

Arduino

Lab solutions



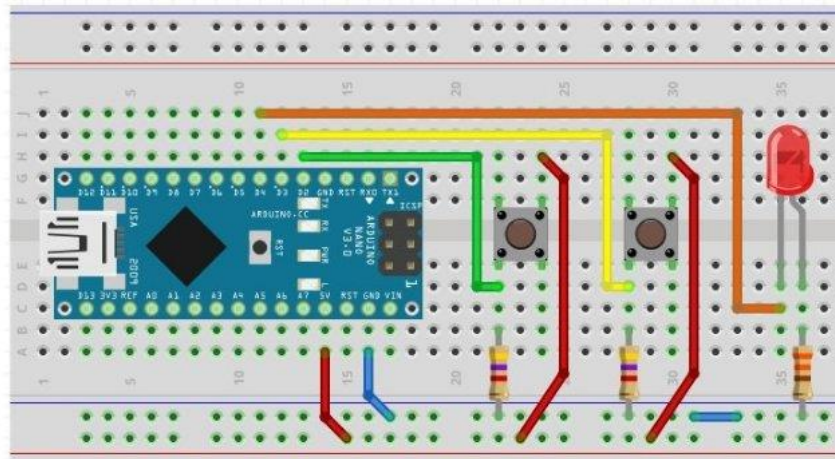
ARTESIS PLANTIJN
HOGESCHOOL ANTWERPEN

Labo 1

Definieer poorten
- Knoppen
- led

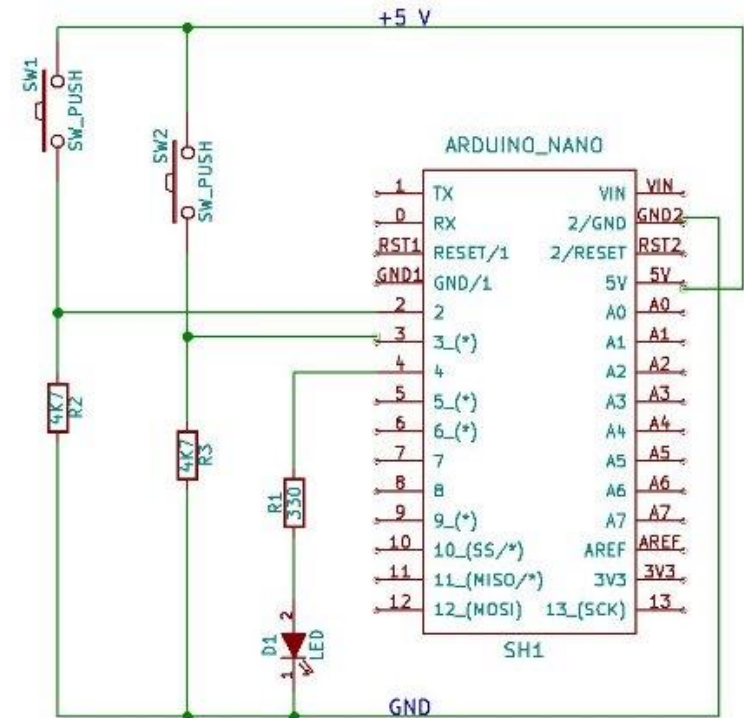
Setup
knoppen: input
led: output

Loop
Knop één ingedrukt? Zet dan de led aan
Knop twee ingedrukt? Zet dan de led uit



"on" "off"

Debounce...?



Labo 1

Setup
Led's: output

Tip! Gebruik een lus bij het instellen van meerdere outputs

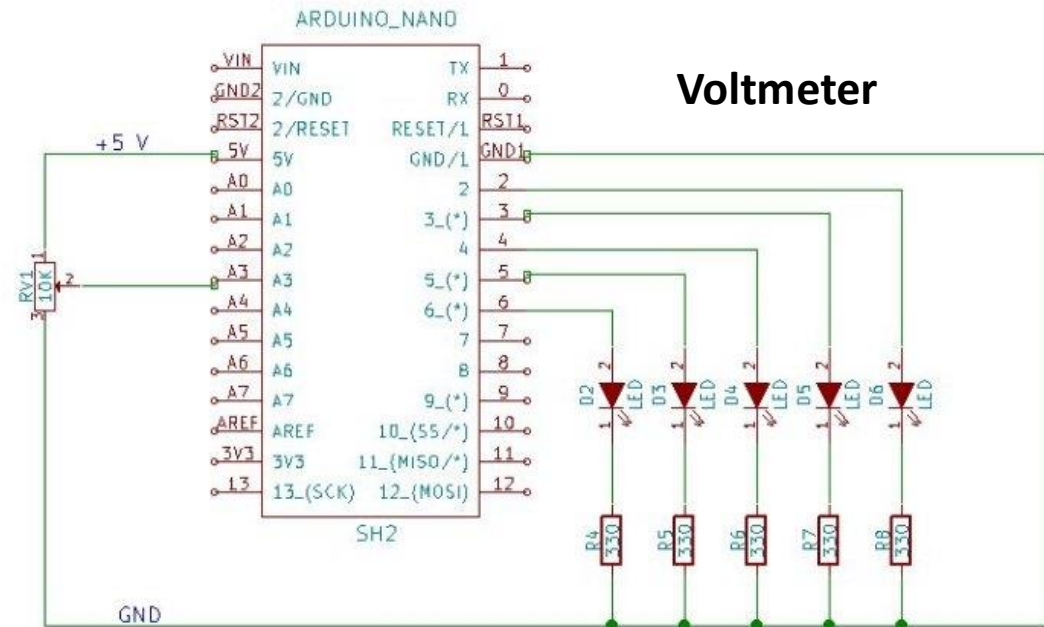


Loop
Lees de spanning op A3

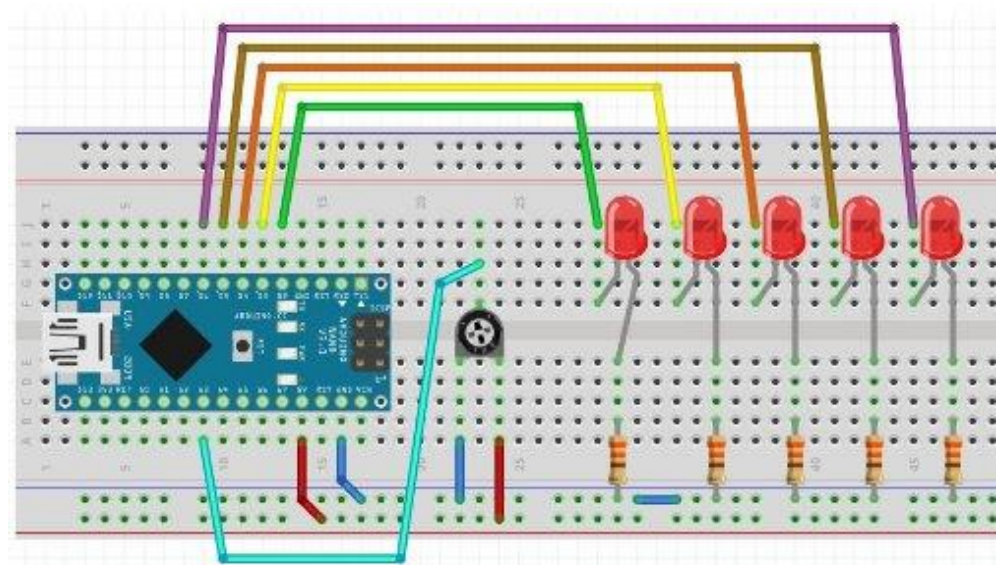
Zet elke $(1023/5) = 204$ stapjes:

- méér spanning op A3 'n extra led aan
- minder spanning op A3 'n extra led uit

Tip! Gebruik een lus



Voltmeter

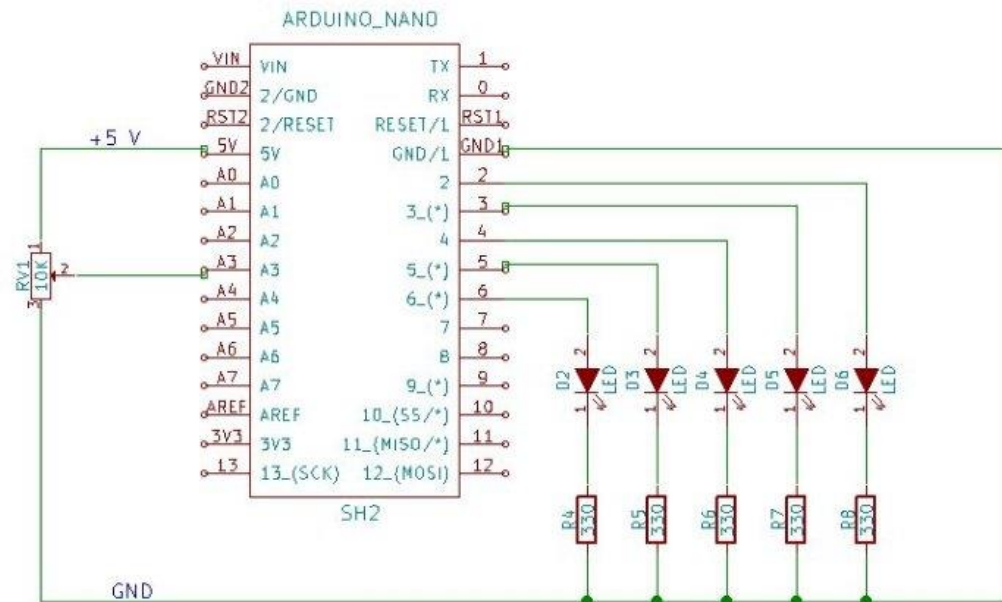


Labo 1

Setup

Led's: output

Gebruik een lus bij het instellen



Running Light

Loop

Lees de waarde van A3

Delay-tijd = waarde van A3

Ga in een lus van led2 tot led5
Zet de led aan, wacht, zet de led uit

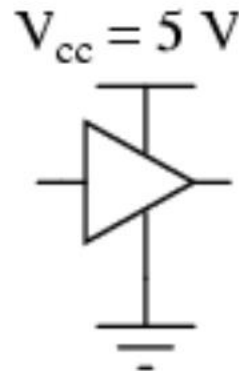
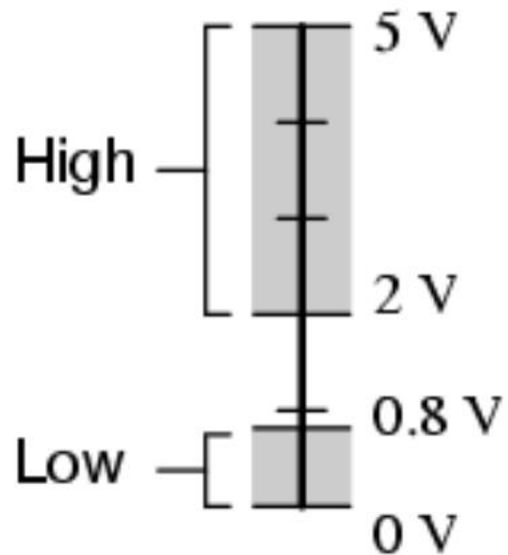
Ga in een lus van led6 tot led3
Zet de led aan, wacht, zet de led uit

We gaan niet tot led6
En gaan niet tot led2

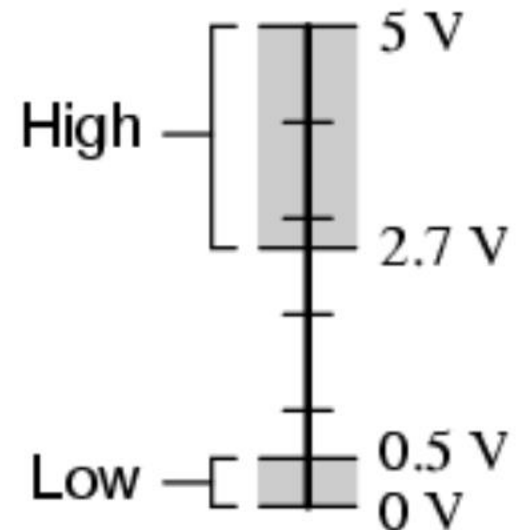
Anders branden deze led's 2 keer na elkaar...

Labo 2

Acceptable TTL gate input signal levels



Acceptable TTL gate output signal levels



Labo 2

Stroom via USB-A-poorten

Apple computers met USB 3 leveren tot 900 mA (milliampère) / 5 V (volt) aan de meeste Apple USB-randapparaten en aan alle USB-randapparaten van andere fabrikanten die voldoen aan de USB-specificaties.

Apple computers en beeldschermen met USB 1.1 of USB 2 leveren tot 500 mA (milliampère) / 5 V (volt) aan de meeste Apple USB-randapparaten en aan alle USB-randapparaten van andere fabrikanten die voldoen aan de USB-specificaties.

Labo 2

De hoeveelheid stroom dat de microcontroller **ATmega328P** verbruikt bij 8MHz en 5V voeding. Dit zoek je in de datasheet van de ATmega 328P bij de “Electrical Characteristics”.

32.2.1. ATmega328 DC Characteristics – Current Consumption

Table 32-3. DC characteristics - $T_A = -40^{\circ}\text{C}$ to 85°C , $V_{CC} = 1.8\text{V}$ to 5.5V (unless otherwise noted)

Symbol	Parameter	Condition		Min.	Typ. ⁽²⁾	Max.	Units
I_{CC}	Power Supply Current ⁽¹⁾	Active 1MHz, $V_{CC} = 2\text{V}$	$T = 85^{\circ}\text{C}$		0.3	0.5	mA
		Active 4MHz, $V_{CC} = 3\text{V}$	$T = 85^{\circ}\text{C}$		1.7	3.5	
		Active 8MHz, $V_{CC} = 5\text{V}$	$T = 85^{\circ}\text{C}$		5.2	12	
		Idle 1MHz, $V_{CC} = 2\text{V}$	$T = 85^{\circ}\text{C}$		0.04	0.5	
		Idle 4MHz, $V_{CC} = 3\text{V}$	$T = 85^{\circ}\text{C}$		0.3	1.5	
		Idle 8MHz, $V_{CC} = 5\text{V}$	$T = 85^{\circ}\text{C}$		1.2	5.5	
	Power-save mode ⁽³⁾	32kHz TOSC enabled, $V_{CC} = 1.8\text{V}$	$T = 85^{\circ}\text{C}$		0.8		μA
		32kHz TOSC enabled, $V_{CC} = 3\text{V}$	$T = 85^{\circ}\text{C}$		0.9		
	Power-down mode ⁽³⁾	WDT enabled, $V_{CC} = 3\text{V}$	$T = 85^{\circ}\text{C}$		4.2	15	
		WDT disabled, $V_{CC} = 3\text{V}$	$T = 85^{\circ}\text{C}$		0.1	2	

Labo 2

Zoek in de datasheet van de ATmega 328P in de Electrical Characteristics de logische niveaus

op:

- VOH
- VOL
- VIH
- VIL

V _{OL}	Output Low Voltage ⁽⁴⁾ except RESET pin	I _{OL} = 20mA, V _{CC} = 5V	T _A =85°C			0.9	V
			T _A =105°C ⁽⁵⁾			1.0	V
		I _{OL} = 10mA, V _{CC} = 3V	T _A =85°C			0.6	V
			T _A =105°C ⁽⁵⁾			0.7	V
V _{OH}	Output High Voltage ⁽³⁾ except Reset pin	I _{OH} = -20mA, V _{CC} = 5V	T _A =85°C	4.2			V
			T _A =105°C ⁽⁵⁾	4.1			V
		I _{OH} = -10mA, V _{CC} = 3V	T _A =85°C	2.3			V
			T _A =105°C ⁽⁵⁾	2.1			V

32.2. Common DC Characteristics

Table 32-2. Common DC characteristics T_A = -40°C to 105°C, V_{CC} = 1.8V to 5.5V (unless otherwise noted)

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
V _{IL}	Input Low Voltage, except XTAL1 and $\overline{\text{RESET}}$ pin	V _{CC} = 1.8V - 2.4V	-0.5		0.2V _{CC} ⁽¹⁾	V
		V _{CC} = 2.4V - 5.5V	-0.5		0.3V _{CC} ⁽¹⁾	
V _{IH}	Input High Voltage, except XTAL1 and $\overline{\text{RESET}}$ pins	V _{CC} = 1.8V - 2.4V	0.7V _{CC} ⁽²⁾		V _{CC} + 0.5	V
		V _{CC} = 2.4V - 5.5V	0.6V _{CC} ⁽²⁾		V _{CC} + 0.5	

Labo 2

Indien twee ATmega's digitaal met elkaar communiceren via een draad, hoeveel mag een gezonden '1' zakken om nog als een '1' geïnterpreteerd te worden door de ontvanger. De gezonden '1' is worst case.

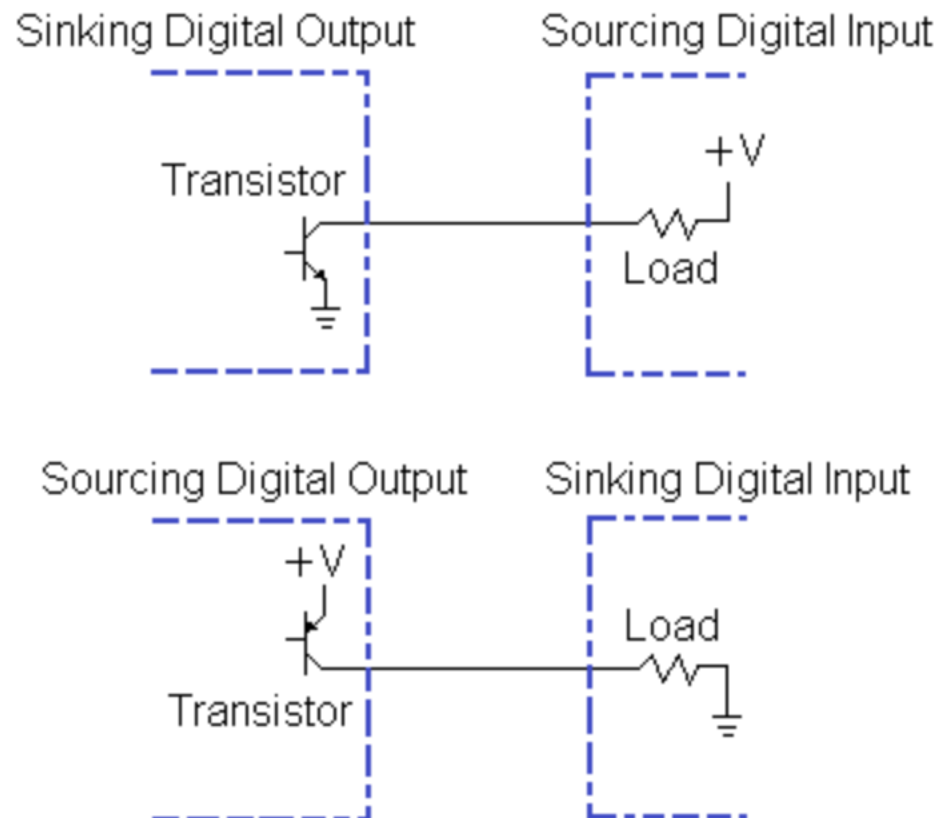
Indien twee ATmega's digitaal met elkaar communiceren via een draad, hoeveel mag een (worst case) gezonden '0' stijgen om nog als een '0' geïnterpreteerd te worden door de ontvanger. De gezonden '0' is worst case.

V_{OL}	Output Low Voltage ⁽⁴⁾ except RESET pin	$I_{OL} = 20\text{mA}$, $V_{CC} = 5\text{V}$	$T_A = 85^\circ\text{C}$			0.9	V
			$T_A = 105^\circ\text{C}^{(5)}$			1.0	V
		$I_{OL} = 10\text{mA}$, $V_{CC} = 3\text{V}$	$T_A = 85^\circ\text{C}$			0.6	V
			$T_A = 105^\circ\text{C}^{(5)}$			0.7	V
V_{OH}	Output High Voltage ⁽³⁾ except Reset pin	$I_{OH} = -20\text{mA}$, $V_{CC} = 5\text{V}$	$T_A = 85^\circ\text{C}$	4.2			V
			$T_A = 105^\circ\text{C}^{(5)}$	4.1			V
		$I_{OH} = -10\text{mA}$, $V_{CC} = 3\text{V}$	$T_A = 85^\circ\text{C}$	2.3			V
			$T_A = 105^\circ\text{C}^{(5)}$	2.1			V

V_{IL}	Input Low Voltage, except XTAL1 and $\overline{\text{RESET}}$ pin	$V_{CC} = 1.8\text{V} - 2.4\text{V}$	-0.5		$0.2V_{CC}^{(1)}$	V
		$V_{CC} = 2.4\text{V} - 5.5\text{V}$	-0.5		$0.3V_{CC}^{(1)}$	
V_{IH}	Input High Voltage, except XTAL1 and $\overline{\text{RESET}}$ pins	$V_{CC} = 1.8\text{V} - 2.4\text{V}$	$0.7V_{CC}^{(2)}$		$V_{CC} + 0.5$	V
		$V_{CC} = 2.4\text{V} - 5.5\text{V}$	$0.6V_{CC}^{(2)}$		$V_{CC} + 0.5$	

Labo 2

Zoek op wat “sinken” en “sources” van stroom wilt zeggen.



Labo 2

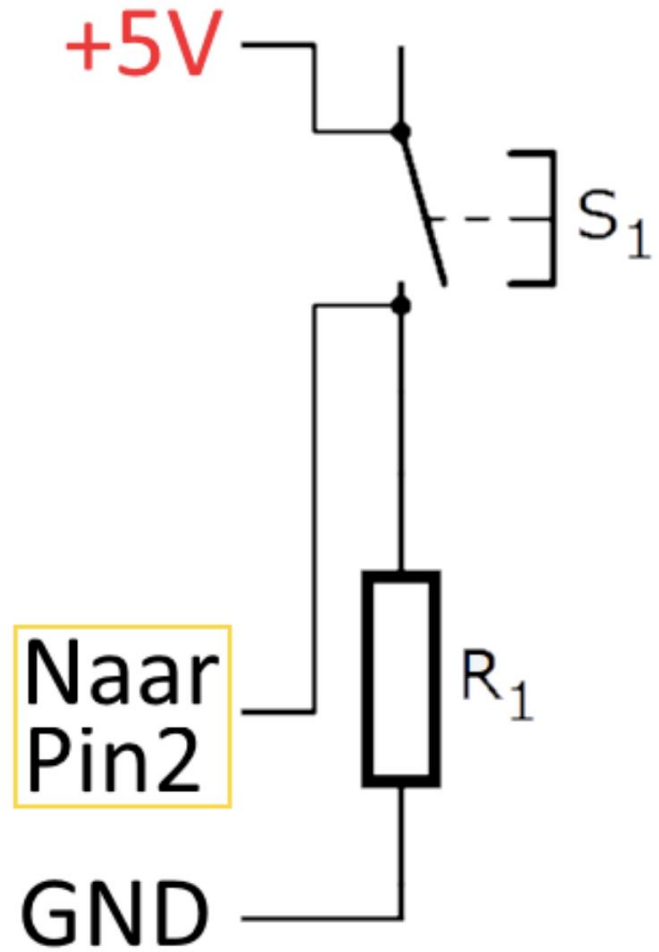
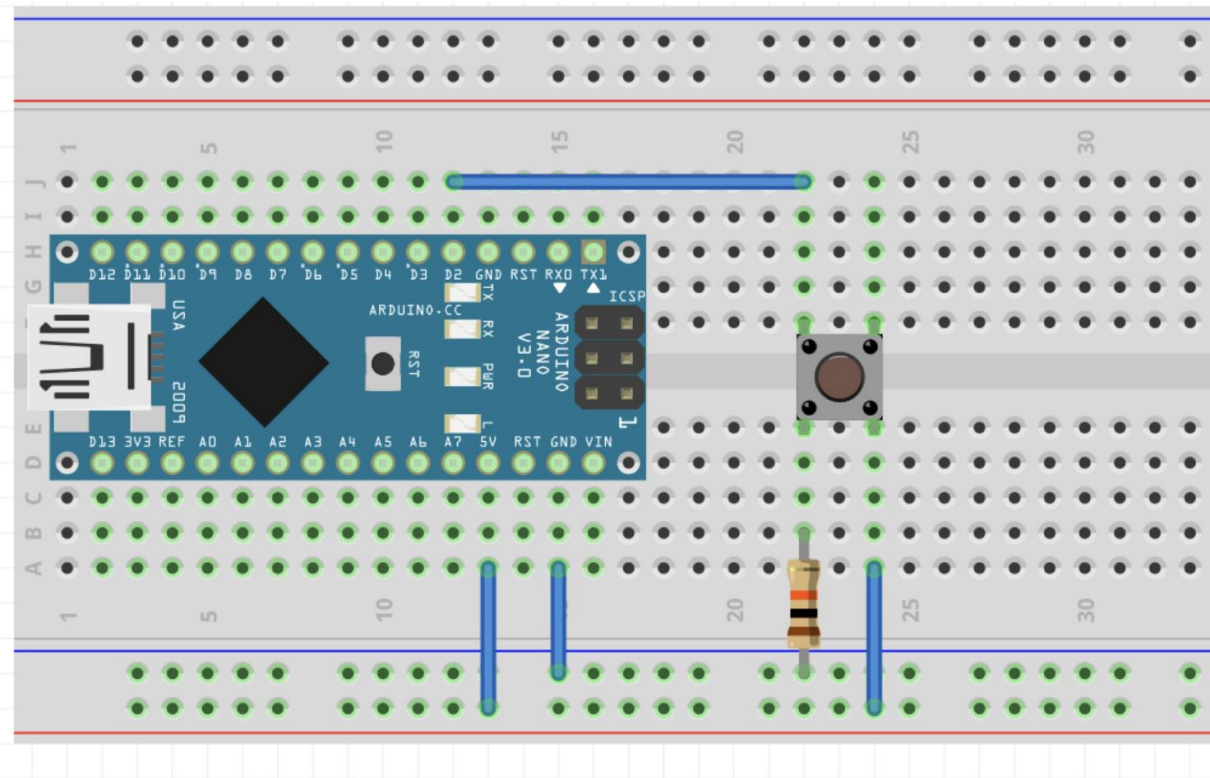
Zoek de teststroom op waarbij de logische niveaus gegarandeerd blijven.

Zoek de waarde voor de totale maximale stroom geleverd door verschillende uitgangen van de ATmega.

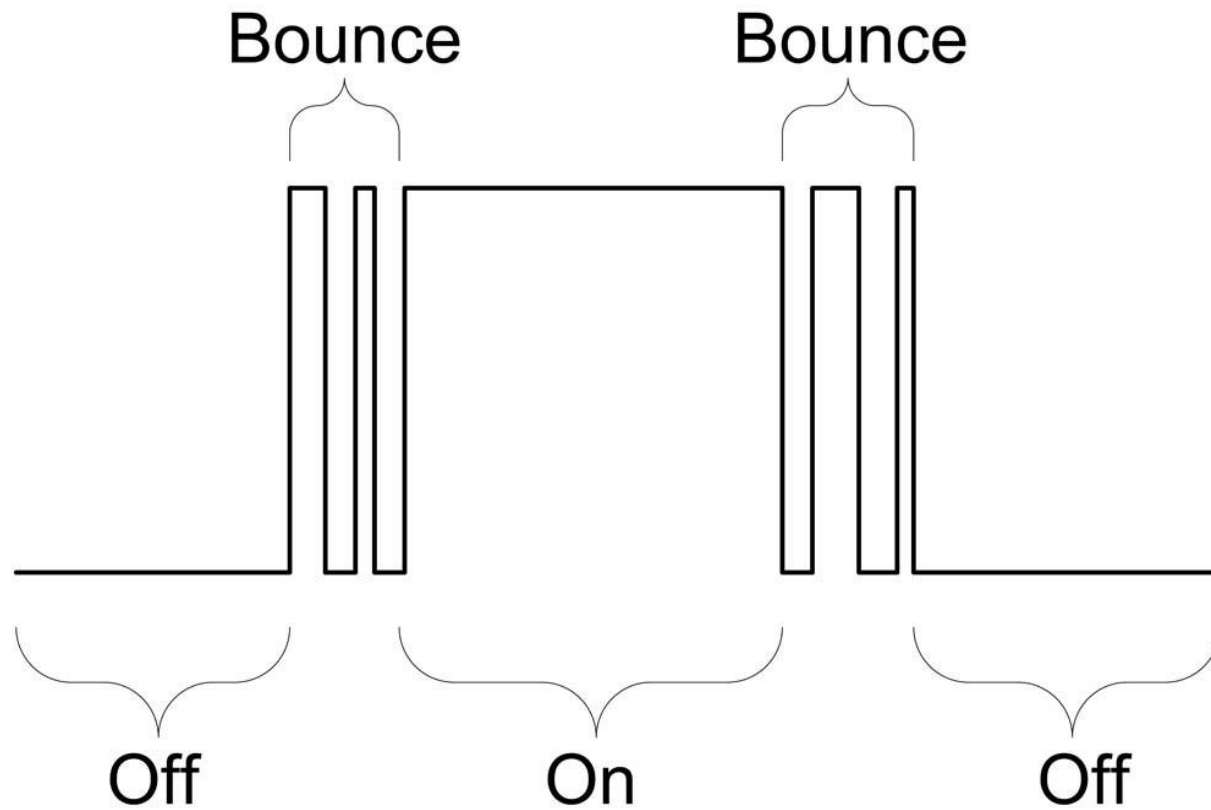
V_{OL}	Output Low Voltage ⁽⁴⁾ except RESET pin	$I_{OL} = 20\text{mA}$, $V_{CC} = 5\text{V}$	$T_A = 85^\circ\text{C}$			0.9	V
			$T_A = 105^\circ\text{C}^{(5)}$			1.0	V
		$I_{OL} = 10\text{mA}$, $V_{CC} = 3\text{V}$	$T_A = 85^\circ\text{C}$			0.6	V
			$T_A = 105^\circ\text{C}^{(5)}$			0.7	V
V_{OH}	Output High Voltage ⁽³⁾ except Reset pin	$I_{OH} = -20\text{mA}$, $V_{CC} = 5\text{V}$	$T_A = 85^\circ\text{C}$	4.2			V
			$T_A = 105^\circ\text{C}^{(5)}$	4.1			V
		$I_{OH} = -10\text{mA}$, $V_{CC} = 3\text{V}$	$T_A = 85^\circ\text{C}$	2.3			V
			$T_A = 105^\circ\text{C}^{(5)}$	2.1			V

DC Current per I/O Pin	40.0mA
DC Current V_{CC} and GND Pins	200.0mA

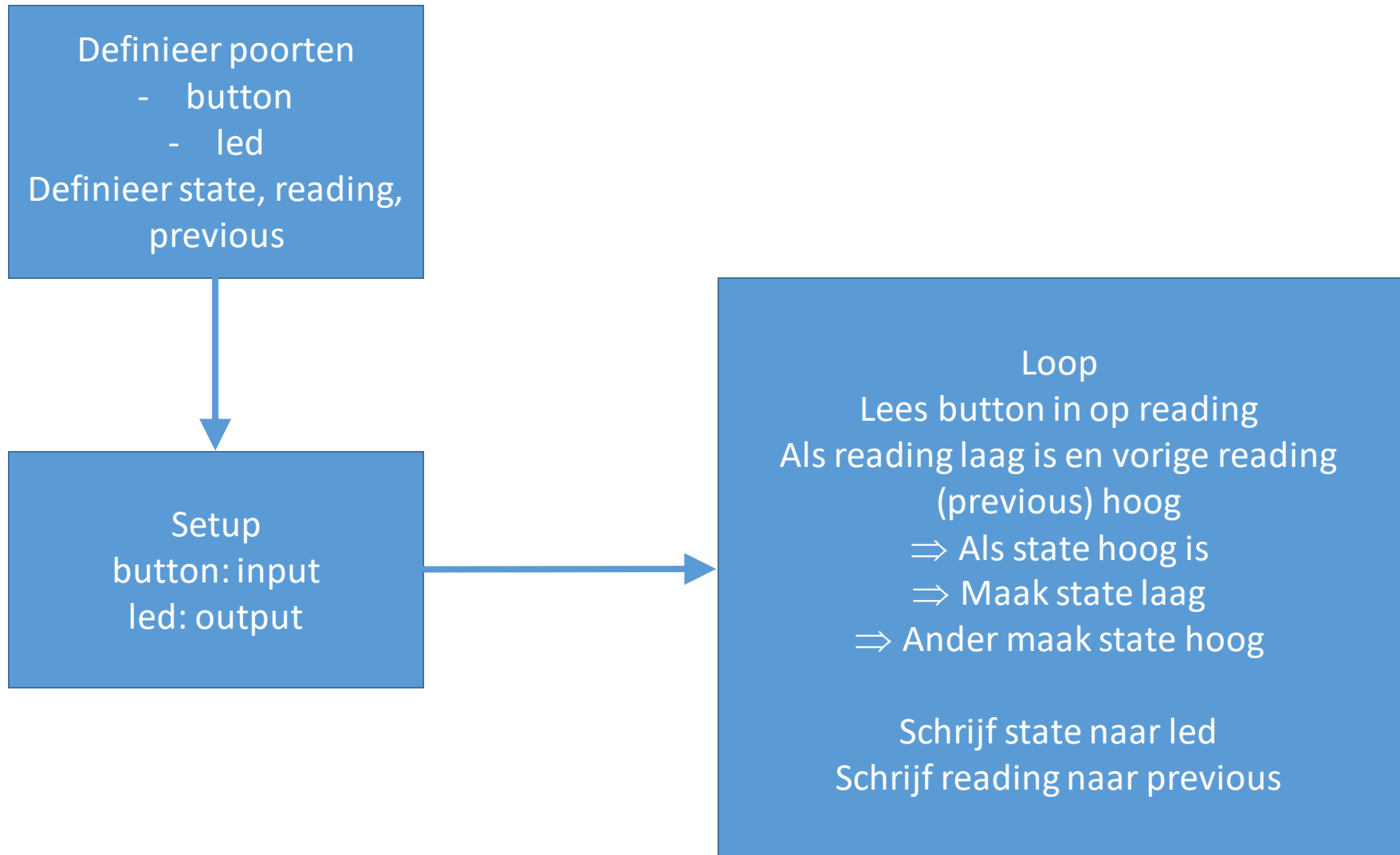
Labo 2



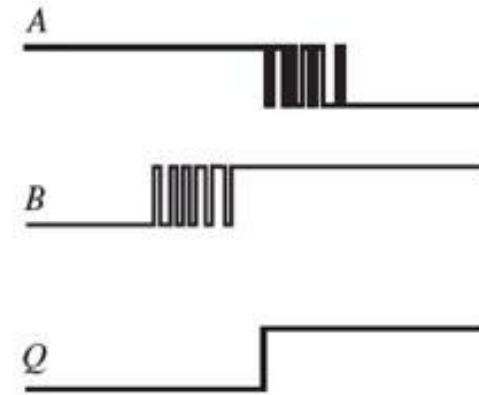
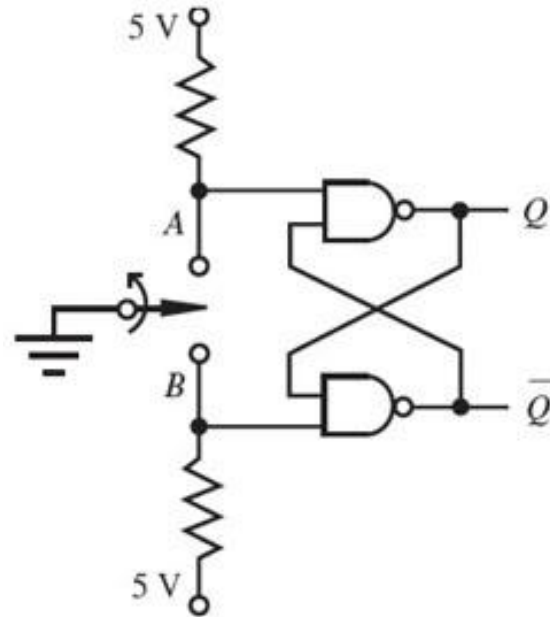
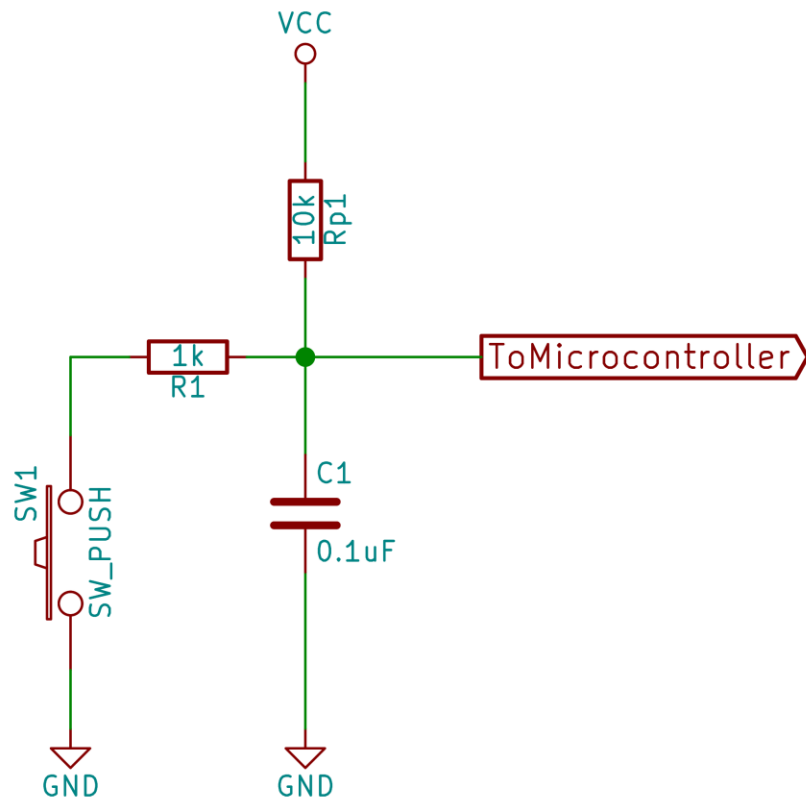
Labo 2



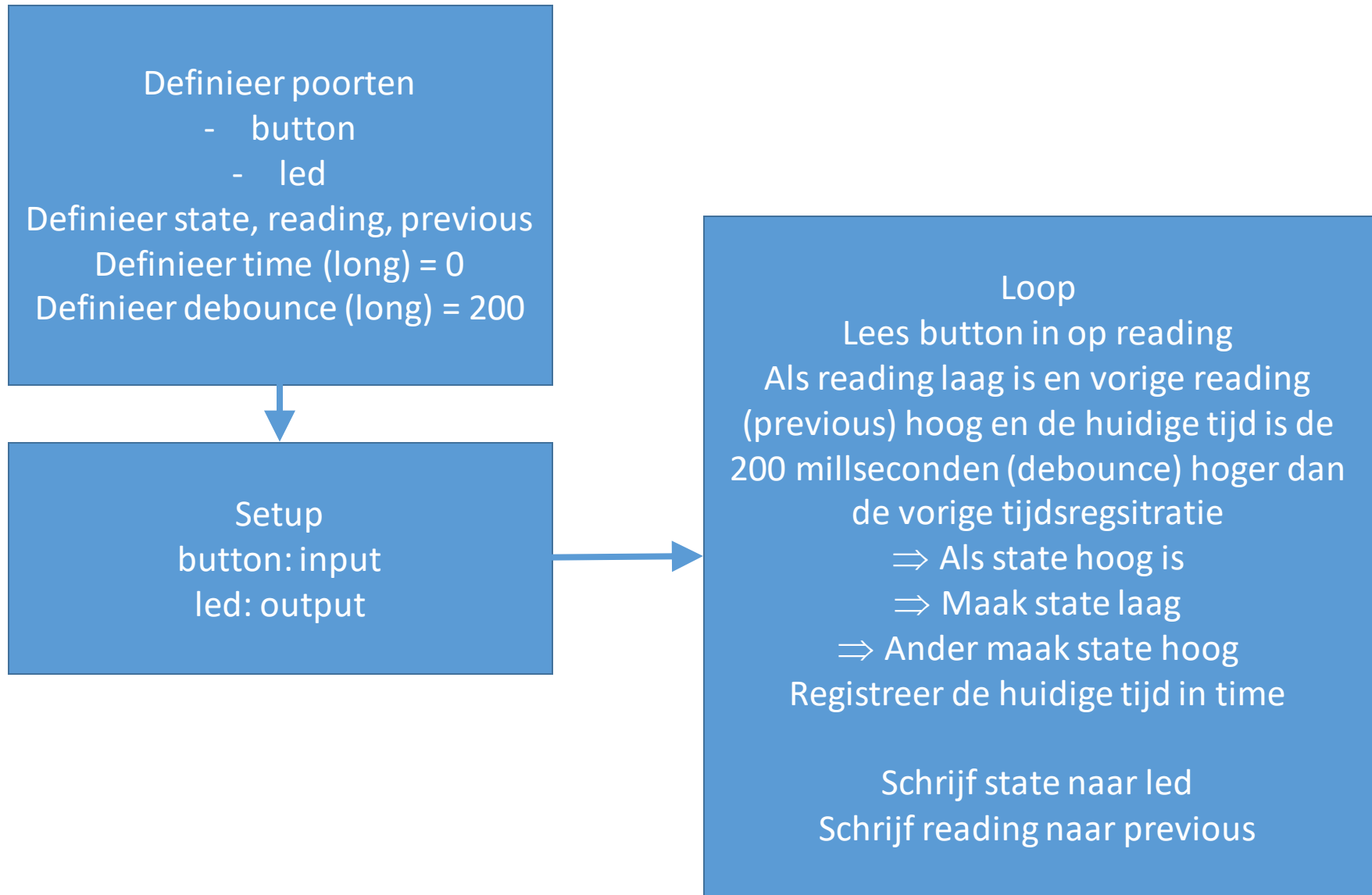
Labo 2 – button as switch



Labo 2



Labo 2 – button as switch debounced



Labo 3

Serial Monitor

Interrupts

Labo 3

Serial Monitor Voltmeter

Definieer een variabele
voor ingang A3

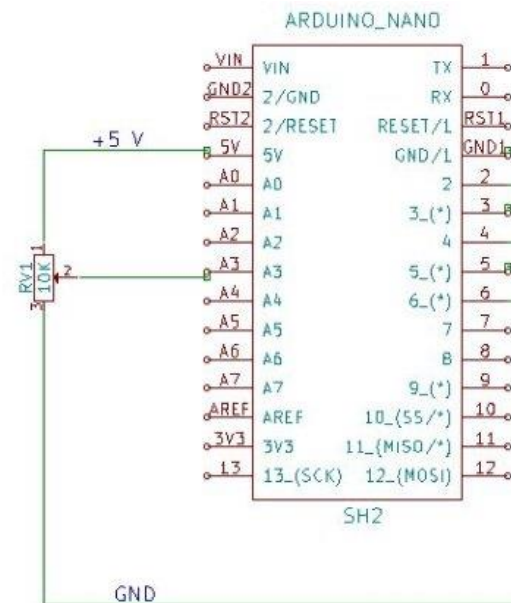


Setup
Stel de snelheid van de seriële verbinding in



Loop
analoge ingang = 0.. 1023
aantal sterretjes = 0.. 50

Druk éérst de "O" op de seriële monitor
Druk in een lus het aantal nodige sterretjes
Ga naar de volgende lijn op het scherm



Labo 3

Definieer variabelen
voor knoppen en led



Setup
Zet led als uitgang
Zet knoppen als interrupt:
is het "rising" of "falling" ... ?

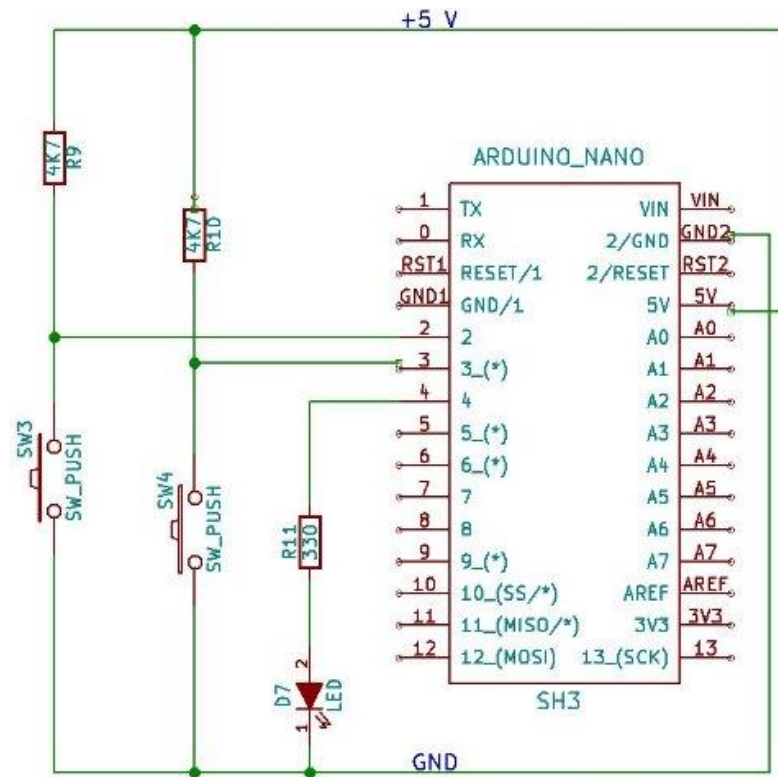


Loop = leeg !

ISR "aan" -> zet led aan

ISR "uit" -> zet led uit

"on" "off" met interrupts



Labo 3

Definieer knoppen en leds
 Definieer vlaggen "aan" en "uit"
 Definieer aantal 'aan' leds
 Definieer debounce tijd

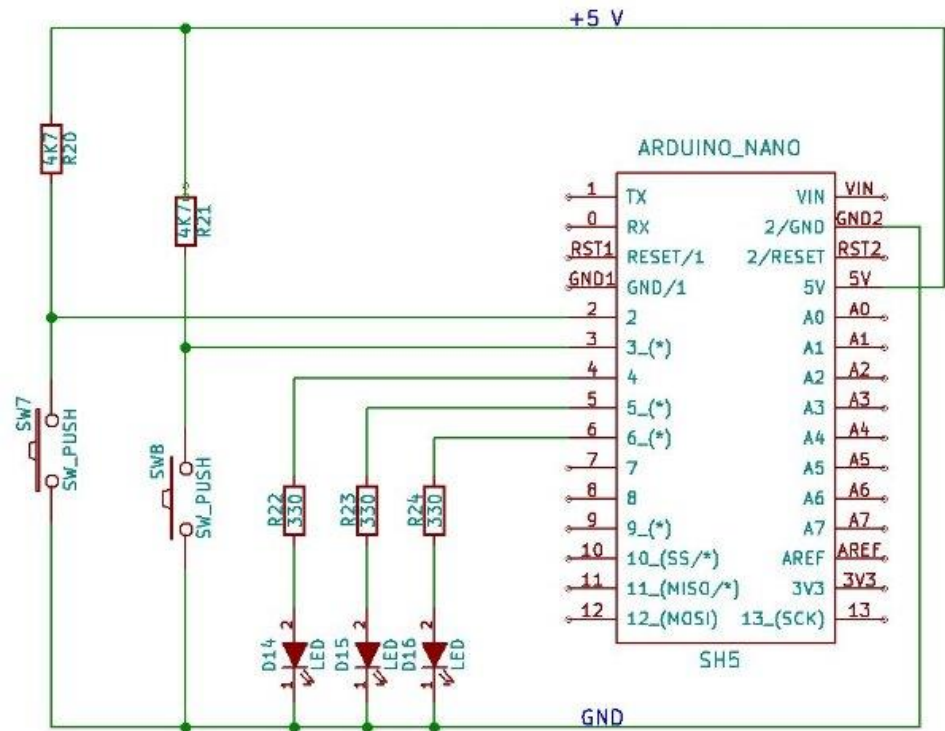


Setup
 Led's = uitgang
 knoppen = interrupt:
 ISR's voor aan en uit

ISR "aan" -> zet de aan vlag

ISR "uit" -> zet de uit vlag

3 led's met interrupts



Labo 3

3 led's met interrupts



Loop:

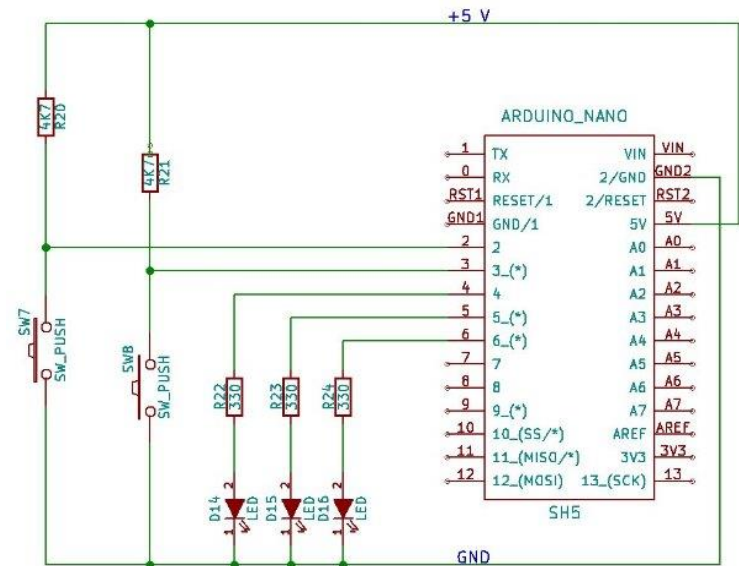
aan vlag gezet?

- debounce !
- aantal 'aan' leds = ééntje méér
- wis aan vlag

uit vlag gezet?

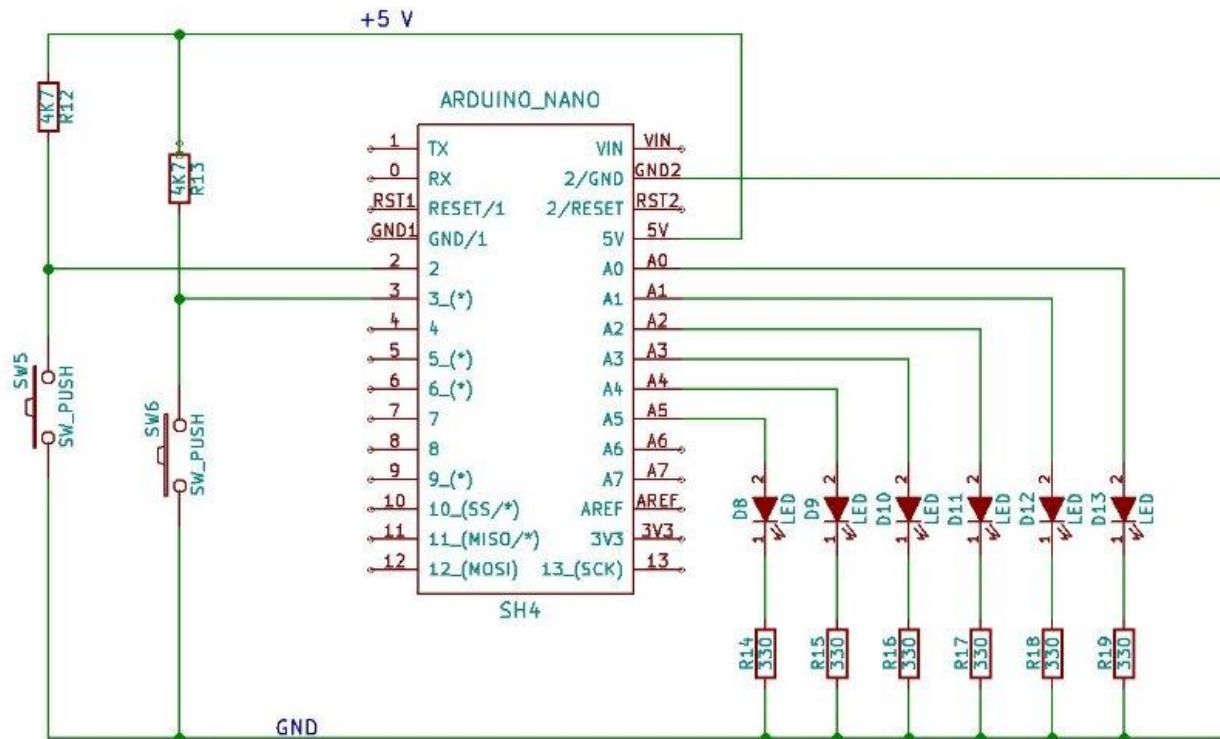
- debounce !
- aantal 'aan' leds = ééntje minder
- wis uit vlag

Zet het aantal 'aan' leds aan,
Zet de andere uit.



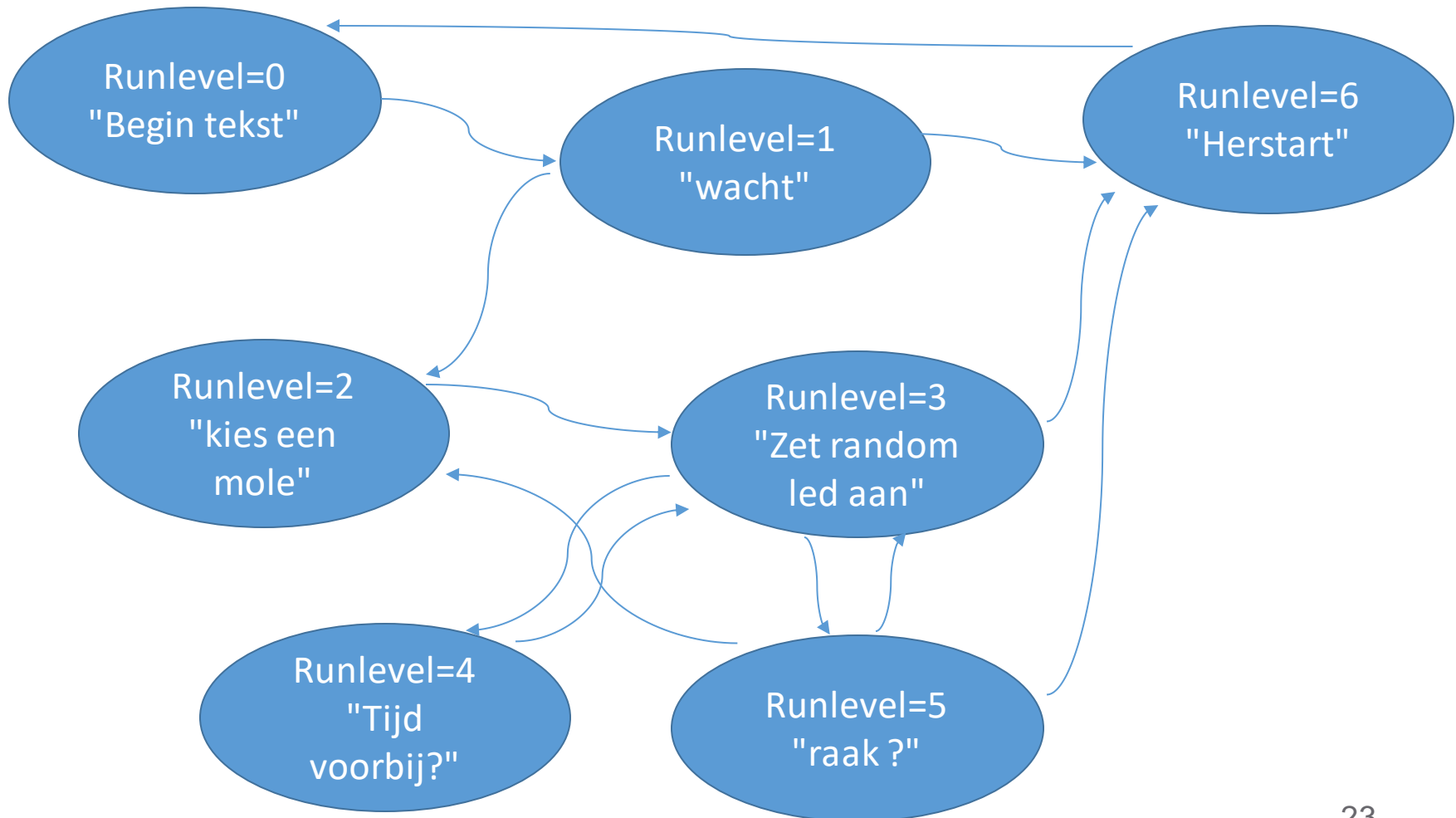
Labo 3

Whack-a-mole



Labo 3

We gebruiken hiervoor **runlevel's** of **state's** = "state machine" principe :



Labo 3

Definieer knoppen & leds
Tip! Gebruik een array voor de leds
Definieer runlevel vlag

Definieer tijd variabelen
- wachttijd
- defaultwachttijd
Definieer spel avariabelen
-aantal pogingen
- te raden led
- random led



Setup
Led's = uitgang
knoppen = interrupt:
ISR's voor "hit" en "einde spel"

ISR "hit"
Start v/h spel gedrukt?
runlevel wordt 2

Tijdens spel gedrukt?
runlevel wordt 5

ISR "einde spel"
runlevel wordt 6

Labo 3

Loop:

Indien Runlevel 0
"Druk op hit om te starten"
Verder naar level 1

Indien Runlevel 1
Doe niets (wachten)

Indien Runlevel 2
Definieer wat de 'mole' is m.b.v. random
Verder naar level 3

Indien Runlevel 3
Laat een random led branden
Bewaar het actuele tijdstip als referentie
Verder naar level 4

Labo 3

Indien Runlevel 4

Hoe lang is het referentie tijdstip geleden?

Indien meer dan de wachttijd :

Doe random led terug uit
terug naar level 3

Indien Runlevel 5

Is de random led = mole ?

Indien ja: "Geraakt ! aantal pogingen = ..."
Verminder de wachttijd variabele
terug naar level 2

Indien nee: "Fout gemikt...."
terug naar level 3

Indien Runlevel 6

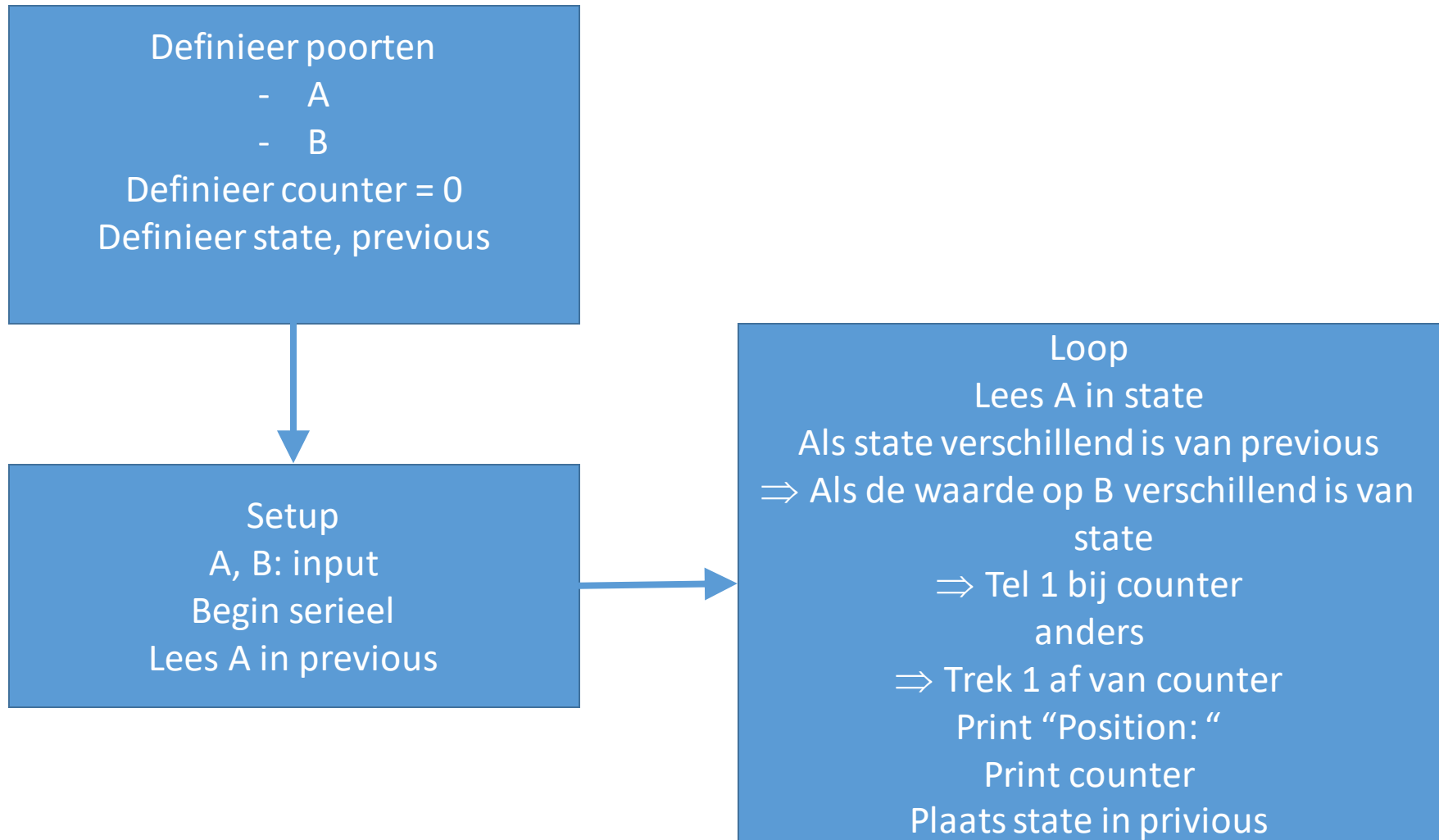
wachttijd = defaultwachttijd

"Spel onderbroken – opnieuw gestart"
terug naar level 0

Labo 4



Labo 4: rotary



Labo 4: 7 seg led test

Setup

Poorten 2-12: output

2-8: ABCDEFG

9-12: Cat 1-3

Alle cat: hoog



Loop

Loop door alle kathodes

- Maak de kathode laag

Loop door alle anodes

- maak ze allen hoog,
- maak ze allen laag
- Maak de kathode hoog

Labo 5

EEPROM

Lezen van de Serial Monitor

Labo 5

Bijhouden boot sequence nr in EEPROM

Voeg EEPROM bibliotheek toe
Definieer geheugen-adres=0x00
Definieer data variabele



Setup
Lees data uit de EEPROM

Begin serieel, snelheid = 9600 baud
Druk data af op seriële monitor

data++

Schrijf data terug naar EEPROM
zelfde geheugen adres



Loop = leeg !

Wat was het verschil tussen reset en reboot ??

Bij reset:

Data wordt telkens één meer: 0, 1, 2, 3...

Bij reboot:

Data wordt telkens twee meer: 0, 2, 4, 6 ...

- Dit komt omdat bij reboot de chip zichzelf nog eens extra gaat resetten*
- Dit feit kan handig gebruikt worden in software om te weten of dat de chip gereset was (door de gebruiker), ofdat hij gereboot is (bvb doordat de spanning wegviel)*

Labo 5

Cijfer lezen van Terminal
&
schrijven naar 7segment

Definieer cijfer
Definieer bcd[10][7] matrix

Functie "sevenWrite"
die één 7-segment aanstuurt via bcd[10][7] matrix

Setup
Schrijf "0" via sevenWrite
Begin serieel, snelheid = 9600 baud

loop
Zit er iets in de seriële buffer ?
Cijfer = resultaat lezen v/d buffer
Schrijf cijfer via sevenWrite

*Wat we lezen van de
seriële-monitor is **ASCII**
= cijfers van 0..127
die letters vertegenwoordigen*

*De keyboard input "3" is niet het
cijfer 3 maar de ASCII code van dit
symbool = 51*

*Truukje:
ASCII waarde 48 = symbool "0"
ASCII waarde 49 = symbool "1"
ASCII waarde 50 = symbool "2"*

*...
Snelle omzetting
van ASCII naar écht cijfer:*

cijfer = ASCII - 48

Labo 5

int's lezen van de Terminal

Definieer int variabele 'getal' te lezen via serieel
Definieer int voor analoge pin



Setup
Begin serieel, snelheid = 9600 baud
pinMode van de LED's



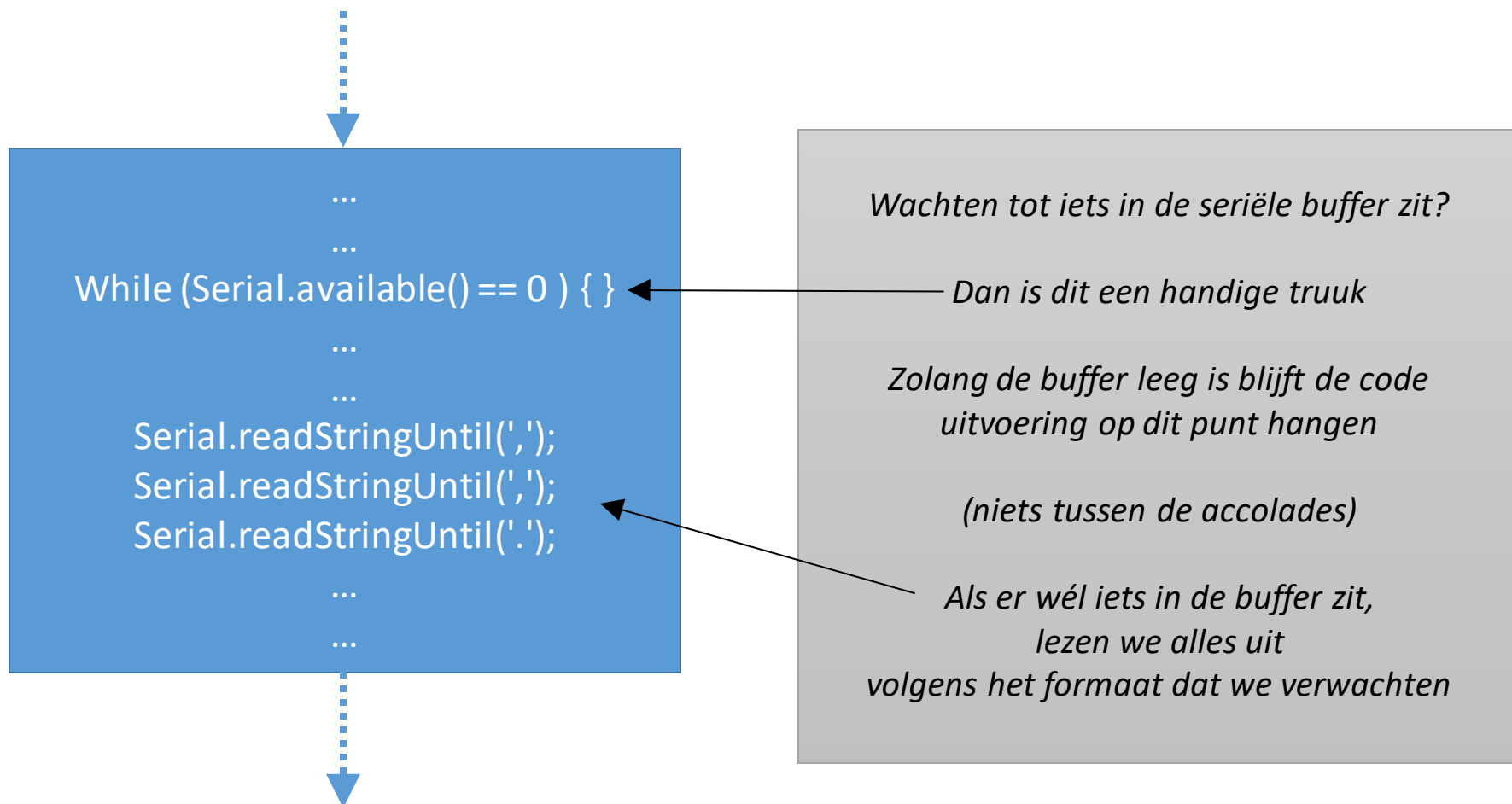
Loop
Lees de analoge ingang

Zit nog iets in de seriële buffer ?
Lees de inhoud met "parseInt" = 'getal'

analoge ingang < getal ? --> rode led aan
analoge ingang > getal ? --> groene led aan

Labo 5

Schrijven van strings in de EEPROM



Labo 5

Schrijven van strings in de EEPROM



...

Lus doorheen alle strings:

Wat is de lengte van de string? `stringvariabele.length()`

Omzetten string naar char array:
`stringvariabele.toCharArray(charlijst, lengte)`

Schrijf char per char de char array naar EEPROM
telkens op het volgende geheugen adres

Schrijf op het einde van de char array
0x00 naar EEPROM op het volgende geheugen adres
= aanduiding einde van de string



Loop = leeg (slechts één keer uitvoering nodig)

Waar schrijf je de strings?

Bvb:

*String1 vanaf
adres 0x100 ...*

*String2 vanaf
adres 0x200 ...*

*String3 vanaf
adres 0x300 ...*

Labo 5

Weergeven welkombericht uit de EEPROM

Setup:

...

For lus van i = 0 tot 2

Adres = i x 100

Todat char = 0x00:
lees char uit EEPROM
druk char af op seriël

Char = 0x00?

Nee? Lees volgende char

Ja ? Ga naar de volgende lijn op de terminal

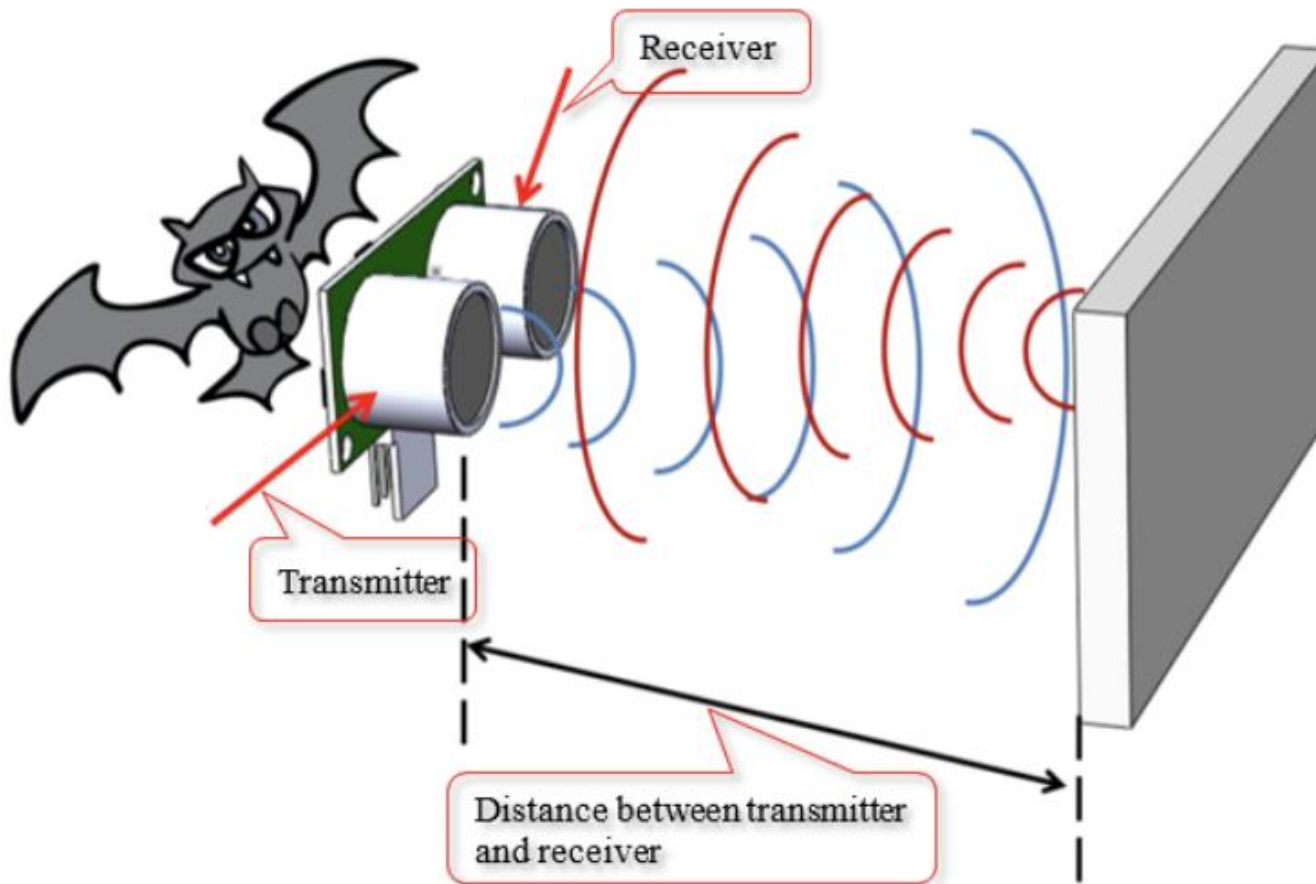
...

...

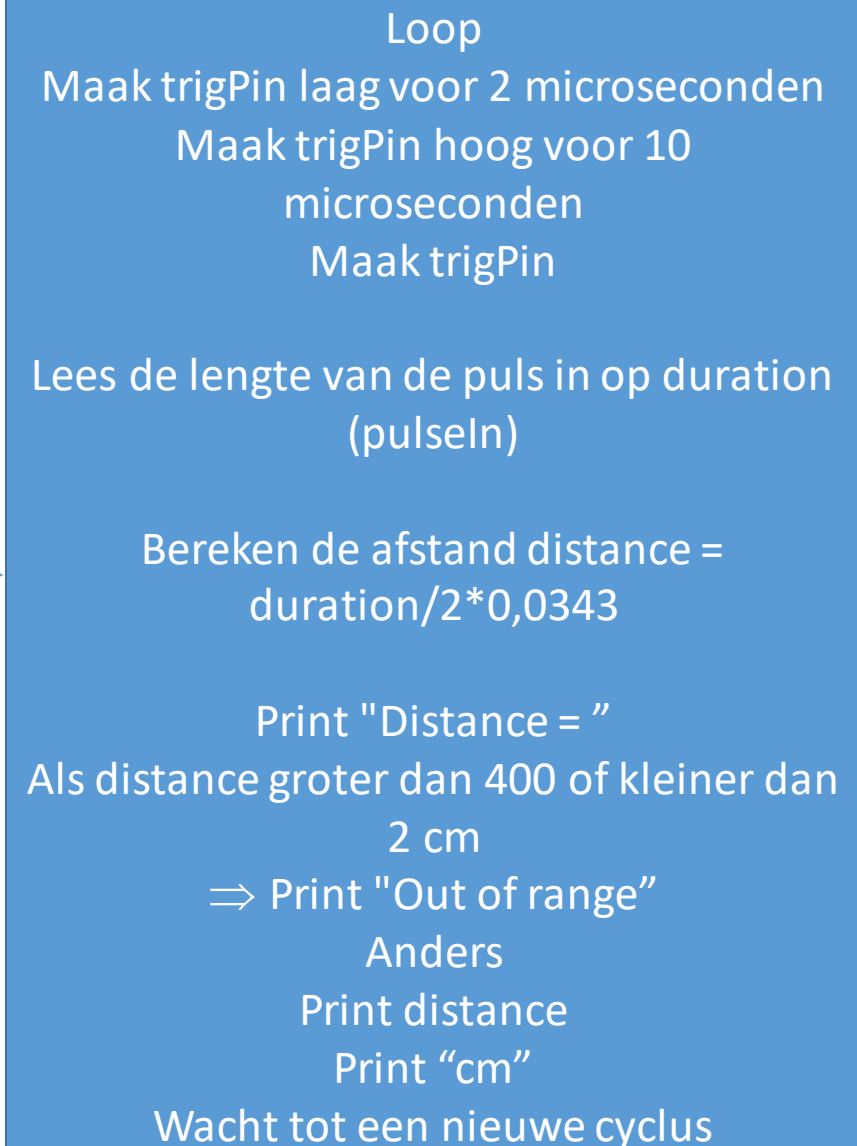
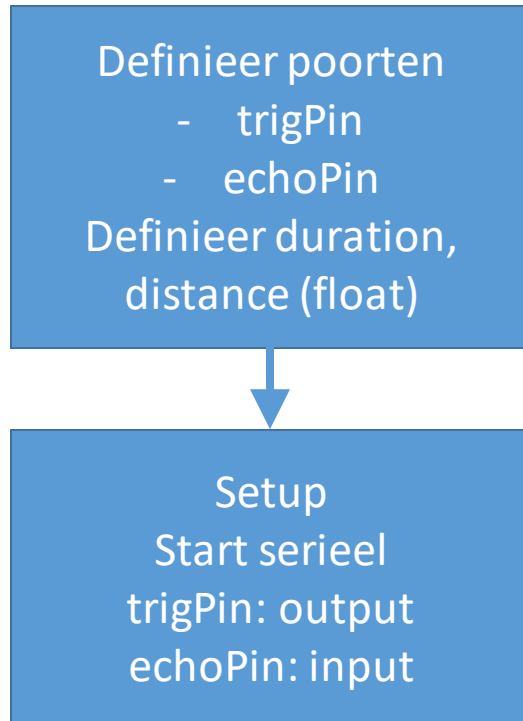
*Dit stukje code voegen we toe in het
bestaande programma nét in het
begin van de setup.*

*Zo zal telkens de chip gereset
wordt de tekst uit de EEPROM op
de seriële monitor verschijnen*

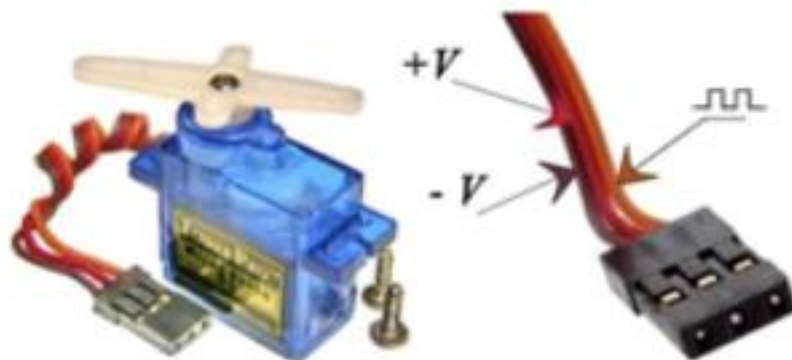
Labo 6 - sensor



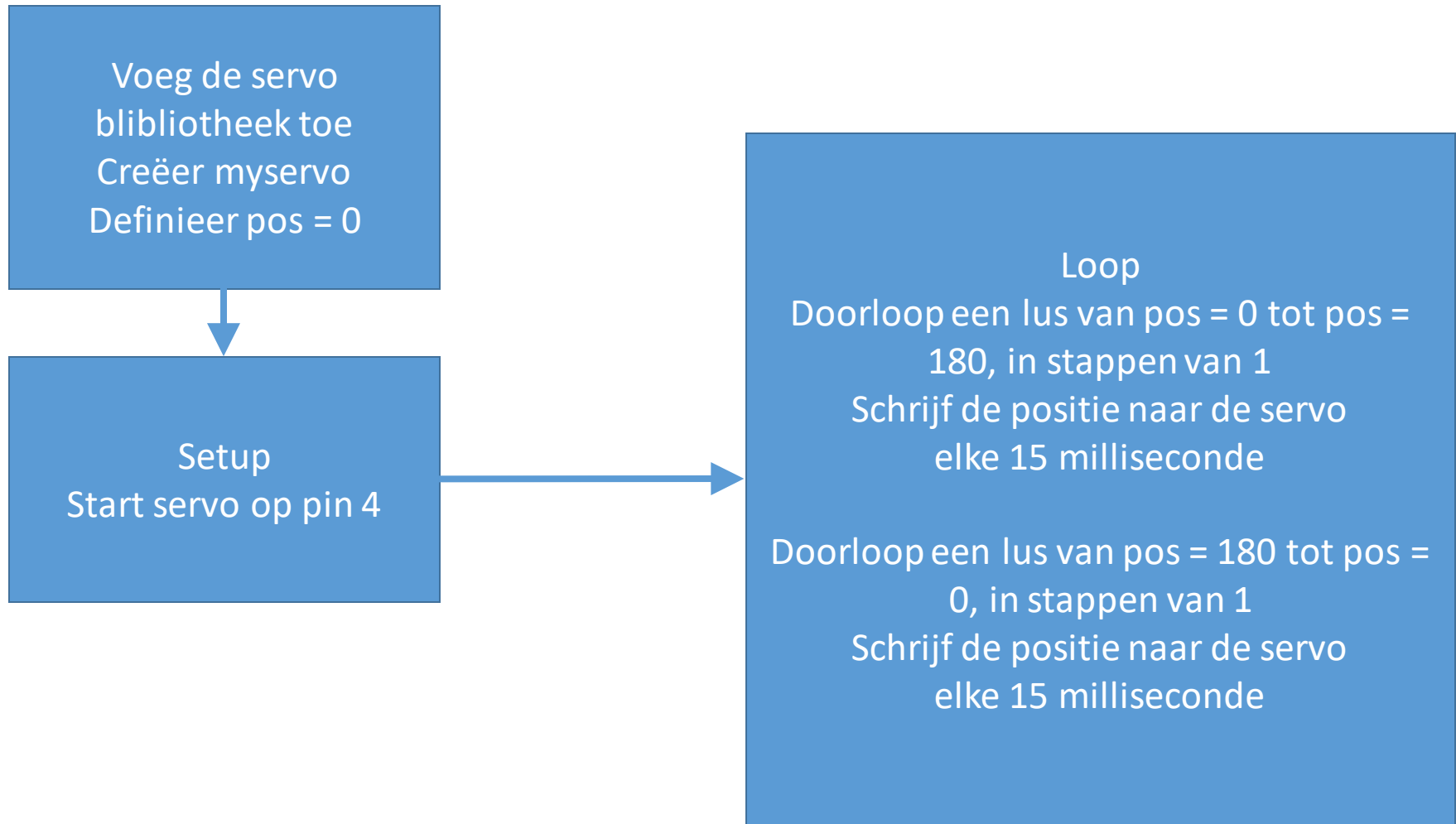
Labo 6 - sensor



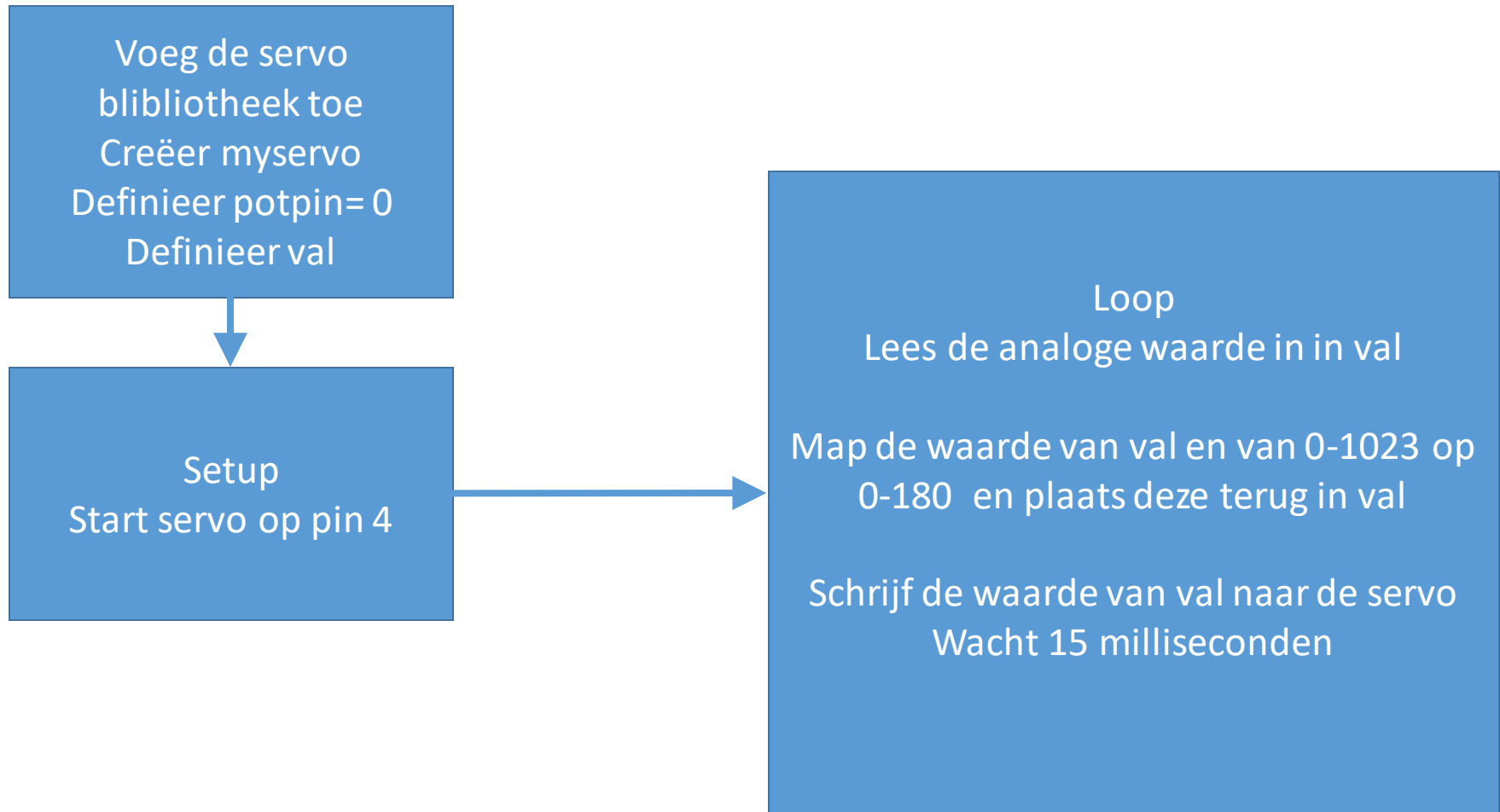
Labo 6 – servo



Labo 6 – servo1



Labo 6 – servo2



Labo 6 – sensor - servo

Voeg de servo blibliotheek toe
Definieer trigPin op 2
Definieer echoPin op 3
Definieer duration, distance
(float)
Creëer myservo



Setup
Start servo op pin 4
Start serieel
trigPin: output
echoPin: input



Loop
Maak trigPin laag voor 2 microseconden
Maak trigPin hoog voor 10
microseconden
Maak trigPin laag
Lees de lengte van de puls in op duration
(pulseIn)
Bereken de afstand $distance = duration / 2 * 0,0343$
Print "Distance = "
Print distance
Print "cm"
Als de afstand groter dan 50 is
⇒ Schrijf 135 naar servo
Als de afstand kleiner dan 20 is
⇒ Schrijf 45 naar servo
Anders
⇒ Schrijf 90 naar servo
Wacht een halve seconde

Labo 7

Pointers

Labo 7

Pointers basics

Definieer

Een byte, int, char en float

Definieer

Een bytewriter, intwriter, charwriter en floatwriter



Setup

Koppel pointers aan de variabelen:
pointerwaarde = adres van variabele

afdrukken alle tekst
op de serial monitor énkél via de pointers:

Vbv pointer p:

p = adres

*p = inhoud van dat adres

```

/dev/ttyUSB0

type byte 263 254
type int 261 99
type char 260 a
type float 256 3.14

Autoscroll No line end
  
```

Direct de waarde van een pointer afdrukken op de seriële monitor gaat niet. De code van de seriële monitor laat dit niet toe.

Oplossing: zet de pointer waarde éérst om in een 'int' en druk dan pas af:

`Serial.print(int (p))`

Labo 7

Pointers: Memory-dump

Definieer character pointer:
char *p



Setup

...

...

Lus doorheen alle adressen
van 0x0100 0x8FF

Lus doorheen 20 lijntjes:
druk éérste geheugen adres af

Lus doorheen 16 opeenvolgende adressen:
Druk inhoud van dit adres af

```

/dev/ttyUSB1
256      #  ?  ?  j  ?  ?
272  ?
-----
288  - - - e i n d e - - -
304  - - - - - 8 N    Z
320      ?  ?  ?  ?  ?
336  ?  ?  ?  ?  ?  " %"
352
368
384
400
416
416  4 1 6
432  4 1 6
448
464
☒ Autoscroll 

```

*Het eenvoudigste om doorheen al deze adressen te lopen in het programma is niet met een for lus maar met een **while***

*Start = 0x00
while adres < 0x900
verhoog adres met één.*

Labo 7

Pointers: Memory-peek

```
char tekst[] = { "Arduino is een opensource-computerplatform bedoeld om microcontrollers toegankelijker "
                "te maken voor wie er pas mee start. Met zo'n Arduino kunnen we vanalles meten, zoals drukknoppen,"
                "temperatuur, licht etc, en vanalles aansturen, zoals LED's, displays, actuatoren of motoren, etc."
                };
```

De naam van een array is reeds een pointer

Dit is een pointer naar het éérste element van de array in het geheugen

*Afrukken van dit adres
is dan ook eenvoudig door :*

`int(tekst)`

Naar de seriële monitor te sturen

```
/dev/ttyUSB1
1856 > ? ? ? ? ? ? ? ? ? ? < ? ? ? ?
1872 ? ? ? ? m ? ? ? ? = { > ? ? o ? ?
1888 ? ? ? ? } ? ? ? ? u ? T ? j ? ? ? n
1904 ? ? ? ? ? UN ? ? N ? ? 5 ? y
1920 ? ? | ? ? ? ? ? w # r > ?
1936 0 ? ? ? ? # ? A ? ?
1952 ? ? ? ? ? ? ? ? z g ?
1968 3 ? w # r " #

1984 # ? w ? ? )
2000 ! ? ? Arduino
2016 is een opensourc
2032 e-computerplatfo
2048 rm bedoeld om mi
2064 crocontrollers t
2080 oegankelijker te
2096 maken voor wie
2112 er pas mee start
2128 , Met zo'n Ardui
2144 no kunnen we van
2160 alles meten, zoa
```

☒ Autoscroll No line ending ▼

Labo 7


Pointers: Memory-peek

```
byte tekst[] = { 0xAA, 0xBB, 0xCC, 0xAA, 0xBB, 0xCC,
                 0xAA, 0xBB, 0xCC, 0xAA, 0xBB, 0xCC,
                 0xAA, 0xBB, 0xCC, 0xAA, 0xBB, 0xCC,
                 0xAA, 0xBB, 0xCC, 0xAA, 0xBB, 0xCC,
                 0xAA, 0xBB, 0xCC, 0xAA, 0xBB, 0xCC };
```

*Opnieuw hier: naam van een array
is reeds een pointer*

*Afdrukken in "HEX" kan door dit mee te
geven in de Serial.print:*

Serial.print (foobar, HEX)



```
/dev/ttyUSB1

0 startadres array = 8DE
100 AA BB CC AA BB CC AA BB CC AA BB CC AA BB CC AA
110 BB CC AA BB CC AA BB CC AA BB CC AA BB CC 0 0
120 0 0 23 1 8C 0 BB 0 6A 1 EC 0 CA 0 DE 0
130 20 73 74 61 72 74 61 64 72 65 73 20 61 72 72 61
140 79 20 3D 20 0 D A 0 20 20 0 2D 2D 2D 2D 2D
150 2D 2D 2D 2D 2D 2D 2D 65 69 6E 64 65 20 2D 2D 2D
160 2D 2D 2D 2D 2D 2D 2D 2D 2D 0 6A 1 3D 1 0 0
170 2 58 1 0 0 22 1 0 0 E8 3 0 0 0 0 0
180 0 C5 0 C4 0 C0 0 C1 0 C2 0 C6 0 1 0 0
190 1F 23 0 0 0 0 0 0 0 0 0 0 0 0 0
1A0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1B0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1C0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

☒ Autoscroll
No line ending ▼
```

Labo 7

Functies met Pointers

```
int i[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 };
```

We creëren een int array "i"

```
void roteer(int *p) {
```

We creëren functie die een adres als parameter neemt

```
    int tijdelijk = *p;
    for ( byte n=0; n<9; n++ ) {
        *p = *(p+1);
        p++;
    }
    *p = tijdelijk;
}
```

We stokeren tijdelijk het eerste element van de array
Dit eerste element = doorgegeven geheugenplaats!

De **inhoud** van het **huidige geheugenadres**
maken we gelijk aan
de **inhoud** van het **volgende geheugenadres**
Opgepast *p+1 is niet hetzelfde als *(p+1) !!!

We gaan naar het volgende adres

Tenslotte kleven we het oorspronkelijke eerste element
terug op het einde van de array in het geheugen

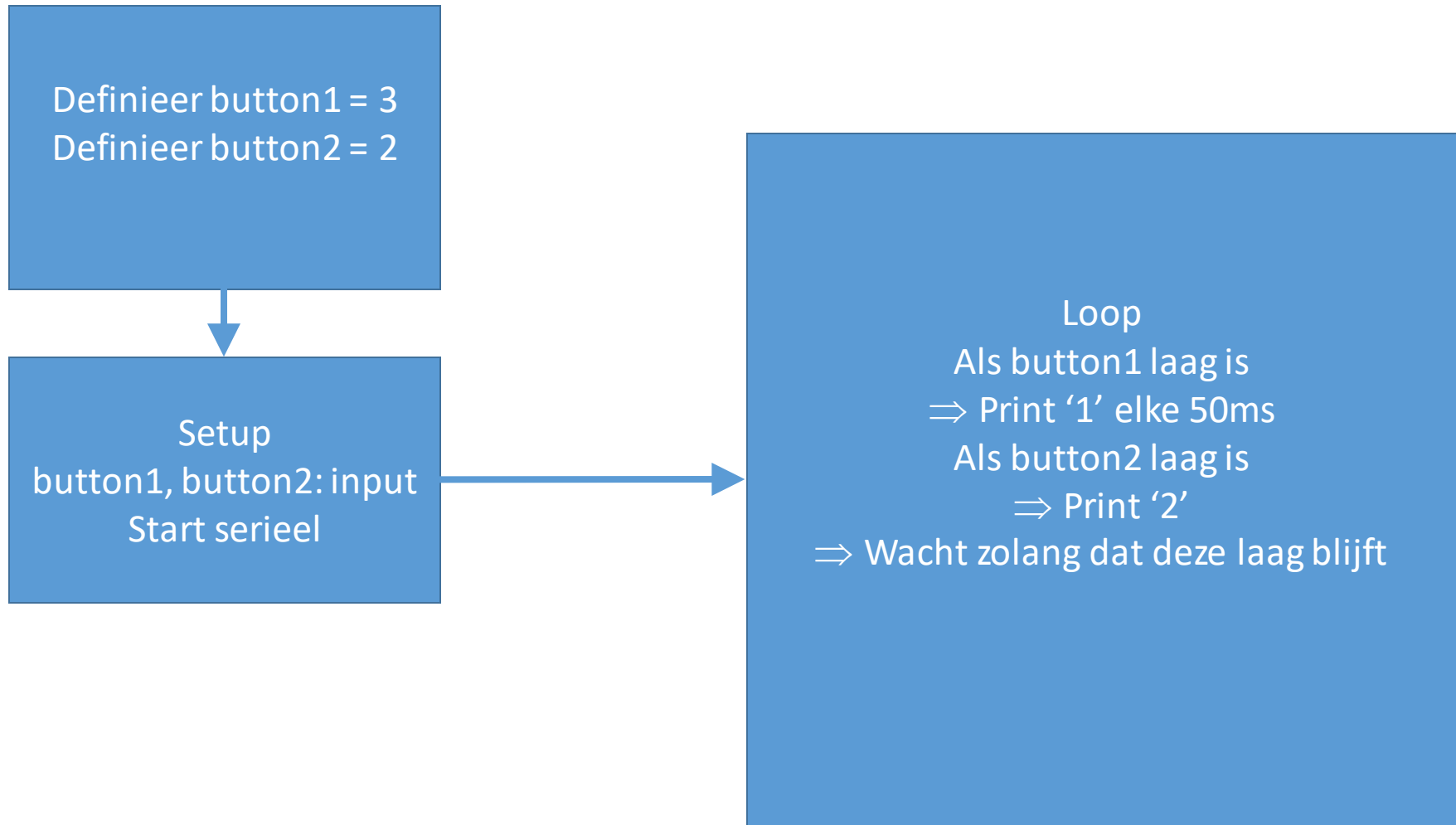
```
roteer(i);
```

In de loop() roepen we onze functie aan met het adres van
het eerste element van de array = de naam van de array

Labo 8



Labo 8: knop-cirkels



Labo 8: lamp aan

Definieer yellow = 13
Definieer green = 12
Definieer red = 11



Setup
Yellow, green, red:
output
Start serieel
Zet seriële timeout om
te wachten op seriële
input



Loop

Definieer een buffer van 50 char tekens
Definieer aantalBytes als byte
Lees alle karakters in tot '\n' in de buffer
Als het aantal karakters in de buffer
groter is dan 0
Als het eerste karakter 'r' en het tweede
karakter in de buffer '='
⇒ Stuur naar de rode led wat het derde
karakter is (converteer van ASCII naar
binaire waarde)
Als het eerste karakter 'g' en het tweede
karakter in de buffer '='
⇒ Stuur naar de groene led wat het
derde karakter is
Als het eerste karakter 'y' en het tweede
karakter in de buffer '='
⇒ Stuur naar de gele led wat het derde
karakter is

Labo 9

LCD Display's

Labo 9

Selectie Menu

Er zijn twee manieren om deze opgave op te lossen:

- functioneel programmeren*
- object georiënteerd programmeren (OOP)*

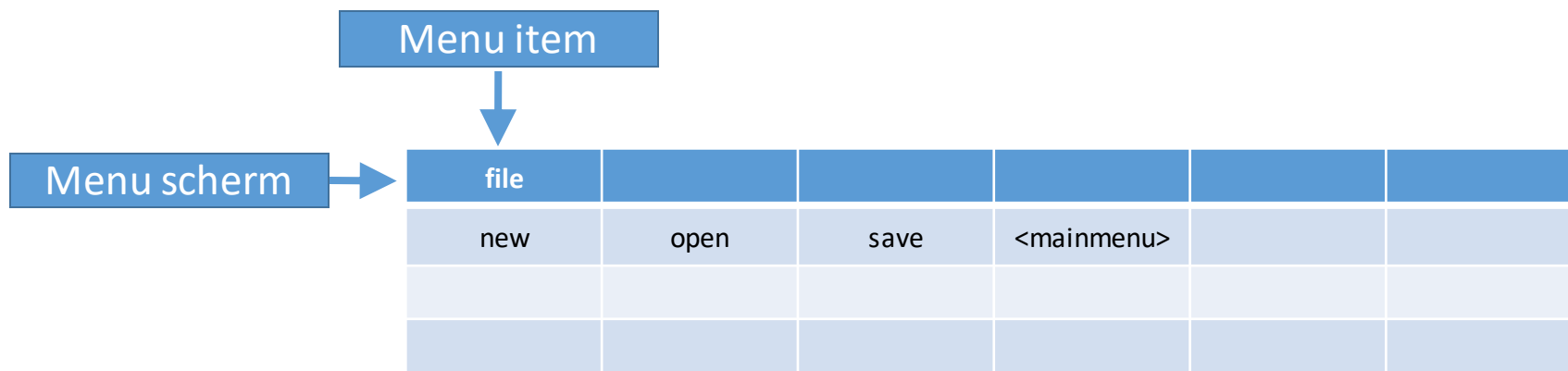
Gezien op het moment van Labo9 OOP nog niet in de theorie besproken was bekijken we hier hoe de opgave met functioneel te programmeren kunnen oplossen.

Labo 9

Selectie Menu

*We gebruiken hiervoor een matrix,
Een truukje gelijkaardig aan de omzet matrix bij de 7-segment display's*

*élk specifiek menu-scherf wordt één rij in deze matrix
En de items van zo'n menu-scherf worden de kolommen in de matrix*



*Met het rij-nummer kunnen we dan een menu-scherf kiezen
Met het kolom-nummer kiezen we een item uit dit menu-scherf
In dit voorbeeld gaan wij bij het kiezen van "file" over naar volgende rij-nummer*

Labo 9

Selectie Menu

voeg éérst de LCD LiquidCristal bibliotheek toe

Initialiseer de LCD: zorg dat pinnen 2 en 3 niet gebruikt worden voor de LCD !
(2 & 3 zijn nodig voor de ISR's ...)



Definiëer een twee dimentionele String matrix met alle tekst van alle menu schermen.

Het best is om alle items aan te vullen met spaties tot 16 characters.
zoniet moet telkens het scherm gewist worden
indien een nieuw item korter zou zijn dan het laatst geschreven item

```
{ "individuele led", "kleur", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " " },
{ "led1", " ", "led2", " ", "led3", " ", "led4", " ", "terug", " ", " ", " ", " ", " ", " " },
{ "rood", " ", "groen", " ", "terug", " ", " ", " ", " ", " ", " ", " ", " ", " " },
{ "uit", " ", "aan", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " " }
```

Definieer een menulevel variabele -> rij-selectie van onze matrix
Definieer een position variabele -> kolom selectie van onze matrix

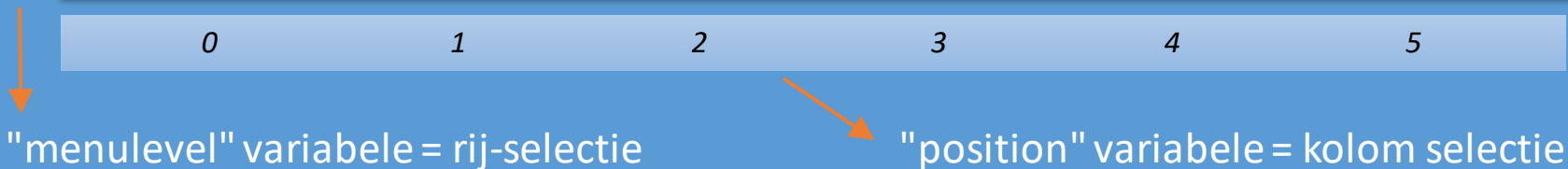


Labo 9

Selectie Menu



0	{ "individuele led", "kleur", " ", " ", " ", " ", " ", " " }
1	{ "led1", " ", "led2", " ", "led3", " ", "led4", " ", "terug", " ", " " }
2	{ "rood", " ", "groen", " ", "terug", " ", " ", " ", " ", " ", " " }
3	{ "uit", " ", "aan", " ", " ", " ", " ", " ", " ", " ", " " }



Definieer een "maximum-position" array met per menu-scherm
het aantal nuttige items in dat menu-scherm



Labo 9

Selectie Menu



0	{ "individuele led", "kleur", " ", " ", " ", " ", " ", " " }
1	{ "led1", " ", "led2", " ", "led3", " ", "led4", " ", "terug", " ", " " }
2	{ "rood", " ", "groen", " ", "terug", " ", " ", " ", " ", " ", " " }
3	{ "uit", " ", "aan", " ", " ", " ", " ", " ", " ", " ", " " }

0

1

2

3

4

5

"menulevel" variabele = rij-selectie

"position" variabele = kolom selectie

Definieer een "maximum-position" array met per menu-scherm
het aantal nuttige items in dat menu-scherm

*Deze array hebben we nodig om te voorkomen dat
als we door een menulevel scrollen we op het einde
een aantal lege strings op het scherm krijgen*

*Dit probleem hebben we niet met OOP
omdat we hier eenvoudig verschillende
objecten kunnen creëren*

Labo 9

Selectie Menu

zo kunnen we de led's kiezen met een index en niet op pinnummer

Definiëer een array voor de led's
Definiëer een variable "ledselect"

Hiermee hebben we een handig instrument voor welke led's we aanspreken:

Definiëer ISR-vlaggen:
volatile bool scrollen
volatile bool select

ledselect = 0 --> led1 (rood)
ledselect = 1 --> led2 (rood)
ledselect = 2 --> led3 (groen)
ledselect = 3 --> led4 (groen)
ledselect = 4 --> led 1+2 (rode leds)
ledselect = 5 --> led 3+4 (groene leds)

Setup:
configureer interrupt voor "scroll-knop" -> ISR = scroll_isr
configureer interrupt voor "select-knop" -> ISR = select_isr

initialiseer LCD scherm als 2 x 16 characters
configureer pinnen LED's als output

Labo 9

Selectie Menu

Functie scroll_isr
Zet de scroll vlag

Hiermee weten we in de loop dat de gebruiker door het menu wil scrollen

Functie select_isr
Zet de select vlag

Hiermee weten we in de loop dat de gebruiker een item uit de menu wil selecteren

Functie menudisplay -> neemt "menulevel" en "position" als parameters

teken een cursor bvb: ">" op plaats (0,0) van het LCD display

Nét na deze cursor, op (1,0) schrijf het actieve menu-item op het scherm

Dit vinden we in de matrix op plaats (menulevel , position)

Op de volgende lijn, op (1,1) schrijf het volgende menu-item op het scherm

Dit vinden we in de matrix op plaats (menulevel , position+1)

Labo 9



Selectie Menu

Loop:

Scroll knop ingedrukt ?

position < max position ?
=> Schuif één item verder op
Wis vlag

Select knop ingedrukt ?

indien menulevel 0 :
position = 0 ? => menulevel = 1
position = 1 ? => menulevel = 2

Select knop ingedrukt ?

indien menulevel 1 :
position = 4 ? => menulevel = 0
position = 0..3 ? =>
ledselect = position
menulevel = 3

Loop (vervolg)

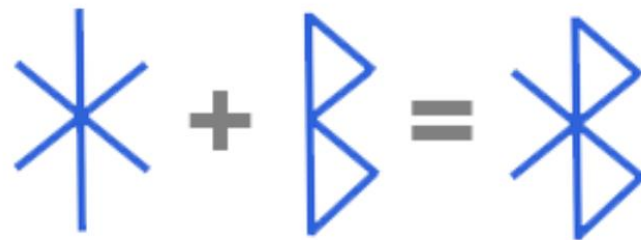
Select knop ingedrukt ?

indien menulevel 2 :
position = 2 ? => menulevel = 0
position = 0..1 ? =>
ledselect = position + 4
menulevel = 3

Select knop ingedrukt ?

indien menulevel 3 :
ledselect < 3 ?
position = 0 => doe led ledselect uit
position = 1 => doe led ledselect aan
ledselect = 4 ?
position = 0 => doe rode leds uit
position = 1 => doe rode leds aan
ledselect = 5 ?
position = 0 => doe groene leds uit
position = 1 => doe groene leds aan
leds klaar ? => menulevel=0, position=0

Labo 10



“H”

“B”



Labo 10

Definieer yellow = 13
Definieer green = 12
Definieer red = 11
Definieer state = 0
Definieer flag = 0



Setup
yellow, green, red:
output
Maak yellow, green, red
laag
Start serieel



Loop
Als er seriële data is
⇒ Status = de waarde van de seriële
buffer
⇒ Zet Flag op 0

Als status = 0
⇒ Maak rood laag
Als flag = 0
⇒ Print "RED: off"
⇒ Zet flag = 1
Anders als status = 1
⇒ Maak rood hoog
Als flag = 0
⇒ Print "RED: on"
⇒ Zet flag = 1

Etc. voor geel & groen

