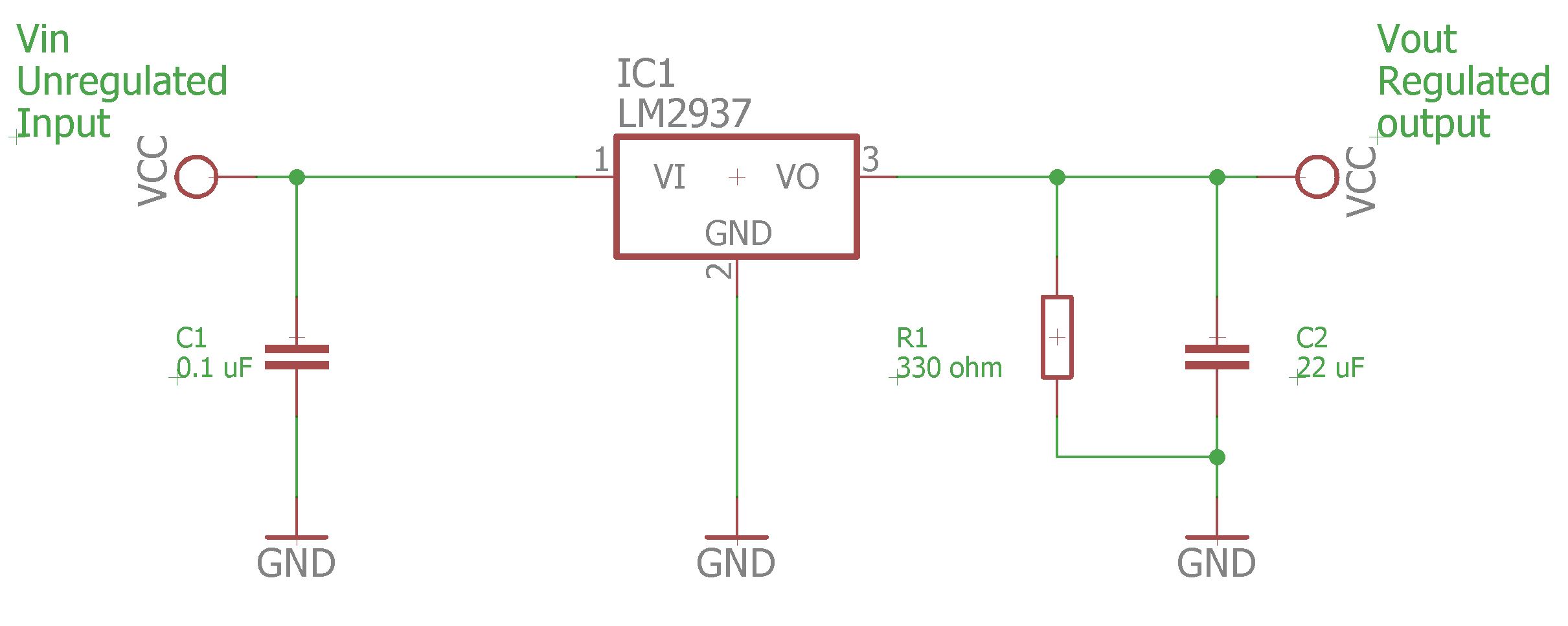
Intro

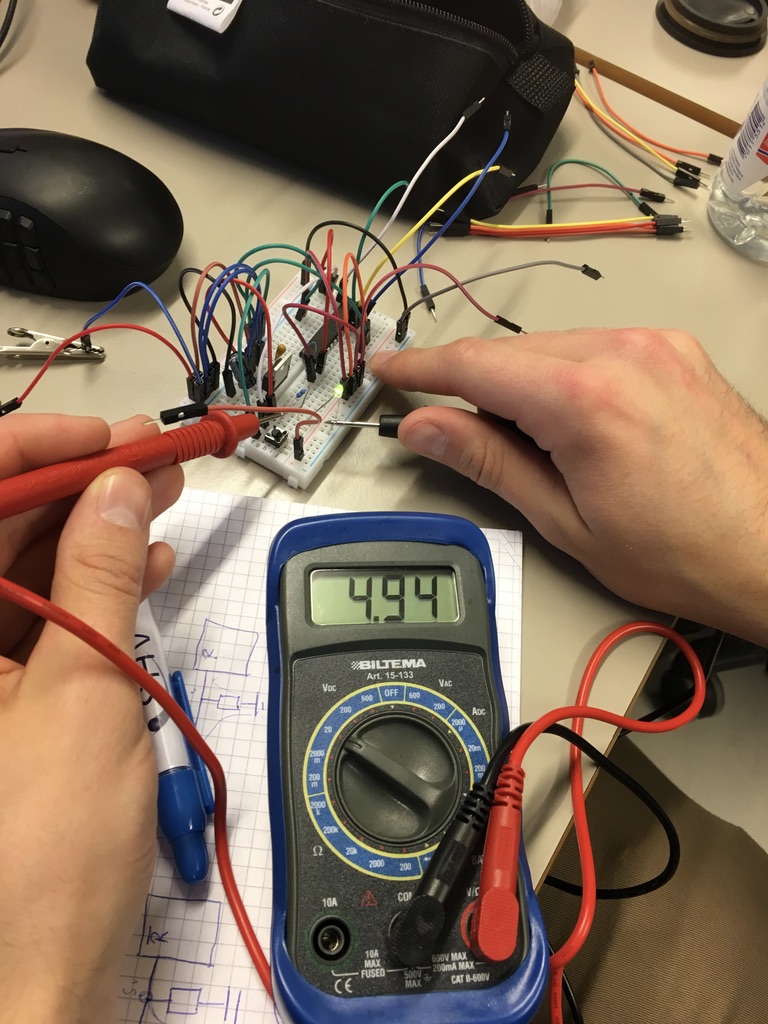
* Skrive til slutt. Hva har vi gjort i dette prosjektet?

**Spenningsregulator**

Vi startet prosjektet med å laste ned to programmer, henholdsvis Autodesk Eagle og Atmel Studio, for tegning av kretser og programmering av mikrokontrolleren. De første timene gikk ut på å sette oss inn i hvordan Eagle fungerer og de forskjellige funksjonene vi måtte bruke for å kunne tegne opp kretsen vår.

Vår første utfordring var å lage en strømforsyning som vi skulle bruke videre i prosjektet. Dette gjorde vi for å kunne forsyne resten av kretskortet våres med en helt stabil spenning. Vi valgte oss en 5V lineær spenningsregulator, LM2931T, som utgangspunkt og fant videre forklaring på oppkobling i databladet til denne.

For å kontrollere at denne fungerte optimalt, ønsket vi å måle spenningen ut med et multimeter. Det står da i databladet til spenningsregulatoren at den ikke vil levere en stabil spenning uten en minimumslast på 5mA, samtidig som den maks tåler en last på 100mA. Vi sjekket derfor etter tilgjengelige resistanser som vi kunne bruke som last, og regnet ut at dette gav en last som var: 5mA < X < 100mA.

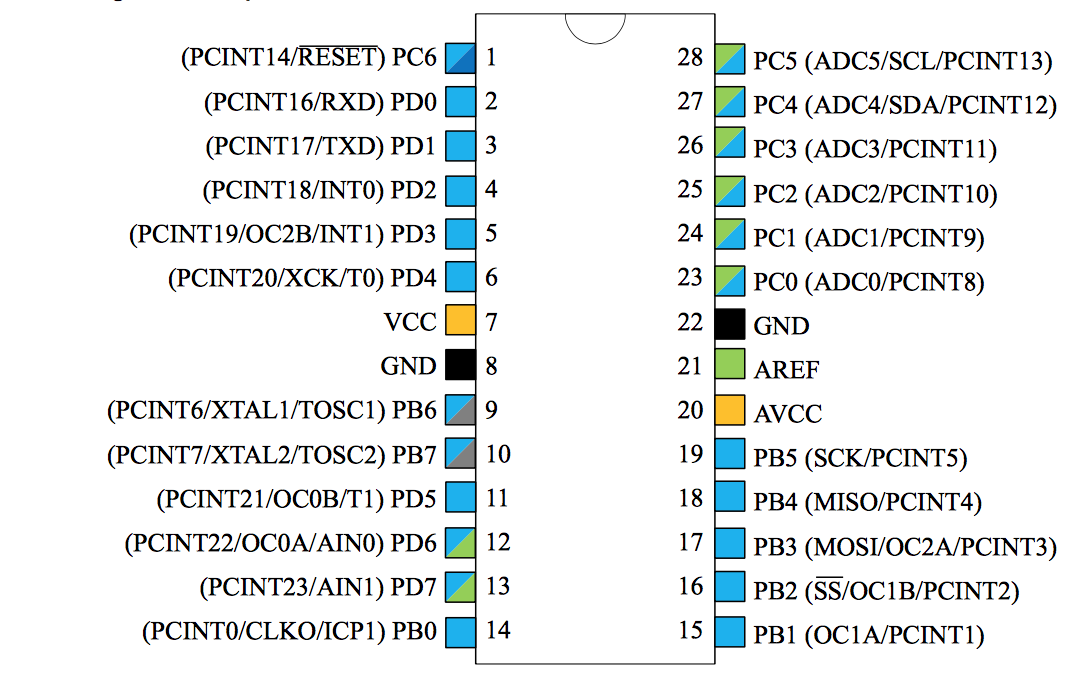
Databladet viste også at spenningen inn på regulatoren måtte være 6V ≤ X ≤ 26V. For å simulere enn ustabil spenning inn på regulatoren, regulerte vi spenningen fra 6-15V med strømforsyningen og målte med multimeteret at vi fikk stabilt 4,94V hele veien.

Vi tror at grunnen til at vi ikke får nøyaktig 5V ut av regulatoren, er fordi vi ble nødt til å erstatte 100uF-kondensatoren fra databladet med 22uF som var tilgjengelig.

**Tegning og oppkobling av mikrokontrolleren**

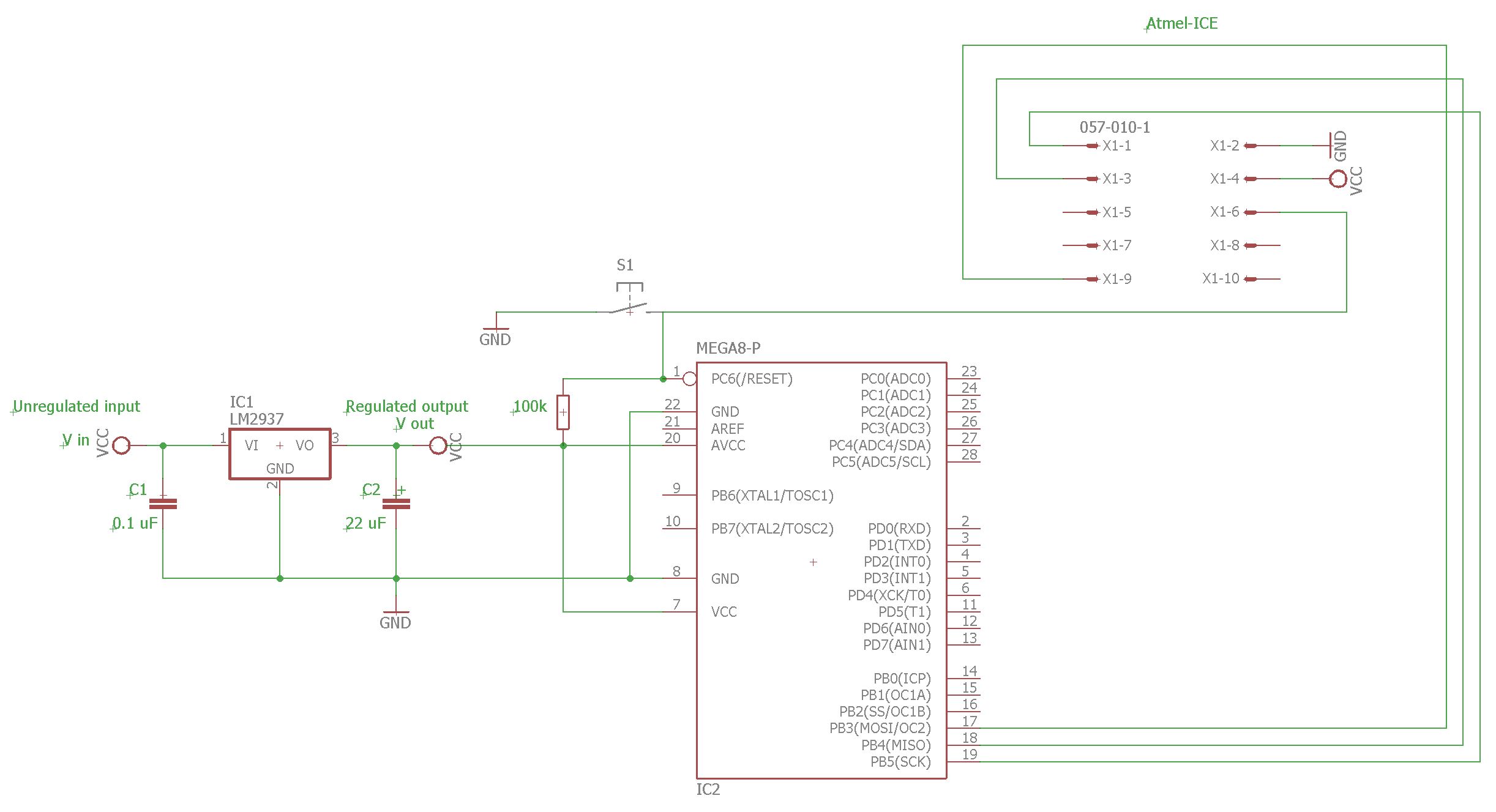
Etter å ha kontrollert at spenningsregulatoren fungerte tilfredsstillende, skulle vi tegne og koble opp mikrokontrolleren vår; en Atmega 168.

Vi søkte opp databladet og fant en oversikt over pin-konfigurasjonen som vi brukte som utgangspunkt for tegningen.

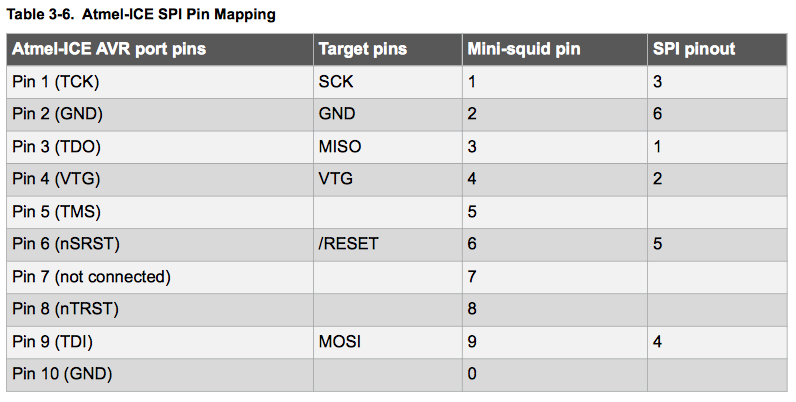


For å kunne resette mikrokontrolleren, valgte vi å lage en ”pull-up” reset. Denne ligger med konstant 5V inn på en invertert reset-pin. Mikrokontrolleren vil altså resette seg først når spenningen går til 0V. Dette skjer da resetbryteren blir trykket og lager en kobling til jord. En motstand på 100k ble brukt mellom Vcc og reset for å hindre støy, samt redusere strømmen gjennom bryteren dersom denne ble trykt.

(Tegne om strømskjema under til å ha kondensatorer mellom Vin og ground på mikrokontrolleren, og forklare hvorfor dette er viktig)



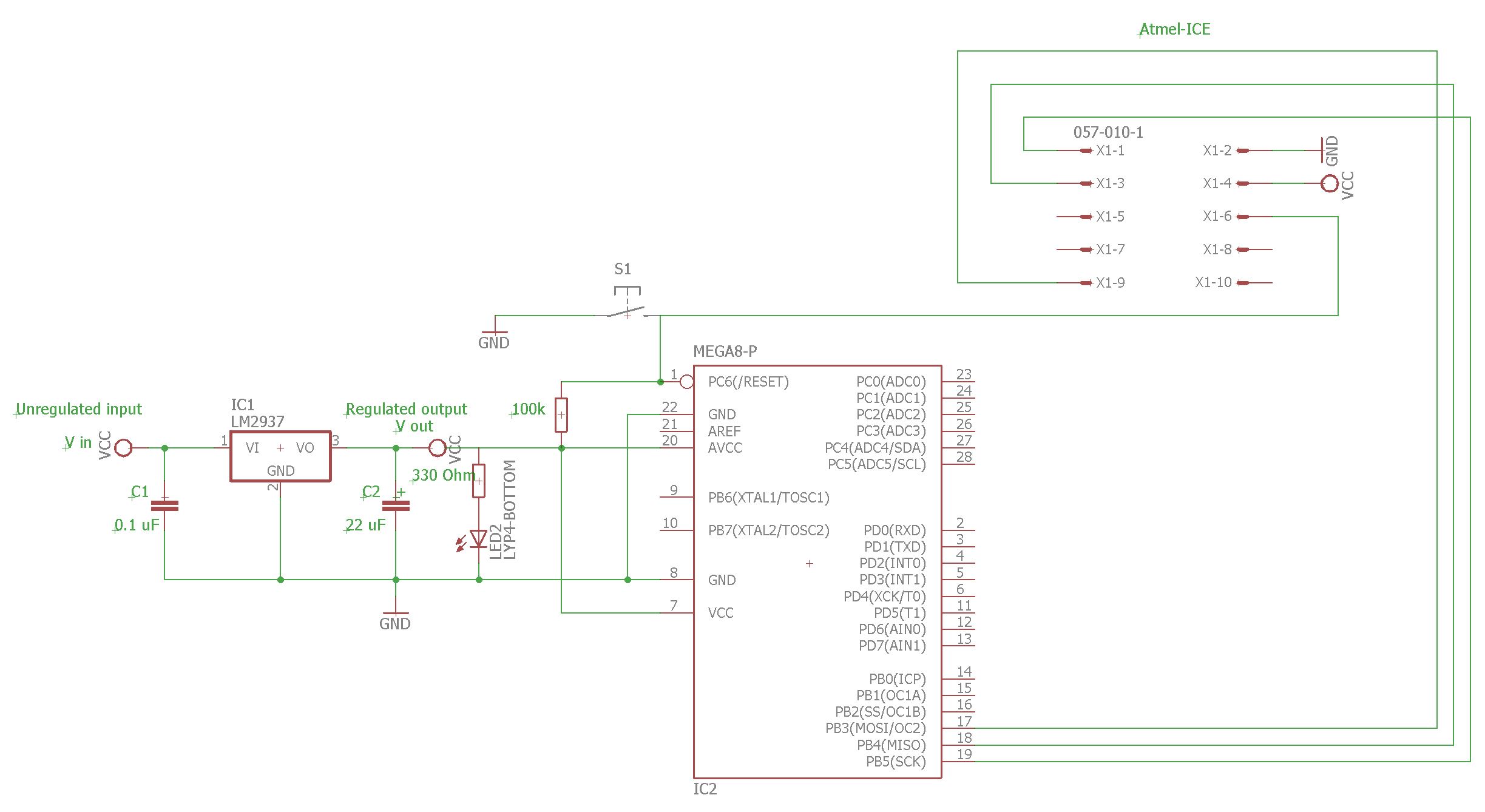
For å kunne programmere mikrokontrolleren, brukte vi en Atmel-ICE som overgang fra pcen. Vi fant tabell for oppkoblingen i databladet til denne, under avsnittet for oppkobling til SPI-target.



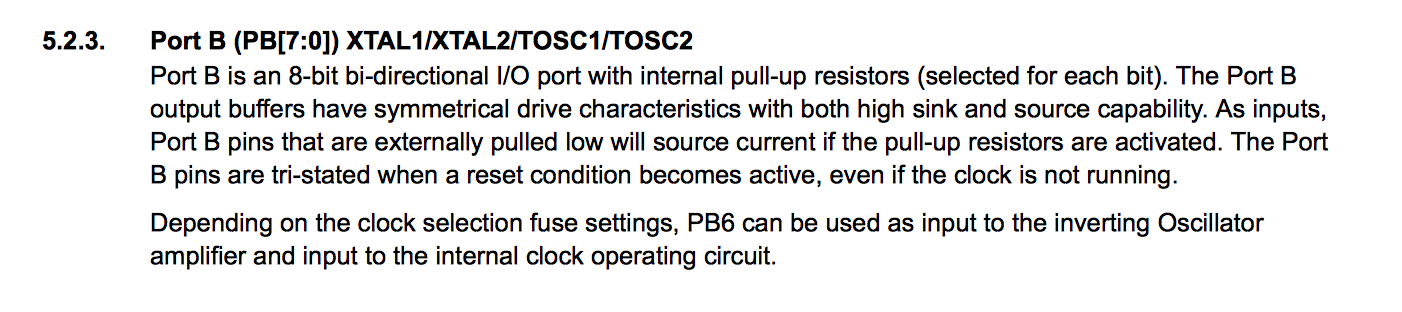
**Atmel Studio og LED-diode**

Startet Atmel Studio for første gang og gjorde oss kjent med programmet. Lærte oss hvordan vi kunne sette de forskjellige pinnene på mikrokontrolleren til å være innganger/utganger, samt skrive en verdi til de. Vi kontrollerte at kompileringen gikk greit.

I neste del av oppgaven koblet vi opp en led mellom utgangen på spenningsregulatoren og jord. Denne skulle lyse for å indikere at vi hadde ”power on”. Vi leste oss opp i databladet til led-en vi brukte, og fant ut at den tålte en maksimal strøm på 30? mA. For å sikre at det ikke gikk mer strøm igjennom dioden, koblet vi inn en motstand på 330 i serie med denne.



Oppgv7,3)

Vi benytter Port B, pin 14. Denne kan være både input og output, og har en intern pull-up resistans.

Dersom vi ikke aktiverer den interne pull-up funksjonen, må vi fysisk koble opp en på brettet. Hvis ikke, vil vi kun lage en connection mellom input og jord.

Tegne inn 2 kondensatorer på 100 nano og begrunne hvorfor med bildene under. Funnet i det generelle databladet til atmega.

