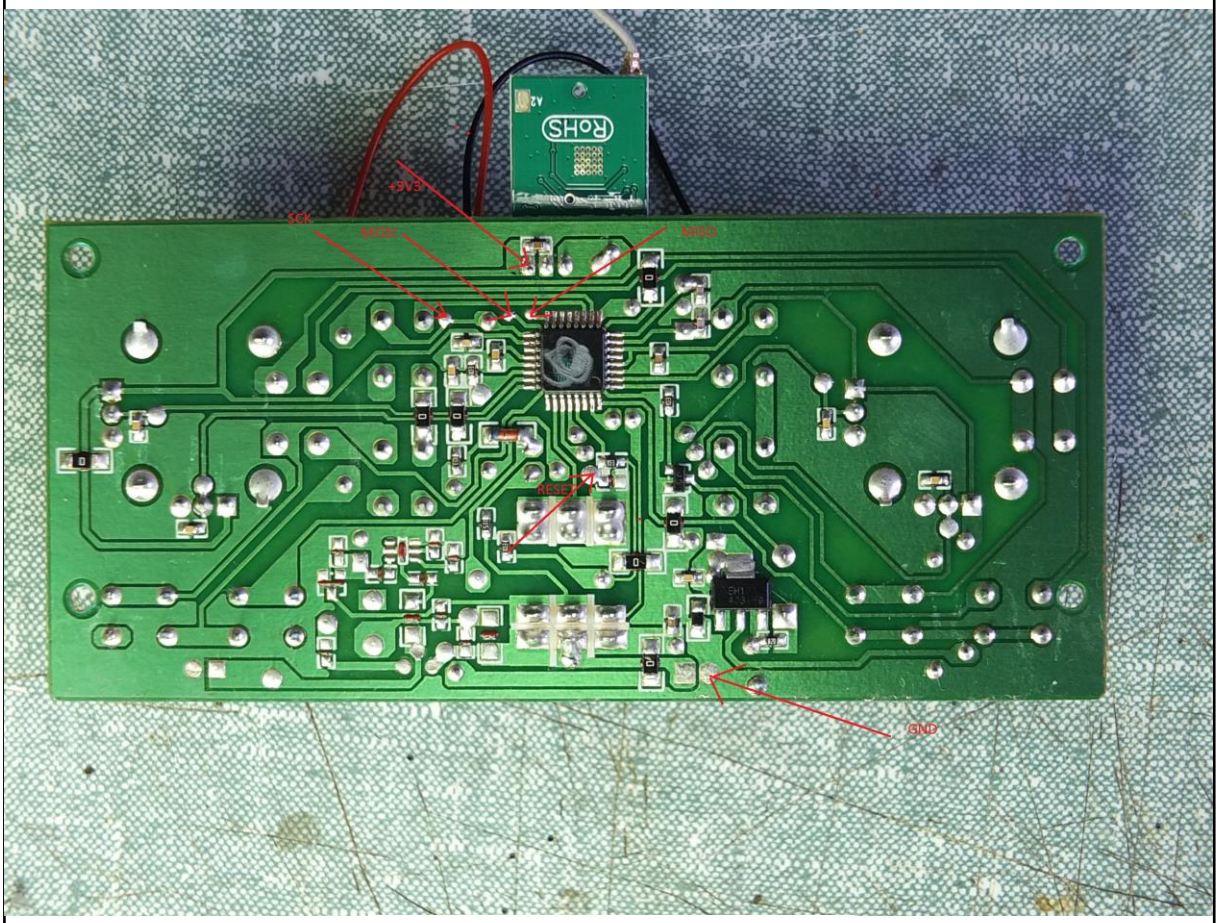
MLP4DSM

Flashen der alternativen Firmware



Vorwort

Diese Firmware ist primär entstanden um damit den Blade Nano CP X zu fliegen.

Der Nano CP X ist so schön klein, dass man ihn gut mal mit auf Reisen nehmen könnte.
Nur leider ist der kleinste Sender mit dem man den Nano CP X fliegen kann die DX4e, und dann nur im „nicht Computer Sender“ Modus. Im Computer Sender Modus muss es dann schon eine DX6i sein.
Beide Sender sind ziemliche Brocken im Koffer.

Die MLP4DSM währe da schon eher die richtige Größe. Nur leider lässt sich der Nano CP X mit der MLP4DSM nicht im den „nicht Computer Sender“ Modus versetzen. Und im „Computer Sender“ Modus geht der Pitch nicht. Warum weiß nur der Hersteller, mit der DX4e geht es ja schließlich auch.

Eigentlich war das Thema damit für mich gegessen, auf Reisen habe ich halt den mSR X mitgenommen.

Irgendwann bin ich beim Googeln dann über eine Custom Firmware für die DX4e gestolpert.
Ja dachte ich das währe auch eine Option.
Also mal nach einer Firmware für die MLP4DSM gesucht, leider Fehlanzeige.
Aber wenigstens einen (fast vollständigen) Schaltplan für die MLP4DSM brachte Google hervor.
http://www.baronerosso.it/forum/attachments/elimodellismo-micromodelli/220463d1327353167-blade-mqx-ultra-micro-quad-copter-mlp4dsm.pdf

Ein ATmega also. Das ist doch schon mal ein Anfang.
Ein schneller Test AVRDUDE „auto detect“ ergab ATmega 88p. Wie man vom Arduino ja weiß, gibt es für die AVRs, mit GCC AVR, eine ganz brauchbaren C Compiler.

Zur Sicherheit habe ich dann noch die Leiterplatte eingescannt und das Layout analysiert. Die Schaltung aus dem WEB stimmt, die zwei fehlenden Signale (PY &PP) wahren auch schnell zugeordnet.

Die Firmware entstand „from scratch“ ging aber eigentlich recht schnell. Die geringe FLASH Größe von 8K erwies sich aber als problematisch.
In Summe habe ich deutlich mehr Zeit dafür verwenden das Programm zu ändern, umzuschreiben und zu optimieren damit es in den Speicher passt, als mit dem eigentlichen Schreiben der Funktionalität.

Derzeit ist der FLASH Speicher zu 95,7% voll. Zwischendurch war es aber schon 3 x so, dass ich bei über 100% war, und noch nicht alle Funktionen implementiert waren die ich haben wollte.

Die Funktionalität für Flächenmodelle ist nicht getestet, da ich gar keine Flugzeuge besitze.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung

Weil es leider erforderlich ist: Hier einige Hinweise.
Lesen Sie diesen Abschnitt auf jeden Fall.
Beginnen Sie nicht mit dem Firmware Update ehe Sie diesen Abschnitt gelesen haben.

1.1 Rechtliches

Mit der Installation einer neuen Firmware auf Ihrer MLP4DSM erschaffen Sie ein neues Gerät (neue Funktionalität). Damit Sind Sie der Hersteller dieses neuen Gerätes.
Wenn Sie die modifizierte MLP4DSM verkaufen oder verschenken sind Sie auch der Inverkehrbringer.
Mit den entsprechenden rechtlichen Konsequenzen.

- Sie verlieren sämtliche Ansprüche gegen den original Hersteller und den Händler.
- Alle rechtlichen Pflichten die der Gesetzgeber bei dem Hersteller oder dem Inverkehrbringer sieht liegen bei Ihnen.

1.2 Haftungsausschluss

Ich übernehme keinerlei Garantie oder Haftung für irgendwas, weder kann ich die beschriebene Funktionalität garantieren, noch für Eigen- oder Fremdschäden einstehen die durch die Verwendung dieser Firmware eventuell entstehen.

1.3 Einverständnis

Installieren Sie diese alternative Firmware nur, wenn Sie mit den Punkten 1.1 und 1.2 einverstanden sind. Andernfalls installieren Sie die Firmware nicht.

1.4 Risiken

- Alle Rechtsansprüche gegen den Hersteller und Ihren Händler gehen verloren.
- Obwohl sehr selten, kann es beim Programmieren des Atmel in sehr seltenen Fällen passieren, dass der Mikrokontroller beschädigt wird.
  Dies ist insbesondere der Fall, wenn eine falsche Spannung verwendet wird oder die Regeln zum elektrostatischen Schutz nicht beachtet werden.
- Die Firmware des HF-Moduls bleibt von dieser Firmware unberührt, die HF Eigenschaften des Senders sollten sich also nicht geändert haben. Ohne teure Laborprüfung kann diese aber nicht nachgewiesen werden. Es besteht also ein theoretische Risiko, dass der Sender, nach dem einspielen der Firmware, nicht mehr den Regeln der EU (CE) entspricht.
- Sollten Sie einen USB zu Serial Wandler verwenden, um den seriellen Datenstrom zu HF Modul mitzulesen, so Achten Sie darauf, dass die Serielle mit TTL oder 3.3V Pegeln arbeitet. Eine reguläre RS232 hat +/- 12V diese Spannung führt mit Sicherheit zu einer Zerstörung des Mikrokontrollers.

2 Bevor Sie beginnen

2.1 Leiterplatte /Platine prüfen

Bitte prüfen Sie zuerst, ob die Platine Ihrer MLP4DSM den folgenden Abbildungen entspricht.
Wenn Ihre Platine anders aussieht müssen Sie erst prüfen ob die Schaltung gleich geblieben ist.
Im Zweifelsfall installieren Sie die Firmware nicht.

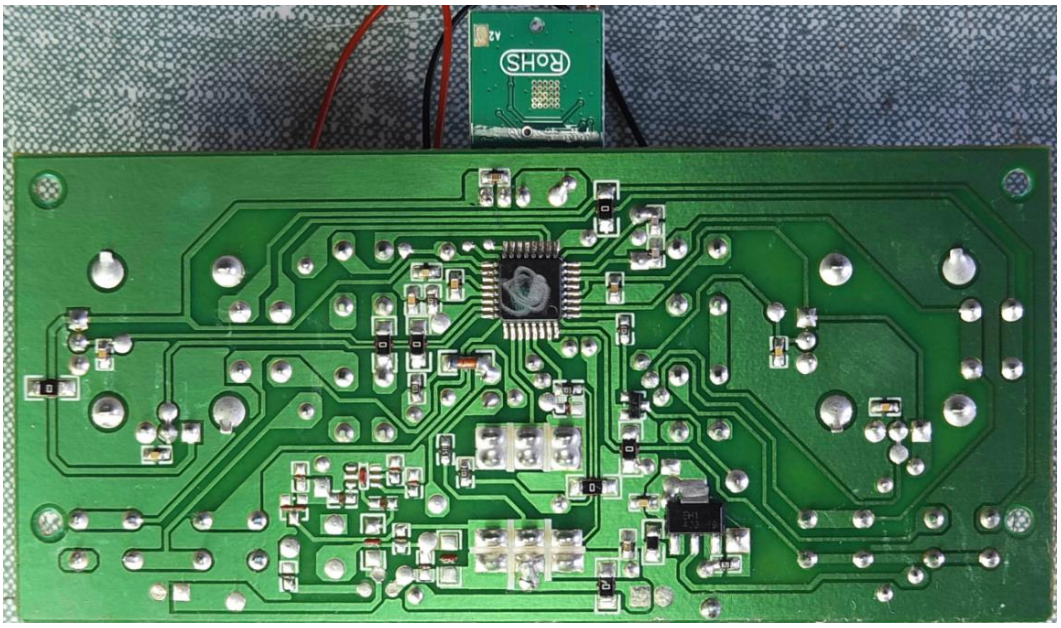


Abbildung 1 Die Leiterplatte der MLP4DSM von unten.

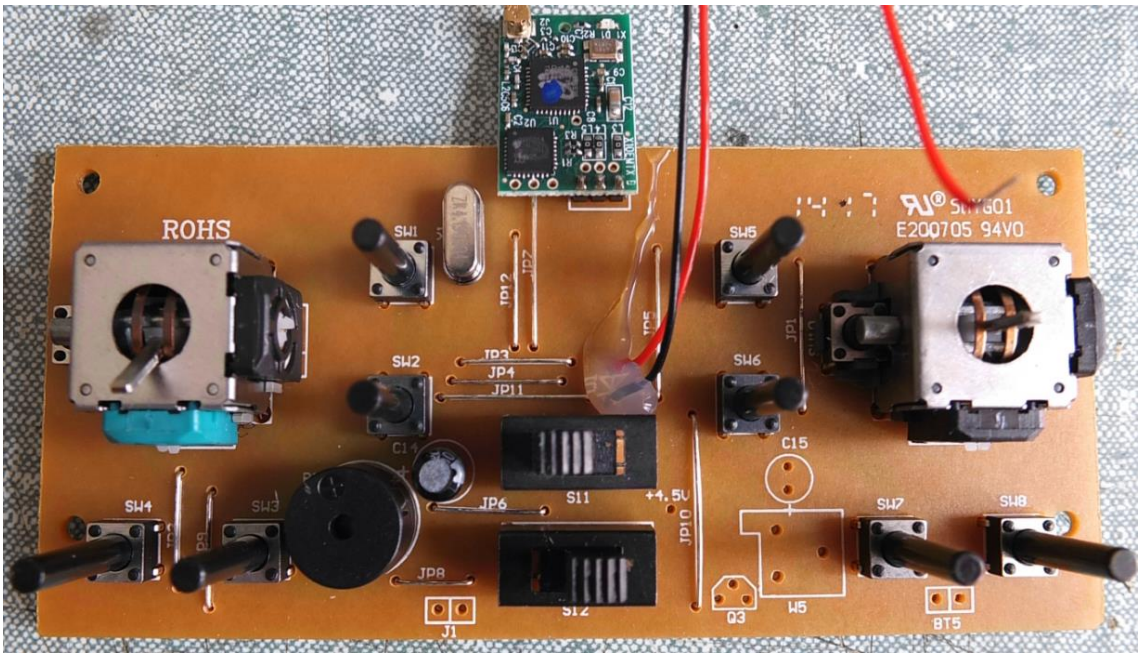


Abbildung 2 Die Leiterplatte der MLP4DSM von Oben.

Achten Sie insbesondere darauf, dass der Quarz eine Frequenz von 4.194304Mhz hat.
Wenn das nicht der Fall ist, können sie die Firmware nicht ohne Änderungen des Quellcodes und neu übersetzen des Selbigen verwenden.

Es gibt mindesten 2 Versionen der Platine, die andere Version hat nicht nur einen anderen Quarz, sondern verwendet auch eine andere Baud Rate zum HF Modul.
Die Version mit dem 4.194304Mhz verwendet eine Baud Rate von 131kBaud die andere Version 135kBaud.

2.2 Voraussetzungen

Für den Firmware-update sind einigen Voraussetzungen zu erfüllen.
In diesem Abschnitt werden diese gelistet.
Sie benötigen:

- Einen AVR Programmiergerät (Programmer).
  Wenn Sie schon eine haben, können Sie natürlich diese verwenden.
  Wenn Sie einen Arduino besitzen, können Sie diesen zu Programmer machen (arduino isp).
  Oder Sie kaufen sich einen einfachen Programmer, ich verwende z.B. einen billigen USBASP Clone.
- Eine AVR Programmiersoftware zum flashen des Progrmmspeichers.
  Ich verwende den AVRDUDE
- 3 bis 4 Leitungen (Adern) um den Programmer mit dem Atmel zu verbinden.
- Lötausrüstung (Feinlötkolben, Lötzinn, …)
- Rechner zum ausführen der Software.

Sie sollten:

- Löten können.
  Wenn Sie unerfahren im Löten sind, mach Sie erst einige Lötübungen auf einer alten Leiterplatte.
- Keine zwei linken Hände haben.
- Die Grundregeln zu ESD-Schutzmaßnahmen kennen und berücksichtigen.
  Elektrostatische Aufladungen können elektronische Bauteile zerstören.

Optional:
Bei der Entwicklung der Firmware aber auch beim Testen der Knüppelkalibrierung hat es sich als nützlich erwiesen, den seriellen Datenstrom zu RF Modul auszuwerten.
Wenn Sie einen FTDI USB zu TTL Converter haben, können Sie das Programm „DSM_Serial_Analyse“ verwenden, um den Datenstrom zu analysieren.
Leider läuft das Programm derzeit nur mit FTDI Umsetzern. Andererseits ist ein FT232RL basierender Umsetzter auch schon für 3€ zu haben.

2.3 Vorbereitung

Um den ATMEGA zu flashen, müssen sie Programmierleitungen an die Platine löten,
und mit Ihrem Programmer verbinden.

Minimum benötigen Sie:
MOSI
MISO
GND

Optional können Sie noch eine Leitung für Plus anlöten. Dann können Sie den Atmel programmieren ohne den Sender einzuschalten. Der Programmer speist dann die Schaltung.

Andernfalls (Keine Plus) müssen Sie Batterien im Sender haben, und diesen zu Programmieren einschalten.

Falls Sie die Option Serial Monitor verwenden wollen, Löten Sie zusätzlich 2 Leitungen (GND & TXD) für der seriellen Signale an.

Die Folgende Abbildung zeigt sie Anschlüsse auf der Platine.

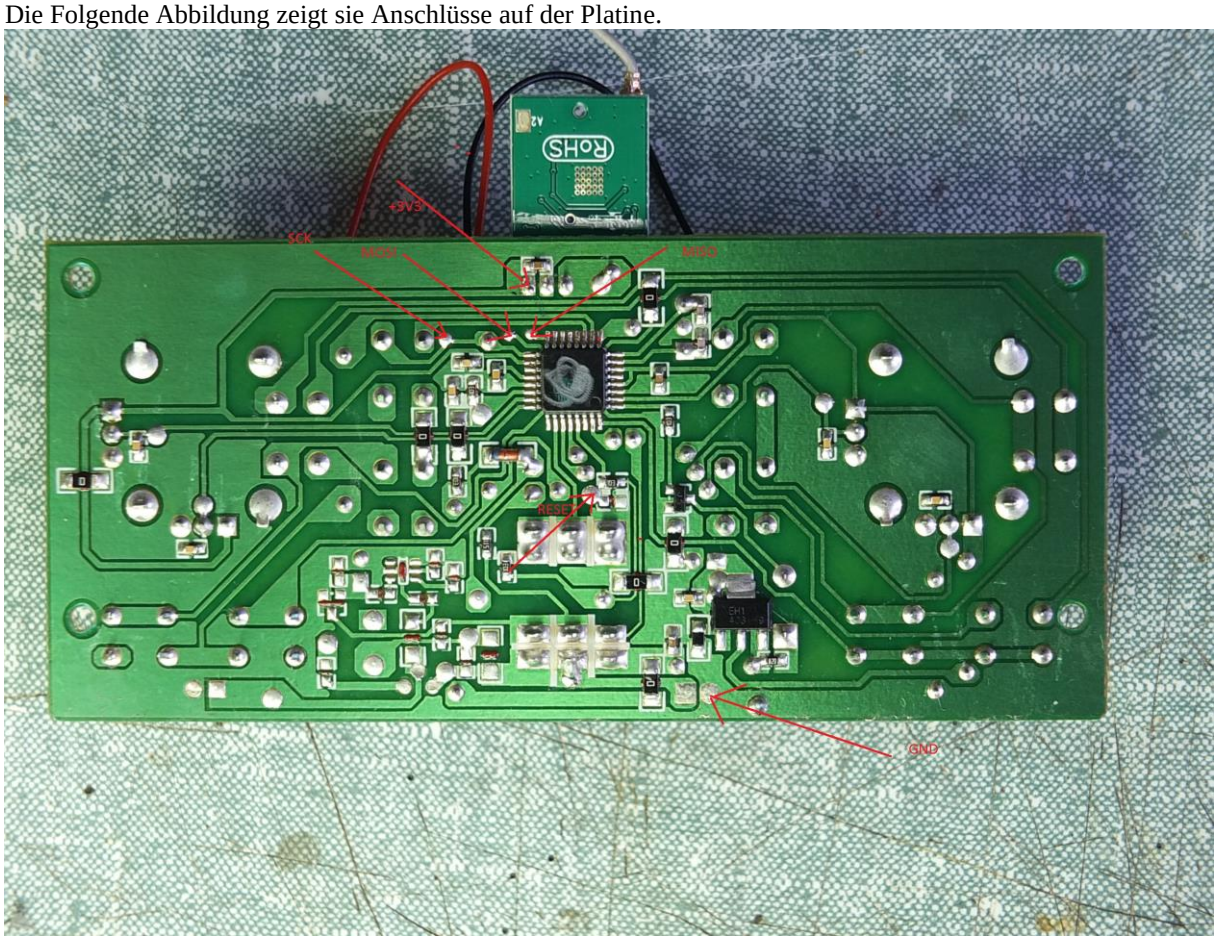


Abbildung 3 Die Anschlüsse auf der Leiterplatte

Hier noch meine Platine mit den Leitungen dran. Zusätzlich zu den Programmierleitungen sind noch zwei für den seriellen Monitor dran.

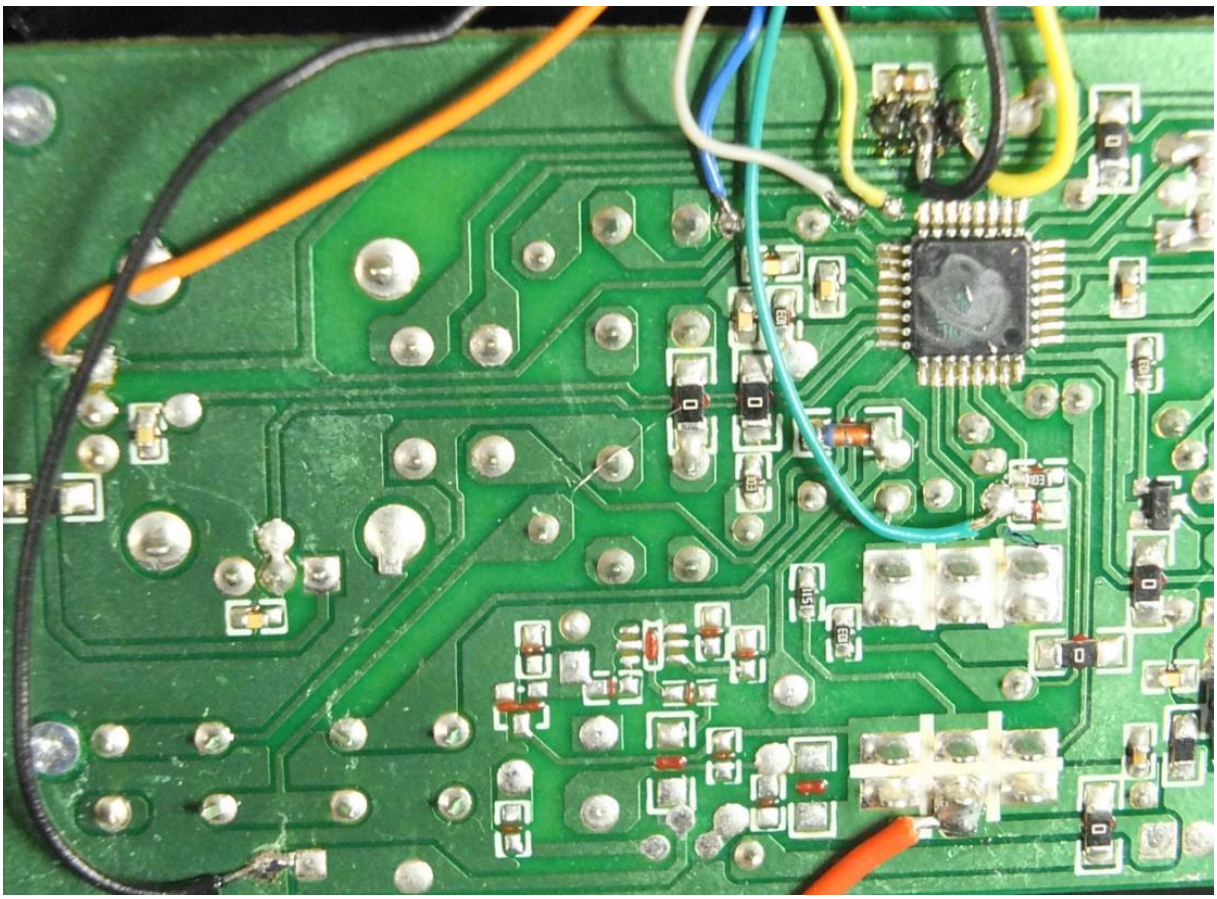


Abbildung 4 Meine LP mit Leitungen dran.

2.4 Vorab Test

Nachdem Sie die Leitungen angelötet und mit ihrem Programmer verbunden haben, ist es Zeit der Funktion zu prüfen.

- Prüfen Sie die Verdrahtung.
- Schließen Sie den Programmer an Ihren Rechner an.
- Starten Sie Ihre Programmiersoftware
- Wählen Sie Ihren Programmer
- Starten Sie „Auto Detect“ es sollte eine ATMEGA 88P erkannt werden.

3 Flashen der Firmware

Wenn der Vorab Test erfolgreich war, könne Sie jetzt die Firmware brennen.

- Wählen Sie die HEX Datei als Quelle.
- Drücken Sie Programm.

Das war‘s.

4 Fragen und Antworten

Frage: Kann ich die Original Firmware von Blade wiederherstellen?

Antwort: Leider nein, der Hersteller hat die Firmware nie als Binärdatei raus gegeben, und auslesen lässt sich die Firmware , wegen aktiver Code Protektion, auch nicht.

Frage: Nach dem Flaschen geht der Sender nicht mehr, was tun?

Antwort: Das sollte nicht passieren. Prüfen Sie die Verdrahtung!

Sie können es nochmal versuchen, oder schauen ob Sie einen Fachmann im Bekantenkreis haben.

Frage: Ich hätte gerne Taumelscheibenmix was muss ich machen?

Antwort: Sorry das unterstützt die Firmware nicht, Sie müssen den Quellcode andern und neu übersetzen.

Frage: Wo müssen MOSI MISO GND am Programmer ran?

Antwort: schauen Sie bitte in das Handbuch zu Ihrem Programmiergerät.

5 Ändern des Quellcodes

Ehe Sie jetzt großen Pläne machen was man noch alles ergänzen könnte. Der Programmspeicher (FLSH) ist zu 95,7% voll. Und das trotz diverser Optimierungen. Kleine Änderungen wie ein anderer Quarz (`F_CPU in LP4DSM.h)`, eine anderen Baud Rate (`USART_BAUDRATE in SPECTRUM_TX.h`) oder andere Default Werte (`DEFAULT.h`) sollten problemlos möglich sein.

Für Erweiterungen muss aber mit Sicherheit was anderes raus oder optimiert werden.

Um die Firmware zu übersetzen benötigen Sie das Atmel Studio 7 oder neuer.

5.1 Aufbau es Quellcodes

- Analog.c

  Einlesen der ADC mit Overscan. Die Daten werden im `ADC conversion complete interrupt` verarbeitet. Angestoßen wird der ADC über `TIMER1_COMPB`.
- Buzzer.c

  Spielt Töne oder Tonfolgen ab. Verwendet Timer 2 `CTC` Mode.

  Tonfolgen werden über den callback `BuzzerCyclic` vom systic Timer1 gesteuert.
- Config.c

  Lesen und Schreigen der Sender und Modellkonfiguration vom und zum EEPROM.
- Curve.c

  Expo, Rate, Sacle, 3 Punkt Kurve, und Mixer
- Keys.c

  Einlesen der Tasten mit overscan, `key_cyclic` wird von systic Timer 1 aufgerufen.

- LED.c

  Ansteuern der LED: Leuchten, Blinken und Sequenzen, verwendet Timer 0 PWM

  `LedCyclic` wird von systic Timer 1 aufgerufen.
- Model.c

  Modelltype spezifische Verarbeitung der Sticks und Tasten werte.
- Spektrum_tx.c

  Ansteuerung des UART und Aufbereitung der Daten für das RF Modul.

  Das Sender der Daten geschieht interrupt gesteuert (`USART_UDRE)`.
- Sticks.c

  Verarbeitung der analogen Signale der Knüppel, Skalieren wegen Input Kalibrierung,

  Trimm hinzurechnen
- Timer.c

  Systic 1ms (`TIMER1_COMPA)` und Takt für ADC Messung(`TIMER1_COMPB)`.

  Im (`TIMER1_COMPB)`wird auch, 7.56ms nachdem das Einlesen der Analogwerte fertig ist, das Senden der seriellen Daten angestoßen. Das ist erforderlich, weil das RF Modul sehr empfindlich auf Jitter (zeitliche Schwankungen) reagiert. Die Verarbeitung der Daten im Hauptprogramm muss in den 7.56ms zwisch `adc_new_data` und `adc_checkSend` abgeschlossen sein, sonst werden die Werte erst 22ms später gesendet.
- User_if.c

  Benutzerschnittstelle, Abfrage der Eingaben, Konfigurationsmenus
- Main.c

  Initialisierung (`InitSystem)`, Abfrage ob Konfigurationsmenu, Bind Key, Range Check Key,

  Hauptschleife verarbeitet `adc_new_data` ließt Tasten, prüft Batteriespannung und Flight Timer.

  Die Verarbeitung der Analogdaten `adc_new_data` muss in weniger als 7.56ms erfolgen (bei allen 3 Modelltypen) ansonsten muss `adc_checkSend` angepasst werden.

5.2 Einsparmöglichkeiten um die Codegröße zu reduzieren

Analog:

Der overcsan wirkt als Tiefpass, um das zu kompensieren wird die Steigung berechnet und dem Ausgangssignal zugemischt (Extrapolation einer Graden). Ob das im Flugbetrieb was bringt habe ich aber nie getestet.

Wahrscheinlich kann das mit der Steigung raus, ohne dass man es merkt.

Da sie Eingangssignal recht gut bedämpft sind, kann vermutlich der ganze overscan weg, dann ist halt ein wenig mehr Rauschen im Signal.

Keys:

Der overscan für die Taster ist definitive nicht zwingend erforderlich.

Möglicherweise bringt es auch schon was das Ergebnis von `Get_keys` nicht zurückzugeben, sondern in einer Globalen variable zu halten. `Get_Key` und `Get_Key_Rising_Edge` bräuchten dann auch einen Parameter weniger. Achtung `GET_KEY` und `GET_KEY_RISING_EDGE` sind derzeit als makros implementiert.

LED:

Der derzeitige Sequenzer ist, mit Sicherheit für die paar Funktionen Leuchten, Blinken, und Blinkfolge nicht die sparsamste Implementierung.

STICKS:

Derzeit wird 2x skaliert erst in `getRawStickValue` die Kalibrierung und dann in `getStickValue` der Trimm. Das ließe sich zusammenfassen, allerdings um den Preis einer etwas ungenaueren Eingabe vor Werten im Konfigurationsmenu.

5.3 FAQ Quellcode

F:Warum skalieren Sie den Eingangsbereich von 0-1023 auf +-2000 wo doch am Ende wieder 0-1023 gebraucht werden?

A: Wollte ich eigentlich nicht, aber dann müssen alle Funktionen in curve.c den Offset abziehen, es hat sich gezeigt, dass das mehr Programmspeicher kostet als das Skalieren am Ein- und Ausgang.

F: Wieso der ganze krempel mit der Verarbeitung im Interrupt, währe es nicht einfacher das direkt im Hauptprogramm zu machen?

A: Ja aber das RF Modul reagiert extrem empfindlich auf Schwankungen im Timing. Nur so habe ich einen stabilen Betrieb erreichen können.

F: Währe es nicht genauer mit Fließkomma zu rechnen?

A: Ja aber leider zu langsam. Schon jetzt braucht die Signalverarbeitung, beim Modeltype Airplane über 6 ms.

Anlage A: Der Schaltplan aus dem Netz

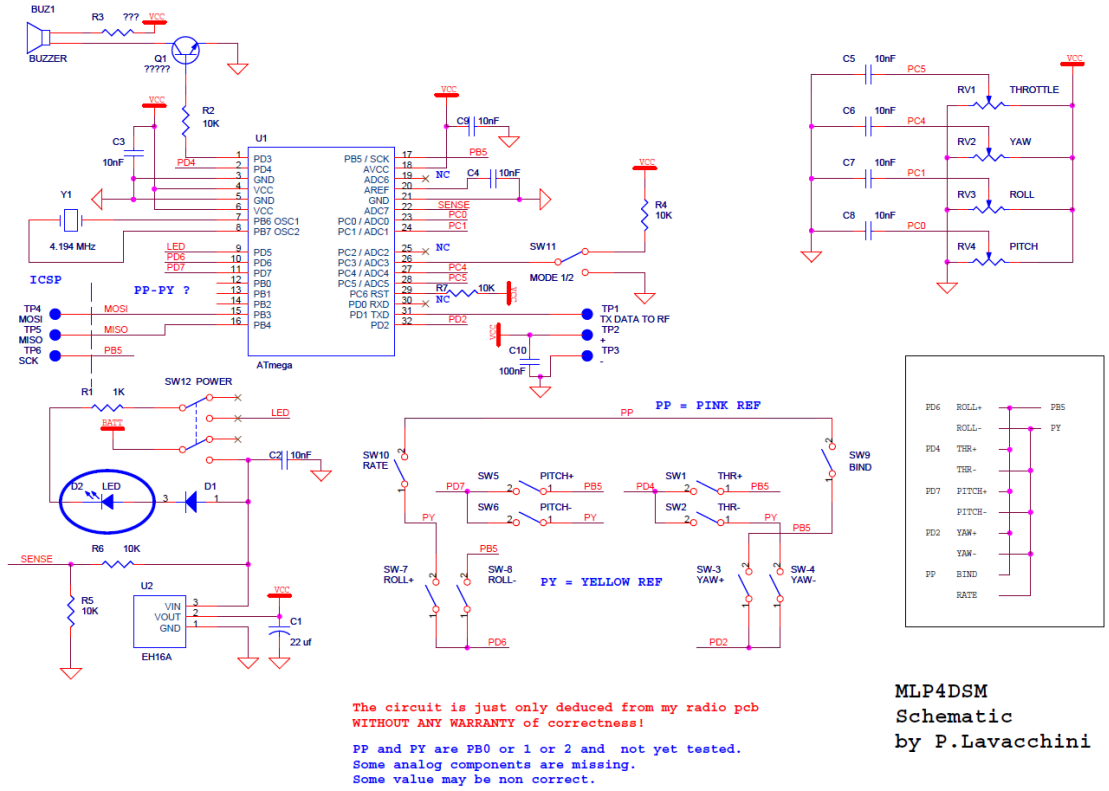


Abbildung 5 Schaltplan

Anlage B: Das gescannten und bearbeitete Layout

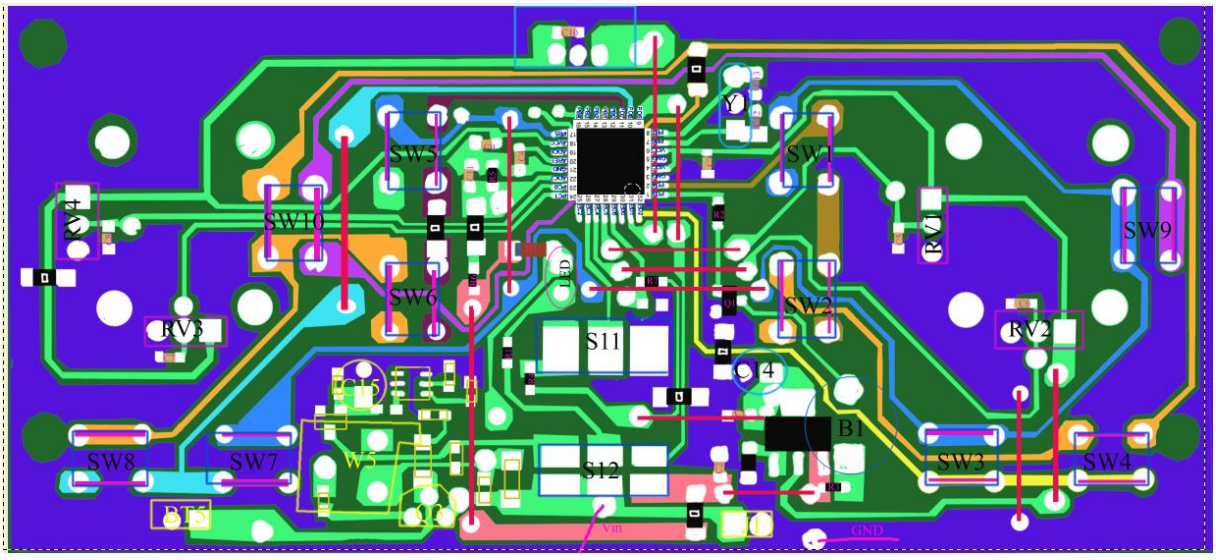


Abbildung 6 Layout von unten mit Komponenten

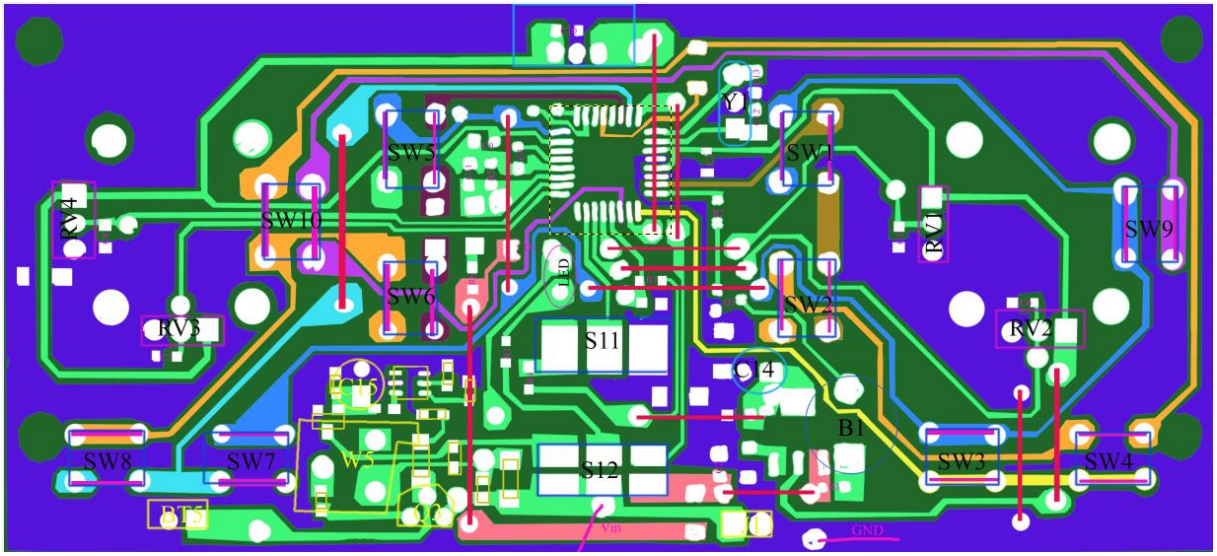


Abbildung 7 Layout von unten ohne Komponenten