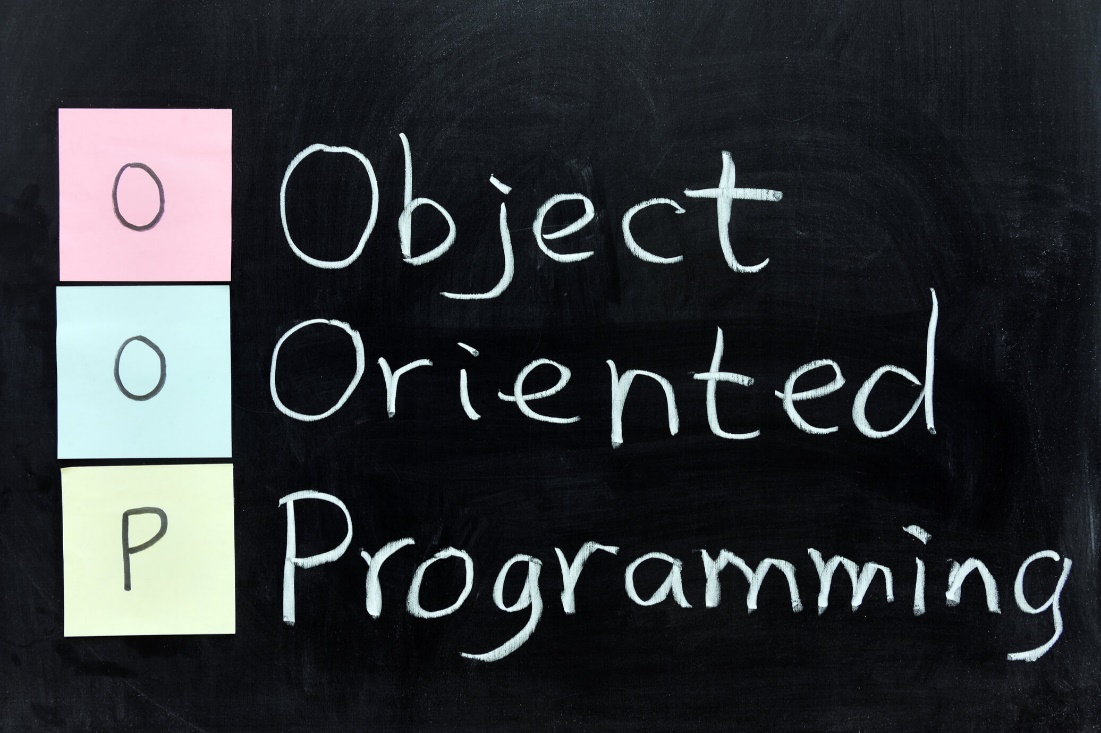
**Startsemester  
Verdiepingsverslag  
Technology  
  
**

**Studentnaam: Marijn Verschuren  
Studentnummer: 510936**

**Klas: PD11  
Vakdocent: Rop Pulles  
  
Versie: 1.0   
Datum: 25-12-2021**

**Versiebeheer**

|  |  |  |  |
| --- | --- | --- | --- |
| **Versienummer** | **Datum** | **Auteur** | **Veranderingen** |
| *V1.0* | *18-12-2022* | *Marijn Verschuren* | *TinkerCad design, Breadboard design, Arduino tester, blinking led, debounce, trafic control, IO techniques (digital, analog, UART/ USART)* |
| *V1.1* | *19-12-2022* | *Marijn Verschuren* | *CAN, I2C, SPI* |
| *V1.2* | *20-12-2022* | *Marijn Verschuren* | *Sensors, Serial plotter* |
|  |  |  |  |
|  |  |  |  |

Inhoudsopgave

Inhoud

[Inhoudsopgave 3](#_Toc121313077)

[1 Inleiding 4](#_Toc121313078)

[1.1 Aanleiding 4](#_Toc121313079)

[1.2 Onderwerp 4](#_Toc121313080)

[1.3 Leeswijzer 4](#_Toc121313081)

[2 Introductie 5](#_Toc121313082)

[3 Tabel Leeruitkomsten 6](#_Toc121313083)

[4 Challenge x: (omschrijving) 7](#_Toc121313084)

[5 Challenge x (Aansluiten van een LED) 8](#_Toc121313085)

[6 Challenge y: (omschrijving) 9](#_Toc121313086)

[7 Reflectie / evaluatie 10](#_Toc121313087)

[7.1 Waar ben ik trots op? 10](#_Toc121313088)

[7.2 Wat doe ik een volgende keer anders? 10](#_Toc121313089)

[7.3 Welke formatieve indicatie zou ik mezelf geven voor de verdieping Technology? 10](#_Toc121313090)

# Inleiding

//Schrijf hier een inleidend verhaal, de subhoofdstukjes zijn een hulpmiddel, je kunt lege hoofdstukjes ook verwijderen en er 1 tekst van maken.

## Aanleiding

*<…>*

## Onderwerp

*<….>*

## Leeswijzer

*<…>*

# Introductie

*// Schrijf hier wat over jezelf, voornamelijk mbt tot school en bij voorkeur over je kennis op het gebied van programmeren en Arduino.*

*<…>*

# Tabel Leeruitkomsten

Je ontwikkelt en programmeert interactieve embedded systemen, waarbij sensoren en actuatoren

toegepast worden, die verschillende I/O technieken gebruiken.

**Verdiepend niveau**

Object oriënted programmeren

* Wat zijn objects, classes en hoe gebruik ik die?
* Wat is encapsulatie en hoe gebruik ik constructors, private fields, properties en methods?
* Hoe maak ik leesbare, onderhoudbare en robuuste programma's?

Sensoren en Actuatoren

* Hoe doe ik een analyse naar sensoren en actuatoren en hoe pas ik deze toe?

I/O technieken

* Wat is pulsbreedtemodulatie, analoge input en wanneer en hoe pas ik deze toe?

Communicatie tussen Arduino en een ander systeem

* Hoe definieer en gebruik ik een protocol met parameters voor de communicatie tussen een Arduino en een ander systeem?
* Hoe zorg ik dat het de communicatie robuust is tegen ongeldige berichten?

Eigen project

* Kan ik alle geleerde concepten in samenhang d.m.v. een eigen project toepassen?

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Object Oriented Programming (OOP) | Classes | Objects | Constuctors | Private Fields | Properties | Methods | Encapsulatie | Robuuste progrs | Sensoren en actuatoren | Analyse sensoren | IO Technieken | PWM | Analoge Input | Communicatie met ander systeem | Protocol | Robuste communicatie |
| Challenge 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Challenge 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| etc… |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Eigen project |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

# Tinker CAD:

**Challenge Beschrijving**:

1. Go to https://www.tinkercad.com/ and create an account and sign in.

2. Choose the ‘Circuits’ section to go to create or edit a circuit.

3. Find a place the Arduino with breadboard in the working area (search for ‘arduino uno’).

4. Find and place the desired components and connect them in the proper way to the breadboard.

5. Select Code->Text to write C code.

6. Click ‘Start Simulation’ to start the simulation.

**Aanpak:**  
Ik heb eerst het circuit opgebouwd en daarna de code gemaakt en getest

**Wat heb ik geleerd?** Ik heb geleerd om tinker CAD te gebruiken

**Resultaat:**Text

Description automatically generated with medium confidence

De volgende twee regels in de code die enigszins bijzonder zijn:

|  |
| --- |
| PORTB ^= (1 << 5); // toggleing pin B5 or 13 which is defined as DIGITAL\_LED\_PIN |

In deze regel wordt een pin telkens hoog en laag geschakeld dit heb ik via een GPIO register gedaan omdat Arduino geen toggle functie heeft. In dit geval heb ik de port en het bit index van pin 13 opgezocht in het pinout diagram (PORTB, PIN5). Daarna word een xor uitgevoerd op bit 5 van PORTB hierdoor word pin13 geschakeld.

|  |
| --- |
| analogWrite(PWM\_LED\_PIN, map(analogRead(POT\_PIN), 0, 0x3ff, 0, 0xff)); |

In deze regel word de analoge pin die van de potmeter afgelezen dit geeft een waarde van 0 tot 1023 dit word omgerekend naar een waarde van 0 tot 255. Deze waarde wordt daarna uitgeschreven naar de PWM pin van de led.

# Breadboard Circuit

**Challenge Beschrijving**:   
Build up the same circuits as above on a real breadboard as above and test them. Do you see any difference?

**Aanpak:**Ik heb het circuit van Tinker CAD gebouwd op een breadnoard en de code geupload

**Resultaat:**

A picture containing text

Description automatically generated

Het nagebouwde circuit had maar een verandering en dat was het bit nummer in PORTB omdat ik een Ardiuno Mega2560 gebruik (code).

# Arduino tester

**Challenge Beschrijving**:   
How nice would it be to be able to quickly test your Arduino? At least you want to know if all digital inputs, digital outputs and all analog inputs are working. Try to think of an efficient way with not too many steps for the tester.

**Aanpak:**Ik test de pins door ze aan te zetten en dan te checken of deze daadwerkelijk aan zijn gezet door te kijken in de PIN registers

**Uitleg:**

Voor deze pin tester heb ik geen externe onderdelen nodig gehad maar als ik een verzie zou maken met een led, button en potentiometer zou ik eerst de led aanzetten en vervolgens het programma laten wachten op een button press waarna een bericht gestuurd zal worden naar de serial console waar in staat wat de waarde van de potmeter was en welke pins nu getest kunnen worden. Ik heb in dit geval voor een simpele software oplossing gekozen waarin de pins alleen getest worden op digitale output (PWM inbegrepen) maar dit is alsnog boeiend voor analoge inputs omdat de pin simpelweg doorgebrand kan zijn hier word dus wel voor getest. Deze tester is dus niet geschikt voor het testen van de ADC peripheral.

Text

Description automatically generated

Deze code zet alle pins die niet in de exclude array staan op high en checkt of deze ook daadwerkelijk op high zijn gegaan door eerst de pin als output te configureren. Dit word gedaan door bits in de DDR registers high te zetten deze DDR registers bevinden zich op IO\_BASE + 1 + 3n. Vervolgens worden deze pins op high gezet door bits in de PORT register high te zetten deze PORT registers bevinden zich op IO\_BASE + 2 + 3n. Dan worden deze pins als input geconfigureerd. Ten slotte word de status afgelezen van de PIN register in IO\_BASE + 3n. Als de bits in de PIN registers niet hetzelfde zijn als in de PORT registers word er een bericht gestuurd naar de serial console.

**Resultaat:**





Door pin A1 (F1) laag te pullen met een 1k resistor word het volgende bericht gestuurd:



Error bij pin 2 in port F (er word geteld van 0) dus dat klopt met A1 (F1).

# Non-Blocking blinking led

**Challenge Beschrijving**:   
write a program that blinks one LED x times each second and another LED y times each second. x and y can have any value > 0

**Aanpak:**Ik heb twee timers opgezet om de lampjes om de zoveel tijd aan en uit te zetten

**Uitleg:**

Text

Description automatically generated

In dit geval is LED\_A\_FREQ 2 en LED\_B\_FREQ 5 (in Hz). Ook hier worden de pins getoggled via de PORT registers.

Als meerdere leds wil blinken zou ik een 2d array maken waarin telkens led\_pin, freq, last\_pulse in staat waardoor alle leds via éen functie geüpdatet kunnen worden door de index door te geven. Deze functie kan dan via een loop gecalled worden. Dit heb ik in dit geval niet gedaan omdat ik maar twee leds heb

**Resultaat:**

<https://youtube.com/shorts/ZkLfpoYSBno>

# Non-Blocking debounce + Button function

**Challenge Beschrijving**:   
Using the knowledge acquired about millis(), have a look at https://www.arduino.cc/en/Tutorial/BuiltInExamples/Debounce and test this using one button and a LED toggle.

When this works, create your own button function so you can handle buttons easily the rest of your embedded career!

Test this function with at least two buttons!

**Aanpak:**Ik heb een class gemaakt voor de buttons waarin alle variables voor de debounce zitten ook heb ik een function pointer toegevoegd waardoor ik een function kan callen als de state switcht hierdoor is het switchen van de leds heel makkelijk en hoef je alleen maar de buttons te updaten in de loop.

**Test resultaat:**

[https://youtube.com/shorts/-94sOB9wIrg](https://youtube.com/shorts/-94sOB9wIrg?feature=share" \t "_blank" \o "https://youtube.com/shorts/-94sOB9wIrg?feature=share)

**Uitleg:**

Voor het debounce circuit heb ik transistors gebruikt omdat ik maar een werkende button heb voor de rest werkt alles hetzelfde. Text

Description automatically generated

Dit is de code snippet van de button class deze class houd elke variable voor het debouncen en een function pointer die gecalled word als de state veranderd.



In deze code snippet word de function pointer type defined voor de button class

A screenshot of a computer

Description automatically generated with low confidence

Hier worden de buttons geinializeerd met de pin number delay en functie die gecalled word als de state veranderd

Text

Description automatically generated

In de loop worden de buttons geupdated en de toggle functies defined. Hier zie je ook dat de leds alleen getoggled worden als de button\_state high is en word dus alleen uitgevoerd op rising edge.

# Trafic control

**Challenge Beschrijving**:   
Create a traffic control program using state machine programming!

**Aanpak:**Ik heb een class gemaakt om de status van elk verkeerslicht bij houd daarna heb ik een functie gemaakt die de states van de verkeerslichten switcht. Deze functie word dan aan een button vast gemaakt. Ook worden de states om de 10 seconden automatish geswitched

**Resultaat:**

<https://youtube.com/shorts/K1wBo79kHvg>

**Uitleg:**

Voor de trafic control code heb ik dezelfde button class als bij de vorige challenge gebruikt samen met een nieuwe trafic light class die alle variables en functies voor kleuren veranderen bevat. Het oranje licht duurt 3.5 seconden en het automatiche switchen duurt 10 seconden. (in de code staat overal “Yellow” omdat ik alleen maar gele lampjes had)

Text

Description automatically generated

In dit code snippet zie je een enum met alle mogelijke states en alle variables van de class. De waardes van de states in de enum zijn ook gelijk de index van de pin in de array pins van de Trafic\_Light class. Dit geld alleen niet voor de NONE state omdat het knipperen van oranje meer code nodig heeft.

Text

Description automatically generated

Hier word de state en dus ook de pins aangepast. Hier zie je ook dat de NONE state word omgezet in oranje.

Graphical user interface, text

Description automatically generated with medium confidence

Hier worden alle variables voor transitie geïnstalleerd.

Text

Description automatically generated

En hier word gecheckt of er iets veranderd moet worden.

Text

Description automatically generated

Deze functies worden gecalled als de buttons (sensors) geactiveerd worden.

Text

Description automatically generated

En ten slotte word er in de loop functie alle objects geupdated en gechecked of er een transitie nodig is.

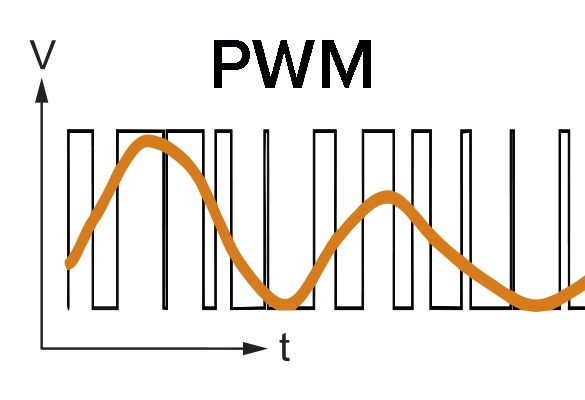
# IO Techniques

**Challenge Beschrijving**:   
For each of the mentioned techniques above, find a source which clearly explains its principle.

For each of the mentioned techniques above, find some sensors or actuators using this technique.

**Uitleg:**

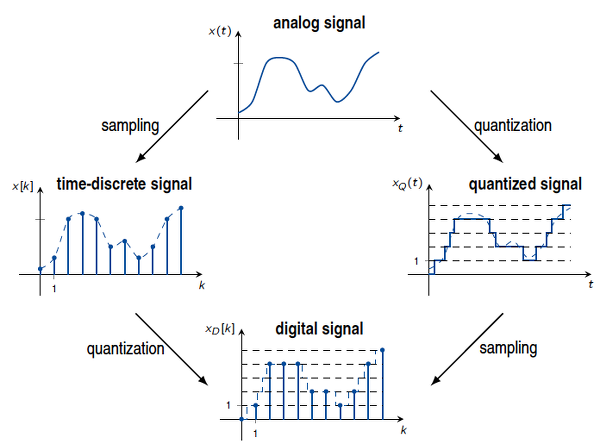
## Digital:

Digital IO is de simpelste techniek en word ook het meest gebruikt met sensors omdat dit (bijna) geen processing nodig heeft en het snelste is. Het enige nadeel is dat het alleen een ON en OFF signaal kan sturen. Er bestaat wek een soort mix tussen Digital en Analoog en dat is PWM (pulse width modulation) met deze techniek kan je meer data versturen door een enkele pin door het signaal voor een gedeeltes van een tijdsperiode aan en uit te laten staan. Door het signaal voor een gedeelte aan of uit te laten staan kan je een soort analoog signaal nadoen (dit is geen echt analoog signaal daarom is het een mix tussen digital en analoog). 

hier zie je ook welk analoge signaal dit PWM signaal benadert

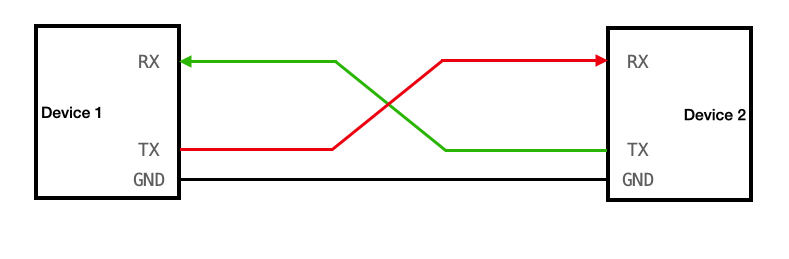
## Analoog:

Een analoog signaal is een signaal met een bepaalde voltage tussen bijvoorbeeld 0v en 5v of 0v en 3.3v. deze voltage kan dan afgelezen worden doormiddel van de ADC peripheral die op meeste microcontrollers zit door de voltage range op te delen en te kijken waar de voltage ligt op deze verdeling.

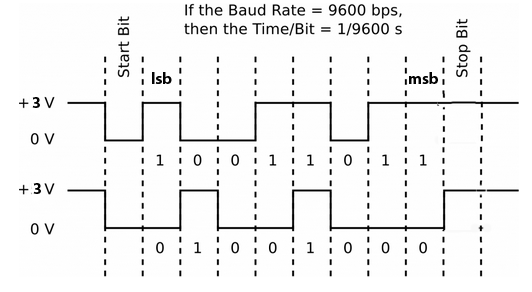


## UART / USART:

UART (universal asynchronous receiver-transmitter) / USART (universal synchronous-asynchronous receiver-transmitter) is een full-duplex data protocol dat betekend dat deze data tegeleikertijd heen en weer kan sturen. Het sturen van deze data wordt bit voor bit via twee of drie draaden gedaan: TX, RX en CLK bij USART. De TX en RX draden moeten omgedraaid worden tussen devices omdat deze bij elk device op de zelfde manier gebruikt word namelijk: de TX voor het versturen van data (transmit) en RX om de data te ontvangen (receive). RX en TX zijn high als ze idle zijn.



De bits worden op een afgesproken snelheid verstuurd en ontvangen. Deze snelheid moet tussen beide devices gelijk zijn omdat omdat anders de bits op het verkeerde moment opgenomen worden deze afgesproken waarde heet de baud rate (of bit rate).



In het geval van USART is een gedeelde clock (CLK draad) toegevoegd waardoor de baud rate niet perse afgesproken hoeft te worden maar door een van de devices (meestal de master) geregeld te worden. Dit zorgt er voor dat er minder fouten gemaakt worden met het opnemen van de bits ook kan de clock uitgerekt worden als een van de microcontrollers het stuur moment niet haalt. Omdat deze clock line voor stabiliteit zorgt kan USART sneller werken dan UART terwijl deze betrouwbaar genoeg blijft.

Rectangle

Description automatically generated with medium confidence

Er zijn nog een paar andere afsplitsingen van het UART/USART protocol zoals bijvoorbeeld RS-232 maar dit wordt bijna niet meer gebruikt dus dit ga ik niet uitleggen.

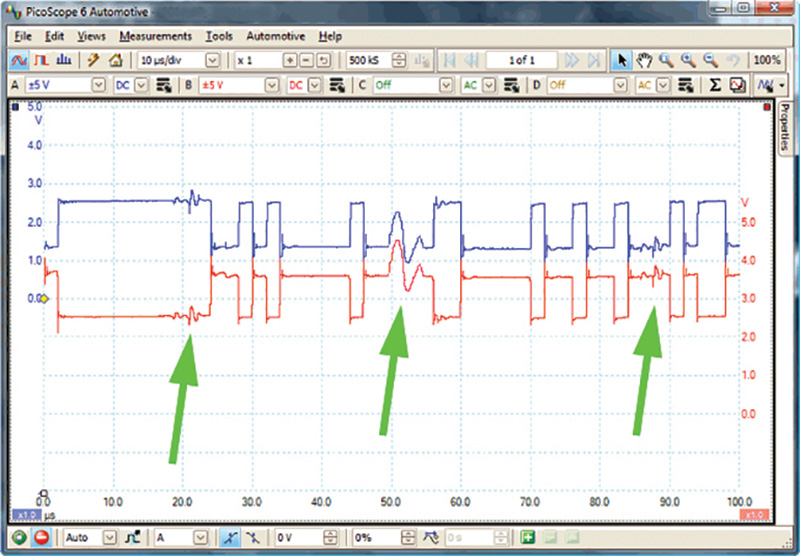
## CAN:

Een CAN (controller area network) bus lijkt heel erg op UART omdat dit ook een asynchrone seriele protocol is. Het grootste verschil is dat CAN een high en low data pin heeft later wordt uitgelegd waarom. Ook is het half-duplex dat betekend dus dat de data maar een kant op kan per keer. Omdat er geen clock pin is moet de baud rate weer afgesproken worden.

A picture containing diagram

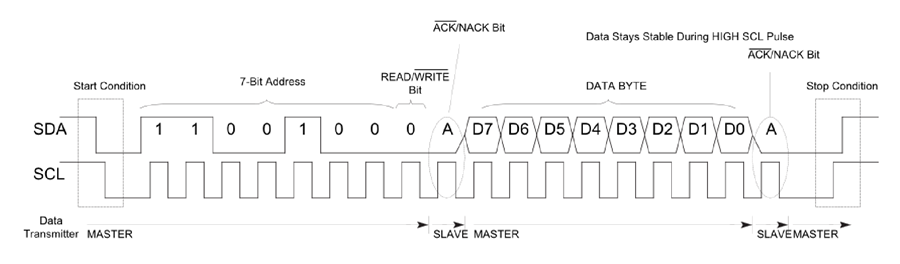
Description automatically generated

Doordat er data low en high pins zijn kunnen errors in het signaal makkelijk gedetecteerd en ontdaan worden door een van de signalen om te draaien en het gemmidelde uit te rekenen. Deze methode werkt omdat de meeste errors in signalen veroorzaakt worden door inductie (waarmee je je telefoon drradloos kan opladen) bij inductie wordt energie draadloos overgedragen door wisselende stroom in een andere draad. Hierdoor wordt dezelfde error op de low en high uitgeoefend omdat deze nu geen overgestelde van elkaar zijn wordt de error bij het omdraaien en optellen uit het signaal gefilterd.



## I2C:

De I2C (inter intergrated circuit) bus lijkt heel erg op de USART omdat ze beide synchronous serial protcols zijn. Het vershil is dat I2C half-duplex is en dat elke transmissie begint met een device adres en een R/nW bit zodat dezelfde I2C bus voor meerdere devices gebruikt kan worden terwijl de per device communicatie behouden kan worden. De I2C bus bestaat uit 2 draaden SCL (serial clock) en SDA (serial data) SCL en SDA zijn high als ze idle zijn. Verder moet na elke byte een acknolagement bit komen van het ontvangende device.



A picture containing text, antenna

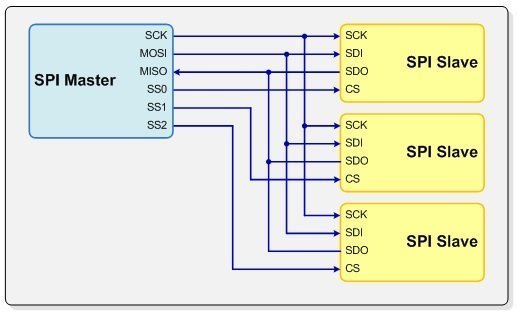
Description automatically generated

## SPI:

SPI is een soort van full-duplex versie van I2C het enige verschil is dat er geen device adressen en R/nW bits gestuurd worden want in plaats van een device adres wordt de CS/SS (Chip Select / Slave Select) pin laag gezet waardoor de chip geselecteerd word. Voor de rest werkt het sturen van data over de twee data pins: MISO (master in slave out) en MOSI (master out slave in) hetzelfde als bij I2C. De data en clock pins zijn voor elke chip gedeeld maar de CS of SS-pin is bij elke chip apart.

Text

Description automatically generated with low confidence



**Sensors:**

## AS5600:

De AS5600 is een magnetische HAL angle sensor die geconfigureerd kan worden via I2C en afgelezen kan worden via I2C of via een Analoog signaal als er geen pullup resistor connected is (R4 op het breakoutboard)

A close-up of a calculator

Description automatically generated with medium confidence

Deze sensor moet geïnstalleerd worden (als er geen setting ingebrand is) dit moet gedaan worden via I2C door de bits in verschillende register aan en uit te zetten dit is waarom hiervoor meestal een library wordt gebruikt jammer genoeg was er geen library voor de STM32F4 boards dus die heb ik gemaakt op basis van de library van de arduino.

Library link: <https://github.com/MarijnVerschuren/STM32F4_AS5600_library>

**Library uitleg:**

De AS5600 heeft de volgende registers:

Table

Description automatically generated

ZPOS bevat de start positie en MPOS de eind positie, MANG bevat de max angle deze registers worden gebruikt om een angle range in de chip te branden (het aantal burn events wordt opgeslagen in ZMCO)

Hierna komt de CONF-registers hierin worden de volgende settings opgeslagen:

Text, table

Description automatically generated with medium confidence

Powermode wordt gebruikt om de sensor op een lagere current te draaien dit kan een negatief effect hebben op de reading.

Hysteresis is een redelijk complex concept dus ik leg alleen uit hoe deze configuratie op de chip werkt. Bij Hysteresis wordt een soort delay toegevoegd waarin een soort gemmidelde wordt gemaakt waardoor snelle veranderingen uit het signaal word gefilterd.

OUTS (zie foto)

PWMF is de PWM-frequentie die uitgezonden wordt als PWM-mode geselecteerd is in de OUTS register

SF en FTH zijn settings voor de measurement filter

WD zet de watchdog aan de watchdog is een functie die om de tijd checkt of de chip nog correct werkt

Nadat de configuratie naar de CONF-register geschreven is wordt de STATUS register afgelezen. Hierin staat de status van het magnetiche veld. Als dit allemaal gelukt is wordt een success ge returnedGraphical user interface, text, application

Description automatically generated

**Library gebruiken:**

Text

Description automatically generated

In deze stap word er een nieuwe instance aangemaakt en bepaalde variables aangepast. Er zijn nog veel meer variables die aangepast kunnen worden maar deze worden in het voorbeeld op de standaard waarden gelaten.

Graphical user interface, text, website

Description automatically generated

Nu de settings in de instance staan worden deze naar de sensor geschreven met deze functie. Deze functie geeft ook een status terug om het programma te laten weten of het goed is gegaan of niet(daarom gebruik ik het meestal in een while loop totdat het succesvol returnt).

Graphical user interface, text, application, chat or text message

Description automatically generated

Nu de sensor gedigitaliseerd is kan deze afgelezen worden via I2C (waar het breakoutboard voor disigned is). Of via analog (werkt alleen als R4 van het breakoutboard is gehaald) met de volgende code:



Hier gebruik ik de DMA om het analoge signaal constant af te lezen en in stata.raw\_angle te stoppen zonder dat ik de CPU bezig hoef te maken. Deze analoge data is niet altijd even goed dus hier is na het ontvangen nog wel wat werk aan.

Resultaat:

<https://www.youtube.com/watch?v=2IKLwBE1TLw>

<https://www.youtube.com/watch?v=rBEvl1Z75_w>

##### BME280:

De BME280 is een temperatuur, vochtigheid en luchtdruk sensor die gebruikt kan worden met I2C en SPI

A picture containing text, electronics, case

Description automatically generated

##### HC-12:

De HC-12 is een lange afstand draadloze transmitter en receiver die aangestuurd moet worden met UART.



# Challenge y: (omschrijving)

**Challenge Beschrijving**:   
<…>

**Flowchart:** <…>

**Aanpak:**<…>

**Resultaat:** (Beschrijving of screenshot (of beide))

<…>

**Uitleg:**

<…>

**Bronnen:**

# Reflectie / evaluatie

## Waar ben ik trots op?

*<vul in>*

## Wat doe ik een volgende keer anders?

*<vul in>*

## Welke formatieve indicatie zou ik mezelf geven voor de verdieping Technology?

|  |  |  |
| --- | --- | --- |
| Leeruitkomst verdieping Technology (bouwt voort op de oriëntatie) | | |
| Onderdeel | Criterium | Rating |
| Interactieve embedded systemen | Je product kan communiceren met een ander systeem volgens een eigen gedefinieerd protocol inclusief parameters waarbij ongeldige berichten worden afgevangen. | *<vul in U S G O met motivatie>* |
| Programmeren | Je past alle imperatieve programmeer-concepten en de volgende OO concepten toe: objects, classes en encapsulation, d.w.z.: constructors, private fields, properties en methods. De focus hierbij is op leesbare (b.v. naamgeving, indentation) en onderhoudbare programma’s en robuustheid van het product. | *<vul in U S G O met motivatie>* |
| Sensoren en Actuatoren | Je past extra sensoren en actuatoren toe waarvan een eigen analyse is gedaan. | *<vul in U S G O met motivatie>* |
| Verschillende I/O technieken | Naast de genoemde I/O technieken kun je ook pulsbreedtemodulatie en analoge input interpreteren en toepassen. | *<vul in U S G O met motivatie>* |