Helikopter

Eind-P Project 2011

Daniël Los
Chris Lokin
Dirk-Jan vd Sanden
Elise-Ann Schrijvers

Inhoudsopgave

Voorwoord 3

1. Inleiding 4

2. Specificaties helikopter 5

2.1 Stabiliteit 5

2.2 Bestuurbaarheid 5

3. Kostenplaatje 6

4. Opbouw van het systeem 7

4.1 Communicatie interface 8

4.1.1 Zenden 8

4.1.2 Ontvangen 8

4.2 Besturing 9

4.3 Regelaar 10

4.4 Sensor interface 11

4.5 Motorsturing 12

4.5.1 Schematisch ontwerp motorsturing 12

4.5.2 PWM en software 12

4.6 Android applicatie 14

4.6.1 Lay-out Android app **Fout! Bladwijzer niet gedefinieerd.**

4.6.2 Software **Fout! Bladwijzer niet gedefinieerd.**

4.7 Printplaat 16

4.7.1 Schematisch ontwerp helikopter 16

4.7.2 PCB ontwerp 16

5. Conclusie 17

6. Bronvermelding 18

Bijlage 1: software motorsturing 19

Voorwoord

Dit verslag is onderdeel van het Eind-P project van het eerste jaar Electrical Engineering te Enschede. Het Eind-P project combineert de verschillende kennisgebieden van het eerste jaar Electrical Engineering in één systeem. In totaal wordt hier drie weken aan gewerkt en loopt van dinsdag 14 juni tot en met vrijdag 1 juli. De eerste week bestaat uit een projectplan schrijven en het specificeren van het te maken product. De overige twee weken worden gebruikt om het product daadwerkelijk te realiseren. Voor dit project is een budget van € 75,- ter beschikking gesteld.

1. Inleiding

Het verslag beschrijft de realisatie van een op afstand bestuurbare helikopter door middel van bluetooth communicatie. Het ontwerp is een kant-en-klaar gekocht model, waarbij de elektronica is vervangen door eigen ontworpen componentenschema's. De motortjes van de helikopters zijn als enige elektronica nog identiek aan het gekochte model. De bluetooth besturing wordt met een Android applicatie op een gsm gerealiseerd.

2. Specificaties helikopter

De volgende eisen worden gesteld aan de helikopter:

- Stabiel hangen in de lucht
- Via bluetooth bestuurbaar

2.1 Stabiliteit

In principe wordt naar onderstaande eisen gestreefd wanneer er geen input wordt ontvangen:

- Binnen het horizontale vlak zal de helikopter nauwelijks een beweging mogen maken.
- Draaiing rond de verticale as wordt tot een minimum beperkt, een eventuele afwijking wordt vast ingesteld en zal dus niet elektronisch geregeld worden. Eventueel kan een soort offset ingesteld worden om de besturing voor de bestuurder iets gemakkelijker te maken.
- *De hoogte wordt elektronisch bijgeregeld, waarbij de sensor tot ongeveer 3 meter kan meten met een afwijking van 5%.*

2.2 Bestuurbaarheid

De helikopter is op afstand bestuurbaar door middel van bluetooth communicatie met een gsm. Een speciaal hiervoor ontworpen Android applicatie maakt het mogelijk stuur- en throttle-acties uit te voeren om zo de helikopter te besturen.

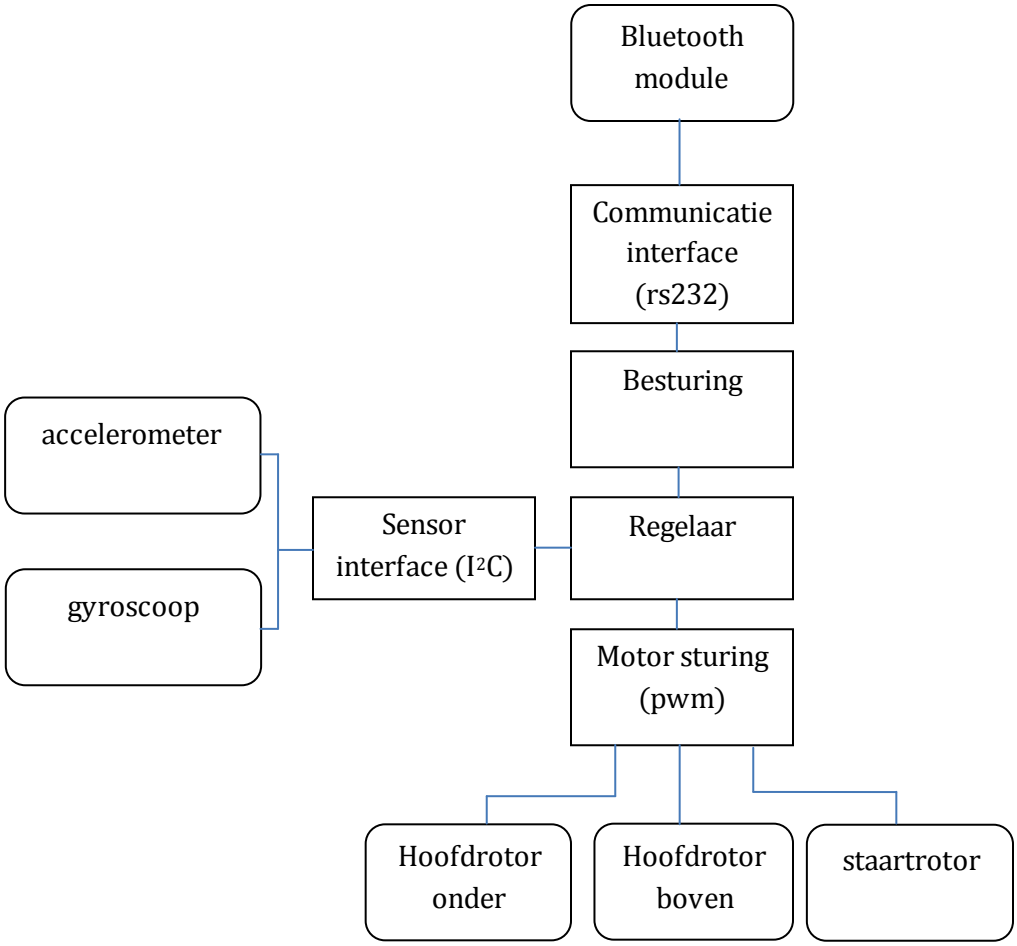
3. Kostenplaatje

Het ter beschikking gestelde budget voor dit project is € 75,-. Hieronder een overzicht van de daadwerkelijk gemaakte kosten.

Aantal	Component	Leverancier	Prijs
1	Bluetooth module	Deal Extreme	€ 5,11
1	Helikopter	Marktplaats	€ 20,00
1	Gyroscoop	Farnell	€ 15,00
1	Accelerometer	Farnell	€ 3,00
2	AtMega88	Stores	€ 5,16
2	Kristal 3.6864 MHz	Stores	€ 0,60
1	Kristal 4 MHz	Stores	€ 0,20
2	Pin header 1 x 40 male	Stores	€ 0,26
1	Fotogevoelige print dubbelzijdig	Stores	€ 2,50
1	Krimpkous	Stores	€ 0,24
		Totaalprijs:	€ 52,07

4. Opbouw van het systeem

Het hele systeem wordt grofweg onderverdeeld in de android applicatie en de helikopter zelf. De android applicatie is uitgewerkt in paragraaf 6 van dit hoofdstuk. De helikopter zelf is op te delen in 5 deelblokken, te zien in figuur 4.1. Deze deelblokken zijn stuk voor stuk uitgewerkt in de paragrafen 1 t/m 5 van dit hoofdstuk.



Figuur 4.1 Blokschema van de opbouw van het systeem

4.1 Communicatie interface

De communicatie met de bluetooth module gebeurd via een RS232 protocol. Als er verbinding is gelegd met de android telefoon kan deze verbinding gezien worden als RS232 interface, alles wat aangeboden wordt aan de bluetooth module zal nu direct doorgestuurd worden naar de telefoon. De gebruikte microcontroller heeft hardware ondersteuning voor het zenden en onvangen via RS232, de USART. Wel kan deze USART maar 1 byte onthouden om te verzenden en 1 byte om te ontvangen.
De bluetooth module heeft ook een commando-set, hiermee kan de module ingesteld worden, er kan gescand worden naar aparaten, er kan een verbinding met een apparaat gelegd worden, etc. De software in de microcontroller maakt hier geen gebruik van omdat we de helicoper als slave gebruiken, de telefoon initiëerd dus de verbinding. Wel is met behulp van deze commando's een naam ingesteld voor het device: "Heli0013". Dit is echter gebeurd vanuit een terminal met een computer.

4.1.1 Zenden

Voor het zenden van data wordt gebruik gemaakt van een ringbuffer. De werking van de ringbuffer is het makkelijkts uit te leggen m.b.v. een voorbeeld; stel we willen de tekst "hallo" verzenden. Deze tekst wordt eerst in een array gezet, zodra de USART zendbuffer van de microcontroller leeg is kan daar de eerste letter in gezet worden, in het geval van het voorbeeld is dit de letter "h". Hierna wordt hetzelfde voor de andere bytes gedaan, als de " h" dus verzonden is, wordt de "a" in de zendbuffer gezet, enzovoort.

4.1.2 Ontvangen

Het ontvangen werkt iets anders. Op het moment dat er data klaar staat in de USART ontvangbuffer wordt er een interrupt uitgevoerd (er wordt opgeslagen waar de microcontroller mee bezig was en even naar een ander stukje code gesprongen). In dat interrupt wordt de onvangbuffer gelezen en weggeschreven in een variable, ook wordt een flag gezet zodat de bestuing weet dat er data verwerkt kan worden.

4.2 Besturing

De besturing zorgt ervoor dat de inkomende data van de bluetooth interface verwerkt wordt en dat de regelaar ingesteld kan worden. De basis van de besturing in een toestands machiene, deze functie weet hoeveel bytes er ontvangen moeten worden en weet wat de commando's betekenen. Vanuit de besturing worden dus de functies aangeroepen van de regelaar, om te sturen, op te stijgen, maar ook de parameters van regelaar worden hiermee ingesteld.

4.3 Regelaar

Regeling

Uiteindelijk is het doel dat de helikopter correct reageert op de besturingsinput. Aan de hand van de opbouw van de helikopter zijn de volgende aansturingsmethodes bedacht:

- Voor het naar voren en naar achter bewegen wordt de snelheid van de staartrotor aangepast. Bij een hogere rotatie tilt de kopter wat naar voren, waardoor hij naar voren zal bewegen. Bij een lagere rotatie gebeurt het omgekeerde en gaat hij naar achter.
- Voor het naar links en naar rechts bewegen gebruiken we het verschil in moment van de hoofdrotors om te sturen. Als een van de hoofdrotors langzamer draait dan de ander, zal de kopter mee gaan draaien in die richting. Denk aan een helikopter waarvan het staartrotortje niet meer werkt.
- Het omhoog en omlaag bewegen van de kopter is eenvoudig te doen: verhoog de rotatie van beide hoofdrotors even veel om op te stijgen, verlaag de rotatie evenveel en de kopter daalt.

Om na te gaan of de uitgevoerde koerscorrectie klopt met de besturing is een regelcircuit nodig. Hiervoor is er sensordata nodig die de daadwerkelijk uitgevoerde verplaatsing en draaiing van de kopter terugvoert aan het regelcircuit. Deze data wordt vergeleken met de besturingsinput, en eventuele afwijkingen worden doorgevoerd naar de output van het regelcircuit.

Als regelcircuit zullen we een P(I)D regelaar gebruiken. Dit is analoog en digitaal uit te voeren. Het voordeel van een analoge schakeling is met spoelen en condensatoren makkelijk een geïntegreerde en gedifferentieerde vorm van het uitgangssignaal is te maken. Ook is het aanpassen van de P-, I- en D-gain makkelijk te doen met wat potmeters. Een nadeel is dat een extra circuit weer extra ruimte inneemt en gewicht toevoegt.

Een voordeel van een digitale schakeling is dat er geen extra componenten nodig zijn als er al een microcontroller in het systeem gebruikt wordt. Ook is de software eenvoudig te schrijven. Een nadeel is dat de gain-factoren in de software zitten. Voor het aanpassen hiervan moet de chip dus opnieuw geprogrammeerd worden. Dit is eventueel op te lossen door de gain-factoren te bepalen door het volgtage te meten aan wat ingangspootjes van de microcontroller en dit voltage aan te passen is door een potmeter.

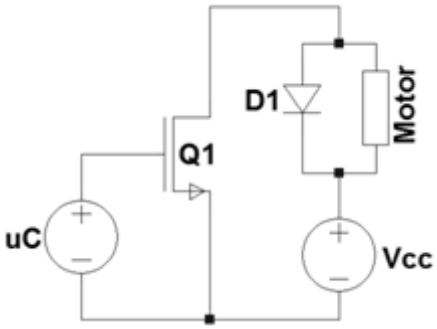
Aangezien we toch al een microcontroller gaan gebruiken in ons ontwerp, kiezen we voor de digitale schakeling.

4.4 Sensor interface

De sensor interface zorgt ervoor dat de registers voor I²C in de microcontroller juist ingesteld worden en dat er data over de I²C bus gestuurd en ontvangen kan worden. Er is gebruik gemaakt van application note AVR315 van Atmel. Het mooie aan deze software is dat het interrupt gebaseerd is, als de software goed gebruikt wordt zitten er dus geen wacht loops in. De regelaar kan dus bijvoorbeeld een functie aanroepen om data van een sensor op te vragen, zodra de data van de sensor ontvangen is kan de regelaar dit verwerken en tussen het sturen en ontvangen van de data kunnen gewoon andere instructie uitgevoerd worden door de microcontroller.

4.5 Motorsturing

De motoren in de helikopter worden aangestuurd met een microcontroller, echter kunnen deze daarop niet direct worden aangesloten. Zou dit echter wel gedaan worden, dan wordt de benodigde stroom uit de microcontroller getrokken, waardoor deze hoogstwaarschijnlijk stuk zal gaan. Een schakeling die de stroom uit de microcontroller versterkt voorkomt dit probleem. In figuur 5.4-1 wordt dit schema weergegeven.



Figuur 5.4-1 Ontwerp schema motoraansturing

4.5.1 Schematisch ontwerp motorsturing

Motor

Afhankelijk van wat de motorratings zijn kunnen de overige componenten worden bepaald. Hieronder een overzicht van deze ratings die zijn gevonden aan de hand van een aantal verrichte metingen:

Hoofdrotor 1 & 2	volle batterij
minimale thrust	2.40 V 450 mA
maximale thrust	3.10 V 670 mA
Staartrotor	volle batterij
minimale thrust	1.60 V 90 mA
maximale thrust	3.10 V 200 mA
Piekstroom	± 800 mA

Transistor

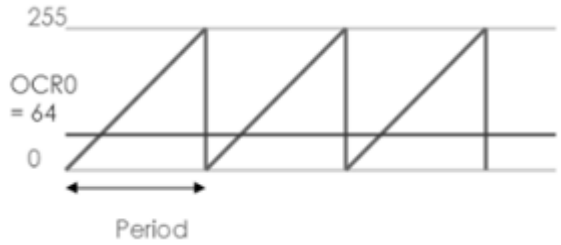
De gekozen MOS-fet heeft een hoge I_d waarde, zodat deze gemakkelijk de piekstroom van de motoren aan kan. Het type is "IRLML2803" deze kan een constante stroom leveren van 1,2 A en is uitgevoerd in een SOT-23 behuizing, dus neemt niet veel ruimte op de printplaat in.

Terugslagdiode

De stroom die door de motor stroomt veroorzaakt een magnetisch veld, wordt de motor uitgeschakeld dan ontstaat er tegengesteld gericht veld. Het tegengesteld gerichte veld zorgt eveneens voor een andere kant opgaande stroom, deze kan zowel de transistor als de microcontroller beschadigen. De terugslagdiode zorgt ervoor dat de ontstane spanning over de diode kan wegvloeien, waardoor het veld zal uitdoven en de componenten niet zullen beschadigen.

4.5.2 PWM en software

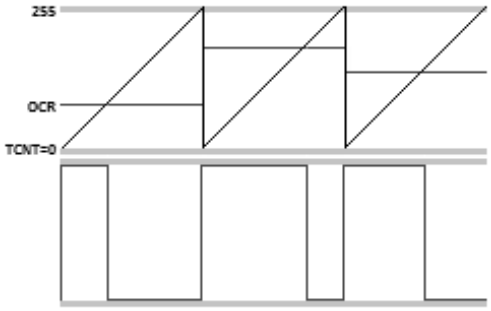
De microcontroller kan in feite enkel een 0 of een 1 op de uitgang zetten, dit is niet handig voor het aansturen van een motor die op verschillende snelheden moet kunnen draaien. Wordt het signaal vanuit de microcontroller pulsbreedte gemoduleerd (PWM) aangeboden op de motor, dan kan de motor toch op verschillende snelheden draaien. Dit komt doordat de duty-cycle van het signaal verandert wordt.



Figuur 5.4-2 Output Compare Register (OCR)

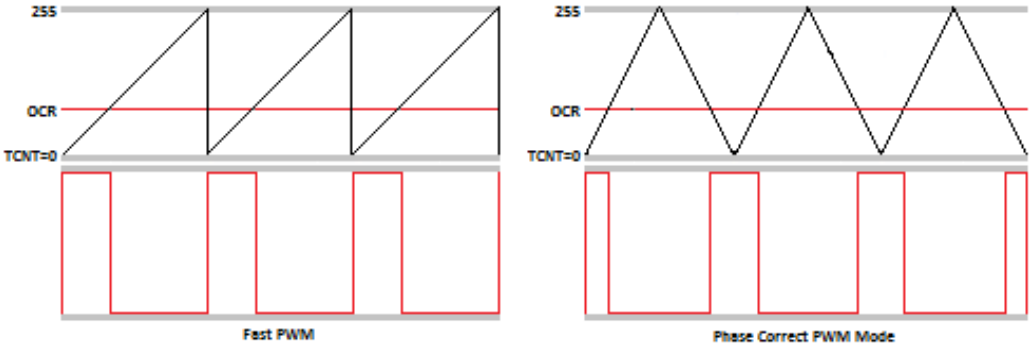
Om een PWM signaal te generen met de Atmega88 kunnen zowel 8-bits als 16-bits timers gebruikt worden. Deze tellen van 0-255 of van 0-65.536. De periode van het signaal wordt verandert door het Output Compare Register (OCR) anders in te stellen. In figuur 5.4-2 wordt dit geïllustreerd aan de hand van een afbeelding. Een signaal met duty-cycle 64 zal $\frac{64}{256} = 25\%$ van de voedingsspanning aan de uitgang zetten.

De waarde voor het Timer Counter Register (TCNT) geeft aan op welke waarde het signaal start met PWM'en en wordt vervolgens de OCR waarde bereikt dan zakt het output signaal van de PIN naar nul, totdat de timer geteld heeft tot 255 of 65536. Om weer opnieuw te beginnen met tellen. Zie figuur 5.4-3 voor een illustratie van deze waarden.



Figuur 5.4-3 PWM Signaal

In het Timer Counter Control Register (TCCR) wordt vastgelegd welke klokken en output pinnen geselecteerd worden. Eveneens welk golfsoort gebruikt gaat worden, Fast PWM of Phase Correct PWM Mode. Deze laatste wordt gebruikt om de motortjes mee aan te sturen. Hierbij verhoogt TCNT zijn waarde van bodem tot de top en verlaagt zich daarna van top tot bodem. In tegenstelling tot de Fast PWM waarbij enkel van bodem tot top wordt gegaan. Bij Phase Correct PWM Mode komt het erop neer dat het signaal vergeleken wordt met de helft van de PWM signaal frequentie, waardoor deze preciezere waarden kan aannemen. In figuur 5.4-4 is dit verschil duidelijk te zien.



Figuur 5.4-4 Fast PWM en Phase Correct PWM Mode

De software die zorgt dat de motoren pulsbreedte gemoduleerd worden aangestuurd staat weergegeven in bijlage 1.

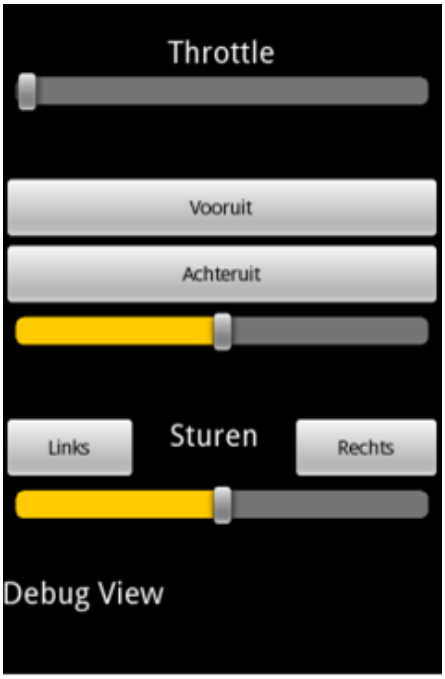
4.6 Android applicatie

Voor de besturing op afstand van de helikopter is een Android applicatie gemaakt.

Lay-out Android app

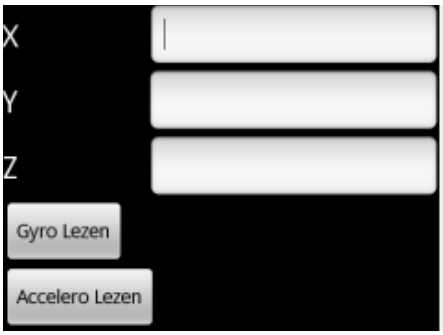
De applicatie bevat een aantal schermen. Het hoofdscherm heeft schuifbalken waarmee de helikopter bestuurd kan worden. In het menu zitten 3 knoppen, één voor het initialiseren van de bluethooth communicatie, één voor het instellen van de regelaar en één voor het uitlezen van de sensoren.

Het hoofdscherm is niet heel handig gemaakt om te besturen. De reden hiervoor is dat het bedoeld is om makkelijk de afzonderlijke delen van de besturing te kunnen testen. Het is niet uitgewerkt tot iets beter bedienbaars omdat dat niet kon voordat er een geregelde helikopter is. Het scherm ziet er uit al in FIGUUR.



FIGUUR

Via de menutoets die elke android telefoon heeft kan er een popup geopend worden. In de connect popup kan er een verbinding worden opgezet met een apparaat. Voor debugging kunnen we hier voor een laptop kiezen en voor de echte test selecteer je de helikopter.



FIGUUR

Voor het instellen van de regelaar is een popup gemaakt waar de de P, I en D waarden ingesteld en verzenden naar de helikopter kunnen worden. Er is ook een mogelijkheid om de huidige waarden uit te lezen.

De derde popup is voor het uitlezen van de sensoren. De X, Y en Z waarden van 1 van de twee sensoren kan uitgelezen worden met de bijbehorende knop. De inhoud van deze popup staat in FIGUUR.

Software

De applicatie heeft een aantal functies zoals zenden, ontvangen en verbinden. Ook is er communicatie van de popups met het hoofdscherm, de popups zijn namelijk andere klassen dus moeten die aan elkaar gekoppeld worden.

Verbinden

Voordat er iets mogelijk is moet er eerst verbinding gemaakt worden met de helikopter. Hiervoor is een standaardprocedure gehanteerd. Er wordt en socket aangemaakt en via dat socket kan er data in een outputStream verzonden worden. Ontvangen data wordt in een inputStream gezet.

Verzenden/Ontvangen

Voor het verzenden van data wordt de write(byte) methode gebruikt. Deze schrijft de meegegeven byte naar de outputStream en komt daarna aan bij de bluetoothmodule in de helikopter.

Het ontvangen van data was lastiger. Dit gebeurt in een aparte thread die constant kijkt of er data in de inputStream zit. Als er data in zit wordt deze data doorgegeven. Dit noem je een InputListener

Interne communicatie

De klassen communiceren met elkaar via static methoden. Dit moet omdat het een methode is van de klasse in het algemeen en niet van een instantie van de klasse.

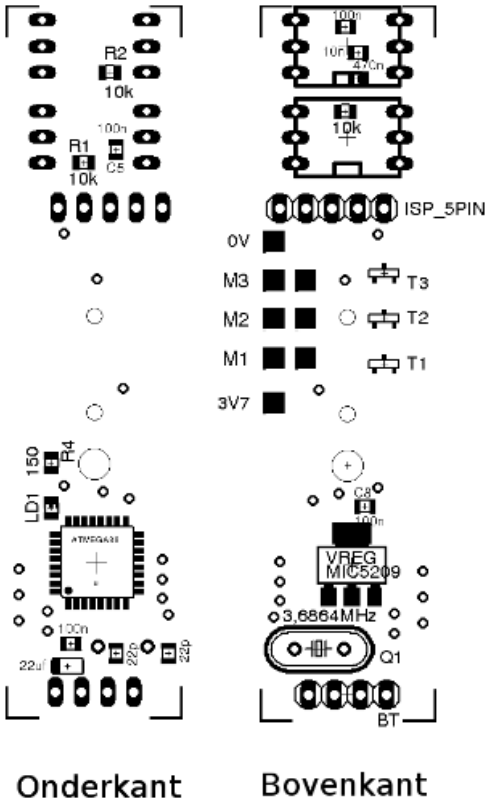
Stel je wil een sensor uitlezen: Vanuit de popup wordt de write() methode van het hoofdscherm aangeroepen. De helikopter ontvangt nu de instructie en zendt de huidige waarde van de sensor. De inputStream bevat nu data en geeft deze data aan een handler. Die handler zorgt ervoor dat de setWaarde() methode van een popup scherm wordt aangeroepen. Vervolgens zorgt setWaarde() ervoor dat de gelezen waarde zichtbaar is in de popup.

4.7 Printplaat

4.7.1 Schematisch ontwerp helikopter

Het schema is vrij eenvoudig opgebouwd door het gebruik van de microcontroller. Het schema is te zien in de bijlage. De sensoren zitten beide via een I²C bus en kunnen op de microcontroller aangesloten. De bluetooth module komt met een connector op de RS232 interface van de microcontroller. De motoren worden aangestuurd met power-mosfets van het type IRLML2803. Het hele circuit(behalve de motoren) wordt gevoed met 3V3, hiervoor is de voltage regelaar verantwoordelijk, de MIC5209. Deze regelaar heeft10mV dropout, dit is een belangrijke eigenschap omdat de batterijspanning in kan zakken tot 3,5V.

4.7.2 PCB ontwerp



Figuur 4.7-1 De onder- en bovenkant van de printplaat met onderdelen

In figuur 4.7-1 is een afdruk van het ontwerp met de onderdelen te zien. Het realiseren van de printlaat is gedaan d.m.v. etsen op een dubbelzijdig fotogevoelige printplaat. De gebruikte componenten zijn voornamelijk in SMD formaat, omdat er weinig ruimte beschikbaar is in de helikopter. De connectoren zijn ook zo optimaal mogelijk gekozen, zo worden de draadjes voor de motoren en de voeding op de bovenzijde van de print gesoldeerd en is gekozen voor een niet standaard programmeer connector.

5. Conclusie

6. Bronvermelding

Bijlage 1: software motorsturing

```
#include <avr/io.h>
#include <avr/iom88.h>
#include <avr/interrupt.h>
#include <inttypes.h>


int main(void){
  //Settings input/output
    //Define ports as input/output
       DDRD |= _BV(PD3); //PIN3 van PORTD output
       DDRD |= _BV(PD5); //PIN5 van PORTD output
       DDRD |= _BV(PD6); //PIN6 van PORTD output


  //Settings Pulse Width Modulation
    //Registers 8-bit PWM
       //Configuration Timer Counter Control Register 0A
       TCCR0A = 0b10100001;
              //1: Compare Match Output A Mode (COM0A1)
              //0: Compare Match Output A Mode (COM0A0)
              //1: Compare Match Output B Mode (COM0B1)
                    //0: Compare Match Output B Mode (COM0B0)
              //0: -
              //0: -
              //0: Waveform Generation Mode (WGM01)
              //1: Waveform Generation Mode (WGM00)

       //Configuration Timer Counter Control Register 0B
       TCCR0B = 0b00000010;
                     //0: Force Output Compare A (FOC0A)
                     //0: Force Output COmpare B (FOC0B)
                     //0: -
                     //0: -
                  //0: Waveform Generation Mode (WGM02)
                     //0: Clock Select (CS02)
                     //1: Clock Select (CS01)
                     //0: Clock Select (CS00)

       //Configuration Timer Counter Control Register 2A:
       TCCR2A = 0b00100001;
                //0: Compare Match Output A Mode (COM2A1)
                //0: Compare Match Output A Mode (COM2A0)
                //1: Compare Match Output B Mode (COM2B1)
                //0: Compare Match Output B Mode (COM2B0)
                //0: -
                //0: -
                //0: Waveform Generation Mode (WGM21)
                //1: Waveform Generation Mode (WGM20)

       //Configuration Timer Counter Control Register 2B:
       TCCR2B = 0b00000010;
              //0: Force Output Compare A (FOC2A)
              //0: Force Output Compare B (FOC2B)
              //0: -
              //0: -
              //0: Waveform Generation Mode (WGM22)
              //0: Clock Select (CS22)
              //1: Clock Select (CS21)
              //0: Clock Select (CS20)
```

```
      //Configuration Timer Counter Interrupt Mask register:
      TIMSK0= (0 << OCIE0B); //Timer0 Output Compare Match 0B
      TIMSK0= (0 << OCIE0A); //Timer0 Output Compare Match 0A
      TIMSK0= (0 << TOIE0);  //Timer0 Overflow Interrupt

      TIMSK2= (0 << OCIE2B); //Timer2 Output Compare Match 2B
      TIMSK2= (0 << OCIE2A); //Timer2 Output Compare Match 2A
      TIMSK2= (0 << TOIE0);  //Timer2 Overflow Interrupt


  //Settings Value Timer Counter Register
    //Value Timer Counter Register 0
    TCNT0=0; //Always starts Register 0 counting from 0

    //Value Timer Counter Register 2
    TCNT2=0; //Always starts Register 2 counting from 0

  //Settings Value Output Compare Register
    //Value Output Compare Register A
      OCR0A=10;  //The OCR0A is continuously compared with TCNT0
      OCR0B=100; //The OCR0B is continuously compared with TCNT0

    //Value Output Compare Register B
      OCR2B=200; //The OCR2B is continuously compared with TCNT2

while (1) { }
}


int get_speed_motor(unsigned char motor){
      switch(motor){
            case 1:
                  return OCR0A;
                  break;
            case 2:
                  return OCR0B;
                  break;
            case 3:
                  return OCR2B;
                  break;
      }
      return 0;
}


int set_speed_motor(unsigned char speed, unsigned char motor){
      switch(motor){
            case 1:
                  OCR0A = speed;
                  break;
            case 2:
                  OCR0B = speed;
                  break;
            case 3:
                  OCR2B = speed;
                  break;
      }
      return 0;
}
```