Robotplatform

30-5-2013
Xios Hogeschool Limburg
Stef Wynants & Nick Goyens

INHOUDSOPGAVE

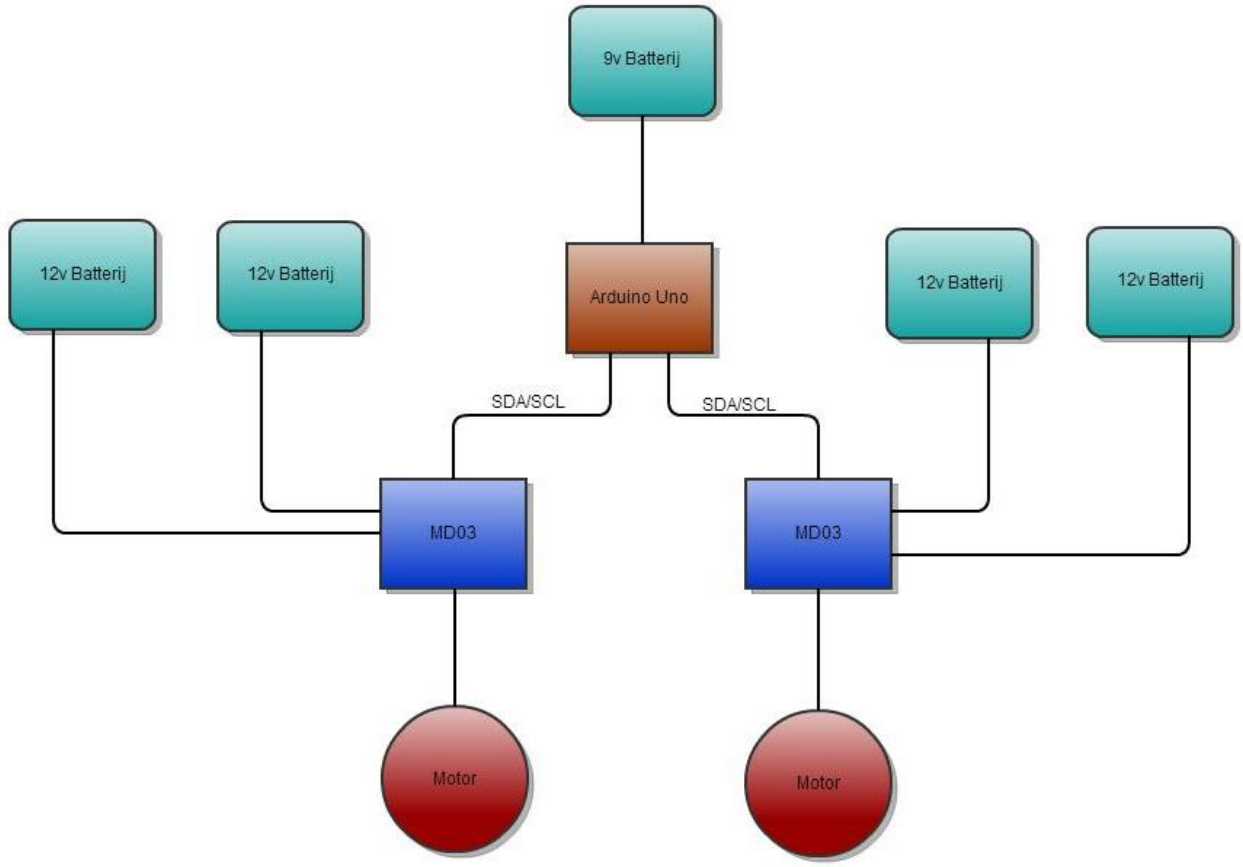
1. INLEIDING

Voor ons project in het tweede en derde trimester van ons tweede jaar Electronica-ICT op de Xios Hogeschool Limburg, hebben wij het 'Robotplatform' als opdracht gekregen. Dit is een platform met 2 elektrische motoren dat wordt aangestuurd door een Arduino Uno en een stuurschakeling.

De motoren worden gevoed door 4 12v batterijen. Er was eventueel ook een optie om deze besturing draadloos te doen met een Android applicatie, de gebruikte draadloze technologie zou dan Bluetooth zijn. Een andere optie is het aansturen van het robotplatform door middel van een joystick of gewoon door een vooraf ingevoerd Arduino programma.

Verder is het ook belangrijk dat de batterijen van de motoren kunnen opgeladen worden door middel van 4 opladers. De sturing van de motor wordt verwezenlijkt door middel van 2 schakelingen genaamd de MD03. Dit alles zal geplaatst worden op de het robotplatform in een behuizing met de bedoeling een compact, rijdend geheel te vormen.

2. BLOKSCHEMA



Bovenaan staat het algemeen blokschema van ons project in een zeer simpele vorm, met als doel de opstelling beter te begrijpen, over de details wordt later gesproken.

De Arduino Uno staat centraal in onze opstelling en wordt via USB geprogrammeerd in het bijgeleverde programma van Arduino. Dit programma kan een standaard programma of kan extern bediend worden door een joystick of door een draadloze Bluetooth sturing. Deze Arduino wordt gevoed door een 9V batterij pack, deze bestaat uit 6x1.5V AA batterijen en zal tevens ook gebruikt worden als 'Pull Up' voor I^2C, dit na omvorming naar 5V door middel van een kleine spanningsomvormer op de print. Het programma in de Arduino zal zoals eerder vermeldt gebruik maken van I^2C. De sturing van de motor wordt over de SDA en SCL lijnen doorgestuurd naar de MD03, gebruik makend van de juiste registers.

De MD03 zal deze data vervolgens ontvangen en doorsturen naar de motoren. De sturing van de MD03's moet gevoed worden door een 5V spanning. Deze komt net zoals bij de 'Pull Up' van de spanningsomvormer.

De 12V batterijen worden per 2 in serie aangesloten op elke MD03 om een totale spanning van 24V te creëren, dit is de spanning die nodig is voor de motoren.

De batterijen kunnen echter in deze serie opstelling niet opgeladen worden door onze opladers. Hierdoor voorzien we een relaisschakeling die de batterijen uit serie schakelt. Als dit is gebeurd kunnen de batterijen veilig opgeladen worden.

Deze volledige opstelling wordt netjes ingebouwd in de robot. De Arduino, MD03's, schakelprint en 9V battery pack wordt ingebouwd in een doosje en vastgemaakt onder het platform, voorzien van de nodige schakelaars om de arduino in en uit te schakelen en LED's om dit aan te geven. De zware 12V batterijen worden ingebouwd in de zwarte behuizing achter de motoren(zie afb. 1). Met schakelaars om de motoren uit serie te schakelen en een noodstop. Ook zijn er hierin aansluitingen voorzien voor de opladers.

3. TECHNOLOGIEËN

Voordat we de werking van het robotplatform meer concreet kunnen uitleggen zullen we eerst de gebruikte technologieën beperkt in het algemeen uitleggen zodat de uiteindelijke opstelling duidelijk blijft.

3.1 I²C

Philips ontwikkelde de I²C bus meer dan 20 jaar geleden en is intussen bij zeer veel toepassingen in gebruik genomen. Het is in synchrone, seriële bus ontwikkelt voor datacommunicatie tussen een microprocessor en andere randapparatuur.

In het fysieke ontwerp van de I2C bus heeft Philips een datalijn SDA en een kloklijn SCL gedefinieerd. Deze twee lijnen creëren een synchrone seriële bus. Het aantal aan te sluiten apparaten op de bus is slechts gelimiteerd door de maximaal toegelaten capaciteit op de bus van 400 pF. Omdat de meeste ICs met een I²Cinteface gebruik maken van low-power CMOS technologie met een hoge ingangsweerstand kunnen veel ICs worden verbonden met de I²C bus voor de maximale capaciteit is bereikt. Het kloksignaal op de bus wordt gegenereerd door één van de masters die met de I²C bus is verbonden. Als er geen communicatie plaatsvindt worden de SDA en SCL lijnen met een pull-up weerstand naar de voedingsspanning getrokken.

Alleen masters kunnen dataoverdracht initiëren op de I²C bus. Wanneer een master op de I²C bus wenst te communiceren met een slave moet het zich eerst de controle over de bus toe-eigenen. Dit is alleen mogelijk wanneer de bus idle is, dat wil zeggen dat zowel de SDA als de SCL lijn hoog zijn. De master creëert een *START conditie* om andere apparaten op de bus te melden dat het de bus overneemt. Om een START conditie te genereren blijft de kloklijn SCL hoog, terwijl de master de SDA lijn omlaag brengt. Dit is een unieke situatie.

Tijdens een normale dataoverdracht zal de SDA datalijn alleen van toestand wijzigen wanneer de SCL kloklijn laag is. Elk apparaat op de I²C bus weet dat een nieuwe communicatie sessie start wanneer de datalijn naar laag gaat terwijl de kloklijn nog hoog is. Hiermee is hersynchronisatie mogelijk wanneer er fouten zijn opgetreden in de vorige dataoverdracht en apparaten de synchronisatie zijn kwijtgeraakt.

Het einde van een communicatiesessie wordt aangegeven met een *STOP conditie*. De STOP conditie wordt gegenereerd door de SDA datalijn hoog te maken terwijl de kloklijn SCL ook hoog is. Net als bij de START conditie kan deze verandering van de SDA lijn niet plaatsvinden tijdens een normale dataoverdracht en aangesloten ICs kunnen hun interne communicatielogica zo resetten wanneer de vorige transfer in een fout is geëindigd. Nadat de STOP conditie is gegenereerd wordt de I²C bus geacht vrij te zijn voor een nieuwe communicatiesessie wanneer een gedefinieerde tijd is verstreken. Deze tijd is afhankelijk van de bussnelheid.

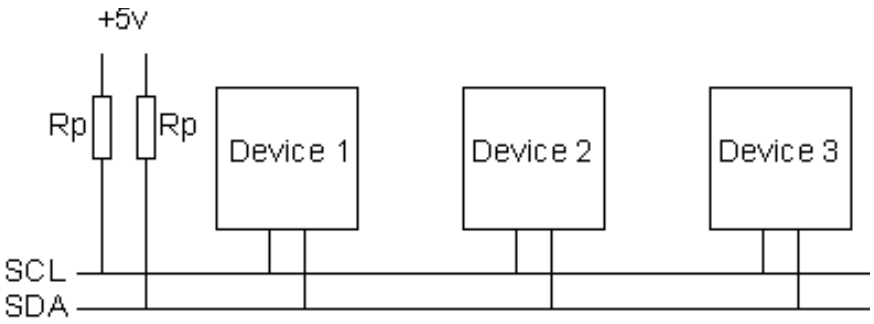
Meerdere herhaalde START condities kunnen voorkomen wanneer een START conditie wordt gegenereerd zonder een STOP conditie die de voorgaande transfer sessie beëindigd. Na een herhaalde START conditie blijft de bus bezet en daarom kan deze situatie in het algemeen worden gezien als een normale START conditie.

DATAOVERDRACHT

Tussen de START en STOP condities vindt de dataoverdracht plaats. De eenheid van de datatransfer op deI²C bus is de byte. Data wordt van de master naar de slave overgedragen, of terug naar de master in bytes. Het eerste byte in elke dataoverdracht is het adres van de slave. Slechts zeven van de acht bits in het adresbyte worden gebruikt om het slave adres de definiëren. Het laagste orde bit geeft aan of het een lees of schrijfopdracht is. Een laag bit geeft een schrijfopdracht aan, terwijl een hoog bit een leesverzoek inhoudt. Bij de dataoverdracht wordt het meest significante bit het eerst verstuurd.

Tijdens de dataoverdracht wijzigt de master de toestand van de kloklijn SCL periodiek van hoog naar laag en terug. Data op de SDA lijn mag alleen worden gewijzigd wanneer de SCL lijn in de laag toestand is. Nadat de SDA lijn is gewijzigd naar de gewenste bit waarde wordt de SCL lijn weer hoog gemaakt om aan de andereICs op de I²C bus te melden dat er een geldig data bit aanwezig is op de lijn. ICs op de I²C bus hebben vaak een zeer gelimiteerd geheugen, logica en snelheid. Daarom moet elke verzonden byte worden bevestigd door de ontvanger. Wanneer een master een byte heeft verzonden zendt het een negende klokpuls waar tijdens de bevestiging plaats zou moeten vinden. Zoals al eerder gezegd wordt de hoog status van de SDA lijn verzorgd door een pull-up weerstand. De slave moet de SDA lijn omlaag trekken tijdens de negende klokpuls. Wanneer dit niet gebeurt en in plaats daarvan blijft de SDA lijn hoog, weet de master dat een fout is opgetreden tijdens de dataoverdracht.

Wanneer een slave bezig is met een opdracht, of eenvoudigweg zeer langzaam in zijn functioneren en het kan niet een volgende byte ontvangen voor de vorige is verwerkt, dan kan de periode na het negende bevestigingsbit worden gebruikt om de master te laten wachten. Opnieuw is hier de logica om de master af te remmen heel eenvoudig. Wanneer de master een nieuw byte wenst te verzenden moet de master de SCL lijn naar de laag toestand brengen. De master wijzigt daarna de SCL lijn naar zijn hoog toestand door zijn uitgang in hoge impedantie te schakelen. Normaal zal de pull-up weestand de toestand van de SCL lijn naar hoog brengen, maar wanneer de slave de communicatie wil vertragen trekt het eenvoudigweg de SCL lijn naar de lage toestand voor de periode die het nodig heeft om zijn voorgaande activiteiten te beëindigen. De master kan alleen de dataoverdracht continueren wanneer het detecteert dat de SCL lijn weer in de hoge toestand is. Hiermee is een zeer eenvoudige en effectieve methode gecreëerd om langzame slaves met gelimiteerde verwerkings- en geheugencapaciteit te laten communiceren met snellere masters. Wanneer alle bytes zijn ontvangen verwerkt wordt de STOP conditie door de master gegenereerd om de bus vrij te geven voor andere communicatie.



Wij zullen het I²C protocol gebruiken voor de communicatie tussen de Arduino Uno microcontroller en de MD03 motosturingen. De aansluitingen op de Arduino voor SDA en SCL zijn A4 en A5, ook de MD03 heeft 2 toegewezen aansluitingen voor I²C.

3.2 Bluetooth

Bluetooth wireless technology is een draadloos communicatie systeem bedoeld om de kabels te vervangen die verschillende types apparaten met elkaar verbinden. Bluetooth wireless technology wordt gebruikt voor spraak en data verbindingen in een breed scala van producten, van mobiele telefoons, tablets, headsets, hartslag monitoren, autotechniek tot medische apparatuur en computers. Kenmerkend voor de techniek is de mogelijkheid tot een hoge datasnelheid en een laag energieverbruik.

Bluetooth versie 4 is de meest recente versie van de bluetooth technology. Op 7 juli 2010 werden de specificaties van deze standaard vastgelegd, waarbij er vooral aan de energiezuinigheid werd gewerkt. Vanaf deze versie zijn bluetoothaccessoires die werken op een knoopcel mogelijk.

Bluetooth is een radioverbinding (in de 2,45 GHz band, (UHF) voor spraak en data op korte afstand. Het werkt 'point to multipoint', wat inhoudt dat een enkele bron meer 'ontvangers' kan bedienen. Wanneer twee Bluetooth apparaten een verbinding hebben opgebouwd, dan ontstaat een zogenoemd piconet. Er kunnen op dezelfde plek meerdere van dergelijke piconets naast elkaar bestaan, in wat men een scatternet noemt. Binnen een piconet ondersteunt Bluetooth maximaal acht actieve verschillende apparaten, terwijl er in totaal 127 apparaten een verbinding kunnen houden (deze zijn dan tijdelijk 'geparkeerd').

Normaal gesproken zal het binnen een straal van 1 tot 10 meter functioneren, maar wanneer het zendvermogen wordt opgevoerd, kan zelfs de 100 meter worden gehaald. Een zogenoemde 'zichtverbinding' (elkaar kunnen zien) is niet nodig; dankzij de GHz-radioverbinding dringt het Bluetooth signaal ook door vaste materialen (zolang het geen metaal is).
De communicatie van digitale spraak behoort tot de standaardmogelijkheden van Bluetooth. Bluetooth ondersteunt binnen een piconet tot drie gelijktijdige full-duplex-gesprekken.

Omdat Bluetooth een vervanger is voor de (korte) kabels, kan het worden gebruikt om allerlei apparaten met elkaar te laten communiceren. De ontwerpers hebben met opzet gebruik gemaakt van een goedkope radiotechniek, zodat Bluetooth zonder veel bezwaar in ieder apparaat kan worden ingebouwd. Omdat Bluetooth normaal gesproken ook weinig stroom verbruikt (30 µA) in 'hold mode' en 8-30 milliampère bij een actieve verbinding, kan het ook worden toegepast in mobiele apparaten die afhankelijk zijn van batterijen. Vanaf versie 1.2 gebruikt Bluetooth frequency hopping, of Adaptive Frequency Hopping (AFH).

Bluetooth apparatuur is verdeeld in 3 verschillende klassen:

- Class 1: Ontworpen voor lange afstandverbindingen (tot ~100m)
- Class 2: Voor normaal gebruik (tot ~10m)
- Class 3: Voor korte afstanden (10 cm - 1 m)

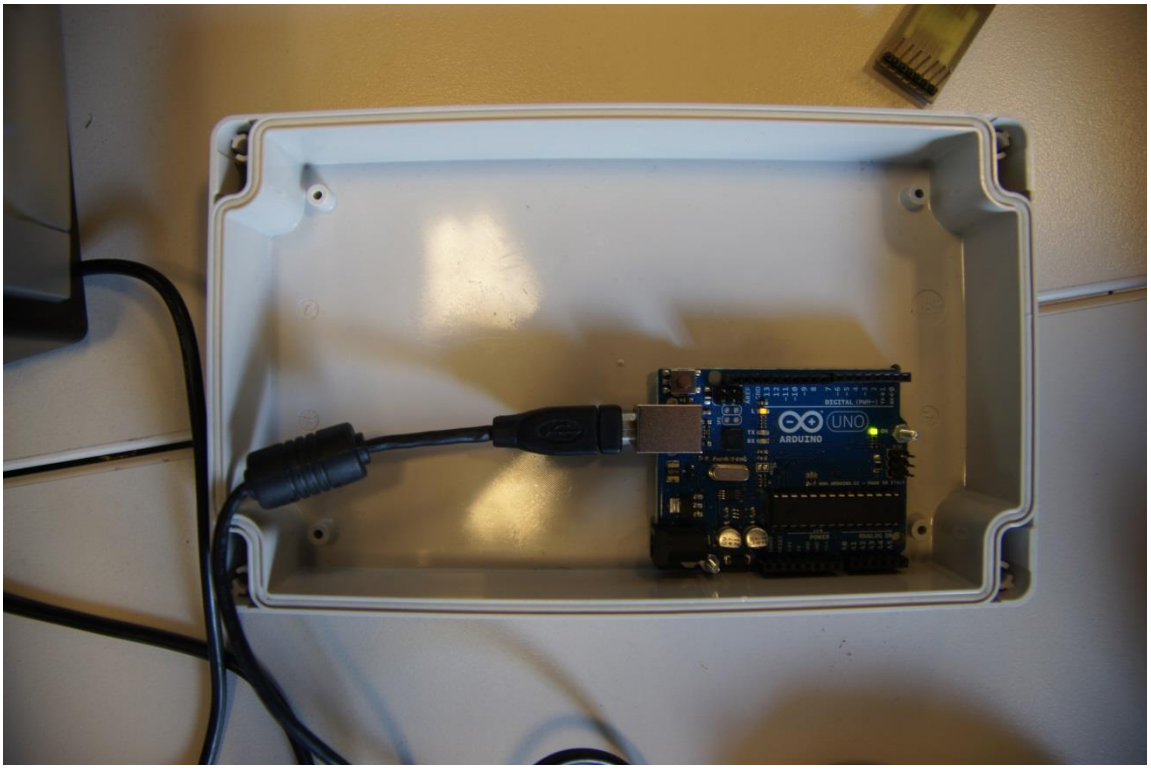
In onze huidige versie van het project zijn we er nog niet in geslaagd om Bluetooth te integreren. Wij zien dit echter als een uitstekende uitbreidingsmogelijkheid in de toekomst om het de robot vanop afstand te kunnen besturen.

4. HARDWARE

We zullen nu de gebruikte hardware meer specifiek bespreken en hoe we deze hebben toegepast in ons project

4.1 ARDUINO UNO

De Arduino Uno is een microcontroller board gebaseerd op de ATmega328. De Uno heeft 14 digitale input/output pinnen aan boord en 6 analoge inputs. De analoge inputs zijn zeer cruciaal voor ons project aangezien we 2 van deze pinnen gebruiken om de analoge waardes van de joystick in te lezen. Deze analoge waardes van de joystick worden ingelezen op deze pinnen en vervolgens omgezet naar een waard van 0-1024, deze waardes gebruiken we verder in ons programma. De A4 en A5 pinnen gebruiken we voor de SDA en SCL lijnen van I^2C. Verder beschikt deze microcontroller over een 16MHz kristal, USB connecter voor aansluiting met de computer en een powerplug voor aansluiting op het 9V battery pack. Dit is een ideale spanning aangezien de Uno werkt op een spanning tussen de 7V en 12V.



We maken ook gebruik van de 5V en de 2 GND pennen. De 5V en 1 GND pen worden gebruikt om spanning te leveren aan de potentiometers van de joystick. Terwijl de tweede GND pen wordt aangesloten aan de gemeenschappelijke ground van alle componenten.

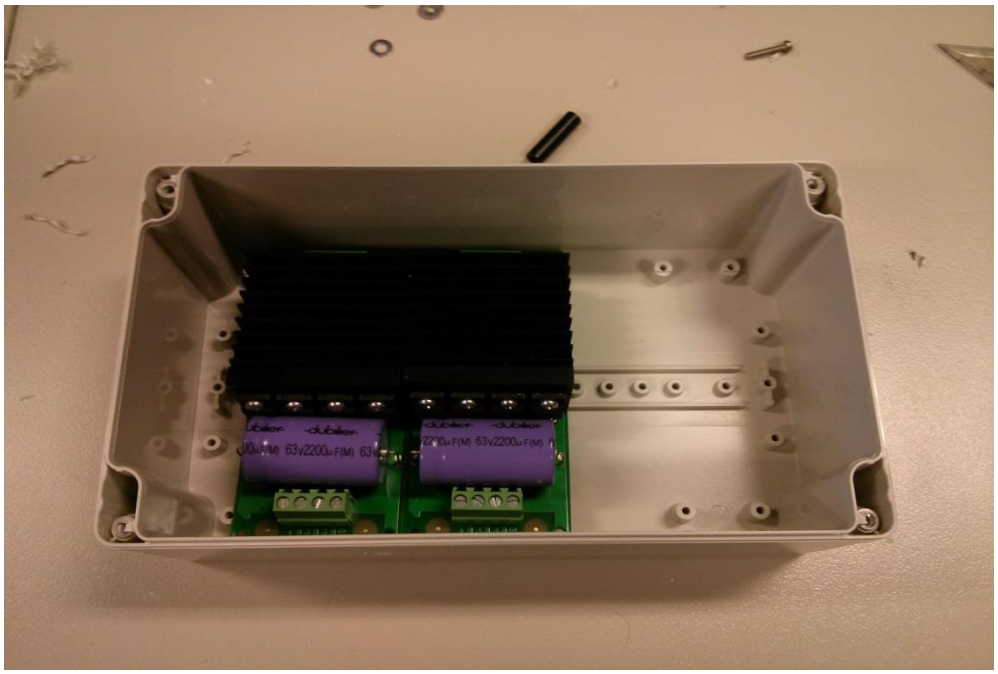
De andere pinnen van de Uno worden momenteel niet door ons gebruikt. Ook het geheugen volstaat meer dan voldoende voor onze benodigdheden.

4.2 MD03

De MD03 is een motordriver met H-burg. De motor snelheid wordt geregeld met PWM op een frequentie van 15kHz. Hij heeft 5 modes waarmee je hem kan besturen.

- I^2C: een maximum van 8 MD03 modules kunnen samen gebruikt worden omdat hij 8 verschillende i^2c adressen heeft die men met de dipswitches kan instellen.
- 0V - 2.5 - 5V analog input.
  0V: full reverse
  2.5V: center stop
  5V: full forward.
- 0V - 5V analoog ingang met aparte richting controle.
- RC mode. Gestuurde richting door de RC ontvanger uitgang.
- PWM. Een simpele geïntegreerde filter zorgt voor een 0%-100% 20khz, of een hogere frequentie, regeling en dit ter vervanging van een analoge regeling.

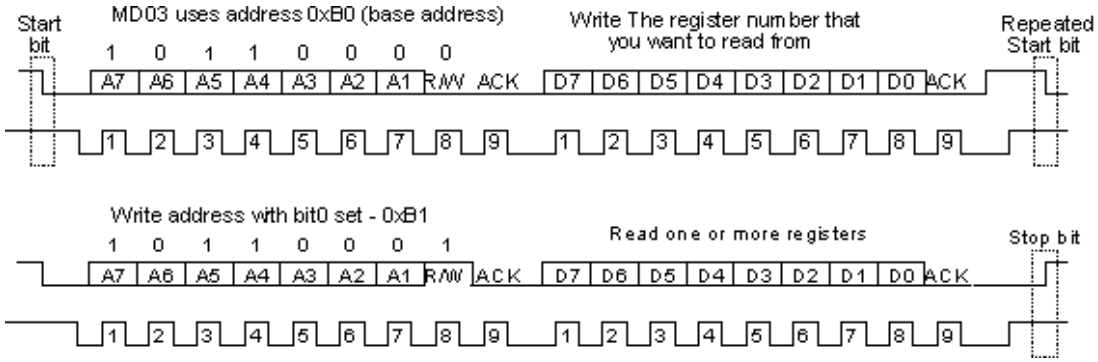
De MD03 heeft 5V DC en een maximum van 50mA nodig om te worden aangestuurd. Hij kan maximaal 24V Dc en 20A leveren aan de Motor.



Als men motor geluid onderdrukking wil doen moet men een 10nF snubbing condensator over de motor zetten. De condensator moet 2 keer de motorspanning aan kunnen.

De Connectie voor I²C is aangeduid op de print.

In I2C-modus kan de MD03 worden aangesloten op populaire controllers zoals de PICAXE, OOPic en BS2p, en een breed scala aan micro-controllers, zoals PIC's, 8051's en H8's.



Het I2C communicatieprotocol van de MD03 module is hetzelfde als populaire EEPROM's zoals de 24C04. Om een register te lezen, moet je eerst zijn adres (0XB0 is het basis - adres) sturen met de lees / schrijf - bit laag. Daarna het register adres dat je wilt lezen/schrijven en dan de waarde die we willen schrijven of waar we de gelezen waarde willen schrijven. Dan stoppen we de transmissie en beginnen terug opnieuw. Hier onder vind je de 8 MD03 registers met hun adres en als we er in kunnen schrijven/lezen.

Registers

Register Adres	Naam	Read / Write	Beschrijving
0	Commando	R / W	Write 01 → Vooruit - 02 →Achteruit (00 → Directe stop)
1	Status	Alleen R	Acceleratie, Temp en huidige status
2	Snelheid	R / W	Snelheid motor (0x00 - 0xFF)
3	Versnelling	R / W	Acceleratie motor (0x00 - 0xFF)
4	Temp	Alleen R	Module temperatuur
5	Motorstroom	Alleen R	Motorstroom
6	Ongebruikt	Alleen R	Nul
7	Software versie	Alleen R	Software versie (9)

Command Register

Dit register zorgt ervoor dat we de motor kunnen starten, stoppen en de draairichting van de motor kunnen veranderen. Indien je de richting van de motor wilt wijzigen of de motor tot stilstand wilt laten komen moet je een andere waarde invoeren en deze overschrijven.

Status Register

Geeft de status van de MD03 weer in dit register kunnen we enkel lezen

Bit 7 (msb)	6	5	4	3	2	1	Bit 0 (lsb)
Busy	-	-	-	-	Over - Temp limiet	Over - current limiet	Acceleratie in progress

Bit 0 : is hoog indien de motor nog accelereert
Bit 1 : is hoog indien de stroom, opgenomen door de motor, 20A bereikt
Bit 2 : is hoog indien de temperatuur te hoog wordt
Bit 7 : is hoog indien er een nieuwe opdracht voor de module is.

Snelheid Register

Hiermee wordt de maximale snelheid ingesteld tot waar hij zal versnellen. Dit is eigenlijk de waarde die naar de PWM controller gaat. Deze moet een waarde van 0 tot 243. Hoe groter de waarde, hoe meer spanning op de motor.

Acceleratie Register

Hiermee wordt de snelheid waarmee de motor versneld of vertraagt ingesteld, dit d.m.v. een waarde van 0 tot 255, hoe groter de waarde, hoe meer tijd de module zal nodig hebben om de nieuwe snelheid te bereiken. Deze waarde controleert een timer die de huidige motorsnelheid in stappen laat stijgen tot de gewenste motorsnelheid. Hoe stel je de gewenste versnelling in? Door volgende berekening: (((versnelling register) * 125) +768) µs.

Een waarde van nul geeft 768µs/step of 243 * 768 = 186624µs (0.187s) te versnellen van 0 tot maximum. Een waarde van 255 is ((255 * 125) +768) * 243 = 7932249µs of net onder de 8 seconden.

Temperatuur Register

Deze waarde wordt gebruikt om inwendig de motor stroom te limiteren wanneer de module te warm wordt. Het is mogelijk deze waarde uitlezen. Hij geeft geen °C uit, wanneer de temperatuur omhoog gaat zal de waarde dalen. Hij leest de doorlaat spanningsval van een diode die onder de 0,003 Ω stroom sensor weerstand ligt. De waarde verlaagt met 1 bij een verhoging van 1,42°C.

Motorstroom Register

De waarde van dit register wordt gebruikt om de inwendig the motorstroom te limiteren tot 20A. Het is mogelijk deze waarde uit te lezen, maar dit is niet noodzakelijk. De waarde is evenredig aan de motorstroom. Bij 20A is de waarde 186.

COMBINATIE VAN DE SWITCHES

Mode	Switch 1	Switch 2	Switch 3	Switch 4
I2C Bus - address 0xB0	On	On	On	On
I2C Bus - address 0xB2	Off	On	On	On
I2C Bus - address 0xB4	On	Off	On	On
I2C Bus - address 0xB6	Off	Off	On	On
I2C Bus - address 0xB8	On	On	Off	On
I2C Bus - address 0xBA	Off	On	Off	On
I2C Bus - address 0xBC	On	Off	Off	On
I2C Bus - address 0xBE	Off	Off	Off	On
0v - 2.5v - 5v Analog	On	On	On	Off
0v - 5v Analog + Direction	Off	On	On	Off
Radio Control	On	Off	On	Off

New Mode (from version 12)

Radio Control, timeout on	Off	Off	On	Off

Alle andere combinaties zijn niet juist, de LED zal knipperen bij een foute combinatie.

4.3 Batterijen

Om de elektrische motoren te voeden maken we gebruik van 4x12V batterijen. 2 van deze batterijen zullen elke 1 motor voeden, zodat we de gewenste spanning krijgen van 24V per motor.

De batterijen in kwestie zijn oplaadbare lood accu's, meer bepaald de Long WP4.5-12. De 4.5 in de naam staat voor de capaciteit van 4.5 Ah. Hieronder wat meer specificaties over de batterij:

Nominal Capacity:

20 hour rate (0.225A to 10.50V) 4.5Ah

10 hour rate (0.428A to 10.50V) 4.275Ah

5 hour rate (0.765A to 10.20V) 3.825Ah

1 C (4.5A to 9.60V) 2.55Ah

3 C (13.5A to 9.60V) 1.8Ah

Weight:

Approx. 1.62kg(3.56Lbs.)

Internal Resistance:

(at 1KHz) Approx. 34.2 mΩ Dimensions mm(inch)

Maximum Discharge Current for:

5 seconds: 67.5A

Charging Methods at 25℃(77℉):

Cycle use:

Charging Voltage 14.4 to 15.0V

Coefficient -5.0mv/℃/cell

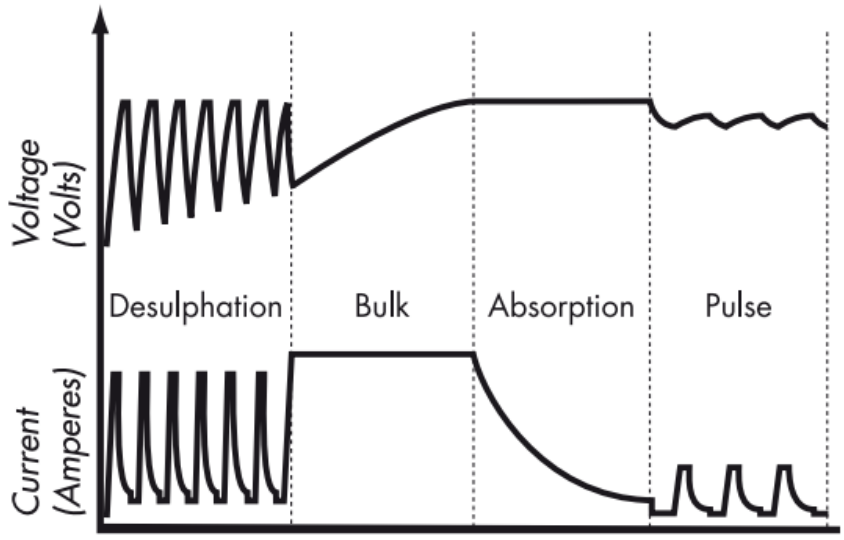
Maximum Charging Current ： 1.35A

Standby use:

Float Charging Voltage 13.50 to 13.80V

4.4 Opladers

Om deze batterijen opnieuw op te laden hebben we Multi XS 3600 opladers ter beschikking. Zoals eerder gezegd moeten we de batterijen uit serie schakelen om deze op te laden, hiervoor maken we gebruikt van een relais schakeling. We hebben ook aansluitingen in de behuizing voorzien om de opladers op aan te sluiten, hierdoor kunnen we veel sneller werken.



Zoals te zien op bovenstaande figuur beschikt deze oplader over 3 fases:

Desulphation: De batterij wordt opgeladen in pulsen, dit gaat de het verlies van sterkte van de batterij tegen.

Bulk: Hier wordt de batterij voor bijna 80% opgeladen. De lader geeft een vaste stroomsterkte tot de volledige spanning is berijkt.

Absorption: Batterij wordt opgeladen tot bijna 100%

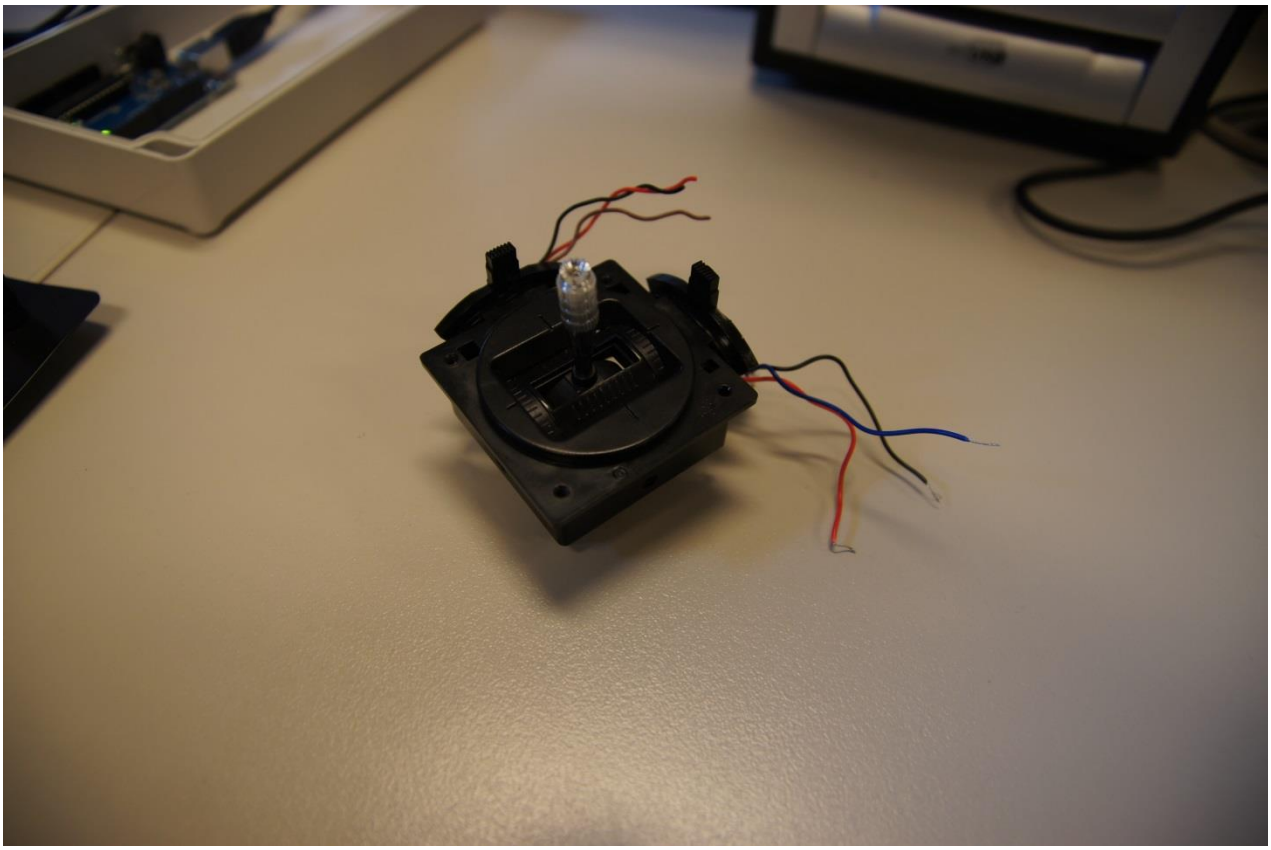
Pulse: Hierbij wordt de batterij blijvend opgeladen als deze niet in gebruik is. Als de spanning zakt wordt er met een puls opgeladen. Hiermee kan de batterij maanden veilig opgeladen worden.

Wij schatten dat het opladen van een volledig lege batterij met deze oplader zo'n 4 tot 5 uur duurt.

4.5 JOYSTICK

Deze Joystick met middenstand is momenteel onze manier om de robot te besturen. Het principe hiervan is zeer simpel, de joystick bestaat uit 1 potentiometer voor elk van de 2 assen. De waarde van de potentiometer zal wijzigen bij het aanpassen van de joystick en deze waarde lezen we op een analoge ingang op de Arduino in.

Deze wordt vervolgens geconverteerd naar een digitale waarde van 0 tot 1024. Deze waarde gebruiken we in ons Arduino programma om de motoren aan te sturen. Dit systeem hebben we getest en geeft ons een goede controle over de robot. De 5V spanning en ground van deze potentiometer zijn ook aangesloten op de Arduino.

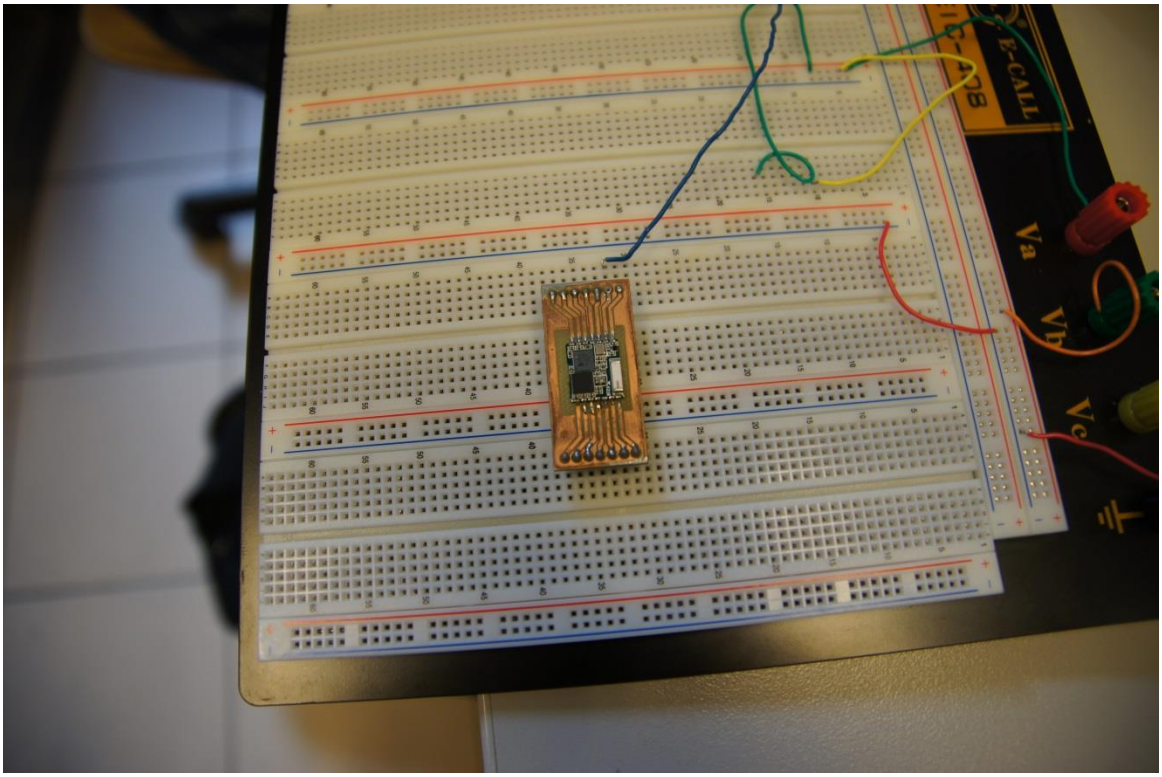


4.6 BT23

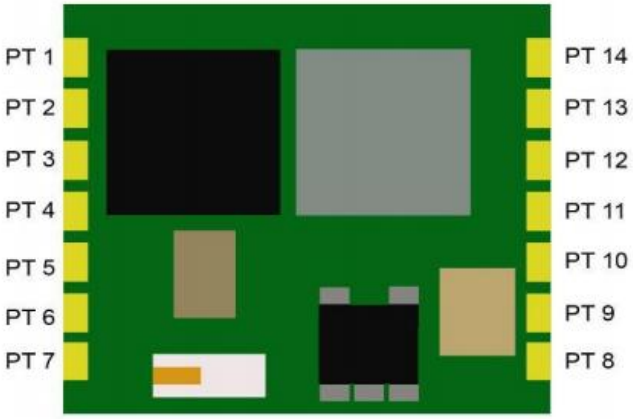
De BT23 is een micro-sized bluetooth module met ingebouwde antenne. De BT23 heeft 6 DAC/ADC IO lijnen, verschillende seriële interface opties en tot 1.5 Mbps data snelheid.

SPECIFICATIES:

- Bluetooth v2.1 Serieel en OBEX profielen
- Class 2 radio
- 128 bit encryptie
- Tot 25m afstand
- ARM Cortex microprocessor 72 MHz
- 256K bytes flash geheugen
- 48K bytes RAM geheugen
- 1.5Mbps data
- UART tot 2 Mbaud
- I2C interface
- 6 I/O
- 4x12-bit A/D input
- 1DAC output
- 1 LPO output
- AT command set



AANSLUITINGEN



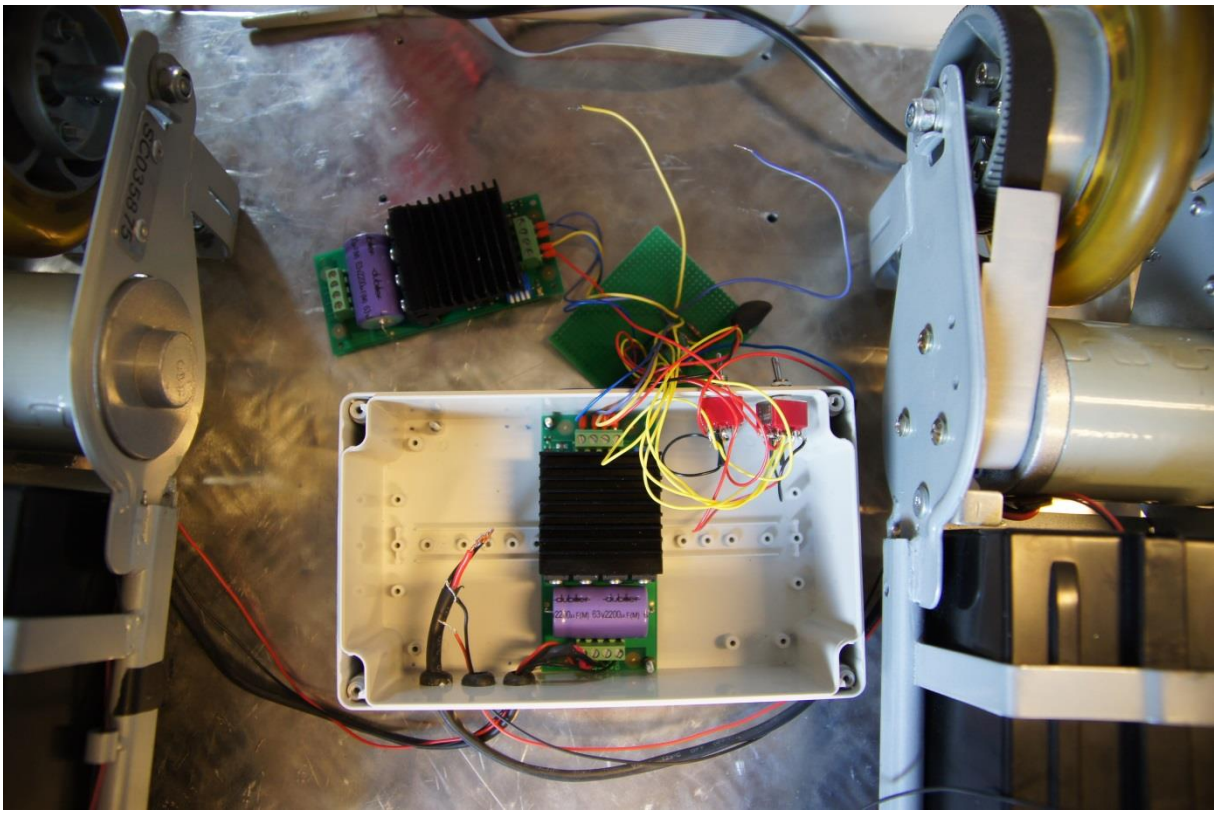
- RXD: PIN 13
- TXD: PIN 14
- Vin: PIN 8
- GND: PIN 7
- Reset: PIN 10

We hebben deze module bestudeerd en proberen deze aan het werken te krijgen, maar uiteindelijk bleek hier te weinig tijd voor te zijn en hebben we ons gefocust op het besturen van de robot met de joystick.

Aangezien dit goed is gelukt voorzien wij het aansturen met bluetooth dus als de volgende belangrijke stap. Wij denken dus dat dit nog steeds een zeer interessante uitbreiding is op ons project, eventueel met besturing van een Android applicatie.

4.7 Elektrisch schema

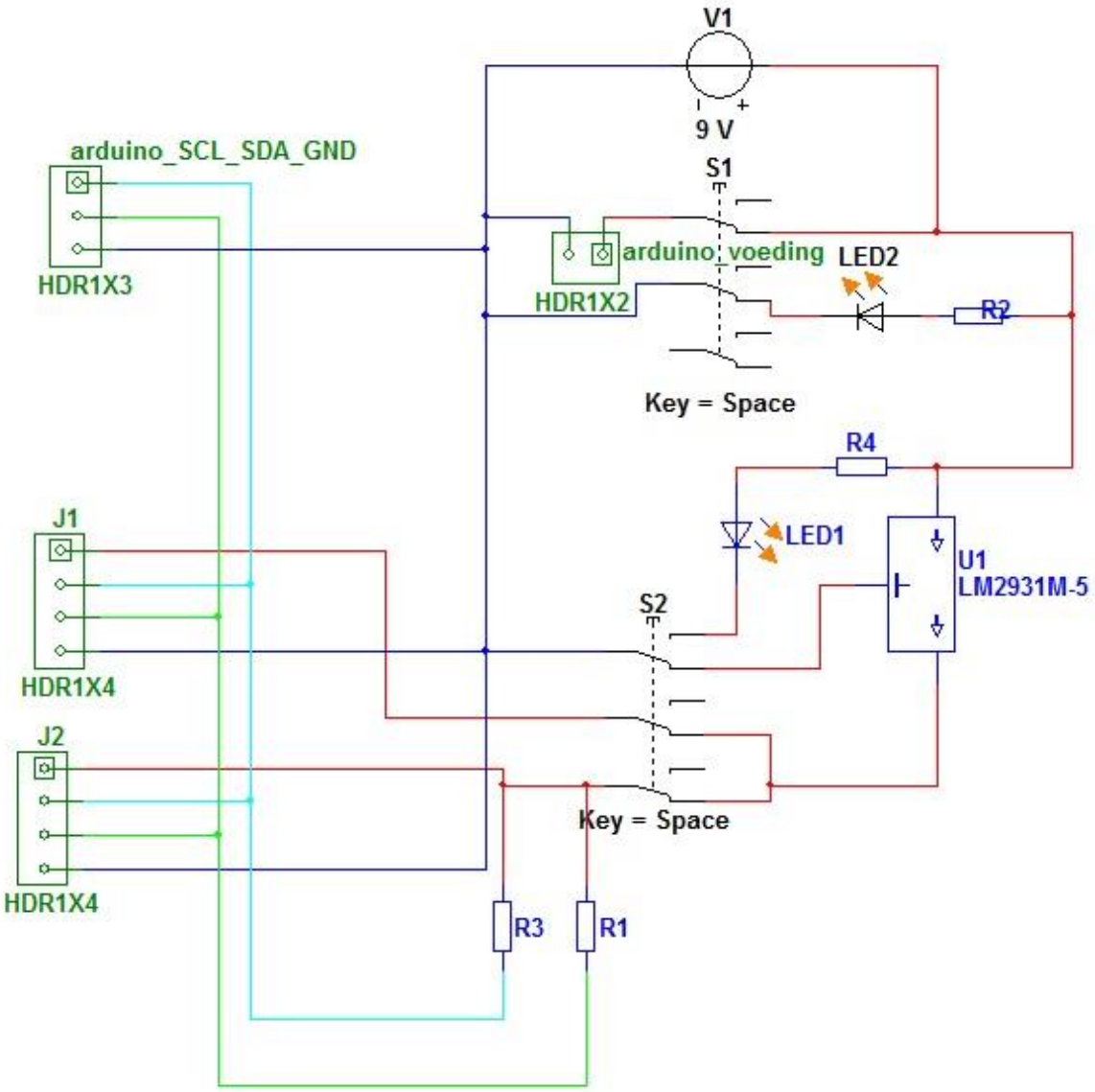
Bijgevoegd is een meer gedetailleerd schema over de opbouw van onze print, aangezien deze niet complex is hebben we besloten deze op te bouwen op gaatjes print. (zie volgende pagina)



Op dit schema is beter te zien hoe we de Arduino voeden met het battery pack en deze ook kunnen in en uit schakelen met een schakelaar S1, er is ook een LED voorzien om dit aan te geven.

Deze spanning wordt vervolgens omgevormd tot 5V en dient voor zowel de voedingspanning van de MD03's als ook de pull up spanning voor de SDA en SCL lijn. Ook dit geheel kan worden uitgeschakeld met schakelaar S2. Als deze is uitgeschakeld zal de tweede LED gaan branden om dit aan te geven.

De ground's van alle componenten zijn ook gemeenschappelijk doorverbonden.


V1
9 V
arduino_SCL_SDA_GND
HDR1X3
S1
arduino_voeding
HDR1X2
LED2
R2
Key = Space
R4
J1
LED1
U1
LM2931M-5
S2
HDR1X4
J2
Key = Space
HDR1X4
R3
R1

5. SOFTWARE

5.1 ARDUINO SOFTWARE

Bijgevoegd is ons software programma geschreven voor de Arduino Uno. Met behulp van dit programma kunnen we de robot in elke richting aansturen met de joystick.

Allereerst definiëren we de verschillende adressen van de MD03 registers en stellen we variabelen in voor de assen van de joystick. Dan wordt in “void setup” i²c en de joystick inputs ingesteld. Daarna in “void loop” wordt de waarde van de joystick opgeslagen in een variabele. Deze variabele gaan we vervolgens telkens in if structuren vergelijken en dan de nodige actie uit voeren.

Zoals in de eerste if structuur ligt de waarde van elke as tussen de 400 en 600, wat wil zeggen dat de joystick in zijn middenstand staat, hierbij zal de motor dus moeten stilstaan. Om dit te doen sturen we simpelweg de juiste command byte naar elke MD03 om een full stop te bekomen. Als de waarde van bijvoorbeeld de voorwaarts/achterwaarts as boven de 600 komt, zullen beide motoren vooruit moeten rijden. Dit doen we door het ‘speed’ en ‘acceleratie’ register van elke MD03 een waarde te geven en met I²C door te sturen en vervolgens de command byte in te stellen. Nu zullen de motoren voorwaarts draaien.

Op deze manier hebben we elke beweging van de joystick geprogrammeerd en zal de motor dus ook naar links en rechts kunnen draaien en ook achteruit rijden.

Ook is het programma zo ingesteld dat de beweging in de hoeken van de joystick anders zal zijn dan in het midden. Zo zal er in het geval dat de joystick zich links vooruit bevindt de robot link vooruit rijden, dit is ook terug te vinden in het programma.

```
#include <Wire.h>
#define MD03_1    0x58
#define MD03_2    0x59
#define CMD_reg   0x00
#define SPEED_reg 0x02
#define ACC_reg   0x03
 int joyFB = A1 ; //joystick forward and backward
 int joyLR = A2 ; //joystick left and right
 int valueFB = 0; //variable value forward backward
 int valueLR = 0; //variable value left right

void setup()
{
  Wire.begin(); // join i2c bus (address optional for master)
  Serial.begin(9600);
  pinMode(joyFB,INPUT);
  pinMode(joyLR,INPUT);
}


void loop()
{

  valueFB = analogRead(joyFB);
  valueLR = analogRead(joyLR);

  if ( valueFB < 600 and valueFB > 400 and valueLR >400 and  valueLR < 600){// staat stil
    Wire.beginTransmission(MD03_1); // transmit to md03 1
    Wire.write(CMD_reg);       // command byte
    Wire.write(0);             // full stop  motor 1
    Wire.endTransmission();    // stop transmitting
    Wire.beginTransmission(MD03_2); // transmit to md03 2
    Wire.write(CMD_reg);       // command byte
    Wire.write(0);             //  full stop motor 2
    Wire.endTransmission();    // stop transmitting

  }
  else if (valueFB > 600 )    {          // rijd voorwaarst

    Wire.beginTransmission(MD03_1);
    Wire.write(SPEED_reg);
    Wire.write(128);
    Wire.endTransmission();
    Wire.beginTransmission(MD03_2);
    Wire.write(SPEED_reg);
    Wire.write(128);
    Wire.endTransmission();

    Wire.beginTransmission(MD03_1);
    Wire.write(ACC_reg);
    Wire.write(27);
    Wire.endTransmission();
    Wire.beginTransmission(MD03_2);
    Wire.write(ACC_reg);
    Wire.write(27);
    Wire.endTransmission();
    Wire.beginTransmission(MD03_1);
```

```
Wire.write(CMD_reg);
Wire.write(1);            // sets motor to forward
Wire.endTransmission();
Wire.beginTransmission(MD03_2);
Wire.write(CMD_reg);
Wire.write(1);            // sets motor to forward
Wire.endTransmission();

 if(valueFB > 600 and valueLR > 600) {                    // rijd links vooruit

    Wire.beginTransmission(MD03_1);
    Wire.write(SPEED_reg);
    Wire.write(128);
    Wire.endTransmission();
    Wire.beginTransmission(MD03_2);
    Wire.write(SPEED_reg);
    Wire.write(64);
    Wire.endTransmission();

    Wire.beginTransmission(MD03_1);
    Wire.write(ACC_reg);
    Wire.write(27);
    Wire.endTransmission();
    Wire.beginTransmission(MD03_2);
    Wire.write(ACC_reg);
    Wire.write(27);
    Wire.endTransmission();
    Wire.beginTransmission(MD03_1);
    Wire.write(CMD_reg);
    Wire.write(1);
    Wire.endTransmission();
    Wire.beginTransmission(MD03_2);
    Wire.write(CMD_reg);
    Wire.write(1);
    Wire.endTransmission();

   }

  else if (valueFB > 600 and valueLR < 400)                  //rijd rechts vooruit
  {
  Wire.beginTransmission(MD03_1);
  Wire.write(SPEED_reg);
  Wire.write(64);
  Wire.endTransmission();
  Wire.beginTransmission(MD03_2);
  Wire.write(SPEED_reg);
  Wire.write(128);
  Wire.endTransmission();

  Wire.beginTransmission(MD03_1);
  Wire.write(ACC_reg);
  Wire.write(27);
  Wire.endTransmission();
  Wire.beginTransmission(MD03_2);
  Wire.write(ACC_reg);
  Wire.write(27);
  Wire.endTransmission();
```

```
    Wire.beginTransmission(MD03_1);
    Wire.write(CMD_reg);
    Wire.write(1);
    Wire.endTransmission();
    Wire.beginTransmission(MD03_2);
    Wire.write(CMD_reg);
    Wire.write(1);
    Wire.endTransmission();
     }

}
  else if (valueFB < 400) {                    //rijd achteruit
    Wire.beginTransmission(MD03_1);
    Wire.write(SPEED_reg);
    Wire.write(128);
    Wire.endTransmission();
    Wire.beginTransmission(MD03_2);
    Wire.write(SPEED_reg);
    Wire.write(128);
    Wire.endTransmission();

    Wire.beginTransmission(MD03_1);
    Wire.write(ACC_reg);
    Wire.write(27);
    Wire.endTransmission();
    Wire.beginTransmission(MD03_2);
    Wire.write(ACC_reg);
    Wire.write(27);
    Wire.endTransmission();
    Wire.beginTransmission(MD03_1);
    Wire.write(CMD_reg);
    Wire.write(2);              // sets motor to backwards
    Wire.endTransmission();
    Wire.beginTransmission(MD03_2);
    Wire.write(CMD_reg);
    Wire.write(2);             // sets motor to backwards
    Wire.endTransmission();

     if( valueFB < 400 and valueLR > 600) {                          // rijd links archteruit
       Wire.beginTransmission(MD03_1);
       Wire.write(SPEED_reg);
       Wire.write(128);
       Wire.endTransmission();
       Wire.beginTransmission(MD03_2);
       Wire.write(SPEED_reg);
       Wire.write(64);
       Wire.endTransmission();

       Wire.beginTransmission(MD03_1);
       Wire.write(ACC_reg);
       Wire.write(27);
       Wire.endTransmission();
       Wire.beginTransmission(MD03_2);
       Wire.write(ACC_reg);
       Wire.write(27);
       Wire.endTransmission();
       Wire.beginTransmission(MD03_1);
```

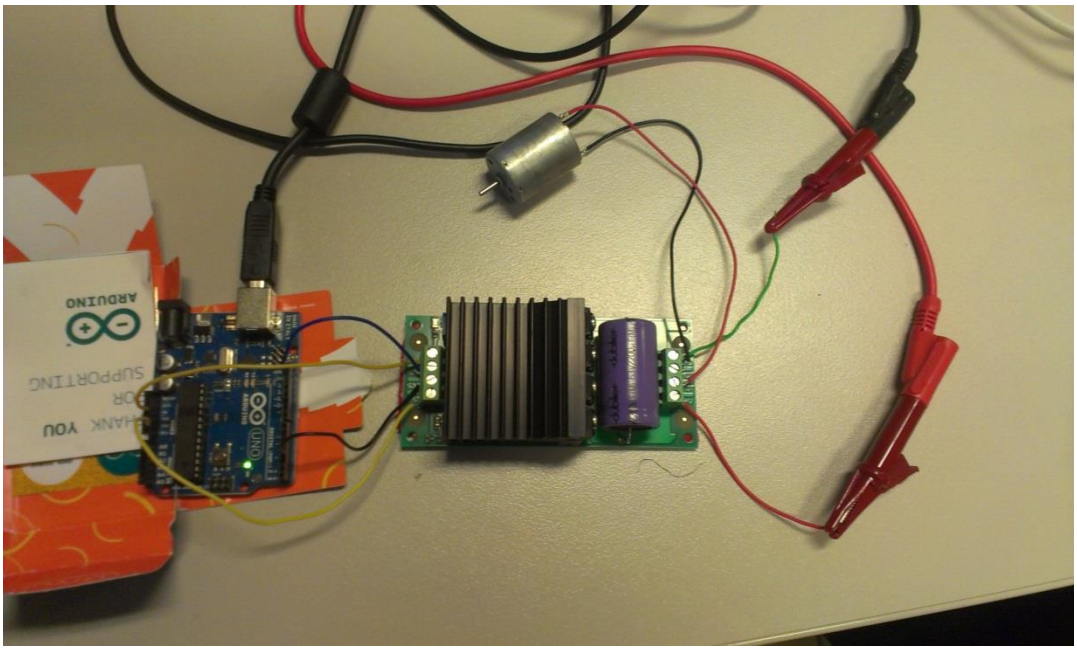
```
      Wire.write(CMD_reg);
      Wire.write(2);
      Wire.endTransmission();
      Wire.beginTransmission(MD03_2);
      Wire.write(CMD_reg);
      Wire.write(2);
      Wire.endTransmission();
     }

     else if ( valueFB < 400 and valueLR < 400)       // rijd rechts achteruit
     {
      Wire.beginTransmission(MD03_1);
      Wire.write(SPEED_reg);
      Wire.write(64);
      Wire.endTransmission();
      Wire.beginTransmission(MD03_2);
      Wire.write(SPEED_reg);
      Wire.write(128);
      Wire.endTransmission();

      Wire.beginTransmission(MD03_1);
      Wire.write(ACC_reg);
      Wire.write(27);
      Wire.endTransmission();
      Wire.beginTransmission(MD03_2);
      Wire.write(ACC_reg);
      Wire.write(27);
      Wire.endTransmission();
      Wire.beginTransmission(MD03_1);
      Wire.write(CMD_reg);
      Wire.write(2);
      Wire.endTransmission();
      Wire.beginTransmission(MD03_2);
      Wire.write(CMD_reg);
      Wire.write(2);
      Wire.endTransmission();
     }
  }
 else if (valueLR > 600){                  //draait links
  Wire.beginTransmission(MD03_1);
  Wire.write(SPEED_reg);
  Wire.write(128);
  Wire.endTransmission();
  Wire.beginTransmission(MD03_2);
  Wire.write(SPEED_reg);
  Wire.write(0);
  Wire.endTransmission();

  Wire.beginTransmission(MD03_1);
  Wire.write(ACC_reg);
  Wire.write(27);
  Wire.endTransmission();
  Wire.beginTransmission(MD03_2);
  Wire.write(ACC_reg);
  Wire.write(27);
  Wire.endTransmission();
  Wire.beginTransmission(MD03_1);
```

```
   Wire.write(CMD_reg);
   Wire.write(1);
   Wire.endTransmission();
   Wire.beginTransmission(MD03_2);
   Wire.write(CMD_reg);
   Wire.write(1);
   Wire.endTransmission();
  }
  else if (valueLR < 400 ) {                    //draait rechts
   Wire.beginTransmission(MD03_1);
   Wire.write(SPEED_reg);
   Wire.write(0);
   Wire.endTransmission();
   Wire.beginTransmission(MD03_2);
   Wire.write(SPEED_reg);
   Wire.write(128);
   Wire.endTransmission();

   Wire.beginTransmission(MD03_1);
   Wire.write(ACC_reg);
   Wire.write(27);
   Wire.endTransmission();
   Wire.beginTransmission(MD03_2);
   Wire.write(ACC_reg);
   Wire.write(27);
   Wire.endTransmission();
   Wire.beginTransmission(MD03_1);
   Wire.write(CMD_reg);
   Wire.write(1);
   Wire.endTransmission();
   Wire.beginTransmission(MD03_2);
   Wire.write(CMD_reg);
   Wire.write(1);
   Wire.endTransmission();

  }

}
```

6. Problemen en valkuilen

We hebben uiteindelijk grotendeels verwezenlijkt wat we wilden doen aan de start van dit project, maar er zijn uiteraard ook een aantal problemen geweest waaruit we dan het nodige hebben geleerd.

- De eerste paar weken hebben we vooral moeite gehad met een systeem te vinden om de batterijen op te laden. Aangezien dit niet in serie kon gebeuren moesten we een andere manier vinden met een relaisschakeling, dit is uiteindelijk gelukt.
- De MD03 aansturen met I²C moest gebeuren met welbepaalde adressen en deze waren niet altijd even duidelijk omdat sommige bits niet werden gebruikt. Uiteindelijk hebben we wel de juist adressen gevonden, in de tussen tijd hebben we nog een test opstelling gemaakt met een kleine motor en PWM



- Onze originele print was gemaakt op gaatjes print met foute soldeer eilanden wat het geheel erg onoverzichtelijk en slordig maakte. Toen deze ook nog niet bleek te werken hebben we besloten om op andere gaatjesprint een nieuw circuit te solderen, deze was veel overzichtelijker en netter.
- Met de omvormer van 9V battery pack naar 5V werkspanning hebben we ook de nodige problemen gehad, deze bleek de MD03's niet correct te voeden, waarschijnlijk een probleem met de stroomsterkte. Met een andere omvormer bleek dit geen probleem meer te zijn, hier zijn we veel tijd verloren.
- We hadden te weinig tijd voor een bluetooth verbinding en android applicatie te ontwikkelen, dit blijft echter een mooie toekomstige uitbreiding.

7.Conclusie

7.1 Future work

Zoals eerder gezegd zien we het aansturen van de robot draadloos de volgende stap, ook moet er gekeken worden hoe betrouwbaar de robot is op lange termijn en hoeveel belasting deze aankan.

7.2 Terugblik plan van aanpak

Ons plan van aanpak dat we op het begin van het project hebben opgesteld zullen we nu even vergelijken met onze uitwerking van het project. Het plan van aanpak zie volgende bladzijde.

- Planning: De planning is op het begin van het project is op het begin voor een groot deel hetzelfde gebleven, de batterijschakeling, aansturing Arduino en proefopstelling zijn gegaan volgens schema, echter de eerste beweging van de robot heeft langer op zich laten wachten door verschillende problemen. Sommige taken van het project hebben we onderschat omdat dit soms moeilijk was in te schatten op het begin van het project.
- Doel: Het grootste vooropgestelde doel was om de robot te laten rijden, hier zijn we in geslaagd en dit met een goede besturing met de joystick. De bijkomende doelstelling van draadloze besturing hebben we helaas nog niet kunnen verwezenlijken.
- Responsibility chart: Onze uiteindelijk taken zijn verandert in vergelijking met het plan van aanpak. Uiteindelijk hebben we de meeste taken gemeenschappelijk aangepakt omdat bij veel taken het inzicht van ons beide toch voor een sneller oplossing heeft gezorgd.

Persoon/Opdracht	Plan van aanpak	Software	Print	Hardware	Documentatie
Wynants	x	x		x	x
Goyens	x		x	x	x

PLAN VAN AANPAK

PROJECTIDENTIFICATIE

Project: PR2_robot

Groep: Stef Wynants en Nick Goyens

Als project hebben wij voor het Robotplatform gekozen.

Hardware:

- Arduino Uno
- 4x12V Batterijen
- 4x Opladers
- Robotplatform
- 2xMotorsturing MD03
- Bluetooth 4.0 shield
- Tablet voor aansturing (eventueel)

Software:

- Arduino programmeeromgeving

Mogelijke besturingsopties:

- Bluetooth (eventueel op android of PC)
- Wii nunchuk
- RC besturing

DOEL

Het doel van dit project is het robotplatform te laten rijden en het van op afstand kunnen besturen met één van bovenstaande besturingsmogelijkheden. Ook zullen we zoveel mogelijk kennis proberen op te nemen over bovenstaande technologieën op zowel hardware als software vlak. De werking en het programmeren van de Arduino Uno microcontroller, het bestuderen van de motorsturing, Bluetooth en een functionele manier zoeken om de batterijen op te laden.

Resultaat

- Verslag: Uitgebreid verslag waarin er meer informatie wordt gegeven over alle gebruikte technologieën zoals de batterijschakeling, motoren, Arduino, vermogenschakeling en software
- Software: Een programma geschreven voor de Arduino waardoor de motoren juist worden aangestuurd met de gebruikte besturingsmogelijkheden, verder moeten we ook zorgen dat we een geschikte manier vinden om de juiste Bluetooth signalen te versturen vanaf een smartphone of tablet.
- Opstelling: Rijdend robotplatform met vermogenschakeling aangestuurd door de Arduino en bluetooth met een werkende en functionele schakeling om de batterijen makkelijk op te laden.

Prioriteiten

- Zorgen dat de robot kan rijden op de opendeurdag
- Zoeken en toepassen van de meest geschikte besturingsoptie
- Kennis rond de verschillende hardware en software technologieën

Risico's en valkuilen

Het opladen van de batterijen, dit willen we doen zonder de batterijen uit de robot te halen. Het probleem hierbij is dat de batterijen serie geschakeld zijn om 24V aan elke motor te leveren. Deze moeten echter uit serie geschakeld worden bij het opladen aangezien we 12V opladers hebben. Ook moeten de batterijen losgeschakeld worden van de motorsturing bij het opladen. Voor deze punten moeten we een oplossing verzinnen met schakelaars. Doordat de motoren tot 5A stroom kunnen trekken moeten we onze schakelmethode hier ook op voorzien. Dit gaat eventueel met een 4-polig relais

We zullen onze vermogenschakeling en Arduino ook moeten voorzien van een kortsluitingbeveiliging met behulp van zekeringen, want dit is waar onze voorgangers die gewerkt hebben aan dit project in de fout zijn gegaan.

De duurbaarheid van de batterij kan een probleem vormen indien de motoren veel stroom trekken, dit zullen we moeten onderzoeken.

Voeding van de Arduino voorzien aangezien we deze willen gescheiden houden van de vermogenschakeling, dit willen we oplossen door aparte batterijen te gebruiken om de Arduino te voeden en niet de motor batterijen te gebruiken.

Behuizing van de vermogenschakeling en deze voldoende koelen, eventueel op te lossen door een ventilator in het doosje te plaatsen.

Planning

Trimester 2:

Fase/week	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9
Plan van aanpak		x	x	x					
Batterijenschakeling				x	x				
Aansturing met Arduino					x	x			
Proefprogramma					x	x			
Beweging van robot						x	x		
Beslissing Besturingswijze						x	x	x	

Trimester 3:

Fase/week	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9
Uitwerking besturing		x	x	x	x				
Definitief programma					x	x			
Afwerken rapport					x	x	x	x	

Deelprojecten

- Vermogenschakeling
- Batterijschakeling
- Arduino Uno
- Programma
- Besturing
- Verslag

Informatieplan

Alle informatie die we vinden over het project plaatsen we op onze website voor dit project https://sites.google.com/site/xiosrobotplatform/ zodat we beide te allen tijde aan deze bestanden kunnen alsook iedereen die dit project volgt.

Communicatieplan

Elke vrijdagochtend werken we samen aan het project in de les. Verder werken we zelfstandig verder aan het project. We proberen wekelijks updates op de site te plaatsen over de vooruitgang die we gemaakt hebben.

Responsibility Chart

Persoon/Opdracht	Plan van aanpak	Batterij schakeling	Software	Afstand besturing	Aansluiting	Documentatie
Wynants	x	x			x	x
Goyens	x		x	x		x

BRONNEN EN DATASHEETS

- Project site: https://sites.google.com/site/xiosrobotplatform/
- http://www.robot-electronics.co.uk/htm/md03tech.htm
- http://arduino.cc/en/Main/arduinoBoardUno
- http://www.hborienteering.com/club/resources/Battery_Charger.pdf
- http://www.klb.com.tw/dbf/WP4.5-12.pdf
- http://www.ampedrf.com/datasheets/BT23_Datasheet.pdf