Modelspoor Club Oost Brabant

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Arduino programmeren
workshop

Datum: 17-3-2026
Auteur: Wim van der Linden

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Inhoudsopgave

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Lesprogramma

Les 1: Basis

- Wat is een Arduino
- Wat kun je ermee
- Basisbegrippen
- Arduino ontwikkelomgeving (Arduino IDE)
- Programmeer Instructies
- Opzet van een programma
- Compileren
- Je eerste programma (BuildIn Led blinking)

Les 2: basis programma's

- Vragen
- Led aansturen
- Andreas kruis
- Servo aansturen
- Stappen motor aansturen
- UART communicatie tussen twee Arduino's
- I2C communicatie tussen twee Arduino's

Les 3: Aansturing van acties

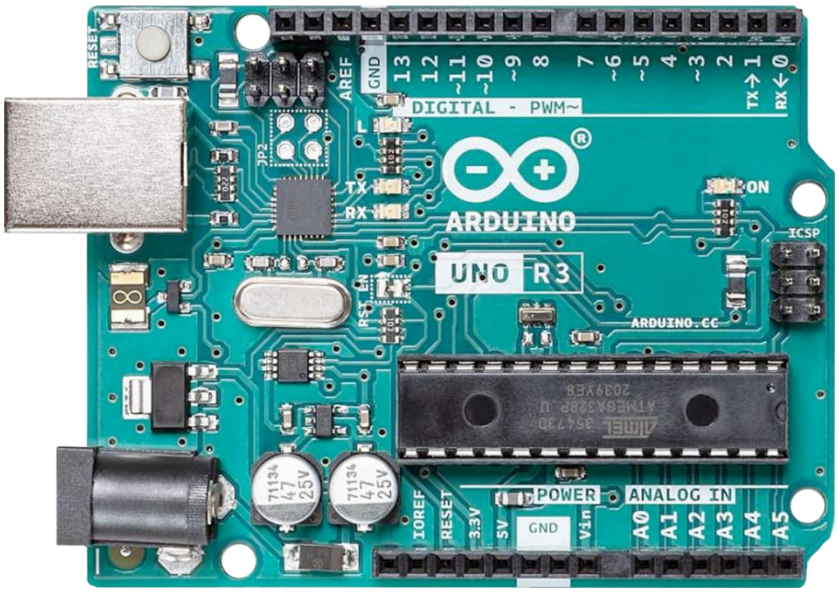
- Vragen
- Input /output
    - Digitaal (1 of 0) met wat kan ik daarmee
    - Analoog (potmeter) met wat ik daarmee
- Simulatie tool

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

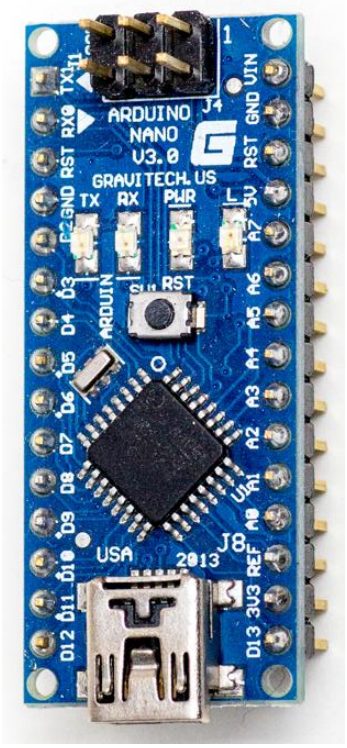
Wat is een Arduino?

Een Arduino is een merk. Het bevat een hele range aan PIC, Peripheral Interface Controller, processorboards met shields. Een shield is een board met een speciale functie die op een Processorboard gestoken kan worden. Dat wordt Stackable, stappelen, genoemd.

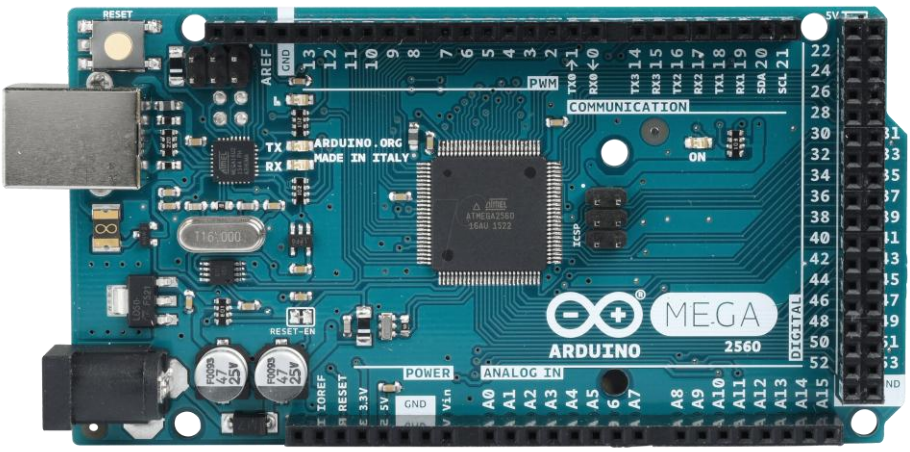
Voorbeelden van Arduino processorboards zijn:



Figuur 1 Arduino UNO

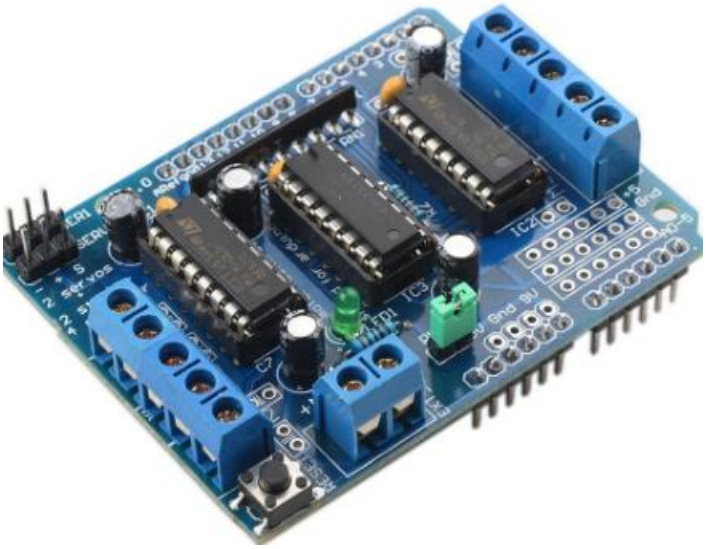


Figuur 2Arduino Nano

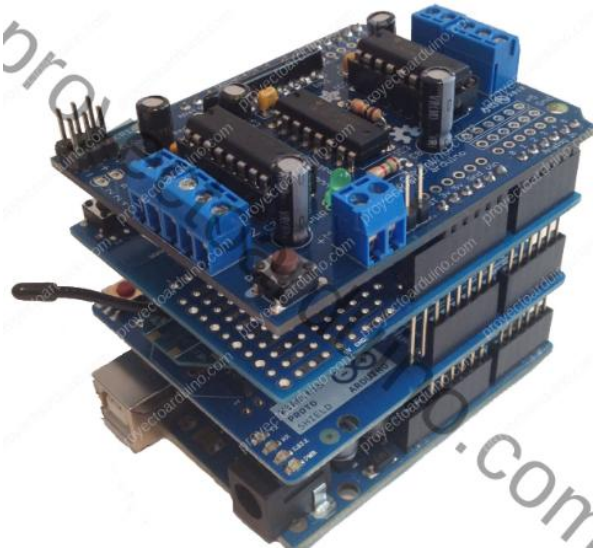


Figuur 3 Arduino MEGA

Voorbeelden van shields:



Figuur 5 Motor driver



Figuur 4 Stackable

Voor meer informatie kijk op de Arduino website: https://www.arduino.cc/

Of kijk op deze link waar voor uitleg op staat.
https://www.conrad.nl/nl/inspiratie/development-kits-en-bouwpakketten/arduino.html

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Arduino Nano

Voor onze workshop gebruiken we de Arduino Nano. Het verschil met de andere Arduino processorboards is onder andere het aantal mogelijkheden, geheugen en kloksnelheid.

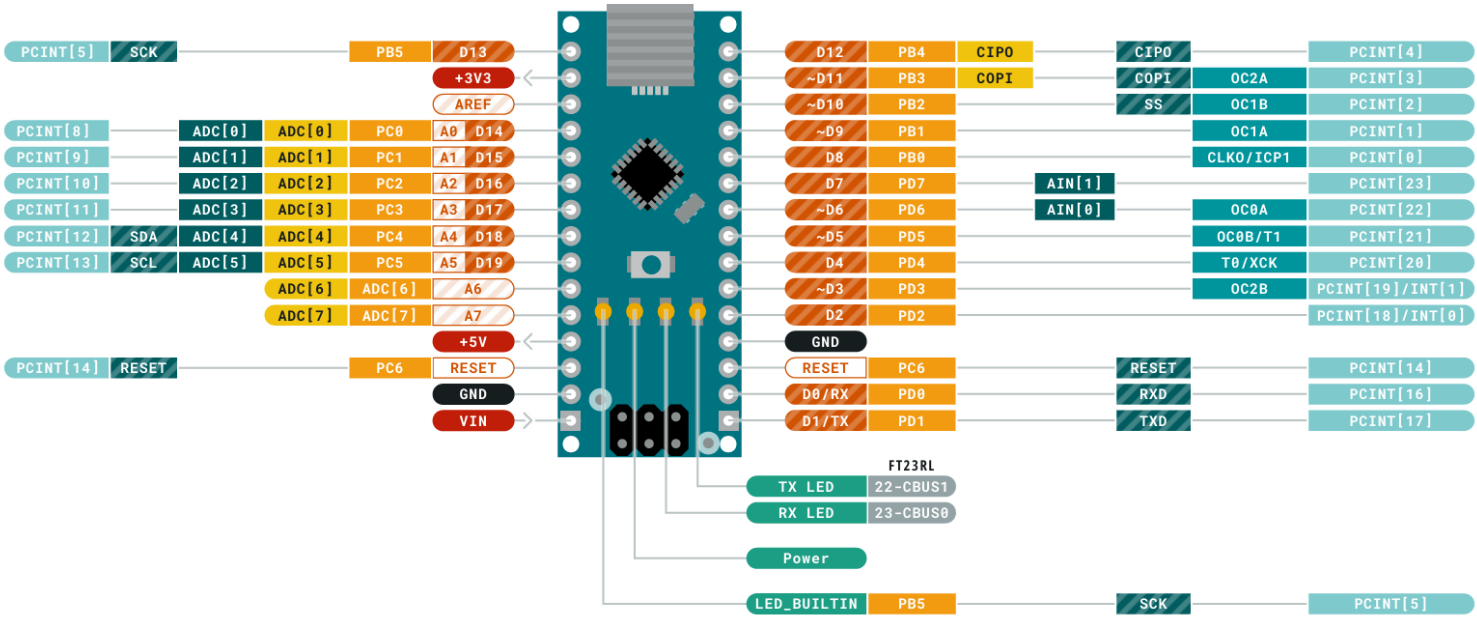
Wat is een Peripheral Interface Controller of PIC?

Het is een microprocessor chip met een groot aantal pinnen die middels een software instructie in een bepaalde modus gezet kan worden. Een dergelijke chip heeft, om te kunnen werken, nog een aantal zaken nodig zoals een Kristal als klok. Daarnaast is het natuurlijk handig als de pinnen ook makkelijk bereikbaar zijn. Arduino heeft deze microprocessor op een board gezet met deze componenten om het handig te maken.

Welke modus zijn er onder andere?

- Input
- Output
- PWM (Pulse With Modulation)
- Analoog in
- TX en RX (seriële communicatie – UART) SoftwareSerial
  #include <SoftwareSerial.h>
- SDA en SLC (voor I2C verbinding)
  #include <Wire.h>

Wat je hier kan komt later aan de orde. In de onderstaande afbeelding is, zoals dat heet, de Pin layout weergegeven. In dit geval van de Arduino Nano.



Kijk voor meer info op: https://docs.arduino.cc/resources/pinouts/A000005-full-pinout.pdf

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Waar is de Arduino voor te gebruiken

Met een Arduino kan het volgende op een relatieve wijze gemaakt worden.

- Led aansturing
  Simpel een led aan of uit zetten. Knipperen. Maar ook bijvoorbeeld het maken van een andreas kruis. Of het in en uit faden van een led.
  Met een led strip met een ws2812b. Dan kan elk led individueel aangestuurd worden.
  Zie: https://www.tweaking4all.nl/hardware/arduino/arduino-ws2812-led/
  #include <Adafruit_NeoPixel.h>
- Servo aansturing
  Een servo kan voor tal van zaken ingezet worden. Met een Arduino kan heel eenvoudig een servo bediend worden. Ook het afstellen van een servo om precies op het juiste punt te starten en te stoppen kan eenvoudig gerealiseerd worden.
  Zie: https://docs.arduino.cc/learn/electronics/servo-motors/
  #include <Servo.h>
- Stappen motor aansturing
  Een stappen motor kan je stapjes laten zetten. Zo kun je heel gecontroleerd iets in beweging krijgen. Met een Arduino kan zeer eenvoudig een stappen motor bediend worden. Bijvoorbeeld met een potmeter of op basis van sensoren die begin en eind aangeven.
  Meer info: https://docs.arduino.cc/learn/electronics/stepper-motors/
  #include <Stepper.h>
- DC motor aansturing
  DC motoren kunnen met een variabele spanning aangestuurd worden. Maar beter te regelen zijn ze wanneer deze met een PWM signaal aangestuurd worden. Een Arduino heeft poorten die standaard een PWM signaal kunnen uitgeven. Dit kan vervolgens verstrekt worden met een driver. Zo kan een DC motor heel gecontroleerd bestuurd worden. Locomotieven worden met DCC ook via een PWM aangestuurd.
  https://support.arduino.cc/hc/en-us/articles/9350537961500-Use-PWM-output-with-Arduino
  No include.

Daarnaast heeft een Arduino de mogelijkheid om via een bus (I2C) meerdere devices aan te sturen. Elke device heeft een adres op deze bus. Middels het aanspreken van dat adres kan een van de devices aangesproken worden. Zo kun je op deze bus een led scherm aansluiten en daar teksten op zetten.
#include <Wire.h>

Daarnaast beschikt de Arduino ook over de RX / TX (UART) bus. Een seriële bus voor communicatie.
#include <SoftwareSerial.h>

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Een Arduino programmeren

Een Arduino wordt geprogrammeerd in de taal C++. Voor het programmeren van een Arduino wordt de Arduino IDE gebruikt. Deze is te downloaden via deze link: https://docs.arduino.cc/software/ide/
Hier vind je ook meer informatie over de werking van dit programma.

Let op: de taal is hoofdletter gevoelig. De statements worden voornamelijk met kleine letters geschreven. Een variabele of naam met kleine of hoofdletters zijn aparte variabelen.

Beschikbare Arduino tutorial

Voor beginners is er de volgende tutorial: https://www.tutorialspoint.com/arduino/index.htm

Er wordt verwezen naar de volgende tutorial als basis:
Learn C++ Programming: https://www.tutorialspoint.com/cplusplus/index.htm

Voor diegene die geïnteresseerd zijn: In deze tutorial wordt uitgelegd hoe een processor werkt:
https://www.tutorialspoint.com/microprocessor/index.htm

Dit is de algemene link naar tutorials: https://www.tutorialspoint.com/top-categories.htm

Enkele Basis begrippen

De volgende begrippen zijn relevant:

- Bit kan de waarde 0 of 1 bevatten
- Byte bevat 8 bits en kan de waarde tussen -127 en +127 bevatten. Bit 8 is het sign bit.
- Integer -INT- bevat 16 bits waarvan het hoogste bit het sign bit is
- Unsigned INT heeft enkel een positieve waarde met een max van 64k.
- Char wordt gebruikt voor het opslaan van een karakter.
- Unsigned Char kan alleen een positieve waarde bevatten met een max van 256.
- Float is kan een floating comma getal bevatten.

Een statement is een instructie in een programma.
Een subroutine is een functie de hergebruikt kan worden.
Een Arduino programma wordt een Sketch genoemd.

Enkele noodzakelijke tools

Arduino IDE: deze kun je hier downloaden:
https://support.arduino.cc/hc/en-us/articles/360019833020-Download-and-install-Arduino-IDE

Simulatie omgevingen

Simulatie omgeving: https://simulide.com/p/downloads/
Wokwi omgeving: https://wokwi.com/

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Arduino programma structuur

Een programma heeft een vaste opbouw. Het wordt een Sketch genoemd en heeft de extensie ino.
De opbouw van een Sketch is als volgt:

- Beschrijving programma (commentaar)
- Include blok
- Define blok
- Globale variabelen en constanten
- Constanten
- Array declaraties (optioneel)
- Strings
- Struct declaraties (optioneel)
- initialisatie
- Setup routine
- Main loop routine
- Subroutines

Onderstaand zijn de verschillende onderdelen van een Sketch uitgelegd.

Commentaar

Een programma dient van duidelijk commentaar voorzien te worden. Je moet er rekening mee houden dat andere ook het programma moeten kunnen begrijpen. Maar ook voor jezelf is het van belang. Want als je een lange tijd het programma niet onder ogen hebt gehad kan het wel eens heel moeilijk worden om de juiste werking weer te begrijpen.
Commentaar kan op twee manieren opgenomen worden. Als blok of als regel.

Syntax

Tekstblok	/* tekst */
Tekstregel	// tekst

Een goede gewoonte is om een programma aan het begin te voorzien van een tekstblok met daarin opgenomen:

```
/*
 auteur  : <naam>
 datum   : <datum eerste versie>
 versie  : <versie nummer>
 functie : <omschrijving van de functie>
 input   : <input variabelen
 output  : <output variabelen>
*/
```

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

#Include

Na het commentaarblok komen de includes van de libraries die gebruikt worden in het programma. Een library bevat functies die in het programma gebruikt kunnen worden. Dit maakt het programmeren een stuk eenvoudiger omdat de code voor de functionaliteit op deze manier hergebruikt kan worden.

Syntax
#include <naam van library>

De “naam van library” heeft meestal als extensie .h
Voorbeeld #include <EEPROM.h>

Met deze include wordt het bijvoorbeeld eenvoudig om data uit een EEPROM adres te lezen of er naartoe te schrijven. Met een instructie als: Count = EEPROM.read (address); kan de inhoud van het opgegeven <address> gelezen en geplaatst worden in de variabele Count.

#define

Na de includes komen de #defines van de variabelen die in het programma gebruikt gaan worden.

Syntax
#define <naam> <waarde>
De “naam” is de naam die in het programma gebruikt wordt.

Een #define wordt gebruikt om het lezen van de code makkelijker te maken.

Voorbeeld:
#define SLAVE_TABLE_EEPROM_START 0 // start adres slave tabel in eeprom = 0

In plaats van 0 kan in de code SLAVE_TABLE_EEPROM_START gebruikt worden. Dat geeft meteen de bedoeling aan. Namelijk het startadres van de Slave tabel in EEPROM. Maar ook als het aangepast moet worden kan dit op een plaats gebeuren en hoeft niet het hele programma doorlopen te worden.

Globale variabelen

Na de defines komen de globale variabelen.

Syntax
<data type> <naam> (= waarde);

Er zijn verschillende data typen: Char, Byte, Bool, unsigned char, etc.

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Verschil tussen Char en Unsigned Char.

- Een Char is 8 bits. Dat is gelijk aan een Byte. Maar het hoogste bit wordt gebruikt als sign. Dus kan de waarde maximaal + 127 en minimaal - 127 zijn.
- Een unsigned Char is ook 8 bits maar heeft geen sign. Dus kan een waarde hebben van 0 tot max 256.

Hier vind je ze allemaal: https://www.tutorialspoint.com/arduino/arduino_data_types.htm

Met "= waarde" kan een initiële waarde meegegeven worden aan een variabele.
Voorbeeld: unsigned char T = 0;
De variabele T kan in het programma gebruikt worden en heeft initieel de waarde 0.

Array

Array is een aantal van dezelfde data types.

Syntax
<Data type> <naam> [aantal];

<Data type> = data type van het array.
<naam> = de naam van het array.
[aantal] = de diepte van het array.

Strings

Strings worden gebruikt om tekst in op te slaan. Ze kunnen worden gebruikt om tekst weer te geven op een LCD of in het Arduino IDE Seriële Monitor-venster. Strings zijn ook nuttig voor het opslaan van gebruikersinvoer. Bijvoorbeeld, de tekens die een gebruiker op een toetsenbord typt.

Er zijn twee type strings:

- Arrays van karakters, die hetzelfde zijn als de strings die in C-programmering worden gebruikt.
- De Arduino String, waarmee we een stringobject in een schets kunnen gebruiken.

String char array

Het eerste type string is de string die een reeks karakters van het type char bevat. Een opeenvolgende reeks van hetzelfde type variabele opgeslagen in het geheugen. Een string is een array van char-variabelen.

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Voorbeeld:

```
char my_str[6];          // an array big enough for a 5 character string

Serial.begin(9600);
my_str[0] = 'H';         // the string consists of 5 characters
my_str[1] = 'e';
my_str[2] = 'l';
my_str[3] = 'l';
my_str[4] = 'o';
my_str[5] = 0;           // 6th array element is a null terminator
Serial.println (my_str);
```

Je kunt ook een string als volgt definiëren: char like[] = "I like coffee and cake"; // create a string

Functies voor strings

Op strings kunnen de volgende functies uitgevoerd worden.

String inkorten

Dit kan met het vullen van een terminating zero: like[21] = 0; // terminate the string
Een string telt altijd vanaf 0.

Lengte van een string

```
num = strlen(str);
```

De lengte is tot de termination 0.

Lengte van de string array

Deze geeft de lengte van het totale array met ook de termination 0

```
num = sizeof(str);
```

Kopieer een string

```
strcpy(out_str, str);
```

String wordt gekopieerd van out_str naar str.

Concatenate twee strings

Plak twee string aan elkaar.

```
strcat(out_str, " sketch.");
```

Grens van array of string

Bij het werken met strings en arrays is het erg belangrijk om binnen de grenzen van strings of arrays te werken. Als de array te klein is gemaakt en we proberen een string die groter is dan de array te kopiëren, wordt de string over het uiteinde van de array gekopieerd. Het geheugen voorbij het einde

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

van de array kan andere belangrijke gegevens bevatten die in de schets worden gebruikt, die vervolgens door onze string worden overschreven. De schets kan dan crashen of onverwacht gedrag veroorzaken.

String object

Een object is een construct dat zowel data als functies bevat. Een string-object kan net als een variabele worden gemaakt en een waarde of string worden toegewezen. Het String-object bevat functies (die in object georiënteerd programmeren (OOP) "methoden" worden genoemd) die werken op de stringgegevens in het String-object.

```
string my_str = "This is my string.";
```

Er zijn verschillende methods die gebruikt kunnen worden in relatie met String objects.

toUpperCase

Converteer de inhoud naar hoofdletters

toLowerCase

converteer de inhoud naar kleine letters

replace()

Vervang een deel in de string

length()

Lengte van de string

Struct

Struct is een verzameling aan data types.

Syntax

Definitie van het struct:

```
struct <naam> {
        <data type> <naam> [aantal];
};
```

<naam> = is de naam van de struct
<data type> = de definitie van het data type
<naam> = is de naam van het data type
[aantal] = indien dit een array is dan het aantal van dat data type

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Aanmaken van het struct: struct event EventQ[aantal];

Voorbeeld:

```
struct event {
   uint8_t slaveID;
   uint8_t deviceAddr;
   uint8_t len;
   uint8_t data[32];
};
```

struct event EventQ [Max_aantal];
de struct EventQ is opgebouwd zoals de definitie van struct event.

uint8_t is een 8 bits variabelen. Dus een unsigned byte.

Variabelen

Variabelen kunnen aangemaakt worden als lokale, globale of volatile variabelen. Lokale variabelen worden in een routine aangemaakt en gebruikt. Buiten deze routine bestaan ze dan niet.

Globale variabelen

Globale variabelen worden gedeclareerd buiten de subroutines, ook wel functies en zijn door het hele programma te gebruiken.

Syntax
<data type> <naam> (=waarde);

Lokale variabelen

Lokale variabelen worden binnen een routine aangemaakt en gebruikt. Er wordt dezelfde syntax gebruikt als bij globale variabelen.

Syntax
<data type> <naam> (=waarde);

Voorbeeld:

```
void loop () {
        // lokale variabelen
        int x , y ;
        int z ;

        x = 0;
        y = 0;
```

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

```
        z = 10;
}
```

volatile variabelen

Deze staan ook op het niveau van Globale variabelen. Dus buiten de routines.

Syntax
volatile <data type> <naam> (=waarde)

Met volatile wordt aangegeven dat deze variabelen door een andere routine aangepast kan worden zonder dat de compiler dat tijdens het compileren van de code kan weten.

Dat is bijvoorbeeld het geval van interrupts. Tijdens een interrupt kan een parameter aangepast. Een compiler optimaliseert tijdens het compileren de code. Daarbij deze mogelijk de variabelen als niet een aan te passen variabelen zien en dus daarmee een fout introduceren omdat de variabelen wel kan wijzigen. Om dit te voorkomen moet je de variabele volatile meegegeven.

Constanten

Een constante variabele is een variabele met een vaste waarde. Globaal of lokaal gedefinieerd.

Syntax
const <data type> <naam> = waarde;

Voorbeelden:

```
const int ledPin = 13;
const float pi = 3.14159;
```

Initialisatie

Sommige functies moeten geinitialiseerd worden zoals bijvoorbeeld een I2C display of een Stepper driver.

Syntax
<naam> <functie> (initialisatie parameters);

Het aantal parameters hangt van de functie af.

Voorbeeld:

```
// initialize LCD1602 I2C display instellen (adres 0x27 of 0x3F afhankelijk van je display)
LiquidCrystal_I2C lcd(0x27, SCL, SDA);
```

Merk op dat hier de SCL en SDA eerder als een #define zijn gedefinieerd.

```
#define SCL 4          // I2C clock
```

```
#define SDA 5          // I2C data line
```

Beter zou zijn geweest als ook het adres met een #define was gedefinieerd.

```
// initialize the stepper library on pins 8, 10, 9, 11,:
Stepper myStepper(stepsPerRevolution, 8, 10, 9, 11);
```

Merk op dat hier stepsPerRevolution eerder als een #define is gedefinieerd.

```
#define stepsPerRevolution 800
```

Beter zou zijn geweest als ook de poorten met een #define waren gedefinieerd.

Subroutine / functie declaraties

Om het programma overzichtelijk en leesbaar te houden wordt gewerkt met subroutines. Ook wel functies genoemd. De subroutines worden onder de main Loop routine gezet. Op deze locatie moeten deze routines gedefinieerd worden.

Syntax

Void <naam subroutine> (<parameters>);
<return waarde data type> <naam subroutine> (<parameters>);
Parameter = <data type> <naam parameter>

Voorbeelden:

```
void onI2CReceive (int len);              // Callback bij I2C ontvangst
void onI2CReques t();                     // Callback bij I2C request
uint8_t resolveSlave (uint8_t deviceAddr);  // Zoek slaveID bij gegeven device adres
```

Setup routine

Als het programma gestart wordt zal als eerste deze routine uitgevoerd worden. Hierin dienen alle variabelen en poorten, etc voor gebruik geinitialiseerd te worden. Deze routine wordt maar een keer aangeroepen.

Syntax

```
void setup () {
    // lokale variabelen

    Serial.begin (9600);
    identifyMyself ();      // identify Arduino and program

    Hier komt je code
}
```

Void (leeg) geeft aan dat er geen waarden teruggegeven worden.

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

identificatie

In deze routine komt minimaal na de initialisatie van de Serial het volgende blok code <identificatie>.

```
/***** identificatie Arduino en Programma *****/
void identifyMyself() {
        Serial.println ("Arduino: <type+versie>");
        Serial.println ("Auteur: <naam>");
        Serial.println ("Programma: <naam>");
        Serial.println("Versie: <versienummer>");
         Serial.println("Functie: <functionele werking> ");
        Serial.println("Input: <input waarden>");
        Serial.println("Output: <output waarden>");
}
/***** einde identificatie *****/
```

Hiermee identificeert de Arduino zich.

main Loop

Deze routine wordt herhaaldelijk doorlopen. Het is een oneindige loop.

Syntax

```
void loop () {
        // lokale variabelen

        Hier de code
}
```

Ook deze routine geeft geen waarde terug.

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Programma instructies

Arduino wordt geprogrammeerd in de taal C++. In de Routines komen de statements die de werking van het programma bepalen.

- Een statement wordt altijd afgesloten met punt comma ";".
- Een blok van bij elkaar horende code wordt aangegeven met { code blok }.
- Spaties hebben geen betekenis.
- In een subroutine / functie naam mogen geen spaties gebruikt worden.

Hieronder zijn de meest voorkomende instructies opgenomen.

Operator statements

Dit zijn de volgende bewerkingen:

- Arithmetic Operators
- Comparison Operators
- Boolean Operators
- Bitwise Operators
- Compound Operators

Arithmetic Operators

A=10 en B=20

Naam	Operator	Omschrijving	Voorbeeld
ssignment operator	=	Slaat de waarde rechts van het gelijkteken op in de variabele links van het gelijkteken.	A = B
addition	+	Voegt twee operanden toe	A + B will give 30
subtraction	-	Trek de tweede operand af van de eerste	A - B will give -10
multiplication	*	Vermenigvuldig beide operanden	A * B will give 200
division	/	Deel de teller door de noemer	B / A will give 2
modulo	%	Modulusoperator en rest van na een gehele deling	B % A will give 0

```
void loop () {
        int a = 9, b = 4, c;
                c = a + b;
```

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

```
        c = a - b;
        c = a * b;
        c = a / b;
        c = a % b;
}
```

Comparison Operators

A=10 en B=20

Naam	Operator	Omschrijving	Voorbeeld
equal to	==	Controleert of de waarde van twee operanden gelijk is of niet, en als dat zo is, wordt de voorwaarde waar.	(A == B) is not true
not equal to	!=	Controleert of de waarde van twee operanden gelijk is of niet; als waarden niet gelijk zijn, wordt de voorwaarde waar.	(A != B) is true
less than	<	Controleert of de waarde van de linker operand kleiner is dan de waarde van de rechter operand; als ja, dan wordt de voorwaarde waar.	(A < B) is true
greater than	>	Controleert of de waarde van de linker operand groter is dan de waarde van de rechter operand, en als dat zo is, dan wordt de voorwaarde waar.	(A > B) is not true
less than or equal to	<=	Controleert of de waarde van de linker operand kleiner is dan of gelijk aan de waarde van de rechter operand; als dat zo is, dan wordt de voorwaarde waar.	(A <= B) is true
greater than or equal to	>=	Controleert of de waarde van de linker operand groter is dan of gelijk aan de waarde van de rechter operand; als ja, dan wordt de voorwaarde waar.	(A >= B) is not true

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Boolean Operators

A=10 en B=20

Naam	Operator	Omschrijving	Voorbeeld
and	&&	Noemt Logical AND operator. Als beide operanden niet nul zijn, dan wordt de voorwaarde waar.	(A && B) is true
or	||	Wordt Logical OR Operator genoemd. Als een van de twee operanden niet nul is, dan wordt de voorwaarde waar.	(A || B) is true
not	!	Wordt Logical NOT Operator genoemd. Keert de logische toestand van zijn operand om. Als een voorwaarde waar is, dan zal de Logische NIET-operator onwaar maken.	!(A && B) is false

Bitwise Operators

A=60 en B=13

Naam	Operator	Omschrijving	Voorbeeld
and	&	Binaire EN Operator kopiëren een bit naar het resultaat als deze in beide operanden aanwezig is.	(A & B) will give 12 which is 0000 1100
or	|	Binaire OF-operator kopieert een bit als deze in een van beide operanden bestaat	(A | B) will give 61 which is 0011 1101
xor	^	De binaire XOR-operator kopieert het bit als het in één operand is gezet, maar niet in beide.	(A ^ B) will give 49 which is 0011 0001
not	~	De complementoperator van binaire enen is unair en heeft het effect van het 'omdraaien' van bits.	(~A) will give -60 which is 1100 0011
shift left	<<	Binaire linker shift-operator. De waarde van de linker operanden wordt naar links verplaatst door het aantal bits dat door het rechter operand is gespecificeerd.	A << 2 will give 240 which is 1111 0000

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

shift right	>>	Binaire rechterschuifoperator. De waarde van de linker operanden wordt naar rechts verplaatst door het aantal bits dat door het rechter operand is gespecificeerd.	A >> 2 will give 15 which is 0000 1111

Compound Operators

A=10 en B=20

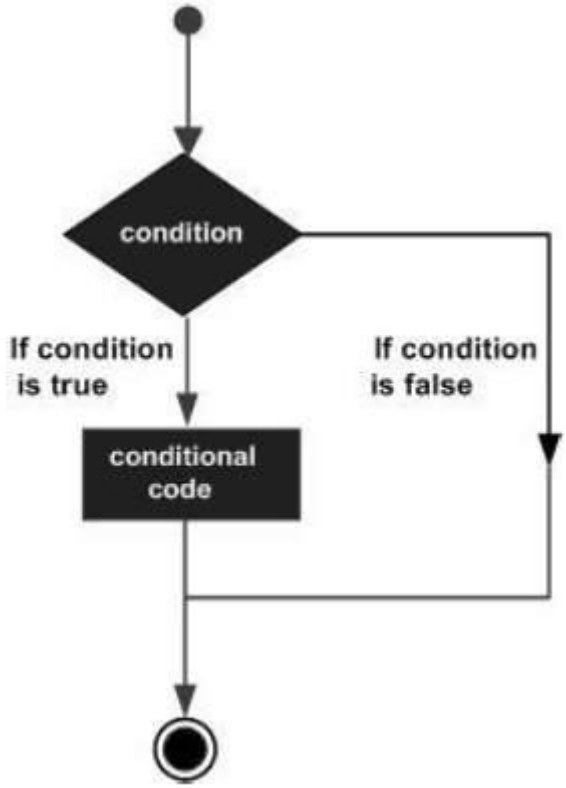
Naam	Operator	Omschrijving	Voorbeeld
increment	++;	Increment-operator, verhoogt de geheel getal met één.	A++; will give 11
decrement	--	Decrement-operator, verlaagt de geheel getal met één.	A-- will give 9
compound addition	+;=	Voeg AND-toewijzingsoperator toe. Het voegt rechter operand toe aan de linker operand en wijst het resultaat toe aan linker operand	B+;= A is equivalent to B = B+; A
compound subtraction	-=	Trek EN de opdrachtoperator af. Het trekt rechteroperand af van de linkeroperand en wijst het resultaat toe aan linker operand	B -= A is equivalent to B = B - A
compound multiplication	*=	Vermenigvuldigen EN toewijzingsoperator. Het vermenigvuldigt rechter operand met de linker operand en wijst het resultaat toe aan linker operand	B*= A is equivalent to B = B* A
compound division	/=	Verdeel EN toewijzingsoperator. Het deelt linker operand met de rechter operand en wijst het resultaat toe aan linker operand	B /= A is equivalent to B = B / A
compound modulo	%=	Modulus EN toewijzingsoperator. Het neemt de modulus met behulp van twee operanden en wijst het resultaat toe aan de linker operand	B %= A is equivalent to B = B % A
compound bitwise or	|=	bitgewijs inclusief OR en toewijzingsoperator	A |= 2 is same as A = A | 2
compound bitwise and	&=	Bitgewijze AND-toewijzingsoperator	A &= 2 is same as A = A & 2

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Control statements

Control Statements zijn elementen in broncode die de flow van programma-uitvoering aansturen.

Besluitvormingsstructuren vereisen dat de programmeur één of meer voorwaarden specificeert die door het programma geëvalueerd of getest moeten worden. Dit moet samen met een of meerdere statements worden uitgevoerd als de voorwaarde als waar wordt bevonden, en optioneel andere statements die worden uitgevoerd als de voorwaarde als onwaar wordt bevonden.



Er zijn de volgende control statements:

If statement

Een uitdrukking tussen haakjes die getoetst wordt op waarheid waarop een statement uitgevoerd wordt

```
if (expression)
  statement;
```

of blok van statements.

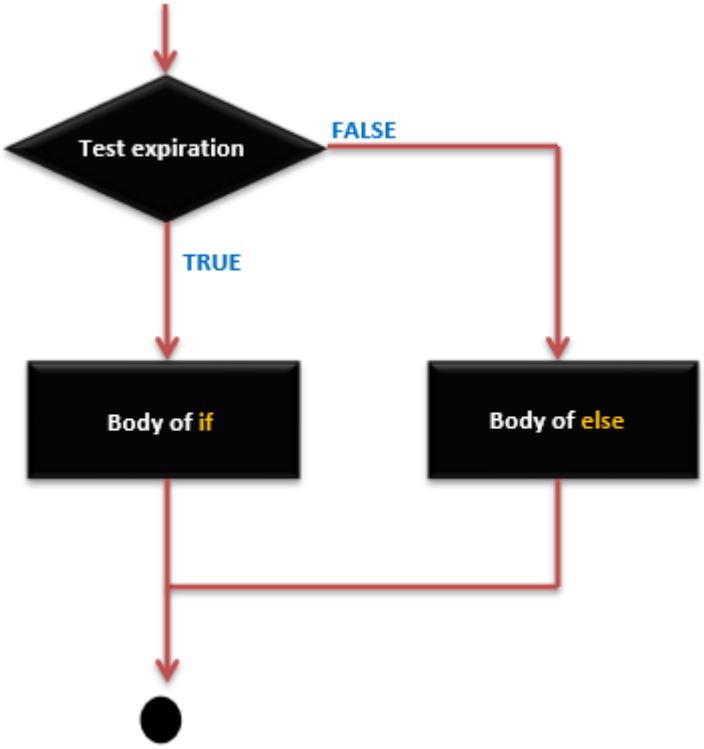
```
if (expression) {
  Block of statements;
}
```

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

If else statement

Een if-instructie kan gevolgd worden door een optionele else-instructie, die wordt uitgevoerd wanneer de uitdrukking onwaar is.

```
if (expression) {
  Block of statements;
}
else {
  Block of statements;
}
```



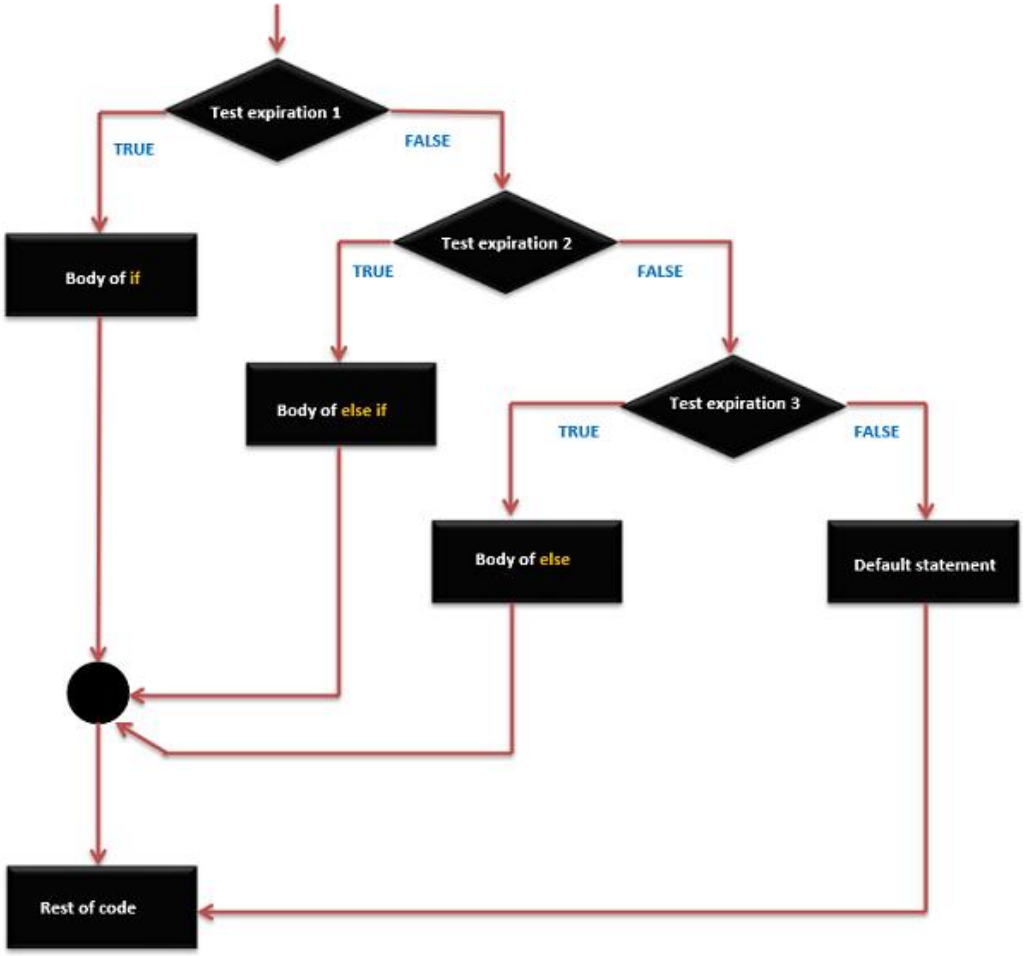
Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

If else if else statement

De if-uitspraak kan gevolgd worden door een optioneel anders if... else-stelling. Nuttig om verschillende condities te testen met behulp van single if... else.

```
if (expression_1) {
  Block of statements;
}

else if(expression_2) {
  Block of statements;
}
.
.
else {
  Block of statements;
}
```



Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

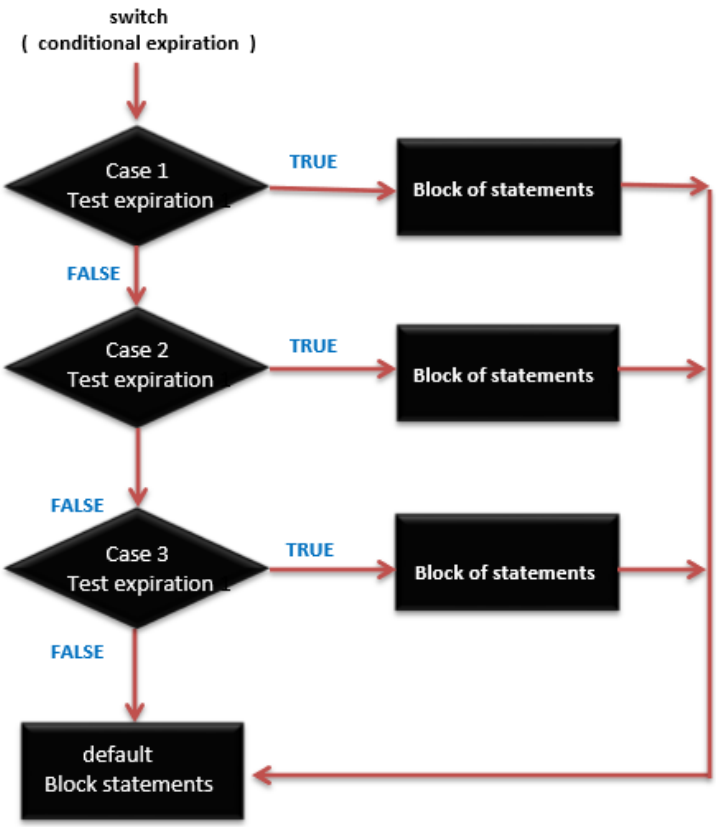
Switch case statement

Net als bij de if-statements, bepaalt switch... case de stroom van programma's door verschillende codes te specificeren die onder verschillende omstandigheden uitgevoerd moeten worden.

```
switch (variable) {
  case label:
  // statements
  break;
}

case label: {
  // statements
  break;
}

default: {
  // statements
  break;
}
```



Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Conditional operator

De conditionele operator? : is de enige ternaire operator in C.

Syntax
?

expression1 ? expression2 : expression3;

Expression1 wordt eerst geëvalueerd. Als de waarde waar is, wordt expressie2 geëvalueerd en wordt expressie3 genegeerd. Als expressie1 als onwaar wordt geëvalueerd, dan wordt expressie3 genomen en wordt expressie2 genegeerd. Het resultaat zal een waarde zijn van ofwel expressie2 of expressie3, afhankelijk van welke van hen als True wordt geëvalueerd.

Voorbeeld:

```
/* Vind de grootste waarde tussen a, b */
        max = ( a > b ) ? a : b;

/* Zet kleine letter om in hoofdletter*/
        c = ( c >= 'a' && c <= 'z' ) ? ( c - 32 ) : c;
```

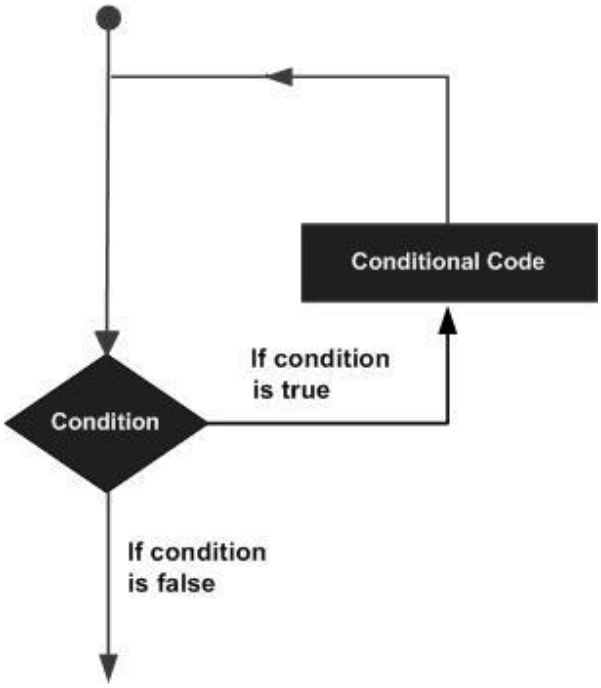
Regels voor Conditional operator

- expressie1 moet een scalair expressie zijn; expressie2 en expressie3 moeten zich aan een van de volgende regels houden.
    - Beide uitdrukkingen moeten van het rekenkundige type zijn.
    - Beide uitdrukkingen moeten van het void-type zijn. Moeten geen waarde terug geven. Dus kan geen functie zijn.

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Loop statements

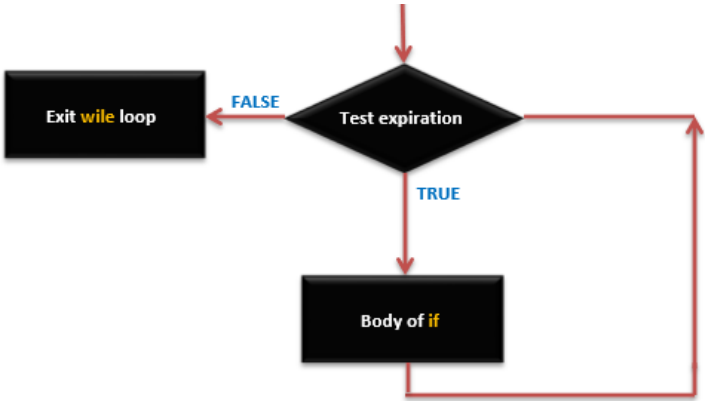
Programmeertalen bieden verschillende besturingsstructuren die complexere uitvoeringspaden mogelijk maken. Een lus-instructie stelt ons in staat een verklaring of groep van uitspraken meerdere keren uit te voeren.



While loop

Lussen zullen continu en oneindig herhalen totdat de uitdrukking binnen de haakjes () onwaar wordt. Er moet iets de geteste variabele veranderen, anders zal de while-lus nooit meer stoppen.

```
while(expression) {
        Block of statements;
}
```



Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

dowhile statement

De dowhile-lus lijkt op de while-lus. In de while-lus wordt de loop-continuation voorwaarde getest aan het begin van de lus voordat het lichaam van de lus wordt uitgevoerd. Bij de dowhile is dat aan het einde van de lus.

```
do {
        Block of statements;
}
while (expression);
```

for loop

Een for-lus voert statements een vooraf bepaald aantal keren uit. De controle-expressie voor de lus wordt geïnitialiseerd, getest en volledig uitgevoerd binnen de for-lus haakjes.

Elke for-lus heeft drie uitdrukkingen die de werking bepalen. Het volgende voorbeeld toont de algemene loop syntax. Let op dat de drie uitdrukkingen in de for-lus argument haakjes gescheiden zijn met puntkomma's.

```
for ( initialize; control; increment or decrement) {
        // statement block
}
```

Voorbeeld:

```
for (counter = 2; counter <= 9; counter++) {
        //statements block will executed 10 times
 }
```

Nested loop

De C-taal laat je toe om de ene lus binnen een andere te gebruiken. Het volgende voorbeeld illustreert het concept.

```
for ( initialize ;control; increment or decrement) {
        // statement block

        for ( initialize; control; increment or decrement) {
                // statement block
        }
}
```

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Voorbeeld:

```
for (counter = 0;counter <= 9;counter++) {
        //statements block will executed 10 times

        for (i = 0;i <= 99;i++) {
                //statements block will executed 100 times
        }
}
```

Infinite loop

Het is de lus zonder afsluitende voorwaarde, dus wordt de lus oneindig.

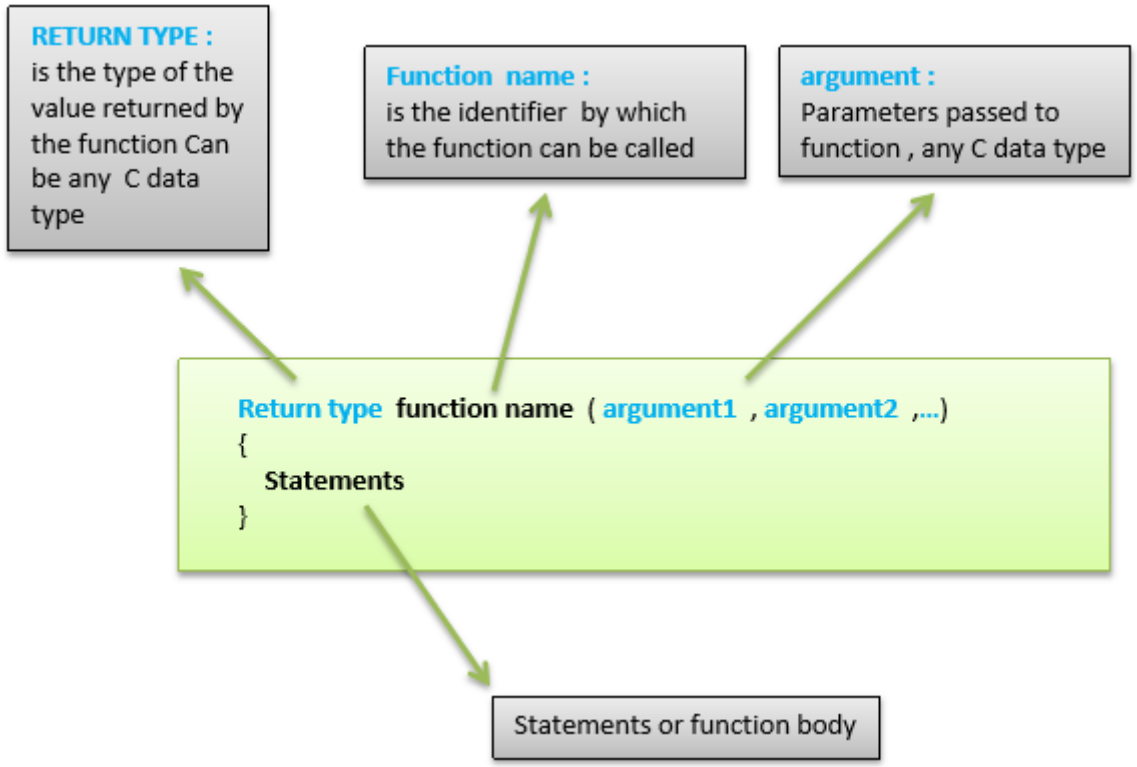
```
while(1) {
        // statement block
}
```

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Functions ook wel subroutines

Functies maken het mogelijk om programma's te structureren en op te bouwn in segmenten van code om individuele taken uit te voeren. Het typische geval voor het creëren van een functie is wanneer men dezelfde actie meerdere keren in een programma moet uitvoeren.

Er zijn twee vereiste functies in een Arduino-sketch of een programma, namelijk setup () en loop (). Andere functies moeten buiten de haakjes van deze twee functies worden aangemaakt.



- Functie-return type
- Functienaam
- Functie argument type: hier moet je de argumentnaam toevoegen
- Functie body: statements binnen de functie die worden uitgevoerd wanneer de functie wordt aangeroepen

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Voorbeeld:

```
int sum_func (int x, int y) {        // function declaration
        // lokale variabelen
        int z = 0;                   // declaratie met initialisatie

        // code segment
        z = x + y ;
        return z;                    // return the value
}
```

Voorbeeld van de aanroep:

```
void loop () {
        // lokale variabelen
        int result = 0 ;

        // code segment
        result = Sum_func (5,6) ;          // function call
}
```

Als de functie onder loop (); komt dan moet deze vooraf gedeclareerd worden. Dat wordt als volgt gedaan.

```
int sum_func (int , int ) ;          // function prototype
```

Dit hebben ook al eerder behandeld in het deel over de structuur van het programma.

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Standaard functies

Standaard functies komen ter beschikking wanneer een library is ge-include in het programma. Echter er zijn ook zogenaamde Build-in functies.

Build-in functies

Time

Er zijn twee time functions die de processor laat wachten:

Delay (ms); // ms = aantal milliseconde delay
delayMicroseconds (us); // us = aantal microseconde delay

Deze functies blokkeren alles. Het systeem staat werkelijk die tijd stil. Derhalve moet deze functie met zorgvuldigheid gebruikt worden.

pinMode () function

Statement voor het zetten van een port in een bepaalde mode.

Syntax
pinMode (<pin#>, <mode>);

<mode> kan zijn:

- INPUT
- INPUT_PULLUP
- OUTPUT

Opmerking
De ingangen kunnen “zweven”. Dat wil zeggen een niet duidelijk 1 of 0 hebben. Om dat te stabiliseren zijn er twee mogelijkheden:

1. Aan de ingang wordt een pull_up weerstand geplaatst. Dat is een 10k Ohm weerstand verbonden met de port en +5V. Of gnd (ground).
2. Gebruik maken van de interne pull_up weerstand met de mode INPUT_PULLUP.

digitalWrite () function

Deze functie kan alleen gebruikt worden met een port die vooraf met pinMode in OUTPUT mode is gezet.

Syntax
digitalWrite (<pin#>, HIGH or LOW);

Met deze functie wordt een uitgang +5V of 0V.

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

digitalRead () functie

Deze functie kan alleen gebruikt worden met een port die vooraf met pinMode in INPUT mode is gezet.

Syntax

<waarde> = digitalRead (<pin#>);

Met deze functie krijgt <waarde> True = 1 (5V op pin) of False = 0 (0V op pin).

analogRead () functie

Deze functie kan alleen gebruikt worden op een analoge port. Deze wordt automatisch in de mode INPUT gezet door deze functie. Met deze functie wordt het “voltage” tussen 0 en 5V op een port gelezen. De hoogte wordt omgezet naar een getal tussen 0 en 1023

Syntax

<waarde> = analogRead (<oin#>);

Met deze functie krijgt <waarde> de hoogte van het voltage in een nummer tussen 0 en 1023.

analogWrite () functie

Deze functie kan alleen gebruikt worden op een port die in de mode OUTPUT is gezet. Met deze functie wordt het “voltage” tussen 0 en 5V op een port gezet. Een getal tussen de 0 en 1023 wordt omgezet naar een “voltage” tussen 0 en 5V.

Syntax

analogRead (<oin#>, <waarde>);

Met deze functie krijgt de pin een voltage tussen 0 en 5V afhankelijk van de hoogte van de <waarde> die kan liggen tussen 0, geeft 0V, en 1023, geeft 5V.

analogReference () functie

Deze functie wordt gebruikt in relatie met de AREF pin. Op AREF pin kan een spanning gezet worden tussen de 0 en 5V als referentie.
Deze functie heeft de volgende instel mogelijkheden als referentie.

- **DEFAULT** – de default 5V (on 5V Arduino boards) of 3.3 volts (on 3.3V Arduino boards).
- **INTERNAL** – De built-in reference, die gelijk is aan 1.1 volts op de ATmega168 of ATmega328 en 2.56 volt op de ATmega8 (niet beschikbaar op de Arduino Mega)
- **INTERNAL1V1** – de built-in 1.1V reference (Arduino Mega only)
- **INTERNAL2V56** – de built-in 2.56V reference (Arduino Mega only)
- **EXTERNAL** – Het voltage dat op de AREF pin (0 to 5V only) is gezet als referentie

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Syntax
analogReference (type);

type is: DEFAULT, INTERNAL, INTERNAL1V1 (Arduino Mega), INTERNAL2V56 (Arduino Mega) of EXTERNAL.

Character functies

Met de functies kun je controleren hoe de inhoud van een string is. Hoofdletters, kleine letters, etc.

- **int isdigit(int c)**
  Geeft 1 terug als c een cijfer is en 0 anders.
- **int isalpha(int c)**
  Geeft 1 terug als c een letter is en 0 anders.
- **int isalnum(int c)**
  Geeft 1 terug als c een cijfer of letter is en verder 0.
- **int isxdigit(int c)**
  Geeft 1 terug als c een hexadecimaal cijferkarakter is en 0 anders.
- **int islower(int c)**
  Geeft 1 terug als c een kleine letter is en 0 anders.
- **int isupper(int c)**
  Geeft 1 terug als c een hoofdletter is; Verder 0.
- **int isspace(int c)**
  Geeft 1 terug als c een witspatie karakternieuwregel is ('\n'), spatie (' '), vormvoer ('\f'), wagenretour ('\r'), horizontaal tab ('\t') of verticale tab ('\v') en 0 anders.
- **int iscntrl(int c)**
  Geeft 1 terug als c een controleteken is, zoals een nieuwe regel ('\n'), formulierfeed ('\f'), carriage return ('\r'), horizontaal tabblad ('\t'), verticale tab ('\v'), alert ('\a'), of backspace ('\b') en 0 anders.
- **int ispunct(int c)**
  Geeft 1 terug als c een printkarakter is anders dan een spatie, een cijfer of een letter en 0 anders.
- **int isprint(int c)**
  Geeft 1 terug als c een printkarakter is inclusief spatie (' ') en 0 anders.
- **int isgraph(int c)**
  Geeft 1 terug als c een printkarakter is anders dan spatie (' ') en 0 anders.

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Wat kun je met een Arduino

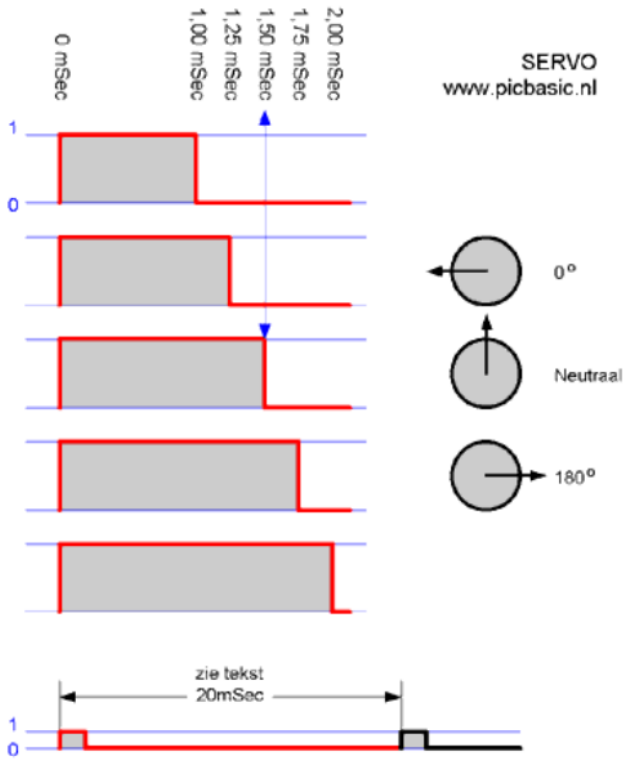
We beperken ons tot de modelspoor wereld.
Op deze site staan leuke leerprogramma's: https://arduino-lessen.nl/

Led verlichting

Door een port in output mode te zetten kan er 5V of 0V op gezet worden. Wordt daar een led op aangesloten dan deze aan of uit gezet worden.

Servo aansturing

Nemen een port die de eigenschap heeft dat deze een PWM signaal kan afgeven kan daarmee een servo bestuurd worden.
Een PWM (Pulse Width Modulation) is een blokgolf van 0 of 5V. Hoe langer de 5V hoe meer energie. De energie van de blokgolf kan vergroot worden met een driver zoals de L298. Hier zijn ook speciale shields voor beschikbaar die tot 16 servo's kunnen aansturen PCA9685. Deze worden aangestuurd via I2C.
Een servo heeft een pulse van 1ms tot 2ms om volledig van 0 naar 180 graden te draaien. Met de Arduino library <Servo.h> kan het met een getal van 0 tot 180.



Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Motor aansturing

Er verschillende motoren. Stappen motor zet elke keer een klein stapje. Een DC motor draait met een bepaalde snelheid.

Stappen motor

Hiet kan de library <Stepper.h> gebruikt worden. Met de library kan eenvoudig de instructie gegeven worden om de motor een stap te laten zetten. We gebruiken hiervoor de A4988 stepper driver.

DC motor

Met de PMW kunnen we een DC motor zeer nauwkeurig aansturen. De locomotieven worden door DCC ook middels een blokgolf van stroom voorzien.

Detectie met reactie

Een port kan in de mode INPUT gezet worden. Komt daar een hoog of laag op dan kan een interrupt gegenereerd worden. Daar kunnen vervolgens acties aangekoppeld wordne.

I2C

Dit is een bus waarmee devices aangestuurd kunnen worden. Elk device heeft een adres. Zo kunnen via een bus meerdere devices aangestuurd worden.

TX en RX (UART)

Dis is een serial bus waarmee ook devices aangestuurd kunnen worden.

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Basis programma's

In dit deel gaan enkele basis programma's programmeren. Hiervoor moeten we wel ook op een breadboard (experimenteel board) een schakeling bouwen.

Om programma's en de te gebruiken worden makkelijker leesbaar te houden, er mogen namelijk geen spaties in worden voorkomen, wordt vaak de volgende schrijfwijze aangehouden:

- parameters met hoofdletters en worden gescheiden door een "_" underscore.
  Voorbeeld: #define LED_PIN 5
- Functie namen beginnen met een kleine letter gevolgd door een hoofdletter voor het volgende woord. Alles aan elkaar geschreven.
  Voorbeeld: void identifyMyself ();
- Lirbary naam wordt altijd met een hoofdletter geschreven en woorden worden verbonden met een "_".

Buildin led

Elke Arduino heeft een BuildIn Led. Met deze led kun je aangeven of je programma draait.
Leerdoel: eerste programma maken.
Opdracht: BuildIn Led aanzetten.

Te gebruiken instructies:

- pinMode
- digitalWrite

Te gebruiken compiler variabelen*:

- LED_BUILDIN
- OUTPUT
- HIGH

* compiler variabelen zijn variabelen die standaard bekend zijn. Zoals ook:

- true
- false

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

LES 2: Opdrachten

Opdracht 1: Led aansturen

Leerdoel: Port gebruik.
Opdracht: een Led laten branden. Led laten knipperen.

- het initialiseren van een port.
- Een port hoog maken zodat de led gaat branden.
- De port hoog en laag maken zodat deze gaat knipperen.

Te gebruiken:

- #define

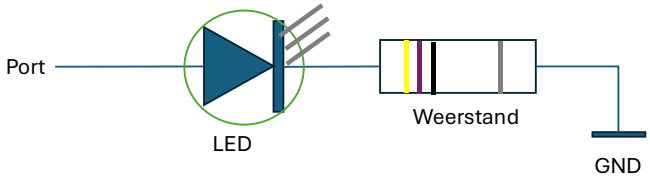
Te gebruiken instructies:

- pinMode
- digitalWrite
- delay

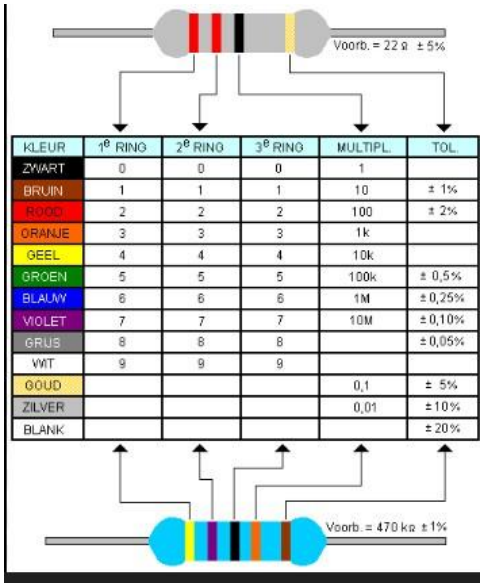
Te gebruiken compiler variabelen*:

- OUTPUT
- HIGH
- LOW

Voor de opdracht moet we een Led aan een port naar keuze koppelen.



De port kan maximaal 10mA leveren rekening houdend met alle ports. Een port is max 20mA.
Een Led kan maximaal 10mA stroom verwerken.
Hoe groot moet de weerstand zijn?
Formule: V = I X R



KLEUR	1e RING	2e RING	3e RING	MULTIPL.	TOL.
ZWART	0	0	0	1	
BRUIN	1	1	1	10	± 1%
ROOD	2	2	2	100	± 2%
ORANJE	3	3	3	1k	
GEEL	4	4	4	10k	
GROEN	5	5	5	100k	± 0,5%
BLAUW	6	6	6	1M	± 0,25%
VIOLET	7	7	7	10M	± 0,10%
GRIJS	8	8	8		± 0,05%
WIT	9	9	9		
GOUD				0,1	± 5%
ZILVER				0,01	± 10%
BLANK					± 20%

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Opdracht 2: Andreas kruis

Leerdoel: Led aansturing met beslissing.
Opdracht : het laten knipperen van twee leds tegen elkaar in.

- het initialiseren van porten.
- Een porten afwisselend hoog en laag maken zodat de leds gaat knipperen.
- De port hoog en laag maken zodat deze gaat knipperen.

Te gebruiken:

- #define
- variabele

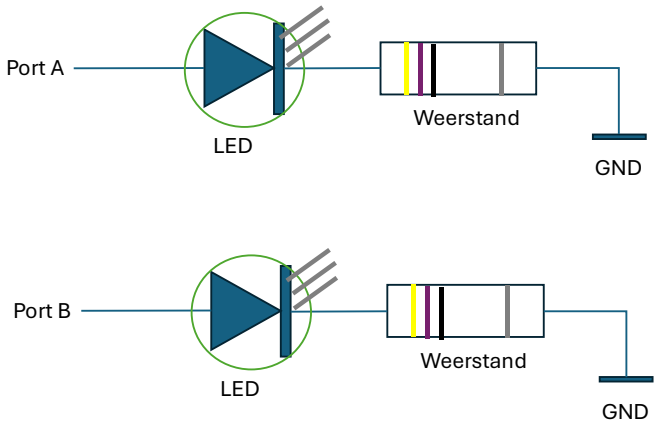
Te gebruiken instructies:

- pinMode
- digitalWrite
- delay
- if else statement

Te gebruiken compiler variabelen*:

- OUTPUT
- HIGH
- LOW

Voor de opdracht moet we twee Leds aan een port naar keuze koppelen.



Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Opdracht 3: Servo aansturen

Leerdoel: het aansturen van een servo.
Opdracht: deze laten draaien tussen 0 en 180 en weer terug.

Te gebruiken:

- #include
- #define
- Variabele
- initialisatie

Te gebruiken instructies:

- servo
- delay
- for statement

Te gebruiken compiler variabelen*: geen

PWM ports

Board	PWM Pins *
UNO (R3 and earlier), Nano, Mini	3, 5, 6, 9, 10, 11
UNO R4 (Minima, WiFi)	3, 5, 6, 9, 10, 11
Mega	2 - 13, 44 - 46
GIGA R1	2 - 13
Leonardo, Micro, Yún	3, 5, 6, 9, 10, 11, 13
UNO WiFi Rev2, Nano Every	3, 5, 6, 9, 10
MKR boards**	0 - 8, 10, A3, A4
MKR1000 WiFi**	0 - 8, 10, 11, A3, A4
Zero**	3 - 13, A0, A1
Nano 33 IoT**	2, 3, 5, 6, 9 - 12, A2, A3, A5
Nano 33 BLE/BLE Sense	1 - 13, A0 - A7
Due***	2-13

Voor deze opdracht moet de volgende schakeling gemaakt worden.

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Opdracht 4: Stappen motor aansturen

Leerdoel: het aansturen van een stappenmotor.
Opdracht: deze een aantal stappen te laten draaien

Te gebruiken:

- #include
- #define
- Variabelen
- constante
- initialisatie

Te gebruiken instructies:

- step
- delay
- for statement

Te gebruiken compiler variabelen*:

Voor deze opdracht moet de volgende schakeling gemaakt worden.

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Opdracht 5: UART communicatie tussen twee Arduino's

Opdracht: de BuildIn led aanzetten via een andere Arduino
Leerdoel: communicatie opzetten via de UART.

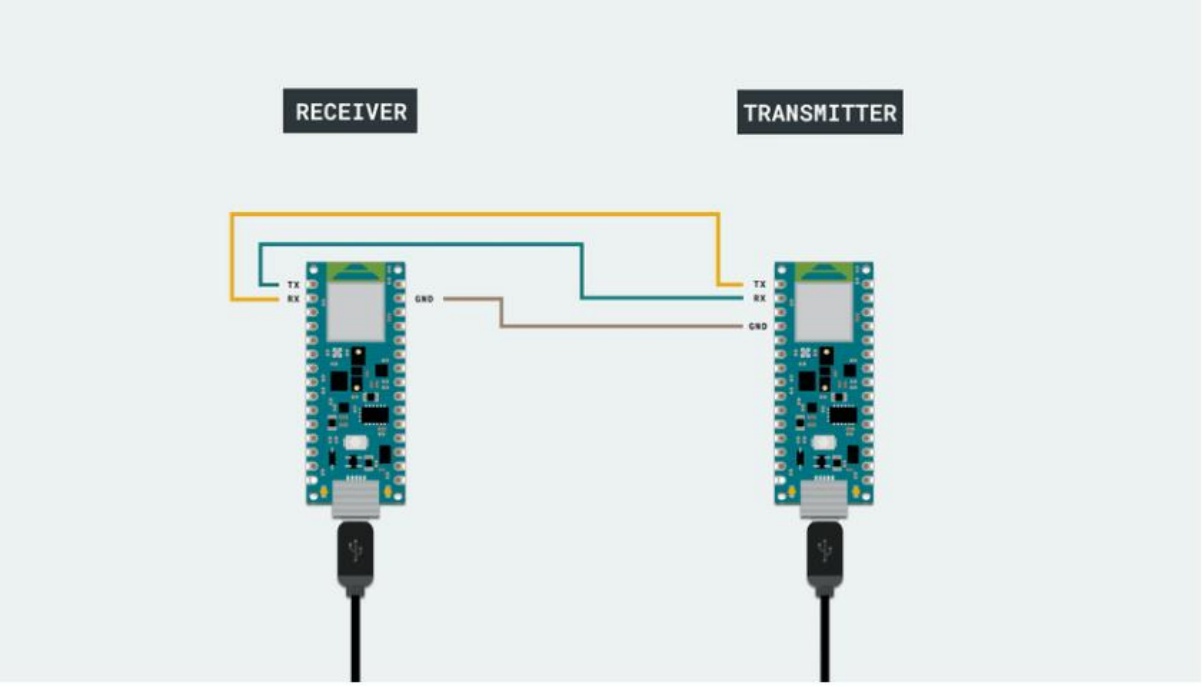
Te gebruiken:

- #include
- #define
- Variabelen
- constante
- initialisatie

Te gebruiken instructies:

-
- delay
- if else statement

Hiervoor worden twee Arduino's met elkaar verbonden via de TX en RX ports. De GND met verbonden worden om te zorgen dat beide Arduino's op het zelfde spanningsniveau staan. Zie figuur.



Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

We zetten de ontvangende Arduino in de juiste mode en de BuildIn Led uit en de UART communicatie snelheid.

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);      // set LED pin as output
  digitalWrite(LED_BUILTIN, LOW);    // switch off LED pin

  Serial1.begin(9600);           // initialize UART with baud rate of 9600
}
```

In de main loop gaan we wachten op een byte van de zender.

```
void loop() {
  while (Serial1.available() >= 0) {
    char receivedData = Serial1.read();   // read one byte from serial buffer and save to receivedData
```

Als we een byte hebben ontvangen gaan we kijken wat het is, een 1 of een 0, en afhankelijk daarvan wordt de BuildIn let aan of uit gezet.

```
if (receivedData == '1') {
     digitalWrite(LED_BUILTIN, HIGH); // switch LED On
    }
    else if (receivedData == '0') {
     digitalWrite(LED_BUILTIN, LOW);  // switch LED Off
    }
  }
}
```

Vervolgens zetten de verzendende Arduino in de juiste mode.

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);      // set LED pin as output
  digitalWrite(LED_BUILTIN, LOW);    // switch off LED pin

  Serial.begin(9600);             // initialize serial communication at 9600 bits per second:
  Serial1.begin(9600);           // initialize UART with baud rate of 9600
}
```

We zetten ook de Serial aan omdat we iets naar de monitor willen sturen.

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Vervolgens gaan we een 1 of een 0 versturen.

```
if (Serial.read() == '1'){
  Serial1.println('1');
  digitalWrite(LED_BUILTIN, HIGH);
  Serial.println("LEDS ON");
}
else if (Serial.read() == '0'){
  Serial1.println('0');
  digitalWrite(LED_BUILTIN, LOW);
  Serial.print("LEDS OFF");
}
```

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Opdracht 6: I2C communicatie tussen twee Arduino's

Opdracht: stuur een character van de ene naar de andere Arduino over I2C.
Leerdoel: communicatie opzetten via de I2C.

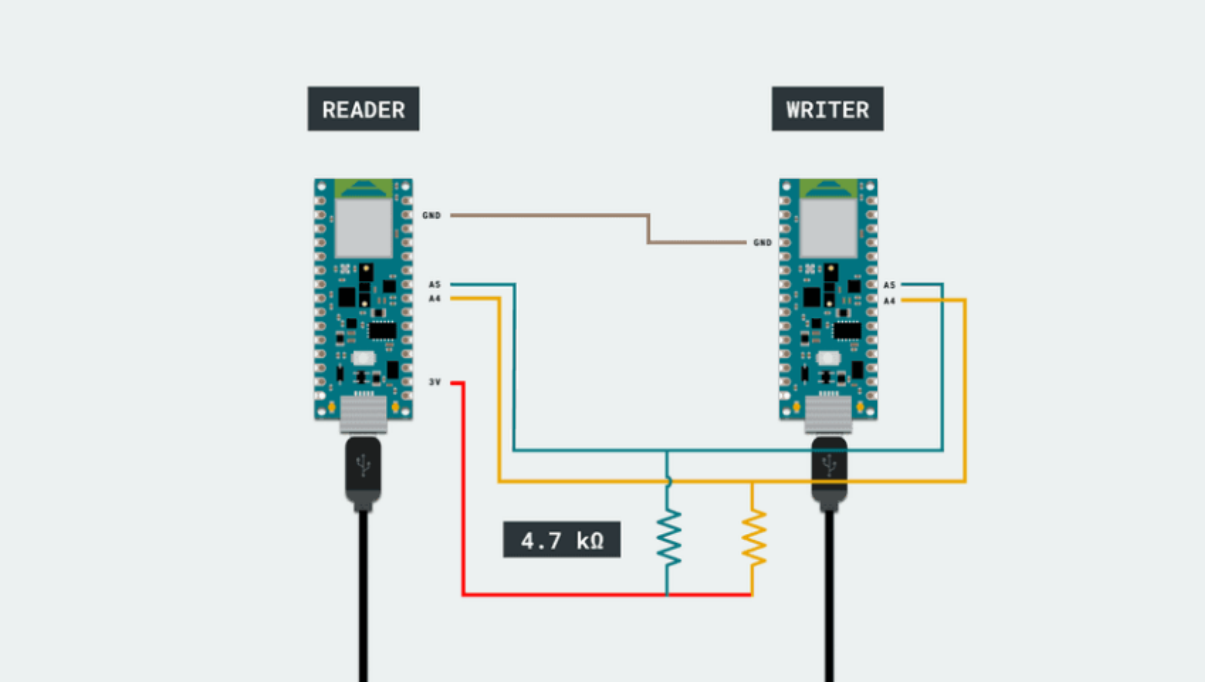
Te gebruiken:

- #include
- #define
- Variabelen
- constante
- initialisatie

Te gebruiken instructies:

-
- delay
- if else statement

Hiervoor worden twee Arduino's met elkaar verbonden via de SDA(A4) en SCL(A5) ports. De GND met verbonden worden om te zorgen dat beide Arduino's op het zelfde spanningsniveau staan. Zie figuur.



Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

LES 3 : Input / Output en Simulatie

Input / Output

- Digitaal (1 of 0) wat kan ik daarmee
- Analoog (potmeter) wat kan ik daarmee

Simulatie tool

De schakeling van de opdrachten bouwen in het simulatie tool.
Leerdoel: aanpassingen en uitbreidingen kunnen ontwerpen en bouwen.

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

Library maken

Hier is een voorbeeld van hoe een library gemaakt wordt.

Een library bestaat uit twee bestanden. Een .h en een .cpp bestand.

LedControl

Voorbeeld van een LedControl library

De LedControl.h bestand bevat:

```
#ifndef LEDCONTROL_H
#define LEDCONTROL_H

#include <Arduino.h>

class LedControl {
  public:
    LedControl(int pin);   // constructor
    void aan();
    void uit();
    void knipper(int delayTijd);

  private:
    int _pin;
};

#endif
```

De bijhorende LedControl.cpp bestand bevat:

```
#include "LedControl.h"

LedControl::LedControl(int pin) {
  _pin = pin;
  pinMode(_pin, OUTPUT);
}

void LedControl::aan() {
  digitalWrite(_pin, HIGH);
}
```

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

```
void LedControl::uit() {
  digitalWrite(_pin, LOW);
}

void LedControl::knipper(int delayTijd) {
  aan();
  delay(delayTijd);
  uit();
  delay(delayTijd);
}
```

Voorbeeld hoe deze gebruikt wordt in een Arduino programma.

```
#include "LedControl.h"

LedControl led(13);  // LED op pin 13

void setup() {
}

void loop() {
  led.knipper(500); // knipper elke 500 ms
}
```

Timer1

```
#ifndef TIMERONE_H
#define TIMERONE_H

#include <Arduino.h>

class TimerOne {
  public:
    void start(long microseconds, void (*callback)());
    void stop();

  private:
    static void (*isrCallback)();
};

#endif
```

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

```
#include "TimerOne.h"

void (*TimerOne::isrCallback)() = nullptr;

void TimerOne::start(long microseconds, void (*callback)()) {
  isrCallback = callback;

  noInterrupts();

  TCCR1A = 0;
  TCCR1B = 0;
  TCNT1 = 0;

  // Bereken compare waarde
  long compare = (16 * microseconds) / 1024 - 1;
  OCR1A = compare;

  // CTC mode
  TCCR1B |= (1 << WGM12);

  // prescaler 1024
  TCCR1B |= (1 << CS12) | (1 << CS10);

  // interrupt enable
  TIMSK1 |= (1 << OCIE1A);

  interrupts();
}

void TimerOne::stop() {
  TIMSK1 &= ~(1 << OCIE1A); // disable interrupt
}

// ISR (globaal!)
ISR(TIMER1_COMPA_vect) {
  if (TimerOne::isrCallback != nullptr) {
    TimerOne::isrCallback();
  }
}
```

Pastoor van Leeuwenstraat 23
5701 JS Helmond
Het Baken

```
#include "TimerOne.h"

TimerOne timer;

void toggleLed() {
  digitalWrite(13, !digitalRead(13));
}

void setup() {
  pinMode(13, OUTPUT);

  timer.start(1000000, toggleLed); // 1 seconde
}

void loop() {
  // niks nodig
}
```