，請繼續依照以下說明建立配合範例程式碼的 Zephyr 開發環境。

  建立一個名為 ***zephyrproject*** 的目錄，作為 west 工作區：

```
$ cd ~
$ mkdir zephyrproject
```

  建立新的 Python 虛擬環境：

```
$ python -m venv zephyrproject/.venv
```

  啟用 Python 虛擬環境：

```
$ source zephyrproject/.venv/Scripts/activate
```

安裝 pip-system-certs 以解決可能發生的憑證驗證失敗問題：

```
$ pip install --upgrade pip-system-certs
```

安裝 west：

```
$ pip install --upgrade west
```

初始化名為 *zephyrproject* 的 west 工作區。這個步驟會下載 *Zephyr* 原始碼：

```
$ west init zephyrproject
```

將範例程式碼的 *ZIP* 檔解壓縮為 *EC_M55M1_Zephyr_ML_KWS*，放到 *zephyrproject* 目錄中，其中已移除版本號部分，以便所示說明不受版本影響：

```
$ unzip -d zephyrproject EC_M55M1_Zephyr_ML_KWS_V1.00.zip
$ mv zephyrproject/EC_M55M1_Zephyr_ML_KWS_V1.00 zephyrproject/EC_M55M1_Zephyr_ML_KWS
```

根據範例程式碼的 west manifest 重新初始化名為 *zephyrproject* 的 west 工作區。這樣後續的 west update 指令就不會拉入大量不必要的模組：

```
$ cd zephyrproject
$ rm -rdf .west
$ west init -l EC_M55M1_Zephyr_ML_KWS
$ west update
```

安裝 Python 相依套件，其中包含 pyOCD 工具：

```
$ west packages pip --install
```

安裝 Zephyr SDK 工具鏈 arm-zephyr-eabi：

```
$ west sdk install \
--install-base toolchains \
--toolchains arm-zephyr-eabi \
--no-hosttools
```

- GCC 交叉編譯器

請依照上述步驟安裝此工具。

請使用 Zephyr SDK 工具鏈，取代 Arm GNU Toolchain，以避免因工具鏈差異導致的建置失敗。

- pyOCD

請依照上述步驟安裝此工具。

Zephyr 會呼叫 pyOCD 來下載已建置的二進位檔。

PyOCD 目前尚未內建支援 M55M1 系列 MCU，請透過 Nuvoton 的 CMSIS Pack 新增支援：

1. 下載 *Nuvoton.NuMicroM55_DFP.x.y.z.pack*，其中 **x.y.z** 為最新版本。
2. 在名為 ***zephyrproject*** 的 west 工作區目錄下，建立一個名為 *pyocd.yaml* 的 pyOCD YAML 檔，其內容如下：

```
pack:
  - C:\Users\<USER>\Downloads\Nuvoton.NuMicroM55_DFP.<x.y.z>.pack
```

- Vela 編譯器（PyPI 套件）（可選）

使用 pip 安裝此工具：

```
$ pip install --upgrade ethos-u-vela
```

在所有工具安裝完成後，離開 Python 虛擬環境：

```
$ deactivate
```

3.2 硬體需求

電路元件

- NuMaker-X-M55M1D 開發板

- 線路示意圖

- MEMS 數位麥克風

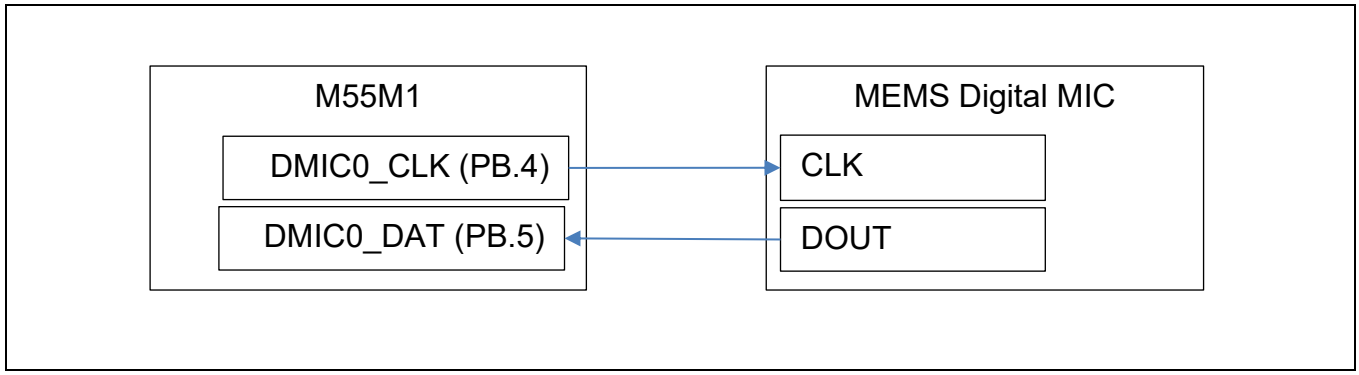

圖 3-1 MEMS 數位麥克風

4. 目錄資訊

目錄結構如下圖所示。

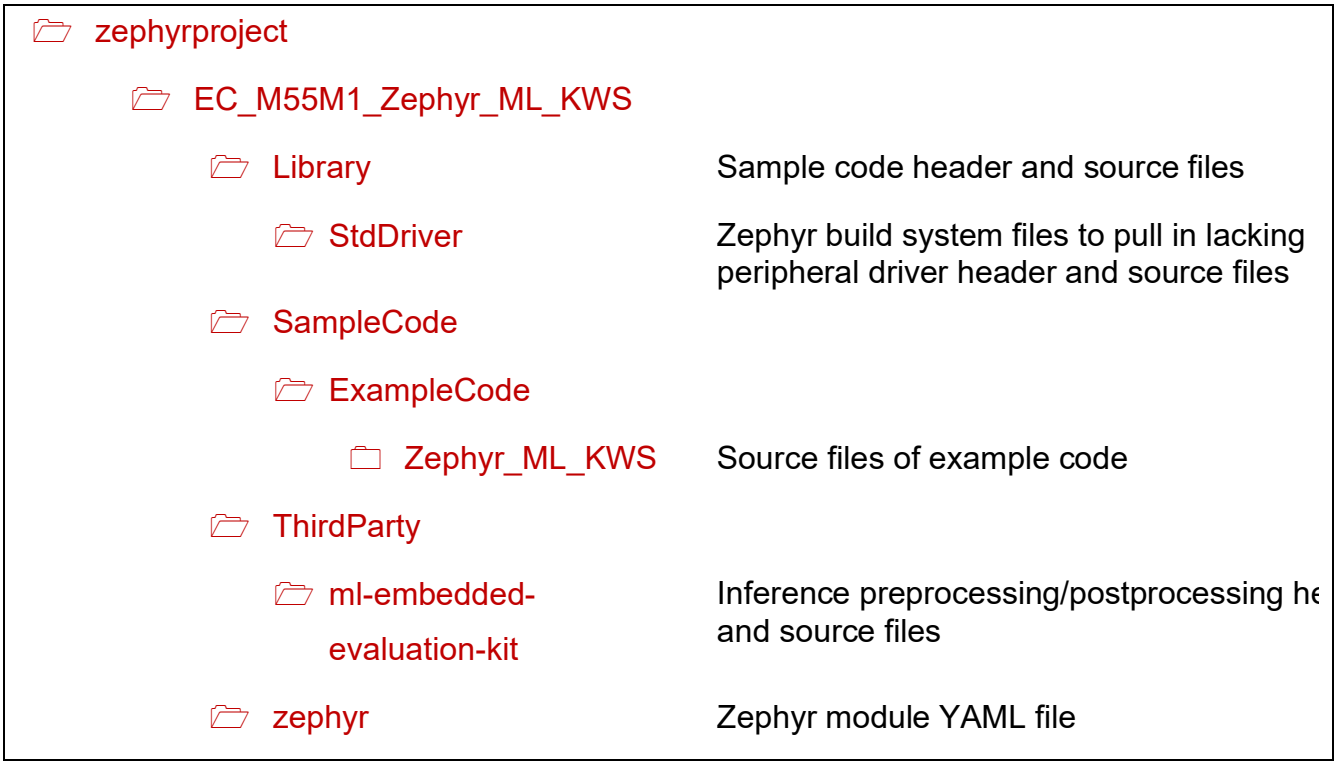

圖 4-1 目錄資訊

5. 範例程式執行

1. 每次進行 Zephyr 開發時，請先進入 Python 虛擬環境：

```
$ cd ~
$ source zephyrproject/.venv/Scripts/activate
```

2. 進入名為 *zephyrproject* 的 west 工作區目錄：

```
$ cd zephyrproject
```

3. 建置範例程式：

```
$ west -v build \
-b numaker_m55m1 \
EC_M55M1_Zephyr_ML_KWS/SampleCode/ExampleCode/Zephyr_ML_KWS
```

4. 下載已建置的二進位檔：

```
$ west flash
```

5. 在結束 Zephyr 開發時，離開 Python 虛擬環境：

```
$ deactivate
```

6. 修訂紀錄

Date	Revision	Description
2026.01.02	1.00	初始發布。

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.