Verslag 03 12 mrt 2019

Les -3- is meteen een stuk serieuzer, er wordt gesproken over typen variabelen
De meest gebruikte data typen zijn Bolean, Integer, Character, String, en Array.

Boolean kent twee waarden: true en false ofwel: waar en niet waar.
False wordt aangegeven met **0** en true krijgt de waarde **1**.
Dit is slechts één bit, maar neemt toch een hele byte in beslag.

Integer (afgekort met int) neemt twee bytes in beslag.
Dit is een getal zonder cijfers achter de komma.

Character (afgekort char) is één letter of cijfer, maar niet de getalswaarde, maar het teken.
De ASCII tabel (zie helemaal onderaan dit verhaal, flink inzoomen!)

String, een rij karakters achter elkaar. Meerdere karakters vormen een string.
Er zijn twee typen strings: **s**tring is een karakter array (recht toe recht aan stuk tekst)
String is een object (meer mogelijkheden)
Array een matrix van getallen of karakters. Een array kan meerdere dimensies hebben.
Ja, wij kennen er slechts 3, maar een array kan uit meer dan 3 dimensies bestaan.

Boolean	1 byte
char	1 byte getallen van -127 tot + 127 ASCII table.
byte	1 byte getallen van 0 tot 255
int	2 bytes getallen van -32.768 tot +32.767
unsigned int	2 bytes getallen van 0 tot 65.535
word	2 bytes gelijk aan unsigned int dit is het zelfde.
long	4 bytes getallen van -2.147.483.648 tot +2.147.283.647
unsigned long	4 bytes getallen van 0 tot 4.294.967.295
float	4 bytes getallen van -3,402.823.5 E38 tot 3,402.823.5 E 38. Gebruik floats zo min mogelijk, de Arduino kan er moeilijk mee omgaan en gebruikt veel code om met floats te rekenen. Dit vertraagd enorm!
string - char	een reeks karakters .
array	een reeks variabelen, karakters.

Variabelen zijn locaties waar de data te vinden is, naast 'wat' is ook het 'waar' belangrijk.
Een variabele is dus de naam van de data (van welk type deze dan ook is)
Namen van variabelen:

- **Beginnen ALTIJD met een letter**(a, b, …, z, A, B, … , Z) of een **underscore** (_)
- mag **letters** bevatten
- mag **underscore**(s) bevatten
- mag cijf**ers** bevatten
- **MAG NOOIT speciale tekens, symbolen of spaties bevatten**

Er zijn twee typen variabelen constanten: **const** en **#define**
const int naamvar = 5; voor onveranderlijke variabelen. Reserveert geheugen naamvar met 5
#define int naamvar 5 geen **;** de compiler vertaalt meteen naamvar in 5.

Een ander verschil is de **scoop**.(het gebied waarbinnen de variabele geldig is)
Binnen de functie is **lokale** variabele.
Helemaal aan het begin declareren, **globale** variabele.
Je kan in meerdere locaties lokale variabelen gebruiken die dezelfde naam hebben.

Dus:

```
int A = 0;                  ← dit is een globale variabele A

void setup() {
  A = 1;                    ← dit is een lokale variabele A
                              Uitsluitend geldig in de setup
                              Mag dezelfde naam als de globale variabele hebben
                              Maakt het wel verwarrend
  Serial.begin(115200);
  Serial.print("Setup: A = ");
  Serial.println(A);
}
void loop() {
  int A;                    ← dit is een lokale variabele A
                              Uitsluitend geldig in de loop
                              Mag dezelfde naam als de globale variabele hebben, maar je mag zelfs ook
                              dezelfde naam gebruiken in andere lokale gebieden, verwarrend.
  A = 2;
  Serial.print("Loop: A = ");
  Serial.println(A);
  delay(1000);
}
```

Wanneer je de lokale variabele met (//) uit zet, dan wordt de globale variabele gekozen.
Een lokale variabele heeft dus voorrang op de globale variabele.

Getallen

In de schets kan je getallen aangeven. Normaal zijn het decimale getallen
Een binair getal geef je aan door vóór het getal **0b** (nul-b) te typen.
Serial.println(0b1010); werkt hetzelfde als **Serial**.println(0b1010,DEC);
,DEC (Print de waarde decimaal), is niet noodzakelijk, wil je binair printen, gebruik dan ,BIN
Voor hexadecimale weergave gebruik ,HEX
Wil je een hexadecimaal getal invoeren dan zet je er **0x** voor.

Voorbeeldje:

```
void setup(){                                  // er moet ALTIJD een void setup() aanwezig zijn!
  Serial.begin(115200);                        // communicatie snelheid met de PC
  int decNum = 10;                             // dec = 10;   hex = A      bin = 1010
  int hexNum = 0x2FA;                          // dec = 762;  hex = 2FA;  bin = 1011111010
  int binNum = 0b10001010;                     // dec = 138;  hex = 8A;   bin = 10001010

  Serial.println ("alle nummers decimaal");    // alle nummers decimaal
  Serial.println (decNum);                     // 10
  Serial.println (decNum,DEC);                 // 10
  Serial.println (hexNum);                     // 762
  Serial.println (hexNum,DEC);                 // 762
  Serial.println (binNum);                     // 138
  Serial.println (binNum,DEC);                 // 138

  Serial.println("alle nummers hexadecimaal"); // alle nummers hexadecimaal
  Serial.println (decNum,HEX);                 // A
  Serial.println (hexNum,HEX);                 // 2FA
  Serial.println (binNum,HEX);                 // 8A

  Serial.println("alle nummers binair");       // alle nummers binair
  Serial.println (decNum,BIN);                 // 1010
  Serial.println (hexNum,BIN);                 // 1011111010
  Serial.println (binNum,BIN);                 // 10001010
}
void loop(){                                   // er moet ALTIJD een void loop() aanwezig zijn!
}
```

Getallenstelsels:
Uiteraard werken wij in het decimale stelsel, de cijfertjes: 0-1-2-3-4-5-6-7-8-9.
Wanneer we méér dan 9 willen aangeven, zijn er geen cijfers meer, dan zetten we er een -1- voor en beginnen gewoon opnieuw met -0-. Na 9 komt dus 10 , maar dat wist je al! (hoop ik.)
In het getal 123 heeft elk cijfer een waarde, de -1- vertegenwoordigd het aantal maal 100
De -2- vertegenwoordigd het aantal malen 10 en de 3 geeft het aantal eenheden aan.
Dus de **positie** van het cijfer bepaald het gewicht van dit cijfer. Zo ook in het binaire en hexadecimale stelsel. In het hexadecimaal tellen we na 9 verder met: A; B; C; D; E; F

Decimaal, grondtal 10: pos 1 = 10^0 ; pos 2 = 10^1 ; pos 3 = 10^2 ; pos 4 = 10^3 ; enz.
Binair, grondtal 2: pos 1 = 2^0; pos 2 = 2^1; pos 3 = 2^2; pos 4 = 2^3; enz.
Hexadecimaal, grondtal 16: pos 1 = 16^0; pos 2 = 16^1; pos 3 = 16^2; pos 4 = 16^3; enz.

Om terug te komen op het getal 123 dat is: $3x10^0 + 2x10^1 + 1x10^2 = 3x1+2x20+1x100 = 123$
Binair: $11010 = 0x2^0 + 1x2^1 + 0x2^2 + 1x2^3 + 1x2^4 = 0 + 2 + 0 + 8 + 16 = 26$.
Hexadecimaal: $2EA6 = 6x16^0+Ax16^1+Ex16^2+2x16^3 = 6x1+10x16+14x256+2x4096 = 11942$
Je ziet, met hexadecimaal heb je het minste aantal karakterposities nodig voor een groot getal.

Operators:
Een operator doet iets met de data. Totaal overzicht: https://www.arduino.cc/reference/en/
Er zijn **6** hoofdgroepen operators:

Rekenkundige operators:	- aftrekken	+ optellen
	/ delen	* vermenigvuldigen
	= toewijzen	% modulo

Modulo is de 'rest' van een deling: 12 % 3 = 0 14 % 3 = 2
A+3=A dit is rekenkundig fout, het betekent: A+3 en zet deze waarde terug in variabele A.
Hiermee is variabele A met 3 verhoogd, de oude waarde van A is hiermee verloren gegaan.

Vergelijking operators:	== is gelijk aan	**!**= is NIET gelijk aan
	< is kleiner dan	> is groter dan
	<= is kleiner of gelijk dan	>= is groter of gelijk dan

Het antwoord van deze bewerking is altijd true of false; 1 of 0; waar of niet-waar

Boolean operators:	&& and (en)	|| or (of)	! not (niet)
Waarheidstabellen:	1 && 1 = 1	1 || 1 = 1	! 1 = 0
	1 && 0 = 0	1 || 0 = 1	! 0 = 1
	0 && 1 = 0	0 || 1 = 1	
	0 && 0 = 0	0 || 0 = 0	

Compound operators, dat zijn gecombineerde operators

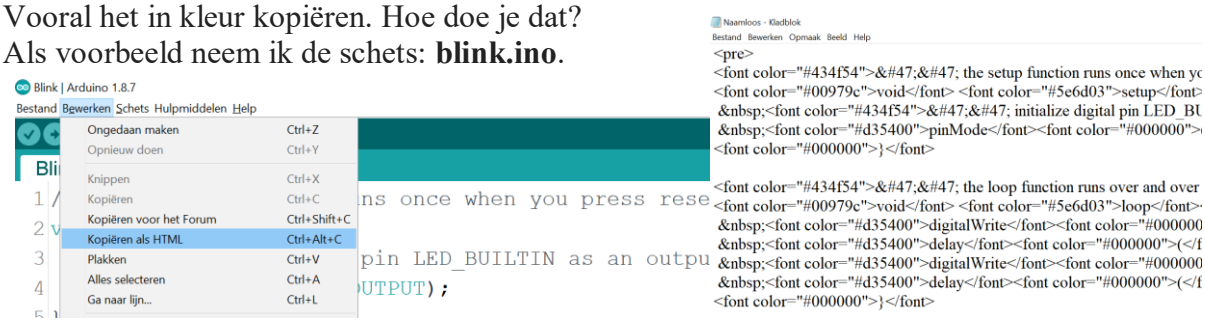
++	A++	hetzelfde als A = A+1	--	A--	hetzelfde als A = A-1
+=	A += B	hetzelfde als A = A+B	-=	A -= B	hetzelfde als A = A-B
*=	A *= B	hetzelfde als A = A*B	/=	A /= B	hetzelfde als A = A/B
%=	A % B	hetzelfde als A % B			

6 groepen operators? Jawel, er zijn ook nog **pointer acces** en **bitwise** operators.
Die laten we even voor wat het is, daarvoor is meer ervaring nodig. Komt later wel!

Later vond ik op het internet wat 'bitwise' operators zijn, lijkt me toch wel handig!

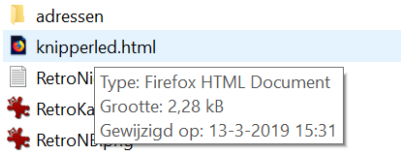
AND	&	in tegenstelling tot &&, dit beslaat een hele byte
OR	|	in tegenstelling tot || ook hier de OR voor een byte.
X-OR (eXclusive or)	^	Ook hier gaat de behandeling over een hele byte.
Left shift	<<	de byte schuift in zijn geheel 1 pos naar links.
Right shift	>>	de byte schuift in zijn geheel 1 pos naar rechts.
NOT	~	de complements waarde, 1 wordt 0 en 0 wordt 1.

Er waren wat moeilijkheden met het kopiëren van schetsen van tekst naar IDE en terug.
Vooral het in kleur kopiëren. Hoe doe je dat?
Als voorbeeld neem ik de schets: **blink.ino**.

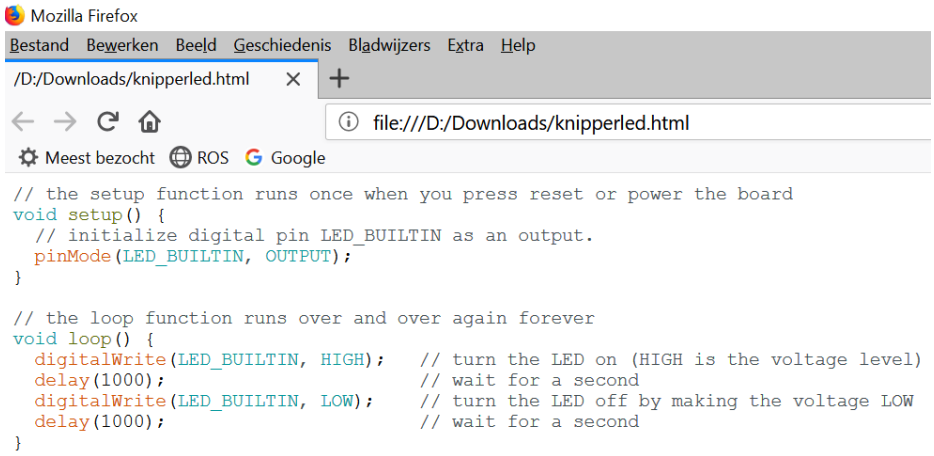


Klik in 'bewerken' op 'kopiëren als HTML' je ziet weinig veranderen.

Open kladblok (notebook) en plak het zojuist gekopieerde bestand.
Sla dit op, verander .TXT in .HTML dit is belangrijk!!



Open de browser, en laad het kladblokbestand, dat je zojuist de extensie .html hebt gegeven.
Je kan in verkenner ook op dit bestand klikken, dan opent de browser vanzelf. Immers, elk HTML bestand is aan de browser gekoppeld.



Hier zie je de schets in de juiste kleuren zoals in de Arduino IDE.
Dit is inderdaad gekleurde tekst, je kan dit kopiëren en in je tekstverwerker verder bewerken.

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage
level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LO
  delay(1000);                       // wait for a second
}
```

Mijn tekstverwerker geeft het weer als Courier New in fontgrootte 10, je kan dat naar eigen smaak aanpassen, ik heb er times new roman font grootte 12 van gemaakt.

Groeten, Dré

De ASCII tabel op ander bestand, Flink inzoomen!