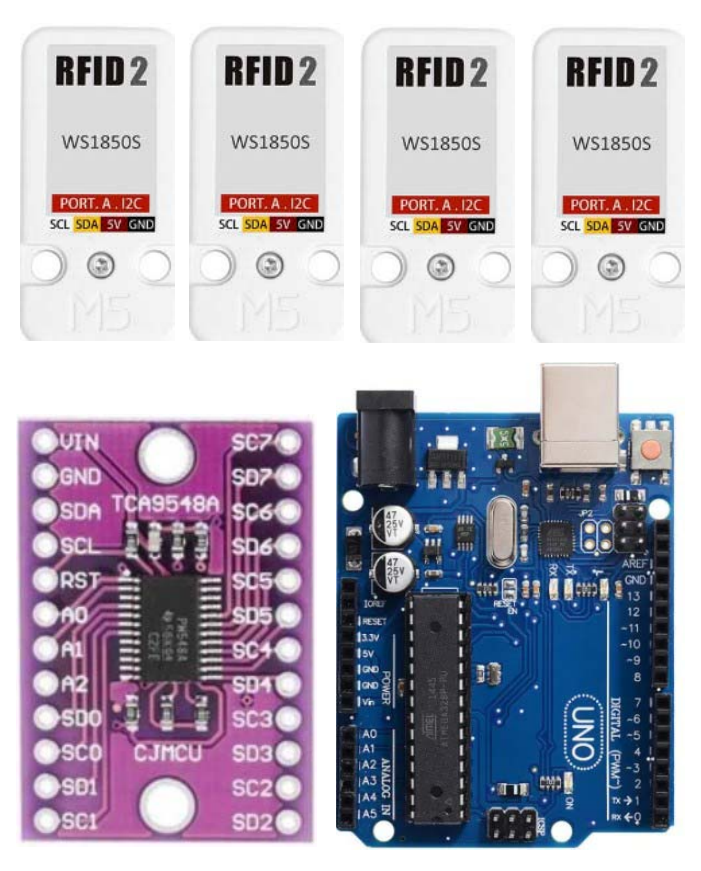
Rev 3.0

VITENSENTERET
TRONDHEIM

Nils Kr. Rossing

Arduino – Kurshefte
Bruk av RFID



Mars 2024

Denne siden er blank

Arduino – Kurshefte
Bruk av RFID

Nils Kr. Rossing

Arduino – Kurshefte, Bruk av RFID

Trondheim 2024

ISBN 978-82-92088-84-5

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
nkr@vitensenteret.com

Kongensgate 1
7011 Trondheim

Postboks 117
7400 Trondheim

Vitensenteret i Trondheim
Telefon: 72 90 90 07
http://www.vitensenteret.com/

Rev 3.0– 11.03.24

Forsidebilde: Bilder hentet fra nettet
Bilde side 3: Bilde av modellen søskenflokken (Foto: Nils Kr. Rossing)

Forord

Noen modeller i utstillingen har behov for å kjenne igjen brikker. Dette har tradisjonelt vært gjort ved å kode brikkene med magneter i unike posisjoner for så å lese av om det befinner seg en magnet eller ikke i den angitte posisjonen som forventet. Dette har vært tilfelle både for "Nærmest null" og "Søskenflokken" til Thomas Angell.

En mer elegant og fleksibel løsning vil være å bruke RFID tagger på brikkene som kan avleses ved hjelp av RFID terminaler. Dette er relativt enkelt å få til dersom man bruker en terminal og mange brikker. Utfordringen blir imidlertid en annen dersom man ønsker å benytte mange like RFID terminaler for å lese av ulike brikker plassert i forskjellige posisjoner. Heftet viser en metode å løse denne utfordringen på ved hjelp av I^2C-multiplekser.

Siden dette er et ganske generelt problem som flere vitensenter kan ha nytte av, er heftet laget som en arbeidsbok med løsningsforslag som kan brukes i opplæring i bruk av RFID på denne måten.

I forkant av kurset ble det utviklet en prototyp til en utstillingsmodell basert på bruk av I^2C buss både til avlesning av RFID-terminaler og skriving til display: Varianter av "Nærmest null". Denne er også beskrevet i heftet.

Vitensenteret i Trondheim
Mars 2024

Nils Kr. Rossing

Innhold

1 Innledning

Noen modeller i utstillingen har behov for å kjenne igjen brikker. Dette har tradisjonelt vært gjort ved å kode brikkene med magneter i utvalgte posisjoner, for så å lese av om det befinner seg en magnet eller ikke i den angitte posisjonen. Dette har vært tilfelle både for "Nærmest null" og "Søskenflokken" til Thomas Angell, "Ut og fly" oppmuntrer publikum til å velge brikker for så å legge dem på en tiltenkt plass. Legges brikken på den angitte plassen, skygger den for en lyssensor og det registreres at brikken er plassert. En slik løsning kan synes enkel, men forutsetter at belysningen er riktig og at brikken gir tilstrekkelig skygge. Dessuten kan en risikere at skygger som publikum lager over modellen, utløser en registrering.

En noe enklere løsning ville ha være å bruke RFID tagger på brikkene som kan avleses ved hjelp av RFID-terminaler. Dette er relativt enkelt å få til dersom man bruker en terminal og mange brikker. Utfordringen blir imidlertid en annen dersom man ønsker å benytte mange like RFID terminaler for å lese av ulike brikker plassert i forskjellige posisjoner. Heftet viser en metode å løse dette problemet på.

1.1 Eksempler på utstillinger der man med fordel kan bruke RFID

Heftet beskriver ulike teknologier og foreslår et valg av komponenter og programvare for å bli kjent med bruken av RFID. Heftet skisserer et kursopplegg primært rettet mot teknikere ved vitensenter som ønsker å ta i bruk denne teknologien i forbindelse med utvikling av utstillinger. Innledningsvis vil vi nevne noen eksempler på utstillinger hvor denne teknologien er godt egnet.

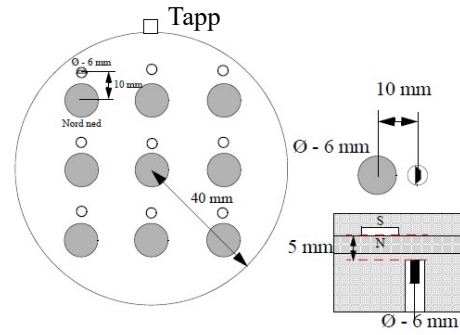
1.1.1 Eksempel 1: Søskenflokken

Thomas (Angell) har 8 søstre og brødre, tilsammen er de 9 søsken. Oppdraget går ut på å ordne dem i henhold til rekkefølgen de er født i, fra den eldste til den yngste, ut fra følgende faktaopplysninger:

- ***Thom*** er født før ***Lorents***

- ***Elisabeth*** er født sist
- ***Margrethe*** er født før ***Morten***, men etter ***Thomas Angell***
- ***Thomas Angell*** ble født rett etter ***Lorents***
- ***Rebekka*** er født etter ***Elisa***
- ***Elisa*** er født etter ***Morten***
- ***Johan*** skal stå mellom ***Elisa*** og ***Rebekka***

Utstillingsmodellen varsler med lyd og lys når riktig rekkefølge er oppnådd.



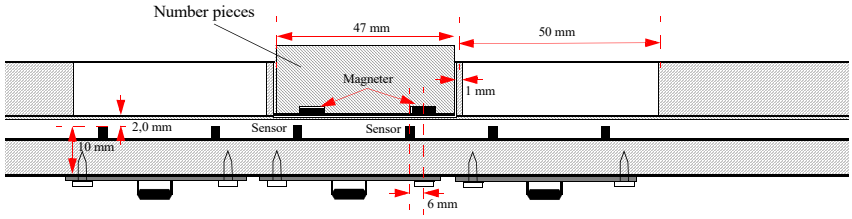
Modellen må identifisere om riktig figur står på riktig plass i rekken. Det er derfor strengt tatt ikke nødvendig å gjenkjenne alle brikkene i alle posisjoner, kun om riktig figur står i riktig posisjon. I dag er dette løst med magneter plassert i en av 9 utvalgte posisjoner og en Hall-sensor som legger en linje til jord dersom en magnet kommer i nærheten av sensoren. Magneten legges med nord ned ca. 10 mm til siden av sentrum av sensoren, der virkningen er størst. For at dette skal fungere må det legges inn en tapp slik at figuren orienteres på samme måte i alle posisjoner. Videre må magnetene legges så langt fra hverandre at sensoren ikke blir påvirket av en magnet i naboposisjonen.

Ved hjelp av RFID kan vi i prinsippet gjenkjenne alle figurer i alle posisjoner og vi kan, om ønskelig, også angi hvor nær man er en løsning.

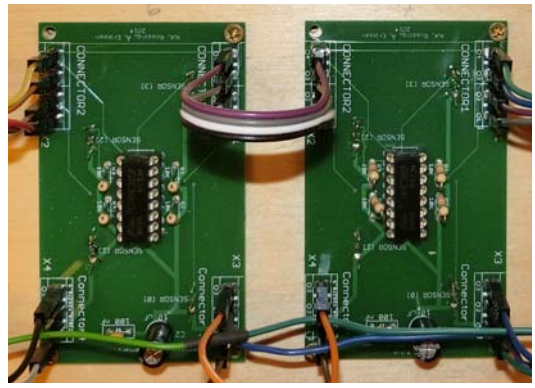
1.1.2 Eksempel 2: "Nærmest null"

Oppdraget i "Nærmest null" er å plassere de 10 brikkene 0 – 9 slik at de danner to multiplikasjonsstykker som hver gir et produkt som vises på to displayer. Produktene trekkes så fra hverandre og differansen vises på et tredje display som vist på bildet under. Øverst vises også dagens beste og personlig beste resultat på to displayer.

Også denne modellen benytter magnetisk deteksjon ved at magneter kan plasseres i fire posisjoner under hver brikke. Ved å la de fire posisjonene være med eller uten magneter, får vi i alt 16 mulige varianter. En av alternativene faller bort da ingen monterte magneter indikerer at ingen brikker ligger i posisjonen.



Siden alle brikker må kunne identifiseres i alle posisjoner, kreves det fire sensorer i hver posisjon. Det er derfor utviklet et eget sensorkort med en bussdriver som kan kobles til en 4-linjers parallell buss. Hver avlesning multiplekses inn på bussen som leses av en mikrokontroller (Arduino MEGA).



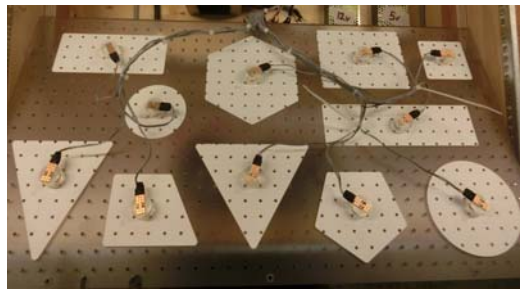
Her er det mye å spare ved å bruke RFID. Da holder det med å klister en "RFID sticker" på hver brikke som leses av en liten RFID-terminal under hver posisjon. Når sifferbrikkene er detektert i de ulike posisjonene, er det lett å beregne produktene og differansen.

For dagens system har vi erfart at posisjoneringen mellom magnet og sensor er ganske kritisk, så kritisk at en liten forskyvning av topplata eller forflytning av sensoren kan gi ustabilitet ved avlesning av brikkene.

1.1.3 Eksempel 3: "Ut og fly"

Oppdraget i denne modellen består av et optimaliseringsproblem. Vi skal ut å reise og skal pakke kofferten, men må begrense hva vi kan ta med oss. Begrensningen går på vekt (maks 20 kg) og man har valgt å optimalisere mht. maksimal verdi.

Modellen tilbyr et utvalg objekter som er angitt med vekt og verdi som vi kan velge mellom. Disse skal legges opp i kofferten og blir da automatisk registrert med vekt og verdi som fortløpende summeres opp og vises på displayer. Ved å trykke på en knapp får vi en antydning av hvor nær eller fjern vi er fra den optimale løsningen.



Teknisk er det valgt en særdeles enkel løsning. Gjenstandene har forskjellige geometrisk utforming som er gjengitt på et "bord" inne i kofferten. Ved å legge riktig geometrisk form på riktig plass vil den bli registrert med riktig vekt og verdi. Det er imidlertid ingen ting i veien for at en gjenstand kan legges på feil sted og dermed få en annen vekt og verdi enn tiltenkt. Observasjoner viser imidlertid at publikum er disiplinerte og legger brikkene der de hører hjemme.

Deteksjonen av brikkene gjøres med lyssensorer i hver posisjon som vist på bildet over, der vi ser lyssensorene på undersiden av topplaten som ligger nede i kofferten. Et problem er at modellen må settes i godt lys og dessuten må gjenstandene svartlakkeres på undersiden for at det skal bli mørkt nok for sikker deteksjon. Det kan være nok at lysstyrken i pærene som belyser modellen endrer seg over tid. Det er også et problem at en kraftig skygge over bordet vil kunne gi registrering av en gjenstand.

For denne modellen er det mye å vinne på å bruke RFID. Ved bruk av slik teknologi kan man detektere en gjenstand uansett på hvilken geometrisk form den legges. Den må imidlertid komme innenfor rekkevidden til en RFID-terminal. Modellen vil ikke være avhengig av en spesiell belysning og brikkene trenger ikke å være svartmalte på undersiden. En trenger heller ikke være redd for at skygge eller dårlig belysning skal forstyrre registreringen.

En må imidlertid ta hensyn til at bordet i kofferten må skråstilles slik at brikkene glir av når det presses luft gjennom hullene i plata. Dette er nødvendig for å tømme bordet og nullstille modellen før neste bruker kommer.

1.2 Forslag til lignende utstillingsmodeller med bruk av RFID

Det finnes en mengde slike matematisk, logiske oppgaver som kan egne seg som modeller i en utstilling. Vi ønsker å avgrense utvalget ved å kreve at bruk av elektronikk og RFID eller lignende vil heve brukerkvaliteten ved modellen. De fleste slike oppgaver vil greie seg godt uten elektronikk.

1.2.1 Matematisk, med tall og tegn

La oss inspireres av "Nærmest null" og "Ut og fly" til å undersøke andre varianter av denne typen oppgaver. Vi kan oppsummere karakteristiske trekk ved slike oppgaver på denne måten:

- Brikker med inndata som kan være tall og/eller enkle regnetegn +, –, * og /
- Modellen består av et regnestykke som gir et tallsvar
- Oppdraget går ut på å plassere brikker med tall og/eller tegn slik at tallsvaret blir riktig

"Nærmest null"

Alternativ 1: Alle sifrene fra 0 – 9 er med og kan flyttes. Tre regnetegn er plassert fast. Oppgaven går ut på flytte tallene slik at resultatet blir så nær null som mulig.

abc*de–fgh*ij = Resultat (a – j = 0 – 9)

Denne modellen kan varieres i det uendelige.

Alternativ 2: Her er et par alternative varianter:

abc + def – ghi = Resultat (a – i = 1 – 9)

Denne varianten er en kjent problemstilling som har en rekke løsninger. Den er da gjerne satt opp på denne måten abc + def = ghi. Denne varianten krever imidlertid at publikum selv regner etter om regnestykket går opp. Det er imidlertid vår hypotese at modellen blir mer attraktiv dersom vi fortløpende regner ut resultatet.

Alternativ 3: Man kan også gjøre regnestykket mer sammensatt:

ab*c + de*f + gh *i = Resultat (a – i = 1 – 9)

Om sistnevnte variant appellerer til publikum er noe usikkert.

De to sistnevnte kan lett realiseres i samme modell som er vist i kapittel 4.

Alternativ 4: Her er nok en variant:

a*b*c + d*e*f + g*h*i = Resultat (a – i = 1 – 9)

De ulike variantene er simulert i Vedlegg C, side 95.

“Nærmest 1000”

I denne varianten av spillet står alle siffer fast fra 1 – 9, 0. Mellom hvert siffer kan man sette +, – eller ingen ting. Dersom det ikke står noe så vil tallene slås sammen, slik at 2 3 blir til 23.

1±2±3±4±5±6±7±8±9±0 = Resultat

Målet er å komme så nær til et gitt tall, f.eks. 1000.

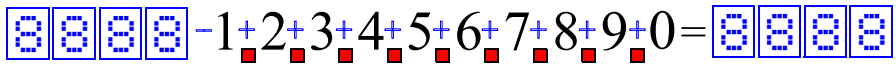
Man kan også sette begrensninger for hvor mange regnesymboler man kan få lov til å bruke.

Figuren under viser en layout for modellen:

Tallene 0 – 9 er faste og kun gravert inn i overflata. Mellom hvert tall har man en trykkbryter. Med denne kan man skifte mellom “+”, “–” og ingen tegn. Uten tegn utgjør tallene siffer i et større tall. Displayet til høyre viser resultatet av regnestykket. Tegnene “+” og “–” dannes med fire rektangulære lysdioder.

Denne modellene er realisert og dokumentert i heftet: N.K. Rossing, *Tallrekkemanipulering – Programmering med Python*, I heftet er det også utført simuleringer for å kartlegge antallet løsninger for hvert tall-svar.

En variant kan være at man setter opp et firesifret tall foran tallrekken som utgjør regnestykket. Tallet foran regnestykket velges tilfeldig fra en generator og oppgaven går ut på at tallet til venstre minus regnestykket skal bli nærmest null. Når man har oppnådd null, genereres et nytt tall og spillet begynner på nytt.



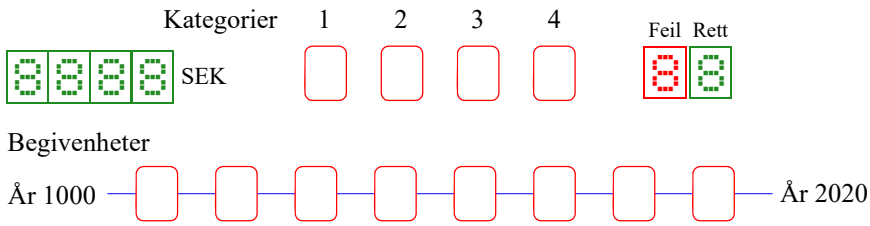
For evt. å øke engasjementet kan man utfordre brukeren til å løse tre oppgaver raskest mulig. I så fall må man ha en tidtaker.

1.2.2 Logisk, med symbolske brikker

I denne typen oppgaver handler det om å plassere brikker med symboler i riktig rekkefølge evt. på riktig sted ut fra et skjult system. Søskenflokken er et eksempel på en slik oppgave.

“Tidslinje”

Man har 8 brikker med historiske begivenheter. Disse skal plasseres riktig rekkefølge langs en tidslinje. Det er kun rekkefølgen som teller, ikke avstanden i tid.



Vi tenker oss at deltakerne kan velge mellom ulike kategorier, eller realfaglige tema. Eksempler på tema kan være:

- Årstall for oppfinnelser
- Fødselsår for kjente vitenskapsmenn
- Årstall for viktige vitenskapelige gjennombrudd

Spillet starter ved at man velger en kategori og tar alle kortene i denne kategorien, samtidig som klokka starter. Kortene legges ut langs tidslinjen i den rekkefølgen man mener er riktig. Når alle kortene er lagt på plass, vil man få vist antall feil og antall riktige. Dernest kan man diskutere rekkefølgen og bytte om på kortene. Jo nærmere i tid de ulike begivenhetene ligger, jo vanskeligere er det. Når samtlige kort ligger på rett plass stopper klokka.

Kobling av ulike kategorier

En type oppgaver som kan være ganske engasjerende og kan være svært krevende, er oppgaver som skal kombinere personer, egenskaper, preferanser o.l. etter et sett av opplysninger. Her er et eksempel på en slik oppgave[1].

I en gate er det fire hus med husnumrene 2, 4, 6 og 8.

I hvert av de 4 husene bor det en elev. Hver elev har sitt favorittfag og dessuten et eget kjæledyr.

a. Eleven i nr. 6 liker fysikk.

b. Hunden er nabo til hesten.

c. Ole liker matematikk og bor på den ene enden.

d. Erik bor mellom hesteeieren og eleven som har fysikk som favorittfag.

e. Kristine bor ved siden av katteeieren.

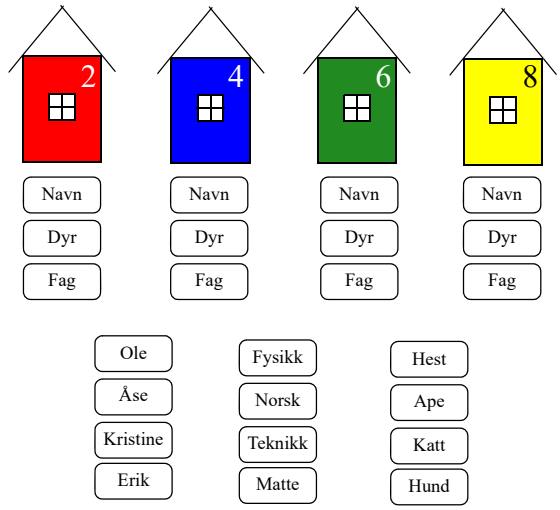
f. Eleven som bor lengst til høyre, har teknikk som favorittfag.

1. Oppgaven er hentet fra: https://www.matematikk.org/trinn5-7/tekstnott.html?tid=105107

g. Katten er nabo til eleven som liker norsk.

I hvilket hus bor Åse, og hvem eier apekatten?

En kan se for seg følgende oppsett:



Brikkene skal legges på riktig plass under hvert av husene etter det skjema som er gitt i oppgaven over.

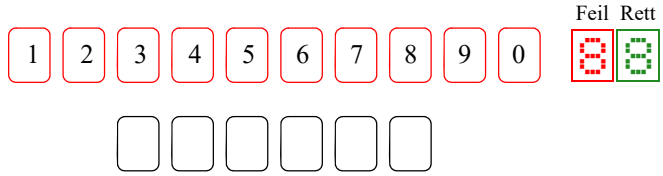
"Det mystiske tallet"[2]

Dette er en problemløsningsoppgave der man skal legge 6 siffer på riktig plass etter følgende regler:

- Tallet har 6 siffer
- Sifrene på enerplassen og tierplassen er de to minste oddetallene. De andre sifrene er partall og ingen av dem er like
- Sifferet på hundrerplassen er lik summen av sifrene på enerplassen og tierplassen
- Sifferet på tusenplassen er 2 ganger sifferet på tierplassen
- Sifferet på hundretusenplassen er det dobbelte av sifferet på hundrerplassen

2. Ideen til denne oppgaven er hentet fra Tone Skori's presentasjon "Problemløsning er bare noe vi må fikse" på Novemberkonferansen 2019

Oppsettet kan se ut som i figuren under.



Kortene kan ligge spredt ut over som vist på figuren over, eller samlet i en bunke. De seks tallene skal legges på rette plass slik at de oppfyller kravene skrevet foran. Når alle kortene er lagt vil antall rette og gale vises av de to tallene.

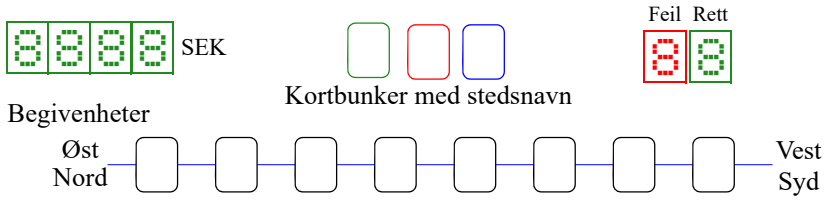
Oppgaven skal ha to løsninger?

"Søskenflokken"

Søskenflokken er av samme type, men her gjelder det å plassere dem i henhold til når de er født. Den eldste lengst til venstre og den yngste lengst til høyre. Oppgaven kan være gitt som et logisk puslespill.

"Geografisk linje"

En kan også tenke seg en oppgave der man skal plassere f.eks. byene langs Sørlandskysten fra øst til vest i riktig rekkefølge.



Her er noen flere forslag:

- Plasser stedsnavnene i riktig rekkefølge fra nord til syd
- Plasser fjordene i riktig rekkefølge fra nord til syd

Det er viktig at kortene i samme kategori har samme farge slik at de ikke så lett blandes. Dersom man har en start knapp kan man også ta tida for å ha klart alle tre kategoriene. Det er imidlertid vanskelig å konkurrere siden man gjerne lærer under veis. Man kan evt. konkurrere med seg selv.

1.3 Beskrivelse av oppdraget

I dette avsnittet skal vi beskrive oppdragene som vi skal bruke i kurset for å lære hvordan vi kan bruk RFID i utstillinger.

Oppdraget:

Det skal lages en fungerende prototyp av en liten utstillingsmodell som anvender inntil 8 RFID-terminaler for å gjenkjenne ulike brikker eller gjenstander i bestemte posisjoner. Modellen skal utfordre og engasjere publikum slik at de blir værende ved modellen over tid, slik at det legges tilrette for læring.

Deloppdrag 1 – Bli kjent med RFID 2 WS1850S

Gjør dere kjent med RFID 2 og WS1850S enheten (se avsnitt 3.1, side 47). Dere finner flere detaljer beskrevet i omtalen av MFRC522 (se avsnitt 2.3, side 28).

For veiledning se avsnitt 3.1, side 47 og 2, side 23.

Deloppdrag 2 – Oppkobling og programmering av RFID 2 via I^2C-bussen

A. I dette oppdraget skal vi koble opp en RFID 2 WS1850S enhet direkte til Arduino UNO og bruke et testprogram som leser av et ID-kort og skriver ut koden i HEX-format. Gå gjennom programmet og forsikre dere om at dere forstår hvordan det fungerer.

For veiledning se avsnitt 3.2, side 48. Løsningsforslaget finnes i vedlegg B.1, side 79.

B. Lag en liten funksjon som gjør de fire bytene i ID-koden om til et heltall (long) og skriv tallet ut i monitoren.

For veiledning se avsnitt 3.2, side 48. Løsningsforslaget finnes i vedlegg B.2, side 80.

Deloppdrag 3

Velg et ID-kort med en gitt kode som skal kunne godkjennes av programmet, andre kort skal bli avvist. Endre programmet fra oppdrag 2 slik at når det godkjente ID-kortet holdes fram for RFID-terminalen, så skal det gjenkjennes og en lysdiode tennes i 5 sekunder, deretter slukkes lyset. Andre kort skal avvises.

For veiledning se avsnitt 3.3, side 51. Løsningsforslaget finnes i vedlegg B.3, side 81.

Deloppdrag 4

Koble multiplekseren til mikrokontrolleren. Deretter kobles det 4 stk. RFID 2 til multiplekseren. Last inn I^2C-scanne-programmet og sjekk at adressene til de fire RFID 2 terminalene kommer opp i lista over tilkoblede enheter og med forventet adresse. Scanne-programmet finnes i vedlegg A.1, side 73.

For veiledning se avsnitt 3.4, side 53.

Deloppdrag 5

Skriv et programmet som leser ett eller flere RFID-kort fra terminalene og skriver ut koden til monitoren hos IDE'en. Bruk gjerne flere RFID-kort samtidig på forskjellige terminaler og sjekk at alle blir lest og skrevet ut på heksadesimal form.

Inkluder og bruk funksjon som gjør om den heksadesimale koden (4 byte) til et desimalt heltall, og legg ID-kodene inn i et array med 8 lokasjoner, en for hver av tallene 1 – 8, selv om dere foreløpig bruker kun fire av dem.

For veiledning se avsnitt 3.5, side 55. Løsningsforslaget finnes i vedlegg B.4, side 83 og B.5, side 84.

Deloppdrag 6

Monter de fire RFID 2 terminalene i en ramme som vist på figuren til høyre. Dere får utdelt 4 ID-kort påfører tallene 1, 3, 5 og 7 eller 2, 4, 6 og 8 og skal skrive et program som beregner produktet og legger det ut i monitoren, eller evt. skriver det ut på et OLED display. Produktet skal kun vises i monitoren når alle kortene er på plass.

Legg kortene slik at dere kommer nærmest produktet 900. Er det mulig å treffe 900 eksakt, evt. hvor nært klarer dere (er det mulig) å komme?

For veiledning se avsnitt 3.6, side 59. Løsningsforslaget finnes i vedlegg B.6, side 87.

Deloppdrag 7

Gå sammen to og to og koble alle 8 RFID-terminalene opp til multiplekseren. Skriv om programmet slik at det leser alle 8 terminalene og beregner de to produktene og skriver dem til monitoren når kortene legges på plass. Beregn så differansen mellom de to produktene og skriv verdien til monitoren. Flytt de 8 kortene rundt i de ulike posisjonene og forsøk å gjør differansen minst mulig og helst null. Produktene og differansen skal kun vises i monitoren når alle kortene er på plass.

For veiledning se avsnitt 3.7, side 62. Løsningsforslaget finnes i vedlegg B.7, side 90.

Deloppdrag 8

Diskuter dere i mellom, og kom opp med forslag til hva dere kan bruke denne teknologien til i en vitensenterutstilling.

2 Trådløs identifikasjon – RFID/NFC[3]

Det finnes flere teknologier som er tatt i bruk for trådløs kortdistanse identifisering. Vi skal her se på de vanligste og vise hvordan vi kan koble opp, programmere og ta dem i bruk, spesielt med tanke på bruk i utstillinger ved vitensentere.

Fra norsk Wikipedia leser vi:

> *Radiofrekvensidentifikasjon (engelsk: radio frequency identification, RFID) er en metode for å lagre og hente data ved hjelp av små enheter kalt RFID-brikker. En RFID-brikke er en liten brikke som kan festes til eller bygges inn i et produkt, et dyr eller en person. RFID-brikker inneholder antenner som gjør dem i stand til å motta og svare på radiofrekvenssignaler fra en RFID-sender. Passive brikker svarer med et svakt radiosignal og trenger ingen strømkilde, mens aktive brikker sender et kraftigere svarsignal over en mulig større avstand og behøver dermed en strømkilde.*
>
> *Kjente bruksområder for RFID er sporing, logistikk, klær, pass, billettering, elektronisk betaling, container-terminaler, varesikring / tyverialarm, evakueringssystemer, adgangskontrollsystemer og dyreidentifisering med mer.*

Det er også et annet begrep som går igjen NFC (Near Field Communication).

Fra norsk Wikipedia leser vi:

> *Nærfeltskommunikasjon (engelsk: near field communication, NFC) er en nettverksteknologi som gjør at to enheter kan kommunisere trådløst med hverandre over korte avstander (ca. 4–20 cm). Nærfeltskommunikasjon anvendes primært til å dele data, parre enheter og utføre transaksjoner, for eksempel mobilbetaling.*
>
> *Teknologien er basert på radiofrekvensidentifikasjon (RFID).*

Det kan se ut som sistnevnte er en variant av RFID, men med noe større rekkevidde og for utveksling av større datamengder. I vårt tilfellet bruker vi det primært til identifisere brikker eller enheter.

2.1 MIFARE og organisering av dataene på ID-kortet

Det finnes mange typer ID-kort og merkelapper for denne typen kommunikasjon. Vi har i denne sammenhengen konsentrert oss om "MIFARE-standarden" som er mye brukt. Teknologien ble opprinnelig lansert av firmaet MICRON (MICRON FARE), som i dag eies av NXP Semiconductors som bli skilt ut fra Philips Electronics i 2006. Teknologien bygger på standarden ISO/IEC 14443 som sender på frekvensen 13,56 MHz.

MIFARE består av i alt fire familier av ID-kort[4]:

3. https://lastminuteengineers.com/how-rfid-works-rc522-arduino-tutorial/
4. https://en.wikipedia.org/wiki/MIFARE

- *MIFARE Classic*
  Bruker en proprietær protokoll som er kompatibel med del 1–3 av ISO/IEC 14443 Type A, med en proprietær NXP-sikkerhetsprotokoll for autentisering og chiffrering. MIFARE Classic IC er kun en minnelagringsenhet, hvor minnet er delt inn i segmenter og blokker med enkle sikkerhetsmekanismer for tilgangskontroll (se under). De er ASIC-baserte og har begrenset beregningskapasitet. På grunn av kortenes pålitelighet og lave kostnad, er disse kortene mye brukt til elektroniske lommebøker, tilgangskontroll, bedrifts-ID-kort, transport eller stadion-billetter. Den bruker en proprietær NXP-sikkerhetsprotokoll (Crypto-1) for autentisering og kryptering.
- *MIFARE Pluss*
  Drop-in-erstatning for MIFARE Classic med sertifisert sikkerhetsnivå (basert på AES-128) som ble lansert i 2008 og som er fullstendig kompatibel med MIFARE Classic. MIFARE Plus etterlater imidlertid fortsatt en åpen dør for angrep, når det brukes i eldre transportsystemer som ennå ikke støtter AES. Selv om det hjelper å dempe trusler fra angrep som har brutt Crypto-1-chifferet på grunn av den relativt svake random generatoren, så hjelper det ikke mot brute force-angrep og kryptoanalytiske angrep.
- *MIFARE Ultralight*
  Rimelige IC-er som er nyttige for bruk i store volumer som offentlig transport, lojalitetskort og arrangementsbilletter.
- *MIFARE DESFire*
  Kontaktløse IC-er som samsvarer med del 3 og 4 av ISO/IEC 14443-4 Type A med et maske-ROM-operativsystem fra NXP. DES i navnet refererer til bruken av en DES, to-nøkkel 3DES, tre-nøkkel 3DES og AES kryptering; mens Fire er et akronym for Rask, innovativ, pålitelig og forbedret.

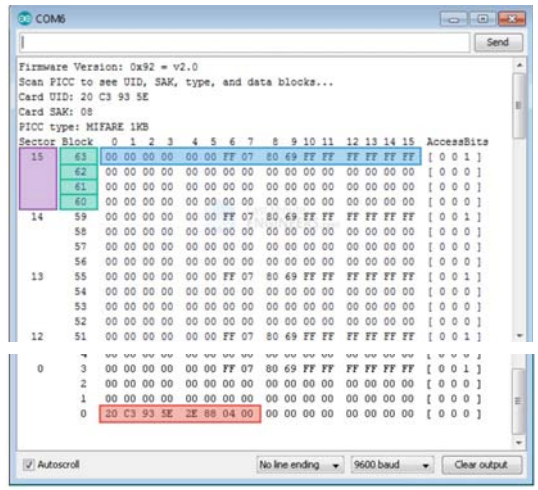
Slike kort finnes i en mengde utførelser.

ID-Kort med RFID

Nøkkelring med RFID

Klistremerke med RFID

Organisering av data hos minnekortene



Ofte har ID-kortene 1kbyte dataminne hvor vi kan lagre data som senere kan leses. Disse dataene er organisert i 16 *sektorer,* hver med fire *blokker,* der hver blokk inneholder 16 *byte,* hvilket betyr at det totalt kan lagres 16 x 4 x 16 byte = 1024 byte (1 kbyte). Figuren til høyre viser de siste sektorene og blokkene.

I sektor 0, blokk 0 finner vi en unik identifikasjonskode for det aktuelle ID-kortet som består av i alt 8 byte, innrammet på figuren til høyre. Denne koden går under betegnelsen *Manufacturer block* eller *Manufacturer data* og inneholder *data om produsenten av brikken* og en *Unik identifier* som kan brukes til å identifisere ID-kortet (brukeren av kortet). En skal være forsiktige med å skrive over denne koden da det kan låse kortet slik at man senere ikke får tilgang.

2.2 WS1850S[5]

Kretsen RFID2 opererer på frekvensen 13,56 MHz som de andre. Den har en innebygd WS1850S-brikke, som muliggjør ulike funksjoner knyttet til RFID-teknologi.

RFID2 anvender magnetisk induksjon som muliggjør kontaktfri og toveis kommunikasjon. RFID2 støtter flere funksjoner, inkludert lesing og skriving av kortdata, kortgjenkjenning, opptak av informasjon og koding av RF-kort.

RFID2 og WS1850S er ganske kompatibel med RS522 (MFRC522) og har ellers følgende egenskaper:

- Frekvens: 13.56 MHz
- I^2C data rate: “Fast mode”: opp til 400 Kbit/s; “High-speed mode”: opp til 3400 kbit/s
- I^2C adresse: 0x28
- Sende/mottaker buffer: 64 bytes
- Protokoller som støttes: ISO14443A, MIFARE og NTAG
- Arbeidstemperatur: – 20°C til 85°C
- Garantert lagringstid: > 10 år

5. Informasjonen er hentet fra: https://shop.m5stack.com/products/rfid-unit-2-ws1850s

- Typisk maksimal leseavstand: < 20 mm
- Støtter følgende teknologi plattformer: Arduino, UIFlow (Blockly, Python)
- 2 x LEGO kompatible hull

RFID2-Terminalene anvender kabel med HY2.0-4P kontakter og er kun ment for bruk av I^2C-buss for kommunikasjon mellom RFID-terminalen og mikrokontrollerkortet. La oss derfor se litt nærmere på installasjon av nødvendige biblioteker og bruk av I^2C-bussen for denne brikken.

2.2.1 Bruk av I^2C og biblioteker

I dette avsnittet skal se hvordan vi kan bruke I^2C buss til å overføre data fra sensorer utstyrt med denne typen kommunikasjonsgrensesnitt, primært ved bruk av RFID 2 WS1850S. Til dette bruker vi spesiallagede biblioteker som må installeres dersom de ikke alt finnes blant bibliotekene som følger med Arduino editoren (IDE).

Den såkalte I^2C bussen[6] består av to kommunikasjonslinjer som gjør det mulig å sende data etter hverandre (på serieform) mellom ulike kretser.

Standardbiblioteket for I^2C følger med Arduino programvaren, men krever at man inkluderer "header"-filen: "wire.h" øverst i koden, som medfører at programmet får tilgang til en del funksjoner knyttet til I^2C-kommunikasjonslinjene.

I neste avsnitt skal vi se hvordan vi kan laste ned og installere biblioteker for bruk av I^2C-bussen (`Wire.h`) og RFID2 WS1850S (`MFRC522_I2C.h`).

2.2.2 Biblioteket – Wire.h:

Også I^2C-bussens bibliotek må lenkes inn i koden når den skal brukes. Bussen funger ved at en "master" styrer kommunikasjonen. Vanligvis er dette Arduino mikrokontrollerkortet som automatisk blir definert som master ved kommandoen *Wire.begin();* kommandoen i setup-funksjonen.

Biblioteket inneholder noen funksjoner som vi skal forsøke å beskrive her:

"Header file":

```
#include <Wire.h>
```

"Header" kommandoen knytter I^2C-biblioteket til programmet slik at kompilatoren vet hvilket bibliotek funksjonene skal hentes fra.

```
Wire.begin();
```

Denne funksjonen definerer Arduino mikrokontrolleren som master for kommunikasjonen, dvs. at all kommunikasjon med sensorer og andre periferikretser som er koblet til I^2C-bussen styres av Arduino'en. Siden Arduino'en er masteren, er det ikke nødvendig å definere en adresse (`Wire.begin(<adresse>);`) for denne. Evt. kan adressen være et tall mellom 0 – 127 som identifiserer masteren på bussen.

```
Wire.beginTransmission(<adresse>);
```

6. I^2C = I2C = IIC = Inter IC Communication

Med funksjonen *Wire.beginTransmission(<adresse>);* kan man starte en dataoverføring til en enhet på bussen med den spesifiserte adressen.

I tillegg har hver enkelt enhet koblet på bussen sin unike adresse. Denne blir initiert i setup()-funksjonen eller som en del av biblioteket for den aktuelle enheten. RFID WS1850S har alle adressen 0x028, hvilket i vårt tilfelle er et problem siden vi ønsker å koble mange like enheter til bussen. Dette har vi løst med I^2C multiplekseren, se avsnitt 2.5, side 38.

I de neste avsnittene skal vi behandle noen varianter av RFID kretser. Siden de er relativt like kan det være greit å vite at det finnes flere og at de til dels kan brukes om hverandre.

2.2.3 Installasjon av bibliotek for bruk av WS1850S

Som for de andre RFID-terminalene trenger vi et bibliotek. Siden WS1850S er kompatibel med RC522 så kan vi bruke dette bibliotekene for å kommunisere med Arduino via I^2C-bussen.

- **Last ned biblioteket**
  En måte å gjøre dette på er å laste ned biblioteket: MFRC522_I2C.h fra f.eks.: https://github.com/semaf/MFRC522_I2C_Library. Biblioteket installeres på vanlig måte.
- **Inkluder nødvendige biblioteker i programmet:** Vi inkluderer bibliotekene helt først i programfila:

  ```
  #include <Wire.h>
  #include "MFRC522_I2C.h"
  ```

  Siden vi skal bruke I^2C-bussen trenger vi også biblioteket Wire.h som er standard i Arduino IDE og dermed ikke trengs å installeres spesielt.
- **Deklarasjon:** Dernest deklareres variabler som refererer til noen av portene i Arduino'en. Vi deklarerer også instansen mfrc522( ) av klassen MFRC522. Disse deklarasjonene legges inn før setup()-funksjonen:

  ```
  #define RST 3   // Configurable, see typical pin layout above
  MFRC522 mfrc522(0x28, RST);   // Create MFRC522 instance.
  ```

  Vi legger merke til at I^2C adressen og reset inngangen (RST) inngår i argumentet til instansen. Vi legger inn RST til tross for at vi ikke kommer til å bruke den.
- **Initialisering:** Så skal vi initialisere RFID kortet og SPI-bussen som gjøres i setup()-funksjonen:

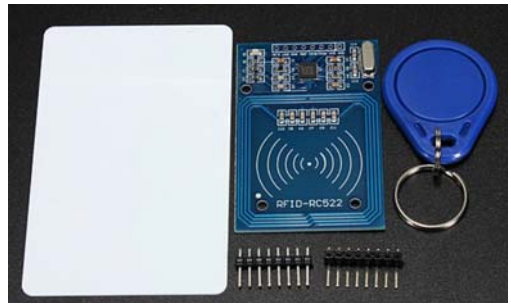
  ```
  void setup() {
    Serial.begin(9600);
    Wire.begin();
    mfrc522.PCD_Init();
    Serial.println("Leser ID");
  }
  ```

  I tillegg til å initialisere RFID-kortet må vi også initialisere I^2C-bussen (Wire.begin();).

Da skal alt være klart til å skrive resten av programmet som avleser RFID-kortene.

Resten er beskrevet i deloppdrag 2 i avsnitt 3.2 på side 48.

2.3 MFRC522[7]



I dette avsnittet skal vi beskrive en RFID-terminal som benytter data brikken MFRC522[8], som har en rekkevidde (0–5 cm). RFID-terminalen leveres gjerne som et sett som består av en sender-mottakerenhet (leseenhet) og et ID-kort og en plast tag som er merket med et unike ID-nummer som kan avleses av terminalen.

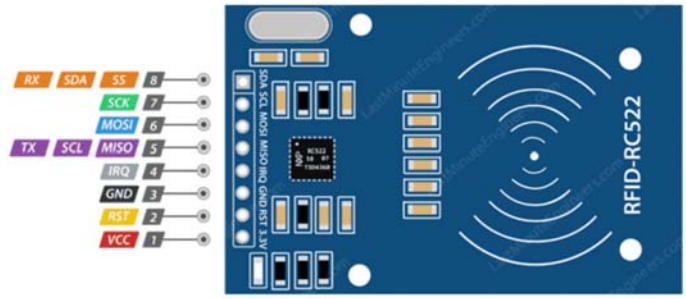
RFID-terminalen opererer på frekvensen 13,56MHz og ID-kortet er utstyrt med en antenne som er avstemt til samme frekvens. ID-kortet inneholder en liten microchip som har 1kbyte lager for å holde informasjon, som omtalt foran. Siden ID-kortet ikke har noen egen energikilde, må energien som skal til for å drive microchipen, overføres fra leseenheten via antennen og til chipen, som dermed tilføres tilstrekkelig energi til å svare RFID-terminalen med sin unike identifikasjonskode. RFID-terminaler av denne typen kan både lese av kortets ID og øvrige data, og skrive inn ny ID og data.

RFID-terminalen kan kobles til Arduino'en via SPI-bussen og kan overføre datarater opp til 10Mbit/s. ISO14443A beskriver protokollen, datarammestruktur, feildeteksjon og krypteringsmetode. Til tross for at anbefalt forsyningsspenning er 3,3 V, tåler leseenheten 5V spenning på terminalene og kan derfor kobles direkte til en Arduino med 5V forsyningsspenning. Strømtrekket er normalt på mellom 13 – 26 mA.

Microchipen som er montert på ID-kortet håndterer også I^2C-buss, men ikke alle "breakout" kortene er tilrettelagt for bruk av I^2C-buss. Skal man bruke I^2C-bussen må man i slike tilfeller gjøre modifikasjoner på kortet, se avsnitt 2.3.4 på side 31.

Oppkobling og programvare finnes på nettsiden til *Last Minute Engeneers*: https://lastminuteengineers.com/how-rfid-works-rc522-arduino-tutorial/ Her finner man også mye interessant stoff om RFID og MFRC522.

Figuren til høyre viser pinningen til kortet MFRC522. Terminalene til SPI-bussen finner vi på pinne 5 (MISO), 6 (MOSI) og 7 (SCK). Videre tilbyr kretsen et interruptsignal (IRQ) som vil gi interrupt til Arduino'en dersom det kommer et ID-kort i nærhtene av leseenheten. Dermed slipper Arduino'en å sjekke status for leseenheten, men kan avbryte sine gjøremål bare når det foreligger et interrupt.



7. https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf
8. https://www.kultogbillig.no/index.php?_route_=RFID-RC522-RF-IC-Card-Sensor-Modul&search=RFID

Reset (RST) vil, når pinnen legges lav (GND), slå kretsen helt av, hvilket kan være fornuftig dersom man ønsker å spare strøm. Den vil i denne tilstanden ikke reagere på ID-kort. Idet spenningen kommer på igjen, resettes kortet.

2.3.1 Oppkobling med bruk av SPI-buss

Dersom man ønsker stor lese- og skrivehastighet kan man benytte SPI bussen som anvender fire linjer, som også deles med de to andre databussene:

MISO / SCL / Tx pinnen fungerer som *Master-In-Slave-Out* (MISO) når SPI-bussen er aktiv, og som *seriedata* utgang (Tx) når vi benytter UART-en.

MOSI (Master Out Slave In) fungerer som *datainngang* når vi benytter SPI-bussen.

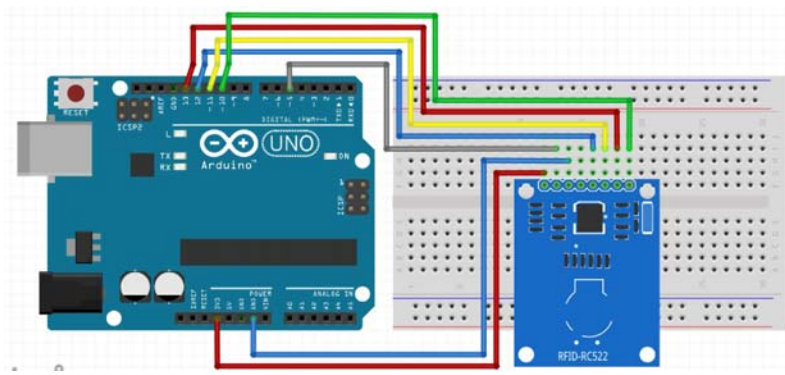
SCK (Serial Clock) fungerer som *serieklokkeinngang* fra SPI-bussens master, f.eks. Arduinoe'en.

SS / SDA / Rx pinnen fungerer som inngangssignal når kretsen benytter SPI-bussen og som seriedata inn (Rx) når UART-bussen benyttes. Denne pinnen er også vanligvis markert med et kvadrat for å indikere at den kan brukes som referanse for de øvrige pinnene.

SPI-bussen er tilknyttet forskjellige porterer hos de ulike Arduino-kortene. Tabellen under gir en oversikt over SPI-pinningen på tre vanlig brukte Arduino-kort:

Arduino	MOSI	MISO	SCK	SS
UNO	11	12	13	10
NANO	11	12	13	10
MEGA	51	50	52	53

Figuren til høyre viser hvordan man kan koble opp MFRC522 til Arduino UNO hvor kommunikasjonen skjer ved hjelp av SPI-bussen.



2.3.2 Programmering med bruk av SPI-buss

- **Installasjon av biblioteket:** Vi skal se hvordan vi kan kommunisere med kortet ved hjelp av SPI-bussen. For å få til dette på en enkel måte laster vi ned biblioteket: MFRC522.h fra f.eks.: https://www.arduinolibraries.info/libraries/mfrc522 Biblioteket installeres på vanlig måte.

- **Inkluder nødvendige biblioteker i programmet:** Vi legger inn bibliotekene:

  ```
  #include <SPI.h>
  #include <MFRC522.h>
  ```

  helt først i programfila

- **Deklarasjon:** Dernest deklareres variabler som refererer til noen av portene i Arduino'en:

  ```
  #define RST_PIN 5   // Configurable, see typical pin layout above
  #define SS_1_PIN 10 // Configurable, take a unused pin, only HIGH/LOW required,
                      diffrent to SS 2
  #define NR_OF_READERS   2
  MFRC522 mfrc522[NR_OF_READERS];   // Create MFRC522 instance.
  ```

  Vi deklarerer også instansen mfrc522[ ] av klassen MFRC522. Disse deklarasjonene legges inn før setup()-funksjonen.

- **Initialisering:** Så skal vi initialisere RFID kortet og SPI-bussen som gjøres i setup()-funksjonen

  ```
  SPI.begin();                              // Init SPI bus
  mfrc522[0].PCD_Init(SS_1_PIN, RST_PIN); // Init the MFRC522 card
  ```

  Vi legger merke til at her angir vi hvilke pinner vi bruker til *SS* og *RST*. I tillegg til å initialisere RFID-kortet må vi initialisere SPI-bussen.

- **Avlesning av ID-kort** og returner ID-nummer ved hjelp av følgende funksjon som kalles fra loop()-funksjonen med kallet: `ID = LesAvRFID();` Selve avlesningen skjer fra funksjonen:

  ```
  int LesAvRFID()
  {
    int R;
    ID = 1000;
    // Sjekk RFID-brikke
    if (mfrc522[0].PICC_IsNewCardPresent() && mfrc522[0].PICC_ReadCardSerial())
    {
       ID = mfrc522[0].uid.uidByte[1];
       mfrc522[0].PICC_HaltA(); // Halt PICC
     }
   return ID;
  }
  ```

2.3.3 Flere RFID-terminaler på samme SPI-buss

Som den oppmerksomme leser kanskje har lagt merke til i koden over, så skal det være mulig å koble flere RFID-terminaler på samme SPI-buss:

```
MFRC522 mfrc522[NR_OF_READERS];   // Create MFRC522 instance.
```

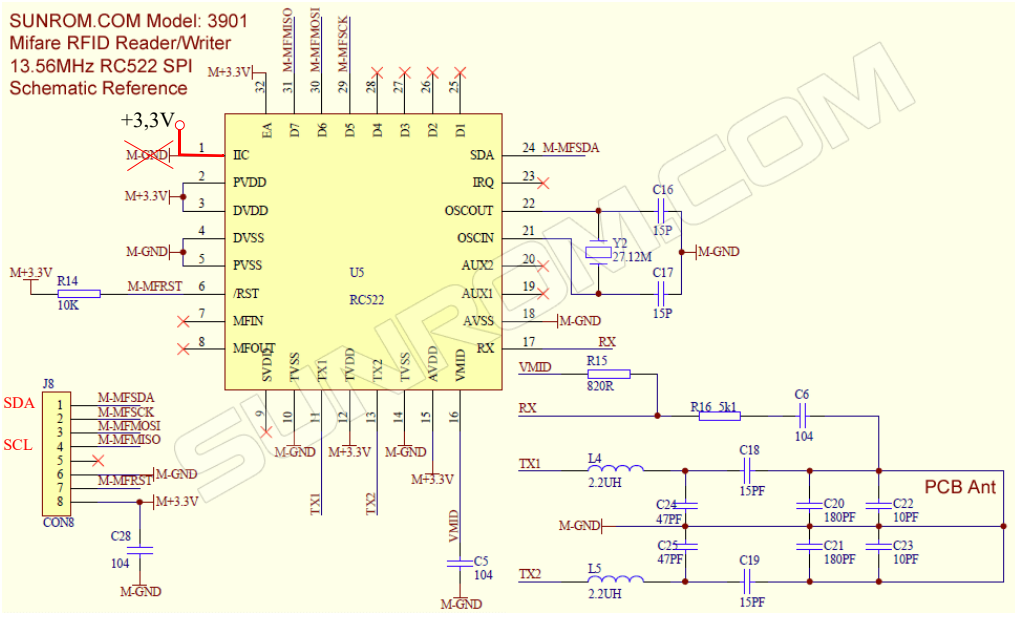
Konstanten `NR_OF_READERS` angir antall lesere eller RFID-terminaler og jeg har funnet eksempler på at noen har parallellkoblet fire terminaler, imidlertid ikke uten problemer[9], men dog løsbart. Man må da bruke en enable-linje til hver av terminalene for å slå dem av mellom hver lesing på bussen. På denne måten kan flere terminaler veksle på å bruke bussen.

2.3.4 Bruk av I²C-buss

Tilsvarene SPI-bussen burde vi kunne koble opp RFID-terminalen ved bruk av I²C-bussen. Også i dette tilfellet er bussen tilknyttet ulike pinner på de forskjellige Arduino-kortene:

Arduino	SDA	SCL
UNO	A4	A5
NANO	A4	A5
MEGA	D20	D21

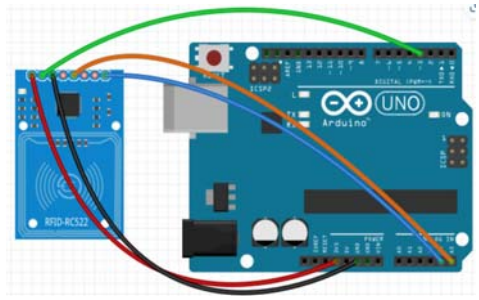
Som nevnt foran så er kretsen MFRC522 forberedt for bruk av I²C-buss, men ikke alle "breakout" kortene tilbyr denne bussen, som vist på tegningen av kretskortet under. Det går imidlertid an å fikse dette om man er litt hendt[10].



9. https://forum.arduino.cc/t/solved-problem-with-reading-from-multiple-mfrc522/698438
10. https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf side 9

For at dette breakout-kortet skal fungere med I^2C, må man gjøre noen modifikasjoner:

- IIC – Pin 1 – må legges høy og ikke lav
- SDA – Pin 24 – koblet til kontakten J8 – 1 (M-MFSDA)
- SCL – Pin 7 – koblet til kontakt J8 – 4 (M-MFMISO)



Vi kan da i prinsippet nå I^2C-bussen dersom vi legger Pin 1 høy i stedet for lav og kobler opp som vist på figuren til høyre.

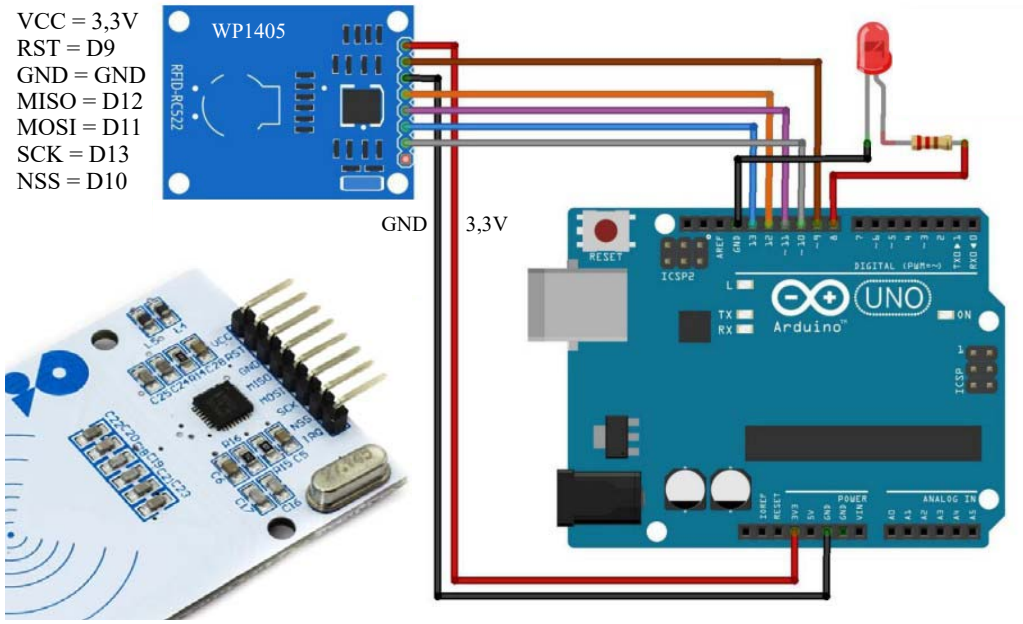
Et godt alternativ kan imidlertid være å velge et annet breakout-kort med tilrettelagt I^2C-buss.

2.3.5 Skriv til RFID kort

I noen tilfeller kan det være aktuelt å skrive til et RFID-kort, f.eks. dersom man ønsker å endre ID-koden, eller legge inn data på kortet. De fleste terminaler har denne muligheten, men slettes ikke alle RFID-kort er forberedt for skriving. For at dette skal være mulig må kortet eller "tagen" være skrivbar og ha rett antall ID-bytes. Vi har her brukt kort med fire byte innkjøpt fra Amazon.com[11].

11.https://www.amazon.com/Changeable-Rewritable-13-56MHz-Mifare-Control/dp/B0895Y5LGT/ref=sr_1_2

For å programmere kortene brukte vi en WP1405 Velleman RFID-kort fra ELFA[12]. Denne kretsen kobler vi opp med en Arduino UNO som vist på figuren under[13].

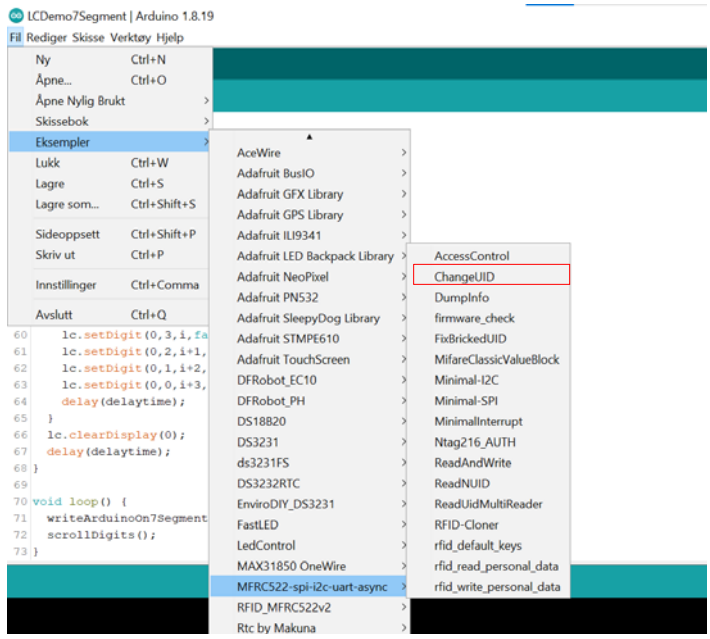


Vi velger å bruke biblioteket MFRC522-spi-i2c-UART-asynk som vi installerer i IDE'en:

12.https://www.elfadistrelec.no/no/rfid-lese-og-skrivemodul-arduino-kompatibel-13-56-mhz-velleman-wpi405/p/30260977

13.Figuren er hentet fra: https://www.elfadistrelec.no/Web/Downloads/_m/an/WPI405_eng_man.pdf

Under eksempeloversikten finner vi filen: *ChangeUID.ino* (Se også vedlegg A.3, side 76).



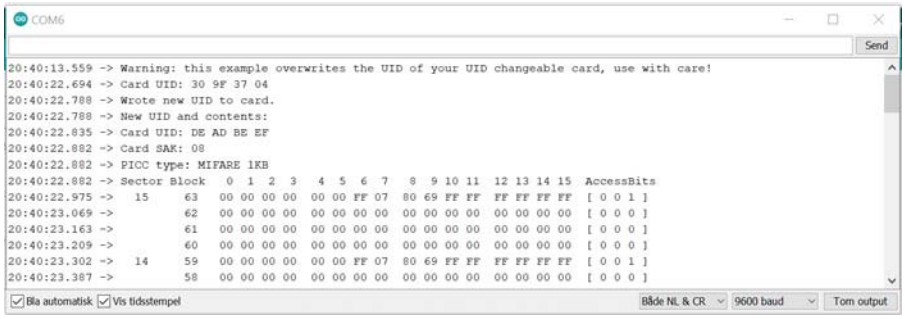
Vi laster opp programmet og går ned til linje 35, der vi kan legge inn vår nye ID-kode:

```
26 #include <SPI.h>
27 #include <MFRC522.h>
28
29 #define RST_PIN   9     // Configurable, see typical pin layout above
30 #define SS_PIN    10    // Configurable, see typical pin layout above
31
32 MFRC522 mfrc522(SS_PIN, RST_PIN);   // Create MFRC522 instance
33
34 /* Set your new UID here! */
35 #define NEW_UID {0xDE, 0xAD, 0xBE, 0xEF}
36
37 MFRC522::MIFARE_Key key;
38
39 void setup() {
```

Her kan vi skrive inn vår nye UID i Hex-kode. Default er satt til:

```
/* Set your new UID here! */
#define NEW_UID {0xDE, 0xAD, 0xBE, 0xEF}
```

Holder vi et skrivbart kort bort til RFID-terminalen, vil vi få følgende utskrift:



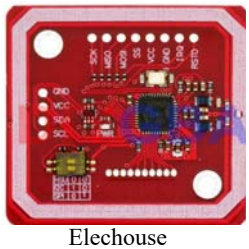
Vi får da skrevet ut både den gamle UID og den nye UID. I tillegg skrives alle de 1000 bytene som er lagret på kortet ut i monitoren.

Legg spesielt merke til at den gamle UID'en skrives over. En må derfor være forsiktige med kort der man ønsker å beholde ID-koden intakt.

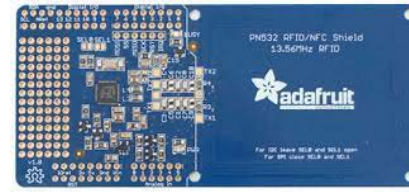
For kun å dumpe innholdet uten å skrive til en RFID-brikke, kan man hente eksempelet: *Dump-info.ino* (Se også vedlegg A.2, side 74).

2.4 PN532[14]

En alternativ RFID-krets som brukes mye er PN532. Denne finnes i ulike utforminger og fra forskjellige fabrikanter. Figuren under viser tre eksempler[15].


Elechouse


Sunfounder



Adafruit

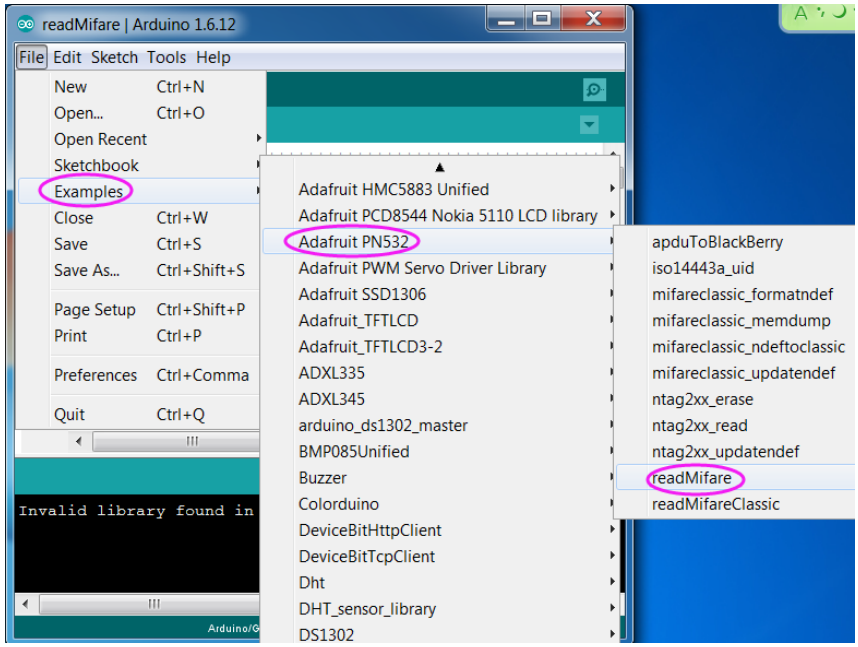
PN532 som MFRC522, en integrert krets for nærfeltskommunikasjon (NFC) som også benytter frekvensen 13,56 MHz. Ved hjelp av en liten bryter på kortet kan man velge mellom kommunikasjonsbussene I^2C-, SPI- og UART (HSU – High Speed UART). En nivåskifter gjør det mulig å bruke disse RFID-terminalene, både med 3,3V og 5V. I tillegg støtter den både RFID-lesing og -skriving. Enkelte, slik som Sunfounder støtter også NFC-funksjon hos Android-telefon, noe som gjør den praktisk for trådløs tilkobling. Maksimal avstand for lesing er ca. 3 cm.

Kortene er godt tilpasset bruk med Arduino. Adafruit har laget et bibliotek som passer til RFID kretser av denne typen.

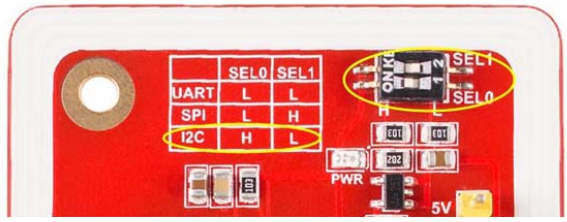
14. Informasjonen og noen av bildene er hentet fra: http://wiki.sunfounder.cc/index.php?title=PN532_NFC_RFID_Module
15. Se også: https://how2electronics.com/interfacing-pn532-nfc-rfid-module-with-arduino/

Biblioteket installeres på vanlig måte f.eks. ved bruk av Sketch->Including Library->Manage Libraries, og skrive PN532 i søkeruta, og velg INSTALL:

Man vi da få tilgang til ulike eksempelkoder ved å velge File—>Examples —>Adafruit PN532—> ReadMifare og åpne ReadMifare.ino som vist i figuren under:



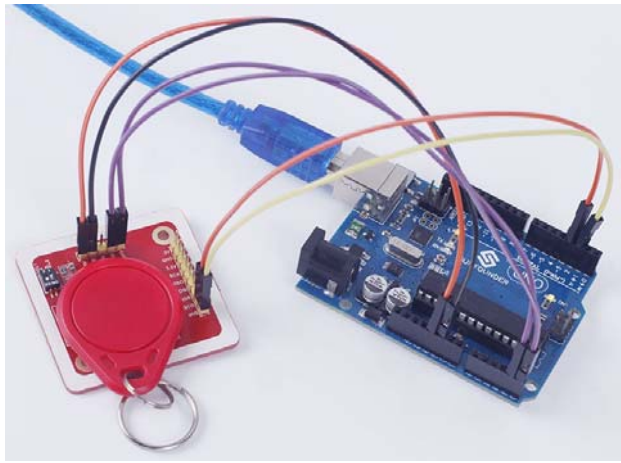
Dernest må vi stille inn bryteren på kretsen for ønsket kommunikasjon. I eksempelet er det valgt I^2C, men vi kan også velge SPI eller UART (HSU).



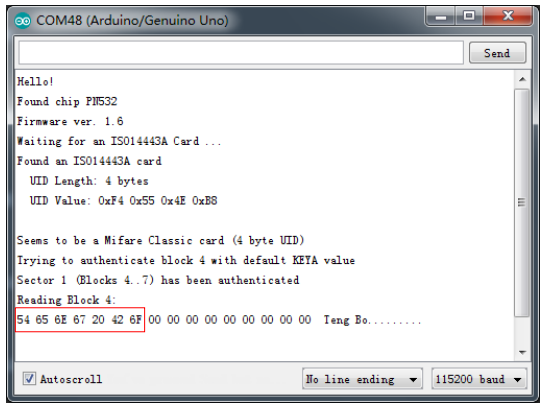
I programmet velger vi inn de riktige kommandolinjene avhengig av hvilken kommunikasjons-buss man velger å bruke.

```
55 // Use this line for a breakout with a software SPI connection (recommended):
56 //Adafruit_PN532 nfc(PN532_SCK, PN532_MISO, PN532_MOSI, PN532_SS);
57
58 // Use this line for a breakout with a hardware SPI connection.  Note that
59 // the PN532 SCK, MOSI, and MISO pins need to be connected to the Arduino's
60 // hardware SPI SCK, MOSI, and MISO pins.  On an Arduino Uno these are
61 // SCK = 13, MOSI = 11, MISO = 12.  The SS line can be any digital IO pin.
62 //Adafruit_PN532 nfc(PN532_SS);
63
64 // Or use this line for a breakout or shield with an I2C connection:
65 Adafruit_PN532 nfc(PN532_IRQ, PN532_RESET);
66
```

En oppkobling med Arduino UNO kan se slik ut.



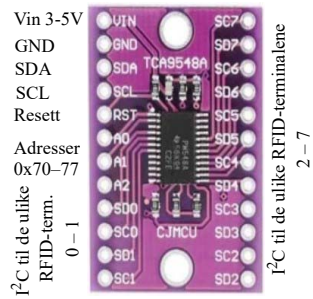
Kjører vi programmet kan resultatet bli slik som vist på figuren under, dersom vi bruker et kort med 7 byte ID-kode. Det er imidlertid vanligere å bruke en ID-kode på 4 byte.



2.5 I²C multiplekser – TCA9548A

I²C-bussen er praktisk å bruke siden mange ulike sensorer kan kobles parallelt på de to linjene SDA og SCL. For å skille data fra de ulike komponentene langs bussen, så gir man sensorene eller aktuatorene unike adresser som gjør at en master, gjerne mikrokontrolleren, kan henvende seg til en spesifikk enhet. Alle enheter som normalt kobles til en I²C-buss må derfor ha en unik adresse, gjerne gitt av fabrikanten.

Problemet oppstår dersom vi skal koble flere like komponenter til bussen. Hos noen komponenter kan man sette opp ulike adresser ved å lodde over eller bryte noen broer. Enkelte display har gjerne denne muligheten, da det er ganske vanlig at flere display kobles på samme bussen for å vise større tall eller tekst. Imidlertid ser det ikke ut til at dette er tilfelle for RFID-terminaler[16]. Disse har en fast adresse som ikke kan endres. I vårt tilfelle 0x28.



For nettopp et slikt tilfelle er det utviklet en I²C-multiplekser som gjør det mulig å koble flere like I²C-kretser på samme buss. Kretsen TCA9548A er en slik krets som gjør det mulig å dele bussen. Adafruit har dessuten laget et "breakout" kort som gjør det lett å ta i bruk kretsen for eksperimentell utforsking. Figuren over til høyre viser kretsen med pinning.

Spenning tilføres kretsen via Vin og GND, man kan bruke fra 3,3 – 5,0 V. SDA og SCL er seriedata og -klokke fra mikrokontrolleren. For Arduino UNO er dette henholdsvis A4 og A5. Det er mulig å endre I²C-adresse for multiplekseren. Dersom disse ikke tilkobles, vil den ha standard adressen 0x70, dernest kan man sette opp i alt 8 adresser fra 0x70 – 0x77 ved å legge A0, A1 og A2 høy eller lav (A0, A1, A2 – 0,0,0 (0x70) til A0, A1, A2 – 1,1,1 (0x77). Dette gjør det mulig å sette opp i alt 8 slike multipleksere med tilsammen 64 I²C-tilkoblinger med samme I²C-adresse. Dernest kan vi f.eks. koble inntil 8 RFID-terminalene til SD0, SC0 – SD7, SC7.

For å kunne bruke I²C-multiplekseren må vi installere et bibliotek.

2.5.1 Installasjon av bibliotek for bruk av TCA9548A

Det er flere måter å installere biblioteker på. Her skal vi viset to alternativer for å installere biblioteket for å håndtere bruken av I²C-multiplekseren TCA9548A.

Her er to måter å installere et bibliotek på hos henholdsvis Arduino IDE versjon 1.8.19 og versjon 2.3.2:

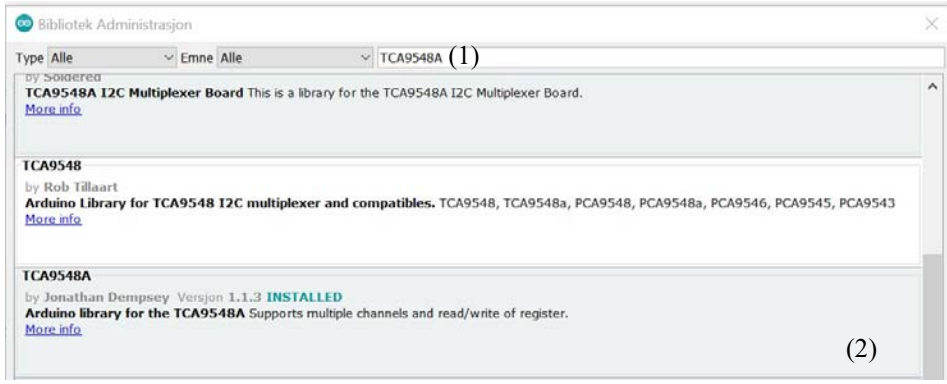
1. **Ved hjelp av Manage Libraries**

   Arduino IDE versjon 1.8.19

   Fra menylinja velger vi *Sketch/Include library/Manage Libraries* og skrive inn den aktuelle komponenten (TCA9548A) i søkefeltet (1). Dernest velger vi INSTALL nederst til høyre i

16. Vi vil ikke påstå at det ikke finnes enheter med variabel I²C-adresse, men vi har ikke funnet noen så langt.

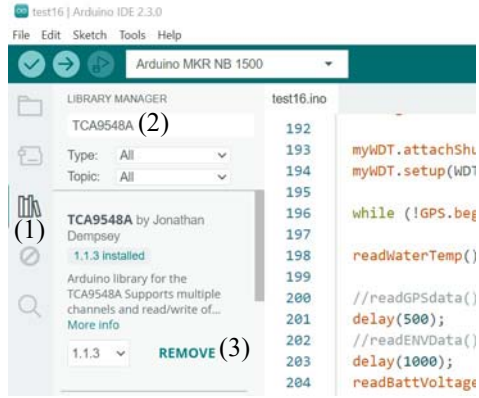
feltet (2). Vi har valgt biblioteket laget av *Jonathan Dempsey* som er det tredje alternativet på lista.



Ofte finnes det flere biblioteker for samme krets, da må man gjøre et valg og se om det fungerer tilfredsstillende. Noen ganger er det lurt å antyde til hvilken teknologi vi skal bruke kretsen. Dette gjelder f.eks. når vi bruker Teensy, ESP eller MKR-serien som alle benytter Arduino IDE. Det er ikke alltid at biblioteker beregnet til Arduino også passer for ESP eller MKR, men at man må installere en variant. Vi har valgt

Arduino IDE versjon 2.3.2

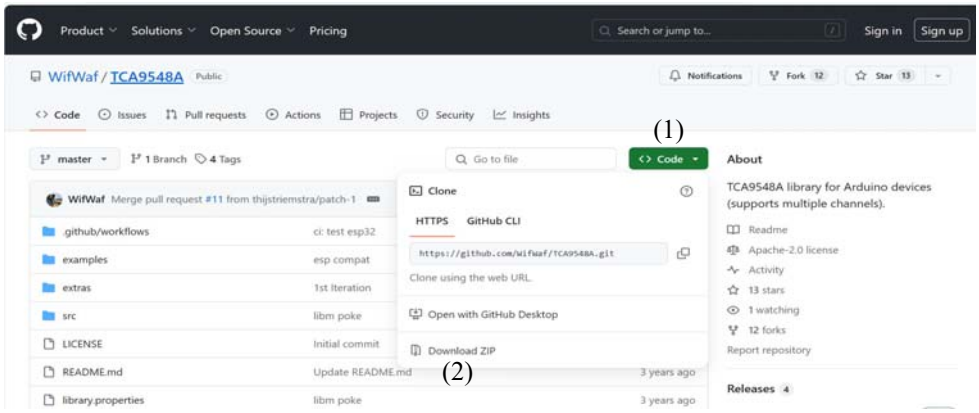
I denne nye versjonen velger man ikonet for bibliotek fra menyen til venstre (1) skriver inn navnet på kretsen (2) og velger ett av forslagene. Vi har valgt biblioteket laget av Jonathan Dempsey. At det står REMOVE på figuren under betyr bare at biblioteket alt er installert og kan evt. fjernes om det er ønskelig.



Som et alternativ kan vi installere biblioteket dersom vi har det pakket som en .zip-fil.

2. Installasjon av .zip bibliotek

Først må man finne biblioteket som gjerne ligger i et arkiv (Github)[17]. Last ned biblioteket i pakket form (*.zip) (figuren under (1) og (2)). Pakken inneholder header-filer (*.h), biblioteksfunksjoner (*.cpp) og gjerne eksempler på bruk av sensoren. Er man heldig inneholder den også en beskrivelse av biblioteksfunksjonene. Man finner gjerne biblioteket på nett ved å søke etter den aktuelle kretsen etterfulgt av ordet *library* om adressen ikke er kjent.



Så installerer man .zip biblioteket ved å velge *Sketch/Include library/Add .ZIP library* fra menylinjen.

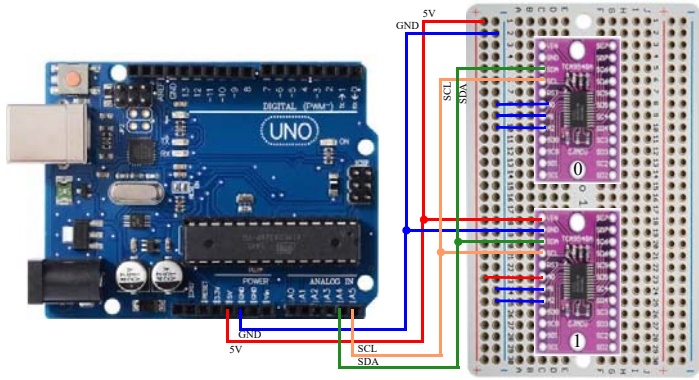
3. **Inkluder headerfilen (*.h) i programmet**
   Det aktuelle biblioteket inkluderes i programmet ved å velge:
   Sketch/Include Library/<og det aktuelle biblioteket>
   Dermed er header-filen på plass i programmet.

2.5.2 Parallellkobling av flere TCA9548A

Siden hver enkelt TCA9548A kan adresseres med tre adresselinjer (A0, A1 og A2) så kan man koble flere i parallell og dermed hente data fra inntil 64 enheter vi I^2C-bussen, til tross for at en - hetene har samme I^2C-adresse.

17.https://github.com/WifWaf/TCA9548A

Figuren under antyder hvordan vi kan koble to multipleksere i parallell, ved hjelp av adresselinjene



De to multiplekserne er satt opp med henholdsvis adresse 0x70 og 0x71 ved hjelp av adresse-linjene A0 og A1.

Vi kan skille de to multplekserne fra hverandre ved å gi objektene forskjellig adresse og navn:

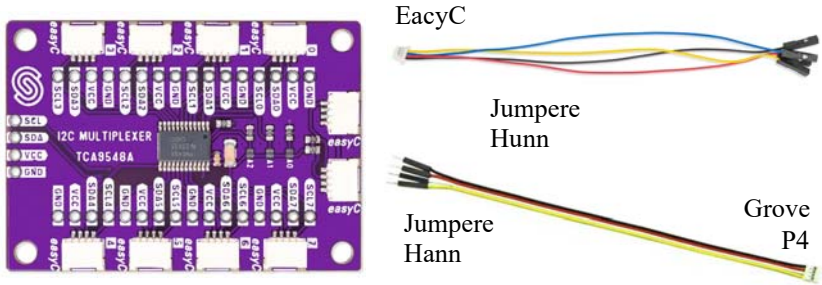
```
TCA9548A I2CMux0(0x70);        // Address can be passed into the constructor
TCA9548A I2CMux1(0x71);        // Address can be passed into the constructor
```

Dernest bruker vi de to objektene avhengig av hvilken av de to multiplekserne vi ønsker å bruke: I2CMux0 og I2CMux1.

2.5.3 "Breakout" kort med TCA9548A og tilkobling med kontakter

Firmaet *Soldered Electronics* i Kroatia har laget et "breakout" kort med TCA9548A og med kontakter som gjør det lett å koble to eller flere I^2C-enheter til flere like RFID-terminaler eller sensorer. Det forutsetter imidlertid at man bruker kontakter som passer til sensorene, eller i vårt tilfelle, RFID-terminalene.

Figuren under viser multipleksern fra Soldered som selges for under 6€. Det finnes imidlertid ingen direkte overgang fra EasyC som er kontakten som brukes på multiplekserkortet, og Grove P4 kontakten, slik at det er mulig å koble seg direkte fra multiplekseren og rett til RFID-kretsen. Imidlertid finnes det overganger til jumper-kontakter, hann og hunn, slik at en overgang er mulig, om enn noe tungvint.



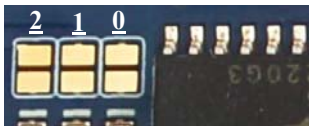
2.6 Display

Selv om vi i dette undervisningopplegget har benyttet monitoren i Arduino editoren(IDE) så vil det i en utstilling være aktuelt å bruke display. Her finnes det en rekke alternative løsninger. Vi har imidlertid valgt å presentere et 8x8 punktdisplay som vi har benyttet en del ved Vitensenteret i Trondheim. Det gir mulighet til begrenset grafikk i tre fager og benytter I^2C-bussen for å overføre informasjon.

2.6.1 Punkt- eller matrisedisplay – 902 - Bicolor LED Square Pixel Matrix

Kort omtale:

Her skal vi omtale to-farge 8 x 8 punktdisplay som kan programmeres til å vise røde, grønne eller gule punkter.: Bicolor LED Square Pixel Matrix med I^2C Backpack fra Adafruit[18] og kan nås via en I^2C buss med 8 programmerbare adresser, se til høyre på bildet til høyre. Hvert display er 32 x 32 mm (1,2" x 1,2"). Normalt leveres selve displayet og "bakpakken" med driverkretsen ??? separat og må loddes opp. Her er det spesielt viktig å passe på at displayet monteres riktig vei på bakpakken, da det er lett å montere den feil. Bakpakken gjør at displayet kan programmeres via I^2C-buss. Ved hjelp av programkommandoer kan lysstyrken til displayet settes i 16 trinn.



210	210
000 – 0x70	100 – 0x74
001 – 0x71	101 – 0x75
010 – 0x72	110 – 0x76
011 – 0x73	111 – 0x77

1 – betyr en strap (lodding)

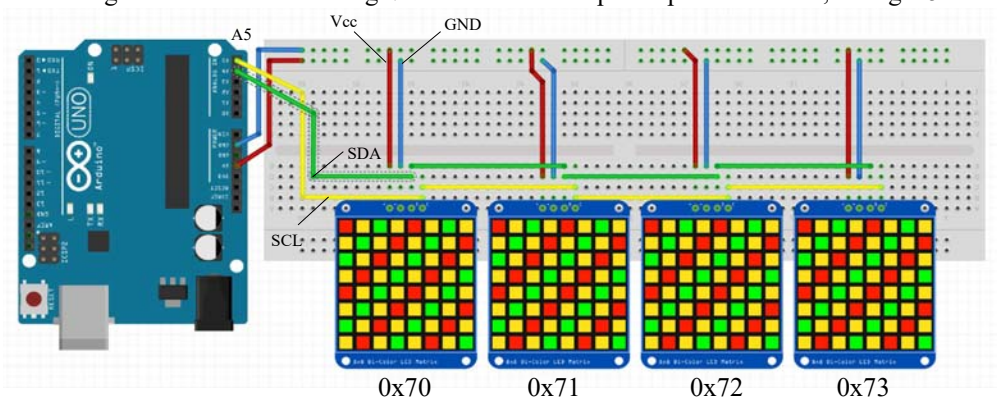
Adresseringen gjøres ved å lodde forbindelser mellom koblingsflatene på bakpakken. Grunnadressen er satt til: 0x70. Ved å strappe flatene som vist på figuren til venstre vil man kunne sette opp i alt 8 adresser fra 0x70 – 0x77.

En grundigere omtale av dette med flere displayer fra Adafruit finnes her: https://www.elfadistrelec.no/Web/Downloads/_t/ds/Adafruit_902_eng_tds.pdf?pid=${product.code}

18. ELFA Distrelec: https://www.elfadistrelec.no/en/bicolor-led-square-pixel-matrix-with-i2c-backpack-adafruit-902/p/30091224

Oppkobling

Oppkoblingen er ganske enkel ved at man kobler alle de ønskede displayene i parallell. Dvs. at alle SDA og SCL kobles sammen og føres til tilsvarende porter på Arduino'en, A4 og A5.



Programmering:

Displayene krever at man installerer biblioteket: Adafruit_LEDBackpack som kan hentes fra: https://github.com/adafruit/Adafruit_LED_Backpack. Dette gjelder bibliotekene: `Adafruit_GFX.h` og `Adafruit_LEDBackpack.h` i tillegg til biblioteket som trengs for å bruke I^2C-buss.

- ***Inkluder bibliotekene i programmet***
  Øverst i programmet inkluderer man de to bibliotekene i tillegg til biblioteket for å bruke I^2C-bussen. Dermed får programmet adgang til alle funksjonene som trengs for å skrive til displayet.

```
// Inkludering av biblioteker
#include <Wire.h>                    // Inkluder bibliotek for kommunikasjon via I2C
#include <Adafruit_GFX.h>            // Inkluder bibliotek for å skrive til display
#include <Adafruit_LEDBackpack.h> // Inkluder bibliotek for å skrive til display
```

  Disse legges inn helt i starten av programmet før deklarasjon av variabler og `setup()`.

- ***Deklarasjon av displayet***

  Dernest må vi deklarere, dvs. gi hvert display et navn (matrix0, matrix1 ...) og en type (Adafruit_BicolorMatrix), i prinsippet kan vi ha flere displayer med forskjellige navn og adresser.

```
// Punktdisplay
Adafruit_BicolorMatrix matrix0 = Adafruit_BicolorMatrix();
Adafruit_BicolorMatrix matrix1 = Adafruit_BicolorMatrix();
Adafruit_BicolorMatrix matrix2 = Adafruit_BicolorMatrix();
Adafruit_BicolorMatrix matrix3 = Adafruit_BicolorMatrix();
```

- ***Sett adressen til displayet***

Displayene får hver sin adresse som programmet må vite om, denne er 0x70, 0x71, 0x72 og 0x73 i heksadesimal kode.

```
matrix0.begin(0x70);  // pass in the address
matrix1.begin(0x71);  // pass in the address
matrix2.begin(0x72);  // pass in the address
matrix3.begin(0x73);  // pass in the address
```

Adressen settes i en initialiseringsfunksjon som vist over. Dette gjøres normalt i setup()-funksjonen siden kun er nødvendig å gjøre det en gang ved oppstart av programmet.

- **Skriv til displayene**

  Så må man skrive til hvert av displayene. Skal man skrive et siffer må dette sifferet legges i en heltallsvariabel og legges inn i skriverutinene. La oss anta at det gjelder `Sif0`:

```
matrix0.setTextColor(color1);// Sett farge
matrix0.clear();             // Slett det som står på displayet
matrix0.setRotation(3);      // Roter sifferet slik at det står rett
matrix0.setCursor(1,0);      // Plasser startstedet
matrix0.print(Sif0);         // Angir hvilket tall (siffer) som skal skri-
ves
matrix0.writeDisplay();      // Skriv til display
```

  `matrix0.setTextColor(color1);` Først setter man fargen som kan ha verdiene: LED_-GREEN, LED_YELLOW, LED_RED. Det angis som en to-farge display med rødt og grønt, men ved å blande rød og grønn får man gul slik at det i praksis vil være et tre-farge display.

  `matrix0.clear();` Dernest slettes alt som står på displayet slik at det er klart for en ny visning.

  `matrix0.setRotation(3);` Så roterer vi sifferet så det står i ønsket posisjon. Sifferet kan roteres 0°, 90°, 180° or 270° ved å benytte parameterne 0, 1, 2 og 3.

  `matrix0.setCursor(1,0);` Så settes startpunktet for markøren, der skrivingen starter. Denne kommandoen gir mening dersom den brukes for å skrive til et display med flere linjer.

  `matrix0.print(Sif0);` Denne kommandoen setter opp det tallet eller den bokstaven som skal sendes til displayet.

  `matrix0.writeDisplay();` Den siste kommandoen sender det grafiske uttrykket til displayet

Alle alfanumeriske tegn er tilgjengelig via print-kommandoen. Dersom vi ønsker andre grafiske uttrykk som f.eks. siffer med komma eller symboler eller ikoner så må vi tegne dem selv. I denne sammenhengen lages et 8x8 array som fylles med 0'ere og 1'ere og som gjengir det symbolet eller det ikonet vi ønsker å gjengi. Der vi har skrevet en 1'er der tennes punktet, der vi har skrevet en 0'er, der er punktet slukket.

Under har vi tegnet tre ikoner kalt BLID, MUTT og SUR. Disse er som navnene sier et smilefjes, et mutt fjes og et surt fjes. Legg merke til at ansiktene er tegnet liggende i matrisen hvilket er litt tilfeldig, en må bare passe på å snu dem rett vei ved visning på displayet.

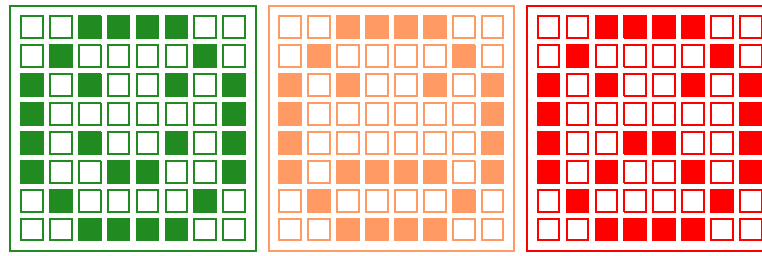
```
static const
uint8_t PROGMEM
  BLID[8] =
  { B00111100,
    B01000010,
    B10101001,
    B10000101,
    B10000101,
    B10101001,
    B01000010,
    B00111100 };
```

```
static const
uint8_t PROGMEM
  MUTT[8] =
  { B00111100,
    B01000010,
    B10100101,
    B10000101,
    B10000101,
    B10100101,
    B01000010,
    B00111100 };
```

```
static const
uint8_t PROGMEM
  SUR[8] =
  { B00111100,
    B01000010,
    B10100101,
    B10001001,
    B10001001,
    B10100101,
    B01000010,
    B00111100 };
```

For å skive ut disse byttes linjen: **`matrix0.print(Sif0);`** ut med linjen: **`matrix3.drawBitmap(0, 0, SUR, 8, 8, color2);`** slik at det blir stående:

```
matrix3.setTextColor(color2);
matrix3.clear();
matrix3.setRotation(4);
matrix3.drawBitmap(0, 0, SUR, 8, 8, color2);
matrix3.writeDisplay();
```



Hvor 0, 0, angir hvor tegningen skal begynne (topp, venstre). SUR er tegningen som skal tegnes, og 8, 8, angir størrelsen 8x8 piksler. Kommandoen gir kanskje mer mening dersom antallet piksler hos displayet er større. Figuren over viser ansiktene BLID, MUTT og SUR som f.eks. kan gi bruker tilbakemelding rett eller galt svar, ev. om svaret var hverken helt rett eller galt, men kan bli bedre. Vi kan også bruke farger for å forsterke uttrykket.

Flere kommandoer i biblioteket fra Adafruit kan finnes her: https://learn.adafruit.com/adafruit-gfx-graphics-library/graphics-primitives

3 Gjennomgang av oppdragene

Vi skal nå gjennomgå deloppdragene, oppdrag for oppdrag.

3.1 Deloppdrag 1 – Bli kjent med RFID 2, WS1850S

RFID 2 WS1850S er en RFID integrert krets som har svært mange likhetstrekk med MFRC522. Kretsene kan derfor gå om hverandre også når det gjelder programvare. For mer informasjon om hvordan RFID fungerer se kapittel 2, side 23.

Spesifikasjonene for RFID 2 WS1850S kan kort oppsummeres:

- Arbeidsfrekvens: 13.56 MHz
- I^2C data rate: Fast mode: Opp til 400 kbit/s; High-speed mode: Opp til 3400 kbit/s
- Protokoller: ISO14443A, MIFARE og NTAG
- Temperaturområde: -20°C – 85°C
- Data lagring: > 10 år
- Leseavstand: < 20 mm
- Plattform: Arduino, UIFlow (Blockly, Python)
- 2 x LEGO kompatible hull

3.1.1 Bakgrunn for valg av denne kretsløsningen

Kretsen leveres normalt i en pakning 48 x 24 x 8 mm med en 20 cm lang kabel med HY2.0-4P (Seeed Grove) kontakter i begge ender. Tilkoblingen er praktisk dersom man har kort med slike kontakter. Det finnes imidlertid kabler[19] som egner seg bedre for uttesting som vist nederst på figuren til høyre. Her er noen fordeler med nettopp denne komponenten:

- Rimelig og foreløpig lett tilgjengelig. Leveres av ELFA Distrelec til kr. 53,40 + MVA[20].
- Benytter I^2C buss som gjør den enkel å koble opp.
- Finnes praktiske overganger til mer hensiktsmessige kabler for eksperimentoppsett.
- Finnes ferdige biblioteker som kan benyttes.

19. https://www.elfadistrelec.no/en/grove-pin-male-jumper-to-grove-seeed-studio-110990210/p/30118352
20. Mars 2024

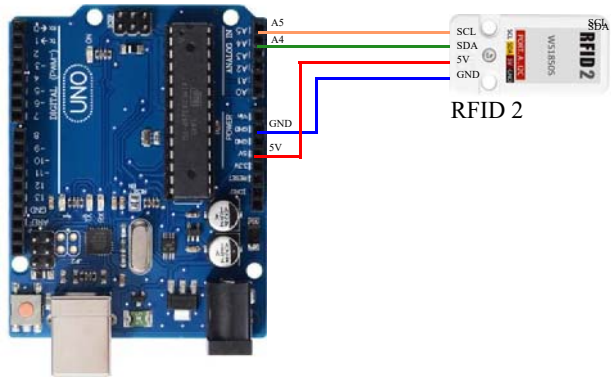
- Liten 48 x 24 x 8 mm, kan også demonteres og tar da enda mindre plass. Vi ser kretskortet som er tatt ut av plastkabinettet lengst til høyre på figuren under (42 x 20 x 6 mm), som er trengs for at den skal fungere

3.2 Deloppdrag 2 – Oppkobling og programmering av én RFID 2 via I²C

I dette oppdraget skal vi koble opp én RFID 2 enhet direkte til Arduino UNO og skrive et test-program som leser av et ID-kort og skriver ut koden både i Hex og som heltall.

3.2.1 Oppkobling av en RFID 2, direkte til Arduino

Vi begynner enkelt ved å koble en RFID 2 til Arduino UNO'en som vist på figuren under. Til dette bruker vi en jumper med en fire-pin Grove-kontakt i den ene enden og fire Dupont jumpere (hann) i den andre enden.



3.2.2 Programmering av WS1850S ved bruk av I²C-buss

Siden WS1850S og MFRC522 kan gå om hverandre kan vi også bruke biblioteker for MFRC522 til WS1850S

- **Installasjon av biblioteket:** Vi skal se hvordan vi kan kommunisere med kortet ved hjelp av I^2C-bussen. For å få til dette på en enkel måte, laster vi ned biblioteket: MFRC522_I2C.h fra f.eks.: https://github.com/semaf/MFRC522_I2C_Library. Biblioteket installeres på vanlig måte.
- **Legg inn nødvendige biblioteker i programmet:** Vi legger inn bibliotekene:

  ```
  #include <Wire.h>
  #include "MFRC522_I2C.h"
  ```

  helt først i programfila.
- **Deklarasjon:** Dernest deklareres variabler som refererer til noen av portene i Arduino'en. Vi deklarerer også instansen mfrc522( ) av klassen MFRC522. Disse deklarasjonene legges inn før setup()-funksjonen:

  ```
  #define RST 3   // Configurable, see typical pin layout above
  MFRC522 mfrc522(0x28, RST);   // Create MFRC522 instance.
  ```

  Vi legger merke til at I^2C adressen og reset inngangen (RST) inngår i argumentet til instansen. Vi legger inn RST til tross for at vi ikke kommer til å bruke den.
- **Initialisering:** Så skal vi initialisere RFID kortet og SPI-bussen som gjøres i setup()-funksjonen:

  ```
  void setup() {
    Serial.begin(9600);
    Wire.begin();
    mfrc522.PCD_Init();
    Serial.println("Leser ID");
  }
  ```

  I tillegg til å initialisere RFID-kortet må vi også initialisere I^2C-bussen (Wire.begin();).
- **Avlesning av ID-kort** Selve avlesningen skjer i loop()-funksjonen:

  ```
  void loop() {
    // Se etter et nytt RFID-kort og les av om det er i nærheten
    if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial())
    {
      // Skriv ut UID
      Serial.print(F("Card UID: "));
      for (byte i = 0; i < mfrc522.uid.size; i++)
      {
       if (mfrc522.uid.uidByte[i] < 0x10) // Legg til en "0" om verden er < 0x10
        Serial.print("0");
        Serial.print(mfrc522.uid.uidByte[i], HEX);
        Serial.print(" ");
      }
      Serial.println();
      mfrc522.PICC_HaltA();
  ```

```
}
```

Først sjekker programmet om det er et nytt kort i nærheten av RFID-sensoren (`mfrc522.PICC_IsNewCardPresent()`), i tillegg sjekkes det om kortet lar seg lese (`mfrc522.PICC_ReadCardSerial()`). Om begge disse er oppfylt så leses av hvor mange byte ID'en består av (`mfrc522.uid.size`), hos oss er det normalt 4 byte. Dernest skrives kortets ID-kode ut i Hex-format (`Serial.print(mfrc522.uid.uidByte[i], HEX);`). Dersom en av bytene har en verdi på under 0x10 (16 DEC) så legges det til en "0" foran siste siffer slik at det alltid skrives ut 4 x 2 siffer. Dersom vi ønsker kun å få returnert ID'en, så kan vi bruke funksjonen (`mfrc522.PICC_HaltA();`) som sjekker om det er et nytt kort eller om det fortsatt er det "gamle# kortet som ligger der. Om det siste er tilfelle så skrives ikke verdien ut på nytt.

- Kjør programmet og sjekk at det kjører som normalt og klarer å lese kortet og skrive ut koden i HEX-format.

3.2.3 Fra 4 byte til en "*unsigned long*"

Dersom vi skal sammenligne ID'en fra et kort med et register, så må vi sjekke at alle fire bytene er identisk. Det er derfor enklere å sjekke ett stort tall enn fire små. Siden en *long* er fire byte lang, så passer det godt å gjøre om kortets ID til et desimalt tall. Dette kan vi gjøre på følgende måte i funksjonen: `convertFourByteToLong()`

Her overfører vi `mfrc522.uid.uidByte[i]`til funksjonen, som kan gjøres om til et heltall på flere måter.

1. Dersom vi antar at UID[0] er det minst signifikante sifferet, mens UID[3] er det mest signifikante sifferet, så vil det totale tallet ha 32 binære siffer og kunne bli et særdeles stort desimalt tall (FF FF FF FF = 4 294 901 759). Ønsker vi å beregne dette tallet så kan vi gjøre det slik:

```
unsigned long convertFourByteToLong(unsigned long UID[4])
{
  unsigned long val = 0;
  val += UID[0];
  val += 256*UID[1];
  val += 65536*UID[2];
  val += 16777216*UID[3];
  return val;
}
```

Vi ser at $256 = 2^8$, at $65536 = 2^{16}$ og at $16777216 = 2^{24}$. I 2 talls systemet så fall tilsvarer dette å flytte tallene henholdsvis 8, 16 og 24 plasser *mot venstre*. For at dette skal være mulig må vi sørge for at tallene har tilstrekkelig med siffer å ta av slik at tallene, når de blir riktig store ikke faller "utfor kanten".

Vi kan da alternativt skrive:

```
unsigned long convertFourByteToLong(unsigned long UID[4])
{
  unsigned long val = 0;
  val += UID[0];
  val += UID[1] << 8;
  val += UID[2] << 16;
```

```
  val += UID[3] << 24;
  return val;
}
```

Vi velger å bruke unsigned long slik at vi unngår negative tall. Flytting av bit er vesentlig mindre ressurskrevende enn å utføre en multiplikasjon.

2. Alternativt kan vi betrakte de fire bytene som fire enkeltstående tall. Summerer vi de fire 8 bits tallene, får vi ett tall som er lettere å håndtere. Det største tallet vi da kan få er 4*255 = 1020. Med denne metoden kan vi imidlertid risikere at to ulike ID-koder gir samme desimaltall. Dette kan skje dersom de fire tallene er like, men med en annen rekkefølge. Eller at tallene er forskjellige, men at summen tilfeldigvis blir den samme.

```
long convertFourByteToLong(byte UID[4])
{
  long val = 0;
  val += UID[0];
  val += UID[1];
  val += UID[2];
  val += UID[3];
  return val;
}
```

Siden vi har full kontroll på ID-kodene på RFID-kortene, kan vi sørge for at dette ikke blir noe problem. Inntil videre velger vi derfor metode 2 da den er kjappere.

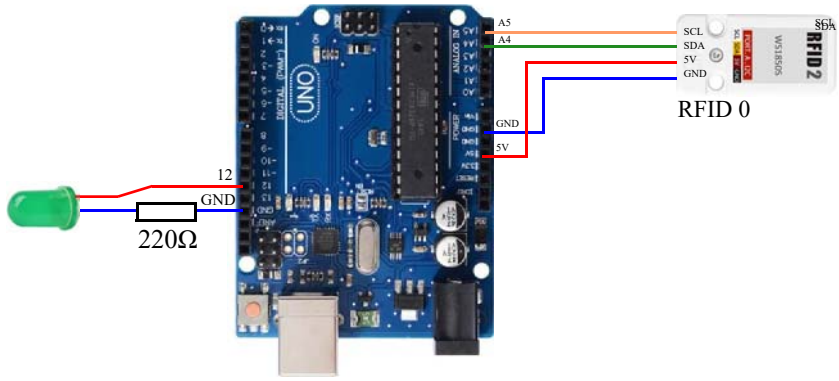
Til slutt returneres den beregnede verdien (`return val;`). Se vedlegg B.2, side 80 for løsningsforslag for testprogrammet.

3.3 Deloppdrag 3 – Program som tenner lysdiode ved visning av ID-kort

I dette oppdraget skal vi bruke programmet utviklet i forrige deloppdrag til å tenne en lysdiode når et akkreditert kort holdes opp mot RFID-terminalen, Andre kort skal avvises. Lysdioden tennes i 5 sekunder før den igjen slukker. Lysdioden kunne like godt være en elektronisk låsemekanisme som gjorde at akkrediterte personer fikk adgang til et rom ved visning av kortet.

3.3.1 Oppkobling

Vi tar utgangspunkt i oppkoblingen fra deloppdrag 2 og kobler opp en grønn lysdiode til pinne 12 som vist på figuren under:



3.3.2 Programmering

Vi tar utgangspunkt i programmet fra deloppdrag 2 og legger til noen kommandoer.

1. **Deklarasjon av variable**
   Vi velger å deklarere to variabler, en som holder ID'en til det akkrediterte kortet og en som holder ID'en til kortet som leses.

   ```
   long ID_kode = 0;
   long ID_akk = 668; // legg inn koden til det akkrediterte kortet
   int pinLED = 12;
   ```

   Dessuten deklarerer vi en variabel som holder port nummeret til lysdioden, vi velger port D12

2. **pinMode**
   Vi må også huske å varsle om at port 12 skal være en utgang. Det gjør vi i setup()-funksjonen:

3. **Sjekk akkreditering og tenn lysdioden**
   Vi leser av ID-koden fra kortet og gjør om HEX-kodene til et heltall. Når det er gjort, sjekker vi den avleste koden mot den akkrediterte koden. Om den avleste stemmer med den akkrediterte så tennes lysdioden i 5 sekunder.

Løsningsforslaget finnes i vedlegg B.3, side 81.

Forslag til tilleggsoppdrag

Her er noen flere oppdrag som kan løses:

1. Suppler med en rød lysdiode. Tenn den røde lysdioden dersom kortet ikke er akkreditert.
2. Bytt ut den grønne lysdioden med en servo med en bom som åpner seg i 5 sekunder dersom kortet er akkreditert.
3. Løs oppdraget dersom det er to kort som er akkreditert.

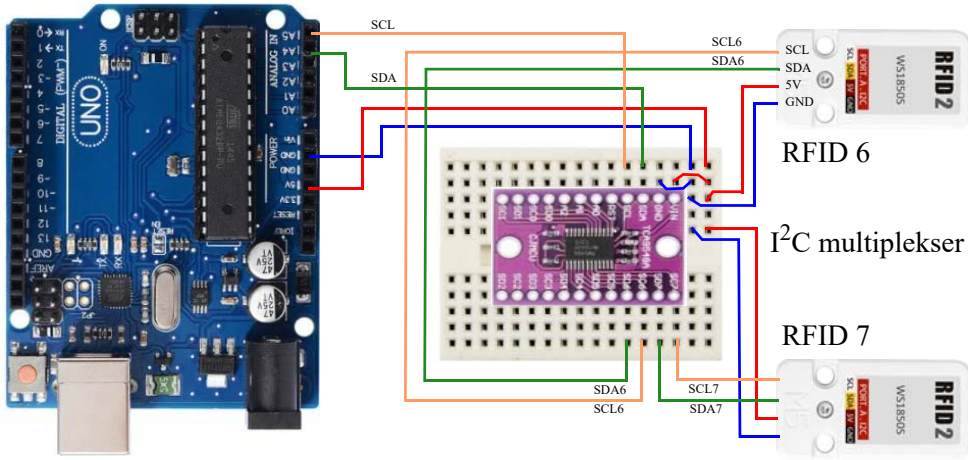
4. Hva om antallet skal økes til 10 eller flere, hvordan vil du da løse oppdraget?

3.4 Deloppdrag 4 – Oppkobling og testing av TCA9548A

I dette deloppdraget skal vi koble opp I^{2}C multiplekseren og fire RFID 2 terminaler for lesing av kort. Vi skal bruke I^{2}C-scanneren og verifisere at multiplekseren fungerer, for deretter å installere et bibliotek som gjør det enkelt å bruke multiplekseren.

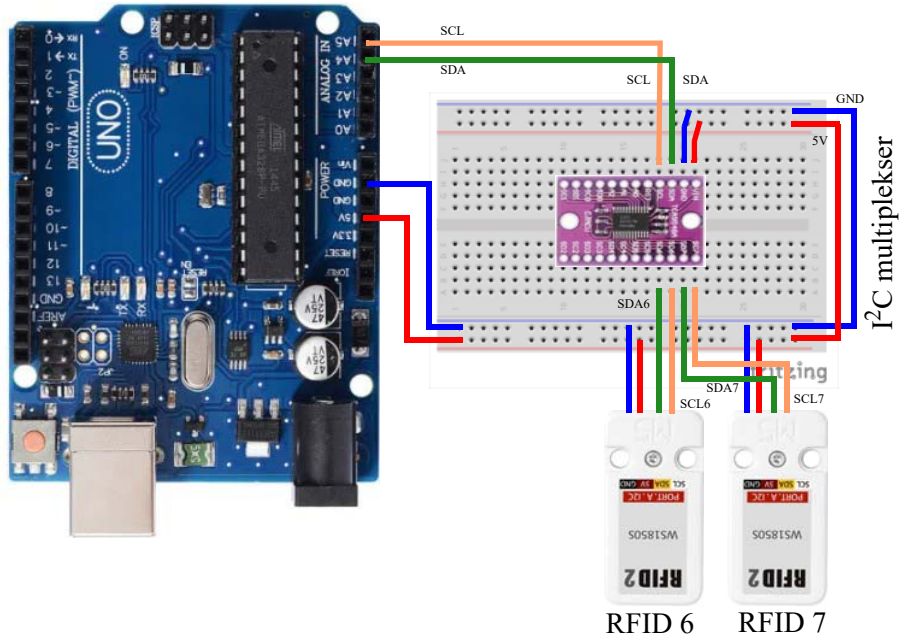
3.4.1 Eksempel på oppkobling av to terminaler

Figuren under viser hvordan vi kan koble opp to RFID på samme I^{2}C-bussen ved hjelp av multiplekseren.



Figuren over viser hvordan vi kan koble opp to RFID-enheter, 6 og 7. På lignende måte kan vi koble opp alle fire. Hver enhet skal ha 5V (Vin), GND (GND), SDA (SDn) og SCL (SCn), akronymene i parentes er betegnelsene vi finner på kretsen TCA9548A hvor n = 0 – 7.

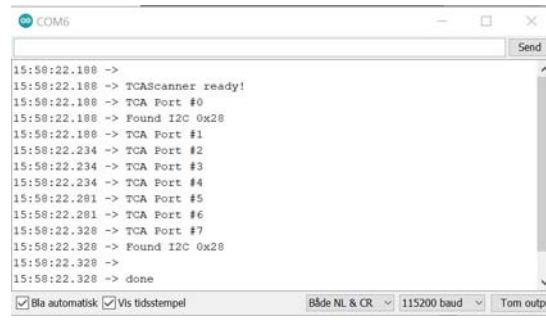
Har vi et større koblingsbrett (half +) kan oppkoblingen bli slik:



Legg merke til at ledningene som følger med settet er gul (SCL) og hvit (SDA), mens vi har benyttet oransje og grønn slik at det skal være lettere å se på papir.

3.4.2 Scanning av oppkoblingen

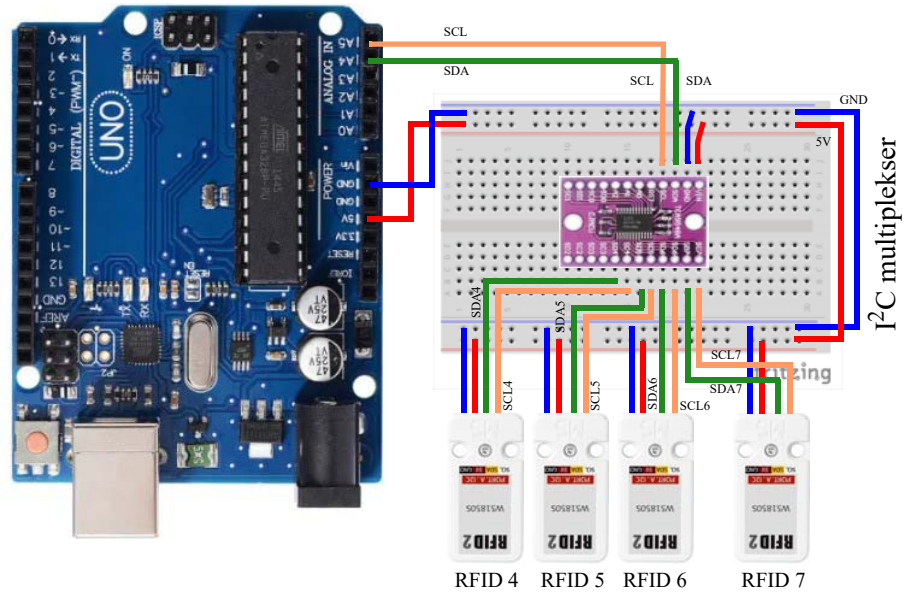
For å se at RFID-enhetene er tilkoblet, kan vi kjøre et scanneprogram (se vedlegg A.1, side 73) som legges over på mikrokontrolleren. Dette programmet vil lete opp alle I^{2}C-adresser som er tilkoblet multiplekseren og hvilken port de er tilkoblet. Når vi lastet opp og kjørte programmet fikk vi følgende resultat, se figuren til høyre.



I dette eksempelet har vi koblet en enhet til port #0 og en til port #7, begge med adresse 0x28.

3.4.3 Eksempel på oppkobling av to terminaler

La oss nå koble opp de to siste terminalene slik at vi i alt har 4 terminaler tilkoblet.



3.5 Deloppdrag 5 – Avlesning av fire enheter via multiplekseren TCA9548A

I dette deloppdraget skal vi lage et program som leser fire RFID 2 enheter via I^2C multiplekseren og skriver ut ID-kodene til alle fire kortene på heksadesimal form. I tillegg skal vi gjøre om den heksadesimale koden til ett heltall slik at det er lettere å sjekke om identitetskortet er akkreditert.

3.5.1 Program for avlesning av fire RFID-terminaler

Vi skal nå se nærmere på hvordan vi kan lese av to RFID 2 eller flere brikker med samme adresse ved å bruke multiplekseren TCA9548A.

Vi velger i første omgang å installere et bibliotek som skal gjøre det lettere å lese av brikkene.

1. Hent og installer biblioteket: https://github.com/WifWaf/TCA9548A på vanlig måte. Biblioteket kan også installeres via *Inkluder bibliotek* –> *Administrer bibliotek* og skriv TCA9548A i søkefeltet. Da kommer følgende alternativ opp:

Vi velger å installere Joathan Dempsey's bibliotek, TCA9548A.

2. **Inkluder nødvendige biblioteker i programmet**
   ved å skrive inn headerfilene:

```
#include "MFRC522_I2C.h"
#include "TCA9548A.h"
#include <Wire.h>
```

`MFRC522_I2C.h` gjelder RFID 2 brikkene, mens `TCA9548A.h` gjelder I^2C-multiplekseren.

3. **Deklarer objekter**
   Her deklarerer vi to objekter, ett for RFID og et for TCA9548A

```
TCA9548A I2CMux;
MFRC522 mfrc522(0x28,3);
```

I^2C-adressen til TCA9548A overføres via biblioteksfunksjonen, så den ser vi ikke mer til, dette kan være en utfordring dersom vi ønsker å bruke flere multipleksere. For å løse dette framtidige problemet kan man se hva andre biblioteker tilbyr, evt. gå inn å gjøre endringer i biblioteket.

Adressen for RFID 2, 0x28, settes i argumentet til objektet sammen med portnummeret til RST (reset) hos RFID. Hos vår RFID-terminal er ikke denne tilgjengelig.

4. **Initialisering i setup()-funksjonen**
   I setup()-funksjonen inkluderer vi initialisering av objektene o.a. Først initialiserer vi serie-kommunikasjonen slik at vi kan skrive til monitoren, og kommunikasjon via I^2C:

```
Serial.begin(9600);
Wire.begin();
```

Deretter initialiserer vi multiplekseren

```
I2CMux.begin(Wire);
```

Tilslutt må vi initialisere hver enkelt av RFID-terminalene. Dette kan vi gjøre i en for-loop. Siden vi for øyeblikket bare har koblet opp fire RFID-terminaler, blar vi oss gjennom de fire, fra 4 til 7:

```
for (int i = 4; i < 8; i++)
  {
    I2CMux.openChannel(i);  // Åpner kanal i (i = 4 - 7)
    delay(20);
    mfrc522.PCD_Init();     // Initaliserer terminalen langs kanal i
    delay(20);
    I2CMux.closeChannel(i); // Lukker kanal i (i = 4 - 7)
  }
```

Lukker så for sikkerhets skyld alle kanaler:

```
I2CMux.closeAll();
```

5. **Gjentatt lesing av RFID-terminaler**

I loopen ønsker vi å lese av hver enkelt ID-kort og skrive ut resultatet til monitoren. Dette gjør vi med følegdne for-loop:

```
for (int i = 4; i < 8; i++)
  {
    I2CMux.openChannel(i);
    readRFID(); // Leser av terminalene
    readRFID(); // Leser av terminalene
    delay(20);
    I2CMux.closeChannel(i);
  }
```

Vi gjennomløper kanalene 4 – 7. Først åpner vi kanalen, så leser vi av kortet for så å lukke kanalen. Det er lagt inn et lite delay før lukking.

Vi har lagt all lesing av RFID 2 i en funksjon som vi skal komme tilbake til. Så hvorfor leser vi to ganger rett etter hverandre?

Under uttestingen av fire terminaler fant vi at den leste terminalene i to omganger. Tilsammen leste disse to omgangene alle terminalene. Vi vet ikke hvorfor det er slik, men problemet synes å kunne løses ved to ganger lesing.

6. **Funksjon for lesing av RFID-terminaler og utskrift av resultatet**

Vi har valgt å legge all lesing av terminalene og utskrift av resultatet i en egendefinert funksjon readRFID();. Vi leser av de fire bytene som inneholder brikkens identitet. Det er i første omgang praktisk å uttrykke disse fire bytene i hexadesimal kode.

```
void readRFID()
  {
  if ( mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial() )
    {
    // Skriv ut UID
    Serial.print(F("Kort UID: "));
    for (byte i = 0; i < mfrc522.uid.size; i++)
      {
      if (mfrc522.uid.uidByte[i] < 0x10) Serial.print("0");
         // Legg til en "0" om verdien er < 0x10
      Serial.print(mfrc522.uid.uidByte[i], HEX);
      Serial.print(" ");
      }
```

```
    Serial.println();
  }
}
```

Funksjonen `mfrc522.PICC_IsNewCardPresent()` sjekker om det er lagt på et kort og funksjonen `mfrc522.PICC_ReadCardSerial()` leser kortet. Om begge disse er vellykket (returnerer de 1) så vet vi:

- ... at variabelen `mfrc522.uid.size` inneholde antall byte med ID-informasjon. I vårt tilfelle er dette fire byte, men det finnes også systemer som benytter 2 eller 7, sistnevnte vil gi langt lavere sannsynlighet for at to kort skal ha lik ID-kode.
- ... at de fire ID-kodene ligger i arrayet `mfrc522.uid.uidByte[i]` slik at vi kan lese dem ut en etter en.
- ... at en byte har 8 bit, hvorav 4 og 4 bit (nibles) kan uttrykkes med et hexsadesimalt siffer (0 – F). En byte kan derfor uttrykkes som to hexadesimale siffer. Dersom byten inneholder et tall mindre enn 0x10 (Hex) så vil første siffer sløyfes med mindre vi setter til en 0, som vi har valgt å gjøre. Dessuten legger vi til et mellomrom mellom hver utskrevet byte. Til slutt skriver vi ut en strek (----------) som gjør at vi kan se hvor en ny utskrift starter.

7. **Utskrift i monitor**
   I monitoren vil vi se følgende utskrift (under til venstre) dersom alle fire ID-kortene er lagt på. Utskriften vil gå fortløpende. Dersom vi fjerner ett av kortene vil vi miste en linje i utskriften (under til høyre):

```
12:14:50.812 -> ----------------
12:14:50.812 -> Kort UID: C6 28 87 F7
12:14:50.912 -> Kort UID: 7C 11 5E 74
12:14:51.011 -> Kort UID: B0 9C 5C 74
12:14:51.058 -> Kort UID: 01 43 D1 C6
12:14:51.112 -> ----------------
12:14:51.159 -> Kort UID: C6 28 87 F7
12:14:51.211 -> Kort UID: 7C 11 5E 74
12:14:51.312 -> Kort UID: B0 9C 5C 74
12:14:51.412 -> Kort UID: 01 43 D1 C6
12:14:51.465 -> ----------------
```

```
12:20:25.282 -> ----------------
12:20:25.383 -> Kort UID: 7C 11 5E 74
12:20:25.483 -> Kort UID: B0 9C 5C 74
12:20:25.519 -> Kort UID: 01 43 D1 C6
12:20:25.614 -> ----------------
12:20:25.698 -> Kort UID: 7C 11 5E 74
12:20:25.782 -> Kort UID: B0 9C 5C 74
12:20:25.862 -> Kort UID: 01 43 D1 C6
12:20:25.899 -> ----------------
```

Vi legger merke til at vi også har valgt å skrive ut klokkeslettet som er en funksjon i monitoren, og vi registrerer at de fire kortene leses og skrives ut i løpet av ca. 250 ms.

3.5.2 Konvertering av fire byte til en integer

Dette har vi gjort tidligere så dette burde være lett.

Bruk funksjonen under til å omforme de fire bytene i `mfrc522.uid.uidByte[i]` til et heltall og skriv tallet ut til monitoren. Gjør det samme for ID-kodene på alle de fire tallene

```
long convertFourByteToLong(byte UID[4])
{
  long val = 0;
  val += UID[0];
  val += UID[1];
```

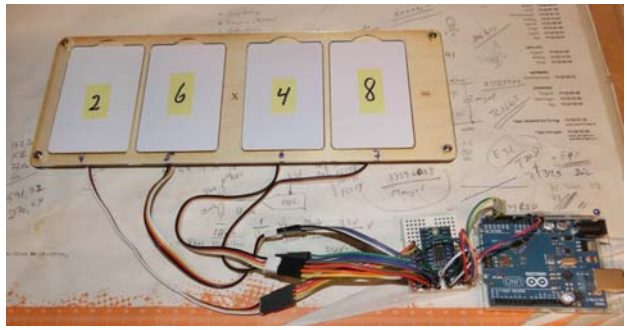
```
    val += UID[2];
    val += UID[3];
    return val;
}
```

Vi overfører arrayet med de fire byte'ne til funksjonen `mfrc522.uid.uidByte[i]` som adderer hver av bytene til en heltallsverdi. Dersom vi bruker denne metoden må vi være oppmerksomme på at vi kan få like ID-koder på to eller flere kort. Slik vi finner den samlede koden til kortet så er det unødvendig å bruke typen `long` på verdien (val), men det vil være nødvendig dersom vi ønsker å beregne den samlede verdien til koden (4 x 8 bit). Se ulike alternativer å gjøre dette på i oppdrag 2 i avsnitt 3.2.3, side 50.

Løsningsforslag finnes i vedlegg B.4, side 83 og i vedlegg B.5, side 84.

3.6 Deloppdrag 6 – Lag en liten multiplikator med fire RFID-kort

For å kunne holde orden på RFID-terminalene har vi laget en liten laserkuttet holder som gjør at de holder seg på plass, se bildet til høyre.



Oppgaven går ut på å knyttet ID-koden til kortene til, i dette tilfellet, tallene 2, 4, 6 og 8, for så å multiplisere tallene knyttet til to og to kort, eksempelvis 26 x 48 som vist på bildet. Resultatet legges ut i monitoren.

Vi ønsker å ta utgangspunkt i besvarelsen på Deloppdrag 5 og ser for oss følgende deloppgaver:

1. Sett tallene 2, 4, 6, og 8 på de fire RFID-kortene
2. Finn RFID-koden til hvert av kortene og knytt dem til tallene.
3. Sjekk at det ligger kort i hver av posisjonene. Det skal ikke beregnes noe produkt før alle kortene er på plass.
4. Beregn og skriv ut produktet av de to tallene
5. Legge tallene inn i regnestykket på rett plass

Vi foreslår at man så langt det er praktisk, lager funksjoner som utfører de ulike operasjonene.

1. Sette tall på RFID-kortene

Vi har skrevet tallene på de fire kortene. Vi har skrevet på tape slik at det skal være enkelt å endre tallene.

2. Knytt RFID-koden til hvert av kortene

Vi velger å lage et todimensjonalt array hvor vi legger kort-koden i den ene dimensjonen og de tilhørende tallene i den andre. Siden vi senere i oppdraget skal bruke 8 kort, så velger vi et array

med 8 x 2 plasser. Inntil videre kjenner vi kort-koden til bare 4 av de 8 kortene, i de resterende kortene legger vi inn dummy-verdier som ikke kommer i konflikt med virkelige kort-koder.

I tillegg deklarerer vi et en dimensjonalt array som holder tallene i de ulike posisjonene i regnestykket (int T[8]). Også her velger vi å ha 8 plasser siden vi vil få flere tall å holde rede på i siste del av oppdraget.

Vi velger å deklarere arrayene som globale slik at vi kan bruke dem i alle funksjonene:

```
int  arrayRFIDTall[8][2] = {{1000, 1}, {425, 2}, {3000, 3}, { 643, 4}, {5000, 5}, { 874, 6},  {7000, 7},  {863, 8}};
int  T[8] = {0, 0, 0, 0, 0, 0, 0, 0};
```

Tallene 2, 4, 6 og 8 har reelle verdier, mens tallene 1, 3, 5 og 7 har inntil videre, fiktive verdier.

3. Sjekk at det ligger kort i alle posisjonene

Vi vet at kort-terminalene returnerer 0 dersom det ikke ligger noe kort der. Forutsetningen for å utføre beregningen er at ingen av kort-terminalene returnerer 0. Vi velger å multiplisere alle verdiene med hverandre. Dersom produktet er 0 så vet vi at i hvertfall ett av kortene mangler og returnerer 0. Alternativt kan vi bruker den logiske eller-funksjonen:

```
if (arrayRFID[4]*arrayRFID[5]*arrayRFID[6]*arrayRFID[7] != 0)
  {
    // Her utføres det som skal gjøres dersom alle 4 kortene er på plass
  }
```

arrayRFID[] holder det avleste resultatet fra RFID-terminalene, se tidligere oppdrag. Siden vi foreløpig bare har fire terminaler og har valgt å bruke terminal 4 – 7, så er det disse vi tester på.

4. Beregn og skriv ut produktet

Det kan virke litt underlig at vi velger å lage programvaren for å beregne og skrive ut produktet før vi har satt tallene på riktig plass. Siden vi opererer med variabler så kan vi gjør nettopp det. Det holder at vi vet hvilke variabler vi skal bruke i regnestykket, vi trenger ikke vite innholdet i variablene. Siden vi bruker et array for å holde tallene, må vi vite hvilken array-indeks vi plasserer i de ulike posisjonene i regnestykket. Vi har også valgt å lage en egen funksjon som utfører og skriver ut beregningen. Med tanke på at dette er ett av to produkter så har vi valgt, med tanke på neste oppgave, å kalle det *produkt2*.

```
int produkt2()
{
  // Regnestykket er AB * CD = Produkt eller (10*A+B)*(10*C+D) = Produkt eller (10*T(4)+T(5))*(10*T(6)+T(7))=Produkt
  int Prod2 = (10 * T[4] + T[5]) * (10 * T[6] + T[7]);
  Serial.print("Produkt 2: ");
  Serial.println(Prod2);
  return Prod2;
}
```

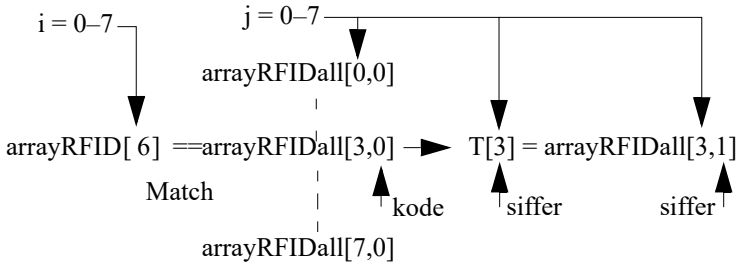
Legg også merke til at produktet returneres fra funksjonen, hvilket betyr at når funksjonen kalles så får vi produktet i retur, i dette eksempelet `Prod2`.

5. Legge tallene på rett plass i regnestykket

Nå når vi vet hvilke variabler vi har valgt å ha med i regnestykket så må vi sørge for at tallene ligger akkurat der. Dette gjør vi med følgende *omstokkingsfunksjon*:

```
void omstokking()
{
  for (int i = 0; i < 8; i++)
  {
    for (int j = 0; j < 8; j++)
    {
      if ((arrayRFID[i] == arrayRFIDTall[j][0]) && (arrayRFID[i] != 0)) T[i] = arrayRFIDTall[j][1];
    }
  }
}
```

RFID-koden leses først inn i arrayRFID[], dermed vet vi hvor kortene blir plassert i regnestykket. Dernest må vi konvertere RFID-kode til tallverdi. Dette gjør vi ved hjelp av en dobbel rekursjon, hvor vi sammenligner `arrayRFID[i]` og `arrayRFIDTall[j][0]`, dersom vi får match kan vi legge det tilhørende tallet inn på riktig plass i tallrekka `T[i] = arrayRFIDTall[j][1]`, slik at produktet blir riktig utregnet. I tillegg må vi passe på at 0-verdier blir holdt utenfor. Så lenge vi har færre kort enn det maksimale antallet, vil vi få flere 0-verdier. Siden vi har lagt inn dummy-verdier i arrayet `arrayRFIDTall[ ][ ]` så er dette strengt tatt ikke nødvendig.



Figuren over viser hvordan omstokkingen foregår. *i* gjennomløper verdiene 0 – 7 i den ytre sløyfa, mens *j* gjennomløper 0 – 7 i den indre sløyfa. Når kortkoden "matcher" legges sifferverdien inn i arrayet av tallverdier, T[ ] på aktuell posisjon. I eksempelet er $i = 6$ og $j = 3$. Der finner vi match mht. koden og legger det korresponderende sifferet inn i T[3].

Tilslutt må vi sørge for å kalle funksjonene fra loop()-funksjonen slik at programmet fungerer som ønsket.

6. Skriv og test programmet

Last opp og kompiler og sjekk at det fungerer som forventet.

Flytt rundt på kortene og sjekk at følgende fungerer som ønsket:

- Forsvinner produktet fra monitoren når ett eller flere kort mangler?
- Regner programmet riktig?
- Oppfattes programmet å ha umiddelbar respons?

Løsningsforslag finnes i vedlegg B.6, side 87.

3.7 Deloppdrag 7 – Lag et matematisk spill – "Nærmest null"

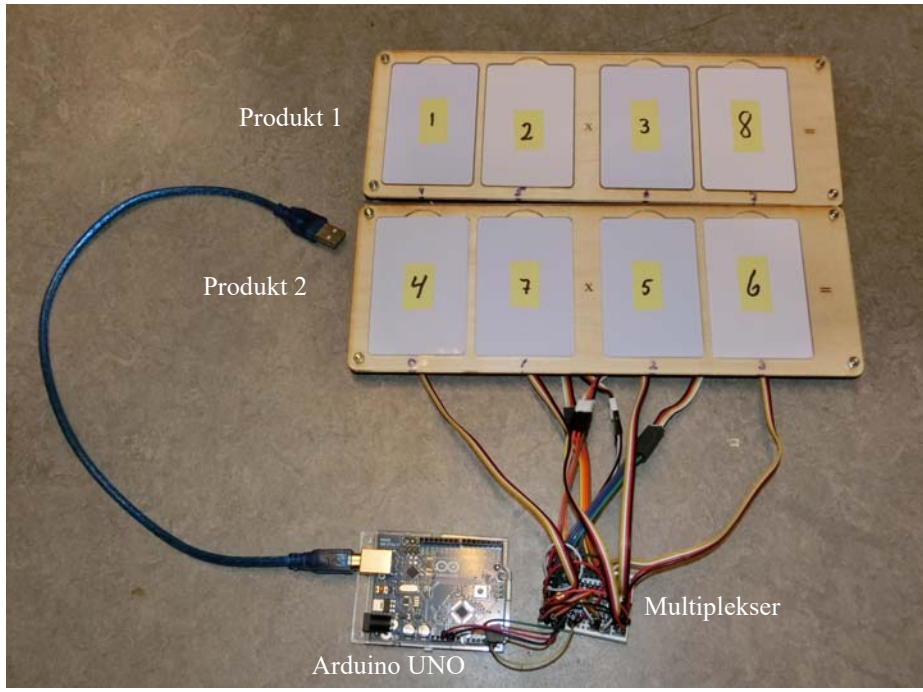
Gå sammen to og to evt. tre og sett sammen to sett hver med fire RFID-kort. Monter alle 8 RFID-terminalene i en I^2C-multiplekser, og still brettene over hverandre. Beregn to produkter og finn differansen mellom dem. Oppgaven går ut på at kortene plasseres slik at differansen mellom produktene kommer nærmest null.

Vi ønsker å ta utgangspunkt i besvarelsen på Deloppdrag 6 og ser for oss følgende deloppgaver:

1. Plasser brettene over hverandre og monter alle RFID-terminalene i en I^2C-multiplekser
2. Modifiser programmet fra Deloppdrag 6 slik at det leser alle 8 RFID-terminalene og legger dem på rett plass i arrayRFID[ ].
3. Modifiser programmet slik at det beregner begge produktene med korrekte tall og beregner differansen
4. Skriv differansen ut i monitoren

3.7.1 Montering og oppkobling

Plasser de to brettene over hverandre slik at *produkt 1* plasseres øverst og *produkt 2* under. Samtlige ledninger føres fram til multiplekseren slik at *produkt1* beregnes på bakgrunn av sifrene 0 til 3 og *produkt2* beregnes på bakgrunn av sifrene 3 til 7. Figuren under viser hvordan det kan ta seg ut.



Gå til Oppdrag 5 og avsnitt 3.4.3, side 55 for å få hjelp til oppkoblingen.

3.7.2 Les alle åtte kortene

Modifiser programmet slik at alle de 8 kortene kan leses. Siden mye av programmet er forberedt for å lese åtte kort så gjelder det bare å endre grensene til for()-loopen slik at tellevariabelen går fra 0 – 7.

3.7.3 Beregn produkt 1 og 2

Kopier funksjonen som beregner produktet slik at dere får to produktfunksjoner produkt 1 og 2. La hver av funksjonene returnere produktet. Pass på å bruke de riktige sifrene i de to produktene.

3.7.4 Beregn differansen

Bruk resultatene fra de to produktene og beregn differansen. Skriv differansen til monitoren. Pass på å endre programmet slik at differansen kun skrives ut når alle kortene er på plass. Så lenge et mangler beregnes ingen produkter eller differansen.

Løsningsforslaget er vedlagt i vedlegg B.7, side 90.

3.7.5 Løsning av oppgaven

Her har vi sifrene 1 – 8 som kan danne to produkter av typen AB · CD – EF · GH = Diff. Her kan man stille en rekke interessante spørsmål:

- Hvor mange løsninger finnes som gir 0?
- Hvor mange løsninger finnes som gir 100, 1000, 10000 o.l.?
- Finnes det mønster mht. hvor mange løsninger det er for hvert tall i tallrekken. Er et noen differanser som forekommer hyppigere enn andre?

3.8 Deloppdrag 8 – Diskuter ulike anvendelser

Diskuter i grupper og kom opp med forslag til hvordan denne teknologien kan brukes på ulike måter i en utstilling. Vurder de ulike forslagene og forbered presentasjon av det beste forslaget. Evt. se også avsnitt 1.2, side 15 og kapittel 4, side 65 hvor vi har realisert en generell prototyp for uttesting av ulike varianter av pusleoppgaver med tall.

I Vedlegg C, side 95 er vedlagt simuleringsprogram for beregning av løsninger på noen matematiske pusleoppgaver knyttet til den generelle prototypen.

4 Eksperimentell utstilling – Varianter av "Nærmest null"

I avsnitt 1.2.1, side 15 har vi omtalt følgende varianter av "Nærmest null":

4.1 Varianter av "Nærmest null":

Variant 1:

Alle sifrene fra 0 – 9 er med og kan flyttes. Tre regnetegn er plassert fast. Oppgaven går ut på flytte tallene slik at resultatet blir så nær null som mulig. Oppgaven har 198 mulige løsninger (C.2.1, side 96).

*abc*de–fgh*ij = Resultat (a – j = 0 – 9)*

Denne modellen kan varieres i det uendelige. Her er et par alternative varianter.

Variant 2:

Her brukes sifrene 1 til 9:

abc + def – ghi = Resultat (a – i = 1 – 9)

Denne varianten er en kjent problemstilling som har i alt 336 løsninger (C.3.1, side 98). Den er da gjerne satt opp på denne måten abc + def = ghi. Denne varianten krever imidlertid at publikum selv regner etter om regnestykket går opp. Det er imidlertid vår hypotese at modellen blir mer attraktiv dersom vi fortløpende regner ut resultatet.

Variant 3:

Denne varianten har 310 løsninger og kan synes relativt enkel, men når du bruker den så er den mer krevende enn du skulle tro.

*a*bc + d*ef – g*hi = Resultat (a – i = 1 – 9)*

Den kan imidlertid være noe krevende å finne løsninger til.

Variant 4:

Man kan også gjøre regnestykket mer sammensatt:

*ab*c + de + fg – hi*j = Resultat (a – j = 0 – 9)*

Om sistnevnte variant appellerer til publikum er noe usikkert.

4.2 Simuleringer

Før man begynner å bygge kan det være lurt å foreta simuleringer med f.eks. Python. for å finne ut om oppgaven har en løsning ev. hvor mange løsninger den har.

Dette er oppgaver som studerer permutasjoner av tallene 1 – 9 evt. 0 – 9. Her kan Python gi oss betydelig hjelp. Programmet som er benyttet er vedlagt i vedlegg C.1, side 95.

4.3 Realisering

For å teste ut ulike varianter som antydet foran kan man lage en generell prototyp der man kan legge på ulike "overlay" eller sjablonger. Ved hjelp av RFID kan programmet detektere hvilket sjablong som er lagt på kan man teste ut det regnestykket som sjablongen indikerer.

Den viste sjablongen realiserer denne utregningen:

*a*bc + d*ef – g*hi = Resultat (a – i = 1 – 9)*

Ulike sjablonger kan så legges på, hver med sin unike RFID-brikke som programmet bruker til å velge riktig utregning. Her ser vi et par eksempler til:

Denne sjablongen realiserer denne utregningen:

abc + def – ghi = Resultat (a – i = 1 – 9)

:

Denne sjablongen realiserer denne utregningen:

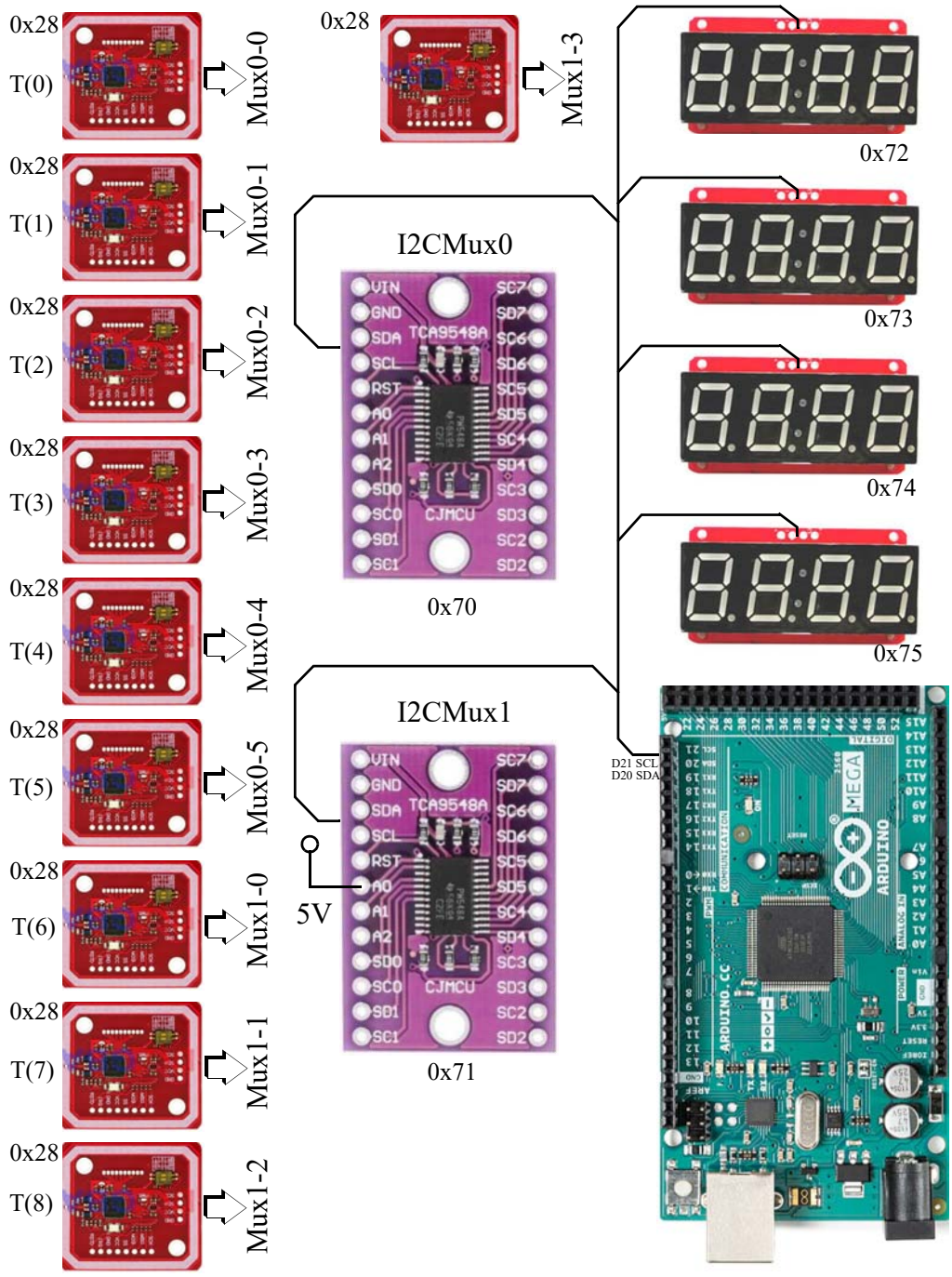
*a*b*c + d*e*f – g*h*i = Resultat (a – i = 1 – 9)*

I det neste avsnittet skal vi se hvordan en slik prototyp kan bygges opp elektronisk.

4.3.1 Koblingsskjema

I denne prototypen har vi benyttet RFID-brikken PN532 som både kommuniserer med I^2C og SPI-buss m.fl. Ved hjelp av en liten bryter velger man kommunikasjonsform. Vi har valgt å benytte I^2C-buss for denne prototypen.

Figuren under viser et blokkdiagram over oppkoblingen.


0x28
T(0)
Mux0-0
0x28
T(1)
Mux0-1
0x28
T(2)
Mux0-2
0x28
T(3)
Mux0-3
0x28
T(4)
Mux0-4
0x28
T(5)
Mux0-5
0x28
T(6)
Mux1-0
0x28
T(7)
Mux1-1
0x28
T(8)
Mux1-2
0x28
Mux1-3
I2CMux0
VIN
GND
SDA
SCL
RST
A0
A1
A2
SDO
SCO
SD1
SC1
TCA9548A
CJMCU
SC7
SD7
SC6
SD6
SC5
SD5
SC4
SD4
SC3
SD3
SC2
SD2
0x70
I2CMux1
5V
0x71
0x72
0x73
0x74
0x75
D21 SCL
D20 SDA
MEGA
ARDUINO
ARDUINO.CC

Samtlige komponenter kommuniserer med mikrokontrolleren via I^2C-bussen. Siden displayene (adresse 0x72, 0x73, 0x74 og 0x75) og multiplekserne (adresse 0x70 og 0x71) kan programmers til ulike adresser på I^2C-bussen, kan disse kobles direkte på bussen til mikrokontrolleren (D20 – SDA og D21 – SCL) tegnet med heltrukken strek på skjemaet. RFID-brikkene har samme I^2C-adresse (0x28) og adressen kan ikke endres, så må disse kobles på multiplekserne. Disse er markert med pil og en kode som antyder hvilken multiplekserport de skal kobles til (Mux0-0 til Mux0-5 og Mux1-0 til Mux1-3). I tillegg skal alle komponentene kobles til jord (GND) og 5V på mikrokontrollerkortet.

4.4 Erfaringer med denne prototypen og PN532

Her er noen erfaringer høstet i arbeidet med denne prototypen:

1. **Bruk av SPI-bussen**
   Koblet først opp ved bruk av SPI-bussen og brukte CS (Chip Select) linjen til å gi den RFID-terminalen tilgang til bussen. Dette syntes å fungere dårlig da det var noe tilfeldig om jeg oppnådde kontakt med RFID-kortet eller ikke. Dessuten krevde bussen en del flere forbindelser enn I^2C-bussen. Gikk derfor over til bruk av I^2C-bussen.

2. **Bruk av I^2C-buss**
   Her kreves bussen kun to linjer i tillegg til spenningsforsyning og GND. Imidlertid erfarte jeg noe av det samme at jeg fikk time out på noen av terminalene før den hadde fått lest av RFID-kortet. Dessuten måtte jeg benytte I^2C-multiplekser for å kunne lese av mer enn en terminal siden alle terminalene har samme I^2C-adresse (0x28).

3. **Smitting mellom RFID-terminalene**
   Siden jeg ønsket å lese av inntil 10 RFID-terminaler benyttet jeg to I^2C-multipleksere av typen TCA9548A kretser. Dessuten oppdaget jeg at når jeg la ned en RFID-tag i en av posisjonene, leste også naboterminalen enkelte ganger av den samme tag'en. Dette løste jeg greit ved å heve tag'en ca. 3 mm opp fra plata, dermed ble også avlesningen bedre, men ikke helt bra.

4. **Ujevne sjablonger i kryssfiner**
   Laget sjablonger av kryssfiner for å legge på toppen av basisplata som holdt RFID-terminalkortene. Kryssfiner er ikke ideelt siden den har en tendens til å slå seg noe. Dermed vil ikke platene ligge tett sammen. Dette er primært et estetisk problem, som løses ved bruk av fiberplata, som f.eks. MDF.

5. **Bruk av stiftlister og hunn-hunn-jumpere**
   Bruk av stiftlister og hunn-hunn-jumpere er en enkel løsning som er lett å koble opp, men som gir betydelig kaos nav ledninger på baksiden. Koblingene mellom terminalkortene var ganske korte. Laget derfor horte hunn-hunn-kontakter noe som reduserte ledningskaoset.

6. **I^2C-terminaler**
   Ved hjelp av et Vero bord med fire striper ble det laget en rekke parallelle I^2C-bussterminaler

7. **Strømforbruk**
   I tillegg til 10 RFID-terminaler og fire 4x7-segment LED-display begynte elektronikken å trekke mye strøm for en Arduino MEGA 2650. Dette ble løst ved å redusere lysstyrken på displayene. Alternativt burde går greit å lage et eget 5V supply for displayene med felles jord.

5 Referanser

[1] TCA4598A I^2C multiplekser
https://learn.adafruit.com/adafruit-tca9548a-1-to-8-i2c-multiplexer-breakout

[2] RFID 2 WS1850S
https://www.elfadistrelec.no/Web/Downloads/_t/ds/U031-B_eng_tds.pdf

Vedlegg A Hjelpeprogrammer

A.1 Scanneprogram[21]

Programmet scanner multiplekseren og sjekker adressen til de tilkoblete enhetene.

```
/*
 * TCA9548 I2CScanner.ino -- I2C bus scanner for Arduino
 * Based on https://playground.arduino.cc/Main/I2cScanner/
 */

#include "Wire.h"

#define TCAADDR 0x70

void tcaselect(uint8_t i) {
  if (i > 7) return;

  Wire.beginTransmission(TCAADDR);
  Wire.write(1 << i);
  Wire.endTransmission();
}

// standard Arduino setup()
void setup()
{
    while (!Serial);
    delay(1000);

    Wire.begin();

    Serial.begin(115200);
    Serial.println("\nTCAScanner ready!");

    for (uint8_t t=0; t<8; t++) {
      tcaselect(t);
      Serial.print("TCA Port #"); Serial.println(t);
```

21. Programmet er hentet fra: https://learn.adafruit.com/adafruit-tca9548a-1-to-8-i2c-multiplexer-breakout/arduino-wiring-and-test

```
    for (uint8_t addr = 0; addr<=127; addr++) {
      if (addr == TCAADDR) continue;

      Wire.beginTransmission(addr);
      if (!Wire.endTransmission()) {
        Serial.print("Found I2C 0x");  Serial.println(addr,HEX);
      }
     }
   }
   Serial.println("\ndone");
}

void loop()
{
}
```

A.2 Kode for dumping av RFID-innhold ved oppkoblet med SPI-buss

```
/*
---------------------------------------------------------------------------------------------------------------------
Example sketch/program showing how to read data from a PICC to serial.
---------------------------------------------------------------------------------------------------------------------
This is a MFRC522 library example; for further details and other examples see: https://
github.com/miguelbalboa/rfid. Example sketch/program showing how to read data from a PICC
(that is: a RFID Tag or Card) using a MFRC522 based RFID. Reader on the Arduino SPI interface.
When the Arduino and the MFRC522 module are connected (see the pin layout below), load this
sketch into Arduino IDE then verify/compile and upload it. To see the output: use Tools, Serial
Monitor of the IDE (hit Ctrl+Shft+M). When you present a PICC (that is: a RFID Tag or Card) at
reading distance of the MFRC522 Reader/PCD, the serial output will show the ID/UID, type and
any data blocks it can read. Note: you may see "Timeout in communication" messages when
removing the PICC from reading distance too early.
If your reader supports it, this sketch/program will read all the PICCs presented (that is: multiple
tag reading). So if you stack two or more PICCs on top of each other and present them to the
reader, it will first output all details of the first and then the next PICC. Note that this may take
some time as all data blocks are dumped, so keep the PICCs at reading distance until complete.
@license Released into the public domain.
Typical pin layout used:
-----------------------------------------------------------------------------------------------------------------------
                MFRC522      Arduino   Arduino   Arduino     Arduino      Arduino
```

```
                Reader/PCD   Uno/101     Mega      Nano v3   Leonardo/Micro   Pro Micro
Signal          Pin          Pin         Pin       Pin       Pin              Pin
-----------------------------------------------------------------------------------------------------------------------
RST/Reset       RST          9           5         D9        RESET/ICSP-5     RST
SPI SS          SDA(SS)      10          53        D10       10               10
SPI MOSI        MOSI         11 / ICSP-4 51        D11       ICSP-4           16
SPI MISO        MISO         12 / ICSP-1 50        D12       ICSP-1           14
SPI SCK         SCK          13 / ICSP-3 52        D13       ICSP-3           15
 */

#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN         9          // Configurable, see typical pin layout above
#define SS_PIN          10         // Configurable, see typical pin layout above

MFRC522 mfrc522(SS_PIN, RST_PIN);  // Create MFRC522 instance

void setup() {
  Serial.begin(9600);  // Initialize serial communications with the PC
  while (!Serial);        // Do nothing if no serial port is opened (added for Arduinos based on ATMEGA32U4)
  SPI.begin();            // Init SPI bus
  mfrc522.PCD_Init();// Init MFRC522
  mfrc522.PCD_DumpVersionToSerial();  // Show details of PCD - MFRC522 Card Reader details
  Serial.println(F("Scan PICC to see UID, SAK, type, and data blocks..."));
}

void loop() {
  // Look for new cards
  if ( ! mfrc522.PICC_IsNewCardPresent()) {
    return;
  }
  // Select one of the cards
  if ( ! mfrc522.PICC_ReadCardSerial()) {
    return;
  }
  // Dump debug info about the card; PICC_HaltA() is automatically called
  mfrc522.PICC_DumpToSerial(&(mfrc522.uid));
```

}

A.3 Kode for endring av UID ved oppkoblet med SPI-buss

```
/*
---------------------------------------------------------------------------------------------------------------------
Example to change UID of changeable MIFARE card.
---------------------------------------------------------------------------------------------------------------------
This is a MFRC522 library example; for further details and other examples see: https://git-hub.com/miguelbalboa/rfid.
This sample shows how to set the UID on a UID changeable MIFARE card.
NOTE: for more informations read the README.rst
@author Tom Clement
@license Released into the public domain.
Typical pin layout used:
---------------------------------------------------------------------------------------------------------------------
             MFRC522      Arduino     Arduino   Arduino     Arduino          Arduino
             Reader/PCD   Uno/101     Mega      Nano v3     Leonardo/Micro   Pro Micro
Signal       Pin          Pin         Pin       Pin         Pin              Pin
---------------------------------------------------------------------------------------------------------------------
RST/Reset    RST          9           5         D9          RESET/ICSP-5     RST
SPI SS       SDA(SS)      10          53        D10         10               10
SPI MOSI     MOSI         11 / ICSP-4 51        D11         ICSP-4           16
SPI MISO     MISO         12 / ICSP-1 50        D12         ICSP-1           14
SPI SCK      SCK          13 / ICSP-3 52        D13         ICSP-3           15
 */

#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN   9     // Configurable, see typical pin layout above
#define SS_PIN    10    // Configurable, see typical pin layout above

MFRC522 mfrc522(SS_PIN, RST_PIN);   // Create MFRC522 instance

/* Set your new UID here! */
#define NEW_UID {0xDE, 0xAD, 0xBE, 0xEF}
```

```
MFRC522::MIFARE_Key key;

void setup() {
  Serial.begin(9600);  // Initialize serial communications with the PC
  while (!Serial);     // Do nothing if no serial port is opened (added for Arduinos based on ATMEGA32U4)
  SPI.begin();         // Init SPI bus
  mfrc522.PCD_Init();  // Init MFRC522 card
  Serial.println(F("Warning: this example overwrites the UID of your UID changeable card, use with care!"));

  // Prepare key - all keys are set to FFFFFFFFFFFFh at chip delivery from the factory.
  for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
  }
}

// Setting the UID can be as simple as this:
//void loop() {
//  byte newUid[] = NEW_UID;
//  if ( mfrc522.MIFARE_SetUid(newUid, (byte)4, true) ) {
//    Serial.println("Wrote new UID to card.");
//  }
//  delay(1000);
//}

// But of course this is a more proper approach
void loop() {

  // Look for new cards, and select one if present
  if ( ! mfrc522.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadCardSerial() ) {
    delay(50);
    return;
  }

  // Now a card is selected. The UID and SAK is in mfrc522.uid.

  // Dump UID
  Serial.print(F("Card UID:"));
```

```
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
  }
  Serial.println();

  // Dump PICC type
//  MFRC522::PICC_Type piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
//  Serial.print(F("PICC type: "));
//  Serial.print(mfrc522.PICC_GetTypeName(piccType));
//  Serial.print(F(" (SAK "));
//  Serial.print(mfrc522.uid.sak);
//  Serial.print(")\r\n");
//  if (  piccType != MFRC522::PICC_TYPE_MIFARE_MINI
//    &&  piccType != MFRC522::PICC_TYPE_MIFARE_1K
//    &&  piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
//    Serial.println(F("This sample only works with MIFARE Classic cards."));
//    return;
//  }

  // Set new UID
  byte newUid[] = NEW_UID;
  if ( mfrc522.MIFARE_SetUid(newUid, (byte)4, true) ) {
    Serial.println(F("Wrote new UID to card."));
  }

  // Halt PICC and re-select it so DumpToSerial doesn't get confused
  mfrc522.PICC_HaltA();
  if ( ! mfrc522.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadCardSerial() ) {
    return;
  }

  // Dump the new memory contents
  Serial.println(F("New UID and contents:"));
  mfrc522.PICC_DumpToSerial(&(mfrc522.uid));

  delay(2000);
}
```

Vedlegg B Løsningsforslag

Her er programmer for uttesting av RFID 2 og multiplekseren samlet. Programmene er samtidig løsningsforslag til oppdragene

B.1 Deloppdrag 2A – Testprogram for enkel RFID koblet til en Arduino UNO

Programmet leser av et ID-kort og skriver ut ID-koden i HEX.

```
/*
 * RFID-MRC522-2A
 * Programmet leser av ID-koden til et kort og skriver den ut som fire HEX tall
 * Nils Kr. Rossing 09.03.23
 */

#include <Wire.h>
#include "MFRC522_I2C.h"

#define RST 3

MFRC522 mfrc522(0x28, RST);   // Create MFRC522 instance.

void setup() {
  Serial.begin(9600);               // Initialiserer serie kommunikasjon med monitoren PC
  Wire.begin();                     // Initialiserer I2C
  mfrc522.PCD_Init();               // Initialiserer MFRC522
  Serial.println("RFID-MRC522-3");
  Serial.println(F("Scan PICC og les av UID"));
}

void loop() {
  // Se etter et nytt RFID-kort og les av om det er i nærheten
  if ( mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial() )
  {
    // Skriv ut UID
    Serial.print(F("Kort UID: "));
    for (byte i = 0; i < mfrc522.uid.size; i++)
    {
      if (mfrc522.uid.uidByte[i] < 0x10) // Legg til en "0" om verden er < 0x10
      Serial.print("0");
      Serial.print(mfrc522.uid.uidByte[i], HEX);
```

```
      Serial.print(" ");
    }
    Serial.println();
    mfrc522.PICC_HaltA();
  }
}
```

B.2 Deloppdrag 2B – Testprogram for enkel RFID koblet til en Arduino UNO

Programmet leser av et ID-kort og skriver ut ID-koden ut som et desimaltall.

```
/*
 * RFID-MRC522-2B
 * Programmet leser av ID-koden til et kort og skriver den ut som fire HEX tall
 * Programmet regner HEX-koden om til et heltall av typen long slik at det skal være lettere å sammenligne
 * Nils Kr. Rossing 09.03.23
 */

#include <Wire.h>
#include "MFRC522_I2C.h"

#define RST 3

MFRC522 mfrc522(0x28, RST);   // Create MFRC522 instance.

void setup() {
  Serial.begin(9600);
  Wire.begin();
  mfrc522.PCD_Init();
  Serial.println("Leser ID");
}

void loop() {
  // Se etter et nytt ID-kort, les kortet
  if ( ! mfrc522.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadCardSerial() ) {
      delay(50);
      return;
  }
  // UID og SAK er lagret i mfrc522.uid.
  Serial.print(" Kortets UID - ");
```

```
  Serial.print(mfrc522.uid.size);
  Serial.print(" Byte: ");
  Serial.println(convertFourByteToLong(mfrc522.uid.uidByte));
}

long convertFourByteToLong(byte UID[4])
{
  long val = 0;
  val += UID[0];
  val += UID[1];
  val += UID[2];
  val += UID[3];
  return val;
}
```

B.3 Deloppdrag 3 – En grønn lysdiode tennes når ID-kortet er akkreditert

En grønn lysdiode skal tennes når kortet som vises har en ID-kode som er akkreditert.

```
/*
   RFID-MRC522-3
   Programmet leser av ID-koden til et kort.
   Programmet regner HEX-koden om til et heltall av typen
   long slik at det skal være lettere å sammenligne med en akkreditert kode
   En lysdiode tennes når et akkreditert kort registreres
   Nils Kr. Rossing 14.03.23
*/

#include <Wire.h>
#include "MFRC522_I2C.h"

MFRC522 mfrc522(0x28, 3);   // Create MFRC522 instance.

long ID_kode = 0;
long ID_akk = 23668;
int pinLED = 12;

void setup() {
  Serial.begin(9600);
  Wire.begin();
```

```
  mfrc522.PCD_Init();

  pinMode(pinLED, OUTPUT);

  Serial.println("RFID-MRC522-4");
}

void loop() {
  // Se etter et nytt ID-kort, les kortet
  if ( mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial() )
  {

    // UID er lagret i mfrc522.uid. Konverter til long.
    ID_kode = convertFourByteToLong(mfrc522.uid.uidByte);

    // Skriv ut ID-kode
    Serial.print(" Kortets UID - ");
    Serial.println(ID_kode);

    // Sjekk om ID-kort er akkreditert
    if (ID_kode == ID_akk)
    {
      digitalWrite(pinLED, HIGH);
      delay(5000);
      digitalWrite(pinLED, LOW);
      ID_kode = 0;
    }

    // Samme ID-kode vises kun en gang for hver berøring
    mfrc522.PICC_HaltA();
  }
}

long convertFourByteToLong(byte UID[4])
{
  long val = 0;
  val += UID[0];
  val += UID[1];
```

```
  val += UID[2];
  val += UID[3];
  return val;
}
```

B.4 Deloppdrag 5A – Testprogram for lesing av 4 RFID koblet til TCA9548A

Programmer leser av fire ID-kort via I^2C-multiplekseren og skriver det ut i heksadesimalt format

```
/*
   RFID-TCA9548A-5A
   Programmet leser av ID-koden til flere kort koblet til en multiplekser
   og skriver ID-kodene ut som fire HEX tall
   Nils Kr. Rossing 14.03.23
*/
#include "MFRC522_I2C.h"
#include "TCA9548A.h"
#include <Wire.h>

TCA9548A I2CMux;            // Adressen 0x70 overføres via biblioteket
MFRC522 mfrc522(0x28, 3);   // Deklarer et MFRC522 objekt, 0x28 er I2C adressen, 3 angir pinne nr. som
styrer Reset som ikke brukes her.

void setup() {
  Serial.begin(9600);
  Wire.begin();

  I2CMux.begin(Wire);           // Wire instance is passed to the library

  for (int i = 4; i < 8; i++)
  {
    I2CMux.openChannel(i);
    delay(20);
    mfrc522.PCD_Init();
    delay(20);
    I2CMux.closeChannel(i);
  }

  I2CMux.closeAll();            // Set a base state which we know (also the default state on power on)

  Serial.println("RFID-TCA9548A-1");
```

```
}

void loop()
{
  for (int i = 4; i < 8; i++)
  {
    I2CMux.openChannel(i);
    readRFID();
    readRFID();
    delay(20);
    I2CMux.closeChannel(i);
  }
  Serial.println("----------------");
}

void readRFID()
{
  if ( mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial() )
  {
    // Skriv ut UID
    Serial.print(F("Kort UID: "));
    for (byte i = 0; i < mfrc522.uid.size; i++)
    {
      if (mfrc522.uid.uidByte[i] < 0x10) Serial.print("0"); // Legg til en "0" om verden er < 0x10
      Serial.print(mfrc522.uid.uidByte[i], HEX);
      Serial.print(" ");
    }
    Serial.println();
  }
}
```

B.5 Deloppdrag 5B – Testprogram for lesing av 4 RFID koblet til TCA9548A

Programmer leser av fire ID-kort via I^2C-multiplekseren og skriver det ut i helltallsformat

```
/*
  RFID-TCA9548A-5B
  Programmet leser av ID-koden til flere kort koblet til en multiplekser
  og skriver ID-kodene ut som fire HEX tall
  Nils Kr. Rossing 12.03.23
```

```
*/
#include <Arduino.h>
#include "MFRC522_I2C.h"
#include "TCA9548A.h"

#define RST 3
long arrayRFID[8];
long val;

TCA9548A I2CMux;              // Address can be passed into the constructor
MFRC522 mfrc522(0x28, RST);   // Deklarer et MFRC522 objekt.

void setup() {
  Serial.begin(9600);

  I2CMux.begin(Wire);             // Wire instance is passed to the library

  for (int i = 4; i < 8; i++)
  {
    I2CMux.openChannel(i);
    delay(20);
    mfrc522.PCD_Init();
    delay(20);
    I2CMux.closeChannel(i);
  }

  I2CMux.closeAll();              // Lokk alle kanalene før start

  Serial.println("RFID-TCA9548A-4");
}

void loop()
{
  for (int i = 4; i < 8; i++)
  {
    arrayRFID[i] = 0;
    I2CMux.openChannel(i);
    readRFID(i);
```

```
    readRFID(i);
    I2CMux.closeChannel(i);
  }
  Serial.println("------------------");
  printArrayRFID();
  delay(100);
}

void readRFID(int i)
{
  if ( mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial() )
  {
    // Skriv ut UID
    val = convertFourByteToLong(mfrc522.uid.uidByte);
    arrayRFID[i] = val;
  }
}

long convertFourByteToLong(byte UID[4])
{
  long val = 0;
  val += UID[0];
  val += UID[1];
  val += UID[2];
  val += UID[3];
  return val;
}

void printArrayRFID()
{
  for (int i = 0; i < 8; i++)
  {
    Serial.print("RFID nr: ");
    Serial.print(i);
    Serial.print(": ");
    Serial.println(arrayRFID[i]);
  }
  Serial.println();
```

}

B.6 Deloppdrag 6 – Testprogram for lesing av fire tall og beregning av produktet

Programmet leser fire kort som er merket med tallene 2, 4, 6 og 8 og bergener produktet av disse når de legges i rammen for produktet.

```
/*
  RFID-TCA9548A-6
  Programmet leser av ID-koden til flere kort koblet til en multiplekser
  og sjekker plasseringen av kortene og knytter ett tall til ID-kortets kode
  Nils Kr. Rossing 16.03.23
*/
#include <Arduino.h>
#include "MFRC522_I2C.h"
#include "TCA9548A.h"

#define RST 3
long arrayRFID[8];
int  arrayRFIDTall[8][2] = {{1000, 1}, {23668, 2}, {3000, 3}, { -30729, 4}, {5000, 5}, { -11834, 6},  {7000, 7}, {24180, 8}};
int  T[8] = {0, 0, 0, 0, 0, 0, 0, 0};
long val;

TCA9548A I2CMux;            // Address can be passed into the constructor
MFRC522 mfrc522(0x28, RST);   // Deklarer et MFRC522 objekt.

void setup() {
 Serial.begin(9600);

 I2CMux.begin(Wire);          // Wire instance is passed to the library

 for (int i = 4; i < 8; i++)
 {
  I2CMux.openChannel(i);
  delay(20);
  mfrc522.PCD_Init();
  delay(20);
  I2CMux.closeChannel(i);
 }
```

```
  I2CMux.closeAll();              // Lokk alle kanalene før start

  Serial.println("RFID-TCA9548A-5");
}

void loop()
{
  for (int i = 4; i < 8; i++)
  {
    arrayRFID[i] = 0;
    I2CMux.openChannel(i);
    readRFID(i);
    readRFID(i);
    I2CMux.closeChannel(i);
  }
  Serial.println("------------------");
  printArrayRFID();

  if (arrayRFID[4]*arrayRFID[5]*arrayRFID[6]*arrayRFID[7] != 0)
  {
    omstokking();
    produkt2();
  }
  delay(100);
}

void readRFID(int i)
{
  if ( mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial() )
  {
    // Skriv ut UID
    val = convertFourByteToLong(mfrc522.uid.uidByte);
    arrayRFID[i] = val;
  }
}

long convertFourByteToLong(byte UID[4])
```

```
{
  long val = 0;
  val += UID[0];
  val += UID[1];
  val += UID[2];
  val += UID[3];
  return val;
}

void printArrayRFID()
{
  for (int i = 0; i < 8; i++)
  {
    Serial.print("RFID nr: ");
    Serial.print(i);
    Serial.print(": ");
    Serial.println(arrayRFID[i]);
  }
  Serial.println();
}

void omstokking()
{
  for (int i = 0; i < 8; i++)
  {
    for (int j = 0; j < 8; j++)
    {
      if ((arrayRFID[i] == arrayRFIDTall[j][0]) && (arrayRFID[i] != 0)) T[i] = arrayRFIDTall[j][1];
    }
  }
}

int produkt2()
{
  // Regnestykket er AB * CD = Produkt eller (10*A+B)*(10*C+D) = Produkt eller
(10*T(4)+T(5))*(10*T(6)+T(7))=Produkt
  int Prod2 = (10 * T[4] + T[5]) * (10 * T[6] + T[7]);
  Serial.print("Produkt 2: ");
  Serial.println(Prod2);
```

```
  return Prod2;
}
```

B.7 Deloppdrag 7 – Testprogram for lesing av 2x4 tall og beregning av differansen mellom to produkter

Programmet leser åtte kort som er merket med tallene 1, 2, 3, 4, 5, 6, 7 og 8 og beregner differansen av to produkter som legges i to rammer for.

```
/*
  RFID-TCA9548A-7
  Programmet leser av ID-koden til flere kort koblet til en multiplekser
  og sjekker plasseringen av kortene og knytter ett tall til ID-kortets kode
  Nils Kr. Rossing 16.03.23
*/
#include <Arduino.h>
#include "MFRC522_I2C.h"
#include "TCA9548A.h"

#define RST 3
long arrayRFID[8];
int  arrayRFIDTall[8][2] = {{17744, 1}, {23668, 2}, {-2992, 3}, { -30729, 4}, {-3504, 5}, { -11834, 6},  {32336, 7}, {24180, 8}};
int  T[8] = {0, 0, 0, 0, 0, 0, 0, 0};
long val;

TCA9548A I2CMux(0x70);              // Address can be passed into the constructor
MFRC522 mfrc522(0x28, RST);   // Deklarer et MFRC522 objekt.

void setup() {
  Serial.begin(9600);

  I2CMux.begin(Wire);             // Wire instance is passed to the library

  for (int i = 0; i < 8; i++)
  {
    I2CMux.openChannel(i);
    delay(20);
    mfrc522.PCD_Init();
    delay(20);
    I2CMux.closeChannel(i);
```

```
  }

  I2CMux.closeAll();              // Lokk alle kanalene før start

  Serial.println("RFID-TCA9548A-6");
}

void loop()
{
  for (int i = 0; i < 8; i++)
  {
    arrayRFID[i] = 0;
    I2CMux.openChannel(i);
    readRFID(i);
    readRFID(i);
    I2CMux.closeChannel(i);
  }
  Serial.println("------------------");
  printArrayRFID();

  if (arrayRFID[0]*arrayRFID[1]*arrayRFID[2]*arrayRFID[3]*arrayRFID[4]*arrayRFID[5]*arrayRFID[6]*arrayR-
FID[7] != 0)
  {
    omstokking();
    //produkt1();
    //produkt2();
    differanse(); // produkt1 - produkt2 = Differnse
  }
  delay(100);
}

void readRFID(int i)
{
  if ( mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial() )
  {
    // Skriv ut UID
    val = convertFourByteToLong(mfrc522.uid.uidByte);
    arrayRFID[i] = val;
  }
```

```
}

long convertFourByteToLong(byte UID[4])
{
  long val = 0;
  val += UID[0];
  val += UID[1];
  val += UID[2];
  val += UID[3];
  return val;
}

void printArrayRFID()
{
  for (int i = 0; i < 8; i++)
  {
    Serial.print("RFID nr: ");
    Serial.print(i);
    Serial.print(": ");
    Serial.println(arrayRFID[i]);
  }
  Serial.println();
}

void omstokking()
{
  for (int i = 0; i < 8; i++)
  {
    for (int j = 0; j < 8; j++)
    {
      if ((arrayRFID[i] == arrayRFIDTall[j][0]) && (arrayRFID[i] != 0)) T[i] = arrayRFIDTall[j][1];
    }
  }
}

int produkt1()
{
  // Regnestykket er AB * CD = Produkt eller (10*A+B)*(10*C+D) = Produkt eller
(10*T(4)+T(5))*(10*T(6)+T(7))=Produkt
```

```
  int Prod1 = (10 * T[0] + T[1]) * (10 * T[2] + T[3]);
  Serial.print("Produkt 1: ");
  Serial.println(Prod1);
  return Prod1;
}

int produkt2()
{
  // Regnestykket er AB * CD = Produkt eller (10*A+B)*(10*C+D) = Produkt eller
(10*T(4)+T(5))*(10*T(6)+T(7))=Produkt
  int Prod2 = (10 * T[4] + T[5]) * (10 * T[6] + T[7]);
  Serial.print("Produkt 2: ");
  Serial.println(Prod2);
  return Prod2;
}

int differanse()
{
  // Regnestykket er
  int Diff = produkt1() - produkt2();
  Serial.print("Differanse: ");
  Serial.println(Diff);
  return Diff;
}
```

Vedlegg C Simulering av ulike varianter av "Nærmest null"

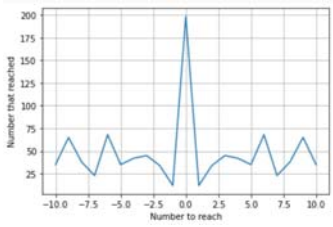
C.1 Simuleringsprogram i Python for simulering av oppgaver

Simuleringsprogram i Python for simulering av oppgaver med 9 siffer og ulike kombinasjoner av multiplikasjon og addisjon. Programmet kan f.eks. kjøres i Spyder.

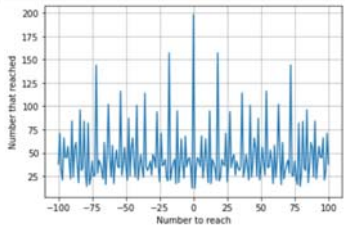
```
# -*- coding: utf-8 -*-
"""
Created on Thu Jul 27 15:57:50 2023

@author: nikro
"""

import itertools
import numpy as np

permlist = [ ]

val = [1, 2, 3, 4, 5, 6, 7, 8, 9]

perm_set = itertools.permutations(val)

for i in perm_set:
    permlist.append(i)

perm_arr = np.array(permlist)
print(perm_arr)

k = 0
j = 0
for E in perm_arr:
    a = perm_arr[k][0]
    b = perm_arr[k][1]
    c = perm_arr[k][2]
    d = perm_arr[k][3]
    e = perm_arr[k][4]
    f = perm_arr[k][5]
    g = perm_arr[k][6]
    h = perm_arr[k][7]
    i = perm_arr[k][8]
    sum = (a * (10 * b + c)) + (d * (10 * e + f)) - (g * (10 * h + i))
    #sum = (100 * a + 10 * b + c) + (100 * d + 10 * e + f) - (100 * g + 10 * h + i)
    #sum = (a * b * c) + (d * e * f) - (g * h * i)
    if (sum == 0):
       print(a, '*', b, c, '+', d, '*', e, f, '-', g, '*', h, i, '= 0')
       #print(a, b, c, '+',d, e, f, '-', g, h, i, '= 0')
       #print(a, '*', b, '*', c, '+', d, '*', e, '*', f, '-', g, '*', h, '*', i, '= 0')
       j = j + 1
    k = k + 1
print(j)
```

C.2 “Nærmest null” – Variant 1 (Tradisjonell utgave)

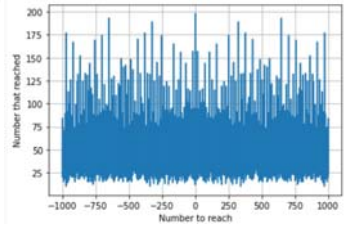
Følgende oppstilling er simulert: abc*de – fgh*ij = 0. Et interessant spørsmål er hvor mange løsninger ligningen har dersom svaret skal være forskjellig fra 0. Plottene under viser antall løsninger mellom -10 og 10, mellom -100 - 100 og mellom -1000 og 1000.



fra -10 til + 10



fra -100 til + 100



fra -1000 til + 1000

I alt er det 198 løsninger som gir verdien 0. I området +/– 1000 så er 0 det resultatet som har flest løsninger.

C.2.1 Løsninger som gir 0 for variant 1

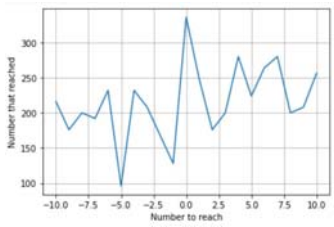
Tabellen under angir alle 198 løsningene på oppstilling: abc*de – fgh*ij = 0

046*79-158*23=0	154*29-638*07=0	276*45-138*90=0	480*27-135*96=0	732*60-915*48=0
054*69-138*27=0	156*23-897*04=0	279*30-186*45=0	481*70-962*35=0	732*80-915*64=0
054*93-186*27=0	158*23-046*79=0	279*30-465*18=0	485*26-130*97=0	736*51-408*92=0
058*67-134*29=0	158*23-079*46=0	290*38-145*76=0	485*26-970*13=0	748*20-935*16=0
058*69-174*23=0	158*32-064*79=0	293*14-586*07=0	485*32-160*97=0	754*31-806*29=0
058*73-146*29=0	158*32-079*64=0	296*35-148*70=0	485*32-970*16=0	759*08-132*46=0
058*96-174*32=0	160*97-485*32=0	296*51-408*37=0	485*62-310*97=0	760*23-184*95=0
063*74-259*18=0	165*78-429*30=0	301*74-259*86=0	485*62-970*31=0	760*31-248*95=0
064*79-158*32=0	170*94-235*68=0	306*27-459*18=0	486*30-972*15=0	760*41-328*95=0
067*58-134*29=0	174*23-058*69=0	307*58-614*29=0	496*35-217*80=0	762*45-381*90=0
069*54-138*27=0	174*23-069*58=0	309*54-618*27=0	504*89-712*63=0	782*09-153*46=0
069*58-174*23=0	174*32-058*96=0	310*76-248*95=0	504*91-728*63=0	782*53-901*46=0
073*58-146*29=0	174*32-096*58=0	310*97-485*62=0	506*34-187*92=0	795*24-318*60=0
073*96-584*12=0	184*95-230*76=0	318*60-795*24=0	518*29-406*37=0	806*29-754*31=0
074*63-259*18=0	184*95-760*23=0	327*18-654*09=0	532*14-076*98=0	819*54-702*63=0
076*98-532*14=0	185*92-370*46=0	328*95-410*76=0	532*14-098*76=0	890*36-712*45=0
079*46-158*23=0	185*92-460*37=0	328*95-760*41=0	534*09-267*18=0	895*24-716*30=0
079*64-158*32=0	186*27-054*93=0	329*14-658*07=0	538*07-269*14=0	895*32-716*40=0
093*54-186*27=0	186*27-093*54=0	345*26-897*10=0	546*09-273*18=0	897*04-156*23=0
096*58-174*32=0	186*45-279*30=0	351*96-702*48=0	584*12-073*96=0	897*10-345*26=0
096*73-584*12=0	186*90-372*45=0	360*89-712*45=0	584*12-096*73=0	901*28-476*53=0
098*76-532*14=0	187*92-506*34=0	360*91-728*45=0	586*07-293*14=0	901*46-782*53=0
123*56-984*07=0	195*72-468*30=0	370*46-185*92=0	614*29-307*58=0	902*38-451*76=0
130*97-485*26=0	195*84-273*60=0	372*45-186*90=0	618*27-309*54=0	910*36-728*45=0
132*46-759*08=0	217*80-496*35=0	380*29-145*76=0	618*45-927*30=0	915*48-732*60=0
134*29-058*67=0	230*76-184*95=0	381*90-762*45=0	630*27-945*18=0	915*64-732*80=0
134*29-067*58=0	235*68-170*94=0	406*37-518*29=0	632*51-408*79=0	927*30-618*45=0
135*96-270*48=0	235*68-940*17=0	408*37-296*51=0	638*07-154*29=0	935*16-748*20=0
135*96-480*27=0	237*60-948*15=0	408*79-632*51=0	654*09-327*18=0	940*17-235*68=0
136*27-459*08=0	248*95-310*76=0	408*92-736*51=0	658*07-329*14=0	945*18-270*63=0
138*27-054*69=0	248*95-760*31=0	410*76-328*95=0	702*48-351*96=0	945*18-630*27=0
138*27-069*54=0	259*18-063*74=0	429*30-165*78=0	702*63-819*54=0	948*15-237*60=0
138*90-276*45=0	259*18-074*63=0	451*76-902*38=0	712*45-360*89=0	962*35-481*70=0
145*76-290*38=0	259*86-301*74=0	459*08-136*27=0	712*45-890*36=0	970*13-485*26=0
145*76-380*29=0	267*18-534*09=0	459*18-306*27=0	712*63-504*89=0	970*16-485*32=0
146*29-058*73=0	269*14-538*07=0	460*37-185*92=0	716*30-895*24=0	970*31-485*62=0
146*29-073*58=0	270*48-135*96=0	465*18-279*30=0	716*40-895*32=0	972*15-486*30=0
148*70-296*35=0	270*63-945*18=0	468*30-195*72=0	728*45-360*91=0	984*07-123*56=0
153*28-476*09=0	273*18-546*09=0	476*09-153*28=0	728*45-910*36=0	
153*46-782*09=0	273*60-195*84=0	476*53-901*28=0	728*63-504*91=0	

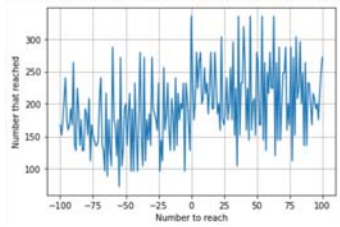
C.3 “Nærmest null” – Variant 2

Følgende oppstilling er simulert: abc + def – ghi = 0
I alt finnes det 336 løsninger. En del av dem er “symmetriske”, dvs. at første og andre siffer bare har byttet plass.

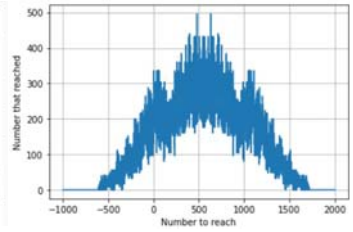
Et interessant spørsmål er hvor mange løsninger ligningen har dersom svaret skal være noe forskjellig fra 0. Plottene under viser antall løsninger mellom -10 og 10, mellom -100 og 100 og mellom -1000 og 1000.



fra -10 til + 10



fra -100 til + 100



fra -1000 til + 1000

Vi legger merke til kurven er symmetrisk om ca. 550.

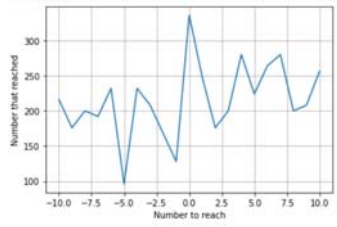
C.3.1 Løsninger som gir 0 for variant 2

Tabellen under angir alle 336 løsningene på oppstilling: abc + def – ghi = 0

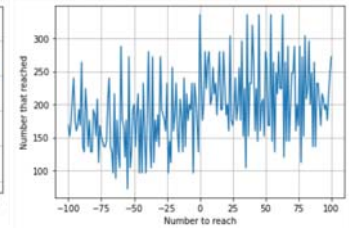
124+659-783=0	215+478-693=0	294+381-675=0	418+275-693=0	627+318-945=0
125+739-864=0	215+748-963=0	295+173-468=0	419+238-657=0	627+354-981=0
127+359-486=0	216+378-594=0	296+541-837=0	428+139-567=0	628+317-945=0
127+368-495=0	216+738-954=0	297+351-648=0	429+138-567=0	629+154-783=0
128+367-495=0	218+349-567=0	314+658-972=0	438+129-567=0	634+158-792=0
128+439-567=0	218+376-594=0	316+278-594=0	438+219-657=0	634+257-891=0
129+357-486=0	218+439-657=0	317+529-846=0	439+128-567=0	637+254-891=0
129+438-567=0	218+475-693=0	317+628-945=0	439+218-657=0	638+154-792=0
129+654-783=0	218+736-954=0	318+249-567=0	452+187-639=0	642+195-837=0
129+735-864=0	218+745-963=0	318+276-594=0	452+367-819=0	643+275-918=0
134+658-792=0	219+348-567=0	318+627-945=0	457+182-639=0	645+192-837=0
135+729-864=0	219+438-657=0	318+654-972=0	457+362-819=0	645+273-918=0
138+429-567=0	219+564-783=0	319+248-567=0	462+357-819=0	654+129-783=0
138+654-792=0	219+654-873=0	319+527-846=0	467+352-819=0	654+138-792=0
139+428-567=0	234+657-891=0	324+567-891=0	475+218-693=0	654+219-873=0
139+725-864=0	235+746-981=0	324+657-981=0	478+215-693=0	654+237-891=0
142+596-738=0	236+718-954=0	327+159-486=0	482+157-639=0	654+318-972=0
142+695-837=0	236+745-981=0	327+168-495=0	482+193-675=0	654+327-981=0
143+586-729=0	237+654-891=0	327+519-846=0	483+192-675=0	657+234-891=0
145+692-837=0	238+419-657=0	327+564-891=0	487+152-639=0	657+324-981=0
146+583-729=0	238+716-954=0	327+618-945=0	492+183-675=0	658+134-792=0
146+592-738=0	239+418-657=0	327+654-981=0	493+182-675=0	658+314-972=0
152+487-639=0	241+596-837=0	328+167-495=0	514+269-783=0	659+124-783=0
152+784-936=0	243+576-819=0	328+617-945=0	517+329-846=0	659+214-873=0
154+629-783=0	243+675-918=0	329+157-486=0	519+264-783=0	671+283-954=0
154+638-792=0	245+673-918=0	329+517-846=0	519+327-846=0	673+245-918=0
154+782-936=0	245+718-963=0	341+586-927=0	524+367-891=0	673+281-954=0
157+329-486=0	245+736-981=0	342+576-918=0	527+319-846=0	675+243-918=0
157+482-639=0	246+573-819=0	346+572-918=0	527+364-891=0	681+273-954=0
158+634-792=0	246+591-837=0	346+581-927=0	529+317-846=0	683+271-954=0
159+327-486=0	246+735-981=0	348+219-567=0	541+296-837=0	692+145-837=0
159+624-783=0	248+319-567=0	349+218-567=0	541+386-927=0	695+142-837=0
162+387-549=0	248+715-963=0	351+297-648=0	542+196-738=0	715+248-963=0
162+783-945=0	249+318-567=0	352+467-819=0	542+376-918=0	716+238-954=0
163+782-945=0	251+397-648=0	354+618-972=0	543+186-729=0	718+236-954=0
167+328-495=0	254+619-873=0	354+627-981=0	543+276-819=0	718+245-963=0
167+382-549=0	254+637-891=0	357+129-486=0	546+183-729=0	725+139-864=0
168+327-495=0	257+391-648=0	357+291-648=0	546+192-738=0	729+135-864=0
173+286-459=0	257+634-891=0	357+462-819=0	546+273-819=0	735+129-864=0
173+295-468=0	259+614-873=0	357+624-981=0	546+291-837=0	735+246-981=0
175+293-468=0	264+519-783=0	358+614-972=0	546+372-918=0	736+218-954=0
176+283-459=0	269+514-783=0	359+127-486=0	546+381-927=0	736+245-981=0
182+367-549=0	271+593-864=0	362+187-549=0	564+219-783=0	738+216-954=0
182+394-576=0	271+683-954=0	362+457-819=0	564+327-891=0	739+125-864=0
182+457-639=0	273+186-459=0	364+527-891=0	567+324-891=0	745+218-963=0
182+493-675=0	273+195-468=0	367+128-495=0	569+214-783=0	745+236-981=0
182+754-936=0	273+546-819=0	367+182-549=0	571+293-864=0	746+235-981=0
182+763-945=0	273+591-864=0	367+452-819=0	572+346-918=0	748+215-963=0
183+276-459=0	273+645-918=0	367+524-891=0	573+246-819=0	752+184-936=0
183+492-675=0	273+681-954=0	368+127-495=0	573+291-864=0	754+182-936=0
183+546-729=0	275+193-468=0	372+546-918=0	576+243-819=0	762+183-945=0
183+762-945=0	275+418-693=0	376+218-594=0	576+342-918=0	763+182-945=0
184+392-576=0	275+643-918=0	376+542-918=0	581+346-927=0	782+154-936=0
184+752-936=0	276+183-459=0	378+216-594=0	583+146-729=0	782+163-945=0
186+273-459=0	276+318-594=0	381+294-675=0	586+143-729=0	783+162-945=0
186+543-729=0	276+543-819=0	381+546-927=0	586+341-927=0	784+152-936=0
187+362-549=0	278+316-594=0	382+167-549=0	591+246-837=0	
187+452-639=0	278+415-693=0	382+194-576=0	591+273-864=0	
192+384-576=0	281+394-675=0	384+192-576=0	592+146-738=0	
192+483-675=0	281+673-954=0	384+291-675=0	593+271-864=0	
192+546-738=0	283+176-459=0	386+541-927=0	596+142-738=0	
192+645-837=0	283+671-954=0	387+162-549=0	596+241-837=0	
193+275-468=0	284+391-675=0	391+257-648=0	614+259-873=0	
193+482-675=0	286+173-459=0	391+284-675=0	614+358-972=0	
194+382-576=0	291+357-648=0	392+184-576=0	617+328-945=0	
195+273-468=0	291+384-675=0	394+182-576=0	618+327-945=0	
195+642-837=0	291+546-837=0	394+281-675=0	618+354-972=0	
196+542-738=0	291+573-864=0	397+251-648=0	619+254-873=0	
214+569-783=0	293+175-468=0	415+278-693=0	624+159-783=0	
214+659-873=0	293+571-864=0	418+239-657=0	624+357-981=0	

C.4 “Nærmest null” – Variant 3

Følgende oppstilling er simulert: a*bc + d*ef – g*hi = 0. Et interessant spørsmål er hvor mange løsninger ligningen har dersom svaret skal være forskjellig fra 0. Plottene under viser antall løsninger mellom -10 og 10, mellom -100 og - 100.



fra -10 til + 10



fra -100 til + 100

Vi ser at denne varianten har opp mot 500 løsninger og er usymmetrisk i det minste rundt 0.

C.4.1 Løsninger som gir 0 for variant 3

Tabellen under angir alle 310 løsningene på oppstilling: a*bc + d*ef – g*hi = 0

1*23+9*48-7*65=0	2*87+3*65-9*41=0	4*18+9*27-5*63=0	6*29+3*18-4*57=0	9*21+7*45-8*63=0
1*25+9*63-8*74=0	2*87+4*15-6*39=0	4*19+3*78-5*62=0	6*29+4*15-3*78=0	9*23+6*18-7*45=0
1*27+6*48-9*35=0	2*93+4*75-6*81=0	4*19+8*36-7*52=0	6*31+2*95-8*47=0	9*24+5*16-8*37=0
1*27+9*53-6*84=0	2*93+5*48-6*71=0	4*21+3*87-5*69=0	6*35+4*91-7*82=0	9*26+1*58-4*73=0
1*28+3*69-5*47=0	2*94+1*67-3*85=0	4*21+6*73-9*58=0	6*39+2*51-7*48=0	9*27+1*45-8*36=0
1*28+5*63-7*49=0	2*95+4*17-3*86=0	4*25+8*19-7*36=0	6*42+5*81-9*73=0	9*27+3*15-6*48=0
1*29+5*83-6*74=0	2*95+6*31-8*47=0	4*28+9*35-7*61=0	6*42+9*35-7*81=0	9*27+4*18-5*63=0
1*32+4*79-6*58=0	2*95+7*14-8*36=0	4*29+1*85-3*67=0	6*45+1*72-9*38=0	9*28+1*63-7*45=0
1*37+8*26-5*49=0	2*97+1*58-4*63=0	4*31+8*65-7*92=0	6*47+2*15-8*39=0	9*28+3*16-4*75=0
1*38+6*72-5*94=0	3*12+4*89-7*56=0	4*35+7*16-9*28=0	6*47+5*12-9*38=0	9*34+2*17-5*68=0
1*42+6*93-8*75=0	3*12+6*94-8*75=0	4*37+9*18-5*62=0	6*48+1*27-9*35=0	9*34+2*18-6*57=0
1*43+6*92-7*85=0	3*12+9*54-6*87=0	4*57+8*12-9*36=0	6*72+1*38-5*94=0	9*34+2*71-8*56=0
1*45+9*27-8*36=0	3*14+9*62-8*75=0	4*62+1*85-9*37=0	6*72+3*18-9*54=0	9*35+4*28-7*61=0
1*48+3*65-9*27=0	3*15+9*27-6*48=0	4*65+2*17-3*98=0	6*73+4*21-9*58=0	9*35+6*42-7*81=0
1*49+2*68-5*37=0	3*16+9*28-4*75=0	4*75+2*93-6*81=0	6*81+3*54-9*72=0	9*36+1*52-8*47=0
1*49+3*87-5*62=0	3*18+4*95-7*62=0	4*75+3*82-6*91=0	6*92+1*43-7*85=0	9*36+2*41-7*58=0
1*52+9*36-8*47=0	3*18+5*72-9*46=0	4*79+1*32-6*58=0	6*93+1*42-8*75=0	9*38+2*51-6*74=0
1*58+2*97-4*63=0	3*18+6*29-4*57=0	4*82+5*31-7*69=0	6*93+2*15-7*84=0	9*41+5*27-8*63=0
1*58+9*26-4*73=0	3*18+6*72-9*54=0	4*85+3*29-7*61=0	6*94+3*12-8*75=0	9*42+6*13-8*57=0
1*63+9*28-7*45=0	3*18+7*26-4*59=0	4*89+2*35-6*71=0	7*12+5*48-9*36=0	9*43+7*15-6*82=0
1*67+2*94-3*85=0	3*21+4*98-7*65=0	4*89+3*12-7*56=0	7*13+9*45-8*62=0	9*45+2*18-7*63=0
1*68+3*45-7*29=0	3*21+5*84-7*69=0	4*91+2*73-6*85=0	7*14+2*95-8*36=0	9*45+3*21-6*78=0
1*69+2*83-5*47=0	3*21+8*49-7*65=0	4*91+6*35-7*82=0	7*14+8*32-6*59=0	9*45+3*27-6*81=0
1*69+3*82-7*45=0	3*21+9*45-6*78=0	4*93+2*57-6*81=0	7*15+9*43-6*82=0	9*45+7*13-8*62=0
1*72+3*96-8*45=0	3*27+9*45-6*81=0	4*95+3*18-7*62=0	7*16+2*85-3*94=0	9*48+1*23-7*65=0
1*72+6*45-9*38=0	3*28+9*16-4*57=0	4*98+3*21-7*65=0	7*16+4*35-9*28=0	9*53+1*27-6*84=0
1*73+2*86-5*49=0	3*29+4*85-7*61=0	5*12+3*78-6*49=0	7*18+2*54-6*39=0	9*54+3*12-6*87=0
1*76+2*49-3*58=0	3*41+5*87-9*62=0	5*12+3*96-4*87=0	7*19+3*65-4*82=0	9*62+3*14-8*75=0
1*78+5*42-3*96=0	3*45+1*68-7*29=0	5*12+6*47-9*38=0	7*21+3*86-9*45=0	9*63+1*25-8*74=0
1*79+2*68-5*43=0	3*45+6*18-9*27=0	5*14+2*79-6*38=0	7*21+5*84-9*63=0	9*63+2*14-7*85=0
1*82+3*96-5*74=0	3*46+1*87-9*25=0	5*16+9*24-8*37=0	7*24+8*15-3*96=0	
1*83+6*27-5*49=0	3*49+8*26-5*71=0	5*21+3*69-4*78=0	7*26+3*18-4*59=0	
1*84+2*75-6*39=0	3*51+7*49-8*62=0	5*24+1*87-3*69=0	7*28+3*56-4*91=0	
1*85+4*29-3*67=0	3*54+6*81-9*72=0	5*24+6*19-3*78=0	7*29+8*14-5*63=0	
1*85+4*62-9*37=0	3*56+7*28-4*91=0	5*26+3*78-4*91=0	7*36+9*12-8*45=0	
1*87+3*46-9*25=0	3*58+6*19-4*72=0	5*26+9*18-4*73=0	7*42+3*85-9*61=0	
1*87+5*24-3*69=0	3*58+9*12-6*47=0	5*27+4*18-3*69=0	7*45+3*81-9*62=0	
1*92+5*38-6*47=0	3*65+1*48-9*27=0	5*27+9*41-8*63=0	7*45+9*21-8*63=0	
1*96+2*47-5*38=0	3*65+2*87-9*41=0	5*31+4*82-7*69=0	7*49+3*51-8*62=0	
2*14+9*63-7*85=0	3*65+7*19-4*82=0	5*32+9*18-7*46=0	7*49+6*12-5*83=0	
2*15+6*47-8*39=0	3*67+9*15-8*42=0	5*38+1*92-6*47=0	8*12+4*57-9*36=0	
2*15+6*93-7*84=0	3*69+1*28-5*47=0	5*38+6*21-4*79=0	8*14+7*29-5*63=0	
2*16+5*47-3*89=0	3*69+5*21-4*78=0	5*41+3*97-8*62=0	8*15+6*27-3*94=0	
2*17+4*65-3*98=0	3*72+9*16-8*45=0	5*42+1*78-3*96=0	8*15+7*24-3*96=0	
2*17+5*94-8*63=0	3*78+2*65-4*91=0	5*42+6*17-8*39=0	8*19+4*25-7*36=0	
2*17+9*34-5*68=0	3*78+4*19-5*62=0	5*47+2*16-3*89=0	8*21+3*94-6*75=0	
2*18+9*34-6*57=0	3*78+5*12-6*49=0	5*48+2*93-6*71=0	8*26+1*37-5*49=0	
2*18+9*45-7*63=0	3*78+5*26-4*91=0	5*48+7*12-9*36=0	8*26+3*49-5*71=0	
2*35+4*89-6*71=0	3*81+6*27-9*45=0	5*63+1*28-7*49=0	8*27+6*15-9*34=0	
2*41+9*36-7*58=0	3*81+7*45-9*62=0	5*72+3*18-9*46=0	8*32+6*19-5*74=0	
2*45+9*16-3*78=0	3*82+1*69-7*45=0	5*81+6*42-9*73=0	8*32+7*14-6*59=0	
2*45+9*18-7*36=0	3*82+4*75-6*91=0	5*83+1*29-6*74=0	8*34+9*12-5*76=0	
2*47+1*96-5*38=0	3*85+7*42-9*61=0	5*84+3*21-7*69=0	8*36+4*19-7*52=0	
2*49+1*76-3*58=0	3*85+9*21-6*74=0	5*84+7*21-9*63=0	8*49+3*21-7*65=0	
2*51+6*39-7*48=0	3*86+7*21-9*45=0	5*87+3*41-9*62=0	8*65+4*31-7*92=0	
2*51+9*38-6*74=0	3*87+1*49-5*62=0	5*94+2*17-8*63=0	9*12+3*58-6*47=0	
2*53+9*18-4*67=0	3*87+4*21-5*69=0	6*12+7*49-5*83=0	9*12+7*36-8*45=0	
2*54+7*18-6*39=0	3*94+8*21-6*75=0	6*13+9*42-8*57=0	9*12+8*34-5*76=0	
2*57+4*93-6*81=0	3*96+1*72-8*45=0	6*15+8*27-9*34=0	9*15+3*67-8*42=0	
2*65+3*78-4*91=0	3*96+1*82-5*74=0	6*17+5*42-8*39=0	9*16+2*45-3*78=0	
2*65+9*18-4*73=0	3*96+4*18-5*72=0	6*18+3*45-9*27=0	9*16+3*28-4*57=0	
2*68+1*49-5*37=0	3*96+5*12-4*87=0	6*18+9*23-7*45=0	9*16+3*72-8*45=0	
2*68+1*79-5*43=0	3*97+5*41-8*62=0	6*19+3*58-4*72=0	9*18+2*45-7*36=0	
2*71+9*34-8*56=0	3*98+4*12-6*57=0	6*19+5*24-3*78=0	9*18+2*53-4*67=0	
2*73+4*91-6*85=0	4*12+3*98-6*57=0	6*19+8*32-5*74=0	9*18+2*65-4*73=0	
2*75+1*84-6*39=0	4*15+2*87-6*39=0	6*21+5*38-4*79=0	9*18+4*37-5*62=0	
2*79+5*14-6*38=0	4*15+6*29-3*78=0	6*23+9*18-4*75=0	9*18+5*26-4*73=0	
2*83+1*69-5*47=0	4*17+2*95-3*86=0	6*27+1*83-5*49=0	9*18+5*32-7*46=0	
2*85+7*16-3*94=0	4*18+3*96-5*72=0	6*27+3*81-9*45=0	9*18+6*23-4*75=0	
2*86+1*73-5*49=0	4*18+5*27-3*69=0	6*27+8*15-3*94=0	9*21+3*85-6*74=0	

C.5 “Nærmest null” – Variant 4

Følgende oppstilling er simulert: a*b*c + d*e*f – g*h*i = 0 og gir 1296 svar som oppfyller ligningen Lista under viser noen av de 1296 riktige svarene.

6*2*1+3*8*7-4*9*5=0 6*5*1+4*3*8-9*2*7=0 7*2*6+9*1*4-3*8*5=0 7*6*2+9*4*1-8*3*5=0 8*3*7+1*6*2-4*9*5=0
6*2*1+3*8*7-5*4*9=0 6*5*1+4*3*8-9*7*2=0 7*2*6+9*1*4-5*3*8=0 7*6*2+9*4*1-8*5*3=0 8*3*7+1*6*2-5*4*9=0
6*2*1+3*8*7-5*9*4=0 6*5*1+4*8*3-2*7*9=0 7*2*6+9*1*4-5*8*3=0 7*8*3+1*2*6-4*5*9=0 8*3*7+1*6*2-5*9*4=0
6*2*1+3*8*7-9*4*5=0 6*5*1+4*8*3-2*9*7=0 7*2*6+9*1*4-8*3*5=0 7*8*3+1*2*6-4*9*5=0 8*3*7+1*6*2-9*4*5=0
6*2*1+3*8*7-9*5*4=0 6*5*1+4*8*3-7*2*9=0 7*2*6+9*1*4-8*5*3=0 7*8*3+1*2*6-5*4*9=0 8*3*7+1*6*2-9*5*4=0
6*2*1+7*3*8-4*5*9=0 6*5*1+4*8*3-7*9*2=0 7*2*6+9*4*1-3*5*8=0 7*8*3+1*2*6-5*9*4=0 8*3*7+2*1*6-4*5*9=0
6*2*1+7*3*8-4*9*5=0 6*5*1+4*8*3-9*2*7=0 7*2*6+9*4*1-3*8*5=0 7*8*3+1*2*6-9*4*5=0 8*3*7+2*1*6-4*9*5=0
6*2*1+7*3*8-5*4*9=0 6*5*1+4*8*3-9*7*2=0 7*2*6+9*4*1-5*3*8=0 7*8*3+1*2*6-9*5*4=0 8*3*7+2*1*6-5*4*9=0
6*2*1+7*3*8-5*9*4=0 6*5*1+8*3*4-2*7*9=0 7*2*6+9*4*1-5*8*3=0 7*8*3+1*6*2-4*5*9=0 8*3*7+2*1*6-5*9*4=0
6*2*1+7*3*8-9*4*5=0 6*5*1+8*3*4-2*9*7=0 7*2*6+9*4*1-8*3*5=0 7*8*3+1*6*2-4*9*5=0 8*3*7+2*1*6-9*4*5=0
6*2*1+7*3*8-9*5*4=0 6*5*1+8*3*4-7*2*9=0 7*2*6+9*4*1-8*5*3=0 7*8*3+1*6*2-5*4*9=0 8*3*7+2*1*6-9*5*4=0
6*2*1+7*8*3-4*5*9=0 6*5*1+8*3*4-7*9*2=0 7*3*8+1*2*6-4*5*9=0 7*8*3+1*6*2-5*9*4=0 8*3*7+2*6*1-4*5*9=0
6*2*1+7*8*3-4*9*5=0 6*5*1+8*3*4-9*2*7=0 7*3*8+1*2*6-4*9*5=0 7*8*3+1*6*2-9*4*5=0 8*3*7+2*6*1-4*9*5=0
6*2*1+7*8*3-5*4*9=0 6*5*1+8*3*4-9*7*2=0 7*3*8+1*2*6-5*4*9=0 7*8*3+1*6*2-9*5*4=0 8*3*7+2*6*1-5*4*9=0
6*2*1+7*8*3-5*9*4=0 6*5*1+8*4*3-2*7*9=0 7*3*8+1*2*6-5*9*4=0 7*8*3+2*1*6-4*5*9=0 8*3*7+2*6*1-5*9*4=0
6*2*1+7*8*3-9*4*5=0 6*5*1+8*4*3-2*9*7=0 7*3*8+1*2*6-9*4*5=0 7*8*3+2*1*6-4*9*5=0 8*3*7+2*6*1-9*4*5=0
6*2*1+7*8*3-9*5*4=0 6*5*1+8*4*3-7*2*9=0 7*3*8+1*2*6-9*5*4=0 7*8*3+2*1*6-5*4*9=0 8*3*7+2*6*1-9*5*4=0
6*2*1+8*3*7-4*5*9=0 6*5*1+8*4*3-7*9*2=0 7*3*8+1*6*2-4*5*9=0 7*8*3+2*1*6-5*9*4=0 8*3*7+6*1*2-4*5*9=0
6*2*1+8*3*7-4*9*5=0 6*5*1+8*4*3-9*2*7=0 7*3*8+1*6*2-4*9*5=0 7*8*3+2*1*6-9*4*5=0 8*3*7+6*1*2-4*9*5=0
6*2*1+8*3*7-5*4*9=0 6*5*1+8*4*3-9*7*2=0 7*3*8+1*6*2-5*4*9=0 7*8*3+2*1*6-9*5*4=0 8*3*7+6*1*2-5*4*9=0
6*2*1+8*3*7-5*9*4=0 6*7*2+1*4*9-3*5*8=0 7*3*8+1*6*2-5*9*4=0 7*8*3+2*6*1-4*5*9=0 8*3*7+6*1*2-5*9*4=0
6*2*1+8*3*7-9*4*5=0 6*7*2+1*4*9-3*8*5=0 7*3*8+1*6*2-9*4*5=0 7*8*3+2*6*1-4*9*5=0 8*3*7+6*1*2-9*4*5=0
6*2*1+8*3*7-9*5*4=0 6*7*2+1*4*9-5*3*8=0 7*3*8+1*6*2-9*5*4=0 7*8*3+2*6*1-5*4*9=0 8*3*7+6*1*2-9*5*4=0
6*2*1+8*7*3-4*5*9=0 6*7*2+1*4*9-5*8*3=0 7*3*8+2*1*6-4*5*9=0 7*8*3+2*6*1-5*9*4=0 8*3*7+6*2*1-4*5*9=0
6*2*1+8*7*3-4*9*5=0 6*7*2+1*4*9-8*3*5=0 7*3*8+2*1*6-4*9*5=0 7*8*3+2*6*1-9*5*4=0 8*3*7+6*2*1-4*9*5=0
6*2*1+8*7*3-5*4*9=0 6*7*2+1*4*9-8*5*3=0 7*3*8+2*1*6-5*4*9=0 7*8*3+2*6*1-9*4*5=0 8*3*7+6*2*1-5*4*9=0
6*2*1+8*7*3-5*9*4=0 6*7*2+1*9*4-3*5*8=0 7*3*8+2*1*6-5*9*4=0 7*8*3+6*1*2-4*5*9=0 8*3*7+6*2*1-5*9*4=0
6*2*1+8*7*3-9*4*5=0 6*7*2+1*9*4-3*8*5=0 7*3*8+2*1*6-9*4*5=0 7*8*3+6*1*2-5*4*9=0 8*3*7+6*2*1-9*4*5=0
6*2*1+8*7*3-9*5*4=0 6*7*2+1*9*4-5*3*8=0 7*3*8+2*1*6-9*5*4=0 7*8*3+6*1*2-9*4*5=0 8*3*7+6*2*1-9*5*4=0
6*2*7+1*4*9-3*5*8=0 6*7*2+1*9*4-5*8*3=0 7*3*8+2*6*1-4*5*9=0 7*8*3+6*1*2-9*5*4=0 8*4*3+1*5*6-2*7*9=0
6*2*7+1*4*9-3*8*5=0 6*7*2+1*9*4-8*3*5=0 7*3*8+2*6*1-4*9*5=0 7*8*3+6*1*2-9*5*4=0 8*4*3+1*5*6-7*2*9=0
6*2*7+1*4*9-5*3*8=0 6*7*2+1*9*4-8*5*3=0 7*3*8+2*6*1-5*4*9=0 7*8*3+6*2*1-4*5*9=0 8*4*3+1*5*6-7*9*2=0
6*2*7+1*4*9-5*8*3=0 6*7*2+4*1*9-8*3*5=0 7*3*8+2*6*1-5*9*4=0 7*8*3+6*2*1-5*4*9=0 8*4*3+1*5*6-9*2*7=0
6*2*7+1*4*9-8*3*5=0 6*7*2+4*1*9-3*8*5=0 7*3*8+2*6*1-9*4*5=0 7*8*3+6*2*1-4*9*5=0 8*4*3+1*5*6-9*7*2=0
6*2*7+1*4*9-8*5*3=0 6*7*2+4*1*9-3*8*5=0 7*3*8+2*6*1-9*5*4=0 7*8*3+6*2*1-5*9*4=0 8*4*3+1*5*6-9*7*2=0
6*2*7+1*9*4-3*5*8=0 6*7*2+4*1*9-5*3*8=0 7*3*8+6*1*2-4*5*9=0 7*8*3+6*2*1-9*4*5=0 8*4*3+1*6*5-2*7*9=0
6*2*7+1*9*4-3*8*5=0 6*7*2+4*1*9-8*3*5=0 7*3*8+6*1*2-4*9*5=0 7*8*3+6*2*1-9*5*4=0 8*4*3+1*6*5-2*9*7=0
6*2*7+1*9*4-5*3*8=0 6*7*2+4*1*9-8*5*3=0 7*3*8+6*1*2-5*4*9=0 7*8*3+6*2*1-5*9*4=0 8*4*3+1*6*5-7*2*9=0
6*2*7+1*9*4-5*8*3=0 6*7*2+4*9*1-3*5*8=0 7*3*8+6*1*2-5*9*4=0 8*3*4+1*5*6-2*7*9=0 8*4*3+1*6*5-7*9*2=0
6*2*7+1*9*4-8*3*5=0 6*7*2+4*9*1-3*8*5=0 7*3*8+6*1*2-9*4*5=0 8*3*4+1*5*6-2*9*7=0 8*4*3+1*6*5-9*2*7=0
6*2*7+1*9*4-8*5*3=0 6*7*2+4*9*1-5*3*8=0 7*3*8+6*1*2-9*5*4=0 8*3*4+1*5*6-7*2*9=0 8*4*3+1*6*5-9*7*2=0
6*2*7+4*1*9-3*5*8=0 6*7*2+4*9*1-5*8*3=0 7*3*8+6*2*1-4*5*9=0 8*3*4+1*5*6-7*9*2=0 8*4*3+1*6*5-9*7*2=0
6*2*7+4*1*9-3*8*5=0 6*7*2+4*9*1-8*3*5=0 7*3*8+6*2*1-4*9*5=0 8*3*4+1*5*6-9*2*7=0 8*4*3+5*1*6-2*9*7=0
6*2*7+4*1*9-5*3*8=0 6*7*2+4*9*1-8*5*3=0 7*3*8+6*2*1-5*4*9=0 8*3*4+1*5*6-9*7*2=0 8*4*3+5*1*6-7*9*2=0
6*2*7+4*1*9-5*8*3=0 6*7*2+4*9*1-8*5*3=0 7*3*8+6*2*1-5*9*4=0 8*3*4+1*6*5-2*7*9=0 8*4*3+5*1*6-7*9*2=0
6*2*7+4*1*9-8*3*5=0 6*7*2+9*1*4-3*5*8=0 7*3*8+6*2*1-9*4*5=0 8*3*4+1*6*5-2*9*7=0 8*4*3+5*1*6-9*7*2=0
6*2*7+4*1*9-8*5*3=0 6*7*2+9*1*4-3*8*5=0 7*3*8+6*2*1-9*5*4=0 8*3*4+1*6*5-7*2*9=0 8*4*3+5*1*6-9*7*2=0
6*2*7+4*9*1-3*5*8=0 6*7*2+9*1*4-5*3*8=0 7*6*2+1*4*9-3*5*8=0 8*3*4+1*6*5-7*9*2=0 8*4*3+5*1*6-9*7*2=0
6*2*7+4*9*1-3*8*5=0 6*7*2+9*1*4-8*3*5=0 7*6*2+1*4*9-3*8*5=0 8*3*4+1*6*5-9*2*7=0 8*4*3+5*1*6-9*7*2=0
6*2*7+4*9*1-5*3*8=0 6*7*2+9*1*4-8*5*3=0 7*6*2+1*4*9-5*3*8=0 8*3*4+1*6*5-9*7*2=0 8*4*3+5*1*6-9*7*2=0
6*2*7+4*9*1-5*8*3=0 6*7*2+9*4*1-3*5*8=0 7*6*2+1*4*9-5*8*3=0 8*3*4+5*1*6-2*7*9=0 8*4*3+5*1*6-7*9*2=0
6*2*7+4*9*1-8*3*5=0 6*7*2+9*4*1-3*8*5=0 7*6*2+1*4*9-8*3*5=0 8*3*4+5*1*6-2*9*7=0 8*4*3+5*1*6-9*2*7=0
6*2*7+4*9*1-8*5*3=0 6*7*2+9*4*1-5*3*8=0 7*6*2+1*4*9-8*5*3=0 8*3*4+5*1*6-7*2*9=0 8*4*3+5*1*6-9*2*7=0
6*2*7+9*1*4-3*5*8=0 6*7*2+9*4*1-5*8*3=0 7*6*2+1*4*9-3*5*8=0 8*3*4+5*1*6-7*9*2=0 8*4*3+5*1*6-9*2*7=0
6*2*7+9*1*4-3*8*5=0 6*7*2+9*4*1-8*3*5=0 7*6*2+1*4*9-8*3*5=0 8*3*4+5*1*6-9*2*7=0 8*4*3+6*1*5-2*7*9=0
6*2*7+9*1*4-5*3*8=0 6*7*2+9*4*1-8*5*3=0 7*6*2+1*4*9-8*5*3=0 8*3*4+5*1*6-7*9*2=0 8*4*3+6*1*5-2*9*7=0
6*2*7+9*1*4-5*8*3=0 7*2*6+1*4*9-3*5*8=0 7*6*2+1*9*4-5*8*3=0 8*3*4+5*6*1-2*7*9=0 8*4*3+6*1*5-7*9*2=0
6*2*7+9*1*4-8*3*5=0 7*2*6+1*4*9-3*8*5=0 7*6*2+1*9*4-8*3*5=0 8*3*4+5*6*1-2*9*7=0 8*4*3+6*1*5-9*2*7=0
6*2*7+9*1*4-8*5*3=0 7*2*6+1*4*9-5*3*8=0 7*6*2+1*9*4-8*5*3=0 8*3*4+5*6*1-7*2*9=0 8*4*3+6*1*5-9*7*2=0
6*2*7+9*4*1-3*5*8=0 7*2*6+1*4*9-5*8*3=0 7*6*2+4*1*9-3*5*8=0 8*3*4+5*6*1-7*9*2=0 8*4*3+6*1*5-9*7*2=0
6*2*7+9*4*1-3*8*5=0 7*2*6+1*4*9-8*3*5=0 7*6*2+4*1*9-3*8*5=0 8*3*4+6*1*5-2*7*9=0 8*4*3+6*1*5-7*9*2=0
6*2*7+9*4*1-5*3*8=0 7*2*6+1*4*9-8*5*3=0 7*6*2+4*1*9-5*3*8=0 8*3*4+6*1*5-2*9*7=0 8*4*3+6*1*5-9*2*7=0
6*2*7+9*4*1-5*8*3=0 7*2*6+1*9*4-3*5*8=0 7*6*2+4*1*9-5*8*3=0 8*3*4+6*1*5-7*2*9=0 8*4*3+6*1*5-9*7*2=0
6*2*7+9*4*1-8*3*5=0 7*2*6+1*9*4-3*8*5=0 7*6*2+4*1*9-8*3*5=0 8*3*4+6*1*5-7*9*2=0 8*4*3+6*1*5-9*7*2=0
6*2*7+9*4*1-8*5*3=0 7*2*6+1*9*4-5*3*8=0 7*6*2+4*1*9-8*5*3=0 8*3*4+6*5*1-2*7*9=0 8*4*3+6*5*1-9*7*2=0
6*5*1+3*4*8-2*7*9=0 7*2*6+1*9*4-5*8*3=0 7*6*2+4*9*1-3*5*8=0 8*3*4+6*5*1-2*9*7=0 8*4*3+6*5*1-7*9*2=0
6*5*1+3*4*8-7*2*9=0 7*2*6+1*9*4-8*3*5=0 7*6*2+4*9*1-3*8*5=0 8*3*4+6*5*1-7*2*9=0 8*4*3+6*5*1-9*7*2=0
6*5*1+3*4*8-7*9*2=0 7*2*6+1*9*4-8*5*3=0 7*6*2+4*9*1-5*3*8=0 8*3*4+6*5*1-7*9*2=0 8*4*3+6*5*1-9*7*2=0
6*5*1+3*4*8-9*2*7=0 7*2*6+1*9*4-5*3*8=0 7*6*2+4*9*1-8*3*5=0 8*3*4+6*5*1-9*2*7=0 8*4*3+6*5*1-9*7*2=0
6*5*1+3*4*8-9*7*2=0 7*2*6+4*1*9-3*5*8=0 7*6*2+4*9*1-5*3*8=0 8*3*4+6*5*1-9*7*2=0 8*4*3+6*5*1-9*7*2=0
6*5*1+3*8*4-2*7*9=0 7*2*6+4*1*9-3*8*5=0 7*6*2+4*9*1-8*3*5=0 8*3*4+6*5*1-7*9*2=0 8*4*3+6*5*1-9*7*2=0
6*5*1+3*8*4-2*9*7=0 7*2*6+4*1*9-5*3*8=0 7*6*2+4*9*1-8*5*3=0 8*3*4+6*5*1-9*7*2=0 8*4*3+6*5*1-9*7*2=0
6*5*1+3*8*4-7*2*9=0 7*2*6+4*1*9-8*3*5=0 7*6*2+9*1*4-3*5*8=0 8*3*7+1*2*6-4*5*9=0 8*4*3+6*1*2-5*9*4=0
6*5*1+3*8*4-7*9*2=0 7*2*6+4*1*9-8*5*3=0 7*6*2+9*1*4-3*8*5=0 8*3*7+1*2*6-4*9*5=0 8*4*3+6*1*2-5*4*9=0
6*5*1+3*8*4-9*2*7=0 7*2*6+4*9*1-3*5*8=0 7*6*2+9*1*4-5*3*8=0 8*3*7+1*2*6-5*4*9=0 8*4*3+6*1*2-5*9*4=0
6*5*1+3*8*4-9*7*2=0 7*2*6+4*9*1-3*8*5=0 7*6*2+9*1*4-8*3*5=0 8*3*7+1*2*6-5*9*4=0 8*4*3+6*1*2-4*5*9=0
6*5*1+4*3*8-2*7*9=0 7*2*6+4*9*1-5*3*8=0 7*6*2+9*1*4-5*8*3=0 8*3*7+1*2*6-4*5*9=0 8*4*3+6*1*2-4*9*5=0
6*5*1+4*3*8-2*9*7=0 7*2*6+4*9*1-8*3*5=0 7*6*2+9*4*1-3*5*8=0 8*3*7+1*2*6-5*9*4=0 8*4*3+6*1*2-5*4*9=0
6*5*1+4*3*8-7*2*9=0 7*2*6+4*9*1-8*5*3=0 7*6*2+9*4*1-5*3*8=0 8*3*7+1*2*6-4*9*5=0 8*4*3+6*1*2-5*9*4=0
6*5*1+4*3*8-7*9*2=0 7*2*6+4*9*1-5*3*8=0 7*6*2+9*4*1-5*8*3=0 8*3*7+1*6*2-4*5*9=0 8*4*3+6*1*2-9*4*5=0

C.6 Programløsning for generell prototyp

Programmet inkluderer tre ulike varianter av matematisk regnepussel, men kan også utvides til andre varianter med å legge på nye sjablonger og justere programmet.

```
/*
  RFID-TCA9548A-3C
  Programmet leser av ID-koden til flere kort koblet til en multiplekser
  og skriver ID-kodene ut som 9 desimal tall, beregner produket og sum og differanse
  og skriver ut produktene på displayet. Skriver resultatet til punktdisplay
  Nils Kr. Rossing 08.03.24
*/

#include <Wire.h>
#include <SPI.h>
#include <PN532_I2C.h>
#include <PN532.h>
#include <NfcAdapter.h>
#include "TCA9548A.h"
#include "HT16K33.h"

TCA9548A I2CMux0(0x70);          // Adressen 0x70 I2CMux0
TCA9548A I2CMux1(0x71);          // Adressen 0x71 I2CMux1
HT16K33 seg2(0x72);              // Adressen 0x72 display2
HT16K33 seg3(0x73);              // Adressen 0x73 display3
HT16K33 seg4(0x74);              // Adressen 0x74 display4
HT16K33 seg5(0x75);              // Adressen 0x75 display4

#define HARDWARE_TYPE MD_MAX72XX::FC16_HW
#define MAX_DEVICES 4

PN532_I2C pn532i2c(Wire);
PN532 nfc(pn532i2c);

uint32_t versiondata = 0;
uint8_t brightness = 1;
int arrayRFID[10];
int arrayRFIDTall[9][2] = {{600, 1}, {363, 2}, {415, 3}, {555, 4}, {528, 5}, {557, 6}, {424, 7}, {402, 8}, {431, 9}};
int T[9];
#define DEBUG                    // Kommenter bort denne dersom Serial.print ikke skal inkluderes
```

```
void setup() {

  #ifdef DEBUG
    Serial.begin(115200);
  #endif
  Wire.begin();
  Wire.setClock(400000);

  seg2.begin();
  seg3.begin();
  seg4.begin();
  seg5.begin();

  seg2.displayOn();
  seg2.displayColon(0);
  seg3.displayOn();
  seg3.displayColon(0);
  seg4.displayOn();
  seg4.displayColon(0);
  seg5.displayOn();
  seg5.displayColon(0);

  seg2.setBrightness(brightness);
  seg3.setBrightness(brightness);
  seg4.setBrightness(brightness);
  seg5.setBrightness(brightness);

  I2CMux0.begin(Wire);            // Wire instance is passed to the library
  I2CMux1.begin(Wire);            // Wire instance is passed to the library

  for (int i = 0; i < 6; i++) {
    I2CMux0.openChannel(i);
    nfc.begin();
    versiondata = nfc.getFirmwareVersion();
    if (!versiondata) {
      Serial.print("Didn't find PN53x board");
      while (1); // halt
```

```
    }
    #ifdef DEBUG
      Serial.println("Found chip PN5");
    #endif
    nfc.setPassiveActivationRetries(0xFF);
    nfc.SAMConfig();
    I2CMux0.closeChannel(i);
  }
  I2CMux0.closeAll();              // Set a base state which we know (also the default state on power on)

    for (int i = 0; i < 4; i++) {
    I2CMux1.openChannel(i);
    nfc.begin();
    versiondata = nfc.getFirmwareVersion();
    if (!versiondata) {
      Serial.print("Didn't find PN53x board");
      while (1); // halt
    }
    #ifdef DEBUG
      Serial.println("Found chip PN5");
    #endif
    nfc.setPassiveActivationRetries(0xFF);
    nfc.SAMConfig();
    I2CMux1.closeChannel(i);
  }
  I2CMux1.closeAll();              // Set a base state which we know (also the default state on power on)

  Serial.println("RFID-TCA9548A-3A");
}

void loop()
{
  for (int i = 0; i < 6; i++) {
    I2CMux0.openChannel(i);
    readRFID(i);
    I2CMux0.closeChannel(i);
  }
  for (int i = 0; i < 4; i++) {
```

```
  I2CMux1.openChannel(i);
  readRFID(i+6);
  I2CMux1.closeChannel(i);
}
#ifdef DEBUG
  Serial.println("----------------");
#endif

omstokking();

// Realiserer regnestykket: A*BC + D*EF - G*HI = sumDiff
if(arrayRFID[9]==472){
  printDisplayInt(2, produkt1());
  printDisplayInt(3, produkt2());
  printDisplayInt(4, produkt3());
  printDisplayInt(5, sumDiff());
  #ifdef DEBUG
    Serial.print("Totalt = "); Serial.println(sumDiff());
  #endif
}
// Realiserer regnestykket: ABC + DEF - GHI = sumDiffTall
else if(arrayRFID[9]==531){
  printDisplayInt(2, tall_1());
  printDisplayInt(3, tall_2());
  printDisplayInt(4, tall_3());
  printDisplayInt(5, sumDiffTall());
  #ifdef DEBUG
    Serial.print("Totalt = "); Serial.println(sumDiffTall());
  #endif
}

  // Realiserer regnestykket: A*B*C + D*E*F - G*H*I = prodSumDiff
else if(arrayRFID[9]==0){
  printDisplayInt(2, prodABC());
  printDisplayInt(3, prodDEF());
  printDisplayInt(4, prodGHI());
  printDisplayInt(5, prodSumDiff());
  #ifdef DEBUG
```

```
    Serial.print("TotalI = "); Serial.println(prodSumDiff());
   #endif
  }
}


void readRFID(int j) {

  boolean success;
  uint8_t uid[] = { 0, 0, 0, 0 };          // Buffer to store the returned UID
  uint8_t uidLength;                       // Length of the UID (4 or 7 bytes depending on ISO14443A card type)
  long value;

  success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, &uid[0], &uidLength);

  if (success) {
   #ifdef DEBUG
     Serial.print("UID Value: ");
     for (uint8_t i = 0; i < uidLength; i++)
     {
       Serial.print(" ");
       Serial.print(uid[i], DEC);
     }
   #endif
   value = sumFourByteToLong(uid);
   arrayRFID[j] = value;
   #ifdef DEBUG
     Serial.print(", Value "); Serial.print(j); Serial.print(": "); Serial.print(arrayRFID[j]);
     Serial.println("");
   #endif
  }
  else
  {
   // PN532 probably timed out waiting for a card
   #ifdef DEBUG
     Serial.println("Timed out waiting for a card");
   #endif
  }
```

```
}

long sumFourByteToLong(byte UID[4])
{
  long val = 0;
  val += UID[0];
  val += UID[1];
  val += UID[2];
  val += UID[3];
  return val;
}

void omstokking()
{
  for (int i = 0; i < 9; i++)
  {
    for (int j = 0; j < 9; j++)
    {
      if ((arrayRFID[i] == arrayRFIDTall[j][0]) && (arrayRFID[i] != 0)) T[i] = arrayRFIDTall[j][1];
    }
  }
  #ifdef DEBUG
    Serial.print("Omstokking arrayRFID[8]: ");
    Serial.println(arrayRFID[8]);
    Serial.print("Omstokking arrayRFIDTall[8][0]: ");
    Serial.println(arrayRFIDTall[8][0]);
  #endif
}

int produkt1()
{
  // Regnestykket er A * BC = Produkt eller A*(10*B+C) = Produkt
  int Prod1 = (T[2] * (10 * T[3] + T[6]));
  #ifdef DEBUG
    Serial.print("Produkt 1: ");
    Serial.println(Prod1);
  #endif
  return Prod1;
```

```
}

int produkt2()
{
  // Regnestykket er A * BC = Produkt eller A*(10*B+C) = Produkt
  int Prod2 = (T[1] * (10 * T[4] + T[7]));
  #ifdef DEBUG
    Serial.print("Produkt 2: ");
    Serial.println(Prod2);
  #endif
  return Prod2;
}

int produkt3()
{
  // Regnestykket er A * BC = Produkt eller A*(10*B+C) = Produkt
  int Prod3 = (T[0] * (10 * T[5] + T[8]));
  #ifdef DEBUG
    Serial.print("Produkt 3: ");
    Serial.println(Prod3);
  #endif
  return Prod3;
}

int sumDiff()
{
  int sumDiff = produkt1() + produkt2() - produkt3();
  return sumDiff;
}

int tall_1()
{
  // Regnestykket er ABC = tall_1 eller 100*A+10*B+C) = tall_1
  int tall_1 = T[2]*100 + T[3]*10 + T[6];
  return tall_1;
}

int tall_2()
```

```
{
  // Regnestykket er ABC = tall_2 eller 100*A+10*B+C) = tall_2
  int tall_2 = T[1]*100 + T[4]*10 + T[7];
  return tall_2;
}

int tall_3()
{
  // Regnestykket er ABC = tall_3 eller 100*A+10*B+C) = tall_3
  int tall_3 = T[0]*100 + T[5]*10 + T[8];
  return tall_3;
}

int sumDiffTall()
{
  return (tall_1() + tall_2() - tall_3());
}

int prodABC(){
  // Regnestykket er A*B*C = prodABC
  return (T[2] * T[3] + T[6]);
}

int prodDEF(){
  // Regnestykket er D*E*F = prodDEF
  return (T[1] * T[4] + T[7]);
}

int prodGHI(){
  // Regnestykket er G*H*I = prodGHI
  return (T[0] * T[5] + T[8]);
}

int prodSumDiff(){
  // Regnestykket er prod ABC + prodDEF - prodGHI = prodSumDiff
  return (prodABC() + prodDEF() - prodGHI());
}
```

```
void printDisplayInt(int displayNo, int Number)
{
if (displayNo == 2) seg2.displayInt(Number);
delay(10);
if (displayNo == 3) seg3.displayInt(Number);
delay(10);
if (displayNo == 4) seg4.displayInt(-Number);
delay(10);
if (displayNo == 5) seg5.displayInt(Number);
delay(10);
}

void test_printfloat()
{

  for (int i = 0; i < 2000; i++)
  {
    float f = i * 0.001;
    seg2.displayFloat(f);
    seg3.displayFloat(f);
    seg4.displayFloat(f);
    delay(10);
  }
}
```

Noen modeller i utstillingen har behov for å kjenne igjen brikker. Dette har tradisjonelt vært gjort ved å kode brikkene med magneter i unike posisjoner for så å lese av om det befinner seg en magnet eller ikke i den angitte posisjonen som forventet. Dette har vært tilfelle både for "Nærmest null", "Søskenflokken" til Thomas angel og "Ut og fly".

En langt enklere løsning vil være å bruke RFID tagger på brikkene som kan avleses ved hjelp av RFID terminaler. Dette er relativt enkelt å få til dersom man bruker en terminal og mange brikker. Utfordringen blir imidlertid en annen dersom man ønsker å benytte mange like RFID terminaler for å lese av ulike brikker plassert i forskjellige posisjoner. Heftet viser en metode å løse dette problemet på ved hjelp av I^2C-multiplekser.

Siden dette er et ganske generelt problem som flere vitensenter kan ha nytte av er heftet laget som en arbeidsbok som kan brukes i opplæring i bruk av RFID på denne måten.

Nils Kr. Rossing (nkr@vitensenteret.com)
Dosent emeritus i naturfagdidaktikk ved NTNU og tilknyttet Vitensenteret i Trondheim.

ISBN 978-82-92088-84-5
Rev. 3.0 – 11.03.24