Rexus Service Modul Simulator

Forschungsarbeit

von

Jakov Kholodkov

geboren am 28. Oktober 1996 in Moskau

8 Januar 2014

Betreuer:
Dipl.-Ing. Max Rössner
OStR. Stefan Kiessling
Thomas Grübler

Technische Universität München
Fakultät für Elektrotechnik und Informationstechnik
Lehrstuhl Messsystem - und Sensortechnik

in Kooperation mit

Otto-von-Taube-Gymnasium in Gauting

Abstract

The project deals with the design and development of a circuit-board, which is called REXUS-service-module-simulator (RXSMS). It is used for testing other circuits and experiments that are going to be flown on the sounding rocket REXUS. The REXUS rocket is a part of the REXUS-BEXUS project, which offers students the opportunity to test newly developed applications under space-conditions. These applications have to be tested before flight, since the rocket flies only once. The RXSMS gives the opportunity to verify the experiment's and the ground-station's basic functions. During the test both have to be connected to the RXSMS, which represents the rocket service module and a noisy radio connection between the rocket- and the ground antenna. This means, the RXSMS simulates the control-signals of the rocket service-module and the radio link to the ground.

Inhaltsverzeichnis

1 Projektumfeld

Die Forschungsarbeit ist Teil des REXUS Projekts, welches Studententeams europäischer Universitäten die Möglichkeit zur Verfügung stellt Experimente auf einer Forschungsrakete in die Stratosphäre zu befördern. Diese Gelegenheit wird von den Studenten genutzt um neue Technologien unter Weltraumbedingungen zu testen. Darüber hinaus lernen die Studenten, in einem Team an einem größeren Projekt zusammen zu arbeiten und sich dabei an strikte Vorgaben und Abgabetermine zu halten. Gefördert wird dieses Projekt vom Deutschen Zentrum für Luft- und Raumfahrt (DLR) und von der Schwedischen Nationalen Raumfahrtbehörde (SNSB).

Abbildung 1: REXUS auf der Startrampe[1]

Das REXUS Service Modul

Die 5.6 Meter lange Höhenforschungsrakete trägt bis zu 90 kg Last (Payload) in eine Höhe von ca. 80 km. Während der etwa 7 minütigen Flugphase werden die Experimente der Studententeams durchgeführt. Für das Starten der Experimente und für das Versenden der Messdaten verfügt die Rakete über das REXUS-Service Modul (RXSM), welches wiederum mit einer Funkeinheit verbunden ist. Die Daten gelangen vom Experiment über das RXSM zur Funkeinheit und werden über eine Radioverbindung an die Erde gesendet. Anschließend werden sie von einer Antenne empfangen, an die Bodenstation weitergeleitet, dort ausgewertet und analysiert. Die Rakete stellt für die Kommunikation mit der Payload ein D-SUB15 Interface mit folgenden Signalen zur Verfügung (vgl. Abbildung 2)[2, Seite 39]:

- eine 28 V ungeregelte Spannungsversorgung
- ein Telecommand/Telemetrie über das RS422 Bussystem (Funkverbindung zur Erde)
- die Steuersignale
  - Lift-Off (LO), wird beim Abheben der Rakete betätigt (active-low)
  - Start of Experiment (SOE), einzeln einstellbar für jedes Experiment (active-low)
  - Start of Data Storage (SODS), ebenfalls einzeln einstellbar (active-low)

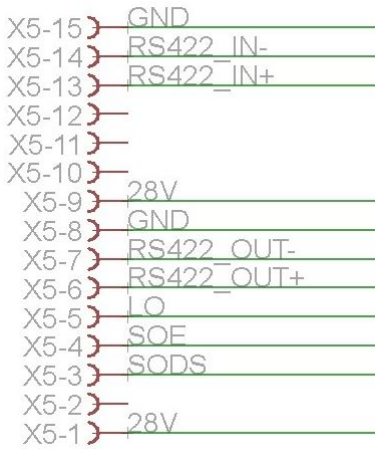

Abbildung 2: RXSM Interface

2 Problemstellung

Vor dem Flug der Rakete müssen die mitfliegenden Experimente auf Vibrationsfestigkeit und Kälteresistenz überprüft werden, um sicher zu gehen, dass die Einflüsse des Weltraums schadenfrei überstanden werden. Desweiteren müssen die internen Programmabläufe der verbauten Mikrochips, das Interface zum RXSM und das RS422 Bussystem auf ihre Funktionalität überprüft werden. Ebenso ist das Herausfiltern der Übertragungsfehler, die durch das Rauschen der Radioverbindung in die Messdaten gelangt sind, eine sehr wichtige Funkion der Bodenstation und muss ebenso genau getestet werden. Außerdem muss die Bodenstation kurze Unterbrechungen der Datenverbindung verarbeiten können. Diese enstehen durch Vibrationen der Antenne, mit der die REXUS Rakete angepeilt wird um das Radiosignal zu empfangen. Kommt die Antenne etwas von der Projektorie der Rakete ab, wird das empfangene Radiosignal zu schwach um es zu verarbeiten und es kommt zu einer Unterbrechung der Datenverbindung.
Das Problem bestand nun darin, ein Gerät zu entwickeln mit dem sich ein Test des Experiments und der Bodenstation unter Einbezug der genannten Gegebenheiten – wie zum Beispiel dem Rauschen der Funkstrecke – durchführen ließe. Bislang existieren auf dem Markt keine Testgeräte mit denen sich der Flug einer Rakete simulieren lässt. Anhand von Abbildung 3 lassen sich die essentiellen Bestandteile des Versuchsaufbaus erkennen. Das zu entwickelnde Testgerät simuliert das RXSM und die Funkstrecke. Es wird sich folgend REXUS Service Modul Simulator (RXSMS) nennen.

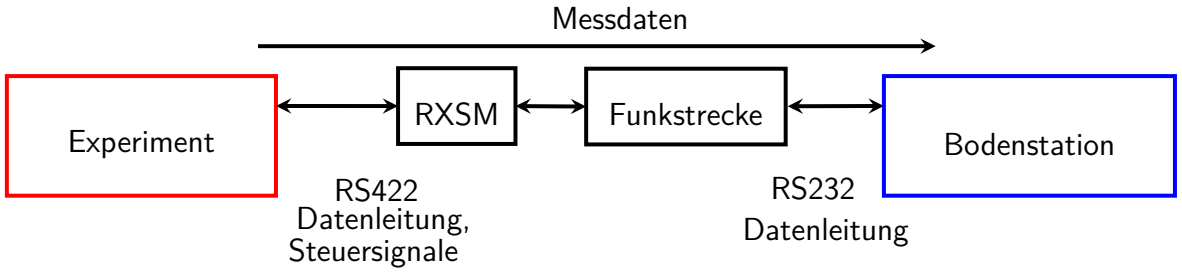

Abbildung 3: Veranschaulichung des Versuchaufbaus ohne berücksichtigung der Antenne

3 Entwicklung des RXSMS

3.1 Entscheidende Grundüberlegungen für die Entwicklung

3.1.1 Analog oder Digital

Die Herausforderung bestand nun darin, eine Schaltung zu entwerfen, die das RXSM und eine rauschende Radioverbindung simulieren kann. Doch zuerst stellt sich die Frage, welcher Entwurf verwendet werden soll. Bei der Entwicklung einer Schaltung muss man sich zwischen einem analogen und einem digitalen Schaltungsentwurf entscheiden. Ein gutes Design zeichnet die Möglichkeit aus, die Schaltung im Nachhinein verändern zu können. Möchte man an einer fertigen analogen Schaltung etwas verändern, muss man zwangsweise direkt in die Hardware eingreifen und Bauteile wie Widerstände oder Kondensatoren wechseln. Dieser Eingriff zieht möglicherweise Lötfehler und somit eine langwierige Fehlersuche nach sich. Falls man ein neues Schaltungskonzept ausprobieren möchte, muss man einen vollkommen neuen Schaltplan entwerfen und eine neue Schaltung aufbauen.
Möchte man hingegen eine digitale Schaltung verändern, reicht es meist aus, die bereits geschriebene Software des Mikrocontrollers abzuändern. Das bedeutet wesentlich weniger Arbeits- und Zeitaufwand, als eine vollkommen neue Schaltung zu entwerfen. Zudem ist die Fehlersuche mithilfe von Debuggern einfacher als die Fehlersuche in einer analogen Schaltungen nach defekten Bauteilen.
Deshalb wurde für dieses Projekt ein Entwurf mit einem Mikrocontroller gewählt. Dieser hat eine zentrale Position in der Schaltung und steuert alle zeitlichen Abläufe.

3.1.2 Simulation des Rauschens und weiterer Übertragungsfehler

Um ein Rauschen simulieren zu können, ist es wichig zu verstehen welche Auswirkungen es auf die Datenverbindung hat. Das Rauschen einer Radioverbindung entsteht durch thermisches Rauschen, Elektro-Smog und durch natürliche kosmische Strahlung. Dabei übelagert sich das elektrische Feld des Nutzsignals und des Rauschsignals und es kommt zur Interferenz, bei der die Daten des Nutzsignals verfälscht werden. Betrachtet man eine digitale Radioverbindung, so bedeutet das, dass während der Übertragung zufällige Bits invertiert werden. Der Empfänger erhält an unbekannter Stelle statt einer digitalen „1“ eine „0“ und statt einer digitalen „0“ eine „1“. Je höher die Zahl dieser Fehler, auch Bitfehler genannt, umso höher ist das Bitfehlerverhältnis.// Lässt man die Beschleunigungsphase der Rakete außer Betracht, wird bei der Funkstrecke zwischen Rakete und der Bodenstation ein Bit-

fehlerverhältnis von ca. 10^{-8} erwartet.
Bei der Simulation von Übertragungsfehlern muss ein weiterer wichtiger Punkt in Betracht gezogen werden. Während des Flugs wird die Rakete von einer Antenne angepeilt, um eine optimale Radioverbindung zu gewährleisten. Doch durch Vibrationen der Antenne und die starke Beschleunigung der Rakete beim Takeoff kommt es zu kurzen Unterbrechungen der Radioverbindung. Die Dauer dieser Unterbrechungen kann von wenigen Millisekunden bis zu zwei Sekunden variieren.
Diese zwei Formen der Übertragungsfehler, das Rauschen und die Unterbrechungen, müssen vom Mikrocontroller des RXSMS simuliert werden. Dabei spielt der Mikrocontroller die Rolle der Radiostrecke. Die Messdaten gelangen vom Experiment in den Mikrocontroller, werden dort verfälscht und anschließend weiter an die Bodenstation geschickt. Auch die Unterbrechungen des Signals lassen sich so nachstellen.

3.1.3 Wichtige Bauteile

Für die Simulation des RXSM wurden dieselben Bauteile wie auf der REXUS Rakete verwendet. Das hat den Zweck das Testen des Experiments möglichst realitätsnah zu gestalten und somit Fehlerquellen zu vermeiden. Bei einem fehlerhaft kalkulierten Stromverbrauch des Experiments lässt sich mit den originalen Bauteilen schnell feststellen, ob dies nicht zur Zerstörung der Bauteile auf der Rakete führt. Um zu erfahren welche Bauteile im Service Modul verwendet werden, wurde beim DLR nach den gewünschten Bauteilnummern gefragt.

Folgende Bauteile sind an der Schnittstelle zwischen Experiment und RXSM verbaut:

- Für die Stromversorgung des Experiments wird ein Highside Power Switch des Typs **BTS6143** von der Firma *Infineon* verwendet.
- Die differentielle RS422 Busverbindung wird mit dem Pegelwandler-Chip **SP490** der Firma *Sipex* bewerkstelligt.
- Die **UDN2596** Treiberstufe der Firma *Allegro* steuert die Signale LO, SOE und SODS

3.1.4 Modellhaftes Layout der Schaltung und die Funktion ihrer Teile

Mithilfe der Abbildung 4 lässt sich die Funktion der verschiedenen Bauteile im RXSMS verdeutlichen. Die Steuerung des µ-Controller besteht aus Tastern und Potentiometern. Die Taster schalten die Steuersignale und die Stromversorgung des Experiments. Mit den Potentiometern lässt sich das Bitfehlerverhältnis – des Rauschen – und die Dauer der Unterbrechungen einstellen. Die LEDs werden verwendet, um eine benutzerfreundliche Übersichtlichkeit zu schaffen, sodass sich auch ohne Bildschirme schnell erkennen lässt, welche Steuersignale geschaltet sind und ob die Datenkommunikation funktioniert. Für die Stromversorgung des Experiments wird ein **BTS6143** Highside Power-Switch verwendet, der bei Tastendruck vom µ-Controller angesteuert wird. Für die Messdatenübertragung wird das differentielle RS422 Signal von dem **SP490** Pegelwandler in ein gewöhnliches Signal umgewandelt. Dieses wird vom Mikrocontroller mit einer zufällig erzeugten Bitmaske XOR überlagert und an einen gewöhnlichen RS232 Pegelwander weitergegeben. Dieser wandelt die Pegel von 3,3 V auf 12 V und ermöglicht so ein Empfangen der Daten am Computer.

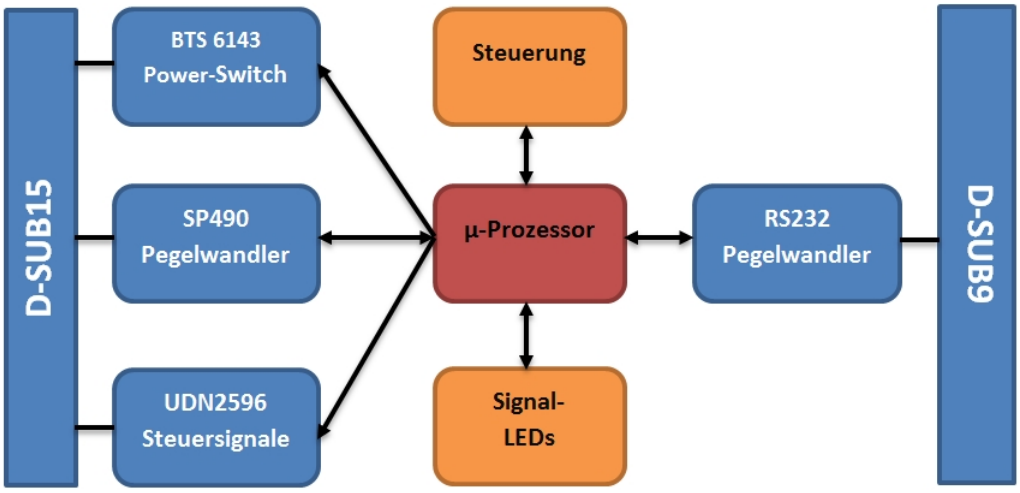

Abbildung 4: Layoutvorlage für den RXSMS

3.1.5 Kostenpunkt und Zeitaufwand

Bei der Entwicklung des RXSMS spielten die Kosten keine entscheidende Rolle. Das Erstellen einer Kostenrechung war nicht nötig. Dennoch wurde bei der Wahl der nicht festgelegten Bauteile auf einen minimalen Preis geachtet. Bei einer wöchentlichen Arbeitszeit von 5 Stunden musste das RXSMS in einem Zeitraum von sechs Monaten entwickelt und fertiggestellt werden.

3.2 Umsetzung der Ideen

3.2.1 Wahl der Bauteile

Um den RXSMS möglichst klein und handlich bauen zu können, wurden SMD-Bauteile verwendet. Der für die Schaltung gewählte Mikrocontroller ist ein **ATXMEGA32A4U** [3] [4] der Firma *Atmel*. Er verfügt über zwei USART's (Universal Synchronous Asynchronous Receiver Transmitter) und genügend Pins, um Taster und Potentiometer anzuschließen. Zudem stellt die Firma *Atmel* eine benutzerfreundliche Programmierumgebung, das Atmel Studio, zur Verfügung. Ergänzend zu den Mikrocontrollern existieren verschiedene Tools zum programmieren und debuggen. Für dieses Projekt wurde ein **AVR-Dragon** mit einem sechs-poligem PDI (Programming-Debugging-Interface) verwendet. Für die Schnittstelle zum Experiment werden die auf der Seite 7 angeführten Bauteile verwendet. Für die Spannungsversorgung des RXSMS wird ein Step-Down Regler des Typs **TSR 1-2450** der Firma *Tracopower* verwendet. Er liefert eine gefilterte Gleichspannung von 5 V am Ausgang. Diese Spannung ist für die Versorgung der ICs **SP490** und **UDN2596**. Die für den Mikrocontroller benötigten 3,3 V Spannung werden von einem **LM1117** Linearregler geliefert. Die Pegelwandlung des RS232 Signals übernimmt ein Standart IC **MAX3232** der Firma *Maxim Integrated*. Für eine angenehme Bedienung wurden Taster mit großer Tastfläche verbaut. Gelbe LEDs sollen anzeigen, welche Steuersignale gesetzt sind. Rote LEDs zeigen einen Datenfluss durch den Mikrocontroller an. Eine grüne LED signalisiert eine einwandfreie Funktion der Stromversorgung des RXSMS.

3.2.2 Bau eines Prototypen

Bei der Entwicklung einer Platine ist es von Bedeutung erst einen Prototypen und dann ein fertiges Release zu bauen. Der Prototyp wird benötigt, um Konzepte auszuprobieren und um festzustellen, ob essentielle Bauteile vergessen worden sind oder mögliche Denkfehler den Entwurf unbrauchbar gemacht haben.[1] Statt des **SP490** Pegelwandlers wurde ein ähnlicher Baustein **MAX3488** verwendet. Der Treiberbaustein für die Steuersignale wurde durch drei **BSS123** MOSFETs imitiert. Da man keine 5 V Spannung für die Versorgung der Bausteine **UDN2596** und **SP490** benötigt, wurde der **TSR 1-2450** durch einen **TSR 1-2433** ersetzt. Für das Schalten der Stromversorgung wurde bereits der Baustein **BTS6143** verlötet. Für das Einstellen des Bitfehlerverhältnisses und der Dauer der

[1] Bei dem Prototypen wurden die vorgegebenen Bauteile nicht verbaut, da die Idee – dieselben Bauteile wie im RXSM zu verwenden – erst während des Projektverlaufs aufkam.

Unterbrechungen wurden provisorisch drei Potentiometer montiert. Wie sich während des Projektverlaufs herausgestellt hat, war dies eine sinnvolle Entscheidung.
Der Schaltplan des Prototypen wurde mit dem Schaltungsdesignprogramm EAGLE angefertigt, wobei auf ein größtmögliches Maß an Übersichtlichkeit geachtet wurde. Für das Erstellen des Schaltplans war das Schema auf Seite 8 von großem Nutzen. Für Bauteile wie Potentiometer, Taster und Power-Switch mussten neue Eagle-Bibliotheken erstellt werden. Beim Layouten der Schaltung auf einer doppelseitigen ½ - Europlatine musste auf eine sinnvolle Platzierung der Bauteile geachtet werden. Bei einer Betrachtung der Platine von oben werden die Bedienelemente, Stecker und Buchsen geordnet um den zentral positionierten Mikrocontroller platziert. Beim Verlegen der Leiterbahnen zwischen den Bauteilen Bauteile mussten rechte Winkel in den Leiterbahnen vermieden werden, da dies bei höheren Signalfrequenzen zu einer erhöhten Impedanz der Leiterbahnen führen kann. Der Schaltplan und das Layout können kann im Anhang auf Seite 23 und 24 und auf er beigelegten DVD angesehen werden. Das Bild des fertigen Prototypen befindet sich ebenfalls im Anhang auf Seite 25.

3.2.3 Software des RXSMS

Wichtige Überlegungen Die Software für den **ATXMEGA34A4U** musste so geschrieben werden, dass der RXSMS dieselben Möglichkeiten bietet wie das echte RXSM. Vor dem Takeoff ist eine bidirektionale Verbindung zwischen dem Experiment und der Bodenstation möglich. Das ist nötig, damit die Bodenstation vor dem Takeoff dem Experiment Kontrollsequenzen zuschicken kann, um es vor dem Flug noch ein letztes mal zu initialisieren. Nach dem Takeoff besteht nur eine unidirektionale Radioverbindung, sodass nur das Experiment Datenpakete auf die Erde schicken kann. Eine Verbindung von der Bodenstation in Richtung Rakete ist nicht mehr möglich. Für die Programmierung der Software bedeutet das, dass beim Drücken des LO-Tasters (Lift-Off) die Datenkommunikation von der Bodenstation in Richtung Experiment unterbrochen werden muss. Der zeitliche Einsatz der Steuersignale (Start of Experiment) und SODS (Start of Data Storage) ist von jedem Studententeam für deren Experiment individuell zu bestimmen. Diese Tatsache ist insofern von Bedeutung, dass die Signale vollkommen unabhängig voneinander implementiert werden müssen.

Aufgaben des ATXMEGA32A4U Die wichtigste Aufgabe des Mikrocontrollers ist das Weiterleiten der Daten. Er empfängt das Byte im Empfangsregister einer USART-

Schnittstelle und schreibt anschließend das Byte in das Senderegister der anderen USART-Schnittstelle. Dazwischen müssen noch zufällige Bits invertiert werden. Bei einem beispielhaften Bitfehlerverhältnis von 10^{-8} wird jedes hundertmillionste Bit umgekehrt.
Des weiteren muss der Datenfluss an zufälliger Stelle für eine variable Zeit unterbrochen werden können. Weitere Aufgaben des Mikrocontrollers sind das Auslesen der Taster und der Potentiometer und das dementsprechende Setzen der LEDs und der Steuersignale, das Einstellen des Bitfeherverhältnisses als auch das ändern der Unterbrechungsdauer.

Implementierung der Aufgaben Die Aufgaben des Mikrocontrollers werden mithilfe vom Atmel Studio in der Programmiersprache *C* [5] implementiert. Ein Programm für einen Mikrocontroller besteht aus der Hauptschleife und den Interrupts. Ein Interrupt, wie der Begriff es bereits sagt, unterbricht die Hauptschleife und lässt eine Programmsequenz ablaufen. Danach springt das Programm an die vorher unterbrochene Stelle zurück und fährt mit der Hauptschleife fort. Die Programmsequenz im Interrupt wird zeitunabhängig von anderen Prozessen durch den „Interrupt-Handler“ aufgerufen und abgearbeitet. Das verschafft dieser Programmsequenz die zeitlich höchste Priorität. Wird der Interrupt von einer Interrupt-Quelle, zum Beispiel von der USART-Einheit aufgerufen, wird somit eine festgelegte Programmsequenz abgearbeitet. Die Familie der XMEGA-Controller verfügt über ein dreischichtiges Interrupt-Leveling System, um verschiedenen Interrupts Prioritäten zuzuweisen. Bei einem Aufruf eines Interrupt höchster Priorität, während ein Interrupt niedriger Priorität eine Programmsequenz abgearbeitet wird, wird der Interrupt niedriger Priorität unterbrochen, der höher priorisierte Interrupt abgehandelt, und anschließend das Programm des niedriger priorisierten Interrupt fertiggestellt. Mit einem solchen System lassen sich die wichtigsten Aufgaben des µ-Controllers nach Priorität auf die verschiedenen Interrupt-Schichten, auch Interrupt-Level genannt, verteilen. Zeitunabhängige Aufgaben hingegen werden in der Hauptschleife ausgeführt. Die Weiterleitung der Daten wird dem höchsten Interrupt-Level zugeteilt. Das Programm dieses Interrupts hat die Funktion, die vom Experiment kommenden Daten mit einer einstellbaren Bitfehlerrate an die Bodenstation weiterzuleiten. Eine Voraussetzung für eine fehlerfreie Kommunikation ist, dass die Pins, an denen die seriellen Signale angeschlossen sind, korrekt als Ein- und Ausgänge definiert sind. Wie in der Schaltung des Prototypen auf Seite 23 zu erkennen ist, hat das Experiment-USART die Ein- und Ausgänge am PORTD an PIN2 und PIN3 und das Bodenstation-USART am PORTE an PIN2 und PIN3. Außerdem müssen beide USART Schnittstellen vor dem Betrieb mit folgenden Informationen initialisiert werden:

- Geschwindigkeit, **Baudrate** der Datenleitung: *38,4 kbit/sec*
- **Datenpaketgröße**: *8 Bits*
- Zahl der **Stopbits**: *1 Stopbit*
- Keine Prüfbits: *no parity*

Da die Baudraten beider USART-Verbindungen identisch sind, werden keine Kontrollalgorithmen benötigt um Verklemmungen der Datenkommunikation zu vermeiden, da die Daten in den Mikrocontroller genauso schnell hinein, wie auch herauskommen. Der Bitfehler wird erzeugt, indem man eine vom Zufallsgenerator [6, Seite 3] erzeugte Zahl mit dem Ergebnisraum von $0–2^{32}$ mit dem Spannungswert des ersten Potentiometers [Poti 1, Abbildung 5] vergleicht. Der Zahlenbereich des Potentiometers umfasst dabei Werte von $0–2^{14}$. Ist der Stellwert des Drehreglers höher als der Betrag der Zufallszahl, wird das Bit mit einer XOR Maske überlagert, und somit invertiert. Ist der Stellwert des Spannungsreglers kleiner als der Betrag der Zufallszahl, wird das Bit ungehindert durchgelassen. Je nach Verhältnis zwischen dem Stellwert des Potentiometers und der Dimension der Zufallszahl lässt sich eine Bitfeherwahrscheinlichkeit[2] einstellen. Bei den konkreten Werten liegt die Bitfehlerwahrscheinlichkeit zwischen 10^{-10} und ca. $3 \cdot 10^{-6}$.

Für die Unterbrechungen der Datenkommunikation wird ein Timer verwendet, der nach

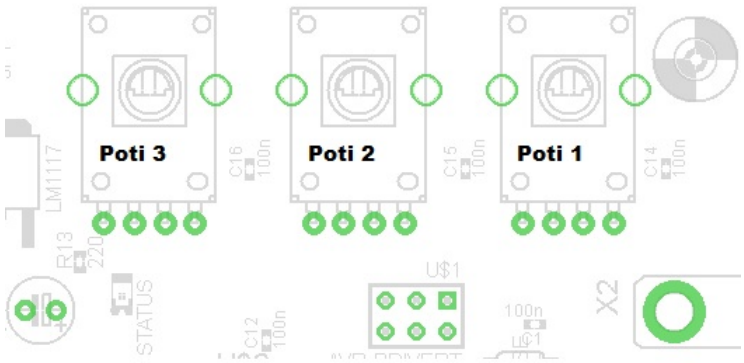

Abbildung 5: Nummerierung der Potentiometer

seiner Aktivierung den Datenfluss für eine einstellbare Zeit sperrt. Die Dauer der Unterbrechung lässt sich mithilfe des zweiten Potentiometers [Poti 2, Abbildung 5] von 200 *ms*

[2]Eine bestimmte Bitfehlerwahrscheinlichkeit hat zwangsweise ein bestimmtes Bitfehlerverhältnis zur Folge

bis 2 *s* einstellen. Um den Zeitpunkt, an dem der Datenfluss unterbrochen wird zufällig zu wählen, wird die Zahl der Bitfehler in einer Variablen gespeichert. Sobald die Variable den Stellwert des dritten Potentiometers [Poti 3, Abbildung 5] übersteigt, wird die Unterbrechung ausgelöst. Der Zeitpunkt der Unterbrechungen hängt direkt mit dem Zufall der Bitfehler zusammen und ist somit zufällig.
Für ein Auslesen der Potentiometer wird der ADC (Analog-Digital Converter) des Controllers verwendet. Ein ADC transformiert einen analogen Spannungswert vom Potentiometer in einen digitalen Zahlenwert, der als Vergleichsvariable in den oben genannten Operationen verwendet wird. Mithilfe einer linearen Transformation lässt sich der Zahlenwert des Potentiometers auf größere Zahlenbereiche erweitern. Da das Auslesen der Potentiometer nicht zeitkritisch ist, wird dieser Prozess in der Hauptschleife behandelt. Um die Taster auslesen zu können müssen im Mikrocontroller sogenannte „Pull-Up Widerstände“ aktiviert werden, damit bei Tastendruck eine Potentialänderung am Eingang des Mikrocontrollers eine fallende Flanke auslöst. Speziell für solche Zwecke hat der Controller einen Flankendetektor an jedem Pin. Auf den Tastendruck folgend löst der Flankendetektor einen Interrupt aus, der den Zustand der Taster ausliest und dementsprechend die Ausgänge für die Steuersignale, die Stromversorgung und die Status-LEDs setzt. Im Programmablaufplan auf Seite 14 und Seite 15 lassen sich die Zusammenhänge der genannten Schritte erkennen.

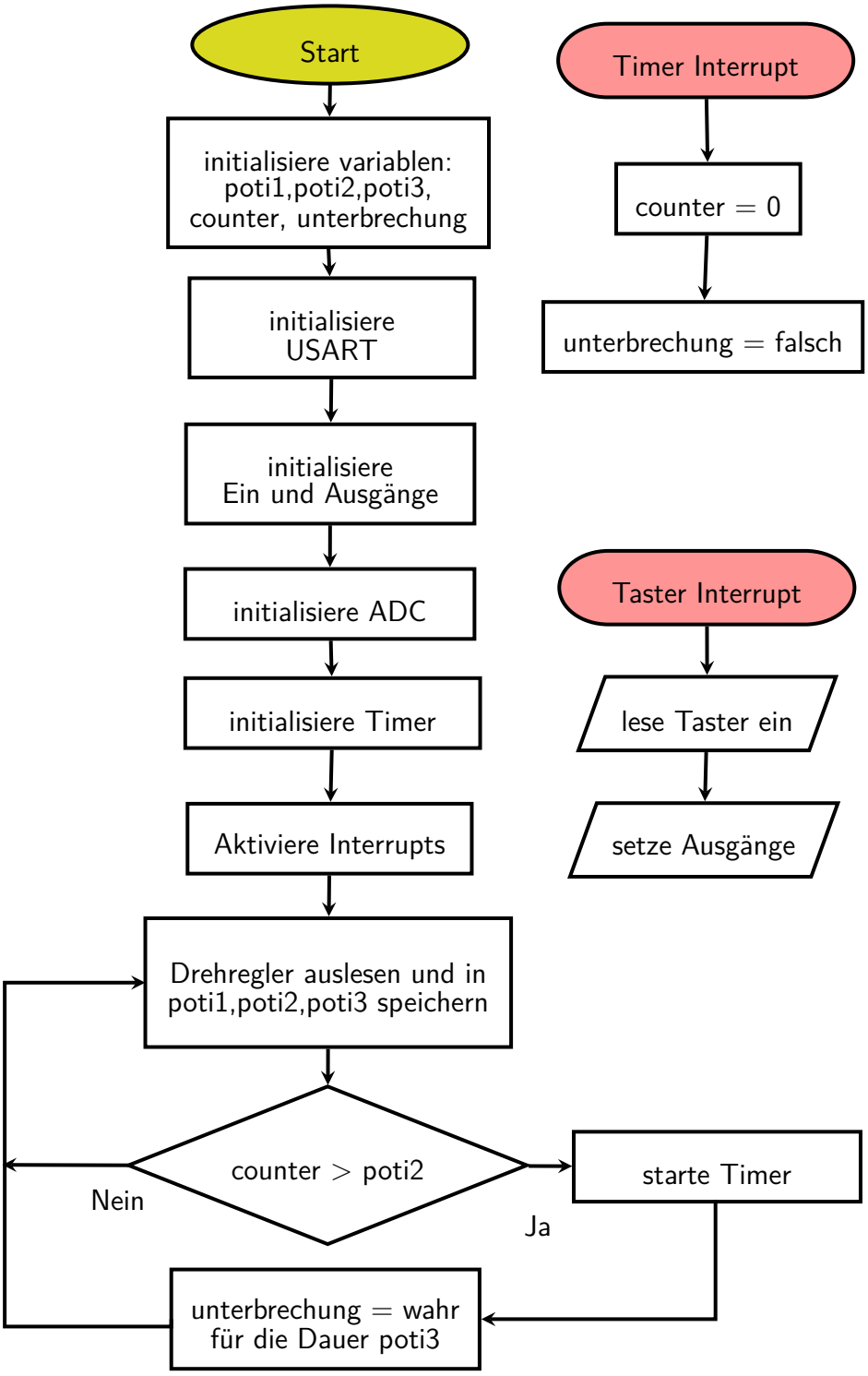

Abbildung 6: Hauptprogramm mit Timer-und Taster-Interrupts

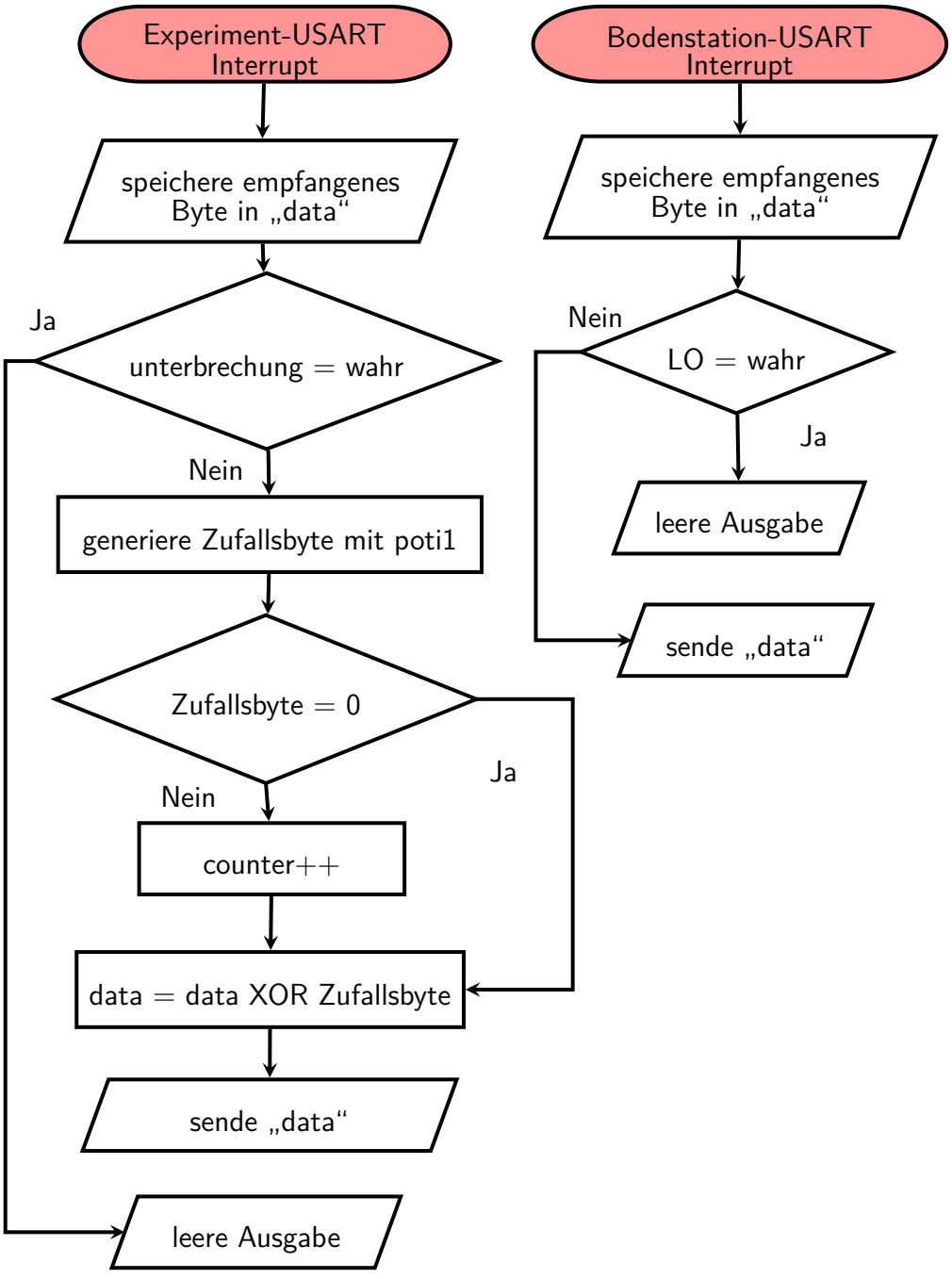

Abbildung 7: USART-Interrupts

3.2.4 Test des RXSMS-Prototypen

Bevor der Prototyp eingesetzt werden konnte, wurde er auf die wichtigsten Funktionen überprüft. Die active-low Steuersignale wurden mit einem Pull-Up Widerstand und einem Oszilloskop kontrolliert. An die Stromversorgung des Experiments, welche vom **BTS6143** gesteuert wird, wurde ein Leistungswiderstand angeschlossen. Die Weiterleitung der Daten wurde mit einer zusätzlichen Platine getestet, welche das differentielle Signal des RXSMS in ein gewöhnliches Signal umgewandelt und an einer DSUB-9 Buchse ausgibt. Mit dem Test der Funktionen der Schaltung geht automatisch ein Test der Bauteile einher, da die Schaltung ohne intakte Bauteile nicht arbeiten würde.
Nach einem Test mit dem Oszilloskop wurde der Prototyp beim Test FOVS-Projekt eingesetzt. Das Studentenprojekt FOVS „Fiber-Optic Vibration Experiment“ überprüft die Funktionsfähigkeit glasfaseroptischer Messsysteme im Einsatz auf einer Rakete. Dafür werden faseroptische Vibrationssensoren gebaut und getestet. Für das Auslesen der Sensoren wurde eine Platine mit Mikrocontroller gebaut und erfolgreich mit dem RXSMS getestet. Bei dem Test wurden Verbesserungsmöglichkeiten für das RXSMS entdeckt, die unter dem Stichpunkt „Optimierungen“ erläutert werden.

3.2.5 Änderungen vom Prototypen zum Release

Einbau der vom DLR vorgegebenen Bauteile Die wichtigste Änderung vom Prototypen zum Release ist die Verwendung der Bauteile wie im RXSMS. Mit der Verwendung dieser Bauteile entsteht eine weitere Aufgabe: Für die Spannungsversorgung des µ-Controllers benötigt man 3,3 V und für die Versorgung des Pegelwandlers **SP490** und der Treiberstufe **UDN2596** werden 5 V benötigt. Das bedeutet wiederum, dass der Signalpegel von 3,3 V des Mikrocontrollers nicht ausreicht, um die Eingänge des differenziellen Pegelwandlers zu steuern. Diese Aufgabe lässt sich durch eine gestaffelte Stromversorgung, wie in Abbildung 8, und zusätzliche Pegelwandler zwischen den Bauteilen lösen. Die Pegelwandler zwischen dem µ-Controller und dem **SP490** bestehen aus **BSS123** MOSFETs in Gate-Schaltung [7]. Für die Steuerung des **UDN2596** werden keine Pegelwandler benötigt, sondern Djembes, da die Eingänge active-low sind und der erste Transistor im Baustein einen Kollektorwiderstand hat. Somit reicht es aus, die Ausgänge des Mikrocontrollers direkt an die Eingänge des Treibers zu schalten.

Optimierungen Beim Betrieb des RXSMS Prototypen haben sich folgende Optimierungsideen entwickelt: Ein weiterer Taster wird benötig, der bei Betätigung eine nterbre-

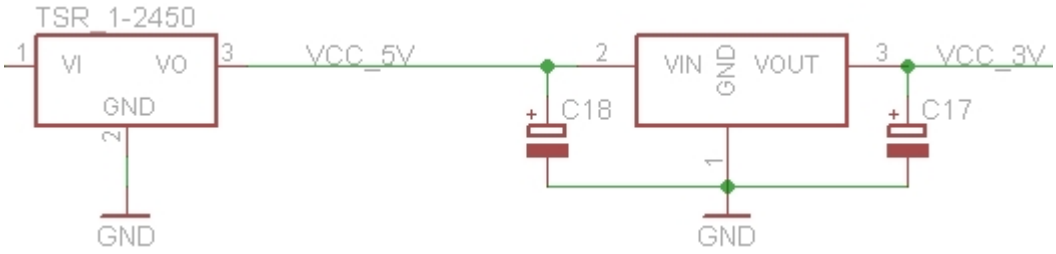

Abbildung 8: Stromversorgung des RXSMS

chungsfreie Weiterleitung der Daten gewährleistet. Außerdem wird auf der Platine mehr Platz zwischen den Potentiometern gebraucht, um Drehknöpfe für eine angenehmere Bedienung aufstecken zu können. Eine weitere Verbesserung ist eine für die Software einfachere Verkabelung der Taster, der Steuerausgänge und der Status LEDs mit dem µ-Controller. Eine weitere Optimierung ist die Verwendung größerer Vorwiderstände für die LED, da diese beim Betrieb des Prototypen sehr hell leuchteten, sodass anwesende Personen beim Blick auf die LED geblendet wurden. Desweiteren wurde beim Layouten und verlöten des Prototypen der Fehler gemacht, statt einer „DSUB-9 female“ eine „DSUB-9 male“ zu verwenden. Das hatte zur Folge, dass beim Betrieb des Prototypen ein zusätzlicher Adapter gebraucht wurde, um eine Datenverbindung zur Bodenstation herstellen zu können. Außerdem wurden für die Verankerungen dieser Buchsen an der Platine keine Lötstellen vorgesehen.

Schaltplan und Layout des finalen RXSMS Mit den genannten Veränderungsideen wurde der Schaltplan und das Layout des RXSMS mit dem Programm EAGLE neu entworfen. Beide Abbildungen lassen auf Seite 26 und Seite 27, als auch auf der beigelegten DVD ansehen. Die Platine des finalen RXSMS wurde bereits angefertigt und verlötet.

4 Zusammenfassung

Aktueller Projektstatus Ein bereits erledigter Schritt in der Entwicklung des RXSMS ist das Designen, Layouten, Löten, Programmieren und das erfolgreiche Testen eines Prototyps. Für das finale RXSMS ist das Layout bereits erstellt und die Platine verlötet worden. Die Software des Prototypen muss für das Release noch vereinfacht werden, da die Verdrahtung der Taster und der LEDs mit dem Mikrocontroller verändert wurde.

Ausblick Die nächsten Schritte in der Entwicklung des RXSMS belaufen sich auf das Umprogrammieren der Software und dem Einbau der Platine in ein Gehäuse. Da diese Schaltung zum Testen zukünftiger REXUS Projekte benötigt wird, werden etwa zehn Exemplare davon angefertigt und an die am REXUS Projekt teilnehmenden Studententeams verschickt. Zudem wird geplant, das gesamte Projekt „REXUS-Service Module Simulator“ unter der GNU GPL-Lizenz zu veröffentlichen.

Danksagung

Im Folgenden möchte ich mich bei meinen Projektbertreuern Thomas Grübler und Max Rößner herzlichst für die Ratschläge bei der Arbeit und die mentale Unterstützung bedanken. Außerdem bedanke ich mich beim Otto-von-Taube-Gymnasium und bei der Technischen Universität München, ohne deren Kooperation eine solche Arbeit nicht zustandegekommen wäre.

Selbstständigkeitserklärung

Hiermit erkläre ich, daß die vorliegende Arbeit selbstständig angefertigt, und ohne fremde Hilfe von mir Verfasst worden ist. Sämtliche wissentlich verwendete Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.

Literatur

[1] DLR. REXUS auf der Startrampe. `http://www.dlr.de/DesktopDefault.aspx/tabid-4530/3681_read-9269/3681_page-5/gallery-1/gallery_read-Image.1.3523/`, 2007. zuletzt aufgerufen: 27.02.2014.

[2] M. Fittock and M.Inga. REXUS User Manual. `http://www.rexusbexus.net/images/stories/rexus/RX_REF_user_manual_v7-7_06Sep12.pdf`, 2012. zuletzt aufgerufen: 02.01.2014.

[3] Atmel Corporation. *XMEGA A MANUAL*, 2012.

[4] Atmel Corporation. *8/16-bit Atmel XMEGA Microcontroller*, 2012.

[5] Jürgen Wolf. *C von A bis Z*. Galileo Computing, 2009. wurde für das Erlernen der Programmiersprache C verwendet, klare Seitenzahlen nicht möglich.

[6] Tobias Litte. Zufallsgeneratoren. 2004,2005. Seite 3.

[7] D.Holmes. A simple and elegant solution using a MOSFET to level-shift control and bi-directional data lines between circuits running at different voltages. `http://delphys.net/d.holmes/hardware/levelshift.html`, 2001. zuletzt aufgerufen: 02.01.2014.

Abbildungsverzeichnis

Abkürzungen

ADC Analog-Digital Converter

DLR Deutsches Zentrum für Luft- und Raumfahrt

IC Integrated Circuit

MST Lehrstuhl für Messsysteme- und Sensortechnik

PDI Programming-Debugging-Interface

Poti(ugs.) Potentiometer

REXUS Rocket Experiments for University Students

RXSM REXUS Service Modul

RXSMS REXUS Service Modul Simulator

SMD Surface mounted Device

SNSB Swedish National Space Board

USART Universal Synchronous Asynchronous Receiver Transmitter

A Anhang

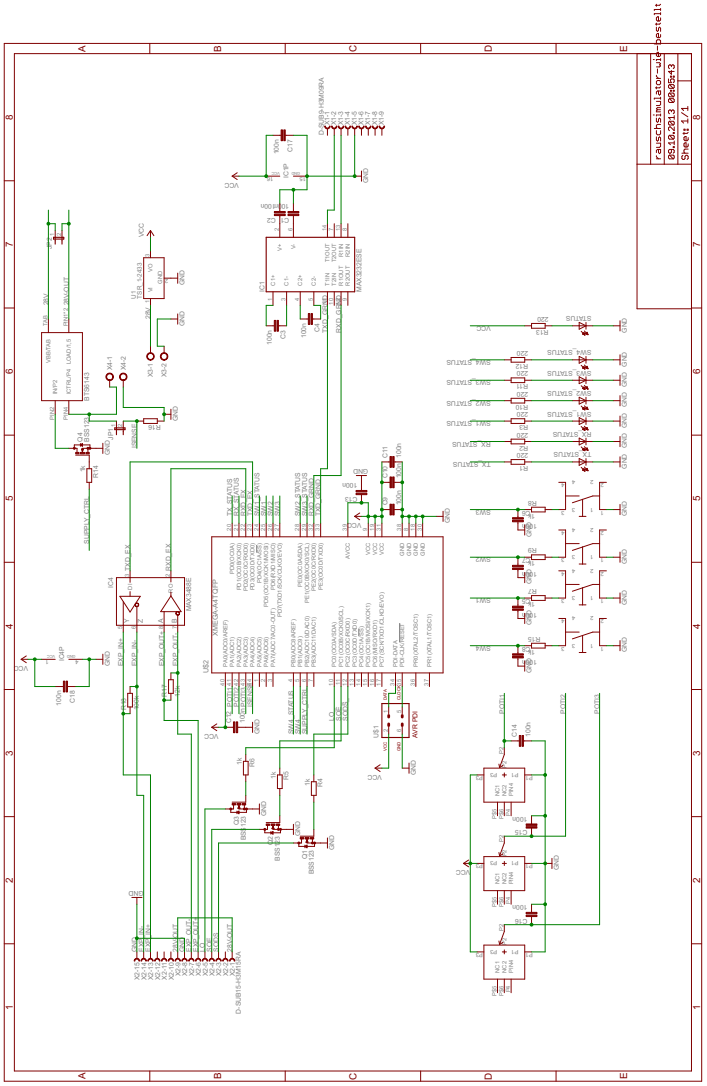

Abbildung 9: Schaltplan des Prototypen

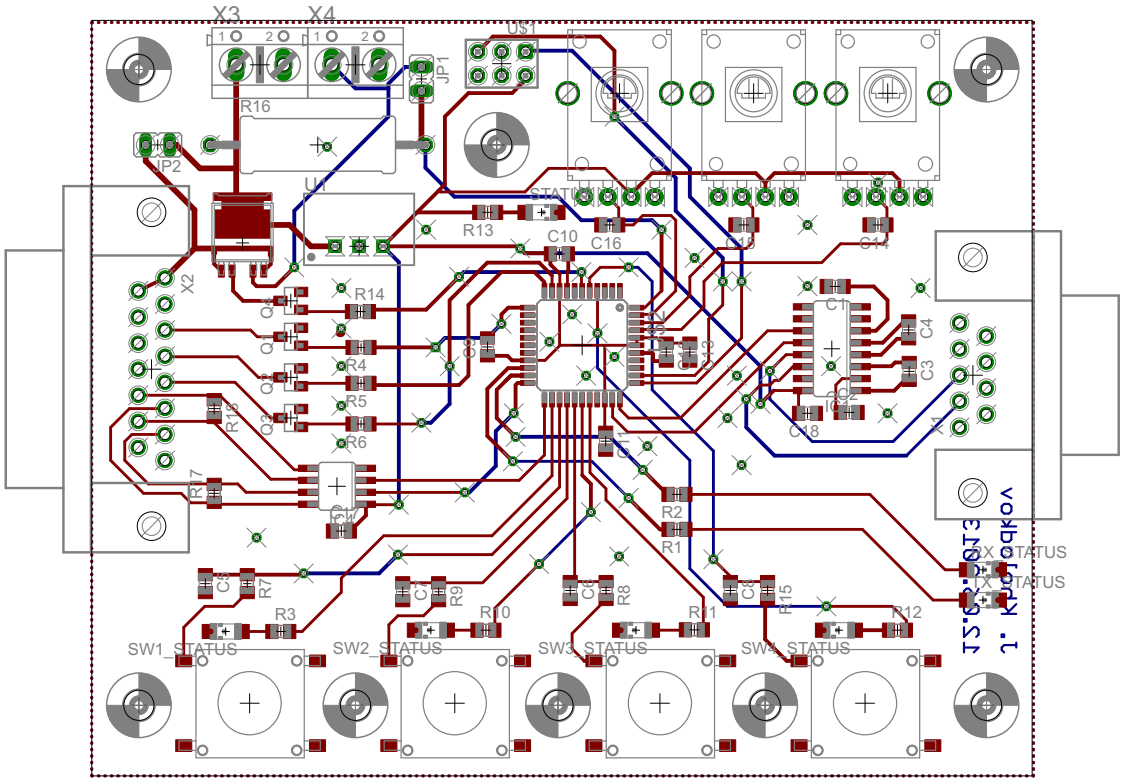

Abbildung 10: Layout des Prototypen

Abbildung 11: Prototyp des RXSMS im Einsatz

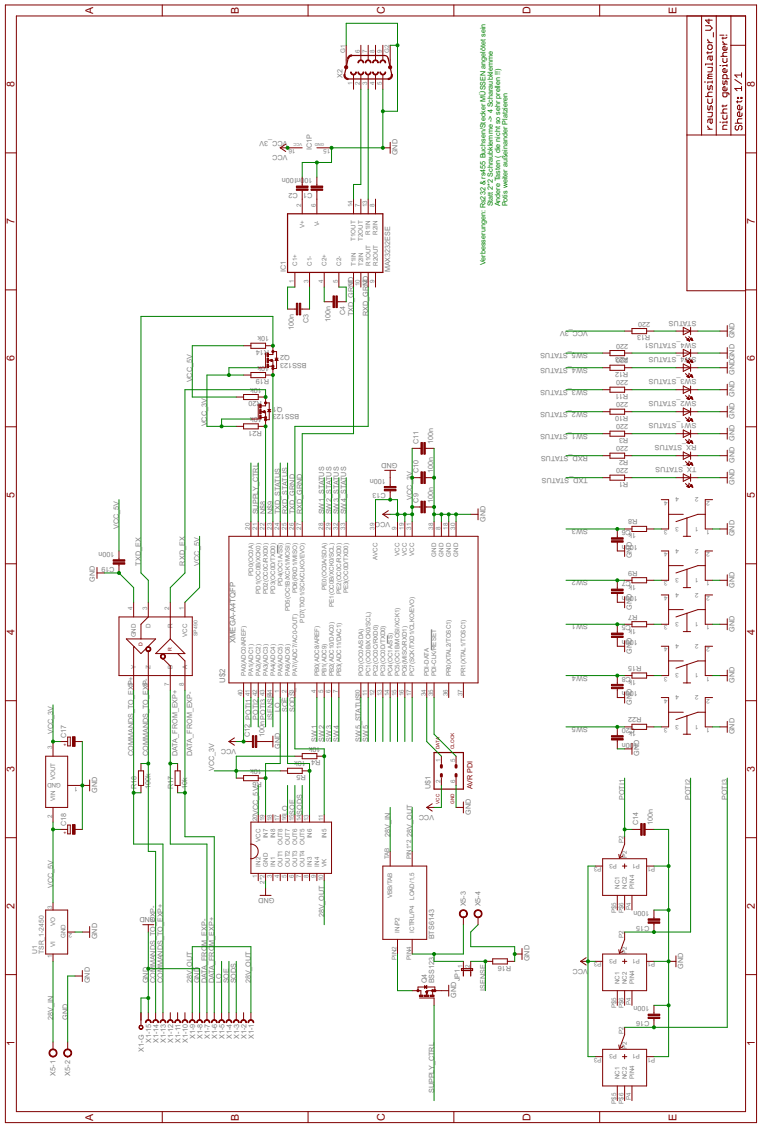

Abbildung 12: Schaltplan des finalen RXSMS

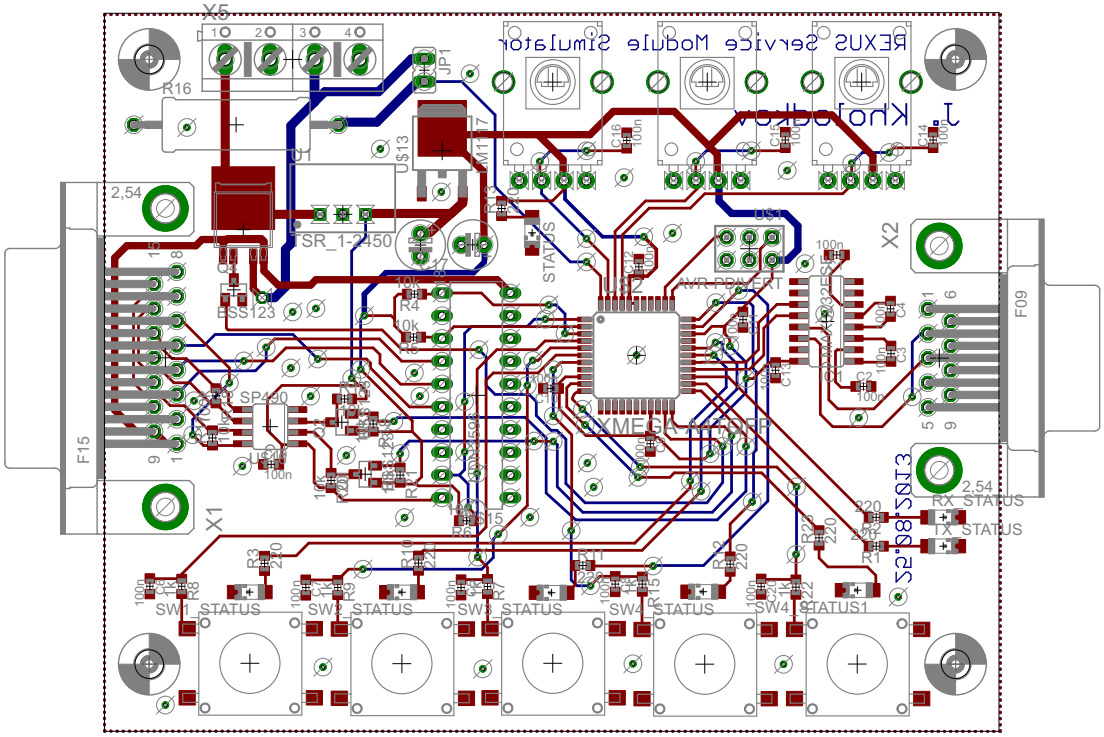

Abbildung 13: Layout des finalen RXSMS