Prototype 11 Meerdere meetkasten die configureerbaar zijn en meetwaardes versturen » History » Version 69

Version 65 *(Tan Hoang, 05/31/2024 04:31 PM)* → Version 69/172 *(Lars van Duijnhoven, 05/31/2024 04:48 PM)*

h1. Prototype 11 Meerdere meetkasten die configureerbaar zijn en meetwaardes versturen

!https://sasscm.han.nl/svn/intoft25_2023_intoft25_p4_01/Afbeeldingen/hanlogo.png!

Meerdere meetkasten die configureerbaar zijn en meetwaardes versturen - IoT Project - HBO ICT - 2024

Klas: ITN-IOT-A-f
Groepsnaam: Elektronische Levens Hulp (ELH)
Course: IoT Project
Versienummer: 1.0
Locatie: Nijmegen
Docent(en):
Eddy Luursema (Opdrachtgever)
Nils Bijleveld (Professional Skills)
Chris van Uffelen (Procesbegeleiding)
Student(en):
J.J.A. Visch (2104400)
N.T. Hoang (2112106)
R. Jurrius (631340)
L.P.W van Duijnhoven (2103948)
C.G.M van Kempen (2107890)
Datum: 29 mei 2024

h2. Inhoudsopgave

{{toc}}

h2. Inleiding

Om te laten zien dat ons systeem flexibel is en configurabel is, gaan we meerdere meetkasten/microcontrollers aansluiten. We hebben nu maar 1 meetkast die alles doet, maar dit zouden we dus willen uitbreiden. Verder willen we bij dit prototype ook integreren dat de configuratietabellengte dynamisch wordt verstuurd. Anders zou je steeds moeten aanpassen in de meetkast code.

h2. Scenario

h3. Huidige situatie:

In de Mobiele Medische Unit wordt er maar nu 1 meetkast gebruikt. De meetkast krijgt nu de hele configuratietabel binnen. In de meetkast is ingesteld na hoeveel rijen configuratie dat er een stop bericht wordt verstuurt aan de centrale unit zodat die stopt met configuratie gegevens sturen. Daarna kijkt de meetapparaat in de configuratietabel welke rij voor zijn eigen meetkast is bedoelt en stuurt die meetgegevens.

h3. Toekomstige situatie:

In de Mobiele Medische Unit worden er twee meetkasten gebruikt, zodat er niet 1 meetkast overbelast wordt. De meetkast krijgt de hele configuratietabel doorgestuurd, zodat deze een andere meetkast over kan nemen, indien deze kapot is gegaan voor welke reden dan ook. De centrale unit stuurt de configuratietabel over de CAN en stopt pas met versturen zodra alle meetkasten zijn geconfigureerd. De meetkasten kunnen wel alvast beginnen met meetgegevens sturen zodra er eentje is geconfigureerd zonder dat de andere is geconfigureerd.

!https://sasscm.han.nl/svn/intoft25_2023_intoft25_p4_01/Prototype11_Meerdere_Meetkasten/Afbeeldingen/Prototype_11_Scenario.jpg!

h2. Concerns

* Werkt de can bus wel als je 3 can modules aansluit?
* Hoe weet de centrale unit wanneer die moet stoppen met het versturen van de configuratietabel naar de meetapparaten?
* Hoe controleren we wanneer een configuratietabel helemaal is verstuurd?
* Hoe gaan we de configuratietabel in ons geheugen zetten?
* Is onze code zo aanpasbaar dat de 2e meetkast toe te voegen is?

h2. Logboek + Fasering

Kleur	Betekenis
%{color:red}Rood%	New
%{color:orange}Oranje%	In progress
%{color:purple}Paars%	Ready to review
%{color:lime}Groen%	Done

29-05

!https://sasscm.han.nl/svn/intoft25_2023_intoft25_p4_01/Prototype11_Meerdere_Meetkasten/Afbeeldingen/Planningsgraven/Planningsgraaf29-05.png!

30-05
We hebben vandaag besloten dat we eerst issue 5 en 6 doen en daarna pas 3 en 4. Dit komt omdat we beter het tabel lengte eerst implementeren en daarna de stop berichten.
!https://sasscm.han.nl/svn/intoft25_2023_intoft25_p4_01/Prototype11_Meerdere_Meetkasten/Afbeeldingen/Planningsgraven/PlanningsgraafPrototype1130-05.png!

31-05
!https://sasscm.han.nl/svn/intoft25_2023_intoft25_p4_01/Prototype11_Meerdere_Meetkasten/Afbeeldingen/Planningsgraven/PlanningsgraafPrototype1131-05.png!

Hoofdissue: #13541

Nummer van planningsgraaf	Link naar issue+Logboek(Comments in issue)
1.	#13542
2.	#13543
3.	#13544
4.	#13546
5.	#13554
6.	#13555
7.	#13556
8.	#13557
9.	#13558

h2. Onderzoeksverslag

h4. Gebruikte hardware

Opstelling:

!https://sasscm.han.nl/svn/intoft25_2023_intoft25_p4_01/Prototype11_Meerdere_Meetkasten/Opstelling/2meetkasten.png!

h4. Gebruikte software

Hier een overzicht voor welke software op welke microcontroller hoort.

h5. Voor de centrale unit/esp32:
https://sasscm.han.nl/!/#intoft25_2023_intoft25_p4_01/view/head/Prototype11_Meerdere_Meetkasten/Code/ESP32_MMU

h5. Voor de meetkast1/arduino uno en meetkast2/arduino uno:
https://sasscm.han.nl/!/#intoft25_2023_intoft25_p4_01/view/head/Prototype11_Meerdere_Meetkasten/Code/MeetKastConfig

Zorg ervoor dat je voor de meetkasten bij de file can.h dat je de variable MEASUREUNITID verandert naar 1 of 2.

h5. benodigde libraries die je moet downloaden:
https://github.com/autowp/arduino-mcp2515

h4. Bevindingen/resultaten

h4. Conclusie

We gaan hier elke concern langs en geven aan of het ons is gelukt om een oplossing te vinden voor deze concern.

h5. Werkt de can bus wel als je 3 can modules aansluit?

We hebben eerst een tweede arduino toegevoegt met een can module. Zo hebben 2 arduino's en 1 esp32 nu een can module die zijn aangesloten op de can bus.
Daarna hebben we test code gemaakt om te kijken of we de configuratie op beide arduino's kregen via de can bus.

Hier de gebruikte code: https://sasscm.han.nl/!/#intoft25_2023_intoft25_p4_01/view/head/Prototype11_Meerdere_Meetkasten/Code/MeerdereMeetkastCanTest

Dit is ons gelukt, hier het resultaat:
!https://sasscm.han.nl/svn/intoft25_2023_intoft25_p4_01/Prototype11_Meerdere_Meetkasten/Afbeeldingen/2tegelijkontvangen.png!

Verder hadden we nog de vraag hoe het bericht nu uitgelezen werd, want de ID's zijn wel hetzelfde die we binnenkregen.
Dit hebben we dus onderzocht en zijn a.d.h.v. "deze":https://www.youtube.com/watch?v=oYps7vT708E video (vooral bij vanaf tijdstip 1:40) te weten gekomen hoe dit werkt.
Zoals de video uitlegt is dat elke module die aangesloten is op de bus het bericht binnenkrijgt en dat er dan besloten mag worden door diezelfde module of er iets mee gedaan moet worden of niet.

h5. Hoe controleren we wanneer een configuratietabel helemaal is verstuurd?

Ons idee hiervoor was om de CU (Centrale Unit) alleen maar 11-bit identifiers te laten versturen, zodat de meetkast alleen maar 29-bit identifiers kan gebruiken.
Hierdoor kan de meetkast alleen naar 11-bit identifiers luisteren en de CU alleen naar 29-bit identifiers.
Deze keuze is gemaakt, omdat je anders het probleem zou krijgen van hoe onderscheidt je een config bericht van een measurement van een andere meetkast?
Daarnaast heeft het als voordeel dat we nu de stop messages prioriteit kunnen geven, maar dat wordt in het kopje hieronder verder uitgelegd.
We hebben besloten om 11-bit voor de config messages te gebruiken, omdat deze dan voorrang hebben over de measurements. Als je namelijk een patiënt juist heel goed in de gaten wilt houden, maar je moet 30 sec wachten omdat er eerst allemaal metingen verstuurd worden wordt je niet heel blij. Of ja, vooral de patiënt zelf dan waarschijnlijk niet.
Kortom, hierdoor weten we zeker dat de configuratie berichten verstuurd worden vóór de meetwaardes.

Om te controleren wanneer een configuratietabel helemaal is verstuurd wilden we dan aan het begin (dus voordat de configtabel verstuurd wordt) een bericht met 11-bit identifier "0" versturen. Als data hebben we dan de hoeveelheid berichten die ontvangen moeten worden om te zorgen dat de tabel gevuld is, dus gewoon aantal rows . Dit kan max 2047 zijn, vanwege het 11 bits limiet.
Als de hoeveelheid ontvangen berichten overeen komt met de hoeveelheid die het zou moeten zijn wordt de stop message gestuurd. Hier wordt in het kopje hieronder meer over uitgelegd.
Zie de foto en uitleg hieronder voor een voorbeeld.

!https://sasscm.han.nl/svn/intoft25_2023_intoft25_p4_01/Prototype11_Meerdere_Meetkasten/Afbeeldingen/Screenshot_resultaat_rows_en_stopmessage.png!

Je kan hier zien dat er een bericht verstuurd wordt met 11-bit CAN-ID-Rows 0. Deze bevat 4 bytes die samen voor een long zorgen als dit samegevoegd wordt door een union.
In dit voorbeeld was het maximale aantal rows maar 14 en dit kan je dus ook zien in de eerste byte.
Dit bericht wordt vóór het versturen van de configuratieberichten verstuurd, dus zolang er configuratieberichten verstuurd worden, zullen er ook config-rijen berichten verstuurd worden.

Hier zie je hoe de meetkast het getal 14 ontvangt

!https://sasscm.han.nl/svn/intoft25_2023_intoft25_p4_01/Prototype11_Meerdere_Meetkasten/Afbeeldingen/2tegelijkTabelLengteOntvangen.png!

Het lijkt alsof er nu 2 berichten zijn met CAN-ID 0 wordt ontvangen, maar dat is niet zo. Het eerste bericht is degene met daadwerklijk CAN-ID 0. De volgende CAN-ID 0, is eigenlijk 1, maar om onze configuratie tabel sortering goed te doen, moeten we -1 doen.

We hebben het zo geïmplementeerd dat de meetkast niet hoeft te wachten totdat die de tabellengte heeft ontvangen, maar dat die gewoon elke punt van de configuratiebericht kan lezen. Dus stel je nou voor dat de centrale unit al was begonnen met configuratie berichten versturen, maar de meetkasten waren nog niet aan het lezen, leest die in plaats van config bericht 1, leest die config bericht 20. Dit is niet erg, want dan zet de meetkast die waardes alvast in de tabel en stuurt die pas een stop message als de tabel lengte is ontvangen(en als de ontvangen tabel lengte overeenkomt met de ontvangen configuratie).

We wilde eerst dat de meetkast begon met luisteren naar configuratie berichten als die eerst de tabel lengte heeft ontvangen, maar dat is niet zo efficiënt aangezien de kans bestaat dat die net niet de tabel lengte heeft ontvangen en moet wachten op de volgende iteratie wanneer de configuratie opnieuw wordt verstuurd.

h5. Hoe weet de centrale unit wanneer die moet stoppen met het versturen van de configuratietabel naar de meetapparaten?

Zoals hierboven stond, houden we dus de 11- en 29-bit identifiers apart voor de meetkast en CU (Centrale Unit).
Om de CU te laten weten dat een meetkast klaar is maken wij gebruik van de 2047 beschikbare bits. Deze zijn 29 bits, maar wel met een hogere dezelfde prioriteit dan als de meetwaardes, 11 bits, omdat deze stop bericht canID's lager zijn dan de meetwaarde bericht ID's. Bijvoorbeeld 4 (stop bericht meetkast 4) heeft een hogere prioriteit dan 2052 (meetwaarde 4). Dit komt doordat CAN-id rond dezelfde waardes hangt (en niet ergens rond de meetwaardes pas beginnen bij 2048, zodat deze gescheiden zijn van de stop berichten. 1000000 ofzo).
Als meetkast 1 klaar is, stuurt deze de stop message met als ID zijn moduleID, dus in dit geval 1. De CU kan dan bijhouden hoeveel messages hij hiervan binnenkrijgt en of iedereen dan klaar is of niet. Hieronder een voorbeeld/ demonstratie.

In deze foto zie je 2 seriële monitoren, beide zijn aangesloten op 1 Arduino.

!https://sasscm.han.nl/svn/intoft25_2023_intoft25_p4_01/Prototype11_Meerdere_Meetkasten/Afbeeldingen/2tegelijkontvangen+versturen.png!

Wat hier vooral interessant aan is is dat je hier echt kan zien dat op beide Arduino's een apart stop bericht verstuurd wordt. Namelijk een met ID 1 en eentje met ID 2. Dit wordt pas verstuurd nadat de tabel volledig gevuld en geprint is, een gedeelte hiervan kan je ook zien. Namelijk alles tot de "STOP SENDING MESSAGE" hoort bij de tabel.
Hierna zie je dat de stop messages gestuurd worden met hun eigen ID. Deze ID's zijn kleiner dan 2048, zodat deze als stop message gezien worden en ook voorrang hebben op de measurement berichten.

h5. Hoe gaan we de configuratietabel in ons geheugen zetten?

We hebben nu gekozen om een vaste lengte te gebruiken voor onze configuratietabellen bij onze meetkasten (Arduino Uno's). De reden hiervoor is omdat we achter kwamen dat we eigenlijk niet zoveel configuratie rijen in ons geheugen kunnen zetten.

Als je kijkt naar onze configuratie tabel, stoppen we hier steeds een structure genaamd Sensor in.

```
typedef struct{
uint32_t measureUnit;
uint32_t canId;
uint32_t frequency;
uint32_t resolution;
uint32_t port;
PortType portType;
uint64_t previousMillis;
} Sensor;
```

Bijna elk datatype is 32 bit of hoger oftewel 32/8 = 4 bytes. 1 structure kost dan (4 * 6) + (8 * 1) = 32 bytes. Onze arduino heeft 2048 bytes aan ram.
Voor andere dingen die we in de ram zitten die niet te maken hebben met de configuratie tabel kost 504 bytes.

Dus als we het bereken kunnen we maar maximaal 2048 - 504 = 1544 / 32 = 48 rijen. We kunnen dus maar 48 rijen toevoegen. Om meer rijen toe te voegen hebben we gekeken of we de datatypes wat kleiner kunnen maken.

```
typedef struct{
uint8_t measureUnit;
uint32_t canId;
uint16_t frequency;
uint8_t resolution;
uint8_t port;
PortType portType;
uint64_t previousMillis;
} Sensor;
```

Door deze aanpassing kost een rij maar (1 * 3) + (2 * 2) + (4 * 1) + (8 * 1) = 19 byte's. We moeten ook nog rekening houden dat we een boolean tabel ook instellen met het aantal rijen, dus dan komt er nog 1 byte bij.

Nu kunnen we 1544/19 = 81 rijen gebruiken voor ons configuratie, maar onze opdrachtgever heeft aangegeven dat het niet veilig is om alle ram te gebruiken. We hebben nu gezegd dat we het ongeveer op 80% willen houden, maar dit zou na overleg met de opdrachtgever hoger/lager worden.

We zetten hem nu op 40 aangezien we niet zoveel config data versturen.

Dit is voor nu onze keuze om de configuratietabellengte hardcoded te doen. Het heeft voor ons nu geen waarde om dat dynamisch te laten doen, aangezien we er toch maar weinig rijen aan kunnen toevoegen en onze arduino mogelijk vast gaat lopen. Verder zouden we onze NASA regel verbreken om na initialisatie nog geheugen alloceren. Dit zou dan nog in een ander prototype opgepakt kunnen worden.

Dit betekent niet dat je nu 40 config moet sturen, we hebben wel dynamisch dat je het aantal rijen verstuurt die je gaat sturen. Dus als de configtabel lengte 40 lang is kan je gewoon dynamisch bijvoorbeeld 14 sturen en wacht de meetkast ook maar op 14 configuraties.

Verder als er in de MMU een microcontroller komt met meer ram, zou natuurlijk de tabellengte groter kunnen worden en eventueel de dynamische configuratietabellengte.

h5. Is onze code zo aanpasbaar dat de 2e meetkast toe te voegen is?

Voor 80% wel ja, we hebben vrij weinig meetkast code aan moeten passen voor de 2e meetkast.
Deze updates zijn "hier":https://sasscm.han.nl/redmine/projects/intoft25_2023_intoft25_p4_01/repository/diff/Prototype11_Meerdere_Meetkasten/Code/MeetKastConfig?utf8=%E2%9C%93&rev=294&rev_to=282 te vinden. Helaas moest aan de kant van de CU wel meer code aangepast worden, maar gelukkig bleef dit bij aanpassen en hoefden we geen volledige functies te herschrijven. De aanpassingen hiervoor zijn "hier":https://sasscm.han.nl/redmine/projects/intoft25_2023_intoft25_p4_01/repository/diff/Prototype11_Meerdere_Meetkasten/Code/ESP32_MMU?utf8=%E2%9C%93&rev=295&rev_to=283 te vinden.

We zijn langer bezig geweest met bedenken hoe we het aan zouden pakken dan de code aanpassen, omdat de code aanpassen redelijk vanzelf ging (na wat bug fixes).
Zoals we het nu hebben kan je met een const uint8_t (geen #define, want die wilde niet werken :/, hij gaf dan telkens dezelfde waarde als voorheen i.p.v. de waarde die ingevuld was) variabele de ID aanpassen. Als je het ID 1 maakt en dit op Arduino 1 upload kan je op Arduino 2 exact dezelfde code uploaden, alleen dan met ID 2. Dit werkt dus gewoon en de rest wordt automatisch gedaan.

h2. Testplan

h3. Algemene pre-condities

Alles is aangesloten volgens het hierboven genoemde aansluitschema (met dus de juiste code van de links op de juiste ESP en Arduino), tenzij er van afgeweken moet worden om het scenario te laten voordoen. Indien nodig moet de aansluiting voor de kabels moet gedaan worden a.d.h.v. het KiCad schema die bovenaan bij Scenario staat.
Beide microcontrollers staan aan en kunnen de code runnen (dus bijvoorbeeld genoeg RAM en flash), tenzij hiervan afgeweken moet worden om het scenario te laten voordoen. Dit kan je doen door beide microcontrollers met USB-kabels aan de laptop te verbinden.
De debug variabele in debug.cpp moet op true staan.

h4. _Scenario 1: Wat gebeurt er als er 0 ingevuld wordt bij aantal config rows?_

h5. Pre conditie(s)

h5. Stappenplan

h5. Verwacht resultaat

h4. _Scenario 2:_

h5. Pre conditie(s)

h5. Stappenplan

h5. Verwacht resultaat

h4. _Scenario 3:_

h5. Pre conditie(s)

h5. Stappenplan

h5. Verwacht resultaat

h4. _Scenario 4:_

h5. Pre conditie(s)

h5. Stappenplan

h5. Verwacht resultaat

h4. _Scenario 5:_

h5. Pre conditie(s)

h5. Stappenplan

h5. Verwacht resultaat

h2. Testrapport