FlightStrip

Version 0.6

Handbuch

von Yannik Wertz

Inhatsverzeichnis

1. Was ist FlightStrip

FlightStrip ist eine Arduino Library, die die Anwendung von RGB LED Strips vom Typ WS2812 vereinfacht.
Mit der Library ist es selbst für wenig Erfahrene ein Leichtes eine umfangreiche Beleuchtung in wenigen Schritten individuell zu erstellen. Dazu sei gesagt, dass diese LED Strips eine gewisse Komplexität haben und daher Grundkenntnisse im Umgang mit Arduino, Elektronik und Programmieren von Vorteil sind.
FlightStrip dient nicht zur reinen Ansteuerung der LED Strips. Dies geschieht z.B. durch die Neopixel Library von Adafruit.

2. Anwendungsbeispiel

Im Folgenden wird die Vorgehensweise anhand eines Fallbeispiels erläutert.
Dieses Fallbeispiel wird als Beispiel mit der Library mitgeliefert.
In der Praxis sind meist mehr Bereiche, also FlightStrip Objekte, nötig nur dies würde den Rahmen dieser Dokumetation sprengen. Ein Praxisbeispiel ist ebenfalls der Library mitgeliefert (FlightStrip Advanced Example with RC).

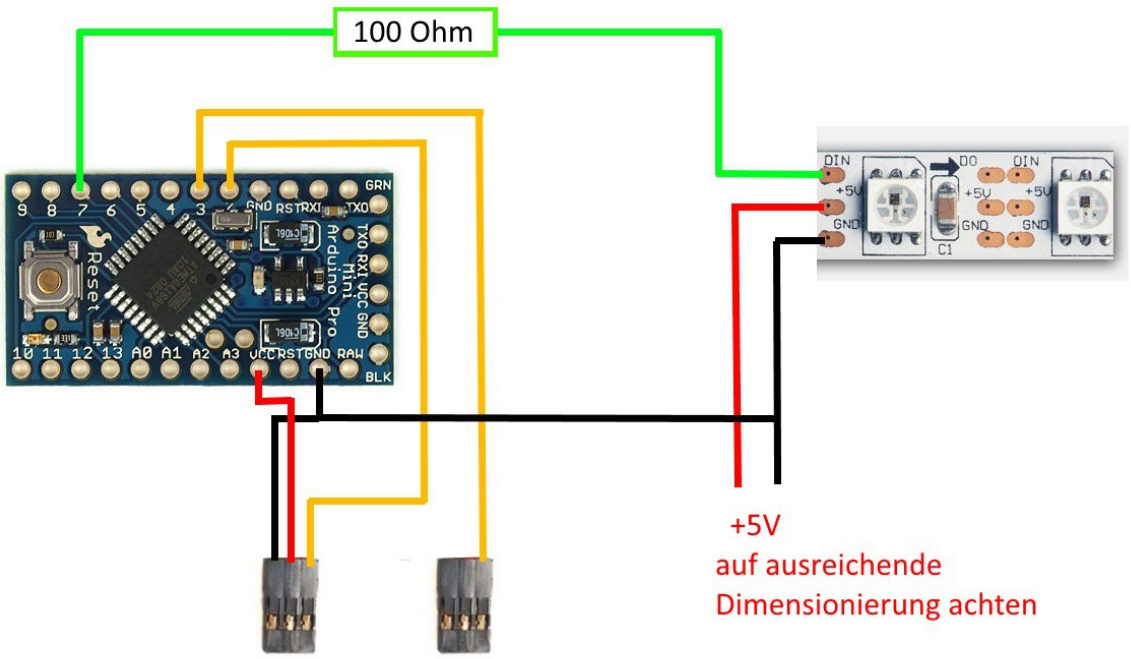
2.1 Hardware

Das Flugzeug wird mit folgenden LED Strips ausgestattet :

- Fläche, 30 LEDs , Nr. 0 bis 29
- Rumpf, 20 LEDs , Nr. 30 bis 49
- → Gesamt : 50 LEDs

Die Strips werden in der oben angegebenen Reihenfolge verkabelt.

Der Anfang des ersten Strips wird nun folgendermaßen mit dem Arduino verkabelt :
(Achtung, im Moment wird nur ein RC Kanal an PIN 2 benutzt)



2.2 Software

Als erstes müssen die benötigten Libraries importiert werden

```
/*############################################################
##                  Libraries                                ##
############################################################*/

#include <Adafruit_NeoPixel.h>
#include "FlightStrip.h"
```

Anschließend müssen die Hardwarevorgaben in der Software eingetragen werden :

```
/*############################################################
##                  Defines                                  ##
############################################################*/

#define STRIP_PIN   7                   //Anschlusspin
#define LED_ANZAHL  50                  //Gesamt LED Anzahl
#define FLAECHE_BOT 0                   //Erste LED des Flächen Strips
#define FLAECHE_UP  29                  //Letzte LED des Flächen Strips
#define RUMPF_BOT   30                  //Erste LED des Rumpf Strips
#define RUMPF_UP    49                  //Letzte LED des Rumpf Strips
#define MODE_ANZAHL 2                   //Anzahl der Modes
#define CHANGE_TIME 10000               //Dauer in Millisekunden eines Modus
```

Nun werden alle benötigten Objekte erzeugt

```
/*############################################################
##                  Objects                                  ##
############################################################*/

//Zuerste wird das Strip Objekt erstellt
Adafruit_NeoPixel strip = Adafruit_NeoPixel(LED_ANZAHL, STRIP_PIN, NEO_GRB + NEO_KHZ800);

//Anschließend die beiden Teile des Strips mit den oben defineirten Grenzen
FlightStrip flaeche = FlightStrip(strip, FLAECHE_BOT, FLAECHE_UP);
FlightStrip rumpf = FlightStrip(strip, RUMPF_BOT, RUMPF_UP);
```

Als nächstes werden noch einige Variablen definiert, die das Programm später braucht

```
/*############################################################
##                  Variablen                                ##
############################################################*/

//Array von Standardfarben wird angelegt. Dies erleichtert später die Anwendung
uint32_t farben[] =  {strip.Color(255,0,0),         //farben[0] = Rot
                      strip.Color(0,255,0),         //farben[1] = Grün
                      strip.Color(0,0,255),         //farben[2] = Blau
                      strip.Color(255,0,255),       //farben[3] = Magenta
                      strip.Color(0,255,255),       //farben[4] = Hellblau
                      strip.Color(255,255,255),     //farben[5] = Weiß
                      strip.Color(255,255,0),       //farben[6] = Gelb
                      strip.Color(255,140,0),       //farben[7] = Orange
                      strip.Color(0,127,127)};      //farben[8] = Türkis

uint32_t lastShow = 0;   //Diese Variable wird zum aktualisieren der Strips benötigt

uint32_t aendern = 0;    //Zähler um alle X Sekunden automatisch in den nächsten Modus zu schalten
uint8_t modus = 0;       //Aktueller Modus
```

Nun sind alle Hardwaredefinitionen und Grundlagen eingetragen. Als nächstes müssen nun die Effekte programmiert und eingestellt werden.

```
/*############################################################
##                    Setup                                 ##
############################################################*/

void setup()
{
  strip.begin();                                    //Strip(s) initialisieren
  strip.show();

  //Strips einlernen
  flaeche.learnMode(0, CHASE_TRIPLE_COLOR);         //Fläche Mode 0 → Chase Triple Color
  rumpf.learnMode(0, DOUBLEFLASH);                  //Rumpf Mode 0 → Doubleflash

  flaeche.learnMode(1, FILL_AGAINST);               //Fläche Mode 1 → Fill Against
  rumpf.learnMode(1, THUNDERSTORM);                 //Rumpf Mode 1 → Thunderstorm

  //Chase Triple Color bekommt die Farben Türkis, Magenta und Grün zugeordnet
  flaeche.updateParameter(CHASE_TRIPLE_COLOR, farben[8], farben[3], farben[1], 0);
}
```

Das war auch schon nahezu alles was eingestellt werden muss/kann.
Jetzt folgt die Loop Schleife die vom Arduino immer wieder durchlaufen wird.

```
/*############################################################
##                    Loop                                  ##
############################################################*/

void loop()
{
  flaeche.update();                                 //Beide FlightStrip Objekte werden aktualisiert
  rumpf.update();
  showStrips();                                     //Die aktuellen Strip Daten werden an die Strips übertragen

  if((millis() - aendern) > CHANGE_TIME)            //Ist die in CHANGE_TIME eingetragen Zeit vergangen wird
{                                                   //in den nächsten Mode geschaltet
    aendern = millis();
    if(modus < (MODE_ANZAHL - 1))
      modus++;
    else
      modus = 0;

    flaeche.setMode(modus);                         //Der Modus ist nun festgelegt, muss aber noch allen
    rumpf.setMode(modus);                           //FlightStrip Objekten mitgeteilt werden
  }
}
```

```
/*############################################################
##                    Functions                             ##
############################################################*/

//Aktualisiert alle Strips und speicher Zeit (hier muss der User nichts verändern)
void showStrips()
{
  if((micros() - lastShow) >= 25000)
  {
    strip.show();
    lastShow = micros();
  }
}
```

To be continued...

3. Effekte und Parameter

3.1 Funktionen

Im Folgenden werden nun alle benötigten Funktionen der Flightstrip Library erklärt und anhand eines Beispiels verdeutlicht.

3.1.1 FlightStrip(Adafruit_NeoPixel &stripToUse, uint16_t startLed, uint16_t endLed);

Konstruktor, erzeugt die FlightStrip Objekte mit den übergebenen Parametern.

Parameter :
Adafruit_neoPixel &stripToUse : Strip Objekt
uint16_t startLed : Nummer der ersten LED des Strips
uint16_t startLed : Nummer der letzten LED des Strips

Beispiel :

```
#define STRIP_PIN   7                 //Anschlusspin
#define LED_ANZAHL  30                //Gesamt LED Anzahl
#define FLAECHE_BOT 0                 //Erste LED des Flächen Strips
#define FLAECHE_UP  19                //Letzte LED des Flächen Strips
#define RUMPF_BOT   20                //Erste LED des Rumpf Strips
#define RUMPF_UP    29                //Letzte LED des Rumpf Strips

//Globales Strip Objekt
Adafruit_NeoPixel strip = Adafruit_NeoPixel(LED_ANZAHL, STRIP_PIN, NEO_GRB + NEO_KHZ800);

//Erstellen des FlightStrip Objekts für die Fläche mit oben desinierten Grenzen
FlightStrip flaeche = FlightStrip(strip, FLAECHE_BOT, FLAECHE_UP);
//Erstellen des FlightStrip Objekts für den Rumpf mit oben definierten Grenzen
FlightStrip rumpf = FlightStrip(strip, RUMPF_BOT, RUMPF_UP);
```

3.1.2 void learnMode(uint8_t modeNumber, uint8_t effectNumber);

Dient zum Einlernen, welcher Effekt welchem Modus zugeordnet wird.

Parameter :
uint8_t modeNumber : Nummer des Modus dem ein Effekt zugeordnet werden soll
uint8_t effectNumber : Name des Effekts (siehe Kapitel 3.2)

Beispiel :

```
flaeche.learnMode(0, KNIGHTRIDER);          //Fläche Mode 0 → KNIGHTRIDER
rumpf.learnMode(0, CONST_COLOR);            //Rumpf Mode 0 → CONST_COLOR

flaeche.learnMode(1, THUNDERSTORM);         //Fläche Mode 1 → THUNDERSTORM
rumpf.learnMode(1, FILL_TRIPLE_COLOR);      //Rumpf Mode 1 → FILL_TRIPLE_COLOR
```

3.1.3 void updateParameter(uint8_t effectNumber, uint32_t color1, uint32_t color2, uint32_t color3, uint16_t interval);

Anpassen der Parameter eines Effektes.
Achtung : Gilt global für einen Effekt, NICHT für einen Modus. Soll ein Effekt bei zwei Modi mit unterschiedlichen Parametern genutzt werden müssen die Parameter beim Modewechsel neu eingestellt werden.

Parameter :
uint8_t effectNumber : Name des Effekts (siehe Kapitel 3.2)
uint32_t color1 : Parameter 1 (siehe Kapitel 3.2)
uint32_t color2 : Parameter 2 (siehe Kapitel 3.2)
uint32_t color3 : Parameter 3 (siehe Kapitel 3.2)
uint32_t interval : Parameter 4 (siehe Kapitel 3.2)

Beispiel :

```
//Einstellen des Knightrider Effekts, Erklärung siehe Kapitel 3.2
flaeche.updateParameter(KNIGHTRIDER, 0, strip.Color(25,25,25), 1, 100);
```

3.1.4 void reverse();

Manche Effekte sind richtungsabhängig. Mit dem reverse Befehl kann die Richtung aller Effekte eines FlightStrip Objekts umgestellt werden.

Parameter :
-

Beispiel :

```
//Die Richtung aller Effekte für das Rumpf Objekt wird umgedreht
rumpf.reverse();
```

3.1.5 void setMode(uint8_t actMode);

Mit diesem Befehl wird der aktuelle Modus geändert.
Achtung : mit setBounds gesetzte Grenzen und der disable Befehl werden bei Moduswechsel automatisch rückgängig gemacht.

Parameter :
uint8_t actMode : Nummer des Modus, der verwendet werden soll

Beispiel :

```
if((millis() - aendern) > 10000)              //Alle 10 Sekunden wird der Modus geändert
{
   aendern = millis();
   if(modus < (MODE_ANZAHL - 1))
     modus++;
   else
     modus = 0;

   flaeche.setMode(modus);
   rumpf.setMode(modus);
 }
```

3.1.6 void update();

Dies ist der eigentliche Berechnungsprozess der Library. Hier wird das Pseudo Multitasking ausgeführt, daher ist es wichtig diesen Befehl möglichst oft aufzurufen.
Der Befehl sollte also in jedem Loop Durchgang aufgerufen werden.
Achtung : Für das Pseudo Multitasking ist es wichtig, dass in der Loop Schleife keine zeitintensiven Befehle wie delay() o.Ä. verwenden werden.
Durch das Pseudo Multitasking entsteht in jedem Durchgang ein minimaler zeitlicher Drift. Umso häufiger der Update Befehl aufgerufen wird, desto kleiner ist dieser Drift. Beim Modewechsel wird der Drift allerdings immer wieder auf Null zurückgesetzt.

Parameter :
-

Beispiel :

```
void loop()
{
  flaeche.update();
  rumpf.update();
  showStrips();
}
```

3.1.7 void setBounds(uint16_t startLed, uint16_t endLed);

Mit diesem Befehl können die Grenzen eines Objekts nachträglich geändert werden.
Achtung : Dieser Befehl ist nur für fortgeschrittene Benutzer. Die Anwendung ist nicht ganz einfach und es gilt einiges zu beachten.

Parameter :
uint16_t startLed : Neue untere Grenze des Strips
uint16_t endLed : Neue obere Grenze des Strips

Beispiel :

```
//Im Folgenden soll das Flächen Objekt nur den vollen Strip (von 0-29) bekommen
flaeche.setBounds(0, 29);

//Da nun einige LEDs doppelt belegt sind muss der doppelte Bereich des anderen Strips deaktiviert werden
rumpf.disable();
```

3.1.8 void disable();

Deaktiviert den Strip.

Parameter :
-

Beispiel :

```
//Im Folgenden soll das Flächen Objekt nur den vollen Strip (von 0-29) bekommen
flaeche.setBounds(0, 29);

//Da nun einige LEDs doppelt belegt wären muss der überschriebene Bereich des anderen Strips deaktiviert werden
rumpf.disable();
```

3.2 Effekte und Parameter

3.2.1 Übersicht aller Effekte

CONST_COLOR : Eine Farbe leuchtet dauerhaft
CONST_TRIPLE_COLOR : Drei Farben im zeitlichen Wechsel
CHASE_SINGLE_COLOR : Eine Farbe füllt und leert den Strip immer wieder
CHASE_TRIPLE_COLOR : Drei Farben füllen und leeren den Strip im Wechsel
FILL_TRIPLE_COLOR : Drei Farben füllen den Strip im Wechsel
BLINK_COLOR : Eine Farbe blinkt
DOUBLE_FLASH : Ein Doppelplitz (z.B. Positionsbeleuchtung)
FILL_AGAINST : Eine Farbe füllt den Strip von der einen Seite, eine andere anschließend von der anderen Seite
THEATHER_CHASE : Einzelne Leuchtpunkte als Lauflicht
KNIGHTRIDER : Was gibt es dazu zu sagen?! ;)
THUNDERSTORM : Wildes Blitzen
POLICE_RIGHT : Amerikanisches Polizeilicht (in Verbindung mit POLICE_LEFT)
POLICE_LEFT : Amerikanisches Polizeilicht (in Verbindung mit POLICE_RIGHT)
RAINBOW : Regenbogen Lauflicht
BLOCKSWITCH : Abwechseln leuchten benachbarte Blöcke in verschiedenen Farben auf
RANDOMFLASH : Wildes leuchten verschiedener Farben

3.2.2 Parameter

Mit dem updateParameter Befehl (siehe 3.1.3) können die Effekte individuell eingestellt werden. Im Folgenden eine Übersicht über alle einstellbaren Parameter.
Die Standardwerte sind in Klammern angegeben.

Die Farben sind immer als kodierte Zahlen angegeben (und nur so zu verwenden).
Hier einige Beispiele :
Rot : 16711680
Grün : 65280
Blau : 255
Orange : 16747520
Weiß : 16777215

Die Adafruit Neopixel Library bringt eine Funktion mit, mit deren Hilfe diese kodierten Zahlen aus RGB Werten erzeugt werden können :

```
strip.Color(255,0,0); //strip.Color(R, G, B); (R, G, B, zwischen 0-255)
```

Statt den festen Zahlen kann also auch die Funktion strip.Color(R, G, B) übergeben werden.

Bei der Einstellung werden nur Eingaben berücksichtige die NICHT 0 sind. Überall wo eine 0 übergeben wird bleibt der Standardwert erhalten.

Beispiel :

```
//Einstellen des Knightrider Lauflichts
flaeche.updateParameter(KNIGHTRIDER, 0, strip.Color(25,25,25), 5, 50);

//Name des Effekts : KNIGHTRIDER
//Parameter 1 : 0 → Die Farbe des Lauflichts (soll hier nicht geändert werden und daher 0)
//Parameter 2 : strip.Color(25,25,25) → Farbe des restlichen Strips (hier ein gedimmtes Weiß)
//Parameter 3 : 5 → Breite des Lauflichts (das Lauflicht in nun 5 LEDs breit)
//Parameter 4 : 50 → Das Zeitintervall des Lauflichts (wurde hier auf 50ms halbiert → doppelt so schnell)
```

3.2.2.1 CONST_COLOR

Parameter 1 : Farbe (16711680)
Parameter 2 : keine Bedeutung
Parameter 3 : keine Bedeutung
Parameter 4 : keine Bedeutung

3.2.2.2 CONST_TRIPLE_COLOR

Parameter 1 : Farbe 1 (16711680)
Parameter 2 : Farbe 2 (65280)
Parameter 3 : Farbe 3 (255)
Parameter 4 : Zeit bis zum Wechsel in Millisekunden (200)

3.2.2.3 CHASE_SINGLE_COLOR

Parameter 1 : Farbe (16711680)
Parameter 2 : keine Bedeutung
Parameter 3 : keine Bedeutung
Parameter 4 : Zeit bis zum Ein- bzw. Ausschalten der nächsten LED (kleinere Zahl → schnellerer Durchlauf) (50)

3.2.2.4 CHASE_TRIPLE_COLOR

Parameter 1 : Farbe 1 (16711680)
Parameter 2 : Farbe 2 (65280)
Parameter 3 : Farbe 3 (255)
Parameter 4 : Zeit bis zum Ein- bzw. Ausschalten der nächsten LED (kleinere Zahl → schnellerer Durchlauf) (50)

3.2.2.5 FILL_TRIPLE_COLOR

Parameter 1 : Farbe 1 (16711680)
Parameter 2 : Farbe 2 (65280)
Parameter 3 : Farbe 3 (255)
Parameter 4 : Zeit bis zum Einschalten der nächsten LED (kleinere Zahl → schnellerer Durchlauf) (50)

3.2.2.6 BLINK_COLOR

Parameter 1 : Farbe 1 (16777215)
Parameter 2 : Zeit die die LEDs an sind (50)
Parameter 3 : Zeit die die LEDs aus sind (150)
Parameter 4 : keine Bedeutung

3.2.2.7 DOUBLEFLASH

Parameter 1 : Farbe (16777215)
Parameter 2 : keine Bedeutung
Parameter 3 : keine Bedeutung
Parameter 4 : keine Bedeutung

3.2.2.8 FILL_AGAINST

Parameter 1 : Farbe 1 (16711680)
Parameter 2 : Farbe 2 (65280)
Parameter 3 : keine Bedeutung
Parameter 4 : Zeit bis zum Einschalten der nächsten LED (kleinere Zahl → schnellerer Durchlauf) (50)

3.2.2.9 THEATER_CHASE

Parameter 1 : Farbe (16747520)
Parameter 2 : keine Bedeutung
Parameter 3 : keine Bedeutung
Parameter 4 : Zeit bis zum Einschalten der nächsten LED (kleinere Zahl → schnellerer Durchlauf) (50)

3.2.2.10 KNIGHTRIDER

Parameter 1 : Farbe des Lauflichts (16711680)
Parameter 2 : Farbe des restlichen Strips (0 = Aus)
Parameter 3 : Breite des Lauflichts in Anzahl an LEDs (3)
Parameter 4 : Zeit bis zum Weiterschalten (kleinere Zahl → schnellerer Durchlauf) (70)

3.2.2.11 THUNDERSTORM

Parameter 1 : Farbe (16777215)
Parameter 2 : keine Bedeutung
Parameter 3 : keine Bedeutung
Parameter 4 : keine Bedeutung

3.2.2.12 POLICE_RIGHT

Parameter 1 : keine Bedeutung
Parameter 2 : keine Bedeutung
Parameter 3 : keine Bedeutung
Parameter 4 : keine Bedeutung

3.2.2.13 POLICE_LEFT

Parameter 1 : keine Bedeutung
Parameter 2 : keine Bedeutung
Parameter 3 : keine Bedeutung
Parameter 4 : keine Bedeutung

3.2.2.14 RAINBOW

Parameter 1 : keine Bedeutung
Parameter 2 : keine Bedeutung
Parameter 3 : keine Bedeutung
Parameter 4 : Zeitintervall (kleinere Zahl → schnellerer Durchlauf) (5)

3.2.2.15 BLOCKSWITCH

Parameter 1 : Farbe des ersten Blocks (16711680)
Parameter 2 : Farbe des zweiten Blocks (65280)
Parameter 3 : Breite der Blöcke (5)
Parameter 4 : Zeit bis zum Wechsel (kleinere Zahl → schnellerer Durchlauf) (150)

3.2.2.16 RANDOMFLASH

Parameter 1 : keine Bedeutung
Parameter 2 : keine Bedeutung
Parameter 3 : keine Bedeutung
Parameter 4 : Zeit bis zum Wechsel (kleinere Zahl → schnellerer Durchlauf) (25)

4. Changelog

V0.5 (28.02.2015)

- erste Release Version
- Handbuch angelegt
- 13 Effekte implementiert

V0.6 (01.03.2015)

- enable Befehl entfernt (Strip wird beim Moduswechsel automatisch wieder aktiviert und die Grenzen zurückgesetzt)
- Rainbow Effekt hinzugefügt
- Blockswitch Effekt hinzugefügt
- Randomflash Effekt hinzugefügt
- Standardzeiten teilweise geändert