Mehrere „Feuer-Programme“ für unsere Stimmungslaterne

Soweit funktioniert unsere Laterne schon ganz gut, und kann schon mal mit den Basisfunktionen fernbedient werden. Jedoch, und das hatte ich ja auch schon im vorherigen teil etwas durchscheinen lassen, werden wir heute im letzten Teil unserer Stimmungslaternenreihe unsere Laterne noch etwas „pimpen“ und ihr nicht nur ein Flackerprogramm spendieren, sondern insgesamt 5 verschiedene Feuersimulations-Programme, die jeweils auch noch per Fernsteuerung einzeln auswählbar sind !

Der Code kann und soll! Natürlich auch um eigene Simulationsalgorithmen ergänzt werden, sodass dem geneigten Leser hier eine unendliche Anzahl an Möglichkeiten zur Verfügung steht der eigenen Kreativität freien Lauf zu lassen. Beschäftigen wir uns aber erst einmal mit den 5 Algorithmen die ich eingebaut habe und die voneinander abgeleitet sind.

Diese währen im Einzelnen:

- Programm 1: Kein Flackern, schnell wechselnde unterschiedliche Flammentemperatur.
- Programm 2: Kein Flackern, dauerhafte gleiche Flammentemperatur. Stehendes Licht.

- Programm 3: Starkes, häufiges Flackern, schnell wechselnde unterschiedliche Flammentemperatur.
- Programm 4: Starkes, seltenes Flackern, gleiche Flammentemperatur.
- Programm 5: mildes, häufiges Flackern, schnell wechselnde unterschiedliche Flammentemperatur.
- Programm 6: Starkes, seltenes Flackern, schnell wechselnde unterschiedliche Flammentemperatur.

Programm 5 ist das „Standardprogramm“ aus dem 2 Teil dieser Maker-Reihe.

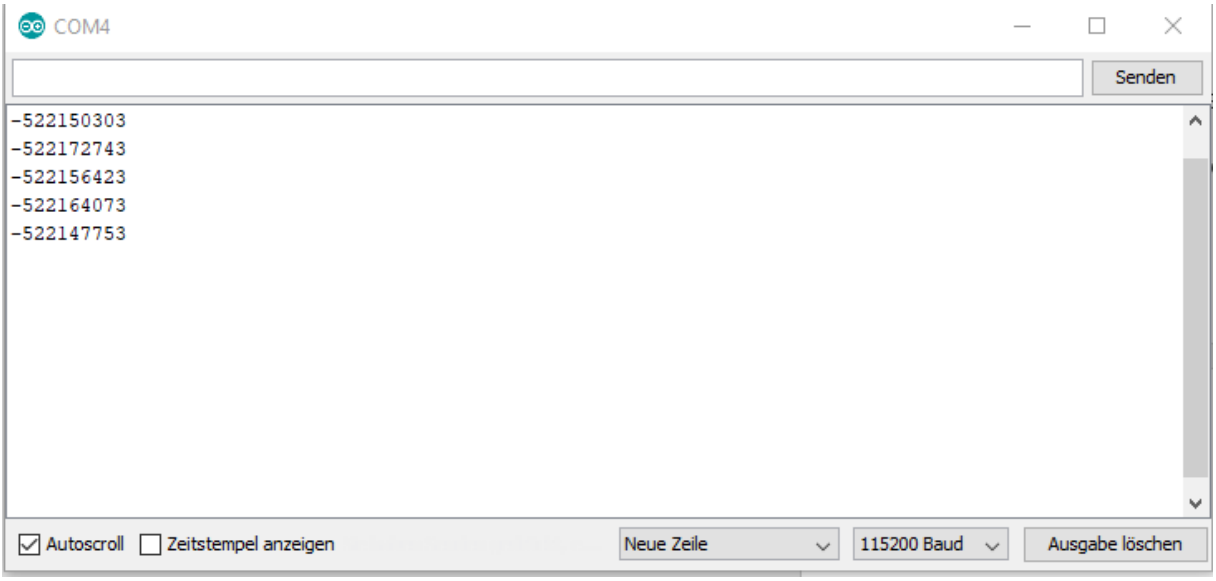
Der geneigte Leser wird wahrscheinlich schon erkannt haben, das über die „switch-case“ Struktur dies unterschiedlichen Simulationsalgorithmen aufgerufen, bzw. gesteuert werden. Dies passiert min Anhängigkeit des eingelesenen Fernsteuercodes in folgendem Abschnitt:

```
switch (IRCode)
        {
        case -522182433:        // In my case 1  on my TV - Remote
           {
           FireON = !FireON;
           FireSequence = 1;
          }
        break;
        case -522149793:        // In my case 2  on my TV - Remote
           {
           FireON = !FireON;
           FireSequence = 2;
          }
        break;
        case -522166113:        // In my case 3  on my TV - Remote
           {
           FireON = !FireON;
           FireSequence = 3;
          }
        break;
        case -522186513:        // In my case 4  on my TV - Remote
           {
           FireON = !FireON;
           FireSequence = 4;
          }
        break;
        case -522153873:        // In my case 5  on my TV - Remote
           {
           FireON = !FireON;
           FireSequence = 5;
          }
        break;
        case -522173873:        // In my case 6  on my TV - Remote
           {
```

```
        FireON = !FireON;
        FireSequence = 5;
       }
    break;

    default:
    Serial.println(IRCode);
    break;
    }
```

Um nun das Ganze mit unserer eigenen Fernbedienung zum laufen zu bringen, müssen wir uns nur noch über die serielle Schnittstelle als erstes (Funktioniert auch mit dem Code aus Teil 2) die Fernbedienungscodes von 5 Tasten ausgeben lassen, die wir zur Steuerung der Laterne verwenden wollen:



Wir laden folgenden Code auf unseren Arduino hoch, nachdem wir die Zahlen in der Case Anweisung entsprechend eingetragen haben:

```
#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 = 300;
  randomSeed(analogRead(0));
  }

void SimulateFire (bool On, byte FireSq)
  {
  byte LightValue[NUMPIXELS* 3];

  if (millis()-FirelastTime>=interval)
  {
  if (On)
    {
    FireOFF = false;
    FirelastTime=millis();
    interval = 200;
    if (FireSq == 1)
      {
       for(int i=0; i<NUMPIXELS; i++)
        { // For each pixel...
        LightValue[i*3] = random(200,255); // 250
        LightValue[i*3+1] = random(30,70); // 50
        LightValue[i*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.
       interrupts();
      }
    if (FireSq == 2)
      {
       for(int i=0; i<NUMPIXELS; i++)
        { // For each pixel...
```

```
      pixels.setPixelColor(i, 250, 50,0);
      }
     noInterrupts();
     pixels.show();   // Send the updated pixel colors to the hardware.
     interrupts();
    }
  if (FireSq == 3)
    {
   interval = random(50,100);
    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 out
    byte LightsOff  = random(0,6);
    for(int i=0; i<LightsOff; i++)
      {
      byte Selected = random(NUMPIXELS);
      LightValue[Selected*3] = 0;
      LightValue[Selected*3+1] = 0;
      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.
    interrupts();
    }
  if (FireSq == 4)
    {
   interval = random(80);
    for(int i=0; i<NUMPIXELS; i++)
      { // For each pixel...
      LightValue[i*3] = 250;  //random(240,255); // 250
      LightValue[i*3+1] =  50;//random(30,60); // 50
      LightValue[i*3+2] = 0;
      }
    // Switch some lights out if Chance Hit

    byte ChanceforLightsOff  = random(0,40);
    if (ChanceforLightsOff > 35)
      {
      byte LightsOff  = random(5);
      for(int i=0; i<LightsOff; i++)
        {
        byte Selected = random(NUMPIXELS);
        LightValue[Selected*3] = 0;
```

```
        LightValue[Selected*3+1] = 0;
        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.
    interrupts();
    }
   if (FireSq == 5)
    {
   interval = random(150,200);
    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.
    interrupts();
    }
   if (FireSq == 6)
    {
   interval = random(80);
    for(int i=0; i<NUMPIXELS; i++)
     { // For each pixel...
     LightValue[i*3] = random(240,255); // 250
     LightValue[i*3+1] = random(40,60); // 50
     LightValue[i*3+2] = 0;
     }
    // Switch some lights out if Chance Hit
    byte ChanceforLightsOff  = random(0,40);
    if (ChanceforLightsOff > 35)
     {
```

```
       byte LightsOff  = random(5);
       for(int i=0; i<LightsOff; i++)
         {
         byte Selected = random(NUMPIXELS);
         LightValue[Selected*3] = 0;
         LightValue[Selected*3+1] = 0;
         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.
     interrupts();
     }
   }
  else
    {
    if (!(FireOFF))
      {
      pixels.clear();
      noInterrupts();
      pixels.show();   // Send the updated pixel colors to the hardware.
      interrupts();
      FireOFF = true;
      }
    }
  }
 }

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

void IRCommandProcessor (long IrCommand)
  {
   if (IRCode == OLDIRCode) { TimerlastTime=millis();}            //Some stuff about
debouncing IR Remote
   if (millis()-TimerlastTime >=300) { OLDIRCode = 0 ;}           //Some stuff about
debouncing IR Remote
```

```
  if ((IRCode < -1) & (IRCode != OLDIRCode) & (IRCode > -600000000) & (IRCode
< -500000000))  // Valid IR Signal Received
    {
    OLDIRCode = IRCode;                              //Some stuff about debouncing
IR Remote
    switch (IRCode)
       {
       case -522182433:       // In my case 1  on my TV - Remote
          {
          FireON = !FireON;
          FireSequence = 1;
         }
       break;
       case -522149793:       // In my case 2  on my TV - Remote
          {
          FireON = !FireON;
          FireSequence = 2;
         }
       break;
       case -522166113:       // In my case 3  on my TV - Remote
          {
          FireON = !FireON;
          FireSequence = 3;
         }
       break;
       case -522186513:       // In my case 4  on my TV - Remote
          {
          FireON = !FireON;
          FireSequence = 4;
         }
       break;
       case -522153873:       // In my case 5  on my TV - Remote
          {
          FireON = !FireON;
          FireSequence = 5;
         }
       break;
       case -522178353:       // In my case 6  on my TV - Remote
          {
          FireON = !FireON;
          FireSequence = 6;
         }
       break;

       default:
       Serial.println(IRCode);
       break;
       }
    }
 }
```

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

Die Funktion der Tasten hat sich etwas verändert. Es gibt jetzt keine dedizierte Taste mehr zum Ein- und Ausschalten der Laterne. Vielmehr kann jetzt jede der 6 definierten Tasten die Laterne durch eine weitere (erneute) Betätigung die Laterne wieder ausschalten., sofern zwischen den einzelnen Tastendrücken eine Pause von mindestens 1 Sekunde eingehalten wird (Taste loslassen).

Ich wünsche euch viel Spaß beim Upgrade euer Laternenfirmware.