Christopher O'Connor

BBBaden

M242

Mikroprozessoranwendungen realisieren

Inhaltsverzeichnis

[2 Abbildungsverzeichnis 2](#_Toc63068332)

[3 Management Summary 3](#_Toc63068333)

[3.1 Projektübersicht 3](#_Toc63068334)

[3.2 Lehrbetrieb 3](#_Toc63068335)

[3.3 Involvierte Personen 3](#_Toc63068336)

[4 Aufgabenstellung 4](#_Toc63068337)

[4.1 Ausgangslage 4](#_Toc63068338)

[4.2 Zielsetzung 4](#_Toc63068339)

[4.2.1 Ampelkonfiguration 5](#_Toc63068340)

[4.2.2 Shift-Register 5](#_Toc63068341)

[4.2.3 Hardware-Komponenten verbinden 5](#_Toc63068342)

[4.2.4 Queue 5](#_Toc63068343)

[4.2.5 Timed Interupts 5](#_Toc63068344)

[4.3 Erweiterungen 5](#_Toc63068345)

[4.4 Erkannte Risiken und Massnahmen 5](#_Toc63068346)

[4.4.1 Input & Output Erweiterung 5](#_Toc63068347)

[4.4.2 Queue Abhängigkeit 5](#_Toc63068348)

[4.4.3 Sensoren Input 6](#_Toc63068349)

[4.5 Mengengerüst 6](#_Toc63068350)

[4.6 Rahmenbedingungen 6](#_Toc63068351)

[4.6.1 Hilfsmittel 6](#_Toc63068352)

[4.6.2 Vorkenntnisse 6](#_Toc63068353)

[4.6.3 Arbeitsumgebung 6](#_Toc63068354)

[5 Projektplanung 6](#_Toc63068355)

[5.1 Zeitplan 7](#_Toc63068356)

[6 Analyse / Entscheidung 8](#_Toc63068357)

[7 Realisierung 9](#_Toc63068358)

[7.1 Shift Register Out (LEDs) 9](#_Toc63068359)

[7.2 Ampel Konfiguration 10](#_Toc63068360)

[7.2.1 Konfigurationen 10](#_Toc63068361)

[7.3 Shift Register In (Sensoren / Schalter) 11](#_Toc63068362)

[7.4 Hardware-Komponenten verbinden 11](#_Toc63068363)

[7.5 Queue 12](#_Toc63068364)

[7.5.1 Neuer Eintrag 12](#_Toc63068365)

[7.5.2 Eintrag entnehmen 13](#_Toc63068366)

[7.5.3 Keine Einträge 13](#_Toc63068367)

[7.6 Timed Interrupts 13](#_Toc63068368)

[8 Testing 14](#_Toc63068369)

[8.1 Testfallspezifikation 14](#_Toc63068370)

[8.2 Testprotokoll 14](#_Toc63068371)

[9 Auswertung 14](#_Toc63068372)

[10 Arbeitsjournal 15](#_Toc63068373)

[11 Anhang 17](#_Toc63068374)

# Abbildungsverzeichnis

Abbildung 1 Skizze der Ampelsteuerung, Vogelperspektive 4

Abbildung 2 Shift Register Output 9

Abbildung 3 Shift Register Input 10

Abbildung 4 Vorlage Kreuzung 11

# Management Summary

## Projektübersicht

Diese Dokumentation und die daraus entstandene Präsentation sind Bewertungsgrundlagen für das Modul 242.

Nach den Standards des Projekt Managements werden folgende Projektphasen dokumentiert:

* Konzept
* Planung
* Analyse / Entscheidung
* Realisierung
* Testphase
* Auswertung

Zusätzliche Informationen, welche in der nachfolgenden Dokumentation zu finden sind:

* Abbildungsverzeichnis
* Q & A
* Arbeitsjournal
* Anhang

## Lehrbetrieb

SANTIS Training AG

Hohlstrasse 550

8048 Zürich

[training@santismail.ch](mailto:training@santismail.ch)

## Involvierte Personen

|  |  |  |
| --- | --- | --- |
| Auszubildender: | Christopher O’Connor | [christopher.oconnor@santismail.ch](mailto:christopher.oconnor@santismail.ch) |
| Lehrperson: | Manuel Bachofner | [manuel.bachofner@bbbaden.ch](mailto:manuel.bachofner@bbbaden.ch) |

# Aufgabenstellung

## Ausgangslage

Ich habe mich schon immer gefragt wie eine Ampelsteuerung funktioniert. Ich bin auch davon überzeugt, dass nicht alle gleich sind. Ich habe mich dazu beschlossen meine eigene zu realisieren in einem Arduino Projekt. Ich erhoffe mir dadurch besser zu verstehen können, wie sie funktionieren könnten.

## Zielsetzung

Mit dem Arduino Projekt soll eine komplexe automatisierte Ampelsteuerung von einer Kreuzung realisiert werden. Dazu werden Sensoren verwendet welche Fahrzeuge frühzeitig erkennen und die Ampeln entsprechend Auslastung korrekt umschalten. Damit auch Fussgänger die Möglichkeit haben die Strasse sicher zu überqueren, gibt es pro Strassenübergang einen Schalter wie bei einer echten Kreuzung. Wird dieser gedrückt, wird auf diese entsprechend Rücksicht genommen.

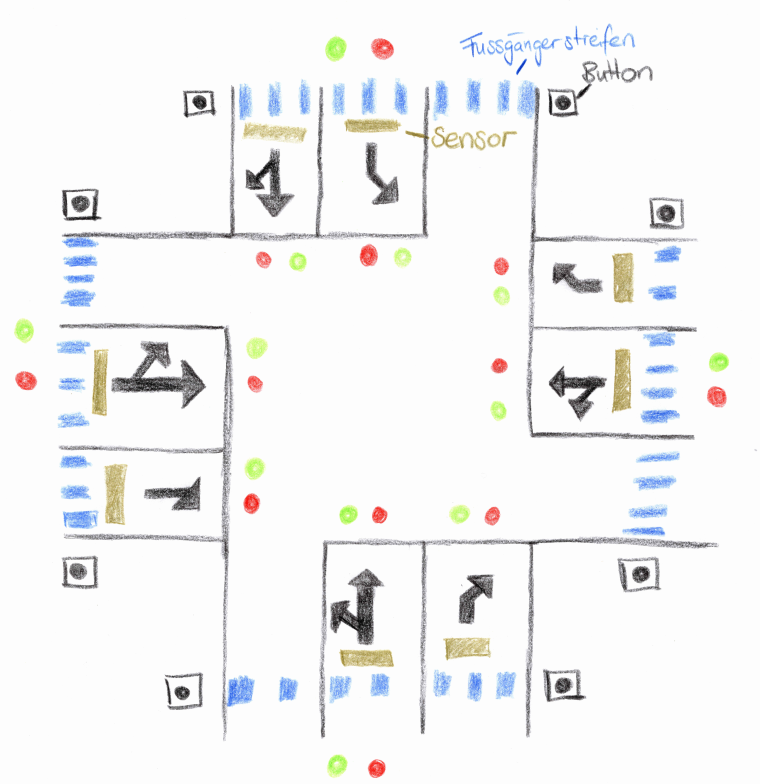
Nachfolgend eine Hand-Skizze der Kreuzung der Vogelperspektive:  


Abbildung Skizze der Ampelsteuerung, Vogelperspektive

### Ampelkonfiguration

Bei einer Kreuzung können mehrere Ampeln gleichzeitig Grün sein. Diesbezüglich muss die Abhängigkeit ausfindig gemacht und notiert werden. Damit auch andere zu einem späteren Zeitpunkt die Konfigurationen nachvollziehen können benötigt es noch eine gute Naming Convention der einzelnen Ampeln.

### Shift-Register

Die Anzahl an Input & Outputs sind auf einem Arduino begrenzt. Da das Ampelsystem über 40 Outputs und 16 Inputs verfügt muss mit 5x Shift-Register die Verfügbaren Slots erweitert werden.

### Hardware-Komponenten verbinden

Die Hardware (Arduino, Shift-Register, LEDs, Sensoren und Schalter) müssen auf mehrere Breadboards korrekt miteinander verbunden werden.

### Queue

Eine Queue oder etwas Ähnliches wird verwendet, um die Informationen von den Sensoren zwischen zu speichern in einer Liste. Zum Beispiel Ampel 1 kommt ein Auto. Ampel 4 kommt ein weiteres. Fussgänger-Schalter 4 wurde gedrückt. Die Queue wird dann vom Programm fortlaufend abarbeitet und schaltet die Kreuzung entsprechend.

### Timed Interupts

Die Sensoren müssen in der Lage sein «gleichzeitig» Inputs zu lesen und entsprechend in die Queue zu setzten, während eine Ampelkonfiguration aktiv ist. Mit den entsprechenden «Timed Interupts» kann dies ermöglicht werden.

## Erweiterungen

Sollte noch genügen Zeit vorhanden sein, könnten folgende Ideen noch zusätzlich realisiert werden:

* Orange LED als Bestätigung, dass Fussgänger den Knopf gedrückt haben.
* Remote Controll für eine Buslinie. Wird diese betätigt, hat der Bus Priorität vor der Queue.
* Rot Blitzer

## Erkannte Risiken und Massnahmen

### Input & Output Erweiterung

Die Anzahl an Input & Outputs sind auf einem Arduino begrenzt. Da das Ampelsystem über 40 Outputs und 16 Inputs verfügt muss mit 5x Shift-Register die Verfügbaren Slots erweitert werden.

### Queue Abhängigkeit

Eine Queue oder etwas Ähnliches wird verwendet, um die Informationen von den Sensoren zwischen zu speichern. Weil die Ampeln aber voneinander abhängig sind, sprich es ist nicht nur immer eine Ampel grün, sondern mehrere gleichzeitig, muss dies in der Queue berücksichtigt werden damit eine Ampelkonfiguration von mehreren einzelnen Ampeln nicht mehrfach in der Queue ist. Beispiel:

Ampel 1 & 2 haben gleichzeitig grün und bei beiden wurde ein Auto erkannt. In der Queue wird aber nur 1x ein Eintrag gemacht und nicht Ampel 1 und Ampel 2 zweimal nacheinander auf grün geschaltet.

### Sensoren Input

Die Sensoren müssen in der Lage sein «gleichzeitig» Inputs zu lesen und entsprechend in die Queue zu setzten, während eine Ampelkonfiguration aktiv ist.

Leider wird Multithreading vom Arduino nicht unterstützt. Hier heisst die Lösung «Timerinterrupt». Dadurch kann Multithreading ähnliches verhalten simuliert werden.

## Mengengerüst

|  |  |  |
| --- | --- | --- |
| Anzahl | Produkt | Beschreibung |
| 1x | Arduino UNO REV3 |  |
| 8x | HC-SR501 | PIR Motion Detection Sensor |
| 3x | 74HC595 | Shift Register 8bit Output |
| 2x | 74HC165 | Shift Register 8bit Input |
| 8x | Schalter / Button |  |
| 12x | LED-Grün | Grüne Ampel |
| 12x | LED-Rot | Rote Ampel |
| 4x | LED-Orange |  |
| 4x | Breadboard |  |

## Rahmenbedingungen

### Hilfsmittel

Zur Realisierung der Arbeit sind sämtliche Hilfsmittel erlaubt, welche im Rahmen einer LB zugelassen sind. Inbegriffen sind sowohl bestehende Dokumentationen, Code und Literatur als auch das Internet und andere, deklarierte Quellen. Befragung von Mitarbeiter und/oder externen Hilfspersonen müssen nachweisbar dokumentiert werden im Q&A Dokument.

Primäres Dokumentationsmittel ist Microsoft Word. Zusätzliche Hilfsprogramme wie PowerPoint, Notepad++ und AVR-Studio sind ebenfalls zugelassen. Bildschirmaufzeichnungen können mit der Software Snipping Tool durchgeführt werden.

### Vorkenntnisse

Eine transparente Bewertung der durchgeführten Projektarbeit erfordert die Preisgabe der fundierten Vorkenntnisse. Projektrelevante Vorkenntnisse werden stichwortartig festgehalten:

* ECDL Advanced Zertifikat
* Modul 121 Steuerungsaufgaben bearbeiten
* Programmiersprache C – Grundlagen

### Arbeitsumgebung

Primäre Arbeitsort ist ein Schulraum an der BBBaden. Ausweichungen ins HomeSchooling sind möglich.

# Projektplanung

Die Planung stellt ein wichtiges Hilfsmittel dar und wurde deshalb äusserst sorgfältig erstellt. Hier kann der Projektablauf mit der verfügbaren Zeit überprüft werden. Gegebenenfalls können Zeitknappheiten frühzeitig erkannt und angegangen werden.

## Zeitplan

asdf

# Analyse / Entscheidung

* Alle Varianten
* Wer hat was, wann und wie entschieden.

Mind. 4x!

# Realisierung

## Shift Register Out (LEDs)

Ein Shift Register bestehend aus 8bit und speichert entsprechend denn Zustand von jedem bit als 1 -ON oder 0 - OFF. Die Shift Register können aneinandergekoppelt werden. Für meine Kreuzung benötige ich genau 24 LEDs, 3x 8bit.

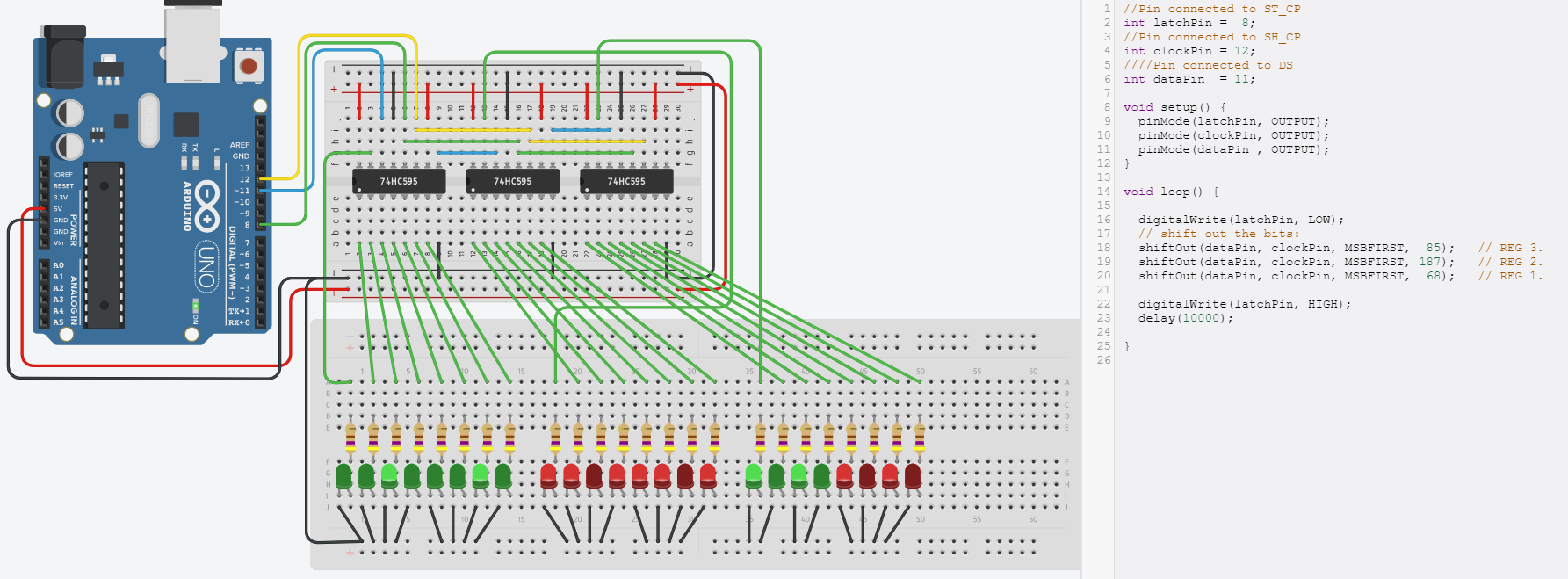


Abbildung Shift Register Output

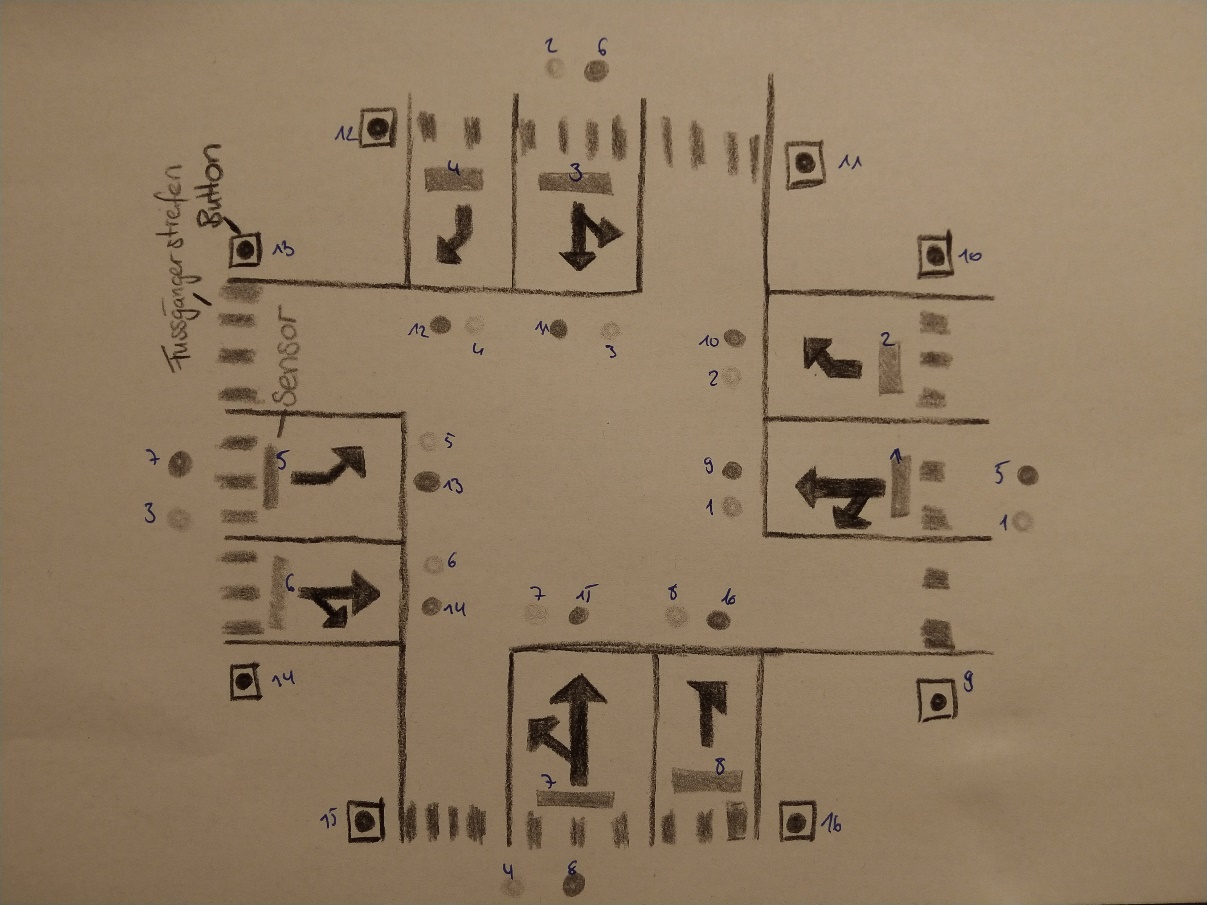
Ich habe mich dazu entschieden den ersten Shift Register für die grünen Ampeln, den zweiten Shift Register für die roten Ampeln und den dritten Shift Register für die Fussgänger Ampeln (0-4 grün / 4-8 rot) zu verwenden. Möchte man nun alle Ampeln auf Rot setzen wäre dies:

**[00000000] [11111111] [00001111]**

Die Standard Library von Arduino ermöglicht die Kommunikation mit einem Shift Register auch im Dezimalsystem. Dies ist zwar schwerer zu lesen aber weniger Fehleranfällig. Folglich wird aus den 3Byte: **[0] [255] [15]**

## Ampel Konfiguration

Bei einer Kreuzung können mehrere Ampeln gleichzeitig Grün sein. Diesbezüglich muss die Abhängigkeit ausfindig gemacht und notiert werden. Mithilfe der Skizze habe ich alle Scenarios durchgespielt und mir Notizen dazu gemacht. Ausgangspunkt war jeweils der Sensor welcher direkt mit einer Ampel abhängig ist. Danach konnte ich alle anderen Ampeln welche grün sind ausfindig machen. Aus nachfolgendem Bild kann die Nummerierung entnommen werden:



### Konfigurationen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | 1. Byte in Dez | 2. Byte in Dez | 3. Byte in Dez | Sensor/Schalter IDs |
| 0 | 0 | 255 | 240 | Default config |
| 1 | 34 | 221 | 240 | 1, 5 |
| 2 | 137 | 118 | 240 | 0, 3, 7 |
| 3 | 4 | 251 | 180 | 2, 12, 13 |
| 4 | 16 | 239 | 105 | 4, 8, 9, 14, 15 |
| 5 | 64 | 191 | 225 | 6 |
| 6 | 129 | 126 | 150 | 10, 11 |

## Shift Register In (Sensoren / Schalter)

Gleiches gilt für diesen Shift Register. Einer besteht aus 8bit Informationen und kann entsprechend 8 Sensoren oder Schalter verwalten. Für die Kreuzung werden 8 Sensoren zum Erfassen der Autos benötigt und 8 Schalter für die Fussgänger. Dies bedeutet es sind 2 Shift Register aneinandergekoppelt. In der Prototyprealisierung auf TinkerCad sieht die Schaltung wie folgt aus:

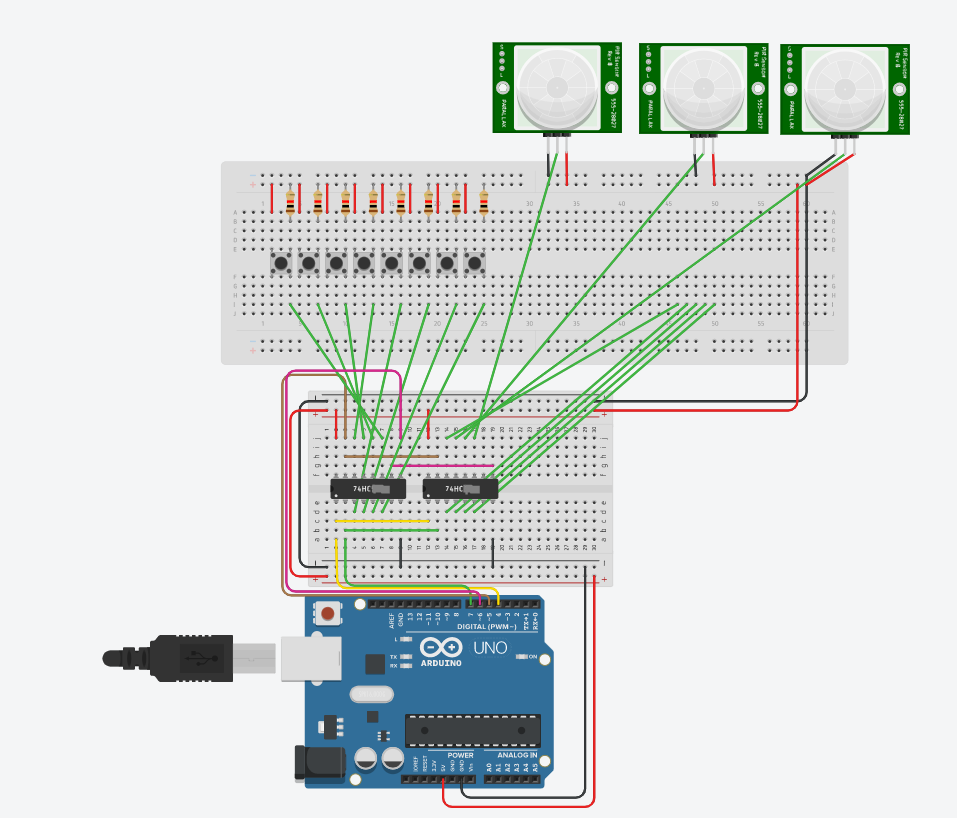


Abbildung Shift Register Input

Die fehlenden 5 Inputs sind auch PIR Sensoren. Aus Platzmangel habe ich diese in den Prototypen nicht hinzugefügt.

Die Shift-Register funktionieren beim Input genau umgekehrt wie beim Output. Der Zustand der einzelnen Bits können zu einem beliebigen Zeitpunkt gespeichert und vom Arduino ausgelesen werden.

## Hardware-Komponenten verbinden

Ich habe alle Teil-Komponenten auf <https://www.tinkercad.com/> zusammengesetzt und getestet. Dies ermöglichte schnelle Anpassungen und eine gute Fehlersuche. Die daraus entstandenen Vorlagen werden für die reale Schaltung verwendet und daraus abgeleitet. Zu den beiden Shift-Register gehören noch ein Passendes Layout, welches der Kreuzung entspricht. Als Vorlage habe ich eine Strassenseite auf TinkerCad erstellt:

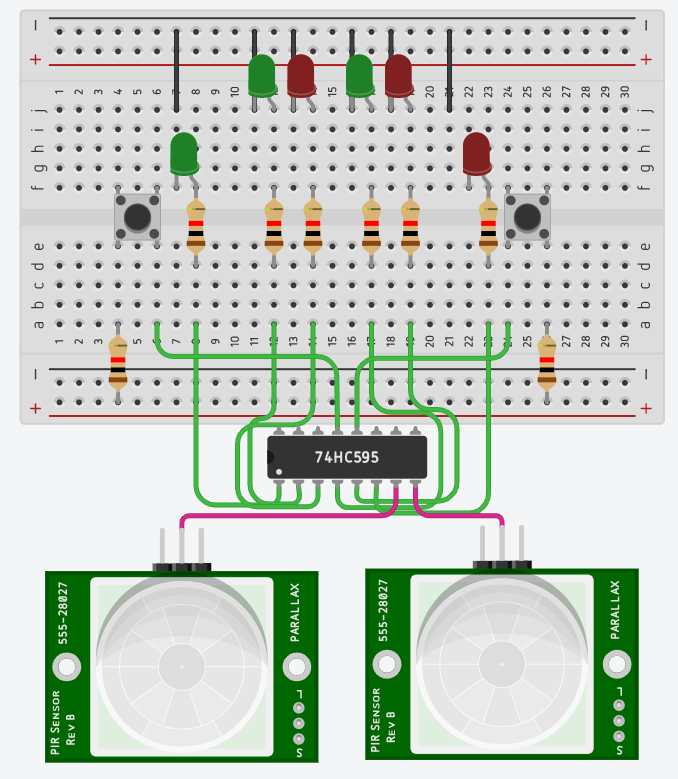


Abbildung Vorlage Kreuzung

Die Anschliessung am Shit-Register dient nur symbolisch. Die Widerstände imitieren zusätzlich den Zebrastreifen. Die beiden Schalter sind für die Fussgänger und die anschliessenden LEDs für die Ampeln Grün & Rot. Die zweiergruppen LEDs sind für Fahrbahn 1 und Fahrbahn 2 mit den entsprechenden Sensoren. Spiegelt man dieses kleine Circuit auf alle vier Himmelsrichtungen steht bereits die Kreuzung.

## Queue

Die Kreuzung benötigt 6 verschiedene Konfigurationen. Es können also entsprechend in der Queue maximal 6 Einträge sein. Ich habe mich für ein einfaches Array entschieden mit einem Index von 6. Das Array beinhaltet zu Beginn nichts, also: **{0, 0, 0, 0, 0, 0}**. Ein Eintrag entspricht einer Zahl von 1-6 referenzierend auf die 6 Konfigurationen.

### **Neuer Eintrag**

Für einen neuen Eintrag kann das Array von links nach rechts iteriert werden und jede Stelle mit dem neuen Eintrag verglichen werden. Ist dieser Eintrag noch nicht im Array vorhanden wird an der ersten freien Stelle (0) der Eintrag hinzugefügt.

### Eintrag entnehmen

Der Eintrag im Index [0] ist immer der älteste, weil die neuen Einträge am Ende hinzugefügt werden. Die Kreuzung schaltet entsprechend Eintrag um. Das Array muss danach jeden Index eins nach links verschieben um die Queue zu aktualisieren. Danach wiederholt sich das Ganze.

### Keine Einträge

Sind keine Einträge vorhanden wird Konfiguration 0 geladen. Für demonstrationszwecke und Überprüfung ist dies: Alle roten LEDs leuchten und alle grünen nicht.

## Timed Interrupts

Die Timed Interrupts werden benötigt, dass das Main Programm Inputs vom Shift Register entgegennehmen kann und diese in der Queue abspeichern kann, aber nicht sofort eine Änderung an der aktuellen Konfiguration vornimmt. Dadurch wird gewährleistet, dass die Ampeln eine definierte Zeit grün geschalten bleiben.

# Testing

## Testfallspezifikation

## Testprotokoll

# Auswertung

asdf

# Arbeitsjournal

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Datum | Benötigte Zeit | Arbeitsschritt | Bemerkungen | Visum |
| 05.01 | 60’ | Recherche | Teilkomponenten & Mögliche Probleme | CO |
| 05.01 | 120’ | Vorbereiten Doku | Aufgabenstellung + Mengengerüst | CO |
| 05.01 | 60’ | Planen | Grobe Planung des Projekts | CO |
| 12.01 | 60’ | Teilaufgaben definiert | Genauere Spezifikation von den Zielen und Unterteilung in unterziele | CO |
| 12.01 | 60’ | Zeitplan erstellen |  | CO |
| 12.01 | 200’ | Shift Register Output | Ich hatte Probleme eine gute Quelle zu finden. Viele Anleitungen / Erklärungen hatten falsche oder nicht vollständige Informationen und verwendeten andere Methoden oder gar andere Librarys. Das Umsetzen auf TinkerCad hat auch viel Zeit gekostet. | CO |
| 12.01 | 30’ | Shift Register Input | Circuit erstellt auf TinkerCad. Nach dem Auseinandersetzen und wirklichem verstehen des Outputs war der Input sehr einfach verständlich. | CO |
| 19.01 | 30’ | TinkerCad Kreuzung Design | Syntax überprüft auf TinkerCad | CO |
|  | 10’ | TinkerCad Queue | Konzept der Queue überprüft mit einfachem Input/Output Programm sowie Syntax. | CO |
|  | 5’ | TinkerCad Switch Case | Syntax überprüft auf TinkerCad | CO |
|  | 5’ | TinkerCad Timed Interrupt | Konzept überprüft mit einfachem Input/Output Programm sowie Syntax. | CO |
| 19.01 | 30’ | Shift Register Output | Auf Breadboard realisiert und als einzelne Komponente getestet. | CO |
| 19.01 | 30’ | PIR Testing | Sensoren getestet und Einstellungen vorgenommen (Reichweite & wie oft ein Signal gesendet wird). | CO |
| 19.01 | 30’ | Shift Register Input | Auf Breadboard realisiert und als einzelne Komponente getestet. | CO |
| 19.01 | 100’ | Kreuzung nachgebaut | LEDs, Sensoren, Widerstände & Schalter entsprechend Vorlage nachgebaut und alle Komponenten Verbunden. | CO |
| 19.01 | 60’ | Code entwickelt / angepasst | Den Code von den einzelnen Komponenten auf TinkerCad zusammengeführt und getestet. | CO |
| 19.01 | 30’ | Gorilla Testing |  | CO |
| 20.01 | 10’ | Videoaufnahme | Aufnahme des Produkts für die Abgabe. | CO |
| 01.02 | 100’ | Präsentation erstellt |  | CO |
| 01.02 | 20’ | Präsentation aufgenommen für LB |  | CO |
| 02.02 |  | Testing |  |  |
|  |  | Dokumentation |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Anhang