|  |  |
| --- | --- |
| tallinna tehnikaülikool | |
| Infotehnoloogia teaduskond | |
| Arvutisüsteemide instituut | |
|  | |
| Jörgen Vedom 123584 IASB | |
| Mitme Kiirendusanduriga kukkumise tuvastamise süsteemi arendus | |
| Bakalaureusetöö | |
| Juhendaja: | Mairo Leier |
|  | Doktorikraad |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Jörgen Vedom

16.04.2017

Annotatsioon

Lõputöö on kirjutatud keeles ning sisaldab teksti leheküljel, peatükki, joonist, tabelit.

Abstract

The thesis is in and contains pages of text, chapters, figures, tables.

Lühendite ja mõistete sõnastik

|  |  |
| --- | --- |
| ATI | TTÜ Arvutitehnika instituut |
| DPI | *Dots perinch*, punkti tolli kohta |
| SAI |  |
| UART  USART  I2C  LED  USB  CAN  CEC  SDIO  NA  CRC |  |
| IDE  API | integreeritud programmeerimiskeskkond  Application Programming Interface |

Sisukord

[2.1 Ülevaade sarnastest lahendustest 11](#_Toc482564593)

[2.2 Süsteemi nõuded 12](#_Toc482564594)

[2.3 Ühendused 13](#_Toc482564595)

[2.4 Tehnilised andmed 14](#_Toc482564596)

[2.5 Protokollid 19](#_Toc482564597)

[3.1 Arendusplaadi ühendamine arvutiga 24](#_Toc482564598)

[3.2 Arendusplaadi seadistamine 24](#_Toc482564599)

[STM32CubeMX seadistamine: 24](#_Toc482564600)

[4.1 Programmi töö 28](#_Toc482564601)

Jooniste loetelu

[Joonis 1 - Arendusplaadi PIN skeem 13](#_Toc482564611)

[Joonis 2 - Blokk diagramm 16](#_Toc482564612)

[Joonis 3 - UART 19](#_Toc482564613)

[Joonis 4 - RS485 22](#_Toc482564614)

[Joonis 5 - STM32CubeMX Projekti vaade 25](#_Toc482564615)

[Joonis 6 - Programmi töö 28](#_Toc482564616)

[Joonis 7 - Arendusplaadi initsialiseerimine 29](#_Toc482564617)

[Joonis 8 - Sensori initsialiseerimine 30](#_Toc482564618)

[Joonis 9 - Andmete päring 31](#_Toc482564619)

[Joonis 10 - Häirete kontroll 32](#_Toc482564620)

[Joonis 11 - Realterm Port seaded 34](#_Toc482564621)

[Joonis 12 - Realterm Displei seaded 34](#_Toc482564622)

[Joonis 13 - Paigal olek 35](#_Toc482564623)

[Joonis 14 - Liikumine 35](#_Toc482564624)

[Joonis 15- Tugevam põrutus 36](#_Toc482564625)

[Joonis 16 - Arendusplaadi pilt 39](#_Toc482564626)

[Joonis 17- Led tuluke 40](#_Toc482564627)

Tabelite loetelu

[Tabel 1 – Sensor 1 13](#_Toc482564602)

[Tabel 2 - Sensor 2 14](#_Toc482564603)

[Tabel 3 - Registrisse kirjutamine 20](#_Toc482564604)

[Tabel 4 - Kinnituse vastus 20](#_Toc482564605)

[Tabel 5 - Registrist lugemine 21](#_Toc482564606)

[Tabel 6 - Registrist õnnestunud lugemise vastus 21](#_Toc482564607)

[Tabel 7 - Registrist nurjunud lugemise vastus 21](#_Toc482564608)

[Tabel 8 - Saadetud baidid 22](#_Toc482564609)

[Tabel 9 - Vastuvõetud baidid 23](#_Toc482564610)

# Sissejuhatus

Töö eesmärgiks on arendada baas platvorm, mille abil oleks võimalik välja töötada algoritm kukkumise tuvastuseks. Selletõttu on arvestatud võimalusega, et saaks ühendada rohkem kui ühe kiirendusanduri.

Seadet hakatakse kasutama laevanduses, kui keegi kukub vette, siis teavitatakse peasüsteemi. Seade ise hakkab paiknema targas riietuses.

Sensorid võivad asuda üksteise suhtes erinevates kohtades. Näiteks üks sensor asub õlal ning teine sääre juures, sellisel puhul tuleb arvestada, et sensorite ühendamisel on kasutusel juhtmed. Juhtmed võivad mingil määral venida, sest riide kandja võib liigutada end, mille tagajärjel riie venib.

Andmete edastamisel tuleb kasutada veakindlat protokolli, mis toimib ka pikematel distantsidel.

Algoritmi välja töödeldes tuleb arvestada, et laev ei ole staatiline, vaid liigub ja kõigub. Seadme kandja võib laeval millegi vastu põrgata, mis ei tähendada vette kukkumist. Seadme kandjatel on erinevad pikkused ning kaalud.

Järgnevalt on välja toodud lõputöö käigus lahendatavad ülesanded:

* Prototüübi riistvara arendus
* Mikrokontrolleri seadistamine töötamaks mitme kiirendusanduriga
* Kiirendusandurite seadistamine vastavalt algoritmi poolt sisendandmetele esitatavatele nõetele

# Süsteemi ülesehitus

Süsteemi komponendid:

* STM32 -F446RE arendusplaat
* Kiirendusandur Bosch BNO055
* D-SUN „USB to TTL“

**STM32 F446RE:**

Arendusplaadi platvormiks sai valitud STM32 Nucleo F446RE[[1]](#endnote-1). STM32 Nucleo F446RE on paindlik platvorm mille peal prototüüpe arendada. Arendusplaadi valimisel oli oluline, et PIN-e oleks piisavalt, et saaks ühendada rohkem kui ühe sisend väljund seadme. STMicroelectronics pakub enda mikrokontrolleritele tarkvara STM32Cube, mis lihtsustab arendaja tööd. Arendaja määrab pordid, protokollid, PIN-id ja konfiguratsiooni seaded ning STM32Cube programm genereerib esialgse koodi.

**Bosch BNO055:**

Sensoriks sai valitud Bosch BNO055. BNO055 sensor on võimekas, sest tal on mitmeid erinevaid sensorid sisseehitatud (kiirendusandur, güroskoop, geomagnetiline sensor ning 32 bitine cortex M0+ mikrokontroller, mis jooksutab BSX3.0 FusionLib sensori tarkvara. Tarkvara pakub võimaluse sensoril töötada fusioon režiimil, mis tähendab, et absoluut orientatsiooni andmete arvutamine toimub kiirendusanduri, güroskoobi ja magnetomeetri abil. Antud töös, seda režiimi ei kasutata. Andmeedastuseks kasutab sensor I2C ja UART liidest.

**D-SUN „USB to TTL“:**

Pakub ünhedust USB ning UART liidese vahel. Hea riistvara, millega saab kontrollida, mis pakette saadetakse välja STM32 F446RE poolt või, mis pakette saadakse kätte Bosch BNO055 sensori poolt. Lisaks saab selle kaudu saata ka andmeid sensorile või ka arendusplaadile.

## Ülevaade sarnastest lahendustest

Kukkumis tuvastus seadmeid on mitmeid, isegi Philips on tulnud sellise tootega turule. Enamasti on need seadmed suunatud vanematele inimestele ning maismaale, mis ei ole seotud käsitletava probleemiga.

**Wearable sensors for Reliable fall detection** [[2]](#endnote-2) Tegu on Ad hoc tüüpi võrguga. Mõte on selles, et on üks baas seade, mille külge kõik teised seadmed ühilduvad. Sensor pannakse keha külge ning baas seadmele saadetakse sensori informatsioon. Ehk siis andmete töötlus toimub baas seadmes. Lahendusel on mõned puudused. Andmete töötlus toimub baas jaamas, see tähendab, et kui on palju sensoreid, siis andmete töötlus aeglustab kogu protsessi. Samuti võib esineda ka andmete kadu. Sensoriks on võetud kasutusele MICA2DOT 2 sensor [[3]](#endnote-3), mis kasutab operatsiooni süsteemiks TinyOS 1.0. Lisaks töötab see hästi vaid sisetingimustes.

**Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information** [[4]](#endnote-4) Kasutatakse nii kiirendusandurite kui ka güroskoopi. See võimaldab tuvastada kukkumised, mis olid tegelikult näiteks istumine, tõusmine. Kasutatakse mitut sensorit, mille abil saab tuvastada keha hoiakuid: Seismine, istumine, venitamine ning lamamine. Lahendus tundub väga hea, kuid see on suunatud kõvadele pindadele. Laeva peal lahendus ei toimiks.

**Philips GoSafe** [[5]](#endnote-5) Tegemist on seadmega, mis on suunatud nii sise- kui välistingimustesse. Asukoha tuvastamiseks on kasutusel Wifi, GPS, helialarm ja Intelligent Bread Grumbs (aegajalt salvestab asukoha, näiteks kasutades GPS-i).   
Tootel on mõned puudused. Tootel on igakuine maks ning see jaotub pakettidesse, kallimad paketid pakuvad paremaid lahendusi. Kui tegu on mitme inimesega, osutub see kulukaks. Kõne ei suunata hädaabisse, vaid reageerimis keskusele, kus uuritakse, mis inimesel täpsemalt juhtus ning seejärel tugiisik võtab ühendust hädaabiga.

**Shimmer sensing** [[6]](#endnote-6) Firma, mis keskendub kantavatele sensoritele. Sensorid on andmete kogumiseks ning analüüsiks. Platvormi kasutatakse prototüübi arenduseks. Kogutakse andmeid ning andmete põhjal töötatakse välja lahendus.Neilt tellitud töö jaguneb kolme faasi.

**Faas 1:**

* Idee
* Äri analüüs
* Nõuete analüüs ning spetsifikatsioon
* Süsteemi definitsioon

**Faas 2:**

* Aplikatsioon ja disain
* Prototüüp

**Faas 3:**

* Arendus ja testimine
* Kasutuselevõtt ning integratsioon
* Tootmine

Antud probleemile oleks võimalik neilt tellida lahendus, kuid see oleks kulukas ning aeganõudev juba ainuüksi andmekaabel maksab 39 €.

## Süsteemi nõuded

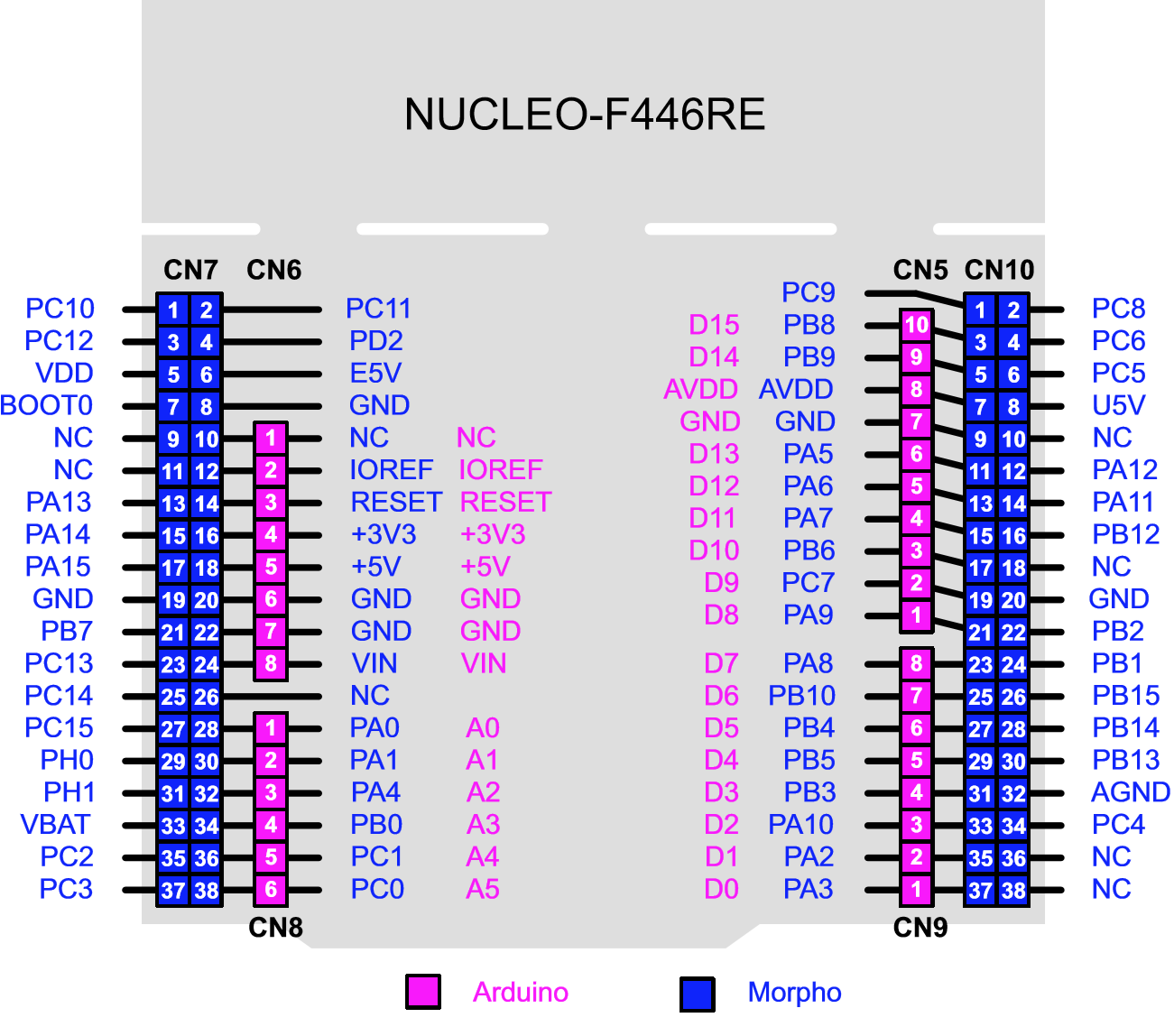
Protokollid, mille vahel oli võimalik valida, valikut kitsendas sensor Bosch BNO055[[7]](#endnote-7):

* I2C
* UART
* RS-485

Protokolli valikuks osutus UART ning RS-485. I2C protokoll langes välja, selletõttu, et lõppkokkuvõttes kasutatakse RS-485 protokolli, RS-485 tagab suurima veakindluse. Prototüüp arendatakse UART baasil ning lõpus minnakse üle RS-485 protokollile. Üleminek UART protokollilt RS-485 protokollile on lihtsustatud, sest RS-485 põhineb UART protokollil.

## Ühendused

**STM32 F446RE:**

****

Joonis - Arendusplaadi PIN skeem

Sensor 1 on ühendatud järgmistesse PIN-idesse, mida näitab tabel 1.

Tabel – Sensor 1

|  |  |
| --- | --- |
| **Sensori Port** | **Arendus plaadi Port** |
| PS0 | GND |
| PS1 | +3V3 |
| Vin | +3V3 |
| GND | GND |
| SDA | PA10 |
| SCL | PA9 |

Sensor 2 on ühendatud järgmistesse PIN-idesse, mida näitab tabel 2.

Tabel - Sensor 2

|  |  |
| --- | --- |
| **Sensori Port** | **Arendus plaadi Port** |
| PS0 | GND |
| PS1 | +3V3 |
| Vin | +3V3 |
| GND | GND |
| SDA | PC5 |
| SCL | PB10 |

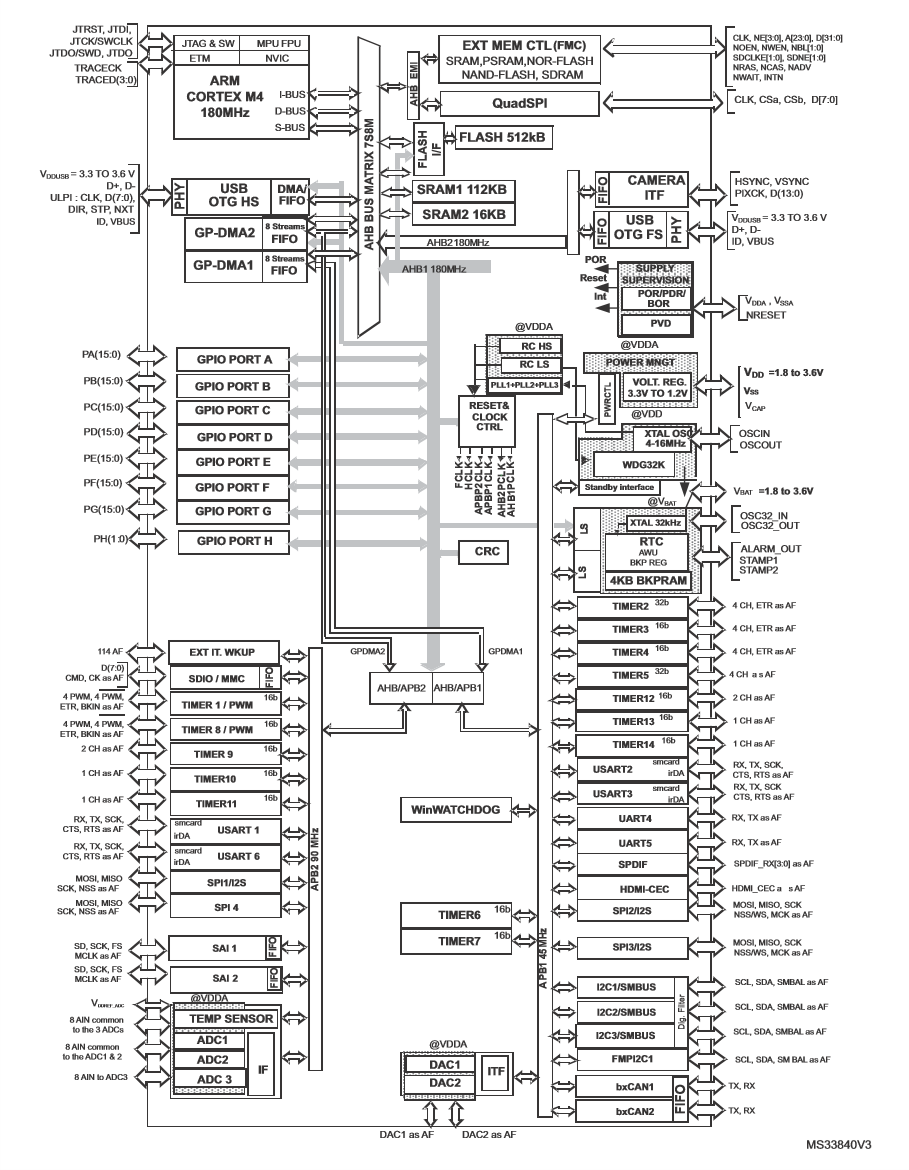
Juhul, kui tuleb, siis Bluetooth läheb siia, USART6

## Tehnilised andmed

**STM32-F446RE arendusplaadi tähtsamad tehnilised andmed**:

* Protsessor
  + 180 Mhz Cortex-M4 [[8]](#endnote-8)
* Mälu
  + 512 kB Välk mälu
  + 128 KB SRAM
* Toide
  + 1.7 V kuni 3.6 V aplikatsioon ja sisend väljund seadmed.
* Debug mode
* Kuni 114 Sisend Väljund porti sekkumise võimekusega
* Kuni 20 kommunikatsiooni liidest
  + SPDIF-Rx
  + Kuni 4 x
  + Kuni 4 USART/2 UART
  + Kuni 4 SPI
  + Kuni 2 SAI
  + Kuni 2 CAN
  + SDIO liides
* Lisa ühendused
  + USB 2.0
  + USB 2.0
* CRC arvutus blokk
* Tavalised kasutusalad
  + Autondus
  + Meditsiiniseadmed
  + Tööstus seadmed: PLC, inverteerid, kaitselülitid

Programmi koodis tuleb anda ette port, mille külge on PIN ühendatud. Seetõttu on blokk diagramm hea joonis kuhu pöörduda, et näha, mis PIN on pordi külge ühendatud. Seda näitab joonis 2.

****

Joonis - Blokk diagramm

**BNO055 tähtsamad tehnilised andmed** [[9]](#endnote-9):

* Sensori omadused
  + Väljastab fusioon sensori andmeid
  + Toite haldus
    - Tava režiim
    - Madala toite režiim
    - Seisaku režiim
  + Pinge
    - Vdd pinge vahemik 2.4 – 3.6 V
  + Digitaalsed liidesed
    - HID-I2C
    - I2C
    - UART
  + Töötemperatuur
    - -40 - +85 °C
* Kiirendusanduri omadused
  + Programmeeritav funktsionaalsus
    - Kiirendus vahemik
      * ± 2 g
      * ± 4 g
      * ± 8 g
      * ± 16 g
    - Töö režiimid
      * Normaal
      * Seisak
      * Madal tarne
      * Oote
      * Sügav seisak
* Güroskoobi omadused
  + Programmeeritav funktsionaalsus
    - Ranges from
      * ± 125 kuni ± 2000 °/ s
    - Töö režiimid
      * Normaal
      * Kiire tööle minek
      * Sügav seisak
      * Seisak
      * Täiustatud energiasääst
* Magnetomeetri omadused
  + Flexible functionality
    - Magnetvälja vahemik
      * ± 1300 µT (x-, y-telg)
      * ± 2500 µT (z-telg)
    - Magnetvälja lahutusvõime
      * ~ 0.3 µT
    - Töö režiimid
      * Low tarne
      * Regulaarne
      * Täiustatud regulaarne
      * Kõrge täpsus
    - Tarne režiimid
      * Normaal
      * Uinaku
      * Seisak
      * Jõu
* Tavalised kasutusalad
  + Navigatsioon
  + Robootika
  + Heaolu
  + Augmenteeritud reaalsus
  + Konteksti teadlikus
  + Tahvel – ja sülearvutid

## Protokollid

Arendusplaadi ja sensori vaheline andmevahetus toimub läbi UART protokolli. Järgnevalt on toodud selgitus, kuidas protokoll töötab.

**UART[[10]](#endnote-10)**

Laialdaselt kasutatud protokoll, mida kasutatakse andmete saatmiseks. UART protokoll on asünkroonne, mis tähendab, et puudub kell signaal, vaid andmetel on algus ja lõpp sõnumil. Oluline on, et mõlemad seadmed töötaksid samal boodikiirusel. UART protokolli andmevahetuse kaugus on ~2m.

Arendusplaadil kasutatavad signaalid:

* RXD – võtab vastu andmeid, ühendatud saatja TXD signaali külge
* TXD – saadab andmeid, ühendatud vastuvõtja RXD signaali külge

Andmete formaat on ehitatud järgmiselt: **start – andmed – paarsus – stopp**

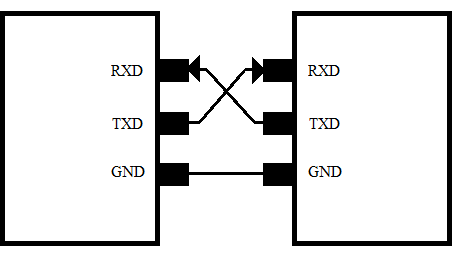
**Start** – alati on ainult 1 start bitt.

**Andmed –** andmed varieeruvad, 5-9 bitti. Tavaliselt 8 bitti.

**Paarsus** – sellel on mitmeid seadeid: puudub, paaris, paaritu, mark ja tühik

**Stopp –** tavaliselt on stopp bitte kas 1 või 2.

UART Ühendus on näidatud joonisel 3.



Joonis - UART

Bosch BNO055 toetab UART protokolli järgmiste seadetega: 115200 bps (bitti sekundis), 8N1 (8 andme bitti, 0 paarsus bitti, 1 stopp bitt). Maksimaalne pikkus lugemiseks ja kirjutamiseks on 128 Baiti. Järgnevalt on kirjeldatud lugemise ja kirjutamis struktuur. Registrisse kirjutamine on näidatud tabelis 3. Kinnituse vastus on näidatud tabelis 4. Registrist lugemine on näidatud tabelis 5. Registrist õnnestunud lugemise vastus on näidatud tabelis 6. Registrist nurjunud lugemise vastus on näidatud tabelis 7.

Tabel - Registrisse kirjutamine

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Bait 1** | **Bait 2** | **Bait 3** | **Bait 4** | **Bait 5** | **…** | **Bait (n + 4)** |
| Algus Bait | Kirjutamine | Registri aadress | Pikkus | Andmed 1 | … | Andmed n |
| 0xAA | 0x00 | <..> | <..> | <..> | … | <..> |

Tabel - Kinnituse vastus

|  |  |
| --- | --- |
| **Bait 1** | **Bait 2** |
| Vastuse algus bait | Staatus |
| 0xEE | 0x01: Kirjutamine õnnestus  0x03: Kirjutamine nurjus  0x04: Vale Registri aadress  0x05: Registri kirjutamine väljalülitud  0x06: Vale algus bait  0x07: Siini ülekoormuse viga  0x08: Maksimaalse pikkuse viga  0x09: Minimaalse pikkuse viga  0x0A:Vastuvõetud märgi taimaut |

Tabel - Registrist lugemine

|  |  |  |  |
| --- | --- | --- | --- |
| **Bait 1** | **Bait 2** | **Bait 3** | **Bait 4** |
| Algus Bait | Kirjutamine | Registri aadress | Pikkus |
| 0xAA | 0x01 | <..> | <..> |

Tabel - Registrist õnnestunud lugemise vastus

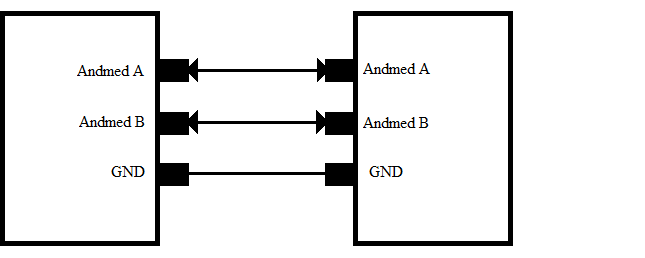
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bait 1** | **Bait 2** | **Bait 3** | **…** | **Bait (n + 2)** |
| Algus Bait | Pikkus | Andmed 1 | … | Andmed n |
| 0xBB | <..> | <..> | … | <..> |

Tabel - Registrist nurjunud lugemise vastus

|  |  |
| --- | --- |
| **Bait 1** | **Bait 2** |
| Vastuse algus bait | Staatus |
| 0xEE | 0x02: Lugemine nurjus  0x04: Vale Registri aadress  0x05: Registri kirjutamine väljalülitud  0x06: Vale algus bait  0x07: Siini ülekoormuse viga  0x08: Maksimaalse pikkuse viga  0x09: Minimaalse pikkuse viga  0x0A:Vastuvõetud märgi taimaut |

**RS485[[11]](#endnote-11)**

RS485 Kasutab sama loogikat kui UART, kuid kasutab erinevat liinidraiverit. Liinidraiver konverteerib ühesuunalise UART signaali kahesuunaliseks. See annab kaks andme liini. Andmeliin A ja andmeliin B. Kahesuunalise signaali eeliseks on see, et süsteemil on väiksem müra ning lubab pikemaid ühendusi. RS485 protokolli andmevahetuse kaugus ~1200m. RS485 ühendus on näidatud joonisel 4.

****

Joonis - RS485

**Bosch BNO055 register:[[12]](#endnote-12)**

Register on jagatud kaheks loogiliseks leheks. Leht 1 omab endas sensori põhist konfiguratsooni seadeid. Leht 0 omab kõiki muid konfiguratsiooni parameetreid ning väljund andmeid.

Sisselülitamisel on registri leht 0 valitud. PAGE\_ID registrit saab kasutada, et tuvastada, mis leht parasjagu valitud on, samuti saab selle kaudu ka registri lehte muuta.

**Näide kuidas lugeda temperatuuri:**

Temperatuuri andmed asuvad registris 34, mis ise asub 0 registri lehel. Kokku tuleb saata selleks saata 4 baiti, millest 1 bait - algus bait, 2 bait – lugemis bait, 3 – loetava registri aadress, 4 – pikkus (kui palju andmeid saata). Kui sensor saab õiged andmed kätte, siis vastab ta 3 baidiga, millest 1 bait – vastuse päise, 2 – bait pikkus (kui palju andmeid saadetakse), 3 – andmed. Saadetud baitide näidised asuvad tabelis 8 ning vastuvõetud baidid asuvad tabelis 9.

Tabel - Saadetud baidid

|  |  |  |  |
| --- | --- | --- | --- |
| Bait 1 | Bait 2 | Bait 3 | Bait 4 |
| 0xAA | 0x01 | 0x34 | 0x01 |

Tabel - Vastuvõetud baidid

|  |  |  |
| --- | --- | --- |
| Bait 1 | Bait 2 | Bait 3 |
| 0xBB | 0x01 | 0x20 |

# Tarkvara

## Arendusplaadi ühendamine arvutiga

Vajaminevad driverid ja programmid:

* Driverid
  + ST-LINK/V2-1
* Programmid
  + Java
  + MDK-ARM V5
  + STM32CubeMX
  + Realterm

**Tüüpiline konfiguratsioon:**

* USB cable type A to mini-B
* Arvuti Windows XP, 7, 8

**Developer Toolchain:**

* IAR EWARM
* Keil MDK-ARM
* GCC-based IDE

## Arendusplaadi seadistamine

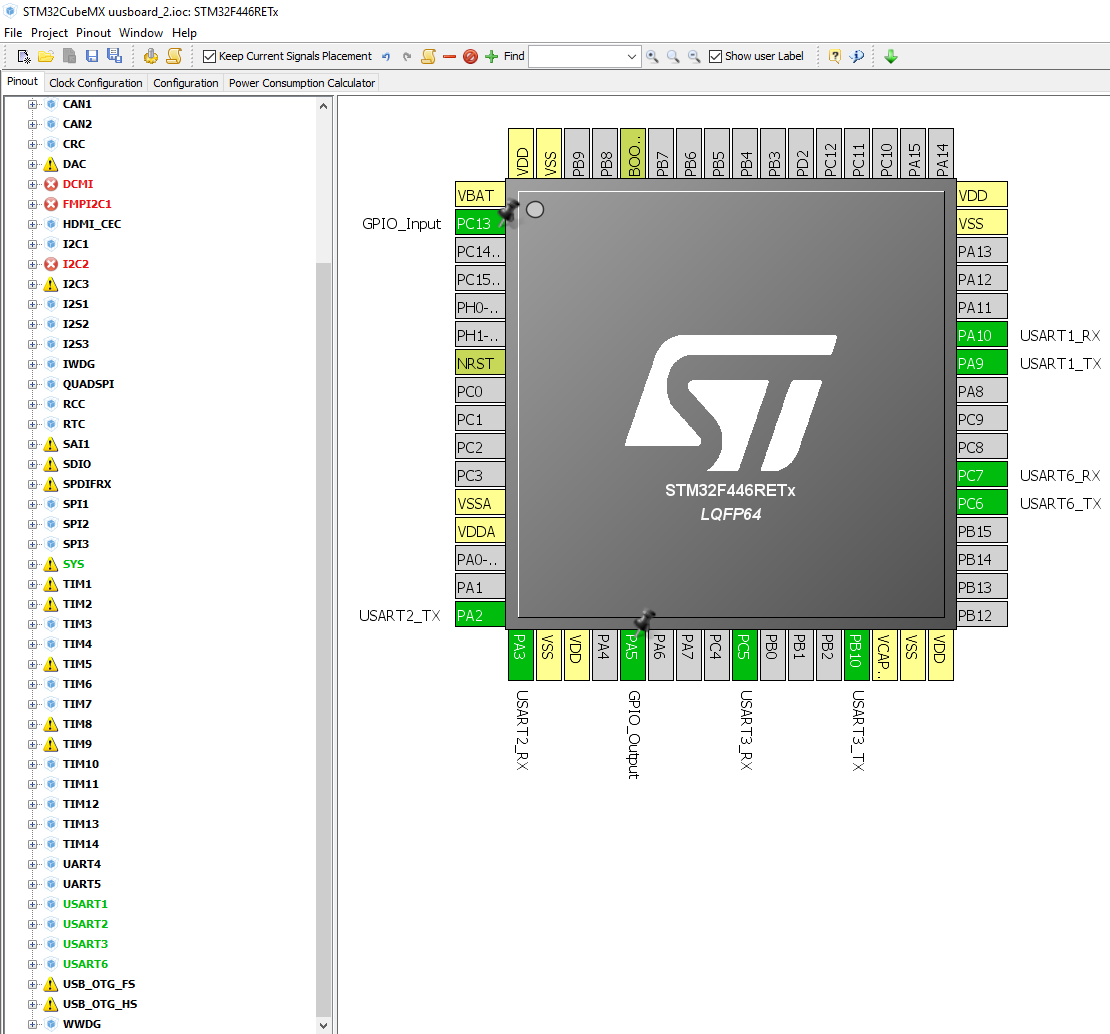
STMicroelectronics on loonud aplikatsiooni STM32CubeMX. Rakenduse genereerib initsialiseerimis koodi kasutades selleks graafilist liidest.

### STM32CubeMX seadistamine:

STM32CubeMX programmis olulised projekti seaded. IDE valik – MDK-ARM V5. Koodi genereerimise alam menüüst tuleb vaadata, et kasutaja kood jääks alles igakord kui programm genereerib uue koodi.

**Arendusplaadi seaded:**

Aktiveerida tuleb UART4, USART1, USART2. (UART4 eemaldada, üldiselt tegele sellega, kui kood on valmis.)



Joonis - STM32CubeMX Projekti vaade

#### USART1 Seadistus:

Vajalikud seaded, et seadistada USART1. USART1 kaudu toimub arendusplaadi ning esimese sensori vahel andmete saatmine.

**Seaded:**

* Baud rate – 115200 Bits/s
* Word length – 8 Bits (including Parity)
* Parity – None
* Stop Bits – 1
* Global interrupt aktiveerida

#### USART3 Seadistus:

Vajalikud seaded, et seadistada USART3. USART3 kaudu toimub arendusplaadi ning teise sensori vahel andmete saatmine.

**Seaded:**

* Baud rate – 115200 Bits/s
* Word length – 8 Bits (including Parity)
* Parity – None
* Stop Bits – 1
* Global interrupt aktiveerida

#### USART2 Seadistus:

Arendusplaat on võimeline saatma läbi USB ühenduse UART liidesega andmeid. USART2 kasutatakse, et kuvada testimise otstarbel andmeid.

**Seaded:**

* Baud rate – 115200 Bits/s
* Word length – 8 Bits (including Parity)
* Parity – None
* Stop Bits – 1
* Global interrupt aktiveerida

**USART6 juhul, kui Bluetooth tuleb ka.**

# Programm

Programmikood on kirjutatud C keeles. C keel on laialdaselt kasutatud süsteemne programmeerimis keel. Keel loodi aastatel 1977 – 1979 paralleelselt operatsiooni süsteemi UNIX-iga. Esialgu kasutati keelt UNIX süsteemi jaoks, kuid tänapäeval on see kujunenud üheks peamiseks arvutitööstuses kasutatavaks programmeerimis keeleks. C keele eelisteks on tema kiirus ning vähene mälukasutus. [[13]](#endnote-13)

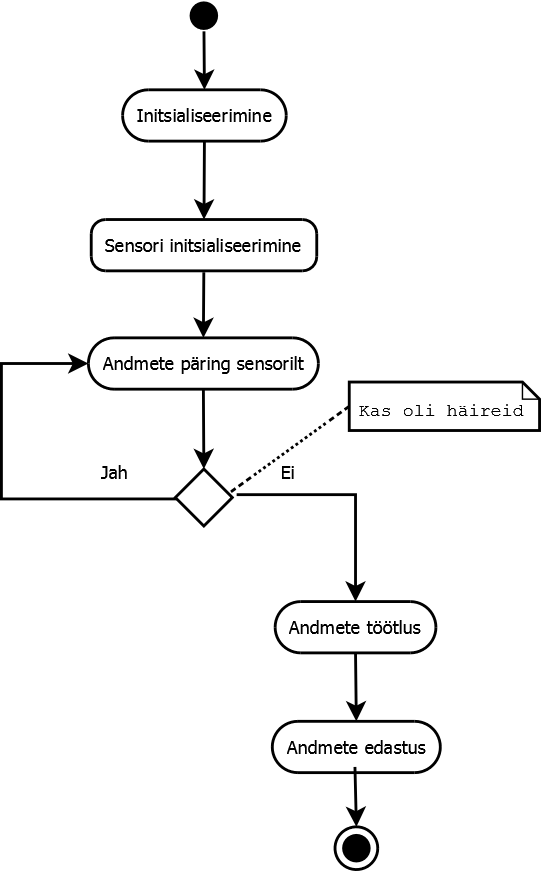
## Programmi töö

Esialgu initsialiseeritakse kõik sisemised seadistused arendusplaadil, seejärel saadetakse konfiguratsiooni seaded sensorile. Peale initsialiseerimist toimub andmete päring sensorilt. Juhul kui andmed olid vigased, küsitakse andmed sensorilt uuesti.

Andmete kontroll toimub järgnevalt. Kokku X, Y, Z koordinaatide jaoks tuleb küsida 6 andme baiti. On võimalik, et sensor saadab hoopis vea teate, see võib juhtuda andmete pärimise alguses või lõpus. Sellest hoolimata vigaste andmete korral, ei tohi neid töödelda.

Kui andmete päring õnnestus, toimub andmete töötlus. Et saada loetavad andmeid, tuleb panna 2 baiti kokku. Täisarv koosneb kahest baidist, kus esimene bait tähistab vähima kaaluga baiti ning teine bait suurimat. Programmi tööd kirjeldab joonis 6.

Näiteks: MSB – 1111 1110 (OxFE) . LSB – 0000 0110 (0x06), siis töödeldud kujul on esitatud arvu -506.



Joonis - Programmi töö

**Arendusplaadi initsialiseerimine:**

Arendusplaadi initsialiseerimine on näidatud joonisel 7.

**Hal teegi initsialiseerimine** [[14]](#endnote-14)

Peab olema esimene funktsioon, mida käivitatakse. HAL draiver on disainitud, et pakkuda rikkaliku API-si, et suhelda aplikatsiooni kõrgemate kihtidega. Iga draiver koosneb funktsioonide hulgast, mis katavad enamus väliseadme funktsioone.

**Näide**

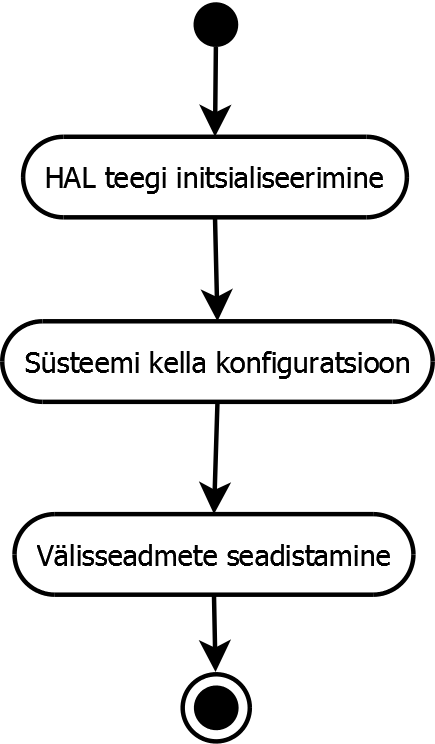
HAL\_UART\_Abort (UART\_HandleTypeDef \*huart) – Tühistab käimasolevad edastused.

**Süsteemi kella konfiguratsioon:**

Initsialiseerib sisemise väljund pinge. Initsialiseerib CPU, AHB ja ABP siini kellad. Süsteemi kell käsitleb arvutisüsteemis kõiki sünkroniseerimisist.

**Välisseadmete seadistamine:**

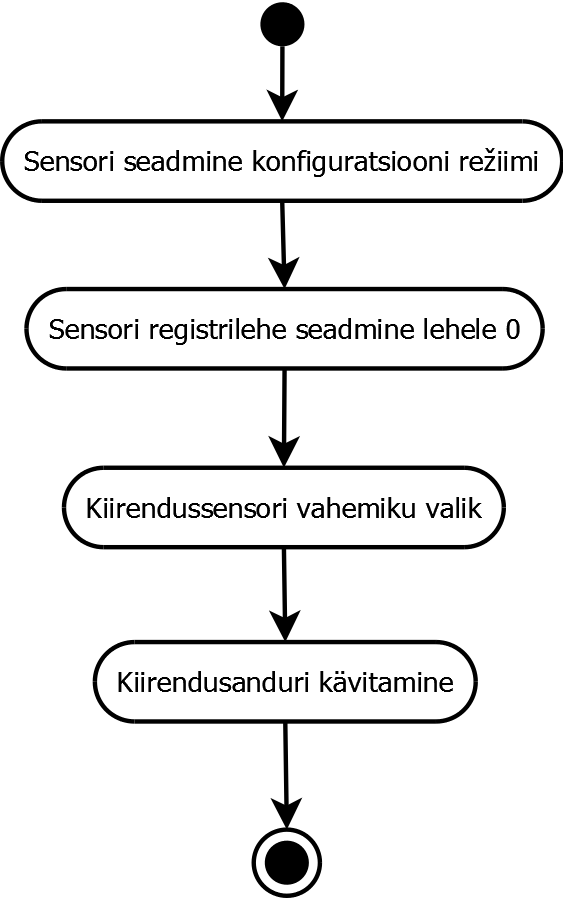
Initsialiseerib kõik konfigureeritud väljundid: GPIO, UART, USART.



Joonis - Arendusplaadi initsialiseerimine

**Sensori Initsialiseerimine:**

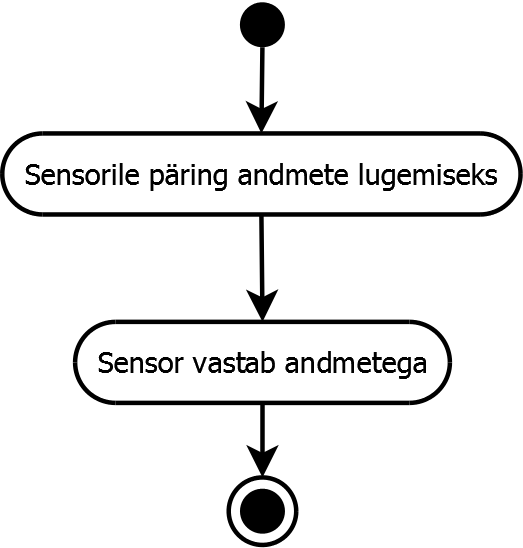
Sensori käivitamisel saadetakse sensorile konfiguratsiooni seaded, mis seadetega sensor tööle hakkab. Esmalt lülitakse sensor konfiguratsiooni režiimi, seejärel valitakse registrileht 0. Sensorile saadetakse kiirendusanduri vahemik 8G ning seejärel käivitatakse sensor kiirendusanduri režiimis. Sensori initsialiseerimis kirjeldab joonis 8.



Joonis - Sensori initsialiseerimine

**Andmete päring:**

Andmete saamiseks tuleb sensorile saata andmete lugemiseks päring, mis on näidatud tabel 5-es. Sensor saadab selle peale vastuse. Vastus võib endas sisaldada kas õigeid andmeid või viga. Õiged andmed on kirjeldatud tabelis 6 ning vigane vastus on kirjeldatud tabelis 7. Andmete päringut kujutab joonis 9.

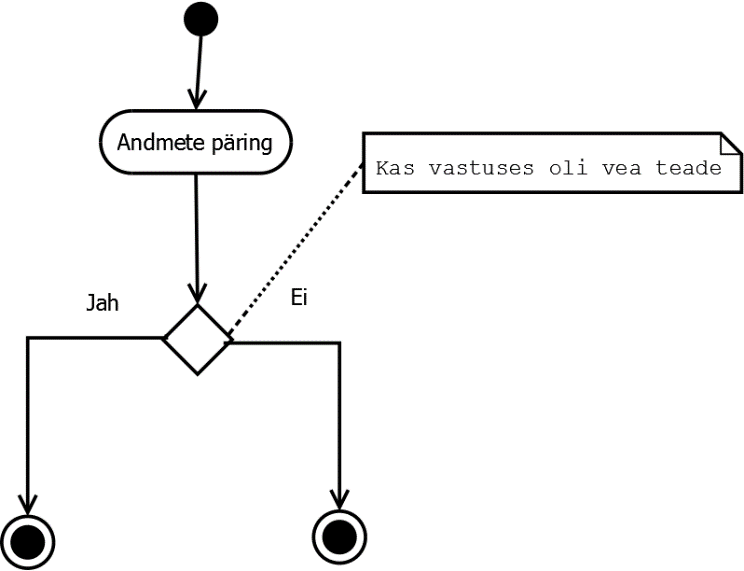


Joonis - Andmete päring

**Häirete kontroll:**

Peale andmete päringut kontrollitakse, kas andmed olid õiged. Õnnestunud lugemise vastus on kirjeldatud tabelis 6. Nurjunud lugemise vastus on kirjeldatud tabelis 7.

Juhul kui algus bait on 0xBB on tegemist vigadeta andmetega. Juhul kui algus bait on 0xEE on tegemist nurjunud lugemisega. Vigase lugemise korral andmeid ei töödelda.



Joonis - Häirete kontroll

**Andmete edastus:**

Juhul kui andmed on õiged, siis saadetakse andmed USB porti. Andmeid saab lugeda kasutades programmi RealTerm[[15]](#endnote-15).

**Kiirendusanduri kalibreerimine:**

Kiirendusandureid on võimalik kalibreerida, selleks tuleb hoida all sinist nuppu viieks sekundiks, mis on näidatud joonisel 13 B1.

Seejärel on kahesekundiline auk. Juhul kui selle kahesekundi jooksul vajutada sinist nuppu seadistatakse sensor kahte. Juhul kui nuppu ei vajutata, seadistatakse sensor üks.

Tulukest LD2 ning nuppu B1 kasutatakse seadistamisel joonis 13 ning joonis 14.

Esialgu seadistatakse sensori X telge. Tuluke LD2 hakkab vilkuma sagedusega 1000 ms. Sensor tuleb paigutada X telje suhtes tasasele pinnale ning seejärel tuleb vajutada sinist nuppu B1.

Järgmisena seadistatakse sensori Y telge. Tuluke LD2 hakkab vilkuma sagedusega 400ms. Sensor tuleb paigutada Y telje suhtes tasasele pinnale ning seejärel tuleb vajutada sinist nuppu B1.

Viimasena seadistatakse sensori Z telge. Tuluke LD2 hakkab vilkuma sagedusega 100 ms. Sensor tuleb paigutada Z telje suhtes tasasele pinnale ning seejärel tuleb vajutada sinist nuppu B1.

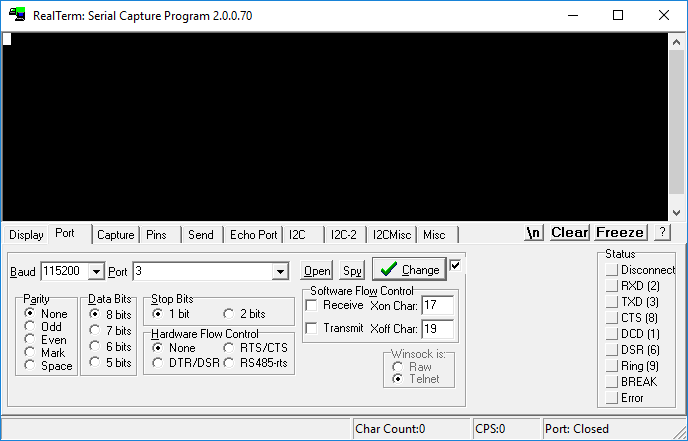
Järgnevalt saadetakse sensorile andmed kalibreerimis andmed ning sensor on seadistatud.

**Testimine:**

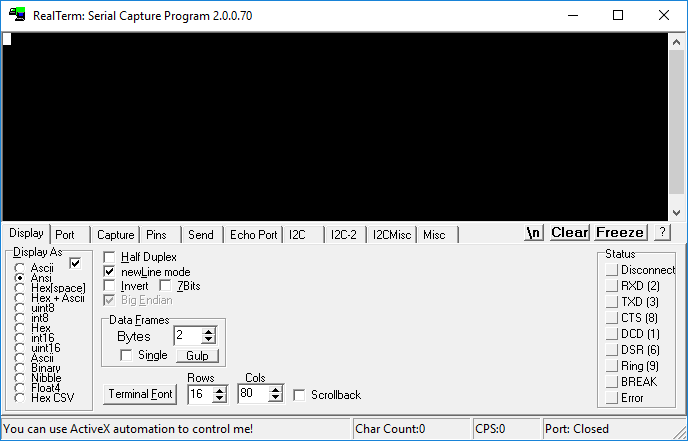
Andmete kuvamiseks on kaks võimalust. Üheks on ühendada arendusplaat USB kaabliga arvuti külge ning kasutada programmi RealTerm, et jälgida andmeid.

**Realterm seaded:**

Realterm seaded on näidatud joonisel 11 ja joonisel 12.

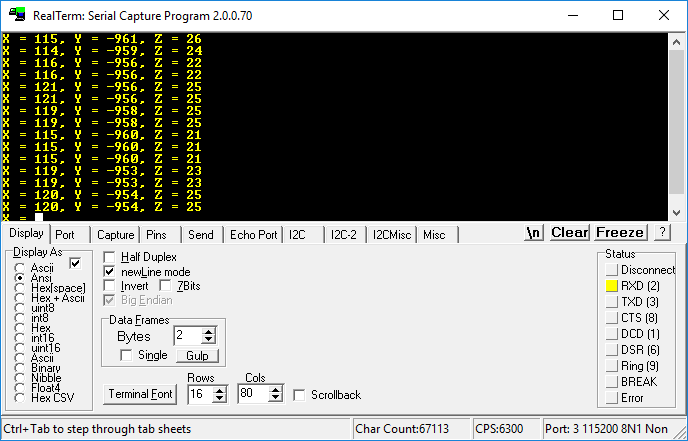
****

Joonis - Realterm Port seaded

****

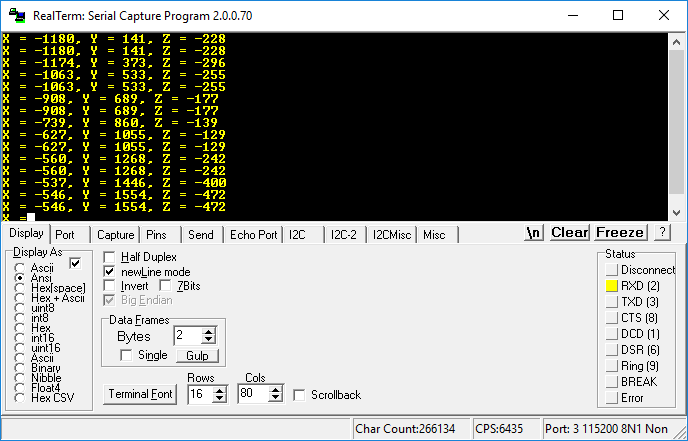
Joonis - Realterm Displei seaded

Normaal tingimustes peaksid andmed jääma -981 ja 981 vahemiku. Nagu näidatud joonisel 13.



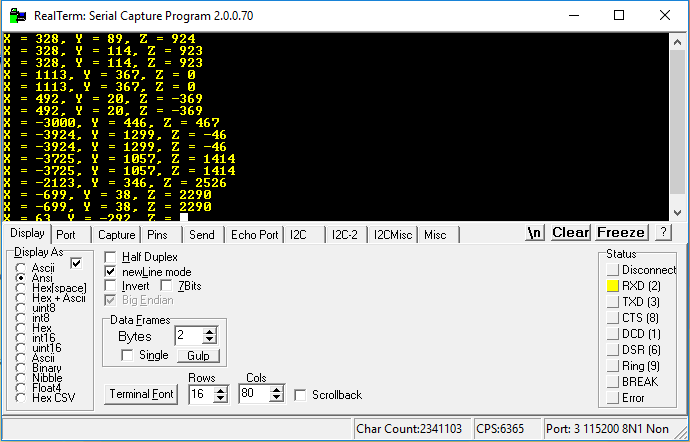
Joonis - Paigal olek

Kui toimub liikumine näiteks laev kõigub, siis andmed peaksid jääma -2000 ja 2000 vahele. Nagu näidatud joonisel 14.



Joonis - Liikumine

Kui toimub tugevam kokkupõrge peaksid andmed olema -2000 ja 8000 vahemikus. Nagu näitab joonis 15.



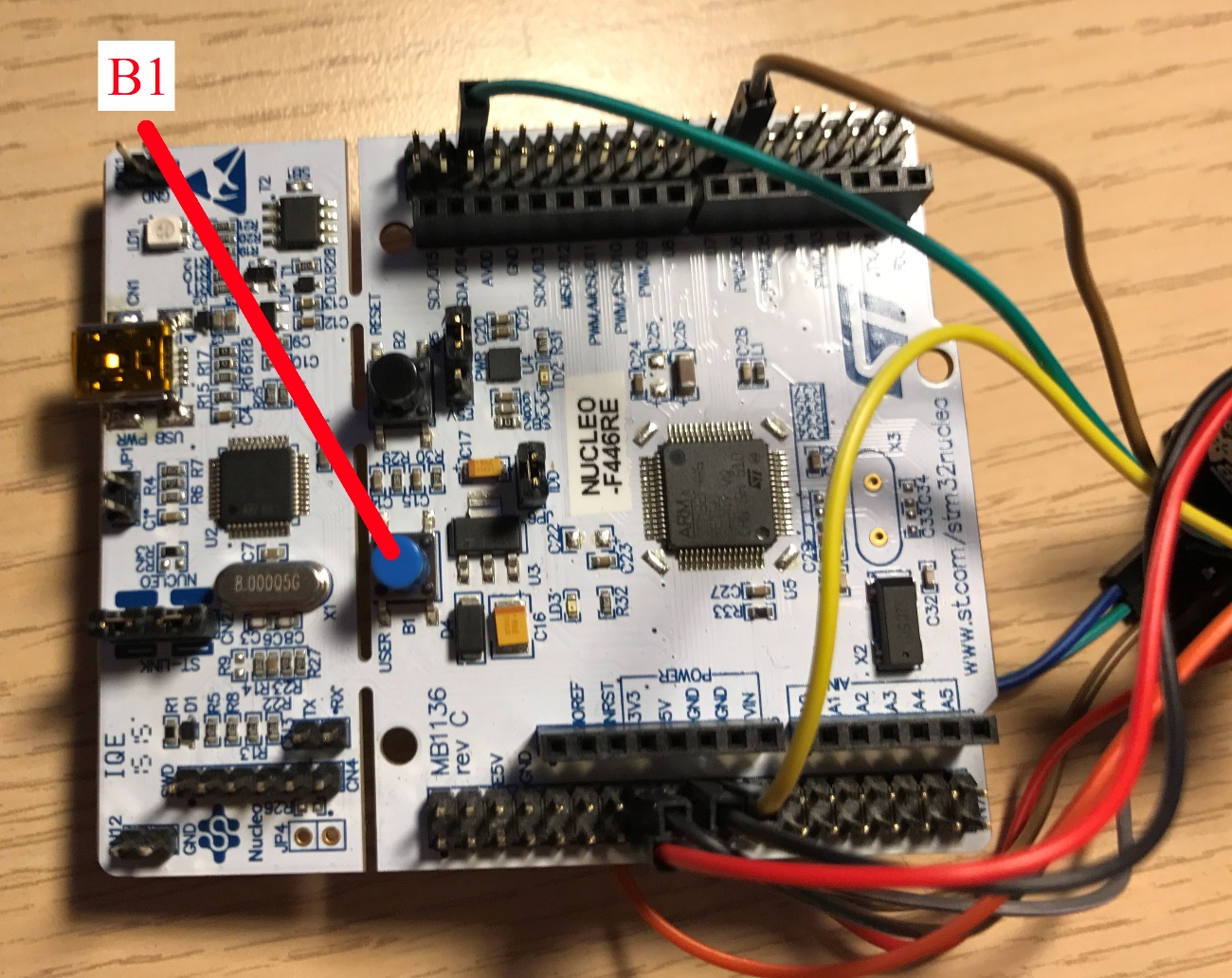
Joonis - Tugevam põrutus

# Kokkuvõte

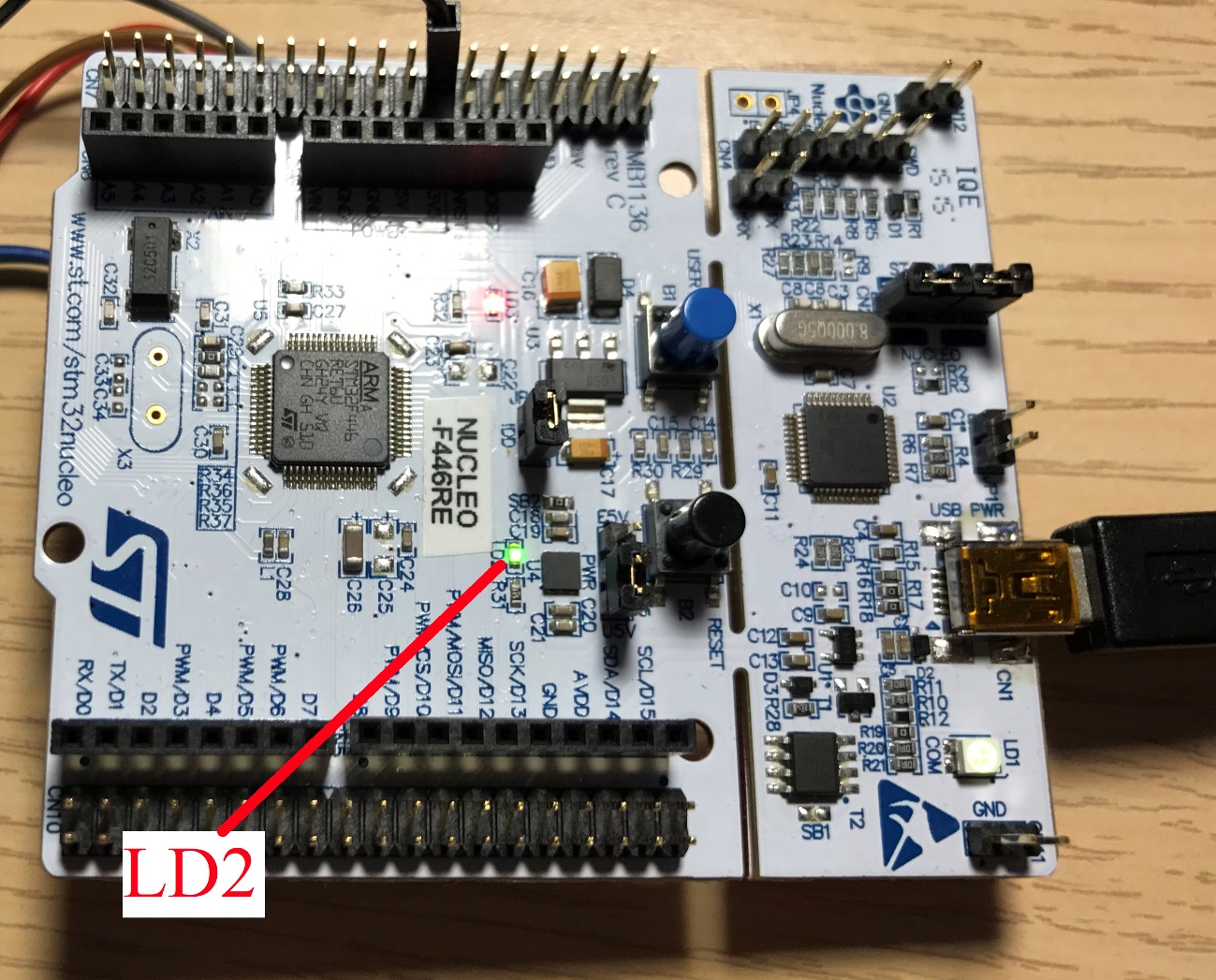
Kasutatud kirjandus

**There are no sources in the current document.**

Lisa 1 – Registri leht



Joonis - Arendusplaadi pilt



Joonis - Led tuluke

**Registri leht 0:**

|  |  |
| --- | --- |
| **Registri aadress** | **Registri nimi** |
| 7F-6B | Reserved |
| 6A | MAG\_RADIUS\_MSB |
| 69 | MAG\_RADIUS\_LSB |
| 68 | ACC\_RADIUS\_MSB |
| 67 | ACC\_RADIUS\_LSB |
| 66 | GYR\_OFFSET\_Z\_MSB |
| 65 | GYR\_OFFSET\_Z\_LSB |
| 64 | GYR\_OFFSET\_Y\_MSB |
| 63 | GYT\_OFFSET\_Y\_LSB |
| 62 | GYR\_OFFSET\_ X\_MSB |
| 61 | GYR\_OFFSET\_X\_LSB |
| 60 | MAG\_OFFSET\_Z\_MSB |
| 5F | MAG\_OFFSET\_Z\_LSB |
| 5E | MAG\_OFFSET\_Y\_MSB |
| 5D | MAG\_OFFSET\_Y\_LSB |
| 5C | MAG\_OFFSET\_X\_MSB |
| 5B | MAG\_OFFSET\_X\_LSB |
| 5A | ACC\_OFFSET\_Z\_MSB |
| 59 | ACC\_OFFSET\_Z\_LSB |
| 58 | ACC\_OFFSET\_Y\_MSB |
| 57 | ACC\_OFFSET\_Y\_LSB |
| 56 | ACC\_OFFSET\_X\_MSB |
| 55 | ACC\_OFFSET\_X\_LSB |
| 43 – 54 | Reserved |
| 42 | AXIS\_MAP\_SIGN |
| 41 | AXIS\_MAP\_CONFIG |
| 40 | TEMP\_SOURCE |
| 3F | SYS\_TRIGGER |
| 3E | PWR\_MODE |
| 3D | OPR\_MODE |
| 3C | Reserved |
| 3B | UNIT\_SEL |
| 3A | SYS\_ERR |

|  |  |
| --- | --- |
| **Registri aadress** | **Registri nimi** |
| 39 | SYS\_STATUS |
| 38 | SYS\_CLK\_STATUS |
| 37 | INT\_STA |
| 36 | ST\_RESULT |
| 35 | CALIB\_STAT |
| 34 | TEMP |
| 33 | GRV\_Data\_Z\_MSB |
| 32 | GRV\_Data\_Z\_LSB |
| 31 | GRV\_Data\_Y\_MSB |
| 30 | GRV\_Data\_Y\_LSB |
| 2F | GRV\_Data\_X\_MSB |
| 2E | GRV\_Data\_X\_LSB |
| 2D | LIA\_Data\_Z\_MBS |
| 2C | LIA\_Data\_Z\_LSB |
| 2B | LIA\_Data\_Y\_MBS |
| 2A | LIA\_Data\_Y\_LSB |
| 29 | LIA\_Data\_X\_MBS |
| 28 | LIA\_Data\_X\_LSB |
| 27 | QUA\_Data\_Z\_MSB |
| 26 | QUA\_Data\_Z\_LSB |
| 25 | QUA\_Data\_Y\_MSB |
| 24 | QUA\_Data\_Y\_LSB |
| 23 | QUA\_Data\_X\_MSB |
| 22 | QUA\_Data\_X\_LSB |
| 21 | QUA\_Data\_W\_MSB |
| 20 | QUA\_Data\_W\_LSB |
| 1F | EUL\_Pitch\_MSB |
| 1E | EUL\_Pitch\_LSB |
| 1D | EUL\_Roll\_MSB |
| 1C | EUL\_Roll\_LSB |
| 1B | EUL\_Heading\_MSB |
| 1A | EUL\_Heading\_LSB |
| 19 | GYR\_DATA\_Z\_MSB |
| 18 | GYR\_DATA\_Z\_LSB |
| 17 | GYR\_DATA\_Y\_MSB |
| 16 | GYR\_DATA\_Y\_LSB |
| 15 | GYR\_DATA\_X\_MSB |
| 14 | GYR\_DATA\_X\_LSB |
| 13 | MAG\_DATA\_Z\_MSB |
| 12 | MAG\_DATA\_Z\_LSB |
| 11 | MAG\_DATA\_Y\_MSB |
| 10 | MAG\_DATA\_Y\_LSB |
| F | MAG\_DATA\_X\_MSB |
| E | MAG\_DATA\_X\_LSB |
| D | ACC\_DATA\_Z\_MSB |

|  |  |
| --- | --- |
| **Registri aadress** | **Registri nimi** |
| C | ACC\_DATA\_Z\_LSB |
| B | ACC\_DATA\_Y\_MSB |
| A | ACC\_DATA\_Y\_LSB |
| 9 | ACC\_DATA\_X\_MSB |
| 8 | ACC\_DATA\_X\_LSB |
| 7 | Page ID |
| 6 | BL\_Rev\_ID |
| 5 | SW\_REV\_ID\_MSB |
| 4 | SW\_REV\_ID\_LSB |
| 3 | GYR\_ID |
| 2 | MAG\_ID |
| 1 | ACC\_ID |
| 0 | CHIP\_ID |

**Registri leht 1:**

|  |  |
| --- | --- |
| **Registri aadress** | **Registri nimi** |
| 7F – 60 | Reserved |
| 5F – 50 | UNIQUE\_ID |
| 4F – 20 | Reserved |
| 1F | GYR\_AM\_SET |
| 1E | GYR\_AM\_THRES |
| 1D | GYR\_DUR\_Z |
| 1C | GYR\_HR\_Z\_SET |
| 1B | GYR\_DUR\_Y |
| 1A | GYR\_HR\_Y\_SET |
| 19 | GYR\_DUR\_X |
| 18 | GYR\_HR\_X\_SET |
| 17 | GYR\_INT\_SETING |
| 16 | ACC\_NM\_SET |
| 15 | ACC\_NM\_THRE |
| 14 | ACC\_HG\_THRES |
| 13 | ACC\_HG\_DURATION |
| 12 | ACC\_INT\_Settings |
| 11 | ACC\_AM\_THRES |
| 10 | INT\_EN |
| F | INT\_MSK |
| E | Reserved |
| D | GYR\_Sleep\_Config |
| C | ACC\_Sleep\_Config |
| B | GYR\_Config\_1 |
| A | GYR\_Config\_0 |
| 9 | MAG\_Config |
| 8 | ACC\_Config |
| 7 | Page ID |
| 6 - 0 | Reserved |

1. http://www.st.com/en/microcontrollers/stm32f446.html?querycriteria=productId=LN1875 [↑](#endnote-ref-1)
2. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1617246 [↑](#endnote-ref-2)
3. https://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/DataSheets/mica2dot.pdf [↑](#endnote-ref-3)
4. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5226903 [↑](#endnote-ref-4)
5. https://www.lifeline.philips.com/safety-solutions/gosafe.html [↑](#endnote-ref-5)
6. https://www.shimmersensing.com/products/ [↑](#endnote-ref-6)
7. https://www.bosch-sensortec.com/bst/products/all\_products/bno055 [↑](#endnote-ref-7)
8. https://www.arm.com/products/processors/cortex-m/cortex-m4-processor.php [↑](#endnote-ref-8)
9. https://ae-bst.resource.bosch.com/media/\_tech/media/datasheets/BST\_BNO055\_DS000\_14.pdf [↑](#endnote-ref-9)
10. https://ae-bst.resource.bosch.com/media/\_tech/media/datasheets/BST\_BNO055\_DS000\_14.pdf [↑](#endnote-ref-10)
11. http://www.ftdichip.com/Support/Documents/TechnicalNotes/TN\_111%20What%20is%20UART.pdf [↑](#endnote-ref-11)
12. https://ae-bst.resource.bosch.com/media/\_tech/media/datasheets/BST\_BNO055\_DS000\_14.pdf [↑](#endnote-ref-12)
13. https://www.bell-labs.com/usr/dmr/www/chist.pdf [↑](#endnote-ref-13)
14. http://www.st.com/content/ccc/resource/technical/document/user\_manual/2f/71/ba/b8/75/54/47/cf/DM00105879.pdf/files/DM00105879.pdf/jcr:content/translations/en.DM00105879.pdf [↑](#endnote-ref-14)
15. https://realterm.sourceforge.io/ [↑](#endnote-ref-15)