

Prof. Dr. Paweł Buczek

4.12.2019

Abgabedatum

11.12.2019

Felix Rehaag

Rami Chaari

4/1/6

Inhalt

[Abbildungsverzeichnis 2](#_Toc26808778)

[Tabellenverzeichnis 2](#_Toc26808779)

[Einleitung 3](#_Toc26808780)

[Laborgeräte 3](#_Toc26808781)

[Aufgabe 1: Treppenverfahren 3](#_Toc26808782)

[Aufbau 3](#_Toc26808783)

[Struktogramm 5](#_Toc26808784)

[C-Programm 5](#_Toc26808785)

[Auswertung 7](#_Toc26808786)

[Aufgabe 3: Interner A/D Umsetzer: Dimmen einer LED unter 8](#_Toc26808787)

[Zuhilfenahme eines analogen XY Joysticks 8](#_Toc26808788)

[Aufbau 8](#_Toc26808789)

[Struktogramm 10](#_Toc26808790)

[C-Programm 10](#_Toc26808791)

[Auswertung 12](#_Toc26808792)

[Fazit 13](#_Toc26808793)

# Abbildungsverzeichnis

[Abbildung 1: Schaltung der Aufgabe 1 4](#_Toc26808794)

[Abbildung 2: Aufbau der Schaltung im Labor 4](#_Toc26808795)

[Abbildung 3: Messverlauf der Spannung am Port K 7](#_Toc26808796)

[Abbildung 4: Aufbau der Schaltung für Aufgabe 3 8](#_Toc26808797)

[Abbildung 5: Aufbau der Schaltung im Labor für Aufgabe 3 9](#_Toc26808798)

[Abbildung 6: PWM-Signal mit 95% High-Pegel 13](#_Toc26808799)

# Tabellenverzeichnis

[Tabelle 1: Übersetzungsverhalten des D/A-Umsetzers 3](#_Toc26808800)

[Tabelle 2: Soll- und Ist-Werte der Messung des Treppenverfahrens 8](#_Toc26808801)

# Einleitung

In diesem Versuch wird ein digital zu analog Umwandler mittels dem TM4C1294 realisiert. Dabei werden jeweils der interne ADC vom Mikrokontroller und ein externer D/A- Umsetzer verwendet.

# Laborgeräte

Folgende Laborgeräte wurden benutzt:

• Mikrocomputer (Tiva TM4C1294)

• Oszilloskop

• BCD 7-Segment Anzeige

• XY Joystick

• D/A –Umsetzer

• True RMS Digital Multimeter 4150 von PeakTech

# Aufgabe 1: Treppenverfahren

Über Port K legt man dual ansteigende Eingangswerte an den D/A-Umsetzer (0000 0000, 0000 0001, ..., 1111 1111. Der Umsetzer wandelt diese Werte gemäß Tabelle 1 in eine treppenförmige Ausgangsspannung Uout. Wenn Uout größer als die zu messende Eingangsspannung UE wird, schaltet der Komparator. Der letzte digitale Eingangswert ist zur Eingangsspannung UE proportional. Die gemessene Spannung ist dreistellig über die Ports L und M auszugeben.

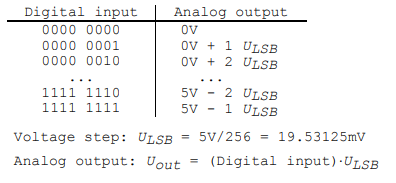


Tabelle 1: Übersetzungsverhalten des D/A-Umsetzers

## Aufbau

Abbildungen 1 und 2 zeigen die Schaltung.

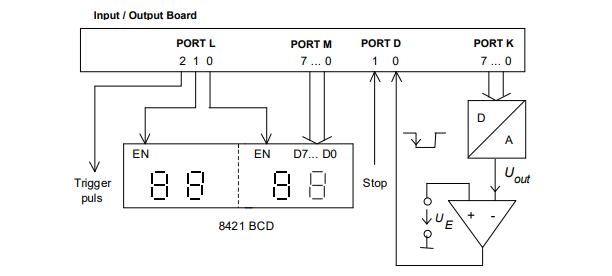


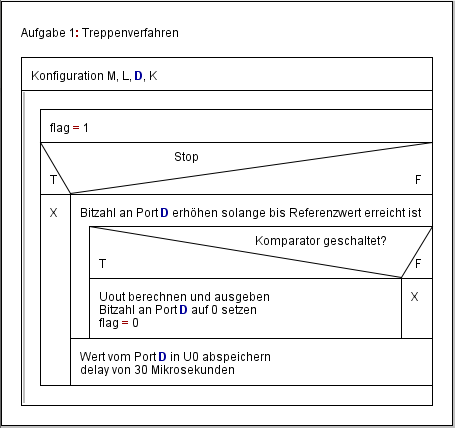
Abbildung 1: Schaltung der Aufgabe 1

Ein Bild, das Elektronik, Schaltkreis enthält.

Automatisch generierte Beschreibung

Abbildung 2: Aufbau der Schaltung im Labor

## Struktogramm



## C-Programm

/\*---------------------------------------------------------------------------------------

 \* Felix Rehaag und Rami Chaari

 \* Aufgabe 1: A/D-Umwandler mittels Treppenverfahren

\* Erstellt mit https://tohtml.com/c/

 \*---------------------------------------------------------------------------------------

\*/

#include "tm4c1294ncpdt.h"

#include "stdio.h"

#define DELAY 30e-6

#define ULSB 19.53125 //mV

**void** timer\_0(**float**);

**void** ausgabe(**int**);

**int** main(**void**)

{ // Port Clock Gating Control

SYSCTL\_RCGCGPIO\_R |= 0x800; // Clock Port M enablen

SYSCTL\_RCGCGPIO\_R |= 0x400; // Clock Port L enablen

SYSCTL\_RCGCGPIO\_R |= 0x200; // Clock Port K enablen

SYSCTL\_RCGCGPIO\_R |= 0x008; // Clock Port D enablen

**while**((SYSCTL\_PRGPIO\_R & 0x800)==0); //Auf Port M warten

**while**((SYSCTL\_PRGPIO\_R & (0x400)) == 0); //auf port L warten

**while**((SYSCTL\_PRGPIO\_R & 0x200)==0); //Auf Port K warten

**while**((SYSCTL\_PRGPIO\_R & (0x008)) == 0); //auf port D warten

//Aktivierung Pins

GPIO\_PORTD\_AHB\_DEN\_R |= 0x03;

GPIO\_PORTK\_DEN\_R |= 0xFF;

GPIO\_PORTL\_DEN\_R |= 0x07;

GPIO\_PORTM\_DEN\_R |= 0xFF;

// Richtungen der Ports

GPIO\_PORTD\_AHB\_DIR\_R &= 0xFC;

GPIO\_PORTK\_DIR\_R |= 0xFF;

GPIO\_PORTL\_DIR\_R |= 0x07;

GPIO\_PORTM\_DIR\_R |= 0xFF;

**float** U0 = 0.0;

**float** Uout = 0.0;

**int** flag = 1;

**while**(1){

flag = 1;

**for**(GPIO\_PORTK\_DATA\_R = 0;((GPIO\_PORTK\_DATA\_R < 255) & (flag));GPIO\_PORTK\_DATA\_R ++){

//Uout = i\*ULSB;

**if**(!(GPIO\_PORTD\_AHB\_DATA\_R & 0x02)){

**if**(GPIO\_PORTD\_AHB\_DATA\_R == 0x00){ //comparator

Uout = U0 \* ULSB;

GPIO\_PORTK\_DATA\_R = 0x00;

flag = 0;

//printf("%lf\n", Uout);

ausgabe(Uout);

}

U0 = GPIO\_PORTK\_DATA\_R; //zwischenspeichern

timer\_0(DELAY);

}

}

//GPIO\_PORTM\_DATA\_R = GPIO\_PORTK\_DATA\_R;

}

}

**void** ausgabe(**int** result){

result = result / 10;

**int** U0 = result % 10; //LSB

result = result / 10;

**int** U1 = result % 10;

result = result / 10;

**int** U2 = result % 10; //MSB

printf("%i.%i%i\n", U2,U1,U0);

GPIO\_PORTL\_DATA\_R = 0x01; //BCD1 enablen

GPIO\_PORTM\_DATA\_R = (U1 << 4) | U0;

GPIO\_PORTL\_DATA\_R = 0x00; //BCD1 disablen

GPIO\_PORTL\_DATA\_R = 0x02; //BCD2 enablen

GPIO\_PORTM\_DATA\_R = (0 << 4) | U2;

GPIO\_PORTL\_DATA\_R = 0x00; //BCD2 disablen

}

**void** timer\_0(**float** periode){

**int** wt=0; // aux. variable for very short time wait

SYSCTL\_RCGCTIMER\_R = 0x00000001; wt++; // clock enable timer

TIMER0\_CTL\_R &= 0xFFFFFFFE; // stop timer 0

TIMER0\_CFG\_R =0x00000000; // timer 0 in 32 bit mode

TIMER0\_TAMR\_R |= 0x02; // timer 0 in periodic mode

TIMER0\_TAMR\_R &= 0xFFFFFFEF; // timer in downward counting mode

TIMER0\_TAILR\_R = (16000-1)\*periode; // start value 0.1 sec =16MHz/1.6Mio

TIMER0\_ICR\_R |= 1; // clear timeout flag of timer 0A

TIMER0\_CTL\_R |= 0x0000001; // start timer 0

**while**(!(TIMER0\_RIS\_R & 0x00000001)); // wait and poll flag for timer 0 timeout

TIMER0\_ICR\_R |= 0x00000001; // clear flag of timer 0

TIMER0\_CTL\_R |= 0x0000001; // restart timer 0

}

## Auswertung

Abbildung 3 zeigt den Messverlauf der Spannung am Port K. Sobald der Komparator schaltet, wird die Ausgangsspannung auf 0 gesetzt.

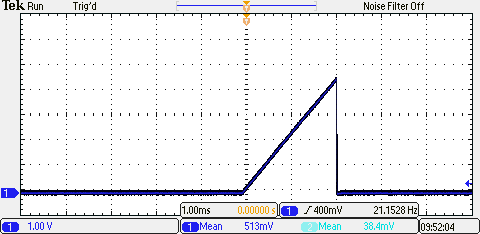


Abbildung 3: Messverlauf der Spannung am Port K

Die Messwerte sind in der Tabelle 2 zusammengefasst. Für die Messung wird das True RMS Digital Multimeter 4150 von PeakTech benutzt.

|  |  |
| --- | --- |
| Multimeteranzeige in V | Digitale Anzeige in mV |
| 0,775 | 78 |
| 0,784 | 80 |
| 2 | 201 |
| 3,437 | 343 |
| 5,173 | 494 |
| 3,766 | 377 |
| 2,084 | 209 |
| 4,867 | 477 |
| 1,349 | 132 |
| 3,487 | 347 |
| 2,988 | 299 |

Tabelle 2: Soll- und Ist-Werte der Messung des Treppenverfahrens

Der Vergleich zwischen Soll- und Ist-Werten zeigt einen maximalen Unterschied von 23,3 mV. Um diesen Unterschied noch kleiner zu machen, muss man das Treppenverfahren mit einer höheren Anzahl von Bits durchführen. Je mehr Bits benutzt werden, desto genauer werden die Messwerte.

# Aufgabe 3: Interner A/D Umsetzer: Dimmen einer LED unter

# Zuhilfenahme eines analogen XY Joysticks

In dieser Aufgabe soll mittels dem internen ADC des Mikrokontrollers ein C-Programm geschrieben werden, dass dem Benutzer ermöglicht die Helligkeit der LED mittels XY-Joysticks zu verändern. Dazu ist folgende Funktionsweise vorgesehen:

* Wird der Joystick nach rechts gedrückt, soll die Helligkeit langsam zunehmen.
* Wird der Joystick nach links gedrückt, soll die Helligkeit langsam abnehmen.
* In mittlerer Ruhestellung bleibt die letzte Helligkeitsstufe erhalten.
* Bei dem kurzzeitigen Drücken des Joystick Tasters, soll abwechselnd die maximale und minimale Helligkeit bleibend sichtbar werden.

## Aufbau

Abbildung 4 und 5 zeigen den Aufbau der Schaltung.

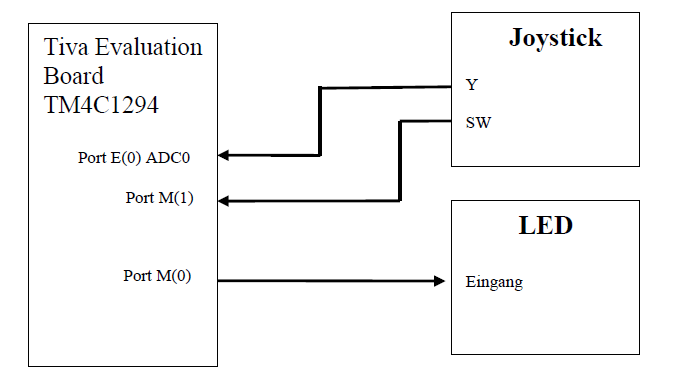


Abbildung 4: Aufbau der Schaltung für Aufgabe 3

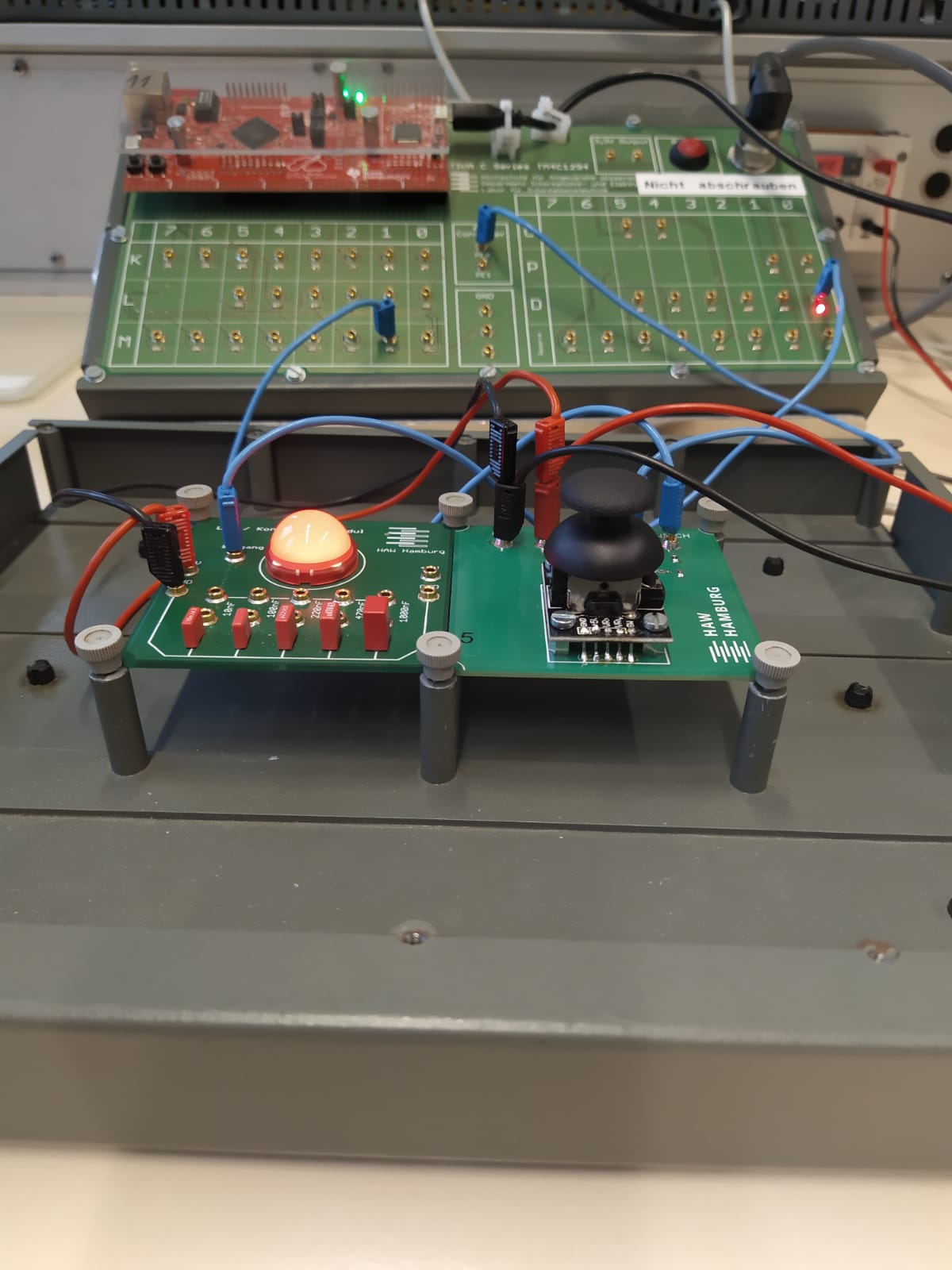
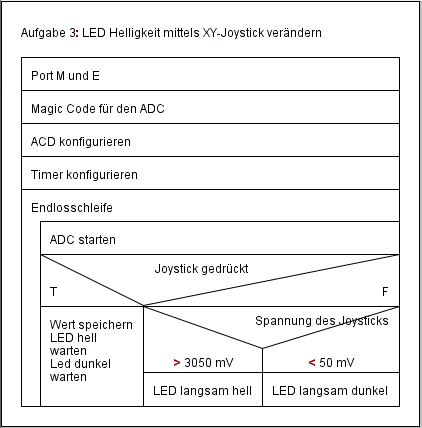


Abbildung 5: Aufbau der Schaltung im Labor für Aufgabe 3

## Struktogramm



## C-Programm

/\*---------------------------------------------------------------------------------------

 \* Felix Rehaag und Rami Chaari

 \* Aufgabe 3: LED-Helligkeit mittels XY-Joysticks verändern

\* Erstellt mit https://tohtml.com/c/

 \*---------------------------------------------------------------------------------------

\*/

#include "tm4c1294ncpdt.h"

#include "stdio.h"

#define ADC\_VREF 3300.0 // voltage on V\_REF+ pin in mV

#define V\_LSB (ADC\_VREF / 4095) // V\_LSB voltage in mV

#define V\_COEFF ((4.70 + 9.137) / 9.137 \* V\_LSB)

#define WAIT 200000

**void** busy (**unsigned** **long**);

**int** main(**void**)

{ **unsigned** **int** ADCoutput, wt;

// Port Clock Gating Control

SYSCTL\_RCGCGPIO\_R |= 0x800; // clock Port M enablen

SYSCTL\_RCGCGPIO\_R |= 0x010; // clock Port E enablen

SYSCTL\_RCGCGPIO\_R |= 0x008;

SYSCTL\_RCGCADC\_R |= 0x1; wt++; // Clock ADC0 enable set

**while**((SYSCTL\_PRGPIO\_R & 0x800)==0); //Auf Port M warten

**while**((SYSCTL\_PRGPIO\_R & 0x010)==0); //Auf Port E warten

**while**((SYSCTL\_PRGPIO\_R & 0x008)==0);

//Aktivierung Pins

// Magic code for start the ADC Clocking

// => see tm4c1294ncpdt Datasheet, 15.3.2.7 ADC Module Clocking

SYSCTL\_PLLFREQ0\_R |= SYSCTL\_PLLFREQ0\_PLLPWR; // power on the PLL

**while**(!(SYSCTL\_PLLSTAT\_R & SYSCTL\_PLLSTAT\_LOCK)); // wait till PLL has locked

ADC0\_CC\_R = 0x01 ; wt++; // select PIOSC (internal RCOsc) as ADC analog clock

SYSCTL\_PLLFREQ0\_R &= ~SYSCTL\_PLLFREQ0\_PLLPWR; // power off the PLL (s. above)

// end of magic code ...

GPIO\_PORTM\_DEN\_R |= 0x02;

GPIO\_PORTD\_AHB\_DEN\_R = 0x01;

GPIO\_PORTM\_DIR\_R |= 0x01; // Richtungen der Ports

GPIO\_PORTD\_AHB\_AFSEL\_R |= (1 << 0);

GPIO\_PORTD\_AHB\_PCTL\_R = 0x03;

// Prepare Port Pin PE0 as AIN3

GPIO\_PORTE\_AHB\_AFSEL\_R |=0x01; // PE0 Alternative Pin Function enable

GPIO\_PORTE\_AHB\_AMSEL\_R |=0x01; // PE0 Analog Pin Function enable

GPIO\_PORTE\_AHB\_DEN\_R &=~0x01; // PE0 Digital Pin Function DISABLE

GPIO\_PORTE\_AHB\_DIR\_R &=~0x01; // Allow Input PE0

ADC0\_ACTSS\_R &= ~0x0F; // disable all 4 sequencers of ADC 0

ADC0\_SSMUX0\_R =0x00000003; // Sequencer 0 channel AIN3 only no mux

ADC0\_SSCTL0\_R |=0x00000002; // Set "END=0" sequence length 1 (one sample sequence)

ADC0\_CTL\_R =0x0; // V\_REF = Vdda 3.3V ... if Bit0 is clear

ADC0\_ACTSS\_R |= 0x01; // enable sequencer 0 of ADC 0

// configure Timer 0

SYSCTL\_RCGCTIMER\_R |= (1<<0); // timer 0

**while**(!(SYSCTL\_PRTIMER\_R & (1<<0))); // wait for timer 0 activation

TIMER0\_CTL\_R &= ~0x0001; // disable Timer 0

TIMER0\_CFG\_R = 0x04; // 2 x 16-bit mode

// compare mode, down, pwm: TAAMS=1, periodic: TAMR=0x2

TIMER0\_TAMR\_R = 0x000A;

TIMER0\_CTL\_R |= (1<<6); // TAPWML=1 (inverting)

TIMER0\_TAILR\_R = 16000-1; // f = 1 kHz

TIMER0\_TAMATCHR\_R = 4000-1; // MATCH = 0.5 ms

GPIO\_PORTD\_AHB\_DATA\_R |=(1<<1); // set PD(0) - Startsignal

//start on 2nd channel

TIMER0\_CTL\_R |= 0x0001; // enable Timer 0A

**while**(1)

{

ADC0\_PSSI\_R|=0x01; // Start ADC0

**while**(ADC0\_SSFSTAT0\_R & 0x000000100); // wait for FIFO (inverted) Flag "EMPTY = False"

ADCoutput = (**unsigned** **long**) ADC0\_SSFIFO0\_R; // Take avalue from FIFO output

**double** joyVolt = ADCoutput \* V\_COEFF + 0.5;

// Calculate Output in mV with respect to voltage divider 5:3 in the Lab

printf("0x%3x=%4d (dec) ==> %04d [mV] \n", ADCoutput, ADCoutput, (**int**) (joyVolt));

**int** joystickSwitch = (GPIO\_PORTM\_DATA\_R && 0x02); //switch value

printf("%i\n", TIMER0\_TAMATCHR\_R);

**double** speicher = TIMER0\_TAMATCHR\_R;

**if**(!joystickSwitch){

**if**((joyVolt > 3050)){

ADCoutput = (**unsigned** **long**) ADC0\_SSFIFO0\_R; // Take avalue from FIFO output

joyVolt = ADCoutput \* V\_COEFF + 0.5;

**while**(TIMER0\_TAMATCHR\_R < (14000-1)\*0.95){

TIMER0\_TAMATCHR\_R++;

}

}

**else** **if**((joyVolt < 50)){

ADCoutput = (**unsigned** **long**) ADC0\_SSFIFO0\_R; // Take avalue from FIFO output

joyVolt = ADCoutput \* V\_COEFF + 0.5;

**while**(TIMER0\_TAMATCHR\_R > (14000-1)\*0.05){

TIMER0\_TAMATCHR\_R--;

}

}

}

**else**{

speicher = TIMER0\_TAMATCHR\_R;

TIMER0\_TAMATCHR\_R = (16000-1) \* 0.78375;

busy (WAIT);

TIMER0\_TAMATCHR\_R = 0;

busy (WAIT);

TIMER0\_TAMATCHR\_R = speicher;

}

}

}

**void** busy (**unsigned** **long** zeit){

**unsigned** **long** i;

**for**(i=0;i<zeit;i++);

}

## Auswertung

Abbildung 6 zeigt ein PWM Signal mit 95% High-Pegel. Dies liegt auf dem Port M(0) wenn der Joystick nach rechts bewegt wird. Wird der Joystick jedoch nach links bewegt, liegt auf Port M(0) ein PWM-Signal mit 5% High-Pegel.

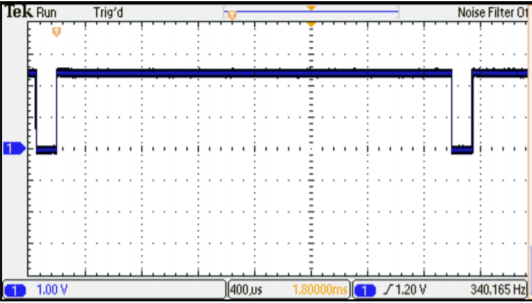


Abbildung 6: PWM-Signal mit 95% High-Pegel

Die Breite des PWM-Signals beeinflusst die Helligkeit der LED. Je höher die Breite wird, desto heller wird die LED.

# Fazit

Das Treppenverfahren ist eine simple Methode, um analoge Werte in digitale umzuwandeln. Dennoch ist dieses Verfahren ungenauer als andere Messgeräte. Der interne A/D Umwandler des Mikrokontroller ist zwar etwas komplexer zu programmieren, dennoch ist das eine effizientere Lösung als einen externen Umwandler zu benutzen.