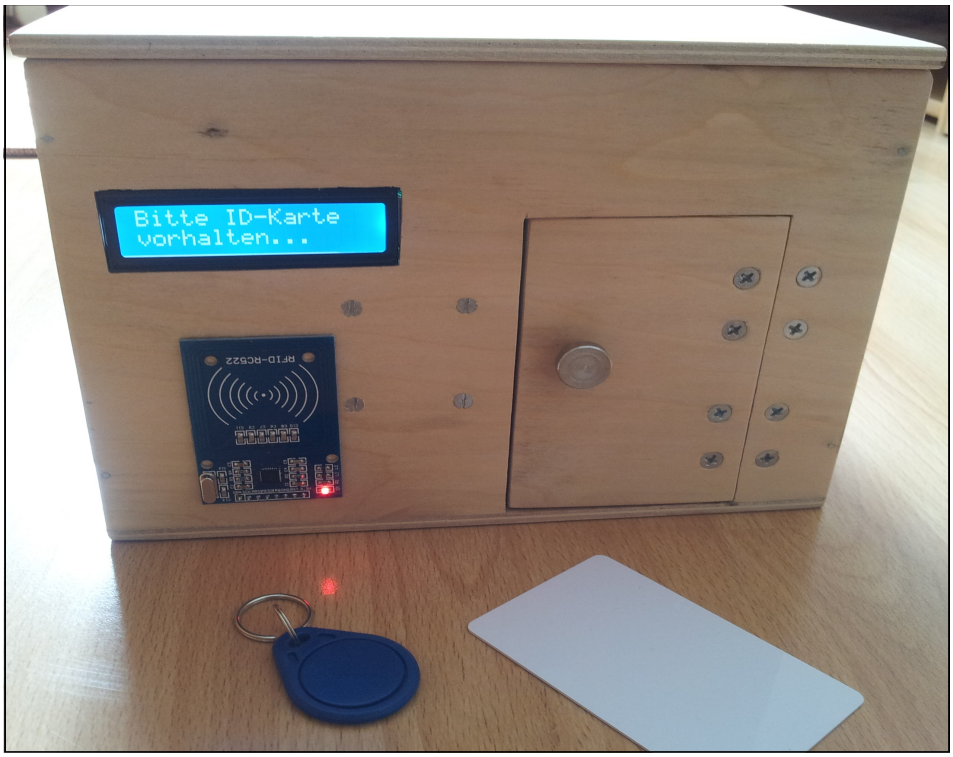
Hochschule Ostwestfalen-Lippe
University of Applied Sciences

FACHBEREICH ELEKTROTECHNIK UND TECHNISCHE INFORMATIK

Aufbau, Programmierung und Realisierung einer RFID-Zugangskontrolle mit einem Arduino Mega 2560



Name: Cihan Aydogdu
Matrikelnummer: xxxxxxxx
Modul: Projektarbeit
Prüfer: Prof. Dr. rer. nat. Ernst Beckmann
Abgabedatum: 10.06.2016

Inhaltsverzeichnis

1.0 Einleitung

1.1 RFID

Radio Frequency Identification – oder kurz: RFID – ist eine Technologie, um Daten berührungslos und ohne Sichtkontakt lesen und speichern zu können. Die RFID-Technologie ist heutzutage kaum wegzudenken, da sie in vielen Bereichen, wie z.B. in der Logistik, im Handel, in Produktionsbetrieben Anwendung findet. Aufgabe und Ziel der automatischen Identifikation ist die Bereitstellung von Informationen zu Personen, Tieren, Gütern und Waren.
Da RFID diverse Vorteile, wie z.B. Sicherheit, kontaktloses Lesen und Schreiben, große Reichweite, etc., mit sich bringt, findet sie immer mehr an Verbreitung.

1.2 Arduino

Arduino (seit März 2015 auch Genuino) ist eine aus Soft- und Hardware bestehende Physical-Computing-Plattform. Beide Komponenten sind im Sinne von Open Source quelloffen. Die Hardware besteht aus einem einfachen E/A-Board mit einem Mikrocontroller und analogen und digitalen Ein- und Ausgängen. Das Arduino Board gibt es in verschiedenen Varianten wie Arduino Uno, Arduino Mega, Arduino Zero oder Arduino Yún. Entsprechend lässt sich das Arduino Board vielfältig einsetzen um spielerische, wissenschaftliche oder künstlerische Projekte zu realisieren [1].

Die Entwicklungsumgebung basiert auf Processing und soll auch technisch weniger Versierten den Zugang zur Programmierung und zu Mikrocontrollern erleichtern. Die Programmierung erfolgt in C bzw. C++. Umfangreiche Libraries und Beispiele vereinfachen die Programmierung [2].

1.2.1 Funduino

Das Board, das in diesem Projekt eingesetzt wird, ist nicht von Arduino, sondern von Funduino. Laut Hersteller ist das Funduino Mega 2560 Board baugleich mit dem Arduino Mega 2560. Es gibt z.Z. ca. 100 Arduino Nachbauten, die sowohl hardware- als auch softwaretechnisch kompatibel zu Arduino sind [3]. Diese nachgebauten Boards können mit der Arduino IDE problemlos programmiert werden. Funduino ist auch einer von vielen Nachbauten, welches in diesem Projekt Anwendung findet.
In diesem Dokument wird nicht mehr auf das Funduino Board eingegangen, stattdessen wird fortlaufend Arduino genannt, weil das Arduino allgemein gültig ist und das Projekt mit der Arduino IDE erstellt wurde.

1.3 Zugangskontrolle mit Arduino

Arduino und auch dessen Nachbauten bieten diverses Zubehör an, wie z.B. Sensoren, LEDs, Widerstände, Aktoren, etc. Es gibt viele Möglichkeiten, um mit einem Arduino verschiedene Projekte zu realisieren. Da Arduino viele Vorteile mit sich bringt, wie einfache Programmierung, fertige Libraries und Erweiterungsmöglichkeiten, eignet sich Arduino bestens für dieses Projekt. Mittels des RFID-Moduls, welches mit einem Arduino Kit erworben werden kann und einer Elektromagnetverriegelung kann man eine Zugangskontrolle realisieren.

2.0 Beschreibung des Arduino Mega 2560 Boards

2.1 Technische Details

Mikrocontroller	ATmega2560
Betriebsspannung	5 V
Eingangsspanung (empfohlen)	7-12 V
Eingangsspannung (Limit)	6-20 V
Digitale I/O Pins	54 (15 davon sind PWM Ausgänge)
Analoge Input Pins	16
Stromstärke (DC) per I/O Pin	20 mA
Stromstärke (DC) für 3,3 V Pin	50 mA
Flash-Speicher	256 KB (8 KB von Bootloader benutzt)
SRAM	8 KB
EEPROM	4 KB
Taktfrequenz	16 MHz
Länge	101,52 mm
Breite	53,3 mm

2.2 Entwicklungsumgebung - Arduino IDE

Zur Programmierung des Arduino Boards wurde die Arduino Entwicklungsumgebung Arduino IDE in der Version 1.6.9 eingesetzt, die kostenlos auf der Arduino Webseite zur Verfügung steht. Dabei handelt es sich um eine plattformunabhängige Java-Anwendung, die auf Windows, Linux und Mac OS X läuft.
Die Entwicklungsumgebung ist sehr einfach zu bedienen und beinhaltet unter anderem einen Code-Editor, Compiler und einen Debugger. Nachdem man ein fertiges Programm geschrieben hat, wird es mit einem Klick auf Upload auf Fehler überprüft, kompiliert und auf den Chip des Arduinos übertragen.

2.3 Beispielprogramm (Sketch)

Für ein funktionsfähiges Programm reicht es aus, zwei Methoden zu definieren:

`setup()` - wird beim Start des Programms (entweder nach dem Übertragen auf das Board oder nach Drücken des Reset-Tasters) einmalig aufgerufen, um z. B. Pins als Eingänge oder Ausgänge zu definieren.
`loop()` - wird durchgehend immer wieder durchlaufen, solange das Arduino-Board eingeschaltet ist.

Nachfolgend ein kleines Programm (Sketch), welches an einem Arduino Board angeschlossene LED blinken lässt:

```
int ledPin = 13; // die LED ist an Pin 13 angeschlossen, was in der Variablen
                 // ledPin gespeichert ist

void setup() {
    pinMode(ledPin, OUTPUT); // legt den LED-Pin als Ausgang fest
}

void loop() {
    digitalWrite(ledPin, HIGH); // LED anschalten (auf HIGH setzen)
    delay(1000); // 1 Sekunde warten
    digitalWrite(ledPin, LOW); // LED ausschalten (auf LOW setzen)
    delay(1000); // 1 Sekunde warten
}
```

3.0 Vorbereitung des Projekts „RFID-Zugangskontrolle“

3.1 Gehäuse

Das Gehäuse des Miniaturhauses besteht größtenteils aus dünnen Sperrholzplatten, da diese einfach zu bearbeiten sind und ein eventueller Nachbau einfach ist. Ein weiterer Vorteil ist, dass dieses Material kostengünstig ist. Das Gehäuse hat die Maße 215 x 165 x 155 mm. Diese Größe ist so gewählt, dass genug Raum für das Arduino Mega 2560 Board, die Elektromagnetverriegelung, die Relaisplatine, sowie für weitere Bauteile und Kabel ist.

3.2 Benötigte Bauteile

Um eine RFID-Zugangssteuerung mit dem Arduino zu realisieren, werden folgende Bauteile benötigt:

- Arduino Mega 2560 Board
- Elektromagnetverriegelung
- RFID-Reader-Modul (RFID-RC522)
- Relais
- LCD (optional)
- Widerstand
- Potentiometer
- Steckbrett (Breadboard)
- diverse Patchkabel/Jumper-Kabel
- 12V Netzadapter für die Elektromagnetverriegelung

Außer der elektronischen Bauteile wurden Schrauben und Muttern für die Scharnieren verwendet. Da ein Miniaturhaus gebaut wurde, werden diese Teile für die Montage der Tür verwendet. Das Holz des Hauses wurde mit Holzkleber festgeklebt.
Die Elektromagnetverriegelung wurde ebenfalls mit Schrauben und Muttern befestigt.
Damit alle Teile fest sitzen und vor Erschütterung Stand halten, wurde an einigen Stellen Heißklebestoff eingesetzt.

3.3 Pinbelegung und Verbindungen

RFID-Reader-Modul (RFID-RC522):

RFID-Reader-Modul Pin	Arduino Pin
SDA	53
SCK	52
MOSI	51
MISO	50
IRQ	unbelegt
GND	GND
RST	7
3,3V	3,3V

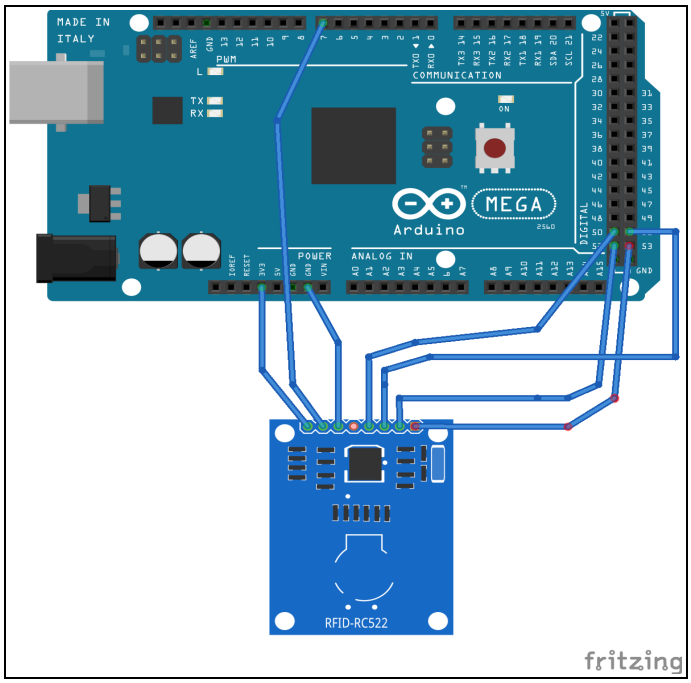


Abbildung 1: RFID-Reader am Arduino

Relaisplatine:

Relaisplatine Pin	Arduino Pin
VCC	5V
GND	GND
IN	38

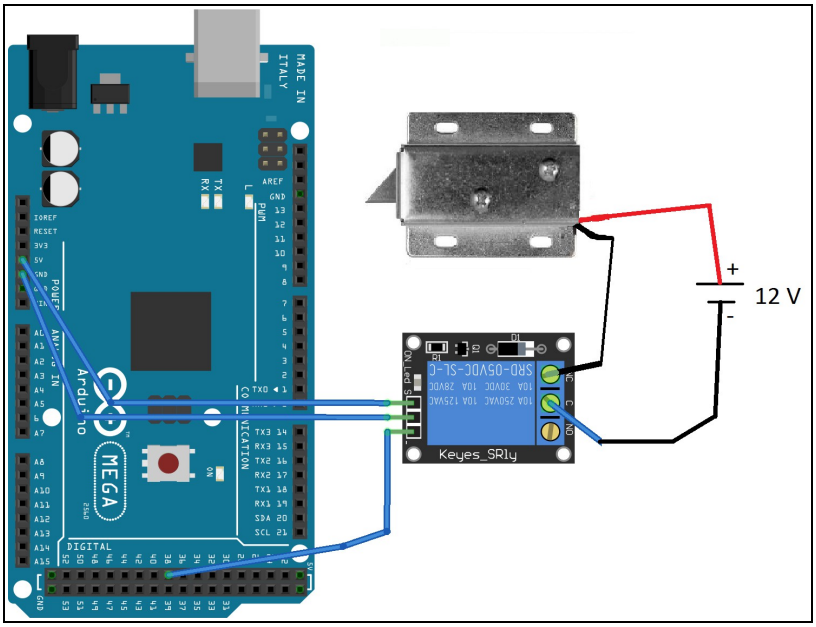


Abbildung 2: Relais und Elektromagnetverriegelung am Arduino

LCD:

LCD Pin	Arduino Pin
VSS (GND)	GND
VDD (+)	5V
V0 (Kontrast-Regelung)	GND
RS (Register Select)	12
RW (Read/Write)	GND
E (Enable)	11
D0 (Data 0)	unbelegt
D1 (Data 1)	unbelegt
D2 (Data 2)	unbelegt
D3 (Data 3)	unbelegt
D4 (Data 4)	5
D5 (Data 5)	4
D6 (Data 6)	3
D7 (Data 7)	2
A (LED Hintergrundbeleuchtung +)	5V
K (LED Hintergrundbeleuchtung -)	GND

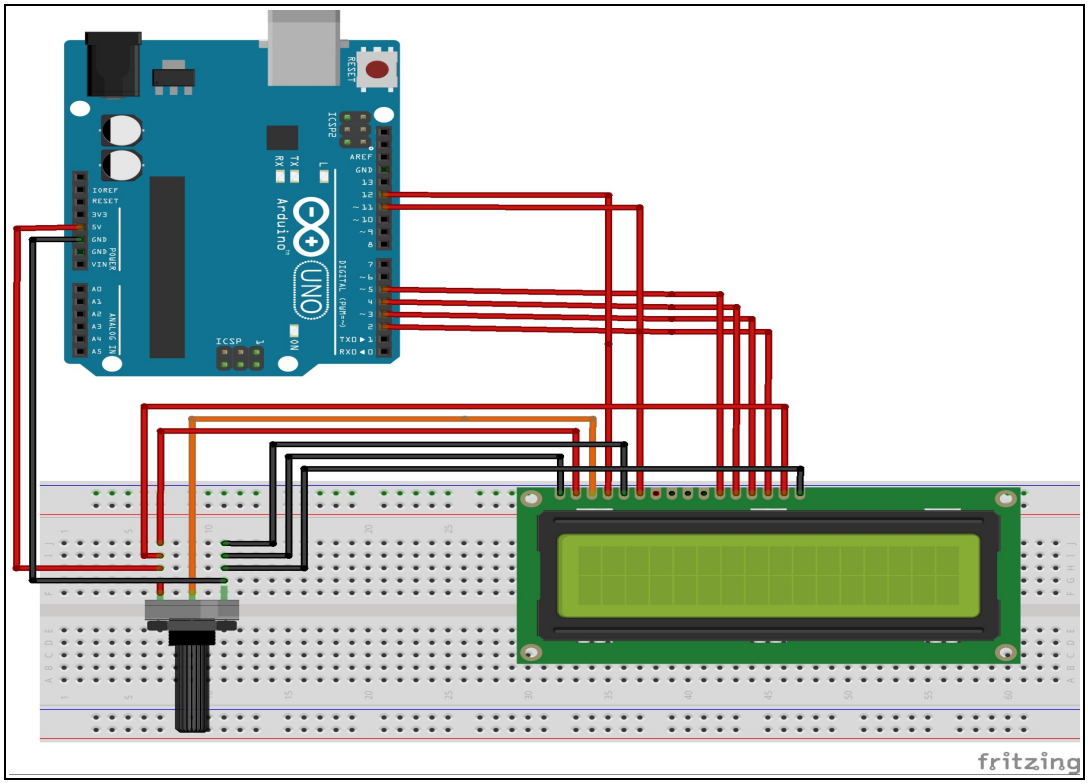


Abbildung 3: LCD mit Potentiometer am Arduino

3.4 Verwendete RFID-Tags

Die RFID-Tags, die in diesem Projekt verwendet wurden, sind 13,56 MHz RFID-Tags, da der RFID-Reader (RFID-RC522) kontaktlos mit 13,56 MHz Tags erkennt. Die Tags können bis zu 10 cm gelesen werden. Aufgrund der hohen Frequenz müssen die Tags sehr nah (ca. 1 cm) vor den Reader gehalten werden.

Abb. 4: RFID-Tag

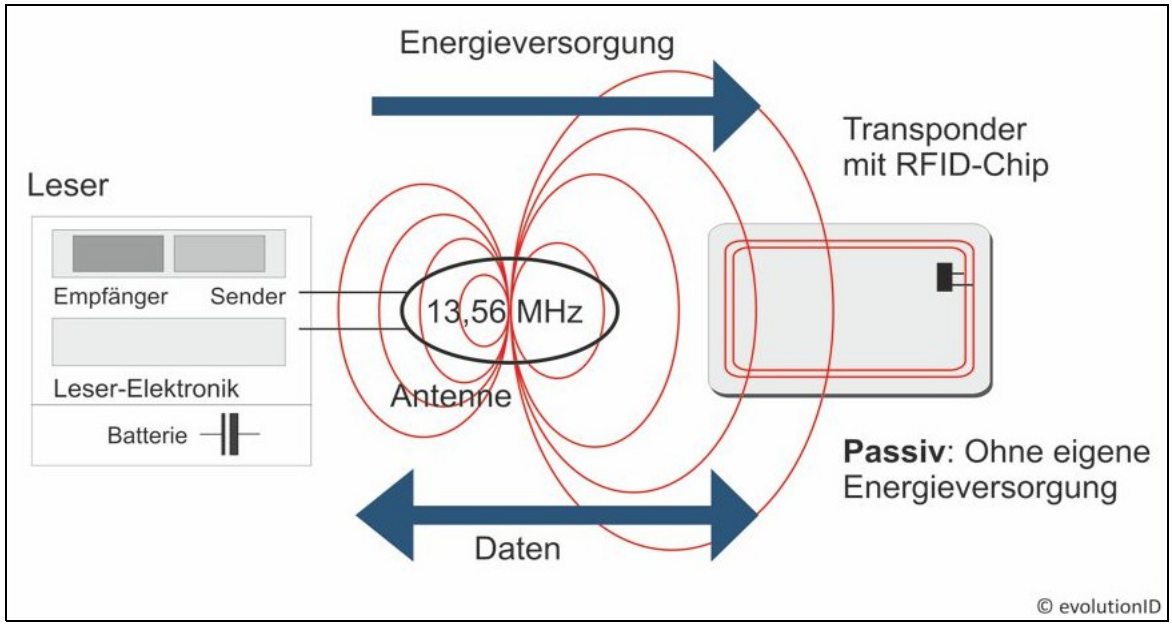


Abbildung 5: Funktionsweise RFID-Reader und Transponder

4.0 Programmierung des Arduinos

Das Arduino Board wurde mit Hilfe der Entwicklungsumgebung Arduino IDE und der Programmiersprache C programmiert (siehe Kapitel 2.2).
Für die Programmierung wurden vier Libraries benutzt, die schon in der Arduino IDE enthalten waren oder aus dem Internet geladen werden können:

SPI.h	Library für serielle Kommunikation (in Arduino enthalten)
LiquidCrystal.h	Library für LCD (in Arduino enthalten)
MFRC522.h	Library für den RFID-Reader (im Internet verfügbar)
Time.h	Library für die Zeiterfassung (im Internet verfügbar)

In den nachfolgenden Kapiteln wird auf das geschriebene Quellcode, welcher im Anhang zu finden ist, näher eingegangen.

4.1 Erklärung des Quellcodes

Teil 1:
Im oberen Teil des Quellcodes werden die benötigten Bibliotheken (siehe Kapitel 4.0) mit dem Befehl „include“ eingebunden.

Teil 2:
Anschließend werden Variablen für die Pins definiert, an denen am Arduino die Hardware-Teile angeschlossen sind bzw. im weiteren Programm angesprochen werden. Außerdem wird eine Array-Liste mit den IDs der RFID-Tags angelegt, die Zugriff haben sollen.

Teil 3:
Nun folgen alle Funktionen, die benötigt werden. Die setup- und loop-Funktion reichen aus, um ein funktionsfähiges Programm zu schreiben. Da dieses Projekt umfangreicher ist, werden weitere Funktionen benötigt.
Nachfolgend werden alle verwendeten Funktionen erklärt:

setup-Funktion:
Diese Funktion wird beim Start des Programms (entweder nach dem Übertragen auf das Board oder nach Drücken des Reset-Tasters auf dem Arduino-Board) einmalig aufgerufen. In dieser Funktion werden die Bauteile wie RFID-Reader, Relais und LCD initialisiert. Zusätzlich wird eine serielle Verbindung aufgebaut, um z.B. den Serial Monitor zu benutzen. Mit setTime wird bei Programmstart die Uhrzeit und das Datum eingestellt.

is_code_in_array_list-Funktion:
Diese Funktion wertet aus, ob die an sie übergebene ID in der oben definierten Array-Liste vorkommt. Es wird ein TRUE oder FALSE zurückgegeben.

loop-Funktion:
Die loop-Funktion ist der eigentliche Teil des Programms. Nachdem die setup-Funktion ausgeführt wird, wird diese Funktion aufgerufen und solange ausgeführt bis Arduino ausgeschaltet wird.
Wie der Name loop=Schleife/Wiederholung schon sagt, wird diese Funktion wiederholend ausgeführt und nach Aktionen gewartet.
In unserem Fall steht auf dem LCD: Bitte ID-Karte vorhalten... Diese Meldung wird solange angezeigt bis ein RFID-Tag in die Nähe des RFID-Readers gehalten wird. Wenn vom RFID-Reader die interne UID (Unique Identification) erkannt wird, werden die einzelnen Blöcke aufaddiert und mit zehn multipliziert, damit nicht eine große Zahl entsteht, da das Programm diese Zahl sonst nicht verarbeiten kann. Es entsteht eine berechnete ID, die an die Funktion is_code_in_array_list gesendet wird (siehe oben).
Wenn es sich um eine gültige (in der Array gespeicherte) ID handelt, schaltet das Relais und die Verriegelung wird eingezogen und die Tür lässt sich öffnen. Gleichzeitig wird auf dem LCD eine Meldung ausgegeben: Zutritt gewährt, Ihre ID: XXXXXXX
Nachdem 5 Sekunden gewartet wurde, schaltet das Relais erneut und die Elektromagnetverriegelung bekommt kein Strom mehr. Die Verriegelung rastet wieder zurück. Die Tür kann anschließend zugezogen werden. Um das ganze protokollarisch festzuhalten, wird auf dem Serial Monitor die UID, die berechnete ID und der Zeitpunkt an dem das Ereignis stattfand festgehalten.
Wenn jedoch eine ungültige Kennung festgestellt wird, wird auf dem LCD mitgeteilt, dass der Zutritt verweigert wurde und die Tür wird nicht entsperrt. Dieser unautorisierter Vorgang wird ebenfalls protokolliert.

digitalClockDisplay-Funkion:
Wenn diese Funktion aufgerufen wird, sendet sie die aktuelle Uhrzeit und Datum zurück. Weil in der setup-Funktion die Zeit bei Programmstart eingestellt wurde und im Hintergrund weiterläuft, kann die aktuelle Zeit abgefragt werden.

printDigits-Funktion:
Diese Funktion macht nichts anderes als eine Null vor die Einer 0-9 zu schreiben, um eine für den Menschen konforme Ausgabe zu erhalten.
Beispiel:
`10:3:9 1.6.2016`
wird zu
`10:03:09 01.06.2016`

4.2 Zustandsgraph

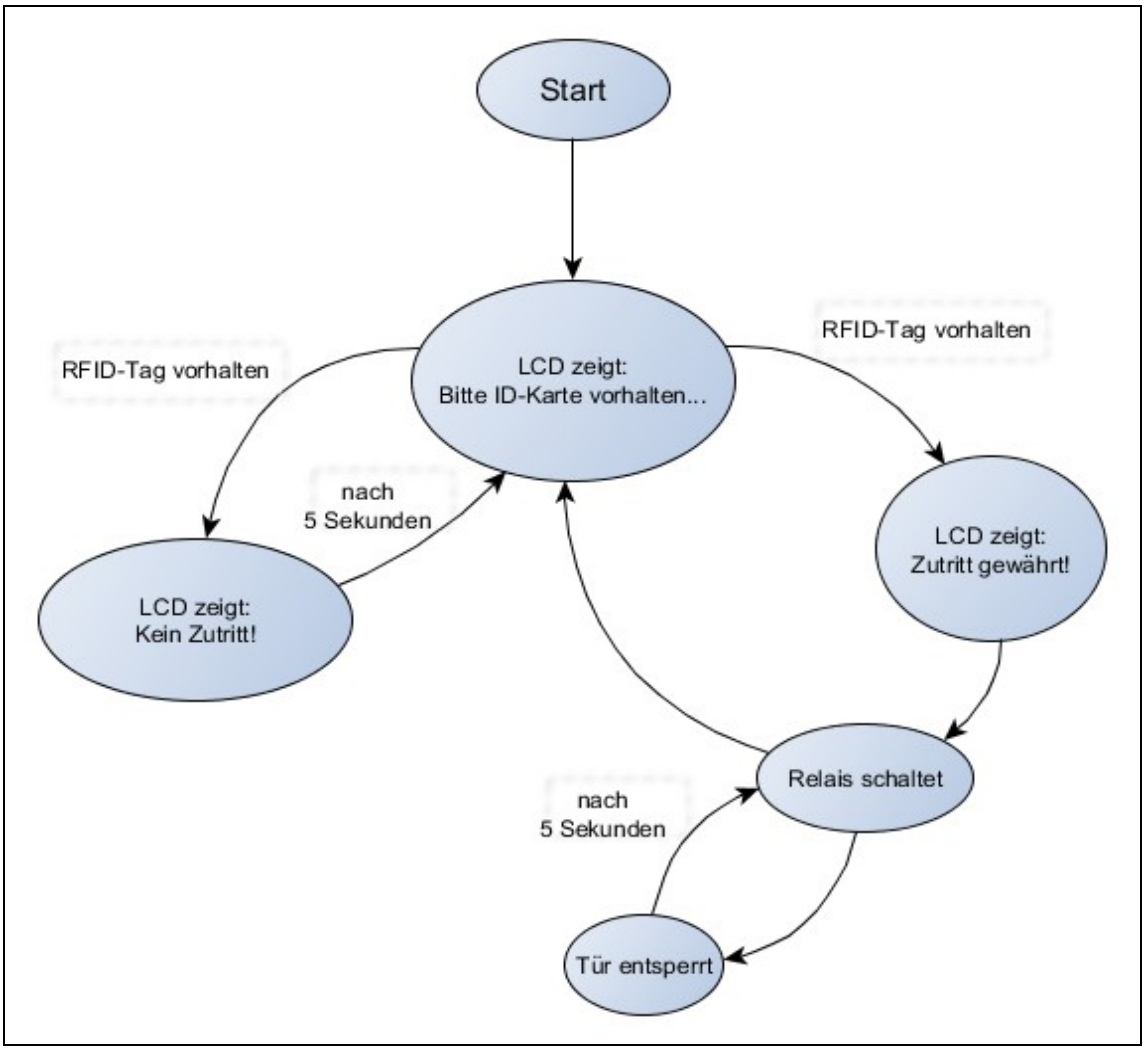


Abbildung 6: Zustandsgraph der RFID-Zugangskontrolle

5.0 Fazit

Heutzutage ist es einfach mit etwas Einarbeitung komplexe Projekte zu realisieren. Sowohl die leistungsfähige Arduino Hardware mit ihren diversen Erweiterungsmöglichkeiten als auch die benutzerfreundliche Arduino IDE ermöglichen es eine RFID-Zugangskontrolle zu bauen. Statt Geld für teure Technik auszugeben, können Technik versierte kostengünstig und mit wenig Aufwand solche oder ähnliche Projekte realisieren. Dabei kann jeder sein eigenes Projekt nach eigenen Bedürfnissen anpassen bzw. erweitern.
Das Programmieren mit dem Arduino lehrt einen lernbereiten Menschen in vieler Hinsicht. Von der Elektronik bis hin zur Programmierung und Inbetriebnahme werden viele Erfahrungen gesammelt, die für spätere Projekte hilfreich sind. Der Umgang mit Mikrocontrollern wird näher gebracht. In Schulen, Hochschulen, Universitäten, Unternehmen oder auch im privaten Hobbybereich ist der Arduino bestens geeignet.

Anhang

Quellcode des Projektes:

```
/*
    Projektarbeit
    Hochschule Ostwestfalen-Lippe
    Fachbereich 5 - Eletrotechnik und Technische Informatik
    Projekt: RFID-Zugangskontrolle
    Autor: Cihan Aydogdu
    Version: 1.0.3
*/

// Benoetigte Bibliotheken werden eingebunden
#include <SPI.h> // SPI-Bibiothek hinzufuegen
#include <MFRC522.h> // RFID-Bibiothek hinzufuegen
#include <LiquidCrystal.h> // LCD-Bibliothek hinzufuegen
#include <Time.h> // Zeit-Bibliothek hinzufuegen

// Zeit und Datum festlegen
#define HR 14
#define MN 0
#define SC 0
#define DY 10
#define MT 6
#define YR 16

// RFID Pins definieren
#define RFID_SDA_PIN 53
#define RFID_RST_PIN 7

// LCD Pins definieren
#define LCD_DATA7 2
#define LCD_DATA6 3
#define LCD_DATA5 4
#define LCD_DATA4 5
#define LCD_ENABLE 11
#define LCD_REGISTER_SELECT 12

// Relais Pin definieren
#define RELAY_IN 38

// Array-Liste mit gueltigen IDs (mit Dummy-Werten)
// ID1: 1509570
// ID2: 1504830
                                        //Ab hier Dummy-Werte und dienen zur
Demonstration
                                        //muessen spaeter entfernt werden
unsigned long int id_list[] = { 1509570, 0000000, 1111111, 2222222 };

// RFID-Empfaenger benennen und Pins angeben
MFRC522 mfrc522(RFID_SDA_PIN, RFID_RST_PIN);
```

```
// LCD benennen und Pins angeben
LiquidCrystal lcd(LCD_REGISTER_SELECT, LCD_ENABLE, LCD_DATA4, LCD_DATA5,
LCD_DATA6, LCD_DATA7);

// Setup-Funktion (Initialisierung der Bauteile)
void setup()
{
  Serial.begin(9600); // Serielle Verbindung starten (Monitor)
  SPI.begin(); // SPI-Verbindung aufbauen
  mfrc522.PCD_Init(); // Initialisierung des RFID-Empfaengers
  pinMode(RELAY_IN, OUTPUT); // Initialisierung des Relais
  setTime(HR,MN,SC,DY,MT,YR); // Uhrzeit und Datum werden bei Programmstart
festgelegt
  delay(500); // Kurze Wartezeit damit LCD-Anzeige nicht "flackert"
  lcd.begin(16, 2);  // Initialisierung des LCDs
}

// Abfrage-Funktion (ist ID in Array-Liste oder nicht)
bool is_code_in_array_list(long int candidate){
    int arraysize = sizeof(id_list) / sizeof(long int);
    int i;
    for (i=0; i < arraysize; i++)
    {
      if (id_list[i] == candidate)
      {
        return true;
      }
    }
    return false;
}

// Loop-Funktion (eigentlicher Teil, der wiederholend waehrend des Betriebes
laeuft)
void loop()
{
  //hier KEIN lcd.clear(), sonst "flackert" LCD
  lcd.setCursor(0, 0);
  lcd.print("Bitte ID-Karte");
  lcd.setCursor(0, 1);
  lcd.print("vorhalten...");

  if ( ! mfrc522.PICC_IsNewCardPresent()) // Wenn eine Karte in Reichweite
ist...
  {
    return; // gehe weiter...
  }

  if ( ! mfrc522.PICC_ReadCardSerial()) // Wenn ein RFID-Sender ausgewaehlt
wurde...
  {
    return; // gehe weiter...
  }
```

```
  // In dieser Variable wird die ID gespeichert
  unsigned long int code = 0;

  // Alle 4 Bloecke werden ausgelesen und zusammengefasst
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {
    code = ((code+mfrc522.uid.uidByte[i])*10);
  }

  // Ausgabe fuer Serial-Monitor
  Serial.print("Die ID des RFID-Tags lautet:  ");
  // UID des RFID-Tags wird ausgelesen
  // Ausgabe in dezimal, hexadezimal oder binaer Form moeglich
  for (byte j = 0; j < mfrc522.uid.size; j++)
  {
    Serial.print(mfrc522.uid.uidByte[j], DEC);
    //Serial.print(mfrc522.uid.uidByte[j], HEX);
    //Serial.print(mfrc522.uid.uidByte[j], BIN);
    Serial.print(" ");
  }
  Serial.println();
  Serial.print("Berechnete Kennung:           ");
  Serial.print(code);
  Serial.println();

  // Abfrage, ob ID in Array-Liste ist
  if (is_code_in_array_list(code))
  {
    // Ausgabe fuer Serial-Monitor (z.B. als Protokoll)
    Serial.println("Autorisierung:                JA");
    digitalClockDisplay();
    Serial.println();

    // Ausgabe fuer LCD (z.B. fuer Mitarbeiter)
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Zutritt gew\xE1hrt");
    lcd.setCursor(0, 1);
    lcd.print("Ihre ID: ");
    lcd.print(code);

    // Tuer entsperren und nach 5 Sekunden wieder sperren
    digitalWrite(RELAY_IN, HIGH);
    delay(5000); // 5 Sekunden
    digitalWrite(RELAY_IN, LOW);

    // 1,5 Sekunden warten, damit LCD nicht "flackert" und dann LCD-Anzeige
loeschen
    delay(1500);
    lcd.clear();

  }
  else
```

```
  {
    // Ausgabe fuer Serial-Monitor
    Serial.println("Autorisierung:                NEIN");
    digitalClockDisplay();
    Serial.println();

    // Ausgabe fuer LCD
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Kein Zutritt!");
    lcd.setCursor(0, 1);
    lcd.print("Ihre ID: ");
    lcd.print(code);

    delay (5000); // fuer 5 Sekunden
    lcd.clear();
  }
}

// Funktion fuer Datum und Uhrzeit
void digitalClockDisplay(){
  Serial.print("Zeitpunkt:                    ");
  Serial.print(hour());
  Serial.print(":");
  printDigits(minute());
  Serial.print(":");
  printDigits(second());
  Serial.print("  ");
  printDigits(day());
  Serial.print(".");
  printDigits(month());
  Serial.print(".");
  Serial.print(year());
  Serial.println();
}

// Funktion, um eine Null vor die Einer zu schreiben
void printDigits(int digits){
  if(digits < 10)
  {
    Serial.print('0');
  }
  Serial.print(digits);
}
```

Quellenverzeichnis

[1]	http://www.heise.de/download/arduino-ide-1184057.html
[2]	https://de.wikipedia.org/wiki/Arduino_(Plattform)
[3]	https://en.wikipedia.org/wiki/List_of_Arduino_boards_and_compatible_systems

Abbildungsverzeichnis

Titelbild	Eigenes Foto
[1]	Eigene Grafik mit Fritzing erstellt
[2]	Eigene Grafik mit Fritzing erstellt
[3]	http://www.funduino.de/wp-content/uploads/2016/01/LCD-Arduino-Anleitung.jpg
[4]	http://www.emartee.com/Images/websites/emartee.com/tag.jpg
[5]	http://www.soaa-standard.com/images/rfidfunktion01b.jpg
[6]	Eigene Grafik mit yEd Graph Editor erstellt