“C FOR ARDUINO”

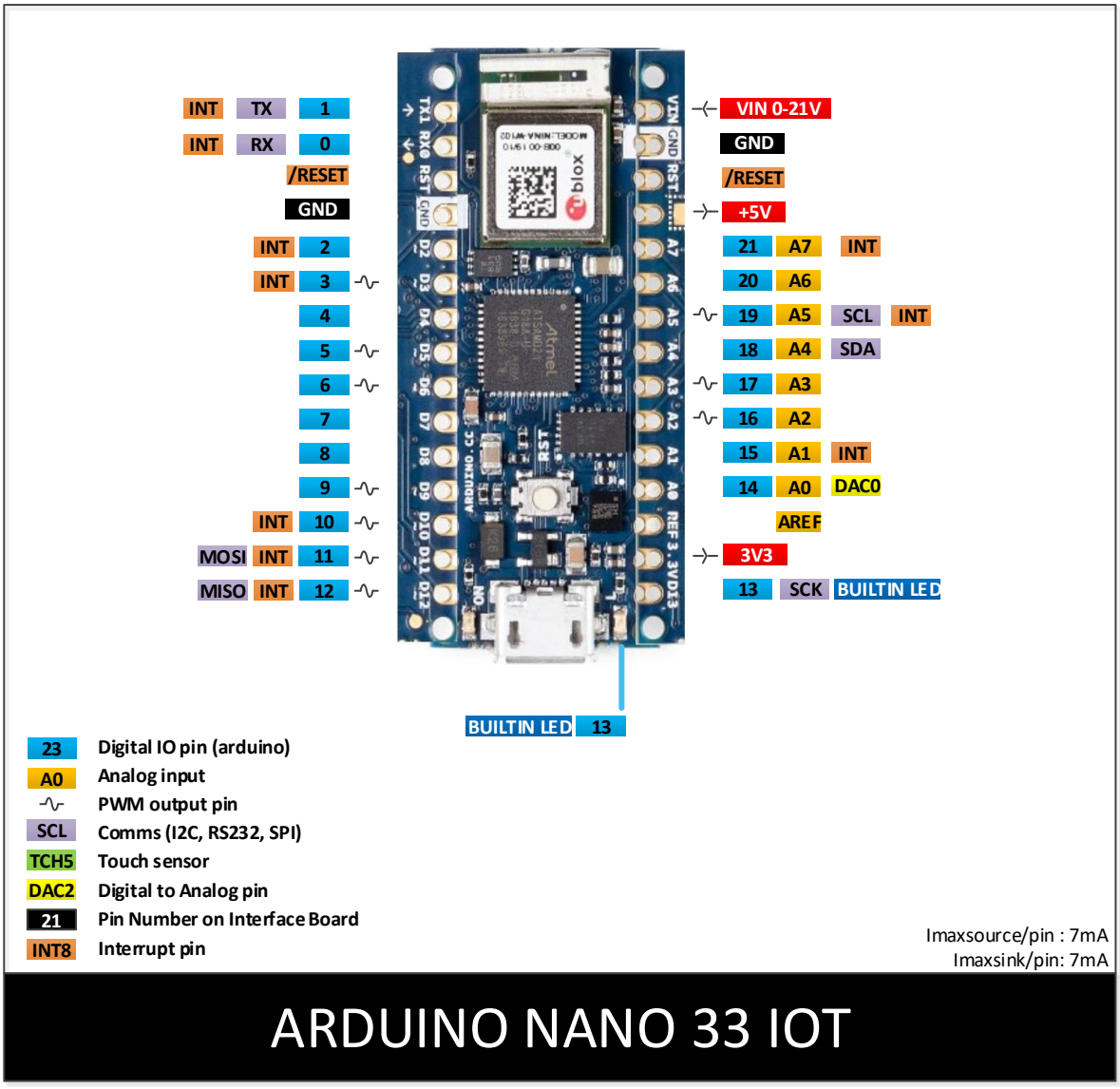
VERSIE VOOR LEERLINGEN

B. Huyskens
FEB 2023

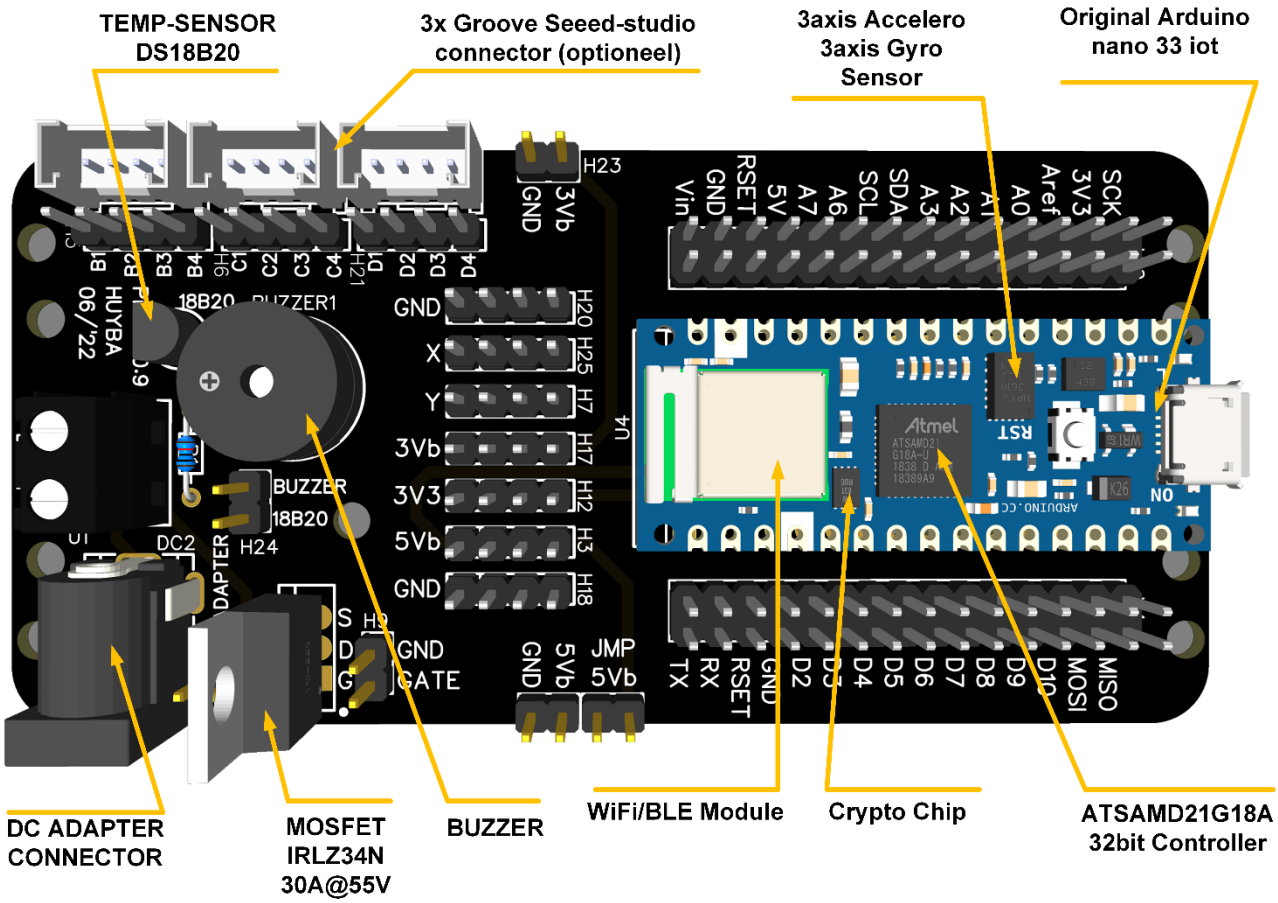
Gebruikte hardware

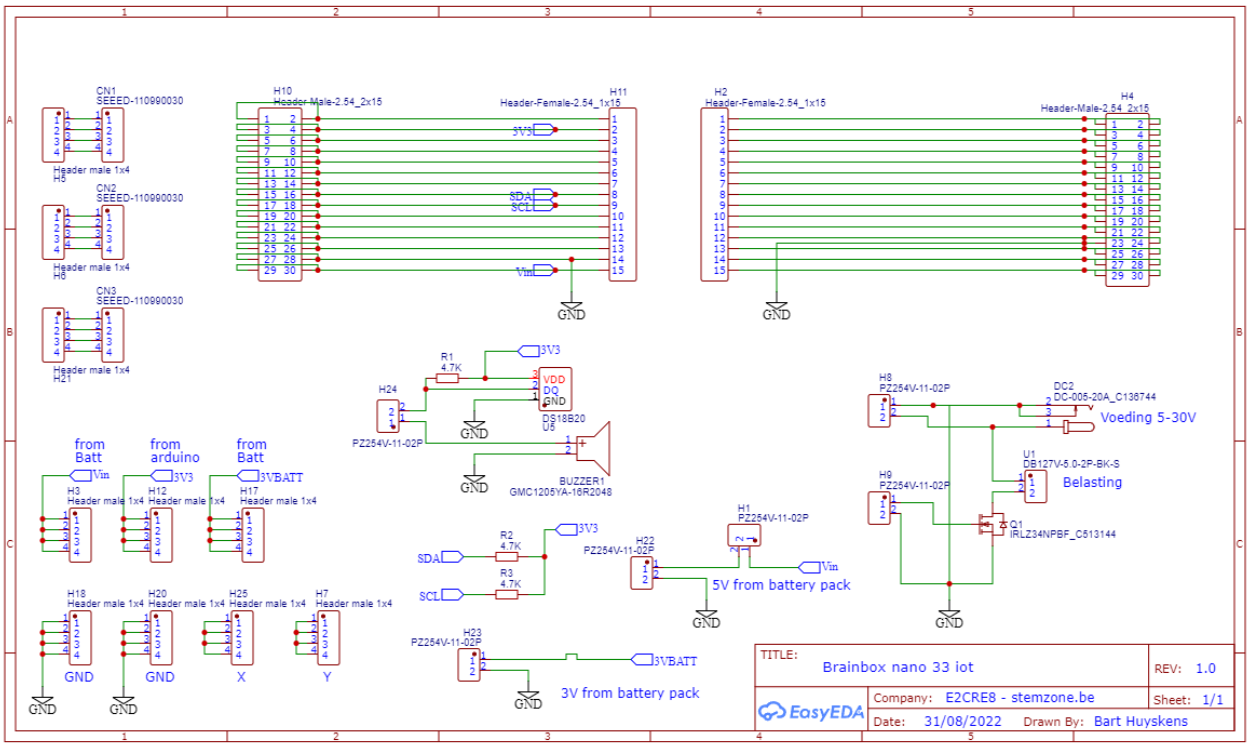
Brainbox nano 33 iot

Printplaat met connectoren, sensor en actuator—reeds gesoldeerd, Arduino nano 33 iot, USB kabel, jumper-wires.

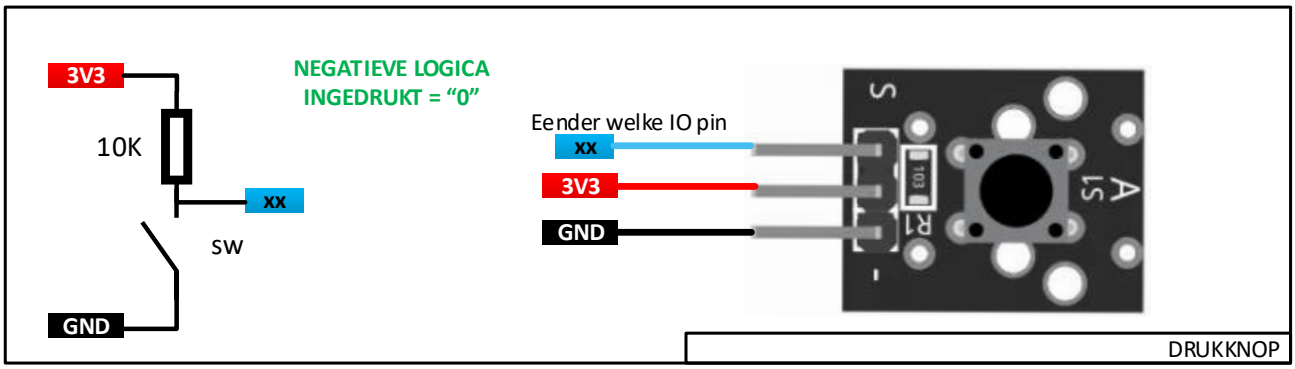


Brainbox nano 33 iot

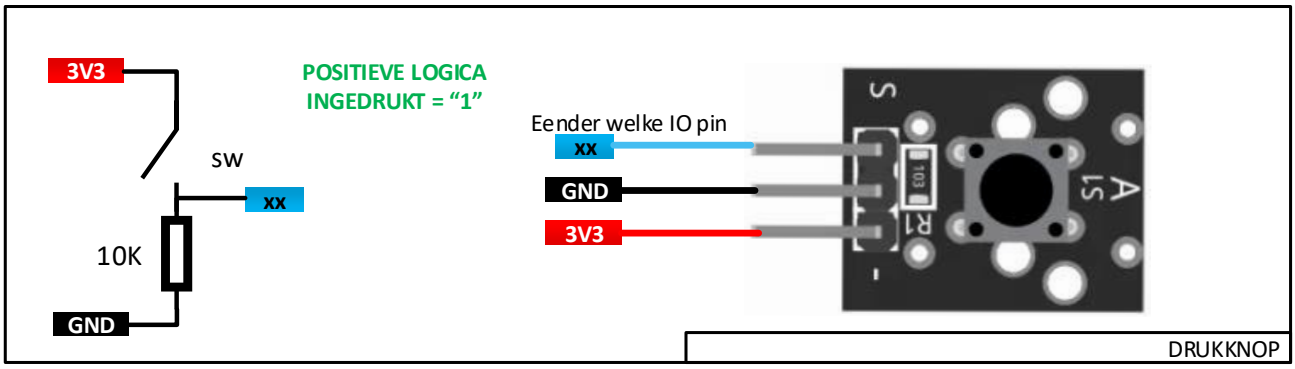




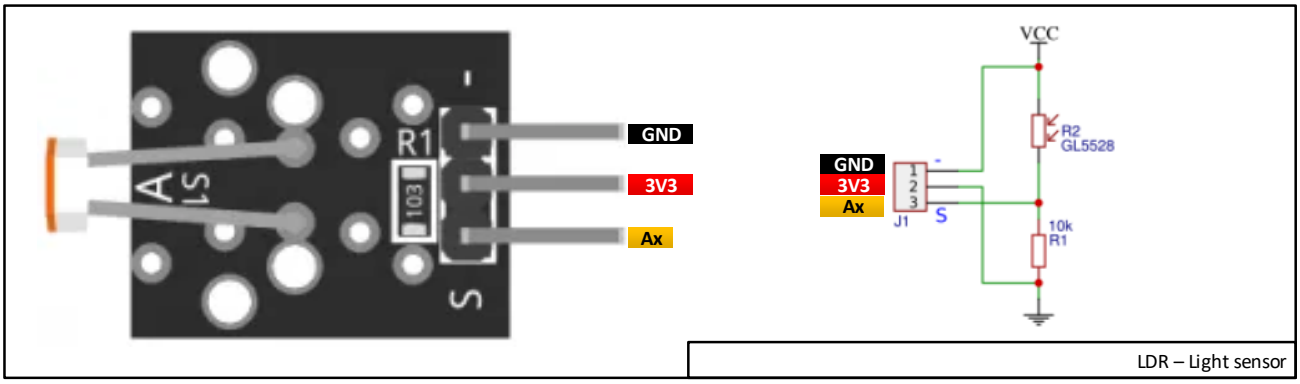
Sensoren



DRUKKNOP

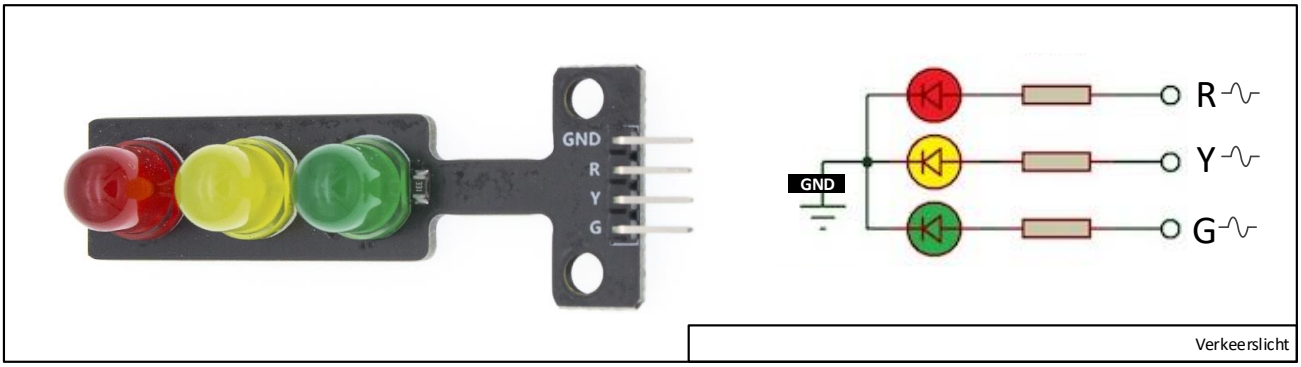


DRUKKNOP



LDR – Light sensor

Actuatoren



Verkeerslicht

Arduino IDE 2.0 installeren

Voor Windows, Mac en Linux geven we de voorkeur aan de offline (gedownloade en geïnstalleerde) nieuwe Arduino IDE 2.0. De installatie-instructie kan u via deze link raadplegen.

https://docs.arduino.cc/software/ide-v2/tutorials/getting-started/ide-v2-downloading-and-installing

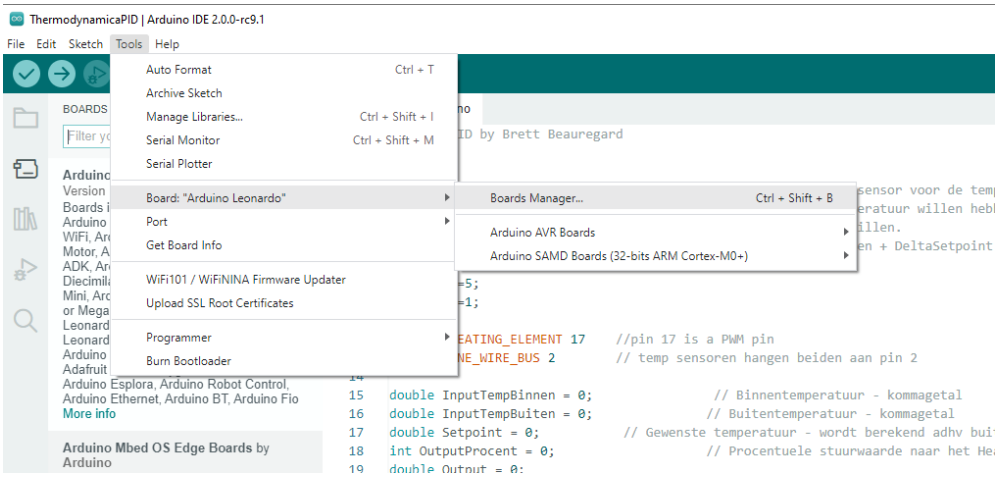
Voor Chromebook bestaat er online versie. De installatie-instructie kan u via deze link raadplegen.

https://support.arduino.cc/hc/en-us/articles/360016495639-Use-Arduino-with-Chromebook

SAMD Hardware driver activeren

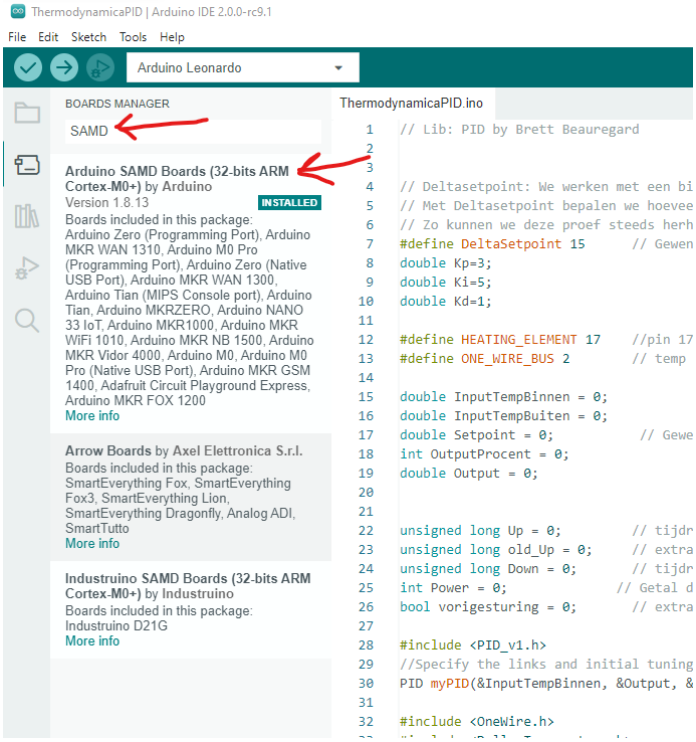
De Arduino nano 33 iot processor die we gebruiken is van de SAMD familie. We moeten hiervoor de SAMD driver installeren.

Klik op ‘tools’ -> ‘Board’ -> ‘Board manager’

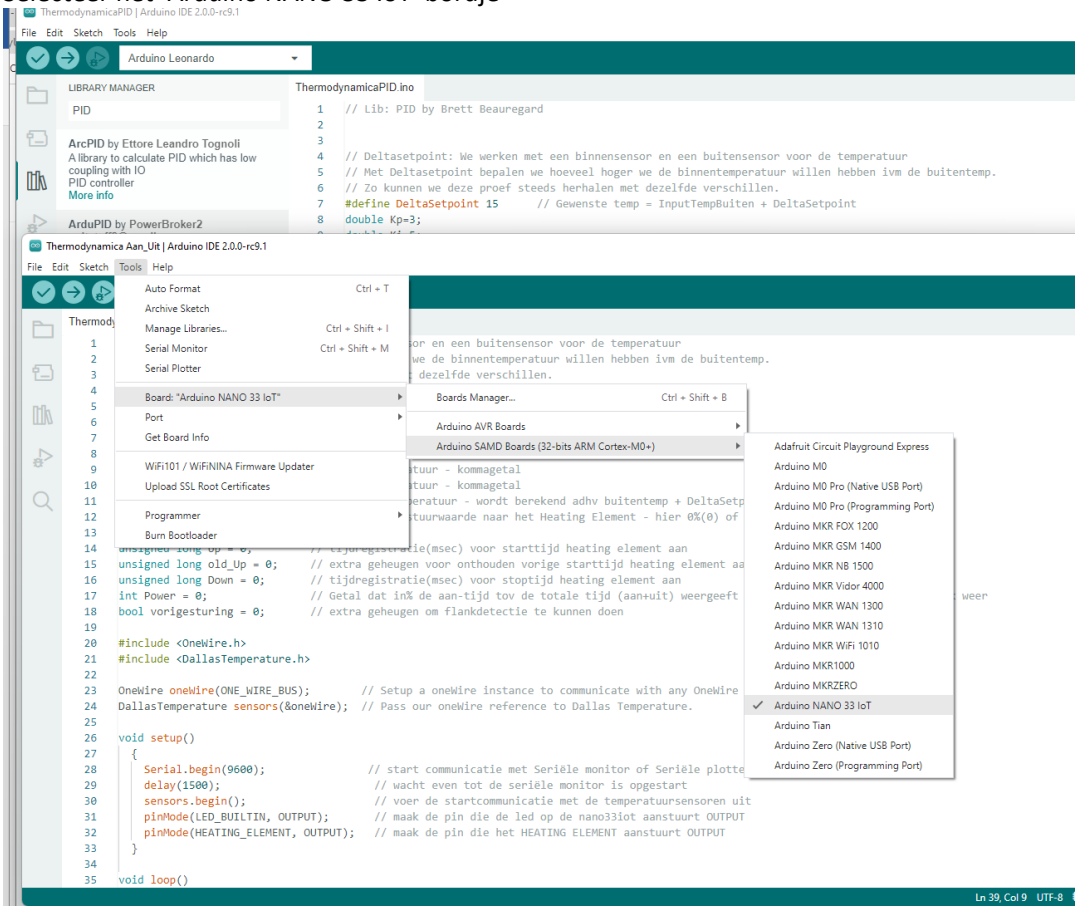


Typ bovenaan de letters ‘SAMD’ om de board te filteren en selecteer de ‘Arduino SAMD Boards….’ Om te installeren.

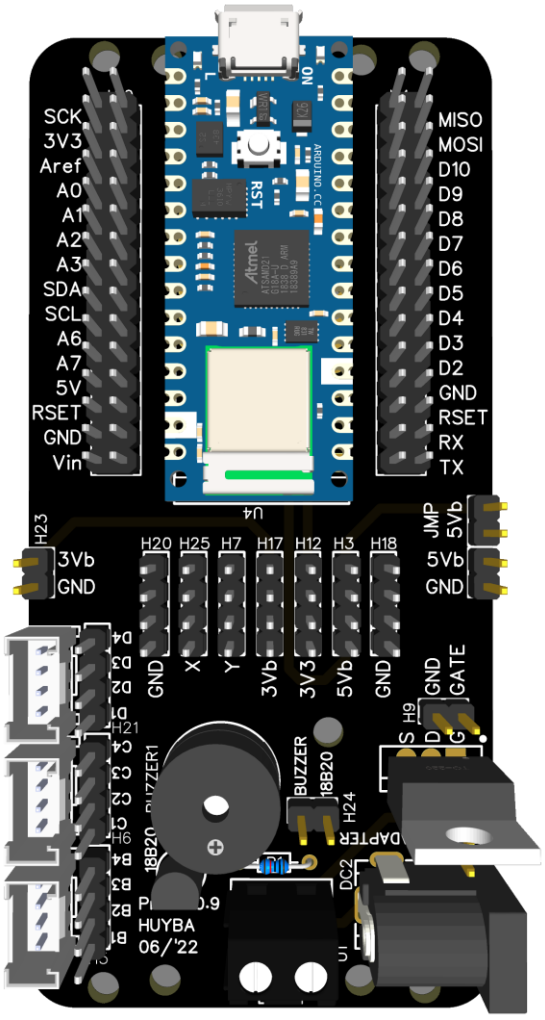
Na installatie staat er bij deze boards ‘INSTALLED’



Selecteer het ‘Arduino NANO 33 IoT’ bordje

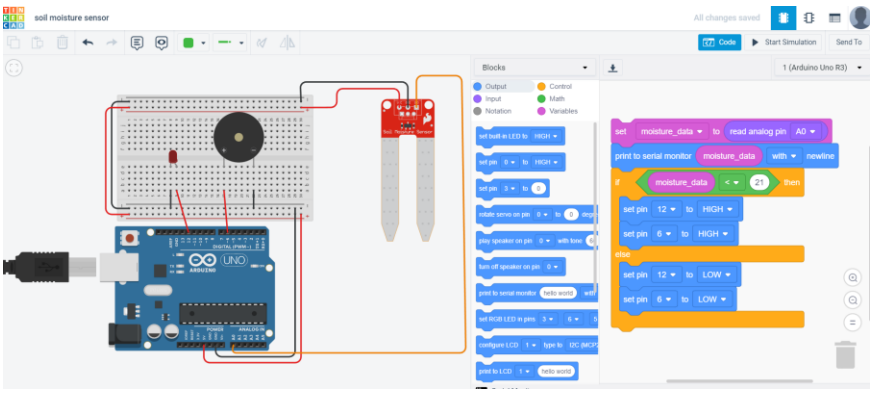


Leeg Aansluitschema

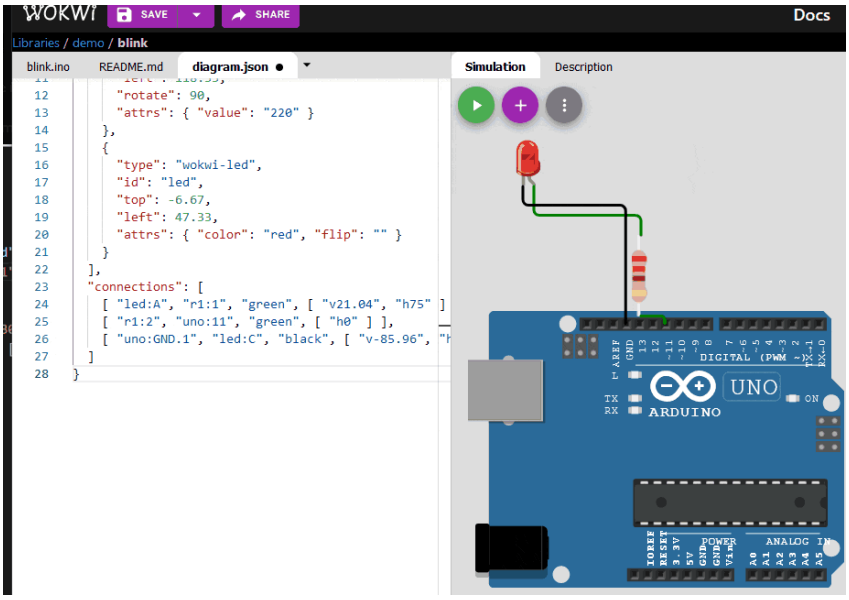


Arduino Simulator

Tinkercad



Wokwi



Arduino Cheatsheet

Deze ‘spiekbrief’ verzamelt de meeste Arduino IDE instructies en datatypes op 1 blad, samen met de pinout van de Brainbox AVR. Deze tool mag steeds gebruikt worden, ook tijdens testen en examens.

Deze CheatSheet is te downloaden van www.e2cre8.be

Arduino BBnano33 Programming CheatSheet

Primary source: Arduino Language Reference
https://arduino.cc/en/Reference/

WWW.E2CRE8.BE >> BBiot

Structure & Flow

Basic Program Structure
```
void setup() {
  // Runs once when sketch starts
}
void loop() {
  // Runs repeatedly
}
```

Control Structures
```
if (x < 5) { ... } else { ... }
while (x < 5) { ... }
for (int i = 0; i < 10; i++) { ... }
break;     // Exit a loop immediately
continue;  // Go to next iteration
switch (var) {
  case 1:
    ...
    break;
  case 2:
    ...
    break;
  default:
    ...
}
return x;  // x must match return type
return;    // For void return type
```

Function Definitions
```
<ret. type> <name>(<params>) { ... }
e.g. int double(int x) {return x*2;}
```

Operators

General Operators

=	assignment		
+	add	-	subtract
*	multiply	/	divide
%	modulo		
==	equal to	!=	not equal to
<	less than	>	greater than
<=	less than or equal to		
>=	greater than or equal to		
&&	and	||	or
!	not		

Compound Operators

++	increment
--	decrement
+=	compound addition
-=	compound subtraction
*=	compound multiplication
/=	compound division
&=	compound bitwise and
|=	compound bitwise or

Bitwise Operators

&	bitwise and	|	bitwise or
^	bitwise xor	~	bitwise not
<<	shift left	>>	shift right

Pointer Access

&	reference: get a pointer
*	dereference: follow a pointer

Built-in Functions

Pin Input/Output
```
Digital I/O - pins 0-13 A0-A5
  pinMode(pin,
    [INPUT, OUTPUT, INPUT_PULLUP])
  int digitalRead(pin)
  digitalWrite(pin, [HIGH, LOW])

Analog In - pins A0-A5
  int analogRead(pin)
  analogReference(
    [DEFAULT, INTERNAL, EXTERNAL])

PWM Out - pins 3 5 6 9 10 11
  analogWrite(pin, value)
```

Advanced I/O
```
tone(pin, freq_Hz)
tone(pin, freq_Hz, duration_ms)
noTone(pin)
shiftOut(dataPin, clockPin,
  [MSBFIRST, LSBFIRST], value)
unsigned long pulseIn(pin,
  [HIGH, LOW])
```

Time
```
unsigned long millis()
  // Overflows at 50 days
unsigned long micros()
  // Overflows at 70 minutes
delay(msec)
delayMicroseconds(usec)
```

Math
```
min(x, y)   max(x, y)   abs(x)
sin(rad)    cos(rad)    tan(rad)
sqrt(x)     pow(base, exponent)
constrain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)
```

Random Numbers
```
randomSeed(seed) // long or int
long random(max) // 0 to max-1
long random(min, max)
```

Bits and Bytes
```
lowByte(x)    highByte(x)
bitRead(x, bitn)
bitWrite(x, bitn, bit)
bitSet(x, bitn)
bitClear(x, bitn)
bit(bitn)  // bitn: 0=LSB 7=MSB
```

Type Conversions
```
char(val)     byte(val)
int(val)      word(val)
long(val)     float(val)
```

External Interrupts
```
attachInterrupt(interrupt, func,
 [LOW, CHANGE, RISING, FALLING])
detachInterrupt(interrupt)
interrupts()
noInterrupts()
```

Variables, Arrays, and Data

Data Types

boolean	true | false
char	-128 - 127, 'a' '$' etc.
unsigned char	0 - 255
byte	0 - 255
int	-32768 - 32767
unsigned int	0 - 65535
word	0 - 65535
long	-2147483648 - 2147483647
unsigned long	0 - 4294967295
float	-3.4028e+38 - 3.4028e+38
double	currently same as float
void	i.e., no return value

Strings
```
char str1[8] =
  {'A','r','d','u','i','n','o','\0'};
  // Includes \0 null termination
char str2[8] =
  {'A','r','d','u','i','n','o'};
  // Compiler adds null termination
char str3[] = "Arduino";
char str4[8] = "Arduino";
```

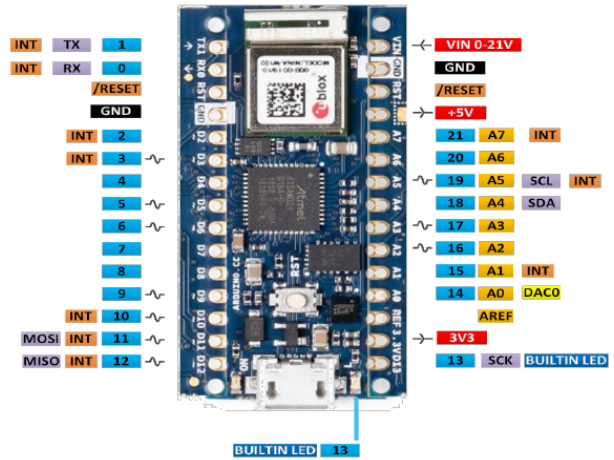
Numeric Constants

123	decimal
0b01111011	binary
0173	octal - base 8
0x7B	hexadecimal - base 16
123U	force unsigned
123L	force long
123UL	force unsigned long
123.0	force floating point
1.23e6	1.23*10^6 = 1230000

Qualifiers

static	persists between calls
volatile	in RAM (nice for ISR)
const	read-only
PROGMEM	in flash

Arrays
```
int myPins[] = {2, 4, 8, 3, 6};
int myInts[6];   // Array of 6 ints
myInts[0] = 42;  // Assigning first
                 // index of myInts
myInts[6] = 12;  // ERROR! Indexes
                 // are 0 though 5
```



Libraries

Serial - comm. with PC or via RX/TX
```
begin(long speed)  // Up to 115200
end()
int available()  // #bytes available
int read()   // -1 if none available
int peek()   // Read w/o removing
flush()
print(data)    println(data)
write(byte)    write(char * string)
write(byte * data, size)
SerialEvent()  // Called if data rdy
```

SoftwareSerial.h - comm. on any pin
```
SoftwareSerial(rxPin, txPin)
begin(long speed)  // Up to 115200
listen()       // Only 1 can listen
isListening()  // at a time.
read, peek, print, println, write
  // Equivalent to Serial library
```

EEPROM.h - access non-volatile memory
```
byte read(addr)
write(addr, byte)
EEPROM[index]  // Access as array
```

Servo.h - control servo motors
```
attach(pin, [min_uS, max_uS])
write(angle)  // 0 to 180
writeMicroseconds(uS)
   // 1000-2000; 1500 is midpoint
int read()    // 0 to 180
bool attached()
detach()
```

Wire.h - I²C communication
```
begin()      // Join a master
begin(addr) // Join a slave @ addr
requestFrom(address, count)
beginTransmission(addr) // Step 1
send(byte)              // Step 2
send(char * string)
send(byte * data, size)
endTransmission()       // Step 3
int available() // #bytes available
byte receive()  // Get next byte
onReceive(handler)
onRequest(handler)
```

CC BY SA

by Bart Huyskens
version: 08/01/2018

source: h ps://github.com/liffiton/Arduino-Cheat-Sheet/

- Original: Gavin Smith
- www.e2cre8.be >> Brainbox AVR

Les A Digitale outputs, delay en geluid

Lesonderwerpen

```
Opbouw Arduino programma, accolade, ;, commentaar

void pinMode(uint8_t pin, uint8_t mode)

void digitalWrite(uint8_t pin, uint8_t val)

void delay(unsigned long ms)

void delayMicroseconds(unsigned int us)

void tone(uint8_t _pin, unsigned int frequency, unsigned long duration ms)

pinMode (LED_BUILTIN, OUTPUT);

HIGH/LOW

TRUE/FALSE

LED_BUILTIN

INPUT/OUTPUT
```

Theorie les A

Structuur Arduino programma

De basisstructuur van de Arduino programmeertaal is erg simpel. Het bestaat uit minstens twee blokken. Deze twee blokken of functies vormen een aantal statements (programma commando's). Voorbeeld:

```
void setup()
{
statements;
}

void loop()
{
statements;
}
```

Waarbij setup() de voorbereiding is en loop() de uitvoering. Beide functies zijn nodig om een programma te kunnen laten werken.

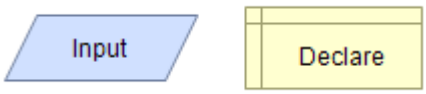
In de setup functie wordt o.a. bepaald welke pinnen ingang of uitgang worden. De setup functie wordt slechts eenmaal doorlopen.

De loop functie volgt na de setup functie en wordt oneindig herhaald. De loop functie leest vaak de inputs en gaat afhankelijk daarvan bepalen wat de outputs moeten doen. Eigenlijk komt het er op neer dat de loop functie de motor van het programma is, dus waar al het werk moet gebeuren.

setup()

De setup() functie wordt één keer aangeroepen wanneer het programma start.
Het wordt gebruikt om de mode van de pinnen te initialiseren of begint met seriële communicatie. De syntax ziet er als volgt uit:

```
void setup()
{
pinMode(pin, OUTPUT);   // maak de 'pin'
als uitgang
}
```



loop()

Nadat de setup() functie aangeroepen is, volgt de loop() functie. Deze doet precies wat de naam zegt en loopt oneindig alle instructies binnen de loop af. Wanneer de laatste instructie werd uitgevoerd start de loop terug bij de eerste instructie. De syntax:

```
void loop()
{
digitalWrite(pin, HIGH);        // zet 'pin' aan
delay(1000);                    // 1 seconde pauze
digitalWrite(pin, LOW);         // zet 'pin' uit
delay(1000);                    // 1 seconde pauze
}
```

{ } krullende haakjes (accolade)

Krullende haakjes geven het begin of het einde aan van een functieblok zoals je ook tegenkomt bij de void loop(). Kijk maar:

```
void loop()
{
Statements;
}
```

De eerste opende accolade (gekrulde haakje) **{** moet altijd afgesloten worden door een (gekruld gesloten) accolade **}**. Het aantal accolades is dus altijd een even

getal. Let daar goed op, want één accolade te weinig en een heel programma kan stoppen met werken. Vind dan de ontbrekende accolade maar eens terug.

; puntkomma

Een puntkomma moet gebruikt worden na elke ingevoerde opdracht (niet bij een vraag!!). Net zoals de gekrulde haakjes, zal bij het ontbreken van een puntkomma een error verschijnen. Voorbeeld:

```
int x = 13;      // declareer variable 'x' as the integer 13
```

Opmerking: Vaak is het zo dat het ontbreken van een puntkomma er voor zorgt dat de Arduino software niet wil compileren en een error aangeeft op een andere plek dan waar de puntkomma vergeten is. Dat wordt dus lastig zoeken.

/… */ blok commentaar*

Blok commentaar zijn gebieden met tekst die door het programma genegeerd worden. Tussen de /* en */ staat meestal uitleg over het programma of over de code die daar staat. Het mooie van "blok commentaar" is dat het over meerdere regels geplaatst kan worden.

```
/* dit is een blok met commentaar Vergeet niet het einde
van het Commentaar aan te geven
*/
```

Commentaar wordt nooit mee geprogrammeerd in de microcontroller. Het neemt dus geen geheugenruimte in beslag. Natuurlijk wordt het wel bewaard in het Arduino programma.

// regel commentaar

Een regel die begint met // en eindigt met tekst of code zal op die regel genegeerd worden. Ook dit neemt geen geheugen in de microcontroller in beslag. Code vooraan in de regel gevolgd door // zal wel uitgevoerd worden alles wat na de // komt op die regel echter niet.

```
// Dit is commentaar op een regel en onderstaande wordt
// niet uitgevoerd omdat het vooraf gaat met //.
// A = B + 3
```

Regel commentaar wordt vaak gebruikt om de genoemde opdrachten uit te leggen.

Regel 1: wij voorzien de belangrijkste regels van zinvolle commentaar, zodat we zelf of anderen later het programma snel kunnen begrijpen.

Ctrl + "/"

Gebruik de toet combinatie **Ctrl + "/"** om een blok code als commentaar te zetten of terug om te zetten naar code

Input output instructies

pinMode(pin, mode)

Wordt gebruikt in de void setup() loop om een specifieke pin te configureren als een INPUT of als een OUTPUT.

```
pinMode(pin, OUTPUT); // sets pin to output
```

Pinnen die geconfigureerd zijn als OUTPUT staan in een laagohmige toestand en kunnen maximaal 7 mA leveren. Dit is ruim genoeg voor een LED, maar veel te weinig voor een motor of bijvoorbeeld een relais.

Kortsluiting tussen verschillende poorten kunnen de poort of de gehele processor stukmaken.

digitalWrite(pin, value)

Schrijf de waarde naar een specifieke pin met als niveau HIGH of LOW. De pin is gespecificeerd als een variabele of een constante.

```
digitalWrite(pin, HIGH);    // maak 'pin' hoog
```

Het volgende voorbeeld leest een drukknop uit die is aangesloten op pin 7 en laat een LED, aangesloten op pin 13 aan gaan als de drukknop ingedrukt is:

```
int led = 13;   // LED op pin 13
int pin = 7;    // drukknop op pin 7 int value = 0;     // variabele value
int value = 0;

void setup(){
pinMode(led, OUTPUT);   // maak van pin 13 een output
pinMode(pin, INPUT);    // maak van pin 7 een input
}

void loop(){
value = digitalRead(pin);   // maak van 'value' de input pin
digitalWrite(led, value);   // zet value over naar de 'led'
}
```

Constanten

De Arduino taal heeft een aantal voor gedefinieerde waarden, die ook wel constanten worden genoemd. Ze worden gebruikt om een programma gemakkelijker te kunnen lezen of schrijven. Constanten zitten ook in verschillende groepen. Constanten worden steeds met hoofdletters geschreven.

TRUE/FALSE

Dit zijn Boolean constanten die een logisch niveau vaststellen. False wordt gedefinieerd als een 0, terwijl true wordt gedefinieerd als een 1 of iedere andere waarde anders dan een 0. In Boolean is -1, 2 en -900 gedefinieerd als true. Voorbeeld:

```
if (b == TRUE);     //Als variabele b waar is of als b niet gelijk is aan 0
{
Doe iets;            // willekeurige code
}
```

HIGH/LOW

Deze constanten definiëren een pin niveau van HIGH of LOW en worden gebuikt om digitale pennen te lezen of schrijven. HIGH is een logische 1 en LOW is een logische 0. Een logische 1 is meestal 5 V. Voorbeeld:

```
digitalWrite(13, HIGH);  // maak digitale pin 13 hoog
```

INPUT/OUTPUT

Deze constanten worden gebruikt met de opdracht pinMode() om te definiëren of een digitale pin INPUT of OUTPUT moet worden. Voorbeeld:

```
pinMode(13, OUTPUT);     //Pin 13 is een output
```

LED_BUILTIN

Meestal hangt de vaste led op een Arduino bordje aan pin13, maar als je het niet zeker weet, kan je het woord LED_BUILTIN gebruiken ipv 13. Arduino IDE weet zelf van elk bordje aan welke pin de led hangt en zal dit voor u vertalen.

```
pinMode(LED_BUILTIN, OUTPUT);     //Pin 13 is een output
digitalWrite(LED_BUILTIN, HIGH);  // maak digitale pin 13 hoog
```

Oefeningen bij les A

- A1: schrijf een programma dat de ingebouwde led laat flikkeren met een frequentie 10Hz. (meet dit na met de oscilloscoop – logic analyser)

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

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(50);
  digitalWrite(LED_BUILTIN, LOW);
  delay(50);
 }
```

- A2: schrijf een programma waarmee je de traagheid van uw ogen test. Vanaf een bepaalde snelheid lijkt het alsof de led constant aan is, terwijl deze toch aan en uit flikkert. Zoek die grens proefondervindelijk op.
- A3: Maak een programma dat een toon van 440Hz laat horen op de buzzer, enkel gebruik makend van delay() en delayMicroseconds() instructies.

Les B Digitale inputs, variabelen, seriële monitor en seriële plotter

Lesonderwerpen

```
int digitalRead(uint8_t pin)

Serial.begin(9600)
Seriële monitor
Seriële plotter

Types of variables
bool            0->1
byte            0->255
int             -32768 -> +32767
long            -2 147 483 648 -> 2 147 483 647
unsigned int    0 -> 65535
unsigned long   0 -> 4 294 967 295
float           kommagetallen -3.4028235*10^38-> -3.4028235*10^38
char
string
uint8_t, uint16_t, uint32_t

local & global variables

#define

&& - ||
```

Theorie bij les B

Variabelen

Een variabele is een manier om een getal-waarde te bewaren voor later gebruik in het programma. Zoals de naam variabele al aangeeft kan de waarde van een variabele ook regelmatig veranderen in de loop van het programma.

Er bestaan ook zogenaamde constanten. Constanten zijn geen variabelen.

Variabelen worden weggeschreven in het RAM geheugen van de processor. Vermits microcontrollers niet veel RAM geheugen hebben, moeten we zuinig omspringen met variabelen. Daarom is het van heel groot belang dat we de grootte van een variabele duidelijk meegeven aan het programma (dat noemen we 'declareren' van de variabele). Grote variabelen kunnen grotere getallen bevatten, maar nemen ook meer RAM geheugen in beslag en berekeningen met grote variabelen duren ook langer.

Een variabele moet op een juiste manier gedeclareerd worden. In de code hieronder wordt een variabele gedeclareerd genaamd Var1 die vervolgens de waarde krijgt die gemeten wordt op de analoge input pin 2:

```
int Var1 = 0;      // declareer een variabele met de naam Var1
// en geef die de waarde 0
// met int geeft je aan dat Var1 een 'Integer' //formaat
// heeft (16 bit) dat getallen tussen 32767 en -//32768 kan bevatten
Var1 = 1589;       // verander de waarde achter Var1 nu naar 1589
```

Een voorbeeld: De volgende code test of de Var1 kleiner is dan 100. Als dat zo is, dan krijgt Var1 de waarde 0. Is de waarde hoger dan 100, dan houdt Var1 de waarde die hij al had. In de derde regel zie je dat een pauze gemaakt wordt waarvan de duur bepaald wordt door de waarde die op dat moment in de variabele Var1 zit, gemeten in msec dan.

```
if (Var1 < 100)      // test of de variabele kleiner is dan 100
{
Var1 = 0;    // zo ja, dan krijgt hij de waarde 0
}

delay(Var1);     // De waarde van de variabele bepaalt de pauze
```

REGEL 2 : Geef variabelen een logische naam bijvoorbeeld TiltSensor of PushButton. Bekijken anderen jouw programma, dan wordt het al een stuk gemakkelijker om het te lezen. Je kunt elk woord voor variabelen gebruiken tenminste als het geen naam is die al gebruikt wordt in de Arduino omgeving.

Variabelen declareren

Alle variabelen moeten eenmalig gedeclareerd worden voordat je ze kunt gebruiken. Er zijn verschillende types zoals int, long, float die elk een ander formaat hebben of andere data kunnen bevatten.

Variabelen bereik

Een variabele kan op verschillende plaatsen in het programma gedeclareerd worden, maar die plaats heeft wel gevolgen voor de bruikbaarheid van deze variabele.

Een globale variabele is een variabele die in een heel programma kunt oproepen. Deze variabele declareer je boven void setup(). Globale variabelen zullen gedurende de volledige looptijd van het programma op dezelfde plaats in het RAM geheugen worden onthouden.

Een lokale variabele is een variabele die alleen gebruikt kan worden in een klein stukje van een programma. Het is een tijdelijke variabele. Zo'n stukje kan bijvoorbeeld in de void loop() zitten. Lokale variabelen nemen enkel tijdens de uitvoering van de betreffende regels code een bepaalde plaats in het RAM geheugen in. Nadien kan deze plaats door andere lokale variabelen gebruikt worden. Dit is een efficiëntere manier om met RAM geheugen om te springen, maar je moet wel opletten dat deze variabele enkel bruikbaar is binnen de huidige functieblok.

Opmerking: Hoe meer variabelen je gebruikt in een microcontroller des te sneller zal het geheugen vol zitten. Dat kan natuurlijk niet de bedoeling zijn.

REGEL 3: Kies met kennis van zake of u een variabele als lokale of als globale variabele wil declareren. Globale variabelen starten steeds met een hoofdletter per woord. Lokale variabelen noteer je steeds in kleine letters.

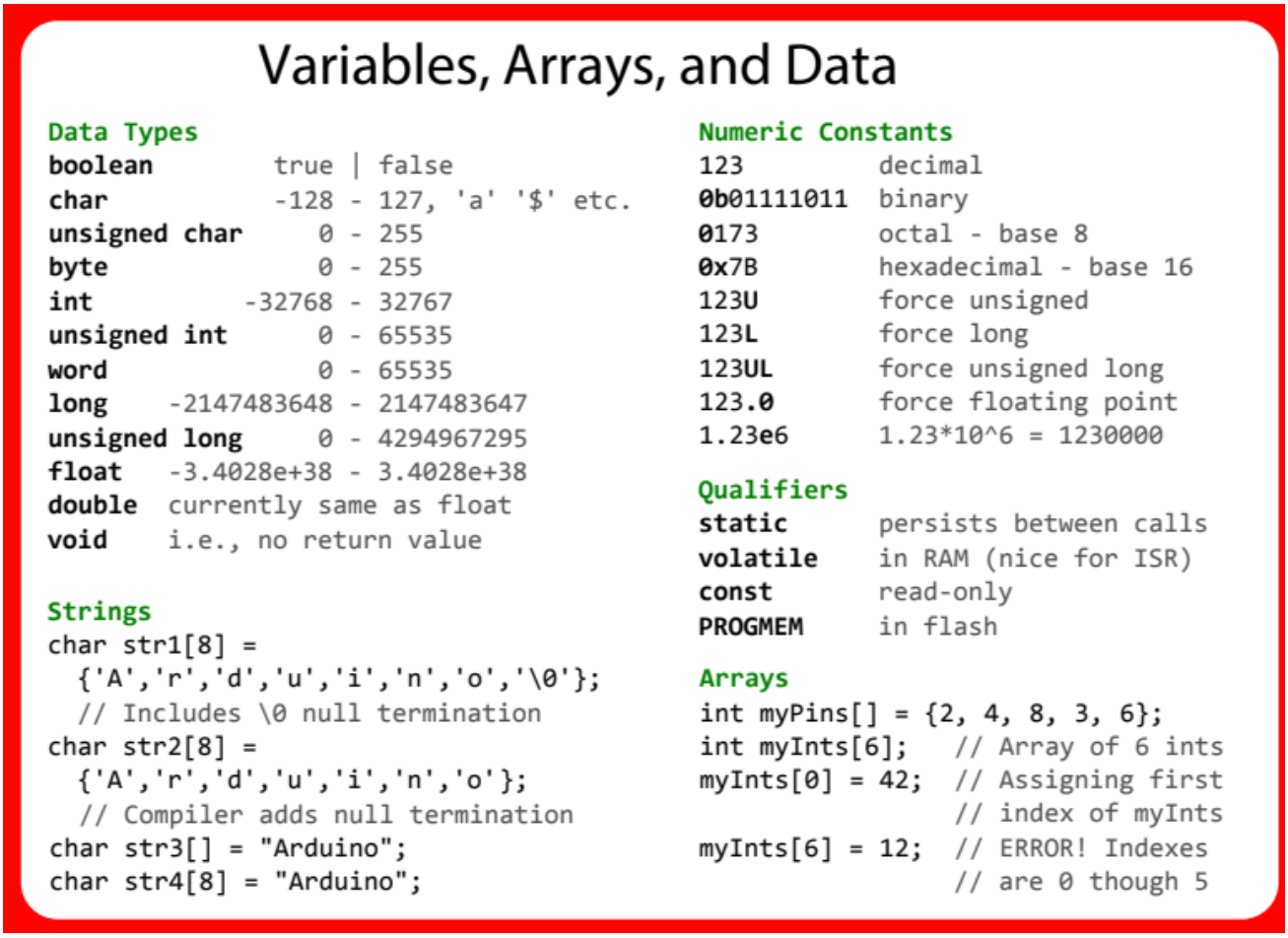
Het volgende voorbeeld zal duidelijk maken hoe de verschillende variabelen werken:

```
int Value = 0;          // globale var 'Value' is zichtbaar in het hele programma

void setup()
{}                  // geen setup nodig voor dit programma

void loop()
{
     for (int i=0; i<20;i++) // lokale var 'i' is alleen zichtbaar in de for loop
     {
       Value = Value + 1;
     }
     float f = 0;            // lokale var 'f' is alleen zichtbaar in void loop()
}
```

Op de volgende bladzijde worden de verschillende type variabelen beschreven.

Datatypes of Formaten van variabelen

In de **cheatsheet** staan de verschillende variabelen en opties mooi samengevat. We bespreken hier enkele van de meest gebruikte types.



Variables, Arrays, and Data

Data Types

boolean	true | false
char	-128 - 127, 'a' '$' etc.
unsigned char	0 - 255
byte	0 - 255
int	-32768 - 32767
unsigned int	0 - 65535
word	0 - 65535
long	-2147483648 - 2147483647
unsigned long	0 - 4294967295
float	-3.4028e+38 - 3.4028e+38
double	currently same as float
void	i.e., no return value

Strings

```
char str1[8] =
  {'A','r','d','u','i','n','o','\0'};
  // Includes \0 null termination
char str2[8] =
  {'A','r','d','u','i','n','o'};
  // Compiler adds null termination
char str3[] = "Arduino";
char str4[8] = "Arduino";
```

Numeric Constants

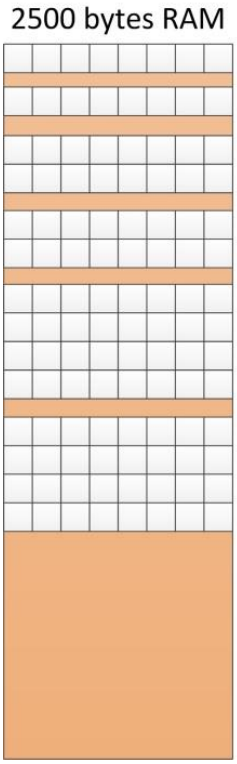
123	decimal
0b01111011	binary
0173	octal - base 8
0x7B	hexadecimal - base 16
123U	force unsigned
123L	force long
123UL	force unsigned long
123.0	force floating point
1.23e6	1.23*10^6 = 1230000

Qualifiers

static	persists between calls
volatile	in RAM (nice for ISR)
const	read-only
PROGMEM	in flash

Arrays

```
int myPins[] = {2, 4, 8, 3, 6};
int myInts[6];    // Array of 6 ints
myInts[0] = 42;   // Assigning first
                  // index of myInts
myInts[6] = 12;   // ERROR! Indexes
                  // are 0 though 5
```



byte x = 27;	8 bit 'byte'	0-255
char y = 'A';	8 bit 'char'	-128 -> +127 of tekens 'A', 'x'
int z = 426;	16 bit 'int'	-32768 -> +32767
unsigned int w = 290;	16 bit 'int'	0 -> 65535
long m = 9999999;	32 bit 'long'	-2,147,483,648 -> 2,147,483,647.
float o = 1.117;	32 bit 'float'	-3.4028235E+38 -> +3.4028235E+38

Variabelen worden opgeslagen in het RAM geheugen van onze µC. Afhankelijk van het type van variabele is het formaat een veelvoud van bytes. Hoe groter de variabele, hoe groter de getallen die er in kunnen worden opgeslagen, maar ook hoe langer het duurt om er mee te werken. Verstandig omgaan met variabelen is dus een belangrijke eigenschap van een goede programmeur.

byte

Byte bewaart een 8-bit numerieke waarde zonder een decimale punt met een bereik van 0-255.

```
byte Button1 = 180;      // declareert 'Button1' als een byte type
```

int

Integers zijn primaire datatypes om getallen te bewaren zonder een decimale punt een 16-bit waarde met een bereik van 32767 tot -32768.

```
int Count4 = 1500; // declareert Count4' als een integer type
```

Opmerking: Een integer variabele kan niet groter zijn dan 32767. Verhoog je 32767 met 1 dan wordt het een negatief getal: -32768.

```
unsigned int Count4 = 1500; // declareert Count4' als een unigned integer
```

Het woord “unsigned” voor de int zorgt ervoor dat deze int var enkel positieve getallen kan bevatten. De range wijzigt dan van 0 tot 65535

long

Datatype voor erg grote getallen, zonder een decimale punt (een soort uitgebreide integer) een 32-bit waarde met een bereik van 2,147,483,647 tot
-2,147,483,648.

```
long SomeVariabele = 90000;      // declareert 'SomeVariabele’ als een long type
```

float

Een datatype voor getallen met een decimale punt. Dus met getallen achter de komma. Floating datatypes nemen meer geheugen in gebruik dan een integer en worden opgeslagen als een 32-bit waarde met een bereik van 3.4028235E+38 tot
-3.4028235E+38.

```
float SomeVariabele = 3.14;    // declareert 'SomeVariabele' als een float-point type
```

Opmerking: Het is van groot belang dat je het formaat van de variabelen correct declareert. D.w.z niet te klein, maar zeker ook niet te groot.

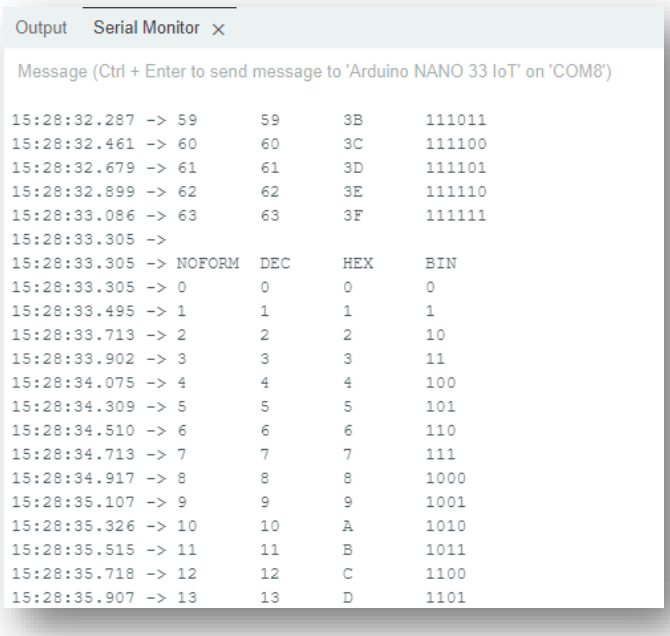
- Grote variabelen nemen meer plaats in in het beperkte RAM geheugen van de processor
- Het rekenen met grote variabelen kost de processor veel meer tijd als rekenen met kleine variabelen.

REGEL 3: Kies het formaat van uw variabele verstandig. Niet te groot, maar ook niet te klein.

Seriële monitor

Onderstaand programma demonstreert de functies van de serial monitor

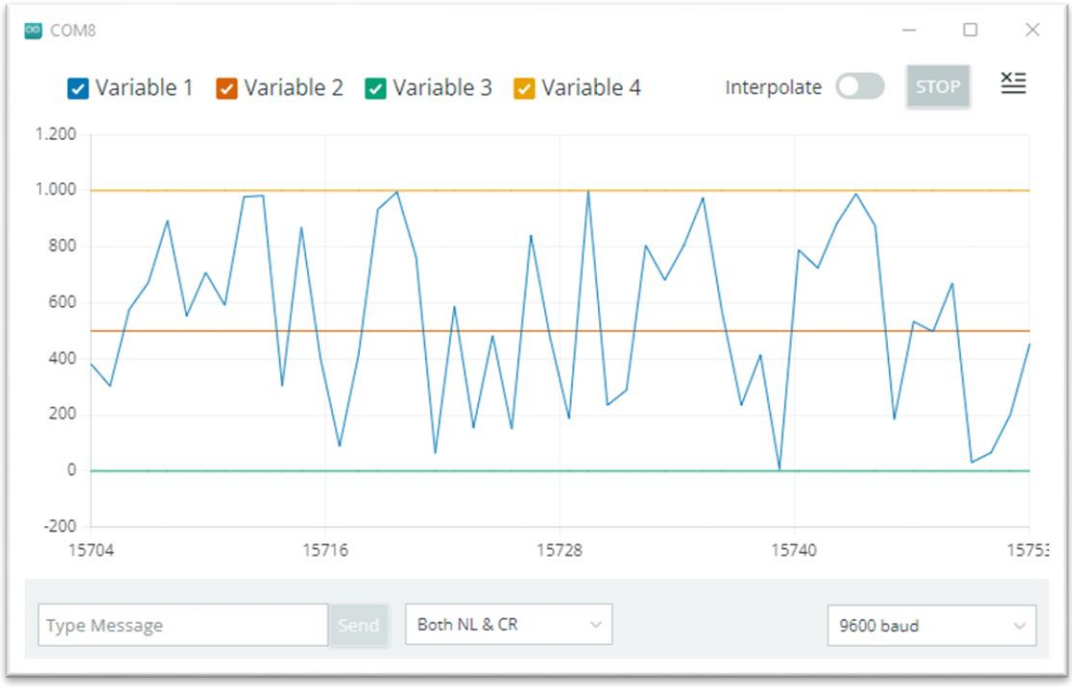
```
void setup() {
  Serial.begin(9600);  // open the
serial port at 9600 bps:
}

void loop() {
  Serial.print("NOFORM");  // print
tekst
  Serial.print("\t");       // prints a
tab (vaste spatie)

  Serial.print("DEC");
  Serial.print("\t");

  Serial.print("HEX");
  Serial.print("\t");

  Serial.print("BIN");
  Serial.print("\t");
  Serial.println("");  // begin een volgende regel - print niets


  for (byte x = 0; x < 64; x++) {  // Enkel de eerste 64 waarden

    // print deze waarden in 4 verschillende talstelsels:
    Serial.print(x);     // print as an ASCII-encoded decimal
    Serial.print("\t");  // prints a tab

    Serial.print(x, DEC);  // print as an ASCII-encoded decimal
    Serial.print("\t");    // prints a tab

    Serial.print(x, HEX);  // print as an ASCII-encoded hexadecimal
    Serial.print("\t");    // prints a tab

    Serial.println(x, BIN);  // print as an ASCII-encoded binary
                             // then adds the carriage return with "println"
    delay(200);              // delay 200 ms, om het geheel te vertragen
  }
  Serial.println("");  // begin een nieuwe lijn
}
```



seriële plotter

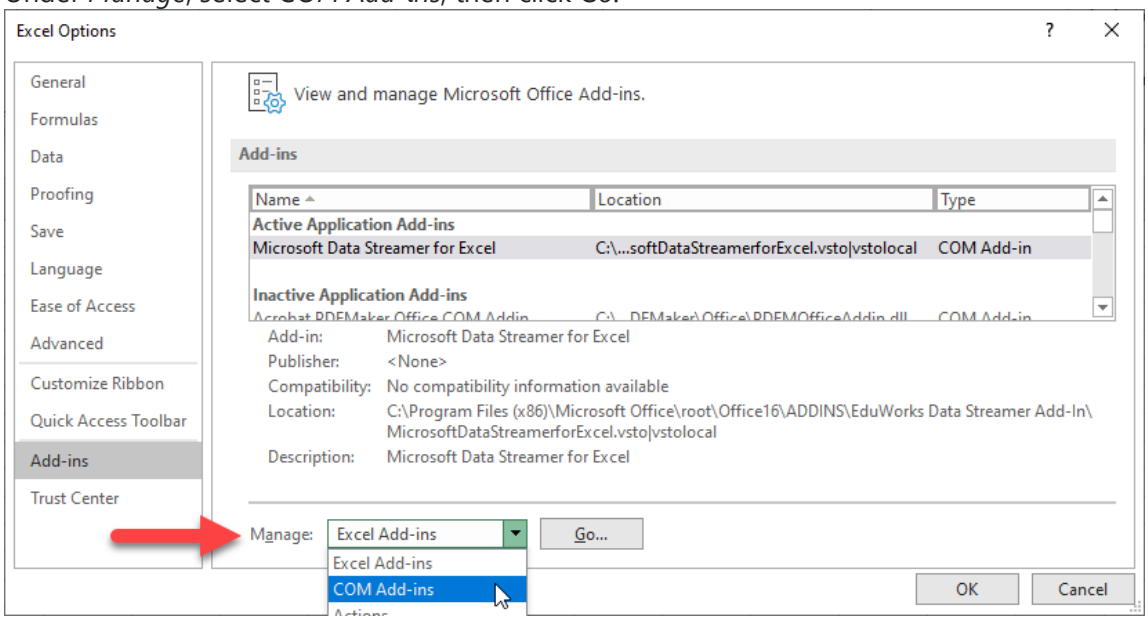
Onderstaand programma demonstreert de functies van de serial plotter

```
int random_variable;
int static_variable = 500;

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

void loop() {
  random_variable = random(0, 1000);

  Serial.print("Variable 1:");
  Serial.print(random_variable);
  Serial.print(",");
  Serial.print("Variable 2:");
  Serial.print(static_variable);
  Serial.print(",");
  Serial.print("Variable 3:");
  Serial.print(0); // print ondergrens mee om verspringen schaal te voorkomen
  Serial.print(",");
  Serial.print("Variable 4:");
  Serial.println(1000); // print bovengrens mee om verspringen schaal te voorkomen
                        // laatste instructie moet een println zijn
}
```

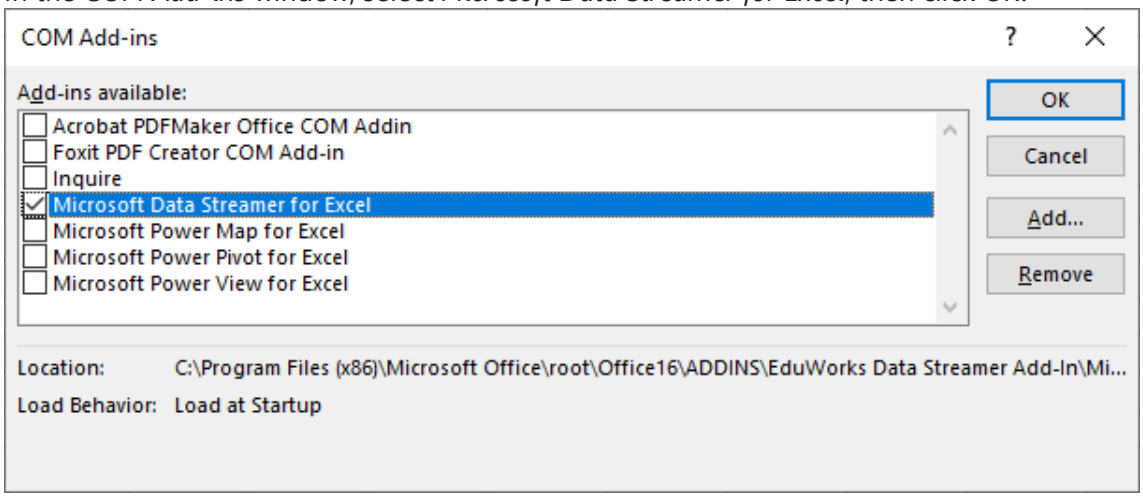


Export naar excel datastreamer

1. Open Excel.
2. Go to *File* > *Options* > *Add-Ins*.
3. Under *Manage*, select *COM Add-ins*, then click *Go*.



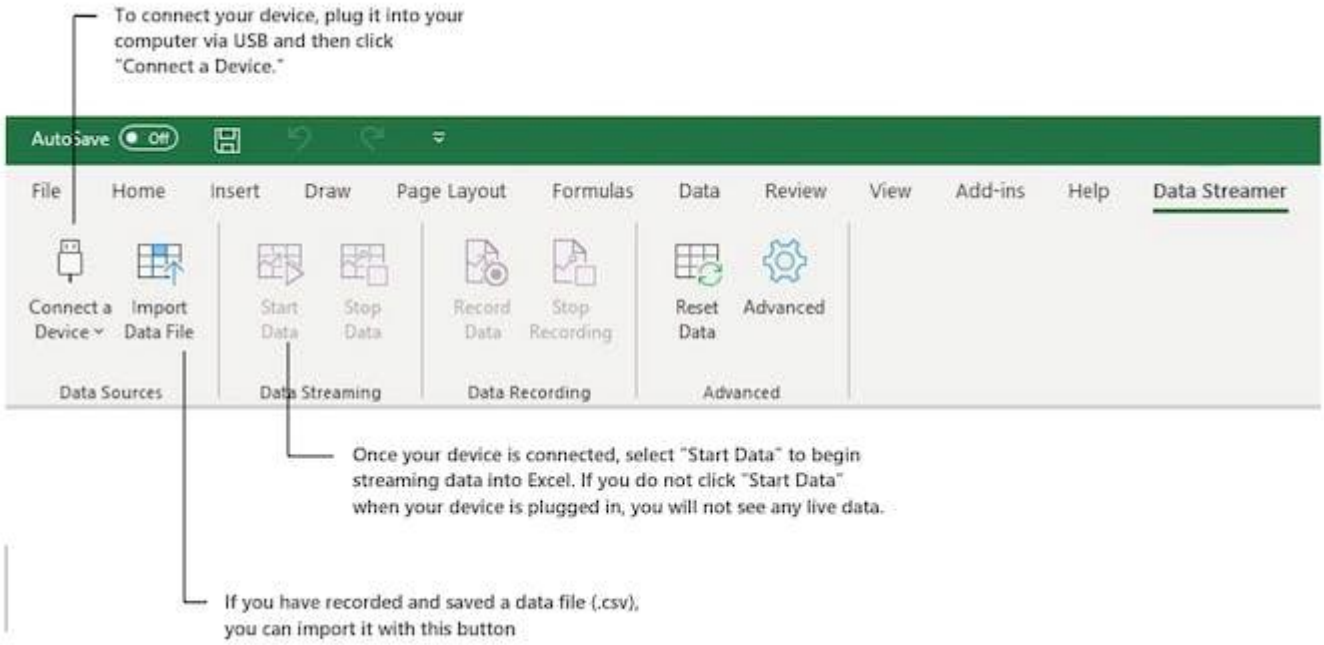
4. In the *COM Add-ins* window, select *Microsoft Data Streamer for Excel*, then click *OK*.



Open Excel and navigate to the Data Streamer tab. Click Connect a Device to connect Excel to the microcontroller.

This will create 3 new sheets:

- Data In: Live data is printed on this sheet.
- Data Out: Send data and/or commands to the Arduino board using this sheet.
- Settings: Change basic settings for Data Streamer, including the amount of live data and number of data columns.



Click Start Data to begin streaming data into Excel.

By default we only get 15 rows of data, but you can gather up to 500 rows of live data (limit is due to Excel bandwidth -- there's a lot happening in the background!).

#define

```
#define ledPin 3
```

Voor het compileren van het programma wordt op alle plaatsen 'ledPin' vervangen door '3'. # geeft aan dat dit info is voor de compiler.

```
int ledPin = 3;
```

zou hetzelfde eindresultaat geven, maar hiermee bezetten we wel een geheugencel. #define is dus beter.

Demoprogramma's met inputs en outputs

```
//input met if structuur

#define BLUELED 13
#define SW1 4
byte varSW1 = 0;
void setup()
{
  pinMode(BLUELED, OUTPUT);
  pinMode(SW1, INPUT);
}

void loop()
{
  varSW1 = digitalRead(SW1);
  if (varSW1 == 1)
  {
    digitalWrite(BLUELED, HIGH);
  }
  else
  {
    digitalWrite(BLUELED, LOW);
  }
}

//input met if structuur korte versie

#define BLUELED 13
#define SW1 4

void setup(){
  pinMode(BLUELED, OUTPUT);
  pinMode(SW1, INPUT);
}

void loop(){
   if (digitalRead(SW1)) digitalWrite(BLUELED, HIGH);
  else digitalWrite(BLUELED, LOW);
}


//AND POORT V1
#define BLUELED 13
#define SW1 4
#define SW2 5

void setup() {
```

```
  pinMode(BLUELED, OUTPUT);
  pinMode(SW1, INPUT);
  pinMode(SW2, INPUT);
}

void loop() {
  if (digitalRead(SW1)) {
    if (digitalRead(SW2)) {
      digitalWrite(BLUELED, HIGH);
    }
  } else digitalWrite(BLUELED, LOW);
}

//AND POORT V2
#define BLUELED 13
#define SW1 4
#define SW2 5

void setup() {
  pinMode(BLUELED, OUTPUT);
  pinMode(SW1, INPUT);
  pinMode(SW2, INPUT);
}

void loop() {
  if ((digitalRead(SW1)) && (digitalRead(SW2)))
    {
      digitalWrite(BLUELED, HIGH);
    }
   else digitalWrite(BLUELED, LOW);
}
```

Oefeningen bij les B

- B1: Maak een programma dat de toestand van een schakelaar weergeeft op een led. Als de schakelaar ingedrukt is moet de led branden, als de schakelaar wordt losgelaten mag de led niet branden.
- B2: Verander het vorige programma zodat de led uit gaat als de schakelaar ingedrukt wordt en aan wanneer die wordt losgelaten. (maak gebruik van variabelen en visualiseer alle parameters ook op de seriële monitor)
- B3: sluit twee schakelaars aan – op twee verschillende digitale inputs. De led mag pas aan gaan als beide schakelaars tezamen ingedrukt zijn (EN-POORT) (maak gebruik van variabelen en visualiseer alle parameters ook op de seriële monitor)
- B4: sluit twee schakelaars aan – op twee verschillende digitale inputs. De led mag pas aan gaan als minimaal 1 van de schakelaars ingedrukt is (OF-POORT) (maak gebruik van variabelen en visualiseer alle parameters ook op de seriële monitor)
- B5: sluit twee schakelaars aan – op twee verschillende digitale inputs. De led mag pas aan gaan als slechts 1 van de schakelaars ingedrukt is (EXOR-POORT) (maak gebruik van variabelen en visualiseer alle parameters ook op de seriële monitor)
- B6: Schrijf een programma dat de led aan laat gaan als SW1 wordt ingedrukt en uit laat gaan als SW2 wordt ingedrukt. (maak gebruik van variabelen en visualiseer alle parameters ook op de seriële monitor)

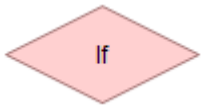
Les C Programmeerstructuren

Theorie bij les C

Programmeerstructuren of FLOW-CONTROL

Onderstaande programmeer-structuren zijn aanwezig in nagenoeg elke bestaande programmeertaal. Het zijn allemaal voorwaardelijke sprong- of herhalingsinstructies die het verloop (of de flow) van het programma kunnen bepalen afhankelijk van bepaalde voorwaarden. Elke flow-control instructie heeft zijn eigen toepassingen en dikwijls kunnen er verschillende instructies gebruikt worden om toch tot een zelfde werkend eindresultaat te komen.

```
Control Structures
if (x < 5) { ... } else { ... }
while (x < 5) { ... }
for (int i = 0; i < 10; i++) { ... }
break;      // Exit a loop immediately
continue;   // Go to next iteration
switch (var) {
  case 1:
    ...
    break;
  case 2:
    ...
    break;
  default:
    ...
}
```



if

De if opdracht test of bepaalde condities bereikt zijn. Denk bijvoorbeeld aan een analoog signaal dat een bepaalde waarde bereikt waarbij ingegrepen moet worden. In dat geval moet er iets gebeuren. Die actie moet dan plaats vinden binnen de haakjes (zie het voorbeeld hier onder). Wordt er niet aan de voorwaarde voldaan, dan wordt de actie tussen de haakjes overgeslagen.
Voorbeeld:

```
if (waardeVariabele == waarde)    //Als waardeVariabele gelijk is aan waarde
{
Doe iets;       // voer deze code uit als de voorwaarde waar is
}
// deze code wordt uitgevoerd als de voorwaarde niet waar is.
```

In het bovenstaande voorbeeld wordt de waardeVariabele vergeleken met een andere waarde.

Opmerking: Een veelvoorkomende schrijffout: **if(x=10).** Deze foute code (=-teken vergeten) zal geen foutmelding geven omdat dit programmatorisch geen fout is. Het geeft x de waarde 10 en heeft als resultaat dus altijd TRUE, omdat x nu niet gelijk is aan 0. Gebruik dus altijd met zorg '==' zodat bij de opdracht if(x==10) getest wordt of x gelijk is aan de waarde 10 of niet. Bedenk: bij '=' aan de term "maak gelijk aan" en bij '==' aan de vraag "is gelijk aan".

if… else

De if… else opdracht maakt het mogelijk hoe dan ook een beslissing te laten nemen. Bijvoorbeeld je meet dat een digitale input pin hoog is, in dat geval wil je dat actie_A start. Is de pin echter laag, dan moet actie_B starten. Dat zou er als volgt uit kunnen zien:.

```
if (inputPin == HIGH)   // controle of inputPin hoog is
{
Voer aktie_A uit;       // gevolg als inputPin hoog is
}
else
{
Voer aktie_B uit;       // gevolg als inputPin laag is
}
```

Else kan ook een andere procedure zijn zodat je meerdere testen in dezelfde lus kunt verwerken. Bekijk het volgende voorbeeld eens:

```
if (inputPin < 500)
{
Voer aktie_A uit;   // gevolg als inputPin kleiner is dan 500
}
else if (inputPin >= 1000)
{
Voer aktie_ B uit;// gevolg als inputPin groter of gelijk is aan 1000
}
else
{
Voer aktie_C uit; // gevolg als inputPin tss 500 en 999
}
```

Opmerking: Kijk goed naar de haakjes en de puntkomma's dat wil op deze wijze best wel eens ingewikkeld worden.

For (loop)

Het for commando wordt gebruikt om een aantal commando's een bekend aantal keren te laten herhalen. Via een teller wordt bijgehouden hoe vaak de lus zich moet herhalen. Het commando ziet er als volgt uit:

```
for (variabele; conditie; expressie)
{
        doeiets;
}
```

Dat lijkt lastig, maar het is een eenvoudige en vaak gebruikte opdracht. Een voorbeeld:

```
for (byte i=0; i<20; i++)          // declareer i, en zolang i kleiner is dan 20, i
wordt elke loop met 1 verhoogd
{
    digitalWrite(13,HIGH);         //  zet pin 13  aan
    delay(250);                      //  1/4 second  pauze
    digitalWrite(13,LOW);          //  zet pin 13  uit
    delay(250);                      //  1/4 second  pauze
}
```

In de eerste regel wordt i gedeclareerd en wordt meteen getest of i kleiner is dan 20. Daarna wordt i met 1 verhoogd (en krijgt de waarde 1). Aangezien i nul was, wordt alle code die er onder staat tussen de haakjes uitgevoerd. Daarna wordt regel 1 opnieuw uitgevoerd. Variabele i heeft nu de waarde 1 en er wordt weer getest of i kleiner is dan 20, hetgeen nog steeds het geval is. Nu wordt i met 1 verhoogd, zodat i de waarde 2 krijgt. Dat blijft zich herhalen tot i de waarde 20 bereikt. Daarna wordt de code tussen de haakjes niet meer uitgevoerd.

Opmerking: In C is de for loop veel flexibeler in te vullen dan in sommige andere computertalen zoals bijvoorbeeld BASIC. De variabele, conditie en expressie kan je naar wens aanpassen. Let op: ze worden wel gescheiden door een puntkomma

While (loop)

De while loop heeft wel wat weg van de for loop. Hij is gemakkelijk uit te leggen met: “Zolang je aan die voorwaarde voldoet, moet je deze instructies uitvoeren”. Die voorwaarde zou bijvoorbeeld het testen van een sensor kunnen zijn. De loop stopt pas als hij niet meer aan de voorwaarde voldoet. Een voorbeeld:

```
while (someVariable == value)
{
        doe iets;
}
```

Het volgende voorbeeld test of de someVariabele kleiner is dan 200. Als dat waar is, blijft de lus zich herhalen totdat someVariabele niet langer kleiner is dan 200. Onderstaande loop wordt dus 200 maal doorlopen.

```
byte someVariabele = 0;       // declareer deze var en zet deze op 0

while (someVariabele < 200)   // test of someVariabele kleiner
// is dan 200
{
Doe iets;                   // voer programmacode uit
someVariabele++;            // verhoog variabele met 1
}
```

Opmerking: indien de laatste regel vergeten zou worden, zal het programma in een oneindige lus terecht komen!

do… while

De do while loop werkt grotendeels hetzelfde zoals de while loop. Het verschil zit hem in het feit dat de conditie onderaan staat in plaats van bovenaan, zoals bij de while loop. Ongeacht de voorwaarde worden de instructies binnen een do while loop altijd minimaal 1 keer doorlopen.

```
do
{
doeiets;    // instructies binnen de do-while loop
                  // worden altijd minimaal 1 keer doorlopen
}
while (someVariable == value); // zolang deze voorwaarde TRUE is.
```

Switch…case

De switch case instructie gelijkt heel erg op de if instructie, maar geeft de programmeur de kans om op een verkorte manier hele specifieke gevolgen te geven aan specifieke situaties.

Met de switch instructie geef je aan welke variabele moet worden bekeken. Wanneer een case-statement wordt gevonden waarvan de waarde overeenkomt met die van de variabele, wordt de code in die case-instructie uitgevoerd.

Het break-sleutelwoord verlaat de switch-instructie en wordt meestal gebruikt aan het einde van elke case. Zonder een break-statement blijft de switch-instructie de volgende uitdrukkingen uitvoeren tot een pauze, of het einde van de instructie switch wordt bereikt.
De default wordt uitgevoerd wanneer geen van de cases een match heeft opgeleverd.

```
switch (var) {
  case 1:       // wanneer var gelijk is aan 1
    statements
    break;      // spring uit de case structuur
  case 2:       // wanneer van gelijk is aan 2
    statements
    break;      // spring uit de case structuur
  default:      // in alle andere gevallen
    statements }
```

Opmerking: switch case instructies kunnen enkel een gelijkheid testen en niet groter of kleiner dan. Daarvoor moet je kiezen voor een if instructie.

Demoprogramma's bij les C programmeerstructuren

De output van alle programma's hieronder is identiek. We laten de vaste led op onze controller een aantal seconden snel flashen en daarna een aantal seconden traag flashen.

```
// FOR LOOP - HERHALING
// 5 sec TRAAG flashen - 5 sec SNEL flashen
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {

  for (int x = 0; x < 10; x++) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(250);
    digitalWrite(LED_BUILTIN, LOW);
    delay(250);
  }
  for (int x = 0; x < 50; x++) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(50);
    digitalWrite(LED_BUILTIN, LOW);
    delay(50);
  }
}


// WHILE(1) loop - eeuwige loop
// 5 sec TRAAG flashen - 5 sec SNEL flashen
// daarna niets meer doen
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {

  for (int x = 0; x < 10; x++) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(250);
    digitalWrite(LED_BUILTIN, LOW);
    delay(250);
  }
  for (int x = 0; x < 50; x++) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(50);
    digitalWrite(LED_BUILTIN, LOW);
    delay(50);
```

```
  }
  while(1){ // doe niets voor eeuwig

  }
}


// WHILE(xx) loop - voorwaardelijke loop
// 5 sec TRAAG flashen - 5 sec SNEL flashen


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

void loop() {

  int x = 0;
  while (x < 10) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(250);
    digitalWrite(LED_BUILTIN, LOW);
    delay(250);
    x = x + 1;
  }

  while (x < 60) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(50);
    digitalWrite(LED_BUILTIN, LOW);
    delay(50);
    x = x + 1;
  }
}


// IF
// 5 sec TRAAG flashen - 5 sec SNEL flashen

byte x = 0;
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  if (x < 10) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(250);
    digitalWrite(LED_BUILTIN, LOW);
```

```
    delay(250);
  }

  if ((x >= 10) && (x < 60)) {
    digitalWrite(LED_BUILTIN, HIGH);
    delay(50);
    digitalWrite(LED_BUILTIN, LOW);
    delay(50);
  }
  x = x + 1;

  if (x > 59) {
    x = 0;
  }
}



// ELSE IF
byte x = 0;
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  if (x < 10) {
    digitalWrite(13, HIGH);
    delay(250);
    digitalWrite(13, LOW);
    delay(250);
  }

  else if (x < 60) {
    digitalWrite(13, HIGH);
    delay(50);
    digitalWrite(13, LOW);
    delay(50);
  }
  x = x + 1;

  if (x > 59) {
    x = 0;
  }
}
```

```
// IF ELSE IF ELSE
byte x = 0;
void setup() {
  pinMode(13, OUTPUT);
}

void loop() {
  if (x < 10) {
    digitalWrite(13, HIGH);
    delay(250);
    digitalWrite(13, LOW);
    delay(250);
  }

  else if (x < 60) {
    digitalWrite(13, HIGH);
    delay(50);
    digitalWrite(13, LOW);
    delay(50);
  }

  else {
    x = 0;
  }
  x = x + 1;
}
```

Oefeningen bij les C

- C1: laat de vaste led op het Arduino bordje aan en uit flikkeren met een aan en uit tijd van 100msec. De maximale delay die je mag gebruiken is 1msec. Schrijf dit programma 3x (met for , while en if elseif else)

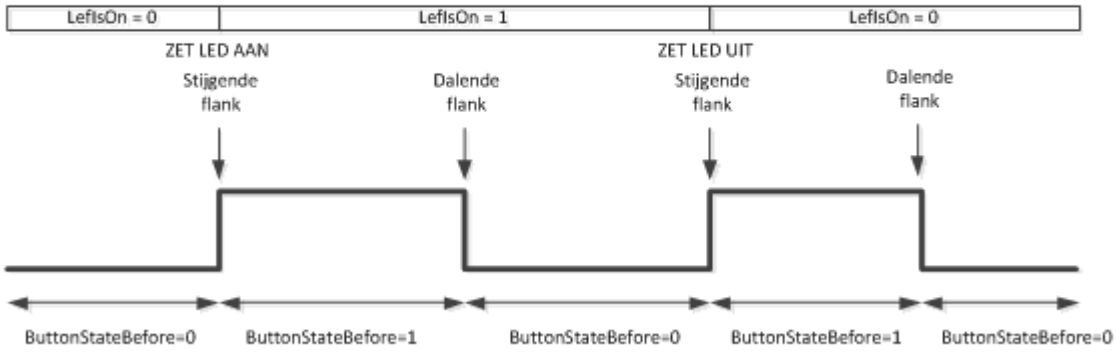
- C2: Maak een programma dat een sirene (500msec 440Hz, 500msec 880Hz) laat horen op de buzzer, enkel gebruik makend van delay() en delayMicroseconds() instructies – schrijf dit programma 3x (met for, while en if elseif else)

- C3: Je hebt twee inputs, C4 en C5. Als input C4 en C5 samen ingedrukt zijn, dan moet de led aan D2 knipperen met een frequentie van 5Hz. Als enkel C4 of enkel C5 ingedrukt is, moet de led aan D2 knipperen met een frequentie van 10Hz. Je mag je delay zelf kiezen.

- C4: Schrijf een START-STOP programma met 1 schakelaar. Wanneer de schakelaar de eerste maal wordt ingedrukt moet de led aan gaan, wanneer de zelfde schakelaar een tweede maal wordt ingedrukt moet de led weer uitgaan.

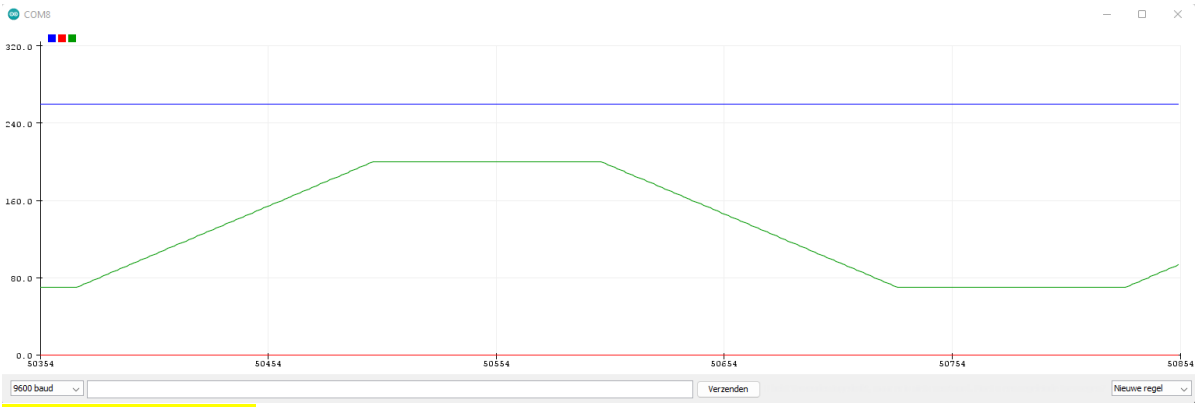
  TIP: Je zal code moeten schrijven die enkel op de positieve flank van de schakelaar reageert en je zal een bepaalde geheugenfunctie moeten programmeren die onthoudt wat de stand was van de led en van de schakelaar. (maak gebruik van variabelen en visualiseer alle parameters ook op de seriële monitor)

  Start stop met 1 drukknop



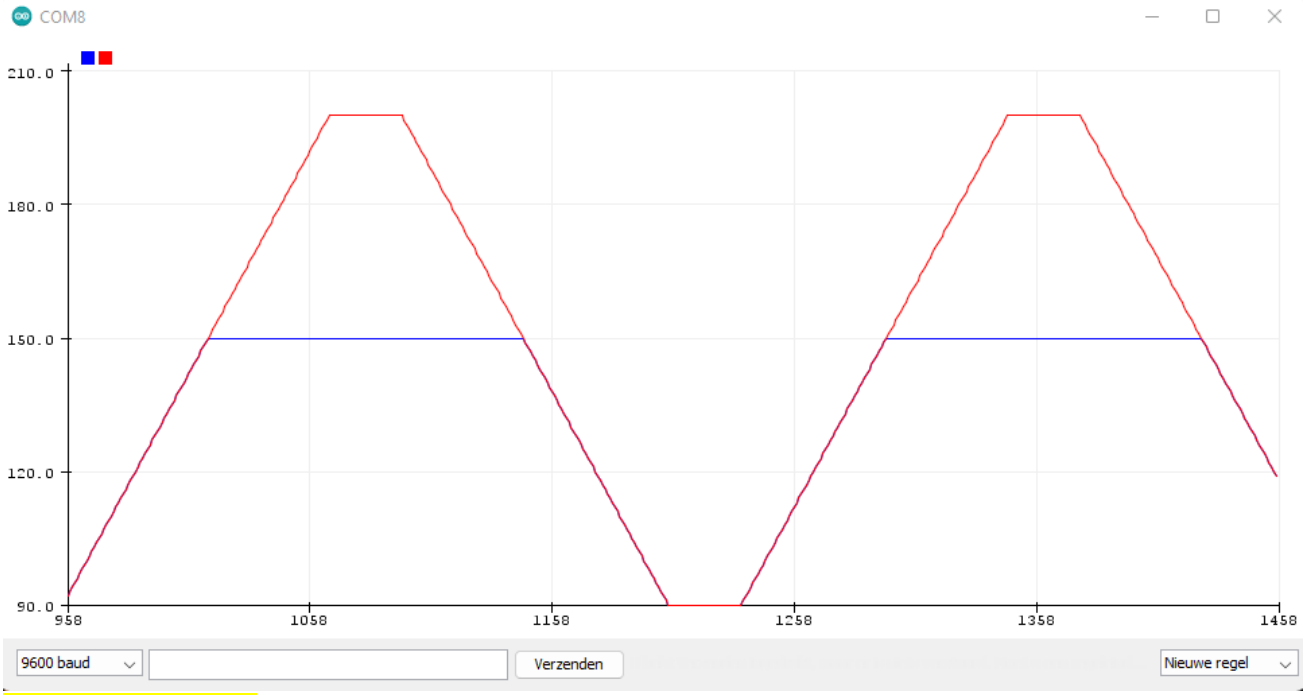
  Oplossing beschikbaar

- C5: schrijf een programma dat onderstaande blokgolf genereert op de serial plotter. We stijgen in stapjes van 70 naar 150, blijven even stabiel op 150 en dalen dan terug van 150naar 70 om ook daar even stabiel op 70 te blijven
  NOTA (Nov 2022 – werkt perfect onder IDE 1.8.5 omdat deze seriële plotter 500 meetpunten weergeeft, maar werkt niet mooi onder IDE 2.0.1 omdat deze seriële plotter slechts 50 meetpunten weergeeft – is reeds gemeld aan Arduino)



Oplossing beschikbaar

- C6: schrijf een programma dat onderstaande blokgolf genereert op de serial plotter. Er zijn hier 2 signalen. De beginwaarde is steeds 90, maar de ene grafiek stijgt tot 150 terwijl de andere doorstijgt naar 200.
  NOTA (Nov 2022 – werkt perfect onder IDE 1.8.5 omdat deze seriële plotter 500 meetpunten weergeeft, maar werkt niet mooi onder IDE 2.0.1 omdat deze seriële plotter slechts 50 meetpunten weergeeft – is reeds gemeld aan Arduino)



Oplossing beschikbaar

Les D Timing en wiskunde

Lesonderwerpen

```
unsigned long millis()
unsigned long micros()
```

wiskunde & Arduino

+ - * / % ! ==

min(x,y) max(x,y)

abs(x)

sqrt(x)

map(value, fromL, fromH, toL,toH)

round(x)

random(max) - random(min, max)

int()

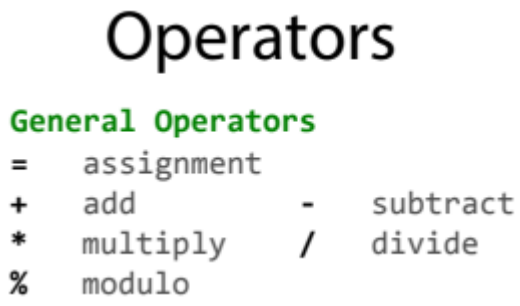
atoi()

float()

Theorie bij les D

Rekenen

Rekenkundige bewerkingen zijn bijvoorbeeld optellen, aftrekken, vermenigvuldigen en delen. Het is altijd een resultaat van twee getallen (operands). Voorbeelden:




```
y = y + 3;
x = x - 7;
i = j * 6;
r = r/ 5;
z = 11%5; //(uitkomst z = 1, % = modulo - rest van een deling)
```

De berekening wordt uitgevoerd afhankelijk van het gekozen datatype. Als er gekozen is voor een integer dan zal het resultaat 9 / 4 = 2 zijn in plaats van 2.25. Indien de datatypes van het type float waren geweest, dan had de uitkomst wel als een kommagetal worden weergegeven.

Pas ook op bij rekenkundige bewerkingen voor een "**overflow error"** bij te grote getallen. Een byte kan tot maximaal 255 bevatten, zodat een variabele die gedeclareerd is als een byte de optelling van 250 + 30 niet juist zal weergeven.
250+5 = 255 +1 = (256, maar dat past niet in een byte en wordt dus weergegeven als 0) 0 + 24 = 24. De uitkomst zal dus worden weergegeven als 24.

Zijn de getallen die je gaat bewerken van twee verschillende types, dan wordt het grootste type gebruikt voor de berekening. Bijvoorbeeld als één van de getallen een integer is en het andere getal een float dan zal de uitkomst een getal zijn van het type float.

Datatypes kunnen ook geforceerd worden omgezet naar een ander type tijdens berekeningen:

```
123U        force unsigned
123L        force long
123UL       force unsigned long
123.0       force floating point
1.23e6      1.23*10^6 = 1230000
```

Kies dus altijd een type variabele dat groot genoeg is voor de gewenste berekening. Zorg dat je weet wat er gebeurt als de gebruikte variabele door een optelling ineens van positief veranderd in negatief. Weet je het niet zeker lees dan een paar pagina's terug hoe je getallen moet declareren. Let echter wel op dat float variabelen veel geheugen in beslag nemen en ook de microcontroller zwaarder belasten (lees: langzamer werken).

Samengestelde opdrachten

Samengestelde opdrachten voor simpele wiskundige berekeningen kent de Arduino ook. Ze worden veel gebruikt in loops en worden later nog beschreven in deze manual. De meest voorkomende samengestelde opdrachten zijn:

```
x ++    // is hetzelfde als x = x + 1, of verhoog x met +1
x --    // is hetzelfde als as x = x - 1, of verlaag x met -1
```

Onderstaande samengestelde opdrachten komen ook voor in sommige programma's, maar wij gebruiken die niet omdat deze de leesbaarheid van een programma niet ten goede komen.

```
x += y  // is hetzelfde als as x = x + y, of verhoog x met +y
x -= y  // is hetzelfde als x = x - y, or of verlaag x met -y
x *= y  // is hetzelfde als x = x * y, of vermenigvuldig x met y
x /= y  // is hetzelfde als x = x / y, of deel x met y
```

Vergelijken van getallen

Variabelen worden vaak met elkaar vergeleken. Op basis daarvan worden er dan beslissingen genomen.

```
if (x == y)    // Als x gelijk is aan y
if (x != y)    // Als x is niet gelijk aan y
if (x < y )    // Als x is kleiner dan y
if (x > y )    // Als x is groter dan y
if (x <= y)    // Als x is kleiner of gelijk aan y
if (x >= y)    // Als x is groter of gelijk aan y
```

Logische berekeningen

Logische berekeningen zijn vergelijkingen die als uitkomst 'waar' of 'niet waar' (TRUE of FALSE) hebben. Er zijn drie logische operaties: AND, OR en NOT die vaak gebruikt worden in zogenaamde if opdrachten:

Logische AND "&&":

```
if ((x > 0) && (x < 5))        // enkel waar als beide vergelijkingen waar zijn
```

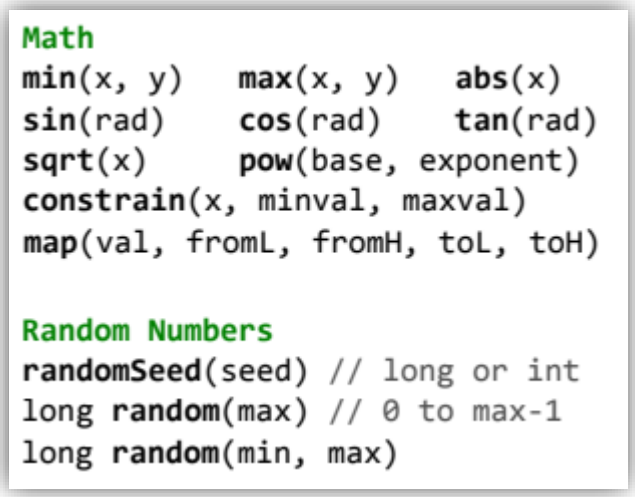
Logische OR "||":

```
if ((x > 0) || (y > 0))        // waar als één van de twee vergelijkingen waar is
```

Logische NOT "!":

```
if (!(x > 0))                  // alleen waar als vergelijking niet waar is
```

Wiskundige functies



1 UIT CHEATSHEET

We bespreken hier enkele van de mogelijke wiskundige functies.

min(x, y)

Berekent het minimum van twee getallen en geeft het kleinste getal weer.

```
waarde = min(getal, 100);
// 'waarde' is gelijk aan 100
// of kleiner dan 100 als getal
// kleiner  dan  100is
```

max(x, y)

Bereken het maximum van twee getallen en geef het grootste getal weer.

```
waarde = max(getal, 100);          // 'waarde' is gelijk aan 100
                                   // of hoger dan 100 als getal
                                   // hoger dan 100 is
```

map(value, fromLow, fromHigh, toLow, toHigh)

De map instructie wordt gebruikt om waarden te herschalen.

map(value**,** fromLow**,** fromHigh**,** toLow**,** toHigh**)**

Stel dat x een variabele is die getallen kan bevatten tussen 0 en 1023, maar dat we die waarde willen omzetten naar een evenredige waarde tussen 0 en 255 die we opslaan in variabele y, dan kunnen we dat zo schrijven:

```
y = map(x, 0, 1023, 0, 255);
```

Tijd

delay(ms)

Wachtlus (pauze) weergegeven in milliseconden, waarbij de waarde 1000 gelijk staat aan 1000msec of 1 seconde.

```
delay(1000); // wacht een seconde
```

delayMicroseconds(us)

Wachtlus (pauze) weergegeven in microseconden, waarbij de waarde 1000 gelijk staat aan 1000usec of 1 milliseconde.

```
delayMicroseconds(1000); // wacht een milliseconde
```

millis()

Laat zien hoeveel milliseconden het Arduino board in werking is na de start van het lopende programma. De weergave is een long variabele.

```
unsigned long Looptijd;   // looptijd kan een heel grote waarde aannemen
                          // en moet dus in een grote var gezet worden

Looptijd = millis(); // looptijd wordt gevuld met millis()
```

Note: Het weer te geven getal zal na verloop van tijd een overflow veroorzaken. (Een reset naar nul), na ongeveer 50 dagen.

micros()

Laat zien hoeveel microseconden het Arduino board in werking is na de start van het lopende programma. De weergave is een long variabele.

```
unsigned long Looptijd;   // looptijd kan een heel grote waarde aannemen
                          // en moet dus in een grote variabele gezet worden

Looptijd = micros(); // looptijd wordt gevuld met micros()
```

Note: Het weer te geven getal zal na verloop van tijd een overflow veroorzaken. (Een reset naar nul), na ongeveer 70 minuten

Oefeningen bij les D

- D1: maak 3 int variabelen aan (a, b = 10, c = 3). Voer de bewerking uit a = b/c. Visualiseer het resultaat in de seriële monitor– wat zie je?

```
int a = 0;
int b = 10;
int c = 3;

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

void loop() {
  delay(3000);

  a = b / c;

  Serial.print(a);
  Serial.print("\t");
  Serial.print(b);
  Serial.print("\t");
  Serial.println(c);
}
```

- D2: maak 3 int variabelen aan (a, b = 10, c = 3). Voer de bewerking uit a = b%c. Visualiseer het resultaat in de seriële monitor– wat zie je?
- D3: maak 3 float variabelen aan (a, b = 10.0, c = 3.0). Voer de bewerking uit a = b/c. Visualiseer het resultaat in de seriële monitor– wat zie je?
- D4: Maak een uitbreiding op oefening D3: maak een int variabele aan (x). Zet met de int() functie de float a om naar int x. Visualiseer het resultaat in de seriële monitor– wat zie je?
- D5: Maak een uitbreiding op oefening D3: maak gebruik van de round(x) functie om de uitkomst af te ronden. Visualiseer het resultaat in de seriële monitor– wat zie je?
- D6: Gebruik de random(min,max) functie om 2 random integer variabelen aan te maken. Gebruik de min(x,y) en max(x,y) functie om hiervan de minimum en maximum waarde uit te halen. Visualiseer het resultaat in de seriële monitor.
  Oplossing beschikbaar

- D7: Gebruik de millis() en micros() functies – in combinatie met de Seriële monitor om het aantal milliseconden weer te geven. De timer wordt gestart met SW1 en gestopt met SW2.
  Oplossing beschikbaar

- D8: Schrijf een programma dat een spel maakt waarbij je met de twee schakelaars zo dicht mogelijk een tijd van 3 seconden moet benaderen. De timer wordt gestart met SW1 en gestopt met SW2. De tijd moet mooi worden weergegeven, maar ook één van de volgende commentaren : Veel te kort, Een beetje te kort, Proficiat – helemaal juist, Een beetje te lang, Veel te lang.

Les E Analoge inputs en outputs

```
int analogRead(pin_size_t pinNumber)
void analogWrite(pin_size_t pinNumber, int value)
```

map(value, fromL, fromH, toL,toH)

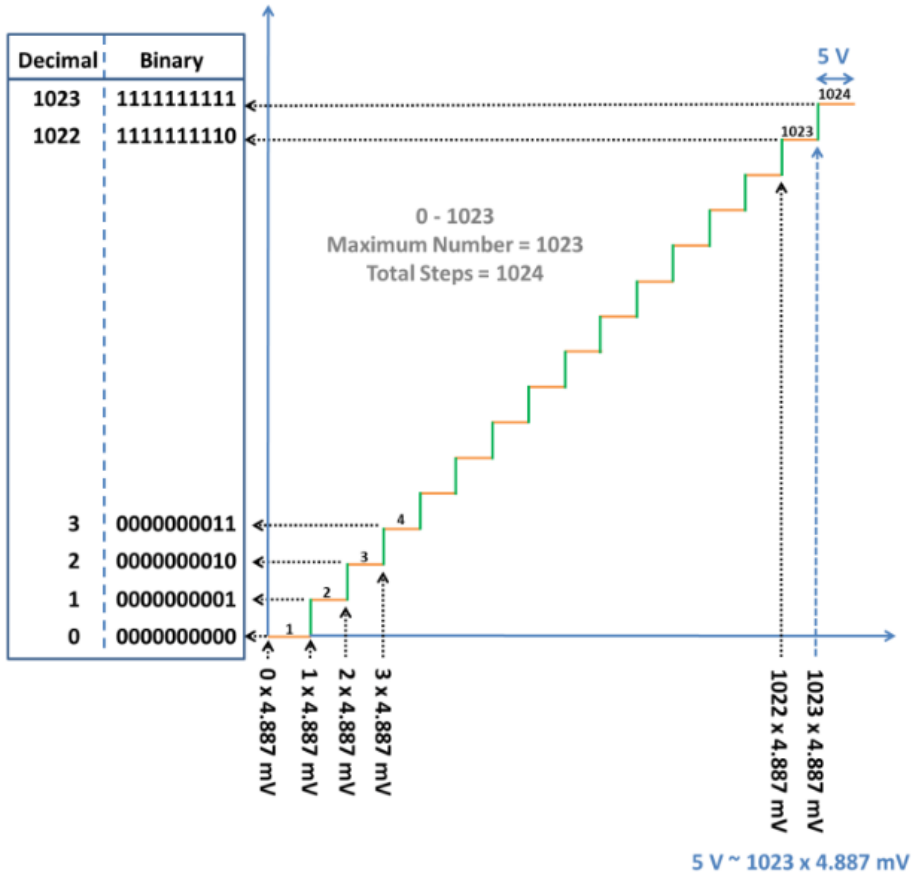
Theorie bij les E

analogRead(pin)

Leest de waarde van een specifieke analoge pin in een 10 bit resolutie. Deze functie werkt alleen op pin 0 t/m 6 – aangegeven met een **A0** . De uitkomst is een integer waarde tussen 0 to 1023.

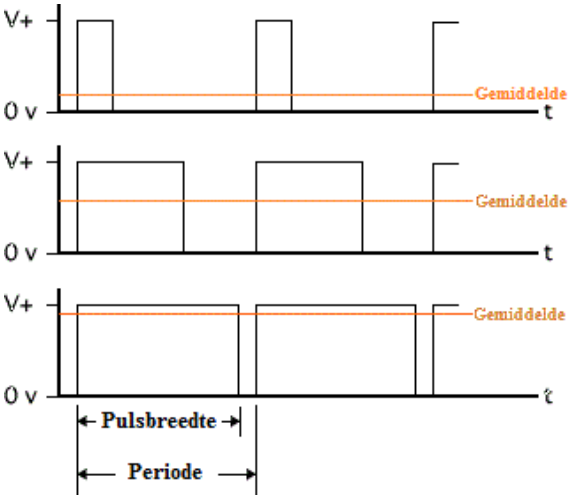
```
waarde = analogRead(pin);          // maak van waarde' wat gelezen
                                   // wordt op de 'pin'
```

Opmerking: Analoge pinnen hoeven niet te worden gedeclareerd als INPUT of OUTPUT. Het zijn automatisch al analoge inputs.



analogWrite(pin, value)

Niet tegenstaande de naam is dit eigenlijk geen analoge uitgang. Onze uC heeft namelijk geen analoge uitgangen, maar we kunnen wel dat effect creëren door op een uitgang een "PWM" puls-breedte (Pulse Width Modulation) modulatie signaal te genereren.



Een PWM signaal is een digitaal signaal met een constante frequentie, waarvan de aan-tijd varieert t.o.v. de uit-tijd. De gemiddelde spanning stijgt dan naarmate de aan-tijd langer wordt tov de Periode (ook wel duty cycle genoemd)

$$duty\ cycle = \frac{Aan - tijd}{Periode}$$

Doordat we zo snel aan en uit schakelen lijkt het voor leds of motoren alsof we de analoge spanning laten stijgen of dalen.

Zo'n PWM signaal genereren kan enkel op pins die gemarkeerd zijn met ~ .

De value of waarde kan worden weergegeven met een getal tussen 0 en 255.

```
analogWrite(pin, waarde);   // schrijf 'waarde' naar analoge 'pin'
```

Een waarde van 0 geeft 0 V als output op de gespecificeerde pin. Een waarde van 255 geeft 5 V als output op de gespecificeerde pin. Voor waarden tussen 0 en 255 vindt er een evenredige pulsbreedte modulatie plaats, waarbij opgemerkt moet worden dat de breedte van de pulsen evenredig groter wordt naarmate de waarde stijgt.

Omdat dit een hardware functie is, zal de uitgegeven pulsbreedte continue het zelfde zijn totdat er een nieuwe analogWrite wordt gedaan.

Het volgende voorbeeld leest een analoge waarde van een analoge INPUT pin. Vervolgens wordt deze waarde aangeboden aan een PWM OUTPUT pin. Let op de gelezen waarde is van 0 – 1023, maar de maximale aangeboden PWM waarde mag niet meer zijn dan 255. Daarom wordt de gelezen waarde herschaald met de map instructie.

```
#define led 6          //  LED met 220 weerstand op pin 6
#define joystickX A0   //  potentiometer op analoge pin 0
int value = 0;         //  value om te lezen
int valuePwm = 0;

void setup() {
  Serial.begin(9600);
}  //  geen setup nodig

void loop() {
```

```
  value = analogRead(joystickX);           //  lees 'value' op 'pin'
  valuePwm = map(value, 0, 1023, 0, 255);  //  converteer 0-1023 to 0-255
  analogWrite(led, valuePwm);              //  outputs PWM signaal naar led

  Serial.print(value);
  Serial.print("\t");
  Serial.print(","); // scheidingsteken voor excel datastreamer
  Serial.println(valuePwm);

  delay(500);
}
```

Oefeningen bij les E

- Bekijk welke pins op uw controller een analoge inputfunctie hebben.
- E1: Schrijf een programma dat de waarde van de potmeter weergeeft op de seriële monitor als een waarde tussen 0 en 1023 en als een waarde tussen 0 en 255. Voorzie deze waarden ook van de nodige tekst en tabs.
- E2: Arduino IDE heeft ook een Seriële plotter ingebouwd die meetwaarden grafisch kan weergeven. Onderzoek hoe deze werkt en schrijf een programma dat de meetwaarde van een potmeter grafisch kan weergeven.
- E3: Gebruik de waarde tussen 0 en 255 uit vorige oefening om:
    - Als de waarde onder of gelijk aan 75 ligt moet de PWM output op 0% liggen
    - Als de waarde ligt tussen 75 en 120 (120 telt nog mee)moet de PWM waarde om 25% liggen
    - Als de waarde ligt tussen 120 en 190 moet de PWM waarde op 50% liggen
    - Als de waarde boven 189 ligt moet de PWM waarde op 75% liggen
    - Houdt ook rekening met uw voorwaarden. Voor elke mogelijke situatie moet een gevolg geprogrammeerd zijn.
    - De waarde van de potmeter en de PWM waarden moet ook op de Serial monitor te volgen zijn.
- E4: kies nu zelf een analoge sensor (geen potmeter) die je correct aansluit op de BBA en waarvan je het meetresultaat correct weergeeft.
- Bekijk welke pins op uw controller een PWM output functie hebben.
- E5: Gebruik de analogWrite() functie om een led rustig harder te laten branden en dan rustig te laten doven. De huidige PWM waarde moet zichtbaar gemaakt worden op de Serial Monitor
- E6: Sluit een potmeter aan (I-AN POTMETER) en bepaal hiermee hoe hard een led gedimd wordt.
- E7: Sluit een **DC motortje** aan via een **Mosfet** en bepaal met een potmeter de snelheid van de DC motor.
- E8: Sluit één of 2 DC motortjes aan via de **H-brug** en bepaal met de joystick de snelheid en draairichting.
- E9: De joystick heeft 2 potmeters die elk in twee gedeeld zijn. Schrijf een programma dat die 4 halve potmeters elk een originele functie geeft. PWM & A/D & Excel

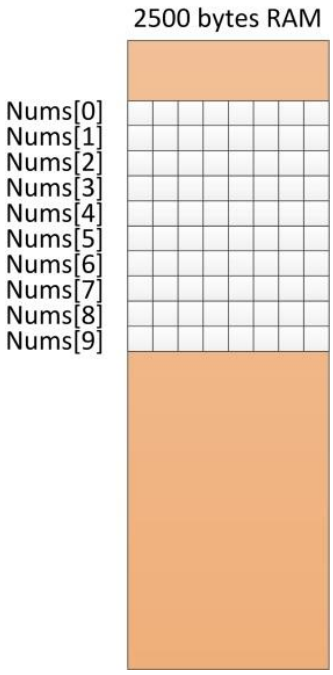
Les F Arrays

Array[x]

Theorie Les F

Het gebruik van Arrays om variabelen te declareren, kan u in een aantal gevallen veel winst opleveren qua programmeerwerk. In een array declareren we een groep variabelen, allemaal met hetzelfde formaat en enkel van elkaar verschillend door een indexnummer. Er zijn verschillende manieren om in Arduino IDE een Array te declareren:

```
int myInts[6];
// declareert een array met 6 variabelen van het type int (16bit)
// de naam van elke variabele in "myInts[index]"

byte myPins[] = {2, 4, 8, 3, 6};
// declareert een array met een niet gedefinieerd aantal elementen
// het formaat is byte (8bit) en de naam is myPins[index]
// We vullen deze array met 5 waarden, Arduino IDE beslist zelf dat
// deze array een array van 5 elementen zal zijn.

int mySensVals[6] = {2, 4, -8, 3, 2};
        // declareert een array van 6 elementen van het int type
        // De array wordt gevuld met 5 waarden.
        //De 6e variabele: mySensVals[5] blijft dus leeg

char message[6] = "hello";
        // declareert een array van het type char.
        //Dit type kan ook tekst bevatten (als ascii)
        //Dit type array noemen we ook een STRING
        // Het formaat moet telkens 1 plaats groter zijn dan het
        //aantal tekens om het '0' karakter (einde woord) ook mee
        // te kunnen opslaan.
```



Voorbeeldprogramma Arrays:

```
#define AantalWaarden 10  // aantal waarden in array
byte Nums[AantalWaarden];
// declareer een 10 byte array en vul die met waarden
int Som = 0;  // declareer een variabale SOM van het type int
int Gemiddelde = 0;

void setup() {
  Serial.begin(9600);  // open the serial port at 9600 bps:
}

void loop() {
  for (byte x = 0; x < AantalWaarden; x++) {
    Nums[x] = random(0, 20);
  }
  for (byte x = 0; x < AantalWaarden; x++) {
    Som = Som + Nums[x];  // alle Values optellen
  }
  Gemiddelde = Som / AantalWaarden;  // Gemiddelde berekenen
  Som = 0;                           // resetten SOM voor volgende loop

  for (byte x = 0; x < AantalWaarden; x++) {
    Serial.print(Nums[x]);
    Serial.print("  ");
  }
  Serial.println("");
  Serial.print("Gemiddelde: ");  // prints a label
  Serial.print("\t");            // prints a tab
  Serial.print(Gemiddelde);  // print de uitkomst
  Serial.println("");        // prints another carriage return
  delay(2000);               // wacht een halve seconde
}
```

Oefeningen bij les F

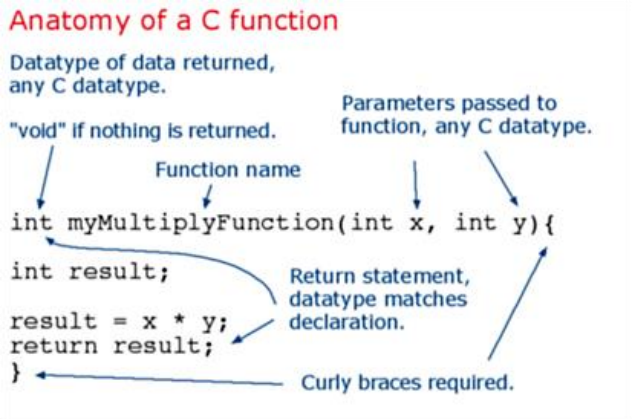
- F1: Pas het vorige programma aan zodat deze het gemiddelde neemt van 20 verschillende getallen
- F2: Maak een programma dat elke seconde aan analoge waarde inleest en dat de laatste 20 ingelezen waarden in een array bewaart (20^{e} waarde wordt telkens weggegooid, nieuwe waarde wordt telkens op positie 0 gezet). Display elke seconde het gemiddelde van deze 20 meetwaarden. Je hebt nu een 'moving average window' geschreven.
- F3: Pas het vorige programma aan zodat nu i.p.v. de berekening van het gemiddelde, de 20 getallen gesorteerd worden. Bestudeer hiervoor eerst verschillende sorteerroutines – je mag geen voorgeschreven Arduino sorteerroutine gebruiken.

Les G Functies

```
void Wacht(void)

void WachtAantal(int x)

int Vermenigvuldig(char a, char b)
```

Theorie bij les G

Het gebruik van functies maakt uw programma korter en overzichtelijker. C heeft een hele knappe functiestructuur. De typsiche Arduino bibliotheken en instructies zijn niet meer dan een verzameling van voorgeschreven functies die in Arduino IDE zijn ingewerkt.



Er bestaan 3 types van functies die alle 3 in onderstaand (nutteloos) programma worden gedemonstreerd.

```
void Wacht(void)        // functie die een delay van 1,5msec maakt
{
  delay(1);
  delayMicroseconds(500);
}
```

Deze functie krijgt geen data mee van de aanvrager (void = leeg) en geeft ook geen data terug aan de aanvrager. In dit geval wordt er een delay van 1.5msec gegenereerd. Zeker als je die meermaals in je programma nodig hebt is het interessant om deze regels telkens aan te roepen met een functie.

```
void WachtAantal(int x)   // functie die een delay van x aantal keer 1,5msec maakt
{
 for (int y = 0; y<x; y++)
 {
  delay(1);
  delayMicroseconds(500);
 }
}
```

Dit is een functie die wel data meekrijgt van de aanvrager **(**int x**)**, maar geen data teruggeeft aan de aanvrager(void). Bij de aanvraag staat de data die wordt meegegeven tussen de haakjes: WachtAantal**(**10**)**;

Die 10 wordt in de functie ingelezen in de variabele x die binnen de functie gebruikt wordt om een aantal keer de delay van 1.5msec te herhalen.

```
int Vermenigvuldig(char a, char b) // functie die a en b vermenigvuldigt en de uitkomst terugstuurt als een integer
{
    int resultaat;
    resultaat = a*b;
    return resultaat; // geef de uitkomst terug mee naar de functieaanvrager
}
```

Deze derde functie krijgt data van de aanvrager – in dit geval twee variabalen die in de functie in de lokale variabelen a en b worden gekopieerd. Binnen deze functie wordt hiervan het product gemaakt en de uitkomst wordt met de return instructie als een integer terug naar de aanvrager gestuurd. De functie wordt als volgt aangeroepen:

```
k = Vermenigvuldig(i,j);  //k heeft nadien de waarde i*j
```

ZINLOOS VOORBEELDPROGRAMMA FUNCTIES

```
byte i = 2;
byte j = 3;
byte k = 0;

void setup() {
  Serial.begin(9600);       // open the serial port at 9600 bps:
}

void loop(){

  k = Vermenigvuldig(i,j);  // roep functie vermenigvuldig aan en geef de waarde van
var i en j mee
                            // de uitkomst moet in var k komen staan
  Serial.print("k: ");      // prints a label
  Serial.print("\t");       // prints a tab

  WachtAantal(1200);          // roep functie wacht aan en geef waarde 1200 mee

  Serial.print(k);          // print de uitkomst k
  Serial.println("");       // prints another carriage return

  Wacht();                  // roep functie Wacht aan
}

void Wacht(void)            // functie die een delay van 1,5msec maakt
{
   delay(1);
   delayMicroseconds(500);
}

void WachtAantal(int x)   // functie die een delay van x aantal keer 1,5msec maakt
```

```
{
  for (int y = 0; y<x; y++)
  {
   delay(1);
   delayMicroseconds(500);
  }
}

int Vermenigvuldig(char a, char b) // functie die a en b vermenigvuldigt en de
uitkomst terugstuurt als een integer
{
 int resultaat;
 resultaat = a*b;
 return resultaat; // geef de uitkomst terug mee naar de functieaanvrager
}
```

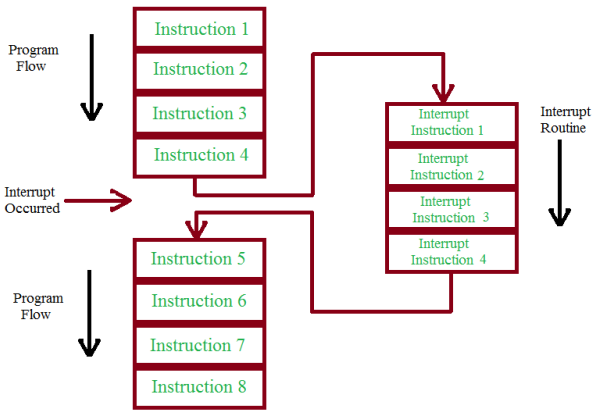
Oefeningen bij les G

- G1: Schrijf een functie van het type “void puls(void)” die een puls van 500msec aan en 300msec uit genereert op pin 2.
- G2: Schrijf een functie van het type “void puls(x,t)” die een puls van t msec aan en 300msec uit genereert op pin x.
- G3: Schrijf een functie van het type” int AD (a)” de analoge waarde van pin A0 meteen omzet naar een waarde tss 0-255 en zo ook teruggeeft.
- G4: Schrijf een programma met een nuttig gebruik van de 3 types functies.

Les H interrupts

Theorie bij Les H

Interrupts zijn een zeer krachtige feature van microcontrollers en maken het mogelijk om een lopend programma éénder waar te onderbreken om meteen te kunnen reageren op een bepaalde belangrijke event door een stukje code uit te voeren (interrupt routine) dat bij dit event hoort en dan terug verder te gaan met het uitvoeren van het hoofdprogramma.



Microcontrollers hebben vele mogelijke interrupt sources. Zo kan er een interrupt gegenereerd worden wanneer een timer overloopt, of wanneer er nieuwe data binnenkomt via de UART of wanneer er een flank gedetecteerd wordt op bepaalde pins.

We moeten echter in het achterhoofd houden dat we geen lege controller aan het programmeren zijn, maar wel een Arduino controller. De bootloader code die van onze controller een Arduino maakt, heeft al een aantal van deze interrupts in gebruik en Arduino is niet altijd even open over welke interrupts al in gebruik zijn. Daarom wordt het gebruik van interrupts in een Arduino programma niet echt gepromoot. Tijdens een interrupt routine onder Arduino IDE telt bv millis()niet op en seriële data wordt op dit moment niet ontvangen.

De interrupts op enkele externe pins van de Arduino Leonardo zijn echter wel te gebruiken, dus die zullen we hier gebruiken om het begrip interrupts te duiden.

!! Een interrupt routine moet zo snel mogelijk worden uitgevoerd. Delay() mag niet gebruikt worden.

Syntax Arduino interrupt

attachInterrupt(digitalPinToInterrupt(pin), ISR, mode); (recommended)

Parameters Arduino Interrupt

pin: **INT8**

ISR: the Interrupt Service Routine to call when the interrupt occurs; this function must take no parameters and return nothing. This function is sometimes referred to as an interrupt service routine.
mode: defines when the interrupt should be triggered. Four constants are predefined as valid values:

LOW	to trigger the interrupt whenever the pin is low,
CHANGE	to trigger the interrupt whenever the pin changes value
RISING	to trigger when the pin goes from low to high,
FALLING	for when the pin goes from high to low.

Voorbeeldprogramma Interrupt

```
#define ledPin 13       // ledpin zal bij compilatie vervangen worden door 13
#define interruptPin 3  // interruptPin zal vervangen worden door 3

volatile bool state = LOW;
// variabelen die op de meest onlogische momenten kunnen
// veranderen declareren we als volatile om de compiler aan te geven
// dat deze variabele zeker in het ram geheugen moet gezet worden.

void setup() {

  pinMode(ledPin, OUTPUT);       // maak pin 13 output (LED)
  pinMode(interruptPin, INPUT);  //maak pin 2 input (schakelaar)
  attachInterrupt(digitalPinToInterrupt(interruptPin), blink, CHANGE);
  // maak de interrupt actief op pin 2, elke keer als de
  //toestand van pin 2 wijzigt (CHANGE)
  // en roep dan telkens interrupt routine "blink" aan.
}

void loop() {
  digitalWrite(ledPin, state);
  // als state = 1, is de led aan, anders is de led uit
}

void blink() {
  state = !state;  // wijzig de toestand van state
}
```

Oefeningen bij les H interrupts

- H1: Wijzig bovenstaand programma zodat de led enkel van toestand verandert bij een stijgende flank.
- H2: Maak gebruik van interrupts om een periodemeter te maken die de periode van een blokgolf (10Hz – 500Hz (0V/5V) van functiegenerator of andere µC) meet in usec.
- H3: Vorm bovenstaand programma om tot een frequentiemeter.
- H4: sluit de Ultrasone afstandsensor correct aan en gebruik een interrupt om de afstand correct te bepalen.

Les I smart sensors - actuators

Theorie bij Les I

Vanaf hier proberen we meer en meer om informatie op te halen van

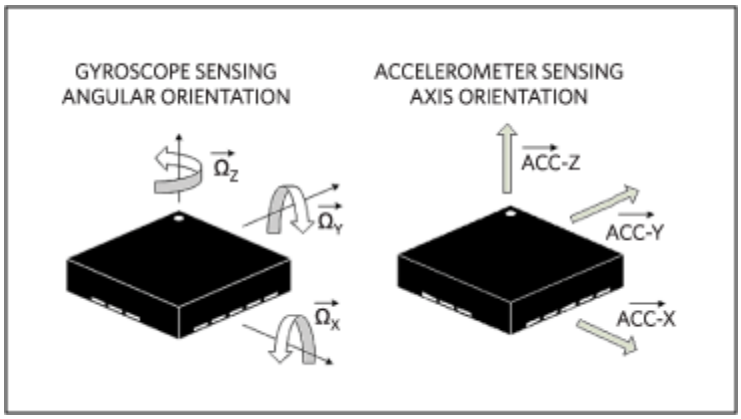
- Arduino website
- Google
- Fora
- File >> Examples

Er moeten meestal ook libraries geïnstalleerd worden die de communicatie met deze sensoren vereenvoudigen.

Oefeningen bij Les I

Acceloro en Gyro sensor

- I1: de Arduino nano 33 iot heeft intern een LSM6DS3 3 assige accelero en 3 assige gyrosensor ingebouwd. Bestudeer hoe je deze 6 sensorwaarde kan inlezen en mooi kan weergeven op de seriële plotter.




```
#include <Arduino_LSM6DS3.h>

void setup() {
  Serial.begin(9600);
  while (!Serial);

  if (!IMU.begin()) {
    Serial.println("Failed to initialize IMU!");

    while (1);
  }

  Serial.print("Accelerometer sample rate = ");
  Serial.print(IMU.accelerationSampleRate());
  Serial.println(" Hz");
  Serial.println();
  Serial.println("Acceleration in g's");
  Serial.println("X\tY\tZ");
}
```

```
void loop() {
  float x, y, z;

  if (IMU.accelerationAvailable()) {
    IMU.readAcceleration(x, y, z);

    Serial.print(x);
    Serial.print('\t');
    Serial.print(y);
    Serial.print('\t');
    Serial.println(z);
  }
}
```

- I2: Bestudeer de data uit B10 en maak een apparaat dat je mee kan inpakken in dozen met breekbare inhoud. Het apparaat moet een alarm geven als de dozen tijdens transport te ruw behandeld werden.

Temperatuursensor DS18B20

- I3: Op de printplaat van de Brainbox Nano 33 iot staat een DS18B20 temperatuursensor. Zoek uit hoe deze werkt en hoe je hiermee een correcte temperatuur kan uitlezen.

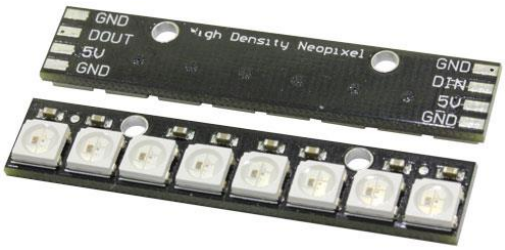

```
/*********************************************************************/
// First we include the libraries
#include <OneWire.h>
#include <DallasTemperature.h>
/*********************************************************************/
// Data wire is plugged into pin 2 on the Arduino
#define ONE_WIRE_BUS 2
/*********************************************************************/
// Setup a oneWire instance to communicate with any OneWire devices
// (not just Maxim/Dallas temperature ICs)
OneWire oneWire(ONE_WIRE_BUS);
```

```
/********************************************************************/
// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);
/********************************************************************/
void setup(void)
{
 // start serial port
 Serial.begin(9600);
 Serial.println("Dallas Temperature IC Control Library Demo");
 // Start up the library
 sensors.begin();
}
void loop(void)
{
 // call sensors.requestTemperatures() to issue a global temperature
 // request to all devices on the bus
/********************************************************************/
 Serial.print(" Requesting temperatures...");
 sensors.requestTemperatures(); // Send the command to get temperature readings
 Serial.println("DONE");
/********************************************************************/
 Serial.print("Temperature is: ");
 Serial.print(sensors.getTempCByIndex(0)); // Why "byIndex"?
   // You can have more than one DS18B20 on the same bus.
   // 0 refers to the first IC on the wire
   delay(1000);
}
```

- I4: Gebruik de temperatuursensor in combinatie met de mosfet om een ventilator te laten draaien als de temperatuur te hoog stijgt.
- I5: Gebruik de temperatuursensor in combinatie met de mosfet om een verwarmingselement in te schakelen als de temperatuur onder een bepaalde waarde komt.

Neopixel – WS2812B

- H6: Zoek uit hoe Neopixels (type WS2812B) werken en maak hier iets origineels mee



```
/*
NeoPixel LEDs basis
BBiot - Jan 2022 - E2CRE8.be - www.stemzone.be
installeer eerste de Adafruit Neopixel lib

*/

#include <Adafruit_NeoPixel.h>

#define PIN      6
#define NUMPIXELS 8


Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

#define DELAYVAL 500 // Time (in milliseconds) to pause between pixels

void setup() {
pixels.begin();
}

void loop() {
pixels.clear();
pixels.setBrightness(20);
pixels.setPixelColor(0, pixels.Color(255, 255, 255)); // waarden voor led 0 (WIT)
pixels.setPixelColor(1, pixels.Color(255, 0, 0));     // waarden voor led 1 (ROOD)
pixels.setPixelColor(2, pixels.Color(0, 255, 0));     // waarden voor led 2 (GROEN)
pixels.setPixelColor(3, pixels.Color(0, 0, 255));   // waarden voor led 3 (BLAUW)
pixels.setPixelColor(4, pixels.Color(255, 0, 255));
pixels.setPixelColor(5, pixels.Color(255, 255, 0));
pixels.setPixelColor(6, pixels.Color(0, 255, 255));
pixels.setPixelColor(7, pixels.Color(0, 100, 100));
pixels.show();
}
```

Internet Of Things

Inleiding Arduino Iot Cloud

Het aantal apparaten die geconnecteerd zijn met het internet stijgt met miljoenen per jaar. Al die geconnecteerde ‘dingen’ vormen samen het ‘internet of things’ of Iot.

De Arduino Iot Cloud omgeving is een platform dat het voor iedereen mogelijk maakt om zelf Iot projecten te verwezenlijken met onze Brainbox nano 33 iot. In deze ‘alles in één’ gebruiksvriendelijke omgeving kunnen gebruikers zowel de hardware configureren als de software schrijven en de gemeten data visualiseren.

Online informatie en tutorials (steeds up to date)

Homepage: https://cloud.arduino.cc/

Cheat Sheet: https://docs.arduino.cc/arduino-cloud/getting-started/technical-reference

Tutorials: https://docs.arduino.cc/arduino-cloud/

Mijn eerste Iot cloud project met Arduino.

Bekijk VIDEOLES: “Kennismaking met Arduino Iot Cloud”

Als eerste gaan we onze brainbox linken met de iot platform en gaan we de werking van de verschillende onderdelen demonstreren a.d.h.v. een eenvoudig programma.

Link de Brainbox nano 33 iot aan de Arduino cloud

1. Maak een Arduino account aan
   - Gebruik bij voorkeur uw google account

   Link: https://login.arduino.cc/login

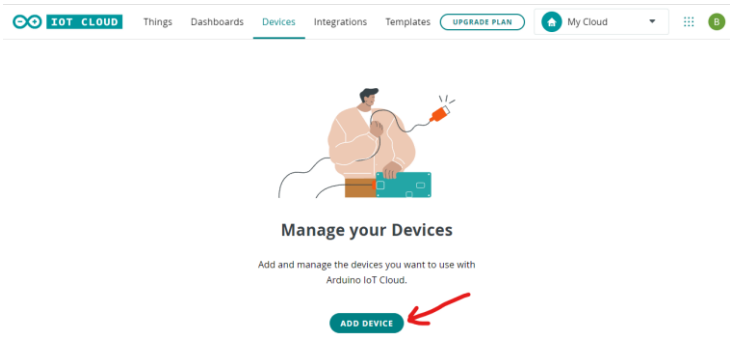
2. Log in bij de ‘Arduino iot cloud’

   Link: https://create.arduino.cc/iot

3. Configureer een ‘Device’

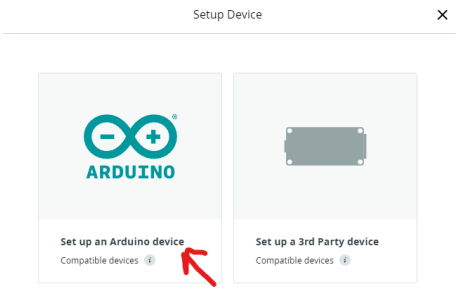
   Doel: Hierdoor wordt onze Arduino nano 33 iot herkent door de Iot Cloud omgeving en krijgt deze een aantal unieke codes zodat het een unieke module wordt op het wereldwijde internet.

   Onder het tabblad ‘Devices’ – klik op ‘ADD DEVICE’



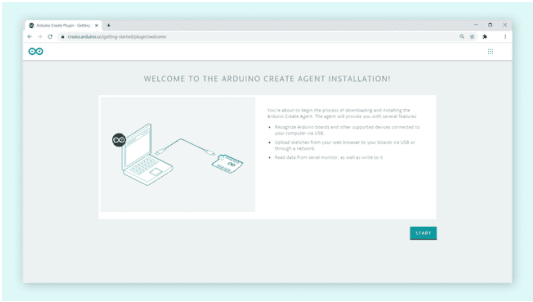
   Connecteer uw Brainbox iot via de USB kabel aan de PC waarop je bezig bent.

   Selecteer ‘Set up an Arduino device’

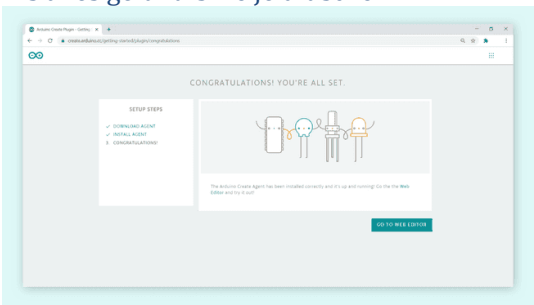


   Installeer de ‘Arduino Create Agent’

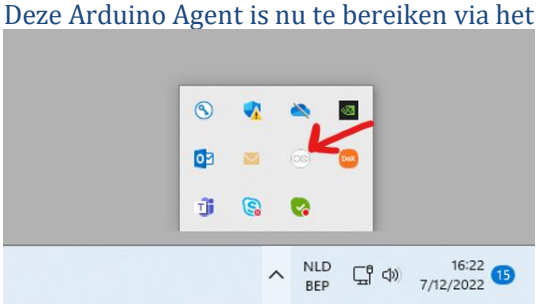
   Dit is een klein programma dat lokaal op uw computer moet geïnstalleerd worden om er voor te zorgen dat het online programma van Iot cloud kan communiceren met de nano 33 iot processor die aan uw PC hangt. De link tussen online en offline dus.



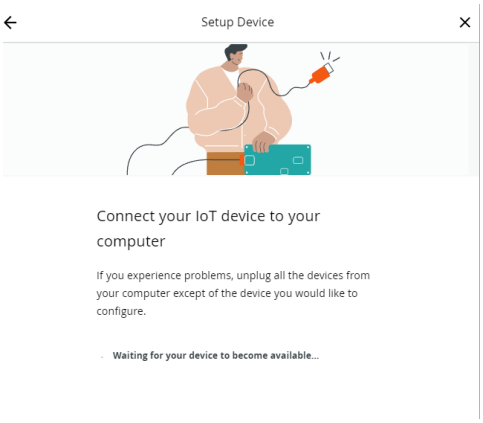
Als alles gelukt is zie je dit scherm:



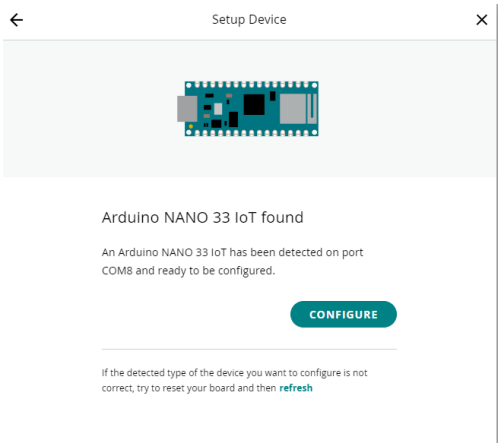
Deze Arduino Agent is nu te bereiken via het pijltje rechts onderaan uw scherm



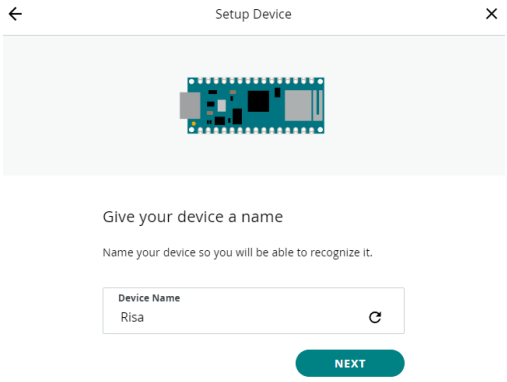
Verbindt de Brainbox nano 33 iot via de USB kabel met een USB poort op uw computer



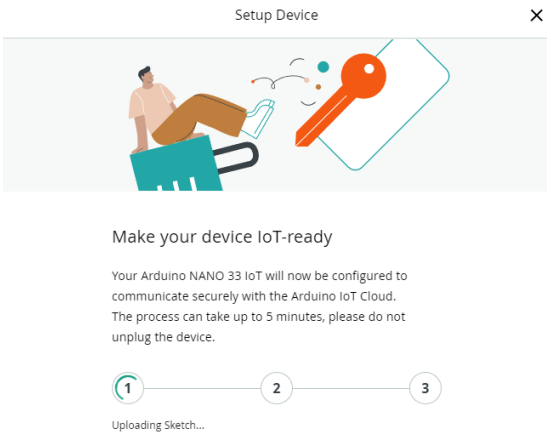
Geef de software even de tijd – wanner het bordje herkend wordt zal onderstaand scherm verschijnen:



Klik op 'configure'



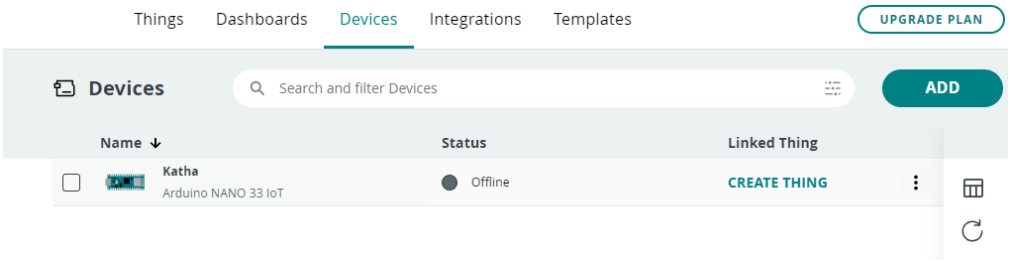
Geef uw device zelf een naam, of neem de voorgestelde naam over.
Klik op 'NEXT'



Bovenstaand scherm heeft dikwijls een hele tijd (tot meer dan 5 min) nodig om tot aan stap 3 te geraken. Er worden namelijk een aantal unieke codes en een basisprogramma in de nano 33 iot geladen om er voor te zorgen dat deze processor als een uniek 'device' herkend wordt op het internet.

Als het niet lukt kunnen volgende acties een oplossing bieden:

- 'Pause' de Arduino agent en 'Resume agent' om de Arduino Create Agent terug op te starten.
- Koppel de USB kabel los en sluit deze terug aan – controleer of de power led brandt
- Zorg ervoor dat er geen hardware op de Arduino aangesloten is
- Reset de Arduino door op de reset knop te drukken, of de USB kabel even los te koppelen.

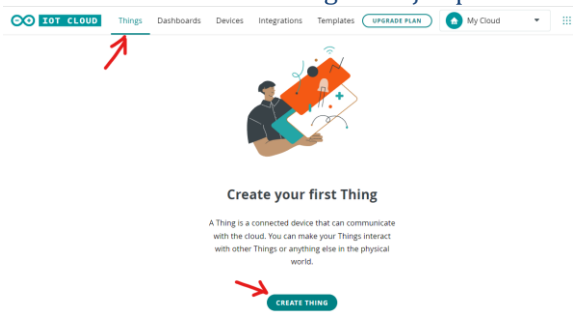


Je hebt nu een device aangesloten

4. Ceëer een 'Thing'

Doel: Hierin configureer en schijf je het programma dat in de Arduino nano 33 iot zal draaien. Alles draait hier om het 'delen' van variabelen tussen de Arduino en de Iot cloud omgeving. Beide platformen moeten gelijktijdig aan dezelfde variabelen kunnen.

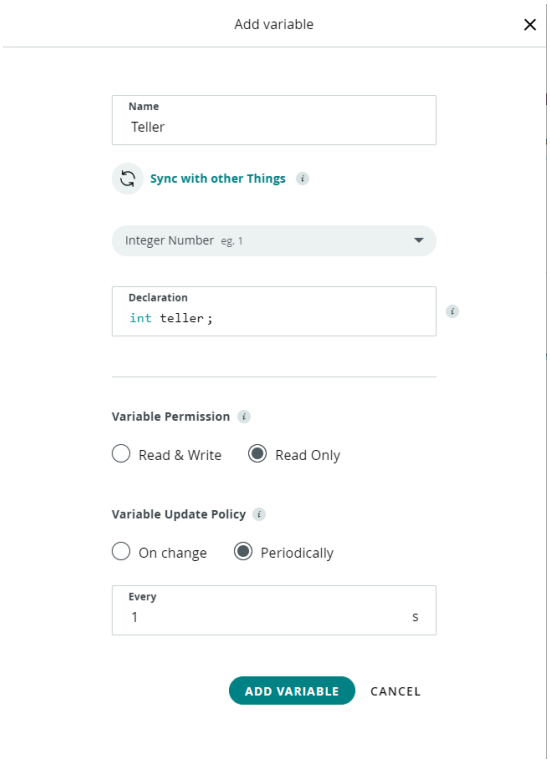
Onder het tabscherm 'Things' klik je op CREATE THING



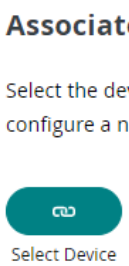
Verander de naam en klik op ADD VARIABLE

Als variabele naam vullen we in onze eerste test de naam Teller in,

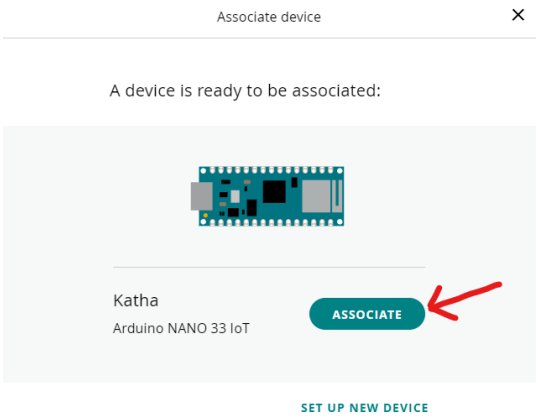
- Als type kiezen we een 'integer number' die grote gehele getallen kan bevatten
- Read only betekent dat data enkel van de Arduino naar de Cloud kan gestuurd worden
- Periodically betekent hier dat we elke seconde data van de Arduino naar de Cloud sturen
- Klik op ADD VARIABLE



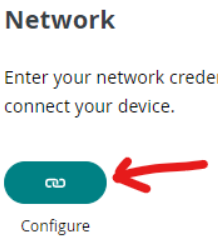
Klik op Select Device



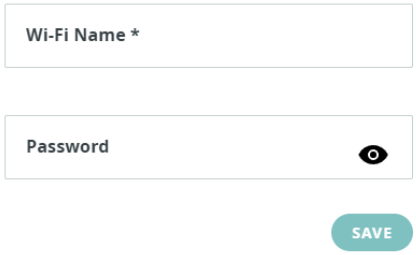
ASSOCIATE nu met het Device van uw keuze (in een cloud kunnen er meerdere devices gelinkt zijn, maar meestal zal dit het device zijn dat je zojuist gelinkt hebt – de nano 33 iot dus.
Je linkt hier uw Thing met uw Device



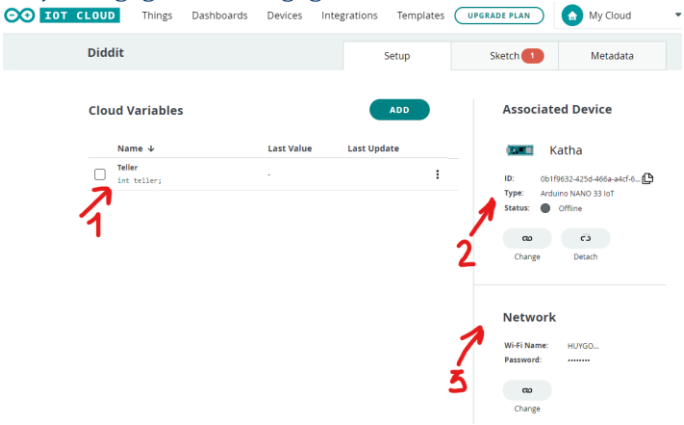
Klik op Configure om het device te verbinden met uw Wifi netwerk



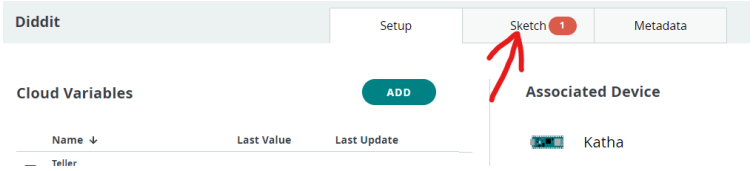
Vul nu de naam van uw Wifi netwerk en het bijpassende Password in en klik op Save



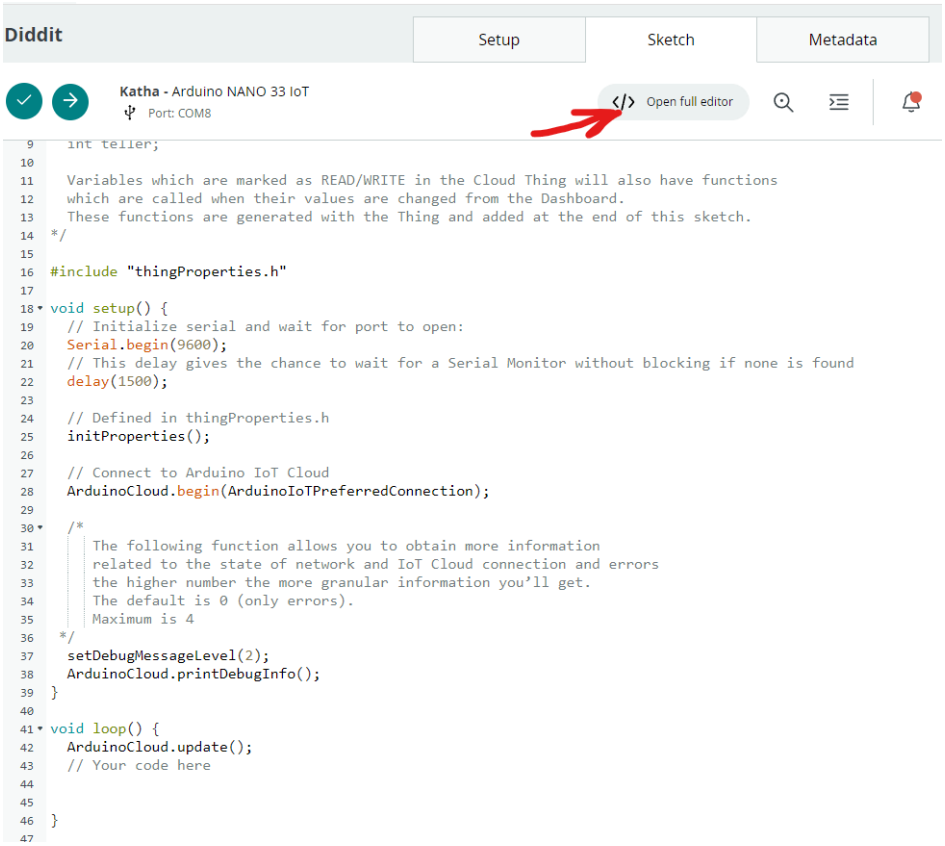
Nu hebben we dus een (1) variabele gedeclareerd (de variabele die we delen tussen de cloud en de arduino). (2) We hebben onze Thing gelinkt met ons device (de nano 33 iot). (3) En we hebben de juiste gegevens meegegeven om ons device te verbinden met ons Wifi netwerk.



5. De code van het Arduino programma
   - Open nu het tabblad 'Sketch'



Je ziet hier nu een Arduino programma staan met een duidelijk herkenbaar void setup() en void loop() gedeelte. Het lukt zeker ook om in deze omgeving te blijven, maar wij openen liever de Full Editor omdat deze ons wat meer mogelijkheden geeft.



Automatisch wordt nu uw online Arduino IDE geopend. Deze in ook gelinkt aan uw google of Arduino account.

Voordelen online Arduino Editor:

- Geen nood voor een offline installatie – werkt op elke PC
- Al uw programma's en versie worden online opgeslagen
- Zeer veel bibliotheken zijn reeds voor-geïnstalleerd

Nadelen

- Arduino agent moet wel geïnstalleerd zijn
- Als andere versie van Arduino IDE of Seriële monitor geopend staan kan dit problemen geven

De code die je nodig hebt om de variabele te delen via Wifi is reeds voorgeschreven. Je moet enkel nog iets doen met die variabele om een effect te zien in de cloud.

We verhogen hier elke seconde de variabele teller met 1. Omdat het programma elke seconde deze variabele naar de cloud zal sturen moeten we vermijden om delay() te gebruiken omdat dit het programma een bepaalde tijd stillegt.

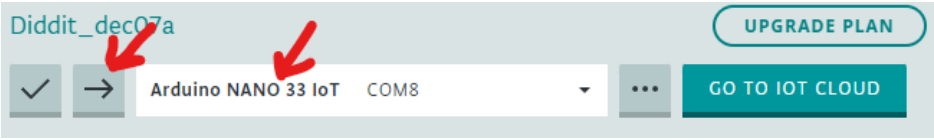
Het is iets complexer – maar veel beter om de interne millis() timer te gebruiken. Dat is een interne teller die elke miliseconde met 1 verhoogt. Als we wachten tot die met 1000 verhoogt is zijn we zeker dat er 1 seconde gepasseerd is.

De regels de we hebben toegevoegd aan het programma werden gemarkeerd.

```
#include "thingProperties.h"
unsigned long previousMillis = 0;
```

```
void setup() {
  // Initialize serial and wait for port to open:
  Serial.begin(9600);
  // This delay gives the chance to wait for a Serial Monitor without blocking if
none is found
  delay(1500);

  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  /*
     The following function allows you to obtain more information
     related to the state of network and IoT Cloud connection and errors
     the higher number the more granular information you'll get.
     The default is 0 (only errors).
     Maximum is 4
 */
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}

void loop() {
  ArduinoCloud.update();

  unsigned long currentMillis = millis();

  if ((currentMillis - previousMillis) >= 1000) {
    // Deze code wordt 1x/sec uitgevoerd - zonder gebruik te maken van delay()
    previousMillis = currentMillis;
    teller = teller + 1;   // Verhoog de waarde van de variabele teller met 1
  }
 }
```

Pas bovenstaande regels code aan, selecteer de juiste com poort voor de Arduino nano 33 iot en klik op "upload and save"



Als alle code correct naar de Arduino nano gestuurd werd – zie je iets gelijkaardigs onderaan uw scherm:

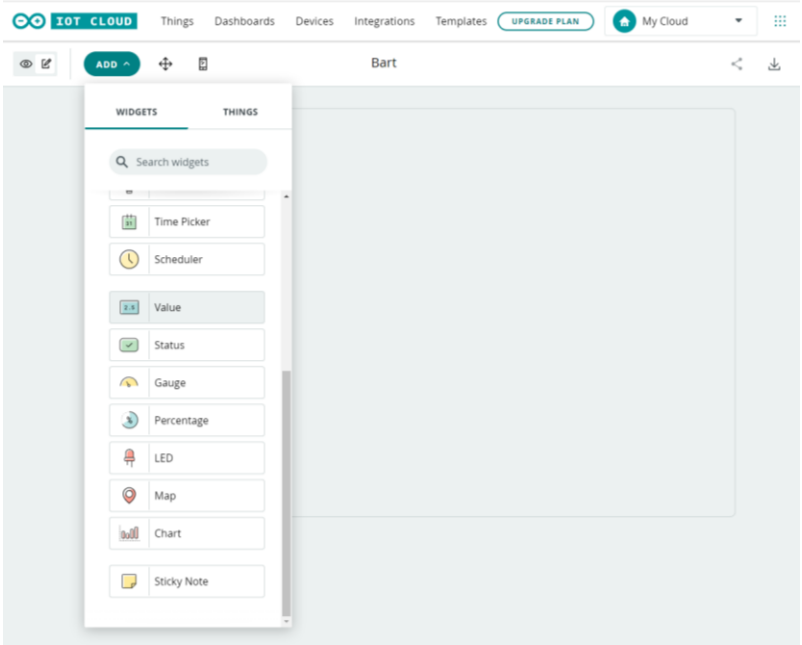
```
Success: Saved on your online Sketchbook and done uploading Diddit_dec07a
checksumBuffer(start_addr=0x18000, size=0x1000) = c049
checksumBuffer(start_addr=0x19000, size=0x59c) = 87c4
Verify successful
done in 0.090 seconds
CPU reset.
readWord(addr=0)=0x20007ffc
readWord(addr=0xe000ed00)=0x410cc601
readWord(addr=0x41002018)=0x10010305
writeWord(addr=0xe000ed0c,value=0x5fa0004)
```

Ga nu terug naar de arduino cloud

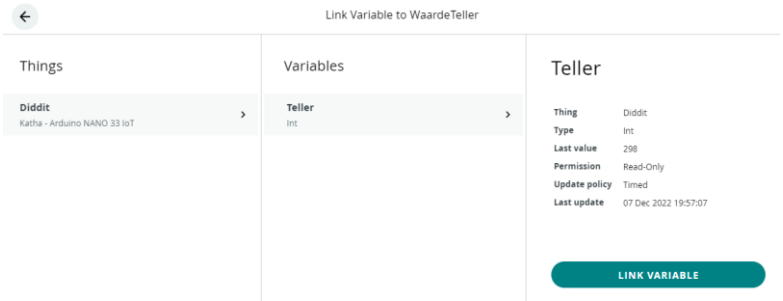
6. Dashboard

Selecteer het tabblad 'dashboard'

Klik op BUILD DASHBOARD
Geef uw dashboard een naam van uw keuze
Selecteer ADD en kies Value



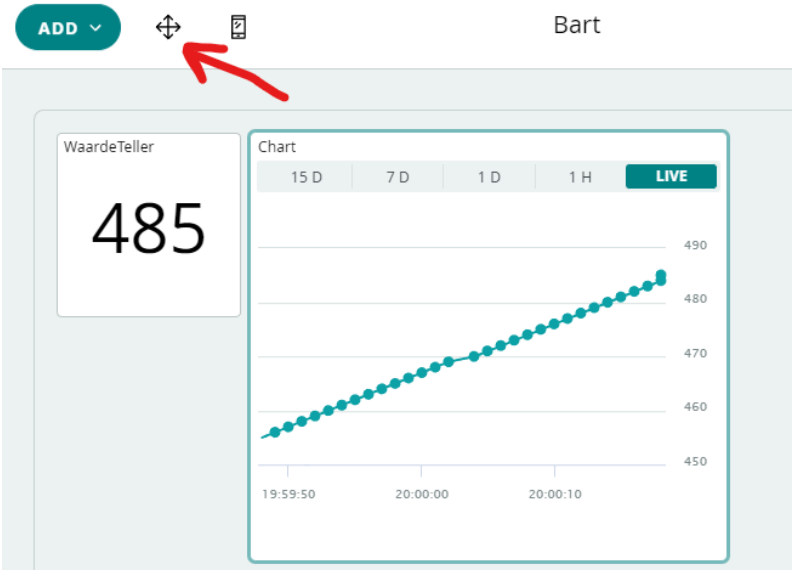
Link deze widget aan de variabele 'teller'



Klik op 'done'

Je ziet nu heel mooi de momentele waarde van uw variabele teller.

Klik op ADD – Chart
Link ook deze widget aan de zelfde variabele teller



Je ziet nu ook een voorstelling van de teller waarde in functie van de tijd
Via het pijltjeskruis kan je alle widgets ordenen.

Proficiat – je hebt nu uw eerste Iot applicatie gemaakt.