VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Jukka Rannanmäki

LAITEOHJELMISTON SIIRTÄMINEN STM32H753:LLE

Tekniikka
2023

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Jukka Rannanmäki
Opinnäytetyön nimi	Laiteohjelmiston siirtäminen STM32H753:lle
Vuosi	2023
Kieli	Suomi
Sivumäärä	31
Ohjaaja	Santiago Chavez

Tämän opinnäytetyön tarkoituksena on siirtää Exertuksen laiteohjelmisto uudelle mikrokontrollerille. Työssä käytetty uusi mikrokontrolleri on STM32H753. Jokainen mikrokontrolleri eroaa aina jollain tavalla muista. Esimerkiksi mikrokontrollereissa on eri kommunikaatiorajapintoja sekä eri määrä A/D- ja D/A-muuntimia. STM:n omalla kääntäjäohjelmalla itse binääritiedoston kääntäminen STM32H753:lle onnistuu. Työn ongelmaksi jää oheislaitteiden konfigurointi toimimaan halutulla tavalla.

Exertus Oy tekee erilaisia CAN-väylää käyttäviä tuotteita. Oheislaitteiden konfigurointi tehtiin tulevan tuotteen suunnittelun pohjalta. Laiteohjelmistolle tärkein ominaisuus on CAN-väylä, joten työn tärkein osa on STM32H753:n kaksi FDCAN-väylää.

Työ aloitettiin tekemällä pohjaa tuleville laitteille, jotka käyttävät STM32H753-mikrokontrolleria. Pohjan kehitys tapahtui käyttäen STM32H753-EVAL2-arviointilevyä. Tämän jälkeen laiteohjelmisto siirrettiin tulevasta tuotteesta tehtyyn prototyyppiin.

Opinnäytetyön tuloksena on siirretty laiteohjelmisto uudelle mikrokontrollerille, jossa on otettu käyttöön STM32H753-EVAL2-arviointilevyn ominaisuuksista näyttö, CAN-väylä ja sarjaportti. Sarjaportti testattiin oskilloskoopilla ja CAN-väylä testattiin Exertuksen Canto2 CAN-ohjelmistotyökalulla.

Avainsanat laiteohjelmisto, sulautetut järjestelmät, Exertus Oy

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikka

ABSTRACT

Author	Jukka Rannanmäki
Title	Porting Firmware to STM32H753
Year	2023
Language	Finnish
Pages	31
Name of Supervisor	Santiago Chavez

The purpose of this thesis is to port Exertus' firmware to a new microcontroller. The new microcontroller used is STM32H753. Each microcontroller always differs in some way from the others. For example, microprocessors have different communication interfaces and a different number of A/D and D/A converters. With STM's own compiler program, it is possible to translate the binary itself for STM32H753. The aim of the thesis was to configure the peripherals to work as desired.

Exertus Ltd. manufactures various products that use CAN bus. The configuration of peripherals was made on the basis of the design of the future product. For the firmware, the most important feature is the CAN bus, so the most important part of the thesis is the two FDCAN buses of the STM32H753.

The project began by laying the foundation for future equipment using the STM32H753 microcontroller. The development of the base took place using the STM32H753-EVAL2 evaluation board. After that, the firmware was transferred to the prototype made of the future product.

The result of the thesis is firmware that has been ported to a new microcontroller. The used peripherals of STM32H753-EVAL2 evaluation board are its display, CAN bus and serial port. The serial port was tested with an oscilloscope and the CAN bus was tested with Exertus' Canto2 CAN software tool.

Keywords firmware, embedded systems, Exertus Ltd

SISÄLLYS

TIIVISTELMÄ

ABSTRACT

KUVALUETTTELO

LYHENNELUETTELO

1 JOHDANTO ... 8

2 STM32H753XI ... 9

2.1 Arm Cortex-M7 ydin ... 10

2.2 Keskeytykset ... 11

2.2.1 NVIC ... 11

2.3 STM32H7 HAL- ja low layer API ... 12

3 OHJELMISTOTYÖKALUT ... 13

3.1 STM32CubeProgrammer ... 13

3.1.1 ST-Link ... 13

3.2 STM32CubeIDE ... 13

3.2.1 GCC kääntötyökaluketju ... 14

3.2.2 GDB ... 14

3.3 Canto2 ... 14

3.4 Guitu ... 15

4 LAITTEISTO ... 16

5 TOTEUTUS ... 17

5.1 Laiteohjelmiston käyttämät kirjastot ... 17

5.2 Laitteen ajastimen alustus ... 18

5.3 FDCANin alustus ... 18

5.4 USARTin alustus ... 19

5.5 Näyttö ... 20

5.6 Virheenkäsittelijä ... 22

5.7 Ohjelman skriptin tekeminen ... 22

6 TESTAUS 25
6.1 CAN-liikenteen testaus Cantolla 25
6.2 Sarjaportin liikenteen testaus 26
6.4 Graafisen käyttöliittymän testaus 28
7 LOPPUPÄÄTELMÄT JA JATKOKEHITYS 30
LÄHTEET 31

KUVALUETTELO

Kuva 1. STM32H753XI:n piirikaavio. 9

Kuva 2. Cortex-M7:n komponenttien lohkokaavio. 10

Kuva 3. Yksinkertainen lohkokaavio työn kuvauksesta 17

Kuva 4. Sarjaportin alustus nopeudella 57600 bits/s 23

Kuva 5. Sarjaportin alustus nopeudella 115200 bits/s 24

Kuva 6. CAN-liikenne Cantossa 25

Kuva 7. Sarjaportin data oskilloskoopilla nopeudella 57600 bits/s 26

Kuva 8. Sarjaportin datan ensimmäinen tavu 27

Kuva 9. Sarjaportin data oskilloskoopilla nopeudella 115200 bits/s 27

Kuva 10. GUITUlla tehty graafinen käyttöliittymä 28

Kuva 11. Käyttöliittymä kokeilukortin näytöllä 29

LYHENNELUETTELO

CAN	Ajoneuvoissa, koneissa ja teollisuudessa käytetty automaatioväylä
FDCAN/CAN-FD	CAN joustavalla tiedonsiirtonopeudella
MCU	Mikrokontrolleri
I/O	Tulo/lähtö
SPI	Synkroninen sarjaväylä
NVIC	Vektoroitu keskeytysohjain
FPU	Liukulukuyksikkö
GCC	Kokoelma GNU-projektin kääntäjiä
API	Ohjelmointirajapinta
GUI	Graafinen käyttöliittymä
GPIO	Yleiskäyttöinen tulo/lähtö
USART	Universaali synkroninen/asynkroninen sarjaliikenteen lähetin/vastaanotin

1 JOHDANTO

Exertus Oy valmistaa tuotteita erilaisia CAN-väylä järjestelmiä. Laitteita on Näytöllisiä CAN-laitteita, CAN-ohjaimia, CAN-antureita sekä CAN-ohjelmistoja. Näissä laitteissa kriittistä on toimivuus, luotettavuus sekä viiveettömyys.

Työssä tehdään pohjaa tuleville tuotteille, jotka käyttävät olemassa olevaa valmista laiteohjelmistoa. Oheislaitteiden ja CAN-väylän konfiguroinnin pohjana käytettiin vanhempia tuotteitta, jotta ohjelmisto toimisi samaan tapaan niiden kanssa. Exertuksen CAN-laitteita on käytössä esimerkiksi metsäkoneissa sekä kaivoskoneissa.

Laiteohjelmiston siirtäminen aloitettiin tekemällä pohjaa STM:n arviointilevyllä STM32H753-EVAL2, jolla testattiin siinä olevia oheislaitteita kuten LCD-näyttöä ja ulkoisia muisteja. Tämän jälkeen laiteohjelmisto siirrettiin tulevan tuotteen prototyypille käyttäen aiemmin tehtyä pohjaa ja testattiin siihen tulevia oheislaitteita.

2 STM32H753XI

STM32H753XI on ST:n valmistama MCU, joka kuuluu STM32H753-sarjaan. STM32H753XI on 32-bittinen, 480 MHz:n mikrokontrolleri, joka käyttää Arm Cortex-M7 ydintä. Kuvasta 1 nähdään mikrokontrollerin ominaisuuksia.

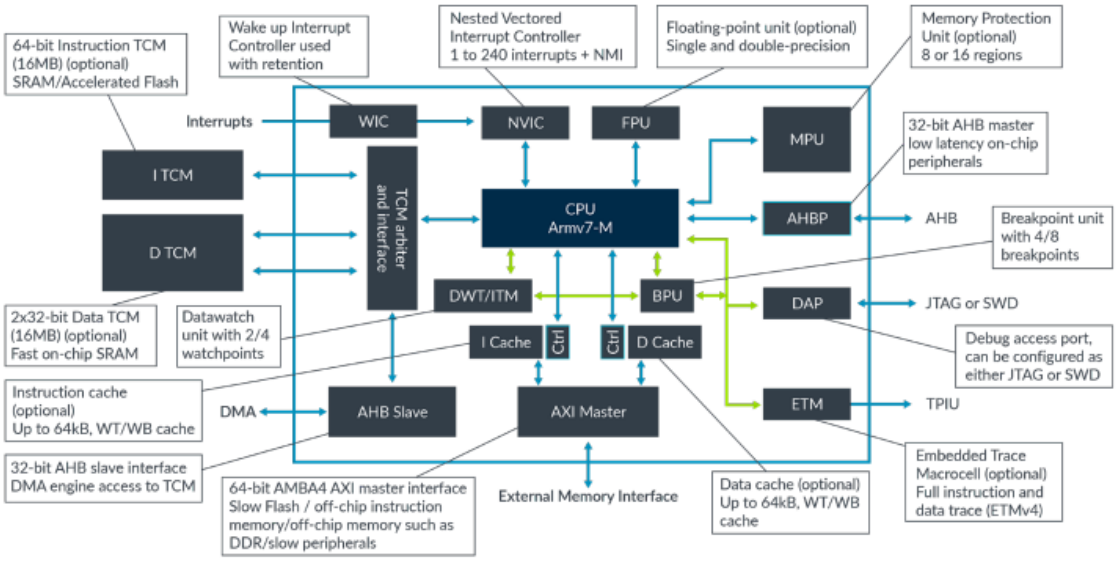
Kuva 1. STM32H753XI:n piirikaavio.[1]

[1] ST, STM32H753XI Overview, 2023

STM32H753-sarja on lähes sama kuin STM32H743-sarja, mutta siinä on lisättyjä turvallisuus ominaisuuksia. STM32H753-mikrokontrollereissa on Crypto/hash-laitteistokiihdytys.[2]

2.1 Arm Cortex-M7 ydin

Cortex-M sarjan prosessorit on suunniteltu kustannusherkkiä ratkaisuja varten. Tämän takia Cortex-M prosessoreita käytetään paljon sulautetuissa järjestelmissä. Cortex-M7 on Cortex-M sarjan suorituskykyisin prosessori. Kuvasta 2 nähdään Cortex-M7:n komponentit.[3]



Kuva 2. Cortex-M7:n komponenttien lohkokaavio.[4]

[2] ST, STM32H753XI Datasheet.pdf, 2023
[3] Arm, About the Cortex-M7 processor and core peripherals, 2023
[4] Arm, Arm Cortex-M7 Processor Datasheet.pdf, 2020

2.2 Keskeytykset

Lähes jokaisella STM32H753XI:n oheislaitteella on oma keskeytys. Siinä on 168 I/O porttia, joissa on keskeytyskyky.

Keskeytyksiä käytetään esimerkiksi saapuvan datan lukemiseen CAN-väylältä tai SPI:tä käyttäviltä oheislaitteilta. Keskeytyksen saadessaan laite menee voidessaan keskeytyksen käsittelevään funktioon ja tekee sille tarvittavat toimenpiteet kuten esimerkiksi lukee tai lähettää CAN-viestin.

2.2.1 NVIC

Arm Cortex-M7 ytimet käyttävät NVIC-keskeytysohjainta. Tällä on mahdollista hallita 16:sta eri prioriteettitasoa ja 150:tä maskilla varustettua keskeytyskanavaa. Tämän lisäksi siinä on 16 FPU:n keskeytyskanavaa.[5]

```
g_pfnVectors:
  .word  _estack
  .word  Reset_Handler

  .word  NMI_Handler
  .word  HardFault_Handler
  .word  MemManage_Handler
  .word  BusFault_Handler
  .word  UsageFault_Handler
  .word  0
  .word  0
  .word  0
  .word  0
  .word  SVC_Handler
  .word  DebugMon_Handler
  .word  0
  .word  PendSV_Handler
  .word  SysTick_Handler
```

[5] ST, STM32H753XI Datasheet.pdf, 2023

Yllä esimerkki STM32:n keskeytysvektorista. Tässä näkyy erilaisia käsitelijäitä, joita STM32H753 mikrokontrollerilla on. Tähän keskeytysvektoriin lisätään lisäksi myös oheislaitteiden keskeytysten käsittelijät, kuten työssä käytetyn sarjaportin keskeytyksen käsittelijä USART1_IRQHandler.

2.3 STM32H7 HAL- ja low layer API

STM32H7 HAL API on kehitetty yksinkertaistamaan STM:n mikrokontrollereiden ohjelmoimista. Tämä sisältää korkeamman tason funktioita pinnien ja oheislaitteiden alustukseen. Tämän APIn tarkoitus on piilottaa mikrokontrollerin ja oheislaitteiden monimutkaisuus.[6] Työssä käytettiin enemmän HAL APIa sen yksinkertaisuuden takia.

STM32H7 LL API tarjoaa monimutkaisemman, mutta nopeamman ja kevyemmän kirjaston STM:n mikrokontrollereiden ohjelmoimiseen. STM32H7 LL API on lähempänä laitteistoa ja se keskittyy mikrokontrollerin rekisterien tasolle.[7]

[6] ST, Description of STM32H7 HAL and low-layer drivers, 2022
[7] ST, Description of STM32H7 HAL and low-layer drivers, 2022

3 OHJELMISTOTYÖKALUT

Laiteohjelmiston kehittämisessä ja kääntämisessä käytettiin STM32CubeIDE-ohjelmaa. Ohjelman lataamisessa ja laitteen muistin muokkauksessa ja tyhjäämisessä käytettiin STM32CubeProgrammer-ohjelmaa.

3.1 STM32CubeProgrammer

STM32CubeProgrammer on STM:n ohjelmistotyökalu, jolla pystytään käsittelemään laitteen muistia helposti. Sillä pystytään lukemaan ja kirjoittamaan laitteen muistia, sekä varmistamaan, että kirjoitettu data on muistilla oikein. Sillä pystyy myös lataamaan laiteohjelmiston. Helppo tapa käyttää STM32CubeProgrammeria on ST-Linkin kautta.

3.1.1 ST-Link

ST-Link on STM:n laitteilla oleva ohjelmointiin ja debug-ominaisuuksiin tarkoitettu liityntä. Tämä toimii yhdellä johdolla, kuten esimerkiksi STM32H753-EVAL2 kokeilukortilla tämä tapahtuu USB-johdolla.

3.2 STM32CubeIDE

STM32CubeIDE on Eclipse IDEn päälle tehty C ja C++ kehitysalusta tarkoitettu STM:n omille mikrokontrollereille ja mikroprosessoreille. Se sisältää STM:n mikrokontrollereille ja mikroprosessoreille tarvittavat GCC kääntötyökalut.

STMCubeIDEssä on myös graafinen pinnien, kellotaajuuksien ja oheislaiteiden konfigurointiin tarkoitettu työkalu. STMCubeIDE generoi näiden konfiguraatioiden pohjalta pohjan omalle koodille. Tämä helpottaan kellojen taajuuksien säätämisessä, sekä pinnien asettamisessa alkutilaan.

STMCubeIDEssä on GDB debug-ominaisuudet. Tämä helpottaa koodin virhe kohtien löytämisessä ja niiden korjaamisessa. Debug-tilassa pystyy myös lukemaan laitteen muistia ja sen rekistereitä.

3.2.1 GCC kääntötyökaluketju

Kääntötyökaluketjua (toolchain) mikrokontrollerille käytettiin STMCubeIDEn mukana tulevaa ARMin arm-none-eabi toolchainia. Tämä sisältää kaikki tarvittavat työkalut ja kääntäjän C-kielisen lähdekoodin kääntämiseen ARM ytimien ymmärtämään binääritiedostoon. Tämän käännetyn binääritiedoston voi ladata laitteelle suoraan STMCubeIDEllä tai STM32CubeProgrammerilla.

3.2.2 GDB

STMCubeIDEssä on sisäänrakennettu GDB. GDB debugger on ohjelmistotyökalu, jolla pystytään pysäyttämään ohjelman ajo tiettyyn kohtaan ongelmakohtien löytämiseksi ja niiden korjaamiseksi.

Kun ohjelman ajo on pysäytetty, GDB:llä pystyy lukemaan tietyn funktion tai julkisia muuttujia, sekä muistia. Testaamisen kannalta voidaan myös lukea ja muuttaa oheislaitteiden rekisterejä, jos halutaan esimerkiksi muuttaa lähdön tilaa.

3.3 Canto2

Canto2 on Exertuksen kehittämä CAN-työkalu, jolla pystytään monitoroimaan CAN-väylää sekä lähettämään ja vastaanottamaan CAN-viestejä. Sillä pystyy monitoroimaan moduulien lähettämiä arvoja, lisäämään niitä tarkkailuun ja piirtämään näistä arvoista käyriä.

Canto2 on kehitetty erityisesti Exertuksen tuotteille ja se toimii CANopen -kommunikaatioprotokollalla. Sillä pystyy myös lisäämään laite kohtaisia PDO- ja SDO-viestejä.[8]

---

[8] Exertus, Canto2 Monitoring and commissioning tool, 2015

3.4 Guitu

GUITU on graafisen käyttöliittymän kehittämiseen tehty työkalu. Myöhemmin GUITUsta tuli myös Exertuksen laitteiden I/O:iden ohjelmointiin tarkoitettu työkalu.[9]

Työssä GUITUa käytettiin tekemään siirrettävään laiteohjelmistoon graafista käyttöliittymää. Sillä tehtiin myös laitaohjelmistolle skripti, jota laiteohjelmisto käyttää ajaessaan. Skriptillä pystytään ohjaamaan laiteohjelmiston käyttäytymistä ja joissain tapauksissa oheislaitteiden käyttäytymistä kuten esimerkiksi muuttamaan CAN-väylän ja sarjaportin nopeutta tekemättä muutoksia itse laiteohjelmistoon.

[9] Exertus, 2015. GUITU Programming environment, 2015

4 LAITTEISTO

Laitteistona työssä käytettiin STM32H753-Eval2 kokeilukorttia. Tällä tarkoituksena oli saada laiteohjelmisto pyörimään STM32H753XI mikrokontrollerilla ja tehdä pohjaa tuleville tuotteille kokeilukortin ominaisuuksien puitteissa.

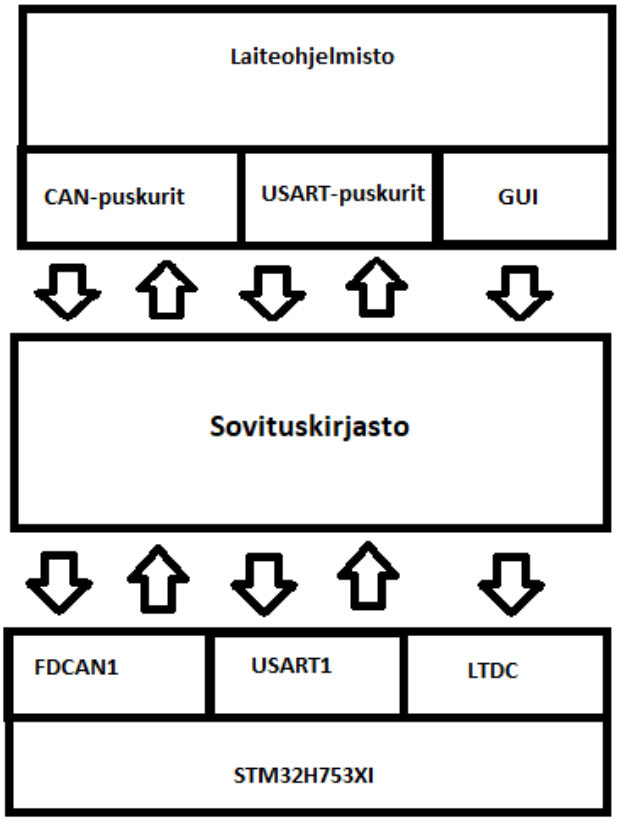
STM32H753-EVAL2 on kehityskortti, jossa on tarvittavat oheislaitteet siirrettävän koodin toimimiseen ja sen testaukseen. Tällä kehityskortilla aloitettiin pohjan tekemistä FDCAN1:lle.

Työssä STM32H753-EVAL2 kehityskortilla pohjan kehitystä on tehty sen oheislaitteille, joita tullaan todennäköisesti käyttämään tulevissa Exertuksen tuotteissa. Näitä ominaisuuksia on mikrokontrollerin FDCAN-moduulit, sen näytön ajurit sekä sarjaportti. Työssä käytettiin myös kokeilukortilta löytyviä nappeja, joita testattiin graafisen käyttöliittymään ohjaamisessa.

5 TOTEUTUS

Opinnäytetyön tarkoituksena oli siirtää Exertuksen laiteohjelmisto STM32H753 mikrokontrollerille ja tehdä pohjaa tuleville tuotteille, jotka tulevat käyttämään kyseistä laiteohjelmistoa.

Työtä lähdettiin tekemään niin, että tehdään mikrokontrollerin ja laiteohjelmiston väliin kirjastoa. Tämä kirjasto sisältää laiteohjelmiston sovittamiseen mikrokontrollerille tarvittavat alustukset ja laiteohjelmiston tarvitsemat muut laitekohtaiset ominaisuudet (**Kuva 3**.).



Kuva 3. Yksinkertainen lohkokaavio työn kuvauksesta

5.1 Laiteohjelmiston käyttämät kirjastot

Työssä käytettävät kirjastot käännettiin lib.a -tiedostoiksi, eli C-ohjelmointikielen kirjasto muotoon. Nämä saa käännettyä helposti STM32CubeIDEllä. Projektiin

käännettyjä kirjastoja olivat STM32:n BSP API, missä on valmiina funktioita STM32H753-EVAL2 kokeilukortille, STM32:n HAL- ja LL API sekä CMSIS. Nämä kaikki tulevat STM32CubeIDEn hakemassa kehityspaketissa.

5.2 Laitteen ajastimen alustus

Oheislaitteiden ajastukseen laiteohjelmisto käyttää TIM6-keskeytystä. Tämä ajastinkeskeytys alustetaan pyörittämään ajastinta millisekunnin välein. Tälle ajastimelle löytyy valmiit alustukset STM32H7:n HAL APIsta.

5.3 FDCANin alustus

STM32H753 mikrokontrollerilla on kaksi FDCAN-moduulia. STM32H753-EVAL2 kokeilukortilla näistä toinen on kytkettynä DB-9 urosliittimeen. Liitimen ja mikrokontrollerin välissä on CAN-väylän lähetinvastaanotin. Tämä muuttaa FDCAN1:en lähetys -signaalin CAN high- ja CAN low -signaaleiksi ja vastaavasti vastaanotetun signaalin CAN high- ja CAN low -signaaleista mikrokontrollerin ymmärtämään vastaanotto -signaaliksi

Kokeilukortilla tämä FDCAN1-moduulin liitin on kytkettynä mikrokontrollerin GPIO:n A-portin pinneihin 11 ja 12. Nämä mikrokontrollerin pinnit alustetaan vaihtelevaan toiminnallisuuteen.

```
void HAL_FDCAN_MspInit(FDCAN_HandleTypeDef* hfdcan) {
    GPIO_InitTypeDef  GPIO_InitStruct = { 0 };

    if(hfdcan->Instance == FDCAN1) {
        HAL_RCC_FDCAN_CLK_ENABLED = 1;

        if(HAL_RCC_FDCAN_CLK_ENABLED == 1) {
            __HAL_RCC_FDCAN_CLK_ENABLE();
        }

        __HAL_RCC_GPIOA_CLK_ENABLE();
        GPIO_InitStruct.Pin       = GPIO_PIN_11 | GPIO_PIN_12;
        GPIO_InitStruct.Mode      = GPIO_MODE_AF_PP;
        GPIO_InitStruct.Pull      = GPIO_PULLUP;
        GPIO_InitStruct.Speed     = GPIO_SPEED_FREQ_LOW;
        GPIO_InitStruct.Alternate = GPIO_AF9_FDCAN1;
        HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
    }
}
```

Yllä olevassa funktiossa FDCAN1:en pinnit alustetaan sen tarvitsemaan vaihtelevaan toiminnallisuuteen Push pull-tilaan pienellä taajuudella. STM32H753 mikrokontrollerin FDCAN-moduulit käyttävät alternate function yhdeksää. Funktiossa laitetaan myös FDCAN-moduulille ja GPIO:n A-portille päälle kellot.

5.4 USARTin alustus

STM32H753-EVAL2 kokeilukortilla on samanlainen DB-9 urosliitin myös sarjaportille kuin CAN-väylälle. Tämän sarjaportin liitin on yhdistetty mikrokontrollerin GPIO:n B-portin pinneihin 14 ja 15. Pinni 15 on vastaanottava pinni ja pinni 14 on lähettävä pinni. Nämä USARTin pinnit alustetaan vaihtelevaan toiminnallisuuteen.

```
void USART_Init(int port) {
    if (port == 0) {
        GPIO_InitTypeDef  GPIO_InitStruct = { 0 };
        UART_HandleTypeDef UartHandle;


        __HAL_RCC_USARTT1_CLK_ENABLE();
        __HAL_RCC_GPIOB_CLK_ENABLE();

        GPIO_InitStruct.Pin       = GPIO_PIN_14;
        GPIO_InitStruct.Mode      = GPIO_MODE_AF_PP;
        GPIO_InitStruct.Pull      = GPIO_PULLUP;
        GPIO_InitStruct.Speed     = GPIO_SPEED_FREQ_VERY_HIGH;
        GPIO_InitStruct.Alternate = GPIO_AF4_USART1;
        HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

        GPIO_InitStruct.Pin = GPIO_PIN_15;
        GPIO_InitStruct.Alternate = GPIO_AF4_USART1;
        HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

        UartHandle.Init.BaudRate   = 115200;
        UartHandle.Init.WordLength = UART_WORDLENGTH_8B;
        UartHandle.Init.StopBits   = UART_STOPBITS_1;
        UartHandle.Init.Parity     = UART_PARITY_NONE;
        UartHandle.Init.HwFlowCtl  = UART_HWCONTROL_NONE;
        UartHandle.Init.Mode       = UART_MODE_TX_RX;
        UartHandle.Init.OverSampling = UART_OVERSAMPLING_16;

        if(HAL_UART_Init(&UartHandle) != HAL_OK) {
            Error_Handler();
        }
    }
}
```

Yllä olevassa funktiossa USART1:en pinnit alustetaan sen vaihtelevaan toiminnallisuuteen alternate function neljän Push pull-tilaan korkealla taajuudella. Alustavasti sarjaportin siirtonopeudeksi asetetaan 115200. Tämä tarkoittaa, että sarjaportti lähettää 115200 bittiä sekunnissa eli 14400 tavua sekunnissa. Tässä funktiossa laitetaan myös USART1:en ja GPIO:n A-portille kellot päälle.

Sarjaportin nopeutta pystyy kuitenkin muuttamaan GUITU-ohjelmalla tehdyllä skriptillä. Tässä tapauksessa laitteen sarjaportin nopeutta voidaan muuttaa kääntämättä laiteohjelmistoa uudelleen.

```
void USART_Set_Speed(unsigned speed) {
    if (speed != USART1->BRR) {
        USART1->BRR = USART1->CR1 & ~0x01;
        USART1->BRR = speed;
        USART1->BRR |= 0x01;
    }
}
```

Sarjaportin uuden siirtonopeuden voi muuttaa kirjoittamalla laskettu uusi nopeus USART1:en siirtonopeuden rekisteriin. USART1 pitää kuitenkin ottaa pois päältä tämän ajaksi, jonka pystyy tekemään nollaamalla USART1:en kontrolli rekisterin ensimmäisen bitin, eli USART enable bitin.

5.5 Näyttö

STM32H753-EVAL2 kokeilukortissa on 640x480 pikselin näyttö, joka on kooltaan 5,7-tuuma. STM32H753-mikrokontrollerilla on LTDC-kontrolleri, joka ohjaa näyttöä.

Tälle kokeilukortin näytölle löytyy valmis alustus funktio STM:n BSP-kirjastosta.

```
int32_t BSP_LCD_Init(uint32_t Instance, uint32_t Orientation);
```

Tässä Instance kertoo, mikä instanssi on kyseessä. Funktion toisella argumentilla kerrotaan näytön orientaatio. Näytön saa pysty- tai vaakasuuntaan. Testatessa näyttöä laiteohjelmiston kanssa, näyttö oli vaaka-asennossa.

Testatessa näytön resoluutioksi määritettiin 640x480 ja näytön tasoja käytössä oli vain yksi. Alla olevassa funktiossa näytön leveydeksi on laitettu 640 pikseliä ja korkeudeksi 480 pikseliä. X- ja Y-akseleiden aloitus pisteiksi on laitettu nollat ja X-akselin lopetuspisteeksi on 640 sekä Y-akselin lopetuspisteeksi on laitettu 480.

Näytön kehyspuskuri asetetaan pLayerCfg-tietueen FBStartAdress-jäseneen. Kehyspuskurissa on kuva, jota näyttö näyttää. Näytön värisyvyydeksi määritetään 16-bittinen RGB565, eli sillä on viisi punaista bittiä, kuusi vihreää bittiä ja viisi sinistä bittiä.

```
void lcd_layer_config (void) {
    pLayerCfg.WindowX0 = 0;
    pLayerCfg.WindowX1 = 640;
    pLayerCfg.WindowY0 = 0;
    pLayerCfg.WindowY1 = 480;

    pLayerCfg.PixelFormat = LTDC_PIXEL_FORMAT_RGB565;
    pLayerCfg.FBStartAdress = (uint32_t)&RGB565_640x480;

    pLayerCfg.Alpha = 255;
    pLayerCfg.Backcolor.Blue = 0;
    pLayerCfg.Backcolor.Green = 0;
    pLayerCfg.Backcolor.Red = 0;
    pLayerCfg.ImageWidth  = 640;
    pLayerCfg.ImageHeight = 480;

    if(HAL_LTDC_ConfigLayer(&hltdc_F, &pLayerCfg, 1) != HAL_OK)
    {
      Error_Handler();
    }
}
```

Näytön ikkunan vaihtamiseen käytettiin kokeilukortilla olevaa Tamper-nappia. Pinnit ovat valmiiksi alustettuna tuloiksi, eli tämä tarvitsee vain sen tilaa lukevan funktion. Tämä voidaan tehdä alla olevalla STM32H7:n HAL APIn funktiolla. Kokeilukortin nappi on kytketty mikrokontrollerin GPIO:n C-portin pinniin 13.

```
HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13);
```

5.6 Virheenkäsittelijä

STM32H753-mikrokontrolerille on erilaisia virheenkäsittelijöitä, joita mikrokontrolleri kutsuu virheen sattuessa. Ohjelmaan lisättiin Error_Hander-käsittelijä, johon ohjelma viedään yleisten virheiden tapahtuessa.

Virheen tapahtuessa virheenkäsittelijä voi laittaa koodin ikuiseen silmukkaan, joka tarkoittaa, että koodin ajo pysähtyy. Alla olevassa koodissa laiteohjelmisto laitetaan ikuiseen silmukkaan. Keskeytykset pitää ottaa tässä tapauksessa pois, että ohjelma ei pääse hyppäämään täältä pois.

```
void Error_Handler(void) {
    __disable_irq()
    while(1) { }
}
```

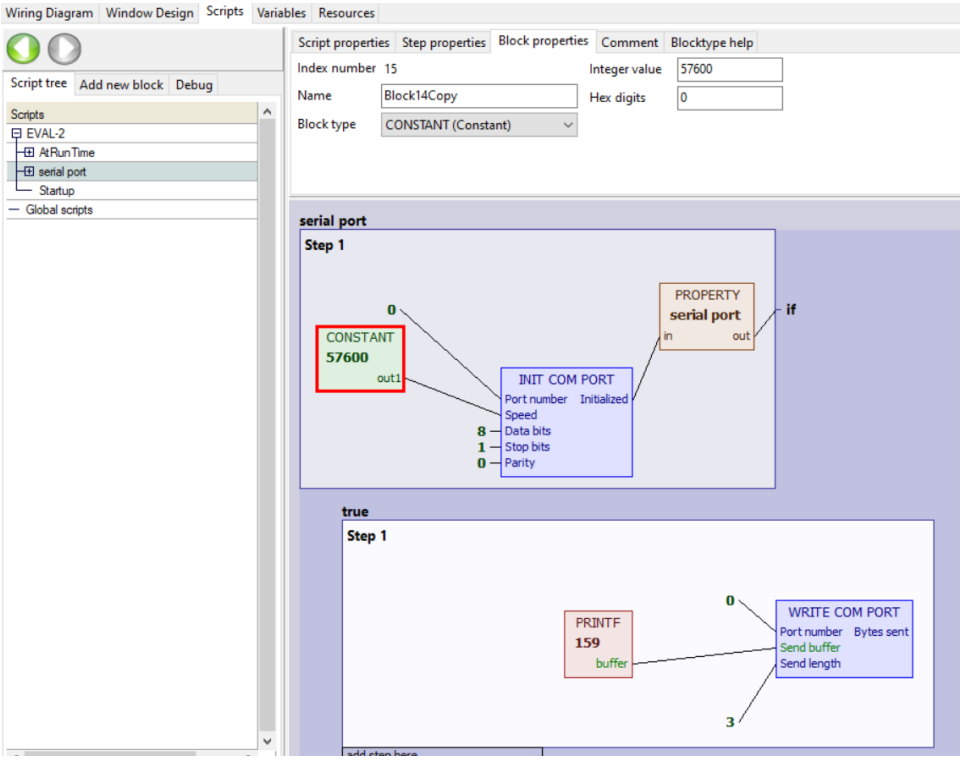
Virheen tapahtuessa laiteen lähdöt on myös hyvä ottaa pois päältä, että mitään ei mene hajalle. Joissain tapauksessa virheenkäsittelijä voi käynnistää laitteen uudelleen alla olevan koodin mukaisesti.

```
void Error_Handler(void) {
    NVIC_SystemReset();
}
```

5.7 Ohjelman skriptin tekeminen

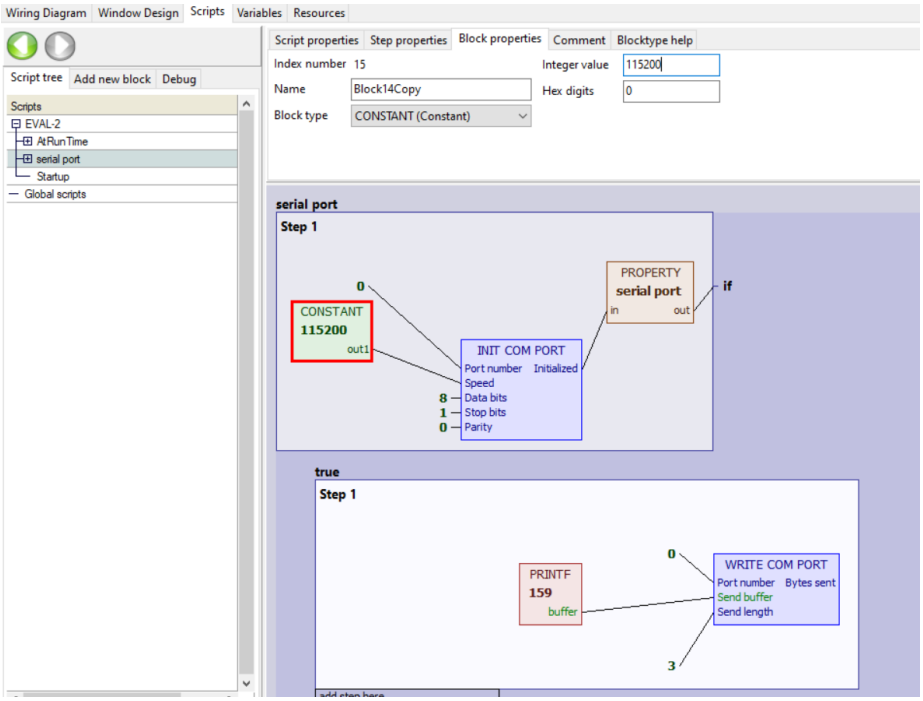
Laiteohjelma skriptin tehtiin Exertuksen Guitu-ohjelmalla. GUITUn Scripts-työkalulla pystytään valitsemaan lohkoja eri toiminnallisuuksilla. Tässä työssä käytettiin sarjaportin lohkoja. GUITUlla pystyy myös muuttamaan CAN-väylien nopeuksia, ja jos laitteella olisi useampi CAN-väylä, niin sillä pystyy määrittämään, mitä muut CAN-väylät lähettävät. Kokeilukortilla on käytössä vain yksi CAN-väylä ja se lähettää oletuksena laitteen tietoja, mikä riittää testauksessa.

Tällä skriptillä testattiin sarjaportin alustusta ja sen nopeuden muuttamista. Aluksi siirtonopeudeksi valittiin 57600 bits/s kuvan 4 mukaisesti. Lähetettäväksi testattavaksi merkkijonoksi on asetettu ”159”. Muutettuna ASCII-muotoon heksadesimaalina nämä merkit ovat 0x31, 0x35 ja 0x39.



Kuva 4. Sarjaportin alustus nopeudella 57600 bits/s

Tämän jälkeen sarjaportin siirtonopeudeksi vaihdettiin 115200 bits/s testaamista varten. Tällä testataan, että nopeus muuttuu oikein. Kuvassa 5 vaihdettuna sarjaportin siirtonopeus. Tämä lähettää saman merkkijonon, kuin pienemmälläkin siirtonopeudella. Nopeuden muutoksen toimivuus on testattu oskilloskoopilla.



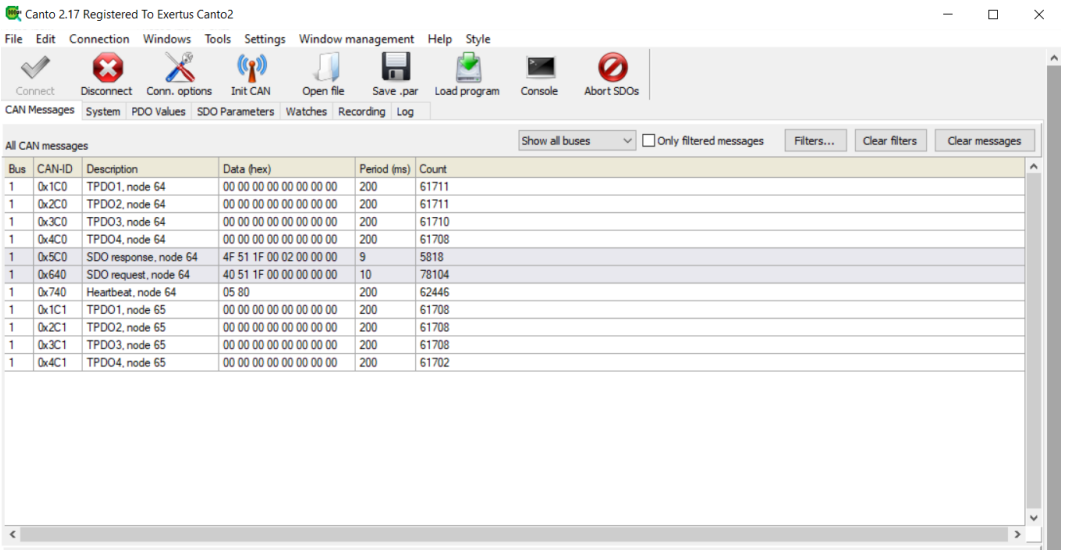
Kuva 5. Sarjaportin alustus nopeudella 115200 bits/s

6 TESTAUS

Testauksessa on testattu CAN-väylän liikennettä sekä sarjaportin liikennettä. CAN-väylän liikennettä testattiin Cantolla ja sarjaportin liikennettä testattiin oskilloskoopilla.

6.1 CAN-liikenteen testaus Cantolla

Laiteohjelma lähettää laitteen tietoja oletuksena ensimmäisen CAN-väylän läpi. CAN-liikennettä pystyy seuraamaan Exeruksen Canto-työkalulla. Kuvassa 6 näkyy laitteen lähettämät TPDO-viestin sekä sen Heartbeat. Cantolla pystyy myös lähettämään CAN-viestejä moduulille, joten näin pystyy testaamaan myös liikenteen toiseen suuntaan.



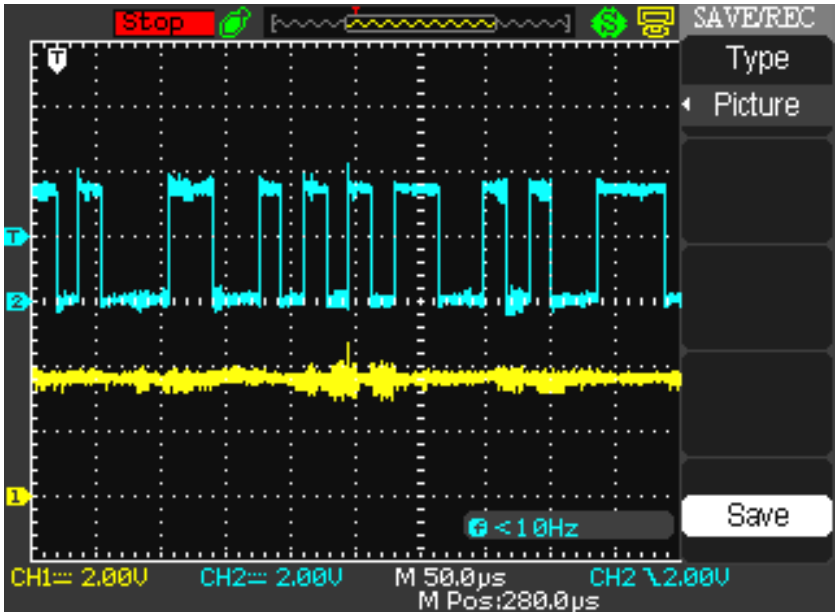
Bus	CAN-ID	Description	Data (hex)	Period (ms)	Count
1	0x1C0	TPDO1, node 64	00 00 00 00 00 00 00 00	200	61711
1	0x2C0	TPDO2, node 64	00 00 00 00 00 00 00 00	200	61711
1	0x3C0	TPDO3, node 64	00 00 00 00 00 00 00 00	200	61710
1	0x4C0	TPDO4, node 64	00 00 00 00 00 00 00 00	200	61708
1	0x5C0	SDO response, node 64	4F 51 1F 00 02 00 00 00	9	5818
1	0x640	SDO request, node 64	40 51 1F 00 00 00 00 00	10	78104
1	0x740	Heartbeat, node 64	05 80	200	62446
1	0x1C1	TPDO1, node 65	00 00 00 00 00 00 00 00	200	61708
1	0x2C1	TPDO2, node 65	00 00 00 00 00 00 00 00	200	61708
1	0x3C1	TPDO3, node 65	00 00 00 00 00 00 00 00	200	61708
1	0x4C1	TPDO4, node 65	00 00 00 00 00 00 00 00	200	61702

Kuva 6. CAN-liikenne Cantossa

CAN-väylän kautta ladataan myös laiteohjelmistolle GUITUlla tehty käyttöliittymän ja skripti tiedosto. Nämä tiedostot ovat isoja graafisen käyttöliittymän takia, joten tällä saa testattua myös CAN-väylän toimivuutta.

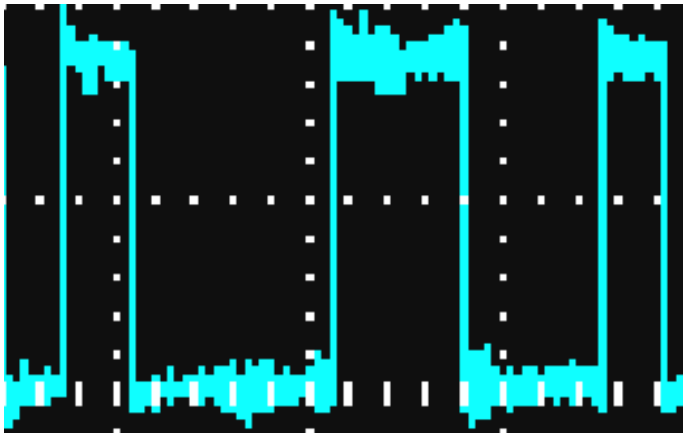
6.2 Sarjaportin liikenteen testaus

Sarjaportti lähettää vähiten merkitsevän bitin ensin (least significant bit). Tämä tarkoittaa, että tavu tulee käänteisessä järjestyksessä. Kuvassa 7 näkyy oskilloskoopilla merkkijono, jonka sarjaportti lähettää. Tämä merkkijono on sama, mikä määritettiin skriptissä eli ”159”.



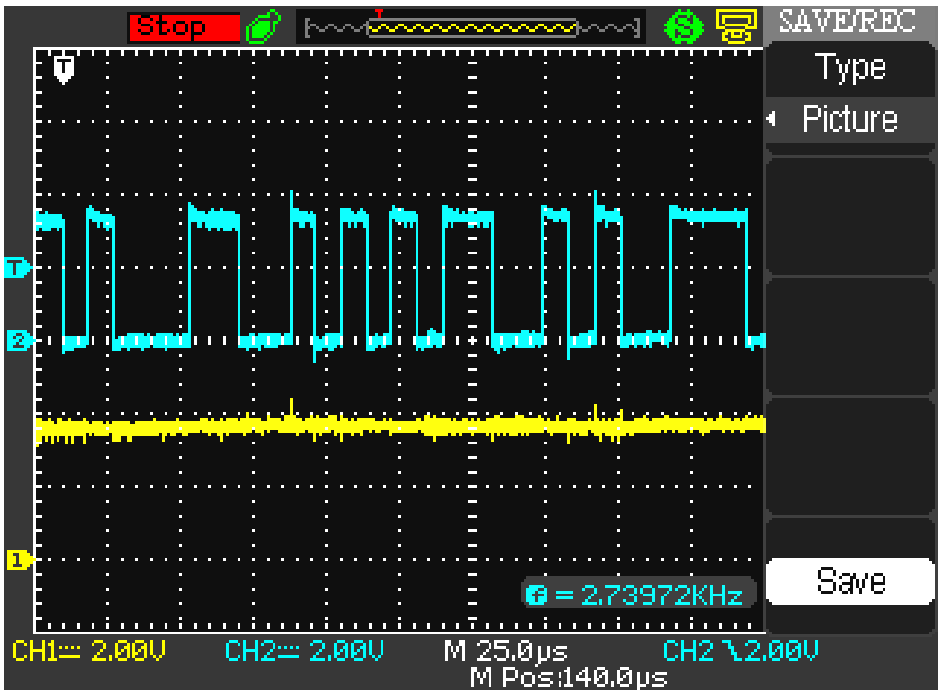
Kuva 7. Sarjaportin data oskilloskoopilla nopeudella 57600 bits/s

Kun otetaan tästä ensimmäinen tavu, niin saadaan alla oleva kuva (Kuva 8). Kun lähdetään vasemmalta oikealla, niin saadaan 1000 1100. Tämä käännettynä on 0011 0001, joka on heksadesimaalina 0x31 ja ASCIIna ”1” eli lähetetyn merkkijonon ensimmäinen merkki. Kuvan 8 viimeinen nouseva reuna on lopetus bitin alku ja alussa oleva laskeva reuna on aloitus bitin alku.



Kuva 8. Sarjaportin datan ensimmäinen tavu

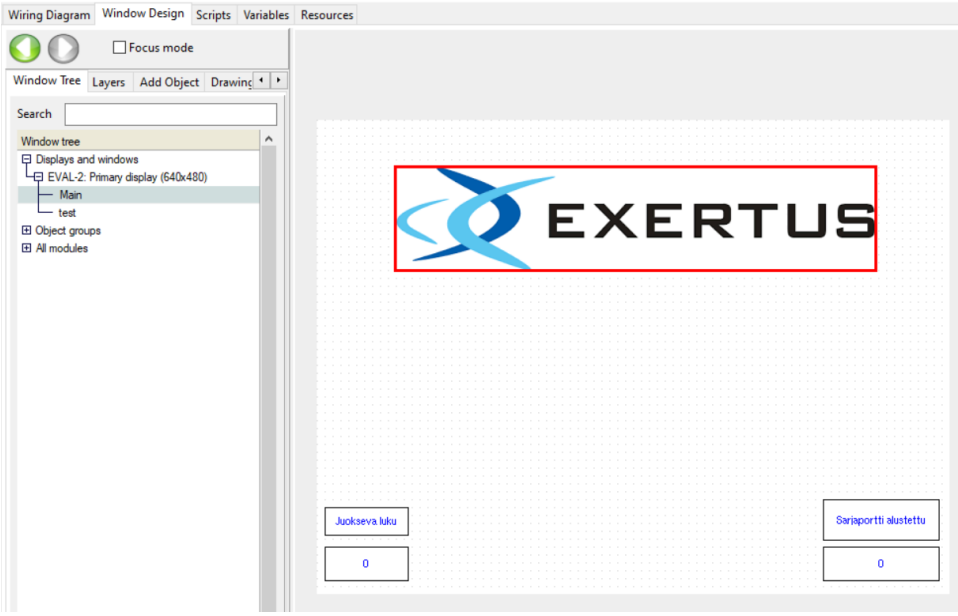
Kuvassa 9 näkyy samanlainen merkkijono kuin kuvassa 7. Tämä kuva on kuitenkin otettu nopeammalla sarjaportilla eli 115200 bittiä sekunnissa lähettävästä sarjaportista. Tämä nähdään siitä, että ruudun ajanjakso kuvassa 9 on 25 mikrosekuntia, kun kuvassa 7 se on 50 mikrosekuntia. Tämä kertoo sen, että sarjaportin nopeus on muutettu oikein, kun nopeus tuplaantuu, niin aika puolittuu.



Kuva 9. Sarjaportin data oskilloskoopilla nopeudella 115200 bits/s

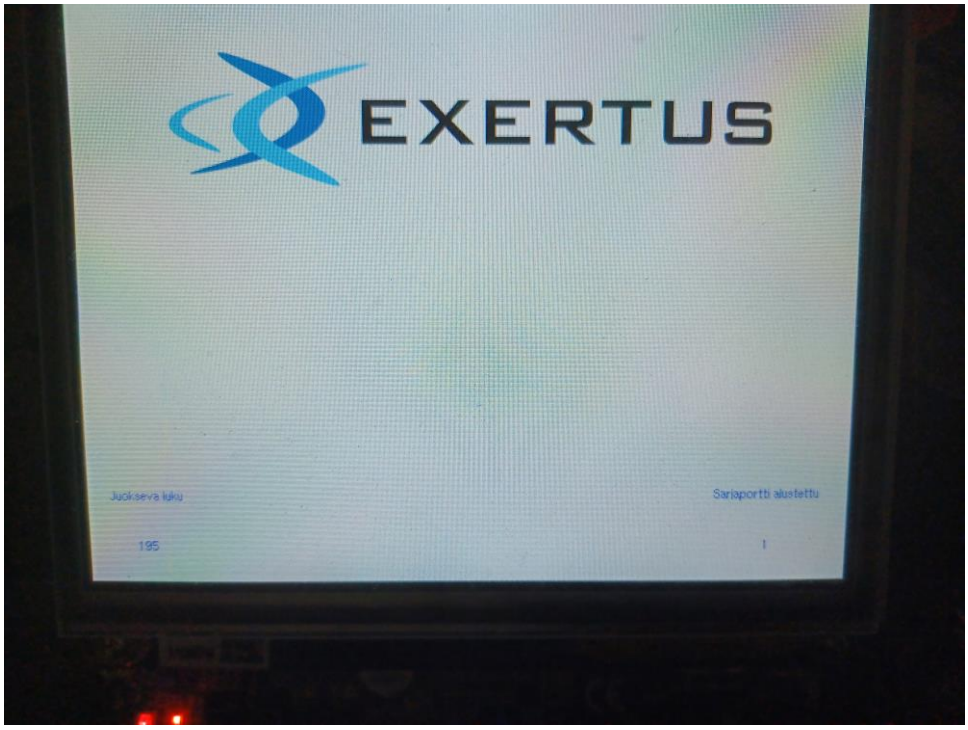
6.4 Graafisen käyttöliittymän testaus

Graafisen käyttöliittymän tekemiseen käytettiin Exertuksen Guitu-ohjelmaa. Tämä testattiin yksinkertaisella käyttöliittymällä. Kuva tästä alla (**Kuva 10**.). Tässä näkyy käyttöliittymän ensimmäinen ikkuna. Käyttöliittymän vasemmassa alareunassa on luku, jota laite pyörittää. Tällä testataan, että laite päivittää näyttöä oikein. Oikeassa alareunassa näkyy, onko sarjaportti alustettuna.



Kuva 10. GUITUlla tehty graafinen käyttöliittymä

Graafinen käyttöliittymä viedään laitteelleen CAN-väylän yli. Kuvassa 11 on kuva käyttöliittymästä kokeilukortin näytöllä. Tässä kuvassa sarjaportin alustuksen alla oikeassa alakulmassa näkyy luku 1. Tämä kertoo siitä, että sarjaportti on alustettu ja tämä on kuvassa 4 ja 5 näkyvän ”serial port” propertyn arvo. Vasemmassa alakulmassa näkyy juoksevaluku. Nämä toimivat oikein, koska näiden arvo ei näytä enää nollaa.



Kuva 11. Käyttöliittymä kokeilukortin näytöllä

7 LOPPUPÄÄTELMÄT JA JATKOKEHITYS

Tämän työn tavoitteena oli siirtää Exertuksen laiteohjelmisto STM32H753XI mikrokontrollerille. Mielestäni tähän päästiin STM32H753-EVAL2 kokeilukortilla ja saatiin testattua tarvittavia ominaisuuksia, joita kokeilukortilla pystyy testaamaan. Laitteiston alustukset ja tarvittavat funktiot tulevat kuitenkin vielä muuttumaan, kun työtä tullaan jatkamaan, kun saadaan uusista tuotteista prototyyppejä. Työn tuloksena saatiin kuitenkin tehtyä pohjaa tulevia tuotteita varten.

Työ oli osittain haastavaa, koska mikrokontrolleri oli minulle tuntematon. STM:n ohjelmistotyökalut olivat myös sellaisia, joita en ollut aikaisemmin käyttänyt. Työssä kului paljon aikaa tutustuessa uuteen laitteistoon sekä uusiin työkaluihin, mutta työ kuitenkin eteni hyvin, kun näihin sai tutustua.

Jatkokehityksessä työtä tullaan jatkamaan tuotteiden prototyypeillä. Prototyypeillä päästään kehittämään laitteiden lähtöjen ohjausta sekä tulojen lukemista ja näiden tietojen välittämistä CAN-väylän yli. Uusille tuotteille voi myös tulla ominaisuuksia, joita kokeilukortilla ei ole. Tällaisia ominaisuuksia voi olla, vaikka useampi CAN-väylä ja erilaisia antureita.

LÄHTEET

Arm, 2023. About the Cortex-M7 processor and core peripherals. Viitattu 10.4.2023. https://developer.arm.com/documentation/dui0646/c/Introduction/About-the-Cortex-M7-processor-and-core-peripherals?lang=en

Arm, 2020. Arm Cortex-M7 Processor Datasheet.pdf. Viitattu 10.4.2023. https://developer.arm.com/documentation/102838/0100

Exertus, 2015. Canto2 Monitoring and commissioning tool. Viitattu 10.5.2023. https://exertusoy.sharepoint.com/:b:/s/exefiles/EbAxsb9dS41Hk9yXxJg-Mz8BxDJpR_8bweTYQn36gNy_zQ

Exertus, 2015. GUITU Programming environment. Viitattu 10.5.2023. https://exertusoy.sharepoint.com/:b:/s/exefiles/EWiTfvlkX9xClzEk_GNH6aEBO23ahanf_f6Ajt35CZbhlA

ST, 2022. Description of STM32H7 HAL and low-layer drivers. Viitattu 10.5.2023. https://www.st.com/resource/en/user_manual/um2217-description-of-stm32h7-hal-and-lowlayer-drivers-stmicroelectronics.pdf

ST, 2023. STM32H753XI Datasheet.pdf. Viitattu 10.4.2023. https://www.st.com/resource/en/datasheet/stm32h753xi.pdf

ST, 2023. STM32H753XI Overview. Viitattu 10.4.2023. https://www.st.com/en/microcontrollers-microprocessors/stm32h753xi.html