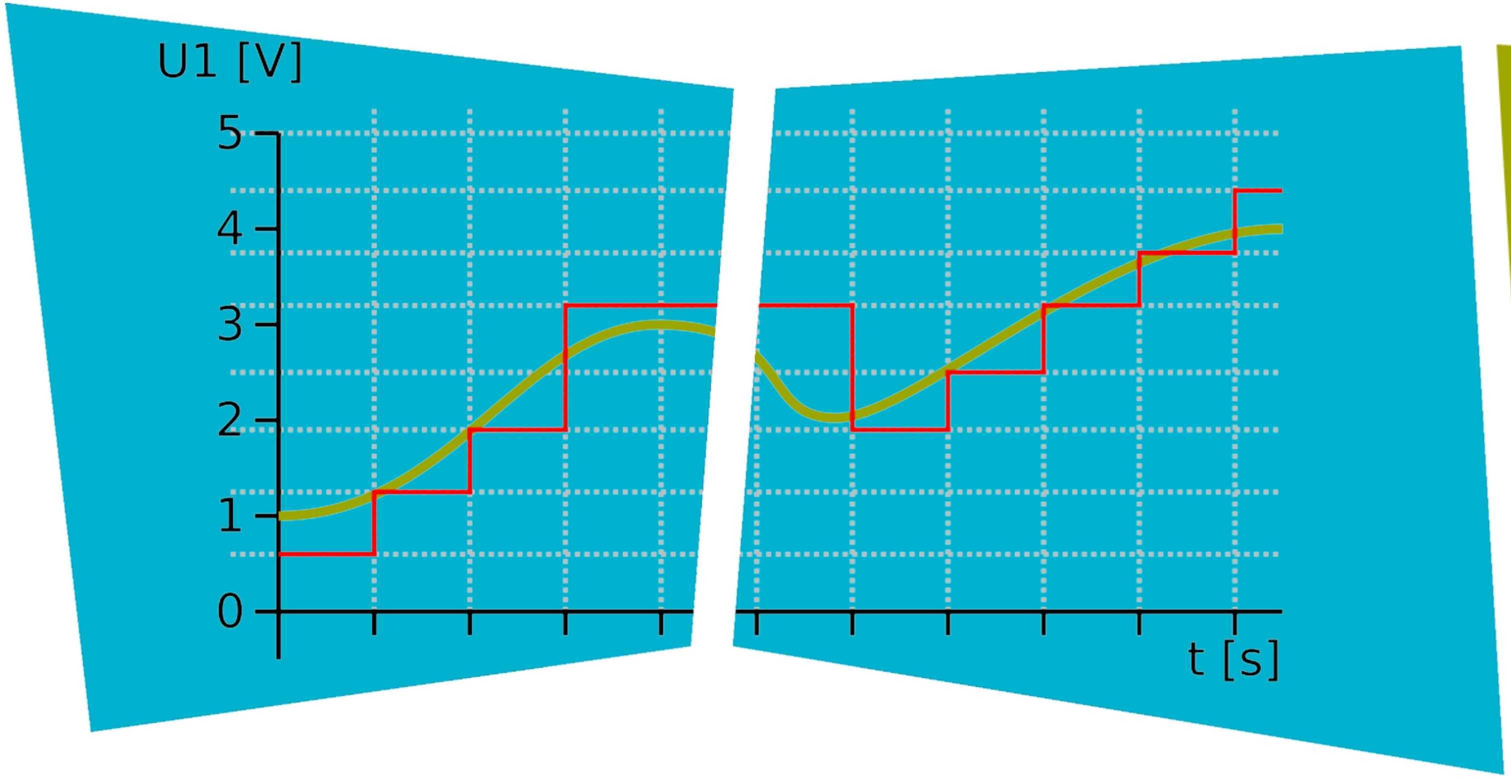


2021

G.M. Tuk
X. van Rijnsoever

ANALOOG-DIGITAALOMZETTER

Microcontroller Programmeren 2 – Week 5

DE HAAGSE HOGESCHOOL

Wat gaan we doen vandaag?

Gebruik `volatile`

Gebruik maken van de ADC:

- Kwantisatie en bemonstering
- Instellen van de ADC
- Uitlezen via polling
- Interrupts

Volatile

- De compiler maakt zo efficiënt mogelijke code
- Variabelen waar niets mee wordt gedaan of die niet worden veranderd, worden weggelaten
- Code die daarvan gebruik maakt, wordt weggelaten

```
for (int i = 0; i < 10000; i++)
{
}
```


DE HAAGSE HOGESCHOOL

Volatile

- Zelfde probleem in ISR's

```
int flag = 0;

ISR(TIMER0_OVF_vect)
{
    flag = 1;
}
```

```
int main(void
{
    …
    if (flag)
    {
        ...
    }
    ...
}
```

Compiler ziet niet dat deze flag wordt aangepast

Volatile

- Gebruik van keyword `volatile` zorgt ervoor dat de compiler de variabele niet weg-optimaliseert
- Gebruik dit voor variabelen die communiceren tussen ISR en andere delen van je programma

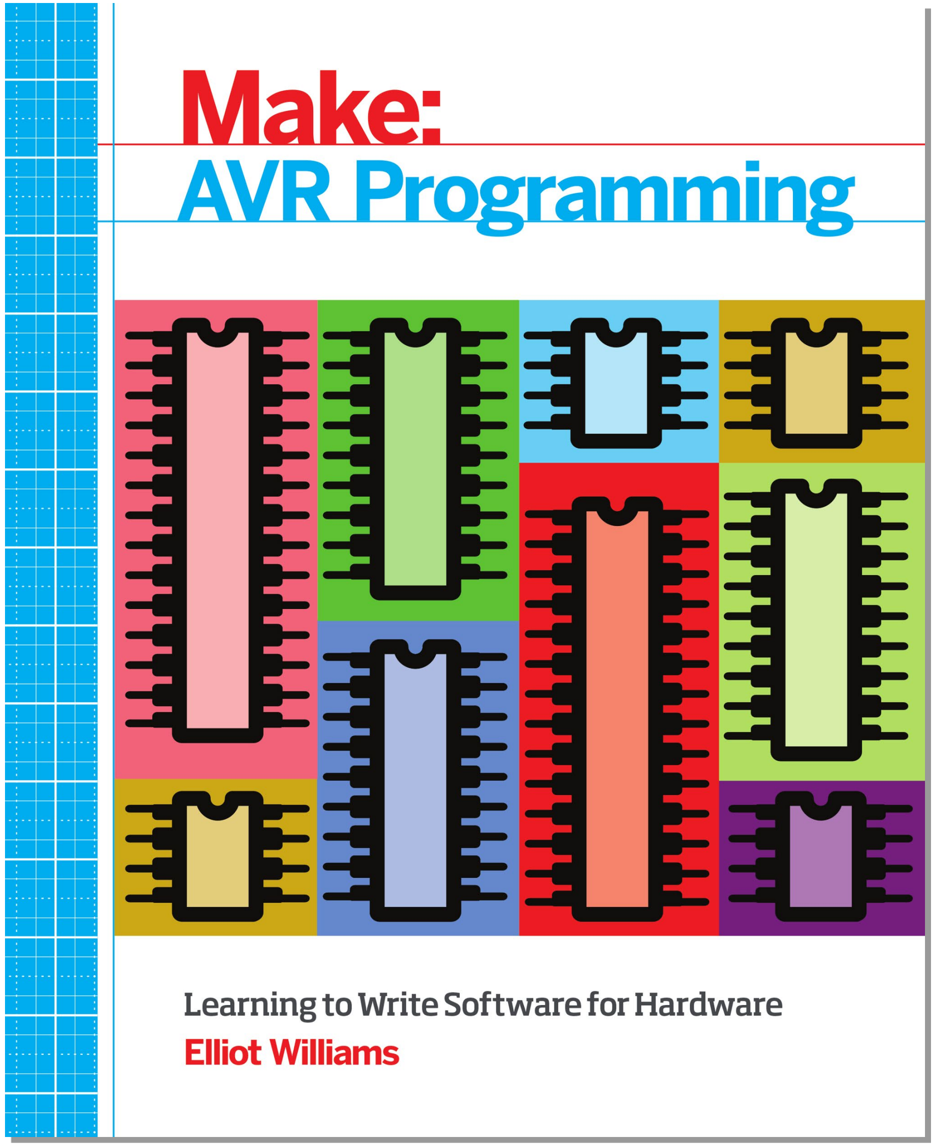
```
volatile int i = 0;
```


DE HAAGSE HOGESCHOOL

Williams

- H1 inleiding
- H2 omgeving
- H3 breadboards, led knipperen
- H4 “Bit Twiddling”
- H6 debouncing
- H8 blok 2
- H9 blok 2

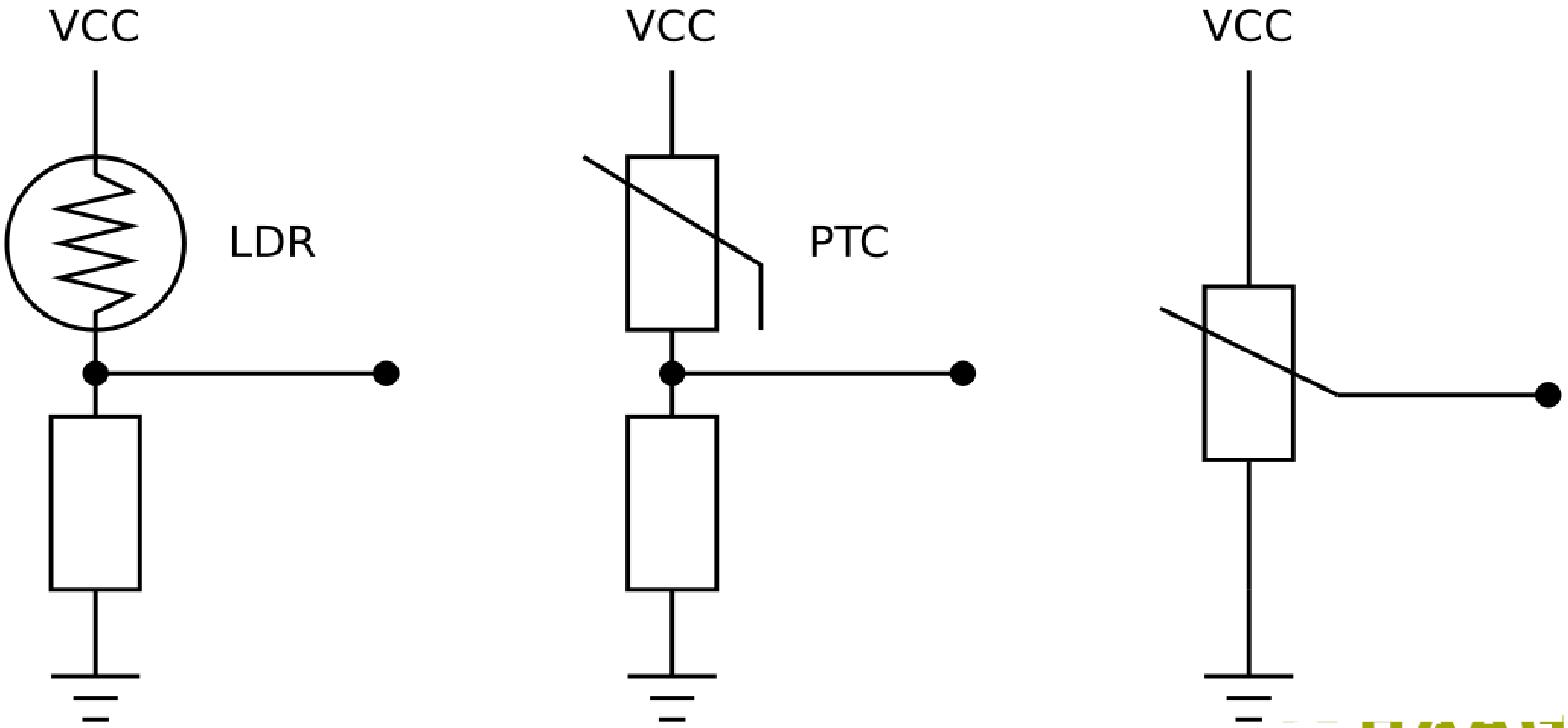
- H5, H7, H10, H11 H12: nuttige dingen voor project



DE HAAGSE HOGESCHOOL

Waarom de ADC?

Waarom de ADC?



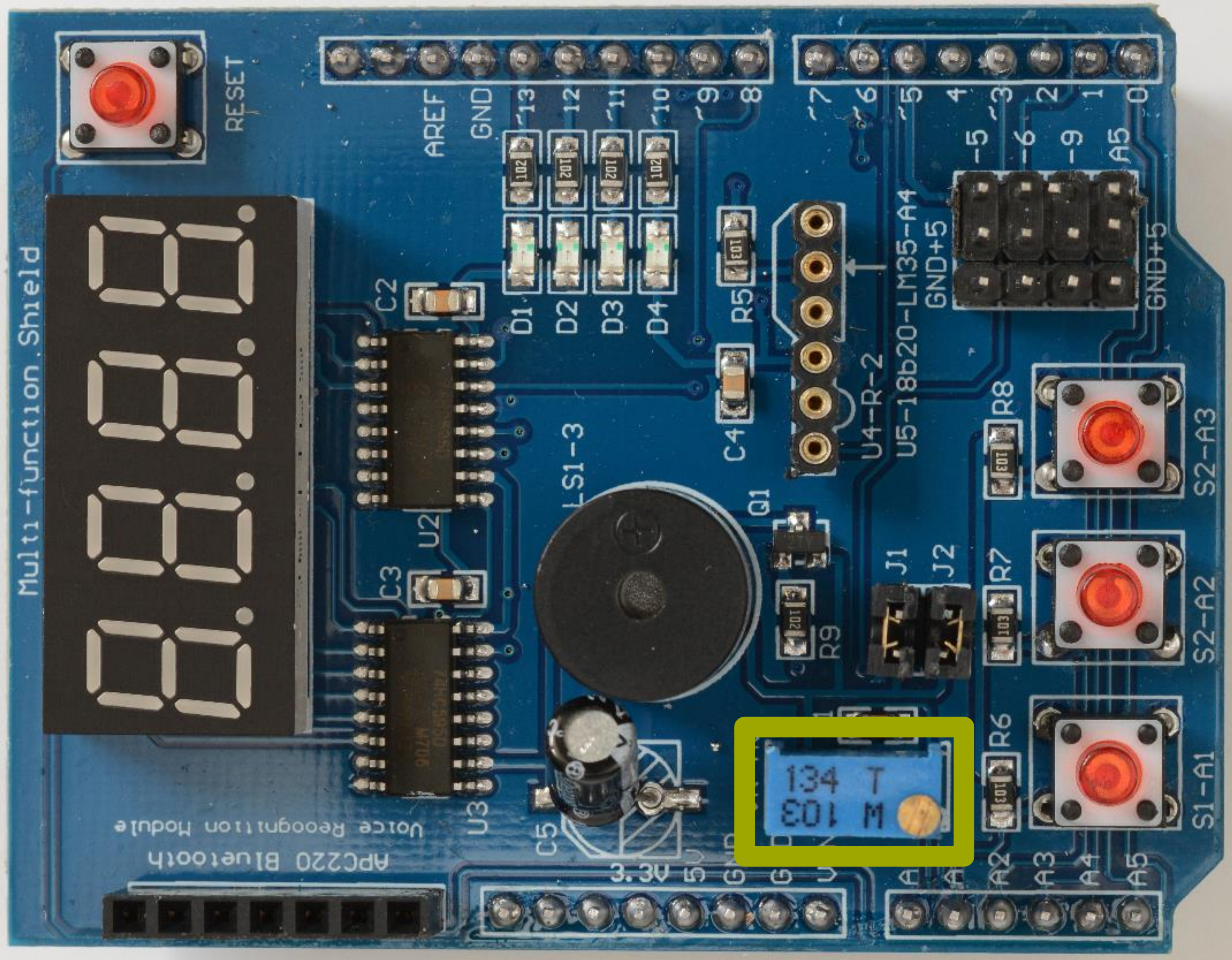
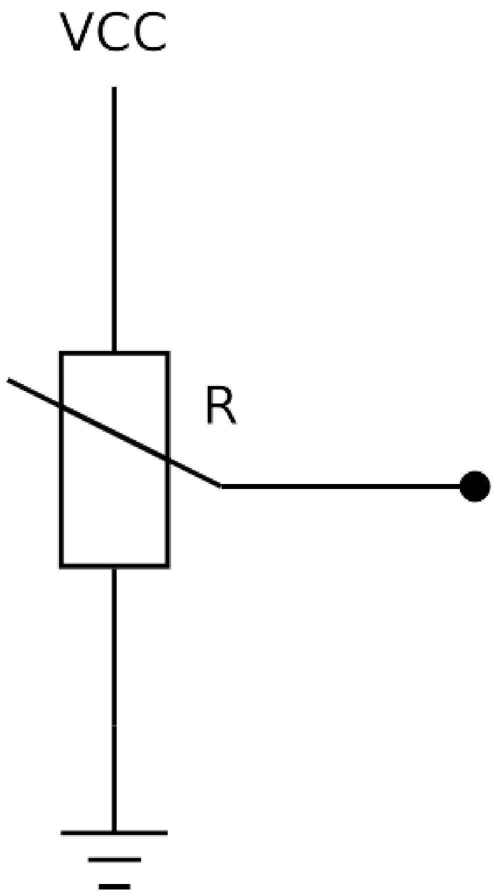
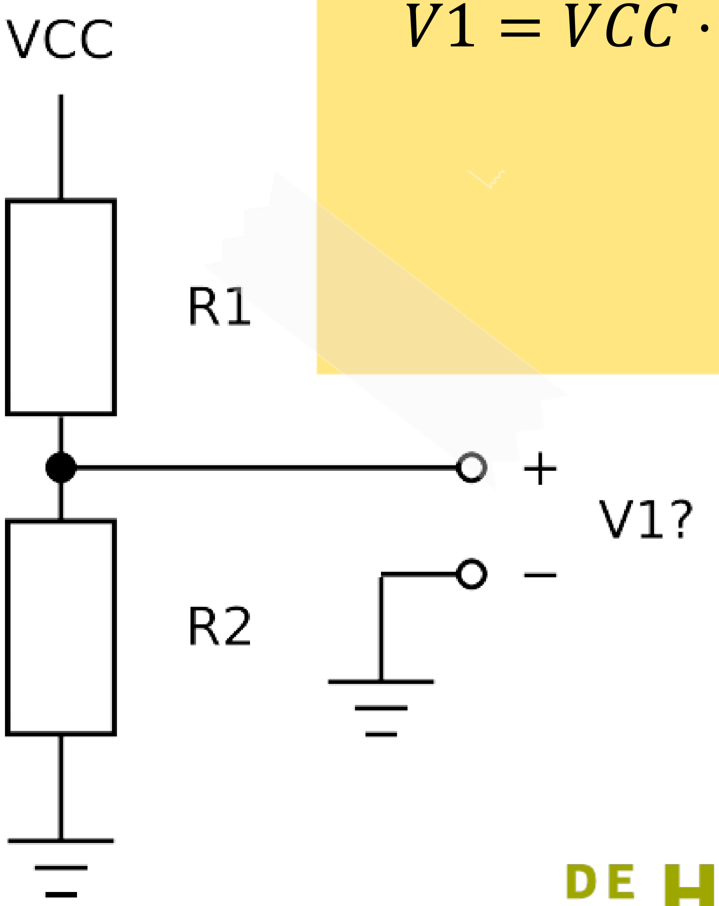


foto: Gerard Tuk

Waarom de ADC?

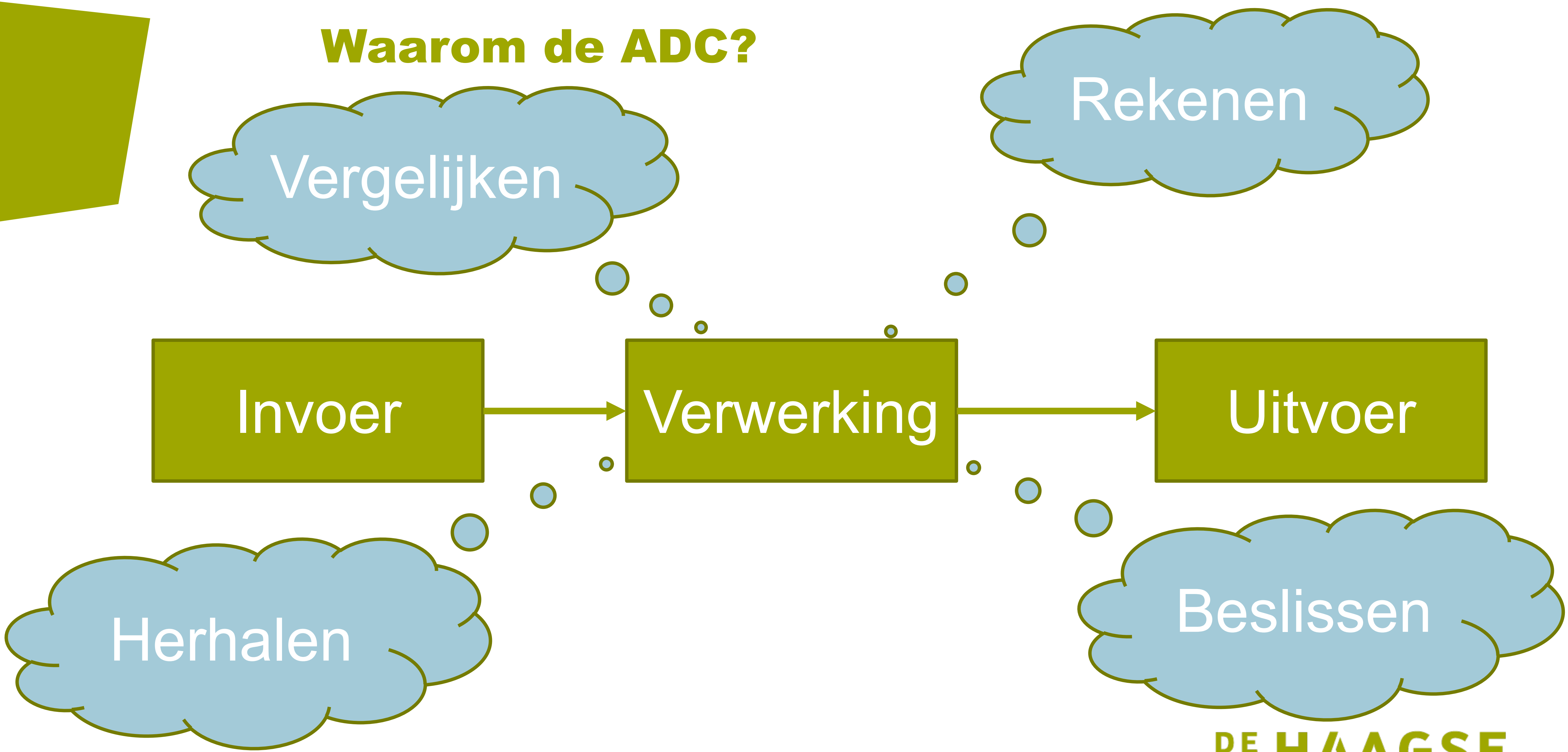
$$V1 = VCC \cdot \frac{R2}{R1 + R2}$$



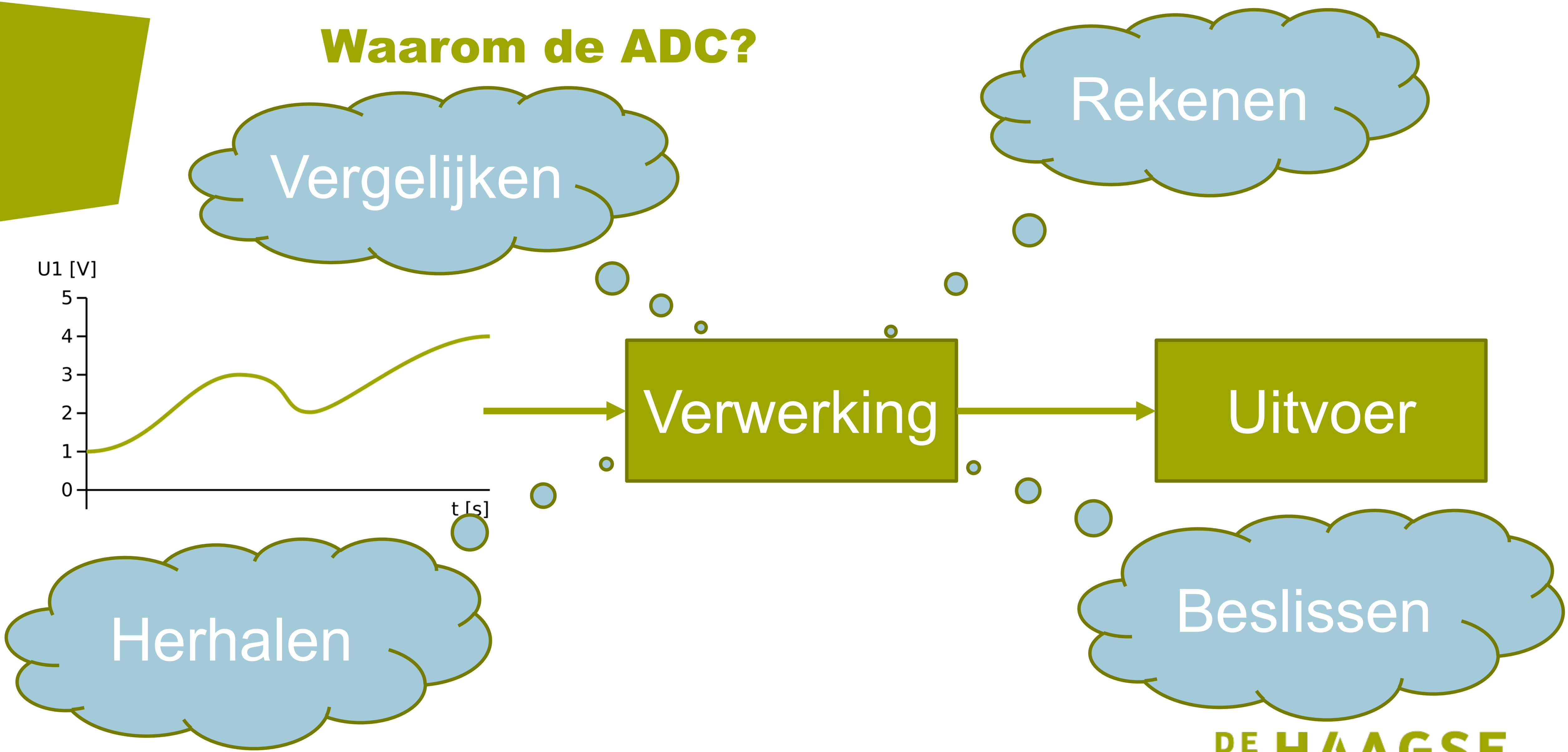


R1 + R2 = R

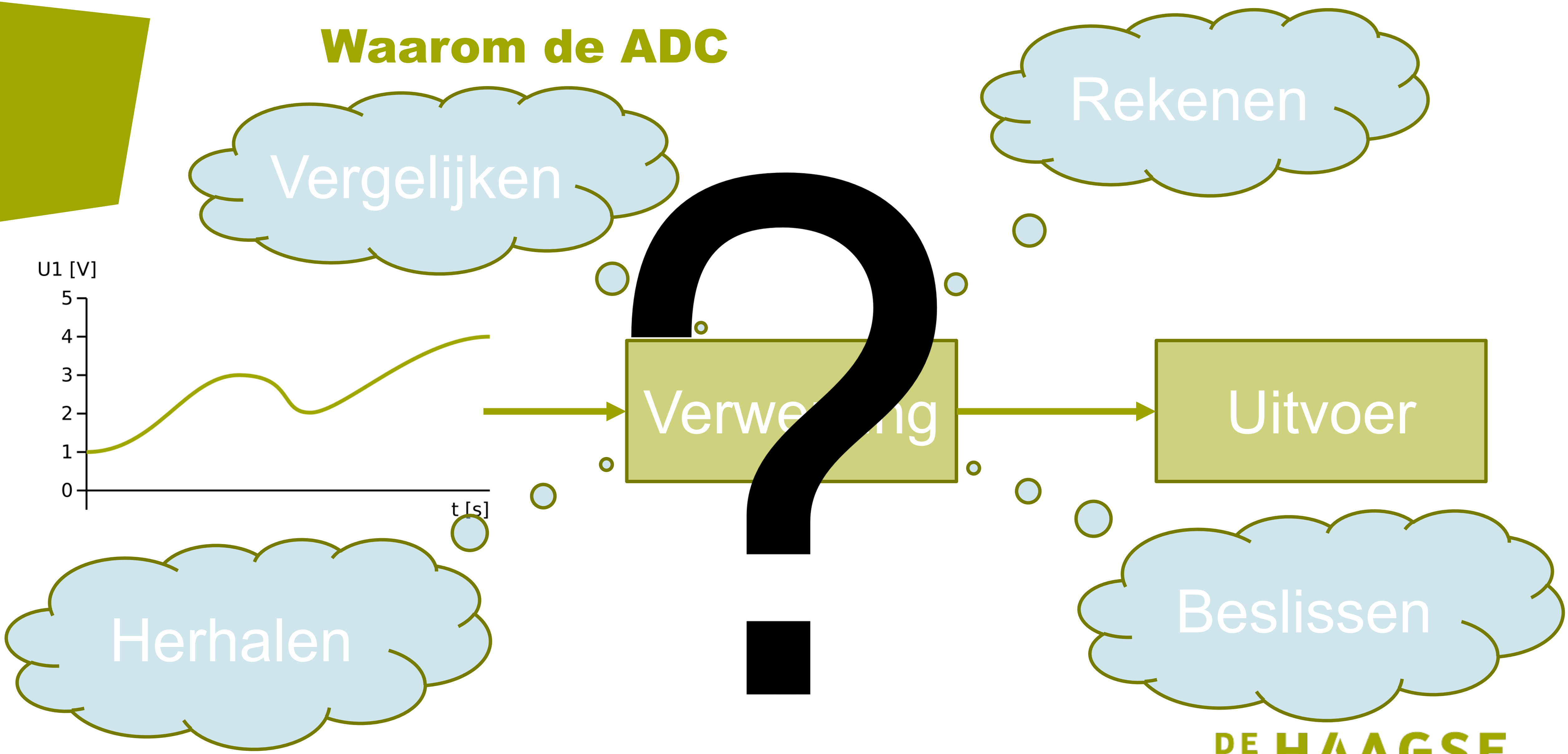
Waarom de ADC?



Waarom de ADC?



Waarom de ADC



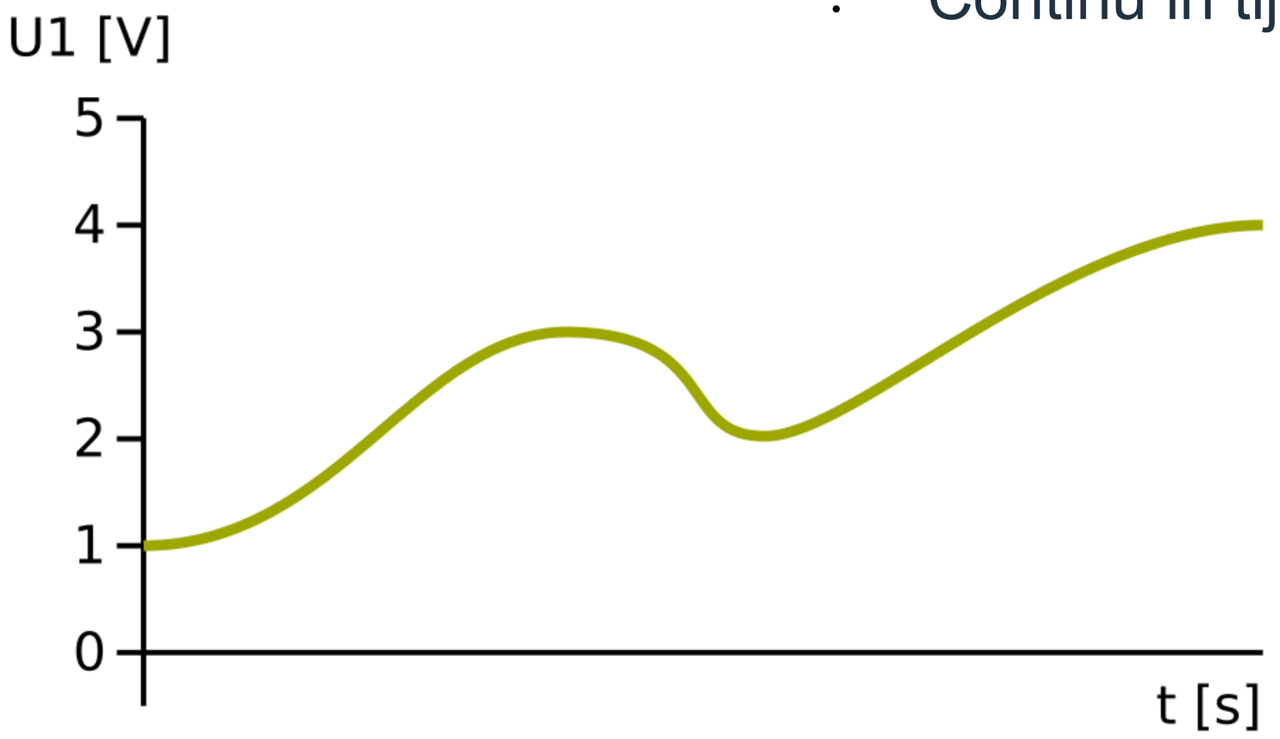
ADC

- Analog-to-digital converter
- Analoog-digitaalomzetter

- Omzetten analoge invoer naar digitale waarde in de AVR

Analoog

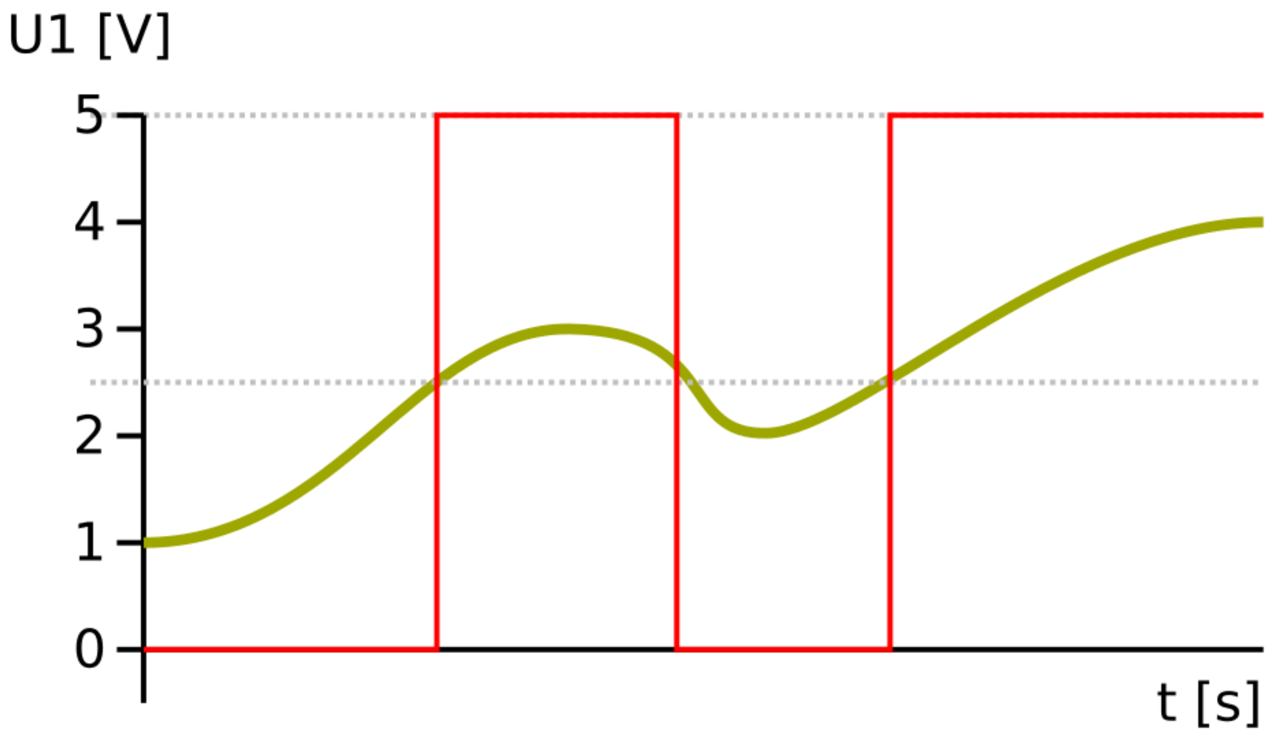
- Continu in amplitude
- Continu in tijd



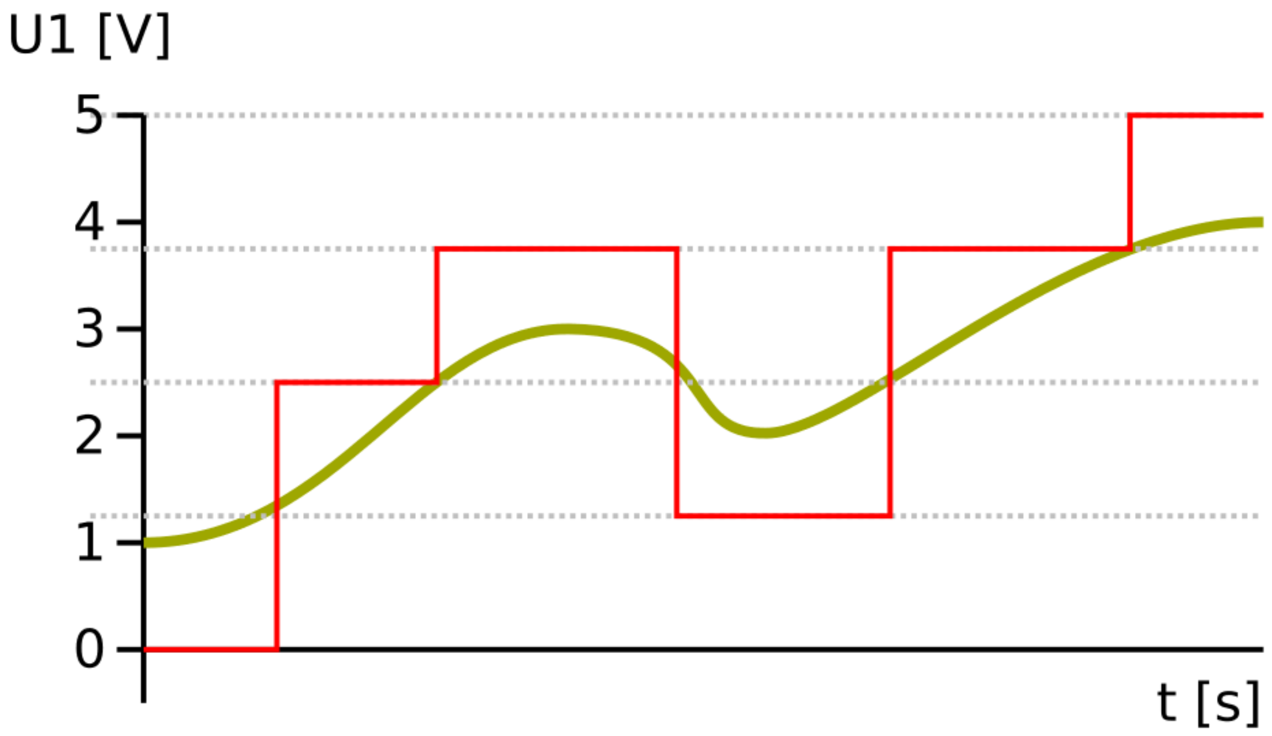
Analoog

- Om te kunnen rekenen in een computer, moeten we de continue amplitude omzetten in een gediscretiseerde waarde
- Kwantisatie (Engels: *quantization*)

Kwantisatie – 1 bit

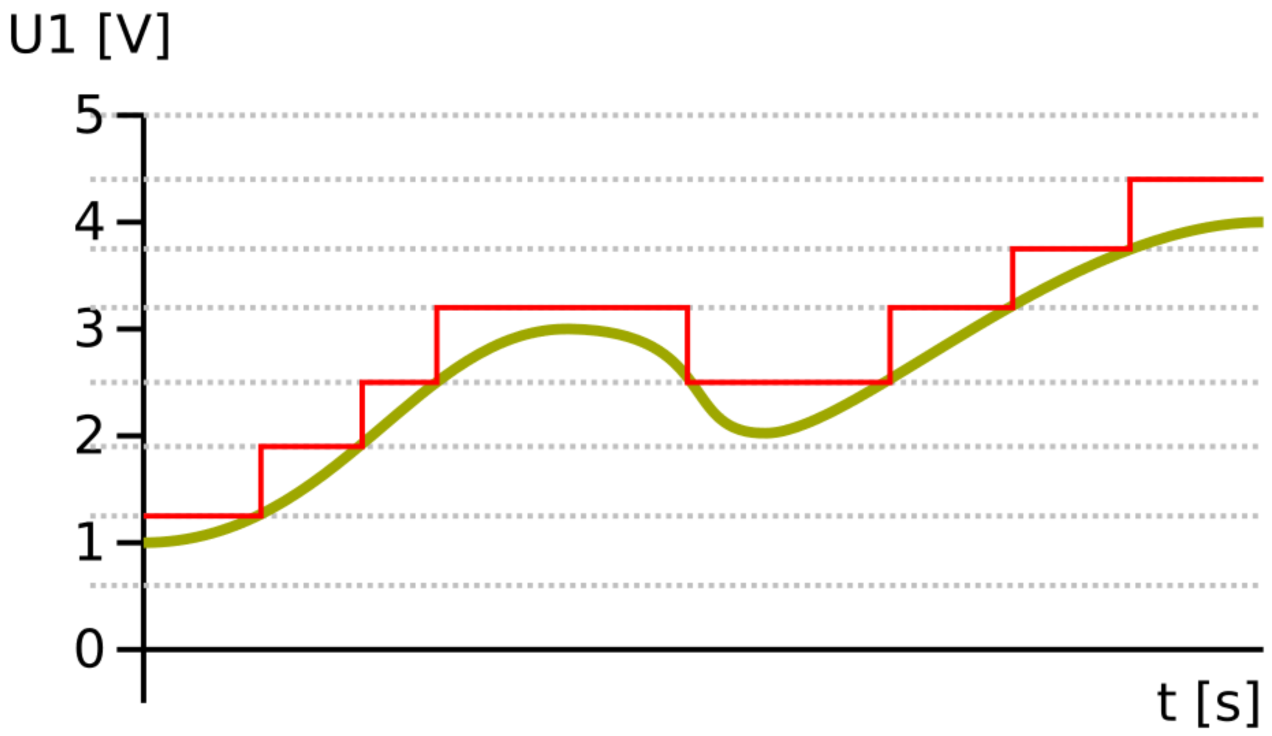


Kwantisatie – 2 bit



DE HAAGSE HOGESCHOOL

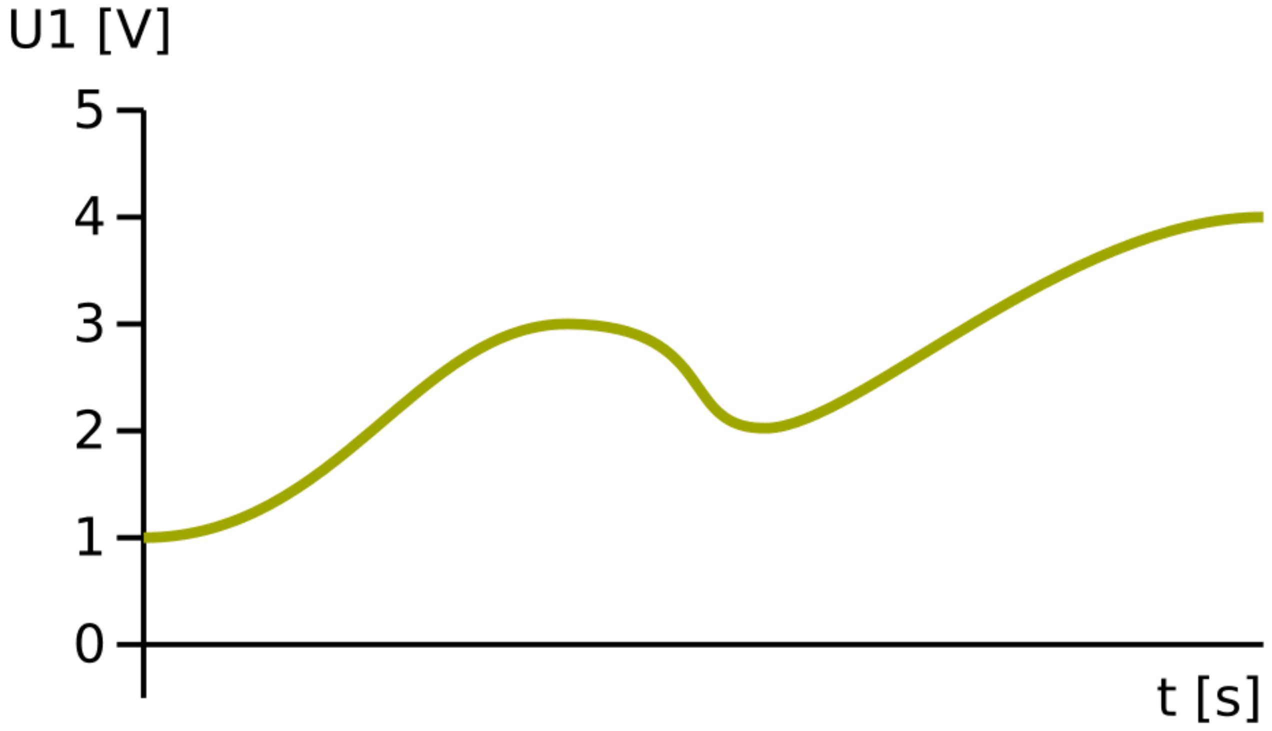
Kwantisatie – 3 bit



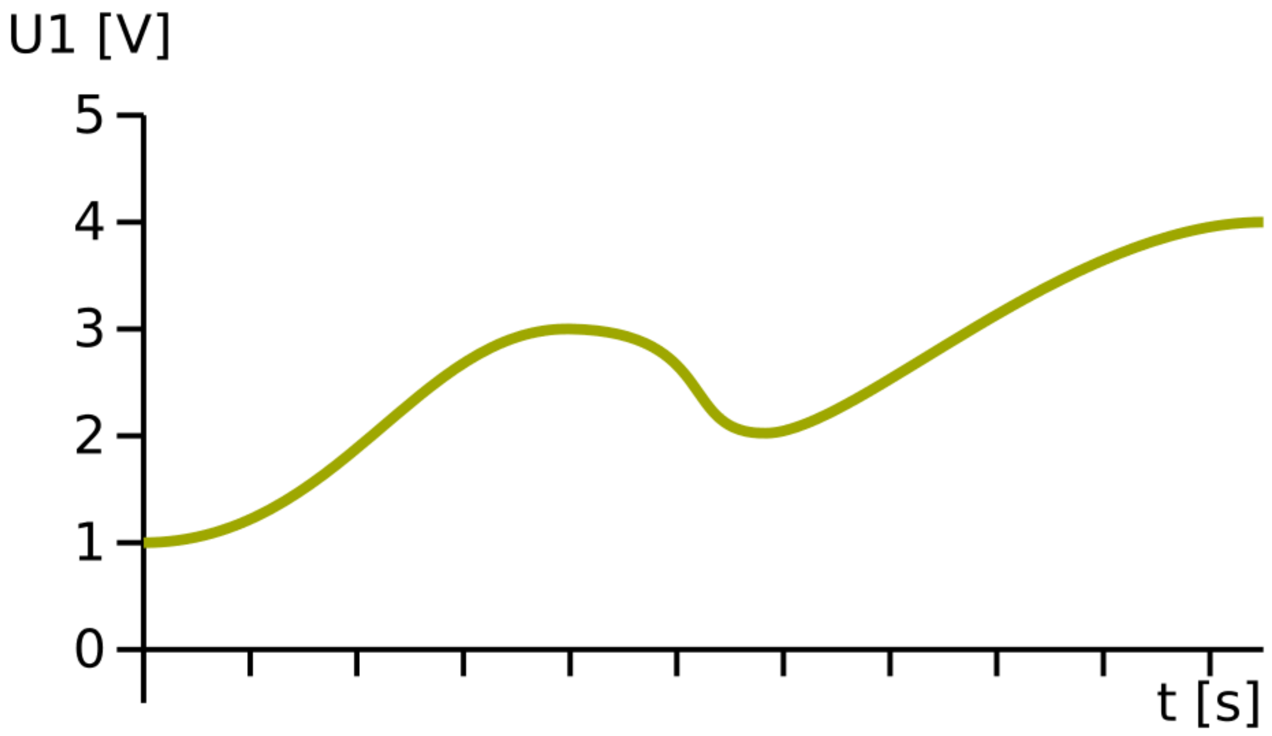
Bemonstering

- We kunnen het signaal ook in de tijd discretiseren
- Bemonstering (Engels: *sampling*)
- Op vaste momenten de waarde van het signaal bepalen

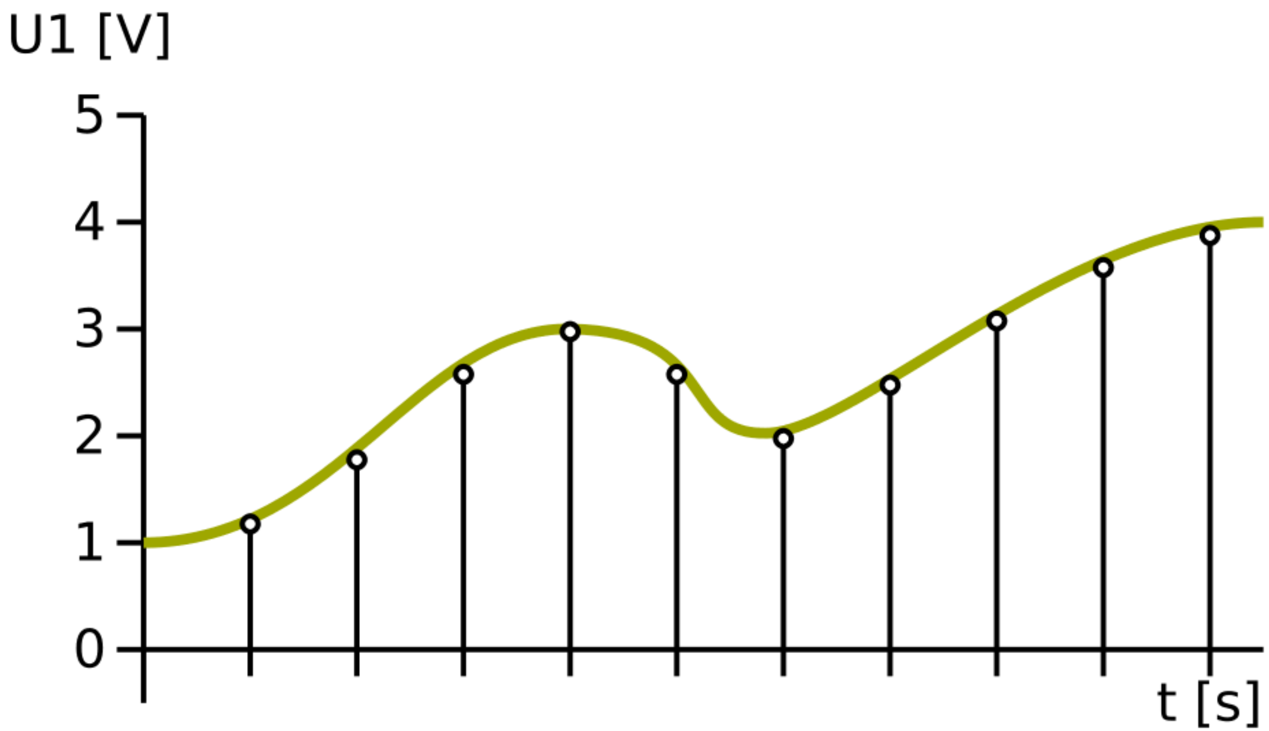
Bemonstering



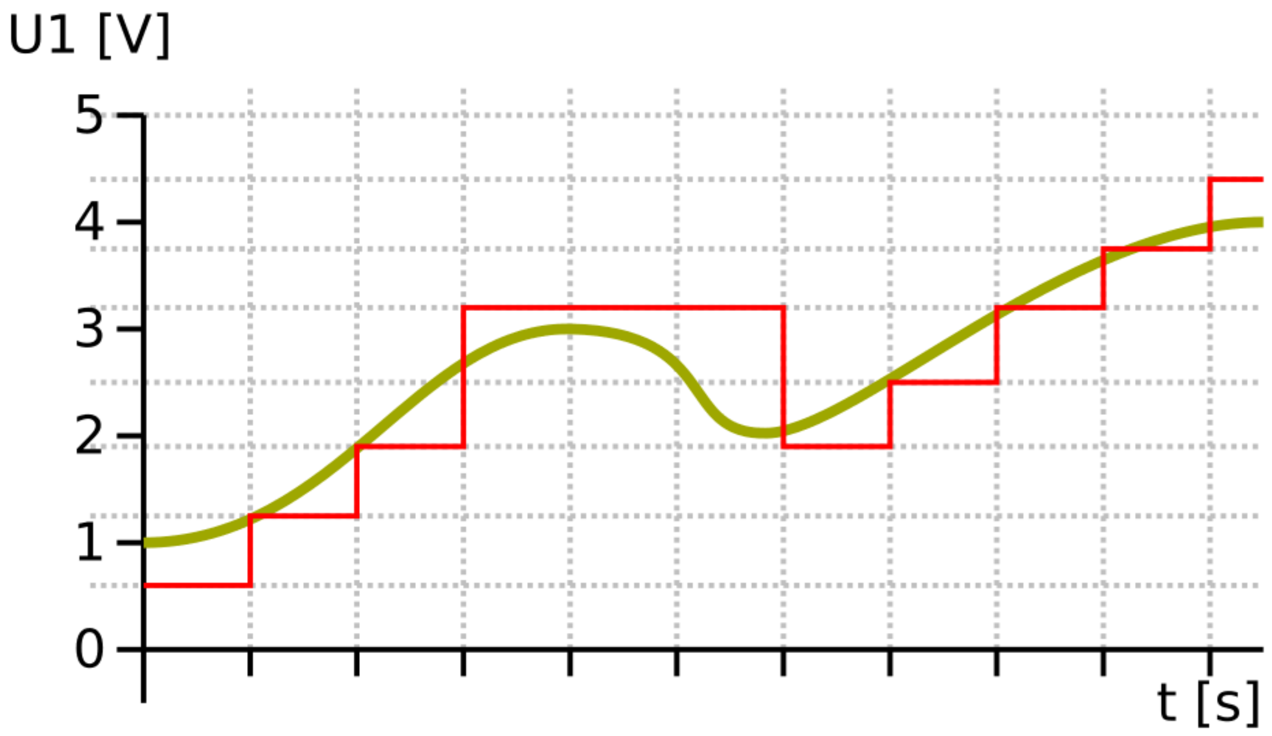
Bemonstering



Bemonstering



Kwantisatie en bemonstering

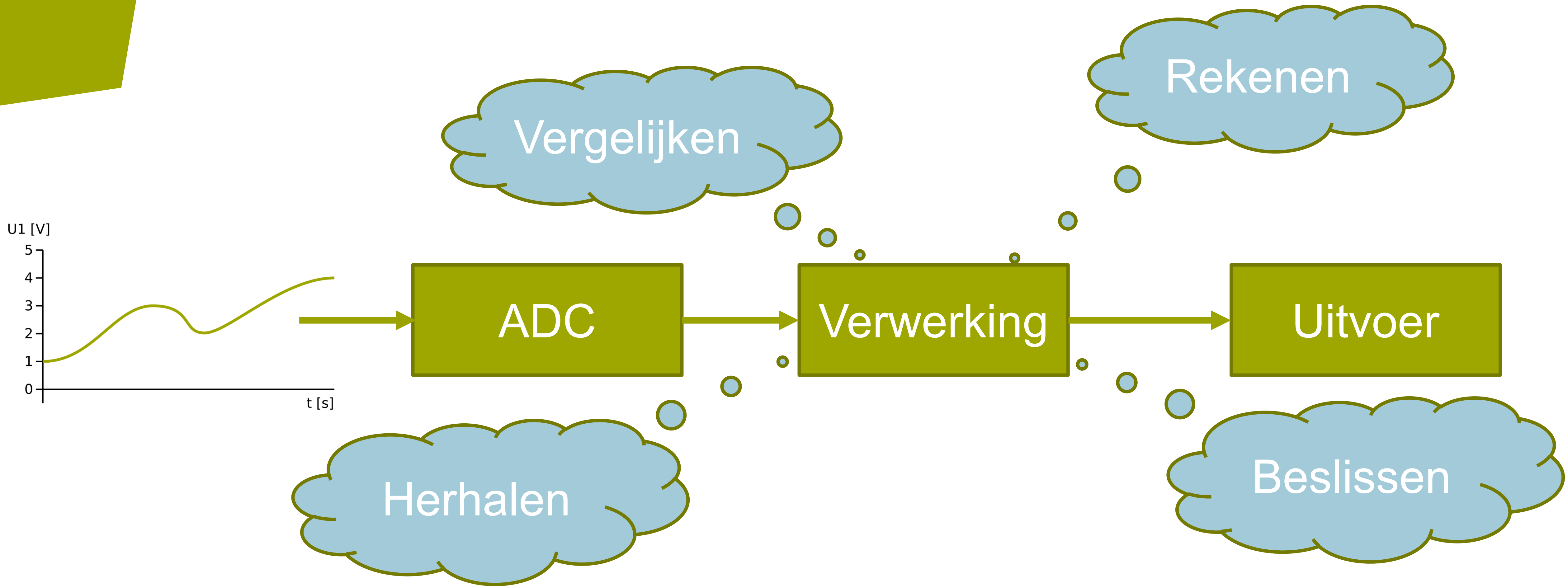


DE HAAGSE HOGESCHOOL

Kwantisatie en bemonstering

- Een tijds- en amplitudediscreet signaal wordt een **digitaal** signaal genoemd
- Een dergelijk signaal kunnen we verwerken in de AVR

Wat is programmeren?



DE HAAGSE HOGESCHOOL

ADC op AVR

- De ADC op de Arduino Uno chip en de Arduino Mega chip verschillen
- We behandelen hier de gezamenlijke eigenschappen
- Gebruik de datasheet voor verdere informatie

ADC op de AVR

26. ADC – Analog to Digital Converter

26.1 Features

- 10-bit Resolution
- 1 LSB Integral Non-linearity
- ±2 LSB Absolute Accuracy
- 13µs - 260µs Conversion Time
- Up to 76.9kSPS (Up to 15kSPS at Maximum Resolution)
- 16 Multiplexed Single Ended Input Channels
- 14 Differential input channels
- 4 Differential Input Channels with Optional Gain of 10× and 200×
- Optional Left Adjustment for ADC Result Readout
- 0V - V_{CC} ADC Input Voltage Range
- 2.7V - V_{CC} Differential ADC Voltage Range
- Selectable 2.56V or 1.1V ADC Reference Voltage
- Free Running or Single Conversion Mode
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Canceler

26. ADC – Analog to Digital Converter

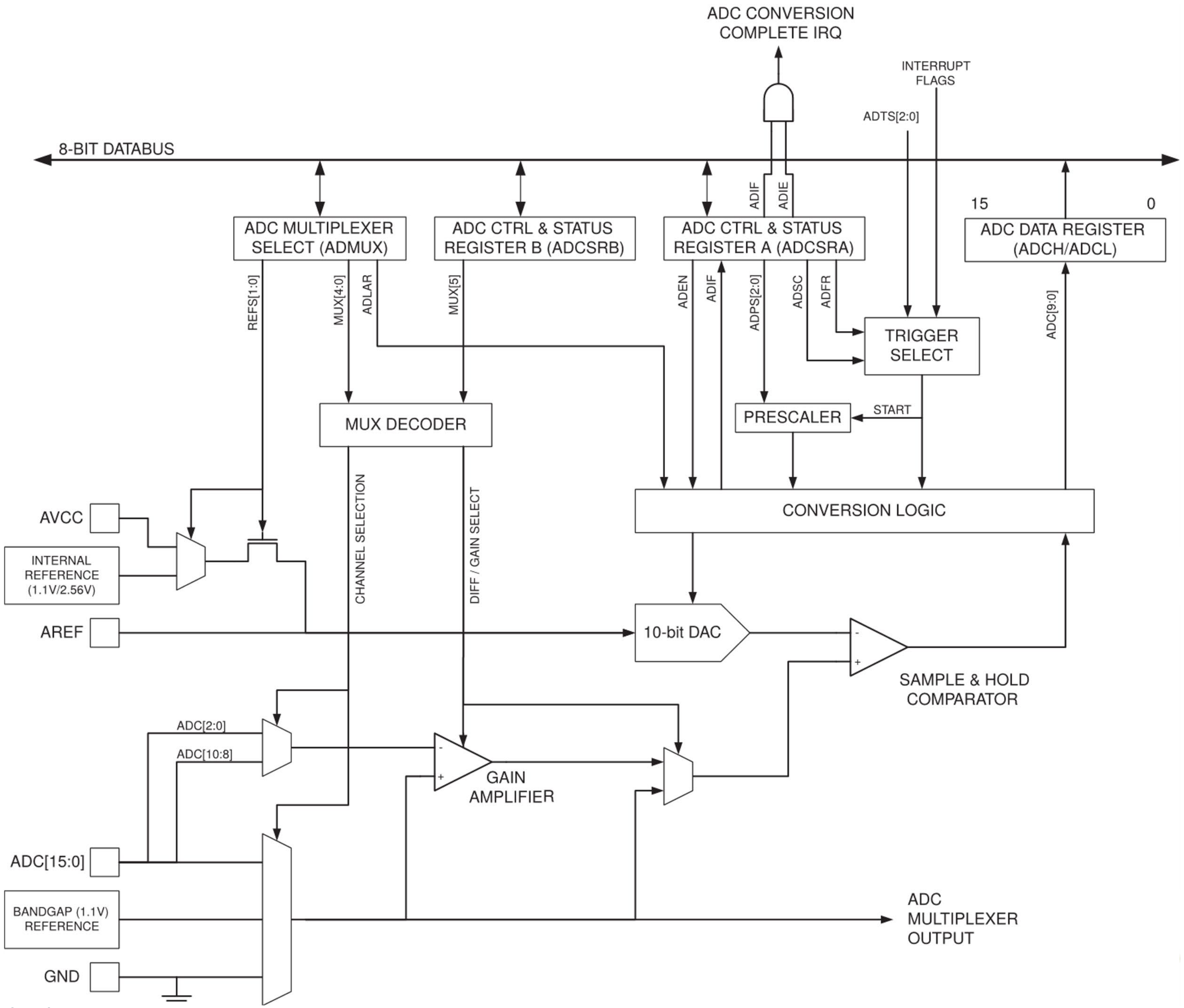
26.1 Features

Specificatie kwantisatie

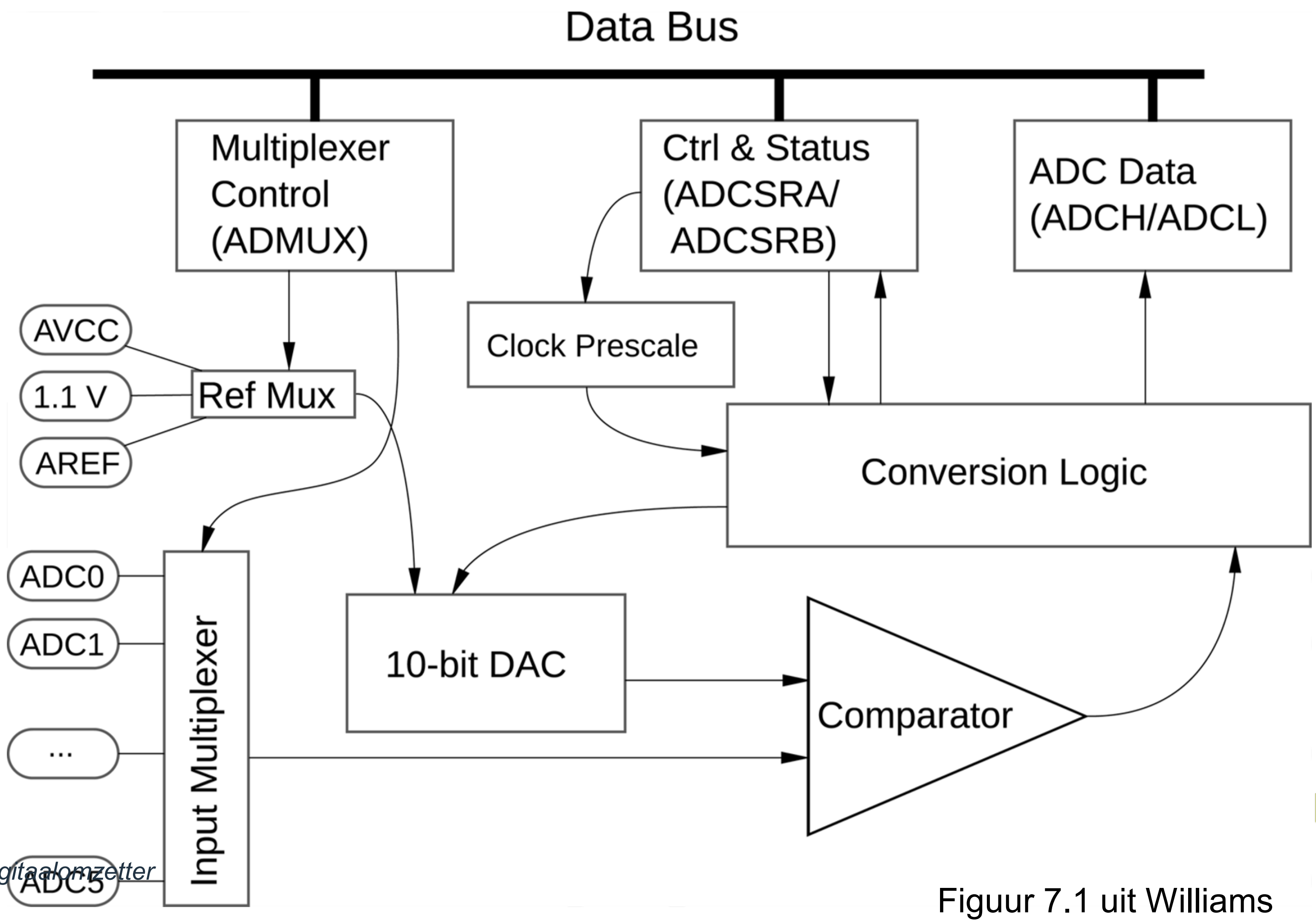
- 10-bit Resolution
- 1 LSB Integral Non-linearity
- ±2 LSB Absolute Accuracy
- 13µs - 260µs Conversion Time
- Up to 76.9kSPS (Up to 15kSPS at Maximum Resolution)
- 16 Multiplexed Single Ended Input Channels
- 14 Differential input channels
- 4 Differential Input Channels with Optional Gain of 10× and 200×
- Optional Left Adjustment for ADC Result Readout
- 0V - V_{CC} ADC Input Voltage Range
- 2.7V - V_{CC} Differential ADC Voltage Range
- Selectable 2.56V or 1.1V ADC Reference Voltage
- Free Running or Single Conversion Mode
- Interrupt on ADC Conversion Complete
- Sleep Mode Noise Canceler

Specificatie bemonstering

DE HAAGSE HOGESCHOOL


ADC CONVERSION COMPLETE IRQ
INTERRUPT FLAGS
ADTS[2:0]
8-BIT DATABUS
ADIF
ADIE
15
0
ADC MULTIPLEXER SELECT (ADMUX)
ADC CTRL & STATUS REGISTER B (ADCSRB)
ADC CTRL & STATUS REGISTER A (ADCSRA)
ADC DATA REGISTER (ADCH/ADCL)
REFS[1:0]
MUX[4:0]
ADLAR
MUX[5]
ADEN
ADIF
ADPS[2:0]
ADSC
ADFR
ADC[9:0]
TRIGGER SELECT
MUX DECODER
PRESCALER
START
CONVERSION LOGIC
CHANNEL SELECTION
DIFF / GAIN SELECT
AVCC
INTERNAL REFERENCE (1.1V/2.56V)
AREF
10-bit DAC
SAMPLE & HOLD COMPARATOR
ADC[2:0]
ADC[10:8]
GAIN AMPLIFIER
ADC[15:0]
BANDGAP (1.1V) REFERENCE
GND
ADC MULTIPLEXER OUTPUT

ADC blokschema

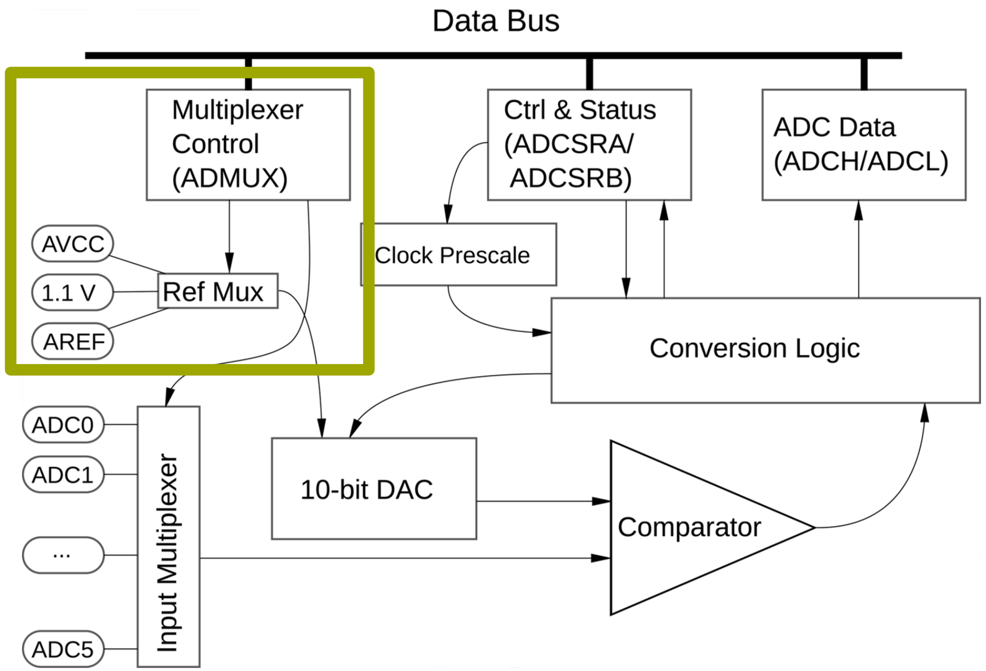


Figuur 7.1 uit Williams

ADC instellen

Instellen:

- ADC referentiespanning

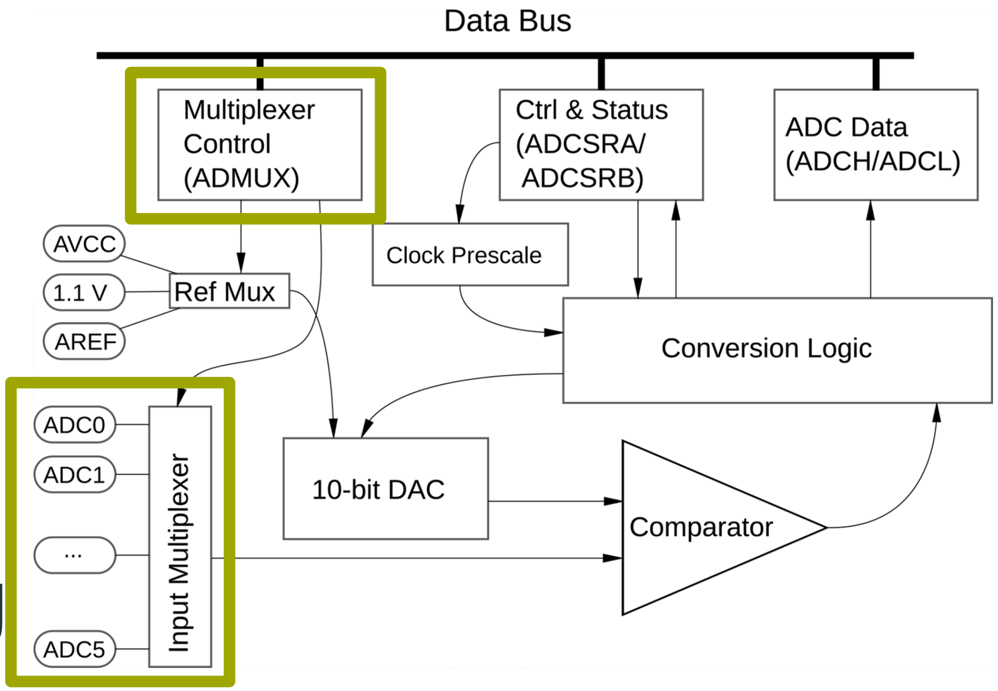


DE HAAGSE HOGESCHOOL

ADC instellen

Instellen:

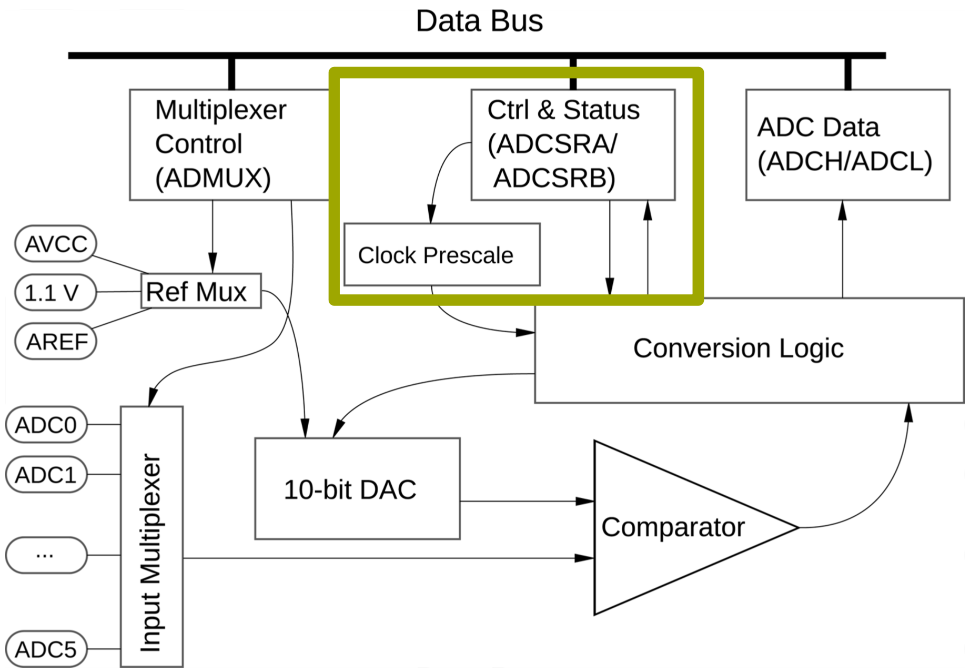
- ADC referentiespanning
- Analoog kanaal



ADC instellen

Instellen:

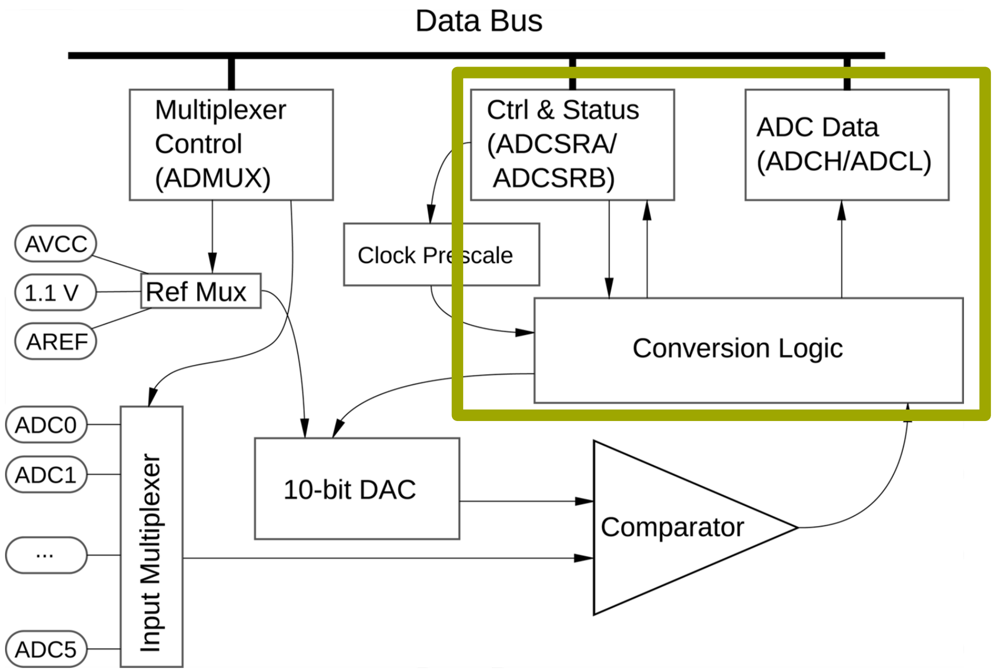
- ADC referentiespanning
- Analoog kanaal
- Klokdeler



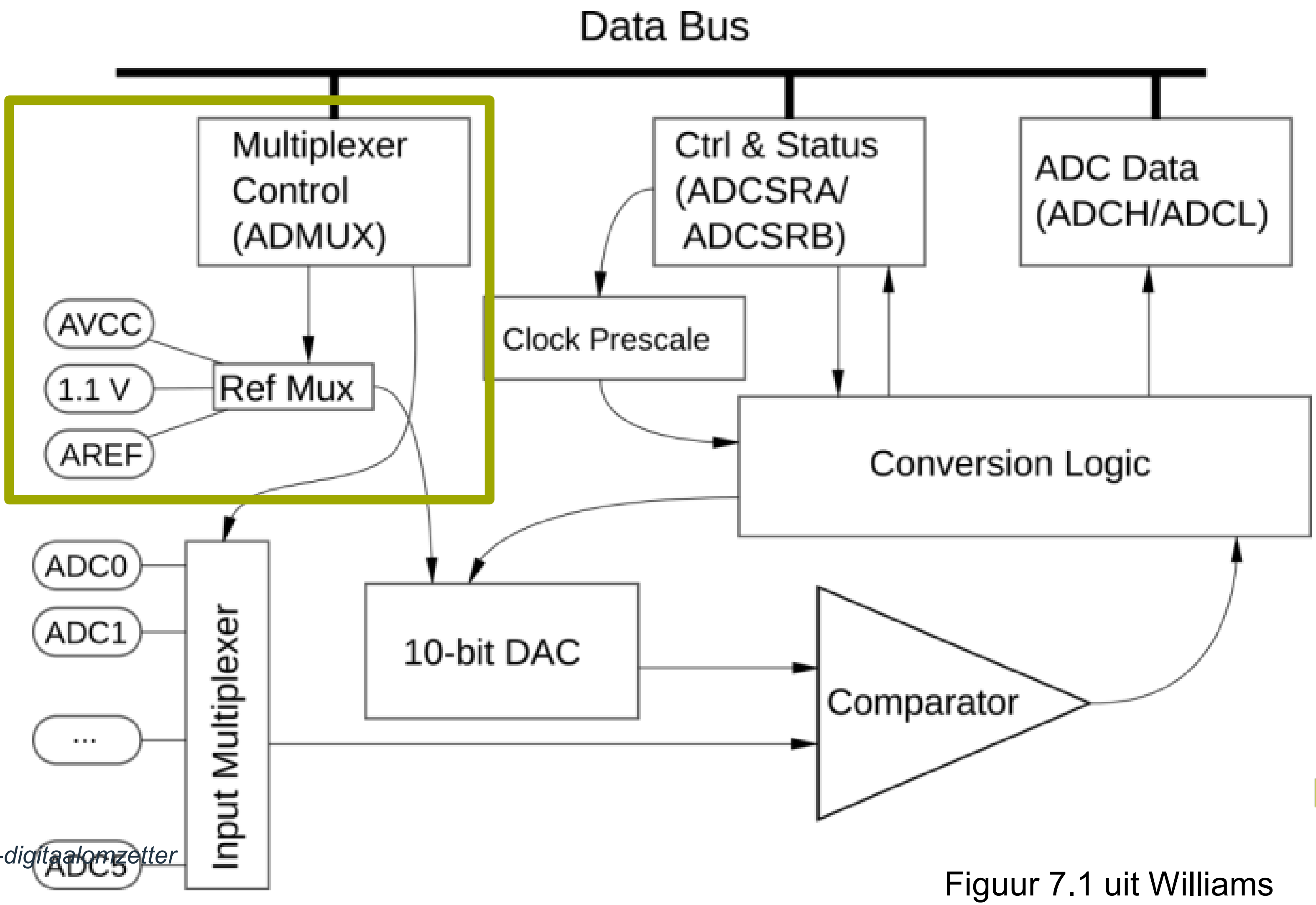
ADC instellen

Instellen:

- ADC referentiespanning
- Analoog kanaal
- Klokdeler
- Activeren ADC



ADC referentiespanning



Figuur 7.1 uit Williams

ADC referentiespanning

"After the conversion is complete (ADIF is high), the conversion result can be found in the ADC Result Registers (ADCL, ADCH).

For single ended conversion, the result is

$$\boxed{ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}}$$

where V_{IN} is the voltage on the selected input pin and $\boxed{V_{REF} \text{ the selected voltage reference}}$ (see Table 26-3 on page 281 and Table 26-4 on page 282). 0x000 represents analog ground, and 0x3FF represents the selected reference voltage minus one LSB."

DE HAAGSE HOGESCHOOL

Instellen referentiespanning

Bit	7	6	5	4	3	2	1	0	
(0x7C)	**REFS1**	**REFS0**	**ADLAR**	**MUX4**	**MUX3**	**MUX2**	**MUX1**	**MUX0**	**ADMUX**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

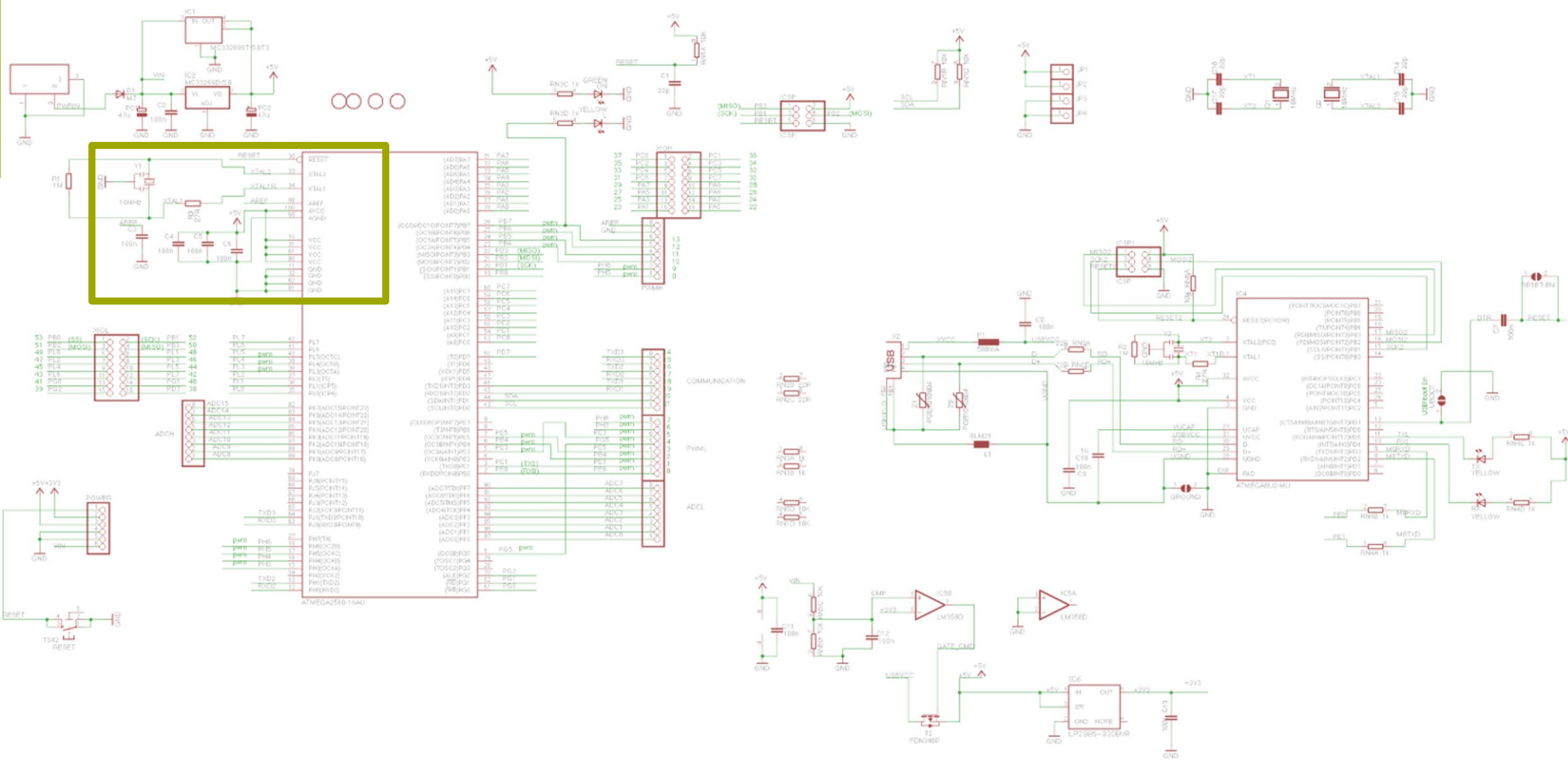
Table 26-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection[1]
0	0	AREF, Internal V_{REF} turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Internal 1.1V Voltage Reference with external capacitor at AREF pin
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

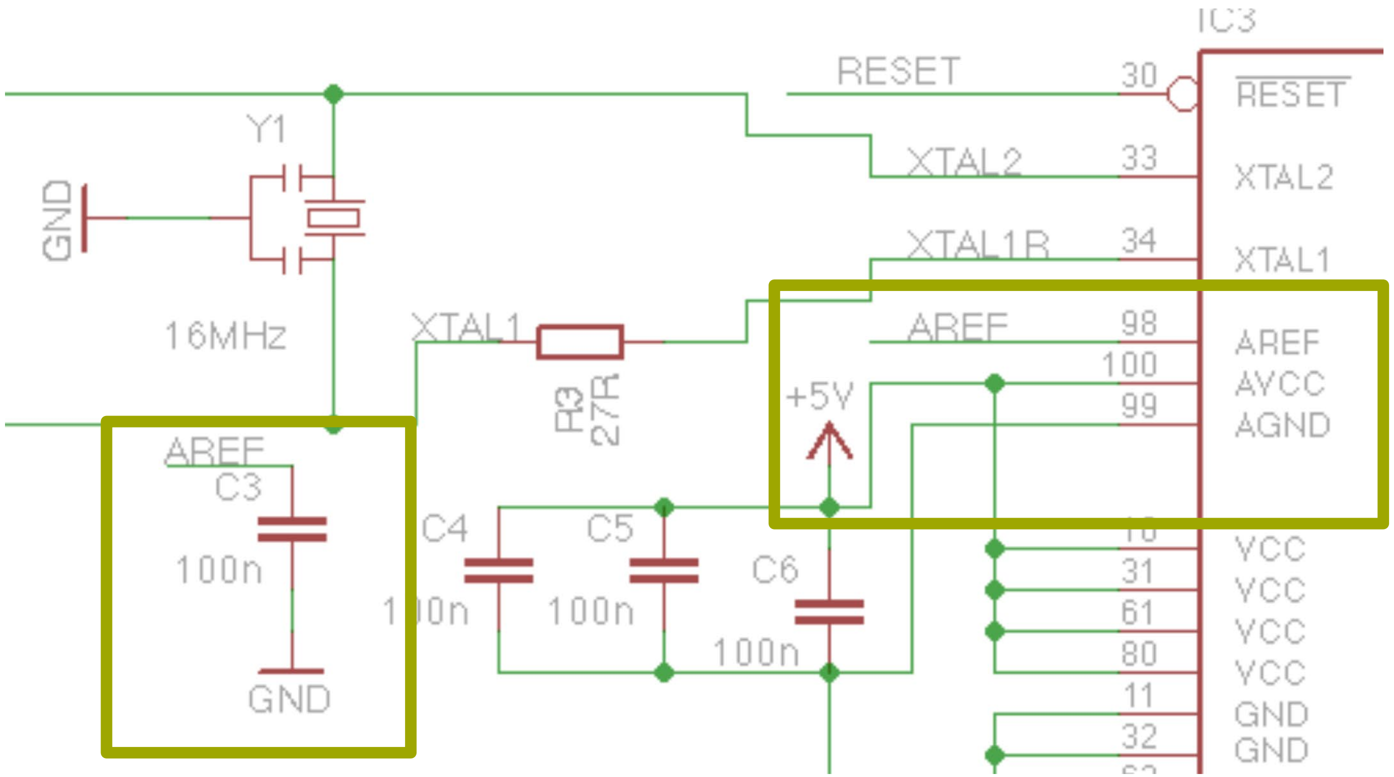
Arduino™ Mega 2560 Reference Design

Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS. Arduino DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE

Arduino may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Arduino reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.



ADC referentiespanning



Instellen referentiespanning

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 26-3. Voltage Reference Selections for ADC

REFS1	REFS0	Voltage Reference Selection[1]
0	0	AREF, Internal V_{REF} turned off
0	1	AVCC with external capacitor at AREF pin
1	0	Internal 1.1V Voltage Reference with external capacitor at AREF pin
1	1	Internal 2.56V Voltage Reference with external capacitor at AREF pin

DE HAAGSE HOGESCHOOL

ADC referentiespanning

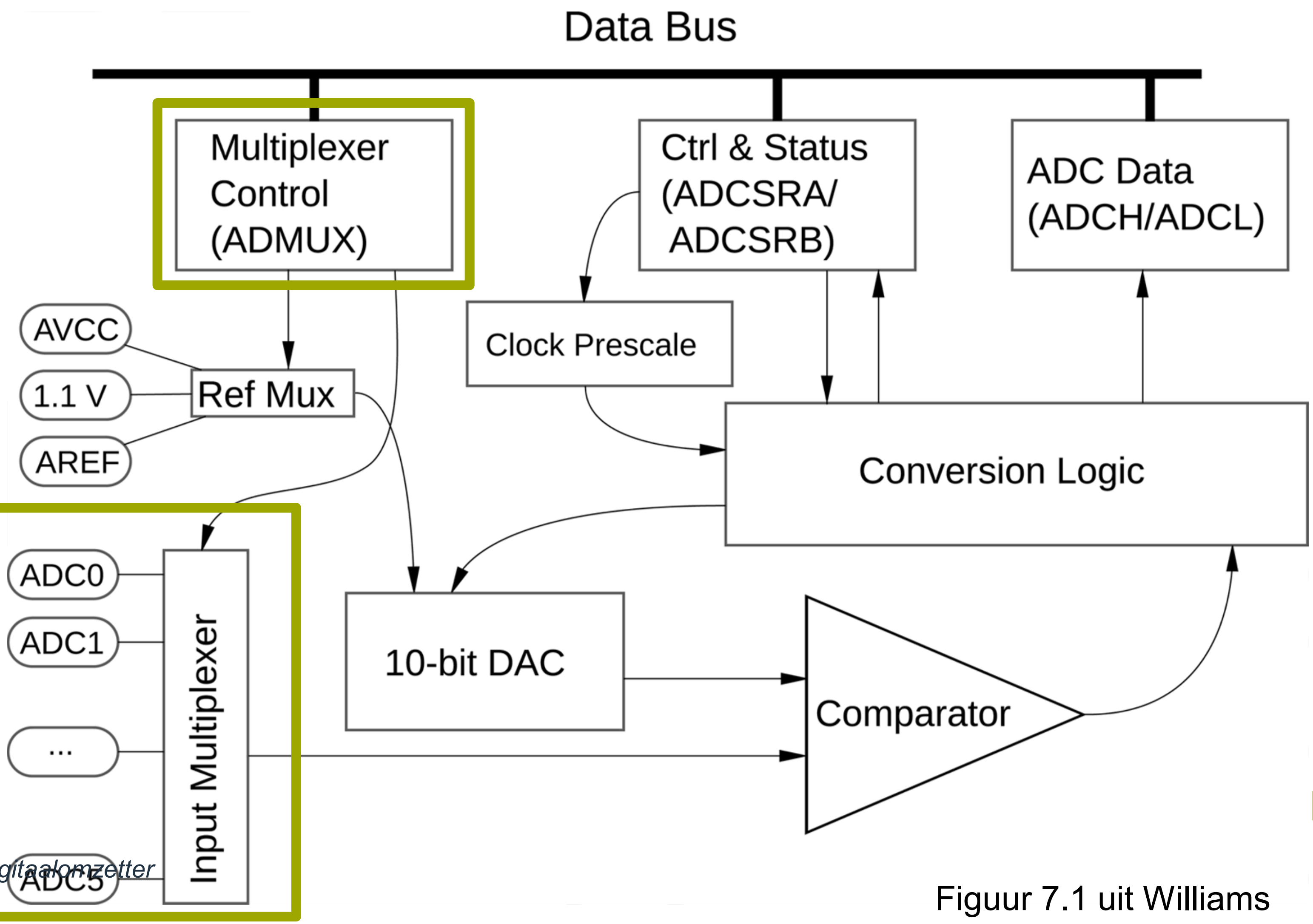
```
void init_adc (void)
{
	ADMUX = (0 << REFS1) | (1 << REFS0);
}
```

ADC instellen

Instellen:

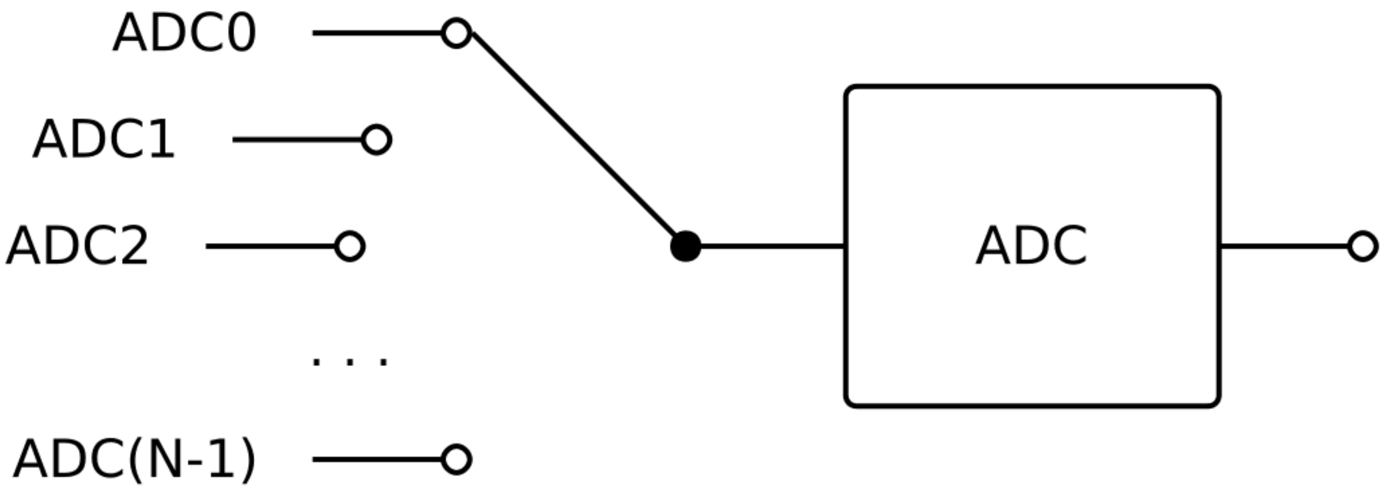
- ADC referentiespanning
- Analoog kanaal
- Klokdeler
- Activeren ADC

ADC analoog kanaal



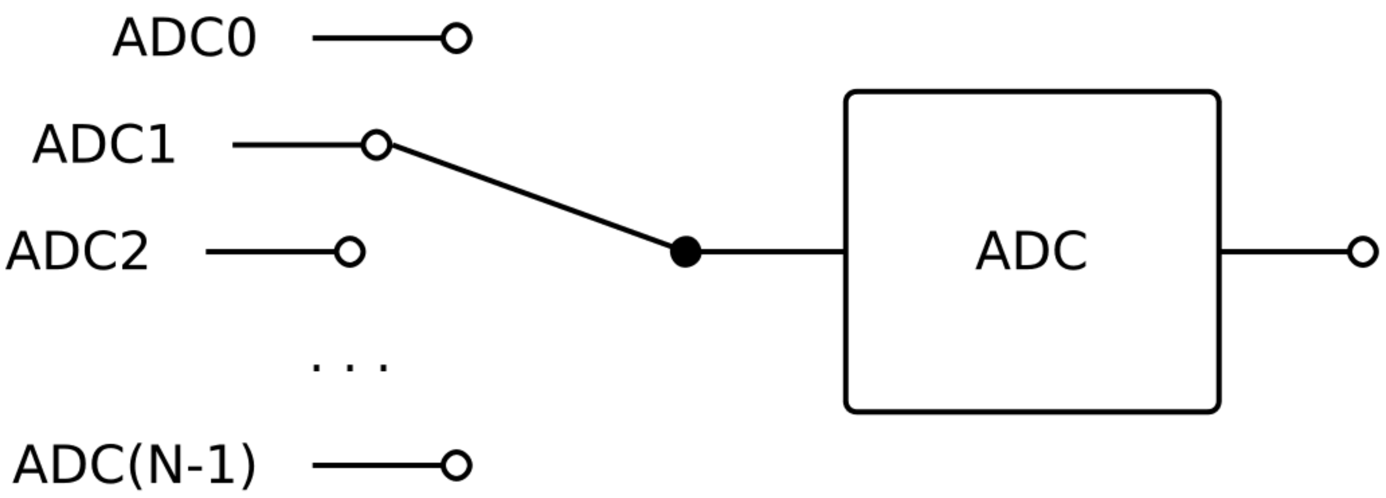
Figuur 7.1 uit Williams

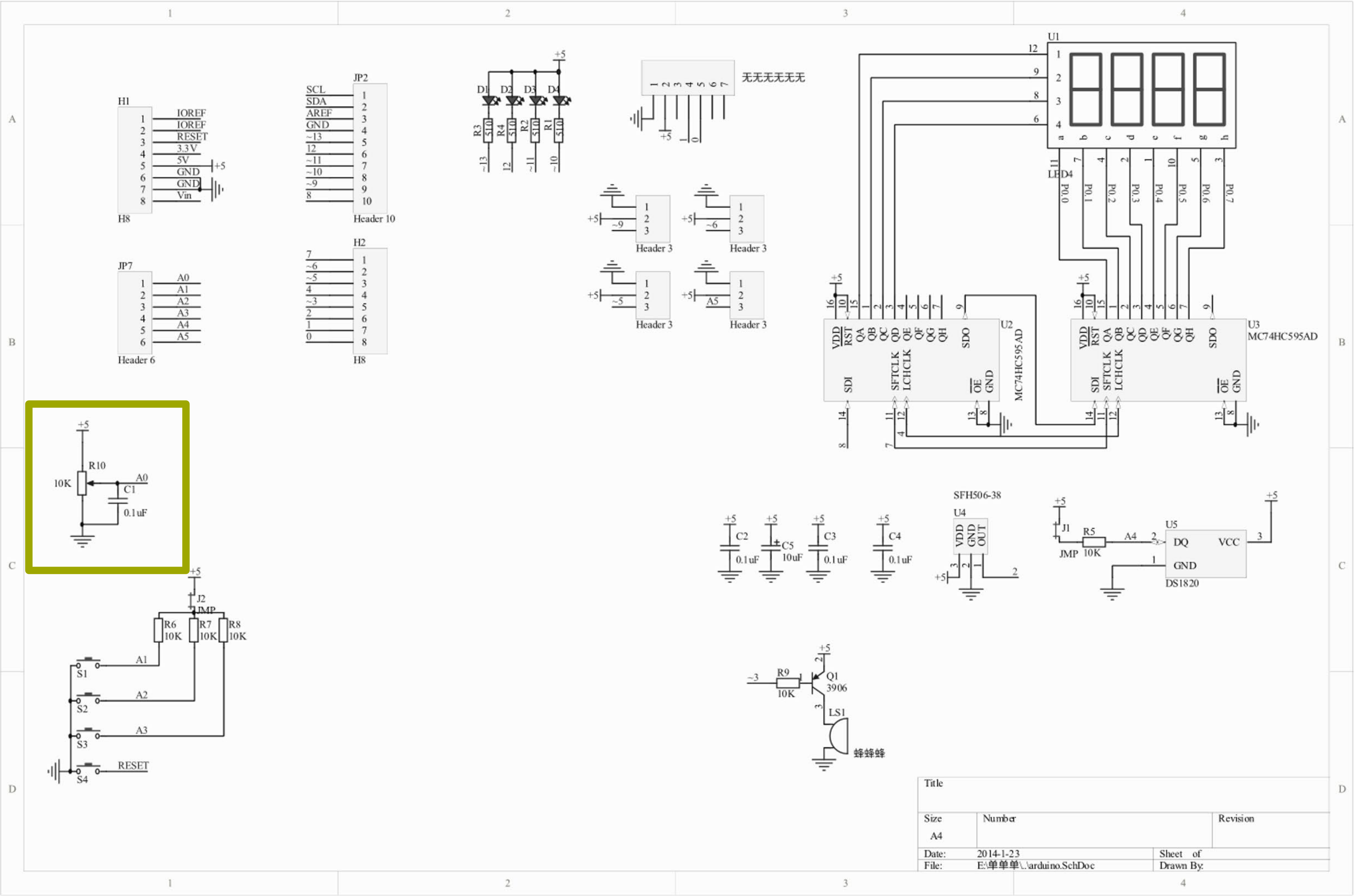
ADC analoog kanaal



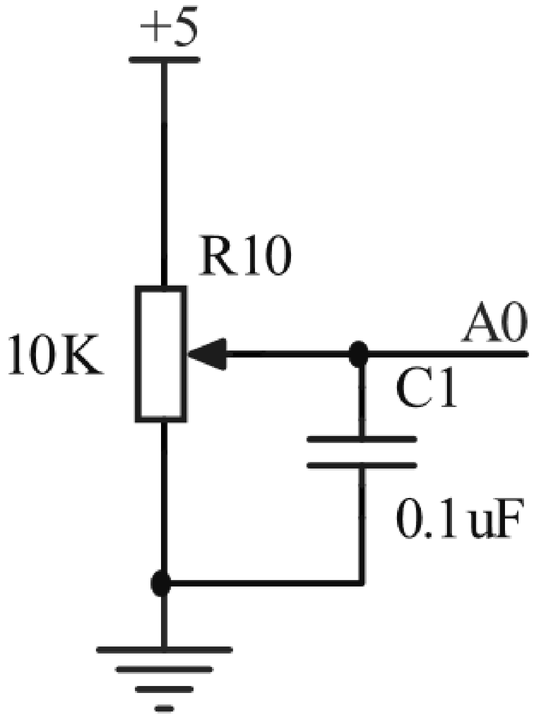
DE HAAGSE HOGESCHOOL

ADC analoog kanaal




H1
IOREF
IOREF
RESET
3.3V
5V
GND
GND
Vin
H8
JP2
SCL
SDA
AREF
GND
~13
12
~11
~10
~9
8
Header 10
JP7
A0
A1
A2
A3
A4
A5
Header 6
H2
7
~6
~5
4
~3
2
1
0
H8
D1
D2
D3
D4
R3
R4
R2
R1
510
+5
无无无无无无
Header 3
U1
LED4
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
U2
MC74HC595AD
U3
MC74HC595AD
VDD
RST
QA
QB
QC
QD
QE
QF
QG
QH
SDO
SDI
SFTCLK
LCHCLK
OE
GND
R10
10K
A0
C1
0.1uF
J2
JMP
R6
R7
R8
10K
A1
A2
A3
RESET
S1
S2
S3
S4
C2
C5
C3
C4
0.1uF
10uF
SFH506-38
U4
VDD
GND
OUT
J1
JMP
R5
10K
A4
U5
DQ
VCC
GND
DS1820
R9
10K
~3
Q1
3906
LS1
蜂蜂蜂
Title
Size
A4
Number
Revision
Date:
2014-1-23
File:
E:\单单单\..\arduino.SchDoc
Sheet of
Drawn By:

ADC analoog kanaal



A0 zit aan PF0/ADC0 (Mega)

ADC analoog kanaal

Bit	7	6	5	4	3	2	1	0	
(0x7C)	**REFS1**	**REFS0**	**ADLAR**	**MUX4**	**MUX3**	**MUX2**	**MUX1**	**MUX0**	**ADMUX**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0x7B)	–	**ACME**	–	–	**MUX5**	**ADTS2**	**ADTS1**	**ADTS0**	**ADCSRB**
Read/Write	R	R/W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 26-4. Input Channel Selections

MUX5:0	Single Ended Input	Positive Differential Input	Negative Differential Input	Gain
000000	ADC0	N/A		
000001	ADC1			
000010	ADC2			
000011	ADC3			
000100	ADC4			
000101	ADC5			
000110	ADC6			
000111	ADC7			

ADC analoog kanaal

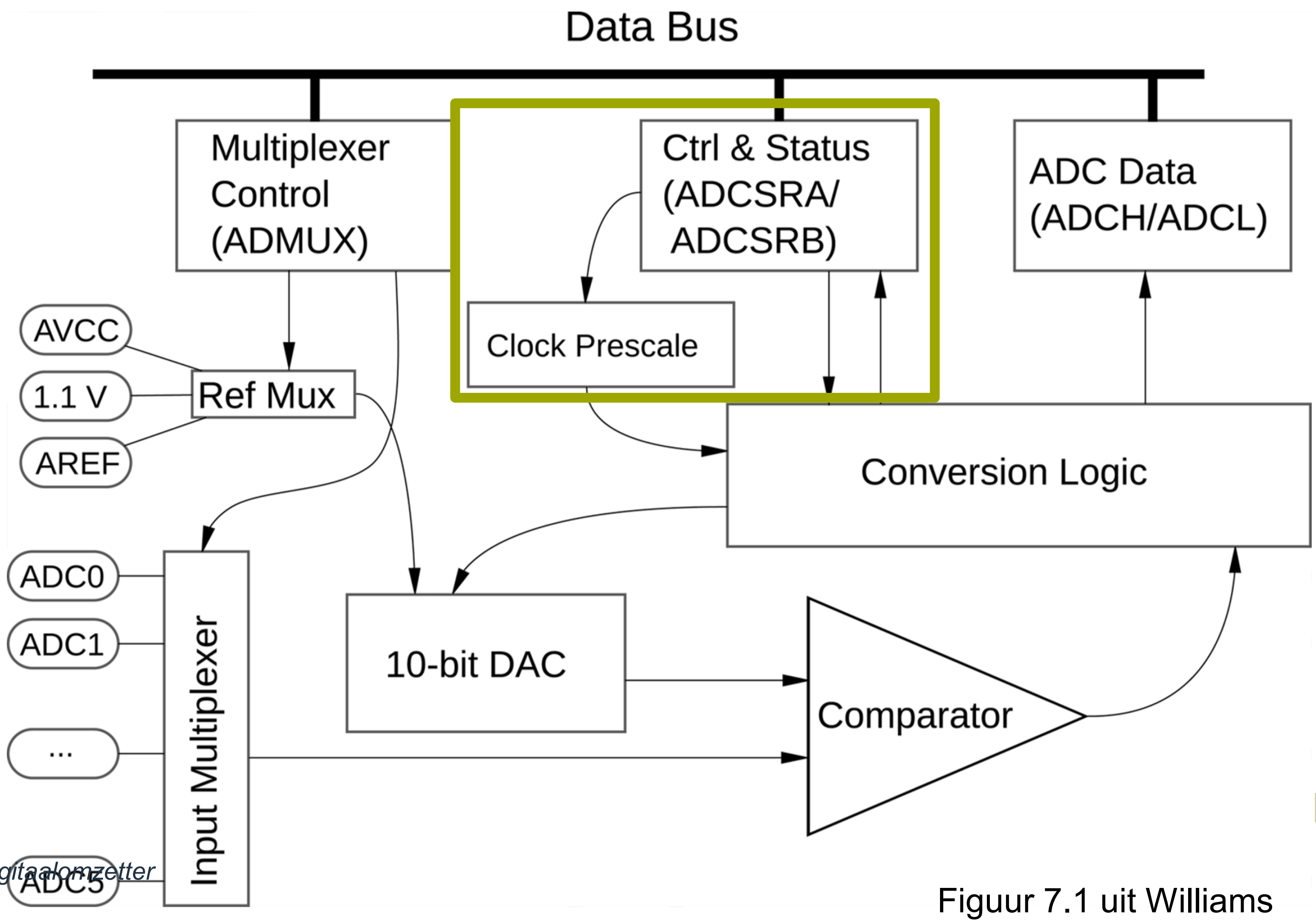
```
void init_adc (void)
{
	ADMUX = (0 << REFS1) | (1 << REFS0);
}
```

ADC instellen

Instellen:

- ADC referentiespanning
- Analoog kanaal
- Klokdeler
- Activeren ADC

ADC analoog kanaal



Figuur 7.1 uit Williams

ADC klokdeler

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 26-5. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

DE HAAGSE HOGESCHOOL

ADC klokdeler

“By default, the successive approximation circuitry requires an input clock frequency between **50 kHz and 200 kHz**. If a lower resolution than 10 bits is needed, the input clock frequency to the ADC can be as high as 1000 kHz to get a higher sample rate.”

DE HAAGSE HOGESCHOOL

ADC klokdeler

Klokfrequentie: 16 MHz

ADC frequentie: 50 – 200 kHz

Bereken de waarde van de klokdeler

Division Factor
2
2
4
8
16
32
64
128

DE HAAGSE HOGESCHOOL

ADC klokdeler

Bit	7	6	5	4	3	2	1	0	
(0x7A)	**ADEN**	**ADSC**	**ADATE**	**ADIF**	**ADIE**	**ADPS2**	**ADPS1**	**ADPS0**	**ADCSRA**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 26-5. ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

DE HAAGSE HOGESCHOOL

ADC klokdeler

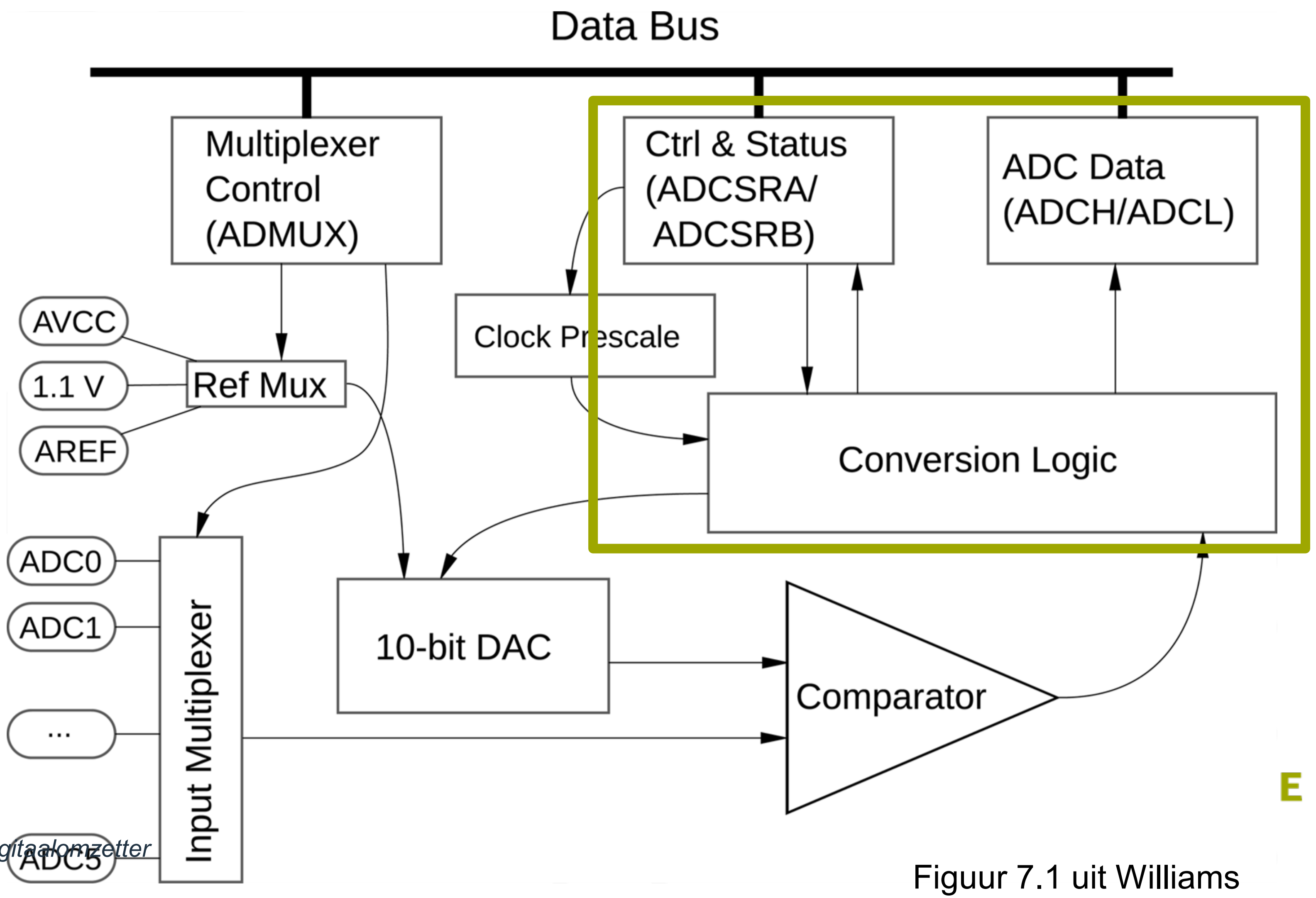
```
void init_adc (void)
{
	ADMUX = (0 << REFS1) | (1 << REFS0);
	ADCSRA = (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
}
```

ADC instellen

Instellen:

- ADC referentiespanning
- Analoog kanaal
- Klokdeler
- Activeren ADC

ADC activeren



Figuur 7.1 uit Williams

ADC activeren

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while a conversion is in progress, will terminate this conversion.

ADC klokdeler

```
void init_adc(void)
{
	ADMUX = (0 << REFS1) | (1 << REFS0);
	ADCSRA = (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
	ADCSRA |= (1 << ADEN);
}
```


DE HAAGSE HOGESCHOOL

ADC conversie starten

“ A single conversion is started by writing a **logical one** to the ADC Start Conversion bit, **ADSC**. This bit stays high as long as the conversion is in progress and will be cleared by hardware when the conversion is completed. If a different data channel is selected while a conversion is in progress, the ADC will finish the current conversion before performing the channel change. ”

ADC conversie starten

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit to one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead of the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

```c
void init_adc(void)
{
	ADMUX = (0 << REFS1) | (1 << REFS0);
	ADCSRA = (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
	ADCSRA |= (1 << ADEN);
}

int main(void)
{
	init_adc();
	ADCSRA |= (1 << ADSC);

	while (1) { }
}
```


DE HAAGSE HOGESCHOOL

ADC conversie gereed

“A single conversion is started by writing a logical one to the ADC Start Conversion bit, ADSC. **This bit stays high as long as the conversion is in progress and will be cleared by hardware when the conversion is completed**. If a different data channel is selected while a conversion is in progress, the ADC will finish the current conversion before performing the channel change.”

DE HAAGSE HOGESCHOOL

```c
void init_adc(void)
{
	ADMUX = (0 << REFS1) | (1 << REFS0);
	ADCSRA = (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
	ADCSRA |= (1 << ADEN);
}

int main(void)
{
	init_adc();
	ADCSRA |= (1 << ADSC);
	while (ADCSRA & (1 << ADSC)) { }
	while (1) { }
}
```

ADC conversie gereed

After the conversion is complete (ADIF is high), the conversion result can be found in the ADC Result Registers (ADCL, ADCH).

For single ended conversion, the result is

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

where V_{IN} is the voltage on the selected input pin and V_{REF} the selected voltage reference (see Table 26-3 on page 281and Table 26-4 on page 282). 0x000 represents analog ground, and 0x3FF represents the selected reference voltage minus one LSB.

DE HAAGSE HOGESCHOOL

```c
void init_adc(void)
{
	ADMUX = (0 << REFS1) | (1 << REFS0);
	ADCSRA = (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
	ADCSRA |= (1 << ADEN);
}

int main(void)
{
	init_adc();
	ADCSRA |= (1 << ADSC);
	while (ADCSRA & (1 << ADSC)) { }
	int adc_value = ADC;
	while (1) { }
}
```

ADC interrupts

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 3 – ADIE: ADC Interrupt Enable**

When this bit is written to one and the I-bit in SREG is set, the ADC Conversion Complete Interrupt is activated.

```
ISR(ADC_vect)
{
}
```

sei() en volatile

“WE KUNNEN NU ANALOGE INGANGSSIGNALEN VERWERKEN”

DE HAAGSE HOGESCHOOL