Fachhochschule Münster University of Applied Sciences

Labor für Halbleiterbaulelemente und Bussysteme

Übung Bussysteme SS2015
Tutor: Diplom Informatiker Gregor Rebel

Praktikum 1
LED-Pulsweitenmodulation

Die meisten Mikrocontroller können die Spannung an ihren Ausgangpins nur zwischen 0V und ihrer Betriebsspannung (z.B. 3,3V) umschalten (Abb. 1). Für viele Steuerungsaufgaben ist jedoch ein analoges Ausgangssignal gefordert. Anstatt einer analogen Spannnung kann der Mikrocontroller jedoch das Verhältnis zwischen Einschalt- und Ausschaltzeit in mehreren Stufen variieren. Dieses gestufte Tastverhältnis läßt sich einfach in eine entsprechende analoge, elektrische Leistung umwandeln. Abb. 2 zeigt eine digitale Spannung mit zunehmendem Tastverhältnis sowie die mittlere Leistung über jeweils eine Periode über einem ohmschen Widerstand.

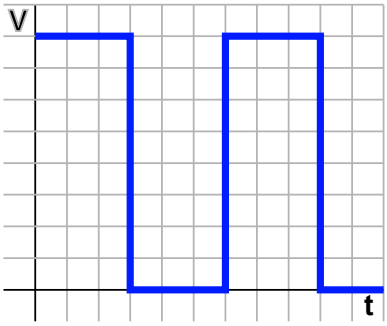


Abb. 1: Digitale Ausgabe

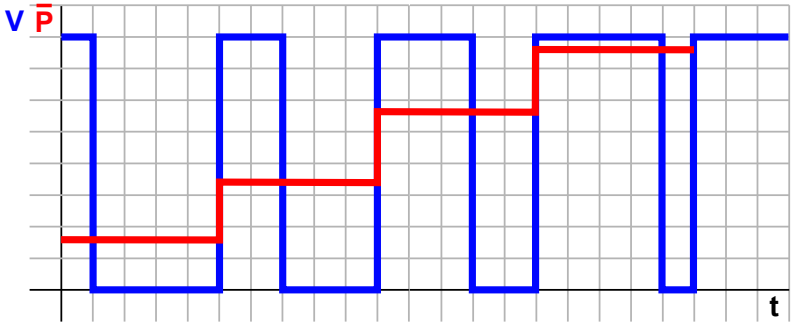


Abb. 2: Pulsweitenmodulation und gemittelte Leistung

Ein einfaches Beispiel ist die Steuerung der Helligkeit einer Leuchtdiode (LED). Hierbei wird die Ausgangsspannung über einen Widerstand und eine LED geleitet. Der Widerstand wandelt die Spannung in einen Strom durch die Leuchtdiode. Die Leuchtdiode gibt die Spannungsänderung je nach ihrer Bauform mehr oder weniger schnell als Helligkeitsänderung weiter. Die eigentliche Glättung zu einem Analogsignal geschieht im deutlich langsamer aufnehmendem menschlichen Auge. Dieses kann Helligkeitsänderungen die schneller als 20/s sind nicht erkennen. Stattdessen sieht das Auge einen gemittelten Helligkeitswert. Abb. 3 zeigt die beiden wesentlichen Möglichkeiten LEDs an einen Mikrocontroller anzuschließen. Die obere LED ist sogenannt **high-aktiv** angeschlossen. Ein High-Pegel an Pin 1 legt eine positive Spannung (hier 3,3V) gegen Masse (0V) an den linken Anschluss der LED D1 an. Der einsetzende Stromfluss durch die D1 wird durch den Widerstand R1 begrenzt um die LED zu schützen. Die LED leuchtet also wenn Pin 1 high ist. Die LED D2 dagegen ist umgekehrt gepolt und mit ihrer Anode über den Widerstand R2 an +3,3V angeschlossen. D2 leuchtet auf, wenn der Mikrocontroller Pin 3 aktiv auf Low (0V) zieht. Daher ist D2 **low-aktiv** angeschlossen. Wie eine LED angeschlossen ist, kann dem Schaltplan des verwendeten Mikrocontrollerboards entnommen werden. Dort ist auch aufgezeichnet, welche Pins des Mikrocontrollers mit welchen Geräten (LED, Taster, Schnittstellen, usw.) verbunden sind.

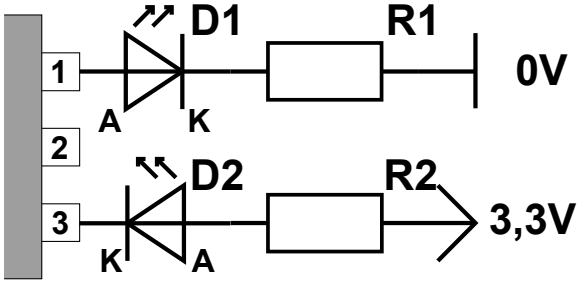


Abb. 3: Schaltplan - LEDs am Mikrocontroller

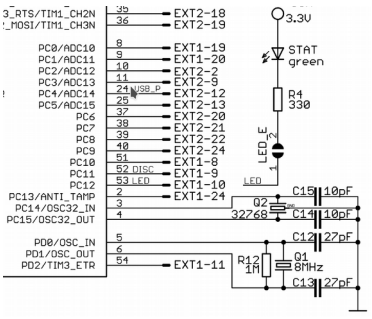


Abb. 4: Schaltplanausschnit Prototypboard H107

Diese Übung dient dazu zunächst den Umgang mit der Entwicklungsumgebung The ToolChain an einer einfachen Aufgabe zu üben. Die in der C-Einführung erarbeiteten Grundlagen sollen hier zum Einsatz kommen.
Loggen Sie sich dazu auf einem der Arbeitsplatzrechner ein und starten sie das Programm konsole (rechter Mausklick → Befehl ausführen → eintippen: „konsole“) oder yakuake (F12 drücken) falls installiert.

Die Konsole stellt eine grafische Textanzeige bereit in der eine sogenannte Shell läuft. Die Shell ist eine kommandozeilenorientierte Schnittstelle zum Betriebsystem. In der Shell können die Inhalte der Festplatte angezeigt und modifiziert werden. Die Shell erlaubt jedes auf der Festplatte installierte Programm egal ob grafisch oder textorientiert zu starten. Viele Programme können nur textuell bedient werden. Textuelle Programme haben den großen Vorteil, das ihr Ablauf einfach automatisiert werden kann. Auch kann ein geübter Anwender mit dem Textinterface deutlich schneller sein als mit einer grafischen Oberfläche. Ein weiterer Vorteil ist, dass textuelle Ausgaben wenig Bandbreite benötigen und sich daher gut von einem entfernten Rechner über langsame Internetverbindungen benutzt werden können.

Die wichtigsten Befehle zur Shell Nutzung sind hier aufgelistet:
http://thetoolchain.com/documentation/Linux_Shell_Essentials.pdf

Die wesentlichen Punkte in Kürze:

cd VERZEICHNIS	in anderes Verzeichnis wechseln
ll	Inhalt des aktuellen Verzeichnisses auflisten (Alias für ls -l)
./PROGRAMM	Programm im aktuellen Verzeichnis starten
pwd	Aktuelles Verzeichnis ausgeben
~	Das Heimatverzeichnis des aktuellen Benutzers (z.B. ~/Source/)

Wichtig: Namen von Verzeichnissen und Dateien in Unixdateisystemen beachten immer **GROSS-kleinschreibung**!
Hilfe: Drücken der Tabulatortaste →| wirft einen Blick ins Dateisystem um Dateipfade zu vervollständigen.

Wechseln Sie nun mit Hilfe der Shell in folgenden Ordner: ~/Source/TheToolChain

Erstellen Sie ein neues ToolChain Projekt durch Ausführen des Skriptes createNewProject.pl.
Gruppe1:

./createNewProject.pl Test_BS_G1_E1 G

Gruppe2:

./createNewProject.pl Test_BS_G2_E1 G

Dadurch wird ein neues Projekt im Ordner ~/Source/ Test_BS_G1_E1/ bzw. ~/Source/ Test_BS_G2_E1/ erstellt.
Wechseln Sie nun in diesen Projektordner.

Zum Editieren von Quelltexten verwendet The ToolChain von Haus aus das Programm QtCreator. Dieses ist eigentlich zum Erstellen von grafischen Anwendungen in C++ geschrieben worden. Es erlaubt jedoch auch die Bearbeitung beliebiger C-Projekte. In jedem TTC-Projekt liegt ein Shellskript um den QtCreator mit dem aktuellen Projekt zu starten. Starten Sie nun dieses Skript:
./qtcreator.sh

Damit wird die grafische Anwendung QtCreator gestartet und das aktuelle Projekt automatisch geladen.

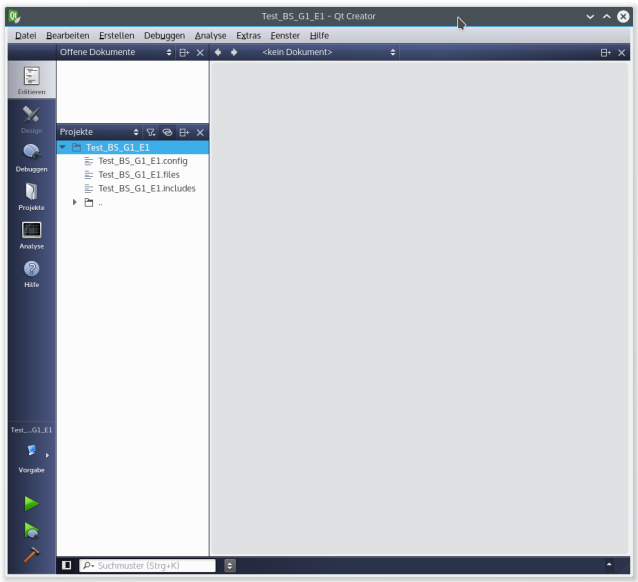


Abb. 5: QtCreator Projekt geladen

Die main.c Datei mit der Funktion main() ist in der Projektübersicht im Ordner .. zu finden. In main() beginnt der Programmfluss jedes C-Programmes.
Sie können den Programmfluss einfach nachverfolgen indem Sie die Maus über eine Funktion bewegen und die **Taste F2** drücken. Der Editor springt dann an die Stelle wo die Variable/ Funktion/ der Datentyp definiert wurde.

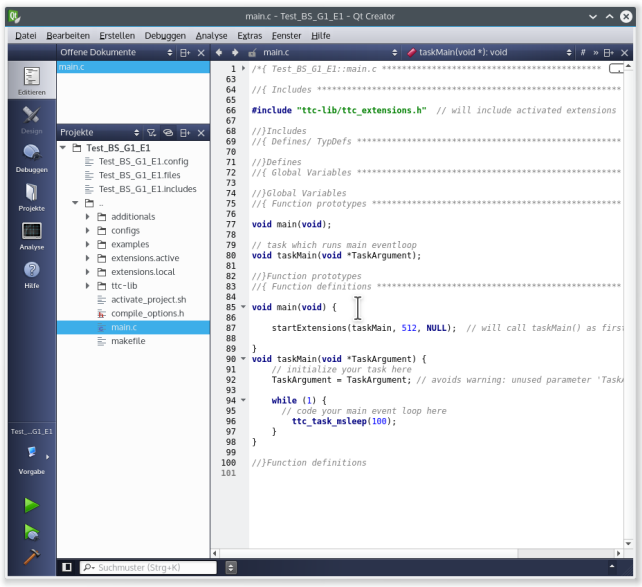


Abb. 6: Die main-Funktion

Aufgabe 1

Verfolgen Sie den Programmfluss bis zur Funktion example_leds_start().

Welche Funktionen wurden auf diesem Weg durchlaufen?

_____
_____
_____

example_leds_start()

Das Beispiel example_leds ist in jedem ToolChain Projekt zu Beginn aktiviert. Es bietet einen einfachen Test und sollte auf allen unterstützten Prototypboards laufen.

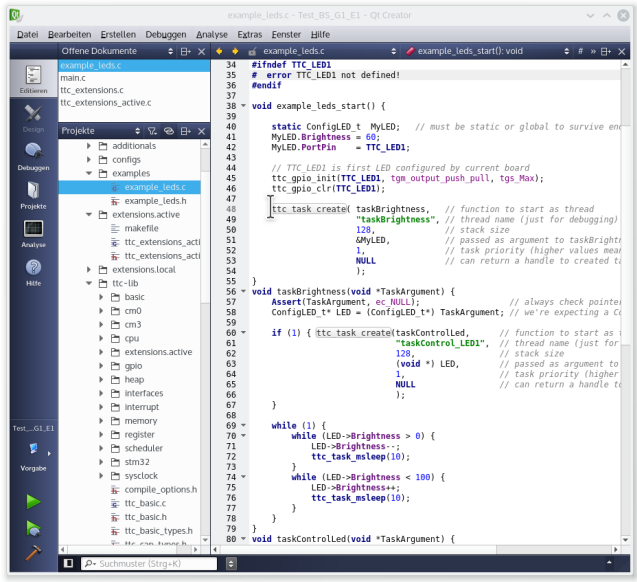


Abb. 7: example_leds_start()

Funktionen mit der Endung _start() dienen in The ToolChain der Initialisierung beim Systemstart. Diese Funktionen werden noch ohne Multitasking gestartet. Hier können Speicherbereiche belegt, Schnittstellen initialisiert und Tasks gestartet werden. Ein Task wird durch die Funktion ttc_task_create() gestartet. Die Funktion erhält als Argument u.a. den Zeiger auf die Taskfunktion. Die Taskfunktion ist eine gewöhnliche C-Funktion mit einem einzelnen void* Argument und einer Endlosschleife. Task-Funktionen dürfen niemals enden!
Auch kann ein einzelnes Argument übergeben werden das an die Taskfunktion weitergereicht wird. Dies kann z.B. ein Zeiger auf Daten sein die vom Task benötigt werden.
In diesem Beispiel wird zunächst die Funktion taskBrightness() als Task gestartet. Das dritte Argument von ttc_task_create() ist ein Zeiger auf die statische Variable LED. Dieses Argument wird direkt an die Taskfunktion taskBrightness() weitergereicht.
Statische Variablen behalten ihren Speicherplatz auch nach Ende einer Funktion. Das ist in diesem Fall wichtig, da der Task taskBrightness() erst gestartet wird nachdem die Funktion example_leds_start() bereits verlassen wurde. Wäre die Variable nicht static oder global deklariert, so würde ich Speicherplatz am Funktionsende freigegeben. Spätere Zugriffe auf ihre Adresse würden dann andere Daten überschreiben.
Wie im Programmfluss vorher zu sehen, wird in ttc_extensions.c:startExtensions() nach ttc_extensions_init() die Funktion ttc_task_start_scheduler() aufgerufen. Diese startet dann alle zuvor erstellten Tasks.
Die Tasks werden dann vom Multitasking Scheduler immer stückweise abgearbeitet. Nach ein paar Mikrosekunden in dem einen Task, schaltet der Scheduler auf einen anderen Task um und arbeitet diesen etwas ab. Durch die schnelle Umschaltung zwischen allen Tasks entsteht der Eindruck, dass alle Tasks parallel arbeiten.

Aufgabe 2

Ergänzen Sie die Zeichnung in um Pfeile, welche die Datenzugriffe auf die Variable MyLED.Brightness von den verschiedenen Tasks aus zeigen.
Verwenden Sie folgende Pfeile:
Schreibzugriff: Variable ← Task
Lesezugriff: Variable → Task

Nummerieren Sie Schreibzugriffe nach ihrer Reihenfolge. Warum kann es zu keiner Kollision zwischen taskBrightness() und taskControlLed() kommen?

_____

_____

_____

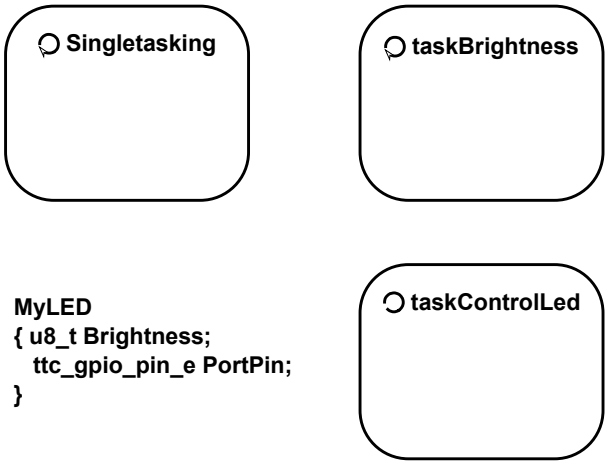


Abb. 8: Datenzugriffe aus verschiedenen Tasks

GPIO-Pins sind das Schweizer Messer in der Welt der Mikrocontroller. Ein General Purpose Input Output Pin kann per Software als digitaler Eingang oder Ausgang konfiguriert werden. Eingänge fragen z.B. Taster ab. Ausgänge schalten LEDs ein, fahren Motoren, erzeugen Töne und vieles andere. Die Zuordnung einzelner Beinchen am Mikrocontrollergehäuse zu logischen GPIO-Pins ist im Datenblatt des Mikrocontrollers und häufig auch im Schaltplan eines gut dokumentierten Prototypboards beschrieben. Die Schaltpläne zu den unterstützten Prototypboards finden Sie im Ordner **~/Source/TheToolChain/Documentation/Boards/**.

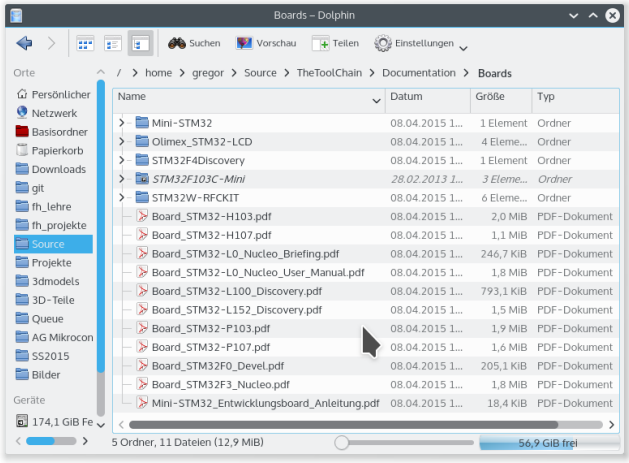


Abb. 9: Dokumentationen der Prototypboards im Dolphin

Aufgabe 3

Öffnen Sie den Schaltplan Ihres aktuellen Prototypboards. Nutzen Sie dazu den Dateimanager dolphin, den Sie direkt aus der Shell starten können:
dolphin &

Finden Sie anhand des Schaltplans heraus an welchen GPIO-Pins und welchen Beinchen die verschiedenen LEDs und Taster angeschlossen und ob diese high- oder low-aktiv verschaltet sind.

	GPIO-Pin	Beinchen #	High/Low Aktiv
LED1			
LED2			
SWITCH1			
SWITCH2			

Für GPIO-Pins is ttc_gpio zuständig. Wie bei allen ToolChain Modulen beginnen alle darin bereitgestellten Funktion mit dem Prefix ttc_gpio_. Die wesentlichen Funktionen sind die folgenden:

- **ttc_gpio_init()**
  Konfiguriert einen GPIO-Pin als Eingang, Ausgang oder für eine spezielle Verwendung.
- **ttc_gpio_set()**
  Setzt einen als Ausgang konfigurierten GPIO-Pin auf logisch high (z.B. 3,3V)
- **ttc_gpio_clr()**
  Setzt einen als Ausgang konfigurierten GPIO-Pin auf logisch low (0V). Wurde der Pin als Push-Pull Ausgang konfiguriert, so wird der Ausgang aktiv gegen 0V (GND) gezogen.
- **ttc_gpio_get()**
  Liest den aktuellen logischen Zustand eines als Eingang konfigurierten Pins. Das Ergebnis ist 0, wenn die Spannung am Pin < 0,5 * Vcc (z.B. < 0,5 * 3,3V) ist. Ansonsten ist das Ergebnis 1.

Die genauen elektrischen Eigenschaften können den Datenblatt des eingesetzten Mikrocontrollers entnommen werden (z.B. ~/Source/TheToolChain/Documentation/uC/STM32F1xx/STM32_RM0008_....pdf)

Aufgabe 4

Springen Sie im QtCreator zur Deklaration von ttc_gpio_init() und lesen Sie die Dokumentation dieser Funktion.
Springen Sie auch zu den Definitionen der für die Argumente verwendeten Datentypen.
Tragen Sie unten die Namen der Argumente sowie gültige Werte für folgende Konfigurationen ein:

Pin C1, Push-Pull Ausgabe, Maximale Geschwindigkeit

Argument	Wert

Pin A2, Floatende Eingabe, Minimale Geschwindigkeit

The ToolChain bietet universelle Namen für Taster und LEDs an. Jede Boarderweiterung konfiguriert diese automatisch. Die LEDs heissen TTC_LED1, TTC_LED2, … während die Taster TTC_SWITCH1, TTC_SWITCH2, … heissen.
Nutzen Sie die Möglichkeiten des QtCreator um die aktuellen Werte herauszufinden:
TTC_LED1 _____
TTC_LED2 _____
TTC_SWITCH1 _____
TTC_SWITCH2 _____

Aufgabe 5

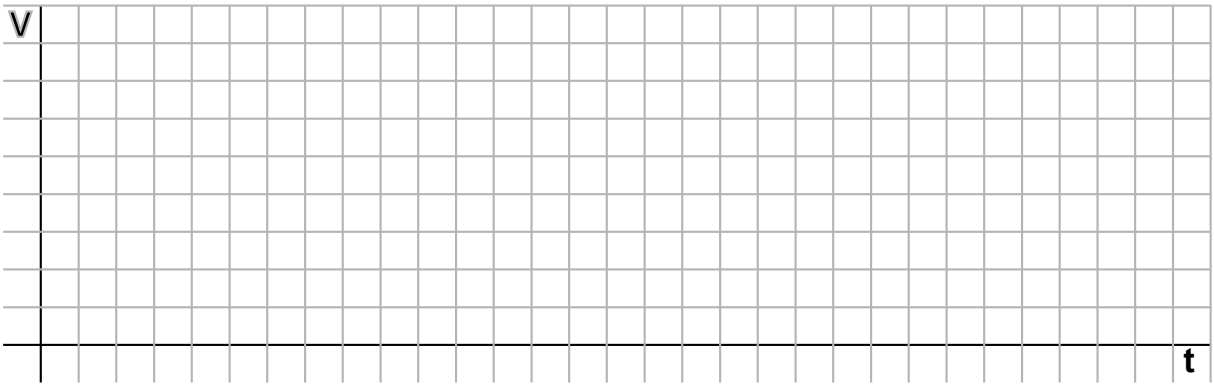
Ergänzen Sie das Beispiel in example_leds.c so, daß die LED nur noch dann eingeschaltet wird, wenn der Taster TTC_SWITCH1 gedrückt wird. Tragen Sie dazu eine weitere Variable Enable in die Struktur Config_LED ein. Den Taster fragen Sie regelmäßig in taskBrightness() ab. Ist der Taster gedrückt, so setzen Sie Enable auf 1, sonst auf 0. In taskControlLed() fragen Sie die Variable Enable ab. Ist Enable == 0, so wird ttc_gpio_set() nicht ausgeführt.

Die ausgegebene Pulsweitenmodulation ist zu schnell um vom menschlichen Auge als Flackern wahrgenommen zu werden. Die PWM kann jedoch mit einem digitalen Speicheroszilloskop dargestellt werden. Ein Oszilloskop schreibt fortlaufend den an einem elektrischen Eingang gemessenen Spannungsverlauf von links nach rechts auf eine Anzeige. Um die Anzeige mit einem sich wiederholenden (periodischen) Signal zu synchronisieren wird ein sogenannter Triggerevent festgelegt. Das kann z.B. eine positive Signalflanke (der Durchlauf des Spannungspegels von unten durch einen festgelegten Spannungswert) sein. Dazu bietet jedes Oszilloskop die Einstellung eines Triggerlevels. Der Triggerlevel ist der Spannungswert der über- (positive Flanke) oder unterschritten (negative Flanke) werden muss um die Darstellung auszulösen. Ist der Triggerlevel falsch eingestellt, wird an der falschen Stelle oder gar nicht angezeigt.

Schließen Sie ein Oszilloskop an Ihr Protoboard an. Verbinden Sie zunächst die Krokodilklemme des Tastkopfes mit einem gekennzeichneten GND Pin des Protoboards. Dann führen Sie die Spitze des Tastkopfes an einen mit 3,3V gekennzeichneten Pin. Alternativ können Sie auch einen Pin an einem Bauteil benutzen, der im Schaltplan des Protoboards an das 3,3V Netz angeschlossen ist.

Finden Sie nun am Oszilloskop die Einstellungen für den Triggerlevel. Drehen Sie den Levelregler so weit, dass die angezeigte Linie mittig zwischen der aktuell gemessenen 3,3V Linie und der 0V Markierung (am linken Rand) liegt.

Nach dem Einstellen des Triggerlevels auf die zu erwartende Spannung, führen Sie den Tastkopf nun zu beiden Enden der auf und abschwellenden LED. Drücken Sie am Oszilloskop die Stop-Taste um die Anzeige einzufrieren. Zeichnen Sie den angezeigten Signalverlauf in folgendes Diagramm:



Name: _____ Name: _____

Unterschift: _____ Unterschrift: _____