Vhodno izhodne naprave

Laboratorijska vaja 6 - VP 6

STM32F4 – ADC, PWM, SPI-Accel, I2C, Tracing, Osciloskop

VIN projekt - VP6: STM32F4 VIN Demo

- Osvežitev: STM32F4

- CubeIDE projekt STM32F4 in V/I naprave :
  - CubeIDE projekt, GPIO in VCOM port
  - PWM - LED dimmer, brenčač
  - SPI - LIS3DSH pospeškomer
  - I2C - CS43L22 zvočni čip
  - ADC

- Sledenje („tracing“) - CubeMonitor, osciloskop

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:

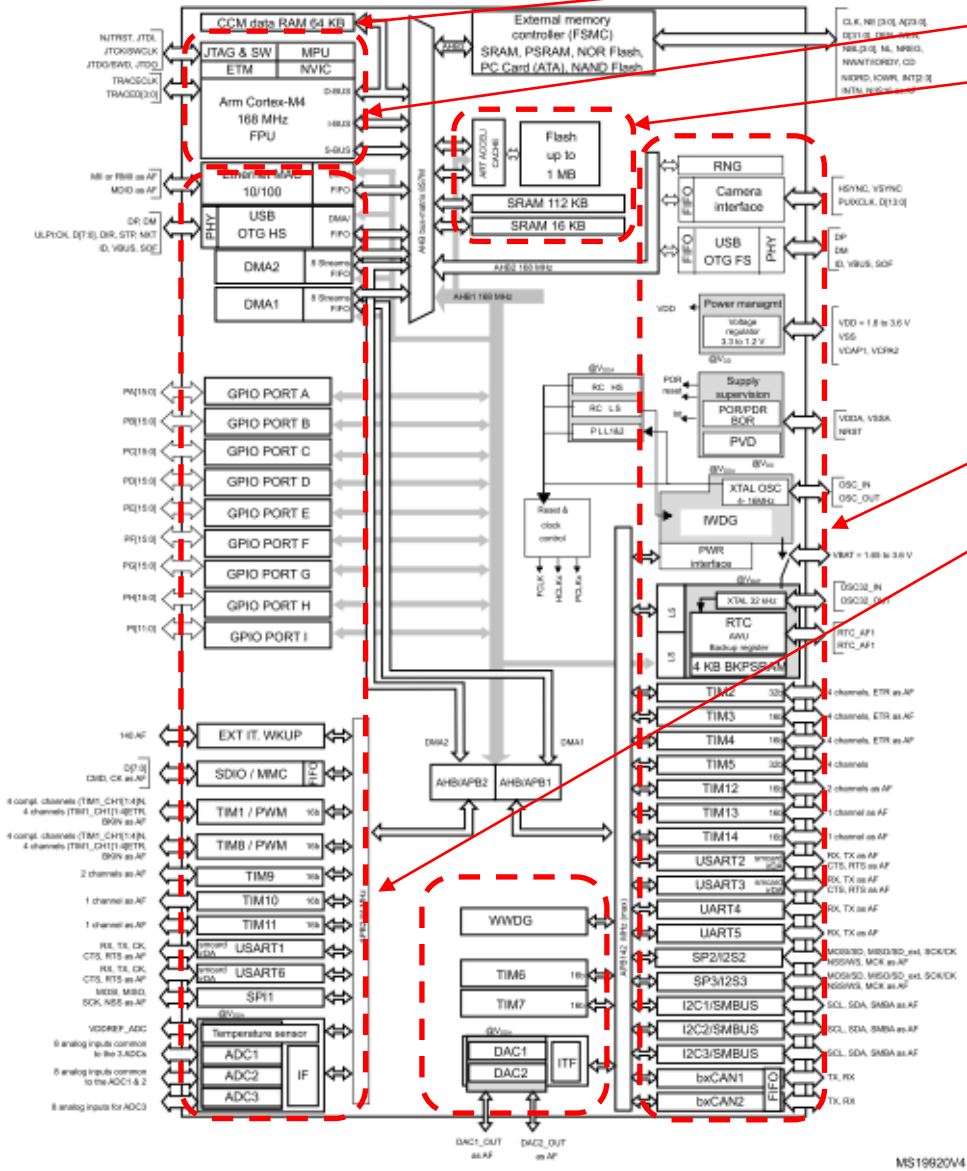
- •USB with Micro-AB
- •Stereo headphone output jack
- •2.54 mm pitch extension header for all LQFP100 I/Os for quick connection to prototyping board and easy probing

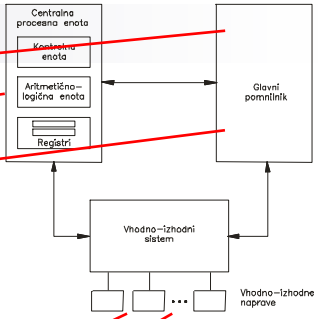
•External application power supply: 3 V and 5 V

STM32

STM32F4

STM32F407VG

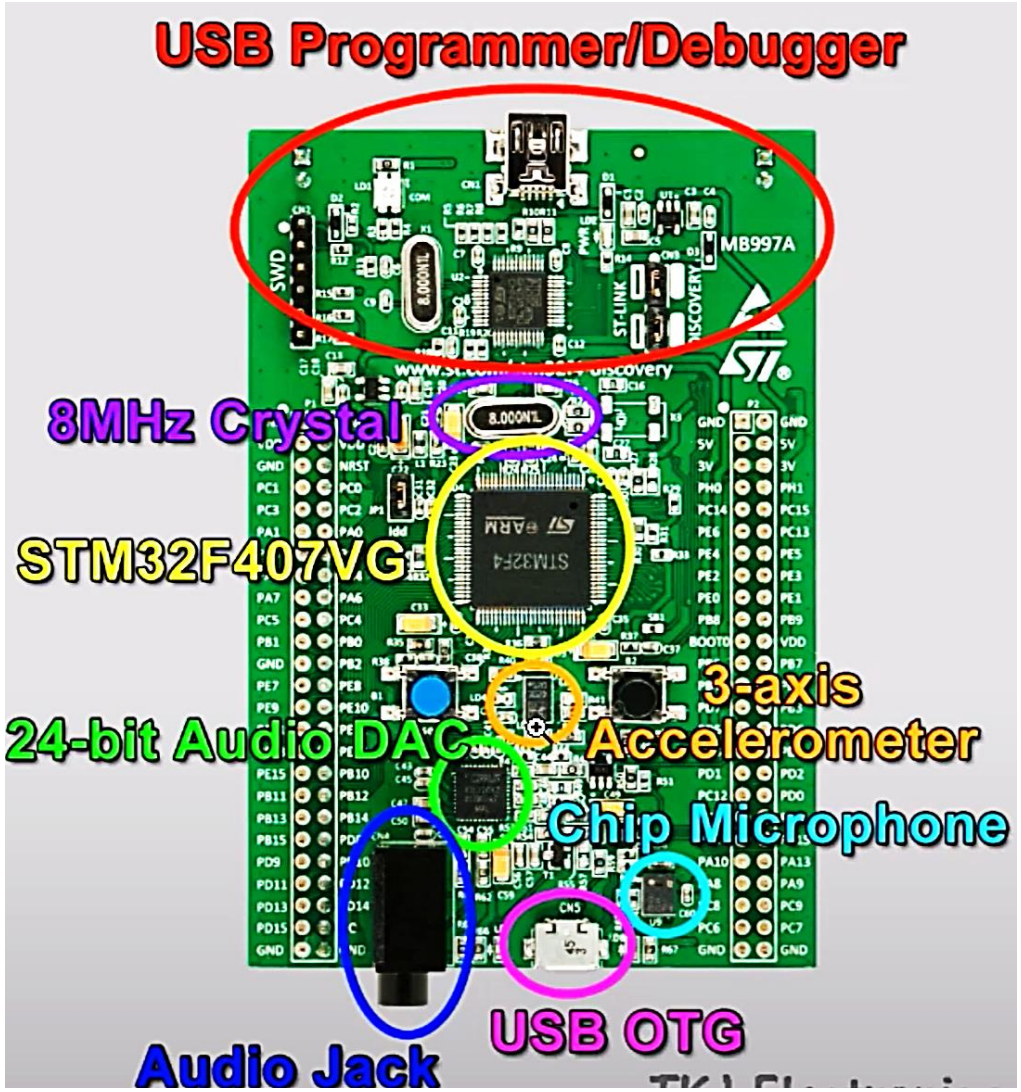




STM32F4

STM32F4DISCOVERY

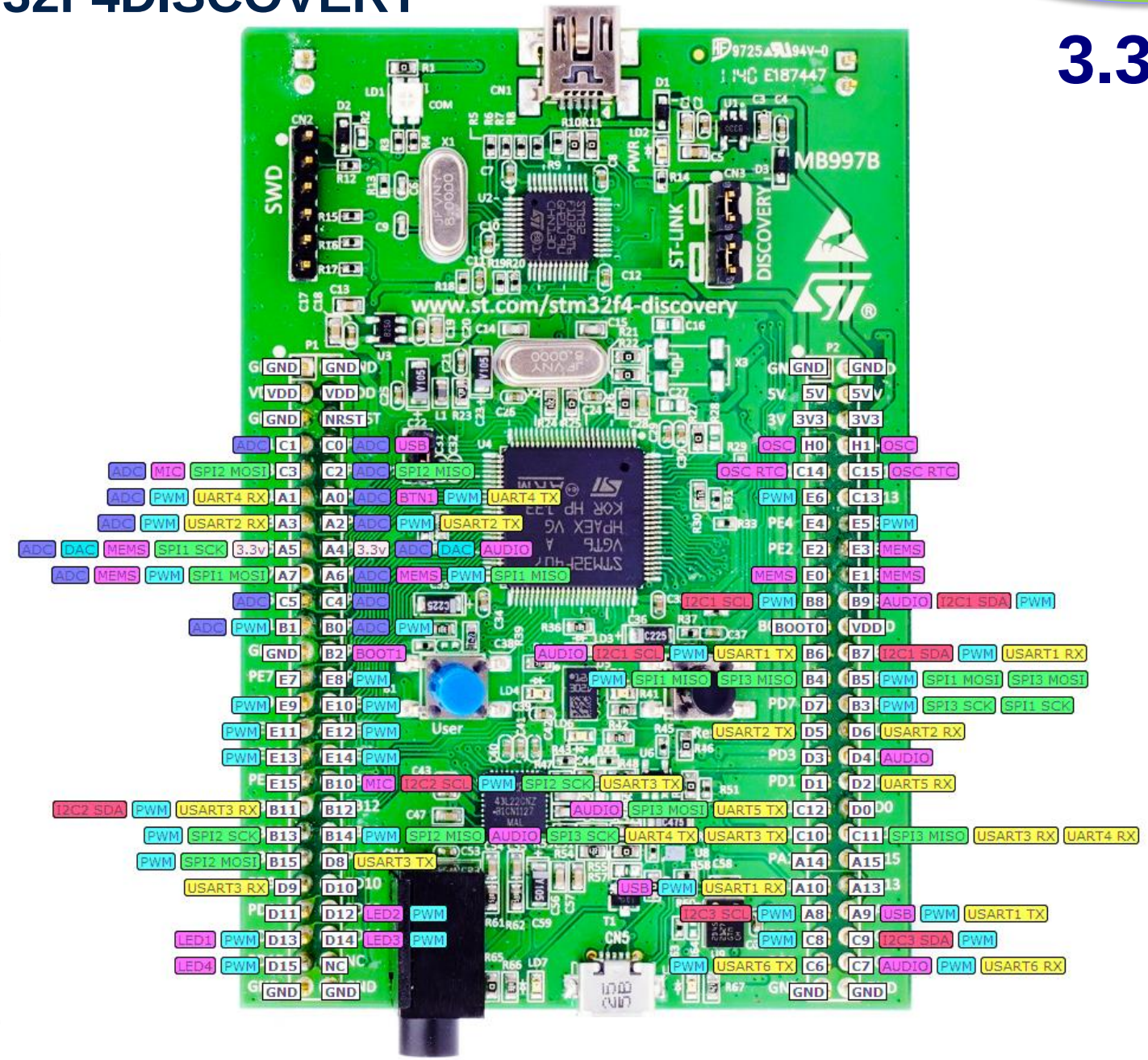
3.3V !!!



STM32F4DISCOVERY

STM32F4

3.3V !!!



Delo na STM32F4 razvojnem sistemu

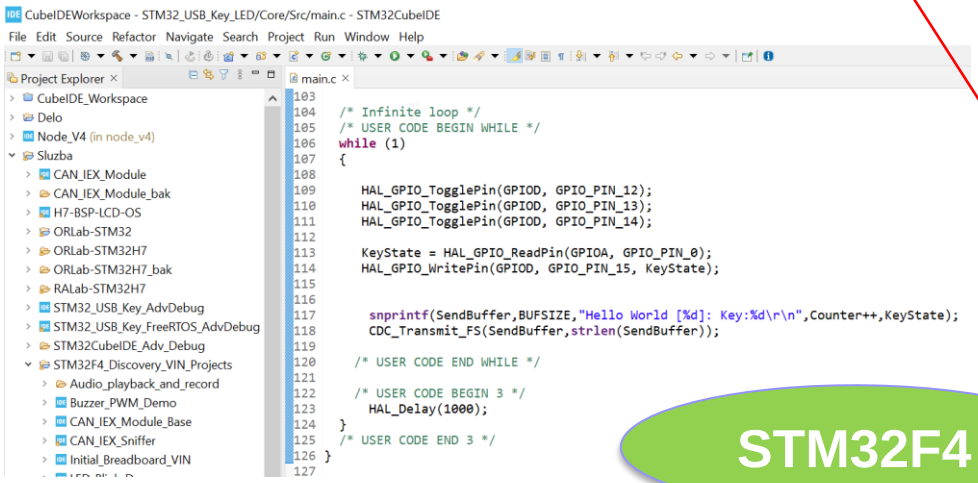
Mikro USB VCom-port

Priključitev :

- **Mini USB** priklop na **krajši stranici**, svetita rdeči **LED** diodi
- **Micro USB** priklop (VCom port)

Poseben začetni projekt za *STM32F4* (e-učilnica) :

- ***dodajanje vsebine (main.c):***



STM32F4

--------- Razvojni sistem STM32F407 Discovery ----------

- STM32F4DISCOVERY Discovery kit with STM32F407VG MCU
- VINLab-STM32 - GitHub repozitorij
- ORLab-STM32 - GitHub repozitorij
- STM32F4 - Dokumentacija

Lastni viri :

https://github.com/LAPSyLAB/STM32F4_Discovery_VIN_Projects

https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples

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

Osnovni projekt CubeIDE – GPIO – različni 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

RA, OR

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

VIN

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 */
```

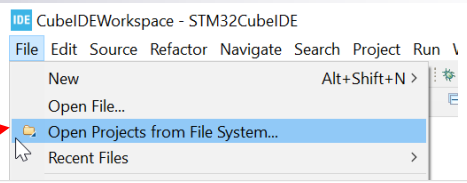
```
void HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx,
uint16_t GPIO_Pin)
{
  uint32_t odr;

  /* 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);
}
```

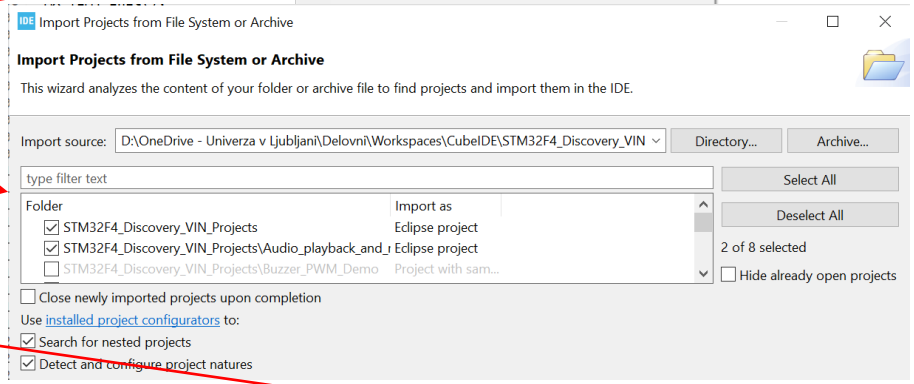
https://github.com/LAPSyLAB/STM32F4_Discovery_VIN_Projects/tree/main/LED_Blink_Demo

Delo v CubeIDE

Vzpostavitev začetnega projekta :

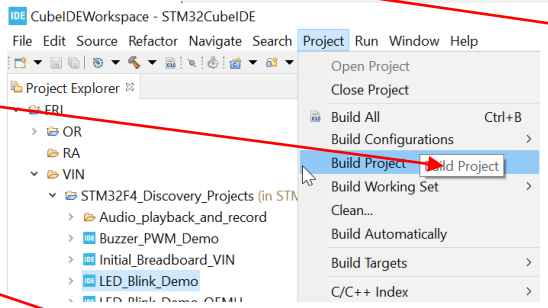
- **Uvoz obstoječega**
  - Open projects from File System
  - Select project(s)
- **Nov projekt Cube MX**
- **Kopiranje obstoječega**

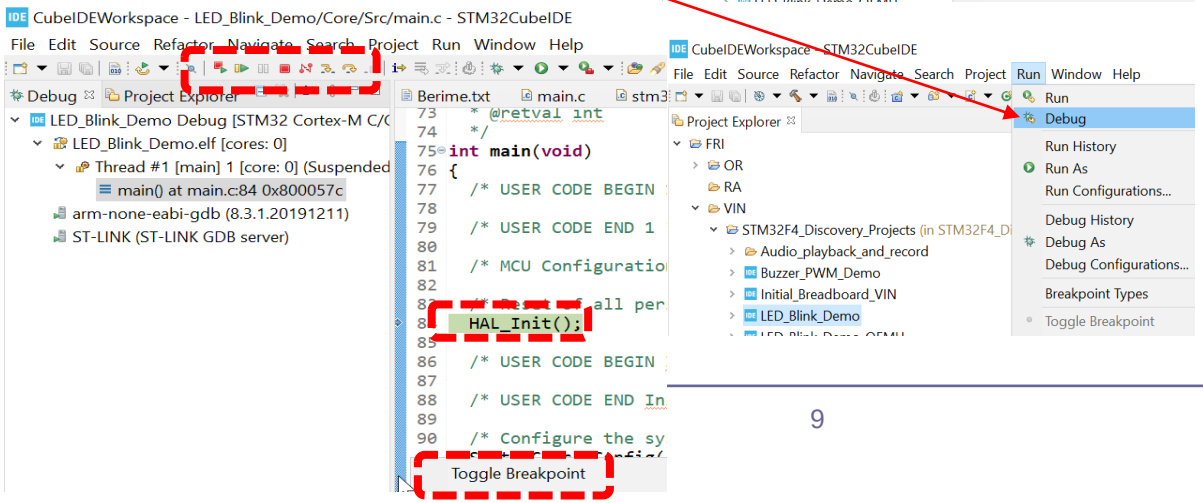




Prevajanje, zagon :

- Project -> Build Project
- Run -> Debug
- Step (Into,Over), Breakpoints





Navodila :

- CubeIDE asm projekt
  1) Edit > Copy.
  2) Edit > Paste.
  3) Delete the Debug.launch file.
  4) Project > Clean.
  5) Project > Build Project.
  6) Debug As Stm32 Application.
  7) And debug the application
  8) Add breakpoint on first instruction if necessary
- CubeIDE projekt z CubeMX
  1) Edit > Copy.
  2) Edit > Paste.
  3) Rename the ioc files.
  4) Delete the Debug.launch file.
  5) Project > Clean.
  6) Generate the CubeMX.
  7) Project > Build Project.
  8) Debug As Stm32 Application.
  9) And debug the application.

Skopiram, preimenujem, generiram ioc, clean in build

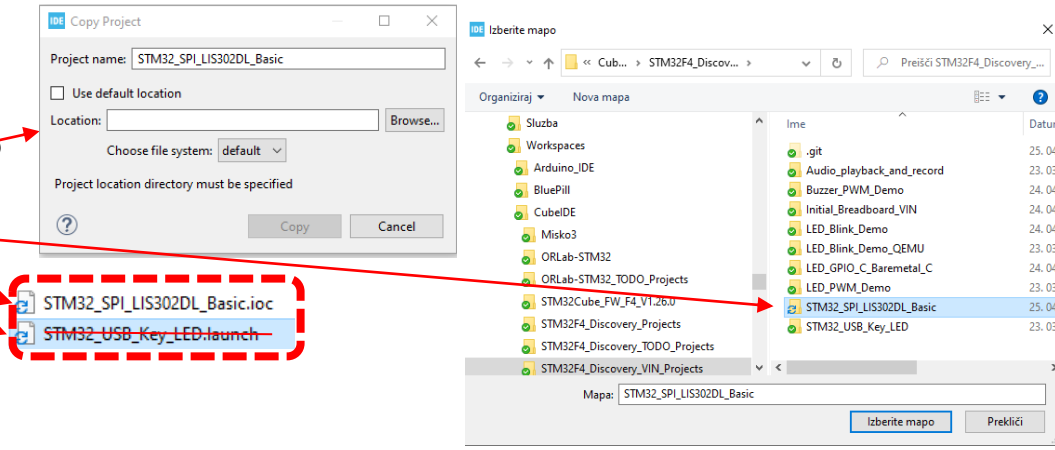
CubeIDE – delo s projekti

Kopiranje/preimenovanje projekta :

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

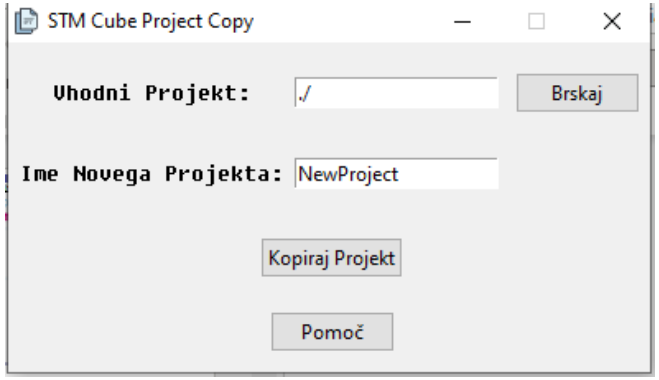
Kopiranje CubeIDE projekta z CubeMX .ioc datoteko

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

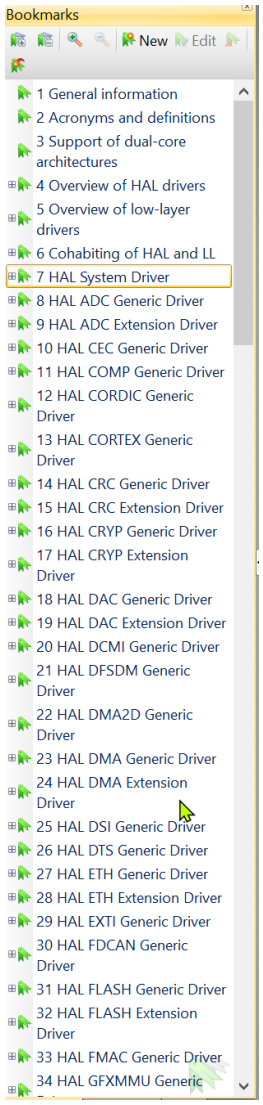
- **Kopiranje projekta Cube MX II**:
  - Uporaba orodja



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

Vir: UM2217 - HAL and Low-layer drivers

STM32F4



Programska knjižnica

Vsebuje predpripravljene funkcije za delo s sistemskimi in V/I napravami.

life.augmented

UM2217

User manual

Description of STM32H7 HAL and low-layer drivers

This section contains the following APIs:

- HAL_Init()
- HAL_DeInit()
- HAL_MspInit()
- HAL_MspDeInit()
- HAL_InitTick()

This section contains the following APIs:

- HAL_USART_Transmit()
- HAL_USART_Receive()
- HAL_USART_TransmitReceive()
- HAL_USART_Transmit_IT()
- HAL_USART_Receive_IT()
- HAL_USART_TransmitReceive_IT()
- HAL_USART_Transmit_DMA()
- HAL_USART_Receive_DMA()
- HAL_USART_TransmitReceive_DMA()

This section contains the following APIs:

- HAL_IncTick()
- HAL_GetTick()
- HAL_GetTickPrio()
- HAL_SetTickFreq()
- HAL_GetTickFreq()
- HAL_Delay()
- HAL_SuspendTick()
- HAL_ResumeTick()
- HAL_GetHalVersion()

35.2.4 **IO operation functions**

This section contains the following APIs:

- HAL_GPIO_ReadPin()
- HAL_GPIO_WritePin()
- HAL_GPIO_TogglePin()
- HAL_GPIO_LockPin()
- HAL_GPIO_EXTI_IRQHandler()
- HAL_GPIO_EXTI_Callback()

This section contains the following APIs:

- HAL_I2C_Init()
- HAL_I2C_DeInit()
- HAL_I2C_MspInit()
- HAL_I2C_MspDeInit()
- HAL_I2C_RegisterCallback()
- HAL_I2C_UnRegisterCallback()
- HAL_I2C_RegisterAddrCallback()
- HAL_I2C_UnRegisterAddrCallback()

STM32F4DISCOVERY

3.3V !!!

Table 11. Voltage characteristics

Symbol	Ratings	Min	Max	Unit
V_{DD}–V_{SS}	External main supply voltage (including V_{DDA}, V_{DD})(1)	–0.3	4.0	V
V_{IN}	Input voltage on five-volt tolerant pin(2)	V_{SS}–0.3	V_{DD}+4	
	Input voltage on any other pin	V_{SS}–0.3	4.0	
$\lvert\Delta V_{DDx}\rvert$	Variations between different V_{DD} power pins	-	50	mV
$\lvert V_{SSX} - V_{SS}\rvert$	Variations between all the different ground pins including V_{REF-}	-	50	
$V_{ESD(HBM)}$	Electrostatic discharge voltage (human body model)	see *Section 5.3.14: Absolute maximum ratings (electrical sensitivity)*		

Table 12. Current characteristics

Symbol	Ratings	Max.	Unit
I_{VDD}	Total current into V_{DD} power lines (source)(1)	240	mA
I_{VSS}	Total current out of V_{SS} ground lines (sink)(1)	240	
I_{IO}	Output current sunk by any I/O and control pin	25	
	Output current source by any I/Os and control pin	25	
$I_{INJ(PIN)}$ (2)	Injected current on five-volt tolerant I/O(3)	–5/+0	
	Injected current on any other pin(4)	±5	
$\Sigma I_{INJ(PIN)}$(4)	Total injected current (sum of all I/O and control pins)(5)	±25	

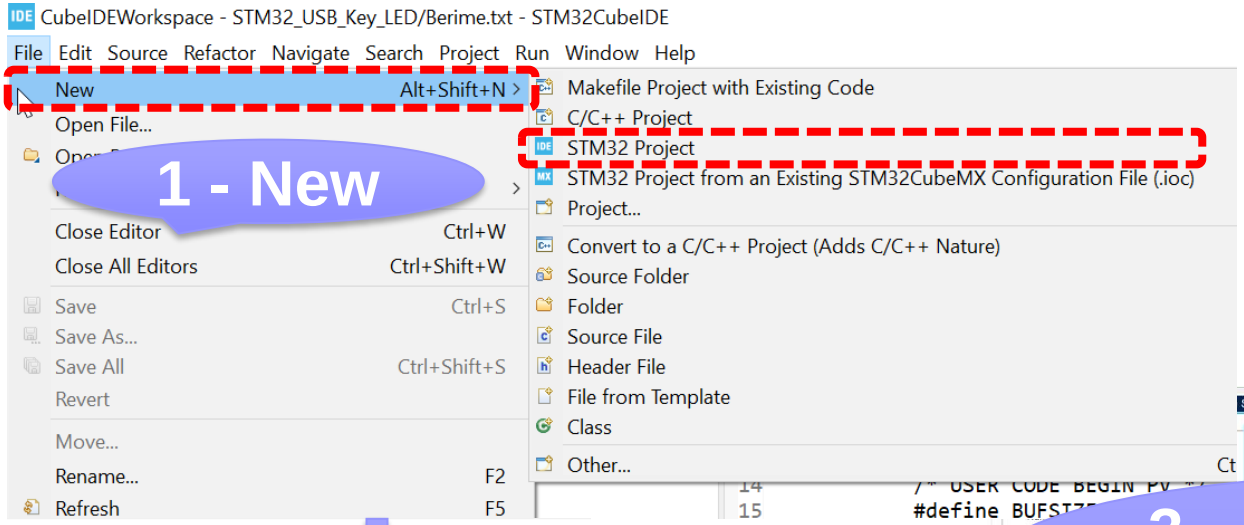
VIN projekt - VP6: STM32F4 VIN Demo

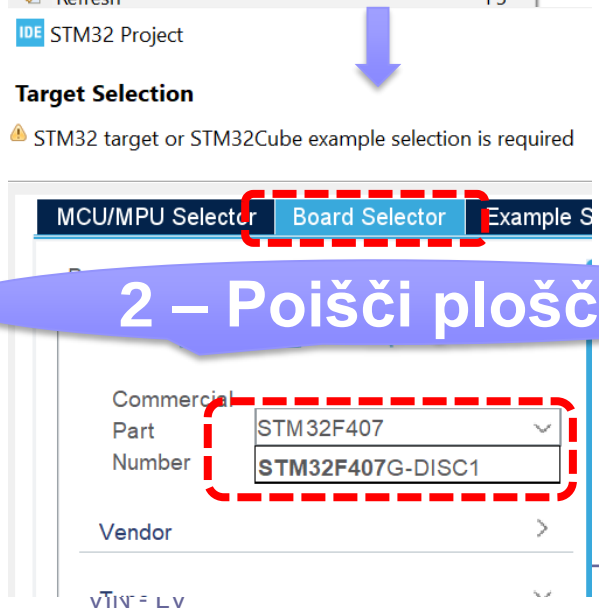
- Osvežitev: STM32F4
- CubeIDE projekt STM32F4 in V/I naprave :
  - CubeIDE projekt, GPIO in VCOM port
  - PWM - LED dimmer, brenčač
  - SPI - LIS3DSH pospeškomer
  - I2C - CS43L22 zvočni čip
  - ADC
- Sledenje („tracing“) - CubeMonitor, osciloskop

CubeIDE – Vzpostavitev novega projekta

STM32F4

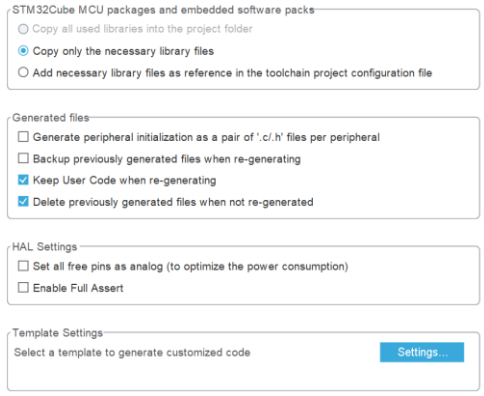
Nov projekt :

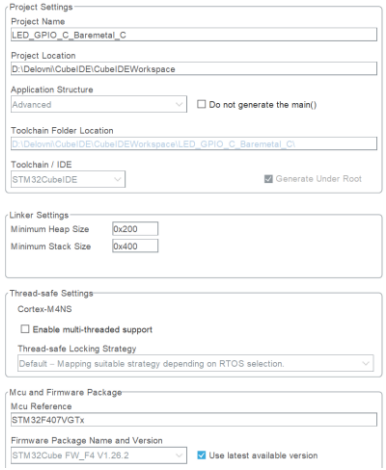


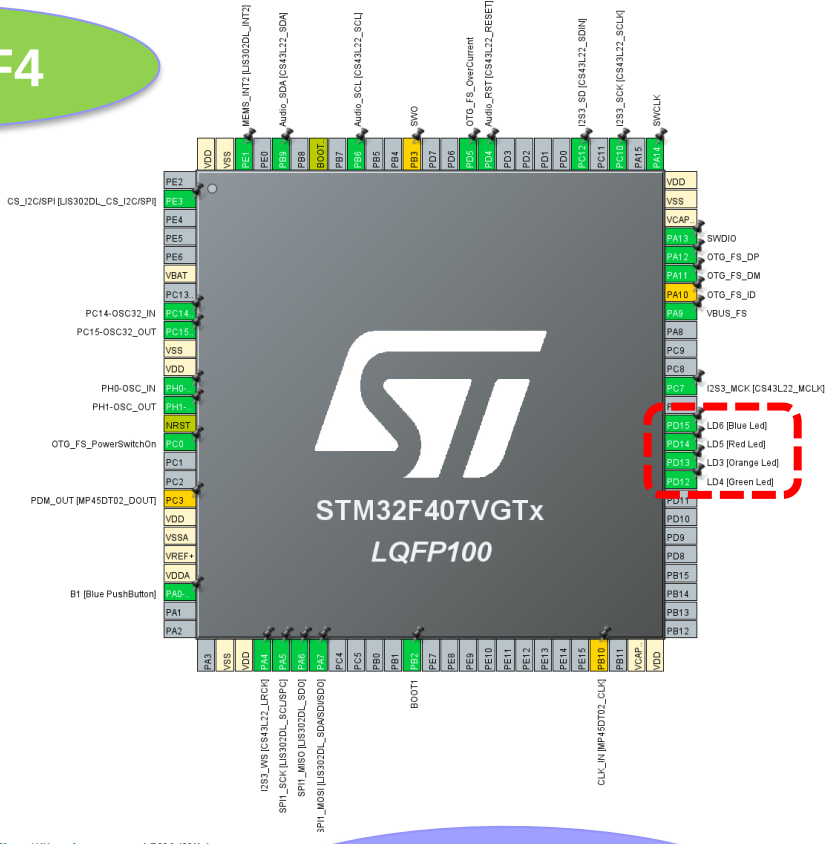


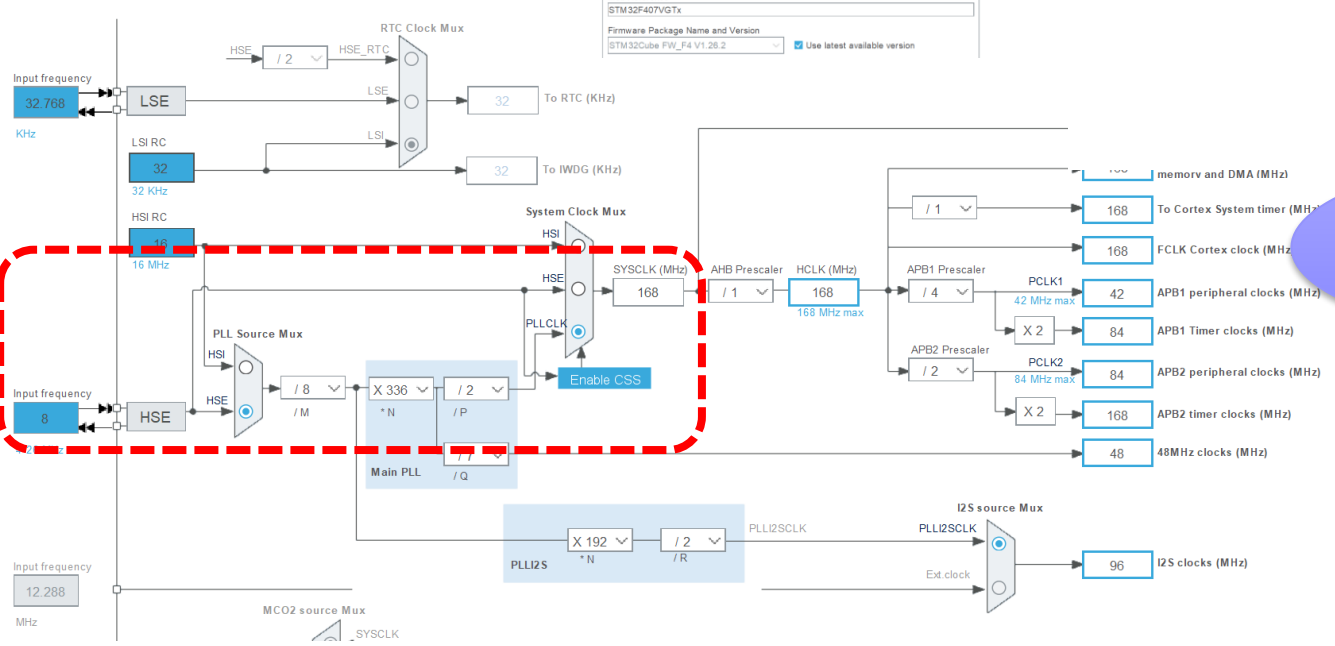
Osnovni projekt CubeIDE – CubeMX

Konfiguracija : priključki, knjižnice STM32F4





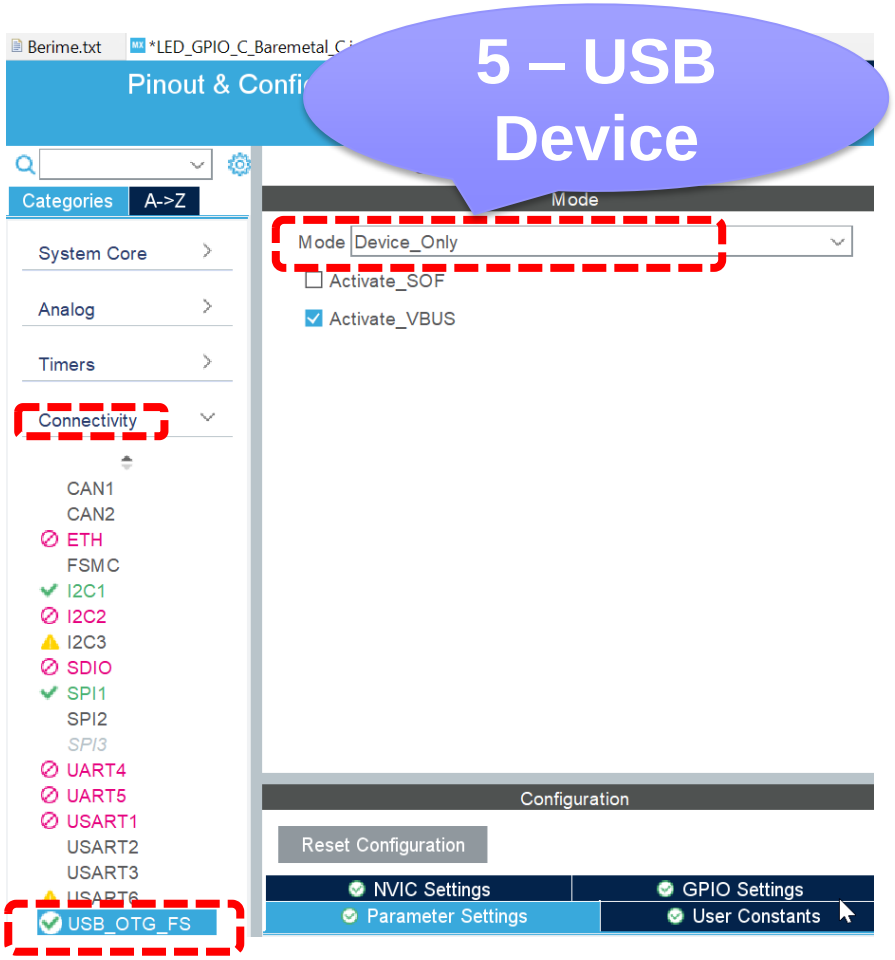


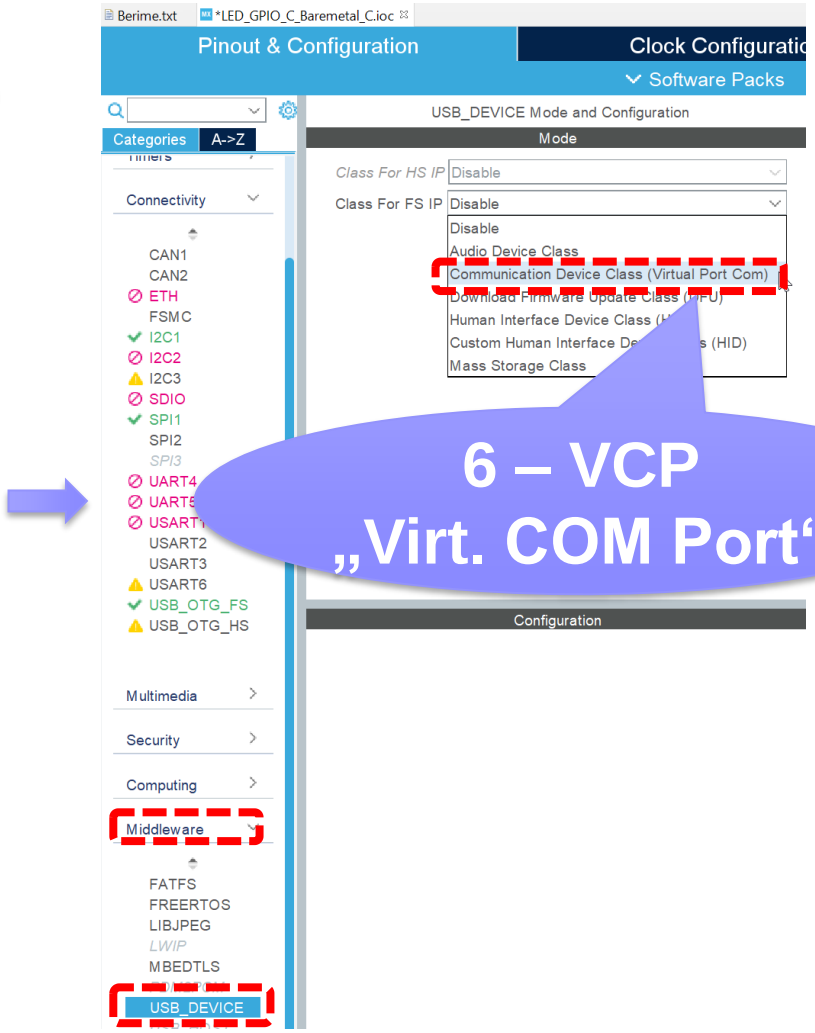


4 – Preveri nastavitve

Osnovni projekt CubeIDE – USB Virtual COM Port

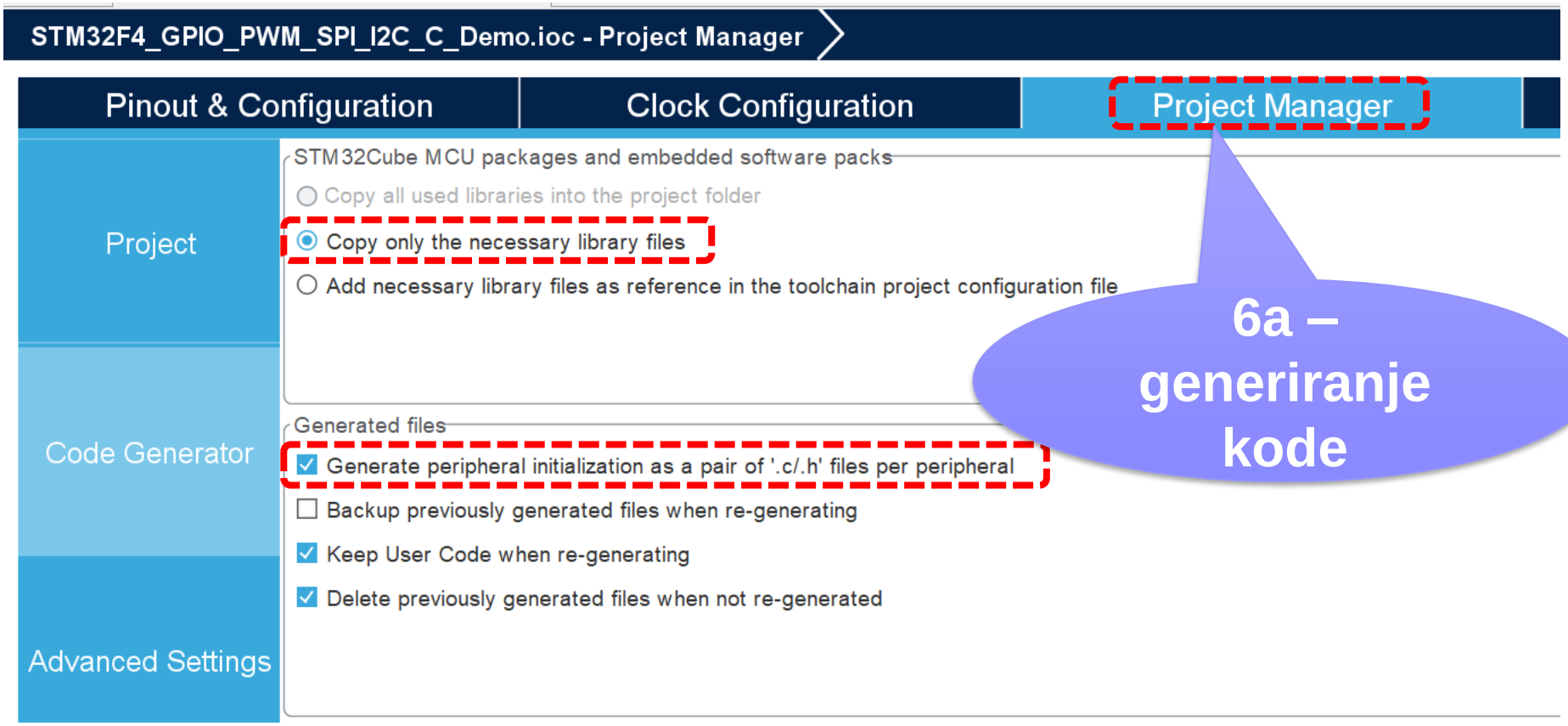
Konfiguracija : USB Device,CDC Class = Virtual COM Port





Osnovni projekt CubeIDE – USB Virtual COM Port

Konfiguracija : USB Device,CDC Class = Virtual COM Port

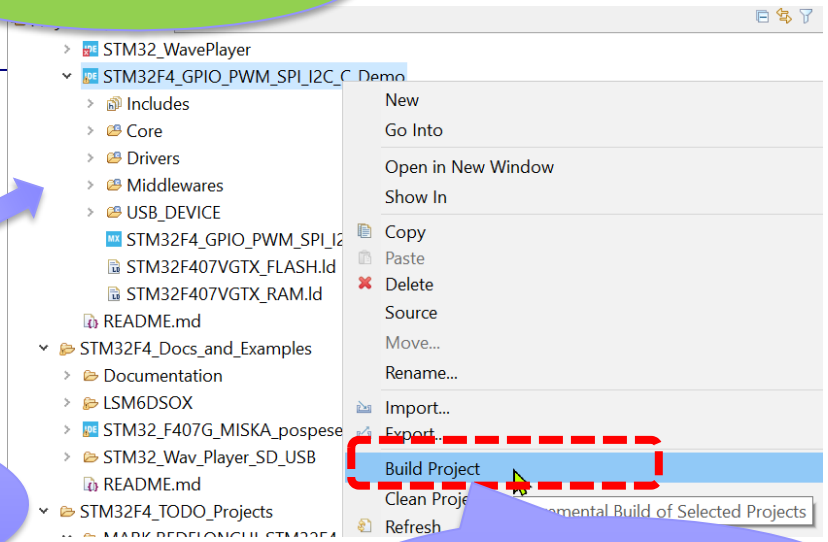


STM32F4

Program : za pošiljanje po USB Virtual COM Port

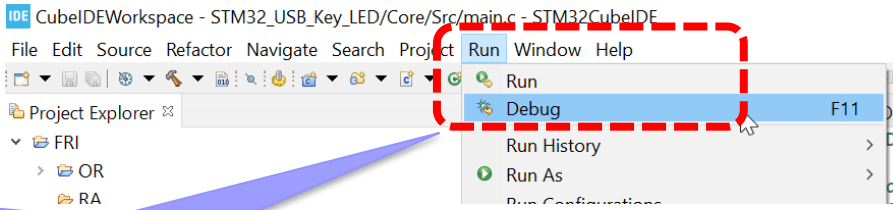
```
/* Private variables ------------*/

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

7 – USB VCP koda



8 – Build project

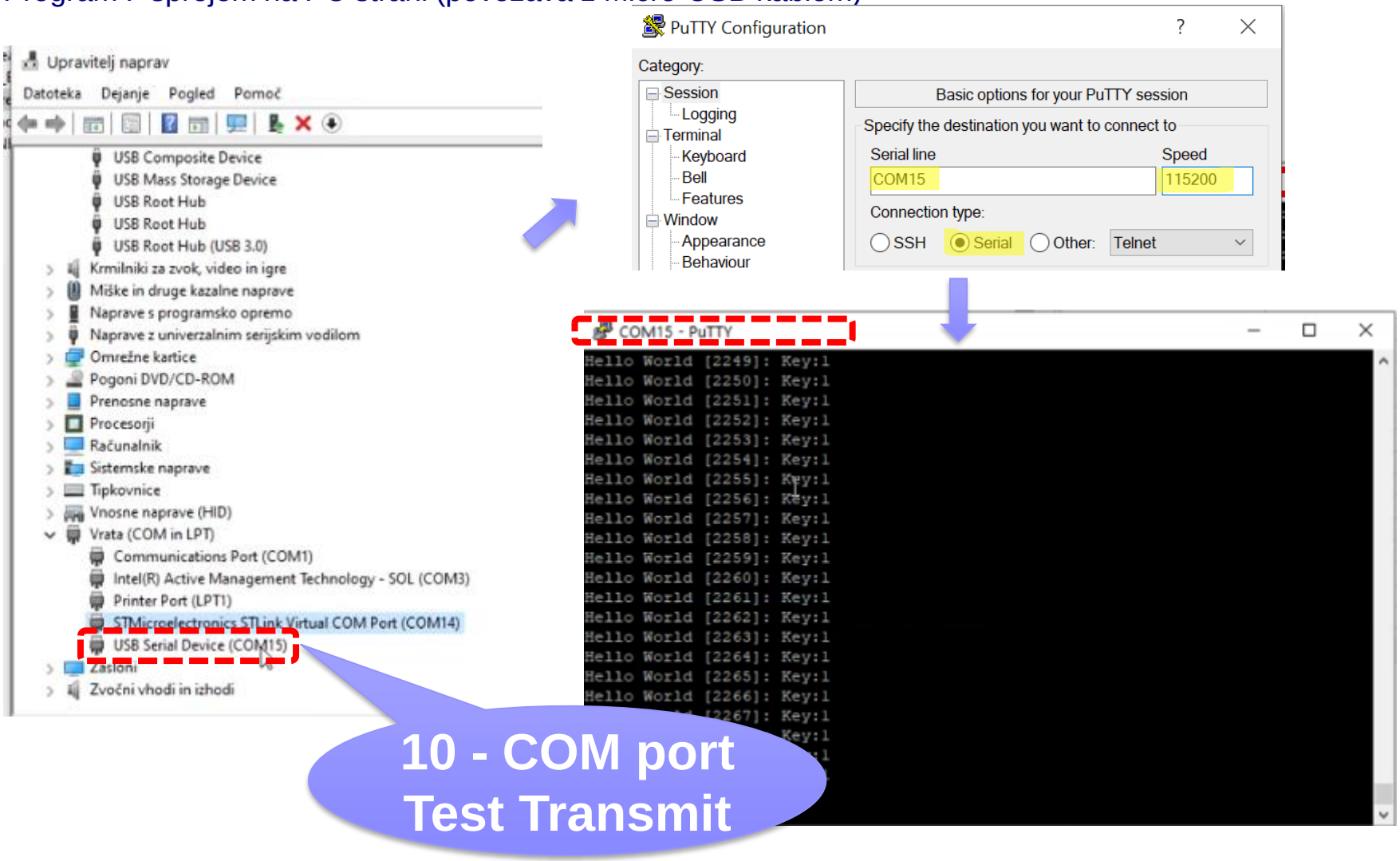
```
/* Infinite loop */
 /* USER CODE BEGIN WHILE */
 while (1)
 {
         snprintf(SendBuffer,BUFSIZE,"Hello World [%d]\r\n",Counter++);
         CDC_Transmit_FS(SendBuffer,strlen(SendBuffer));
 /* USER CODE END WHILE */

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



9 – Debug project

Osnovni projekt CubeIDE – USB Virtual COM Port

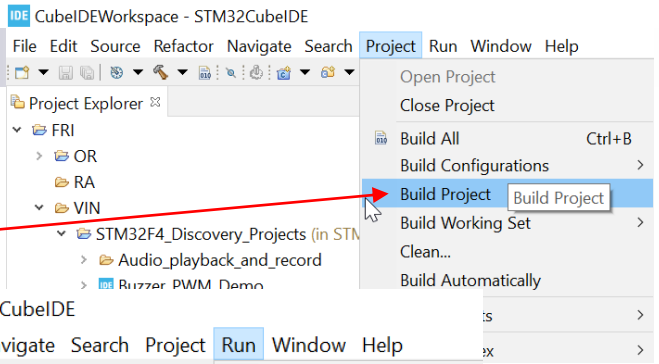
Program : sprejem na PC strani (povezava z Micro-USB kablom)

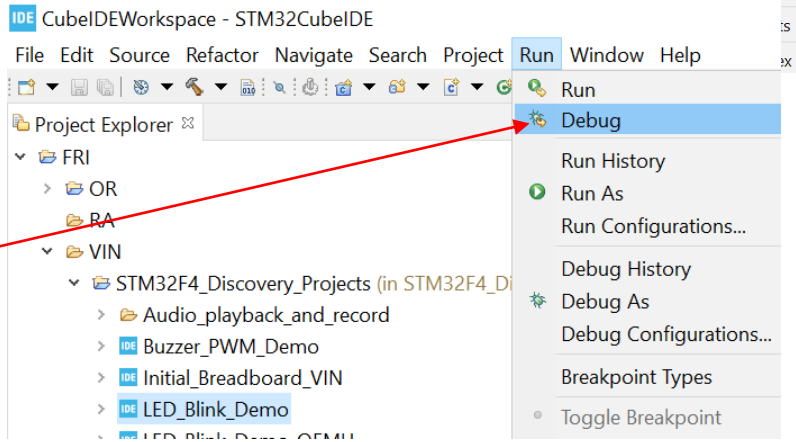


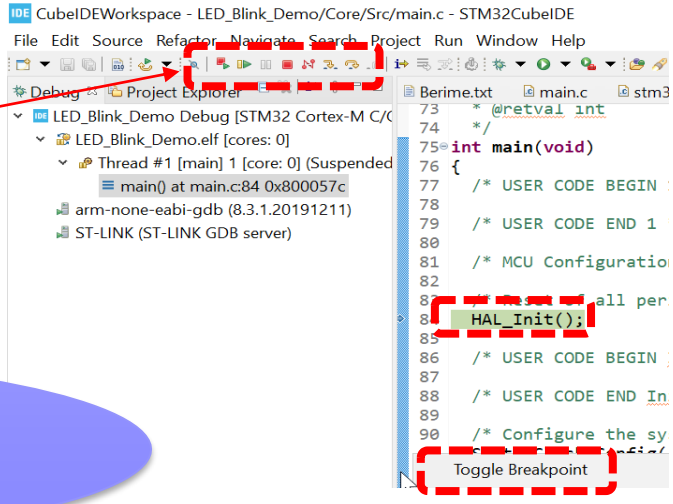
CubeIDE – Zagon, debug

Prevajanje, zagon :

- Project -> Build Project
- Run -> Debug
- Step (Into,Over), Breakpoints

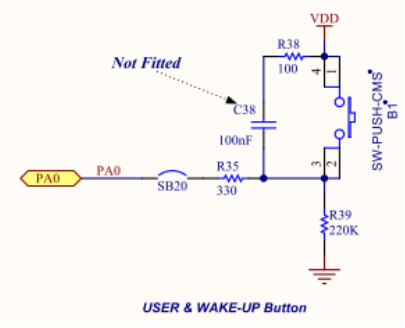


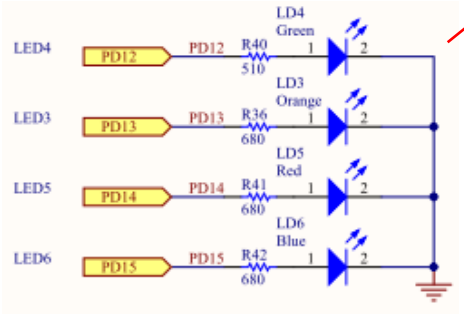


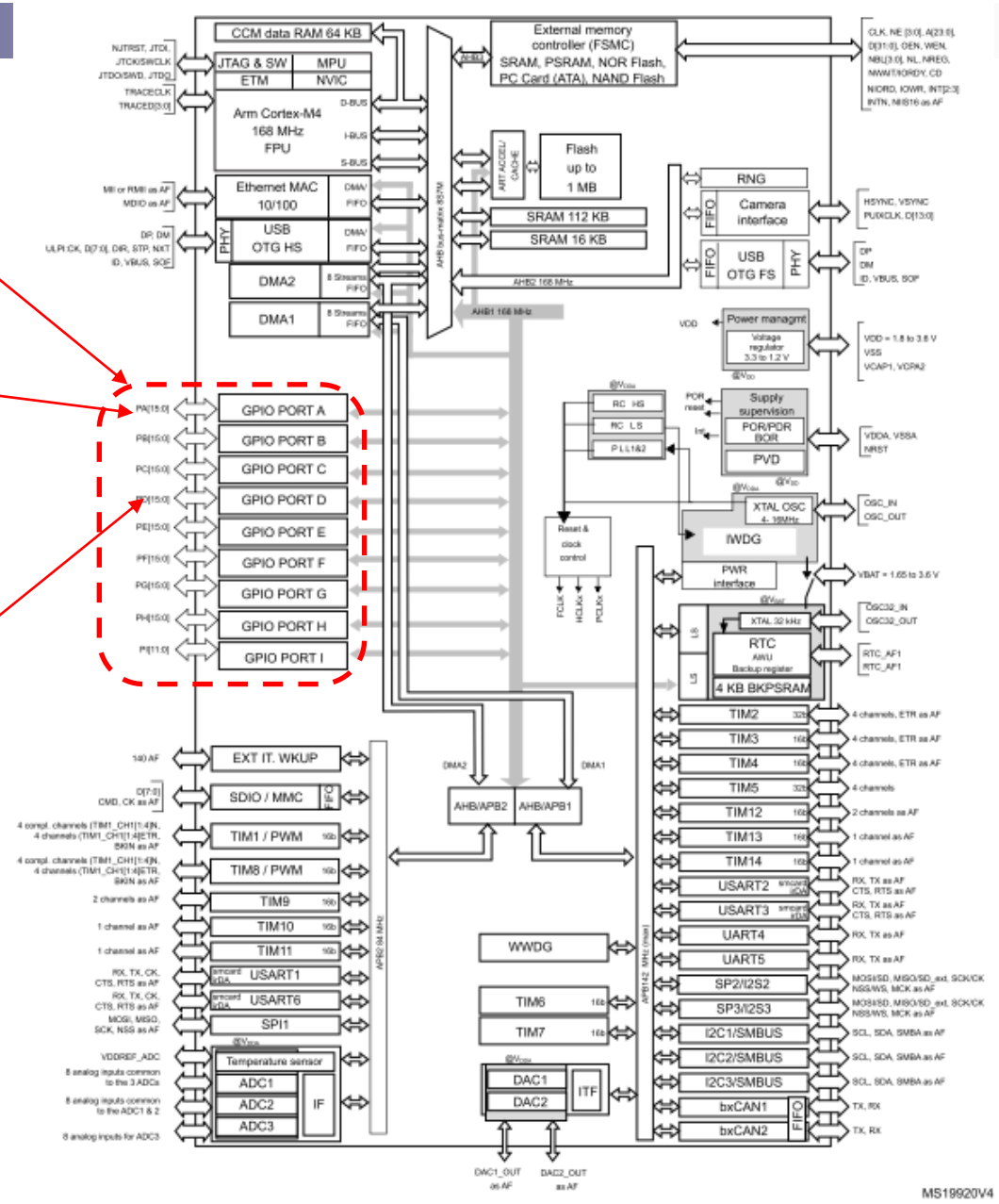


Build <->Debug project, …

GPIO Krmilnik







STM32F4

Vir: RM0090 Reference manual

life.augmented

RM0090
Reference manual

STM32F405/415, STM32F407/417, STM32F427/437 and STM32F429/439 advanced Arm®-based 32-bit MCUs

8 General-purpose I/Os (GPIO)

This section applies to the whole STM32F4xx family, unless otherwise specified.

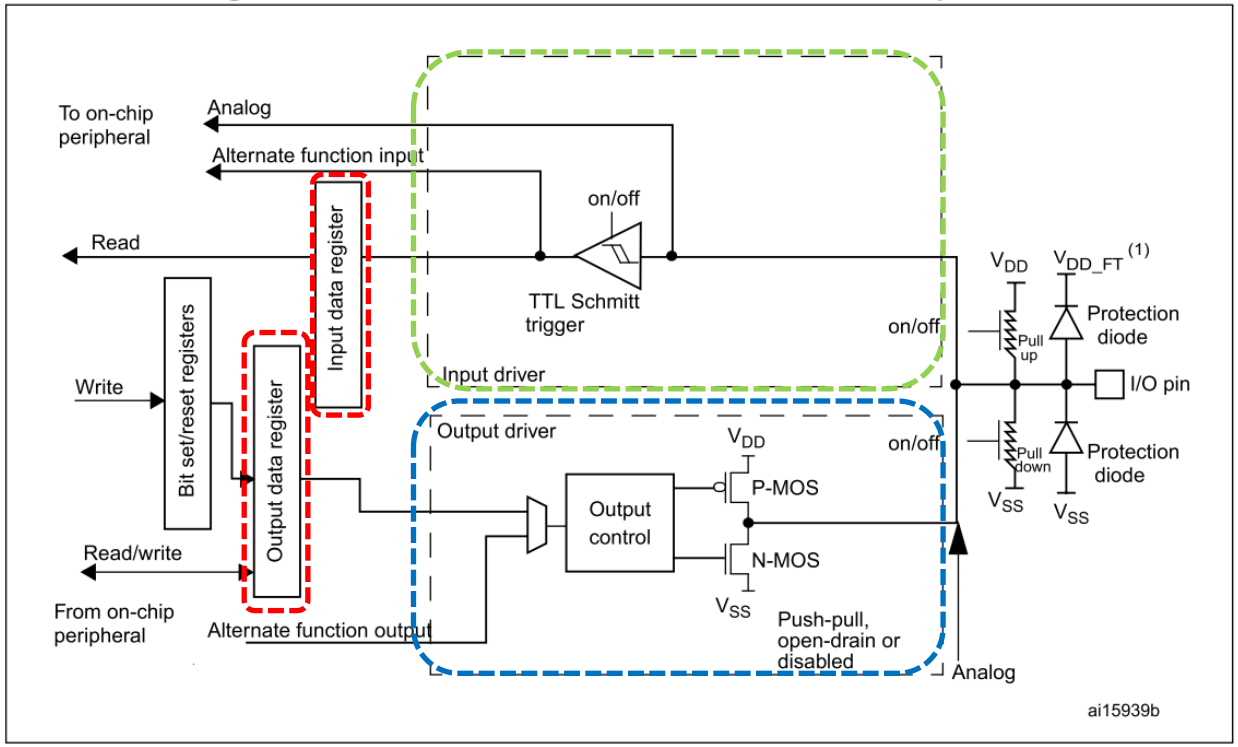
8.1 GPIO introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), a 32-bit set/reset register (GPIOx_BSRR), a 32-bit locking register (GPIOx_LCKR) and two 32-bit alternate function selection register (GPIOx_AFRH and GPIOx_AFRL).

STM32F4

GPIO krmilnik – vhod/izhod

Figure 25. Basic structure of a five-volt tolerant I/O port bit



Potrebni koraki za krmiljenje izhoda:

1. RCC_AHB1ENR(Peripheral Clock Register): b_3=1 .. Port D Enable
2. **MODER (Mode Register): 01: General purpose output mode**
3. Default vrednosti že ustrezne v registrih :
   OTYPER (Output TYPE Register): 0: Output push-pull (reset state)
   OSPEEDR (Output SPEED Register): 00 – Low speed (reset state)
   PUPDR (Pull Up/Down Register): 00 – No pull (reset state)
4. določi stanje izhoda s pisanjem v ODR ali BSRR (nastavljamo na 1/0)

Osnovni projekt CubeIDE – GPIO – različni mogoči 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

Potrebni koraki za krmiljenje izhoda:

1. RCC_AHB1ENR(Peripheral Clock Register): b_3=1 .. Port D Enable
2. **MODER (Mode Register): 01: General purpose output mode**
3. Default vrednosti že ustrezne v registrih :
   OTYPER (Output TYPE Register): 0: Output push-pull (reset state)
   OSPEEDR (Output SPEED Register): 00 – Low speed (reset state)
   PUPDR (Pull Up/Down Register): 00 – No pull (reset state)
4. določi stanje izhoda s pisanjem v ODR ali BSRR (nastavljamo na 1/0)

Osnovni projekt CubeIDE – GPIO – tipka, LED diode

HAL - C

```
/* USER CODE BEGIN PV */
#define BUFSIZE 256
char SendBuffer[BUFSIZE];
int Counter;
int KeyState=0;

/* USER CODE END PV */
```

```
/* 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\r\n",Counter++,KeyState);
         CDC_Transmit_FS(SendBuffer,strlen(SendBuffer));

   /* USER CODE END WHILE */

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

UM1725

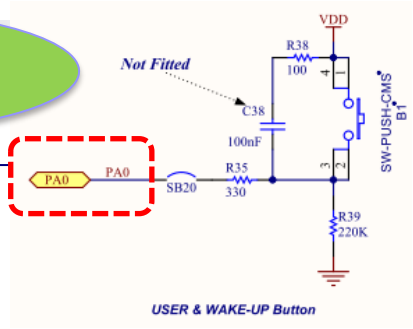
User manual

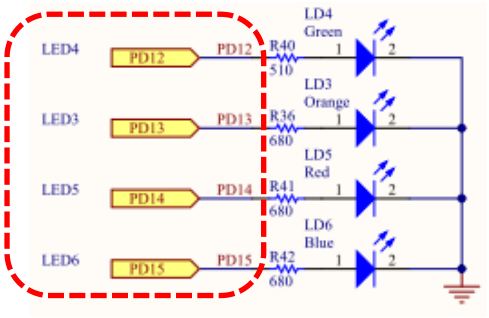
Description of STM32F4 HAL and low-layer drivers

31.2.4 IO operation functions

This section contains the following APIs:

- *HAL_GPIO_ReadPin()*
- *HAL_GPIO_WritePin()*
- *HAL_GPIO_TogglePin()*
- *HAL_GPIO_LockPin()*
- *HAL_GPIO_EXTI_IRQHandler()*
- *HAL_GPIO_EXTI_Callback()*



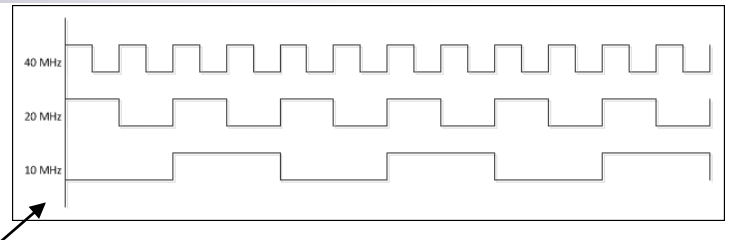


https://github.com/LAPSyLAB/STM32F4_Discovery_VIN_Projects/tree/main/STM32_USB_Key_LED

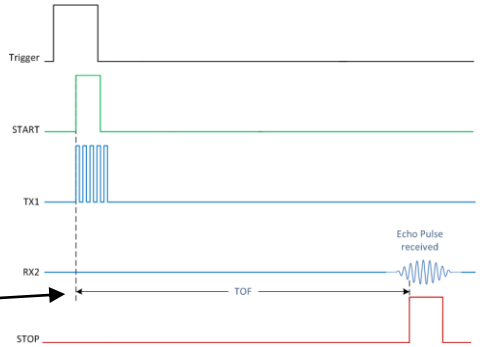
VIN projekt - VP6: STM32F4 VIN Demo

- Osvežitev: STM32F4
- CubeIDE projekt STM32F4 in V/I naprave :
  - CubeIDE projekt, GPIO in VCOM port
  - PWM - LED dimmer, brenčač
  - SPI - LIS3DSH pospeškomer
  - I2C - CS43L22 zvočni čip
  - ADC
- Sledenje („tracing“) - CubeMonitor, osciloskop

Timer - Counter (časovnik - števec)

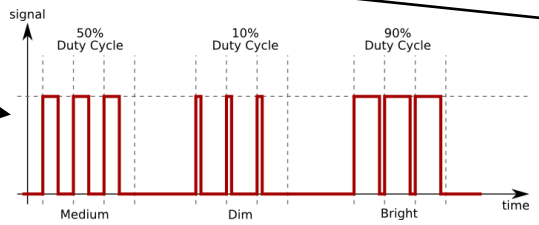


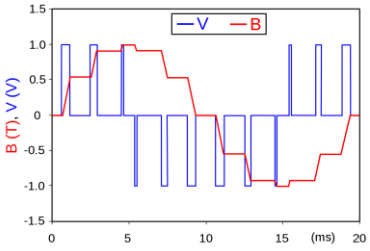
- Običajno več enakovrednih kanalov
- Uporabni za
  - štetje dogodkov (Capture)
  - tvorjenje časovnih signalov (Waveform)
  - zakasnitve (DELAY s časovnikom !)
  - merjenje intervalov
  - periodične prekinitve
  - tvorba signalov s pulzno širinsko modulacijo (PWM)



Variacije „Duty Cycle“:

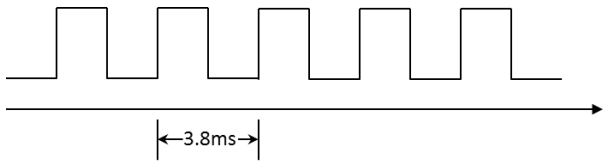
- LED „dimmer“
- krmiljenje hitrosti motorjev
- „enostavni“ DAC – povprečje
- kodiranje podatkov





Variacije periode:

- krmiljenje servo motorjev
- približek sinusnih tonov (50% duty)
  - primer: nota C2 = 262Hz, perioda T=1/262=3.8ms
  - izhod: brenčač



STM32F4 – PWM signali za LED diode (LED dimmer)

HAL - C

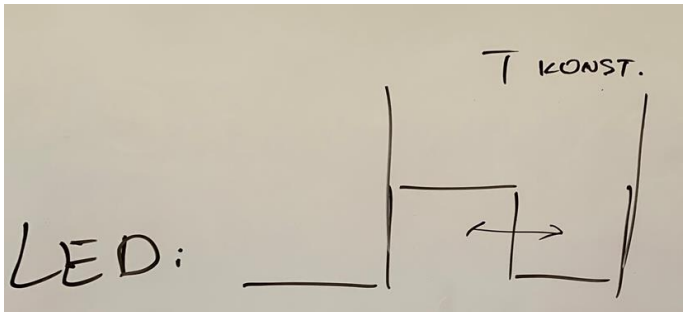
STM32F4

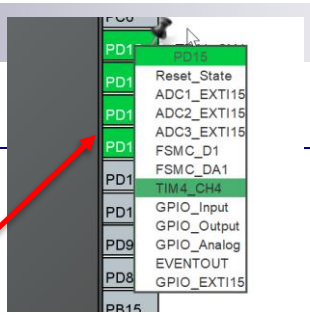
CubeMX :

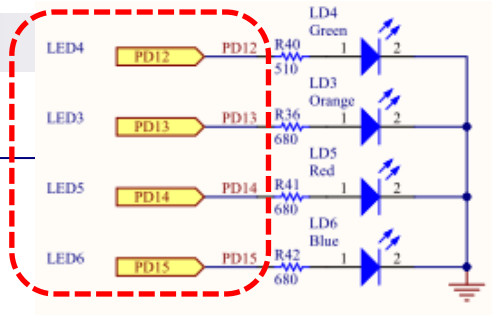
1. New project -> STM32 Project -> Board -> 407DISC1
2. CubeMX: Spremeniti USB Host v USB Device :
   - Connectivity -> USB_OTG_FS -> Mode v Device Only
   - Middleware -> DEVICE_USB in Class Virtual Com Port

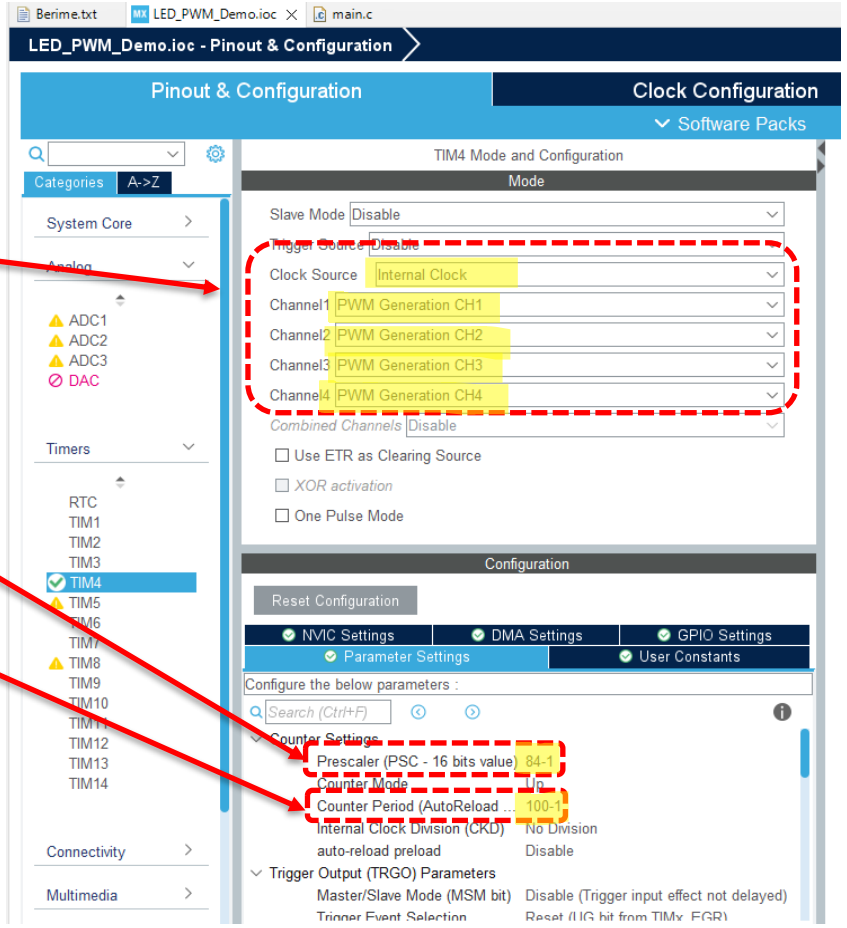
3. Spremeniti pine PD12-PD15 (LED diode) v TIM4_CH1-4
   - tim4 Vse kanale spremeniti na PWM Generation CH1-4

4. Clock :
   - Ura števca = 1 MHz
     - Prescaler (PSC - 16 bits value) = 84-1 = 83 (clock 1Mhz)
     - (PSC - 16 bits value) must be between 0 and 65 535
   - Perioda štetja je 100 (duty cycle pa lahko 0-100)
     - Counter Period = 100-1 = 99
     - (AutoReload Register - 16 bits value)







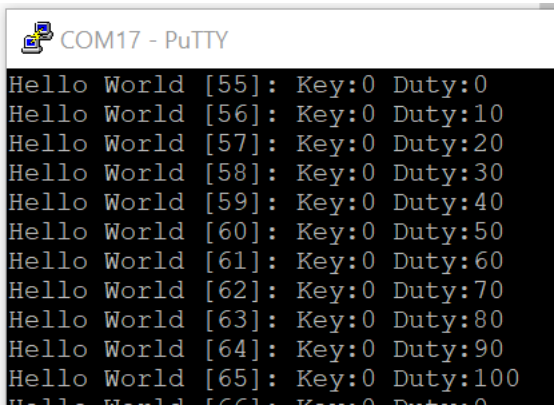


Osnovni projekt CubeIDE – GPIO – PWM, LED diode
HAL - C

```
/* USER CODE BEGIN PV */
#define BUFSIZE 256
char SendBuffer[BUFSIZE];
int Counter;
int KeyState=0;
int Duty=0;
/* USER CODE END PV */

/* USER CODE BEGIN 2 */

 HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_1);
 HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_2);
 HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_3);
 HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_4);

/* USER CODE END 2 */
/* Infinite loop */
 /* USER CODE BEGIN WHILE */
 while (1)
 {
       htim4.Instance->CCR1 = Duty;
       htim4.Instance->CCR2 = 100-Duty;
       htim4.Instance->CCR3 = Duty;
       htim4.Instance->CCR4 = 100-Duty;


       KeyState = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0);


       snprintf(SendBuffer,BUFSIZE,"Hello World [%d]: Key:%d Duty:%d\r\n",Counter++,KeyState,Duty);
       CDC_Transmit_FS(SendBuffer,strlen(SendBuffer));
       /* USER CODE END WHILE */

   /* USER CODE BEGIN 3 */
       Duty = (Duty + 10) ;
       if (Duty > 100 )
       Duty = 0;

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

CubeMX :

1. New project -> STM32 Project -> Board -> 407DISC1
2. CubeMX: Spremeniti USB Host v USB Device :
   - Connectivity -> USB_OTG_FS -> Mode v Device Only
   - Middleware -> DEVICE_USB in Class Virtual Com Port
3. Spremeniti pine PD12-PD15 (LED diode) v TIM4_CH1-4
   - tim4 Vse kanale spremeniti na PWM Generation CH1-4
4. Clock :
   - Ura števca = 1 MHz
     - Prescaler (PSC - 16 bits value) = 84-1 = 83 (clock 1Mhz)
     - (PSC - 16 bits value) must be between 0 and 65 535
   - Perioda štetja je 100 (duty cycle pa lahko 0-100)
     - Counter Period = 100-1 = 99
     - (AutoReload Register - 16 bits value)



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

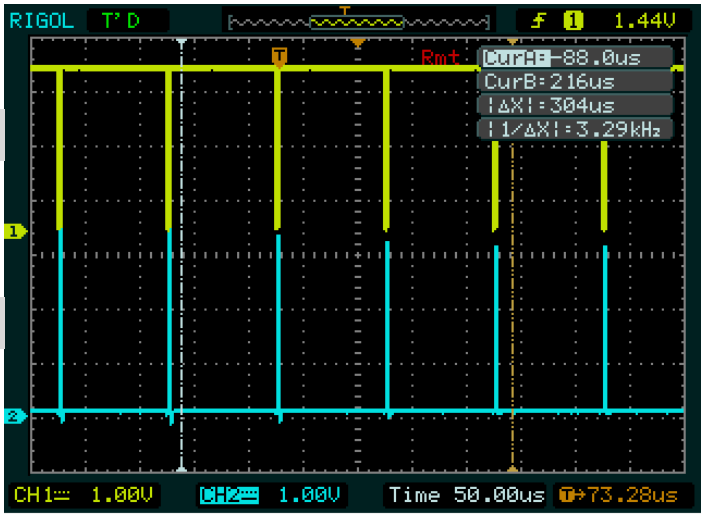
Osnovni projekt CubeIDE – GPIO – PWM, LED diode

HAL - C

```
 /* USER CODE BEGIN PV */
 #define BUFSIZE 256
 char SendBuffer[BUFSIZE];
 int Counter;
 int KeyState=0;
 int Duty=0;
 /* USER CODE END PV */
  /* USER CODE BEGIN 2 */

  HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_1);
  HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_2);
  HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_3);
  HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_4);

  /* USER CODE END 2 */
/* Infinite loop */
 /* USER CODE BEGIN WHILE */
 while (1)
 {
       htim4.Instance->CCR1 = Duty;
       htim4.Instance->CCR2 = 100-Duty;
       htim4.Instance->CCR3 = Duty;
       htim4.Instance->CCR4 = 100-Duty;


       KeyState = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0);


       snprintf(SendBuffer,BUFSIZE,"Hello World [%d]: Key:%d Duty:%d\r\n",Counter++,KeyState,Duty);
       CDC_Transmit_FS(SendBuffer,strlen(SendBuffer));
       /* USER CODE END WHILE */

   /* USER CODE BEGIN 3 */
       Duty = (Duty + 10) ;
       if (Duty > 100 )
       Duty = 0;

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

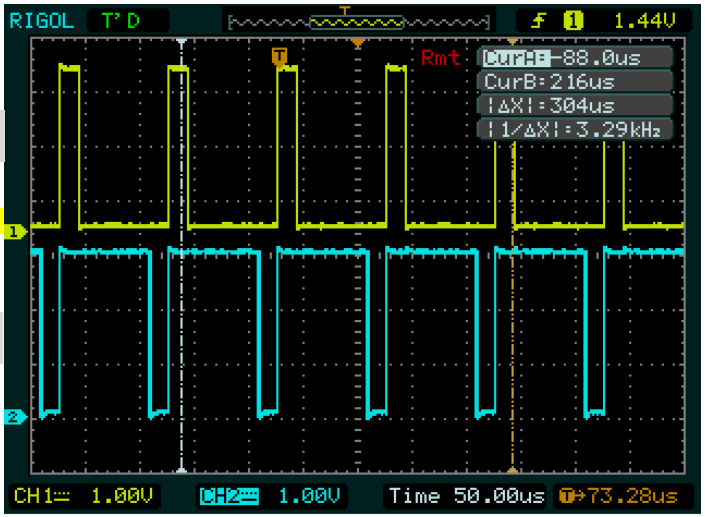
Max Duty

Min Duty



Min Duty

Max Duty



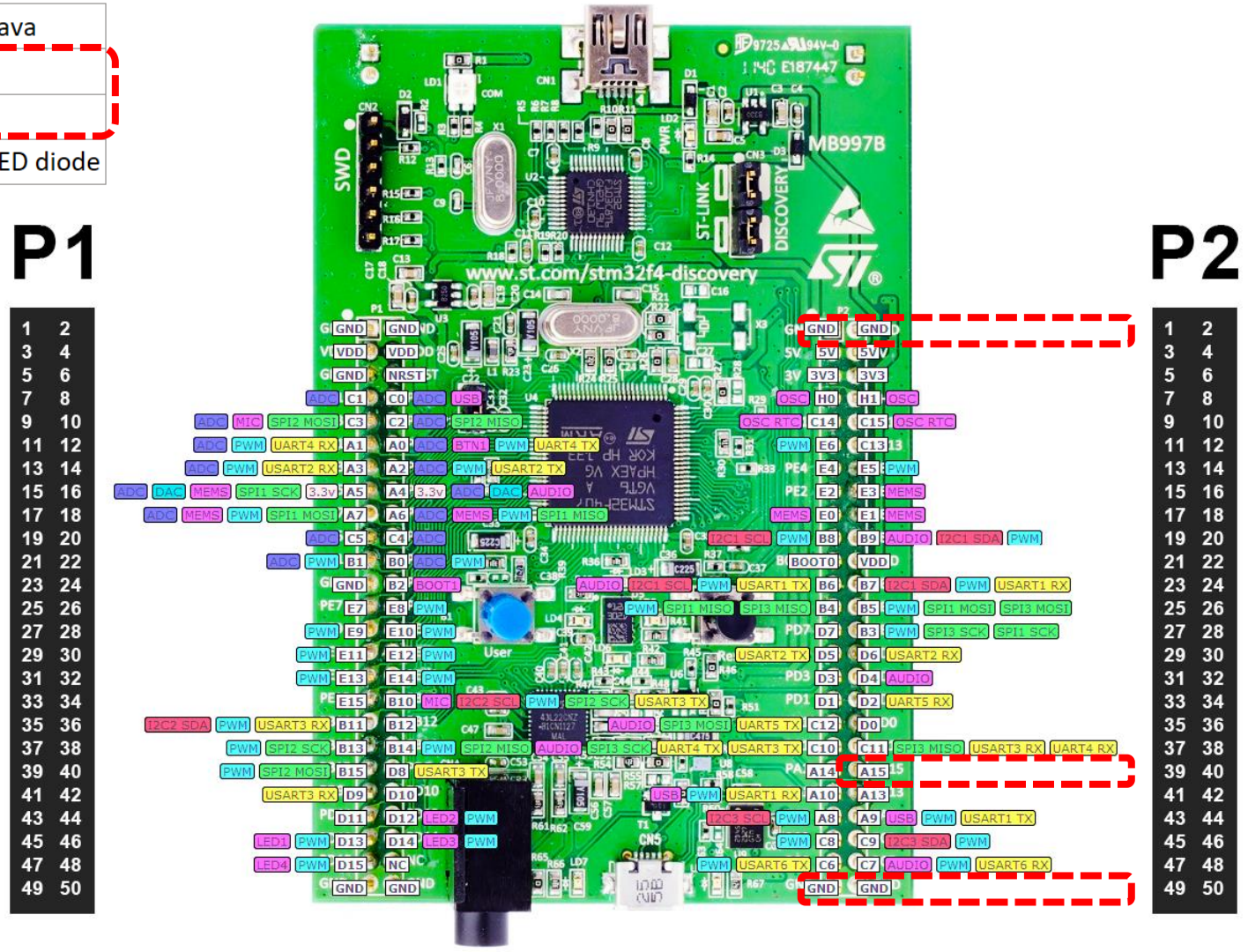
https://github.com/LAPSyLAB/STM32F4_Discovery_VIN_Projects/tree/main/LED_PWM_Demo

Vezava brenčača

Priključitev na STM32 : 1 x PWM izhod za brenčač (PA15), 1 x GND

Testno vezje (primer) :

GPIO	Vrsta	Povezava
PA15	Brencac	+
GND	Brencac	-
PD12-PD15	Dig. Izhodi	vgr. LED diode



STM32F4 – PWM signali/melodija za brenčača (Buzzer)

HAL - C

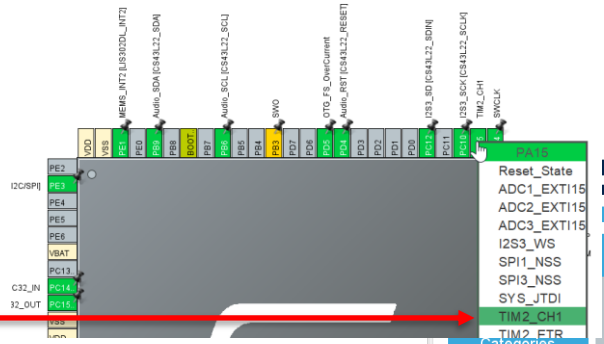
Brencac se prikljuci na **PA15 (TIM2->CH1) in GND**

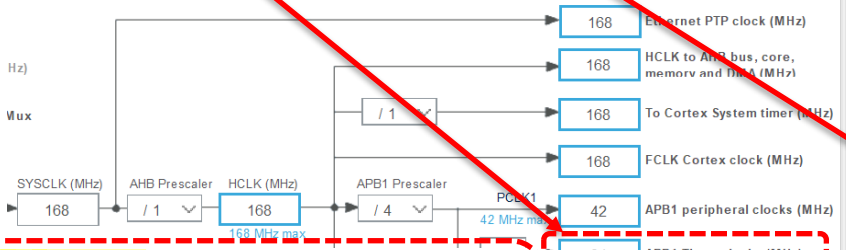
CubeMX :
1. New project -> STM32 Project -> Board -> 407DISC1
2. CubeMX: Spremeniti USB Host v USB Device :
Connectivity -> USB_OTG_FS -> Mode v Device Only
Middleware -> DEVICE_USB in Class Virtual Com Port

3. Spremeniti pin PA15 v TIM2->CH1
tim2 kanal 1 spremeniti na PWM Generation CH1

4. Clock :
Ura števca = 1 MHz
Prescaler (PSC - 16 bits value) = **84-1 = 83 (clock 1Mhz)**
Perioda štetja se bo določala glede na noto (duty cycle je vedno 50%)
Counter Period (AutoReload Register - 16 bits value)
ARR = 1000000 (ura števca) / Frekv.note[Hz]
CCR1 bo vedno ARR/2 (50% duty cycle)

Več informacij : BeriMe.txt





Nota:

```
ARR_period = (int)(1000000/NoteFreq);  //Already prescaled to 1 MHz
setPWM(htim2, TIM_CHANNEL_1, ARR_period, ARR_period/2);

Delaymsecs = noteDurations[melodyIndex][noteIndex] * melodySlowfactor[melodyIndex];

snprintf (SendBuffer,BUFSIZE,"Melody[%d],Note #%d F=%d Hz Duration:%d ms| ARR=%d CCR
CDC_Transmit_FS(SendBuffer,strlen(SendBuffer));

HAL_Delay(Delaymsecs);
```

STM32F4

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

STM32F4 – PWM signal/ton za brenčača (Buzzer)

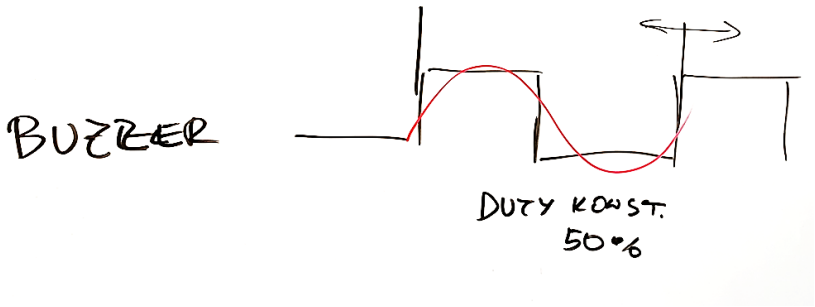
HAL - C

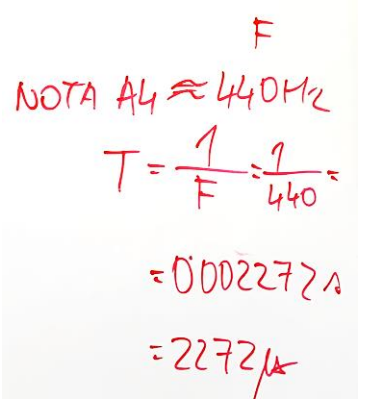
```
/* USER CODE BEGIN PV */
…
int NoteFreq = 440 ; // Note A4 = 440 Hz
int NotePeriod; //Already prescaled to 1 MHz

/* USER CODE END PV */
```

```
/* USER CODE BEGIN 2 */
…
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
/* USER CODE END 2 */
```

```
/* USER CODE BEGIN 0 */
void setPWM(TIM_HandleTypeDef timer, uint32_t channel, uint16_t period, uint16_t pulse)
{
HAL_TIM_PWM_Stop(&timer, channel); // stop generation of pwm
TIM_OC_InitTypeDef sConfigOC;
timer.Init.Period = period; // set the period duration
HAL_TIM_PWM_Init(&timer); // reinititialise with new period value
sConfigOC.OCMode = TIM_OCMODE_PWM1;
sConfigOC.Pulse = pulse; // set the pulse duration
sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
HAL_TIM_PWM_ConfigChannel(&timer, &sConfigOC, channel);
HAL_TIM_PWM_Start(&timer, channel); // start pwm generation
}
/* USER CODE END 0 */
```

```
/* USER CODE BEGIN WHILE */
while (1)
{
htim4.Instance->CCR1 = Duty;
htim4.Instance->CCR2 = 100-Duty;
htim4.Instance->CCR3 = Duty;
htim4.Instance->CCR4 = 100-Duty;

KeyState = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0);

NotePeriod = (int)(1000000/NoteFreq); //Already prescaled to 1 MHz
setPWM(htim2, TIM_CHANNEL_1, NotePeriod, NotePeriod/2);

snprintf(SendBuffer,BUFSIZE,"Hello World [%d]: Key:%d Duty:%d PWM-Freq:%d PWM-Period:%d\r\n",Counter++,KeyState,Duty,NoteFreq,NotePeriod);
```





STM32F4 – PWM signali/melodija za brenčača (Buzzer)

HAL - C

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

melodyCount = sizeof(melodySizes)/ sizeof(uint32_t);

for(melodyIndex = 0; melodyIndex < melodyCount; melodyIndex++)
{
        for(noteIndex = 0; noteIndex < melodySizes[melodyIndex]; noteIndex++)
          {
              //    buzzerSetNewFrequency(melody[melodyIndex][noteIndex]);
                NoteFreq = melody[melodyIndex][noteIndex];
                if (NoteFreq == 0) NoteFreq = 1;

                ARR_period = (int)(1000000/NoteFreq);  //Already prescaled to 1 MHz
                setPWM(htim2, TIM_CHANNEL_1, ARR_period, ARR_period/2);

                Delaymsecs = noteDurations[melodyIndex][noteIndex] * melodySlowfactor[melodyIndex];

                HAL_Delay(Delaymsecs);
          }
        snprintf (SendBuffer,BUFSIZE,"\r\n\r\nEnd of Melody[%d]\r\n\r\n",melodyIndex);
            CDC_Transmit_FS(SendBuffer,strlen(SendBuffer));

}
```

Melody.h:

```
//###############**"Crazy Frog" song of Crazy frog album**###############//
const uint32_t   CrazyFrog_notes[] = {
  NOTE_D4, 0, NOTE_F4, NOTE_D4, 0, NOTE_D4, NOTE_G4, NOTE_D4, NOTE_C4,
  NOTE_D4, 0, NOTE_A4, NOTE_D4, 0, NOTE_D4, NOTE_AS4, NOTE_A4, NOTE_F4,
  NOTE_D4, NOTE_A4, NOTE_D5, NOTE_D4, NOTE_C4, 0, NOTE_C4, NOTE_A3, NOTE_E4,
NOTE_D4,
  0,NOTE_D4,NOTE_D4
};

const uint32_t   CrazyFrog_durations[] = {
  8, 8, 6, 16, 16, 16, 8, 8, 8,
  8, 8, 6, 16, 16, 16, 8, 8, 8,
  8, 8, 8, 16, 16, 16, 16, 8, 8, 2,
  8,4,4
};
//############End of Crazy Frog##############//
```

Melody.h:

```
const uint32_t* melody[] ={marioMelody, secondMelody,
Titanic_Melody,Pirates_notes,CrazyFrog_notes};
const uint32_t* noteDurations[] = {marioDuration, secondDuration,
Titanic_duration,Pirates_durations,CrazyFrog_durations};
const uint16_t melodySlowfactor[] ={15, 30, 20, 20, 20};

const uint32_t melodySizes[] ={sizeof(marioMelody)/sizeof(uint32_t),
sizeof(secondDuration)/sizeof(uint32_t),
sizeof(Titanic_duration)/sizeof(uint32_t),
sizeof(Pirates_durations)/sizeof(uint32_t),
sizeof(CrazyFrog_durations)/sizeof(uint32_t)};
```

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

STM32F4, H7 – PWM signali/melodija za brenčača (Buzzer)

HAL - C

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

melodyCount = sizeof(melodySizes)/ sizeof(uint32_t);

for(melodyIndex = 0; melodyIndex < melodyCount; melodyIndex++)
{
        for(noteIndex = 0; noteIndex < melodySizes[melodyIndex]; noteIndex++)
          {
              //    buzzerSetNewFrequency(melody[melodyIndex][noteIndex]);
                NoteFreq = melody[melodyIndex][noteIndex];
                if (NoteFreq == 0) NoteFreq = 1;

                ARR_period = (int)(1000000/NoteFreq);  //Already prescaled to 1 MHz
                setPWM(htim2, TIM_CHANNEL_1, ARR_period, ARR_period/2);

                Delaymsecs = noteDurations[melodyIndex][noteIndex] * melodySlowfactor[melodyIndex];

                HAL_Delay(Delaymsecs);
          }
        snprintf (SendBuffer,BUFSIZE,"\r\n\r\nEnd of Melody[%d]\r\n\r\n",melodyIndex);
            CDC_Transmit_FS(SendBuffer,strlen(SendBuffer));

}
```

Melody.h:

```
const uint32_t* melody[] ={marioMelody, secondMelody,
Titanic_Melody,Pirates_notes,CrazyFrog_notes};
const uint32_t* noteDurations[] = {marioDuration, secondDuration,
Titanic_duration,Pirates_durations,CrazyFrog_durations};
const uint16_t melodySlowfactor[] ={15, 30, 20, 20, 20};

const uint32_t melodySizes[] ={sizeof(marioMelody)/sizeof(uint32_t),
sizeof(secondDuration)/sizeof(uint32_t),
sizeof(Titanic_duration)/sizeof(uint32_t),
sizeof(Pirates_durations)/sizeof(uint32_t),
sizeof(CrazyFrog_durations)/sizeof(uint32_t)};
```

Melody.h:

```
// Zapisi not v [Hz]
#define NOTE_C4  262
#define NOTE_CS4 277
#define NOTE_D4  294
#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


// Zapisi melodij v notah [Hz] in trajanju
//###############**"Crazy Frog" song of Crazy frog
album**###############//
const uint32_t   CrazyFrog_notes[] = {
  NOTE_D4, 0, NOTE_F4, NOTE_D4, 0, NOTE_D4, NOTE_G4,
NOTE_D4, NOTE_C4,
  NOTE_D4, 0, NOTE_A4, NOTE_D4, 0, NOTE_D4, NOTE_AS4,
NOTE_A4, NOTE_F4,
  NOTE_D4, NOTE_A4, NOTE_D5, NOTE_D4, NOTE_C4, 0,
NOTE_C4, NOTE_A3, NOTE_E4, NOTE_D4,
  0,NOTE_D4,NOTE_D4
};

const uint32_t   CrazyFrog_durations[] = {
  8, 8, 6, 16, 16, 16, 8, 8, 8,
  8, 8, 6, 16, 16, 16, 8, 8, 8,
  8, 8, 8, 16, 16, 16, 16, 8, 8, 2,
  8,4,4
};
//############End of Crazy Frog##############//
```

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

VIN projekt - VP6: STM32F4 VIN Demo

- Osvežitev: STM32F4
- CubeIDE projekt STM32F4 in V/I naprave :
  - CubeIDE projekt, GPIO in VCOM port
  - PWM - LED dimmer, brenčač
  - SPI - LIS3DSH pospeškomer
  - I2C - CS43L22 zvočni čip
  - ADC
- Sledenje („tracing“) - CubeMonitor, osciloskop

5 Digital main blocks

5.1 State machine

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

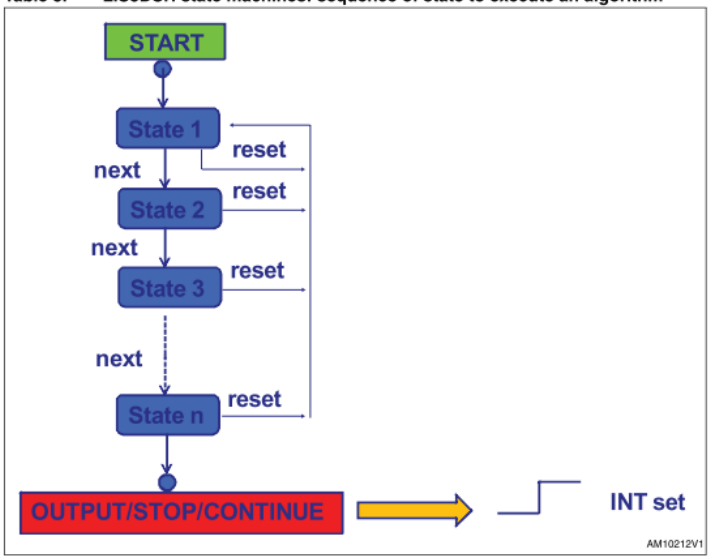
The program is made up of a set of instructions that define the transition to successive states. Conditional branches are possible.

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

	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

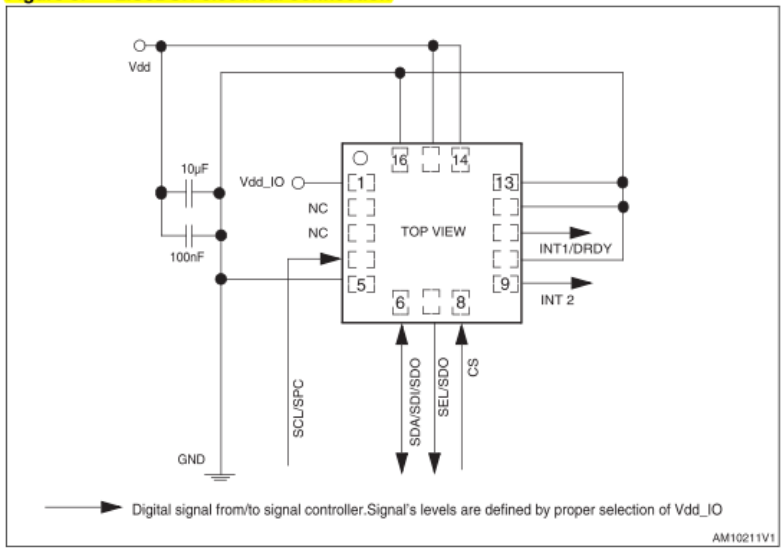
Parameter	I²C standard mode (1) Min.	I²C standard mode (1) Max.	I² fast mode (1) Min	I² fast mode (1) Max.	Unit
SCL clock frequency	0	100	0	400	kHz

Table 7. Absolute maximum ratings

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

Application hints

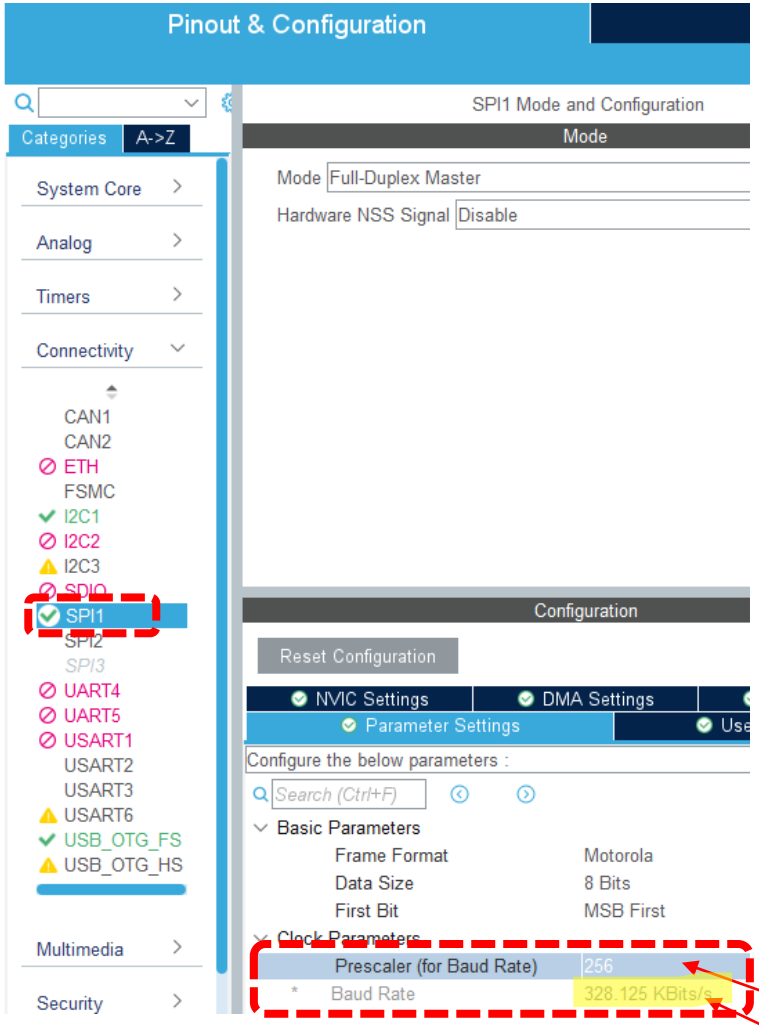
Figure 5. LIS3DSH electrical connection

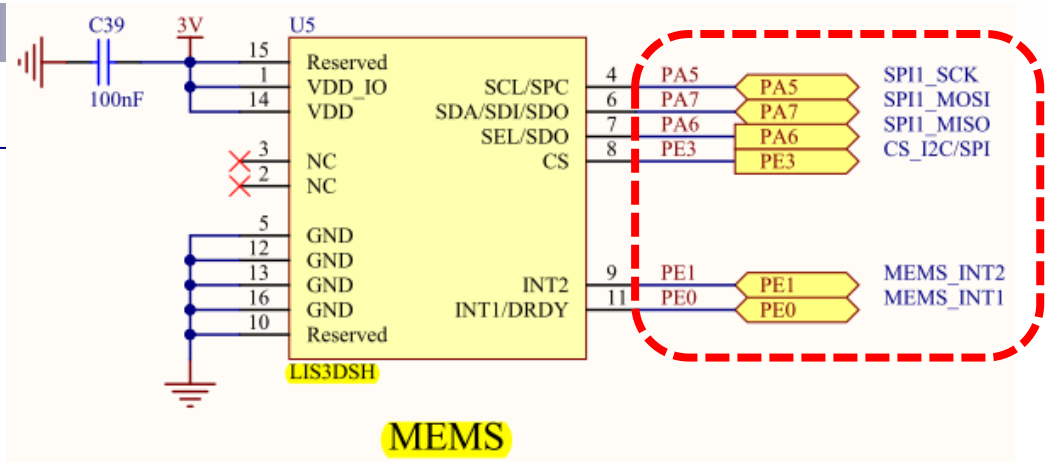


https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples/blob/main/STM32F407_Discovery_kit/LIS3DSH.pdf

VP 6 - STM32 CubeIDE, SPI in LIS3DSH

CubeMX nastavitve :

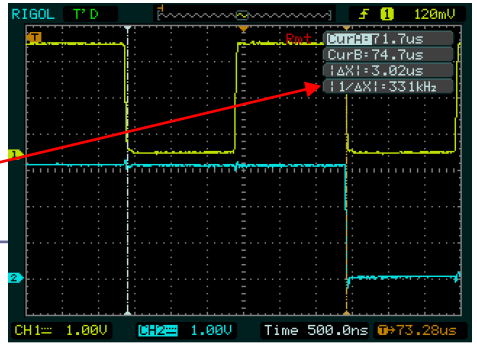




spi.c:

```
/* 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)



SPI - serial peripheral interface

Subject to general operating conditions for Vdd and Top.

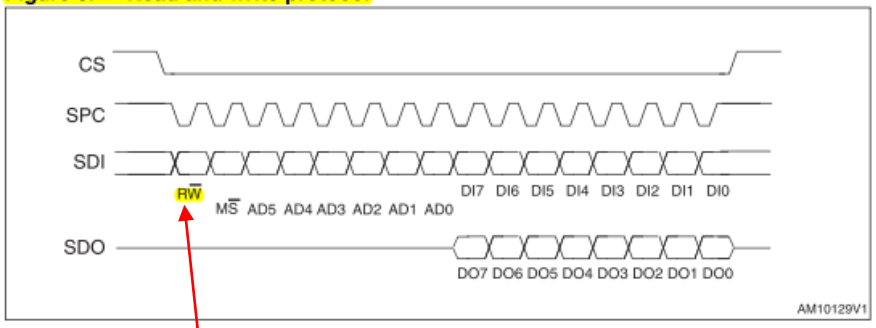
SPI slave timing values

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

Table 7. Absolute maximum ratings

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

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).

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

```
// 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);

// Write to CTRL register (enable 3 axes meaurements on 25Hz)
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);
```

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

Table 23. CTRL_REG4 register description

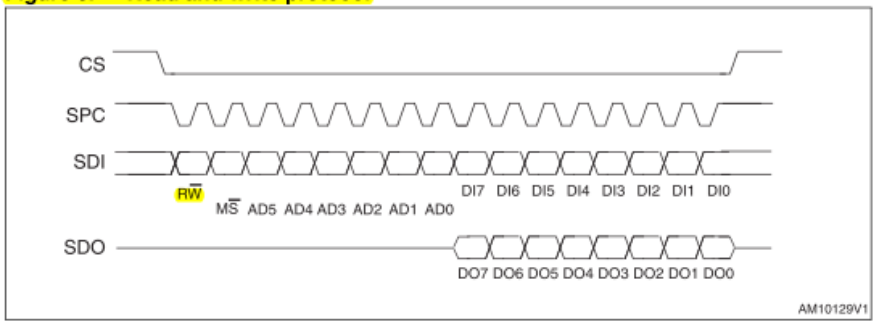
ODR 3:0	Output data rate and power mode selection. Default value:0000 (see *Table 24*)
BDU	Block data update. Default value:0 0:continuos update,1:output registers not updated until MSB and LSB reading)
Zen	Z axis enable. Default value:1 (0:Z axis disabled; 1:Z axis enabled)
Yen	Y axis enable. Default value:1 (0:Y axis disabled; 1:Y axis enabled)
Xen	X axis enable. Default value:1 0=X axis disabled; 1=X axis enabled

Table 24. CTRL4 ODR configuration

ODR3	ODR2	ODR1	ODR0	ODR selection
0	0	0	0	Power down
0	0	0	1	3.125 Hz
0	0	1	0	6.25 Hz
0	0	1	1	12.5 Hz
0	1	0	0	25 Hz

https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples/blob/main/Documentation/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).

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	Register address 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		

8.23 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

```
// Read x,y,z axes
  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);
  AccelX = indata[1];

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

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

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

VP 6 - STM32 CubeIDE, SPI in LIS3DSH

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)
 {

// Read x,y,z axes
  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);
  AccelX = indata[1];

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

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

…

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

   /* USER CODE END WHILE */
```

https://github.com/LAPSyLAB/STM32F4_Discovery_VIN_Projects/tree/main/STM32_SPI_LIS302DL_Basic

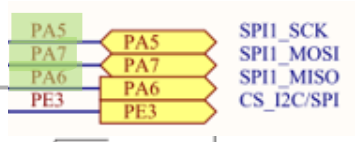
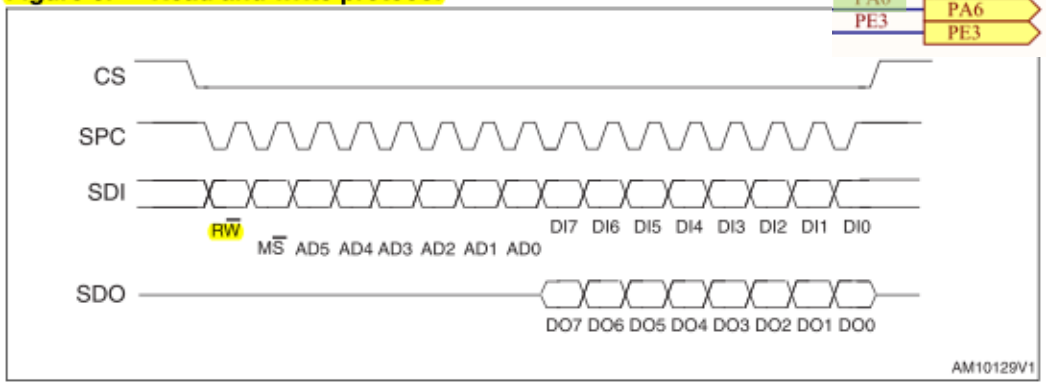


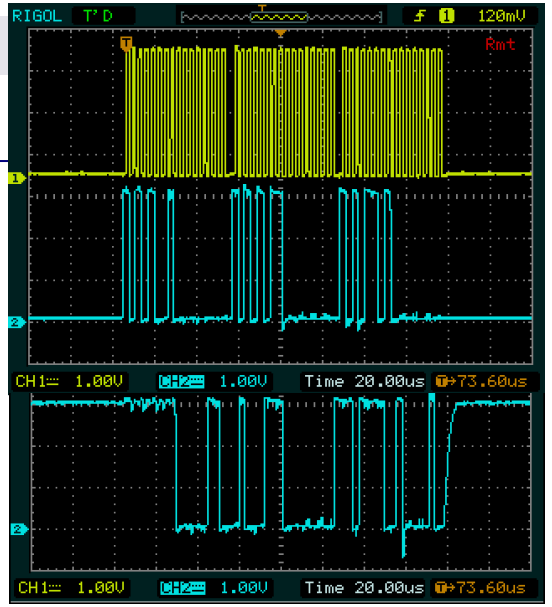
Figure 6. Read and write protocol



SCK

MOSI

MISO

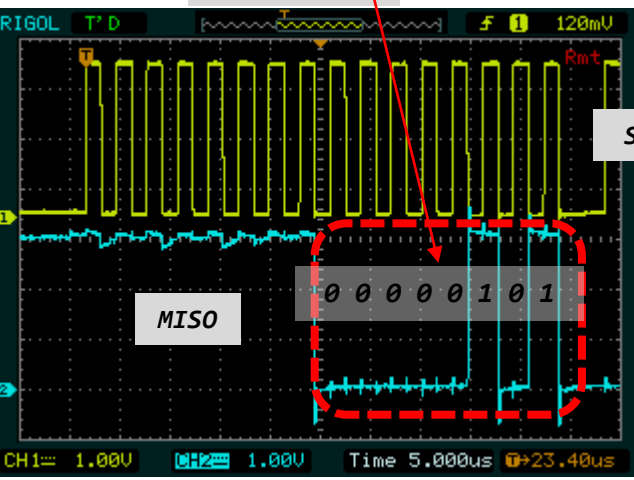


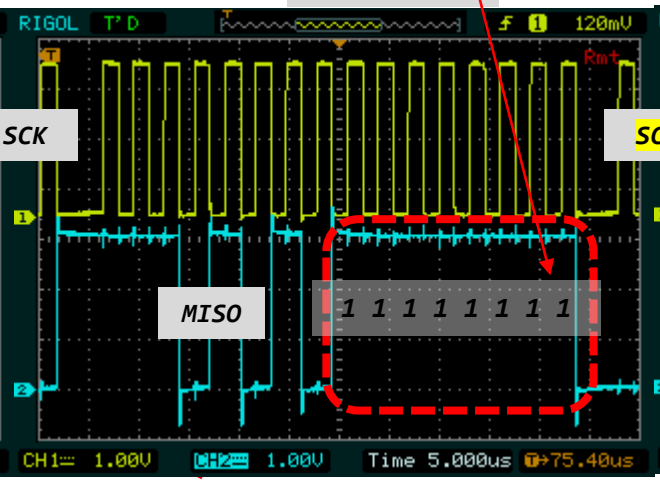

```
Hello World [3530]: Key:0000 Accel[ID:00] X:0005 Y:-1 Z:0066
Hello World [3531]: Key:0000 Accel[ID:00] X:0005 Y:-1 Z:0067
```

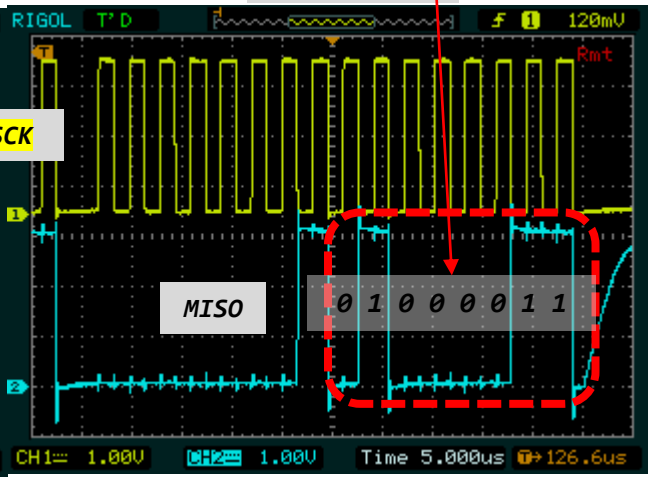
X-Accel: 5

Y-Accel: -1

Y-Accel: 67







VIN projekt - VP6: STM32F4 VIN Demo

- Osvežitev: STM32F4
- CubeIDE projekt STM32F4 in V/I naprave :
  - CubeIDE projekt, GPIO in VCOM port
  - PWM - LED dimmer, brenčač
  - SPI - LIS3DSH pospeškomer
  - I2C - CS43L22 zvočni čip
  - ADC
- Sledenje („tracing“) - CubeMonitor, osciloskop

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.

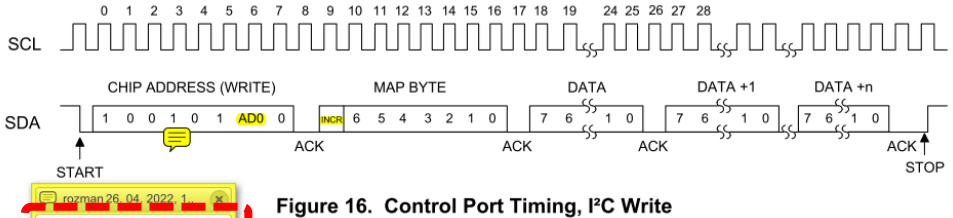


Figure 16. Control Port Timing, I²C Write

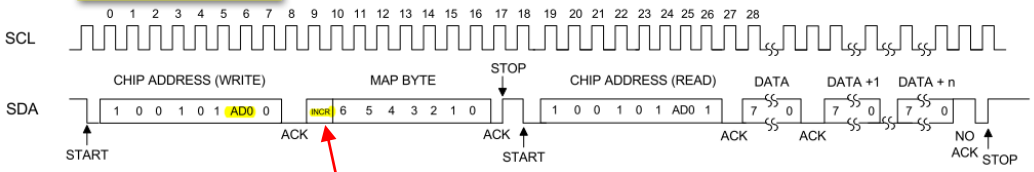


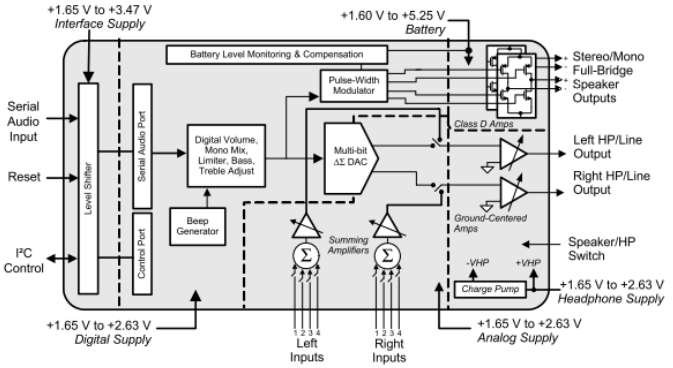
Figure 17. Control Port Timing, I²C Read

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/Documentation/STM32F407_Discovery_kit/CS43L22_F2.pdf

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*

UM1725
Contents

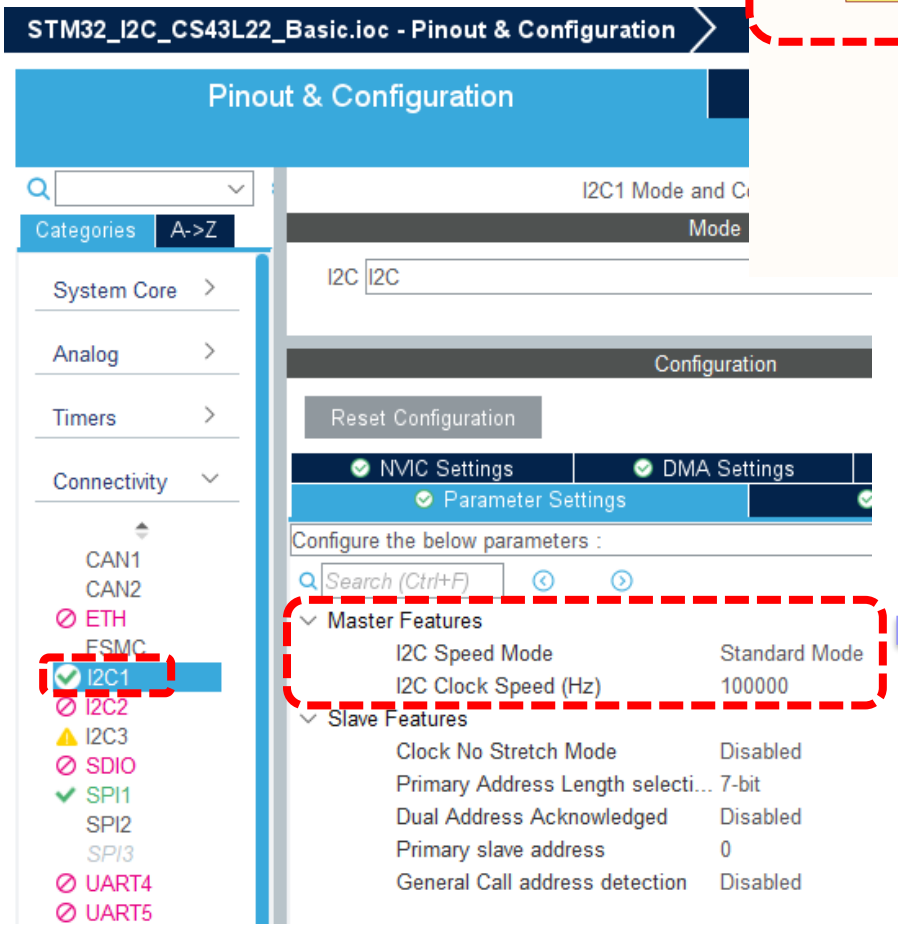
Contents

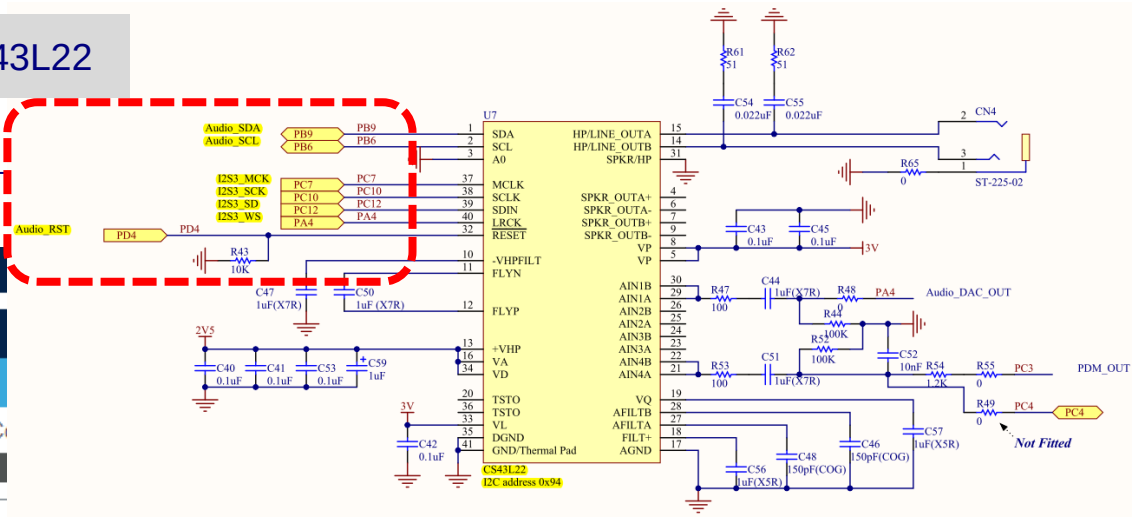
Lastni viri :

https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples

VP 6 - STM32 CubeIDE, I2C in CS43L22

CubeMX nastavitve
(I2C1 že nastavljen)





i2c.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 */

}
```

main.c : dodana koda

Spremenljivke

```
/* USER CODE BEGIN PV */
…
uint8_t ChipID;

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

Inicializacija

```
/* 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 */
```

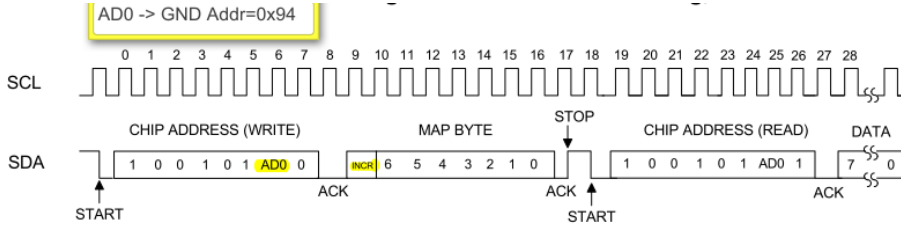


Figure 17. Control Port Timing, I²C Read

Glavna zanka

```
  /* 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 Duty:%d PWM-Freq:%d PWM-Period:%d Accel[ID:%02x]
X:%04d Y:%d Z:%04d
ChipID:%02x\r\n",Counter++,KeyState,Duty,NoteFreq,NotePeriod,lis_id,AccelX,AccelY,AccelZ,ChipID);
  CDC_Transmit_FS(SendBuffer,strlen(SendBuffer));

    /* USER CODE END WHILE */

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

VIN projekt - VP6: STM32F4 VIN Demo

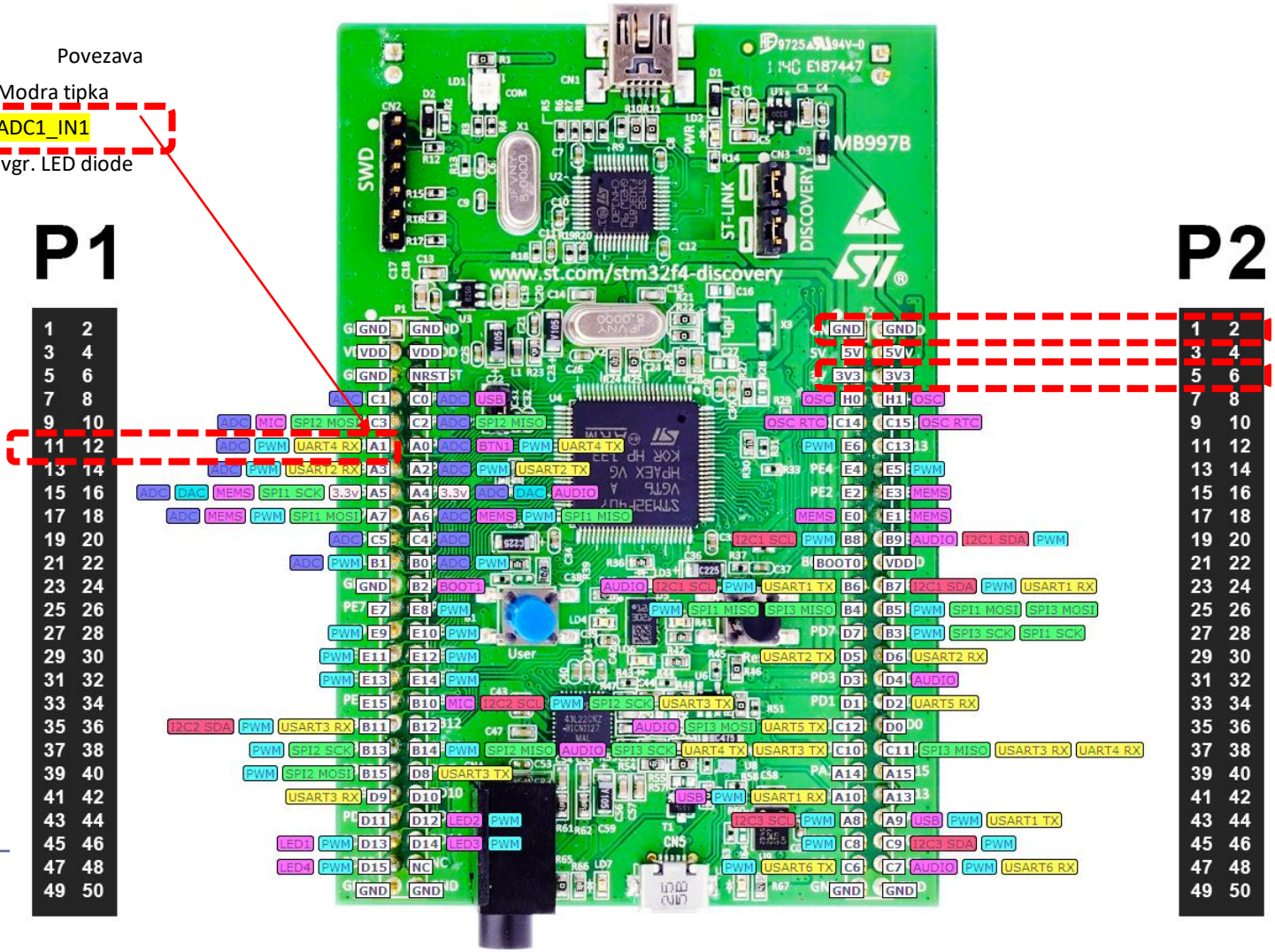
- Osvežitev: STM32F4
- CubeIDE projekt STM32F4 in V/I naprave :
  - CubeIDE projekt, GPIO in VCOM port
  - PWM - LED dimmer, brenčač
  - SPI - LIS3DSH pospeškomer
  - I2C - CS43L22 zvočni čip
  - ADC
- Sledenje („tracing“) - CubeMonitor, osciloskop

Breadboard vezava – STM32F4

Priključitev na STM32 : 3 x analogni, 1x digitalni vhod, 4x vgrajene LED diode

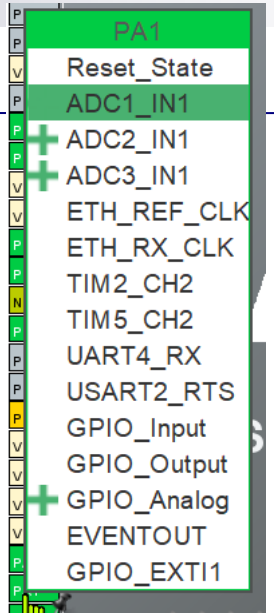
Testno vezje (primer) - STM32F4 :

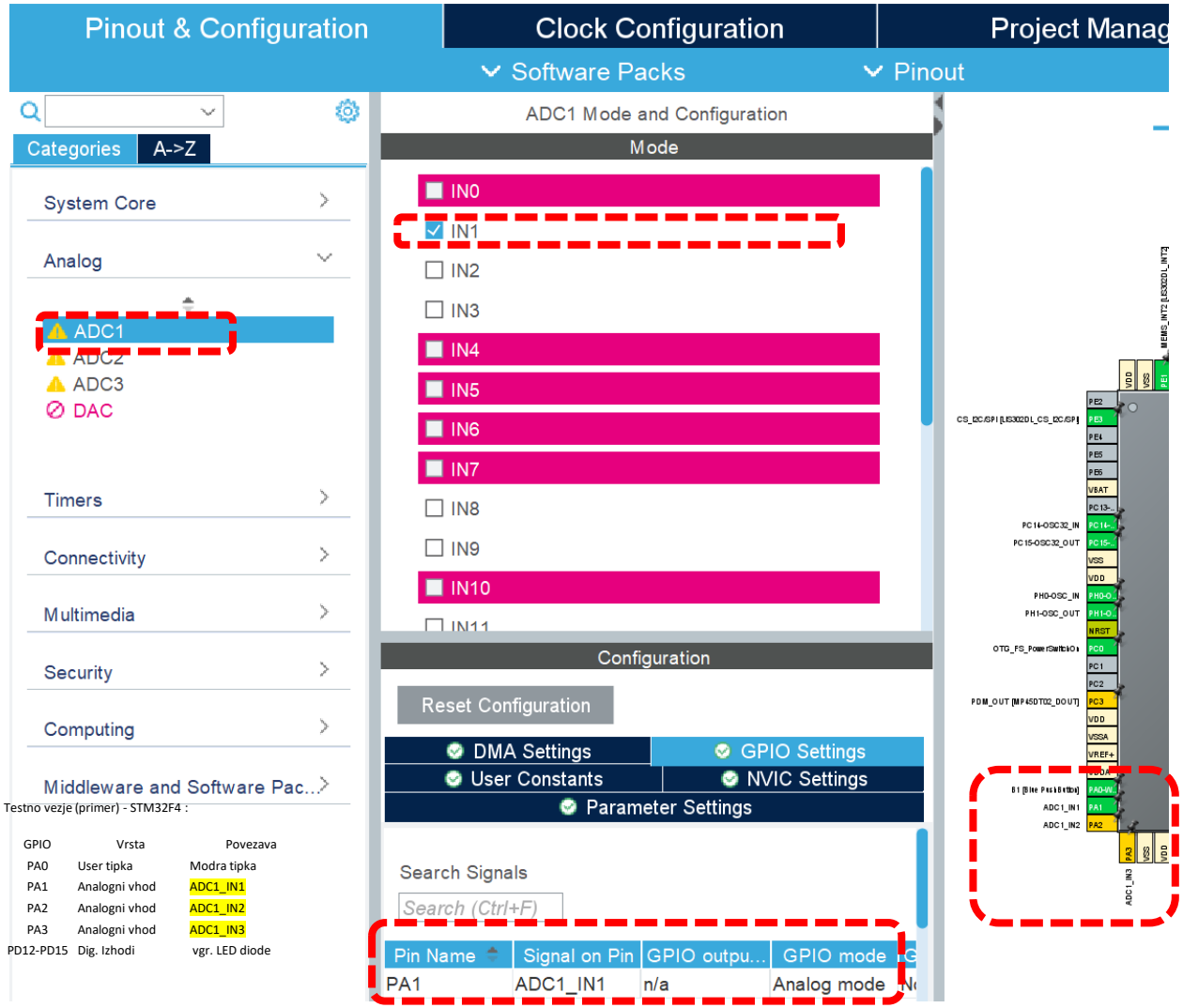
GPIO	Vrsta	Povezava
PA0	User tipka	Modra tipka
PA1	Analogni vhod	ADC1_IN1
PD12-PD15	Dig. Izhodi	vgr. LED diode



VIN projekt - VP 6 ADC

Konfiguracija 2: (PA1 ADC1_IN1)





Testno vezje (primer) - STM32F4 :

GPIO	Vrsta	Povezava
PA0	User tipka	Modra tipka
PA1	Analogni vhod	ADC1_IN1
PA2	Analogni vhod	ADC1_IN2
PA3	Analogni vhod	ADC1_IN3
PD12-PD15	Dig. Izhodi	vgr. LED diode

VIN projekt - VP 6 STM32-CubeIDE projekt, ADC

Program : za branje tipal in pošiljanje po USB Virtual COM Port

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

    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
    AnalogValue1 = HAL_ADC_GetValue(&hadc1);

    HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12);   // On-board LED

    HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_5);   //External LED on PB5
    KeyState = HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_4); //External Key on PB4

     snprintf(SendBuffer,BUFSIZE,"Hello World [%d]: Key:%d Duty:%d PWM-Freq:%d PWM-Period:%d
     Accel[ID:%02x] X:%04d Y:%d Z:%04d ChipID:%02x
     ADC1:%d\r\n",Counter++,KeyState,Duty,NoteFreq,NotePeriod,lis_id,AccelX,AccelY,AccelZ,ChipID,AnalogVal
     ue1);
     CDC_Transmit_FS(SendBuffer,strlen(SendBuffer));
   /* USER CODE END WHILE */

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

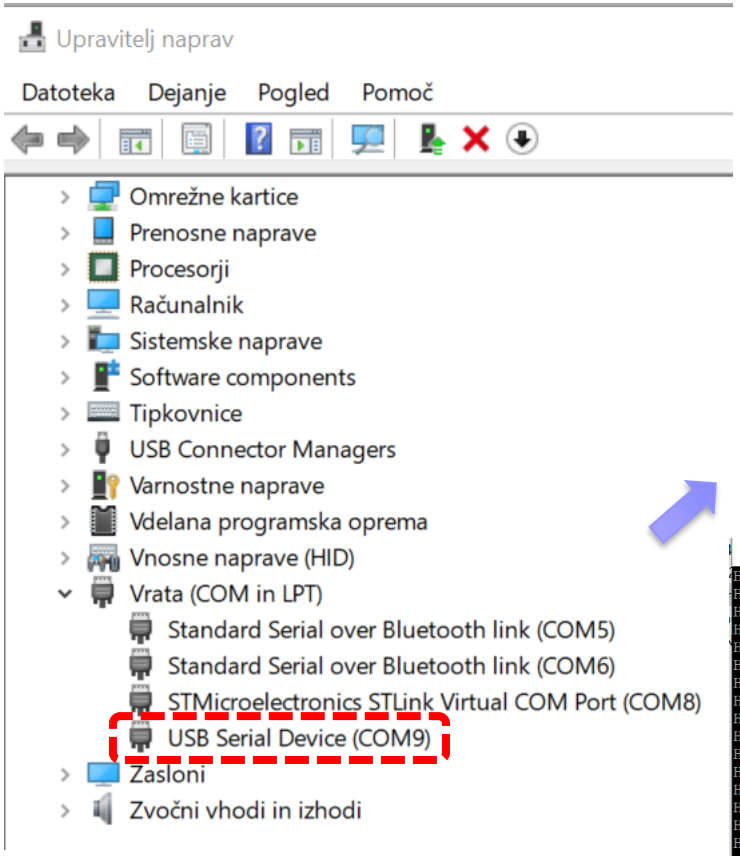
```
/* USER CODE BEGIN PV */
…

int AnalogValue1,AnalogValue2,AnalogValue3;
/* USER CODE END PV */
```

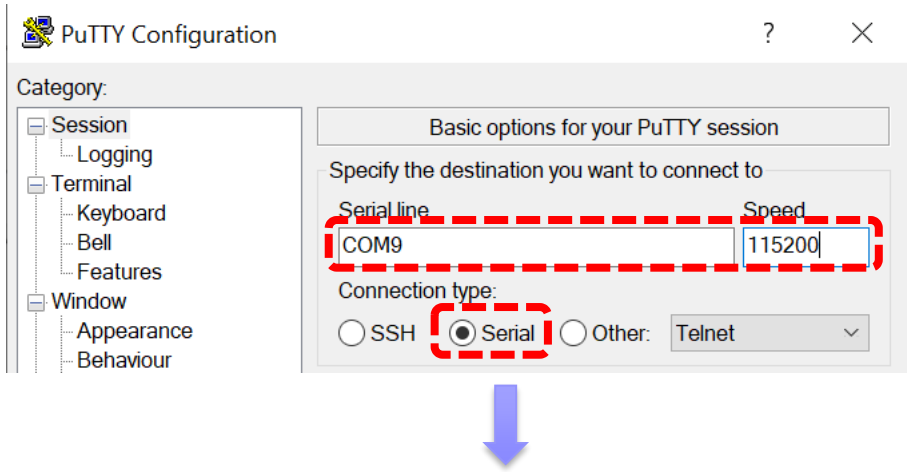
COM17 - PuTTY

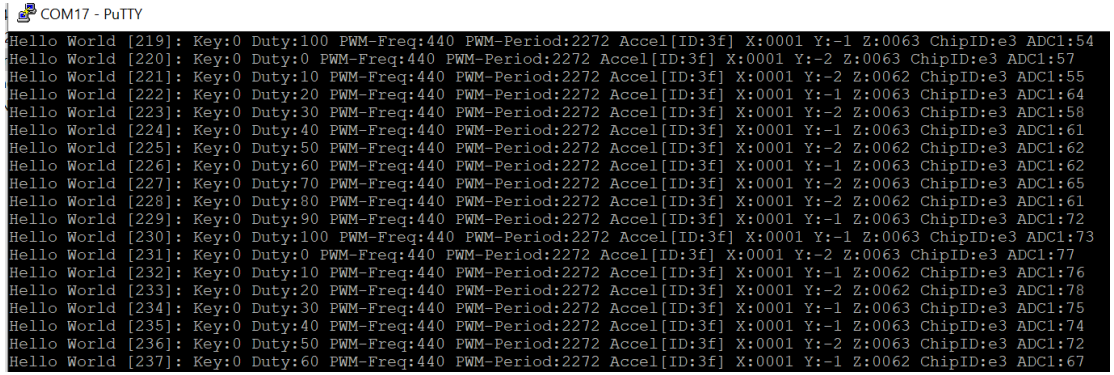
```
Hello World [219]: Key:0 Duty:100 PWM-Freq:440 PWM-Period:2272 Accel[ID:3f] X:0001 Y:-1 Z:0063 ChipID:e3 ADC1:54
Hello World [220]: Key:0 Duty:0 PWM-Freq:440 PWM-Period:2272 Accel[ID:3f] X:0001 Y:-2 Z:0063 ChipID:e3 ADC1:57
Hello World [221]: Key:0 Duty:10 PWM-Freq:440 PWM-Period:2272 Accel[ID:3f] X:0001 Y:-2 Z:0062 ChipID:e3 ADC1:55
Hello World [222]: Key:0 Duty:20 PWM-Freq:440 PWM-Period:2272 Accel[ID:3f] X:0001 Y:-1 Z:0063 ChipID:e3 ADC1:64
Hello World [223]: Key:0 Duty:30 PWM-Freq:440 PWM-Period:2272 Accel[ID:3f] X:0001 Y:-2 Z:0063 ChipID:e3 ADC1:58
Hello World [224]: Key:0 Duty:40 PWM-Freq:440 PWM-Period:2272 Accel[ID:3f] X:0001 Y:-1 Z:0063 ChipID:e3 ADC1:61
Hello World [225]: Key:0 Duty:50 PWM-Freq:440 PWM-Period:2272 Accel[ID:3f] X:0001 Y:-2 Z:0062 ChipID:e3 ADC1:62
Hello World [226]: Key:0 Duty:60 PWM-Freq:440 PWM-Period:2272 Accel[ID:3f] X:0001 Y:-1 Z:0063 ChipID:e3 ADC1:62
Hello World [227]: Key:0 Duty:70 PWM-Freq:440 PWM-Period:2272 Accel[ID:3f] X:0001 Y:-2 Z:0063 ChipID:e3 ADC1:65
Hello World [228]: Key:0 Duty:80 PWM-Freq:440 PWM-Period:2272 Accel[ID:3f] X:0001 Y:-2 Z:0062 ChipID:e3 ADC1:61
Hello World [229]: Key:0 Duty:90 PWM-Freq:440 PWM-Period:2272 Accel[ID:3f] X:0001 Y:-1 Z:0063 ChipID:e3 ADC1:72
Hello World [230]: Key:0 Duty:100 PWM-Freq:440 PWM-Period:2272 Accel[ID:3f] X:0001 Y:-1 Z:0063 ChipID:e3 ADC1:73
Hello World [231]: Key:0 Duty:0 PWM-Freq:440 PWM-Period:2272 Accel[ID:3f] X:0001 Y:-2 Z:0063 ChipID:e3 ADC1:77
Hello World [232]: Key:0 Duty:10 PWM-Freq:440 PWM-Period:2272 Accel[ID:3f] X:0001 Y:-1 Z:0062 ChipID:e3 ADC1:76
Hello World [233]: Key:0 Duty:20 PWM-Freq:440 PWM-Period:2272 Accel[ID:3f] X:0001 Y:-2 Z:0062 ChipID:e3 ADC1:78
Hello World [234]: Key:0 Duty:30 PWM-Freq:440 PWM-Period:2272 Accel[ID:3f] X:0001 Y:-1 Z:0063 ChipID:e3 ADC1:75
Hello World [235]: Key:0 Duty:40 PWM-Freq:440 PWM-Period:2272 Accel[ID:3f] X:0001 Y:-1 Z:0063 ChipID:e3 ADC1:74
Hello World [236]: Key:0 Duty:50 PWM-Freq:440 PWM-Period:2272 Accel[ID:3f] X:0001 Y:-2 Z:0063 ChipID:e3 ADC1:72
Hello World [237]: Key:0 Duty:60 PWM-Freq:440 PWM-Period:2272 Accel[ID:3f] X:0001 Y:-1 Z:0062 ChipID:e3 ADC1:67
```

Osnovni projekt CubeIDE – USB Virtual COM Port

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







VIN projekt - VP6: STM32F4 VIN Demo

- Osvežitev: STM32F4

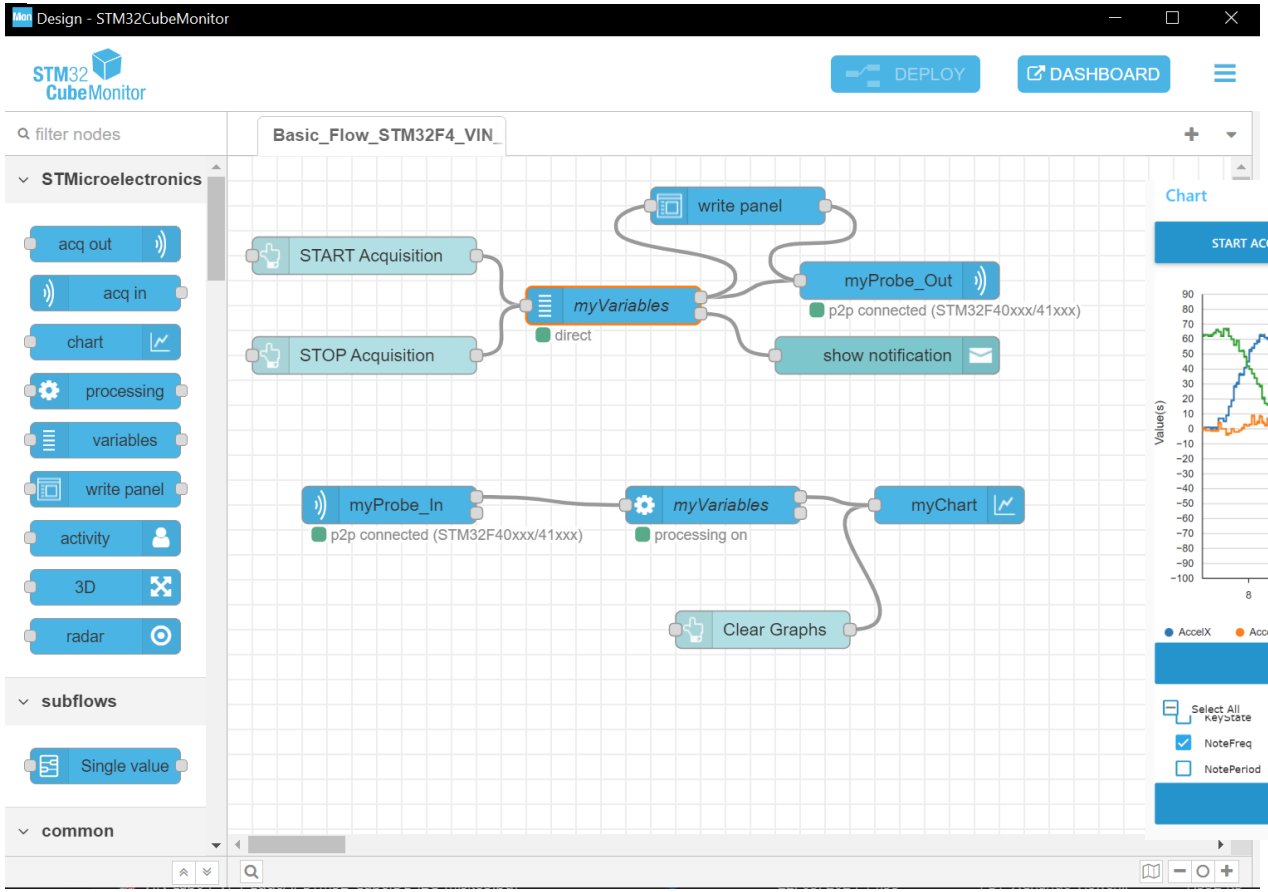
- CubeIDE projekt STM32F4 in V/I naprave :
  - CubeIDE projekt, GPIO in VCOM port
  - PWM - LED dimmer, brenčač
  - SPI - LIS3DSH pospeškomer
  - I2C - CS43L22 zvočni čip
  - ADC

- Sledenje („tracing“) - CubeMonitor, osciloskop

STM32CubeMonitor

STM32CubeMonitor is a tool that allows real-time sampling and visualization of user variables while the application is running. It runs on Windows, Linux or macOS, and provides a browser-based interface.

The user can define their own flow to monitor variables for their STM32 microcontroller-based application.
Example design and dashboard views are shown below.



https://wiki.stmicroelectronics.cn/stm32mcu/wiki/Category:STM32CubeMonitor

VIN projekt - VP 6 STM32-CubeIDE projekt

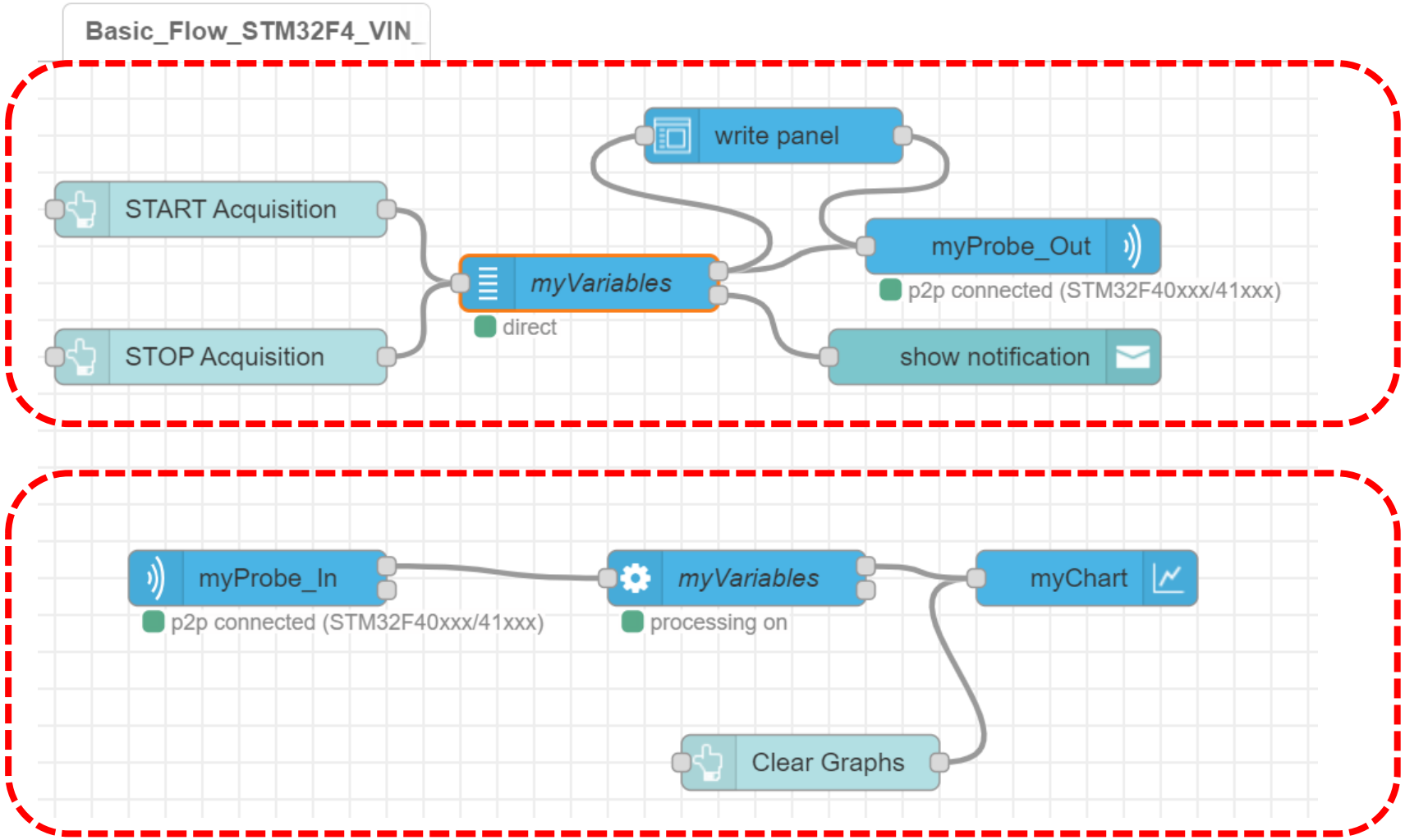
Program : za demonstracijo različnih funkcionalnosti – ADC, PWM – LED, Buzzer, SPI - Accel, I2C - audio

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
        while (1)
        {
        htim4.Instance->CCR1 = Duty;
        htim4.Instance->CCR2 = 100-Duty;
        htim4.Instance->CCR3 = Duty;
        htim4.Instance->CCR4 = 100-Duty;

        KeyState = HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_0);

        // Read x,y,z axes
        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);
        AccelX = indata[1];

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

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

        HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
        AnalogValue1 = HAL_ADC_GetValue(&hadc1);
        HAL_ADC_Start(&hadc1);
/* USER CODE END WHILE */
```

```
/* USER CODE BEGIN 3 */
if ( (HAL_GetTick() - TickLast) > 1000) { // Do this each second !
Duty = (Duty + 10) ; // Add 10 if delay 1 sec, add 1 on shorter delay...
if (Duty > 100 )
Duty = 1;

// 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);

// Change Period and set 50% duty for buzzer PWM output
NotePeriod = (int)(1000000/NoteFreq); //Already prescaled to 1 MHz
setPWM(htim2, TIM_CHANNEL_1, NotePeriod, NotePeriod/2);

// Print values on USB VComPort
snprintf(SendBuffer,BUFSIZE,"Hello World [%d]: Key:%d Duty:%d PWM-Freq:%d PWM-
Period:%d Accel[ID:%02x] X:%04d Y:%d Z:%04d ChipID:%02x
ADC1:%d\r\n",Counter++,KeyState,Duty,NoteFreq,NotePeriod,lis_id,AccelX,AccelY,A
ccelZ,ChipID,AnalogValue1);
CDC_Transmit_FS(SendBuffer,strlen(SendBuffer));

TickLast = HAL_GetTick(); // Reset counter
};

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

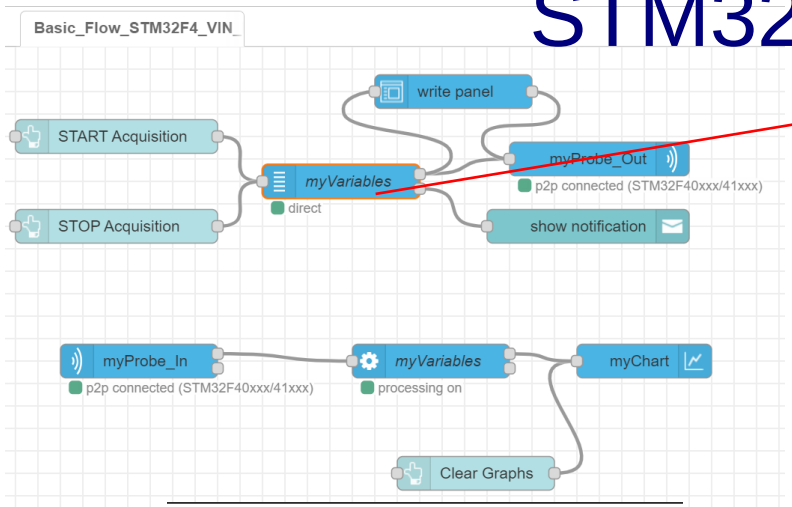
https://github.com/LAPSyLAB/STM32F4_Discovery_VIN_Projects/tree/main/STM32F4_GPIO_PWM_SPI_I2C_C_Demo

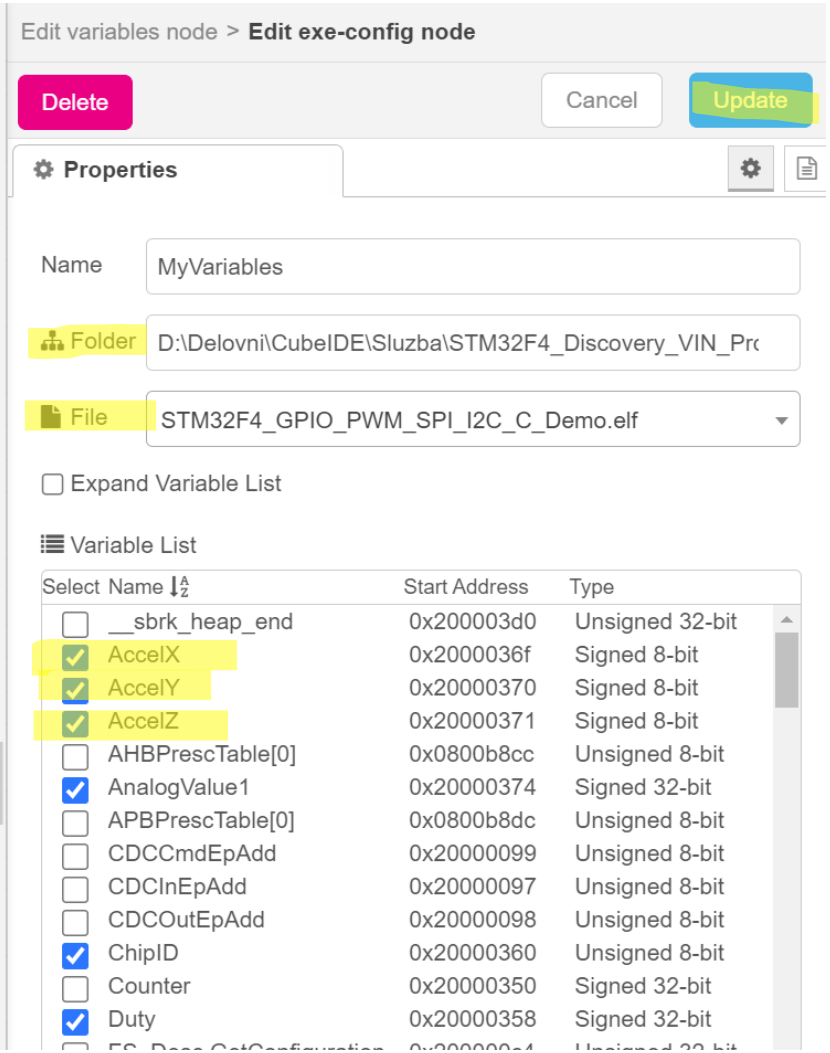
STM32CubeMonitor



https://wiki.stmicroelectronics.cn/stm32mcu/wiki/Category:STM32CubeMonitor

STM32CubeMonitor





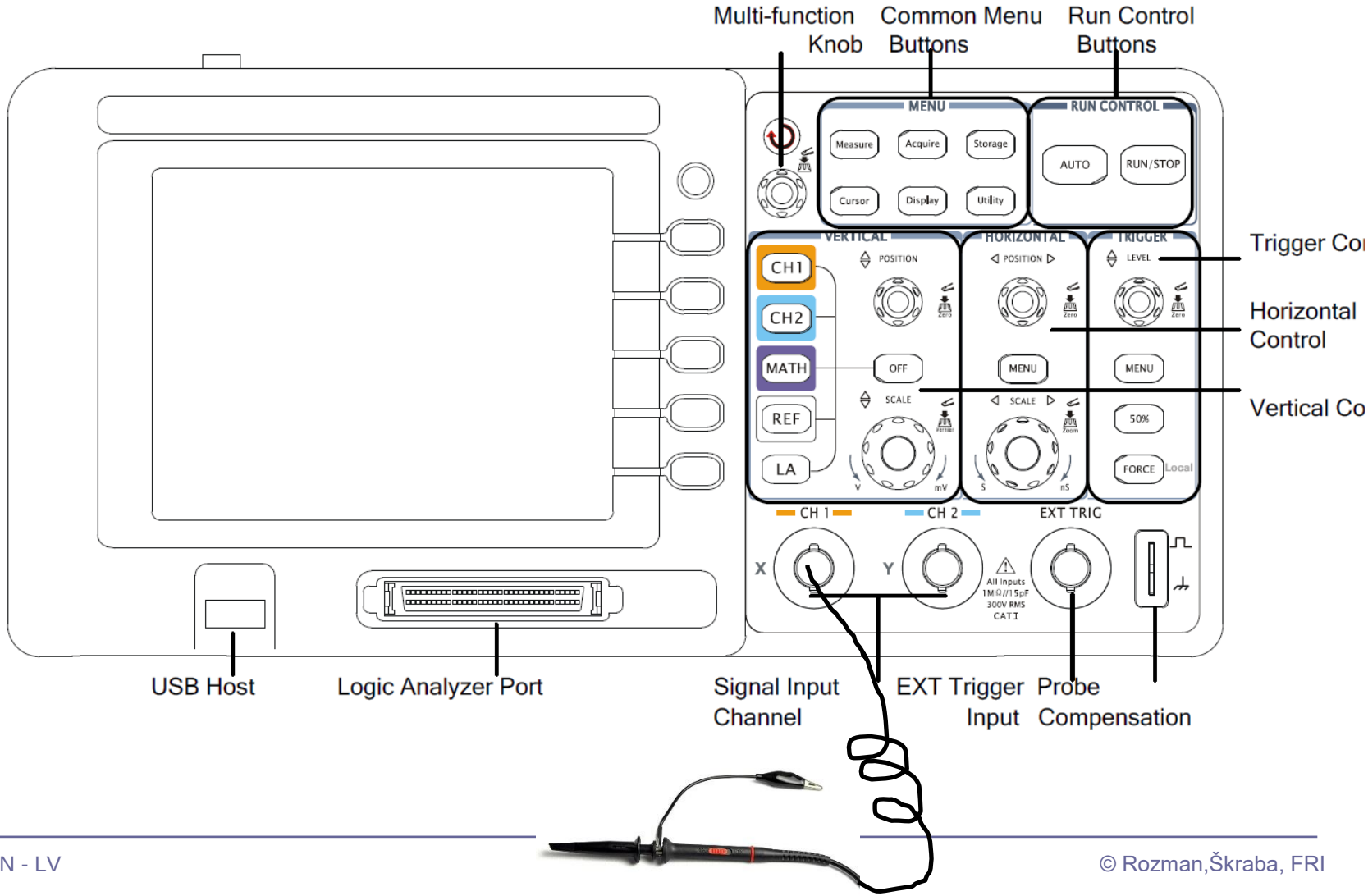
VIN projekt - VP6: STM32F4 VIN Demo

- Osvežitev: STM32F4

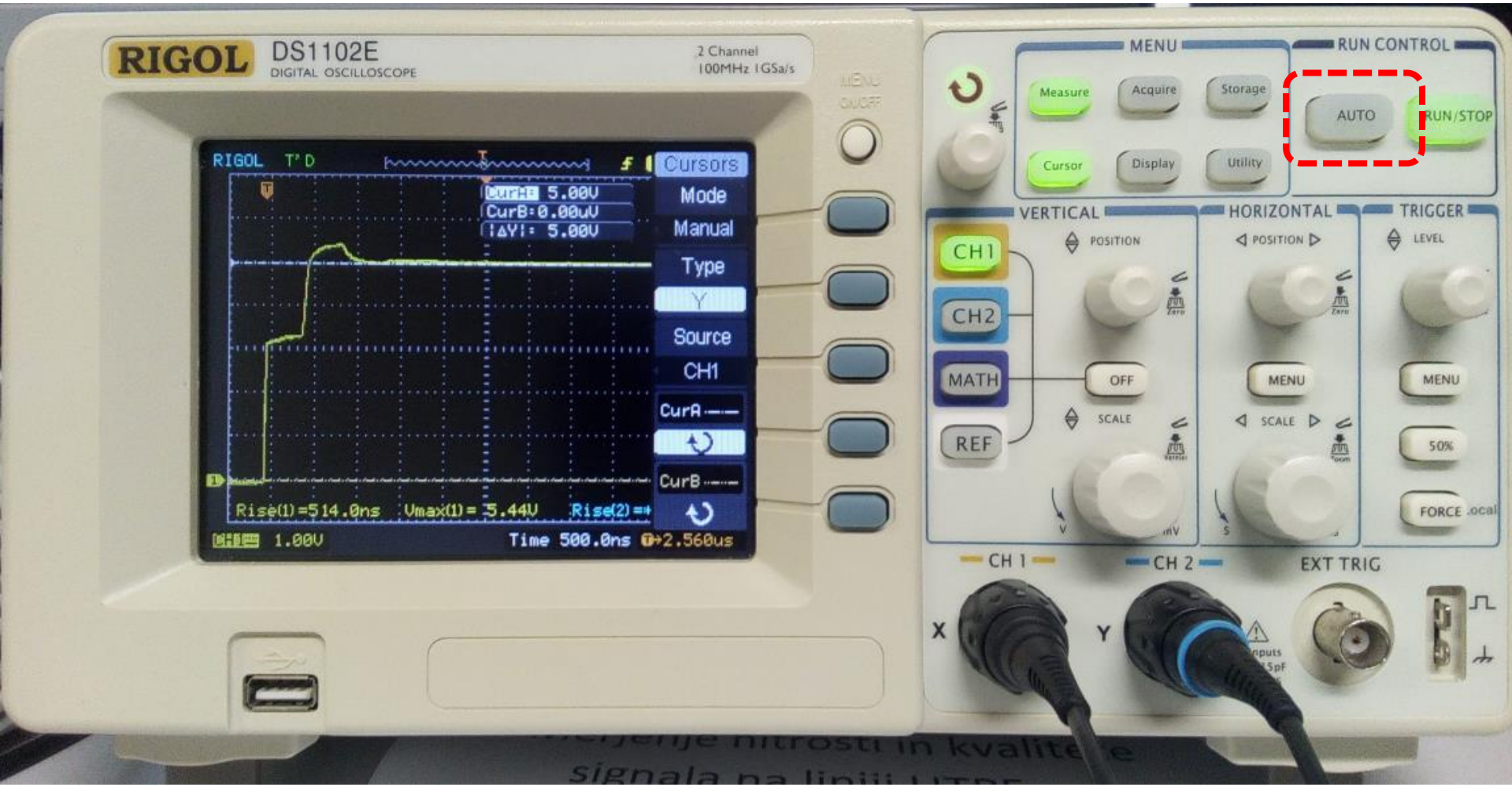
- CubeIDE projekt STM32F4 in V/I naprave :
  - CubeIDE projekt, GPIO in VCOM port
  - PWM - LED dimmer, brenčač
  - SPI - LIS3DSH pospeškomer
  - I2C - CS43L22 zvočni čip
  - ADC

- Sledenje („tracing“) - CubeMonitor, osciloskop

Prednja stran osciloskopa - shema



Prednja stran osciloskopa - realna



Prednja stran osciloskopa - kontrole

Y-os (el. napetost)

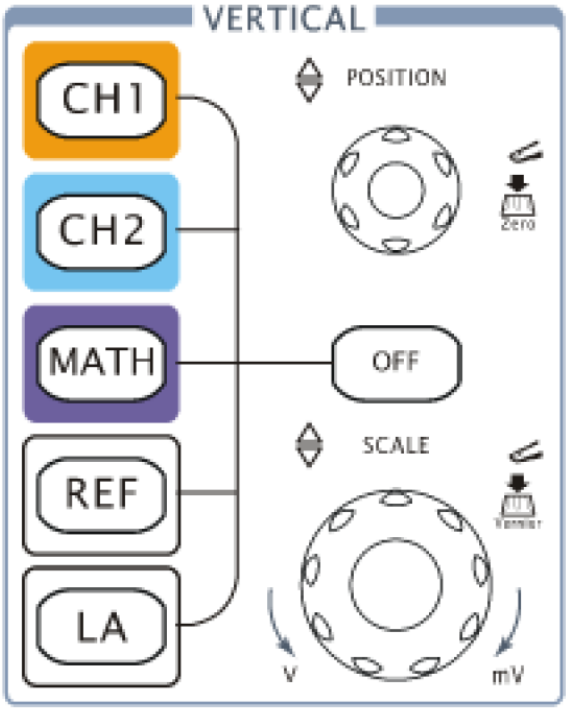
- **nastavitev merila [V/razdelek]**
- **pozicioniranje**

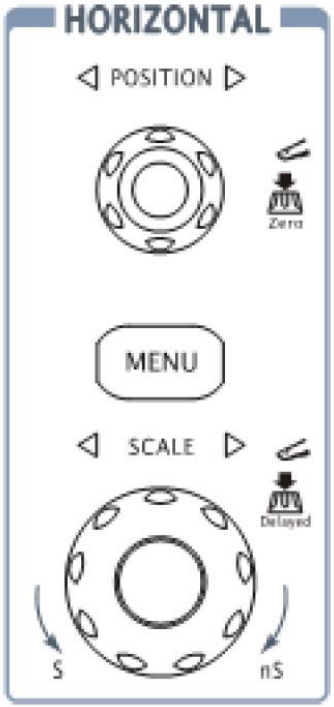
X-os (čas)

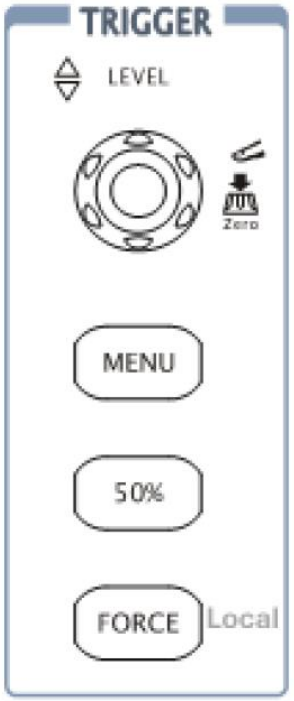
- **nastavitev merila [s/razdelek]**
- **pozicioniranje**

Prožilnik

- **začetek dogodka**
- **običajno 50%**



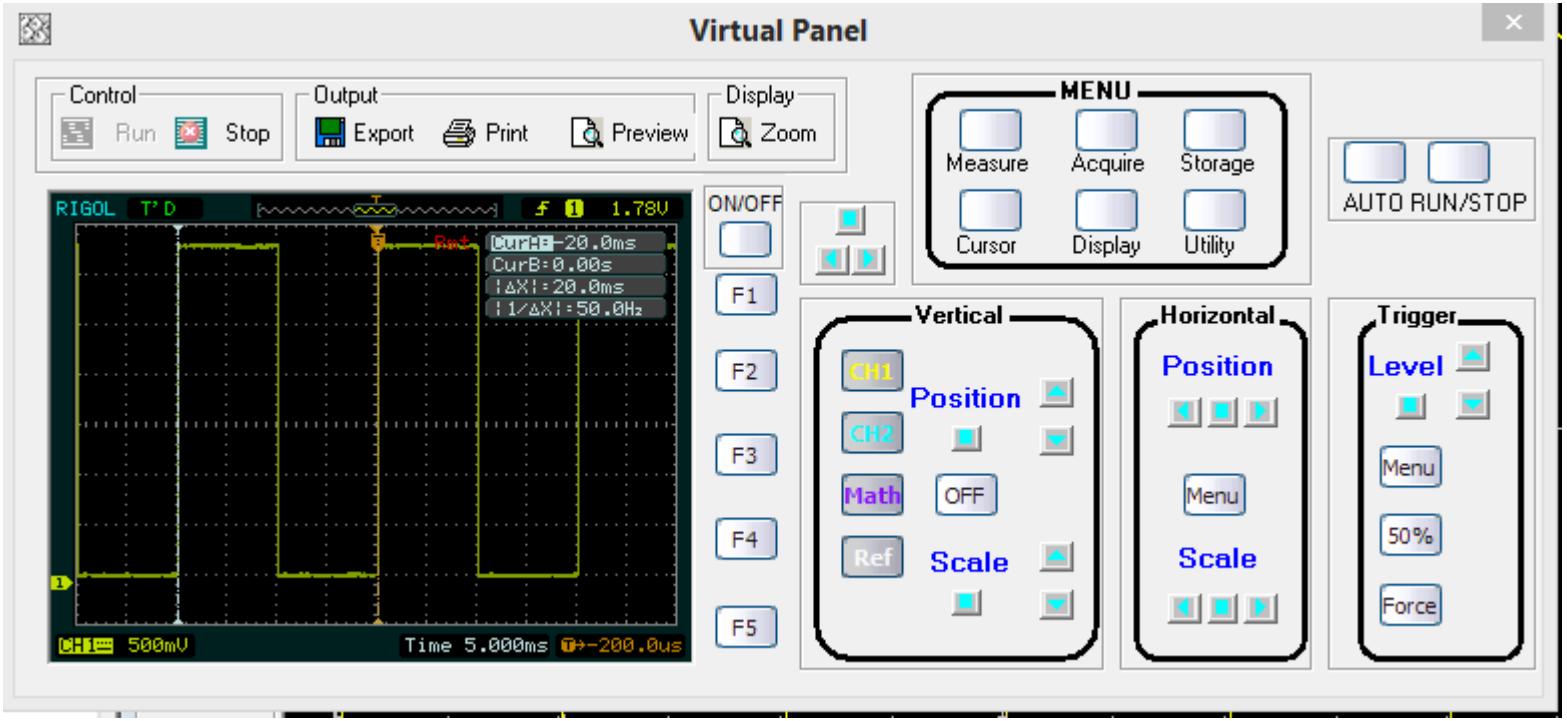




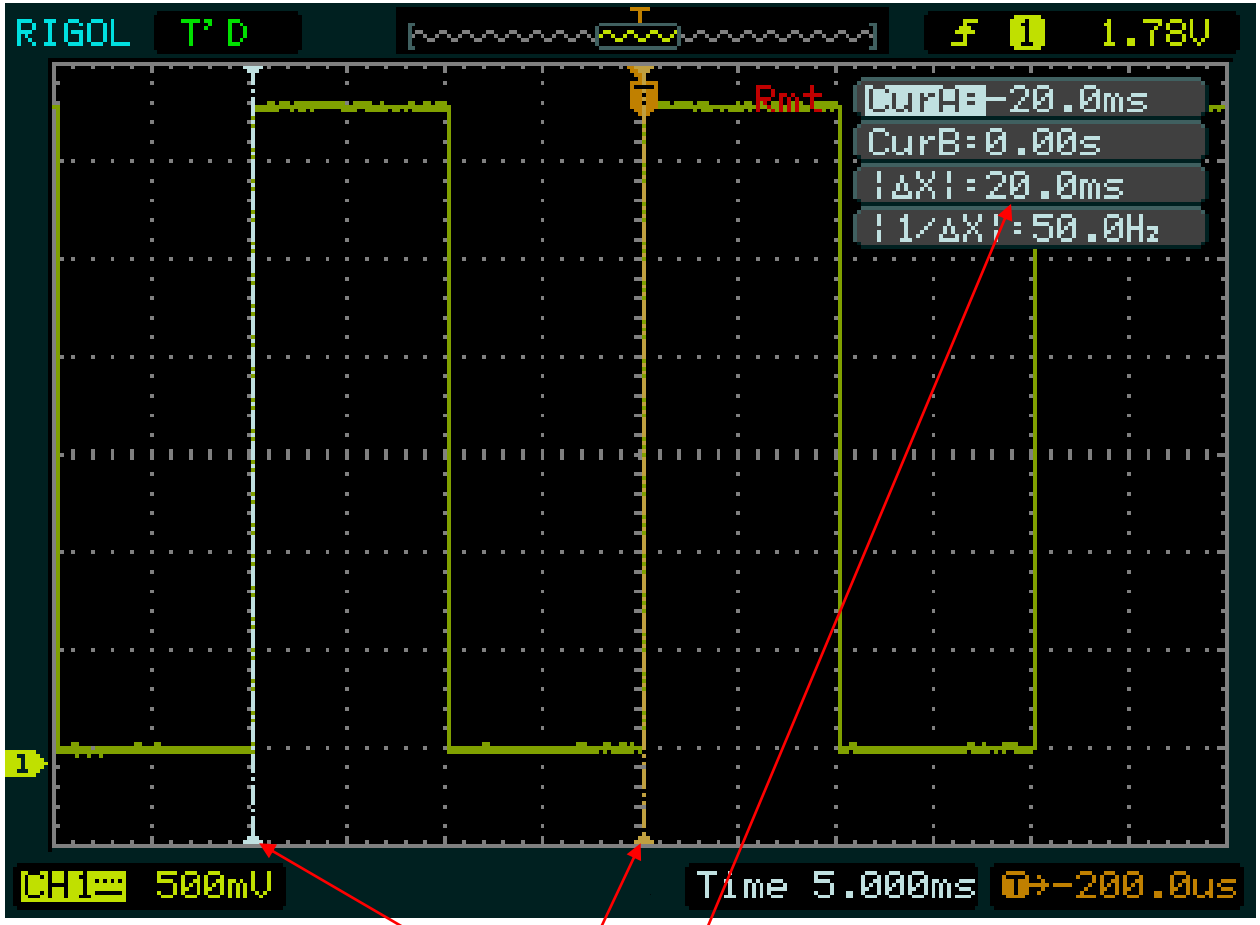
https://www.rigolna.com/products/digital-oscilloscopes/1000/
https://www.youtube.com/watch?v=TAQflYAa2VM

Spoznavanje merilne opreme…

PC aplikacija za osciloskop (USB povezava)



Zaslon osciloskopa – meritev periode



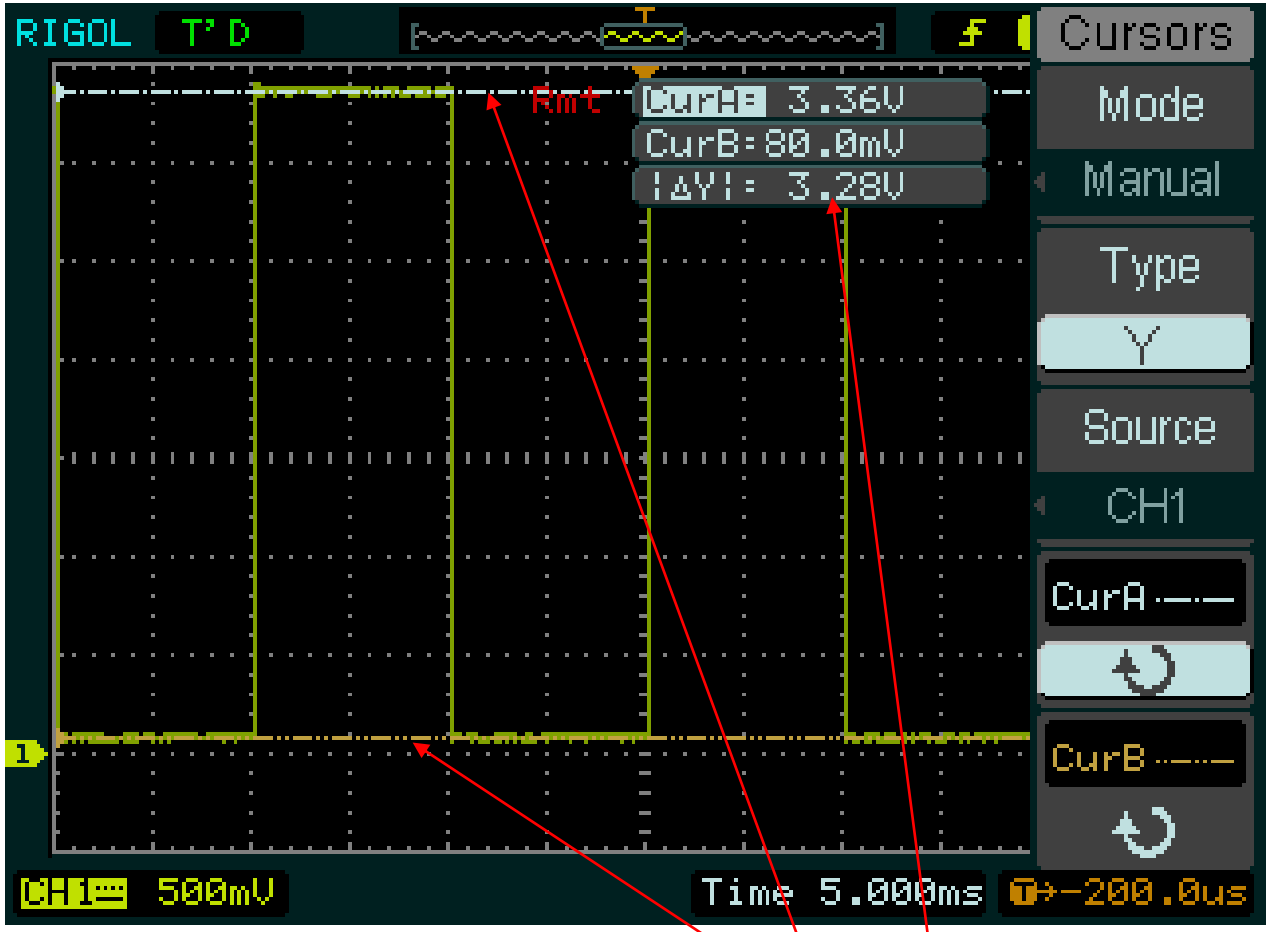
Programska nastavitev:

- **Delay 10ms**
- **Perioda 20ms**

Meritev periode signala:

- **20ms**

Zaslon osciloskopa – meritev amplitude



Meritev amplitude signala:

- **3.28V**

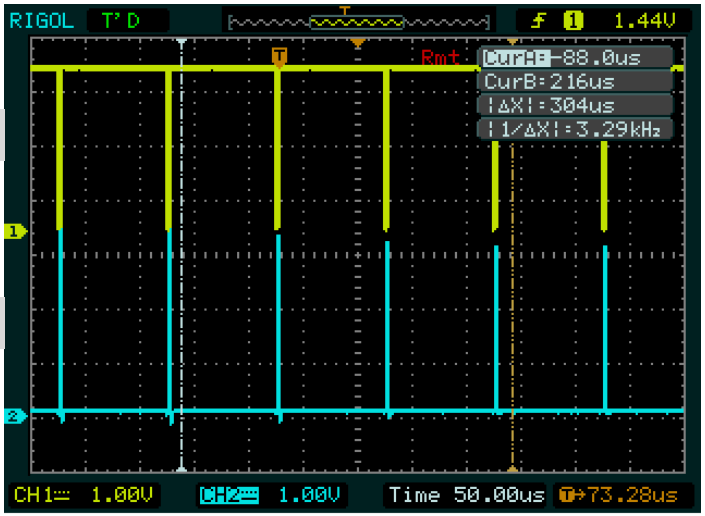
Osnovni projekt CubeIDE – GPIO – PWM, LED diode

HAL - C

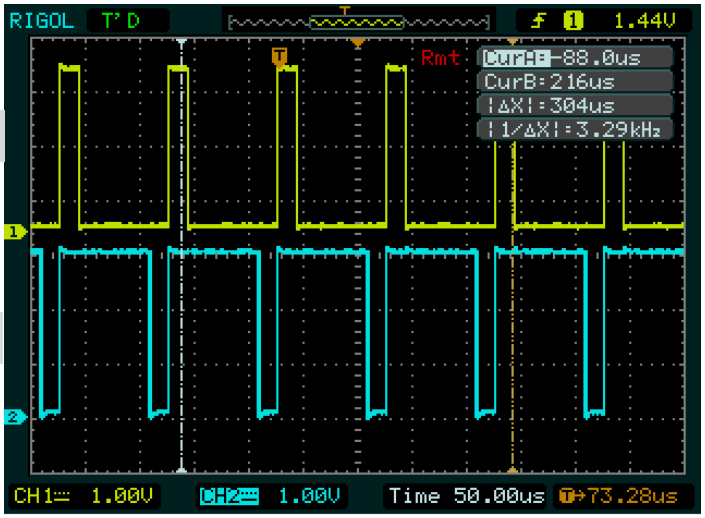
```
/* USER CODE BEGIN PV */
#define BUFSIZE 256
char SendBuffer[BUFSIZE];

/* USER CODE END PV */
/* USER CODE BEGIN 2 */

  HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_1);
  HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_2);
  HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_3);
  HAL_TIM_PWM_Start(&htim4, TIM_CHANNEL_4);

  /* USER CODE END 2 */

 /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    htim4.Instance->CCR1 = duty;
    htim4.Instance->CCR2 = 100-duty;
    htim4.Instance->CCR3 = duty;
    htim4.Instance->CCR4 = 100-duty;

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    snprintf (SendBuffer,BUFSIZE,"USB:0.1 secs. Duty=%d%%\r\n",duty);
    CDC_Transmit_FS(SendBuffer,strlen(SendBuffer));

      duty = (duty + 1) ;
      if (duty > 100 )
        duty = 0;


    HAL_Delay(100);

  }
  /* USER CODE END 3 */
```

Max Duty

Min Duty



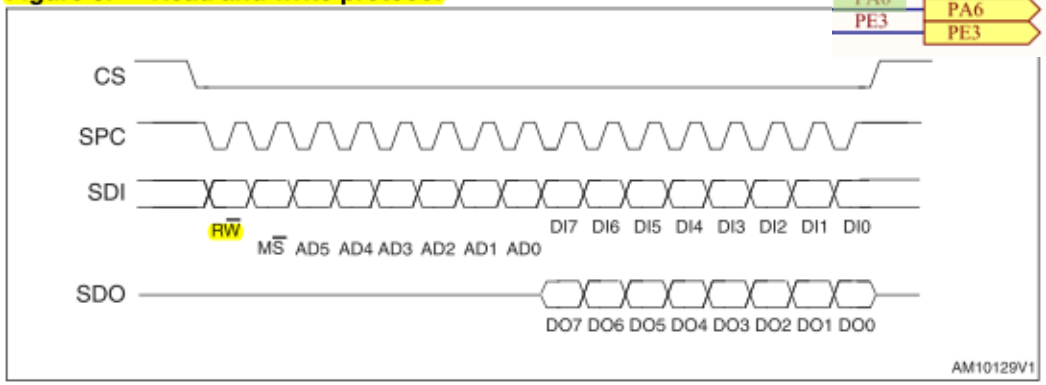
Min Duty

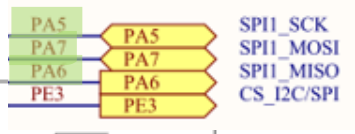


Max Duty

GPIO priključki :
PD12-PD15: 4 LED diode

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

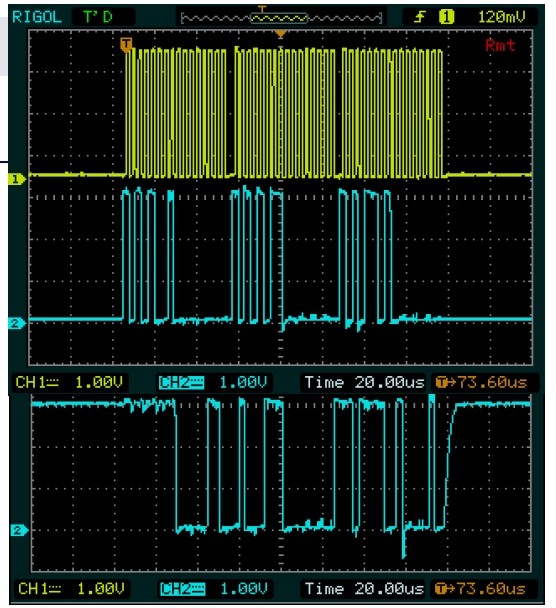




SCK

MOSI

MISO



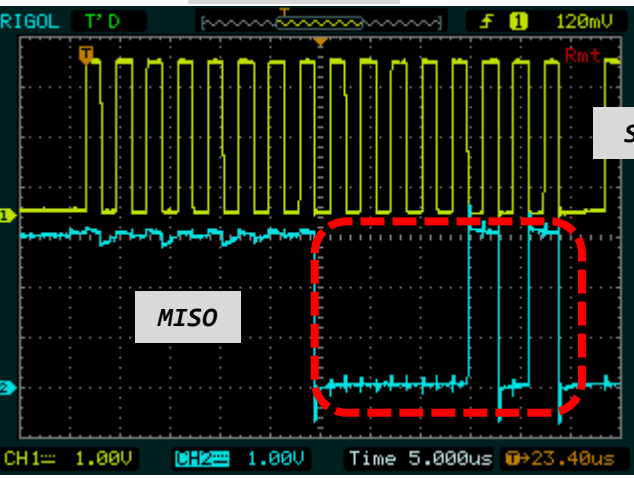
STM32F4

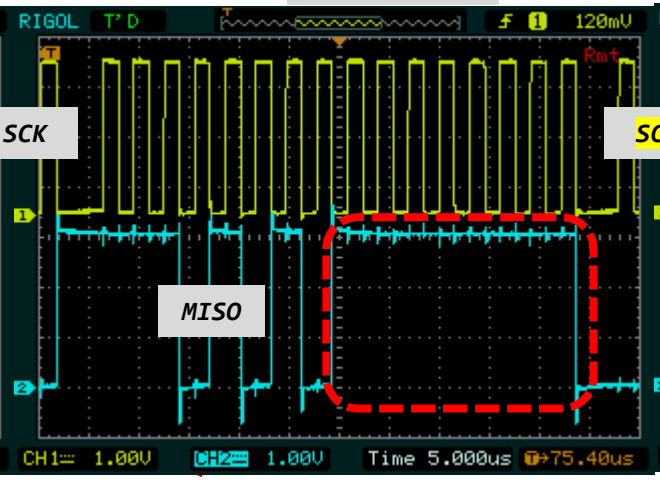
```
Hello World [3530]: Key:0000 Accel[ID:00] X:0005 Y:-1 Z:0066
Hello World [3531]: Key:0000 Accel[ID:00] X:0005 Y:-1 Z:0067
```

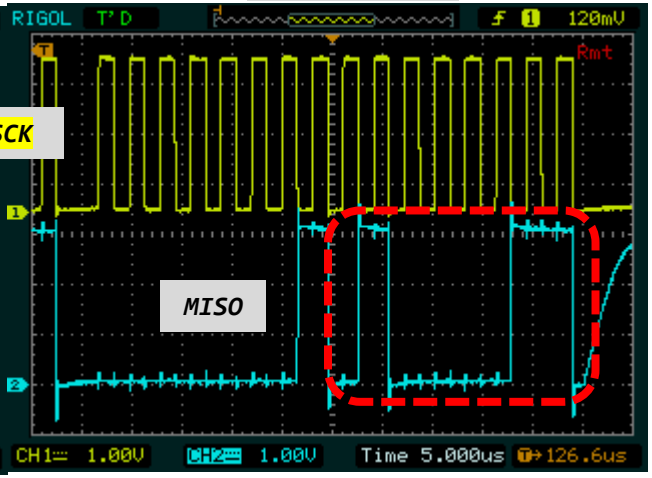
GPIO priključki :
PA5-PA7: SPI,
PE3 Chip Select

X-Accel: 5

Y-Accel: -1







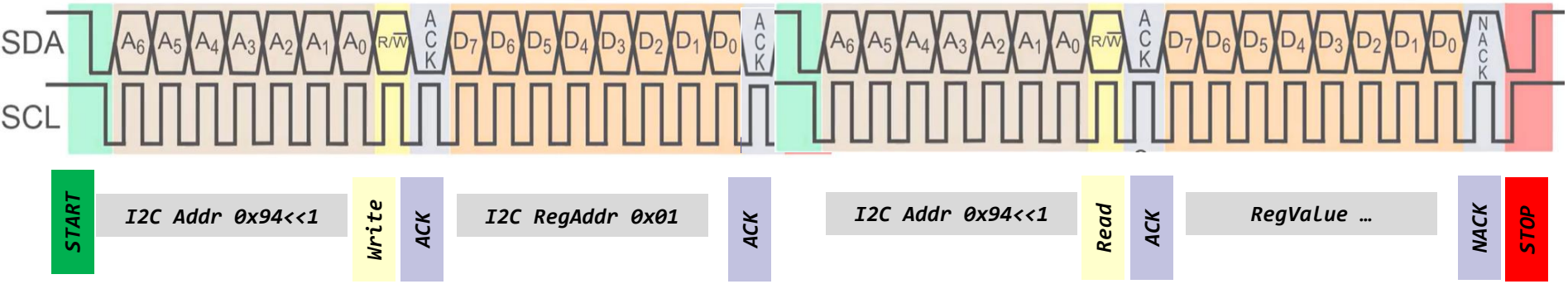
STM32F4

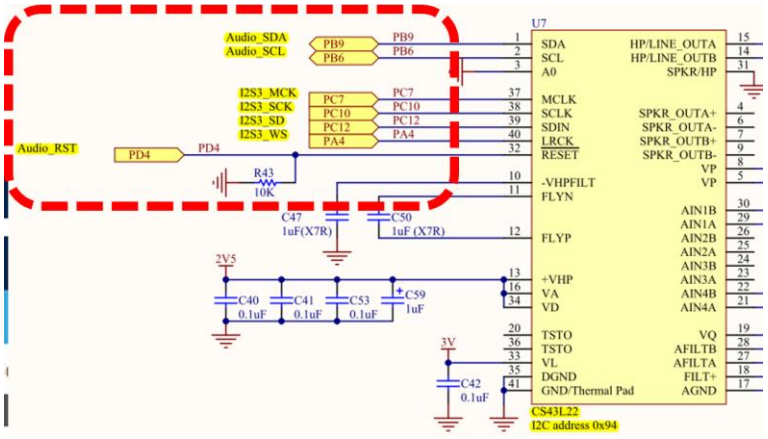
I2C branje

GPIO priključki :
PB6,9: I2C1,
PD4: Reset (1=Dev-ON)

main.c : dodana koda

```
/* 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 */
```





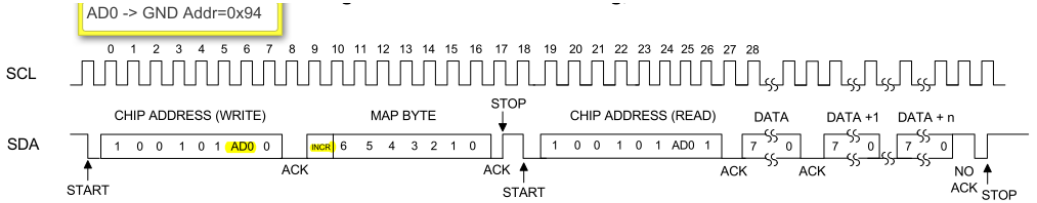
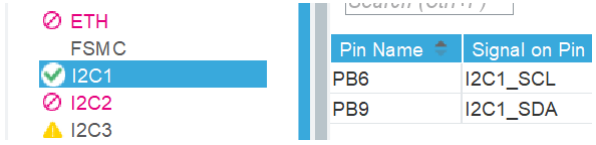


Figure 17. Control Port Timing, I²C Read



https://github.com/LAPSyLAB/STM32F4_Discovery_VIN_Projects/tree/main/STM32_I2C_CS43L22_Basic

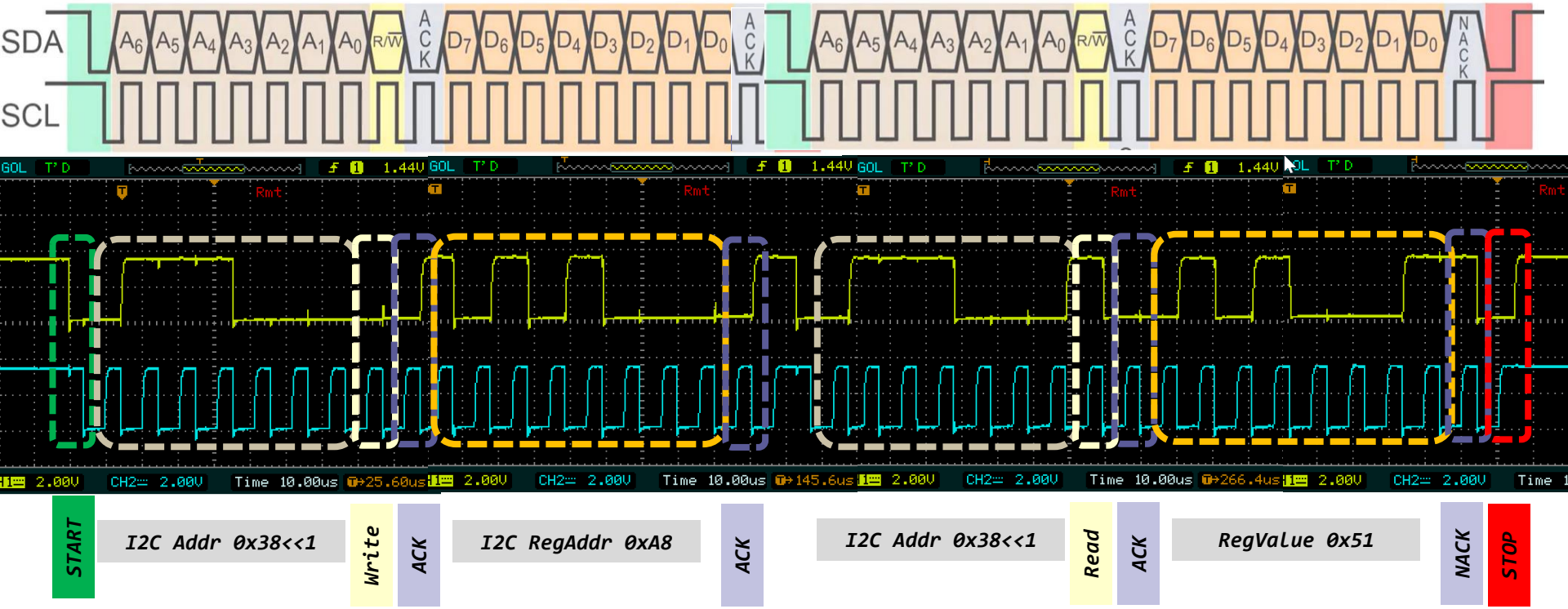
STM32H7

I2C branje

GPIO priključki :
PD12,13: I2C4

main.c : dodana koda

```
// Reading from address 0x38 register Vendor's Chip ID (addr. 0xA8) default value should be 0x51=81

retval = HAL_I2C_Mem_Read(&hi2c4, (0x38 << 1), 0xA8, I2C_MEMADD_SIZE_8BIT,&dataBuffer[5], 1, HAL_MAX_DELAY);
```



https://github.com/LAPSyLAB/STM32H7_Discovery_VIN_Projects/tree/main/STM32H750B-DK_I2C_Basic_Demo

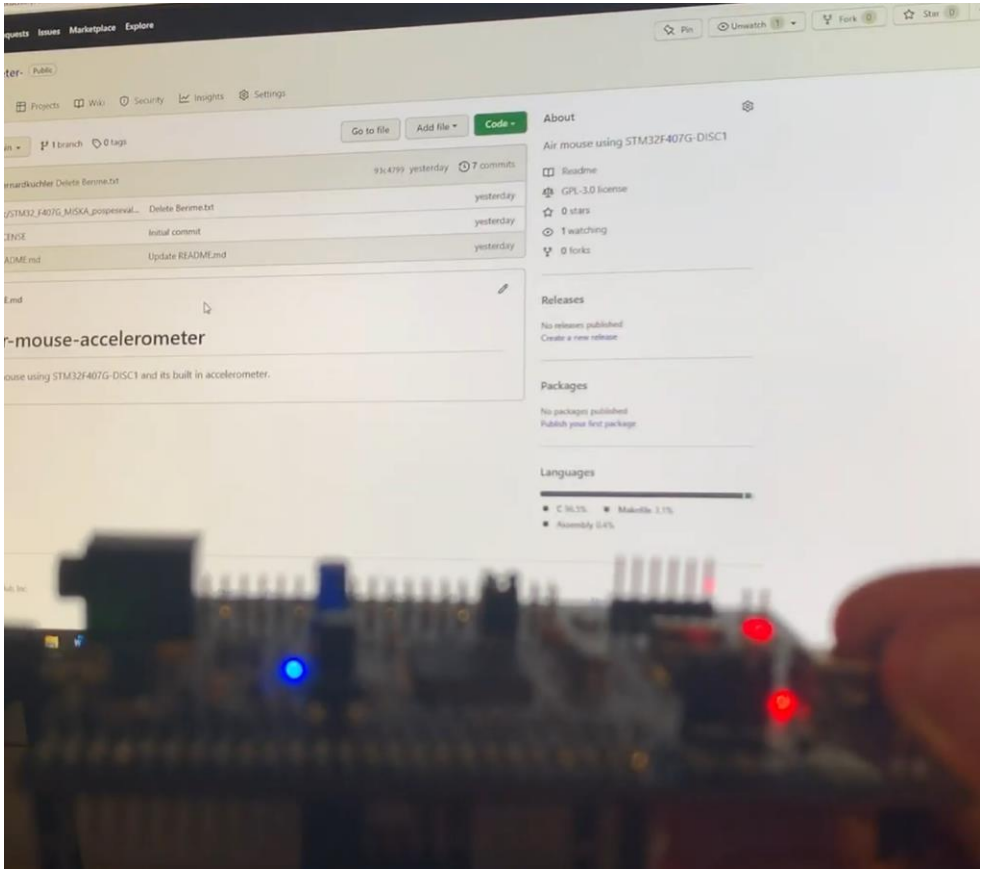
VIN projekt - VP6: STM32-breadboard vezave

- Osvežitev: STM32 breadboard vezave
- Osciloskop
- Uporaba osciloskopa – VP4 :
  - SPI
  - PWM
  - I2C
- VIN Projekt – STM32F4 (AirMouse, Wav-audio), Smarteh

AirMouse STM32F4

Beremo pospeškomer in sporočamo premike kazalca na zaslonu preko ustreznega USB profila

Avtor: Bernard Kuchler



USB_DEVICE Mode and Configuration

Mode

Class For HS IP: Disable

Class For FS IP: Human Interface Device Class (...

- Disable
- Audio Device Class
- Communication Device Class (Virtual P
- Download Firmware Update Class (DF
- Human Interface Device Class (HID)
- Custom Human Interface Device Class
- Mass Storage Class

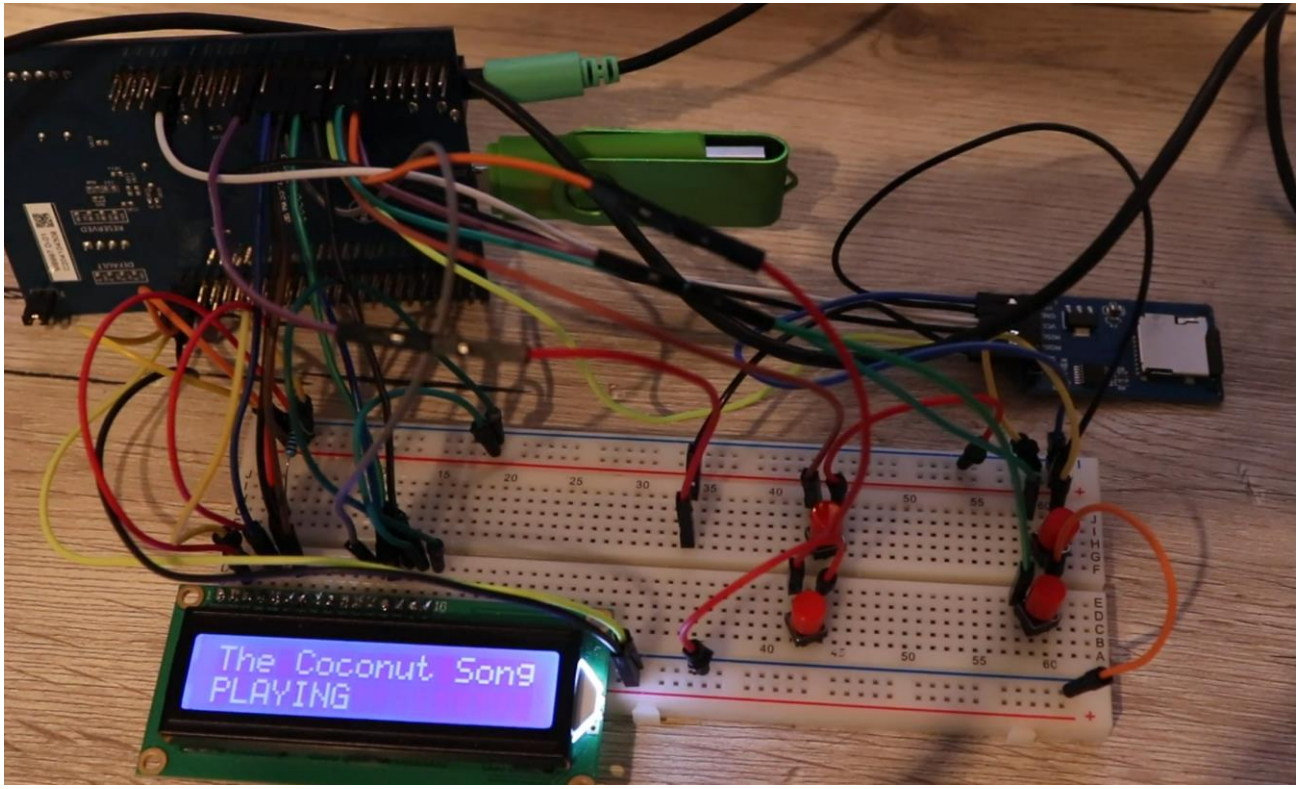
```
while (1)
{

// Read accel values into AccelX,Y,Z

if (AccelX < min_xval){
newxval = AccelX - min_xval;
} else if (AccelX > max_xval) {
newxval = AccelX - max_xval;
}

if (AccelY < min_yval){
newyval = AccelY - min_yval;
} else if (AccelY > max_yval){
newyval = AccelY - max_yval;
}

if ((newxval > 10) || (newxval <-10)) //Determines the necessary
amount of change in value from the sensor to start moving the mouse
cursor
{
mousehid.mouse_y = (newxval/10); //Divides the value from the
sensor by 10 in order to make a slower acceleration of the mouse
cursor and thereby making it more accurate to use
}
else mousehid.mouse_y = 0;


if ((newyval > 10) || (newyval <-10)) {
mousehid.mouse_x= (newyval)/10;
} else mousehid.mouse_x = 0;
…
USBD_HID_SendReport(&hUsbDeviceFS,&mousehid, sizeof (mousehid));
//Send data to USB
```

https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples/tree/main/STM32_F407G_MISKA_pospesevalniki

Primer kompleksnejše demo USB-Audio aplikacije :

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

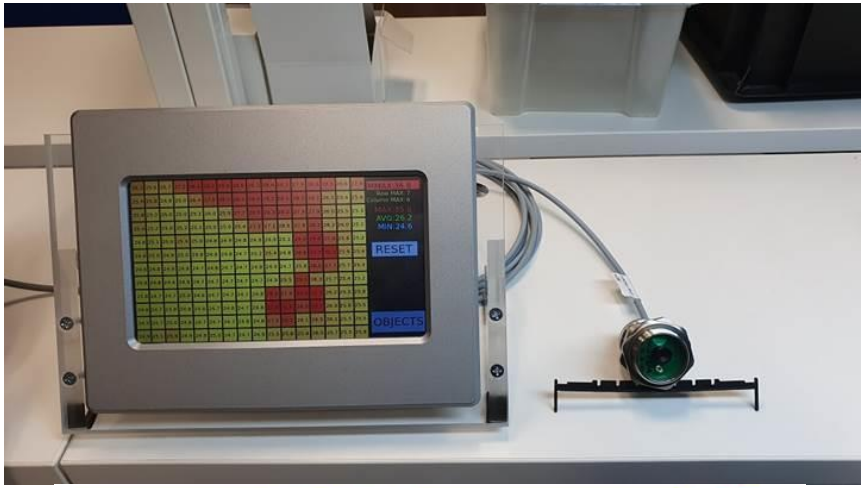
Avtor: Matic Pristavnik Vrešnjak

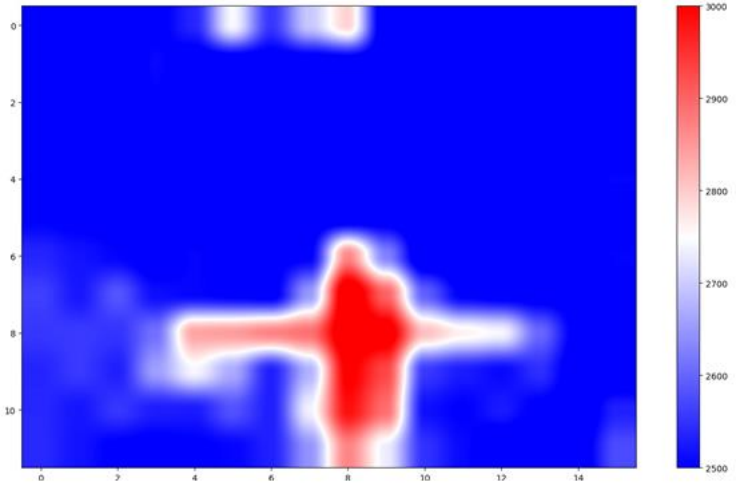


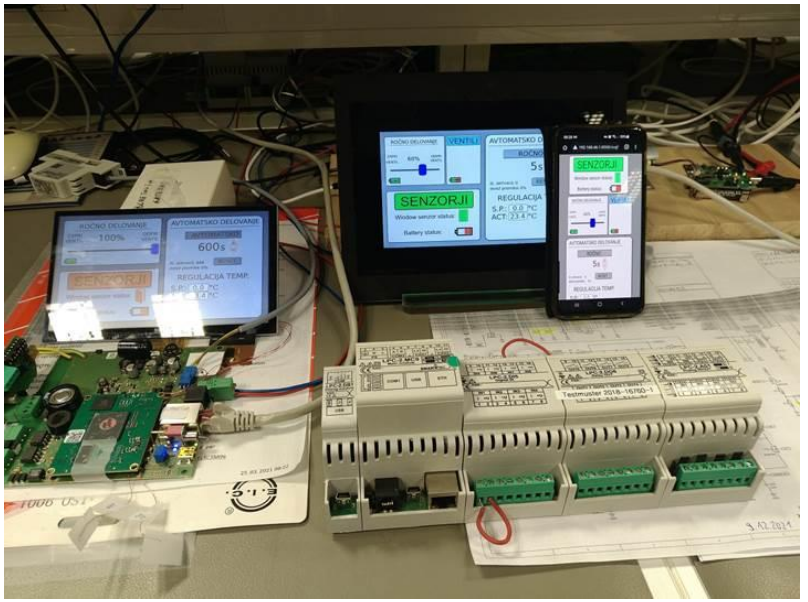
https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples/tree/main/STM32_Wav_Player_SD_USB

Smarteh

Matrični senzorji (LIR, LIDAR)







Osnovni PLC komplet za uporabo :

- Model pametne hiše
- VIN LAB vaje
  - RS485, Modbus, osciloskopi, …

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

- Diskusija, vprašanja ?