

Wiki

- [1 Ontwikkelomgeving](#)
 - [Kicad plantuml](#)
 - [Logic analyzer](#)
 - [VScode SVN RedMine](#)
- [2 Microcontrollers](#)
 - [Assembly code](#)
 - [Microcontrollers Vergelijking](#)
- [3 Toolchain](#)
 - [Toolchain 1](#)
 - [Toolchain 2](#)
 - [Toolchain 3](#)
- [4 OSI model](#)
 - [Opdracht Wireshark](#)
- [5 Elektronica](#)
 - [Opdrachten elektronica](#)
- [6 Beveiliging](#)
 - [Opdrachten AES-XTEA-SPECK](#)
- [7 Bussen](#)
 - [Opdrachten I2C](#)
 - [Opdrachten One Wire](#)
 - [Opdrachten SPI&CAN](#)
- [8 FreeRTOS](#)
 - [Opdracht FreeRTOS Task switching](#)
- [9 Eindopdracht](#)
 - [Eindopdracht - fouttentabel](#)

1 Ontwikkelomgeving

Kicad PlantUML

1. Diagram

1. PlantUML Code:

```
@startuml EPD

state "Sein op groen" as SeinOpGroen
state "Sein op oranje" as SeinOpOranje
state "Sein op rood" as SeinOpRood
state "Slagbomen dalen" as SlagbomenDalen
state "Slagbomen gesloten" as SlagbomenGesloten
state "Trein passeert" as TreinPasseert
state "Slagbomen stijgen" as SlagbomenStijgen

[*] --> SeinOpGroen

SeinOpGroen --> SeinOpOranje : Trein nadert
SeinOpOranje --> SeinOpRood : Na 1,5 seconden
SeinOpRood --> SlagbomenDalen

SlagbomenDalen --> SlagbomenGesloten : Slagbomen volledig gesloten
SlagbomenGesloten --> TreinPasseert : Trein rijdt over de overgang
TreinPasseert --> SlagbomenStijgen : Trein voorbij
SlagbomenStijgen --> SeinOpGroen : Slagbomen volledig geopend

@enduml
```

Elektrisch Schema

1. Diagram

Arduino Sketch

1. Code:

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define POT_PIN A0
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

void setup() {
  Serial.begin(9600);

  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("Scherm niet gevonden"));
    while (true);
  }
}
```

```

    }

    display.clearDisplay();
    display.setTextColor(SSD1306_WHITE);
    display.setTextSize(1);
    display.setCursor(0, 0);
    display.println(F("Potentiometer Stand:"));
    display.display();
    delay(2000);
}

void loop() {
    int potValue = analogRead(POT_PIN);

    display.clearDisplay();
    display.setTextSize(2);
    display.setCursor(0, 20);
    display.println(potValue);
    display.display();
}

```

OLED Scherm

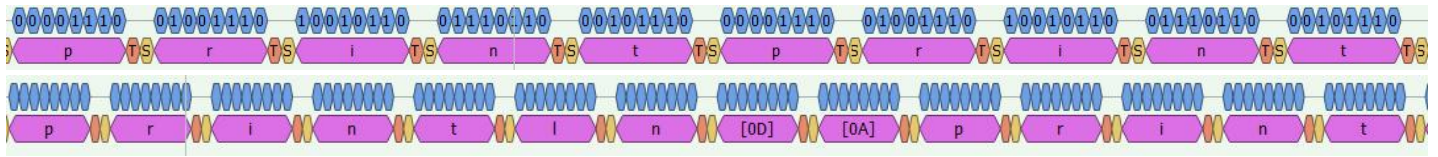
1. Foto
oled.jpeg
-

Files			
EPD.png	27.5 KB	02/21/2025	Mike Ackerschott
statediagram.puml	719 Bytes	02/21/2025	Mike Ackerschott
elektrisch_schema.png	200 KB	02/21/2025	Mike Ackerschott
elektrisch_schema.kicad_sch	24.3 KB	02/21/2025	Mike Ackerschott
statediagram.puml	719 Bytes	02/21/2025	Mike Ackerschott
main.cpp	802 Bytes	02/21/2025	Mike Ackerschott
oled.jpg	253 KB	02/21/2025	Mike Ackerschott

Logic analyzer

Vraag: Wat is het verschil is het gedecodeerde TX-signaal als we in de Arduino-code print of println gebruiken?

Antwoord: Bij het gebruik van println worden 2 extra karakters (bytes) verstuurd, namelijk een carriage return en een line feed



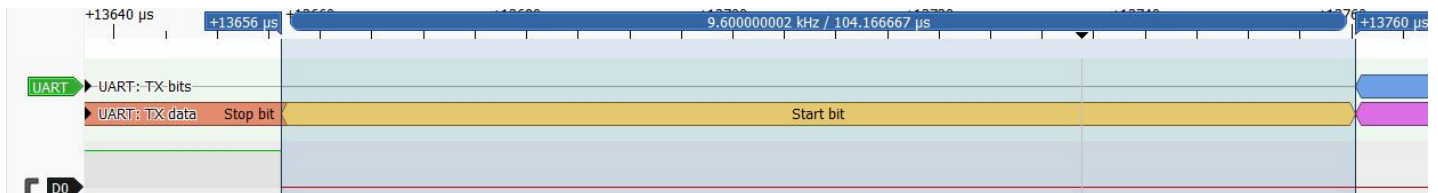
Vraag: Als we de pulsbreedte van de LED-pin opvragen met de - knop blijkt dit nooit exact 10 msec te zijn, maar net iets meer. Waarom klopt het niet?

Antwoord: Het uitvoeren van het omhoog of omlaag zetten van de LED-pin kost tijd, wat na de 10 msec sleep plaatsvindt. Hierdoor is de pulsbreedte altijd hoger dan 10 msec



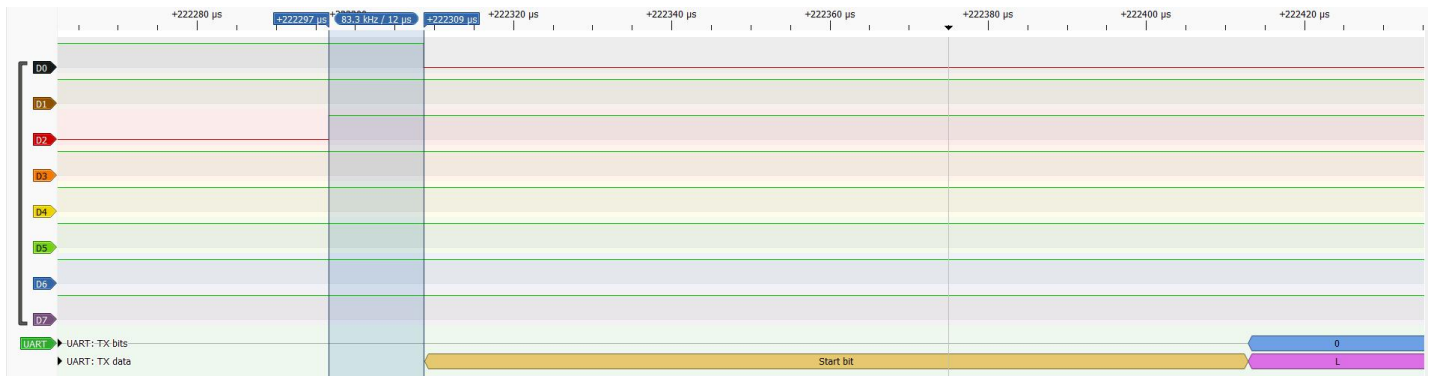
Vraag: Als we de breedte van een startbit opvragen met de - knop, waarom staat er dan bij de frequentie/pulsbreedte niet altijd 9600 Hz, terwijl we dat wel ingesteld hebben in de Arduino? Licht het misschien aan de meting?

Antwoord: De logic analyzer die gebruikt wordt is plusminus 10 euro en kan dus gevoelig zijn voor fouten. Ook de gebruikte Arduino Uno is aan de oudere kant. Ook kan de innerlijke klok van een Arduino Uno R3 lichtelijk achterlopen met de realiteit.



Vraag: Hoe lang duurt het na het omklappen van de LED-pin voordat het verzenden van het eerste bit van de Serial.print is gestart en kun je dit antwoord verklaren?

Antwoord: Dit duurt ongeveer 12 microseconden. Voordat er data verstuurd kan worden moet het eerst in een buffer gestopt worden. Dit kost tijd

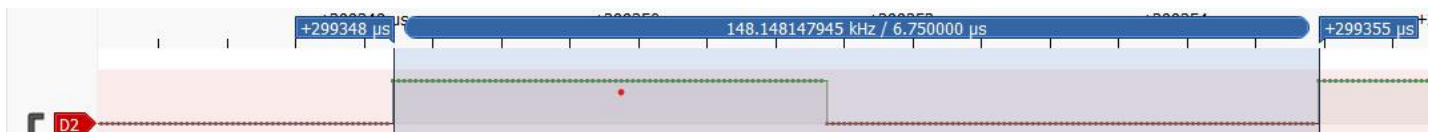


Vraag: Wat gebeurt er als de pulsbreedte van de LED-pin inkorten, terwijl de bitrate van de TX hetzelfde blijft?

Antwoord: Er wordt meer data verzonden, aangezien de delay blocking is.

Vraag: Wat is de maximale knipperfrequentie als we de delay() en de println() weghalen?

Antwoord: 6750 nanoseconden voor een aan-en-uit cyclus



Vraag: Wat is de maximale knipperfrequentie als we gebruik maken van directe IO-poort-manipulatie zoals beschreven in <https://www.arduino.cc/en/Reference/PortManipulation>?

Antwoord: 500 nanoseconden voor een aan-en-uit cyclus



Files

print serial.png	29.4 KB	02/21/2025	Mike Ackerschott
println serial.png	34.5 KB	02/21/2025	Mike Ackerschott
pinflip.png	13.5 KB	02/21/2025	Mike Ackerschott
logic.png	4.87 KB	02/21/2025	Mike Ackerschott
startbit.png	12.7 KB	02/21/2025	Mike Ackerschott
print-after-flip.png	24.7 KB	02/21/2025	Mike Ackerschott
pinflip.png	13.5 KB	02/21/2025	Mike Ackerschott
direct_port_manipulation.png	7.41 KB	02/21/2025	Mike Ackerschott

VScode SVN RedMine

```
#include <Arduino.h>
```

```
#define POT_PIN A0
```

```
void setup() {  
    Serial.begin(9600);  
    while(!Serial){  
  
    }  
}  
  
void loop() {  
    Serial.println(analogRead(POT_PIN));  
}
```

```
from serial import *  
from threading import Thread  
import time  
import sys  
import matplotlib.pyplot as plt  
import numpy as np
```

```
last_received = ''  
data_buffer = []  
max_data_points = 100
```

```
def receiving(ser):  
    global last_received, data_buffer  
    buffer = ''  
  
    while True:  
        try:  
            buffer += ser.read(ser.inWaiting()).decode('utf-8', errors='ignore')  
            if '\n' in buffer:  
                last_received, buffer = buffer.split('\n')[-2:]  
                try:  
                    value = int(last_received)  
                    if 0 <= value <= 1023:  
                        data_buffer.append(value)  
                        if len(data_buffer) > max_data_points:  
                            data_buffer = data_buffer[-max_data_points:]  
                except ValueError:  
                    pass  
            time.sleep(0.1)  
        except SerialException as e:  
            print(f"Serial error: {e}")  
            break
```

```
def plot_data():  
    plt.ion()  
    fig, ax = plt.subplots()  
    ax.set_ylim(0, 1023)  
    ax.set_xlim(0, max_data_points)  
    line, = ax.plot([], [], 'b-', label="Serial Data")  
    ax.set_xlabel('Data Points')  
    ax.set_ylabel('Value')  
    ax.set_title('Real-Time Plot of Serial Data')  
    ax.legend()
```

```
while True:  
    if len(data_buffer) > 0:  
        line.set_xdata(np.arange(len(data_buffer)))  
        line.set_ydata(data_buffer)  
        plt.draw()  
        plt.pause(0.1)  
        time.sleep(0.1)
```

```
def main():
```

```

try:
    ser = Serial(
        port='COM4',
        baudrate=9600,
        bytesize=EIGHTBITS,
        parity=PARITY_NONE,
        stopbits=STOPBITS_ONE,
        timeout=0.1,
        xonxoff=0,
        rtscts=0,
        interCharTimeout=None
    )

    thread = Thread(target=receiving, args=(ser,))
    thread.daemon = True
    thread.start()

    plot_thread = Thread(target=plot_data)
    plot_thread.daemon = True
    plot_thread.start()

    while True:
        time.sleep(1)

except SerialException as e:
    print(f"Error opening serial port: {e}")
finally:
    print("Exiting gracefully...")
    if ser.is_open:
        ser.close()

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        print("\nProgram interrupted by user. Exiting...")
        sys.exit(0)

```

Files

main.cpp	178 Bytes	02/21/2025	Mike Ackerschott
plotter.py	2.42 KB	02/21/2025	Mike Ackerschott

2 Microcontrollers

Assembly code

- Over hoeveel clockcycles per seconde beschikt de Arduino Uno?

De Arduino Uno R3 beschikt over 16.000.000 clockcycles per seconde.

- Wat is de relatie tussen assembly code en clockcycles?

Elke assembly-instructie kost een aantal clockcycles voor de Arduino Uno om uit te voeren. Minder assembly-instructies betekent minder clockcycles en dus snellere uitvoering van het programma.

- Als je programmeert schrijf je geen assembly code. Wie genereert deze code en waarom?

De **compiler** zet de C-code om naar assembly. Dit gebeurt omdat de processor zelf geen C-code begrijpt, maar wel assembly-instructies.

- Assembly code instructies worden uitgevoerd door welk deel van de processor in onderstaande afbeelding?

De **CU** (Control Unit) voert de assembly-instructies uit. Voor berekeningen wordt de **ALU** (Arithmetic Logic Unit) gebruikt.

- Waarom resulteert dezelfde code niet in dezelfde assembly code als je je target verandert (en opnieuw compileert) van bv een Arduino UNO -> NodeMcu ESP8266?

Omdat een andere target een andere hardwarearchitectuur heeft, waardoor de instructies naar de Control Unit anders moeten zijn. Anders was van target wisselen ook niet nodig.

- De code in het bestand firmware.lst is te lezen voor mensen en wordt ook wel genoemd.

De code in het bestand `firmware.lst` is te lezen voor mensen en wordt ook wel **disassembly** genoemd.

- Wordt de code in firmware.lst ook zo geïnterpreteerd door de processor? En zo nee; hoe dan wel?

Nee, de daadwerkelijke code die de processor uitvoert is niet leesbaar voor mensen. Het is in machinecode. De processor leest en interpreteert de machinecode direct.

- Waarom verandert er maar een klein beetje assembly code als je de code in je sketch verandert? Waarvoor dient de rest en waar komt het vandaan?

Omdat een groot deel van de assemblycode wordt gebruikt voor het ondersteunen van de functionaliteit in de **Arduino.h** header, waarmee de hardware van de Arduino aangestuurd wordt om bijvoorbeeld pinflips te kunnen uitvoeren.

- Als ik een println() toevoeg komt er ineens veel meer in de listing te staan. Hoe komt dat?

Serieel printen kost veel clockcycles, waardoor er dus ook veel extra assemblycode gegenereerd wordt. Als er niet geprint wordt in de code, zorgt de compiler er dynamisch voor dat deze library en de benodigde **Serial struct** niet meegecompileerd worden, wat het aantal assemblycode aanzienlijk verlaagt.

1. Microcontrollers

1.1 Inleiding

In dit onderzoek wordt een vergelijking gemaakt tussen verschillende microcontrollers op basis van een vooraf gedefinieerde set criteria. De methodologie die hierbij wordt toegepast is *"Multi-criteria decision making"* (Bonestroo, et al., 2018), waarmee we verschillende eigenschappen van de microcontrollers op een gestructureerde manier kunnen analyseren.

Het onderzoek bestaat uit twee onderdelen:
In het eerste onderdeel wordt er een literatuuronderzoek uitgevoerd om inzicht te krijgen in de eigenschappen van de microcontrollers.
In het tweede onderdeel worden metingen verricht door middel van labonderzoek, waarin specifieke criteria verder worden onderzocht.

1.2 Onderzoeksvraag

De hoofdvraag van dit onderzoek luidt:
“Wat zijn de voor een IoT-toepassing relevante eigenschappen van microcontrollers?”

Deze vraag wordt beantwoord door verschillende microcontrollers met elkaar te vergelijken aan de hand van een set van vooraf gedefinieerde criteria. Aangezien de lijst van mogelijke eigenschappen van een microcontroller zeer uitgebreid kan zijn, wordt in dit onderzoek de focus gelegd op de volgende eigenschappen:

- Vcc
- Power consumption
- Clock
- RAM
- Flash
- #IO
- #PWM
- #Timers
- #UART
- #I2C
- #SPI
- #CAN
- #Wifi
- #ADC
- #DAC
- Crypto
- FPU

Deze eigenschappen zijn geselecteerd omdat ze direct van invloed kunnen zijn op de prestaties en efficiëntie van een microcontroller in een IoT-systeem.

1.3 Voorgescreven microcontrollers

De microcontrollers die in dit onderzoek worden vergeleken, zijn afkomstig van de volgende ontwikkelborden:

- Arduino Uno R3
- ATTiny45
- Lolin MCU V2 met ESP8266
- ESP32
- Raspberry Pi
- STM32F103RBT6

2. Wat zijn de eigenschappen van de microcontrollers?

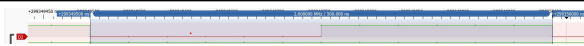

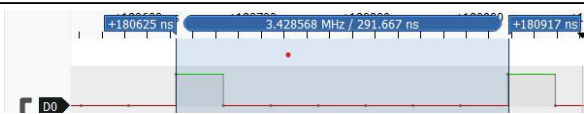
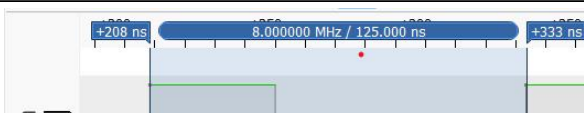
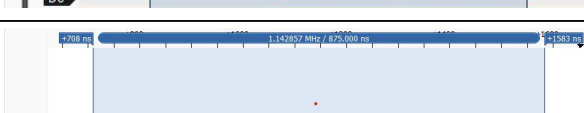
Eigenschap	Arduino Uno R3	ATTiny45	Lolin MCU V2 met ESP8266	ESP32	Raspberry Pi 4 2GB	STM32F103RBT6
Vcc	5V & 3.3V	5V	3.3V	3.3V	3.3V & 5V	2.0 tot 3.6V
Power consumption	98.4 mA	0.1 - 5 mA	50-170 mA	160-260 mA	600-1200 mA	27-36 mA
Clock	16MHz	8MHz	80MHz - 160Mhz	240MHz	4 core 1.5 GHz	72 MHz
RAM	2KB	256 bytes	64Kb	520 KB	2GB	20 KB
Flash	32KB	4KB	4MB	4 MB	Via SD-kaart of USB-poort	128 KB
IO	20	5	16	34	28	64
PWM	6	3	4	28	2	6
Timers	2x 8-bit, 1x 16-bit	2x 8-bit	1x 32-bit	4x 64-bit	4x 32-bit, 1x	3x 16-bit, 1x

					64-bit, ARM Timer gebaseerd op ARM SP804	pwm-timer
UART	Ja	Nee	Ja	Ja, 3 interfaces	6 interfaces	3 interfaces
I2C	Ja	Ja	Ja	Ja, 2 interfaces	6 interfaces	2 interfaces
SPI	Ja	Ja	Ja	Ja, 4 interfaces	5 interfaces	2 interfaces
CAN	Nee, externe modules mogelijk	Nee, externe modules mogelijk	Nee, externe modules mogelijk	Ja, ingebouwde CAN-controller	Nee, externe modules mogelijk	Ja
Wifi	Nee, externe modules mogelijk	Nee, externe modules mogelijk	Ja, 2.4GHz	Ja, 2.4GHz	Ja, 2.4GHz	Nee, externe modules mogelijk
ADC	6	3	1	18	0	2 kanalen
DAC	Nee	Nee	Nee	Ja, 2x 8-bit	Nee	Nee
Crypto	Nee	Nee	Nee	Ja, AES SHA-2 RSA ECC RNG	Nee	Nee
FPU	Nee	Nee	Nee	Ja	Ja	Nee

3. Hoe snel zijn de microcontrollers in praktijk?

Voor het laboratoriumonderzoek wordt er voor een aantal microcontrollers een pinflip uitgevoerd. Hoelang dit duurt wordt gemeten met een logic analyser. Dit word gedaan voor de volgende micocontrollers:

Arduino Uno R3
Lolin MCU V2
ESP32
Raspberry Pi in C
Raspberry Pi in Python

Microcontroller	Pinflip Speed
Arduino Uno R3	
Lolin MCU V2 met ESP8266	
ESP32	
Raspberry Pi 4 2GB in C	
Raspberry Pi 4 2GB in Python	

4. Conclusie

In dit onderzoek is gezocht naar een antwoord op de vraag: "Wat zijn de relevante eigenschappen van microcontrollers voor een IoT-toepassing?"

Uit de resultaten van het onderzoek naar relevante eigenschappen en het laboratoriumonderzoek naar de snelheid van GPIO-pin flips, komt de Raspberry Pi als beste uit de test. De Raspberry Pi biedt ondersteuning voor meerdere programmeertalen en heeft bovendien de snelste GPIO-pin flip, met een snelheid van ongeveer 125 nanoseconden. Voor huidige IoT-toepassingen is de Raspberry Pi dan ook de beste keuze.

Geciteerde werken

Bonestroo, W., Meesters, M., Niels, R., Schagen, J., Henneke, L., & van Turnhout, K. (2018). **ICT Research Methods**. Amsterdam: HBO-i.

- [Arduino Reduce Power Consumption](<https://diyi0t.com/arduino-reduce-power-consumption/>)
- [ATmega328P Datasheet](https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf)

- [Use PWM output with Arduino](https://support.arduino.cc/hc/en-us/articles/9350537961500-Use-PWM-output-with-Arduino)
- [ATTiny45 Datasheet](https://ww1.microchip.com/downloads/en/devicedoc/atmel-2586-avr-8-bit-microcontroller-attiny25-attiny45-attiny85_datasheet.pdf)
 - https://components101.com/sites/default/files/component_datasheet/ESP8266%20NodeMCU%20Datasheet.pdf
 - https://sub.nanona.fi/esp8266/timing-and-ticks.html
 - https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf
 - https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/
 - https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
 - https://www.raspberrypi.com/documentation/computers/processors.html
 - https://datasheets.raspberrypi.com/bcm2711/bcm2711-peripherals.pdf
 - https://docs.rs-online.com/aa25/0900766b81385f70.pdf
 - https://www.st.com/resource/en/datasheet/cd00161566.pdf
 - https://www.jotrin.com/product/parts/STM32F103RBT6

Files			
pi_c.png	4.48 KB	02/20/2025	Mike Ackerschott
nodemcu_esp8266.png	5.02 KB	02/20/2025	Mike Ackerschott
pi_python.png	7.06 KB	02/20/2025	Mike Ackerschott
arduino uno r3.png	7.41 KB	02/20/2025	Mike Ackerschott
esp32.png	5.54 KB	02/20/2025	Mike Ackerschott

3 Toolchain

Toolchain 1

De gebruikte datasheet is [hier](#) te vinden

2.1 Opdracht 1: gebruik toolchain

Voer de stappen zoals die voorgedaan zijn in de les en op de sheets zelfstandig uit binnen de Arduino IDE slechts gebruik makend van avrlib. Zie de “readerToolchain” voor de gebruikte code. Verander de knipperfrequentie en geef aan in de assembly-code en in de hexadecimale dump waar er iets veranderd is.

```
#include <avr/io.h>
#include <util/delay.h>

#define MS_DELAY 500

int main (void) {
    //page 13.2.4 page 61
    DDRB |= _BV(DDB5);

    while(1) {
        //13.2.2 Toggling a pin
        PORTB |= _BV(PORTB5);
        _delay_ms (MS_DELAY);
        PORTB &= ~_BV(PORTB5);
        _delay_ms (MS_DELAY);
    }
}
```

Relevante informatie in de datasheet waren hoofdstuk 13.2.4 voor het zetten van de pinmode (input of output) en hoofdstuk 13.2.2 voor het daadwerkelijk flippen van een port

De enige aanpassingen in de disassembly wanneer de knipperfrequentie van 500 naar 1000ms gewijzigd wordt, zit in de <main> sectie van de disassembly

48 00000080 <main>:	49 80: 25 9a sbi 0x04, 5 ; 4	48 00000080 <main>:	49 80: 25 9a sbi 0x04, 5 ; 4
50 82: 2d 9a sbi 0x05, 5 ; 5		50 82: 2d 9a sbi 0x05, 5 ; 5	
51 84: 2f ef ldi r18, 0xFF ; 255		51 84: 2f ef ldi r18, 0xFF ; 255	
52 86: 83 ed ldi r24, 0xD3 ; 211		52 86: 89 e6 ldi r24, 0x69 ; 105	
53 88: 98 e3 ldi r25, 0x30 ; 48		53 88: 98 e3 ldi r25, 0x18 ; 24	
54 8a: 21 50 subi r18, 0x01 ; 1		54 8a: 21 50 subi r18, 0x01 ; 1	
55 8c: 80 40 sbci r24, 0x00 ; 0		55 8c: 80 40 sbci r24, 0x00 ; 0	
56 8e: 90 40 sbci r25, 0x00 ; 0		56 8e: 90 40 sbci r25, 0x00 ; 0	
57 90: e1 f7 brne .-8 ; 0x8a <main+0xa>		57 90: e1 f7 brne .-8 ; 0x8a <main+0xa>	
58 92: 00 c0 rjmp .+0 ; 0x94 <main+0x14>		58 92: 00 c0 rjmp .+0 ; 0x94 <main+0x14>	
59 94: 00 00 nop		59 94: 00 00 nop	
60 96: 2d 98 cbi 0x05, 5 ; 5		60 96: 2d 98 cbi 0x05, 5 ; 5	
61 98: 2f ef ldi r18, 0xFF ; 255		61 98: 2f ef ldi r18, 0xFF ; 255	
62 9a: 83 ed ldi r24, 0xD3 ; 211		62 9a: 89 e6 ldi r24, 0x69 ; 105	
63 9c: 98 e3 ldi r25, 0x30 ; 48		63 9c: 98 e3 ldi r25, 0x18 ; 24	
64 9e: 21 50 subi r18, 0x01 ; 1		64 9e: 21 50 subi r18, 0x01 ; 1	
65 a0: 80 40 sbci r24, 0x00 ; 0		65 a0: 80 40 sbci r24, 0x00 ; 0	
66 a2: 90 40 sbci r25, 0x00 ; 0		66 a2: 90 40 sbci r25, 0x00 ; 0	
67 a4: e1 f7 brne .-8 ; 0x9e <main+0x1e>		67 a4: e1 f7 brne .-8 ; 0x9e <main+0x1e>	
68 a6: 00 c0 rjmp .+0 ; 0xa8 <main+0x28>		68 a6: 00 c0 rjmp .+0 ; 0xa8 <main+0x28>	
69 a8: 00 00 nop		69 a8: 00 00 nop	
70 aa: eb cf rjmp .-42 ; 0x82 <main+0x2>		70 aa: eb cf rjmp .-42 ; 0x82 <main+0x2>	

1 :10000000C9434000C943E000C943E000C943E0082	1 :10000000C9434000C943E000C943E000C943E0082
2 :10001000C943E000C943E000C943E000C943E0068	2 :10001000C943E000C943E000C943E000C943E0068
3 :10002000C943E000C943E000C943E000C943E0058	3 :10002000C943E000C943E000C943E000C943E0058
4 :10003000C943E000C943E000C943E000C943E0048	4 :10003000C943E000C943E000C943E000C943E0048
5 :10004000C943E000C943E000C943E000C943E0038	5 :10004000C943E000C943E000C943E000C943E0038
6 :10005000C943E000C943E000C943E000C943E0028	6 :10005000C943E000C943E000C943E000C943E0028
7 :10006000C943E000C943E0011241F8CFEFD8E04C	7 :10006000C943E000C943E0011241F8CFEFD8E04C
8 :10007000C943E000C943E000C943E000C943E0000F	8 :10007000C943E000C943E000C943E000C943E0000F
9 :10008000259A2D9A2FEF89E698E1215080409040E3	9 :10008000259A2D9A2FEF89E698E1215080409040E3
10 :10009000E1F70C000002D982FEF89E698E1215080	10 :10009000E1F70C000002D982FEF89E698E1215080
11 :1000A00080409040E1F70C00000EBCFF894FFCF14	11 :1000A00080409040E1F70C00000EBCFF894FFCF14
12 :00000001FF	12 :00000001FF
13 :	13 :

2.2 Opdracht 2: 2 LED's knipperen

Breid de code uit zodat twee LED's (op pin 12 en 13) alternerend knipperen gebruik makend van avrlib.

```
#include <avr/interrupt.h>

#define MS_DELAY_PORTB5 1000
#define MS_DELAY_PORTB4 2000

uint8_t timer_flag_PORTB5 = 0;
uint8_t timer_flag_PORTB4 = 0;

//calculating this value is explained in the datasheet 14.7.2
int8_t calculateOCRA(uint32_t clockspeak, uint32_t prescaler, uint32_t frequency) {
```

```

    return clockspeed / (prescaler * frequency) - 1;
}

void timer0_init() {
    // CTC on timer 0 - datasheet 14.7.2
    TCCR0A |= (1 << WGM01);

    // Set prescaler to 1024. datasheet table 14-9
    TCCR0B |= (1 << CS02) | (1 << CS00);

    // Trigger interrupt once every MS
    OCR0A = calculateOCR0A(16000000, 1024, 1000);

    // Enable Timer0 compare interrupt. datasheet 14.9.6
    TIMSK0 |= (1 << OCIE0A);

    // Enable global interrupts
    sei();
}

void handle_timer(uint16_t *count, uint8_t *flag, uint16_t delay) {
    (*count)++;
    if (*count >= delay) {
        *flag = 1;
        *count = 0;
    }
}

// calls an interrupt when TCNT0 is equal or greater than OCR0A and resets TCNT0 (CTC mode)
ISR(TIMER0_COMPA_vect) {
    static uint16_t count_PORTB5 = 0;
    static uint16_t count_PORTB4 = 0;
    handle_timer(&count_PORTB5, &timer_flag_PORTB5, MS_DELAY_PORTB5);
    handle_timer(&count_PORTB4, &timer_flag_PORTB4, MS_DELAY_PORTB4);
}

int main(void) {
    DDRB |= _BV(DDB5);
    DDRB |= _BV(DDB4);
    timer0_init();

    while (1) {
        if (timer_flag_PORTB5) {
            PORTB ^= _BV(PORTB5);
            timer_flag_PORTB5 = 0;
        }
        if (timer_flag_PORTB4) {
            PORTB ^= _BV(PORTB4);
            timer_flag_PORTB4 = 0;
        }
    }
}

```

Relevante onderdelen uit de datasheet waren 14.7.2 voor het configureren van Timer0 en Table 14-9 voor het configureren van de prescaler

2.3 Opdracht 3: Knipperen op 2 frequenties keuze drukknop

Breid de code uit zodat twee LED's alternerend knipperen met een frequentie van 1 Hz of 10 Hz en de knipperfrequentie wordt beïnvloed door een drukknop (zonder ontddenderen) op Arduino Uno pin 4.

```

#include <avr/interrupt.h>

#define MS_DELAY_1HZ 500
#define MS_DELAY_10HZ 50

volatile uint8_t timer_flag = 0;
uint16_t blink_delay = MS_DELAY_1HZ;

//calculating this value is explained in the datasheet 14.7.2
int8_t calculateOCR0A(uint32_t clockspeed, uint32_t prescaler, uint32_t frequency) {

```



```

    return clockspeed / (prescaler * frequency) - 1;
}

void timer0_init() {
    // CTC on timer 0. datasheet 14.7.2
    TCCR0A |= (1 << WGM01);

    // Set prescaler to 1024. datasheet table 14-9
    TCCR0B |= (1 << CS02) | (1 << CS00);

    // Trigger interrupt once every MS
    OCR0A = calculateOCR0A(16000000, 1024, 1000);

    // Enable Timer0 compare interrupt. datasheet 14.9.6
    TIMSK0 |= (1 << OCIE0A);

    // Enable global interrupts
    sei();
}

void handle_timer(uint16_t *count, uint8_t *flag, uint16_t delay) {
    (*count)++;
    if (*count >= delay) {
        *flag = 1;
        *count = 0;
    }
}

// calls an interrupt when TCNT0 is equal or greater than OCR0A and resets TCNT0 (CTC mode)
ISR(TIMER0_COMPA_vect) {
    static uint16_t count = 0;
    handle_timer(&count, &timer_flag, blink_delay);
}

void setup_pins() {
    DDRB |= (1 << DDB5) | (1 << DDB4);

    // Set pin 4 as input. Datasheet 13.2.1 for input/output. 13.4.9 for registers
    DDRD &= ~(1 << DDD4);
    // PORTD |= (1 << PORTD4);
}

int main(void) {
    setup_pins();
    timer0_init();

    while (1) {
        //check button pressed
        if (!(PIND & (1 << PIND4))) {
            blink_delay = MS_DELAY_10HZ;
        } else {
            blink_delay = MS_DELAY_1HZ;
        }

        //blink
        if (timer_flag) {
            PORTB ^= (1 << PORTB5);
            PORTB ^= (1 << PORTB4);
            timer_flag = 0;
        }
    }
}

```

Relevante onderdelen uit de datasheet waren 13.2.1 om een I/O port op input/output te zetten en 13.4.9 gaf informatie over de registers.

2.4 Opdracht 4: toolchain Platform IO

Herhaal dezelfde opdracht maar dan door gebruik te maken van Platform IO gebruik makend van avrlib. Laat ook weer een assembly- en een hexadecimale output genereren door het framework.

./pio/build/uno/firmware.elf: file format elf32-avr

Disassembly of section .text:

```
00000000 <__vectors>:
 0: 0c 94 34 00      jmp     0x68      ; 0x68 <__ctors_end>
 4: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
 8: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
 c: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
10: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
14: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
18: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
1c: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
20: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
24: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
28: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
2c: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
30: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
34: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
38: 0c 94 53 00      jmp     0xa6      ; 0xa6 <__vector_14>
3c: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
40: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
44: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
48: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
4c: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
50: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
54: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
58: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
5c: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
60: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>
64: 0c 94 51 00      jmp     0xa2      ; 0xa2 <__bad_interrupt>

00000068 <__ctors_end>:
68: 11 24           eor     r1, r1
6a: 1f be           out     0x3f, r1      ; 63
6c: cf ef           ldi     r28, 0xFF      ; 255
6e: d8 e0           ldi     r29, 0x08      ; 8
70: de bf           out     0x3e, r29     ; 62
72: cd bf           out     0x3d, r28     ; 61

00000074 <__do_copy_data>:
74: 11 e0           ldi     r17, 0x01      ; 1
76: a0 e0           ldi     r26, 0x00      ; 0
78: b1 e0           ldi     r27, 0x01      ; 1
7a: ee e5           ldi     r30, 0x5E      ; 94
7c: f1 e0           ldi     r31, 0x01      ; 1
7e: 02 c0           rjmp    .+4           ; 0x84 <__do_copy_data+0x10>
80: 05 90           lpm     r0, Z+
82: 0d 92           st      X+, r0
84: a2 30           cpi     r26, 0x02      ; 2
86: b1 07           cpc     r27, r17
88: d9 f7           brne    .-10          ; 0x80 <__do_copy_data+0xc>

0000008a <__do_clear_bss>:
8a: 21 e0           ldi     r18, 0x01      ; 1
8c: a2 e0           ldi     r26, 0x02      ; 2
8e: b1 e0           ldi     r27, 0x01      ; 1
90: 01 c0           rjmp    .+2           ; 0x94 <.do_clear_bss_start>

00000092 <.do_clear_bss_loop>:
92: 1d 92           st      X+, r1

00000094 <.do_clear_bss_start>:
94: a5 30           cpi     r26, 0x05      ; 5
96: b2 07           cpc     r27, r18
98: e1 f7           brne    .-8           ; 0x92 <.do_clear_bss_loop>
9a: 0e 94 7c 00      call    0xf8          ; 0xf8 <main>
9e: 0c 94 ad 00      jmp     0x15a         ; 0x15a <_exit>

000000a2 <__bad_interrupt>:
a2: 0c 94 00 00      jmp     0            ; 0x0 <__vectors>

000000a6 <__vector_14>:
a6: 1f 92           push    r1
a8: 0f 92           push    r0
```

```

aa: 0f b6      in    r0, 0x3f      ; 63
ac: 0f 92      push   r0
ae: 11 24      eor    r1, r1
b0: 2f 93      push   r18
b2: 3f 93      push   r19
b4: 8f 93      push   r24
b6: 9f 93      push   r25
b8: 20 91 00 01 lds    r18, 0x0100      ; 0x800100 <__data_start>
bc: 30 91 01 01 lds    r19, 0x0101      ; 0x800101 <__data_start+0x1>
c0: 80 91 02 01 lds    r24, 0x0102      ; 0x800102 <__data_end>
c4: 90 91 03 01 lds    r25, 0x0103      ; 0x800103 <__data_end+0x1>
c8: 01 96      adiw   r24, 0x01      ; 1
ca: 90 93 03 01 sts    0x0103, r25      ; 0x800103 <__data_end+0x1>
ce: 80 93 02 01 sts    0x0102, r24      ; 0x800102 <__data_end>
d2: 82 17      cp     r24, r18
d4: 93 07      cpc    r25, r19
d6: 38 f0      brcs   .+14          ; 0xe6 <__vector_14+0x40>
d8: 81 e0      ldi    r24, 0x01      ; 1
da: 80 93 04 01 sts    0x0104, r24      ; 0x800104 <timer_flag>
de: 10 92 03 01 sts    0x0103, r1      ; 0x800103 <__data_end+0x1>
e2: 10 92 02 01 sts    0x0102, r1      ; 0x800102 <__data_end>
e6: 9f 91      pop    r25
e8: 8f 91      pop    r24
ea: 3f 91      pop    r19
ec: 2f 91      pop    r18
ee: 0f 90      pop    r0
f0: 0f be      out    0x3f, r0      ; 63
f2: 0f 90      pop    r0
f4: 1f 90      pop    r1
f6: 18 95      reti

```

000000f8 <main>:

```

f8: 84 b1      in     r24, 0x04      ; 4
fa: 80 63      ori    r24, 0x30      ; 48
fc: 84 b9      out    0x04, r24      ; 4
fe: 54 98      cbi    0x0a, 4          ; 10
100: 5c 9a      sbi    0x0b, 4          ; 11
102: 84 b5      in     r24, 0x24      ; 36
104: 82 60      ori    r24, 0x02      ; 2
106: 84 bd      out    0x24, r24      ; 36
108: 85 b5      in     r24, 0x25      ; 37
10a: 85 60      ori    r24, 0x05      ; 5
10c: 85 bd      out    0x25, r24      ; 37
10e: 8e e0      ldi    r24, 0x0E      ; 14
110: 87 bd      out    0x27, r24      ; 39
112: 80 91 6e 00 lds    r24, 0x006E      ; 0x80006e <__TEXT_REGION_LENGTH__+0x7e006e>
116: 82 60      ori    r24, 0x02      ; 2
118: 80 93 6e 00 sts    0x006E, r24      ; 0x80006e <__TEXT_REGION_LENGTH__+0x7e006e>
11c: 78 94      sei
11e: 44 ef      ldi    r20, 0xF4      ; 244
120: 51 e0      ldi    r21, 0x01      ; 1
122: 62 e3      ldi    r22, 0x32      ; 50
124: 70 e0      ldi    r23, 0x00      ; 0
126: 20 e2      ldi    r18, 0x20      ; 32
128: 90 e1      ldi    r25, 0x10      ; 16
12a: 4c 99      sbic   0x09, 4          ; 9
12c: 11 c0      rjmp   .+34          ; 0x150 <main+0x58>
12e: 70 93 01 01 sts    0x0101, r23      ; 0x800101 <__data_start+0x1>
132: 60 93 00 01 sts    0x0100, r22      ; 0x800100 <__data_start>
136: 80 91 04 01 lds    r24, 0x0104      ; 0x800104 <timer_flag>
13a: 88 23      and    r24, r24
13c: b1 f3      breq   .-20          ; 0x12a <main+0x32>
13e: 85 b1      in     r24, 0x05      ; 5
140: 82 27      eor    r24, r18
142: 85 b9      out    0x05, r24      ; 5
144: 85 b1      in     r24, 0x05      ; 5
146: 89 27      eor    r24, r25
148: 85 b9      out    0x05, r24      ; 5
14a: 10 92 04 01 sts    0x0104, r1      ; 0x800104 <timer_flag>
14e: ed cf      rjmp   .-38          ; 0x12a <main+0x32>
150: 50 93 01 01 sts    0x0101, r21      ; 0x800101 <__data_start+0x1>
154: 40 93 00 01 sts    0x0100, r20      ; 0x800100 <__data_start>
158: ee cf      rjmp   .-36          ; 0x136 <main+0x3e>

```

0000015a <_exit>:

```

15a:      f8 94          cli

0000015c <__stop_program>:
15c:      ff cf          rjmp     .-2          ; 0x15c <__stop_program>

:10000000C9434000C9451000C9451000C94510049
:10001000C9451000C9451000C9451000C9451001C
:10002000C9451000C9451000C9451000C9451000C
:10003000C9451000C9451000C9451000C9453000C945100FA
:10004000C9451000C9451000C9451000C945100EC
:10005000C9451000C9451000C9451000C945100DC
:10006000C9451000C94510011241FBECFEFD8E026
:10007000DEBFCDBF11E0A0E0B1E0EEE5F1E002C0EF
:1000800005900D92A230B107D9F721E0A2E0B1E0CE
:1000900001C01D92A530B207E1F70E947C000C94CC
:1000A000AD000C9400001F920F920FB60F92112416
:1000B0002F933F938F939F932091000130910101E3
:1000C0008091020190910301019690930301809326
:1000D00002018217930738F081E0809304011092A7
:1000E0000301109202019F918F913F912F910F90E8
:1000F0000FB0F901F90189584B1806384B95498F7
:100100005C9A84B5826084BD89EF87BD80916E0062
:10011000826080936E0085B5836085BD789444EFDE
:1001200051E062E370E020E290E14C9911C07093DD
:10013000010160930001809104018823B1F385B12E
:10014000822785B985B1892785B910920401EDCF41
:0E0150005093010140930001EECF894FFCFD1
:02015E00F401AA
:00000001FF

```

2.5 Opdracht 5: Platform IO wel/niet Arduino

Wat gebeurt er als het Arduino Framework de ene keer wel en de andere keer niet wordt geïnclude in de configuratie van Platform IO. Wat is het verschil als het Arduino Framework ingeschakeld wordt? Zijn er verschillen in het compileren? Zijn er verschillen in het geproduceerde hexadecimale bestand voor het uploaden?

In de disassembly en hexadecimale bestanden zit geen verschil. In het compilatieproces zit wel een verschil als je niet direct aangeeft dat het arduino framework gebruikt moet worden. Ten eerste wordt de framework-arduino-avr package niet gebruikt en zoekt het niet naar compatible libraries hiervoor

```
Executing task in folder platformIO zonder arduino: C:\Users\mikea\.platformio\penv\Scripts\platformio.exe run
```

```

Processing uno (platform: atmelavr; board: uno)
-----
Verbose mode can be enabled via `-v, --verbose` option
CONFIGURATION: https://docs.platformio.org/page/boards/atmelavr/uno.html
PLATFORM: Atmel AVR (5.1.0) > Arduino Uno
HARDWARE: ATMEGA328P 16MHz, 2KB RAM, 31.50KB Flash
DEBUG: Current (avr-stub) External (avr-stub, simavr)
PACKAGES:
- toolchain-atmelavr @ 1.70300.191015 (7.3.0)
LDF: Library Dependency Finder -> https://bit.ly/configure-pio-ldf
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 0 compatible libraries
Scanning dependencies...
No dependencies
Building in release mode
Checking size .pio\build\uno\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM: [          ] 0.2% (used 5 bytes from 2048 bytes)
Flash: [          ] 1.1% (used 350 bytes from 32256 bytes)

```

```
Executing task in folder platformIO met arduino: C:\Users\mikea\.platformio\penv\Scripts\platformio.exe run
```

```
Processing uno (platform: atmelavr; board: uno; framework: arduino)
```

```
-----  
Verbose mode can be enabled via `-v, --verbose` option  
CONFIGURATION: https://docs.platformio.org/page/boards/atmelavr/uno.html  
PLATFORM: Atmel AVR (5.1.0) > Arduino Uno  
HARDWARE: ATMEGA328P 16MHz, 2KB RAM, 31.50KB Flash  
DEBUG: Current (avr-stub) External (avr-stub, simavr)  
PACKAGES:  
  - framework-arduino-avr @ 5.2.0  
  - toolchain-atmelavr @ 1.70300.191015 (7.3.0)  
LDF: Library Dependency Finder -> https://bit.ly/configure-pio-ldf  
LDF Modes: Finder ~ chain, Compatibility ~ soft  
Found 5 compatible libraries  
Scanning dependencies...  
No dependencies  
Building in release mode  
Checking size .pio\build\uno\firmware.elf  
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"  
RAM: [          ] 0.2% (used 5 bytes from 2048 bytes)  
Flash: [          ] 1.1% (used 350 bytes from 32256 bytes)
```

verschil compilatie.png

2.6 Opdracht 6: avrdude configuratie

Opdracht: in de datasheet van de atmega328P staan op blz. 255 een aantal verplichte wachttijden voor het programmeren van flash en EEPROM. Waar vind je die terug in configuratie van avrdude?

Bij de minwritedelay en maxwritedelay zijn ze te vinden voor zowel flash en eeprom

```
7836      memory "eeprom"  
7837      paged          = no;  
7838      page_size      = 4;  
7839      size           = 256;  
7840      min_write_delay = 3600;  
7841      max_write_delay = 3600;  
7842      readback_p1    = 0xff;  
7843      readback_p2    = 0xff;  
  
7869      memory "flash"  
7870      paged          = yes;  
7871      size           = 4096;  
7872      page_size      = 64;  
7873      num_pages       = 64;  
7874      min_write_delay = 4500;  
7875      max_write_delay = 4500;  
7876      readback_p1    = 0x00;  
7877      readback_p2    = 0x00;
```

Files

verschil assembly 500 100ms.png	86 KB	02/21/2025	Mike Ackerschott
Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf	8.19 MB	02/21/2025	Mike Ackerschott
verschil hexadecimal 500 100ms.png	50.8 KB	02/21/2025	Mike Ackerschott
EEPROM.png	16.2 KB	02/21/2025	Mike Ackerschott
flash.png	18.7 KB	02/21/2025	Mike Ackerschott

Toolchain 2

De gebruikte datasheet is [hier](#) te vinden

2.1 Opdracht 1: interrupt

Maak een programma waarbij het indrukken van de drukknop de ene of de andere LED laat branden gebruik makend van avrlib. Het indrukken van de drukknop moet verlopen via een interrupt. Ontdenderen van drukknop is ongewenst; het is juist het doel om denderen waar te nemen.

```
#include <avr/io.h>
#include <avr/interrupt.h>

#define LED1 PB5
#define LED2 PB4

// Define what should happen when Low to High transition is detected
ISR(INT0_vect)
{
    PORTB ^= (1 << LED1); // Toggle LED1
    PORTB ^= (1 << LED2); // Toggle LED2
}

void setup_interrupt()
{
    DDRD &= ~(1 << PD2); // Set pin 2 as input. Datasheet 13.2.1 for input/output. 13.4.9 for registers
    PORTD |= (1 << PD2); // Enable internal pull-up resistor on pin 2

    // Pin Low to High triggers an interrupt. 12.2.1
    EICRA |= (1 << ISC01) | (1 << ISC00);

    // Enable external interrupts. 12.2.2
    EIMSK |= (1 << INT0);

    // Enable global interrupts
    sei();
}

void setup_leds()
{
    DDRB |= (1 << LED1) | (1 << LED2); // Set LED pins as output
    PORTB &= ~(1 << LED1) | (1 << LED2); // Turn off both LEDs initially
    PORTB ^= (1 << LED2); // Toggle LED2
}

int main()
{
    setup_leds();
    setup_interrupt();

    while (1)
    {
        // Main loop does nothing, all work is done in ISR
    }
}
```

Relevante informatie uit de datasheet waren hoofdstuk 12.2.1 voor het configureren van een interrupt op een rising edge (wanneer een poort van Low naar High gaat. Hoofdstuk 12.2.2 was relevant voor het daadwerkelijk aanzetten van externe interrupts, zodat de geconfigureerde interrupt ook toegepast werd. Ook was hoofdstuk 13.3.3 gebruikt om te weten welke pin gebruikt kan worden voor een external interrupt input.

Laat in de assemblycode de inhoud zien van de vectortabel en waar deze wijst naar de ISR.

In de code maak ik gebruik van de assembly marker "nop". Deze marker komt voor in de sectie 00000090 <__vector_1>. Dit is ook terug te vinden in de vectors lijst op de 2de regel. Deze 2de regel wijst dus naar de ISR.

```
./pio/build/uno/firmware.elf:      file format elf32-avr
```

```
Disassembly of section .text:
```

```

00000000 <__vectors>:
 0: 0c 94 34 00    jmp     0x68    ; 0x68 <__ctors_end>
 4: 0c 94 40 00    jmp     0x80    ; 0x80 <__vector_1>
 8: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
 c: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
10: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
14: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
18: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
1c: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
20: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
24: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
28: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
2c: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
30: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
34: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
38: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
3c: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
40: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
44: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
48: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
4c: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
50: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
54: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
58: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
5c: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
60: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>
64: 0c 94 3e 00    jmp     0x7c    ; 0x7c <__bad_interrupt>

```

```

00000068 <__ctors_end>:
68: 11 24          eor     r1, r1
6a: 1f be          out     0x3f, r1    ; 63
6c: cf ef          ldi     r28, 0xFF    ; 255
6e: d8 e0          ldi     r29, 0x08    ; 8
70: de bf          out     0x3e, r29    ; 62
72: cd bf          out     0x3d, r28    ; 61
74: 0e 94 58 00    call    0xb0    ; 0xb0 <main>
78: 0c 94 6b 00    jmp     0xd6    ; 0xd6 <_exit>

```

```

0000007c <__bad_interrupt>:
7c: 0c 94 00 00    jmp     0        ; 0x0 <__vectors>

```

```

00000080 <__vector_1>:
80: 1f 92          push    r1
82: 0f 92          push    r0
84: 0f b6          in      r0, 0x3f    ; 63
86: 0f 92          push    r0
88: 11 24          eor     r1, r1
8a: 8f 93          push    r24
8c: 9f 93          push    r25
8e: 00 00          nop
90: 85 b1          in      r24, 0x05    ; 5
92: 91 e0          ldi     r25, 0x01    ; 1
94: 89 27          eor     r24, r25
96: 85 b9          out     0x05, r24    ; 5
98: 85 b1          in      r24, 0x05    ; 5
9a: 92 e0          ldi     r25, 0x02    ; 2
9c: 89 27          eor     r24, r25
9e: 85 b9          out     0x05, r24    ; 5
a0: 00 00          nop
a2: 9f 91          pop     r25
a4: 8f 91          pop     r24
a6: 0f 90          pop     r0
a8: 0f be          out     0x3f, r0    ; 63
aa: 0f 90          pop     r0
ac: 1f 90          pop     r1
ae: 18 95          reti

```

```

000000b0 <main>:
b0: 84 b1          in      r24, 0x04    ; 4
b2: 83 60          ori     r24, 0x03    ; 3
b4: 84 b9          out     0x04, r24    ; 4
b6: 85 b1          in      r24, 0x05    ; 5
b8: 8c 7f          andi    r24, 0xFC    ; 252
ba: 85 b9          out     0x05, r24    ; 5
bc: 85 b1          in      r24, 0x05    ; 5

```

```


be:    92 e0          ldi    r25, 0x02    ; 2
c0:    89 27          eor     r24, r25
c2:    85 b9          out     0x05, r24    ; 5
c4:    52 98          cbi     0x0a, 2      ; 10
c6:    80 91 69 00    lds     r24, 0x0069    ; 0x800069 <__TEXT_REGION_LENGTH__+0x7e0069>
ca:    83 60          ori     r24, 0x03    ; 3
cc:    80 93 69 00    sts     0x0069, r24    ; 0x800069 <__TEXT_REGION_LENGTH__+0x7e0069>
d0:    e8 9a          sbi     0x1d, 0      ; 29
d2:    78 94          sei
d4:    ff cf          rjmp    .-2          ; 0xd4 <main+0x24>

000000d6 <_exit>:
d6:    f8 94          cli

000000d8 <__stop_program>:
d8:    ff cf          rjmp    .-2          ; 0xd8 <__stop_program>

```

Bepaal met de logic analyzer de interrupt service time d.w.z. de totaal benodigde tijd vanaf het waarnemen van de interrupt t/m de terugkeer uit de ISR.

De tijd tussen het drukken van de knop en het aanzetten van de led is 2.5 microseconden


2.2 Opdracht 2: gebruik MCU voor 2 knipperende LED's op Nano

```

#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

#define LED1 PB0
#define LED2 PB1

// ISR for Timer1 compare match
ISR(TIMER1_COMPA_vect)
{
    asm volatile("nop"); // Begin marker
    PORTB ^= (1 << LED1);
    PORTB ^= (1 << LED2);
    asm volatile("nop"); // End marker
}

void setup_timer()
{
    // Set CTC mode (Clear Timer on Compare Match) 15.11.2 & table 15-5
    TCCR1B |= (1 << WGM12);

    // Set prescaler to 1024 and start the timer 15.11.2 - Table 15-6
    TCCR1B |= (1 << CS12) | (1 << CS10);

    // 15.9.2
    // focna = fclock / (2 * prescaler * (1 + fOCnA))
    // prescaler = 1024
    // fOCnA = 1Hz
    // fclock = 16MHz

    // Set compare value for 1Hz used in CTC. When it reaches this value,
    // it will trigger an interrupt and reset the timer to 0. See 15.9.2
    OCR1A = 16000000 / (2 * 1024 * 1);

    // Enable Timer1 compare interrupt 15.7
    // Compare match A interrupt enable 15.11.8
    // Compare the counter value with the compare value in register A (OCR1A)
    TIMSK1 |= (1 << OCIE1A);
}

void setup_leds()
{
    DDRB |= (1 << LED1) | (1 << LED2);
    PORTB &= ~(1 << LED1) | (1 << LED2));
    PORTB ^= (1 << LED2);
}

```



```

}

int main()
{
    setup_leds();
    setup_timer();

    // Enable global interrupts
    sei();

    while (1)
    {
        // Main loop does nothing, all work is done in ISR
    }
}

```

Relevante onderdelen uit de datasheet waren 15.11.2 voor het configureren van CTC mode op Timer1 en het zetten van de prescaler naar 1024.

2.3 Opdracht 3: 2 alternerende LED's met drukknop via ISR

Herhaal de proef met de logic analyzer op een Arduino Nano en meet de ISR-tijd.

```

#include <avr/io.h>
#include <avr/interrupt.h>

#define LED1 PB0
#define LED2 PB1

// Define what should happen when Low to High transition is detected
ISR(INT0_vect)
{
    PORTB ^= (1 << LED1); // Toggle LED1
    PORTB ^= (1 << LED2); // Toggle LED2
}

void setup_interrupt()
{
    DDRD &= ~(1 << PD2); // Set pin 2 as input. Datasheet 13.2.1 for input/output. 13.4.9 for registers
    PORTD |= (1 << PD2); // Enable internal pull-up resistor on pin 2

    // Pin Low to High triggers an interrupt. 12.2.1
    EICRA |= (1 << ISC01) | (1 << ISC00);

    // Enable external interrupts. 12.2.2
    EIMSK |= (1 << INT0);

    // Enable global interrupts
    sei();
}

void setup_leds()
{
    DDRB |= (1 << LED1) | (1 << LED2); // Set LED pins as output
    PORTB &= ~(1 << LED1) | (1 << LED2); // Turn off both LEDs initially
    PORTB ^= (1 << LED2); // Toggle LED2
}

int main()
{
    setup_leds();
    setup_interrupt();

    while (1)
    {
        // Main loop does nothing, all work is done in ISR
    }
}

```

2.4 Opdracht 4: UART

Bestudeer het artikel op <https://www.kanda.com/AVR-C-Code-UART.php> en verander de code zodanig dat de Arduino Nano een ontvangen teken meteen weer verzendt; dit betekent dat het een soort echo-machientje wordt. Er wordt natuurlijk nog steeds gebruik gemaakt van de avrlib.

https://ece-classes.usc.edu/ee459/library/documents/avr_intr_vectors/
https://ww1.microchip.com/downloads/aemDocuments/documents/MCU08/ProductDocuments/DataSheets/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf

Files			
old	14.4 KB	02/21/2025	Mike Ackerschott
logic_interrupt.png	14.2 KB	02/23/2025	Mike Ackerschott
logic nano interrupt.png	14.2 KB	02/23/2025	Mike Ackerschott

Toolchain 3

De gebruikte datasheet is [hier](#) te vinden

Opdracht 1: 2 knipperende LED's

Herhaal de eerdere opgave met 2 alternerend knipperende LED's op een ATTiny45 met de Arduino Uno als programmer via Platform IO.

```
#include <avr/interrupt.h>

#define MS_DELAY_1HZ 500
#define MS_DELAY_10HZ 50

volatile uint8_t timer_flag = 0;
uint16_t blink_delay = MS_DELAY_1HZ;

// datasheet 11.7.2
int8_t calculateOCR0A(uint32_t clockspeed, uint32_t prescaler, uint32_t frequency) {
    return clockspeed / (prescaler * frequency) - 1;
}

void timer0_init() {
    TCCR0A |= (1 << WGM01); // Set CTC mode. 11.5 Data sheet

    OCR0A = calculateOCR0A(8000000, 64, 1000); // Set compare value for 1ms interrupt

    TIMSK |= (1 << OCIE0A); // Enable compare match interrupt. 11.9.7 Data sheet

    TCCR0B |= (1 << CS01) | (1 << CS00); // Set prescaler to 64 and start timer. Table 11-6 Data sheet

    sei(); // Enable global interrupts
}

void handle_timer(uint16_t *count, uint8_t *flag, uint16_t delay) {
    (*count)++;
    if (*count >= delay) {
        *flag = 1;
        *count = 0;
    }
}

ISR(TIMER0_COMPA_vect) {
    static uint16_t count = 0;
    handle_timer(&count, &timer_flag, blink_delay);
}

void setup_pins() {
    DDRB |= (1 << DDB0) | (1 << DDB1); // Set PB0 and PB1 as output for LEDs
    DDRB &= ~(1 << DDB2); // Set PB2 as input for button
    PORTB |= (1 << PORTB2); // Enable pull-up resistor on PB2
    PORTB ^= (1 << PORTB0); // Toggle LED on PB0
}

int main(void) {
    setup_pins();
    timer0_init();

    while (1) {
        // Check if button is pressed
        if (!(PINB & (1 << PINB2))) {
            blink_delay = MS_DELAY_10HZ;
        } else {
            blink_delay = MS_DELAY_1HZ;
        }

        // Blink LEDs
        if (timer_flag) {
            PORTB ^= (1 << PORTB0); // Toggle LED on PB0
            PORTB ^= (1 << PORTB1); // Toggle LED on PB1
            timer_flag = 0;
        }
    }
}
```

```
}
```

Relevante onderdelen uit de datasheet waren 11.7.2 voor het correct instellen van CTC mode, het kiezen van de prescaler en het berekenen van de counter voor in de OCR0A register. Verder gaf tabel 11-6 de nodige informatie voor het configureren van de prescaler en was hoofdstuk 11.9.7 belangrijk om daadwerkelijk Timer0 aan te zetten.

Opdracht 2: 2 knipperende LED's met timer interrupt

Maak een sketch om op basis van een timer interrupt 2 LED's alternerend op 10 Hz te laten knipperen.

```
#include <avr/io.h>
#include <avr/interrupt.h>

volatile uint16_t counter = 0;

// datasheet 11.7.2
int8_t calculateOCR0A(uint32_t clockspeed, uint32_t prescaler, uint32_t frequency) {
    return clockspeed / (prescaler * frequency) - 1;
}

int main(void)
{
    cli(); // Disable global interrupts

    TCCR0A |= (1 << WGM01); // Put timer 0 in CTC mode. 11.7.2 Datasheet

    OCR0A = calculateOCR0A(8000000, 1024, 10000); // Set compare value for 10ms interrupt

    TIMSK |= (1 << OCIE0A); // Enable compare match interrupt. 11.9.7 Data sheet

    TCCR0B |= ((1 << CS02) | (1 << CS00)); // Timer 0 prescaler to 1024. Table 11-6 Datasheet

    DDRB |= (1 << PB0); // Set PortB Pin0 as an output
    DDRB |= (1 << PB1); // Set PortB Pin1 as an output
    PORTB |= (1 << PB0); // Set PortB Pin0 high to turn on LED
    sei(); // Enable global interrupts
    while(1) { } // Don't do anything in main
}

ISR(TIMER0_COMPA_vect) // Interrupt Service Routine
{
    // Use xor to toggle the LED
    PORTB ^= (1 << PB0);
    PORTB ^= (1 << PB1);
}
```

4 OSI model

Opdracht Wireshark

1. Vragen DNS en HTTP

Vraag: Wat zijn DNS berichten?

Antwoord: DNS-berichten zijn berichten die een client verstuurt naar een server om namen van een internet-domein om te zetten naar een IP-adres.

Vraag: Gaat DNS via TCP of UDP?

Antwoord: DNS gebruikt TCP voor Zone transfer en UDP voor name en queries, zowel regulier (primaire) als reverse.

(Referentie: <https://learn.microsoft.com/en-us/troubleshoot/windows-server/networking/dns-works-on-tcp-and-udp>)

Vraag: In welk protocol van laag 4 zitten HTML pakketten verpakt?

Antwoord: In het TCP-protocol. Zie "Transmission Control Protocol" in de screenshot.

1.3 vraag 3.png

Vraag: Waarom wordt de webpagina in zoveel brokken verzonden?

Antwoord: Omdat het maximale aantal bytes in een TCP-bericht maar rond de 1500 is. Hierdoor moeten er meerdere berichten plaatsvinden om grotere websites in te laden.

(Referentie: <https://www.baeldung.com/cs/tcp-max-packet-size>)

Vraag: Waarom worden er extra TCP-connecties gemaakt bij bepaalde sites b.v. <https://nos.nl/>?

Antwoord: Het inladen van websites kost meer TCP-berichten, omdat de inhoud van hun website groter is. Ook wordt er bij nos.nl gebruik gemaakt van HTTPS in plaats van het eerder genoemde voorbeeld dat HTTP gebruikt. Hierdoor moet er via het TCP-protocol extra beveiliging worden gemanaged, wat extra berichten kost. Daarnaast maken sommige websites gebruik van JavaScript-calls naar externe bronnen, wat voor nog meer berichten zorgt.

Vraag: Verifieer de grote hoeveelheid TCP-connecties met de debug-functie. Plak een afdrukje van de debugger in je verslag.

*Antwoord: * 1.3 vraag 6.png

Vraag: Hoe kun je de DNS cache van je laptop opvragen/verwijderen?

Antwoord:

Voor opvragen:

Open PowerShell

Typ het volgende in de opdrachtregel: Get-DnsClientCache

Druk op Enter

Voor verwijderen:

Open een opdrachtprompt

Typ het volgende in de opdrachtregel: ipconfig /flushdns

Druk op Enter

<https://www.digicert.com/nl/faq/dns/how-do-you-flush-a-dns-cache>

Vraag: Is de inhoud van de webpagina zichtbaar in Wireshark van een site zoals http://kbrandt.com/post/alert_status/? Zo ja waarom wel, zo nee waarom niet?

*Antwoord: * Ja, in de http responses kun je de inhoud van de website inzien als je de packet opent als een uncompressed entity body. Dit kan omdat deze website gebruik maakt van http, waardoor berichten plaintext (wel compressed zodat meer data in een bericht past) naar de client wordt gestuurd. Als HTTPS gebruikt zou worden zou de data in de packet encrypted zijn en niet in te lezen zijn

1.3 vraag 8.png

2 Ping van Laptop naar Pi

Vraag: Hoe kan ik pingen van laptop naar Pi?

Antwoord: Door in een commandprompt een "ping" command te sturen naar het IP-adres van de Pi

Vraag: Hoe kan ik het IP-adres van mijn eigen laptop opvragen?

Antwoord: Door in een commandprompt een "ipconfig" command uit te voeren

Vraag: Hoe kan ik het IP-adres van mijn de Pi opvragen?

Antwoord: Via een SSH-verbinding kun je op de Pi "ifconfig" uitvoeren. Als je nog geen SSH-verbinding kan opzetten (omdat je het ipadres niet weet), maar de Pi wel verbinding heeft met de WiFi-router kun je ook vaak inloggen op de router om er zo achter te komen. Dat is mijn persoonlijke voorkeur.

1.3.1 vraag 3.png

Vraag: Wat is de functie van ARP?

Antwoord: Het omzetten van een IP-adres naar een vast uniek MAC-adres

<https://www.fortinet.com/resources/cyberglossary/what-is-arp>

Vraag: Wat is de functie van Time-To-Live in een IP-pakket?

Antwoord: De functie van Time-To-Live op een packet is ervoor zorgen dat een packet automatisch wordt verwijderd wanneer het te vaak een "hop" heeft uitgevoerd (van een machine naar een andere machine verstuurd worden). Hierdoor worden packets die zijn bestemming niet op tijd kunnen verbinden automatisch opgeruimd zodat het netwerk hierdoor niet overbelast raakt

<https://www.techtarget.com/searchnetworking/definition/time-to-live>

Vraag: Wat is het Ethernet-adres van een broadcast?

Antwoord: FF:FF:FF:FF:FF:FF

<https://www.sciencedirect.com/topics/computer-science/broadcast-address>

Vraag: Welke laag is Ethernet?

Antwoord: Ethernet is laag 2 van het OSI-model als het gaat om de data die over ethernet verstuurd wordt. Als in deze vraag de fysieke aansluitingen, zoals een network switch, wordt bedoeld, gaat het om laag 1

https://en.wikipedia.org/wiki/OSI_model

Vraag: Welke laag is ICMP?

Antwoord: ICMP is laag 3

<https://www.ibm.com/docs/en/zos-basic-skills?topic=nll3-internet-control-message-protocol-icmp-other-layer-3-protocols>

Vraag: Hoe kan ik de ARP-cache van de laptop bekijken/flushen?

Antwoord: arp -a in een terminal

Vraag: Hoe kan ik de ARP-cache van de Pi bekijken/flushen?

Antwoord: arp -n in een SSH-verbinding naar de Pi typen

3 Ping van Pi naar Laptop

Vraag: Waarom zie ik na het flushen van de ARP-cache ineens meer pakketten?

Antwoord: Omdat de hardware-adressen opnieuw opgevraagd moeten worden via het netwerk zodat deze weer in de cache komen.

4 Ping van Pi naar Pi

Vraag: Waarom zien we de ping-berichten niet in Wireshark? Wat voor soort netwerkkapparatuur hebben we in de laboratoriumdoos? Hoe kun je nog meer bewijzen dat dit een hub/switch/router/gateway is?

Antwoord: Omdat de switch dynamisch de correcte route kiest via een eigen ARP-tabel. De switch weet dat mijn laptop niet om dat bericht heeft gevraagd en stuurt het dus ook niet naar de laptop.

5 Opzetten van een putty-sessie naar Pi

Vraag: Welke laag is IP?

Antwoord: Laag 3

<https://www.cloudflare.com/learning/network-layer/what-is-the-network-layer/>

Vraag: Welke laag is TCP?

Antwoord: Laag 4

<https://www.a10networks.com/glossary/what-is-layer-4-of-the-osi-model/>

Vraag: Welke laag is SSH?

Antwoord: Laag 7

<https://www.cloudflare.com/learning/access-management/what-is-ssh/>

Vraag: Er zitten meerdere checksums in het pakket. Op welke lagen?

Antwoord:

Laag 2, 3 en 4 maken gebruik van checksums

<https://datatracker.ietf.org/doc/html/rfc1662> - checksum laag 2

<https://datatracker.ietf.org/doc/html/rfc791> - checksum laag 3

<https://www.rfc-editor.org/rfc/rfc793> - checksum laag 4

Vraag: Waarvoor is de SYN-ACKSYN-ACK bedoeld?

Antwoord: Om een 3-way handshake op te zetten tussen de client en server. de server geeft met een syn aan te willen verbinden. de server stuurt een synack terug om te laten weten dat dit mag en de client stuurt een ack terug om te laten weten dat de syn-ack is aangekomen. Hierdoor is het voor zowel client als server duidelijk dat de connectie goed is

<https://www.guru99.com/nl/tcp-3-way-handshake.html>

Vraag: Waarom vertoont het veld SEQ zulke rare sprongen in de eerste 5 pakketten? De nummers zijn niet opeenvolgend 1,2,3 etc., maar eerder 3, 40, 281 etc.

Antwoord: gedurende de SYN-fase stuurt mijn client een bericht met een willekeurige SEQ (op windows lijkt dit default 0). De Pi ontvangt dit en geeft dit aan via een SYN-ACK waarbij de ACK op 1 staat. Dan geeft mijn laptop aan dat het nu data wilt via een ACK door de SEQ op 1 te zetten. Vanuit daar wordt de SEQ gezet op hoeveel bytes verzonden zijn

Vraag: Waarom vertoont het veld ACK precies dezelfde rare sprongen in de eerste 5 pakketten? De nummers zijn niet opeenvolgend 1,2,3 etc.

Antwoord: Het acknowledgmentnummer (ACK) in een TCP-pakket geeft aan welk volgend byte-nummer de ontvanger verwacht. Dit hangt dus samen met hoeveel data er is verstuurd waardoor er een zichtbaar patroon is in de waardes van SEQ en ACK

Vraag: Welke poortnummers worden gebruikt?

Antwoord: Voor de server wordt standaard poort 22 gebruikt voor SSH-connecties. Voor clients is dit willekeurig tussen 1024 en 32,767. In mijn geval is het Source Port: 22574

Vraag: Wat gebeurt er als je een tijdje niets intypt? Op welke manier wordt ervoor gezorgd dat de verbinding niet wordt verbroken?

Antwoord: Putty by default heeft geen keep alive messages dus in principe gebeurt er niks. Waar deze vraag waarschijnlijk naar doelt is het toepassen van Keep-alive messages om de connectie in stand te houden en te checken of de connectie nog bestaat. Zie [dit artikel](#)

Vraag: Wat gebeurt er als de verbinding bewust wordt verbroken door de PuTTY-sessie af te sluiten?

Antwoord: Er wordt vanuit de client een FIN, ACK bericht gestuurd naar de PI. de PI ontvangt dit en stuurt een FIN, ACK terug naar de laptop, waarna de SSH-sessie aan beide kanten stopt

1.3.4 vraag 10.png

Vraag: Wat gebeurt er als ik een tweede PuTTY-sessie start? Hoe kan het dat de juiste ingetypte commando's uiteindelijk output geven in het juiste PuTTY-scherm?

Antwoord: De 2de sessie vindt plaats op een andere poort (22820 en 22821). Hierdoor wordt de afhandeling van beide SSH-sessies apart gehouden.

1.3.4 vraag 11.png

6 Leesbaarheid berichten

Vraag: Als je telkens een vaste letter intypt binnen een putty-sessie, waarom wordt er steeds een veel groter en ook steeds een ander pakket op laag 7 verzonden?

Antwoord: Het bericht wordt niet groter. voor elk teken wordt er een encrypted package van 64 bytes gestuurd naar de PI en wordt er vanuit de PI dan weer een Encrypted packet van 64 bytes teruggestuurd. Encryptie zorgt ervoor dat eenzelfde karakter er in een packet heel anders uit kan zien.
1.3.5 vraag 1.png

Vraag: Worden steeds ieder teken op de terminal als los pakket verzonden of gaan er soms ook meerdere tekens in één pakket? Waarom worden ze dan per teken in losse pakketten verzonden?

Antwoord: Elk getypt karakter vanuit de laptop naar de pi zorgt voor een nieuwe packet dat verstuurd wordt naar de Pi. Output van een commando, zoals ls, wordt in één (of meerdere als de output niet past in één packet) verstuurd naar de client.

7 Filezilla

Vraag: Waarin komt het dataverkeer bij gebruik van Filezilla i.p.v. putty overeen c.q. verschilt het?

Antwoord: Filezilla gebruikt ook de SYN -> SYN, ACK -> ACK structuur voor het opzetten van een connectie. Filezilla gebruikt FTP in plaats van SSHv2 voor het opvragen van de bestanden op de Pi. Filezilla gebruikt poort 21 i.p.v 22
1.4 vraag 1.png

Opdrachten elektronica

multimeter

Meet de spanning op de 3.3V- en 5V-aansluiting van alle aanwezige Arduino's en NodeMCU en noteer deze in het verslag. Waarom verschillen bepaalde spanning onderling niet en andere onderling wel van elkaar.

nano3v3.jpg
Nano 3.3V
nano5v.jpg
Nano 5V
nodemcu_3v3.jpg
NodeMCU 3.3V
uno3v3.jpg
Uno 3.3V
uno5v.jpg
Uno 5V

Vraag: Waarom verschillen bepaalde spanning onderling niet en andere onderling wel van elkaar?:

Antwoord: De microcontrollers hebben verschillende voltage-regelaars. Deze zijn anders per microcontroller en kunnen dus voor verschillen zorgen. Ook kan het zijn dat de microcontrollers anders geijkt zijn

Meet de weerstandswaarde van een paar weerstanden.

10kohm.jpg
10K ohm
220ohm.jpg
220 ohm
330kohm.jpg
330K ohm

Vraag: Vergelijk de gemeten waarden met de opgegeven waarde. Wat valt op?

Antwoord: De gemeten waarden zijn altijd lager dan de opgegeven waarden

DAC

Vraag: Maak een kleine sketch om zo snel als de DAC-communicatie maar aan kan digitale samples weg te schrijven vanuit een Arduino sketch naar de DAC via I2C. Meet m.b.v. de logic analyzer (door een pin te toggelen aan het begin en aan het einde van het versturen van een sample naar de DAC) hoe lang het duurt voordat één sample in de DAC staat.

```
#include <Wire.h>
#include <Adafruit_MCP4725.h>
Adafruit_MCP4725 dac;
void setup(void)
{
  Serial.begin(9600);
  Serial.println("Hello!");
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);
  dac.begin(0x60);
}

void loop(void)
{
  PINB = _BV(PINB5);
  PINB = _BV(PINB5);
  dac.setVoltage(2025, false);
}
```

DAC_speed.png

Antwoord: 163.375 microseconden

Vraag: Maak een sketch om het analoge signaal uit de DAC opnieuw te samplen met een instelbare frequentie op een ander board.

Antwoord: Sender:

```
#include <Wire.h>
#include <Adafruit_MCP4725.h>
```

```

Adafruit_MCP4725 dac;

void setup(void)
{
  Serial.begin(9600);
  dac.begin(0x60);
}

void loop(void)
{
  uint32_t counter;
  for (counter = 0; counter < 4095; counter++)
  {
    dac.setVoltage(counter, false);
  }

  for (counter = 4095; counter > 0; counter--)
  {
    dac.setVoltage(counter, false);
  }
}

```

Receiver:

```

unsigned long interval = 0;
unsigned long previousMillis = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  if (Serial.available() > 0) {
    String input = Serial.readStringUntil('\n');
    if (input.length() > 0 && input.toInt() > 0) {
      interval = input.toInt();
    }
  }

  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    Serial.println(analogRead(A0));
  }
}

```

Vraag: Begin met een zo hoog mogelijke samplefrequentie en maak een afdruk van de plot.

Antwoord: serial plotter normal.png

Vraag: Verlaag vervolgens de samplefrequentie totdat er vreemde tekeningen uit de plot komen. Maak daarvan een afdruk en probeer de plot te verklaren.

Antwoord: serial plotter 400ms.png

De vreemde tekeningen zijn te verklaren doordat de samplefrequentie zo laag is, dat een groot deel van de "zaagtand" al op de analoge poort is gezet voordat er een meting gebeurt aan de ontvangstkant. Hierdoor is de zaagtand niet meer zichtbaar in de plotter en lijkt het op willekeurige lijntjes.

Week 6 Beveiliging

Opdrachten AES-XTEA-SPECK

Inhoudsopgave

1 Inleiding

In IoT-toepassingen gebruikt men vaak encryptie. Het is belangrijk om te weten welke criteria hiervoor gelden. Dit onderzoek richt zich op criteria voor het gebruik van symmetrische cryptografische algoritmen. Eerst komt de onderzoeksvraag aan bod, gevolgd door de onderzoeksmethode. Daarna worden de criteria uitgelegd en de resultaten beschreven.

2 Onderzoeksvraag

Wat zijn de snelheden van verschillende symmetrische encryptiealgoritmen?

3 Onderzoeksmethode

Om tot een concreet antwoord te komen op de hoofdvraag, wordt er gebruik gemaakt van de ICT Research Methods kaarten (Methods - ICT Research Methods, 2020). Hierbij is gekozen voor het toepassen van [Multi criteria decision making](#) om voor elke encryptiemethode verschillende criteria op te sommen. Verder wordt er gebruik gemaakt van [Component test](#) om alle encryptiemethoden individueel te testen op snelheid.

4 Criteria

Criteria	Rationale		
Blokgrootte	Bepaalt de grootte van gegevensblokken die het algoritme verwerkt.		
Sleutelgrootte	Geeft aan hoe sterk de encryptiesleutel is.		
Maximale snelheid in pakketten/s	Meet de verwerkingssnelheid in pakketten per seconde.		
Maximale snelheid in bytes/s	Meet de verwerkingssnelheid in bytes per seconde.		
Programmageheugen	Geeft aan hoeveel geheugen het programma gebruikt.		
RAM-geheugen	Geeft aan hoeveel werkgeheugen het algoritme nodig heeft.		
Beveiligingsniveau	Beoordeelt de sterkte van de beveiliging.		
Cross-platform mogelijkheid	Bepaalt of versleutelde data op verschillende platforms interpreteerbaar is.		

Criteria	Speck	XTEA	AES
Blokgrootte	32, 48, 64, 96 of 128 bits.	64 bits	128 bits
Sleutelgrootte	64, 72, 96, 128, 144, 192 of 256 bits	128 bits	128, 192 of 256 bits
Maximale snelheid in pakketten/s	6477 pakketten/s	1633 pakketten/s	1779 pakketten/s
Maximale snelheid in bytes/s	1 pakket = 16 bytes, dus 103.632 bytes/s	1 pakket = 8 bytes, dus 13064 bytes/s	1 pakket = 16 bytes, dus 28.464 bytes/s
Programmageheugen	2416 bytes	942 bytes	3886 bytes
RAM-geheugen	346 bytes	33 bytes	276 bytes
Beveiligingsniveau	Vanaf 2018 is er geen succesvolle aanval bekend op de volledige versie van Speck, ongeacht de variant. Door de interesse in Simon en Speck zijn er ongeveer 70 cryptografische analyses over gepubliceerd. Zoals gebruikelijk bij iteratieve versleutelingen zijn varianten met minder rondes wel succesvol aangevallen. De beste bekende aanvallen op Speck binnen het standaard aanvalmodel (CPA/CCA met onbekende sleutel) zijn gebaseerd op differentiële cryptanalyse. Deze aanvallen kunnen ongeveer 70–75% van de rondes van de meeste varianten doorbreken.	In 2004 is een related-key differential attack op 27 van de 64 rondes van XTEA geslaagd. In 2009 is dit omhoog gegaan naar 35 via related-key rectangle attack. XTEA met 64 rondes is dus nog niet volledig gekraakt	het AES-algoritme (met sleutellengtes van 128, 192 en 256 bits) zijn voldoende om geclassificeerde informatie tot het niveau SECRET te beschermen. Informatie met het niveau TOP SECRET vereist het gebruik van de sleutellengtes 192 of 256. In maart 2016 presenteerden C. Ashokkumar, Ravi Prakash Giri en Bernard Menezes een side-channel attack op AES-implementaties die de volledige 128-bits AES-sleutel kan herstellen met slechts 6-7 blokken plaintext/ciphertext. Dit is een aanzienlijke verbetering ten opzichte van eerdere werken die tussen de 100 en een miljoen

			encrypties vereisen. De voorgestelde aanval vereist standaard gebruikersrechten en sleutelherstelalgoritmes die binnen een minuut draaien.
Cross-platform mogelijkheid	Ja, zolang dezelfde sleutel gebruikt wordt	Ja, zolang dezelfde sleutel gebruikt wordt.	Ja, zolang dezelfde sleutel gebruikt wordt.

5 Bronnen

Wikipedia contributors. (2025, 12 maart). Advanced Encryption Standard. Wikipedia. https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

Wikipedia contributors. (2025a, januari 23). XTEA. Wikipedia. <https://en.wikipedia.org/wiki/XTEA>

Wikipedia contributors. (2023, 11 december). Speck (cipher). Wikipedia. [https://en.wikipedia.org/wiki/Speck_\(cipher\)](https://en.wikipedia.org/wiki/Speck_(cipher))

XTEA (eXtended TEA). (z.d.). <https://asecuritysite.com/encryption/xtea>

SPECK. (z.d.). <https://asecuritysite.com/encryption/speck>

AES. (z.d.). <https://asecuritysite.com/encryption/aes>

Hbo-I. (z.d.). ICT Research Methods — Methods Pack for Research in ICT. ICT Research Methods. <https://ictresearchmethods.nl/>

Opdrachten I2C

1 opgave poortscanner

Verbindt een I2C-module uit de laboratoriumdoos aan een microcontroller b.v. de 10-DOF-sensor of wat er maar in het bakje 'I2C multi' zit aan een Arduino.

Vraag: • Zoek met een I2C-poortscanner uit welke I2C-adressen beschikbaar zijn.

Antwoord:

```
#include <Wire.h>
#include <Arduino.h>
void setup()
{
    Wire.begin();
    Serial.begin(9600);
    while (!Serial);
    Serial.println("\nI2C Scanner");
}

void loop()
{
    byte error, address;
    int nDevices;
    Serial.println("Scanning...");
    nDevices = 0;

    for(address = 1; address < 127; address++ )
    {
        Wire.beginTransmission(address);
        error = Wire.endTransmission();
        if (error == 0)
        {
            Serial.print("I2C device found at address 0x");
            if (address<16)
                Serial.print("0");
            Serial.print(address,HEX);
            Serial.println(" !");
            nDevices++;
        }
        else if (error==4)
        {
            Serial.print("Unknown error at address 0x");
            if (address<16)
                Serial.print("0");
            Serial.println(address,HEX);
        }
    }
    if (nDevices == 0)
        Serial.println("No I2C devices found\n");
    else
        Serial.println("done\n");
    delay(5000);
}
```

Voor de sensor die ik gebruik (VL53L0X) is het beschikbare adres 0x29

Vraag: Kijk meteen met de Logic Analyzer

Antwoord:

I2C_portscanning.png

2 I2C opgave sensor

Vraag: Ga met een sketch de sensorwaarden uitlezen en bestudeer het I2C-verkeer met de logic analyzer.

Antwoord:

```
#include "Adafruit_VL53L0X.h"
```



```
#include <Arduino.h>

Adafruit_VL53L0X lox = Adafruit_VL53L0X();
VL53L0X_RangingMeasurementData_t measure;

void setup() {
  Serial.begin(9600);

  // wait until serial port opens for native USB devices
  while (! Serial) {
    delay(1);
  }

  pinMode(13, OUTPUT);

  Serial.println("Adafruit VL53L0X test");
  if (!lox.begin()) {
    Serial.println(F("Failed to boot VL53L0X"));
    while(1);
  }
}

void loop() {
  digitalWrite(13, HIGH);
  digitalWrite(13, LOW);

  // if(lox.isRangeComplete)
  lox.getRangingMeasurement(&measure);
}
```

I2C_sensor.png

Vraag: • Maak elektrisch schema

Antwoord:

electric scheme i2c sensor.png

Vraag: • Hoeveel tijd kost het ophalen van één enkele meetwaarde?

Antwoord: 2910 microseconds

Vraag: • Hoeveel meting per seconde kun je met jouw sensor waarnemen?

Antwoord: 343.5 metingen per seconde

3 OLED

Vraag: Sluit de OLED aan op een microcontroller (0.9 inch OLED van Tinytronics). Zoek een klein testprogrammaatje en bestudeer het dataverkeer tussen microcontroller en OLED.

Antwoord:

```
#include <Arduino.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 32 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
// The pins for I2C are defined by the Wire-library.
// On an arduino UNO:          A4(SDA), A5(SCL)
// On an arduino MEGA 2560:  20(SDA),  21(SCL)
// On an arduino LEONARDO:   2(SDA),   3(SCL), ...
#define OLED_RESET -1        // Reset pin # (or -1 if sharing Arduino reset pin)
#define SCREEN_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C for 128x32
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

void setup()
{
  Serial.begin(9600);
```

```

// SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS))
{
    Serial.println(F("SSD1306 allocation failed"));
    for (;;)
        ; // Don't proceed, loop forever
}

// Show initial display buffer contents on the screen --
// the library initializes this with an Adafruit splash screen.
display.display();
delay(2000); // Pause for 2 seconds

// Clear the buffer
display.clearDisplay();

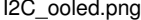
pinMode(13, OUTPUT);
digitalWrite(13, LOW);

display.setCursor(0, 0);
display.setTextSize(1);
display.setTextColor(SSD1306_WHITE);
}

void loop()
{
    PORTB ^= 1 << 5;
    PORTB ^= 1 << 5;
    display.println(F("Hello, world!"));
    display.display();
    display.setCursor(0, 0);
}

```

Vraag: Hoeveel tijd kost het verzenden van één regel tekst op het display in totaal?

Antwoord: 22.3 milliseconden om text naar het display te schrijven en de cursor weer op 0,0 te zetten


Opdrachten One Wire

Sluit de DHT11 temperatuursensor uit de componentendoos aan op de Arduino.

Maak gebruik van de Arduino library voor deze sensor en bestudeer de gegevensuitwisseling met de logic analyzer.

Vraag: Hoe lang duurt de uitwisseling van één meetwaarde?

Antwoord: Een meting waarbij de temperatuur en de luchtvochtigheid wordt gemeten, duurt 161.7 milliseconden
DHT11 speed.png

Vraag: Hoeveel metingen per seconde kan deze sensor hanteren?

Antwoord: 6 metingen per seconde

Vraag: Hoe vaak adviseer je om de sensor een meetwaarde te laten bepalen?

Antwoord: Er zijn niet veel scenario's waarbij het weten van de temperatuur of luchtvochtigheid vaker dan 6 keer per seconde moet gebeuren. In een normale gebruiksomgeving zou ik kiezen voor 1 keer per minuut, wanneer je bijvoorbeeld je CV dynamisch wilt aansturen

Vraag: Is er sprake van een checksum?

Antwoord: Ja, in de library wordt de opgevraagde data gecheckt via een checksum
checksum.png

```
#include <DHT11.h>
#include <Arduino.h>

DHT11 dht11(2);

void setup() {
    dht11.setDelay(140);
    pinMode(13, OUTPUT);
    digitalWrite(13, LOW);
}

void loop() {
    PORTB ^= (1 << 5);
    PORTB ^= (1 << 5);

    int temperature = 0;
    int humidity = 0;

    int result = dht11.readTemperatureHumidity(temperature, humidity);

    while (result != 0) {
        result = dht11.readTemperatureHumidity(temperature, humidity);
    }
}
```

DHT_scheme.png

Opdrachten SPI&CAN

SPI zend

Vraag: Maak/pak een klein voorbeeld uit de library waarbij met enige regelmaat (vanwege het 'vangen' met de analyzer) een bericht wordt verzonden op de ene microcontroller en ontvangen op een andere microcontroller om daar via Serial te worden afgedrukt.

Antwoord:

electric__schema.png

CAN_SEND:

```
#include <SPI.h>
#include <mcp2515.h>

struct can_frame canMsg1;
// struct can_frame canMsg2;
MCP2515 mcp2515(10);

void setup() {
    canMsg1.can_id = 0x0F6;
    canMsg1.can_dlc = 8;
    canMsg1.data[0] = 0x8E;
    canMsg1.data[1] = 0x87;
    canMsg1.data[2] = 0x32;
    canMsg1.data[3] = 0xFA;
    canMsg1.data[4] = 0x26;
    canMsg1.data[5] = 0x8E;
    canMsg1.data[6] = 0xBE;
    canMsg1.data[7] = 0x86;

    // canMsg2.can_id = 0x036;
    // canMsg2.can_dlc = 8;
    // canMsg2.data[0] = 0x0E;
    // canMsg2.data[1] = 0x00;
    // canMsg2.data[2] = 0x00;
    // canMsg2.data[3] = 0x08;
    // canMsg2.data[4] = 0x01;
    // canMsg2.data[5] = 0x00;
    // canMsg2.data[6] = 0x00;
    // canMsg2.data[7] = 0xA0;

    mcp2515.reset();
    mcp2515.setBaudrate(CAN_125KBPS);
    mcp2515.setNormalMode();

    pinMode(3, OUTPUT);
    digitalWrite(3, LOW);

    // Serial.println("Example: Write to CAN");
}

void loop() {
    PIND |= (1 << PIND3);
    PIND |= (1 << PIND3);

    while(mcp2515.sendMessage(&canMsg1) != MCP2515::ERROR_OK) {
        // Serial.println("Error sending message");
    }
    // mcp2515.sendMessage(&canMsg2);

    // Serial.println("Messages sent");
}
```

CAN_READ

```
#include <SPI.h>
```

```
#include <mcp2515.h>

struct can_frame canMsg;
MCP2515 mcp2515(10);

void setup() {
  Serial.begin(115200);

  mcp2515.reset();
  mcp2515.setBaudrate(CAN_125KBPS);
  mcp2515.setNormalMode();

  Serial.println("----- CAN Read -----");
  Serial.println("ID   DLC   DATA");
}

void loop() {
  if (mcp2515.readMessage(&canMsg) == MCP2515::ERROR_OK) {
    Serial.print(canMsg.can_id, HEX); // print ID
    Serial.print(" ");
    Serial.print(canMsg.can_dlc, HEX); // print DLC
    Serial.print(" ");

    for (int i = 0; i<canMsg.can_dlc; i++) { // print the data
      Serial.print(canMsg.data[i], HEX);
      Serial.print(" ");
    }

    Serial.println();
  }
}
```

Vraag: • Maak een opname van het SPI-verkeer voor het zenden van 8 databytes

Antwoord:

CANFRAME_8byte_speed.png

Vraag: • Noteer de opvallende zaken aan de uitvoer van de logic analyzer

Antwoord: Op het MISO-kanaal wordt constant de bytereeks FF FF 08 gestuurd, totdat het canframe uiteindelijk verzonden wordt. Dan is er op het MISO kanaal eerst een FF FF 00 byte-reeks te zien, waarna het CAN-frame op het MOSI-kanaal verstuurd wordt. Waarschijnlijk betekent een FF FF 08 byte-reeks op het MISO-kanaal dat de MCP2515 chip nog niet klaar is om een nieuw canframe te verzenden. Wanneer FF FF 00 wordt gestuurd, is de chip dus wel klaar en wordt het canframe over het MOSI kanaal naar de canbus verstuurd.
interessant SPI zender.png

Vraag: • Hoe lang duurt het SPI-transport van één enkel CAN-bericht van 8 bytes naar de CAN-module?

Antwoord: 40.125 microseconden
canframe_8byte_toslave.png

CAN zend

Vraag: • Maak opname CAN-frame transport voor het zenden van 8 bytes

Antwoord:
CANFRAME_8byte_transport.png

Vraag: • Noteer de opvallende zaken aan de uitvoer van de logic analyzer

Antwoord: De verstuurde data, message length en ID is in plaintext zichtbaar via de logic analyzer. Het can-bericht begint met een Start of Frame bit met waarde 0 en eindigt met een end of frame sectie van 7 bits (1111111) en een Inter Frame Spacing sectie van 3 bits (111)

Vraag: • Hoe lang duurt het verzenden van één CAN-bericht door de CAN-module?

Antwoord: 122.916667 microseconden

Vraag: • Hoe kun je zien of het frame door de ontvanger is geaccordeerd?

Antwoord: de laatste 2 bits van de EOF sectie geven hierover informatie. de een na laatste geeft aan of het frame is ontvangen en de laatste geeft aan of het correct is verzonden

Vraag: • Wat is de overhead van een CAN-frame om 1 databyte te verzenden?

Antwoord: Op de screenshot van pulseview is te zien dat in dit voorbeeld in totaal 122 bits worden verstuurd voor het verzenden van de CAN-frame. als 8 bits hiervan daadwerkelijk data zijn, dan zijn er 114 bits aan overhead. de overhead in percentage is dan $100 - 8/122 * 100$, oftewel 93.4426229508%

Vraag: • Wat is de overhead van een CAN-frame om 8 databytes te verzenden?

Antwoord: Op de screenshot van pulseview is te zien dat in dit voorbeeld in totaal 122 bits worden verstuurd voor het verzenden van de CAN-frame. als 64 bits hiervan daadwerkelijk data zijn, dan zijn er 58 bits aan overhead. de overhead in percentage is dan $100 - 64/122 * 100$, oftewel 47.5409836066%

SPI ontvangstzijde

Vraag: • Maak opname SPI-transport voor het ontvangen van 8 bytes

Antwoord:
SPI receive CAN.png

Vraag: • Noteer de opvallende zaken aan de uitvoer van de logic analyzer

Antwoord: Op het MISO kanaal zijn alle databytes die ik verstuur over het CAN-netwerk te zien (FF 00 00 00 00 00 00 FF)

Vraag: • Hoe lang duurt het binnenhalen van de 8 databytes vanuit de CAN-module naar de microcontroller?

Antwoord: het opvragen en ontvangen van het CAN-bericht duurt 123.75 microseconden. Enkel de 8 databytes duurt 29.083 microseconden
SPI receive CAN speed.png
SPI receive CAN speed only 8bytes.png

Vraag: • Hoe lang duurt het voordat de ontvanger weer gaat pollen via SPI om te kijken of er een nieuw CAN-bericht is ontvangen?

Antwoord: 25.946 milliseconden
SPI receive CAN polltime.png

Totaal zenden en ontvangen

Vraag: • Wat is de totale tijdsduur om één pakket te verzenden en ontvangen in Arduino-code?

Antwoord: Dit kan berekend worden door de tijden van het SPI en CAN verkeer voor een CAN-frame van 8 bytes bij elkaar op te tellen. Deze data is uit voorgaande vragen verkregen, dus:

$40.125 + 122.916667 + 123.75 + 29.083333 = 315,875$ microseconden

Zenden buffering één pakket

Vraag: Pas de sketch uit het voorbeeld aan zodat er één oplopend getal (in ASCII) met regelmatige intervallen wordt verzonden. Dus 'zend 1' – interval – 'zend 2' – interval – 'zend 3' – interval etc.

Door dit getal op de ontvanger af te drukken via Serial kunnen we makkelijker controleren of er geen pakketten verloren zijn geraakt.

Antwoord:

```
#include <SPI.h>
#include <mcp2515.h>

struct can_frame canMsg1;
// struct can_frame canMsg2;
MCP2515 mcp2515(10);
int count = 0;

void setup()
{
    Serial.begin(9600);

    while (!Serial)
    {
        ;
    }

    canMsg1.can_id = 0x0F6;
    canMsg1.can_dlc = 8;
    canMsg1.data[0] = 'z';
    canMsg1.data[1] = 'e';
    canMsg1.data[2] = 'n';
    canMsg1.data[3] = 'd';
    canMsg1.data[4] = ' ';
    canMsg1.data[5] = ' ';
    canMsg1.data[6] = ' ';
    canMsg1.data[7] = ' ';

    int result = 0;

    result = mcp2515.reset();

    if (result == MCP2515::ERROR_OK)
    {
```

```

    Serial.println("MCP2515 reset OK");
}
else
{
    Serial.println("MCP2515 reset FAIL");
}

mcp2515.setBaudrate(CAN_1000KBPS, MCP_8MHZ);

if (result == MCP2515::ERROR_OK)
{
    Serial.println("MCP2515 setBaudrate OK");
}
else
{
    Serial.println("MCP2515 setBaudrate FAIL");
}

result = mcp2515.setNormalMode();

if (result == MCP2515::ERROR_OK)
{
    Serial.println("MCP2515 setNormalMode OK");
}
else
{
    Serial.println("MCP2515 setNormalMode FAIL");
}
}

void loop()
{
    String countStr = String(count);
    for(int i = 0; i < 3 && i < countStr.length(); i++)
    //parse it as ASCII. only 3 bytes left in the canframe. problem when > 999
    {
        int iterator = 5 + i;
        canMsg1.data[iterator] = countStr[i];
    }

    while (mcp2515.sendMessage(&canMsg1) != MCP2515::ERROR_OK)
    {
        Serial.println("Error sending message");
    }
    Serial.println("Messages sent");
    count++;

    delay(100);
}

#include <SPI.h>
#include <mcp2515.h>

struct can_frame canMsg;
MCP2515 mcp2515(10);

void setup() {
    Serial.begin(115200);

    mcp2515.reset();
    mcp2515.setBaudrate(CAN_1000KBPS, MCP_8MHZ);
    mcp2515.setNormalMode();

    Serial.println("----- CAN Read -----");
    Serial.println("ID  DLC  DATA");
}

void loop() {
    if (mcp2515.readMessage(&canMsg) == MCP2515::ERROR_OK) {
        Serial.print(canMsg.can_id, HEX); // print ID
        Serial.print(" ");
    }
}

```

```

Serial.print(canMsg.can_dlc, HEX); // print DLC
Serial.print(" ");

for (int i = 0; i<canMsg.can_dlc; i++) { // print the data
  Serial.print((char)canMsg.data[i]);
}

Serial.println();
}
}

```

Vraag: Maak opnames van de analyzer en bestudeer deze.

Antwoord: In zowel de seriële monitor van de ontvanger als de logic analyzer lijken alle berichten goed aan te komen
CAN send buffer 1 packet.png

Zenden buffering meerdere pakketten

Vraag: Pas de sketch uit het voorbeeld aan zodat er twee aan twee oplopend getallen (in ASCII) met regelmatige intervallen wordt verzonden. Dus 'zend 1' – 'zend 2' – interval – 'zend 3' – 'zend 4' interval – 'zend 5' – 'zend 6' – interval etc.

Antwoord:

```

#include <SPI.h>
#include <mcp2515.h>

struct can_frame canMsg1;
struct can_frame canMsg2;
MCP2515 mcp2515(10);
int count = 0;

void setup()
{
  Serial.begin(9600);

  while (!Serial)
  {
    ;
  }

  canMsg1.can_id = 0x0F6;
  canMsg1.can_dlc = 8;
  canMsg1.data[0] = 'z';
  canMsg1.data[1] = 'e';
  canMsg1.data[2] = 'n';
  canMsg1.data[3] = 'd';
  canMsg1.data[4] = ' ';
  canMsg1.data[5] = ' ';
  canMsg1.data[6] = ' ';
  canMsg1.data[7] = ' ';

  canMsg2 = canMsg1;

  int result = 0;

  result = mcp2515.reset();

  if (result == MCP2515::ERROR_OK)
  {
    Serial.println("MCP2515 reset OK");
  }
  else
  {
    Serial.println("MCP2515 reset FAIL");
  }

  mcp2515.setBaudrate(CAN_1000KBPS, MCP_8MHZ);

  if (result == MCP2515::ERROR_OK)
  {

```



```

        Serial.println("MCP2515 setBtirate OK");
    }
    else
    {
        Serial.println("MCP2515 setBtirate FAIL");
    }

    result = mcp2515.setNormalMode();

    if (result == MCP2515::ERROR_OK)
    {
        Serial.println("MCP2515 setNormalMode OK");
    }
    else
    {
        Serial.println("MCP2515 setNormalMode FAIL");
    }
}

void loop()
{
    String countStr = String(count);
    String countStr2 = String(count + 1);
    for (int i = 0; i < 3 && i < countStr.length(); i++)
// parse it as ASCII. only 3 bytes left in the canframe. problem when > 999
    {
        int iterator = 5 + i;
        canMsg1.data[iterator] = countStr[i];
    }

    for (int i = 0; i < 3 && i < countStr2.length(); i++)
// parse it as ASCII. only 3 bytes left in the canframe. problem when > 999
    {
        int iterator = 5 + i;
        canMsg2.data[iterator] = countStr2[i];
    }

    mcp2515.sendMessage(&canMsg1);
    mcp2515.sendMessage(&canMsg2);

    Serial.println("Messages sent");
    count += 2;

    delay(100);
}

```

Vraag: Controleer met de analyzer of alles goed gaat en of de pakketten daadwerkelijk aankomen bij de ontvanger.

Antwoord: Zowel de logic analyzer als de seriele monitor geven aan dat alle berichten succesvol aankomen.

SPI receive CAN buffer multiple.png

Zie het SPI verkeer wanneer een bericht binnenkomt

Vraag: Bekijk ook wanneer de CAN-module het eerste CAN-bericht verstuurd t.o.v. het SPI-verkeer voor het tweede pakket.

Antwoord: In de screenshot is te zien dat terwijl het CAN-bericht verstuurd wordt, het tweede CAN-bericht al via SPI naar de CAN-module wordt gestuurd.

CAN_SEND_SPI_SECOND.png

Zenden buffering problemen

Vraag: Verhoog het aantal direct achter elkaar verzonden getallen naar 3, 4 en 5.

Antwoord:

```

#include <SPI.h>
#include <mcp2515.h>

struct can_frame canMsg1;
MCP2515 mcp2515(10);
int32_t count = 0;

//union to split int32 into 4 bytes
typedef union

```

```

{
    int32_t i32;
    uint8_t u8[4];
} union_32;

union_32 u32_1;

void setup()
{
    u32_1.i32 = 0;

    Serial.begin(9600);

    while (!Serial)
    {
        ;
    }

    canMsg1.can_id = 0x0F6;
    canMsg1.can_dlc = 8;
    canMsg1.data[0] = 'z';
    canMsg1.data[1] = 'e';
    canMsg1.data[2] = 'n';
    canMsg1.data[3] = 'd';
    canMsg1.data[4] = ' ';
    canMsg1.data[5] = ' ';
    canMsg1.data[6] = ' ';
    canMsg1.data[7] = ' ';

    int result = 0;
    result = mcp2515.reset();

    if (result == MCP2515::ERROR_OK)
    {
        Serial.println("MCP2515 reset OK");
    }
    else
    {
        Serial.println("MCP2515 reset FAIL");
    }

    mcp2515.setBtrrate(CAN_1000KBPS, MCP_8MHZ);

    if (result == MCP2515::ERROR_OK)
    {
        Serial.println("MCP2515 setBtrrate OK");
    }
    else
    {
        Serial.println("MCP2515 setBtrrate FAIL");
    }

    result = mcp2515.setNormalMode();

    if (result == MCP2515::ERROR_OK)
    {
        Serial.println("MCP2515 setNormalMode OK");
    }
    else
    {
        Serial.println("MCP2515 setNormalMode FAIL");
    }
}

void updateCanMsg()
{
    canMsg1.data[4] = u32_1.u8[3];
    canMsg1.data[5] = u32_1.u8[2];
    canMsg1.data[6] = u32_1.u8[1];
    canMsg1.data[7] = u32_1.u8[0];
    u32_1.i32++;
    mcp2515.sendMessage(&canMsg1);
}

void loop()

```

```

{
    updateCanMsg ();
    updateCanMsg ();
    updateCanMsg ();
    updateCanMsg ();
    updateCanMsg ();

    delay (10);
}

```

Vraag: Wat gebeurt er precies in het dataverkeer tussen microcontroller en zendende CAN-module?

Antwoord: het CAN-frame wordt correct via SPI verstuurd naar de CAN-module. daar wordt het in een buffer gestopt om later te verzenden. Hierdoor klopt de volgorde van de berichten niet meer. bericht 474 kan bijvoorbeeld eerder aankomen dan bericht 472. In de bovenstaande code heb ik in totaal 884 CAN-berichten gestuurd waarvan er 708 aan zijn gekomen. Het lijkt erop dat de buffer overstroomd en er hierdoor data verloren gaat

Vraag: Wat zou er gebeuren als we proberen op maximale SPI-snelheid zoveel mogelijk CAN-berichten per seconde te versturen?

Antwoord: De buffer overstroomt veel sneller en veel berichten zullen door de buffer verwijderd worden

Vraag: Beschrijf twee mogelijke oplossingen (niet bouwen) om zo zoveel mogelijk CAN-berichten per seconde te versturen.

Antwoord: Ik neem aan dat hier bedoelt wordt om zoveel mogelijk CAN-berichten te versturen zonder dat er berichten verloren gaan.

1: de verstuurder 2 verschillende ID's laten gebruiken en 2 ontvangers gebruiken in plaats van 1. Hierdoor wordt de load van het lezen van CAN-berichten 50/50 verdeeld en kunnen er meer berichten afgehandeld worden.

2: Een snellere CAN-bus gebruiken voor de ontvanger, aangezien hier de bottleneck ligt

3: Niet per se meer berichten, maar zolang je maar 1 byte gebruikt in je ID, kun je nog de rest van de 29-bit CAN-ID veld gebruiken voor extra data. hierdoor kan een bericht toch weer 21 extra bits gebruiken in 1 can-frame en heb je wel meer data over het netwerk

Ontvangen buffering

Vraag: Bestudeer het ophalen middels SPI-berichten van ontvangen CAN-berichten. Waarom zitten er pauzes in het ophalen van de berichten?

Antwoord: Aan de ontvangskant wordt een bericht binnengehaald via SPI en daarna geprint in de seriële monitor. dit kost tijd waardoor er niet gelijk een nieuw bericht wordt gepolpt via SPI. Ook zullen er waarschijnlijk pauzes zijn toegevoegd tussen SPI-calls om de klok tussen de CAN-module en de microcontroller synchroon te houden.

De code van de ontvanger:

```

#include <SPI.h>
#include <mcp2515.h>

struct can_frame canMsg;
MCP2515 mcp2515(10);

//union to split int32 into 4 bytes
typedef union
{
    int32_t i32;
    uint8_t u8[4];
} union_32;

union_32 u32_1;

void setup()
{
    Serial.begin(115200);

    pinMode(3, OUTPUT);
    digitalWrite(3, LOW);

    mcp2515.reset();
    mcp2515.setBitrate(CAN_1000KBPS, MCP_8MHZ);
    mcp2515.setNormalMode();

    Serial.println("----- CAN Read -----");
    Serial.println("ID  DLC  DATA");
}

void loop()
{
    if (mcp2515.readMessage(&canMsg) == MCP2515::ERROR_OK)
    {

```

```

PORTD ^= 1 << 3;
PORTD ^= 1 << 3;

Serial.print(canMsg.can_id, HEX); // print ID
Serial.print(" ");
Serial.print(canMsg.can_dlc, HEX); // print DLC
Serial.print(" ");

for (int i = 0; i < 4; i++)
{ // print the data
  Serial.print((char)canMsg.data[i]);
}

u32_1.u8[0] = canMsg.data[7];
u32_1.u8[1] = canMsg.data[6];
u32_1.u8[2] = canMsg.data[5];
u32_1.u8[3] = canMsg.data[4];

Serial.print(" ");
Serial.println(u32_1.i32);
}
}

```

Datasheet MCP2515

Vraag: Bestudeer de architectuur van de chip op de CAN-module, de MCP2515 en verklaar bovenstaande meetresultaten/problemen bij het zenden en ontvangen.

Antwoord: De mcp2515 heeft twee receive buffers. Hiermee kan één bericht gecheckt worden door de mcp2515 op ingestelde criteria/filters terwijl een tweede bericht nog kan binnenkomen. Wanneer er dan nog een bericht binnenkomt, wordt dit bericht genegeerd en wordt er een flag op de mcp2515 getoggled om aan te geven dat de buffer is overstroomd. De zender maakt ook gebruik van "One-Shot mode" wat betekent dat elk bericht maar één keer wordt verzonden, ook wanneer het niet correct wordt ontvangen. Hierdoor is het dus mogelijk dat berichten niet aankomen en dat de CAN-bussen er niks aan doen.

[gebruikte datasheet](#)

Opdracht FreeRTOS Task switching

FreeRTOS taken

Installatie

vraag: Hoeveel flash/RAM kost FreeRTOS geïmplementeerd op een Arduino Uno. Literatuur in en praktijk?

antwoord:

Vanuit de [FreeRTOS FAQ](#)

RAM:	Item	RAM Used
	Scheduler Itself	236 bytes (can easily be reduced by using smaller data types).
	For each queue you create, add	76 bytes + queue storage area (see FAQ Why do queues use that much RAM?)
	For each task you create, add	64 bytes (includes 4 characters for the task name) + the task stack size.

FLASH:

This depends on your compiler, architecture, and RTOS kernel configuration.
The RTOS kernel itself required about 5 to 10 KBytes of ROM space.

PRAKTIJK:

Om dit te testen heb ik een simpele toepassing van FreeRTOS gemaakt. Hierbij wordt 1 taak aangemaakt die een 10 milliseconden delay uitvoert:

```
#include <Arduino_FreeRTOS.h>
#include <Arduino.h>

// Task function prototype
void vTaskDoNothing(void *pvParameters);

void setup() {
    // Create the task
    xTaskCreate(
        vTaskDoNothing,    // Task function
        "Noth",            // Task name
        128,               // Stack size (in words, not bytes)
        NULL,              // Task parameters
        1,                 // Task priority
        NULL               // Task handle
    );
}

void loop() {
    // Empty loop as FreeRTOS handles tasks
}

void vTaskDoNothing(void *pvParameters) {
    (void) pvParameters; // Suppress unused parameter warning

    for (;;) {
        // Do nothing
        vTaskDelay(pdMS_TO_TICKS(10)); // Delay for 10 ms
    }
}
```

In totaal kostte dit 172 bytes aan RAM en 6784 bytes aan Flash

RAM per taak

vraag: Maak één taak en vervolgens twee taken aan in FreeRTOS. Hoeveel Flash wordt telkens meer gebruikt en hoeveel RAM?

antwoord:

1 taak:

RAM: [=] 8.4% (used 172 bytes from 2048 bytes)
Flash: [==] 21.0% (used 6784 bytes from 32256 bytes)

2 taken:

RAM: [=] 8.7% (used 178 bytes from 2048 bytes)

Flash: [==] 21.0% (used 6782 bytes from 32256 bytes)

Voor elke taak wordt dus 6 bytes aan RAM ingenomen. De Flash lijkt omlaag te gaan als er een nieuwe taak toegevoegd wordt. Dit is waarschijnlijk door optimalisaties in de compiler

gebruikte code:

```
#include <Arduino_FreeRTOS.h>
#include <Arduino.h>

// Task function prototype
void vTaskDoNothing1(void *pvParameters);
void vTaskDoNothing2(void *pvParameters);

void setup()
{
    // Create the task
    xTaskCreate(
        vTaskDoNothing1, // Task function
        "Noth1",         // Task name
        128,             // Stack size (in words, not bytes)
        NULL,            // Task parameters
        1,               // Task priority
        NULL              // Task handle
    );
    xTaskCreate(
        vTaskDoNothing2, // Task function
        "Noth2",         // Task name
        128,             // Stack size (in words, not bytes)
        NULL,            // Task parameters
        1,               // Task priority
        NULL              // Task handle
    );
}

void loop()
{
    // Empty loop as FreeRTOS handles tasks
}

void vTaskDoNothing1(void *pvParameters)
{
    (void)pvParameters; // Suppress unused parameter warning

    for (;;)
    {
        // Do nothing
        vTaskDelay(pdMS_TO_TICKS(10)); // Delay for 10 ms
    }
}

void vTaskDoNothing2(void *pvParameters)
{
    (void)pvParameters; // Suppress unused parameter warning

    for (;;)
    {
        // Do nothing
        vTaskDelay(pdMS_TO_TICKS(10)); // Delay for 10 ms
    }
}
```

RAM toewijzing

vraag: Maak een klein voorbeeld met één en twee taken ieder met een niet default stackgrootte. Hoeveel RAM kost iedere taak extra en waar/wanneer gebeurt de RAM-toewijzing voor de stack?

antwoord:

```
#include <Arduino_FreeRTOS.h>
#include <Arduino.h>
```

```

// Task function prototype
void vTaskDoNothing1(void *pvParameters);
void vTaskDoNothing2(void *pvParameters);

void setup()
{
    // Create the task
    xTaskCreate(
        vTaskDoNothing1, // Task function
        "Noth1",         // Task name
        500,             // Stack size (in words, not bytes)
        NULL,            // Task parameters
        1,               // Task priority
        NULL              // Task handle
    );
    // xTaskCreate(
    //     vTaskDoNothing2, // Task function
    //     "Noth2",         // Task name
    //     650,             // Stack size (in words, not bytes)
    //     NULL,            // Task parameters
    //     1,               // Task priority
    //     NULL             // Task handle
    // );
}

void loop()
{
    // Empty loop as FreeRTOS handles tasks
}

void vTaskDoNothing1(void *pvParameters)
{
    (void)pvParameters; // Suppress unused parameter warning

    for (;;)
    {
        // Do nothing
        vTaskDelay(pdMS_TO_TICKS(10)); // Delay for 10 ms
    }
}

void vTaskDoNothing2(void *pvParameters)
{
    (void)pvParameters; // Suppress unused parameter warning

    for (;;)
    {
        // Do nothing
        vTaskDelay(pdMS_TO_TICKS(10)); // Delay for 10 ms
    }
}

```

Deze code verbruikt 172 bytes from 2048 bytes RAM.

De 2de taak uncommenten:

```

#include <Arduino_FreeRTOS.h>
#include <Arduino.h>

// Task function prototype
void vTaskDoNothing1(void *pvParameters);
void vTaskDoNothing2(void *pvParameters);

void setup()
{
    // Create the task
    xTaskCreate(
        vTaskDoNothing1, // Task function
        "Noth1",         // Task name
        500,             // Stack size (in words, not bytes)

```



```

    NULL,          // Task parameters
    1,             // Task priority
    NULL           // Task handle
);
xTaskCreate(
    vTaskDoNothing2, // Task function
    "Noth2",         // Task name
    650,             // Stack size (in words, not bytes)
    NULL,            // Task parameters
    1,               // Task priority
    NULL             // Task handle
);
}

void loop()
{
    // Empty loop as FreeRTOS handles tasks
}

void vTaskDoNothing1(void *pvParameters)
{
    (void)pvParameters; // Suppress unused parameter warning

    for (;;)
    {
        // Do nothing
        vTaskDelay(pdMS_TO_TICKS(10)); // Delay for 10 ms
    }
}

void vTaskDoNothing2(void *pvParameters)
{
    (void)pvParameters; // Suppress unused parameter warning

    for (;;)
    {
        // Do nothing
        vTaskDelay(pdMS_TO_TICKS(10)); // Delay for 10 ms
    }
}

```

Zorgt ervoor dat 178 bytes van de 2048 bytes gebruikt worden. Voor elke taskcreate call in een FreeRTOS programma wordt dus 6 bytes aan ram verbruikt en de taken worden waarschijnlijk pas gedurende run-time ge-allocate.

Pin toggle

Maak nogmaals de snelle pin toggle (zonder FreeRTOS). Deze wordt later gebruikt om metingen uit te voeren aan FreeRTOS. Bestudeer deze toggle nauwkeurig met de logic analyzer.

vraag: • Om de ong. 1 ms wordt de pin toggle onderbroken. Hoe lang duurt deze onderbreking en wat is de oorzaak? Toon aan dat je de juiste oorzaak hebt gevonden door de code te wijzigen.

antwoord:

```

#include <Arduino.h>

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

void loop()
{
    PORTB = PORTB ^ B00100000;
}

```

onderbreking.png

De onderbreking duurt op de Arduino Uno 6.5 microseconden. Dit komt doordat er een interrupt in de achtergrond draait (waarschijnlijk voor de millis() functie) die elke milliseconden wat logica uit moet voeren. deze interrupt annuleert tijdelijk de logica van de pin toggle. Door alle interrupts uit

te zetten gebeurt dit niet meer. Ook kost een iteratie van de loop() functie meer tijd dan hetzelfde uitvoeren in een while(true) loop in de setup() functie. Zie code

```
#include <Arduino.h>

void setup()
{
  pinMode(13, OUTPUT);
  noInterrupts(); // Schakel alle interrupts uit

  while(1){
    PORTB = B00100000;
    PORTB = B00000000;
  }
}

void loop()
{
}
```

Het itereren over de while(1) loop kost alsnog wat tijd, waarschijnlijk omdat de conditie gecheckt moet worden. Hierdoor is het sneller om de while(1) loop te vullen met honderden pinflip calls, aangezien de conditie dan mindersnel wordt gecheckt.

vraag: • Wat is de minimaal haalbare pulsbreedte?

antwoord: 125 nanoseconden op de arduino uno

fastest possible.png

Drie taken

We gaan drie taken implementeren. Daartoe is de code van

<https://circuitdigest.com/microcontroller-projects/arduino-freertos-tutorial1-creating-freertos-task-to-blink-led-in-arduino-uno> aangepast om een taakoverzicht zichtbaar te maken.

vraag: Welke taken draaien er uiteindelijk? Verklaar de aanwezigheid van de draaiende taken.

antwoord: Er zijn in totaal 5 taken geregistreerd op de Arduino. Hiervan zijn de drie ingestelde aanwezig. De andere 2 zijn IDLE en Tmr Svc. De IDLE-taken heeft de laagst mogelijke prioriteit en zorgt ervoor dat de CPU slaapt als er geen andere taken zijn om te draaien. Dit is efficiënt voor het stroomverbruik. de Tmr Svc (oftewel timer service / daemon) houdt een lijst met software timers bij en draait wanneer de software timer voorbij is om hem te updaten. Dit zijn standaard processen die nodig zijn om FreeRTOS correct te laten functioneren.

<https://mcuoneclipse.com/2018/05/27/tutorial-understanding-and-using-freertos-software-timers/>

<https://m.freertos.org/Documentation/02-Kernel/02-Kernel-features/01-Tasks-and-co-routines/15-Idle-task>

vraag: Verklaar de status van de taken (B, R, X).

antwoord: gedurende het printen van de taken is de taskprint task de enigste die draaiende is, aangezien de arduino uno niet meerdere taken tegelijk kan draaien. De IDLE task staat op ready, wat betekend dat deze taak elk moment uitgevoerd kan worden als het moet (dus wanneer er geen taak met hogere prioriteit gedraaid hoeft te worden). De andere taken zijn in blocked, omdat ze wachten op externe events, zoals een delay. task1 en task2 doen dit door het aanroepen van vTaskDelay, waardoor ze na deze call in de blocked state komen totdat de timer voorbij is. de Tmr Svc task doet waarschijnlijk hetzelfde.

De status

status.png

vraag: Maak een afdruk van de twee LED-signalen op de logic analyzer.

antwoord:

pins.png

En de loop?

vraag: • Schakel het printen van de taken uit (serial.print is op zichzelf een interrupt-gestuurde taak met blokkerende aanroep)

antwoord:

```
// Copied from https://circuitdigest.com/microcontroller-projects/arduino-freertos-tutorial1-creating-freertos-task-to-blink-led-in-arduino-uno
#include <Arduino_FreeRTOS.h>
#include <Arduino.h>

const byte LEDPIN1 = 12;
const byte LEDPIN2 = 11;
```

```

const int DELAY1 = 200;
const int DELAY2 = 300;
const int DELAYPRINT = 1000;

const unsigned char bufferSize = 250;
char ptrTaskList[bufferSize];

void TaskBlink1( void *pvParameters );
void TaskBlink2( void *pvParameters );

void setup() {
    xTaskCreate(
        TaskBlink1
        , "task1"
        , 128
        , NULL
        , 1
        , NULL );

    xTaskCreate(
        TaskBlink2
        , "task2"
        , 128
        , NULL
        , 1
        , NULL );

    vTaskStartScheduler();

}

void loop()
{
}

void TaskBlink1(void *pvParameters) {
    pinMode(LEDPIN1, OUTPUT);
    while (1)
    {
        digitalWrite(LEDPIN1, HIGH);
        vTaskDelay( DELAY1 / portTICK_PERIOD_MS );
        digitalWrite(LEDPIN1, LOW);
        vTaskDelay( DELAY1 / portTICK_PERIOD_MS );
    }
}

void TaskBlink2(void *pvParameters)
{
    pinMode(LEDPIN2, OUTPUT);
    while (1)
    {
        digitalWrite(LEDPIN2, HIGH);
        vTaskDelay( DELAY2 / portTICK_PERIOD_MS );
        digitalWrite(LEDPIN2, LOW);
        vTaskDelay( DELAY2 / portTICK_PERIOD_MS );
    }
}

```

vraag: • Breid het programma uit met de snelle pin-toggle in de loop()

antwoord:

```

// Copied from https://circuitdigest.com/microcontroller-projects/arduino-freertos-tutorial1-creating-freertos
-task-to-blink-led-in-arduino-uno
#include <Arduino_FreeRTOS.h>
#include <Arduino.h>

const byte LEDPIN1 = 12;
const byte LEDPIN2 = 11;
const byte LEDPIN3 = 10;
const int DELAY1 = 200;
const int DELAY2 = 300;
const int DELAYPRINT = 1000;

```

```

const unsigned char bufferSize = 250;
char ptrTaskList[bufferSize];

void TaskBlink1(void *pvParameters);
void TaskBlink2(void *pvParameters);

void setup()
{
    pinMode(LEDPIN3, OUTPUT);
    xTaskCreate(
        TaskBlink1, "task1", 128, NULL, 1, NULL);

    xTaskCreate(
        TaskBlink2, "task2", 128, NULL, 1, NULL);

    vTaskStartScheduler();
}

void loop()
{
    while (true)
    {
        PORTB ^= (1 << 2);
    }
}

void TaskBlink1(void *pvParameters)
{
    pinMode(LEDPIN1, OUTPUT);
    while (1)
    {
        digitalWrite(LEDPIN1, HIGH);
        vTaskDelay(DELAY1 / portTICK_PERIOD_MS);
        digitalWrite(LEDPIN1, LOW);
        vTaskDelay(DELAY1 / portTICK_PERIOD_MS);
    }
}

void TaskBlink2(void *pvParameters)
{
    pinMode(LEDPIN2, OUTPUT);
    while (1)
    {
        digitalWrite(LEDPIN2, HIGH);
        vTaskDelay(DELAY2 / portTICK_PERIOD_MS);
        digitalWrite(LEDPIN2, LOW);
        vTaskDelay(DELAY2 / portTICK_PERIOD_MS);
    }
}

```

vraag: • Toon op de logic analyzer de uitvoer van de drie pinnen

antwoord:

3 pins.png

vraag: Wat is waar te nemen op de logic analyzer en verklaar de uitvoer. Wat is de task switch time? Waar is de loop() gebleven?

antwoord: De loop wordt nog steeds gedraaid in mijn code. dat is waar de pinflip voor pin10 plaatsvindt. Deze pinflip wordt geannuleerd gedurende het draaien van de andere taken. Hiermee kun je beredeneren wat de taskswitchtime is door te kijken wanneer het flippen van pin10 stopt en een andere pin vanuit task1 of task2 wordt uitgevoerd. Dat is 38.125 microseconden. Ook lijkt er ongeveer elke milliseconden (ongeveer elke 1015 microseconden) een task uitgevoerd te worden die niet pin 11 of 12 flipt. Dit is waarschijnlijk de Tmr Svc task, zodat het aantal milliseconden runtime up-to-date blijft.

taskswitchtime.png

millisecond interrupt.png

En de setup()

In de ESP32-versie is de setup() volgens

<https://www.digikey.nl/en/maker/projects/introduction-to-rtos-solution-to-part-2-freertos/b3f84c9c9455439ca2dcb8ccfce9dec5> een aparte taak

vraag: Is dat bij de Arduino ook zo (hint: gebruik uxTaskGetNumberOfTasks()) ?

antwoord: Nee, als onderstaande code wordt uitgevoerd geeft de functie uxTaskGetNumberOfTasks() aan dat er 0 taken aanwezig zijn. FreeRTOS

start dus pas op nadat de setup() functie is afgelopen

```
#include <Arduino_FreeRTOS.h>
#include <Arduino.h>

void setup()
{
    Serial.begin(9600);
    while (!Serial)
        ;

    Serial.println(uxTaskGetNumberOfTasks());
}

void loop()
{
}
```

vTaskDelay of delay

Om in te zien waarom er gebruik wordt gemaakt van de FreeRTOS-API vTaskDelay vervangen we alle aanroepen door de gewone delay. Let op: de arduino FreeRTOS library heeft de delay methode by-default gekoppeld aan vTaskDelay, dus om het verschil te kunnen zien moet configUSE_PORT_DELAY op 0 ingesteld worden in de FreeRTOSConfig.h

vraag: Wat gebeurt er op de logic analyzer?

antwoord: Als we de volgende code gebruiken:

```
// Copied from https://circuitdigest.com/microcontroller-projects/arduino-freertos-tutorial1-creating-freertos
-task-to-blink-led-in-arduino-uno
#include <Arduino_FreeRTOS.h>
#include <Arduino.h>

const byte LEDPIN1 = 12;
const byte LEDPIN2 = 11;
const byte LEDPIN3 = 10;
const int DELAY1 = 200;
const int DELAY2 = 300;
const int DELAYPRINT = 1000;

const unsigned char bufferSize = 250;
char ptrTaskList[bufferSize];

void TaskBlink1(void *pvParameters);
void TaskBlink2(void *pvParameters);

void setup()
{
    pinMode(LEDPIN3, OUTPUT);
    xTaskCreate(
        TaskBlink1, "task1", 128, NULL, 1, NULL);

    xTaskCreate(
        TaskBlink2, "task2", 128, NULL, 1, NULL);

    vTaskStartScheduler();
}

void loop()
{
    while (true)
    {
        PORTB |= (1 << PORTB2);
        PORTB &= ~(1 << PORTB2);
    }
}

void TaskBlink1(void *pvParameters)
{
    pinMode(LEDPIN1, OUTPUT);
    while (1)
    {
        digitalWrite(LEDPIN1, HIGH);
    }
}
```

```

        vTaskDelay(DELAY1 / portTICK_PERIOD_MS);
        digitalWrite(LEDPIN1, LOW);
        vTaskDelay(DELAY1 / portTICK_PERIOD_MS);
    }
}

void TaskBlink2(void *pvParameters)
{
    pinMode(LEDPIN2, OUTPUT);
    while (1)
    {
        digitalWrite(LEDPIN2, HIGH);
        vTaskDelay(DELAY2 / portTICK_PERIOD_MS);
        digitalWrite(LEDPIN2, LOW);
        vTaskDelay(DELAY2 / portTICK_PERIOD_MS);
    }
}

```

Krijgen we de volgende meting:

vtaskdelay.png

Hierin is te zien dat alle tasks volgens hun ingestelde timing correct worden uitgevoerd. FreeRTOS start dus de taken volgens de ingestelde delays.

Wanneer we de vTaskDelay calls vervangen met delay calls:

```

// Copied from https://circuitdigest.com/microcontroller-projects/arduino-freertos-tutorial1-creating-freertos
-task-to-blink-led-in-arduino-uno
#include <Arduino_FreeRTOS.h>
#include <Arduino.h>

const byte LEDPIN1 = 12;
const byte LEDPIN2 = 11;
const byte LEDPIN3 = 10;
const int DELAY1 = 200;
const int DELAY2 = 300;
const int DELAYPRINT = 1000;

const unsigned char bufferSize = 250;
char ptrTaskList[bufferSize];

void TaskBlink1(void *pvParameters);
void TaskBlink2(void *pvParameters);

void setup()
{
    pinMode(LEDPIN3, OUTPUT);
    xTaskCreate(
        TaskBlink1, "task1", 128, NULL, 1, NULL);

    xTaskCreate(
        TaskBlink2, "task2", 128, NULL, 1, NULL);

    vTaskStartScheduler();
}

void loop()
{
    while (true)
    {
        PORTB |= (1 << PORTB2);
        PORTB &= ~(1 << PORTB2);
    }
}

void TaskBlink1(void *pvParameters)
{
    pinMode(LEDPIN1, OUTPUT);
    while (1)
    {
        digitalWrite(LEDPIN1, HIGH);
        delay(DELAY1 / portTICK_PERIOD_MS);
        digitalWrite(LEDPIN1, LOW);
        delay(DELAY1 / portTICK_PERIOD_MS);
    }
}

```

```

    }
}

void TaskBlink2(void *pvParameters)
{
    pinMode(LEDPIN2, OUTPUT);
    while (1)
    {
        digitalWrite(LEDPIN2, HIGH);
        delay(DELAY2 / portTICK_PERIOD_MS);
        digitalWrite(LEDPIN2, LOW);
        delay(DELAY2 / portTICK_PERIOD_MS);
    }
}

```

Krijgen we de volgende meting:

delay.png

Alle taken lijken veel sneller dan ingesteld te draaien en de loop wordt niet meer uitgevoerd. De delay() calls zijn dus blocking aangezien de loop() functie niet kan worden uitgevoerd. De timing van FreeRTOS werkt niet meer goed als de standaard delay wordt gebruikt.

Prioriteiten

vraag: Dat het werken met prioriteiten nog niet zo makkelijk is, wordt nu onderzocht. Pak dezelfde twee LED-taken LED1 en LED2 (uit opdracht 2.7) en maak een toggle-taak (gebaseerd op 2.4) met een derde LED. Geef deze taken prioriteiten volgens onderstaande tabel (waarbij de Romeinse cijfers iteraties zijn). Vul de tabel aan met de waarnemingen op de logic analyzer.

antwoord: De volgende code wordt hiervoor gebruikt:

```

// Copied from https://circuitdigest.com/microcontroller-projects/arduino-freertos-tutorial1-creating-freertos
-task-to-blink-led-in-arduino-uno
#include <Arduino_FreeRTOS.h>
#include <Arduino.h>

const byte LEDPIN1 = 12;
const byte LEDPIN2 = 11;
const byte LEDPIN3 = 10;
const int DELAY1 = 200;
const int DELAY2 = 300;
const int DELAY3 = 500;

const unsigned char bufferSize = 250;
char ptrTaskList[bufferSize];

void TaskBlink1(void *pvParameters);
void TaskBlink2(void *pvParameters);
void TaskBlink3(void *pvParameters);

void setup()
{
    pinMode(LEDPIN1, OUTPUT);
    pinMode(LEDPIN2, OUTPUT);
    pinMode(LEDPIN3, OUTPUT);
    xTaskCreate(
        TaskBlink1, "task1", 128, NULL, 1, NULL);

    xTaskCreate(
        TaskBlink2, "task2", 128, NULL, 1, NULL);

    xTaskCreate(
        TaskBlink3, "task3", 128, NULL, 1, NULL);

    vTaskStartScheduler();
}

void loop()
{
}

void TaskBlink1(void *pvParameters)
{
    while (1)
    {
        digitalWrite(LEDPIN1, HIGH);
    }
}

```

```

    vTaskDelay(DELAY1 / portTICK_PERIOD_MS);
    digitalWrite(LEDPIN1, LOW);
    vTaskDelay(DELAY1 / portTICK_PERIOD_MS);
}
}

void TaskBlink2(void *pvParameters)
{
    while (1)
    {
        digitalWrite(LEDPIN2, HIGH);
        vTaskDelay(DELAY2 / portTICK_PERIOD_MS);
        digitalWrite(LEDPIN2, LOW);
        vTaskDelay(DELAY2 / portTICK_PERIOD_MS);
    }
}

// toggle task based on 2.4
void TaskBlink3(void *pvParameters)
{
    while (1)
    {
        PORTB |= (1 << PORTB2);
        PORTB &= ~(1 << PORTB2);
    }
}

```

Prioriteiten van de taken bij gebruik van `delay()`

Taak	I	II	III	IV	V
LED1	1	1	2	1	2
LED2	1	1	1	2	2
Toggle	1	2	1	1	1
Waarneembare activiteit	LED1 en LED2 worden veel sneller uitgevoerd dan ingesteld. de Toggle-taak wordt consistent geblokt door de delays uit LED1 en LED2, maar wordt buiten de delays nog wel uitgevoerd. De pin van LED1 staat langer op LOW dan op HIGH	Enkel Toggle wordt uitgevoerd. de pins op LED1 en LED2 blijven op LOW	Enkel LED1 wordt uitgevoerd. de pins op LED2 en Toggle blijven op LOW	Enkel LED2 wordt uitgevoerd. de pins op LED1 en Toggle blijven op LOW	LED1 en LED2 worden uitgevoerd. Toggle niet. de pin op Toggle blijft op LOW

Prioriteiten van de taken bij gebruik van `vTaskDelay()`

Taak	I	II	III	IV	V
LED1	1	1	2	1	2
LED2	1	1	1	2	2
Toggle	1	2	1	1	1
Waarneembare activiteit	Alle taken worden correct afgehandeld door de planning van FreeRTOS. De toggletask draait constant tenzij een andere taak gestart moet worden. Dan is er een korte pauze in het togglen en gaat het daarna verder	De toggle-taak is de enigste geconfigureerde taak die draait	Alle taken draaien weer. Gedurende de vTaskDelay van de taak LED1 kunnen dus andere taken gedraaid worden.	Hetzelfde gedrag als bij test III	Visueel hetzelfde gedrag als in test III en IV. gedurende de vTaskDelay van LED1 en LED2 kan de toggle-taak blijven draaien. als de delays van LED1 of LED2 klaar zijn, moet de pin geflipt worden en wordt er opnieuw een vTaskDelay uitgevoerd, waardoor

					de toggle-taak weer uitgevoerd kan worden
--	--	--	--	--	-------------------------------------------------

9 Eindopdracht

Eindopdracht - fouttentabel

Foutnummer	Omschrijving	Code regel	Originele code	Verbetervoorstel
1	Lights worden verkeerd aangezet gedurende de startop routine. Rode en gele led branden terwijl enkel rood moet branden in de STOP-state	104	<code>`LIGHTS[country][light][STOP]`</code>	<code>`LIGHTS[country][STOP][light]`</code>
2	Lights worden verkeerd gewijzigd gedurende de setSignal functie. Geeft eenzelfde resultaat als fout 1	176	<code>`LIGHTS[country][light][aspect]`</code>	<code>`LIGHTS[country][aspect][light]`</code>
3	<code>`switch case`</code> voor case 'N' mist een <code>`break;`</code> waardoor het systeem nooit naar de Nederlandse stand gezet kan worden.	236	<code>`case 'N':`</code>	<code>`case 'N': break;`</code>
4	<code>`switch`</code> mist een <code>`default: break;`</code> . Vormt geen fout/probleem, maar geeft in veel compilers wel een warning	231	<code>`switch (...) { ... }`</code>	<code>`switch (...) { ... default: break; }`</code>
5	ASCII naar decimale conversie is incorrect. Alle commando's met een cijfer (A1, R1, H1, etc.) worden niet goed afgehandeld, aangezien de cijfer na de eerste karakter niet correct naar zijn decimale waarde wordt geconverteerd	230	<code>byte signalOrGateNumber = inputCommand[1] - '0' - 1;</code>	<code>byte signalOrGateNumber = inputCommand[1] - '1';</code>
6	Incorrecte <code>`for`-loop`</code> zorgt ervoor dat gedurende een A1 commando, de rode led van het 2de signaal volledig uit gaat. De iterator mag tot en met de waarde 3 komen, terwijl de LIGHTPINS[signal] array maar 3 indexes heeft. Hierdoor gaat het out of bounds en wordt indirect een digitalwrite uitgevoerd op LIGHTPINS[signal+1][0], wat de rode led van signaal 2 is.	174	<code>`for (int light = 0; light <= NROFLIGHTSPERSIGNAL; light++)`</code>	<code>`for (int light = 0; light < NROFLIGHTSPERSIGNAL; light++)`</code>
7	R-sigitaal zet signalen op BRAKE terwijl de opdracht beschrijft dat het op DRIVE moet.	239	<code>`case 'R': setSignal(signalOrGateNumber, BRAKE);`</code>	<code>`case 'R': setSignal(signalOrGateNumber, DRIVE);`</code>
8	A-sigitaal zet signalen op DRIVE terwijl de opdracht beschrijft dat het op BRAKE moet.	241	<code>`case 'A': setSignal(signalOrGateNumber, DRIVE);`</code>	<code>`case 'A': setSignal(signalOrGateNumber, BRAKE);`</code>
9	<code>`if`-statement</code> gebruikt <code>`=`</code> in plaats van <code>`==`</code> , waardoor lightsRunTestStatus altijd gelijk wordt gezet aan LIGHTSTESTING	251	<code>`if (lightsRunTestStatus = LIGHTSTESTING)`</code>	<code>`if (lightsRunTestStatus == LIGHTSTESTING)`</code>
10	<code>`previousTestBlinkMillis`</code> wordt opnieuw lokaal geïnitieerd, terwijl deze al globaal aanwezig is. Hierdoor gebruikt de scope	155	<code>`unsigned long previousTestBlinkMillis = 0;`</code>	Verwijder regel 155

	van de functie niet meer de globale variabele en wordt de timer nooit bereikt.			
11	<p>Signalen en functionaliteiten om Ledstatussen te wijzigen worden geaccepteerd binnen de LED-testfase. Hierdoor kan het knipperen van de gele led plaatsvinden tijdens de led-test state. Na het exitten van de led-test state worden de leds enkel uitgezet en niet weer aangezet. Hierdoor is het mogelijk om alle leds uit te zetten door 2 keer het T commando door te sturen. Ook is het mogelijk om enkel de gele led te laten knipperen door de lampen te configureren in de Duitse stand, het afremmen signaal te sturen en 2 keer het T commando te sturen, aangezien het knipperen van de gele led wordt geupdate door een timer die na de led-test fase draait. Het lijkt me ongewenst om de leds te kunnen aanpassen gedurende de led-test fase en ook lijkt het me ongewenst dat alle leds uitblijven na het exitten van de led-test fase</p>	172, 181, 139	<pre> void setSignal(byte signal, byte aspect) { signalStatus[signa l] = aspect; for (int light = 0 ; light <= NROFLIGHT SPERSIGNAL; light++) { if ((LIGHTS[coun try][aspect][light] == ON) (LIGHTS[co untry][aspect][light] == OFF)) { digitalWrite(L IGHTPINS[signal][lig ht], LIGHTS[country] [light][aspect]); } } } void blinkLights() { long currentBlinkM illis = millis(); if (currentBlinkMi llis - previousBlink Millis >= BLINKINTER VAL) { previousBlinkMil lis = currentBlinkMi llis; for (byte signal = 0; signal < NROFS IGNALS; signal++) { for (byte ligh t = 0; light < NROFL IGHTSPERSIGNAL; ligh t++) { if (LIGHTS[c ountry][signalStatus [signal]][light] == BLINK) { digitalWri te(LIGHTPINS[signal] [light], blinkStatus); } } } blinkStatus = (b linkStatus == HIGH) ? LOW : HIGH; } } void toggleLightsTes t() { lightsRunTestStatu s = (lightsRunTestSt atus == LIGHTSTESTIN G) ? LIGHTSRUNNING : LIGHTSTESTING; testSignalNumber = 0; testLightNumber = 0; allLightsOff(); } </pre>	<p>Bovenaan `setSignal` en `blinkLights` de volgende check toevoegen: `if (lightsRunTestStatus == LIGHTSTESTING) { return; }`</p> <p>Onderaan `toggleLightsTest` de signalen op STOP zetten, aangezien dit hetzelfde gedrag is wat gebeurd gedurende het opstarten van het systeem: `if (lightsRunTestStatus) { setupLightPins(); }`</p>

12	In de Debug opdracht.docx is de potentiometer is incorrect aangesloten op de digitale poort 2, terwijl in de code de analoge poort 2 wordt uitgelezen om de waarde van de potentiometer op te vragen	256	gateInterval = map(analogRead(POTENTIOPIN), 0, 1023, MINGATEINTERVAL, MAXGATEINTERVAL);	De potentiometer aansluiten op analoge poort 2
----	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----	-----------------------------------------------------------------------------------------	------------------------------------------------

Eindopdracht - TODO

- Check servo's. Deze functioneren volledig niet? misschien een elektrolytische condensator ertussen?
- Check potentiometer. Deze is op een digitale poort aangesloten? Kan dat?

Eindopdracht - Originele code

```

/*Beginning of Auto generated code by Atmel studio */
#include <Arduino.h>

/*End of auto generated code by Atmel studio */

/*
  Decoder for train signals and gates
*/
#include <Servo.h>
//Beginning of Auto generated function prototypes by Atmel Studio
void setupLightDefinitions();
void setupLightPins();
void setupGatePins();
void allLightsOff();
void toggleLightsTest();
void toggleGatesTest();
void testLights();
void setSignal(byte signal, byte aspect);
void blinkLights();
void moveGates();
//End of Auto generated function prototypes by Atmel Studio

const byte NROFSIGNALS = 2;
const byte NROFLIGHTSPERSIGNAL = 3;
const byte NROFASPECTS = 3;
const byte NROFGATES = 2;
const byte NROFCOUNTRIES = 2;

enum lamp_t {OFF, ON, BLINK};
enum light_t {RED, YELLOW, GREEN};
enum lightsRunTestStatus_t {LIGHTSTESTING, LIGHTSRUNNING};
enum aspect_t {DRIVE, BRAKE, STOP};
enum country_t {D, NL};
lamp_t LIGHTS [NROFCOUNTRIES][NROFASPECTS][NROFLIGHTSPERSIGNAL] ;
byte LIGHTPINS[NROFSIGNALS][NROFLIGHTSPERSIGNAL];
byte signalStatus[NROFSIGNALS];
lightsRunTestStatus_t lightsRunTestStatus;
byte blinkStatus;

enum gate_t {OPEN, CLOSED, OPENING, CLOSING};
enum gatesRunTestStatus_t {GATETESTING, GATERUNNING};
byte GATEPINS[NROFGATES] = {9, 8};
const byte POTENTIOPIN = 2;
gatesRunTestStatus_t gatesRunTestStatus;
country_t country;
gate_t gateStatus[NROFGATES];

byte testSignalNumber;
byte testLightNumber;
unsigned long previousBlinkMillis = 0;
unsigned long previousTestBlinkMillis = 0;
byte TESTLIGHTINTERVAL = 250;
const int BLINKINTERVAL = 500;
unsigned int previousGateMillis = 0;

byte gatePosition[NROFGATES];

```

```

const int GATEMAXPOSITION = 180;
const byte GATEINCREMENT = 2;
const int MINGATEINTERVAL = 5;
const int MAXGATEINTERVAL = 500;

int gateInterval = 200;

Servo myservo[NROFGATES];

void setupLightDefinitions() {
    LIGHTPINS[0][RED] = 13;
    LIGHTPINS[0][YELLOW] = 12;
    LIGHTPINS[0][GREEN] = 11;
    LIGHTPINS[1][RED] = 7;
    LIGHTPINS[1][YELLOW] = 6;
    LIGHTPINS[1][GREEN] = 5;

    LIGHTS[NL][STOP][RED] = ON;
    LIGHTS[NL][STOP][YELLOW] = OFF;
    LIGHTS[NL][STOP][GREEN] = OFF;
    LIGHTS[NL][BRAKE][RED] = OFF;
    LIGHTS[NL][BRAKE][YELLOW] = ON;
    LIGHTS[NL][BRAKE][GREEN] = OFF;
    LIGHTS[NL][DRIVE][RED] = OFF;
    LIGHTS[NL][DRIVE][YELLOW] = OFF;
    LIGHTS[NL][DRIVE][GREEN] = ON;

    LIGHTS[D][STOP][RED] = ON;
    LIGHTS[D][STOP][YELLOW] = OFF;
    LIGHTS[D][STOP][GREEN] = OFF;
    LIGHTS[D][BRAKE][RED] = OFF;
    LIGHTS[D][BRAKE][YELLOW] = ON;
    LIGHTS[D][BRAKE][GREEN] = BLINK;
    LIGHTS[D][DRIVE][RED] = OFF;
    LIGHTS[D][DRIVE][YELLOW] = OFF;
    LIGHTS[D][DRIVE][GREEN] = ON;

    blinkStatus = HIGH;
}

//initialize digital pins as output and set signal and lights to aspect STOP
void setupLightPins() {
    for (byte signal = 0; signal < NROFSIGNALS; signal++) {
        for (byte light = 0; light < NROFLIGHTSPERSIGNAL; light++) {
            pinMode(LIGHTPINS[signal][light], OUTPUT);
            digitalWrite(LIGHTPINS[signal][light], LIGHTS[country][light][STOP]);
        }
        signalStatus[signal] = STOP;
    }
}

//initialize gate pins for servo and position to 0
void setupGatePins() {
    for (byte gate = 0; gate < NROFGATES; gate++) {
        gatePosition[gate] = 0;
        gateStatus[gate] = CLOSED;
        myservo[gate].attach(GATEPINS[gate]);
        myservo[gate].write(gatePosition[gate]);
    }
}

// initialize system; at startup no test
void setup() {
    setupLightDefinitions();
    setupLightPins();
    TESTLIGHTINTERVAL = 1000;
    lightsRunTestStatus = LIGHTSRUNNING;
    gatesRunTestStatus = GATERUNNING;
    Serial.begin(9600, SERIAL_7E2);
    Serial.setTimeout(500);
}

void allLightsOff() {
    for (byte signal = 0; signal < NROFSIGNALS; signal++) {
        //switch all lights off not in student version. Not specified
    }
}

```

```

    for (int light = 0; light < NROFLIGHTSPERSIGNAL; light++) {
        digitalWrite(LIGHTPINS[signal][light], OFF);
    }
}

void toggleLightsTest() {
    lightsRunTestStatus = (lightsRunTestStatus == LIGHTSTESTING) ? LIGHTSRUNNING : LIGHTSTESTING;
    testSignalNumber = 0;
    testLightNumber = 0;
    allLightsOff();
}

void toggleGatesTest() {
    gatesRunTestStatus = (gatesRunTestStatus == GATETESTING) ? GATETESTING : GATERUNNING;
    for (byte gate = 0; gate < NROFGATES; gate++) {
        gateStatus[gate] = OPENING;
    }
}

//in main loop testLights is called frequently to blink all lights of all signals one after the other
void testLights() {
    long previousTestBlinkMillis;
    if ((millis() - previousTestBlinkMillis) >= TESTLIGHTINTERVAL) {
        previousTestBlinkMillis = millis();
        digitalWrite(LIGHTPINS[testSignalNumber][testLightNumber], LOW);
        testLightNumber++;
        if (testLightNumber >= NROFLIGHTSPERSIGNAL) {
            testLightNumber = 0;
            testSignalNumber++;
            if (testSignalNumber >= NROFSIGNALS) {
                testSignalNumber = 0;
            }
        }
        digitalWrite(LIGHTPINS[testSignalNumber][testLightNumber], HIGH);
    }
}

//set certain signal to new aspect and set lights in that signal accordingly
void setSignal(byte signal, byte aspect) {
    signalStatus[signal] = aspect;
    for (int light = 0; light <= NROFLIGHTSPERSIGNAL; light++) {
        if ((LIGHTS[country][aspect][light] == ON) || (LIGHTS[country][aspect][light] == OFF)) {
            digitalWrite(LIGHTPINS[signal][light], LIGHTS[country][light][aspect]);
        }
    }
}

void blinkLights() {
    long currentBlinkMillis = millis();
    if (currentBlinkMillis - previousBlinkMillis >= BLINKINTERVAL) {
        previousBlinkMillis = currentBlinkMillis;
        for (byte signal = 0; signal < NROFSIGNALS; signal++) {
            for (byte light = 0; light < NROFLIGHTSPERSIGNAL; light++) {
                if (LIGHTS[country][signalStatus[signal]][light] == BLINK) {
                    digitalWrite(LIGHTPINS[signal][light], blinkStatus);
                }
            }
        }
        blinkStatus = (blinkStatus == HIGH) ? LOW : HIGH;
    }
}

void moveGates() {
    long currentMillis = millis();
    if ((currentMillis - previousGateMillis) >= gateInterval) {
        previousGateMillis = currentMillis;
        for (byte gate = 0; gate < NROFGATES; gate++) {
            if (gateStatus[gate] == OPENING) {
                gatePosition[gate] = (gatePosition[gate] + GATEINCREMENT);
                if (gatePosition[gate] > GATEMAXPOSITION) {
                    if (gatesRunTestStatus == GATERUNNING) {
                        gateStatus[gate] = OPEN;
                    } else
                        gateStatus[gate] = CLOSING;
                }
            }
        }
    }
}

```

```

    } else {
        myservo[gate].write(gatePosition[gate]);
    }
} else if (gateStatus[gate] == CLOSING) {
    gatePosition[gate] = (gatePosition[gate] - GATEINCREMENT);
    if (gatePosition[gate] < 0) {
        if (gatesRunTestStatus == GATERUNNING) {
            gateStatus[gate] = CLOSED;
        } else
            gateStatus[gate] = OPENING;
    } else {
        myservo[gate].write(gatePosition[gate]);
    }
}
}
}
}

void loop() {
    if (Serial.available() > 0) {
        String inputCommand = Serial.readString();
        byte firstLetterCommand = inputCommand[0];
        byte signalOrGateNumber = inputCommand[1] - '0' - 1;
        switch (firstLetterCommand) {
            case 'T': toggleLightsTest();
                break;
            case 'G': toggleGatesTest();
                break;
            case 'N': country = NL;
            case 'D': country = D;
                break;
            case 'R': setSignal(signalOrGateNumber, BRAKE);
                break;
            case 'A': setSignal(signalOrGateNumber, DRIVE);
                break;
            case 'H': setSignal(signalOrGateNumber, STOP);
                break;
            case 'O': gateStatus[signalOrGateNumber] = OPENING;
                break;
            case 'S': gateStatus[signalOrGateNumber] = CLOSING;
                break;
        }
    }
    if (lightsRunTestStatus == LIGHTSTESTING) {
        testLights();
    }
    moveGates();
    blinkLights();
    gateInterval = map(analogRead(POTENTIOPIN), 0, 1023, MINGATEINTERVAL, MAXGATEINTERVAL);
}

```

Beroepsproduct - Aansluiting

aansluiting_1.jpeg
aansluiting_2.jpeg