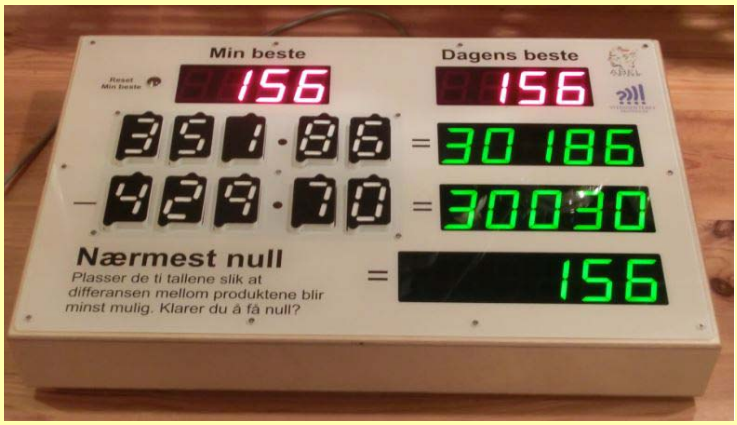
Vitensenteret

ABEL PRISEN

Nils Kr. Rossing

Servicehefte – "Nærmest null"



April 2015

Denne siden er blank

Servicehefte – “Nærmest null”

Nils Kr. Rossing

Servicehefte – "Nærmest null"

Trondheim 2015

ISBN 978-82-92088-54-8

Bidragsytere:
Nils Kr. Rossing, *(nkr@vitensenteret.com)* Vitensenteret i Trondheim

Layout og redigering: Nils Kr. Rossing, Vitensenteret i Trondheim

Tekst og bilder: Nils Kr. Rossing, Vitensenteret i Trondheim

Faglige spørsmål rettes til:
Vitensenteret i Trondheim
v/Nils Kr. Rossing, 73 59 77 23
nils.rossing@vitensenteret.com

Kongensgate 1
7013 Trondheim

Postboks 117
7400 Trondheim

Vitensenteret i Trondheim
Telefon: 73 59 61 23
Telefaks: 73 59 61 20
http://www.vitensenteret.com/

Rev 1.3 – 24.04.15

Prosjektet er gjennomført i med støtte fra Niels Henrik Abels Minnefond.

Forord

Heftet gir en kortfattet orientering om bruk og vedlikehold av installasjonen "Nærmest null" som er utviklet som et delprosjekt under Abel-fond-prosjektet som startet i 2011 og avsluttes ved årsslutt 2015. Utvikling av rike interaktive installasjoner for bruk ved Vitensenter, har vært et samarbeid mellom Vitensenteret i Trondheim og VilVite i Bergen. I løpet av prosjektet er det utviklet to interaktive modeller ved VilVite og to ved Vitensenteret i Trondheim (ViT). Det har vært et krav at utstillingene skal handle om matematikk og utviklet i tråd med intensjonen om "Active prolonged engagement", et prinsipp utviklet ved Exploratorium i San Fransisco.

ViT har utviklet modellene "Nærmest null", hvorav fire leveres i februar og de siste fire i midten av mars, og fire "Ut og fly – størst verdi" som vil bli levert i løpet av juni 2015. Prototypene er utviklet ved ViT av **Nils Kr. Rossing**, **Steinar Løken-Olsen**, **Henning Lundh** og **Frode Willmann**. Science Project Ltd. London ved **Guy Griffin** har utviklet og produsert de endelige kabinettene som er sendt til Trondheim og montert og testet av **Torgeir Rossing** og Nils Kr. Rossing.

En takk til **Torgeir Ekeland**, prosjektleder ved VilVite, som fant alle 198 løsningene på det matematiske problemet.

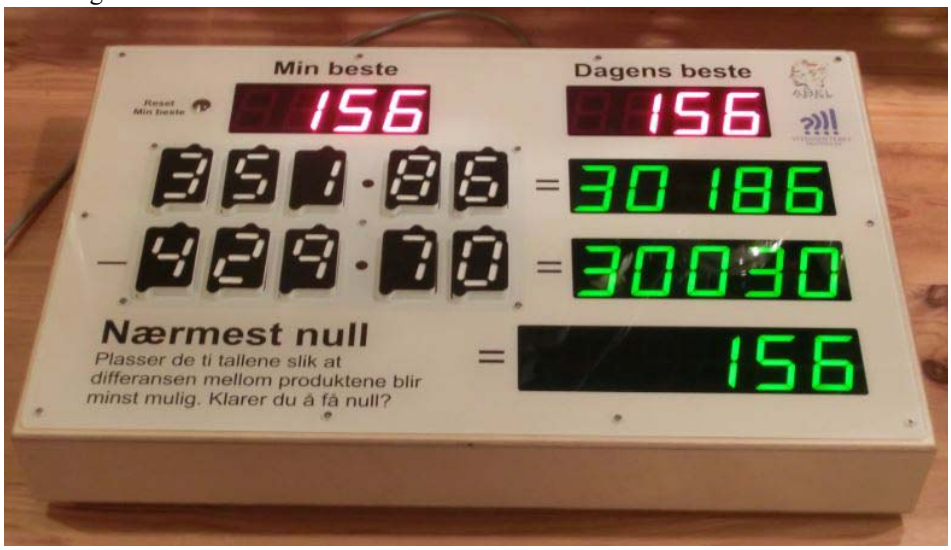
Dette heftet gir en første innføring i bruk og vedlikehold av "Nærmest null".

Vitensenteret
April 2015

Nils Kr. Rossing

1 Innledning

Notatet gir en kort beskrivelse av “Nærmest null”.



Ved hjelp av sifferbrikkene 0 – 9 skal det lages to produkter. Oppgaven går ut på å plassere sifrene slik at differansen mellom produktene blir minst mulig og helst null. Etter som man flytter brikkene rundt, beregnes differansen av produktene automatisk. Dagens beste verdi vises i displayet merket “Dagens beste”, øverst til høyre i rødt når differansen er kommet under 9999.

Dersom man ønsker å konkurrere med seg selv, kan man resette “Min beste” verdi med trykkbryteren til venstre for displayet som viser “Min beste”. Bryteren holdes nede et par sekunder og verdien resettes til 9999 eller den aktuelle øyeblikksverdien av differansen om denne er mindre enn 9999.

Også “Dagens beste” kan resettes. Dette gjøres ved å trykke på en knapp som sitter under kabinettet bak til venstre, og vil normalt betjenes av ansatte. Hele installasjonen kan også resettes ved å slå av hovedstrømmen med bryteren som sitter bak til venstre på kabinettet ved strøminntaket.

Når en eller flere av tall-brikkene ikke er på plass, settes produktene og differansen til ---------.

Når man oppnår differansen 0, vil “beste”-verdiene begynne å blinke. Disse vil blinke til brikkene flyttes ut av posisjon, da vil begge “beste”-verdiene settes tilbake til 9999 eller den aktuelle differansen om denne er mindre enn 9999. Dermed kan man begynne på nytt igjen. Det anbefales at løsningen som gir null noteres før brikkene rokeres.

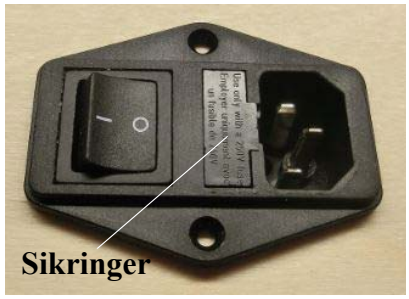
Er det mulig å oppnå en differanse på null?

Ja det finnes mange løsninger som gir en differanse like null. Det er funnet nærmere 50 løsninger og det finnes sikker flere. Et utvalg løsninger finnes i vedlegg A.

Det finnes også mange ulike strategier for å finne løsninger. Noen av disse er omtalt i vedlegg B.

2 Oppstart og service

2.1 Oppstart og reset



Modellen startes ved å sette stikkontakten i veggen og slå på hovedbryteren som er bak til venstre. Mellom bryteren og nett-tilkoblingen sitter sikringsholderen. Ved skifte av sikring vippes holderen ut med et skrujern og sikringene erstattes med *trege* 1 A sikringer (5 x 20 mm). Det er viktig at de er trege, da raske sikringer ikke tåler startstrømmene til power supplyet.

Det finnes to reset-knapper på istallasjonen:

1. ***Reset "Min beste":*** Denne resetter displayet som viser min beste verdi, og settes til 9999 eller, om differansen for øyeblikket er lavere, til denne verdien. Når en "Min beste" passerer "Dagens beste" vil de følge hverandre.
2. ***Reset "Dagens beste":*** Denne sitter under kabinettet til venstre bak, og resetter displayet som viser dagens beste verdi. Denne settes til 9999 eller, om differansen for øyeblikket er lavere, til denne verdien. Det samme vil skje dersom man slår av og på hovedbryteren bak.

2.2 Tall-brikkene

Det følger med 10 tall-brikker med sifrene 0 – 9. Disse er merket med magneter på baksiden. Magnetene er presset inn i hullene og er ikke limt. Om noen av magnetene skulle falle ut, kan de erstattes med sylindriske magneter Ø10 mm x 2 mm (uten hull) som kan kjøpes hos Clas Ohlson. Magnetene slås inn i hullet med en hammer og en plaksiglassstav som brukes som dor.

Dersom noen av brikkene forsvinner ta kontakt med Vitensenteret i Trondheim.

2.3 Oppdatering av programvare

Siden dette er første gang utstyret er i salg, vil sannsynligvis programvaren ikke være optimal til tross for at installasjonen er langtidstestet i utstilingen ved Vitensenteret i Trondheim. Ta kontakt med Vitensenteret i Trondheim og beskriv ev. problemer som er oppstått. Oppgradert programvare kan da oversendes og installers av lokale teknikere.

Nedlasting og installasjon av programvare

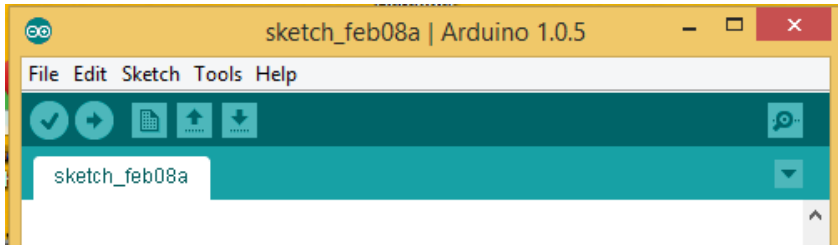
Det er programeditoren til Arduino som benyttes. Denne hentes ned fra

http://arduino.cc/en/pmwiki.php?n=main/software

pakkes ut og installers på en PC eller MAC. Aktuell versjon er i skrivende stund Arduino 1.0.6.

Slå på PC og plugg USB-kablene fra modellen inn i en USB-port. Kabelen for installasjonen skal være fastet under kabinettet, bak til venstre. Dra kabelen ut av hullet og koble den til PC'en/ MAC'en. Om du er heldig vil den autmatisk be deg installere drivere. Følg da anvisningen.

For å sjekke om driverne er installert, kan du starte Arduino-editoren ved å velge ikonet vist til høyre. Man kommer da inn i editoren og får opp følgende vindu (bare øverste del er vist i figuren under).

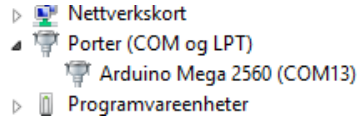


Ved å velge *Tools* fra mentlinjen vil man få flere valg, deriblant *Board* og *Serial Port*.

Board – velg: *Arduino Mega 2560*

Serial Port – velg: *COM* <høyste nr.>

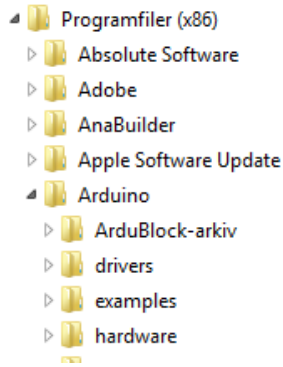
Dersom driverne ikke er installert riktig vil man hos PC'er bare finne *COM 1* eller ingen (Serial Port er "grått"). I så tilfelle kan man gå til Kontrollpanel/System/Enhetsbehandling her skal det komme opp et alternativ i listen som heter *Porter*. Velger man denne vil man se noe i likhet med det som er vist på figuren under:



Hvor Arduino Mega 2560 er tildelt port COM13, eller man skal få varsel om at det er oppdaget en ukjent port. I så fall høyre-klikker man på den ukjente porten og velger:

Oppdater driver programvare

Og man kommer inn i programvaren for installasjon av drivere. Her kan det være lurt å *peke på driverprogrammvaren* i stedet for å velge *Automatisk*. Driverne ligger under Programfiler (x86)/ Arduino/Drivers

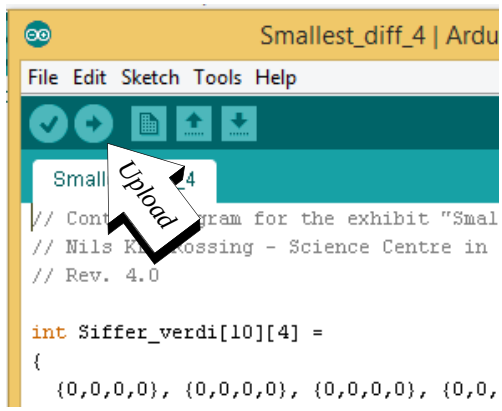


Installer ny programvare i "Nærmest null"

Programfilen for innlasting vil ha navnekonvensjonen:

Smallest_diff_<Rev nr.>.ino den medfølgende er Smallest_diff_4.ino

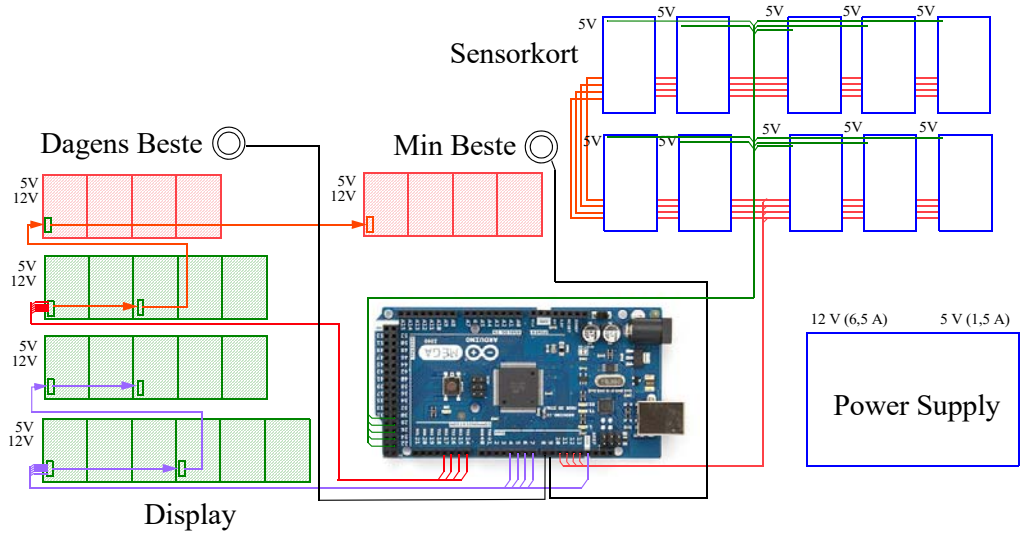
Programfilen hentes inn i Arduino editoren (IDE) ved å velge *File/Open*. Deretter kompileres og lastes filen over til den interne Arduino MEGA ved å velge *File/Upload* (pil mot høyre).



Programvaren kan lastes opp mens modellen er avslått siden Arduino Mega kortet får spenning via USB-kabelen. Den kan selvfølgelig også lastes opp mens modellen er påslått.

3 Kort beskrivelse av installasjonens oppbygning

Blokkdiagrammet under viser en overordnet skisse av hvordan maskinvaren er bygget opp.



Kontrollerkortet er en Arduino Mega 2650 med shieldkort for at det skal være lett å lodde til ledningene.

Sensorkortene

Sensorene er hall switcher som slutter når en magnet kommer i nærheten. Nordpolen på magnetene skal peke ned mot sensorene. Hvert kort består av fire sensorer slik at en i prinsippet kan skille mellom 16 ulike magnetkonstellasjoner. Null-tilstanden (ingen magneter) brukes ikke for å detektere ett av sifrene, men for å detektere manglende brikker. En tri-state bussdriver gjør at sensorene kan leses av kort for kort, siffer for siffer ved hjelp av enable linjer til hvert av kortene.

Displaykortene

Displaykortene er utviklet sv Science Project. Disse leveres med fire siffer, men kan deles opp etter behov. Kortene kobles i kjede slik at dataene sendes inn på seriell form i kortet lengst mot høyre for så å riple gjennom kortene. På grunn av driveregenskapene til kontrollerkortet benyttes to separate serielinjer. Den ene linjen forskyner Differansen og Produkt 2, mens den andre forskyner Produkt 1 og displayene til de to “beste”-resultatene.

Reset knappene

De to resetknappene kobles til to dgitale innganger (d8 og d9) på Arduino Mega. Ved tilkobling benyttes en 22 nF kondensator og 10 kOhm for å dempe uønsket prell.

Strømforskyningen (Power supply)

Strømforskyningen består av en 5 V forskyning som kan levere inntil 1,5 A og en 12 V forskyning som kan levere inntil 6,5 A. Arduino Mega forskynes med 12 V som reguleres ned til 5 V på kortet. Sensorkortene forskynes med 5 V og vil dermed ikke belaste regulatoren på kontroller-kortet.

4 Mulige feil

4.1 Feilregistrering av tall

Det er oppdaget at det er mulig å registrere feil tall akkurat ide det siste tallet legges ned på sin plass. Dette kan skje dersom tallet legges langsomt ned, og det har to mint to magneter som sitter langt fra hverandre. Da kan det skje at bare den ene magneten registreres på vei ned i hullet, slik at produktet et kort øyeblikk blir feil. Dette er imidlertid uten betydning da det riktige tallet kommer på plass omtrent umiddelbart etter.

Imidlertid kan det skje at det "gale" tallet gir et bedre "beste resultatet". Dermed blir dette "gale" besteresultatet stående uten at det egentlig er oppnådd.

Feilen er så skjelden at vi har valgt å ikke gjøre noe med den inntl videre.

4.2 Ustabil registrering av tall

Dersom man oppdager at en tallbrikke gir forskjellig tall avhengig av om den er plassert til høyre eller til venstre i sin nedfelte posisjon, så er det stor sannsynlighet for at dette skyldes at man har fått en skjevstilling av en av sensorene bak denne brikken. Da gjør man som følger:

- Finn hvilken posisjon som gir feil.
- Finn hvilken av de fire mulige magnetposisjonen som gir feil.
  Dette kan man gjøre ved å bruke brikkene 1, 2, 4 og 8. Disse benytter hver sin av de fire sensorene. Legg hver av de fire brikkene i den aktuelle posisjonen og se hvilke av de fire som gir den nevnte feilen. Når den aktuelle brikken er funnet, snur man den og sjekker for hvilken magnet feilen oppstår.
- Skru av bakdekselet og finn de ti sensorkortene (se oversiktsbilde på side 14)
- Skru løs sensorkortene, så mange som nødvendig for å komme til.
- Løft opp kortetog juster sensoren slik at den nærmer seg den kanten hvor feilen oppsto.
- Sett kortene tilbake og lukk boksen.

4.3 Umulige tall på displayet

Vi har i noen svært få tilfeller registreret at displayet har vist "tall" som er feil eller ikke er noe tall. I slike tilfeller har vi byttet displaymodelen og antatt at det er en svakhet i den aktuelle modu-

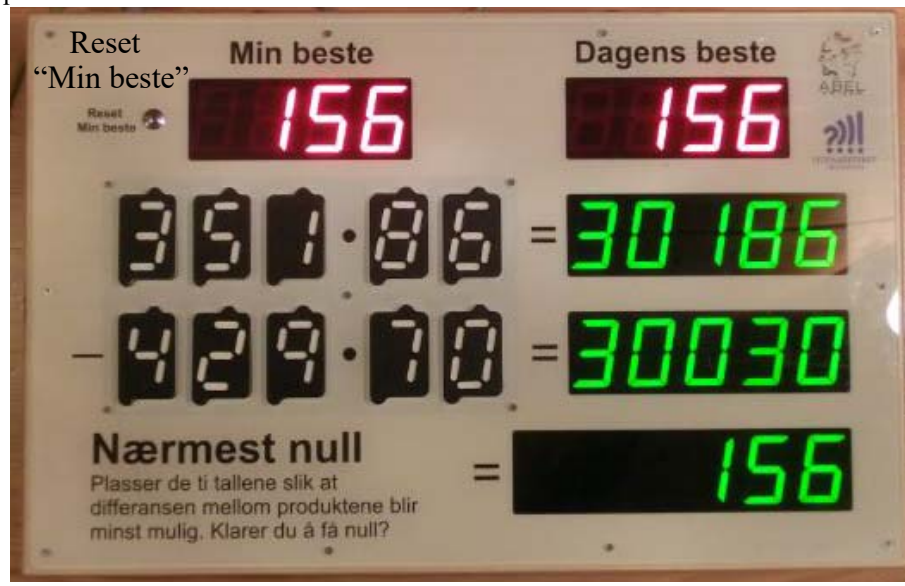
len. Selv om vi har testet modellene i mange dager så vet vi lite om ev. feil som kan oppstå på sikt. Vi vil derfor gjerne ha melding derom slik feil skulle dukke opp.

4.4 Tallbrikker som forsvinner

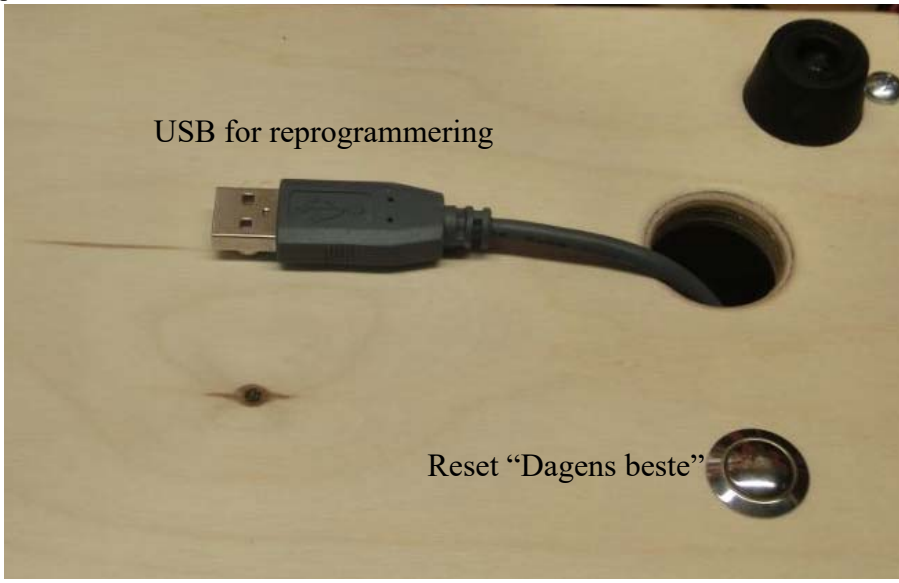
Det er bestilt 4 komplette sett med tallbrikker. Dersom en eller flere brikker skulle forsvinne så ta kontakt med Vitensenteret i Trondheim, så vil vi ettersende nye brikker. Det må påregnes en kostnad på ca. 150,- kr pr. ny brikke.

5 Bilder av interiør

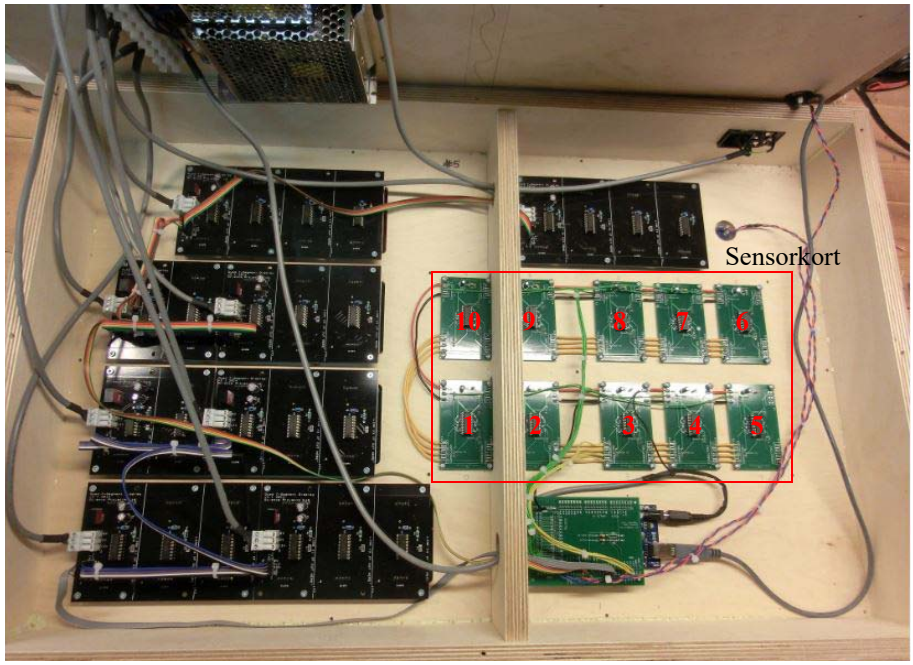
Toppplate med tall-brikker



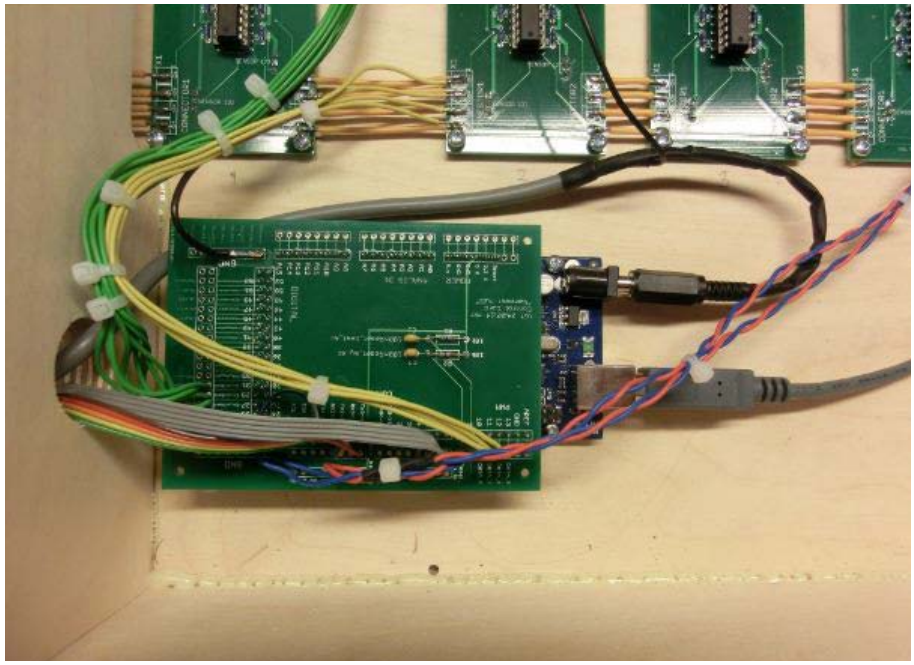
Bunnplate



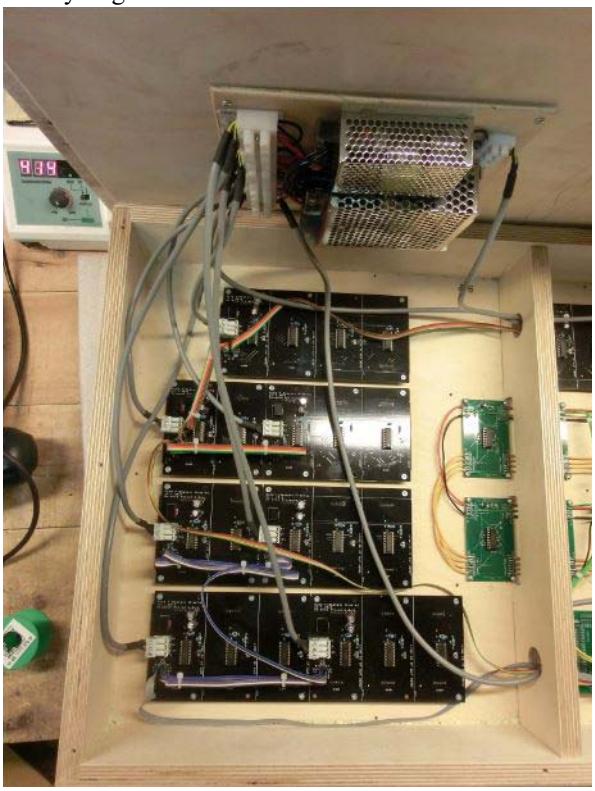
Oversiktsbilde av elektronikk



Kontrollerkort med shield



Displaykort og strømforskyning



Nettilkobling og sikringer.

Vedlegg A Løsninger som gir null

Torgeir Ekeland prosjektleder ved Vitensenteret i Bergen, har ved hjelp av en programalgoritme funnet samtlige løsninger 198 løsninger.

(046x79)-(158x23)
(054x69)-(138x27)
(054x93)-(186x27)
(058x67)-(134x29)
(058x69)-(174x23)
(058x73)-(146x29)
(058x96)-(174x32)
(063x74)-(259x18)
(064x79)-(158x32)
(067x58)-(134x29)
(069x54)-(138x27)
(069x58)-(174x23)
(073x58)-(146x29)
(073x96)-(584x12)
(074x63)-(259x18)
(076x98)-(532x14)
(079x46)-(158x23)
(079x64)-(158x32)
(093x54)-(186x27)
(096x58)-(174x32)
(096x73)-(584x12)
(098x76)-(532x14)
(123x56)-(984x07)
(130x97)-(485x26)
(132x46)-(759x08)
(134x29)-(058x67)
(134x29)-(067x58)
(135x96)-(270x48)
(135x96)-(480x27)
(136x27)-(459x08)
(138x27)-(054x69)
(138x27)-(069x54)
(138x90)-(276x45)
(145x76)-(290x38)
(145x76)-(380x29)
(146x29)-(058x73)
(146x29)-(073x58)
(148x70)-(296x35)
(153x28)-(476x09)
(153x46)-(782x09)
(154x29)-(638x07)
(156x23)-(897x04)
(158x23)-(046x79)
(158x23)-(079x46)
(158x32)-(064x79)
(158x32)-(079x64)
(160x97)-(485x32)
(165x78)-(429x30)
(170x94)-(235x68)
(174x23)-(058x69)
(174x23)-(069x58)
(174x32)-(058x96)
(174x32)-(096x58)
(184x95)-(230x76)
(184x95)-(760x23)
(185x92)-(370x46)
(185x92)-(460x37)
(186x27)-(054x93)
(186x27)-(093x54)
(186x45)-(279x30)
(186x90)-(372x45)
(187x92)-(506x34)
(195x72)-(468x30)
(195x84)-(273x60)
(217x80)-(496x35)
(230x76)-(184x95)
(235x68)-(170x94)
(235x68)-(940x17)
(237x60)-(948x15)
(248x95)-(310x76)
(248x95)-(760x31)
(259x18)-(063x74)
(259x18)-(074x63)
(259x86)-(301x74)
(267x18)-(534x09)
(269x14)-(538x07)
(270x48)-(135x96)
(270x63)-(945x18)
(273x18)-(546x09)
(273x60)-(195x84)
(276x45)-(138x90)
(279x30)-(186x45)
(279x30)-(465x18)
(290x38)-(145x76)
(293x14)-(586x07)
(296x35)-(148x70)
(296x51)-(408x37)
(301x74)-(259x86)
(306x27)-(459x18)
(307x58)-(614x29)
(309x54)-(618x27)
(310x76)-(248x95)
(310x97)-(485x62)
(318x60)-(795x24)
(327x18)-(654x09)
(328x95)-(410x76)
(328x95)-(760x41)
(329x14)-(658x07)
(345x26)-(897x10)
(351x96)-(702x48)
(360x89)-(712x45)
(360x91)-(728x45)
(370x46)-(185x92)
(372x45)-(186x90)
(380x29)-(145x76)
(381x90)-(762x45)
(406x37)-(518x29)
(408x37)-(296x51)
(408x79)-(632x51)
(408x92)-(736x51)
(410x76)-(328x95)
(429x30)-(165x78)
(451x76)-(902x38)
(459x08)-(136x27)
(459x18)-(306x27)
(460x37)-(185x92)
(465x18)-(279x30)
(468x30)-(195x72)
(476x09)-(153x28)
(476x53)-(901x28)
(480x27)-(135x96)
(481x70)-(962x35)
(485x26)-(130x97)
(485x26)-(970x13)
(485x32)-(160x97)
(485x32)-(970x16)
(485x62)-(310x97)
(485x62)-(970x31)
(486x30)-(972x15)
(496x35)-(217x80)
(504x89)-(712x63)
(504x91)-(728x63)
(506x34)-(187x92)
(518x29)-(406x37)
(532x14)-(076x98)
(532x14)-(098x76)
(534x09)-(267x18)
(538x07)-(269x14)
(546x09)-(273x18)
(584x12)-(073x96)
(584x12)-(096x73)
(586x07)-(293x14)
(614x29)-(307x58)
(618x27)-(309x54)
(618x45)-(927x30)
(630x27)-(945x18)
(632x51)-(408x79)
(638x07)-(154x29)
(654x09)-(327x18)
(658x07)-(329x14)
(702x48)-(351x96)
(702x63)-(819x54)
(712x45)-(360x89)
(712x45)-(890x36)
(712x63)-(504x89)
(716x30)-(895x24)
(716x40)-(895x32)
(728x45)-(360x91)
(728x45)-(910x36)
(728x63)-(504x91)
(732x60)-(915x48)
(732x80)-(915x64)
(736x51)-(408x92)
(748x20)-(935x16)
(754x31)-(806x29)
(759x08)-(132x46)
(760x23)-(184x95)
(760x31)-(248x95)
(760x41)-(328x95)
(762x45)-(381x90)
(782x09)-(153x46)
(782x53)-(901x46)
(795x24)-(318x60)
(806x29)-(754x31)
(819x54)-(702x63)
(890x36)-(712x45)
(895x24)-(716x30)
(895x32)-(716x40)
(897x04)-(156x23)
(897x10)-(345x26)
(901x28)-(476x53)
(901x46)-(782x53)
(902x38)-(451x76)
(910x36)-(728x45)
(915x48)-(732x60)
(915x64)-(732x80)
(927x30)-(618x45)
(935x16)-(748x20)
(940x17)-(235x68)
(945x18)-(270x63)
(945x18)-(630x27)
(948x15)-(237x60)
(962x35)-(481x70)
(970x13)-(485x26)
(970x16)-(485x32)
(970x31)-(485x62)
(972x15)-(486x30)
(984x07)-(123x56)

Forslag til oppgaver for de av publikum som vil noe mer:

- Klarer du å finne en løsning som ingen i publikum ennå har funnet?
- Finn de løsningene hvor produktene har minst eller størst verdi
- Be publikum om å beskrive den strategien de brukte for å komme fram til en løsning.
- Finn ut hvilke av de nevnte løsninger som er vanligst blant publikum, eller fordeler det seg jent over.
-

Vedlegg B Noen løsningsmetoder

Som omtalt blir publikum utfordret til å lage to produkter med sifrene 0 – 9 som vist i eksempelet under:

314 x *79* = 24806
265 x *80* = 21200
3606

Sifrene i kursiv er "løse" og kan fritt plasseres i de to produktene. Oppgaven går ut på å plassere dem slik at differansen mellom produktene kommer så nær null som mulig, og helst helt ned til null. Det oppgis at det er 3.628.800 (10!) mulige permutasjoner av de 10 sifrene, og at det så langt er funnet ca. 50 løsninger som gir null.

Vi skal se på noen strategier for å løse oppgaven og i hvilken grad disse oppfyller våre kriterier for en rik modell:

Enkel ombytting

Denne strategien forsøker ved enkel ombytting av to siffer å redusere produktet som er størst og/eller øke produktet som er minst. Denne strategien utfordrer vår evne til å vurdere konsekvensen av å øke et av sifrene i et tall samtidig som et annet siffer reduseres i et annet (ev. samme) tall. Eksempelet under viser dette:

341 x *79* = 26939
256 x *80* = 20480
6459

Her ser vi at det øverste produktet er for stort og det nederste er for lite. Vi ser umiddelbart at vi kan redusere det øverste ved å bytte om på sifrene 4 og 1. Dette skulle redusere det øverste tallet med 27 x 79, eller ca. 2000. Likeså kan vi øke det nederste produktet ved å bytte om på sifrene 5 og 6, som vil øke produktet med ca. 9 x 80 som er drøyt 700. Vi ser at dette gjør at de to produktene nærmer seg hverandre med knapt 3000. Det er ikke nok, men et stykke på vei. Slik kan vi fortsette å gjøre enkle ombyttinger så lenge det er mulig. Senere blir det vanskeligere, og en må ev. foreta flere påfølgende ombyttinger for å oppnå redusert differanse, hvilket er en krevende øvelse.

Nabotall

Siden differansen skal bli null, er det nærliggende å tenke at de to tallene til høyre i regnestykket må være så nær hverandre som mulig. Tilsvarende må tallene til venstre også være så nær hverandre som mulig, men i motsatt rekkefølge (se eksempelet over).

I eksempelet over har vi valgt nabotallene 79 og 80 til høyre. Vi legger fort merke til at vi ikke kan velge nabotall av typen 12 og 13, 25 og 26 eller 56 og 57 og lignende. Siden alle disse tallene gjenbruker ett av sifrene. Dermed ender vi raskt opp med tallkombinasjonene vist i rammen til høyre:. Flere enn disse finnes ikke (09 og 10, og 89 og 90 inneholder begge to like siffer).

19 og 20	59 og 60
29 og 30	69 og 70
39 og 40	79 og 80
49 og 50	

Det neste en kan tenke seg å gjøre, er å velge nabosiffer som første siffer i de to tallene til venstre i produktene. I eksempelet over er valgt tallkombinasjonen 3 og 2. Med det største tallet øverst (3) og det minste tallet nederst (2), etter som tallene til høyre har sitt minste tall øverst og sitt største tall nederst.

Vi kan så sette opp en tabell med alle mulige kombinasjoner avhengig av hvilke tall vi velger til høyre som vist i tabellen under.

Mest signifikante siffer hos tallene til venstre							X	tall til høyre	Beste resultat
8 7	**7** **6**	6 5	5 4	4 3	| |	| |	X	**19** **20**	246
8 7	**7** **6**	6 5	5 4	| |	| |	| 1	X	**29** **30**	939
8 7	**7** **6**	6 5	| |	| |	| |	2 1	X	**39** **40**	719
8 7	7 6	| |	| |	| |	**3** **2**	2 1	X	**49** **50**	1134
8 7	| |	| |	| |	**4** **3**	3 2	2 1	X	**59** **60**	1088
8 |	| |	| |	5 4	**4** **3**	3 2	2 1	X	**69** **70**	1478
| |		6 5	5 4	**4** **3**	3 2	2 1	X	**79** **80**	3348

Det tallparet i mest signifikante siffer i tallene til venstre, som gir lavest verdi for differansen, er gjort fete og skrevet i rødt. De gjenværende fire siffer plasseres slik at tallverdien til tallene til venstre kommer nærmest mulig hverandre (i eksempelet 314 og 265). Det vil si at det øverste tallet, som skal være størst av de to, gjøres minst mulig (314), og det nederste tallet til venstre, som skal være minst, gjøres så stort som mulig (265). Dermed vil tallverdien til tallene bli så like som mulig under de nevnte forutsetningene. Dette vises i eksempelet omtalt innledningsvis.

246 er derfor det minste differansen som kan oppnås med denne strategien:

734 x *19* = 13946
685 x *20* = 13700
246

For å komme videre må man velge tall til høyre som ikke ligger nærmest hverandre. La oss før vi går videre se hvordan dette harmonerer med våre kriterier:

Alle kriterier unntatt nr. 3 synes å være oppfylt. Modellen gir umiddelbar respons, stimulerer til konstruktive tankeprosesser og krever at vi bruker matematiske kunnskaper, som f.eks. overslagsregning, forholdstall og posisjonssystemet. Man er heller ikke særlig i tvil om at det er matematikk man bedriver.

Så langt kan vi si at vi oppfyller alle kriteriene unntatt den tredje om at vi skal kunne klare å komme helt i mål ved å bruke ***en eller flere suksessive strategier.*** *Spørsmålet er derfor hvordan vi oppnår lavere verdier for differansen.*

En måte å gjøre dette på er å unngå lokale minimumspunkter.

Lokale minima

Som vi erfarte klarte vi ikke å komme helt i mål med metoden foran. Dvs. at vi ikke er i stand til å komme lavere enn 246 ved hjelp av enkle ombyttinger. Vi sier at vi har nådd et ***lokalt minimumspunkt***. For å komme videre må vi flytte om på flere brikker og forlate tanken på at tallene til høyre skal komme etter hverandre.

Det er ikke vanskelig å tenke seg at det er mulig å oppnå null selv om avstanden mellom tallene til høyre øker. Det er bare å øke avstanden mellom tallene til venstre tilsvarende mye.

Det er imidlertid en langt mer krevende og tilfeldig øvelse å finne nullpunkter på denne måten. Dersom vi studerer listen over oppstillinger som gir null (se vedlegg A), vil vi oppdage at forskjellen mellom tallene til høyre kan være fra 4 til 72 eller kanskje mer. Med andre ord spredningen er stor. Umiddelbart synes det heller ikke som om det er noe tydelig system i forskjellen mellom disse tallene. Det kan bety at den beste måten å komme videre på er å velge tallene relativt tilfeldig for så å forsøke å minimalisere differansen derfra.

En fornuftig framgangsmåte kan derfor være:

- *Ta ut alle tallene og legg dem inn på en tilfeldig måte*
- *Bytt om to og to tall på venstre side av multiplikasjonstegnet slik at differansen nærmer seg null*
- *Dersom du oppnår et godt resultat uten å komme til null, noterer du oppstillingen og resultatet på en lapp og gjentar prosedyren foran*

Umiddelbart synes dette å være en lite tilfredsstillende måte å gjøre det på, da den fortsetter en *ikke* suksessiv prosess fra start til mål (null i differanse). Observasjoner av publikum viser da også at de fleste nekter å bryte opp en "god" løsning for å oppnå noe enda bedre. Hver gang de har gjort en endring og ser at den ikke gir den ønskede reduksjon i differansen, så går de tilbake til sin bestenotering. Etter hvert skaper den fastlåste situasjonen frustrasjon. I verste fall forlater de modellen.

For å hjelpe dem bort fra en slik situasjon, foreslår vi at de skal notere sitt beste resultat og blande brikkene og begynne på nytt. Siden deres beste oppstilling er notert, er det lav "risiko" ved å begynne på nytt. De kan alltid komme tilbake til sin tidligere bestenotering.

Ut fra dette resonnementet må vi konkludere med at modellen så langt ikke synes å ha oppfylt følgende krav:

- *Oppdraget skal kunne løses ved hjelp av en eller flere **suksessive** strategier*

Når *det* er sagt er den nevnte metoden med å bryte opp en god løsning for å begynne helt på nytt, en anerkjent matematisk metode i matematikken for å komme seg bort fra lokale minima. Den oppleves imidlertid ikke "riktig" i denne sammenhengen. Da tilfeldig prøving og feiling ikke faller inn under konstruktive tankeprosesser (eller gjør de det?).

Siste siffer likt

En annen ting vi kan legge merke til er at i metoden omtalt foran så er siste siffer i tallene til høyre i produktet 9 og 0. Dvs. at det nederste produktet (685 x 20) vil alltid ende på '0', mens det øverste produktet (734 x 19) vil aldri ende på '0'. Om det skulle kunne skje så må tallet til venstre ende på null hvilket det ikke kan fordi '0'en' er brukt i tallet 20 (se eksempelet under). Vi vet også at skal vi kunne ende opp med en differanse lik null så må også siste siffer være likt.

734 x *19* = 13946
685 x *20* = 13700
246

Så hvilke siffer kan det være lurt å bruke i siste posisjon hos de to tallene? Her er ett eksempel:

158 x *23* = 3634
046 x *79* = 3634
0

Vi ser at 8 x 3 = 24 og 6 x 9 = 64, som begge har 4 som siste siffer. Vi vet altså at løsninger som gir en differanse på null har siste siffer i de fire tallene som multiplisert med hverandre to og to gir det samme siste siffer.

La oss sette opp den lille multiplikasjonstabellen og undersøke hvilke to siffer som multiplisert med hverandre gir produkter som henholdsvis ender på 0, 1, 2 og 3:

Produkter som ender på 0

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	26	45	54	63	72	81

Produkter som ender på 1

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	26	45	54	63	72	81

Produkter som ender på 2

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	26	45	54	63	72	81

Produkter som ender på 3

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	42	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	26	45	54	63	72	81

De resterende 6 tabellene er ikke tatt med. Under finnes alle kombinasjoner av tall som gir samme siffer på begge produktene. I praksis må vi minst ha to tallkombinasjoner som gir samme endesiffer. Dermed ser vi at ingen løsninger har produkter som ender på 1 og 9, mens sannsynligheten for å finne løsninger som ender på 0 er relativt stor. Vi må også sørge for å velge to tallkombinasjoner hvor alle siffer er ulike. Dermed vil ikke | 5·2 | og | 5·4 | være mulig siden 5-tallet er med i begge kombinasjonene.

Ender på 0: | 1·0 | 2·0 | 3·0 | 4·0 | 5·0 | 6·0 | 7·0 | 8·0 | 9·0 | 5·2 | 5·4 | 6·5 | 8·5 |

Ender på 1: | 7·3 | (i praksis ingen)

Ender på 2: | 2·1 | 4·3 | 6·2 | 8·4 | 6·7 | 9·8 |

Ender på 3: | 3·1 | 9·7 |

Ender på 4: | 4·1 | 6·4 | 7·2 | 8·3 | 9·6 |

Ender på 5: | 5·1 | 5·3 | 7·5 | 9·5 |

Ender på 6: | 6·1 | 3·2 | 8·2 | 9·4 |

Ender på 7: | 7·1 | 9·3 |

Ender på 8: | 8·1 | 4·2 | 6·3 | 7·4 | 8·6 | 9·2 |

Ender på 9: | 9·1 | (i praksis ingen)

Når plasseringen av fire siffer er bestemt burde være enklere å plassere de seks siste slik at differansen blir null. Vi kan uttrykke prosedyren slik:

1. Velg to tallkombinasjoner som gir samme siste siffer i begge produktene
2. Dersom du velger 0 som siste siffer, så vil det være flere kombinasjoner som kan gi en differanse på 0 enn noen andre. Dessuten er bare tre siffer låst, siden det fjerne skal multipliseres med null og dermed vil gi null uansett hva det multipliseres med.
3. Minimaliser differansen ved å bytte om to og to siffer.
4. Om du ikke kommer helt til null, prøv en annen kombinasjon av tall som gir likt siste siffer i de to produktene.

Dersom vi studerer listen over de løsningene vi har funnet, så vil vi se at det kan være lurt å velge siffer som gir 0 i siste siffer i begge produkter. Hele 24 av det 47 kjente løsningene har produkter som ender på 0.

Forholdstall

Forholdet mellom tallene til venstre i produktene er det samme som forholdet mellom tallene til høyre i produktene. En strategi kan derfor være å forsøke å plassere tallene slik at de nevnte forholdstallene blir så like som mulig.

Dette er imidlertid en vanskelig strategi som krever svært god evne til å gjøre overslag over forholdstallene.

Vedlegg C Styreprogram

Under er vist styreprogrammet som ligger i kontrolleren.

```
// Control program for the exhibit "Smallest differance"
// Nils Kr. Rossing - Science Centre in Trondheim 31. Jan. 2015
// Rev. 4.0

int Siffer_verdi[10][4] =
{
  {0,0,0,0}, {0,0,0,0}, {0,0,0,0}, {0,0,0,0},
  {0,0,0,0}, {0,0,0,0}, {0,0,0,0}, {0,0,0,0},
  {0,0,0,0}, {0,0,0,0}
};

int pinData[4] =
  {13, 12, 11, 10};

int pinEn_siffer[10] =
  {22, 23, 24, 25, 26, 27, 28, 29, 30, 31};

int En_siffer[10] =
  {1, 1, 1, 1, 1, 1, 1, 1, 1, 1};

int decSiffer[10] = {5, 6, 2, 101, 1, 3, 0, 9, 8, 101};

int conTabell[16][5] =
{
  {1,1,1,1,100}, {1,1,1,0,1},   {1,1,0,1,2},   {1,1,0,0,3},
  {1,0,1,1,4},   {1,0,1,0,5},   {1,0,0,1,6},   {1,0,0,0,7},
  {0,1,1,1,8},   {0,1,1,0,9},   {0,1,0,1,0},   {0,1,0,0,101},
  {0,0,1,1,102}, {0,0,1,0,100}, {0,0,0,1,100}, {1,1,1,1,100}
};
```

```
long multiplikator_1, multiplikator_2;
long multiplikand_1, multiplikand_2;

long product_1, product_2;
long differance = 10000;
long high_score = 9999;
long my_high_score = 9999;
int reset_my_high_score = 1;
int reset_high_score = 1;
int sign;

int tidSet_enable = 2;
int tidLes_data = 2;

int i, m, n, p;
int antallSiffer = 10;
int antallPosisjoner = 16; //Antall posisjoner i konverteringstabell

// Declaration for display

// unsigned char Sif[5];

long Siffer_1 = 0;
long Siffer_10 = 0;
long Siffer_100 = 0;
long Siffer_1000 = 0;
long Siffer_10000 = 0;

int pinCLK_1 = 7;
int pinSDI_1 = 6;
int pinLE_1 = 5;
```

```
int pin_OE_1 = 4;

int pinCLK_2 = 14;
int pinSDI_2 = 15;
int pinLE_2 = 16;
int pin_OE_2 = 17;

int pin_reset_my_high_score = 8;
int pin_reset_high_score = 9;

int Sif[24];
byte SifCon[12];
int leadingZeroFlagg = 0;
int initialFlagg = 1;

void setup()   {

 for (i = 0; i < 4; i++)
 {
  pinMode(pinData[i], INPUT);
  digitalWrite(pinData[i], HIGH); // Set pullup resistor
 }

 for (i = 0; i < 10 ; i++)
 {
  pinMode(pinEn_siffer[i], OUTPUT);
 }

 pinMode(pinCLK_1, OUTPUT);
 pinMode(pinSDI_1, OUTPUT);
 pinMode(pinLE_1, OUTPUT);
 pinMode(pin_OE_1, OUTPUT);
```

```
  pinMode(pinCLK_2, OUTPUT);
  pinMode(pinSDI_2, OUTPUT);
  pinMode(pinLE_2, OUTPUT);
  pinMode(pin_OE_2, OUTPUT);

  pinMode(pin_reset_my_high_score, INPUT);
  pinMode(pin_reset_high_score, INPUT);
  digitalWrite(pin_OE_1, LOW);
  digitalWrite(pinLE_1, LOW);
  digitalWrite(pin_OE_2, LOW);
  digitalWrite(pinLE_2, LOW);

  SifCon[0]  = B11111100; // 0 (ABCDEFG DP)
  SifCon[1]  = B01100000; // 1 (ABCDEFG DP)
  SifCon[2]  = B11011010; // 2 (ABCDEFG DP)
  SifCon[3]  = B11110010; // 3 (ABCDEFG DP)
  SifCon[4]  = B01100110; // 4 (ABCDEFG DP)
  SifCon[5]  = B10110110; // 5 (ABCDEFG DP)
  SifCon[6]  = B10111110; // 6 (ABCDEFG DP)
  SifCon[7]  = B11100000; // 7 (ABCDEFG DP)
  SifCon[8]  = B11111110; // 8 (ABCDEFG DP)
  SifCon[9]  = B11110110; // 9 (ABCDEFG DP)
  SifCon[10] = B00000010; // - (ABCDEFG DP)
  SifCon[11] = B00000000; // ' ' (ABCDEFG DP)

  Serial.begin(9600);

  delay(1000);   // This delay should guarantee that the
                 // display is initialised before the program
                 // start comunicating with the display cards
}
```

```
void loop()
{

 // Les siffer m og digit n

 for (m = 0; m < (antallSiffer); m++)
 {
  digitalWrite(pinEn_siffer[m], LOW);  // Read digit m
  delay(tidSet_enable);
  for (n = 0; n < 4; n++)
  {
   Siffer_verdi[m][n] = digitalRead(pinData[n]); // Read position n
   delay(tidLes_data);
  }
  digitalWrite(pinEn_siffer[m], HIGH); // Finish reading digit m
 }

// Converting from magnetpositions to digits

 for (m = 0; m < antallSiffer; m++)
 {
  for (p = 0; p < antallPosisjoner; p++)
  {
   if ((Siffer_verdi[m][0] == conTabell[p][0]) &&
       (Siffer_verdi[m][1] == conTabell[p][1]) &&
       (Siffer_verdi[m][2] == conTabell[p][2]) &&
       (Siffer_verdi[m][3] == conTabell[p][3]))
   {
    decSiffer[m] = conTabell[p][4];
   }
  }
 }
```

```
if ((decSiffer[0] == 100) ||
    (decSiffer[1] == 100) ||
    (decSiffer[2] == 100) ||
    (decSiffer[3] == 100) ||
    (decSiffer[4] == 100) ||
    (decSiffer[5] == 100) ||
    (decSiffer[6] == 100) ||
    (decSiffer[7] == 100) ||
    (decSiffer[8] == 100) ||
    (decSiffer[9] == 100) ||
    (initialFlagg == 1))
    {

    for (i=0; i<16; i++)
    {
      Sif[i] = 10;
    }
    BigDisplay ();
    }
    else
    {

    // Calculating the products/sums

    multiplikator_1 = 100*decSiffer[5]+10*decSiffer[6]+decSiffer[7];
    multiplikand_1 = 10*decSiffer[8]+decSiffer[9];
    product_1 = multiplikator_1 * multiplikand_1;

    multiplikator_2 = 100*decSiffer[4]+10*decSiffer[3]+decSiffer[2];
    multiplikand_2 = 10*decSiffer[1]+decSiffer[0];
    product_2 = multiplikator_2 * multiplikand_2;
```

```
// Product/sum 1
// Detect the sign of det difference

  if (product_1 > product_2)
  {
    differance = product_1 - product_2;
    sign = 1;
  }
  else if (product_1 < product_2)
  {
    differance = product_2 - product_1;
    sign = 0;
  }
  else
  {
    differance = 0;
    sign = 1;
  }

// Product 1
// Converting the numbers to digits

  Siffer_10000 =  (product_1/10000);
  Siffer_1000  =  (product_1 - 10000*Siffer_10000)/1000;
  Siffer_100   =  (product_1 - 10000*Siffer_10000 - 1000*Siffer_1000)/100;
  Siffer_10    =  (product_1 - 10000*Siffer_10000 - 1000*Siffer_1000 - 100*Siffer_100)/10;
  Siffer_1     =  (product_1 - 10000*Siffer_10000 - 1000*Siffer_1000 - 100*Siffer_100 -
10*Siffer_10);

  Sif[11] = Siffer_1;
  Sif[12] = Siffer_10;
```

```
Sif[13] = Siffer_100;
Sif[14] = Siffer_1000;
Sif[15] = Siffer_10000;

// Removing leading zero's

if (Sif[15] == 0)
{
Sif[15] = 11;
if (Sif[14] == 0)
  {
  Sif[14] = 11;
  if (Sif[13] == 0)
    {
    Sif[13] = 11;
    if (Sif[12] ==0)
      {
      Sif[12] = 11;
      }
    }
  }
}

// Product 2
// Converting the numbers to digits

Siffer_10000 =  (product_2/10000);
Siffer_1000  =  (product_2 - 10000*Siffer_10000)/1000;
Siffer_100   =  (product_2 - 10000*Siffer_10000 - 1000*Siffer_1000)/100;
Siffer_10    =  (product_2 - 10000*Siffer_10000 - 1000*Siffer_1000 - 100*Siffer_100)/10;
Siffer_1     =  (product_2 - 10000*Siffer_10000 - 1000*Siffer_1000 - 100*Siffer_100 -
10*Siffer_10);
```

```
Sif[6] = Siffer_1;
Sif[7] = Siffer_10;
Sif[8] = Siffer_100;
Sif[9] = Siffer_1000;
Sif[10] = Siffer_10000;

// Removing leading zero's

if (Sif[10] == 0)
{
Sif[10] = 11;
if (Sif[9] == 0)
  {
  Sif[9] = 11;
  if (Sif[8] == 0)
    {
    Sif[8] = 11;
    if (Sif[7] ==0)
      {
      Sif[7] = 11;
      }
    }
  }
}

// Differance
// Converting the numbers to digits

Siffer_10000 =  (differance/10000);
Siffer_1000  =  (differance - 10000*Siffer_10000)/1000;
```

```
Siffer_100   =  (differance - 10000*Siffer_10000 - 1000*Siffer_1000)/100;
Siffer_10    =  (differance - 10000*Siffer_10000 - 1000*Siffer_1000 - 100*Siffer_100)/10;
Siffer_1     =  (differance - 10000*Siffer_10000 - 1000*Siffer_1000 - 100*Siffer_100 -
                  10*Siffer_10);


Sif[0] = Siffer_1;
Sif[1] = Siffer_10;
Sif[2] = Siffer_100;
Sif[3] = Siffer_1000;
Sif[4] = Siffer_10000;

// Wright the sign of the difference to the display
// Removing leading zero's
if (sign == 0)
  {
  Sif[05] = 10;                    // If differance is negative use "-" before first digit
  }
else
  {
  Sif[05] = 11;
  }

 if (Sif[4] == 0)
   {
   Sif[4] = 11;
   if (Sif[3] == 0)
     {
     Sif[3] = 11;
     if (Sif[2] == 0)
       {
       Sif[2] = 11;
       if (Sif[1] ==0)
```

```
            {
            Sif[1] = 11;
            }
          }
        }
      }

// Check if High score

  if ((differance < high_score) && (initialFlagg == 0))
    {
    high_score = differance;
    }

  // Check if My high score

  if ((differance < my_high_score) && (initialFlagg == 0))
    {
    my_high_score = differance;
    }

  // High score
  // Converting the numbers to digits

  Siffer_1000  =  (high_score/1000);
  Siffer_100   =  (high_score - 1000*Siffer_1000)/100;
  Siffer_10    =  (high_score - 1000*Siffer_1000 - 100*Siffer_100)/10;
  Siffer_1     =   (high_score - 1000*Siffer_1000 - 100*Siffer_100 - 10*Siffer_10);

  Sif[16] = Siffer_1;
  Sif[17] = Siffer_10;
  Sif[18] = Siffer_100;
```

```
Sif[19] = Siffer_1000;

// Removing leading zero's

if (Sif[19] == 0)
 {
 Sif[19] = 11;
 if (Sif[18] == 0)
   {
   Sif[18] = 11;
   if (Sif[17] == 0)
     {
     Sif[17] = 11;
     }
   }
 }

// My high score
// Converting the numbers to digits

Siffer_1000  =  (my_high_score/1000);
Siffer_100   =  (my_high_score - 1000*Siffer_1000)/100;
Siffer_10    =  (my_high_score - 1000*Siffer_1000 - 100*Siffer_100)/10;
Siffer_1     =   (my_high_score - 1000*Siffer_1000 - 100*Siffer_100 - 10*Siffer_10);

Sif[20] = Siffer_1;
Sif[21] = Siffer_10;
Sif[22] = Siffer_100;
Sif[23] = Siffer_1000;

// Removing leading zero's
```

```
  if (Sif[23] == 0)
  {
  Sif[23] = 11;
  if (Sif[22] == 0)
    {
    Sif[22] = 11;
    if (Sif[21] == 0)
      {
      Sif[21] = 11;
      }
    }
  }

}
  // Check if reset is pressed
  reset_high_score = digitalRead(9);

  if (reset_high_score == 0)
  {
    high_score = 9999;
    reset_high_score = 1;
    // Serial.print("Reset Dagens beste");
  }

  reset_my_high_score = digitalRead(8);

  if (reset_my_high_score == 0)
  {
    my_high_score = 9999;
    reset_my_high_score = 1;
    // Serial.print("Reset Min beste");
  }
```

```
BigDisplay ();

// If high_score is zero, blink the high_score and reset the high_score to 9999

if (high_score == 0)
{

 for(i = 0; i < 5; i++)
 {

  Sif[16] = 0;
  Sif[17] = 0;
  Sif[18] = 0;
  Sif[19] = 0;

  BigDisplay ();
  delay(500);

  Sif[16] = 11;
  Sif[17] = 11;
  Sif[18] = 11;
  Sif[19] = 11;

  BigDisplay ();
  delay(500);
 }

high_score = 9999;
my_high_score = 9999;
differance = 10000;
}
```

```
  initialFlagg = 0;
}

void BigDisplay ()
{

 // Serial.println("BigDisplay is Called");

 // unsigned char tall;
 byte tall;
 char c = 0;

 digitalWrite(pinCLK_1, LOW);
 delay(250);
 for (n = 11; n > -1; n--)
    {
     tall = SifCon[Sif[n]];
     // Serial.println(int(tall));
     for (m = 0; m < 8; m++)
        {
          //Serial.println(int(tall));
          if (tall & 128)
             {
               digitalWrite(pinSDI_1, HIGH);
             }
          else
             {
               digitalWrite(pinSDI_1, LOW);
             }
          delayMicroseconds(100);
          tall = tall << 1;
```

```
        digitalWrite(pinCLK_1, HIGH); // Clocking bit into shiftregister
        delayMicroseconds(100);
        digitalWrite(pinCLK_1, LOW);
      }
      delay(1);
   }

digitalWrite(pinCLK_2, LOW);
delay(250);
for (n = 23; n > 10; n--)
   {
    tall = SifCon[Sif[n]];
    // Serial.println(int(tall));
    for (m = 0; m < 8; m++)
       {
        //Serial.println(int(tall));
        if (tall & 128)
          {
           digitalWrite(pinSDI_2, HIGH);
          }
        else
          {
           digitalWrite(pinSDI_2, LOW);
          }
        delayMicroseconds(100);
        tall = tall << 1;
        digitalWrite(pinCLK_2, HIGH); // Clocking bit into shiftregister
        delayMicroseconds(100);
        digitalWrite(pinCLK_2, LOW);
       }
      delay(1);
   }
```

```
    digitalWrite(pinLE_1,HIGH);  // Latch display data 1
    digitalWrite(pinLE_2,HIGH);  // Latch display data 1
    delayMicroseconds(200);
    digitalWrite(pinLE_1,LOW);
    digitalWrite(pinLE_2,LOW);
}
```

Heftet gir en kortfattet orientering om bruk og vedlikehold av installasjonen "Nærmest null" som er utviklet som et delprosjekt under Abel-fond-prosjektet som startet i 2011 og avsluttes ved årsskiftet 2015/16. *Utvikling av rike interaktive installasjoner for bruk ved Vitensenter* har vært et samarbeid mellom Vitensenteret i Trondheim og VilVite i Bergen. I løpet av prosjektet er det utviklet to interaktive modeller ved VilVite og to ved Vitensenteret i Trondheim (ViT). Det har vært et krav at utstillingene skal handle om matematikk og være utviklet i tråd med intensjonen om "Active prolonged engagement", et prinsipp utviklet ved Exploratorium i San Fransisco..

ISBN 978-82-92088-54-8
Rev. 1.3 - 24.04.15