CARMINA

Zwischenstand 18.03.2020

von Doreen St. Pierre, Projekt im Kurs „Creative Coding“ bei Julian Hespenheide im Wintersemester 2019/20 an der Hochschule Magdeburg-Stendal

Programmcode

Klasse Carmen

Die Klasse „Carmen“ bildet die Hauptklasse des Programms und dient lediglich der Instanziierung der anderen Klassen. Momentan wird innerhalb Carmens noch mit einer while-Schleife gearbeitet, um die Anzahl der gefundenen Wörter auf maximal 10 zu beschränken. Dies dient lediglich Testzwecken und wird im endgültigen Programmcode nicht vorhanden sein.

Klasse Randomizer

In dieser Klasse findet eine zufällige Auswahl von Buchstaben statt. Bei der Instanziierung dieser Klasse kann durch die Mitgabe des Parameters *length* bestimmt werden, wie viele Buchstaben immer gleichzeitig ausgewählt werden sollen. Die zurällige Auswahl von Buchstaben erfolgt über random.sample.
Bei der Initialisierung der Klasse wird das Array *alphabet* angelegt, in dem sich alle Buchstaben des Alphabets befinden.
Die Klasse enthält eine Funktion *create_random_string*. In ihr werden beliebige Buchstaben in dem Array *random_characters* gespeichert. Die Buchstaben werden zufällig aus einer Liste durch *random.sample* ausgewählt. Als Liste dient hierbei das zuvor definierte Array *alphabet* mit allen Buchstaben des Alphabets. Die durch diese Funktion zufällig ausgewählten Buchstaben werden durch den return Wert *random_characters* zurückgegeben.

Klasse WordProcessor

In dieser Klasse findet der Abgleich zwischen den zufällig ausgewählten Buchstaben und einem Wörterbuch statt, um herauszufinden, ob ein Wort aus den Buchstaben generiert werden kann. Die Klasse beinhaltet 4 Funktionen. In der ersten Funktion *create_n_character_word_list* werden zunächst alle Wörter des Wörterbuchs im *Array word_list* gespeichert. Eine anschließende For-Schleife durchläuft dieses Array. Sie filtert diejenigen Wörter heraus, die der beim Aufruf der Funktion festgelegten Anzahl an Buchstaben entspricht und speichert diese im Array *n_character_word_list*. Die zweite Funktion *create_word_index* verwandelt Wörter in einen Index. Dies geschieht, indem die einzelnen Buchstaben dieses Wortes alphabetisch sortiert werden. Die alphabetisch sortierten Buchstaben bilden den Index. Dieser wird über einen return Wert ausgegeben. In der Funktion *sort_words_into_index* wird für jedes Wort aus dem Array *n_character_word_list* dieser Index gebildet. Die letzte Funktion *find_words_from_characters* kreiert den gleichen Index für die zufällig ausgewählten Buchstaben. Der Index dient dazu, auf das Dictionary zuzugreifen, in dem alle Worte des Wörterbuchs sortiert nach Index gespeichert sind. Die dadurch gefundenen möglichen Worte werden über den return Wert *possible_words* ausgegeben.

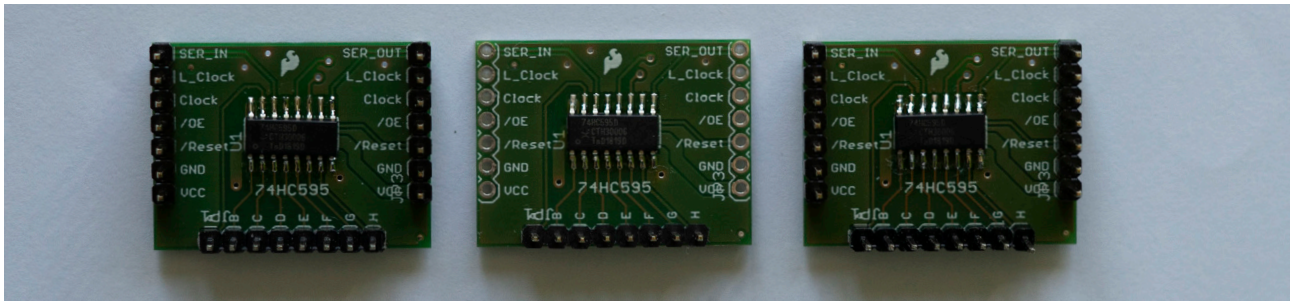
Klasse Ascii

Die Klasse Ascii ermöglicht es, aus einem beliebigen Text einen Text in Ascii-Art zu machen. Dazu wird die Bibliothek pyfiglet eingebunden, die verschiedene Ascii-Art Schriften enthält. In der Funktion *create_ascii_text* wird die Schriftart festgelegt, die verwendet werden soll.

Hardware

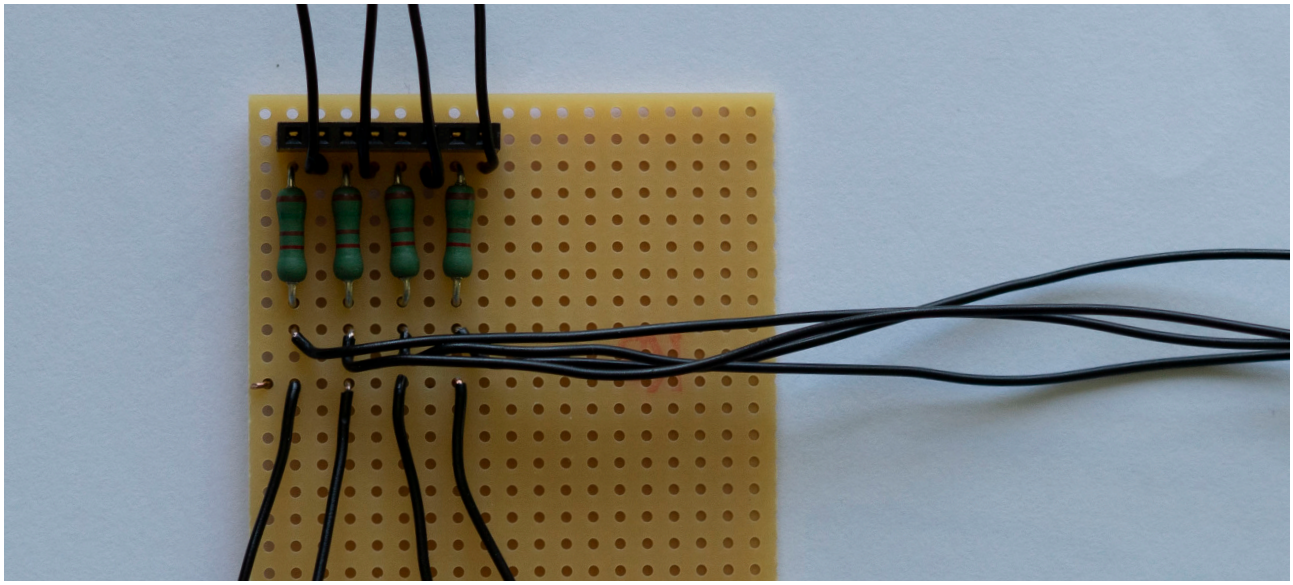
Schieberegister

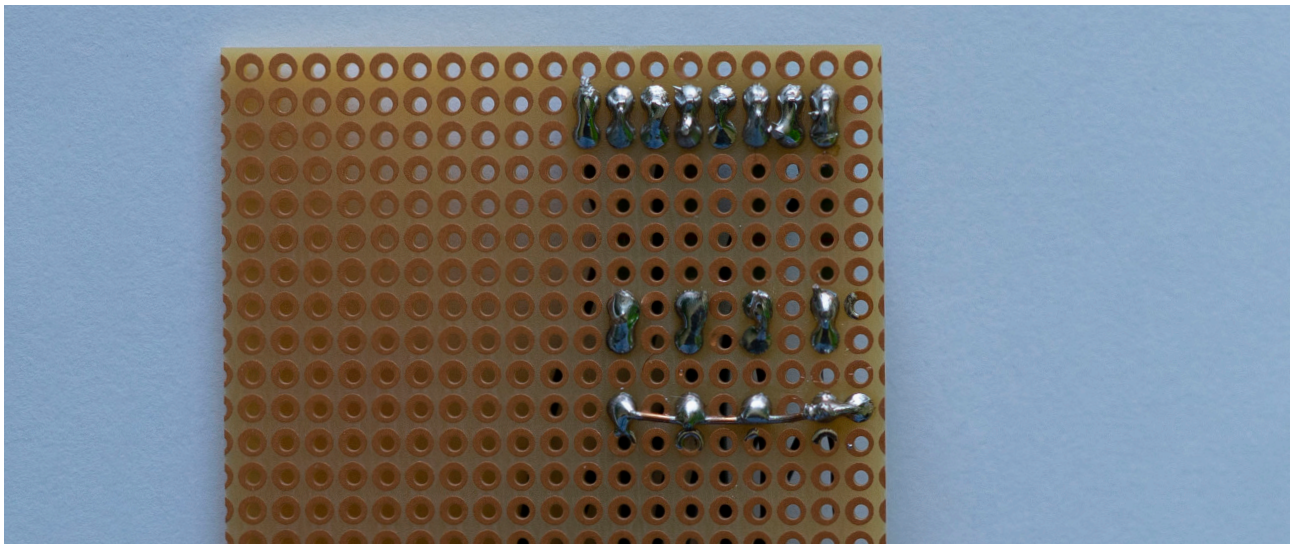
Es werden 15 Schieberegister benötigt, um die 60 Buchstaben und die damit verbundenen 120 LEDs ansteuern zu können. Um die Kaskade aus 15 Schieberegistern besser verbinden zu können, wurde eine Platine bestellt. Die HC595 wurden auf diese gelötet. Außerdem wurden Stiftleisten angebracht, um die einzelnen Platinen und die LEDs besser miteinander verbinden zu können. Alle Platinen erhielten einen Stiftleiste für die Verbindung mit den LEDs. Jede zweite Platine erhielt zusätzlich dazu zwei Stiftleisten, um die Platinen untereinander verbinden zu können.



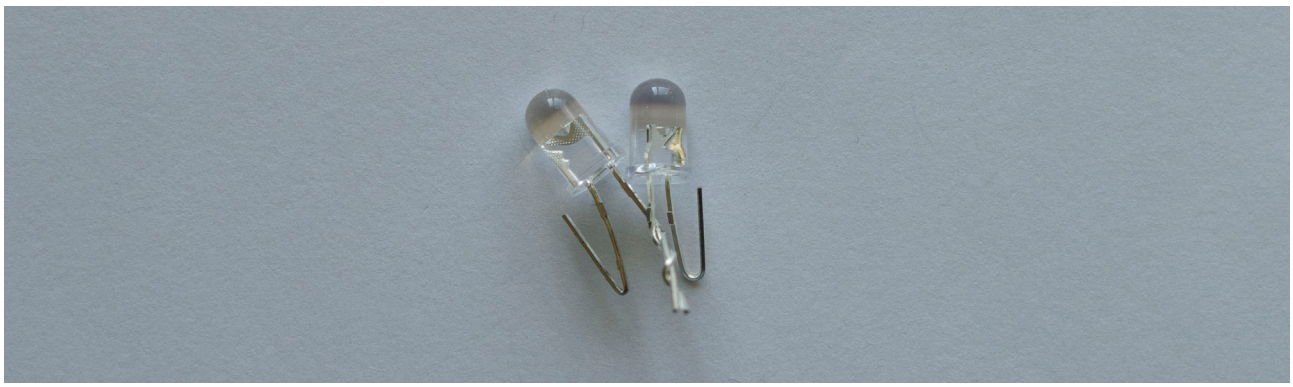
LEDs

Beim Vergleich der Leuchtkraft zwischen den blauen und den weißen LEDs stellte sich heraus, dass die weißen LEDs wesentlich heller leuchten. Um das auszugleichen, wurde der Schaltplan angepasst: Durch Einsatz eines 150 Ohm Widerstands vor der weißen LED wurde die Leuchtkraft ausgeglichen. Für jedes HC595 Bauteil wurde eine Platine für die Ansteuerung der LEDs angefertigt. Diese besteht aus einer Buchsenleiste, die den Kontakt zwischen der Platine des Schieberegisters und der Platine der LEDs herstellt. Neben der Buchsenleiste besteht jede dieser Platinen aus 4 Widerständen und 12 Leitungen für die LEDs. Die Leitungen, die direkt an den Buchsenleisten angebracht sind, steuern die blauen LEDs an, während bei den Leitungen für die weißen LEDs ein Widerstand zwischengeschaltet ist. Vier Kabel wurden durch ein Stück Draht miteinander verlötet, sie bilden die Masseanschlüsse für die LEDs.



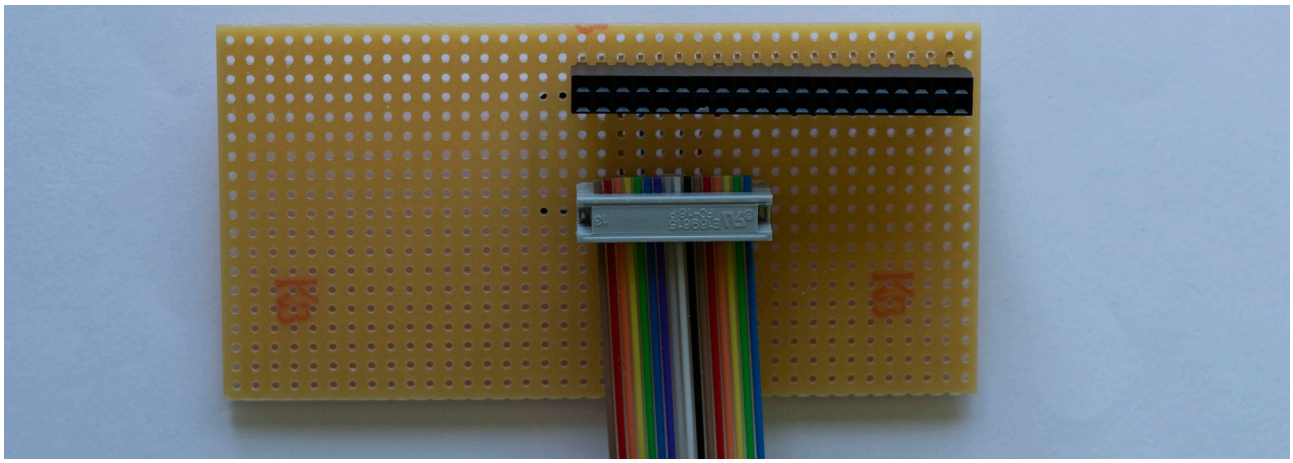


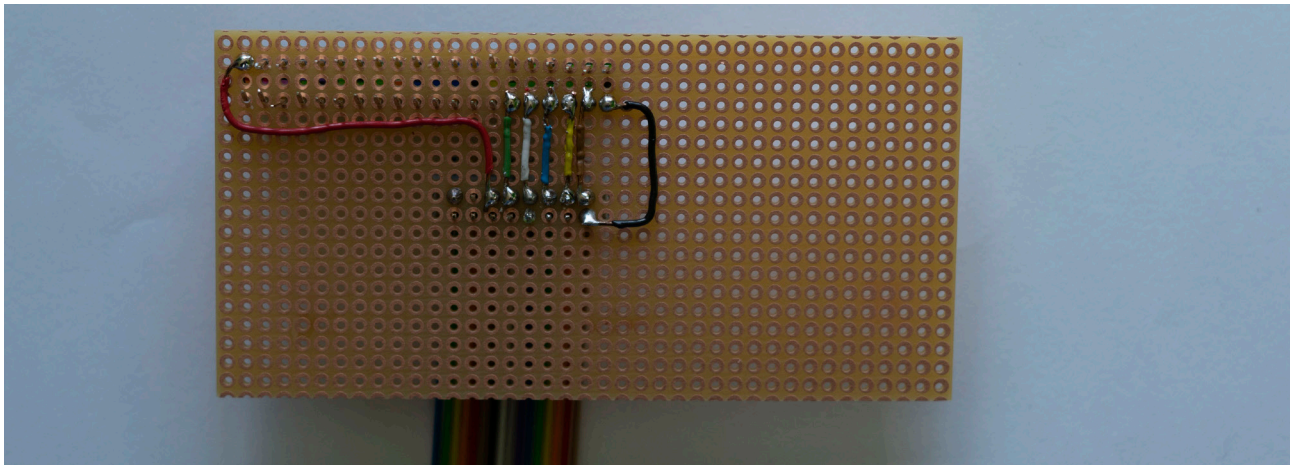
An diese Platine wurden die LEDs angeschlossen. Dazu wurde immer die Kathode einer weißen und einer blauen LED miteinander verbunden und an eine gemeinsame Masseleitung angelötet. Die Anoden erhielten jeweils unterschiedliche Leitungen.



Rasberry Pi

Für die Anschlüsse des Rasberry Pi wurde ebenfalls eine Buchsenleiste auf eine Platine gelötet. Die Verbindung zwischen den Anschlüssen des Rasberry Pi und dem ersten Schieberegister in der Kaskade erfolgt über einen Flachbandstecker. Dafür wurden die Leitungen des Flachbandsteckers mit der Buchsenleiste verlötet.





Die nachstehende Tabelle zeigt die Belegung der einzelnen Pins und Kabel.

Rasberry Pi Pin	Farbe Leitung	Farbe Verdrahtung	Funktion
GND	braun	schwarz	GND
GPIO 26	rot	braun	Reset
GPIO 19	gelb	gelb	OE
GPIO 13	blau	blau	Clock
GPIO 6	grau	weiß	L_Clock
GPIO 5	schwarz	grün	SER_IN
5 V	rot	rot	VCC

Nächste Schritte

Im nächsten Schritt wird der Teil der Software geschrieben, der die Schieberegister ansteuert. Bisher wurden 5 Platinen für die Ansteuerung der LEDs gelötet, die verbleibenden 10 müssen noch gelötet werden. Außerdem muss noch das Programm für die Ansteuerung des Druckers geschrieben und der Drucker an den Rasberry Pi angeschlossen werden.

Aktualisierter Schaltplan