PROGRAMMEREN IN C#

les 1

Deel 1

1) Anatomie van een programma
2) Structuur
3) Syntax
4) Variabelen
5) Lineair programmaverloop

Deel 2

1) functies
2) arduino tijdsfuncties
3) arduino monitor
4) arduino I/O
5) arduino random functies

Anatomie van een arduino-programma

```
int pin=13;
void setup() {
pinMode(pin, OUTPUT);
}
void loop() {
digitalWrite(pin, HIGH);
delay(1000);
digitalWrite(pin, LOW);
delay(1000);
}
```

- structuur
- syntax
- variabelen
- operatoren
- functies

structuur

C# kent (**verplicht**) twee delen:

```
// knipperled

void setup() {
   INT Led = 13;
   pinMode(Led, OUTPUT);
}

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

Initialisatielus: eenmalig

Hoofdlus (voortdurend)

SYNTAX

C# kent de volgende regels:

- Elke opdrachtregel eindigt op **;**
- Elk deel van een programma (BLOK) staat tussen accolades: **{ }**
- Een commentaarregel wordt voorafgegaan door **//**
- Een commentaarblok (=meerdere regels) begint met **/*** en eindigt op ***/**
- Commentaarregels worden genegeerd tijdens compileren
- Spaties en CR niet van belang

Datatype, variabele, functie

```
// knipperled

void setup() {
   INT Led = 13;
   pinMode(Led, OUTPUT);
}

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

Datatype; variabele

Functie, argument

SETUP() en LOOP() zijn functies van het type VOID (geen waarde ove

Voorbeeld: functie met teruggave

```
Int potVal()  /functie levert waarde tussen 0 en 1023 afhankelijk van potmeter
{
int v;
v = analogRead(pot);
return v;
}
```

Let op de betekenis van ‘=’ ! Lees dit als ‘wordt’ !

variabelen

– zijn symbolische namen om geheugenplaatsen met data aan te wijzen

– van elke variabele moet het **datatype** vooraf ‘gedeclareerd’ worden

– er zijn **globale** en **lokale** variabelen

– globale variabelen: te declareren vóór SET-UP

– lokale variabelen: alleen betekenis binnen een blok of functie

– caveat! Namen van variabelen zijn hoofdletter gevoelig! Aap ≠ aap!

– variabelennamen: alleen a-z, 0-9, underscore. Nummers niet als eerste teken!

datatypen

Overzicht:

void	Boolean	char	Unsigned char	byte	int	Unsigned int	word
long	Unsigned long	short	float	double	array	String-char array	String-object

VOID: alleen in functie-declaraties; geen informatieteruggave bij aanroep vd functie
BOOLEAN: kent slechts 2 waarden: true of false; kost 1 byte
CHAR variabele voor letters; (1 byte per letter)
UNSIGNED CHAR: voor nummers tussen 0 en 255; (1 byte)
BYTE: voor nummers tussen 0 en 255; (1 byte)
INT: voor gehele getallen tussen -32,768 en +32,767 (-2^31 en+2^31-1); (2 bytes)
UNSIGNED INT: voor gehele getallen tussen 0 en 65,535; (2 bytes)
WORD: voor 16 bit natuurlijk getal

datatypen(vervolg)

void	Boolean	char	Unsigned char	byte	int	Unsigned int	word
long	Unsigned long	short	float	double	array	String-char array	String-object

LONG: gehele getallen tussen -2.147.483.648 en +2.147.483.647 (4 bytes)
UNSIGNED LONG:
SHORT: 16 bit geheel getal (2 bytes)
FLOAT: reele getallen tussen -3.402823.10^38 en -3.402823.10^38 (4 bytes)
DOUBLE: bij arduino duo en nano hetzelfde als FLOAT
ARRAY:
STRINGCHAR ARRAY:
STRINGOBJECT:

constanten

- Een constante is een variabele waarvan de waarde niet verandert tijdens de uitvoering
- Te declareren met CONST

  vb: const int rood = 5; const floid pi = 3.1416
- Voorgedefinieerde constanten:

  (true, false), (HIGH,LOW), (INPUT,OUTPUT)

Samenvatting variabelen

Table 3-1. Arduino Data Types

Name	Size	Range
boolean	1 bit	true or false
byte	8 bit	0 to 255
char	8 bit	-128 to 127
int	16 bit	-32,768 to 32,767
long	32 bit	-2,147,483,648 to 2,147,483,647
float	32 bit	-3.4028235E+38 to 3.4028235E+38

Operatoren

1. Toewijzingsoperator =

Int mijnGetal = 255 (lees: mijnGetal *wordt* 255)

Voorbeelden:

```
mijnGetal = 4 + 38;

mijnGetal = mijnGetal – 5

float pi = 3.14159;
int straal = 5;
float omtrek;
omtrek = 2 * pi * straal;

int mijnGetal;
mijnGetal = 9/5;          quid?
```

2. Rekenkundige operatoren

(A=10 B=6)

operator	naam	voorbeeld	resultaat
+	som	A + B	16
-	verschil	A – B	4
*	vermenigvul-diging	A * B	60
/	deling	A / B	1.666...
%	modulo	A % B	0,666...

Gemengde rekenkundige operatoren

(int A = 9)

operator	betekenis	voorbeeld
A++	A = A + 1	10
A--	A = A – 1	8
A+=10	A = A+10	19
A-=7	A= A – 7	2
A*=1.5	A=A *1.5	13 (!)
A/=4	A=A/4	2 (!)

Gemengde rekenkundige operatoren

A++	A=A+1
A--	A=A-1
B+=A	B=B+A
B-=A	B=B-A
B*=A	B=B*A
B\=A	B=B\A
B%=A	B=B%A

3a. vergelijkende operatoren

Operator	Beschrijving
x == y	Is gelijk aan
x != y	Is niet gelijk aan
x < y	Is kleiner dan
x > y	Is groter dan
x <= y	Is kleiner dan of gelijk aan
x >= y	Is groter dan of gelijk aan

If (getal1 == getal2) …..

If (getal1 >= getal2) …..

3b. Logische operatoren
(werken bitgewijze)

Bitgewijze operatoren werken op de opeenvolgende bits van twee gehele getallen
De bewerkingen: **NOT (~)**, **AND (&), OR (I), XOR (^)**

Voorbeeld: int A= 92 int B= 101 d.w.z 01011100 en 01100101
A & B → 01000100 (36 decimaal)
A I B → 01111101 (125 decimaal)
A ^ B → 00111000 (104 decimaal)

NOT (~) ~A → 10100011 (160 decimaal)

Bitshift **LEFT (<<)** A<<3 → 11100000 3 plaatsen opschuiven
RIGHT(>>) A>>2 → 00010111 2 plaatsen opschuiven
(afhankelijk van signed/unsigned int!)

Gemengde logische operatoren

A I = B	A = A I B
A & = B	A = A & B

3c. booleanse operatoren

(werken op expressies)

Operator	Beschrijving
&&	AND
||	OR
!	NOT

Booleaanse operatoren worden gebruikt om beslissingen te nemen.
Voorbeeld: (a=3 en b=4)

if (a < 5) && (b== 5) …..

Als *expressie 1*= true; en *expressie 2* = false:

1&&2 → false *1* II *2* → true !*1* → false

4. stringoperatoren

Operator	Beschrijving
“C” + “#”	De tekenreeksen worden aan elkaar geplakt.

LINEAIR PROGRAMMAVERLOOP

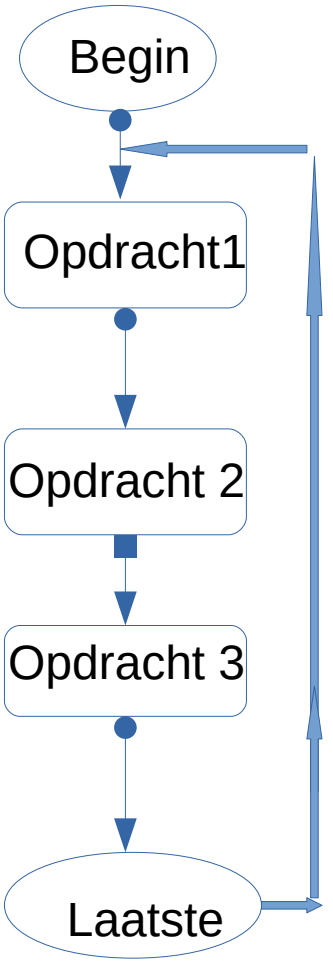
– Het setup-gedeelte wordt eenmaal uitgevoerd na reset

– Het loop-gedeelte wordt lineair uitgevoerd:
    eerste instructie
    tweede instructie
    derde instructie
    enz…

.

– bij de laatste opdracht wordt teruggesprongen naar de eerste

– **het is mogelijk om het programmaverloop te wijzigen door gebeurtenissen, voorwaarden, beperkingen, enz…**



functies

- Gebruikt om herhaling van blokken te vermijden
- Ook gebruikt bij specifieke hardware
- Worden gecreeerd buiten setup() en loop()
- Algemene vorm:

Returntype functienaam(argument1, argument2,..) {instructies}

Drie ‘soorten’ functies:

- ‘home-made’ functies;
- in Arduino-IDE opgenomen functies;
- in (externe) libraries opgenomen functies

Vandaag:

timing functies
monitor-functies
I/O-functies, digitaal en analoog
random-generator functie

Voorbeeld home-brewed functie:
functie die twee getallen (type int) optelt en de som teruggeeft:

boven loop()

```
int som_func(int x, int y);
    { int z=0;
      z = x + y;
      return z }

int som_func(int , int );
```

onder loop()

```
int som_funct(int , int );
void setup()
.....
void loop()

int som_func(int x, int y);
    { int z=0;
    z= x + y;
    return z}
```

Arduino timing functies

- millis()
- micros()

Zonder argument! Retourneert het aantal ms of us, verlopen sinds de start van het programma in unsigned long (max 50 dagen, resp. 70s).

- delay(arg)
- delayMicroseconds(arg)

Dient om een pauze-tijd in te lassen in een programma. Argument is unsigned int. Bij delayMicroseconds maximaal 15000. Pas op! Beide functies stoppen de verwerking, met Uitzondering van interrupthandling!

ARDUINO MONITOR

Drie stappen om de monitor in te schakelen:

1. Neem in je setup() op Serial.begin(arg)

   om de baudrate in te stellen – arg: 300, 600,1200,…,115200

   meestal 9600

2. Serial.print(arg) of Serial.println(arg)

   om iets te versturen (tekst of data) – zie voorbeelden

3. start monitor via ‘hulpmiddelen → seriele monitor’

voorbeelden

Serial.print(78)	78
Serial.print(3.1416)	1.23
Serial.print(“hallo”)	hallo
Serial.print(“\t”)	*tab*
Serial.print(78, BIN)	1001110
Serial.print(78, DEC)	78
Serial.print(78, HEX)	4E
Serial.print(3.1416, 0)	3
Serial.print(3.1416, 2)	3.14
Serial.println(78)	78 + CR
Serial.println(3,1416, 2)	3,14+CR

Arduino I/O

Digitale I/O:

- pinMode(pin, INPUT/OUTPUT/INPUT-PULLUP))
- digitalWrite(pin, LOW/HIGH)
- digitalRead(pin)

pinMode(arg1, arg 2)

- Om een pin te configureren als input of output
- Arg1 is het pinnummer
- Arg2 is INPUT of OUTPUT (of INPUT_PULLUP)
- Te gebruiken in de SETUP()

digitalRead(arg)

- dient om de toestand van een pin te lezen
- arg is het pinnummer
- retourneert 0 of 1
- analoge pinnen (A0, A1…) kunnen als digitaal gebruikt worden
- als de pin niet verbonden is, is het bresultaat willekeurig

digitalWrite(arg1,arg2)

- Stelt de toestand van een pin (0 V of 5V)
- Arg1 is het pinnummer
- Arg2 is HIGH of LOW
- Als de pin geconfigureerd is als input (via digitalWrite) wordt een interne weerstand ingeschakeld (HIGH) of uitgeschakeld

Analoge I/O:

- AnalogRead(pin)
- AnalogWrite(pin, value)
- AnalogReference(argument)

analogRead(arg)

- dient om een analoge pin (A0, A1…) te lezen
- arg = pinnummer
- 10 bit resolutie: retourneert 1024 waarden tussen 0V en 5 V
- vergt 100 µs (maximaal 10000 samples/s)
- als de pin niet verbonden is: willekeurige waarden!

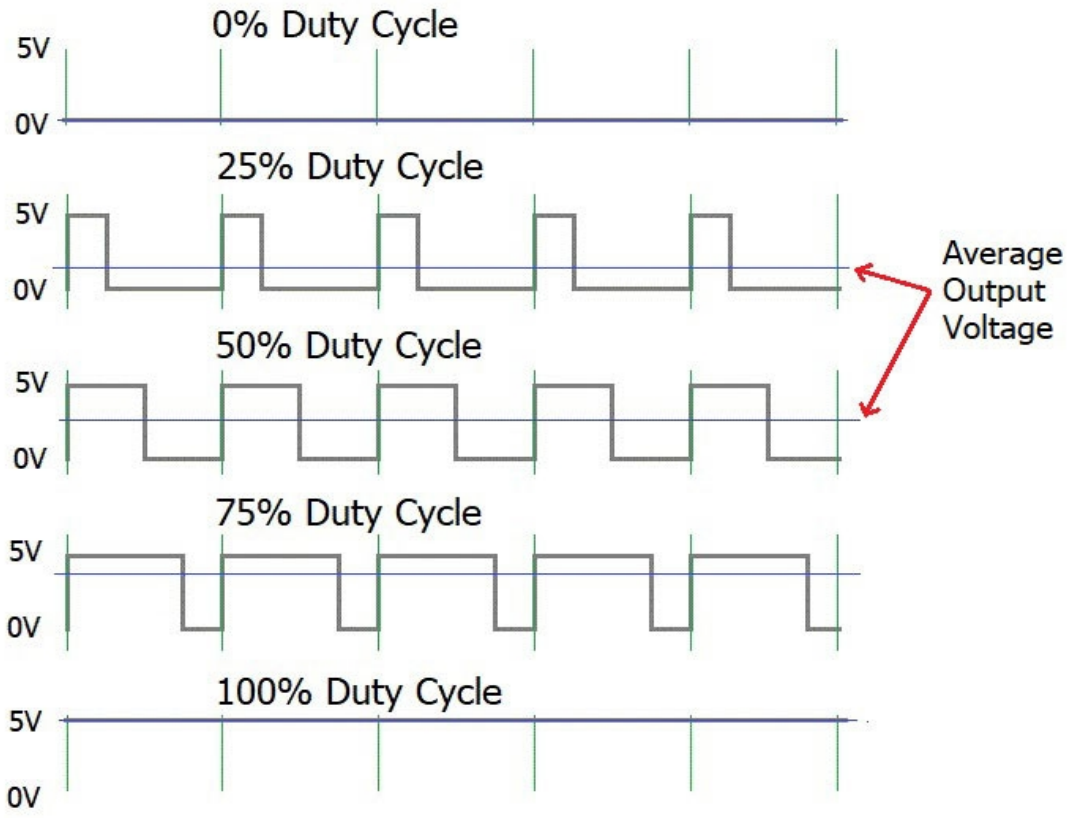
analogReference(arg)

- Legt de referentiespanning vast
- Default = 5 V
- Arg is DEFAULT, INTERNAL (1,1V*), EXTERNAL (Aref-pin)
- NOOIT meer dan 5 V op Aref!

analogWrite(arg1, arg2)

- arduino kent geen DAC. Er zijn alleen PMW-uitgangen
- arg1 = het pinnummer
- arg2 = een int tussen 0 en 255; bepaalt de duty-cycle van de PWM (0 = 0%; 255 = 100%)
- frequentie PWM is ongeveer 490 Hz (behalve op de UNO en NANO pin 5 en 6: ongeveer 980 Hz)
- PinMode() is niet nodig bij analogWrite
-

Pulswidth modulation (PWM)



Arduino random-functies

- (long) random()
- (long) random(arg1)
- (long) random(arg1,arg2)
- randomSeed() (long/int)

random(arg1, arg2)

- Genereert een (pseudo-) willekeurig getal tussen grenzen
- Eerste/enige argument: ondergrens
- Tweede argument: bovengrens (niet inbegrepen)
- Geen arg: retourneert willekeurige long int tussen – 2,147.483.648 en 2,147.483.647

Voorbeeld: int lottogetal = random(0,10) levert 0, 1….8, 9

randomSeed(argument)

- om de pseudo-randomgenerator een nieuwe reeks getallen te laten genereren
- Argument is een int of een long int-type

  vb: void randomSeed(42)
- Tip: gebruik een niet aangesloten analoge pen als zaadje

  vb: void randomSeed(analogRead(A0))

En toen kwam er een varken...

Voorbeeld:

knipperLED (met serieweerstand) op digitale pen 13

Programma:

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

void loop()
{
digitalWrite(13, HIGH);
delay(1000);
digitalWrite(13, LOW);
delay(1000);
}
```

Voorbeeld: een led dimmen

Sluit een led in serie met een weerstand aan D9 en compileer de sketch:

```
int led = 9;
int helderheid = 0;
int toename = 5;

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

void loop()
{
analogWrite(led, helderheid);
helderheid = helderheid + toename;
if (helderheid == 0 II helderheid == 255);
      {
      toename = – toename;
      }
delay(300);
}
```

Uitleg bij de sketch:

pinMode(2,OUTPUT**)**

je moet aangeven of een bepaalde aansluitpen (2) als input of output gebruikt wordt
Pinmode() is een ingebouwde functie.
Een aansluitpen mag maximaal 20 mA leveren! (en alle pennen samen maximaal ……. mA)

digitalWrite(2, HIGH**)**

je stuurt een 1 of 5V naar de pen. Bij LOW stuur je een 0 of 0V naar de pen.

delay(1000**)**

Ook delay() is een ingebouwde functie. 1000 is een vertraging van 1000 ms.

Toelichting

- Pen D3, D5, D6, D9, D10 en D11 : niet alleen digitale pennen, maar ook voor PWM
- AnalogWrite() is een functie om o.a.van PMW gebruik te maken
- if …. formuleert een voorwaarde voor uitvoering van het blok tussen {}
- De voorwaarde is een boolean OR (II)
- Bemerk: als de OR true is, wordt toename= – toename

Opgaven 1

- 1. Laat 2 LED’s om beurt knipperen
- 2. Laat 6 LED’s achtereenvolgens aangaan (looplicht)
- 3. Laat de RGB-led achtereenvolgens zijn drie primaire kleuren tonen
- 4. Laat een RGB-led alle kleuren van de regenboog maken
- 5. Maak met vijf ledjes een dobbelsteen
- 6. Laat een led onregelmatig knipperen met tussen-tijden van 0,5 en 5,0 seconden