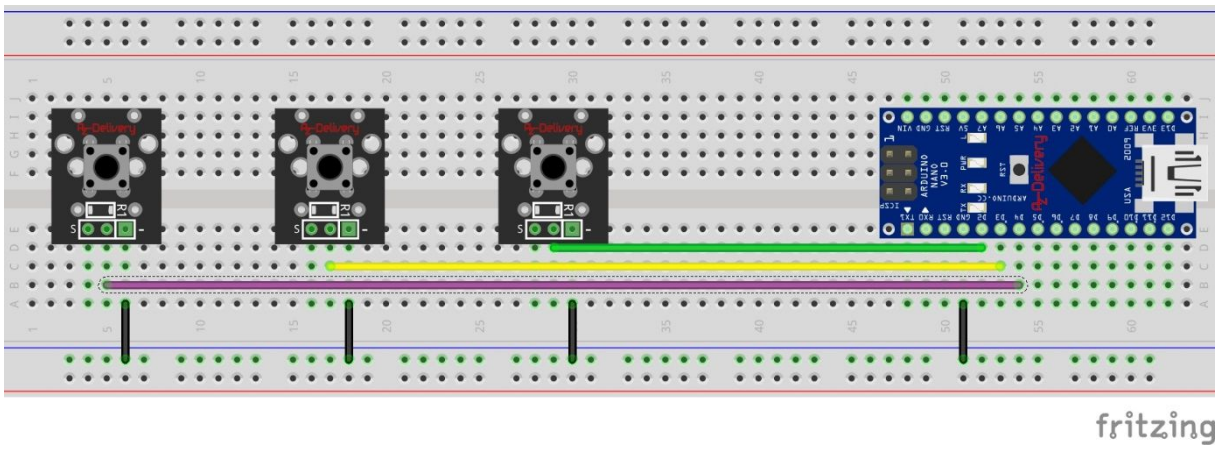
DMX Chaser Codegenerator (Teil 3 der DMX Reihe)

Willkommen zu einem weiteren Teil unserer DMX Controller Reihe. In unserem heutigen Teil möchte ich eine Erweiterung des zweiten Teils der Reihe vorstellen. Im zweiten Teil haben wir einen DMX Chaser für das Lied „The Final Countdown“ von Europe programmiert. Wer sich den Chaser-Code angesehen hat, wird festgestellt haben, dass dieser einen hohen Aufwand beim programmieren verursacht, da jede einzelne Sequenz und jede einzelne Pause dazwischen exakt auf das Lied abgestimmt bzw. angepasst werden muss. Das ist von Hand und sehr aufwendig und mühsam. Um Ihnen das Programmieren von weiteren Chasern zu erleichtern programmieren wir eine „Programmierhilfe“. Das bedeutet, wir schreiben ein Programm das wiederum selbst auf der seriellen Schnittstelle DMX bezogenen Arduino Quellcode ausgibt, der direkt wieder kompiliert werden kann. Mithilfe unseres heutigen Programmes können beliebige Chaser für beliebige (eigene) Lieder in relativ kurzer Zeit erstellt werden. Bevor wir aber loslegen unseren eigenen Chaser zu programmieren, schauen wir uns die Funktion mal etwas genauer an.

Wir brauchen nicht viel an Hardware für das Projekt:

Anzahl	Bauteil
1	Arduino Nano
3	Taster

Der notwendige Aufbau kann der folgendem Verdrahtungsplan entnommen werden:



Wir laden folgenden Code auf den Arduino Nano hoch:

```
#define SEQ_Pin 2
#define STARTSEQ_Pin 3
#define ENDSEQ_Pin 4

long DMX_Pattern_Number = 1;
unsigned long ElapsedMillis = 0;
bool ReadPin = true;
bool EndReached = false;

void ISR_SEQ()
{
noInterrupts();
delay(200);
bool PinState = digitalRead(SEQ_Pin);
if (!(PinState))
 {
 DMX_Pattern_Number = 1;
 Serial.println("//Start Counting..");
 ElapsedMillis = millis();
 EndReached = false;
 }
interrupts();
}

void ISR_STARTSEQ()
{
noInterrupts();
bool PinState = digitalRead(STARTSEQ_Pin);
if (!(PinState)and !(EndReached) )
 {
 Serial.print("delay(");
 Serial.print (millis()- ElapsedMillis);
 Serial.println(");");
 Serial.print("DmxPattern(");
 Serial.print(DMX_Pattern_Number);
 Serial.println(");");
 DMX_Pattern_Number++;
 ElapsedMillis = millis();
 }
delay(200);
interrupts();
}

void PrintCodeHeader ()
{
Serial.println(F("// DMX Chaser to Song ' ... ' . by Tobias Kuch 2020 V1.0"));
Serial.println(F("#define Start_Switch_Pin 5"));
Serial.println(F("// Define DMX Actor Primary Address"));
Serial.println(F("#define DMX_Actor_Address 1"));
Serial.println(F(" "));
Serial.println(F("// Define DMX Device Channel Functions"));
Serial.println(F("#define DMX_RED_Ch 0 + DMX_Actor_Address"));
```

```
Serial.println(F("#define DMX_GREEN_Ch 1 + DMX_Actor_Address"));
Serial.println(F("#define DMX_BLUE_Ch 2 + DMX_Actor_Address"));
Serial.println(F("#define DMX_MACRO_Ch 3 + DMX_Actor_Address"));
Serial.println(F("#define DMX_STROBE_Ch 4 + DMX_Actor_Address"));
Serial.println(F("#define DMX_MODE_Ch 5 + DMX_Actor_Address"));
Serial.println(F("#define DMX_MASTERDIM_Ch 6 + DMX_Actor_Address"));
Serial.println(F(" "));
Serial.println(F("#include <DmxSimple.h>"));
Serial.println(F(" "));
Serial.println(F("byte DMX_MASTERDIM_VAL = 255;"));
Serial.println(F("bool ReadPin;"));
Serial.println(F(" "));
Serial.println(F("void setup() {"));
Serial.println(F("  pinMode(Start_Switch_Pin,INPUT_PULLUP);"));
Serial.println(F("  DmxSimple.usePin(3);"));
Serial.println(F("  DmxSimple.maxChannel(512);"));
Serial.println(F("  do"));
Serial.println(F("    {"));
Serial.println(F("    ReadPin = digitalRead (Start_Switch_Pin);"));
Serial.println(F("    } while (!(ReadPin));"));
Serial.println(F("  } "));
Serial.println(F(" "));
Serial.println(F("void DmxPattern (long DMX_Pattern_Number)"));
Serial.println(F("{"));
Serial.println(F("DmxSimple.write(DMX_RED_Ch,dmxval(DMX_RED_Ch,DMX_Pattern_Number));")
);
Serial.println(F("DmxSimple.write(DMX_GREEN_Ch,dmxval(DMX_GREEN_Ch,DMX_Pattern_Numb
er));"));
Serial.println(F("DmxSimple.write(DMX_BLUE_Ch,dmxval(DMX_BLUE_Ch,DMX_Pattern_Number));
"));
Serial.println(F("DmxSimple.write(DMX_MACRO_Ch,
dmxval(DMX_MACRO_Ch,DMX_Pattern_Number));"));
Serial.println(F("DmxSimple.write(DMX_STROBE_Ch,dmxval(DMX_STROBE_Ch,DMX_Pattern_Num
ber));"));
Serial.println(F("DmxSimple.write(DMX_MODE_Ch,
dmxval(DMX_MODE_Ch,DMX_Pattern_Number));"));
Serial.println(F("DmxSimple.write(DMX_MASTERDIM_Ch,
dmxval(DMX_MASTERDIM_Ch,DMX_Pattern_Number));"));
Serial.println(F("}"));
Serial.println(F(" "));
Serial.println(F("void loop() {"));
}

void PrintCodeFooter (long DMX_Pt_Nbr )
{
Serial.println(F("do {} while (true);"));
Serial.println(F("}"));
Serial.println(F(" "));
Serial.println(F("//^^^^^^^^^^^^^^^^^^^  ROT GRN BLAU MAC STR MOD Master DIM
^^^^^^^^"));
Serial.print(F("byte DmxData["));
Serial.print (DMX_Pt_Nbr);
Serial.println(F("][7] = {"));
```

```
for (long z = 1; z <= DMX_Pt_Nbr;z++)
{
Serial.println(F("                 { 0, 0, 0, 0, 0, 0,DMX_MASTERDIM_VAL},"));
}
Serial.println(F("                };"));
Serial.println(F("//-------------------  Common Example Patterns           ---------"));
Serial.println(F("//                  {255, 0, 0, 0, 0, 0,DMX_MASTERDIM_VAL}, // volles Rot "));
Serial.println(F("//                  { 0,255, 0, 0, 0, 0,DMX_MASTERDIM_VAL}, // volles Grün "));
Serial.println(F("//                  { 0, 0,255, 0, 0, 0,DMX_MASTERDIM_VAL}, // volles Blau "));
Serial.println(F("//                  {255,255,255, 0, 0, 0,DMX_MASTERDIM_VAL}, // volles weiss "));
Serial.println(F("//                  {255,255,255, 0,255, 0,DMX_MASTERDIM_VAL}, // schnelles
Stroboskop "));
Serial.println(F("//                  {255,255,255, 0, 0, 70,DMX_MASTERDIM_VAL}, // Weiß
Aufblenden "));
Serial.println(F("//                  {255,255,255, 0, 0, 60,DMX_MASTERDIM_VAL}, // Weiß
Abbblenden "));
Serial.println(F("//                  {255,255,255, 20, 0, 60,DMX_MASTERDIM_VAL}, // Weiß
langsames blinken "));

Serial.println(F(" "));
Serial.println(F("byte dmxval(byte DMX_Ch,long DMX_Pattern_Number)"));
Serial.println(F("{"));
Serial.println(F("  byte u =  DmxData[DMX_Pattern_Number-1][DMX_Ch-1];"));
Serial.println(F("  return u;"));
Serial.println(F("}"));
}



void setup() {
  //Initalisierung
  Serial.begin(115200);
  pinMode(SEQ_Pin,INPUT_PULLUP);
  pinMode(STARTSEQ_Pin,INPUT_PULLUP);
  pinMode(ENDSEQ_Pin,INPUT_PULLUP);
  noInterrupts();
  attachInterrupt(0, ISR_SEQ, CHANGE);
  attachInterrupt(1, ISR_STARTSEQ, CHANGE);
  PrintCodeHeader ();
  interrupts();
  ElapsedMillis = millis();
}

void loop() {
 // put your main code here, to run repeatedly:
 ReadPin = digitalRead (ENDSEQ_Pin);
 if (!(ReadPin) and !(EndReached))
  {
   PrintCodeFooter (DMX_Pattern_Number-1);
   delay(100);   //Debouncing
   EndReached = true;
  }
  delay(100);
```

```
}
```

Zur Funktion ein paar Grundlagen: Für ein funktionsfähiges Programm werden der Init Teil, in der die Variablen deklariert, der Setup teil, der die ersten Initialisierungen ausführt und der Loopt Teil, der die Hauptaufgaben des Programms ausführt benötigt. Zuletzt wird der Loop Teil abgeschlossen und es folgen ggf. noch ein paar Hilfsfunktionen. Genau nach dieser Reihenfolge arbeiten die Taster:

Beim Druck auf den Taster an Port 3 wird der Initialisierungs- und der Setup Teil unseres Chasers auf der Seriellen Schnittstelle ausgegeben. Dabei wird folgender Codeblock auf der seriellen Schnittstelle ausgegeben und ein interner Timer gestartet:

```
// DMX Chaser to Song ' ... ' . by Tobias Kuch 2020 V1.0
#define Start_Switch_Pin 5
// Define DMX Actor Primary Address
#define DMX_Actor_Address 1

// Define DMX Device Channel Functions
#define DMX_RED_Ch 0 + DMX_Actor_Address
#define DMX_GREEN_Ch 1 + DMX_Actor_Address
#define DMX_BLUE_Ch 2 + DMX_Actor_Address
#define DMX_MACRO_Ch 3 + DMX_Actor_Address
#define DMX_STROBE_Ch 4 + DMX_Actor_Address
#define DMX_MODE_Ch 5 + DMX_Actor_Address
#define DMX_MASTERDIM_Ch 6 + DMX_Actor_Address

#include <DmxSimple.h>

byte DMX_MASTERDIM_VAL = 255;
bool ReadPin;

void setup() {
 pinMode(Start_Switch_Pin,INPUT_PULLUP);
 DmxSimple.usePin(3);
 DmxSimple.maxChannel(512);
 do
  {
  ReadPin = digitalRead (Start_Switch_Pin);
  } while (!(ReadPin));
 }

void DmxPattern (long DMX_Pattern_Number)
{
DmxSimple.write(DMX_RED_Ch,dmxval(DMX_RED_Ch,DMX_Pattern_Number));
DmxSimple.write(DMX_GREEN_Ch,dmxval(DMX_GREEN_Ch,DMX_Pattern_Number));
DmxSimple.write(DMX_BLUE_Ch,dmxval(DMX_BLUE_Ch,DMX_Pattern_Number));
DmxSimple.write(DMX_MACRO_Ch, dmxval(DMX_MACRO_Ch,DMX_Pattern_Number));
DmxSimple.write(DMX_STROBE_Ch,dmxval(DMX_STROBE_Ch,DMX_Pattern_Number));
DmxSimple.write(DMX_MODE_Ch, dmxval(DMX_MODE_Ch,DMX_Pattern_Number));
DmxSimple.write(DMX_MASTERDIM_Ch, dmxval(DMX_MASTERDIM_Ch,DMX_Pattern_Number));
```

```
}

void loop() {
```

Damit haben wir die Grundlagen für den nun eigentlichen Hauptteil des Programms gelegt.

Durch Druck auf den Taster an Port 2 werden jeweils die Zeitlichen Abstände zwischen den Tastendrücken gemessen und in folgende Befehlssequenz übersetzt:

```
delay(Zeit);
DmxPattern(Sequenz);
```

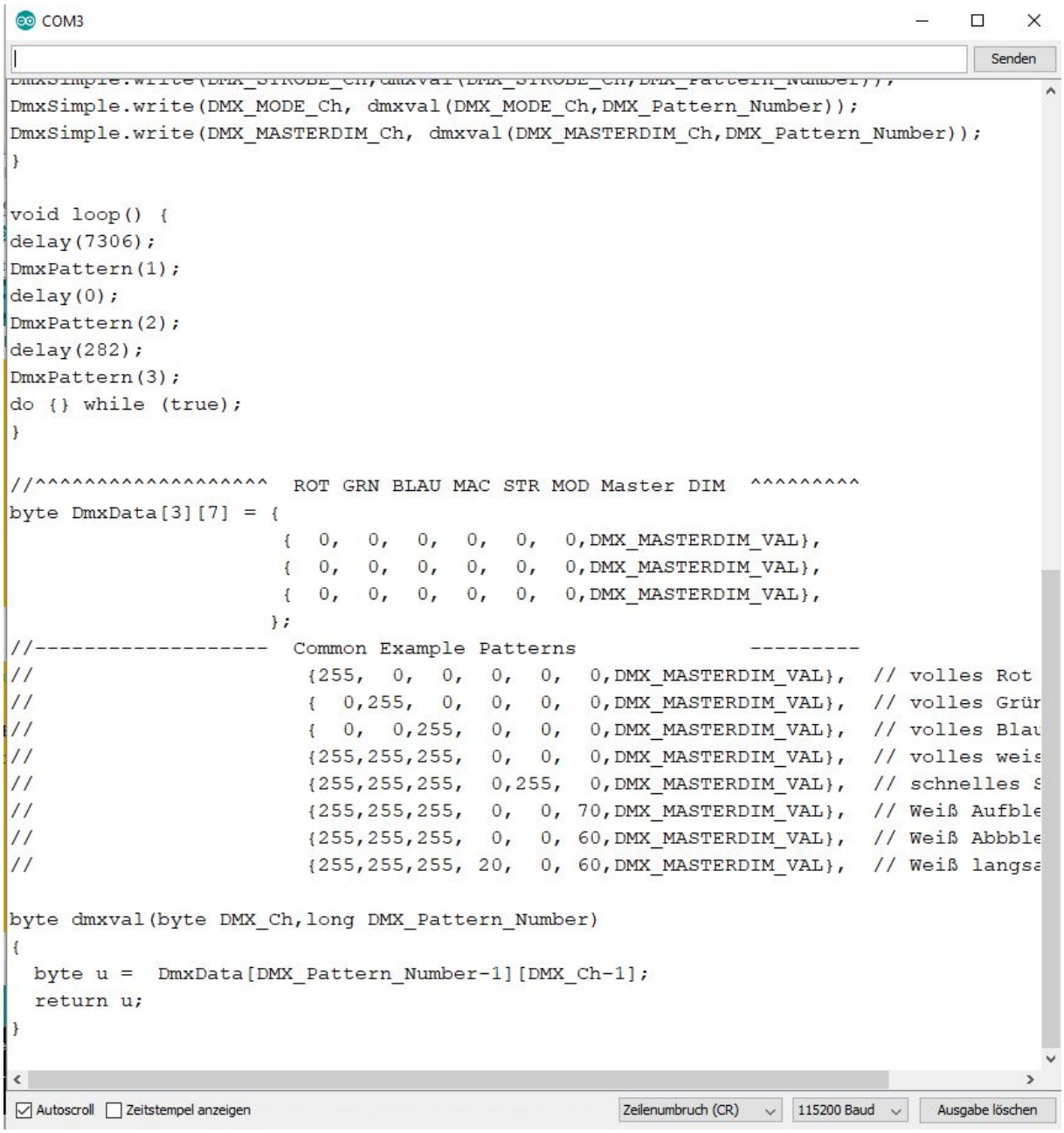
Die Chasergeneration erfolgt also durch rhythmischen drücken des Tasters an Port 2 im Takt bzw Bassschlag der Musik. Wenn der Chaser abgeschlossen werden soll, **drücken Sie den Taster an Port 4**. Damit wird der Sketch fertiggstellt. Es wird folgender Endteil an den Sketch angehängt:

```
do {} while (true);
}

//^^^^^^^^^^^^^^^^^^^  ROT GRN BLAU MAC STR MOD Master DIM  ^^^^^^^^^
byte DmxData[3][7] = {
                      {  0,  0,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                      {  0,  0,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                      {  0,  0,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                     };
//------------------  Common Example Patterns            ---------
//                     {255,  0,  0,  0,  0,  0,DMX_MASTERDIM_VAL},  // volles Rot
//                     {  0,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},  // volles Grün
//                     {  0,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},  // volles Blau
//                     {255,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},  // volles weiss
//                         {255,255,255,  0,255,  0,DMX_MASTERDIM_VAL},  // schnelles
Stroboskop
//                   {255,255,255,  0,  0, 70,DMX_MASTERDIM_VAL},  // Weiß Aufblenden
//                   {255,255,255,  0,  0, 60,DMX_MASTERDIM_VAL},  // Weiß Abbblenden
//                   {255,255,255, 20,  0, 60,DMX_MASTERDIM_VAL},  // Weiß langsames
blinken

byte dmxval(byte DMX_Ch,long DMX_Pattern_Number)
{
  byte u =  DmxData[DMX_Pattern_Number-1][DMX_Ch-1];
  return u;
}
```

Auf dem seriellen Monitor sehen wir folgende Ausgabe:



Zum Abschluss und vor dem Hochladen des neuen Sketches auf einen Arduino müssen Sie das DmxData Array mit eigenen Lichtwerten befüllen. Dabei wird von folgender Belegung der DMX Kanäle ausgegangen: Kanal Rot, Kanal Grün, Kanal Blau, Speed, Effect, Strobe, Masterdimmer. Für die Verkabelung und den Betrieb des Caser Programms gilt unverändert 1:1 der vorgängerteil (Teil 2) dieser Reihe.

Ich wünsche Ihnen viel Spaß beim Nachbau.