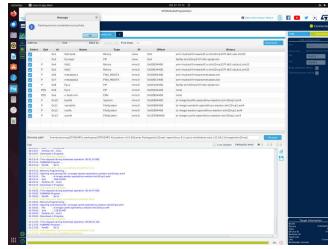
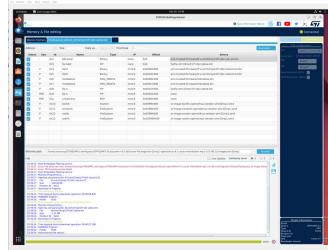
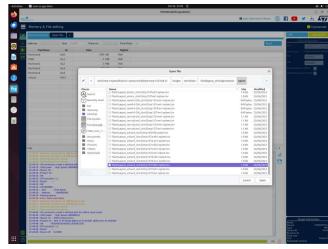
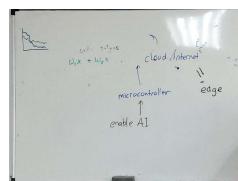


Logbook

Thursday, 26 October, 2023 11:40 PM



We'll connect five of the nine pins on the board to your Arduino. The five pins you need are labeled **GND**, **3.3v**, **OUTPUT**.

LD₊ and **LO₊**:

Board Label/Pin	Function	Arduino Connection
GND	Ground	GND
3.3v	Power Supply 3.3v	-
OUTPUT	Output Signal	A0
LD ₊	Leads-off Detect	I1
LO ₊	Leads-off Detect +10	-
SDN	Shutdown	Not used

Use sudo ./

STM32CubeIDE 1.22.14

Integrated Development Environment for STM32

Get Started Documentation Tools & Software

Product overview

Description API Reference Code Examples Feature Vehicle Recommended By You

Description

STM32CubeIDE is an all-in-one multi-OS development tool, which is part of the STM32Cube software ecosystem. It integrates the STM32CubeMX configuration tool, the STM32CubeIDE editor, and the STM32CubeProgrammer tool.

STM32CubeIDE includes built-in documentation to provide users with ready reference about project setup and memory requirements.

STM32CubeIDE also includes standard and advanced debugging features including analysis of CPU core registers, memory, and graphical memory, as well as the complete trace. Since this version contains a bug fix, update now.

All features

Integration of services from STM32Cube, STM32 Microcontroller, development platform and example project sources.

STM32CubeIDE supports the STM32CubeMX configuration tool, the STM32CubeIDE editor, and the STM32CubeProgrammer tool.

STM32CubeIDE also includes standard and advanced debugging features including analysis of CPU core registers, memory, and graphical memory, as well as the complete trace. Since this version contains a bug fix, update now.

STM32CubeIDE - Downloads

Preparing to unpack .../7-stm32cube-udev-rules-1.0.3-2-lts-v42.deb ...

Selecting previously unselected package 7-stm32cube-udev-rules.

Preparing to unpack .../8-stm32cubeide-1.12.2-18220-2623994-1601_amd64.deb ...

Inspecting st-m32cubeide-1.12.2-18220-2623994-1601 ...

Setting up 7-stm32cube-udev-rules (1.0.3-2) ...

Setting up st-m32cubeide (1.12.2-18220-2623994-1601) ...

Setting up st-m32cubeide-libs (1.12.2-18220-2623994-1601) ...

Setting up st-m32cubeide-common (1.12.2-18220-2623994-1601) ...

Setting up libpython2.7-stdlib (2.7.16-1ubuntu1) ...

Setting up libpython2.7-minimal (2.7.16-1ubuntu1) ...

Setting up libpython2.7 (2.7.16-1ubuntu1) ...

Processing triggers for libstdc++6 (9.3.0-17ubuntu1) ...

Processing triggers for libgcc1 (9.3.0-17ubuntu1) ...

Processing triggers for desktop file-utils (0.26-1ubuntu3) ...

Processing triggers for man-db (2.8.5-1ubuntu1) ...

w: download is performed unattended as root as file '/home/xiancong/Downloads/n' is owned by user 'xiancong'. pppacquire:Run (13: Permission denied)

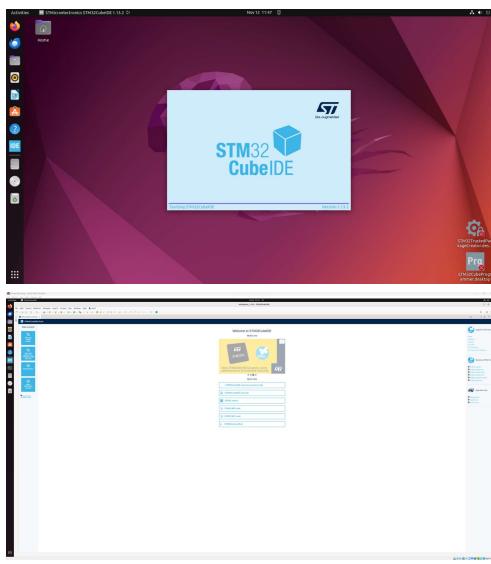
pppacquire uninstalled successfully

2023-10-26 11:40:40

13: Permission denied

x@ppacquire:~\$ are you root?

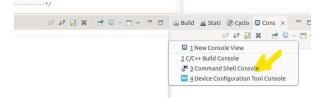
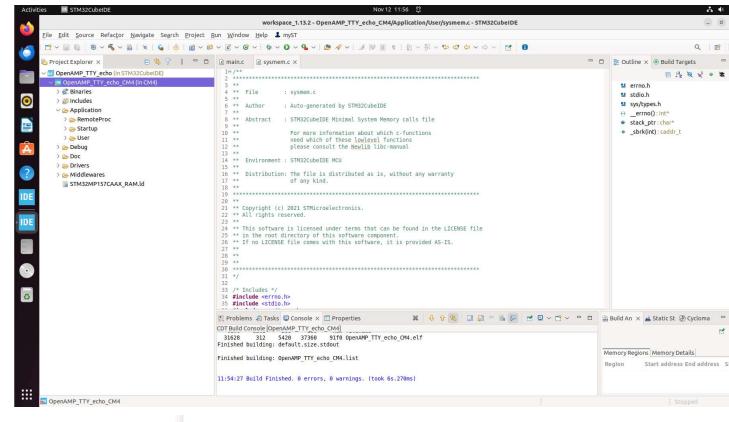
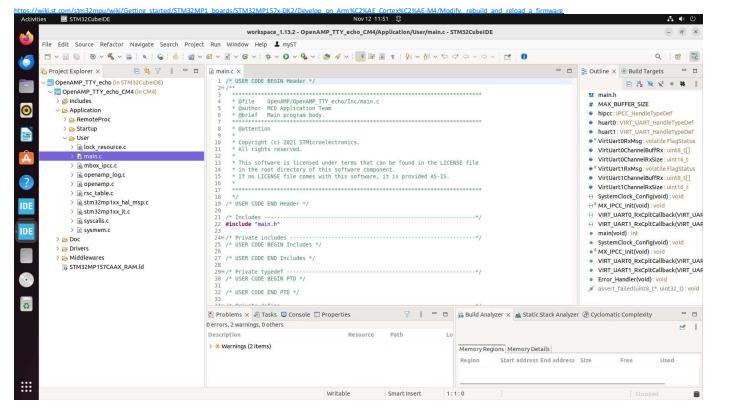
FYI Page 1



3 STM32CubeMP1 package structure

The STM32CubeMP1 Package is located in your \$HOME/STM32MPU_workspace/STM32MP1-Ecosystem-v5.0.0/Developer-Package directory. It is named STM32Cube_FW_MPU_V1.6.0.

```
|   BSP           BSP drivers for the supported STM32MP1 boards
|   [...]
|   CMSIS         [...]
|   HAL           HAL drivers for the supported STM32MP1 devices
|   [...]
|   STM32MP1xx_HAL_Driver
|   [...]
|   [...]
|   License.md
|   Middleware
|   [...]
|   package.xml
|   Projects
|     STM32MP157C-EVK.html  List of examples and applications for STM32CubeMP1 package
|     STM32MP157C-EVK      Set of examples and applications - STM32MP15 Discovery kits
|     [...]
|     STM32MP157C-EV1      Set of examples and applications - STM32MP15 Evaluation boards
|   [...]
```




```

10 // Copyright (c) 2021 STMicroelectronics.
11 // All rights reserved.
12 //
13 // This software is licensed under the terms of the MIT license.
14 // For the latest version see the file LICENSE in the root of this software component.
15 // File: main.c
16 // If no connection is made, the code will run in user mode.
17 // Otherwise, it will run in kernel mode.
18 // User code can be written in either mode.
19 // User code can be written in C or assembly.
20 // User code can be written in C or assembly.
21 // User code can be written in C or assembly.
22 // User code can be written in C or assembly.
23 // User code can be written in C or assembly.
24 // User code can be written in C or assembly.
25 // User code can be written in C or assembly.
26 // User code can be written in C or assembly.
27 // User code can be written in C or assembly.
28 // User code can be written in C or assembly.
29 // User code can be written in C or assembly.
30 // User code can be written in C or assembly.
31 // User code can be written in C or assembly.
32 // User code can be written in C or assembly.
33 // User code can be written in C or assembly.
34 // User code can be written in C or assembly.
35 // User code can be written in C or assembly.
36 // User code can be written in C or assembly.
37 // User code can be written in C or assembly.
38 // User code can be written in C or assembly.
39 // User code can be written in C or assembly.
40 // User code can be written in C or assembly.

```

6 Check connection to the target*

STM32CubeIDE requires to be connected to Linux running on STM32MP1 device through serial connection.
This connection is automatically detected and configured when you plug the cable on ST-Link port and board has booted.

```

24 #include <stdio.h>
25 #include <stdlib.h>
26
27 // MPU Serial (CONNECTED)
28 // -> /dev/ttyACM0
29 // Serial port: /dev/ttyACM0
30 // Baud rate: 115200
31 // Data size: 8
32 // Parity: None
33 // Stop bits: 1

```

7 Connect to the target

STM32CubeIDE requires to be connected to Linux running on STM32MP1 device through serial connection.

This connection is automatically detected and configured when you plug the cable on ST-Link port and board has booted.

```

24 #include <stdio.h>
25 #include <stdlib.h>
26
27 // MPU Serial (CONNECTED)
28 // -> /dev/ttyACM0
29 // Serial port: /dev/ttyACM0
30 // Baud rate: 115200
31 // Data size: 8
32 // Parity: None
33 // Stop bits: 1

```

8 Check connection to the target*

STM32CubeIDE requires to be connected to Linux running on STM32MP1 device through serial connection.
This connection is automatically detected and configured when you plug the cable on ST-Link port and board has booted.

```

24 #include <stdio.h>
25 #include <stdlib.h>
26
27 // MPU Serial (CONNECTED)
28 // -> /dev/ttyACM0
29 // Serial port: /dev/ttyACM0
30 // Baud rate: 115200
31 // Data size: 8
32 // Parity: None
33 // Stop bits: 1

```

9 Connect to the target

STM32CubeIDE requires to be connected to Linux running on STM32MP1 device through serial connection.

This connection is automatically detected and configured when you plug the cable on ST-Link port and board has booted.

```

24 #include <stdio.h>
25 #include <stdlib.h>
26
27 // MPU Serial (CONNECTED)
28 // -> /dev/ttyACM0
29 // Serial port: /dev/ttyACM0
30 // Baud rate: 115200
31 // Data size: 8
32 // Parity: None
33 // Stop bits: 1

```

10 Connect to the target

STM32CubeIDE requires to be connected to Linux running on STM32MP1 device through serial connection.

This connection is automatically detected and configured when you plug the cable on ST-Link port and board has booted.

```

24 #include <stdio.h>
25 #include <stdlib.h>
26
27 // MPU Serial (CONNECTED)
28 // -> /dev/ttyACM0
29 // Serial port: /dev/ttyACM0
30 // Baud rate: 115200
31 // Data size: 8
32 // Parity: None
33 // Stop bits: 1

```

11 Connect to the target

STM32CubeIDE requires to be connected to Linux running on STM32MP1 device through serial connection.

This connection is automatically detected and configured when you plug the cable on ST-Link port and board has booted.

```

24 #include <stdio.h>
25 #include <stdlib.h>
26
27 // MPU Serial (CONNECTED)
28 // -> /dev/ttyACM0
29 // Serial port: /dev/ttyACM0
30 // Baud rate: 115200
31 // Data size: 8
32 // Parity: None
33 // Stop bits: 1

```

12 Connect to the target

STM32CubeIDE requires to be connected to Linux running on STM32MP1 device through serial connection.

This connection is automatically detected and configured when you plug the cable on ST-Link port and board has booted.

```

24 #include <stdio.h>
25 #include <stdlib.h>
26
27 // MPU Serial (CONNECTED)
28 // -> /dev/ttyACM0
29 // Serial port: /dev/ttyACM0
30 // Baud rate: 115200
31 // Data size: 8
32 // Parity: None
33 // Stop bits: 1

```

13 Connect to the target

STM32CubeIDE requires to be connected to Linux running on STM32MP1 device through serial connection.

This connection is automatically detected and configured when you plug the cable on ST-Link port and board has booted.

```

24 #include <stdio.h>
25 #include <stdlib.h>
26
27 // MPU Serial (CONNECTED)
28 // -> /dev/ttyACM0
29 // Serial port: /dev/ttyACM0
30 // Baud rate: 115200
31 // Data size: 8
32 // Parity: None
33 // Stop bits: 1

```

14 Connect to the target

STM32CubeIDE requires to be connected to Linux running on STM32MP1 device through serial connection.

This connection is automatically detected and configured when you plug the cable on ST-Link port and board has booted.

```

24 #include <stdio.h>
25 #include <stdlib.h>
26
27 // MPU Serial (CONNECTED)
28 // -> /dev/ttyACM0
29 // Serial port: /dev/ttyACM0
30 // Baud rate: 115200
31 // Data size: 8
32 // Parity: None
33 // Stop bits: 1

```

15 Connect to the target

STM32CubeIDE requires to be connected to Linux running on STM32MP1 device through serial connection.

This connection is automatically detected and configured when you plug the cable on ST-Link port and board has booted.

```

24 #include <stdio.h>
25 #include <stdlib.h>
26
27 // MPU Serial (CONNECTED)
28 // -> /dev/ttyACM0
29 // Serial port: /dev/ttyACM0
30 // Baud rate: 115200
31 // Data size: 8
32 // Parity: None
33 // Stop bits: 1

```

16 Connect to the target

STM32CubeIDE requires to be connected to Linux running on STM32MP1 device through serial connection.

This connection is automatically detected and configured when you plug the cable on ST-Link port and board has booted.

```

24 #include <stdio.h>
25 #include <stdlib.h>
26
27 // MPU Serial (CONNECTED)
28 // -> /dev/ttyACM0
29 // Serial port: /dev/ttyACM0
30 // Baud rate: 115200
31 // Data size: 8
32 // Parity: None
33 // Stop bits: 1

```

17 Connect to the target

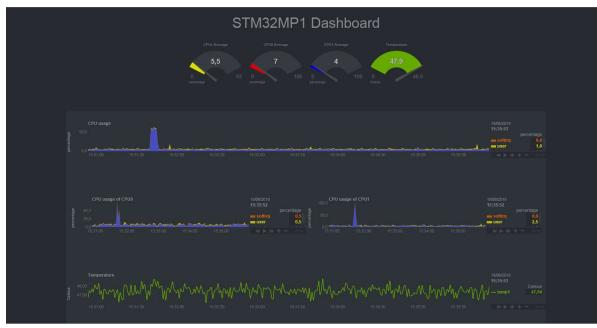
STM32CubeIDE requires to be connected to Linux running on STM32MP1 device through serial connection.

This connection is automatically detected and configured when you plug the cable on ST-Link port and board has booted.

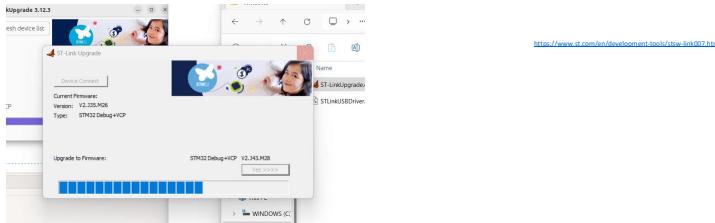
```

24 #include <stdio.h>
25 #include <stdlib.h>
26
27 // MPU Serial (CONNECTED)
28 // -> /dev/ttyACM0
29 // Serial port: /dev/ttyACM0
30 // Baud rate: 115200
31 // Data size: 8
32 // Parity: None
33 // Stop bits: 1

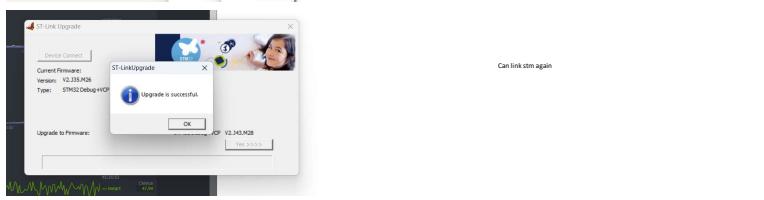
```

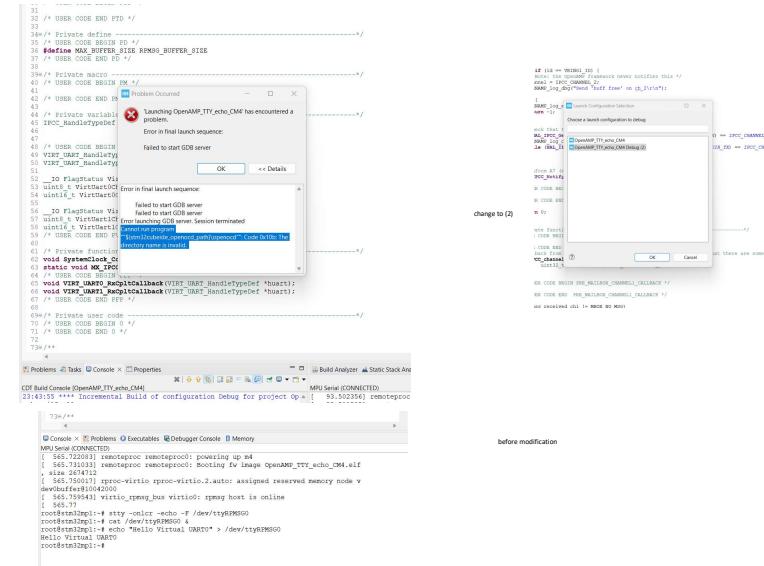
<https://wiki.oscon.org/Stm32mp1/Hwdata>



<https://www.st.com/en/development-tools/st-link007.html>

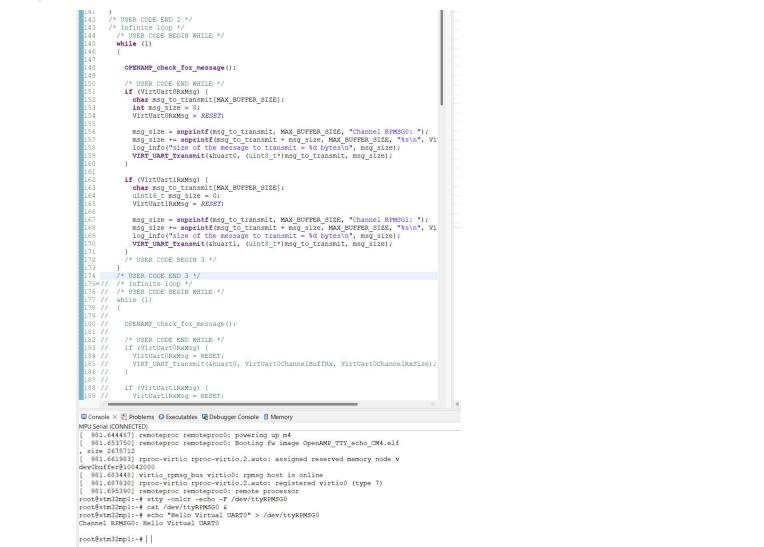


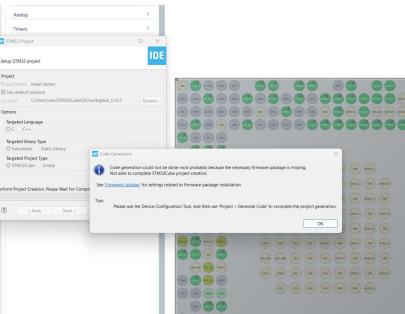
[Can link stm again](#)



[change to \[2\]](#)

[OK](#)





ST

ARDUINO® interface

Figure 16 shows the pinout of the ARDUINO® connectors.

Figure 16. ARDUINO® connectors pinout.

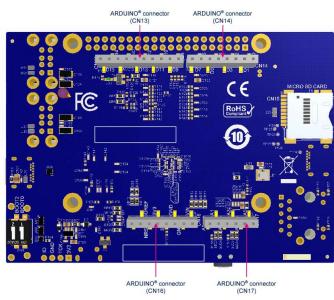


Table 23 describes the I/O configuration of the ARDUINO® interface.

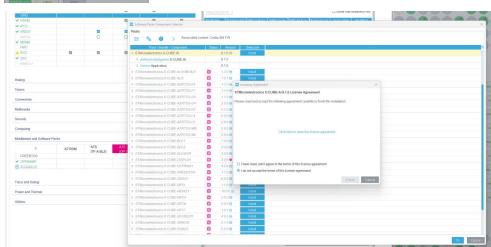
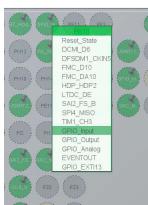
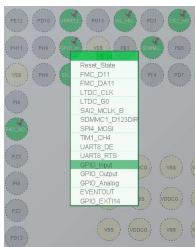
<https://www.st.com/resource/en/user-manual/um2637-discovery-kits-with-increase-if-frequency-800-phu-02157-arduino-connectors.pdf>
<https://www.st.com/resource/en/hardware-description/MB1273-CN2.html>
<https://www.st.com/resource/en/hardware-description/MB1273-CN2.html>

<https://learn.adafruit.com/tutorial/348723-hardware-setup-and-configuration>

Board Label	Pin Function	Arduino Connection
QMC	3.3v	3.3v Power Supply, 3.3v
SPI4	Output Signal	A0
LO-	Lead-off Detect -	11
LO+	Lead-off Detect +	10
SDN	Shutdown	Not used

```
void setup() {
    // Initialize the serial communication:
    Serial.begin(9600);
    pinMode(10, INPUT); // Setup for lead-off detection LO-
    pinMode(11, INPUT); // Setup for lead-off detection LO+
}

void loop() {
    if(digitalRead(10) == 1 || (digitalRead(11) == 1)){
        Serial.println("1");
    }
    else{
        // send the value of analog input 0:
        Serial.println(analogRead(0));
        //Wait for a bit to keep serial data from saturating
        delay(1);
    }
}
```



Olivier GALLIEN
ST Employee

2020-10-27 06:27 AM

Hi @Community member ,
Where did you read that Boot 0 = 0, Boot2 = 1 stand for "Reserved" and Boot0 = 0, Boot2 = 0 for "Engi" ?
Reading here : https://wiki.st.com/stm32mpu/wik/STM32MP157x-DKv1_hardware_descriptions#Boot_related_switches
Boot0 = 0, Boot2 = 0 => Forced USB boot for flashing
Boot0 = 0, Boot2 = 1 => Engineering boot
Consistent with behavior you noticed.
Oliver

5 Boot related switches*

The following table defines the different combination of the boot mode selection switches:

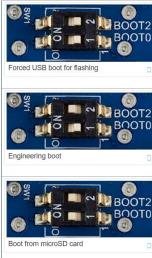
Information

Boot1 ("1" is grounded (always) to the Discovery kits)

Boot mode	Boot 0	Boot 1	Boot 2
Forced USB boot for flashing	0	0	0
Not supported	1	0	0
Engineering boot	0	0	1
microSD card	1	0	1

Boot modes

Below the corresponding configurations of the boot related switches ([SW1](#) on the board overview picture):



6 REPLIES

PavanCP
2020-10-27 09:30:46
Hi Olivier,
Except if I have very tight real-time constraints, you should probably being able to do what you want by using STLINK-USB-KEY instead of jumper pads.

Concerning the ADC, I'm not sure there is no other line connection (e.g. resistor, high input mobility etc...).

Some ADC and PTY1011 pins with 1mA current limit are possible.

One can need to choose 2.5mA or 15mA depending on how fast resolution is needed.

<https://community.st.com/thread/15084>

Some ADC pins are available in this header package. Two in EN2 boot pinces also refer to [here](#).

• ADC1 (PA0, PA1, PA2, PA3) on an AN801 (Analog connector A)
• ADC2 (PA0, PA1, PA2, PA3) on an AN801 (Analog connector A)
• ADC3 (PA0, PA1, PA2, PA3) on an AN801 (Analog connector A)
• ADC4 (PA0, PA1, PA2, PA3) on an AN801 (Analog connector A)
• ADC5 (PA0, PA1, PA2, PA3) on an AN801 (Analog connector A)
• ADC6 (PA0, PA1, PA2, PA3) on an AN801 (Analog connector A)

For USART :

See <https://community.st.com/thread/15084> for more details.
• USART 1 (PA9, PA10, PA11, PA12) on an AN801 (Analog connector C)
• USART 2 (PA9, PA10, PA11, PA12) on an AN801 (Analog connector C)
For CAN, you can use the CAN pins provided with Western Waypad:
<https://www.st.com/waypad/can-pins-with-western-waypad>

Some ADC pins are available in the header package. Two in EN2 boot pinces also refer to [here](#).

Regards.

In order to give some visibility to the previous topic, please click on "Mark as feedback" on the reply which solved your issue or answered your question.

[Mark](#) [Reply](#)

Typical values

Unless otherwise specified, typical data are based on $T_J = 25^{\circ}\text{C}$, $V_{DD} = 3.3\text{ V}$, $V_{DDCORE} = 1.2\text{ V}$. They are given only as design guidelines and are not tested.

Typical ADC accuracy values are determined by characterization of a batch of samples from a standard diffusion lot over the full temperature range, where 95% of the devices have an error less than or equal to the value indicated (mean±2σ).

67 static void HAL_DMA_Init(void);
2 References, Search

```
110 // I2C Initialization */  
111 MX_I2C_Init();  
112 // SPI Initialization */  
113 MX_SPI_Init();  
114 // USB CODE REGIS SysInit */  
115 /* USB CODE END SysInit */  
116 // Initialize all configured peripherals */  
117 MX_USART_1_Init();  
118 MX_USART_2_Init();  
119 // USB CODE REGIS 2 */  
120 MX_USB_Device_Init();  
121 // USB CODE REGIS 3 */  
122 MX_ADC_Init();  
123 // USB CODE REGIS 4 */  
124 HAL_ADC_Start_DMA(&ahc62, (uint32_t *)bfr, 1024);  
125 // HAL_DeplayDMA */  
126 // USB CODE END 2 */  
127 // USB CODE END 3 */  
128 // USB CODE END 4 */  
129 // Infinite loop */  
130 // Main loop BEGIN WHILE */  
131 while (1)  
132 {  
133     int lo_plus = HAL_GPIO_ReadPin(GPIOE, GPIO_PIN_11);  
134     int lo_minus = HAL_GPIO_ReadPin(GPIOE, GPIO_PIN_10);  
135     //HAL_ADC_Start();  
136     //HAL_ADC_Cont();  
137     //HAL_ADC_Stop();  
138     //HAL_ADC_Cont();  
139     for (int i = 0; i<1024; i++)  
140     {  
141         bfr_Value[i] = (float)Out[i]*3/365;  
142     }  
143 }
```

3.1 Boot time assignment*

3.1.1 On STM32MP15x lines

The ADC is usually not used at boot time but it may be used by the SSSLB (see [Boot chain overview](#)), to check for power supplies for example.

Click on the right to expand the legend..

Domain	Peripheral	Instance	CoreA7 secure (ROM code)	CoreA7 non-secure (ROM code)	CoreA7 ROM boot	CoreA7 (S-Boot)	Comment
Analog	ADC	ADC					

<https://community.st.com/thread/15084>

 Kevin HUBER
ST Employee

2022-02-10 07:17 AM

Hi @mische1,

Thanks to have shared your .loc.

To be able to get a useable value by reading the ADC, you must have a reference which is VREFBUF.

I advice you to read the reference manual (RM0436 revision S) page 1993:
http://www.st.com/resource/reference_manual/DM00032759.pdf

and the ADC page on the wiki: https://wiki.at.com/stm32ref/wik/ADC_Internal_Ref/adc#Features

Then you have two choices to enable VREFBUF.

- Engineering mode (without Linux)

If you are not using Linux you have to enable the VREFBUF after the ADC_Init() but before starting it:

```
_HAL_RCC_VREF_CLK_ENABLE(); // Enable the VREF clock
HAL_SysTickConfig(SYSCFG_VREFBUF_HDWR_DISABLE); // Disable the high impedance mode. In the default one read page 1994 of refman
HAL_Sysev_VREFBUF_SetVoltageScaling((SYSCFG_VREFBUF_VOLTAGE_SCALE1)); // To set the voltage to 2.5V
HAL_Sysev_EnableVREFBUF(); // To enable VREFBUF
```

Then you must calibrate your adc before using it:

```
#if HAL_ADC_Calibration_Start(&hadc1, ADC_CALIB_OFFSET_LINEARITY, ADC_SINGLEENDED) != HAL_OK
{
    /* Calibration Error */
    Error_Handler();
}
```

And finally you can use the code that you wrote in your first post to read the value.

- In Production mode (with linux running):



Virtual Desktop (VDI) and High Performance Computing (HPC) Services

Dear All Students,

We hope you're settled into the new semester. IT Services would like to take this opportunity to highlight the following two virtual computing services which are available for use.

Virtual Desktop (VDI)

The VDI service allows students to access teaching and learning software directly from our servers and is available at Teaching Computer Rooms (TCR) 1, 2, 3 & 4 in Block F2 and remotely.

For all Engineering and Environmental & Geographical Science Students, you will have access to two types of virtual desktop:

- TCR
- TCR-GPU (for software requiring GPU)

All other students will only see one virtual desktop:

- TCR

To access the virtual desktop from any TCR, please refer to: [VDI internal access](#)

To access the virtual desktop from off campus using your own device, please refer to: [VDI external access](#)

NOTE: All files not saved to external or cloud storage will be lost after you logout.

NOTE: For TCR 4, the interface is slightly different, please refer to [TCR 4 note](#). The virtual desktops remain the same.

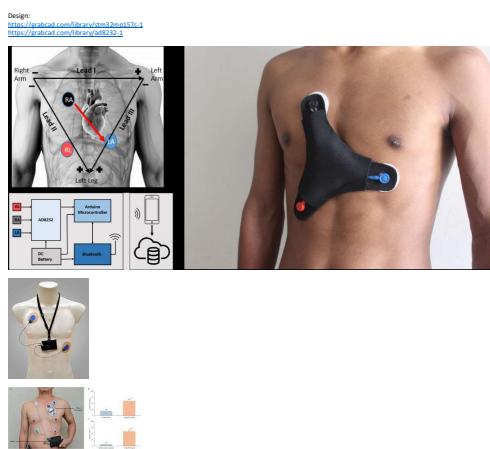
To see the software installed on each virtual desktop image, please refer to the [software list here](#).

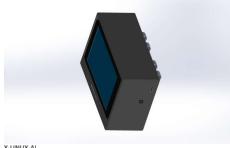
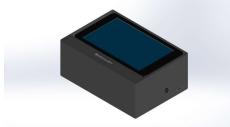
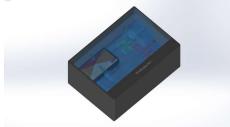
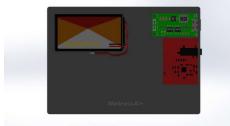
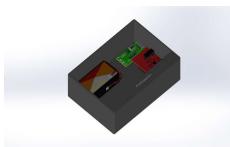
High Performance Computing (HPC)

The University of Nottingham Mainframe offers world-class high-performance computing facilities specifically designed for artificial intelligence (AI) and other research, teaching and learning via the NVIDIA DGX A100 Supercomputer.

For more information on how to access the HPC service [please click here](#).

If you have any questions please email ITServiceDesk@nottingham.ac.uk





X-LINUX-AI

http://wiki.ct.com/itm32mpu/wls/X-Linux-AI_OpenTuner_Expansion_Package

Model Zoo: <https://github.com/CTMicroelectronics/itm32ai-models>

<https://towardsdatascience.com/my-journey-in-converting-pytorch-to-tensorflow-lbs-d244170ebed>

<https://code.engineering/qweqzvzjpm3514xu>

Convert script to tflite

https://www.tensorflow.org/tutorials/quantization/freeze_tflite

https://www.tensorflow.org/tutorials/quantization/tensorflowjs_compatible_tensorflow_minimal

create a virtual environment, install then switch to python 3.6.5

```
$ conda create -n tensorflow python=3.7
$ conda activate tensorflow
$ conda config --add channels d3t
$ pip install tensorflow
```

activate the environment when you want to use tensorflow

https://www.tensorflow.org/api_docs/python/tf/export_saved_model_from_checkpoint

<https://codelabs.developers.google.com/codelabs/tensorflow-training/index.html#0>

Model Zoo: <https://github.com/CTMicroelectronics/itm32ai-models>

TensorFlow.js: <https://github.com/CTMicroelectronics/itm32ai-tensorflowjs>

https://www.tensorflow.org/tutorials/quantization/tensorflowjs_compatible_tensorflow_minimal

TensorFlow.js: https://www.tensorflow.org/tutorials/quantization/tensorflowjs_compatible_tensorflow_minimal

II.1. Data from EmotivEYEE

The EmotivEYEE Professional was used: http://www.biosignalplus.com/EmotivEYEE_Professional

Please refer to [1] for details on sensor placement (see Section 3.2 there). All signals were sampled

at 700 Hz and raw data is contained in SX_respiration.txt. There are 10 columns here. First column:

sequential line number. Second column: ignore. Columns 3-10: raw data of the 8 sensor channels.

The order of the channels is defined in the table below. The entries 'XYZ' refer to the 3-channel

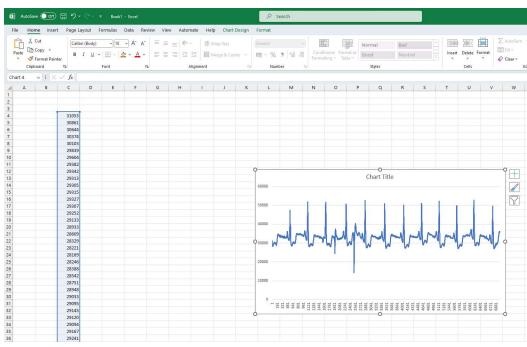
sensor channels. The 'ignore' channel is present in the raw data.

In order to convert the raw sensor values into SI units, each channel has to be transformed based on

the formulas given below (signal contains the raw sensor values, voff=3, chan_bit=2^16).

- ECG (mV): ($(\text{signal} \text{chan_bit} - 0.5) \times 1000$)
Details: http://www.biosignalplus.com/datasheets/ECG_Sensor_Datasheet.pdf
- EDA (µS): ($((\text{signal} \text{chan_bit} / 1000) - 12)$)
Details: http://www.biosignalplus.com/datasheets/EDA_Sensor_Datasheet.pdf

10 sec WESAD:



The time interval of each condition is defined as start and end time, see the lines 3 and 4 in SX_quest.csv. Time is given in the format (minutes:seconds). Time is counted from the start of the ResponDAV device's start of recording.

III.1. Study protocol

The order of the different conditions is defined on the second line in SX_quest.csv. Please refer to [1] for further details on each of the conditions (see Section 3.3 there). Please ignore the elements "break", "Read", and "head"; these are not relevant for this dataset.

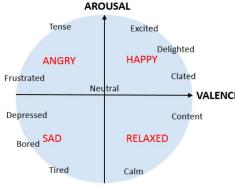
Comparing the self-reports after the assessment and baseline condition reveals that the manipulation had no significant effect on the subjects' slightly higher scores on valence and arousal (dimensional approach, DIA) and less anxiety (STA). However, the effect of the condition is rather small. In contrast, the impact of the stressor was pronounced, across all questionnaires. The analysis of the SSSQ scores indicates that the subjects felt more engaged and worried than distracted during the TST task (Engagement: 11.7 ± 2.3 , Distress: 6.0 ± 2.9 , Worry: 10.6 ± 2.3).

Table 2: Evaluation of the questionnaires.

	PANAS positive	PANAS negative	STA1 valence	DIA arousal
Baseline	25.5±0.0	12.3±2.0	10.8±1.9	6.7±0.9
Stress	31.3±4.7	22.0±6.4	18.5±2.0	4.5±1.6
Assessment	25.8±5.1	11.4±2.1	9.3±2.0	7.5±0.6
				3.0±1.6

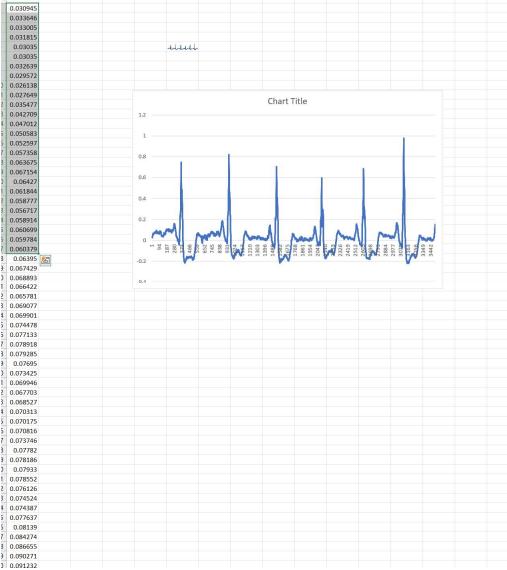
The high 'Engaged' mean response shows the subjects' high motivation to perform well in the given task. The high 'Worry' suggests that the subjects were determined to give a good impression on the panel. In our opinion, these scores demonstrate that most subjects believed our cover story of the TST.

After the stressor condition, the subjects showed mixed scores with respect to positive (PA) and negative affect (NA). The high PA score indicates that subjects felt energized and concentrated during the TST, which coincides with the high engagement values reported in the SSSQ. The elevated NA score indicates an increased level of subjective distress. The DIA scores support these observations, indicating an increase in arousal and a decrease in valence.



<https://drive.google.com/drive/u/0/folders/1LJyqHg14uI>
https://drive.google.com/drive/u/0/folders/1LJyqHg14uI/two_ParTone_Emotion_recording_using_Electroencephalogram_Analysis/preprocess/

Wocad (normalized, label 0,5,6,7 removed)



PROCESS PICKLE FILE:

```
import pickle
import pandas as pd
import numpy as np
# Loads the pickle file
with open("data_preprocessing\\label\\S17.pkl", "rb") as file:
    # Extract 'label' and 'ECG' arrays
    label = np.load(file)
    ekg_data = np.load(file)[label['sheet']]['ECG']

    # Create a DataFrame with 'label' and 'ECG' columns
    ekg_df = pd.DataFrame(ekg_data['ECG'].flatten(), columns=['ECG'])
    ekg_df['label'] = label['label'].flatten()

    # Filter out rows where the label is 1, 5, 6, or 7
    filtered_df = ekg_df[~(label['label'] == 1) & ~(label['label'] == 5) & ~(label['label'] == 6) & ~(label['label'] == 7)].copy() # Make a copy to avoid SettingWithCopyWarning

    # Create a '16-bit' column based on the modified formula using loc
    filtered_df['label'] = filtered_df['label'].apply(lambda x: ((x - 1) * 16) + 1)

    # Reshape 'label' and '16-bit' arrays to have the same number of rows
    num_rows = len(filtered_df) // 7000 * 7000
    label_data = np.zeros((num_rows, 1), dtype='int16')
    filtered_label = np.zeros((num_rows, 1), dtype='int16')

    # Create a DataFrame with transposed '16-bit' data
    filtered_ecg = np.transpose(filtered_df['ECG'].values.reshape(-1, 7000))
    filtered_ecg = filtered_ecg[0:num_rows, :].values.reshape(-1, 7000)

    # Create a DataFrame with transposed 'label' data
    label_ecg = np.transpose(label_data[0:num_rows, :].values.reshape(-1, 7000))

    # Create a DataFrame with transposed '16-bit' data
    final_ecg = np.transpose(filtered_ecg)
    final_label = np.transpose(label_ecg)

    # Save the final DataFrame to CSV
    final_df.to_csv('data_preprocessing\\output_Filter_Label\\output_S17.csv', index=False)
```


6.3.22 16-bit ADC characteristics

Unless otherwise specified, the parameters given in [Table 77](#) are derived from tests performed under the ambient temperature, f_{clock} frequency and V_{DDA} supply voltage conditions summarized in [Table 13: General operating conditions](#).

Table 77. ADC characteristics⁽¹⁾⁽²⁾

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DDA}	Analog power supply	-	1.62	-	3.6	V
$V_{\text{REF}+}$	Positive reference voltage	$V_{\text{DDA}} \geq 2 \text{ V}$	2	-	V_{DDA}	
$V_{\text{REF}-}$	Negative reference voltage	$V_{\text{DDA}} < 2 \text{ V}$	-	V_{SSA}		
f_{ADC}	ADC clock frequency	$2 \text{ V} \leq V_{\text{DDA}} \leq 3.3 \text{ V}$	$\text{BOOST} = 1$	0.12	-	36
			$\text{BOOST} = 0$	0.12	-	20
						MHz
t_s	Sampling time	-	1.5	-	810.5	
t_{CONV}	Total conversion time (including sampling time)	N-bit resolution	$t_s + N/2^{(4)}$			

servo.default_dens_inquiry 0

1x7000

SubConnected

samples (248/10000)

base (248)

1x248

FullConnected

samples (248/2448)

base (248)

1x2448

HalfConnected

samples (124/1248)

base (124)

1x1248

LowConnected

samples (62/624)

base (62)

1x624

LowConnected

samples (31/312)

base (31)

1x312

LowConnected

samples (16/16)

base (16)

1x16

LowConnected

samples (8/8)

base (8)

1x8

LowConnected

samples (4/4)

base (4)

1x4

LowConnected

samples (2/2)

base (2)

1x2

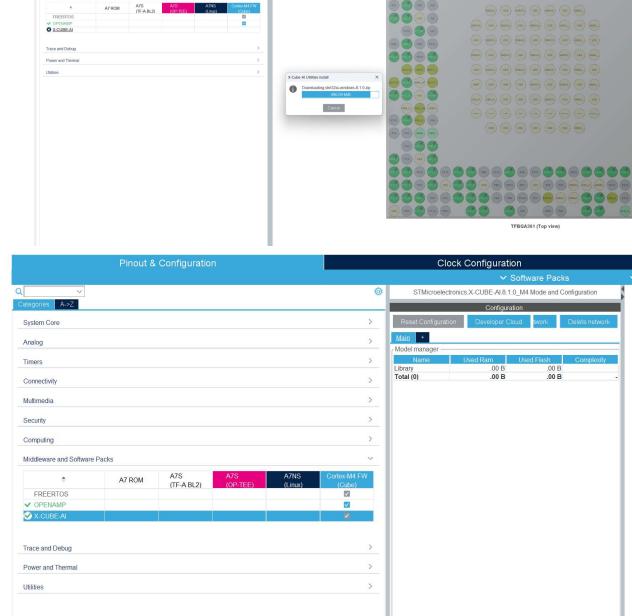
Logistic

samples (1/1)

base (1)

1x1

StateOfParticularCell0 (2)




```

Analyzing model
C:\Users\user\STM32Cube\Repository\Pack\STMicroelectronics\X-CUBE-AI\1.0\Utilities\windows\stm32c_analyze -name emotion_m
Nottingham\model\eq_emotion_model.tflite --type tflite --compression none --verbosity 1 --workspace C:\Users\user\STM32Cube\Temp\xmA1_workspace21094894246634000338483275935583190 --output C:\Users\user\stm32cube\emotion_m
serial stm32mp1 --allocate-outputs

Neural Network Tools for STM32 family v1.7.0 (stm32c v9.1.0-19520)

Exec/report summary (analyze)
model file : D:\Nottingham\FYP\Final-Year-Project-Nottingham\model\eq_emotion_model.tflite
type : tflite
t_name : emotion_model
compression : none
options : allocate_inputs, allocate_outputs
optimization : balanced
target_arch : cortex-a4
workspace dir : C:\Users\user\AppData\Local\Temp\xmA1_workspace21094894246634000338483275935583190
output dir : C:\Users\user\stm32cube\emotion_model_output
model_file : eq_emotion_model
model_name : eq_emotion_model
model_hash : 445b91a7e2344620f00ae00de50952
params # : 17,132,760 items (14.35 MB)
params : -----
inputs 1/1 : "serving_default_dense_input0" (domain:activations/*/*default*)
    7000 items, 27.4 Kib, al_float, float, (1,7000)
output 1/1 :
    4 items, 16 B, al_float, float, (4)
macs : 17,135,682
weights (Co) : 69,531,040 B (69.53 Mib) (1 segment)
activations (Ro) : 14,192 B (35.34 Kib) (1 segments)
ram (Total) : 36,192 B (35.34 Kib) > 36,192 = 0 + 0
-----
(*) *input*/output buffers can be used from the activations buffer
>>> Shell execution has FAILED (returned code = 1)
$ arm-none-eabi-gcc -T C:\Users\user\STM32Cube\Repository\Packs\STMicroelectronics\X-CUBE-AI\1.0\Utilities\windows\targets\mthumb\mcu=cortex-a4 -mfpu=fpv4-sp-d16 -march=cortex-a4 -mfloat-abi=hard -Wl,-Mapemotion_model.map,-crt0=*.crt-sections
m32c_start_group.c:8:1: error: 'stm32cube' was not declared in this scope
        _start();
```

```

asm.F2_F32 : 17,132,760 100.0%
sg.F2_F32 : 2,932 0.0%
Complexity report (model)
-----
```

#	name	c_mac	c_ram	c_id
0	serving_default_dense_input0	111111111111111111111111	03.7%	[0]
1	sequential_dense_1_MatMul	111111111111111111111111	3.1%	[1]
2	sequential_batch_normalization_bachnorm_mul_1	111111111111111111111111	0.0%	[2]
3	n1_3	111111111111111111111111	0.0%	[3]
4	sequential_batch_normalization_1_bachnorm_mul_1	111111111111111111111111	0.0%	[4]
5	n1_4	111111111111111111111111	0.0%	[5]
6	sequential_batch_normalization_2_bachnorm_mul_1	111111111111111111111111	0.0%	[6]
7	n1_7	111111111111111111111111	0.0%	[7]
8	sequential_batch_normalization_3_bachnorm_mul_1	111111111111111111111111	0.0%	[8]
9	n1_8	111111111111111111111111	0.0%	[9]
10	sequential_batch_normalization_4_bachnorm_mul_1	111111111111111111111111	0.0%	[10]
11	n1_11	111111111111111111111111	0.0%	[11]
12	sequential_dense_7_MatMul	111111111111111111111111	0.0%	[12]
13	n1_33	111111111111111111111111	0.0%	[13]

```

macs=17,135,682 weights=69,531,040 activations=36,192 ram=0
Creating txt report file C:\Users\user\stm32cube\emotion_model_output\emotion_model_analyze_report.txt
Elapsed time (analyze): 32.912s
Model file: eq_emotion_model.tflite
Total size: 20500 B (20.50 KB)
Weights: 6931040 B (69.30 MB)
Library: 39720 B (30.00 KB)
Total: 20898 B (20.80 KB)
Activations: 36192 B (35.34 KB)
Library: 0 B
Input: 20500 B (20.50 KB included in Activations)
Output: 16 B (Included in Activations)
Done
Analyze complete at At Model
Request RAM or Flash size for emotion_model is bigger than available Ram or Flash
```

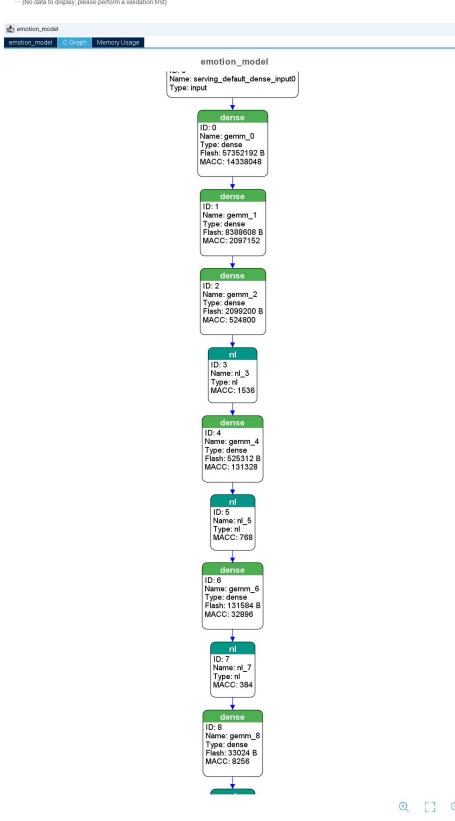
Compression: None Optimization: Balanced

Validation inputs: Random numbers Validation outputs: None

Analysis metrics:

- Show graph
- Analyze
- Validate endpoint
- Validate on target

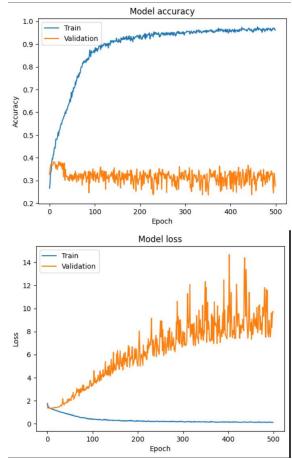
Validation metrics: (No data to display, please perform a validation first)





```

73/73 [====] -> 1s 10ms/step - loss: 0.1065 - accuracy: 0.9703 - val_loss: 0.1530 - val_accuracy: 0.2685
73/73 [====] -> 1s 10ms/step - loss: 0.1341 - accuracy: 0.9668 - val_loss: 0.0688 - val_accuracy: 0.2461
73/73 [====] -> 1s 10ms/step - loss: 0.1164 - accuracy: 0.9666 - val_loss: 0.3984 - val_accuracy: 0.3184
73/73 [====] -> 1s 10ms/step - loss: 0.1089 - accuracy: 0.9783 - val_loss: 0.1640 - val_accuracy: 0.3332
73/73 [====] -> 1s 10ms/step - loss: 0.1165 - accuracy: 0.9642 - val_loss: 0.7415 - val_accuracy: 0.2995
73/73 [====] -> 1s 10ms/step - loss: 0.1234 - accuracy: 0.9612 - val_loss: 0.7642 - val_accuracy: 0.2754
  
```



```

Starting AI validation on desktop with custom dataset: D:\Nottingham\PPV\Fiscal-Year\Project-Nottingham\data_preprocessing\output_filter_label\test_output_transposed.csv
C:\Users\user\STM32Cube\Projects\STM32cube\stm32cube_emotion\emotion_model\src\main.cu
Year-Project-Nottingham\data_preprocessing\output_filter_label\test_label\test_output_transposed.csv
Neural Network Model: emotion_model.tflite
Model Size (MB): 1.43 MB
Model Type: tflite
Model Version: 1
Setting training data...
Model type: tflite
Model file path: D:\Nottingham\PPV\Fiscal-Year\Project-Nottingham\data_preprocessing\output_filter_label\test_output_transposed.csv
- samples are reshaped (7000, 1) -> (1, 1, 7000)
[[1] (1, 1, 1, 7000), min/max(12180.000, 65521.000), mean/std(32894.710, 4433.729), serving_default, dense_input0]
Neural Network Model is successfully generated
Copying the AI runtime results to the user workspace: C:\Users\user\AppData\Local\Temp\mk\ai\workspace212471994139949196552889088349175\inspector_emotion_model\workspace
Execution report summary (validate)

```

```

Execution report summary (validate)
model file : D:\Nottingham\PPV\Fiscal-Year\Project-Nottingham\model\rec_emotion_model.tflite
type : tflite
c_name : emotion_model
components : none
options : - allocate_inputs, allocate_outputs
optimization : balanced
workspace_size : 1000000
author : Aggreata, LLC
output_dir : C:\Users\user\STM32Cube\stm32cube_emotion\emotion_model\output
vinpu files : D:\Nottingham\PPV\Fiscal-Year\Project-Nottingham\data_preprocessing\output_filter_label\test_output_transposed.csv
model_name : rec_emotion_model
model_hash : 637835524
parameters : 129.158 items (3.59 MB)

```

```

Input 1/2 : learning_default dense_input0 (domain:activations/**default**)
  - 7000 items, 27.34 kB, ai.Root, float, (1,7000)
Output 1/2 : id_12 (domain:activations)**default**
  - 4 items, 16 bytes, in activations buffer (1,4)
macc : 3.796.684
weights (w) : 15.036.243 (14.33 kB) -> 1 (gamma)
activations (w) : 3.796.684 (29.34 kB) -> 1 (sigma), 0
ram (total) : 30.048 (29.34 kB) -> 30.048 (0 kB)

```

```

(*) Input/Output buffers can be used from the activations buffer
Running the STM32 model (AI RUNTIME). -name=emotion_model, mode=x86
X:\User\user\STM32Cube\Projects\STM32cube\stm32cube_emotion\emotion_model\src\main.cu
C:\Users\user\STM32Cube\Projects\STM32cube\stm32cube_emotion\emotion_model\src\main.cu
Summary: emotion_model -- workspace_model

```

```

inputs/outputs : 1/2
input_1 : input_1, (1,1,7000), float32, 24,000 bytes, in activations buffer
output_1 : output_1, (1,1,1,4), float32, 16 bytes, in activations buffer
o, o_idx, o_end : 1, 1, 1, 4
compute_datetime : Feb 4 2024 15:59:10
activations : 1
weights : 15.036.243 (14.33 kB) -> 1 (gamma)
weights (w) : 3.796.684 (29.34 kB) -> 1 (sigma), 0
macc : 3.796.684

```

```

runtime : STM32AU (8.1.0) Tools & v1.0 -
capabilities : IO_ONLY, PER_LAYER, PER_LAYER_WITH_DATA
device : STM32CubeMX_CU810_V1.0.2
NOTE: duration and exec time per layer is just an indication. They are dependent of the HOST-machine work-load.
STM32 Profiling results v1.2 - emotion_model

```

```

nb_samples() : 1
duration : 2.03 ms by sample (2.013/2.013/2.000)
macc : 3.796.684

```

```

HOST duration : 0.0051 (total)

```

```

Inference time per node

```

c_id	m_id	type	dur (ms)	%	name
0	0	Dense	1.893	94.0%	ai_node_0
1	3	Dense	0.004	0.4%	ai_node_1
2	1	NL	0.001	0.0%	ai_node_2
3	3	Dense	0.020	1.0%	ai_node_3
4	4	NL	0.001	0.0%	ai_node_4
5	5	Dense	0.004	0.2%	ai_node_5
6	6	NL	0.027	0.0%	ai_node_6
7	7	Dense	0.004	0.0%	ai_node_7
8	8	NL	0.027	0.0%	ai_node_8
9	9	Dense	0.004	0.0%	ai_node_9
10	10	NL	0.001	0.0%	ai_node_10
11	11	Dense	0.004	0.0%	ai_node_11
12	12	NL	0.012	0.0%	ai_node_12

```

total : 2.013

```

```

Statistic per tensor

```

```

tensor shape/type   min   max   mean   std name
10  (1, 1, 2000)float32 23880.000 6521.000 33684.710 6433.729 input_1
0.0  (1,1,4)float32 0.000  0.995  0.468  0.470 output_1_

```

```

Running the TRTE model...

```

```

Saving validation data...
output directory: C:\Users\user\STM32Cube\stm32cube_emotion\emotion_model\output\emotion_model\emotion_model_val.h5p
create_zarr : 1
c_outputs : 1 (1, 1, 4)float32, min/max=[0.000, 0.995], mean/std=[0.467, 0.470], rl_12
c_outputs : 1 (1, 1, 4)float32, min/max=[0.000, 0.995], mean/std=[0.468, 0.470], rl_12
Compute time: 0.000121404 ms
Cross accuracy report #1 (reference vs c-model)

```

```

notes: - ACC metric is not computed ("classifier" option can be used to force it)
- the output reference model is used as ground truth/reference
- 1 samples (4 floating point samples)
accuracy : 0.9999999999999999, max=-0.001248886, D=-0.003298092, nse=100.00%, cos=100.00%
Evaluation report (summary)

```

```

Output    rmse    mae    l2r    mean    std    nse    cos    tensor
X-cross#1  n.a.  0.0021879  0.0012499  0.0032981  -0.0009206  0.0019848  0.9999973  0.99999961  rl_12, ai_float, (1,4), m_id<12

```

```

aic : Classification accuracy (all classes)
rmse : Root Mean Squared Error
mae : Mean Absolute Error

```


12r : L2 relative error
mse : Nash-Sutcliffe efficiency criteria
cos : cosine similarity

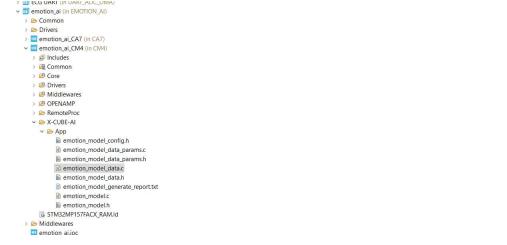
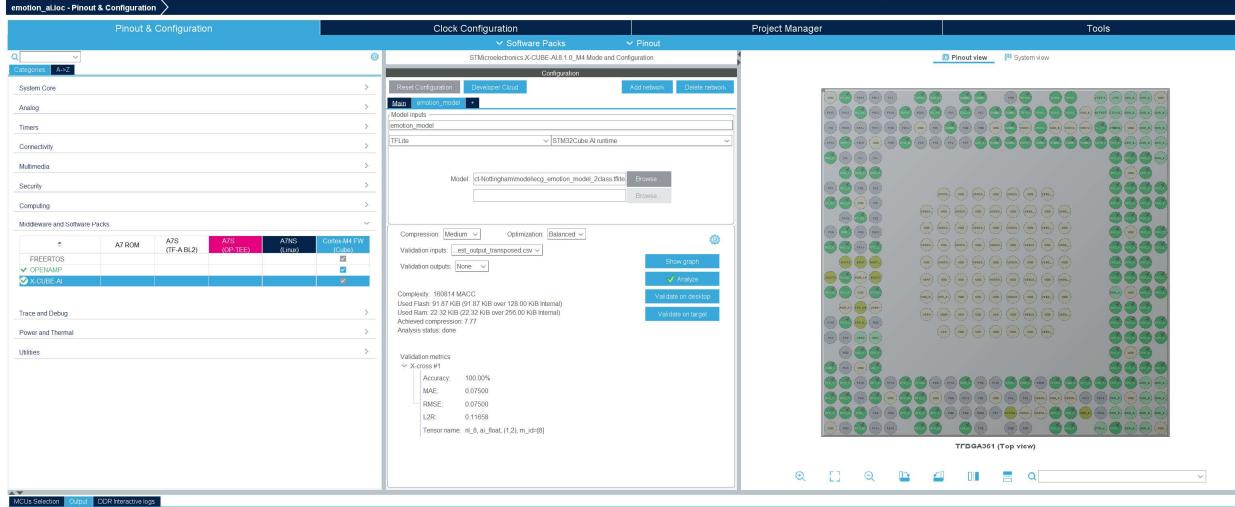
Creating validation file C:\Users\user\stm32cubem\emotion_model_output\emotion_model_validate_report.txt

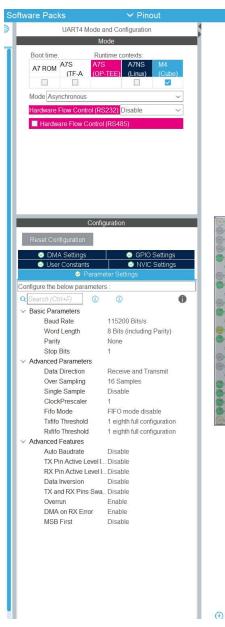
elapsed time (validated): 4.512s

Validation ended



Validation metrics:
(No data to display, please perform a validation first)





https://wiki.st.com/stm32mcu/wiki/AI:How_to_perform_motion_sensing_on_STM32L4_IoTnode
I'm using X-CUBE_AI 7.1.0 but having trouble compiling this example project for motion detection.

RBenf.11
Associate
2022-02-11 10:48 PM

```
ai_imu0[0].n_batches = 1;
'at_buffer' (aka 'struct ai_buffer_') has no member named 'n_batches'

network.h

AI_DEPRECATED
#define AI_NETWORK_IN
#include "network.h"
AI_NETWORK_IN(HANDLE NULL NULL)

This comes up invalid initializer error.

Any ideas? The initial code compiled ok with the acc data apart from a bug in cubemx where the lsm6dsh.h could not be found.

Labels: STM32L4 Series
```

0 Kudos

KK-10
Associate II
2023-02-28 08:35 AM

X-CUBE-AI 7.0 code example doesn't work with later versions. In my case 7.3.0.
Completely disregard example code for 7.0 version and use code from ST example: "How to perform motion sensing on STM32L4 IoT node".
Section 6 "Create an STM32Cube AI application using X-CUBE-AI".


```

Prediction i_1 = -0.0008
Running inference
Prediction i_1 = -0.0008
Prediction i_0 = -0.0008
Running inference
Prediction i_1 = -0.0008

```



```

LIBROSAR: https://community.st.com/N/6e013/m-pur-products/ecg-and-shimmer-things-on-ecg-mat-builds/a/8989N
  - [text] leekk is not implemented and will always fail
C:/ST/STM32CubeIDE_1.13.2/STM32CubeIDE/plugins/com.st.stm32cube.ide.mcu.externaltools.gnu-tools-for
C:/ST/STM32CubeIDE_1.13.2/STM32CubeIDE/plugins/com.st.stm32cube.ide.mcu.externaltools.gnu-tools-for
[!] [text] leekk is not implemented and will always fail
finished building target: emotion_ai_CM4.elf

arm-none-eabi-size emotion_ai_CM4.elf
arm-none-eabi-size -s -A emotion_ai_CM4.elf > "emotion_ai_CM4.list"
[14488 1228 9328 96944 17ab0] emotion_ai_CM4.elf
[+] [text] file size: 17ab0
[+] [text] file size: 17ab0
Finished building: emotion_ai_CM4.list
[!] [text] 0 errors, 0 warnings
[!] [text] 0 errors, 0 warnings. (took 3s.95ms)
  246 16

```

ARM Dual Cortex-A9 up to 800 MHz

ARM Dual Cortex-A9 up to 800 MHz

External Memories	Dual SDRAM	Dual Quad-SPI	8-Bit SLC NAND & Bi-ECO
Internal Memories	MD2 System RAM	MD2 Persistence RAM	
System RAM 256kB	Black Cap RAM 4kB	OTF Flash 64kB	
Connectivity	3x SPI (Serial, RS 2.0)	System	
Up to 100 Mbps/1 Gbps	MIPI-CSI camera	SD LOGG	
Up to 2.4 Mbit/s	LCD-ITY connector	Internal and External	
Camera Interface	MIPI-CSI sensor	WIFI: 2.4 GHz	
USB 2.0	MIPI-CSI connector	BT: 2.1 GHz	
Security	MIPI-CSI	BT: 5 GHz	
MIPI-CSI	MIPI-CSI	3x watchdog	
MIPI-CSI	MIPI-CSI	1x RTC	
MIPI-CSI	MIPI-CSI	1x I2C	
(2-channel PWR/Reset)	3x Tanger Pin with	2x 16-bit advanced	
Up to 4.15V	Secure Reset	motor control timers	
Bt 4.1		2x 10-bit timers	
Bt 5.0 (BLE)		2x 16-bit timers	
SPI		Aeing	
Analogue trng RNS		Analogue	
8-bit 16-bit digital ID		2x 16-bit ADCs	
		2x 12-bit DACs	

Valid for STM32MP157C and STM32MP157CZ

Access request for 'DREAMER: A Database for Emotion Recognition through EEG and ECG Signals from Wireless Low-cost Off-the-Shelf Devices' was accepted.

Zenodo <reply@zenodo.org>
to me
Your access request for 'DREAMER: A Database for Emotion Recognition through EEG and ECG Signals from Wireless Low-cost Off-the-Shelf Devices' was accepted.

[Access the record](#)

[Reply](#) [Forward](#)

Published April 13, 2017 | Version v1

DREAMER: A Database for Emotion Recognition through EEG and ECG Signals from Wireless Low-cost Off-the-Shelf Devices

Katsigianis, Stavros¹; Ramzan, Naeem¹

Data collectors: Cuntz, Thomas¹; Park, Sebastian¹
Researchers: Katsigianis, Stavros¹; Ramzan, Naeem¹

We present DREAMER, a multi-modal database consisting of electroencephalogram (EEG) and electrocardiogram (ECG) signals recorded during affect elicitation by means of audio-visual stimuli. Signals from 23 participants were recorded along with the participants' self-assessment of their affective state. The database consists of two parts: one part contains the EEG signals captured using portable, wearable, wireless, low-cost and off-the-shelf equipment that has the potential to allow the use of affective computing methods in everyday applications. The Emotiv EPOC wireless EEG headset was used for EEG and the Shimmer2 ECG sensor for ECG.

Classification results for valence, arousal and dominance of the proposed database are comparable to the ones achieved for other datasets that use non-portable, expensive, medical grade devices.

The proposed database is made publicly available in order to allow researchers to achieve a more thorough evaluation of the suitability of these capturing devices for affect recognition applications.

Please cite as:

K. Katsigianis, N. Ramzan, "DREAMER: A Database for Emotion Recognition Through EEG and ECG Signals from Wireless Low-cost Off-the-Shelf Devices," IEEE Journal of Biomedical and Health Informatics, vol. 22, no. 1, pp. 98-107, Jan. 2018. doi: 10.1109/JBHI.2017.2668239

Files

[DREAMER.pdf](#) (453.4 kB)

DREAMER v1.0.2

DREAMER: A Database for Emotion Recognition through EEG and ECG Signals from Wireless Low-cost Off-the-Shelf Devices

Abstract

We present DREAMER, a multi-modal database consisting of electroencephalogram (EEG) and electrocardiogram (ECG) signals recorded during affect elicitation by means of audio-visual stimuli. Signals from 23 participants were recorded along with the participants' self-assessment of their affective state. The database consists of two parts: one part contains the EEG signals captured using portable, wearable, wireless, low-cost and off-the-shelf equipment that has the potential to allow the use of affective computing methods in everyday applications. The Emotiv EPOC wireless EEG headset was used for EEG and the Shimmer2 ECG sensor for ECG.

Files (453.4 kB)

Restricted

The record is publicly accessible, but files are restricted to users with access.

Name	Size	Action
DREAMER.mat	ref:30566949/0e40a52149f0f15ed4	402.8 MB
DREAMER.pdf	ref:11a2609f0390215550c80d4040	549.7 kB

Dataset

32K 6K

Show more details

Versions

Version v1 Apr 13, 2017

10.1109/JBHI.2017.2668239

External resources

Indexed in

Keywords and subjects

Affect, Emotion, EEG, ECG, physiological signals, wireless devices, affect recognition

Details

DOI [DOI: 10.1109/JBHI.2017.2668239](#)

Resource type Dataset

Publisher Zenodo

Citation

Katsigianis, S.; Ramzan, N. (2017). DREAMER: A Database for Emotion Recognition through EEG and ECG Signals from Wireless Low-cost Off-the-Shelf Devices [Data set]. Zenodo. [https://doi.org/10.1109/JBHI.2017.2668239](#)

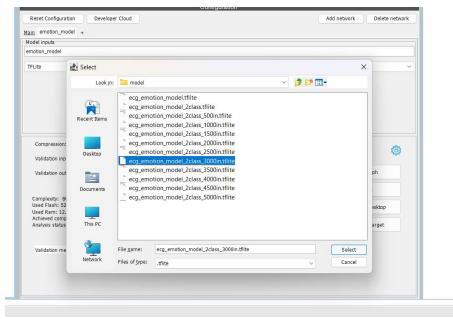
Style APA

Export

JSON

Technical metadata

Created April 13, 2017
Modified October 2017



To fix this you can use the following:

- Increasing the size of your training set
- Regularization techniques
- Early stopping
- Reduce the complexity of your model
- Use different hyperparameters like learning rate

Some things I would personally try:

1. Try decreasing the learning rate.
2. Try training the model without the augmented data and see how the validation loss behaves.
3. Try splitting the augmented data so that it's also contained in the validation set and see how the model behaves.

(counting down from 2023 by steps of 17) tasks. The stress condition lasted for about 10 minutes. The amusement condition was for 6.5 minutes long, where the participant was shown any video clips. In total, the raw ECG data from the chest worn device, sampled at 700 Hz. To be consistent with the labels used by the authors, we consider baseline and amusement conditions as the no-stress class.

2) SWELL-KW: SWELL knowledge work dataset [3] contains 10 subjects (1 female) who did several office tasks (writing reports and making a presentation) under three conditions - neutral, email interruptions, time pressure. During the email interruption session, 8 emails were sent - most were irrelevant and some required a reply. In the time pressure session, the participant had to complete 10 tasks in 2/3rd of the time allotted for the neutral session. The neutral and email interruption sessions lasted for around 45 minutes each, whereas the time pressure session lasted for around 30 minutes. We use the ECG signal sampled at 500 Hz, which were recorded on a TMSA Mobi device. The participants did not report feeling stressed in any of the conditions. However, they indicated higher temporal demand (they felt time pressure due to the pace of the task) during the time pressure session. In a subsequent study [4], the authors labelled the data from email interruption sessions as stress and the time pressure session as no-stress for a binary stress classification task. Hence, we also consider the data belonging to email interruptions and time pressure sessions as stress samples.

B. Classification Models

We describe our stress detection models and their training parameters below:

1) *Deep ECNNE*: This is a CNN-LSTM stress detection model proposed in [8]. The idea of CNN-LSTM is to use the CNN layers as feature extractors and train the LSTM layers to learn temporal patterns in the extracted features. The model has an initial convolution block containing a 1D convolutional layer, a pooling layer, a dropout layer, and a batch normalization layer. The activation function used for the convolution layer is rectified linear unit (ReLU). The 1D convolution layer has 50 filters with a kernel size corresponding to 0.6 s. The pooling layer has a size equivalent to 0.8 s of data. For a 256 Hz input, kernel size is 154, and pooling size is 20. The next block consists of two 1D convolutional layers and a final prediction layer. The LSTM layer has 32 neurons and the second one has 16 units. We add a dropout layer and a batch normalization layer between the two LSTM layers. The activation function for the LSTM layers is Tanh, and for the prediction layer is Softmax. We use a dropout rate of 0.2 for both the dropout layers.

2) *ECG Emotion Recognition Model*: This CNN model is proposed in [6] for emotion recognition on various datasets, including WESAD and SWELL-KW. The model has three convolutional layers, each consisting of a 1D convolution, 1D convolution layers and a pooling layer. The convolutional layers belonging to a block have identical parameters such as kernel size and the number of filters. From block 1 to block 3, the

number of filters are 32, 64, and 128, whereas the kernel sizes are 32, 16, and 8. The pooling layers are of size 8 with strides of 2. The convolution block consists of two convolution layers with 128 filters each. We add a dropout layer (dropout rate = 0.0) after each fully connected layer. Finally, the model is connected to a stress prediction layer with Softmax activation. All the convolution layers and the fully connected layers have ReLU activation.

3) *Multi-Layer Perceptron*: This is a simple neural network with an input layer, two hidden layers, and a prediction layer. The hidden layers have 12 and 6 nodes. The activation function for hidden layers is ReLU and for the prediction layer is Sigmoid. To prevent over-fitting, we add a dropout layer (dropout rate = 0.2) after the first input layer.

4) *RTC*: This is a decision tree model that trains a certain number of decision trees on various subsets of the training set and uses their output to make a final prediction. This reduces over-fitting and performs well even if the individual classifiers are weak. Similar to [1], the number of decision trees (or estimators) is set to 100 and the minimum number of samples for splitting a node is set to 20.

5) *SVM*: This is a commonly used supervised learning model. Similar to [4], we use SVM with Radial basis function (RBF) kernel.

We use Tensorflow to train neural networks and Scikit learn to train other machine learning models. For all the models, we use a weighted loss to tackle class imbalance in the training dataset. For the neural network models, we use the Adadelta optimizer (learning rate = 1.0) and cross-entropy loss. We train them for 200 epochs with a batch size of 128.

C. Evaluation metrics

We use *Precision*, *Recall*, and *Average* metrics for evaluation. Accuracy is the ratio of number of correctly predicted samples to total number of samples in the test set. F1-score is computed as the harmonic mean of precision and recall. Precision is the number of correctly predicted samples of a class out of all the samples predicted belonging to the class. Recall is the number of correctly predicted samples of a class out of all the samples belonging to the class. To tackle the class imbalance in the datasets, we compute macro f1-score, i.e., compute f1-score for each class and average them. We perform within-dataset LOSO evaluation as described in section IV-C. We repeat a similar cross-dataset evaluation process. However, this is not an indicator of generalizability of the models on data collected using a different sensor or a different stressor. Hence, we evaluate the models using cross-dataset validation. This validation involves training a model on a dataset A and evaluating it using samples from another dataset B.

D. Pre-processing

The datasets used in the two datasets have different sampling rates. We use a constant for HRV features, but the ECG-based deep learning models require the input length to be the same. To keep the data consistent for all models, we down-sample the ECG signals in both datasets to 256 Hz.

You what parameters should I use for prescaler for TIM16 in STM32MP157F-DK2

ChatGPT

In the STM32MP157F-DK2, which is based on the STM32MP1 series microcontroller, TIM16 is a general-purpose timer peripheral. To configure the prescaler for TIM16, you typically need to consider the desired timer clock frequency and the input clock frequency.

Here's a general procedure to calculate the prescaler value:

1. Determine the desired timer clock frequency (TIM16).
2. Determine the input clock frequency (usually the system clock or a peripheral clock).
3. Calculate the prescaler value based on the desired timer clock frequency and the input clock frequency.

Here's a formula to calculate the timer clock frequency:

```
mathematica  Copy code
Timer Clock Frequency = (Input Clock Frequency) / (Prescaler + 1)
```

For example, if you want TIM16 to run at 1MHz, and your input clock frequency is 16MHz, you can calculate the prescaler value as follows:

```
mathematica  Copy code
Prescaler = (Input Clock Frequency / Timer Clock Frequency) - 1
= (16 MHz / 1 MHz) - 1
= 15
```

So, you would set the prescaler value to 15.

However, please note that the actual input clock frequency might vary depending on how your microcontroller is configured (PLL settings, clock dividers, etc.). You should consult the datasheet, reference manual, or the Clock Configuration Tool provided by STMicroelectronics to determine the exact input clock frequency and its configuration. Then, you can use the formula above to calculate the appropriate prescaler value for your specific setup.

ChatGPT

To configure the TMR16 clock prescaler correctly, you need to determine the desired frequency for your timer. The prescaler value will depend on the clock source of the timer and the desired frequency. Here's a general guideline:

1. Identify Clock Source: Determine what clock source your TMR16 is using. It could be the system clock (SYSLCK), a peripheral clock (e.g., APB1 or APB2), or an external clock source.
2. Calculate Desired Timer Frequency: Determine the desired frequency for your timer. Let's say you want a frequency of $f_{desired}$ Hz.
3. Determine Timer Counter Range: Check the maximum counter value for your TMR16. This will depend on the resolution of your timer, typically it's 16-bit, meaning it can count from 0 to 65536.
4. Calculate Prescaler Value: Use the formula to calculate the prescaler value:

$$\text{Prescaler} = \left\lceil \frac{\text{Clock Source Frequency}}{f_{desired} \times \text{Timer Counter Range}} \right\rceil - 1$$

1. Configure Prescaler: Set the prescaler value in the appropriate register of your microcontroller.

For example, if your TMR16 is using the system clock (SYSLCK) as the clock source and you want a frequency of 1kHz, and assuming the timer is 16-bit, you would calculate the prescaler as follows:

Assuming:

- $f_{SYSLCK} = 100 \text{ MHz}$
- $f_{desired} = 1 \text{ kHz}$
- Timer Counter Range = 65536 (16-bit timer)

$$\text{Prescaler} = \left\lceil \frac{100 \text{ MHz}}{1 \text{ kHz} \times 65536} \right\rceil - 1$$

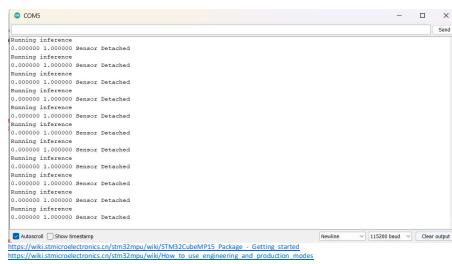
$$\text{Prescaler} = \left\lceil \frac{100000}{65536} \right\rceil - 1$$

$$\text{Prescaler} = \lceil 1.525 \rceil - 1 = 2 - 1 = 1$$

So, you would configure the prescaler of your TMR16 to 1.

Make sure to adjust the values according to your specific requirements and the capabilities of your microcontroller. If you provide more details about your specific microcontroller and requirements, I can help you with a more specific calculation.




```

170     AI_SunInData, aiOutData);
171     /* Output results */
172     for (uint8_t i = 0; i < AI_NUMTO_NODC_CONF_1_SIZE; i++) {
173         buf_len = spinbufRead(*buf, &iOutData[i]);
174         HAL_DMAT_Transmit1ch4r4((uint8_t *)buf, buf_len, 100);
175     }
176 //    uint8_t i;
177 //    for (i = 0; i < AI_NUMTO_NODC_CONF_1_SIZE; i++) {
178 //        buf_len = spinbufRead(*buf, &iOutData[i]);
179 //        HAL_DMAT_Transmit1ch4r4((uint8_t *)buf, buf_len, 100);
180 //    }
181 //    if (aiOutData[0] >= 0x80) {
182 //        uint8_t i;
183 //        for (i = 0; i < AI_NUMTO_NODC_CONF_1_SIZE;
184 //            buf_len = spinbufRead(*buf, &iOutData[i]));
185 //            HAL_DMAT_Transmit1ch4r4((uint8_t *)buf, buf_len, 100);
186 //        }
187 //    }
188 //    else {
189 //        uint8_t i;
190 //        for (i = 0; i < AI_NUMTO_NODC_CONF_1_SIZE;
191 //            buf_len = spinbufRead(*buf, &iOutData[i]));
192 //            HAL_DMAT_Transmit1ch4r4((uint8_t *)buf, buf_len, 100);
193 //        }
194 //    }
195 //    /* Wait before doing it again */
196 //    HAL_Delay(5000);
197 //}
198 ///* USER CODE END WHILE */
199 ///* USER CODE BEGIN 3 */
200 //*/
201 ///* USER CODE END 3 */
202 //*/
203 //*/
204 // Brief System Clock Configuration
205 // * Reset the System Clock via HAL库
206 // * @param None
207 void SystemClock_Config(void)
208 {
209     /* Configure the system clock */
210     /* System Clock Selection: PLL as System Clock source */
211     /* HSE = 48MHz */
212     /* PLL = HSE * 4 = 192MHz */
213     /* HSE = 48MHz */
214     /* HSI = 4MHz */
215     /* HSE = 48MHz */
216     /* HSI = 4MHz */
217     /* HSE = 48MHz */
218     /* HSI = 4MHz */
219     /* HSE = 48MHz */
220     /* HSI = 4MHz */
221     /* HSE = 48MHz */
222     /* HSI = 4MHz */
223     /* HSE = 48MHz */
224     /* HSI = 4MHz */
225     /* HSE = 48MHz */
226     /* HSI = 4MHz */
227     /* HSE = 48MHz */
228     /* HSI = 4MHz */
229     /* HSE = 48MHz */
230     /* HSI = 4MHz */
231     /* HSE = 48MHz */
232     /* HSI = 4MHz */
233     /* HSE = 48MHz */
234     /* HSI = 4MHz */
235     /* HSE = 48MHz */
236     /* HSI = 4MHz */
237     /* HSE = 48MHz */
238     /* HSI = 4MHz */
239     /* HSE = 48MHz */
240     /* HSI = 4MHz */
241     /* HSE = 48MHz */
242     /* HSI = 4MHz */
243     /* HSE = 48MHz */
244     /* HSI = 4MHz */
245     /* HSE = 48MHz */
246     /* HSI = 4MHz */
247     /* HSE = 48MHz */
248     /* HSI = 4MHz */
249     /* HSE = 48MHz */
250     /* HSI = 4MHz */
251     /* HSE = 48MHz */
252     /* HSI = 4MHz */
253     /* HSE = 48MHz */
254     /* HSI = 4MHz */
255     /* HSE = 48MHz */
256     /* HSI = 4MHz */
257     /* HSE = 48MHz */
258     /* HSI = 4MHz */
259     /* HSE = 48MHz */
260     /* HSI = 4MHz */
261     /* HSE = 48MHz */
262     /* HSI = 4MHz */
263     /* HSE = 48MHz */
264     /* HSI = 4MHz */
265     /* HSE = 48MHz */
266     /* HSI = 4MHz */
267     /* HSE = 48MHz */
268     /* HSI = 4MHz */
269     /* HSE = 48MHz */
270     /* HSI = 4MHz */
271     /* HSE = 48MHz */
272     /* HSI = 4MHz */
273     /* HSE = 48MHz */
274     /* HSI = 4MHz */
275     /* HSE = 48MHz */
276     /* HSI = 4MHz */
277     /* HSE = 48MHz */
278     /* HSI = 4MHz */
279     /* HSE = 48MHz */
280     /* HSI = 4MHz */
281     /* HSE = 48MHz */
282     /* HSI = 4MHz */
283     /* HSE = 48MHz */
284     /* HSI = 4MHz */
285     /* HSE = 48MHz */
286     /* HSI = 4MHz */
287     /* HSE = 48MHz */
288     /* HSI = 4MHz */
289     /* HSE = 48MHz */
290     /* HSI = 4MHz */
291     /* HSE = 48MHz */
292     /* HSI = 4MHz */
293     /* HSE = 48MHz */
294     /* HSI = 4MHz */
295     /* HSE = 48MHz */
296     /* HSI = 4MHz */
297     /* HSE = 48MHz */
298     /* HSI = 4MHz */
299     /* HSE = 48MHz */
299

```

