CPROUC/2021-2022

Ad van den Bergh

CPROUC

Introductie Arduino UNO

Week 12

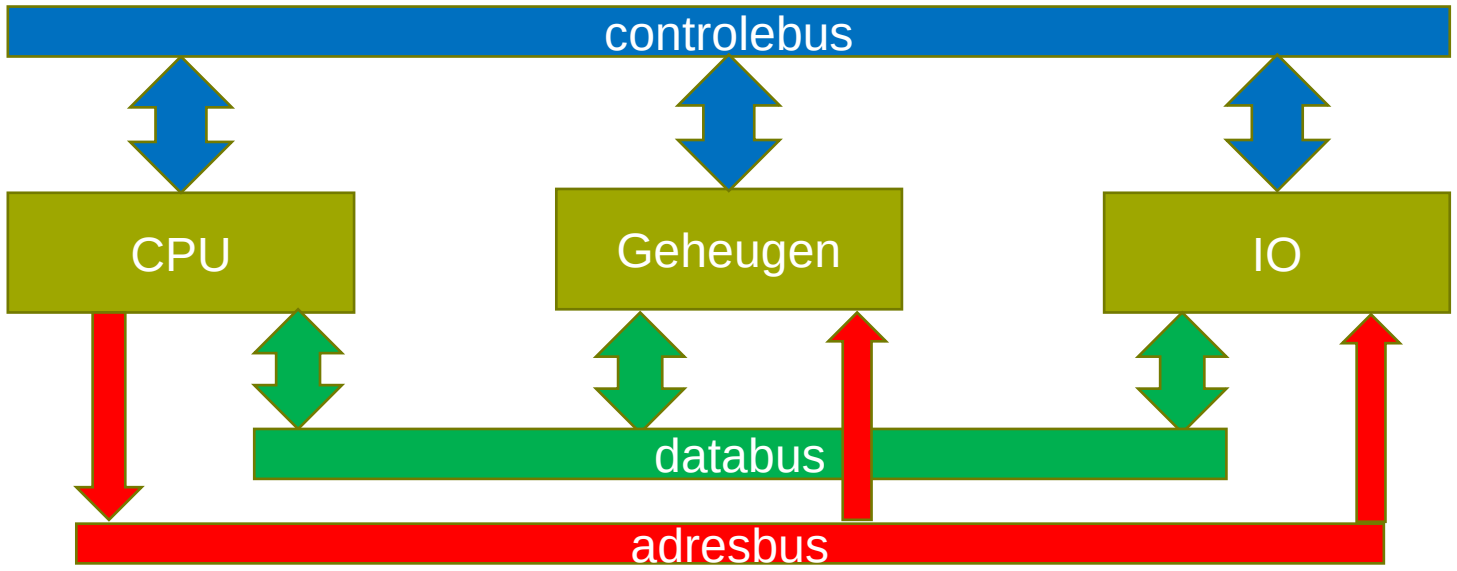
DE HAAGSE HOGESCHOOL

C programmeren voor technische systemen

- In CPROUC was eerste deel gericht op het aanleren van de programmeertaal C voor op een PC of laptop.
- Dat zijn geen “embedded systemen” die gemaakt zijn om één doel te bereiken (apparaat).
- We onderscheiden administratieve systemen en technische systemen, waarbij technische systemen altijd te maken hebben met de werkelijke omgeving.
- Een technisch systeem probeert informatie uit zijn omgeving te halen door gebruik te maken van *sensoren* (temperatuur, druk, licht, toetsen..).
- De software die door de processor wordt uitgevoerd besluit welke acties uitgevoerd moeten worden afhankelijk van de informatie van de *inputs* én de *toestand* waarin het systeem zich bevindt.
- De acties kunnen resulteren in het aansturen van de *outputs* waaraan *actuatoren* zijn aangesloten (motoren, leds, relais, verwarmingselementen).

Microprocessorsysteem

- De processor is de CPU en die kan via bussen communiceren naar andere componenten, zoals geheugens, hard-disks, latches etc.
- De adresbus wordt alleen geschreven door de CPU en de databus en controlbus kunnen beschreven en gelezen worden vanuit allerlei componenten (als die componenten toestemming hebben van de CPU.



DE HAAGSE HOGESCHOOL

MicroCONTROLLERsysteem

- componenten CPU, RAM, ROM en bussen intern op één chip.
- Alleen kristal voor de oscillator en inputs en outputs.
- Voordelen:
  - Compleet systeem op één chip.
  - Goedkoop
- Nadelen:
  - Slecht uitbreidbaar
  - Vaak in- en outputs met speciale functies (Tx, Rx, analoge inputs en outputs met PWM)

ATmega328

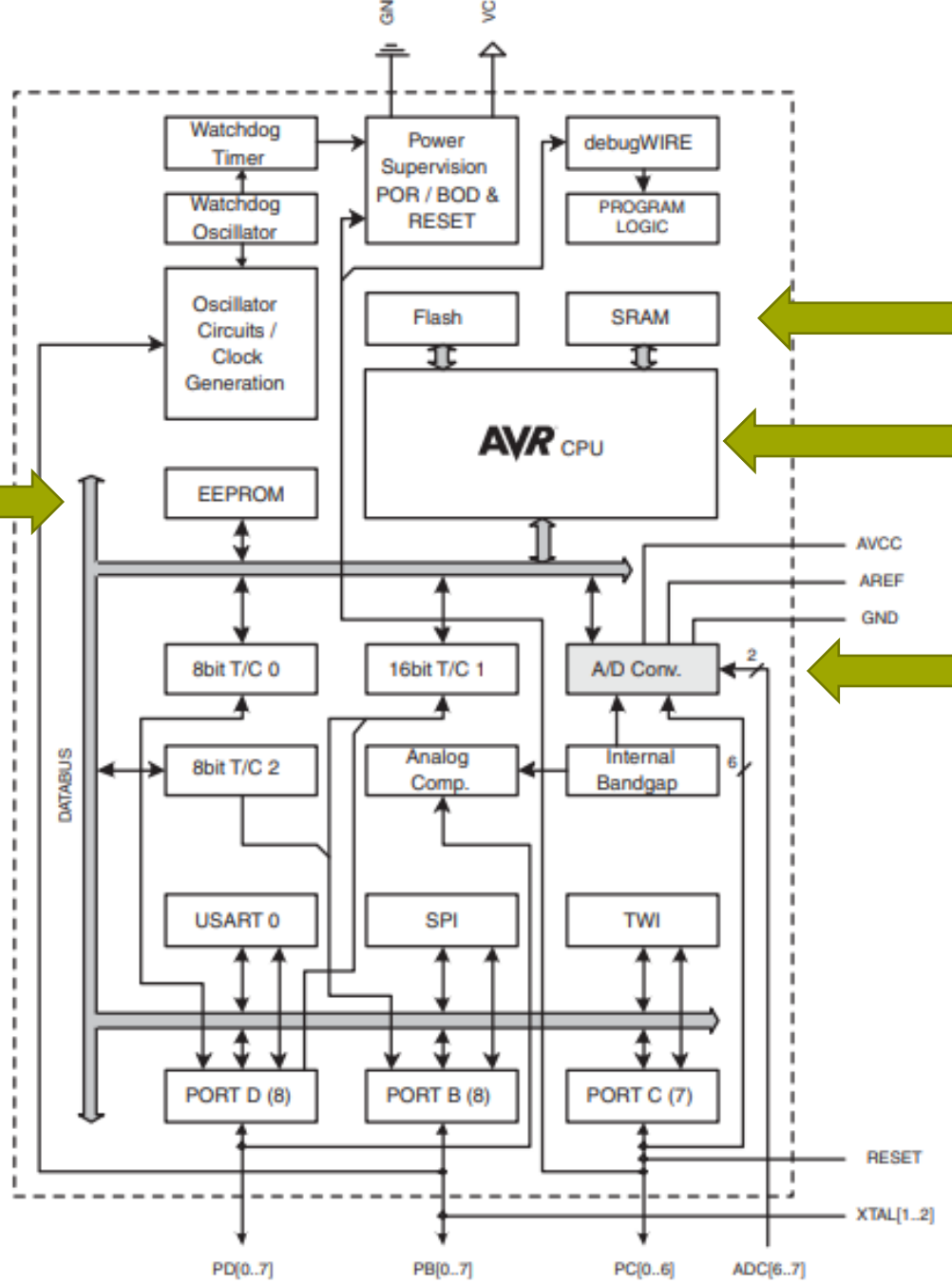
Allemaal in één chip

geheugen

3x timer/counter

Seriële communicatie

I/O poorten



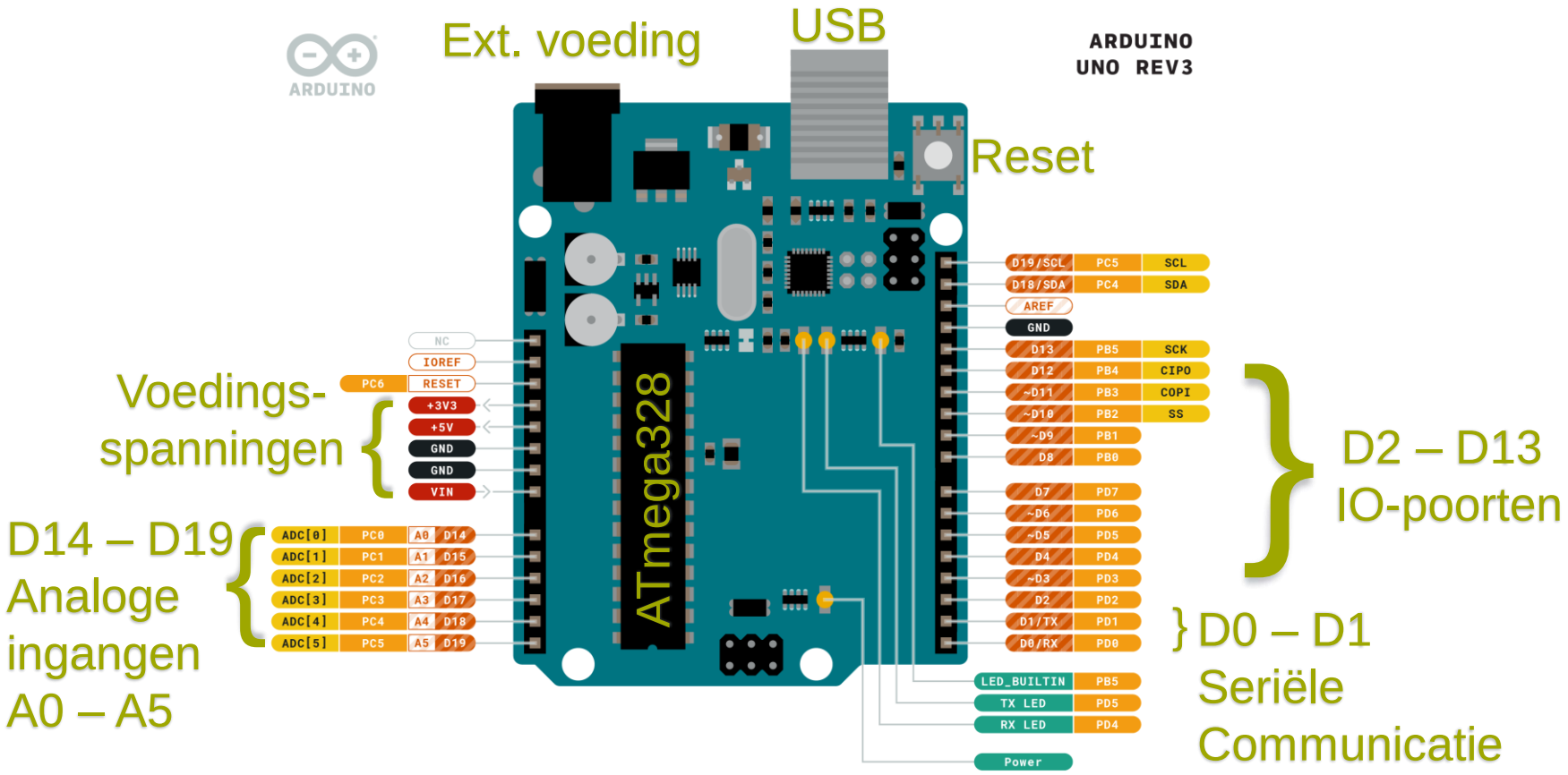
geheugen

Central Processing Unit

Analoog/ Digitaal Converter

DE HAAGSE HOGESCHOOL

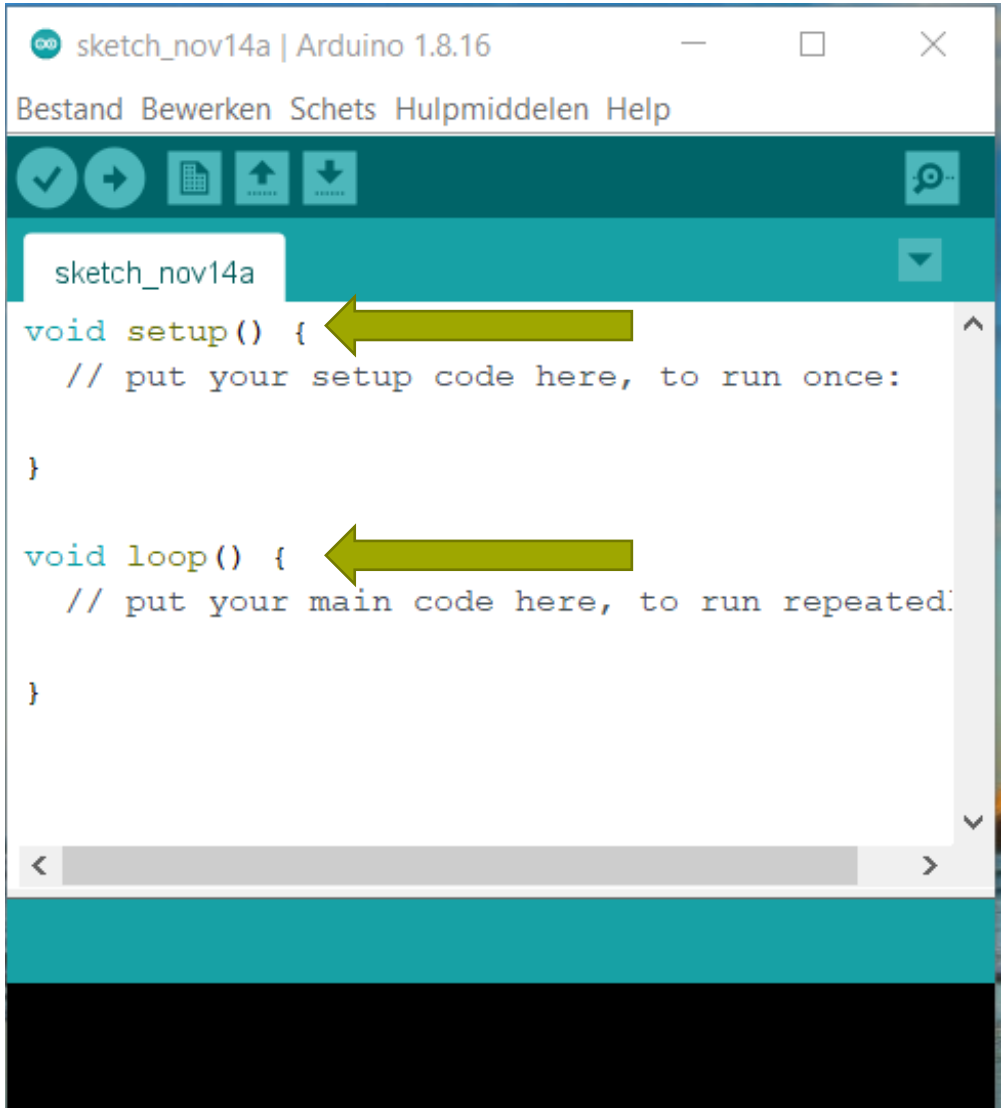
ARDUINO UNO aansluitingen



DE HAAGSE HOGESCHOOL

ARDUINO Software structuur

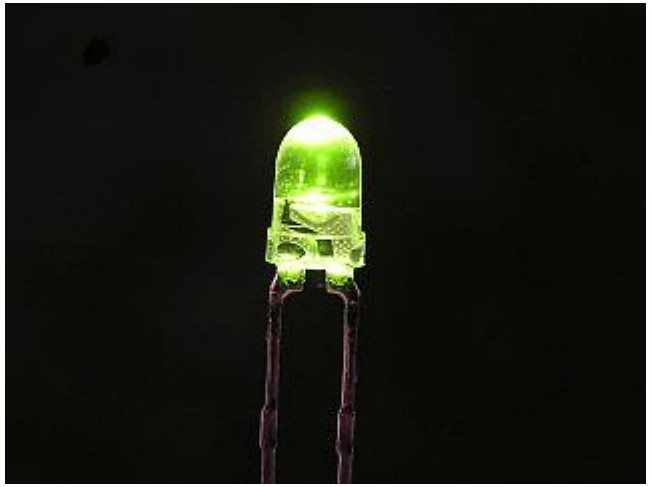
- Arduino heeft geen Operating System
- Een programma bestaat uit een Setup en een eindeloze Loop
- Setup:
  - Wordt maar één keer uitgevoerd
  - Configuratie van IO-poorten en initiatie en evt. testprogramma, welkom…..
- Loop:
  - Eindeloze loop – Round Robin principe
  - Eenvoudigste OS
  - Als loop maar snel genoeg dan geen probleem
- Nadeel:
  - Geen supersnelle verwerking mogelijk (tenzij interrupts worden gebruikt)



DE HAAGSE HOGESCHOOL

“Hello World” voor technische systemen

- Bijna ieder boek over een programmeertaal begint met het voorbeeld om de tekst “Hello World” op een scherm te laten verschijnen.
- Technische systemen hebben (vaak) geen beeldscherm ter beschikking en daarom is het standaard voorbeeld altijd: “The Blinking Led”.



De sketch voor de “Blinking Led”

- Een programma bij Arduino wordt een “sketch” genoemd.
- Arduino sketches zijn gebaseerd op de programmeertaal C.
- Maar er zijn extra instructies toegevoegd om het eenvoudiger te maken om de hardware van de Arduino te besturen.
- Dit heet de Hardware Application Library (HAL).
- Onder andere om de IO-poorten te configureren als input of als output:
  - Als de Arduino spanning krijgt worden alle IO-poorten (vanwege de veiligheid) als input geschakeld (dus geen uitgangsspanning maar een ingang met hoge impedantie).
  - Een IO-poort kan als uitgang geschakeld worden door de HAL-instructie: `pinMode( pinnummer, OUTPUT);`

De setup() voor de “Blinking Led”

```
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27   // initialize digital pin LED_BUILTIN as an output.
28   pinMode(LED_BUILTIN, OUTPUT);
29 }
```

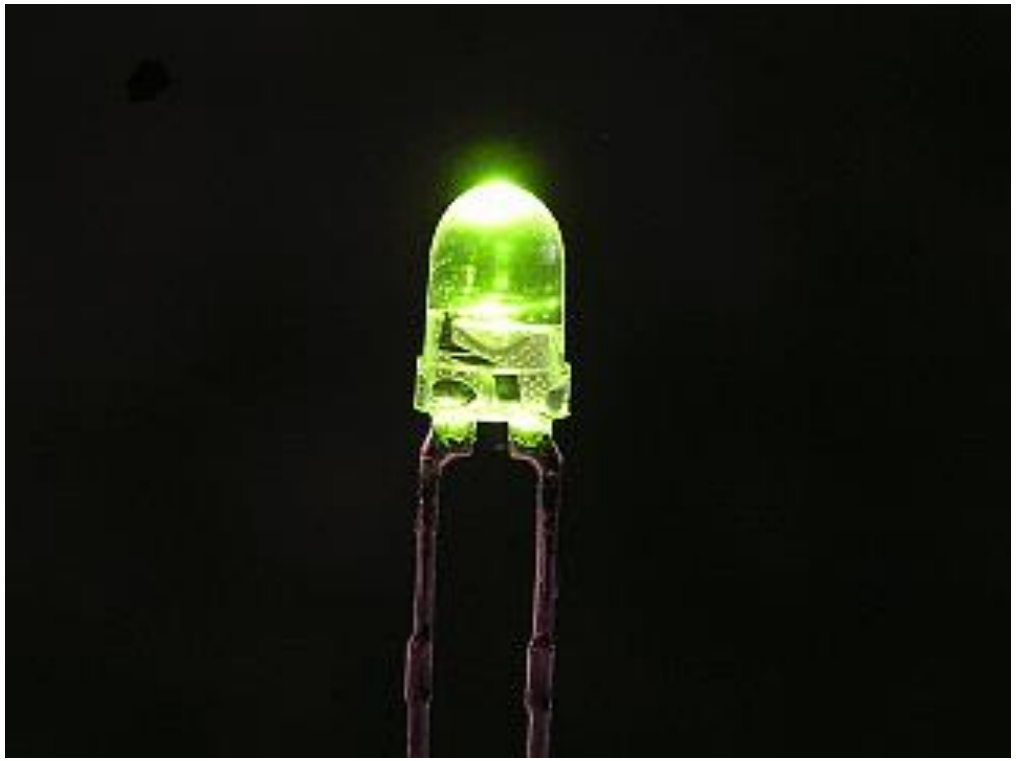
- `LED_BUILTIN` is bij de Arduino UNO pin 13 wat het gele ledje is wat op het board is geplaatst. { `pinMode(13, OUTPUT);` is ook goed}
- Let op hoofdletters en kleine letters (case sensitive) !

De loop() voor de “Blinking Led”

```
31 // the loop function runs over and over again forever
32 void loop() {
33   digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
34   delay(1000);                       // wait for a second
35   digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
36   delay(1000);                       // wait for a second
37 }
```

- In regel 33 schrijft `digitalWrite` de waarde van `HIGH` (+5V) naar output `LED_BUILTIN` (pin 13) waardoor de gele led op het board aan gaat.
- In regel 34 zorgt `delay(1000);` ervoor dat de processor 1000 ms (dus 1 seconde) wacht.
- In regel 35 wordt de gele led weer uitgezet (`LOW` is 0V).
- Alle gebruikte functies en constanten zijn gedefinieerd in de HAL.
- Let op hoofdletters en kleine letters (case sensitive) !

Demo van de “Blinking Led”



Setup voor 3 leds met test (antwoorden)

```
 5 #define AAN LOW
 6 #define UIT HIGH
 7
 8 void setup() {
 9   pinMode(13, OUTPUT);
10   pinMode(12, OUTPUT);
11   pinMode(11, OUTPUT);
12   for (uint8_t i = 0; i < 3; i++) {
13     digitalWrite(11 + i, AAN); // alle leds aan voor test
14   }
15   delay(1000);
16   for (uint8_t i = 0; i < 3; i++) {
17     digitalWrite(11 + i, UIT); // alle leds weer uit
18   }
19 }
```

- Ledtest: Door for-loop worden de leds op pin 11, 12 en 13 eerst aangezet en na 1 seconde weer uitgezet.
- Door de `#define` kan nu heel makkelijk op één plaats omgeschakeld worden als de leds anders aangesloten worden.

Loop voor 3 leds (looplicht)

```
22 void loop() {
23   digitalWrite(13, AAN);    // Led 13 aan
24   delay(250);               // wacht kwart seconde
25   digitalWrite(13, UIT);    // Led 13 weer uit
26   digitalWrite(12, AAN);    // Led 12 aan
27   delay(250);               // wacht kwart seconde
28   digitalWrite(12, UIT);    // Led 12 weer uit
29   digitalWrite(11, AAN);    // Led 11 aan
30   delay(250);               // wacht kwart seconde
31   digitalWrite(11, UIT);    // Led 11 weer uit
32 }
```

- Dit werkt wel, maar het kan slimmer! Hoe?

Slimmere Setup voor 3 leds met test

```
5  #define AAN LOW
6  #define UIT HIGH
7  int ledPins[] = { 8, 9, 10};   // array voor aansluitingen LEDs
8  uint8_t i;
9  uint8_t aantalLEDs = sizeof( ledPins) / sizeof (ledPins[0]);
10
11 void setup() {
12   for (i = 0; i < aantalLEDs; i++) {
13     pinMode(ledPins[i], OUTPUT);      // zet de digitale pin als output (uitgang)
14   }
15 }
```

- Nu staan pinnummers in een array en wordt het aantal automatisch aangepast.
- Dat kan dan ook gebruikt worden in de loop……

Slimmere Loop voor 3 leds (looplicht)

```
18 void loop() {
19   for (i = 0; i < aantalLEDs; i++) {
20     digitalWrite(ledPins[i], AAN);  // zet de LED aan
21     delay(250);                     // wacht 1/4 seconde (= 250 ms)
22     digitalWrite(ledPins[i], UIT);  // zet de LED uit
23   }
24 }
```

- Het aantal regels code is aanzienlijk verminderd én…
- Je hoeft alleen de array met pinnummers aan te passen als het aantal leds veranderd.
- Dus ook een RGB-led is met dezelfde sketch aan te sturen……
- Demo ..

Gebruik van digitale inputs

- Ook inputs kunnen worden geconfigureerd met twee opties:
  - `pinMode( pinnummer, INPUT);`
  - `pinMode( pinnummer, INPUT_PULLUP);`
- De eerste vormt een hoogohmige input , terwijl de tweede een interne pull-up weerstand ingeschakeld zodat als de ingang niet aangesloten is er een logische 1 wordt gelezen.
- Dat scheelt met name bij toetsen en schakelaars weerstanden.
- Let op: als de normale INPUT wordt gekozen en er wordt niets op de ingang aangesloten, dan kan de ingang door o.a. EMC of aanraking onverwacht reageren.
- Het inlezen van de waarde van een digitale input vindt plaats met de instructie: `digitalRead( pinnummer);`
- De waarde die teruggegeven wordt is 0 of 1 en is dus ook als een boolean waarde te interpreteren.

De setup voor een looplicht

```
 9 int ledPins[] = { 8, 9, 10};   // array voor aansluitingen LEDs
10 uint8_t i, teller = 0;
11 uint8_t aantalLEDs = sizeof( ledPins) / sizeof (ledPins[0]);
12 bool toetsgedrukt = false;
13 #define Toets 4
14
15 void setup() {
16   pinMode(4, INPUT_PULLUP);
17   for (i = 0; i < aantalLEDs; i++) {
18     pinMode(ledPins[i], OUTPUT);      // zet de digitale pin als output (uitgang)
19   }
20 }
```

- Haal straks bij de demo de `_PULLUP` maar eens weg uit regel 16……
- Maar nu eerst kijken naar de `loop()`.

Waarom?

De loop voor een looplicht

```
22 void loop() {
23   for (i = 0; i < aantalLEDs; i++) {
24     if (i == teller) {
25       digitalWrite(ledPins[i], AAN);  //      de LED aan
26     }
27     else {
28       digitalWrite(ledPins[i], UIT);  //      de LED uit
29     }
30   }
31   while (!digitalRead(Toets)); // wacht tot toets is losgelaten
32   while (digitalRead(Toets));  // wacht tot toets wordt gedrukt
33   teller ++;
34   if (teller >= aantalLEDs) {
35     teller = 0;
36   }
37   delay(25); // om denderen toets op te vangen
38 }
```

Debouncing

- Kleine (goedkope) toetsen (maar ook relais) hebben vaak “denderende” contacten, waardoor de processor denkt dat een toets meerdere malen is ingedrukt.
- Bij tellers levert dat problemen op.
- Oplossing: debouncing.
- In vorige sketch opgelost door onderin de loop() een delay op te nemen van 25 ms.
- Dat kan alleen als er niets anders te doen is.
- Mooier om te controleren na 25 ms of de toets nog steeds ingedrukt is; zo niet…. Nog maar een keer wachten op de toetsdruk!
- Zie de oplossing in de volgende sheet….

Debouncing (beter)

```
32    while (!digitalRead(Toets)); // wacht tot toets is losgelaten
33    while (digitalRead(Toets)); // wacht tot toets wordt ingedrukt
34    toetsgedrukt = false;
35    while (!toetsgedrukt){
36      delay(25);
37      if (digitalRead(Toets)) toetsgedrukt = true;
38    }
39    teller ++;
40    if (teller >= aantalLEDs) {
41      teller = 0;
42    }
43  }
```

- Natuurlijk moet toetsgedrukt wel bovenin gedeclareerd zijn als

```
bool toetsgedrukt = false;
```

- Demo verbeterde debouncing.

let’s change