Univerzitet u Nišu

ELEKTRONSKI FAKULTET

PROJEKAT

Tema: “Digitalni akcelerometar”

Predmet: Napredni mikrokontroleri

Mentor:

Prof. dr Goran Nikolić

Studenti:

Milutin Dinić 17596

Stefan Milojević 17778

Aleksandar Pantović 17860

Niš, 2023.

SADRŽAJ

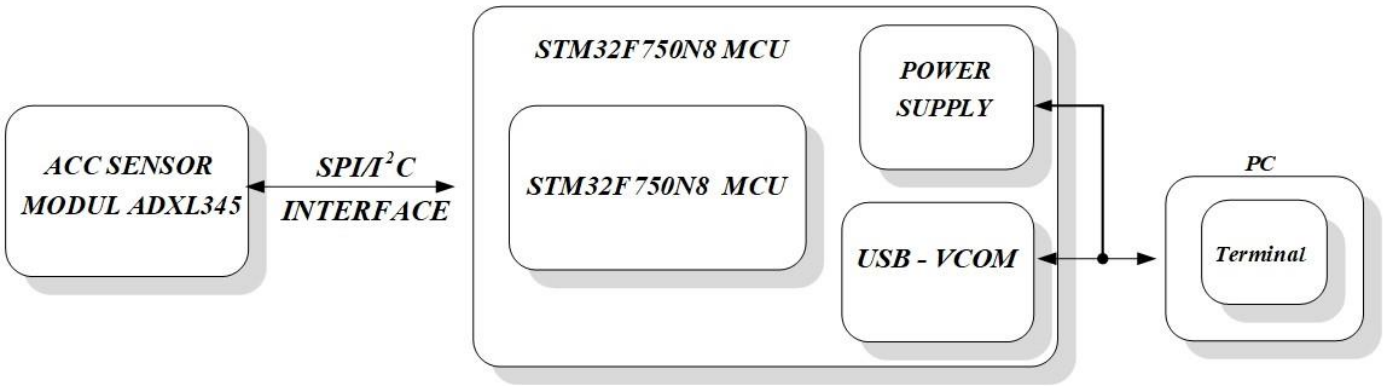
UVOD

Cilj projekta je merenje sile gravitacije posredstvom akcelerometra **ADXL345**. To je elektromehanička komponenta koja u sebi sadrži mikro-strukturekoje se savijaju usled momenta i gravitacije. Merenje ubrzanja je moguće usled savijanja mikro-struktura, čija je deformacija proporcionalna ubrzanju objekta koji se meri. Spregnut sa mikrokontrolerom **STM32F750DK**, koji ima ulogu kontrolne jedinice, u odgovrajućem režimu rada (merenja), senzor meri sile ubrzanja. Ovesile mogu biti statičke prirode, kao što je sila gravitacije ili dinamičkog tipa,uzrokovane pomeranjem uređaja na kome se akcelerometar nalazi. Ovaj akcelerometar, kao i drugi senzori, interno jeste analgoni senzor, međutim, pored toga u sebi sadrži AD-konvertor, memorijski prostor u kome se čuvaju kalibracionifaktori kao i odgovrajuću podršku za interfejs tj. podršku za I2C i SPI komunikacioni standard. Ovakve podrške nam omogućavaju digitalno sprezanje akcelerometra sa odgovrajućim mikrokontrolerom (u našem slučaju samikrokontrolerom STM32F750DK).

Parametri akscelrometra koji se u ovom projektu uzimaju u obzir su:

- Izlazni signal modula koji može biti analogni ili digitalni. Zbog spomenutog sprezanja akcelerometra sa mikrokontrolerom, neophodna je spomenuta podrška digitalnog interfejsa.
- Broj osa koje se koriste za merenje ubrzanja.
- Opseg merenja – predstavlja opseg maksimalnog ubrzanja koje može biti izmereno.
- Tip protokola koji se koristi u komunikaciji između mastera i slave-a.
- Osetljivost – Pouzdanost rezultata merenja direkno je srazmeran ovom parametru.

U okviru projekta obrađen je režim merenja. Nakon procesa merenja i digitalizacije, mikrokontroler preko I2C interfejsa čita izmerene vrednosti koje predstavljaju “sirove” podatke, koje je kasnije potrebno obraditi, što činimo korišćenjem transfer funkcije senzorskog modula.



Slika 1: Sprega uC-a i senzorskog mod

KRATAK OPIS RAZVOJNOG OKRUZENJA

STM32F750DK je razvojna ploča zasnovana na mikrokontroleru STM32F750VBH6 iz serije STM32F7, proizvedenoj od strane STMicroelectronics. Ova ploča je namenjena za razvoj aplikacija i prototipiranje, posebno u oblastima kao što su IoT (Internet stvari), industrijska automatizacija, medicinska oprema i mnoge druge primene koje zahtevaju napredne mogućnosti mikrokontrolera.

Mikrokontroler STM32F750VBH6 se bazira na ARM Cortex-M7 jezgru, koja pruža visoku računarsku snagu i podržava rad sa frekvencijom do 216 MHz. Takođe poseduje 1 MB Flash memorije, 340 KB RAM-a i brojne periferije, uključujući UART, SPI, I2C, USB, DMA kontrolere i mnoge druge.

STM32F750DK ploča dolazi sa različitim ugrađenim funkcionalnostima koje olakšavaju prototipiranje i testiranje. Na ploči se nalaze LED diode, tasteri, konektori za mikrofon, zvučnik i audio ulaz/izlaz, kao i razni senzori poput akcelerometra, žiroskopa i magnetometra.

Takođe poseduje LCD ekran dijagonale 4,3 inča i rezolucije 480x272 piksela, koji omogućava prikazivanje grafičkih korisničkih interfejsa i informacija. Tu je i Ethernet priključak koji omogućava komunikaciju putem mreže, kao i USB priključak za programiranje i serijsku komunikaciju sa računarom.

STM32F750DK ploča podržava razvojno okruženje STM32Cube, koje pruža bogat skup alata, biblioteka i primera kako bi se olakšao razvoj softvera za STM32 mikrokontrolere. Pored toga, ploča je kompatibilna sa raznim drugim razvojnim okruženjima kao što su Keil MDK, IAR Embedded Workbench i GCC.

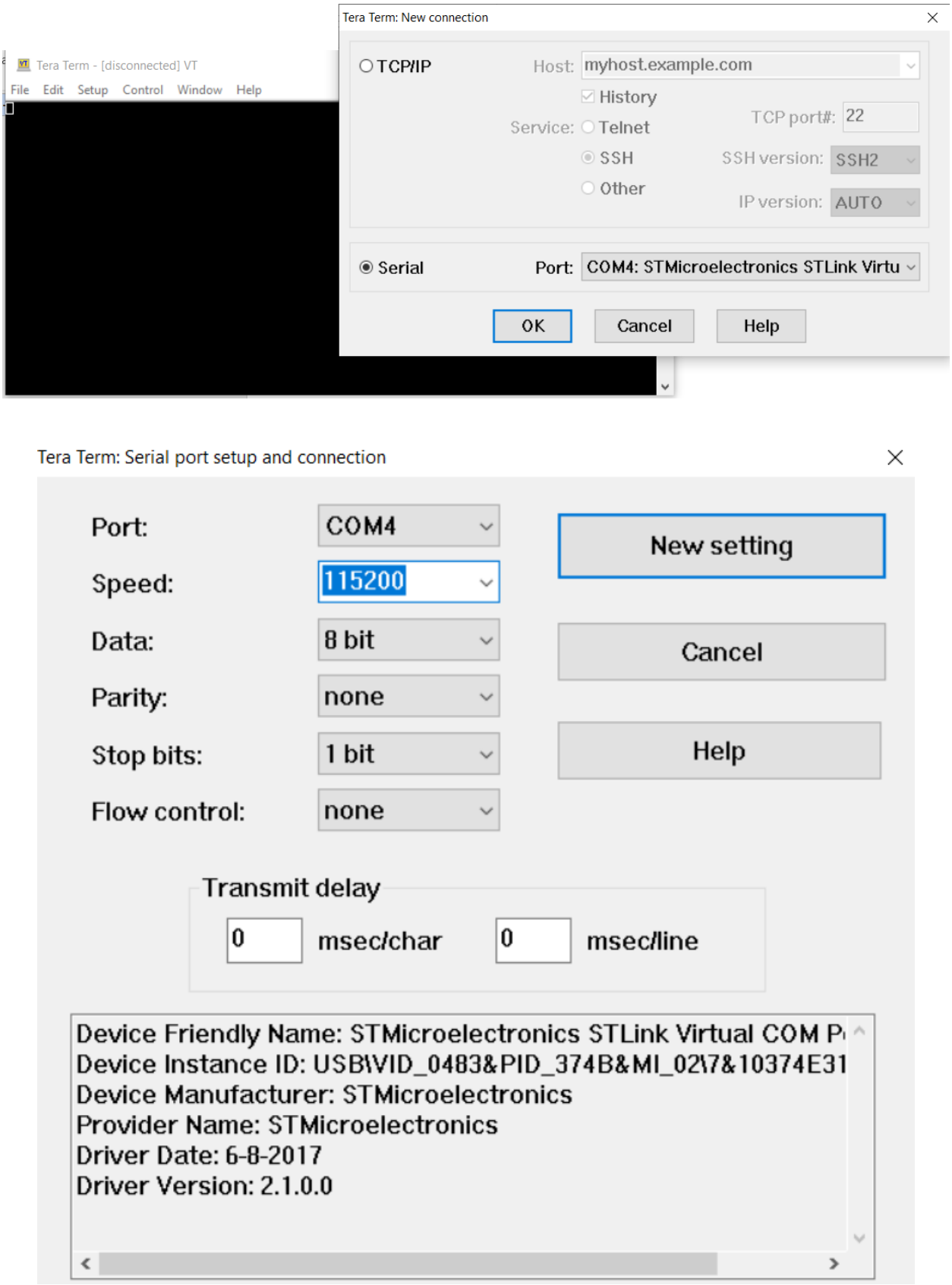
Ukratko, STM32F750DK je moćna razvojna ploča koja omogućava programerima da iskoriste napredne mogućnosti mikrokontrolera STM32F750VBH6. Sa različitim perifernim funkcionalnostima, senzorima i komunikacionim interfejsima, ova ploča je idealna za razvoj različitih aplikacija u industriji, IoT-u i drugim oblastima.

TERATERM

Teraterm je program za emulaciju terminala koji se često koristi za pristupanje i upravljanje udaljenim računarima putem različitih protokola kao što su Telnet, SSH, Serial, i druge. Ovaj program omogućava korisnicima da se povežu sa udaljenim uređajima, kao što su ruteri, prekidači ili serveri, i da izvršavaju različite operacije kao što su konfiguracija, dijagnostika ili prenos podataka.

Teraterm pruža korisnički interfejs koji simulira terminalni ekran i tastaturu, omogućavajući korisnicima da unose komande i da pregledaju izlaz sa udaljenog uređaja. Ovaj program je posebno popularan među sistem administratorima, mrežnim inženjerima i programerima koji često rade sa udaljenim računarima ili uređajima koji nemaju grafičko korisničko sučelje.

Teraterm podržava mnoge funkcionalnosti koje olakšavaju rad sa udaljenim uređajima, uključujući automatsko logovanje, prenos datoteka, podršku za različite kodne stranice, makro snimanje i reprodukciju, kao i mogućnost prilagođavanja izgleda i podešavanja. Takođe, Teraterm je besplatan i otvorenog koda, što ga čini popularnim izborom među korisnicima koji traže pouzdanu i fleksibilnu aplikaciju za emulaciju terminala.



Slika 2. Izgled i podesavanje terminala

Postavke se biraju na osnovu inicijalizacionih uart funkcija. U ovom slučaju imamo BAUD RATE od 115200, 8-bitne podatke, bez bita parnosti i jednom stop bitom.

OPIS ADXL345 MODULA

- **Osnovne informacije**

ADXL345 je digitalni akcelerometar koji služi za merenje g sila u opsegu do
±16g, formatiranog kao 16-to bitni dvoični komplement pri čemu modul meri pomeraj u 3 ose, x, y i z.

- **Rezolucija ADXL345**

Digitalni izlaz senzora je formatiran da bude u 10-bitnom dvojičnom komplementu. Razlog dvojičnog komplementa je merenje vrednosti na negativnim delovima (*x y z*) ose. Korišćena rezolucija merenja senzora je **±4g**. U zavisnostiod rezolucije AD-konvertora dobijamo:

$$Q = \frac{E_{\mathrm{FSR}}}{2^M},$$

$$E_{\mathrm{FSR}} = V_{\mathrm{RefHi}} - V_{\mathrm{RefLow}},$$

$$N = 2^M,$$

gde **E** predstavlja opseg vrednosti napona, **M** broj bitova rezolucije, a **V** (*RefHi*) i**V** (*RefLow*) viši i niži ekstrem napona koji se meri, dobijamo tkz. kvant rezolucije **Q** koji iznosi LSB.

U našem slučaju zbog visoke rezolucije rada senzora inkrement stanja od jednog LSB (bit najmanje težine) ekvivalentno je povećanju vrednosti za 7.8mg. Dobijenidigtalni podatak će biti multipliciran sa 7.8mg kako bi izmerili odgovrajuće ubrzanje (**g**).

Interfejs ADXL345

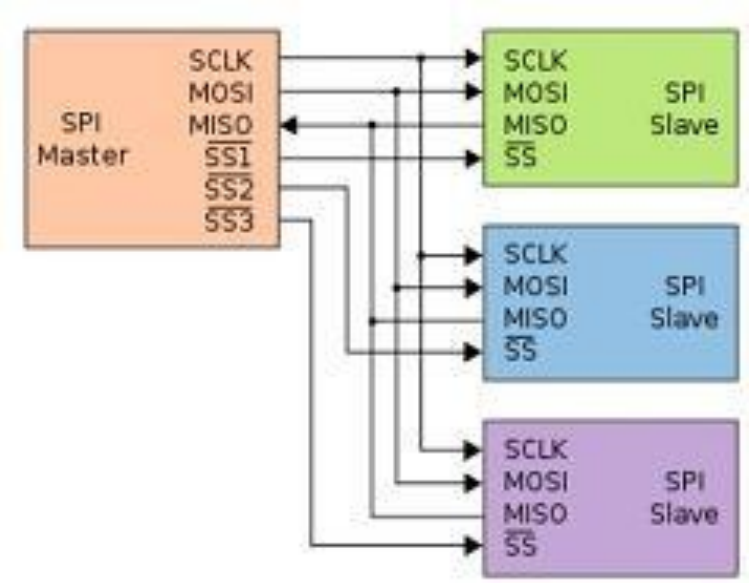
Što se tiče interfejsa između mikrokontrolera i bilo kog modula moguće je koristiti interfejse paralelnog i serijskog tipa. Paralelni interfejs podrazumeva da se ceo podatak šalje u ustom trenutku, a svaki bit ima svoj put kojim putuje do mikrokontrolera. Problem kod paralelnog interfejsa je taj što je hardverski dosta zahtevniji, u smislu zauzeća pinova. Naš akcelerometar je orijentisan prema serijskoj komunikaciji da bi se smanjio broj pinova koji su na modulu. Serijska komunikacija podrazumeva da se svi bitovi šalju preko jedne žice, sa pina na pin. Za serijsku komunikaciju se mogu koristiti asinhroni (UART) i sinhroni (SPI, I2C).

- **SPI**

SPI je jednostavan način serijskog sinhronog prenosa podataka koji se koristi za komunikaciju na kratkoj udaljenosti, primarno u embeded sistemima. Interfejs je nastao u kompaniji Motorola, prvenstveno za povezivanje mikrokontrolera sa periferijama (uloga je slična I2C standardu). Tipična primena uključuje senzore Secure Digital kartice i LCD monitore. SPI se naziva još ičetvorožična veza zbog četiri signala koji učestvuju u komunikaciji. SPI ostvaruje dvosmernu, dupleks vezu između jednoj glavnog uređaja, master, i jednog ili više perifernih uređaja, slejv. Master, koji je po pravilu mikrokontroler, generiše takt (SCLK). Frekvencija takta može iznositi između 1MHz i 70MHz. Master i slejv uređaji imaju osmobitni (mada su dozvoljene i druge veličine) pomerački registar. Komunikacija počinje kada glavni računar aktivira signal SS(slave select), to jest postavi logičku nulu na tu liniju (signal SS je na slici označena sa CS). Komunikacija je uvek dvosmerna, jedan bajt ide od mastera ka slejvu, drugi bajt istovremeni se prenosi od slejva ka masteru. Oba prenosa se obavljaju istovremenočak i kad neki od njih nema smisla. Na primer, ako mikrokontroler šalje veći broj bajtova ka periferiji, a periferija nema podataka koje bi slala mikro-kontroleru, smer od periferije ka mikrokontroleru prenosi podatke koji nemaju smisla, ali se ipak prenose.

Signali pomoću kojih se odvija komunikacija su sledeći:

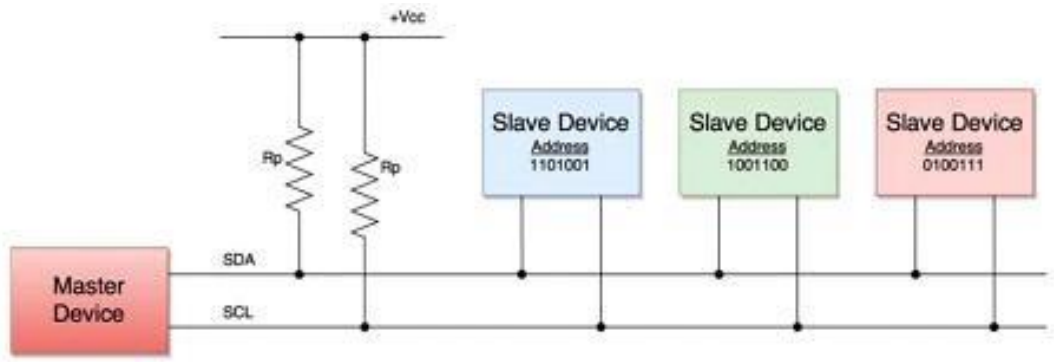
- **SCLK** – Serial Clock. Signal takta koji generiše master, a kojim se vrši sinhronizacija mastera i slejva. Smer je od mastera ka slejvu.
- **MOSI** – Master Output Slave Input. Ovim signalom se podatak prenosi od mastera ka slejvu.
- **MISO** – Master Input Slave Output. MISO signalom se vrši prenos podataka koje slejv želi da pošalje masteru.
- **SS** – Slave Select. Signal za odabir slejva sa kojim master želi da komunicira. Odabir slejva vrši se postavljanjem odgovarajućeg SS signala na 0.



Slika 3. Povezivanje uređaja pomoću SPI interfejsa

- **I2C**

I2C (*Inter Integrated Circuit*) magistrala je standardni dvosmerni interfejs koji koristi kontroler, poznat kao master, za interčipovsku komunikaciju sa podređenim uređajima, na malim odstojanjima. Podređeni uređaj, slejv, ne može inicirati prenos podataka. Svaki uređaj na I2C magistrali ima jedinstvenu adresu uređaja za razlikovanje ostalih uređaja koji se nalaze na istoj I2C magistrali. Mnogislejv uređaji zahtevaju neki vid konfiguracije koji definiše ponašanje uređaja. Ovo se obično vrši kada master pristupa internoj registarskoj mapi slejva. Uređaj može imati jedan ili više registara u kojima se podaci čuvaju, pišu ili čitaju i svaki od tih registara ima svoju adresu koja se naziva subadresa. Prenos informacija po I2C magistrali se vrši po bajtovima. Za prenos svakog bajta rezerviše se dodatno bitkoji se označava sa ACK/NACK. Ovo znači da se za prenos svakog bajta informacije koristi 9 bitova. Za adresiranje slejv uređaja rezervisano je 7 bitova veće težine što omogućava adresiranje 128 slejv uređaja. Bit najmanje težineadresnog bajta, sa ozakom R/W, master koristi da definiše smer transfera podatka koji slede. Kada master želi da upisuje podatke u slejv, R/W ima vrednost nula, a kade se podaci očitavaju iz slejv uređaja R/W ima vrednost jedan.



Slika 4. Povezivanje uređaja preko I2C interfejsa

Osnovna procedura upisa podataka od strane mastera ka slejvu je sledeća:

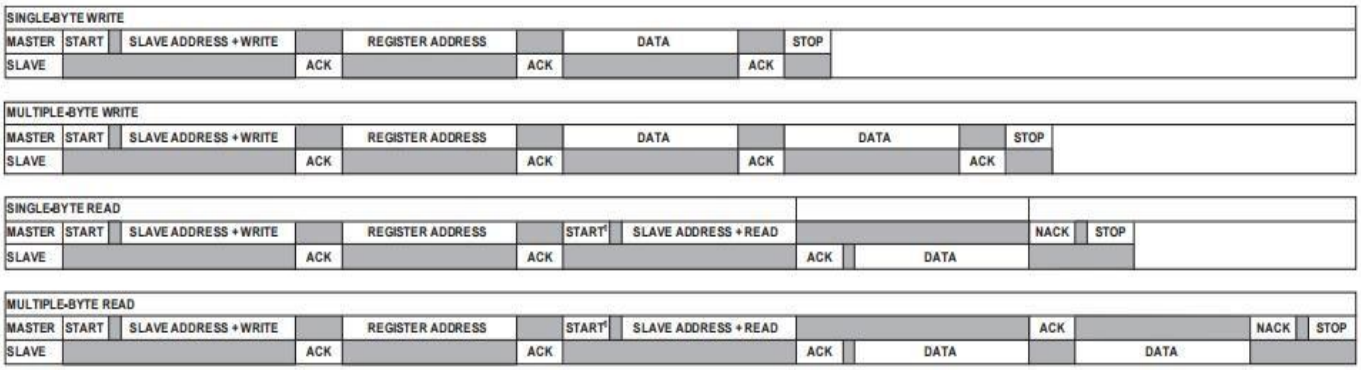
• Master emituje start uslov i šalje adresu slejv uređaja pri čemu definišeR/W=0.
• Adresirani slejv mora odgovoriti generisanjem ACK signala.
• Master emituje jedan ili više bajtova podataka koje želi upisati u slejv.Nakon svakog primljenog bajta slejv mora odgovoriti generisanjem ACK signala.
• Kada master ne želi više da šalje podatke emituje stop uslov.

Za slučaj da slejv uređaj zahteva dodatnu konfiguraciju protokol se moramodifikovati o čemu će biti reči kasnije.

Osnovna procedura čitanja podataka iz sleva je sledeća:

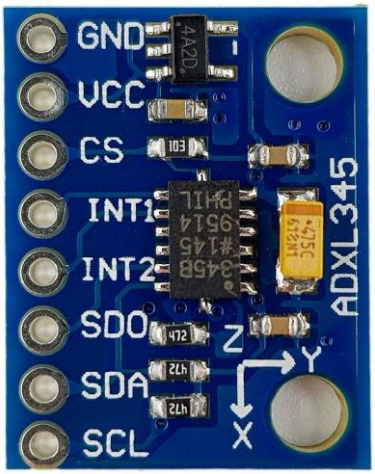
• Master emituje start uslov i šalje adresu slejv uređaja pri čemu definišeR/W=1.
• Adresirani slejv uređaj mora odgovoriti generisanjem ACK signala.
• Master generiše taktni signal na SCK liniji i očitava 8 bitova podataka kojeemituje slejv.
• Nakon očitanih 8 bitova svakog bajta master generiše ACK signal.
• Kada ne želi više da očitava podatke iz slejv uređaja master emituje stop uslov.

Za ovaj projekat je upravo i korišćen I2C interfejs da bi se ostvarilakomunikacija izmedju kontrolera 8051 i datog akcelerometra.



Slika 5. I2C ciklusi upisa i čitanja podataka

➢ Pin konfiguracija



Slika 6. ADXL345 akcelerometar

Pinovi od interesa za sistem su :

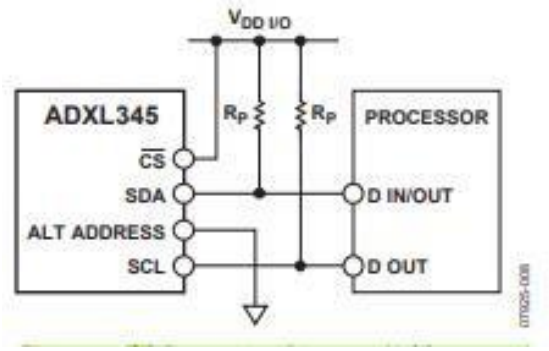
- VIN je napon napajanja čipa u opsegu od 2 do 3,6 V, u projektu povezan na sam pin za napajanje mikrokontrolera (na portu P0)
- GND je pin za dovođenje mase tj za izjednačenje mase modula i mikrokontrolera (povezan na masi na portu P0)
- SDA je port za prenos podataka preko I2C interfejsa (povezan na P0.0pin)
- SCL je port za prenos takta preko I2C interfejsa (povezan na P0.1 pin)

Pin No.	Mnemonic	Description
1	$V_{DD\ I/O}$	Digital Interface Supply Voltage.
2	GND	This pin must be connected to ground.
3	RESERVED	Reserved. This pin must be connected to V_S or left open.
4	GND	This pin must be connected to ground.
5	GND	This pin must be connected to ground.
6	V_S	Supply Voltage.
7	$\overline{CS}$	Chip Select.
8	INT1	Interrupt 1 Output.
9	INT2	Interrupt 2 Output.
10	NC	Not Internally Connected.
11	RESERVED	Reserved. This pin must be connected to ground or left open.
12	SDO/ALT ADDRESS	Serial Data Output (SPI 4-Wire)/Alternate I^2C Address Select (I^2C).
13	SDA/SDI/SDIO	Serial Data (I^2C)/Serial Data Input (SPI 4-Wire)/Serial Data Input and Output (SPI 3-Wire).
14	SCL/SCLK	Serial Communications Clock. SCL is the clock for I^2C, and SCLK is the clock for SPI.

Slika 7. Tabela sa brojevima i namenom pinova

SERIJSKA KOMUNIKACIJA

Čip select mora biti postavljen na visoki naponski nivo, tj. povezuje se na **Vdd**, da bi **I2C** bio u koristi. Za alternativnu 7-bitnu **I2C** adresu **0x53**, praćena 8. bitom R/W što za posledicu ima preslikavanje write adrese na **0xA6** a read na **0xA7**,potrebno je pin 12 (*ALT ADRESS*) povezati za masu. Moguće je povezati ga na **Vdd** medjutim, 7-bitna **I2C** adresa je sada **0x1D** praćena 8. R/W bitom.

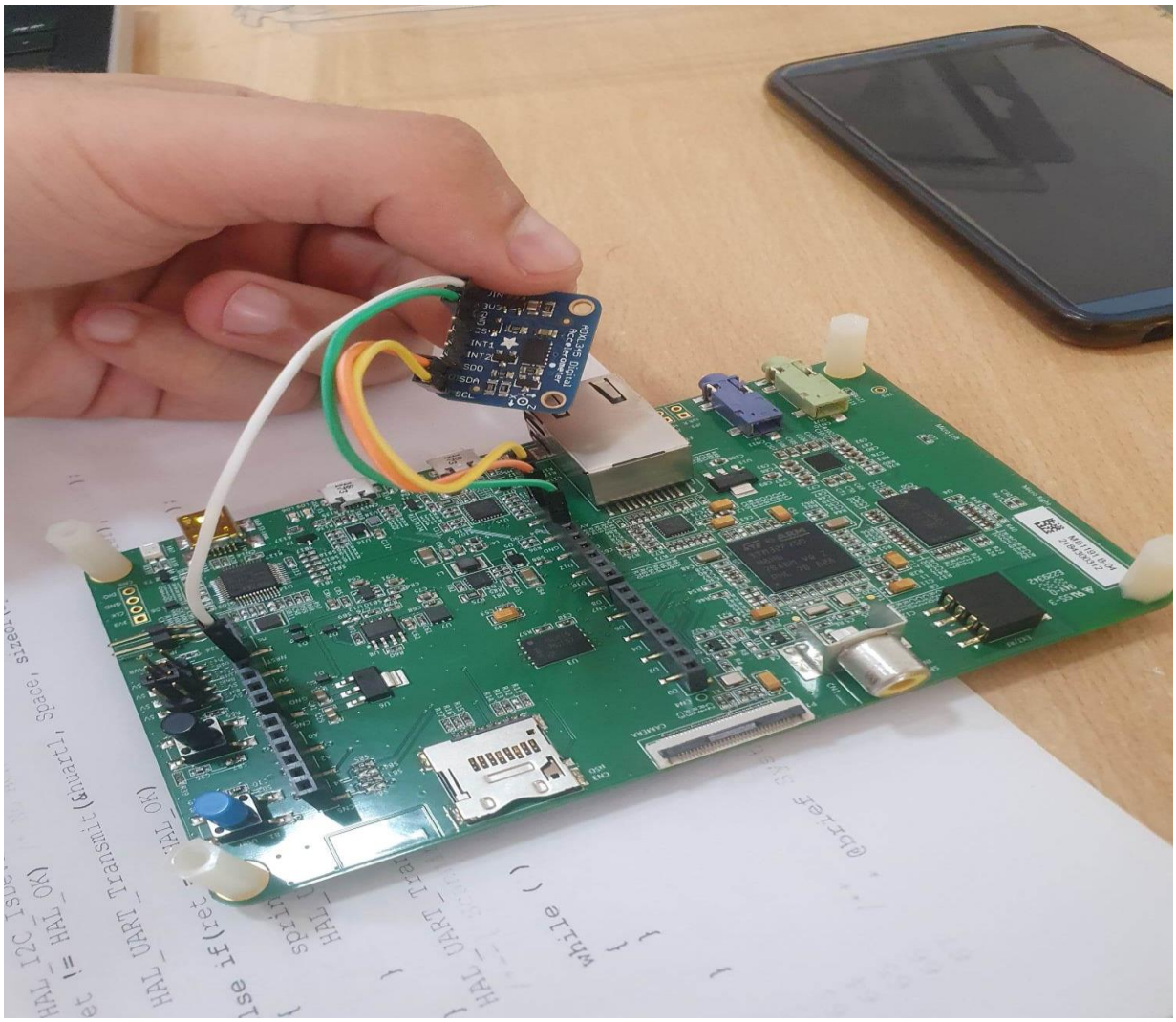


Slika 8. Povezivanje senzora i uC-a

Ne postoje interni pull-up ili pull-down otpornici za bilo koje neiskorišćene pinove. Posledica ovoga je floating stanje signala na ovim pinova, tj. stanje signala na pinovima je nedefinisano. Zbog tog razloga se može povezazti CS za **Vdd** ali nije neophodna. **ALT ADDRESS** povezti za masu ili **Vdd**.

POVEZIVANJE

Da bismo izvršili uspešno merenje prvo što treba da uradimo je da uspešno povežemo mikrokontroler i senzor. Povezivanje je konkretno opisano u opisu pinova od interesa za projekat. Ovde je prikazan izgled povezanog modula i mikrokontrolera.



Slika 9. Fizičko vezivanje mikrokontrolera sa senzorom

KOMUNIKACIJA SENZORA I MIKROKONTOLERA

Mikrokontroler se putem interfejsa (*I2C*) obraća senzoru posredstvom polja registra senzora. Polje registra se ponaša kao sprežno polje između samog senzora i mikrokontrolera. Bilo koje komande, promene stanja senzora ili inicijalizacije, koje se izdaju prema senzoru se obavljaju putem polja registra. Senzor je memorijski preslikan na mikrokontroleru, koji se od strane mikrokontrolera vidi kao skup registra (*registar map*).

0x2C	44	BW_RATE	R/$\overline{W}$	00001010	Data rate and power mode control
0x2D	45	POWER_CTL	R/$\overline{W}$	00000000	Power-saving features control
0x2E	46	INT_ENABLE	R/$\overline{W}$	00000000	Interrupt enable control
0x2F	47	INT_MAP	R/$\overline{W}$	00000000	Interrupt mapping control
0x30	48	INT_SOURCE	R	00000010	Source of interrupts
0x31	49	DATA_FORMAT	R/$\overline{W}$	00000000	Data format control
0x32	50	DATAX0	R	00000000	X-Axis Data 0
0x33	51	DATAX1	R	00000000	X-Axis Data 1
0x34	52	DATAY0	R	00000000	Y-Axis Data 0
0x35	53	DATAY1	R	00000000	Y-Axis Data 1
0x36	54	DATAZ0	R	00000000	Z-Axis Data 0
0x37	55	DATAZ1	R	00000000	Z-Axis Data 1
0x38	56	FIFO_CTL	R/$\overline{W}$	00000000	FIFO control
0x39	57	FIFO_STATUS	R	00000000	FIFO status

Slika 10. Registri senzora

REGISTRI

Korišćeni registri u projektu su:

- **Registar 0x2C—BW_RATE(Read/Write)**

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	LOW_POWER	Rate			

Slika 11. Sadržaj BW_RATE registra

Uloga ovog registra je kontrola režima rada čipa sa aspekta potrošnje, da li će sistem biti u **LOW_POWER** modu ili će raditi u normalnom režimu rada. Određuje i frekvenciju izabcivanja podataka kao i propusni opseg na komunikacionoj liniji.

- **LOW_POWER** bit postavljen na 0 postavlja čip u normalan režim rada, a ukoliko je setovan, radi u **LOW_POWER** modu pri čemu ima malo veće šumove.
- **Rate** bitovi određuju propusni opseg (*bandwith*) i frekvenciju izbacivanja podataka(*output data rate*). Za 100 HZ je sekvenca bitova (D3 D2 D1 D0) jednaka **1010** ili **0x0A** što smo i postavili po datasheet-u. Output rate mora biti usklađen sa propusnim opsegom i izabranim protokolom komunikacije kako bi se izbegla kolizija.
- U konkretnom slučaju za rad senzora iskoristili smo vrednost **0x1A**, gde viši nibl označava korišćenje **LOW_POWER** kontrolnog bita, a niži po datasheet-u.

Output Data Rate (Hz)	Bandwidth (Hz)	Rate Code	I_{DD} (µA)
400	200	1100	90
200	100	1011	60
100	50	1010	50
50	25	1001	45
25	12.5	1000	40
12.5	6.25	0111	34

Slika 12. Tabela sa pridruženim vrednostima i kodovima

- **Registar 0x2D—POWER_CTL(Read/Write)**

D7	D6	D5	D4	D3	D2	D1	D0
0	0	Link	AUTO_SLEEP	Measure	Sleep	Wakeup	

Slika 13: Sadržaj POWER_CTL registra

Registar ima ključnu ulogu u ovom projektu. **D3** bit, kao što je spomenuto, postavlja sistem u režim merenja. Pored ove funkcije, registar ima mogućnost postavljanja sistema u **sleep** mod ukoliko je detekovana neaktivnost čipa.

- **Link** bit - ovaj bit serijski povezuje funkcije aktivnosti i neaktivnosti čipa. Ukoliko je jednak nuli aktivne i neaktivne funkcije se dešavaju konkurentno, ako je bit jednak jedinici, uslovljava detekciju neaktivnosti čipa ukoliko je prethodono aktivnost bila detektovana.
- **AUTO_SLEEP -** bit jednak 1 postavlja čip u režim automatskog ‘spavanja’ ukoliko je detktovana neaktivnost.
- **MEASURE** - setovanje ovog bita je neophodno za sistem jer acc postavljamo u režim merenja. Ukoliko ima vrednost nula senzor se nalazi u **standby** režimu rada.
- **SLEEP** bit - postavlja senzor u **sleep** modu ako je jednak 1. Ako ima vrednost 0 senzor radi u normalnom režimu rada.
- **WAKEUP -** Ovi bitovi određuju frekvenciju čitanja podataka u **sleep** modu.

- **Registar 0x31—DATA FORMAT(Read/Write)**

D7	D6	D5	D4	D3	D2	D1	D0
SELF_TEST	SPI	INT_INVERT	0	FULL_RES	Justify	Range	

Slika 14. Sadržaj DATA FORMAT registra

Ovaj registar ima ulogu u formatiranju podataka koje dobijamo kao rezultatmerenja preko senzora.

Kontroliše format podataka koji se nalazi u registaru **0x32** preko registra
0x37. Bitovi od interesa za projekat su :

- **FULL_RESS** bit - za vrednost 1 daje podatke u 16-to bitnom režimu, dok vrednost 0 daje podatke u 10-to bitnom režimu, što smo i koristili u ovom projektu.
- **RANGE** bitovi - odredjuju punu skalu za merenje g sile. U ovom projektuse meri opseg od ±4g.

Setting		
D1	D0	*g* Range
0	0	±2 *g*
0	1	±4 *g*
1	0	±8 *g*
1	1	±16 *g*

Slika 15. Tabela sa pridruženim vrednostima i kodovima

- **JUSTIFY** bit se koristi za određivanje poravnanja podatala LSB ili MSB.Za 1 je levo poravnanje, a za 0 je desno, što je korišćeno u ovom projekt

- **Registar 0x38—FIFO_CTL(Read/Write)**

D7	D6	D5	D4	D3	D2	D1	D0
FIFO_MODE		Trigger	Samples				

Slika 16. Sadržaj FIFO_CTL registra

FIFO je komunikacioni uređaj koji služi za komunikaciju između modula pričemu oba modula mogu da ga otvaraju, a čitanje podataka se vrši u istom redosledu kao i upis. U FIFO režimu rada rezultati merenja senzora sa x, y i z osese skladište u FIFO. Kada je broj uzoraka merenja jednak broju sepcifiranom u **FIFO_CTL** registru **0x38, watermark interrupt** se setuje. FIFO nastavlja da akumulira uzorke merenja dok se ne napuni. **Watermark interrupt** nastavlja da se dešava sve dok je broj uzoraka u FIFO manji od broja uzoraka specifiranih u sample bitovima **FIFO_CTL** registara.

- **FIFO_MODE** bitovi - određuju režim rada FIFO-a prema sledećoj tabeli:

Setting			
D7	D6	Mode	Function
0	0	Bypass	FIFO is bypassed.
0	1	FIFO	FIFO collects up to 32 values and then stops collecting data, collecting new data only when FIFO is not full.
1	0	Stream	FIFO holds the last 32 data values. When FIFO is full, the oldest data is overwritten with newer data.
1	1	Trigger	When triggered by the trigger bit, FIFO holds the last data samples before the trigger event and then continues to collect data until full. New data is collected only when FIFO is not full.

Slika 17. Tabela sa pridruženim vrednostima i modovima

- **Sample bitovi** - funkcija ovih bitova zavisi od selektovanog FIFO moda. U FIFO modu (D7D6 = 01) ovi bitovi odredjuju uslov inicijalizacije**watermark interrupt**-a. Značenje ovih bitova zavisi

od režima rada FIFO-a

FIFO Mode	Samples Bits Function
Bypass	None.
FIFO	Specifies how many FIFO entries are needed to trigger a watermark interrupt.
Stream	Specifies how many FIFO entries are needed to trigger a watermark interrupt.
Trigger	Specifies how many FIFO samples are retained in the FIFO buffer before a trigger event.

Slika 18. Tabela sa pridruženim režimima rada i značenjem bitova

SOFTVER, ALGORITAM

```
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stdio.h"
#include <string.h>

#define adxl_address 0x53<<1

I2C_HandleTypeDef hi2c1;

UART_HandleTypeDef huart1;

/* USER CODE BEGIN PV */
uint8_t Buffer[25] = {0};
uint8_t Space[] = " - ";
uint8_t StartMSG[] = "Starting I2C Scanning: \r\n";
uint8_t EndMSG[] = "Done! \r\n\r\n";
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
static void MX_USART1_UART_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */

uint16_t x, y, z;
float xg, yg, zg;
void adxl_write(uint8_t reg, uint8_t value);
void adxl_read(uint8_t reg, uint8_t numberofbytes);
void adxl_init(void);
```

```
62 uint8_t data_rec[6];
63
64 void UART_TransmitString(const char* str)
65 {
66   HAL_UART_Transmit(&huart1, (uint8_t*)str, strlen(str), HAL_MAX_DELAY);
67 }
68
69 void UART_TransmitInt(int value)
70 {
71   char buffer[16];
72   sprintf(buffer, "%d\r\n", value);
73   UART_TransmitString(buffer);
74 } // ne koristi se
75
76 void UART_TransmitFloat(float value)
77 {
78       char buffer[16];
79       sprintf(buffer, "%f\r\n", value);
80       UART_TransmitString(buffer);
81 } //Mora da se u project/properties/c build/settings/mcu settings i tu da se stiklira podrska za float!!!
82
83 //funkcija za upis u registre
84 void adxl_write(uint8_t reg, uint8_t value) {
85     uint8_t data[2];
86     data[0] = reg;
87     data[1] = value;
88     HAL_I2C_Master_Transmit(&hi2c1, adxl_address, data, 2, 10);
89 }
90
91 //funkcija za citanje vrednosti
92 void adxl_read(uint8_t reg, uint8_t numberofbytes) {
93     HAL_I2C_Mem_Read(&hi2c1, adxl_address, reg, 1, data_rec, numberofbytes, 100);
94 }
95
96 //Inicijalizacija vrednosti
```

Slika 19. Prikaz definicije funckija i ukljucivanje potrebnih biblioteka

```
102     adxl_write(0x31, 0x01); //+- 4g
103 }
104
```

Glavni program

```
int main(void)
{
  /* USER CODE BEGIN 1 */
    uint8_t i = 0, ret;
  /* 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_I2C1_Init();
  MX_USART1_UART_Init();
  /* USER CODE BEGIN 2 */
  HAL_Delay(1000);

     /*-[ I2C Bus Scanning ]-*/
     HAL_UART_Transmit(&huart1, StartMSG, sizeof(StartMSG), 10000);
     for(i=1; i<128; i++)
     {
        ret = HAL_I2C_IsDeviceReady(&hi2c1, (uint16_t)(i<<1), 3, 5);
        if (ret != HAL_OK) /* No ACK Received At That Address */
        {
            HAL_UART_Transmit(&huart1, Space, sizeof(Space), 10000);
         }
        else if(ret == HAL_OK)
        {
            sprintf(Buffer, "0x%X", i);
            HAL_UART_Transmit(&huart1, Buffer, sizeof(Buffer), 10000);
        }
     }
```

Slika 20. Glavni program

```
    }
    HAL_UART_Transmit(&huart1, EndMSG, sizeof(EndMSG), 10000);

    adxl_init();

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    adxl_read(0x32, 6);
          x = (data_rec[1]<<8 | data_rec[0]);
          y = (data_rec[3]<<8 | data_rec[2]);
          z = (data_rec[5]<<8 | data_rec[4]);

          if(x > 32767) {
              int xn = x - 65535;
              xg = xn * .0078;
          }
          else {
              xg = x * .0078;
          }
          if(y > 32767) {
              int yn = y - 65535;
              yg = yn * .0078;
          }
          else {
              yg = y * .0078;
          }
          if(z > 32767) {
              int zn = z - 65535;
              zg = zn * .0078;
          }
          else {
              zg = z * .0078;
          }
          //xg = (float)x * .0078;
          //yg = (float)y * .0078;
          //zg = (float)z * .0078;

          UART_TransmitString("X: ");
          UART_TransmitFloat(xg);
```

Slika 21. Glavni program

```
            UART_TransmitString("Y: ");
            UART_TransmitFloat(yg);

            UART_TransmitString("Z: ");
            UART_TransmitFloat(zg);

            UART_TransmitString("\n\n");

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

/**
  * @brief System Clock Configuration
  * @retval None
  */



void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure LSE Drive Capability
  */
  HAL_PWR_EnableBkUpAccess();
  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = 25;
  RCC_OscInitStruct.PLL.PLLN = 400;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
```

Slika 22. Kraj glavnog programa i definicija funkcije SystemClock_Config

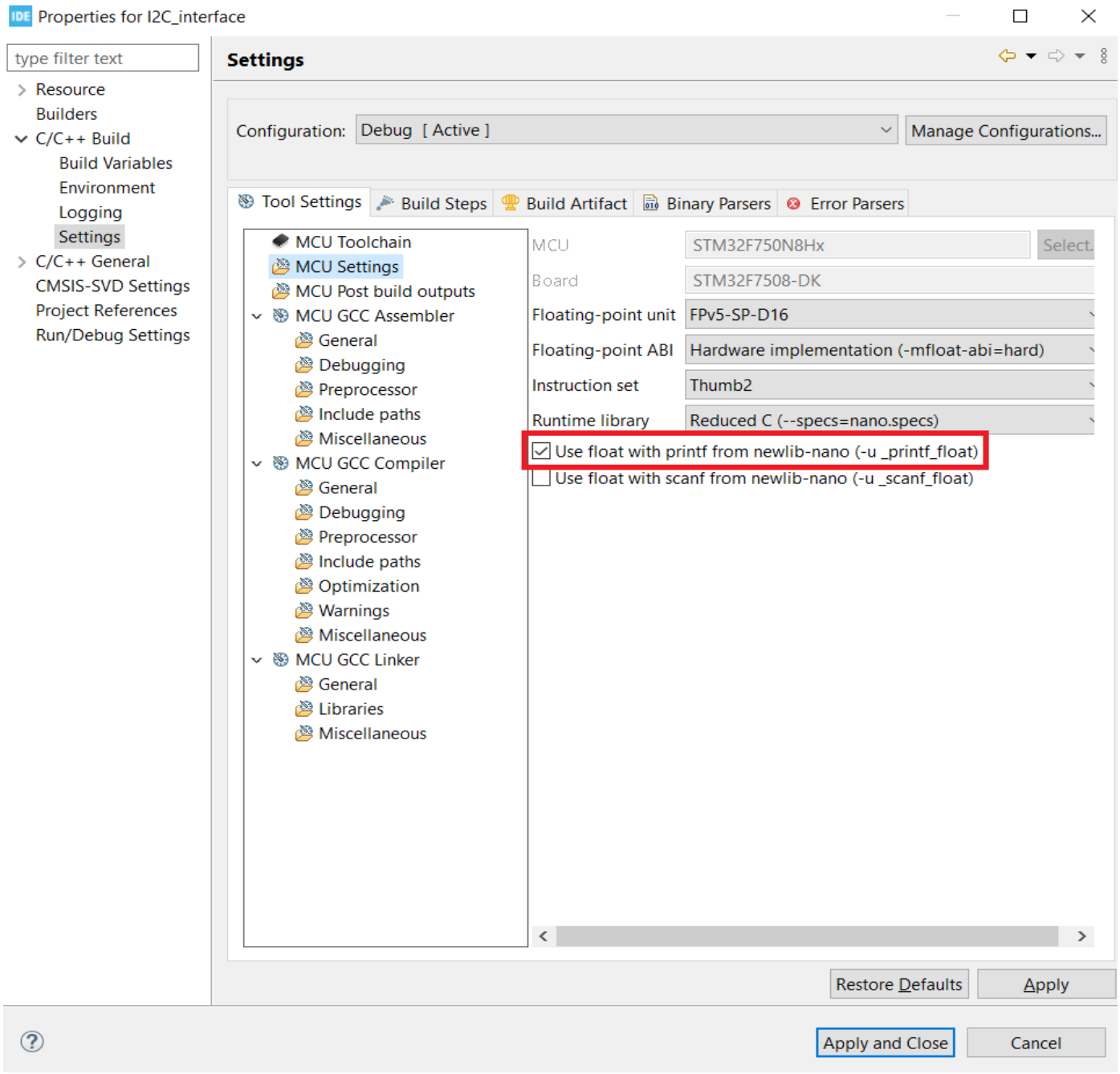
Ostatak koda

Ostatak koda dobija se preko alata **CubeMX.** Tako da nije od interesa da ga ovaj rad poseduje u celosti. Ovde su dati neki delovi, da uocimo kako to izgleda.

```
{
  Error_Handler();
}

static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOE_CLK_ENABLE();
  __HAL_RCC_GPIOG_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOJ_CLK_ENABLE();
  __HAL_RCC_GPIOI_CLK_ENABLE();
  __HAL_RCC_GPIOK_CLK_ENABLE();
  __HAL_RCC_GPIOF_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(OTG_FS_PowerSwitchOn_GPIO_Port, OTG_FS_PowerSwitchOn_Pin, GPIO_PIN_SET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOI, ARDUINO_D7_Pin|ARDUINO_D8_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LCD_BL_CTRL_GPIO_Port, LCD_BL_CTRL_Pin, GPIO_PIN_SET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LCD_DISP_GPIO_Port, LCD_DISP_Pin, GPIO_PIN_SET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(DCMI_PWR_EN_GPIO_Port, DCMI_PWR_EN_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOG, ARDUINO_D4_Pin|ARDUINO_D2_Pin|EXT_RST_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : LCD_B0_Pin */
  GPIO_InitStruct.Pin = LCD_B0_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
```

Slika 23. Primer koda dobijem alatom CubeMX

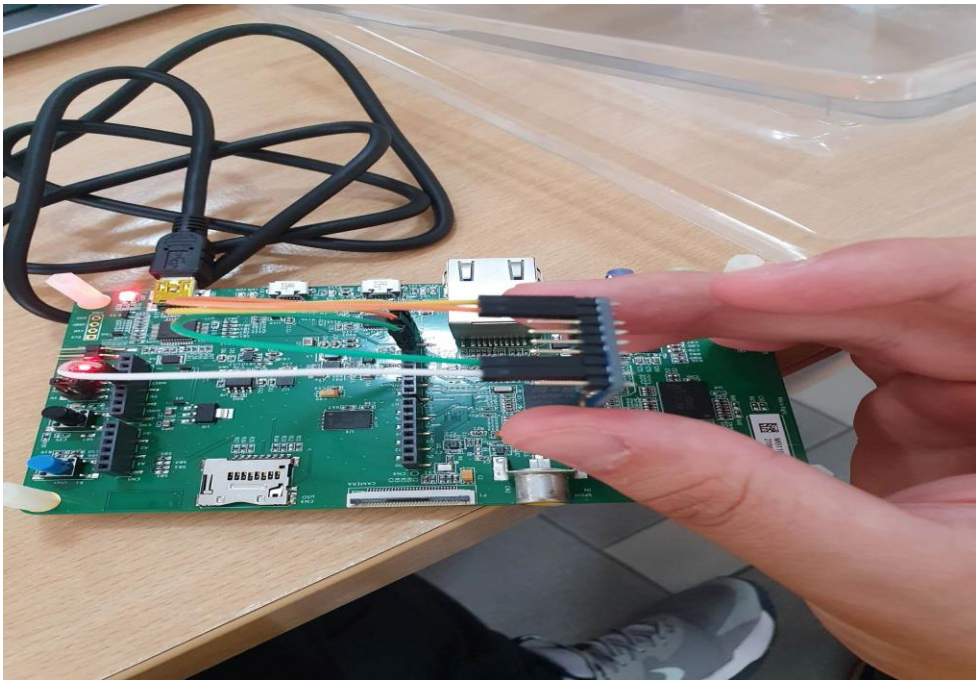
- Bitna napomena je korišćenje sprintf funkcije sa float vrednošću. Mora da se konfiguriše CubeIDE okruženje da prihvati taj format!!!
  Putanja podešavanja: **Project/Properties/C-C++ Build/Settings/MCU Settings/Check box use float with printf**



Slika 24. Izled prozora u alatu CubeIDE

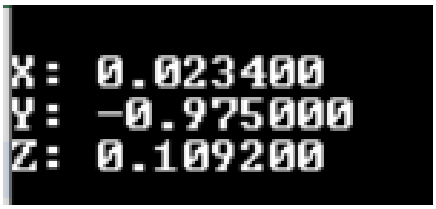
Prikaz rezultata

Pozicija 1:



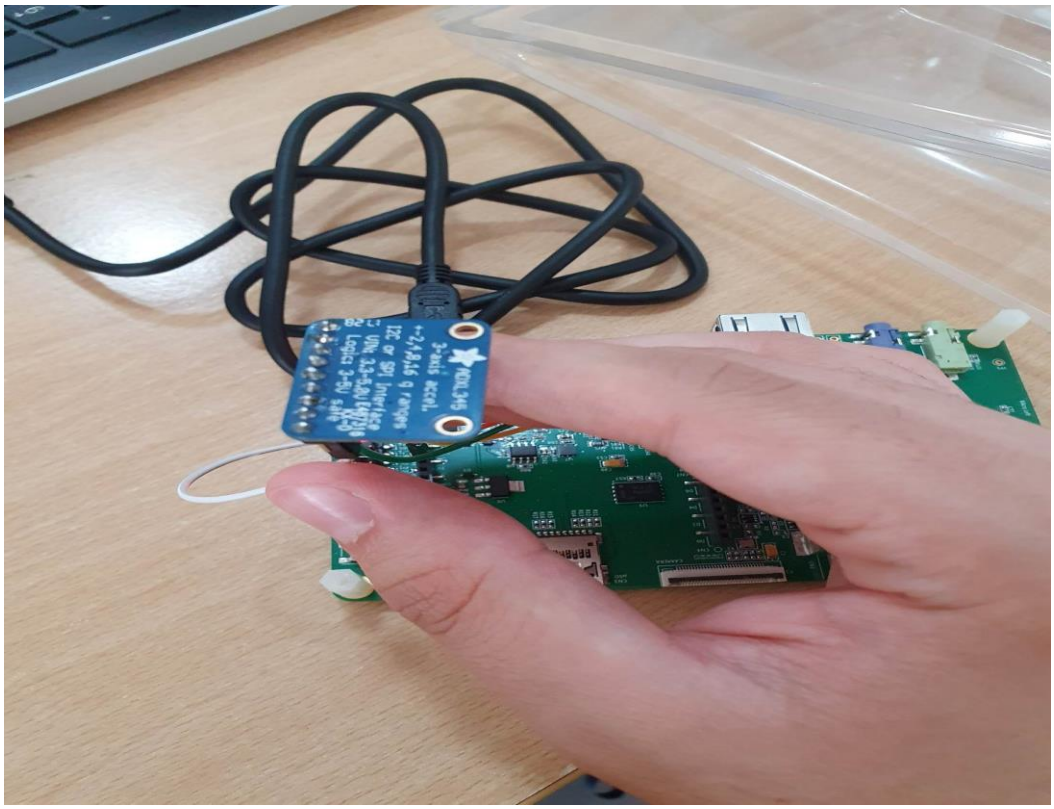
Slika 25. Pozicija 1

Rezultat 1:



Slika 26. Dobijene vrednosti

Pozicija 2:

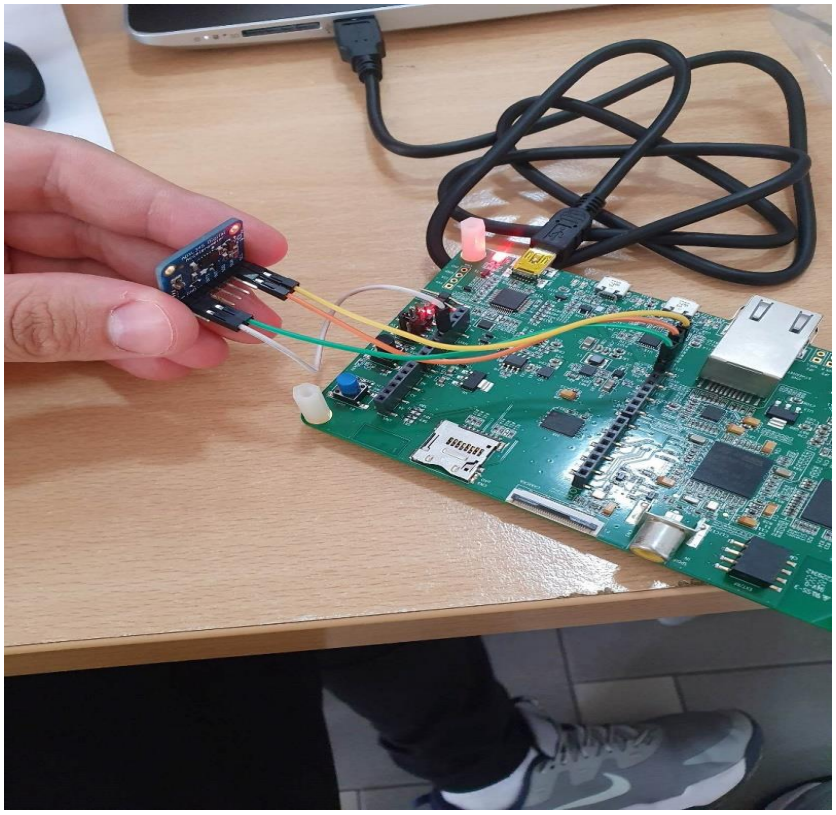


Slika 27. Pozicija 2

Rezultat 2:

Slika 28. Dobijene vrednosti

Pozicija 3:

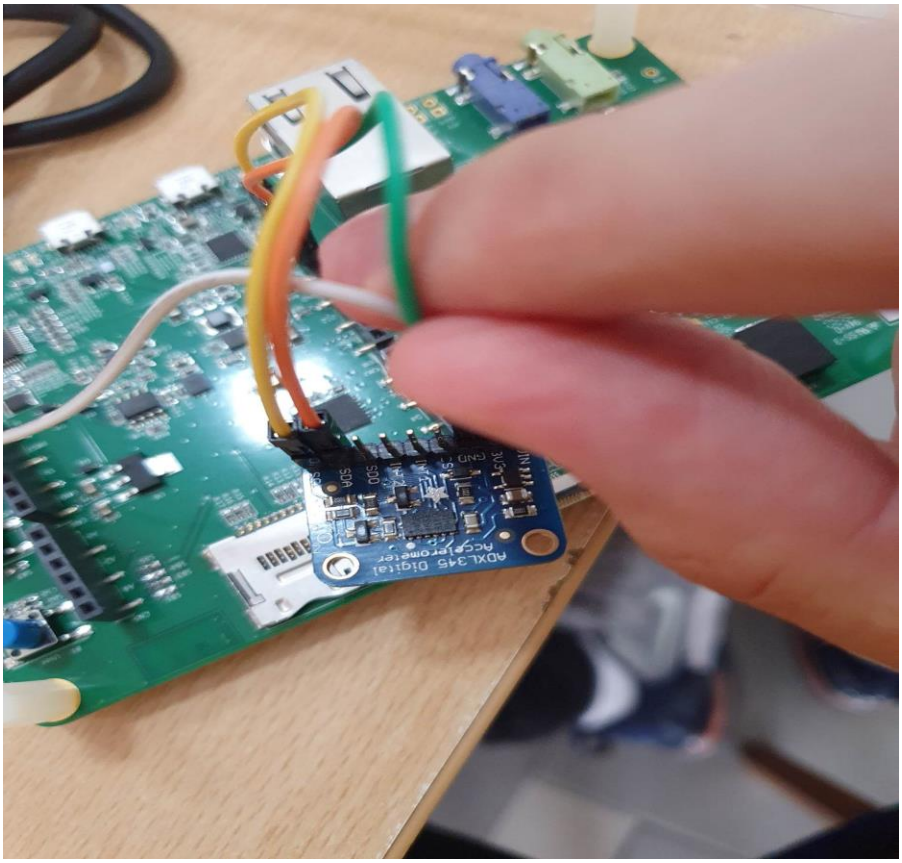


Slika 29. Pozicija 3

Rezultat 3:

Slika 30. Dobijene vrednosti

Pozicija 4:



Slika 31. Pozicija 4

Rezultat 4:

Slika 32. Dobijene vrednosti

FORMATIRANJE PODATAKA

Formatiranje izlaznih podataka na 3200Hz i 1600Hz, brzina izlaznih podataka se menja zavisno od moda operacije (full-resolution ili fixed 10-bit).

Kada koristimo 3200Hz ili 1600Hz izlazne brzine podataka u full-resolution ili +- 2g, 10-bit operacija, LSB od izlazne reči je uvek 0. Kada je podatak validan, on odgovara bitu D0 od DATAx0 registra, kao što je prikazano na slici. Kadapodatak nije validan i deo radi u 2g, 10-bit modu, LSB izlaznog podatka-reči je bit D6 od DATAx0 registra. U full-resolution operaciji, kada podatak nije validan, lokacija LSB se menja u skladu sa selektovanim izlaznim opsegom.

Za opseg od +-2g, LSB je bit D6 od DATAx0 registra; za +-4g bit je D5 od DATAx0 registra; za +-8g bit je D4 od DATAx0 registra i za +-16g, bit je D3 od DATAx0 registra. Korišćenje 3200Hz i 1600Hz izlazne brzine podataka za fixed 10-bit operacija u +-4g, +-8g, +-16g izlaznig opsega daje neki LSB koji je validan ikoji se menja u zavisnosti od primenjenog ubrzanja. Dakle, u ovim operacionim modovima bit D0 je uvek 0 kada izlazni podatak validan i bit D6 je uvek 0 kada izlazni podatak nije validan. Rad pri bilo kojoj brzini prenosa podataka od 800Hzili manje takođe pruža validan LSB u svim opsezima i modovima koji se menjaju uzavisnosti od primenjenog ubrzanja.

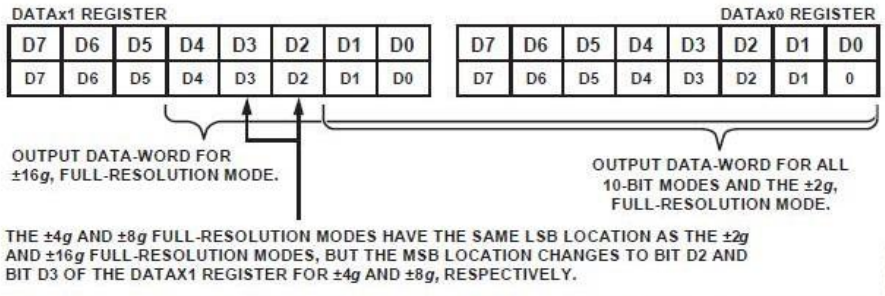


Figure 49. Data Formatting of Full-Resolution and ±2 g, 10-Bit Modes of Operation When Output Data Is Right Justified

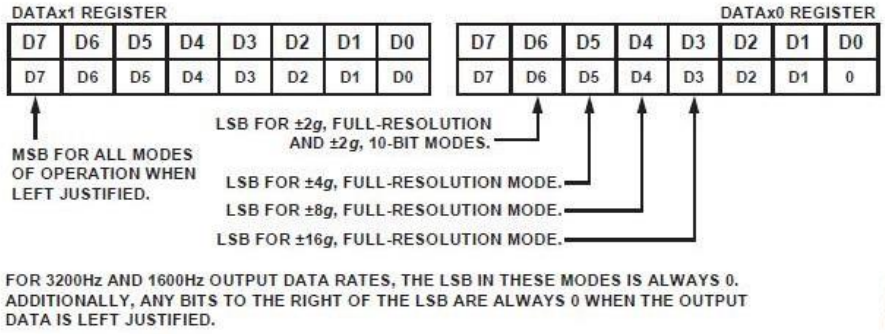


Figure 50. Data Formatting of Full-Resolution and ±2 g, 10-Bit Modes of Operation When Output Data Is Left Justified

Slika 33: Formatiranje podataka

Što se tiče same konverzije postupak je sledeći:

1. Kada se podaci pročitaju sa acc modula oni su u formu sirovih podataka (raw data)

2. Raw data je u opsegu od 0 do 65535, a oni preko pola (32767-65535) su negativni pa je potrebno da od njih oduzmemo 65535, a oni koji su u nižem opsegu su pozitivni i ne moraju da se modifikuju

3. Za **±4g** srednja vrednost LSB je 7.8mg, zato što je rezolucija 10-bitna modifikovani (ili nemodifikovani) sirovi podatak množimo sa $7.8/2^{10}$ (0.0078) i tako se dobija tačan podatak izražen u g.

```
adxl_read(0x32, 6);
      x = (data_rec[1]<<8 | data_rec[0]);
      y = (data_rec[3]<<8 | data_rec[2]);
      z = (data_rec[5]<<8 | data_rec[4]);

      if(x > 32767) {
          int xn = x - 65535;
          xg = xn * .0078;
      }
      else {
          xg = x * .0078;
      }
      if(y > 32767) {
          int yn = y - 65535;
          yg = yn * .0078;
      }
      else {
          yg = y * .0078;
      }
      if(z > 32767) {
          int zn = z - 65535;
          zg = zn * .0078;
      }
      else {
          zg = z * .0078;
      }
```

Slika 34. Konverzija podataka

LITERATURA

Potrebni linkovi:

1. STM32F750DK DATA SHEET NA LINKU https://www.st.com/resource/en/data_brief/stm32f7508-dk.pdf
2. ADXL345 – Accelerometer modul – http://adafruit/1231(*već je data šemarealizovanog modula*).