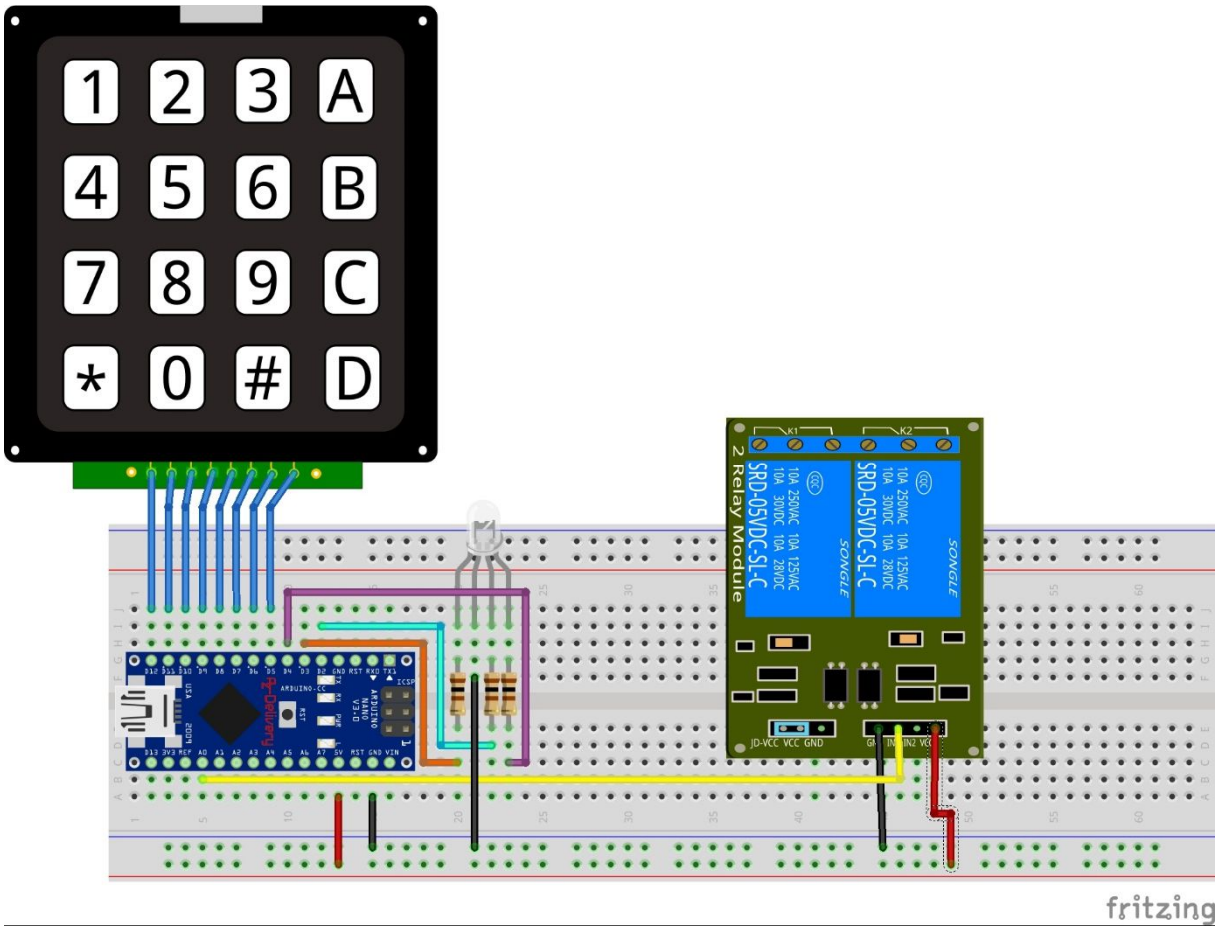
Ein Codeschloss mit unserem Arduino Teil 1

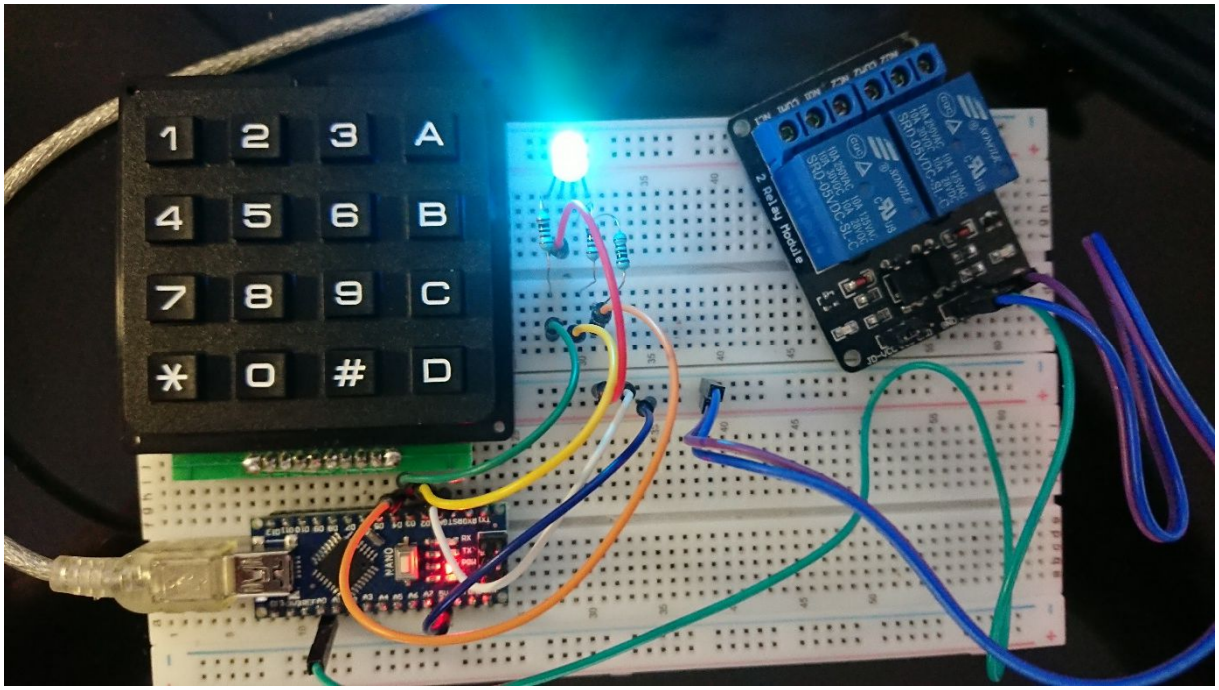
Hallo und willkommen zu einer neuen Reihe rund um unser Lieblingsthema: Arduino. Im heutigen ersten Teil eines mehrteiligen Blogs bauen wir uns die Basisversion eines Codeschlosses. Das Funktionsprinzip dabei ist sehr simpel: Eine bekannte Geheimzahlkombination, die vorab in unseren Code „einegbaut“ wird und dabei eine Länge von 1-20 hat, schaltet bei richtiger Eingabe ein Relais um. Eine RGB Led zeigt dabei durch die Farben „rot“ oder „grün“ an, ob die richtige Zahlenkobination eingegeben wurde. Wir werden in den folgendenm Teilen der Codeschloss Reihe die Funktionen, die Sicherheit aber auch den Komfort unseres Codeschloss steigern. Wir nutzen als Controller einen Arduino Nano, der mit seiner Rechenkapazität und Schnittstellen bestens geeignet ist für unser Vorhaben. Wir brauchen folgende Teile für unser Codeschloss aus unserem Shop:

Anzahl	Bezeichnung	Anmerkung
1	Relais Modul	
1	Arduino Nano	
1	4x4 Keypad	
3	Widerstände 120 Ohm	
1	RGB Led	

Die Teile bauen wir nun nach folgendem Schaltplan zusammen:



Aufgebaut auf einem Breadboard sieht das ganze dann schon ziemlich passabel aus:



(Zu sehen: 4x4 Keypad mit Arduino Uno, RGB Led, Relaisblock)

Bevor wir nun ans an das hochladen des Codes auf das Codeschloss machen können, müssen wir noch die Library für das Keypad runterladen und importieren. Gehen Sie dazu zu diesem Link: http://robojax.com/learn/arduino/robojax-Keypad.zip und downloaden die ".zip"-Datei. Um sie zu Ihrem Uno hinzuzufügen kann in der IDE der Punkt: Sketch > Include Library > Add .ZIP Library genutzt werden. Hier fügen wir die vorab heruntergeladene ZIP Datei ein. Damit steht die „Keypad.h“ Bibiothel zur Verfügung. Als nächstes muss die Zeile:

```
byte PinCode[MaxPinCodeLength] = {1,2,3,13};
```

Mit einem eigenen Code zwischen 1 bis 20 Zahlen und Buchstaben angepasst werden.

Die Zuordnung ist dabei wie folgt:

A = 12, B = 23, C = 33, D = 43, 0 = 10, Zahlen 1..9 = 1..9, * = Eingabe löschen, # = Eingabe bestätigen.

Beispiel für den 5-Stelligen Code: 7 A 0 B 1:

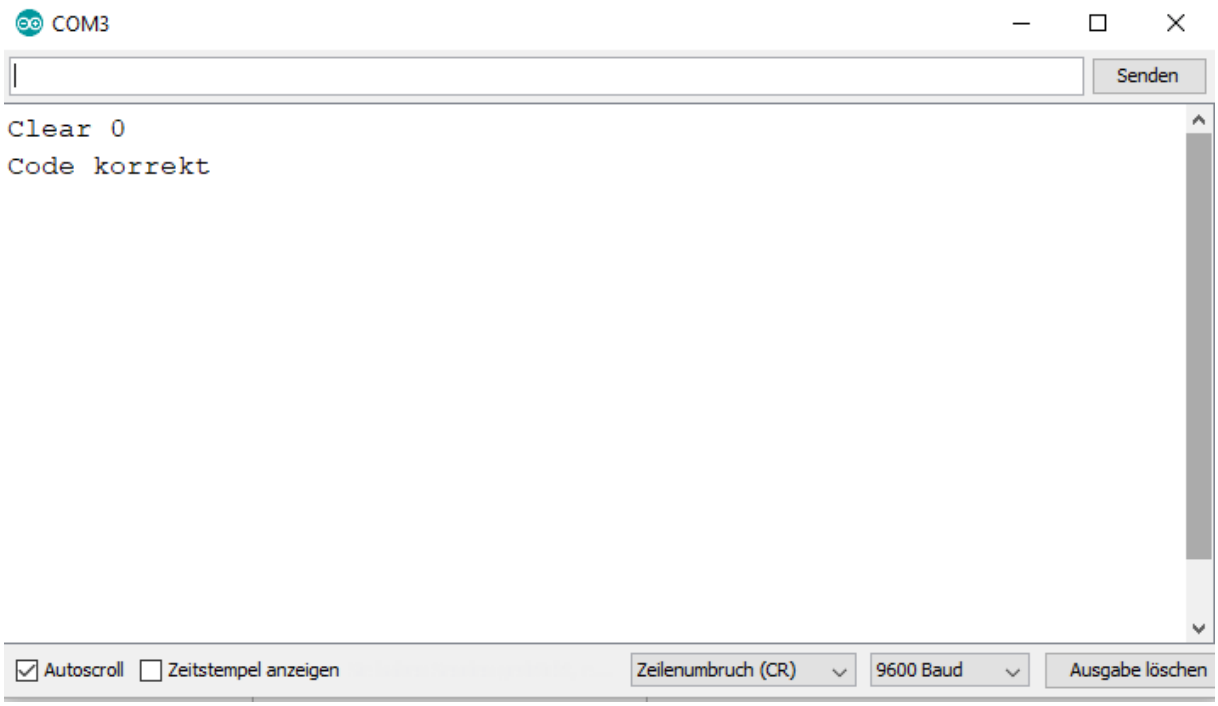
```
byte PinCode[MaxPinCodeLength] = {7,12,10,23,1};
```

Nach Anpassung kann der folgende Code nun hochgeladen werden:

```
// Codeschloss Tobias Kuch 2020 GPL 3.0
#include <Keypad.h>

#define RGBLED_G 2  //LED Port
#define RGBLED_B 4
#define RGBLED_R 3
#define RELAIS_A A0

const byte ROWS = 4;
const byte COLS = 4;
const byte MaxPinCodeLength = 20;

char keys[ROWS][COLS] = {
                         {1,2,3,13},
                         {4,5,6,23},
                         {7,8,9,33},
                         {40,10,42,43}
                        };
byte rowPins[ROWS] = {8,7,6,5}; //5,6,7,8};
byte colPins[COLS] = {12,11,10,9}; // {9,10,11,12};
byte KeyPadBuffer[MaxPinCodeLength];
byte PinCode[MaxPinCodeLength] = {1,2,3,13}; // Standard Pincode: 123A  - Bitte
Ändern gemäß Beschreibung -
byte BufferCount = 0;

Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

void setup()
{
 Serial.begin(9600);
 pinMode(RGBLED_G,OUTPUT); // Ausgang RGB LED Grün
 pinMode(RGBLED_R,OUTPUT); // Ausgang RGB LED Rot
 pinMode(RGBLED_B,OUTPUT); // Ausgang RGB LED Blau
 pinMode(RELAIS_A,OUTPUT); //Relais Output
 digitalWrite(RELAIS_A,HIGH);
 BufferCount = 0;
 for (byte a = 0; a <= MaxPinCodeLength; a++)
  {
  KeyPadBuffer[a] = 0;
  }
 digitalWrite(RGBLED_R,LOW);
 digitalWrite(RGBLED_G,HIGH);
 digitalWrite(RGBLED_B,HIGH);
}

void loop()
{
   char key = keypad.getKey();
   if(key)
   {
     if(key ==40)   // Clear Keypad Buffer  Key: *
```

```
      {
       for (byte a = 0; a <= MaxPinCodeLength; a++)
        {
        KeyPadBuffer[a] = 0;
        }
       digitalWrite(RGBLED_R,HIGH);
       digitalWrite(RGBLED_G,LOW);
       digitalWrite(RGBLED_B,HIGH);
       Serial.print("Clear ");
       Serial.println(BufferCount);
       BufferCount = 0;
      } else
     if(key ==42)   // Enter Keypad Buffer  Key: #
      {
       bool AcceptCode = true;
       for (byte a = 0; a <= BufferCount; a++)
        {
        if (!(PinCode[a] == KeyPadBuffer[a])) {AcceptCode = false; }
        }
        if (AcceptCode)
          {
            Serial.println("Code korrekt");
            digitalWrite(RELAIS_A,(!digitalRead(RELAIS_A)));
            digitalWrite(RGBLED_R,LOW);
            digitalWrite(RGBLED_G,HIGH);
            digitalWrite(RGBLED_B,LOW);
          } else
          {
            digitalWrite(RGBLED_R,HIGH);
            digitalWrite(RGBLED_G,LOW);
            digitalWrite(RGBLED_B,LOW);
          }
        for (byte a = 0; a <= MaxPinCodeLength; a++) { KeyPadBuffer[a] = 0; }
        BufferCount = 0;
        delay(1000);
        digitalWrite(RGBLED_R,LOW);
        digitalWrite(RGBLED_G,HIGH);
        digitalWrite(RGBLED_B,HIGH);
      } else
      {
       KeyPadBuffer[BufferCount] = key;
       if (BufferCount < MaxPinCodeLength ) { BufferCount++; }
      }
   }
}
```

Auf der seriellen Schnittstelle werden zur Funktionskontrolle einige Statusmeldungen ausgegeben.



Ich wünsche viel Spaß beim Nachbau und bis zum nächsten Teil der Reihe