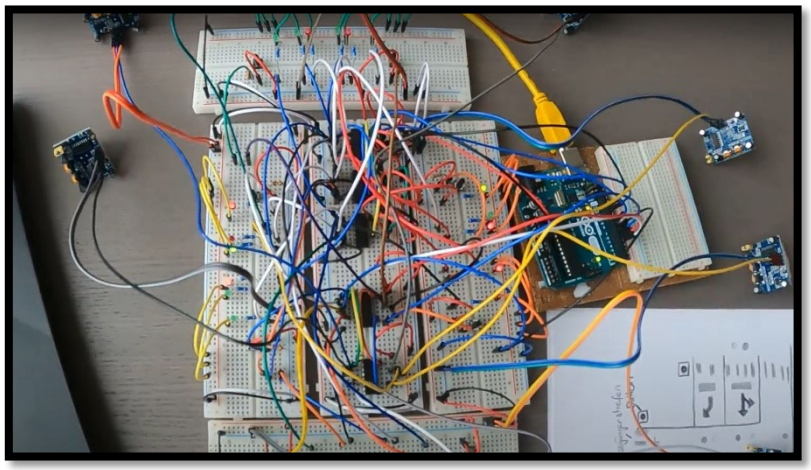
2.2.2021

M242

Mikroprozessoranwendungen realisieren



Christopher O'Connor
BBBADEN

1 INHALTSVERZEICHNIS

2 ABBILDUNGSVERZEICHNIS

3 MANAGEMENT SUMMARY

3.1 PROJEKTÜBERSICHT

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

Nach den Standards des Projekt Managements werden folgende Projektphasen dokumentiert:

- Informieren / Konzept
- Planung
- Entscheidung / Analyse
- Realisierung
- Kontrolle / Testphase
- Auswertung

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

- Abbildungsverzeichnis
- Arbeitsjournal
- Anhang
- Q & A

3.2 LEHRBETRIEB

SANTIS Training AG
Hohlstrasse 550
8048 Zürich
training@santismail.ch

3.3 INVOLVIERTE PERSONEN

Auszubildender:	Christopher O'Connor	christopher.oconnor@santismail.ch
Lehrperson:	Manuel Bachofner	manuel.bachofner@bbbaden.ch

4 AUFGABENSTELLUNG

4.1 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.

4.2 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:

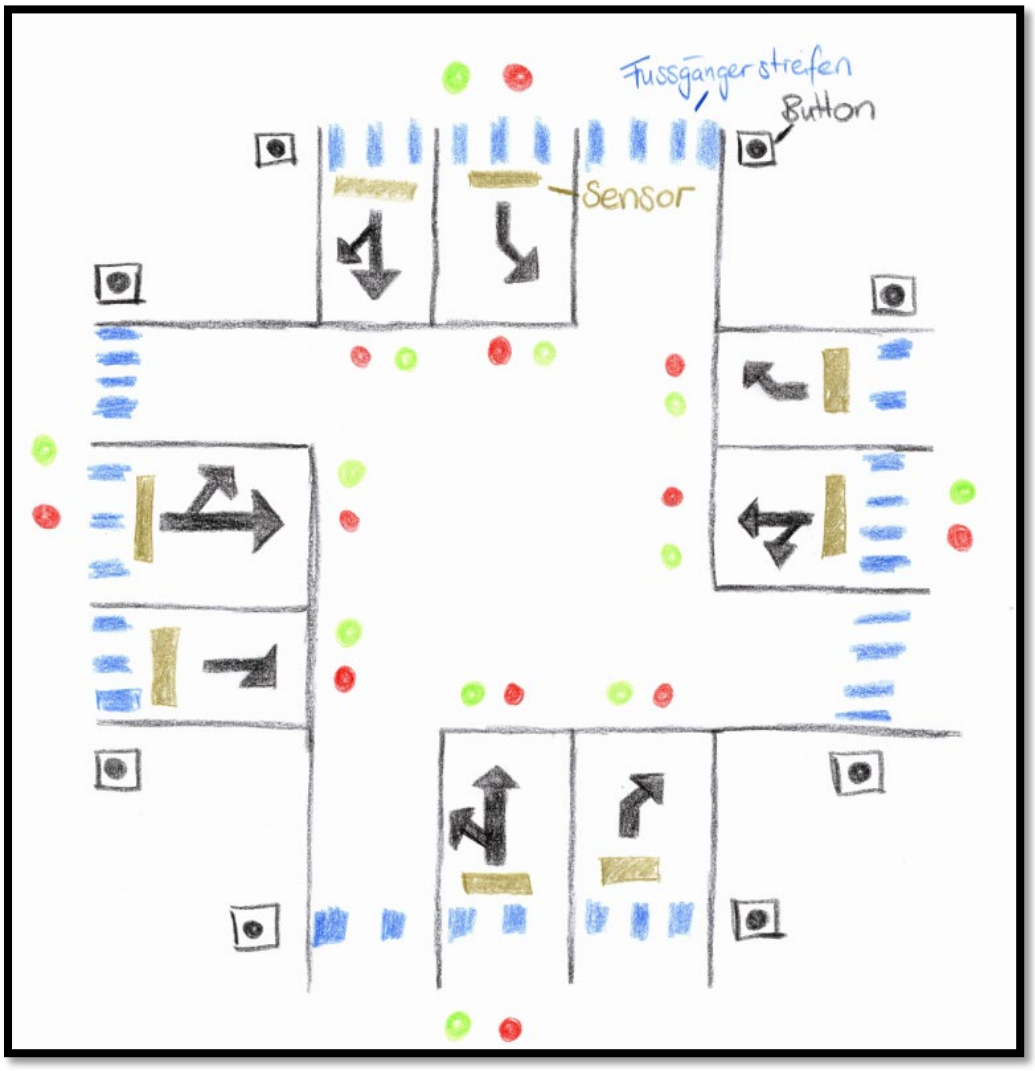


Abbildung 1 Skizze der Ampelsteuerung, Vogelperspektive

4.2.1 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.

4.2.2 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.

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

4.2.4 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.

4.2.5 Timer Interrupts
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 «Timer Interrupts» kann dies ermöglicht werden.

4.3 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-Control für eine Buslinie. Wird diese betätigt, hat der Bus Priorität vor der Queue.
- Rot Blitzer

4.4 Erkannte Risiken und Massnahmen

4.4.1 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.

4.4.2 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.

4.4.3 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 zu einem bestimmten Zeitpunkt etwas ausgeführt werden.

4.5 FLUSSDIAGRAMM

Nachfolgen soll ein Flussdiagramm vom möglichen Ablauf des Programmes aufklären:

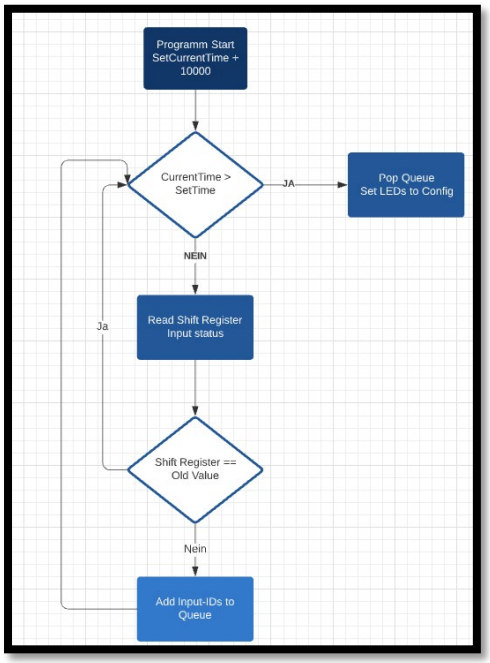


Abbildung 2 Flussdiagramm

4.6 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	Breadboard	

4.7 RAHMENBEDINGUNGEN

4.7.1 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.

4.7.2 Vorkenntnisse

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

- Modul 121 Steuerungsaufgaben bearbeiten
- Programmiersprache C – Grundlagen

4.7.3 Arbeitsumgebung

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

5 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.

5.1 ZEITPLAN

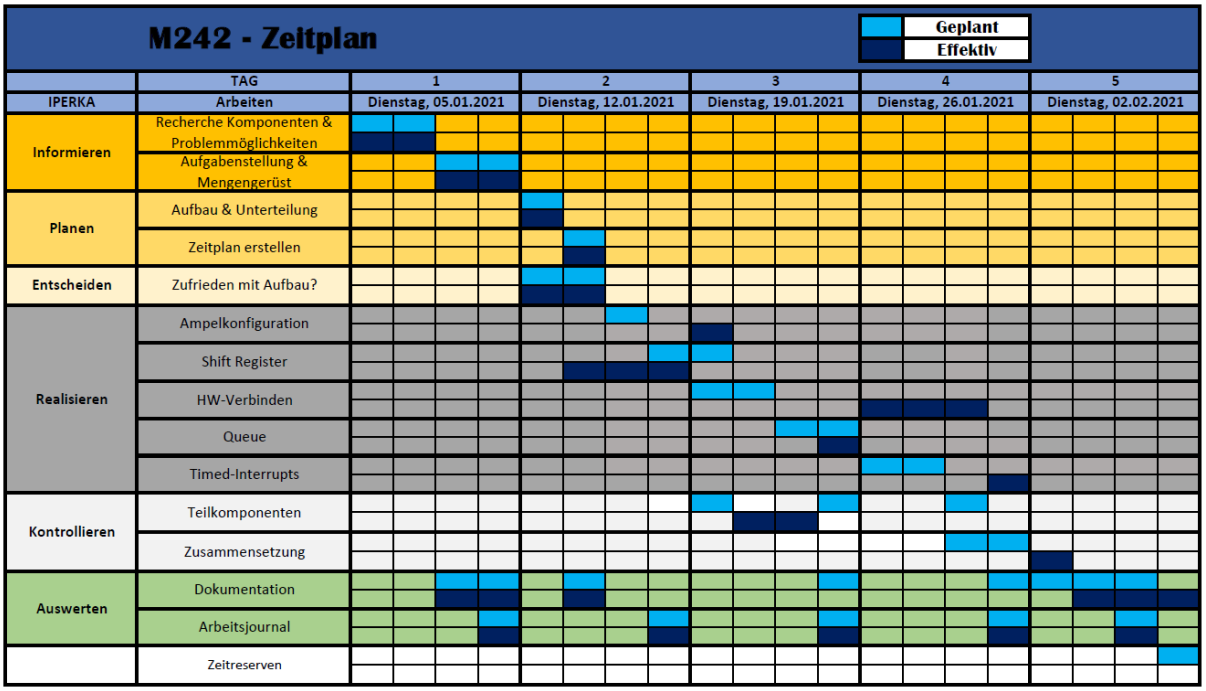


Abbildung 3 Zeitplan

6 Analyse / Entscheidung

6.1 Zu wenig Anschlüsse am Arduino?

1. Shift Register
2. Ansteuerbare LED-Streifen
3. EZ-Expander Shield

Die Kostengünstigste und von Herr Bachofner empfohlene Lösung hat mich überzeugt Shift Register zu verwenden. Dies ist die Komplexeste Lösung aber auch die Interessanteste! Diese funktionieren für den Output sowie für den Input. Zwei Probleme mit der Gleichen Lösung. Das EZ-Expander Shield ist bereits ein vorgefertigtes Shift Register mit einer Library nur für Output. [05.01.2021]

6.2 Wie wird die Queue realisiert?

1. Externe Library von einem Stack/Queue verwenden
2. Eigene Queue entwickeln mit einem Array

Ich habe mich dazu entschieden, eine eigene Queue zu entwickeln mit einem Array. Durch diverse Recherchen habe ich Bugs und Stack Overflow Reports zur externen Library gefunden. [05.01.2021]

6.3 Welche Sensoren können Kraftfahrzeuge erkennen?

1. PIR Sensor
2. Infrarot Distanzsensor

Der PIR Sensor hat mich bei Tutorials überzeugt. Mit diesem kann sehr einfach eine Bewegung erkannt werden. Die alternative mit einem Distanzsensor immer wieder zu messen und mit einen Referenzwert abzugleichen klingt sehr mühsam. [05.01.2021]

6.4 Wie viele LEDs und Sensoren sollen verwendet werden?

1. 2x LEDs pro Ampel (grün/rot), 2x Button pro Fussgängerstreifen & 1x Sensor pro Fahrbahn (8).
2. 3x LEDs pro Ampel (grün/orange/rot), 2x Button pro Fussgängerstreifen & 1x Sensor pro Fahrtrichtung (4).

Ich habe mich dazu entschieden die Variante 1. Zu verwenden. Das extra LED sprengt ansonsten den Rahmen und die Übersichtlichkeit nimmt stark ab. Zudem nur 1 Sensor pro Fahrtrichtung nicht wirklich sinnvoll ist und da die Unterteilung wichtig ist. [05.01.2020]

7 REALISIERUNG

7.1 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.

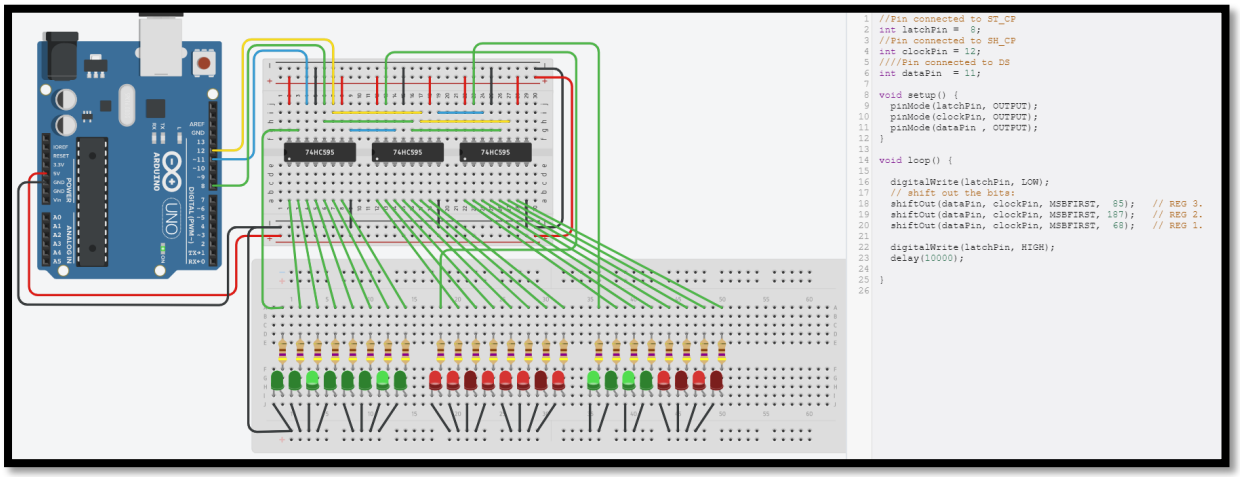


Abbildung 4 Shift Register Output

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]**

7.2 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:

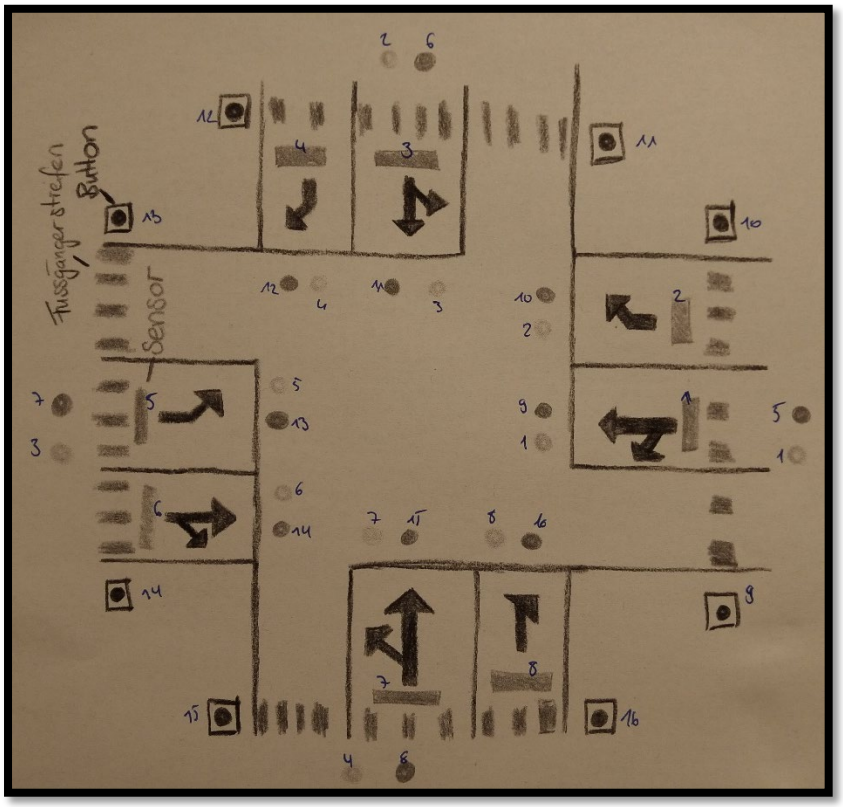


Abbildung 5 Ampelkonfiguration mit IDs

7.2.1 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

7.3 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:

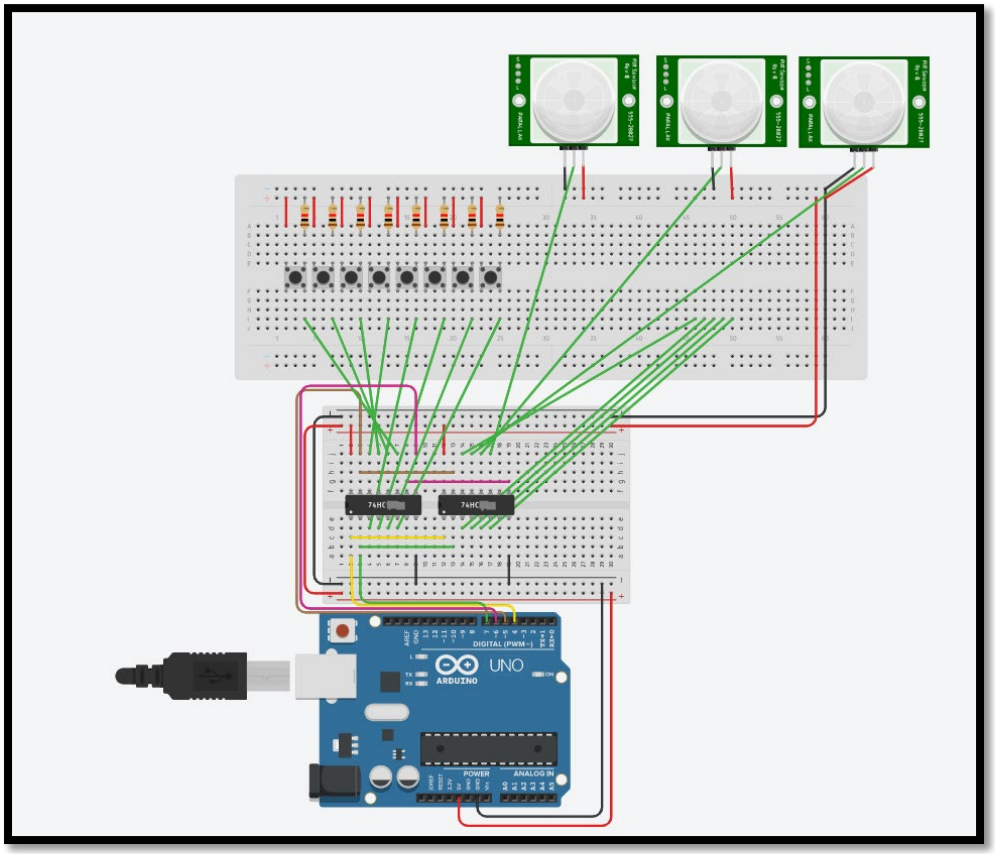


Abbildung 6 Shift Register Input

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.

7.4 Hardware-Komponenten verbinden

Ich habe alle Teil-Komponenten auf https://www.tinkercad.com/ zusammengesetzt und getestet. Dies ermöglicht eine schnelle Anpassung 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:

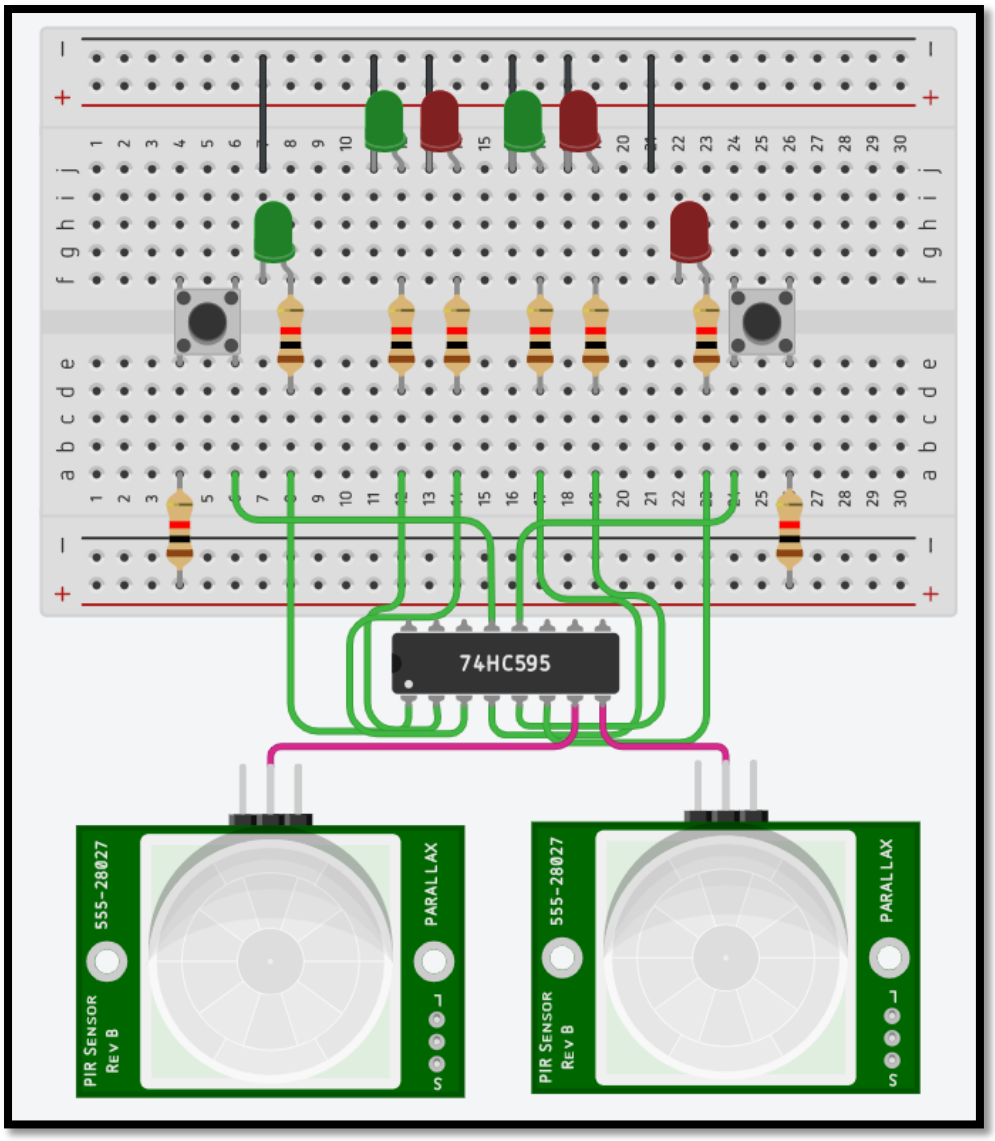


Abbildung 7 Vorlage Kreuzung

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.

7.5 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.

7.5.1 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.

7.5.2 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.

7.5.3 Keine Einträge

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

7.6 TIMER INTERRUPTS

Die Timer 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.

8 TESTING

8.1 TESTFALLSPEZIFIKATION

Testfallnummer	*01*
Getestete Anforderungen	1 (Starten + Default config)
Voraussetzung	Arduino + Programm starten
Eingabe	
Ausgabe	Alle rote LDs leuchten und alle grünen leuchten nicht.

Testfallnummer	*02*
Getestete Anforderungen	1 (Input von Sensoren)
Voraussetzung	Arduino + Programm starten
Eingabe	Sensor / Taster 1 Drücken -> Wiederholen für alle 1-16.
Ausgabe	Serial Monitor -> INPUT: [ID von Sensor] z.B. 1

Testfallnummer	*03*
Getestete Anforderungen	4 (Input, Queue hinzufügen, Queue abarbeiten, Output)
Voraussetzung	Arduino + Programm starten
Eingabe	Sensor / Taster aktivieren ID: 1 & 5
Ausgabe	Serial Monitor INPUT: 1 INPUT: 5 Q: 1 Showing Config: 1 -> LEDs überprüfen nach Skizze der Config.

Testfallnummer	*04*
Getestete Anforderungen	4 (Input, Queue hinzufügen, Queue abarbeiten, Output)
Voraussetzung	Arduino + Programm starten
Eingabe	Sensor / Taster aktivieren ID: 0, 3, 7
Ausgabe	Serial Monitor INPUT: 0 INPUT: 3 INPUT: 7 Q: 2 Showing Config: 2 -> LEDs überprüfen nach Skizze der Config.

Testfallnummer	*05*
Getestete Anforderungen	4 (Input, Queue hinzufügen, Queue abarbeiten, Output)
Voraussetzung	Arduino + Programm starten
Eingabe	Sensor / Taster aktivieren ID: 2, 12, 13
Ausgabe	Serial Monitor INPUT: 2 INPUT: 12 INPUT: 13 Q: 3 Showing Config: 3 -> LEDs überprüfen nach Skizze der Config.

Testfallnummer	*06*
Getestete Anforderungen	4 (Input, Queue hinzufügen, Queue abarbeiten, Output)
Voraussetzung	Arduino + Programm starten
Eingabe	Sensor / Taster aktivieren ID: 4, 8, 9, 14, 15
Ausgabe	Serial Monitor INPUT: 4 INPUT: 8 INPUT: 9 INPUT: 14 INPUT: 15 Q: 4 Showing Config: 4 -> LEDs überprüfen nach Skizze der Config.

Testfallnummer	*07*
Getestete Anforderungen	4 (Input, Queue hinzufügen, Queue abarbeiten, Output)
Voraussetzung	Arduino + Programm starten
Eingabe	Sensor / Taster aktivieren ID: 6
Ausgabe	Serial Monitor INPUT: 6 Q: 5 Showing Config: 5 -> LEDs überprüfen nach Skizze der Config.

Testfallnummer	*08*
Getestete Anforderungen	4 (Input, Queue hinzufügen, Queue abarbeiten, Output)
Voraussetzung	Arduino + Programm starten
Eingabe	Sensor / Taster aktivieren ID: 10, 11
Ausgabe	Serial Monitor INPUT: 10 INPUT: 11 Q: 6 Showing Config: 6 -> LEDs überprüfen nach Skizze der Config.

Testfallnummer	*09*
Getestete Anforderungen	2 (Input, Queue, Timer Interrupt)
Voraussetzung	Arduino + Programm starten
Eingabe	Sensor / Taster aktivieren ID: 10, 3
Ausgabe	Serial Monitor INPUT: 10 INPUT: 3 Current Queue [10, 3, 0, 0, 0, 0] -> nach 10s : Current Queue [3, 0, 0, 0, 0, 0] -> nach 10s : Current Queue [0, 0, 0, 0, 0, 0]

Testfallnummer	*10*
Getestete Anforderungen	1 (Queue duplizierter Eintrag)
Voraussetzung	Arduino + Programm starten
Eingabe	Sensor / Taster aktivieren ID: 4, 8, 14
Ausgabe	Serial Monitor INPUT: 4 INPUT: 8 INPUT: 14 Current Queue [4, 0, 0, 0, 0, 0]

8.2 TESTPROTOKOLL

Testfall	Resultat	Datum	Tester	Bemerkung
01	Funktioniert	02/02/21	CO	-
02	Funktioniert	02/02/21	CO	-
03	Funktioniert	02/02/21	CO	-
04	Funktioniert	02/02/21	CO	-
05	Funktioniert	02/02/21	CO	-
06	Funktioniert	02/02/21	CO	-
07	Funktioniert	02/02/21	CO	-
08	Funktioniert	02/02/21	CO	-
09	Funktioniert	02/02/21	CO	-
10	Funktioniert	02/02/21	CO	-

9 AUSWERTUNG

9.1 ALLGEMEIN

Dieses Projekt forderte sehr viel organisatorisches Geschick und eine sehr gute Planung vor allem für die Verkabelung. Ich bin zufrieden mit dem Resultat. Es gibt aber einige Verbesserungsmöglichkeiten und ich konnte sehr viel Lernen im Projekt sowie wichtige Erfahrungen machen in der Präsentation und der Dokumentation. Für zukünftige Projekte werde ich auf jeden Fall mitnehmen, dass eine präzise Planung / Vorlage zwar selbst viel Zeit in Anspruch nimmt dafür aber die eigentliche Realisierung viel einfacher ist.

9.2 PLANUNGSDISKREPANZ

Vom Zeitplan mit soll- und ist- Einträgen kann man klar erkennen, dass nicht alles nach Plan verlief. Ich hatte einerseits Mühe damit, einzuschätzen wie lange ich für eine Tätigkeit habe, weil ich im vorab nur wusste was ich machen möchte z.B. Shift Register, ich aber keine Ahnung hatte was das ist, wie man es verwendet und wie komplex die Verkabelung ist. Zusätzlich kamen noch Lieferverspätungen der HW dazu sowie das Unterschätzen von wie viele Kabel es tatsächlich braucht.

9.3 TESTING

Die Strukturierte Testphase war für mich sehr einfach und alle Tests verliefen sofort nach Plan. Bevor ich mit dem Bestücken der Breadboards gestartet bin, habe ich bereits alle Teilaufgaben auf Tinkercad erstellt und getestet. Auch habe ich, während dem Realisieren bereits sehr viele Tests gemacht und Prototypen erstellt.

9.4 PRÄSENTATION

Durch die aktuelle Lage und HomeSchooling durfte ich die Präsentation im Lehrbetrieb machen und aufzeichnen. Ich konnte sehr viel dadurch Lernen in Bezug auf Präsentationen aufnehmen. Ich konnte leider folgende Erfahrungen machen:

1. Präsentation auf Videoaufnahme richten
   Hintergrund sollte dunkel sein und nicht hell sonst erkennt man Schriften im Video nicht.
2. Laserpointer nicht verwenden!
   Der Laserpointer erkennt man im Video nur auf einem dunklen Hintergrund.
3. Passende Kamera mit höherer Video- und Ton- Qualität.

10 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 Timer 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	40’	Präsentation aufgenommen für LB		CO
02.02	90’	Dokumentiertes Testing	Nach dem Prototyp-Testing und Gorilla-Testing benötigt es noch ein konkretes Testing nach Vorgabe.	CO
02.02	10’	Dokumentation	Zeitplan ergänzt und übertragen	CO
02.02	30’	Dokumentation	Source Code fertig dokumentiert und in Doku übertragen	CO
02.02	300’	Dokumentation	Auswertung + Feinschliff + Projektmanual überprüfen	CO

11 ANHANG

11.1 Q & A

11.1.1 Frage 1

Ich weiss noch nicht wie ich 24+ LEDs & 16+ Sensoren/Schalter mit nur einem Arduino gezielt ansteuern kann.

Antwort von M. Bachofner am 09.12.2020:
Für die Outputs für die LEDs wäre ein sogenanntes Shift-Register eine Möglichkeit.

11.1.2 Frage 2

Wenn ich die Ampeln Zeit gesteuert anlassen möchte, also einen Delay hinzufüge damit eine Ampel ein paar Sekunden grün bleibt, wie kann ich dann trotzdem die anderen Informationen der Sensoren der Queue zwischenzeitlich hinzufügen?

Antwort von M. Bachofner am 09.12.2020:
Interrupts ist hier das Stichwort. Sie bräuchten einen Timerinterrupt.

11.1.3 Frage 3

Am 08.01.2021 ist der letzte Modul Tag, wann ist der Abgabetermin für die Dokumentation (Zeit) und wann findet die Präsentation statt?

Antwort von M. Bachofner am 09.12.2020:
Abgabetermin ist der 02.02.2020 um 23.59Uhr. Sie müssen die Präsentation an diesem Tag halten. Der Rest der Klasse wird um 13.00 die Leistungsbeurteilung des Moduls 153 schreiben. Um 16.15Uhr machen Sie dann die Präsentation. Ich denke, das Vorgehen so ist ziemlich ideal.

11.1.4 Frage 4

Demo Video abgegeben: Reicht das Ihnen so oder haben Sie noch einen speziellen Wunsch, was im Video möglicherweise nicht ersichtlich ist?

Antwort von M. Bachofner am 09.12.2020:
Super, vielen Dank. Bitte verlinken Sie das Video noch in der Abgabe des Projektes. Ich werde alles Zusammen korrigieren. Ein Feedback zum Video kann ich ihnen noch nicht geben, da das ja Teil der LB ist.

11.2 VIDEO – DEMO

https://drive.google.com/file/d/1SFWNpjLfrfJSFlRaE-nkVhIn2RrcvT63/view?usp=sharing

11.3 VIDEO – PRÄSENTATION

https://drive.google.com/file/d/1lnSonvm0khwSzprJnFxAGBBbWiANGQaa/view?usp=sharing

11.4 Bild des Projekts

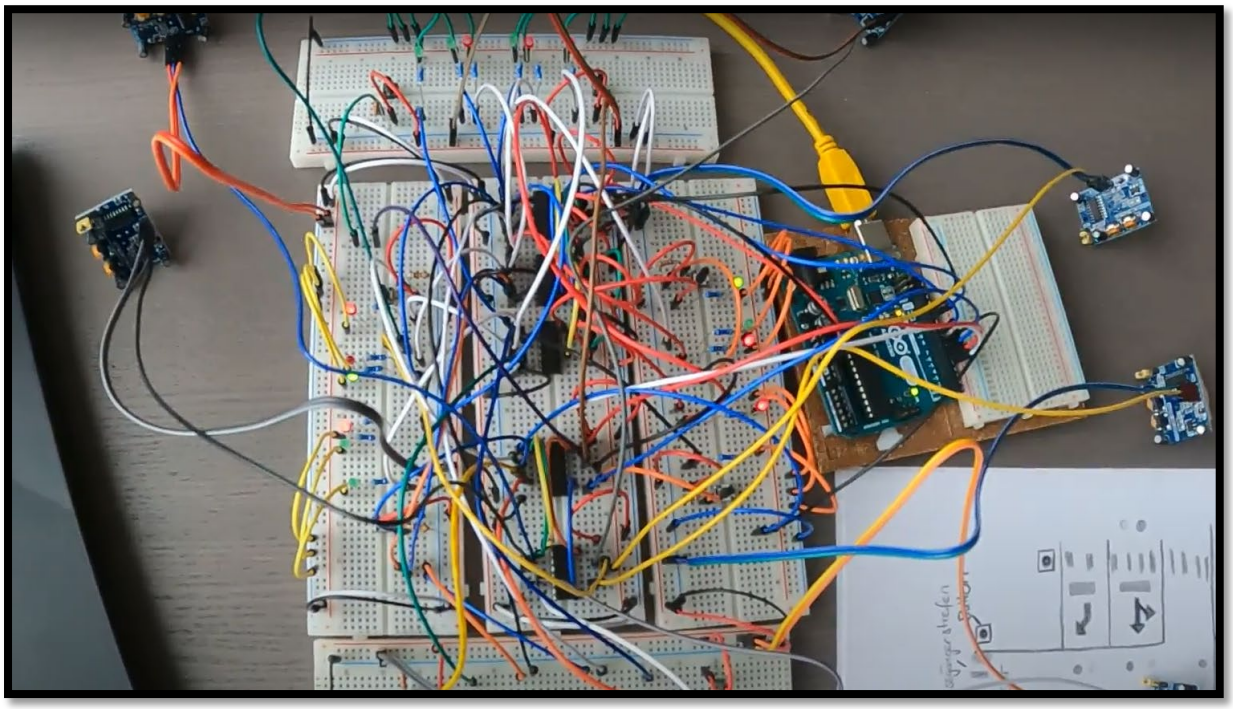


Abbildung 8 Abschlussbild der Realisierung

11.5 Source Code

```
/**
 *  BBBaden Modul 424
 *
 *     Author: O'Connor Chris In18z
 *  Date:     02/07/2021
 *
 */

// Shift Register Input (165)
int ploadPin        = 4;     // Connects to Parallel load pin
int clockEnablePin  = 5;     // Connects to Clock Enable pin
int dataPin         = 6;     // Connects to the Q7 pin
int clockPin        = 7;     // Connects to the Clock pin

// Shift Register Output (595)
int latchPinOUT =  8;        // Pin connected to ST_CP
int clockPinOUT = 12;        // Pin connected to SH_CP
int dataPinOUT  = 11;        // Pin connected to DS

// Queue and size
int qSize   = 6;
int queue[] = {0, 0, 0, 0, 0, 0};

// Timer Interrupt
unsigned long targetTime = millis() + 10000;

// Configurations for LEDs
int light_configs [7][3] = {{0, 255, 240}, {34, 221, 240}, {137, 118, 240}, {4, 251,
180}, {16, 239, 105}, {64, 191, 225}, {129, 126, 150}};

#define BYTES_VAL_T unsigned int
BYTES_VAL_T pinValues;
BYTES_VAL_T oldPinValues;

/*
 * This function is essentially a "shift-in" routine reading the
 * serial Data from the shift register chips and representing
 * the state of those pins in an unsigned integer.
 * Source: https://playground.arduino.cc/Code/ShiftRegSN74HC165N/
 */
BYTES_VAL_T read_shift_regs()
{
       long bitVal;
       BYTES_VAL_T bytesVal = 0;

       // Trigger a parallel Load to latch the state of the data lines,
       digitalWrite(clockEnablePin, HIGH);
       digitalWrite(ploadPin, LOW);
       delayMicroseconds(5);
       digitalWrite(ploadPin, HIGH);
       digitalWrite(clockEnablePin, LOW);

       /* Loop to read each bit value from the serial out line
       * of the SN74HC165N.
       */
       for(int i = 0; i < 16; i++)
       {
              bitVal = digitalRead(dataPin);

              // Set the corresponding bit in bytesVal.
              bytesVal |= (bitVal << ((16-1) - i));

              // Pulse the Clock (rising edge shifts the next bit).
              digitalWrite(clockPin, HIGH);
```

```
                delayMicroseconds(5);
                digitalWrite(clockPin, LOW);
        }

        return(bytesVal);
}

/*
 * Evaluate Input Pin Values.
 */
void eval_pin_values()
{
        for(int i = 0; i < 16; i++)
        {
                if((pinValues >> i) & 1 ) {
                  Serial.print("INPUT: ");
                  Serial.println(i);
                  mapInputToQueue(i);
                }

        }

        Serial.print("\r\n");
}

/*
 * Display Configuration on LEDs.
 * Source: https://www.arduino.cc/en/Tutorial/Foundations/ShiftOut
 */
void display_led(int arrIndex)
{
        Serial.print("Showing Config: ");
        Serial.println(arrIndex);

        digitalWrite(latchPinOUT, LOW);
        // shift out the bits:
        shiftOut(dataPinOUT, clockPinOUT, MSBFIRST,  light_configs[arrIndex][2]); // REG
3.
        shiftOut(dataPinOUT, clockPinOUT, MSBFIRST,  light_configs[arrIndex][1]); // REG
2.
        shiftOut(dataPinOUT, clockPinOUT, MSBFIRST,  light_configs[arrIndex][0]); // REG
1.
        digitalWrite(latchPinOUT, HIGH);

}

/*
 * Map Input Sensor ID to Configuration ID.
 */
void mapInputToQueue(int i)
{
        switch (i) {
                case  1:
                case  5: addQ(1); break;
                case  0:
                case  3:
                case  7: addQ(2); break;
                case  2:
                case 12:
                case 13: addQ(3); break;
                case  4:
                case  8:
                case  9:
                case 14:
                case 15: addQ(4); break;
                case  6: addQ(5); break;
```

```
                case 10:
                case 11: addQ(6); break;
        }
}

void addQ(int val)
{
        Serial.print("Q: ");
        Serial.println(val);
        for(int i = 0; i < qSize; i++) {

                // Exit if already in Q exists
                if(queue[i] == val) {
                        break;
                }

                // ADD
                if(queue[i] == 0) {
                        queue[i] = val;
                        break;
                }

        }

}

int popQ()
{
        int val = queue[0];

        for(int i = 0; i < qSize; i++) {

                if(i == qSize - 1) {
                        queue[qSize - 1] = 0;
                } else {
                        queue[i] = queue[i + 1];
                }

        }

        return val;
}

void printQ()
{
        Serial.print("Current Queue [");
        for(int i = 0; i < qSize; i++) {
                Serial.print(queue[i]);
        }
        Serial.println("}");
}



void setup()
{
        Serial.begin(9600);

        // Initialize pins...
        pinMode(latchPinOUT, OUTPUT);
        pinMode(clockPinOUT, OUTPUT);
        pinMode(dataPinOUT , OUTPUT);

        pinMode(ploadPin, OUTPUT);
        pinMode(clockEnablePin, OUTPUT);
        pinMode(clockPin, OUTPUT);
```

```
        pinMode(dataPin, INPUT);

        digitalWrite(clockPin, LOW);
        digitalWrite(ploadPin, HIGH);

        // Read Shift Register and Display Config 0 (default).
        pinValues    = read_shift_regs();
        oldPinValues = pinValues;
        display_led(0);
}

/*
 * MAIN
 */
void loop()
{
        // Timer Interrupt - Display 1. config from Queue
        if(millis() > targetTime) {
                display_led(popQ());
                targetTime = millis() + 10000;
                printQ();
        }

        pinValues = read_shift_regs();

        if(pinValues != oldPinValues) {
                eval_pin_values();
                oldPinValues = pinValues;
        }

        delay(500);
}
```