Aarhus School of Engineering

Electronic Engineering

Projekt

Enhedstest

Author:
Nicolai Glud
Johnny Kristensen
Rasmus Lund-Jensen
Mick Holmark
Jakob Roesen

16. december 2012

Indholdsfortegnelse

Indledning 1

Dette dokument specificerer enhedsstesten af projektet BROS.

Versionshistorik

Ver.	Dato	Initialer	Beskrivelse
1.0	25-11-2012	NG	Oprettet

1.0.1 Formål

Dokumentet specificerer enhedstests og vil i udfyldt stand udgøre enhedstestdokumentationen
Testdelen af udviklingsprocessen er opdelt i tre faser:

- Enhedstest:
  Dette omfatter test af de enkelte funktioner implementeret i komponenter og klasserne (modulerne), som produktet bestående af hardware og software er sammenstykket af.
- Integrationstest:
  Dette omfatter test af grænseflader mellem komponenter og klasser (moduler), der indgår i det samlede system eller produkt. Det er altså samspillet mellem de moduler der er testet i enhedstesten.
- Accepttest:
  Dette omfatter en samlet test af funktionelle krav fra kravspecifikationen for hele systemets funktionalitet.

Testtproceduren er udviklet i rækkefølgen accepttest → integrationstest → enhedstest jvf. V-modellen.
Dette dokument omhandler testniveau 1 - enhedstesten.
Væsentlige ændringer i enhedstesten beskrives i dokumentets versionshistorie.

1.0.2 Referencer

1. Detaljeret hardware design
2. Detaljeret software design

1.0.3 Omfang

Denne enhedstest undersøger de forskellige modulers funktionalitet. Testen ligger forud for integrationstesten da vi sikre at modulet fungere inden vi sætter moduler sammen. Testen laves da det er vigtigt at moduler ikke udsender signaler der kan skade andre moduler eller ødelægge funktionalitet i programmer.

1.0.4 Godkendelseskriterier

Godkendelsen af systemtesten består af to trin:

- Godkendelse af enhedstestspecifikationen
  Dette gøres på forsiden af dokumentet i "Godkendt af" feltet.
- Godkendelse af selve enhedstesten. Dette gøres i afsnit Testresultat

Enhedstesten er afsluttet, når alle de i afsnit Testprocedure specificerede testcases er gennemført og godkendt.
Hvis der under integrationstesten opstår fejl, der umuliggør fortsat udførsel af de efterfølgende testcases afbrydes denne test.
Såfremt en test afbrydes eller et testcase underkendes, skal problemet undersøges og for så vidt muligt løses. Dette skal dokumenteres i loggen.

Test 2

I dette afsnit følger selve testen.

2.1 Testcases

Dette afsnit er delt op i 2 dele. Hardware og software:

2.1.1 Hardware

I dette afsnit forklares hvordan enhedstest af hardware udføres.

SM

Case	Formål	Udførelse
1	Indstil Accelerometer	Der skrives 0x06 på P15[2:0]
2	Verificer RS232	Der sættes en jumper mellem RX_out og TX_out og derefter sendes der 5 forskellige chars via UART

VBTE

Case	Formål	Udførelse
1	At teste ventilkreds	Der toggles 5V med 500ms interval ud fra PSoC'en på ben P0_0 og P0_2. Der lyttes på ventilerne for at bekræfte at de åbner og lukker.
2	At teste transmitterkreds	Der sendes et 40kHz signal fra funktionsgeneratoren med 12Vp-p. Der læses med oscilloskop på receiveren og det valideres om signalet bliver transmitteret.
3	At teste receiverkredsen	Der sendes burst fra transmitterkredsen med 500ms interval med en varighed på $\sim$250μs. Der indsættes to analoge pinde i PSoCdesignet der kobles til udgangen på PGA'en og udgangen af mixeren. Signalet valideres med oscilloskopbilleder.

2.1.2 Software

I dette afsnit forklares hvordan enhedstests af hardware udføres.

SM

Case	Formål	Udførelse
1	GetLevel	GetLevel kaldes som funktionskald med en stub. Stubben verificere returnværdien.
2	getFromKI	Et program køres hvor getFromKI kaldes i en while løkke. SM modulet sættes sammen med en teststub der sender 6 forskellige cases, 1000 gange.

VBTE

Case	Metode	Udførelse
1	SendBurst	Metoden kaldes i intervaller på 500ms og der måles med osciloskop på ben P0_1 at der bliver sendt burst's med en varighed på ∼250μs og med en frekvens på ∼40kHz.
2	CalculateDistance	Metoden kaldes 100 gange med forskellige input-værdier. Outputtet ligges i et array og der valideres på disse værdier.
3	ConvertMMtoPercent	Metoden kaldes 100 gange med forskellige input-værdier. Outputtet ligges i et array og der valideres på disse værdier.
4	ChangeState	Metoden kaldes med alle forskellige slags input og 3 værdier uden for input. Der lyttes på ventilerne og der valideres om de åbner/lukker som de skal.
5	I2C_handle	Der anvendes en stub der agerer som SM. Denne sender alle værdier fra protokollen samt 3 værdier uden for protokollen. Der kontrolleres om der modtages alle værdier korrekt ved at udskrive dem på displayet. Der kontrolleres også om den rigtige værdi sendes retur til SM stub'en.
6	I2C_decode	Metoden kaldes med de forskellige værdier for protokollen samt 3 uden for protokollen. Returværdien kontrolleres for at validere det korrekte state.
7	Init	Metoden kaldes og der kontrolleres om der returneres 1 tilbage.

KI

Tabel 2.1. "AKTIVER MANUEL HÆLDNINGSREGULERING"-knappen

Case	Formål	Udførelse
1a	At teste hvorvidt en ændret manuel vinkling sendes ud serielt.	Der indsættes en manuel vinklingsregulering på den grafiske brugergrænseflade. Alle kombinationer af side og værdi afprøves. Der verificeres i terminalen at RS232-klassen udsender værdierne til SM.
1b	Det testes hvordan programmet reagerer hvis man efter at have trykket på "AKTIVER MANUEL HÆLDNINGSREGULERING"fortryder sit valg ved tryk på "Cancel-knappen.	Tryk på knappen. Tryk på "Cancel".
1c	Det testes hvordan programmet reagerer hvis der ingen forbindelse er til SM når man forsøger at sætte en manuel hældning.	Tryk på "AKTIVER MANUEL HÆLDNINGSREGULERING"uden forbindelse til SM. Bekræft værdier. Aflæs GUI'en og terminalen.

Tabel 2.2. Opdatering af grafisk brugergrænseflade

Case	Formål	Udførelse
2	At teste hvorvidt en status struct kan requestes fra SM-klassen og sendes til databasen. SM-klassen returnerer en status-stub. Det verificeres i terminalen at dataserver-klassen udsender værdierne til databasen.	Start programmet. Vent til GUI'en opdateres. Aflæs terminalen.

Tabel 2.3. "AKTIVER AUTOMATISK HÆLDNINGSREGULERING"-knappen

Case	Formål	Udførelse
3a	Det testes hvordan programmet reagerer hvis man forsøger at aktivere automatisk regulering, når denne allerede er aktiveret.	Automatisk hældningsregulering skal ikke være aktiveret. Tryk på "AKTIVER AUTOMATISK HÆLDNINGSREGULERING". Aflæs terminalen og GUI'en.
3b	Det testes hvordan programmet reagerer hvis man forsøger at aktivere automatisk regulering, når denne ikke er aktiveret.	Der trykkes på knappen "AKTIVER AUTOMATISK HÆLDNINGSREGULERING". Tryk på "YES". Aflæs terminalen og GUI'en.
3c	Det testes hvordan programmet reagerer ved tryk på "AKTIVER AUTOMATISK HÆLDNINGSREGULERING"når der ingen forbindelse er til SM.	Tryk på knappen "AKTIVER AUTOMATISK HÆLDNINGSREGULERING". Tryk på "YES". Aflæs GUI.

Tabel 2.4. "LUK BROS"-knappen

Case	Formål	Udførelse
4a	Det testes hvordan programmet reagerer hvis man ønsker at lukke programmet med et tryk på "LUK BROS-knappen og efterfølgende bekræfter ved tryk på "YES-knappen.	Tryk på "LUK BROS-knappen. Tryk på "YES". Aflæs terminalen og GUI'en.
4b	Det testes hvordan programmet reagerer hvis man efter tryk på "LUK BROS-knappen fortryder sit valg ved at trykke "NO".	Tryk på "LUK BROS-knappen. Tryk på "NO". Aflæs terminalen og GUI'en.

2.2 Testresultater

Dette afsnit er delt op i 2 dele baseret på ovenstående tests.

2.2.1 Hardware

I dette afsnit findes forventede resultater samt resultater på testcases fra ovenstående hardware kapitel.

SM

Case	Forventet resultat	Resultat	Status
1	Accelerometeret er indstillet.	Det observeres at accelerometeret er indstillet og aktivt.	√
2	De afsendte chars bliver modtaget via UART.	De afsendte chars blev modtaget via UART.	√

VBTE

Case	Forventet resultat	Resultat	Status
1	Ventilerne åbner og lukker	Det høres tydeligt at ventilerne åbnes og lukkes.	√
2	Der ses signal på osciloskopet	Signalet ses på osciloskop. Se *figur 3.6*	√
3	Der ses burst efter PGA'en samt "tapper"efter mixeren via. oscilloskop.	Ved første test blev et markant svagere end antaget modtaget. Gain i PGA blev justeret til 32 og testen kunne godkendes. Testresultet ses på *figur 3.1, 3.2 og 3.3* i bilag.	√

2.2.2 Software

I dette afsnit findes forventede resultater samt resultater på testcases fra ovenstående software kapitel.

SM

Case	Forventet resultat	Resultat	Status
1	Level bliver returneret og verificeret	Level blev returneret og verificeret	√
2	teststubben printer til skærmen at alle cases er succesfulde	teststubben printede Success: 6000	√

VBTE

Case	Forventet resultat	Resultat	Status
1	Metoden laver et burst på ∼250μs og med en frekvens på ∼40kHz	Der er blevet målt med osciloskop på P0_1. Se resultat på *figur 3.5 og 3.4* i bilag.	√
2	Metoden returnerer 100 værdier der stemmer overens med funktionaliteten	Metoden returnerede de forventede værdier.	√
3	Metoden returnerer 100 værdier der stemmer overens med funktionaliteten	Metoden returnerede de forventede værdier.	√
4	Metoden togler ventilerne som forventet	Ventilerne blev toglet som forventet.	√
5	Metoden udskriver værdierne på displayet og svarer SM stub'en	Der blev aflæst det forventede på displayet og svaret til SM stemte overens med forventningerne.	√
6	Metoden returnerer de forventede resultater og returnerer luk ventiler ved værdier uden for protokollen	Metoden returnerede de forventede værdier.	√
7	Metoden returnerer 1	Metoden returnerede 1.	√

KI

Tabel 2.5. "AKTIVER MANUEL HÆLDNINGSREGULERING-knappen

Case	Forventet resultat	Resultat	Status
1a	I terminalen aflæses det at valget er bekræftiget og at RS232-klassen udsender værdien for kommandoen og dernæst hældningen i overensstemmelse med protokollen[1]. I programmet kan det aflæses hvilken værdi der manuelt er indstillet til	Resultatet kan ses i **??** og stemmer overens med forventningerne.	√
1b	Programmet vender tilbage til stadiet før det første tryk på "AKTIVER MANUEL HÆLDNINGSREGULERING"og trykket har ingen konsekvenser.	Programmet foretog sig intet i relation til trykket. GUI'en er uændret.	√
1c	Det samme som 1b.		√

Tabel 2.6. Opdatering af grafisk brugergrænseflade

Case	Forventet resultat	Resultat	Status
2	I terminalen udskrives status-structstubben i SM-klassen. Den udskrives efterfølgende igen af dataserverklassen som den sendes til databasen. Her sendes navnet på skibet og tiden siden sidste opdatering fra SM. Disse er tilføjet Kontrolinterfaceklassen.		

Tabel 2.7. "AKTIVER AUTOMATISK HÆLDNINGSREGULERING-knappen

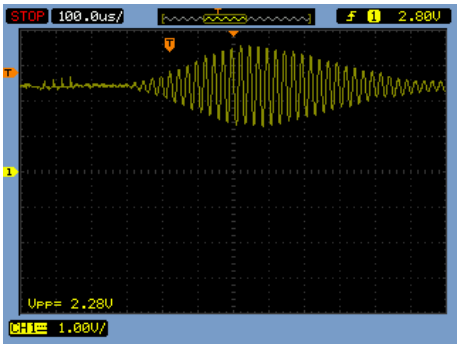
Case	Forventet resultat	Resultat	Status
3a	Det forventes at programmet bringer en dialog op hvori der informeres om at denne reguleringstype allerede er aktiveret.	Programmet reagerede blot med dialogen. [2].	√
3b	Det forventes at der popper en dialog frem hvor der skal bekræftiges at man ønsker at gå væk fra manuel hældning. Ved bekræftelser udskrives det af RS232-klassen[3] at kommandoen er sendt. Ved tryk på "NO"lukker dialogen og trykket har ingen videre konsekvens.	Dialogen kom frem og kan ses på figur [4]	√
3c	Det forventes at der poppe en dialog op som i 3b, men at der ved tryk på "YES"intet sker i GUI'en, da aktiveringen ikke bekræftiges af SM.	Programmet agerede som forventet.	√

Tabel 2.8. "LUK BROS-knappen

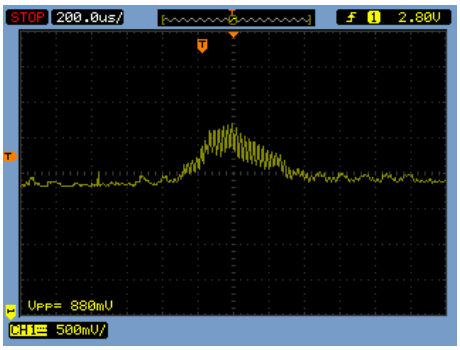
Case	Forventet resultat	Resultat	Status
4a	Det forventes at programmet sender protokolkorrekte termineringskoder til både databasen og styringsmodulet og herefter lukker ned. Hvis programmet ikke får et svar fra styringsmodulet afbrydes termineringen med en dialog med teksten: Ingen kontakt til Styringsmodulet. Af sikkerhedsmæssige årsager kan programmet ikke lukkes".	Programmet kunne ikke lukkes ned. Se Integrationstesten for test af korrekt termineringen af programmet.[5]	√
4b	Det forventes at programmet blot vender tilbage til stadiet før trykket på "LUK BROS"uden yderligere handling.	Programmet vende korrekt tilbage og foretog sig intet yderligere i forhold til trykket.	√

Bilag 3

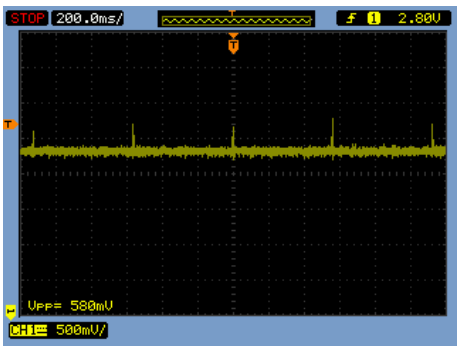
3.1 VBTE hardware



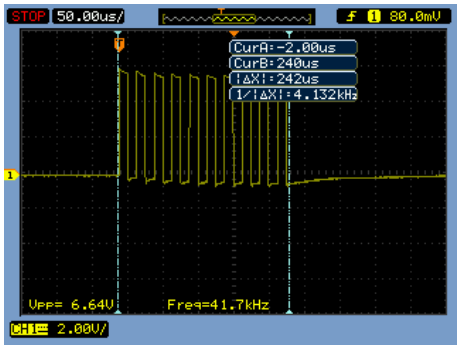
Figur 3.1. Burst set mellem PGA og mixer



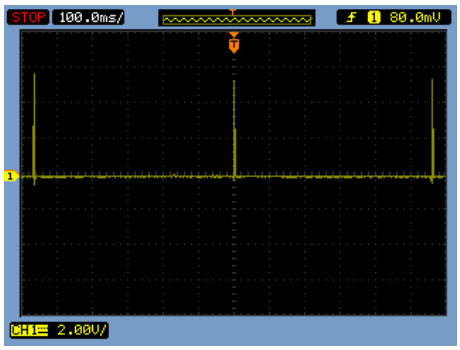
Figur 3.2. Burst set mellem mixer og delta-sigma



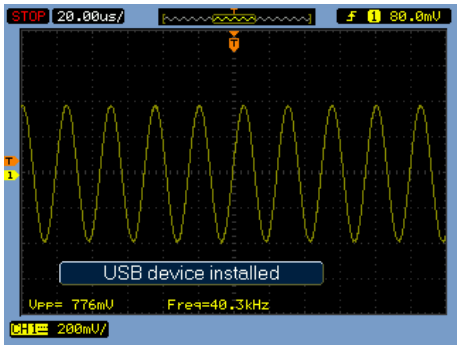
Figur 3.3. gentagene burst set mellem mixer og delta-sigma



Figur 3.4. Burst sendt fra PSoC'en



Figur 3.5. gentagene burst sendt fra PSoC'en



Figur 3.6. 40kHz signal fra transmitter set på receiver

3.2 Test kode til VBTE

```
1  //===============================================================
2  // ENHED        : VBTE
3  // CASE ID      : 1
4  // BESKRIVELSE : SendBurst skal testes for at kontrollere
5  //          bredden af burst's. Kontrol via oscilloskop.
6  //===============================================================
7
8  CyDelay(500);
9  SendBurst();
10
11 //===============================================================
12 // ENHED        : VBTE
13 // CASE ID      : 2
14 // BESKRIVELSE : CalculateDistance skal testes for at
      kontrollere
15 //          udregningen.
16 //===============================================================
17
18 int i;
19 int n = 100;
20 double result[n] = 0;
21 static uint32 BurstTimerVal = 0;
22 static uint32 DistanceTimerVal = 0;
23 for(i = 0; i < n; i++){
24   BurstTimerVal = 1000*i;
25   DistanceTimerVal = 1500*i;
26   result[i] = CalculateDistance();
27 }
28
29 //===============================================================
30 // ENHED        : VBTE
31 // CASE ID      : 3
32 // BESKRIVELSE : ConvertMMtoPercent skal testes for at
      kontrollere
33 //          udregningen.
34 //===============================================================
35
36 int i;
37 int n = 100;
38 uint8 result[n] = 0;
39 float distMM = 0;
40 for(i = 0; i < n; i++){
41   distMM = 2.5*i;
42   result[i] = ConvertMMtoPercent(distMM);
```

```
43 }
44
45 //=================================================================
46 // ENHED        : VBTE
47 // CASE ID      : 4
48 // BESKRIVELSE : ChangeState testes for om ventilerne
      aabner/lukker
49 //=================================================================
50
51 uint8 state[7] = {0,1,2,3,4,5,6};
52 int i;
53 int n = 7;
54 for(i = 0; i < n; i++){
55   ChangeState(state[i]);
56   CyDelay(500);
57 }
58
59 //=================================================================
60 // ENHED        : VBTE
61 // CASE ID      : 5
62 // BESKRIVELSE : I2C_handle kaldes og der ses paa displayet
      om de
63 //          rigtige vaerdier udskrives paa displayet.
64 //=================================================================
65
66 uint8 BufferSize = 8;
67 uint8 ReadBuffer[BufferSize];
68 uint8 WriteBuffer[BufferSize];
69 uint8 DistancePercent = 23;
70 while(1){
71   I2C_handle( WriteBuffer, ReadBuffer, BufferSize,
        DistancePercent );
72 }
73
74 //=================================================================
75 // ENHED        : VBTE
76 // CASE ID      : 6
77 // BESKRIVELSE : I2C_decode testes med 7 vaerdier. 4 som
      protokollen
78 //          foreskriver og 3 uden for protokollen.
      Returvaerdien
79 //          kontrolleres.
80 //=================================================================
81
82 uint8 BufferSize = 8;
83 uint8 ReadBuffer[BufferSize];
```

```
84 uint8 WriteBuffer[BufferSize];
85 uint8 DistancePercent = 23;
86 while(1){
87   I2C_handle( WriteBuffer, ReadBuffer, BufferSize,
       DistancePercent );
88 }
89
90 //================================================================
91 // ENHED        : VBTE
92 // CASE ID      : 6
93 // BESKRIVELSE : I2C_decode testes med 7 vaerdier. 4 som
       protokollen
94 //          foreskriver og 3 uden for protokollen.
       Returvaerdien
95 //          kontrolleres.
96 //================================================================
97
98 uint8 read[7] = {0,1,2,3,4,5,6};
99 uint8 result[7] = 0;
100 int i;
101 int n = 7;
102 for(i = 0; i < n; i++){
103   result[i] = I2C_decode(read[i]);
104   CyDelay(500);
105 }
106
107 //================================================================
108 // ENHED        : VBTE
109 // CASE ID      : 7
110 // BESKRIVELSE : Init testes ved at kontrollere at 1 bliver
       returneret
111 //================================================================
112
113 uint8 BufferSize = 8;
114 uint8 ReadBuffer[BufferSize];
115 uint8 WriteBuffer[BufferSize];
116 uint8 result = 0;
117 result = init( ReadBuffer, WriteBuffer, BufferSize);
```

Test/test_VBTE.c