Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Department Informations- und Elektrotechnik
Labor für Digitale Informationstechnik
Praktikum Mikroprozessortechnik

Aufgabe **1** *Exercise*	**Digitalvoltmeter** ***Digital Voltmeter***	
Semester / Gruppe *Semester / Group* 4/1/6	Abgabedatum 11.12.2019	Protokollführer *Chairperson* Rami Chaari
Versuchstag *Day of Exercise* 4.12.2019		Versuchsteilnehmer *Participants* Felix Rehaag
Professor *Professor* Prof. Dr. Paweł Buczek		

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

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.

Digital input	Analog output
0000 0000	0V
0000 0001	0V + 1 U_{LSB}
0000 0010	0V + 2 U_{LSB}
...	...
1111 1110	5V - 2 U_{LSB}
1111 1111	5V - 1 U_{LSB}

Voltage step: U_{LSB} = 5V/256 = 19.53125mV

Analog output: U_{out} = (Digital input)·U_{LSB}

Tabelle 1: Übersetzungsverhalten des D/A-Umsetzers

Aufbau

Abbildungen 1 und 2 zeigen die Schaltung.

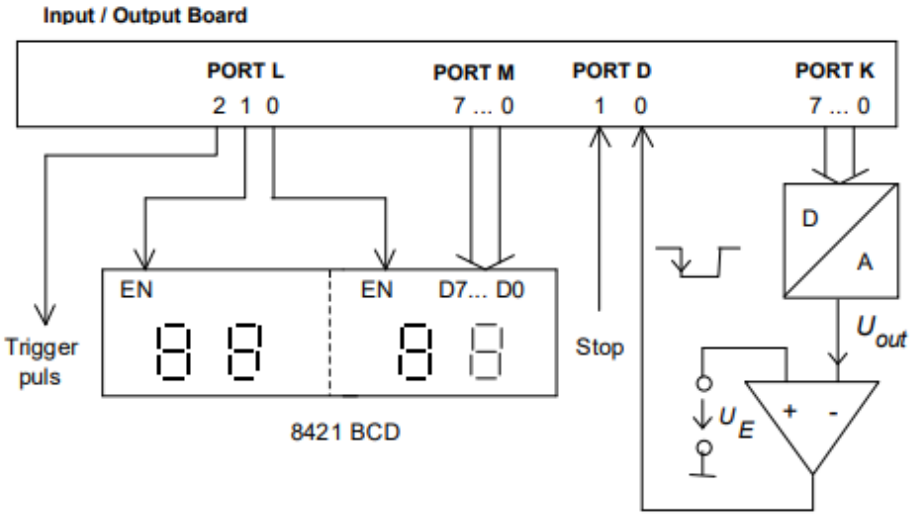


Abbildung 1: Schaltung der Aufgabe 1

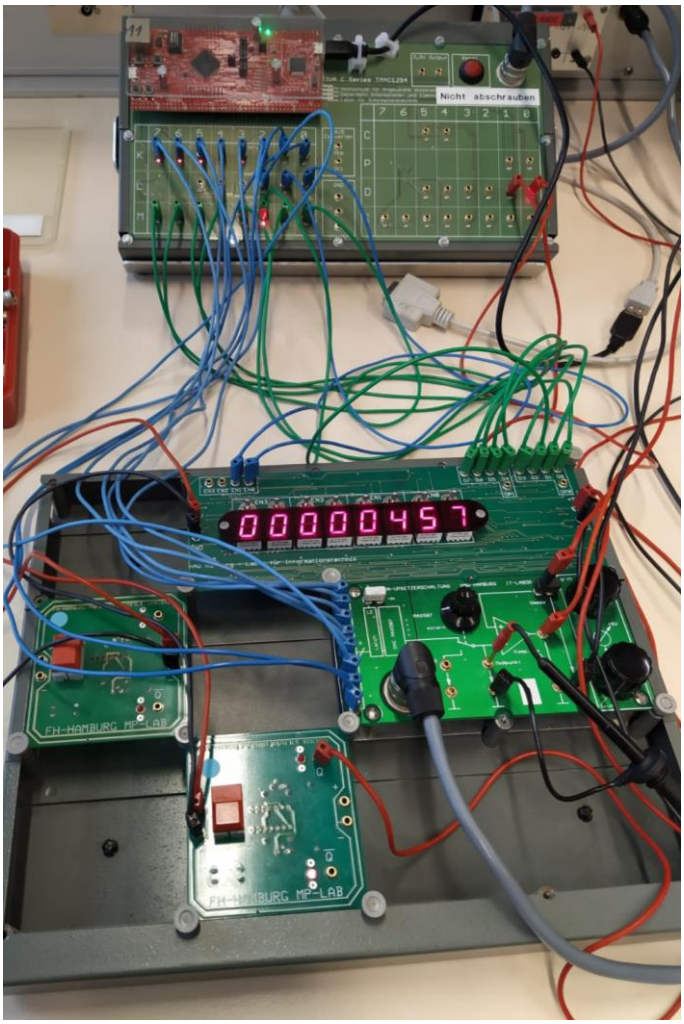
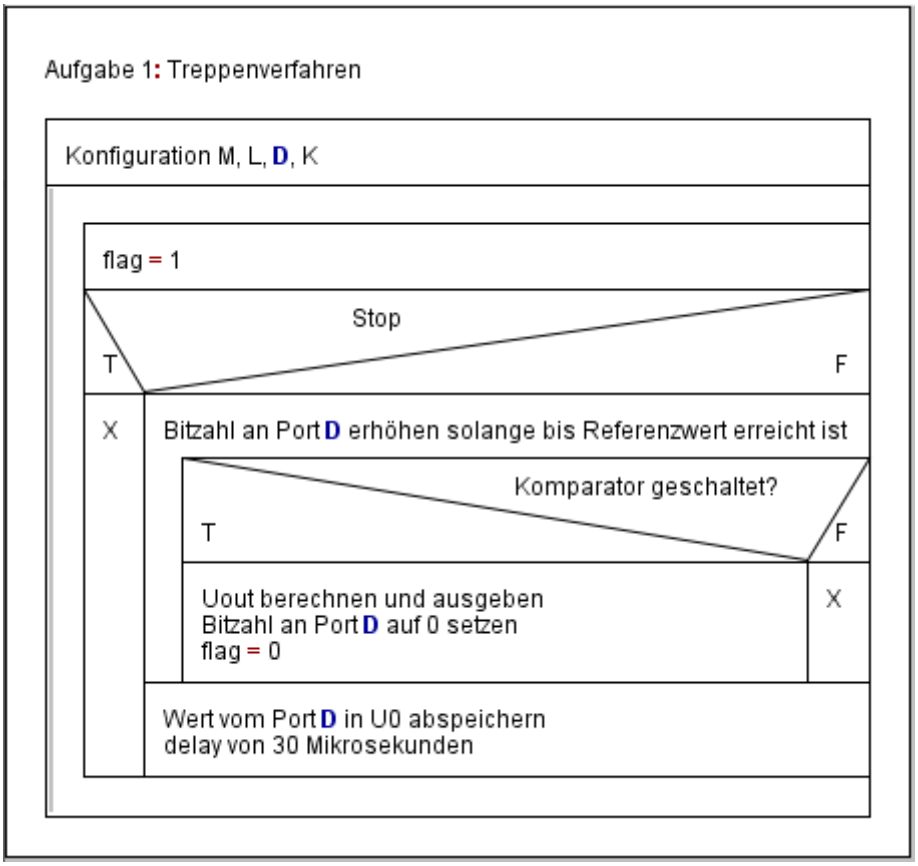


Abbildung 2: Aufbau der Schaltung im Labor

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.

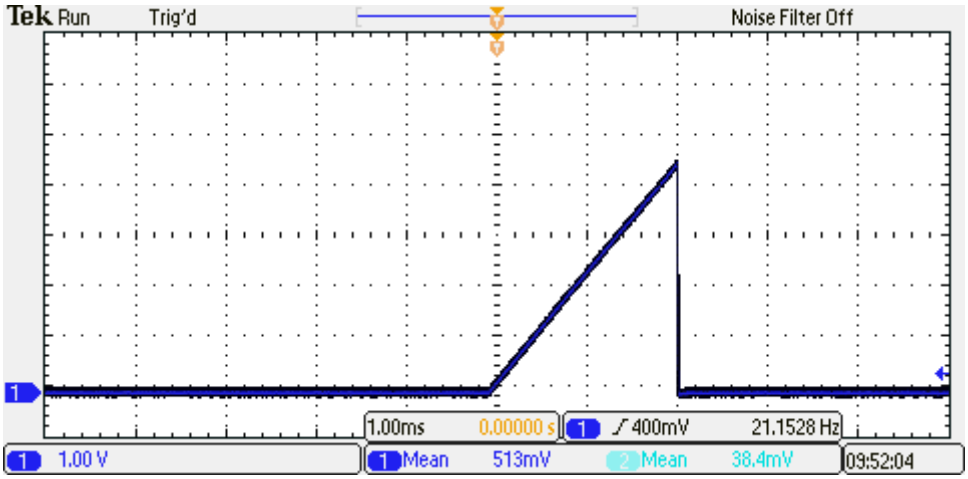


Abbildung 3: Messverlauf der Spannung am Port K

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.

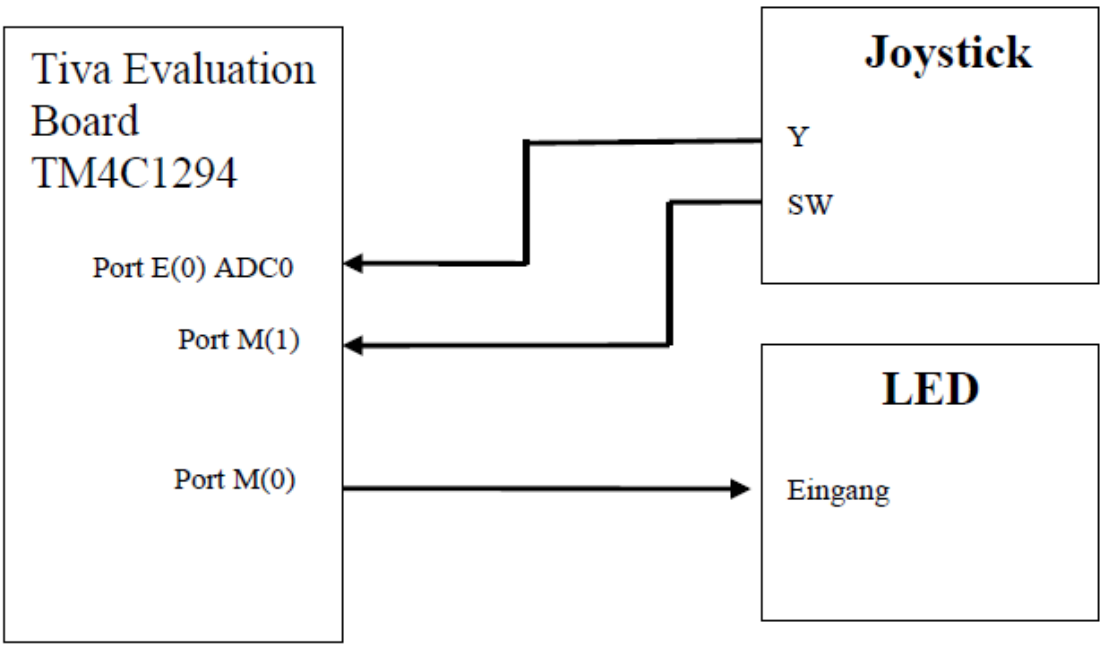


Abbildung 4: Aufbau der Schaltung für Aufgabe 3

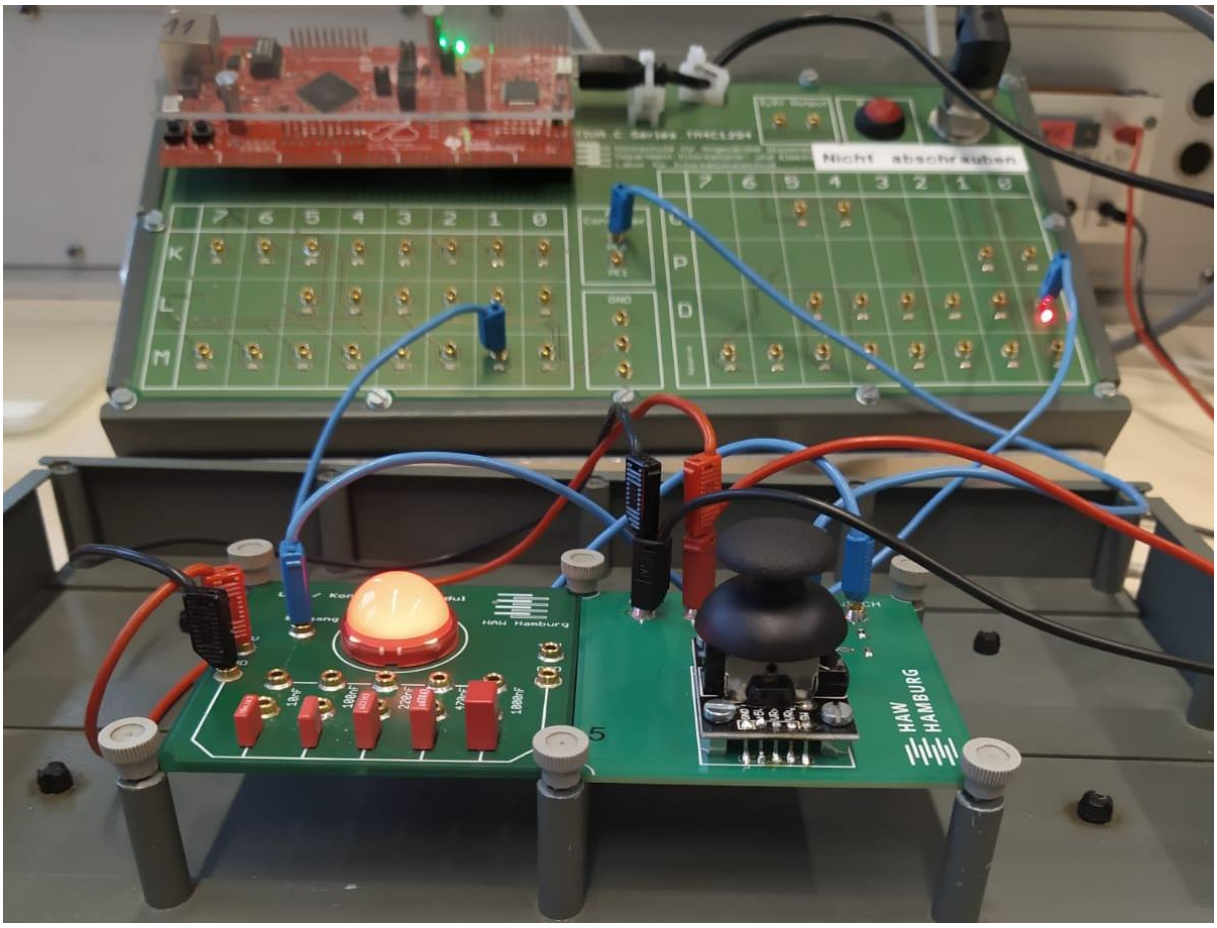
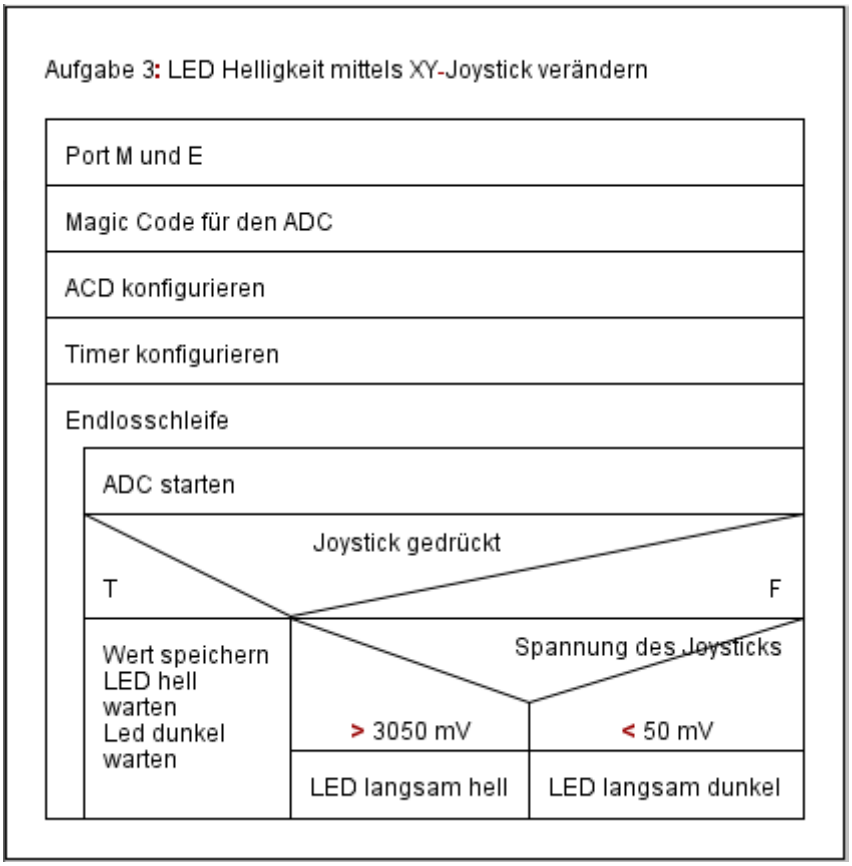


Abbildung 5: Aufbau der Schaltung im Labor für Aufgabe 3

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.

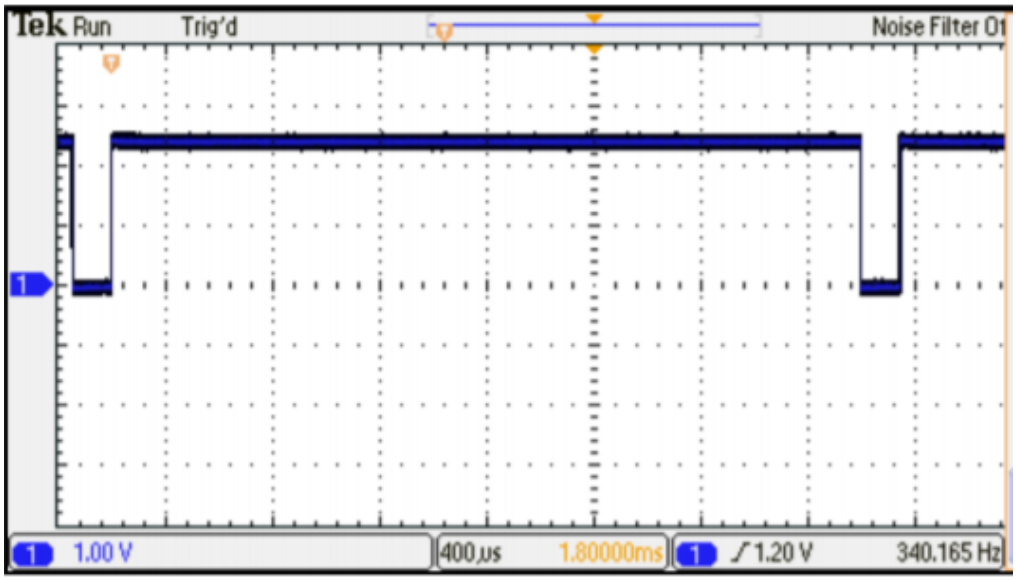


Abbildung 6: PWM-Signal mit 95% High-Pegel

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.