INF1510: Bruksorientert design

Ukeoppgaver i Arduino - uke 2
Vår 2017

Innhold

1. Analoge signaler

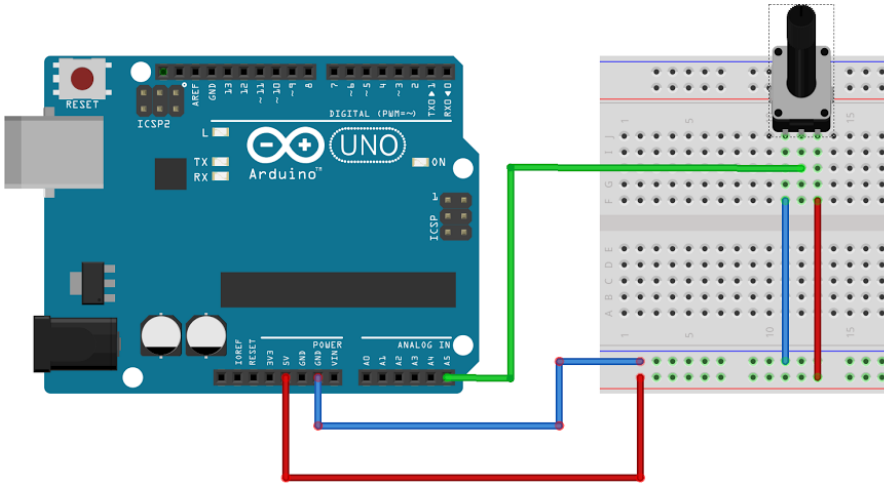
1.1. Lese og skrive analoge signaler

Når vi jobber med digitale systemer forholder vi oss som regel til to tilstander. Av eller på (strøm eller ikke strøm). For eksempel: er knappen trykket ned eller ikke? Dette kalles gjerne digitale signaler. I den fysiske verden er det derimot mer enn bare av eller på. Temperatur er for eksempel ikke bare "av" eller "på". Selv om Arduino er et digitalt system er det også mulig å lese og skrive analoge signaler ved hjelp av Arduinoens innebygde ADC (analog-to-digital converter).

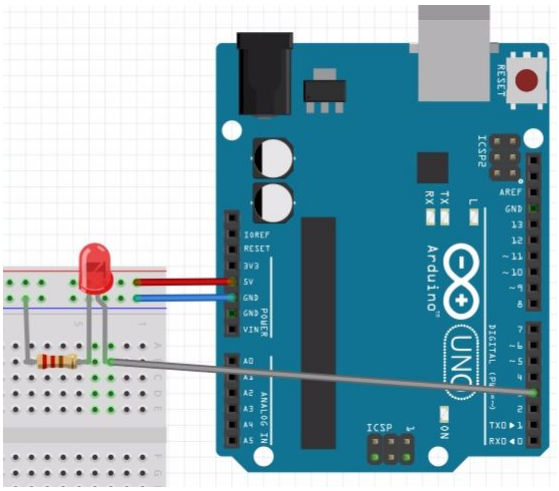
1. Fra hvilke porter kan du lese analoge signaler og hvordan gjør du dette?
2. Fra hvilke porter kan du sende ut analoge signaler og hvordan gjør du dette?

1.2. Potensiometer og serial monitor

I foilene er det et eksempel på hvordan du kan lese verdier fra et potensiometer for så å sende disse videre til serial monitor. Dette kan være lurt å gjøre hvis en for eksempel ønsker å teste om potensiometeret fungerer som normalt. Finn et potensiometer i Arduino-settet og koble opp kretsen som vist nedenfor og skriv kode som printer verdiene til serial monitor.
Tips: legg inn et kort delay slik at du kan se mer tydelig hvordan verdiene forandrer seg.



1.3. Pulserende lys



Koble en LED til en av portene til Arduinoen som lar deg sende analoge signaler slik som i figuren til høyre.

1. Bruk funksjonen analogWrite() til å sette lysstyrken til LED-pæren ved å sende med forskjellige verdier mellom 0 og 255.
2. Skriv et program som får LED-pæren til å "pulsere". Med andre ord, få LED-pæren til å fade inn og ut i en jevn rytme. Prøv med forskjellige delay-tider for å finne en passe hastighet på pulseringen.
3. Prøv til slutt å skrive et program som lar deg kontrollere lysstyrken med et potentiometer (Her er kretsen fra oppgaven om potensiometer nyttig). Her må du bruke både analogRead() for å lese fra potentiometeret og analogWrite() til å sette lysstyrken.
4. Les beskrivelsene av komponenter *Arduino Projects Book* side 6 - 9. Kommer du på flere måter å kontrollere lysstyrken på? For eksempel ved å bruke noen av komponentene i startsettet?
5. Er det mulig å lage en krets som lar et potensiometer styre lysstyrken til en LED, uten å bruke Arduino som mer enn en strømkilde?

1.4. RGB LED

I Arduinosettet følger det med en såkalt RGB LED. Denne er litt annerledes enn de andre.

1. Hva er spesielt med denne og hvorfor har den fire "føtter" i stedet for to?
2. Finn tre porter på Arduinoen som støtter analoge signaler (pwm), for eksempel 9-11, og koble hver av disse portene til hver sin fot på RGB-en. Om du trenger litt hjelp til å koble opp kretsen kan det lønne seg å se på oppgave 4 i Arduinoboka. Husk også å bruke resistorer (viktig!) og koble den fjerde foten til jord.
3. Når du har koblet opp kretsen: Prøv å skrive enkel testkode for å finne ut om RGB-en fungerer ordentlig. Prøv å sende forskjellige verdier mellom 0 og 255 til de forskjellige føttene med analogWrite() å se at RGB-en endrer farge.
4. Det følger også med tre potensiometere i Arduino-settet. Bruk disse til å kontrollere fargen på RGB-en der hvert potensiometer styrer hver farge.

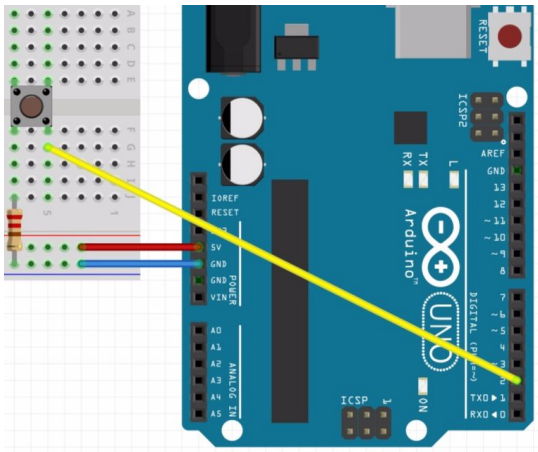
Hint: Verdiene du får inn ved å bruke analogRead() er mellom 0 og 1023, men største verdi du kan sende ut med analogWrite() er 255. Du må derfor skalere verdiene du får fra analogRead() slik at de er innenfor rekkevidden 0 til 255.
Dette kan gjøres på flere måter, en av disse er funksjonen map() som du kan lese om her https://www.arduino.cc/en/Reference/Map

1.5. Konvertering av signaler

1. Kan en konvertere analoge signaler til digitale? Forklar eventuelt hvordan, tegn en slik krets, **eller** skriv en slik funksjon i arduino-språket.
2. Kan en konvertere digitale signaler til analoge? Forklar eventuelt hvordan, tegn en slik krets, **eller** skriv en slik funksjon i arduino-språket.

2. Registrering av knappetrykk

2.1. Upålitelig knappetrykk



```
void setup() {
  pinMode(2, INPUT);
  Serial.begin(9600);
}

void loop() {
  Serial.println(digitalRead(2));
  delay(200);
}
```

1. Hva gjør denne kombinasjon av krets og kode for upålitelig til å informere oss om knappen er trykket ned eller ikke?
2. Endre koden slik at systemet kan lese (på en pålitelig måte) om knappen blir trykket ned.

Hint: Her kan det være veldig nyttig å lese www.arduino.cc/en/Tutorial/InputPullupSerial.
For en litt bedre forklaring kan du se denne videoen: http://youtu.be/wxjerCHCEMg.

3. Endre kretsen slik at systemet kan lese (på en pålitelig måte) om knappen blir trykket ned, men bruk koden fra bildet i oppgaven.

2.2. Knappetrykk og serial monitor

Koble opp en knapp til Arduinoen slik at du kan lytte på den. Skriv et program som skriver "Trykket!" til serial monitor når knappen trykkes. Når programmet kjører, åpne serial monitor på PC-en.

1. Hva skjer i serial monitor når du holder knappen inne?
2. Kan du forklare hvorfor dette skjer?

2.3. Tid og serial

Funksjonen millis() returnerer antall millisekunder som har gått siden Arduinoen startet. Dette kan være nyttig for en rekke bruksområder. Det krever derimot litt ekstra planlegging og at du holder tunga rett i munn. Det kan derfor være lurt å øve seg litt på hvordan millis() fungerer.

1. Skriv et program som skriver "Det har gått fem sekunder!" til serial monitor hvert femte sekund ved å bruke millis().

2.4. Knapp og blink

Koble til en LED til Arduinoen og programmer LED-pæren slik at den blinker med ett sekunds mellomrom ved hjelp av delay(). Koble deretter til en knapp og skriv "Trykket" til serial monitor hver gang knappen blir trykket.

1. Trykk på knappen mange ganger etter hverandre og følg med i serial monitor. Hvorfor blir bare et fåtall av knappetrykkene registrert?
2. Tilpass koden: Løs oppgaven ved bruk av millis() i stedet for delay().
3. Ut i fra observasjonene fra oppgavene ovenfor: I hvilke situasjoner må man bruke millis() i stedet for delay()?
4. Burde man alltid bruke millis()?

2.5. Debounce

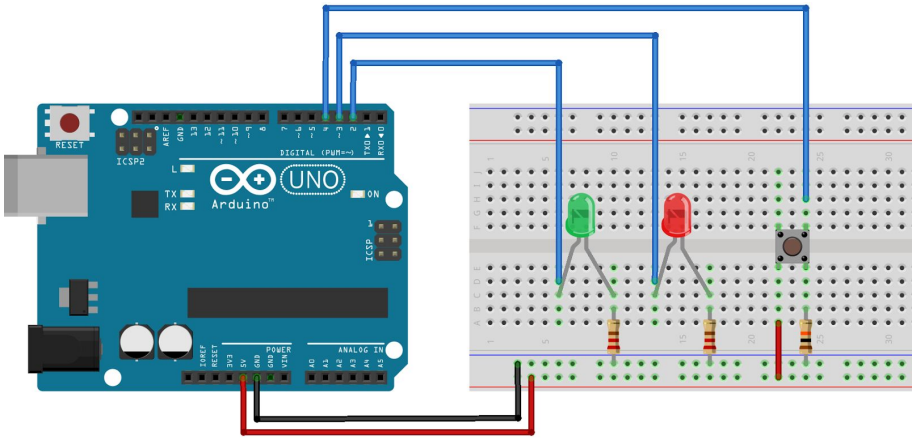
Av og til må vi ta hensyn hvor ofte og hvor raskt funksjonen loop() blir kjørt. Dette blir tydelig når en jobber med å registrere knappetrykk: Hvordan kan vi vite om det er et nytt knappetrykk hver gang vi leser at knappen er trykket ned? Å takle dette problemet kalles for "debounce". Dette snakkes det om i forelesningen 06.02, men kan også leses om her:
https://www.arduino.cc/en/Tutorial/Debounce

Debounce er ikke nødvendig for å få alle systemer til å fungere, men det er en typisk feil som kan føre til at systemet ikke fungerer slik det skal. SOS singalsystemet fra uke 1 trengte ikke debounce for å virke, men hvis for eksempel antall knappetrykk spiller en rolle må du ta hensyn til dette.

1. Skriv kode for å registrere knappetrykk der du tar hensyn til debounce ved bruk av delay(). Skriv antall knappetrykk til serial monitor når knappen trykkes.
2. Tilpass koden slik at du bruker millis() i stedet for delay.
3. **Ekstra:** Tilpass koden slik at en kan holde inne knappen i ubestemt tid uten at det tolkes som flere trykk.

2.6. Knapper og lys

Et knappetrykk kan behandles på forskjellige måter i koden ut i fra hva vi ønsker å oppnå. Koble sammen kretsen som vist i figuren nedenfor.



1. Skriv et program som gjør at lysene oppfører seg på følgende måte: Når knappen ikke er trykket inn skal den røde lampen lyse og den grønne være mørk. Når knappen holdes inne skal den grønne lampen lyse og den røde være mørk.
2. Skriv et program som får knappen til å fungere som en lysbryter. Når knappen trykkes første gang skal begge LED-pærene starte å lyse. Når knappen trykkes neste gang skal de slukke igjen osv.
3. Skriv et program som gjør at LED-pærene bytter på å lyse hver gang knappen trykkes.
4. Ta utgangspunkt i punkt to ovenfor men legg til funksjonalitet som teller hvor mange ganger lyset er skrudd på. Skriv ut denne informasjonen til serial monitor.

2.6. Knappetrykk, lys og forsinkelse

a) Gass og brems

Koble opp en LED og to knapper som du kaller “gass” og “brems”. Når systemet starter opp skal LED-pæren blinke med et tidsintervall på et sekund. Når du trykker på “brems”-knappen skal tidsintervallet reduseres med 100 millisekunder og når du trykker på “gass”-knappen skal tidsintervallet øke med 100 millisekunder. Her må du bruke millis() og ikke delay().

b) Fading

Koble til en RGB LED til Arduinoen slik du gjorde i oppgave 1.4. Koble deretter til tre knapper som du kaller rød, grønn og blå. Det som skal skje når du trykker på en av knappene er at RGB-en skal fade fra en farge til en annen. Så hvis RGB-en for eksempel lyser rødt og brukeren trykker på grønn-knappen skal RGB-en fade sakte fra rød til grønn.

3. Modularisering

3.1. Modularisering av "Digital Hourglass"

Se på koden til oppgave 8 (Digital Hourglass) i Arduinoboken. Selv om programmet har relativt lite funksjonalitet ser vi at det fort kan bli mye kode i loop(). Dette gjør at det kan bli et rotete program som er vanskelig å lese.

1. Ser du noen måte å modularisere denne koden på for å gjøre den mer oversiktlig?
   Hint: prøv å identifiser deler av koden som kan uttrykkes som et funksjonskall. For eksempel på formen "knappTrykket()" eller "blink()".

3.2. Modularisering

Modularisering er veldig viktig i emnet INF1510. Med ryddig kode blir det raskt og enkelt for deg selv og andre på gruppen å endre koden når systemet skal endre seg.

1. Sammenlign de to programmene nedenfor. Hva gjør programmet til venstre, og hva gjør programmet til høyre?

```ino
void setup() {
  pinMode(2, INPUT);
  pinMode(3, OUTPUT);
}

void loop() {
  if(erKnappTrykketNed()){
    lagSosSignal();
  }
}

boolean erKnappTrykketNed(){
  return digitalRead(2);
}

void lagSosSignal(){
  kortBlink();
  kortBlink();
  kortBlink();

  langtBlink();
  langtBlink();
  langtBlink();

  kortBlink();
  kortBlink();
  kortBlink();
}

void kortBlink(){
  blinkI(300);
}

void langtBlink(){
  blinkI(800);
}

void blinkI(unsigned int milliSec){
  digitalWrite(3, HIGH);
  delay(milliSec);
  digitalWrite(3, LOW);
  delay(milliSec);
}
```

```ino
void setup() {
  pinMode(3, OUTPUT);
  pinMode(2, INPUT);
}

void loop() {
  if(digitalRead(2)){
    digitalWrite(3, HIGH);
    delay(300);
    digitalWrite(3, LOW);
    delay(300);
    digitalWrite(3, HIGH);
    delay(300);
    digitalWrite(3, LOW);
    delay(300);
    digitalWrite(3, HIGH);
    delay(300);
    digitalWrite(3, LOW);
    delay(300);
    digitalWrite(3, HIGH);
    delay(800);
    digitalWrite(3, LOW);
    delay(800);
    digitalWrite(3, HIGH);
    delay(800);
    digitalWrite(3, LOW);
    delay(800);
    digitalWrite(3, HIGH);
    delay(800);
    digitalWrite(3, LOW);
    delay(800);
    digitalWrite(3, HIGH);
    delay(300);
    digitalWrite(3, LOW);
    delay(300);
    digitalWrite(3, HIGH);
    delay(300);
    digitalWrite(3, LOW);
    delay(300);
    digitalWrite(3, HIGH);
    delay(300);
    digitalWrite(3, LOW);
    delay(300);
  }
}
```

2. Endre begge programmene ovenfor:
    a. Etter hvert blink skal det skrives “Blink!” i Serial.
    b. Endre de korte blinkene slik at de blir på 150 millisekunder, og de lange slik at de blir på 500 millisekunder i begge programmene.
    c. Hvilket program var enklest å endre? Begrunn svaret kort.

4. Skjold (shields)

4.1. Lyd

Se for deg at du ønsker å lage lage et system med Arduino som skal spille av MP3 lydfiler.

1. Finn en komponent som gir Arduino denne muligheten.
2. Hvordan kan man styre denne komponenten i Arduino-koden din? (for eksempel spille av en lydfil, gå til neste lydfil, stoppe avspillingen, etc.). Gi en kort forklaring/beskrivelse. Her trenger du ikke skrive kode.

5. Nøtter

1. Hvor mye volt gir et 5Volts-batteri som er halvveis tomt?
2. Hva skjer hvis du kobler en lader som har kapasitet på 4A til en telefon som trenger en lader som har kapasitet på 2A?

6. Tilleggsoppgaver

Dette er oppgaver laget for spesielt interesserte som ønsker å lære litt mer. Følgende oppgaver er ikke direkte rettet mot Oblig 2, men til INF1510-prosjektet.

6.1. Kommunikasjon

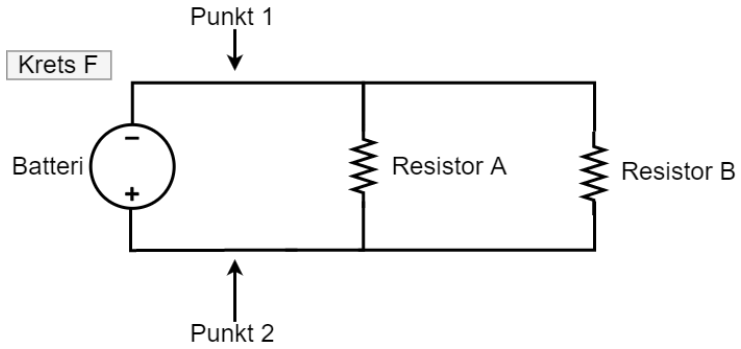
I INF1510-prosjektet kan det hende en arduino ikke er nok til å implementere alle de andre komponentene dere ønsker i en prototype.

1. Finnes det noen måte for to arduino'er å kommunisere? Beskriv alle metoder du finner/kjenner til kort.
2. Sett opp krets og kode for en av tilnærmingene du ga som svar til oppgave 1.

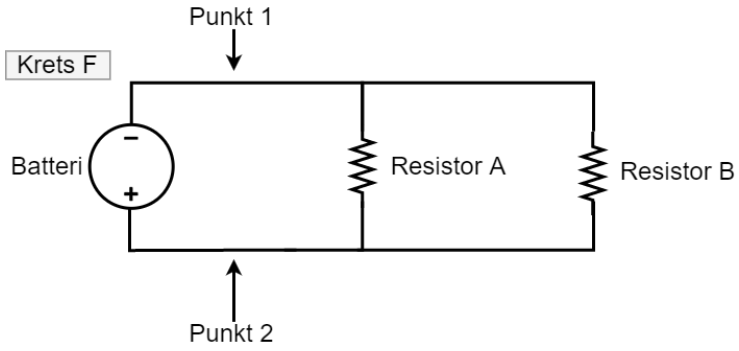
6.2. Multimeter

Et multimeter et er verktøy som lar en måle blant annet volt, strøm, og motstand. Dette er meget nyttig i feilsøking og undersøking av kretser og komponenter. Det finnes flere multimeter på Sonen (Rom Ada 3407) på IFI. Du kan lese om hvordan et multimeter brukes her:
https://learn.sparkfun.com/tutorials/how-to-use-a-multimeter

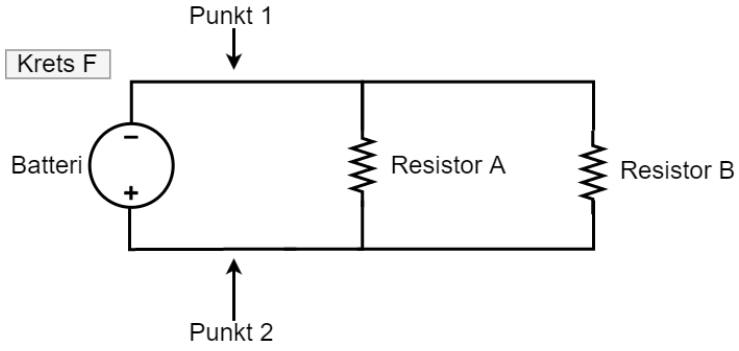
1. Hvordan måler en spenningen over resistor A med et multimeter? Tegn inn et multimeter i kretsen.



2. Hvordan måler en strømmen gjennom resistor A med et multimeter? Tegn inn et multimeter i kretsen.



3. Hvordan måler en motstanden til resistor B med et multimeter? Tegn inn et multimeter i kretsen.



6.3. Motstand i serie og parallell

Motstander er sentrale komponenter i de fleste kretser. Derfor er det viktig å forstå hvordan de oppfører seg.

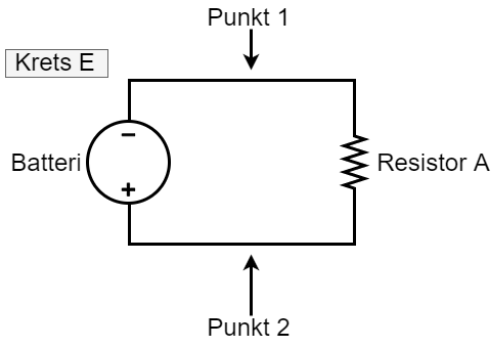
1. Er motstanden mellom punkt 1 og 2 forskjellige i kretsene E, F, og G? Begrunn svaret.
2. Finn motstanden mellom punkt 1 og 2 i kretsene E, F, og G. Begrunn svaret.

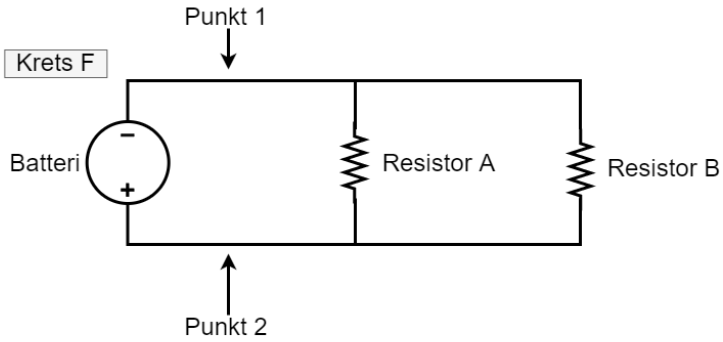
   Hint: Her kan en ikke vite hvor mange ohm, men motstanden kan uttrykkes som for eksempel "R_A-R_B".

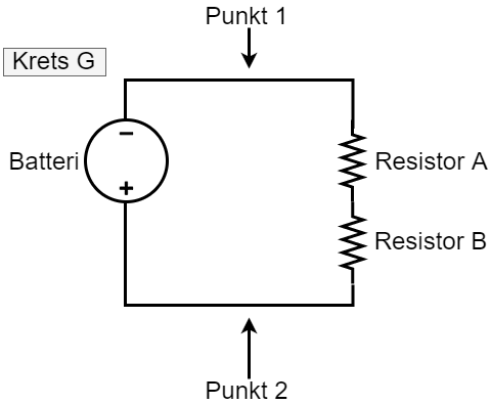
   En kan finne den totale motstanden ("R_{tot}") til to resistorer (R_1,R_2) i parallell med denne formelen:

$$R_{tot} = \frac{R_1 \cdot R_2}{R_1 + R_2}$$

(Du kan lese mer om dette her:
https://learn.sparkfun.com/tutorials/resistors/series-and-parallel-resistors)







6.4. Strøm i serie og parallell

Dette kan du lese om her:
https://learn.sparkfun.com/tutorials/series-and-parallel-circuits

1. Er strømmen gjennom punkt 1, punkt 2, resistor A, og resistor B forskjellige i krets F?
2. Beskriv strømmen gjennom punkt 1, punkt 2, resistor A, og resistor B i krets F. Hint

6.5. Knappetrykk og millis()

Lag et spill som går ut på å teste hvor mange ganger brukeren klarer å trykke på en knapp i løpet av en tidsperiode på fem sekunder. Koble også til 5 LED-pærer som representere antall sekunder som er igjen. Før brukeren starter å trykke skal alle LED-pærene lyse. Når brukeren starter å trykke starter du en "stoppeklokke" som teller ned til null ved å slukke en LED for hvert sekund. Skriv til slutt ut poengsummen (antall trykk) til serial monitor.

6.6. Oppgaver om sikkerhet

De typiske måtene å ødelegge arduinoen på er ved å

- Koble opp kortsluttninger (Så husk å ha nok motstand! Og ta ut strømkilden når dere kobler om.)
- Koble for mye elektrisitet til noen av Arduino sine pins.

Mer konkret: å koble mer enn 5V inn i "5V", for mye strøm, mer enn 13V til "Reset Pin", kortslutte pins til jord, kortslutte pins til hverandre, og mer enn 3.3V til "3.3V").

Men de andre mer grunnleggende komponentene i "Arduino starter kit" kan også bli ødelagt.

1. Finn de tekniske spesifikasjonene til to forskjellige komponenter fra "Arduino starter kit" og finn ut av hvor mye strøm og volt de tåler. Finn også ut hvor mye strøm og volt de trenger for å operere.
2. Hvor mye volt og strøm er trygt å få gjennom kroppen?
3. Hvorfor holder mange ofte en hånd bak ryggen eller i lomma når de jobber med farlig mengde elektrisitet?