NanoEdge AI Studio to STM32CubeIDE app that recieves Anomaly Detection input via COM Port

No Physical Sensors Required
Presenter Stephen MacKenzie

Resources

- Download NanoEdge AI Studio:
  - https://stm32ai.st.com/download-nanoedgeai/
- ST Electronics NanoEdge AI Studio docs and tutorials:
  - https://wiki.st.com/stm32mcu/wiki/Category:NanoEdgeAI
- Anomaly Detection YouTube from ST:
  - https://www.youtube.com/watch?v=yXMUv_C5FGk
- My GibHub (links to the project(s), see last slide)
  - https://github.com/stevemac321

Overview – The Goal

- Build and demonstrate a functional anomaly detection pipeline with:
  - No physical sensor input
  - Simple pattern signal files, spaces and comma delim.
  - An STM32 board with ST-Link over USB
  - Core logic spanning NanoEdge AI Studio → Embedded App → COM port client

GitHub Project Structure

- anomaly_spaces_for_nanoedge.csv,regular_spaces_for_nanoedge.csv for NanoEdge AI Studio signal input.
  - Anomaly.csv 5x140, regular.csv 20x140 for client-controller runtime.
- Embedded STM32CubeIDE project
  - USB STLink on COM Port
  - USART enabled, interrupt enabled
- NanoEdge AI Studio benchmarked algorithm, lib and header
  - Already “deployed” into Core/Inc and Core/lib into the STMCubeIDE project
- Python clients, console and option QT PySide6.

Recommended WorkFlow for Portability

- Clone the NanoEdge_Anomaly_Detection project from:
  - https://github.com/stevemac321/NanoEdgeAI_Anomaly_Detection
  - This gives you access to main.c and stm32xxxx_it interrupt code
  - Signal files, you skip the NanoEdge AI Studio if you want and just get the header and lib from Core/Inc and Core/Lib
- Create New STM32CubeIDE project for your board (unless you have a NUCLEO-H723ZG)
  - Enable USART (I use usart3 or whatever shares STLK_VCP_RX/TX)
  - Make sure baud rate matches your STLink COM port Enable interrupts,on windows open Device Manager (see readme for Linux)

NanoEdge AI Studio — Creating the Anomaly Detection Model

-
- New Project → Anomaly Detection
- Import signals
  - regular_spaces_for_nanoedge.csv
  - anomaly_spaces_for_nanoedge.csv
- Run Benchmark to evaluate candidate models
- Validation tab to choose best lib
- Deploy, creates emulator, lib, and header(s)

Deploying the zipfile to the STMCube32IDE app

- Create STM32CubeIDE app for your board
- STM32CubeIDE app
- Extract zip
  - Copy NanoEdgeAI.h to STMCubeIDE Project: Core/Inc
  - Copy libneai.a to Core/lib
- Add libpath to Core/lib in project properties:
  - C/C++ General → Library Paths (add /<projectdirectory>/Core/lib)
  - C/C++ General → Libraries (add neai, not libneai and no file extension .a)

Adding Training and Detection Code to the STM32CubeIDE embedded app

- You can just copy the cloned main.c into your project or these steps:
- In USER CODE BEGIN Includes section of main.c
  - Include headers: NanoEdgeAI.h, string.h (for memcpy), optional stdio.h for printf (you will need a _write function, its in the cloned main.c.
- In USER CODE BEGIN PM, #define DATA_INPUT_USER 140
- In USER CODE BEGIN PV:
  - static float EKG_Input[DATA_INPUT_USER];// pass to model
  - static uint8_t rx_chunk[4]; // Temporary buffer to hold 4 bytes of one float
  - static uint16_t ekg_index = 0; // Index into EKG_Input
  - extern float training_data[][DATA_INPUT_USER];
  - extern size_t training_data_len;

Adding Code in main.c (continued)

- In USER CODE BEGIN PFP add:
  - enum neai_state train_model();
- In USER CODE BEGIN WHILE add: (see cloned main.c)
  - if (train_model() != NEAI_MINIMAL_RECOMMENDED_LEARNING_DONE) {
  - HAL_UART_Abort_IT(&huart3);
  - __HAL_UART_DISABLE_IT(&huart3, UART_IT_RXNE);
  - } else {
  - // Start UART receive in interrupt mode
  - // Request the next 4 bytes (one float)
  - HAL_StatusTypeDef hal_ret = HAL_UART_Receive_IT(&huart3, rx_chunk, sizeof(rx_chunk));
  -
    - if(hal_ret != HAL_OK) {
    - //printf("receiveIT failed %d", hal_ret);
    - return 0;
  - }
  - }
  -

Adding Code in main.c (more continued)

- In USER CODE BEGIN PFP add:
  - enum neai_state train_model();
- In USER CODE BEGIN WHILE add: the code cloned main.c that calls train_model
- In USER CODE BEGIN 4 (or whatever), add implementations:
  - _write (optional if you want printf
  - RxCpltCallback
  - train_model

Adding Interrupt Code

- Open the cloned Core/Src/stm32h7xx_it.c file, find:
  - USART3_IRQHandler
    - Make sure that HAL_UART_IRQHandler is called in your generated stm32xxxx_it.c file.
  - Add this the code in USER CODE BEGIN USART3_IRQn 1:
    - HAL_UART_RxCpltCallback into your stm32xxxx_it.c file.

Build, Debug

- Build the STM32CubeIDE project
- Debug (will flash your board)
- Once it is flashed, it will run, debugging is optional
- To send the board data, on your host run: “python EKG_Simulator.py
- If you want to debug, set a breakpoint in HAL_UART_RxCpltCallback, inspect:
  - Status (see NanoEdgeAI.h for enum values)
  - Simularity (100 is “most regular”, 0 is “most anomaly”
  - EKG_Input is the float array buffer from uint8_t rx_chunk from HAL_UART_Receive_IT, check rx_chunk too.

EKG_Simulator Results

- Return bytes
- Neai status
- Simularity (100 normal, 0 anomaly)
- There are just two samples hardcoded
- There is another project you can clone for a bit more variety:
  - https://github.com/stevemac321/NanoEdge_Client
  - This is a QT PySide6 app (see the readme in that project)

Links

- This project enlistment:
  - https://github.com/stevemac321/NanoEdgeAI_Anomaly_Detection
- QT PySide6 Client enlistment:
  - https://github.com/stevemac321/NanoEdge_Client
- About Me:
  - https://www.linkedin.com/in/stevemac321/
- My Music and Technology YouTube Channel:
  - https://www.youtube.com/@stephenmackenzie6782