IN1060: Bruksorientert design

Individuell obligatorisk oppgave: Arduino 1

Publisert: 21.01.2019
Frist: 08.02.2019 kl 23:59

Arduino skal benyttes som prototypingsverktøy i IN1060, og i denne individuelle obligatoriske oppgaven skal du bli kjent med grunnleggende konsepter.
Oppgaven består av et utvalg fra ukesoppgavene. Dette sikrer at alle har vært igjennom det viktigste før neste oblig, og prosjektarbeidet. Vi anbefaler uansett alle å gjøre samtlige ukesoppgaver, men i denne innleveringen skal dere kun levere de som er oppgitt her.

Innhold i innleveringen
Oppgavene er delt inn i to typer: tekstoppgaver og kodeoppgaver. Dere skal levere følgende:

1. Dokument i PDF-format, hvor alle tekstlige besvarelser er inkludert. Oppgavene merkes med oppgavenummer.
2. Hver kodeoppgave i egen .ino-fil. Merk filen med oppgavenummer både i filnavn og som kommentar i toppen av filen.

Leveres i Devilry innen 08.02.2019 kl 23:59.

Lykke til!

Innhold

1. Tekstoppgaver

På alle oppgavene i del 1 skal dere gi korte tekstsvar på spørsmålene og eventuelt gi eksempler. Leveres som dokument i PDF-format (se forsiden).

1.1. Elektrisitet

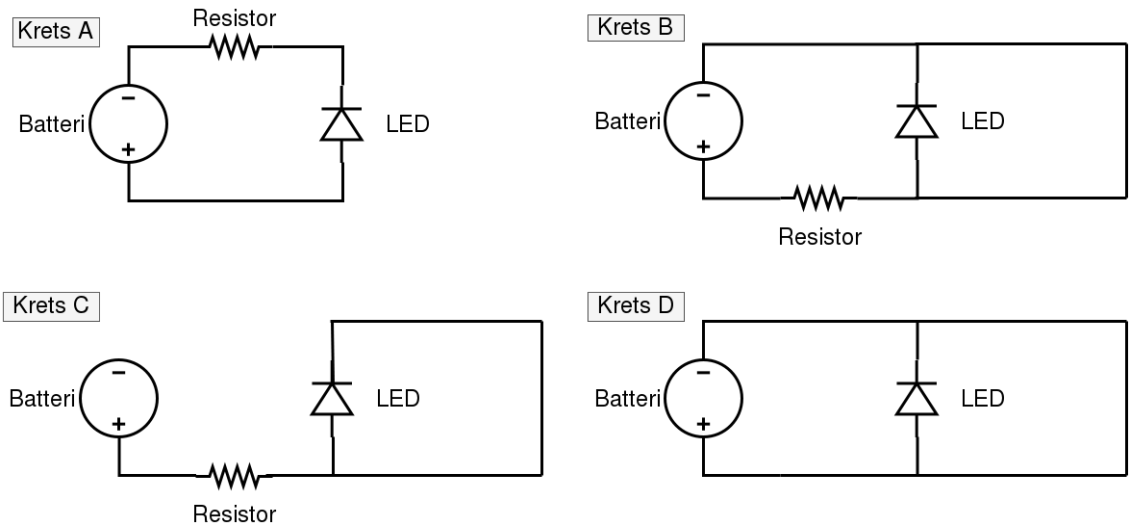
En mye brukt analogi for å forklare elektrisitet er hvordan vann oppfører seg i en vannkran. Bruk denne analogien for å forklare begrepene strøm, spenning og motstand.

1. Hvordan representeres strøm?
2. Hvordan representeres motstand?
3. Hvordan representeres spenning?
4. Hvordan henger strøm, spenning og motstand sammen?

1.2. Dårlige kretser

Kortslutninger og åpne kretser er typiske problemer dere kan møte i arbeid med Arduino. Typisk om en endrer på en krets mens elektrisitet fortsatt er koblet til.

1. Hva er en kortslutning? Ikke skriv mer enn et par setninger.
2. Hva er en åpen krets? Ikke skriv mer enn et par setninger.
3. Vil LED-pæren lyse i følgende kretser? Begrunn svaret kort.



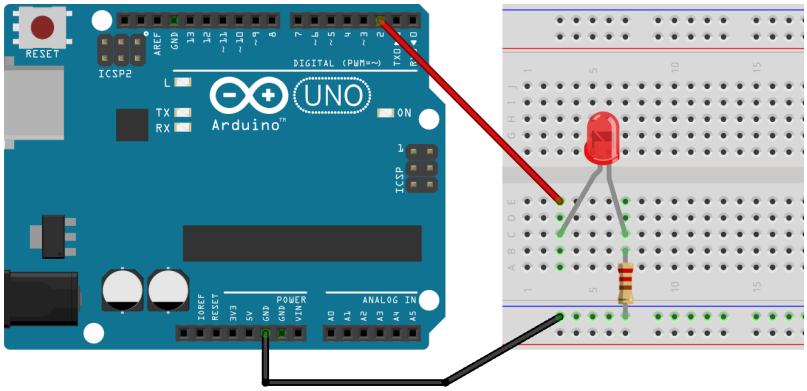
1.3. Analoge og Digitale signaler

Komponenter krever forskjellig elektrisitet for å operere/fungere. En må for eksempel ta hensyn til forskjellen på analoge og digitale signaler.

1. Forklar forskjellen på analoge og digitale signaler. Ikke skriv mer enn et par setninger.
2. Gi et eksempel på noen komponenter i arduino-kittet som sender/mottar/kan brukes med digitale signaler, og gi et eksempel på noen komponenter som sender/mottar/kan brukes med analoge signaler.

1.4. Loop og blink

Anta at port 2 er koblet til en LED slik som i figuren nedenfor.



Hvordan vil lampen oppføre seg når disse kodesnuttene kjører? Begrunn svaret kort.

1.

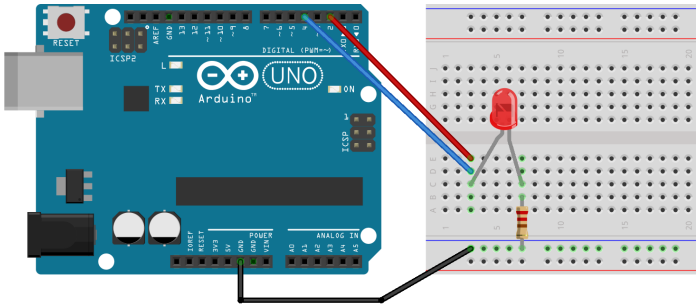
```
7  void loop() {
8      digitalWrite(2, HIGH);
9      delay(1000);
10     digitalWrite(2, LOW);
11 }
```

2.

```
7  void loop() {
8      digitalWrite(2, HIGH);
9      delay(1000);
10     digitalWrite(2, LOW);
11     delay(1000);
12 }
```

3. Følgende kode, med følgende krets

```
5  void setup() {
6      pinMode(2, OUTPUT);
7      pinMode(4, INPUT);
8  }
9
10 void loop() {
11     digitalWrite(2, !digitalRead(4));
12     delay(1000);
13 }
```



1.5. 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?

1.6. 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() funksjonen. Dette gjør at det kan bli et rotete program som er vanskelig å lese og jobbe videre med.

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

1.7. Lyd

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

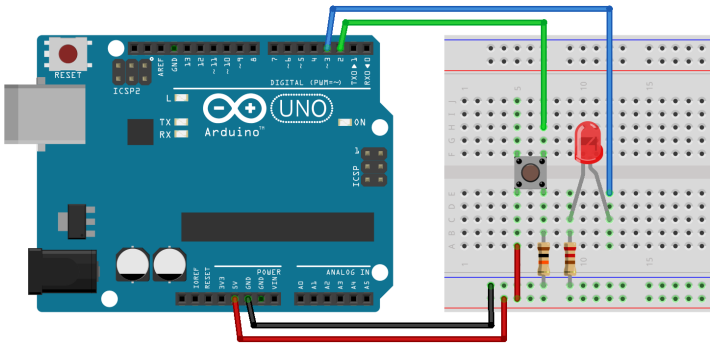
1. Finn en komponent som gir Arduino denne muligheten.
2. Hvordan kan man styre denne komponenten i Arduino-koden din? (for eksempel spille av en lydfil, gå til neste lydfil, stoppe avspillingen, etc.). Gi en kort forklaring/beskrivelse. Her trenger du ikke skrive kode.

2. Kodeoppgaver

For oppgavene i del 2 skal dere levere kode (se forsiden). **For oppgaver med flere deloppgaver hvor koden endres, holder det at dere leverer programmet slik det ser ut til slutt etter siste deloppgave.**

2.1. SOS signalsystem

Lag et SOS signalsystem: Når knappen trykkes ned skal LED-pæren blinke i et SOS-mønster (tre korte blink, tre lange blink, og så tre korte blink). Gjør dette på følgende måte:



1. Koble opp kretsen vist til høyre.

2. Bruk funksjonen "pinMode()" til å sette opp en pin som digital output, og en pin som digital input i funksjonen "setup()".

3. I funksjonen "loop()": Skriv kode som lytter til om knappen er trykket ned eller ikke. Gjør dette gjennom den digitale input pin-en du satte opp i "setup()" funksjonen.

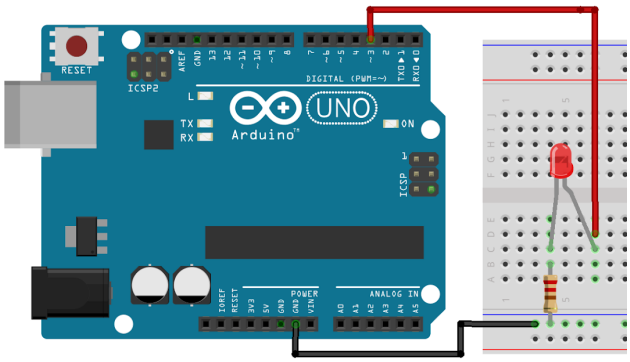
4. For å gjøre programmet ditt enklere å lese, lag en ny void funksjon med navn "sos()". Når knappen trykkes skal funksjonen "sos()" kalles.

5. Skriv kode som gjør at LED-pæren blinker i mønsteret til en sos-beskjed i funksjonen "sos()".

   Hint: Her kan det lønne seg å skrive flere funksjoner. F.eks en funksjon "blinkI(milliSec)" som du kaller tre ganger med et lavt tall, tre ganger med et høyt tall, og så tre ganger med et lavt tall igjen. Eller metodene "kortBlink()" og "langtBlink()" hvor du kaller "langtBlink()" tre ganger, men "kortBlink()" tre ganger før og etter det.

2.2. 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 potensiometer (Her er det nyttig å forstå kretsen og koden på side 74 i forelesningen om Prototyping med arduino del 2). Her må du bruke både analogRead() for å lese fra potentiometeret og analogWrite() til å sette lysstyrken. Denne deloppgaven er uavhengig fra forrige deloppgave.

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? Legg ved en kommentar i koden hvor dere besvarer dette kort.

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? Legg ved en kommentar i koden hvor dere forklarer kort hvordan.

For denne oppgaven holder det å levere koden du har for løsningen av deloppgave 3. Svar også på deloppgave 4. og 5 som korte kommentarer i koden.

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 ofte litt ekstra planlegging for at koden skal fungere som man vil. 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. 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 04.02, 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.

For denne oppgaven holder det å levere koden du har for løsningen av deloppgave 2.