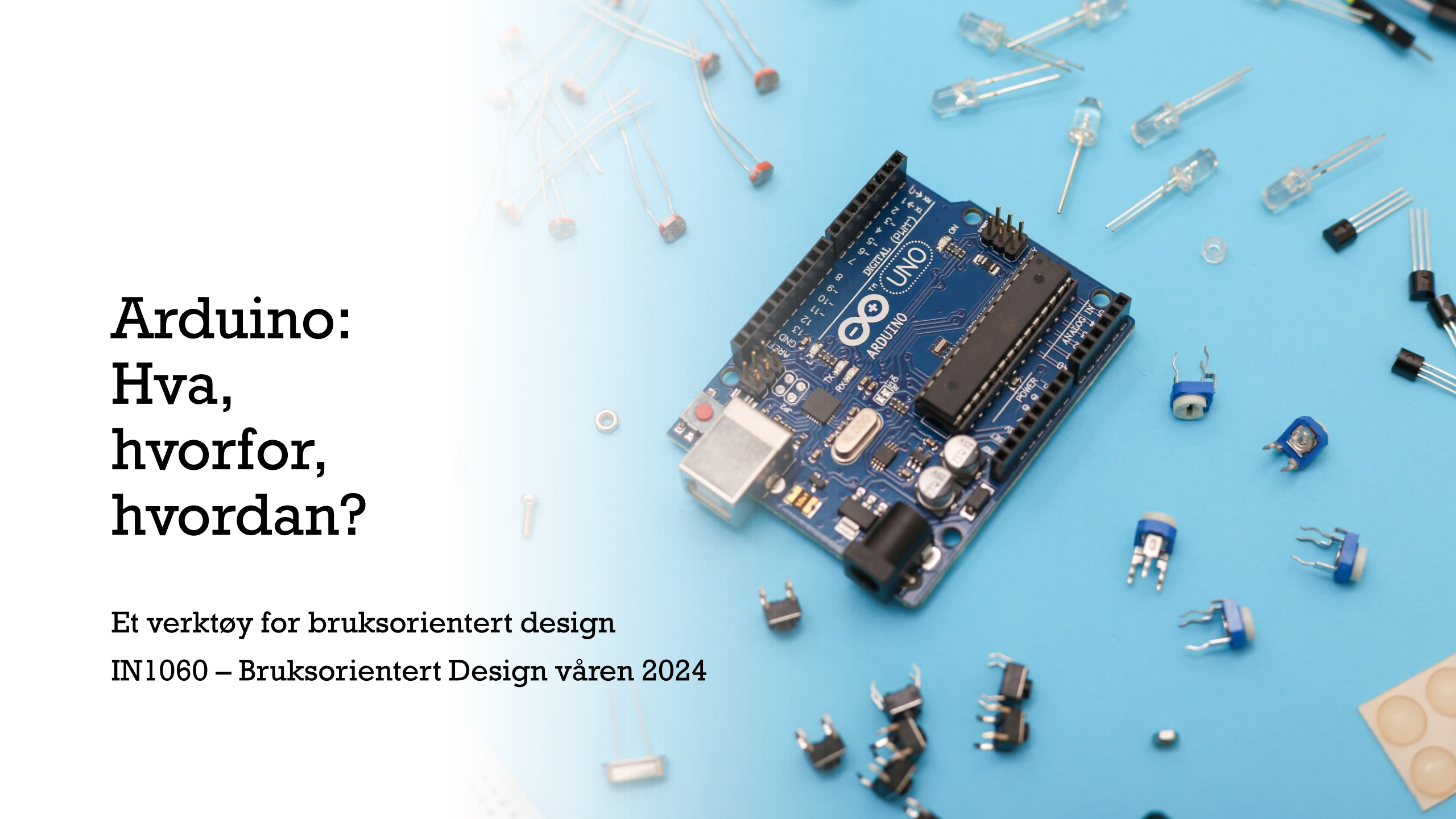
Arduino: Hva, hvorfor, hvordan?

Et verktøy for bruksorientert design

IN1060 – Bruksorientert Design våren 2024



Dagens forelesning: hva, hvorfor, hvordan?

1: Informasjon om undervisning, innleveringer og ressurser

2: Bakgrunn: **Hva** er Arduino, og **hvorfor** velger vi Arduino som verktøy i IN1060?

3: **Hvordan** bruker vi Arduino som verktøy for bruksorientert design,

og hvilke ferdigheter trenger vi å lære?

a) Design: brukerundersøkelser og fysisk prototyping

b) Programmering av Arduino

c) Elektriske kretser med Arduino

grunnleggende prinsipper for strøm og kretser

se nærmere på noen av delene i Arduino-settet

Arduinoundervisningen februar 2024

Undervisning

Forelesninger tirsdag 10.15-12.00

Orakeltjeneste onsdag 10.15-12.00

Hjelp med ukesoppgaver og arduino-obliger, gjennomgang av utvalgte oppgaver, sitte sammen og jobbe, diskutere ideer og problemløsning.

Gruppetimer (se timeplanen for din gruppe)

Få hjelp med gruppedannelse, ukesoppgaver, obliger og prosjektutviklingen, gjennomgang av utvalgte eksempler

Kontakt

Teamskanal for IN1060, egen Arduinokanal

Arduino: Knut

Gruppearbeid og obliger: Gruppelærer

Oppgaver

2 ukesoppgaver

2 individuelle obliger

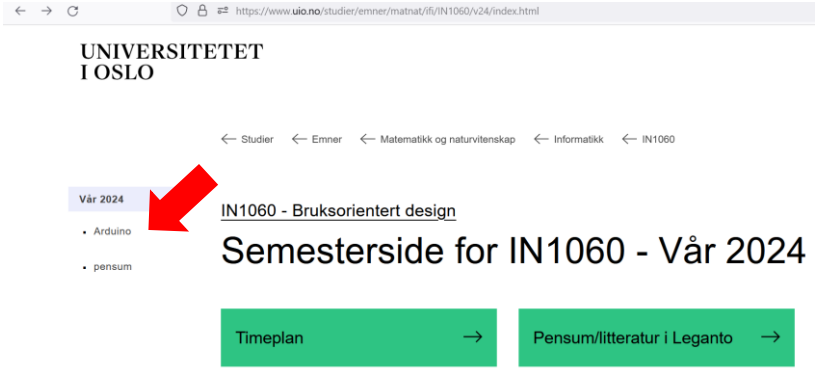
Oblig 1: Frist 13.02.23 kl.10.00

Oblig 2: Frist 05.03.23 kl.10.00

> Konkurranse!

Kl. 10.00 tirsdager av hensyn til de som har gruppetimen på mandager

De første oppgavesettene er nå publisert på semestersidene under Arduino, se meny oppe til venstre på semestersiden.



Hvordan få god hjelp med Arduino i IN1060

Beskriv og dokumenter:

- Hva prøver du å få til?
- Hva slags problem har du fått?
- Hva har du gjort for å prøve å løse det, og hva skjedde da?
- Send med kode
- Send bilde av krets
- Send koblingsskjema/kretstegning/kretsdiagram (mer om det i neste time)
- Vi vet det kan være frustrerende, men prøv å holde frustrasjonen ute av henvendelsene – vit at du får hjelp til å løse problemet
- Prøv å løse oppgaver i god tid, så har du tid til å spørre om hjelp før fristen går ut

Kontakt riktig person: Knut og Liv

- Gruppelærerne kan mye om Arduino, men er ansatt for å bistå dere med bruksorientert design. Knut og Liv er Arduino-orakler og kan hjelpe dere med Arduino. Hvis det er spørsmål som må diskuteres meg faglærer tar Knut og Liv eller gruppelærere kontakt med meg. På den måten slipper dere å vente på svar.
- Spør en person av gangen, så slipper vi dobbeltarbeid

Bruk Sonen!

ÅPEN SONE FOR EKSPERIMENTELL INFORMATIKK

Vi har et eget makerspace på Ifi!

https://www.mn.uio.no/ifi/tjenester/publikum/sonen/

Du finner Sonen i 3. etg, på rom Ada (3407).

Åpent ti, ons og to 12.15-16.00

Adrian som driver Sonen kan også hjelpe dere. Han kommer og hilser på i neste time! Han kommer også til kick-off'en og holder 3D-printingskurs for dere.

Følg rådene for å få god hjelp også når dere spør Adrian om hjelp

Arduinosett

Fra i fjor har vi brukt Arduino Student kit.

Settet kan kjøpes på Akademika Blindern

Dere kan kjøpe settet hvor som helst hvor det selges

Dere kan også bruke Arduino startsett som har blitt brukt tidligere

(- mister kode til nettressurs, men kan bruke bok)

Litt om hva jeg jobber med

Teknologi som materiale for design

Elektriske komponenter, programmering og AI kan tenkes på som materialet vi skaper design med

Gjøre teknologi tilgjengelig og forståelig i designprosessen

Kan fysisk prototyping hjelpe deltagere med å utvikle bedre teknisk forestillingsevne (tecknical imagination)?

Fysisk prototyping i PD

Hva skjer i designprosessen når vi lager noe sammen versus snakker og skriver om noe?

Finne gode måter å involvere deltagere i designprosesser

Bruke fysisk prototyping (lage ting) sammen med deltagere for å skape designmuligheter basert på deres behov og ønsker

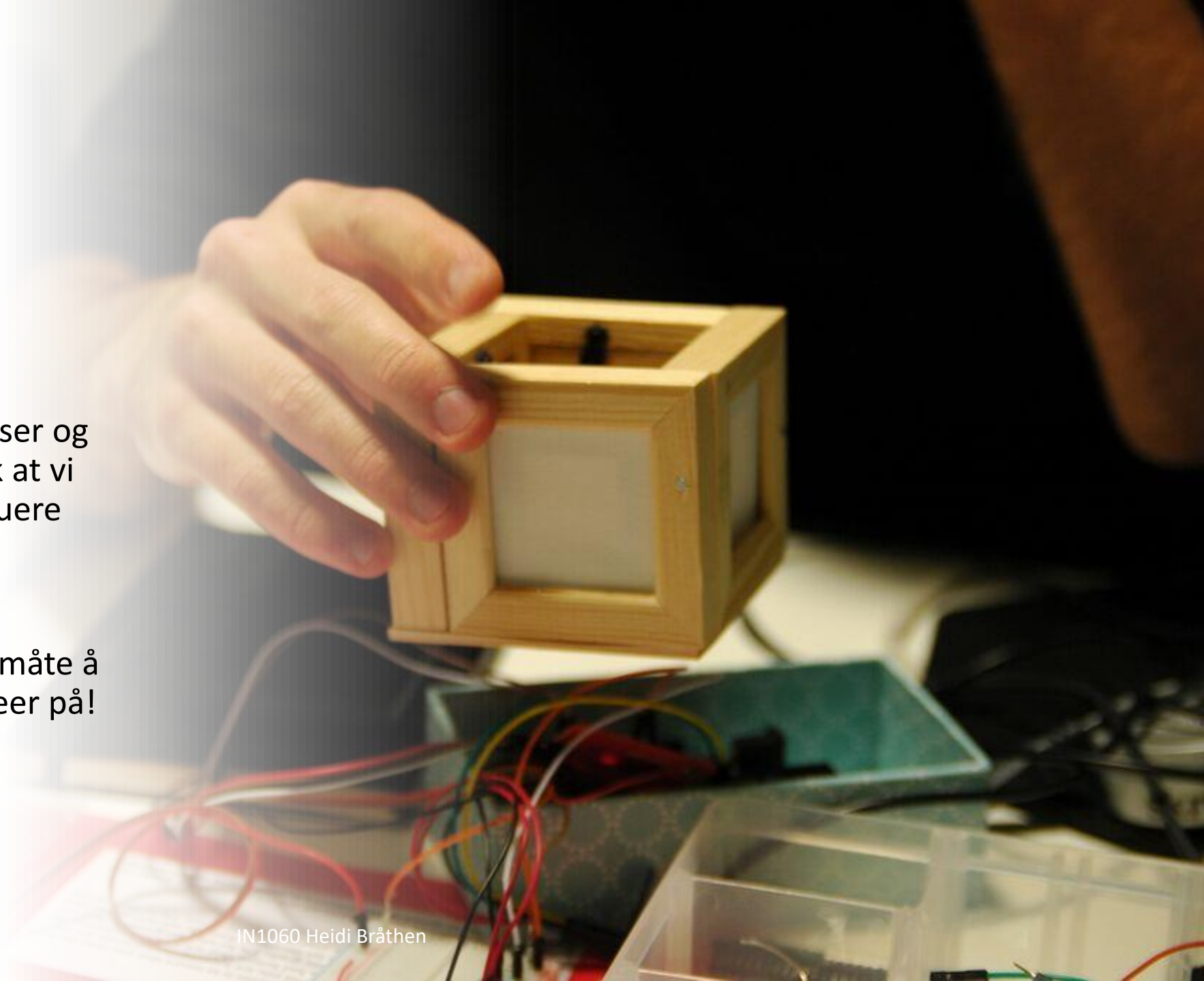
HVA er Arduino?

Et åpent prototypingsverktøy som lar oss bygge interaktive ting

Arduino er et verktøy for prototyping

PROTOTYPING er å lage skisser og modeller av designideer slik at vi kan diskutere, teste og evaluere dem

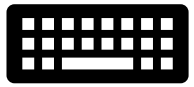
Prototyping er også en god måte å generere nye eller bedre ideer på!



Arduino er et verktøy som lar oss bygge «interaktive ting»

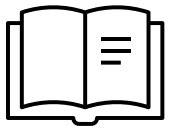
INTERAKTIVITET er interaksjon mellom bruker og et system, hvor brukerens handlinger påvirker systemet, og omvendt.

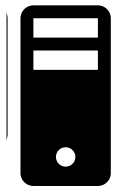
Arduino lar oss:



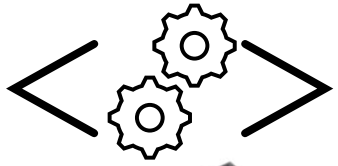
lese *signaler fra brukere og omgivelser via sensorer «sanser» eller «leser»*

«Input»



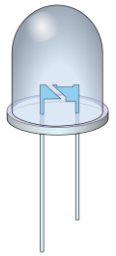


gjøre *noe med signalene programmering behandler data behandler/bearbeider*



sende *signaler ut til forskjellige komponenter*
«skriver»

«Output»

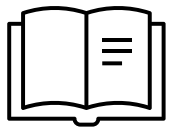


Arduino er en plattform for elektronikk

Arduino er en åpen prototypingsplattform som lar oss bygge elektroniske kretser og styre dem med enkel programmering

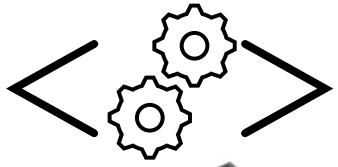
Arduino kan sanse elektriske signaler som kommer inn, behandle dem i henhold til programmeringen og sende elektriske signaler ut

***lese** signaler fra sensorer*

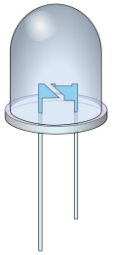


Sensorer er komponenter som tar inn signaler fra omverdenen

***gjøre** noe med signalene*
Programmering/ skript



***sende** eller skrive signaler ut aktuatorer*



Aktuatorer er komponenter som gjør om energi til bevegelse, som motorer. I vår modell bruker vi begrepet om output

Åpen kildekode (Open Source)

Arduino har åpen kildekode og maskinvare (software & hardware)

ÅPEN KILDEKODE

Utviklerne publiserer all koden for alle, slik at alle som ønsker kan delta i å foreslå endringer, fikse bugs og videreutvikle prosjektet. Arduino IDE er åpen. Et annet kjent eksempel er Linux.

ÅPEN MASKINVARE

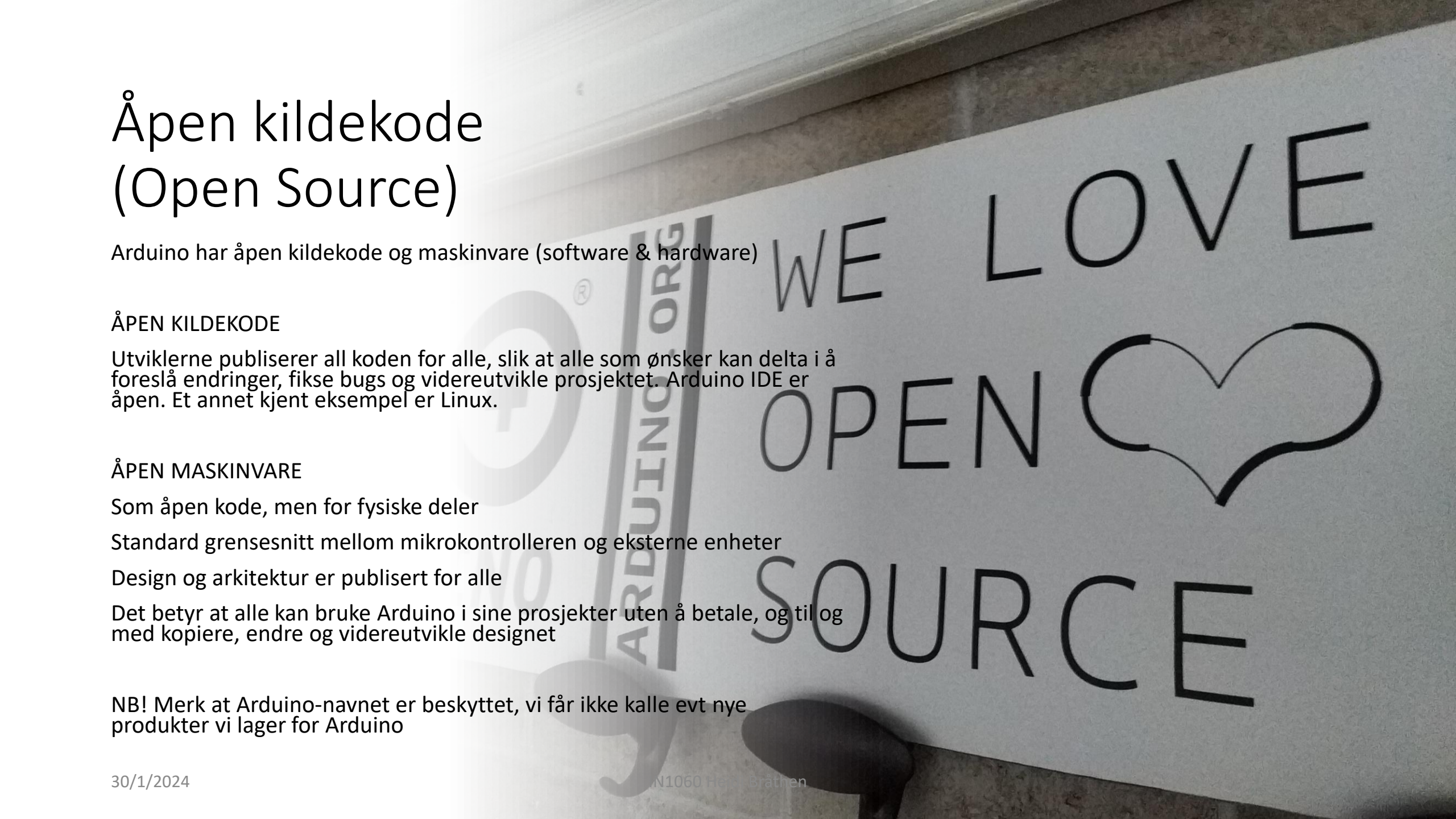
Som åpen kode, men for fysiske deler

Standard grensesnitt mellom mikrokontrolleren og eksterne enheter

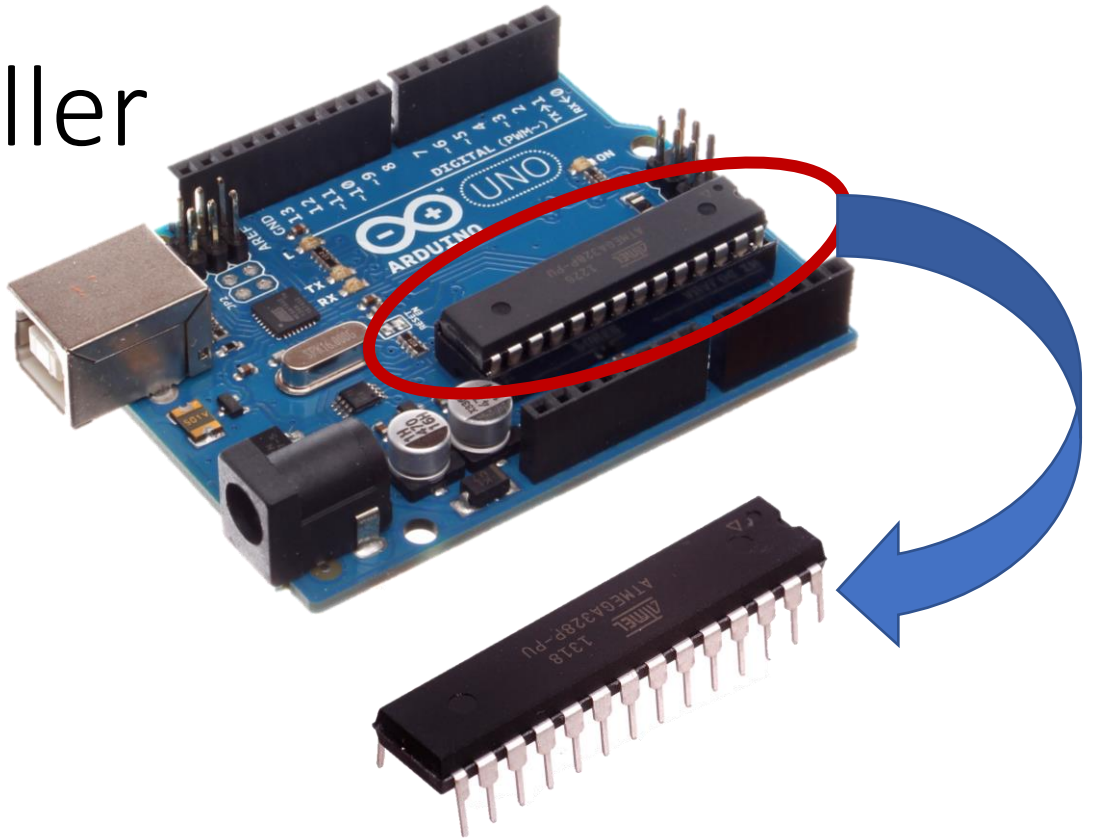
Design og arkitektur er publisert for alle

Det betyr at alle kan bruke Arduino i sine prosjekter uten å betale, og til og med kopiere, endre og videreutvikle designet

NB! Merk at Arduino-navnet er beskyttet, vi får ikke kalle evt nye produkter vi lager for Arduino



Arduino er en mikrokontroller

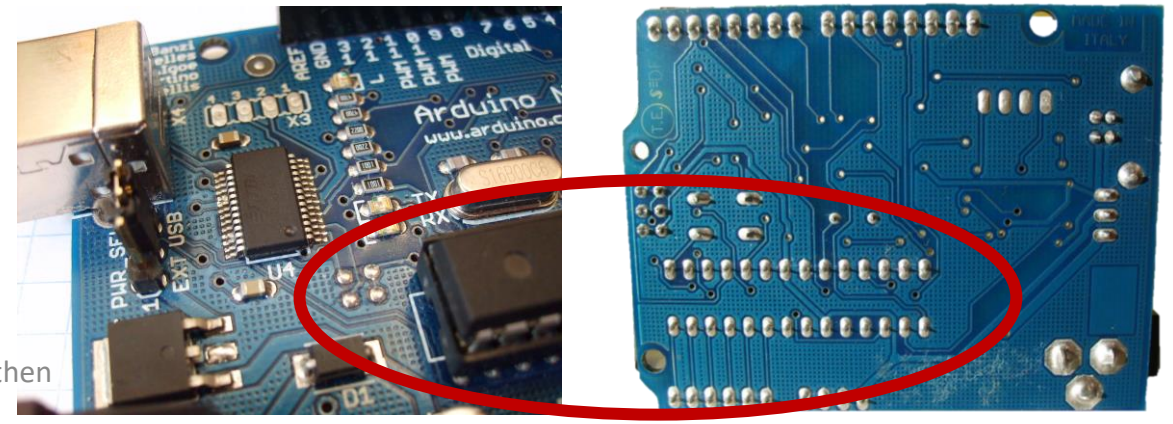


Arduino minner om en liten datamaskin, men er mye enklere enn de PC-ene vi bruker.

En MIKROKONTROLLER er en enkel datachip med minimal prosessorkraft, ofte laget for automatisk kontroll av en ekstern enhet

Vi kan kalle hele Arduinobrettet for en mikrokontroller. Selve mikrokontrolleren, eller chipen, er den lange, svarte brikken.

Resten er brukergrensesnittet med kontakter («pin»), nødvendige komponenter og reset-knapp. De er knyttet til mikrochipen med ledningsspor



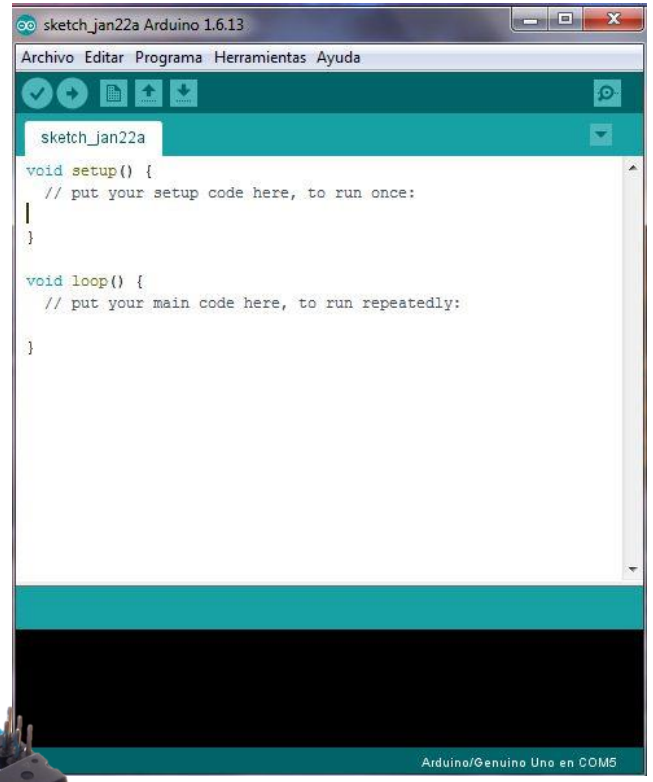
Arduino er mer enn mikrokontrolleren

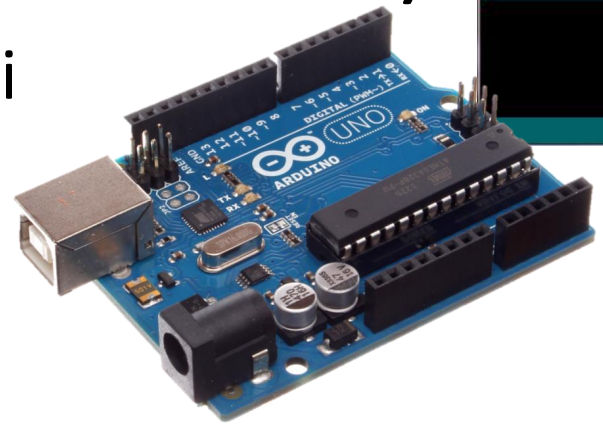
Arduino består av:

Mikrokontrolleren (brettet)

Arduino IDE (Integrated Development Environment)

Programmet vi programmerer Arduino i





Historien: Elektronikk for legfolk

Arduino ble laget for å støtte kunststudenter

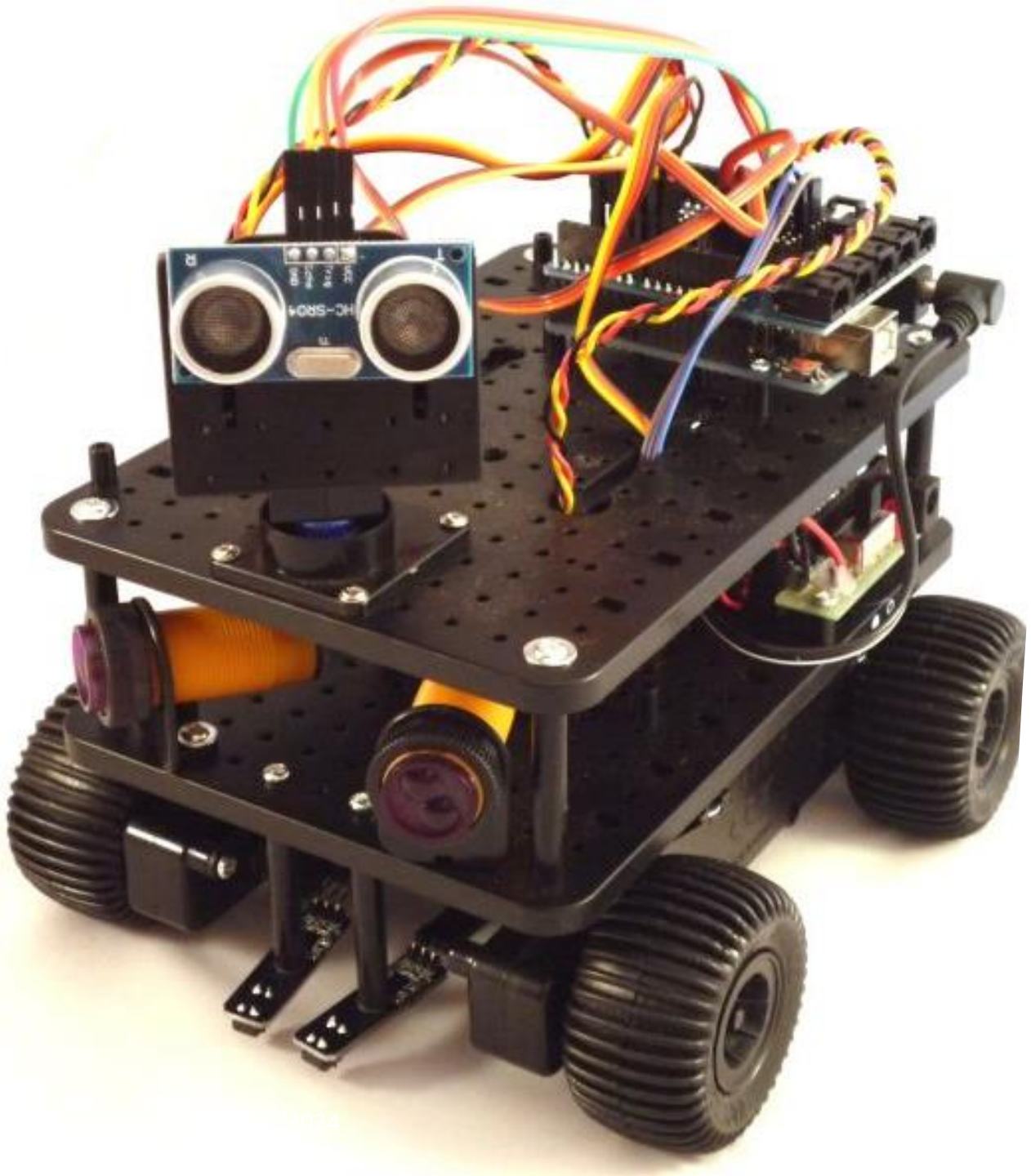
Kunststudenter i Italia hadde behov for en enkel måte å animere kunstprosjektene sine med elektronikk og programmering

Utviklet for å gjøre det mulig og enkelt for folk uten teknisk bakgrunn å programmere og lage elektriske kretser

NB! Krever likevel at man lærer en god del om programmering og bygging av kretser

2003: Wiring og Processing Interaction Design Institute Ivera (IDII) i Italia

2005: Arduino og Arduino IDE David Cuartielles og Massimo Banzi



Muligheter og begrensninger

Designe innenfor rammer

tid, kompetanse, økonomi, tekniske muligheter

Mange muligheter for prototyping!

Arduino project hub: https://create.arduino.cc/projecthub

Begrensninger

Begrenset prosessor og minne betyr at for eksempel videoredigering og veldig kompliserte utregninger er utenfor de tekniske rammene

(det finnes alternativer, som Raspberry Pi)

HVORFOR velger vi Arduino?

Hvorfor Arduino?

- Viktig at interaksjonsdesigner kjenner til andre grensesnitt enn skjermbaserte
  - Lar oss utforske den tekniske siden av design
  - Eksempel på designmateriale man kan jobbe med
- Åpent
- Aktivt community, mye tilgjengelig kunnskap
  Mange bidrag = mange muligheter
- Fungerer på tvers av plattformer: Windows/Mac/Linux
- Krever ikke mye *forkunnskaper* i programmering og elektronikk
  Bygger på det dere kan fra før
  Kan teste tidlig
  Mulig å utvide til avansert bruk

HVORDAN skal vi bruke Arduino?

Og HVORDAN virker Arduino?

Hvordan skal vi bruke Arduino i dette kurset?

Brukerundersøkelser først

Brukernes behov og krav til systemet vi skal lage må komme før valg av Arduino-løsninger

Arduinoens roller i prosjektet:

Utforske og eksperimentere for å oppdage muligheter for design

generere ideer

utforske ideer

kommunisere ideer

teste ideer

evaluere med brukere

brukere kan gi tilbakemelding på tidlige prototype

demonstrere løsninger

Endelig teknisk løsning skal være forankret i Arduino

Hva skal vi lære om Arduino i dette kurset?

Mål for februar

- Bli komfortable med å uttrykke egne designideer med Arduino
- Kunne bidra aktivt til prosjektgruppens Arduinoprosjekt

Ferdigheter vi trenger

Programmering i Arduino

Elektriske kretser

Vi må også lære nok om måten Arduino er bygget opp på for å bruke det vi kan til å ta i bruk stadig nye moduler:

Lære «språket»: navn på komponenter og programmeringsfunksjoner

Problemløsning og debugging

Hvilke ressurser som finnes og hvordan vi bruker dem

Mer øvelse = mer erfaring = flere muligheter = større *designspace* = mer kreativ utfoldelse

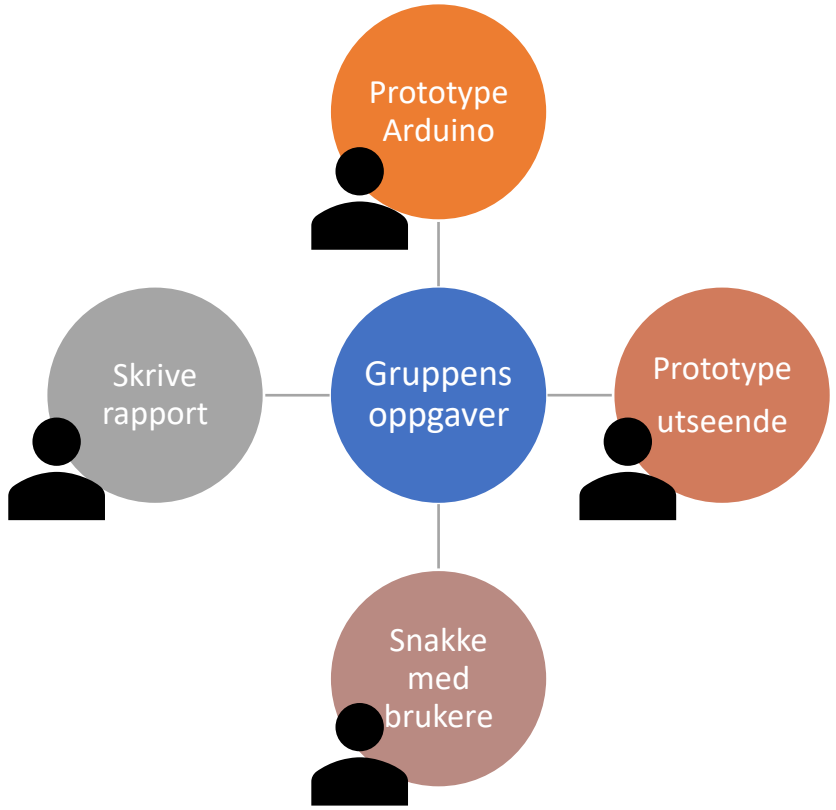
Mål: Alle skal bidra til prosjektgruppens Arduino-prototype

Vanlig problem: En på gruppa lager Arduinoprosjektet alene fordi mange føler at de ikke kan nok, og man deler prosjektet mellom seg i for store biter

Alle bør involvere seg i alt, selv om det er fint at en eller to har hovedansvar for hver del

- Skaper bedre design

- Mer robust gruppe og prosjekt



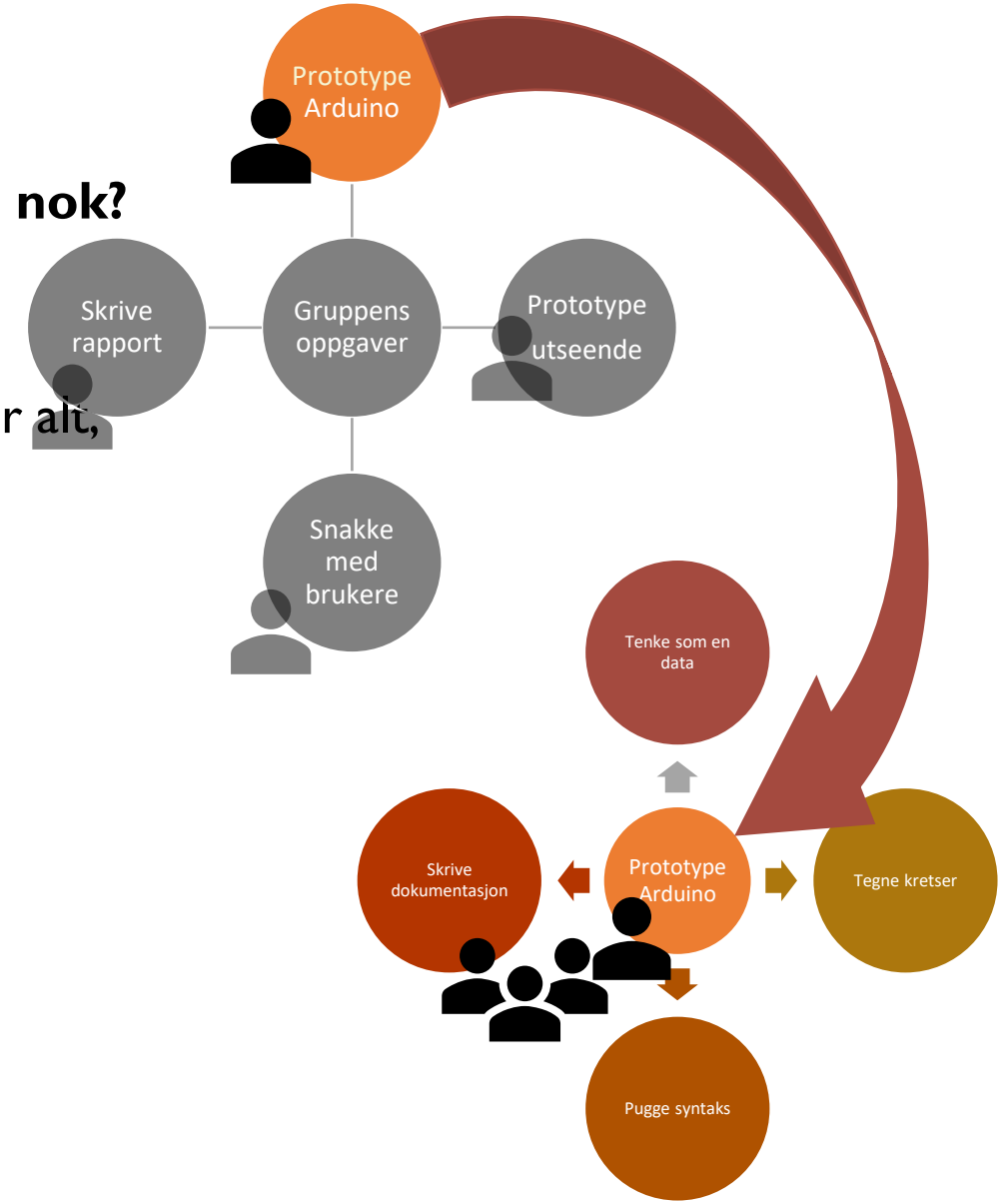
Mål: Alle skal bidra til prosjektgruppens Arduino-prototype

Hvordan kan man bidra hvis man ikke føler at man kan nok?

Prototyping med Arduino krever ikke bare én type ferdighet

Man kan for være gode på mange ting selv om man ikke mestrer alt, som:

- pugge og huske syntaks
- huske og forstå hvordan komponenter virker
- lage god dokumentasjon (kommentere kode, tegne kretsdiagrammer)
- å løse problemer som en datamaskin kan forstå
- (skrive algoritmer)
- feilsøke kretser eller kode
- utvikle ideer for hvordan Arduino kan brukes som designløsning



Introduksjon til Arduinoprogrammering

Hvordan virker Arduino?

Arduinoprogrammering 101

Vi må lage programmer i Arduino IDE, (på PC) og sende dem til Arduinoen for å kontrollere kretsene vi bygger

Et Arduinoprogram kalles en skisse (sketch, fra kunstverdenen)

Arduino har et eget standardbibliotek, en samling med innebygde ferdige elementer vi skal bruke, som funksjoner og variabler

Arduinobiblioteket er skrevet spesielt for kombinasjonen av mikrokontrolleren og grensesnittet på Arduinobrettet

variabler tar vare på verdier når programmet kjører, de er som merkede parkeringsplasser eller oppbevaringsbokser for verdier. Vi må deklarere variabler med riktig datatype, for eksempel int for heltall

funksjoner() er kodeblokker omsluttet av klemmer {} som utfører konkrete oppgaver. funksjoner() er det samme som funksjoner i Python og metoder i Java. Funksjonsnavn avsluttes med parantes (). Funksjoner kan ta imot variabler og returnere verdier.

Deklarere en variabel:

```
datatype variabelnavn = verdi;
```

Bibliotek

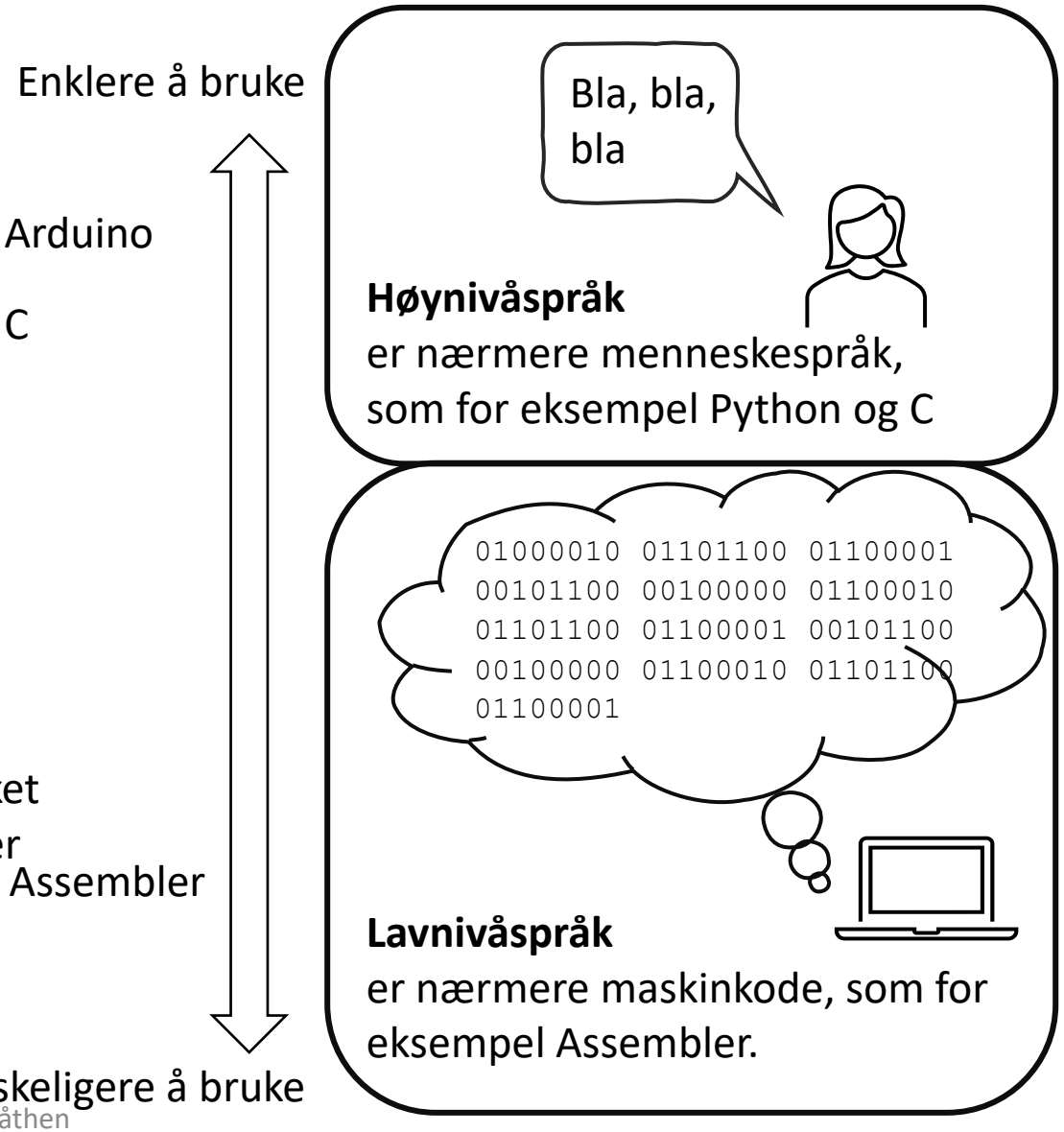
mapper som samler kode-elementer med samme tema. Arduino-biblioteket inneholder mange elementer som alle brukes til å kontrollere selve Arduinobrettet med kode

Hvilket språk er egentlig Arduino?

Mikrokontrolleren i Arduino kan programmeres med Arduino. Arduino er skrevet for kombinasjonen av mikrokontrolleren og grensesnittet på Arduinobrettet. Arduinospråket er egentlig et bibliotek skrevet i C++

Mikrokontrolleren i Arduino kan programmeres med C, men da kan man ikke dra nytte av det spesielle grensesnittet på Arduinokortet, som f.eks. pins.

Mikrokontrolleren i Arduino kan programmeres med lavnivåspråket Assembler, men da må man kun bruke 282 kortform-kommandoer



Arduinobibliotekets tre hoveddeler

Arduino language reference gir full oversikt over biblioteket: https://www.arduino.cc/reference/en/

Arduino har et eget standardbibliotek med innebygde ferdige elementer som bruker egenskapene til Arduinobrettet. De deles i tre undergrupper

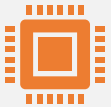
funksjoner() er kodeblokker omsluttet av klemmer {} som utfører konkrete oppgaver. funksjoner() er det samme som funksjoner i Python og metoder i Java. Funksjonsnavn avsluttes med parantes () og tar imot variabler.

Eks: pinMode(), digitalWrite()

variabler tar vare på verdier når programmet kjører, de er som merkede parkeringsplasser for verdier

KONSTANTER er variabler som aldri endres. De skrives med store bokstaver. F.eks: LED_BUILTIN, HIGH, OUTPUT

Datatyper angir hva slags verdier som lagres F.eks: int, string



«struktur» i Arduino-språket er resten; det som ikke er funksjoner eller variabler.

F.eks: setup() og loop(), operatører (+, -, *), sammenlignere (<, >, =),

«kontrollstrukturer» (if, else, do-while), «annen syntaks» (klemmer {})

Variabler i Arduino

variabler tar vare på verdier når programmet kjører, de er som merkede parkeringsplasser eller oppbevaringsbokser for verdier. Vi må deklarere variabler med riktig datatype, for eksempel int for heltall

datatype forteller kompilatoren hvor mye minne variablene vil trenge

Deklarere en variabel (syntaks)
`datatype variabelnavn = verdi;`

Bruke en variabel
`Variabelnavn = verdi;`

Skop
Skop er hvor variabelen kan brukes av programmet.

Globale variabler deklareres utenfor metodene i programmet og er tilgjengelig for alle metodene.

Lokale variabler deklareres i en metode, eller en løkke, og er da kun tilgjengelig innenfor skopet til den funksjonen/kodeblokken.

Beslutninger: if

If sjekker en betingelse og utfører en kommando eller et sett med kommandoer hvis betingelsen er «true»

Operatører i Arduino

Symboler som utfører en operasjon, som et regnestykke eller en sammenligning

Syntaks

```
if (betingelse) {
   //skal noe skje }
```

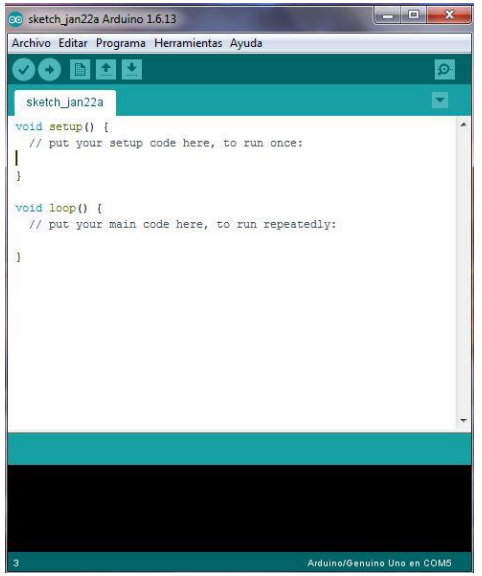
Parametre

Et bolsk utttrykk som kan være sant/true eller ikke/false

Eksempel

```
{// sjekker om en knapp er trykket på
if (buttonState == HIGH)

// hvis sant, skru på LED
        digitalWrite(ledPin, HIGH);

        } else {

        digitalWrite(ledPin, LOW);
```

Arduino IDE
Integrated Development Environment



- Åpne Arduino IDE
- Koble Arduinoen til PC-en med USB-kabelen

For å installere Arduino IDE

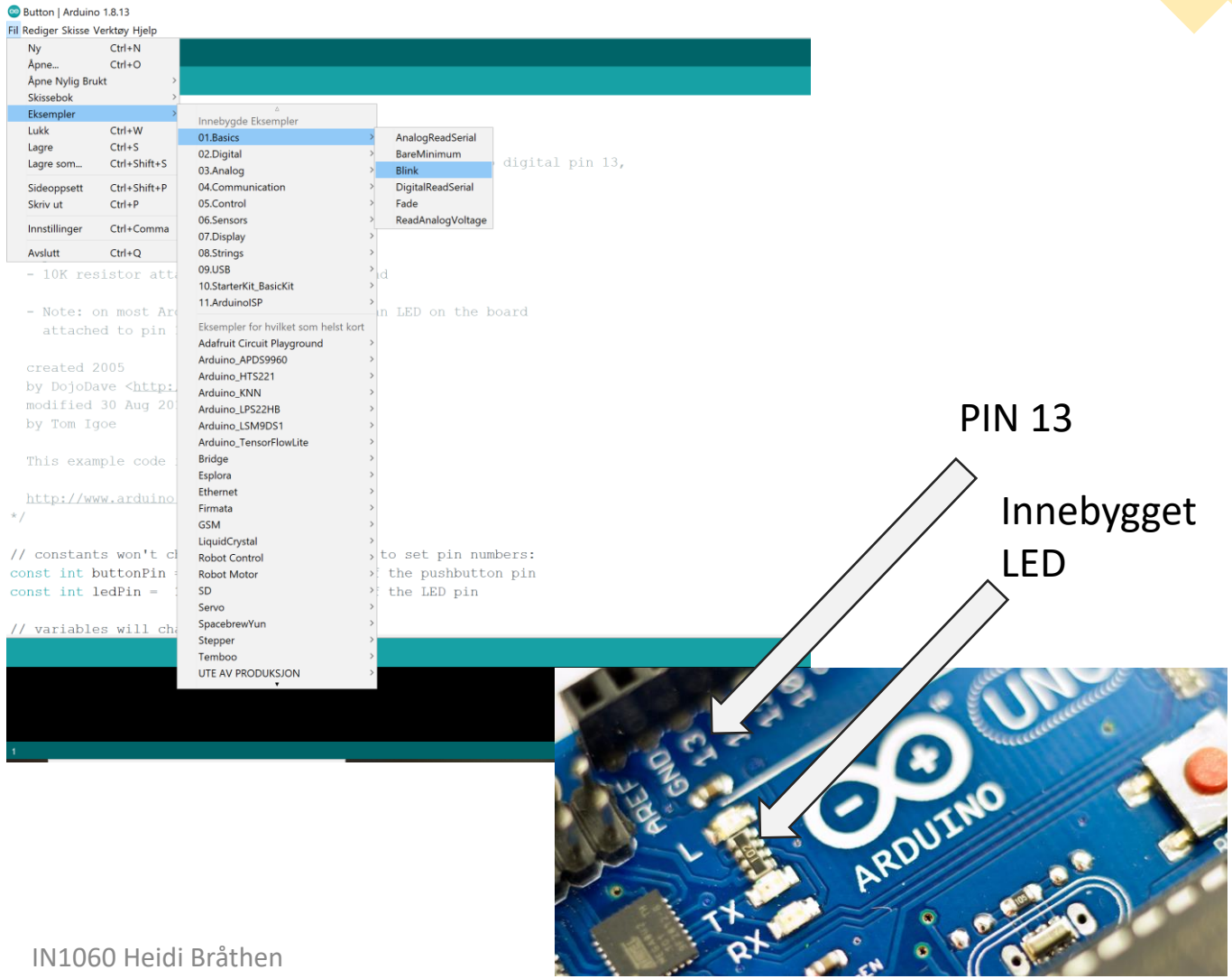
- Gå til http://arduino.cc/en/main/software og last ned versjonen som passer ditt operativsystem
- Kjør installasjonen

Arduinoens Hello World: LED (Light Emitting Diode)

- Arduinobrettet har en innebygget test-LED knyttet til pin 13
- Vi skal bruke eksempelkode for å få test-LEDen til å blinke
- I Arduino IDE, åpne Fil – Eksempler – 01 Basics – Blink

Dere trenger:

- **Det blå Arduinobrettet,**
- **PC med Arduino IDE**
- **USB-kabel**



Eksempelkode

Arduino IDE har en rekke små kode-snutter som hjelper oss å komme raskt i gang

Send skissen til Arduinoen

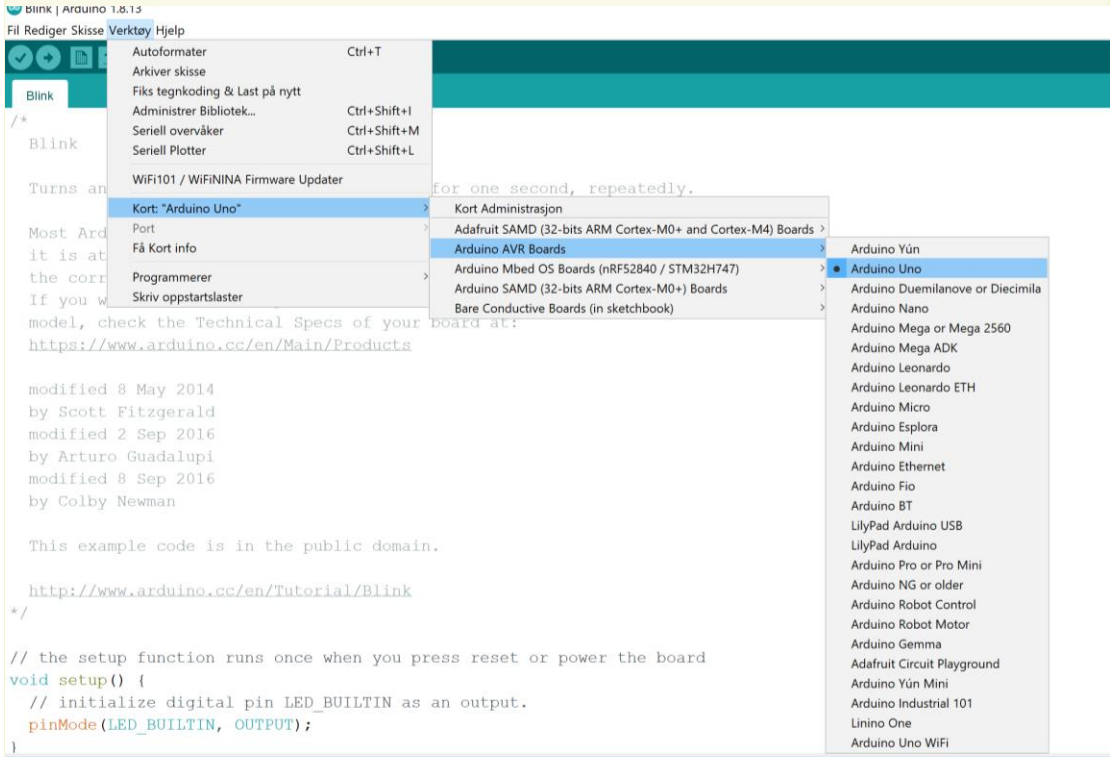
Velg Arduino-typen (vi har Arduino Uno) ved å gå til:

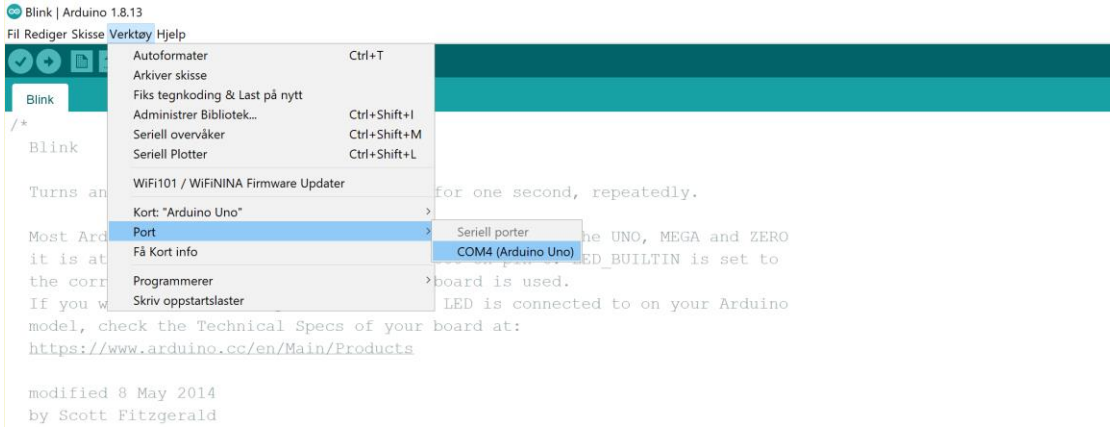
- Verktøy/Tools – Kort – Arduino AVR Boards – Arduino Uno

Velg riktig port ved å gå til

- Verktøy/Tools – Port – COMx (Arduino Uno)

NB! Arduinoen må være koblet til Pcen før du kan velge port, porten er ofte indikert med Arduino Uno i parantes





Last opp skissen til Arduinoen

- Klikk på ikonet for opplasting
- Se på det grønne og svarte meldingsfeltet nederst, der kommer status og eventuelle feilmeldinger
- Når riktig port er valgt, kan skisse lastes opp, det svarte meldingsfeltet gir en status for hvor mye minne som er brukt og hvor mye som er tilgjengelig
- Her ser vi en feilmelding som skyldes at porten ikke er valgt

Innebygde funksjoner og variabler

Gjennomgang av Blink-skissen

Grunnstrukturen til alle Arduino-skisser: setup() og loop()

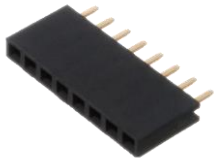
Grunnstruktur i alle skisser

```
void setup(){
// Denne funksjonen kjører en gang
// Dette er en kommentar

}
void loop(){
// Denne funksjonen kjører hele tiden
}
```

Blink-eksempelet

```
void setup()
{
pinMode(LED_BUILTIN, OUTPUT);
}


void loop()
{
digitalWrite(LED_BUILTIN, HIGH);
delay(1000);
digitalWrite(LED_BUILTIN, LOW);
delay(1000);
}
```

Velge pin med pinMode()



Kontaktene som er montert på Arduinoen heter headerpins. Derfor kalles kontaktene til Arduinoen «pin» i programmeringen

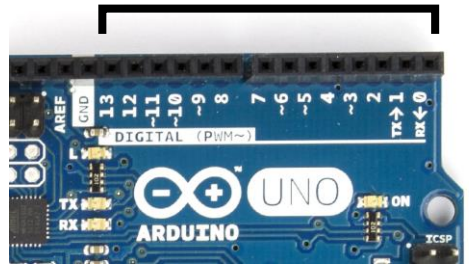
/ Kommentaren */)* på begynnelsen av skissen forklarer at den innebygde LEDen er på pin 13

pinMode(pin-nr, modus inn eller ut**)** tar imot variabler om hvilken pin som brukes, og om den skal motta input eller sende output

PinMode(pin, mode);

LED_BUILTIN pin-en som styrer den innbygde LED-en (pin13)

OUTPUT setter pin-en til å sende elektriske signaler ut (INPUT setter tilsvarende pin-en til å ta imot elektriske signaler)




```
/*
Blink
Turns an LED on for one second, then off for
one second, repeatedly.
Most Arduinos have an on-board LED you can
control. On the UNO  it is attached to
digital pin 13
*/

void setup()
{
pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
digitalWrite(LED_BUILTIN, HIGH  delay(1000);
digitalWrite(LED_BUILTIN, LOW); delay(1000);
}
```

Sende kommando til pin med digitalWrite()

digitalWrite() tar imot variabler om hvilken pin det skal sendes noe til (digitalRead() tar tilsvarende i mot variabler om hvilken pin som skal leses av)

HIGH sender strøm til pin ved å angi strømmen som på, tilsvarer 5 volt og true

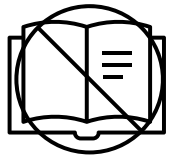
LOW slår av strøm til pin ved å angi strømmen som av, tilsvarer 0 volt og false

delay() pauser programmet, tar imot variabel i antall millisekunder, 1 sek = 1000 ms

```
void setup()
{
pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
digitalWrite(LED_BUILTIN, HIGH);
delay(1000);
digitalWrite(LED_BUILTIN, LOW);
delay(1000);
}
```

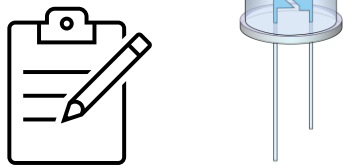
Blink-skissen

I dette eksempelet hadde vi ingen sensorer, vi gav dataene rett i programmet:



~~lese~~ *~~signaler fra brukere og omgivelser via sensorer «sanser» eller «leser»~~*

gjøre *noe med signalene programmering behandler data behandler/bearbeider*



sende *signaler ut til forskjellige komponenter*
«skriver»

Funksjonen **digitalWrite()** sender eller «skriver» ut signalet til aktuatoren (LEDen)

(digitalRead() tar tilsvarende i mot variabler om hvilken pin som skal leses av)

Utforskning: Juster millisekunder i delay()

Hvor lavt kan man sette millisekunder og fortsatt se at LEDen blinker?

Juster millisekunder ved å endre verdien og trykk på «last opp»-ikonet

Blink-eksempelet

```
void setup() {
pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
digitalWrite(LED_BUILTIN, HIGH);
delay(1000);
digitalWrite(LED_BUILTIN, LOW);
delay(1000);
}
```

Etter pausen

Bygge kretser

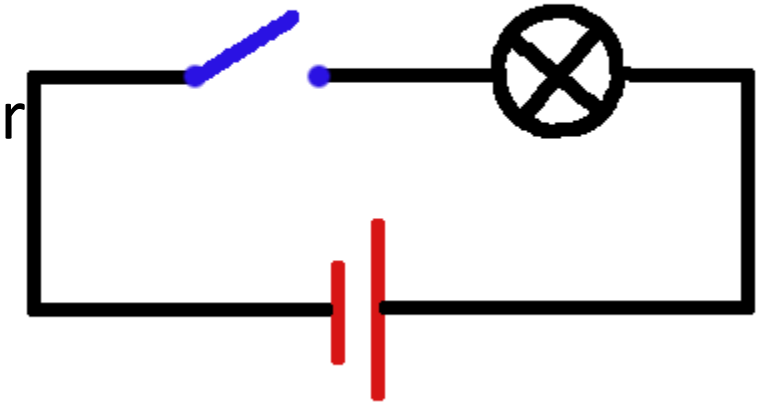
Strøm: grunnleggende prinsipper

Sikkerhet

Hva skjer ved kortslutning?

Brytere (knapper)

To vanlige problemer med brytere + løsninger



Kreditering av bilder i presentasjonen

Creative commons lisenser:

Slide 12: Love Open Source fra Valerio Bozzolan, CC0, via Wikimedia Commons

Slide 15: Arduinokunst fra

Art Photo by Taelynn Christopher on Unsplash

Korrall art https://create.arduino.cc/projecthub/febriy/rainbow-coral-interactive-art-9a6680

Slide 16: Arduinorobot fra Simonwilmot, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons

Slide 10,11,13,14, 31, 36, 38:

Arduinobrett fra Arduino005.jpg fra oomlout, CC BY-SA 2.0 <https://creativecommons.org/licenses/by-sa/2.0>, via Wikimedia Commons
Knapp fra 5 pushbuttons oomlout, CC BY-SA 2.0 <https://creativecommons.org/licenses/by-sa/2.0>, via Wikimedia Commons
ATMEGA328PU.jpg fra oomlout, CC BY-SA 2.0 <https://creativecommons.org/licenses/by-sa/2.0>, via Wikimedia Commons
Arduino backside fra DustyDingo, Public domain, via Wikimedia Commons
Arduino_ftdichip fra DustyDingo, Public domain, via Wikimedia Commons
Header pin fra oomlout, CC BY-SA 2.0 <https://creativecommons.org/licenses/by-sa/2.0>, via Wikimedia Commons

Slide 40: Animasjon av krets/ enkel krets fra MikeRun, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons

Øvrige:

Slide 5: Sonen logo fra https://ordenen.ifi.uio.no/association/sonen/

Slide 6, 9, 26: Heidi Bråthen

Slide 5: Arduino Student kit fra Arduino.cc produktbilde