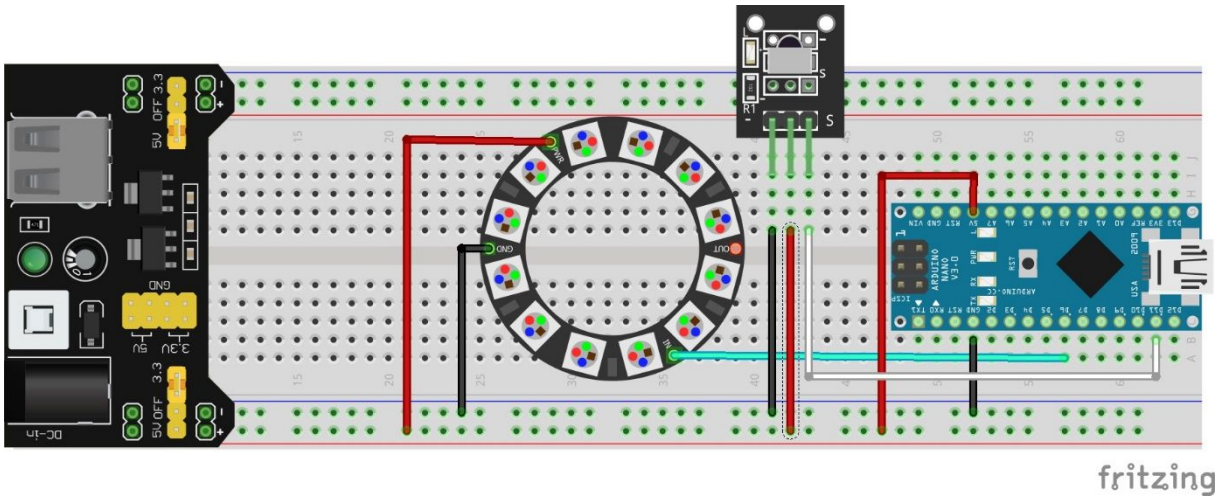
Eine Fernsteuerung für unsere Stimmungslaterne

Die Laterne steht bei mir zuhause in einer Ecke, die nicht so leicht mit der Hand zu erreichen ist. Jedes Mal also den Netzteilstecker ziehen oder einen Schalter betätigen um die Laterne ein bzw. auszuschalten, finde ich daher etwas mühselig. Das muss doch auch anders gehen! Und ja, es geht auch anders, nämlich, getreu dem Fernseher Vorbild, mit einer Infrarot Fernbedienung. Bevor wir uns jedoch bequem in die Couch setzen können und unser Laternchen mit der Fernbedienung ein und ausschalten können, müssen wir wie immer einige Äderungen und Vorbereitungen treffen. Das wichtigste was wir zuerst benötigen, ist zunächst die Fernbedienung selbst. Ich habe mich für meine Fernseher Fernbedienung entschieden, die noch ein paar ungenutzte Tasten hat, die ich dazu verwenden möchte. Dazu aber später mehr. Das nächste was wir Hardwareseitig benötigen, ist das KY-022 IR-Empfänger Modul. aus unserem AZ Delivery Shop. Dieses benötigen wir, damit unser Nano die Infrarot Signale empfangen und auswerten kann. Wir verdrahten also unser neues Modul wie folgt:



Wie zu erkennen ist, werden die Signale des IR Receivers an Pin 11 des Nanos geleitet und von dem Arduino ausgewertet. Damit der Arduino die Infrarotsignale auswerten kann brauchen wir für unseren Code die Bibliothek IRRemote die wir herunterladen und vorher in der IDE installieren müssen. Nach der Installation der Bibliothek laden wir folgenden Code hoch:

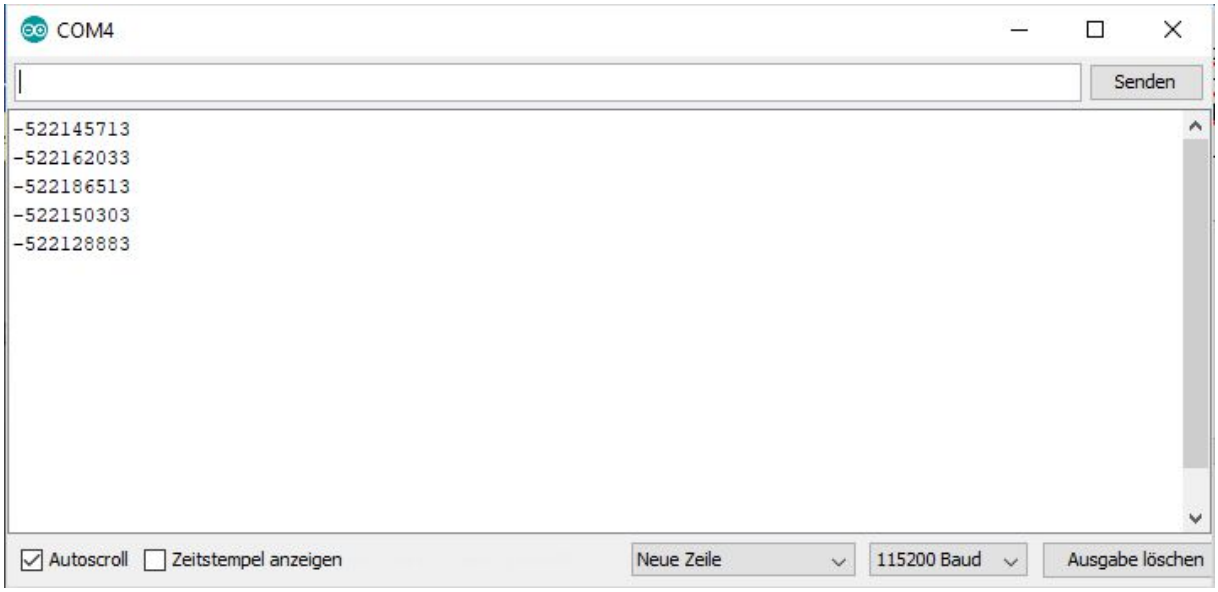
```
#include <Adafruit_NeoPixel.h>
#include <IRremote.h>

#define PIN       6  // Which pin on the Arduino is connected to the NeoPixels?
#define RECV_PIN 11  // define IR input pin on Arduino
#define NUMPIXELS 12 // How many NeoPixels are attached to the Arduino? //
Popular NeoPixel ring size

Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
IRrecv irrecv(RECV_PIN);
decode_results results; // decode_results class is defined in IRremote.h

long FirelastTime = 0;
long IRlastTime = 0;
long TimerlastTime = 0;
int interval;
long IRCode = 0;
long OLDIRCode = 0;
bool FireON = false;
bool FireOFF = false;
byte FireSequence = 0;

void setup()
  {
  Serial.begin(115200);
  while (!Serial);   //wait until Serial is established - required on some Platforms
  irrecv.enableIRIn(); // Start the receiver
  pixels.begin(); // INITIALIZE NeoPixels
```

```
  pixels.show(); // Initialize all pixels to 'off'
  interval = 400;
  randomSeed(analogRead(0));
  }

void SimulateFire (bool On, int FireSq)
  {
  if (millis()-FirelastTime>=interval)
  {
  if (On)
    {
      FireOFF = false;
      FirelastTime=millis();
      byte LightValue[NUMPIXELS* 3];
      byte FireColor = 60;
      interval = random(150,200);
      FireColor = 60; //random(0,50);
      for(int i=0; i<NUMPIXELS; i++)
        { // For each pixel...
        LightValue[i*3] = random(240,255); // 250
        LightValue[i*3+1] = random(30,60); // 50
        LightValue[i*3+2] = 0;
        }
      // Switch some lights darker
      byte LightsOff  = random(0,4);
      for(int i=0; i<LightsOff; i++)
        {
        byte Selected = random(NUMPIXELS);
        LightValue[Selected*3] = random(50,60);
        LightValue[Selected*3+1] = random(5,10);
        LightValue[Selected*3+2] = 0;
        }
      for(int i=0; i<NUMPIXELS; i++)
        { // For each pixel...
       pixels.setPixelColor(i, LightValue[i*3], LightValue[i*3+1], LightValue[i*3+2]);
        }
      noInterrupts();
      pixels.show();   // Send the updated pixel colors to the hardware.
      irrecv.resume();
      interrupts();
    }
  else
    {
    if (!(FireOFF))
      {
      pixels.clear();
      noInterrupts();
      pixels.show();   // Send the updated pixel colors to the hardware.
      irrecv.resume();
      interrupts();
      FireOFF = true;
```

```
     }
    }
  }
 }

long ReceiveIrCommand ()
{
  long result = 0;
    if (millis()-IRlastTime>=200)
     {
       IRlastTime=millis();
       if (irrecv.decode(&results))
         {
         result = results.value;
         irrecv.resume(); // Receive the next value
         return result;
         }
    //     irrecv.resume(); // Receive the next value
     }
  return 0 ;
}

void IRCommandProcessor (long IrCommand)
  {
   if (IRCode == OLDIRCode) { TimerlastTime=millis();}             //Some stuff about
debouncing IR Remote
   if (millis()-TimerlastTime >=400) { OLDIRCode = 0 ;}          //Some stuff about
debouncing IR Remote
   if ((IRCode < -1) & (IRCode != OLDIRCode))   // IR Signal Received
     {
     OLDIRCode = IRCode;                                        //Some stuff about
debouncing IR Remote
     switch (IRCode)
        {
        case -522164073:       // In my case a blue Switch on my TV - Remote
           {
           FireON = !FireON;
          }
        break;
        default:
        Serial.println(IRCode);
        break;
        }
     }
  }

void loop()
  {
   IRCode = ReceiveIrCommand();
   IRCommandProcessor(IRCode);
   SimulateFire(FireON,FireSequence);
```

```
}
```

Bevor wir nun aber unsere eigene Fernbedienung einbinden können, müssen wir vorher erst einmal das Terminalfester bei einer Schnittstellengeschwindigkeit von 115200 Baud öffnen. Nun halten wir unsere gewählte Fernbedienung vor den Sensor und drücken ein paar Tasten. Es sollte eine Ausgabe ungefähr wie folgt zu sehen sein:



Die negativen langen Zahlen die bei jedem Tastendruck auf der Fernbedienung angezeigt werden, entsprechen dem Infrarot Code, der jeweiligen Taste auf der Fernbedienung. Beispielsweise entspricht die Zahl -522145713 der Taste „1“ auf der Fernbedienung. Wollen wir also die Taste „1“ zum Ein und Ausschalten unserer Laterne verwenden, müssen wir diesen „Zahlencode“ in unserem Arduino Code eintragen. Dies tun wir an folgender Stelle im Code:

```
switch (IRCode)
        {
        case -522145713:       // Hier den eigenen ermittelten IR Zahlencode für
                               // die Funktion ein/aus eintragen
           {
           FireON = !FireON;
          }
```

Ab sofort wird bei Betätigung der ausgewählten Fernbedienungstaste der Code NICHT mehr im Seriellen Monitor angezeigt, jedoch bei Empfang die Laterne ein bzw. ausgeschaltet.

Der geneigte Leser wird wahrscheinlich schon erkannt haben, das über die „switch - case“ Struktur nicht nur Fernbedienungscodes zum Ein- und Ausschalten der Laterne sich nutzen lassen, sondern damit beliebige Erweiterungen der Funktionen realisieren

lassen. Im nächsten Teil der Reihe schauen wir uns genau das an und erweitern die Funktionen unserer Laterne um verschiedene Feuersimulationstypen. Ich freue mich schon euch bald den nächsten Teil zur Verfügung stellen zu können und wünsche euch viel Spaß beim Nachbauen. Fragen sind natürlich wie immer herzlich willlkommen.