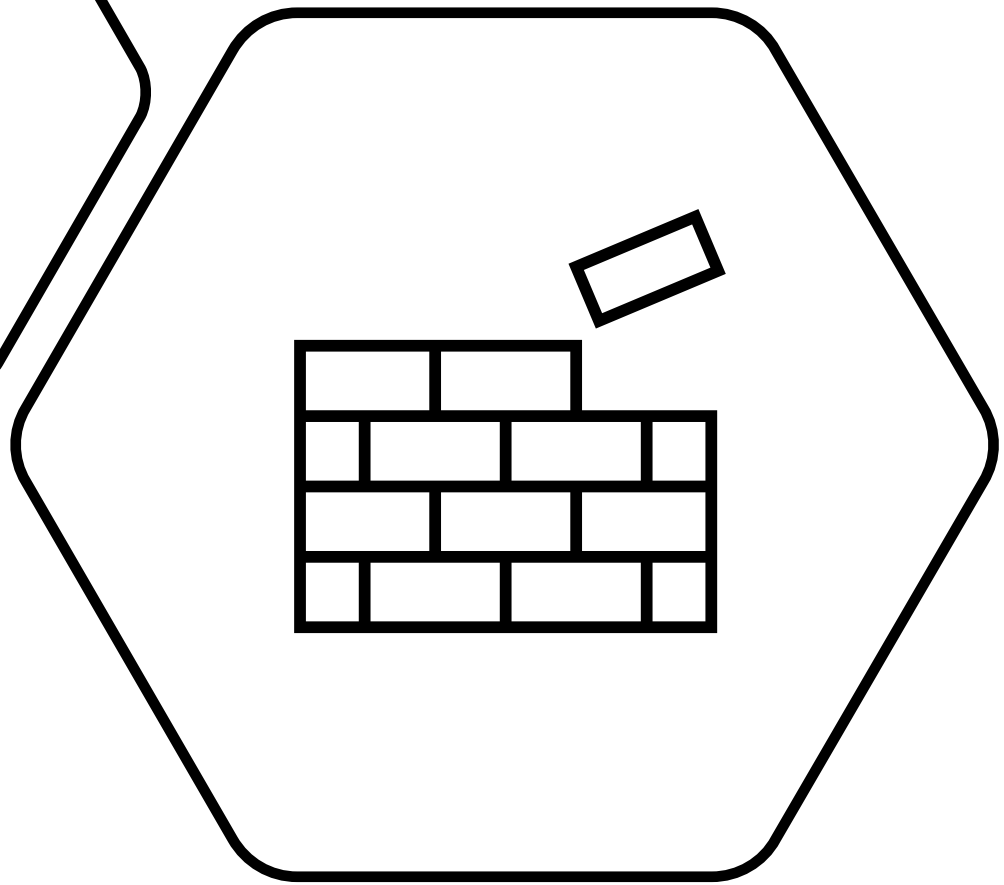


Arduino 2.time

Timing og modulering

Modulering

Ordne koden i funksjoner

Funksjoner i Arduino

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

Lage egne funksjoner

Ikke noe spesielt nøkkelord for å definere funskjoner i Arduino

Navn på funksjon: Stor forbokstav for å skille mellom variabler og funksjoner (valgfritt)

Definer funksjonen utenfor setup() og void() slik at den blir global, tilgjengelig i hele skissen

Kall funksjonen med navn

FunksjonsNavn();

```
datatype Funksjonsnavn(){
        //kode
}

FunksjonsNavn();

setup(){
}
loop(){
FunksjonsNavn();
}

datatype Funksjonsnavn(){
        //kode
}
```

Returnere verdier fra funksjoner

Funksjoner kan returnere en verdi til programmet i form av både tall, tekststrenger og verdier som inneholder tall og tekststrenger

Avslutt koden med

return verdi;

return verdi;

Funksjoner som returnerer noe må være definert med datatype

Datatypen avgjøres av hva slags verdi som skal returneres fra funksjonen

Returverdien må være av samme datatype som funksjonen

Hvis returverdien er en variabel må den være deklarert med samme datatype som funksjonen

```
datatype
Funksjonsnavn(){
    //kode
    return verdi;
}
```

Få returverdien fra en funksjon

For å få verdien fra funksjonen må vi kalle den og legge den i en variabel inne i programmet

```
datatype verdi;


verdi =
funksjonsNavn();
```

Funksjoner som ikke returnerer verdier

Erklær funksjonen som void hvis den ikke skal returnere noen verdi

```
void datatype
Funksjonsnavn(){
      //kode
}
```

Funksjoner som ikke returnerer en verdi kan utføre oppgaver i programmet, som å skrive ut tekst

```
void SkrivUt(){
      println(«Hello World!»)
}
```

setup() og loop() er definert som void

Sende inn parametre til funksjoner

funksjonsNavn(parameter){

//gjør noe med argument;

Parameter er del av funksjonens «signatur»

Deklarasjon:
datatype funksjonsNavn(parameter){
//gjør noe med argument;

Kall:
funksjonsNavn(argument);

Argument er verdien vi sender når vi kaller funksjonen

```
boolean knappTrykketPaa(int pin){
      return digitalRead(pin);
}
```

Eksempel

```
boolean utsettelse(int endring, int intervall){
if((millis() - endring) >= intervall){
      return true;}
}
```

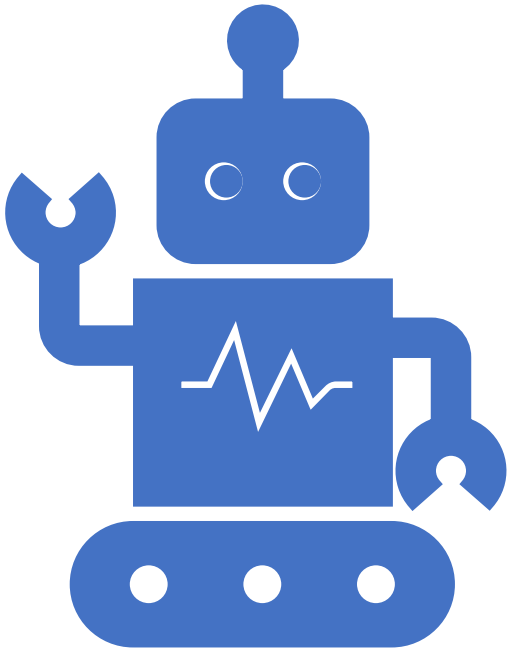
Når er det greit å kopiere kode?

Arduino er open source og mye Arduinokode er frigitt under ulike creative commons lisenser som betyr at vi har lov til å bruke det til våre egne prosjekter

Bruk alle eksemplene i Arduino IDE!

Når vi benytter andres arbeid er det viktig å opplyse om det og kreditere opphavspersonen

Benytt kommentarlinjene på toppen av skissen til å opplyse om hvor ikke-originale deler av koden er hentet fra



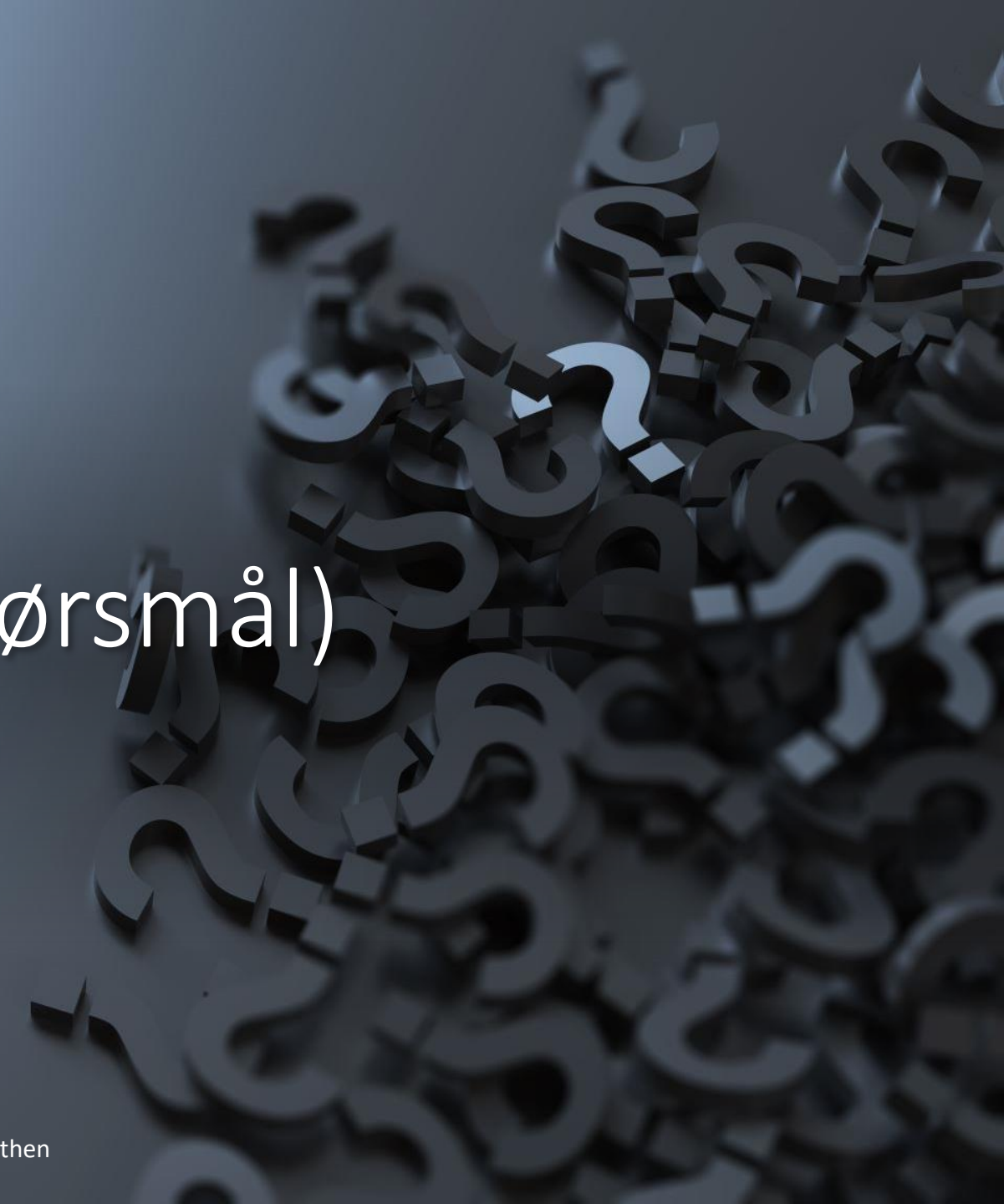
UiO GPT og Microsoft Copilot (kun Bing Chat Enterprise)

Mer om kreativ bruk av AI-tjenester for kode neste uke

Kort intro i dag:

- Hva er lov? – Å bruke tjenester som UiO har avtale med
- Hva er juks? – Å fremstille arbeid som sitt eget når det ikke er det
- Hva er smart? – Å bruke hodet selv! Få hjelp med prosessen heller enn å hoppe til et svar

(Menti-spørsmål)



Jobbe med timing

millis() og delay()

Jobbe med timing: delay()

delay()

Pauser programmet i x antall millisekunder

delay() pauser hele programmet og Arduinoen i det tidsrommet

+ Enkel i bruk

- Kan gi problemer i enkelte situasjoner

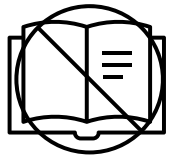
    for eksempel når Arduinoen egentlig skal lese av flere sensorer samtidig

Eksempel på problem som kan oppstå

- Kontrollere millisekunder i delay(ms) i Blinkskissen med en ekstern sensor

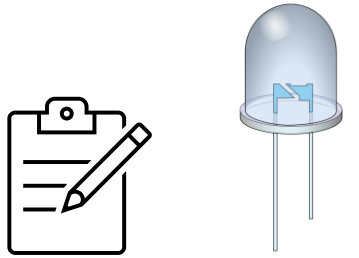
(repetisjon) Blink-skissen

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)

(repetisjon?) Utforskning: Juster millisekunder i delay()

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

Juster millisekunder i skissen 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);
}
```

Justere millisekunder med en lyssensor

https://www.tinkercad.com/things/jlTLKT0361l-blink-med-lyssensor?sharecode=WBAAV7cXnCPy2CO6GwxzlMBu5Zb2p4bzJe-6gI1zYiA

Eksempel på bruk av funksjoner for tid: *Debounce*

Debounce = sprettende input

vanlig problem med knapper

Switchbounce

Når vi trykker ned knappen tar det noen millisekunder før metallbitene kommer i stabil kontakt med hverandre

Fordi loop() kjører veldig fort klarer Arduinoen å registere flere knappetrykk idet vi trykker ned knappen

Løsning: Debounce

Vi kan programmere Arduinoen til å vente noen millisekunder og dermed kun registrere ett av trykkene

(repetisjon) Debounce med delay()

delay()

pauser programmet i antall millisekunder

`delay(millisekunder)`

Prøv det selv (senere):

- Trykk på knappen, må du holde inn en liten stund for å få LEDen til å lyse stabilt?
- Hva skjer når du trykker knappen raskt inn og slipper?
- Se på seriell overvåker mens du trykker ned knappen

```
*/ Eksempelskisse: Button

const int buttonPin = 2;
const int ledPin =  13;
int buttonState = 0;

void setup() {
pinMode(ledPin, OUTPUT);
pinMode(buttonPin, INPUT);
}

void loop() {
 buttonState = digitalRead(buttonPin);
if (buttonState == HIGH) {
          delay(500);
          digitalWrite(ledPin, HIGH);
          } else {
                    delay(500);
                    digitalWrite(ledPin, LOW);
          }
}
```

Flere funksjoner for å jobbe med tid

Arduino Language reference:

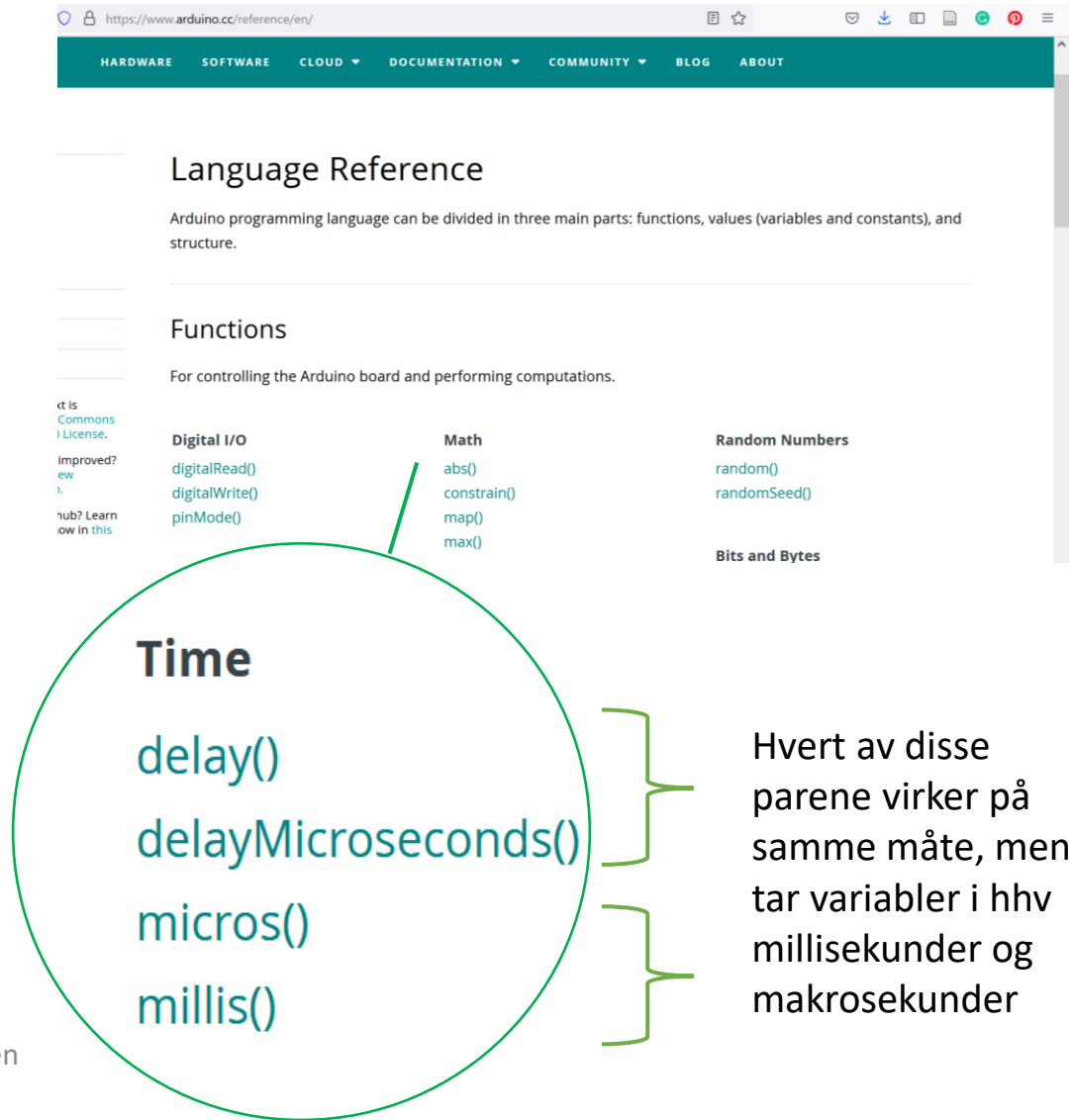
https://www.arduino.cc/reference/en/

https://www.arduino.cc/reference/en/language/functions/time/millis/

Funksjoner for tid

delay()

millis()



Hvert av disse parene virker på samme måte, men tar variabler i hhv millisekunder og makrosekunder

Jobbe med timing: millis()

Fordel hvis Arduinoen skal gjøre andre oppgaver samtidig

Måler tiden som har gått siden Arduinoprogrammet startet

starter når det blir lastet opp

starter på nytt når man trykker på reset-knappen

tid = millis() //tar ingen variabler, gir bare tiden

NB! Bruk unsigned long når du jobber med tid, plassproblemer ved int

Skriv millis() til serial overvåker

Vi kan se verdien til millis() ved å skrive den til seriell overvåker

```
void setup() {
Serial.begin(9600);
}

void loop() {
Serial.println(millis());
}
```

(repetisjon) Serial: et verktøy for problemløsning (rep.)

Arduinoen har et verktøy som lar oss se signalene som går ut og inn i form av tekst

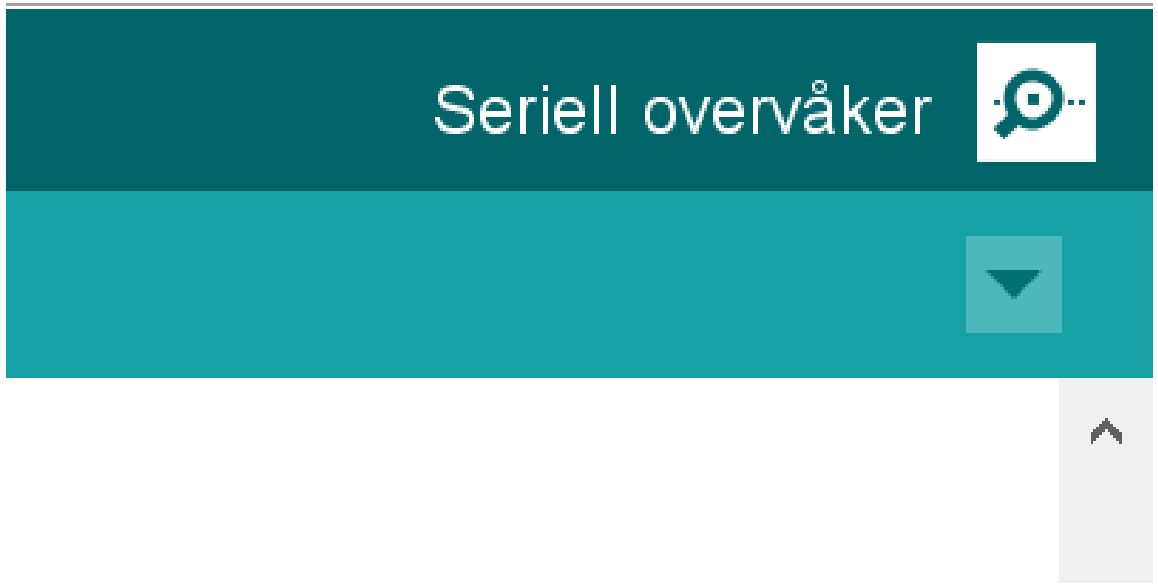
Gå til:

Verktøy – seriell overvåker eller ctrl+shift+m eller ikon øverst til høyre

Vises som popup-vindu (integrert i 2.0)

Seriell overvåker (serial monitor)

Skriver ut returverdier fra funksjonene når programmet kjører i form av tall eller tekst



(repetisjon) Funksjoner for seriell kommunikasjon #1

For å starte seriell kommunikasjon må vi initialisere den med

Serial.begin(9600)

Serial.begin()-funksjonen tar inn en variabel for baud rate.

Baud rate refererer til hastigheten på kommunikasjonen over en datakanal. På Arduino er det vanlig å sette den til 9600.

Serial.print()- funksjonen lar oss skrive ut beskjeder til overvåkeren og er nyttig for å følge med på hva som egentlig skjer i programmet

```
DigitalReadSerial eksempelskisse

int pushButton = 2;

void setup() {
Serial.begin(9600); // start seriell
kommunikasjon ved 9600 bits per sekund
pinMode(pushButton, INPUT);
}

void loop() {
int buttonState =
digitalRead(pushButton);
Serial.println(buttonState); //skriv ut
tilstanden til knappen
  delay(1); //pause mellom avlesningene
for stabilitet
}
```

Debounce med millis() i stedet for delay()

Fordel hvis Arduinoen skal gjøre andre oppgaver samtidig som man trykker på knapper

millis(): **Har det gått lang nok tid fra noe skjedde til at vi kan gjøre noe mer?**

```
if((millis() - tidForForrigeEndring) >= intervallForUtsettelse){
if(knappTilstand == HIGH){
digitalWrite(led, !ledTilstand); //skifter tilstanden på LEDen
  tidForForrigeEndring = millis(); //setter starttid for telling på nytt
}
}
```

Debounce med millis()

millis() måler millisekunder siden Arduinoen startet. Det kan vi bruke til å måle tid.

- millisekunder

millis() kort oppsummert

Har det gått lang nok tid fra noe skjedde til at vi kan gjøre noe mer?

Vi kan hente ut hvor lenge programmet har gått akkurat nå ved å se

unsigned long akkuratNå = millis();

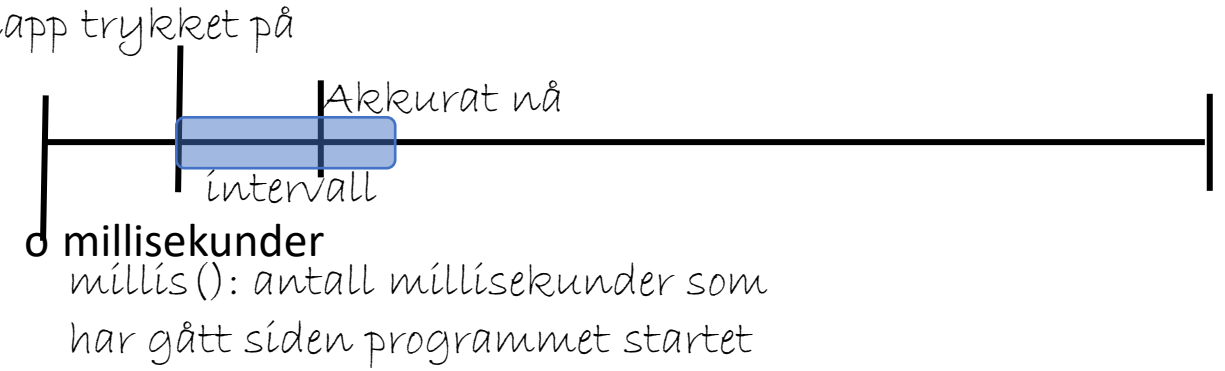
Debounce med millis()

Vi kan bestemme et intervall

(f.eks hvor lenge LED skal lyse i blink eller hvor lenge programmet skal vente for å avgjøre om det er ett eller flere knappetrykk ved debounce)

Int intervall = 50;

Debounce med millis()



Vi kan måle hvor lang tid siden det gikk siden knappen først «endret tilstand», altså gikk fra å ikke bli trykket ned til å bli trykket ned

1: Lagre tidspunktet da knappen først ble trykket ned, for eksempel

float tidspunktTrykket;

2: Regne ut hvor lang tid som har gått siden da

Tid som har gått er lik millis() – tidspunkt trykket

3: Sjekke om det har gått mer tid enn det intervallet vi bestemte

If(tid som har gått > intervall)

//kan programmet registrere et nytt trykk fra knappen

debounce med millis()

```
const int knapp = 2; //knapp pin 2
const int led = 13;
int intervallForUtsettelse = 50; //intervall
int ledTilstand = LOW;
unsigned long tidForForrigeEndring = 0;
unsigned long tidAkkuratNaa;

void setup() {
pinMode(led, OUTPUT);
pinMode(knapp, INPUT_PULLUP);
ledTilstand = LOW;
Serial.begin(9600);
}
```

```
void loop() {
tidAkkuratNaa = millis(); //her kunne vi kanskje bare brukt millis();
ledTilstand = digitalRead(led);
int knappTilstand = digitalRead(knapp);
if((millis() - tidForForrigeEndring) >= intervallForUtsettelse){
if(knappTilstand == HIGH){
digitalWrite(led, !ledTilstand);
  tidForForrigeEndring = tidAkkuratNaa(); //altså, er lik millis();
}
}
}
```

Bruk denne eksempelkoden!

Bruk eksempelkoden fra forelesningen med referanse, og kommenter med dine egne ord hva som skjer

Se også eksempelskissen i Arduino IDE: Blink without delay

Eksempelskisse: Blink without delay

```
 https://www.arduino.cc/en/Tutorial/BuiltInExamples/BlinkWithoutDelay
*/

// constants won't change. Used here to set a pin number:
const int ledPin = LED_BUILTIN;  // the number of the LED pin

// Variables will change:
int ledState = LOW;  // ledState used to set the LED

// Generally, you should use "unsigned long" for variables that hold time
// The value will quickly become too large for an int to store
unsigned long previousMillis = 0;  // will store last time LED was updated

// constants won't change:
const long interval = 1000;  // interval at which to blink (milliseconds)

void setup() {
  // set the digital pin as output:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // here is where you'd put code that needs to be running all the time.

  // check to see if it's time to blink the LED; that is, if the difference
  // between the current time and last time you blinked the LED is bigger than
  // the interval at which you want to blink the LED.
```

```
  https://www.arduino.cc/en/Tutorial/BuiltInExamples/BlinkWithoutDelay

  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;

    // if the LED is off turn it on and vice-versa:
    if (ledState == LOW) {
      ledState = HIGH;
    } else {
      ledState = LOW;
    }

    // set the LED with the ledState of the variable:
    digitalWrite(ledPin, ledState);
  }
}

```

Negasjon

! !=

(repetisjon: negasjon i skissen)
Debounce med millis() i stedet for delay()

Fordel hvis Arduinoen skal gjøre andre oppgaver samtidig som man trykker på knapper

millis(): **Har det gått lang nok tid fra noe skjedde til at vi kan gjøre noe mer?**

```
if((millis() - tidForForrigeEndring) >= intervallForUtsettelse){
if(knappTilstand == HIGH){
digitalWrite(led, !ledTilstand); //skifter tilstanden på LEDen
  tidForForrigeEndring = millis(); //setter starttid for telling på nytt
}
}
```

Bruk av negasjon:
!= Not equal to og ! boolsk ikke

!= Er ikke lik

Operatør som sammenligner en variabel med en annen variabel eller verdi og returnerer true hvis de ikke er like

```
if (x != y) { // tester om x ikke er lik y
  // gjør noe dersom de ikke er like
}
```

! Ikke

Boolsk operatør som tester om ev variabel er false

Hvis variabelen er false, returnerer ! True

```
if (!x) { // hvis x ikke er true
  // gjør noe
}
```

Kan også brukes til å sette den bolske verdien av en variabel til det motsatte av hva den er:

```
x = !y; // x får motsatt Verdi av det y har
```

Arduinoprogrammering

String()

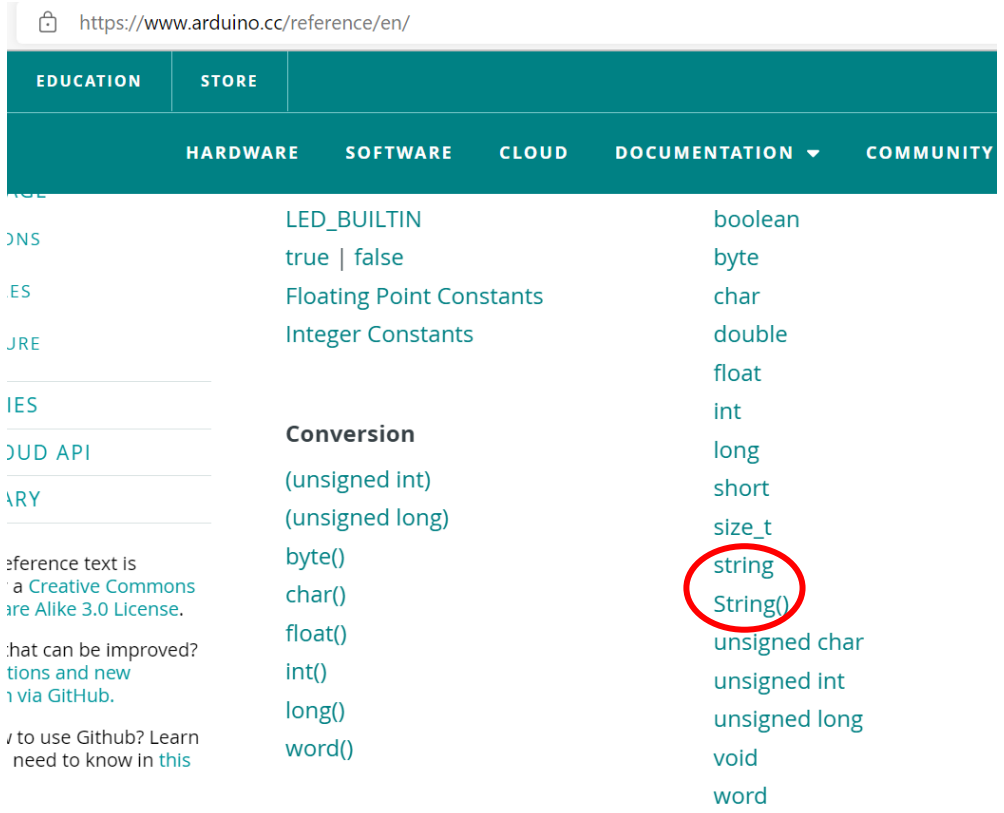
String()

Arduino har et eget String()-objekt

I Arduino language reference finner vi både String og String() under variabler (datatyper)

string er C-programmering. I C må man lage hver string som en array med forhåndsdefinert størrelse

String() er Arduinos innebygde String-objekt med en rekke funksjoner vi kan bruke for å lage, manipulere og sjekke strenger, slik vi er vant til fra Python og Java



Lage et String()-objekt i Arduino

1: Lage en variabel med String som datatype

```
String variabelNavn = «Dette er en streng»;
```

2: Bruke String-klassens konstruktørmetode og konvertere variabel til String-objekt:

```
String variabelNavn = String(‘a’); //et tegn
String variabelNavn = String(«En streng»); //en streng av tegn
String variabelNavn = String(13); //integer – String() kan lages med datatyper som int, float og long
String variabelNavn = String(variabelNavn + " mer");
// konkatenerer to strenger
```

Bruke strenger

```
String variabelNavn = «Dette er en streng»;
variabelNavn = «her skriver vi noe nytt»;
```

String()-funksjoner for tekst og tegn

Operatører i String()

+= (append) legger til data på slutten av strengen
Legger sammen to strenger på samme måte som concat()

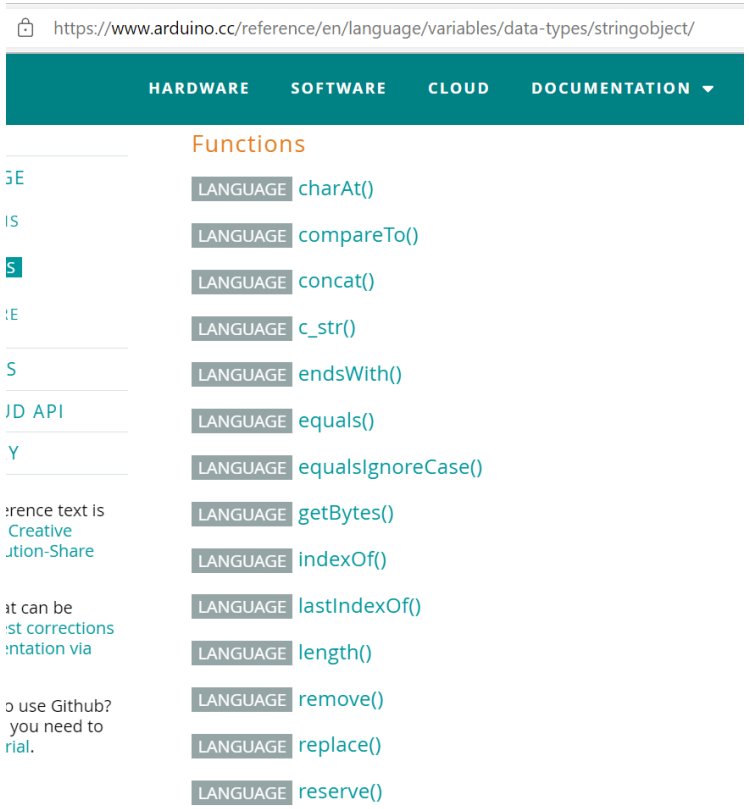
Funksjoner i String()

equals() sammenligner strenger

```
String streng1 = «tekst»;
String streng2 = «Tekst»;
If(streng1.equals(streng2){
Serial.println(«streng1 er lik streng 2»);
```

Ignorer forskjeller i store og små bokstaver med equalsIgnoreCase

```
Streng1.equalsIgnoreCase(streng2);
```



String()-funksjoner for tekst og tegn

substring(from, to) får tak i del av en streng

streng1.substring(from); //fra indeks

streng1.substring(from, to); //indeksen *etter* substringen slutter

delAvStreng = streng1.substring(3, 7);

NB! Det er smart å sjekke lengden på strengen før man prøver å hente verdier fra indekser

```
If(lengde >= to){
//hent substring
}
```

length() sjekker lengden på en streng

streng1.length();

Int lengde = streng1.length();

Løkker og arrays

Løkker og arrays

Løkker

Løkker lar oss repetere kodeblokker til en bestemt betingelse blir møtt eller et bestemt antall ganger

Forenkler repeterende oppgaver

Færre kodelinjer å skrive og lese

Bra for å behandle eller gå gjennom mange verdier

Tre typer løkker i Arduino

while-løkker

do-while-løkker

for-løkker

While-løkker

Utfører kodeblokken inne i klemmene så lenge (while) betingelsen som sjekkes er sann/true.

Sjekker ofte en variabel som oppdateres inne i kodeblokken for å kontrollere nøyaktig hvor lenge kodeblokken repeteres

Kan også sjekke f.eks. sensor-input

NB! Hvis ingenting endres og betingelsen aldri blir usann/false vil man få en evig løkke

Syntaks

```
while(betingelse){
        //gjør noe
}
```

Eksempel

```
void loop(){
variabel = 0;
// gjør dette 200 ganger:
while (variabel < 200) {
//tar verdien i variabelen og legger til 1
variabel++;
}
```

do-while-løkker

Kjører kodeblokken først og sjekker deretter om betingelsen er sann

Det sikrer at kodeblokken kommer til å kjøre minst en gang

Syntaks

```
do {
// statement block }
while (condition);
```

Eksempel

```
int x = 0;
do {
delay(50); // wait for sensors to stabilize
x = readSensors(); // check the sensors
}
while (x < 100);
```

for-løkker

«alt-i-ett-løkke»

Sjekker betingelser og oppdaterer variabler i betingelsen

Kan være mer oversiktlig å bruke enn while og do-while-løkkene

Syntaks

```
for (initialization; condition; increment) {
// gjør noe; }
```

Eksempel: Dimme en LED med en løkke

```
int ledPin = 10;
void setup() {
pinMode(ledPin, OUTPUT);
}
void loop() {
//(255 er maks verdi for analog ut-pin)
for (int i = 0; i <= 255; i++) {
analogWrite(ledPin, i);
delay(10);
}
}
```

De tre parametrene til for-løkker

Syntaks

```
for (initialization; condition; increment) {
// gjør noe;
}
```

Eksempel

```
for (int i = 0; i <= 255; i++) {
// gjør noe:
}
```

Parametere

Initialization utføres kun en gang

Condition betingelsen testes. Hvis den er true går programmet videre og utfører increment og resten av kodeblokken, hvis den er false stopper den og går ut av løkken her

Increment oppdaterer verdien før kodeblokken kjører igjen

Arrays

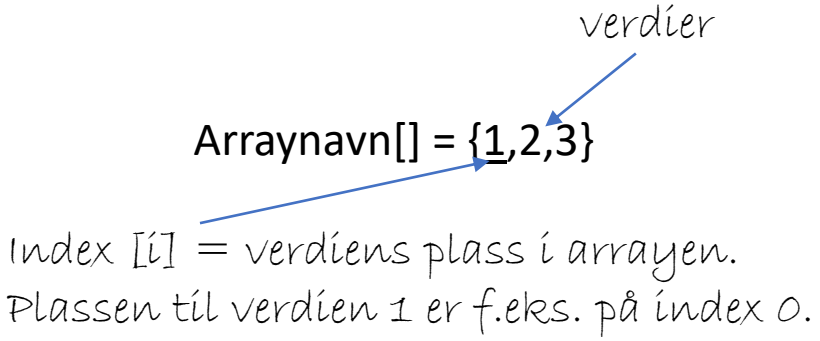
Arrays lar oss samle mange verdier under ett variabelnavn

Forenkler håndtering av store mengder data

Syntaks (flere muligheter for å deklarere arrays)

```
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[3] = {2, 4, -8};
char message[6] = "hello";
```

Arrays



Legg til verdi i array

arrayNavn[0] = 11;

Hent en verdi fra en array

verdi = arrayNavn[0]; //verdien er 11

NB! Vær oppmerksom på at det å hente verdier i array-plasser som ikke finnes gjør at Arduinoen henter ut info fra andre steder i minnet. Dette kan føre til feil som er vanskelige å feilsøke.

For eksempel:

Arraynavn[] = {1,2,3}

I denne arrayen er det 3 plasser, men det høyeste index-tallet er 2 fordi indexen begynner på 0!

Hvis vi prøver å hente ut

Verdi = arrayNavn[3];

vil Arduinoen hente ut informasjon fra et annet sted i minnet, fordi arrayNavn[3] ikke finnes.

Arraynavn[] = {1,2,3}

verdi	**1**	**2**	**3**
index	0	1	2

Arrays og løkker

For at løkker skal kunne gå gjennom mange ulike verdier må verdiene være samlet på ett sted

Arrays lar oss samle mange verdier under ett variabelnavn

Arrays kan manipuleres i for-løkker ved å bruke *counter* som *index* for hver verdi i arrayen

Eksempel: skrive ut alle verdiene i arrayen myPins[]

```
for (byte i = 0; i < 5; i = i + 1) {
Serial.println(myPins[i]);
}
```

Array eksempel: deklarere mange ledPins

Deklarere flere ledPiner:

```
int led1 = 2;
int led2 = 4;
int led3 = 8;
int led4 = 9;
int led5 = 10;
int led6 = 13;
```

Deklarere ledPin'er med array:

```
int ledPins[] = {2,4,8,9,10,13};
```

Array og for-løkke eksempel: sette alle ledPinene som OUTPUT

Sette pinMode()

```
void setup() {
pinMode(led1, OUTPUT);
pinMode(led2, OUTPUT);
pinMode(led3, OUTPUT);
pinMode(led4, OUTPUT);
pinMode(led5, OUTPUT);
pinMode(led6, OUTPUT);
}
```

Sette pinMode() med for-løkke og array

```
void setup() {
  for(int i=0; i < 6; i++){
    pinMode(led[i], OUTPUT);
  }
```

Slå på alle LEDene med for-løkke og array

```
void loop(){
digitalWrite(led1, HIGH);
delay(1000);
digitalWrite(led2, HIGH);
delay(1000);
digitalWrite(led3, HIGH);
delay(1000);
digitalWrite(led4, HIGH);
delay(1000);
digitalWrite(led5, HIGH);
delay(1000);
digitalWrite(led6, HIGH);
delay(1000);
}
```

Med for-løkke og array

```
void loop(){
for(int i = 0; i < 6; i++){
        digitalWrite(led[i], HIGH);
}
delay(1000);
}
```

Til neste gang

Oblig 1 har frist i fredag 13/2 kl. 10.00

Kom på orakeltimen i morgen klokken 10.15 for å jobbe med obligen

Takk for i dag!