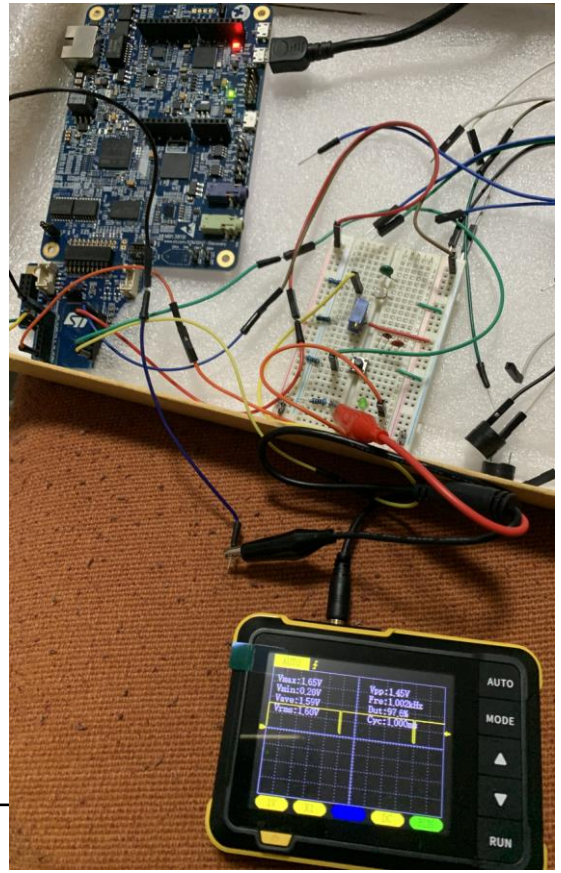
ARM

Practical project for STM32H7 embedded system (informative, additional content)

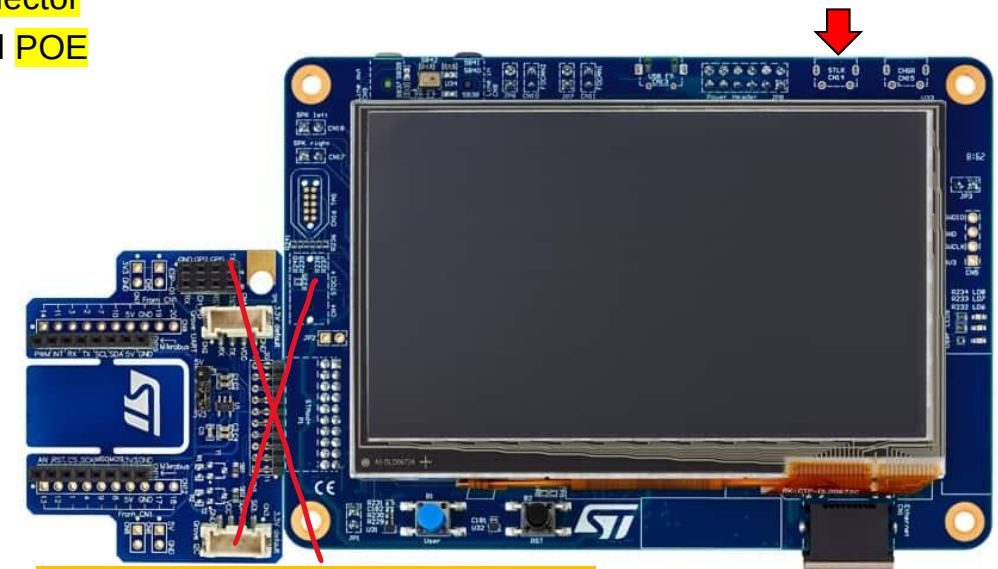
Blinking LED diode and buzzer melody

Arduino IDE + STM32 extension



STM32H750B-DK Discovery development system

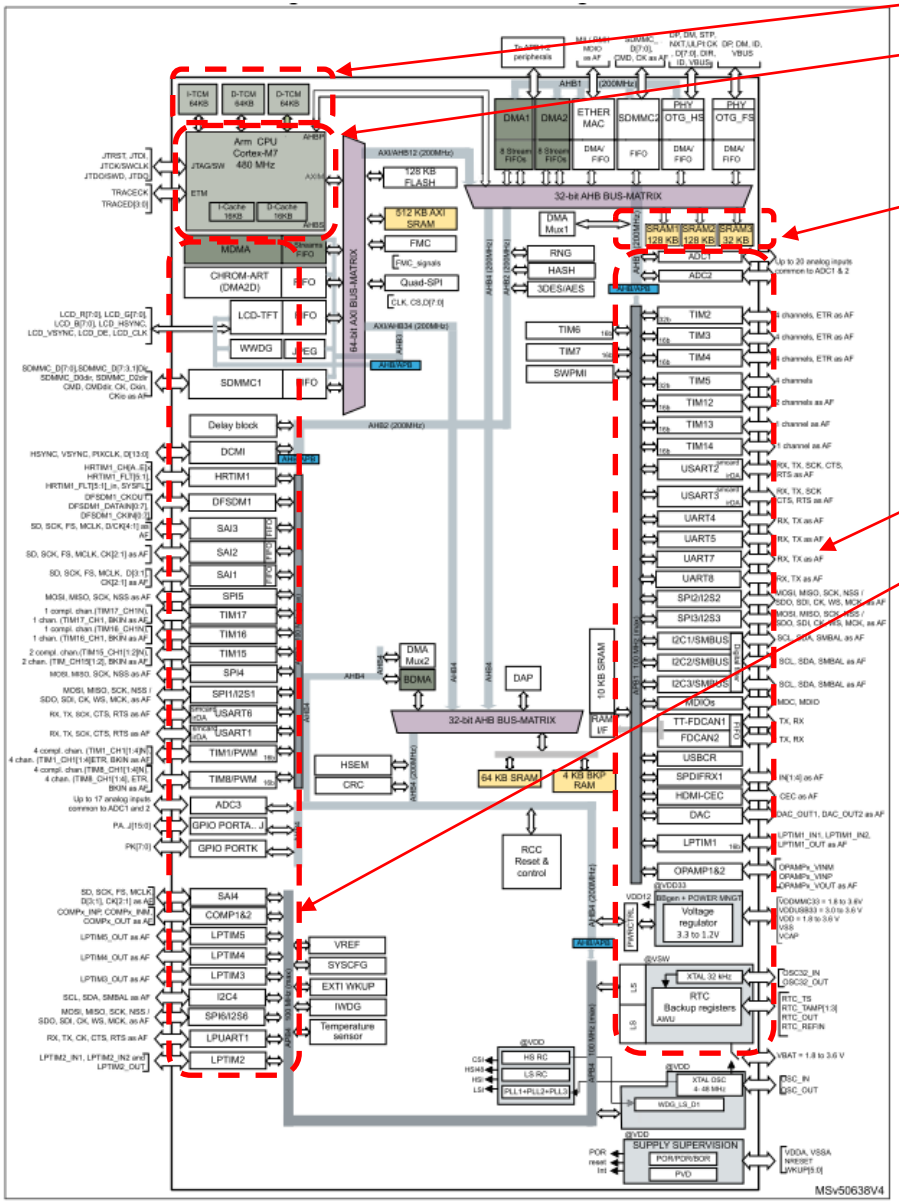
- Arm® Cortex® core-based microcontroller with 128 Kbytes (STM32H750XBH6) of Flash memory and 1 Mbyte of RAM, in TFBGA240+25 package
- 4.3” RGB interface LCD with touch panel connector
- Ethernet compliant with IEEE-802.3-2002, and POE
- USB OTG FS with Micro-AB connector
- SAI audio codec
- One ST-MEMS digital microphone
- 2 x 512-Mbit Quad-SPI NOR Flash memory
- 128-Mbit SDRAM
- 4-Gbyte on-board eMMC
- 1 user and reset push-button
- Fanout daughterboard
- 2 x FDCANs
- Board connectors:
  - USB FS Micro-AB connectors
  - ST-LINK Micro-B USB connector
  - USB power Micro-B connector
  - Ethernet RJ45
  - Stereo headset jack including analog microphone input
  - Audio header for external speakers
  - Arduino™ Uno V3 expansion connectors
  - STMod+

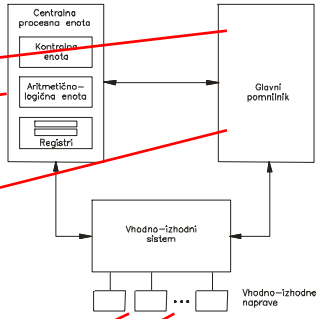


Wrong connection shown on photo !

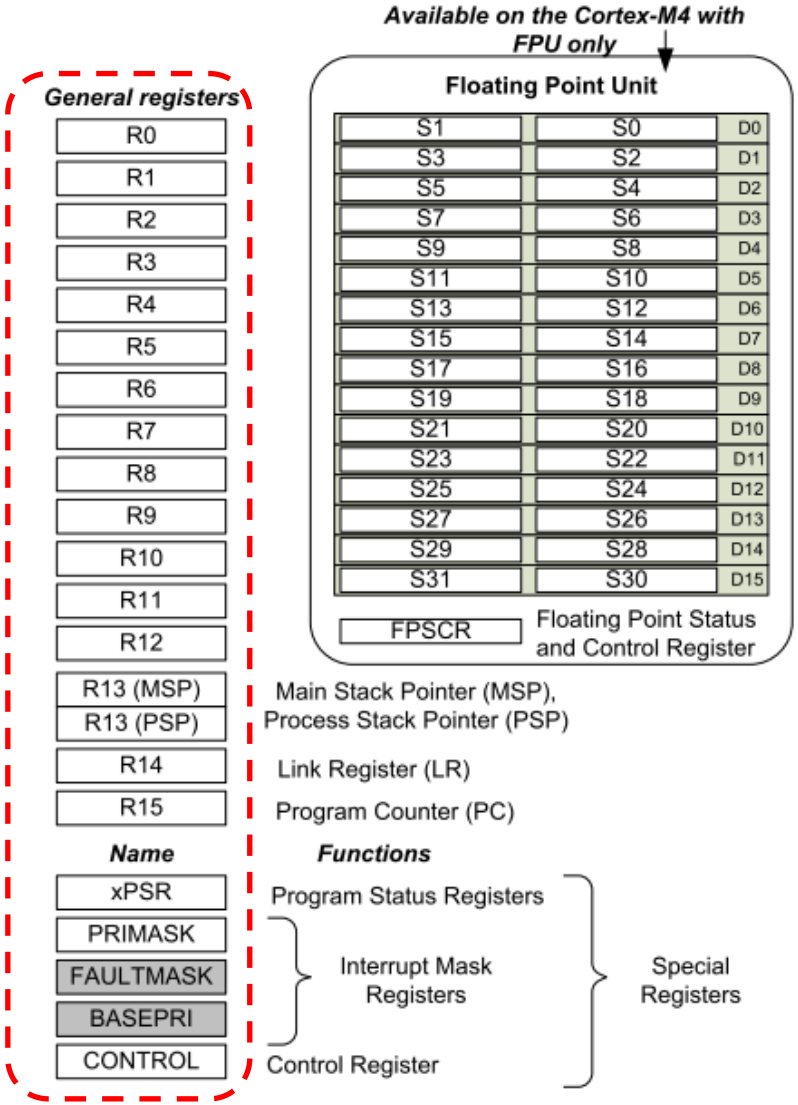
https://www.st.com/en/evaluation-tools/stm32h750b-dk.html

STM32H750XB MCU



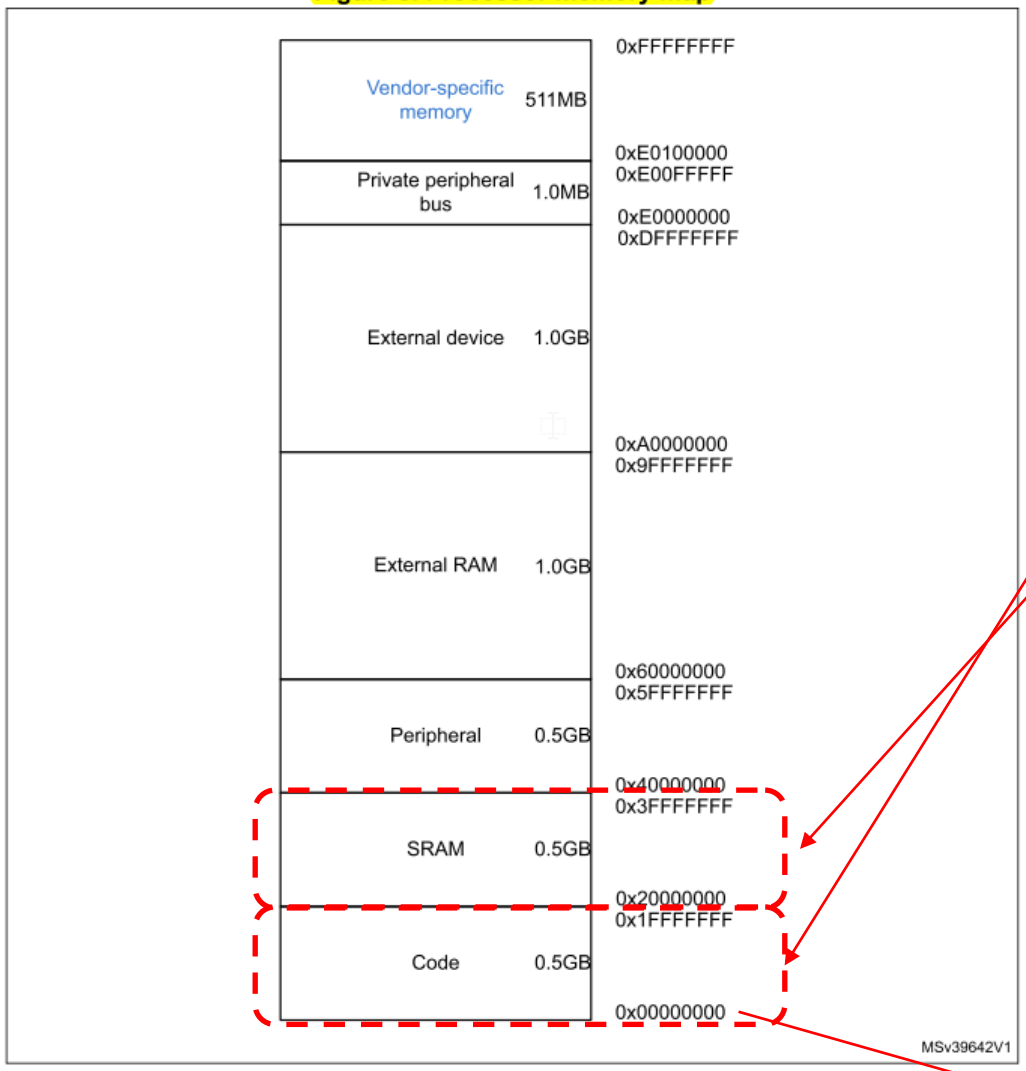


ARM Cortex M – Program model



ARM Cortex M – Memory (address) space

Figure 8. Processor memory map



Memory mapping description – linker

```
MEMORY
{
  FLASH (rx)    : ORIGIN = 0x08000000, LENGTH = 128K
  DTCMRAM (xrw) : ORIGIN = 0x20000000, LENGTH = 128K
  RAM_D1 (xrw)  : ORIGIN = 0x24000000, LENGTH = 512K
  RAM_D2 (xrw)  : ORIGIN = 0x30000000, LENGTH = 288K
  RAM_D3 (xrw)  : ORIGIN = 0x38000000, LENGTH = 64K
  ITCMRAM (xrw) : ORIGIN = 0x00000000, LENGTH = 64K
}
```

SysTick vector	0x0000003C
PendSV vector	0x00000038
Not used	0x00000034
Debug Monitor vector	0x00000030
SVC vector	0x0000002C
Not used	0x00000028
Not used	0x00000024
Not used	0x00000020
SecureFault (ARMv8-M Mainline)	0x0000001C
Usage Fault vector	0x00000018
Bus Fault vector	0x00000014
MemManage vector	0x00000010
HardFault vector	0x0000000C
NMI vector	0x00000008
Reset vector	0x00000004
MSP initial value	0x00000000

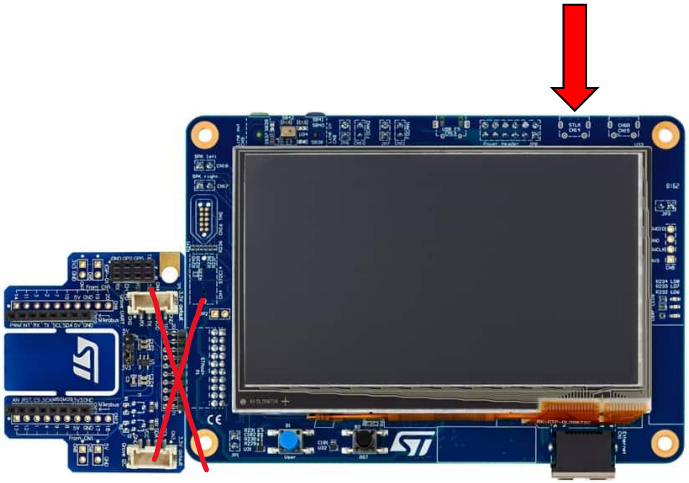
ARM Cortex M – Vector table – Initial start

Vector Table	Vector address (initial)
Interrupt#239 vector	0x000003FC
Interrupt#31 vector	0x000000BC
Interrupt#1 vector	0x00000044
Interrupt#0 vector	0x00000040
SysTick vector	0x0000003C
PendSV vector	0x00000038
Not used	0x00000034
Debug Monitor vector	0x00000030
SVC vector	0x0000002C
Not used	0x00000028
Not used	0x00000024
Not used	0x00000020
SecureFault (ARMv8-M Mainline)	0x0000001C
Usage Fault vector	0x00000018
Bus Fault vector	0x00000014
MemManage vector	0x00000010
HardFault vector	0x0000000C
NMI vector	0x00000008
Reset vector	0x00000004
MSP initial value	0x00000000

Address 4 („Reset vector“) contains the address of the 1st instruction of the „initialization“ part, i.e. the initial code is located in the Reset_Handler subroutine.

```
.section  .text.Reset_Handler
 .weak   Reset_Handler
 .type   Reset_Handler, %function

Reset_Handler:

  …

  ldr   sp, =_estack      /* set stack pointer */

/* Call the application's entry point.*/

  bl  main

  bx  lr
```

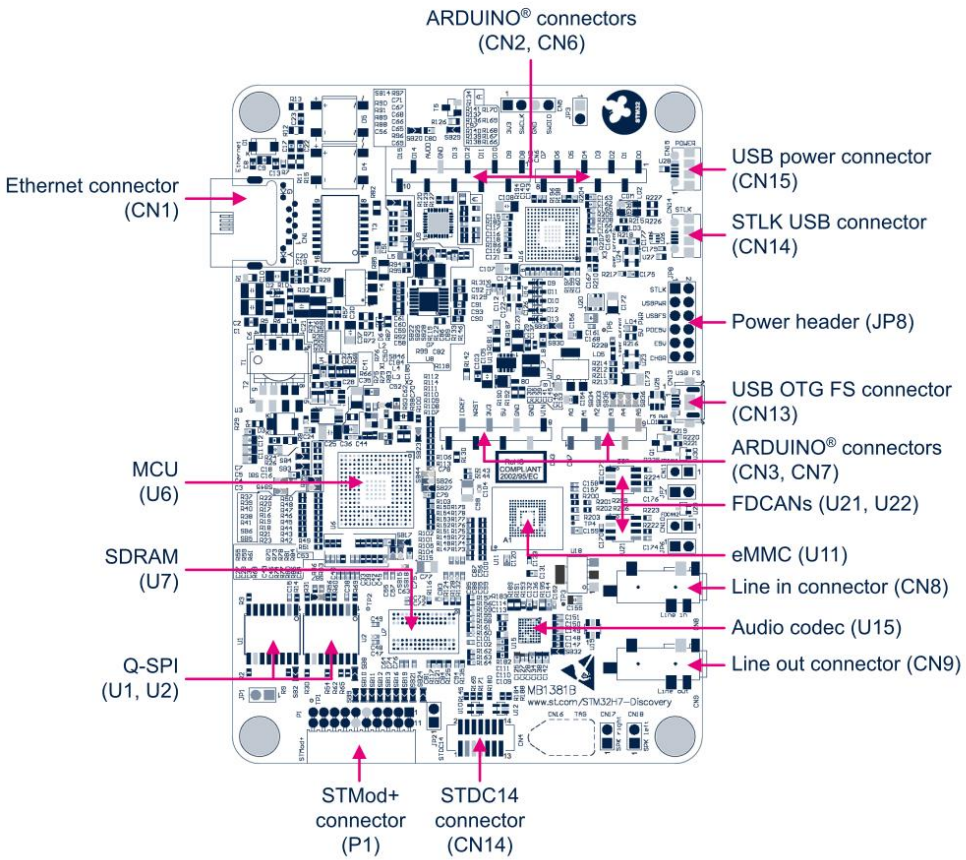
Work on STM32H750B-DK board



Connection :

- **Micro USB** connector above the screen (arrow!), screen lights up.

Figure 5. **STM32H745I-DISCO and STM32H750B-DK Discovery board bottom layout**



Work on STM32H750B-DK board

Compile

Upload

STM32H750_Basic_GPIO_SpeedTest_Serial | Arduino 1.8.19

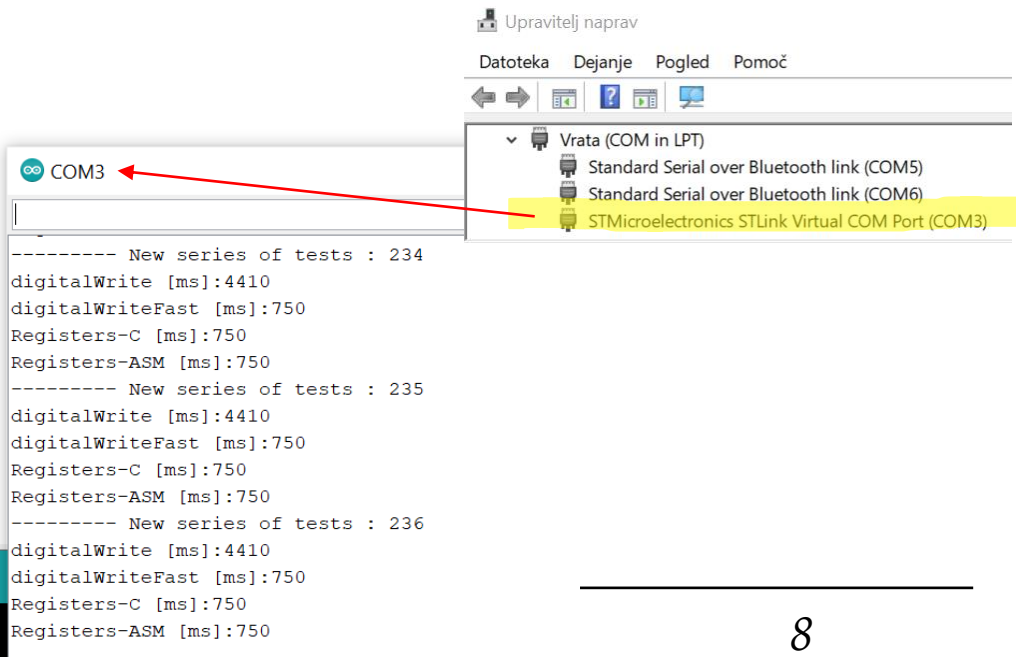
File Edit Sketch Tools Help

STM32H750_Basic_GPIO_SpeedTest_Serial

```
// #include "stm32h750.h"

int i,ms,RegisterCMs,RegisterASMMs,WriteFastMs,WriteMs;
int j = 1;

//                        RX    TX
HardwareSerial Serial3(PB_11, PB_10);



void setup() {
        // put your setup code here, to run once:
        pinMode(PD_12, OUTPUT);
        pinMode(PD_13, OUTPUT);

        pinMode(PI_13, OUTPUT);
        pinMode(PJ_2, OUTPUT);

        Serial3.begin(115200);
}


void loop() {
        // put your main code here, to run repeatedly:

        Serial3.print("--------- New series of tests : ");
        Serial3.println(j);


        // 1. test - I/O access through Arduino library API
        i=0;
        Serial3.print("digitalWrite [ms]:");
        ms=millis();
        while (i<1000000) {
          // statement(s)
                digitalWrite(PD_12, HIGH);
```

Special distribution for STM32H750B-DK (e-učilnica) :

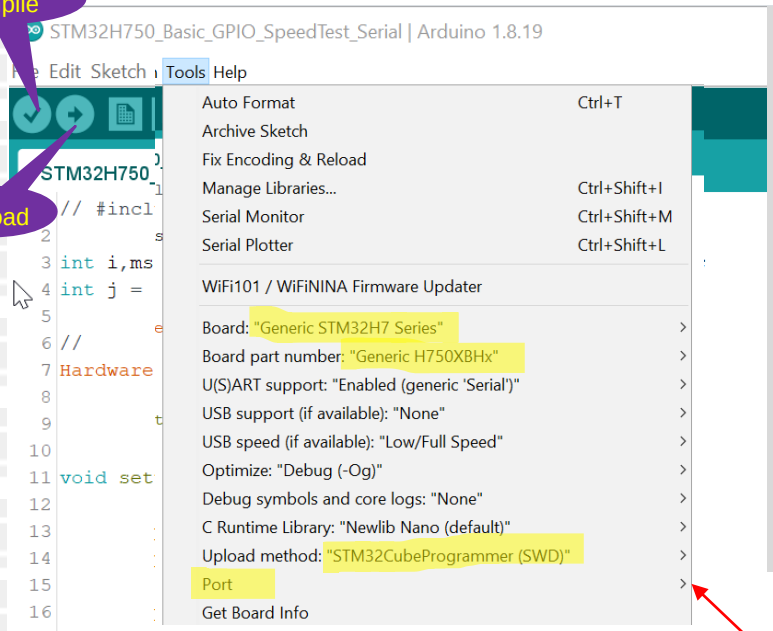
- ***Arduino IDE environment with STM32 extension***
  - portable installation (to D:\RAVINOR)
    - https://github.com/LAPSyLAB/STM32Duino_for_H7/releases/tag/V0.1
  - features:
    - simple programming
    - use of basic I/O devices
      - LED diods
      - GPIO inputs/outputs
      - PWM (play melody on buzzer)
      - https://github.com/LAPSyLAB/STM32Duino_for_H7



Work on STM32H750B-DK board - Settings

Compile

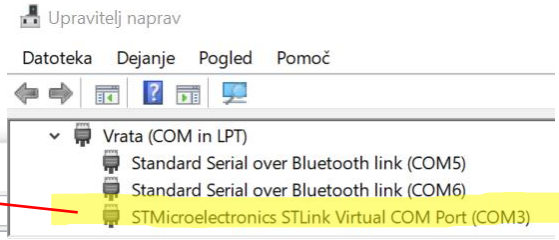
Upload

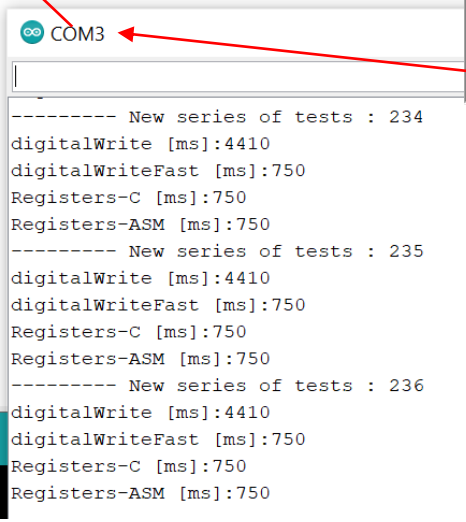



```
int i,ms
int j =
//
Hardware
void set
        pinMode(PJ_2, OUTPUT);
        Serial3.begin(115200);
}
void loop() {
        // put your main code here, to run repeatedly:
        Serial3.print("--------- New series of tests : ");
        Serial3.println(j);
        // 1. test - I/O access through Arduino library API
        i=0;
        Serial3.print("digitalWrite [ms]:");
        ms=millis();
        while (i<1000000) {
          // statement(s)
                digitalWrite(PD_12, HIGH);
```

Special distribution for STM32H750B-DK (e-učilnica) :

- ***Arduino IDE environment with STM32 extension***
  - portable installation (to D:\RAVINOR)
    - https://github.com/LAPSyLAB/STM32Duino_for_H7/releases/tag/V0.1
  - features:
    - simple programming
    - use of basic I/O devices
      - LED diods
      - GPIO inputs/outputs
      - PWM (play melody on buzzer)
      - https://github.com/LAPSyLAB/STM32Duino_for_H7





Basic program for LED blinking and serial communication

STM32H750_Basic_LED_Serial.ino

```
int i;
//                        RX    TX
HardwareSerial Serial3(PB_11, PB_10);

void setup() {
  // put your setup code here, to run once:
  // initialize digital pins PI13,PJ2 as outputs.
  pinMode(PI_13, OUTPUT);
  pinMode(PJ_2, OUTPUT);
  Serial3.begin(115200);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(PI_13, HIGH);   // turn the LED on (HIGH is the voltage level)
  digitalWrite(PJ_2, LOW);    // turn the LED off by making the voltage LOW

  delay(1000);              // wait for a second
  i++;

  digitalWrite(PI_13, LOW);    // turn the LED off by making the voltage LOW
  digitalWrite(PJ_2, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second

  Serial3.print("Hello World: ");
  Serial3.println(i);
}
```

Keypoints:

- int i;
  - variables declaration (integer)
- setup () { … }:
  - *runs once on startup*
- loop () { … }:
  - *runs permanently*

Functions:

pinMode (set pin input/output)
digitalWrite (set state on pin 1/0)
delay (n) – wait n ms
Serial3 – serial communication object
Print
Println (add newline)

Basic program for control of GPIO pins (1/0)

STM32H750_Basic_GPIO_SpeedTest_Serial.ino

```
pinMode(PD_12, OUTPUT);

digitalWrite(PD_12, HIGH);
digitalWrite(PD_12, LOW);


digitalWriteFast(PD_12, HIGH);
digitalWriteFast(PD_12, LOW);

//Set D12 (HIGH)
GPIOD->BSRR = 0b0001000000000000 << 16; //move to upper 16 bits
//Set D12 (LOW)
GPIOD->BSRR = 0b0001000000000000; //move to upper 16 bits

 asm (

  "  ldr.w r0, = 0x58020C18                \n" // GPIOD->BSRR =   (0x58020C00 + 0x18 ) SetReset Register

  "  ldr r3,   = 0b0001000000000000 << 16  \n" // Bit 12
  "  str r3, [r0]                           \n" // Set port high

  "  ldr r3,   = 0b0001000000000000        \n" // Bit 12
  "  str r3, [r0]                           \n" // Set port low
: : : "r0", "r3");
```

Control of GPIO pins on different levels

- **Higher level programming language:**
  - Arduino API functions (digitalWrite)
  - STM32 API functions (digitalWriteFast)
- **Direct control of GPIO pins:**
  - write access to registers – higher level
  - write access to registers – assembler

```
--------- New series of tests : 519
digitalWrite [ms]:4410
digitalWriteFast [ms]:750
Registers-C [ms]:750
Registers-ASM [ms]:750
```

Functions:

digitalWrite	*(Arduino API)*
digitalWriteFast	*(STM32 HAL API)*
GPIOD->BSRR =	*(direct write – C++)*
str r3, [r0]	*(direct write – assembler)*

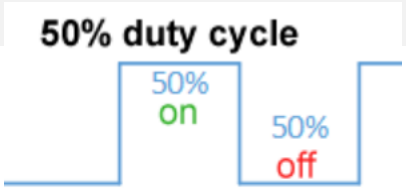
Basic program for buzzer control with PWM signal

STM32H750_Basic_PWM_Melody.ino

```
void setup() {
        // put your setup code here, to run once:
        // initialize digital pins PI13,PJ2 as outputs.
        pinMode(PA_3, OUTPUT);

        Serial3.begin(115200);
}

void loop() {

          // Write PWM signal at frequency to pin
          analogWriteFrequency(Frekvenca_Hz); // Set PMW period to Note Hz
          analogWrite(PA_3, 128); // Start PWM on Pin, at Frekvenca_Hz with 50% duty cycle
          delay(noteDuration);

 // stop the tone playing:
          analogWrite(PA_3, 0); // Start PWM on Pin, at melody[thisNote] Hz with 0% duty cycle
          // to distinguish the notes, set a minimum time between them.
          int pauseBetweenNotes = noteDuration * 0.3;
          delay(pauseBetweenNotes);
          …
}
```

Control with square signal, connected to buzzer:

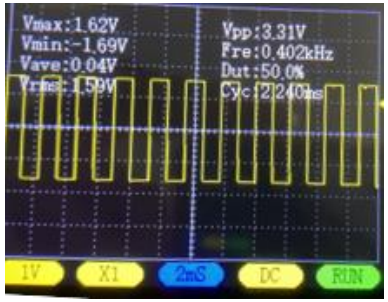
- half period high, half period low state
- variations of frequency, i.e. period of the signal produces musical tones




```
Note (#0) [Hz]:349    Duration:250
Note (#1) [Hz]:392    Duration:250
Note (#2) [Hz]:440    Duration:125
Note (#3) [Hz]:392    Duration:250
Note (#4) [Hz]:440    Duration:125
Note (#5) [Hz]:466    Duration:250
```

pitches.h:

```
/*****************
 * Public Constants
 *****************/

…
#define NOTE_DS4 311
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_FS4 370
#define NOTE_G4  392
#define NOTE_GS4 415
#define NOTE_A4  440
#define NOTE_AS4 466
```



Functions:

analogWriteFrequency (sets frequency)
analogWrite (pin, 50% duty)

Frequency must be set before the call of analogWrite.

Challenges

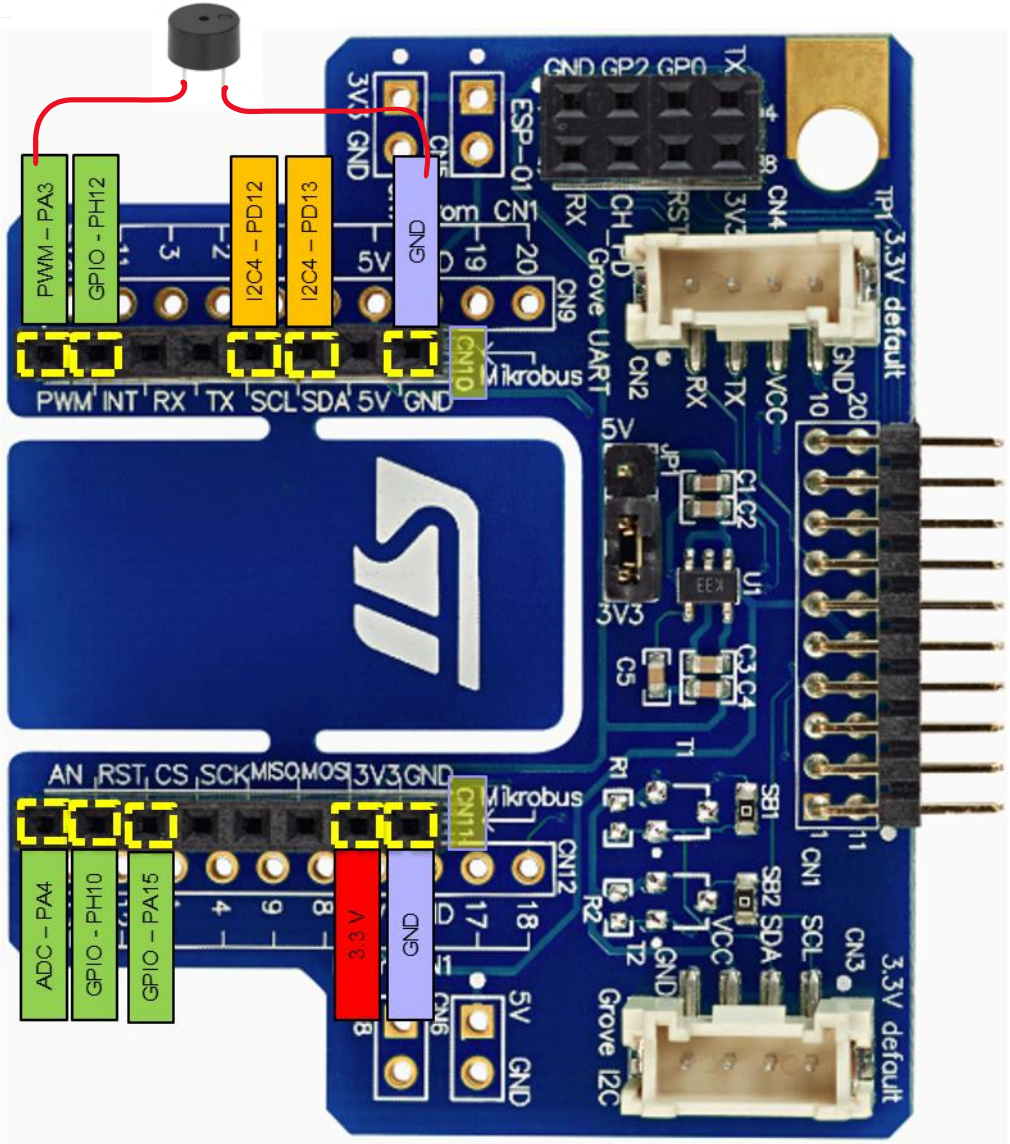
- **LED blinking**
  - shorten time of on/off state
  - change relation oif on/off duration („dimmer“, „PWM-dimmer“)

- **Control of GPIO pins on different levels**
  - high level programming language:
    - Arduino API functions (digitalWrite)
    - STM32 HAL API functions (digitalWriteFast)
  - direct control of GPIO:
    - write access to registers – higher level
    - write access to registers – assembler

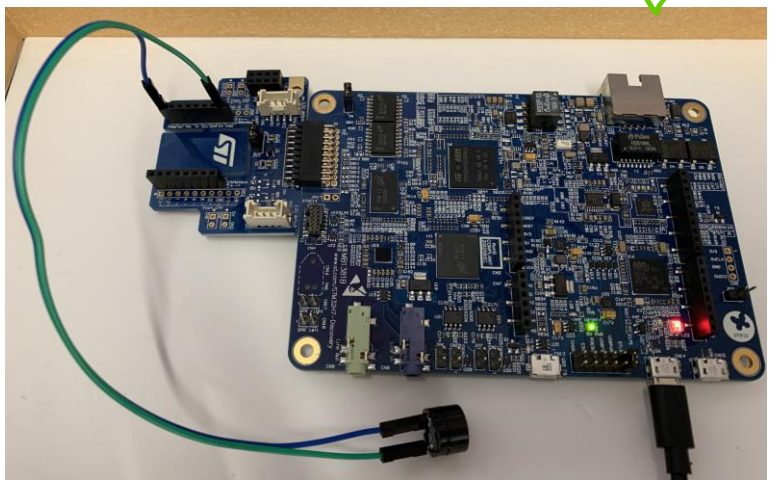
- **Use of extension board:**
  - set pins on/off on board
  - Generate PWM signal to control buzzer:
    - connect buzzer to PA3 & GND
    - tones, melodies, …

STM32H750B – DISCOVERY StMod+ connector

3.3V !!!



Proper connection



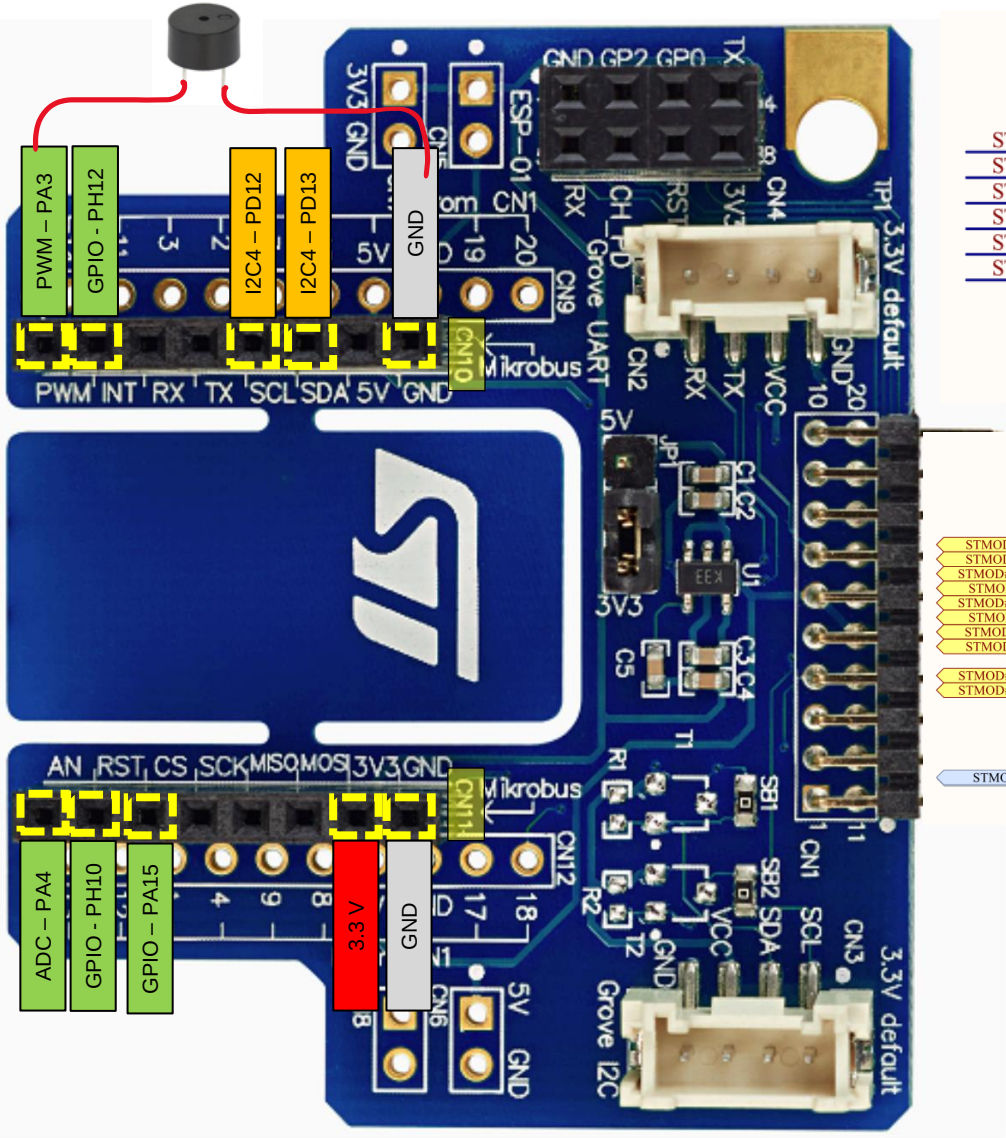
Improper connection !!!

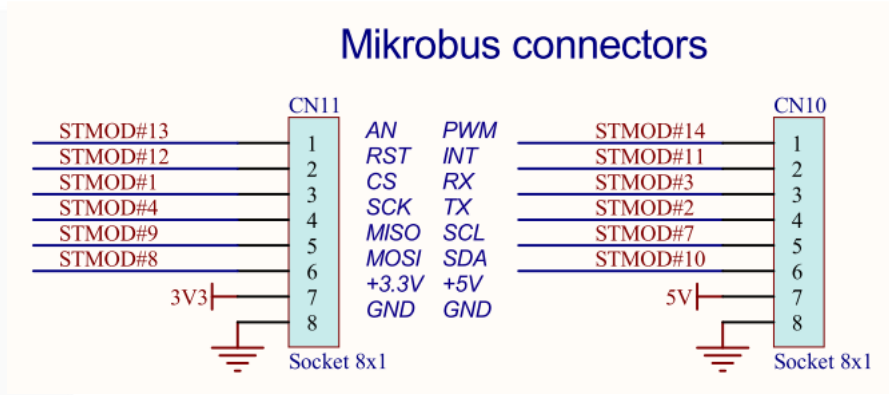
https://www.st.com/en/evaluation-tools/stm32h750b-dk.html

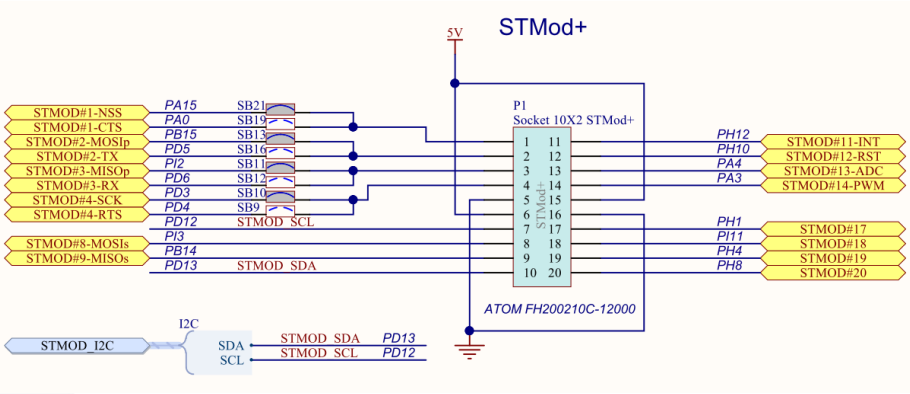
Wrong connection shown on photo !

STM32H750B – DISCOVERY StMod+ connector

3.3V !!!







Informative, additional content

Challenges –Arduino IDE&STM32Duino programming

Language Reference

Arduino programming language can be divided in three main parts: functions, values (variables and constants), and structure.

Functions

For controlling the Arduino board and performing computations.

Digital I/O
digitalRead()
digitalWrite()
pinMode()

Analog I/O
analogRead()
analogReference()
analogWrite()

Zero, Due & MKR Family
analogReadResolution()
analogWriteResolution()

Advanced I/O
noTone()

Math
abs()
constrain()
map()
max()
min()
pow()
sq()
sqrt()

Trigonometry
cos()
sin()
tan()

Characters

Random Numbers
random()
randomSeed()

Bits and Bytes
bit()
bitClear()
bitRead()
bitSet()
bitWrite()
highByte()
lowByte()

External Interrupts
attachInterrupt()
detachInterrupt()

Reference > Libraries > Stm32duino examples

STM32duino Examples

Other

Provides several examples for the Arduino core for STM32 MCUs.
Arduino STM32 core, libraries and examples are available here: https://github.com/stm32duino

Author: several

Maintainer: stm32duino

Read the documentation

Compatibility

This library is compatible with the **stm32** architecture.

- **https://www.arduino.cc/reference/en/**
- **plenty other Internet resources**
- **some specifities for the use of STM32Duino library**
  - https://www.arduino.cc/reference/en/libraries/stm32duino-examples/