Antwerp Maritime Academy

Arduino in maritieme context

Thomas Zwaenepoel

Scriptie voorgedragen tot het behalen

van de graad van

Master in de Nautische Wetenschappen

aan de Hogere Zeevaartschool

Promotor: Peter Bueken

Copromotor: Carine Reynaerts

Academiejaar 2022-2023

Woord vooraf

Het was in de lessen elektronica van het vijfde middelbaar dat ik een introductie kreeg in Arduino. Na de overstap van de tweede graad Moderne Wetenschappen naar de derde graad Industriële Wetenschappen maakte ik voor het eerst echt kennis met het vak elektronica. Een praktische toepassing kwam er met de deelname van onze school aan het wetenschapsevenement WetenschapsExpoSciences, waar jongeren wetenschappelijke en technologische projecten mogen demonstreren. Het idee van onze groep bestond uit het integreren van windturbines in de vangrails langs de snelweg om hiermee energie te genereren voor de straatverlichting. Al werd ons Arduino experiment niet geselecteerd, het was een boeiend traject. Ik was aangenaam verrast om te vernemen dat ook Arduino centraal staat in het labo elektronica (deel 2) in het derde jaar van de Bacheloropleiding aan de Antwerp Maritime Academy. In de masteropleiding schreef ik me in voor het keuzevak 'scheepsbouw, propulsie en automatisatie', waar Arduino terugkeerde in het seminarie automatisatie. Dit opleidingsonderdeel is dan ook de aanleiding geworden voor deze thesis die bekijkt welke plaats Arduino inneemt binnen de opleiding tot zeevarenden. Graag nog een woord van dank voor de begeleiding en het geduld van mijn promotor Peter Bueken en copromotor Carine Reynaerts bij het schrijven van deze thesis.

Samenvatting

Deze thesis is een onderzoek naar mogelijkheden tot het optimaliseren van het opleidingsonderdeel ‘leren programmeren’ binnen een maritieme opleiding, meer bepaald het seminarie automatisatie in de master aan de AMA. Hiervoor zullen drie maritiem geïnspireerde Arduino projecten en opstellingen gemaakt worden. Deze kunnen gebruikt worden als leerplatform, waarbij studenten de programmeertaal en de verschillende componenten op een aantrekkelijke manier kunnen leren begrijpen. En inzicht krijgen in het belang van ‘leren programmeren’ binnen een maritieme opleiding.

Na een beknopte inleiding met daarin de opzet van de thesis, volgt een schets van het ontstaan en de betekenis van de STCW-code. Volgens die STCW-code is ‘leren programmeren’ binnen een maritieme opleiding een optioneel vak en laat dus ruimte voor interpretatie. We gaan kijken hoe dit aan de AMA in het vak Elektronica (deel 2) wordt onderwezen. Maar vooraleer we dat doen, lichten we microcontrollers en Arduino met al zijn mogelijkheden toe.

Met maritieme artikelen als uitgangspunt worden tot slot drie Arduino projecten voorgesteld. Daarbij wordt gebruik gemaakt van de Arduino microcontroller en verschillende componenten zoals een gas sensor, drie-assige gyro sensor en een ultrasone sensor. De oefeningen worden telkens op dezelfde wijze opgebouwd. Na een toelichting van de verschillende componenten volgt de toepassing.

Abstract

This thesis is a research paper into the possibilities of optimizing the course 'learning to program' within a maritime education, more specifically the automation seminar in the master at the AMA. Three maritime-inspired Arduino projects and setups will be created. These can be used as a learning platform, where students can learn to understand the programming language and the various components in an attractive way. And gain insight into the importance of 'learning to program' within a maritime education.

After a brief introduction outlining the structure of the thesis, an outline of the origin and meaning of the STCW code follows. According to that STCW code, 'learning to program' is an optional subject within maritime training and therefore leaves room for interpretation. We will look at how this is taught at the AMA in the subject of Electronics (Part 2). But before we do that, we explain microcontrollers and Arduino with all its possibilities.

Finally, using maritime articles as a starting point, three Arduino projects are presented. In these, the Arduino microcontroller and various components are used, such as a gas sensor, three-axis gyro sensor and an ultrasonic sensor. The exercises are built up each time in the same way. After an explanation of the various components, the application follows.

Inhoudstafel

Lijst van Figuren

Lijst van Tabellen

Lijst van afkortingen

ADC	: Analog to Digital Converter
ALU	: Arithmetic Logic Unit
AMA	: Antwerp Maritime Academy
AREF	: Analog Reference
ARM	: Advanced RISC Machines
AVR	: Alf and Vegard's RISC
CISC	: Complex Instruction Set Computer
CPU	: Central Processing Unit
DAC	: Digital to Analog Converter
EEPROM	: Electrically Erasable Programmable Read Only Memory
FAC	: Facilitation Committee
GPIO	: General Purpose Input/Output
ICSP	: In Circuit Serial Programming
IDE	: Integrated Development Environment
IMO	: International Maritime Organization
IoT	: Internet of Things
IR	: Infrarood
ISA	: Instruction Set Architecture
LCD	: Liquid Crystal Display
LDR	: Light Dependent Resistor
LED	: Light Emitting Diode
LEG	: Legal Committee
LNA	: Low Noise Amplifier
MARPOL	: International Convention for the Prevention of Pollution from Ships
MEMS	: Micro-Elektromechanisch Systeem
MEPC	: The Marine Environment Protection Committee
MSC	: Maritime Safety Committee
NTSB	: National Transportation Safety Board
PIC	: Peripheral Interface Controller
PWM	: Pulse-Width Modulation
RAM	: Random-Access Memory
RISC	: Reduced Instruction Set Computer
ROM	: Read-Only Memory
SCL	: Serial Clock
SDA	: Serial Data
SOLAS	: International Convention for the Safety of Life at Sea
SRAM	: Static Random-Access Memory
STCW	: Standards of Training, Certification and Watchkeeping for Seafarers
TDR	: Temperature Dependent Resistor
USB	: Universal Serial Bus
VCC	: Common Collector Voltage

1 Inleiding

De opleiding Nautische Wetenschappen bestaat uit een breed aanbod van exacte en toegepaste wetenschappelijke cursussen. Niet van alle vakken wordt door de studenten het belang in het kader van een opleiding tot zeevarende begrepen. Zo is van het vak elektronica (deel 2) in de derde jaar van de bacheloropleiding het verband tussen de theorie en de praktijk aan boord niet of onvoldoende duidelijk voor de studenten. In de masteropleiding is er de mogelijkheid om het keuzevak 'scheepsbouw, propulsie en automatisatie' op te nemen. Binnen dit vak handelt het seminarie automatisatie specifiek over Arduino. Een onderzoek naar de mogelijkheden tot het optimaliseren van dit opleidingsonderdeel zal het onderwerp vormen van deze master thesis.

Deze thesis bestaat uit vijf delen. Ten eerste worden de voorschriften van het STCW-verdrag besproken met specifieke aandacht voor het luik programmeren. Ten tweede nemen we microcontrollers onder de loep, met extra aandacht voor Arduino waarmee lesgegeven wordt aan de Antwerp Maritime Academy (AMA). In hoofdstuk 5 volgt een uiteenzetting van het labo elektronica (deel 2) aan de AMA.

In het laatste deel wordt programmeren in de maritieme context bekeken. Hier gaan we kijken hoe we een Arduino project kunnen maken binnen een maritieme context.

2 Standards of Training, Certification and Watchkeeping

2.1 Internationale Maritieme Organisatie

De *International Maritime Organization* (IMO) of Internationale Maritieme Organisatie heeft tot doel de scheepvaart zo veilig en milieuvriendelijk mogelijk te maken. Ze doet dit aan de hand van internationale verdragen met haar deelnemende lidstaten en regelt ook administratieve en wettelijke aangelegenheden gerelateerd aan haar doelstelling. De IMO is een zogenaamde gespecialiseerde organisatie van de Verenigde Naties, werd in 1948 opgericht en trad in 1958 in werking.

Het hoogste orgaan van de IMO is de Algemene Vergadering, waar alle lidstaten deel van uitmaken en die elke twee jaar samenkomt in een zitting. De Algemene Vergadering beslist over programma, budget en financiën en kiest ook de Raad, het op een na belangrijkste, uitvoerend orgaan van de IMO. Vier belangrijke commissies maken ook deel uit van de organisatie. Het MSC, MEPC, LEG en FAC zijn commissies, gespecialiseerd in respectievelijk veiligheid, milieu, legale en facilitaire zaken. Samen met nog een aantal subcommissies worden ontwikkelingen in de scheepvaart en maritieme industrieën voortdurend opgevolgd. Daaruit kan de behoefte aan nieuwe verdragen of aanpassingen aan bestaande verdragen ontstaan. Aan het laten aannemen van dergelijke verdragen gaat doorgaans een lang proces vooraf omdat alle betrokken regeringen hun akkoord moeten geven. Bekende voorbeelden hiervan zijn SOLAS, MARPOL en STCW (IMO, 2019).

2.2 STCW-verdrag

Het eerste verdrag dat op internationaal niveau basisvereisten vastlegde over opleiding, certificering en wachtdienst voor zeevarenden was de *International Convention on Standards of Training, Certification and Watchkeeping for Seafarers* of afgekort STCW. Dit verdrag, aangenomen in 1978, ging van kracht in 1984 en wijzigde in achtereenvolgens 1995 en 2010. Vóór het eerste STCW-verdrag bepaalde elk land eigen normen en procedures met vele onderlinge verschillen tussen de landen als gevolg. Het gebrek aan uniformiteit kwam de veiligheid van de scheepvaart, toch een internationale industrie, niet ten goede.

Bij de wijziging in 1995 kwam naast het STCW-verdrag ook de STCW-code tot stand, waarin de bepalingen uit het verdrag nader gespecifieerd worden. De STCW-code bestaat uit twee delen. Deel A is verplicht en bevat de minimumnormen die vereist zijn voor zeevarend personeel op professionele en commerciële schepen. Deel B is niet verplicht maar aanbevolen en bestaat uit aanvullingen en richtlijnen voor de praktische toepassing van de bepalingen van het STCW-verdrag. De gegeven voorbeelden vormen een handleiding en zijn tot stand gekomen door overleg binnen het IMO en andere internationale organisaties (IMO, 2018).

2.3 Model Course 7.03

Na de goedkeuring van het STCW-verdrag in 1978 kwam ook het voorstel van een aantal IMO-lidstaten om in modelopleidingen te voorzien.
Een reeks cursussen, de IMO *Model Course*, kreeg vorm. Deze cursussen hebben als doel de maritieme opleidingsinstituten van de IMO-lidstaten te helpen bij de organisatie van hun opleidingsprogramma en de voortdurende actualisering ervan. Want de snel evoluerende maritieme technologie vereist voortdurend verbetering en aanvulling van bestaande trainingen om kwaliteit en doeltreffendheid te garanderen.

Zeevaartopleidingen verschillen behoorlijk van land tot land en hun instructeurs zijn onmisbaar voor het overbrengen van kennis en vaardigheden. De IMO *Model Course*s hebben niet de bedoeling om deze bestaande opleidingen door lesgevers te vervangen door audiovisueel of voorgeprogrammeerd materiaal. De modelcursussen zijn als leidraad bedoeld en kunnen flexibel worden ingepast binnen bestaande opleidingsprogramma's. Ook wordt de cursusduur aangegeven als een indicatie van de verwachte tijd die nodig is om de vereiste kennis van een onderdeel te verwerven. De modelcursussen bestaan uit een inleiding, een cursuskader, een algemene en gedetailleerde tekst, een handleiding voor het gebruik van de cursus en een hoofdstuk betreft evaluatie. Cursussen worden regelmatig bijgewerkt en aangepast.

Binnen het aanbod modelcursussen gaan we hier dieper in op *Model Course* 7.03, *Officer in Charge of a Navigational Watch* en meer bepaald appendix 2. Hierin wordt het onderwerp van fysica behandeld, vereist voor het behalen van STCW vaarbevoegdheidsbewijzen A-II/1 en A-II/2, respectievelijk *Officers in Charge of a Navigational Watch on Ships of 500 gross Tonnage or more* en *Masters and Chief Mates on Ships of 500 gross Tonnage or more*.

In dit appendix is een hoofdstuk aan elektronica gewijd dat optioneel is in de opleiding tot *Officer in Charge of a Navigational Watch.* Het bevat eenvoudige en praktische proeven als ondersteunend lesmateriaal bij de theoretische cursus elektronica. In de lesmomenten in

het labo worden eenvoudige schakelingen gemaakt met een printplaat. Het begrijpen van het functioneren van de verschillende schakelingen en componenten is hierbij belangrijker dan de technische details ervan (International Maritime Organization, 2014).

Omdat het opleidingsonderdeel elektronica en meer bepaald programmeren optioneel is. Is een onderzoek naar een mogelijke optimalisatie van het opleidingsonderdeel elektronica binnen een maritieme opleiding is dan ook op zijn plaats.

2.4 Zeevarenden leren programmeren

Leren programmeren is niet meteen het eerste opleidingsonderdeel dat men verwacht binnen een traditioneel vormingstraject voor zeevarenden. Maar zoals in vele andere sectoren nemen ook binnen de scheepvaartsector de digitalisatie en de afhankelijkheid van nieuwe technologieën sterk toe. Die snelle en ingrijpende evolutie in de professionele maritieme wereld heeft ook gevolgen voor de opleidingen van de mensen die er in de toekomst werkzaam in willen zijn. Uiteraard blijft een klassieke vorming voor zeevarenden onmisbaar en vereist de scheepvaart nog steeds traditionele vaardigheden zoals gedreven en toegewijd werken in een team, praktisch handelen en probleemoplossend denken.

Maar wie in de moderne maritieme wereld aan de slag wil zal in contact komen met steeds complexere systemen, zowel aan boord van schepen als aan de wal. Systemen die niet enkel bediend, maar ook onderhouden en zo nodig hersteld moeten worden. Het is dan ook niet ondenkbaar dat in de op technologisch vlak snel evoluerende zeevaart, ook vaardigheden zoals programmeren en kunnen coderen tot de essentiële vaardigheden voor zeevarenden zullen gaan behoren (Eve Jones, 2020).

Over een bepaalde technische kennis beschikken zal dan ook voor zeevarenden steeds meer een troef zijn. Want wie met deze kennis bepaalde problemen kan helpen oplossen zal, zeker in geval van nood, een daadwerkelijke aanwinst voor een schip zijn. Dit blijkt ook uit het feit dat sommige rederijen zowel hun nieuwe rekruten als bemanningsleden in dienst een opleiding in programmeren aanbieden (Martide blog, 2019).

3 Microcontroller

Een microcontroller is een geïntegreerde schakeling die verschillende elementen omvat, waaronder een microprocessor, timers, tellers, input/output (I/O) -poorten, *random access memory* (RAM), *read-only memory* (ROM) en enkele andere componenten. Deze onderdelen werken samen om een reeks specifieke voorgeprogrammeerde taken uit te voeren. Een microcontroller is dus als een kleine computer die de besturing van een elektronisch apparaat verwerkt en zelfs uitvoert.

Een *development board* zoals ontwikkeld door Arduino wordt vaak gelabeld als microcontroller. Maar Arduino wordt echter nauwkeuriger omschreven als een ontwikkelingsplatform dat draait om gemakkelijk programmeerbare *boards* met microcontrollers als belangrijkste onderdeel. De Arduino zelf is slechts een uitgebreid bord dat alle componenten van een microcontroller bevat, samen met enkele extra poorten en andere functies. Dit geeft de mogelijkheid de microcontroller te programmeren en te herprogrammeren voor verschillende taken.

Om te begrijpen waarom microcontrollers een zo belangrijk onderdeel vormen van onze moderne elektronicawereld, moeten we alleen maar naar enkele apparaten te kijken die voor ons direct toegankelijk zijn. Stel, een wagen heeft de functionaliteit van automatische ruitenwissers. Deze ruitenwissers gaan automatisch aan bij regenbuien en gaan ook weer automatisch uit als het ophoudt met regenen. Hoe kan een zo specifieke en repetitieve functie nu tot stand komen?

In zo'n systeem is een microcontroller verantwoordelijk, die precies voor dit scenario is voorgeprogrammeerd. Input van de sensoren die op de voorruit zijn aangesloten, wordt geleverd aan en verwerkt door de CPU, die vervolgens die signalen vergelijkt met het programma dat al in het geheugen van de microcontroller is opgeslagen. En als de microcontroller de input krijgt dat het regent, geeft hij het commando aan de bijbehorende uitgang, in dit geval de ruitenwissermotor. Zo verwerkt de microcontroller niet alleen de

invoer, maar stuurt hij ook de uitvoer in het systeem aan. En al deze elementen zijn ingebouwd in een enkele chip die soms kleiner is dan een vingernagel.

Zoals eerder vermeld, zijn microcontrollers een soort kleine computer op een chip. En zoals bij normale computers, vormen verschillende elementen belangrijke onderdelen van een microcontroller en helpen ze bij het functioneren ervan. Laten we elk van deze elementen kort bespreken (Gharge, 2022).

3.1 Basisbegrippen van microcontrollers

- RAM

De RAM wat de afkorting is voor *Random-Access Memory,* is het tijdelijke opslaggeheugen dat informatie alleen opslaat als de stroom is ingeschakeld. Het helpt bij het uitvoeren en berekenen van de programma's die de microcontroller moet uitvoeren. Het wordt voortdurend overschreven terwijl het in gebruik is.

- ROM

De ROM wat de afkorting is voor *Read-Only Memory*, is het programmageheugen in een microcontroller, waarin alle programma-instructies zijn opgeslagen. De CPU haalt zijn instructies op van de ROM en voert ze één voor één uit. In tegenstelling tot RAM slaat ROM noodzakelijkerwijs gegevens op, zelfs nadat het apparaat is uitgeschakeld. Bij microcontroller *development boards* heeft men het vaak over EEPROM en flash. EEPROM, afkorting voor *Electrically Erasable Programmable Read Only Memory*, is geheugenruimte die programmeurs kunnen gebruiken om informatie op lange termijn op te slaan. Een groot voordeel van EEPROM is dat het gewist en herschreven kan worden. Flashgeheugen (programmaruimte), is waar de microcontroller-schets wordt opgeslagen. (Teja, 2021)

- CPU

De CPU wat de afkorting is voor *Central Processing Unit,* staat vooral bekend als het brein van de microcomputer. De centrale verwerkingseenheid of gewoonweg de processor neemt invoer op, interpreteert ze en voert bewerkingen uit, allemaal volgens het programma in het geheugen. Het werkt samen met RAM en ROM om snel en met hoge efficiëntie de gewenste output te produceren.

- I/O POORTEN

De I/O poorten wat staat voor *Input/Output* poorten, bestaan uit een of meerdere communicatiepoorten, meestal in de vorm van verbindingspennen. Hiermee kan de microcontroller worden verbonden met andere componenten.

- TIMERS & TELLERS

Timers & tellers of *timers & counters* zijn de elementen die de meest uiteenlopende toepassingen in de microcontroller vinden. Ze worden gebruikt om alle timing- en telfuncties van de microcontroller te besturen. De timer kan bewerkingen uitvoeren zoals klokfuncties, puls generaties, modulaties, meetfrequentie of oscillaties maken. De belangrijkste functie van de teller is het tellen van externe pulsen.

- ADC

De ADC, afkorting voor *Analog to Digital Converter*, zet de analoge ingangssignalen van de externe sensoren om in digitale signalen. Dit is nodig omdat de CPU digitale invoer nodig heeft om bewerkingen uit te voeren. Op deze manier vormt de ADC een brug tussen analoge ingang en de CPU.

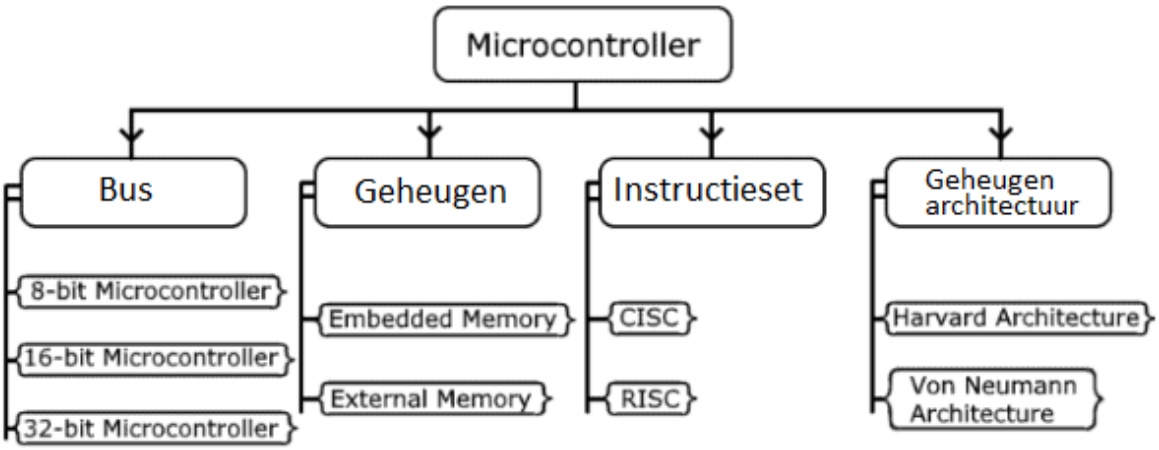
- DAC

De DAC, afkorting voor *Digital to Analog Converter,* is het omgekeerde van de ADC en zet digitale signalen om naar analoge uitgangen. Deze worden vervolgens gebruikt om externe apparaten te bedienen.(*jf-parede*, 2022; *Tesca Technologies Ltd.*, 2021)

3.2 Microcontrollers - types en soorten

3.2.1 Types microcontrollers

Er bestaan verschillende types microcontrollers die ingedeeld worden op basis van een aantal criteria. Hierbij een overzicht.



Afbeelding 1 Indeling Microcontrollers

Bron: *Electricaltechnology*, 2020

- Bus

Bus verwijst over het algemeen naar de parallelle lijnen die de verschillende componenten verbinden. De functie is om gegevens te verzenden tussen CPU, geheugen en invoer-/uitvoerpoorten. Microcontrollers bevatten drie soorten bussen: databus, adresbus en besturingbus. De grootte van de bus is belangrijk omdat deze bepaalt hoeveel data (in bits) tegelijk kan worden verzonden. Een grotere bus verbetert de CPU-prestaties aanzienlijk. Er bestaan microcontrollers van 4, 8, 16, 32 of zelfs meer bits. Op basis van bus worden drie types onderscheiden: 8, 16, 32 bits microcontrollers.

- 8 bits microcontroller

De bus breedte van dergelijke microcontrollers is 8 bits (1 byte) breed. Het betekent dat het de gegevens van 8 bits in een enkele cyclus kan overbrengen en verwerken. De wiskundige bewerkingen vormen hierbij de grootse hindernis. In een 8 bits microcontroller is de ALU (1) ook 8 bits. Om een grote hoeveelheid gegevens van bijvoorbeeld 16 bits te

1 De *Arithmetic Logic Unit* (ALU) of Rekenkundig-logische eenheid is het centrale onderdeel van de processor dat rekenkundige en logische bewerkingen uitvoert.

verwerken, gebruikt het meerdere cycli om een eenvoudige wiskundige functie te voltooien. Het resulteert in een slechte prestatie van de algehele logische schakeling.

Een ander belangrijk kenmerk van de 8 bits microcontroller is de timer. Een 8 bits timer heeft een maximaal bereik van 0x00 (0) tot 0xFF (255). Het kan onnauwkeurigheid veroorzaken bij het genereren van tijdvertragingsfuncties.

- 16 bits microcontroller

De bus breedte is 16 bits (2 bytes) breed. Het kan gegevens van 16 bits in een enkele cyclus overbrengen en verwerken. De 16 bits ALU is efficiënter in prestaties in vergelijking met de 8 bits microcontroller. En de 16 bits timer biedt een breed bereik van 0x0000 (0) tot 0xFFFF (65535), wat de beste nauwkeurigheid biedt voor elke toepassing of projecten die timerfuncties vereisen.

- 32 bits microcontroller

Een 32 bits microcontroller heeft een bus van 32 bits (4 bytes) breed. De prestaties en nauwkeurigheid van een dergelijke microcontroller zijn hoger dan die van een 16 bits microcontroller, maar ze zijn ook duur en verbruiken veel stroom. De hogere verwerkingssnelheid maakt het de beste kandidaat voor het uitvoeren van complexe taken zoals bij voorbeeld audio- en videosignaalverwerking of verwerken van afbeeldingen.

- Geheugen
  - *Embedded Memory Microcontroller*

Bij microcontrollers met ingebouwd geheugen zijn alle essentiële geheugenblokken of modules in een enkel pakket ingebouwd. Sommige van deze functionele blokken zijn programma- en gegevensgeheugen of timers en tellers. Deze geheugenblokken zijn vast en niet vervangbaar, maar een microcontroller die externe ROM ondersteunt, kan zijn opslaggeheugen uitbreiden. Deze zijn compact van ontwerp.

  - *External Memory Microcontroller*

Bij microcontrollers met extern geheugen zitten niet alle essentiële geheugenblokken in de chip. Om goed te kunnen functioneren moeten deze extern worden aangesloten. Het gebruik van externe modules vergroot de fysieke omvang van het totale apparaat.

- Instructieset

Een *Instruction Set Architecture* (ISA) maakt deel uit van het abstracte model van een microcontroller dat definieert hoe de CPU wordt bestuurd door de software. De ISA fungeert als een interface tussen de hardware en de software en specificeert zowel wat de processor kan doen als hoe het wordt gedaan. Volgens de instructieset worden de microcontrollers in twee typen ingedeeld. (Arm Ltd, 2022)

- CISC

CISC staat voor *Complex Instruction Set Computer.* Een microcontroller van dit type beschikt over een CPU dat ontworpen is om een enkele complexe opdracht uit te voeren. Hij kan meerdere instructies of stappen uitvoeren aan de hand van één enkele instructie. Het voordeel van de CISC-microcontroller is het kleine programma. Maar vanwege de grote omvang van de instructieset met veel adresseringsmodi, moet het meerdere machinecycli uitvoeren en duurt het langer om uit te voeren. Bijkomend probleem is dat in CISC parallelle uitvoering van een instructie niet mogelijk is.

- RISC

RISC staat voor *Reduced Instruction Set Computer*. De CPU van dit type microcontroller is ontworpen om kleinere eenvoudige instructies uit te voeren. Aangezien er één machinecyclus nodig is om een enkele instructie uit te voeren, kan het aantal instructies worden verminderd om een complexe taak met een hogere snelheid uit te voeren dan een CISC (Thornton, 2018).

- Geheugen-architectuur
  - Harvard-architectuur

De op Harvard-architectuur gebaseerde microcontroller heeft fysiek gescheiden geheugenopslag voor programmacode en de gegevens, respectievelijk bekend als de ROM- en de RAM- geheugen. Ze hebben dus afzonderlijke buslijnen en zijn beide tegelijkertijd toegankelijk. Daarom kan de op Harvard-architectuur gebaseerde microcontroller een instructie in één cyclus voltooien.

- Von Neumann-architectuur

De architectuur van Von Neumann of Princeton stelt voor om één geheugen te gebruiken voor zowel de programma- als de gegevensopslag. Dit concept werd in 1945 voorgesteld door de wiskundige Von Neumann en het is de meest gebruikte architectuur tot nu toe in alle computers. Er is slechts één bus nodig om toegang te krijgen tot de gegevens en voor het ophalen van instructies. Beide bewerkingen kunnen dus niet tegelijkertijd worden uitgevoerd en moeten worden gepland. Dit is de reden waarom de op de architectuur gebaseerde microcontroller van Von Neumann twee machinecycli nodig heeft om een instructie te voltooien (Technology, 2021).

3.2.2 Belangrijkste soorten microcontrollers

- 8051

De 8051 microcontroller van Intel, ontworpen in de jaren ’80, is tot op vandaag een van de meest universeel gebruikte microcontrollers. De 8051 is een 8 bits CISC microcontroller, met 4 kilobytes ROM en een RAM-adresruimte van 128 bytes. Omdat hij gebaseerd is op de Harvard-architectuur, blijven het programmageheugen en datageheugen gescheiden. Dit zorgt voor hoge rekensnelheden en voorkomt overlappingen. 8051 microcontrollers worden in veel toepassingen gebruikt omdat ze gemakkelijk in kleinere projecten kunnen worden geïntegreerd (Dinesh, 2007).

- AVR

AVR microcontrollers zijn chips die je minstens één keer bent tegengekomen als je van elektronica houdt. Ze worden gebruikt in veel ontwikkelborden. Zo gebruikt Arduino de AVR ATmega328P. Het geheugen architectuur is zoals die van de 8051 gebaseerd op de Harvard-architectuur, maar met opmerkelijke verbeteringen. De AVR is gebaseerd op de RISC-instructieset, waardoor ze met veel hogere snelheden kunnen rekenen en toch minder stroom verbruiken dan de 8051(*Tesca Technologies Ltd.*, 2021).

- PIC

PIC staat voor *Peripheral Interface Controller*. Dit type microcontroller werd in 1975 ontwikkeld en wordt vooral gebruikt bij industriële toepassingen. Hij is ook gebaseerd op de Harvard-architectuur en de instructieset van het RISC-type. PIC's hebben ook een kleinere instructieset dan AVR's, waardoor ze gemakkelijker te leren zijn. Enkele zeer beroemde PIC microcontrollers zijn PIC18fXX8, PIC16f88X en PIC32MXX (Gharge, 2022).

- ARM

ARM microcontrollers zijn ontwikkeld door Advanced RISC Machines of ARM Ltd.
ARM Ltd. produceert de microcontrollers niet zelf, maar geeft zijn ontwerpen in licentie aan chipfabrikanten zoals Texas Instruments. ARM microcontrollers bestaan al sinds midden jaren ’80 en zijn erg populair dankzij hun lage kosten, laag stroomverbruik en lage warmteontwikkeling. Je treft ARM Microcontrollers vaak aan in smartphones, laptops, tablets en *embedded systems*. De ARM is gebaseerd op de Harvard-architectuur. ARM heeft een *Instruction Set Architecture* dat gebaseerd is op RISC. De ARM-Cortex is misschien wel de meest populaire ARM microcontroller. De ARM-Cortex is beschikbaar in een 32 bits variant en een 64 bits variant (Gharge, 2022; *Tesca Technologies Ltd.*, 2021).

3.3 Development boards

Development boards of ontwikkelborden helpen systeemontwerpers om projecten eenvoudig en snel te ontwikkelen en te testen. Deze borden bevatten een microcontroller met meestal extra geheugen, invoer- en uitvoerpoorten, leds, schakelaars en enkele andere randapparaten.

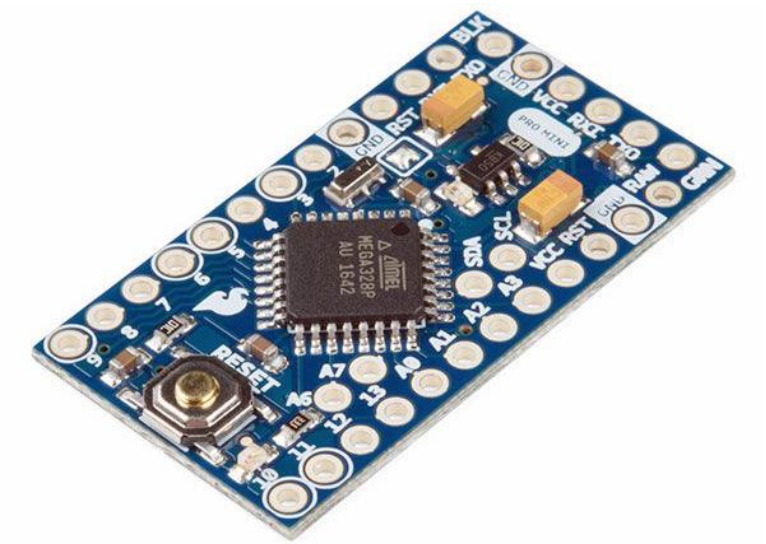
3.3.1 Populaire modellen

1. Arduino UNO R3

Afbeelding 2 Arduino UNO R3
Bron: *Components101*, 2021

Elektronica- en technologiebedrijf Arduino bracht in 2011 de R3-versie van Arduino UNO uit. Het is gebaseerd op de AVR 8 bits ATmega328P microcontroller met kloksnelheid 16 megahertz. De Arduino UNO heeft een flashgeheugen van 32 kilobyte, SRAM 2 kilobyte en een EEPROM van 1 kilobyte. Het bord heeft verschillende I/O-pinnen waarmee je het kunt koppelen aan andere borden en circuits. Er zijn verschillende poorten aangesloten waaronder een USB-aansluitpoort, veertien I/O-pinnen, een ICSP-header (*In Circuit Serial Programming*), een voedingsaansluiting en een resetknop. Het kan eenvoudig via een USB-kabel rechtstreeks op pc of laptop worden aangesloten. De Arduino UNO is een van de populairste modellen.

2. Arduino Pro Mini 328

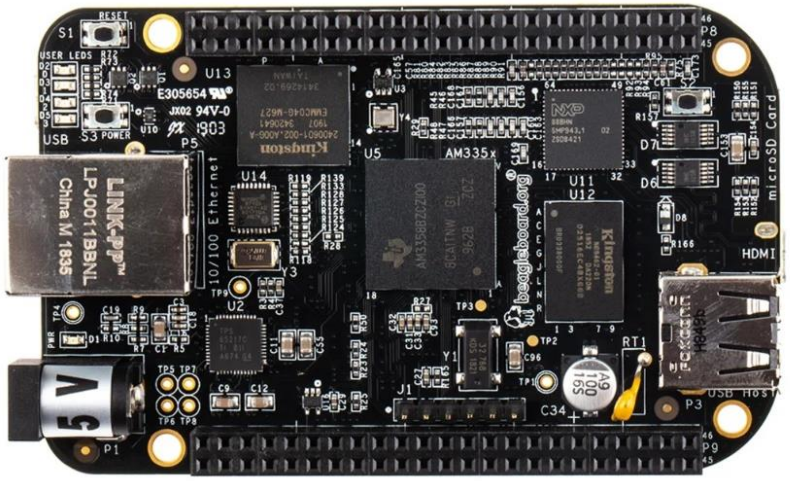


Afbeelding 3 Arduino Pro Mini

Bron: *Components101*, 2021

Arduino Pro Mini is een ander microcontroller bord van Arduino. Dit *miniboard* is alleen bruikbaar voor kleinschalige toepassingen tot 5 V. De Arduino Pro Mini is zoals de Arduino UNO gebaseerd op de AVR 8 bits ATmega328P microcontroller met kloksnelheid 16 megahertz. De Arduino Pro Mini heeft een flashgeheugen van 32 kilobyte, SRAM 2 kilobyte en een EEPROM van 1 kilobyte. Het bord wordt niet geleverd met de ingebouwde aansluitingen en poorten, dus moet men de verbinding zelf solderen. Want als het eenmaal in een applicatie is geplaatst, zijn connectoren in principe nutteloos. Dit bord is eerder geschikt voor gebruik in een permanente applicatie. Voor het kleine budget is dit microcontroller bord een goede keuze.

3. BeagleBone Black



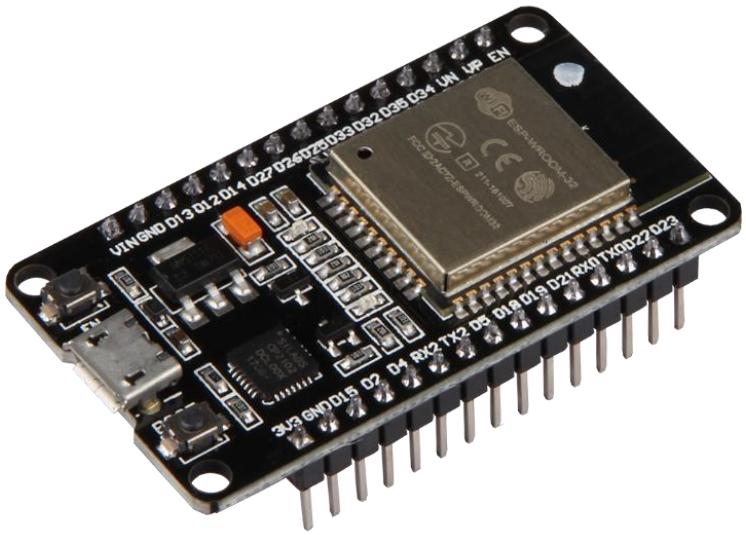
Afbeelding 4 BeagleBone Black

Bron: *Components101*, 2021

BeagleBone Black is een van de duurdere ontwikkelborden. Beaglebone Black is niet minder dan een computer. Het zit vol met alles wat je op een desktop of een laptop vindt. Een krachtige processor, geheugen en grafische versnelling, allemaal verkleind als chips en gesoldeerd in een enkele printplaat. Daarom kunnen we het een *single-board computer* noemen. De Beaglebone Black is gebaseerd op de 32 bits ARM Cortex-A8 met kloksnelheid 1 gigahertz.

Dit krachtige bord kan worden aangesloten op een beeldscherm, luidsprekers, Ethernet-netwerk, toetsenbord en muis. Bovendien kan het worden gebruikt om een LINUX-besturingssysteem op te starten. Het bestaat uit 512 megabyte RAM met 4 gigabyte flash-opslag. Het heeft 46 × 2 aantal header-pinnen, Ethernet, 2 USB-poorten. Het hogere aantal I/O-pinnen maakt het meer geschikt voor elektronicaprojecten. Het heeft ook een lager stroomverbruik zonder dat er koellichamen nodig zijn.

4. NodeMCU ESP32

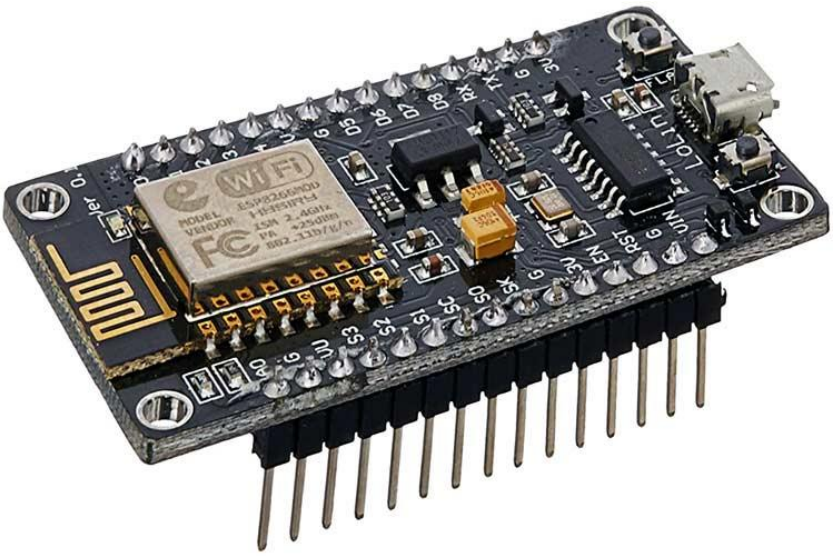


Afbeelding 5 ESP32

Bron: *Components101*, 2021

Het NodeMCU ESP32 microcontroller bord is een bluetooth en wifi combo op een single-chipbord met een ultra laag stroomverbruik. Het bord wordt beschouwd als de beste keuze voor toepassingen waar de beste radiofrequentie prestaties vereist zijn. De ESP32 is gebaseerd op de AVR 32 bits Xtensa LX7 met kloksnelheid 240 megahertz. Het ESP32 microcontroller bord is niet duur en wordt gebruikt voor doe-het-zelf projecten zoals smart home en op IoT gebaseerde projecten. De ESP32 is de opvolger van de ESP8266, maar in elk geval niet een vervanger. Het bevat snellere wifi en meer GPIO's dan de ESP8266 en ondersteunt bluetooth. Met de ESP32 is het mogelijk om via de Arduino IDE te programmeren.

5. NodeMCU ESP8266



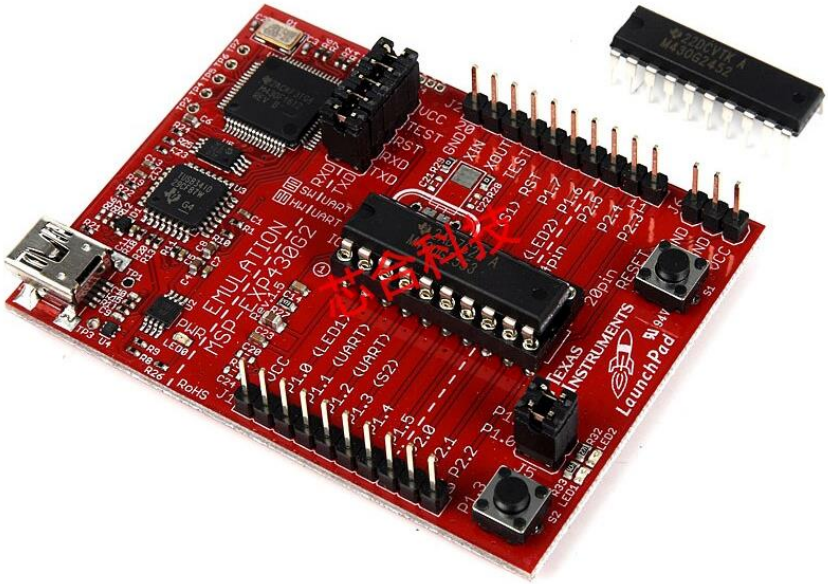
Afbeelding 6 ESP8266

Bron: *Components101*, 2021

De NodeMCU ESP8266 is klein van formaat in vergelijking met andere microcontrollers met IoT-mogelijkheden (2) en is erg budgetvriendelijk. Het kan worden gebruikt voor doe-het-zelf-systemen zoals *smart home* projecten en IoT. De ESP8266 is gebaseerd op de 32 bits Tensilica l106 met kloksnelheid 80 megahertz. Het bevat 128 kilobyte RAM met 4 megabyte flashgeheugen. Wat dit ontwikkelbord interessant maakt, is dat het kan worden gebruikt om zijn eigen netwerk te maken zodat andere apparaten er verbinding mee kunnen maken.

2 Het *Internet of Things* (IoT) bestaat uit fysieke voorwerpen, zoals auto's, huishoudelijke apparaten en microcontroller borden, die met internet zijn verbonden en online gegevens kunnen verzenden.

6. MSP430 Launchpad

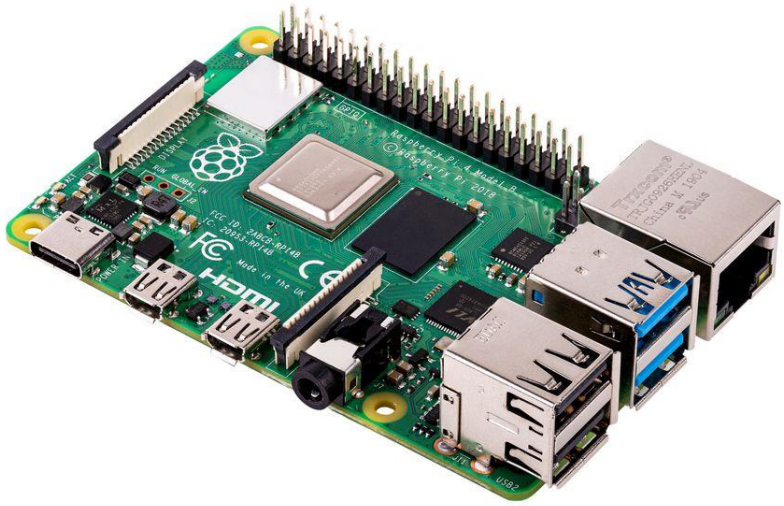


Afbeelding 7 Launchpad MSP430

Bron: *Components101*, 2021

De Launchpad MSP430 is door Texas Instruments ontwikkeld als een extreem energiezuinige 16 bits microcontroller voor gebruik in ingebedde toepassingen met laag vermogen, lage kosten en energiebeperking. De Launchpad MSP430 is gebaseerd op de AVR 16 bits TI MSP430 met kloksnelheid 16 megahertz. Met 512 byte RAM en een flashgeheugen van 16 kilobyte, kan de MSP430 worden gebruikt als een minder krachtig alternatief voor Arduino UNO R3. Het heeft speciale programmeersoftware (IDE) die wordt gebruikt voor codering en debuggen genaamd Energia. Die is gebruiksvriendelijk en lijkt op Arduino's IDE.

7. Raspberry Pi 4



Afbeelding 8 Raspberry Pi 4 B

Bron: Raspberry Pi, 2022

De Raspberry Pi 4, uitgebracht in 2019, is het snelste microcontroller bord dat vandaag beschikbaar is. De Raspberry Pi 4 is gebaseerd op de 64 bits ARM Cortex-A72 met kloksnelheid 1,5 gigahertz. Met een RAM van 1, 2, 4 of zelfs 8 gigabyte kun je krachtige en geavanceerde elektronische projecten bouwen. Raspberry Pi 4 kan tot 1,2 A stroom leveren voor USB-apparaten. Meer functies zijn onder meer *on-board* draadloos LAN, bluetooth 5.0, twee USB 2.0- en USB 3.0-poorten, twee Micro HDMI-poorten en Ethernet-poort. De Raspberry Pi 4 heeft zelf geen interne opslag maar ondersteunt een microSD-kaart tot wel 2 terabyte(Delgado, 2023).

8. STM32 Blue Pill

Afbeelding 9 STM32 Blue Pill

Bron: *Components101*, 2021

Een alternatief voor Arduino is het op microcontrollers gebaseerde STM32-ontwikkelbord, dat vaak de Blue Pill wordt genoemd. Deze microcontroller is gebaseerd op de ARM Cortex-M3 met kloksnelheid 72 megahertz en wordt vervaardigd door STMicroelectronics. De Blue Pill is een zeer krachtige microcontroller en met zijn 32 bits CPU kan hij Arduino UNO gemakkelijk verslaan in prestatie. Pluspunt is ook dat je dit bord kan programmeren met de Arduine IDE. De Blue Pill heeft een flashgeheugen van 64 kilobyte en SRAM van 20 kilobyte.

9. Teensy 4.0

Afbeelding 10 Teensy 4.0

Bron: *Components101*, 2021

Het Teensy 4.0 microcontroller bord is klein van formaat in vergelijking met andere borden. Het kan worden gebruikt om verschillende soorten doe-het-zelf projecten te maken. Teensy 4.0 kan worden geprogrammeerd met de Arduino IDE. De Teensy 4.0 is gebaseerd op de 32 bits ARM Cortex-M7 met kloksnelheid 600 megahertz. De Teensy heeft 1024 kilobyte RAM, wat heel wat meer is dan de 2 kilobyte RAM van Arduino UNO en meer geavanceerde toepassingen mogelijk maakt. Wanneer een applicatie meer nauwkeurigheid en verwerkingskracht nodig heeft, is de Teensy te verkiezen boven de Arduino(Aimal Khan, 2021).

3.3.2 Overzicht

Tabel 1 Overzicht Microcontrollers

Bron: ***Components101*, 2022**

Naam	Kern	Klok	Geheugen	I/O	Prijs
Arduino UNO R3	AVR 8 bits ATmega328P	16 MHz	FLASH 32 kB SRAM 2 kB EEPROM 1 kB	14 GPIO-pins (6 met PWM) 6 analoog	€ 24,00
Arduino Pro Mini 328	AVR 8 bits ATmega328P	16 MHz	FLASH 32 kB SRAM 2 kB EEPROM 1 kB	14 GPIO-pins 8 analoog	€ 9,99
BeagleBone Black rev C	ARM 32 bits TI Sitara AM3358 Cortex A8	1 GHz	SDRAM 512 MB FLASH 4 GB	2x46 GPIO-pins	€ 99,99
NodeMCU ESP32-S	32 bits Xtensa LX7	240 MHz	SRAM 320 kB ROM 128 kB FLASH 1 GB	48 GPIO-pins (25 met PWM) 15 analoog	€ 9,99
NodeMCU ESP8266	32 bits Tensilica l106	80 MHz	SRAM 36 kB FLASH 4 MB	16 GPIO-pins 1 analoog	€ 5,99
MSP430 Launchpad	16 bits TI MSP430	16 MHz	FLASH 16 kB RAM 512 Bytes	16 GPIO-pins	€ 21,35
Raspberry Pi 4 B	ARM 64 bits Cortex-A72	1.5 GHz	SDRAM 2 GB, 4GB of 8 GB Geheugenkaart	40 GPIO-pins	€ 49,95- € 92,30
STM32 Blue Pill	ARM 32 bits Cortex-M3 CPU	72 MHz	FLASH0 64 kB SRAM 20 kB	27 GPIO-pins	€ 13,50
Teensy 4.0	ARM 32 bits Cortex-M7	600 MHz	RAM 1 MB FLASH 2 MB	31 GPIO-pins 14 analoog	€ 34,55

(*Components101*, 2021)

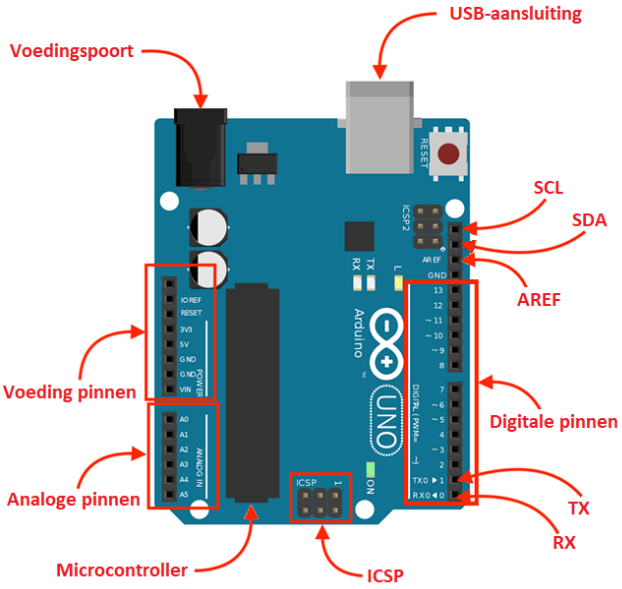
4 Arduino

Het vak elektronica (deel 2) in het derde jaar bachelor aan de AMA wordt aan de hand van Arduino onderwezen. Vooraleer daar verder op in te gaan is een gedetailleerde inkijk in Arduino en meer specifiek de Arduino UNO aangewezen.

Een Arduino is een microcontroller bord dat geprogrammeerd kan worden op basis van een opensource-computerplatform. De microcontroller kan digitale en analoge ingangssignalen ontvangen, verwerken in het programma en omzetten in een digitaal of analoog uitgangssignaal. Ingangssignalen kunnen ontvangen worden door bijvoorbeeld: schakelaars, sensoren, potentiometer of door commando's van een seriële interface. Uitgangssignalen kunnen dan weer zaken aansturen zoals lampen, motoren of beeldschermen. Het Arduino bord maakt gebruik van de ATmega328P single-chip microcontroller met een CPU van 8 bits en flash memory van 32 kB. De microcontroller bevat een *bootloader* functie die ervoor zorgt dat het laatst opgeslagen programma kan worden opgestart zonder externe programmatie.

De Arduino UNO is het model dat aan de AMA wordt gebruikt. Om te starten is het belangrijk om te weten welke capaciteiten elke pin heeft en wat men er mee kan uitvoeren. Deze informatie kan men altijd terugvinden op de website van Arduino (*Arduino.cc*, 2021).

4.1 Hardware



Afbeelding 11 Arduino UNO pin

Bron: Javaid, 2022a

De componenten van Arduino UNO.

- USB-aansluiting

De USB-aansluiting op de UNO heeft twee functies. Een ervan dient voor communicatie, om verbinding te maken met de computer en om de schets in de Arduino te laden. De tweede is om de Arduino van stroom te voorzien.

- Voedingspoort

Het Arduino bord kan worden gevoed met een AC-DC-adapter of een batterij.

Het Arduino UNO bord werkt op een spanning van 5 V, maar is bestand tegen een maximale spanning van 20 V. Als het bord van een hogere spanning wordt voorzien, is er een spanningsregelaar (deze zit tussen de voedingspoort en de USB-connector) die het bord beschermt tegen doorbranden.

- Microcontroller

De Arduino UNO maakt gebruik van de ATMega328P microcontroller. Dit is een single-chip microcontroller gemaakt door Atmel in de AVR-familie. Het heeft een gemodificeerde Harvard-architectuur 8 bits RISC-processorkern. Het combineert 32 kB flashgeheugen met lees- en schrijfmogelijkheden, 1 kB EEPROM, 2 kB SRAM. Dit is het brein van de Arduino.

- Analoge pinnen

Het Arduino UNO bord heeft zes analoge pinnen, A0 tot en met A5. Deze pinnen kunnen het signaal lezen van een analoge sensor zoals bijvoorbeeld een vochtigheidssensor. Deze pinnen hebben een ADC-mogelijkheid die een analoog signaal kan omzetten in een digitale waarde. De ADC is een 10 bits ADC, wat betekent dat het een digitale waarde geeft in het bereik van 0 - 1023. Dit is het aantal waarden dat het kan produceren over het bereik van analoge waarden (0 V-5 V).

- Digitale pinnen

We kunnen deze pinnen vinden met het label 'Digital 0 tot 13'. Deze pinnen kunnen worden gebruikt als invoer- of uitvoerpinnen. Bij gebruik als ingangspinnen lezen ze de signalen van de aangesloten component. Bij gebruik als uitgang fungeren deze pinnen als voedingsbron voor de aangesloten componenten. Ze leveren 40 mA stroom bij 5 V. Sommige digitale pinnen zijn gelabeld met het (~) symbool. Deze pinnen fungeren als normale digitale pinnen, maar kunnen ook worden gebruikt voor pulsbreedtemodulatie (PWM), die analoge uitvoer simuleert, zoals het dimmen van een led.

- Voeding pinnen

We weten al dat we voeding kunnen aansluiten op de Arduino via een adapter en een USB. Er is nog een derde manier, via de Vin pin. We hebben ook 2 voeding pinnen (3,3 V en 5 V) om componenten van stroom te voorzien. Ten slotte zijn er nog de 3 GND pinnen die verantwoordelijk zijn voor de elektronische circuitverbinding met aarde.

- SCL en SDA

De poorten SCL en SDA zijn datalijnen die geconnecteerd kunnen worden met apparaten met een I2C bus (uitleg I2C zie 6.2.1.1). Dit is ook een vorm van informatie-uitwisseling tussen Arduino UNO en externe modules. De analoge pin A4 is ook een SDA pin en A5 is ook een SCL pin.

- TX en RX

Op de Arduino UNO zitten ook een TX (*transmitter*) en een RX (*receiver*) poort. Deze poorten dienen om seriële communicatie door te sturen naar en te ontvangen van een aparte module zoals radio of bluetooth modules. Bij gebruik van deze poorten moet de TX-poort van de Arduino UNO aangesloten worden op de RX-poort van de module.

- AREF

De AREF (*Analog Reference*) is een pin die kan gebruikt worden wanneer men wil werken met een lagere spanning dan de normale voedingsspanning van 5 V. De AREF-pin kan dan aan een lagere voedingsspanning worden aangesloten, bijvoorbeeld 1,5 V.

- ICSP

De ICSP-pin heeft als functie een *master/slave* regeling mogelijk te maken. Hierdoor kan er informatie worden uitgewisseld met een andere Arduino of uitbreidingsmodules (Vasudhendra Badami, 2016).

4.2 Software

Bij het programmeren met Arduino wordt een IDE of *Integrated Development Environment* gebruikt. Dit is de software waarin gewerkt zal worden om een programma te schrijven. Een IDE bevat een *code-editor*, een *compiler* en een *debugger*, waartoe de ontwikkelaar toegang heeft via een grafische *interface*.

De Arduino IDE is geschreven in Java en gebaseerd op Processing, een opensourcesoftware met als doel niet-programmeurs de basis van computerprogrammeren in een visuele context bij te brengen. De Arduino IDE maakt het gemakkelijk om een schets te schrijven en naar de Arduino te uploaden en kan met elk type Arduino worden gebruikt. De Arduino IDE draait op Windows, Mac OS X en Linux. De Arduino IDE ondersteunt de talen C en C++ en wordt geleverd met een set standaardbibliotheken voor veelgebruikte functionaliteiten.

De grootste kracht van Arduino zit eerder in de software dan in de hardware. Hoewel er nogal wat microcontrollers op de markt zijn, is men er met de Arduino het best in geslaagd om de complexe en soms onoverzichtelijke details van het programmeren met microcontrollers in een gebruiksvriendelijk verpakken geheel.

Er is een aanzienlijk aanbod van IDE's op de markt. Als men bijvoorbeeld met Raspberry Pi gaat programmeren bestaat er niet zoiets als een standaard IDE. Men moet dus op zoek gaan naar de beste IDE voor het geplande project, en dit in functie van programmeervaardigheid, gewenste programmeertaal, enz (CompareCamp, 2019).

4.3 Programmeertaal

De programmeertaal die Arduino gebruikt, lijkt erg op C en C++, een populaire taal in de computerwereld. De code die men leert schrijven voor Arduino lijkt op de code die men in een andere computertaal schrijft. De basisbegrippen blijven dezelfde, men zou het eenvoudig kunnen vergelijken met leren werken met een ander dialect.

Bij het schrijven van een programma voor microcontrollers zoals Arduino zijn prestaties erg belangrijk. Daarom zijn sterke en snelle programmeertalen nodig. C en C++ behoren tot de krachtigste talen die er zijn. Omwille van hun snelheid en ook stabiliteit zijn ze een goede keuze voor microcontrollers (‘Elektronica voor Jou’, 2018).
”De programmeertaal C ligt dicht bij de machine en ook dicht bij de mens. Dit betekent dat machines C goed kunnen begrijpen en de programmeertaal is leesbaar voor mensen”(Dingemans, 2020).

4.4 Tinkercad

Tot slot nemen we Tinkercad nog onder de loep.

4.4.1 Wat is Tinkercad

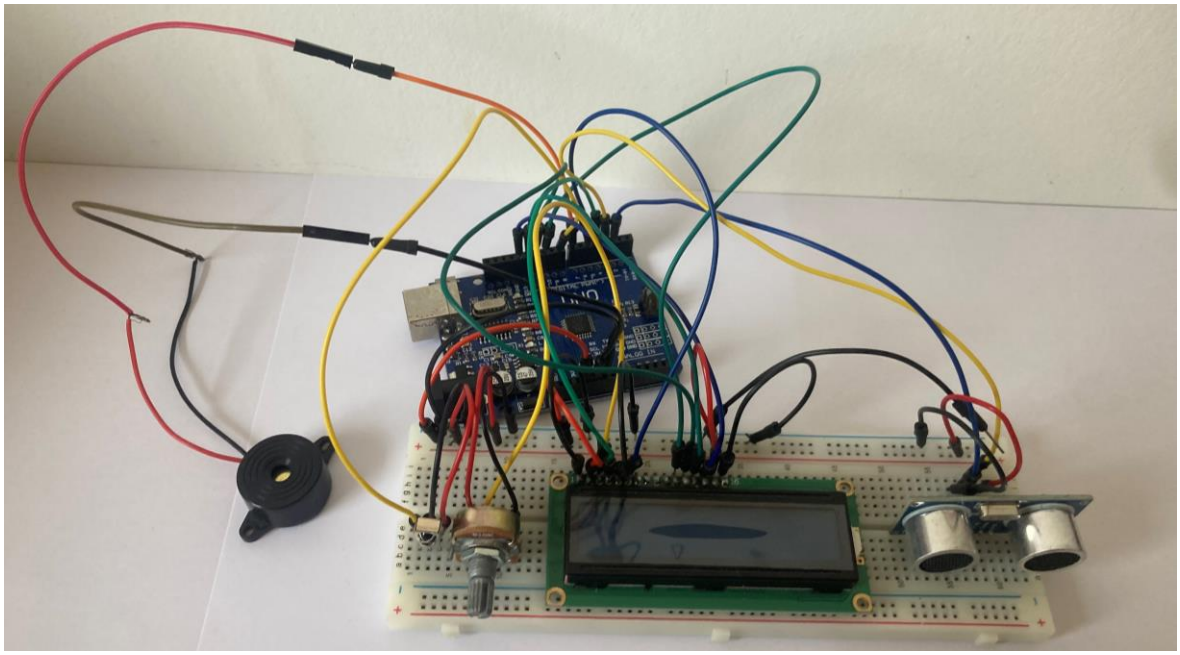
Tinkercad is een veelzijdige online applicatie, en bovendien gratis. Deze bijzonder gebruiksvriendelijke app is geschikt voor 3D-ontwerp, elektronica en codering en wordt gebruikt door leraren, studenten, hobbyisten en ontwerpers. Je kan er elektronische circuits en projecten mee ontwerpen, modelleren en uitvoeren. Met de *drag-and-drop interface* en de intuïtieve ontwerptools kunnen zowel beginners als gevorderden snel en gemakkelijk aan de slag.

Bovendien zijn in Tinkercad zelfstudie lessen beschikbaar die zich richten op de basisprincipes van de Tinkercad tool waardoor je deze software snel kan leren gebruiken.

4.4.2 Een praktisch voorbeeld

Hoe handig de Tinkercad app kan zijn bij het opstellen van schema's zal duidelijk worden met volgend voorbeeld. Als uitgangspunt nemen we de elektronische schakeling uit de oefening in hoofdstuk 6.3.2.4.

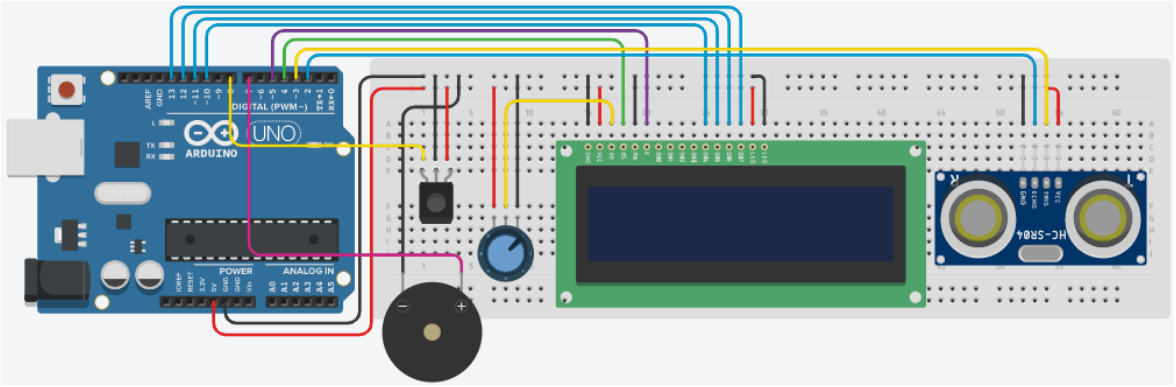
Op onderstaande afbeelding zie je dat deze schakeling niet echt duidelijk is.



Afbeelding 12 Foto oefening 6.3.2.4

Bron: Eigen werk

Met Tinkercad kunnen dergelijke circuits gemodelleerd en gevisualiseerd worden. Dit vergemakkelijkt het begrijpen en analyseren van componenten en verbindingen en is bovendien een stuk aangenamer voor zowel studenten als lesgever. Hoe overzichtelijk het resultaat is als we dit toepassen op de schakeling uit de oefening is te zien op onderstaande afbeelding 13. Bovendien kunnen deze schematische weergaven geëxporteerd worden als afbeeldingen, waardoor ze kunnen worden gebruikt in verslagen.



Afbeelding 13 Schema oefening 6.3.2.4 met Tinkercad

Bron: Eigen werk met *Tinkercad*, 2022

Kortom, als je op zoek bent naar een krachtige en gebruiksvriendelijke tool om te werken met Arduino, dan is Tinkercad zeker de moeite waard.

Wij zijn vooral geïnteresseerd in de toepassing van elektronica en codering, genaamd Tinkercad Circuits. (*ISTE*, 2022).

4.4.3 Tinkercad Circuits

Ideaal voor het werken met Arduino bestaat er nu ook Tinkercad Circuits.

Tinkercad Circuits is een eenvoudige manier om studenten op weg te helpen met elektronica en meer bepaald Arduino. Met behulp van de interactieve circuiteditor kunnen studenten virtuele projecten verkennen, verbindingen maken en coderen met tal van gesimuleerde componenten. Beschikbaar op elke computer met een internetverbinding door de webapplicatie. Tinkercad Circuits is ook een ideale tool bij het schrijven van een verslag. Zo kunnen we de schakeling die we maken op een heldere en overzichtelijke manier weergeven. (*Tinkercad-blog*, 2021).

4.4.4 Aan de slag met Tinkercad

Men begint met het aanmaken van een account. Na het inloggen met dit account komt men op het Tinkercad *dashboard.* De rechter kolom toont verschillende mogelijkheden waarop geklikt kan worden.

Lessen: Tinkercad heeft een functie waar men kan leren van lessen die door verschillende gebruikers online zijn gezet en die hier te vinden zijn.

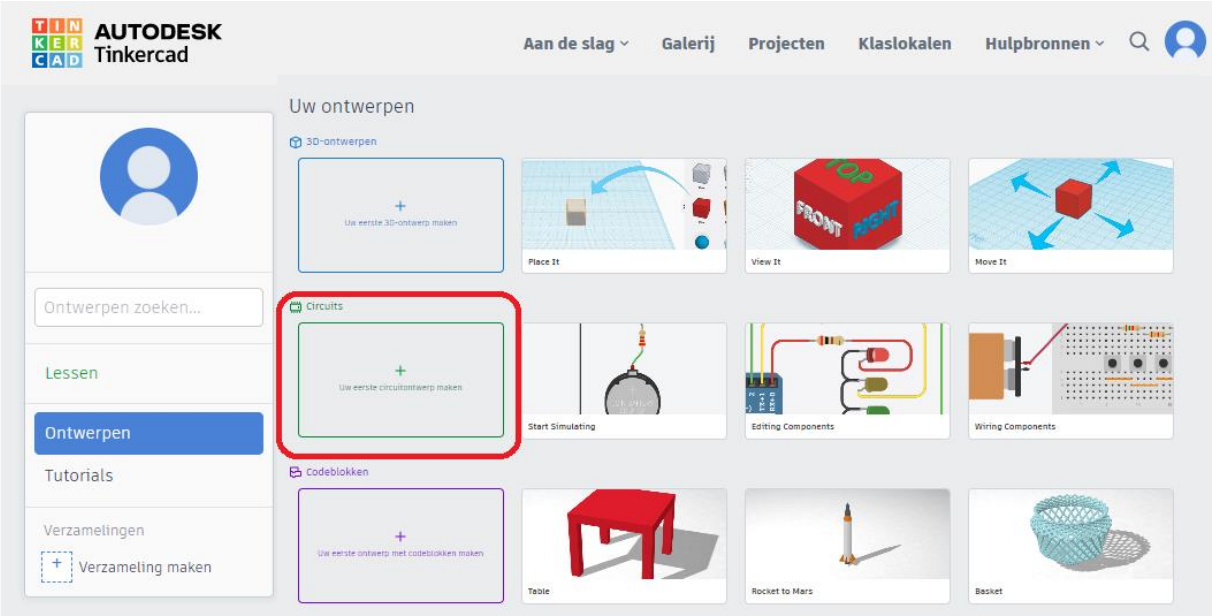
Ontwerpen: Tinkercad heeft een functie waarmee we kunnen ontwerpen. Deze functie bestaat uit 3 onderdelen.

3D-ontwerp: TinkerCAD is vergelijkbaar met CAD-software waarmee 3D-modellen kunnen ontworpen worden voor 3D printen. De CAD-software is gebaseerd op Constructive Solid Geometry (CSG), waarmee gebruikers complexe modellen kunnen maken door eenvoudig objecten met elkaar te combineren.

Circuits: Dit onderdeel is bijzonder interessant. Het is de optie die helpt bij het maken, het programmeren en het testen van een virtueel circuit.

Codeblokken: Deze nieuwe functie helpt bij het maken van een blokprogrammeerboom. Daarin worden de 3D-modellen stap voor stap gevormd door de overzichtelijke boominstructies te volgen.

Tutorials: Voor de drie bovenstaande onderdelen, 3D-ontwerp, Circuits en Codeblokken, bestaan er lessen waarin de werking van alle functies wordt uiteengezet. In Tutorials zijn deze lessen terug te vinden.



Afbeelding 14 Tinkercad dashboard

Bron : Eigen werk m.b.v. *Tinkercad*, 2022

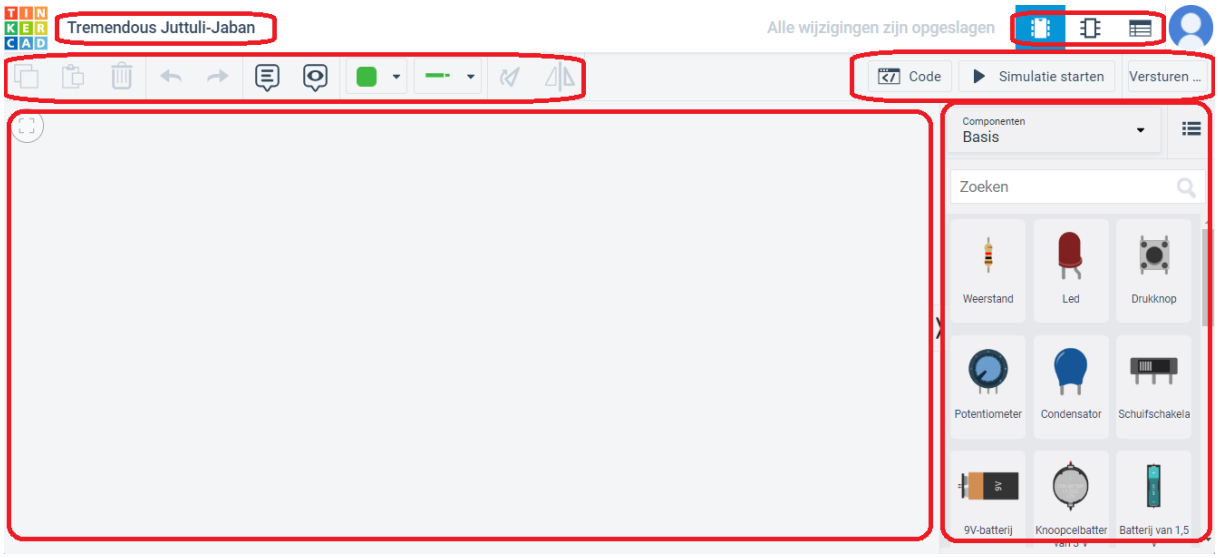
Tinkercad Circuits

We gaan ons eerste circuit maken. We klikken hiervoor op het vak met “Uw eerste circuitontwerp maken”.

1. We krijgen hierdoor onze werkruimte. In de linkerbovenhoek typen we de naam die we aan het project willen geven (in dit vb. Tremendous Juttuli-Jaban).
2. In de rechterbovenhoek zien we drie symbolen. Dit zijn de verschillende opties voor de mogelijke weergaven van ons project. De eerste optie is de schakelweergave, handig om onze schakeling te maken en aan te passen. De tweede optie is de schematische weergave die, zoals de naam het zegt, de schakeling in schematische vorm weergeeft. En als laatste hebben we een lijst van alle componenten, waar je eenvoudig alle componenten kan zien die gebruikt zijn in je schakeling.
3. Onder de naam van ons project in de linkerbovenhoek bevindt zich een tabblad met alle opties voor het bewerken ervan. We kunnen hiermee knippen, plakken, deleten, draaien en een acties ongedaan maken. We kunnen hiermee ook notities toevoegen en ze zichtbaar maken of niet. We kunnen de draad en de kleur van de

draad ook aanpassen. Onthoud dat we altijd rood voor 5 V gebruiken en zwart voor de aarding.

4. Onder de weergave opties in de rechterbovenhoek bevindt zich een tabblad met tools voor het programmeren van onze schakeling. Het geeft hulp om het programma te schrijven, het gebruik van een seriële monitor, het starten van een simulatie, het exporteren van de code en het delen van het project.
5. Het grote witte veld is de ruimte is waar we alle componenten zullen plaatsen. Hierin kunnen deze componenten verplaatst of bewerkt worden of met elkaar verbonden worden.
6. De verschillende componenten vinden we op een tabblad rechts naast het bewerkingsveld.



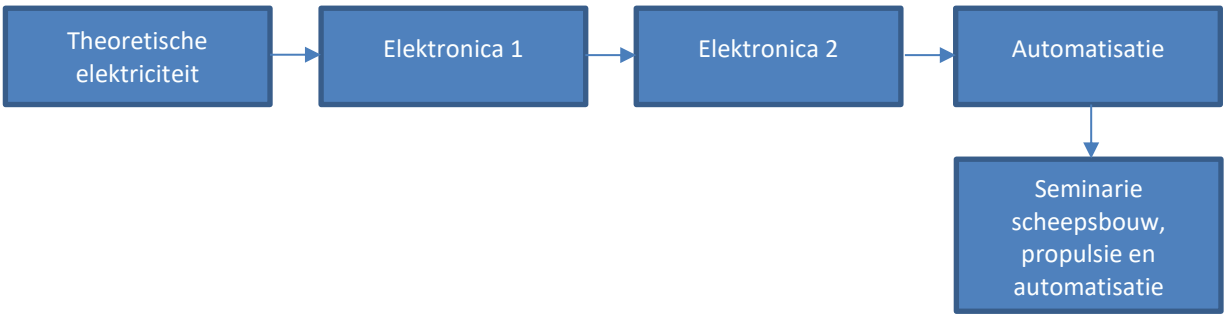
Afbeelding 15 Tinkercad Circuits

Bron: Eigen werk m.b.v. *Tinkercad*, 2022

Vanuit het veld Componenten Basis kan een *dropdown* menu geopend worden dat verschillende basiscomponenten bevat. We weten nu waar alles staat en voor wat het dient. Nu kunnen we aan het werk met Tinkercad Circuits (Maker.io, 2022) .

5 Arduino op AMA

Na de analyse van de specificaties van de Arduino UNO gaan we in dit hoofdstuk bekijken welke weg er wordt afgelegd aan de Antwerp Maritime Academy (AMA) in de context van elektronica. Na een blik op de leerinhoud van de vakken elektriciteit, elektronica en automatisatie , volgt een uiteenzetting van de praktische invulling van het vak elektronica (deel 2), waar er les wordt gegeven met Arduino.



5.1 Leerinhouden voor elektriciteit-elektronica

In het eerste jaar van de opleiding Nautische Wetenschappen krijgen we theoretische elektriciteit. Dit vak bestaat uit twee delen. Elk deel wordt in één module gegeven en wordt gevolgd door een schriftelijk examen. In deel 1 van theoretische elektriciteit maken we kennis met elektrostatica en gelijkstroomtheorie. We leren technieken voor het voorspellen van gedragingen van weerstanden en het berekenen van de verschillende grootheden bij gelijkstroomnetwerken. We gaan continu de leerstof concretiseren aan de hand van voorbeelden en oefeningen.

In deel 2 maken we kennis met het gedrag van condensatoren en met elektromagnetisme en wisselstroomtheorie. We gaan inzicht verwerven in overgangsverschijnselen bij spoelen en condensatoren. We leren technieken voor het voorspellen van het gedrag van componenten en het berekenen van de verschillende grootheden in wisselstroom netwerken. We passen de leerstof toe door middel van voorbeelden en oefeningen. In beide delen verwerven we elektrische kennis, inzichten en vaardigheden ter ondersteuning van andere vakken en van het tot stand brengen van een scriptie.

In het tweede jaar krijgen we het vak elektronica (deel 1). Dit vak bestaat uit twee delen. Een theoretisch gedeelte gevolgd door een schriftelijk examen met mondelinge verdediging en een praktisch gedeelte met labo's en permanente evaluatie. In het theoretisch gedeelte verwerven we een basiskennis van elektronica. We kunnen hiervoor terugvallen op de cursus elektriciteit deel 1 en 2. We krijgen een overzicht van een aantal halfgeleider componenten en hun toepassingen en lossen vraagstukken op met betrekking tot wisselstromen en -spanningen aan de hand van fasoren en impedanties.

In de labo's maken we kennis met een aantal meettoestellen die in elk labo elektronica terug te vinden zijn, zoals multimeter, functiegenerator, gelijkspanningsbron en oscilloscoop. Aan de hand van praktische oefeningen gaan we toepassen wat we gezien hebben in de cursussen Elektriciteit deel 1 en 2, Elektronica (deel 1) theorie.
Voorbeelden van de elektronische circuits die de student(e) zal behandelen zijn: resonantiekringen, gelijkrichting van wisselstroom, versterkers met transistoren en operationele versterkers.

In het derde jaar krijgen we het vak elektronica (deel 2). We krijgen hier een inleiding in het vakgebied van de digitale elektronica. We leren een opsomming van logische poorten en we leren deze gebruiken in combinatorische en sequentiële schakelingen. We bestuderen op een theoretische wijze de basisprincipes van telecommunicatie. We maken kennis met verschillende modulatietechnieken zoals amplitudemodulatie en frequentiemodulatie. We maken ook kennis met enkele technieken gebruikt in digitale communicatie zoals sampling en het moduleren van deze samples. Als laatste leren we gebruik te maken van locusdiagrammen om netwerken te analyseren in het frequentiedomein.

In de labo's elektronica (deel 2) leren we eenvoudige schakelingen bouwen met digitale en analoge sensoren, actuatoren en een microprocessor, en leren we een microprocessor programmeren om deze schakelingen aan te sturen en uit te lezen. In hoofdstuk 5.2.2 gaan we hier dieper op in.

In het master jaar krijgen we het vak automatisatie. Hier maken we ons vertrouwd met de theoretische grondslagen van de regelsystemen gebruikt ter automatisatie van processen.

We leren processen wiskundig uitdrukken door middel van blokschema's en transferfuncties door analytisch denken. We worden vertrouwd gemaakt met de verschillende types van regelaars, hun afstelmogelijkheden en hun realisatie. Daarna krijgen we ook een introductie over de meer moderne technieken zoals PLC’s en microcontroller geregelde systemen.

We krijgen labo’s automatisatie, en leren werken met een leiding en instrumentatie schema. We maken ook kennis met een P&ID regelaar in de machinekamersimulator. Aan de hand van beschreven methodes leren we hoe deze regelaar ingesteld kan worden. Ook de instelparameters van de regelaar in de automatische piloot aan boord van een schip zullen we onderzoeken (HZS, 2021).

5.2 Elektronica (deel 2) labo

In dit hoofdstuk gaan we de labo's elektronica (deel 2) in detail bekijken. In totaal worden er 9 lesuren besteed aan Arduino, verdeeld over 3 labo's van 3 uur. In deze labo's worden een 25-tal voorbeeldoefeningen behandeld.

5.2.1 Introductie

Als introductie is er op Blackboard een document beschikbaar met de titel *A brief Introduction to (*Arduino*) Programming* dat het schrijven van een programma globaal behandelt. Elk Arduino-programma heeft drie hoofdonderdelen. De sectie variabele declaratie, de set-up en de lus-sectie (*loop*).

Worden meestal aan het begin van een programma aangegeven. Alle variabelen die we in een schets gebruiken, moeten hier worden vermeld, vóór de set-up en lus-secties. Een variabele is een plaats om een stuk data op te slaan. Het heeft een naam, een waarde en een type. Bijvoorbeeld deze verklaring "Int pin = 13". Dit creëert een variabele waarvan het type int is, waarvan de naam pin is en waarvan de waarde 13 is. Later in het programma kun je deze variabele bij zijn naam noemen, waarna de waarde ervan wordt opgezocht en gebruikt. Int staat voor *integers (gehele getallen)* en zijn een primair gegevenstype voor nummeropslag.

De functie setup() wordt aangeroepen wanneer een schets wordt gestart. Gebruik het om de variabelen, pin-modi te initialiseren, bibliotheken te gaan gebruiken, enz. De setup-functie wordt slechts één keer uitgevoerd, na elke inschakeling of reset van het Arduino bord. Deze sectie begint met de regel "void setup() {"en eindigt met een gesloten accolade "}". Merk op dat alle programma regels in de setup-sectie tussen de accolades staan.

In de lus functie wordt beschreven wat er gedaan moet worden. Deze sectie begint met de regel "void loop() {". De instructies die in de lus staan worden eindeloos herhaald tot de stroom wordt uitgezet. De lus begint en eindigt altijd met een accolade, maar binnen de lus kunnen kleinere blokken programma staan die weer hun eigen paar accolades hebben. Als je in de Arduino sketch op een accolade klikt laat de IDE je zien waar de tweede staat. (*Tutorialspoint*, 2022)

Extra info geven aan een programma of anderen informeren kan door gebruik te maken van "//**".** Het programma negeert alles op de regel die na de "//" komt. Een reeks instructies die meerdere keren kunnen worden gebruikt, zijn geprogrammeerd in een apart segment dat kan worden opgehaald zonder de hele schets te herhalen. Deze programma's binnen in een programma worden functies of *subroutines* genoemd. Functies kunnen ook buiten het kader van een bepaald programma beschikbaar zijn en zijn gebundeld in zogenaamde bibliotheken (*libraries).*

Vooraleer men een programma kan laten uitvoeren moet het worden gecompileerd. Dat betekent dat het vanuit de programmeertaal vertaald wordt naar de machinecode van de microcontroller. Tegelijkertijd wordt het programma gecontroleerd op syntactische fouten, *errors*. Wanneer fouten optreden volgt een proces van opsporen en oplossen van mogelijke fouten, debuggen. Wanneer ten slotte alle instructies correct uitgevoerd zijn, zou men de gewenste output moeten verkrijgen (Bueken et al., 2019; *Tutorialspoint*, 2022).

5.2.2 Praktijk

5.2.2.1 Labo 1

Het eerste labo start met het programma Blink dat men kan uploaden uit de IDE-basisprogramma's. Dit eenvoudige programma laat de ingebouwde led van de Arduino 1 seconde branden en 1 seconde uitgaan waarna dit knipperen zich herhaalt.

In de set-up gaan we een opgegeven pin configureren om zich te gedragen als een uitgang. We doen dit met de functie "pinMode(pin, modus)". De pin in deze code is de ingebouwde led van de Arduino (LED_BUILTIN) en de modus is uitgang (OUTPUT). In de lus functie wordt beschreven wat er gedaan moet worden. We starten met het aanzetten van de led. We doen dit met de functie "digitalWrite(pin, waarde)". De pin is LED_BUILTIN en de waarde is hoog (HIGH). Daarna pauzeren we de code met "delay()" gedurende de tijd (in milliseconden) die als parameter is opgegeven. We gaan de led nu afzetten door de modus naar laag (LOW) in te stellen met terug een pauze. Dit blijft zich herhalen.

```
// de setup-functie wordt één keer uitgevoerd wanneer u op reset drukt of
het bord van stroom voorziet
void setup() {
  pinMode(LED_BUILTIN, OUTPUT); // verklaar de ingebouwde led als een
OUTPUT
}
void loop() { // de loop-functie wordt altijd herhaald
  digitalWrite(LED_BUILTIN, HIGH);// zet de ingebouwde led aan
  delay(1000);                    // pauze van 1 seconde
  digitalWrite(LED_BUILTIN, LOW); // zet de ingebouwde led uit
  delay(1000);   }                // pauze van 1 seconde
```

We kunnen deze code wijzigingen door de tijd aan te passen. Hiermee kunnen we dan bijvoorbeeld een SOS-morse code maken. We doen dit met korte en lange knippers. Kort 200ms hoog en lang 400ms hoog. De code die we krijgen zijn 9 keer een knipper wat de code inefficiënt maakt. We kunnen hier gebruik maken van een functie. Door code in functies te segmenteren, kunnen we modulaire stukjes code maken die een gedefinieerde taak uitvoeren zoals lang of kort knipperen. De functies moeten buiten de accolades van de set-up en lus worden gemaakt. Dit doen we met "void functie_naam() {de instructies die de functie moet doen}".

```
void lang() { maken van functie lang met lange toon
  digitalWrite(LED_BUILTIN, HIGH); // zet de ingebouwde led aan
  delay(400);                      // pauze van 400 milliseconden
  digitalWrite(LED_BUILTIN, LOW); // zet de ingebouwde led uit
  delay(200);   }                 // pauze van 200 milliseconden
```

```
void kort() { maken van functie kort met korte toon
  digitalWrite(LED_BUILTIN, HIGH);// zet de ingebouwde led aan
  delay(200);                     // pauze van 200 milliseconden
  digitalWrite(LED_BUILTIN, LOW); // zet de ingebouwde led uit
  delay(200);  }                  // pauze van 200 milliseconden
```

Om deze functies dan te gebruiken in je programma (*calling*) gebruiken we de functienaam gevolgd door haakjes openen en sluiten. Beëindig tenslotte de instructie die de functie aanroept met een puntkomma. Bijvoorbeeld “lang();”.

De ingebouwde led van de Arduino (LED_BUILTIN) is natuurlijk niet de enige led die we kunnen aansturen. We kunnen ook een externe led op een digitale pin van de Arduino plaatsen. Belangrijk bij het gebruik van een led in een circuit, is dat men een voorschakelweerstand in serie moet schakelen. Dit is nodig omdat een led zich gedraagt als een diode en stroom bijgevolg maar in één richting kan doorlaten. Een led is geen ideale diode omdat er in de doorlaatrichting een zekere spanning moet staan voordat ze stroom doorlaat en gaat branden. Die spanning wordt de drempelspanning genoemd. Als er te veel spanning over de led staat zal die kapotgaan. Deze drempelspanning (U_f) is de hoeveelheid spanning die verloren gaat in de led bij gebruik met een bepaalde referentiestroom. De drempelspanning hangt voornamelijk af van de kleur van de led. Hierbij een voorbeeld met een rode led met een drempelspanning van 1,8 V bij 20 mA (0,020 A). Wetende dat de Arduino pin een output heeft van 5 V, kan nu met de wet van Ohm de voorschakelweerstand berekend worden.

$$\boldsymbol{R = \frac{U - U_f}{I} = \frac{5V - 1,8V}{0,020A} = 160\Omega}$$

Waarbij:

I	stroom [A]
R	weerstand [Ω]
U	uitgangsspanning [V]
U_f	drempelspanning [V]

Als zo een weerstand van 160 Ω berekend wordt, is het aanbevolen om uit de beschikbare weerstanden een weerstand van 160 Ω of hoger te nemen. Weerstandwaardes worden weergegeven via een kleurcode, waarbij de eerste 2 ringen een getalwaarde vormen en de

volgende ringen een vermenigvuldigingsfactor. Soms is er ook een tolerantie ring aanwezig. Elke kleur heeft een specifieke waarde. Zwart (0), bruin (1), rood (2), oranje (3), geel (4), groen (5), blauw (6), violet (7), grijs (8) en wit (9). Om de kleurenvolgorde te onthouden kan men de volgende zin als ezelsbruggetje gebruiken. Zij Bracht Rozen Op Gerrits Graf Bij Vies Grauw Weer.

Met de functie “pinMode(pin, modus)” zouden we nu gewoon de pin waarde kunnen instellen van de pin die we gebruiken, bijvoorbeeld pin 10. We kunnen ook de led instellen in de variabelen met “int naam = pin;” bijvoorbeeld “int RodeLed = 10;”. De naam van de digitale pin 10 is nu RodeLed. Het voordeel van een variabele in dit geval is dat je het daadwerkelijke nummer van de pin maar één keer hoeft op te geven, maar je kunt het vaak gebruiken. Dus mocht je later besluiten om van pin 10 naar pin 12 te gaan, dan hoef je dit maar op één plek in de code te wijzigen. We geven de led ook een beschrijvende naam i.p.v. een getal om de betekenis van de variabele duidelijk te maken.

We kunnen nu een *if*-instructie gebruiken om alle letters van het alfabet in de code te programmeren en zo elke letter op te vragen. De *if*-instructie controleert op een voorwaarde en voert de volgende instructie of een reeks instructies uit als de voorwaarde “waar” is. We maken een functie morse met een parameter. De parameter is “char morse_letter” waarin “char” een gegevenstype is dat wordt gebruikt om een tekenwaarde op te slaan. Letterlijke tekens worden tussen enkele aanhalingstekens geschreven, zoals ‘A’. morse_letter is de naam van de parameter voorwaarde. De *if*-instructie controleert of de voorwaarde “waar” is, als dit zo is krijgen we de morsecode van A. Als we alle letters in de morse functie hebben geprogrammeerd kunnen we nu in de lus functie elke letter opvragen met de functie morse(‘letter’).

```
void morse (char morse_letter) {
  if (morse_letter == 'A')
    { kort(); lang(); } }

void loop() {
   morse ('A');
   end_of_word(); }
```

De voorwaarden in de *if*-instructie worden na elkaar getest. In bepaalde gevallen willen we alleen de instructies uitvoeren die overeenkomen met één situatie, alle volgende tests moeten niet worden uitgevoerd. Hier kunnen we gebruik maken van een *else*- instructie. Voor langere programma's is de *switch/case*-instructie soms gemakkelijker te gebruiken. Net als *if*-instructie bestuurt *switch case* de stroom van het programma door verschillende code te specificeren die onder verschillende omstandigheden moet worden uitgevoerd. De *switch*-instructie vergelijkt de waarde van een parameter met de waarden die zijn opgegeven in *case*-instructies. We doen dit met de functie "switch(parameter)". Wanneer een *case*-instructie wordt gevonden waarvan de waarde overeenkomt met die van de parameter dan wordt de code in die *case*-instructie uitgevoerd. Om een *switch/case*-instructie in gebruik te nemen, moet er een lijst met opties (*cases*) gemaakt worden. Als we dit toepassen op de alfabet code zou elke letter een case zijn.

```
switch (morse_letter) {
   case 'A':
     kort(); lang();
   break;
```

We kunnen vanuit de computergegevens naar de code sturen en vanuit de Arduino gegevens naar de computer verzenden via de seriële poort die ze verbindt. Hier gaan we gebruikmaken van de seriële monitor die geopend kan worden door op het vergrootglas in de rechterbovenhoek van de Arduino IDE te klikken. De seriële monitor is een essentieel hulpmiddel bij het maken van projecten met Arduino. Het kan rechtstreeks communiceren met het Arduino bord. We kunnen nu in plaats van in de lus functie de letter te typen deze laten lezen in de seriële monitor. We kunnen dit doen met de functie "Serial.read();", deze functie leest wat we in de monitor typen.

```
void loop() {
  char LETTER = Serial.read();
  morse(LETTER); }
```

We kunnen ook informatie afdrukken in de monitor. We kunnen bijvoorbeeld in elke letter functie een "serial.print()" functie toevoegen. Als we dan een functie opvragen krijgen we naast het knipperen ook een tekst op de seriële monitor. De tekst die we willen afdrukken moet tussen de haakjes staan met dubbele aanhalingstekens. Dat zou er bijvoorbeeld als volgt kunnen uitzien voor de letter A: "serial.print("alpha")". We kunnen ook "serial.println()" gebruiken. Dit drukt gegevens af naar de seriële poort als een teken voor een nieuwe regel. Zo komen gegevens onder elkaar te staan.

De volgende oefening bestaat er uit een led te dimmen aan de hand van *pulse-width modulation* (PWM) of pulsbreedtemodulatie. Dit is een modulatietechniek waarbij pulsen in een vaste frequentie worden uitgezonden maar waarvan de breedte gevarieerd wordt. De digitale pinnen van de Arduino kunnen geen spanning geven tussen 0 V en 5 V. De spanning wisselt sprongsgewijs tussen deze twee niveaus, ook wel blokgolfsignaal genoemd. Wat wel kan aangepast worden is de aan/uit snelheid van de led waardoor de helderheid van de led verandert. Met de functie "*analogWrite(pin, PWMwaarde")* kunnen we een pin een analoge waarde geven. Dit is een waarde tussen 0 en 255.

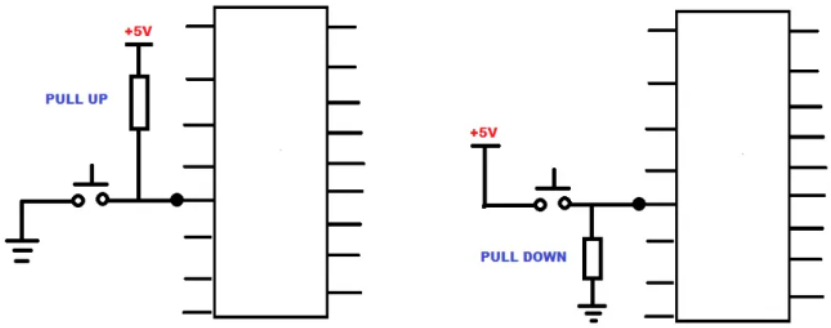
Om de led nu geleidelijk aan te laten gaan kunnen we de *for*-instructie gebruiken. Die wordt gebruikt om een blok met code tussen accolades te herhalen. We gaan hier een PWM-waarde maken die van 0 naar 255 gaat. We doen dit met de functie "(int i=0; i<255; i++)", i is hier de PWM-waarde. I start gelijk aan 0, als i kleiner is dan 255 dan verhogen we de waarde i met 1. We kunnen dit ook omgekeerd doen om de led geleidelijk te dimmen met "(int i=255; i>0; i--)".

5.2.2.2 Labo 2

In het tweede labo wordt gebruik gemaakt van een drukknop die op een digitale poort wordt aangesloten. In de set-up gaan we een opgegeven pin configureren om zich te gedragen als een ingang. We gebruiken hiervoor de functie “pinMode(pin, modus)”. De pin in deze functie is de digitale pin waar de drukknop op aangesloten is en de modus is ingang(INPUT). We kunnen de data uitlezen van de digitale pin met de functie “digitalRead(pin)".

Een drukknop heeft een weerstand nodig wanneer deze wordt gebruikt als ingang. De belangrijkste reden om het te gebruiken is ervoor te zorgen dat een gedefinieerde status (0 of 1) bij de ingang van de Arduino juist is. Bij het niet plaatsen van een weerstand krijgen we niet altijd de verwachte status. Door allerlei omstandigheden kan er ruis ontstaan. Omdat de digitale poort alleen een absolute 0 V of 5 V kan meten, kan door de ontstane ruis dus de drempelspanning overschreden worden. En wordt er naar 5 V gegaan zonder indrukken van de drukknop. De ruis kan dus onverwachte resultaten veroorzaken.

Er zijn twee typische configuraties bij het gebruik van een weerstand en drukknop is *pull-up* en *pull-down.* Een *pull-up* weerstand is een weerstand die de pin van de Arduino naar een hoog-status trekt, zodat deze normaal hoog (1) is, tenzij opzettelijk laag wordt gebracht. Dit circuit werkt doordat de weerstand direct is verbonden met de positieve spanning. Door de open schakelaar valt alle spanning van de voeding over de weerstand. Wanneer alleen de knop wordt ingedrukt, wordt de status 0 gepresenteerd aan de ingang van de Arduino. Bij de *pull-down* configuratie is de drukknop aangesloten op de voedingsspanning, terwijl de weerstand tussen de drukknop en de microcontroller met massa is verbonden. Wanneer de drukknop niet ingedrukt is, is de status aan de ingang van de microcontroller 0 omdat deze via de *pull-down*-weerstand met aarde is verbonden. Door op de knop te drukken, wordt de ingang met de voedingsspanning verbonden, waardoor de status van 0 naar 1 verandert (Oscarromero, 2017).



Afbeelding 16 Pull-up & Pull-down

Bron: Lancaster, 2021

De volgende oefening demonstreert het gebruik van *analogRead().* Het leest een waarde van een analoge pin. De Arduino heeft een analoog naar digitaal converter (ADC). Dit betekent dat het de ingangsspanningen tussen 0 V en de werkspanning 5 V in kaart zal brengen in gehele waarden. De ADC van de Arduino is een 10 bit ADC wat betekent dat het een digitale waarde geeft in het bereik van 0 - 1023 (2^{10}). Op een Arduino UNO bijvoorbeeld levert dit een resolutie op tussen metingen van 5 V/1024 eenheden en 0,0049 V/eenheid. Vervolgens wordt een spanningsdeler gemaakt met een schakeling van 2 weerstanden. Als de analoge waarde uitgelezen wordt kan zo berekend worden wat de weerstand op het circuit is.

5.2.2.3 Labo 3

Het derde en laatste labo begint met een oefening om een led dimmer te maken. Daarbij zal gebruik gemaakt worden van een potentiometer, een eenvoudige draaiknop die een variabele weerstand biedt. Deze wordt aangesloten op een analoge pin die uitgelezen zal worden met *analogRead().* Er wordt ook een led met een voorschakelweerstand aangesloten. Daar wordt een signaal naar gestuurd met *analogWrite().* De Arduino heeft een *analogRead*-bereik van 0 tot 1023 (10 bits) en een *analogWrite*-bereik van 0 tot 255. Daarom moeten de gegevens van de potentiometer worden geconverteerd om in het kleinere bereik te passen voordat deze gebruikt wordt om de led te dimmen. Om deze waarde te converteren gebruikt men een functie met de naam *map().* Dit zou er als volgt uitzien: “map(waarde, 0, 1023, 0, 255); “. De nieuwe waarde wordt vervolgens uitgevoerd naar de led, waardoor deze gedimd of helderder kan gemaakt worden door aan de potentiometer te draaien.
Deze oefening kan nog worden uitgebreid met het toevoegen van een analoge sensor. Naast de led wordt een *Light Dependent Resistor* (LDR) of lichtgevoelige weerstand geschakeld. Een LDR is een sensor waarvan de weerstand wordt beïnvloed door de hoeveelheid licht die erop schijnt. De weerstandswaarde wordt kleiner als de hoeveelheid licht sterker wordt. De waarde wordt afgedrukt in de seriële uitgang. Als de potentiometer nu aangepast wordt wijzigt de helderheid van de led en zo ook de waarde van de LDR.

Bij de volgende oefening zal nu een andere sensor gebruikt worden, namelijk een *Temperature Dependent Resist*or (TDR) of temperatuurgevoelige weerstand. Deze sensor wordt geschakeld op een analoge pin. Daarna moet gekeken worden naar het type sensor om uit te zoeken wat de analoge waarde geeft voor welke temperatuur. Dit vraagt een paar bewerkingen. Vooreerst dient de analoge waarde omgezet te worden naar een spanning. Hiervan trekt men een *offset* spanning af om negatieve temperaturen te meten. Als dit gedeeld wordt door de hoeveelheid spanning per graad Celsius wordt een temperatuur bekomen. De *offset* spanning is de hoeveelheid spanningsverschil tussen de minimum en nul temperatuur van de sensor. De *offset* spanning en de hoeveelheid spanning per Celsius moet men gaan opzoeken want dit verschilt van type tot type.

De laatste oefening is een toepassing op het gebruik van een bibliotheek. Temperatuur en vochtigheid worden gemeten. In plaats van hierbij analoge sensoren te gebruiken om dan via een bewerking de resultaten te verkrijgen, zal nu gebruik gemaakt worden van een digitale sensor. Er moet vervolgens in *manage libraries* gezocht worden naar de bibliotheek die bij deze sensor past. Is het correcte type sensor gevonden dan kan deze bibliotheek geïnstalleerd worden. Als de sensor en de bibliotheek nu nog gedefinieerd worden, kunnen we met een simpele code de vochtigheid en temperatuur verkrijgen (*Arduino Project Hub*, 2022; Bueken et al., 2020).

5.2.3 Examen

Het examen dat peilt naar de opgedane kennis en vaardigheden is van het type open boek examen. Dit betekent dat er gebruik gemaakt mag worden van cursusmateriaal op Blackboard en notities. Hulp van buitenaf inroepen is vanzelfsprekend niet toegelaten. Het examen bestaat uit het berekenen van een voorschakelweerstand van een led. Afhankelijk van de instructies moeten dan bepaalde componenten geschakeld worden, zoals leds, een potentiometer, sensoren, een drukknop. Ten slotte worden instructies gegeven om de code te schrijven. Daarbij moet de structuur van het programma nauwgezet gevolgd worden om de correcte uitkomst als resultaat te bekomen. In een mondelinge verdediging bij de docent zal de student uitleggen hoe hij/zij tewerk gegaan is.

6 Arduino in een maritieme context

In dit hoofdstuk gaan we drie maritiem geïnspireerde oefeningen maken, die als leerstof gebruikt zouden kunnen worden in het seminarie automatisatie. Aanleiding voor deze oefeningen zijn maritieme publicaties over reële cases, die in de vakpers verschenen. Deze artikelen zijn te vinden in de bijlages. De oefeningen onderscheiden zich door het gebruik van verschillende sensoren, respectievelijk een gassensor, een gyrosensor en een ultrasonische sensor. Elke oefening start met de uitleg en werking van desbetreffende sensor. Vervolgens worden stappen toegevoegd en werken we naar een complexer eindresultaat toe.

6.1 Gassensor (MQ-5)

We geven het volgende Arduino-project een neus voor gassen met de MQ-5 gassensor module. Tegenwoordig is aardgas overal. Het is gemakkelijk te vervoeren en levert energie voor verschillende taken in zowel huishoudelijk als industrieel gebruik. Het bevat ook een aanzienlijk brandgevaar als het niet zorgvuldig wordt behandeld. Dit is waar de MQ-5 sensor in het spel komt. De MQ-5 sensor kan H_2, CH_4, CO, alcohol en LPG detecteren (een vorm van LPG is ook butaan, het gas dat in een aansteker zit). In dit hoofdstuk gaan we de MQ-5-sensor bespreken en die aan een Arduino UNO koppelen om te zien wat er gebeurt als gassen in contact komen met deze sensor.

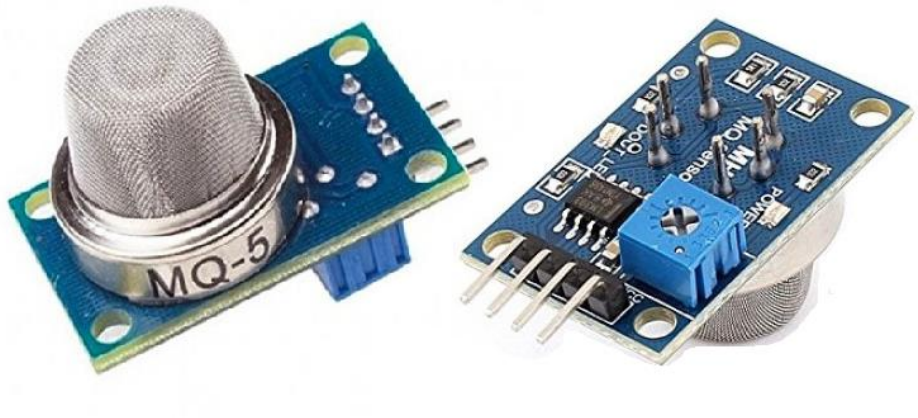
Afbeelding 17 MQ5 sensor
Bron: *ElectronicsComp*, 2022

6.1.1 Algemene informatie over de gassensor (MQ-5)

De MQ-5 gassensor is een gassensor die metaaloxidemateriaal gebruikt om de aanwezigheid van bepaalde gassen in de lucht te detecteren. Dit soort sensor meet de elektrische weerstand van het metaaloxidemateriaal, die in bepaalde gassen verandert als gevolg van een chemische reactie aan het oppervlak van het materiaal.

De MQ-5 gassensor heeft een verwarmingselement nodig om effectief brandbaar gas te detecteren. Het plaatsen van een verwarmingselement in de buurt van brandbaar gas kan schadelijk zijn. Daarom heeft de sensor een explosieveilige filterbehuizing, waarin het

verwarmingselement zich bevindt (zie Afbeelding 17 MQ5 sensor. De behuizing beschermt ook tegen stof en andere zwevende deeltjes.

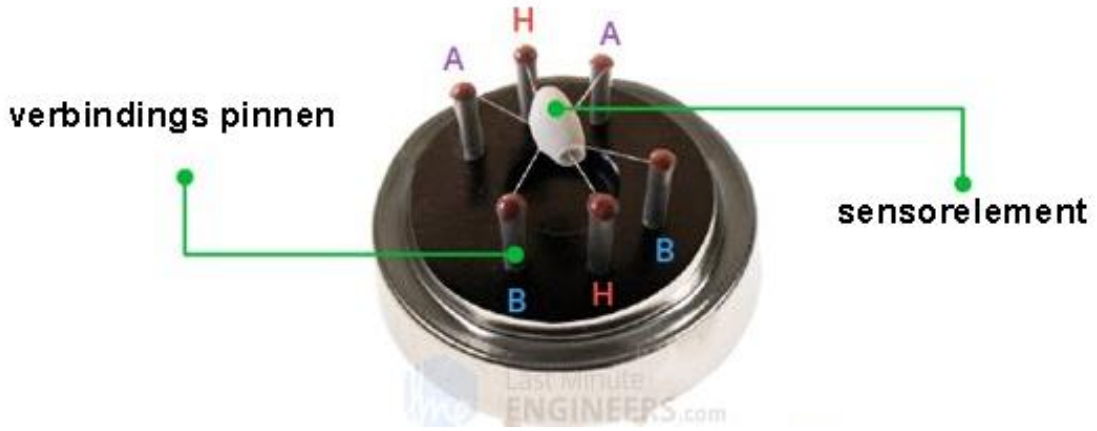


Afbeelding 18 MQ-5 module

Bron: invento, 2023

In dit project gaan we werken met een MQ-5 gassensor module. Dit is een printplaat waarop de sensor gesoldeerd is. Op de achterzijde van de module hebben we 2 ingebouwde leds. De power-led gaat branden als het bord aan staat. En de andere (DOUT LED) gaat branden als er een signaal is.

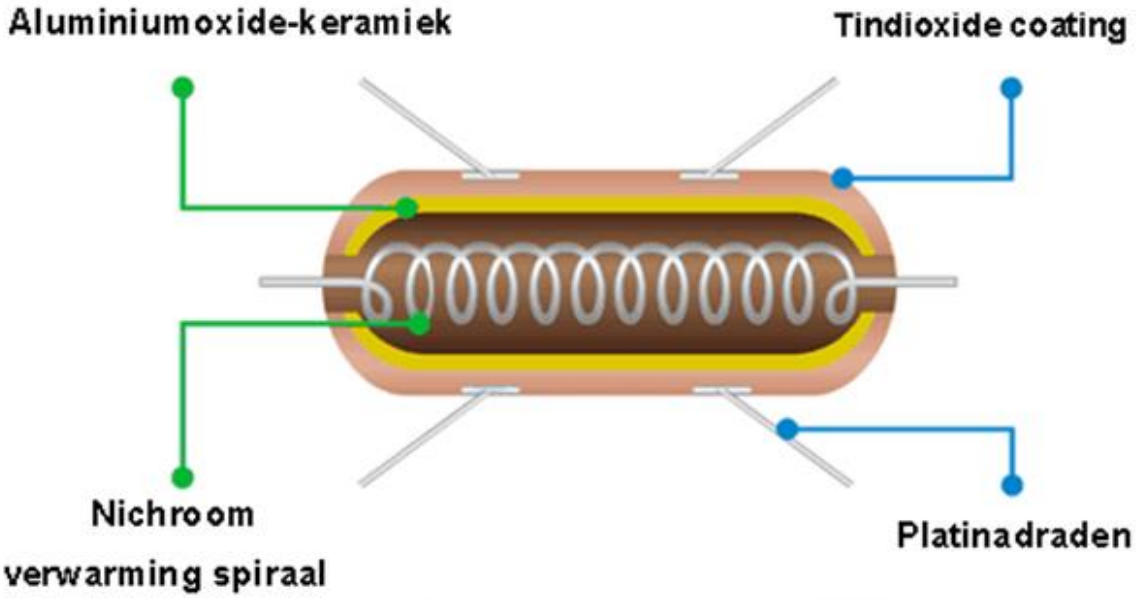
Als we de behuizing verwijderen die uit twee dunne lagen roestvrijstalen gaas bestaat, kunnen we de elektrochemische sensor waarnemen (zie Afbeelding 19 Binnenkant MQ5 sensor). De behuizing is via een verkoperde klemring aan de rest van het lichaam gebonden. De stervormige structuur die we in de behuizing waarnemen wordt gevormd door het sensorelement en zes verbindingspinnen die op een bakelieten basis bevestigd zijn. Bakeliet is bijzonder geschikt vanwege zijn buitengewoon hoge weerstand tegen elektriciteit, hitte en chemische werking. Van de zes pinnen zijn twee draden (H) verantwoordelijk voor het verwarmen van het sensorelement en zijn verbonden via een nichroom (nikkel-chroom) spiraal. De overige vier pinnen (A & B) zijn verantwoordelijk voor de uitgangssignalen, en zijn verbonden met platinadraden. Deze draden zijn verbonden met het lichaam van het sensorelement en brengen kleine veranderingen over in de stroom die door het sensorelement gaat (Harshil, 2022; Infinity learn, 2022).



Afbeelding 19 Binnenkant MQ5 sensor

Bron: *Lastminuteengineers*, 2018

In het midden van de stervorm is het sensorelement (zie Afbeelding 20 Sensorelement MQ5. Het heeft een coating van tindioxide (SnO_2) en de binnenkant is gemaakt van aluminiumoxide-keramiek (Al_2O_3).Het tindioxide is gevoelig voor brandbare gassen. De keramische laag verhoogt echter alleen het verwarmingsrendement en zorgt ervoor dat het sensorgebied constant wordt verwarmd tot de werktemperatuur. De nichroom spiraal loopt door het aluminiumoxide-keramiek en vormt het verwarmingssysteem. De platinadraden die verbonden zijn met de coating van tindioxide vormen het detectiesysteem.



Afbeelding 20 Sensorelement MQ5

Bron: *Lastminuteengineers*, 2018

Wanneer we tindioxide bij hoge temperatuur in schone lucht verwarmen, wordt zuurstof aan het oppervlak geabsorbeerd. Dit voorkomt dat elektrische stroom vloeit. Wanneer er reducerende gassen aanwezig zijn neemt de dichtheid van zuurstof af op de oppervlakte naarmate het reageert met de reducerende gassen. Vervolgens komen elektronen vrij in het tindioxide, waardoor de stroom vrij door de sensor kan stromen.

Deze gassensor vereist een opwarm- of stabilisatiefase voordat het instrument goed kan werken. In de technische data van deze sensor is te lezen dat de opwarmtijd of preheat time 24 uur bedraagt. Dit betekent dat men de sensor 24 uur constant zou moeten laten draaien, wil men de standaard prestatiewaarden verkrijgen zoals in de technische data vermeld. Gezien het kleine formaat van de sensor, zal het thermisch evenwicht vrijwel zeker binnen 30 minuten worden bereikt, maar niet perfect (*Lastminuteengineers*, 2018).

De sensor kan gas detecteren en kan ook de hoeveelheid gasconcentraties meten. Het detecteren van gas gebeurt als volgt. Als een hoeveelheid gas toeneemt zal er een stijging zijn in de uitgangsspanning. Naarmate de drempelspanning is bereikt zal de groene led (DOUT LED) van de module branden. Die drempelspanning bepaalt de gevoeligheid van de sensor, en deze kan aangepast worden met de potentiometer op de module.

Laten we nu eens kijken naar de pinnen en de hardware. De MQ-5 sensor is bevestigd op een printplaat en heeft vier pinnen. Een voedingspin, een aardingspin en twee pinnen die gelijktijdig analoge en digitale gegevens kunnen produceren. De sensor kan via de digitale pin de aanwezigheid van brandbare gassen detecteren. Met de analoge pin kan men de concentratie bepalen in lucht. Omdat de werkspanning van de module 5 V is met een nauwkeurigheid van 0,1 procent, gebruiken we altijd de 5 V pin van de Arduino om het circuit van stroom te voorzien (Osoyoo, 2018).

6.1.2 Gassensor met Arduino

Met het maritiem artikel over de Deepwater Horizon als uitgangspunt gaan we nu onze kennis van een gassensor gebruiken in een reële toepassing. In een Arduino project gaan we de technische omstandigheden namaken zoals ze zich voordeden op de Deepwater Horizon.

In Bijlage 1: Artikel, het artikel over de Deepwater Horizon's safety systems, lezen we dat het gas alarm met een *bypass* systeem op stil was gezet. *Bypass* is de term die in de alarmwereld wordt gebruikt voor het opzettelijk deactiveren van een bepaalde sensor of zone in een beveiligingssysteem. *Bypass* zones worden onvolledig of niet bewaakt door het beveiligingssysteem en zullen bijgevolg geen alarm veroorzaken als de zone wordt geschonden of verstoord Dit gebeurde omdat er zich af en toe een vals alarm voordeed dat de slaap van de crew verstoorde (Almeida & Konrad, 2011; Charleston Security Systems, 2015; Pilkington, 2010).

Tabel 2 Componenten Oef MQ-5

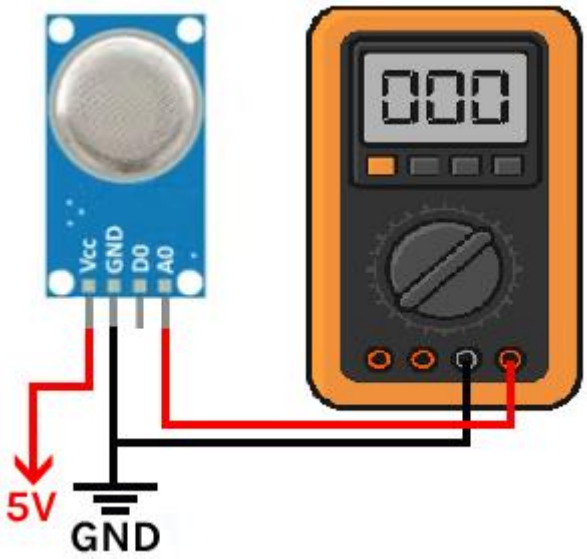
Bron : Eigen werk

Componentenlijst
Gas sensor MQ-5 module
Rode led
Groen led
Blauwe led
Drukknop 6x6x4
Piezo buzzer
Potentiometer

6.1.2.1 Gasdetector analoog

Nu we begrijpen hoe de MQ-5 gassensor werkt gaan we hem implementeren in de oefening. We weten dat de werking van de gassensor gebaseerd is op de eigenschap van het tindioxide materiaal dat bij verwarming zuurstof uit de lucht gaat absorberen. Dit voorkomt dat elektrische stroom voortvloeit. Wanneer er gassen aanwezig zijn neemt de dichtheid van geabsorbeerde zuurstof af. Vervolgens komen elektronen vrij in het tindioxide, waardoor de stroom vrij door de sensor kan stromen. Dit principe gaan we nu uitmeten met een multimeter.

We gaan de sensor schakelen op de Arduino, maar alleen VCC en aarding verbinden. We schakelen de multimeter met de COM wat staat voor *common* op de aarding van het circuit en de poort waarmee stroom, spanning en weerstand worden gemeten schakelen we op de analoge poort van de sensor.



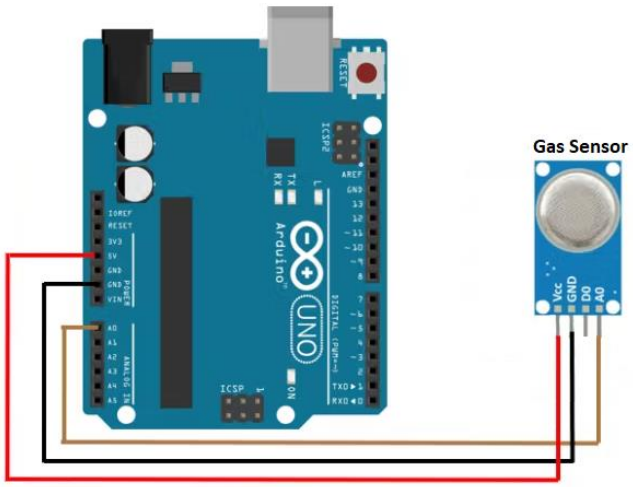
Afbeelding 21 MQ-sensor met multimeter

Bron: *Lastminuteengineers*, 2018

Als de schakeling compleet is en we laten de sensor een tijdje opwarmen, lezen we een laag voltage af van de multimeter. Als we nu met een aansteker gas bij de sensor loslaten, neemt de gasconcentratie in de omgeving toe. We zien dat spanningswaarde stijgt.

6.1.2.2 Gasdetector digitaal uitlezen

We weten uit hoofdstuk 3.1 dat de Arduino een ADC heeft. Hiermee kunnen we het analoog signaal omzetten naar een digitaal. We werken verder op ons circuit en schakelen de analoge pin op één van de zes analoge poorten van de Arduino.



Afbeelding 22 MQ-sensor met Arduino

Bron: Eigen werk m.b.v. *Tinkercad*, 2022

We verklaren twee variabelen, de eerste is de invoerpin met naam sensor en analoge pin nul. De tweede is de gaswaarde.

```
int sensor = A0;          // selecteer de ingangspin voor de sensor
int gas_value;            // variabele om de waarde op te slaan die van de
                          sensor komt
```

In de set-up gaan we een opgegeven pin configureren om zich te gedragen als een ingang. Met de functie “pinMode(pin, modus)”.

```
void setup(){
pinMode(sensor,INPUT);          // verklaar de sensor als een INPUT
Serial.begin(9600);
}
```

We beginnen de lus functie met het uitlezen van de analoge waarde van de sensor, we linken deze waarde aan de naam *“gas_value”*. Deze waarde drukken we af op de seriële monitor.

```
void loop(){
gas_value = analogRead(sensor);         // lees de waarde van de sensor
Serial.println(gas_value);              // print de waarde van de sensor
}
```

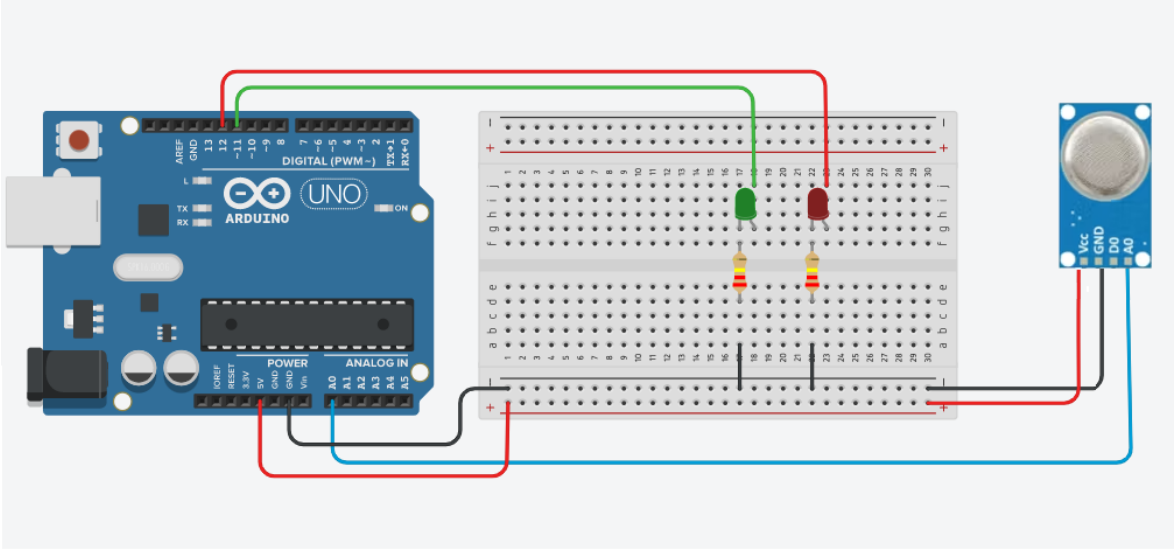
Als we het programma uploaden kunnen we de digitale waarde aflezen in de seriële monitor. Als we met de aansteker gas lossen bij de gassensor zien we dat de waarde stijgt (Rucksikaa, 2021).

Afbeelding 23 MQ-sensor seriële monitor

Bron: Eigen werk m.b.v. Arduino IDE

6.1.2.3 Gasdetector met led

We werken verder op het circuit, maar gaan nu twee leds toevoegen en een opwarmperiode voor de sensor. Als de sensor een waarde meet over een bepaalde drempelwaarde dan moet de rode led gaan branden. Als de drempelwaarde niet wordt overschreden dan blijft de groene led aan. We schakelen een voorschakelweerstand op beide leds (zoals in 5.2.2.1).



Afbeelding 24 Schakeling MQ sensor met leds

Bron: Eigen werk m.b.v. *Tinkercad*, 2022

We beginnen met het verklaren van de twee leds en we verklaren een drempelwaarde met naam “sensorVal”.

```
int sensor = A0;          // selecteer de ingangspin voor de sensor
int gas_value;            // variabele om de waarde op te slaan die van de
                          sensor komt
int redLed = 12;          // selecteer de pin voor de rode LED
int greenLed = 11;        // selecteer de pin voor de groene LED
int sensorVal = 400;      // variabele van de drempelwaarde
```

In de set-up configureren we de twee leds als uitgang en gaan een opwarmtijd instellen van twintig seconden (20000 ms). Deze opwarmperiode zal in de seriële monitor verduidelijkt worden.

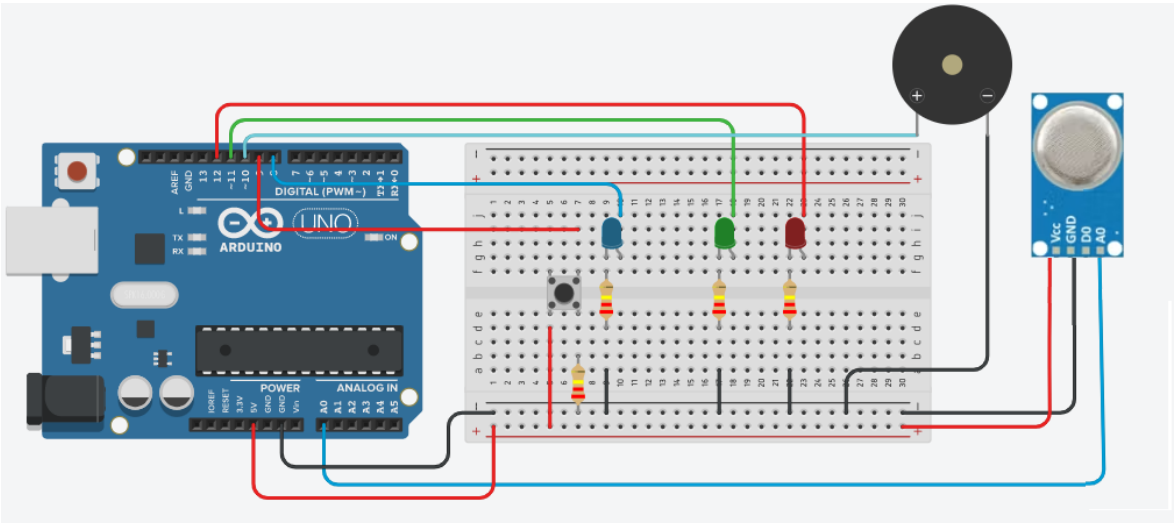
```
void setup(){
  pinMode(sensor,INPUT);          // verklaar de sensor als een INPUT
  pinMode(redLed, OUTPUT);        // verklaar de rode led als een OUTPUT
  pinMode(greenLed, OUTPUT);      // verklaar de groen led als een OUTPUT
  Serial.begin(9600);
  Serial.println("MQ5 Heating Up!");
  delay(20000);                   // Opwarmtijd 20 seconds
  Serial.println("Ready");
}
```

We gebruiken een *if-else*-instructie om te bepalen welke instructies er moeten uitgevoerd worden. De *if*-instructie controleert op een specifieke voorwaarde en voert de volgende instructie of reeks instructies uit als de voorwaarde correct is. Dit doen we met de functie "*if* (voorwaarde) {instructie(s)}". We willen dat de rode led gaat branden wanneer de drempelwaarde van de sensor overschreden is. Als de gaswaarde onder de drempelwaarde is, brandt de groene led. Hierbij is drempelwaarde de voorwaarde en de instructie is de toestand van de rode led en groene led. De toestand veranderen we met de functie "*digitalWrite(pin, value)"*, waar de pin de digitale pin is of de naam van een digitale pin. *Value* is de waarde die de pin moet hebben, dit kan hoog of laag zijn (Bichon, 2021).

```
void loop(){
  gas_value=analogRead(sensor);         // lees de waarde van de sensor
  Serial.print("gas value = ");
  Serial.println(gas_value);            // print de waarde van de sensor
  if (gas_value > sensorVal)            // controle van de drempelwaarde
  {
    digitalWrite(redLed, HIGH);         // zet de rode led aan
    digitalWrite(greenLed, LOW);        // zet de groene led uit
  }
  else
  {
    digitalWrite(redLed, LOW);          // zet de rode led uit
    digitalWrite(greenLed, HIGH);       // zet de groene led aan
  }
   delay(100);
}
```

6.1.2.4 Gasdetector met bypass systeem met buzzer

We werken terug verder op ons circuit. We willen naast de leds een buzzer die afgaat wanneer we de drempelwaarde overschrijden. We gaan ook een bypass systeem toevoegen die de buzzer kan afzetten. Dit bypass systeem is actief als de knop ingedrukt is. Als deze knop ingedrukt is moet er een controle led aan gaan.



Afbeelding 25 Schakeling MQ sensor met buzzer

Bron: Eigen werk m.b.v. *Tinkercad*, 2022

We beginnen met het verklaren van de onderdelen die we toevoegen aan onze schakeling. Namelijk de buzzer op pin 10, de drukknop op pin 9 en de blauwe led op pin 8. We verklaren ook de toestand waar de knop in is met de naam *buttonState.*

```
int buzzer = 10;        // selecteer de pin voor de buzzer
int button = 9;         // selecteer de pin voor de knop
int blueLed = 8;        // selecteer de pin voor de blauwe led
int buttonState;        // variabele om de waarde op te slaan die van de
                        knop komt
```

In de set-up configureren we de knop als een ingang en de blauwe led als een uitgang.

```
pinMode(button, INPUT);         // verklaar de drukknop als een INPUT
pinMode(blueLed, OUTPUT);       // verklaar de blauwe led als een OUTPUT
pinMode(buzzer, OUTPUT);        // verklaar de buzzer als een OUTPUT
```

Het eerste wat we toevoegen aan de lus functie is het uitlezen van de digitale waarde van de drukknop, we linken deze waarde aan de naam *buttonState.* De *if* functie heeft hier nu twee voorwaarden. De eerste voorwaarde is of de gaswaarde wordt overschreven, de tweede voorwaarde is de toestand van de knop. In de *if* functie kunnen we meerdere voorwaarden toevoegen door het gebruik van “&&”. We kunnen dit weergeven in een

waarheidstabel, waarin we de twee voorwaarden hebben die waar of fout kunnen zijn. We hebben ook onze drie leds en de buzzer die waar (aan) of fout (uit) kunnen zijn.

Tabel 3 waarheidstabel mq5

Bron: Eigen werk

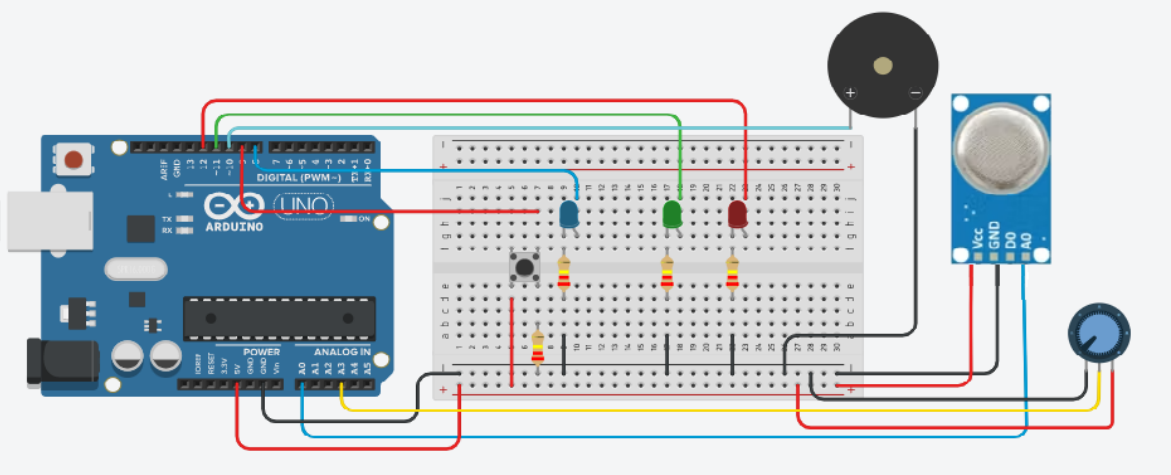
Gas val > sensor_val	*buttonState*	redLed	greenLed	blueLed	tone
W	F	W	F	F	F
W	W	W	F	W	W
F	F	F	W	F	F
F	W	F	W	W	F

We krijgen hier nu vier mogelijke combinaties en gaan hiermee vier if functies schrijven, telkens met hun specifieke instructies. Voor de functie om de buzzer af te laten gaan gebruiken we “tone (pin, frequentie, duur)”. Pin is de digitale pin of de naam van een digitale pin, frequentie is de frequentie van de toon in hertz en duur is de lengte van de toon in milliseconden (*Arduino Documentation*, 2015; James, 2015).

```
void loop(){
  gas_value=analogRead(sensor);         // lees de waarde van de sensor
  Serial.print("gas value = ");
  Serial.println(gas_value);            // print de waarde van de sensor
  buttonState=digitalRead(button);
 if (gas_value > sensorVal && buttonState == 0 ) // controle van de
drempelwaarde en drukknop
 {
      digitalWrite(redLed, HIGH);       // zet de rode led aan
      digitalWrite(greenLed, LOW);      // zet de groen led uit
      digitalWrite(blueLed, LOW);       // zet de blauwe led uit
      tone( buzzer , 2000 , 50 );       // speel tone op buzzer af
 }
  else if (gas_value > sensorVal && buttonState == 1 )
 {
     digitalWrite(redLed, HIGH);
     digitalWrite(greenLed, LOW);
     digitalWrite(blueLed, HIGH);
 }
  else if (gas_value <= sensorVal && buttonState == 0 )
 {
    digitalWrite(redLed, LOW);
    digitalWrite(greenLed, HIGH);
    digitalWrite(blueLed, LOW);
 }
 else if (gas_value <= sensorVal && buttonState == 1 )
 {
    digitalWrite(redLed, LOW);
    digitalWrite(greenLed, HIGH);
    digitalWrite(blueLed, HIGH);
 }
   delay(100); }
```

6.1.2.5 Gasdetector met schakelknop en PWM

In de laatste oefening gaan we een potentiometer toevoegen om de frequentie van de buzzer aan te passen. In de voorgaande oefening moesten we de drukknop ingedrukt houden om de buzzer te dempen, in deze oefening gaan we de drukknop zo programmeren dat deze werkt als een schakelknop. Het enige wat we aan de schakeling veranderen is de potentiometer die we toevoegen. We sluiten de analoge pin van de potentiometer aan op de analoge pin A3.



Afbeelding 26 Schakeling MQ sensor met potentiometer

Bron: Eigen werk m.b.v. *Tinkercad*, 2022

We beginnen terug met het verklaren van de onderdelen die we toevoegen aan onze schakeling. In dit geval is het alleen de potentiometer met naam pot of pin A3. We creëren een variabele integer genaamd “blueLedState” die eerder is ingesteld met waarde *LOW.* We verklaren ook nog de frequentie waarde van de buzzer met naam “buzzValue” en de vorige toestand van de knop met naam “lastButtonState”.

```
int pot = A3;           // selecteer de ingangspin voor de potentiometer
int blueLedState = LOW ; // variabele om de waarde op te slaan van de
                            blauwe led
int buzzValue;  // variabele om de frequentie op te slaan van de buzzer
int lastButtonState; // variabele om de vorige toestand van de knop op te
slaan
```

We starten de lus functie met het uitlezen van de sensorwaarde en de waarde van de potentiometer. Daarna zeggen we dat de “lastButtonState” gelijk is aan de “buttonState”. Hiermee onthouden we de toestand van knop van de voorafgaande lus. We gaan

vervolgens de nieuwe toestand van de knop uitlezen. We printen dan zoals voordien de gaswaarde af in de seriële monitor.

In de vorige oefening hadden we twee voorwaarden, de eerste was de gaswaarde en de tweede was de toestand van de knop. We gaan in deze oefening de tweede voorwaarde vervangen door de toestand van de blauwe led met naam “blueLedState”.

We voegen nu een nieuwe *if* functie toe die kijkt of drukknop is ingedrukt en losgelaten. Als dit zo is gaan we de we de waarde van blauwe led inverteren. We doen dit met “waarde = !waard”.

```
void loop(){
  gas_value=analogRead(sensor); // lees de waarde van de sensor
  buzzValue = analogRead(pot); // lees de waarde van de potentiometer
  lastButtonState = buttonState;    // de vorige knop toestand opslaan
  buttonState = digitalRead(button);    // lees de waarde van de knop
  Serial.print("gas value = ");
  Serial.println(gas_value);   // print de waarde van de sensor
  if(lastButtonState == HIGH && buttonState == LOW) {  // controle of de
knop toestand hoog is geweest
  blueLedState = !blueLedState;} // waarde van de blauwe led om keren
```

Als we de eerste *if* functie van onze vier mogelijkheden nemen als voorbeeld, moeten we alleen de naam van de “buttonState” naar “blueLedState” vervangen. We gaan hier ook in de *tone* functie de frequentie aanpassen met de potentiometer. We doen dit door de frequentie te vervangen met een map functie “map(waarde, van laag, van hoog, naar laag, naar hoog)”. In deze functie is de waarde de waarde van de potentiometer met naam “buzzValue”, deze waarde gaat van 0 tot 1023. We weten dit omdat de analoge pin een 10 bits ADC heeft. We gaan deze waarde omzetten naar hertz met een minimale waarde van bijvoorbeeld 1000hz en een maximale van 4000 (*Arduino Project Hub*, 2022; *Arduino Reference*, 2022; ArduinoGetStarted.com, 2022).

```
 if (gas_value > sensorVal && blueLedState == 0 ) // controle van de
drempelwaarde en blauweled
 {
      digitalWrite(redLed, HIGH);      // zet de rode led aan
      digitalWrite(greenLed, LOW);     // zet de groen led uit
      digitalWrite(blueLed, LOW);      // zet de groen led uit
      tone( buzzer , (map(buzzValue, 0, 1023, 1000, 4000)) , 50 );
        // speel tone op buzzer af met potentiometer frequentie
 }
```

6.2 Gyrosensor (GY-521)

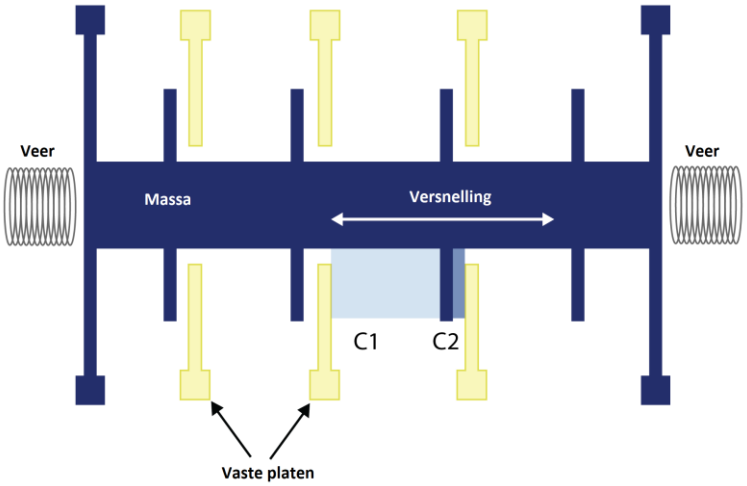
In dit project leren we hoe je de GY-521 module met Arduino kunt gebruiken. We zullen eerst deze module onder de loep nemen en kijken hoe deze werkt.

6.2.1 Algemene informatie over de GY-521 module

De GY-521 module is een printplaat voor de MPU-6050 MEMS (Micro-Elektromechanisch Systeem). De module bevat een drie-assige MEMS versnellingsmeter met een versnelling bereik van ±2g tot ±16g en een drie-assige MEMS gyroscoop met een hoeksnelheid bereik van ±250°/sec tot ±2000°/sec.

MEMS versnellingsmeter

MEMS-versnellingsmeters worden gebruikt om lineaire beweging te meten, zonder een vaste referentie. Het basisprincipe van een versnellingsmeter is dat een bewegend object een traagheid heeft, wat betekent dat het weerstand biedt tegen veranderingen in bewegingstoestand. Dit principe wordt gebruikt om de versnelling van een object te meten. De versnellingsmeters meet de verandering van de capaciteit tussen vaste platen en platen die aan de massa zijn bevestigd. Deze capaciteitsverandering wordt gemeten met een ADC en vervolgens wordt de versnelling berekend op basis van de snelheid waarmee de capaciteit verandert. In MPU-6050 wordt dit vervolgens omgezet in leesbare waardes en vervolgens overgedragen naar het I2C-masterapparaat (*Last Minute Engineers*, 2020; Thinkology, 2022).

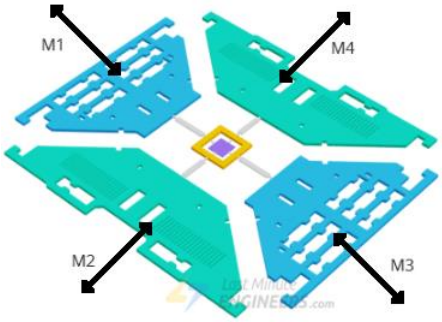


Afbeelding 27 MEMS versnellingsmeter

Bron: leveldevelopments, 2022

MEMS gyroscoop

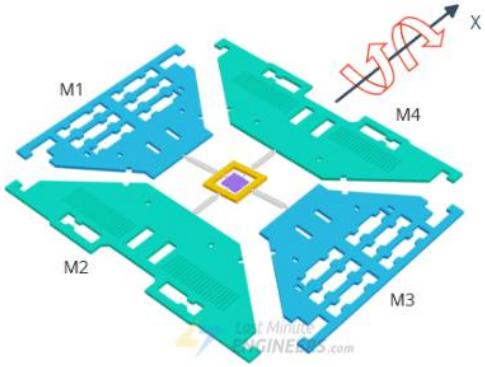
De MEMS- gyroscoop bestaat uit een massa (bestaande uit vier delen M1, M2, M3 en M4) die in een continue oscillerende beweging wordt gehouden zodat deze kan reageren op het Corioliseffect. Ze bewegen naar binnen en naar buiten in het horizontale vlak.



Afbeelding 28 MEMS gyroscoop oscillatie

Bron: *LastMinuteEngineers*, 2020

Wanneer we de structuur draaien, zorgt de corioliskracht die op de bewegende massa inwerkt ervoor dat de trilling verandert van horizontaal naar verticaal. Er zijn drie modi, afhankelijk van de as waarlangs de hoekrotatie wordt toegepast, namelijk rollen, gieren en stampen. We nemen de rol beweging als voorbeeld. Wanneer een draai beweging langs de x-as wordt toegepast, zullen M1 en M3 op en neer uit het vlak bewegen als gevolg van het Corioliseffect. Dit veroorzaakt een verandering in de rolhoek.



Afbeelding 29 MEMS gyroscoop rollen

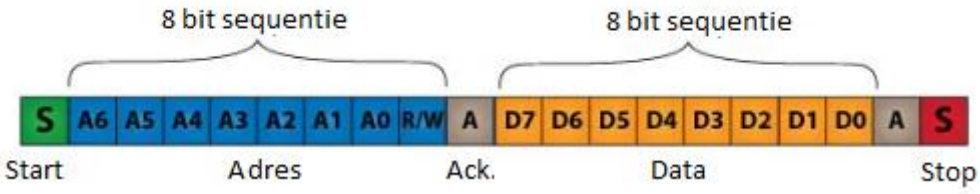
Bron : *LastMinuteEngineers*, 2020

Deze beweging wordt opgevangen door de condensator. Het geproduceerde signaal wordt vervolgens versterkt, gedemoduleerd en gefilterd om een spanning te produceren die evenredig is met de hoeksnelheid. Deze spanning wordt vervolgens gedigitaliseerd met behulp van een ADC. Het communicatieprotocol van deze module is I2C en kunnen deze aansluiten op verschillende microprocessoren, waaronder de Arduino met behulp van SCL- en SDA-pinnen (Agarwal, 2019; *LastMinuteEngineers*, 2020).

6.2.1.1 I2C

De I²C of I2C-communicatiebus is erg populair en wordt veel gebruikt door veel elektronische apparaten. Het kan gemakkelijk worden geïmplementeerd in elektronische ontwerpen die communicatie vereisen tussen een master bijvoorbeeld een Arduino en een *slave*-apparaat (3) bijvoorbeeld een sensor. De opbouw is eenvoudig omdat er slechts twee draden nodig zijn voor communicatie tussen maximaal 128 apparaten bij gebruik van 7 bits adressering en tot 1024 apparaten bij gebruik van 10 bits adressering.

Het is mogelijk om te communiceren tussen zoveel apparaten met slechts 2 draden, doordat elk apparaat een vooraf ingestelde ID of een uniek apparaat adres heeft. Hierdoor kan de master kiezen met welke apparaten het zal communiceren. De twee draden of lijnen worden *Serial Clock* (SCL) en *Serial Data* (SDA) genoemd. De SDA-lijn is de lijn die de gegevens draagt. De tweede lijn is de SCL-lijn, over deze lijn verzenden we het kloksignaal. Het kloksignaal is een continu signaal dat wordt gebruikt om de timing van gegevensoverdracht tussen de *master*- en *slave*-apparaten te synchroniseren (Dejan, 2015).



Afbeelding 30 I2C SDA-lijn

Bron: Dejan, 2015

Gegevens worden opgedeeld in twee soorten frames. Een adresframe, waarbij de controller aangeeft naar welk apparaat het bericht wordt verzonden. Daarop volgt één of meerdere dataframes, dit zijn gegevensberichten die worden doorgegeven van *master* naar *slave*-apparaat of vice versa (Sparkfun, 2018).

3 *Master/slave* is een model voor een communicatieprotocol waarin een of meer apparaten of processen (bekend als de master) totale controle heeft over een of meer andere apparaten of processen (bekend als *slaves*).

6.2.2 Gyrosensor met Arduino

Voor het Arduino project met de gyrosensor GY-521 module baseren we ons op het artikel over het lot van F/V Sage Catherine Lane en onze kennis van de bewegingssensor. Ons project zal bestaan uit het namaken van het technische aspect van het voorval.

In Bijlage 2 : Artikel , het artikel over het wedervaren van F/V Sage Catherine Lane, lezen we dat de kapitein zijn stuurhut verlaten heeft en het schip op automatische piloot vaart. Na een korte periode voelt de kapitein het schip abrupt naar bakboord draaien. Hij keert terug naar de stuurhut en probeert weg te draaien van de steiger, maar de Sage Catherine Lane raakt de steiger en loopt aan de grond. Drie dagen later zinkt het schip.

Uit het onderzoek bleek dat de kapitein twee dagen voor het incident niet in staat was de automatische piloot uit te schakelen en controle over het roer te krijgen. De kapitein onderzocht de automatische piloot en constateerde problemen met de roerstand aanwijzer en de roerstand sensor bij de roerpaal en ondernam actie om de problemen op te lossen. De NTSB (*National Transportation Safety Board*) zei dat hoewel de reparatie aanvankelijk werkte, de scherpe bocht van het schip naar bakboord aangaf dat het systeem faalde en de reparatie niet effectief was (Schuler, 2022).

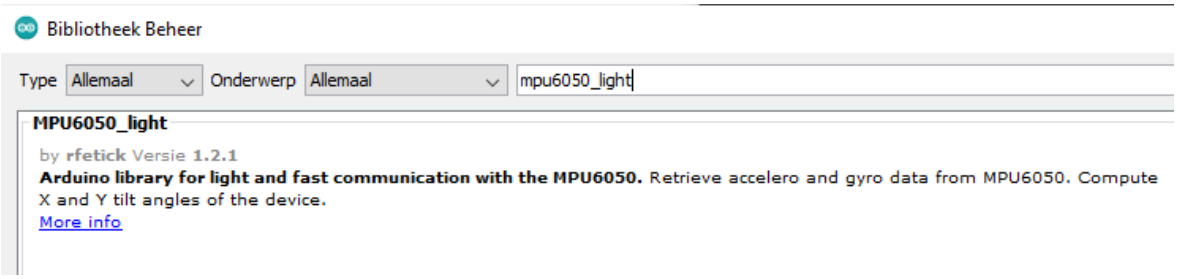
Tabel 4 Componenten Oef GY-521

Bron : Eigen werk

Componentenlijst
GY-521
Servo MG90S 180
Rode led
Groen led
Blauwe led
LCD 1602 met I2C

6.2.2.1 GY-521

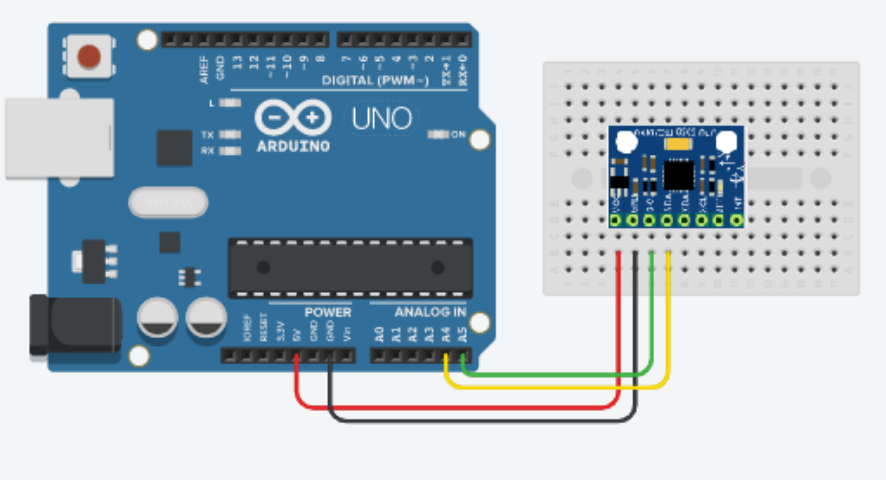
We gaan onze module gebruiken om een miniversie van een automatische piloot te maken, maar eerst moeten we de GY-521 module aan de praat krijgen. We starten dit project met het installeren van de bibliotheek “MPU6050_light”. We doen dit door de Arduino IDE te openen en naar schets te gaan, bibliotheek gebruiken en dan bibliotheken beheer (Ctrl + Shift + I). Daar zoeken we naar “MPU6050_light” en installeren de bibliotheek.



Afbeelding 31 MPU6050_light bibliotheek

Bron: Eigen werk m.b.v. Arduino IDE

Deze bibliotheek komt met voorbeelden, we openen “GetAngle”. Deze schets geeft de kantelhoeken op x en y, en de rotatiehoek op de z-as. We beginnen met de module aan de Arduino te schakelen. De GND-pin van de GY-521 is verbonden met de aarding van het Arduino en de VCC-pin van de GY-521 is aangesloten op 5 V. De SDA-pin van GY-521 is verbonden met de SDA-pin (A4) van Arduino en SCL-pin van de GY-521 is verbonden met de SCL- pin (A5).



Afbeelding 32 GY-521 met Arduino

Bron: Eigen werk m.b.v. *Tinkercad*, 2022

Als we kijken naar de code van de voorbeeldschets dan beginnen we met het opnemen van de bibliotheek “wire.h”. De bibliotheek is een standaard bibliotheek van de Arduino IDE en zorgt voor I2C communicatie. We nemen uiteraard ook de bibliotheek “MPU6050_light” op. We maken een mpu-object, waardoor we toegang krijgen tot functies van de bibliotheek en we maken een variabele timer die gelijk is aan 0.

```
#include "Wire.h"
#include <MPU6050_light.h>
MPU6050 mpu(Wire);
unsigned long timer = 0;
```

In de set-up van de code initialiseren we de seriële communicatie met de computer en de I2C communicatie. We maken een variabele status en vergelijken die met de mpu start. We controleren of die effectief start door de status te controlen. Als deze niet start blijft deze de status controleren, dit doen we met een while functie. Deze functie zorgt dat een programma stopt totdat een bepaalde voorwaarde waar is, in dit geval de status. Van zodra de status niet meer 0 is gaan we een boodschap printen naar de seriële monitor. De boodschap is dat we de module niet mogen verplaatsen omdat we hem gaan kalibreren. We kalibreren de module met een functie uit de bibliotheek. Als de kalibratie gedaan is printen we “Done!” naar de monitor.

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

  byte status = mpu.begin();
  Serial.print(F("MPU6050 status: "));
  Serial.println(status);
  while(status!=0){ } // stop everything if could not connect to MPU6050

  Serial.println(F("Calculating offsets, do not move MPU6050"));
  delay(1000);
  mpu.calcOffsets(); // gyro and accelero
  Serial.println("Done!\n");
}
```

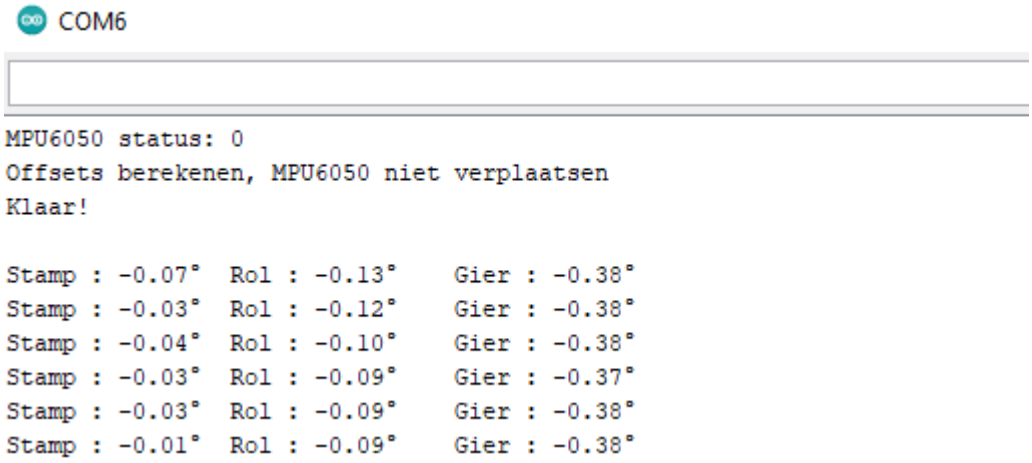
In de lus functie gaan we om de 10 ms de gegevens printen. We gaan nu de drie bewegingshoeken opvragen met “mpu.getAngle”, een functie van de bibliotheek (Fetick, 2022).

```
void loop() {
  mpu.update();

  if((millis()-timer)>10){ // print data every 10ms
        Serial.print("X : ");
```

```
        Serial.print(mpu.getAngleX());
        Serial.print("\tY : ");
        Serial.print(mpu.getAngleY());
        Serial.print("\tZ : ");
        Serial.println(mpu.getAngleZ());
        timer = millis();
}}
```

Als we de code nu uploaden naar de Arduino en we openen de seriële monitor van de Arduino IDE, krijgen we voor de drie bewegingen een waarde.

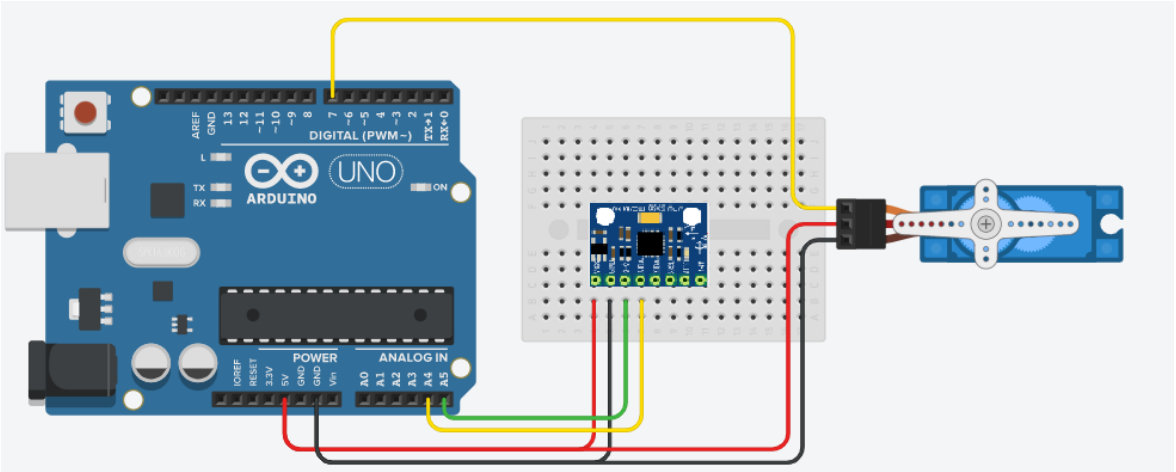


Afbeelding 33 GY-521 module seriële monitor

Bron: Eigen werk m.b.v. Arduino IDE

6.2.2.2 Automatische piloot

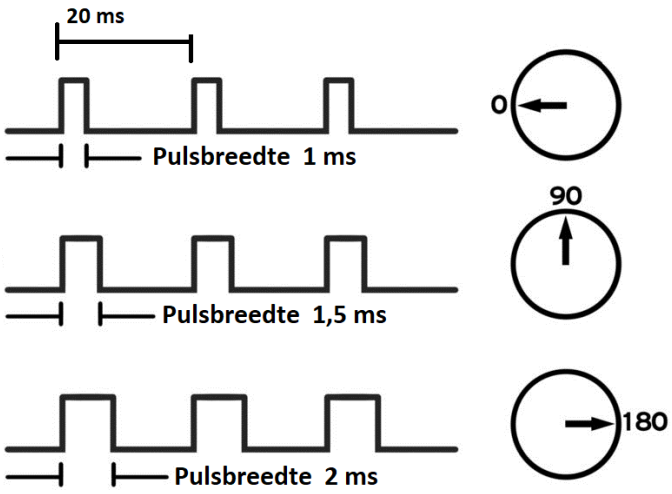
We kunnen nu informatie van onze GY-521 module krijgen. We gaan onze module gebruiken om een miniversie van een automatische piloot te maken. Hierbij gaan we een servo motor gebruiken die ons roer zal zijn en de module zal de scheepsbeweging geven. We schakelen de servo op onze Arduino.



Afbeelding 34 GY-521 met servo

Bron: Eigen werk m.b.v. *Tinkercad*, 2022

Een servomotor wordt bestuurd door een PWM signaal te sturen. De frequentie van het stuursignaal moet 50 Hz zijn of er moet elke 20 milliseconden een puls optreden. De breedte van de puls bepaalt de hoekpositie van de servo. De servo die we in deze oefening gebruiken heeft een bewegingslimiet van 180 graden.



Afbeelding 35 PWM signaal servo

Bron : EG Projects, 2022

Over het algemeen komen pulsen met een duur van 1 milliseconde overeen met een positie van 0 graden, een duur van 1,5 milliseconde met een positie van 90 graden en 2 milliseconden met een positie van 180 graden. Let wel op, de minimale en maximale duur van de pulsen variëren met verschillende merken. Ze kunnen 0,5 milliseconde zijn voor 0 graden en 2,5 milliseconden voor 180 graden positie. Dit kan je met wat testen snel zelf zoeken. In deze oefening gaan we kijken naar een handigere manier om een servo te besturen. Dat is met behulp van de bibliotheek (Dejan, 2018).

We werken verder op onze vorige schets. We starten met het toevoegen van de bibliotheek "servo.h", een standaard bibliotheek die we in de Arduino IDE kunnen vinden. We maken een servo-object, waardoor we toegang krijgen tot functies van de bibliotheek. We maken ook twee variabelen, de hoek van de servo ("servoAngle") en de scheepshoek ("AngleZ").

```
#include <Servo.h>      // servo-bibliotheek implementeren
Servo servo;    // object creëren om met servo-bibliotheek te werken
int servoAngle;         // variabele om de waarde op te slaan van de servo
int AngleZ;     // variabele om de waarde op te slaan van de z-as
```

In de set-up gaan we twee dingen toevoegen. We moeten de servo aan een pin linken, in dit voorbeeld gebruiken we de digitale pin 7. We doen dit met het commando "attach()". Met "write()" kunnen we een waarde naar de servo sturen en stelt dit de hoek van de as in (in graden). De servo die we hier gebruiken is een MG90 of SG90 met een draaihoek van 180 graden. We willen dat de servo in het begin van de oefening in het midden staat, we vullen dus 90 in.

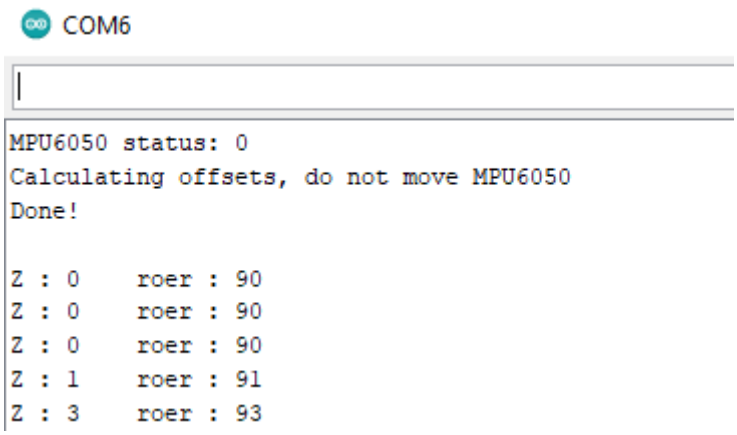
```
 servo.attach(7);       // selecteer de pin voor de servo
 servo.write(90);       // selecteer start hoek voor de servo(0-180)
```

In de lus functie gaan we nog steeds om de 10 ms de gegevens printen en nu ook de servo aansturen. We zijn geïnteresseerd in de rotatiehoek van Z, we vragen die rotatiehoek op en stellen deze gelijk aan "AngleZ". We printen deze waarde in de seriële monitor. We gaan nu de hoek van de servo bepalen aan de hand van de rotatiehoek. We doen dit met een map functie waar de waarde de rotatiehoek is en stellen -90 graden en 90 graden in als minimum en maximum van de rotatiehoek. De hoek van de servo gaat van 0 graden tot 180 graden en we stellen deze gelijk aan "servoAngle". We gaan nu deze waarde afprinten op de seriële monitor en doorsturen naar de servo met "write(servoAngle)" (Cox, 2020).

```
void loop() {
  mpu.update();
  if((millis()-timer)>10){ // gegevens elke 10 ms afdrukken
  AngleZ =  mpu.getAngleZ();   // gegevens z-as
  Serial.print("Z : ");
  Serial.print(AngleZ);        // print gegevens z-as

  servoAngle = map(AngleZ, -90, 90, 0, 180); // zet gegevens z-as om naar
gegevens servo
  Serial.print("\t roer : ");
  Serial.println(servoAngle); // print gegevens servo
  servo.write(servoAngle); // zet servo naar juiste hoek
  timer = millis();
  }}
```

Als we de code nu uploaden naar de Arduino en we openen de seriële monitor van de Arduino IDE krijgen we de rotatiehoek en de hoek van de servo.

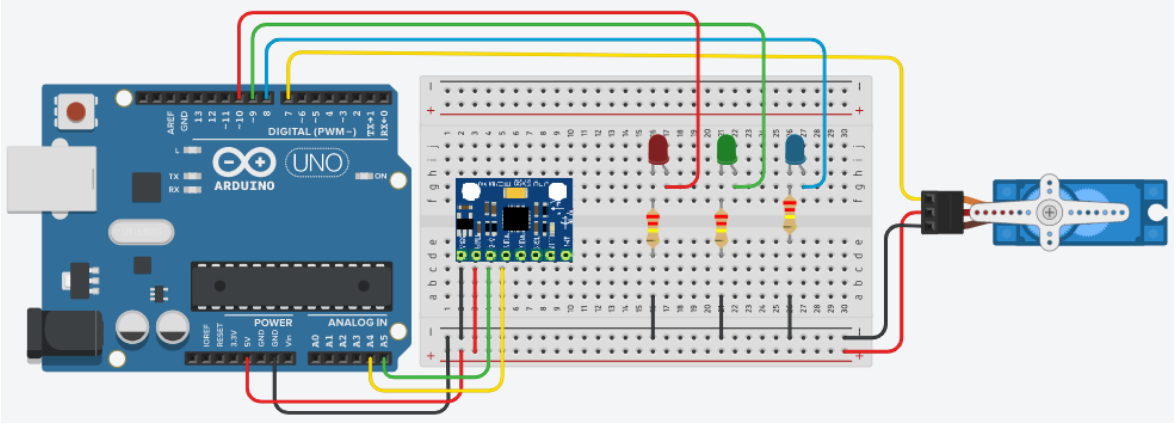


Afbeelding 36 seriële monitor GetAngle

Bron: Eigen werk m.b.v. Arduino IDE

6.2.2.3 Automatische piloot met led roer indicator

We gaan nu drie leds toevoegen, twee leds als roer indicator (rood en groen) en een derde led als indicator als we op de originele koers zijn. We schakelen alle 3 de leds elk met zijn bijbehorende voorschakelweerstand.



Afbeelding 37 GY-521 met leds

Bron: Eigen werk m.b.v. *Tinkercad*, 2022

We verklaren de drie leds met bijhorende namen en digitale pinnen.

```
int redLed = 10;        // selecteer de pin voor de rode led
int greenLed = 9;       // selecteer de pin voor de groene led
int blueLed = 8;        // selecteer de pin voor de blauwe led
```

In de set-up gaan we de drie digitale pinnen instellen als uitvoer.

```
  pinMode(redLed, OUTPUT);     // verklaar de groen led als een OUTPUT
  pinMode(greenLed, OUTPUT);   // verklaar de groene led als een OUTPUT
  pinMode(blueLed, OUTPUT);    // verklaar de blauwe led als een OUTPUT
```

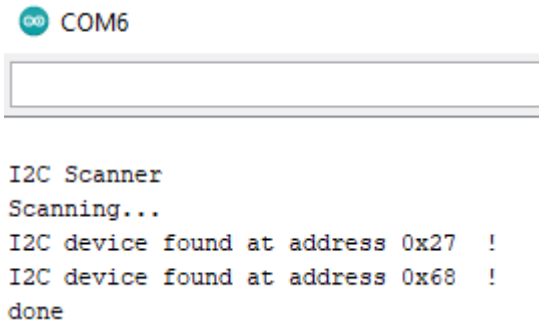
In de lus voegen we drie *if* statements toe, de waarde die we gaan vergelijken is de servo waarde. Als deze kleiner is dan 90 dan moet de rode led aan, is deze groter dan 90 de groene en als deze gelijk is aan 90 de blauwe (Bichon, 2021).

```
  if (servoAngle < 90 ) {      // controle van de servowaarde
  digitalWrite(greenLed, LOW);         // zet de groene led uit
  digitalWrite(blueLed, LOW);          // zet de blauwe led uit
  digitalWrite(redLed, HIGH);}         // zet de rode led aan
  if (servoAngle == 90 ) {     // controle van de servowaarde
  digitalWrite(greenLed, LOW);
  digitalWrite(blueLed, HIGH);
  digitalWrite(redLed, LOW);}
  if (servoAngle > 90 ) {      // controle van de servowaarde
  digitalWrite(greenLed, HIGH);
  digitalWrite(blueLed, LOW);
  digitalWrite(redLed, LOW);}
```

6.2.2.4 Automatische piloot met LCD roer indicator

In de volgende oefening gaan we een lcd-scherm toevoegen aan ons project. Hierbij gaan we een lcd schakelen met een I2C module. Het voordeel hiervan is dat er minder bekabeling nodig is. Het aansluiten van een I2C lcd is veel eenvoudiger dan het aansluiten van een standaard lcd. We weten dat, als de verschillende apparaten een verschillend adres hebben, ze op dezelfde SDA- en SCL-lijn geschakeld kunnen worden.

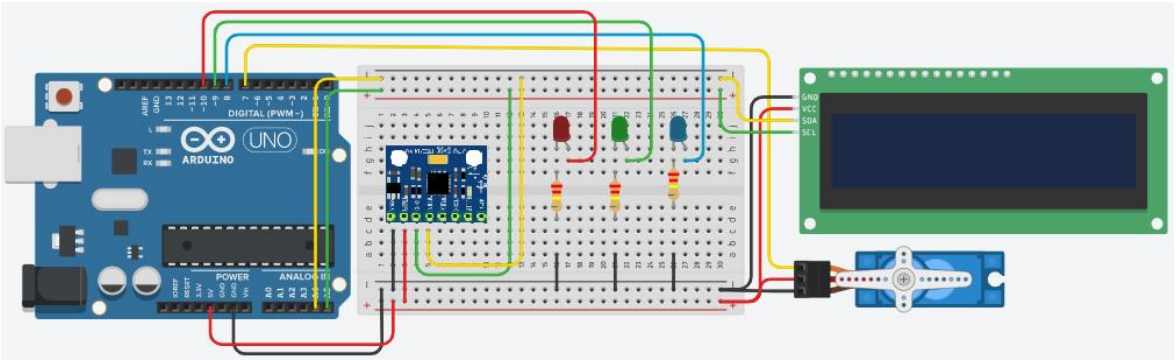
We kunnen het adres van onze componenten makkelijk vinden met een kant-en-klare scanner die we van het internet kopiëren. Als we de scanner uitvoeren en kijken naar onze seriële monitor krijgen we het volgende resultaat. (Kashif, 2022)



Afbeelding 38 seriële monitor I2C scanner

Bron: Eigen werk m.b.v. Arduino IDE

We vinden twee apparaten met twee verschillende adressen, 0x27 en 0x68. We hebben dus geen probleem en kunnen ze allebei gebruiken op dezelfde bus. We schakelen dus SCL en SDA van de GY-521 module samen met die van het lcd-scherm.

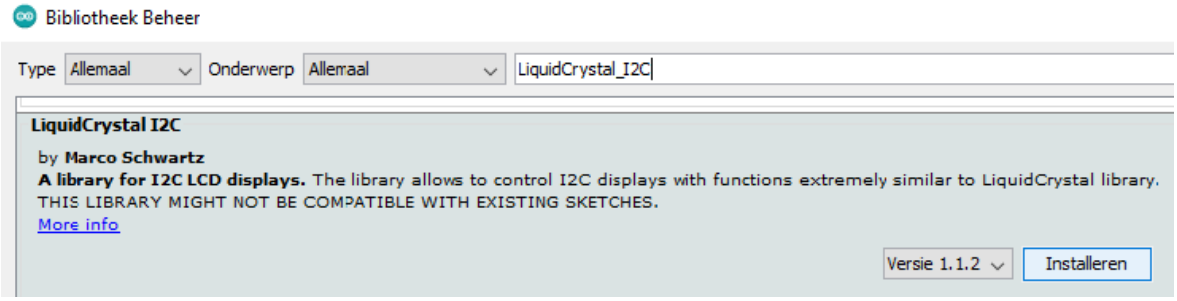


Afbeelding 39 GY-521 met lcd

Bron: Eigen werk m.b.v. *Tinkercad*, 2022

We gaan de bibliotheek LiquidCrystal_I2C installeren, die sterk lijkt op de LiquidCrystal-bibliotheek die standaard in de Arduino IDE staat. Met deze bibliotheek kunnen we het I2C-

scherm besturen. De functies van de I2C bibliotheek zijn zo goed als hetzelfde als die van de standaard bibliotheek.



Afbeelding 40 LiquidCrystal_I2C bibliotheek

Bron: Eigen werk m.b.v. Arduino IDE

We voegen onze bibliotheek toe aan de schets en verklaren de hoek die het roer maakt. We stellen de lcd in met adres 0x27 en de grootte van het scherm, zestien kolommen en twee rijen.

```
#include <LiquidCrystal_I2C.h> // LiquidCrystal-bibliotheek implementeren
int rudderAngle; // variabele om de waarde op te slaan van de roerhoek
LiquidCrystal_I2C lcd(0x27,16,2); // stel het adres, het aantal kolommen
en het aantal rijen in
```

In de set-up voegen we “init()”toe, dit is voor het initialiseren van het lcd-scherm. Met “backlight()” gaat de achtergrondverlichting van het scherm aan.

```
  lcd.init();           // initialiseer het lcd-scherm
  lcd.backlight();      // schakel de achtergrondverlichting in
```

In de lus starten we met het berekenen van de roerhoek. Dit is de absolute waarde van de servo hoek min 90 graden.

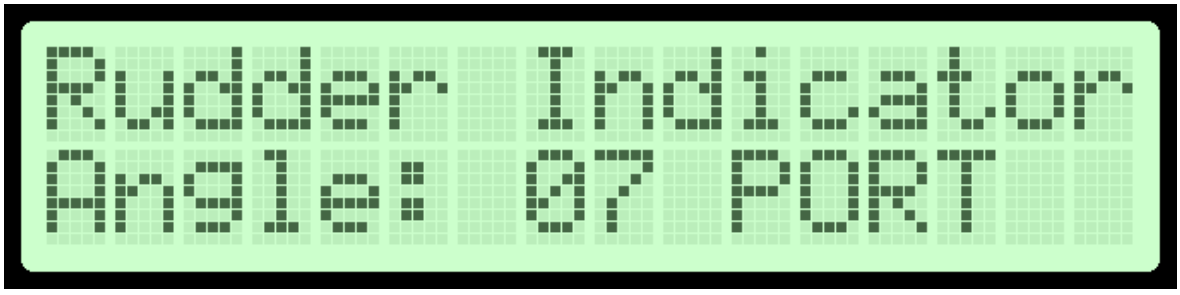
```
 rudderAngle = abs(servoAngle-90);     // bereken de roerhoek
```

We werken verder op onze drie *if*-instructies. We nemen één instructie als voorbeeld. Met “setCursor” kunnen we de locatie bepalen waar we moeten printen op de lcd. Bijvoorbeeld 0,1 is kolom nul en rij één, de eerste kolom en rij start bij 0. Met “print()” kunnen we tekst en waardes printen naar de lcd, alle tekens tussen aanhalingstekens kunnen we rechtstreeks printen. We voegen ook een *if* functie toe, wanneer de roerhoek waarde kleiner is dan 10 dan zetten we er een 0 voor. We doen dit omdat een karakter op de lcd gewist moet worden of overschreven anders blijft deze staan. In deze oefening zou dan de 0 van 10 blijven staan.

We zouden hier de “clear()” functie kunnen gebruiken. De “clear()” functie zou bij elke lus het scherm wissen, wat voor flikkerend scherm zorgt. Daarom gaan we de waarde overschrijven. We printen op plaats 9,1 of we stuurboord, bakboord of midscheeps zijn.

```
  if (servoAngle < 90 ) {      // als servo hoek kleiner is als 90
  digitalWrite(greenLed, LOW); // zet de groene led uit
  digitalWrite(blueLed, LOW);  // zet de blauwe led uit
  digitalWrite(redLed, HIGH);  // zet de rode led aan
  lcd.setCursor(0,0); // zet de cursor van de lcd op de positie (kolom 0,
rij 0)
  lcd.print("Rudder Indicator");      // print af op lcd
  lcd.setCursor(0,1);  // zet de cursor van de lcd op de positie (0,1)
  lcd.print("Angle:"); // print af op lcd
  lcd.setCursor(7,1);  // zet de cursor van de lcd op de positie (7,1)
  if (rudderAngle < 10){ lcd.print("0");} // als de roerhoek kleiner is
dan 0 print “0”
  lcd.print(rudderAngle);      // print gegevens roerhoek af op lcd
  lcd.setCursor(9,1);  // zet de cursor van de lcd op de positie (9,1)
  lcd.print (" PORT");}        // print af op lcd
```

Als we nu alles uploaden krijgen we het volgende op ons lcd-scherm. (*Arduino Project Hub*, 2022; *LastMinuteEngineers*, 2020)



Afbeelding 41 Rudder Indicator lcd

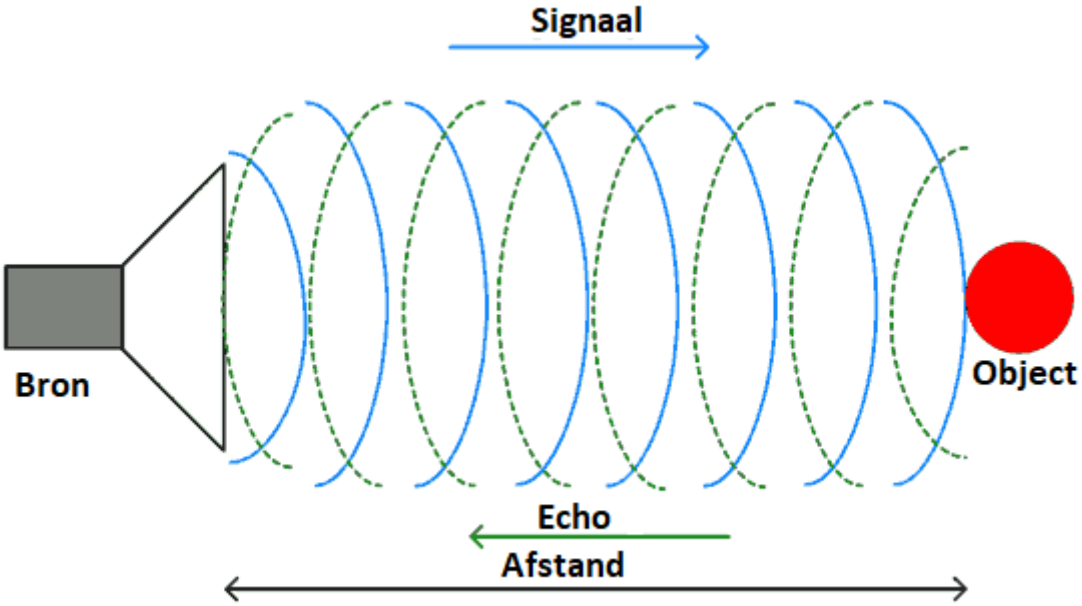
Bron: Eigen werk m.b.v. *LCD Generator*, 2022

6.3 Ultrasonische Sensor (HC-SR04)

In dit Arduino project gaan we kijken hoe we de HC-SR04 ultrasone afstandssensor kunnen gebruiken.

6.3.1 Algemene informatie over de HC-SR04 module

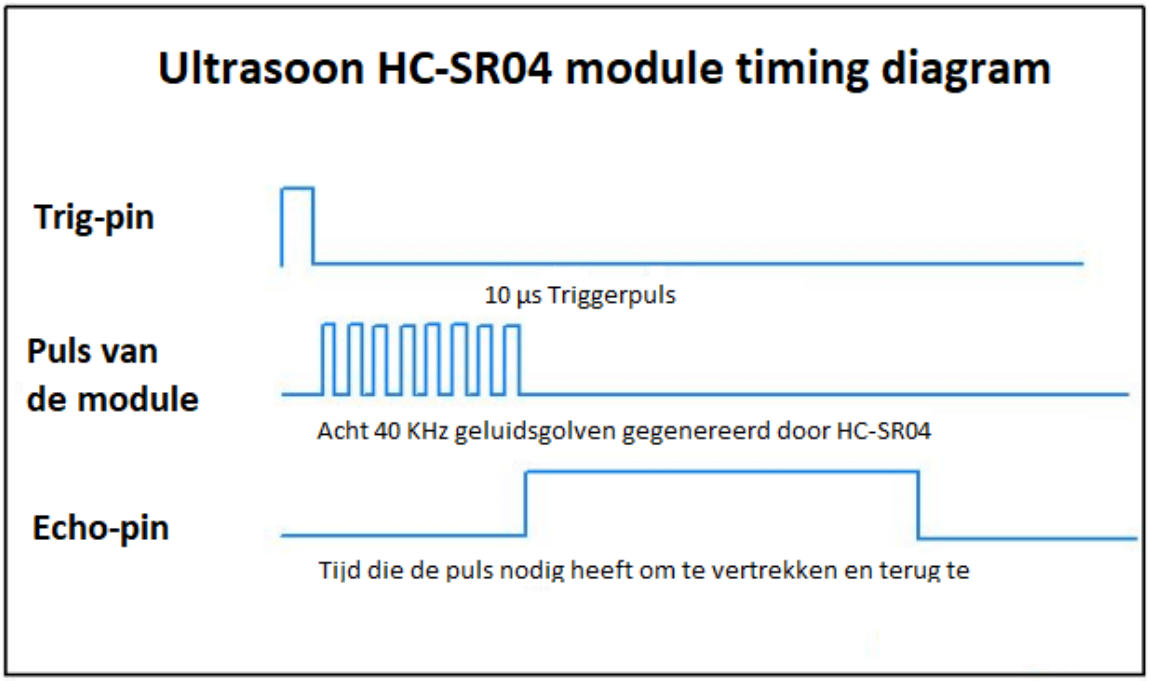
De ultrasone sensor werkt volgens het radarsysteem dat wordt gebruikt om de afstand tot een object te bepalen. Een ultrasone sensor genereert een geluidsgolf met hoge frequentie (echografie). Wanneer dit geluid het object raakt, wordt het weerkaatst als echo die door de ontvanger wordt waargenomen, zoals weergegeven in onderstaande afbeelding.



Afbeelding 42 HC-SR04 afstandssensor principe

Bron: *Sprout Robotic Solutions*, 2019

Een HC-SR04 ultrasone afstandssensor bestaat eigenlijk uit twee ultrasone transducers. De ene fungeert als zender (*Trig*) die het elektrische signaal omzet in 40 kHz ultrasone geluidspulsen. De andere fungeert als ontvanger (Echo) en luistert naar de uitgezonden pulsen. Wanneer de ontvanger deze pulsen ontvangt, produceert deze een uitgangspuls waarvan de breedte evenredig is met de afstand van het object ervoor. Deze sensor biedt een uitstekende contactloze detectie van een bereik tussen 2 cm en 400 cm met een nauwkeurigheid van 3 mm. (*Electronicwings*, 2018)



Afbeelding 43 HC-SR04 module tijds diagram

Bron: Jabbaar, 2019

We kijken nu naar de werking van de HC-SR04 ultrasone afstandssensor. Het begint wanneer de triggerpin 10 µs HOOG is ingesteld. Als reactie zendt de sensor een ultrasone patroon uit van acht pulsen op 40 kHz. Dit 8-pulsenpatroon is speciaal ontworpen zodat de ontvanger de uitgezonden pulsen kan onderscheiden van ultrasoon omgevingsgeluid.
Deze acht ultrasone pulsen reizen door de lucht weg van de zender. Ondertussen gaat de echo-pin HOOG om het echo-back-signaal te initiëren.

Als die pulsen niet worden teruggekaatst, treedt er een time-out van het echosignaal op en wordt het laag na 38 ms (38 milliseconden). Dus een puls van 38 ms geeft aan dat er geen obstructie is binnen het bereik van de sensor. Als die pulsen worden teruggekaatst, gaat de echo-pin laag zodra het signaal wordt ontvangen. Dit genereert een puls op de echo-pin waarvan de breedte varieert van 150 µs tot 25 ms, afhankelijk van de tijd die nodig is om het signaal te ontvangen>.

De breedte van de ontvangen puls wordt gebruikt om de afstand tot het gereflecteerde object te berekenen. Dit kan worden uitgewerkt met behulp van de eenvoudige afstand-snelheid-tijdvergelijking (Afstand = Snelheid * Tijd).

Laten we een voorbeeld nemen om het duidelijker te maken. Stel dat we een object voor de sensor hebben op een onbekende afstand en we ontvangen een puls van 900 µs breed op de echo-pin. Laten we nu berekenen hoe ver het object zich van de sensor bevindt. We hebben de waarde van tijd, 900 µs en we kennen de snelheid. Dit is natuurlijk het de snelheid van het geluid. Het is 340 m/s. Om de afstand te berekenen moeten we de geluidssnelheid omrekenen naar cm/µs. Het is 0,034 cm/µs. Met die informatie kunnen we nu makkelijk de afstand berekenen. Maar vergeet niet dat de echopuls aangeeft hoelang het duurt voordat het signaal wordt verzonden en teruggekaatst. Dus om de afstand te krijgen, moet je je resultaat door twee delen (Jabbaar, 2019).

$$Afstand = \frac{Snelheid * Tijd}{2} = \frac{900\ \mu s * 0{,}034\ \mathrm{cm}/\mu}{2} = 15{,}3\ \mathrm{cm} \qquad (1)$$

6.3.2 Ultrasonische Sensor met Arduino

In dit laatste deel van hoofdstuk zes gaan we ons baseren op een onderzoeksrapport van het passagiersschip Akademik Loffe en de kennis van de ultrasone sensor. Hierbij gaan we het technische aspect van het artikel namaken in een Arduino Project

In Bijlage 3 : Artikel vinden we een rapport van de Canadese onderzoeksraad voor transportveiligheid. In het rapport lezen we dat de Akademik Loffe aan de grond liep, 78 zeemijl ten noordwesten van Kugaaruk, Nunavut. De Akademik Loffe voer door een versmalling in een afgelegen gebied van het Canadese poolgebied. Het gebied was niet onderzocht volgens moderne of adequate hydrografische normen, en niemand van de bemanningsleden was er ooit geweest.

Het schip liep met een snelheid van 7,6 knopen aan de grond. De personen op de brug hielden de dieptemeters niet nauwlettend in de gaten en de afname van de waterdiepte onder de kiel bleef meer dan vier minuten onopgemerkt. Dit gebeurde omdat de laagwateralarmen van de dieptemeters waren uitgeschakeld.

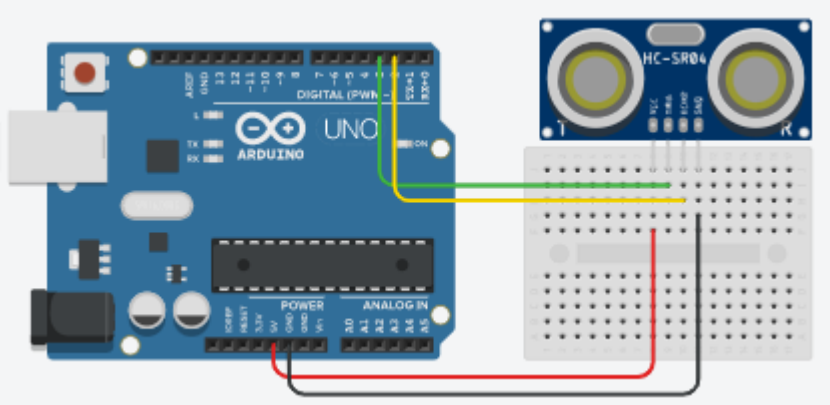
Tabel 5 componenten oef HC-SR04

Bron : Eigen werk

Componentenlijst
HC-SR04 Ultrasonische afstandssensor
LCD blauw 1602
Piezo buzzer
Infrarood Ontvanger
Infrarood afstandsbediening

6.3.2.1 Echo sounder

We gaan onze sensor gebruiken om een kleine variant van een echo sounder te maken, maar eerst moeten we de HC-SR04 module aan de praat krijgen. We schakelen de module, de *trig*-pin en echo-pin schakelen we op twee digitale pinnen.



Afbeelding 44 Arduino met HC-SR04

Bron: Eigen werk m.b.v. *Tinkercad*, 2022

Voor de code beginnen we eerst met de *trig*-pin en echo-pin te definiëren, in dit project is de echo-pin gekoppeld aan D2 en *trig*-pin aan D3. We definiëren vervolgens variabelen voor de duur en afstand. We werken hier met kleine waarden (microseconden), daarom definiëren we de duur als een “long” data type. “long” variabelen zijn variabelen met een uitgebreide grootte voor nummeropslag en slaan 32 bits (4 bytes) op. (*Arduino Reference*, 2020)

```
#define echoPin 2 // definieer pin nummer van echopin
#define trigPin 3 // definieer pin nummer van echopin
long duration; // variabele om de waarde op te slaan van de tijd (long
data type)
int distance; // variabele om de waarde op te slaan van de afstand
```

In de set-up configureren we de *trig*-pin als uitgang en de echo-pin als ingang.

```
void setup() {
  pinMode(trigPin, OUTPUT); // verklaar de trigpin als een OUTPUT
  pinMode(echoPin, INPUT);  // verklaar de echopin als een INPUT
  Serial.begin(9600); }
```

We weten uit het voorgaande stuk (6.1.1) dat als de *trig*-pin tien microseconden hoog ingesteld is, dan gaat de module een signaal uitsturen dat we kunnen ontvangen. We beginnen de lus met de *trig*-pin laag te houden. We doen dit twee microseconden, met de functie “delayMicroseconds(µs)”. Daarna gaan we ze tien microseconden hoog instellen met daaropvolgend terug laag. Daarna gaan we wachten op een puls die wordt ontvangen

door de echo-pin, we doen dit met “pulseIn(pin, waarde)”. Daarna gaan we de tijd die tussen het uitzenden en ontvangen van de puls omzetten naar een afstand ($Afstand = \frac{Snelheid * Tijd}{2}$). We printen vervolgens deze afstand af op de seriële monitor. (Dejan, 2015)

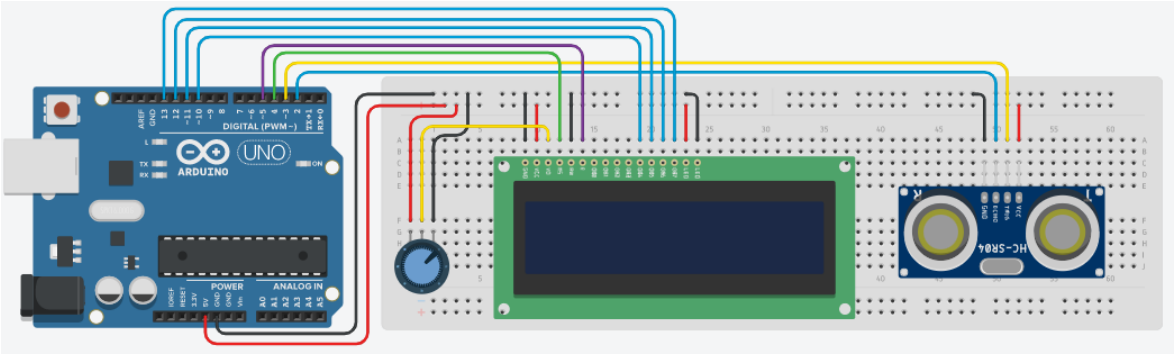
```
void loop() {
  digitalWrite(trigPin, LOW);  // zet de trigpin uit
  delayMicroseconds(2);        // pauze van 2 microseconden
  digitalWrite(trigPin, HIGH); // zet de trigpin aan
  delayMicroseconds(10);       // pauze van 10 microseconden
  digitalWrite(trigPin, LOW);  // zet de trigpin uit
  duration = pulseIn(echoPin, HIGH); // leest de puls op een de echopin
en stel deze waarde gelijk aan de tijd
  distance = duration * 0.034 / 2; // snelheid van geluidsgolf maal de
tijd gedeeld door 2 (heen en terug)
  Serial.print("Distance: ");
  Serial.print(distance);      // print afstand
  Serial.println(" cm");}
```

6.3.2.2 Echo sounder met LCD

In de volgende oefening gaan we een lcd-scherm toevoegen aan onze dieptemeter. Hierbij gaan we nu een lcd schakelen zonder een I2C module. De lcd heeft zestien pinnen, de eerste pinnen die we schakelen zijn de eerste twee pinnen en de laatste twee pinnen. De eerste twee zijn de spanning en aarding voor de lcd, en de laatste twee zijn de spanning en aarding voor de led van de lcd. De 3de pin is de V0 pin of contrastaanpassing-pin. We gaan hier een variabele weerstand schakelen, zoals een potentiometer. Als we deze vijf pinnen hebben aangesloten en de USB-aansluiting van de Arduino aansluiten, kunnen we het contrast aanpassen met de potentiometer.

Pinnen 4–6 worden gebruikt om de prestaties van het beeldscherm te regelen. De vierde pin is de RS-pin (*Register Select*), deze is verantwoordelijk voor het selecteren van een van de twee weergaveregisters (dataregister en instructieregister). De bibliotheek regelt de RS-pin en we sluiten deze aan op een digitale pin. De volgende pin is de RW-pin (*read/write*), deze kan gegevens uit de lcd-dataregister lezen en instructies sturen. We sluiten de RW-pin van de lcd aan met de aarding, omdat we nooit gegevens uit de lcd-dataregister zullen lezen, maar alleen instructies zullen sturen die u op de lcd wilt weergeven. Pin zes van de lcd (E) is een pin die wordt gebruikt om de LCD in te schakelen, deze sluiten we aan op een digitale poort. (Sitompul & Sihombing, 2022)

De volgende 8 pinnen zijn datapinnen (D0 – D7), er zijn twee mogelijkheden waarmee het scherm kan worden gekoppeld. De eerste mogelijkheid gebruikt de vier datapinnen van D4 tot D7 en de andere gebruikt alle acht datapinnen van de lcd. In de modus met vier pinnen zijn de gegevens die vanuit Arduino naar de displaymodule worden verzonden in 4 bit en met acht pinnen in 8 bit. We gaan de 4 bits modus schakelen want het voordeel is dat er minder pinnen van de Arduino bezet zijn en dat de overige pinnen voor andere doeleinden kunnen worden gebruikt. (Javaid, 2022b)



Afbeelding 45 HC-SR04 met lcd

Bron: Eigen werk m.b.v. *Tinkercad*, 2022

We werken verder op de vorige oefening. We voegen de bibliotheek “LiquidCrystal” toe, dit is een standaard bibliotheek van de Arduino IDE en kan de lcd besturen. We creëren zes variabelen van onze verbindingen tussen de lcd en de digitale pinnen van de Arduino. We initialiseren de bibliotheek en maken een lcd-object met de parameters RS, EN, D4, D5, D6 en D7.

```
#include <LiquidCrystal.h> // LiquidCrystal-bibliotheek implementeren
// initialiseer de bibliotheek door elke benodigde lcd-pin te koppelen
// met het arduino-pinnummer waarmee het is verbonden
const int rs = 4, en = 5, d4 = 10, d5 = 11, d6 = 12, d7 = 13;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
```

In de set-up Initialiseren we het lcd-scherm en specificeert de afmetingen (breedte en hoogte) van het scherm.

```
 lcd.begin(16,2); // stel het aantal kolommen en rijen van het lcd-scherm
in (16,2)
```

In de lus is het principe hetzelfde als met een I2C lcd. We zeggen waar de cursor moet staan en printen de informatie naar het scherm. (Söderby, 2022)

```
  lcd.setCursor(0,0); // zet de cursor van de lcd op de positie (kolom 0,
rij 0)
  lcd.print("Distance: ");      // print af op lcd
  lcd.print(distance);          // print afstand af op lcd
  lcd.print(" cm");             // print af op lcd
  delay(500);                   // pauze van een halve seconden
```

6.3.2.3 Echo sounder met alarm

We gaan in de volgende oefening een buzzer toevoegen. De buzzer moet afgaan wanneer de afstand over een bepaalde grens gaat. De buzzer moet ook sneller beginnen piepen als deze afstand kleiner wordt. We maken een object van de buzzer en linken deze aan een digitale pin.

```
int buzzer = 7;          // selecteer de pin voor de buzzer
```

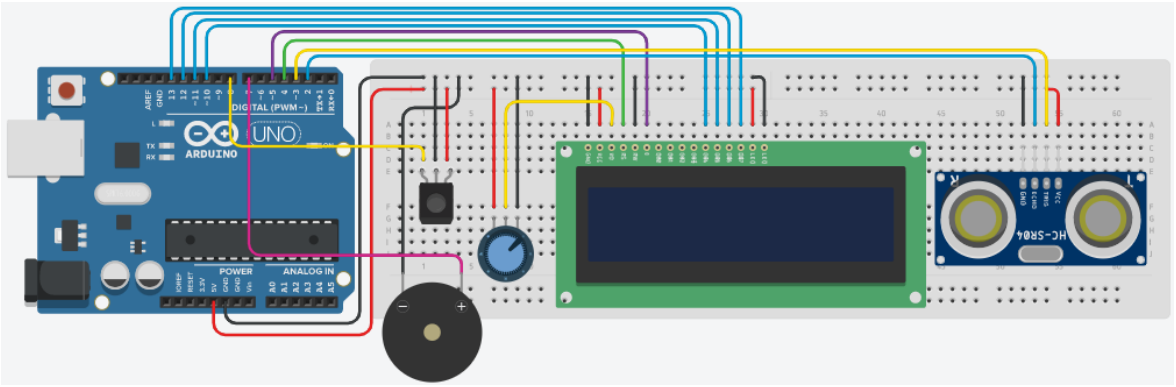
In de lus functie gaan we meerder *if*-instructies maken. Als de afstand minder is dan twintig centimeter dan gaat de buzzer af. We doen dit met een functie die we maken buiten onze lus functie. We laten deze toon 200 milleseconden afgaan met daaropvolgend een pauze van 1000 milleseconden. We doen dit nu meermaals voor kleinere afstanden en gaan de pauze verkleinen. Zo krijgen we telkens een snellere toon bij een kleinere afstand. We geven deze functies de naam “Tone” met bijpassend nummer.

```
  if (distance < 20){ Tone0 (); } // als afstand < 20 roep de Tone0
functie op
  if (distance < 15){ Tone1 (); } //als afstand <  15 roep de Tone1
functie op
  if (distance < 10){ Tone2 (); } // als afstand <  10 roep de Tone2
functie op
  if (distance < 5){ Tone3 (); }// als afstand < 5 roep de Tone3 functie
op
}
```

```
void Tone0 () {                 // maken van functie tone0
  digitalWrite(buzzer, HIGH); // zet de buzzer aan voor 200 milliseconden
  delay(200);
  digitalWrite(buzzer, LOW); // zet de buzzer aan voor 1000 milliseconden
  delay(1000);          }
```

6.3.2.4 Echo sounder met IR bypass

Onze echo sounder heeft nu vier verschillende alarmen die afgaan bij verschillende afstanden. We gaan hier nu een bypass systeem op toepassen zodanig dat we ze individueel kunnen in- en uitschakelen. We zouden dit met vier drukknoppen kunnen doen, maar we gaan hier gebruik maken van een infrarood afstandsbediening of kort IR-afstandsbediening. We gaan hier een infrarood ontvanger schakelen. Een infrarood ontvanger komt meestal voor in de vorm van een aparte diode of een diode op een bordje geschakeld. In ons geval is het een aparte diode met drie pinnen, een voedingspin, de aarding en een uitgangspin die we schakelen op een digitale pin (in dit voorbeeld D8).



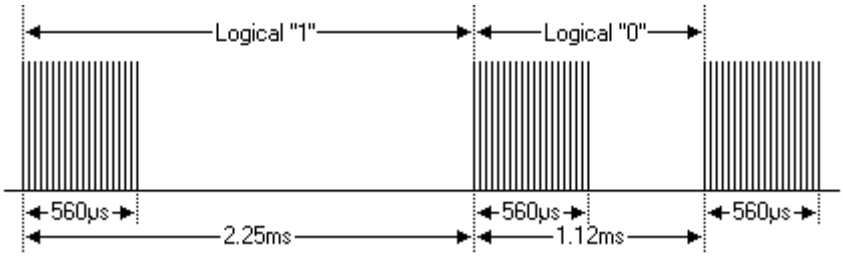
Afbeelding 46 HC-SR04 met bypass

Bron: Eigen werk m.b.v. *Tinkercad*, 2022

Voor we beginnen met onze code te schrijven van onze IR-afstandsbediening gaan we eerst kijken hoe deze juist werkt. De IR-afstandsbediening en de ontvanger communiceren met elkaar door een signaal in de vorm van een gepulseerde IR-straling te verzenden en te decoderen. IR-straling is een type elektromagnetische straling met een golflengten van 700 nm tot 1 mm. Mensen kunnen alleen licht zien met golflengten van ongeveer 400 nm tot 700 nm, daarom is IR-straling onzichtbaar voor het menselijk oog.

Elk object dat een temperatuur heeft, straalt uit in het infrarode spectrum. Al deze IR stralingen kan de communicatie tussen de afstandsbediening en de ontvanger verstoren, om dit tegen te gaan maken we gebruik van signaalmodulatie. De IR-lichtbron aan het uiteinde van de afstandsbediening knippert met een bepaalde frequentie. In consumentenelektronica ligt deze draaggolffrequentie meestal rond de 38 kHz (deze frequentie komt zelden voor in de natuur).

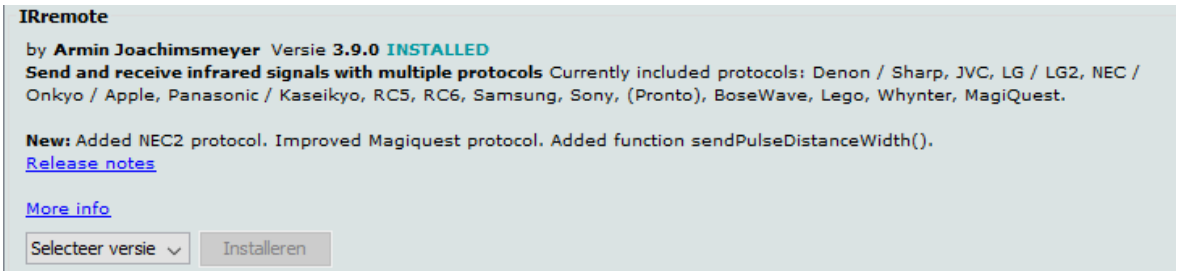
De ontvanger is zo gebouwd dat hij alleen IR doorlaat dat binnenkomt op 38 kHz. Het gedemoduleerde binaire signaal wordt vervolgens naar de Arduino gestuurd waar het wordt gedecodeerd. Onze IR-afstandsbediening werkt volgens het NEC-protocol waar het totale bericht meestal bestaat uit vier bytes. Wat belangrijk is, is de hoeveelheid tijd dat de IR-lichtbron aan het uiteinde van de afstandsbediening hoog of laag wordt gehouden. In het NEC-protocol, worden de bits als volgt weergegeven. Elke bit bestaat uit een 560 µs lange draaggolf van 38 kHz, gevolgd door een pauze. Een logische "1" heeft een totale verzendtijd van 2,25 ms, terwijl een logische "0" slechts 1,125 ms heeft. Zo kunnen volledige signalen versturen. (Pattabiraman, 2017)



Afbeelding 47 NEC-protocol

Bron: Rubashka, 2022

Ne we weten hoe het principe van de IR-afstandsbediening werkt kunnen we aan de slag met onze code. Het eerste wat we gaan doen is een bibliotheek toevoegen aan de Arduino IDE. De bibliotheek om met de IR-afstandsbediening te communiceren is “IRremote”.



Afbeelding 48 IRremote bibliotheek

Bron: Eigen werk m.b.v. Arduino IDE

Nu we de bibliotheek hebben geïnstalleerd gaan we kijken naar welk signaal een knop uitstuurt. We gaan hiervoor een klein programma gebruiken “IRrecvDemo”, dit zat vroeger standaard in de bibliotheek. We kunnen dit programma snel terugvinden op het internet. De code start natuurlijk met het toevoegen van de bibliotheek. We definiëren de pin van de IR-ontvanger en we creëren twee objecten voor de ontvanger en het resultaat.

```
#include <IRremote.hpp> // IRremote-bibliotheek implementeren
#define RECEIVER_PIN 8 // definieer pin nummer van de IR ontvanger
IRrecv receiver(RECEIVER_PIN); // maak een ontvanger object voor de bib
IRrecv
decode_results results; // maak een resultaat object voor de bib IRrecv
```

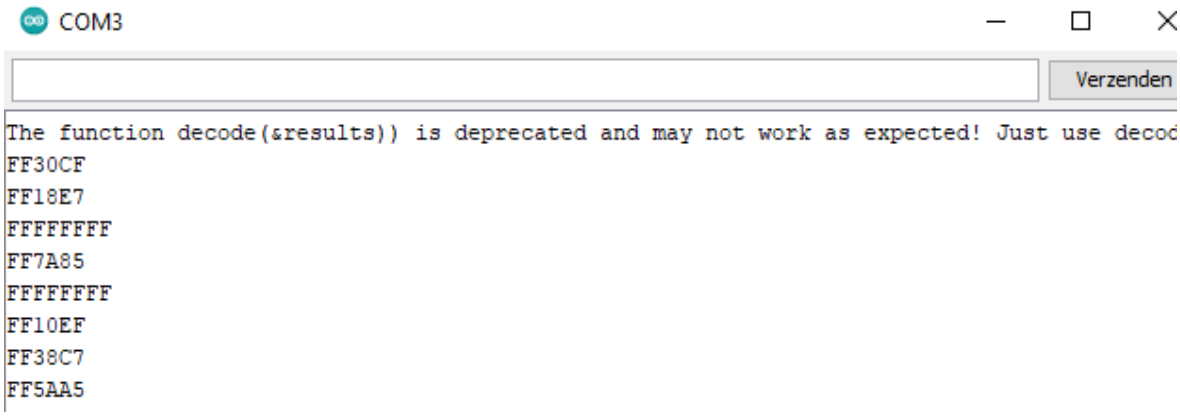
We starten de ontvanger.

```
void setup() {
  Serial.begin(9600);
  receiver.enableIRIn(); } // schakel de ontvanger in
```

Als we een signaal ontvangen, decoderen we het en slaan het op in het resultaat. We drukken het resultaat af in de seriële monitor en resetten de ontvanger voor de volgende code.

```
void loop() {
  if (receiver.decode(&results)) {  // het ontvangen signaal decoderen en
opslaan in results
    Serial.println(results.value, HEX); // print het resultaat af in
hexadecimale getallen op de seriële monitor
    receiver.resume(); }} // reset de ontvanger voor de volgende code
```

Als we dan gaan kijken naar de seriële monitor en wat knoppen van de afstandsbediening gaan indrukken krijgen we het volgende resultaat.



Afbeelding 49 IR ontvanger seriële monitor

Bron: Eigen werk m.b.v. Arduino IDE

De codes die beginnen met FF en dan cijfers of letters hebben, zijn de codes waar we naar op zoek zijn. Een FFFFFFFF is storing, als we de knop te lang ingedrukt houden blijft het signaal hoog en krijgen we dus FFFFFFFF op de seriële monitor. We kunnen nu alle knoppen van de afstandsbediening afgaan en krijgen dit als resultaat. (Benne, 2020; Pattabiraman, 2017)

Tabel 6 IR remote codes

Bron: Eigen werk

FFA25D	FF629D	FFE21D
FF22DD	FF02FD	FFC23D
FFE01F	FFA857	FF906F
FF6897	FF9867	FFB04F
FF30CF	FF18E7	FF7A85
FF10EF	FF38C7	FF5AA5
FF42BD	FF4AB5	FF52AD

Afbeelding 50 IR afstandsbediening

Bron: *Makers store*, 2022

We weten nu de codes van alle knoppen, nu kunnen we teruggaan naar ons project. We gaan onze code zo schrijven dat als we op knop nummer 1 drukken het alarm van 20 cm aan of uit gaat, we doen dit ook voor 15 cm met knop 2 enzovoort. We starten de code met hetzelfde als “IRrecvDemo”. We voegen de bibliotheek toe, definiëren de pin van de IR-ontvanger en we creëren 2 objecten voor de ontvanger en het resultaat. We creëren een variabele “key_value” en stellen deze gelijk aan 0. We definiëren vier codes, dit zijn de hexadecimale codes van knop 1 tot en met knop 4. We zetten het voorvoegsel “0x” voor de code om aan te geven dat het nummer in hexadecimaal wordt geschreven.

We zouden hier nu ook vier variabelen kunnen maken voor de 4 toestanden van de knoppen, maar we kunnen dit ook efficiënter aanpakken. We gaan hier gebruik maken van een *array* of reeks, dit is een verzameling variabelen die toegankelijk is met een indexnummer. Een reeks is op nul geïndexeerd, dus het eerste element van de reeks staat op index 0. We krijgen de waarde van de eerste variabele met “status[0]”.

```
#include <IRremote.hpp>  // IRremote-bibliotheek implementeren
#define RECEIVER_PIN 8  // definieer pin nummer van de IR ontvanger
IRrecv receiver(RECEIVER_PIN); // maak een ontvanger object voor de bib
IRrecv
decode_results results; // maak een resultaat object voor de bib IRrecv
unsigned long key_value = 0;  // variabele om de waarde op te slaan van
de key_value en stel gelijk aan 0
#define code1 0xFF30CF         // code ontvangen van knop 1
#define code2 0xFF18E7         // code ontvangen van knop 2
#define code3 0xFF7A85         // code ontvangen van knop 3
#define code4 0xFF10EF         // code ontvangen van knop 4
```

```
int status[] = {0,0,0,0}; // 4 variabelen om de waarde op te slaan van de status
```

In de set-up voegen we enkel de opstart van de infrarood ontvanger toe.

```
 receiver.enableIRIn();        // schakel de ontvanger in
```

We voegen aan onze instructies die ons alarm laten afgaan een extra conditie. Deze conditie is als de waarde van de status gelijk is aan 1. We gaan dus in het volgende deel van de code regelen hoe we de status kunnen aanpassen.

```
  if (distance > 15 && distance <= 20 && status[0] == 1){Tone0();}
// als de afstand tussen 15&20 is en status[0] gelijk aan 1 roep dan de Tone0 functie op
 else if (distance > 10 && distance <= 15 && status[1] == 1){Tone1();}
// als de afstand tussen 10&15 is en status[1] gelijk aan 1 roep dan de Tone1 functie op
 else if (distance > 5 && distance <= 10 && status[2] == 1){Tone2();}
// als de afstand tussen 5&10 is en status[2] gelijk aan 1 roep dan de Tone2 functie op
 else if (distance <= 5 && status[3] == 1){Tone3();}
// als de afstand < 5 is en status[3] gelijk aan 1 roep dan de Tone3 functie op
```

Als we een signaal ontvangen, decoderen we het en slaan het op in het resultaat. We voegen ook een *if*-instructie toe, dat als het ontvangen resultaat gelijk is aan 0xFFFFFFFF dan zeggen dat de waarde gelijk is aan 0. Hiermee werken we de 0xFFFFFFFF fout weg. Als er geen fout is opgemerkt gaan we naar een switch-instructie. We doen dit zoals we weten uit het labo van het derde jaar (5.2.2.1), vergelijkt een switch-instructie een specifieke conditie, hier gaan we de conditie gelijkstellen aan de hexadecimale code van de verschillende knoppen.

```
 if (receiver.decode(&results)) {  // het ontvangen signaal decoderen en opslaan in results
 if (results.value == 0xFFFFFFFF) { // als de waarde gelijk is aan 0xFFFFFFFF
  results.value = key_value; // stel de waarde in op de key_value
       switch (results.value) { // vergelijk de waarde met de volgende cases
```

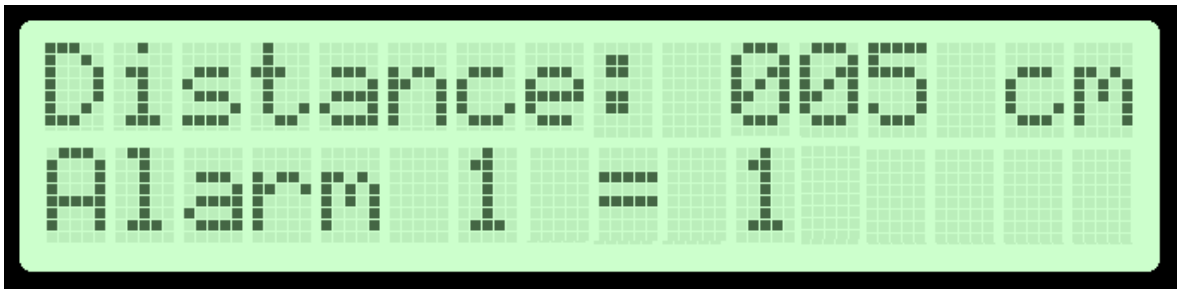
We nemen de code voor knop 1 als voorbeeld. Als we knop 1 indrukken gaan we de case overlopen. We vergelijken de status waarde als deze gelijk is aan 1 zetten we ze naar 0 en visa versa. We printen de status van het alarm uit op de seriële monitor en op de tweede rij van de lcd.

```
 case code1: // als het resultaat gelijk is aan code van knop1
    if(status[0] == 1) { // als de status[0] gelijk is aan 1 dan
    status[0] = 0;}    // stel status[0] gelijk aan 0
    else {             // anders is de status[0] gelijk  aan 0 dan
    status[0] = 1;}    // stel status[0] gelijk aan 1
    Serial.print("Alarm 1 = "); //print af op seriele monitor
    Serial.println(status[0]);
    lcd.setCursor(0,1);        // zet de cursor van de lcd op (0,1)
    lcd.print("Alarm 1 = ");   //print af op lcd
```

```
    lcd.print(status[0]);
    delay(1500);                // pauze van 1,5 seconden
    lcd.clear();        //lcd-scherm leegmaken
break ;                 // verlaat de switch-instructie
```

We doen dit viermaal voor elke knop die we gebruiken. Als laatste zetten we het ontvangen resultaat terug gelijk aan 0 en resetten de ontvanger.(*Arduino Project Hub*, 2022; Ettron, 2019)

```
  key_value = results.value;   // sla de waarde op als key_value
  receiver.resume(); } // reset de ontvanger voor de volgende code
```



Afbeelding 51 lcd HC-SR04

Bron: Eigen werk m.b.v. *LCD Generator*, 2022

Afbeelding 52 seriële monitor echo met alarm

Bron: Eigen werk m.b.v. Arduino IDE

7 Conclusie

Deze thesis is een onderzoek naar de mogelijkheden tot het optimaliseren van het seminarie automatisatie. Studenten die het keuzevak seminarie scheepsbouw, propulsie en automatisatie hebben gekozen, hebben al een bepaalde basiskennis programmeren met Arduino vanuit de lessen elektronica (deel 2). Maar specifieke aandacht voor het maritieme aspect is hierbij beperkt.

Arduino is een toegankelijk en democratisch platform en wordt vaak gebruikt voor beginnersprojecten. Het biedt studenten een gebruiksvriendelijk en visueel programma (Arduino IDE). Hierdoor is dit platform zeer populair. Door de vele gebruikers van Arduino is er ook veel informatie te vinden op het internet. De Arduino UNO is voor de opstellingen in deze thesis krachtig genoeg en beschikt over voldoende pinnen.

De gemaakte opstellingen in deze thesis hebben als doel het seminarie automatisatie te optimaliseren. Deze opstellingen zijn bedoeld om studenten een variatie aan maritieme projecten voor te stellen en de mogelijkheid ermee te experimenteren. De geschreven schetsen zijn een leidraad en kunnen aangepast worden naar eigen keuze of herschreven worden.

Een schip is een complex geheel en bevat tal van elektronische apparaten. Bij niet correct functioneren daarvan kan het verkeerd lopen aan boord van schepen. Dat dit zich regelmatig voordoet is te lezen in de vele technische artikelen over dit onderwerp in de vakpers. Deze kunnen een interessante en realistische bron van inspiratie zijn bij het opstellen van maritieme oefeningen met Arduino.

Bijlage

Bijlage 1: Artikel

Deepwater Horizon alarms were switched off 'to help workers sleep'

Alarms and safety mechanisms on gulf disaster oil rig were disabled, chief technician at Transocean reveals.

Afbeelding 53 Deepwater Horizon
Bron: (Gerald Herber, 2010)

Vital warning systems on the Deepwater Horizon oil rig were switched off at the time of the explosion in order to spare workers being woken by false alarms, a federal investigation has heard.

The revelation that alarm systems on the rig at the centre of the disaster were disabled – and that key safety mechanisms had also consciously been switched off – came in testimony by a chief technician working for Transocean, the drilling company that owned the rig.

Mike Williams, who was in charge of maintaining the rig's electronic systems, was giving evidence to the federal panel in New Orleans that is investigating the cause of the disaster on 20 April, which killed 11 people.

Williams told the hearing today that no alarms went off on the day of the explosion because they had been "inhibited". Sensors monitoring conditions on the rig and in the Macondo oil well beneath it were still working, but the computer had been instructed not to trigger any alarms in case of adverse readings.

Both visual and sound alarms should have gone off in the case of sensors detecting fire or dangerous levels of combustible or toxic gases.

Williams said he discovered that the physical alarm system had been disabled a full year before the disaster. When he asked why, he said he was told that the view from even the most senior Transocean official on the rig had been that "they did not want people woken up at three o'clock in the morning due to false alarms".
(Pilkington, 2010)

Bijlage 2 : Artikel

Fishing Vessel Grounded After Captain Left Wheelhouse While on Autopilot, NTSB Says

A captain's decision to leave the wheelhouse unattended while transiting the St. Marys River on autopilot led to the grounding and sinking of a fishing vessel, the National Transportation Safety Board said Tuesday.

On June 9, 2021, the F/V Sage Catherine Lane was transiting outbound on the St. Marys River when the vessel grounded on the north jetty of the St. Marys Entrance channel, south of Cumberland Island, Georgia. After the Sage Catherine Lane began to flood, the three-person crew abandoned the vessel and were rescued by a nearby Good Samaritan vessel.

The vessel later sank with about 2,300 gallons of fuel, engine oil and hydraulic oil on board. A crewmember sustained a minor injury and the vessel was declared a total loss at $1 million.

The NTSB detailed its finding in Marine Investigation Report 22/14 released Tuesday.

According to the report, the vessel was transiting outbound, the captain set the vessel's autopilot to maintain the vessel's heading out of the inlet. He answered a phone call and left the wheelhouse, but shortly after, the captain felt the vessel turn abruptly to port.

Returning to the wheelhouse, he attempted to turn away from the jetty, but the Sage Catherine Lane struck the jetty and grounded. The vessel broke later apart and sank following a thunderstorm three days after the grounding.

The investigation found that two days before the grounding, the captain was unable to disengage the autopilot and gain control of the helm as the vessel was proceeding into St. Augustine. The captain examined the autopilot system and found problems with the rudder angle indicator and rudder angle sensor at the rudder post and he took actions to correct the issues. That NTSB said that while the repairs worked initially, the vessel's sharp turn to port indicated the system failed and the repairs were not effective.

“Leaving the wheelhouse unattended is imprudent, especially when navigating areas like the St. Marys Entrance, which included a narrow navigation channel, two jetties and vessel traffic,” the report said. “Had the captain stayed in the wheelhouse after engaging the autopilot, he would have been able to respond and take control of the vessel after the autopilot system failed and caused the rudder to turn to port.”

The NTSB determined the probable cause of the grounding of the Sage Catherine Lane was the captain’s decision to leave the wheelhouse unattended as the vessel transited the St. Marys Entrance on autopilot, leaving insufficient time to respond when the autopilot failed and caused the vessel to go off the set course.

“Autopilot use does not relieve the operator of responsibility to conduct a proper navigation watch. Use of autopilot should not be a justification for an operator to leave the wheelhouse or bridge unattended in confined waters,” the report said. “Navigating in channels and harbors requires quicker reaction times due to traffic, currents encountered, and frequent course changes, and more rudder due to slower speeds. Therefore, autopilot use is often discouraged or prohibited in a harbor entrance or narrow channel.”(Schuler, 2022)

Bijlage 3 : Artikel

Marine transportation safety investigation report M18C022

Executive summary

On 24 August 2018, the passenger vessel *Akademik Ioffe* ran aground 78 nautical miles north-northwest of Kugaaruk, Nunavut. The *Akademik Ioffe* was sailing through narrows in a remote area of the Canadian Arctic that was not surveyed to modern or adequate hydrographic standards, and where none of the vessel crew had ever been. The vessel ran aground at a speed of 7.6 knots before the bridge team could take evasive action; team members were not closely monitoring the echo sounders, and the steady decrease of the under-keel water depth went unnoticed for more than 4 minutes, because the echo sounders' low water depth alarms had been turned off.

In his assessment of the occurrence voyage plan, the master relied on a Canadian chart that contained incomplete bathymetric data.Footnote1 Because the chart indicated spot soundings that showed localized sufficient water depths, and because the chart did not show any shoals or other navigational hazards, the bridge team of the *Akademik Ioffe* considered that the narrows were safe to transit, and consequently did not implement any additional precautions. Following the grounding, the Canadian Coast Guard vessels *Pierre Radisson* and *Amundsen* were tasked to assist, and five aircraft were dispatched by the Canadian Armed Forces. The vessel self-refloated later that night and, on 25 August 2018, its passengers were evacuated and transferred to the sister passenger vessel *Akademik Sergey Vavilov*. The *Akademik Ioffe* sustained serious damage to its hull: 2 ballast water tanks and 2 fuel oil bunker tanks were breached and took on water. An estimated 80.51 L of the vessel's fuel oil was released in the environment. No injuries were reported.

The investigation determined that if a vessel's crew conducts passage planning and assessment based on incomplete and unreliable navigational data, and without taking mitigating measures, there is an increased risk to the safety of the vessel and its complement. Also, if bridge navigation equipment is not optimally operated and automatic safety features such as alarms are turned off, there is a risk that a bridge team will miss

critical information, especially in situations where the prevailing navigating conditions create a high workload for bridge team members. Moreover, if the bridge team composition is inadequate during periods of high workload, such as when transiting confined waters, there is a risk that critical navigational parameters, such as the under-keel water depth, will not be properly monitored, compromising vessel safety.

The TSB investigation into this occurrence revealed safety deficiencies that led the Board to issue a safety recommendation. (Government of Canada, 2021)

Bibliografie

Agarwal, T. (2019, september 5). *MPU6050: Pin Diagram, Circuit Working, Specifications & Applications*. ElProCus - Electronic Projects for Engineering Students. https://www.elprocus.com/mpu6050-pin-diagram-circuit-and-applications/

Aimal Khan. (2021, oktober 25). *10 Best Microcontroller Boards for Engineers and Geeks*. https://www.engineeringpassion.com/10-best-microcontroller-boards-for-engineers-and-geeks/

Almeida, R., & Konrad, J. (2011, april 27). *USCG Deepwater Horizon Investigation Report REVIEW*. GCaptain. https://gcaptain.com/uscg-deepwater-horizon-investigation/

Arduino Documentation. (2015, juli 28). https://docs.arduino.cc/built-in-examples/digital/Button

Arduino Project Hub. (2022). https://create.arduino.cc/projecthub

Arduino Reference. (2020, augustus 19). https://www.arduino.cc/reference/en/language/variables/data-types/long/

Arduino Reference. (2022). https://www.arduino.cc/reference/en/

Arduino.cc. (2021). https://www.arduino.cc/

ArduinoGetStarted.com. (2022). *Arduino—Button Toggle LED | Arduino Tutorial*. https://arduinogetstarted.com/tutorials/arduino-button-toggle-led

Arm Ltd, A. (2022). *What is Instruction Set Architecture (ISA)?* Arm | The Architecture for the Digital World. https://www.arm.com/glossary/isa

Benne. (2020, augustus 23). IR Remote and Receiver with Arduino Tutorial (4 Examples). *Makerguides.Com*. https://www.makerguides.com/ir-receiver-remote-arduino-tutorial/

Bichon. (2021, oktober 29). *Projecthub led*. https://projecthub.arduino.cc/bichon5962/bddff573-c003-4835-8879-880ee4e219a2

Bueken, P., Geerts, T., & Reynaerts, C. (2019). *A brief introduction to (Arduino) programming*.

Bueken, P., Reynaerts, C., & Geerts, T. (2020). *Electronics—Programming Arduino*.

Charleston Security Systems. (2015, november 4). *Security System Quick Tip: How To Bypass A Zone On Your Alarm System*. Charleston Security Systems. https://charlestonsecuritysystems.net/security-tips/security-system-quick-tip-how-to-bypass-a-zone-on-your-alarm-system/

CompareCamp. (2019, maart 15). Arduino IDE Review: Pricing, Pros, Cons & Features. *CompareCamp.Com*. https://comparecamp.com/arduino-ide-review-pricing-pros-cons-features/

Components101. (2021). https://components101.com/development-boards

Components101. (2022). https://components101.com/

Cox, C. (2020, mei 8). How to Control Servos With the Arduino. *Circuit Basics*. https://www.circuitbasics.com/controlling-servo-motors-with-arduino/

Dejan. (2015, oktober 5). How I2C Communication Works? Arduino and I2C Tutorial. *How To Mechatronics*. https://howtomechatronics.com/tutorials/arduino/how-i2c-communication-works-and-how-to-use-it-with-arduino/

Dejan. (2018, februari 15). How to Control Servo Motors with Arduino—Complete Guide. *How To Mechatronics*. https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/

Delgado, C. (2023, januari 17). *Best Raspberry Pi SD Card in 2023*. PC Guide. https://www.pcguide.com/raspberry-pi/guide/best-sd-card/

Dinesh, S. (2007). *Internal Architecture of 8051*.

Dingemans, B. (2020, januari 26). *C programmeertaal: Wat is het en wat kan je ermee?* ProgrammeerPlaats. https://programmeerplaats.nl/c-programmeertaal/

EG Projects. (2022). *Controlling Servo Motor with Stm32f103 microcontroller using stm32cubemx code configurator by STMicroelectronics and keil uvision 5 ide for cortex m1 series microcontrollers*. Engineers Garage. https://www.engineersgarage.com/interfacing-servo-motor-with-stm32/

Electricaltechnology. (2020). [Electricaltechnology]. https://www.electricaltechnology.org/2020/05/types-of-microcontrollers.html#on-the-basis-of-bus-width

Electronicwings. (2018). https://www.electronicwings.com/sensors-modules/ultrasonic-module-hc-sr04

Elektronica voor Jou. (2018). *Elektronica Voor Jou*. https://elektronicavoorjou.nl/arduino-programming-language/

Ettron. (2019, januari 13). How to Make Arduino Remote Control Light Switch. *ETTRON*. https://ettron.com/arduino-remote-control-light-switch/

Eve Jones. (2020, augustus 13). *Do You Have the Essential Skills & Personality Traits for Jobs at Sea?* Martide Seafarer Blog. https://www.martide.com/en/blog/seafarers/technology-redefining-seafarers/

Fetick, R. (2022). *MPU6050_light* [C++]. https://github.com/rfetick/MPU6050_light

Gerald Herber. (2010, april 21). *AP Images*. AP Images. http://www.apimages.com/metadata/Index/Gulf-Oil-Spill-Trial/e0feaddb2e604d858535807c0ee1973f

Gharge, P. (2022, januari 14). *What Is a Microcontroller? – Simply Explained*. All3DP. https://all3dp.com/2/what-is-a-microcontroller/

Government of Canada, T. S. B. of C. (2021, mei 19). *Marine transportation safety investigation report M18C0225—Transportation Safety Board of Canada*. https://www.tsb.gc.ca/eng/rapports-reports/marine/2018/m18c0225/m18c0225.html

Harshil, P. (2022). *MQ-5 Combustible Gas Sensor Interfacing with Arduino*. https://circuitdigest.com/microcontroller-projects/interfacing-mq5-gas-sensor-with-arduino

HZS. (2021). *Studiegids HZS*.

IMO. (2018). *International Convention on Standards of Training, Certification and Watchkeeping for Seafarers (STCW)*. https://www.imo.org/en/About/Conventions/Pages/International-Convention-on-Standards-of-Training,-Certification-and-Watchkeeping-for-Seafarers-(STCW).aspx

IMO. (2019). *Conventions*. https://www.imo.org/en/About/Conventions/

In-Depth: Interface MPU6050 Accelerometer & Gyroscope Sensor with Arduino. (2020, december 29). Last Minute Engineers. https://lastminuteengineers.com/mpu6050-accel-gyro-arduino-tutorial/

Infinity learn. (2022, april 6). Chemistry. *Infinity Learn*. https://infinitylearn.com/surge/blog/neet/important-topic-of-chemistry-bakelite/

International Maritime Organization. (2014). *Model course 7.03* (2014de dr.). IMO.

invento. (2023). *MQ5 Gas Sensor Module for LPG Smoke/CO/Methane Detector Module for DIY*. https://inventosales.com/MQ5-Gas-Sensor-Module-for-LPG-Smoke-CO-Methane-Detector-Module-for-DIY

ISTE. (2022). https://www.iste.org/standards/seal-of-alignment/tinkercad

Jabbaar, A. A. (2019, september 17). *Ultrasonic Sensor HC-SR04 with Arduino Tutorial*. Arduino Project Hub. https://create.arduino.cc/projecthub/abdularbi17/ultrasonic-sensor-hc-sr04-with-arduino-tutorial-327ff6

James, M. (2015, juli 21). Use tone() with Arduino for an Easy Way to Make Noise. *Programming Electronics Academy*. https://www.programmingelectronics.com/an-easy-way-to-make-noise-with-arduino-using-tone/

Javaid, A. (2022a). *Arduino Uno Pinout Guide*. https://linuxhint.com/arduino-uno-pinout-guide/

Javaid, A. (2022b). *How to Interface LCD with Arduino in 4-bit and 8-bit Modes*. https://linuxhint.com/interface-lcd-4-bit-8-bit-modes-arduino/

Jf-parede. (2022). https://nl.jf-parede.pt/microcontrollers-types-their-applications

Kashif. (2022). *How to Use Multiple I2C devices with Arduino*. https://linuxhint.com/use-multiple-i2c-devices-arduino/

Lancaster, D. (2021, januari 8). Do push buttons need resistors? *Electronic Guidebook*. https://electronicguidebook.com/do-push-buttons-need-resistors/

Lastminuteengineers. (2018, augustus 14). Last Minute Engineers. https://lastminuteengineers.com/mq2-gas-senser-arduino-tutorial/

LastMinuteEngineers. (2020, november 8). Last Minute Engineers. https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/

LCD Generator. (2022). http://avtanski.net/projects/lcd/

leveldevelopments. (2023). *MEMS Accelerometer*. Level Developments. https://www.leveldevelopments.com/wp/wp-content/uploads/MEMS-Accelerometer.png

Maker.io. (2022, april 4). *Digikey*. https://www.digikey.be/en/maker/blogs/2022/getting-started-with-tinkercad-circuits

Makers store. (2022). https://mstore.ibda3vision.com/index.php?route=product/product&product_id=1671

Martide blog. (2019, mei 9). *Should Seafarers Learn To Code?. When viewed through a traditional lens… | by Martide Blog | Martide | Medium*. https://medium.com/martide-com/should-seafarers-learn-to-code-f460b82b8505

Oscarromero. (2017). *Pull-Down weerstand—Arduino*. https://oscarromero-arduino.weebly.com/pull-down-weerstand.html

Osoyoo. (2018, november 15). *Basic Lesson – MQ-5 Gas Sensor « osoyoo.com*. https://osoyoo.com/2018/11/15/arduino-lesson-mq-5-gas-sensor/

Pattabiraman, K. (2017, mei 29). How to Set Up an IR Remote and Receiver on an Arduino. *Circuit Basics*. https://www.circuitbasics.com/arduino-ir-remote-receiver-tutorial/

Pilkington, E. (2010, juli 23). Deepwater Horizon alarms were switched off 'to help workers sleep'. *The Guardian*. https://www.theguardian.com/environment/2010/jul/23/deepwater-horizon-oil-rig-alarms

Raspberry Pi, L. (2022). *Buy a Raspberry Pi 4 Model B*. Raspberry Pi. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

Rubashka, V. (2022, december 21). *MCS Electronics*. https://www.mcselec.com/index.php?option=com_content&task=view&id=223

Rucksikaa, R. (2021, februari 16). *Projecthub mq*. https://projecthub.arduino.cc/RucksikaaR/036f01a8-c56e-48b7-96b4-b327d4936fa0

Schuler, M. (2022, mei 10). *Fishing Vessel Grounded After Captain Left Wheelhouse While on Autopilot, NTSB Says*. GCaptain. https://gcaptain.com/fishing-vessel-grounded-captain-left-wheelhouse-autopilot-ntsb/

Sitompul, D., & Sihombing, P. (2022). The LCD Interfacing and Programming. In *Liquid Crystals*. IntechOpen. https://doi.org/10.5772/intechopen.102408

Söderby, K. (2022, december 15). *Liquid Crystal Displays (LCD) with Arduino | Arduino Documentation*. https://docs.arduino.cc/learn/electronics/lcd-displays

Sparkfun. (2018). https://learn.sparkfun.com/tutorials/i2c/all

Sprout Robotic Solutions. (2019). https://onlinesrs.co/product/ultrasonic-wave-detector-ranging-module-hc-sr04-hc-sr04-hcsr04-distance-sensor/

Technology, E. (2021, mei 22). *What is Microcontroller? Types of Microcontrollers & Applications*. ELECTRICAL TECHNOLOGY. https://www.electricaltechnology.org/2020/05/types-of-microcontrollers.html

Teja, R. (2021, januari 30). Different Types of Memory on Arduino | Flash, EEPROM, SRAM. *Electronics Hub*. https://www.electronicshub.org/types-of-memory-on-arduino/

Tesca Technologies Ltd. (2021, april 29). https://www.tescaglobal.com/blog/what-is-a-microcontrollers-and-how-does-it-work/

Thinkology. (2022). *Silicon Sensing | MEMS Accelerometers*. https://www.siliconsensing.com/technology/mems-accelerometers/

Thornton, S. (2018). *RISC vs. CISC Architectures: Which one is better?* https://www.microcontrollertips.com/risc-vs-cisc-architectures-one-better/

Tinkercad. (2022). Tinkercad. https://www.tinkercad.com/dashboard

Tinkercad-blog: Official Guide to Tinkercad Circuits. (2021, augustus 5). Tinkercad. https://www.tinkercad.com/blog/official-guide-to-tinkercad-circuits

Tutorialspoint. (2022). https://www.tutorialspoint.com/arduino/arduino_program_structure.htm

Vasudhendra Badami. (2016, oktober 21). A tour of the Arduino UNO board. *HackerEarth Blog*. https://www.hackerearth.com/blog/developers/a-tour-of-the-arduino-uno-board/