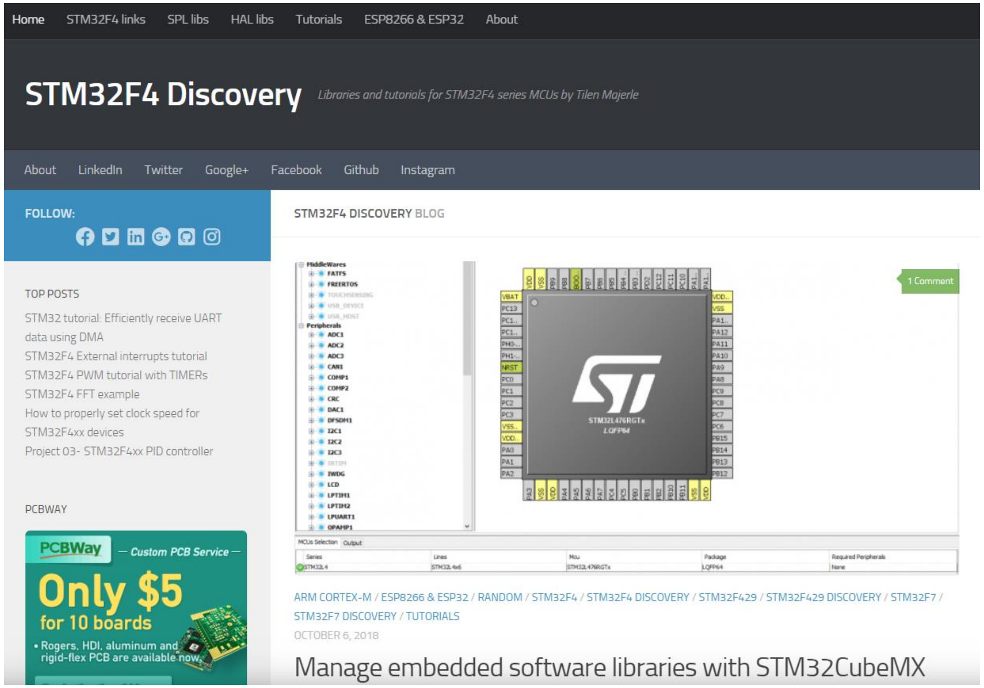
Vhodno izhodne naprave

Laboratorijska vaja 9 - VP 5
VIN projekt, „Edge AI“, STM32 projekti, Miško3 Demo

VIN projekt - VP5: STM32-Edge computing, CubeIDE projekti, Miško3

- VIN projekt
- AI v vgrajenih napravah („Edge Computing“)
- STM32 CubeIDE – Delo s projekti
- STM32 CubeIDE, SPI in LIS3DSH
- STM32 CubeIDE, I2C in CS43L22
- Miško3 – demo projekt

Delo na STM32F4 razvojnem sistemu - zgodba



https://stm32f4-discovery.net/

VIN Projekt – Osnovna platforma

STM32F407 ST Discovery

STM Discovery F4 (Cortex M4)

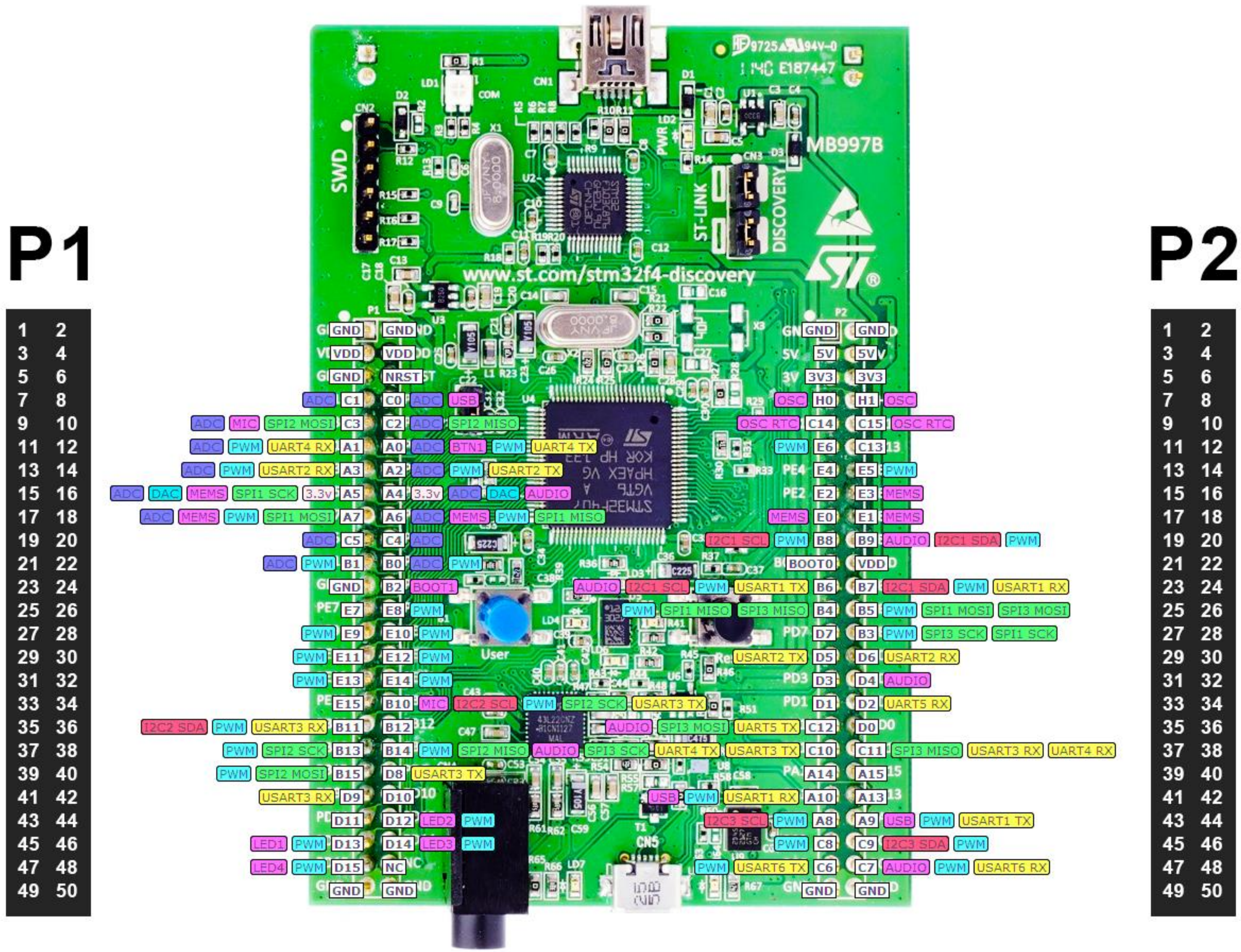
- STM32F407VGT6 microcontroller featuring 32-bit Arm® Cortex®-M4 with FPU core, 1-Mbyte Flash memory and 192-Kbyte RAM in an LQFP100 package
- USB OTG FS
- ST MEMS 3-axis accelerometer
- ST-MEMS audio sensor omni-directional digital microphone
- Audio DAC with integrated class D speaker driver
- User and reset push-buttons
- Eight LEDs:
    - LD1 (red/green) for USB communication
    - LD2 (red) for 3.3 V power on
    - Four user LEDs, LD3 (orange), LD4 (green), LD5 (red) and LD6 (blue)
- Board connectors:
    - USB with Micro-AB
    - Stereo headphone output jack
    - 2.54 mm pitch extension header for all LQFP100 I/Os for quick connection to prototyping board and easy probing
- External application power supply: 3 V and 5 V

STM32

STM32

STM32F4DISCOVERY

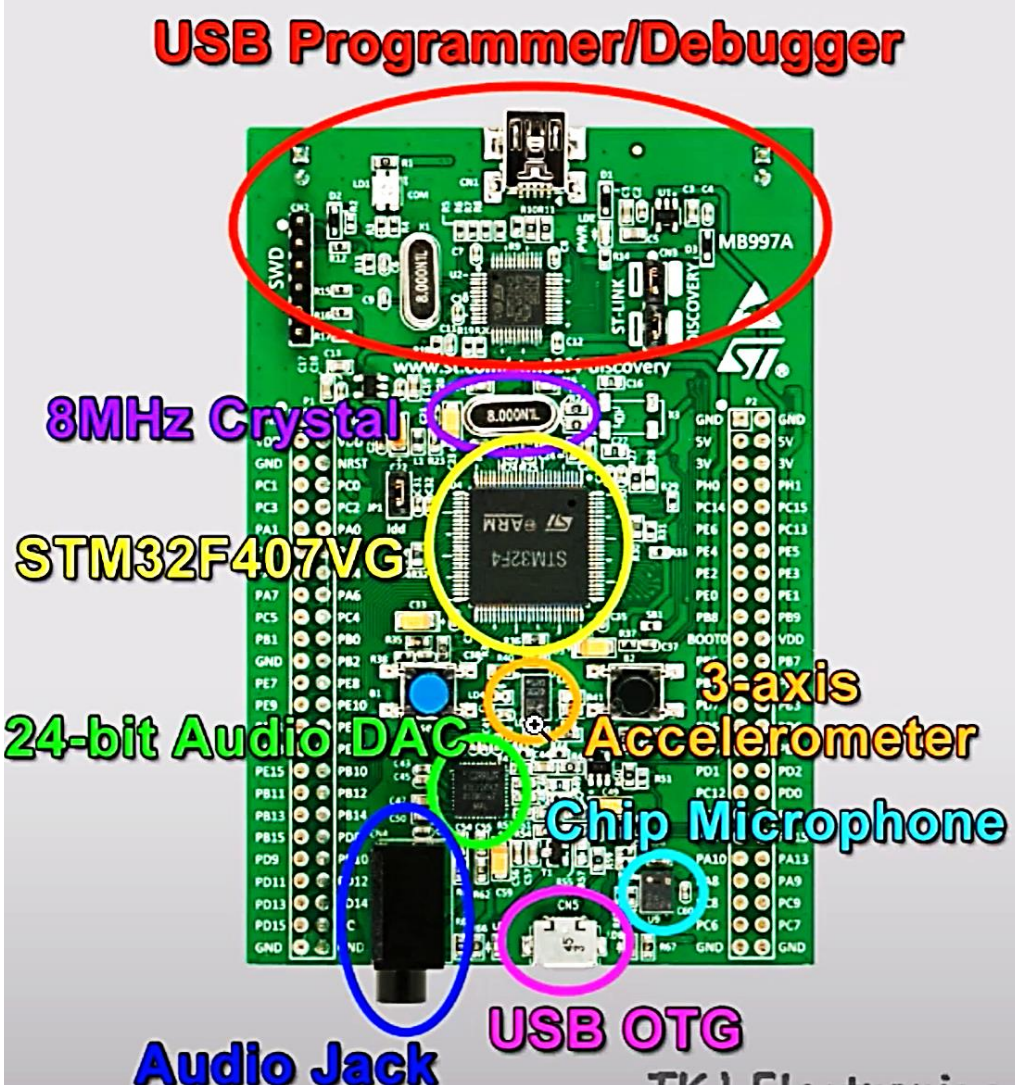
3.3V !!!



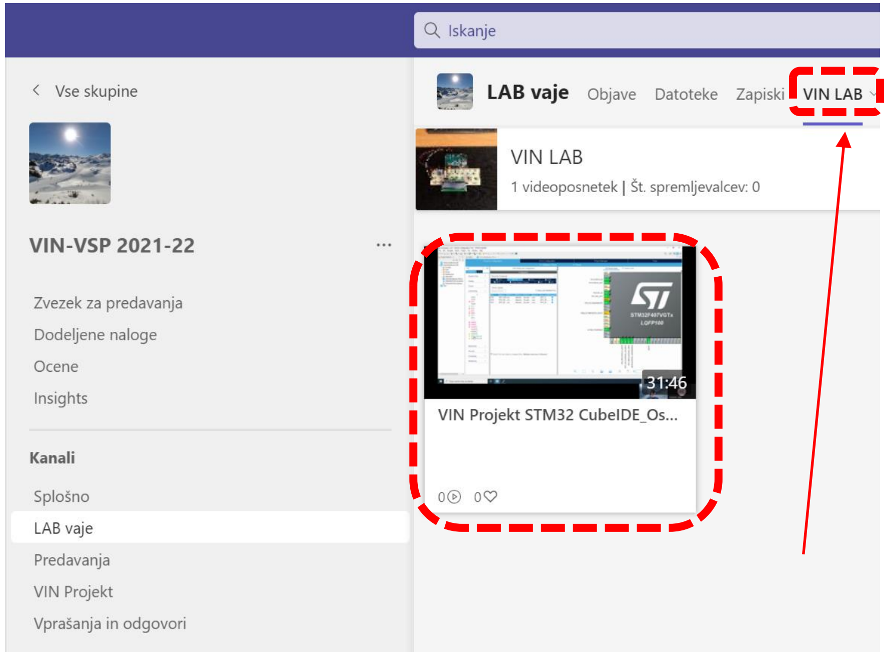
5

STM32F4DISCOVERY

3.3V !!!



Delo na STM32F4 razvojnem sistemu



Lastni viri :

https://github.com/LAPSyLAB/STM32F4_Discovery_VIN_Projects

https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples

https://github.com/LAPSyLAB/ORLab-STM32

Delo na STM32F4 razvojnem sistemu

life.augmented

UM1725

User manual

Description of STM32F4 HAL and low-layer drivers

36 HAL I2C Generic Driver

36.1 I2C Firmware driver registers structures

36.1.1 I2C_InitTypeDef

I2C_InitTypeDef is defined in the stm32f4xx_hal_i2c.h

Data Fields

- ***uint32_t ClockSpeed***
- ***uint32_t DutyCycle***
- ***uint32_t OwnAddress1***
- ***uint32_t AddressingMode***
- ***uint32_t DualAddressMode***
- ***uint32_t OwnAddress2***
- ***uint32_t GeneralCallMode***
- ***uint32_t NoStretchMode***

Field Documentation

- ***uint32_t I2C_InitTypeDef::ClockSpeed***
  Specifies the clock frequency. This parameter must be set to a value lower than 400kHz
- ***uint32_t I2C_InitTypeDef::DutyCycle***
  Specifies the I2C fast mode duty cycle. This parameter can be a value of ***I2C_duty_cycle_in_fast_mode***
- ***uint32_t I2C_InitTypeDef::OwnAddress1***
  Specifies the first device own address. This parameter can be a 7-bit or 10-bit address.
- ***uint32_t I2C_InitTypeDef::AddressingMode***
  Specifies if 7-bit or 10-bit addressing mode is selected. This parameter can be a value of ***I2C_addressing_mode***

Contents

Lastni viri :

https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples

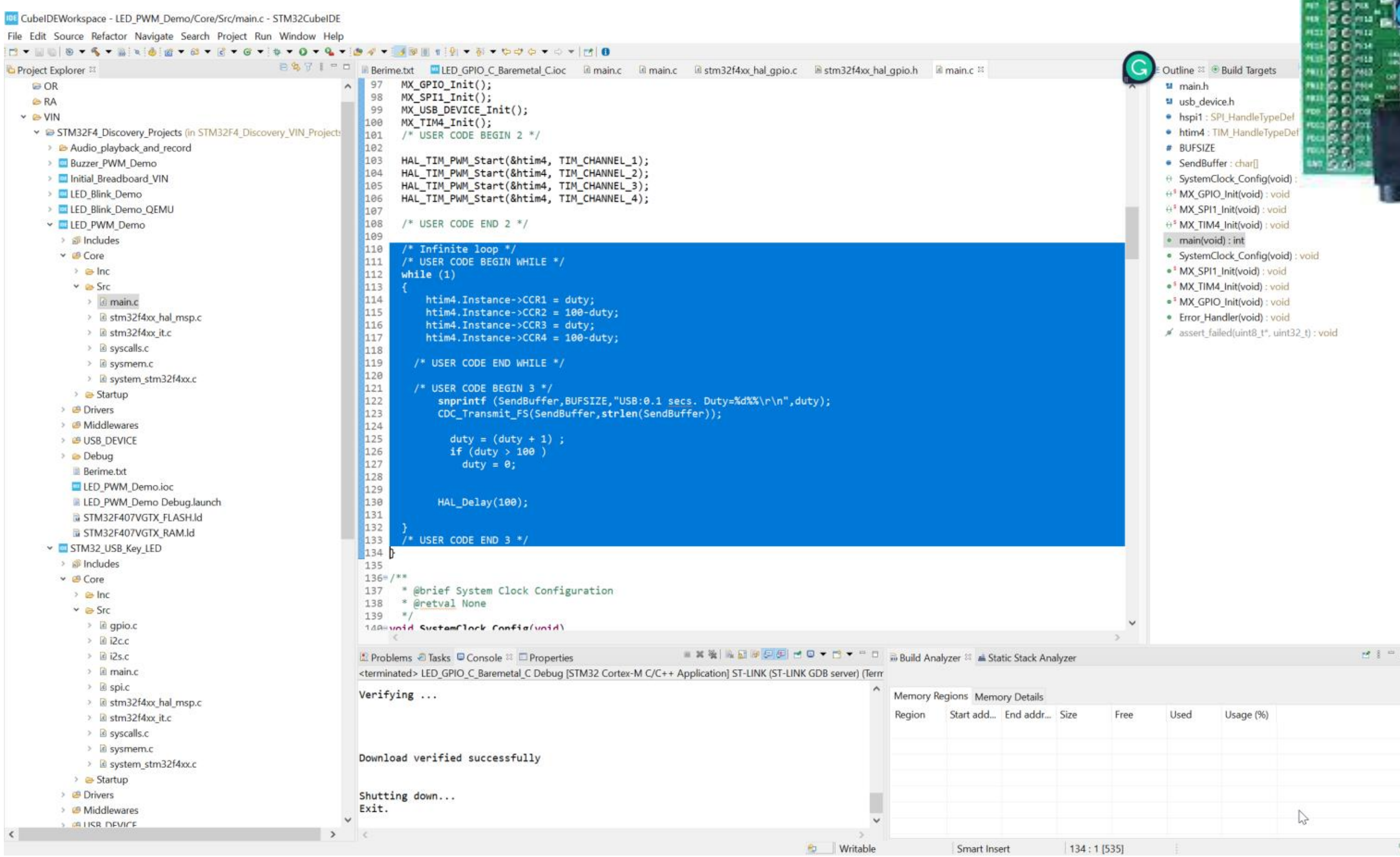
Delo na STM32F4 razvojnem sistemu

Priključitev :

- **Mini USB** priklop na **krajši stranici**, svetita rdeči **LED** diodi

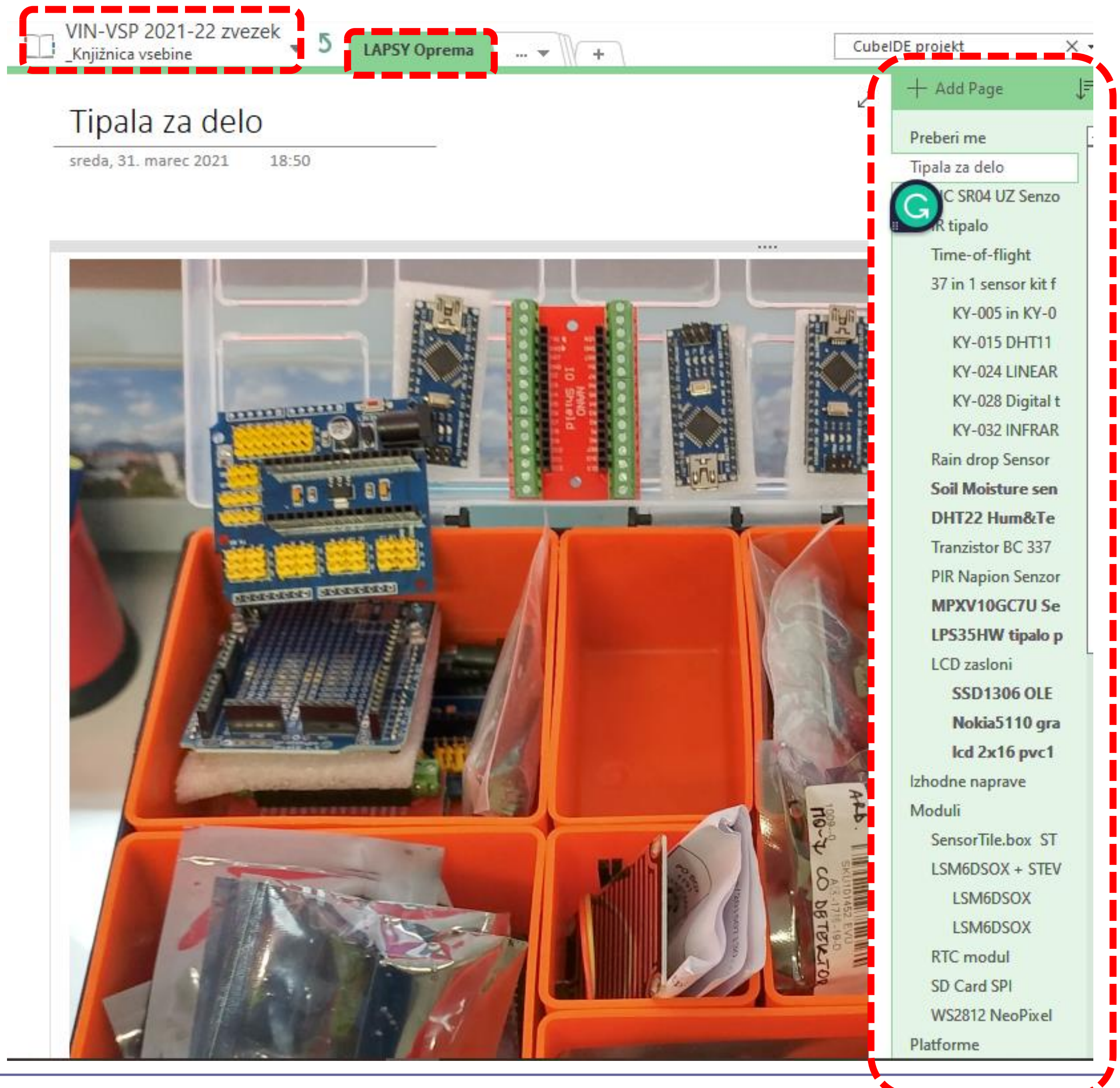
STM32 CubeIDE

- **https://www.st.com/en/development-tools/stm32cubeide.html**



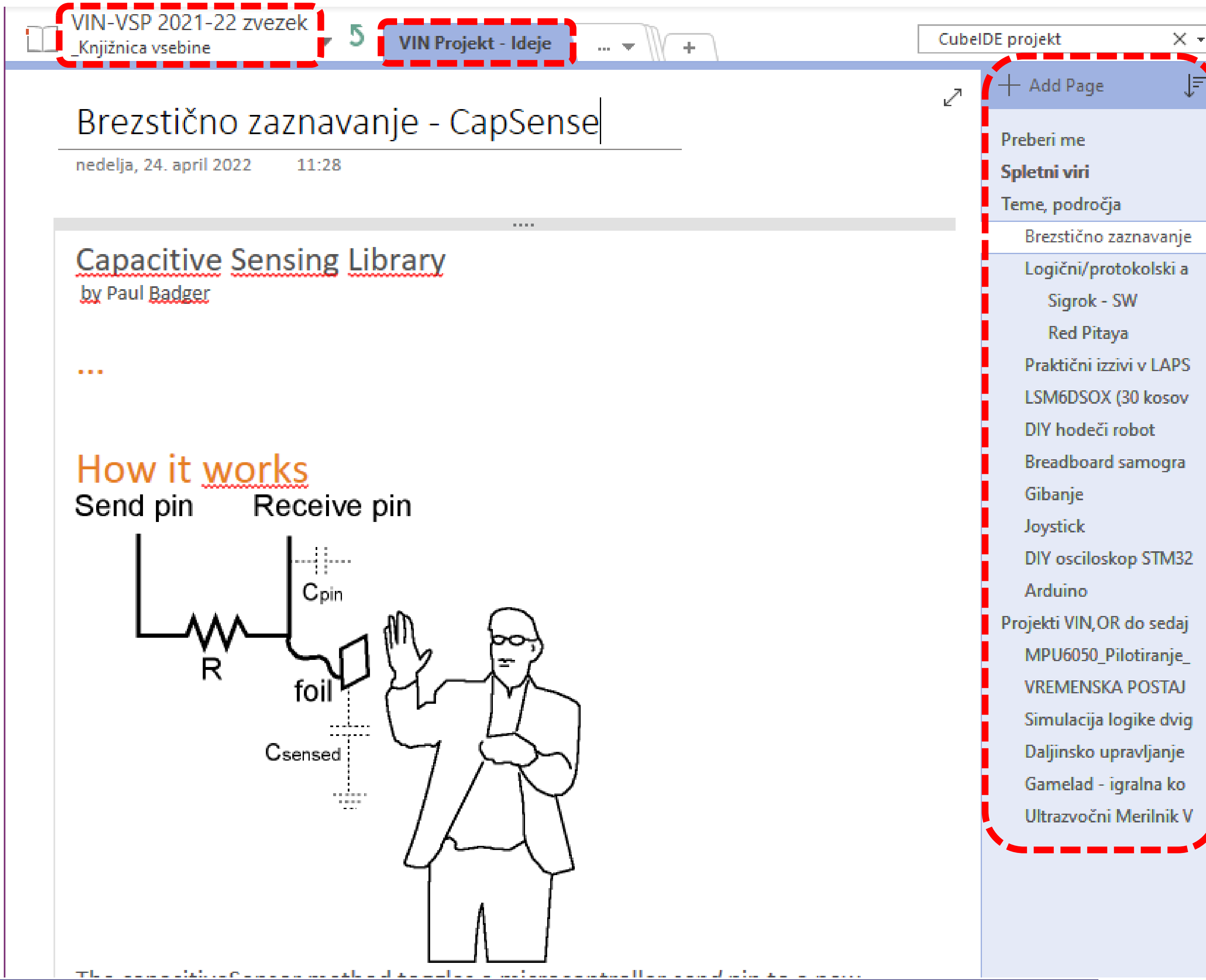
VIN projekt

Spisek opreme



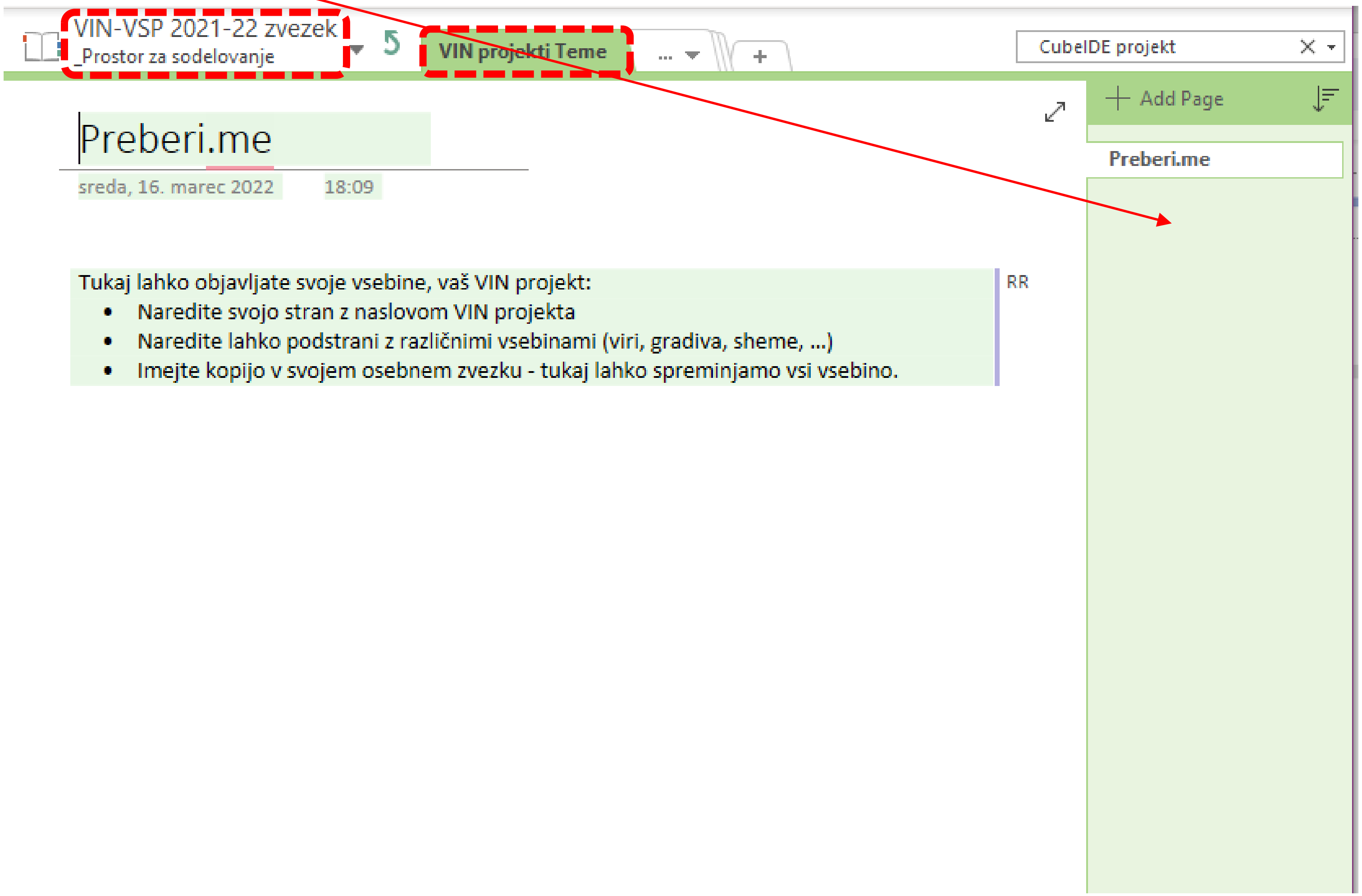
VIN projekt

Ideje



VIN projekt

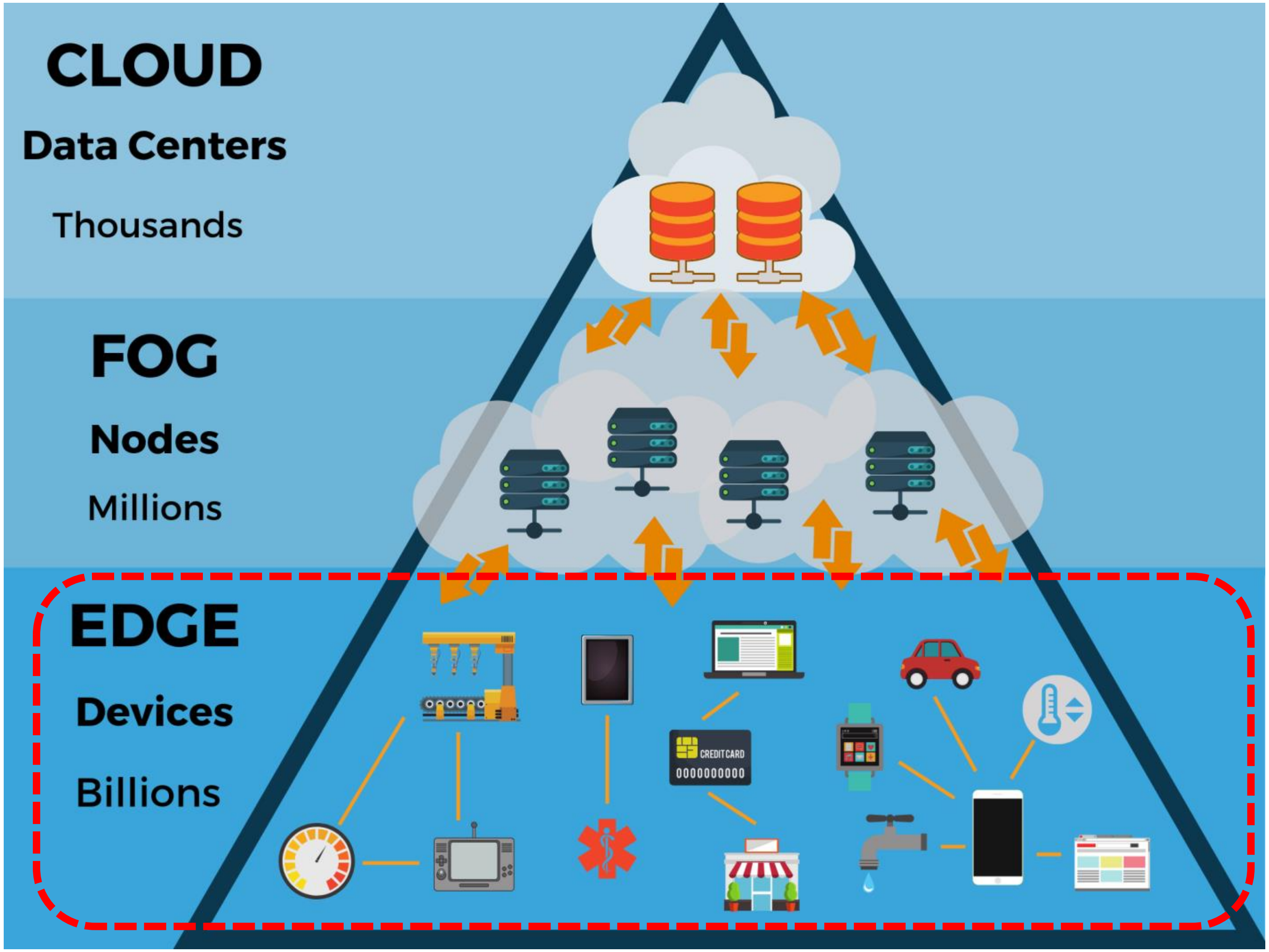
Vaša tema ?



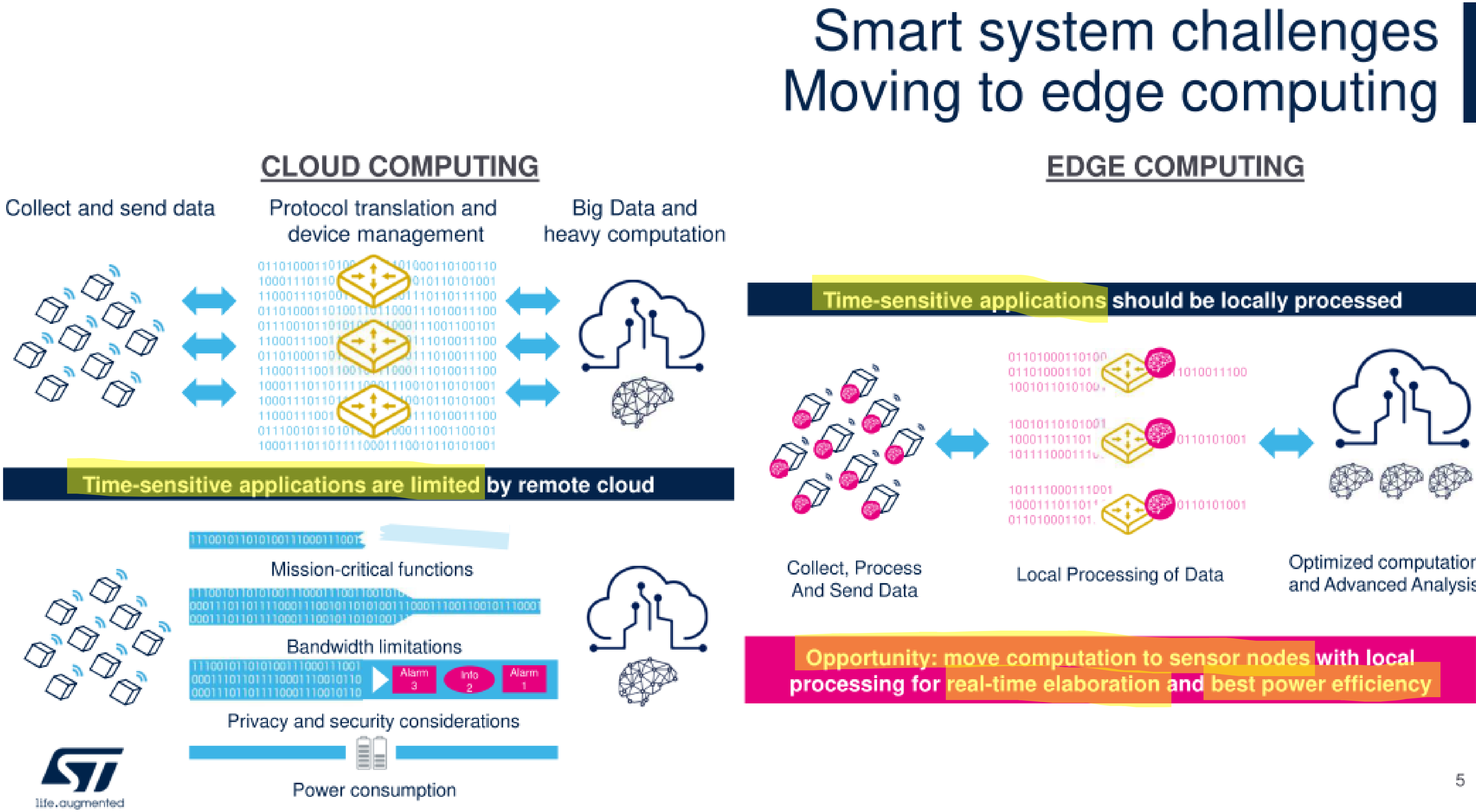
VIN projekt - VP5: STM32-Edge computing, CubeIDE projekti, Miško3

- VIN projekt
- AI v vgrajenih napravah („Edge Computing“)
- STM32 CubeIDE – Delo s projekti
- STM32 CubeIDE, SPI in LIS3DSH
- STM32 CubeIDE, I2C in CS43L22
- Miško3 – demo projekt

Edge computing



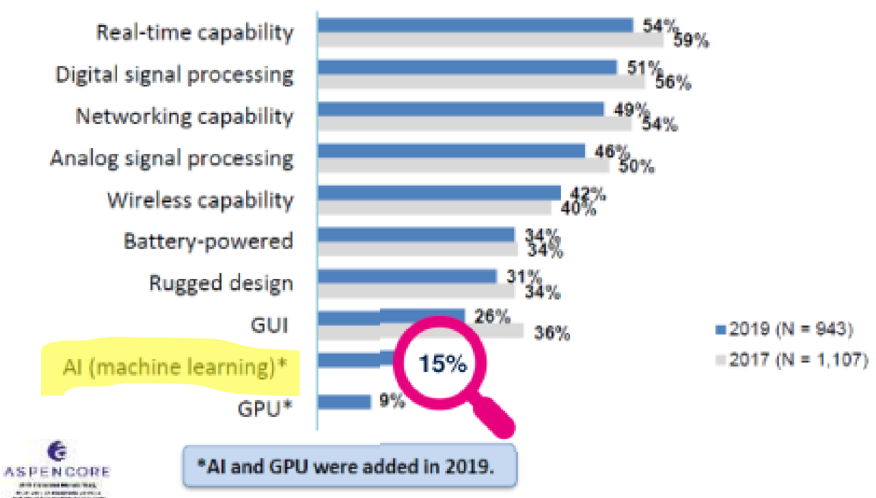
Edge computing



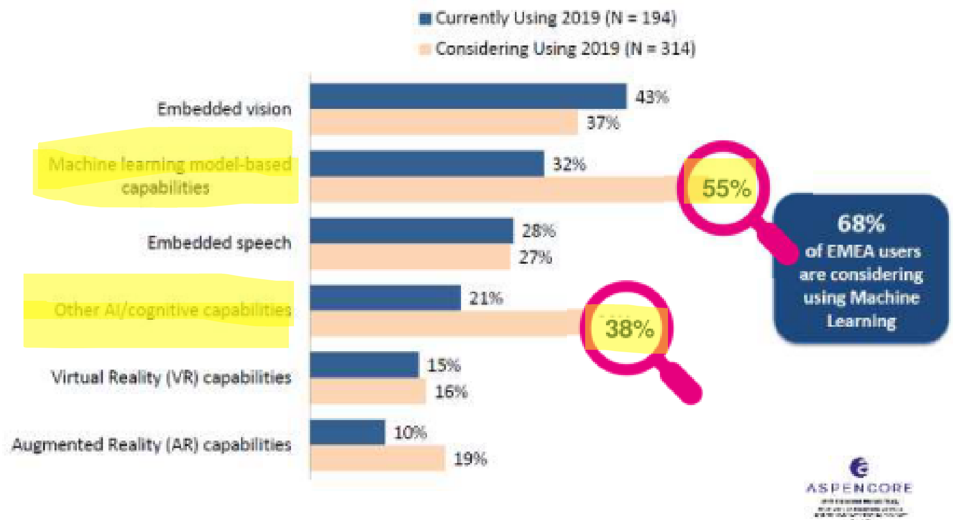
Edge computing

AI is moving to the edge

Capabilities included in a project



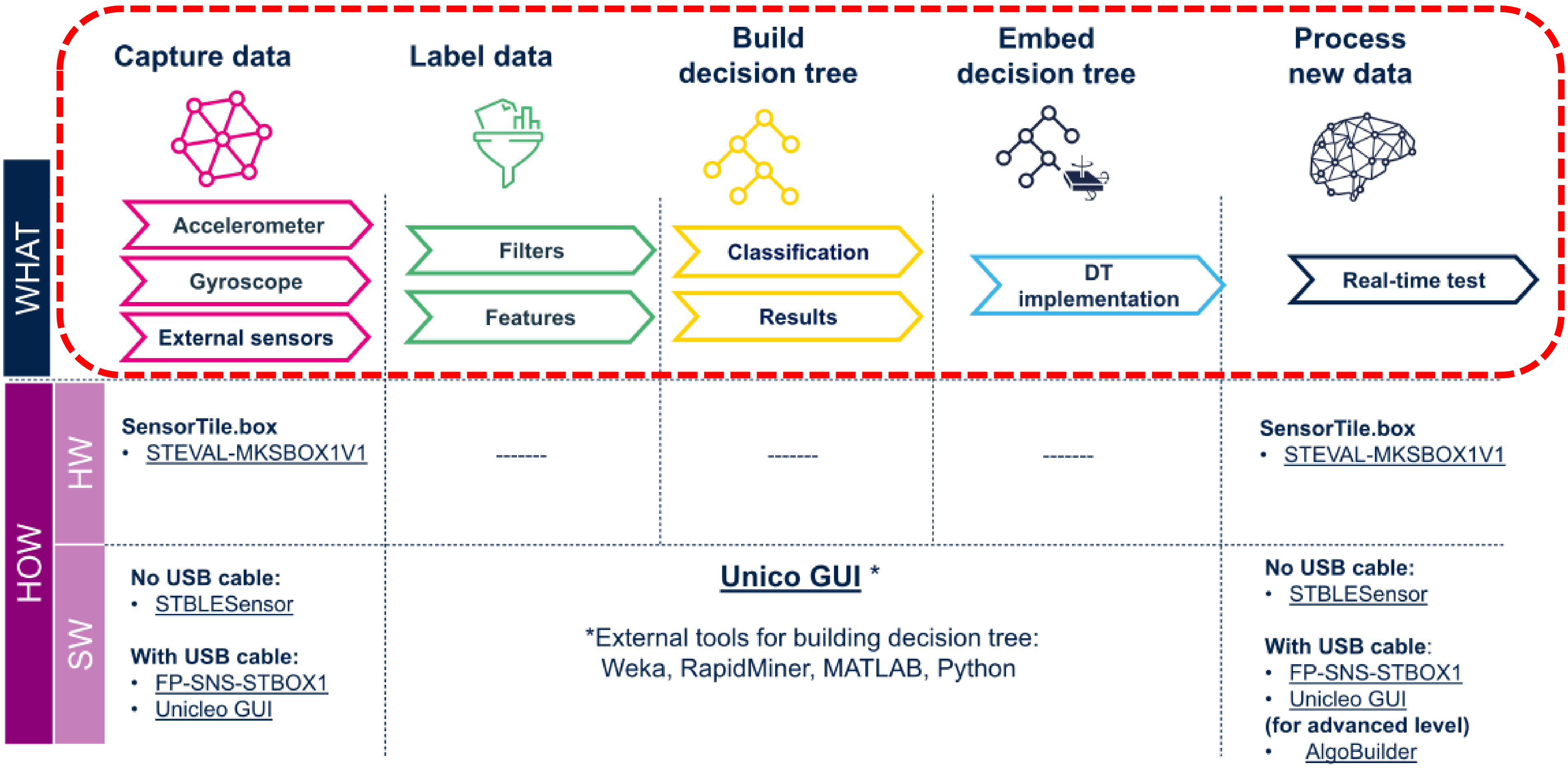
Advanced technology in a project



- **15% of embedded projects already include AI in 2019**
- **Pervasion of Machine Learning and other AI capabilities**

Edge computing – moduli, tipala

LSM6DSOX – SensorTile.box



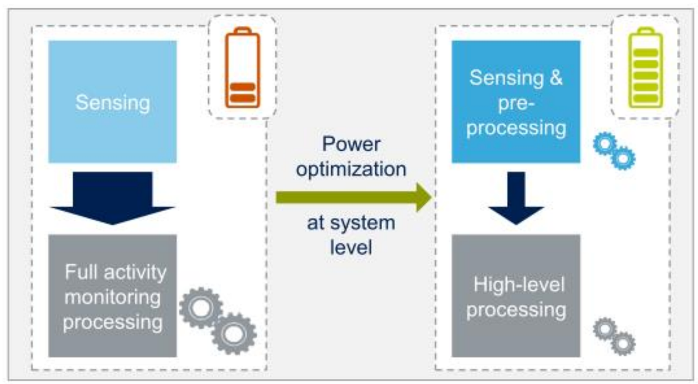
Edge computing – moduli, tipala

BHI260AP

Ultra-low power, high performance, self-learning AI smart sensor with integrated accelerometer and gyroscope

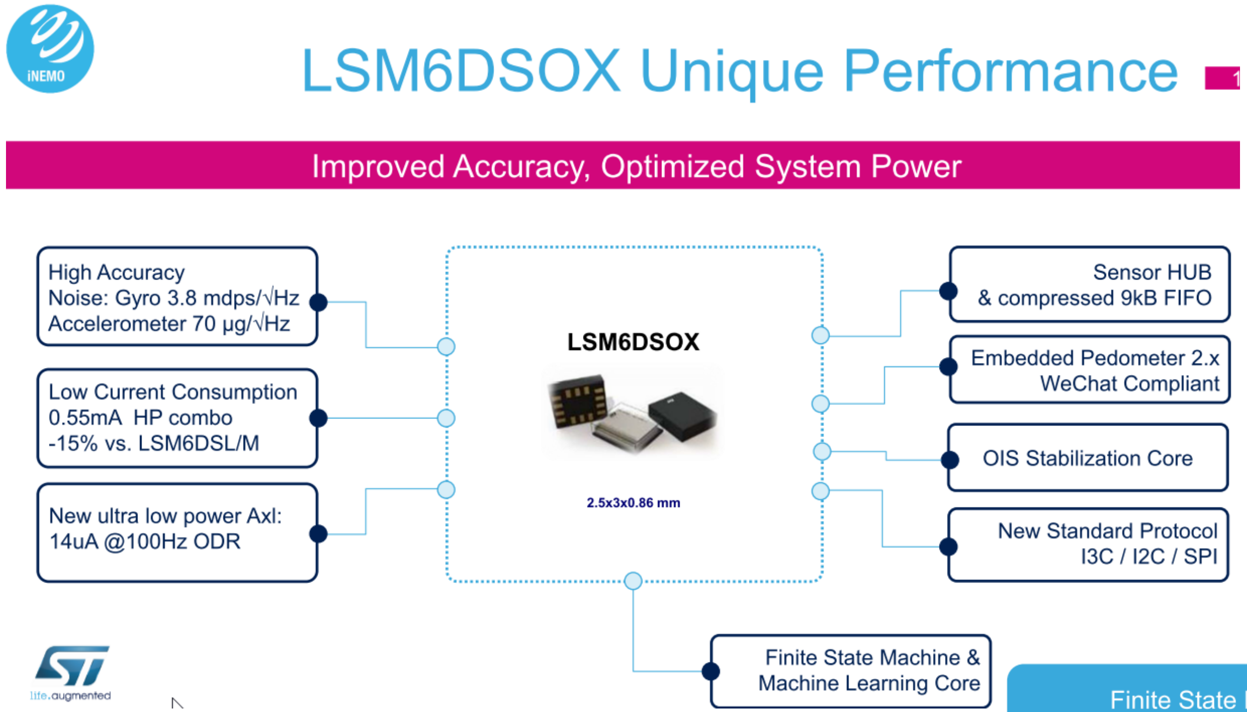
BOSCH

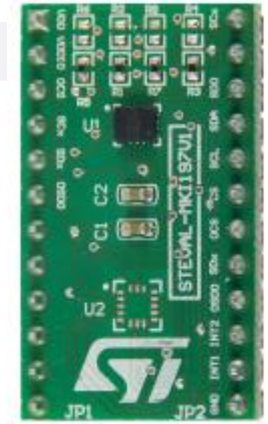
LSM6DSOX Unique Performance



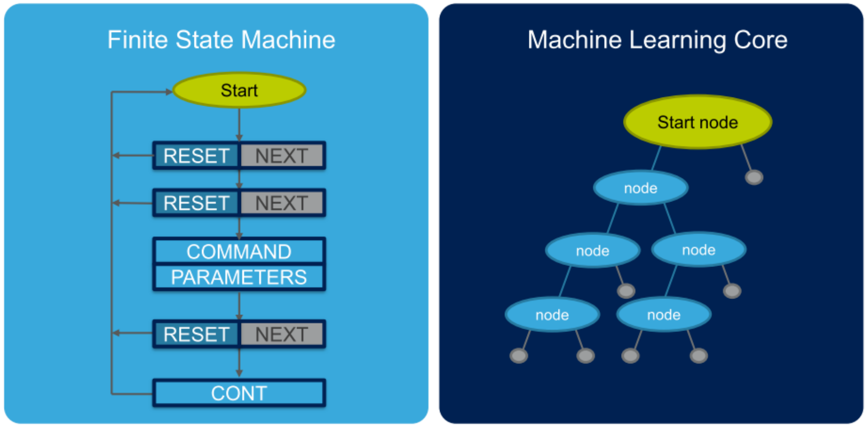
LSM6DSOX adapter board for a standard DIL24 socket

Edge computing – moduli, tipala





LSM6DSOX adapter board
for a standard DIL24 socket



FSM & MLC allows sensors to process data with reduced help of a host MCU

VIN projekt - VP5: STM32-Edge computing, CubeIDE projekti, Miško3

- VIN projekt
- AI v vgrajenih napravah („Edge Computing“)
- STM32 CubeIDE – Delo s projekti
- STM32 CubeIDE, SPI in LIS3DSH
- STM32 CubeIDE, I2C in CS43L22
- Miško3 – demo projekt

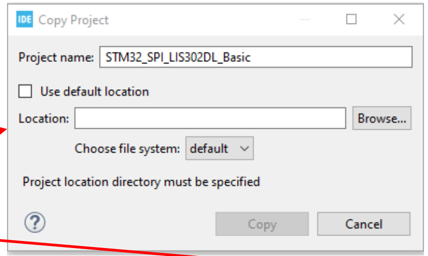
CubeIDE – delo s projekti

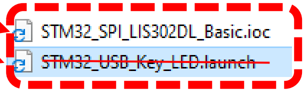
Vzpostavitev začetnega projekta :

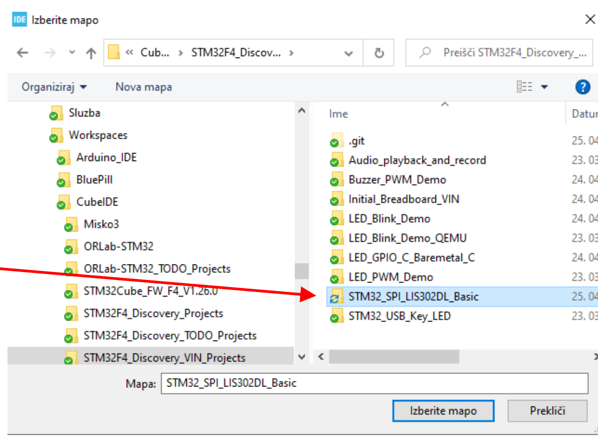
- **Kopiranje projekta Cube MX I**:
  - Znotraj CubeIDE

Kopiranje CubeIDE projekta z CubeMX .ioc datoteko

1) Edit > Copy.
2) Edit > Paste.
3) Preimenuj .ioc datoteko.
4) Zbriši Debug.launch datoteko.
5) Project > Clean.
6) Generiraj kodo s CubeMX.
7) Project > Build Project.
8) Debug As Stm32 Application.
9) Debug aplikacije.

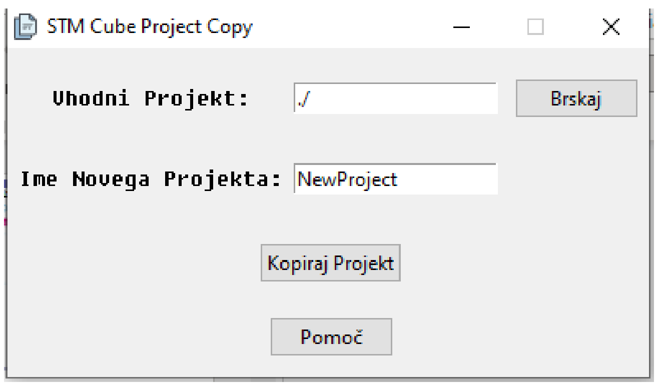






Skopiram, preimenujem ioc, generiram kodo, brišem Debug.launch, clean in build

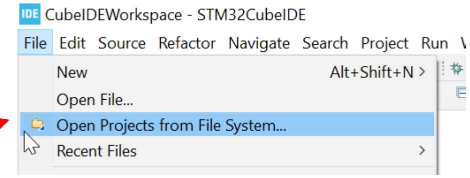
- **Kopiranje projekta Cube MX II**:
  - Uporaba orodja



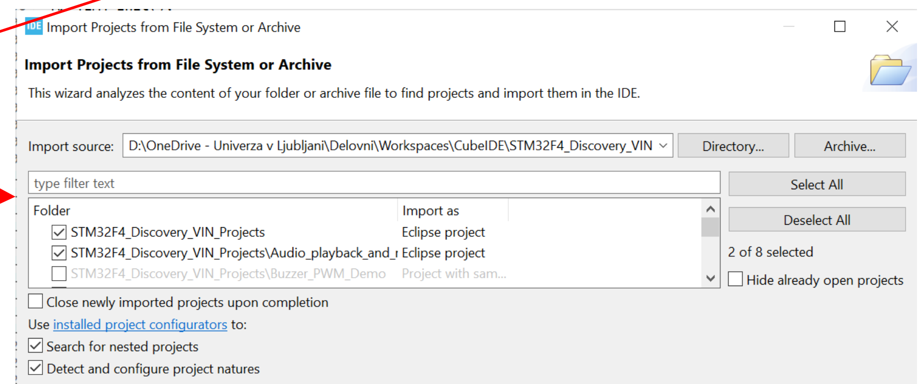
https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples/tree/main/CubeIDE

CubeIDE – delo s projekti

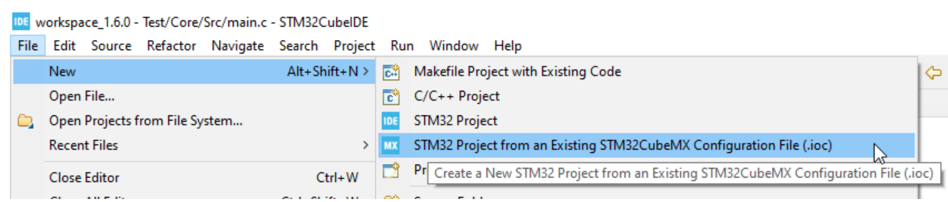
Vzpostavitev začetnega projekta :



- **Uvoz obstoječega (npr. Github)**
  - Open projects from File System
  - Select project(s)



- **Nov projekt CubeMX z .ioc datoteko**
  - .ioc datoteka vsebuje tudi vezave
  - New -> STM32 Project from… (.ioc)
  - Miško3 ima samo .ioc datoteko

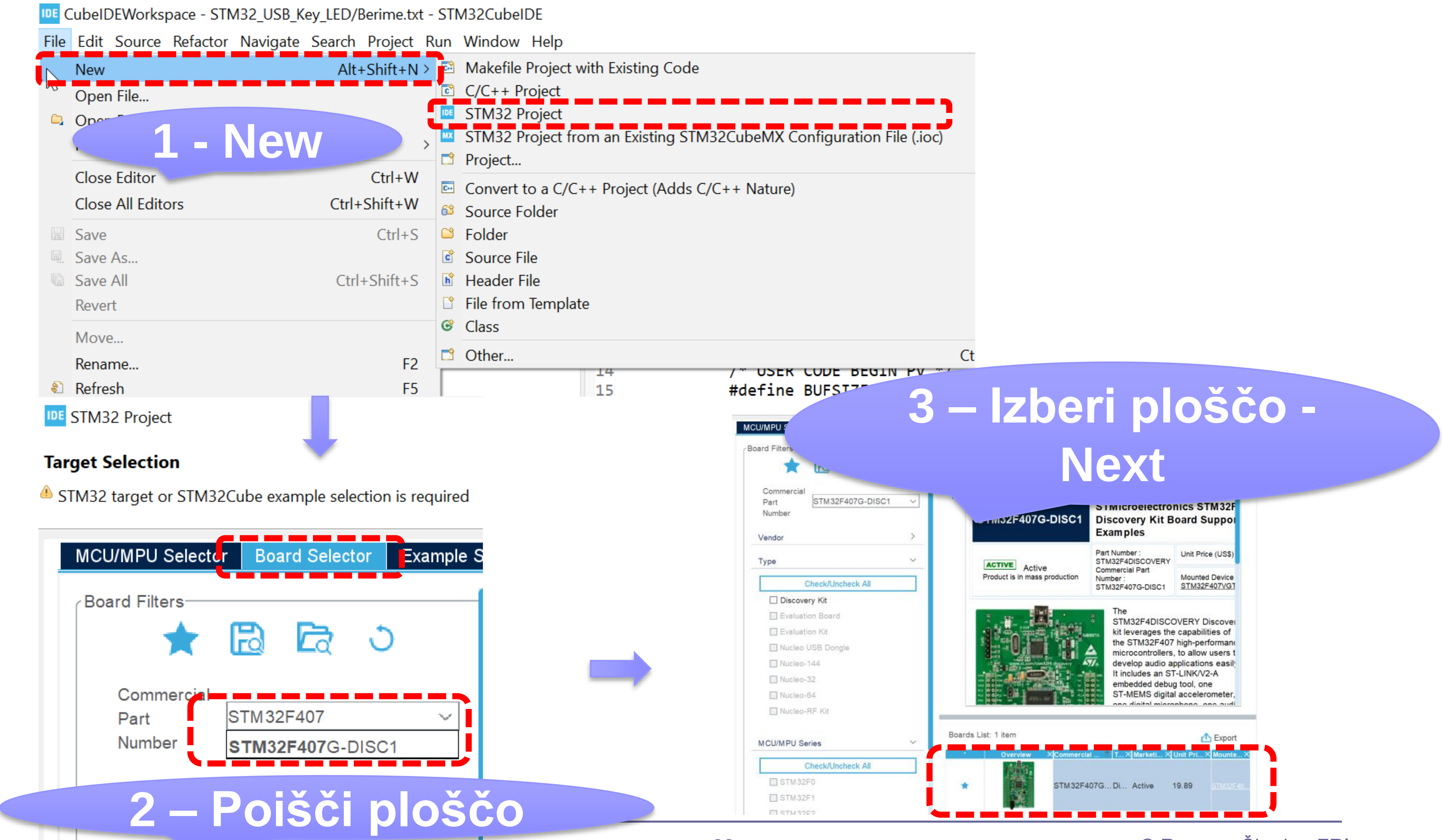


- **Nov projekt CubeMX ->**
  - New -> STM32 Project

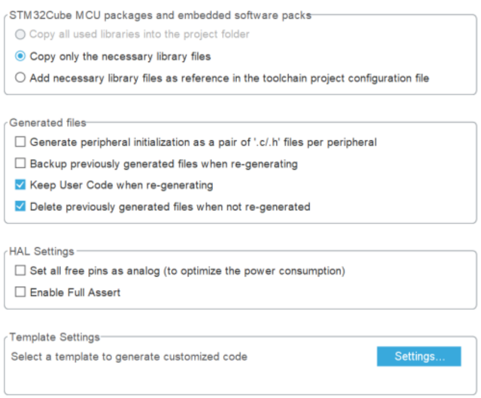
  (že naredili – sledi povzetek)

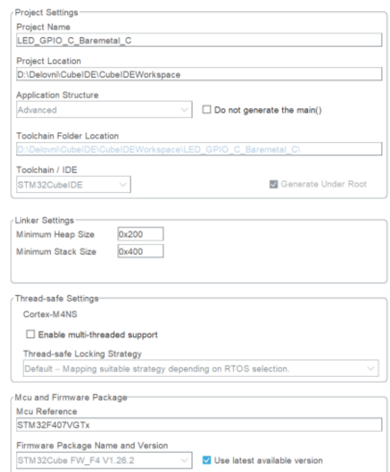
CubeIDE – Vzpostavitev novega projekta

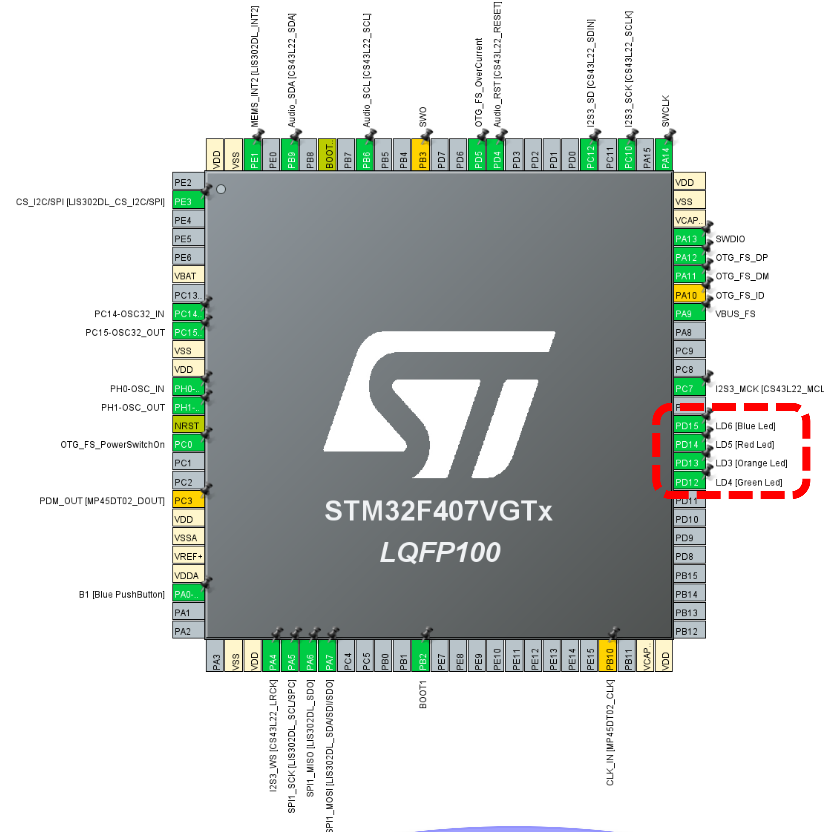
Nov projekt :

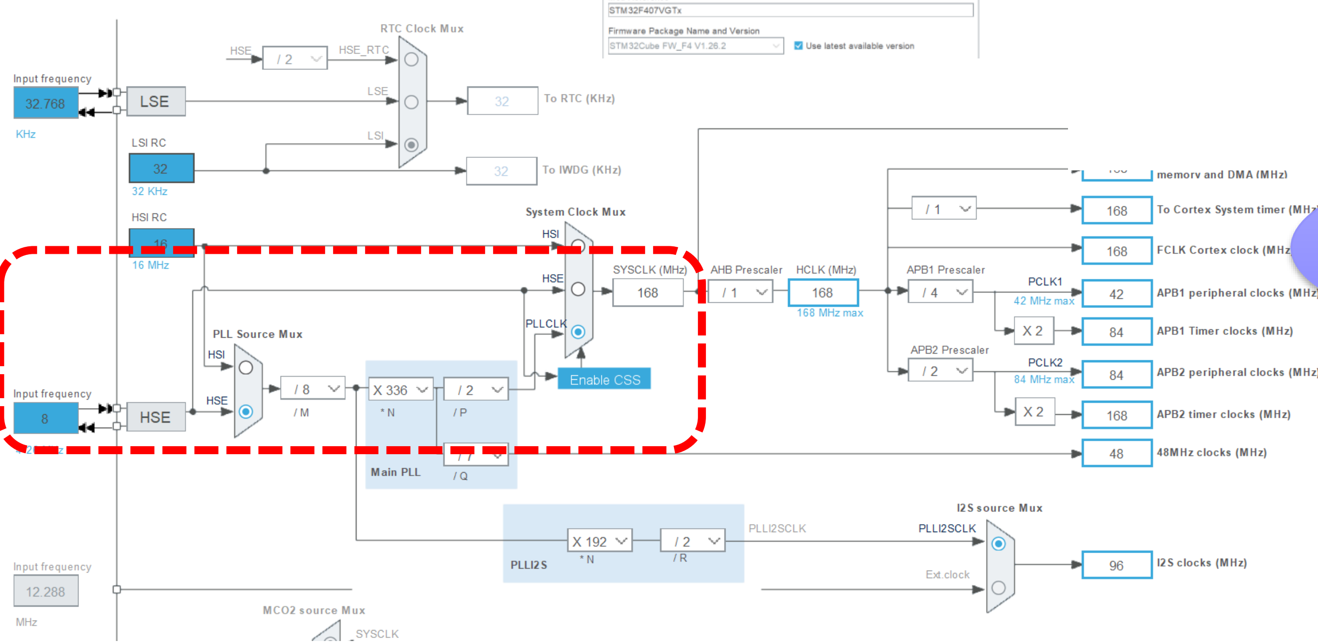


Konfiguracija : priključki, knjižnice



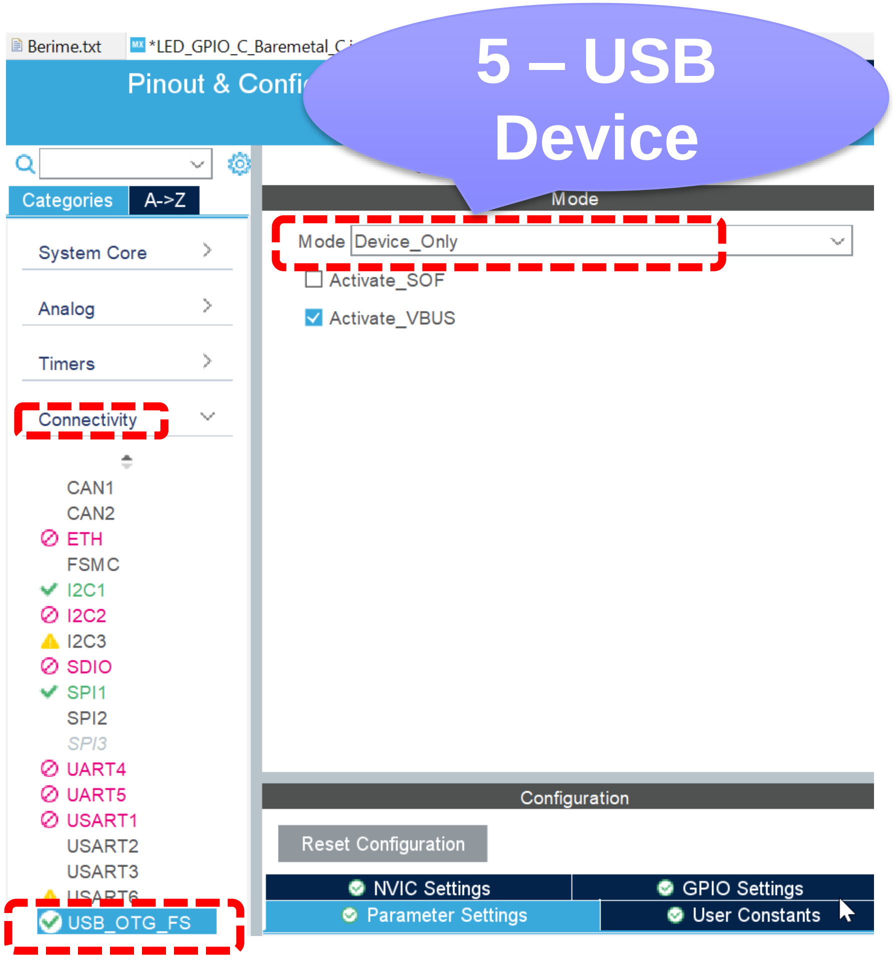


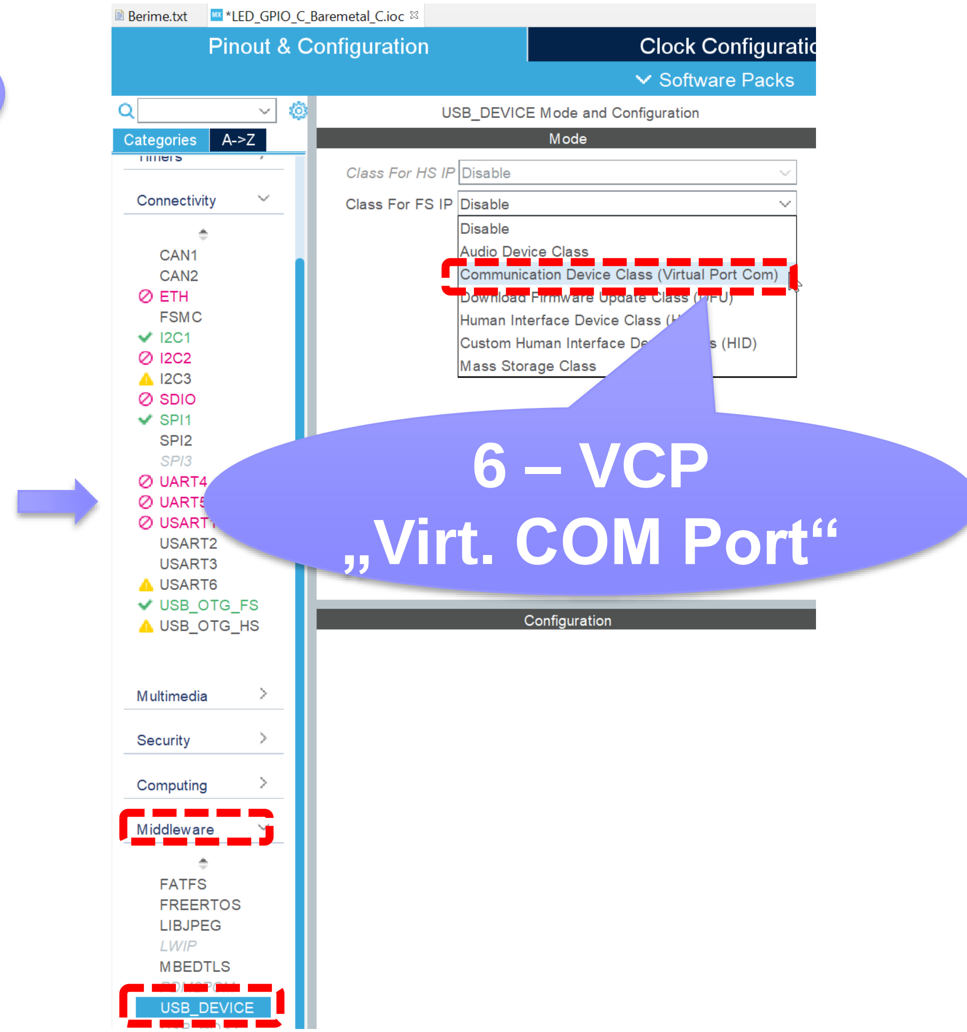




4 – Preveri nastavitve

Konfiguracija : USB Device,CDC Class = Virtual COM Port



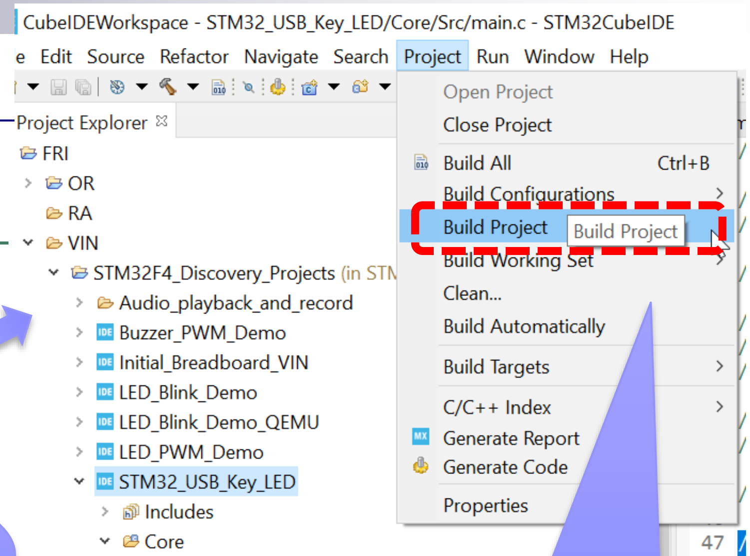


Osnovni projekt CubeIDE – USB Virtual COM Port

Program : za pošiljanje po USB Virtual COM Port

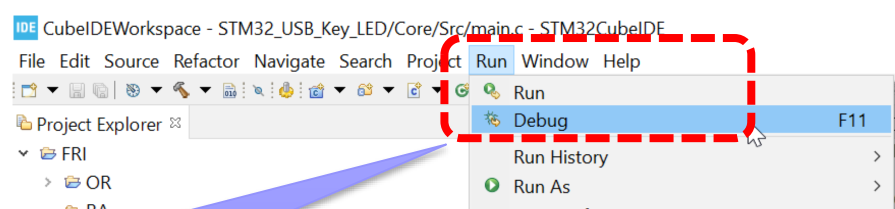
```c
/* Private variables ---------------------------------

/* USER CODE BEGIN PV */
#define     BUFSIZE 256
char        SendBuffer[BUFSIZE];
int         Counter;
/* USER CODE END PV */
```

7 – USB VCP koda



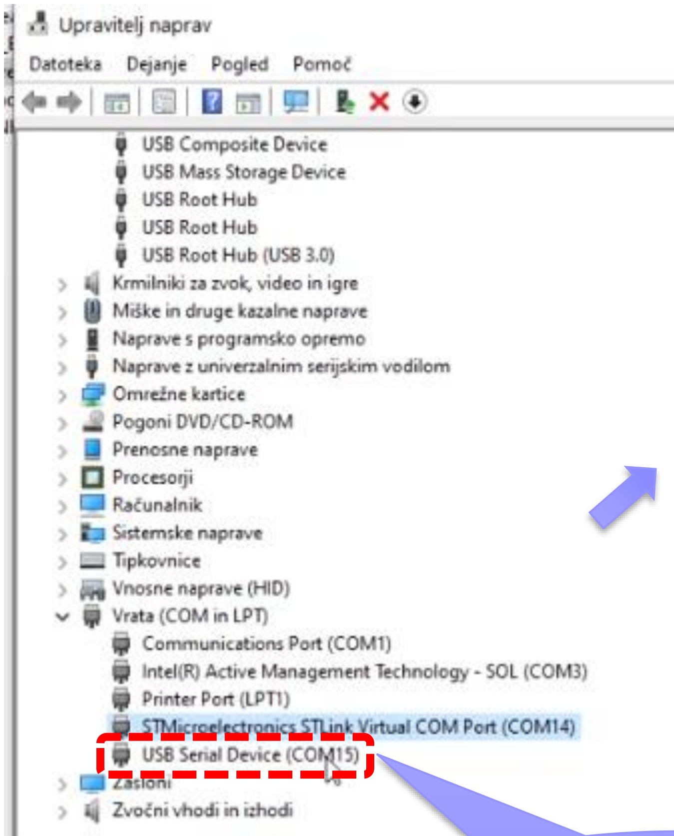
8 – Build project

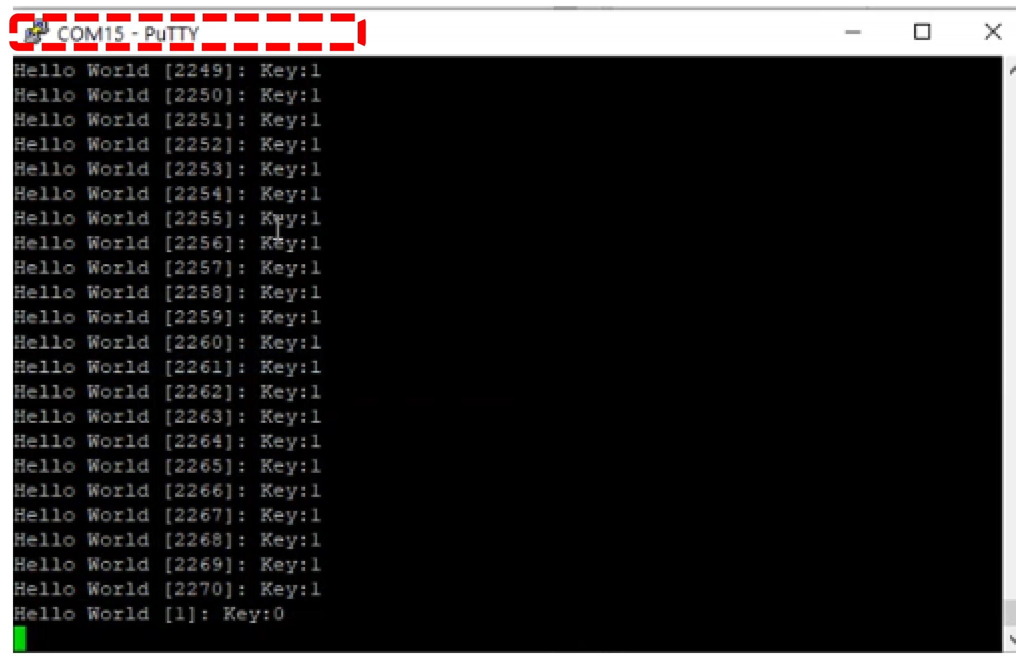
```c
/* Infinite loop */
 /* USER CODE BEGIN WHILE */
 while (1)
 {
         snprintf(SendBuffer,BUFSIZE,"Hello World [%d]\r\n",Counter++);
         CDC_Transmit_FS(SendBuffer,strlen(SendBuffer));
 /* USER CODE END WHILE */

 /* USER CODE BEGIN 3 */
         HAL_Delay(1000);
 }
 /* USER CODE END 3 */
```



9 – Debug project

Program : sprejem na PC strani (povezava z Micro-USB kablom)



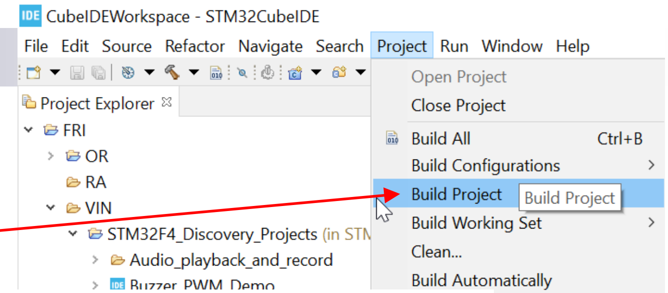


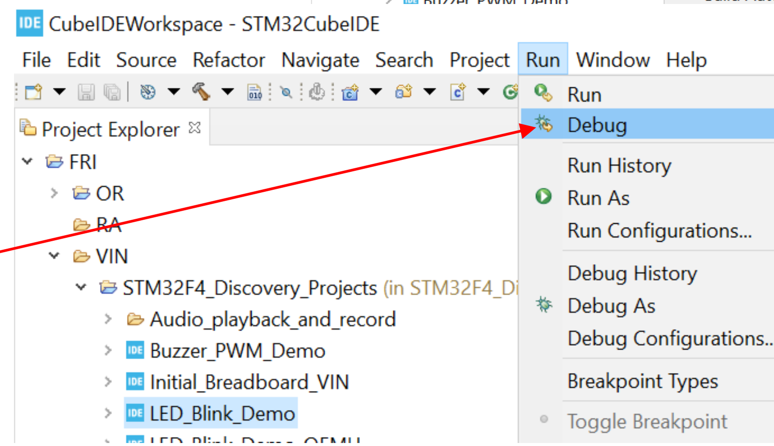
10 – Test project

CubeIDE – Zagon, debug

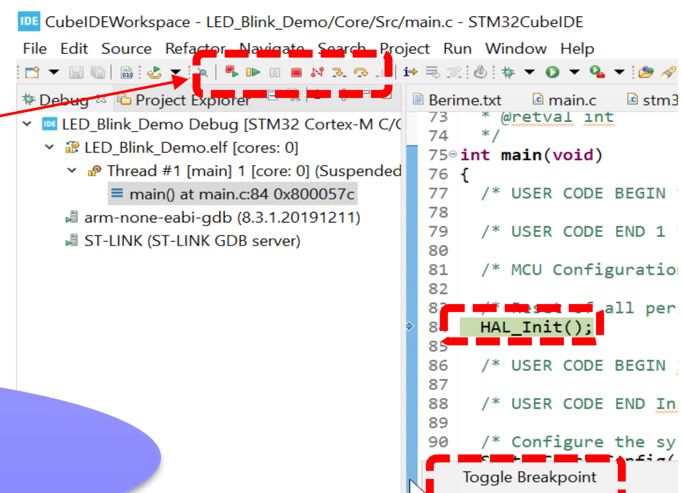
Prevajanje, zagon :

- Project -> Build Project





- Run -> Debug



- Step (Into,Over), Breakpoints

11 – Build <->Debug project, …

Osnovni projekt CubeIDE – GPIO – nivoji programiranja

Baremetal - zbirnik

```
INIT_IO:
  push {r5, r6, lr}
    // Enable GPIOD Peripheral Clock (bit 3 in AHB1ENR register)
    ldr r6, = RCC_AHB1ENR       // Load peripheral clock reg address to r6
    ldr r5, [r6]                // Read its content to r5
    orr r5, 0x00000008          // Set bit 3 to enable GPIOD clock
    str r5, [r6]                // Store result in peripheral clock register

    // Make GPIOD Pin12 as output pin (bits 25:24 in MODER register)
    ldr r6, =GPIOD_BASE       // Load GPIOD BASE address to r6
    ldr r5, [r6,#GPIOD_MODER]  // Read GPIOD_MODER content to r5
    and r5, 0x00FFFFFF          // Clear bits 31-24 for P12-15
    orr r5, 0x55000000          // Write 01 to bits 31-24 for P12-15
    str r5, [r6]                // Store result in GPIOD MODER register
  pop {r5, r6, pc}


LED_ON:
    push {r5, r6, lr}
    // Set GPIOD Pins to 1 (through BSSR register)
    ldr    r6, =GPIOD_BASE       // Load GPIOD BASE address to r6
    mov    r5, #LEDs_ON
    str    r5, [r6,#GPIOD_BSSR] // Write to BSRR register
    pop {r5, r6, pc}

LED_OFF:
    push {r5, r6, lr}
    // Set GPIOD Pins to 0 (through BSSR register)
    ldr    r6, =GPIOD_BASE       // Load GPIOD BASE address to r6
    mov    r5, #LEDs_OFF
    str    r5, [r6,#GPIOD_BSSR] // Write to BSRR register
    pop {r5, r6, pc}
```

https://github.com/LAPSyLAB/ORLab-STM32/tree/main/GPIO_LEDs

Baremetal - C

```
  /* USER CODE BEGIN 2 */


  RCC->AHB1ENR |= 0x08;
// Enable clock for GPIOD
  GPIOD->MODER |= 0x01000000;        //
MODE Register: bit 12 == out

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    GPIOD->ODR ^= 0x1000;             //
Toggle PD12

/* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    for (int i=0; i<0x1000000; i++) {};
// waste some time
  }
  /* USER CODE END 3 */
```

https://github.com/LAPSyLAB/STM32F4_Discovery_VIN_Projects/tree/main/LED_GPIO_C_Baremetal_C

HAL - C

```
/* Infinite loop */
 /* USER CODE BEGIN WHILE */
 while (1)
 {

HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12);

  /* USER CODE END WHILE */

   /* USER CODE BEGIN 3 */
 HAL_Delay(1000);
 }
 /* USER CODE END 3 */



void HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx,
uint16_t GPIO_Pin)
{
  uint32_t odr;

  /* Check the parameters */
  assert_param(IS_GPIO_PIN(GPIO_Pin));

  /* get current Ouput Data Register value
*/
  odr = GPIOx->ODR;

  /* Set selected pins that were at low
level, and reset ones that were high */
  GPIOx->BSRR = ((odr & GPIO_Pin) <<
GPIO_NUMBER) | (~odr & GPIO_Pin);
}
```

https://github.com/LAPSyLAB/STM32F4_Discovery_VIN_Projects/tree/main/LED_Blink_Demo

VIN projekt - VP5: STM32-Edge computing, CubeIDE projekti, Miško3

- VIN projekt
- AI v vgrajenih napravah („Edge Computing“)
- STM32 CubeIDE – Delo s projekti
- STM32 CubeIDE, SPI in LIS3DSH
- STM32 CubeIDE, I2C in CS43L22
- Miško3 – demo projekt

5 Digital main blocks

5.1 State machine

The LIS3DSH embeds two state machines able to run a user defined program.

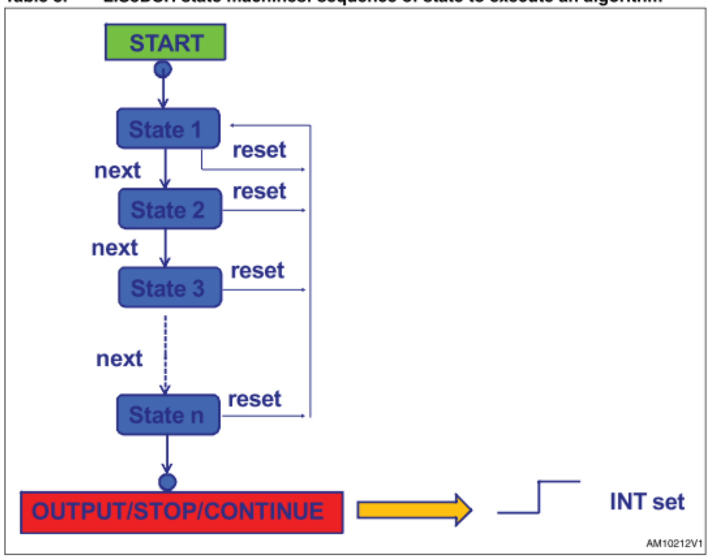
The program is made up of a set of instructions that define the transition to successive states. Conditional branches are possible.

From each state (n) it is possible to have transition to the next state (n+1) or to reset state. Transition to reset point happens when "RESET condition" is true; Transition to the next step happens when "NEXT condition" is true.

Interrupt is triggered when output/stop/continue state is reached.

Each state machine allows to implement gesture recognition in a flexible way, free-fall, wake-up, 4D/6D orientation, pulse counter and step recognition, click/double click, shake/double shake, face-up/face-down, turn/double turn:

Table 8. LIS3DSH state machines: sequence of state to execute an algorithm



SPI - serial peripheral interface

Subject to general operating conditions for Vdd and Top.

SPI slave timing values

ol	Parameter	Value (1) Min.	Value (1) Max.	Unit
:)	SPI clock cycle	100		ns
:)	SPI clock frequency		10	MHz
:)	CS setup time	6		

I²C - inter IC control interface

Subject to general operating conditions for Vdd and Top.

I²C slave timing values

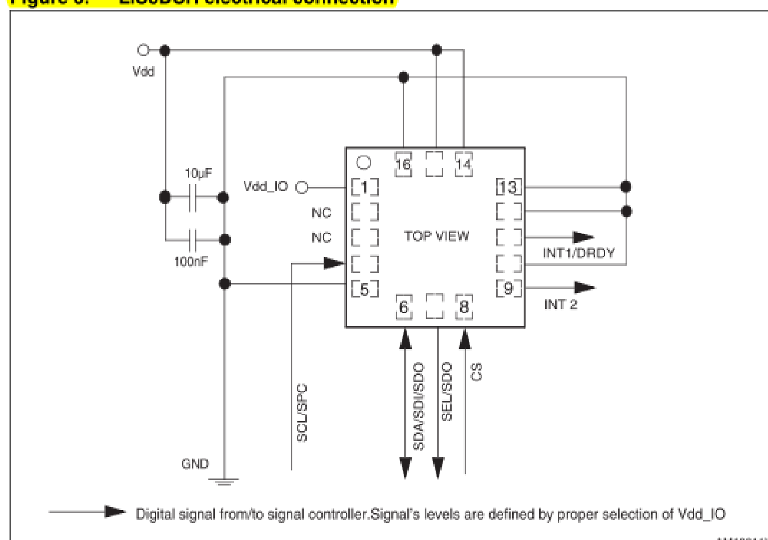
Parameter	I²C standard mode (1) Min.	I²C standard mode (1) Max.	I²C fast mode (1) Min.	I²C fast mode (1) Max.	Unit
SCL clock frequency	0	100	0	400	kHz

Table 7. Absolute maximum ratings

Symbol	Ratings	Maximum value	Unit
Vdd	Supply voltage	-0.3 to 4.8	V

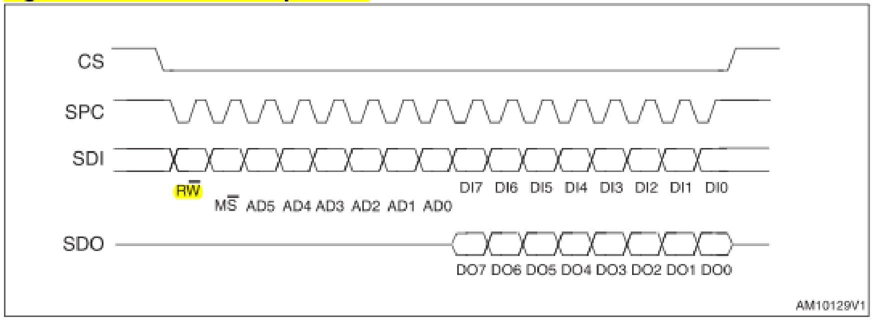
Application hints

Figure 5. LIS3DSH electrical connection



https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples/blob/main/STM32F407_Discovery_kit/LIS3DSH.pdf

Figure 6. Read and write protocol



bit 0: RW bit. When 0, the data DI(7:0) is written into the device. When 1, the data DO(7:0) from the device is read. In the latter case, the chip drives **SDO** at the start of bit 8.

bit 1-7: address AD(6:0). This is the address field of the indexed register.

bit 8-15: data DI(7:0) (write mode). This is the data that is written into the device (MSb first).

bit 8-15: data DO(7:0) (read mode). This is the data that is read from the device (MSb first).

SPI - serial peripheral interface

Subject to general operating conditions for Vdd and Top.

SPI slave timing values

...ol	Parameter	Value (1) Min.	Max.	Unit
...)	SPI clock cycle	100		ns
...)	SPI clock frequency		10	MHz

Table 7. Absolute maximum ratings

Symbol	Ratings	Maximum value	Unit
Vdd	Supply voltage	-0.3 to 4.8	V

7 Register mapping

Table 16 provides a list of the 8/16-bit registers embedded in the device and the related address:

Table 16. Register address map

Name	Type	Register address Hex	Binary	Default	Comment
INFO1	r	0D	00001101	0010 0001	Information register 1
INFO2	r	0E	00001110	0000 0000	Information register 2
WHO_AM_I	r	OF	00001111	0011 1111	Who I am ID
OUT_X_L	r	28	00101000	0000 0000	Output registers
OUT_X_H	r	29	00101001		
OUT_Y_L	r	2A	00101010		
OUT_Y_H	r	2B	00101011		
OUT_Z_L	r	2C	00101100		
OUT_Z_H	r	2D	00101101		

8.3 WHO_AM_I (0Fh)

Who_AM_I register.

rozman 26. 04. 2022, 0..
0x3F

Table 19. WHO_AM_I register default value

0	0	1	1	1	1	1	1

8.5 CTRL_REG4 (20h)

Control register 4.

rozman 26. 04. 2022, 0..
0x47 (25Hz, all axes on)

Table 22. Control register 4

ODR3	ODR2	ODR1	ODR0	BDU	ZEN	YEN	XEN

8.23 OUT_X (28h - 29h)

X-axis output register.

Table 49. OUT_X_L register default value

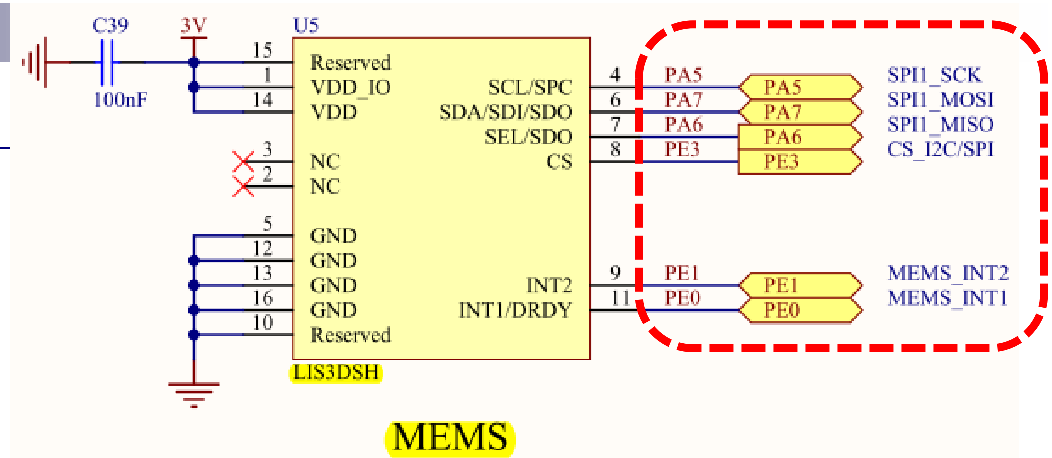
0	0	0	0	0	0	0	0

Table 50. OUT_X_H register default value

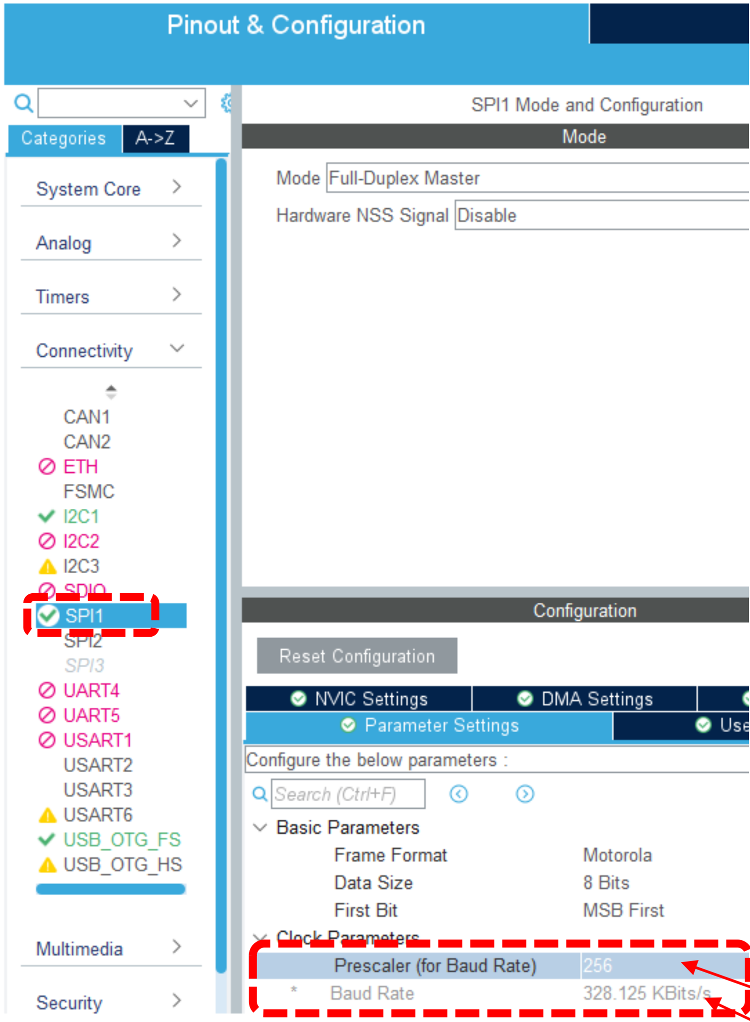
0	0	0	0	0	0	0	0

https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples/blob/main/STM32F407_Discovery_kit/LIS3DSH.pdf

VP 5 - STM32 CubeIDE, SPI in LIS3DSH



CubeMX nastavitve :



spi.c:

```c
/* USER CODE END SPI1_Init 1 */
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_256;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 10;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN SPI1_Init 2 */
```

Spremenimo iz 2 v 256
(počasnejša komunikacija)

main.c : dodana koda

Spremenljivke

```
/* USER CODE BEGIN PV */
#define BUFSIZE 256
charSendBuffer[BUFSIZE];
intCounter;
int KeyState=0;

// Global variables
uint8_t indata[2];
uint8_t outdata[2] = {0,0};
uint8_t lis_id;
int8_t AccelX;
int8_t AccelY;
int8_t AccelZ;

HAL_StatusTypeDef SPIStatus;

/* USER CODE END PV */
```

Inicializacija

```
/* USER CODE BEGIN 2 */

  // Config accelerometer
  // Read WHOAMI register
  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET);
  outdata[0] = 0x0f | 0x80 ;  // read whoami
  HAL_SPI_TransmitReceive(&hspi1, &outdata, &indata, 2, HAL_MAX_DELAY);
  lis_id = indata[1];
  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET);

  HAL_Delay(500);

  // Set CTRL register 0x47 -> [0x20]
  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET);
  outdata[0] = 0x20 ;  // switch on axes
  outdata[1] = 0x47 ;
  HAL_SPI_TransmitReceive(&hspi1, &outdata, &indata, 2, HAL_MAX_DELAY);
  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET);

  HAL_Delay(500);
  outdata[1] = 0x00 ;

  /* USER CODE END 2 */
```

Glavna zanka

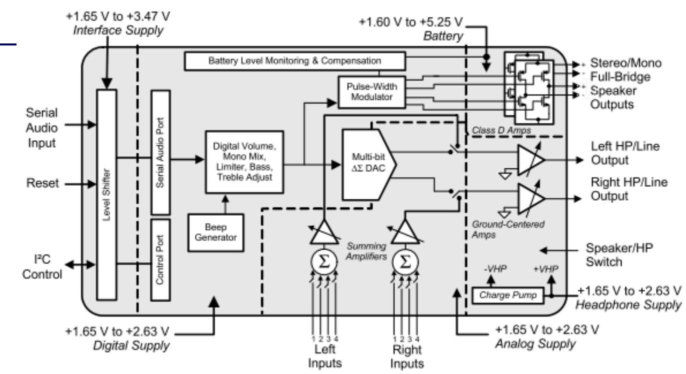
```
 /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {

  outdata[0] = 0x29 | 0x80  ;  // read x
  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET);
  HAL_SPI_TransmitReceive(&hspi1, &outdata, &indata, 2, HAL_MAX_DELAY);
// HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET);
  AccelX = indata[1];

  outdata[0] = 0x2B | 0x80  ;  // read y
//  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET);
  HAL_SPI_TransmitReceive(&hspi1, &outdata, &indata, 2, HAL_MAX_DELAY);
// HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET);
  AccelY = indata[1];

  outdata[0] = 0x2D | 0x80  ;  // read z
//  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET);
  HAL_SPI_TransmitReceive(&hspi1, &outdata, &indata, 2, HAL_MAX_DELAY);
  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET);
  AccelZ = indata[1];

 HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12);
 HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_13);
 HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_14);

 KeyState = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0);
 HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, KeyState);


  snprintf(SendBuffer,BUFSIZE,"Hello World [%d]: Key:%04d Accel[ID:%02x]
X:%04d Y:%d Z:%04d\r\n",Counter++,KeyState,lis_id,AccelX,AccelY,AccelZ);
  CDC_Transmit_FS(SendBuffer,strlen(SendBuffer));

    /* USER CODE END WHILE */
```

VIN projekt - VP5: STM32-Edge computing, CubeIDE projekti, Miško3

- VIN projekt
- AI v vgrajenih napravah („Edge Computing“)
- STM32 CubeIDE – Delo s projekti
- STM32 CubeIDE, SPI in LIS3DSH
- STM32 CubeIDE, I2C in CS43L22
- Miško3 – demo projekt

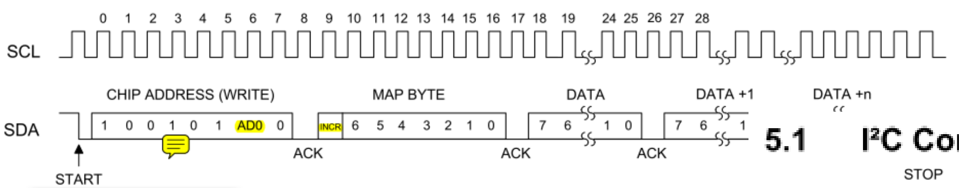
CIRRUS LOGIC® **CS43L22**

Low Power, Stereo DAC w/Headphone & Speaker Amps



5.1 I²C Control

The upper 6 bits of the address field are fixed at 100101. To communicate with the CS43L22, the chip address field, which is the first byte sent to the CS43L22, should match 100101 followed by the setting of the AD0 pin. The eighth bit of the address is the R/W bit. If the operation is a write, the next byte is the Memory Address Pointer (MAP), which selects the register to be read or written. If the operation is a read, the contents of the register pointed to by the MAP will be output. Setting the auto-increment bit in MAP allows successive reads or writes of consecutive registers. Each byte is separated by an acknowledge bit. The ACK bit is output from the CS43L22 after each input byte is read and is input to the CS43L22 from the microcontroller after each transmitted byte.



5.1 I²C Control

rozman 26. 04. 2022, 1...

AD0 -> GND Addr=0x94

Figure 16. Control Port Timing, I²C Write

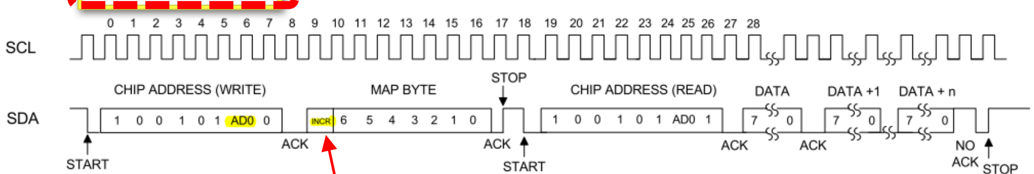


Figure 17. Control Port Timing, I²C Read

5.1.1 Memory Address Pointer (MAP)

The MAP byte comes after the address byte and selects the register to be read or written. Refer to the pseudo code above for implementation details.

5.1.1.1 Map Increment (INCR)

The device has MAP auto-increment capability enabled by the INCR bit (the MSB) of the MAP. If INCR is set to 0, MAP will stay constant for successive I²C writes or reads. If INCR is set to 1, MAP will auto-increment after each byte is read or written, allowing block reads or writes of successive registers.

7. REGISTER DESCRIPTION

All registers are read/write except for the chip I.D. and Revision Register and Interrupt Status Register which are read only. See the following bit definition tables for bit assignment information. The default state of each bit after a power-up sequence or reset is shown as shaded in the table. Unless otherwise specified, all "Reserved" bits must maintain their default value.

7.1 Chip I.D. and Revision Register (Address 01h) *(Read Only)*

7	6	5	4	3	2	1	0
CHIPID4	CHIPID3	CHIPID2	CHIPID1	CHIPID0	REVID2	REVID1	REVID0

7.1.1 *Chip I.D. (Read Only)*

I.D. code for the CS43L22.

CHIPID[4:0]	Device
11100	CS43L22

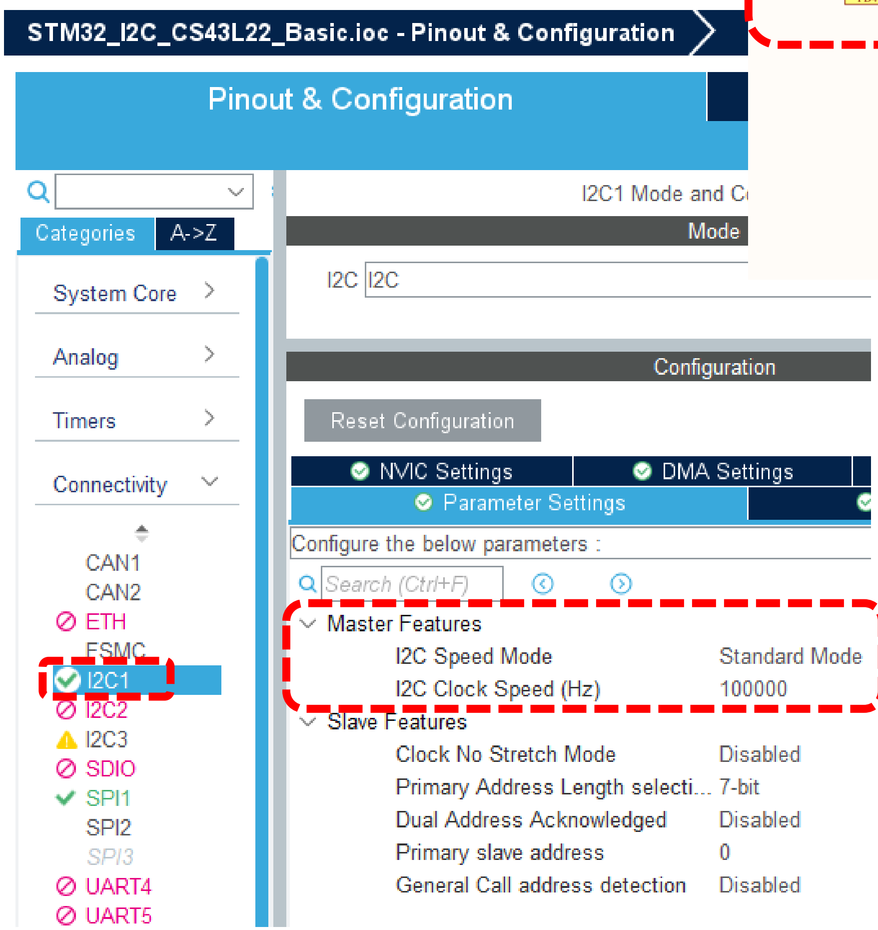
7.1.2 *Chip Revision (Read Only)*

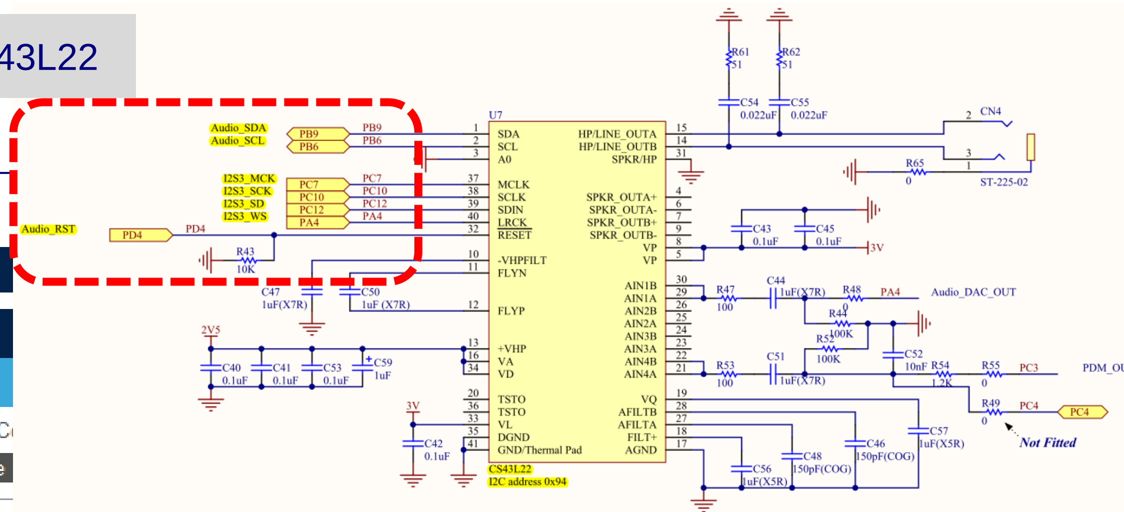
CS43L22 revision level.

REVID[2:0]	Revision Level
000	A0
001	A1
010	B0
011	B1

https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples/blob/main/STM32F407_Discovery_kit/CS43L22_F2.pdf

CubeMX nastavitve
(I2C1 že nastavljena)





i2c.c:

```c
/* I2C1 init function */
void MX_I2C1_Init(void)
{
  /* USER CODE BEGIN I2C1_Init 0 */
  /* USER CODE END I2C1_Init 0 */

  /* USER CODE BEGIN I2C1_Init 1 */
  /* USER CODE END I2C1_Init 1 */

  hi2c1.Instance = I2C1;
  hi2c1.Init.ClockSpeed = 100000;
  hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
  hi2c1.Init.OwnAddress1 = 0;
  hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c1.Init.OwnAddress2 = 0;
  hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN I2C1_Init 2 */

  /* USER CODE END I2C1_Init 2 */

}
```

VP 5 - STM32 CubeIDE, I2C in CS43L22

main.c : dodana koda

Spremenljivke

```c
/* USER CODE BEGIN PV */
#define BUFSIZE 256
charSendBuffer[BUFSIZE];
intCounter;
int KeyState=0;

HAL_StatusTypeDef retval;
uint8_t ChipID;
/* USER CODE END PV */
```

Inicializacija

```c
/* USER CODE BEGIN 2 */

HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4,GPIO_PIN_SET);    // Set Reset line to 1 (switch device on)

HAL_Delay(1000);    // recomended by datasheet

// From Device with address=0x94, Read register with address 0x01 and put value in ChipID
// DevAddress_0x94, tMemAddress=0x01, MemAddSize=8b, *pData,Size, Timeout);
retval = HAL_I2C_Mem_Read(&hi2c1, 0x94, 0x01, I2C_MEMADD_SIZE_8BIT, &ChipID, 1, 1000);

/* USER CODE END 2 */
```

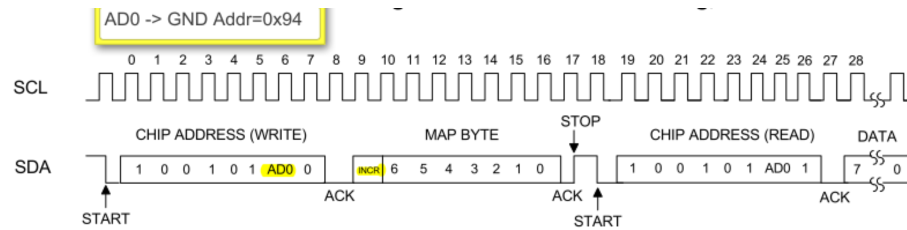


Figure 17. Control Port Timing, I²C Read

Glavna zanka

```c
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {

 HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12);
 HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_13);
 HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_14);

 KeyState = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0);
 HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, KeyState);


  snprintf(SendBuffer,BUFSIZE,"Hello World [%d]: Key:%d | Id:%02x \r\n",Counter++,KeyState,ChipID);
  CDC_Transmit_FS(SendBuffer,strlen(SendBuffer));

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  HAL_Delay(1000);
  }
  /* USER CODE END 3 */
}
```

Primer kompleksnejše demo USB-Audio aplikacije :

"Wave player - Predvajalnik .wav datotek iz USB ključka na izhod za slušalke"

WAVEPLAYER using STM32 || I2S AUDIO || CS43L22 || F4 DISCOVERY

From <https://www.youtube.com/watch?v=_Pm0L1ropJs>

AN3997 Application note Audio playback and recording using the STM32F4DISCOVERY

https://www.st.com/resource/en/application_note/an3997-audio-playback-and-recording-using-the-stm32f4discovery-stmicroelectronics.pdf

WavePlayer using STM32 Discovery

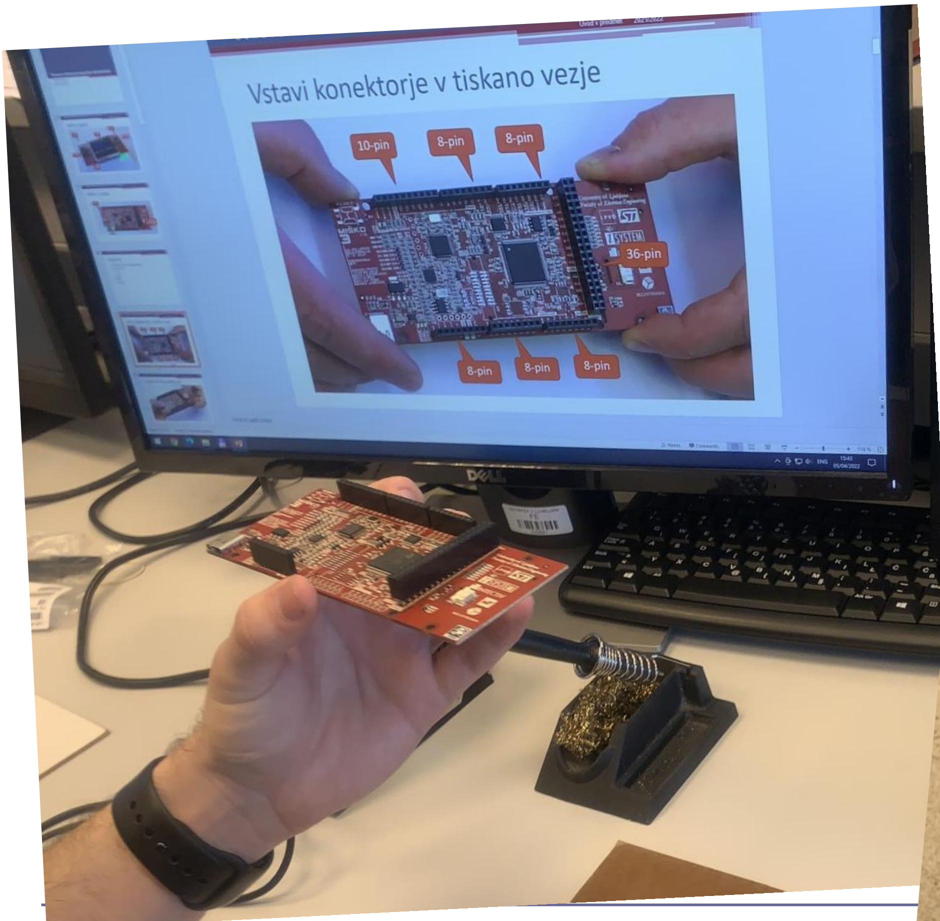
From <https://controllerstech.com/waveplayer-using-stm32-discovery/>

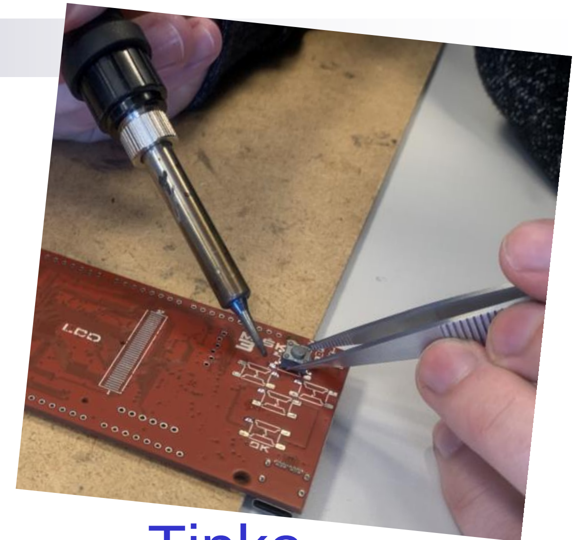
VIN projekt - VP5: STM32-Edge computing, CubeIDE projekti, Miško3

- VIN projekt
- AI v vgrajenih napravah („Edge Computing“)
- STM32 CubeIDE – Delo s projekti
- STM32 CubeIDE, SPI in LIS3DSH
- STM32 CubeIDE, I2C in CS43L22
- Miško3 – demo projekt

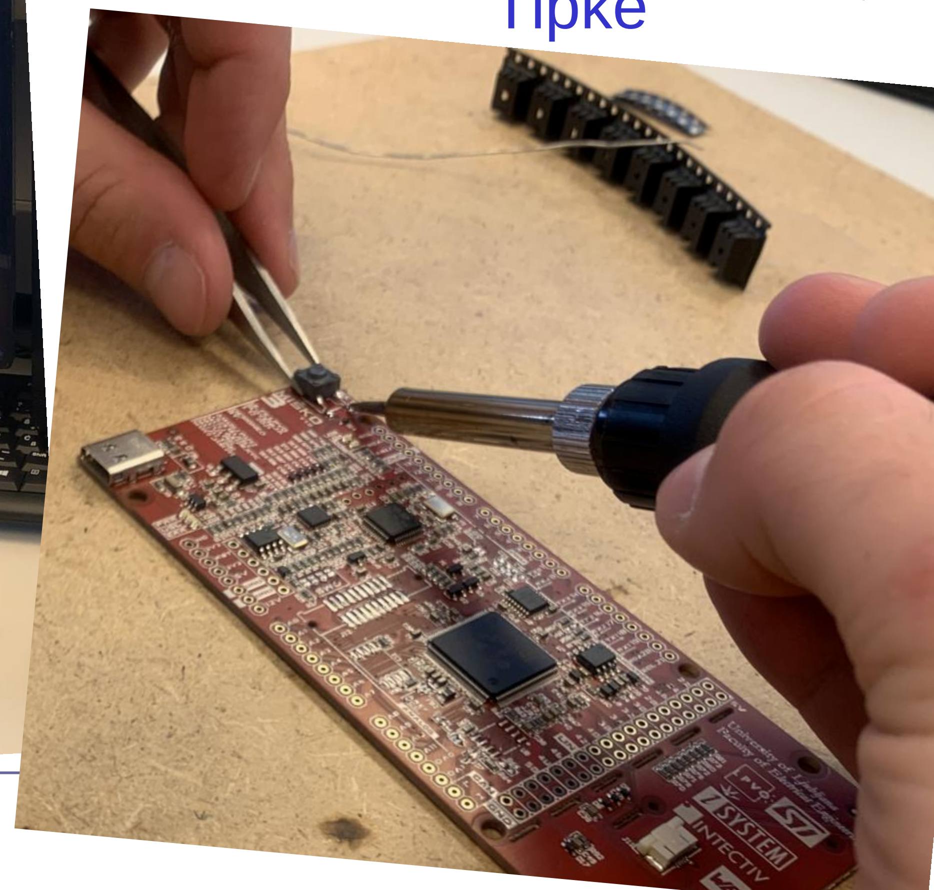
Miško 3 in „Spajka“ party 2022

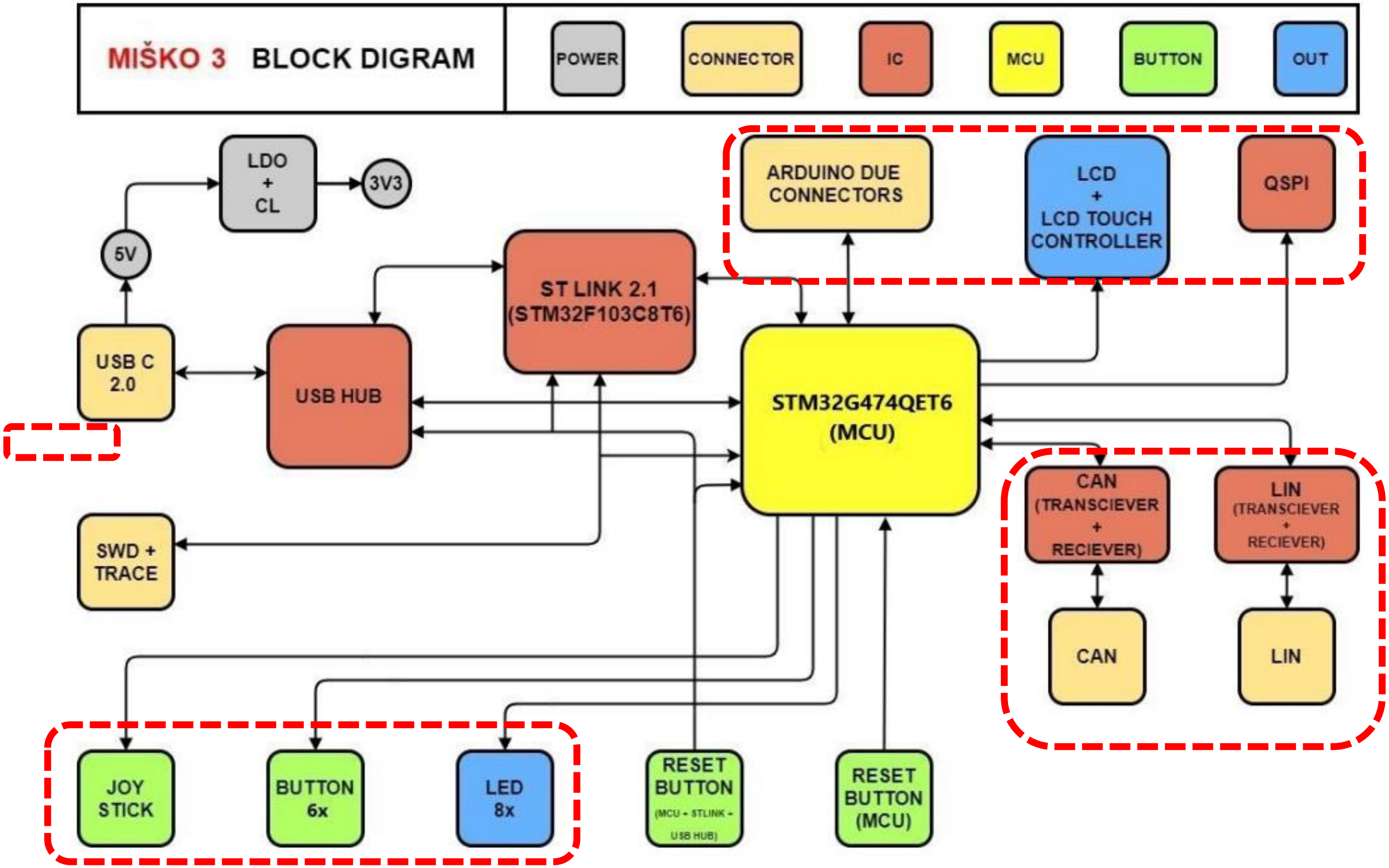
Konektorji





Tipke





Slika 1: Bločni diagram razvojnega Sistema MiŠKo 3

```c
int main(void)
{
  /* USER CODE BEGIN 1 */
coord_t joystick_raw, joystick_out;
joystick_t joystick;
uint8_t MSG[100]={0};
uint16_t touch_x = 0, touch_y = 0;

char str[10];
float framerate;

  /* USER CODE END 1 */

  /* MCU Configuration------------------
--------------------------------------*/

  /* Reset of all peripherals, Initializes
 the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_ADC1_Init();
MX_ADC2_Init();
MX_FMC_Init();
MX_I2C2_Init();
MX_UART4_Init();
MX_UART5_Init();
MX_USART1_UART_Init();
MX_USART2_UART_Init();
MX_QUADSPI1_Init();
MX_SPI1_Init();
MX_TIM5_Init();
MX_TIM8_Init();
MX_TIM20_Init();
MX_ADC3_Init();
MX_DAC1_Init();
MX_DAC2_Init();
MX_FDCAN2_Init();
MX_I2C1_Init();
MX_TIM15_Init();
MX_USART3_UART_Init();
MX_ADC4_Init();
MX_USB_Device_Init();
MX_DMA_Init();
MX_CRC_Init();
MX_TIM6_Init();

  /* USER CODE BEGIN 2 */

  LED_init();
  KBD_init();
  SCI_init();
  joystick_init(&joystick);

  for (uint8_t i=0;i<3;i++)
  {
  HAL_Delay(250);
  LEDs_on(0xFF);
  HAL_Delay(250);
  LEDs_off(0xFF);
  }

  LCD_Init();
  UG_Init(&gui, UserPixelSetFunction,
ILI9341_GetParam(LCD_WIDTH),
ILI9341_GetParam(LCD_HEIGHT));
  UG_FontSelect(&FONT_8X12);
  UG_SetForecolor(C_WHITE);
  UG_SetBackcolor(C_BLACK);
  UG_DriverRegister(DRIVER_FILL_FRAME, (void
*)_HW_FillFrame_);
  UG_DriverEnable(DRIVER_FILL_FRAME);

  DrawStartScreen();
  framerate = DrawColors(80);

  UG_SetForecolor(C_WHITE);
  UG_FontSelect(&FONT_16X26);
  sprintf(str,"%.0f fps",framerate);
  UG_PutString(5,105,str);

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
```

https://github.com/LAPSyLAB/Misko3_Docs_and_Projects

VP 5 – Miško 3 – Glavna zanka

```c
 while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

//LEDs
  LED_set(LED0, !KBD_get_button_state(BTN_OK));
  LED_set(LED1, !KBD_get_button_state(BTN_DOWN));
  LED_set(LED2, !KBD_get_button_state(BTN_RIGHT));
  LED_set(LED3, !KBD_get_button_state(BTN_UP));
  LED_set(LED4, !KBD_get_button_state(BTN_LEFT));
  LED_set(LED6, !KBD_get_button_state(BTN_ESC));
  LED_set(LED7, !KBD_get_button_state(BTN_JOY));

 // Joystick
 HAL_ADC_Start(&hadc4);
 HAL_ADC_PollForConversion(&hadc4,10);// Waiting for ADC conversion
 joystick_raw.x=HAL_ADC_GetValue(&hadc4);

 HAL_ADC_Start(&hadc4);
 HAL_ADC_PollForConversion(&hadc4,10);// Waiting for ADC conversion
 joystick_raw.y=HAL_ADC_GetValue(&hadc4);
 HAL_ADC_Stop(&hadc4);

 joystick_get(&joystick_raw, &joystick_out, &joystick);
 UG_DrawCircle(joystick_out.x+250, joystick_out.y+50,5, C_YELLOW);
```

```c
// Touchscreen
if(XPT2046_TouchPressed())
{
uint16_t x = 0, y = 0;

if(XPT2046_TouchGetCoordinates(&x, &y, 0))
{
touch_x = x;
touch_y = y;
UG_FillCircle(x, y,2, C_GREEN);
UG_FillCircle(250, 50, 49, C_BLACK);
}
}

 sprintf(MSG, "Joystick X:%05d, Y:%05d, Touch: X:%05d, Y:%05d
\r",joystick_out.x,joystick_out.y, touch_x, touch_y);

SCI_send_string(MSG);
 CDC_Transmit_FS(MSG, strlen(MSG));
 UG_DrawCircle(250, 50, 50, C_RED);

 HAL_Delay(20);
  }

  /* USER CODE END 3 */
```

https://github.com/LAPSyLAB/Misko3_Docs_and_Projects

VIN projekt - VP5: STM32-Edge computing, CubeIDE projekti, Miško3

- Diskusija, vprašanja ?