Flat-Field Tool für ASKAR 230 / Sharpstar TS EDPH61

Normalerweise verwende ich für meine Teleskope (80er APO und 200er Newton) eine DIN A3 LED Zeichenplatte, den ich umgekehrt(!) auf das senkrecht stehende Teleskop lege und meine Flats mache. Umgekehrt, weil die direete Beleuchtung ansonsten auch in der niedrigsten Stufe zu stark ist.

So erziele ich die halbe ADU meiner 14Bit Kamera zusammen mit dem integrierten SW-Flat Tool in der EKOS-SW (Astroberry).

Für meinen Neuzugang TS EDPH61 habe ich aber bei Thingiverse eine 3-teilige Druckvorlage gefunden. Das Tool wird dann auf den Tubus aufgesteckt, die EL-Folie aktiviert und los geht's mit den Flats. https://www.thingiverse.com/thing:5275670

Die untere Platte dient als Boden und Halter für die EL-Folie sowie den Hochspannungsgenerator, der seitlich angeklebt wird. Da sich bei meinem ersten Versuch herausstellte, dass die Folie nur auf niedrigster Stufe (1 von 255) meinen gewünschten Helligkeitswert erzielte, habe ich hier in der neuen Version noch ein Stück weiße Kunststofffolie eingelegt. Auf Stufe 9 von 255 bei Steuerung über einen Arduino erziele ich sowmit eine ca. 3 Sekunden Aufnahme bei der gewünschten ADU.

Die Platte mit dem Rohransatz passt stark klemmend in die Bodenplatte. Die Ecken habe ich etwas gerundet, denn Ecke auf Ecke paßt nicht wirklich.

Wirklich kritisch ist der Tubusring, denn der muß möglichst saugend auf den Teleskop-Tubus als auch in den Rohransatzpassen. Da mein Drucker nicht perfekt arbeitet, habe ich den Ring neu konstruiert und 6 Probedrucke mit jeweils um 1/10 mm unterschiedlich Durchmesser gedruckt. Der letzte passte dann nahezu. Ich mußte aber innen noch mal ganz kurz mit der Dremel und einem Fächerschleifer durchgehen.

Die EL-Folie und den Hochspannungsgenerator gibt's hier:

https://www.el-light.shop/El-Folien-standard/fertige-El-Folien

Die Folie hat aber den Anschluss nicht mittig, sondern seitlich. Man muß also mit einer feinen Säge einen kleinen zusätzlichen Ausschnitt in die Bodenplatte sägen. Ich habe die 12V Ausführung des Generators gewählt und probeweise an ein einstellbares Steckernetzteil angeschlossen. Die Leuchtstärke läßt sich schon beginnend ab 3V Betriebsspannung stufenweise einstellen. Das ist aber sehr grob.

Mithilfe eines solchen Stellers (http://www.amazon.de/dp/B08JPS6DDP/)sollte auch eine stufenlose und feinfühlige Einstellung möglich sein.

Ich bin aber noch einen Schritt weitergegangen und habe einen Arduino NANO und ein FET-Modul (https://www.ebay.de/itm/323636532570) vorgeschaltet. Damit ist die Einstellung über ein Astro-Programm möglich, welches den „ALNITAK-Code" Code beherrscht. Dies ist u.a. der Fall bei N.I.N.A., SQG Pro und EKOS (Astroberry). Damit läßt sich die Folie ein- und ausschalten sowie in 255 Stufen dimmen.

Ich selbst nutze wie gesagt das Flat-Tool in EKOS auf einem Astroberry. Dort stellt man die gewünschte ADU ein (z.B. halbe mögliche ADU, bei 14Bit Astro-Kameras 8192ADU), sowie einen Toleranzwert (habe mal 300 ADU eingegeben).

Dann eine Sequenz generieren (z.B. 50 Aufnahmen mit der korrekten Gain / Offset / Temperatur) bei 1sek (!) Belichtungszeit. Nun das Tool starten und es wird solange die Belichtungszeit angepasst, bis die gewünschte ADU erreicht wird. Dann werden die Aufnahmen gemacht.

In meinem Fall muß ich beim filterlosen EDPH61 und eine Lage Kunststoffolie vor der EL-Folie auf einen PWM-Wert von 9 dimmen, um ca. 3sek Bild zu erreichen bei halber ADU. Mit der Kunststofffolie zur Dämpfung habe ich somit ausreichend Spiel nach unten und oben, um auch bei Einsatz mit Filtern noch genügend Leuchtstärke zu erreichen.

Den Arduino-Code gibt's hier. Achtung, die Entwickler von APT haben den ALNITAK Code fehlerhaft implementiert. Die Befehle beinahlten eigentlich 0-Werte, während APT O's verwendet hat. man muß also den passenden Code nehmen oder einige wenige Code-Zeilen anpassen und je nach Einsatz bzw. Programm die Befehle mit 3 x O oder 3 x 0 ausstatten. Den Code für Astroberry und Stellarmate habe ich hier angehängt.

Ich habe den Code noch ein wenig abgeändert, um anstatt mit ca. 970Hz PWM mit 62,5kHz takten zu können. Hier die betreffenden Code-zeilen:

```
void setup()
{
  // initialize the serial communication:
  Serial.begin(9600);
  //Serial.begin(115200);
  // initialize the ledPin as an output:

//Taktrate geändert
//TCCR0B = TCCR0B & B11111000 | B00000010; // for PWM frequency of 7812.50 Hz
TCCR0B = TCCR0B & B11111000 | B00000001; // for PWM frequency of 62500.00 Hz

 //folgende Zeile freischalten für langsames takten mit 970Hz
  //pinMode(ledPin, OUTPUT);
  analogWrite(ledPin, 0);
}
```

Noch ein bisschen Lesefutter:

- https://www.cloudynights.com/topic/536533-diy-alnitak-flat-panel/page-5
- https://www.blackwaterskies.co.uk/2020/03/cheap-diy-remote-controlled-flat-panel/

Hier das INO File für die Sequence Generator Pro Version und für N.I.N.A., Ekos, usw.
https://github.com/jwellman80/ArduinoLightbox/blob/master/LEDLightBoxAlnitak.ino

ttps://github.com/red-man/Alnitak-Lightbox-Clone/commit/55aa19c80eae0a40f6c9678343a71b814a9e7cec

Alternative PWM Frequenzen für „experten“

https://www.etechnophiles.com/how-to-change-the-pwm-frequency-of-arduino-nano/

TCCR0B = TCCR0B & B11111000 | B00000001; // for PWM frequency of **62500.00 Hz**

TCCR0B = TCCR0B & B11111000 | B00000010; // for PWM frequency of **7812.50 Hz**

TCCR0B = TCCR0B & B11111000 | B00000011; // for PWM frequency of **976.56 Hz** (The DEFAULT)

TCCR0B = TCCR0B & B11111000 | B00000100; // for PWM frequency of **244.14 Hz**

TCCR0B = TCCR0B & B11111000 | B00000101; // for PWM frequency of **61.04 Hz**

Arduino Code für EKOS, einfach per Drag & Drop in die Arduino IDE kopieren:

```
/*
What: LEDLightBoxAlnitak - PC controlled lightbox implmented using the
  Alnitak (Flip-Flat/Flat-Man) command set found here:
  http://www.optecinc.com/astronomy/pdf/Alnitak%20Astrosystems%20GenericCommandsR3.pdf
Who:
  Created By: Jared Wellman - jared@mainsequencesoftware.com
  Updated By: red-man
Changelog:
2020-06-25
  Updated to align with spec in link above.  Previously it was appending '000\n' instead
  of 'OOO\r' to the responses and did not work with NINA.
Typical usage on the command prompt:
Send     : >SOOO\n      //request state
Recieve  : *S19OOO\n    //returned state
Send     : >B128\n      //set brightness 128
```

```
Recieve  : *B19128\n    //confirming brightness set to 128
Send     : >JOOO\n      //get brightness
Recieve  : *B19128\n    //brightness value of 128 (assuming as set from above)
Send     : >LOOO\n      //turn light on (uses set brightness value)
Recieve  : *L19OOO\n    //confirms light turned on
Send     : >DOOO\n      //turn light off (brightness value should not be changed)
Recieve  : *D19OOO\n    //confirms light turned off.

Modifiziert -> OOO gegen 000 getauscht
hohe Taktrate am PWM Ausgang 6 eingestellt
*/

volatile int ledPin = 6;      // the pin that the LED is attached to, needs to be a PWM pin.
int brightness = 0;

enum devices
{
  FLAT_MAN_L = 10,
  FLAT_MAN_XL = 15,
  FLAT_MAN = 19,
  FLIP_FLAT = 99
};

enum motorStatuses
{
  STOPPED = 0,
  RUNNING
};

enum lightStatuses
{
  OFF = 0,
```

```
  ON
};

enum shutterStatuses
{
  UNKNOWN = 0, // ie not open or closed...could be moving
  CLOSED,
  OPEN
};



int deviceId = FLAT_MAN;
int motorStatus = STOPPED;
int lightStatus = OFF;
int coverStatus = UNKNOWN;

void setup()
{
  // initialize the serial communication:
  Serial.begin(9600);
  //Serial.begin(115200);
  // initialize the ledPin as an output:

//Taktrate geändert
//TCCR0B = TCCR0B & B11111000 | B00000010; // for PWM frequency of 7812.50 Hz
TCCR0B = TCCR0B & B11111000 | B00000001; // for PWM frequency of 62500.00 Hz

 //folgende Zeile freischalten für langsames takten mit 970Hz
  //pinMode(ledPin, OUTPUT);
  analogWrite(ledPin, 0);
}
```

```
void loop()
{
  handleSerial();
}



void handleSerial()
{
  if( Serial.available() >= 6 )  // all incoming communications are fixed length at 6 bytes including the
\n
  {
    char* cmd;
  char* data;
    char temp[10];


    int len = 0;


    char str[20];
    memset(str, 0, 20);


  // I don't personally like using the \r as a command character for reading.
  // but that's how the command set is.
    Serial.readBytesUntil('\r', str, 20);


  cmd = str + 1;
  data = str + 2;


  // useful for debugging to make sure your commands came through and are parsed correctly.
    if( false )
    {
      sprintf( temp, "cmd = >%c%s;\n", cmd, data);
      Serial.print(temp);
```

```
}



switch( *cmd )
{
/*
Ping device
  Request: >POOO\n
  Return : *PiiOOO\n
    id = deviceId
*/
  case 'P':
  sprintf(temp, "*P%d000\n", deviceId);
  Serial.print(temp);
  break;

  /*
Open shutter
  Request: >OOOO\n
  Return : *OiiOOO\n
    id = deviceId
  This command is only supported on the Flip-Flat!
*/
  case 'O':
  sprintf(temp, "*O%d000\n", deviceId);
  SetShutter(OPEN);
  Serial.print(temp);
  break;



  /*
```

```
Close shutter
  Request: >COOO\n
  Return : *CiiOOO\n
    id = deviceId
  This command is only supported on the Flip-Flat!
*/
  case 'C':
  sprintf(temp, "*C%d000\n", deviceId);
  SetShutter(CLOSED);
  Serial.print(temp);
  break;

/*
Turn light on
  Request: >LOOO\n
  Return : *LiiOOO\n
    id = deviceId
*/
  case 'L':
  sprintf(temp, "*L%d000\n", deviceId);
  Serial.print(temp);
  lightStatus = ON;
  analogWrite(ledPin, brightness);
  break;

/*
Turn light off
  Request: >DOOO\n
  Return : *DiiOOO\n
    id = deviceId
*/
  case 'D':
```

```
  sprintf(temp, "*D%d000\n", deviceId);
  Serial.print(temp);
  lightStatus = OFF;
  analogWrite(ledPin, 0);
  break;

/*
Set brightness
  Request: >Bxxx\n
    xxx = brightness value from 000-255
  Return : *Biiyyy\n
    id = deviceId
    yyy = value that brightness was set from 000-255
*/
  case 'B':
  brightness = atoi(data);
  if( lightStatus == ON )
    analogWrite(ledPin, brightness);
  sprintf( temp, "*B%d%03d\n", deviceId, brightness );
  Serial.print(temp);
    break;

/*
Get brightness
  Request: >JOOO\n
  Return : *Jiiyyy\n
    id = deviceId
    yyy = current brightness value from 000-255
*/
  case 'J':
    sprintf( temp, "*J%d%03d\n", deviceId, brightness);
    Serial.print(temp);
```

```
    break;

  /*
  Get device status:
    Request: >SOOO\n
    Return : *SidMLC\n
      id = deviceId
      M  = motor status( 0 stopped, 1 running)
      L  = light status( 0 off, 1 on)
      C  = Cover Status( 0 moving, 1 closed, 2 open)
  */
    case 'S':
      sprintf( temp, "*S%d%d%d%d\n",deviceId, motorStatus, lightStatus, coverStatus);
      Serial.print(temp);
      break;

  /*
  Get firmware version
    Request: >VOOO\n
    Return : *Vii001\n
      id = deviceId
  */
    case 'V': // get firmware version
    sprintf(temp, "*V%d001\n", deviceId);
    Serial.print(temp);
    break;
  }

while( Serial.available() > 0 )
  Serial.read();

}
```

```
}

void SetShutter(int val)
{
  if( val == OPEN && coverStatus != OPEN )
  {
    coverStatus = OPEN;
    // TODO: Implement code to OPEN the shutter.
  }
  else if( val == CLOSED && coverStatus != CLOSED )
  {
    coverStatus = CLOSED;
    // TODO: Implement code to CLOSE the shutter
  }
  else
  {
    // TODO: Actually handle this case
    coverStatus = val;
  }

}
```