2016

Ben De Lathouwer

Project 2

Paranoid android project

1.inhoudstafel

2.voorwoord

2.1. kenismaking

Ik ben Ben De Lathouwer en ben 21 jaar oud

Ik ben student aan het Thomas more instituut campus De Nayer en ik zit in men 2de jaar elektronica-ICT hobby’s zijn elektronica, treinen en muziek. Ik ben sinds kort lid van een groep die zich bezighoud met het exploiteren van een oude spoorlijn. Namelijk het Stoomcentrum in Maldegem

3.De opdracht

Mijn opdracht bestaat er in om een robot te bouwen en te kunne besturen met een rc transmitter. Eveneens is in deze robot een anti bots systeem .Dit systeem zorgt er voor dat de auto niet bots me een ander voorwerp.

3.1. de opstelling

In dit project heb ik gebruik gemaakt van de volgende onderelen :

- 1 psoc 1

- robot chasis

-1 motor controller

- 1 ultrasoon sensor

-1 rc controller

4. plan van eisen (pve)

1.De robot moet relatief goedkoop zijn

2. De robot moet op afstand bestuurbaar zijn

3. De robot mag niet botsen met andere voorwerpen.

4. In een botsings situatie moet de robot nog steeds bestuur baar blijven in allee richtingen maar niet in de voorste richting

5. de print moet zo universeel mogelijk gemaakt zijn

5. Plan van aanpak

1. keuze robot chasis

2. keuze remote controller

a. zelfbouw of kant en klaar.

3. keuze microcontroller board

4. ieder element apart programmeren

5. alles samen voegen

6. prototype pcb maken

7. prototype pcb testen.

8. alles samen voegen

6. Benodigde stappen.

1. brainstormen rond project keuze

2. project voorstel maken

3.brainstormen met project mentor

4. herhaal stap 2 en 3 3 maal

5. brainstormen over het brein van de robot.

6. rc controller kopen

7. beginnen met programmeren van project

8. eerste microcontroller opblazen

9. vloeken en me afvragen waarom het opgeblazen is

10. nieuwe controller zoeken

11. nieuwe controller gevonden

12. onderzoek doen naar motordriver

13. motor driver gekozen

14. motordriver gaan halen.

15 onderzoek doen naar ultrasoon sensor

16. hc-04 ultrasoon sensor gekozen

17. rc controller programmeren en testen

18. ultrasoon sensor programmeren en testen

19 motordriver programmeren en testen

20 alles integreren op print

21 genieten van het resultaat

7. Robots

Wat verstaan we onder robots? Onder robots verstaan we machines die geprogrammeerd zijn om verschillende taken uit te voeren . Hierin verschild hij van een numerieke machine .Deze is geprogrammeerd om slechts 1 taak uit te voeren . Terwijl een robot verschillende taken kan doen . Een welgekend voorbeeld van een robot zijn degene die in de Audi fabriek staan te vorst. Of voor de mensen die dit niet kenen er staat zo een robot op onze school. Eveneens moete er een bepaalde artificiële intelligent zijn bij een machine voor we van een robot kunnen spreken . Maar dit wordt echter door de minder ingewijden door elkaar gebruikt .

8. Waarom dit project?

Ik heb voor dit project gekozen omdat ik graag altijd al een robot heb willen bouwen .Eveneens omdat ik graag eens wat willen doen met een rc remote controller.

9. versimpeld blokschema

In dit blokschema ziet u de drie grootte bouwstenen van mijn project. Deze bestaan uit de afstands sensor , afstand bediening , de verwerkingen eenheid en als laatst de motoren.Deze drie blokjes ga ik in het volgende stukjes voor u nauwgezet uit de doeken doen .

Motoren

Verwerkings eenheid

Afstands sensor

+ Afstandsbediening

+-

9.1 de afstands sensor

Als afstands sensor heb ik gekozen voor eentje op het ultrasone werkings princiepe. Deze sensoren meten de afstand doormidden van een geluidspuls . eens deze geluidspuls is uitgezonden wacht de sensor tot deze weerkaatst is van een voorwerp. Dit type sensor werkt op 3 verschillende methodes.

Namelijk:

1) **Broadband methode**. Bij deze methode worden er meerderen frequenties uitgezonden . Maar wel binnen een bepaalde bandbreete ( bv in de bandbreete van 0 tot 20KHz) en afhankelijk van de afgelegde tijd van de weerkaatste golven kan men de afstand bepalen

2) **Narrowband methode.** Bij deze methode zenden we 1 toon uit op 1 bepaalde frequentie . Waardoor men uit faseverschuivingen de afstand kan bepalen . Deze methode kan echter niet aleen gebruikt worden om afstandsmetingen te doen. Dit omdat de aantal voledig doorlopende cycli van de golf niet gekend zijn . Maar als we deze methode combineren met de broadband methode is deze methode echter wel bruikbaar

3) **de correlatie methode**. We zenden een bepaalde pulsgrootte of pulstrein uit , gekend door de receiver. Zodat men aan de hand van de opgetreden vervorming de afstand kan bepalen.\

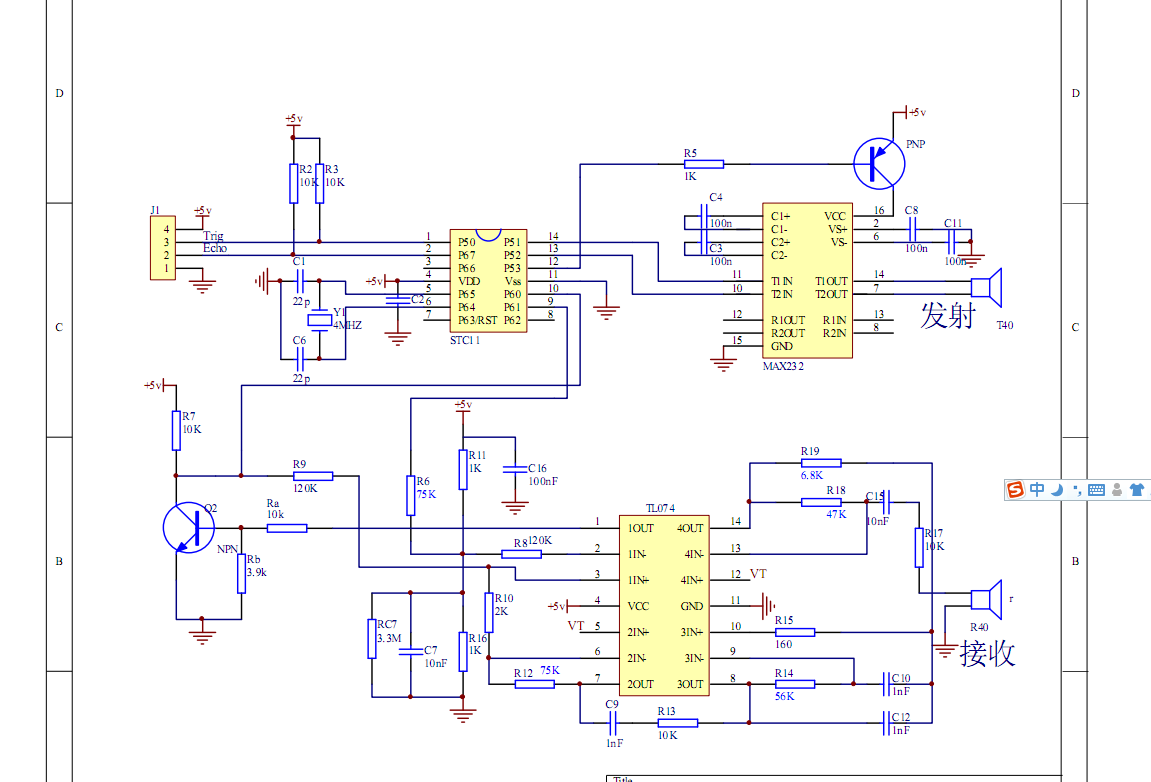
De ultrasoone afstands meter bestaad uit 2 groote delen namelijk: De zender en de ontvanger.

De zender: Als zender j maken we gebruik van het piëzo-elektrisch effect(Als we een spanning op een piëzo crystal aanleggen krijgen we geluid) . Deze buzzer zend golven uit met een wel bepaalde frequentie en een oscillator zorgt ervoor dat we op die welbepaalde frequente kunnen zenden.

De ontvanger : Hierbij maken we weerom gebruik van het het piëzo-elektrisch effect( maar nu maken we gebruik van het feit dat als we een druk uitoefenen op het piezo crystal er een spanning ontstaad over het crystal).Deze zet dus mechanische trillingen om naar electrische trillingen. Deze worden dan vergeleken en zo hebben we de afstand kunnen meten.

Als Ultrasoon sensor heb ik een HCSR04 gekozen. Niet omdat ik die beter is maar om de simpele reden dat ik deze al reeds in mijn bezit heb en er is iets deftig wou mee doen.

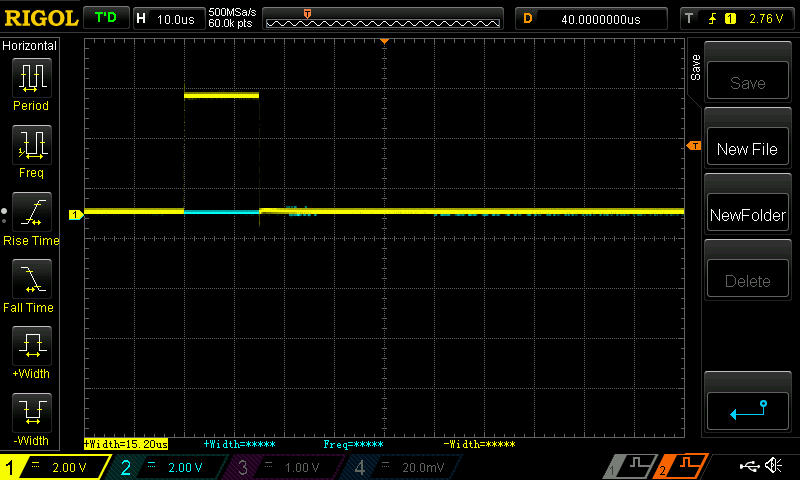
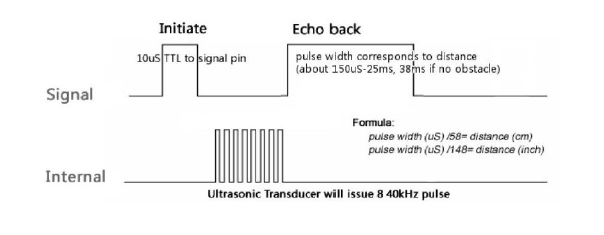
Het schema van zo een ultrasoon sensor ziet er als volgend uit:



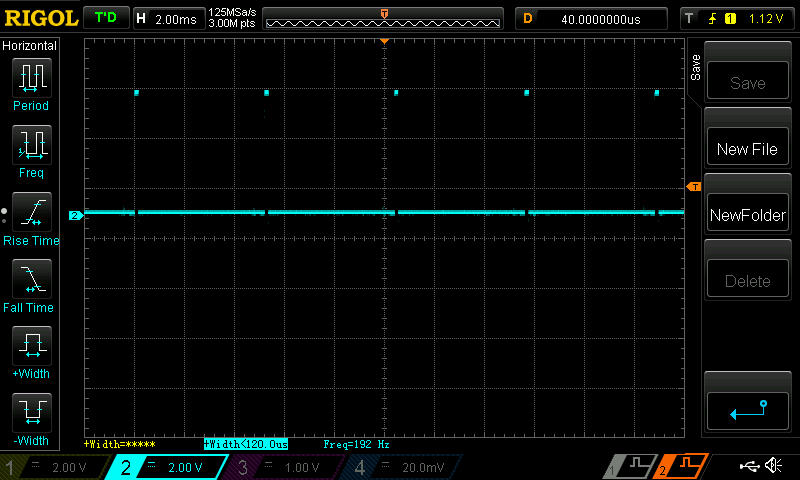
Links boven in het schema zien we onze oscillator. Deze zorgt voor onze dat onze frequentie nauwkeurig en constant blijft. Deze eigenschappen zijn te danken aan het kristal en zijn condensatoren .Dit kristal gaat aan een zeer nauwkeurig trillen als er een spanning word op gezet. De capaciteiten die rond het kristal zijn aangebracht. Deze dienen om storingen van buiten af op je kristal te vermijden. Deze blokgolf gaat dan naar een MAX232 ic. Dit ic wordt eigenlijk gebruikt als interface chip tussen een microcontroller en een rs232 apparaat. Maar in deze toepassing wordt het ic eigenlijk een beetje misbruikt. We gebruiken hier namelijk de charge pump mogelijkheden van dit ic . Onder charge pump verstaan we een schakeling die er voor zorgt dat onze spanning wordt opgevoerd. Dit kunnen we vergelijken met een boost convertor. Na dat deze spanning is opgedreven sturen we deze naar onze ultrasoon transmitter. De ontvanger staad aangesloten op een comparator schakeling. Een comparator schakeling is een schakeling die een gekende spanning met onze ingang en gaat naderhand omklappen . Als Vin <Vgekend dan is Vout laag en als Vin> Vout dan klapt de comparator om naar een hoge toestand. En zo word onze uitgangsspanning bepaald ivm met onze afstand .

Om deze sensor aan de gang te krijgen moet ik de oscillator in gang kunnen steken. Dit doen we door een 10uS puls op de trigger ingang aan te sluiten . en dan kunnen we onze e echo pin meten op de pulsbreete die everedig is met de afstand.(pulsbreete tijd/58 is de afstand in cm )

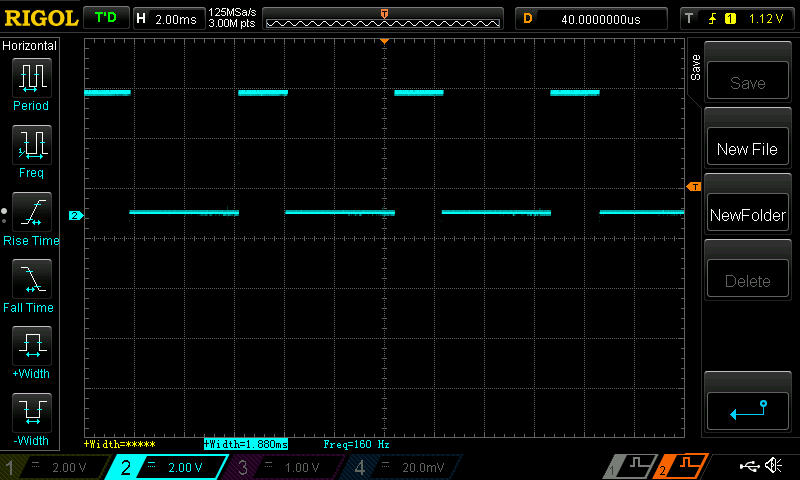
\Hierboven zit u de theoretische werking van mijn ultrasoon sensor



Hierboven ziet u de triggerpuls



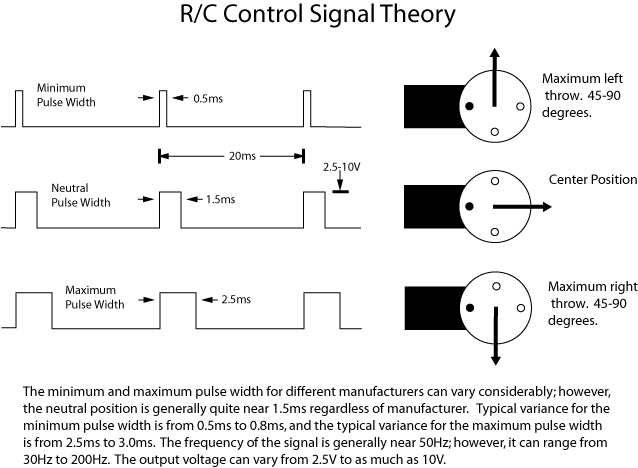
Hierboven zit u de echopuls in de minimumafstand



Hierboven zietu de echo pin op de max afstand

9.2 de afstandsbediening

Als afstand sensor heb ik gebruik gemaakt van een standaard modelbouw controller met spectrum technologie. Deze afstandsbediening werkt op radio golven . Met een frequentie van 2.9Ghz onder de spectrum technologie verstaan we dat de zender de eerste best vrije band kiest binnen dit spectrum . Aan de ontvangers kan hebben we te maken met een servo signaal



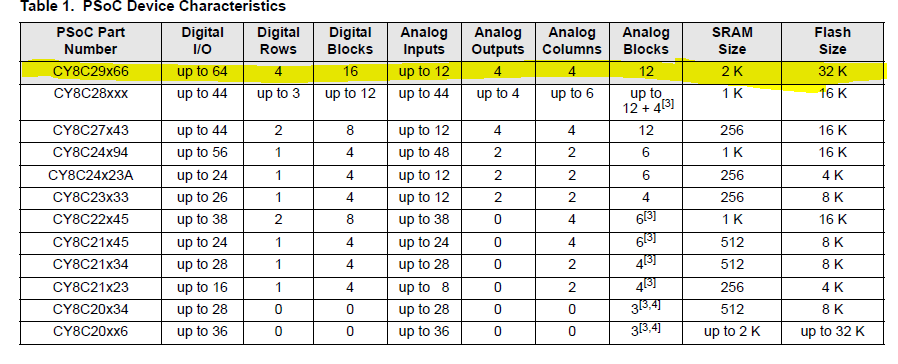
Hierboven ziet u het servo signaal en waarbij de pulsbreete van onze blokgolf afhangt van de positie van de stick op de zender.

10. de verwerkings eenheid

Als verwerking ’s eenheid heb ik gekozen voor een Psoc . Dit is aan afkorting Programebel system on chip. Ik heb voor dit soort controller gekozen omdat ik Altijd eens wat met een psoc wou doen. Een Psoc microcontroller onderscheid zich van andere microcontrollers doordat deze vrij configureerbaar zijn . Dit wilt zeggen als je 10 pwm’s nodig dan kan dat . Of 20 analoge ingangen nodig ? waarom ook niet (natuurlijk binnen de perken van de chip zelf). Analoge mogelijkheden bezit de chip ook . Hiermee wil ik zeggen dat je er ook virtueele opamps. Datik ze heb ik gekozen is om verschillende redenen maar de voornaamste reden is dat deze verkrijgbaar is in dip behuizing. Eveneens had Mr Dams er een developmet board van liggen.

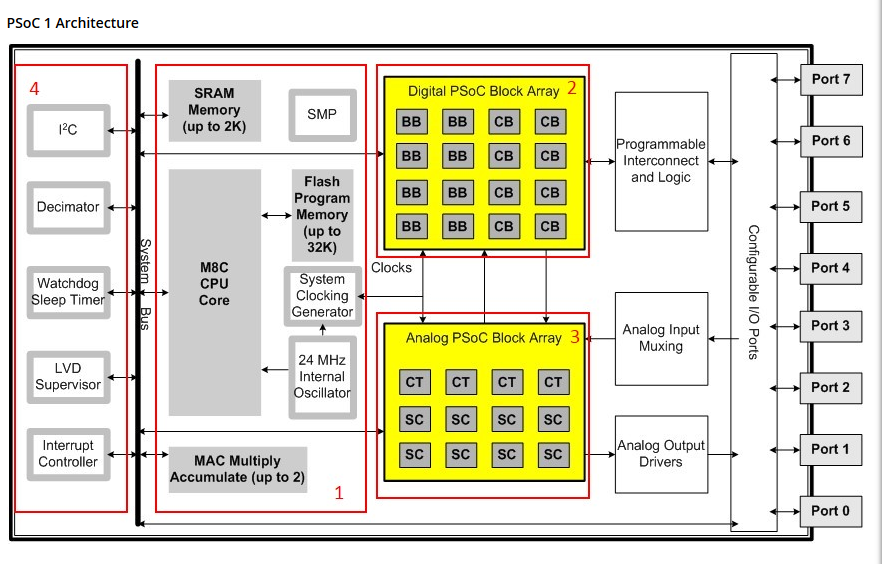
Uit het grote assortiment aan Psoc microcontrollers heb ik gekozen voor een Psoc 1 meerbepaald de CY8C29466PXI. Deze chips bevatten oa: PWMs (8- and 16-bit), Counters (8- to 32-bit)

, Timers (8- to 32-bit), en nog veel meer .

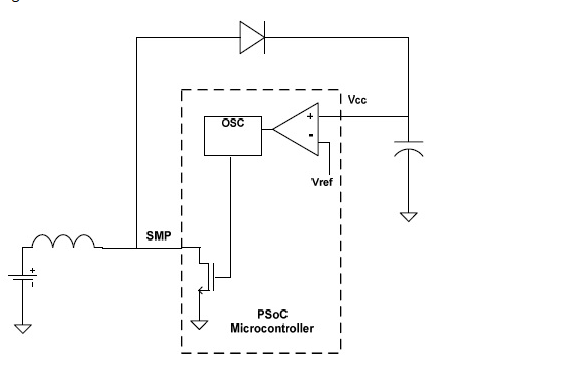


In bovenstaande tabel ziet u de verschilende mogelijkheden van io blocks en geheugen . In het geel ziet u de gegevens van mijn gebruikte ic .De aantal io poorten hangt af van van de groote van je ic package . in mijn geval is het slecht 46 input en output mogelijkheden dit omdat mijn ic een dip package is .

Intern ziet een ic er zo uit :



1. Dit is de echte core van mijn microcontroller. Wat verstaan we nu onder een core verstaan we de verwerkings eenheid van een controller met al zijn periferie . in dit geval de sram , interne oscillator en clock generator . eveneens een SMP wat verstaan we nu onder SMP? SMP staat voor Switch Mode Pump. Deze zorgt er voor dat we onze controller kunnen voeden met een knoopcel.



Hierboven ziet u welke componenten we nodig hebben om zo een SMP aan de gang te krijgen. Namelijk 1 spoel , 1 condensator en 1 diode . Zo maken we een soort van boost convertor nu hoe werkt het bovenstaande schema? Dit kan ik u uitleggen aan de hand van2 stappen namelijk . de mosfet geleid en de mosfet stopt met geleiden.

1. De mosfet geleid: als de mosfet geleid dan ligt de spoel aan massa en wordt er een magnetisch veld in opgebouwd.
2. De mosfet stopt met geleiden dan wilt de spoel zijn magnetisch veld behouden .

Maar dit gaat niet dus zijn magnetisch vel zakt in en er gaat een groote spannings piek onstaan. Die de diode doorlaat en zo de condensator oplaad.

Zo krijg je dus een hogere spanning

De formulle voor de voendings spanning te berkenen is:

Vout =Vin/(1-Vdiodedrop)

2 en 3) hierin vinden we onze analooge en digitaale blokken waar onze componenten in kunnen definieeren

4 ) hierin vinden we systeem eigen componenten. Hieronder verstaan we bv onze watch dog timer , en sleep timer en nog veel meer

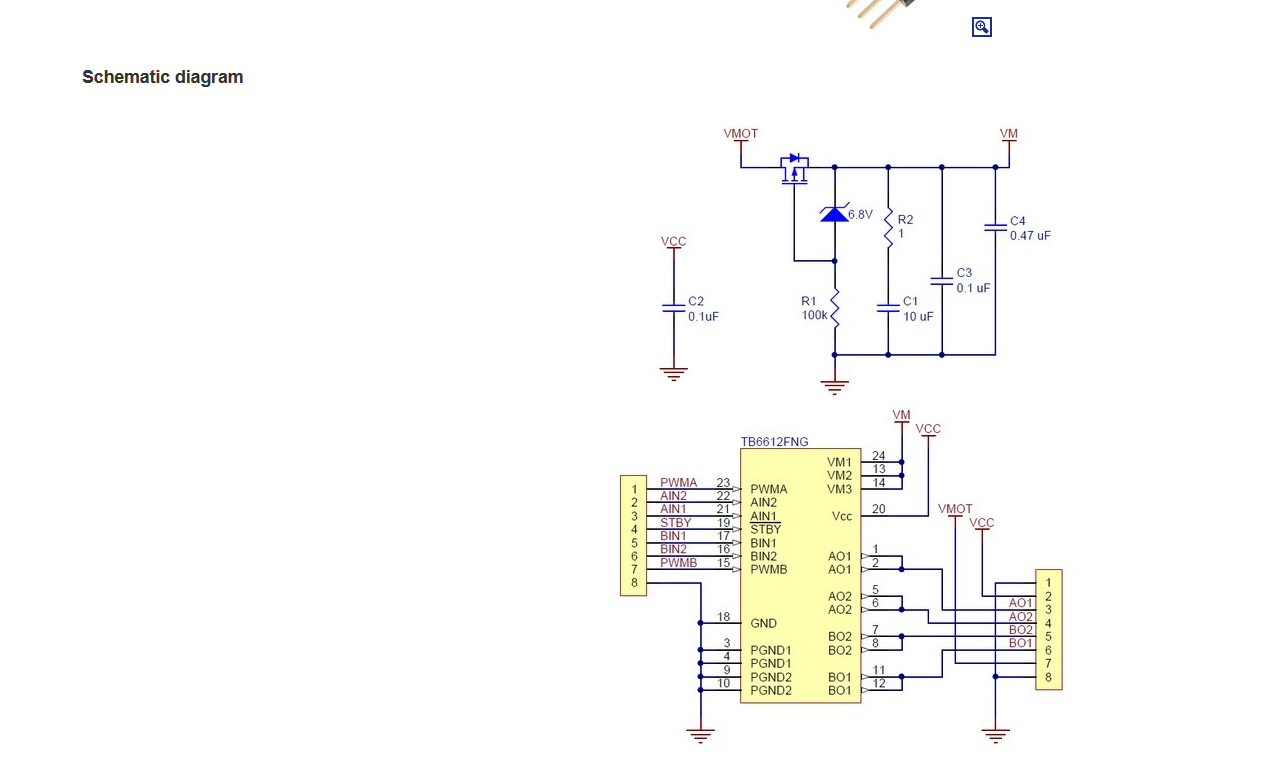
11 de motoren

Als motoren gebruik 2 brushed dc motoren wat is nu een brush dc motor. Een brushed dc motor is een motor dat bestaat uit een rotor en een stator . de rotor bestaat uit spoelen van lakdraad en de stator is voorzien van permanente magneten .Als we nu een spanning aanleggen aan de aansluitklemmen van

|  |  |
| --- | --- |
| **Gelijkstroommotor rotatie** | |
| [Electric motor cycle 1.png](https://commons.wikimedia.org/wiki/File:Electric_motor_cycle_1.png) | Wanneer er een gelijkstroom loopt door de draaiende wikkeling of ankerwikkeling, wordt er in het anker (ook rotor genoemd) een magneetveld opgewekt. De linkerzijde van de rotor wordt weggedrukt van de vaste linker magneet en aangetrokken door de vaste rechtermagneet, waardoor de rotor begint te draaien. |
| [Electric motor cycle 2.png](https://commons.wikimedia.org/wiki/File:Electric_motor_cycle_2.png) | De rotor begint 180° te draaien. |
| [Electric motor cycle 3.png](https://commons.wikimedia.org/wiki/File:Electric_motor_cycle_3.png) | Wanneer de rotor horizontaal aangekomen is bij de juiste polen zal de beweging stoppen. De enige manier om de rotor verder door te doen draaien is het omkeren van de polen op het draaibaar anker. Dit gebeurt door de commutator. Deze keert de richting van de stroom door de spoelen om waardoor het magneetveld in de rotor of anker genaamd omkeert. De linkerzijde van de rotor wordt weggedrukt van de vaste linkermagneet en aangetrokken door de vaste rechtermagneet, waardoor de rotor, ook anker genoemd, verder gaat draaien. |
| [Electric motor.gif](https://commons.wikimedia.org/wiki/File:Electric_motor.gif) | Dit proces herhaalt zich steeds opnieuw. |

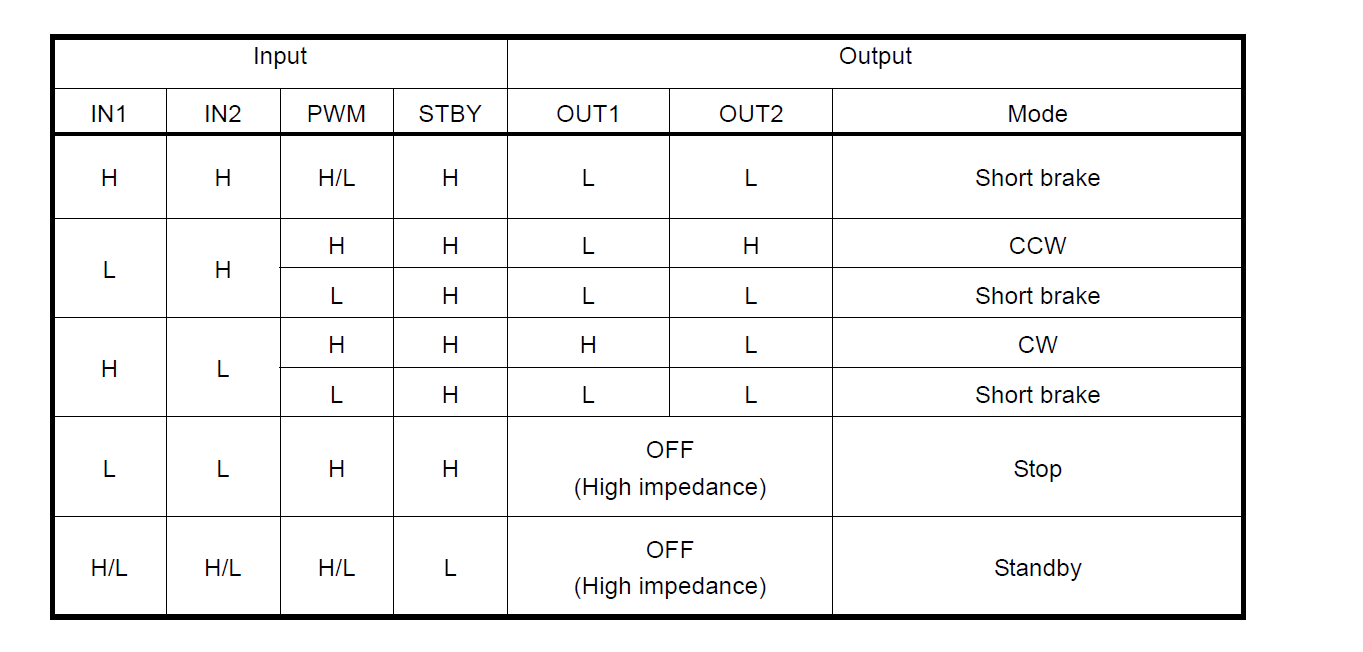
Om deze 2 motoren aan te sturen gebruik ik een motordriver van Polulu motordriver Break out Board.

Op dit break-out board is de TB6612FNG gemonteerd van Toshiba samen met nog wat randcomponenten .



Hierboven ziet u het schema van de gebruikte motordriver. Dit board heeft eveneens een ompoling bescherming. Dit ziet u heelemaal van boven .Als de spanning juist is aangelegt zal de mosfet in gelijding gaan en zo er de motor driver voeden. Indien deze fout is aangesloten zal de mosfet en de zener diode sperren.

Om de motoren in bepaalde richting te laten moeten wen AIN1 En AIN@ evenals BIN1 en BIN 2 volgens de onderstaande tabel aansturen.



11. code uitleg

11.1 Pulsbreete meeting

void Timer2CaptureISR(void)// new function for motorcontroll2

{

if(FlagsElevator & FALLING\_EDGE\_ELEVATOR)//

{

// Read the count on negative edge

CaptureNegEdgeElevator = Timer2\_wReadCompareValue();

// Change the capture to positive edge and clear the FALLING\_EDGE flag

Timer2\_FUNC\_LSB\_REG &= ~0x80;

FlagsElevator &= ~FALLING\_EDGE\_ELEVATOR;// clearing faling edge bit in flags

// Calculate the pulswidth by finding difference between positive edge

// and negative edge counts. As both the numbers are unsigned numbers

// the result will be correct even if there is an underflow in the counter

// The result will be accurate as long as the total pulsewidth is less than

// 65535 timer ticks.

PulseWidthElevator = CapturePosEdgeElevator - CaptureNegEdgeElevator;

// Set the Data available flag

FlagsElevator |= DATA\_AVAILABLE\_ELEVATOR; // BEN: set the correct bit on or off

}

else

{

// Read the count on positive edge

CapturePosEdgeElevator = Timer2\_wReadCompareValue();

// Change the capture to negative edge and set flag

Timer2\_FUNC\_LSB\_REG |= 0x80;

FlagsElevator |= FALLING\_EDGE\_ELEVATOR;

}

}

Hierboven ziet u de code die ik gebruik om mijn pulsbreedtes meet . Dit doe ik drie maal . Dit voor men 2 controle orgaanen en 1 keer voor de echo pin op mijn ultrasoon sensor .

Het princiepe van pulsbreete meeting op basis van interupts rust op het feit dat we onze timer compare value moeten uitlezen op de falling edge en de rising edge . dit doen we zo in onze code : CaptureNegEdgeElevator = Timer2\_wReadCompareValue();

Nu we de tijd weten waneer onze falling edge en onze risinge edgje is gebeurt kunnen we de lengte van onze puls bepalen door de falling edge ( het grootste getal) van de rising edge (kleinste getal) af te trekken .

Als we dit niet doen krijgen we een negatief getal waar we niet mee zijn . En in onze code doen we dit zo:

PulseWidthElevator = CapturePosEdgeElevator - CaptureNegEdgeElevator;

11.2 normaliseeren van de pulsbreete

Voor ik dit stukje code begin uit te leggen ga ik het waarom vertellen van deze code en wat normaliseren is .

Onder normaliseren verstaan we een range van waardes terug brenngen naar een reange van 1 tot -1.

Dit heb ik gedaan zodat ik mijn snelheid maar met 1 variable kan veranderen namelijk MAX\_POWER. //

BOOL Within(WORD value, WORD border, WORD margin)

{

WORD lower = border - margin;

WORD upper = border + margin;

if (value >= lower

&& value <= upper)

return TRUE;

else

return FALSE;

}

Hierboven ziet u de code waarin ik bepaal in welke mijn controle orgaan is geduwd. Dit zal later gebruikt worden om mij controle orgaanen te normaliseren

// normalization function for Aileron (Timer1)

float EvaluateAileron(DWORD value)

{

// Check if pulsewidth data is available

if(FlagsAileron & DATA\_AVAILABLE\_AILERON)

{

#if (DEBUG\_LCD)

LCD\_Position(0,0);

LCD\_PrHexInt(value);

#endif

// stick in center

if (Within(value, CENTER\_AILERON, MARGIN\_AILERON))

{

#if (DEBUG\_LCD)

LCD\_Position(0,5);

LCD\_PrCString("C");

#endif

return 0;

}

else if (value < CENTER\_AILERON) // stick left

{

#if (DEBUG\_LCD)

LCD\_Position(0,5);

LCD\_PrCString("L");

#endif

return ((float)value - CENTER\_AILERON) / (float)(MAX\_AILERON - CENTER\_AILERON);

}

else if (value > CENTER\_AILERON) // stick right

{

#if (DEBUG\_LCD)

LCD\_Position(0,5);

LCD\_PrCString("R");

#endif

return -(CENTER\_AILERON - (float)value) / (float)(CENTER\_AILERON -MIN\_AILERON);

}

// action finished, clear flag to avoid doing it again

FlagsAileron &= ~DATA\_AVAILABLE\_AILERON;

}

return 0;

}

De code in deze functie word uitgevoerd als er data beschikaar is . Dit kunnen we zin aan de FlagsAileron & DATA\_AVAILABLE\_AILERON . als dit is voldaan dan gaan we kijken waar ons controole orgaan zich bevindt. Is dit links of rechts is ( in het andere geval is het boven of onder ),als het recht is dan geven we de genormaliseerde waarde terug van -1 tot 0(0 is controole orgaan in het midden). Als het controle orgaan naar links word bewogen geven we de genormaliseerde waarten tussen 0 en 1 terug.

11.3 het trigeren van de ultrasoon sensor

PRT1DR |= 0x10;// setting p1[4]

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

PRT1DR &= ~0x10;// clearing P1[4]

Dit stukje code gebruik ik om mijn ultrasoon sensor te triggern en is vrij simpel. Eeerst maak ik pin P1[4] hoog en dan wacht ik de voorgeschrven 10us en daarna maak ik deze terug laag.

11.4 het evalueeren van mijn ultrasoon sensor

float EvaluateUltrasoonSensor(void)//(Timer3)

{

if (FlagUltrasoon & DATA\_AVAILABLE\_ULTRASOON)// do if databit is set

{

WORD pulseWidthUltrasoon = PulseWidthUltrasoon;

#if (DEBUG\_LCD)

LCD\_Position(1,12);

LCD\_PrHexInt(pulseWidthUltrasoon);

#endif

FlagUltrasoon &= ~DATA\_AVAILABLE\_ULTRASOON;

// als meting gebeurt is trigger sensor opnieuw en we disablen timer1

TriggerUltrasoon();

return pulseWidthUltrasoon;

}

return MIN\_SAFE\_DISTANCE + 1;

}

In bovenstaan stukje code evalueer ik mijn ultrsoonsensor . en werkt in groote lijnen het zelfde als het normaliseeren van mijn controleorgaanen. Alleen verschilt het hier in dat ik de pulsbreete van mijn ultrasoon sensor terug gef als er iets gededecteerd is . En als er niets in vergrooot ik de veilige afstand

11.5 Main

void main(void)

{

// Enable Global Interrupt

M8C\_EnableGInt;

// Clear the flags

FlagsElevator = 0;

FlagsAileron = 0;// new for motorcontroll2

FlagUltrasoon = 0;

// Start timers and enable interrupt

Timer1\_Start();

Timer1\_EnableInt();

Timer2\_Start();// new for motorcontroll2

Timer2\_EnableInt();// new for motorcontroll2

Timer3\_Start();

Timer3\_EnableInt();

TriggerUltrasoon();

// Init motors

PWM1\_Start();

PWM2\_Start();

#if (DEBUG\_LCD)

LCD\_Start();

#endif

while (TRUE)

{

float aileronNormalized,

elevatorNormalized;

float distance;

float speed, direction;

float motorLeft, motorRight;

BOOL forward;

aileronNormalized = EvaluateAileron(PulseWidthAileron);

direction = fabs(aileronNormalized);

elevatorNormalized = EvaluateElevator(PulseWidthElevator);

speed = fabs(elevatorNormalized);

forward = (elevatorNormalized >= 0);

distance = EvaluateUltrasoonSensor();

if (distance < MIN\_SAFE\_DISTANCE)

{

if (forward)

speed = 0;

}

motorLeft = speed; // default is straight forward

motorRight = speed;

if (aileronNormalized < 0) // turning left

{

motorRight = speed;

motorLeft = speed \* (1 - direction);

}

else if (aileronNormalized > 0) // turning right

{

motorLeft = speed;

motorRight = speed \* (1 - direction);

}

if (forward)

{

// ccw

PRT1DR |= 0x08; // AIN1hoog

PRT1DR &= ~0x02; // AIN2 laag

PRT1DR |= 0x20; // BIN1 hoog

PRT1DR &= ~0x80; // BIN2 laag

}

else

{

//cw

PRT1DR &= ~0x08; // AIN1 laag

PRT1DR |= 0x02; // AIN2 hoog

PRT1DR &= ~0x20; // BIN1 laag

PRT1DR |= 0x80; // BIN2 hoog

}

// Denormalize to Engine

motorLeft \*= (MAX\_POWER - MIN\_POWER);

motorLeft += MIN\_POWER;

motorRight \*= (MAX\_POWER - MIN\_POWER);

motorRight += MIN\_POWER;

PWM1\_WritePulseWidth(motorLeft);

PWM2\_WritePulseWidth(motorRight);

#if (DEBUG\_LCD)

LCD\_Position(0,7);

LCD\_PrHexInt(motorLeft);

LCD\_Position(1,7);

LCD\_PrHexInt(motorRight);

LCD\_Position(0,12);

LCD\_PrCString(forward ? "F" : "B");

#endif

}

}

In dit stukje code ndenormaliseer ik mijn pulsbreete values en stuur ik ze naar de motor .Dit doe ik hier : // Denormalize to Engine

motorLeft \*= (MAX\_POWER - MIN\_POWER);

motorLeft += MIN\_POWER;

motorRight \*= (MAX\_POWER - MIN\_POWER);

motorRight += MIN\_POWER;

en eveneens stuur ik mijn motor driver aan zodat mijn motoren in de goede richting draaien aleen op voorwaarden dat mijn richting vooruit of is dan voer ik dit stukje code uit

// ccw

PRT1DR |= 0x08; // AIN1hoog

PRT1DR &= ~0x02; // AIN2 laag

PRT1DR |= 0x20; // BIN1 hoog

PRT1DR &= ~0x80; // BIN2 laag

En in alle andere gevallen(achteruit) voer ik dit stukje code uit

//cw

PRT1DR &= ~0x08; // AIN1 laag

PRT1DR |= 0x02; // AIN2 hoog

PRT1DR &= ~0x20; // BIN1 laag

PRT1DR |= 0x80; // BIN2 hoog

Eveneens activeer ik hier alle interupts in en alle gebruikte modulles dit doe ik hier :

// Start timers and enable interrupt

Timer1\_Start();

Timer1\_EnableInt();

Timer2\_Start();// new for motorcontroll2

Timer2\_EnableInt();// new for motorcontroll2

Timer3\_Start();

Timer3\_EnableInt();

TriggerUltrasoon();

// Init motors

PWM1\_Start();

PWM2\_Start();

Om een botsing te vermijden diet dit stukje code :

if (distance < MIN\_SAFE\_DISTANCE)

{

if (forward)

speed = 0;

}

Als de afstand kleiner is dan de minimum veiligheids afstand dat zetten we de variable snelheid op o

Om te draaien dien deze code:

if (aileronNormalized < 0) // turning left

{

motorRight = speed;

motorLeft = speed \* (1 - direction);

}

Hierin zeggen we dana ls het controle orgaan naar links wordt gebduwd de rechtse motor zijn snelheid blijft behouden en de linkse motor gelijdelijk aan naar 0 aan het gaan is. En de afstand is de absolute waarde die we terug krijgen uit onze controleorganen functies . dit doen we omdat we met -1 niet hebben als het op richting aankomt. Dit geld ook voor naar rechts te kunnen draaien aleen behoud de linker motor zijn snelheid en gaat de rechter motor gelijdelijk aan naar 0

12. Besluit

Het project was een zeer leerlijk project met hier en daar wat ups en downs die ik overwonnen heb. Evemeens bvondt ik het leuk om eens iets te ontwikelen en wat te leren over motordrivers en ultrasoon sensoren

13.Bijlagen

13.1. Code listing

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// FILENAME: main.c

// Made for project 2 in the second semester of school

//

// DESCRIPTION: Main file of the Psoc\_robot project.

//

//-----------------------------------------------------------------------------

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

////\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//------------------------------------------------------------------------------

// Name: Psoc\_robot

//------------------------------------------------------------------------------

//------------------------------------------------------------------------------

// For Device: CY8C29466

//------------------------------------------------------------------------------

// Required Software: PSoC Designer 5.3

// Project Settings:

//

// Power Setting [Vcc/SysClk Freq]: 5.0V/24MHz

// CPU Clock: Sysclk/1

// VC1 Sysclk/12

// VC2 VC1/2 = Sysclk/24

//

// Timer1: ` For reading one joystick channel aileron puls with

// Clock VC2

// Period 65535

// CompareValue 0

// CompareType Less than or equal to

// Conected to port Port\_0\_0

// Timer2: ` For reading one joystick channel elevation puls with 1

// Clock VC2

// Period 65535

// CompareValue 0

// CompareType Less than or equal to

// Conected to port Port\_0\_1

//

// Timer3: ` For reading the ultrasonic sensor

// Clock VC2

// Period 65535

// CompareValue 0

// CompareType Less than or equal to

// Conected to port Port\_0\_2

//

//

// Pwm conected to port Port\_0\_4

// PWM: For controlling one motor on one side

// Clock VC2

// Enable High

// Period 1000

// PulseWidth 0

// CompareType Less than

// Interrupt Type Terminal count

// LCD:

// LCDPort Port\_2

//

//---------------------------------------------------------------------------------

#include <m8c.h> // part specific constants and macros

#include <math.h>

#include "PSoCAPI.h" // PSoC API definitions for all User Modules

#define DEBUG\_LCD 1

//volatile BOOL ultrasoonInitDone;//dit gedaan om compiler te verplichten waarde terug in te lezen (Caching tegen te gaan )

#define MAX\_POWER 1000

#define MIN\_POWER 0

//

#define DATA\_AVAILABLE\_AILERON 0x01//staat in de eerste bit van de flag

#define FALLING\_EDGE\_AILERON 0x02 // staat in de 2de bit van de flag

WORD CapturePosEdgeAileron;

WORD CaptureNegEdgeAileron;

WORD PulseWidthAileron;

BYTE FlagsAileron;

#define MARGIN\_AILERON 2

#define MIN\_AILERON 0x60

#define CENTER\_AILERON 0x8C

#define MAX\_AILERON 0xB9

//

#define DATA\_AVAILABLE\_ELEVATOR 0x01 // new for motorcontroll2

#define FALLING\_EDGE\_ELEVATOR 0x02 // new for motorcontroll2

WORD CapturePosEdgeElevator;

WORD CaptureNegEdgeElevator;

WORD PulseWidthElevator;

BYTE FlagsElevator;

#define MARGIN\_ELEVATOR 2

#define MIN\_ELEVATOR 0x60

#define CENTER\_ELEVATOR 0x8C

#define MAX\_ELEVATOR 0xB9

// for timer 3 and ultrasoon sensor 1

#define DATA\_AVAILABLE\_ULTRASOON 0x01

#define FALLING\_EDGE\_ULTRASOON 0x02

#define MIN\_SAFE\_DISTANCE 0x50

WORD CapturePosEdgeUltrasoon;

WORD CaptureNegEdgeUltrasoon;

WORD PulseWidthUltrasoon;

BYTE FlagUltrasoon;

void ultrasoonSensor(void);//long ultrasoon sensor(void);

#pragma interrupt\_handler Timer1CaptureISR

#pragma interrupt\_handler Timer2CaptureISR

#pragma interrupt\_handler Timer3CaptureISR

BOOL Within(WORD value, WORD border, WORD margin)

{

WORD lower = border - margin;

WORD upper = border + margin;

if (value >= lower

&& value <= upper)

return TRUE;

else

return FALSE;

}

// normalization function for Aileron (Timer1)

float EvaluateAileron(DWORD value)

{

// Check if pulsewidth data is available

if(FlagsAileron & DATA\_AVAILABLE\_AILERON)

{

#if (DEBUG\_LCD)

LCD\_Position(0,0);

LCD\_PrHexInt(value);

#endif

// stick in center

if (Within(value, CENTER\_AILERON, MARGIN\_AILERON))

{

#if (DEBUG\_LCD)

LCD\_Position(0,5);

LCD\_PrCString("C");

#endif

return 0;

}

else if (value < CENTER\_AILERON) // stick left

{

#if (DEBUG\_LCD)

LCD\_Position(0,5);

LCD\_PrCString("L");

#endif

return ((float)value - CENTER\_AILERON) / (float)(MAX\_AILERON - CENTER\_AILERON);

}

else if (value > CENTER\_AILERON) // stick right

{

#if (DEBUG\_LCD)

LCD\_Position(0,5);

LCD\_PrCString("R");

#endif

return -(CENTER\_AILERON - (float)value) / (float)(CENTER\_AILERON - MIN\_AILERON);

}

// action finished, clear flag to avoid doing it again

FlagsAileron &= ~DATA\_AVAILABLE\_AILERON;

}

return 0;

}

// normalization function for Elevator(Timer2)

float EvaluateElevator(DWORD value)

{

// Check if pulsewidth data is available

if(FlagsElevator & DATA\_AVAILABLE\_ELEVATOR)

{

#if (DEBUG\_LCD)

LCD\_Position(1,0);

LCD\_PrHexInt(value);

#endif

// stick in center

if (Within(value, CENTER\_ELEVATOR, MARGIN\_ELEVATOR))

{

#if (DEBUG\_LCD)

LCD\_Position(1,5);

LCD\_PrCString("C");

#endif

return 0;

}

else if (value > CENTER\_ELEVATOR) // stick up

{

#if (DEBUG\_LCD)

LCD\_Position(1,5);

LCD\_PrCString("U");

#endif

return ((float)value - CENTER\_ELEVATOR) / (float)(MAX\_ELEVATOR - CENTER\_ELEVATOR);

}

else if (value < CENTER\_ELEVATOR) // stick down

{

#if (DEBUG\_LCD)

LCD\_Position(1,5);

LCD\_PrCString("D");

#endif

return -(CENTER\_ELEVATOR - (float)value) / (float)(CENTER\_ELEVATOR - MIN\_ELEVATOR);

}

// action finished, clear flag to avoid doing it again

FlagsElevator &= ~DATA\_AVAILABLE\_ELEVATOR;

}

return 0;

}

void TriggerUltrasoon(void)

{

PRT1DR |= 0x10;// setting p1[4]

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

asm("nop");

PRT1DR &= ~0x10;// clearing P1[4]

}

float EvaluateUltrasoonSensor(void)//(Timer3)

{

if (FlagUltrasoon & DATA\_AVAILABLE\_ULTRASOON)// do if databit is set

{

WORD pulseWidthUltrasoon = PulseWidthUltrasoon;

#if (DEBUG\_LCD)

LCD\_Position(1,12);

LCD\_PrHexInt(pulseWidthUltrasoon);

#endif

FlagUltrasoon &= ~DATA\_AVAILABLE\_ULTRASOON;

// als meting gebeurt is trigger sensor opnieuw en we disablen timer1

TriggerUltrasoon();

return pulseWidthUltrasoon;

}

return MIN\_SAFE\_DISTANCE + 1;

}

void main(void)

{

// Enable Global Interrupt

M8C\_EnableGInt;

// Clear the flags

FlagsElevator = 0;

FlagsAileron = 0;// new for motorcontroll2

FlagUltrasoon = 0;

// Start timers and enable interrupt

Timer1\_Start();

Timer1\_EnableInt();

Timer2\_Start();// new for motorcontroll2

Timer2\_EnableInt();// new for motorcontroll2

Timer3\_Start();

Timer3\_EnableInt();

TriggerUltrasoon();

// Init motors

PWM1\_Start();

PWM2\_Start();

#if (DEBUG\_LCD)

LCD\_Start();

#endif

while (TRUE)

{

float aileronNormalized,

elevatorNormalized;

float distance;

float speed, direction;

float motorLeft, motorRight;

BOOL forward;

aileronNormalized = EvaluateAileron(PulseWidthAileron);

direction = fabs(aileronNormalized);

elevatorNormalized = EvaluateElevator(PulseWidthElevator);

speed = fabs(elevatorNormalized);

forward = (elevatorNormalized >= 0);

distance = EvaluateUltrasoonSensor();

if (distance < MIN\_SAFE\_DISTANCE)

{

if (forward)

speed = 0;

}

motorLeft = speed; // default is straight forward

motorRight = speed;

if (aileronNormalized < 0) // turning left

{

motorRight = speed;

motorLeft = speed \* (1 - direction);

}

else if (aileronNormalized > 0) // turning right

{

motorLeft = speed;

motorRight = speed \* (1 - direction);

}

if (forward)

{

// ccw

PRT1DR |= 0x08; // AIN1

PRT1DR &= ~0x02; // AIN2

PRT1DR |= 0x20; // BIN1

PRT1DR &= ~0x80; // BIN2

}

else

{

//cw

PRT1DR &= ~0x08; // AIN1

PRT1DR |= 0x02; // AIN2

PRT1DR &= ~0x20; // BIN1

PRT1DR |= 0x80; // BIN2

}

// Denormalize to Engine

motorLeft \*= (MAX\_POWER - MIN\_POWER);

motorLeft += MIN\_POWER;

motorRight \*= (MAX\_POWER - MIN\_POWER);

motorRight += MIN\_POWER;

PWM1\_WritePulseWidth(motorLeft);

PWM2\_WritePulseWidth(motorRight);

#if (DEBUG\_LCD)

LCD\_Position(0,7);

LCD\_PrHexInt(motorLeft);

LCD\_Position(1,7);

LCD\_PrHexInt(motorRight);

LCD\_Position(0,12);

LCD\_PrCString(forward ? "F" : "B");

#endif

}

}

void Timer1CaptureISR(void)

{

if (FlagsAileron & FALLING\_EDGE\_AILERON)

{

// Read the count on negative edge

CaptureNegEdgeAileron = Timer1\_wReadCompareValue();

// Change the capture to positive edge and clear the FALLING\_EDGE flag

Timer1\_FUNC\_LSB\_REG &= ~0x80;

FlagsAileron &= ~FALLING\_EDGE\_AILERON;

// Calculate the pulswidth by finding difference between positive edge

// and negative edge counts. As both the numbers are unsigned numbers

// the result will be correct even if there is an underflow in the counter

// The result will be accurate as long as the total pulsewidth is less than

// 65535 timer ticks.

PulseWidthAileron = CapturePosEdgeAileron - CaptureNegEdgeAileron;

// Set the Data available flag

FlagsAileron |= DATA\_AVAILABLE\_AILERON;

}

else

{

// Read the count on positive edge

CapturePosEdgeAileron = Timer1\_wReadCompareValue();

// Change the capture to negative edge and set flag

Timer1\_FUNC\_LSB\_REG |= 0x80;

FlagsAileron |= FALLING\_EDGE\_AILERON;

}

}

void Timer2CaptureISR(void)// new function for motorcontroll2

{

if(FlagsElevator & FALLING\_EDGE\_ELEVATOR)//

{

// Read the count on negative edge

CaptureNegEdgeElevator = Timer2\_wReadCompareValue();

// Change the capture to positive edge and clear the FALLING\_EDGE flag

Timer2\_FUNC\_LSB\_REG &= ~0x80;

FlagsElevator &= ~FALLING\_EDGE\_ELEVATOR;// clearing faling edge bit in flags

// Calculate the pulswidth by finding difference between positive edge

// and negative edge counts. As both the numbers are unsigned numbers

// the result will be correct even if there is an underflow in the counter

// The result will be accurate as long as the total pulsewidth is less than

// 65535 timer ticks.

PulseWidthElevator = CapturePosEdgeElevator - CaptureNegEdgeElevator;

// Set the Data available flag

FlagsElevator |= DATA\_AVAILABLE\_ELEVATOR; // BEN: set the correct bit on or off

}

else

{

// Read the count on positive edge

CapturePosEdgeElevator = Timer2\_wReadCompareValue();

// Change the capture to negative edge and set flag

Timer2\_FUNC\_LSB\_REG |= 0x80;

FlagsElevator |= FALLING\_EDGE\_ELEVATOR;

}

}

void Timer3CaptureISR(void)

{

if(FlagUltrasoon & FALLING\_EDGE\_ULTRASOON)

{

// Read the count on negative edge

CaptureNegEdgeUltrasoon = Timer3\_wReadCompareValue();

// Change the capture to positive edge and clear the FALLING\_EDGE flag

Timer3\_FUNC\_LSB\_REG &= ~0x80;

FlagUltrasoon &= ~FALLING\_EDGE\_ULTRASOON;

// Calculate the pulswidth by finding difference between positive edge

// and negative edge counts. As both the numbers are unsigned numbers

// the result will be correct even if there is an underflow in the counter

// The result will be accurate as long as the total pulsewidth is less than

// 65535 timer ticks.

PulseWidthUltrasoon = CapturePosEdgeUltrasoon - CaptureNegEdgeUltrasoon;

// Set the Data available flag

FlagUltrasoon |= DATA\_AVAILABLE\_ULTRASOON;

}

else

{

// Read the count on positive edge

CapturePosEdgeUltrasoon = Timer3\_wReadCompareValue();

// Change the capture to negative edge and set flag

Timer3\_FUNC\_LSB\_REG |= 0x80;

FlagUltrasoon |= FALLING\_EDGE\_ULTRASOON;

}

}

13.2 bronvermeldingen

Wikipedia : <https://nl.wikipedia.org/wiki/Gelijkstroommotor>

Rc-servo signaal : <http://www.mitchr.me/SS/batteriesRequired/RCcontrolTheory>

Psoc1 smp : <http://www.cypress.com/knowledge-base-article/increasing-output-power-psoc-1-switch-mode-pump-smp-kba94769>

Psoc gegeven : datasheet cy8c2466. Pdf datasheet