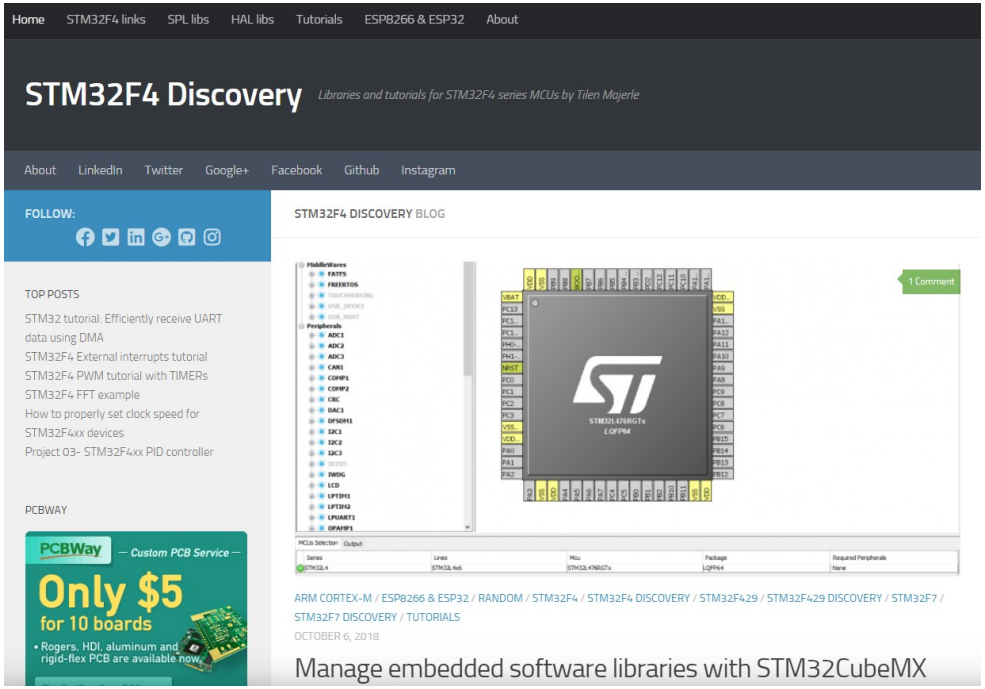
Vhodno izhodne naprave

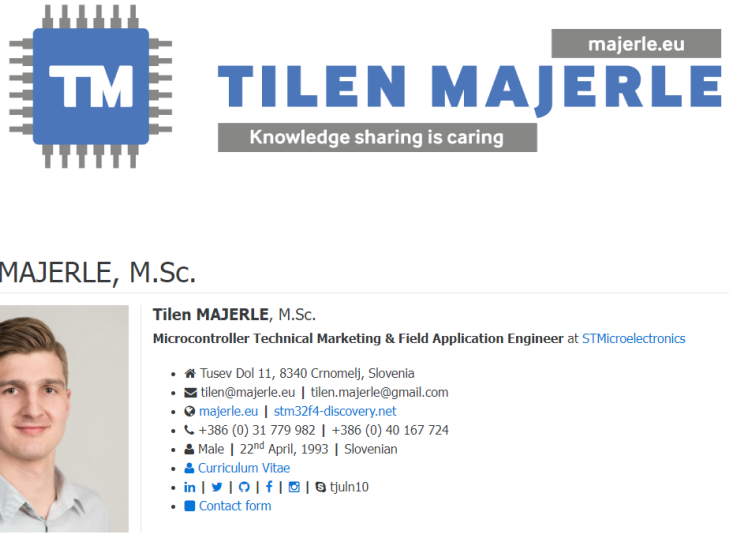
Laboratorijska vaja 5 - VP 5
VIN projekt, Miško3, osciloskop, STM32H7 PWM primeri

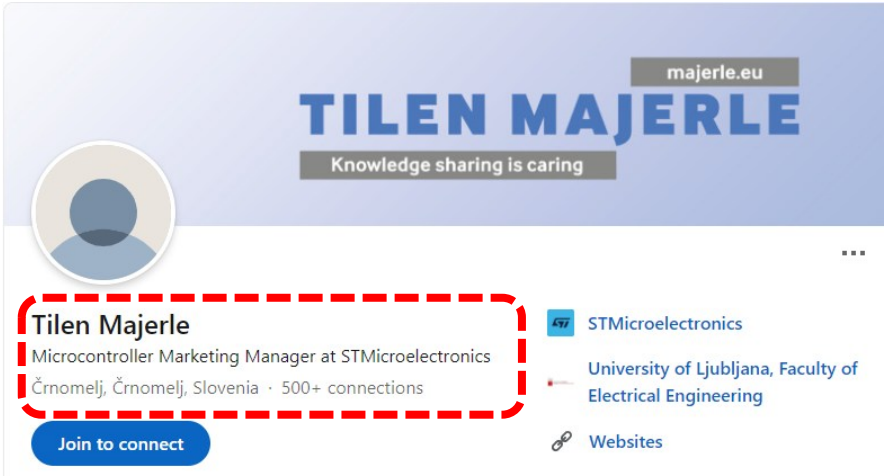
VP 5: VIN projekt (Miško3), osciloskop, STM32H7 PWM primeri

- VIN projekt
  - Miško3 – demo projekt
- Osciloskop - spoznavanje
- HAL knjižnica
- STM32 CubeIDE H7 – PWM izhodi, brenčač
- STM32H7 CubeIDE, I2C (Scan, WM9884, Touch)

Delo na STM32F4 razvojnem sistemu - zgodba







https://stm32f4-discovery.net/

VIN projekt

Spisek opreme

LAPSy Embedded Academy zvezek
_Knjižnica vsebine

Projekt-Delo | Vsebine-TODO | Aktualno | VIN Projekt - Ideje • | Moduli, Tipala •

Išči po zvezkih

Tipala za delo

sreda, 31. marec 2021 18:50

- VIN Sensors Challenge
  - Sharp IR GP2D12
    - Effectively Using GP2D12
    - Sharp IR GP2Y0A21YK •
  - HC SR04 UZ Senzor
    - Ultrasonic Distance Senso... •
  - Devantech El. Compass
  - Devantech SRF04 •
  - LV-MaxSonar-EZ1
  - PIR Napion Senzor
  - MPXV10GC7U Senzor pritiska
  - A3144 Hall effect Sensor
    - A3144 Hall effect Sensor -... •
  - TPA81 Thermopile Array
    - Motion and Gesture Dete... •
    - Easy Motion and Gesture De...
  - IR tipalo TPiS 1S 1385
  - SHT11 Temp./Vlaga
  - HR202L - resistivni vlagomer •
- **Tipala za delo**
  - Tipala kakovosti zraka
    - APC1001U_EK
    - SEK-SCD41-SENSOR
  - Time-of-flight
  - 37 in 1 sensor kit for Arduino

VIN projekt

Ideje

LAPSy Embedded Academy zvezek
_Knjižnica vsebine

Projekt-Delo | Vsebine-TODO | Aktualno | VIN Projekt - Ideje • | Moduli, Tipala •

Išči po zvezkih

Dodaj stran

- Spletni viri
- Teme, področja
  - Edge, AI •
    - Tensoflow Lite •
  - Gesture detection (ToF sensors)
    - Time-of-Flight sensors
  - Termo kamera - LIR tipalo Sm...
  - Smarteh PLC
  - **Brezstično zaznavanje - Cap...**
    - ToF tipala in Theremin
    - Webcam Theremin
    - Make Your Own Simple Ther...
    - Theremin - brezstični glasbe...
    - Let's Design and Build a ST...
    - Bluepill Theremin
    - The "air" theremin
    - RPi
  - Emerging applications of time...
  - Tipala v športnih urah, medicin...
  - Logični/protokolski analizator
    - Sigrok - SW
    - I2c sniffer •
    - Red Pitaya
  - Praktični izzivi v LAPSYLAB
  - LSM6DSOX (30 kosov na voljo)

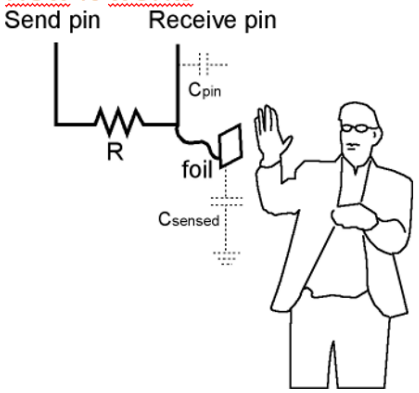
Brezstično zaznavanje - CapSense

nedelja, 24. april 2022 11:28

Capacitive Sensing Library
by Paul Badger

...

How it works



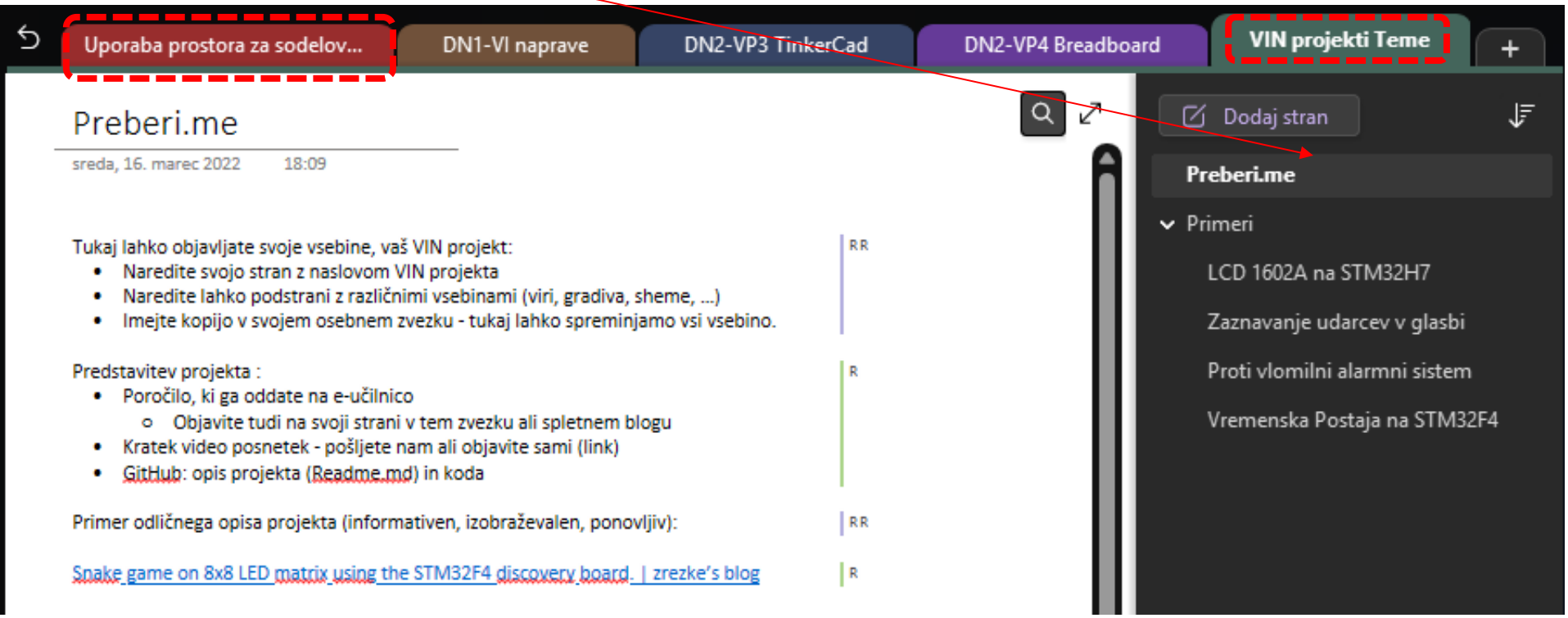
The capacitiveSensor method toggles a microcontroller *send* pin to a new state and then waits for the *receive* pin to change to the same state as the send pin. A variable is incremented inside a *while* loop to time the receive pin's state change. The method then reports the variable's value, which is in arbitrary units.
Watch a short video demonstration (YouTube)

From <https://playground.arduino.cc/Main/CapacitiveSensor/>

Elektrode na pleksi panelu - kot glasbena klavitura

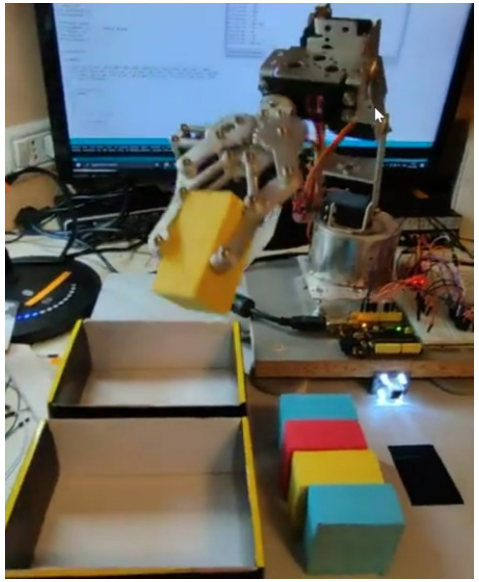
VIN projekt

Vaša tema ?

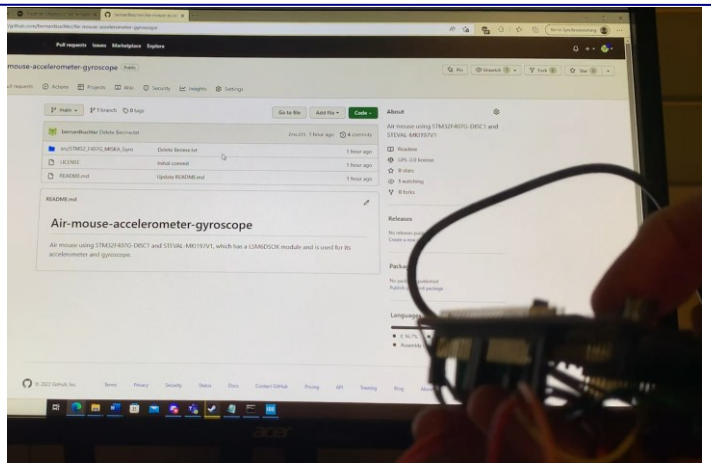


Primer dobrega poročila/predstavitve:
https://zrezke.github.io/jekyll/update/2022/02/23/8by8-snake.html

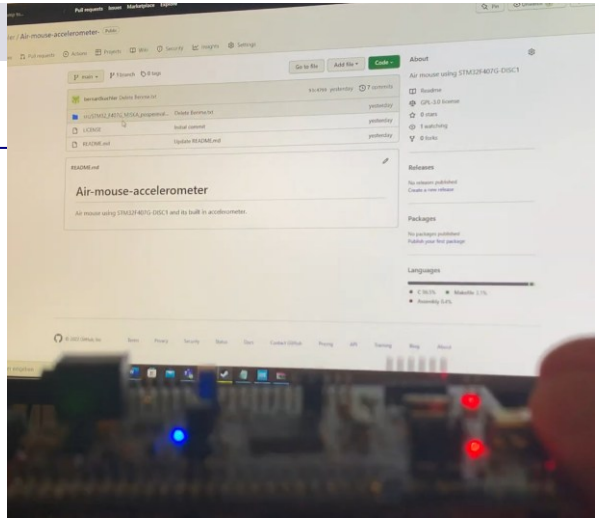
VP – Primeri projektov STM32F4, H7 – 21/24



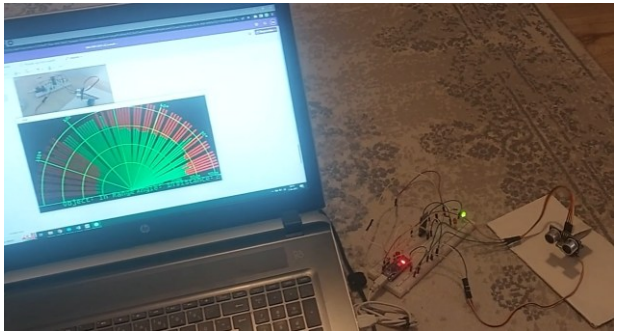
Robot: Colour Box sorter



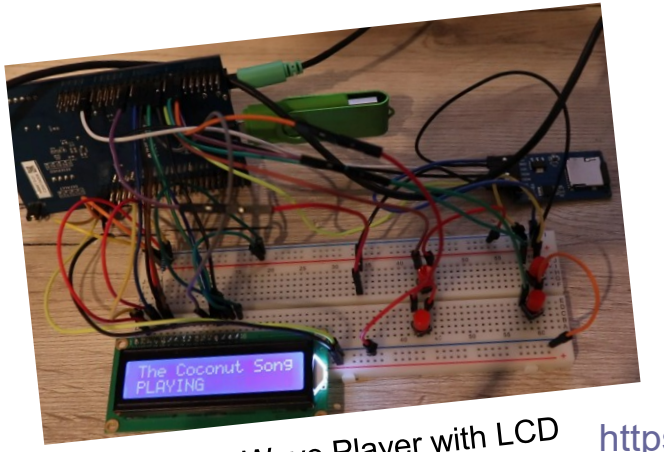
F4: LSM6DSOX – Air Mouse



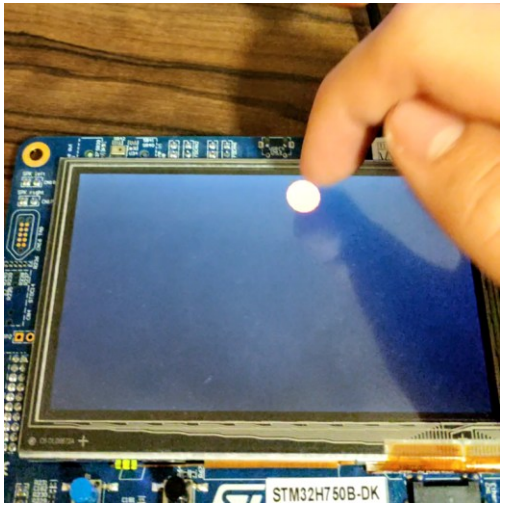
F4: Air Mouse



3D Sonar



F4: Wave Player with LCD

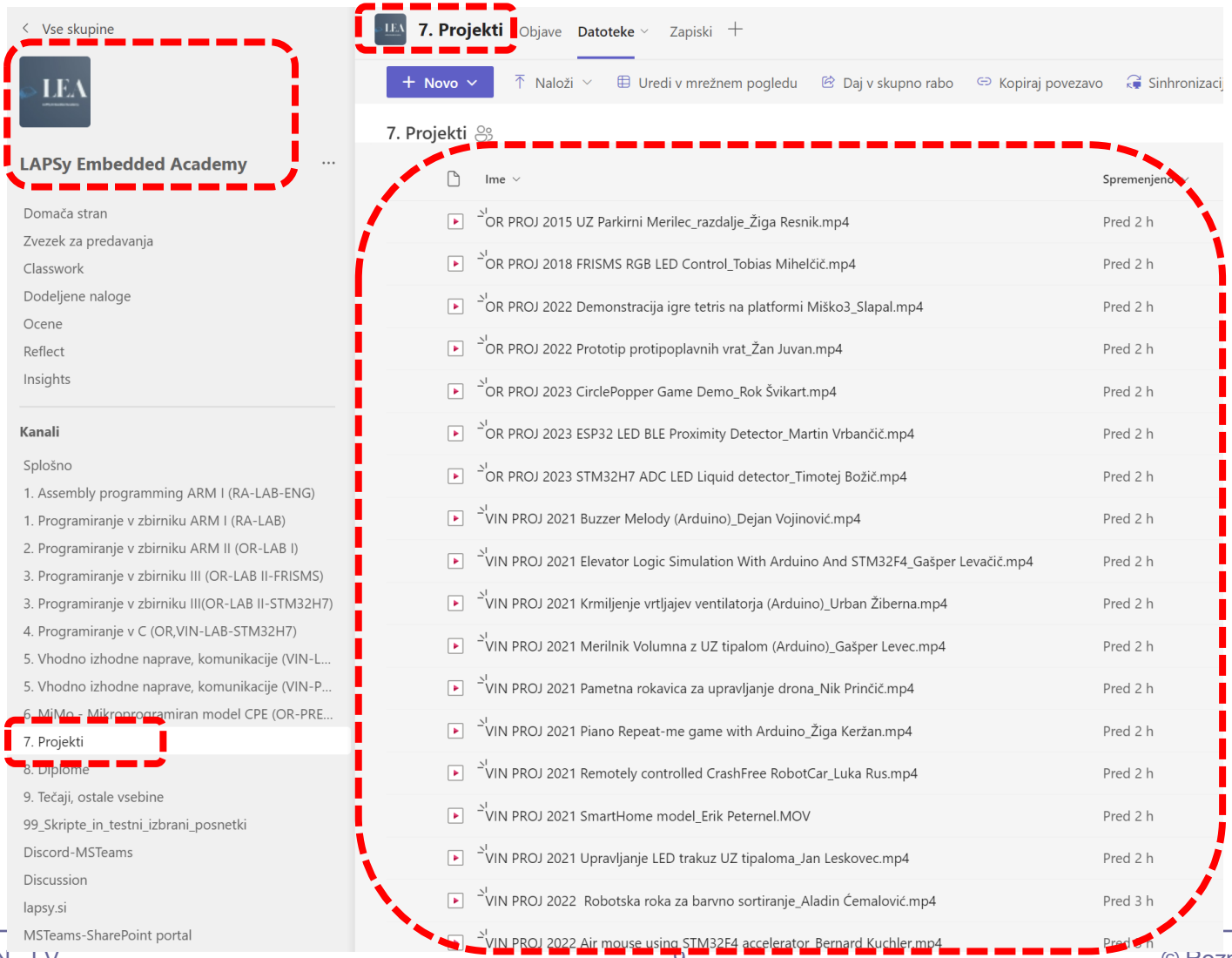


H7: Circle Popper

https://github.com/LAPSyLAB/STM32H750B-DK_Docs_and_Examples
https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples/

VIN projekt

Video posnetki VIN projektov iz prejšnjih let



VIN projekt

Video posnetki VIN projektov iz prejšnjih let

LAPSy Embedded Academy

Različni študentski projekti iz prejšnjih šolskih let pri predmetih RA, OR in VIN.

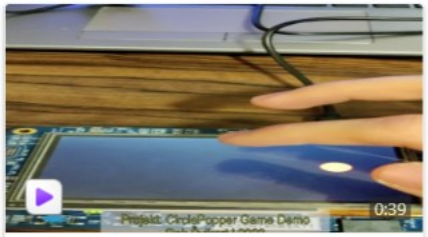
Video predstavitve študentskih projektov

See all

Forms

VIN PROJ 2021 Pametna rokavica za upravljanje drona_Nik Prinčič

Rozman, Robert
Edited Apr 8, 2024



Forms

OR PROJ 2023 CirclePopper Game Demo_Rok Švikart

Rozman, Robert
Edited Apr 8, 2024

Forms

VIN PROJ 2023 Zaznavanje človeka z matrično termo kamero_Jakob_jelovčan

Rozman, Robert
Edited Apr 8, 2024

Forms

_Študentski projekti pri predmetih RA, OR in VIN (RAVINOR Projekti)_Krajši...

Rozman, Robert
Edited Apr 13, 2024

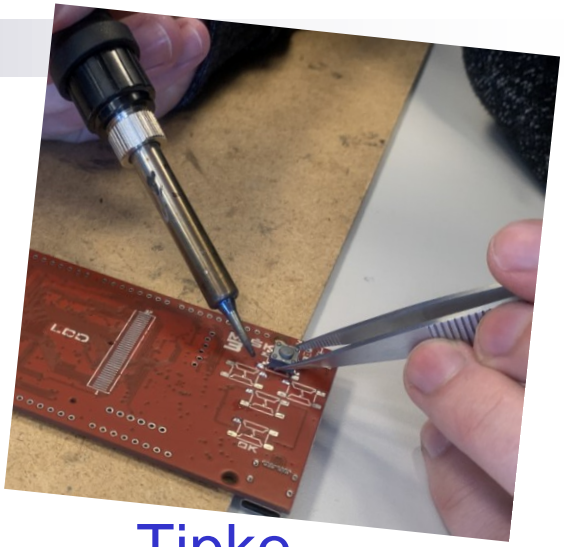
1 of 12

Študentski projekti pri predmetih RA, OR in VIN

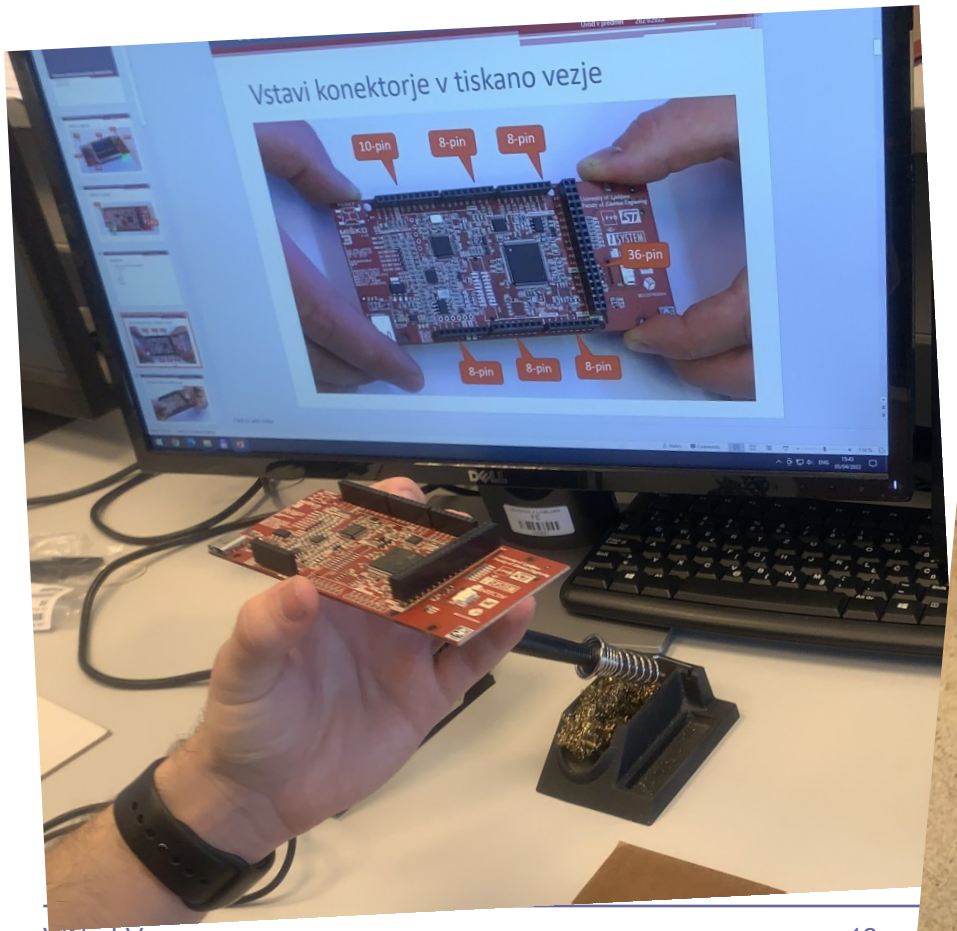
VP 5: VIN projekt (Miško3), „Edge AI“, STM32H7 PWM, I2C primeri

- VIN projekt
  - Miško3 – demo projekt
- Osciloskop - spoznavanje
- HAL knjižnica
- STM32 CubeIDE H7 – PWM izhodi, brenčač
- STM32H7 CubeIDE, I2C (Scan, WM9884, Touch)

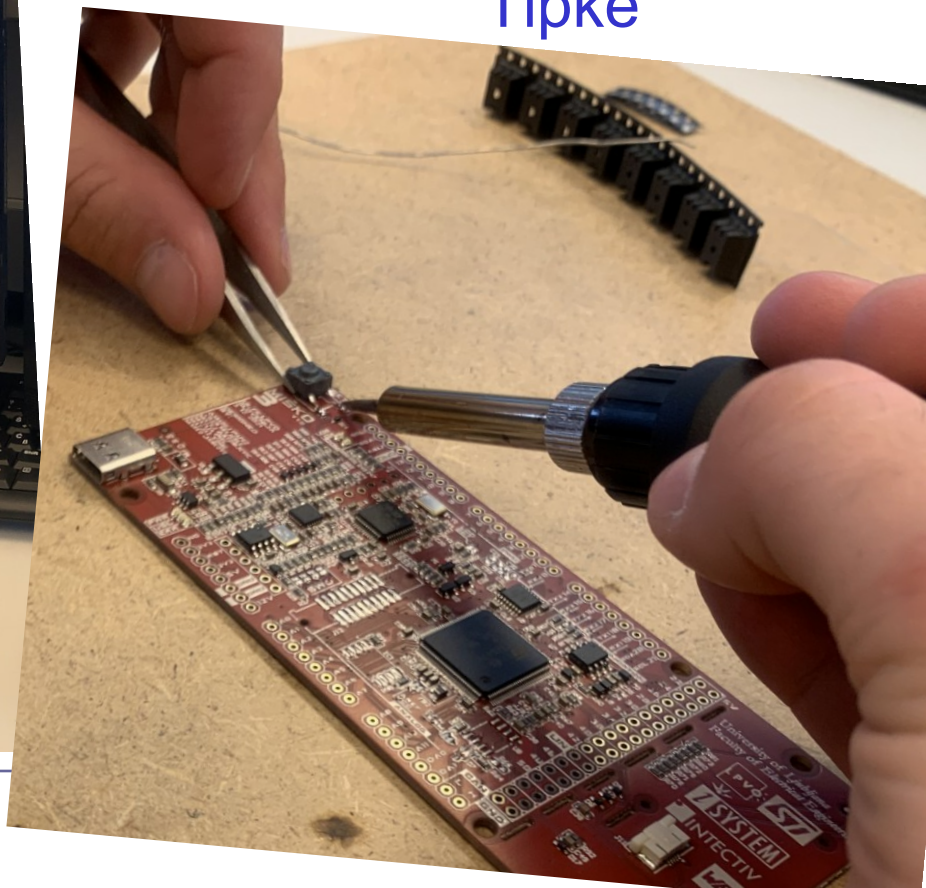
Miško 3 in „Spajka“ party 2022

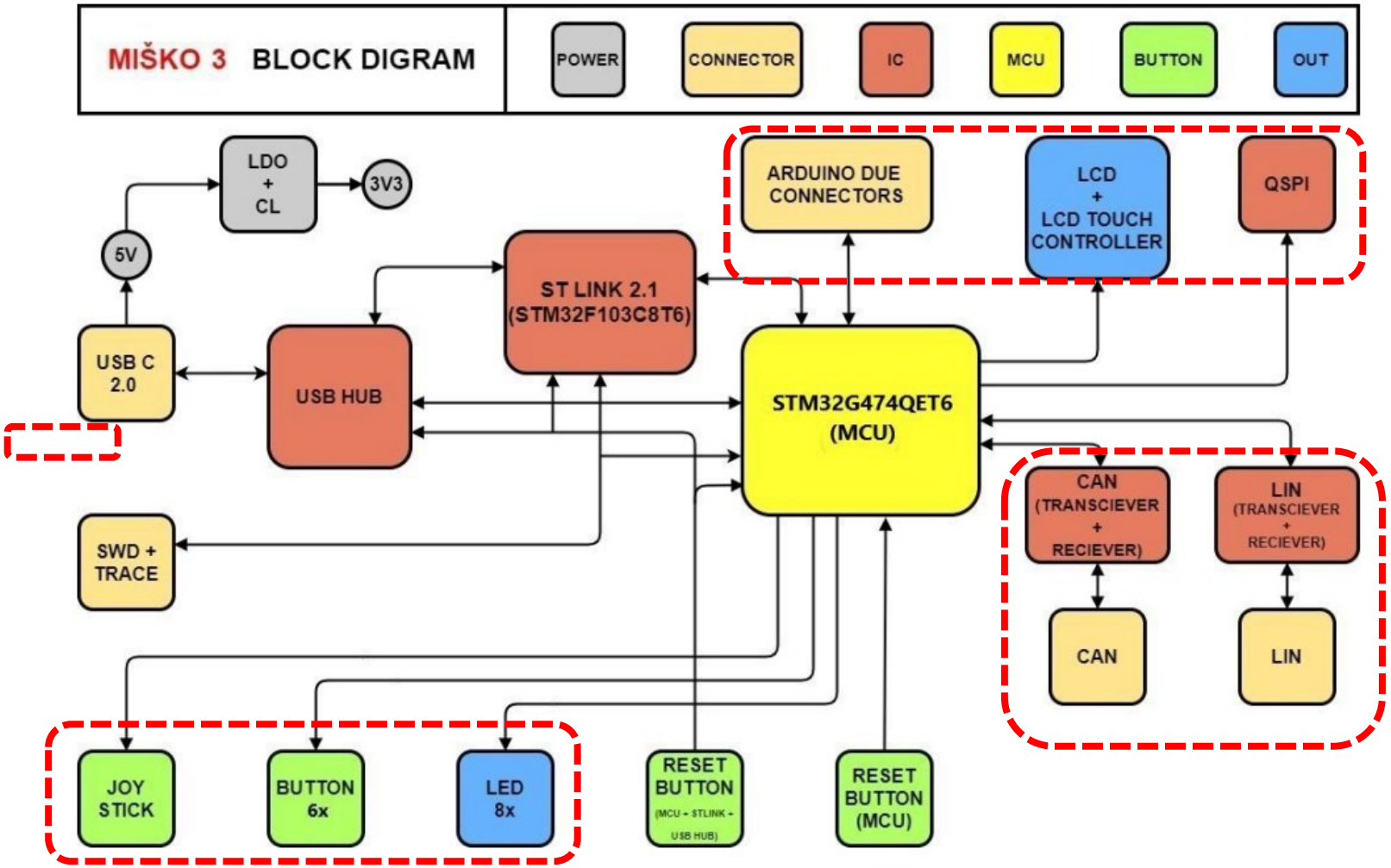


Konektorji



Tipke





Slika 1: Bločni diagram razvojnega Sistema MiŠKo 3

VP – Miško 3 - Inicializacija

```
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 */
```

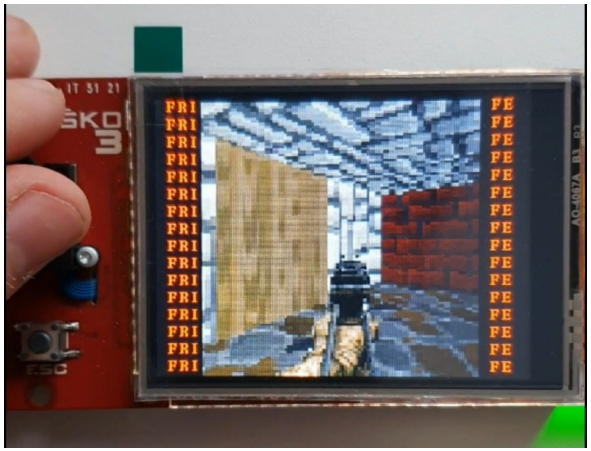
https://github.com/LAPSyLAB/Misko3_Docs_and_Projects

VP – Miško 3 – Glavna zanka

```
 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);
```

```
// 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



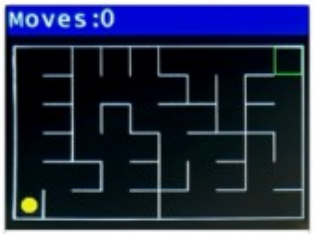
Doom

Maze game

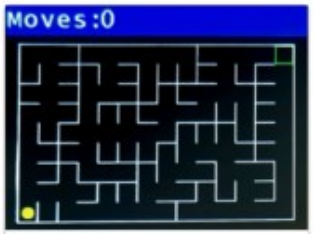
Maze game

Maze game contains a maze, move counter and a yellow circle which represent current player position.

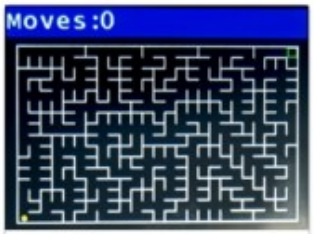
We will implemented three different presets of the maze: easy, normal and hard; which is defined in `MainMenuRefresh()` shown above. Easy dificulty will of size 21x13 and have a cell size of 31, with a (5,43) offset and a player circle radius 7. Normal dificulty will of size 29x19 and have a cell size of 21, with a (12,40) offset and a player circle radius 5. Hard dificulty will of size 51x33 and have a cell size of 12, with a (10,40) offset and a player circle radius 2.



Easy dificulty



Normal dificulty



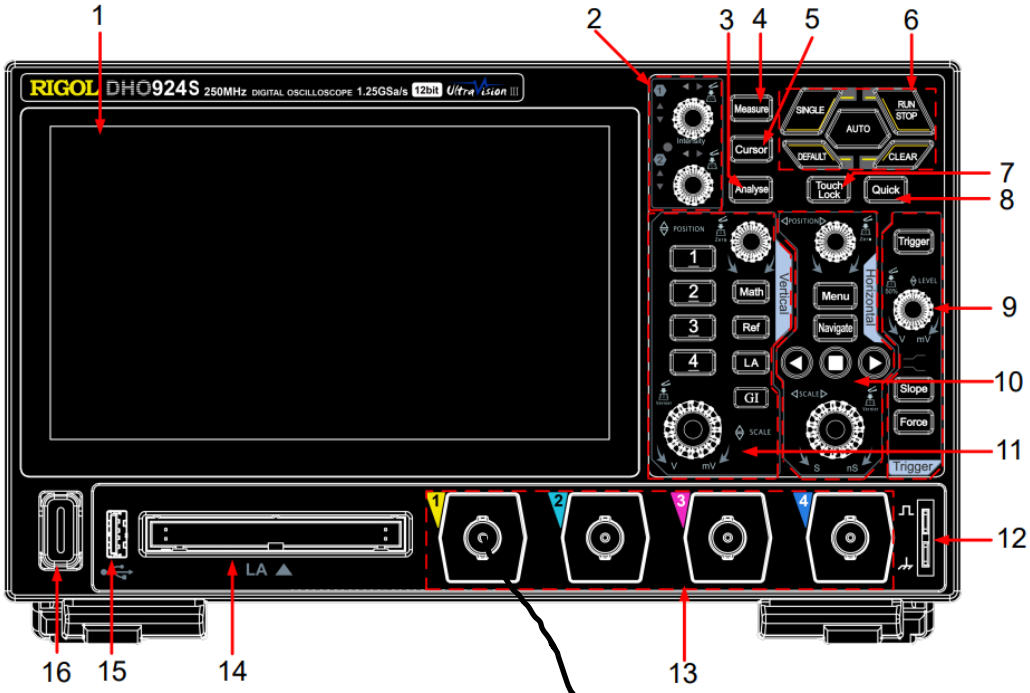
Hard dificulty

https://github.com/LAPSyLAB/Misko3_Docs_and_Projects

VP 5: VIN projekt (Miško3), „Edge AI“, STM32H7 PWM, I2C primeri

- VIN projekt
  - Miško3 – demo projekt
- Osciloskop - spoznavanje
- HAL knjižnica
- STM32 CubeIDE H7 – PWM izhodi, brenčač
- STM32H7 CubeIDE, I2C (Scan, WM9884, Touch)

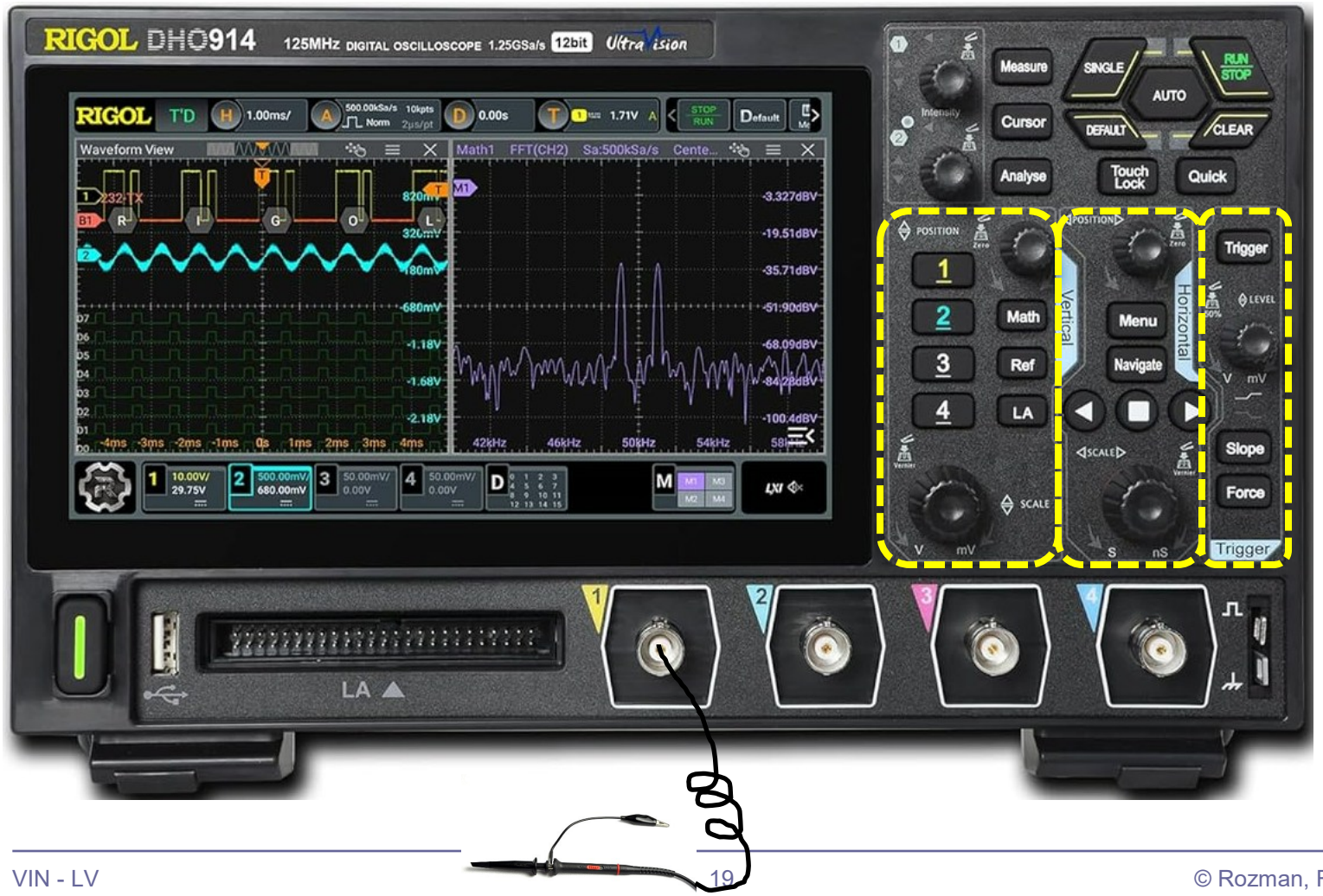
Prednja stran osciloskopa - shema



1	7'' Capacitive Touch Screen	9	Trigger Controls
2	Multipurpose Knobs	10	Horizontal Controls
3	Analyse Key	11	Vertical Controls
4	Measure Key	12	Probe Compensation Signal Output Terminal/Ground Terminal
5	Cursor Key	13	Analog Channel Input Terminals
6	Common Tools Keys	14	Digital Channel Input Terminal
7	Touch Lock Key	15	USB HOST Port
8	Quick Action Key (Self-defined function)	16	Power Key

https://download.rigol.com/en/Manual/Digital%20Oscilloscope/DHO900/DHO900_QuickGuide_EN.pdf

Prednja stran osciloskopa - realna



Prednja stran osciloskopa - kontrole

Y-os (el. napetost)

- **nastavitev merila [V/razdelek]**
- **pozicioniranje y-os**
- **prikaz kanalov da/ne**

X-os (čas)

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

Prožilnik

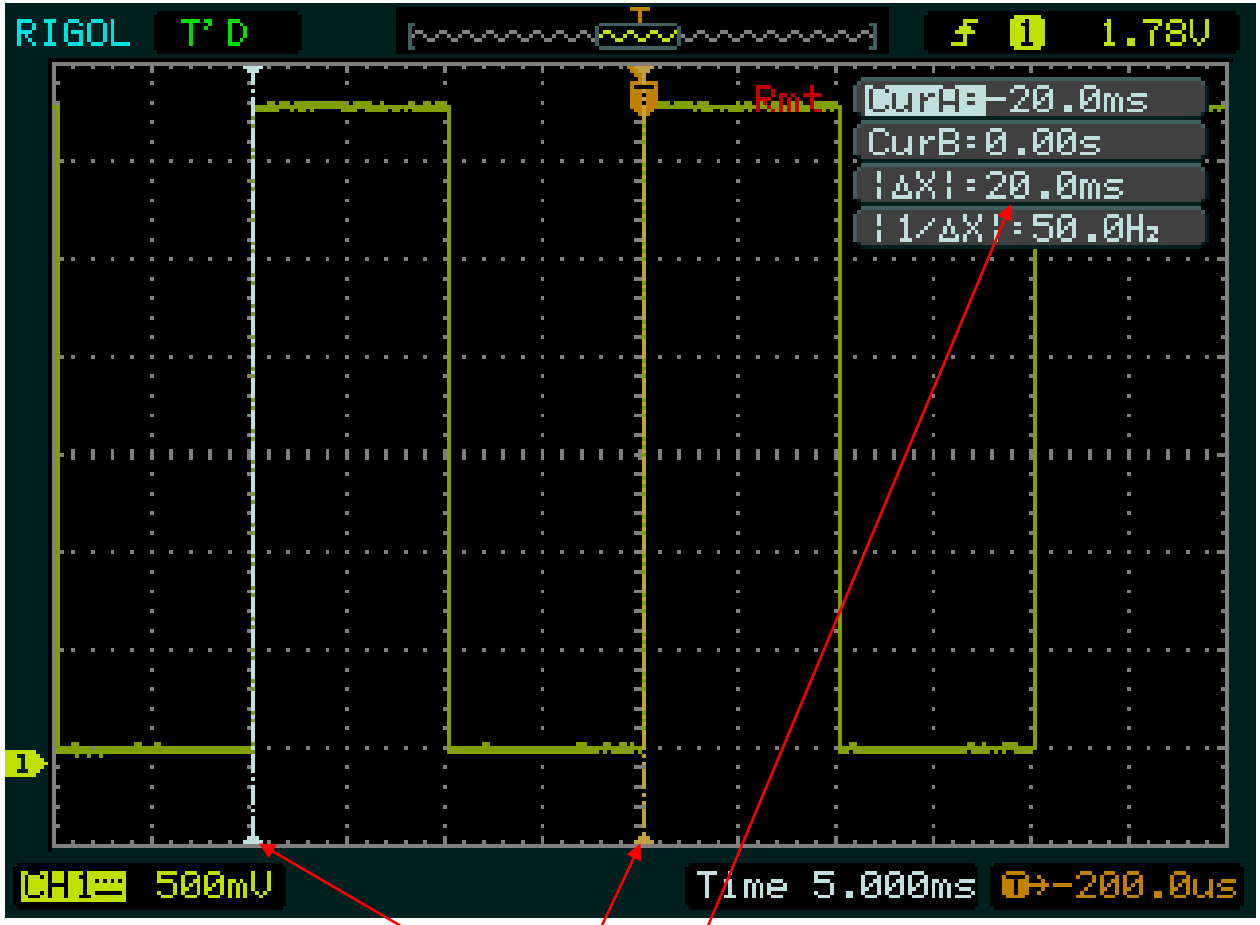
- **začetek prikaza**
- **tipično: poz. fronta in 50%**

https://rigolshop.eu/dho914.html

Spoznavanje merilne opreme…

Meritev testnega signala

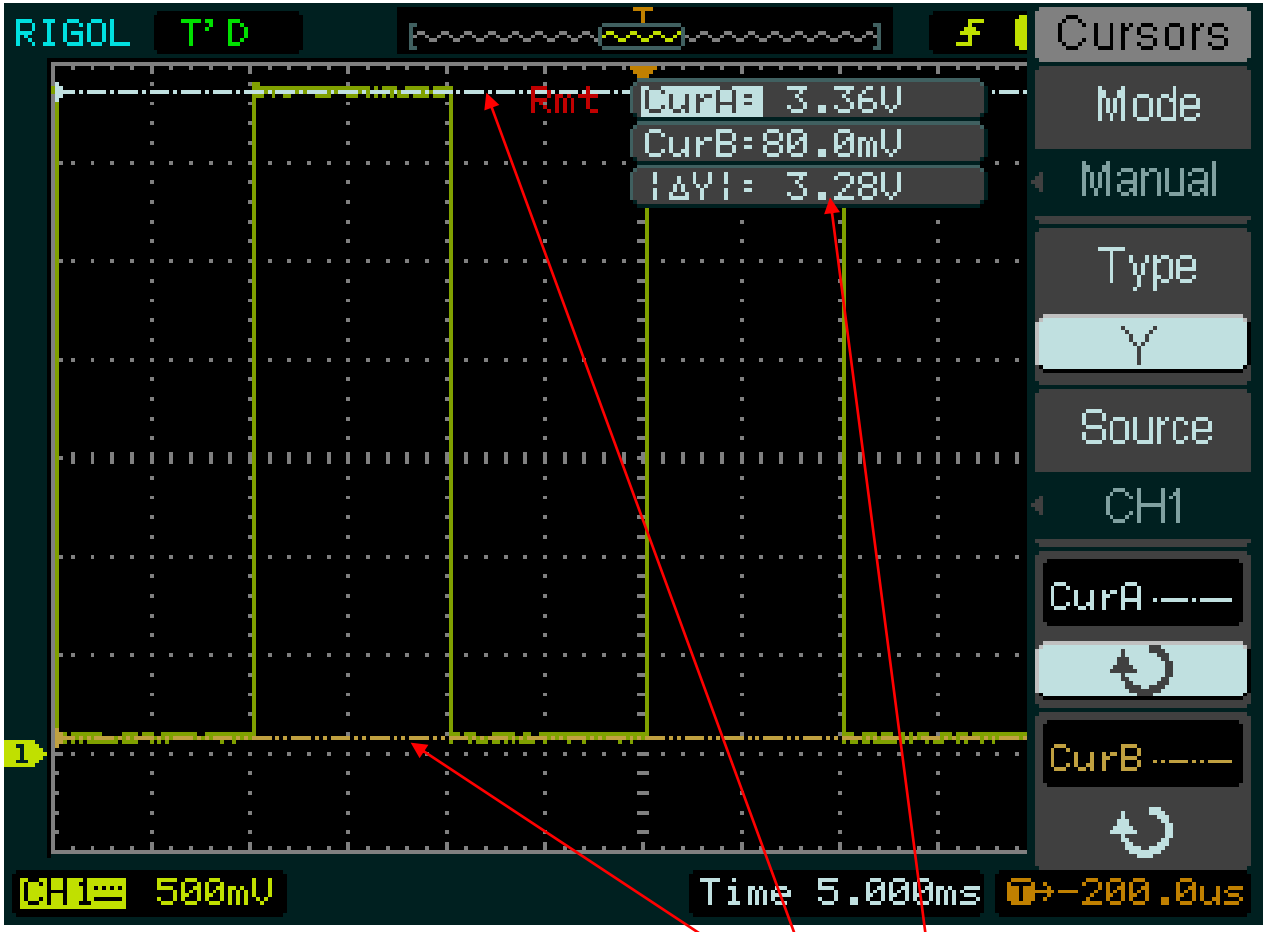
Testni signal – meritev periode, frekvence



Meritev periode/frekvence signala:

- **? ms, ? Hz**

Testni signal – meritev amplitude



Meritev amplitude signala:

- **? V**

VP 5: VIN projekt (Miško3), „Edge AI“, STM32H7 PWM, I2C primeri

- VIN projekt
  - Miško3 – demo projekt
- Osciloskop - spoznavanje
- HAL knjižnica
- STM32 CubeIDE H7 – PWM izhodi, brenčač
- STM32H7 CubeIDE, I2C (Scan, WM9884, Touch)

Osnovni projekt CubeIDE – GPIO – nivoji programiranja

OR – Organizacija računalnikov

VIN

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 */
```

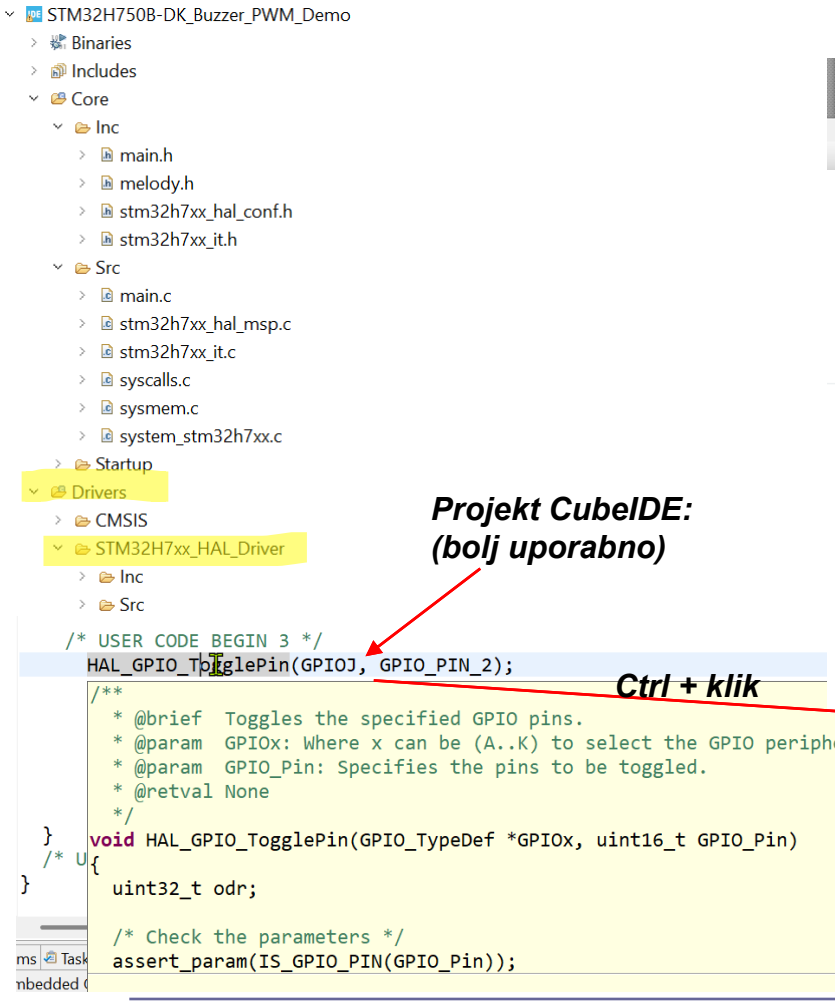
HAL - LL C

```
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

HAL knjižnica – dokumentacija in delo (koda, PDF)

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

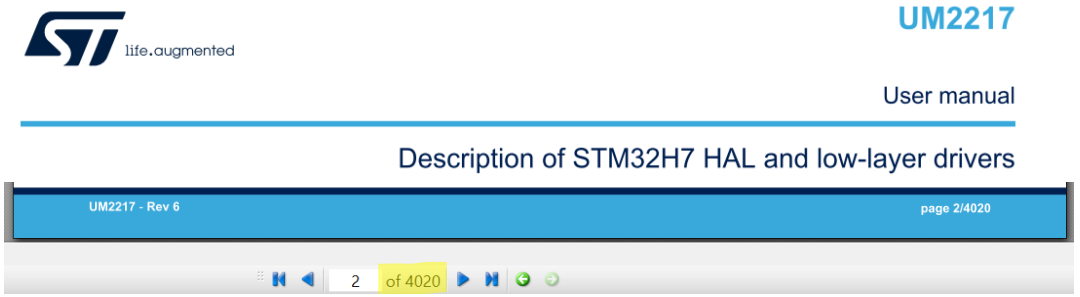


Projekt CubeIDE: (bolj uporabno)

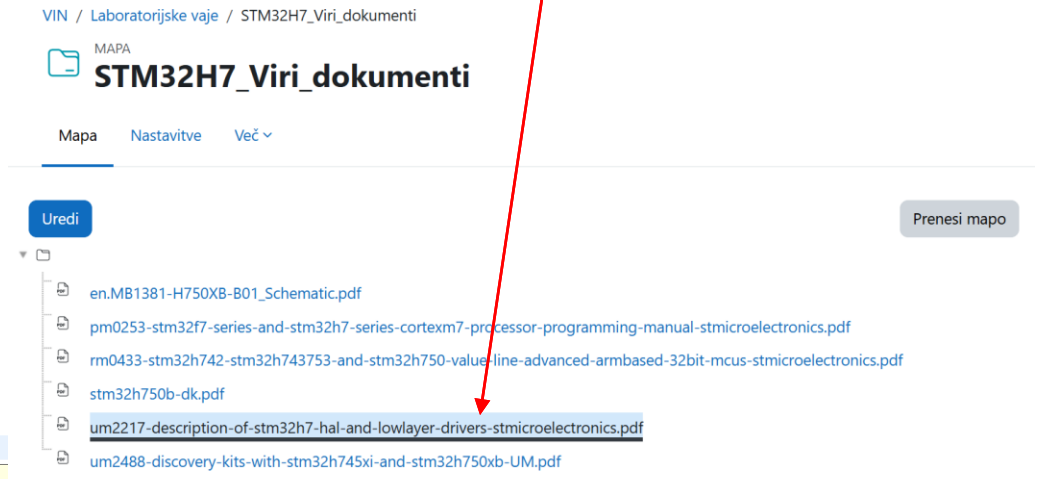
```
/* USER CODE BEGIN 3 */
  HAL_GPIO_TogglePin(GPIOJ, GPIO_PIN_2);
```

Ctrl + klik

```
/**
  * @brief  Toggles the specified GPIO pins.
  * @param  GPIOx: Where x can be (A..K) to select the GPIO peripheral.
  * @param  GPIO_Pin: Specifies the pins to be toggled.
  * @retval None
  */
void HAL_GPIO_TogglePin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)
{
  uint32_t odr;

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

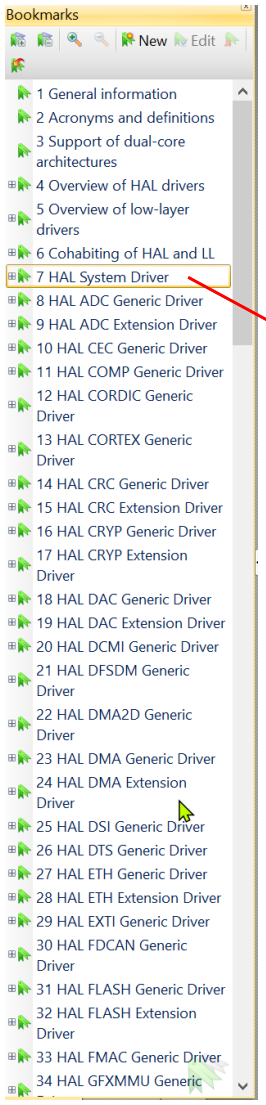


E-učilnica (PDF dokument):




```
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 Output Data Register value */
odr = GPIOx->ODR;
…
```

Vir: UM2217 - HAL and Low-layer drivers - PDF



Programska knjižnica

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

E-učilnica (PDF dokument):

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:

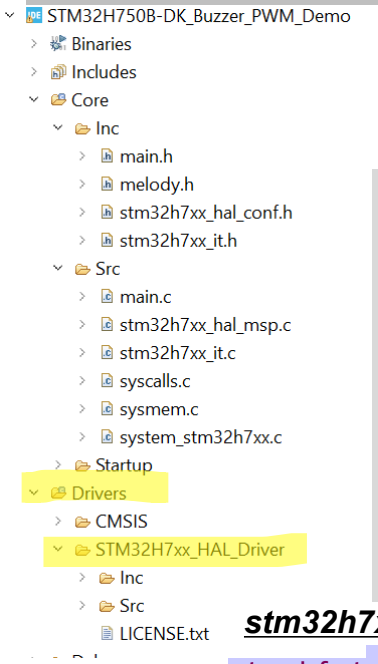
- HAL_GPIO_ReadPin()
- HAL_GPIO_WritePin()
- HAL_GPIO_TogglePin()
- HAL_GPIO_LockPin()
- HAL_GPIO_EXTI_IRQHandler()
- HAL_GPIO_EXTI_Callback()

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()

HAL and Low-layer drivers – delo „skozi“ kodo

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



HAL - LL C

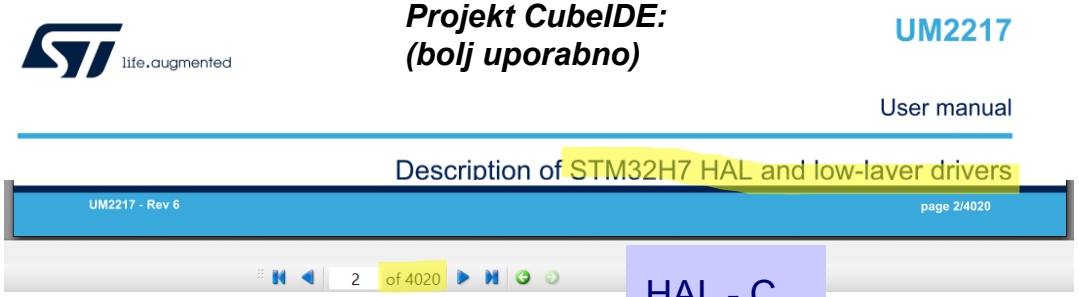
stm32h7xx_ll_gpio.h:

```
/**
* @brief Return full output data register value for a dedicated port.
* @rmtoll ODR ODy LL_GPIO_ReadOutputPort
* @param GPIOx GPIO Port
* @retval Output data register value of port
*/
__STATIC_INLINE uint32_t LL_GPIO_ReadOutputPort(GPIO_TypeDef *GPIOx)
{
return (uint32_t)(READ_REG(GPIOx->ODR));
}

…

#define READ_REG(REG) ((REG))
```

stm32h7xx_gpio.h:

```
typedef struct
{
__IO uint32_t MODER; /*!< GPIO port mode register, Address offset: 0x00 */
__IO uint32_t OTYPER; /*!< GPIO port output type register, Address offset: 0x04 */
__IO uint32_t OSPEEDR; /*!< GPIO port output speed register, Address offset: 0x08 */
__IO uint32_t PUPDR; /*!< GPIO port pull-up/pull-down register, Address offset: 0x0C */
__IO uint32_t IDR; /*!< GPIO port input data register, Address offset: 0x10 */
__IO uint32_t ODR; /*!< GPIO port output data register, Address offset: 0x14 */
__IO uint32_t BSRR; /*!< GPIO port bit set/reset, Address offset: 0x18 */
__IO uint32_t LCKR; /*!< GPIO port configuration lock register, Address offset: 0x1C */
__IO uint32_t AFR[2]; /*!< GPIO alternate function registers, Address offset: 0x20-0x24
*/
} GPIO_TypeDef;
```

Projekt CubeIDE: (bolj uporabno)



HAL - C

stm32h7xx_hal_gpio.c:

```
/**
* @brief Toggles the specified GPIO pins.
* @param GPIOx: Where x can be (A..K) to select the GPIO peripheral.
* @param GPIO_Pin: Specifies the pins to be toggled.
* @retval None
*/
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 Output 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);
}
```

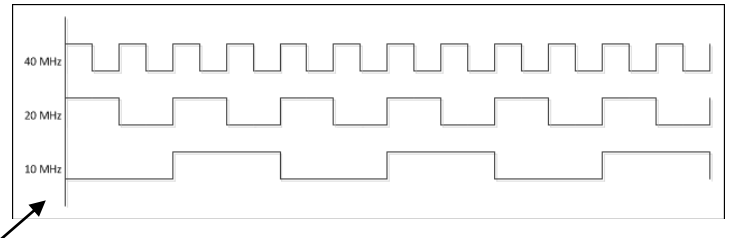
HAL vs HAL-LL (Low Level):

```
/** HAL level **/
/* get current Output Data Register value */
odr = GPIOx->ODR;

/** LL level **/
LL_GPIO_ReadOutputPort(GPIO_TypeDef *GPIOx)
```

VP 5: VIN projekt (Miško3), „Edge AI“, STM32H7 PWM, I2C primeri

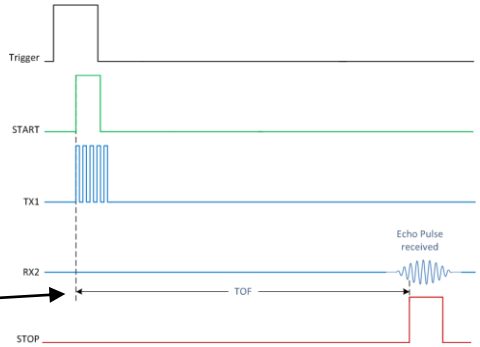
- VIN projekt
  - Miško3 – demo projekt
- AI v vgrajenih napravah („Edge Computing“)
- HAL knjižnica
- STM32 CubeIDE H7 – PWM izhodi, brenčač
- STM32H7 CubeIDE, I2C (Scan, WM9884, Touch)

Timer - Counter (časovnik - števec)



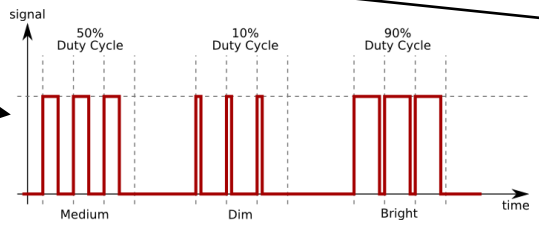
- Običajno več enakovrednih kanalov

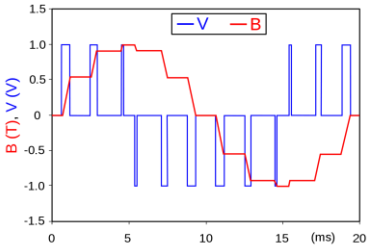
- Uporabni za
  - štetje dogodkov (Capture)
  - tvorjenje časovnih signalov (Waveform)
  - zakasnitve (DELAY s časovnikom !)
  - merjenje intervalov
  - periodične prekinitve
  - tvorba signalov s pulzno širinsko modulacijo (PWM)



Variacije „Duty Cycle“:

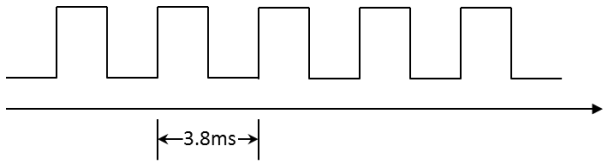
- LED „dimmer“
- krmiljenje hitrosti motorjev
- „enostavni“ DAC – povprečje
- kodiranje podatkov





Variacije periode:

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



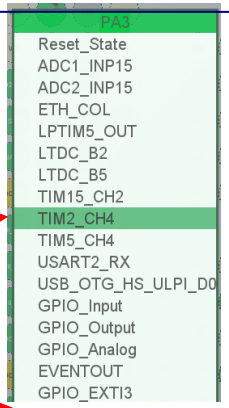
STM32H7 – PWM signal na priključku PA3

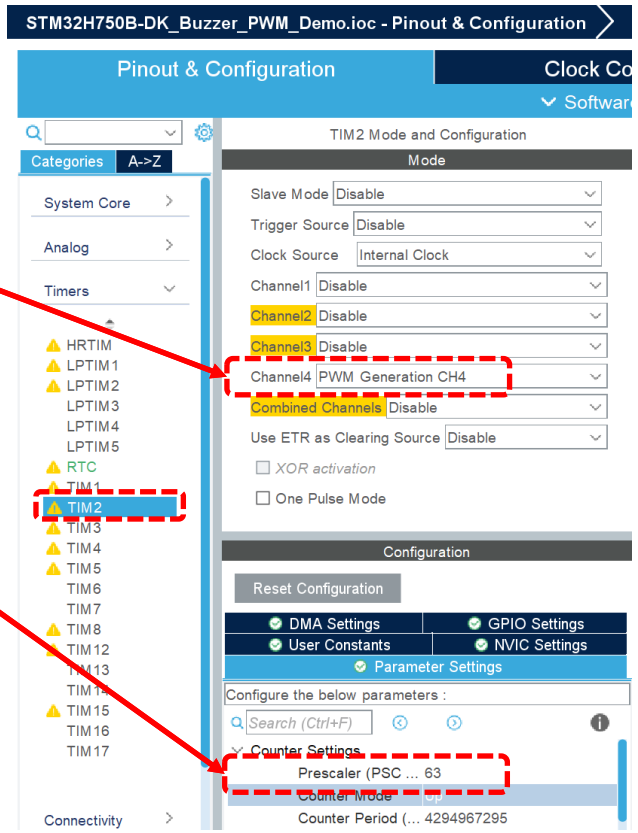
PWM signal lahko spremljamo na priključku **PA3-PWM na STMod+ Clickboard.**
Uporabili bomo TIM2->CH4 funkcionalnost na PA3

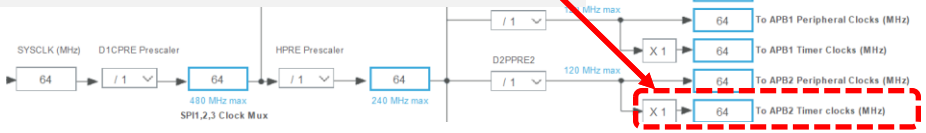
CubeMX nastavitve:

1. Spremeniti pin PA3 v TIM2->CH4
TIM2 kanal 4 spremenimo na PWM Generation CH4

2. Clock & TIM2:
Nastavimo urin signal števca na 1 MHz
Obstoječ urin signal 64Mhz delimo s 64:
Prescaler (PSC - 16 bits value) = 64-1 = 63







Časovnik bo v eni sekundi preštel do enega miljona.
PWM signal določimo s dvema parametroma:
1. Counter Period (AutoReload Register - 16 bits value) =
ARR = 1000000 / številom period PWM signala[Hz]
2. CCR1 določa vrednost znotraj periode za preklop stanja PWM
CCR1 = 0 .. ARR (npr. ARR/2 pomeni 50% duty cycle

Več informacij : BeriMe.txt

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

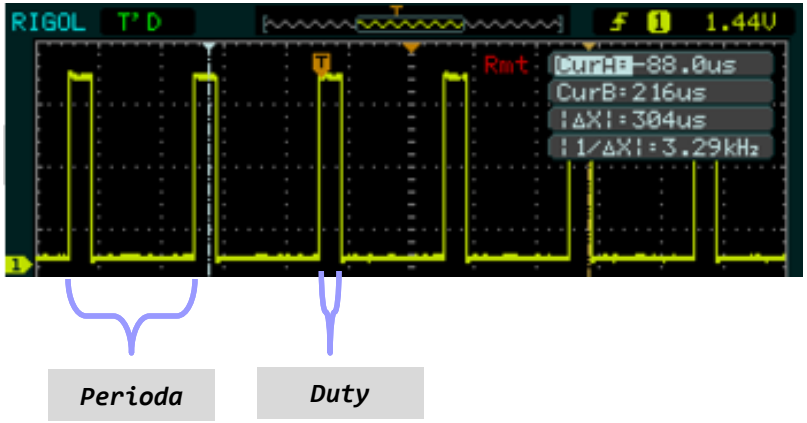
STM32H7 – PWM signal na priključku PA3

Časovnik bo v eni sekundi preštel do enega miljona.
PWM signal določimo s dvema parametroma:
1. Counter Period (AutoReload Register - 16 bits value) =
ARR = 1000000 / številom period PWM signala[Hz]
2. CCR1 določa vrednost znotraj periode za preklop stanja PWM
CCR1 = 0 .. ARR (npr. ARR/2 pomeni 50% duty cycle

Več informacij : BeriMe.txt

```
/**
* @brief The application entry point.
* @retval int
*/
int main(void)
{
/* USER CODE BEGIN 1 */
int ARR_period;
int PWMFreq, PWMDuty;

/* Initialize all configured peripherals */
…

MX_TIM2_Init();

/* USER CODE BEGIN 2 */
HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_4);
/* USER CODE END 2 */


/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
ARR_period = (int)(1000000/PWM_Freq); //Already prescaled to 1 MHz
setPWM(htim2, TIM_CHANNEL_4, ARR_period, PWM_Duty * ARR_period/100);
…
```

```
/* 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 */
```



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

STM32H7 – PWM melodija za brenčača (Buzzer)

Ena nota (frekvenca, trajanje):

```
ARR_period = (int)(1000000/NoteFreq);  //Already prescaled to 1 MHz
setPWM(htim2, TIM_CHANNEL_4, ARR_period, ARR_period/2);

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

HAL - C

Perioda signala PWM se bo določala glede na trenutno noto.
Duty cycle je vedno 50% - približek sinusnega signala.

1. Counter Period (AutoReload Register - 16 bits value) =
   ARR = 1000000 (ura števca) / Frekv.note[Hz]
2. CCR1 bo vedno ARR/2 (50% duty cycle)

Več informacij : BeriMe.txt

Melodija (zaporedje not s trajanji):

```
void PlayMelodies_PWM()
{
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_4, 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);
HAL_UART_Transmit(&huart3,SendBuffer,strlen(SendBuffer),100);

snprintf (SendBuffer,BUFSIZE,"\r\n\r\nEnd of All Melodies[%d]\r\n\r\n",melodyIndex);
HAL_UART_Transmit(&huart3,SendBuffer,strlen(SendBuffer),1);

HAL_Delay(2000);
}
```

Primer: Nota A4 = 440 Hz
Perioda[s]=1/440Hz=0.00227272
Perioda[us]=1/440*1000000=2272 = ARR_period

melody.h (zapisi melodij)

```
//###############**"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,
  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##############//
```

```
void PlayMelodies_PWM()
{
```


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

STM32H7 – PWM signali/melodija za brenčača (Buzzer)

```
void PlayMelodies_PWM()
{

 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_4, ARR_period, ARR_period/2);

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

                 HAL_Delay(Delaymsecs);
                 snprintf (SendBuffer,BUFSIZE,"Melody[%d],Note #%d F=%d Hz Duration:%d ms| ARR=%d
                   CCR1=%d\r\n",melodyIndex,noteIndex,melody[melodyIndex][noteIndex],Delaymsecs,hti
                   m2.Instance->ARR,htim2.Instance->CCR1);
                 HAL_UART_Transmit(&huart3,SendBuffer,strlen(SendBuffer),100);

           }

 }
```

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##############//
```

COM4 - PuTTY

```
Melody[0],Note #37 F=2794 Hz Duration:180 ms| ARR=357 CCR1=0
Melody[0],Note #38 F=3136 Hz Duration:180 ms| ARR=318 CCR1=0
Melody[0],Note #39 F=0 Hz Duration:180 ms| ARR=16960 CCR1=0
Melody[0],Note #40 F=2637 Hz Duration:180 ms| ARR=379 CCR1=0
Melody[0],Note #41 F=0 Hz Duration:180 ms| ARR=16960 CCR1=0
Melody[0],Note #42 F=2093 Hz Duration:180 ms| ARR=477 CCR1=0
Melody[0],Note #43 F=2349 Hz Duration:180 ms| ARR=425 CCR1=0
Melody[0],Note #44 F=1976 Hz Duration:180 ms| ARR=506 CCR1=0
Melody[0],Note #45 F=0 Hz Duration:180 ms| ARR=16960 CCR1=0
Melody[0],Note #46 F=0 Hz Duration:180 ms| ARR=16960 CCR1=0
Melody[0],Note #47 F=2093 Hz Duration:180 ms| ARR=477 CCR1=0
Melody[0],Note #48 F=0 Hz Duration:180 ms| ARR=16960 CCR1=0
```

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)};
```

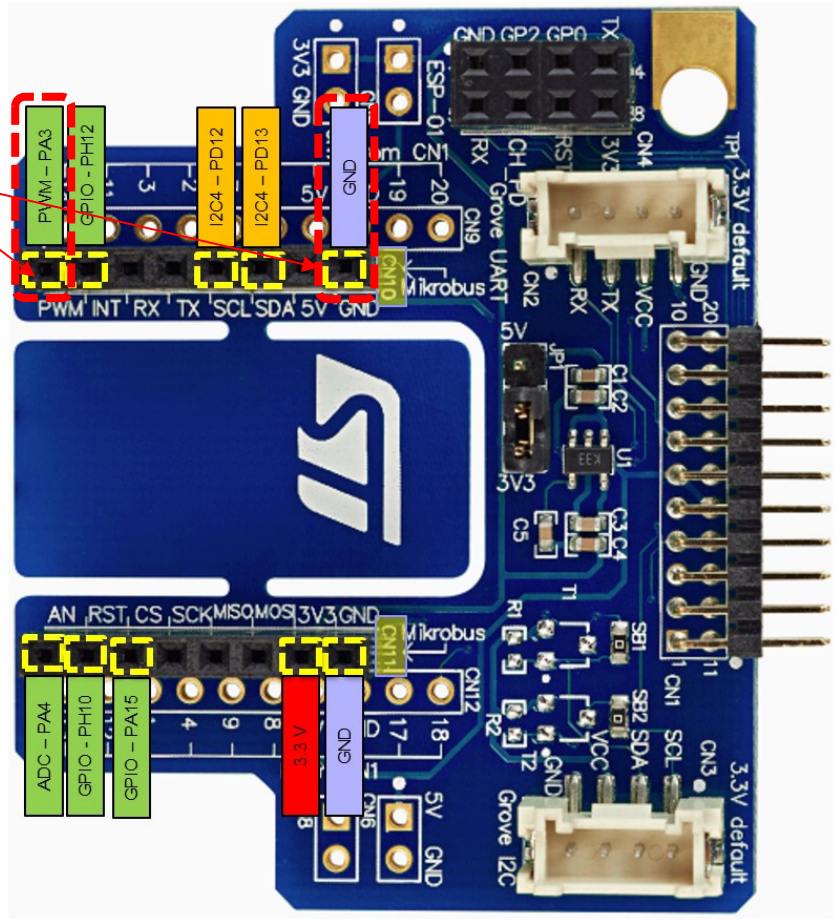
https://github.com/LAPSyLAB/STM32H7_Discovery_VIN_Projects/tree/main/STM32H750B-DK_Buzzer_PWM_Demo

STM32H7 – PWM signali/melodija za brenčača (Buzzer)

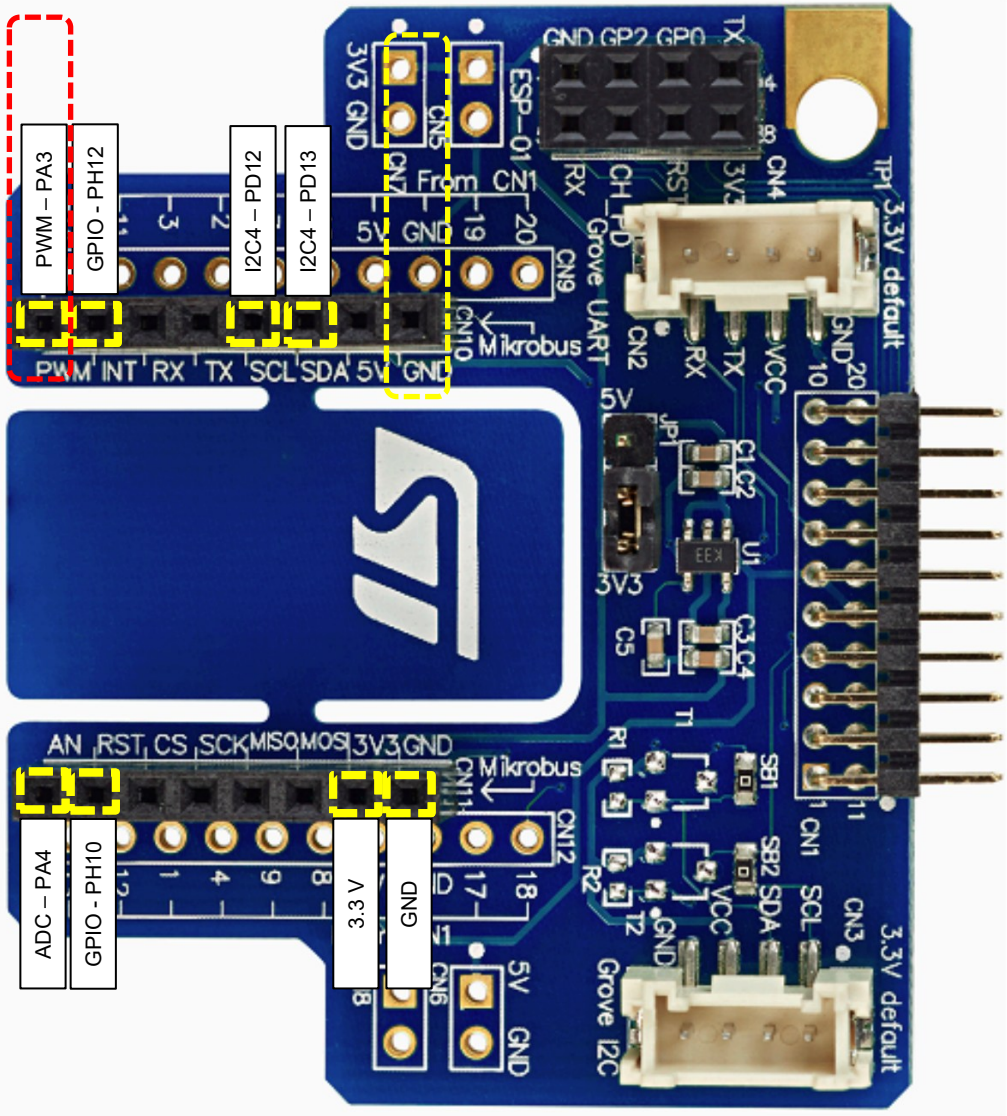
Priključitev na STM32 : 1x analogni, 1x digitalni vhod, 1x digitalni izhod, 3x vgrajene LED diode

Povezava brenčača - STM32H7:

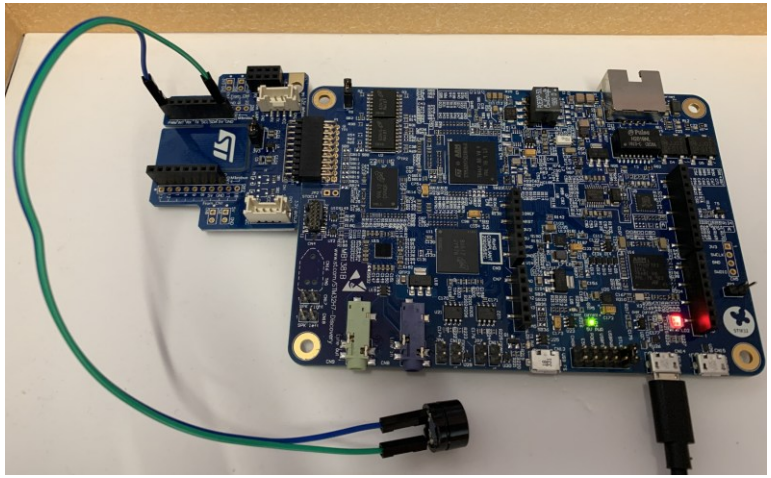
GPIO	Vrsta	Povezava
PA3	Brenčač	+
GND	Brenčač	-



STM32H7 – PWM signali/melodija za brenčača (Buzzer)



Pravilna priključitev



Nepravilna priključitev

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

STM32F4 – PWM signali za LED diode (LED dimmer)

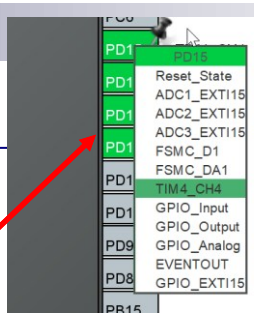
HAL - C

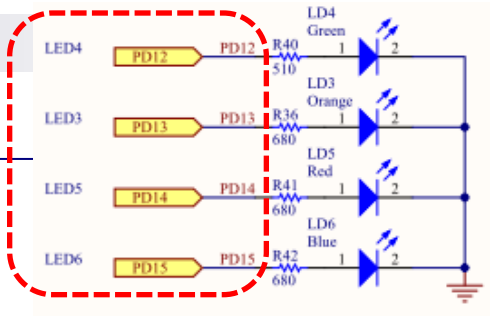
CubeMX :

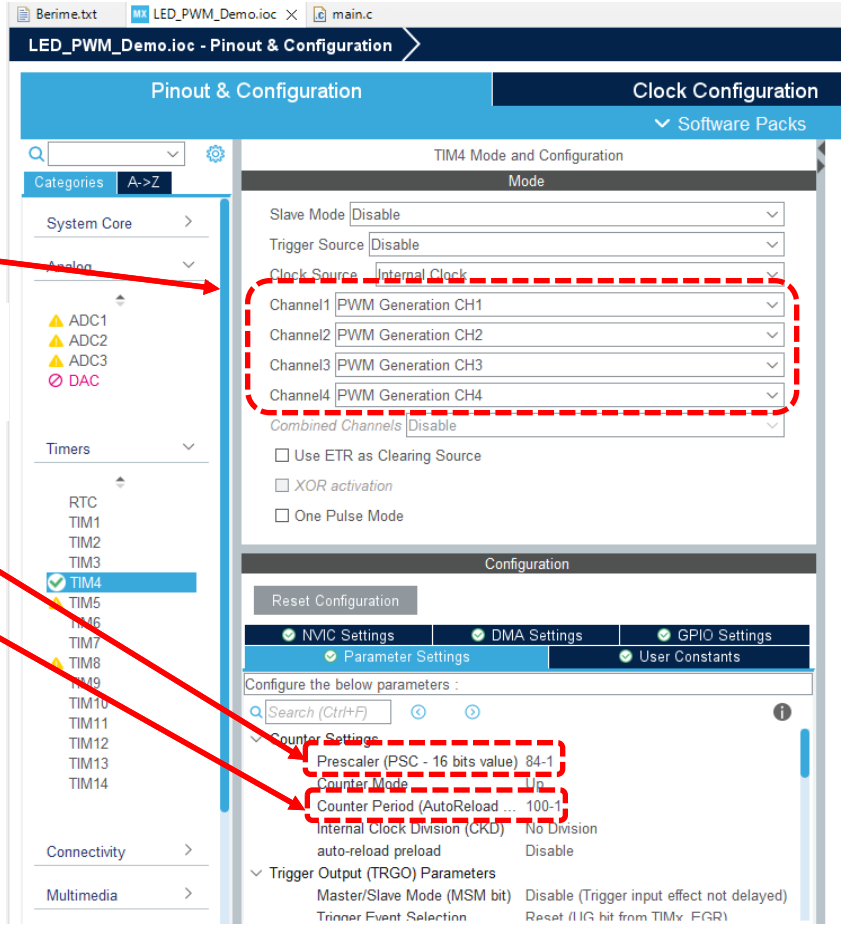
1. New project -> STM32 Project -> Board -> 407DISC1
2. CubeMX: Spremeniti USB Host v USB Device :
   - Connectivity -> USB_OTG_FS -> Mode v Device Only
   - Middleware -> DEVICE_USB in Class Virtual Com Port

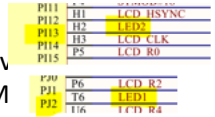
3. Spremeniti pine PD12-PD15 (LED diode) v TIM4_CH1-4
   - tim4 Vse kanale spremeniti na PWM Generation CH1-4

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









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 */
```

CubeMX - dodatne spremembe osnovnega projekta :

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_CH0-3
   tim4 Vse kanale spremeniti na PWM Generation CH0-3

4. Clock :
   Ura števca = 1 MHz
   Prescaler (PSC - 16 bits value) Prescaler (PSC - 16 bits value) must be between 0 and 65 535 = 84-1 = 83 (clock 1Mhz)

   Perioda štetja je 100 (duty cycle pa lahko 0-100)
   Counter Period (AutoReload Register - 16 bits value) Counter Period (AutoReload Register - 16 bits value) = 100-1 = 99

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

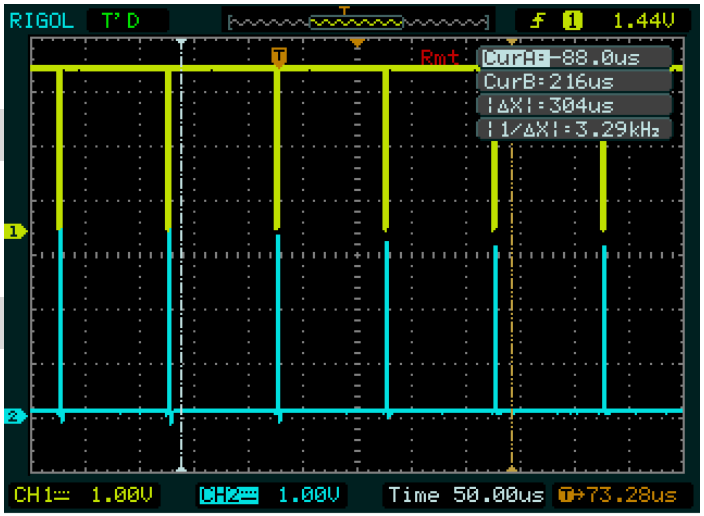
Osnovni projekt CubeIDE – GPIO – PWM, LED diode

HAL - C

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

/* 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 */
```

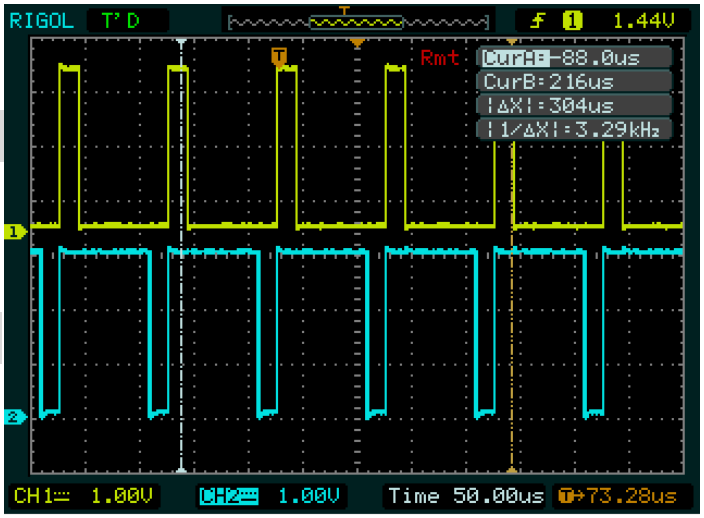
Max Duty

Min Duty



Min Duty

Max Duty

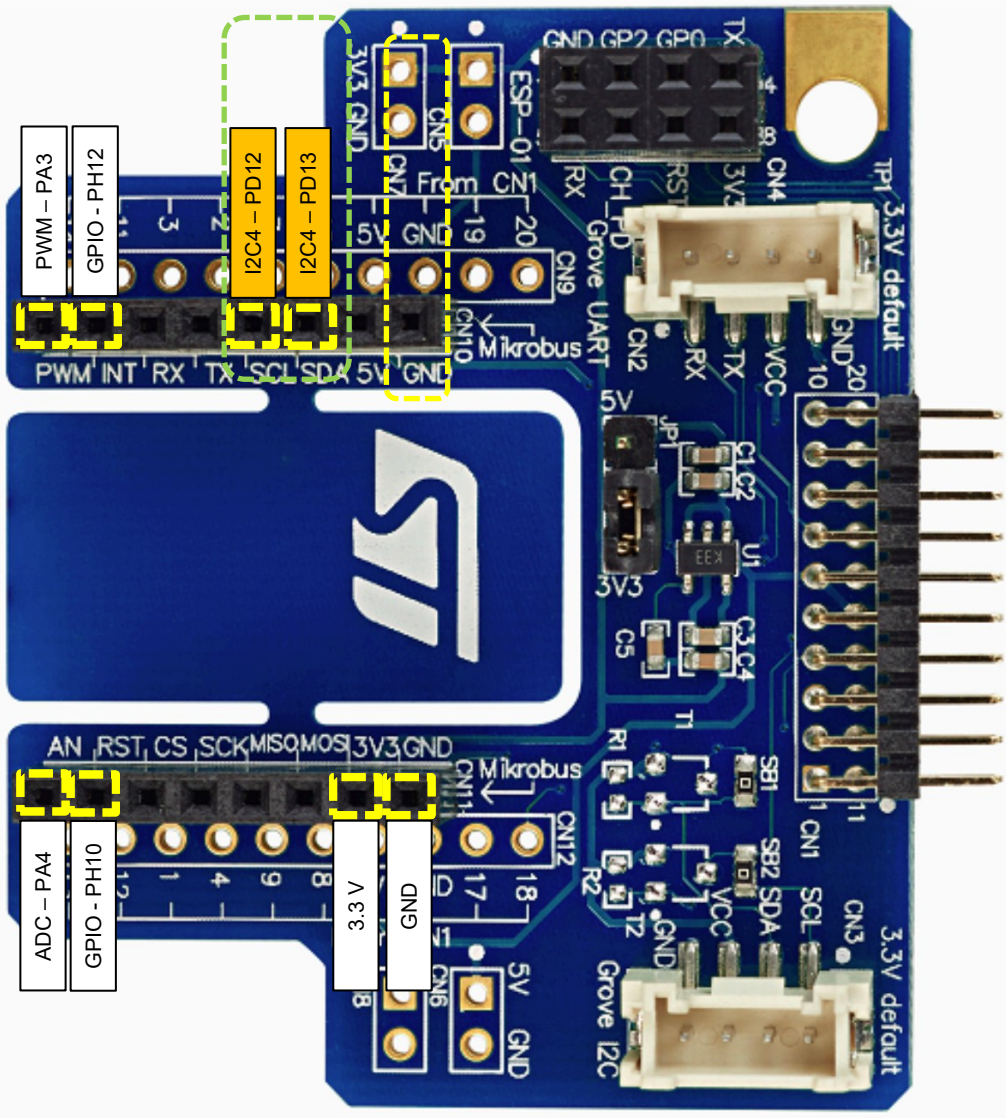


https://github.com/LAPSyLAB/STM32F4_Discovery_VIN_Projects/tree/main/LED_PWM_Demo

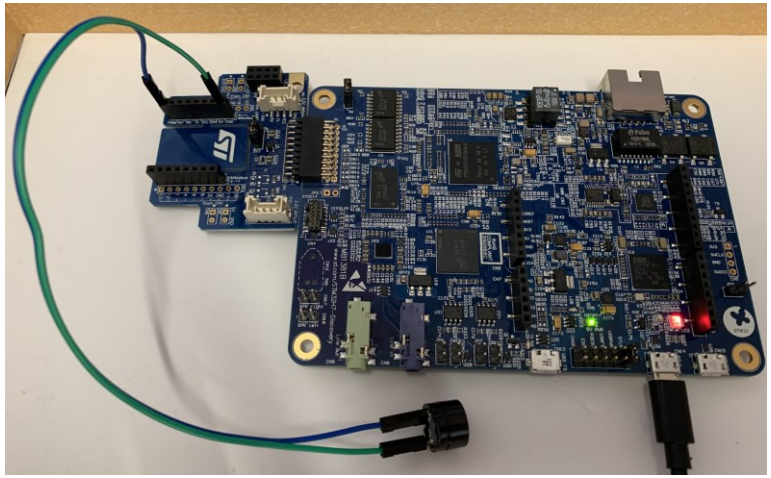
VP 5: VIN projekt (Miško3), „Edge AI“, STM32H7 PWM, I2C primeri

- VIN projekt
  - Miško3 – demo projekt
- Osciloskop - spoznavanje
- HAL knjižnica
- STM32 CubeIDE H7 – PWM izhodi, brenčač
- STM32H7 CubeIDE, I2C (Scan, WM9884, Touch)

STM32H7 – PWM signali/melodija za brenčača (Buzzer)



Pravilna priključitev



Nepravilna priključitev

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

VP 4 - STM32H7 CubeIDE, I2C4 scanner

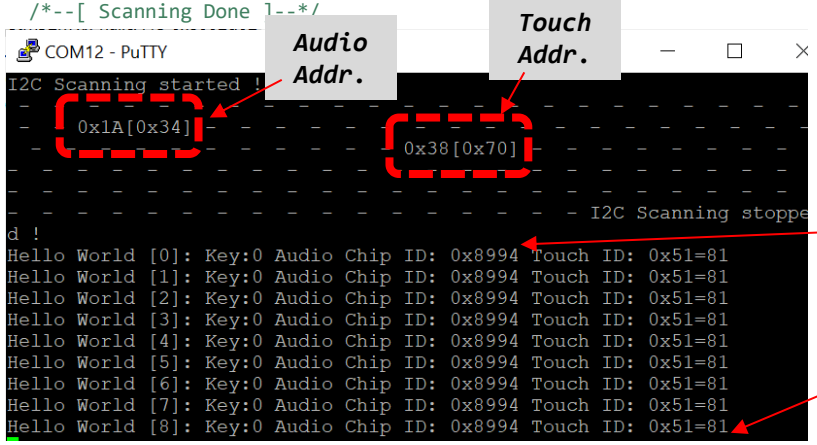
main.c : dodana koda

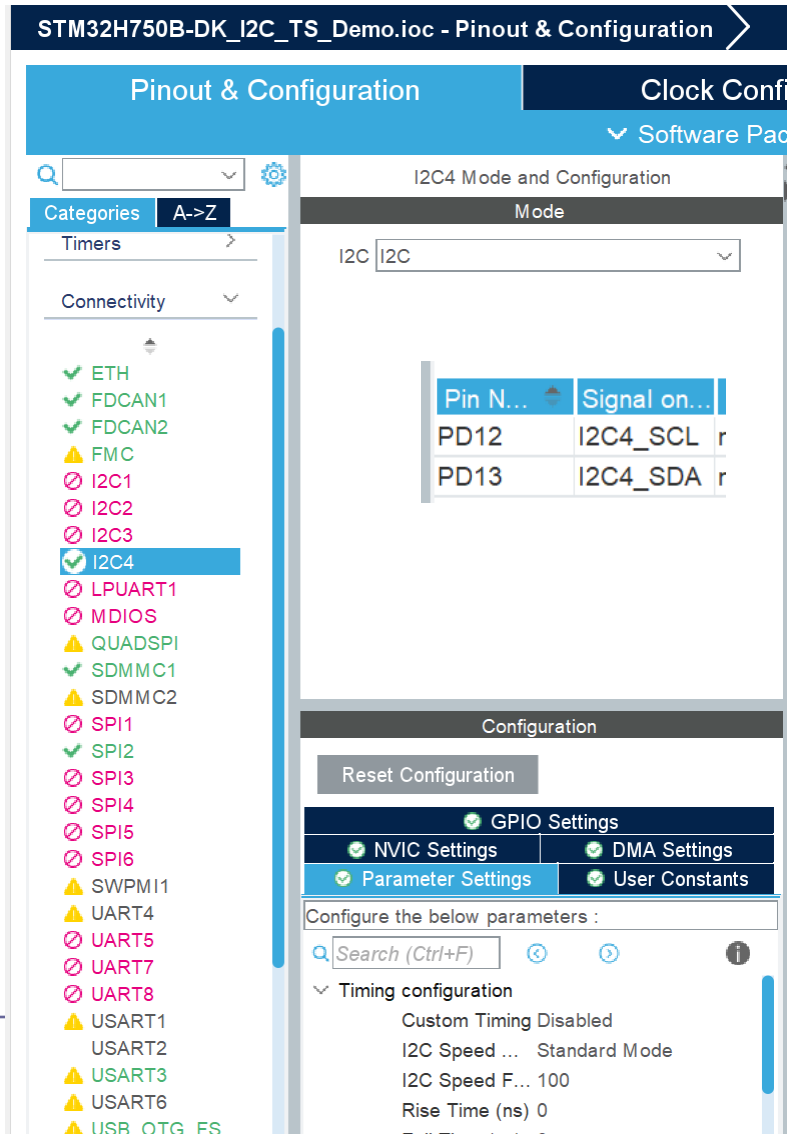
Spremenljivke

```
/* USER CODE BEGIN PV */
#define BUFSIZE 256
charSendBuffer[BUFSIZE];
intCounter;
int KeyState=0;
uint8_t dataBuffer[10];

HAL_StatusTypeDef retval;
uint8_t Answer;
```

I2C Scan (vseh naprav)

```
HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12, 1);     // Set LCD_RST to high

 /*-[ I2C Bus Scanning ]-*/
snprintf(SendBuffer,BUFSIZE,"I2C Scanning started !\n\r");
HAL_UART_Transmit(&huart3,SendBuffer,strlen(SendBuffer),100);

 for(i=1; i<128; i++)
 {
        retval = HAL_I2C_IsDeviceReady(&hi2c4, (uint16_t)(i<<1), 3, 5);
        if (retval != HAL_OK) /* No ACK Received At That Address */
        {
                HAL_UART_Transmit(&huart3, Space, sizeof(Space), 100);
        }
        else if(retval == HAL_OK)
        {
                snprintf(SendBuffer,BUFSIZE,"0x%02X[0x%02X]", i, i<<1);

                HAL_UART_Transmit(&huart3,SendBuffer,strlen(SendBuffer),1);
        }
 }
snprintf(SendBuffer,BUFSIZE,"I2C Scanning stopped !\n\r");
HAL_UART_Transmit(&huart3,SendBuffer,strlen(SendBuffer),100);
   /*--[ Scanning Done ]--*/
```





STM32H7

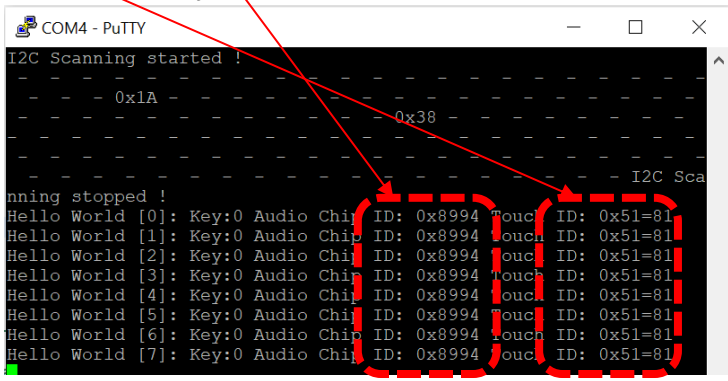
main.c : dodana koda

Spremenljivke

```
/* USER CODE BEGIN PV */
#define BUFSIZE 256
charSendBuffer[BUFSIZE];
intCounter;
int KeyState=0;
uint8_t dataBuffer[10];
int i=0;
uint8_t Space[] = " - ";

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

Glavna zanka

```
 /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    HAL_GPIO_TogglePin(GPIOI, GPIO_PIN_13);

    KeyState = HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13);
    HAL_GPIO_WritePin(GPIOJ, GPIO_PIN_2, KeyState);

…

// Reading from address 0x1a register R0 (addr. 0x00) default value should be 0x8994 - Both variations work !
    //dataBuffer[0] = 0; dataBuffer[1] = 0x00;
    //retval = HAL_I2C_Master_Transmit(&hi2c4, (0x1a << 1), dataBuffer, 2, HAL_MAX_DELAY);
    //retval = HAL_I2C_Master_Receive(&hi2c4, (0x1a << 1), dataBuffer, 2, HAL_MAX_DELAY);
    retval = HAL_I2C_Mem_Read(&hi2c4, (0x1a << 1), 0, I2C_MEMADD_SIZE_16BIT,dataBuffer, 2, HAL_MAX_DELAY);

// Reading from address 0x38 register Vendor's Chip ID (addr. 0xA8) default value should be 0x51=81  - Both variations work !
    //dataBuffer[5] = 0xA8;
    //retval = HAL_I2C_Master_Transmit(&hi2c4, (0x38 << 1), &dataBuffer[5], 1, HAL_MAX_DELAY);
    //retval = HAL_I2C_Master_Receive(&hi2c4, (0x38 << 1), &dataBuffer[5], 1, HAL_MAX_DELAY);
    retval = HAL_I2C_Mem_Read(&hi2c4, (0x38 << 1), 0xA8, I2C_MEMADD_SIZE_8BIT,&dataBuffer[5], 1, HAL_MAX_DELAY);


    snprintf(SendBuffer,BUFSIZE,"Hello World [%d]: Key:%d Audio Chip ID: 0x%4x Touch ID: 0x%2x=%d\n\r",Counter++,KeyState,
dataBuffer[0]*256+dataBuffer[1],dataBuffer[5],dataBuffer[5]);
    HAL_UART_Transmit(&huart3,SendBuffer,strlen(SendBuffer),100);

    HAL_Delay(1000);
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */

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

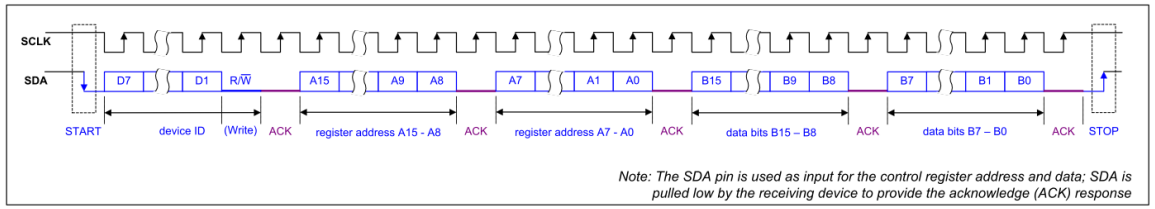


VP 4 - STM32H7 CubeIDE, I2C Audio WM9884 Gradiva

16-bitni naslovi in 16-bitni registri !

2-WIRE (I2C) CONTROL MODE

The sequence of signals associated with a single register write operation is illustrated in Figure 72.



[Brez naslova]

Figure 72 Control Interface 2-wire (I2C) Register Write

The sequence of signals associated with a single register read operation is illustrated in Figure 73.

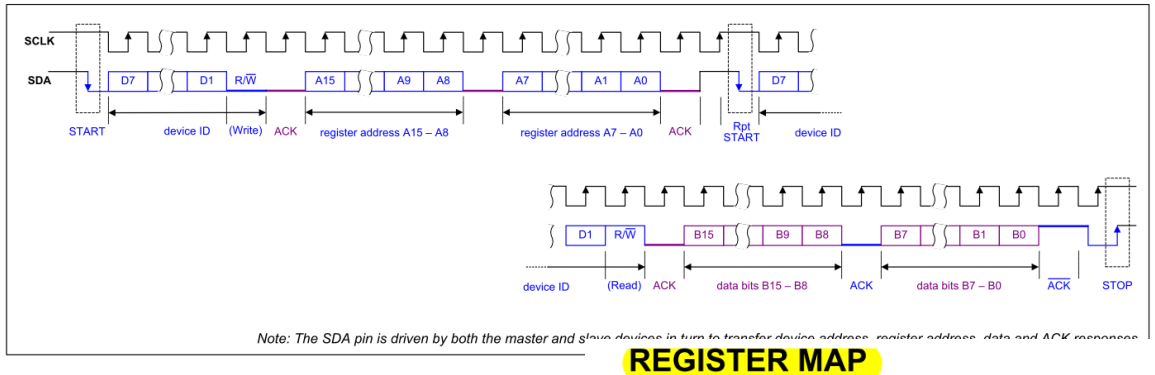
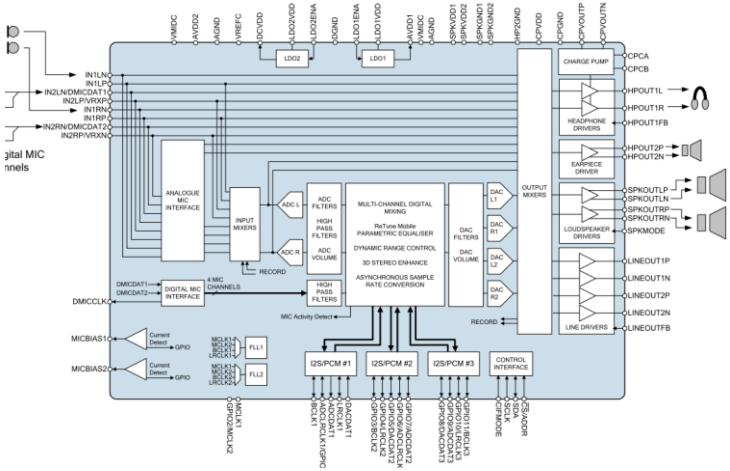


Figure 73 Control Interface 2-wire (I2C) Register Read



https://github.com/LAPSyLAB/STM32H7_Discovery_VIN_Projects/tree/main/STM32H750B-DK_I2C_TS_Demo

REGISTER MAP

The WM8994 control registers are listed below. Note that only the register addresses described here should be accessed; writing to other addresses may result in undefined behaviour. Register bits that are not documented should not be changed from the default values.

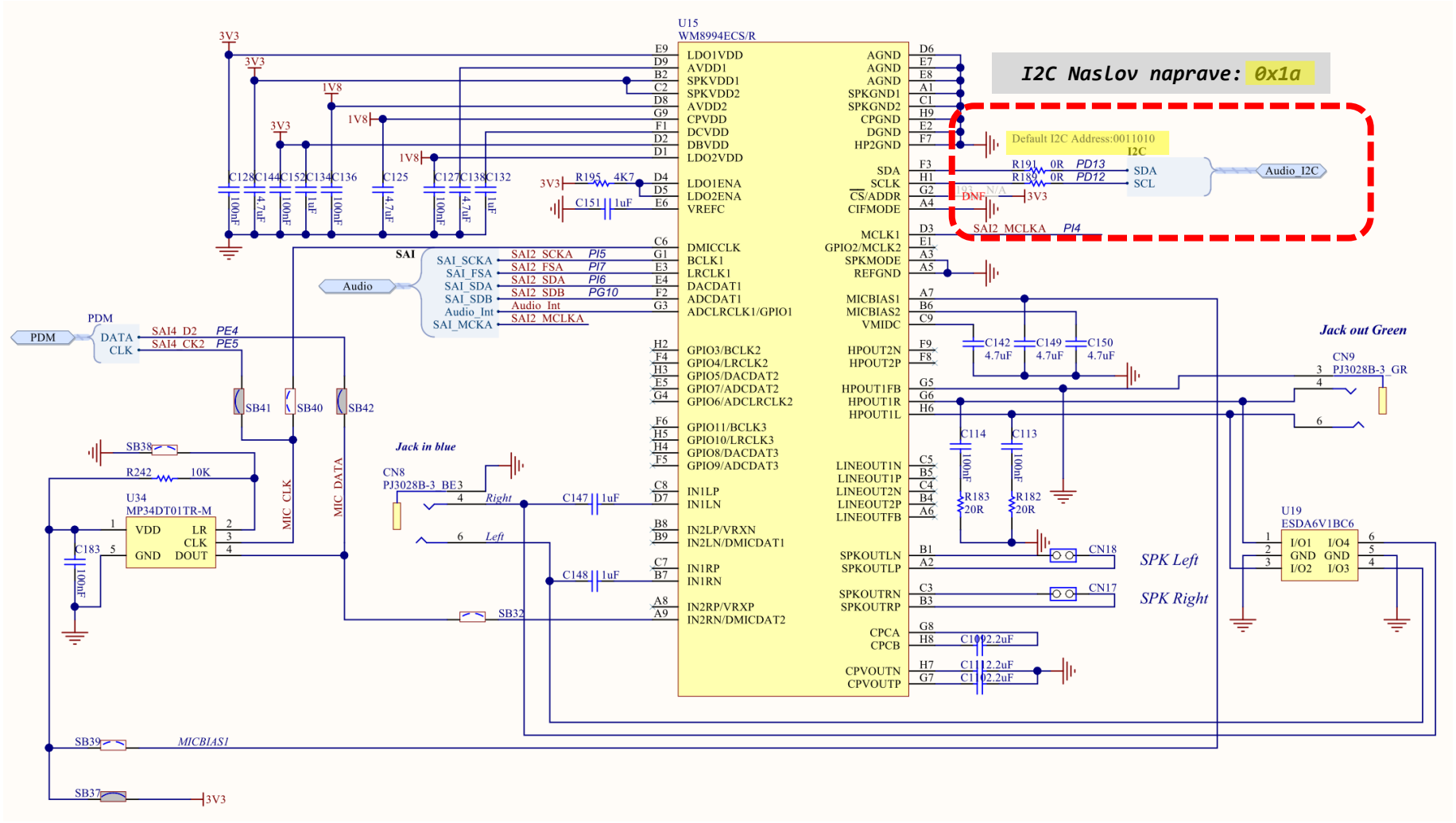
REG	NAME	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	DEFAULT
R0 (0h)	Software Reset	SW_RESET [15:0]																0000h
R1 (1h)	Power Management (1)	0	0	SPKOUTR_ENA	SPKOUTL_ENA	HPOUT2_ENA	0	HPOUT1L_ENA	HPOUT1R_ENA	0	0	MICB2_ENA	MICB1_ENA	0	VMID_SEL [1:0]		BIAS_ENA	0000h
R2 (2h)	Power Management (2)	0	TSHUT_ENA	TSHUT_OPDIS	0	OPCLK_ENA	0	MIXINL_ENA	MIXINR_ENA	IN2L_ENA	IN1L_ENA	IN2R_ENA	IN1R_ENA	0	0	0	0	6000h

STM32H7

VIN - LV

VP 4 - STM32H7 CubeIDE, I2C Audio WM9884 – vezalna shema

STM32H7



https://github.com/LAPSyLAB/STM32H7_Discovery_VIN_Projects/tree/main/STM32H750B-DK_I2C_TS_Demo

Spremenljivke

```
/* USER CODE BEGIN PV */
#define BUFSIZE 256
charSendBuffer[BUFSIZE];
intCounter;
int KeyState=0;
uint8_t dataBuffer[10];

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

main.c : dodana koda

```
// Reading from address 0x1a register R0 (addr. 0x00) default value should be 0x8994
   dataBuffer[0] = 0; dataBuffer[1] = 0x00;
   retval = HAL_I2C_Master_Transmit(&hi2c4, (0x1a << 1), dataBuffer, 2, HAL_MAX_DELAY);
   retval = HAL_I2C_Master_Receive(&hi2c4, (0x1a << 1), dataBuffer, 2, HAL_MAX_DELAY);

   snprintf(SendBuffer,BUFSIZE,"Hello World [%d]: Key:%d
Reg.value1:0x%\n\r",Counter++,KeyState, dataBuffer[0]*256+dataBuffer[1]);

   HAL_UART_Transmit(&huart3,SendBuffer,strlen(SendBuffer),100);
```

Glavna zanka

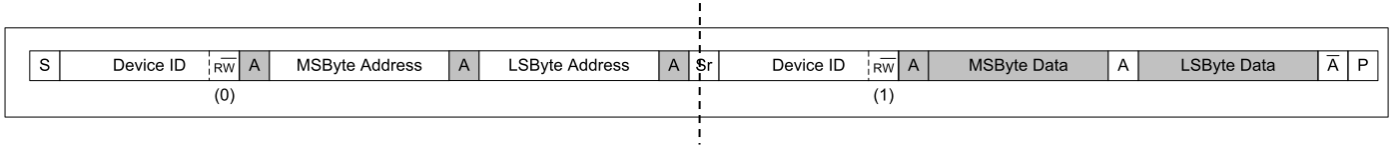


Figure 75 Single Register Read from Specified Address

SOFTWARE RESET AND DEVICE ID

The device ID can be read back from register R0. Writing to this register will reset the device.

The software reset causes most control registers to be reset to their default state. Note that the Control Write Sequencer registers R12288 (3000h) through to R12799 (31FFh) are not affected by a software reset; the Control Sequences defined in these registers are retained unchanged.

The status of the WM8994 digital I/O pins following a software reset is described in Table 141.

The device revision can be read back from register R256.

REGISTER ADDRESS	BIT	LABEL	DEFAULT	DESCRIPTION
R0 (0000h) Software Reset	15:0	SW_RESET [15:0]	8994h	Writing to this register resets all registers to their default state. (Note - Control Write Sequencer registers are not affected by Software Reset.) Reading from this register will indicate device family ID 8994h.

STM32H7

https://github.com/LAPSyLAB/STM32H7_Discovery_VIN_Projects/tree/main/STM32H750B-DK_I2C_TS_Demo

VP 4 - STM32H7 CubeIDE, I2C LCD-Touch RK043FN48H

8-bitni naslovi in 8-bitni registri !

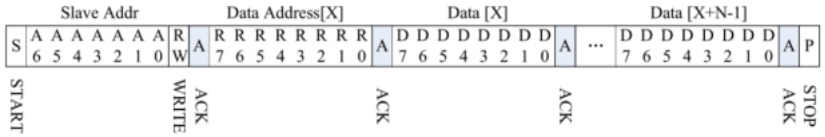
Module P/N: RK043FN48H-CT672B

Version: 1.0

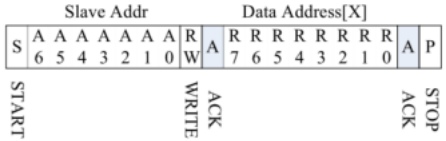
Description : 4.3 inch TFT 480*272 Pixels with LED Backlight and capacitive touch Panel

1.2 I²C Read/Write Interface description

Write N bytes to I2C slave



Set Data Address

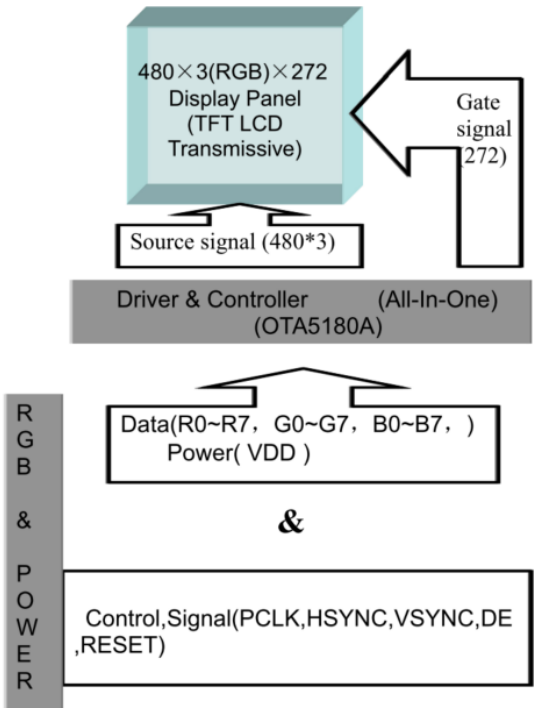


2.1.26 ID_G_ FT5201ID

This register describes vendor's chip id

Address	Bit Address	Register Name	Description
A8h	7:0	ID_G_ FT5201ID	R: xx

Read X bytes from I²C Slave



2.1 Work Mode

In this mode the CTP is fully functional as a touch screen controller. Read and write access address is ju
logical address which is not enforced by hardware or firmware. Here is the operating mode register map.

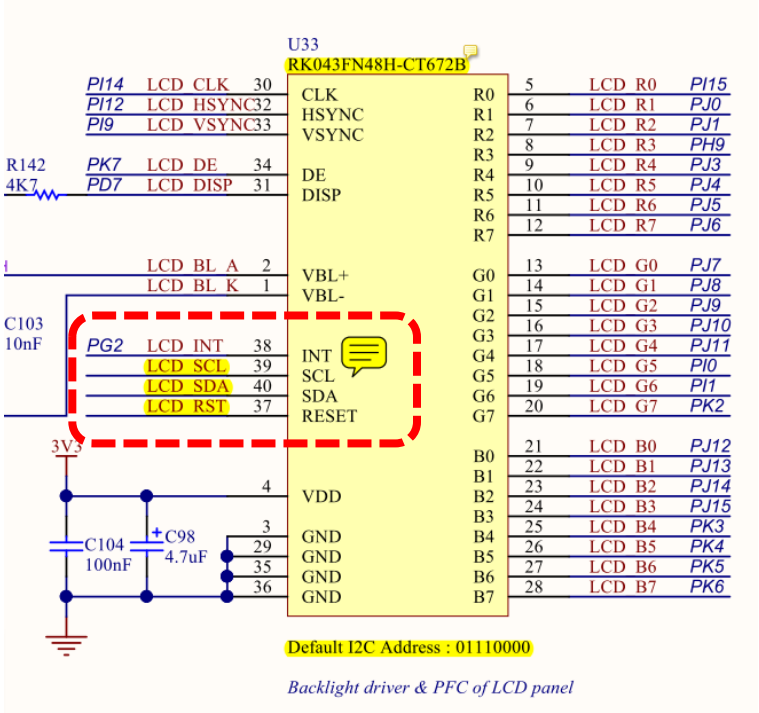
Work Mode Register Map

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Host Access
00h	DEVIDE_MODE		Device Mode[2:0]							RW

Device Mode	Val	Description
Work	000b	Read touch point and gesture
Factory	100b	Read raw data

STM32H7

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



I2C Naslov naprave: 0x38 (ali 0x70 s pomikom na levo – sprostimo prostor za R/W bit)

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

main.c : dodana koda

Spremenljivke

```
/* USER CODE BEGIN PV */
#define BUFSIZE 256
charSendBuffer[BUFSIZE];
intCounter;
int KeyState=0;
uint8_t dataBuffer[10];

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

```
// Reading from address 0x38 register Vendor's Chip ID (addr. 0xA8) default value should be 0x51=81  - Both variations work !
    //dataBuffer[5] = 0xA8;
    //retval = HAL_I2C_Master_Transmit(&hi2c4, (0x38 << 1), &dataBuffer[5], 1, HAL_MAX_DELAY);
    //retval = HAL_I2C_Master_Receive(&hi2c4, (0x38 << 1), &dataBuffer[5], 1, HAL_MAX_DELAY);
    retval = HAL_I2C_Mem_Read(&hi2c4, (0x38 << 1), 0xA8, I2C_MEMADD_SIZE_8BIT,&dataBuffer[5], 1, HAL_MAX_DELAY);


    snprintf(SendBuffer,BUFSIZE,"Hello World [%d]: Key:%d Audio Chip ID: 0x%4x Touch ID: 0x%2x=%d\n\r",Counter++,KeyState,
dataBuffer[0]*256+dataBuffer[1],dataBuffer[5],dataBuffer[5]);
    HAL_UART_Transmit(&huart3,SendBuffer,strlen(SendBuffer),100);

    HAL_Delay(1000);
```

Glavna zanka

Polling mode IO operation

- Transmit in master mode an amount of data in blocking mode using HAL_I2C_Master_Transmit()
- Receive in master mode an amount of data in blocking mode using HAL_I2C_Master_Receive()
- Transmit in slave mode an amount of data in blocking mode using HAL_I2C_Slave_Transmit()
- Receive in slave mode an amount of data in blocking mode using HAL_I2C_Slave_Receive()

Polling mode IO MEM operation

- Write an amount of data in blocking mode to a specific memory address using HAL_I2C_Mem_Write()
- Read an amount of data in blocking mode from a specific memory address using HAL_I2C_Mem_Read()

STM32H7

2.1.26 ID_G_ FT5201ID

This register describes vendor's chip id

Address	Bit Address	Register Name	Description
A8h	7:0	ID_G_ FT5201ID	R: xx

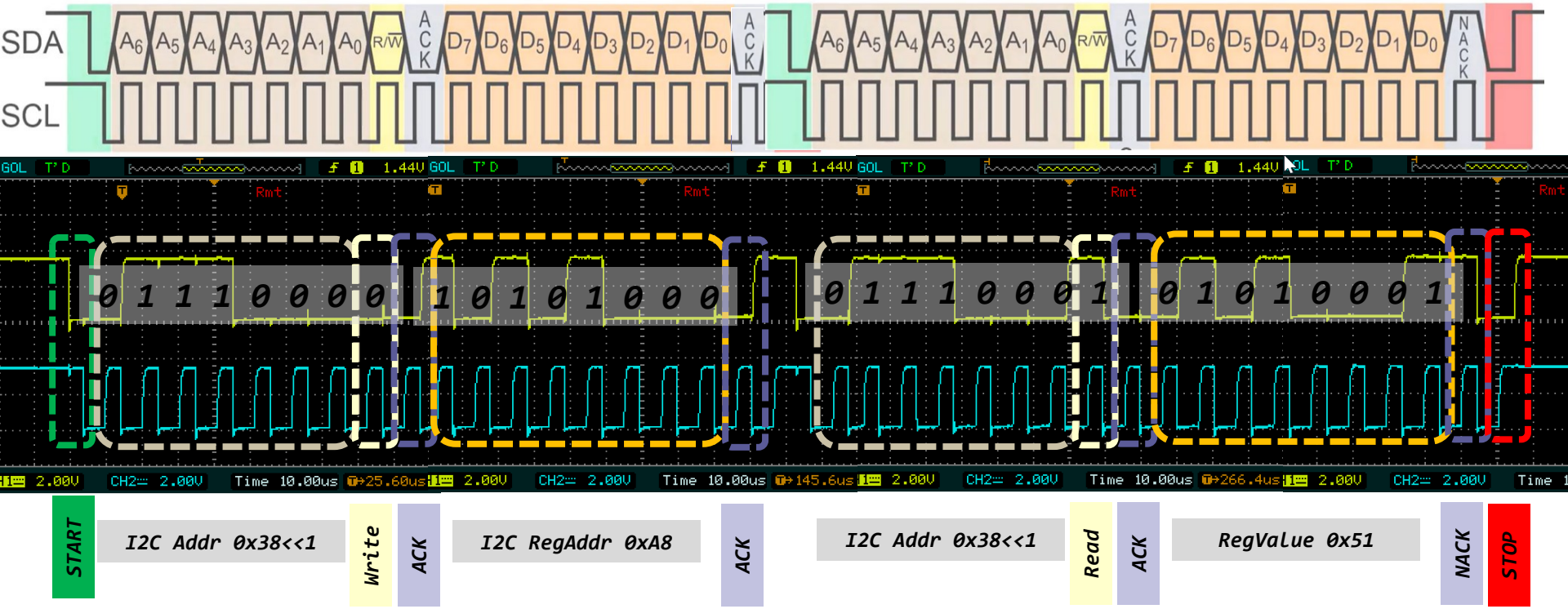
https://github.com/LAPSyLAB/STM32H7_Discovery_VIN_Projects/tree/main/STM32H750B-DK_I2C_Basic_Demo

STM32H7

I2C branje

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

Primer I2C komunikacije STM32H7 - Touch

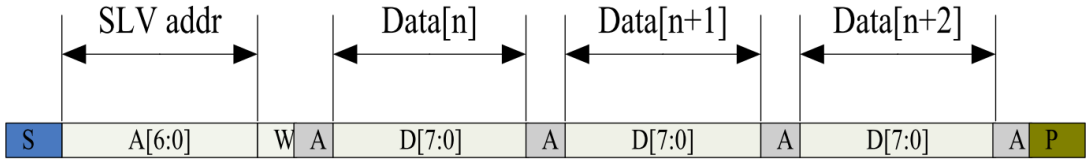


Figure 2-5 I2C master write, slave read

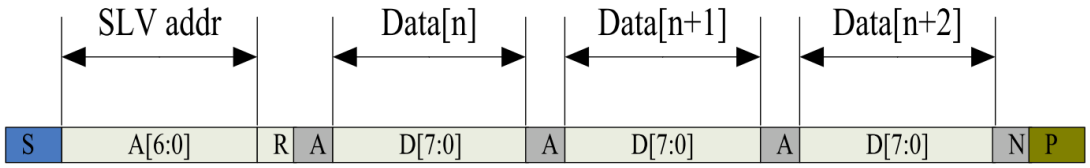
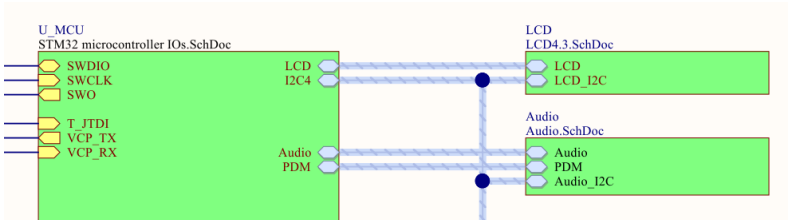


Figure 2-6 I2C master read, slave write



STM32H7

FocalTech **FT5336GQQ**

True Multi-Touch Capacitive Touch Panel Controller

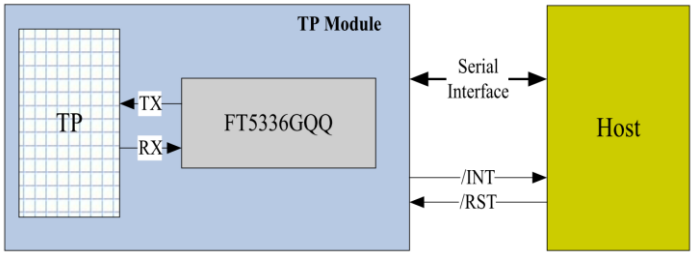


Figure 2-3 Host Interface Diagram

https://github.com/LAPSyLAB/STM32H7_Discovery_VIN_Projects/tree/main/STM32H750B-DK_I2C_Touch_Demo

8-bitni naslovi in registri

Work Mode Register Map

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Host Access
00h	DEVIDE_MODE		Device Mode[2:0]							RW
01h	GEST_ID	Gesture ID[7:0]								R
02h	TD_STATUS					Number of touch points[3:0]				R
03h	TOUCH1_XH	1st Event Flag				1st Touch X Position[11:8]				R
04h	TOUCH1_XL	1st Touch X Position[7:0]								R
05h	TOUCH1_YH	1st Touch ID[3:0]				1st Touch Y Position[11:8]				R
06h	TOUCH1_YL	1st Touch Y Position[7:0]								R
A8h	ID_G_FT5201ID	CTPM Vendor ID								R

Primer I2C komunikacije STM32H7 - Touch

STM32H7

```
// 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,&VendorID, 1, HAL_MAX_DELAY);

retval = HAL_I2C_Mem_Read(&hi2c4, (0x38 << 1), 0x00, I2C_MEMADD_SIZE_8BIT,&DeviceMode, 1, HAL_MAX_DELAY);
retval = HAL_I2C_Mem_Read(&hi2c4, (0x38 << 1), 0x01, I2C_MEMADD_SIZE_8BIT,&Gesture, 1, HAL_MAX_DELAY);
retval = HAL_I2C_Mem_Read(&hi2c4, (0x38 << 1), 0x02, I2C_MEMADD_SIZE_8BIT,&Status, 1, HAL_MAX_DELAY);

retval = HAL_I2C_Mem_Read(&hi2c4, (0x38 << 1), 0x03, I2C_MEMADD_SIZE_8BIT,&dataBuffer, 4, HAL_MAX_DELAY);
if (Status != 0) {
      TouchX = ( (dataBuffer[0] & 0b1111) << 8) + dataBuffer[1];
      TouchY = ( (dataBuffer[2] & 0b1111) << 8) + dataBuffer[3];
} else {
      TouchX = 0;
      TouchY = 0;
}
```

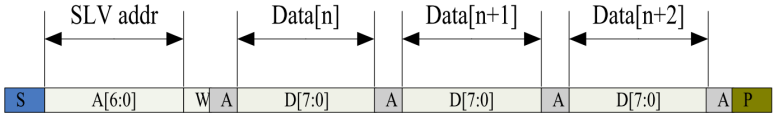


Figure 2-5 I2C master write, slave read

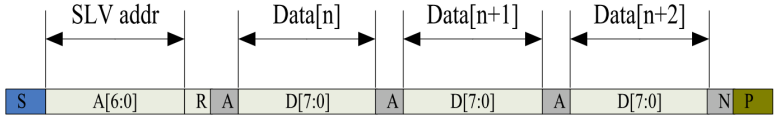


Figure 2-6 I2C master read, slave write

8-bitni naslovi in registri

Work Mode Register Map

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Host Access
00h	DEVIDE_MODE		Device Mode[2:0]							RW
01h	GEST_ID	Gesture ID[7:0]								R
02h	TD_STATUS					Number of touch points[3:0]				R
03h	TOUCH1_XH	1st Event Flag				1st Touch X Position[11:8]				R
04h	TOUCH1_XL	1st Touch X Position[7:0]								R
05h	TOUCH1_YH	1st Touch ID[3:0]				1st Touch Y Position[11:8]				R
06h	TOUCH1_YL	1st Touch Y Position[7:0]								R
A8h	ID_G_FT5201ID	CTPM Vendor ID								R

https://github.com/LAPSyLAB/STM32H7_Discovery_VIN_Projects/tree/main/STM32H750B-DK_I2C_Touch_Demo

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

- Diskusija, vprašanja ?