

Prosjektoppgave, kalkulator, Fys1210

Halvor Klepsvik

Mai 2022

Sammendrag

I dette prosjektet er målet å lage en 2-bit kalkulator. Kalkulatoren skal ha evnen til å addere 2 tall, samt nullstilles. Målet med oppgaven er å forstå hvordan en enkel krets som kan addere tall vil se ut.

Det brukes en mikrokontrolleren som vil ta input fra en numbpad. Den vil så konvertere dette signalet til en output som vil gjøre det enkelt mulig for oss å koble opp logikken som skal utføre funksjonene vi vil gjennomføre.

Alle bilder hentet fra linker i Delliste

1 Introduksjon

For å interagere med kretsen bruker vi en numbpad. Numpaden kobles til en mikrokontroller som konverterer signalet den mottar til bits. Outputet fra mikrokontrolleren skal sendes til en 'flip-flip' hvor det lagres. Outputet fra 2 'flip-floper' skal sendes videre til en 'adder' hvor de legges sammen. Siste steg blir å sende det til en dekoder, som dekoder outputet slik at det kan vises på et 7-segment display.

Det er fullt mulig å utføre denne oppgaven uten bruk av mikrokontroller, den er brukt for å gjøre oppgaven enklest mulig. Mikrokontrolleren veldig allsidig og fungerer som flere komponenter i kretsen, for eksempel strømforsyning og klokkefrekvens. En ekstra fordel er også at den gjør flere av koblingene mindre komplekse slik at feilsøking blir enklere.

For å utføre oppgaven vi bruker følgende komponenter:

- Metro m4 express(mikrokontroller)
- Pmod Ssd(2 digit seven-segment display)
- Pmod KPYD(16-button Keypad)
- CD74HC4511 (BCD til 7-segment dekoder.)

- CD74HC283 (4-bit full adder)
- 2X SN74HC175 (D-type flip-flop)

Målet med oppgaven er å få oversiktlige koblinger og lage en mest mulig modular krets. Det vil si at den vil være lett å oppgradere i etterkant og lett å feilsøke underveis i konstruksjonen.

For å gjøre oppgaven mest mulig oversiktlig, vil vi gjøre den i følgende steg:

1. Oppkobling og testing av skjerm, dekoder og mikrokontroller
2. Oppkobling av Keypad
3. Oppkobling av logikk

2 Oppkobling og testing av skjerm, dekoder og mikrokontroller

2.1 Teori

I denne delen skal vi koble displayet til mikrokontrolleren, mellom dem må vi ha displayets dekoder. Ved å se på hvilke inputs dekoderen trenger, kan vi bestemme hvilke outputs vi vil gi fra mikrokontrolleren.

For å bestemme oss hvordan vi skal koble mikrokontrolleren til dekoderen, vil vi se på dekoderens layout fra databladet vist i figur 1:

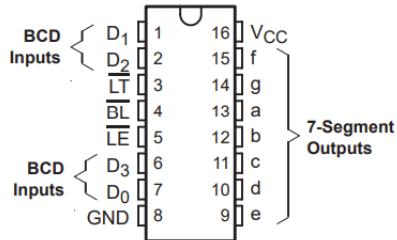


Figure 1: pinout dekoder

Ved å se på figur 1 og tabell 1, som begge er hentet fra dekoderens databladet ser vi at displayet inputs er d0-d3. Vi vil koble dem sammen med 4 koblinger fra mikrokontrollere. Disse 4 koblingene skal representere bits og vil styre displayets output. d0 vil være bakerste bit(least significant), d1 nest bakerste, osv... LE skal alltid være lav ettersom displayet viser blankt om den er høy, dermed kan vi grunde denne. BL og LT skal alltid være satt til høy om vi vil bruke displayet.

Table 1: dekoder tabell

INPUTS				OUTPUTS										
LE	BL	LT	D ₃	D ₂	D ₁	D ₀	a	b	c	d	e	f	g	DISPLAY
X	X	L	X	X	X	X	H	H	H	H	H	H	H	8
X	L	H	X	X	X	X	L	L	L	L	L	L	L	Blank
L	H	H	L	L	L	L	H	H	H	H	H	H	L	0
L	H	H	L	L	L	H	L	H	H	L	L	L	L	1
L	H	H	L	L	H	L	H	H	L	H	H	L	H	2
L	H	H	L	L	H	H	H	H	H	L	L	H	H	3
L	H	H	L	H	L	L	L	H	H	L	L	H	H	4
L	H	H	L	H	L	H	H	L	H	H	L	H	H	5
L	H	H	L	H	H	L	L	L	H	H	H	H	H	6
L	H	H	L	H	H	H	H	H	H	L	L	L	L	7
L	H	H	H	L	L	L	H	H	H	H	H	H	H	8
L	H	H	H	L	L	H	H	H	L	L	H	H	H	9

Videre må vi koble dekoderen til displayet og sørge for at den viser de riktige tallene. Pinouten til displayet vises i figur 2:

Pinout Description Table

Header J1			Header J2		
Pin	Signal	Description	Pin	Signal	Description
1	AA	Segment A	1	AE	Segment E
2	AB	Segment B	2	AF	Segment F
3	AC	Segment C	3	AG	Segment G
4	AD	Segment D	4	C	Digit Selection pin
5	GND	Power Supply Ground	5	GND	Power Supply Ground
6	VCC	Positive Power Supply	6	VCC	Positive Power Supply

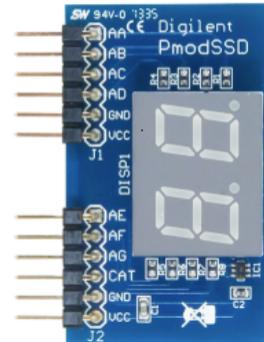


Figure 2: Display pinout med tabell

Som vi ser fra Figur 1 er dekoderens output delt inn i bokstaver fra A-G. I figur 2 ser vi at displayets input har tilsvarende bokstaver og kan kobles sammen, Vcc'ene på displayet kan kobles til Vcc'en på dekoderen, GND kan kobles til ground sammen med pin 4.

2.2 Metode

Første vi må gjøre er å bestemme hvilke outputs fra mikrokontrolleren vi vil bruke. Mikrokontrollerens layout er vist under i Figur 3.

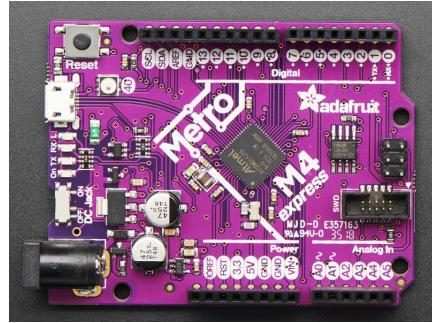


Figure 3: Layout mikrokontroller

Vi velger å bruke utgangene A0-A3(høgre hjørne på figur 3) for å representer tallene vi vil sende som bits, utgangene A4 og A5 vil vi bruke som spenningskilder for å kjøre displayet.
A4 og A5 vil alltid stå som høy slik at de konstant tilfører strøm til displayet, mens A0-A3 vil representer tall i bitsformat. Det er også viktig at alle komponenter jordes.

Da ender vi med en kobling som ser ut slik som i figur 4:

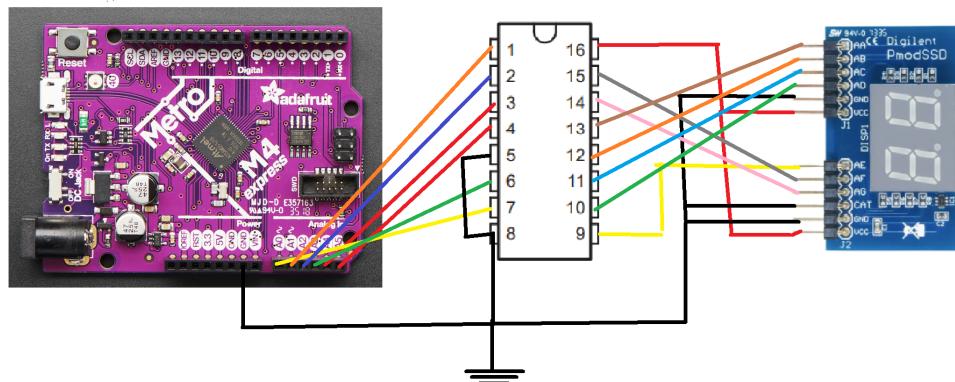


Figure 4: Oppkobling Mellom dekoder, skjerm og mikrokontroller

Etter koblingene er det neste steget å teste. For denne testingen bruker vi koden under som utgangspunkt og plugget inn verdiene vi ville i A0-A3 slik at mikrokontrolleren gir ut de ønskede tallene i bits.

```

void setup() {
    // // put your setup code here, to run once:
    // Set pin A0-A5 on the board to output pins
    pinMode(A0, OUTPUT);
    pinMode(A1, OUTPUT);
    pinMode(A2, OUTPUT);
    pinMode(A3, OUTPUT);
    pinMode(A4, OUTPUT);
    pinMode(A5, OUTPUT);

    void loop() {
        // Set pin output for pin A0-A5 as HIGH or LOW
        digitalWrite(A0, X); // Replace X with HIGH or LOW
        digitalWrite(A1, X); // Replace X with HIGH or LOW
        digitalWrite(A2, X); // Replace X with HIGH or LOW
        digitalWrite(A3, X); // Replace X with HIGH or LOW

        digitalWrite(A4, HIGH);
        digitalWrite(A5, HIGH);
    }
}

```

2.3 Resultater

Etter alt var koblet opp på riktig måte, testet vi ved å vise verdiene 0-6 på displayet. Alt fungerte som forventet.

3 Oppkobling av Keypad

3.1 Teori

For å koble opp numbpadden må vi se på numbpadens krets(figur 6) og pinout(figur 5).

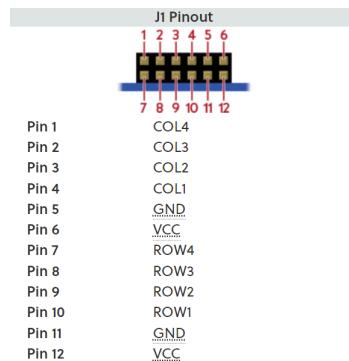


Figure 5: Pinout numypad

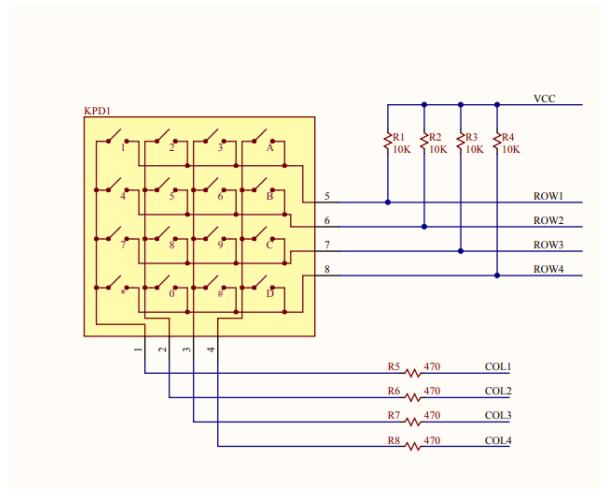


Figure 6: Krets numbpad

Ved å se på pinouten i figur 5 og kretsen i figur 6, kan vi resonere at å trykke på tast '1' vil gi et utslag på Kolonne 1(Pin 4) og rad 1(pin 10), tast '2' vil gi utslag på kolonne 2(pin 3) og rad 1(pin 10), tast '4' vil gi utslag på kolonne 1(pin 4) og rad 2(pin 9), osv...

3.2 Metode

Vi vil koble opp de 2 første radene og de 3 første kolonnene slik at vi kan få alle funksjonene vi trenger for kalkulatoren. Oppkoblingen vises i figur 7.

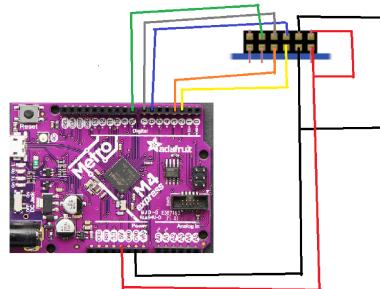


Figure 7: Numpad kobling til arduino

Etter oppkoblingen er neste steg å legge inn koden og teste. Vi bruker 'Keypad.h' biblioteket fra python og bruker eksempelkoden under som utgangspunkt og hvor vi legger inn verdiene vi ville ha for hver av tastene:

```
byte row_pins[] = {2, 3, 4, 5}; //row pins of the keypad
```

```

byte column_pins[] = {6, 7, 8, 9}; //column pins of the keypad

//Declaration of the keys of the keypad
char hexaKeys[sizeof(row_pins) / sizeof(byte)][sizeof(column_pins) /
    sizeof(byte)] =
{
    {'1', '2', '3', 'A'},
    {'4', '5', '6', 'B'},
    {'7', '8', '9', 'C'},
    {'0', 'F', 'E', 'D'}
};

//define object for the keypad
Keypad kypd = Keypad( makeKeymap(hexaKeys), row_pins, column_pins,
    sizeof(row_pins) / sizeof(byte), sizeof(column_pins) /
    sizeof(byte));

// keep last pressed
int numb = 0;

void setup() {
    // put your setup code here, to run once:

    Serial.begin(9600);
    pinMode(A0, OUTPUT); // D0
    pinMode(A1, OUTPUT); // D1
    pinMode(A2, OUTPUT); // D2
    pinMode(A3, OUTPUT); // D3
    pinMode(A4, OUTPUT); // D4
    pinMode(A5, OUTPUT); // D5

    digitalWrite(A4, HIGH); // D4
    digitalWrite(A5, HIGH); // D5
}

void loop() {
    // put your main code here, to run repeatedly:

    //get keypad state
    char current_key = kypd.getKey();

    // writes sign to serial monitor (for debugging)
    Serial.write(current_key);

    // checks last pressed and enables the given pins
    if (current_key == '1') {
        digitalWrite(A3, X);
        digitalWrite(A2, X);
        digitalWrite(A1, X);
        digitalWrite(A0, X);
    }
}

```

```

    }
else if (current_key == '2') {
    digitalWrite(A3, X);
    digitalWrite(A2, X);
    digitalWrite(A1, X);
    digitalWrite(A0, X);
}
else if (current_key == '3') {
    digitalWrite(A3, X);
    digitalWrite(A2, X);
    digitalWrite(A1, X);
    digitalWrite(A0, X);
}
else if (current_key == '4') {
    digitalWrite(A3, X);
    digitalWrite(A2, X);
    digitalWrite(A1, X);
    digitalWrite(A0, X);
}
else if (current_key == '5') {
    digitalWrite(A3, X);
    digitalWrite(A2, X);
    digitalWrite(A1, X);
    digitalWrite(A0, X);
}
else if (current_key == '6') {
    digitalWrite(A3, X);
    digitalWrite(A2, X);
    digitalWrite(A1, X);
    digitalWrite(A0, X);
}
}

}

```

3.3 Resultater

Ved å taste inn verdiene på numbspaden, fikk vi opp tilsvarende verdi på displayet. Nå har vi et display hvor vi kan endre outputet ved å trykke på tastene. For å lage kalkulatoren mangler vi nå bare å legge inn funksjonene via logikken.

4 Oppkoppling av logikk

4.1 Teori

For å få kalkulatoren til å kunne legge sammen 2 tall, vil vi ha logikk å lagre tallene på, vi vil også ha logikk som kan addere sammen tallene.

Vi bruker 2 flip-floper til å lagre tallene, her bruker vi SNx4HC175. Her kreves det et resetsignal, en klokkefrekvens og en input av tallet. Selve kretsen til flip-flopen vil se ut som den i figur 8.

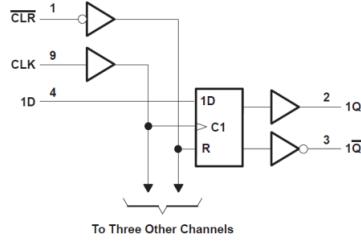


Figure 8: Pinout flip-flop

Så lenge resetten(CLR) er satt til høy vil tallene som skrives inn lagres, om den endres til lav resetter chippen seg. Klokkefrekvensen(CLK) vil gjøre det mulig å overskrive tallene, om den stopper blir siste tallet som er skrevet inn lagret. Inputet(1D) kan enten være lav eller høy, med 4 slike inputs på chippen, kan vi representer et 4-bit tall.I vårt tilfelle skal vi bare bruke 2 av inputsene for å kunne representere et 2-bit tall.

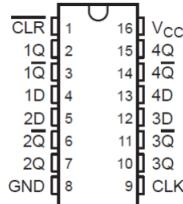


Figure 9: Pinout flip-flop

I Figur 9 vises pinouten på flip-flopen. Den har som sagt 4 inputs, vi vil bare bruke 2 av dem, slik kan vi lagre et 2-bit tall.

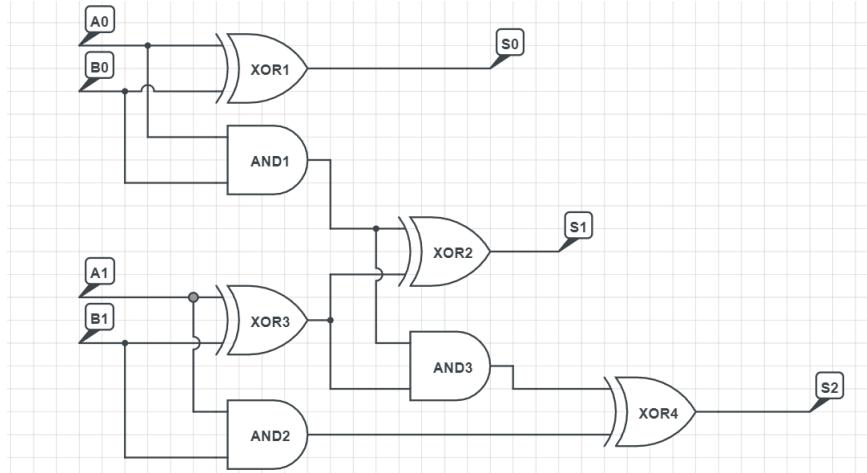


Figure 10: 2-bit adder med 3-bit output

til adderingen velger vi å bruke en CD74HC283. Det er en 4-bit adder, men kan selvfølgelig også brukes til å addere 2-bit tall også. En adder bruker logikk-gater for å legge sammen 2 tall, en xor-gate for å bestemme en bits verdi og en and-gate til å bestemme om tallet skal 'bæres over' til neste bit. Figur 10 viser eksempel på hvordan en krets for en 2-bit adder med 3-bit output kan se ut tegnet i circuitlab.

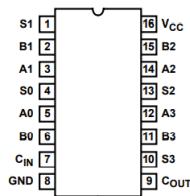


Figure 11: Pinout 4-bit adder

Figur 11 viser pinouten til adderen hvor a0-a3 og b0-b3 representerer inngangene, men s0-s3 representerer utgangene.

4.2 Metode

Fra keypaden bruker vi knapp 1-3 for å representere de tilsvarende tallene, knapp 4 til '0', knapp 5 til '+' og knapp 6 til 'clear/ce'.

Vi vil mappe et output fra clear-knappen(6) til en not-gate, denne vil gå videre til 'clear' på begge flip-flopene. Å skru dette outputtet på vil da slette de lagrede bitsene og resette kalkulatoren. Dette kan du gjøre i programvare med

mikrokontrolleren uten en not-gate, men for eventuelle oppgraderinger eller endringer er det bedre å gjøre det via maskinvaren.

For å kunne lagre tall må vi også kunne kjøre klokkefrekvensen på bare en av flip-flopene om gangen. Derfor vil vi mappe en output fra '+'-knappen(5) til en and- og en nand-gate sammen med klokkepulsen. Endring i outputtet fra '+'-knappen vil da endre hvilken flip-flop som skrives til.

Oppkoblingen til numbpaden og displayet blir den samme. Resten av logikken kan vi koble opp slik som i figur 12:

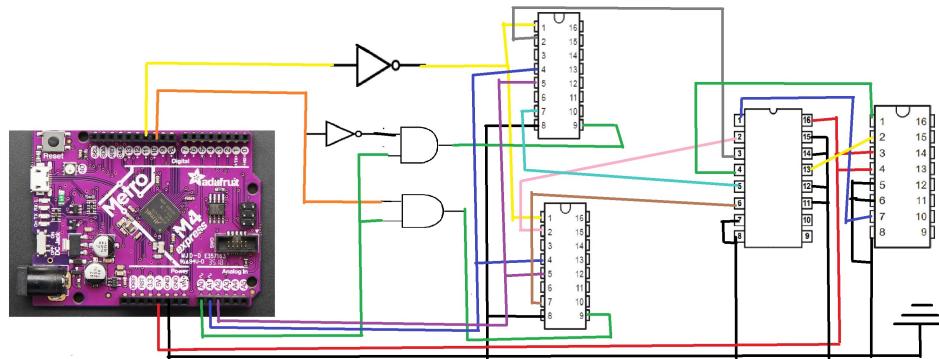


Figure 12: Oppkobling av logikk

Koden under er eksempelkoden vi brukte for å kjøre programmet på mikrokontrolleren.

```

byte row_pins[] = {2, 3, 4, 5}; //row pins of the keypad
byte column_pins[] = {6, 7, 8, 9}; //column pins of the keypad

//Declaration of the keys of the keypad
char hexaKeys[sizeof(row_pins) / sizeof(byte)][sizeof(column_pins) /
    sizeof(byte)] =
{
    {'1', '2', '3', 'A'},
    {'4', '5', '6', 'B'},
    {'7', '8', '9', 'C'},
    {'0', 'F', 'E', 'D'}
};

Keypad kypd = Keypad( makeKeymap(hexaKeys), row_pins, column_pins,
    sizeof(row_pins) / sizeof(byte), sizeof(column_pins) /
    sizeof(byte)); //define object for the keypad

int numb = 0; // used to keep last pressed

```

```

unsigned long prev_mill = 0;
int millper = 100;
int AL = 0;

void setup()
{
    Serial.begin(9600);
    pinMode(A0, OUTPUT); // CLK
    pinMode(A1, OUTPUT); // LSB
    pinMode(A2, OUTPUT); // MSB

    pinMode(10, OUTPUT); // + sign
    pinMode(11, OUTPUT); // CE
}

void loop()
{

    unsigned long cmill = millis(); // saves the time
    char current_key = kypd.getKey(); //get keypad state
    Serial.write(current_key); // writes sign to serial monitor (for
        debugging)

    // checks if the time has gone one designated period and generates a
        CLK pulse
    if (cmill-prev_mill >= millper) {
        prev_mill = cmill;
        if (AL == 1) {
            digitalWrite(A0, HIGH);
            AL = 0;
        }
        else {
            digitalWrite(A0, LOW);
            AL = 1;
        }
    }

    // sets a value to keep last pressed
    if (current_key == '1') {
        numb = 1;
    }
    if (current_key == '2') {
        numb = 2;
    }
    if (current_key == '3') {
        numb = 3;
    }
    if (current_key == '4') {
        numb = 4;
    }
}

```

```

if (current_key == '5') {
    numb = 5;
}
if (current_key == '6') {
    numb = 6;
}

// checks last pressed and enables the given pins
if (numb == 4) {
    digitalWrite(A1, LOW);
    digitalWrite(A2, LOW);
}
else if (numb == 1) {
    digitalWrite(A1, HIGH);
    digitalWrite(A2, LOW);
}
else if (numb == 2) {
    digitalWrite(A1, LOW);
    digitalWrite(A2, HIGH);
}
else if (numb == 3) {
    digitalWrite(A1, HIGH);
    digitalWrite(A2, HIGH);
}
else if (numb == 5) {
    digitalWrite(10, HIGH);
    delay(20);
    digitalWrite(A1, LOW);
    digitalWrite(A2, LOW);
}
else if (numb == 6) {
    digitalWrite(11, HIGH);
    digitalWrite(A1, LOW);
    digitalWrite(A2, LOW);
    digitalWrite(10, LOW);
    delay(20);
    digitalWrite(11, LOW);
}
}

```

4.3 Resultat

i figur 13 vises den ferdige kalkulatoren som kan legge sammen 2 2-bits tall og resettes.

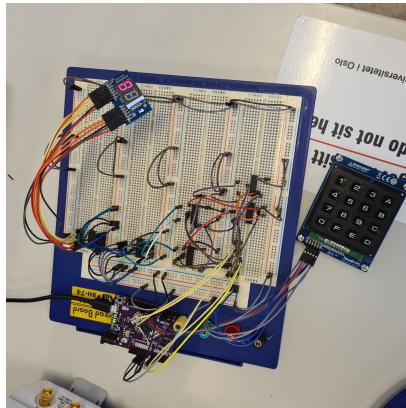


Figure 13: Ferdig kalkulator

5 Diskusjon

Det er selvfølgelig mange oppgraderingen som kan gjøres med denne kalkulatoren. Kompakthet var ikke en prioritet i prosjektet, men du ser åpenbart i figur 12 at mindre brødbrett og mindre mellomrom mellom ic-ene hadde fått den betydelig mer kompakt.

En annen naturlig oppgraderingene vil være å bytte ut mikrokontrolleren med en klokkefrekvens og en strømforsyning. Dette er en naturlig oppgradering ettersom det da vil bli en fullt analogt oppkoblet kalkulator som ikke er avhengig av programvare. Inputsene fra numbaden kunne da gått gjennom andgates istedenfor mikrokontrolleren. For eksempel Row1 og Col1 fra keypaden koblet i en andgate, ville gitt utslag om tasten '1' fra keypaden ble trykket på.

6 Konklusjon

Prosjektet handlet om å lage en mest mulig simpel og oversiktlig kalkulator som kan addere sammen 2 2-bit tall. Det er derfor ikke veldig avansert men gir likevel en god forståelse av bruk av ic-er og andre elektroniske komponenter. Resultatet vi endte opp med var en stor men oversiktig krets som gir et innblikk i hvordan du kan bruke logikk til å gjennomføre enkle regneoperasjoner.

Kildeliste

- lady ada, 12/11/2021, Adafruit Metro M4 Express featuring ATSAMD51 Overview, learn.adafruit.com
- Apperson, Gene, 20/11/2011, Digilent Pmod™ Interface Specification, digilent.com
- Texas Instruments, 11/2003, BDC-TO-7 SEGMENT LATCH/DECODER/-DRIVERS, ti.com
- Texas Instruments, 10/2003, High-Speed CMOS Logic 4-Bit Binary Full Adder with Fast Carry, ti.com
- Texas Instruments, 02/2022, SNx4HC175 Quadruple D-Type Flip-Flops With Clear, ti.com