Laboratorinis darbas Nr. 4.

Slėgio jutiklio LPS25HB nuskaitymas per I2C sąsają.

**Turinys**

[1 Darbo tikslai 2](#_Toc115109348)

[2 Analoginio signalo įvestis 2](#_Toc115109349)

[3 Signalo generavimas panaudojant Skaitmeninį-Analoginį Keitiklį (SAK, angl. DAC) 5](#_Toc115109350)

[3.1 Prievadų funkcijų konfigūravimas 5](#_Toc115109351)

[3.2 Taktavimo sistemos konfigūravimas (Clock system configuration) 7](#_Toc115109352)

[3.3 Signalo generavimo laikmačio konfigūravimas 7](#_Toc115109353)

[3.4 Keil projekto šablono generavimas 9](#_Toc115109354)

[3.5 Projekto šablono papildymas C programos tekstu 9](#_Toc115109355)

[3.5.1 Kuriamo programos kodo paskirtis 9](#_Toc115109356)

[3.5.2 Programinė įranga 10](#_Toc115109357)

[3.6 Programos veikimo tikrinimas 11](#_Toc115109358)

[4 Užduotys 11](#_Toc115109359)

[5 Priedas 1. Skaitmeninio kodo konvertavimas į įtampos vienetus 12](#_Toc115109360)

# Darbo tikslai

1. Išmokti realizuoti analoginio signalo kvantavimą ir diskretizavimą užsiduota sparta (diskretizavimo dažniu), panaudojant ASK (Analoginį-Skaitmeninį Keitiklį, angl. ADC);
2. Išmokti realizuoti analoginio bet kokios formos signalo generavimą su užduotais parametrais (dažniu ir amplitude), panaudojant SAK (Skaitmeninį-Analoginį- Keitiklį, angl. DAC)

# Analoginio signalo įvestis

Kamami makete yra prijungtas skaitmeninis slėgio sensorius LPS25HB (MEMS pressure sensor), kuris matuoja slėgį bei temperatūra.

Diagram, schematic

Description automatically generated A close-up of a computer chip

Description automatically generated with medium confidence

Parodysime, kaip išmatuoti LDR1 varžą. Tam pradžioje išmatuosime įtampą prievade PA1, o paskui pasinaudodami VDD ir R8 vertėmis apskaičiuosime varžą naudodami išraišką (išvedama iš varžinio daliklio įtampos perdavimo lygties)

*,* (1)

čia maketo maitinimo įtampa VDD=3,3 V, atraminė varža .

ASK gali būti nuskaitomas nuolatinės apklausos būdu (polling), aptarnaujant pertrauktį iš keitiklio arba panaudojant tiesioginių mainų kanalą. Kiekvieno iš būdų realizavimui galima naudoti atitinkamas HAL bibliotekos funkcijas ir pavyzdinį šabloną. Programų pavyzdžių šablonų galima rasti kataloge C:\Users\XXX\STM32Cube\Repository\STM32Cube\_FW\_L0\_V1.11.3\Projects\NUCLEO-L073RZ\Examples\I2C\. \* XXX yra katalogas, priskirtas Windows sistemos vartotojui „User“ darbiniame kompiuteryje.

Graphical user interface, text, application

Description automatically generated

Panagrinėsime pavyzdį, kaip kas 1 sekunde nuskaitomi skaitmeninio sensoriaus LPS25HB temperatūros duomenys naudojantis I2C sąsaja bei laikmačiu (TIMER 2). Atsisiuskite pavyzdinį projektą iš Github svetainės:

<https://github.com/euronymous1991/LPS25HB_MCU_LAB>

|  |  |
| --- | --- |
| Failo pavadinimas | Atliekama funkcija |
| 4DIG\_7SEG\_DISPLAY.c  4DIG\_7SEG\_DISPLAY.h | 4 skaitmenų 7 segmentų ekranėlio aptarnavimas |
| LPS25HB.c  LPS25HB.h | Skaitmeninio LPS25HB sensoriaus aptarnavimas |

Toliau detaliau aptarsime LPS25HB.c ir LPS25HB.h failus, bei kaip jų pagalba nuskaitomi temperatūros rodmenys, žemiau pateikti šių failų kodai:

*LPS25HB.h*

|  |
| --- |
| /\*---------------------Adresai------------------------------------------\*/  #define LPS25HB\_ADDRESS 0x5C << 1 //Note that SA0 = 0 so address is 1011101 and not 1011100  #define LPS25HB\_DEVICE\_ID 0xBD //Device ID, the value in the WHO\_AM\_I Register  /\*---------------------Registrai-------------------------------\*/  #define LPS25HB\_WHO\_AM\_I 0x0F //Who am I register location  #define LPS25HB\_STATUS\_REG 0x27  //Tells whether the Pressure Data is ready or is being overrun  #define LPS25HB\_TEMP\_OUT\_L 0x2B // Temperature output value (LSB)  #define LPS25HB\_TEMP\_OUT\_H 0x2C // Temperature output value (MSB)  #define LPS25HB\_CTRL\_REG1 0x20 //Contains PD, BDU and more  #define LPS25HB\_CTRL\_REG2 0x21 //Contains one-shot mode and FIFO settings  #define LPS25HB\_RES\_CONF 0x10 //Pressure and temperature Resolution  /\*---------------------Konfiguraciniai bitai--------------------------------\*/  #define LPS25HB\_CTRL\_REG1\_PD 0x80 //Power Down when 0, active mode when 1 (Default 0)  #define LPS25HB\_CTRL\_REG2\_ONE\_SHOT 0x1 //One shot mode enabled, obtains a new dataset  #define LPS25HB\_RES\_CONF\_AVGP0 0x1 //Pressure resolution Configuration  #define LPS25HB\_RES\_CONF\_AVGP1 0x2 //Pressure resolution Configuration  #define LPS25HB\_STATUS\_REG\_PDA 0x2 //Pressure data available  /\* Private function prototypes -----------------------------------------------\*/  /\* USER CODE BEGIN PFP \*/  uint8\_t LPS25HB\_Initialise**(**I2C\_HandleTypeDef **\***i2cHandle**);**  float LPS25HB\_Measure\_Temperature**(**I2C\_HandleTypeDef **\***i2cHandle**);**  /\* USER CODE END PFP \*/ |

*LPS25HB.c*

|  |
| --- |
| #include "main.h"  #include "LPS25HB.h"  uint8\_t chipID**;**  uint8\_t datatowrite **=** 0**;**  uint8\_t TEMP\_OUT\_H**,** TEMP\_OUT\_L**;** // RAW DATA  int16\_t val**;**  uint8\_t LPS25HB\_Initialise**(**I2C\_HandleTypeDef **\***i2cHandle**){**    // Pradzioje patikriname ID  HAL\_I2C\_Mem\_Read**(** i2cHandle**,** LPS25HB\_ADDRESS**,** LPS25HB\_WHO\_AM\_I**,** 1**,** **&** chipID**,** 1**,** 1000**);**  if **(** chipID **!=** LPS25HB\_DEVICE\_ID**)** **{**  return 1**;}** //jei ID nesutampa inicializacija nutraukiama    datatowrite **=** LPS25HB\_CTRL\_REG1\_PD**;**  if **(**HAL\_I2C\_Mem\_Write**(** i2cHandle**,** LPS25HB\_ADDRESS**,** LPS25HB\_CTRL\_REG1**,** 1**,** **&**datatowrite**,** 1**,** 100**)** **!=** HAL\_OK**)** **{**  return 1**;**  **}**  return 0**;**  **}**  float LPS25HB\_Measure\_Temperature**(**I2C\_HandleTypeDef **\***i2cHandle**)** **{**    datatowrite **=** LPS25HB\_CTRL\_REG2\_ONE\_SHOT**;** // One shot mode  if **(**HAL\_I2C\_Mem\_Write**(** i2cHandle**,** LPS25HB\_ADDRESS**,** LPS25HB\_CTRL\_REG2**,** 1**,** **&**datatowrite**,** 1**,** 100**)** **!=** HAL\_OK**)** **{**  return 1**;**  **}**    HAL\_I2C\_Mem\_Read**(** i2cHandle**,** LPS25HB\_ADDRESS**,** 0x2B**,** 1**,** **&**TEMP\_OUT\_L**,** 1**,** 100**);**  HAL\_I2C\_Mem\_Read**(** i2cHandle**,** LPS25HB\_ADDRESS**,** 0x2C**,** 1**,** **&**TEMP\_OUT\_H**,** 1**,** 100**);**  val **=** **(**TEMP\_OUT\_H **<<** 8**)** **|** **(**TEMP\_OUT\_L**);**  return 42.5f **+** val **/** 480.0f**;**  **}** |

LPS25HB.h preambulės faile surašyti reikalingi adresai, registrai bei konfiguraciniai bitai reikalingi temperatūros nuskaitymui, ši informacija yra paimta iš LPS25HB datasheet registro žemėlapio (Register mapping).

Table

Description automatically generated

Table

Description automatically generated

Bei dviejų vienintelių funkcijų prototipai:

|  |
| --- |
| uint8\_t LPS25HB\_Initialise**(**I2C\_HandleTypeDef **\***i2cHandle**);** |

**ir**

|  |
| --- |
| float LPS25HB\_Measure\_Temperature**(**I2C\_HandleTypeDef **\***i2cHandle**);** |

LPS25HB.c faile yra pateikiamas prieš tai minėtų dviejų funkcijų kodai. LPS25HB\_Initialise**(**I2C\_HandleTypeDef **\***i2cHandle**);**

Ši funkcija skirta sensoriaus inicializavimui panagrinėkime šią funkcija atidžiau.

|  |
| --- |
| uint8\_t LPS25HB\_Initialise**(**I2C\_HandleTypeDef **\***i2cHandle**){**    // Pradzioje patikriname ID  HAL\_I2C\_Mem\_Read**(** i2cHandle**,** LPS25HB\_ADDRESS**,** LPS25HB\_WHO\_AM\_I**,** 1**,** **&** chipID**,** 1**,** 1000**);**  if **(** chipID **!=** LPS25HB\_DEVICE\_ID**)** **{**  return 1**;}** //jei ID nesutampa inicializacija nutraukiama    datatowrite **=** LPS25HB\_CTRL\_REG1\_PD**;**  if **(**HAL\_I2C\_Mem\_Write**(** i2cHandle**,** LPS25HB\_ADDRESS**,** LPS25HB\_CTRL\_REG1**,** 1**,** **&**datatowrite**,** 1**,** 100**)** **!=** HAL\_OK**)** **{**  return 1**;**  **}**  return 0**;**  **}** |

LPS25HB turi naudinga registrą pavadinimu WHO\_AM\_I (0Fh), kurį nuskaičius gaunamas sensoriaus ID, kuris LPS25HB atveju turėtų būti 1011 1101 (0xBD).

Chart

Description automatically generated Tai galima atlikti pasinaudojant žemiau pavaizduota funkcija:

|  |
| --- |
| HAL\_I2C\_Mem\_Read**(** i2cHandle**,** LPS25HB\_ADDRESS**,** LPS25HB\_WHO\_AM\_I**,** 1**,** **&**chipID**,** 1**,** 1000**);** |

Naudojantis šia funkcija nuskaitomas LPS25HB\_WHO\_AM\_I registre esanti reikšmė kuri išsaugoma kintamajame chipID, jeigu registre esanti reikšmė nesutampa su datasheet pažymėta reikšme tai yra 0xBD, tolimesnė LPS25HB inicializacija yra beprasmė, dėl to išeinama iš funkcijos ir pranešamas klaidos pranešimas šiuo atveju return 1.

|  |
| --- |
| if **(** chipID **!=** LPS25HB\_DEVICE\_ID**)** **{**  return 1**;}** //jei ID nesutampa inicializacija nutraukiama |

Atkreipkime dėmesį į registrą CTRL\_REG1 (20h)

Table

Description automatically generated

CTRL\_REG1 (20h) registre vyriausio bito (PD) pagalba galima aktyvuoti arba deaktyvuoti sensorių, kaip matome iš datasheet, pradinė vertė yra nulis, kas reiškia, kad sensorius yra neaktyvus. Norint atlikti sensoriaus aktyvacijai iš LPS25HB.h paimamas konfiguracinis bitas:

|  |
| --- |
| #define LPS25HB\_CTRL\_REG1\_PD 0x80 //Power Down when 0, active mode when 1 (Default 0) |

Ir įrašomas į CTRL\_REG1 registrą naudojantis žemiau pateikta funkcija:

|  |
| --- |
| datatowrite **=** LPS25HB\_CTRL\_REG1\_PD**;**  if **(**HAL\_I2C\_Mem\_Write**(** i2cHandle**,** LPS25HB\_ADDRESS**,** LPS25HB\_CTRL\_REG1**,** 1**,** **&**datatowrite**,** 1**,** 100**)** **!=** HAL\_OK**)** **{**  return 1**;**  **}** |

Jei komunikacija tarp sensoriaus ir mikrovaldiklio nepavyksta ir HAL\_I2C\_Mem\_Write funkcija gražina ne HAL\_OK statusą, išmetamas klaidos pranešimas (return 1), jei viskas įvyksta be klaidų baigdami inicializacijos procedūrą gražiname 0 (return 0).

Taip pat galima atkreipti dėmesį į kitus konfigūracinius bitus kaip ODR [2:0], kurių pagalba galima konfigūruoti duomenų nuskaitymo dažnį tačiau šiam pavyzdžiui LPS25HB bus naudojamas „ONESHOT“ rėžimas, dėl to jų keisti šiai akimirkai nereikia ir galima palikti pradines vertes tai yra „000“.

Toliau projekte belieka vienintelė funkcija tai yra:

|  |
| --- |
| float LPS25HB\_Measure\_Temperature**(**I2C\_HandleTypeDef **\***i2cHandle**);** |

Kuri atlieka temperatūros nuskaitymą, bei neapdorotų duomenų konvertavimą į Celsijus, panagrinėkime šią funkcija atidžiau.

|  |
| --- |
| float LPS25HB\_Measure\_Temperature**(**I2C\_HandleTypeDef **\***i2cHandle**)** **{**    datatowrite **=** LPS25HB\_CTRL\_REG2\_ONE\_SHOT**;** // One shot mode  if **(**HAL\_I2C\_Mem\_Write**(** i2cHandle**,** LPS25HB\_ADDRESS**,** LPS25HB\_CTRL\_REG2**,** 1**,** **&**datatowrite**,** 1**,** 100**)** **!=** HAL\_OK**)** **{**  return 1**;**  **}**    HAL\_I2C\_Mem\_Read**(** i2cHandle**,** LPS25HB\_ADDRESS**,** 0x2B**,** 1**,** **&**TEMP\_OUT\_L**,** 1**,** 100**);**  HAL\_I2C\_Mem\_Read**(** i2cHandle**,** LPS25HB\_ADDRESS**,** 0x2C**,** 1**,** **&**TEMP\_OUT\_H**,** 1**,** 100**);**  val **=** **(**TEMP\_OUT\_H **<<** 8**)** **|** **(**TEMP\_OUT\_L**);**  return 42.5f **+** val **/** 480.0f**;**  **}** |

Kaip buvo pastebėta iš registro CTRL\_REG1 jei parenkiamas „ONESHOT“ veikimo rėžimas reikia pasinaudoti CTRL\_REG2 registru norint gauti kiekvieną kartą naujus duomenis iš LPS25HB sensoriaus. CTRL\_REG2 (21h) registras:

Table

Description automatically generated

Norint kad LPS25HB išduotų naujus duomenis ONE\_SHOT rėžimu, CTRL\_REG2 registro jauniausiame bite ONE\_SHOT reikia įrašyti vienetą, kurį galima pasiimti iš LPS25HB.h failo.

|  |
| --- |
| #define LPS25HB\_CTRL\_REG2\_ONE\_SHOT 0x1 //One shot mode enabled, obtains a new dataset |

Bei jį įrašyti panašiu principu kaip inicializacijoje:

|  |
| --- |
| datatowrite **=** LPS25HB\_CTRL\_REG2\_ONE\_SHOT**;** // One shot mode  if **(**HAL\_I2C\_Mem\_Write**(** i2cHandle**,** LPS25HB\_ADDRESS**,** LPS25HB\_CTRL\_REG2**,** 1**,** **&**datatowrite**,** 1**,** 100**)** **!=** HAL\_OK**)** **{**  return 1**;**  **}** |

Tai padarius naujausi temperatūros duomenys atsiranda temperatūros registruose kuriuos galima nuskaityti, temperatūros sensorius yra 16 bitų, dėl to šie duomenys yra saugomi dvejuose registruose TEMP\_OUT\_L(2Bh) LSB dalis ir TEMP\_OUT\_H(2Ch) MSB dalis.

Table

Description automatically generated

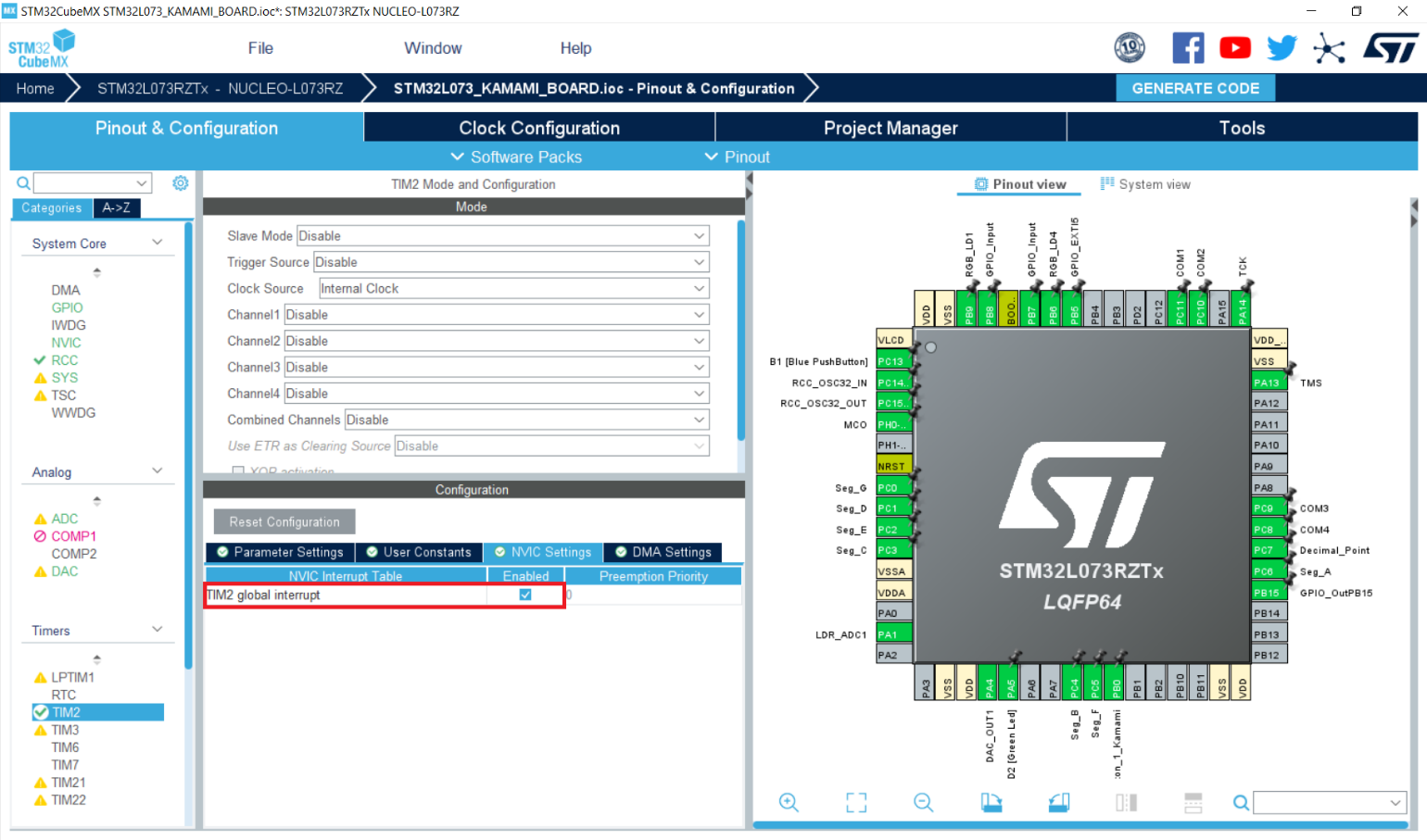
Table

Description automatically generated

Šie abu registrai nuskaitomi dvejomis funkcijomis ir išsaugomi kintamuosiuose TEMP\_OUT\_L ir TEMP\_OUT\_H.

|  |
| --- |
| HAL\_I2C\_Mem\_Read**(** i2cHandle**,** LPS25HB\_ADDRESS**,** 0x2B**,** 1**,** **&**TEMP\_OUT\_L**,** 1**,** 100**);**  HAL\_I2C\_Mem\_Read**(** i2cHandle**,** LPS25HB\_ADDRESS**,** 0x2C**,** 1**,** **&**TEMP\_OUT\_H**,** 1**,** 100**);** |

Stabdymo tašką (breakpoint) kaip parodyta žemiau ir derinimo režime kintamųjų peržiūros lange patikrinkite, kaip priklauso ASK išmatuojamas kodas (ADCpa1), suskaičiuota įtampa (Vpa1) ir suskaičiuota fotorezistoriaus varža (Rldr), priklausomai nuo LDR1 apšvietimo. Apšvietimą galima kaitalioti pridengiant lapeliu LDR1 korpusą.



# Užduotys

1. Pakeiskite programa taip jog 4 skaitmenų 7 segmentų ekranėlyje būtų atvaizduojama temperatūra su kableliu, temperatūros duomenys turi būti atnaujinami tokiu pri.
2. Sukurkite naują funkciją kurios pagalba būtų nuskaitomi slėgio (mBar) duomenys, bei jie atvaizduojami 4 skaitmenų 7 segmentų ekranėlyje.
3. Pakeiskite programą taip kad kiekvieną kartą paspaudus mygtuką pasikeistų ekrane atvaizduojami duomenys iš slėgio į temperatūra ir atgal.