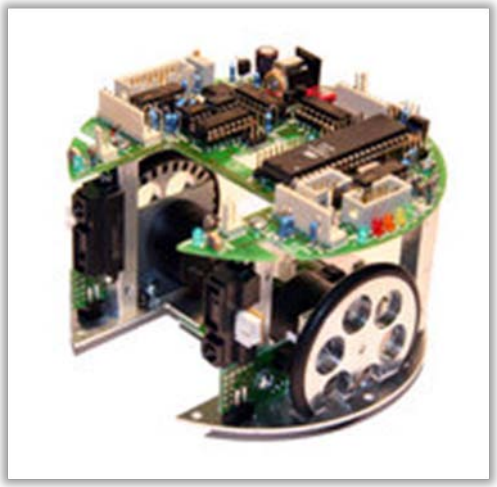
Positionsregelung für CarBoTUC

(car robots of technical university of Clausthal)

In einem Car-robot wird ein runder Magnet mit Querorientierung eingebaut. Ein Sensor Rotationsencoder-AS5040 dient dazu die Änderung des Magnetfelds zu messen.

Die Rotation des Rades wird durch einen magnetischen Senor gemessen. Anschließend werden den Messwerten an das Steuerboard Sparten 3 geliefert. Spartan 3 vergleicht dem Messwerte mit dem Sollwert. Somit kann das Car-Robot mit vordefinierter Geschwindigkeit gesteuert werden. Bei einer Geschwindigkeitsänderung z.B. bei dem Auffahrt oder Abfahrt kann die Fahrgeschwindigkeit nur in sequenzielle Stufen (Schaltung) angepasst werden.



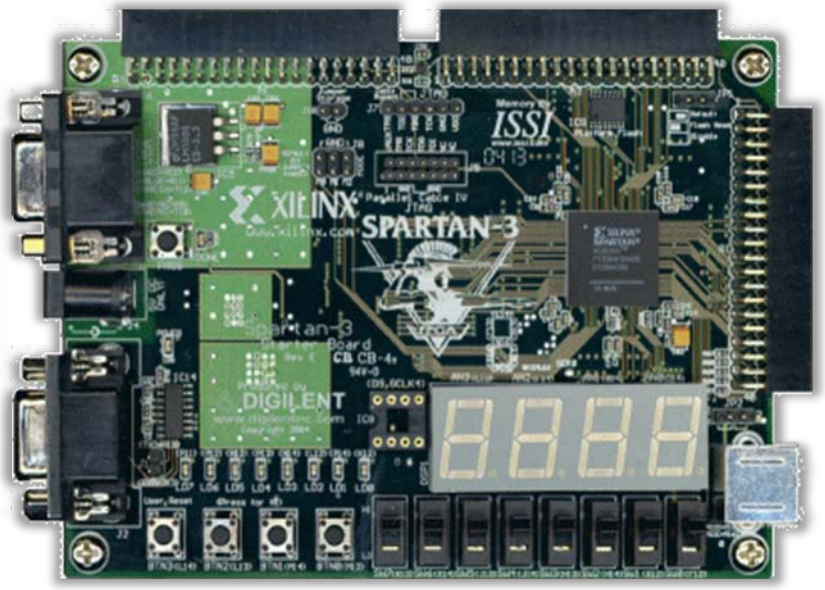
Car-robot 1

Bearbeiter: Dipl.-Ing Hossam Addeen Ramadan und Handong Ma

Inhaltverzeichnis:

Vorbereitung
-Hardware

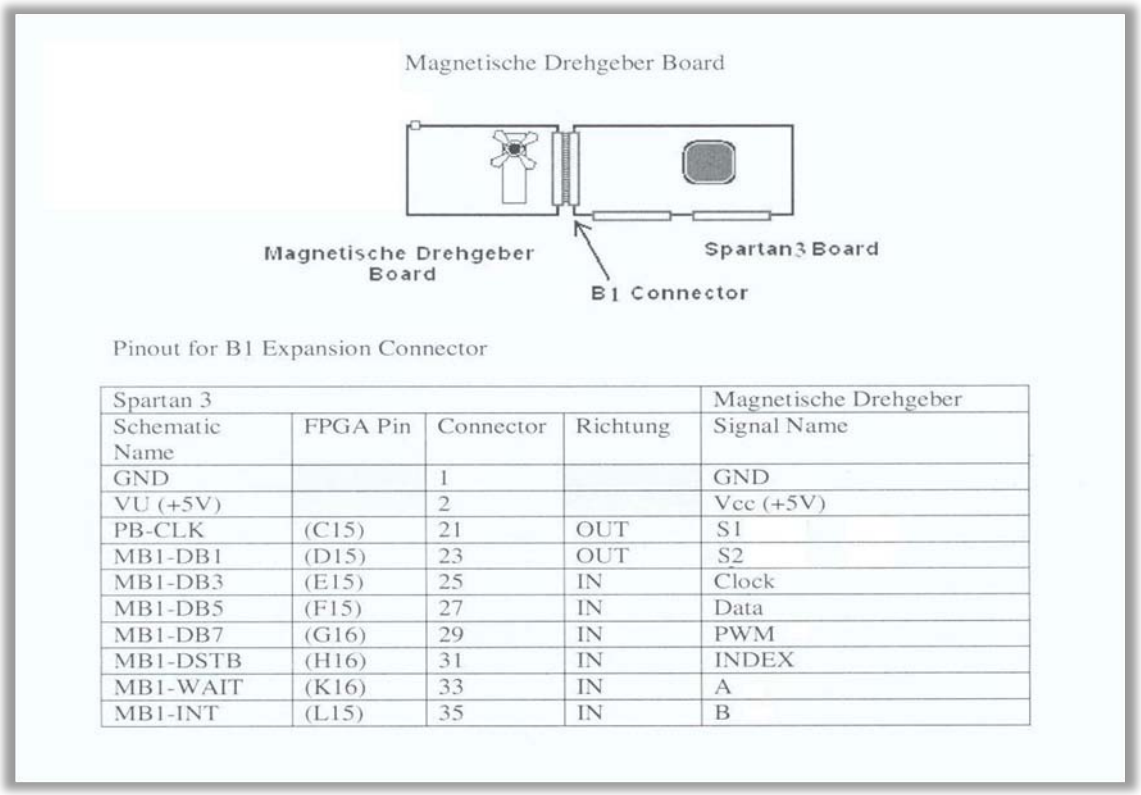
Spartan 3



Spartan 3

In dieser Arbeit wird das Versuchsboard Spartan 3 von Xilinx Inc. eingesetzt. Zwei der vier Buttons werden für die Startfunktion definiert. (Jeweils für eine Rotationsrichtung). Die achte swts werden als verschiedene Geschwindigkeitsstufe festgelegt. Die Daten werden über den I/O Anschluss B1 zwischen Spatan3 und Motor Bauteil ausgetauscht. Das RS-232 Port dient als die Verbindungsanschluss zwischen Spatan3 und Computer. Somit kann das Matlab den Baugruppedaten an das Versuchsboard weitergeben.

Magnetische Drehgeber Board



Pinout for B1 Expansion Connector

Spartan 3				Magnetische Drehgeber
Schematic Name	FPGA Pin	Connector	Richtung	Signal Name
GND		1		GND
VU (+5V)		2		Vcc (+5V)
PB-CLK	(C15)	21	OUT	S1
MB1-DB1	(D15)	23	OUT	S2
MB1-DB3	(E15)	25	IN	Clock
MB1-DB5	(F15)	27	IN	Data
MB1-DB7	(G16)	29	IN	PWM
MB1-DSTB	(H16)	31	IN	INDEX
MB1-WAIT	(K16)	33	IN	A
MB1-INT	(L15)	35	IN	B

--- Magnetische Drehgeber Board

Magnetische Drehgeber Board: Das Board wird von Dipl.-Ing Hossam Addeen Ramadan entwickelt. Darauf werden Rotationsencoder AS5040 und ein Motor mit einem Magnet eingebaut, damit wird die Messung der Rotation eines echten Rades simuliert. Der jeweilige Anschluss wird im obigen Bild verdeutlichen.

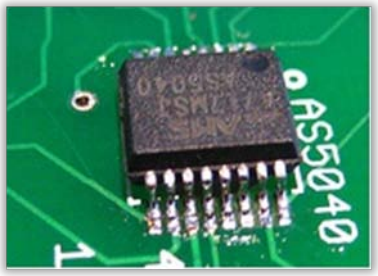
Die wichtigsten Anschlüsse, die in dieser Arbeit eingesetzt werden sind PWM(G16),S1(C15),S2(D15),A(K16) und B(L15).

Erklärung von Anschluss S1 und S2:

Die Anschlüsse S1 und S2 sind direkt mit dem Motor verbunden. Im Ruhezustand sind beide Anschlüsse mit dem Wert „0“ initialisiert, d.h. der Motor hat in diesem Zustand keine Bewegung. Wenn der Wert des Anschlusses S1 auf „1“ gesetzt wird, dreht der Motor auf einer vordefinierten Richtung. Im Gegensatz zum S1 wenn der Wert von S2 „1“ ist, dreht der Motor auf der anderen Richtung. Somit wird die Rotationsrichtung des Rades gesteuert. Zusätzlich kann die Geschwindigkeit durch die Spannung auf der Anschlüsse S1 und S2 gesteuert werden. Je höher die Spannung ist, desto schneller dreht der Motor. Die Höhe der Spannung wird in der Arbeit durch

einen 8-Bits Code repräsentiert. Der Code „11111111“ steht für die höchsten Spannung. Der Code „000000000“ bedeutet, dass die Spannung auf null liegt. Der Motor fängt erst dann zu drehen, wenn die Torsion, die durch die Spannung erzeugt wird, größer ist als die innere Widerstandkraft des Motors. Z.B. Bei dem eingesetzt Versuchsboard fängt der Motor erst mit Spannung „00001111“ zu drehen.

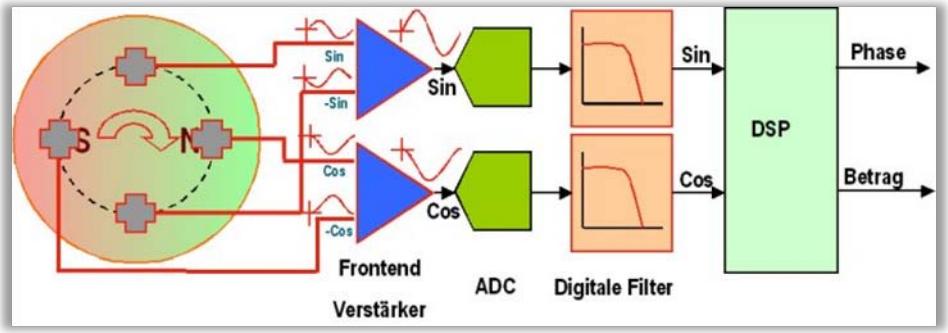
Rotationsencoder AS5040



Rotationsencoder AS5040

Rotationsencoder AS5040,von AUSTRIAMICROSYSTEMS herstellt. Der AS5040 kann absolute Winkelpositionen an einer Achse messen. Die Messung wird indirekt über ein Magnetfeld mittels Hallsensoren durchgeführt. Aufgrund der Anordnung der Hallsensoren und der Integration der Sensoren direkt in Silizium wird eine sehr hohe Genauigkeit bei der Messung erreicht. Das Messprinzip kompensiert außerdem Sportgrößen, wie externes Magnetfeld, Alterungseinflüsse, Temperaturschwankungen und mechanische Toleranzen.

Die komplette Auswertung der Sensorsignale geschieht auf dem Chip.

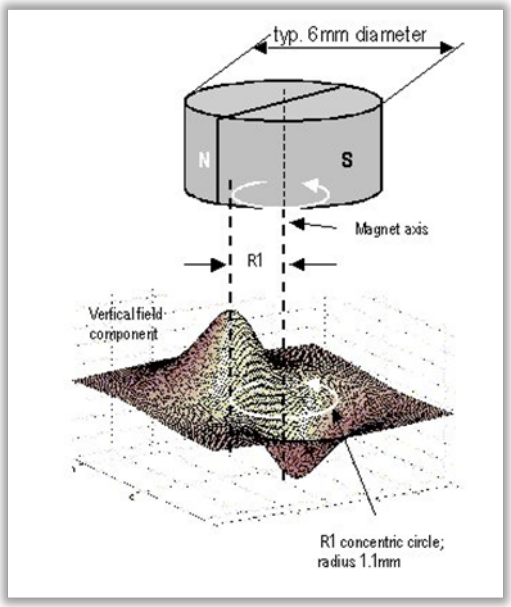


Durch die hohe Integration reduziert sich die zusätzliche Beschaltung des

Chips auf wenige Bauteile, beispielsweise ein paar Filterkondensatoren.

Die Winkelencoderfamilie von austriamicrosystems besitzt vielfältige Schnittstellen, um den Winkelwert zu übertragen

Serial SSI	Standard Schnittstelle für Winkelencoder in der Industrie
PWM	erzeugt abhängig vom Winkel eine entsprechende Pulslänge, lässt sich sehr schön mit der Capturefunktion eines Controllers messen und braucht nur eine Leitung
I2C	direkte Schnittstelle zum Microcontroller
Analog	kompatible Schnittstelle um beispielsweise Potiapplikationen zu ersetzen
Inkrementell	klassische Schnittstelle um relative Bewegungen zu messen, keine Absolutmessung möglich
BLDC	erzeugt direkt die Kommutierung für einen bürstenlosen Motor. Durch die höhere Auflösung gegenüber einer 3-Hall-Schalterlösung kann der BLDC mit einem wesentlich höheren Drehmoment gestartet werden

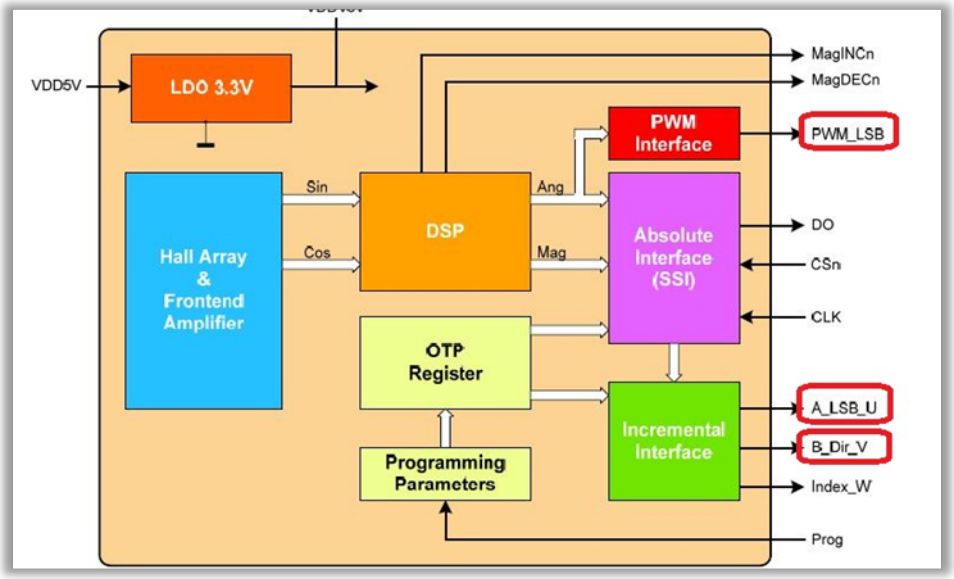


Bei der Auswahl des Magnets ist darauf zu achten, dass sogenannte Selten-Erde-Magnete verwendet werden.

Diese erzeugen ein besonders starkes Magnetfeld, das im 1-Tesla-Bereich liegt. Im Zentrum des Magnetfelds ist ein linearer Bereich, der für die Genauigkeit der Messung ausschlaggebend ist. Solange die Hallsensoren in diesem Bereich liegen, kann eine unkalibrierte Genauigkeit von +/- 0.5 Grad gewährleistet werden. Weiterhin ist zu beachten, dass der Magnet nicht direkt auf eine Eisenwelle montiert wird. Die Eisenwelle verursacht quasi einen magnetischen 'Kurzschluss' und entzieht damit den Hallsensoren das Magnetfeld. Idealerweise sollte eine NE-Welle verwendet werden, oder wenn nicht anders machbar muss eine Isolation aus NE-Material zwischen Eisenwelle und Magnet eingefügt werden.

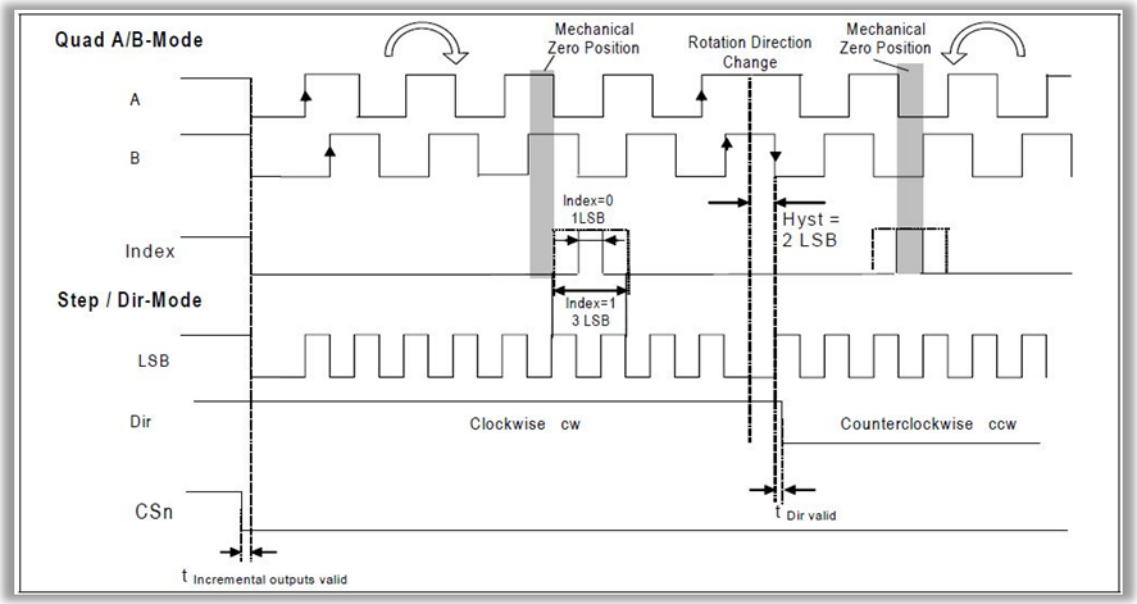
Wie in den letzten Abschnitt eingeführt wird, wird ein Magnet oberhalb des Rotationsencoder-AS5040 eingebaut. Der Motor wird auf der anderen Seit des Magnetes geklebt, so dass die Rotation des Magnet durch die Drehung des Motors erfolgt. Somit kann das Rotationsencoder-AS5040 die Drehung des Motors anhand der Änderung des Magnetfelds messen.

Erklärung von Anschluss A,B und PWM:



In dem obigen Bild werden die wichtigste Anschlüsse, die in dieser Arbeit auf Einsatz gekommen sind, gezeigt. Die Anschlüsse PWM_LSB,A_LSB_U und B_Dir_V werden in dieser Arbeit mit den Abkürzung PWM,A und B genannt.

Channel A und B:

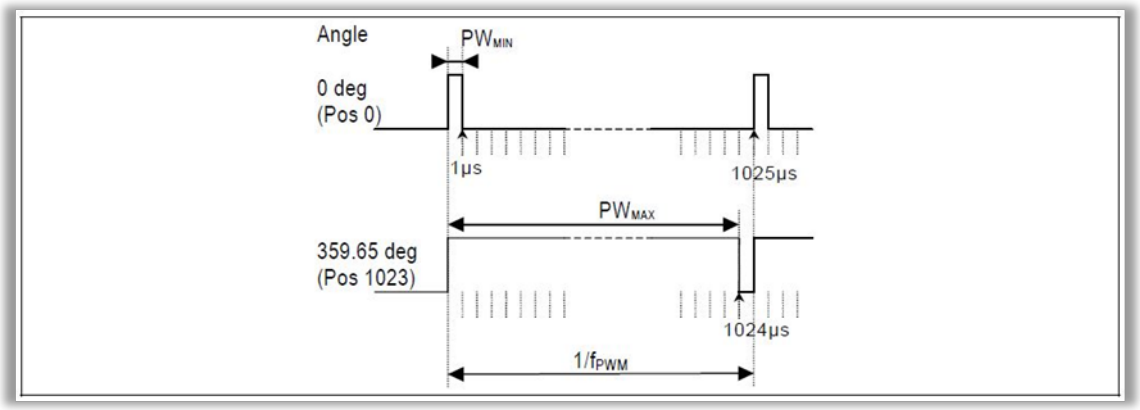


Die Drehrichtung und Drehwinkel können durch Channel A und B wie folgendes definiert werden:

Drehrichtung: In obiger Abbildung wird das Signal von A und B gezeigt. Wenn der Magnet im Uhrzeigersinn dreht (von oben nach unten betrachten), ist das Signal von Channel A 90 Grade schneller als das Signal aus Channel B und umgekehrt.

Drehwinkel (bzw. Drehgeschwindigkeit) : Da das Signal per Periode nur eine steigende Flanke ausgibt, kann man durch die Anzahl der steigenden Flanke die Drehwinkel innhalb eines Zeitraums bestimmen. Die Drehgeschwindigkeit erfolgt dann durch die Division der Drehwinkel mit der verbrauchten Zeit.

Channel PWM:



Das Signal vom PWM Anschluss gibt auch an, wieviel Grade der Magnet gedreht hat. Die Periode des Signals ist 1025 µs. Jede µs kann das Signal den Wert „0“ oder „1“ ausgeben(siehe obige Abbildung). Durch den Anteil des Wertes „1“ in der gesamten Periode kann die Drehwinkel des Magnetes bestimmt werden. Z.B. bei dem nullte Grad wird nur einen Wert mit „1“ ausgegeben. Bei der 359,65 Grad wird dagegen 1024-mal „1“ von dem Anschluss ausgelesen (Lesenhäufigkeit) Mit diesem Anschluss kann eine Genauigkeit von ±5% erreichen werden.

-Programm(VHDL)

--Datenübertragung zwischen Bauteil und Computer:(RS232)

Der Empfang der Daten über die serielle Schnittstelle erfolgt in Anlehnung an die entsprechende VHDL-Laborübung . Die verwendete Struktur besteht aus einem Start-Bit , 8 Daten-Bits , keiner Parität und mindestens einem Stopp-Bit .

Um Matlab mehrere Information von Motor zu erfahren, wird bei der

Absendungsfunktion eine Datenlänge von 32 Bits festgelegt, d.h. 4 mal 8 Bits Daten. Die Steuerungsbefehler von Matlab an dem Motor braucht ledig 8 Bits. Die ersten 5 Bits werden für bestimmte Funktionen reserviert. Die letzte drei Bits gibt die Geschwindigkeitsstufe (von „xxxxx000“ bis „xxxxx101“) an.

--Definition der Geschwindigkeitstufe mit Anschluss S1 und S2:(Geschwindigkeit)

Wie vorher erwähnt wurde, werden die Anschlüsse S1 und S2 direkt mit dem Motor angeschlossen. Daher kann die Geschwindigkeitstufe durch die beiden Anschlüsse definiert werden. Wenn der Strom durch S1 an dem Motor geliefert wird, dreht der Motor auf einer bestimmen Richtung und umgekehrt. Die Geschwindigkeitstufe wird durch die Höhe der Spannung bestimmt. In dieser Arbeit wird die Höhe der Spannung durch ein 8-Bit Befehl definiert. Die Anzahl der Wert „1“ in diesem Befehle entspricht die Höhe der Spannung. Auf dieser Art und Weise werden insgesamt 8 Stufen von Geschwindigkeit definiert, nämlich „1111 1111“ , „1111 1110“ , „1111 1100“ , „1111 1000“ , „1111 0000“ , „1110 0000“ , „1100 0000“ , „1000 0000“. Die Kombination „0000 0000“ entspricht den Stopzustand.

In der Realität arbeitet der Motorsteuerung jedoch nicht mit 8 Bits. Der Chip der Motorsteuerung bekommt jedem Takt ein Bit als Steuersignal. Falls das Bit eine eins ist, wird eine vordefinierte Spannung an dem Motor weitergegeben. Wenn das Bit eine null ist, wird in diesem Takt keinen Strom an dem Motor geliefert. Erst nach acht Takten wird eine Geschwindigkeitsstufe (8 Bits) durch den Steuerchip vollständig durchgesetzt. Wie oft der Steuerchip das Signal ausliest ist dann entscheidende für die Geschwindigkeitsregulierung. Eine viel zu hohe Frequenz (z.B. 5 MHz) führt dazu, dass selbst wenn nur „1000 000“ eingegeben wird, arbeitet der Motor fast immer unter dem vordefinierten Spannung, nämlich mit voller Leistung drehen. Im Gegensatz dazu, wenn die Frequenz zu niedrig (1 Hz) wäre, selbst unter Geschwindigkeitsstufe „1111 1111“, bekommt der Motor fast wie gar keinen Strom. Nach mehreren Experimenten wird eine Frequenz von ca. 97 KHz (mclk_new(10)) in dieser Arbeit ausgewählt. Unter dieser Frequenz arbeitet der Motor sehr ruhig. Die Geschwindigkeitsstufen können auch klar sich voneinander unterschieden werden.

----------Programm Teil----------

```
process(mclk_new(10))
begin
    if mclk_new(10)'event and mclk_new(10)='1' then
----------00000001----------1
        if Richtung_s1='1' and Geschwindigkeit_stufe="00000001" then
```

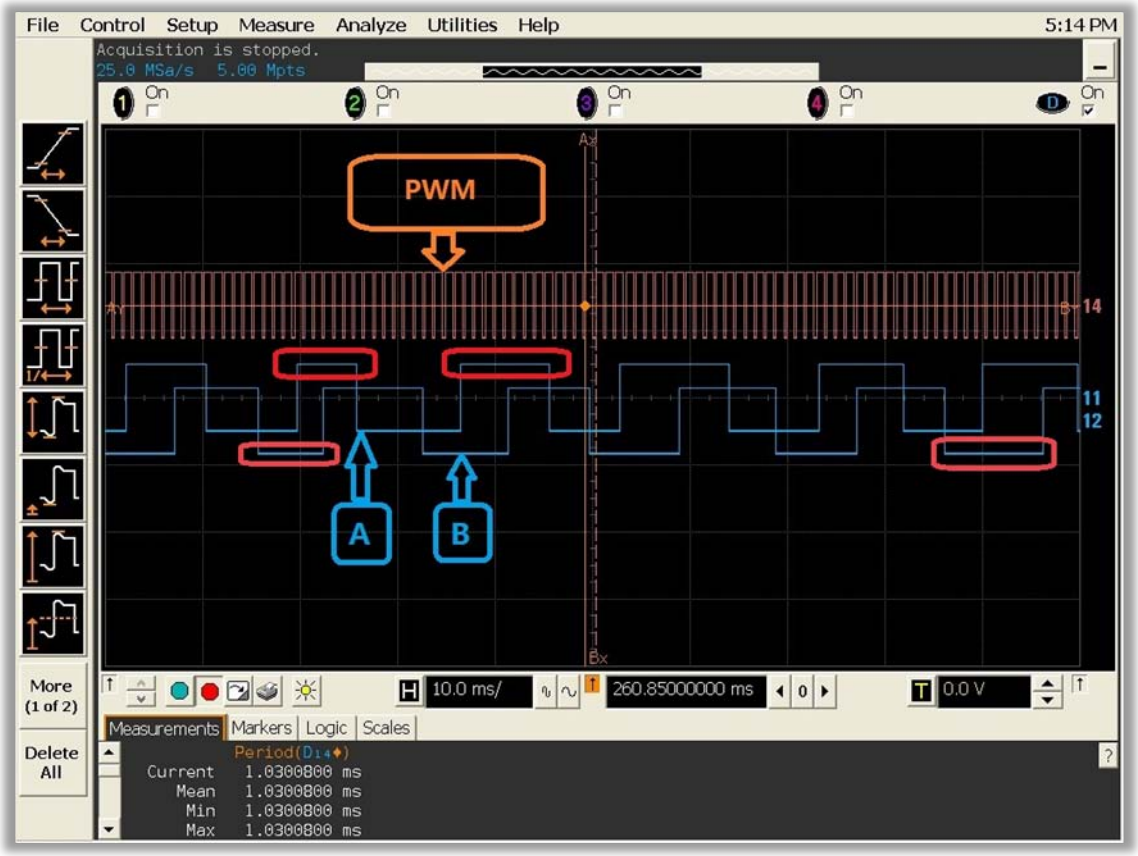
```
                case status is
                    when 0 to 7 =>
                        S1<='0';
                        status<=status + 1;
                    end case;
            elsif Richtung_s2='1' and Geschwindigkeit_stufe="00000001" then
                case status is
                    when 0 to 7 =>
                        S2<='0';
                        status<=status + 1;
                    end case;
----------------------------------------00000010-------------------------------------2
            2…….8
----------------------------------------10000000-------------------------------------8
            else
            S1<='0';
            S2<='0';

            end if;
        end if;
    end process;

-----------------------------------end Programm Teil---------------------------------
```

--Messen(nicht mit A und B,sondern mit PWM):(MotorMessen)

In diesem Abschnitt wird die aktuelle Ist-Geschwindigkeit des Motors gemessen. Die Geschwindigkeit des Motors wird durch Drehwinkelgeschwindigkeit gerechnet. Die kann wiederum von Anschluss A, B und PWM ausgelesen werden.

Warum nicht mit Signal A und B .

Mit den Anschlüsse A und B können die Drehwinkel und die Drehrichtung bestimmt werden. Die Drehrichtung wird aber in diesem Experiment durch Steuerungsignal eingegeben, daher muss sie nicht mehr wieder von dem Motor ausgelesen werden. Die Messergebnisse mit Anschlüsse A und B sind jedoch sehr instabil. Dies führt zu einer großen Abweichung bei dem Vergleich der Ist- und Soll-Geschwindigkeit. In folgende Abbildung werden ein Vergleich zwischen Messergebnis aus Anschlüsse A, B und PWM zusammengestellt.



Signal A , B , PWM on Oszilloskop 1

Signal A , B , PWM on Oszilloskop 2

Es wird durch die Abbildung deutlich, dass die Signale von Anschlüsse A (die blaue Linie, Signal 12 in Abbildung) und B (die blaue Linie, Signal 11 in Abbildung) bei der Messung im Vergleich zum dem Signal von Anschluss PWM (Signal A , B , PWM on Oszilloskop 1) wesentlich instabil sind. Deswegen werden Anschluss A und B nicht für die Messung der Geschwindigkeit eingesetzt.

Messung mit Signal PWM

Wegen der Instabilität des Signals von Anschluss A und B wird in diesem Abschnitt das Signal vom Anschluss PWM analysiert. Das Ziel ist zu überprüfen, ob das Signal von Anschluss PWM eine genügende Stabilität für die Messerung der Motorgeschwindigkeit liefert.

In der Abbildung *Signal A , B , PWM on Oszilloskop 3* wird Teil der Abbildung *Signal A , B , PWM on Oszilloskop 1* vergrößert. Es kann deutlich erkannt werden, dass das PWM-Signal im Vergleich zum dem Signal aus Anschluss A und B wesentlich stabiler ist. Daher wird in dieser Arbeit das PWM-Signal für die Messung der Motorgeschwindigkeit benutzt.

Der PWM liefert pro. µs ein Signal. Um dieses Signal genau zu lesen muss einen passenden Takt dafür ausgewählt werden. Da 1 µs 10^{-6} s entspricht, wird hier eine Frequenz von 1 MHz ausgewählt.

Obwohl der Anschluss PWM ein wesentlich stabileres Signal liefert, muss die Geschwindigkeit nach der Hersteller Angabe mit der folgenden Formel

$$\textbf{Position} = \frac{t_{on} \times 1025}{t_{on} + t_{off}} - 1$$

„t_{on}“: die Anzahl der ausgelieferte „1“;

„t_{off}“: die Anzahl der ausgelieferte „0“;

Berechnet werden. Somit können einige Fehler von der Bearbeitung des Signals ausgeschlossen werden.

----------Programm Teil----------

--1us=10 hoch -6 <=> 10 hoch 6 Hz => 1M Hz for PWM

```
--mclk= 50M Hz
    if mclk_in'event and mclk_in='1' then
----------------------------1MHz signal erzeugen-------------------------
        if ClkVT=24 then
            ClkVT:=0;
            Baude_CLK<=not Baude_CLK;
        else
            ClkVT:=ClkVT+1;
        end if;
    end if;
end process;
--------------------------------PWM messen-------------------------------
process(Baude_CLK)
begin
    if Baude_CLK'event and Baude_CLK = '1' then
            if PWM_in='1' then
            zaehler_1<=zaehler_1 + '1';
                    if zaehler_0/= "0000000000000000" then
                        PWM_Max_0 <= zaehler_0;
                        zaehler_0 <= "0000000000000000";
                    end if;
            else
            zaehler_0<=zaehler_0 + '1';
                    if zaehler_1 /= "0000000000000000" then
                        PWM_Max_1 <= zaehler_1;
                        zaehler_1 <= "0000000000000000";
                    end if;
            end if;
    end if;
end process;

---------------------------------end Programm Teil----------------------------
```

<u>--Messenergebnis umrechnen, um Irrtum zu verkleinern:(Rechnen)</u>

Die Bearbeitung der Daten aus Anschluss PWM erfolgt mit der folgenden Formel. Um möglich wenige Daten zwischen Matlab und Steuerchip auszutauschen wird eine zusätzliche Divisionsfunktion in VHDL implementiert. Somit kann die Position des Motors direkt von dem Steuerchip ausgelesen werden.

$$\textbf{Position} = \frac{t_{on} \times 1025}{t_{on} + t_{off}} - 1$$

$$= \frac{t_{on} \times 1025 - t_{on} - t_{off}}{t_{on} + t_{off}}$$

$$= \frac{t_{on} \times 1024 - t_{off}}{t_{on} + t_{off}}$$

.

----------Programm Teil----------

```
process(mclk_in)
variable Q,erg:std_logic_vector(15 downto 0);
variable status:integer range 0 to 7:=0;
begin
if mclk_in'event and mclk_in='1' then
    Vor_op1<="00000000010000000000";
    op1<=PWM_Max_1*Vor_op1 - PWM_Max_0;
    op2<="00000"&(PWM_Max_1+PWM_Max_0);
    case status is
    when 0 =>
        if op1>op2 then
        status:=1;
        elsif op1=op2 then
        status:=2;
        elsif op1 < op2 then
        status:=3;
        end if;
    when 1 =>
        if op1 >= op2 then
        op1<=op1 - op2;
        Q:=Q + 1;
        status:=status;
        else
        erg:=Q;
        Q:="0000000000000000";
        status:=0;
        end if;
    when 2 =>
        erg:="0000000000000001";
        status:=0;
    when 3 =>
            erg:="0000000000000000";
        status:=0;
    when others =>
```

```
            status:=0;
        end case;
end if;
PWM_out<=erg;
end process;
```

----------ende Programm Teil----------

--umformen der gemessenen Wert , Iswertgeschwindigkeit herausfinden : (IsWertGeschwindigkeit)

Nach der Bearbeitung der Daten aus dem Anschluss PWM kann festgestellt werden, dass das Ergebnis in einem Intervall zwischen 0 und 1024 liegt. Je nach auf welcher Richtung der Motor dreht, wird das Ergebnis sukzessive von 0 auf 1024 vergrößert oder von 1024 auf 0 verkleinert. Da der Fokus der Arbeit liegt an der Regelung der Drehgeschwindigkeit, spielt die Drehrichtung des Motors an der Stelle keinen Rolle. Daher wird in dieser Arbeit eine beliebige Drehrichtung betrachtet, um die Geschwindigkeitsregelung zu realisieren.

Nach der Bearbeitung der Daten von Anschluss PWM kann jetzt die Position des Motors in jedem Zeitpunkt bestimmt werden. Wenn der Motor unter einer konstanten Geschwindigkeit dreht, ist der Veränderungsrat der Position des Motors auch konstant. Eine Erhöhung der Geschwindigkeit führt auch dazu, dass der Motor seine Position schneller ändert. In dieser Arbeit wird die Änderung der Position des Motors in einem bestimmten Zeitintervall gemessen. Je größer die Änderung der Position aufweist, desto schneller dreht der Motor. Daher kann die Geschwindigkeit des Motors indirekt durch die Änderung der Position des Motors repräsentiert werden.

.

Nach der Bearbeitung der PWM Ausgabe bekommt man eine 16 stellige Binärzahl. Um die Genauigkeit der Datenmessung zu garantieren und die Fehlern bei der Messung möglichst auszuschießen werden nur die dreizehnte, vierzehnte und fünfzehnte stelle bei der Berechnung der Geschwindigkeit betrachtet. Um die Geschwindigkeit zu berechnen werden zusätzlich noch zwei Zähler (Z1 und Z2) eingesetzt. Wenn die drei Bits von „000“ aus „111“ erreicht, wird Z1 um „1“ erhöht. Nachdem Z1 „50 000 000“ erreicht, wird Z2 um „1“ erhöht und den Wert von Z1 zurückgesetzt. Als folgendes wird die Geschwindigkeitsstufe des Motors anhand des Wertes Z2 definiert. (siehe folgende Tabelle ohne Toleranz)

Fahrgeschwindigkeitstufe	**Z2(Hexadezimal)**
0	0

1	8
2	F
3	14
4	19
5	25

Da der Motor gewisse Ungenauigkeit von den physikalischen Eigenschaften, wie. z. B. schwankende Temperatur, interne und externe Reibungskraft, Aufbau des magnetischen Drehgeber Boards usw., leidet, wird für jede Geschwindigkeitsstufe eine bestimmte Toleranz erwiesen. Nach mehreren Experimenten wird die Toleranz jeweiliger Geschwindigkeitsstufe wie folgendes festgelegt: (siehe folgende Tabelle mit Toleranz)

Fahrgeschwindigkeitstufe	**Z2(Hexadezimal)**
0	0
1	7,8,9
2	E,F,10
3	13,14,15
4	18,19,1A
5	24,25,26

Der Vorgang des obigen entwickelten Algorithmus wird mit folgende Programmecode implementiert:

```
--------------------------------Anfang Programm Teil-------------------------------
Signal zaehler,zaehler1:std_logic_vector(7 downto 0);
signal status:integer range 0 to 7:=0;
signal status1:integer range 0 to 50000000:=0;
begin
process(mclk_in)
begin
if mclk_in'event and mclk_in='1' then
            case status is

  ---------------------------------------------------------------------------------0
              when 0 =>
                  if PWM_out(3 downto 1)="000" then
                      status<=status + 1;
                  else
                      status<=status;
                  end if;

  ---------------------------------------------------------------------------------1
```

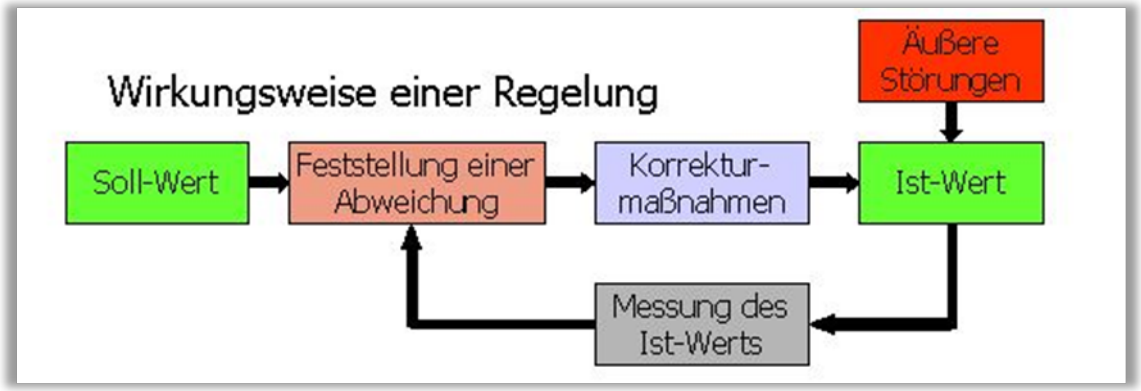
```
                when 1 =>
                    if PWM_out(3 downto 1)="001" then
                        status<=status + 1;
                    else
                        status<=status;
                    end if;

-------------------------------------------------------------------------------2
                when 2 =>
                    if PWM_out(3 downto 1)="010" then
                        status<=status + 1;
                    else
                        status<=status;
                    end if;

-------------------------------------------------------------------------------3
                when 3 =>
                    if PWM_out(3 downto 1)="011" then
                        status<=status + 1;
                    else
                        status<=status;
                    end if;

-------------------------------------------------------------------------------4
                when 4 =>
                    if PWM_out(3 downto 1)="100" then
                        status<=status + 1;
                    else
                        status<=status;
                    end if;

-------------------------------------------------------------------------------5
                when 5 =>
                    if PWM_out(3 downto 1)="101" then
                        status<=status + 1;
                    else
                        status<=status;
                    end if;

-------------------------------------------------------------------------------6
                when 6 =>
                    if PWM_out(3 downto 1)="110" then
                        status<=status + 1;
                    else
```

```
                    status<=status;
                end if;

    ---------------------------------------------------------------------------------7
                when 7 =>
                    if PWM_out(3 downto 1)="111" then
                        status<=status + 1;
                        zaehler<= zaehler+'1';
                    else
                        status<=status;
                    end if;
                end case;
    ---------------------------------------------------------------------------------
                case status1 is
                when 0 =>
                zaehler<="00000000";
                status1<=status1 + 1;
                when 1 to 49999999 =>
                status1<=status1 + 1;
                when 50000000=>
                zaehler1<=zaehler;
                status1<=status1 + 1;
                end case;
        ------------------------------------stufe0(0)-----------------------------
    if zaehler1="00000000" then
        IsWert_Geschwindigkeit<="00000000";
        ------------------------------------stufe1(7,8,9)-----------------------------
    elsif zaehler1="00000111" or zaehler1="00001000" or
zaehler1="00001001" then
        IsWert_Geschwindigkeit<="00000001";
        ------------------------------------stufe2(e,f,10)-----------------------------
    elsif zaehler1="00001110" or zaehler1="00001111" or
zaehler1="00010000" then
        IsWert_Geschwindigkeit<="00000010";
        ------------------------------------stufe3(13,14,15)-----------------------------
    elsif zaehler1="00010011" or zaehler1="00010100" or
zaehler1="00010101" then
        IsWert_Geschwindigkeit<="00000011";
        ------------------------------------stufe4(18,19,1a)-----------------------------
    elsif zaehler1="00011000" or zaehler1="00011001" or
zaehler1="00011010" then
        IsWert_Geschwindigkeit<="00000100";
        ------------------------------------stufe5(24,25,26)-----------------------------
    elsif zaehler1="00100100" or zaehler1="00100101" or
```

```
zaehler1="00100110" then
        IsWert_Geschwindigkeit<="00000101";
        ------------------------------------anders(Error)-----------------------------
    else
        IsWert_Geschwindigkeit<="00001110";
    end if;
end if;
end process;
IsWert_Geschwindigkeit_PWM_genau<=zaehler1;

------------------------------Ende Programm Teil-----------------------------
```

--Regelung festlegen, so dass mit einen gewünschte Geschwindigkeit fahren kann : (Regeln)

Die folgende Abbildung veranschaulicht die Vorgehensweise der Geschwindigkeitsregelung.



Regelung 1

Wie in der Abbildung gezeigt wird, wird der „Soll-Geschwindigkeit“ von Matlab in dem Programm eingegeben. Der „Ist-Geschwindigkeit“ wird mit dem im letzten Abschnitt entwickelten Verfahren vom Motor ausgelesen. Die „Äußere Störungen“ sind hier hauptsächlich die Rückwirkung des unebenen Bodens auf die Fahreigenschaft wie z.B. Bergsteigen und Talfahren.

Die Abweichung erfolgt durch ein direkt Vergleich der Ist-Geschwindigkeit und Soll-Geschwindigkeit. Wenn die Ist-Geschwindigkeit größer als die Soll-Geschwindigkeit ist, wird der Motor untergeregelt und umgekehrt. Bei der Anpassung der Geschwindigkeit wird die Geschwindigkeit Stufe für Stufe nach Oben oder Unten geregelt. Nach jeder Anpassung wird eine aktuelle Abweichung beurteilt. Dieser Prozess wird solange wiederholt, bis der Motor seine Soll-Geschwindigkeit erreicht.

```
-------------------------------------Programm Teil---------------------------------
process(mclk_in)
begin
    if mclk_in'event and mclk_in='1' then
    --------------vergleichen-----------------------------
    if SollWert_Geschwindigkeit(7) = '1'    then
--IsWert_Geschwindigkeit="00001110",1110=E,means Error;
    if IsWert_Geschwindigkeit/="00001110" and IsWert_Geschwindigkeit(2
downto 0) > SollWert_Geschwindigkeit(2 downto 0) then
        KorrigierendWert(6 downto 0)<= IsWert_Geschwindigkeit(6 downto 0)
- '1';
        KorrigierendWert(7)<='1';
    elsif IsWert_Geschwindigkeit/="00001110" and IsWert_Geschwindigkeit(2
downto 0) < SollWert_Geschwindigkeit(2 downto 0) then
        KorrigierendWert(6 downto 0)<= IsWert_Geschwindigkeit(6 downto 0)
+ '1';
        KorrigierendWert(7)<='1';
    elsif IsWert_Geschwindigkeit(2 downto 0) = SollWert_Geschwindigkeit(2
downto 0) then
        KorrigierendWert(6 downto 0)<= IsWert_Geschwindigkeit(6 downto 0);
        KorrigierendWert(7)<='0';
    end if;
else
KorrigierendWert<="00000000";
end if;
    ------------------end vergleichen-----------------------
        Q<=Q + '1';
    ----------------position zu zahl umwanden-------------
        -----------5----------------
        if KorrigierendWert(6 downto 0) = "0000101" then
        KorrigierendWert_um<="10000000";
        ----------4----------------
        elsif KorrigierendWert(6 downto 0) = "0000100" then
        KorrigierendWert_um<="01000000";
        -----------3--------------
        elsif KorrigierendWert(6 downto 0) = "0000011" then
        KorrigierendWert_um<="00100000";
        ------------2--------------
        elsif KorrigierendWert(6 downto 0) = "0000010" then
        KorrigierendWert_um<="00010000";
```

```
          ----------1----------------
          elsif KorrigierendWert(6 downto 0) = "0000001" then
          KorrigierendWert_um<="00001000";
          else
          KorrigierendWert_um<="00000000";
          end if;
     ------------------------------------------------------------------
               Fahr_stufe <= KorrigierendWert_um;
     end if;
end process;
```

----------Ende Programm Teil----------

-Fazit.

Die Arbeit befasst sich um die Regelung der Fahrgeschwindigkeit eines Fahrzeuges. Es wird versucht das Fahrzeug unabhängig von der Bodensturktur auf einer konstanten Geschwindigkeit zu kontrollieren. In dieser Arbeit wird ein Magnetische Drehgeber Board eingesetzt um die Drehung der Räder eines Fahrzeuges zu simulieren. Durch der Onboard AS-5040 Chip wird die Drehung des Motors auf dem Magnetische Drehgeber Board gemessen. Die Messdaten werden anschießende an dem Spatan3 Chip geliefert. Durch den PWM Anschluss des Spartan 3 kann die Geschwindigkeit des Motors ausgelesen. Diese Is-Geschwindigkeit wird mit der Soll-Geschwindigkeit vergleichen. Falls eine Abweichung aufweist, wird eine Kontrollesignal an dem Motor geschickt. Der Motor wird anhand des Kontrollesignal Stufe für Stufe nach Oben oder Unten geregelt bis er die Soll-Geschwindigkeit erreicht.

In der Praxis wird diese Technik oft bei der automatischen Geschwindigkeitsregelung in einem Fahrzeug benutzt. Somit kann der Fahrer dem Fahrzeug mit einer vordefinierten konstanten Geschwindigkeit fahren ohne die Kupplung und das Gaspedal zu bestätigen. Das erleichtet den Fahrer auf einer Lang-Distanz fahrt und erhöht gleichzeitig auch die Fahrkomfort.