Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Laborbericht

Labor 1: I/O Ports

Laborbericht eingereicht von
Lamowski, Michelle
Matrikelnummer 2577377

im Rahmen der Vorlesung *Informatik & Elektronik*
im Studiengang *Media Systems*
am Department Medientechnik der Fakultät DMI
an der Hochschule für Angewandte Wissenschaften Hamburg

Lehrende: Prof. Dr. Tessa Taefi

Eingereicht am: 19.04.2022

Inhalt

1 Zusammenfassung

In dem bereits durchgeführten Labor am 12.04.2022 wurden die Inhalte der Vorlesung Informatik & Elektronik vertieft und Erfahrungen im Umgang mit der Programmiersprache Embedded C, Microchip Studio und SimulIDE gesammelt.

2 Einleitung

Die drei Teilaufgaben des Labors bestehen aus mikrocontrollerbasierten Problemstellungen zum Thema I/O Ports. Diese gilt es mit Hilfe einer strukturierten Anforderungsanalyse zu lösen, um die Methoden Analyse, Design und Implementierung zu beherrschen.

Die Vorbereitung wird im Home-Office in Zweiergruppen durchgeführt. Die Abnahme des Labors geschieht in Präsenz. Das Ziel ist es, jede Teilaufgabe lauffähig vorzuzeigen und die Vorgehensweise erklären zu können. Die Ergebnisse werden fachgerecht dokumentiert.

Die **erste Aufgabe** beschäftigt sich mit dem Anwenden des Programms *LEDOnButtonPress* aus der Vorlesung auf die Hardware der Laborumgebung, sodass ein tieferes Verständnis für die Umgebung Microchip Studio und den verwendeten Microcontroller erlangt wird. Außerdem soll ein neues Programm namens *CountButtonAndBlink* geschrieben werden, welches zählt, wie oft der erste Button gedrückt wurde und die LED entsprechend oft blinken lässt. Das Programm soll mithilfe der Anforderungsanalyse geschrieben werden und in der Laborumgebung getestet werden.

In der **zweiten Aufgabe** wird das selbstentwickelte Shield verwendet. Die Voraussetzung dafür ist es, den beigefügten Schaltplan zu verstehen, sodass der benötigte Strom für eine LED, sowie für die gesamte LED-Bank berechnet werden kann. Davon ausgehend sollte nun geprüft werden, ob der Strom, den die Ports des Microcontrollers zur Verfügung stellen, ausreichend ist. Im weiteren Verlauf der Aufgabe soll das Programm *CountButtonAndBlink* auf die neue Situation mit dem Shield portiert werden. Für ein tieferes Verständnis der Elektronik wird mit dem Pull-up Widerstand des Push-Buttons experimentiert und die Beobachtungen notiert. Des Weiteren wird das Programm durch neue Funktionen erweitert und verändert. Erst soll durch das Drücken des zweiten Buttons ein Zurücksetzen des Blinkens möglich sein. Danach sollen die gezählten Blinks als vorzeichenlose Dualzahl auf den ersten vier LEDs angezeigt werden.

In der **dritten Aufgabe** werden Interrupts thematisiert. Dazu soll ein neues Programm geschrieben werden, welches beim Drücken auf den ersten Button alle grünen LEDs anschaltet, beim Drücken auf den zweiten Button alle roten. Das Programm wird auf zwei Arten geschrieben werden - mit der Verwendung von Polling und mit einem Pin Change Interrupt. Außerdem sollen Macros eingesetzt werden, um die Programmierarbeit zu erleichtern.

3 Materialien und Methoden

Für die Vorbereitung des Labors war es nötig, an der Vorlesung Informatik & Elektronik teilzunehmen, sowie eine ausgiebige Vor- und Nachbereitung der Lehrinhalte. Die Grundlagen der Programmiersprache Embedded C müssen beherrscht werden und die Entwicklungsumgebung Microchip Studio bekannt sein.

Die Aufgaben des Labors wurden bereits vor dem Tag der Abnahme vorbereitet, um verbleibende Fragen rechtzeitig stellen zu können. Diese wurden mithilfe der Anforderungsanalyse strukturiert gelöst. Die Schritte werde ich im folgenden Erläutern.

Abbildung 1: Anforderungsanalyse
Quelle: Eigene Darstellung

Zuerst wird eine **Analyse** der Problemstellung durchgeführt. Dafür wird ein Blockschaltbild erstellt, um einen Systemüberblick zu gewinnen. Anschließend wird ein detaillierter **Schaltplan** skizziert, um sich mit den Bauteilen, den Controllern und den Ein- und Ausgängen vertraut zu machen. Als Nächstes wird ein **UML Aktivitätsdiagramm** erstellt. Dies hilft besonders, um Abläufe übersichtlich darzustellen und ist eine hilfreiche Vorbereitung für die folgende **Implementierung in Microchip Studio**. Im nächsten Schritt kann das C-Programm in **SimulIDE** getestet und simuliert werden.

Am Tag der Abnahme wurden dann die geschriebenen Programme in der Hardware der Laborumgebung implementiert und getestet. Es wurde ein Atmega 328P Xplained Mini Board verwendet. Außerdem wurde für die weiterführenden Aufgaben ein selbstentwickeltes Shield angeschlossen.

Aus Zeitgründen haben wir die letzte Teilaufgabe nicht vorbereitet, sondern diese direkt am Tag der Abgabe versucht zu lösen.

4 Ergebnisse

4.1 Aufgabe 1: Xplained Mini Board

4.1.1 LEDOnButtonPress

Wir haben das Programm *LEDOnButtonPress* problemlos auf dem Xplained Mini Board zum Laufen gebracht.

4.1.2 CountButtonAndBlink

Wir haben das Programm *CountButtonAndBlink* mithilfe der Anforderungsanalyse vorbereitet. Wir initialisieren eine Integer Variable blinkCount. Diese Variable wird erhöht, wenn der erste Button gedrückt wird. Im Anschluss wird die Funktion blink() aufgerufen, welche eine lokale Integer Variable i auf den Startwert 0 setzt. Die Variable i durchläuft eine for-Schleife und wird bei jedem Durchlauf inkrementiert. Die Schleife läuft so lange durch, bis die Variable nicht mehr kleiner als blinkCount ist. Bei jedem Schleifendurchlauf wird außerdem die LED ein- und wieder ausgeschaltet.

1. Analyse

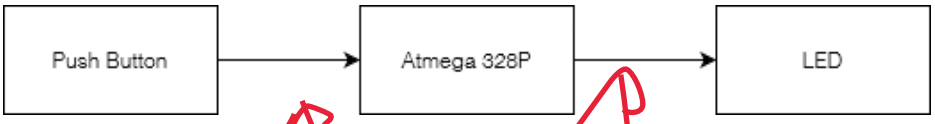

Abbildung 2: Blockschaltbild
Quelle: Eigene Darstellung

2. HW Design mit Schaltplan

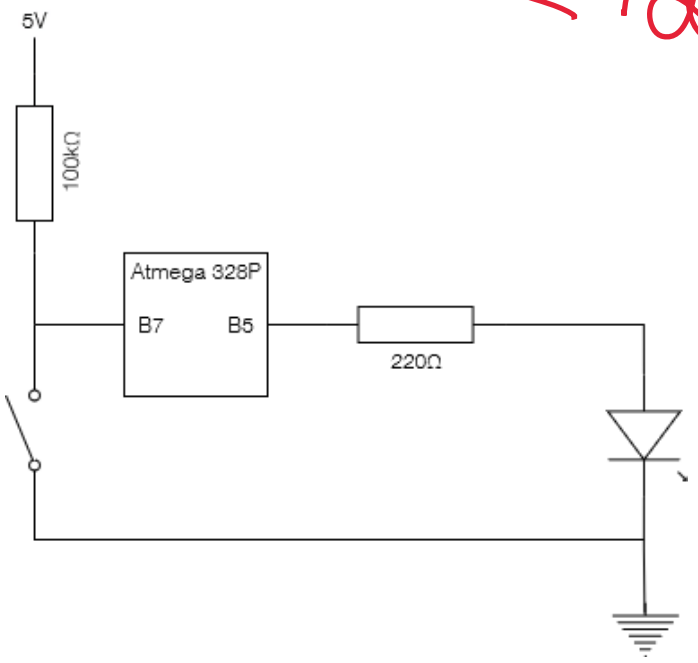

Abbildung 3: Schaltplan
Quelle: Eigene Darstellung

3. SW Design mit UML Aktivitätsdiagramm

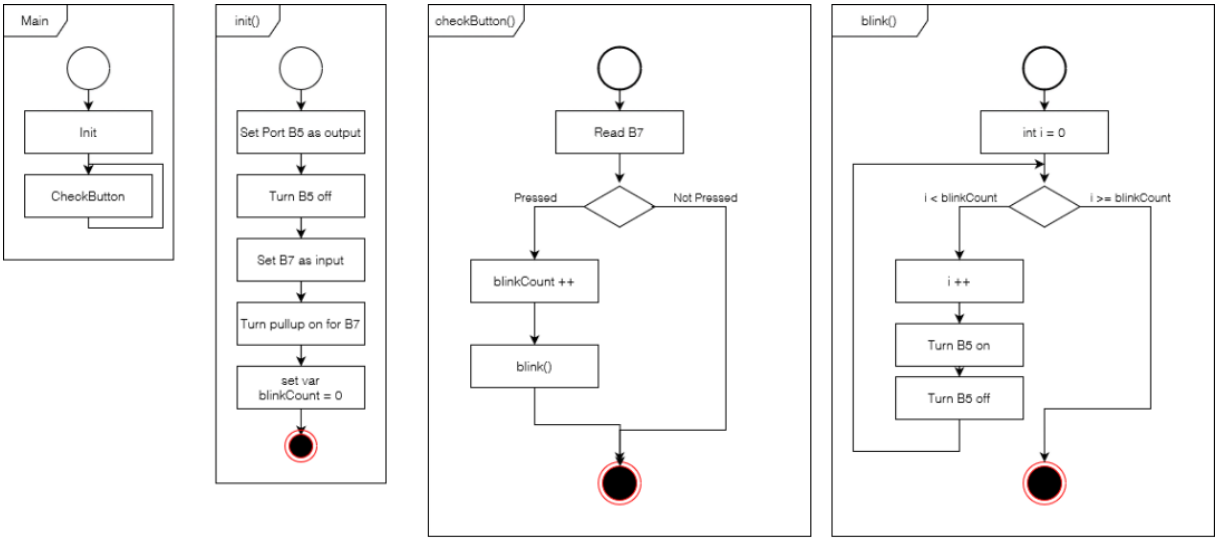

Abbildung 4: Aktivitätsdiagramm
Quelle: Eigene Darstellung

4. Implementieren in Microchip Studio

Mithilfe des Aktivitätsdiagramms haben wir den Code in Microchip Studio implementiert. Der Quellcode ist im Anhang zu finden.

5. Simulation in SimulIDE

Der Schaltplan wurde in SimulIDE nachgebaut und das Programm getestet. Das Programm hat funktioniert.

6. Anwendung in der Laborumgebung

Das Programm hat ebenfalls auf dem Xplained Mini Board funktioniert.

4.2 Aufgabe 2: Shield in Betrieb nehmen

Für die zweite Laboraufgabe haben wir die USB-Verbindung entfernt, sodass das Xplained Board stromlos ist. Im nächsten Schritt haben wir das selbstentwickelte Shield verbunden.

4.2.1 Berechnung des benötigten Stroms

Für die Berechnung des benötigten Stroms einer LED haben wir die Formel

$$I = \frac{U}{R}$$

Formel 1: Berechnung des benötigten Stroms
Quelle: Vorlesung Informatik & Elektronik

verwendet, wobei I die Stromstärke in Ampere ist, U die elektrische Spannung in Volt und R der elektrische Widerstand in Ohm.

1. Benötigter Strom für eine LED berechnen:

$$I = \frac{(5\text{V} - 2{,}3\text{V})}{220\ \Omega} = 12{,}27\ ma$$

Formel 2: Berechnung des benötigten Stroms für eine LED
Quelle: Datenblatt des Shields

2. Benötigter Strom für die LED-Bank berechnen:

$$10 * 12{,}27\ ma = 123\ ma$$

Formel 3: Berechnung des benötigten Stroms der LED-Bank
Quelle: Datenblatt des Shields

Abschließend ist zu prüfen, ob der Strom, den die Ports zur Verfügung stellen, ausreichend ist.

Dafür muss gewährleistet sein, dass die Ports mehr als 123 ma zur Verfügung stellen können. In der Dokumentation des Shields ist beschrieben, dass diese sogar bis zu 150 ma zur Verfügung stellen können. Der Strom, den die Ports zur Verfügung stellen, ist also ausreichend.

4.2.2 Portierung des Programms CountButtonAndBlink

Um das Programm *CountButtonAndBlink* auf die neue Situation mit dem Shield zu portieren, müssen die Ports verändert werden. Dem beigefügten Datenblatt des Shields ist zu entnehmen, dass der Port für die erste LED nun C1 ist und für den ersten Button D1. Entsprechend ändern sich die Bezeichnungen im Schaltplan, im UML Aktivitätsdiagramm, im Quellcode und selbstverständlich in der Simulation.

1. Analyse

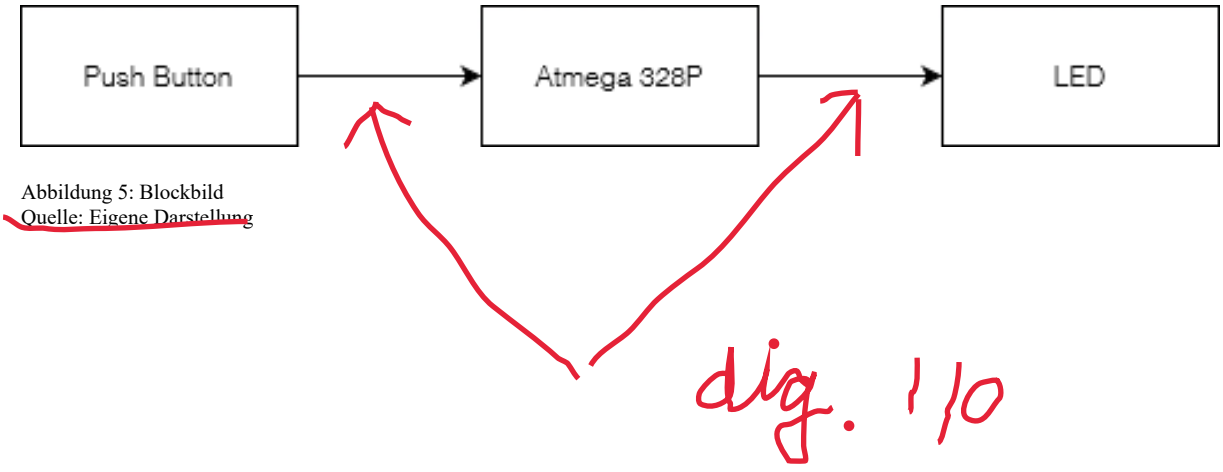

Abbildung 5: Blockbild
Quelle: Eigene Darstellung

2. HW Design mit Schaltplan

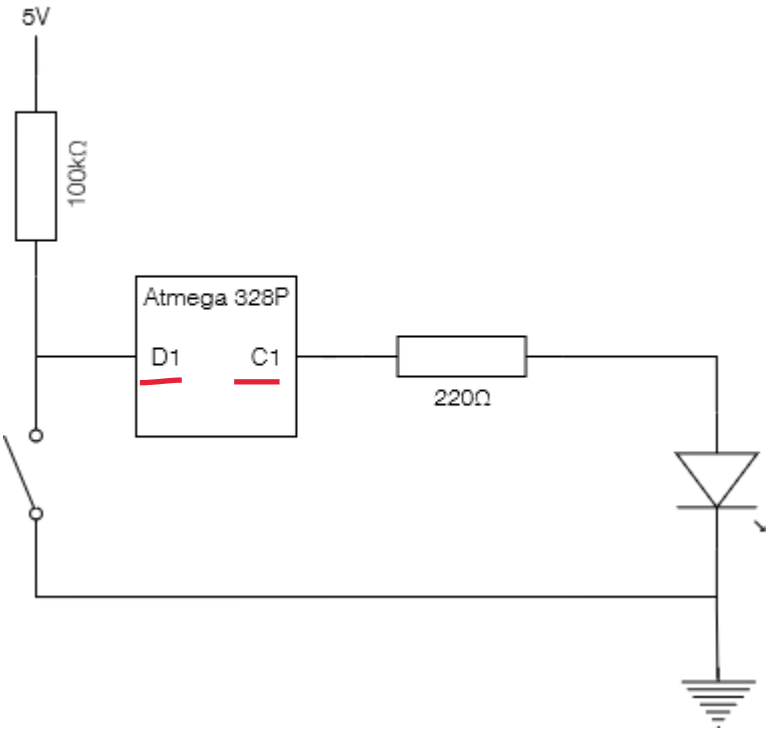

Abbildung 6: Schaltplan
Quelle: Eigene Darstellung

3. SW Design mit UML Aktivitätsdiagramm

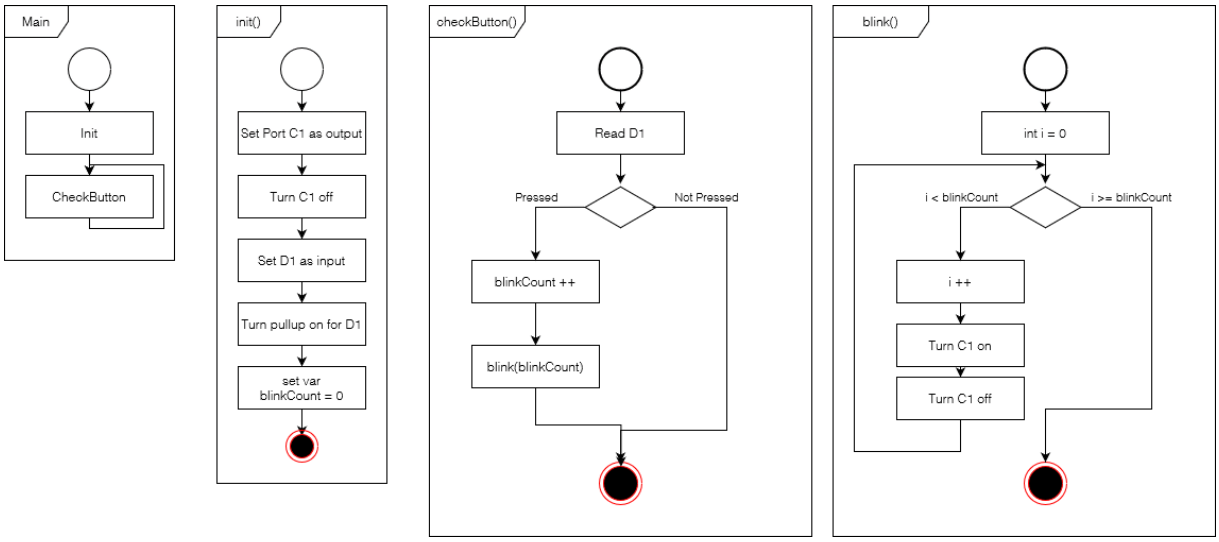

Abbildung 7: Aktivitätsdiagramm
Quelle: Eigene Darstellung

4. Implementieren in Microchip Studio

Mithilfe des Aktivitätsdiagramms haben wir den Code in Microchip Studio implementiert. Der Quellcode ist im Anhang zu finden.

5. Simulation in SimulIDE

Der Schaltplan wurde in SimulIDE nachgebaut und das Programm getestet. Das Programm hat funktioniert.

6. Anwendung in der Laborumgebung

Das Programm ist auch funktionsfähig, nachdem es auf die neue Situation mit dem Shield portiert wurde.

4.2.3 Experiment: Entfernung des Pullup Widerstandes des Push-Buttons

Nachdem wir den Pullup Widerstand des Push-Buttons entfernt haben, blinkt die LED dauerhaft ohne einen Druck auf den Button. In diesem Zustand ist der Pin an nichts angeschlossen, was in dieser Situation als low Spannung erkannt wird und einen Button-press auslöst.

4.2.4 Erweiterung des Programms: Reset von Anzahl der Blinks

Das Programm *CountButtonAndBlink* sollte erweitert werden, sodass die Anzahl der Blinks durch das Drücken von dem zweiten Button zurückgesetzt wird. Dafür muss ein zweiter Button initialisiert werden, dessen Port B1 ist. Außerdem benötigen wir eine neue Funktion, die prüft, ob der neue Button gedrückt wurde. Beim Drücken wird die Variable blinkCount auf 0 gesetzt.

1. Analyse

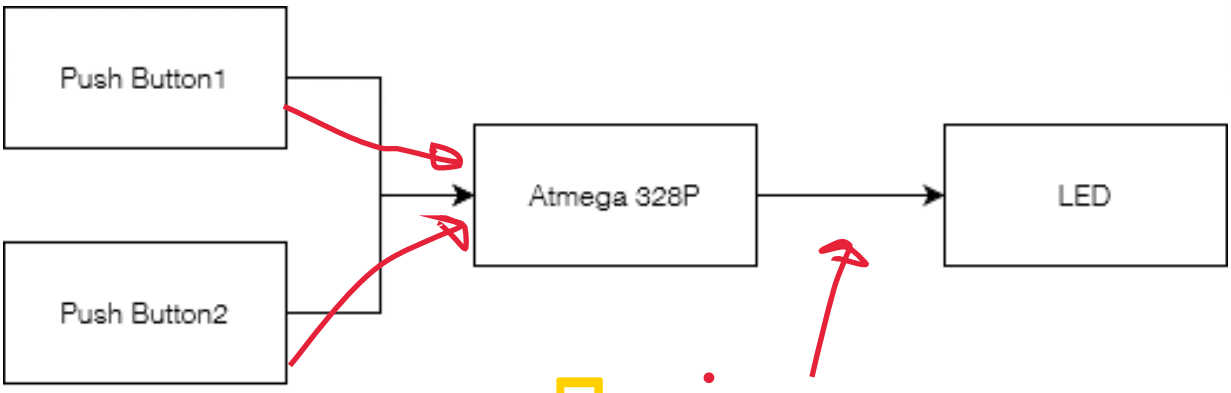

Abbildung 8: Blockbild
Quelle: Eigene Darstellung

2. HW Design mit Schaltplan

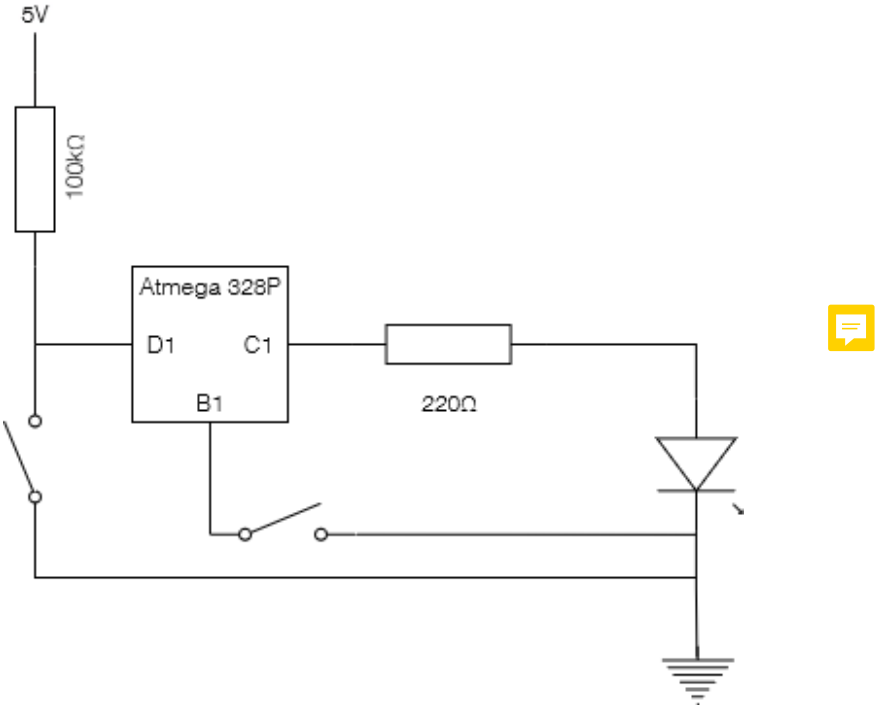

Abbildung 9: Schaltplan
Quelle: Eigene Darstellung

3. SW Design mit UML Aktivitätsdiagramm

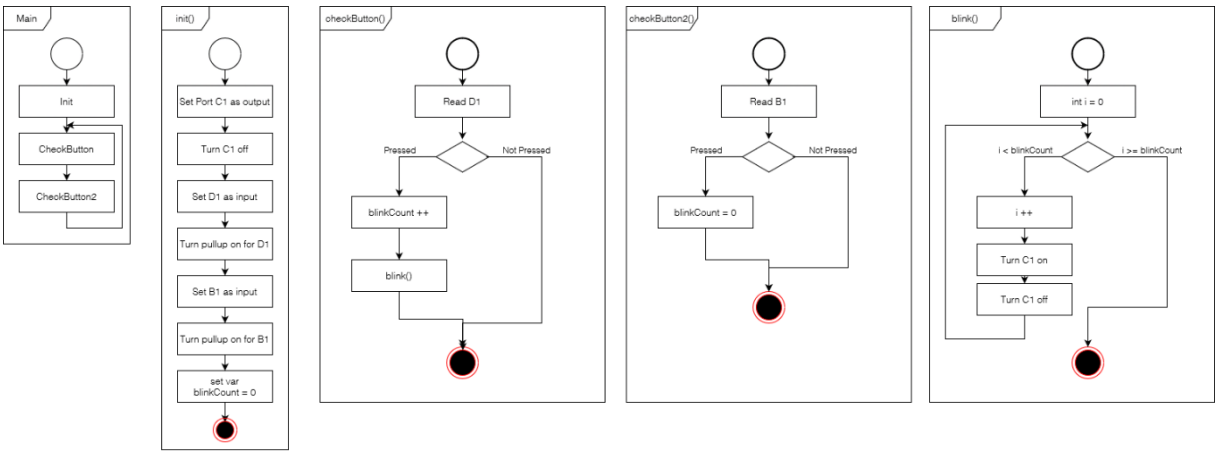

Abbildung 10: Aktivitätsdiagramm
Quelle: Eigene Darstellung

4. Implementieren in Microchip Studio

Mithilfe des Aktivitätsdiagramms haben wir den Code in Microchip Studio implementiert. Der Quellcode ist im Anhang zu finden.

5. Simulation in SimulIDE

Der Schaltplan wurde in SimulIDE nachgebaut und das Programm getestet. Das Programm hat funktioniert.

6. Anwendung in der Laborumgebung

Das Programm ist funktionsfähig. Der neue Button setzt die Anzahl der Blinks zurück.

4.2.5 Veränderung des Programms: Ausgabe der Blinks als Dualzahl

Das Programm *CountButtonAndBlink* sollte verändert werden, sodass die Anzahl der Blinks als vorzeichenlose Dualzahl ausgegeben wird. Dafür sollen die LEDs LED1GN, LED2GN, LED3GN und LED4GN verwendet werden. Die drei neuen LEDs müssen initialisiert werden. Sie laufen über die Ports C2, C3 und C4. Die Funktion blink() wird durch eine neue Funktion dualAusgabe() ersetzt, die sich der Logik bedient, welche bei der Umwandlung von Dezimalzahlen zu Dualzahlen genutzt wird.

1. Analyse

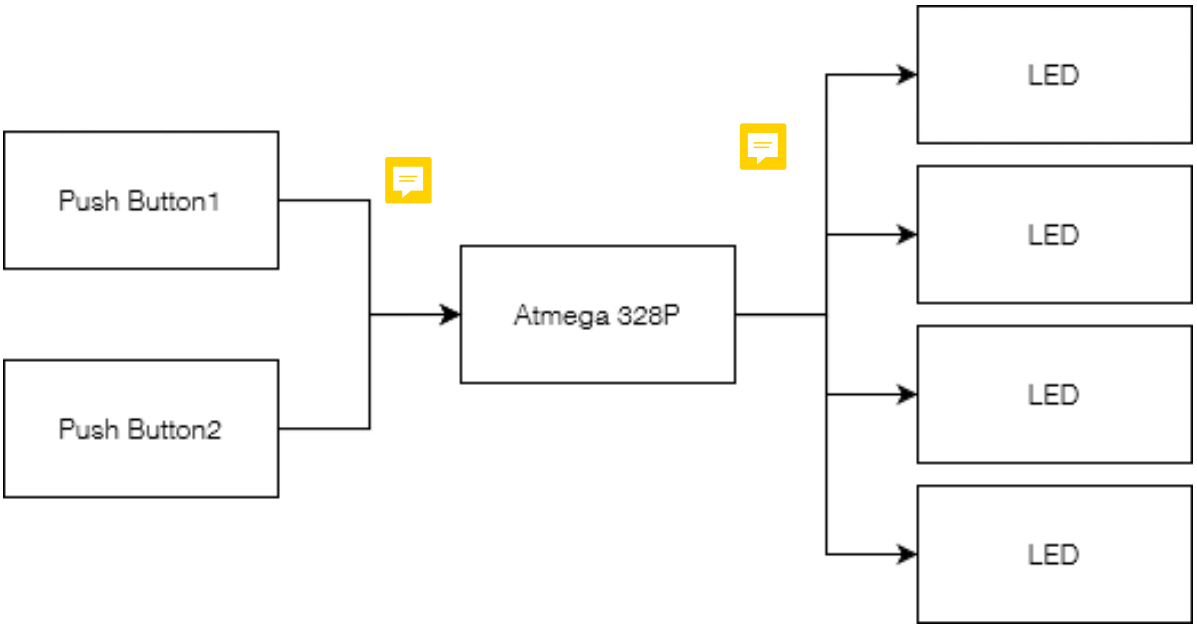

Abbildung 11: Blockbild
Quelle: Eigene Darstellung

2. HW Design mit Schaltplan

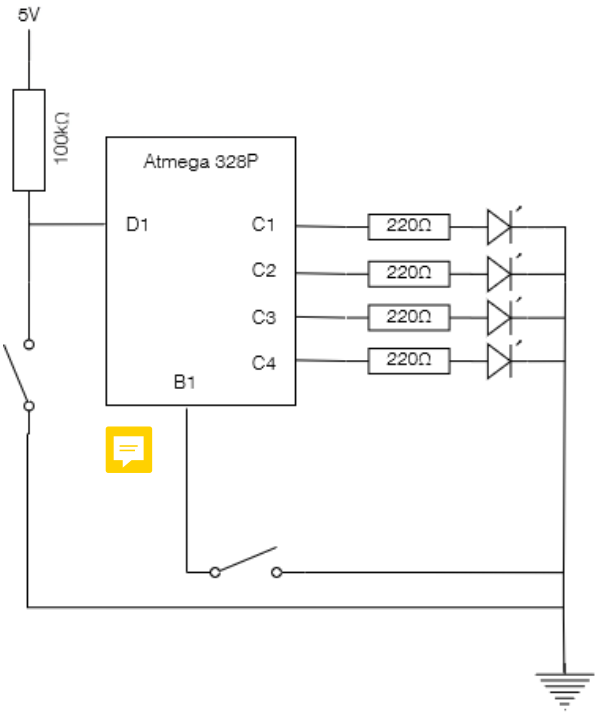

Abbildung 12: SchaltplanQuelle: Eigene Darstellung

3. SW Design mit UML Aktivitätsdiagramm

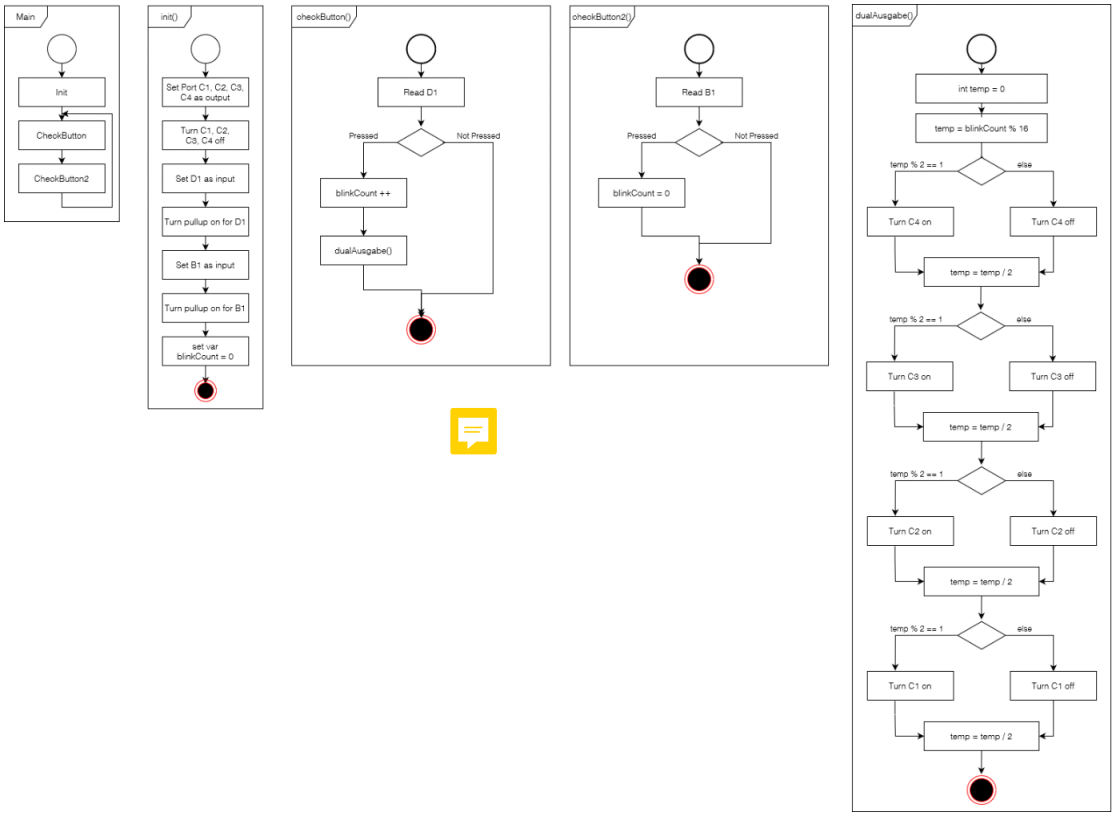

Abbildung 13: Aktivitätsdiagramm
Quelle: Eigene Darstellung

4. Implementieren in Microchip Studio

Mithilfe des Aktivitätsdiagramms haben wir den Code in Microchip Studio implementiert. Der Quellcode ist im Anhang zu finden.

5. Simulation in SimulIDE

Der Schaltplan wurde in SimulIDE nachgebaut und das Programm getestet. Das Programm hat funktioniert.

6. Anwendung in der Laborumgebung

Das Programm ist funktionsfähig. Der Anzahl der Blinks werden nun als vorzeichenlose Dualzahl auf den LEDs 1-4 ausgegeben.

4.3 Aufgabe 3: Interrupts

Es soll ein Programm geschrieben werden, welches beim Drücken des ersten Buttons alle grünen LEDs anschaltet und beim Drücken des zweiten Buttons alle roten LEDs. Dieses Programm haben wir *GNOrRTLEDOnButtonPress* genannt. Da bei diesem Programm 10 LEDs bedient werden, war es nützlich Macros einzusetzen, um die Leserlichkeit des Quellcodes nicht zu gefährden. Die 10 LEDs liegen auf den Ports C1, C2, C3, C4, C5, D4, D5, D6, D7 und D0.

Die Aufgabe sollte einmal mit dem Einsatz von Polling gelöst werden und einmal mit dem Einsatz von einem Pin Change Interrupt, sodass sich die Anforderungsanalyse für die zwei Einsatzmöglichkeiten ab dem 3. Schritt, dem SW Design mit UML Aktivitätsdiagramm, unterscheidet.

1. Analyse

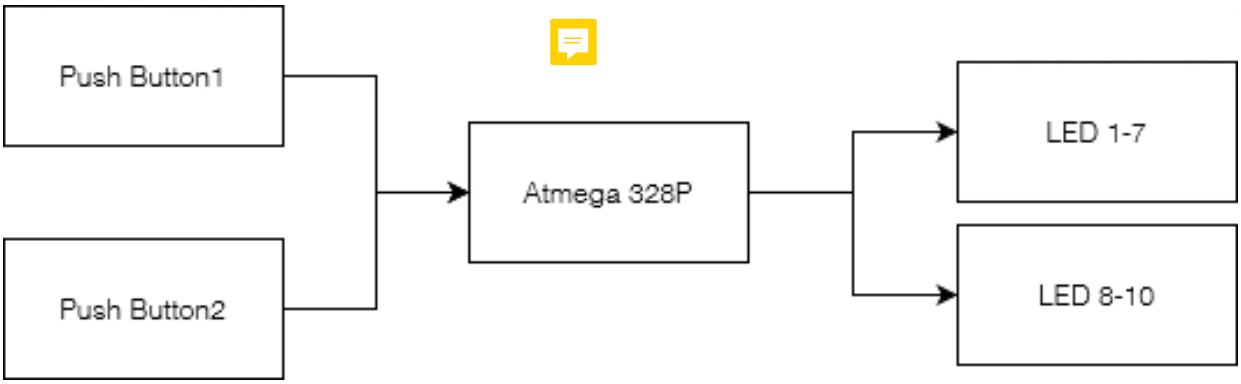

Abbildung 14: Blockbild
Quelle: Eigene Darstellung

2. HW Design mit Schaltplan

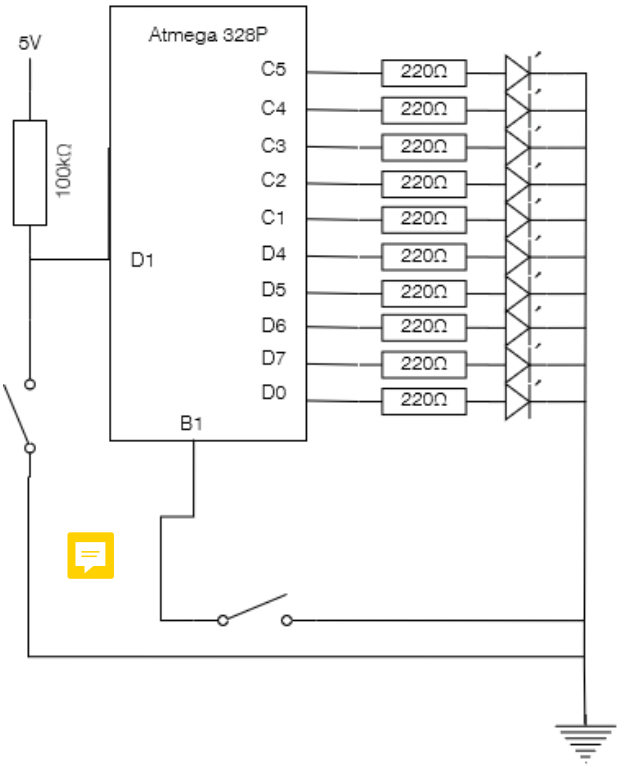

Abbildung 15: Schaltplan
Quelle: Eigene Darstellung

4.3.1 Verwendung von Polling

Die Funktionalität des Programms ist vom Prinzip unverändert. Es gibt zwei Funktionen, die prüfen, ob der jeweilige Button gedrückt wurde, wodurch entweder die grünen oder die roten LEDs leuchten.

3. SW Design mit UML Aktivitätsdiagramm

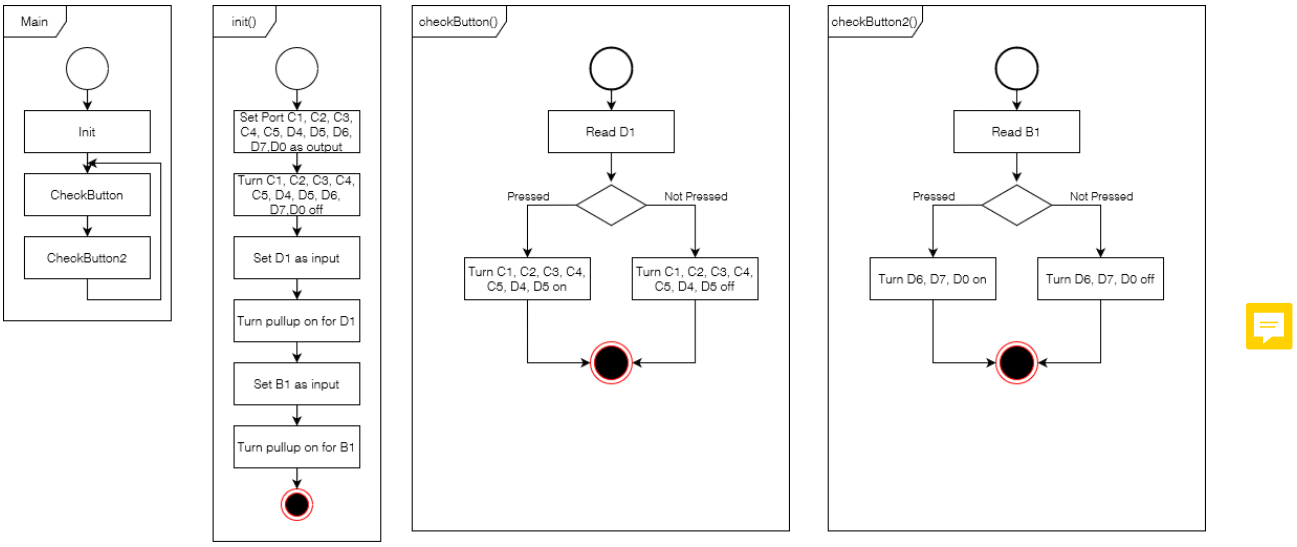

Abbildung 16: Aktivitätsdiagramm
Quelle: Eigene Darstellung

4. Implementieren in Microchip Studio

Mithilfe des Aktivitätsdiagramms haben wir den Code in Microchip Studio implementiert. Der Quellcode ist im Anhang zu finden.

5. Simulation in SimulIDE

Der Schaltplan wurde in SimulIDE nachgebaut und das Programm getestet. Das Programm hat funktioniert.

6. Anwendung in der Laborumgebung

Das Programm konnte noch nicht in der Laborumgebung getestet werden. Die Abnahme erfolgt an unserem nächsten Labortermin.

4.3.2 Verwendung eines Pin Change Interrupt

Für diese Methode wurde das globale Interrupt Bit gesetzt, ein Pin Change Interrupt Bit für Port B und Port D und ein PC Interrupt Bit auf pin B1 und pin D1. Sobald ein Button Change interrupt auf Port B oder Port D eingeht, werden die grünen bzw. die roten LEDs angeschaltet.

3. SW Design mit UML Aktivitätsdiagramm

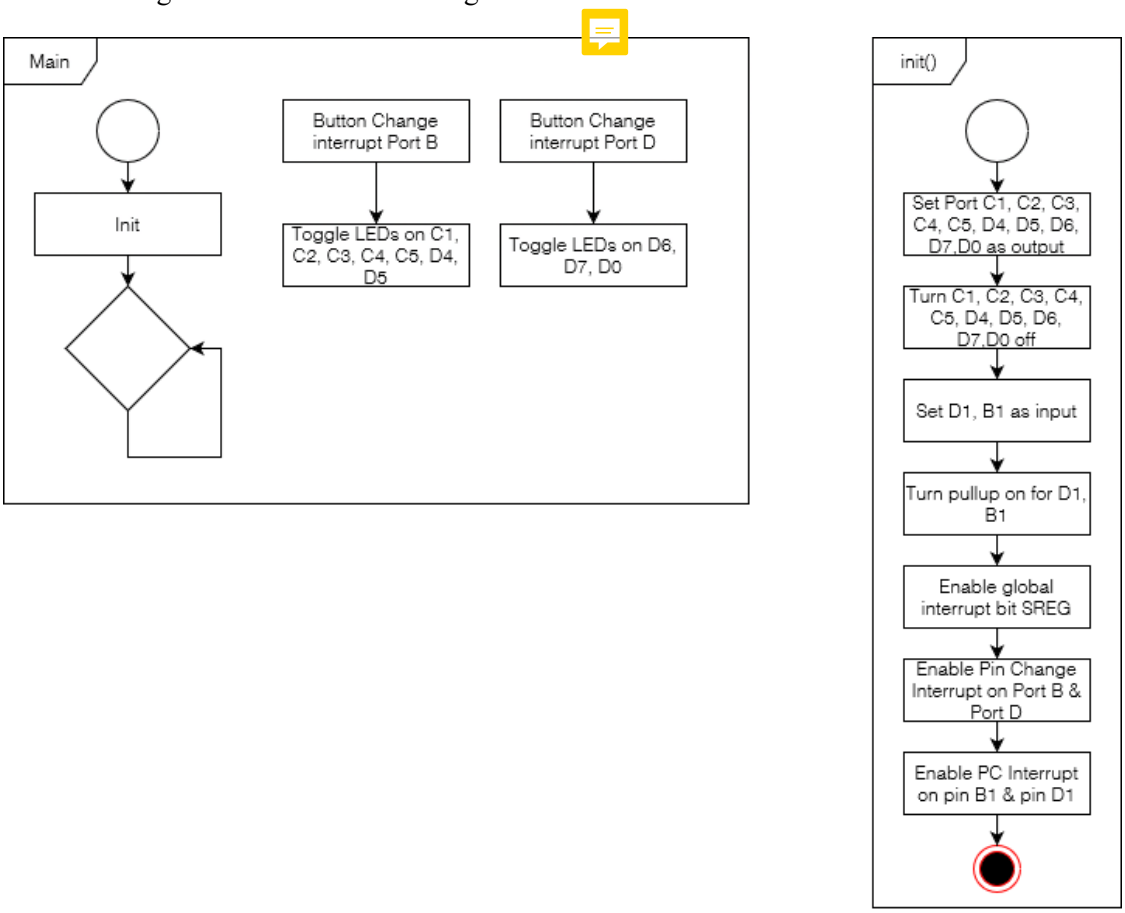

Abbildung 17: Aktivitätsdiagramm
Quelle: Eigene Darstellung

4. Implementieren in Microchip Studio

Mithilfe des Aktivitätsdiagramms haben wir den Code in Microchip Studio implementiert. Der Quellcode ist im Anhang zu finden.

5. Simulation in SimulIDE

Der Schaltplan wurde in SimulIDE nachgebaut und das Programm getestet. Das Programm hat funktioniert.

6. Anwendung in der Laborumgebung

Das Programm konnte noch nicht in der Laborumgebung getestet werden. Die Abnahme erfolgt an unserem nächsten Labortermin.

5 Fazit

Die mikrocontrollerbasierten Problemstellungen wurden ausführlich mithilfe der Anforderungsanalyse vorbereitet. Dadurch beherrschen wir nun die Methoden der Analyse, des Designs und der Implementierung.

Die ersten beiden Aufgaben funktionierten auf Anhieb in der für uns neuen Laborumgebung, sodass wir unser Ziel für diese Aufgaben erreicht haben.

Allerdings haben wir die Aufgabe 2d.: *Ausgabe der Blinks als Dualzahl* rückblickend zu kompliziert gelöst. In Zukunft würden wir diese Aufgabe auch mit konkreteren Macros lösen, um eine bessere Lesbarkeit und einen schnelleren Programmablauf zu gewährleisten.

Da die Aufgaben des Labors sehr zeitaufwendig waren, war es uns nicht möglich, die dritte Aufgabe vorzubereiten. Diese wollten wir im Rahmen des Labors fertig stellen. Leider führte auch dieses Vorhaben nicht zum Erfolg, sodass wir die letzte Aufgabe bei der nächsten Laborveranstaltung abnehmen lassen müssen.

Letztendlich ist zu betonen, dass die Aufgaben des ersten Labors sehr hilfreich waren, um die praktischen Fähigkeiten zum Umgang mit dem Thema I/O Ports zu erlangen und das Wissen aus den bisherigen Vorlesungen zu vertiefen.

Anhang

Quelltexte:

Zu 4.1.2*: CountButtonAndBlink*

Zu 4.2.2 und 4.2.4: *CountButtonAndBlink_Shield*

Zu 4.2.5*: CountButtonAndBlink_Shield_DualAnzeige*

Zu 4.3.1: *GNOrRTLEDOnButtonPress_Polling*

Zu 4.3.2*: GNOrRTLEDOnButtonPress_Interrupt*

Eigenständigkeitserklärung

Hiermit versichere ich/ versichern wir, dass ich/wir das vorliegende Dokument selbständig und nur mit den angegebenen Hilfsmitteln verfasst habe. Alle Passagen, die ich/wir wörtlich aus der Literatur oder aus anderen Quellen wie z. B. Internetseiten übernommen habe/n, habe/n ich/wir deutlich als Zitat mit Angabe der Quelle kenntlich gemacht.

19.04.2022	
Datum	Unterschrift