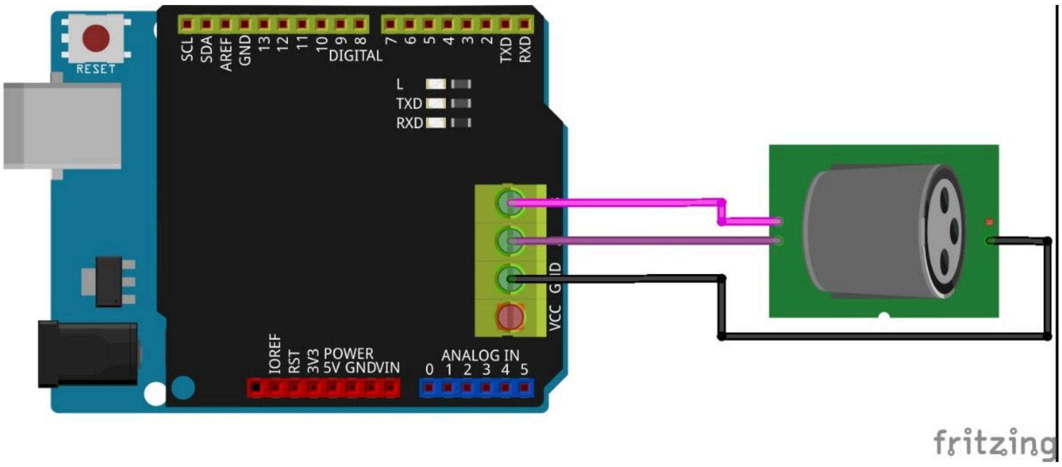
Ein DMX Chaser zum Song „The Final Countdown von Europe“

Hallo und willkommen zu dem zweiten Teil der Reihe DMX Controller. Im heutigen Teil programmieren wir mithilfe unseres Arduinos einen DMX Chaser oder auch eine DMX Sequenz für den bekannten Song „The Final Countdown“ von Europe. Damit bekommen Sie einen guten Eindruck darüber was alles mit DMX machbar ist und bis zum welchem Grad Automation im Bereich der Lichttechnik möglich ist. Der technische Aufbau gleicht hierbei dem ersten Teil der Reihe und muss nicht verändert werden.


RESET
SCL SDA AREF GND 13 12 11 10 9 8 DIGITAL 7 6 5 4 3 2 TXD RXD
L
TXD
RXD
VCC GND
IOREF RST 3V3 POWER 5V GND VIN
ANALOG IN 0 1 2 3 4 5
fritzing

Um die generelle Funktionsweise unseres Chasers zu verstehen, gehen wir den Programmablauf durch. Zunächst definieren wir für unseren Chaser die DMX Adresse des RGB Strahlers über die fest definierte Konstante DMX_Actor_Address in Zeile:

```
#define DMX_Actor_Address 1
```

an. Diese Adresse gibt die Basisadresse (Hauptadresse) des RGB Strahlers an. Es wird im Code davon ausgegangen, dass die einzelnen Farben (rot/grün/blau) separat über einen Kanal steuerbar sind.

Für die Kanalbelegung passen wir nun die Kanaldefinitionen DMX_RED_Ch 0, DMX_GREEN_Ch 1 ,DMX_BLUE_Ch ,DMX_MACRO_Ch,DMX_STROBE_Ch, DMX_MODE_Ch und DMX_MASTERDIM_Ch an:

```
// 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
```

Nach den Anpassungen kann der Code auf den Arduino hochgeladen werden:

```
// DMX Chaser to Song ' Final Countdown von Europe ' . by Tobias Kuch 2020 V1.0
#define Start_Switch_Pin 10
// Define DMX Actor Primary Address
nein

// 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);
  pinMode(12,OUTPUT);
  digitalWrite(12,LOW);
  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() {
DmxPattern(1);
delay(3138);
DmxPattern(2);
delay(1824);
DmxPattern(3);
```

```
delay(1768);
DmxPattern(4);
delay(1995);
DmxPattern(5);
delay(119);
DmxPattern(6);
delay(1951);
DmxPattern(7);
delay(2040);
DmxPattern(8);
delay(2851);
DmxPattern(9);
delay(1970);
DmxPattern(10);
delay(2162);
DmxPattern(11);
delay(135);
DmxPattern(12);
delay(1880);
DmxPattern(13);
delay(1986);
DmxPattern(14);
delay(2007);
DmxPattern(15);
delay(2131);
DmxPattern(16);
delay(740);
DmxPattern(17);
delay(441);
DmxPattern(18);
delay(172);
DmxPattern(19);
delay(1029);
DmxPattern(20);
delay(1530);
DmxPattern(21);
delay(2161);
DmxPattern(22);
delay(185);
DmxPattern(23);
delay(1852);
DmxPattern(24);
delay(1098);
DmxPattern(25);
delay(2926);
DmxPattern(26);
delay(2110);
DmxPattern(27);
delay(497);
DmxPattern(28);
delay(152);
DmxPattern(29);
delay(1437);
```

```
DmxPattern(30);
delay(494);
DmxPattern(31);
delay(550);
DmxPattern(32);
delay(947);
DmxPattern(33);
delay(891);
DmxPattern(34);
delay(592);
DmxPattern(35);
delay(561);
DmxPattern(36);
delay(489);
DmxPattern(37);
delay(578);
DmxPattern(40);
delay(1125);
DmxPattern(41);
delay(442);
DmxPattern(42);
delay(388);
DmxPattern(43);
delay(148);
DmxPattern(44);
delay(2009);
DmxPattern(45);
delay(17308);
DmxPattern(46);
delay(154);
DmxPattern(47);
delay(883);
DmxPattern(48);
delay(533);
DmxPattern(49);
delay(525);
DmxPattern(50);
delay(134);
DmxPattern(51);
delay(353);
DmxPattern(52);
delay(120);
DmxPattern(53);
delay(361);
DmxPattern(54);
delay(1271);
DmxPattern(55);
delay(2767);
DmxPattern(56);
delay(2085);
DmxPattern(57);
delay(1533);
DmxPattern(58);
```

```
delay(550);
DmxPattern(59);
delay(497);
DmxPattern(60);
delay(121);
DmxPattern(61);
delay(360);
DmxPattern(62);
delay(498);
DmxPattern(64);
delay(502);
DmxPattern(65);
delay(2257);
DmxPattern(66);
delay(4874);
DmxPattern(67);
delay(4096);
DmxPattern(68);
delay(4146);
DmxPattern(69);
delay(4494);
DmxPattern(70);
delay(3515);
DmxPattern(71);
delay(5263);
DmxPattern(72);
delay(2459);
DmxPattern(73);
delay(124);
DmxPattern(74);
delay(417);
DmxPattern(75);
delay(1580);
DmxPattern(76);
delay(510);
DmxPattern(77);
delay(474);
DmxPattern(79);
delay(487);
DmxPattern(81);
delay(483);
DmxPattern(82);
delay(2288);
DmxPattern(84);
delay(489);
DmxPattern(85);
delay(120);
DmxPattern(86);
delay(1678);
DmxPattern(87);
delay(491);
DmxPattern(88);
delay(1905);
```

```
DmxPattern(89);
delay(362);
DmxPattern(90);
delay(384);
DmxPattern(91);
delay(460);
DmxPattern(92);
delay(4147);
DmxPattern(93);
delay(459);
DmxPattern(94);
delay(0);
DmxPattern(95);
delay(129);
DmxPattern(96);
delay(347);
DmxPattern(97);
delay(1345);
DmxPattern(98);
delay(567);
DmxPattern(99);
delay(551);
DmxPattern(100);
delay(24599);
DmxPattern(101);
delay(93);
DmxPattern(102);
delay(6742);
DmxPattern(103);
delay(0);
DmxPattern(104);
delay(421);
DmxPattern(105);
delay(425);
DmxPattern(106);
delay(424);
DmxPattern(107);
delay(482);
DmxPattern(108);
delay(6678);
DmxPattern(109);
delay(433);
DmxPattern(111);
delay(481);
DmxPattern(112);
delay(463);
DmxPattern(113);
delay(6714);
DmxPattern(114);
delay(490);
DmxPattern(115);
delay(484);
DmxPattern(116);
```

```
delay(512);
DmxPattern(118);
delay(3077);
DmxPattern(119);
do {} while (true);
}

//^^^^^^^^^^^^^^^^^^^  ROT GRN BLAU MAC STR MOD Master DIM  ^^^^^^^^^
byte DmxData[119][7] = {
                {255,  0,  0,  0,255,  0,DMX_MASTERDIM_VAL}, // DMX - Pattern 1
                {255,  0,  0,  0,  0, 70,DMX_MASTERDIM_VAL},
                {255,  0,255,  0,  0, 70,DMX_MASTERDIM_VAL},
                {  0,255,255,  0,  0, 70,DMX_MASTERDIM_VAL},
                {  0,255,  0,  0,  0, 70,DMX_MASTERDIM_VAL},
                {  0,  0,255,  0,  0, 70,DMX_MASTERDIM_VAL},
                {  0,255,  0,  0,  0, 70,DMX_MASTERDIM_VAL},
                {255,255,  0,  0,  0, 70,DMX_MASTERDIM_VAL},
                {255,  0,255,  0,  0, 70,DMX_MASTERDIM_VAL},
                {255,  0,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {255,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {255,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {255,  0,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {255,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {255,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,255,  0,  0,  0,140,DMX_MASTERDIM_VAL},
                {  0,  0,255,  0,  0,140,DMX_MASTERDIM_VAL},
                {255,  0,  0,  0,  0,140,DMX_MASTERDIM_VAL},
                {255,  0,255,  0,  0,140,DMX_MASTERDIM_VAL},
                {  0,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {255,  0,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {255,255,255,  0,  0,100,DMX_MASTERDIM_VAL},
                {255,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {255,  0,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {255,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {255,  0,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {255,  0,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {255,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,255,255,  0,255,  0,DMX_MASTERDIM_VAL},
{  0,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{  0,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
{  0,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
{  0,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{  0,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
{  0,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{  0,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
{  0,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{  0,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,  0,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{  0,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{  0,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
{  0,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,  0,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
{  0,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{  0,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,  0,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
{  0,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
{  0,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,  0,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{150,  0,150,  0,  0,  0,DMX_MASTERDIM_VAL},
{ 50,  0, 50,  0,  0,  0,DMX_MASTERDIM_VAL},
{ 25,  0, 25,  0,  0,  0,DMX_MASTERDIM_VAL},
{ 50,  0, 50,  0,  0,  0,DMX_MASTERDIM_VAL},
{150,  0,150,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,  0,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,  0,  0,255,  0,  0,DMX_MASTERDIM_VAL},
{255,  0,  0,  0,  0, 60,DMX_MASTERDIM_VAL},
{255,255,255,  0,255, 0 ,DMX_MASTERDIM_VAL},
{180,180,180,  0,  0,  0,DMX_MASTERDIM_VAL},
{150,150,150,  0,  0, 60,DMX_MASTERDIM_VAL},
{255,  0,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
{  0,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
{  0,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{  0,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,255,  0,  0,255,  0,DMX_MASTERDIM_VAL},
{  0,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
{255,255,  0,  0,0,  0,DMX_MASTERDIM_VAL},
{255,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {255,255,  0,  0,0,  0,DMX_MASTERDIM_VAL},
                {255,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {255,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,255,0,  0,  0,  70,DMX_MASTERDIM_VAL},
                {255,  0,  0,  0,0,  0,DMX_MASTERDIM_VAL},
                {  0,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,  0,150,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,  0, 50,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {255,255,255,  0,255,  0,DMX_MASTERDIM_VAL},
                {  0,  0,255,  0,255,  0,DMX_MASTERDIM_VAL},
                {255,  0,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {  0,255,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {255,  0,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {255,255,255,  0,255,  0,DMX_MASTERDIM_VAL},
                {255,  0,  0,  0,  0,  0,DMX_MASTERDIM_VAL},
                {255,255,255,  0,  0,  0,DMX_MASTERDIM_VAL},
                {255,255,255,  0,  0, 60,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 Abblenden
//                  {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;
}
```

Ich wünsche viel Spaß beim Nachbauen und bis zum nächsten mal.