CPROUC/2021-2022

Ad van den Bergh

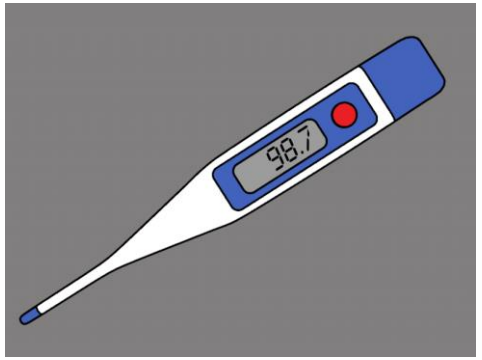
CPROUC

USART en ADC

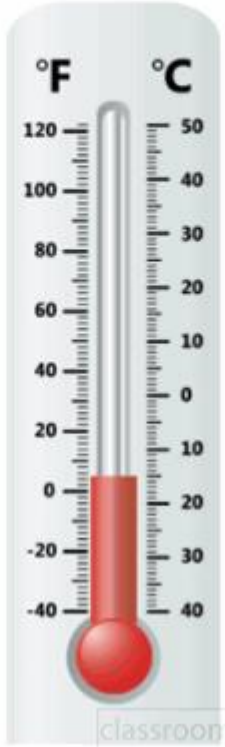
Week 13

DE HAAGSE HOGESCHOOL

Analoge signalen



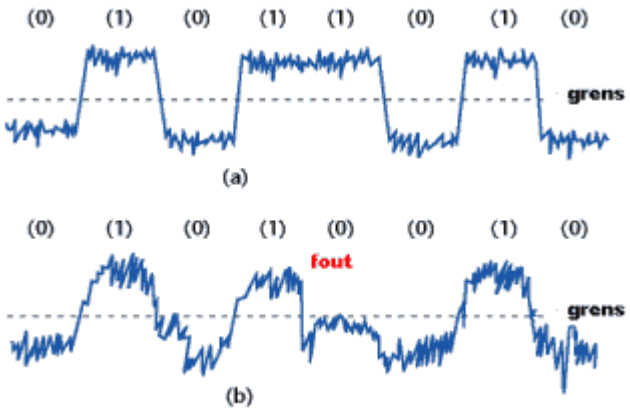
- Technische systemen proberen zichzelf een beeld te vormen van hun omgeving door te meten met *sensoren*.
- In de vorige les hebben we alleen digitale input bekeken met de instructie `digitalRead(pinnummer);`
- Dat gaat goed bij toetsen of sensoren die maar twee waarden afgeven: logisch ‘1’ of ‘0’ (true en false).
- Maar fysieke grootheden, zoals temperaturen, zijn niet tweewaardig, maar kunnen tussen bepaalde grenzen ALLE waarden hebben.



DE HAAGSE HOGESCHOOL

Waarom digitaal maken?

- Analoge signalen zijn storingsgevoeliger dan digitale signalen.
- Stoorsignalen t.g.v. EMC, maar ook overspraak veranderen het signaal (vooral tijdens transport over grote afstanden; telecommunicatie).
- Dit wordt meestal ervaren als *ruis*.
- Bij digitale signalen heeft de ruis minder invloed, als de stoorspanning maar niet te groot wordt en de grens niet wordt overschreden.



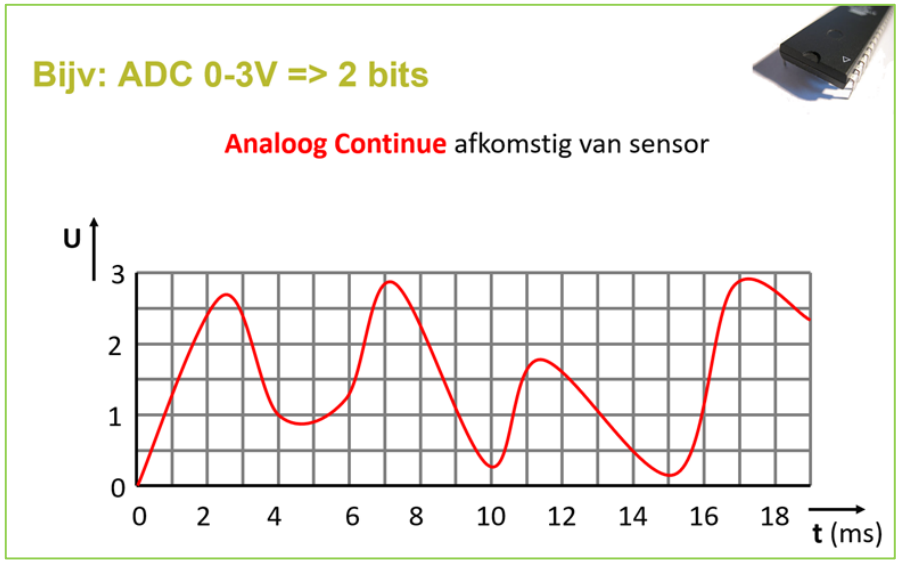
DE HAAGSE HOGESCHOOL

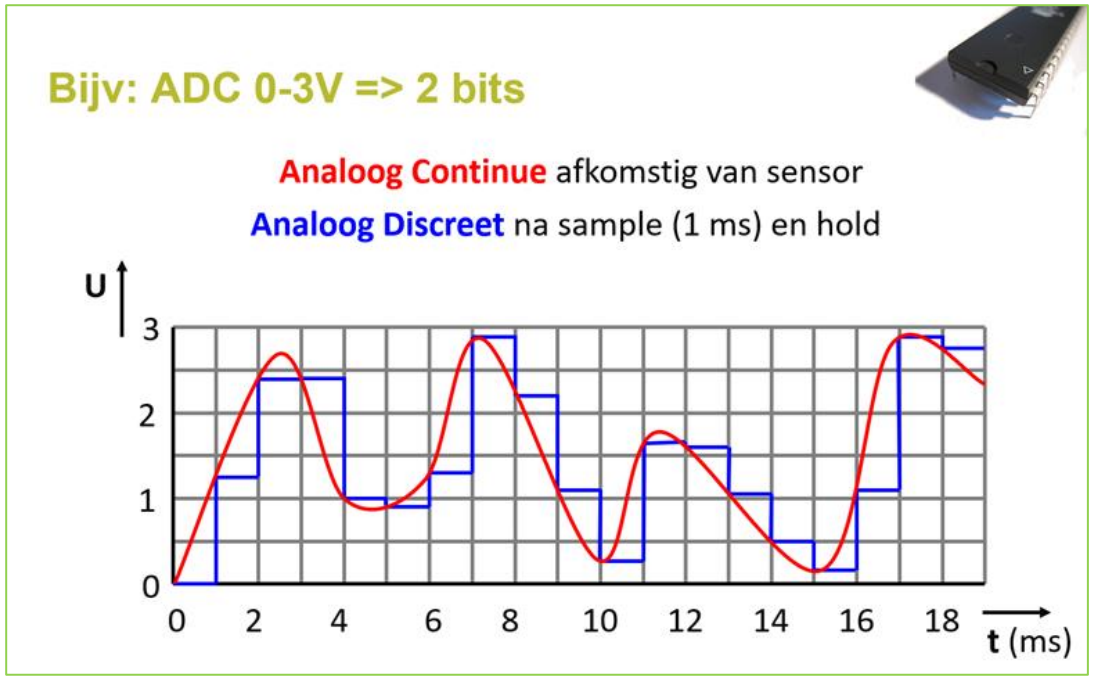
Analoog Digitaal Converter

- Bovendien kunnen digitale signalen worden verwerkt door een processor en analoge signalen niet.
- Dus willen we analoge signalen omzetten naar een reeks digitale *waarden.*
- Dat kan door op vaste regelmatige momenten de analoge waarde vast te houden (bemonsteren = sample) en snel om te zetten naar een digitale waarde (kwantiseren).
- Het eerste heet een Sample & Hold functie (S&H).
- Het kwantiseren doet een Analoog naar Digitaal Converter (ADC).

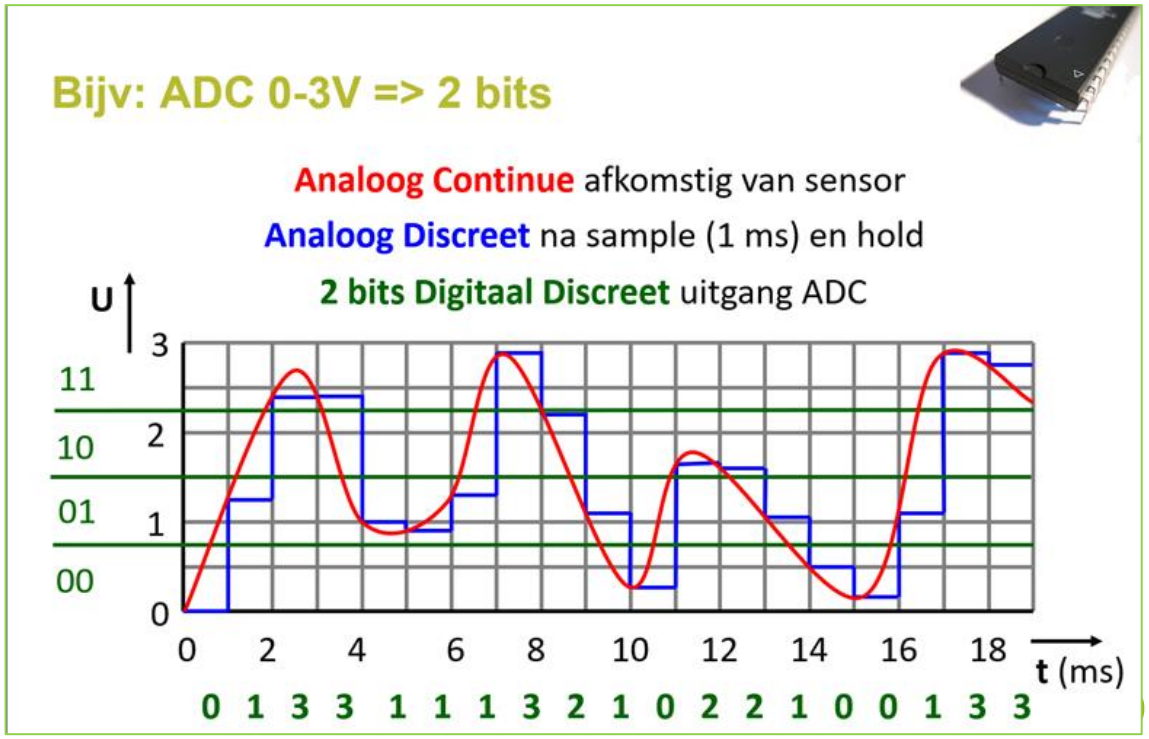
Het AD-proces

Sample & Hold





Kwantiseren door ADC

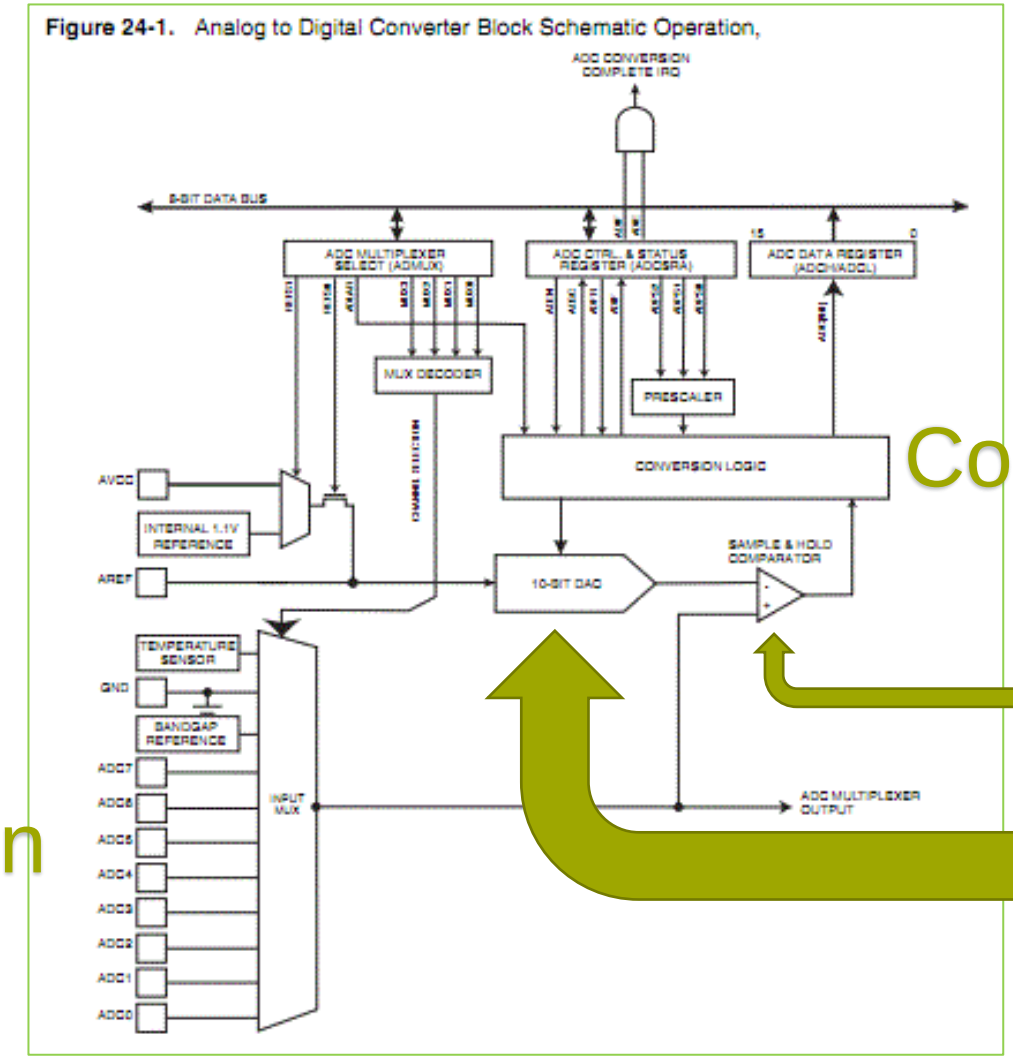


Kwantiseringsfout

- Er wordt bij kwantisering altijd een maximale fout gemaakt van de helft van één gebiedje door afronding.
- In het voorbeeld is één gebiedje gelijk aan 3 V/4 gebiedjes = 0,75 V per gebiedje.
- De maximale kwaniseringsfout is dus 0,75 / 2 = 0,375 V
- Dat betekent dat voor een signaalwaarde in het laagste vakje de kwantiseringsfout gelijk is aan 0,375/0,75 = 50%.
- Voor het bovenste vakje is dit 0,375/3 = 12,5 %.
- Conclusie: hoe groter het signaal, hoe minder groot de kwantiseringsfout.
- Veel gebruikte oplossing: Automatic Gain Control en niet-lineaire ADC gebruiken.

ADConverter in Atmega 328

Bron: documentatie Atmel



Conversie logica

Comparator

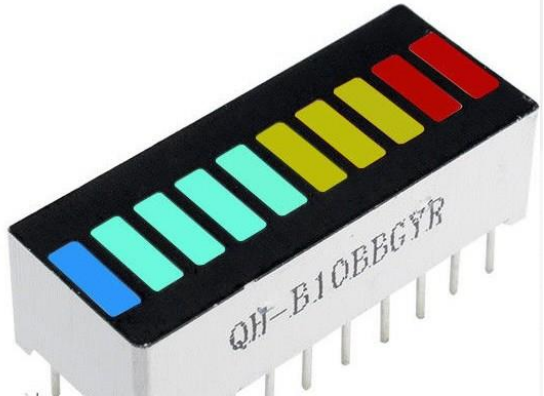
10 bits DAC

8 ADC kanalen

DE HAAGSE HOGESCHOOL

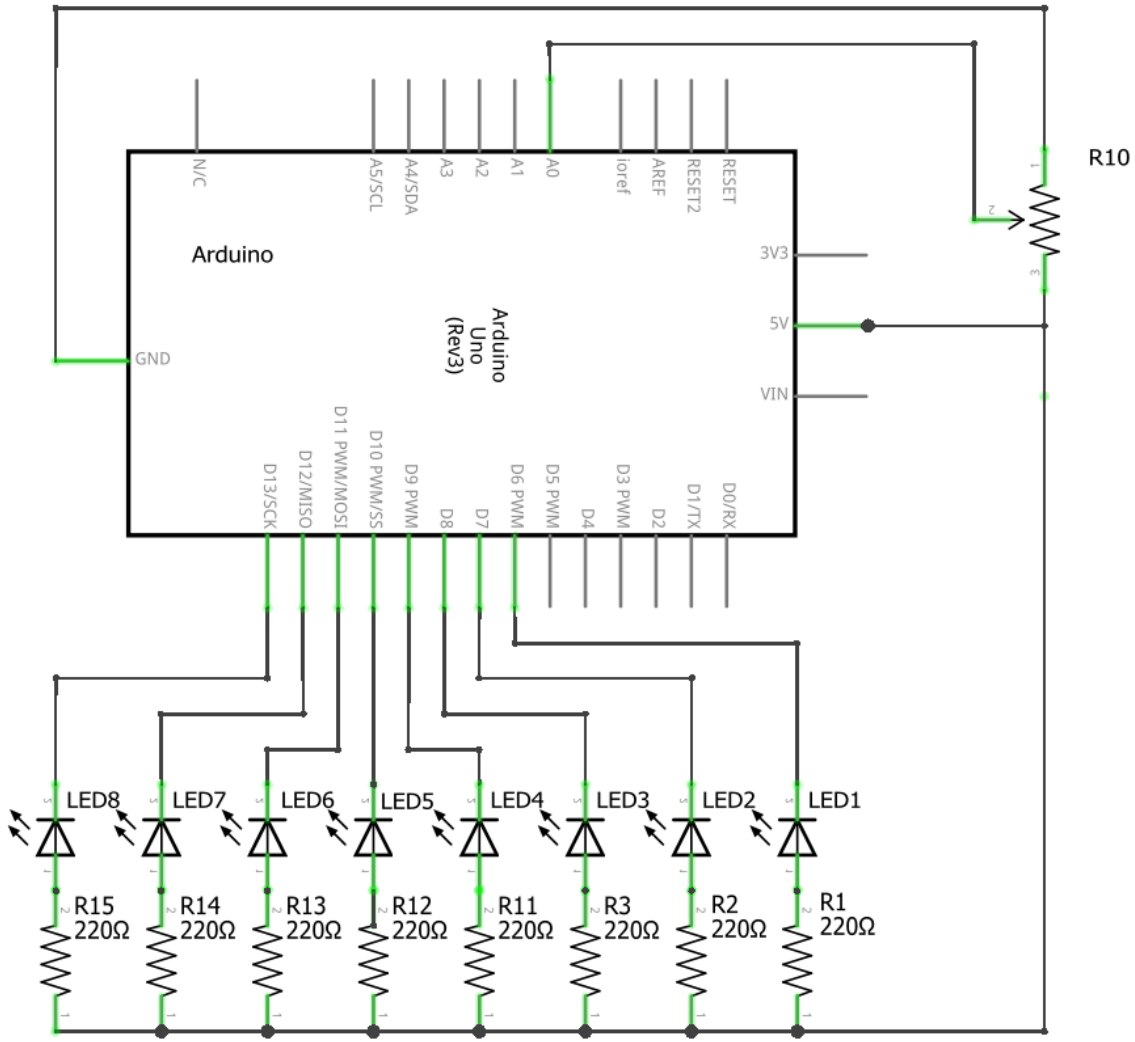
Analoge Ledbar

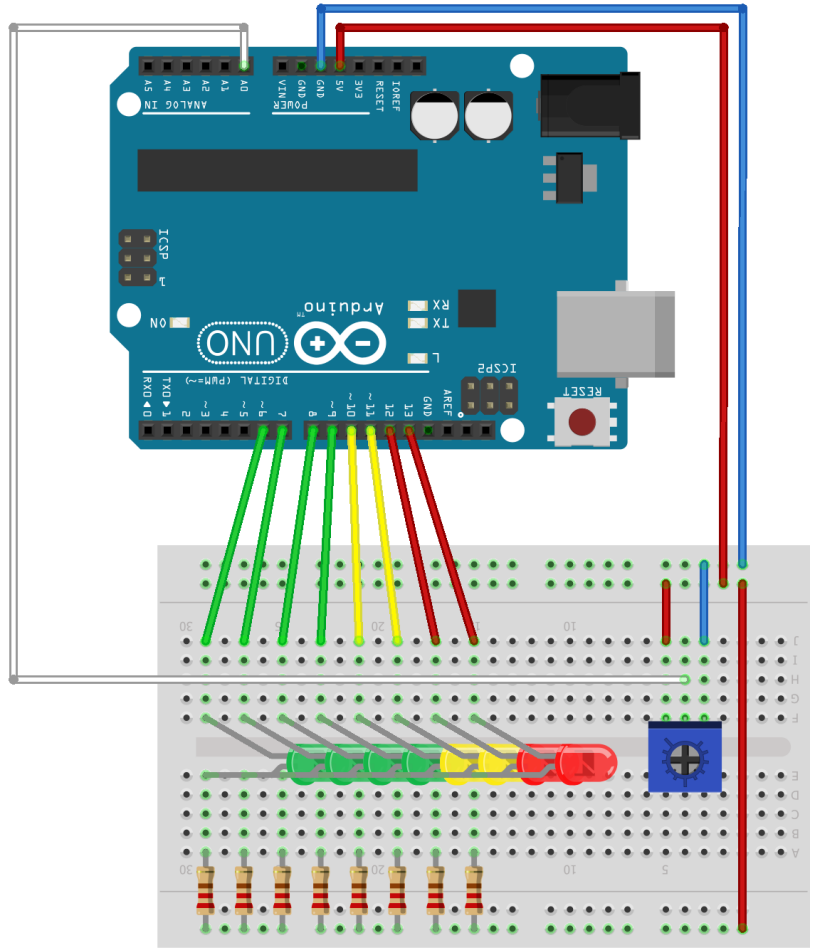
- Met een potentiometer de spanning op een ADC ingang veranderen en op 8 leds de waarde laten zien. (VU-meter uit PROJE1).
- De ingangsspanning kan geregeld worden tussen 0 en 5 Volt.
  - Bij 5 V moeten alle leds aan.
  - Bij 0 V moeten alle leds uit.
  - Bij 2,5 V moeten de eerste 4 leds aan.
- Leds op pin 6 t/m 13.
- Loper van de potmeter op A0



DE HAAGSE HOGESCHOOL

Schema Analoge Ledbar





Mogelijke opstelling

DE HAAGSE HOGESCHOOL

Setup() van de Analoge Ledbar

```
const int ledPins[] = {6, 7, 8, 9, 10, 11, 12, 13}; // array voor aansluitingen LEDs
const int analoogInPin = 0;  // Analoge input aangesloten op loper van de potmeter
uint8_t aantalLEDs = sizeof( ledPins) / sizeof ledPins[0];

const boolean LED_AAN = LOW;
const boolean LED_UIT = HIGH;

int sensorWaarde = 0;  // gelezen waarde van de spanning op de loper van de potmeter
int ledNivo = 0;       // spanning vertaald naar aantal LEDs die aan moeten

void setup()
{
  for (int led = 0; led < aantalLEDs; led++)
  {
    pinMode(ledPins[led], OUTPUT);   // zet de digitale pin als output (uitgang)
  }
}
```

ADC Arduino is 10 bits dus maximaal 1023
Kan dus niet met uint8_t

DE HAAGSE HOGESCHOOL

Loop() van de Analoge Ledbar

```
void loop()
{
  sensorWaarde = analogRead(analoogInPin);  // lees de analoge waarde in
  // verdelen over aantal leds
  ledNivo = map(sensorWaarde, 0, 1023, 0, aantalLEDs+1);
  for (int led = 0; led < aantalLEDs; led++)
  {
    if (led < ledNivo )
    {
      // LEDs aan als de gemeten spanning hoger is
      digitalWrite(ledPins[led], LED_AAN);
    }
    else {
      digitalWrite(ledPins[led], LED_UIT); // zet de LED uit
    }
  }
}
```

Functie map()

- `map(value, minIn, maxIn, minOut, maxOut);`
- Handige functie om schaling mee uit te voeren
- minIn is de minimale waarde van de input
- maxIn is de maximale waarde van de input
- minOut is de minimale waarde van de output
- maxOut is de maximale waarde van de output

- Wat doet `map(waarde, 0, 1023, 0, 255);` ?
- Delen door 4
- Wat doet `map(waarde, 0, 1023, 0, -127);` ?
- Delen door 8 en negatief maken
- Wat doet `map(waarde, 0, 1023, 128, 255);` ?
- Delen door 8 en er 128 bij optellen

Demo

Debuggen van een programma

- In CPROUC werd in het eerste blok gebruik gemaakt van Visual Studio.
- Daarin was een debugger waarmee het programma gestopt kon worden en stapsgewijs door het programma gelopen kon worden.
- Ook de actuele waarden van de variabelen konden zichtbaar gemaakt worden.
- Erg handig (en eigenlijk een MUST) bij het programmeren.
- Maar……
- **De Arduino IDE heeft géén debugger.**

DE HAAGSE HOGESCHOOL

Setup() van de Analoge Ledbar

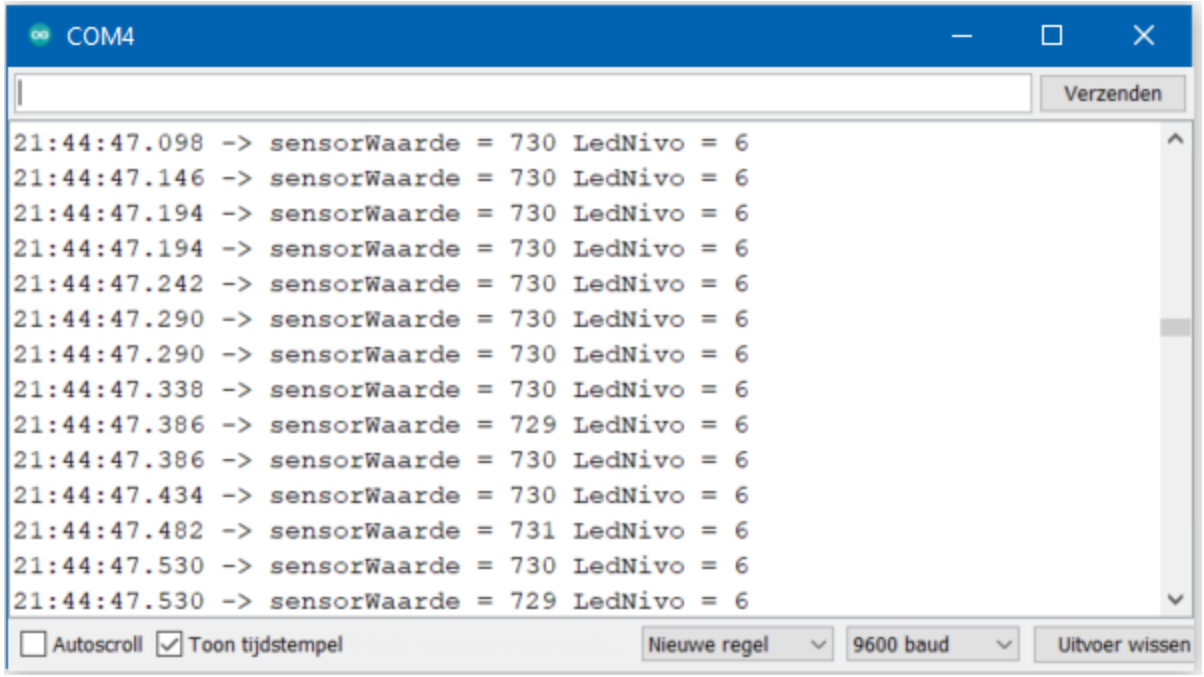
```
void setup()
{
  for (int led = 0; led < aantalLEDs; led++)
  {
    pinMode(ledPins[led], OUTPUT);
  }
  Serial.begin(9600); // open een seriële port
}
```

9600 bits per seconde

Loop() van de Analoge Ledbar met monitor

```
void loop()
{
  sensorWaarde = analogRead(analoogInPin);
  // verdelen over aantal leds
  Serial.print("sensorWaarde = ");
  Serial.print(sensorWaarde);
  ledNivo = map(sensorWaarde, 0, 1023, 0, aantalLEDs + 1);
  Serial.print(" LedNivo = ");
  Serial.println(ledNivo);
  for (int led = 0; led < aantalLEDs; led++) {
……
```

Analoge Ledbar met monitor



Demo

Handig voor looptijd

Communicatiesnelheid

Output laten scrollen

DE HAAGSE HOGESCHOOL

let’s change