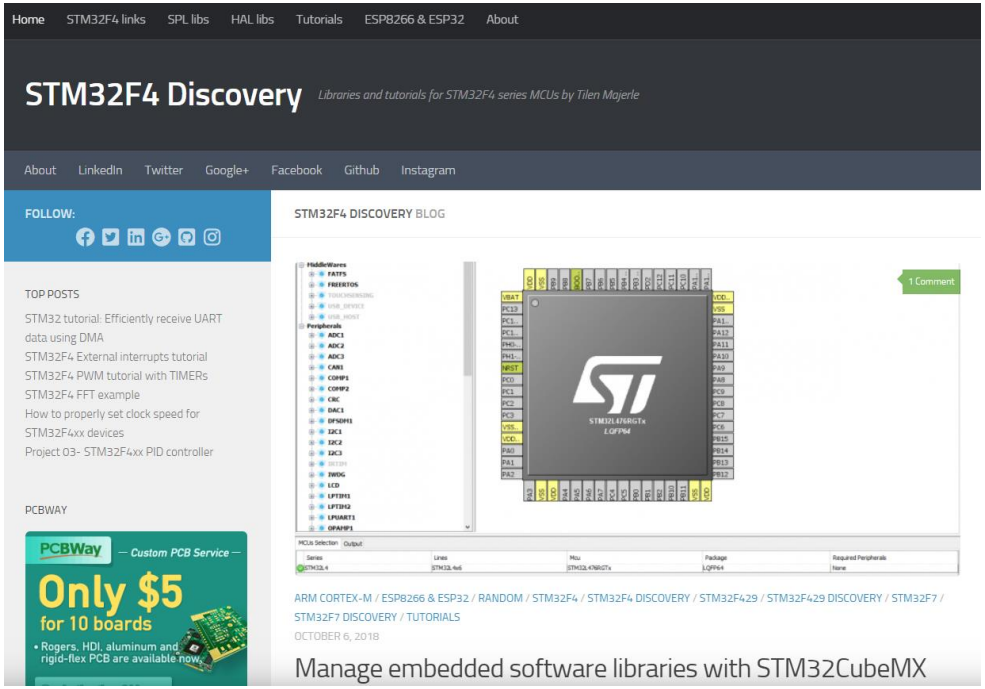
Vhodno izhodne naprave

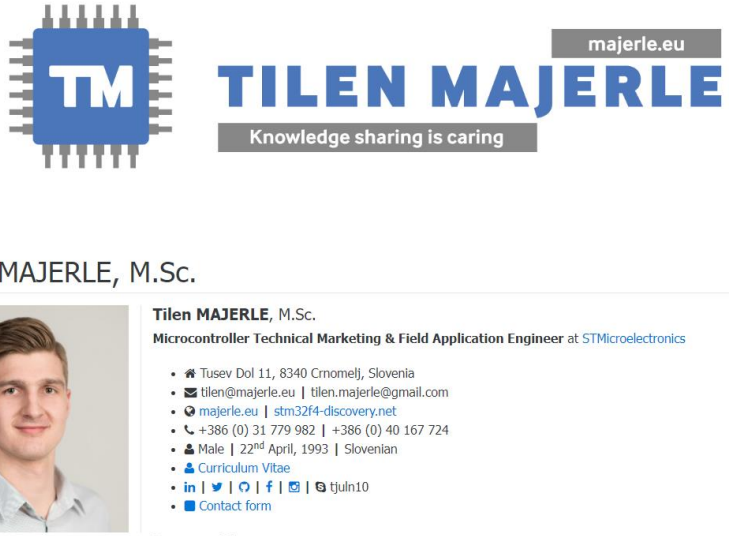
Laboratorijska vaja 4 - VP 4
VIN projekt, „Edge AI“, STM32 F4/H7
PWM-I2C-SPI primeri, Miško3 Demo

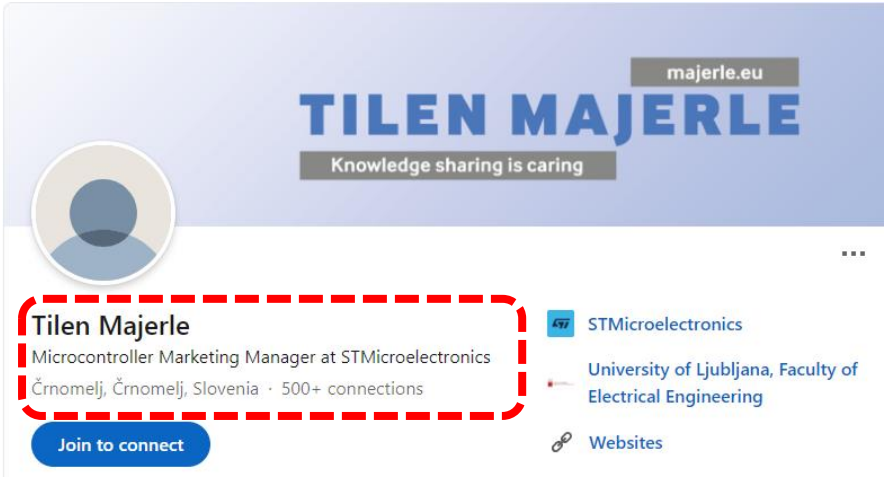
VIN projekt - VP4: STM32-Edge computing, CubeIDE primeri, Miško3

- VIN projekt
- AI v vgrajenih napravah („Edge Computing“)
- STM32 CubeIDE F4,H7 – PWM izhodi
- STM32F4 CubeIDE: SPI in LIS3DSH, I2C in CS43L22
- STM32H7 CubeIDE, I2C
- Miško3 – demo projekt

Delo na STM32F4 razvojnem sistemu - zgodba







https://stm32f4-discovery.net/

VIN projekt

Spisek opreme

VIN projekt

Ideje

VIN-VSP 2022-23 zvezek
Knjižnica vsebine

Tabla 2022-23 | Predavanja 2021-22 | VIN Projekt - Ideje | VIN Projekti - Done | Moduli, Tipala | LAB Oprema | ...

Išči (Ctrl + E)

Dodaj stran

Preberi me
Spletni viri
Teme, področja
Brezstično zaznavanje - CapSense
Logični/protokolski analizator
Sigrok - SW
I2c sniffer
Red Pitaya
Praktični izzivi v LAPSYLAB
LSM6DSOX (30 kosov na voljo)
DIY hodeči robot
Breadboard samogradnja
Gibanje
Joystick
DIY osciloskop STM32
Arduino
Upornost/prevodnost kože
WS2812 Neopixel
Praktični Izzivi
STM32H750B DK + clickboards
STMod+ konektor
STM32F4 Shield + Click boards
Clickboards
LTE IOT 2 CLICK (BG96)
Model Hiške
Arduino Smart Home Kit
CANBUS - IEX modul

Brezstično zaznavanje - CapSense

nedelja, 24. april 2022 11:28

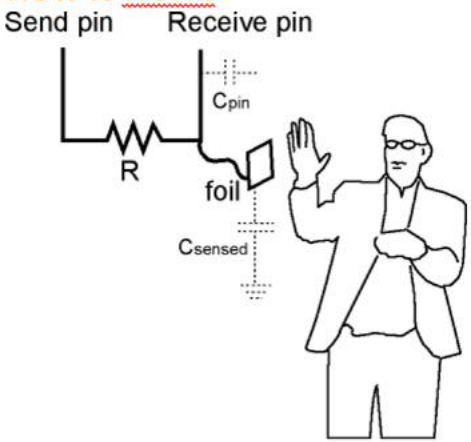
Capacitive Sensing Library

by Paul Badger

...

RR

How it works



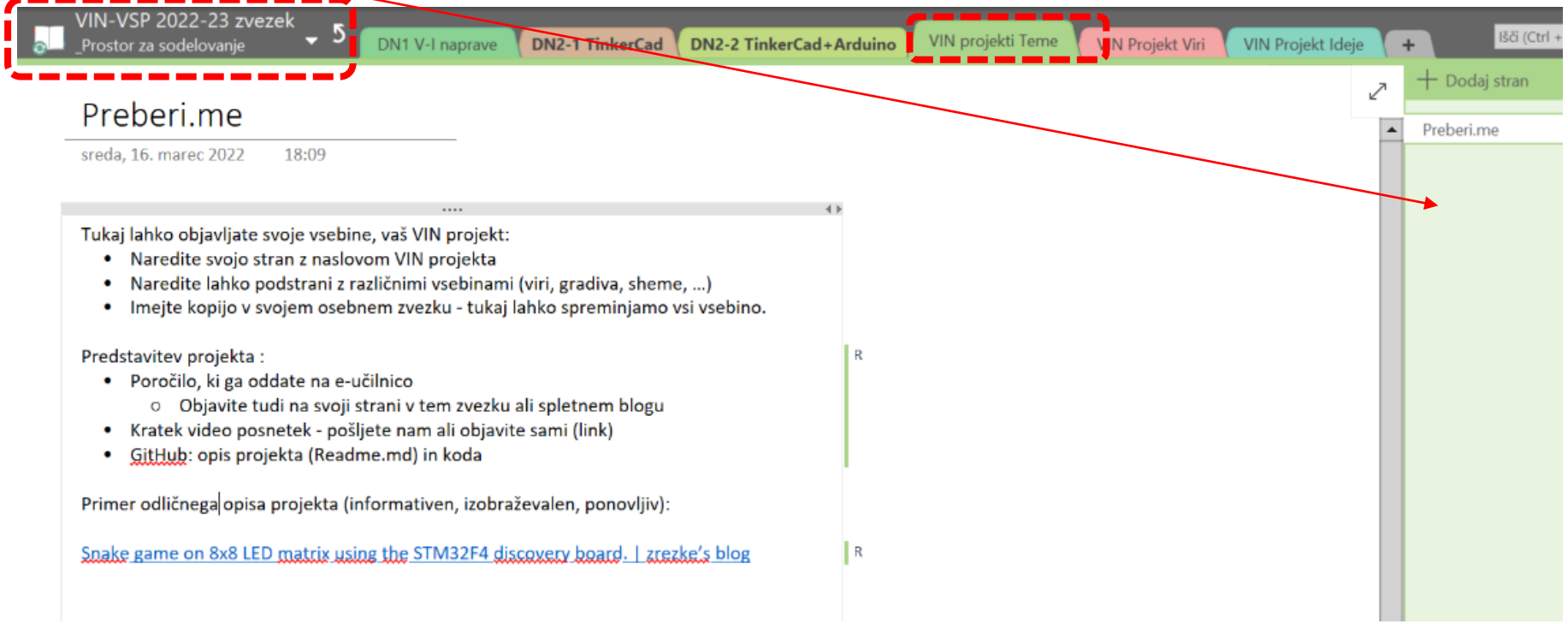
The capacitiveSensor method toggles a microcontroller *send* pin to a new state and then waits for the *receive* pin to change to the same state as the send pin. A variable is incremented inside a *while* loop to time the receive pin's state change. The method then reports the variable's value, which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

Watch a short video demonstration (YouTube)

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

VIN projekt

Vaša tema ?



Osnovni projekt CubeIDE – GPIO – nivoji programiranja

Baremetal - zbirnik

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

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


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

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

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

Baremetal - C

```
  /* USER CODE BEGIN 2 */


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

  /* USER CODE END 2 */

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

/* USER CODE END WHILE */

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

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

HAL - C

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

HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_12);

  /* USER CODE END WHILE */

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

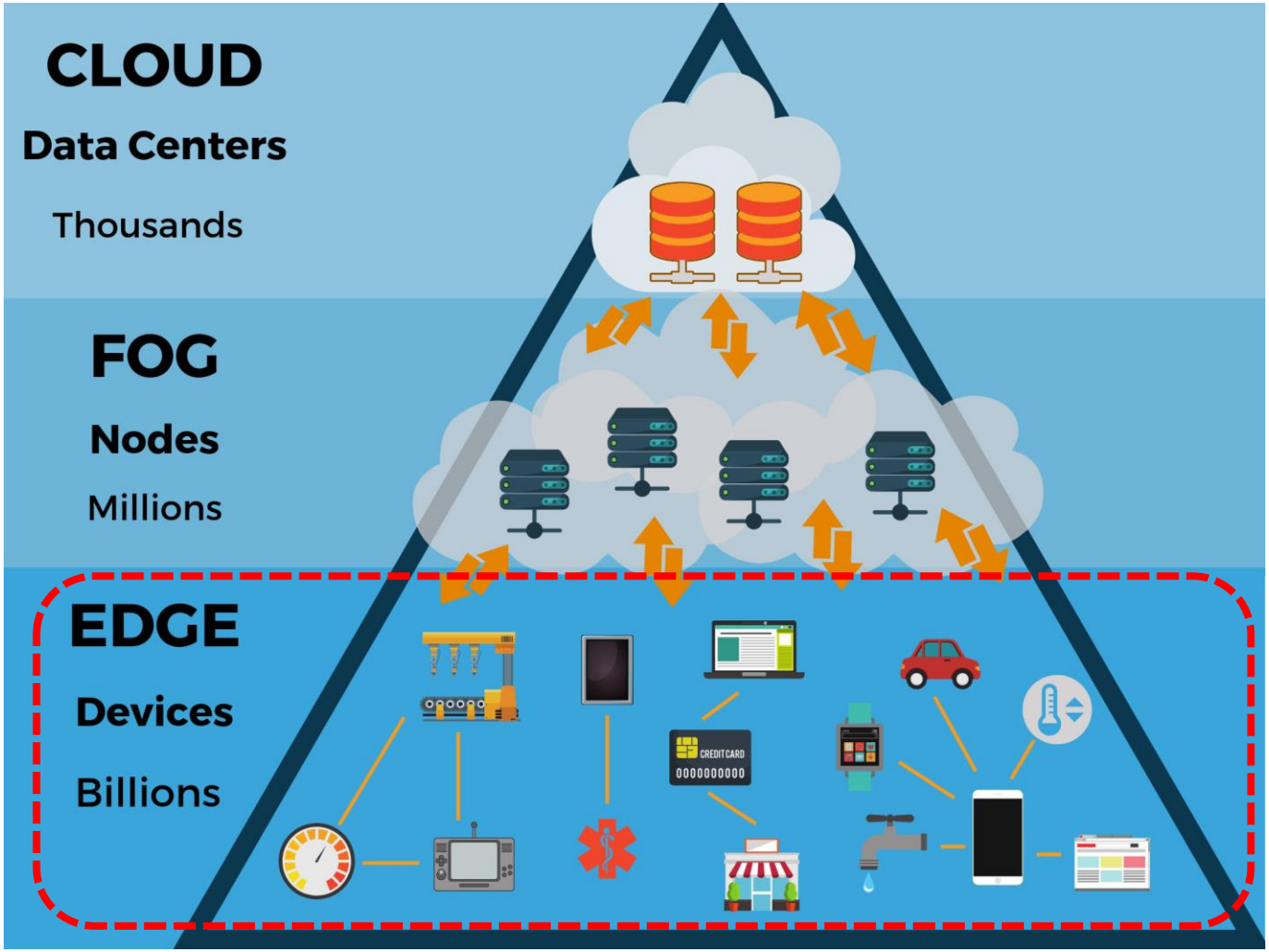
```
void HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx,
uint16_t GPIO_Pin)
{
  uint32_t odr;

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

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

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

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

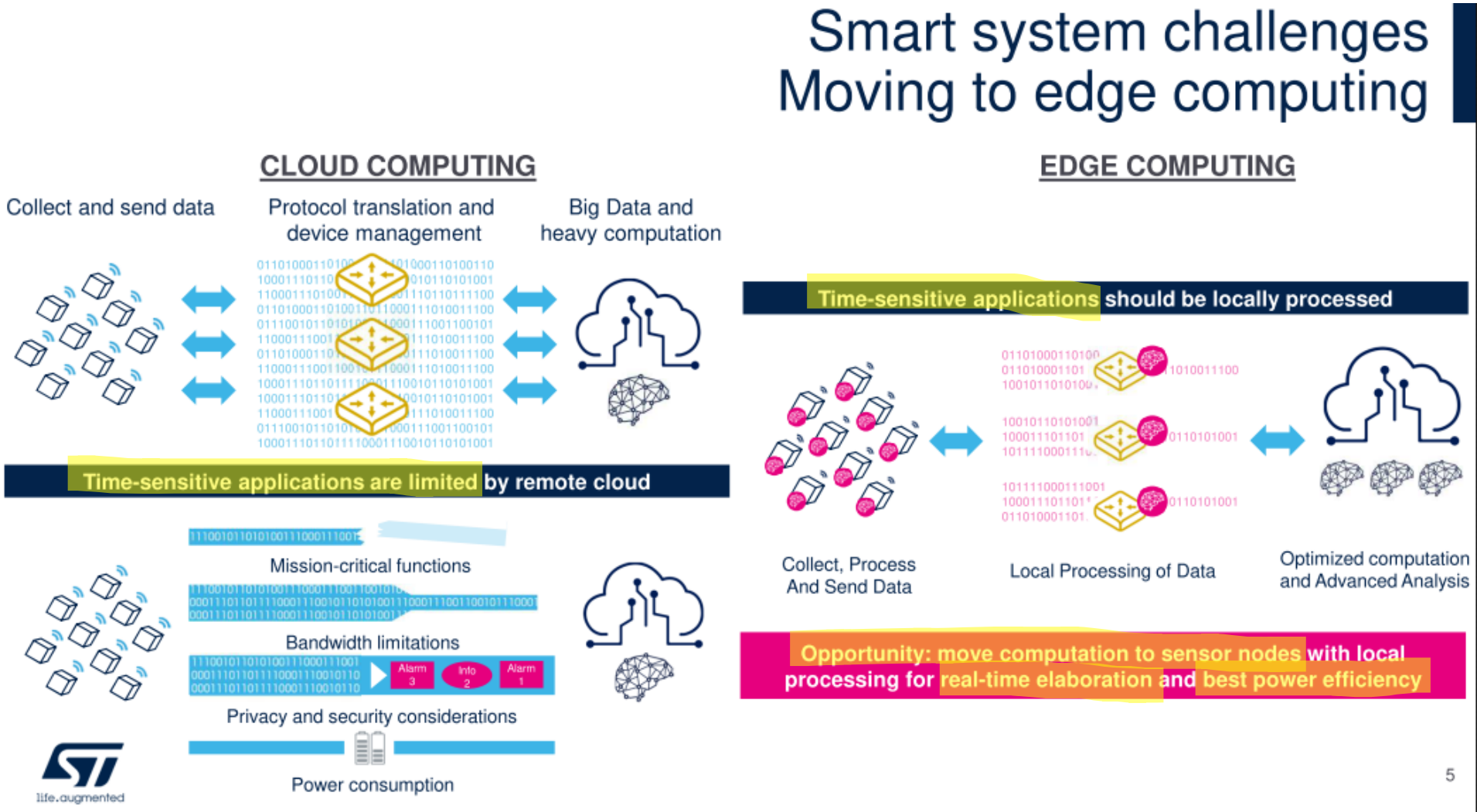
VIN projekt - VP4: STM32-Edge computing, CubeIDE primeri, Miško3

- VIN projekt
- AI v vgrajenih napravah („Edge Computing“)
- STM32 CubeIDE F4,H7 – PWM izhodi
- STM32F4 CubeIDE: SPI in LIS3DSH, I2C in CS43L22
- STM32H7 CubeIDE, I2C
- Miško3 – demo projekt

Edge computing



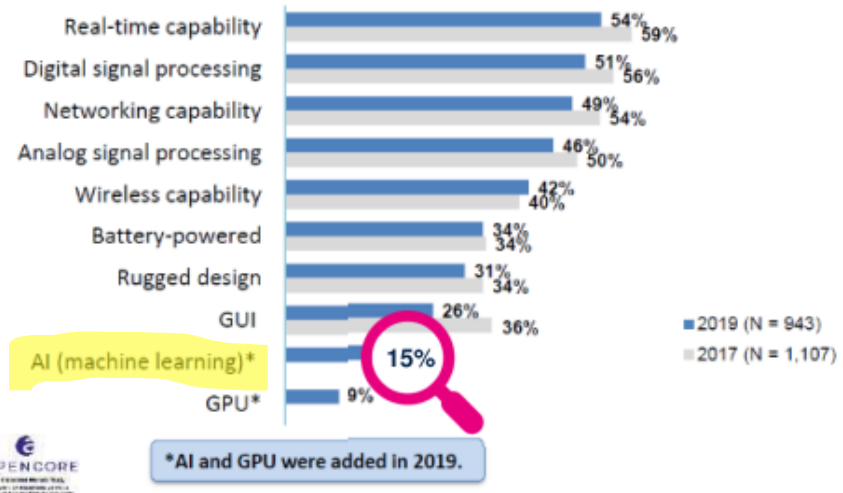
Edge computing



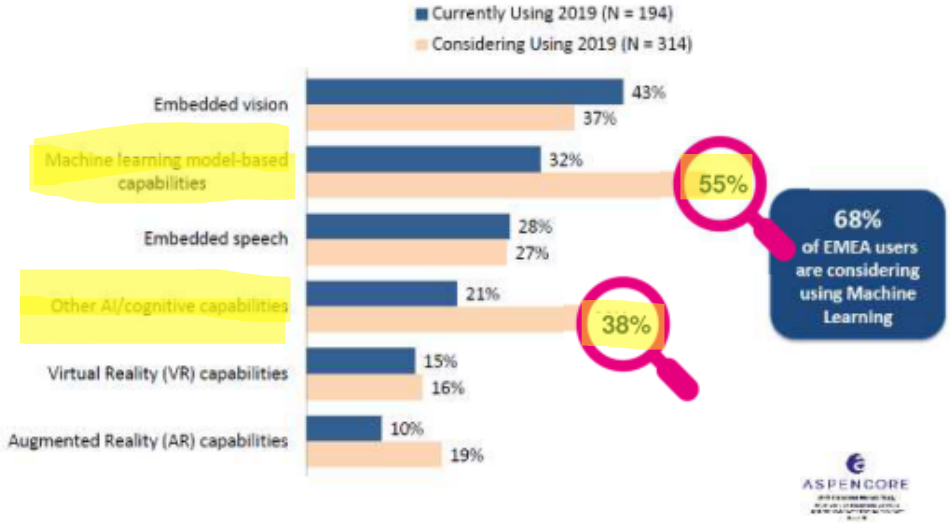
Edge computing

AI is moving to the edge

Capabilities included in a project



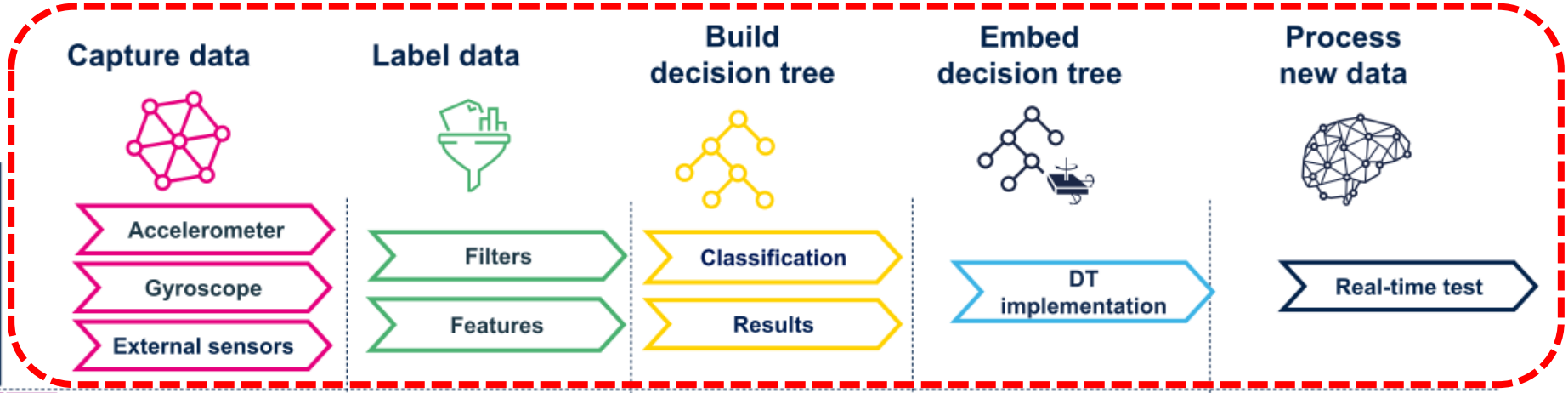
Advanced technology in a project



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

Edge computing – moduli, tipala

LSM6DSOX – SensorTile.box



		Capture data	Label data	Build decision tree	Embed decision tree	Process new data
WHAT		Accelerometer, Gyroscope, External sensors	Filters, Features	Classification, Results	DT implementation	Real-time test
HOW	HW	**SensorTile.box** • STEVAL-MKSBOX1V1	-------	-------	-------	**SensorTile.box** • STEVAL-MKSBOX1V1
HOW	SW	**No USB cable:** • STBLESensor **With USB cable:** • FP-SNS-STBOX1 • Unicleo GUI	**Unico GUI** * *External tools for building decision tree: Weka, RapidMiner, MATLAB, Python			**No USB cable:** • STBLESensor **With USB cable:** • FP-SNS-STBOX1 • Unicleo GUI **(for advanced level)** • AlgoBuilder

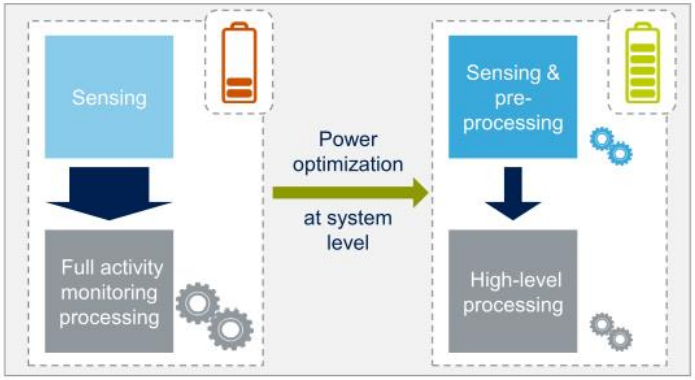
Edge computing – moduli, tipala

BHI260AP

BOSCH

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

LSM6DSOX Unique Performance

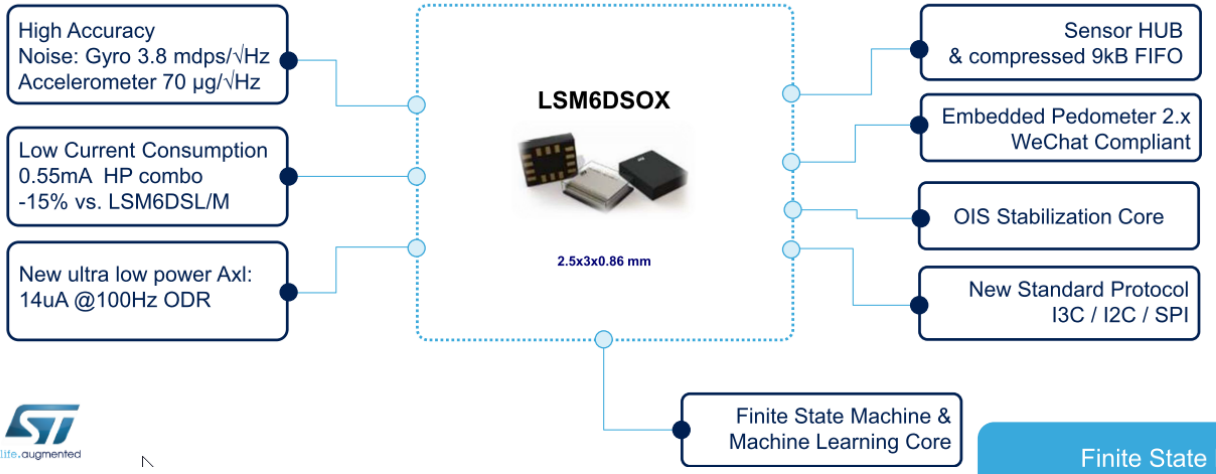


LSM6DSOX adapter board for a standard DIL24 socket

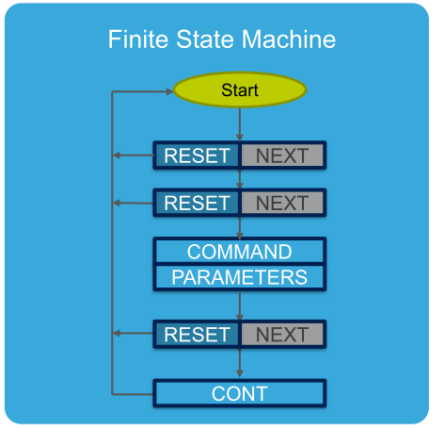
Edge computing – moduli, tipala

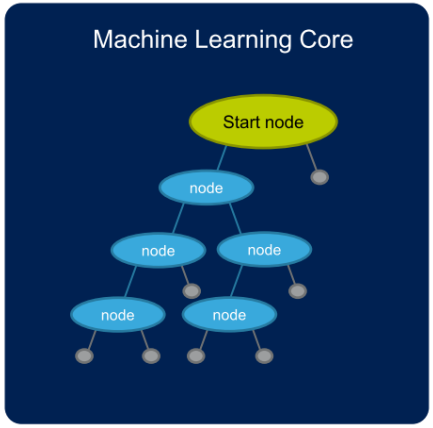
LSM6DSOX Unique Performance

Improved Accuracy, Optimized System Power



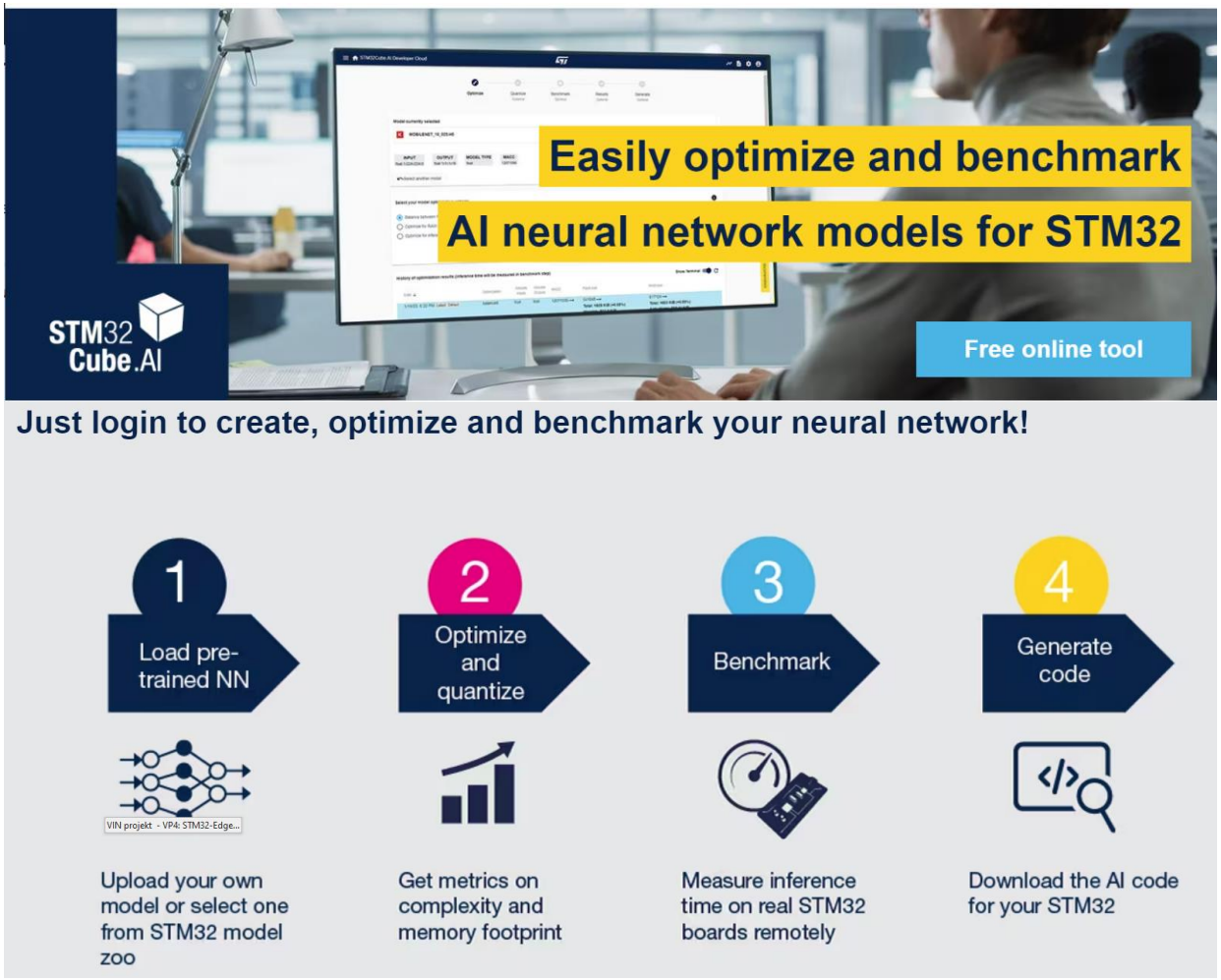
LSM6DSOX adapter board for a standard DIL24 socket





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

Edge computing – Optimizacija AI modelov



https://stm32ai-cs.st.com/home

VIN projekt - VP4: STM32-Edge computing, CubeIDE primeri, Miško3

- VIN projekt
- AI v vgrajenih napravah („Edge Computing“)
- STM32 CubeIDE F4,H7 – PWM izhodi
- STM32F4 CubeIDE: SPI in LIS3DSH, I2C in CS43L22
- STM32H7 CubeIDE, I2C in Touch panel, zaslon
- Miško3 – demo projekt

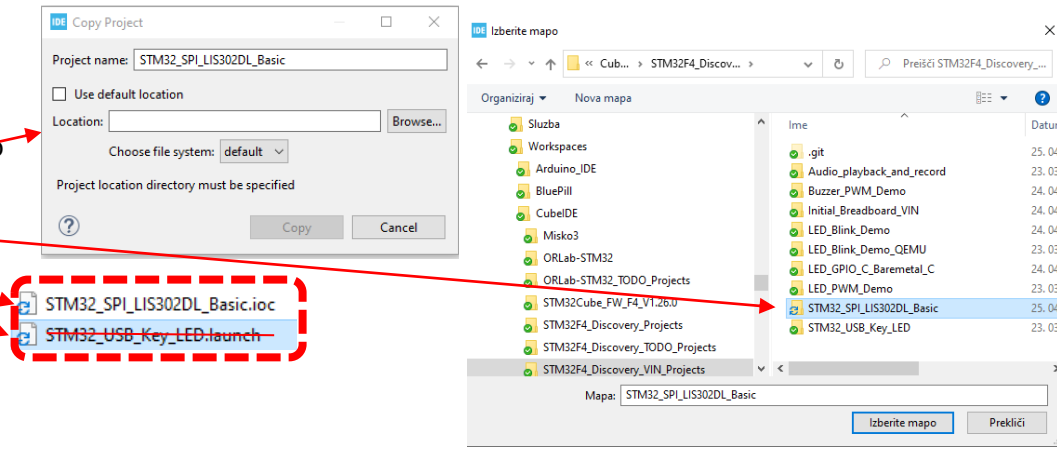
CubeIDE – delo s projekti

Kopiranje/preimenovanje projekta :

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

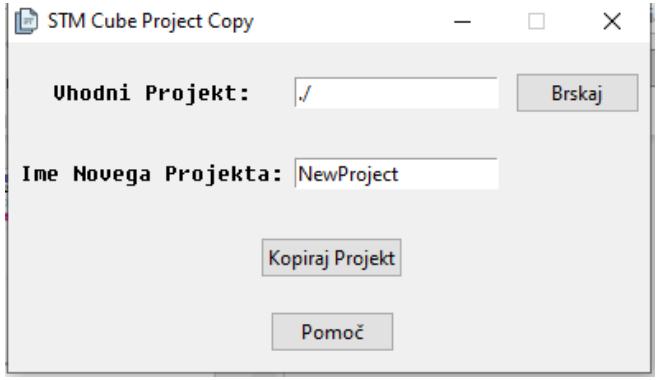
Kopiranje CubeIDE projekta z CubeMX .ioc datoteko

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/CubeIDE

STM32F4 – PWM signali za LED diode (LED dimmer)

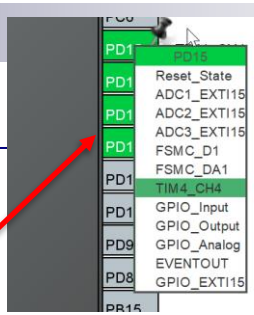
HAL - C

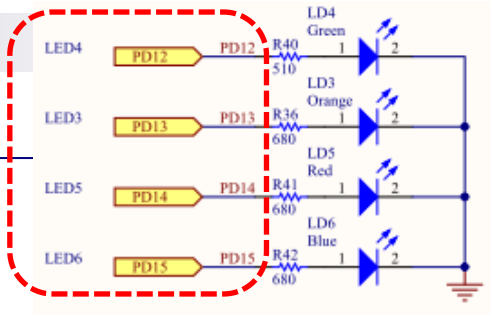
CubeMX :

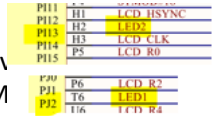
1. New project -> STM32 Project -> Board -> 407DISC1
2. CubeMX: Spremeniti USB Host v USB Device :
   - Connectivity -> USB_OTG_FS -> Mode v Device Only
   - Middleware -> DEVICE_USB in Class Virtual Com Port

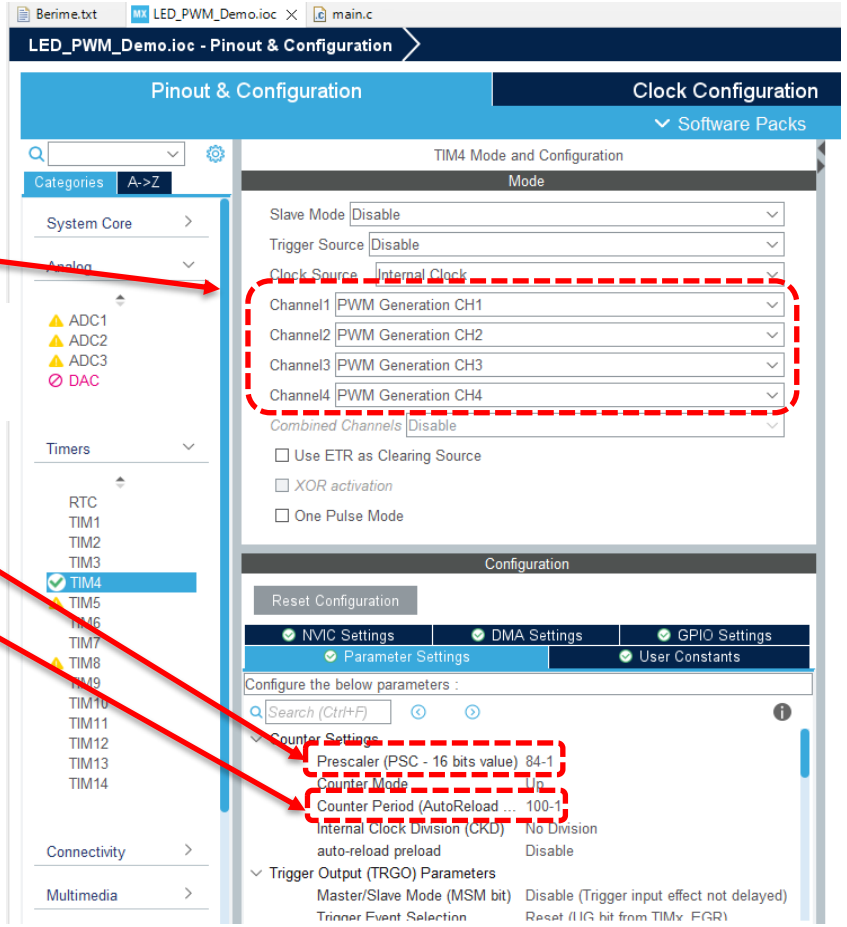
3. Spremeniti pine PD12-PD15 (LED diode) v TIM4_CH1-4
   - tim4 Vse kanale spremeniti na PWM Generation CH1-4

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

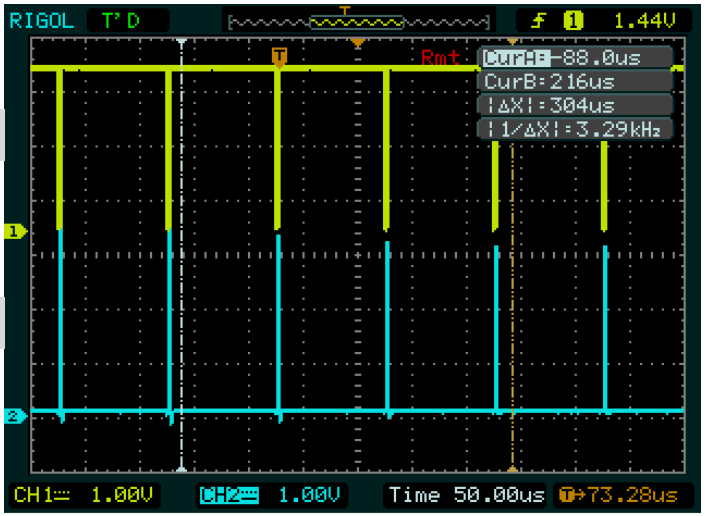
Osnovni projekt CubeIDE – GPIO – PWM, LED diode

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 */
```

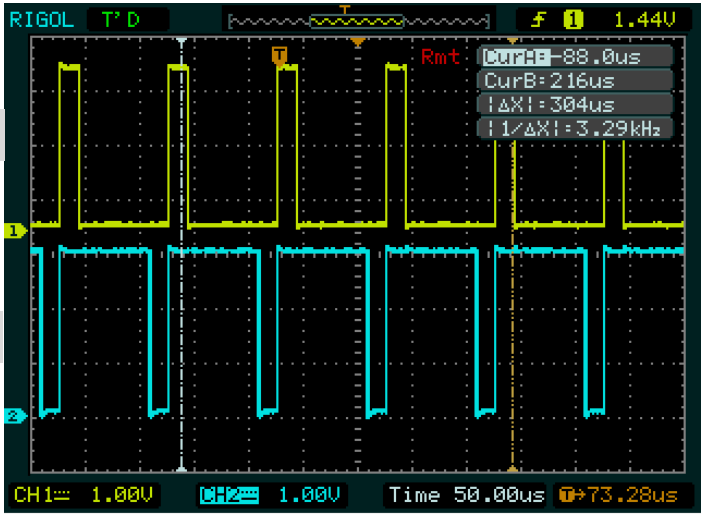
Max Duty

Min Duty



Min Duty

Max Duty



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

STM32F4 – PWM signali/melodija za brenčača (Buzzer)

HAL - C

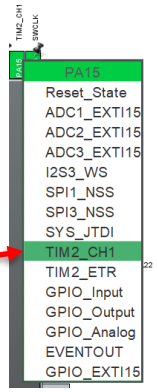
Brencac se prikljuci na **PA15 (TIM2->CH1) in GND**

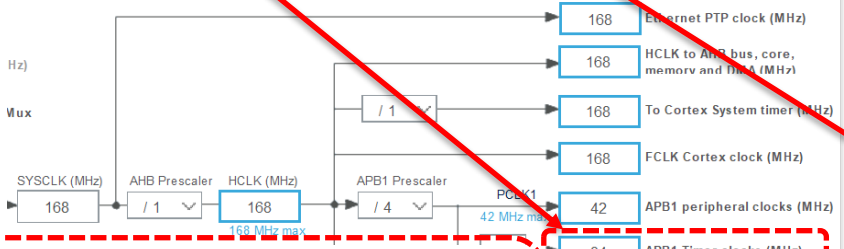
CubeMX :
1. New project -> STM32 Project -> Board -> 407DISC1
2. CubeMX: Spremeniti USB Host v USB Device :
Connectivity -> USB_OTG_FS -> Mode v Device Only
Middleware -> DEVICE_USB in Class Virtual Com Port

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



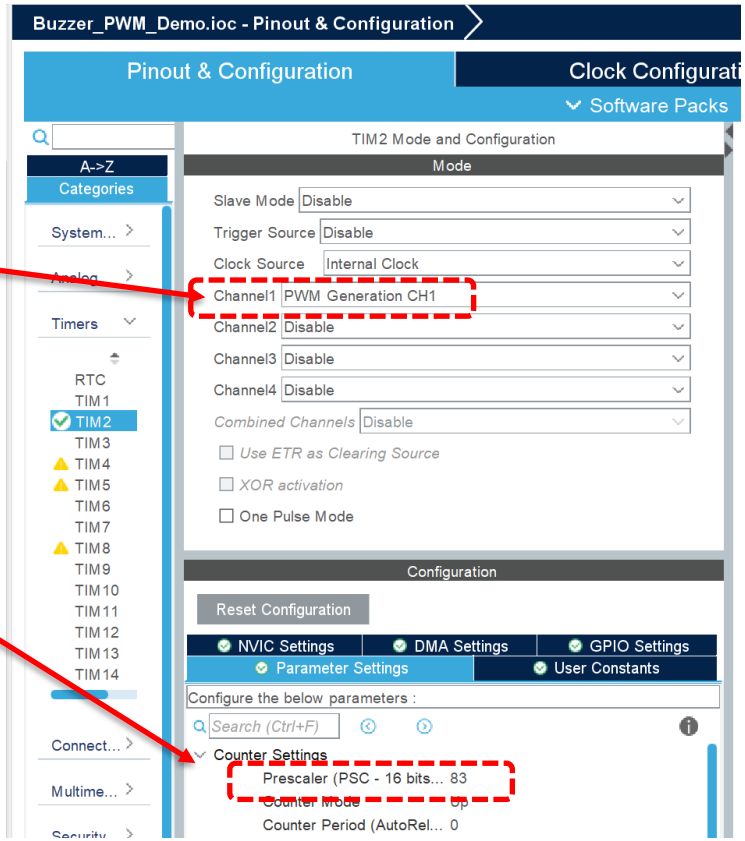


Nota:

```
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);
```

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



STM32F4

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

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

STM32H7 – PWM signali/melodija za brenčača (Buzzer)

HAL - C

Brencac se prikljuci na **PA3-PWM na STMod+ Clickboard (TIM2->CH4) in GND (priključitev lahko naredimo skupaj na naslednji vaji – VP5)**

CubeMX :

1. Osnovna nastavitev plošče

2. Spremeniti pin PA3 v TIM2->CH4
tim2 kanal 4 spremeniti na PWM Generation CH4

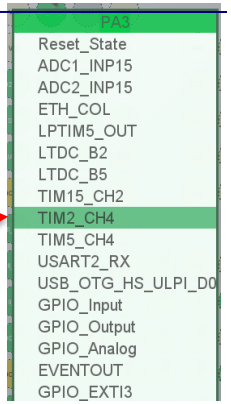
3. Clock & TIM2:
Ura števca = 1 MHz
Prescaler (PSC - 16 bits value) = 64-1 = 63 (clock 1Mhz)

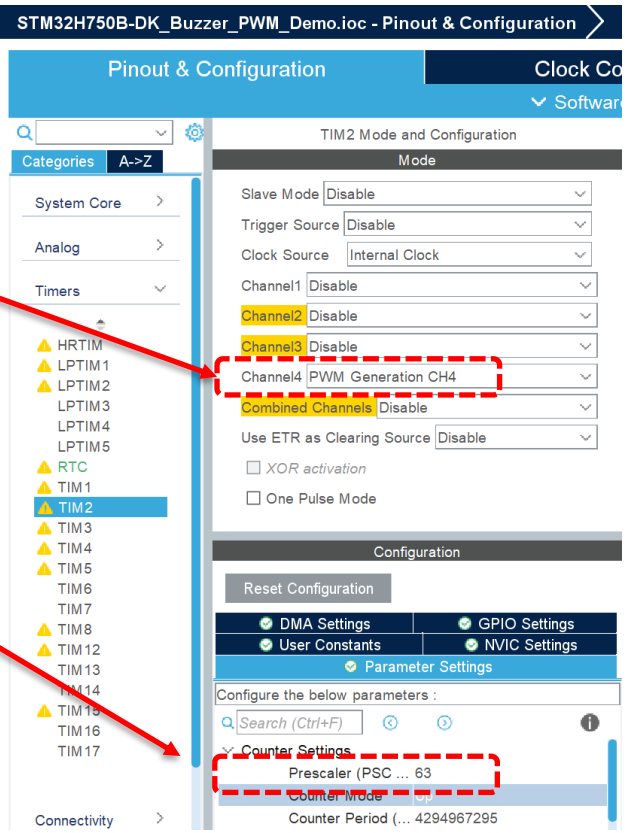
Perioda štetja se bo določala glede na noto (duty cycle je vedno 50%)
Counter Period (AutoReload Register - 16 bits value) =
ARR = 1000000 (ura števca) / Frekv.note[Hz]
CCR1 bo vedno ARR/2 (50% duty cycle)

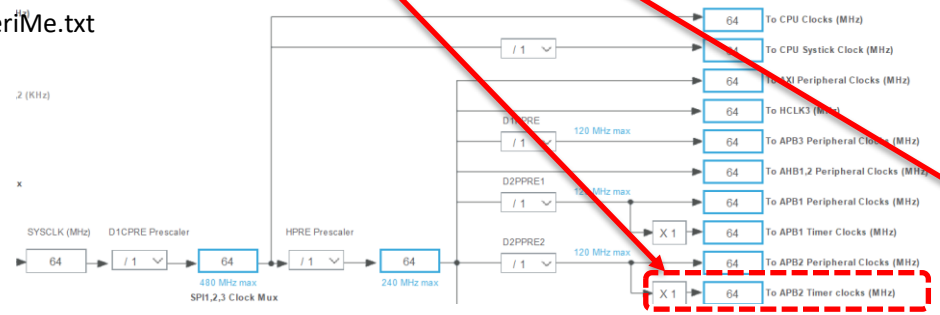
Več informacij BeriMe.txt

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





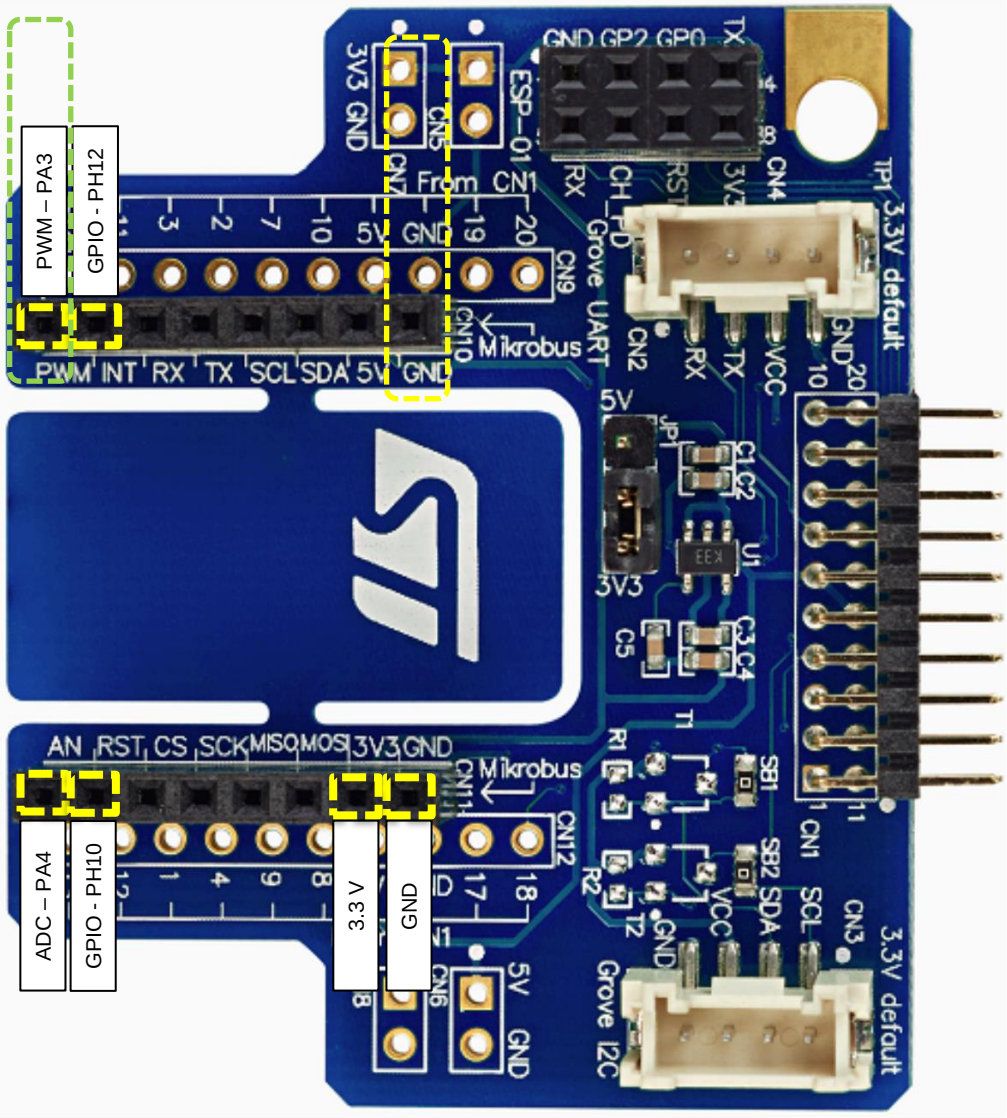


Nota:

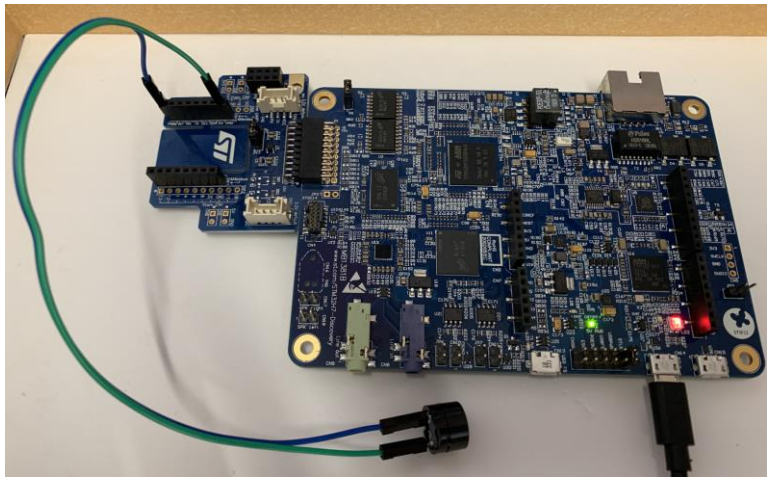
```
ARR_period = (int)(1000000/NoteFreq);  //Already prescaled to 1 MHz
    setPWM(htim2, TIM_CHANNEL_4, ARR_period, ARR_period/2);

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

STM32H7

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



Pravilna priključitev



Nepravilna priključitev

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

STM32H7 – 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_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|
                  CCR1=%d\r\n",melodyIndex,noteIndex,melody[melodyIndex][noteIndex],Delay
                  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

VIN projekt - VP4: STM32-Edge computing, CubeIDE primeri, Miško3

- VIN projekt
- AI v vgrajenih napravah („Edge Computing“)
- STM32 CubeIDE F4,H7 – PWM izhodi
- STM32F4 CubeIDE: SPI in LIS3DSH, I2C in CS43L22
- STM32H7 CubeIDE, I2C
- Miško3 – demo projekt

5 Digital main blocks

5.1 State machine

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

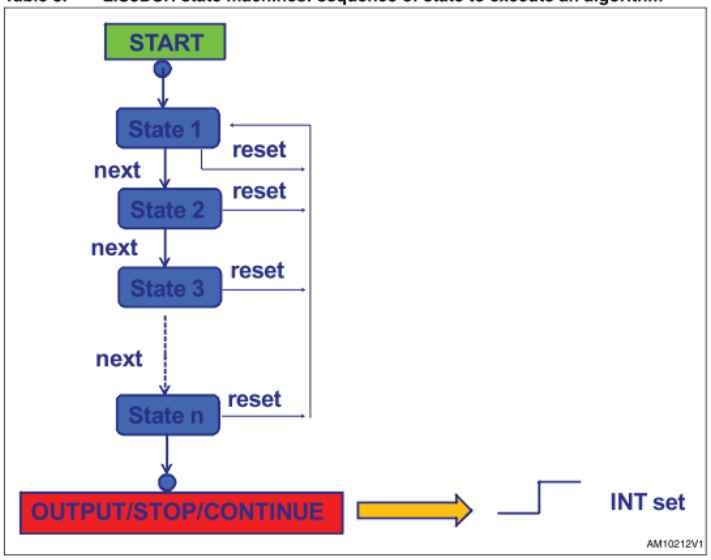
The program is made up of a set of instructions that define the transition to successive states. Conditional branches are possible.

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

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

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

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



SPI - serial peripheral interface

Subject to general operating conditions for Vdd and Top.

SPI slave timing values

	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

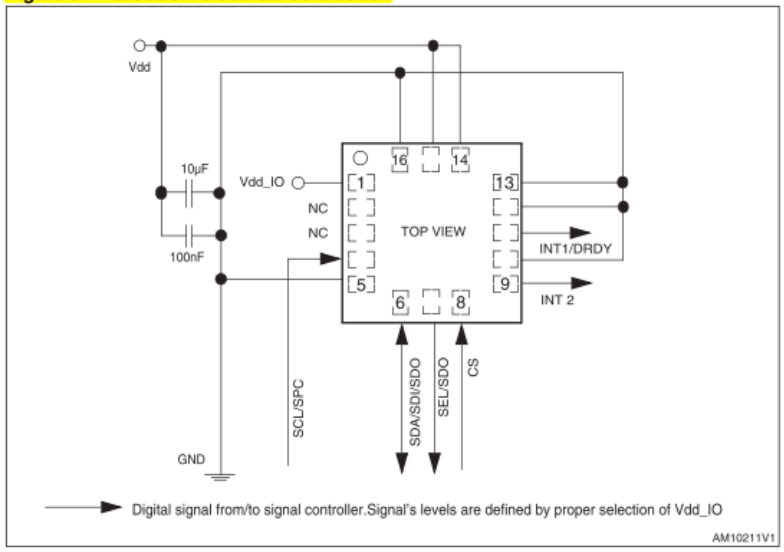
Parameter	I²C standard mode (1) Min.	I²C standard mode (1) Max.	I² fast mode (1) Min	I² fast mode (1) Max.	Unit
SCL clock frequency	0	100	0	400	kHz

Table 7. Absolute maximum ratings

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

Application hints

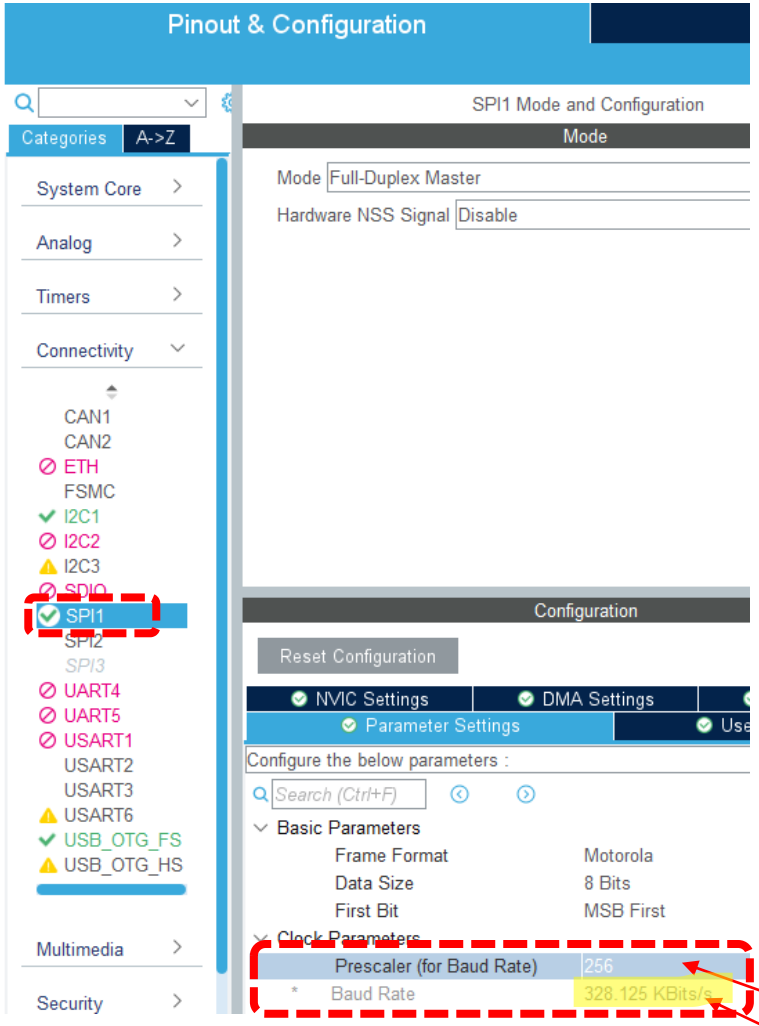
Figure 5. LIS3DSH electrical connection

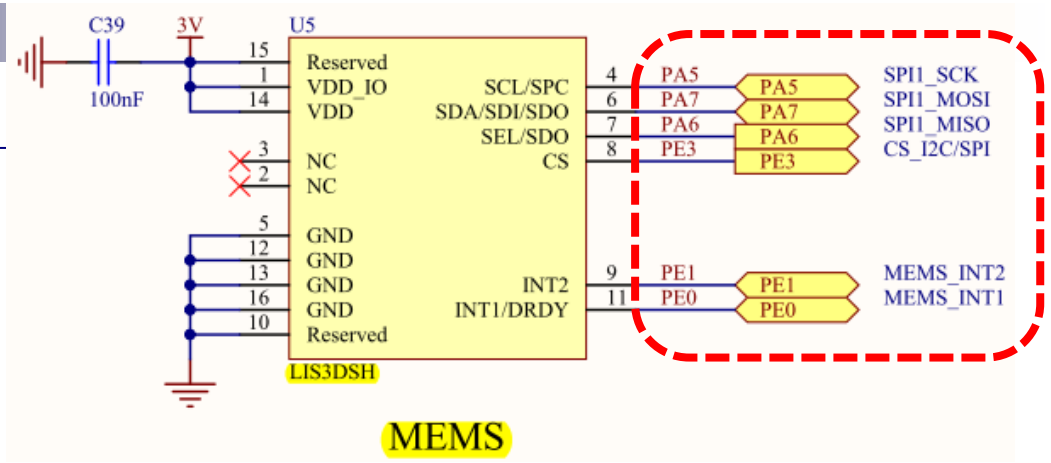


https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples/blob/main/STM32F407_Discovery_kit/LIS3DSH.pdf

VP 4 - STM32 CubeIDE, SPI in LIS3DSH

CubeMX nastavitve :

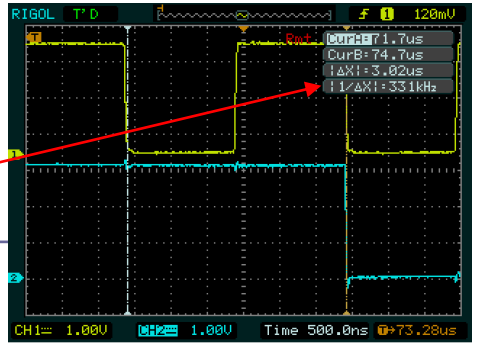




spi.c:

```
/* 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.

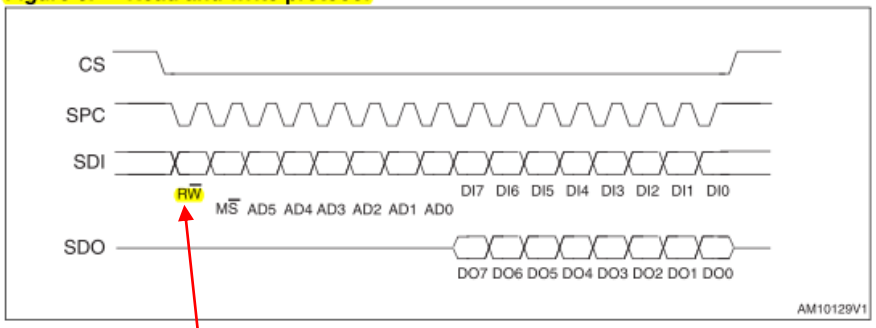
SPI slave timing values

	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

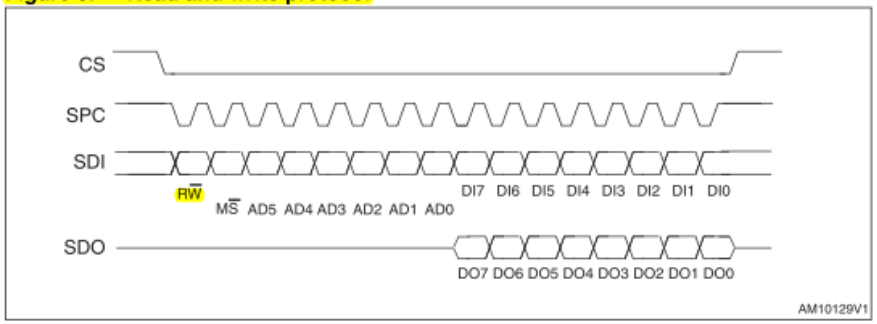
ODR 3:0	Output data rate and power mode selection. Default value:0000 (see *Table 24*)
BDU	Block data update. Default value:0 0:continuos update,1:output registers not updated until MSB and LSB reading)
Zen	Z axis enable. Default value:1 (0:Z axis disabled; 1:Z axis enabled)
Yen	Y axis enable. Default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

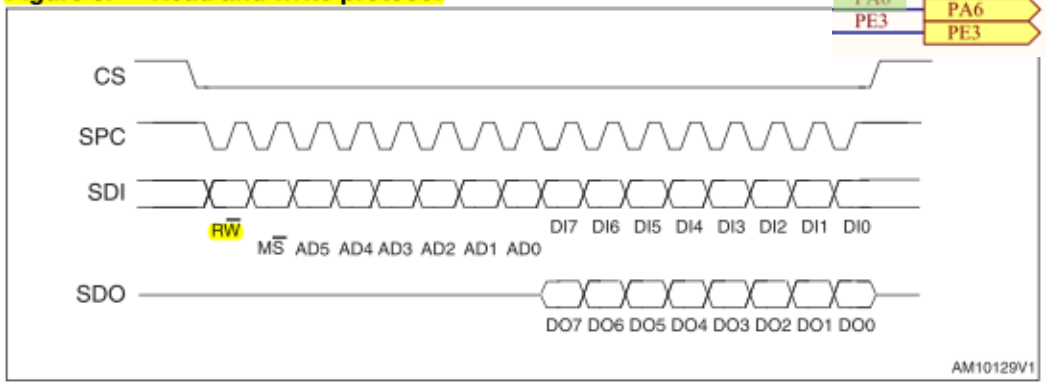
0	0	0	0	0	0	0	0

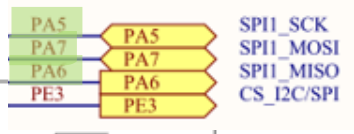
Table 50. OUT_X_H register default value

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

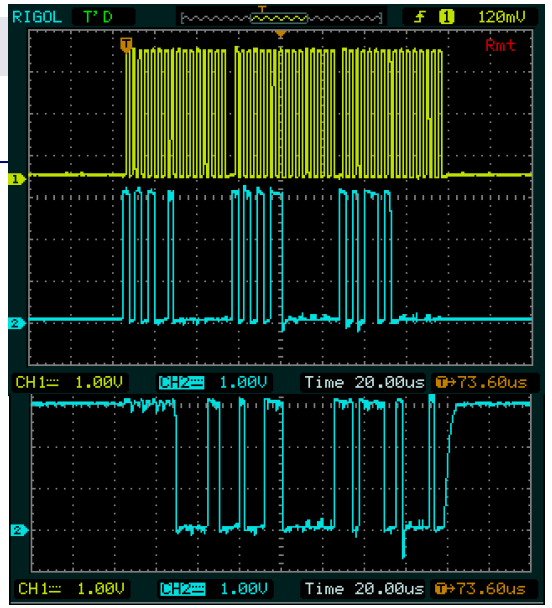




SCK

MOSI

MISO

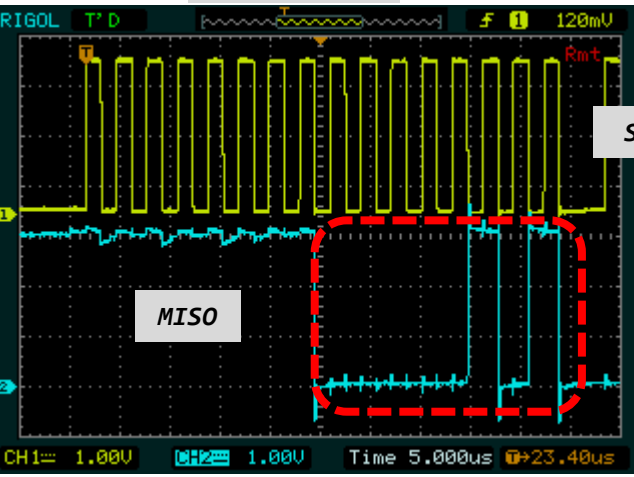


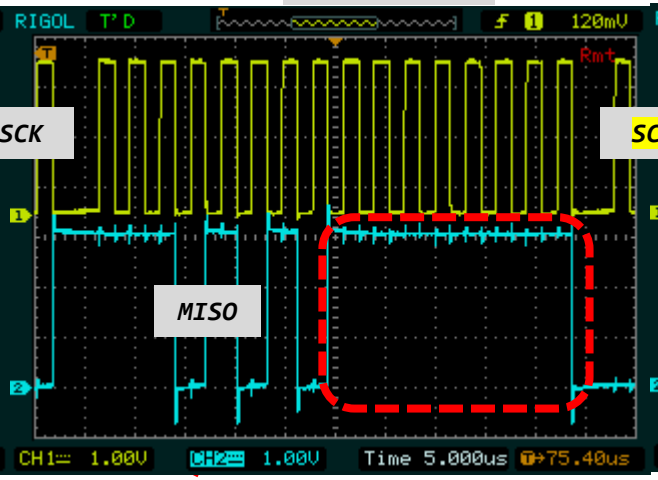

```
Hello World [3530]: Key:0000 Accel[ID:00] X:0005 Y:-1 Z:0066
Hello World [3531]: Key:0000 Accel[ID:00] X:0005 Y:-1 Z:0067
```

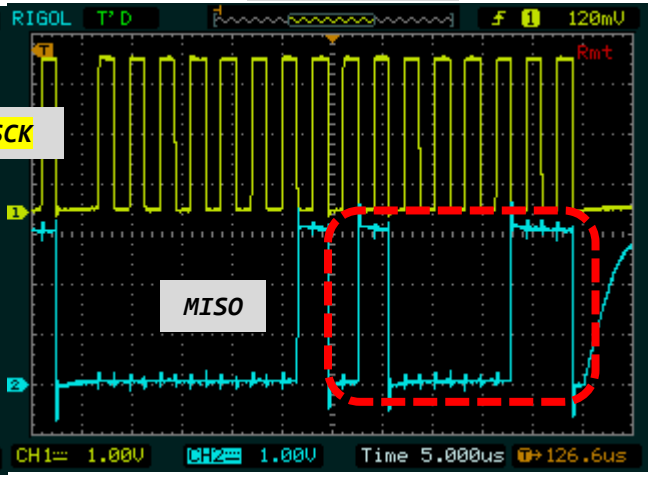
X-Accel: 5

Y-Accel: -1

Y-Accel: 67







VP 5 - 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];

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

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


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

    /* USER CODE END WHILE */
```

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

VIN projekt - VP4: STM32-Edge computing, CubeIDE primeri, Miško3

- VIN projekt
- AI v vgrajenih napravah („Edge Computing“)
- STM32 CubeIDE F4,H7 – PWM izhodi
- STM32F4 CubeIDE: SPI in LIS3DSH, I2C in CS43L22
- STM32H7 CubeIDE, I2C
- Miško3 – demo projekt

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.

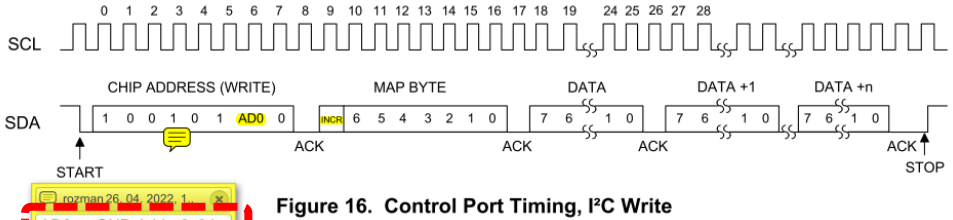


Figure 16. Control Port Timing, I²C Write

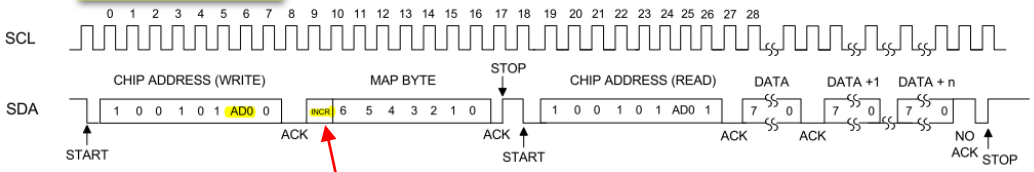


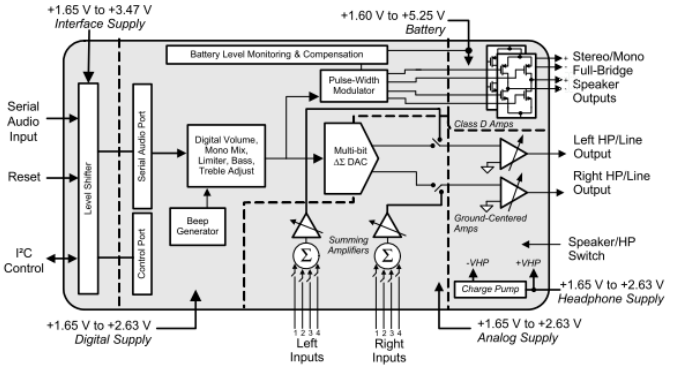
Figure 17. Control Port Timing, I²C Read

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/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

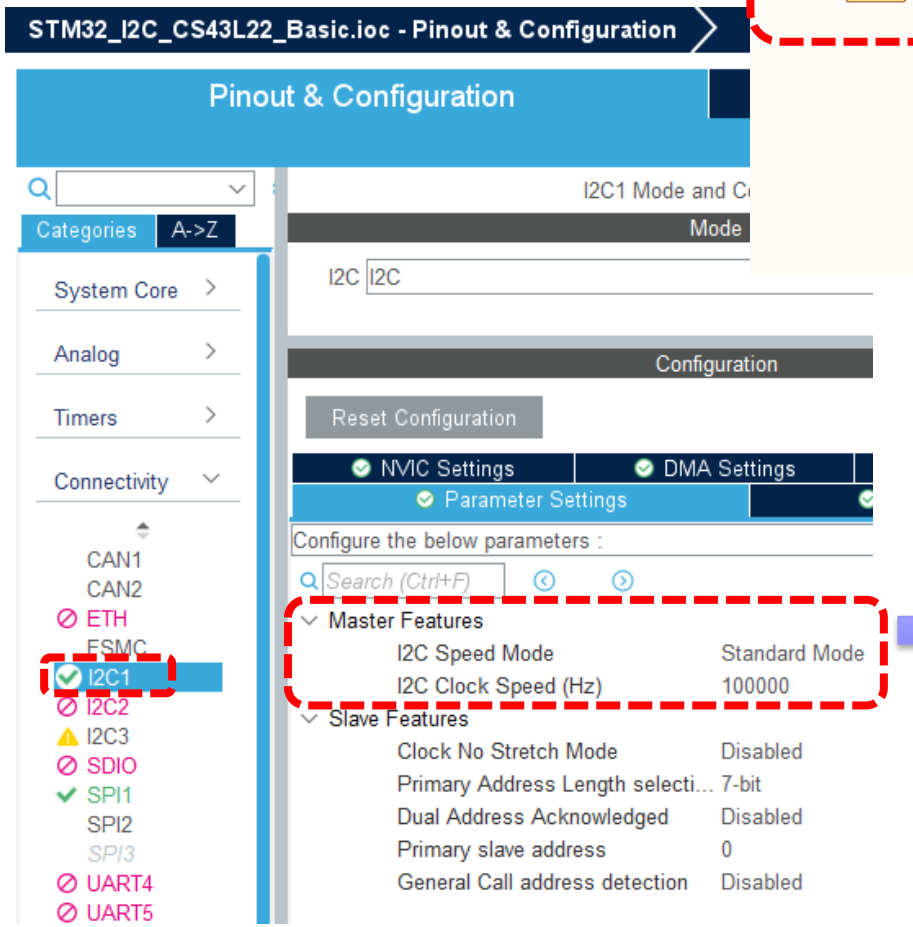
Contents

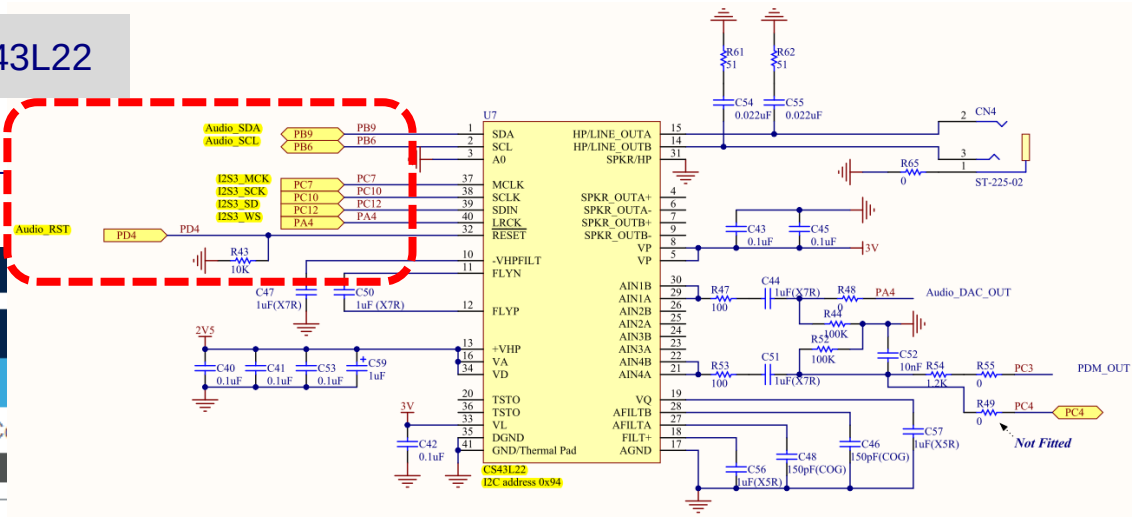
Lastni viri :

https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples

VP 4 - 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 */
#define BUFSIZE 256
charSendBuffer[BUFSIZE];
intCounter;
int KeyState=0;

HAL_StatusTypeDef retval;
uint8_t ChipID;
/* 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 */
```

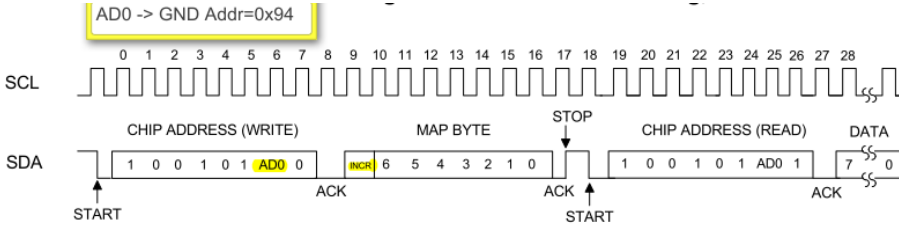


Figure 17. Control Port Timing, I²C Read

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 | Id:%02x \r\n",Counter++,KeyState,ChipID);
  CDC_Transmit_FS(SendBuffer,strlen(SendBuffer));

    /* USER CODE END WHILE */

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

Primer kompleksnejše demo USB-Audio aplikacije :

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

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

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

AN3997 Application note Audio playback and recording using the STM32F4DISCOVERY

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

WavePlayer using STM32 Discovery

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

VIN projekt - VP4: STM32-Edge computing, CubeIDE primeri, Miško3

- VIN projekt
- AI v vgrajenih napravah („Edge Computing“)
- STM32 CubeIDE F4,H7 – PWM izhodi
- STM32F4 CubeIDE: SPI in LIS3DSH, I2C in CS43L22
- STM32H7 CubeIDE, I2C
- Miško3 – demo projekt

VP 4 - STM32H7 CubeIDE, I2C4 scanner

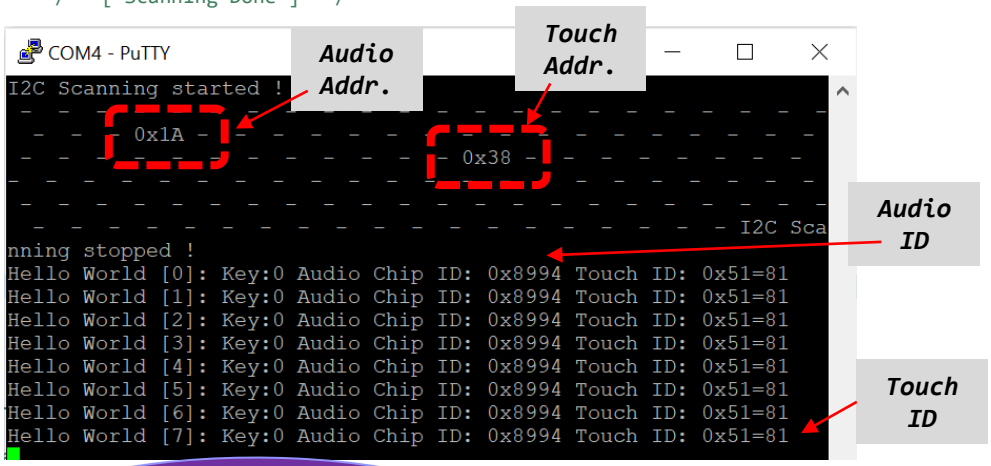
Spremenljivke

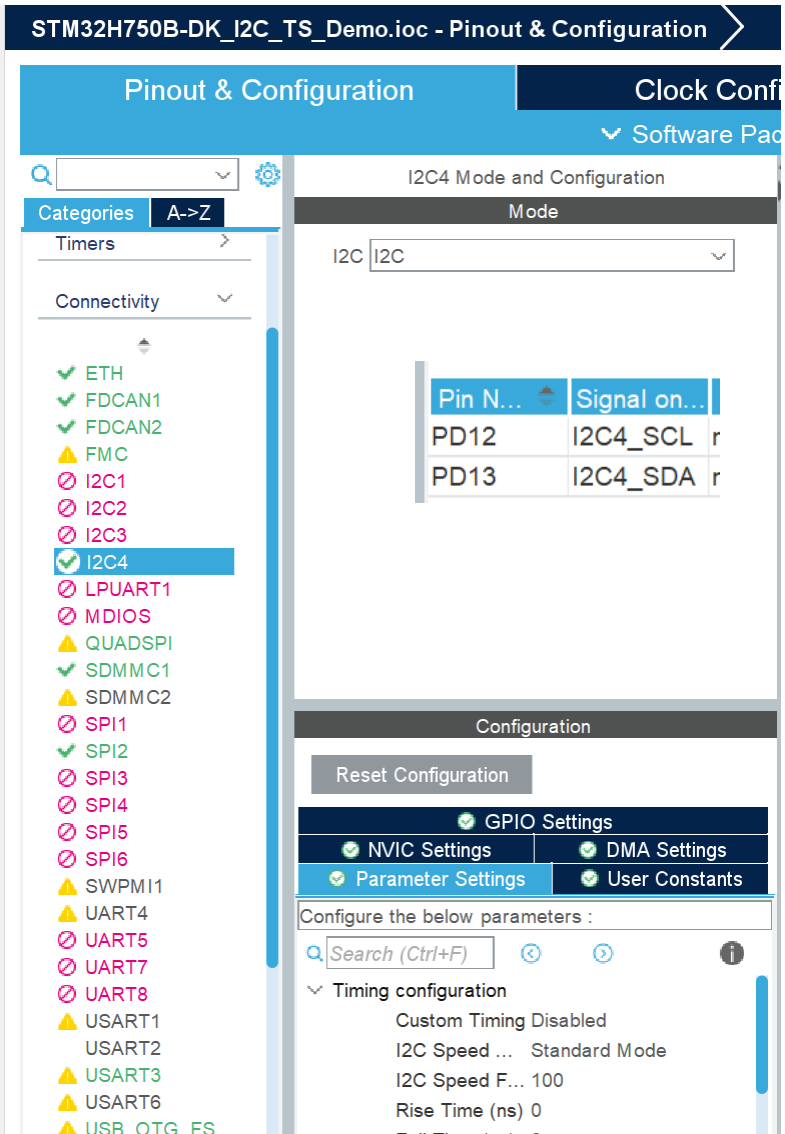
```
/* USER CODE BEGIN PV */
#define BUFSIZE 256
charSendBuffer[BUFSIZE];
intCounter;
int KeyState=0;
uint8_t dataBuffer[10];

HAL_StatusTypeDef retval;
uint8_t Answer;
```

main.c : dodana koda

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%X", i);
                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

VP 4 - STM32H7 CubeIDE, I2C4 scanner

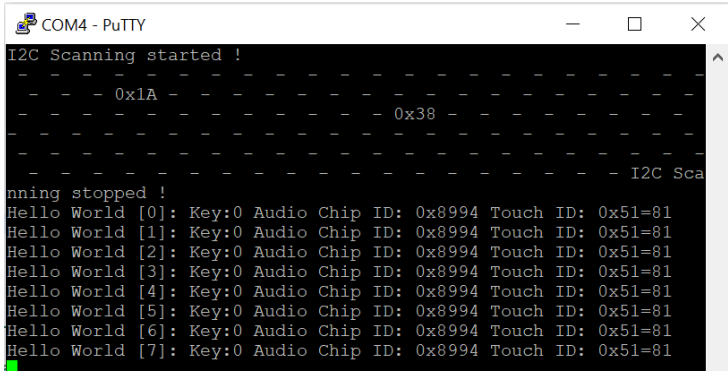
main.c : dodana koda

Spremenljivke

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

Multi-channel Audio Hub CODEC for Smartphones

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.

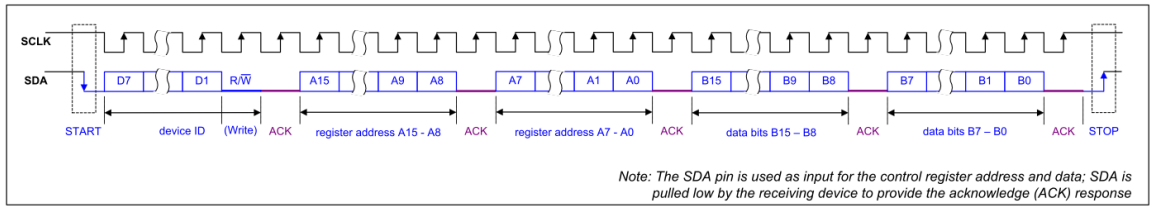


Figure 72 Control Interface 2-wire (I2C) Register Write

[Brez naslova]

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

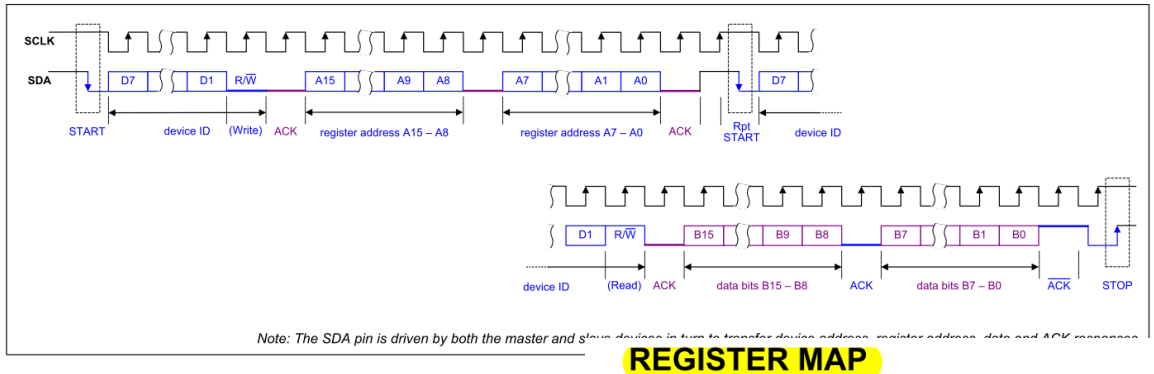
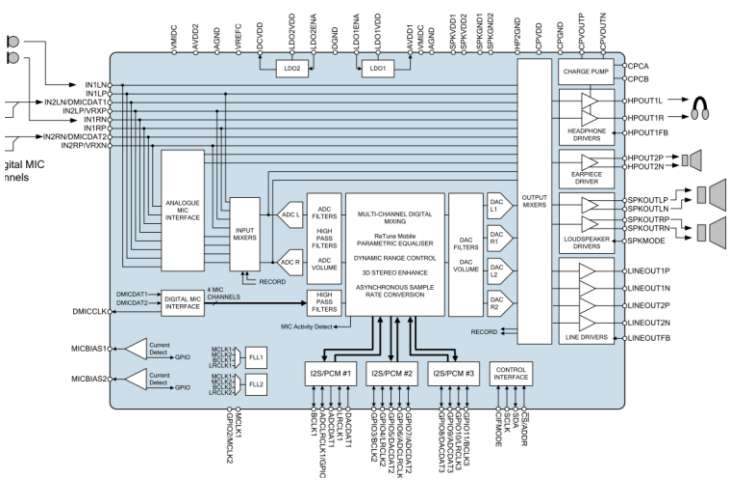


Figure 73 Control Interface 2-wire (I2C) Register Read



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.

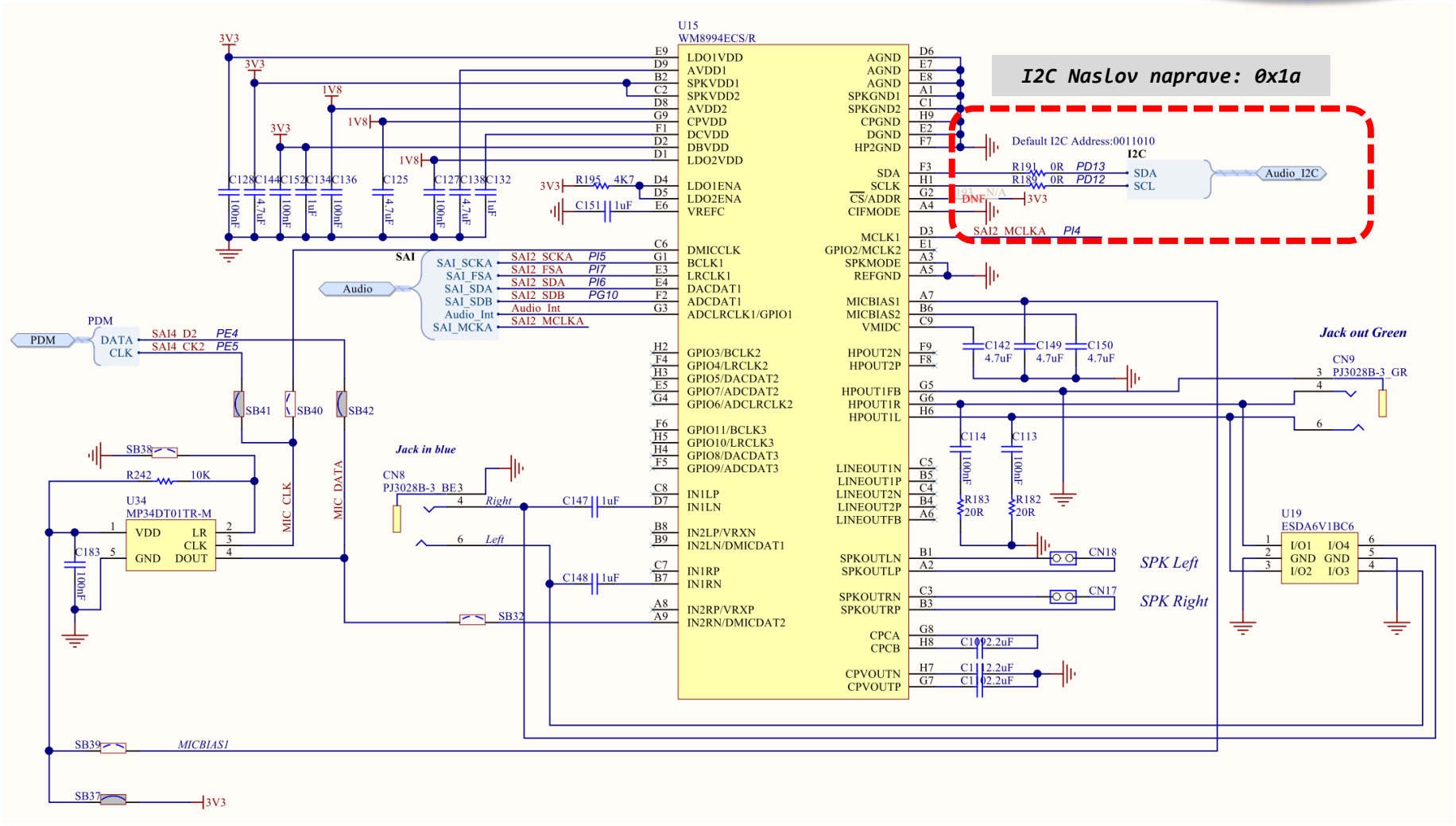
REG	NAME	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	DEFAULT
R0 (0h)	Software Reset	SW_RESET [15:0]																0000h
R1 (1h)	Power Management (1)	0	0	SPKOUTR_ENA	SPKOUTL_ENA	HPOUT2_ENA	0	HPOUT1L_ENA	HPOUT1R_ENA	0	0	MICB2_ENA	MICB1_ENA	0	VMID_SEL [1:0]		BIAS_ENA	0000h
R2 (2h)	Power Management (2)	0	TSHUT_ENA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

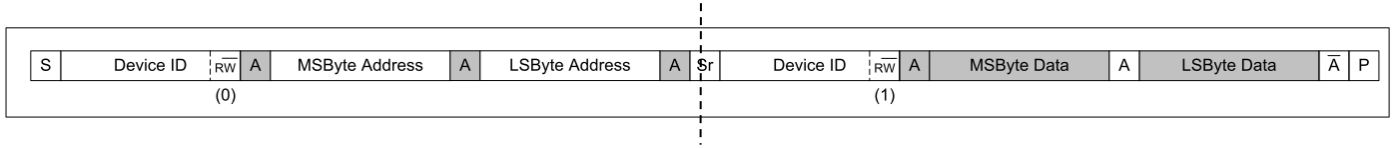


Figure 75 Single Register Read from Specified Address

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 !

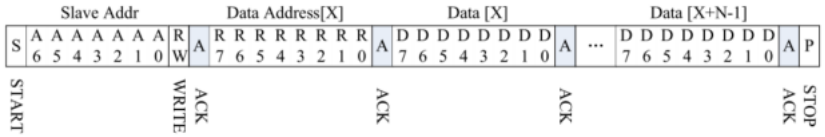
Module P/N: RK043FN48H-CT672B

Version: 1.0

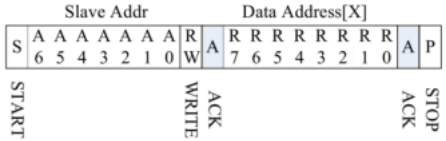
Description : 4.3 inch TFT 480*272 Pixels with LED Backlight and capacitive touch Panel

1.2 I²C Read/Write Interface description

Write N bytes to I2C slave



Set Data Address

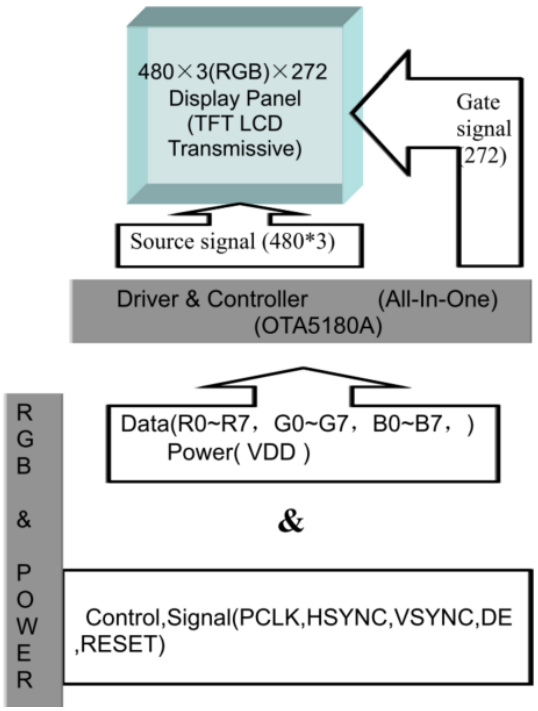


2.1.26 ID_G_ FT5201ID

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

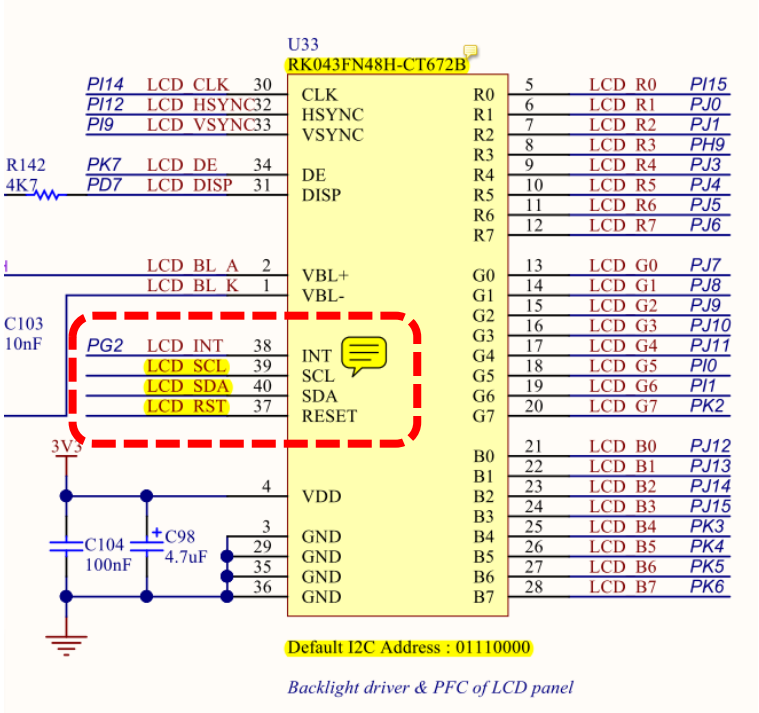
Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Host Access
00h	DEVIDE_MODE		Device Mode[2:0]							RW

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

STM32H7



I2C Naslov naprave: 0x38 (ali 0x70 – pomik)

https://github.com/LAPSyLAB/STM32H7_Discovery_VIN_Projects/tree/main/STM32H750B-DK_I2C_Basic_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 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

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

STM32H7

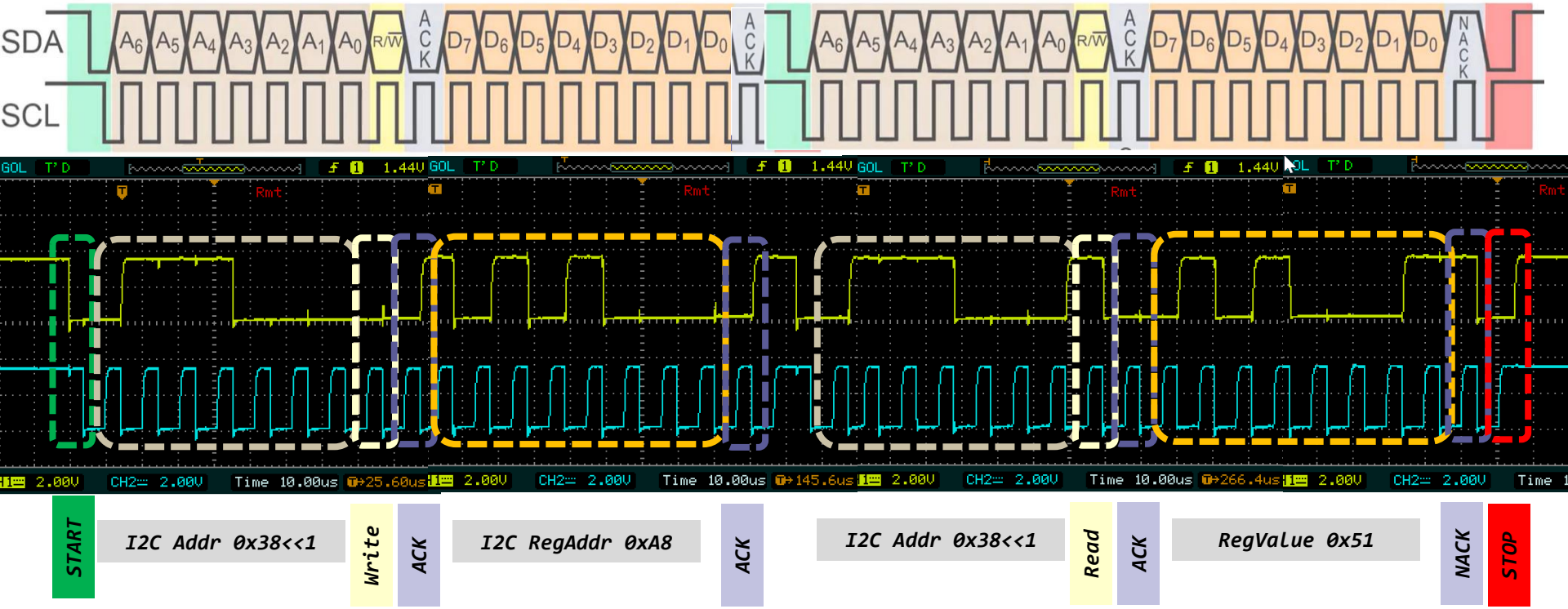
https://github.com/LAPSyLAB/STM32H7_Discovery_VIN_Projects/tree/main/STM32H750B-DK_I2C_Basic_Demo

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

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 */
```

Pomembni registri :

```
// 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);
```

Glavna zanka

```
    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);
```

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

STM32H7

https://github.com/LAPSyLAB/STM32H7_Discovery_VIN_Projects/tree/main/STM32H750B-DK_I2C_Basic_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 */
```

Dodana koda :

```
// 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);
```

Glavna zanka

```
    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);
```

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

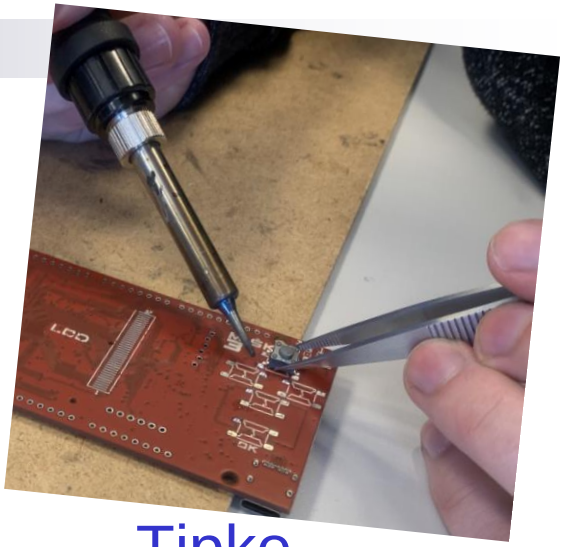
STM32H7

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

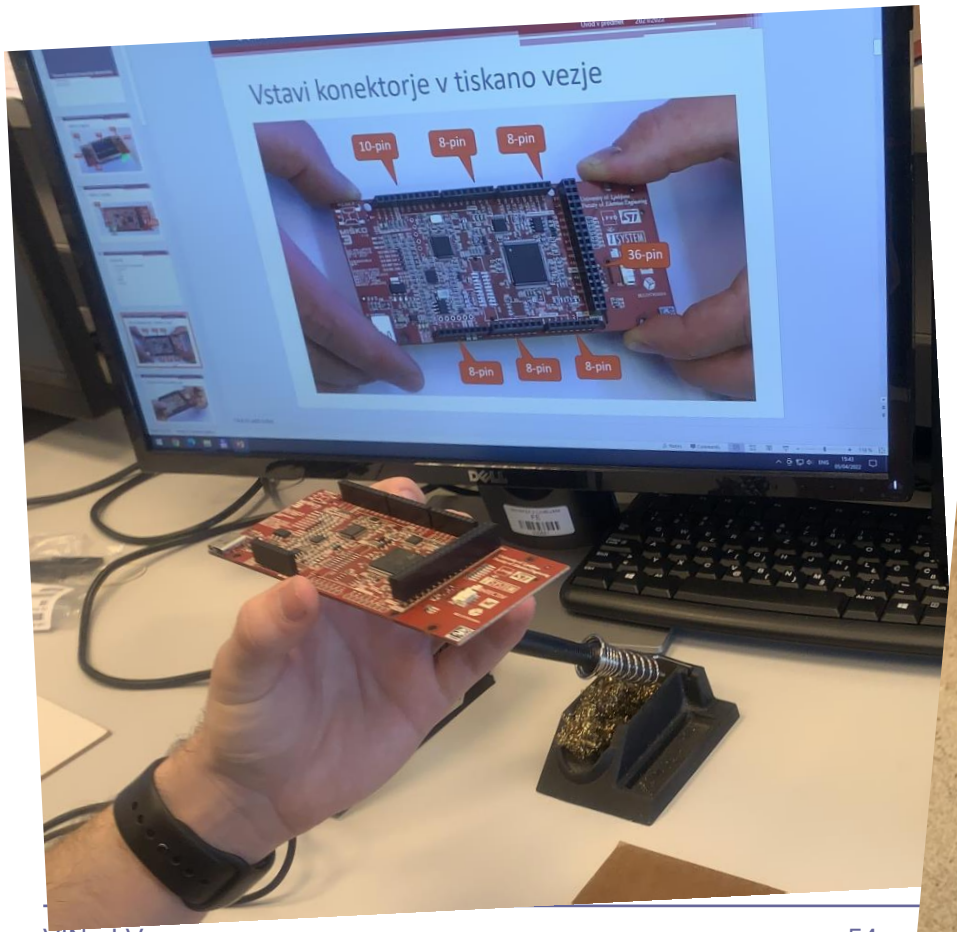
VIN projekt - VP4: STM32-Edge computing, CubeIDE primeri, Miško3

- VIN projekt
- AI v vgrajenih napravah („Edge Computing“)
- STM32 CubeIDE F4,H7 – PWM izhodi
- STM32F4 CubeIDE: SPI in LIS3DSH, I2C in CS43L22
- STM32H7 CubeIDE, I2C
- Miško3 – demo projekt

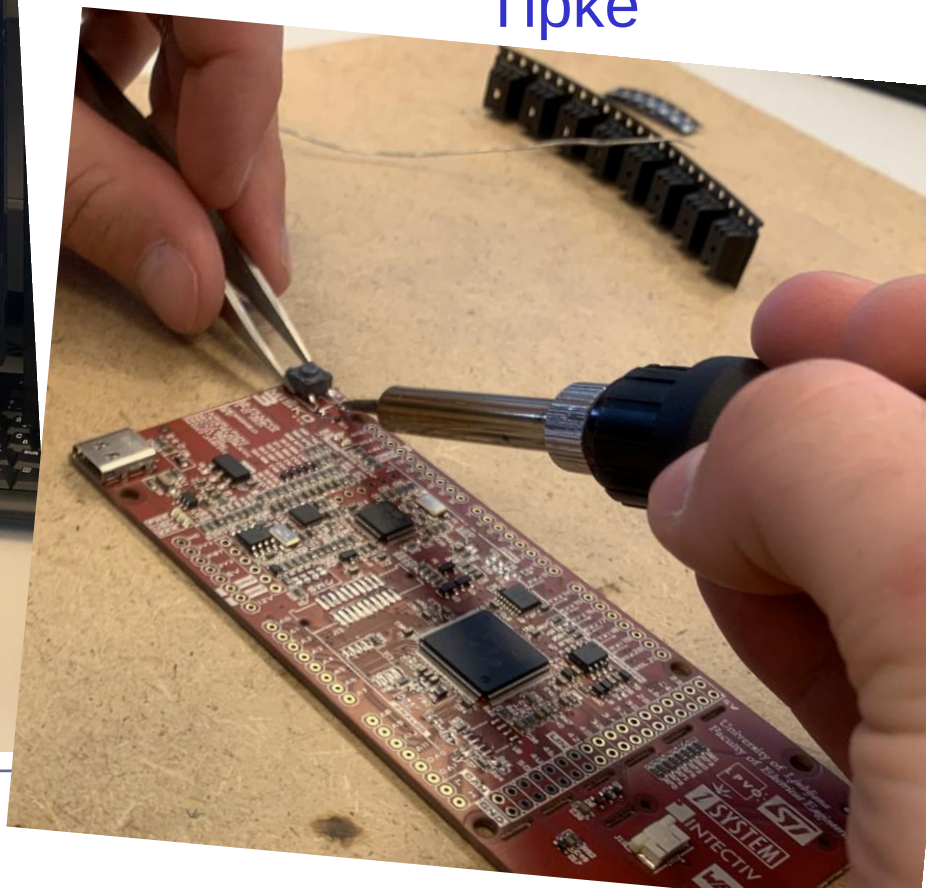
Miško 3 in „Spajka“ party 2022

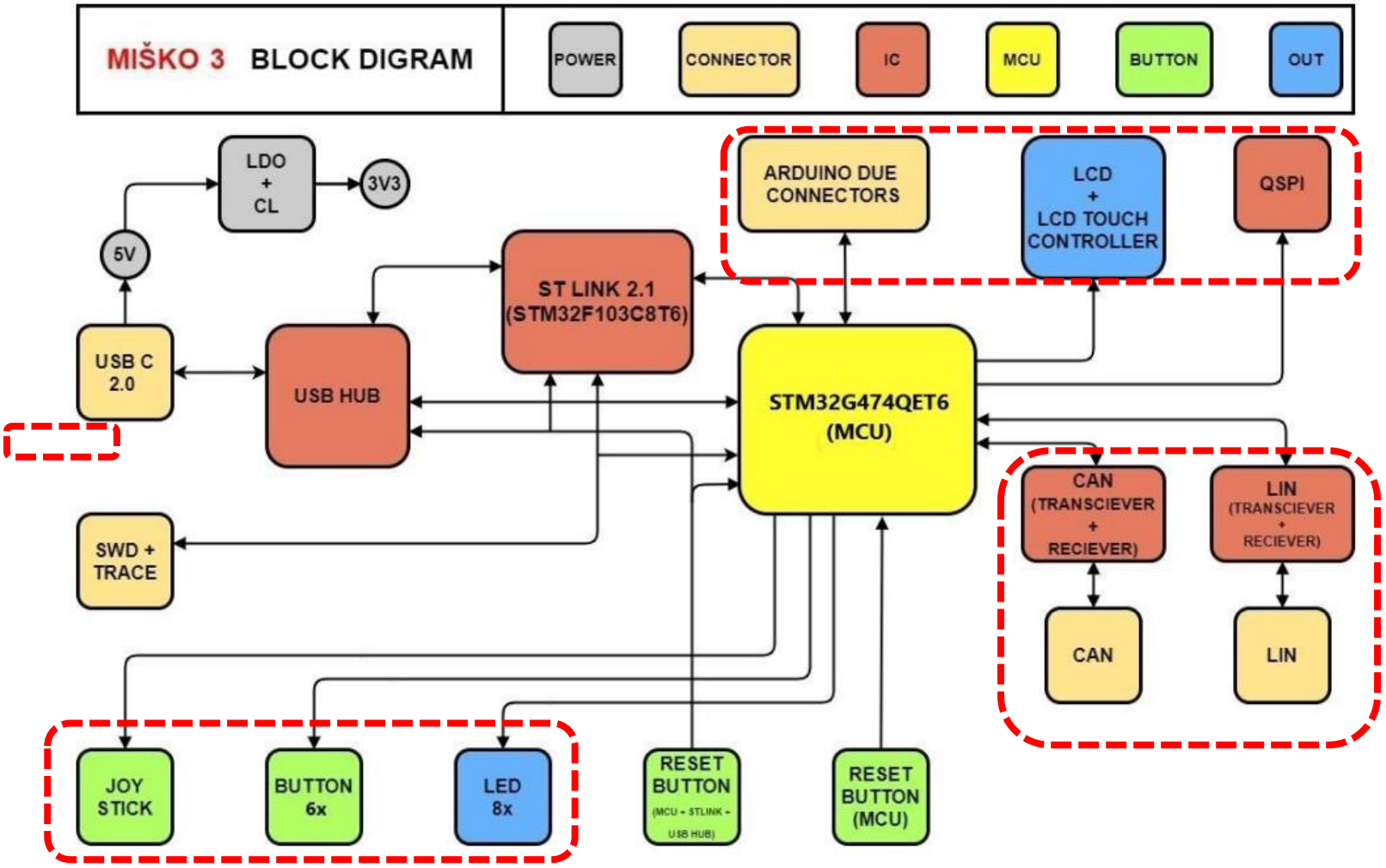


Konektorji



Tipke





Slika 1: Bločni diagram razvojnega Sistema MiŠKo 3

VP 5 – 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 */
```

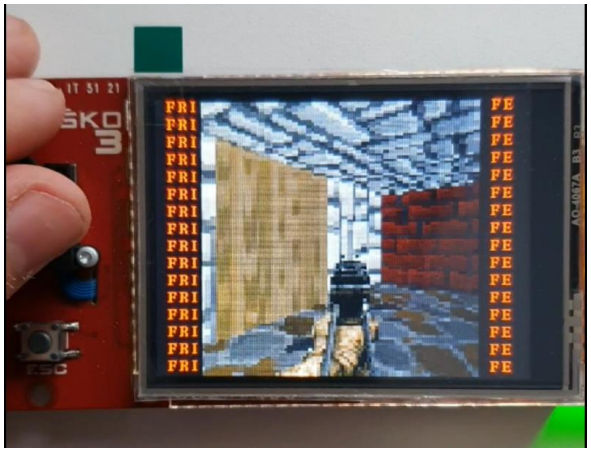
https://github.com/LAPSyLAB/Misko3_Docs_and_Projects

VP 5 – 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



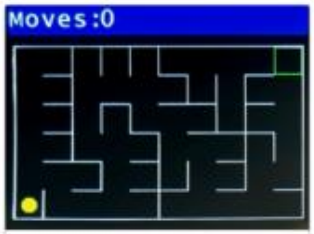
Doom

Maze game

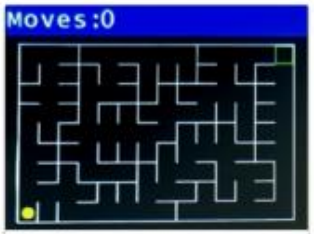
Maze game

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



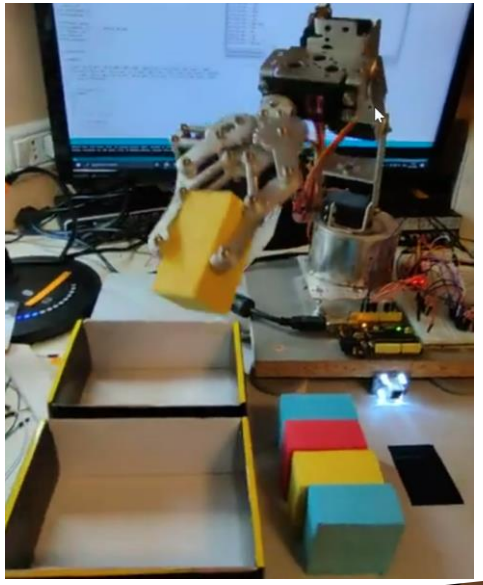
Normal dificulty

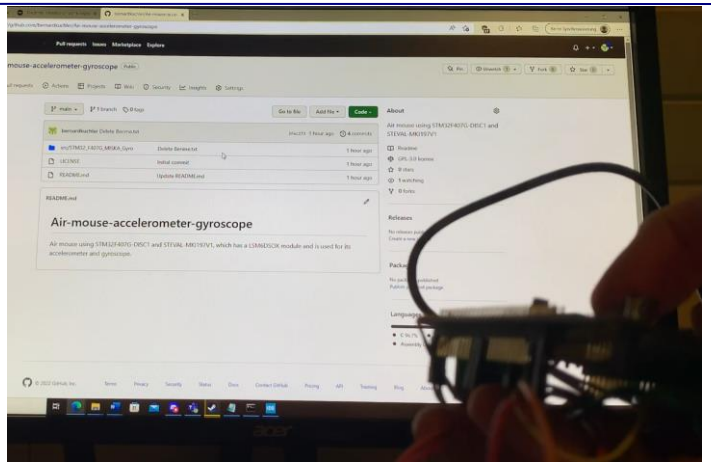
Hard dificulty

https://github.com/LAPSyLAB/Misko3_Docs_and_Projects

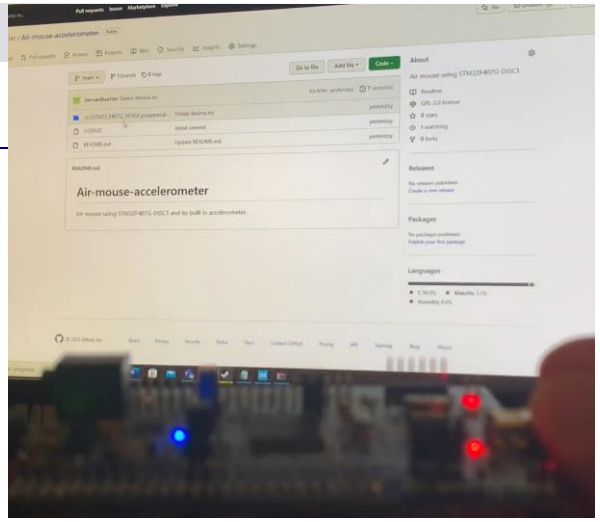
VIN projekt - VP5: STM32-Edge computing, CubeIDE projekti, Miško3

- Diskusija, vprašanja ?

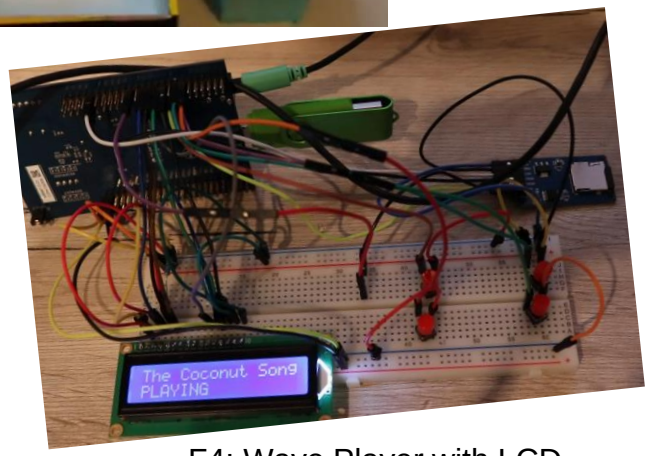




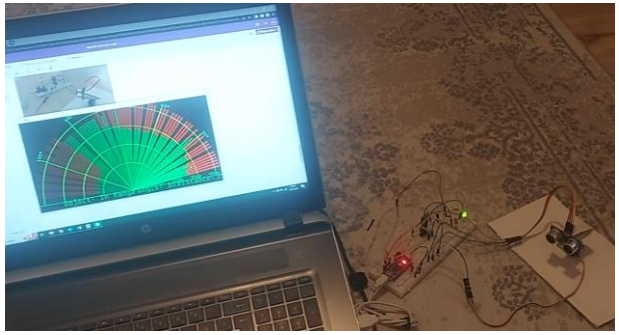
F4: LSM6DSOX – Air Mouse



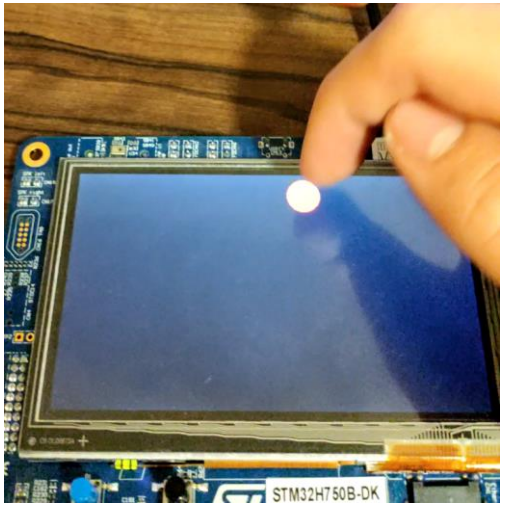
F4: Air Mouse



F4: Wave Player with LCD



3D Sonar



H7: Circle Popper

https://github.com/LAPSyLAB/STM32F4_Docs_and_Examples/

