Embedded Machine Learning for Edge Computing

Introduction to STM32 microcontrollers

Kithmin Wickremasinghe

About Myself

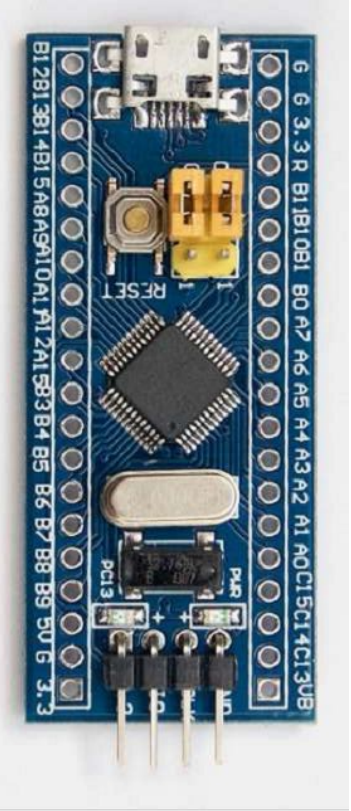
- MASc Student in Electrical and Comp. Eng., UBC, Canada
- B.Sc. in Electronic & Telecom. Eng. from the University of Moratuwa, Sri Lanka (2020)
- Former Intern at Ansys Canada Ltd.
- Former R&D Intern at Telexistence Inc. (Tokyo, Japan)
- www.kithminrw.com

Learning outcomes

- Introduction to STM32.
- How to make a simple LED blink with STM32 and simulating in Wokwi.
- Advanced topics with STM32Cube.
  - STM32Cube Ecosystem
  - STM32CubeMX keysteps
- Overview of STM32 CUBE.AI.
- Case studies with STM32 CUBE.AI.

Why STM32 for Embedded AI

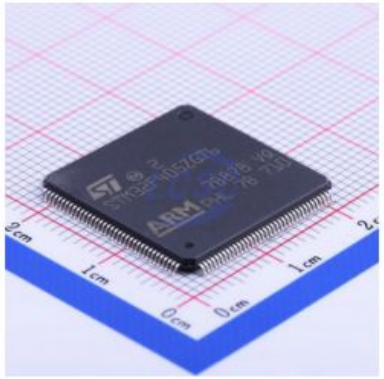
		Uno	STM32 Blue Pill
General	Dimensions	2.08" x 0.9"	4" x 2.1"
	Pricing	$20-23	$3-5
Connectivity	I/O Pins	14	37
	PWM Pins	6	15
	Analog In Pins	6	10
	Analog Out Pins (DAC)	-	-
Computing	Processor	ATMega328P	STM32F103C8T6
	Flash Memory	32 kB	64 kB
	SRAM	2 kB	20 kB
	EEPROM	1 kB	-
	Clock speed	16 MHz	72 MHz
	Voltage Level	5V	3.3V
	USB Connectivity	Standard A/B USB	Micro-USB
Communication	Hardware Serial Ports	1	3
	SPI Support	Yes (1x)	Yes (2x)
	CAN Support	No	Yes
	I2C Support	Yes (1x)	Yes (2x)



- Easy to learn
- Ready made libraries
- Huge community
- Resource availability

- Complexity of initial setup
- Higher costs for purchasing
- Need to write code from scratch.

Why STM32 for Embedded AI



Images are for reference only

Add to Favourites

STMicroelectronics STM32F405ZGT6

Manufacturer	STMicroelectronics
Mfr. Part #	STM32F405ZGT6
LCSC Part #	C92114
Package	LQFP-144_20x20x05P
Customer #	Customer Number
Datasheet	STMicroelectronics STM32F405ZGT6
EasyEDA	EasyEDA Model
Description	192KB 1MB 1.8V~3.6V -40°C~+85°C ARM Cortex-M4 1@x24ch/12bit 1@x2ch/12bit

In Stock: 61

Stock Notification

61 Can Ship in 3 Days

Inquire for More Quantity

1 + −

Add to Cart

Minimum : 1 Multiples : 1

Unit Price	US$ 15.7341
Ext. Price	US$ 15.73

Click to expand

Arduino Nano 33 BLE Sense

Code: ABX00031 / Barcode: 7630049201507

$40.50

Quantity 1

SOLD OUT

ADD TO WISHLIST

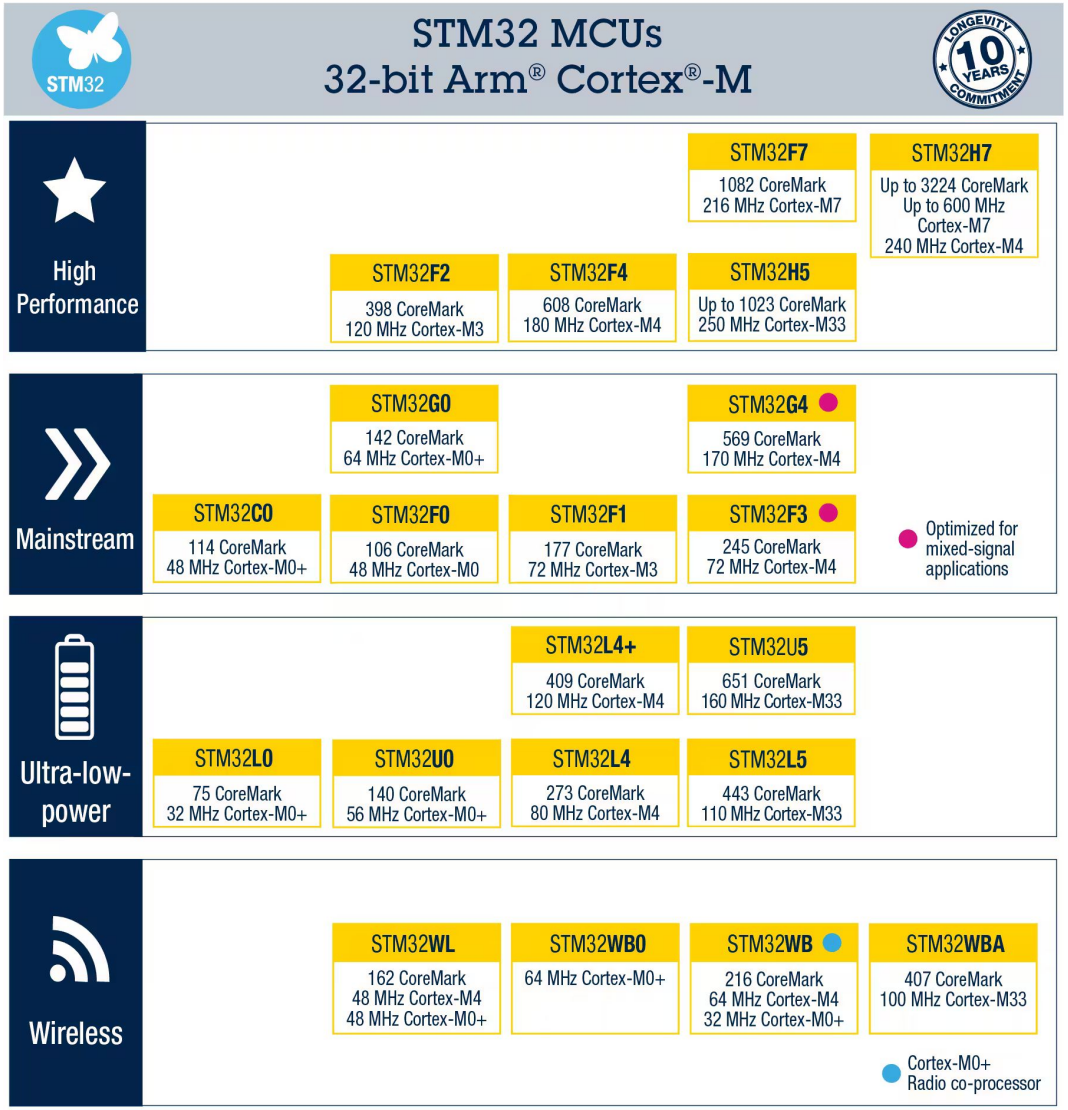
Bring the power of AI to your pocket with Arduino's tiniest form factor.

Enhance your experience, add to your cart:	**$0.00**
☐ Nano Screw terminal adapter (3 boards pack)	$35.97
☐ Arduino MKR IoT Bundle	$92.60
☐ Arduino MKR RGB Shield	$53.90
☐ Gravity: 27 Pcs Sensor Set for Arduino	$101.40
☐ USB Cable Type A Male to Micro Type B Male	$6.90

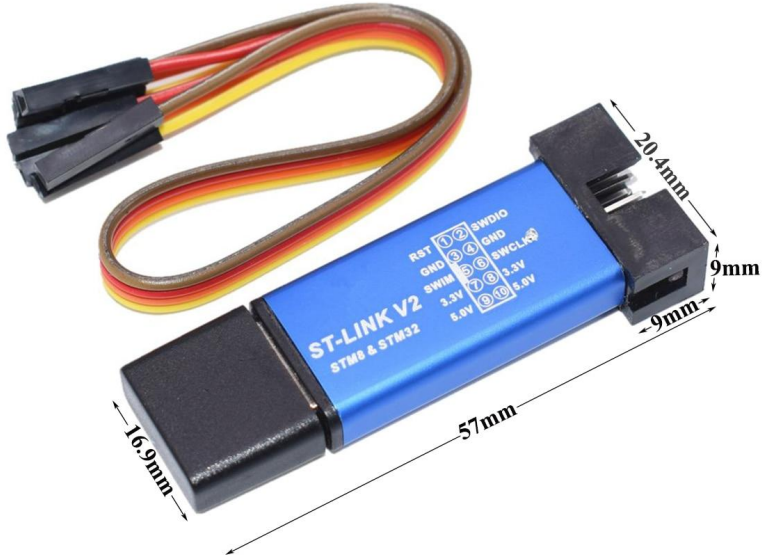
Share this:

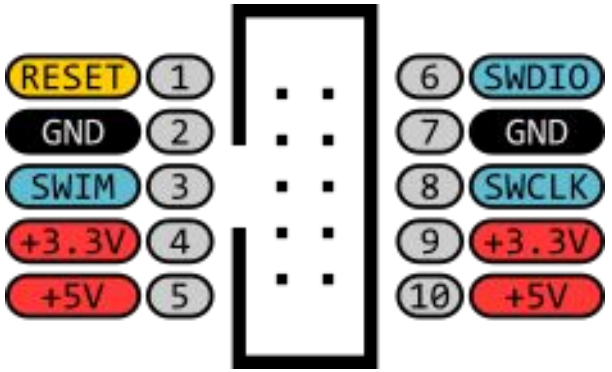
Basic setup with STM32 MCU

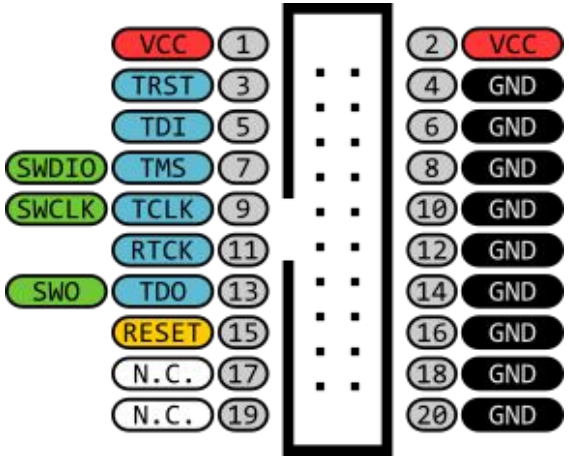
- Things to consider,
  - STM32 is a 3.3V MCU,
    - Supply should be 3.3V.
    - There are 5V tolerant pins therefore be careful only to use them for 5V.
  - Have an ST-link programmer,
    - Can debug and program.
    - Can flash the MCU with UART but no debugging is available in that approach.
  - Select proper memory with Boot 0.
  - Select clock source and frequency.
  - Select peripherals to activate.
  - Select middleware.
  - Write C++ code w/ HAL library from scratch



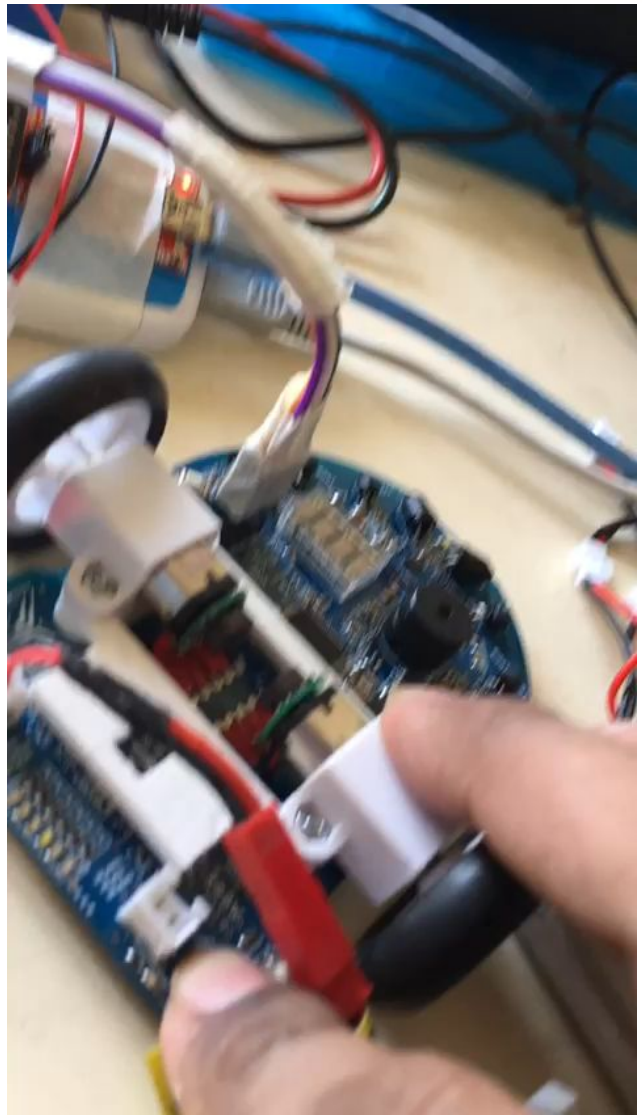
STM32 Microcontroller Programmers





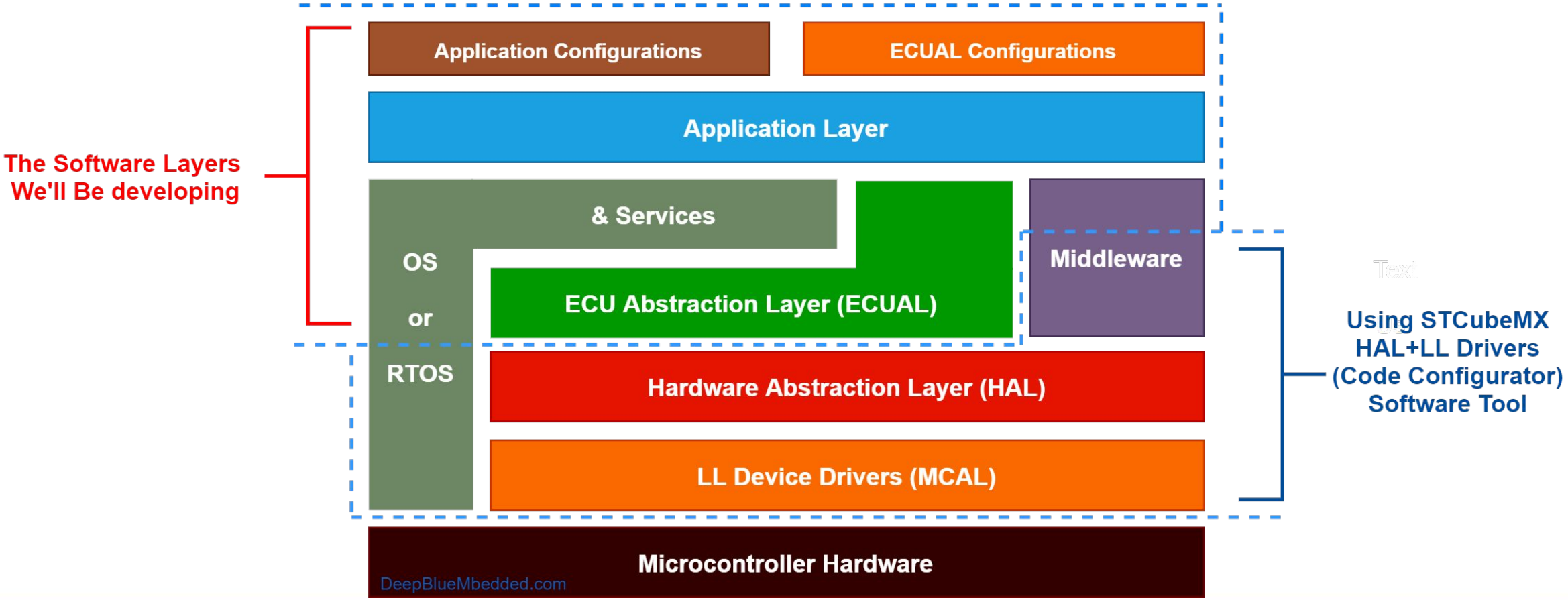


https://stm32-base.org/guides/connecting-your-debugger.html



HAL

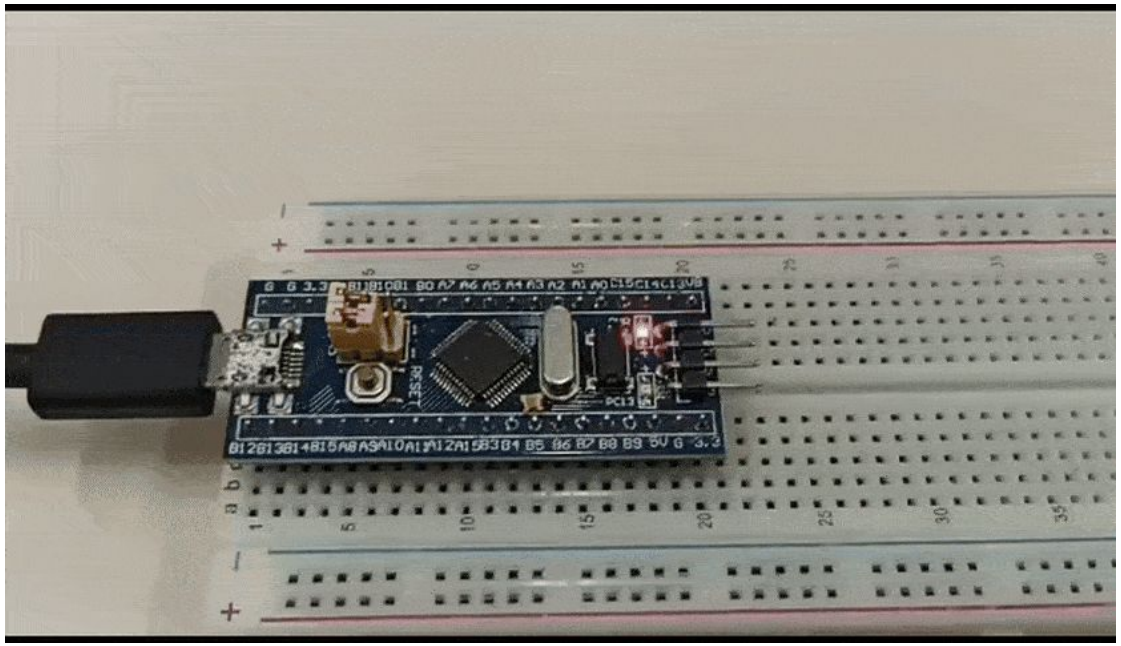
Hardware abstraction layer (HAL) is a layer of programming in an operating system that allows to interact with hardware at a general or abstract level rather than a detailed hardware level. **Hardware abstractions** are sets of routines in software that provide programs with access to hardware resources through programming interfaces. The programming interface allows all devices in a particular class C of hardware devices to be accessed through identical interfaces even though C may contain different subclasses of devices that each provide a different hardware interface.

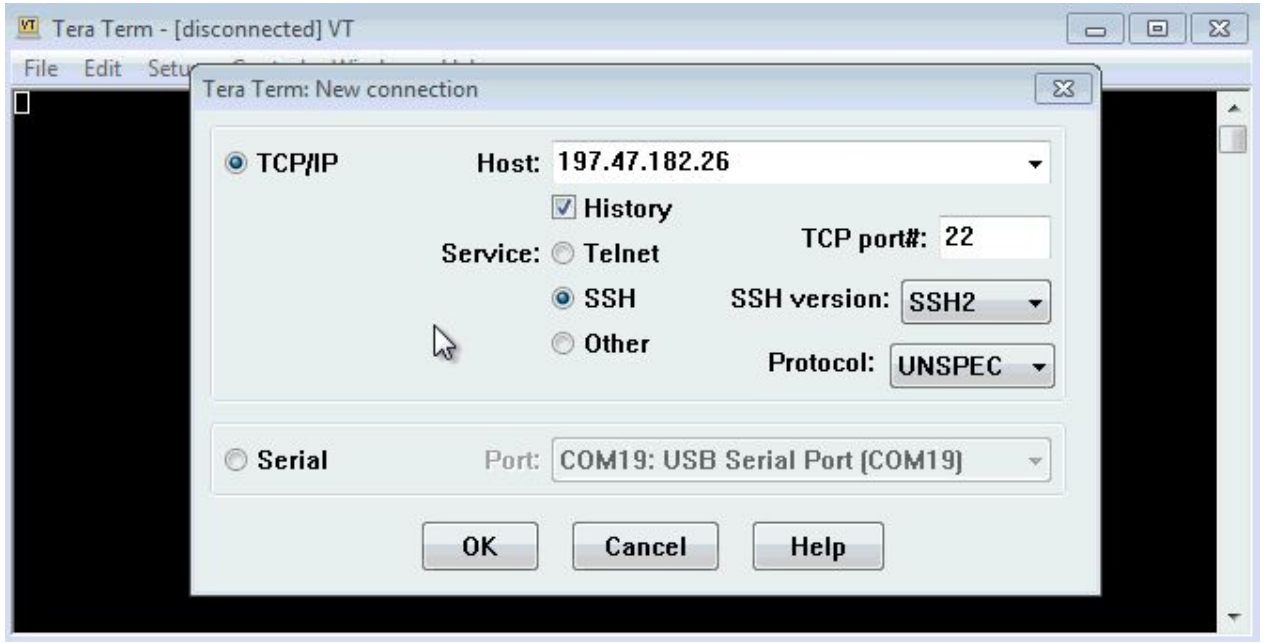


HAL

- Some Examples for STM32 cube HAL,

```
HAL_GPIO_WritePin(DEBUG_LED_GPIO_Port, DEBUG_LED_Pin, GPIO_PIN_SET);
HAL_GPIO_TogglePin(DEBUG_LED_GPIO_Port, DEBUG_LED_Pin);
HAL_UART_Transmit(&huart3, (uint8_t *)buf, buf_len, 100);
```





Getting Started with STM32

1 Setting up the environment needed to begin with STM32

Shalutha ENTC 17 — Apr '20

Hey guys,

So how are you guys? Hope you all are doing well.

From the last article, we told you that we will be looking into the programming side of STM32. But, we thought it would be much better for you guys, if we can give an idea about the hardware you need to do the programming with STM32 first. So today's article will be about the hardware you needed to program a STM32, and the step by step guide to settle them with your PC.

So the topics for today are as follows.

1 - Choosing a STM32 chip/Development Board for learning purposes

2 - Methods of programming the STM32 chip/Development Board

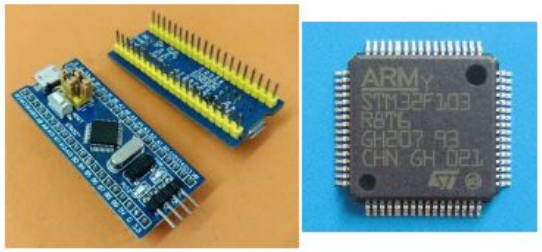
3 - Installing STM32CubeIDE

1 - Choosing a STM32 chip/Development Board for learning purposes

As I mentioned in the previous post, the STM32F1 series and the STMF4 series are the most suitable series for a beginner. For a price of RS 700-1000 range, you can easily buy a STM32F103C8 Development Board in Sri Lanka. F4 series's Nucleo Boards (having chips like STM32F401, STM32F411, etc) are a little bit high in cost in Sri Lanka. So because of the cost issue, we would recommend a STM32F103C8 board for testing purposes.

The chip we are using in our micromouse is STM32F405. The F4 series has a greater processing power and many other advantages (higher flash memory,SRAM, etc) than the F1 series, **but it should be mentioned that we can use the STM32F103C8 board to check every sensor, actuator, etc. that we are using in our mouse .** So that you don't need to wait until you completely build your mouse, you can check every function you are planning to use in your mouse, using the STM32F103C8 board (Bluepill).

Following shows a STM32F103C8 Development Board (Bluepill) & the chip (STM32F103C8) used in it



This one here is a STM32 Nucleo F411RE Development Board

Apr 2020 — 1 / 1 — Apr 2020

Apr 2020

2 Programming using CubeIDE #1- Blinking a LED

Senura ENTC 16 — Apr '20

Hi guys!.

Hope you are now able to set up the environment to start programming with STM32. If you have any questions or problems regarding our previous post (setting up the environment needed to begin with STM32) feel free to ask, because if you want to check your code in the real world that knowledge is a must.

Today, we will be looking at the key features of STM32CubeIDE and how a new project is created

Key Features

1.Integration of STM32CubeMX that provides services for:

- Project creation and STM32 microcontroller and microprocessor selection
- Pinout, clock, peripheral, and middleware configuration
- Generation of the initialization code

2.Based on ECLIPSE™/CDT, with support of ECLIPSE™ add-ons, GNU C/C++ for Arm® toolchain and GDB debugger

3.Additional advanced debug features including:

- CPU core, peripheral register, and memory views
- Live variable watch view
- System analysis and real-time tracing (SWV)
- CPU fault analysis tool

4.Support of ST-LINK (STMicroelectronics) and J-Link (SEGGER) debug probes

5.Import project from Atollic® TrueSTUDIO® and AC6 System Workbench for STM32 (SW4STM32)

6.Multi-OS support: Windows®, Linux®, and macOS®, 64-bit versions only

Project creation and STM32 microcontroller and microprocessor selection

Workspace and project creation :

The first thing to be done when using CubeIDE is to create a project. Before that we will introduce you, the concept of workspace.

A workspace is a container that includes project folders or information about project folders, and a .metadata folder that contains information about the projects. A workspace is simply a folder on a hard drive, which can be located anywhere in the storage media. When STM32CubeIDE starts up, it asks which workspace must be used. This may be changed at any time by selecting [File]>[Switch Workspace] and navigating to another folder.

The easiest way to create a new embedded project is to use the STM32 Project wizard. It is selected through the [File]>[New]>[STM32 Project] menu command, and launches the embedded MCUFinder:

1 - Select the target MCU or board and click next (We have chosen STM32F103C8 MCU).

- This image shows you the target selection window. In light blue color in the upper left corner of this image, you can see now we are at the MCU/MPU selector tab.

Apr 2020 — 1 / 2 — Apr 2020

Apr 2020

Wokwi STM32 Simulator

WOKWI SAVE SHARE Docs

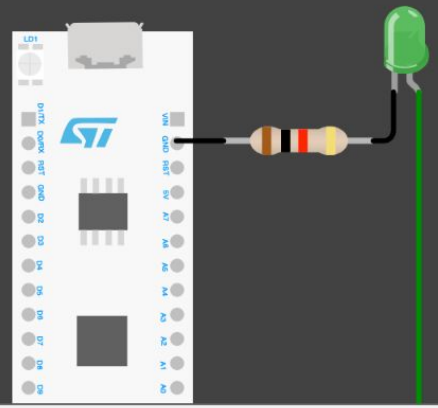
main.c | diagram.json | Library Manager

```
1  // STM32 Nucleo-L031K6 HAL Blink + printf() example
2  // Simulation: https://wokwi.com/projects/367244067477216257
3
4  #include <stdio.h>
5  #include <stdint.h>
6  #include <string.h>
7  #include <stm32l0xx_hal.h>
8
9  // ST Nucleo Green user LED (PB3)
10 #define LED_PORT                GPIOB
11 #define LED_PIN                 GPIO_PIN_3
12 #define LED_PORT_CLK_ENABLE     __HAL_RCC_GPIOB_CLK_ENABLE
13 #define VCP_TX_Pin GPIO_PIN_2
14 #define VCP_RX_Pin GPIO_PIN_15
15
16 UART_HandleTypeDef huart2;
17
18 void SystemClock_Config(void);
19 static void MX_USART2_UART_Init(void);
20
21 void osSystickHandler(void)
22 {
23   // 1 Hz blinking:
24   if ((HAL_GetTick() % 500) == 0)
25   {
26     HAL_GPIO_TogglePin(LED_PORT, LED_PIN);
27   }
28 }
29
30 void initGPIO()
31 {
32   GPIO_InitTypeDef GPIO_Config;
```

Simulation

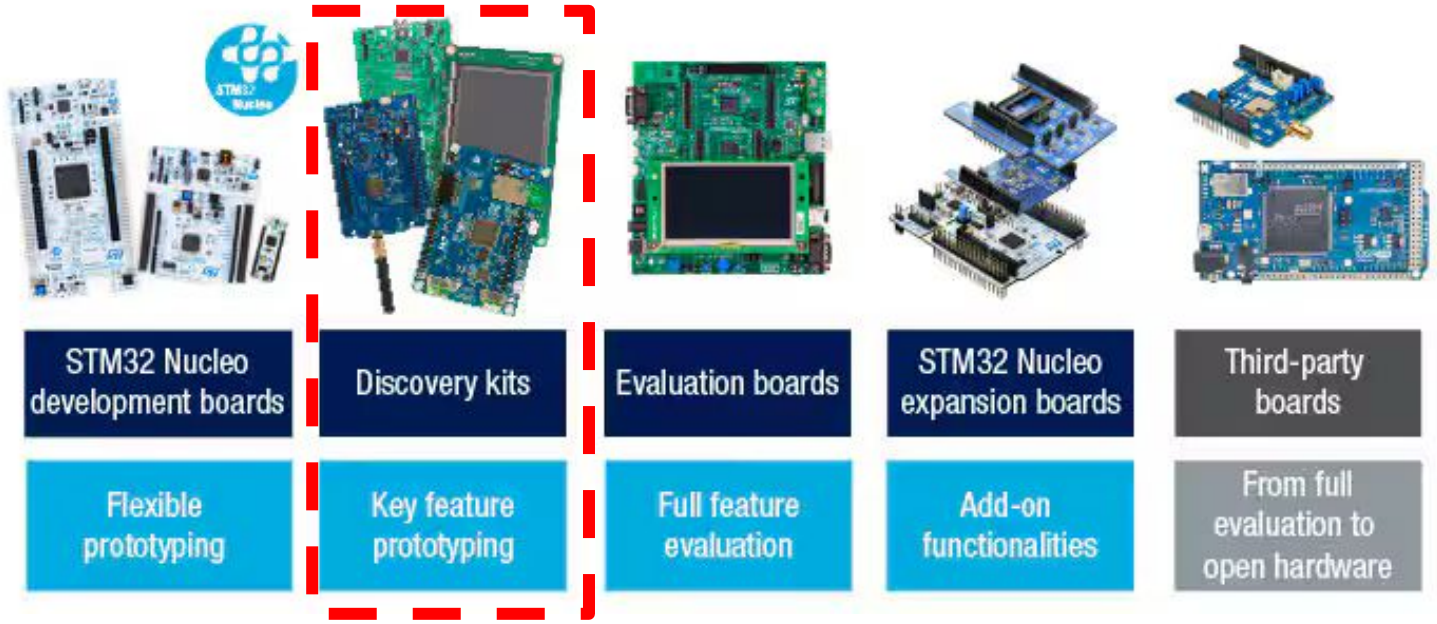
00:14.496 99%

https://wokwi.com/projects/392812608178590721

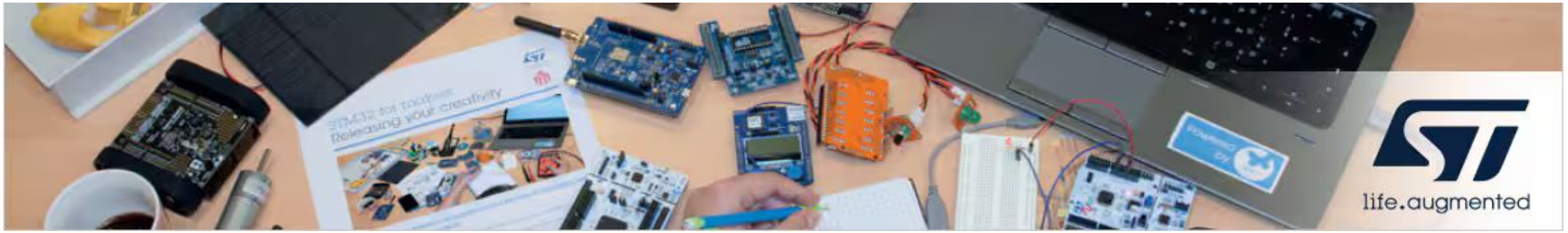


```
Hello, World!
Hello Students! Tracing X = 11
Hello, World!
Hello Students! Tracing X = 12
Hello, World!
Hello Students! Tracing X = 13
```

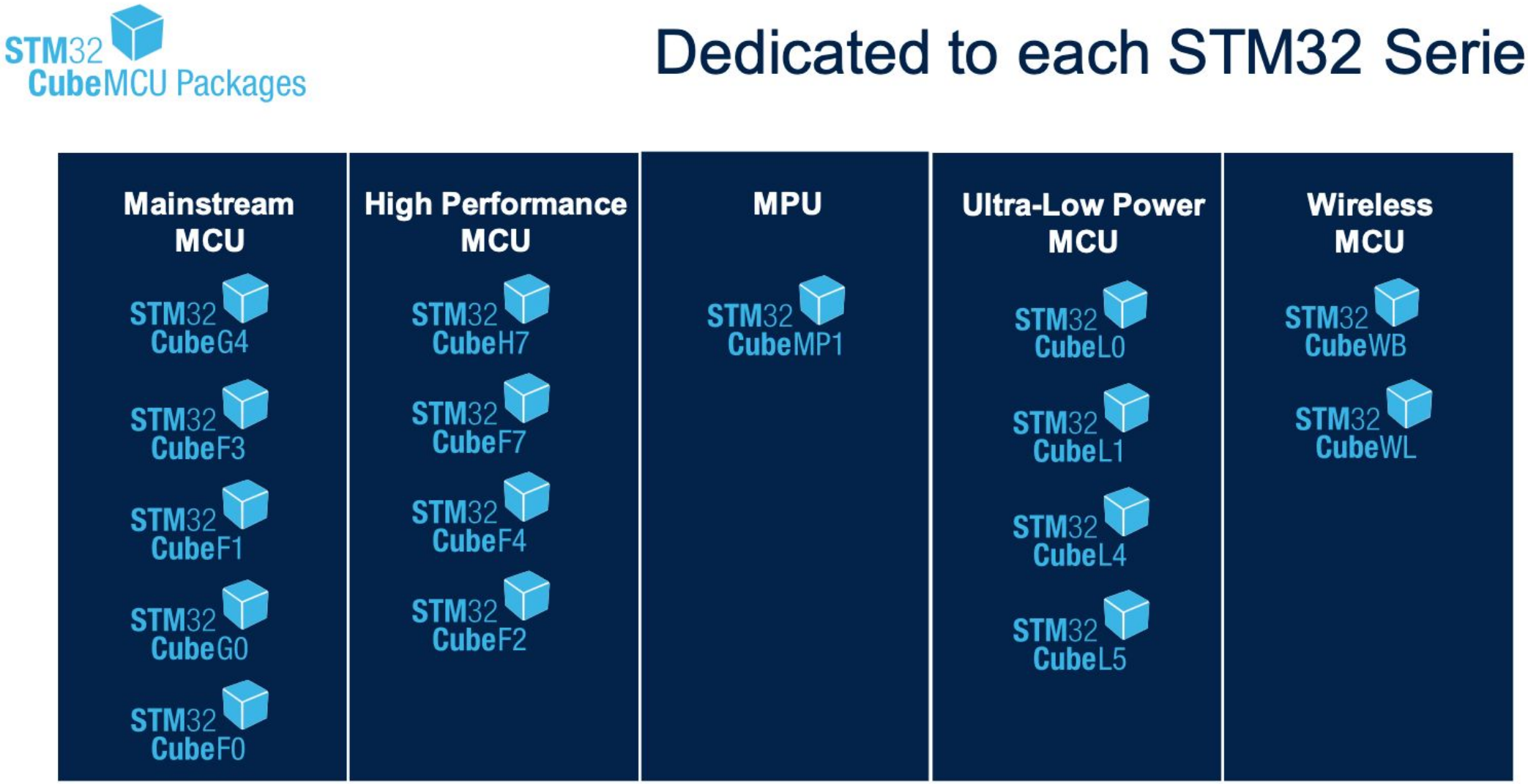
Getting started with STM32



[Discovery kit with STM32F407VG MCU * New order code STM32F407G-DISC1 (replaces STM32F4DISCOVERY) - STMicroelectronics]

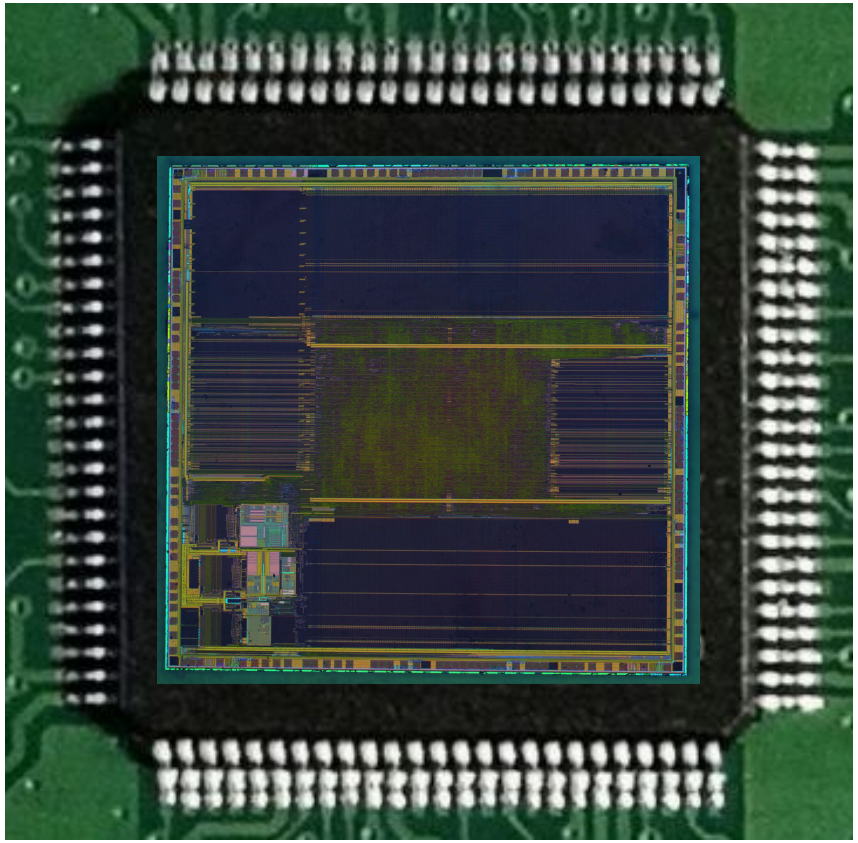


STM32 Series



STM32 Datasheets

- Refer datasheet to get all the necessary information about the chip.
- Discuss important parts of the datasheet



life.augmented

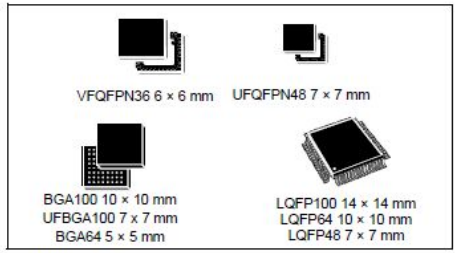
STM32F103x8
STM32F103xB

Medium-density performance line Arm®-based 32-bit MCU with 64 or 128 KB Flash, USB, CAN, 7 timers, 2 ADCs, 9 com. interfaces

Datasheet - production data

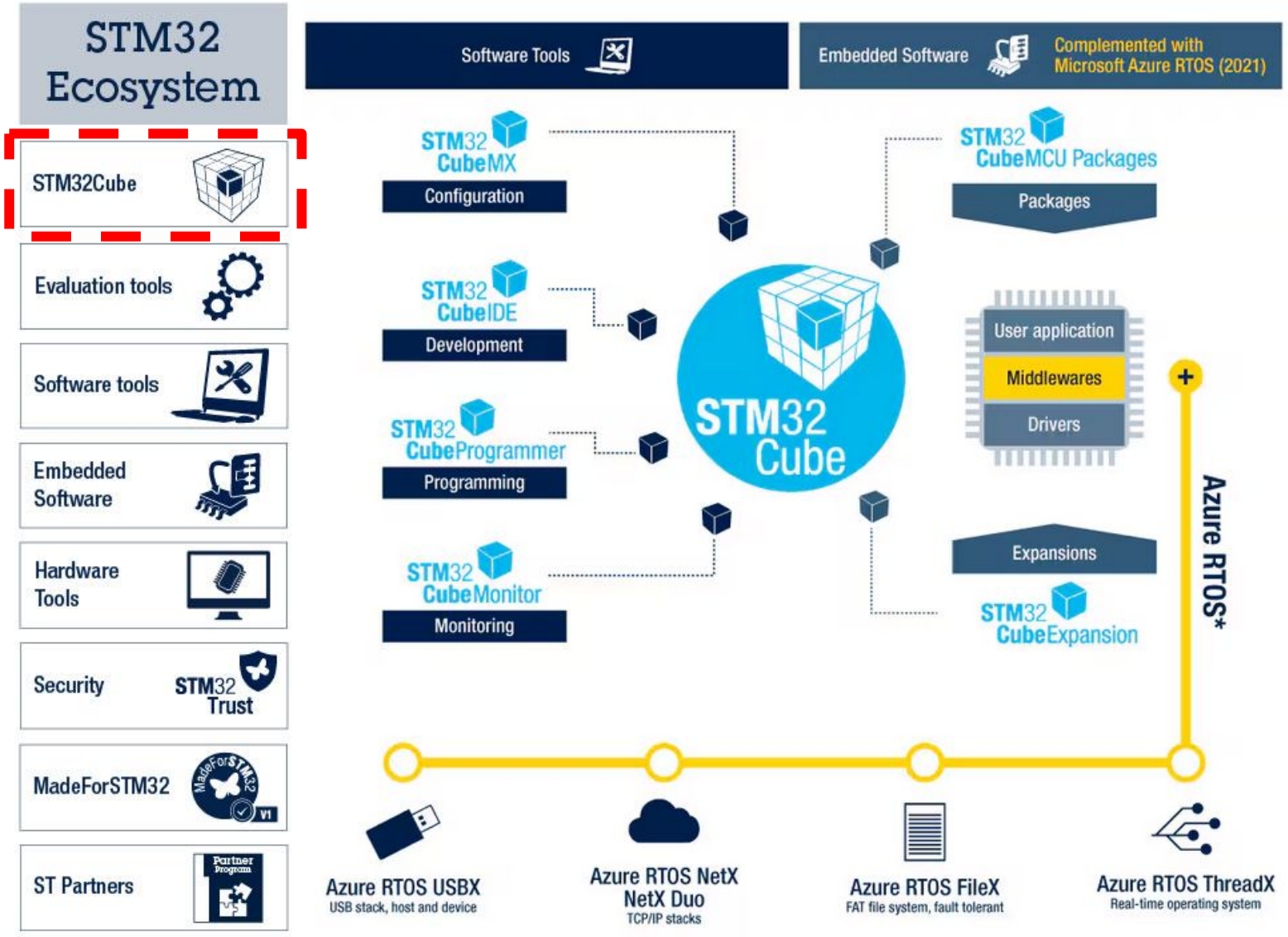
Features

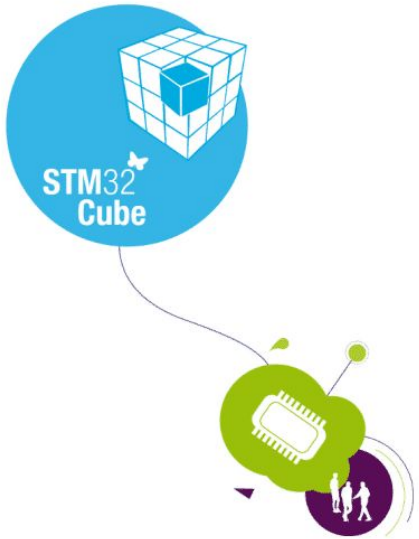
- Arm® 32-bit Cortex®-M3 CPU core
  - 72 MHz maximum frequency, 1.25 DMIPS / MHz (Dhrystone 2.1) performance at 0 wait state memory access
  - Single-cycle multiplication and hardware division
- Memories
  - 64 or 128 Kbytes of Flash memory
  - 20 Kbytes of SRAM
- Clock, reset and supply management
  - 2.0 to 3.6 V application supply and I/Os
  - POR, PDR, and programmable voltage detector (PVD)
  - 4 to 16 MHz crystal oscillator
  - Internal 8 MHz factory-trimmed RC
  - Internal 40 kHz RC
  - PLL for CPU clock
  - 32 kHz oscillator for RTC with calibration
- Low-power
  - Sleep, Stop and Standby modes
  - V_{BAT} supply for RTC and backup registers
- 2x 12-bit, 1 µs A/D converters (up to 16 channels)
  - Conversion range: 0 to 3.6 V
  - Dual-sample and hold capability
  - Temperature sensor



- Debug mode
  - Serial wire debug (SWD) and JTAG interfaces
- Seven timers
  - Three 16-bit timers, each with up to 4 IC/OC/PWM or pulse counter and quadrature (incremental) encoder input
  - 16-bit, motor control PWM timer with dead-time generation and emergency stop
  - Two watchdog timers (independent and window)
  - SysTick timer 24-bit downcounter
- Up to nine communication interfaces
  - Up to two I^2C interfaces (SMBus/PMBus®)
  - Up to three USARTs (ISO 7816 interface, LIN, IrDA capability, modem control)
  - Up to two SPIs (18 Mbit/s)
  - CAN interface (2.0B Active)
  - USB 2.0 full-speed interface
- CRC calculation unit, 96-bit unique ID

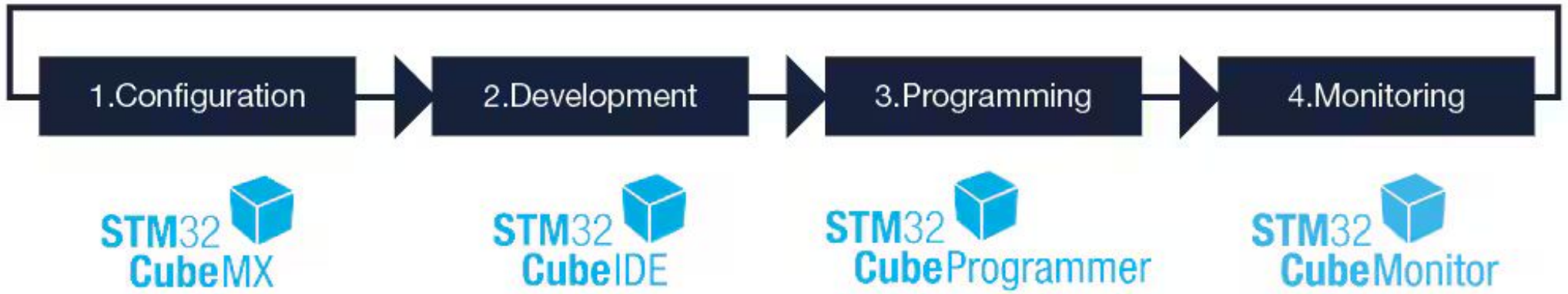
STM32Cube Ecosystem





ST's software ecosystem
STM32Cube

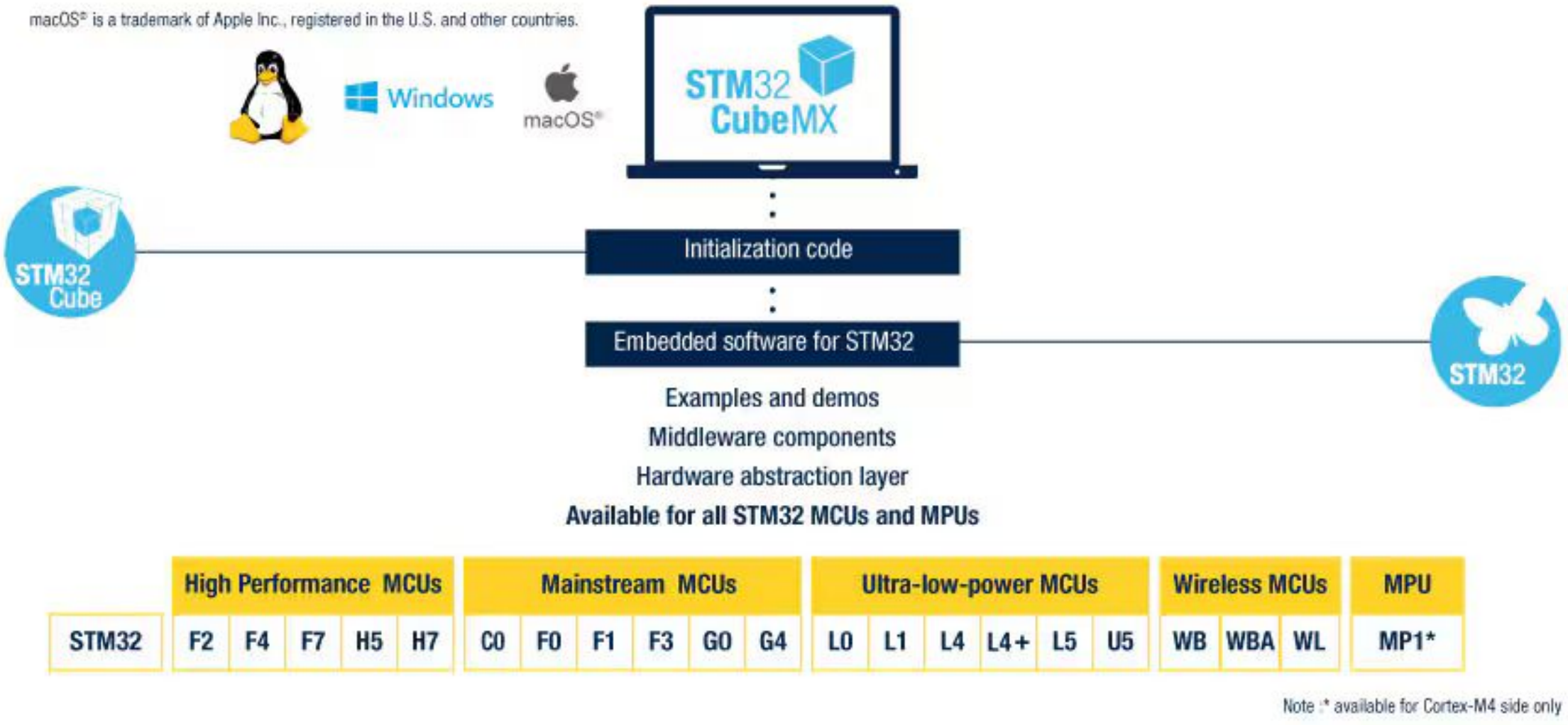
STM32Cube Ecosystem



Part Number	Configuration	Initialization code Generation	Code Edition	Code Building	Debugging	Binary Programming	Monitoring
STM32CubeMX	✔	✔					
STM32CubeIDE	✔ (Integrates STM32CubeMX)	✔ (Integrates STM32CubeMX)	✔	✔	✔	✔	
STM32CubeProgrammer						✔	
STM32CubeMonitor							✔

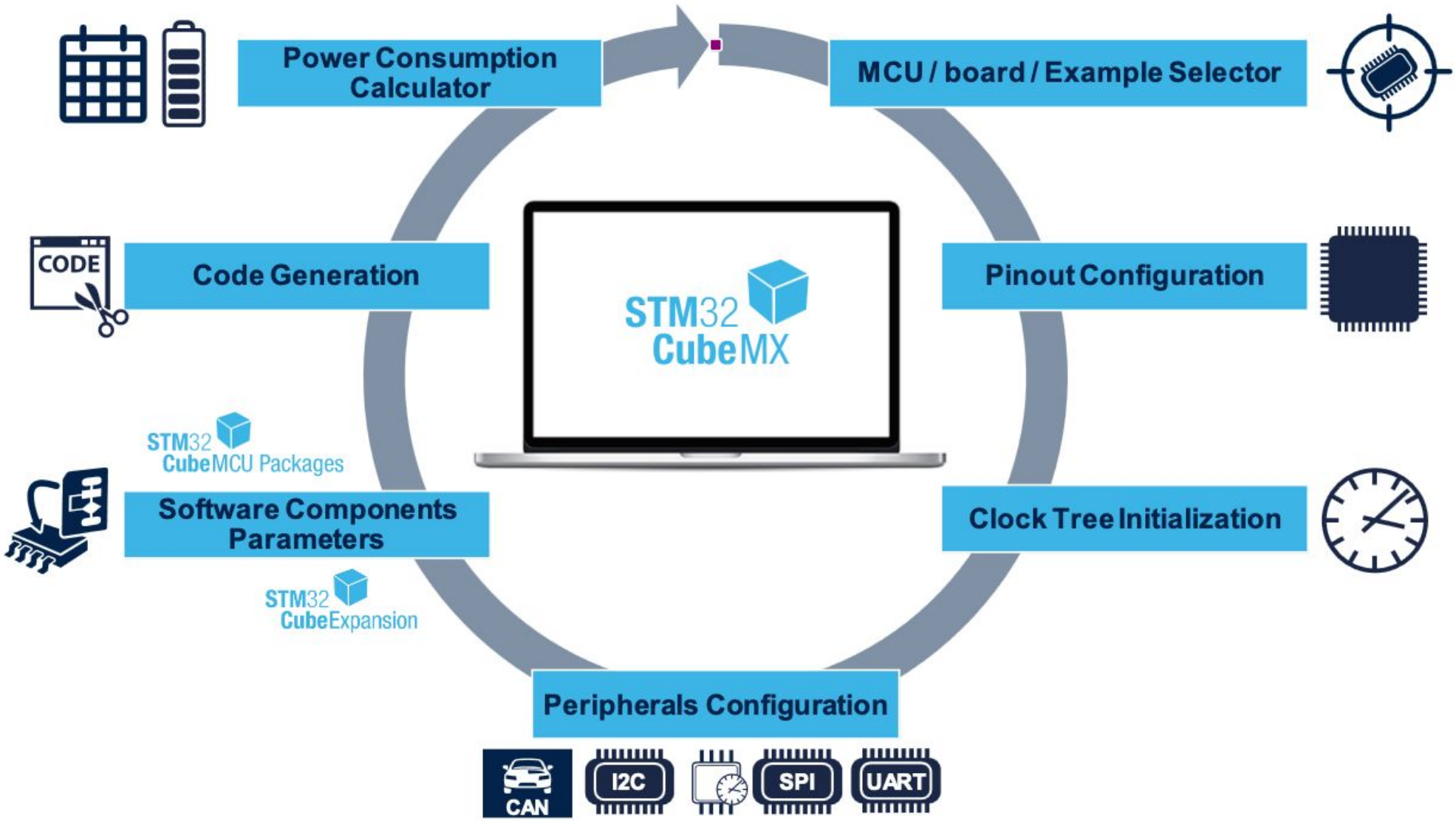
Reference - [Discover the STM32Cube Ecosystem - STMicroelectronics]

STM32CubeMX

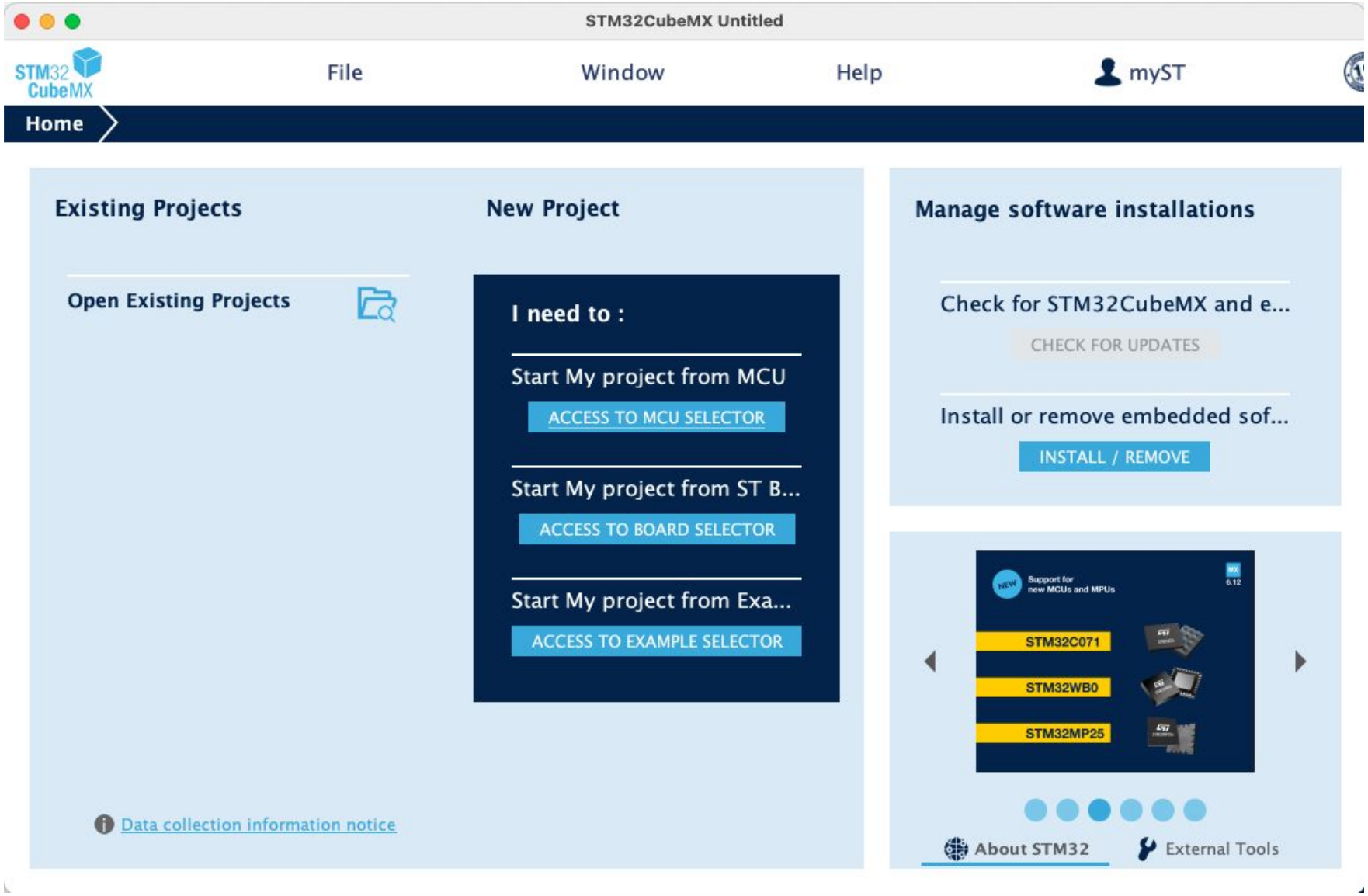


[STM32CubeMX - STM32Cube initialization code generator - STMicroelectronics]

STM32CubeMX key steps



CubeMX IDE

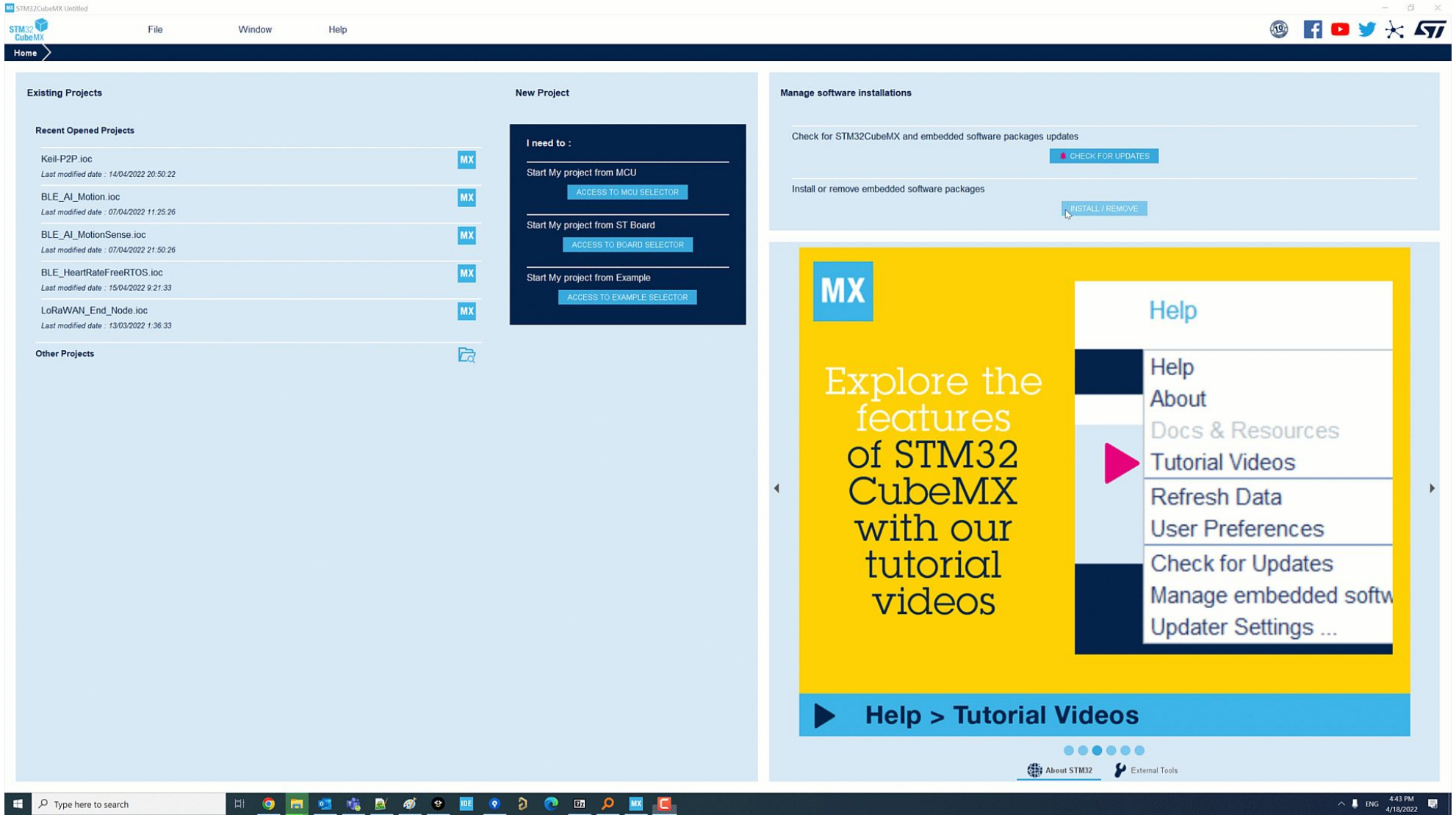


MCU / MPU SELECTOR

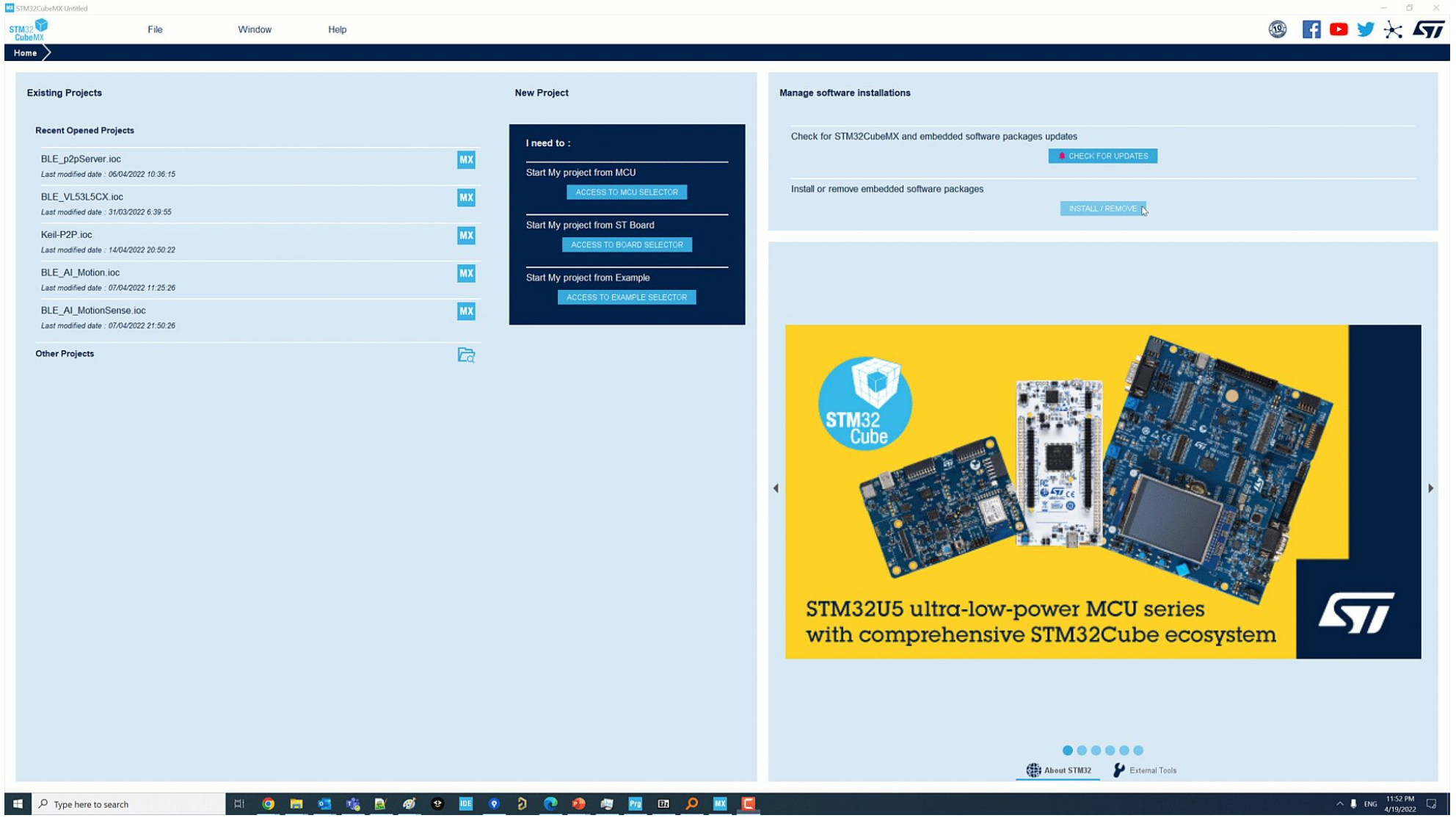
BOARD SELECTOR

EXAMPLE SELECTOR

Install Expansion Middleware - X.CUBE-AI



Install Expansion Middleware - X.CUBE-MEMS1

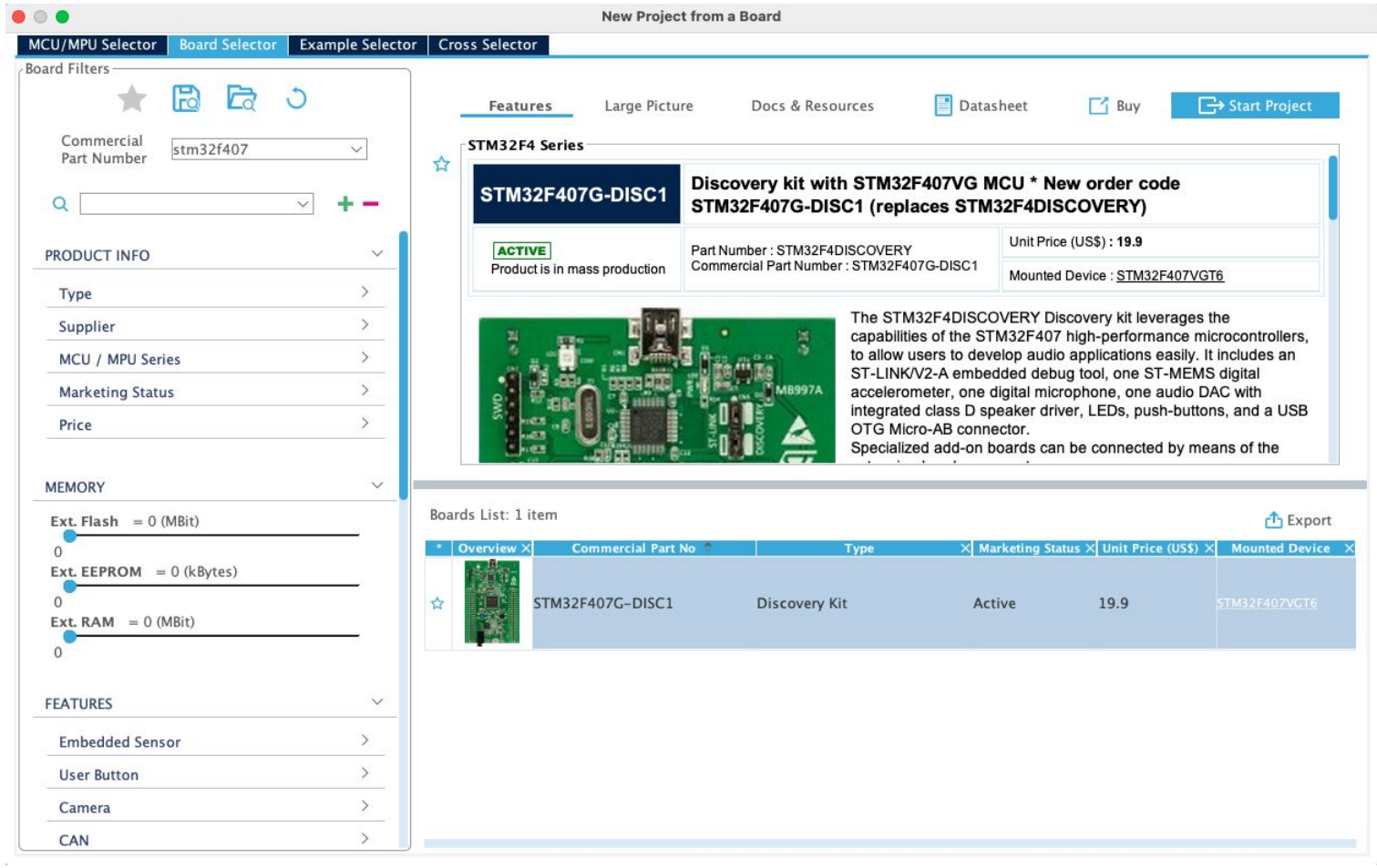


MCU/MPU or Board selector

MCU / MPU SELECTOR

BOARD SELECTOR

EXAMPLE SELECTOR

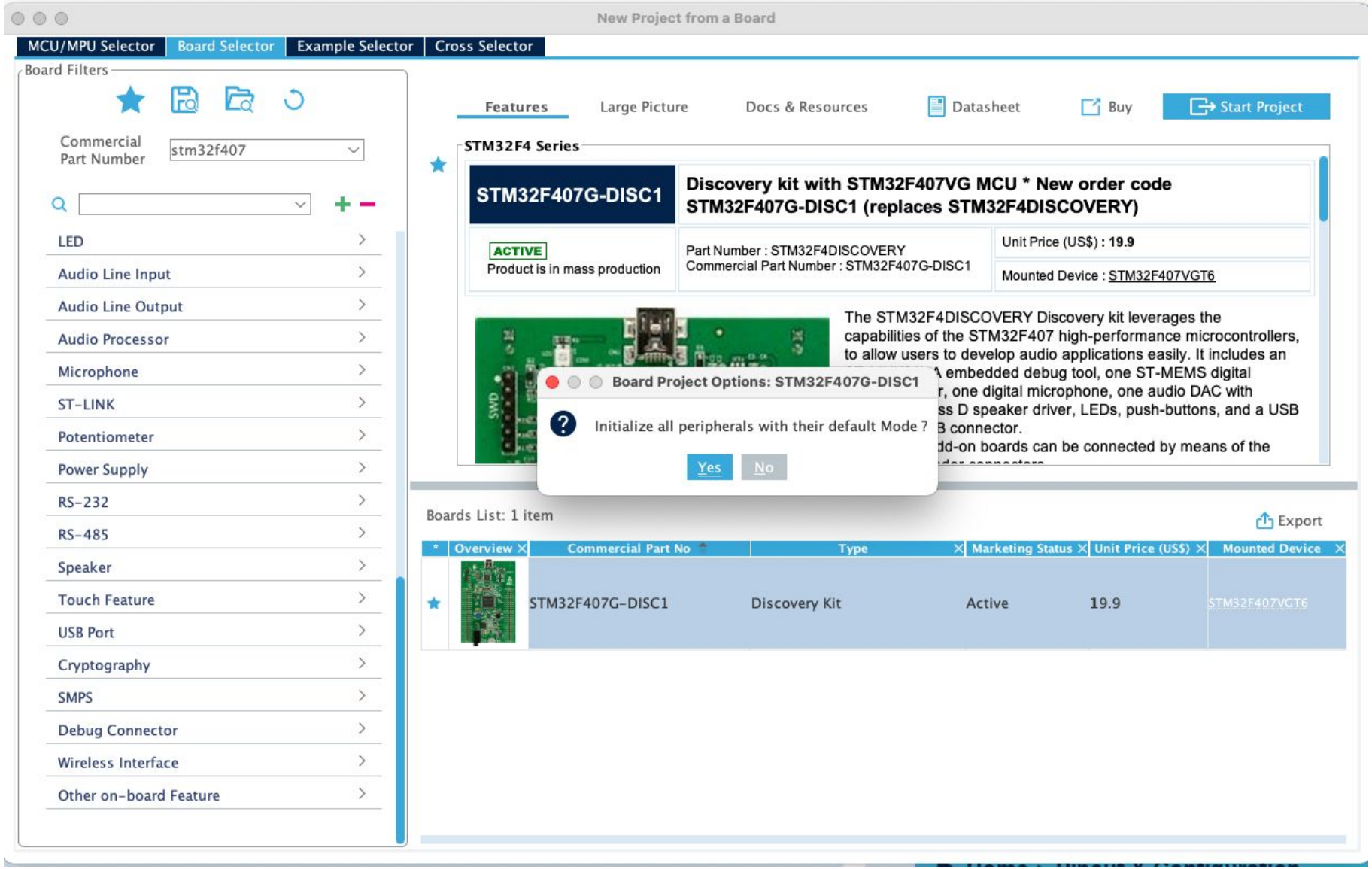


DEDICATED FILTERS

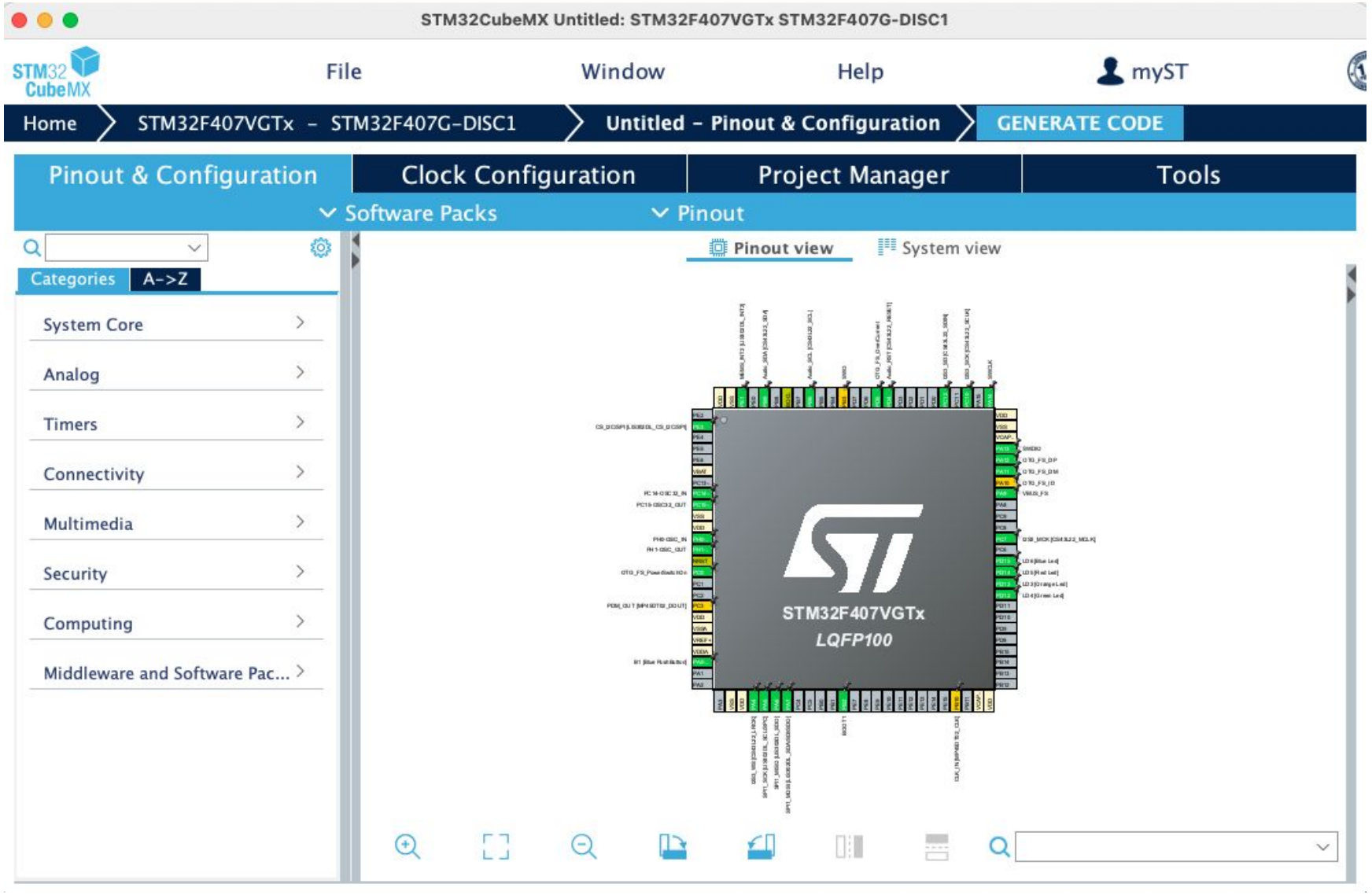
DESCRIPTION & INFORMATION

PRODUCT LIST

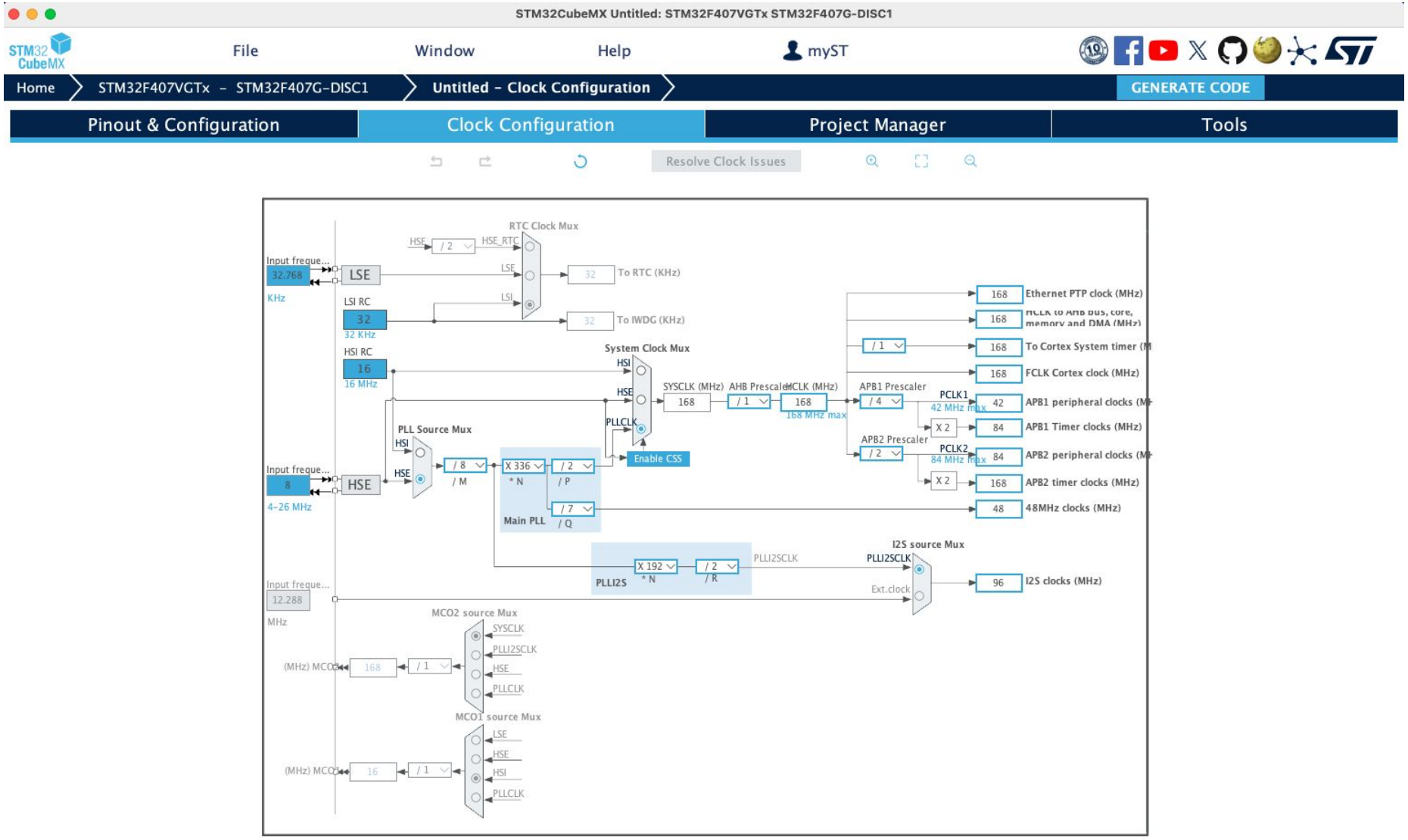
Board selector



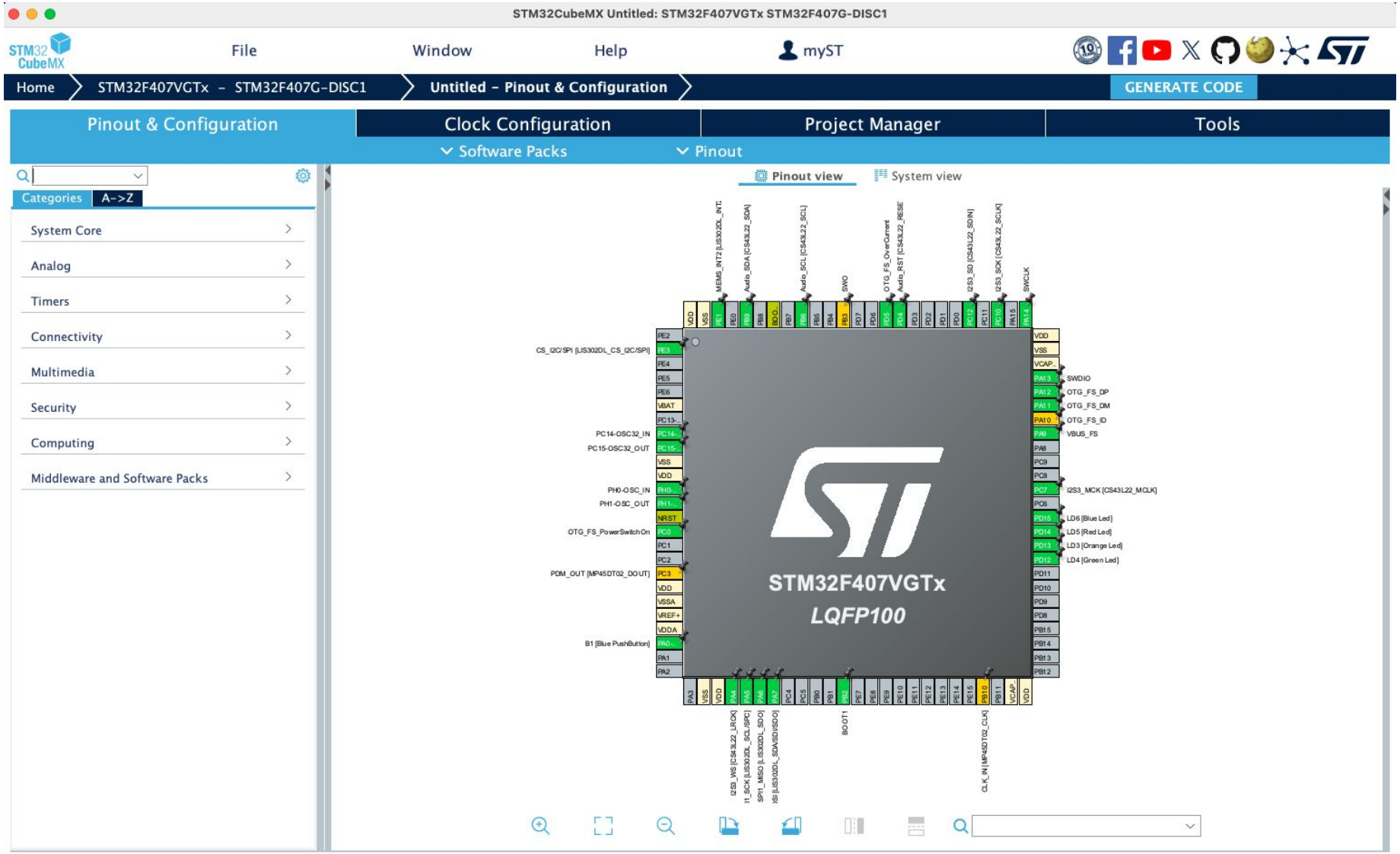
Pinout configuration



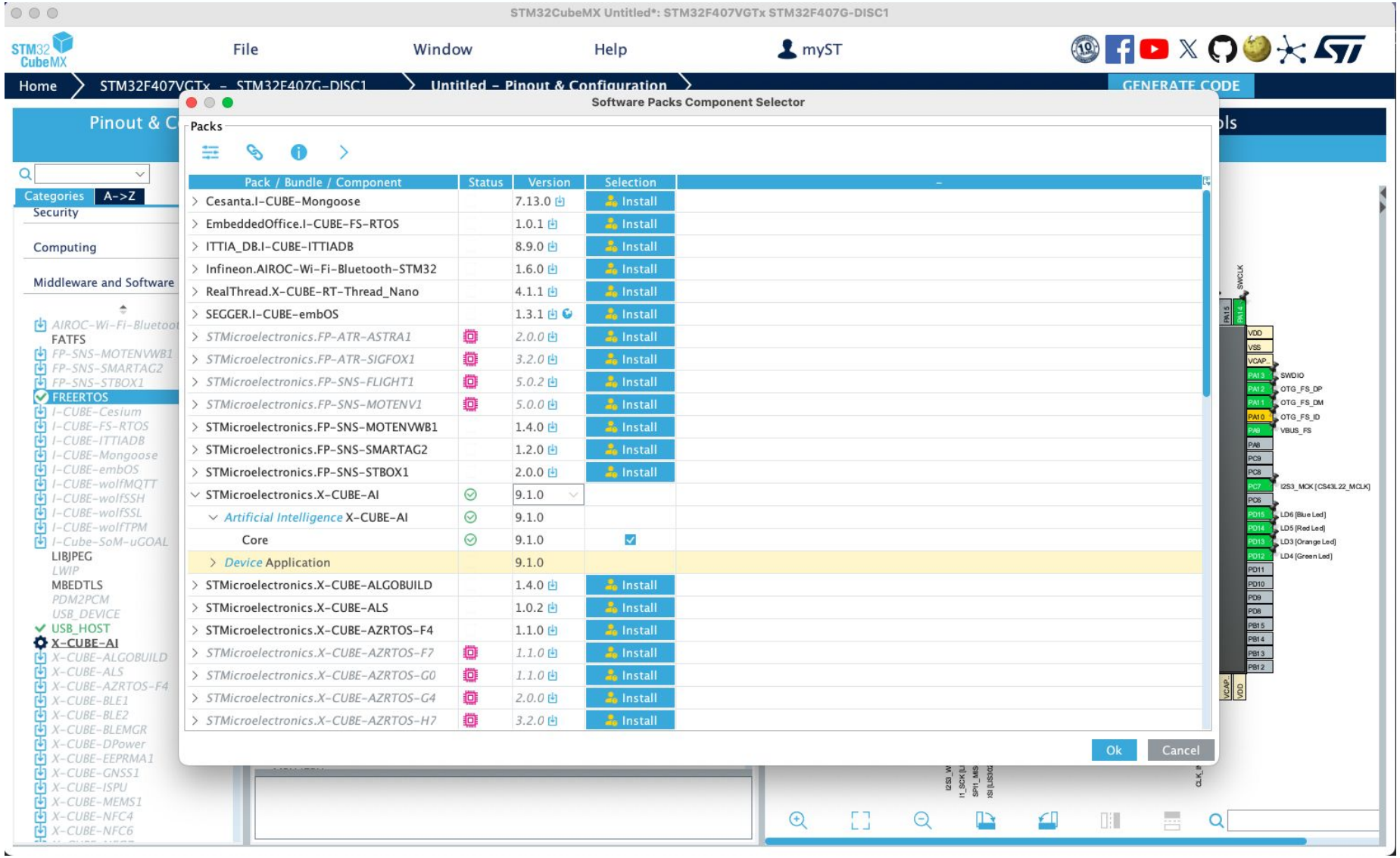
Clock-tree configuration



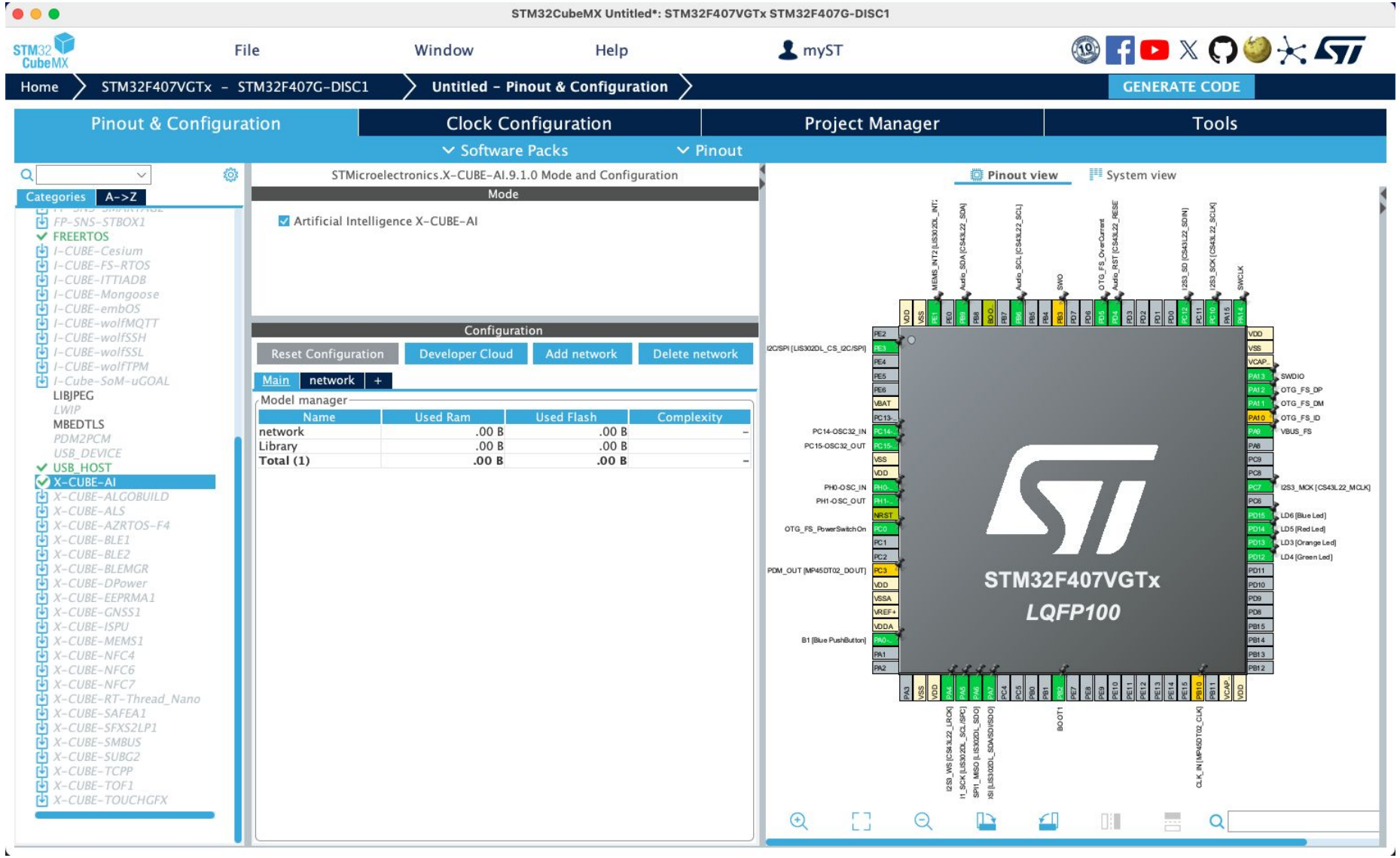
Peripheral parameters



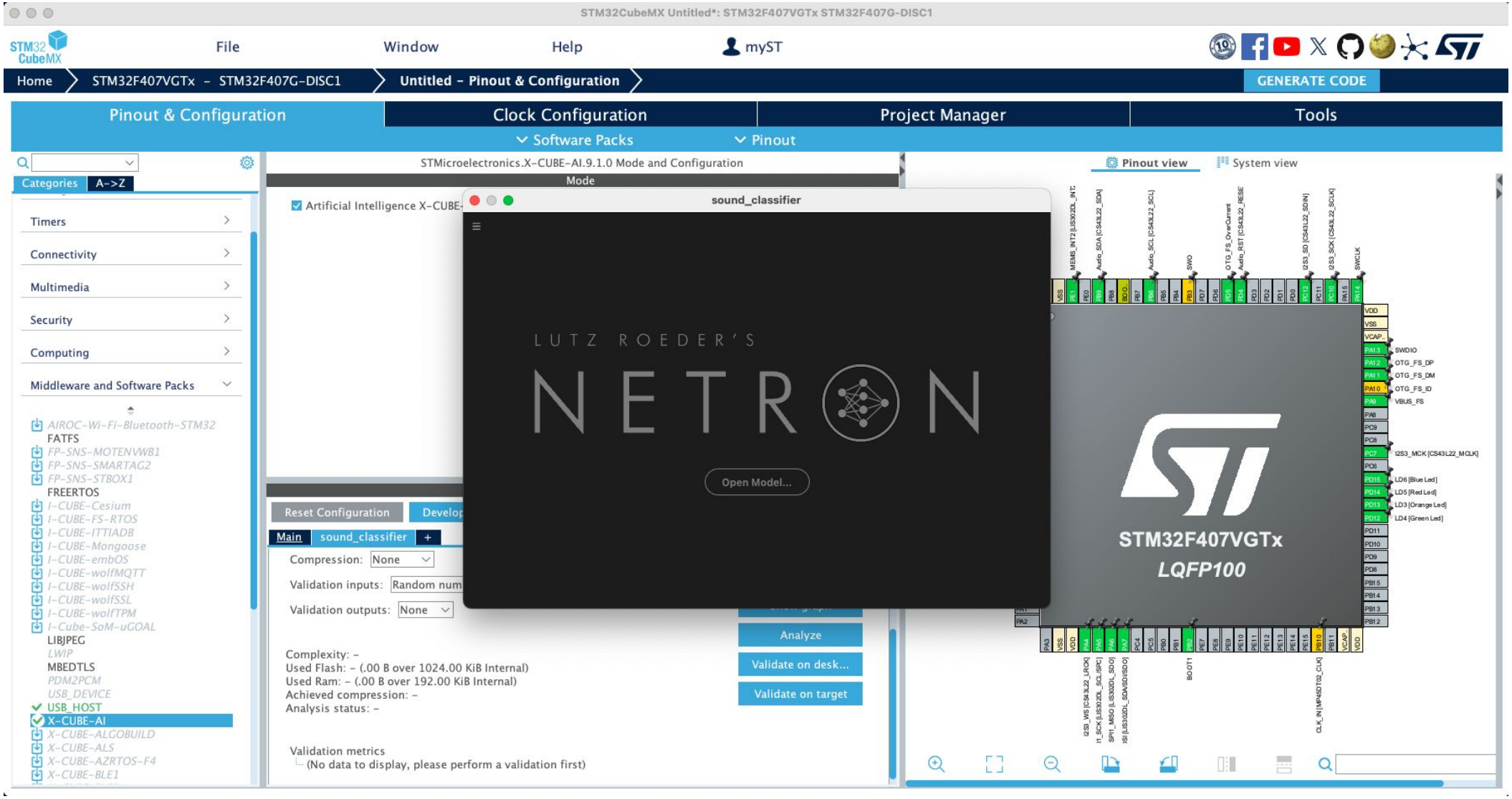
Add Expansion Middleware



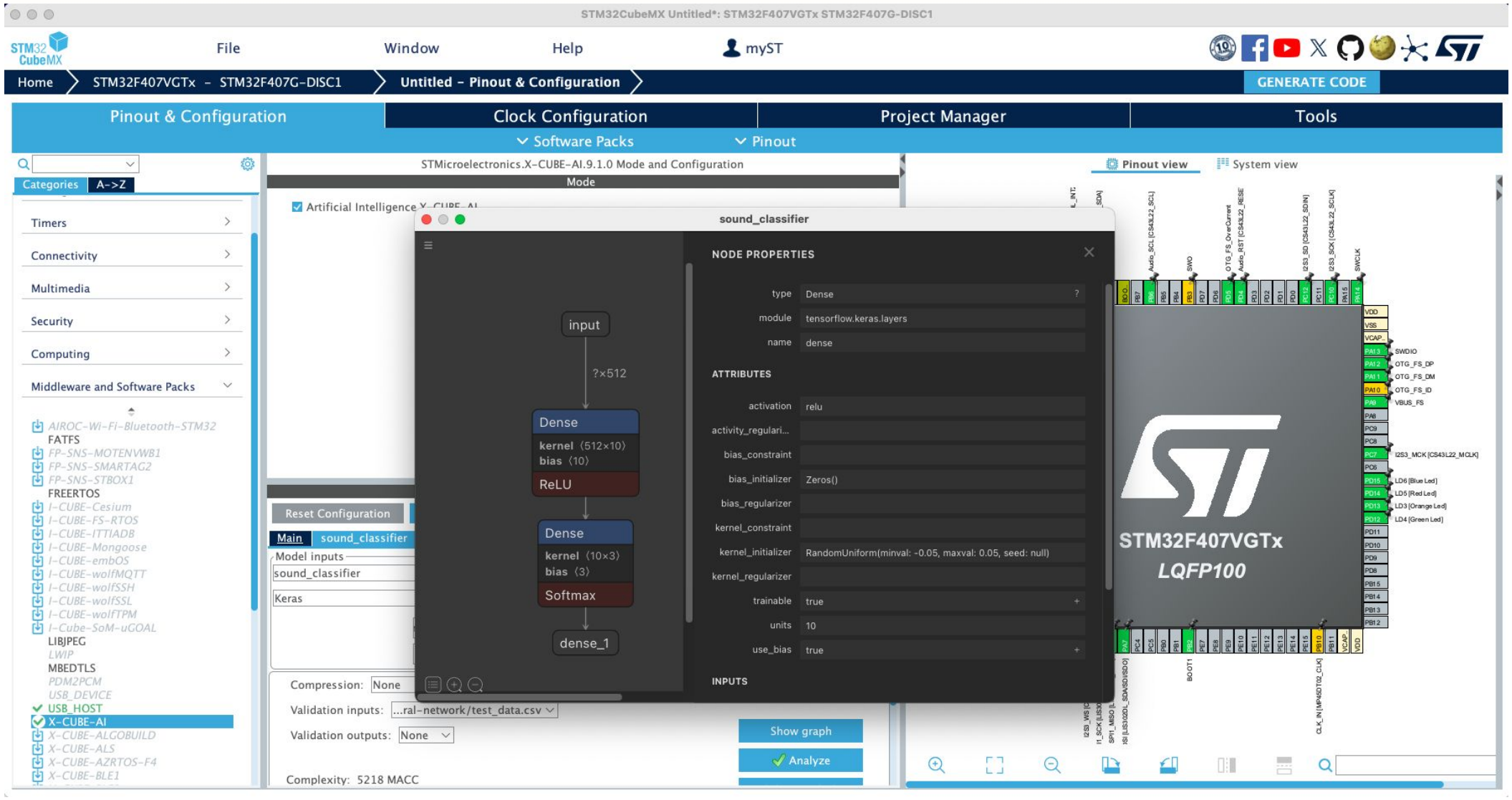
MCU/MPU selection for AI application



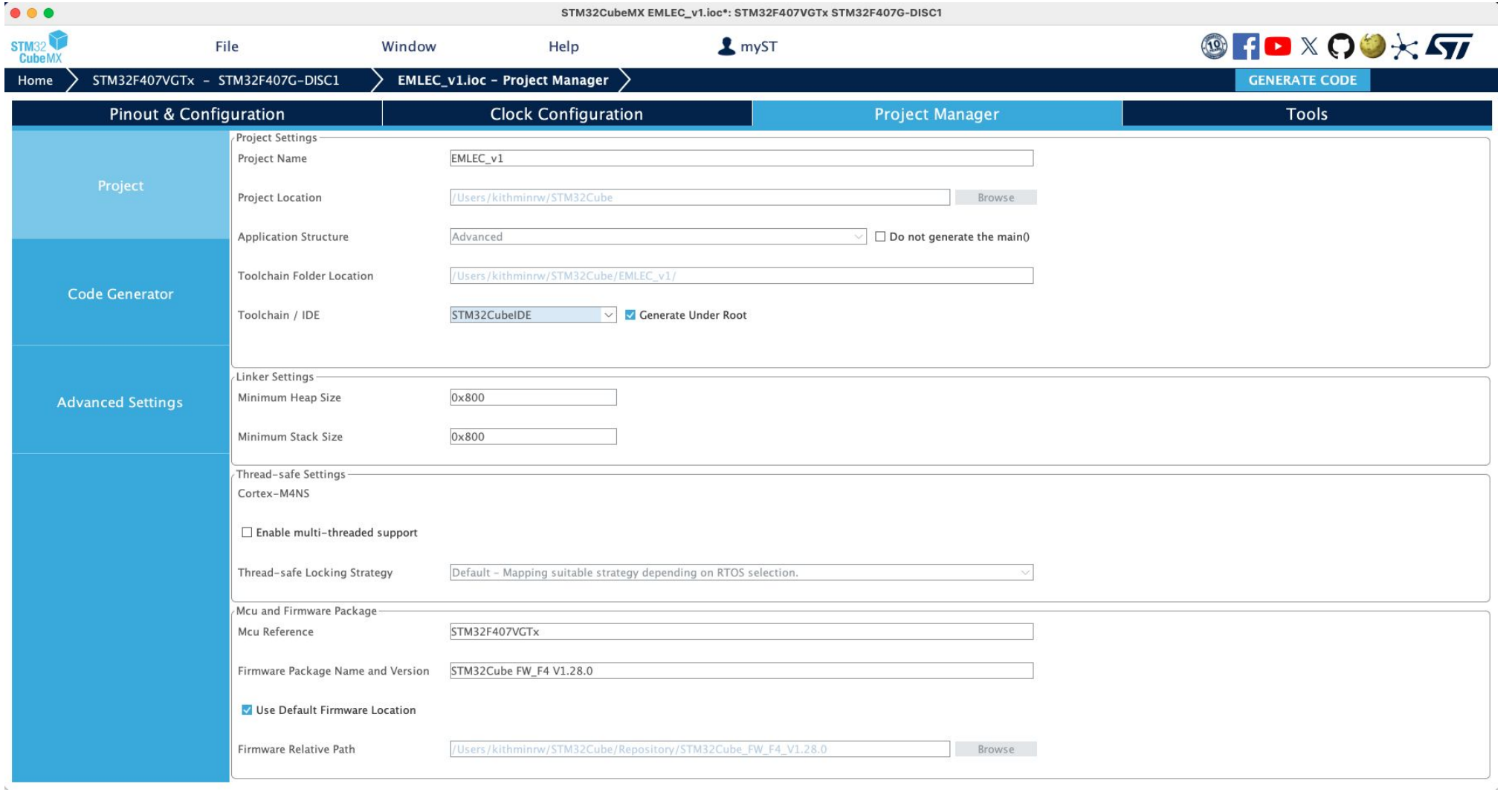
Visualize Model



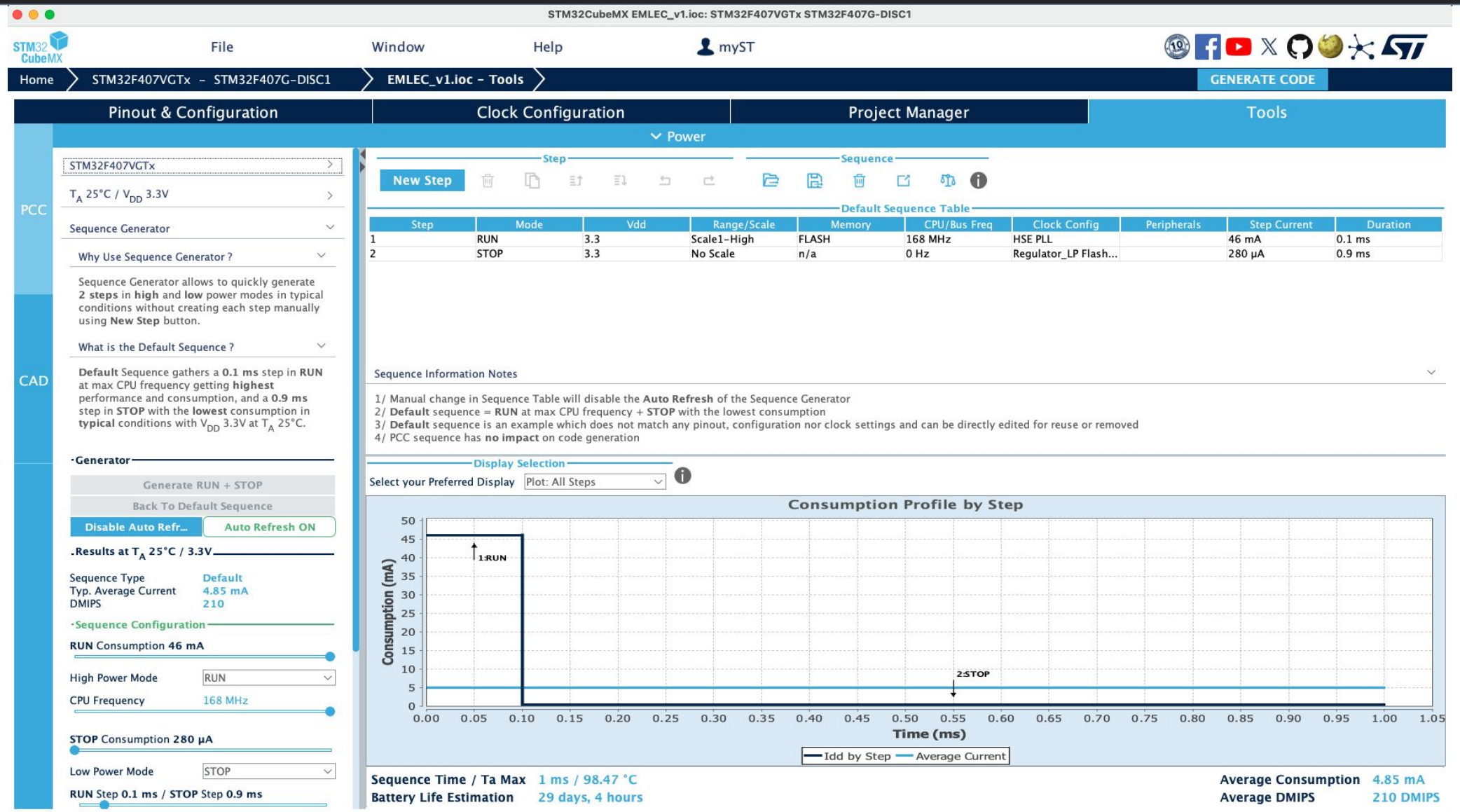
Visualize Model



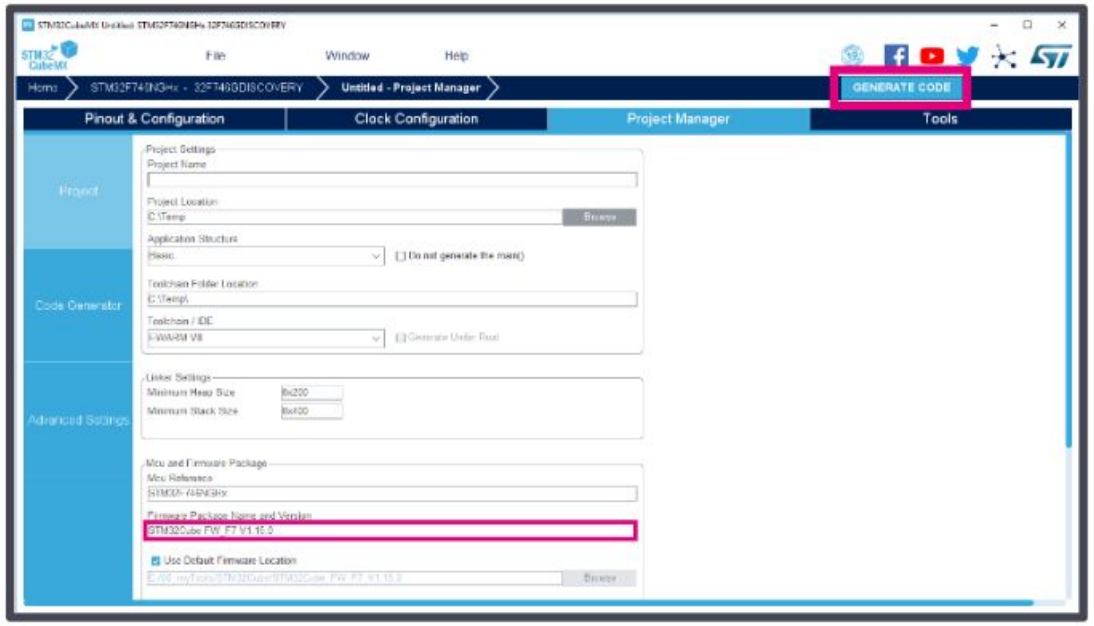
Code generation



Power consumption calculator

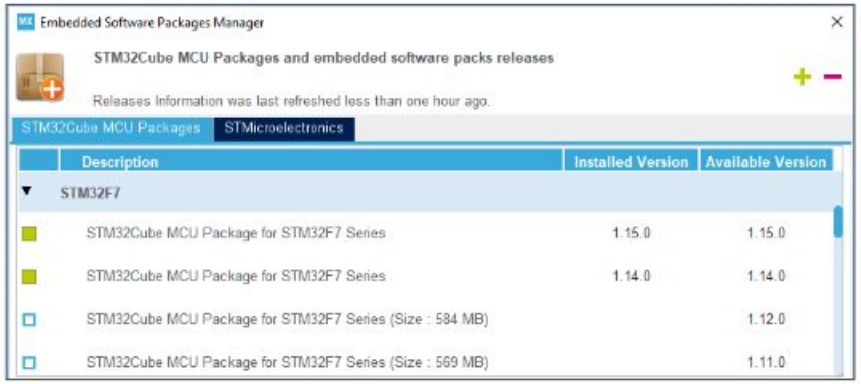


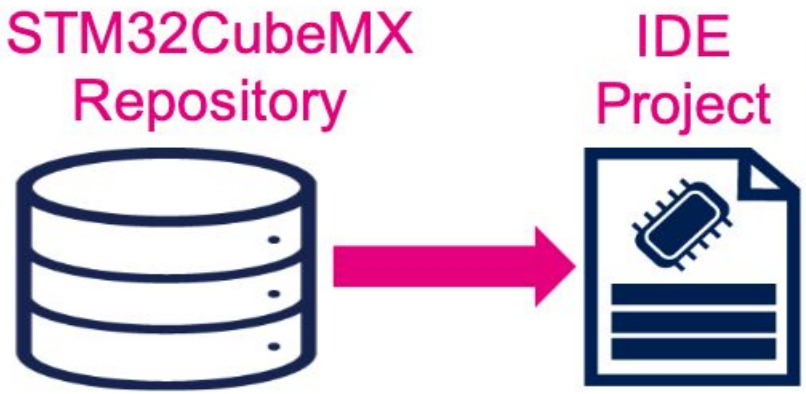
STM32CubeMX repository

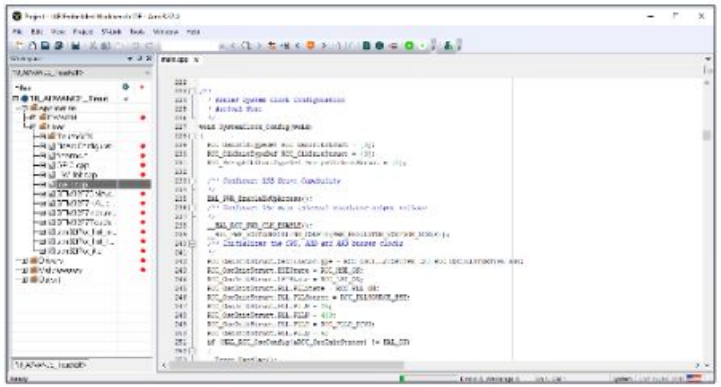


After pressing "GENERATE CODE":

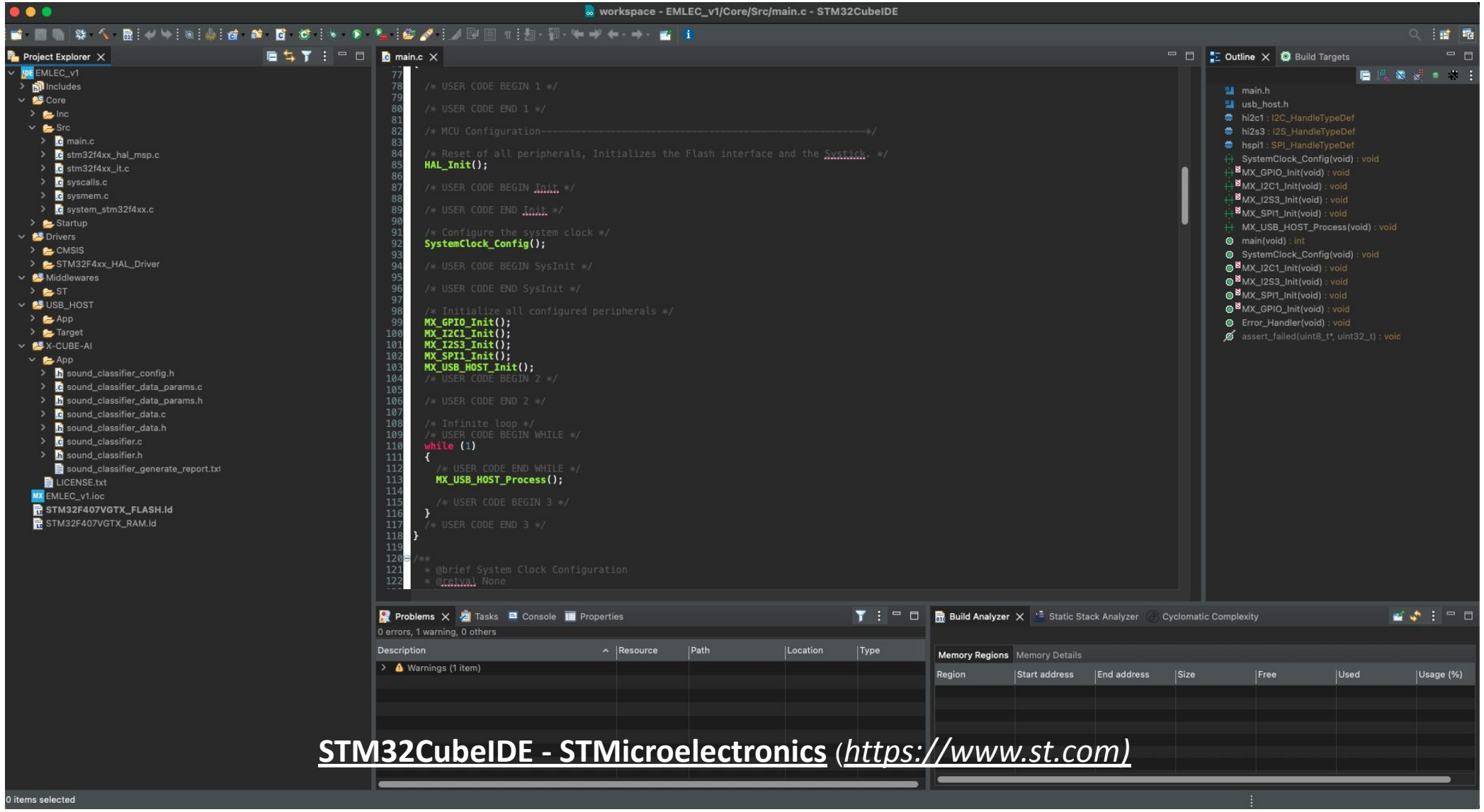
1. STM32CubeMX grabs necessary peripheral drivers based on your pinout/peripheral configuration from STM32Cube MCU Package in STM32CubeMX Repository
2. STM32CubeMX grabs necessary middleware based on your middleware configuration from STM32Cube MCU Package in STM32CubeMX repository
3. Generate IDE project







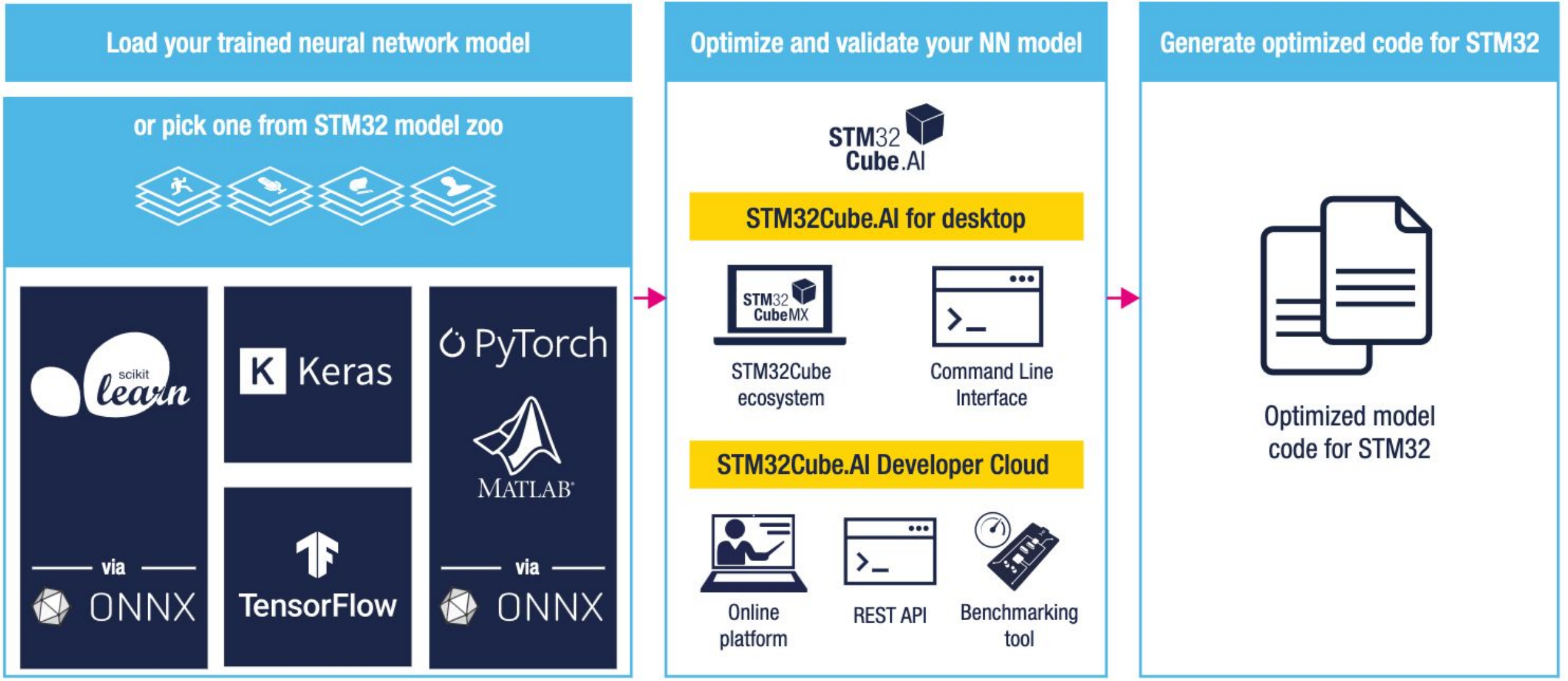
STM32 CubeIDE



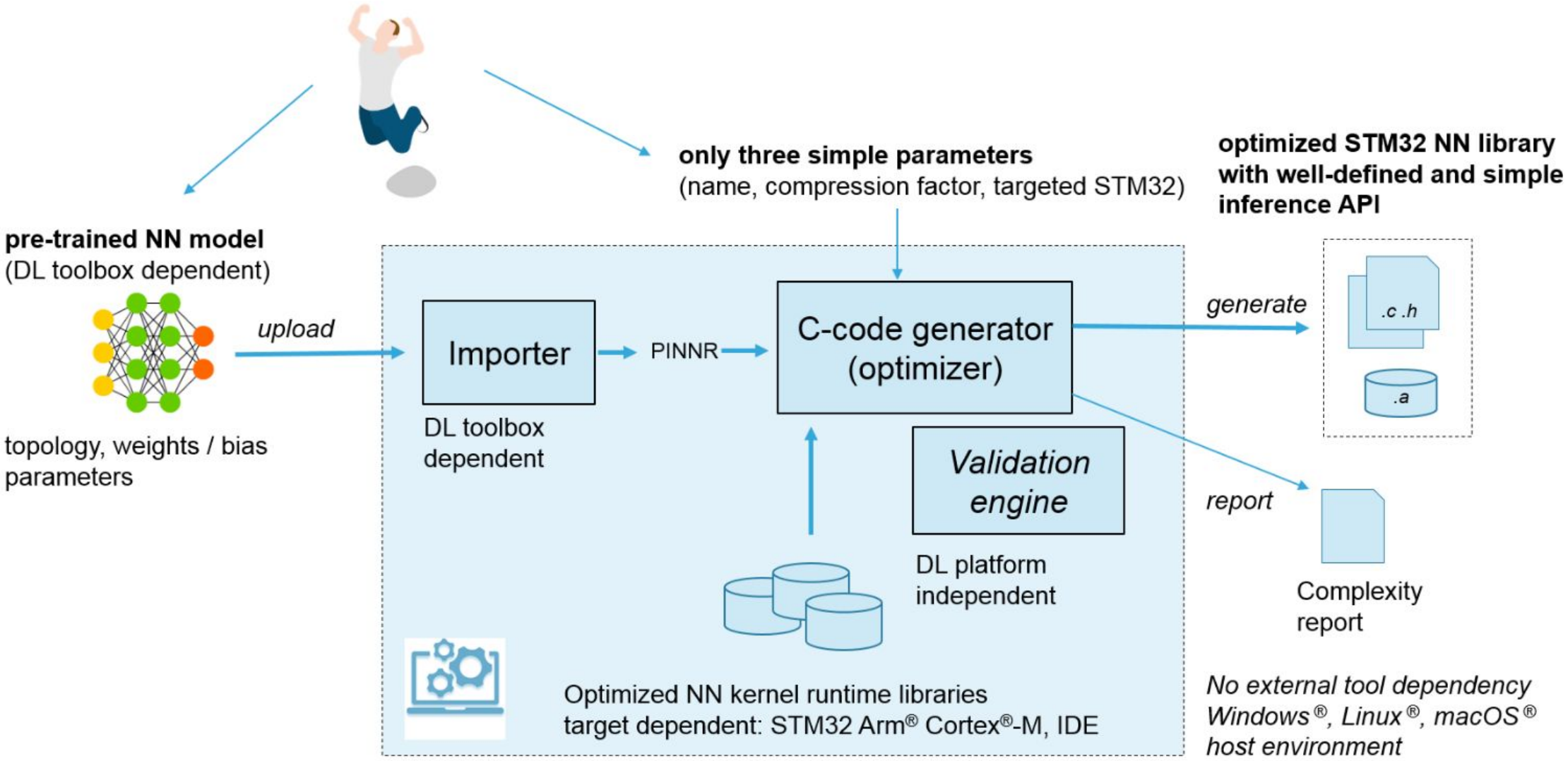
STM32CubeIDE - STMicroelectronics (*https://www.st.com*)

History of STM32 CubeIDE

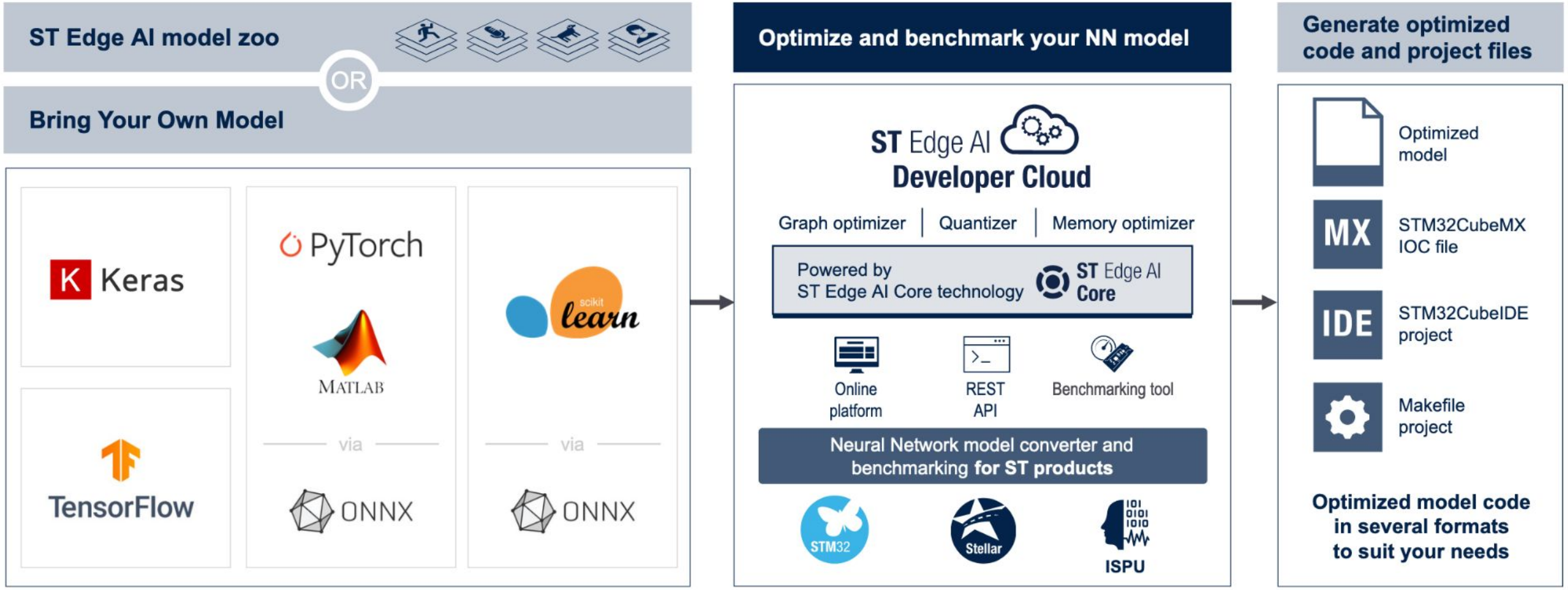
STM32 Cube.AI



X-CUBE-AI core engine



STM32 Cube.AI



AI:X-CUBE-AI documentation - stm32mcu

STM32 model zoo

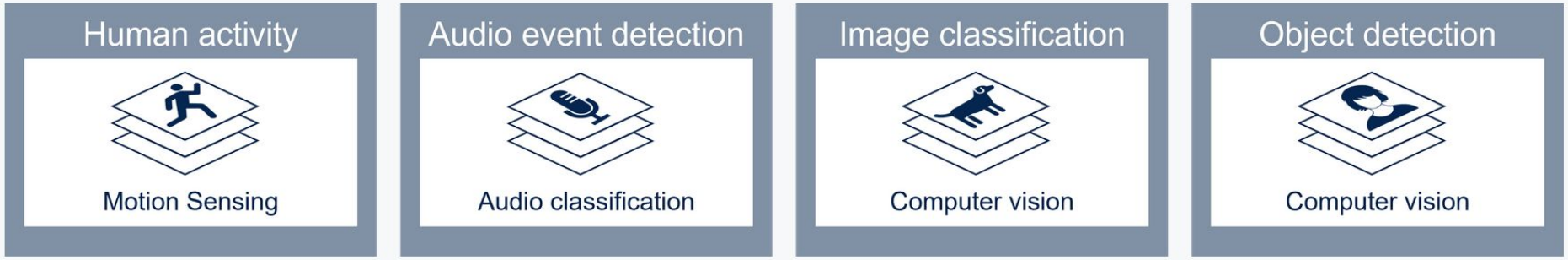
Find the best edge AI model

– A large collection of application-oriented models ready for re-training
– Scripts to easily retrain any model from user datasets
– Application code examples automatically generated from user AI model

STMicroelectronics – STM32 model zoo

Welcome to STM32 model zoo!

The STM32 AI model zoo is a collection of reference machine learning models that are optimized to run on STM32 microcontrollers. Available on GitHub, this is a valuable resource for anyone looking to add AI capabilities to their STM32-based projects.



[STMicroelectronics/stm32ai-modelzoo: AI Model Zoo for STM32 devices]

► **Image classification (IC)**: Models: EfficientNet, MobileNet v1, MobileNet v2, Resnet v1 including with hybrid quantization, SqueezeNet v1.1, STMNIST.

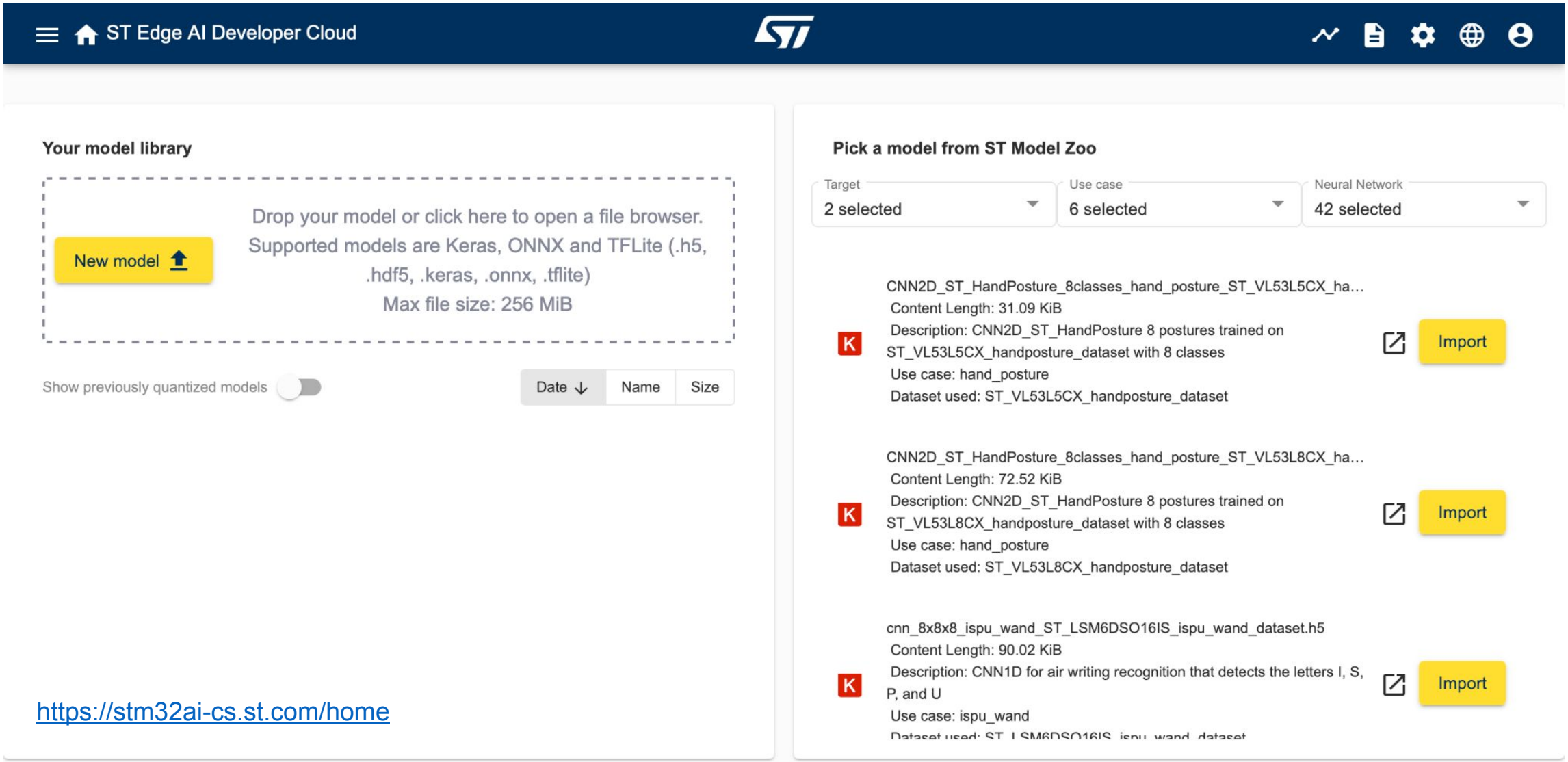
► **Object detection (OD)**: Models: ST SSD MobileNet v1, Tiny YOLO v2, SSD MobileNet v2 fpn lite, ST Yolo LC v1.

► **Human activity recognition (HAR)**: Models: CNN IGN, and CNN GMP for different settings.

► **Audio event detection (AED)**: Models: Yamnet, MiniResnet, MiniResnet v2.

► **Hand posture recognition (HPR)**: Model: ST CNN 2D Hand Posture.

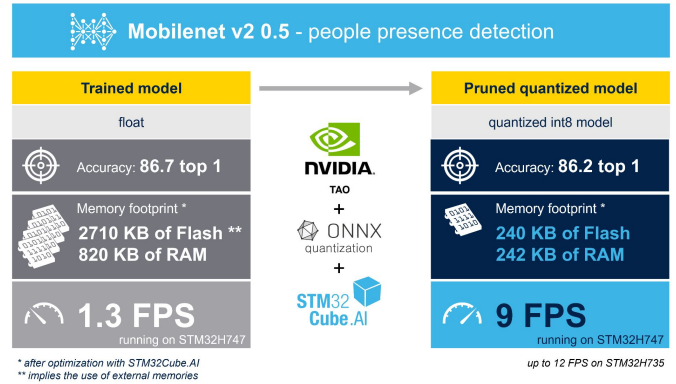
ST Edge AI Developer Cloud



https://stm32ai-cs.st.com/home

STM32AI - Train, Adapt and Optimize!

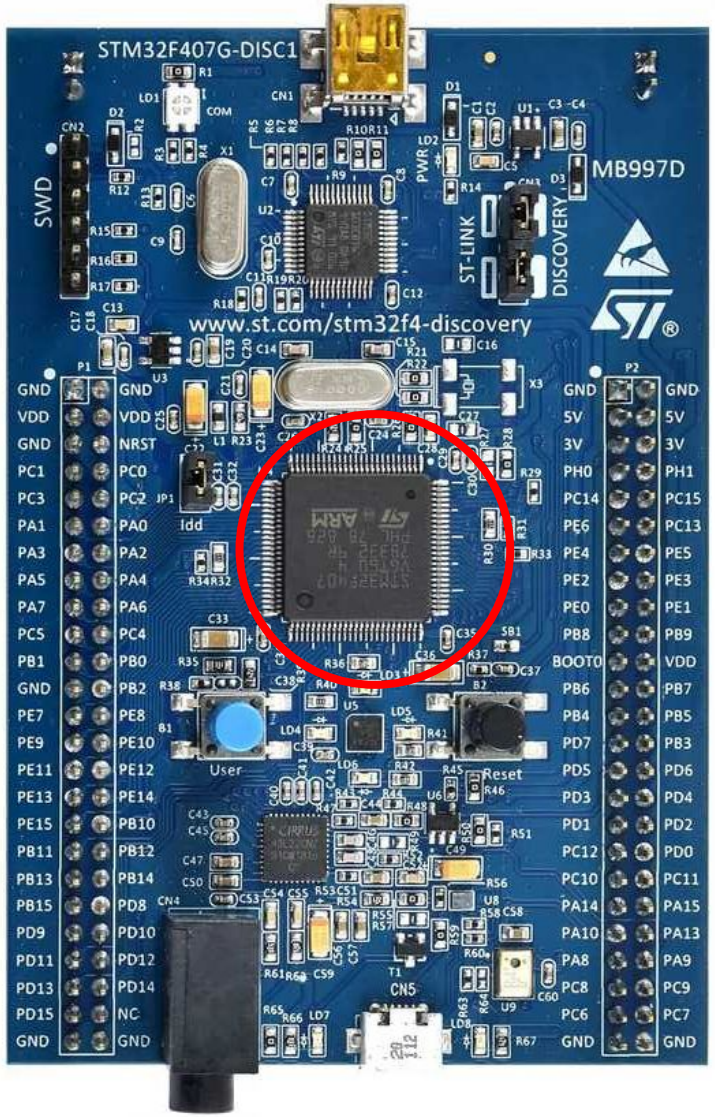
1. Bring your own model weights or choose from NVIDIA's library of model architectures or task-based models
2. Quickly train, adapt, and optimize models with your real or synthetic data
3. Integrate your customized models into your application and deploy

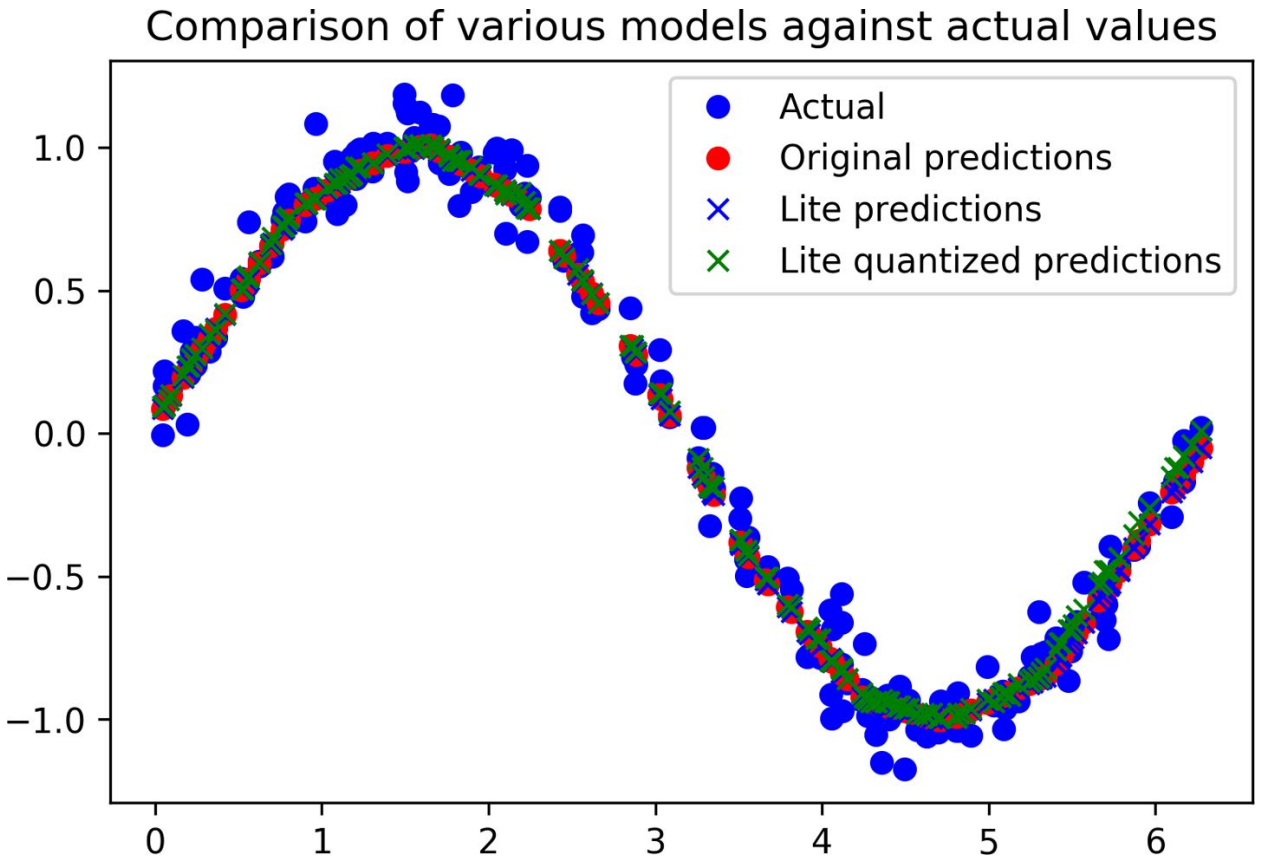


[NVIDIA TAO Toolkit]

[GitHub - STMicroelectronics/stm32ai-tao: Nvidia TAO (Train, Adapt, Optimize) with STM32Cube.AI Developer Cloud]

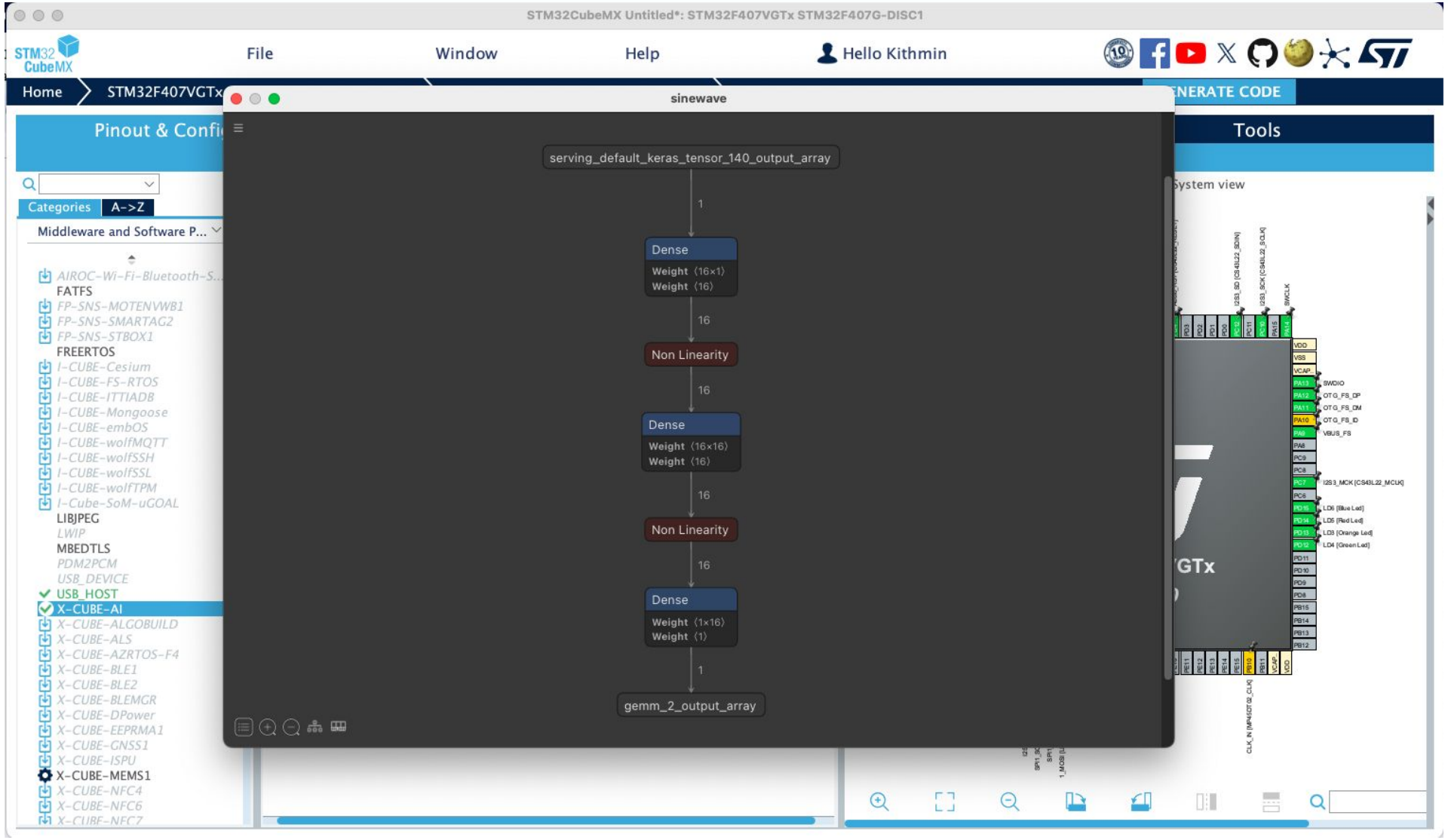
Case Study 1 - tflite_SineWave



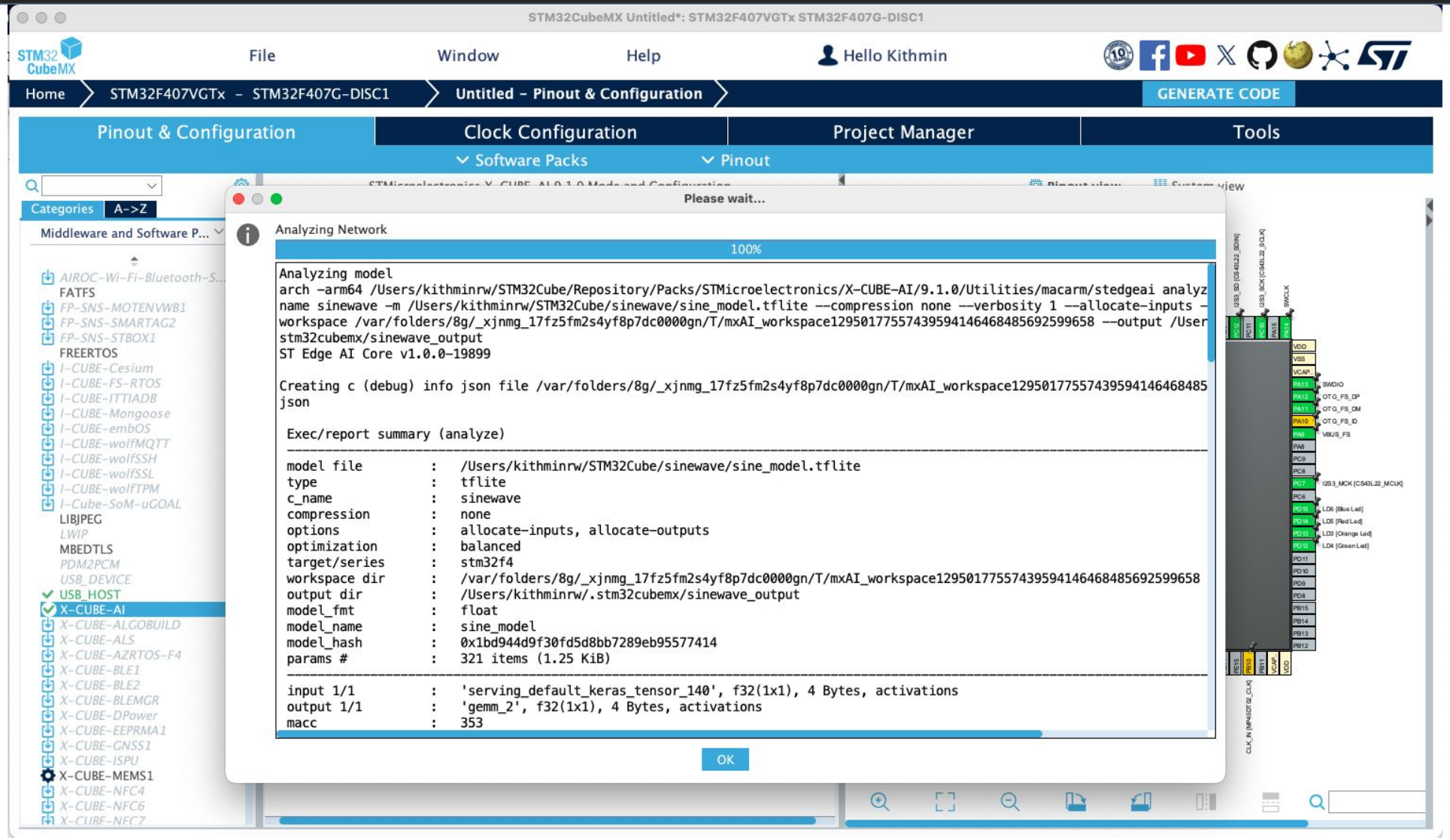


Read - [Chapter 4. The “Hello World” of TinyML: Building and Training a Model]
Video - [TinyML: Getting Started with STM32 X-CUBE-AI | Digi-Key Electronics]
Training Notebook - [TensorFlow Lite Sinewave Regression Training and Conversion] and [Intro to TinyML Part 1: Training a Neural Network for Arduino in TensorFlow | Digi-Key Electronics]

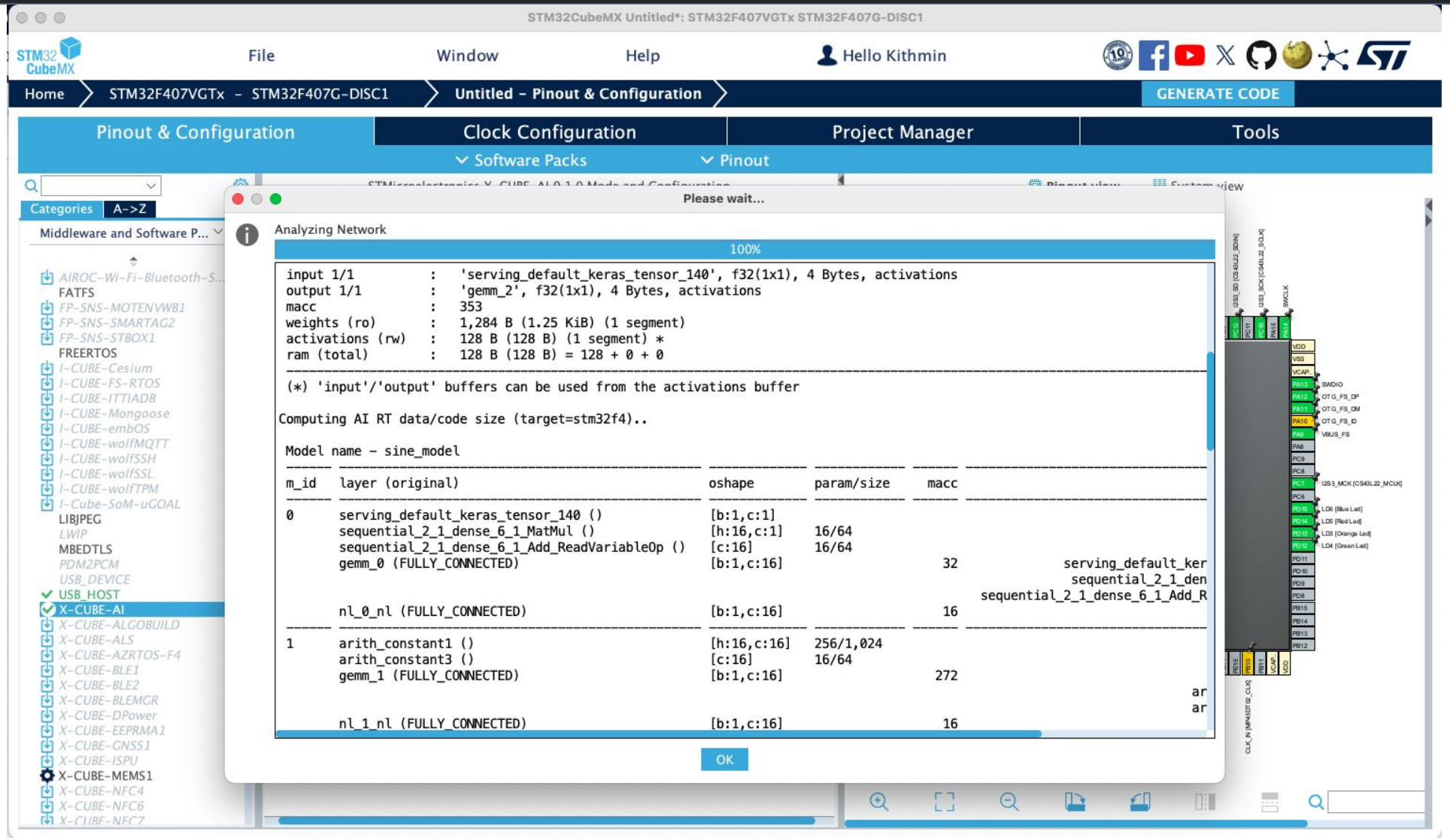
Case Study 1 - tflite_SineWave



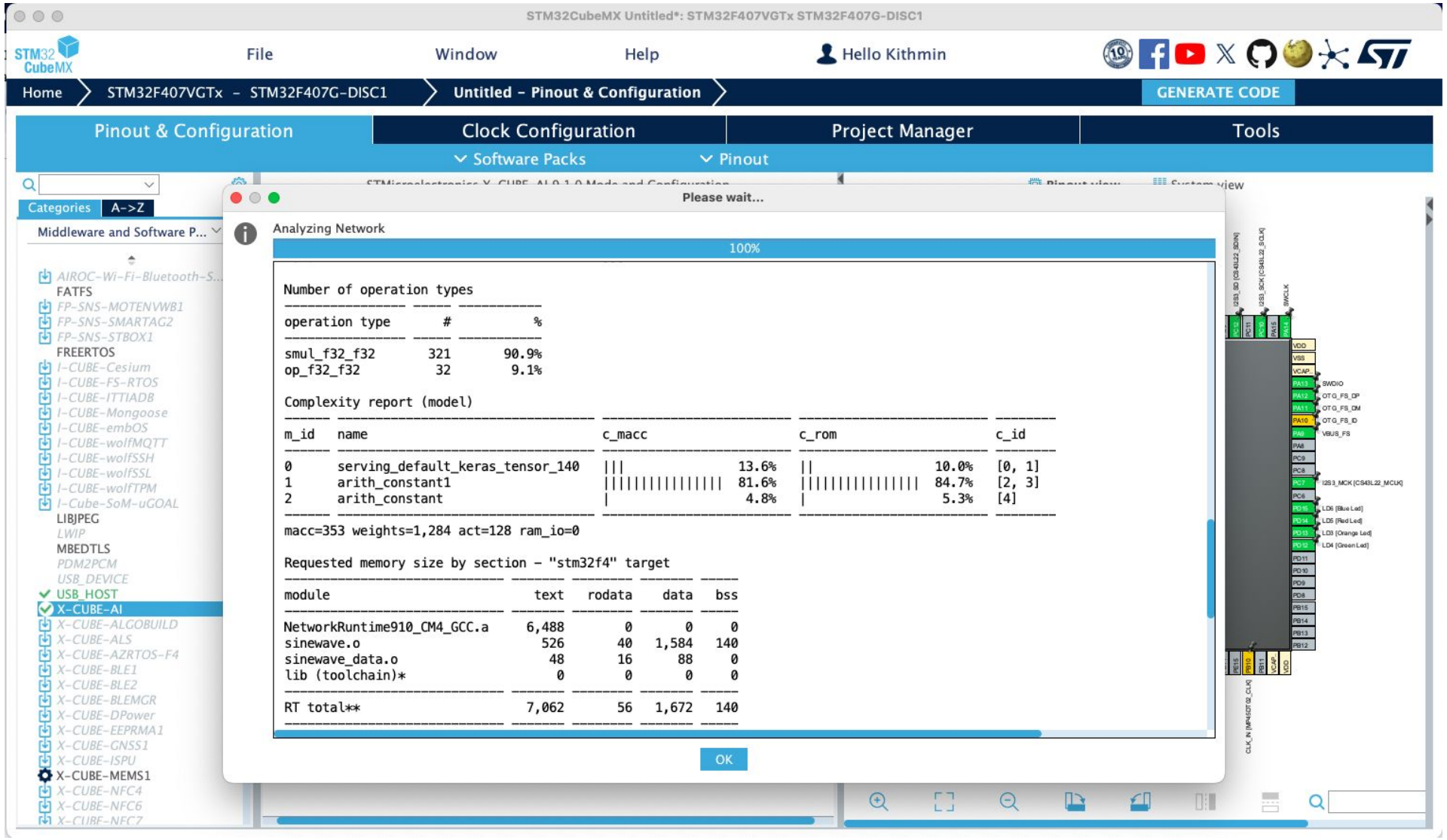
Case Study 1 - tflite_SineWave



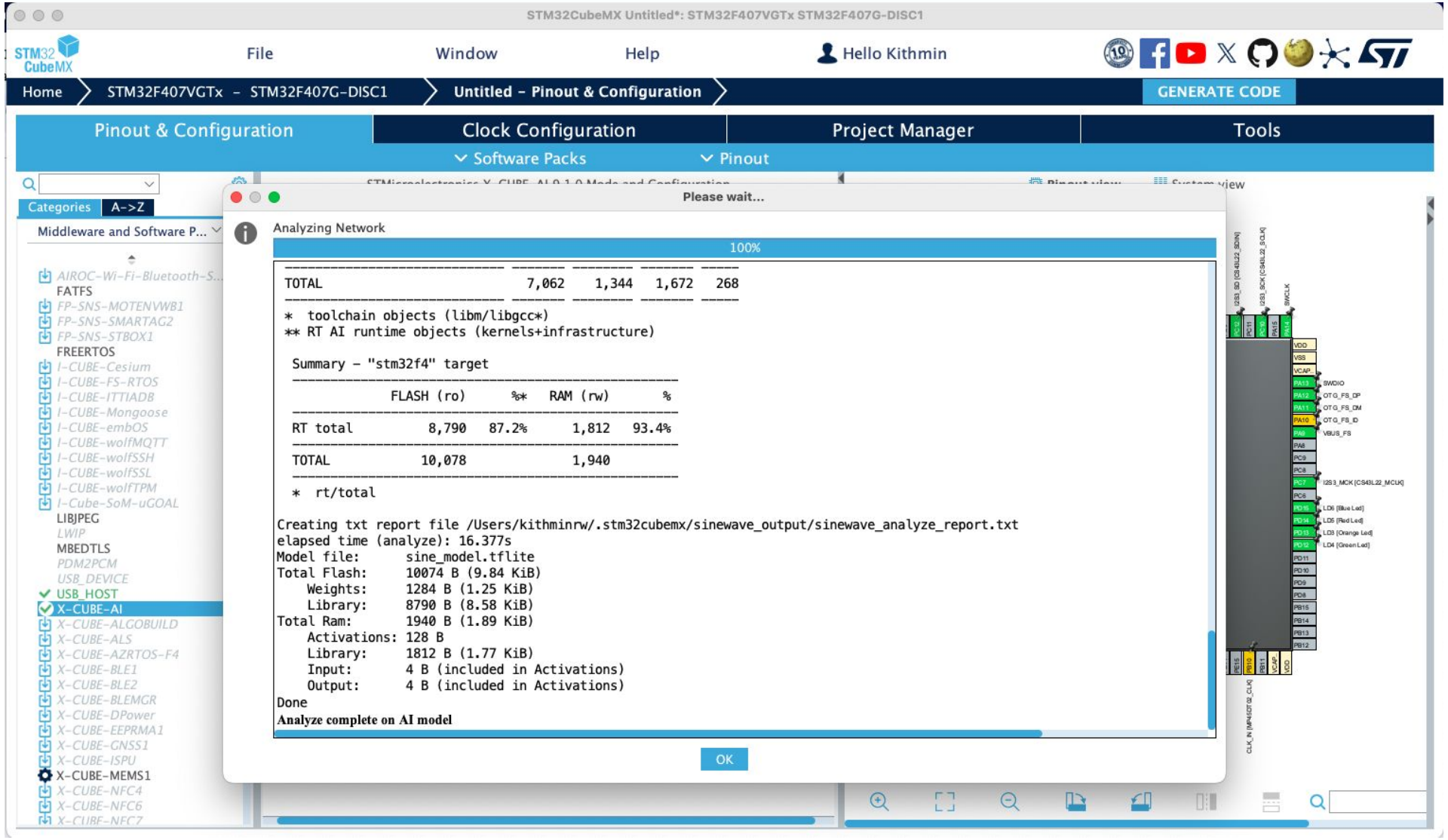
Case Study 1 - tflite_SineWave



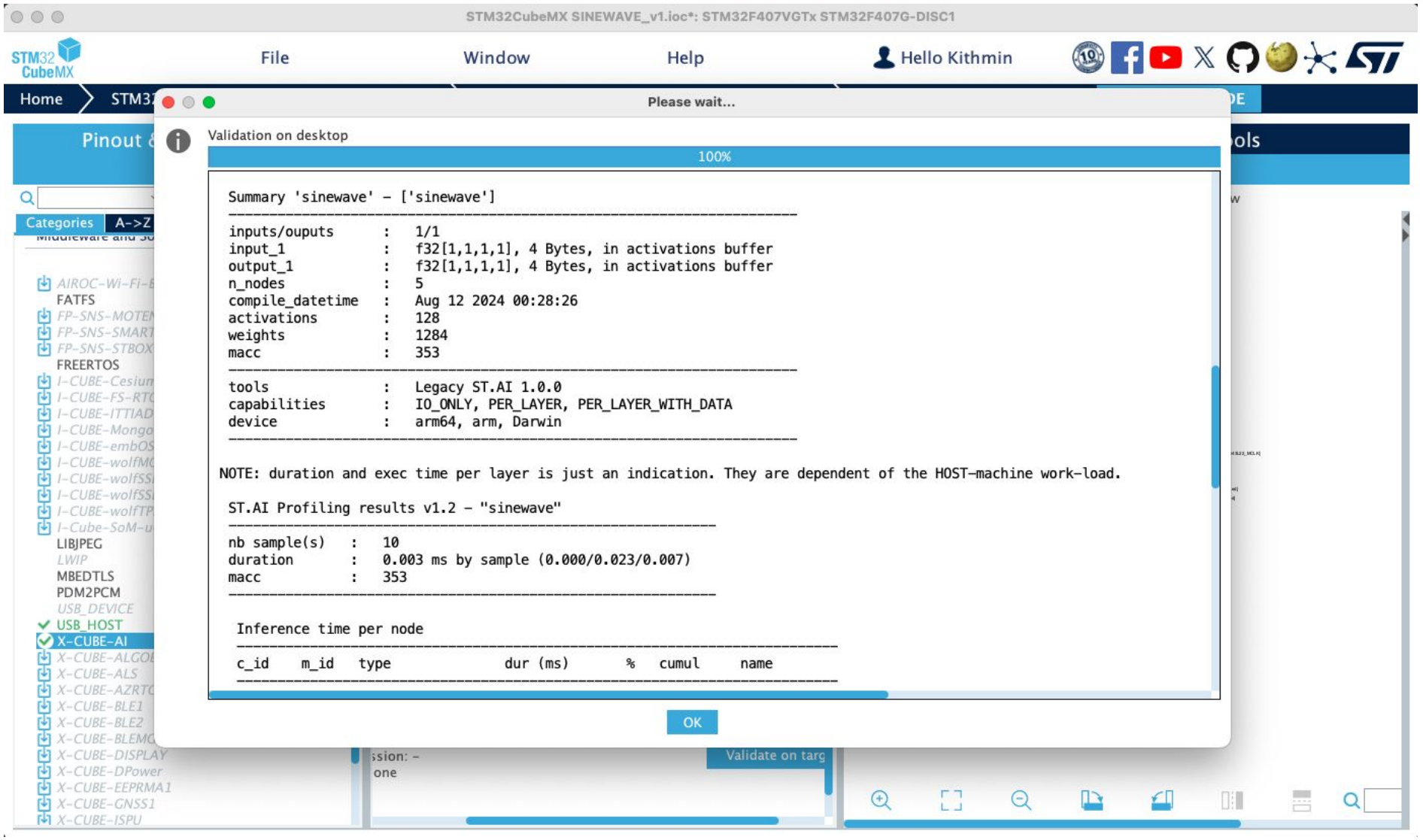
Case Study 1 - tflite_SineWave



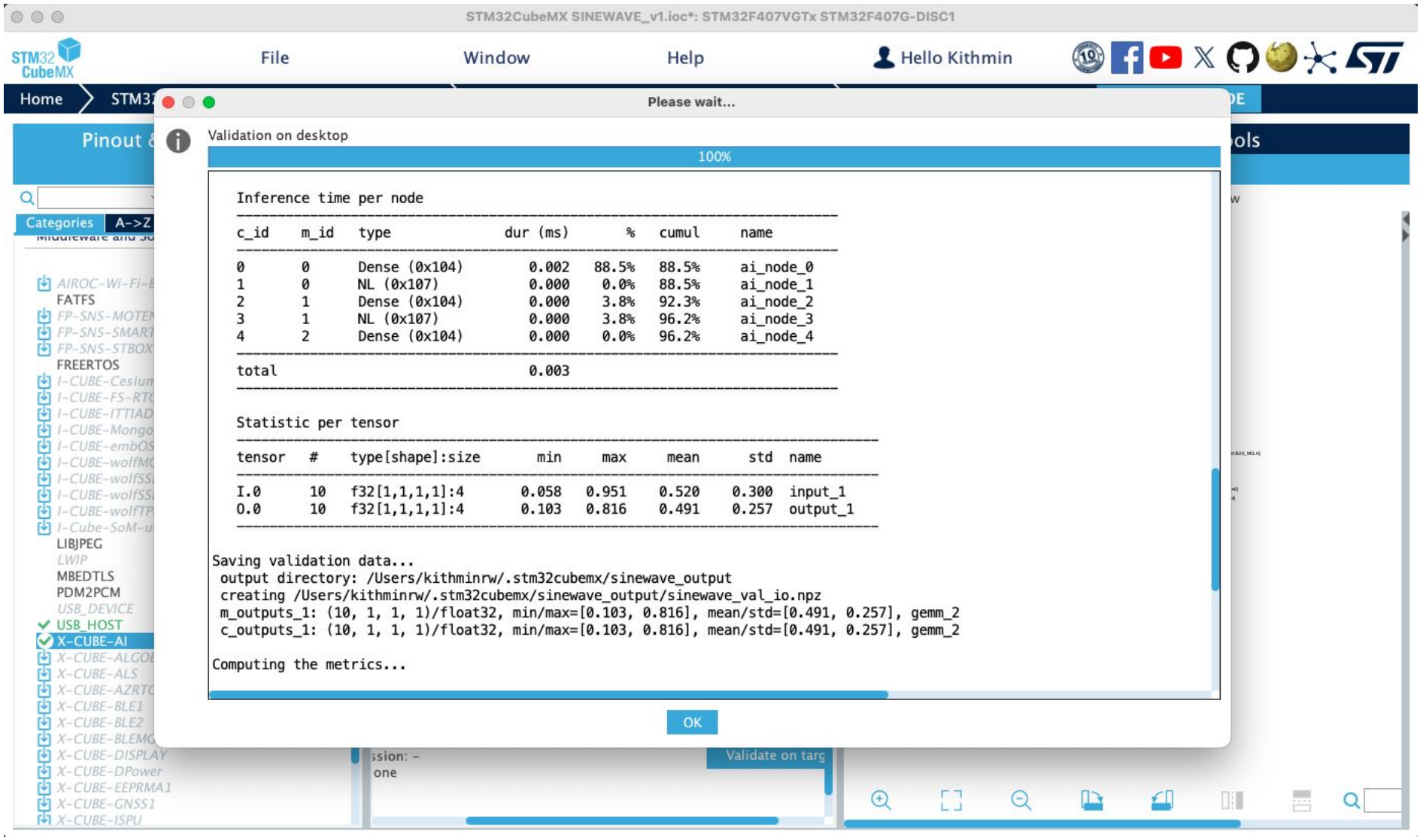
Case Study 1 - tflite_SineWave



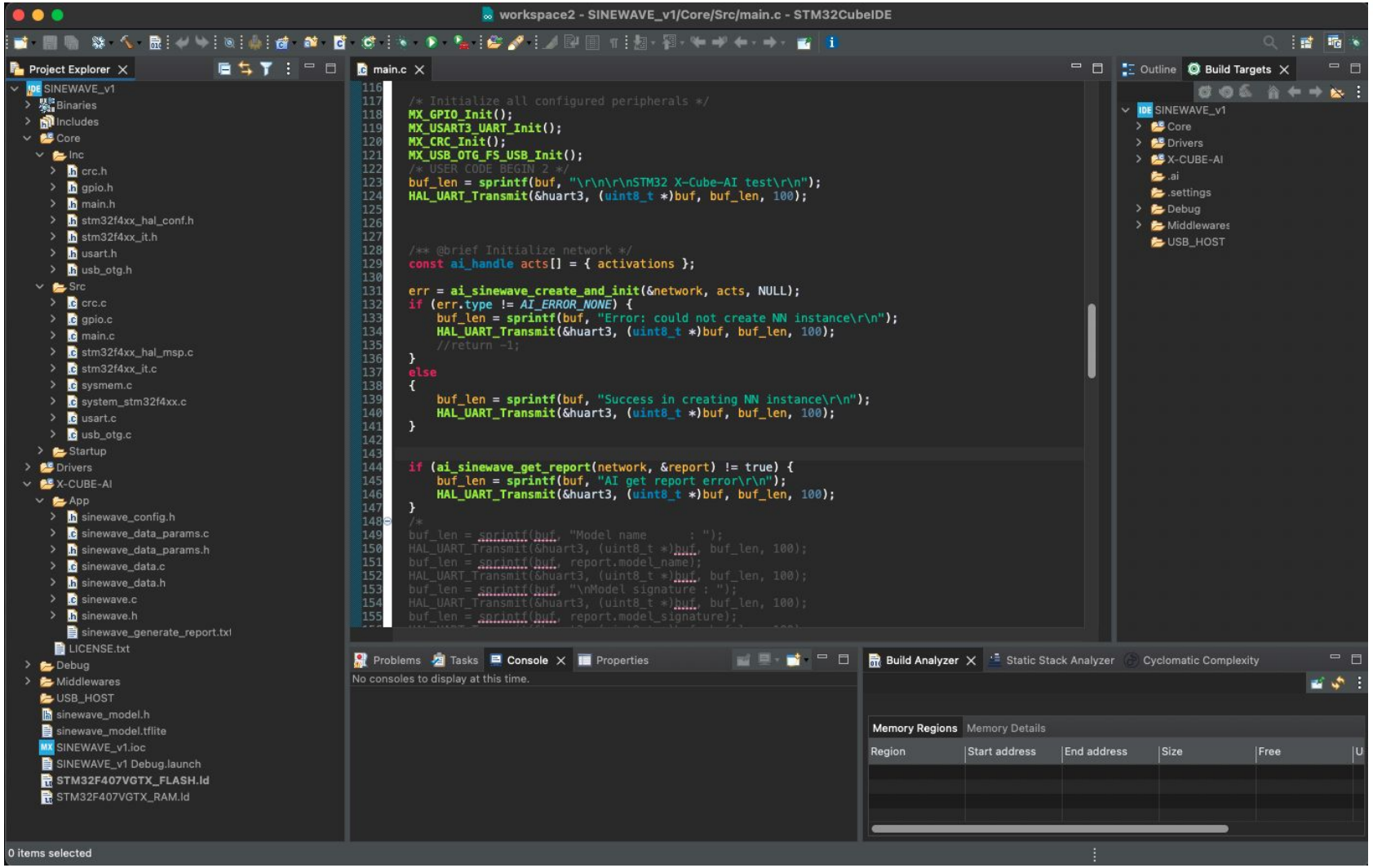
Case Study 1 - tflite_SineWave



Case Study 1 - tflite_SineWave

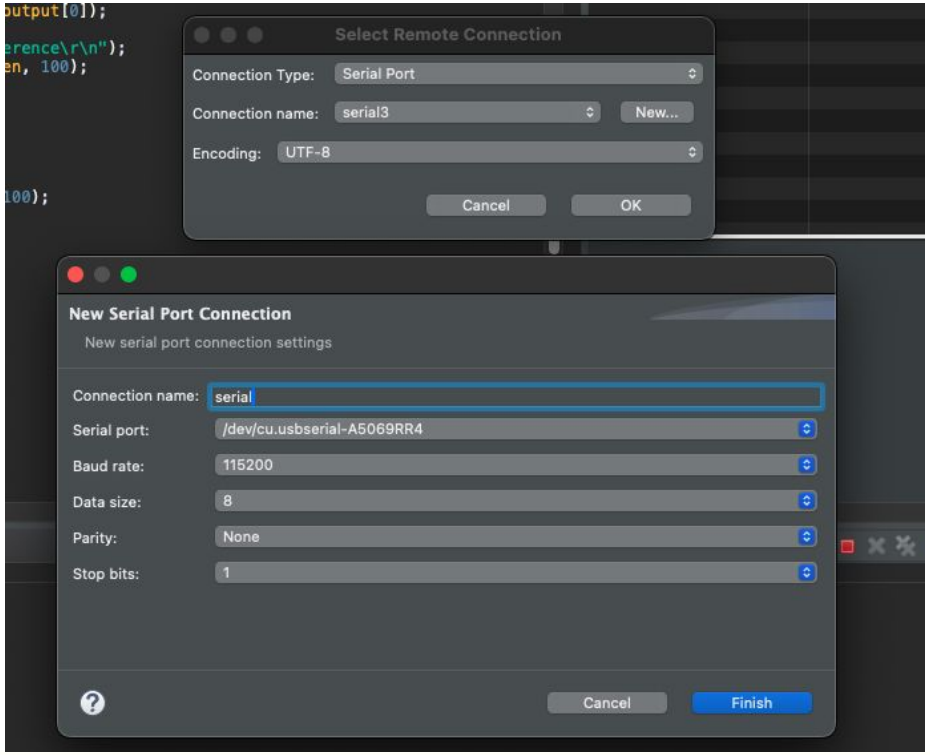


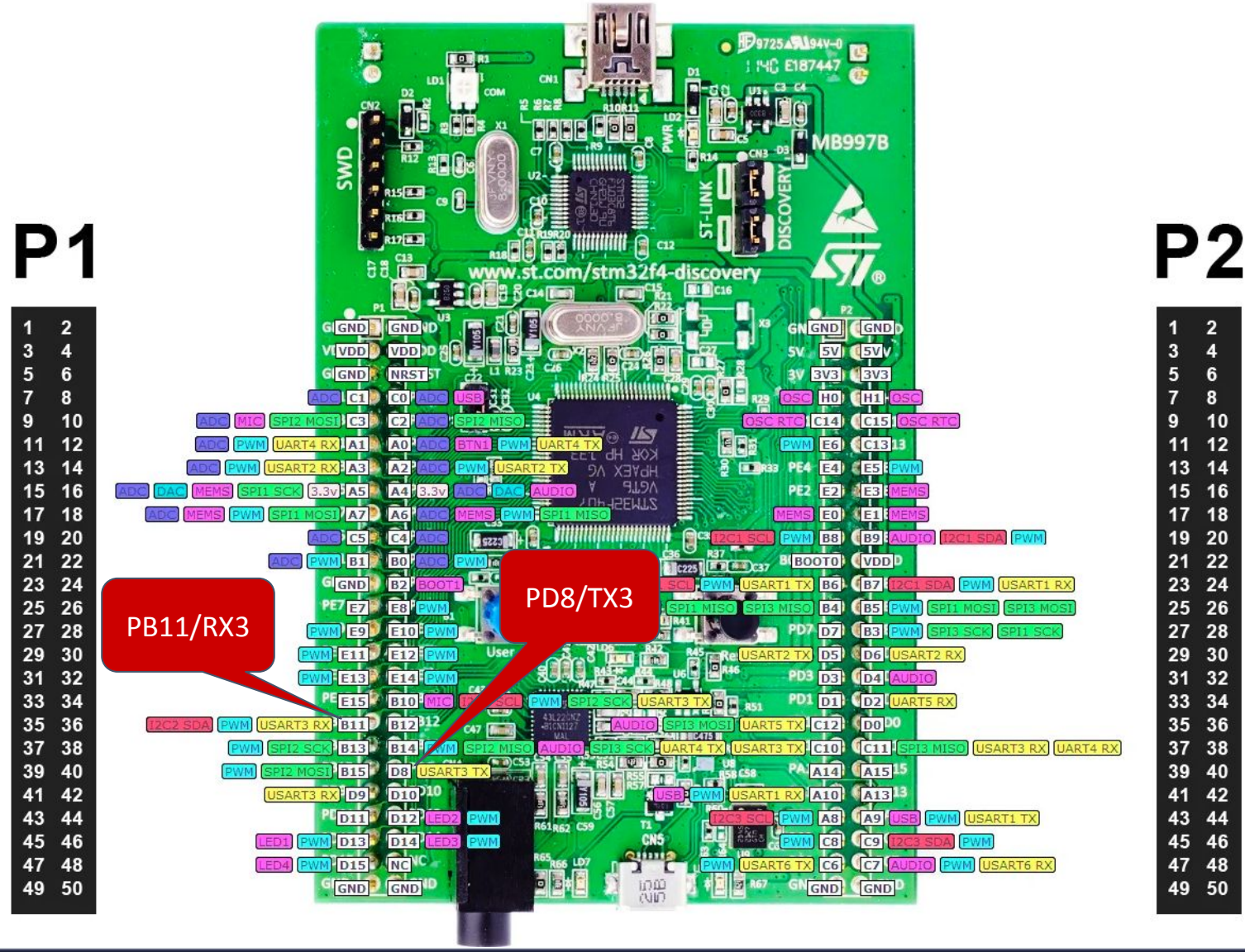
Case Study 1 - tflite_SineWave



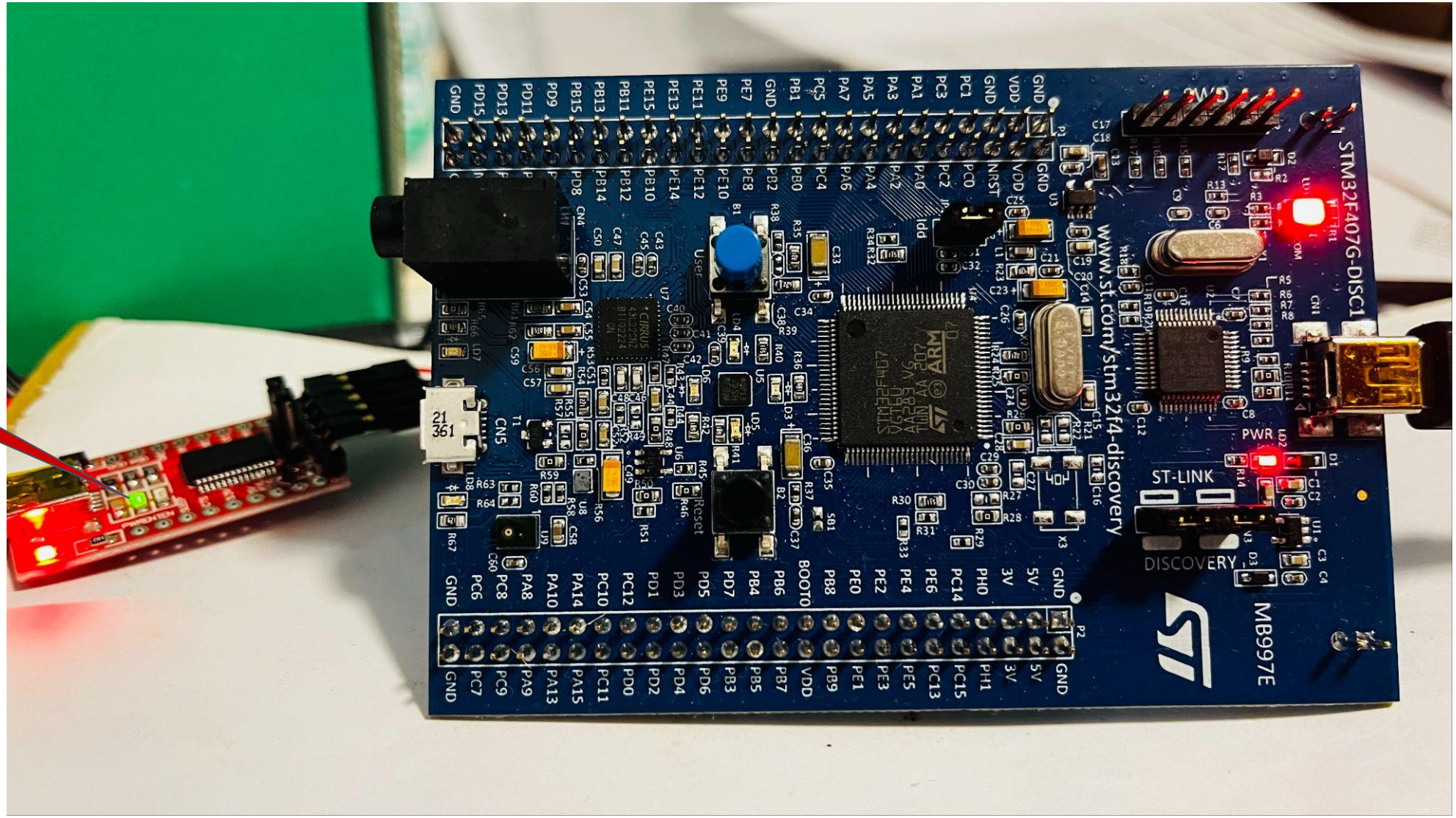
https://github.com/SkillSurf/ML4_edgeAI/tree/main/STM32_cubeAI-sinewave

Case Study 1 - tflite_SineWave





Case Study 1 - tflite_SineWave



Case Study 1 - tflite_SineWave

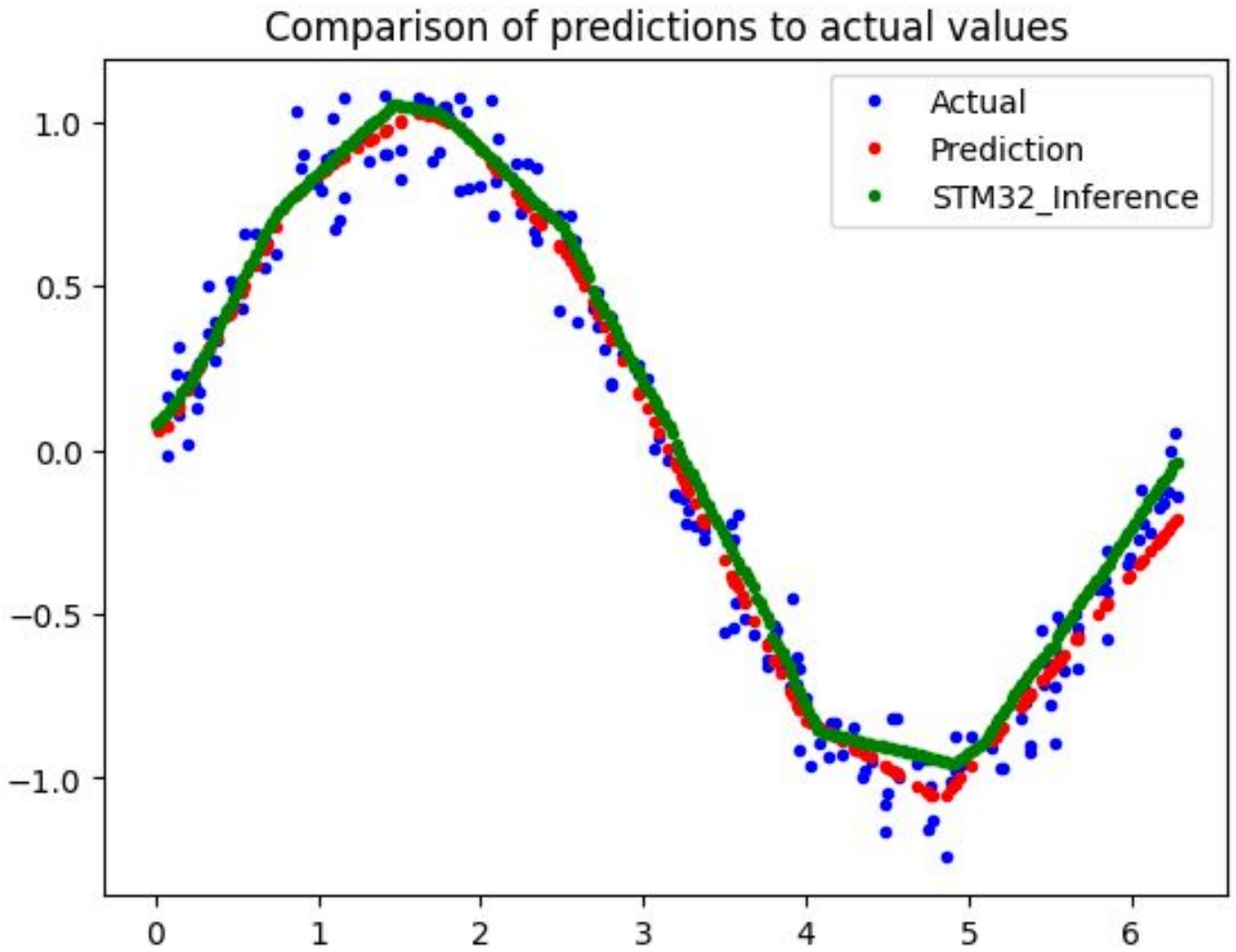
main.c | sinewave.c | usart.c

```
while (1){
    float x = 0.0f;
    buf_len = sprintf(buf, "\r\n\r\nStarting inference... \r\n");
    HAL_UART_Transmit(&huart3, (uint8_t *)buf, buf_len, 100);
    for (int i = 0 ; i < 628; i++){  // Adjust the number 628 to create more sampling points
        x = x + 0.01; // 628 x 0.01 = 6.28; Always keep the multiplication to 6.28 to create one sine wave

        for (uint32_t j = 0; j < AI_SINEWAVE_IN_1_SIZE_BYTES; j++)
            {
                ((ai_float *)in_data)[j] = (ai_float)x;
            }

        nbatch = ai_sinewave_run(network, &ai_input[0], &ai_output[0]);
        if (nbatch != 1) {
            buf_len = sprintf(buf, "Error: could not run inference\r\n");
            HAL_UART_Transmit(&huart3, (uint8_t *)buf, buf_len, 100);
          }

        y_val = ((float *)out_data)[0];

        buf_len = sprintf(buf, "%f \r\n", y_val);

        HAL_UART_Transmit(&huart3, (uint8_t *)buf, buf_len, 100);

        HAL_Delay (10);
    }
    while(1);
```

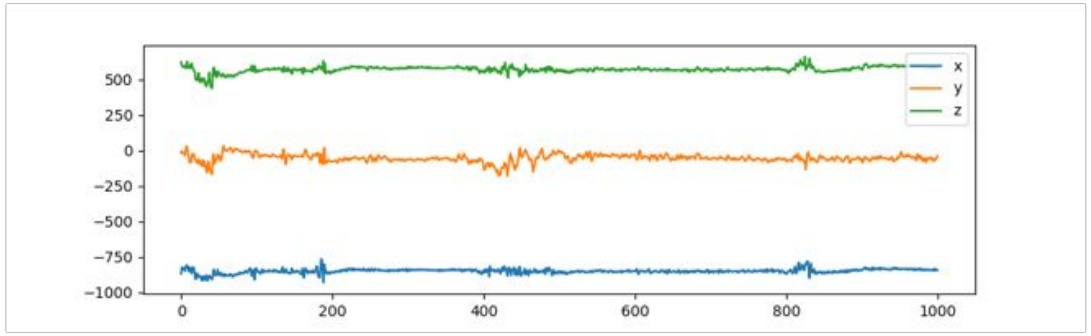
Problems | Tasks | Console | Properties

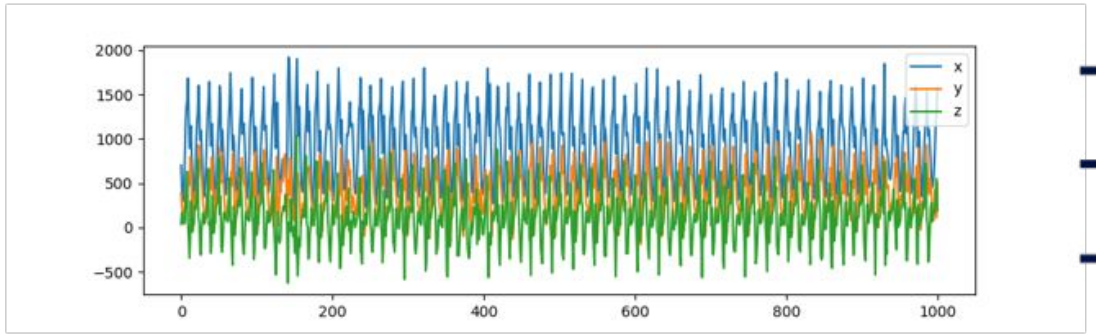
serial3 1 (CONNECTED)

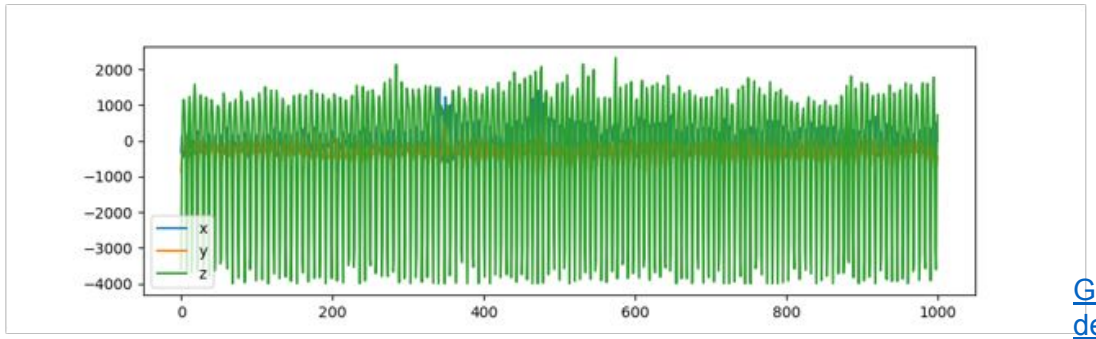
```
Starting inference...
0.079104
0.089237
0.099370
0.109729
0.121459
0.133441
0.145423
0.159124
0.173863
0.188898
0.197630
0.215096
0.232562
0.250211
0.268119
0.286028
0.304675
0.324401
0.344126
0.363852
0.383577
0.403303
0.423029
0.442754
0.462480
0.482205
0.501931
0.521657
0.541382
0.561108
0.580833
0.600559
0.628366
```



Case Study 2 - Activity Recognition







[GitHub - teomaras76/AI_on_STM32_Gesture_recognition: Gesture recognition Deep Learning project for STM32 (based on Keras model)]

Stationary

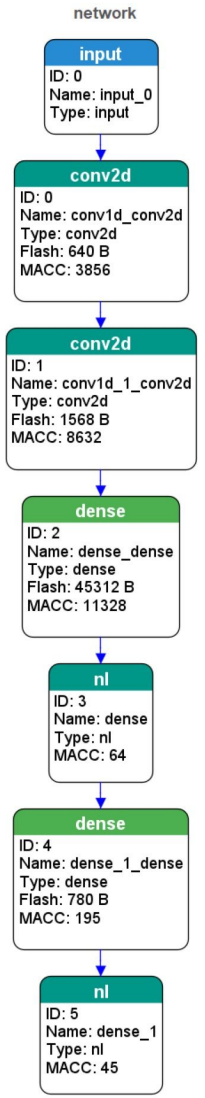
Walking

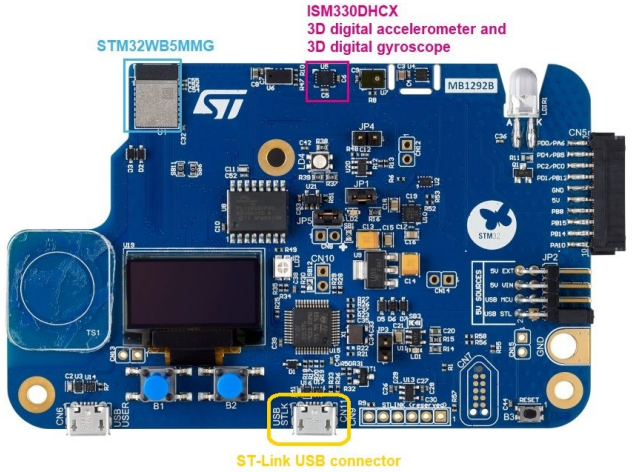
Running

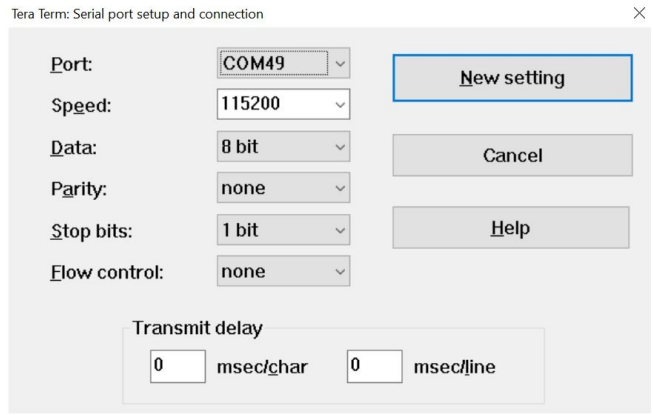
STM32 Cube.AI

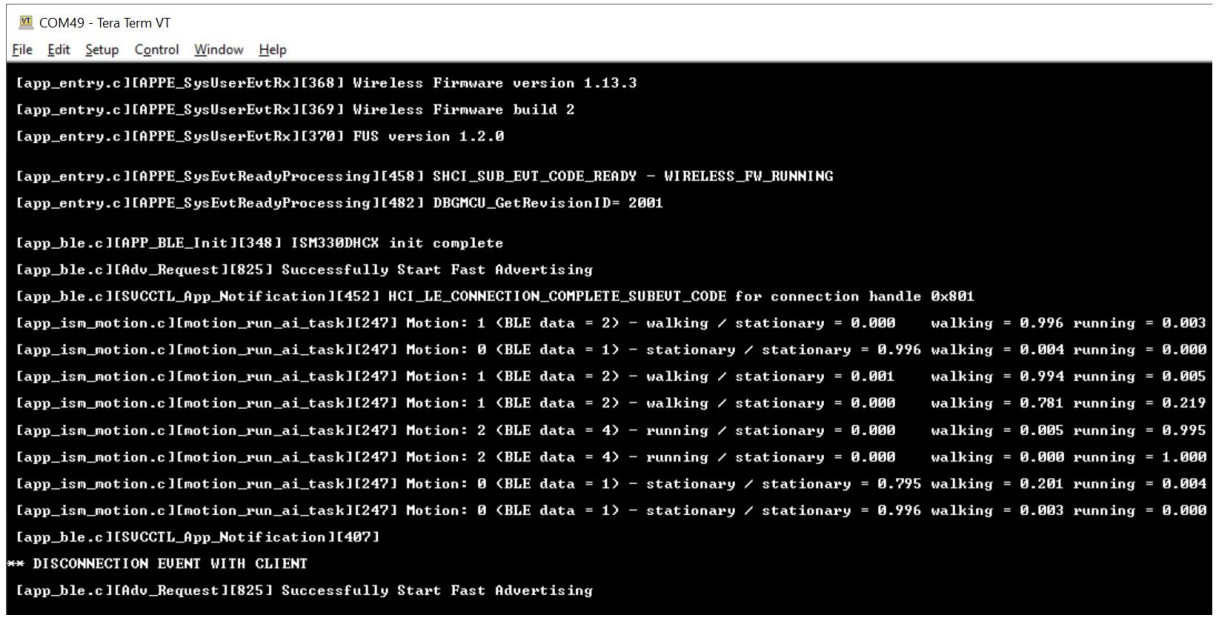
[GitHub - stm32-hotspot/STM32WB-BLE-AI-MotionSense: Example demonstrates how to create a motion sensing application to recognize human activities using machine learning on an STM32WB microcontroller]

Case Study 2 - Activity Recognition



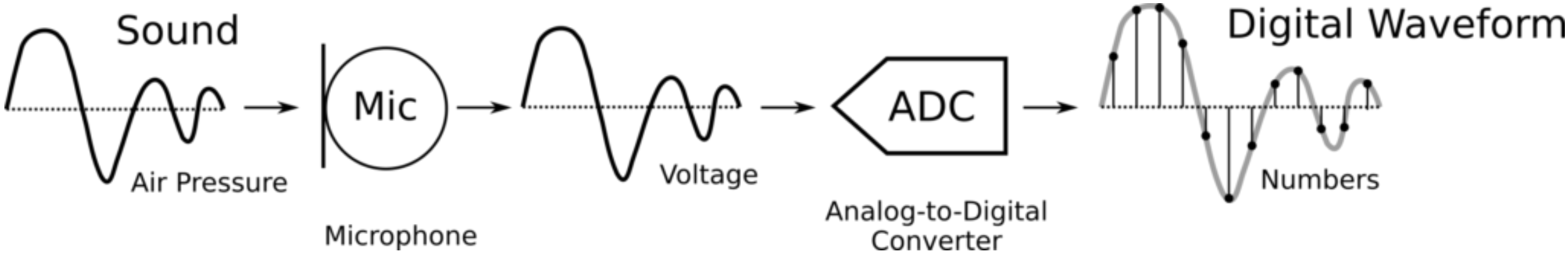


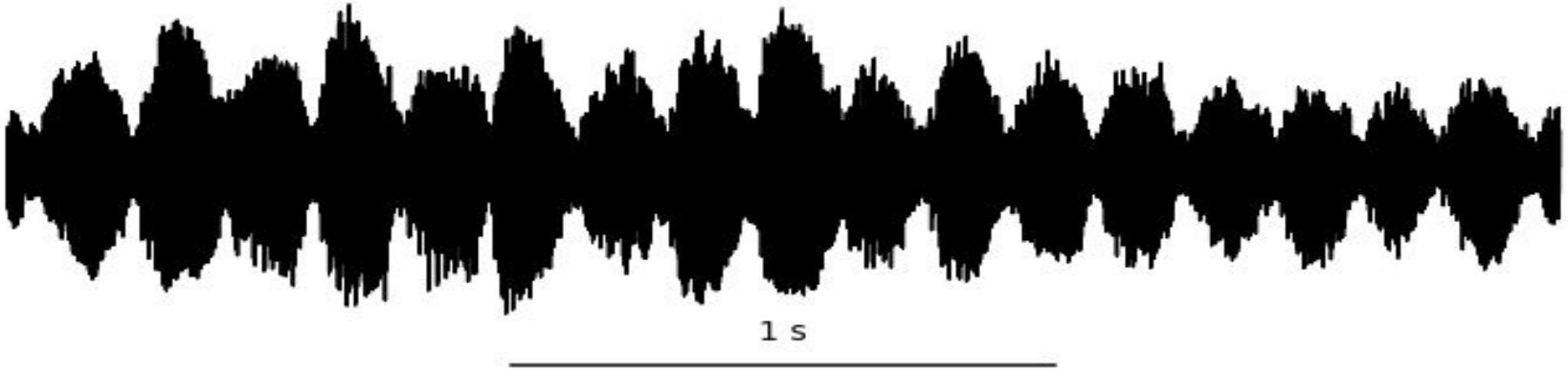




Case Study 2 - Activity Recognition

Case Study 3 - Audio classifier





Case Study 3 - Audio classifier

Microphone

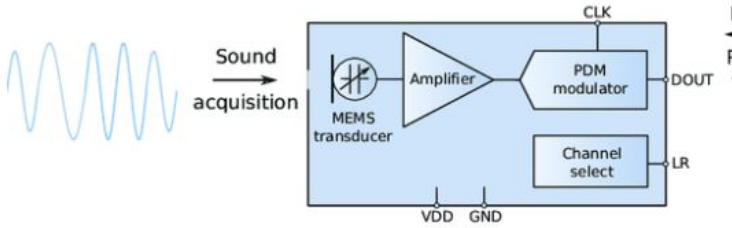
The board is equipped with a **MP45DT02** MEMS microphone, which produces a stream of 1-bit **Digital Pulse Modulation** (PDM) samples. The PDM samples are received using DMA and then converted to 16-bit signed **Pulse Code Modulation** (PCM) samples through the usage of a software low pass filter.
The PCM data is sampled at a frequency of 32 kHz, which is obtained by setting the *PLLI2SN*, *PLLI2SR* and *I2SDIV* registers to appropriate values (see chapters *7.3.23* and *28.4.4* of the datasheet for further details):

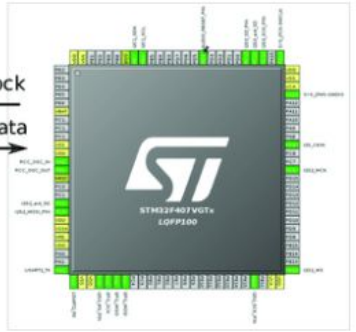
$$f_{(\text{VCO clock})} = f_{(\text{PLLI2S clock input})} * \frac{PLLI2SN}{PLLM} = 8\ \text{MHz}\ \frac{213}{8} = 213\ \text{MHz}$$

$$f_{(\text{PLL I2S clock output})} = \frac{f_{(\text{VCO clock})}}{PLLI2SR} = \frac{213\ \text{MHz}}{2} = 106,5\ \text{MHz}$$

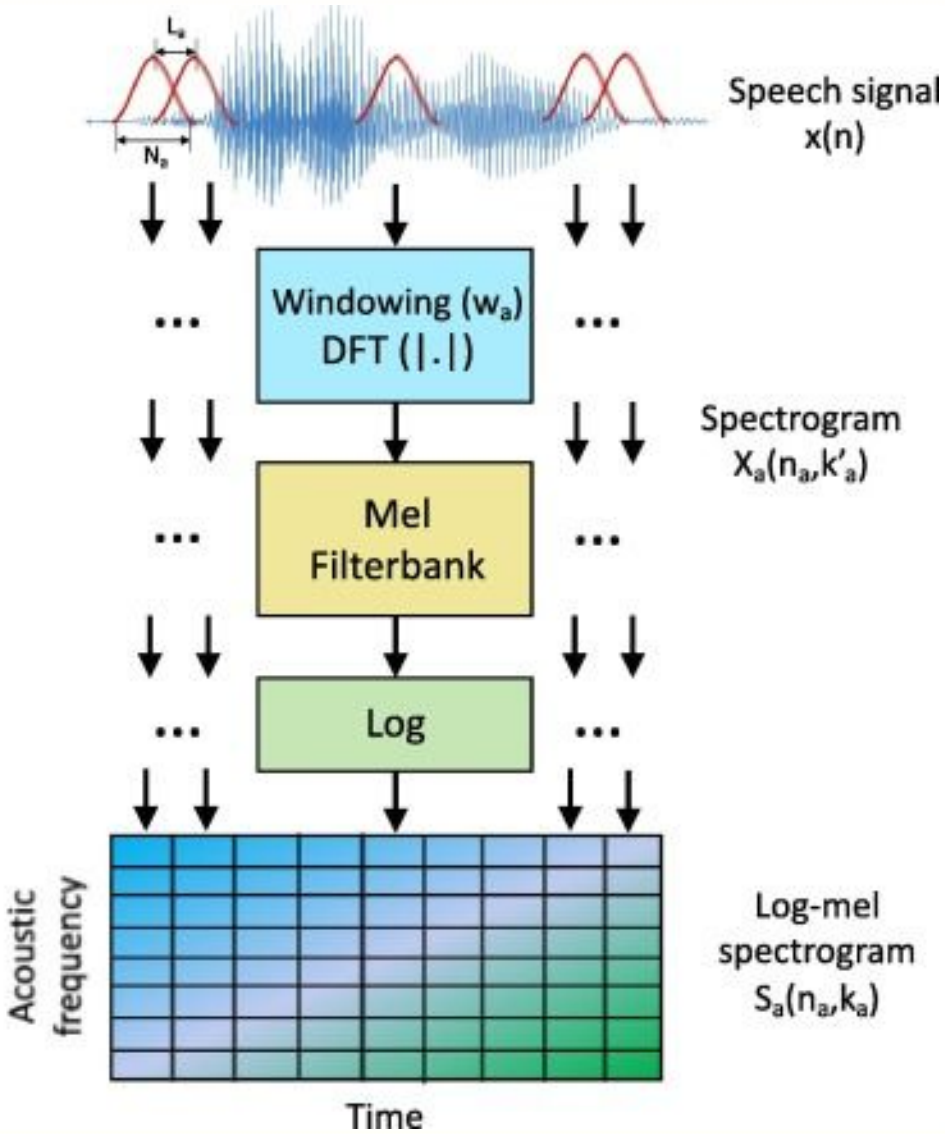
$$f_{(\text{PDM})} = \frac{f_{(\text{PLL I2S clock output})}}{2 * I2SDIV * 8} = \frac{106,5\ \text{MHz}}{2 * 13 * 8} = 512\ \text{kHz}$$

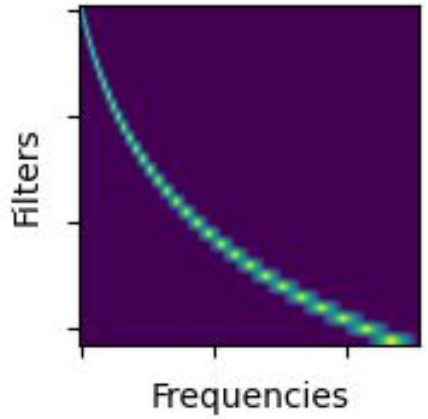
$$f_{(\text{PCM})} = \frac{f_{(\text{PDM})}}{16} = \frac{512\ \text{kHz}}{16} = 32\ \text{kHz}$$

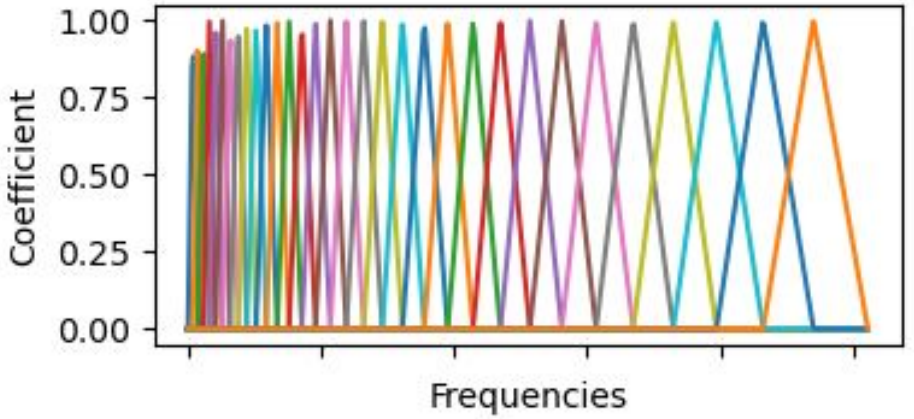


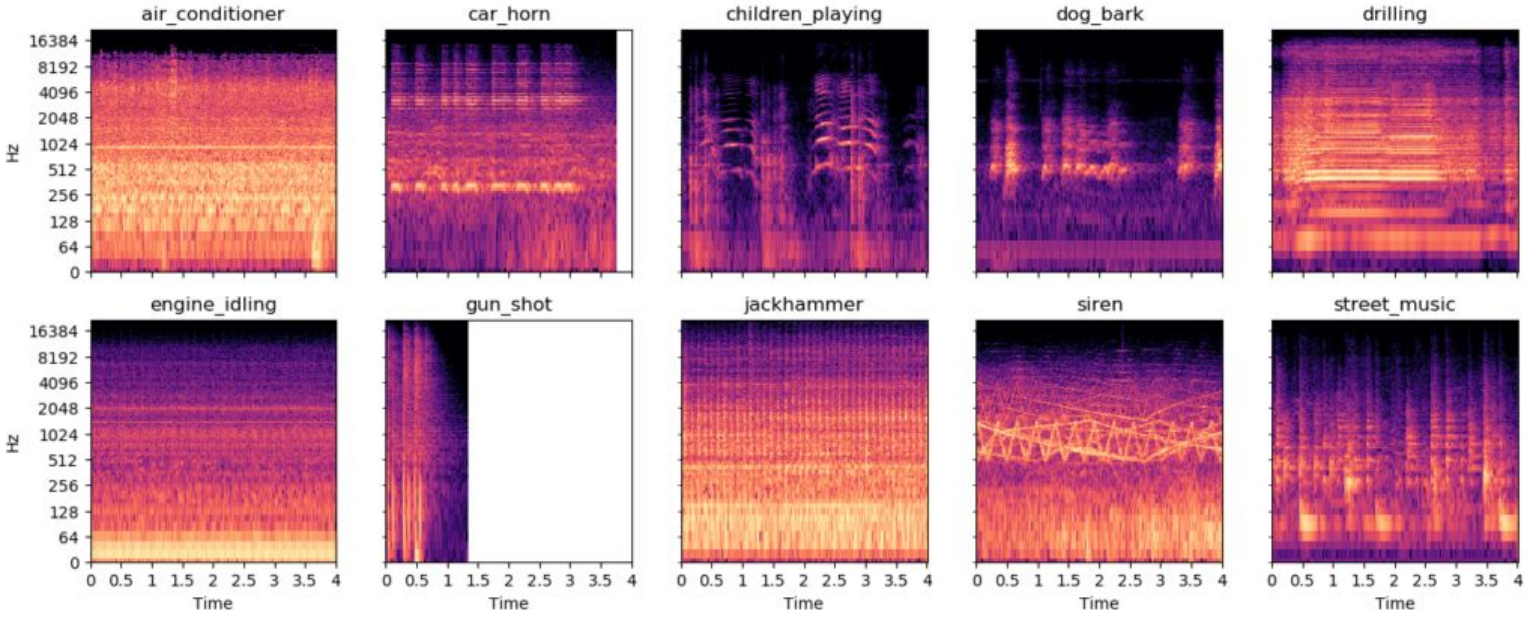


Case Study 3 - Audio classifier

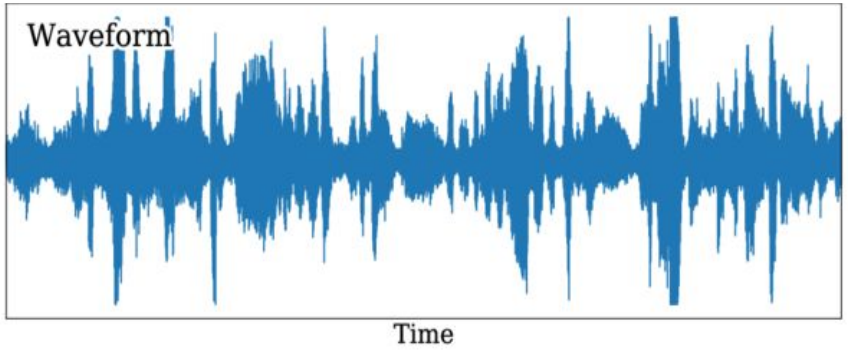


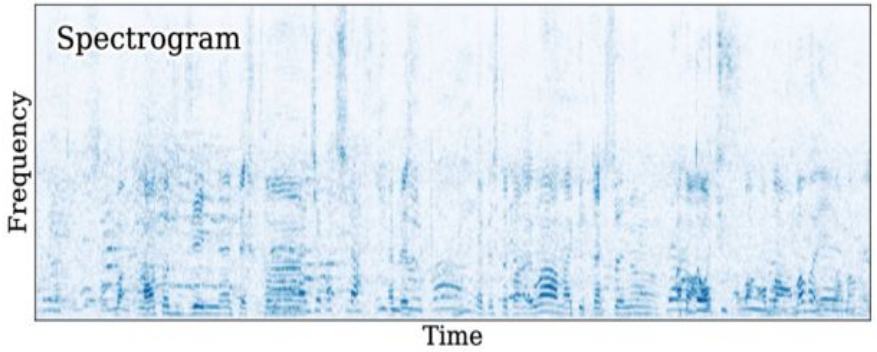


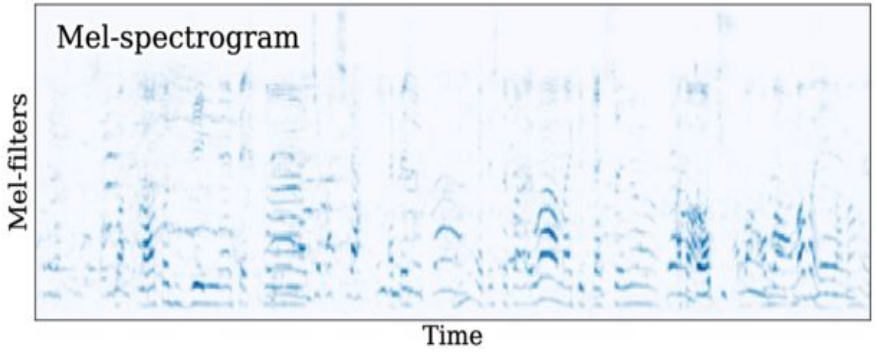


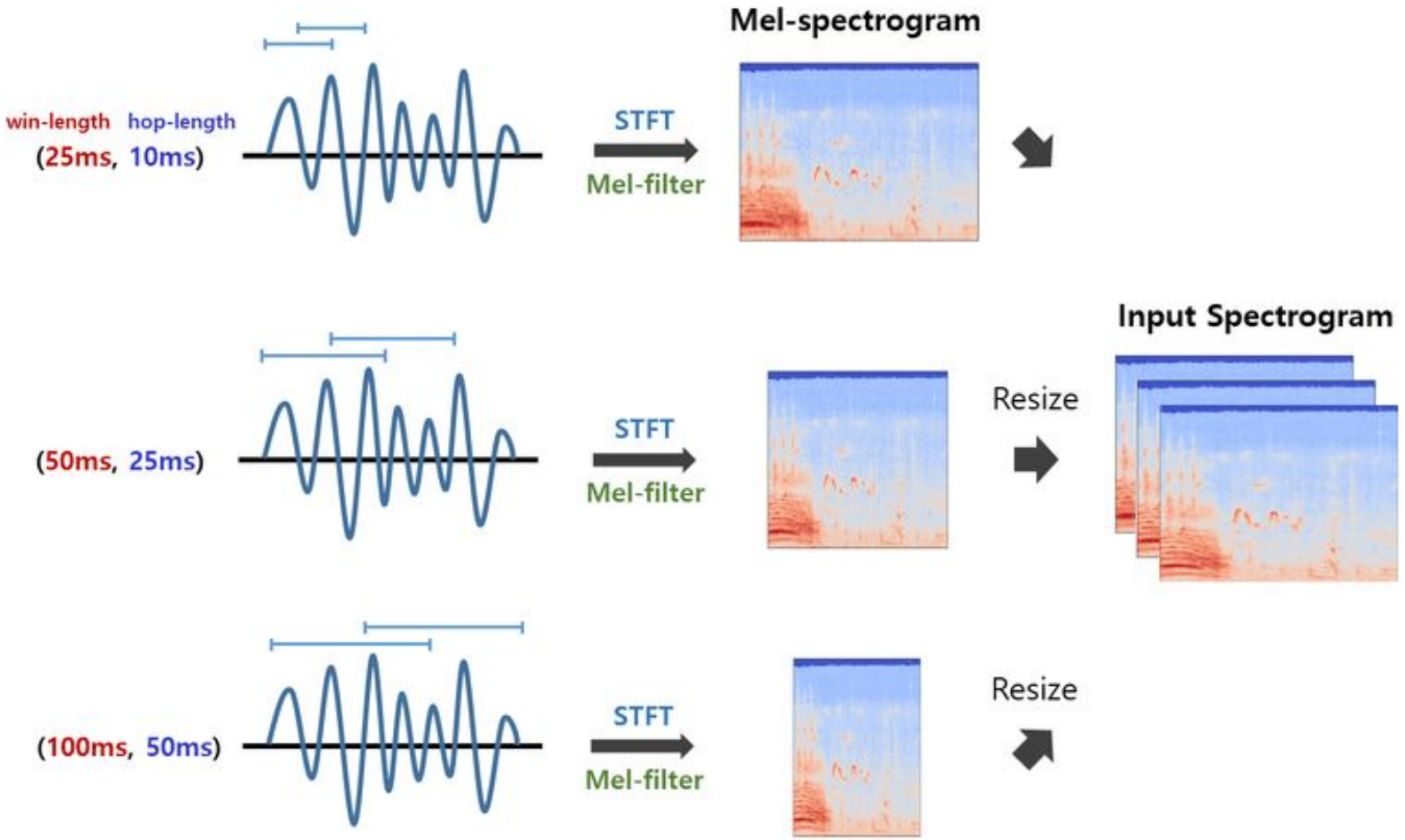


Case Study 3 - Audio classifier

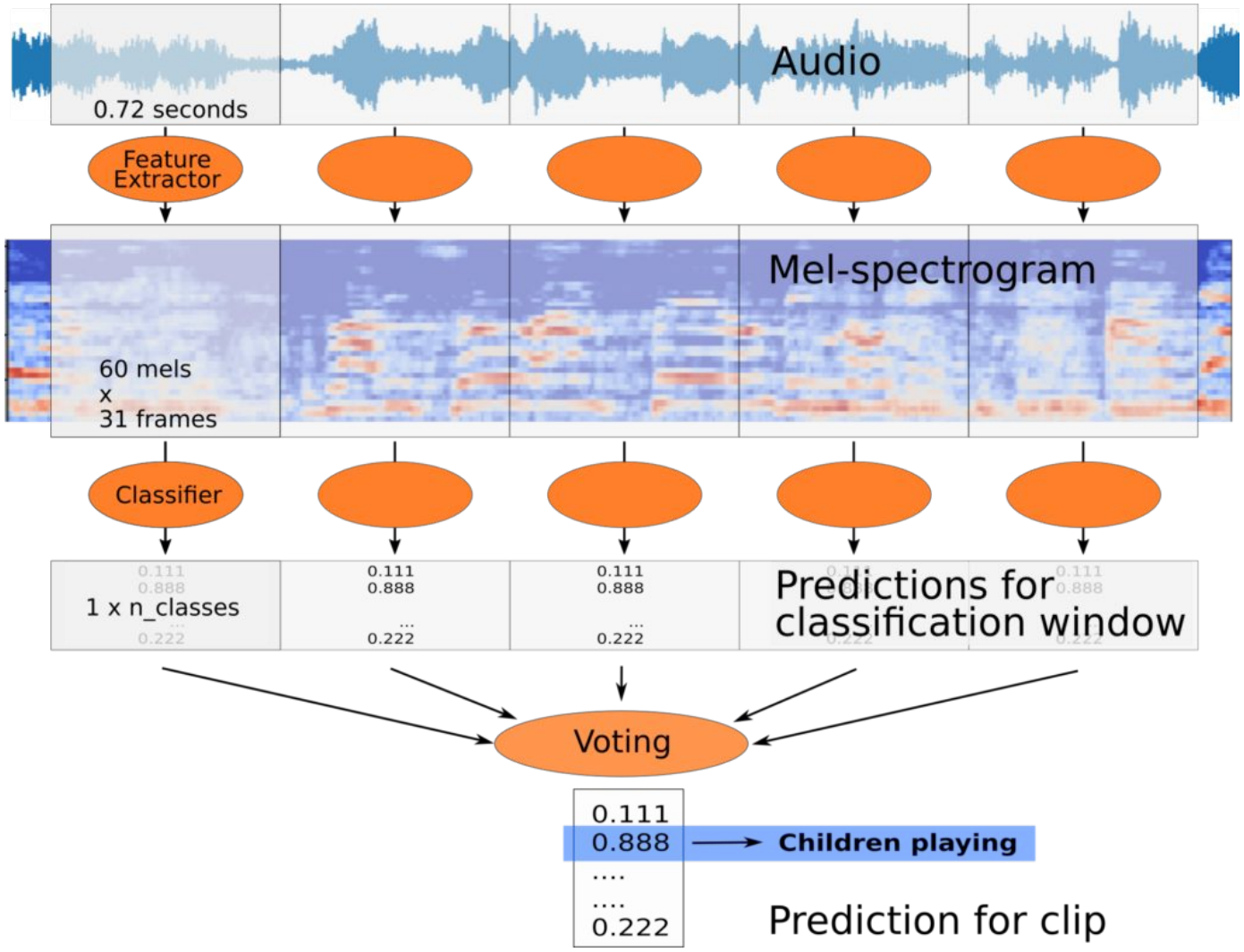


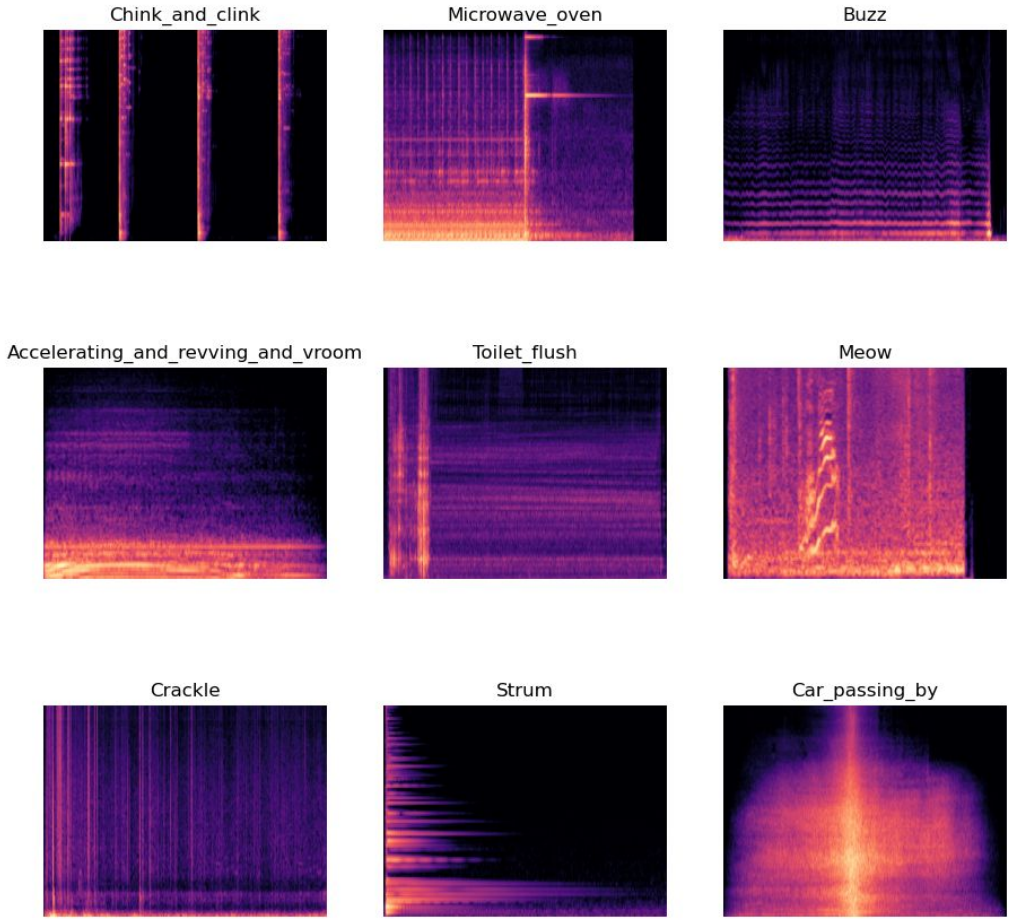






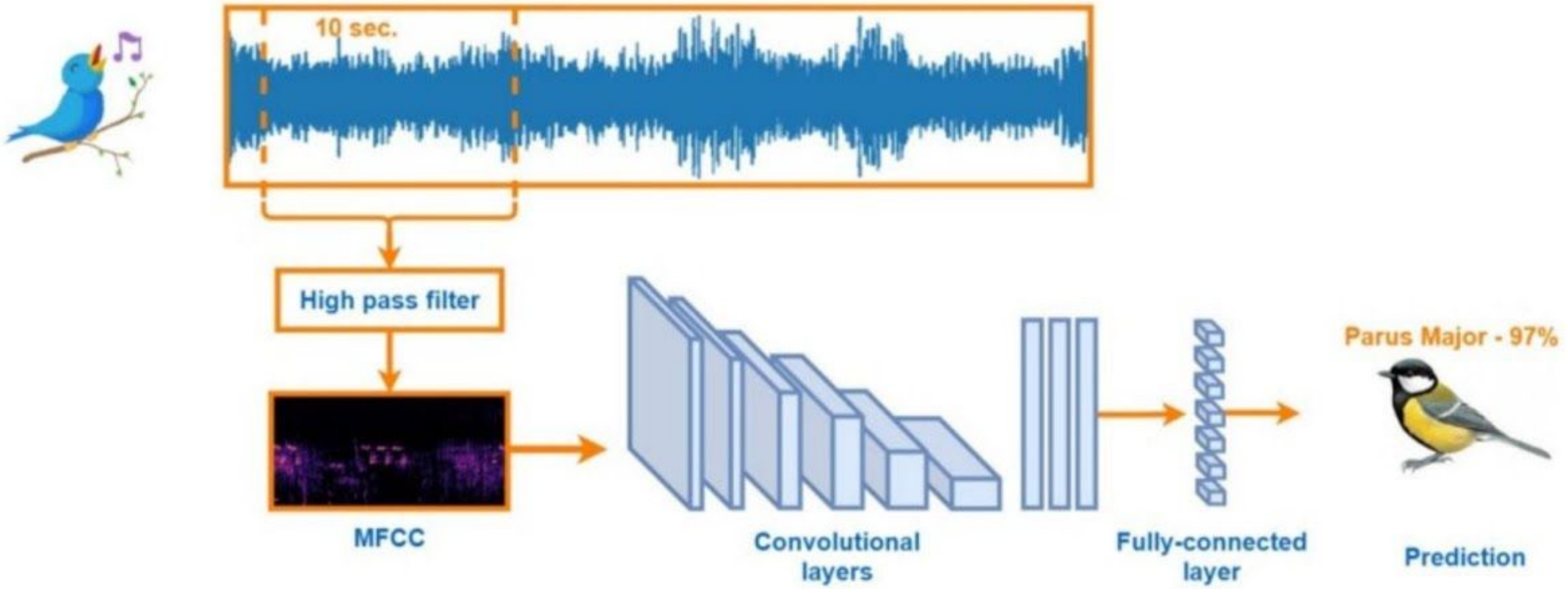
Case Study 3 - Audio classifier





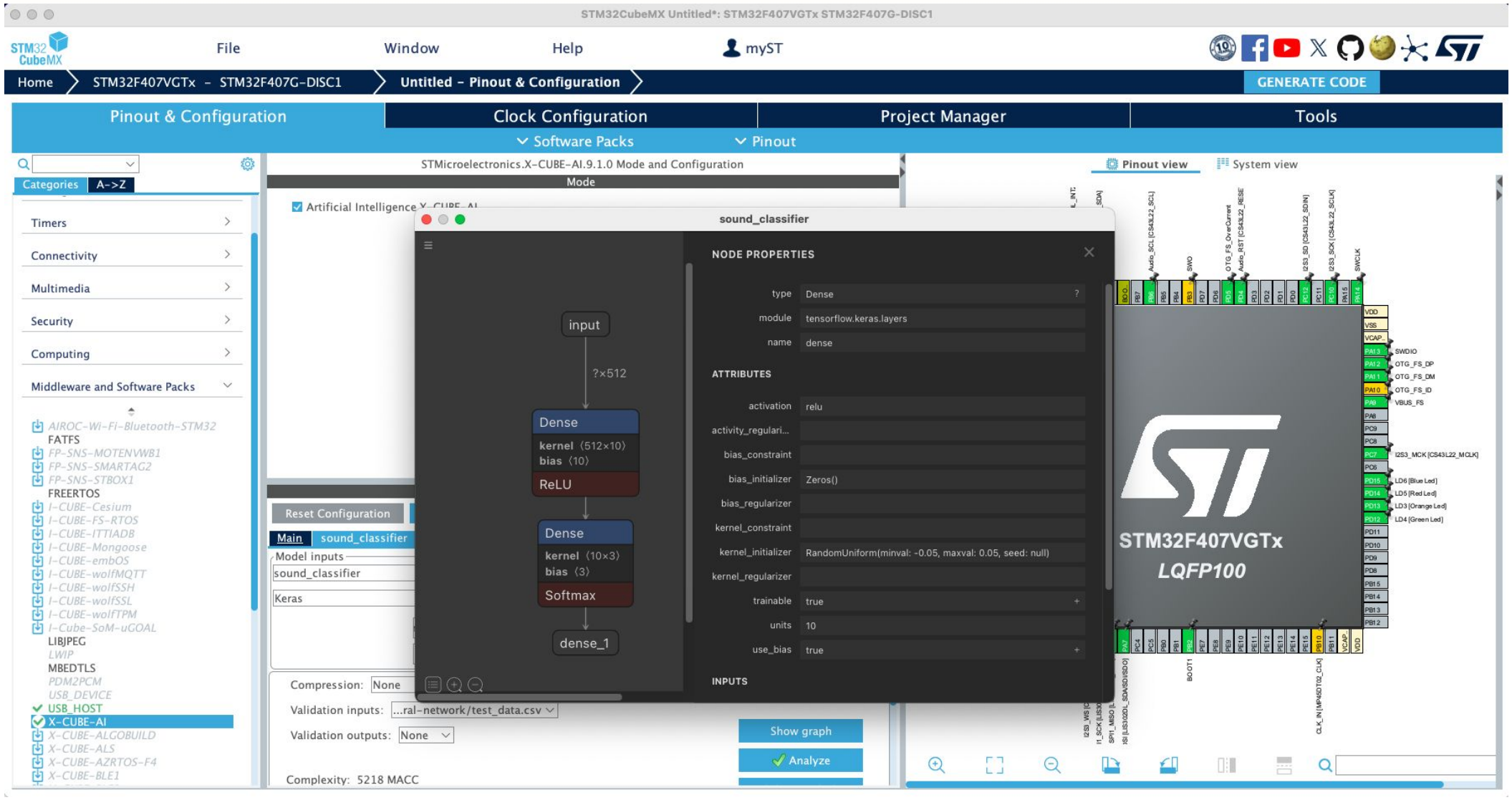
https://www.jonnor.com/tag/audio-classification/

Case Study 3 - Audio classifier

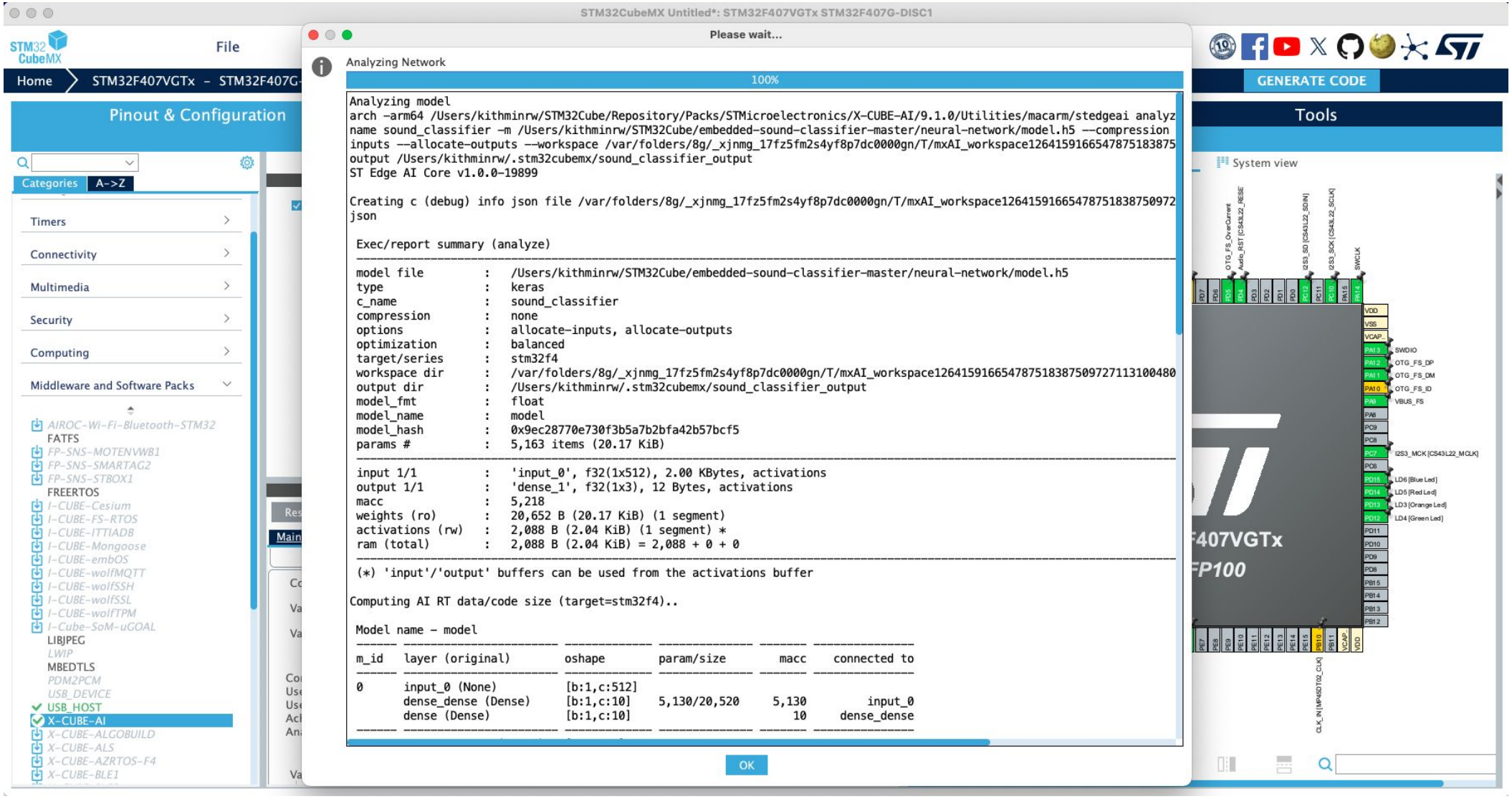


https://www.jonnor.com/tag/audio-classification/

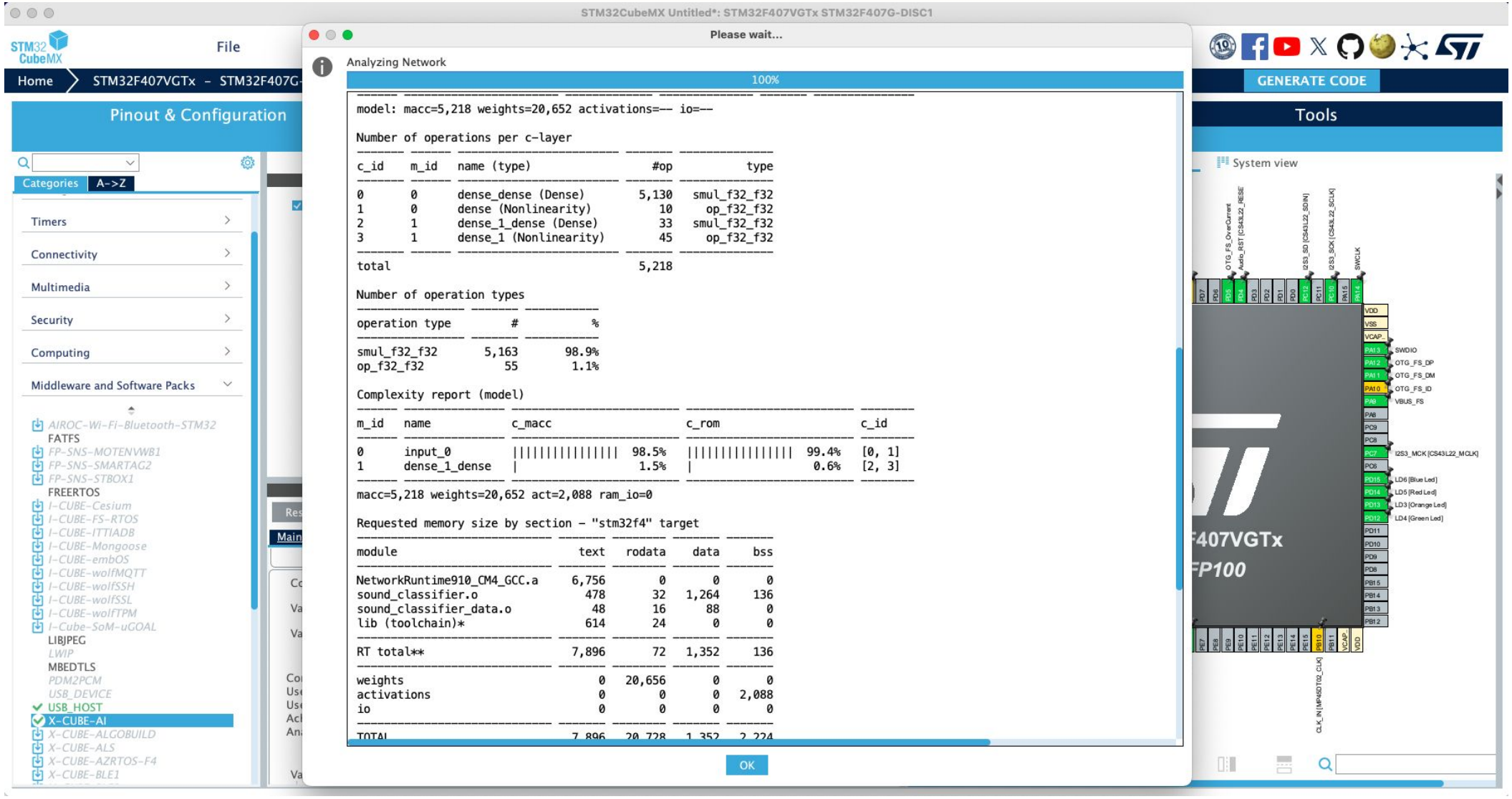
Case Study 3 - Audio classifier



Case Study 3 - Audio classifier

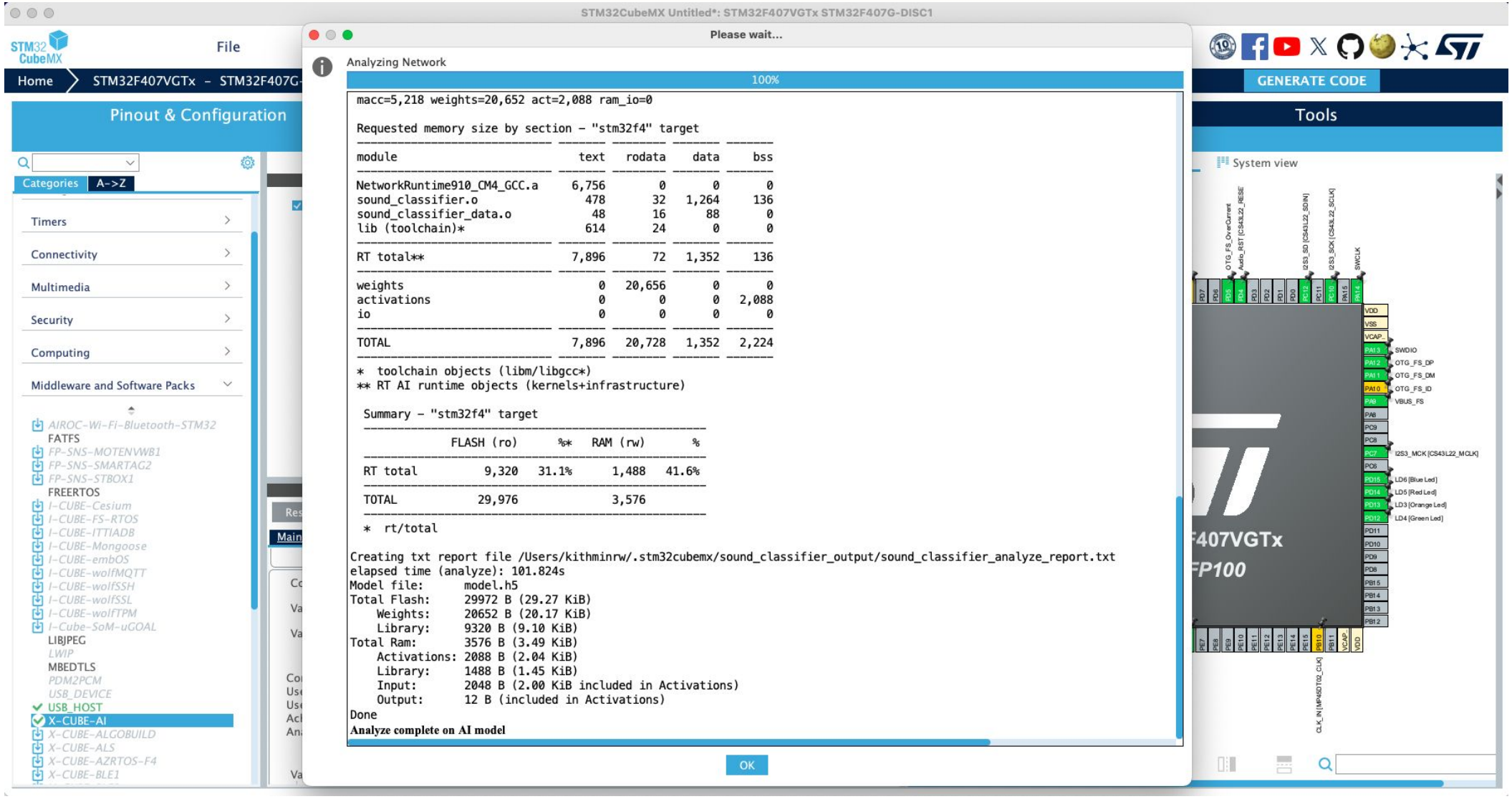


Case Study 3 - Audio classifier



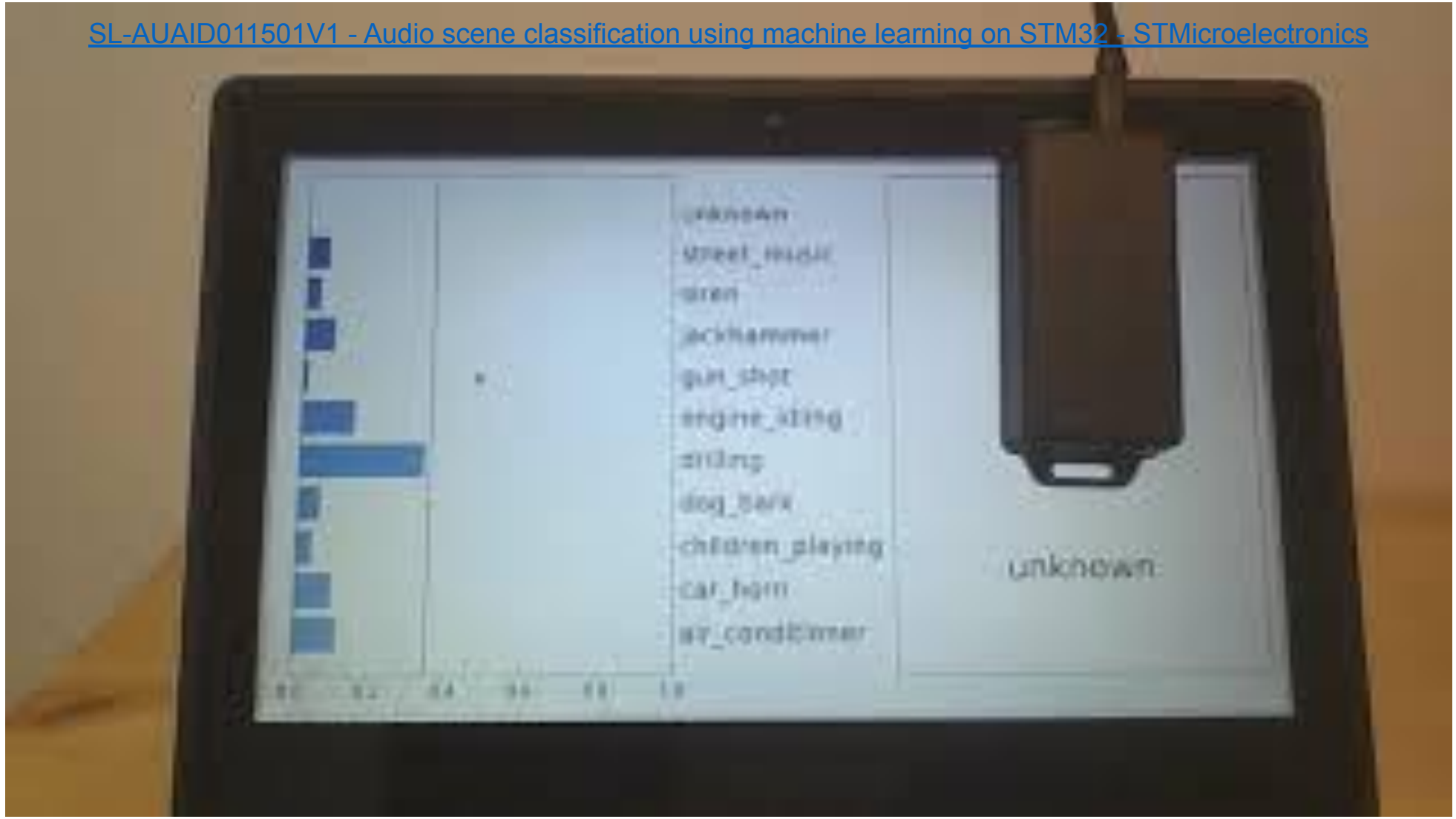

```
model: macc=5,218 weights=20,652 activations=-- io=--

Number of operations per c-layer
c_id    m_id   name (type)                  #op           type
0       0      dense_dense (Dense)        5,130   smul_f32_f32
1       0      dense (Nonlinearity)          10     op_f32_f32
2       1      dense_1_dense (Dense)         33   smul_f32_f32
3       1      dense_1 (Nonlinearity)        45     op_f32_f32
total                                     5,218

Number of operation types
operation type         #           %
smul_f32_f32       5,163       98.9%
op_f32_f32            55        1.1%

Complexity report (model)
m_id   name            c_macc                    c_rom                     c_id
0      input_0         ||||||||||||||||  98.5%   ||||||||||||||||  99.4%   [0, 1]
1      dense_1_dense   |                 1.5%    |                 0.6%    [2, 3]
macc=5,218 weights=20,652 act=2,088 ram_io=0

Requested memory size by section - "stm32f4" target
module                          text   rodata    data     bss
NetworkRuntime910_CM4_GCC.a    6,756        0       0       0
sound_classifier.o               478       32   1,264     136
sound_classifier_data.o           48       16      88       0
lib (toolchain)*                 614       24       0       0
RT total**                     7,896       72   1,352     136
weights                            0   20,656       0       0
activations                        0        0       0   2,088
io                                 0        0       0       0
TOTAL                          7,896   20,728   1,352   2,224
```

Case Study 3 - Audio classifier



Case Study 3 - Audio classifier

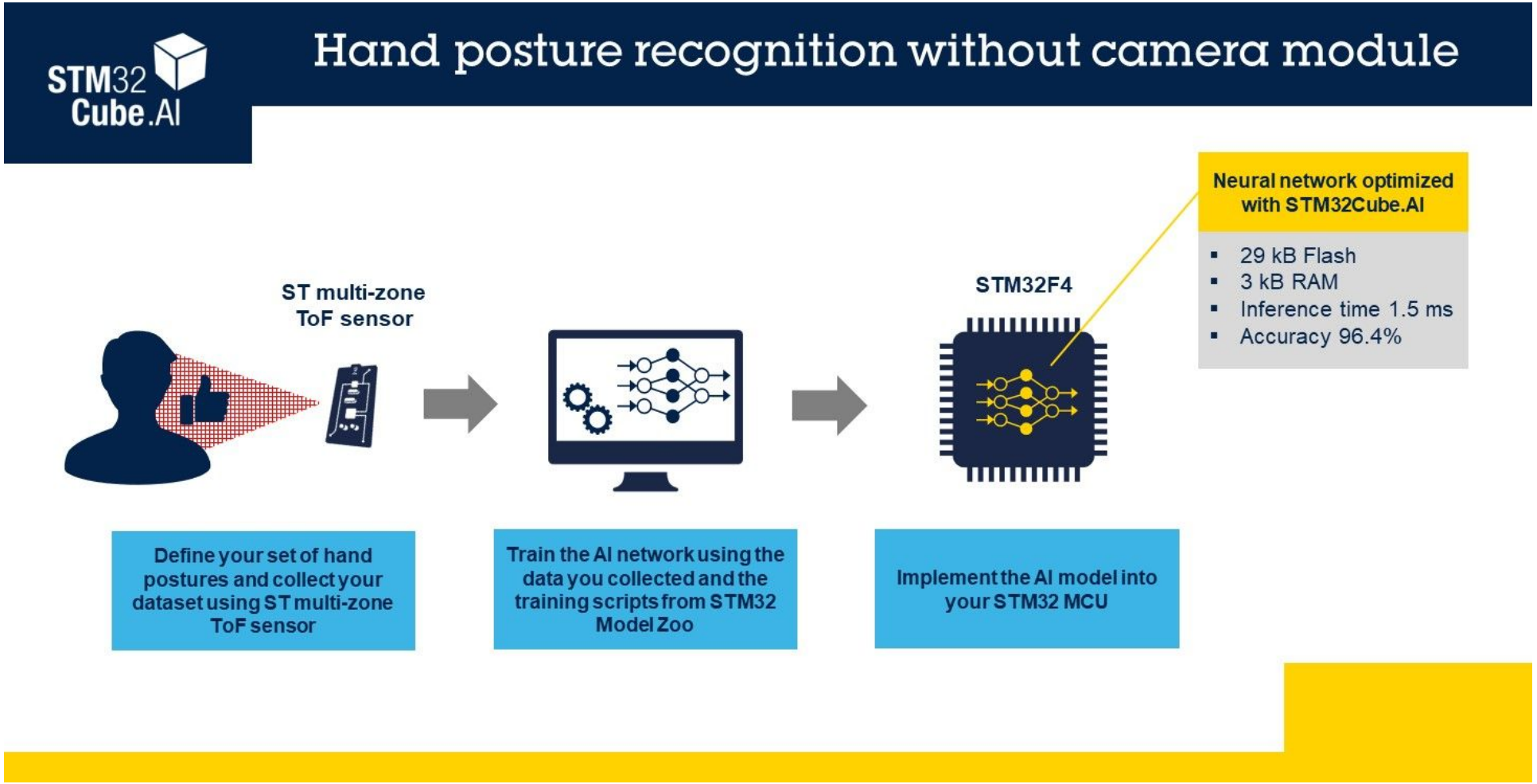
SL-AUAID011501V1 - Audio scene classification using machine learning on STM32 - STMicroelectronics



Case Study 4 - Person detection

More Case Studies

Edge AI case studies - STMicroelectronics



MOOC - STM32Cube.AI workshop - YouTube and Workshop AI

Assignment 4

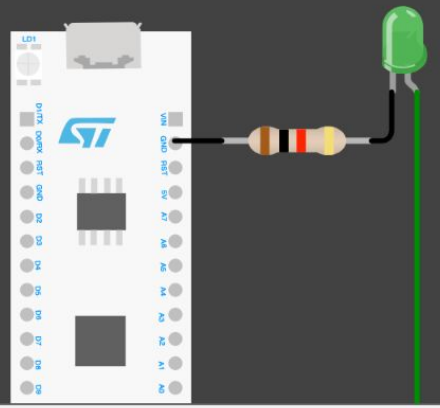
WOKWI SAVE SHARE Docs

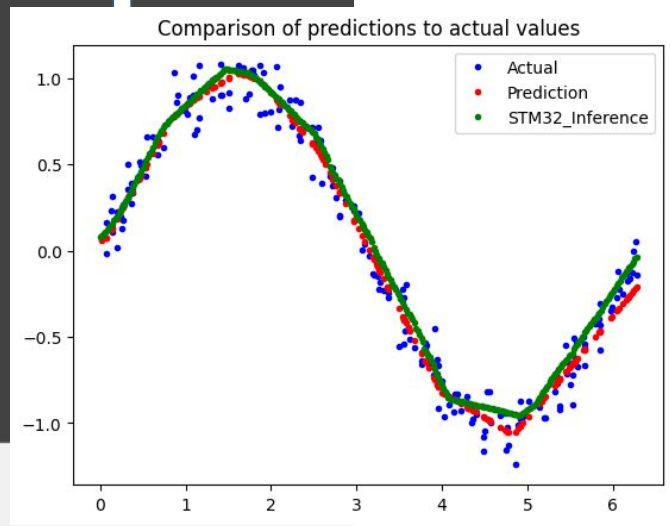
main.c diagram.json Library Manager

```c
// STM32 Nucleo-L031K6 HAL Blink + printf() example
// Simulation: https://wokwi.com/projects/367244067477216257

#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include <stm32l0xx_hal.h>

// ST Nucleo Green user LED (PB3)
#define LED_PORT                GPIOB
#define LED_PIN                 GPIO_PIN_3
#define LED_PORT_CLK_ENABLE     __HAL_RCC_GPIOB_CLK_ENABLE
#define VCP_TX_Pin GPIO_PIN_2
#define VCP_RX_Pin GPIO_PIN_15

UART_HandleTypeDef huart2;

void SystemClock_Config(void);
static void MX_USART2_UART_Init(void);

void osSystickHandler(void)
{
  // 1 Hz blinking:
  if ((HAL_GetTick() % 500) == 0)
  {
    HAL_GPIO_TogglePin(LED_PORT, LED_PIN);
  }
}

void initGPIO()
{
  GPIO_InitTypeDef GPIO_Config;
```

Simulation

00:14.496 99%






```
Hello, World!
Hello Students! Tracing X = 11
Hello, World!
Hello Students! Tracing X = 12
Hello, World!
Hello Students! Tracing X = 13
```

Thank You!