Projekt für Embedded Systems

Schlosssteuerung mit LED und Keypad

Lukas Brüggemann, Markus Reinhold, Sebastian Pötter

Jahrgang: 20INM

im Studiengang
Informatik Master

Embedded Systems (7320)
Prof. Dr.-Ing. Andreas Pretschner

25. März 2021

Inhaltsverzeichnis

Abkürzungsverzeichnis

GPIO General Purpose Input/Output

IDE Integrated Development Environment

LED Light Emitting Diode

MCU Microcontroller Unit

PIN Personal Identification Number

PUK Personal Unblocking Key

RGB Red, Green, Blue

USB Universal Serial Bus

Glossar

Algorithmus : Feste Vorgehensweise um ein spezifisches Problem zu lösen
AT-Mega : Ein Ein-Chip-Mikrocontroller der Firma Atmel aus der megaAVR-Familie.
AVRDUDE : Programmiersoftware für Atmel AVR Controller.
GCC-AVR : AVR-GCC ist ein freier C-Cross-Compiler für AVR-Mikrocontroller [2].

1 Aufgabenstellung

Das Ziel des Projektes ist die Realisierung eines Nummernschlosses mit LED Visualisierung, welches die folgenden Funktionen erfüllen muss:

- Schlüsseleingabe mit PIN bestehend aus den Zeichen 0 bis 9 und A bis D
- Visualisierung erfolgreicher Schlüsseleingabe durch eine LED
- Nach erfolgreicher Schlüsseleingabe kann ein neuer PIN festgelegt werden
- Nach einer festgelegten Anzahl von Misserfolgen während der PIN-Eingabe soll das Schloss gesperrt werden
- Wenn das Schloss gesperrt ist, kann es mit einem PUK wieder entsperrt werden

Dieses Projekt soll mit einem 'Arduino UNO'-Mikrocontroller realisiert werden, ohne zusätzliche Einbindung anderer Bibliotheken mit Ausnahme der Standardbibliothek. Die Implementierung soll mit der Programmiersprache C unter Verwendung der IDE Eclipse vollzogen werden. Das Projekt soll zudem modular aufgebaut werden, sowie vollständig dokumentiert sein. Dies beinhaltet die vollständige Dokumentation der für den Betrieb der Sensoren/Aktoren relevanten Funktionen im Quellcode (Doxygen-Dokumentation).

2 Hardware und Konzept

Für die Aufgabe werden die folgenden Bauteile benötigt, wobei für das Tastenfeld eine 4x4 Matrix aus Tastern benutzt werden kann, welche ähnlich zu dem Schaltplan sind:

- Arduino UNO (im weiteren nur noch Arduino genannt)
- RGB LED
- 3x 470-Ohm Widerstände
- 4x4 Tastenfeld
- Jumper-Kabel zum Verbinden
- USB Kabel für Debug und Datenaustausch

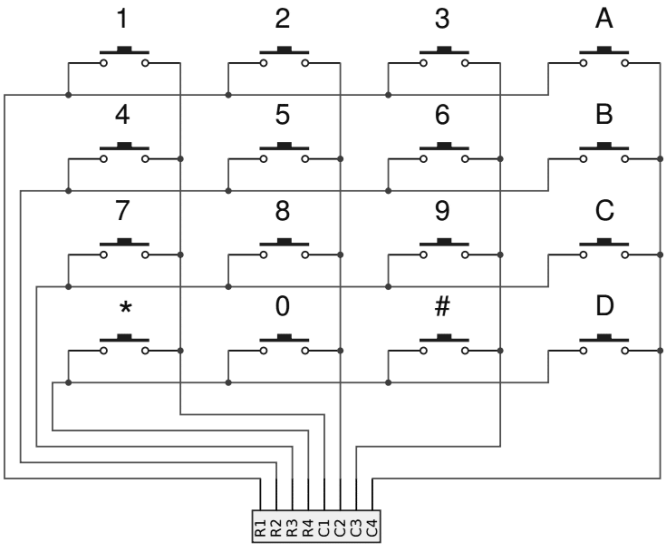
Der Arduino ist ein Mikrocontroller auf Basis des ATmega328P. Er verfügt über 14 digitale Ein-/Ausgangspins, 6 analoge Eingänge, einen 16-MHz-Keramikresonator, einen USB-Anschluss, eine Netzbuchse, einen ICSP-Header und einen Reset-Knopf [1].

Die RGB LED enspricht einer kleinen 5mm LED, welche rotes, blaues und grünes Licht abstrahlen kann. Diese LED wird zur Signalisierung der Zustände des Nummernschlosses benötigt. Die Vorwiderstände von 470 Ohm werden zum Schutz der Dioden verwendet, da der Arduino eine Ausgangsspannung von 5V der GPIO-Pins hat.

2.1 Funktionsweise des Tastenfeldes

Das 4x4 Tastenfeld ist ein 16/8 Multiplexer. Es werden anstelle von 16 Pins, 8 Pins auf dem Aruduino benötigt. Dabei sind 4 Pins Ausgänge (R1-R4) und 4 Pins Eingänge (C1-C4). Auf Abbildung 1 ist der Schaltplan des Tastenfeldes abgebildet. Um eine gedrückte Taste zu ermitteln werden alle Ausgänge hintereinander auf High gesetzt währenddessen geprüft wird ob eine der Eingänge auf High ist. Sind Eingang und Ausgang auf High, ist somit automatisch die gedrückte Taste bekannt.

Abbildung 1: 4x4 Tastenfeld - Schaltplan



2.2 Pinbelegung des Arudinos

Das 4x4 Tastenfeld und die RGB LED werden mit dem Arduino über folgende GPIO Pins verbunden:

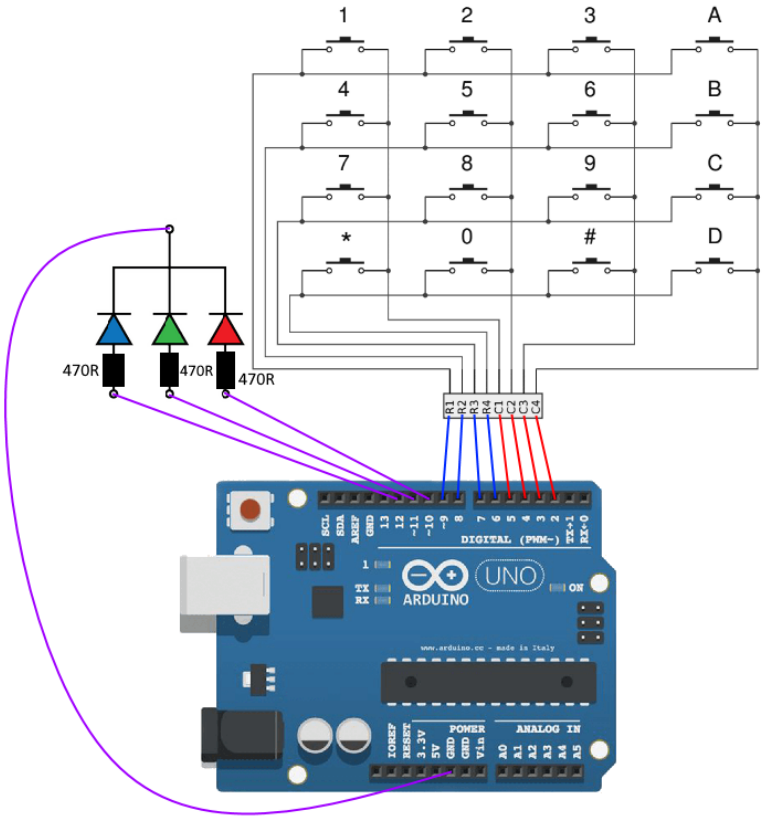
```
const int Output[rows]={2,3,4,5};
const int Input[columns]={6,7,8,9};
const int redPin = 10;
const int greedPin = 11;
const int bluePin = 12;
```

Dabei werden GPIO Pin 2 bis 5 als digitaler Output festgelegt. Diese Pins entsprechen R1 bis R4 des Tastenfelds. GPIO 6 bis 9 sind als digitaler Input eingestellt und entsprechen C1 bis C4. GPIO Pin 10 bis 12 sind digitale Ausgänge um die Intensität der jeweiligen Farben einzustellen.

2.3 Versuchsaufbau

Abbildung 2 ist der Sketch des Zahlenschlosses. Hierbei sind der Arduino-Uno, die RGB-LED, die Vorwiderstände und das Tastenfeld korrekt verbunden. Dabei stimmen die Pin-Belegungen mit dem aus dem Quellcode verwendeten Pins überein.

Abbildung 2: Sketch des Zahlenschlosses

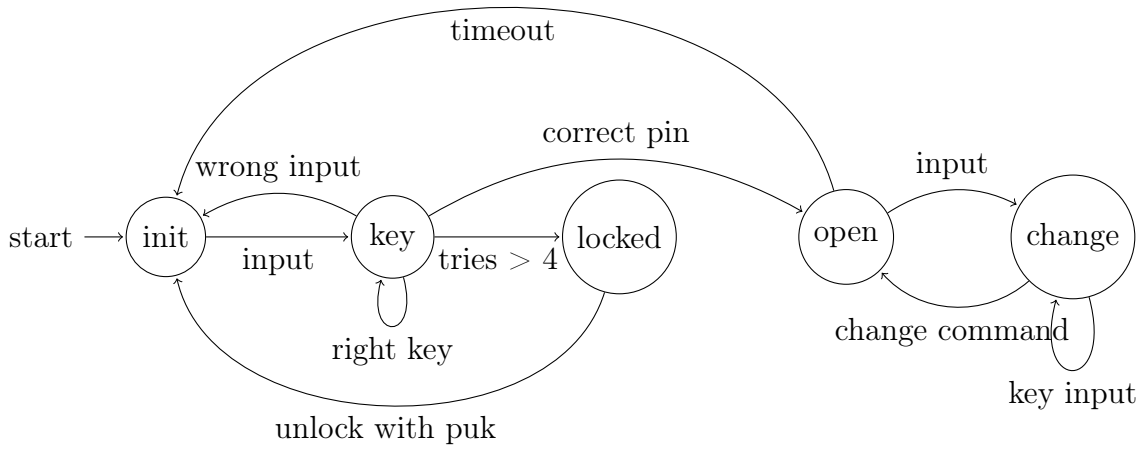


2.4 Zustandsmaschine

Das Verhalten des Nummernschlosses wird als eine Zustandsmaschine beschrieben, welche in Abbildung 3 zu sehen ist. Die PIN und der PUK werden im Quellcode fest definiert. Der Zustand 'init' ist der Startzustand der Maschine. Sobald eine Taste gedrückt wird, wechselt der Zustand in 'key'. Hierbei werden die Eingaben mitgezählt, bei der korrekten Eingabe einer Stelle wird der Zähler erhöht ('right key'). Ist die Eingabe falsch, wird zurück in den 'init' Zustand gewechselt und die Eingabe zurückgesetzt. Nach 4 Fehleingaben wird das Schloss gesperrt und von 'key' in den Zustand 'locked' gewechselt. Dieser Zustand kann nur verlassen werden, wenn die PUK korrekt eingegeben wird. Ist die PIN korrekt, wird in den Zustand 'open' gewechselt und das Schloss ist geöffnet. Hierbei kann der PIN geändert werden oder das Schloss sperrt sich nach einer gewissen Zeitspanne automatisch und wechselt in den Zustand 'init'. Falls eine PIN-Änderung gewünscht wird, muss eine bestimmte

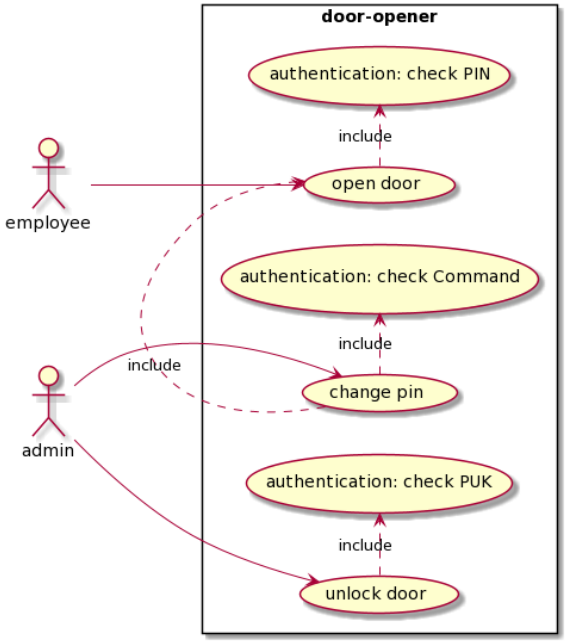
Zeichenfolge eingegeben werden. Nach der Aufzeichnung der Eingaben wird der neue PIN gespeichert und es wird zurück in den 'open' Zustand gewechselt.

Abbildung 3: Zustandsmaschine der Schlosssteuerun



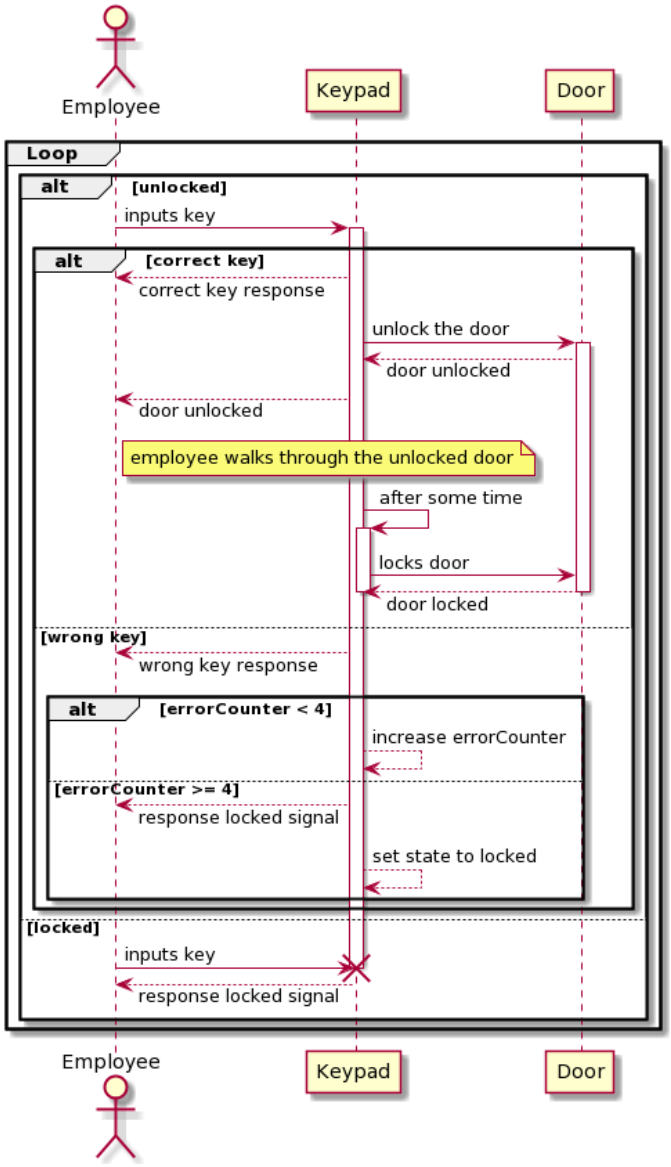
Die Abbildung 4 ist ein Anwendungsfalldiagramm für die Schlosssteuerung. Hierbei gibt es zwei Gruppen von Anwendern. Die erste Gruppe sind die nutzenden Personen ('Employee'), welche die Tür entsperren wollen um durchzugehen. Diese benötigen hierfür einen PIN. Die zweite Gruppe sind die Administratoren, welche sich um die Verwaltung kümmern. Die Administratoren können gesperrte Schlösser mit der Hilfe eines PUK entsperren. Zusätzlich können sie den PIN ändern, wenn das Schloss offen ist und sie einen Befehl auf dem Tastenfeld eingeben.

Abbildung 4: Use-Case Diagramm



In Abbildung 5 wird eine Sequenz von nutzenden Personen ('Employee'), Türschloss ('Keypad') und Tür ('Door') gezeigt. Zum besseren Verständnis wurde die Tür als Objekt noch zusätzlich eingefügt, auch wenn diese nicht im Aufbau erklärt wird. Möchte die nutzende Person die Tür öffnen gibt es zwei Fälle. Wenn das Schloss nicht gesperrt ist, kann die nutzende Person einen PIN eingeben um die Tür zu entsperren. Dabei werden die Antworten des Türschlosses als Farben auf der LED visualisiert (siehe Kapitel 4). Ist die Tür gesperrt, kann die nutzende Person zwar das Tasten klicken aber das Schloss wird nicht reagieren.

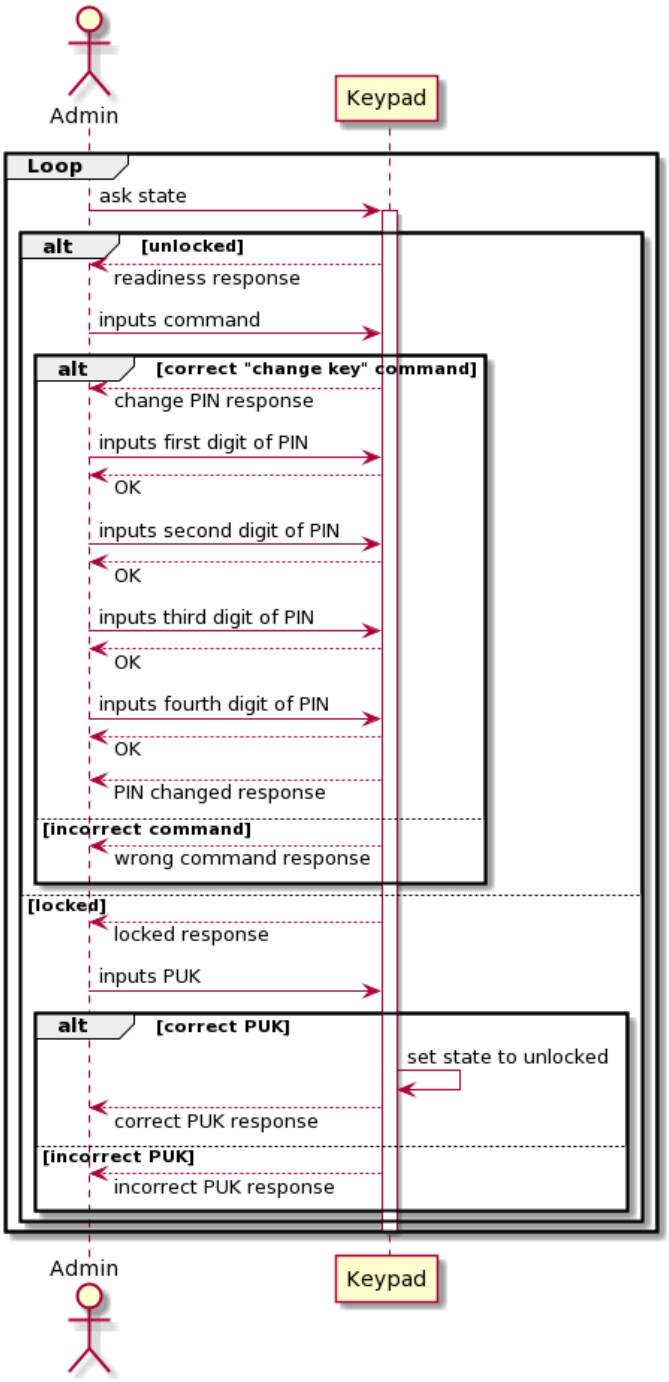
Abbildung 5: Sequenzdiagramm - nutzende Person



Nun wird in der Abbildung 6 die Sequenz von administrierenden Personen ('Admin') und dem Türschloss beschrieben. Dabei wird auf die Möglichkeit eingegangen das Schloss zu entsperren, falls der PIN zu oft falsch eingegeben wurde. Zusätzlich wird auch die Möglichkeit der Änderung des PIN beschrieben. In diesem Fall muss das Türschloss bereits über den PIN geöffnet worden sein. Auch hier werden die

Antworten des Türschlosses mit Hilfe der LED visualisiert (siehe Kapitel 4).

Abbildung 6: Sequenzdiagramm - Administrator



3 Einrichtung der Entwicklungsumgebung

Um das Programm auf den Mikrocontroller zu übertragen müssen folgende Bibliotheken installiert werden:

- binutils : um Werkzeuge wie Assembler, Linker zu bekommen
- gcc-avr : ein GNU C Cross-Compiler für speziell für AVR
- avr-libc: es ist ein Paket für AVR C Bibliotheken
- AVRDUDE : Anwendung zum programmieren des Mikrocontrollers

Darüber hinaus wird ein Makefile benutzt um den Programmcode zu kompilieren und den Mikrocontroller zu programmieren. Für den AVR-GCC Compiler werden noch spezielle Bibliotheken benötigt, diese können mit folgenden Befehlen installiert werden (Linux-Ubuntu) [3]:

```
sudo apt-get install gcc-avr binutils-avr avr-libc
sudo apt-get install avrdude
```

Die AVR-GCC Toolchain kennt das Layout des Arduino Uno nicht, weswegen der konkrete Mikrocontroller richtig angesprochen werden muss. Im Speziellen ist auf dem Arduino Uno der Atmega328p verbaut. Dieser muss dann im Makefile angegeben werden, sodass der Programmcode für den Mikrocontroller passend kompiliert wird.

Als Makefile wird eine abgeänderte Version des von Prof. Pretschner bereitgestellte Makefiles verwendet. So muss die MCU Angabe auf 'atmega328p' geändert, und die Angabe des Programmer für den AVRDUDE muss zu 'arduino' verändert werden. Abhängig davon, wie der Controller angeschlossen ist, muss zudem der Port in den AVRDUDE Einstellungen angepasst werden z. B. '/dev/ttyUSB0'.

4 Implementierung und Umsetzung

Der Programmcode ist modular aufgebaut. Sowohl die Zustandsmaschiene ('states.c') also auch das Tastenfeld ('keypad.c') und die LED-Steuerung ('led.c') sind in separaten Dateien getrennt. Im folgenden werden die Module kurz erklärt.

Jedes C-Programm, enthält die Definition einer Funktion namens 'main', die den designierten Start des Programms bzw den Programmeinstieg darstellt. Die 'main' Funktion wird, wie in diesem Fall, meist in einer Datei namens 'main.c' deklariert. In dieser Funktion werden die Pins jeweils für das Keypad und die LED eingestellt (Zeilen 8-9). Nach der Einstellung und der Initialisierung folgt eine Endlos-Schleife,

welche die Zustandsmaschine ausführt (Zeilen 13-19). Hierbei wird eine kleine Verzögerung eingestellt um den Tastendruck optimal zu erfassen. Hierfür wird derr millistimer benutzt welcher vor der schleife initialisiert wird und mit sei(); w

```
1  int main(void)
2  {
3      setup_LED();
4      setup_keypad();
5    init_millis(16000000UL); //frequency the atmega328p is running at
6    sei();
7
8      while(1) //Infinite loop
9      {
10         if(millis()-kdelay>period) //used to make non-blocking delay
11         {
12             kdelay=millis();  //capture time from millis function
13             stateM();
14         }
15     }
16     return 0;
17 }
```

Die Zustandsmaschine wird in einem 'Switch-Case'-Block realisiert. Dieser wird in der Funktion 'stateM' ausgeführt. Und verändert anhand der Abbildung 3 den Zustand.

```
void stateM()
{ ... }
```

Dabei haben die Zustände zur Visualisierung verschiedene Farben, welche per LED visualisiert werden. Hierbei wird die Funktion 'setLED(r, g, b)' benutzt, welche die jeweiligen Farben ansteuert, die im folgenden aufgeschlüsselt werden:

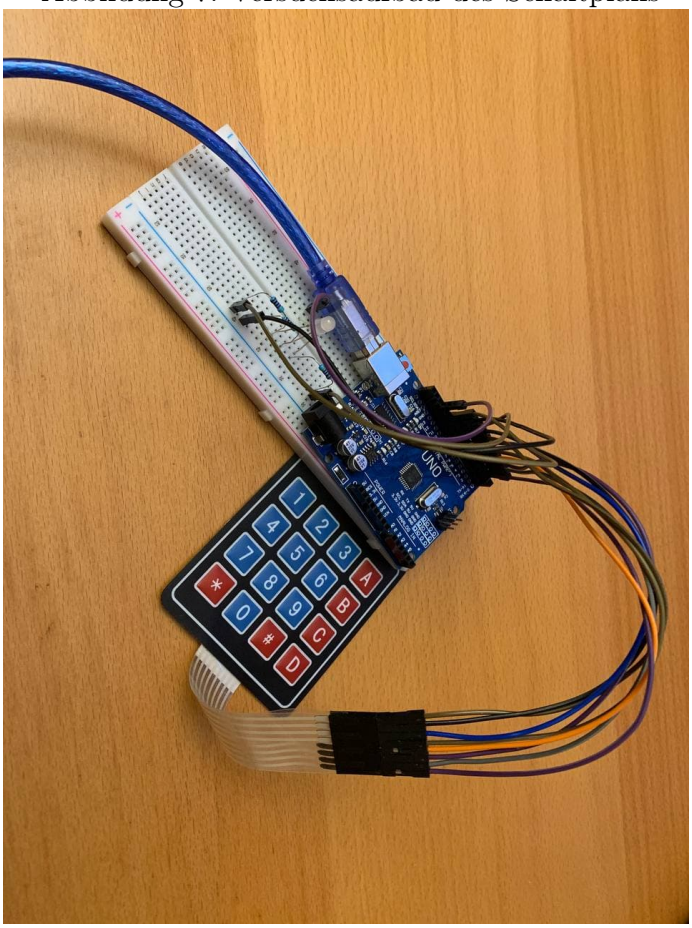
- LOCKED: Rot
- NEWKEY: Gelb
- OPEN: Magenta
- KEY: Weiß
- INIT: Blau
- Gedrückte Taste: kurzes Grün

Innerhalb dieser Methode wird die Funktion 'keypad' aufgerufen, welche für das Erfassen der Tastendrücke zuständig ist. Dabei wird zuerst geprüft ob bereits eine Taste gedrückt wird, ist dies nicht der Fall wird auf einen Tastendruck geprüft. Wenn eine Spalte LOW ist wird auf die Zeile geprüft ob diese High ist. Wenn Zeile und Spalte identifiziert sind, konnte die Taste identifiziert werden und wird zurückgegeben.

```
char keypad()
{ ... }
```

Abbildung 7 zeigt den konkreten Versuchsaufbau des oben vorgestellten Schaltplans von Abbildung 2. Die LED mit den Vorwiderständen wurden auf ein sog. 'Breadboard' gesteckt.

Abbildung 7: Versuchsaufbau des Schaltplans



5 Bedienungsanleitung

Sobald das Gerät mit Strom versorgt wird, kann die Schlosssteuerung benutzt werden. Hierfür visualisiert das grüne Leuchten der LED, dass die Schlosssteuerung korrekt gestartet wurde und auf einen Tastendruck wartet. Die Initial-PIN für das Entsperren des Schlosses lautet 'a123'. Dieser kann nach korrekter Eingabe geändert werden. Wenn eine Taste gedrückt wird, leuchtet die LED kurz Gelb auf um den Tastendruck zu bestätigen. Bei Misserfolg wird das Schloss zurückgesetzt und die vollständige PIN muss nochmals eingegeben werden. Wurde die PIN korrekt eingegeben wird das Schloss freigegeben und die LED leuchtet dauerhaft grün. Hier gibt es nun die Möglichkeit, die PIN zu ändern. Mit der Tastenkombination '#a' im freigegebenen Schloss kann das Ändern der PIN initiiert werden. Nun leuchtet die LED Türkis und eine neue 4-stellige PIN kann eingegeben werden. Nach der Eingabe der neuen PIN wechselt das Schloss wieder in den geöffneten Modus. Nach einem kurzen Zeitintervall ohne Benutzereingabe wird das Schloss wieder geschlossen und die PIN muss erneut eingegeben werden. Sollte der PIN dreimal falsch eingegeben worden sein, wird das Schloss gesperrt und muss mit einem PUK entsperrt werden. Der PUK für das Entsperren lautet '#12345'. Dieser kann beliebig oft eingegeben werden und erst bei erfolgreicher Eingabe wird das Schloss entsperrt.

6 Fazit

Eine Zustandsmaschine ist für ein kleines System eine passende Vorgehensweise. Für komplexere Systeme sollte eine andere Art von der Beschreibung des Systems gewählt werden, da für jeden Zustand ein neuer Wert definiert werden muss und dieser in das 'Switch-Case Statement' eingefügt werden muss. Dies führt bei komplexeren Systemen zu einer Verschlechterung der Code-Qualität und einem unleserlicheren Quellcode. Der Aufgabenstellung konnte entsprochen werden. So wurde ein Nummernschloss entworfen, welches entsprechend der Zustandsmaschine arbeitet und diese Zustände angemessen visualisiert. Auch wurde das Makefile so verändert, dass diese für den benutzen Mikrokontroller (Arduino Uno - ATMega328p) benutzt werden konnte.

Die Verwendung der Open-Source-Software 'Arduino-IDE' [1] wäre aufgrund von weitreichender Unterstützung von Arduino-Mikrocontrollern eine bessere Entscheidung gewesen als die 'Eclipse'-IDE. So gab es am Anfang Probleme bei der Einrichtung der Entwicklungsumgebung, sowie Unstimmigkeiten bei einigen Funktionen. Dennoch konnte die IDE 'Eclipse' für dieses Projekt benutzt werden um den Quellco-

[1] https://github.com/arduino/Arduino

de zu erarbeiten und ein Makefile zu erstellen, sowie zum Hochladen der Anwendung auf den Mikrocontroller.

Literaturverzeichnis

Literatur

[1] *Produktseite vom Arduino Uno Rev3*. 2020. URL: https://store.arduino.cc/arduino-uno-rev3 (besucht am 06. 11. 2020).

[2] *AVR-GCC*. URL: https://www.mikrocontroller.net/articles/AVR-GCC (besucht am 01. 01. 2021).

[3] *Setup AVR-GCC*. URL: https://create.arduino.cc/projecthub/milanistef/introduction-to-bare-metal-programming-in-arduino-uno-f3e2b4 (besucht am 24. 12. 2020).

Abbildungsverzeichnis