Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Michael Szyszylo

Entwicklung eines vielseitig einsetzbaren Bluetooth-Low-Energy Kommunikationsmoduls

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Michael Szyszylo

Entwicklung eines vielseitig einsetzbaren Bluetooth-Low-Energy Kommunikationsmoduls

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Informations- und Elektrotechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.-Ing. Ralf Wendel
Zweitgutachter : Dipl.Ing. Cornelius Jung

Abgegeben am 6. Februar 2017

Michael Szyszylo

Thema der Bachelorthesis

Entwicklung eines vielseitig einsetzbaren Bluetooth-Low-Energy Kommunikationsmoduls

Stichworte

Bluetooth-Low-Energy, BLE, Universal Asynchronous Receiver Transmitter, UART, Serial Peripheral Interface, SPI, Inter–Integrated Circuit, I^2C

Kurzzusammenfassung

Diese Arbeit befasst sich mit der Entwicklung einer kostengünstigen und energieeffizienten Kommunikationsplatine. Hierfür wird die elektronische Schaltung konzipiert. Außerdem wird die dazugehörige Steuersoftware, für die intern agierenden Protokolle (UART, SPI, I^2C) geschrieben, getestet und ihre finale Funktionalität im Zusammenhang mit einer Android-Applikation und einem Zigarettenautomaten erprobt.

Michael Szyszylo

Title of the paper

Development of a multifunctional applicable Bluetooth-Low-Energy communication module

Keywords

Bluetooth-Low-Energy, BLE, Universal Asynchronous Receiver Transmitter, UART, Serial Peripheral Interface, SPI, Inter–Integrated Circuit, I^2C

Abstract

This assignment deals with the development of an economic and energy efficient communicative circuit board. For this purpose, the electronic circuit needs to be planned. Furthermore, the corresponding control software is to be written and tested for the internally acting protocols (UART, SPI, I^2C) and the final functionality in correspondance to an android-application and a cigarette vending machine will be field-tested.

Danksagung

Ich danke vor allem Herrn Dipl.Ing. Cornelius Jung für die tolle Unterstützung und Hilfsbereitschaft bei der Entwicklung dieser Arbeit, sowohl in Hard- als auch Software und täglichen herumschlagenden Lappalien. Es gab keinerlei Fälle, in denen mir nicht aus der Patsche geholfen wurde. Weiterhin mÃ¶chte ich dem Geschäftsführer der Firma Scholz System, Herrn Sebastian Scholz, für die technische als auch verbal freudige Begleitung dieser vergangenen Monate danken. Ebenfalls gedient mein Dank Prof. Dr.-Ing. Ralf Wendel, für die freudige Annahme dieses etwas ungewöhnlicheren Themas einer Bachelorarbeit. Und letztlich meiner Freundin, Sandra G., für die ständige tatkräftige Motivation und finale Korrekturlesung dieses Werkes.

Inhaltsverzeichnis

Hinweise

Aus strukturellen Gründen wird in dieser firmen-intern betreuten Arbeit ab der Beschreibung der Software auf Seite 63 nur eine finale Implementierung in einen Zigarettenautomaten beschrieben, um sich explizit auf ein verwendbares Kommunikationsprotokoll konzentrieren und eine spezifisch dazu entworfene Android-Applikation nutzen zu können.

Fachbegriffe wurden, wenn es sinnvoll erschien, ins Deutsche übertragen. Wurde der Begriff aus der englischsprachigen Fachliteratur übernommen oder war die deutsche Übersetzung eines Fachbegriffes unüblich, wurden die Begriffe in der Originalsprache belassen. Eigennamen oder Fachbegriffe sind ggf. in der Arbeit kursiv gesetzt.

Verwendete Programme:

- *Cadsoft EAGL*E Version 7.6.0 Professional Edition
- *yEd* Version 3.16.2.1
- *Microsoft Visual Studio Community 2015* Version 14.0.25431.01 Update 3
- *Atmel Studio 7* Version 7.0.1188

Der bezüglich dieser Arbeit produzierte Quellcode sowie die hier verwendeten Datenblätter sind bei Herrn Prof. Dr. Ing. Ralf Wendel an der Hochschule für Angewandte Wissenschaften in Hamburg in Form einer DVD hinterlegt.

1 Einführung

Der Alltag eines nahezu jeden Menschen wird stets von einem Smartphone begleitet, ob bei der Arbeit, zu Hause oder gar aktiv, im sich bewegenden Zustand auf der Straße. Dieses Etablissement der heutigen Zeit ist der endlichen Nutzung in bewohnten Zonen, der allzeitigen Erreichbarkeit per E-Mail, SMS oder einer universell benutzten Kommunikationsapplikation, durch das Internet, zu verdanken. Ein weiterer zu erwähnender Aspekt ist der Klimawandel, den die Menschheit seit geraumer Zeit nicht außer Acht lassen kann. Neu entstehende Produkte könnten seit dieser Zeit als rudimentär bezeichnet werden oder würden gar nicht erst in den Handel gelangen, wenn sie nicht die definierten Betriebsnormen, wie z. B. für den Energieverbrauch, einhalten.

Inspiziert man die verfügbaren Funkstandards eines Smartphones und filtert die wichtigsten heraus, dann erhält man, unter anderem, das klassische Bluetooth sowie Bluetooth-Low-Energy (BLE, Bluetooth Smart). Letzteres ist hierbei für die Zwecke dieser Arbeit das Brauchbarere, welches gerade bei batteriebetriebenen Geräten Vorteile bietet, wenn es weniger auf Reichweite und hohe Datenraten, als vielmehr auf lange Akkulaufzeiten mit geringem Stromverbrauch ankommt.

Die Firma Scholz System in Trittau, bei der ich diese Arbeit tätige, entwickelt und produziert täglich für u. a. führende Zigarettenautomaten-Hersteller die dortig verwendeten Dokumentenleser zur Altersverifikation. Diese Zigarettenautomaten besitzen für die heutige Zeit eine zu monotone Bedienbarkeit und sollen zukünftig eine optionale Machbarkeit eines Kaufabschlusses ermöglichen. Dazu bedient die Firma einen führenden Kaffeemühlen-Hersteller, für den sie Steuerplatinen entwickelt und durch enge Korrespondenz auch hier an optional möglichen Änderungen der Bedienbarkeit arbeitet.

Aus diesen Gegebenheiten entstand die Idee und darauffolgende Entwicklung eines kostengünstigen Kommunikationsmoduls, welches als Erweiterung in möglichst viele bestehende u. a. o. g. Geräte eingebaut werden kann und diese befähigt, mit einem Smartphone sehr kleine Datenmengen austauschen zu können. Der Austausch soll hierbei bidirektional über UART als auch SPI möglich sein.

2 Hardware-Analyse

2.1 Theoretische Überlegungen der Kernaufgabe

Betrachtet man z. B. eine in der Gastronomie stehende Kaffeemühle genauer, dann erkennt man die ausreichend vorhandene Bedienbarkeit. Eine Taste für den Mahlvorgang, zwei für die Wahl eines Single- oder Double-Espressos und ggf. Potentiometer zur Einstellung der Mahlzeiten. Wozu also das Alles erweitern wollen? – Geht man von einem praktischen Beispiel aus, bei dem ein leitender Gastronomie-Besitzer Interesse an jährlichen oder sogar täglichen Einsichten seiner Kaffeemühle habe, dann wäre dies mit einer dafür spezifisch entworfenen Smartphone-Applikation möglich. Diese benötigt eine Anlaufstelle, nämlich die in dieser Arbeit zu entwerfende Bluetooth-Low-Energy-Platine. Aus diesem Beispiel schließt man den benötigten Einbau einer kleinen Speichereinheit, um z. B. die von den Nutzern bzw. Bedienern getätigten Anfragen statistisch abspeichern zu können. Eine nicht unwesentliche Information ist die fehlende Zeiteinheit in solchen Kaffeemühlen. Möglicherweise schien die anzuzeigende Uhrzeit irrelevant und die Speicherung von Betriebsdaten nicht nötig. Die Speichereinheit wird damit mittels laufender Real-Time-Clock (RTC), welche ohne kontinuierliche Spannungsversorgung ihre aktuellen Werte verliert, daraufhin auch Zeitangaben verwenden können. Dadurch wird man um den Einbau einer zusätzlichen Batterie nicht herum kommen. Dem Kern wird ein für diese technische Aufgabe ausreichender 8-Bit Mikrocontroller beigefügt, um alle gegenwärtigen Seriellen-Schnittstellen zu unterstützen. Dazu mehr im Kapitel 2.4, Seite 19. Letztere Komponente, ohne der eine Datenübertragung über kurze Distanz nicht möglich wäre, ist ein BLE-Modul, welches bestmöglich die aktuelle Bluetooth Spezifikation 4.2 beinhaltet.

2.2 Relevante Grundlagen

Im Folgenden wird das grundsätzlich genutzte Komponentenverhalten erläutert, so wie ggf. relevante mathematische Spezifikationen dazu erwähnt, da diese ausreichen, um ein Verständnis für die Funktionsweise der zu erstellenden Platine zu erlangen.

2.2.1 Feldeffekttransistor

Transistoren sind Halbleiterbauelemente, die integriert, oder auch einzeln, in elektronischen Schaltungen Anwendung finden. Der Feldeffekttransistor (Fet) besitzt wie der Bipolartransistor drei (*vier*) Anschlüsse: Gate (G), Source (S) und Drain (D), wobei es dazu noch einen zusätzlichen vierten Anschluss gibt, den Substrat (Bulk, B) welcher aus dem Halbleitergrundmaterial besteht. Beide Halbleiterbauelemente gibt es für verschiedene Anwendungsfälle, z. B. Einzeltransistoren, die zur Montage auf eine Leiterplatine genutzt werden, bei denen der Bulk mit Source verbunden ist, da ein Potenzialunterschied zwischen diesen die Eigenschaften des Transistors negativ beeinflusst (body effect). Das *Funktionsprinzip* eines Fet's erklärt sich so, dass eine an Gate und Source anliegende Steuerspannung die Leitfähigkeit der Drain-Source-Strecke beeinflusst und dabei ein intern aufgebautes elektrisches Feld verändert wird, wobei kein Steuerstrom fließt und somit leistungslos gesteuert wird.

Leistungslosigkeit ist der wesentliche Unterschied zwischen einem bipolaren Transistoren, dessen Ausgangsgröße durch den Basisstrom gesteuert wird, und eines Fet's, bei dem diese hingegen durch ein elektrisches Feld (d. h. einer Spannung) gesteuert wird.

Der Mosfet kann dadurch beschrieben werden, dass die Steuerspannung beide Polaritäten annehmen kann, ohne, dass Strom fließt und mindestens immer einer der pn-Übergänge zwischen Source und Bulk bzw. Drain und Bulk gesperrt ist, woraus man sogenannte *selbstleitende* oder *selbstsperrende* Mosfets erhält. Eine Annahme beider Polaritäten ist möglich, da eine Oxid-Schicht (SiO_2), siehe Abb. 2.1, das Gate vom Kanal isoliert [18] [5] [14].

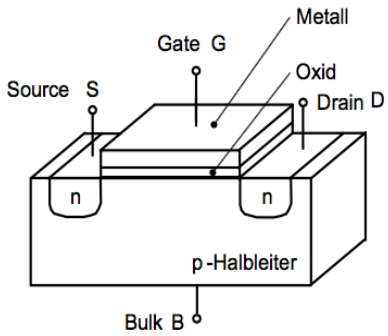


Abbildung 2.1: Schnittbild eines n-Kanal MOS-Feldeffekttransistors [5]

In der Praxis werden n-Kanal Mosfets am häufigsten verwendet, da diese mit positiver Spannung arbeiten und in etwa eine doppelt so schnelle Schaltzeit als p-Kanal Typen aufweisen. Dies begründet sich durch die doppelt so schnelle Beweglichkeit der Elektronen gegensätzlich der Löcher[1] im Silizium, wodurch entsprechend ein 2-3 Mal größerer Strom geliefert wird.

Ein z. B. *selbstsperrender* n-Kanal Mosfet besitzt bei einer anliegenden Gate-Source-Spannung (U_{GS}) von 0V keinen Drainstrom (I_D). Erst durch das Anlegen einer ausreichend großen Spannung bildet sich unter dem Gate ein leitender Kanal, der Source und Drain miteinander verbindet. Die Spannungen U_{GS}, U_{th}, sowie die Drain-Source-Spannung (U_{DS}) und der Drainstrom I_D, besitzen für den Normalbetrieb die auf Abb. 2.2 definierten Polaritäten. Der Drainstrom bestimmt sich durch die Ladung, die sich pro Zeiteinheit durch den Kanal bewegt. P- zu n-Kanal unterscheidet sich bei Strömen als auch Spannungen invers im Vorzeichen voneinander.

Typ	n-Kanal	p-Kanal
Mosfet, selbstsperrend	$U_{th} > 0$ $U_{GS} > U_{th}$ $U_{DS} > 0$ $I_D > 0$	$U_{th} < 0$ $U_{GS} < U_{th}$ $U_{DS} < 0$ $I_D < 0$
Mosfet, selbstleitend	$U_{th} < 0$ $U_{GS} > U_{th}$ $U_{DS} > 0$ $I_D > 0$	$U_{th} > 0$ $U_{GS} < U_{th}$ $U_{DS} < 0$ $I_D < 0$

Abbildung 2.2: Polarität der Spannungen Ströme bei normalem Betrieb [18]

[1] Löcher sind wie die *Elektronen*, die dominierenden Ladungsträgerarten im Halbleitermaterial, und werden auch *Defektelektronen* genannt

Bereiche eines n-Kanal Mosfets

Das Ausgangskennlinienfeld eines Mosfets (Abb. 2.3) beschreibt sich durch zwei nah beieinander liegende Wirkbereiche: Den Widerstandsbereich (2.1), der sich dadurch kennzeichnet, dass der Strom I_D linear mit der Spannung U_{GS} ansteigt und den Sättigungsbereich (2.2), bei dem der Strom quadratisch mit U_{GS} steigt und von der Drain-Source-Spannung (U_{DS}) unberührt bleibt.

$$I_{DS} = \beta_n \left[(U_{GS} - U_{Th}) U_{DS} - \frac{{U_{DS}}^2}{2} \right] \tag{2.1}$$

$$I_{DS} = \frac{\beta_n}{2} ({U_{GS}} - {U_{Th}}^2) \tag{2.2}$$

β_n ist der Verstärkungsfaktor des Transistors, der über das Verhältnis der Kanallänge und Kanalbreite eingestellt werden kann.

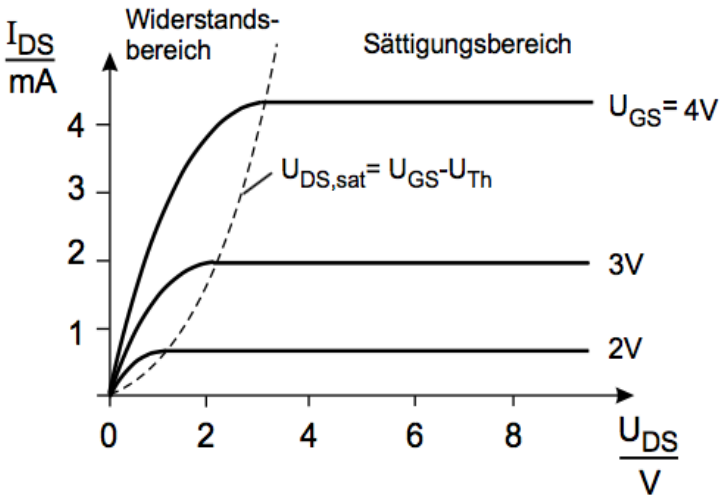


Abbildung 2.3: Ausgangskennlinienfeld des Mosfets

Die Sättigungsspannung ($U_{DS,sat}$), bei der eine sogenannte Kanalabschnürung auftritt, und das Temperaturverhalten eines Feldeffekttransistors werden aus praktischer Nähe unbeachtet gelassen. Weitere vertiefende Themen ebenso.

2.2.2 Bluetooth-Low-Energy (BLE)

BLE ist seit 2009 optionaler Bestandteil der Bluetooth-Spezifikation 4.0 und dafür designt worden, den geringsten Stromverbrauch aller Funkstandards bereit zu stellen, wodurch der Einsatz in sparsameren Architekturen im Sinne der Entwicklung liegt. Das klassische Bluetooth hingegen startete seine Karriere als zukünftige Verbindungsmöglichkeit, der zu dem Zeitpunkt bestehenden unterschiedlichen Welten von Smartphones und Computern bzw. Laptops. Im Laufe der Zeit entwickelten sich weitere nützliche Verwendungszwecke, wie das Herunterladen und Abspielen von Musik (Music-Streaming) oder dem kabellosen Drucken, die nachfolgend immer größere Bandbreiten benötigten. Bluetooth startete mit der Basic Rate (BR), die eine Brutto-Datenrate von 1Mb/s lieferte, und arbeitet gegenwärtig mit bis zu hunderten Mb/s in einer Spate wie die IEEE 802.11x[2] Standards. BLE nutzt die BR und soll ab der Standardisierung von Bluetooth 4.2 eine 2,5-mal so hohe Geschwindigkeit geboten bekommen als das derzeitig genutzte 4.0 bzw. 4.1. Darunter wird auch die Kommunikation selbst energiesparender, da die Datenpaketgrößen verkleinert werden. Ein weiterer ausschlaggebender Punkt, warum BLE entwickelt wurde, ist, dass es Geräte befürwortet, die keine Funk-Technologien integriert haben, und dazu sehr wenig Platz für zusätzliche implementierbare Upgrades bieten. Ein Gegenbeispiel für diesen Fall sind RFID[3]-Marken, die platzsparend, billig und ebenso einen sehr geringen Stromverbrauch bieten. Diese Marken können ohne einen zugehörigen RFID-Scanner aber nicht erkannt werden, wodurch eine Anschaffung eines solchen benötigt wird und gleichauf negativ in das Budget eines Herstellers fällt.

Gerätetypen

Es bestehen zwei mögliche BLE-Varianten, der *single*- und *dual-mode*. Ersterer, und auch älterer, ist darauf ausgelegt, dass das Gerät mit sehr geringem Stromverbrauch, durch u. a. möglichst kleine und preisliche *button-cell* Batterien versorgt, arbeitet und ausschließlich im Stande ist, andere BLE-Geräte zu finden, die mindestens einen der beiden Modi unterstützen. Der *dual-mode* ist die heutig genutzte generelle Bluetoothcontroller-Implementierung, da der Modus nicht nur BLE-Geräte finden, sondern auch die in millionenfacher Ausfertigung existierenden (Klassik) Bluetooth-Geräte entdecken kann [6].

[2] IEEE steht für *Institute of Electrical and Electronics Engineers* und ist der Norm-Standard vom Wireless LAN, welcher den Mediumszugriff und die physische Schicht in verschiedenen Versionen (z. B. 802.11-b, -g und -n) beschreibt, damit u. a. Übertragungsgeschwindigkeiten gewährleistet werden und die lizenzfreie Nutzung bestimmter Frequenzen erlaubt wird [13].

[3] RFID steht für *Radio-Frequency-Identification* und ist eine Funk-Technologie, die über sehr kurze Distanzen durch ein elektromagnetisches Feld angrenzende Marken lesen kann, die ihrerseits elektronische Daten besitzen können und durch den erfassenden RFID-Scanner, per Funkwellen, mit dem zur Übertragung benötigten Energie versorgt werden.

ISM Band

Ein sparsames Design hat zwangsläufig auch Nachteile: Die Nutzung der lizenzfreien Frequenzbereiche, welche weltweit genutzt werden, und bei zu großen Überlappungen Datenverlust hervorrufen können. Trotz großer verfügbarer Spanne an Frequenzbereichen, wurde das 2,4GHz ISM Band gewählt, da Bluetooth selbst davon Gebrauch macht und in Verbindung damit der Vorteil geschaffen wird, den *dual-mode* vom BLE nutzen zu können. Die in den klassischen Bluetooth-Geräten eingebaute Antenne kann somit zur Nutzung von Bluetooth-Low-Energy verwendet werden [6].

Generic Access Profile (GAP) und Generic Attribute Profile (GATT)

Für die fundamentale Funktionsweise von Bluetooth-Low-Energy (BLE) muss das Generic Access Profile (GAP) und Generic Attribute Profile (GATT) erwähnt werden, welche beide Ebenen in der BLE-Architektur (siehe Abb. 2.4, Seite 16) sind. Alle Beziehungen zum BLE-Bindeglied, z. B. zu einem Smartphone, sind im GAP des BLE-Moduls und deklarieren unter anderem wie ein solches Bindeglied eine Verbindung aufbauen darf oder welche Informationen, nach abgeschlossener Verbindung, zur Verfügung stehen werden. Weiterhin sind dort Eigenschaften der Privatsphäre definiert und lassen die Moduladresse privat stellen, sodass dem Modul eine kontinuierlich ändernde Adresse inne steht, wodurch scannende Bindeglieder kein konkret erreichbares Signal erlangen können und darauf gewisse Unsichtbarkeit erlangt wird. Parameter zum Erreichen eines privat-gestellten Moduls sind dort ebenfalls definiert.

Das GATT stellt sogenannte Attribute für das Bindeglied bereit und bestimmt wie bzw. ob diese sichtbar sein sollen. Attribute strecken sich über *characteristics*, *services* und *desciptors*, die wegen der fehlenden Applikationsseite dieser Arbeit nicht tiefgründig erläutert werden. Theoretisch sind es Ansprechpartner mit verschiedenen Eigenschaften, wie Daten lesen und/oder Daten schreiben und müssen explizit auf der Bindeglied-Seite mit auf dem Modul definierten UUID's[4] angesprochen werden.

[4] Ein Universally Unique Identifier (UUID) besitzt die übliche Struktur eines 128-Bit langen Wertes wie z. B. 49535343-FE7D-4AE5-8FA9-9FAFD205E455 und steht für eine *characteristic*, einen *service* oder einen *desciptor*

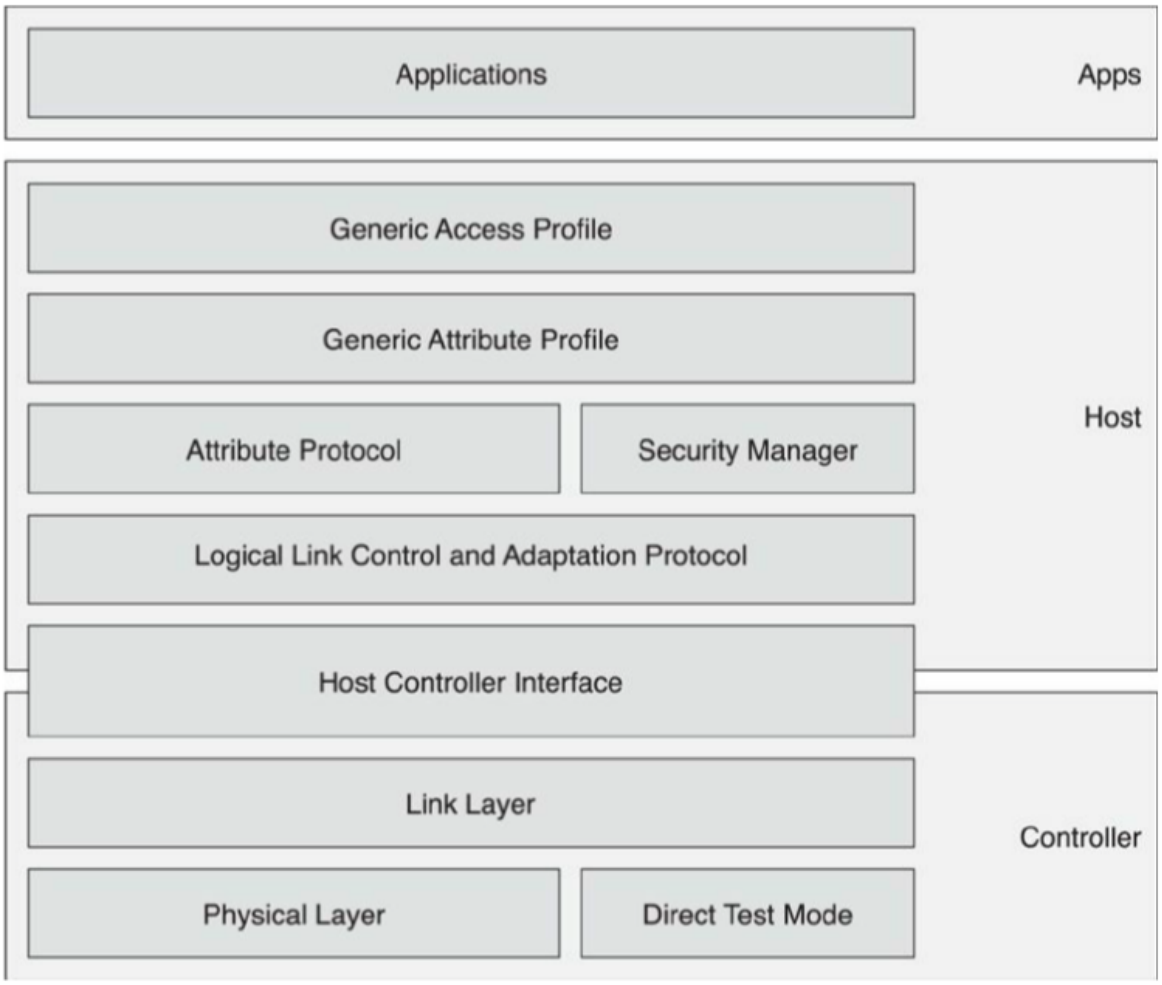


Abbildung 2.4: Komplette Bluetooth-Low-Energy Architektur

2.2.3 Speichercharakteristiken

Sowohl der Mikrocontroller, als auch der eingebaute Speicher selbst, werden vom Verhalten her durch bestimmte Speichercharakteristiken bestimmt. Das für den Programmierer bedeutendere Charakteristik ist das *non-volatile*, ein Speichermedium, dessen Daten auch im spannungslosen Zustand erhalten bleiben. Speziell der *flash memory* speichert die Informationen in Form von elektrischen Ladungen dauerhaft in den Floating-Gates[5] von MISFET's ab, wobei letzterer als auch der physikalisch funktionierende Aspekt dahinter nicht weiter untersucht wird, da dies für die Reichweite dieser Arbeit nicht benötigt wird. Der SRAM ist ein statisch arbeitender Speichertyp, der Informationen nur flüchtig abspeichert und sie somit nach Abschaltung der Betriebsspannung wieder verliert. Gegensätzlich dazu der DRAM, der periodische Spannungsauffrischungen braucht, um Datenverluste richtig vorzubeugen. Beide sind, im Unterschied zu dem Flash, *volatile memory* und bezeichnen den Typ von Speicher, welcher bei Spannungsverlust alle Daten verliert. Jeder einzelne Bit-Zustand wird in einem Integrated circuit (IC), bestehend aus mindestens zwei Komponenten, verwertet und je nach Typ unterschiedlich abgespeichert. Beim DRAM ist dies genau ein Transistor und ein Kondensator, was verhältnismäßig zum SRAM, welcher einen aus (vier) sechs Transistoren zusammengeschalteten IC verwendet, wenig Aufwand sowie Kosten fordert. Dafür schlägt der Komplexere den Simpleren in Schnelligkeit. Resultierend daraus bedienen beide jeweils andere Anwendungsgebiete: SRAM vorzugsweise den Prozessor-Cache[6] und der DRAM den Hauptspeicher eines Computers [3].

2.3 Erläuterung der Hauptkomponenten-Auswahl

Das folgende Thema beschreibt nur Hauptkomponenten, die sich auf ansteuerbare separate Platinen oder IC's beschränken und beschreibt diese nur in ihren generellen funktionellen Eigenschaften sowie den Verwendungszwecken, jedoch noch nicht konkret, welche letztlich eingebaut werden.

[5] Floating Gate: Unterscheidet sich von dem normalen Gate darin, dass es komplett von allen anderen Teilen durch ein Dielektrikum isoliert ist [3].

[6] Cache: Der Cache ist ein definierter Speicherbereich, welcher auf konkrete Prozessorbereiche zeigt bzw. diese dort dupliziert, um so so schnellst wie möglich darauf zugreifen zu können.

2.3.1 Bluetooth-Low-Energy-Modul

Die Entscheidung, warum Bluetooth-Low-Energy und nicht eine verbreitetere Funktechnologie wie das WLAN gewählt wurde, fiel hauptsächlich wegen des immensen Unterschiedes im Stromverbrauch ($\sim$200mA im arbeitenden Betrieb im Gegensatz zu 13mA *worst-case* Strom des BLE-Moduls). Da die Firma Scholz System in enger Kooperation mit ATMEL arbeitet und diese die Firma Microchip gekauft hat, bot sich so ein möglicher weltbekannter Distributor zum Einkauf eines BLE Moduls an. Das BLE Modul-Portfolio des Verkäufers Microchip bietet bereits Platinen an, die Bluetooth 4.2 unterstützen. Ebenfalls haben sie eine integrierte Chip-Antenne, ein simples API zur Einstellung der Kommandos per ASCII-Zeichenketten, UART und die typische Betriebsspannung von 3,3V unterstützt.

2.3.2 Real-Time-Clock (RTC)

Im Kapitel „Theoretische Überlegungen der Kernaufgabe” auf Seite 19 wird ein zukünftig zu nutzendes, realitätsnahes Beispiel erläutert, das einen Einbau einer RTC in eine Kaffeemühle begründet. Speziell für das verwendete Gerät entwickelte Smartphone-Applikationen könnten damit *realtime* Zeitangaben für Referenzierungen benutzter Geräteinformer-Daten auf z. B. einer Graphical User Interface (GUI) liefern. Ein weiterer, optionaler, Verwendungszweck wäre die Anzeige der aktuellen Uhrzeit auf der Mühle, da diese keine eigene Zeiteinheit in der Verwendung hat.

2.3.3 Speichereinheit

Um aus der einzubauenden RTC vollen Nutzen ziehen zu können, und dazu wichtige Aktionen zwischen Smartphone und Low-Energy-Platine zu dokumentieren, ist die Entscheidung gefallen, einen kleinen 1MB Flash-Speicher mit einzubauen.

2.3.4 Batterie (button cell)

Das Ziel der zu entwickelnden Low-Energy-Platine ist eine so lang wie möglich anhaltende, Standby-Werkdauer der RTC, mit dem Vorhaben, eine über mehrere Jahre andauernde Werkdauer, bei nicht anliegender Spannung, zu ermöglichen. Um Geräten, die zukünftig ein solches Upgrade in Anspruch nehmen würden, damit die Möglichkeit zu bieten, in Verbindung eines gegebenen Speichervolumens, Daten auch mit *realtime* Zeitangaben langfristig und kontinuierlich abspeichern zu können. (Bei Kaffeemühlen bereits sicher, Zigarettenautomaten noch unklar)

2.4 Anforderungen des Mikrocontrollers

Um einen geeigneten, kostengünstigen Mikrocontroller wählen zu können, muss einem Hardwareentwickler vorher klar sein, welche und wie viele Schnittstellen mindestens benötigt werden. Die ersten Ideen dazu würden wie auf Abb. 2.5 aussehen und bereits einen Mikrocontroller mit 8Bit qualifizieren, da die Anzahl der beiliegenden Platinen/IC's sehr gering gehalten wird und somit wenige Kommunikationsschnittstellen benötigt werden.

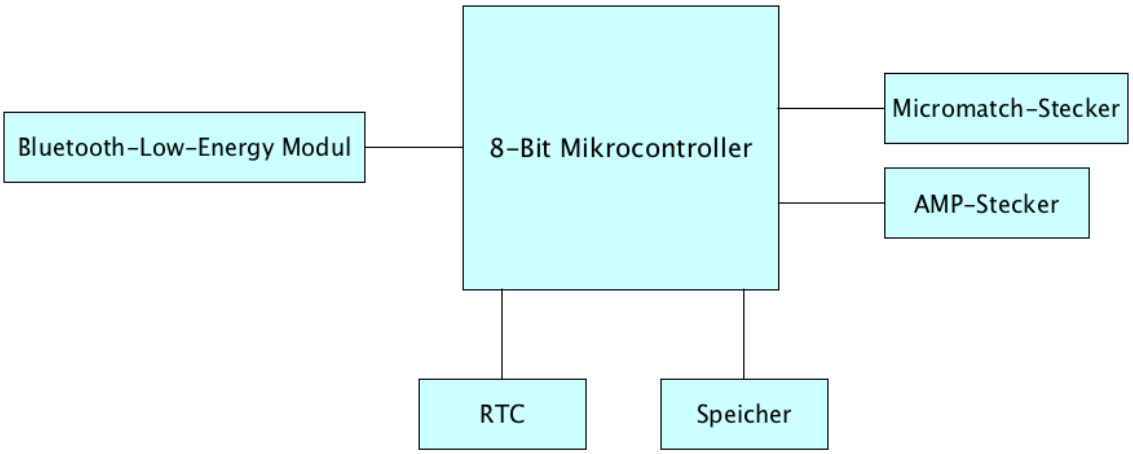


Abbildung 2.5: Unbestimmter Verbindungsaufbau der Hauptkomponenten zum Mikrocontroller

2.4.1 In-System-Programmierung (ISP)

Die Programmierschnittstelle, auch ISP genannt, wird separat zum Mikrocontroller, hier als Sechsfach-Pinarray, auf der Platine platziert. Diese dient der Beschreibung des Mikrocontroller-Flashs (32KByte), um die selbstgeschriebene Software, welche das Miteinander aller Hauptkomponenten regelt, anwenden zu können.

2.4.2 Externer Quarz

Da der interne RC-Oszillator eines Mikrocontrollers auf verschiedene konstante ganzzahlige Werte (1MHz, 2MHz, 4MHz und 8MHz) einstellbar ist und sich nicht unbedingt für die vorhandenen asynchronen UART Schnittstellen verwenden lässt (ungenau), wird ein externer Quarz mit der Frequenz von 7,3728 MHz eingebaut. Diese Frequenz erschließt sich aus dem in Kapitel 3.1.5, auf Seite 30, beschriebenen Optimum. Um einen stabilen Schwingkreis garantieren zu können, werden zwei parallel zueinander stehende Trimmkondensatoren, mit explizit aus dem dazugehörigen Datenblatt entnommenen Werten, an den Quarz angeschlossen. Zusätzlich werden Streueinwirkungen eingerechnet.

Einen Nutzen aus dem ungenauen internen Oszillator könnte man ansonsten, durch ein manuelles Kalibrieren einer jeden bestückten einzelnen Platine ziehen. Dies wäre aber wegen des geringen Preises von Quarzkristallen zu umständlich sowie zeitaufwendig.

Der Unterschied eines Quarzes zu einem Quarzoszillator ist, dass letzterer ein eigener IC ist und nur einen Mikrocontroller-Pin benötigt. Dies ist bedingt dadurch, dass der Oszillator nicht durch einen weiteren Pin zum Schwingen angekurbelt werden muss, sondern schon ein fertiges Rechteck-Signal liefert. Üblich besteht ein Quarzoszillator aus zwei Trimmkondensatoren, mehreren Widerständen, Bipolartransistoren, einer benötigten Spannungsansteuerung und einem Schwing-Quarz.

3 Hardware-Design

Das Hardware-Design, bestehend aus Schaltplan- und Platinen-Design, wird mit dem vom Hause Cadsoft entwickelten Programm, dortig *Schematic*- und *Layout-Editor* genannt, durchgeführt und Computersimulationstests a la PSpice, wegen zu geringer Hardwarekomplexität, nicht.

3.1 Auswahl der Komponenten

Die folgenden Unterkapitel bereiten alle nötigen technischen als auch größenspezifischen[1] Informationen über die zu verbauenden Bauteile vor, um die Platine im Kapitel 3.4 auf Seite 54 designen zu können. Für den Schaltplan im Kapitel 3.3 auf Seite 38, sind die Größenspezifikationen irrelevant, da dieser für die innere Funktionsweise und spätere Beschaltung verwendet wird.

Die Entscheidung über die Komponentenwahl wurde nicht anhand jeweils untereinander herrschenden geringen preislichen Unterschieden, sondern ob bekannte Distributoren wie Digi-Key- oder Mouser-Electronics, diese in Massen vorrätig haben, gemacht.

Eine Zusammenfassung aller Komponenten samt Informationen kann in Kapitel 3.2, auf Seite 36, eingesehen werden.

[1] Eine detaillierte Beschreibung über die Erstellung jedes Bauteils in EAGLE (*Schematic-/Layout-Editor*) wird nicht gemacht, wobei aber als Vorführbeispiel das BLE-Modul („Vorgehensweise zur Verwendung des BLE-Moduls in EAGLE") für die zu erstellende Platine genauer beleuchtet wird

3.1.1 Bluetooth-Low-Energy-Modul

Das RN4870 ist die neueste Entwicklung von den bei Microchip produzierten BLE-Modulen und wird dabei in vier Formen, je zwei Größen, mit oder ohne eingebaute Antenne und Abschirmung, angeboten. Die kleinste, mit letzteren *Features* verfügbare Größe von 9mm/11,5mm, mit dem Namen RN4871 (Siehe Abb. 3.1), ist 36,5% kleiner als das RN4870 Model (12mm/22mm) und besitzt trotz ~50% weniger Pins und somit geringerer Spanne an Möglichkeiten, einen ausreichenden Nutzen für die Zwecke dieser Low-Energy-Platine. Die wichtigsten funktionell gelieferten Eigenschaften sind u. a. eine serielle Anbindung, die durch UART geschaffen wird, eine Betriebsspannung zwischen 1,9V und 3,6V, eine Low-Energy-Performance im frei nutzbaren frequenten Bereich (2,402GHz bis 2,480GHz) beim Senden (0dbm) und Empfangen (-90dbm) und für die Anbindung bzw. Kommunikation mit einer Smartphone-Applikation das GAP[2] und GATT[2], mit vorhandenen Attributen [9].

Abbildung 3.1: Ein RN4871 BLE-Modul von Microchip [9]

Vorgehensweise zur Verwendung des BLE-Moduls in EAGLE

Ein Vorführbeispiel, um in EAGLE ein Bauteil für den Schaltplan (*Schematic*) und das Platinendesign (*Layout*) zu erstellen, und daraufhin nutzen zu können, zeigt das abgewandelte Flussdiagramm der Abb. 3.2, auf Seite 23.

Hinweis:
Dabei wird nicht jeder in EAGLE zu bearbeitende Schritt konkret erklärt und die finale Zusammenführung von Schaltplan und Platine ebenfalls unberücksichtigt gelassen.

[2]GAP und GATT sind im Kapitel 2.2.2, auf Seite 15 erläutert

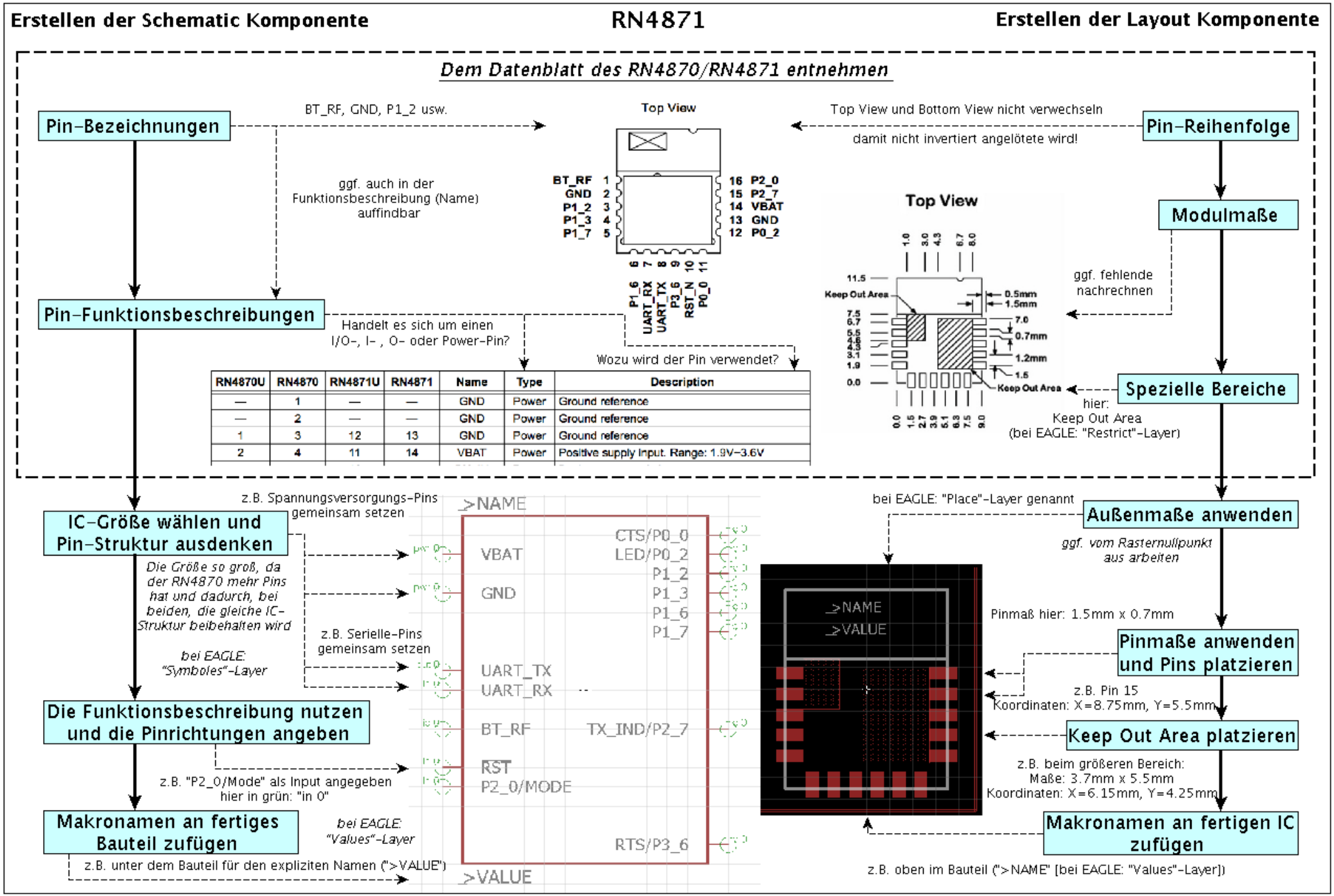
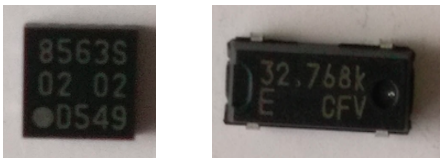


Abbildung 3.2: EAGLE-Bauteilerstellung vom BLE-Modul: RN4871[9]

3.1.2 Real-Time-Clock (RTC)

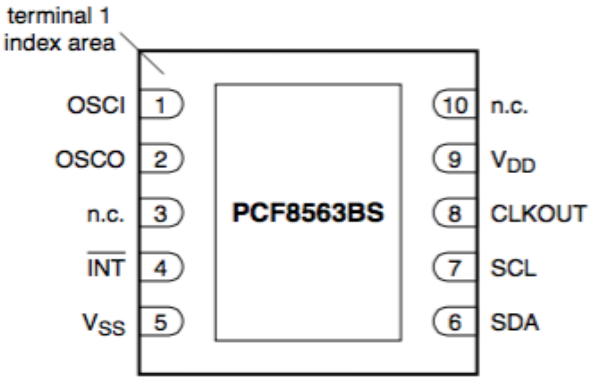
PCF8563BS ist die eingebaute, mit *low power* arbeitende RTC von der Firma NXP, welche ihre zeitlichen Daten per I^2C übertragen lässt und auf einen 32,768kHz Quarz eingerichtet ist. Die Wahl dieser RTC entschied sich dadurch, dass sie bei der Firma Scholz System gängig angewandt wird und sich über einen längeren Zeitraum, auf eingebauten Platinen, im Zeitverhalten als zuverlässig erwies. Ihr Betriebsspannungsbereich liegt zwischen 1,8V – 5,5V und der Stromverbrauch im *worst-case*-Fall bei bis zu 50mA, was übermäßig viel wäre. Im arbeitenden Betrieb sind es hingegen nur zwischen 250μA – 800μA (im Standbybetrieb, bei 3V Betriebsspannung, typischerweise 250nA – 1000nA).

Das Gehäuse ist ein HVSON10[3] mit den Maßen 3mm x 3mm x 0,85mm und wird auf der Abb. 3.3 (a) als Bauteil gezeigt. Für den Quarz wurde der CRYSTALMM20SS gewählt (b). Die SCL- und SDA-Pins dienen der Datenkommunikation des I^2C (siehe untere Abb. [c] Pinbelegung[4]). $\overline{INT}$ und CLKOUT werden nicht benutzt, da Timer- oder Alarmfunktionen der RTC nicht verwendet werden und das hier benutzte Clocksignal keiner Führung nach außen bedarf, wobei standardmäßig CLKOUT, trotz Nicht-Beschaltung, aktiv ist. OSCI und OSCO sind die Ein- und Ausgangspins für die geforderte Frequenz.



(a) RTC

(b) Quarz



(c) Pinbelegung

Abbildung 3.3: (a) Eine Real-Time-Clock (PCF8563BS) (b) Ein zur RTC verwendeter Quarz mit 32,768kHz (CRYSTALMM20SS) (c) Pinbelegung der RTC

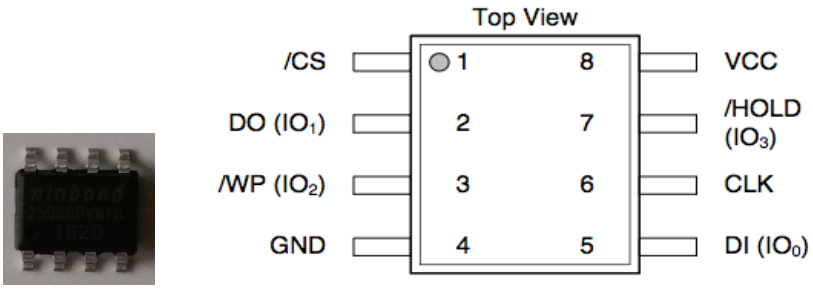
[3] HVSON10 steht für *plastic thermal enchanced very thin small outline package* mit Zehn-Pins [11]

[4] n.c steht für *not connected* und darf nicht beschaltet werden. *terminal 1 index area* steht für die Eckkante des Bauteils, die den Pin 1 markiert

3.1.3 Speichereinheit

Für die Speichereinheit wurde ein 1MByte serieller Flash (W25Q80DVSNIG) genommen, der zwischen 2,7V und 3,6V Spannung benötigt und 1μA im *Power-down*-Betrieb verbraucht. Der Indize „SNIG" steht für ein 150-mil SOIC Gehäuse. Im Standby-Betrieb sind dies lediglich 10μA und im *worstcase* Fall 50μA (25mA im arbeitenden Normalbetrieb). Die über SPI übertragbaren Daten werden im Inneren in einem Array aus 4096 Seiten gespeichert, wobei eine Seite 256Byte groß ist. Beschriebene Daten können in Blöcken von 16 (4KB), 128 (32KB) oder 256 (64KB) gleichzeitig gelöscht werden. Gerade die kleinste Blockeinheit war ein wichtiges Kriterium bei der Auswahl des Speichers, da kurze ASCI-Zeichenketten mit typischen, sehr geringen Byte-Größen zukünftig gespeichert und auch wieder gelöscht werden sollen.

SPI wird in drei Modi unterstützt: Standard SPI für bis zu 104MHz, Dual SPI für 208MHz und Quad SPI für 416MHz, wobei aus nicht benötigter Hochgeschwindigkeitsübertragung nur ersteres benutzt wird. Der Chip-Select Pin ($\overline{CS}$ bzw. auf Abb. 3.4 (b) /CS), Serial Clock Pin (CLK), Serial Data Input (DI (IO_0)) und Serial Data Output (DO (IO_1)) werden für den Standard SPI-Betrieb benutzt. Das *Hold* Signal ($\overline{HOLD}$ bzw. /HOLD (IO_3)) könnte, bei geteilt genutzter CLK- und Daten-Leitung, einem anderen anhängenden Glied diese überlassen und dabei den bestehenden Daten- sowie Kommando-Zustand festhalten bzw. bei widriger Busübergabe fortführen. Beim *Write Protect* Signal ($\overline{WP}$ bzw. /WP (IO_2)) handelt es sich um eine Sicherheit zur Erhaltung der Datenintegrität bzw. zur Verhinderung von Datenverlust beim Ein- und Ausschalten des gesamten Gerätes. Auf der Abb. 3.5, auf Seite 26, wird u. a. veranschaulicht, dass beim Einschaltvorgang die Betriebsspannung VCC unter dem Wert der Trashhold-Spannung V_{WI} das Gerät resetet (*Reset State*) und erst nach der abgelaufenen Zeitkonstante t_{PUW} ein Einpendeln von VCC zu sehen ist. Auch wird der generelle Funktionsmechanismus des Flashs erst ab VCC_{min} allmählich verfügbar. Das Gehäuse ist ein 150-mil SOIC, mit einer Breite von 3,9mm und einer Länge von 4,9mm, und im Vergleich zum VSOP nur im Raster unterschiedlich, wie auf der Abb. 3.4 (a) zu sehen ist [19].



(a) Gehäuse (b) Pinbelegung

Abbildung 3.4: (a) Ein 1MByte Flash im 150-mil SOIC-Gehäuse (W25Q80DVSNIG) (b) Top-View-Pinbelegung des Flash-Speichers [19]

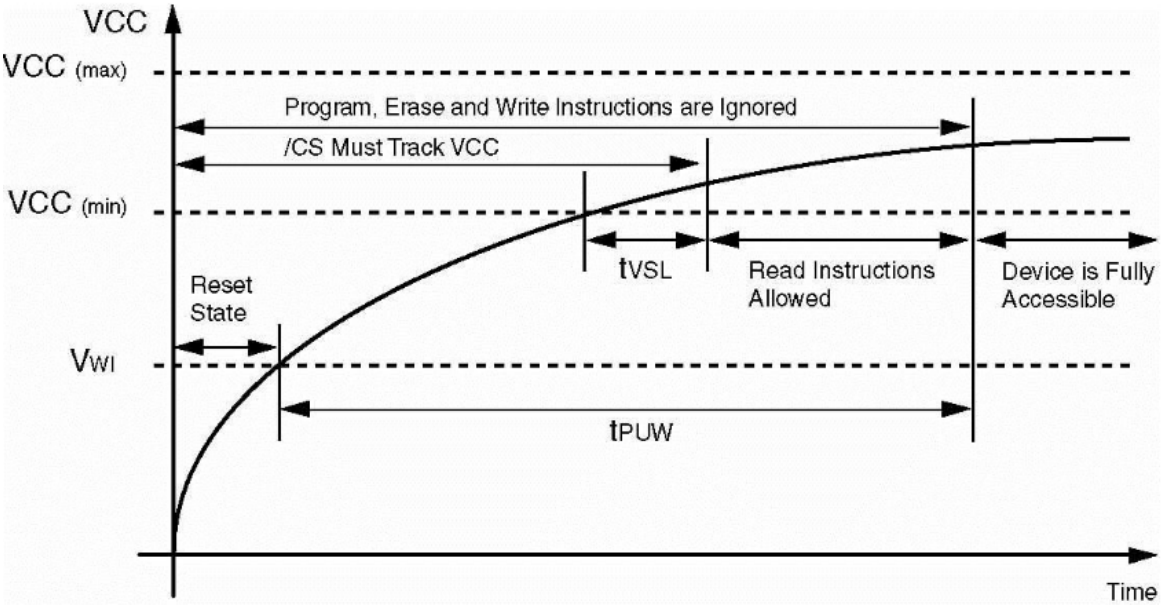
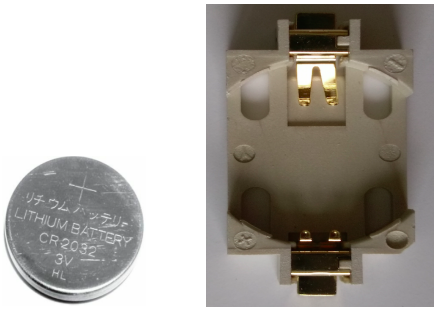


Abbildung 3.5: Power-up Timing und Spannungslevel [19]

3.1.4 Batterie (button cell)

Die gewählte *button cell*-Batterie ist ein üblich benutzter Typ mit dem Namen CR2032, bei der die Ziffernfolge "20" für einen 20mm Durchmesser und "32" für eine Höhe von 3,2mm (Abb. 3.6) steht. Die Kapazität beträgt 230mA/h bei 3V, wobei nur auf den *worstcase* Stromverbrauch der designten Platine im Kapitel „Betriebsspannungs- und Stromanalyse” auf Seite 31 eingegangen wird, da vorerst ein Prototyp ohne Feinschliff entwickelt wird.



(a) Batterie (b) Halterung

Abbildung 3.6: (a) Eine CR2032 (*button cell*) Batterie (b) Eine Halterung der CR2032

3.1.5 8-Bit Mikrocontroller

Aus den Unterkapiteln des Kapitels „Auswahl der Komponenten” und „Spannungsversorgung und nach außen führende Datenverbindung” geht hervor, dass der zu nutzende Mikrocontroller mindestens zwei SPI-, zwei UART- und eine I^2C-Schnittstelle bereitstellen muss, um mit dem unterschiedlichen Datenkommunikationsverhalten jeder Komponente umgehen zu können. Nach umfangreicher Recherche fand sich ein 8-Bit Mikrocontroller mit ausreichend benötigten Kommunikationsschnittstellen, einem Betriebsspannungsbereich von 2,7V – 5,5V und dazu geringem Stromverbrauch von 5,2mA im aktiven Betriebsfall des aus dem Datenblatt entnommenen Referenzbeispiels über 8Mhz, bei 5V Betriebsspannung. Der *worstcase* im Stromverbrauch wird mit 10mA angegeben. Dabei handelt es sich um den von Atmel angebotenen ATmega328PB-MU, wobei der Indize „-MU” für die SMD-Variante (Paketidentifizierung: 32MS1 als VQFN[5]) und für Temperaturen zwischen -40°C – 105°C steht (siehe Abb. 3.8 (a) auf Seite 29) [2].

Aus den allesamt definierten Schnittstellenbestimmungen kann ein „Bestimmter Verbindungsaufbau der Hauptkomponenten zum Mikrocontroller” geschaffen werden, um die endgültig zu entwickelnde Low-Energy-Platine vollends für den vorerst zu entwickelnden Schaltplan vorbereitet zu haben. Die Abbildung zeigt dazu die genauen, auf dem Mikrocontroller befindlichen, Schnittstellen USART, SPI und TWI (I^2C) ‚0’ oder ‚1’, entnommen aus der Abb. 3.15 auf Seite 39.

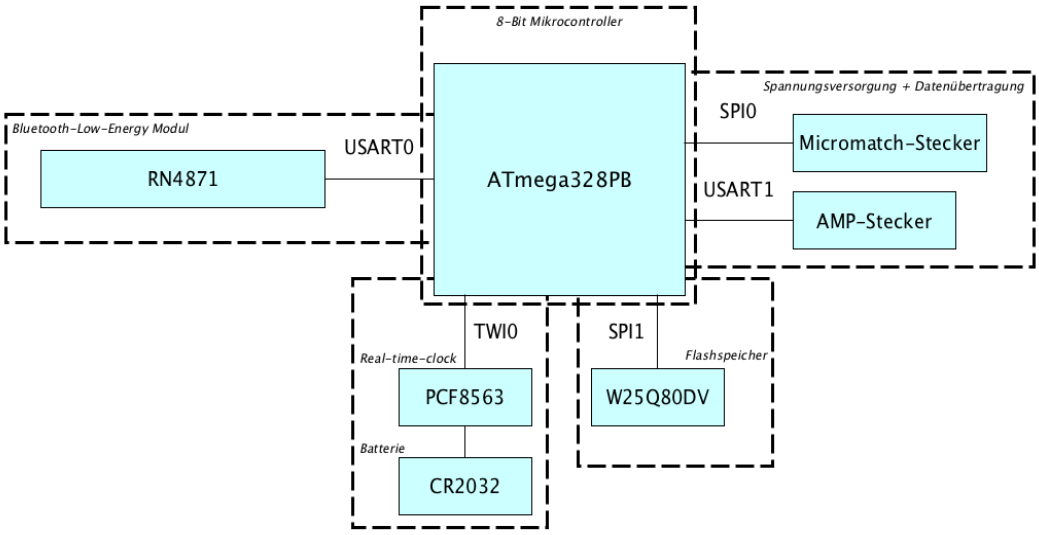
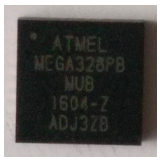


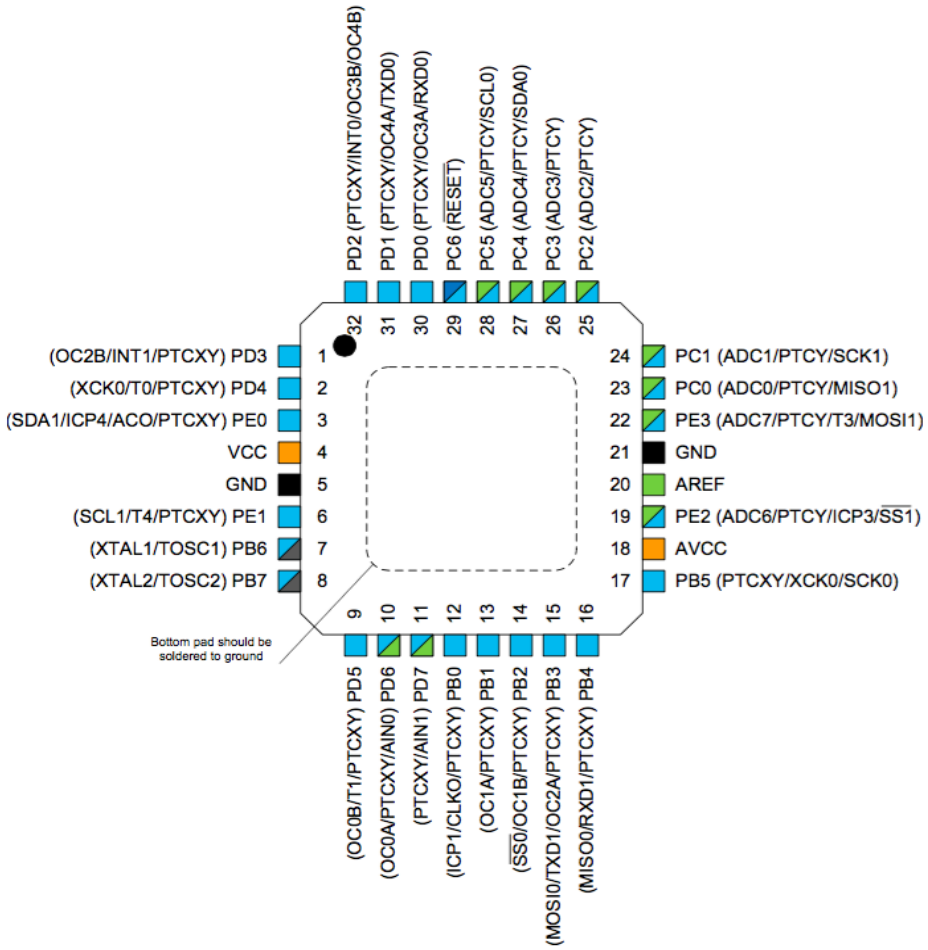
Abbildung 3.7: Bestimmter Verbindungsaufbau der Hauptkomponenten zum Mikrocontroller

[5] Beim *VQFN* steht das ‚V’ für *Very-thin Fine pitch* und „QFN” für *Quad Flat No Lead Package, was bedeutet, dass die Pins unter dem Gehäuse liegen und nicht nach außen ragen sowie eine enorm geringe Gehäusehöhe haben*

Jeder auf Abb. 3.8 (b) zu sehende Pin besitzt eine Vielzahl von Anwendungsmöglichkeiten, wie die für uns wichtige Anbindung eines externen Quarzes via PB6(XTAL1) und PB7(XTAL2), als auch einer Programmierschnittstelle via SPI0 mit PB3(MOSI0), PB4(MISO0) und PB5(SCK0) für den Mikrocontroller, ohne den die später beschriebene Software nicht in den internen Flash gelangen kann. Diese serielle Schnittstelle ist vom Entwickler definiert und muss verwendet werden, wodurch praktisch eine SPI-Schnittstelle weniger zur generellen Verwendung steht, was für unseren Zweck suboptimal ist und in Kapitel Prototyp (Fehlerdiagnose), auf Seite 70, genauer erläutert wird.



(a) Gehäuse



(b) Pinbelegung

Abbildung 3.8: (a) Ein ATmega328PB-MU 8-Bit Mikrocontroller von Atmel (b) Top-View-Pinbelegung des Mikrocontrollers [2]

Externer Quarz

Der zusätzliche extern genutzte Quarz heißt ABMM2-7.3728MHz-E2-T (Abb. 3.9) und wird bei der Firma Scholz System mit einer Fertigungstoleranz[6] von 20 ppm gängig genutzt.

Abbildung 3.9: Ein ABMM2-7.3728MHz Quarz

Ein linear ansteigendes Verhältnis zwischen den Frequenzen 4MHz – 20MHz und Betriebsspannungen 1,8V < V_{CC} < 2,7V sowie 2,7V < V_{CC} < 4,5V kann in der Abb. 3.10 erkannt werden. Warum gerade die Resonanzfrequenz 7,3728MHz das Optimum unserer Zwecke darstellt, entschied unter anderem die innere Betriebsspannungsanalyse auf Seite 31, welche 2,7V als optimal qualifiziert. Weiterhin wird die Wahl durch die untere Abb. begründet, welche 10MHz als Maximum bei 2,7V empfiehlt Der zweite entscheidende Punkt ist das Verhältnis von den Baudraten der UART-Schnittstellen zu den Fehlertoleranzen im Werksbetrieb. Die 7,3728MHz erbringen die bestmöglichen 0,0% bei 2400bps bis 230,4kbps.

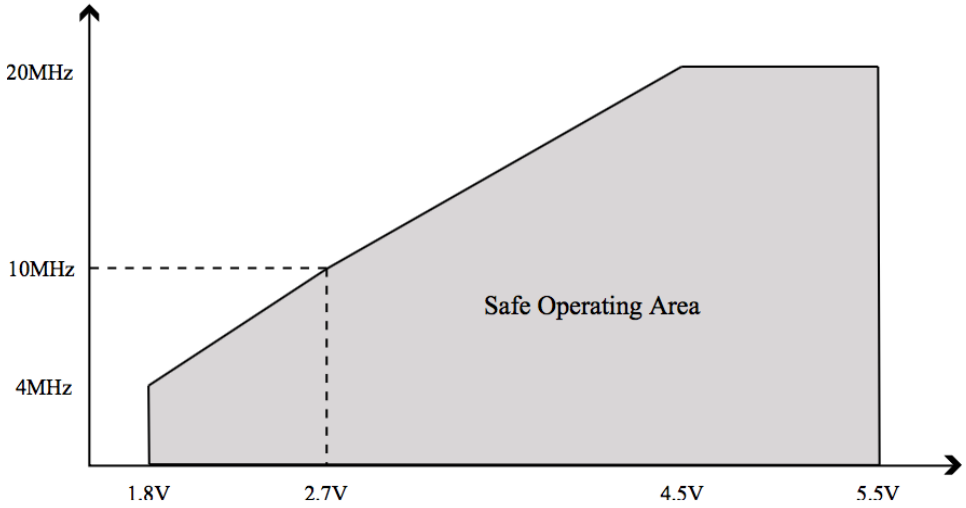


Abbildung 3.10: Verhältnis von V_{CC} zur maximal nutzbaren Frequenz [2]

[6] Die Fertigungstoleranz gibt die vermeintlichen Abweichungen des Quarzes zur Nennfrequenz an, also z. B. bei einem 10MHz Quarz mit ±10 ppm eine beidseitige Abweichung von 100Hz

3.1.6 Spannungsversorgung und nach außen führende Datenverbindung

Ein noch unbehandelter Punkt ist die Spannungsversorgung der Low-Energy-Platine. Diese soll von den bisher in Zigarettenautomaten 12V (oder 40V) und Kaffeemühlen verwendeten 5V geliefert werden, wobei eine separate Platine ggf. als Spannungsumwandler genutzt wird (40V, 12V –> 5V). Um dazu die zwischenkommenden Datenmengen verwenden zu können, muss für beide Fälle ein gemeinsamer verfügbarer Anschluss eingebaut werden. Die Firma Scholz System benutzt in den beiden o. g. Produkten AMP- und Micromatch-Stecker, weswegen in der zu entwerfenden Low-Energy-Platine diese beiden Typen eingebaut werden sollen. Der AMP-Stecker wird dabei eine UART-Schnittstelle und der Micromatch-Stecker, trotz vieler potenziell nutzbarer Pins, eine SPI-Schnittstelle nach außen bieten.

3.1.7 Betriebsspannungs- und Stromanalyse

Jede einzubauende Komponente besitzt einen Betriebsspannungsbereich und im arbeitenden Betrieb einen *worstcase*-Strom. Die Betriebsspannung, welche für die zu entwickelnde Low-Energy-Platine letztlich benutzt wird, ergibt sich aus den Minima der betrachteten Spannungsbereichen jeweiliger Komponenten. Die *worstcase*-Ströme wurden aus den verfügbaren *Datasheets* entnommen und summiert, um einen geeigneten Low-Dropout Regulator finden zu können.

Für die Betriebsspannung ergibt dies 2,7V und darf höchstens 3,6V betragen und diese, um Bauteilschäden zu vermeiden, nicht übermäßig über- bzw. unterschreiten. Der *worstcase*-Strom beträgt 98mA und wird im nächsten Kapitel auf Seite 32 genauer untersucht.

In der Tabelle A.1 wurden diese entnommen und aus den Unterkapiteln des Kapitels 3.1, ab Seite 21, zusammengefasst.

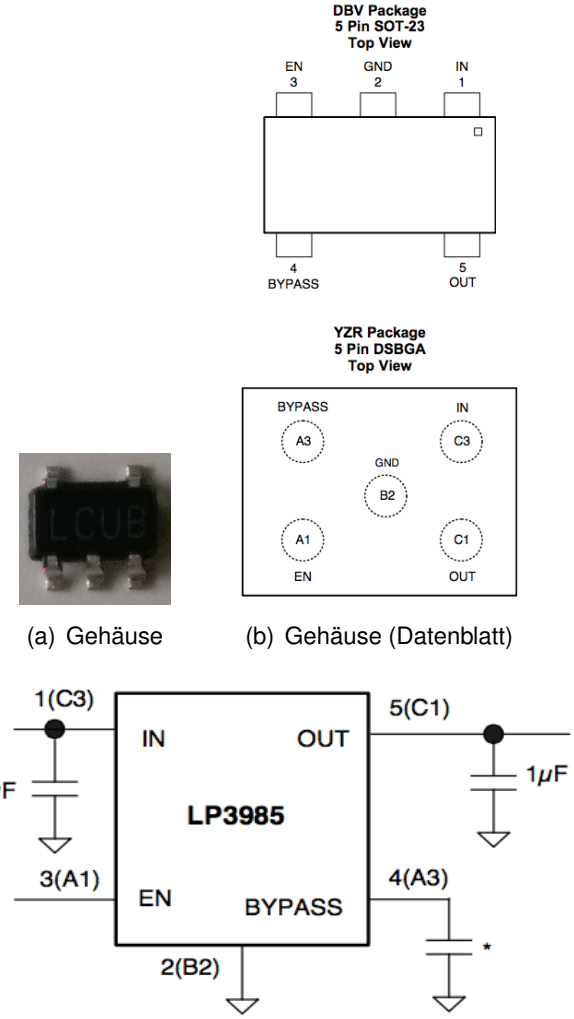
Komponente	**Spannungsbereich[V]**	***worstcase*-Strom[mA]**
Speicher	2,7 – 3,6	25
Real-Time-Clock	1,8 – 5,5	50
BLE-Modul	1,9 – 3,6	13
Mikrocontroller	2,7 – 5,5	10
	2,7 – 3,6	**98**

Tabelle A.1: Betriebsspannungs- und *Worstcase*stromanalyse der einzubauenden Hauptkomponenten der Low-Energy-Platine

3.1.8 Low-Dropout Regulator (LDO)

Der Low-Dropout Regulator (LDO), auch Spannungswandler genannt, wird teils basierend auf den in Tabelle A.1 (Seite 31) beschriebenen Endwerten für die Betriebsspannung und den *worstcase*-Strom gewählt, und dient der Dekrementierung bzw. Stabilisierung extern ankommender Eingangsspannungen auf den gewünschten Wert von 2,7V. Stromschwankungen am Ausgang werden abgefangen. Der errechnete 98mA *worstcase*-Strom wird noch mehrere Milliampere zur Endsumme dazu bekommen, da Pull-up-Widerstände in die Schaltung eingebaut werden. Für die Endsumme ergibt sich ~100mA, woraufhin ein LDO namens LP3985IM5-2.7 von Texas Instruments gewählt wird, der ausreichend bis zu 150mA vom Ausgang stammenden Strom liefern kann, als auch bei Fehlern eine Überschreitung dieser unterbindet. Da dies ein sogenannter *Low-Dropout* Regulator ist, liegt der (mindeste) Unterschied zwischen Ein- und Ausgangsspannung bei maximal 100mV und nicht bei mehreren Volt normaler Regulatoren. Die auf Abb. 3.11 (c), auf Seite 33, zu sehenden Kondensatoren dienen der Stabilisierung des Reglers beim Ein- und Ausgang, um hochfrequente Schwingungen zu kompensieren. Das Gehäuse ist ein SOT-23 (Abbildungsteil [a]), mit einer Breite von 2,9mm und einer Länge von 1,6mm, und im Vergleich zur SMD-Variante DSBGA in Größe (1,502mm x 1,045mm) als auch Pinauslegung unterschiedlich, wie auf der Abb. 3.11 [b] zu sehen ist [17].

Bei der Pinbelegung (Abb. 3.11 [b], Seite 33) steht u. a. der BYPASS-Pin. Dieser kann optional, mithilfe eines im Nanobereich liegenden Kondensators, Spannungsschwingungen (Rauschen) reduzieren, woraufhin höhere Stabilität des Regulators erreicht wird. Der EN-Pin dient dem Ein- und Ausschalten des LDO's. Steigt die Spannung (V_{EN}) über 1,4V (*high*) beginnt der Einschaltvorgang des IC's. Sinkt die Spannung unter 0,4V (*low*) schaltet es sich aus.



(a) Gehäuse (b) Gehäuse (Datenblatt)

(c) Pinbelegung

Abbildung 3.11: (a) Ein LP3985IM5-2.7 Spannungswandler (b – oben) Topview eines LDO zur 2,7V Konvertierung im SOT-23-Gehäuse (LP3985IM5-2.7) (b – unten) und die SMD-Variante davon (b) Pinbelegung, und vom Entwickler beiliegende Beschaltung des LDO [17]

3.1.9 Pegelwandler

Aus Gründen von unidirektionaler oder bidirektionaler Datenkommunikation unterschiedlich arbeitender Betriebsspannungen, z. B. zweier Systeme, wird zur Handhabung dieses Falles ein Pegelwandler, auch *high/low-levelshifter* genannt, angewandt. Ohne eines solchen käme es zu Datenkorruption oder schlimmstenfalls Datenverlust. Dies ergibt sich, da der Spannungspegel erst nach Erreichen eines bestimmten Wertes als *high* bzw. *low* gewertet wird und bei unterschiedlichen Spannungen auch unterschiedliche Grenzen für eine klare Annahme beider Zustände definiert sind. Deshalb müssen bei der Suche eines Wandlers u. a. ansprechende Werte bei den *High-level*- (V_{IH}, V_{OH}) sowie *Low-level*-Spannungen (V_{IL}, V_{OL}) erreicht werden. Texas Instruments bietet hier aus dem Sortiment einen Pegelwandler (TXB0104GYR) für Spannungen zwischen 1,2V – 3,6V, am Port A, und auf der anderen Seite 1,65V – 5,5V, am Port B, an, der für unsere Zwecke bidirektionale Datenkommunikation zulässt und „gute" Werte für die beiden Pegelzustände der 5V und 2,7V Spannung erreicht (siehe Gleichungen 3.1, 3.3 und Abb. 3.12 [Seite 34]). Die Abbildung zeigt eine Pegelnegierung von 5V auf 2,7V und die *Low*- und *High-level*-Spannungen, des Ein- und Ausganges, für den Pegelwandler. Der in der zu entwickelnden Low-Energy-Platine genutzte ATmega328PB-MU definiert einen ausgehenden High-Pegel bei 3V Versorgungsspannung ab 2V und einen Low-Pegel bei max. 0,7V. Eingehende High-Pegel sind bei der Platinenversorgungsspannung von 2,7V bei 1,62V und der Low-Pegel bei 0,81V. Die am außenstehenden System benutzten Mikrocontroller sind mit denen vergleichbar, wo die berechneten Werte als „gut" beurteilt wurden. Ohne jegliche verbaute Widerstände sind dies bei z. B. $V_{IL} = 1,75V$ gleich 1,05V *overhead*, zur min. definierten Spannung eines ausgehenden und 0,94V eines eingehenden *Low*-Pegels des Mikrocontrollers. Da der gewählte Pegelwandler automatisch die Kommunikationsrichtung angibt, wird dafür eine $\pm$2mA *drive-strength*, pro I/O-Pin, zum Signalisieren des zurzeit aktiven Einganges gefordert. Die Portseite, die diese liefert, gilt dann als Eingang [16].

$$V_{IH} = 0,65 \cdot V_{CCI} = 0,65 \cdot 5V = \underline{\underline{3,75V}} \tag{3.1}$$

$$V_{OH} = 0,65 \cdot V_{CCO} = 0,65 \cdot 2,7V = \underline{\underline{1,755V}} \tag{3.2}$$

$$V_{IL} = 0,35 \cdot V_{CCI} = 0,35 \cdot 5V = \underline{\underline{1,75V}} \tag{3.3}$$

$$V_{OL} = 0,35 \cdot V_{CCO} = 0,35 \cdot 2,7V = \underline{\underline{0,945V}} \tag{3.4}$$

V_{IH} und V_{OH} sind die mindest definierten Ein- und Ausgangspegel für ein high-Signal, V_{IL} und V_{OL} sind die mindest definierten Ein- und Ausgangspegel für ein low-Signal, V_{CCI} und V_{CCO} die Ein- bzw. Ausgangsspannungen

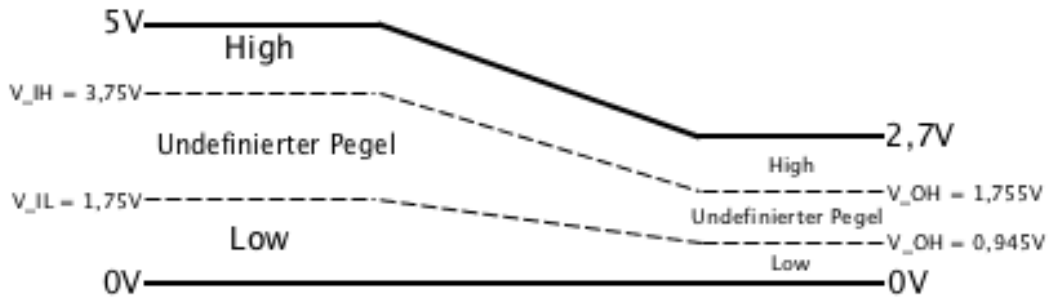


Abbildung 3.12: Die errechneten $V_{IH}(V\_IH), V_{OH}(V\_OH)$- und $V_{IL}(V\_IL), V_{OL}(V\_OL)$-Spannungen, bei 5V Eingangsspannung und 2,7V Ausgangsspannung, des Pegelwandlers: TXB0104GYR mit resultierenden Kennzeichnungen für Pegelzustände am Eingang: „Low" < 1,75V < „Undefinierter Pegel" < 3,75V < „High" und Ausgang: „Low" < 0,945V < „Undefinierter Pegel" < 1,755V > „High'

Der Pegelwandler besitzt zusätzlich zu den beiden Ports (A, B) und Spannungsversorgungen (V_{CCA}, V_{CCB}) einen OE-Pin, welcher zum Ein- bzw. Ausschalten des IC's, vom Port A gespeist, benutzt wird (siehe Abb. 3.14 auf Seite 36). Im ausgeschalteten Fall (OE gleich *low*), des an Port A hängenden Systemes, werden die Pins (A:1 – 4) in den *high-impedance* Zustand gebracht. OE *high* aktiviert den IC.

Für das Gehäuse wurde eines vom Typen VQFN, mit den Maßen 3,5mm x 3,5mm, genommen, wie auf der Abb. 3.13 zu sehen ist [16].

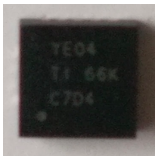


Abbildung 3.13: Ein Pegelwandler im VQFN-Gehäuse (TXB0104GYR) für Ein- und Ausgangsspannungen von 1,2V – 3,6V und 1,65V – 5,5V

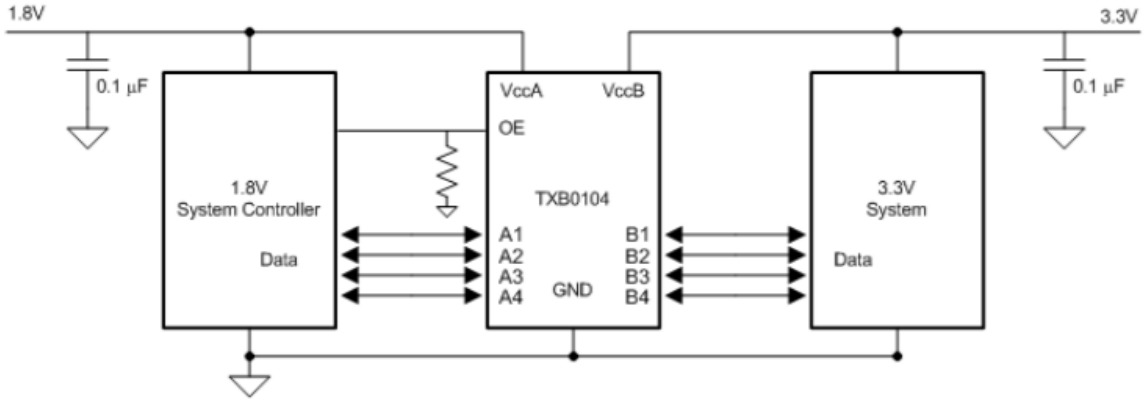


Abbildung 3.14: Vom Entwickler beiliegende, beispielhafte Beschaltung zweier Systeme, unterschiedlicher Betriebsspannungen, mit dem Pegelwandler TXB0104 [16]

3.2 Zusammengefasste Spezifikationen

Die zusammengefassten Spezifikationen dienen lediglich einer generellen Übersicht aller Komponenten (Widerstände, Kondensatoren, Dioden, Transistoren nicht inbegriffen) und derer wichtigen, nennenswerten Details sowie deren Nutzen.

In der Tabelle 3.2, auf Seite 37, wurden Abkürzungen für die wichtigsten elektrischen Eigenschaften benutzt, welche hier erklärt werden.

- *OR* steht für **Operating Range** und gibt den Betriebsspannungsbereich an
- *PC* steht für **Power Consumation** und gibt den vermeintlichen Stromverbrauch an
- *In* steht für **Input** und gibt den Eingangs-Betriebsspannungsbereich an
- *Out* steht für **Output** und gibt den Ausgangs-Betriebsspannungsbereich an
- *wcc* steht für ***worstcase*-current** und gibt den höchst anfallenden Strom an
- *MaxD* steht für **Maximum Dropout** und gibt den minimal erlaubten Unterschied zwischen In und Out an
- *C* steht für die **Kapazität** und gibt den Stromverbrauch, pro Stunde, an

	ATmega328PB *(8-Bit Mikrocontroller)*	**RN4871** *(BLE-Modul)*	**TXB0104** *(Pegelwandler)*	**LP3985** *(LDO)*	**W25Q80DV** *(Flash-Speicher)*	**PCF8563** *(RTC)*	**CR2032** *(Batterie)*
Verwendungszweck	Verbindungseinheit und Softwarehalter aller Komponenten der Low-Energy-Platine	Modul zur Übertragung von Daten im frei frequenten Bereich	Wandler, der Spannungen für Kommunikationsschnittstellen umwandelt	Wandelt extern ankommende Spannungen auf einen spezifischen Wert um	Einfache Speichereinheit zur Sicherung kleiner Datenmengen	Komponente zur Echtzeitgenerierung	Batterie für die Versorgung der RTC im betriebsspannungslosen Fall
Elektrische Eigenschaften	*OR*: 2,7V – 5,5V (0 – 10MHz) *PC*: 5,2mA bei (8MHz, V_{CC} = 5V) *wcc*: 10mA bei (8MHz, V_{CC} = 5V) V_{OL}: 0,7V V_{OH}: 2V bei V_{CC} = 3V	*OR*: 1,9V – 3,6V *PC*: 0,23mA – 3,9mA (V_{CC} = 3,3V) *wcc*: 13mA (V_{CC} = 3.3V)	*OR1*: 1,2V – 3,6V *OR2*: 1,65V – 5,5V V_L: $0,35 \cdot V_{CC}$ V_H: $0,65 \cdot V_{CC}$	*In*: 2,5V – 6V *Out*: 2,7V *MaxD*: 100mV *voc*: 150mA	*OR*: 2,7V – 3,6V *PC*: 10μA – 20mA *wcc*: 25mA	*OR*: 1,8V – 5,5V bei 400kHz *PC*: 200μA – 800μA bei arbeitender RTC	*C*: 230mA/h bei 3V
Sonstiges	27 I/O-Pins, Je zwei serielle USART, SPI und I^2C-Inferfaces	ISM Band 2,402 zu 2,480GHz, UART, Bluetooth 4,2			1MByte groß, SPI, Löschvorgänge 4KB, 32KB oder 64KB	basiert auf 32,768kHz, I^2C	
Referenzen	3.1.5 (Seite: 28)	3.1.1 (Seite: 22)	3.1.9 (Seite: 34)	3.1.8 (Seite: 3.1.8)	3.1.3 (Seite: 25)	3.1.2 (Seite: 25)	3.1.4 (Seite: 27)

Tabelle A.2: Zusammengefasste Spezifikationen aller Hauptkomponenten der zu entwickelnden Low-Energy-Platine

3.3 Schaltplan

Im folgenden wird es nur um das Schaltplan-Design, mit dem Hilfsprogramm EAGLE, im *Schematic*-Editor gehen, wobei auf alle verwendeten Komponenten sowie Sekundärschaltungen explizit eingegangen wird. Für die Sekundärkomponenten[1] wurden welche der vielzählig, bei bekannten Distributoren, auf Lager verfügbaren gewählt und werden weder vom Typ noch anderweitig genauer betrachtet. Auf die eigenen EAGLE-Funktionen wird nicht eingegangen sondern ggf. lediglich allgemein erläutert.

Der komplette Schaltplan wird auf Seite 52 (Schaltplanseite 1/2) und 53 (Schaltplanseite 2/2) gezeigt:

Allgemeines zur Schalplanrealisierung

Ein Beispiel zur Komponentenerstellung in EAGLE wird auf der Abb. 3.2, auf der Seite 23, zum BLE-Modul gegeben.

- Jede Hauptkomponente besitzt einen aufsteigenden Indizie[2]: ICx, Sekundärkomponenten: Rx [Widerstand], Cx [Kondensator], LEDx [Diode] und Qx [Quarz oder Mosfet], Zusätzliche Hauptkomponente Gx [Batterie]
- Offene, unbenutzte **Pins** wurden mit einem Testpad (TPx) versehen, um die später mögliche Verwendung der dahinterliegenden Funktionen zu ermöglichen
- **Pins** wurden nicht nach Pinnummer, sondern nach der Bezeichnung, von der Oberseite betrachtet, absteigend sortiert, um gerade bei großen IC's einen Überblick zu gewährleisten
- **Pins** wurden nach Nutzungszweck (**Power-Pin** [z. B. Gnd oder V_{CC}], **I/O-Pins** [z. B. CLK oder (XTAL1/TOSC1)PB6]) sortiert, um ein Durcheinander bei der Beschaltung zu vermeiden
- Jede Hauptkomponente bekommt mindestens einen Blockkondensator[3] zur Spannungsversorgung und Ground parallelgeschaltet
- Miteinander verbundene Komponenten sind auf den Abbildungen durch Namensschilder voneinander getrennt, wobei die ggf. vorhandenen Zahlenfolgen hinter dem '/' (z. B. RTC_SCL/1.3B) für die Schaltplanseite (1), folgend von der vertikalen (3) und der horizontalen Position (B) stehen

[1] Sekundärkomponenten bezeichnen Widerstände, Kondensatoren, Dioden oder Transistoren

[2] Das kleine 'x' (z. B. bei TPx) repräsentiert eine Nummerierung in EAGLE von '1' ausgehend

[3] Der Blockkondensator, auch „Angstkondensator" genannt, dient dem Abfangen von Spannungsspitzen beim u. a. Beschalten der Ports eines IC's

3.3.1 ATmega328PB

Der ATmega328PB wird die Koordinationseinheit der Low-Energy-Platine und bietet an den 27 benutzbaren I/O-Pins eine Vielzahl an alternativen Verwendungsmöglichkeiten, hingegen einer einfachen General-purpose Input/Output (GPIO)-Verwendung. Betrachtet man das „Block Diagramm des ATmega328PB-Mikrocontrollers [2]" genauer, so fällt z. B. auf, dass dies eine Komplettübersicht über alle anschließbaren Pins ist oder/und die innere Kommunikation zur CPU veranschaulicht wird. Nicht zu sehen ist, dass die bidirektional verwendbaren I/O-Pins optional je einen internen, fixen, *pull-up*-Widerstand haben oder alternative Funktionen, und diese in der Software eingestellt werden müssen.

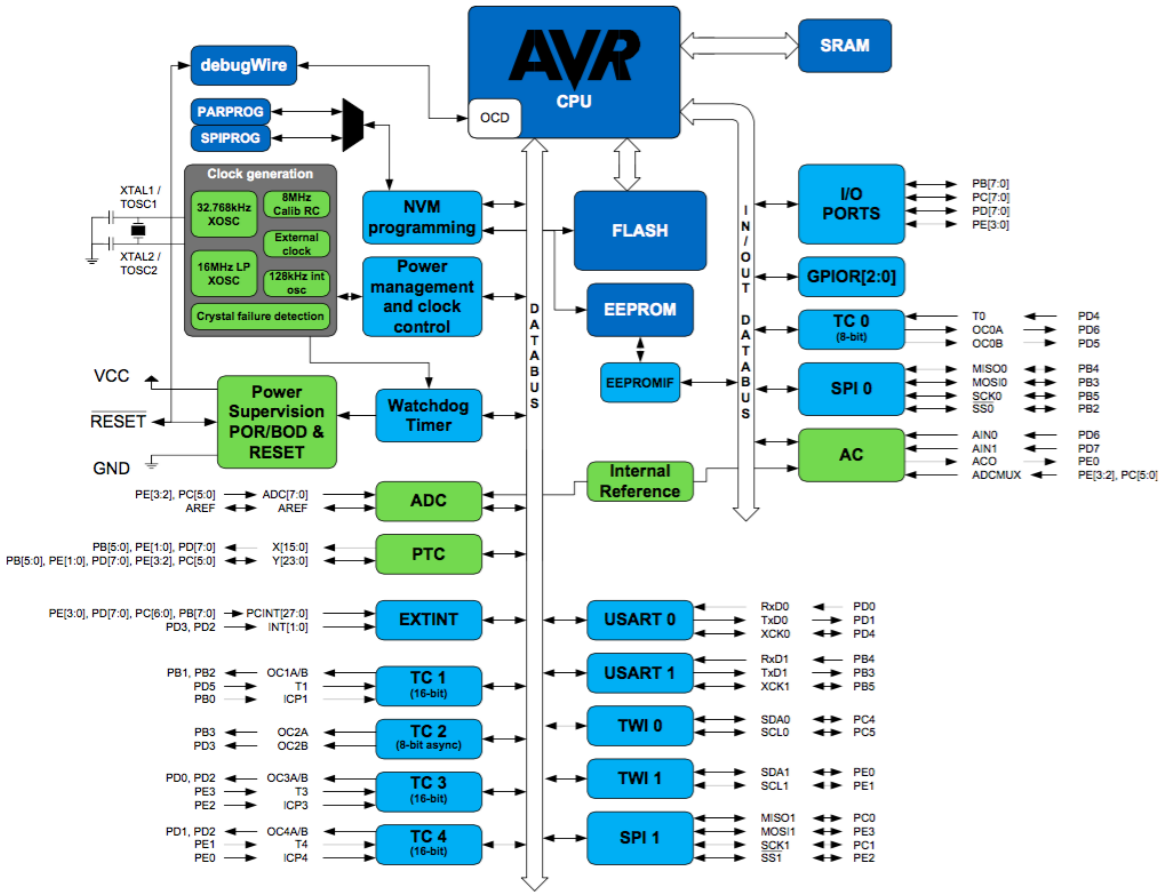


Abbildung 3.15: Block Diagramm des ATmega328PB-Mikrocontrollers [2]

Beginnend wurde die linke Seite des IC's (Abb. 3.16) beschaltet, welche sich auf die **Power**-Pins bezieht. VCC dient der normalen Spannungsversorgung des Bauteiles, AVCC der separaten Versorgung der nicht benutzten ADC's und AREF der Referenzspannung dieser. Trotz Nichtbenutzung des Analog-Wandlers wird der Pin mit einem Blockkondensator (C14 = 100nF) auf GND[4] gelegt, um Rauschen zu kompensieren. VCC bekommt zum Blockkondensator (C1) einen zusätzlich gleichwertigen Kondensator (C13), der für die zweite Versorgungsspannung AVCC zuständig ist. Von den, auf der rechten Seite paarweise angeordneten, Ports wurden die Pins (XTAL1/TOSC1)PB6 und (XTAL2/TOSC2)PB7 an den externen Quarz (ABMM2-7.3728MHz-E2-T) angeschlossen, die dazu zwei typen- und wertengleiche (27pF) Trimmkondensatoren (C15, C16, berechnet in Gleichung 3.5) zur Schwingkreisstabilisierung in Reihe geschaltet bekommen.

$$C15 = C16 = 2 \cdot (C_{load} - C_{stray}) = 2 \cdot (18pF - 4,5pF) = \underline{\underline{27pF}} \tag{3.5}$$

C_{load} ist die aus dem Datenblatt entnommene Ladekapazität, C_{stray} die Streukapazität, die sich zwischen 2pF – 5pF aufhält und sich aus bewährten Werten erschloss [1].

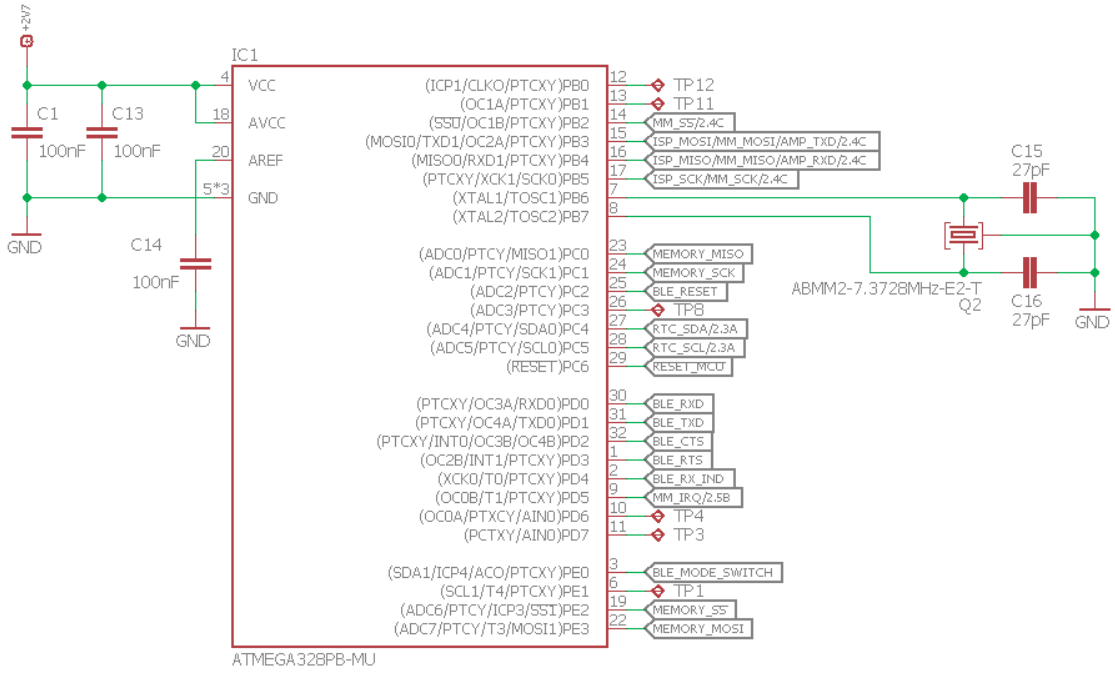


Abbildung 3.16: Fertiger Schaltplanpart des ATmega328PB-Mikrocontrollers im *Schematic*-Editor

[4]GND 5*3 steht für die gemeinsam auf Ground geschalteten Pins 5 und 3

3.3.2 RN4871

Bei dem BLE-Modul RN4871 (siehe Abb. 3.17 auf Seite 42) wird ein vom Entwickler zusätzlich verlangter Blockkondensator (C17 = 10μF) parallel zu VBAT und GND eingebaut, da das Modul einen durch die Bluetoothfunktionen behafteten impulsartigen Stromverbrauch aufweisen kann. Der $\overline{RST}$-Pin (Reset) muss wegen der Invertierung, bzw. da dieser *activ low* ist, durch einen *pull-down* Widerstand von 4,7kΩ gestützt werden, um die Anbindung an einen unbenutzten I/O-Pin des Mikrocontrollers zu stabilisieren. Dieser benutzte I/O-Pin hat einen internen *pull-up* Widerstand von 20kΩ – 50kΩ. Der interne 4,7kΩ, der E12 Reihe, wurde gewählt, da die Reset *threshold*-Spannung des BLE-Moduls bei 1,6V definiert ist und das Ergebnis des Spannungsteilers 3.6 ist. Damit wird ein niedriger Wert von 0,5V erreicht, welcher definitiv als *low* gewertet werden kann. Das Reset wird vom Mikrocontroller getätigt. UART_TX (Out) und UART_RX (In) sind die Sende- und Empfangsleitungen des asynchronen UART-Inferfaces. CTS/P0_0 (I/O) und RTS/P3_6 (I/O) gehören ebenso dazu und werden im Softwarepart erläutert.

$$V_p = \frac{V_{CC} \cdot R_3}{R_{mc} + R_3} = \frac{2,7V \cdot 4,7k\Omega}{20k\Omega + 4,7k\Omega} = \underline{\underline{0,5V}} \tag{3.6}$$

V_{CC} ist die Versorgungsspannung, R_{mc} der interne pull-up Widerstand des Mikrocontrollers, R_3 der pull-down Widerstand des BLE-Modulresets und V_p die Ausgangsspannung vom Mikrocontroller zum Modul.

Microchip empfiehlt zur Signalisierung des Ein-/Auszustandes den Einbau einer Diode (LED1) in Reihe eines vordefinierten Widerstandes (R6 = 330Ω) an den Pin LED/P0_2. Das Datenblatt empfiehlt dabei eine Mindestspannung von 3V, wobei dort davon ausgegangen wird, dass die Diode blau ist. Die eingebaute rote Diode hingegen, kommt mit niedrigeren Spannungen aus. Eine zusätzliche Berechnung des Vorwiderstandes, in der Gleichung 3.7, bestätigt den Einbau des vorsichtshalber höher gewählten Widerstandes von 330Ω.

$$R6 = \frac{V_{CC} - V_F}{I_F} = \frac{2,7V - 2V}{25mA} = \underline{\underline{280\Omega}} \tag{3.7}$$

V_F ist die Durchlassspannung und I_F der Durchlassstrom der Diode [12].

Um das Modul ständig im sogenannten APP Mode zu belassen und keinesfalls den Test Mode zu nutzen, wird der Pin P2_0/Mode direkt *high* geschlossen. Der Test Mode wäre zum Testen der Beschaltung und Vermeidung einer möglichen Beschädigung des Moduls. Der APP Mode stellt das Innere des Moduls hingegen in den normalen Verwendungsmodus [9].

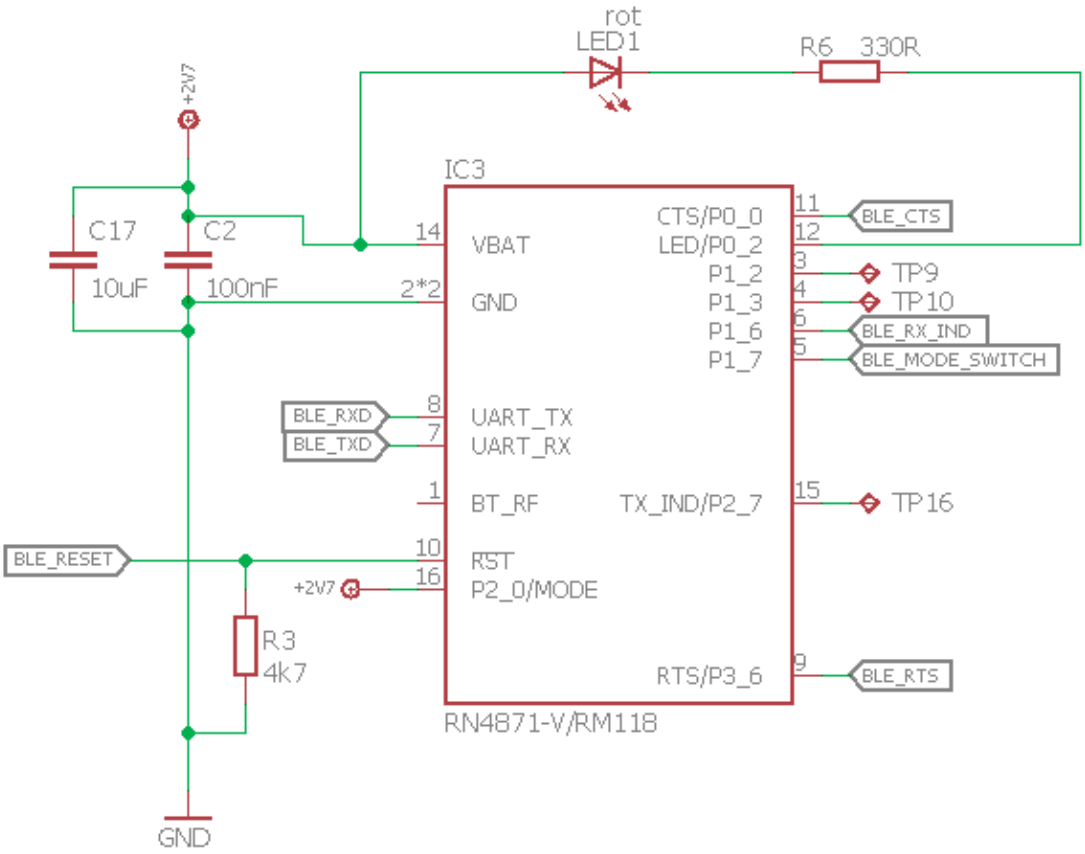


Abbildung 3.17: Fertiger Schaltplanpart des RN4871-BLE-Moduls im *Schematic*-Editor

3.3.3 W25Q80DV

Im Schaltplanpart der Speichereinheit, auf Abb. 3.18, werden der Pin $\overline{HOLD(IO3)}$ und $\overline{WP(IO2)}$ an V_{CC} geschlossen, da beide *active low* sind und nicht benutzt werden.

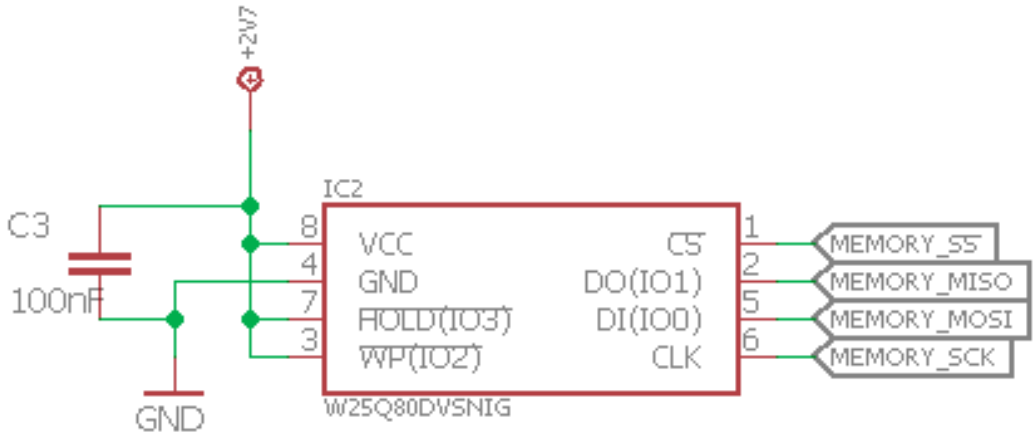


Abbildung 3.18: Fertiger Schaltplanpart des W25Q80DV-Speichers im *Schematic*-Editor

3.3.4 LP3985

Der zur Rauschunterbindung optional einsetzbare Kondensator (C7) wurde mit dem aus dem Datenblatt des LDO's entnommenen Wert für *bypass capacitor* (10nF) zwischen den BYPASS-Pin und GND gesetzt. Die zwei weiteren Kondensatoren (C5 = C6 = 1μF) sind vom Entwickler vorgegeben und von IN, EN und OUT an die jeweilige Betriebsspannung ($VIN = 2,7V$) und Ground geschlossen. Der dazugehörige Schaltplanpart ist auf der nächsten Seite auf der Abb. 3.19 zu sehen.

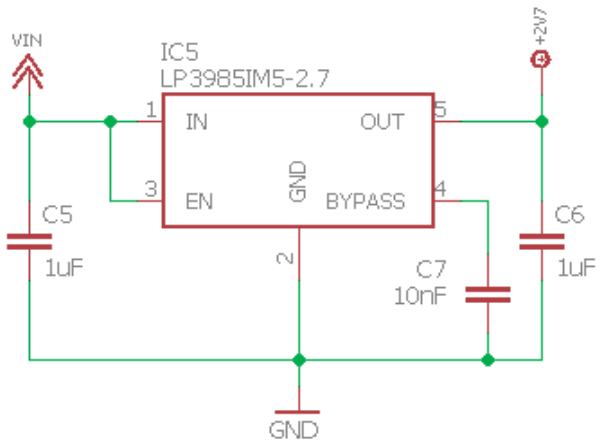


Abbildung 3.19: Fertiger Schaltplanpart des LP3985-Low-Dropout Regulator im *Schematic*-Editor

3.3.5 TXB0104

Die niedere Spannung der Low-Energy-Platine von 2.7V muss und wird an den Port A (VCCA) angeschlossen, woraufhin Port B (VCCB) für die 5V zuständig ist. OE wird durch das *active high*-Verhalten und, da es ein reiner Eingang ist, direkt an die innere Betriebsspannung angeschlossen, um das Ein- und Ausschaltverhalten, von vorhandener Spannung abhängig zu machen (siehe Abb. 3.20 auf Seite 45). Wäre es ein I/O-Pin, sollten *pull-up/pull-down* Widerstände benutzt werden, um z. B. ein in der Software versehentliches auf „Ausgang" setzen zu unterbinden, welches zu Kurzschlüssen führen würde.

Da die berechneten Werte für High- und Lowpegel, Gleichungen 3.1, 3.2, 3.3 und 3.4 auf Seite 34, verlustfrei betrachtet wurden, kann mit der Beschaltung an den AMP- sowie Micromatch-Stecker (siehe Kapitel 3.3.6) und den jeder Leitung in Reihe liegenden Widerstand von 100Ω der zusätzliche Verlust aller vier Pegelspannungen berechnet werden. Das Ergebnis sind 0,2V (Gleichung 3.8), die durch den großen Unterschied des beschriebenen Beispieles (3.1.9, Seite 34), der ein- bzw. ausgehenden Low-Pegel, bei $V_{IL} = 1,75V$, nun noch 0,85V und 0,74V *overhead* aufweisen, statt den vorherigen 1,05V und 0,94V.

$$V_{verlust} = R_{reihe} \cdot I_{ds} = 100\Omega \cdot 2mA = \underline{\underline{0,2V}} \tag{3.8}$$

$V_{verlust}$ ist die errechnete Verlustspannung durch den jeweils in Reihe stehenden Widerstand (R_{reihe}) und der benötigten drive-strength (I_{ds}).

Die vier Pins (2, 3, 4 und 5) des Ports A sind an den Mikrocontroller angeschlossen (Kapitel 3.3.1 der Seite 39) und die Pins (10, 11, 12 und 13) des Ports B an die beiden Stecker, die im nächsten Kapitel 3.3.6 für die Kommunikation nach außen, beschrieben werden.

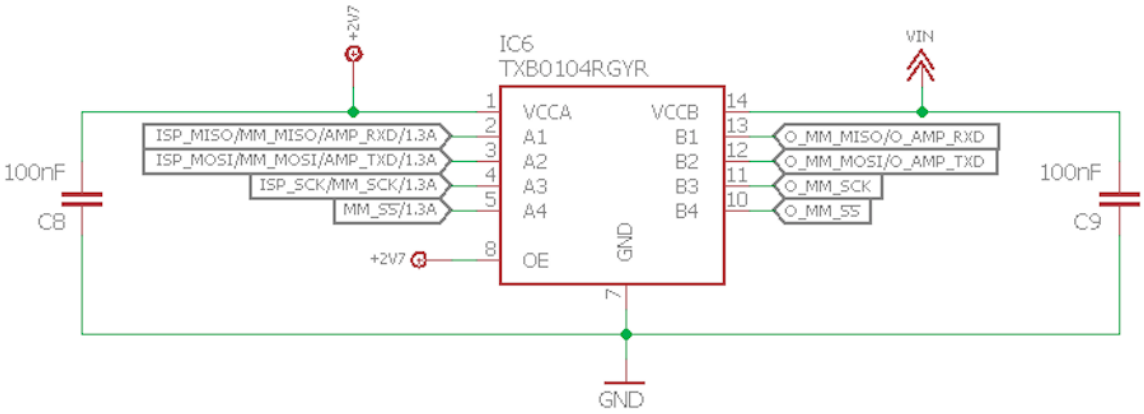


Abbildung 3.20: Fertiger Schaltplanpart des TXB0104-Pegelwandlers im *Schematic*-Editor

3.3.6 AMP- und Micromatch-Stecker

Um gegen vermeintliche Kurzschlüsse und Störungen geschützt zu sein, werden an die beiden Stecker, genauer die Kommunikationsschnittstellenleitungen, 100Ω Widerstände (R1, R2 und RN1[vierpolig]) angeschlossen. Beim Micromatch-Stecker (siehe Abb. 3.22 [b] auf Seite 47) hat die SPI-Leitung $O\_MM\_\overline{SS}$ einen *pull-up* Widerstand von 4,7kΩ (R4) erhalten, da die *active low* Slave Select Leitung nicht durchgehend an, und damit auf *high*, stehen soll. Beide weisen eine genaue Schaltstruktur auf, die in der Firma Scholz System für diese Stecker-Arten so gehandhabt wird und eingehalten wurde (am Ausgang stehende Versorgungsspannung [VIN] an Pin 1 etc.).

Zu beachten ist, dass die Low-Energy-Platine ein weiterer Slave im Inneren einer Kaffeemühle oder eines Zigarettenautomaten ist. Der Interrupt-Pin (MM_IRQ), welcher am Mikrocontroller hängt, dient der Kommunikationsanfrage eines Slaves (z. B. der Low-Energy-Platine) nach außen zu dem Master und muss wegen mehrerer vorhandener Slaves, die die gleiche Interrupt-Leitung nutzen, mit einem Mosfet (IRLML0040) abgesichert werden. Die Sicherung dient vermeintlichen Pegelüberschneidungen. Der am Drain sitzende Widerstand (R11 = 20Ω) schützt vor zu viel Strom (bis zu 45mA bei abgezogener Schwellspannung, berechnet in der Gleichung 3.9, Seite 46) und dient dazu, eine Beschädigung des Transistors zu vermeiden. Der Transistor kann generell aber bis zu $I_D = 350mA$ ab, siehe Abb. 3.21, im untersten Verlauf für $V_{DS} = 2,5V$, auf Seite 46. Der Widerstand R11 wurde, wie aus dem Strom analysiert, sehr großzügig gewählt.

Je mehr Strom fließt, desto geringer wird die Gate-Source-Spannung. Schreitet sie unter die aus dem Datenblatt entnommene Schwellspannung $V_{GS(th)} = 1,8V$, dann sperrt der Transistor. Der am MM_IRQ-Pin sitzende *pull-down* Widerstand (R12) von 4,7kΩ dient dem auf *low*-Ziehen durch den Mikrocontroller. Eine beispielhafte Kommunikationsanfrage an den außenstehenden Master wäre, dass ein Slave den Interrupt-Pin *high* setzt und somit signalisiert (leitender Mosfet zieht den Pegel wiederum *low*), eine Kommunikation beginnen zu wollen. Daraufhin fragt der Master alle anhängenden Slaves durch die Slave Select-Leitung, wer die Anfrage getätigt hat [8].

$$I_{R11} = \frac{V_{CC} - V_{GS(th)}}{R11} = \frac{2,7V - 1,8V}{20\Omega} = \underline{\underline{45mA}} \tag{3.9}$$

I_{R11} ist der maximal auftretende Strom, welcher durch die Versorgungsspannung V_{CC}, subtrahiert mit der Schwellspannung $V_{GS(th)}$ und dividiert durch den großzügig gewählten Widerstand R11, durchgelassen wird.

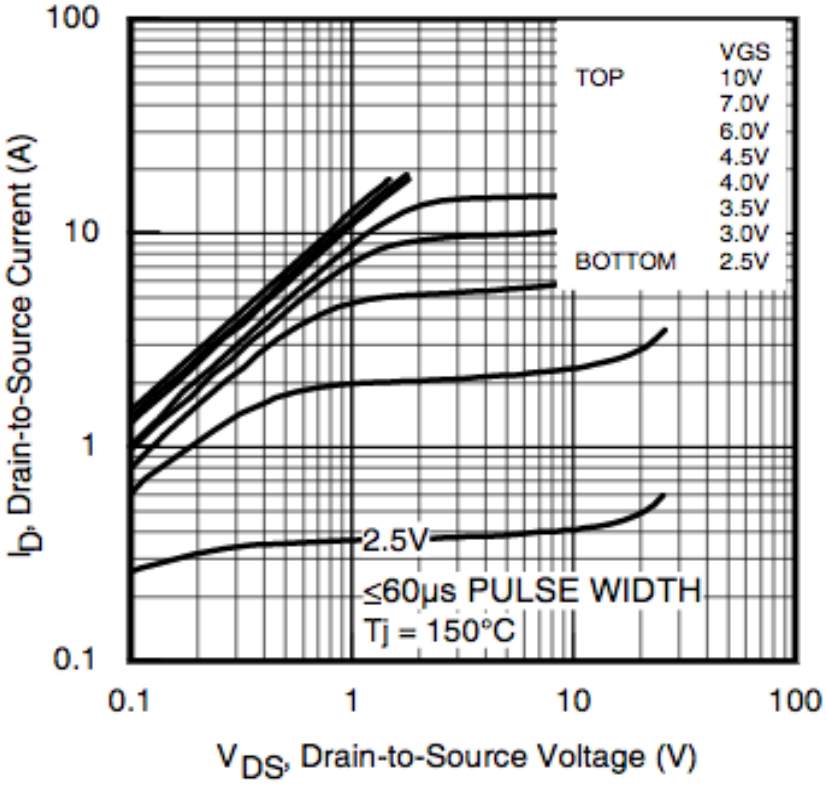
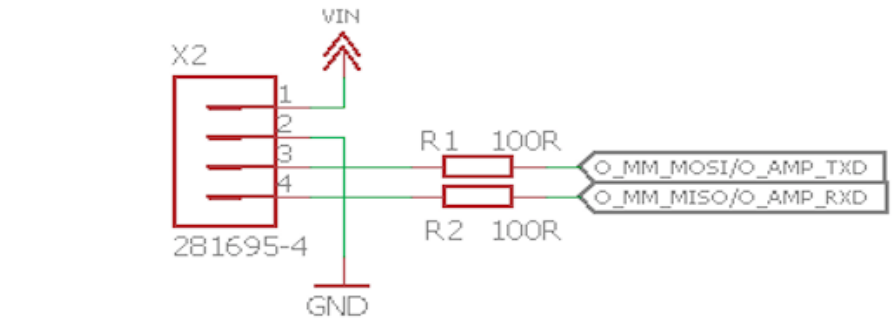
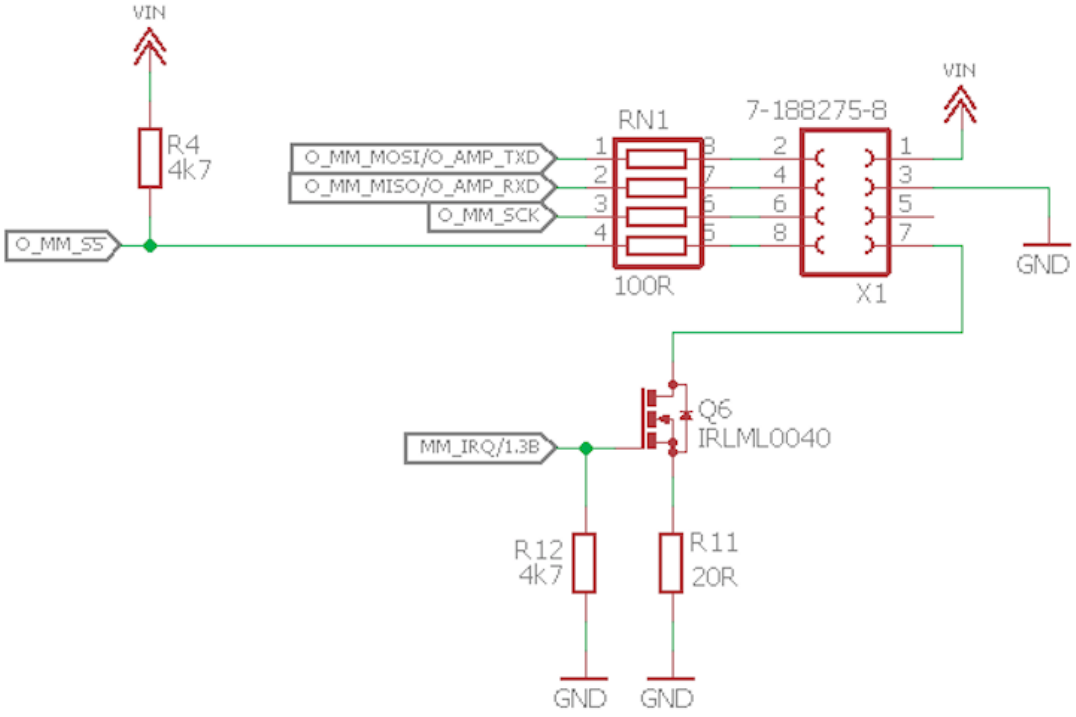


Abbildung 3.21: Typische Ausgangscharakteristiken [8]



(a) AMP (X2)



(b) Micromatch (X1)

Abbildung 3.22: (a) Fertiger Schaltplanpart des Micromatch-Steckers im *Schematic*-Editor (b) Fertiger Schaltplanpart des AMP-Steckers im *Schematic*-Editor

3.3.7 PCF8563

Die in der auf Abb. 3.23 zu sehende RTC wird an die OSCI- und OSCO-Pins der 32,768kHz-Quarz angeschlossen (CRYSTALMM20SS). VSS ist die „negative Versorgungsspannung" oder einfach GND. Im Datenblatt der RTC und I^2C-Spezifizierung wird gefordert, die SCL- und SDA-Leitungen an die Betriebsspannung, durch einen *pull-up* Widerstand, zu schließen (siehe Gleichung 3.10). Daraus schließend wurde sich für ein Widerstandswert von 2kΩ entschieden. Diese beiden Widerstände halten die Leitungen bei Nichtverwendung *high*. $\overline{INT}$ und CLKOUT werden offen gelassen und nicht verwendet. Die Versorgungsspannung (VDD) soll entweder aus der Batterie stammen (3V) oder aus der normalen Betriebsspannung von 2,7V und wird ab Seite 49 näher erläutert [11].

$$R_{pu} = \frac{t_r}{C_b} = \frac{1\mu s}{400pF} = \frac{1\mu s\Omega}{400ps} = \underline{\underline{2,5k\Omega}} \tag{3.10}$$

R_{pu} ist der maximal wählbare pull-up Widerstandswert, aus der definierten Zeitkonstante t_r, welche die rise time für SDA und SDL für den normalen Übertragungsmodus angibt und der, für beide Leitungen definierten, je, kapazitiven Ladung C_b.

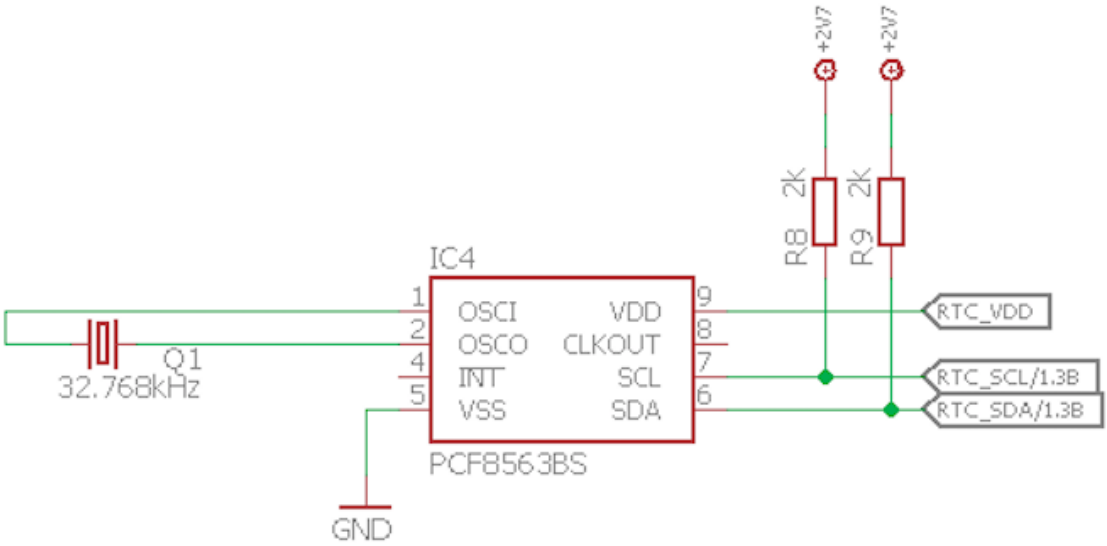


Abbildung 3.23: Fertiger Schaltplanpart der Real-Time-Clock im *Schematic*-Editor

3.3.8 CR2032

Da die Batterie eine höhere Spannung als die Betriebsspannung liefert und vor allem bei unbeachtetem gleichzeitigen Betrieb (nur mit Dioden abgesichert) zuerst die Batterie erschöpft, wurde eine Schaltung aus den auf Abb. 3.24, Seite 50, zu sehenden P-kanal-Mosfets (IRLM6402) gebaut. Diese erbringt durch die gegebene Schwellspannung ($V_{th(min)} = -0,4V$) ein ideales Schaltverhalten zur Differenz beider Spannungen und nutzt bei gleichzeitigem Betrieb die standhafte Versorgungsspannung von 2,7V. Betrachtet man den ersten der beiden eintreffenden Fälle (Fall 1: Betriebsspannung AN [$V_{CC} = 2,7V$]), so sperren die beiden Transistoren Q7 und Q3, woraufhin Q5 und Q4 leiten. Dies ist so, da bei dem ersten eine Gate-Source-Spannung von $V_{GS(Q7)} = -0,3V$ und bei dem zweiten von $V_{GS(Q3)} = 0V$ anliegt. Das Verhältnis $U_{GS} < U_{th}$ definiert dabei den Fall für das Leiten, der inverse Vergleichsoperator davon, das Sperren. Die Gate-Source-Spannung bei Q5 und Q4 ist -2,7V (Gate-Spannung 0V) und somit kleiner als -0,4V, was das Leiten begründet. Für den zweiten Fall (Fall 2: Betriebsspannung AUS [$V_{CC} = 0V$]) sind Q7 und Q3 leitend und Q5 und Q4 sperrend (siehe Tabelle A.3). Die Widerstände R10 = R7= 100kΩ dienen dazu, den undefinierten Pegel auf Null zu ziehen. Beim ersten Fall, beispielhaft, werden die beiden Drain's von Q7 und Q3, sowie Gate's von Q5 und Q4 stromsparend auf Null gezogen [7].

Fall	**Name**	**Zustand**
$V_{CC} = 2,7V$	$V_{GS(Q7)} = -V_{CC(bat)} + V_{CC} = -3V + 2,7V = \underline{\underline{-0,3V}}$	sperrt
	$V_{GS(Q3)} = V_{CC} - V_{CC} = 2,7V - 2,7V = \underline{\underline{0V}}$	sperrt
	$V_{GS(Q5)} = V_{Q7\&Q3} - V_{CC} = 0V - 2,7V = \underline{\underline{-2,7V}}$	leitet
	$V_{GS(Q4)} = V_{Q7\&Q3} - V_{CC} = 0V - 2,7V = \underline{\underline{-2,7V}}$	leitet
$V_{CC} = 0V$	$V_{GS(Q7)} = -V_{CC(bat)} + V_{CC} = -3V + 0V = \underline{\underline{-3V}}$	leitet
	$V_{GS(Q3)} = V_{CC} - V_{Q7\&Q3} = 0V - 3V = \underline{\underline{-3V}}$	leitet
	$V_{GS(Q5)} = V_{Q7\&Q3} - V_{CC} = 3V - 0V = \underline{\underline{3V}}$	sperrt
	$V_{GS(Q4)} = V_{Q7\&Q3} - V_{Q7\&Q3} = 3V - 3V = \underline{\underline{0V}}$	sperrt

Tabelle A.3: Zusammenfassung der P-Kanal-Mosfet Zustände für die beiden Fälle (Fall 1: Betriebsspannung AN [$V_{CC} = 2,7V$], Fall 2: Betriebsspannung AUS [$V_{CC} = 0V$]), hinsichtlich der Gate-Source-Spannung $V_{GH(Qx)}$, wobei die Spannung $V_{Q7\&Q3}$ die Source- (Q3) oder Drain-Spannung von Q7 und Q3 ist

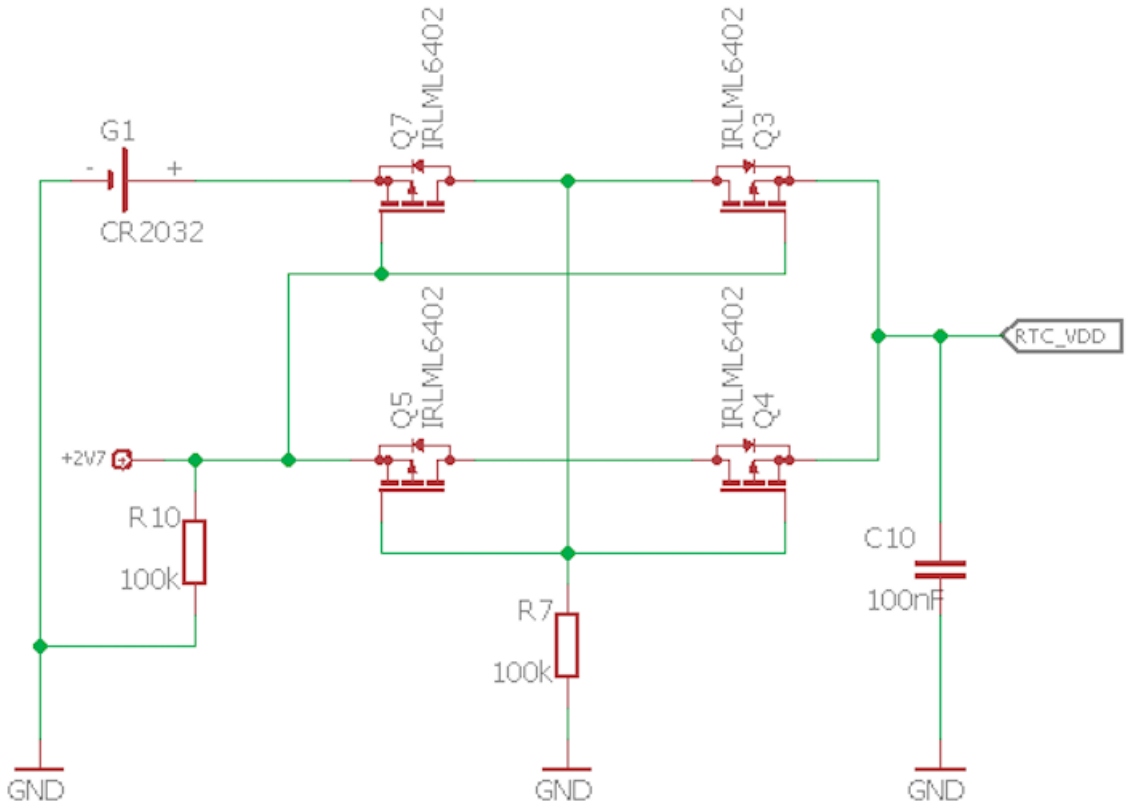
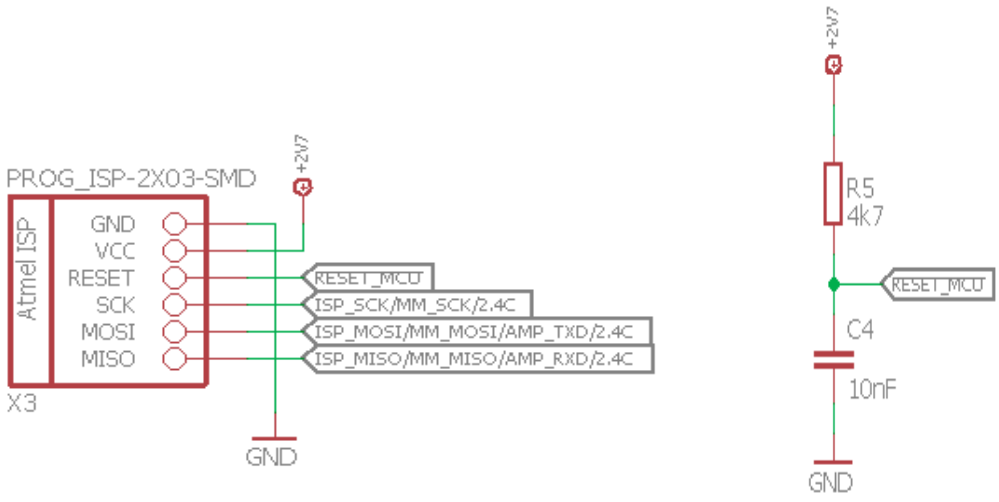


Abbildung 3.24: Fertiger Schaltplanpart der Batterie im *Schematic*-Editor

3.3.9 PROG_ISP und Mikrocontroller-Reset

Vom Datenblatt des ATmega328PB werden die dortigen Pins 15, 16 und 17 für die Beschreibung des Flashs via der, auf der Abb. 3.25 (a) zu sehenden, sechsfach Programmierschnittstelle definiert und an die drei Pins SCK, MOSI und MISO angeschlossen. Die Programmierschnittstelle muss vor jedem Neubeschreiben einen Reset einleiten, welcher im ATmega328PB *active low* ist und somit bei Werksbetrieb *high* sein muss. Daraufhin wird der auf Abb. 3.25 (b) zu sehende Tiefpassfilter, mit einem Widerstand (*pull-up*) von R5 = 4,7kΩ und einem Kondensator C4 = 10nF parallel gesetzt, benutzt. Dieser dient dazu, möglicherweise auftretende Impulse zu filtern, welche einen unbeabsichtigten Reset einleiten könnten. Das Auftreten dieser Impulse hängt größtenteils von störfähigen (rauschfähigen) Geräten ab. Ein ISP-Programmiergerät, welches zur Beschreibung verwendet wird, weiß nicht, was eine Slave Select-Leitung der SPI-Schnittstelle ist und beansprucht, ohne Nachfrage, bei der Beschreibung den SPI-Bus für sich. Für Testzwecke wird ein Jumper am AMP-Stecker (*male*) angelötet, um ein schnelles Wechseln zum Flashen zu ermöglichen.



(a) ISP (b) Reset

Abbildung 3.25: (a) Fertiger Schaltplanpart des ISP's im *Schematic*-Editor (b) Fertiger Schaltplanpart des Resets im *Schematic*-Editor

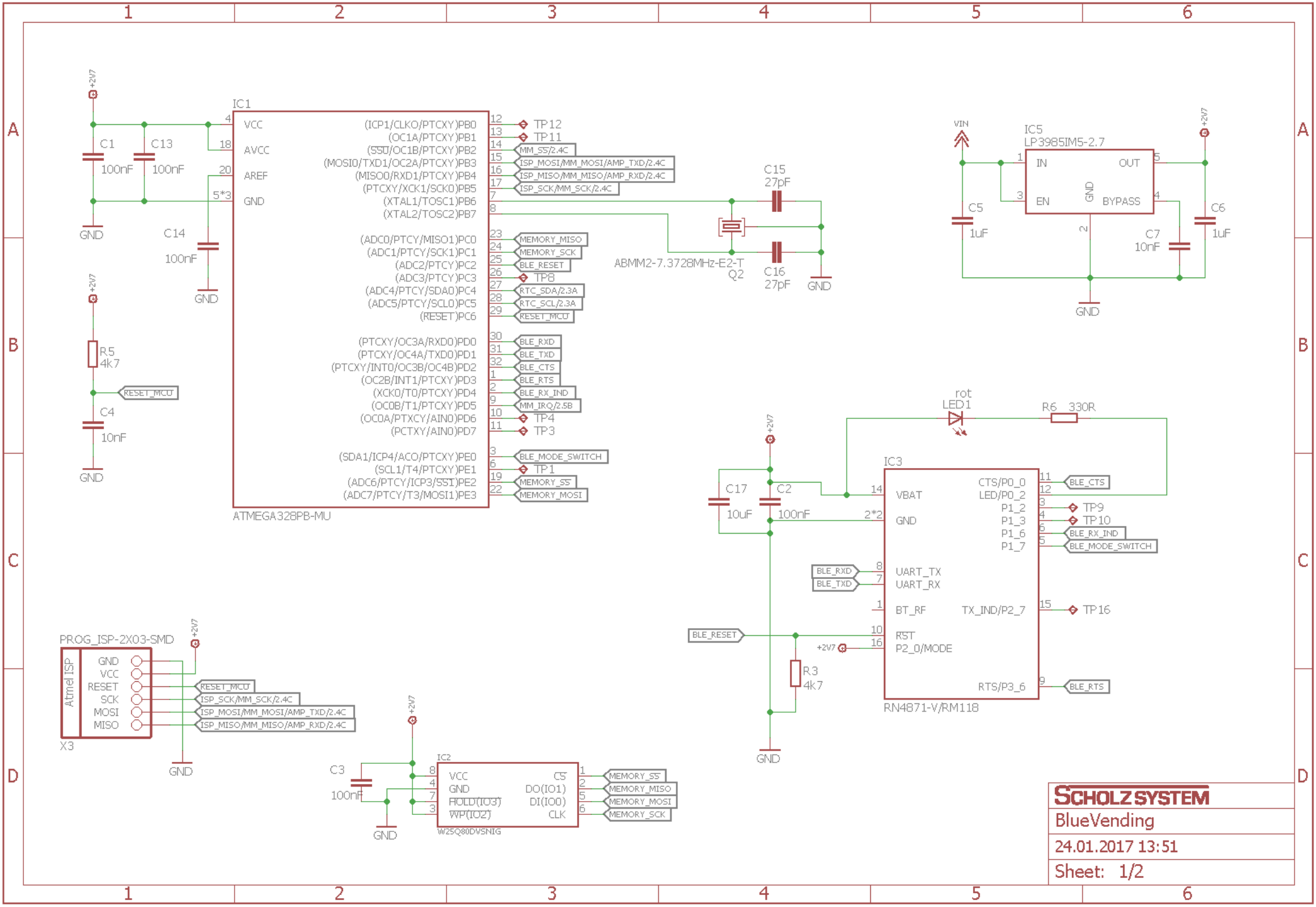


Abbildung 3.26: Kompletter Schaltplan der Low-Energy-Platine in EAGLE (Seite 1/2)

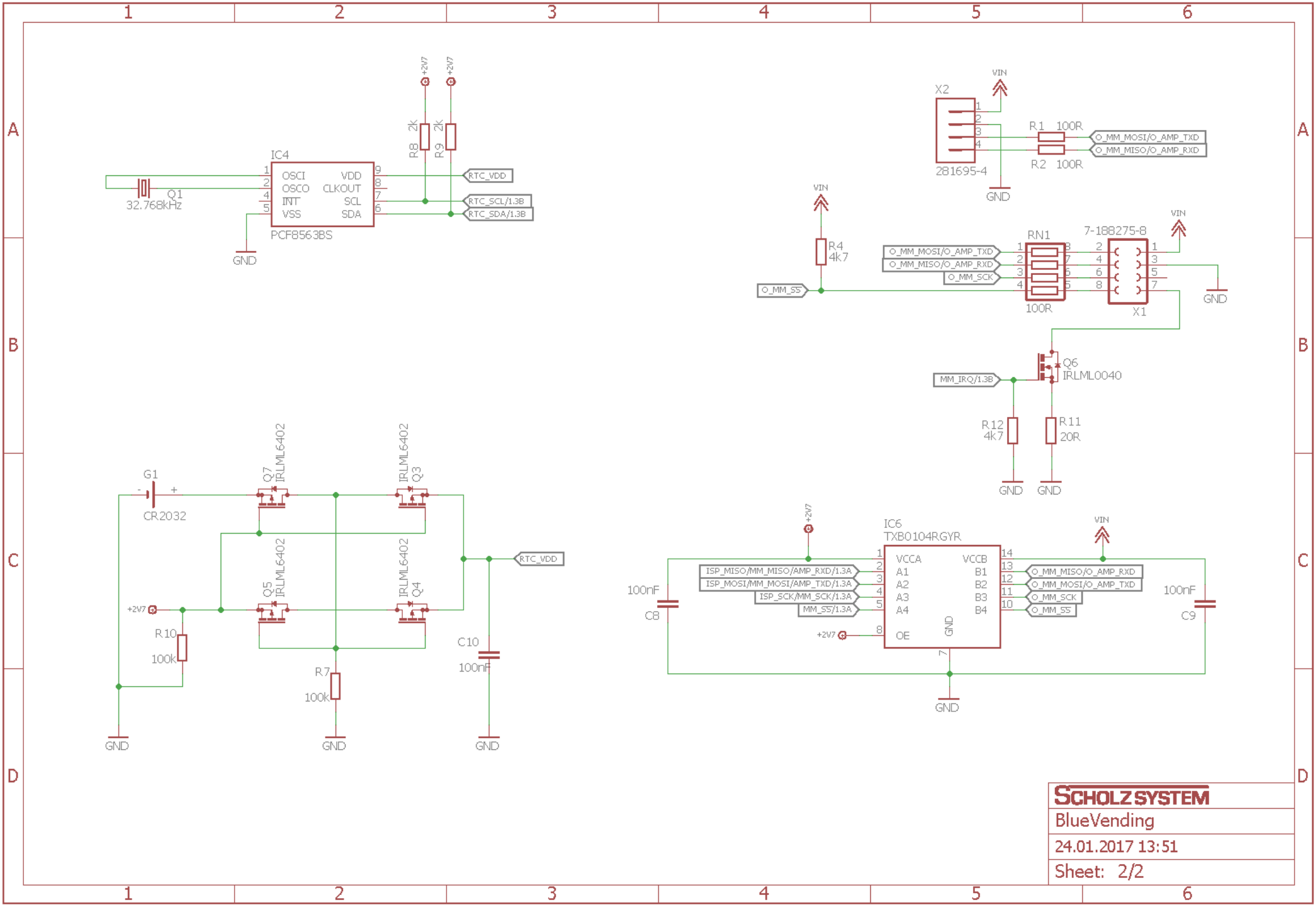


Abbildung 3.27: Kompletter Schaltplan der Low-Energy-Platine in EAGLE (Seite 2/2)

3.4 Platine

Folgend geht es um das Platinen-Design, welches mit dem Hilfsprogramm EAGLE im *Layout*-Editor erstellt wurde, wobei außerhalb der Betrachtung der fertigen Platine an sich, auch auf die Verfahrensweise beim Designen eingegangen wird (erstes Borddesign) und entstandene Herausforderungen erläutert werden.

3.4.1 Anforderungen an die Platine

Folgende Anforderungen wurden für den Aufbau vor Beginn der Arbeit festgehalten:

- Eine **kleine Platinengröße**, mit maximalen Verhältnis: 50mm x 80mm
- Die Platine wird innerhalb **zweier Layer** designt (Top-Layer [*rot*], Bottom-Layer [*blau*]), wobei das farbliche Kontingent den Platinenabbildungen aus EAGLE zu entnehmen ist
- Es wird versucht, **platzsparend** zu designen
- Ein Abschrägen der Platine wird hinsichtlich des BLE-Moduls getätigt (Siehe „Empfohlene BLE-Modul-Ausrichtung", Seite 56)
- Zur späteren Befestigung innerhalb eines Gerätes werden vier Bohrungen und zwei Passamarken[5] benötigt
- Testpads werden gemeinsam gehalten
- Anschlüsse sind nur am äußeren Rand zu setzen
- Trimmkondensatoren für die zwei Quarze sind so nah wie möglich an deren Anschlüsse zu platzieren, um die hochfrequenten Störungen so gut wie möglich kompensieren zu können
- Leitungen sind nur so kurz wie möglich und in Winkeln von 45° zu legen, um u. a. Reflexion zu vermeiden und die Schwingungsneigung zu verringern

[5] Eine Passamarke ist eine auf der Platine ersichtliche Markierung für den bei der Firma Scholz System benutzen Bestückungsautomaten

3.4.2 Einstellungen für das Designen

Bevor es zum eigentlichen Designen kommt, können bereits allerhand Einstellungen über die z. B. Leitungsbreiten, Leitungsabstände, Objektabstände und Via[6]-Größen gemacht werden. Viele werden von dem späteren Preis, bei dem zu bestellenden Distributor, beeinflusst. Gerade kleinere Leitungsabstände oder Bohrdurchmesser erhöhen den Preis je Platine.

Folgende Tabelle zeigt alle eingehaltenen relevanten Einstellungen:

Einstellung	**Wert**
Platinendicke (Top + Bottom + Isolation)	1,57mm (0,035mm je Layer)
min. Abstand (Leitungen)	6mil
min. Abstand (Objekte)	0,5mm
min. Breite (Leitungen)	8mil
min. Bohrdurchmesser (Leitungen)	0,3mm
min. Via-Bohrdurchmesser	0,3mm

Tabelle A.4: Relevante EAGLE-Einstellungen, die für die zu entwerfende Platine verwendet werden

[6] Via's sind Platinenlöcher, welche reine Durchkontaktierungen zwischen den verschiedenen Layern darstellen

3.4.3 Empfohlene BLE-Modul-Ausrichtung

Da das Kernelement das Bluetooth-Low-Energy-Modul ist und damit die bestmögliche Performance erreicht werden will, wird auf die Empfehlungen des Datenblattes geachtet. Das dort benannte Optimum empfiehlt das Einbauen des Moduls so, dass die Bluetooth-Antenne (siehe Abb. 3.28, *weißes untermalenes Kreuz*) mit der Endstörfläche (obere 40% des Moduls) oberhalb der Hauptplatinenfläche schwebt und nicht in die Mitte eingesetzt wird.

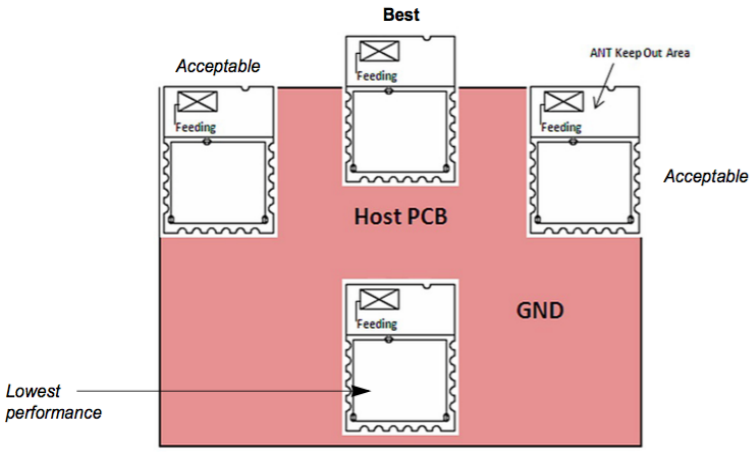


Abbildung 3.28: Die aus dem Datenblatt entnommene bestmöglich empfohlene BLE-Modul-Ausrichtung für die zu entwickelnde Low-Energy-Platine

3.4.4 Aufbau des ersten Borddesigns

Um einen größeren Einblick in die Entstehungsphase der fertigen Low-Energy-Platine zu erlangen und was erste Ansätze eines unerfahrenen Ingenieurs beim Designen sind, zeigt das unfertige (erste und ebenfalls unschöne) Borddesign (siehe Abb. 3.29 auf Seite 57). Unerfahrenerweise habe ich mich an den *gelben*, nicht statischen, Hilfslinien orientiert, da ich zu wissen glaubte, dass diese die generellen statischen Verbindungen zeigen. Außerdem besaß ich zu dem Zeitpunkt noch kein Verständnis dafür, wie viel Platz die eigentlichen Leitungen mit den Mindestabständen einnehmen würden und platzierte alle Komponenten zu nah beieinander als auch möglichst synchron bzw. parallel zueinander. Letztlich wurde ich eines Besseren belehrt und strukturierte alles dem Schaltplan nach um und bereitete mehr Platz vor, um darauf die Leiterbahnen verbinden zu können. Die Passamarken und Testpads habe ich ebenfalls anders platziert, um Engpässe der Leiterbahnen zu vermeiden.

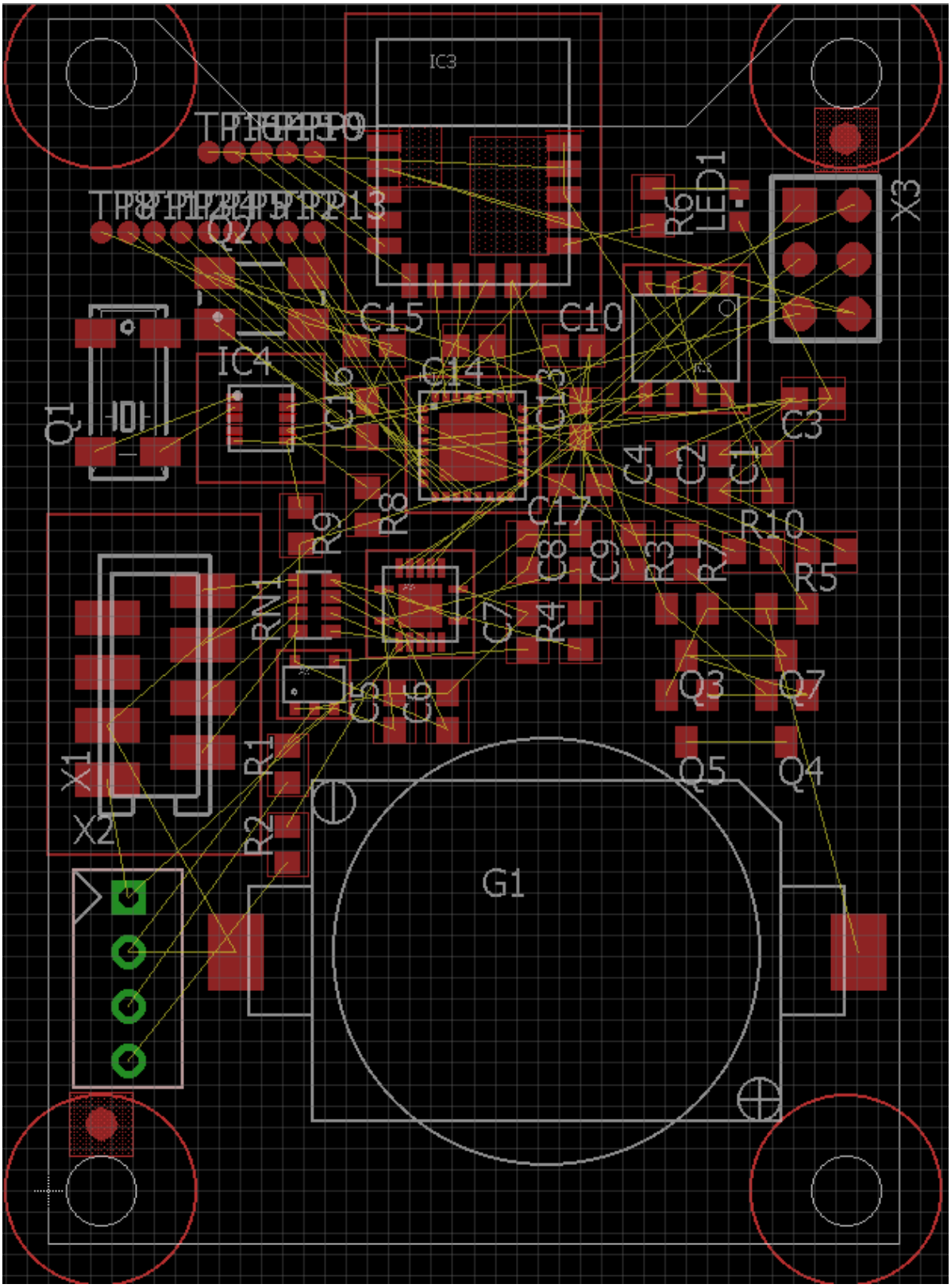
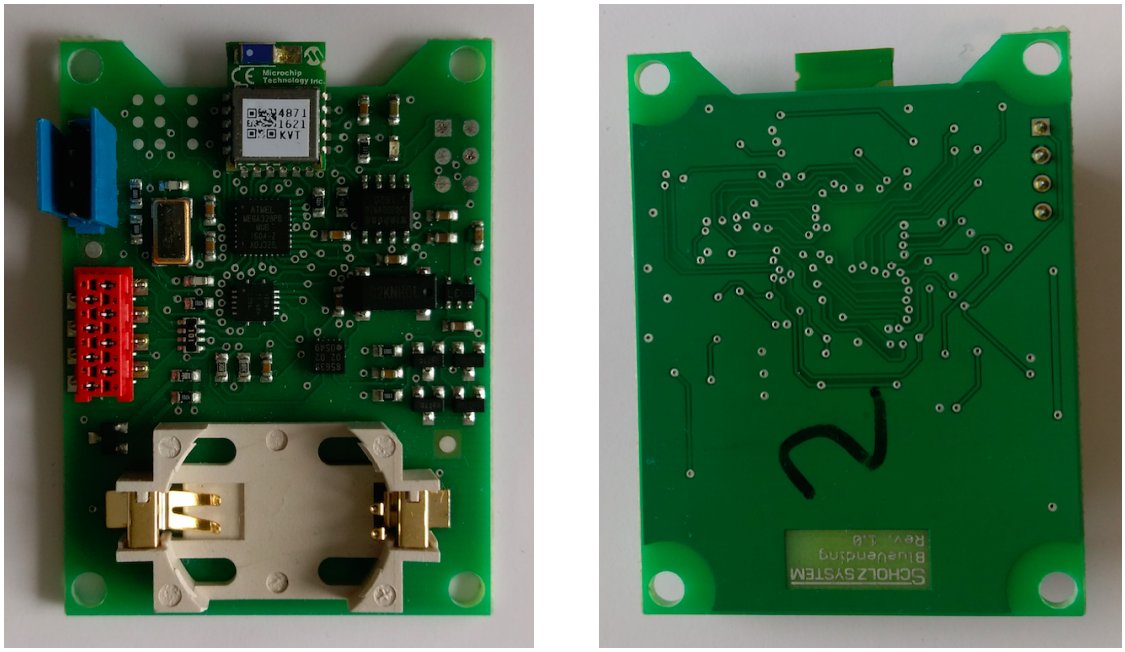


Abbildung 3.29: Das unkorrigierte und nicht angeschlossene erste Platinendesign im *Layout*-Editor von EAGLE (nur mit Komponentenindizes und *gelben* Anschlussmarkierungen)

3.4.5 Aufbau des fertigen Prototypen

Der erste Aspekt bezieht sich darauf, ob die in den Anforderungen stehende Platinengröße (max: 50mm x 80mm) eingehalten wird. Für das auf Abb. 3.31 (Seite 59) zu sehende Gesamtresultat ergibt sich nur eine Größe von 40mm x 53mm, was vergleichsweise klein ist. Gerade im Vergleich zu dem ersten Borddesign ist eine weitreichendere Verteilung aller Komponenten entstanden, um für die Leiterbahnen und Via's ausreichend Platz zu schaffen. Herausfordernd war, dass je mehr Leiterbahnen gelegt wurden, ein immer größeres Platzproblem entstand und dadurch öfter Komponenten umplatziert werden mussten, was bei bereits verbundenen, eine Kettenreaktion hervorrufen konnte. Der Kern, von dem alles ausgeht, wird vom ATmega328PB geleitet und erstreckt sich im oberen Bereich zum abstehenden RN4871-Modul. Rechts davon liegen der ISP, W25Q80DV-Flash und LP3985-Spannungswandler. Unten ist der Pegelwandler TXB0104, die RTC (PCF8563) sowie die CR2032 Batterie, dessen Halterung fast die Hälfte an Platinenplatz einnimmt. In dem letzten Teil befinden sich die zwei Anschlüsse (AMP/ Micromatch). Größtenteils sind diese Platzierungen so entstanden, dass alle benutzten Mikrocontroller-Pins, ihrer liegenden Seite basierend, den zugeordneten Hauptkomponenten leicht zugänglich gemacht werden.

Auf der Abb. 3.30 (a) (und *Bottomview* [b]) wird die fertig gelötete und für die Software vorbereitete Platine gezeigt.



(a) *Topview* (b) *Bottomview*

Abbildung 3.30: (a) Fertige Platine im Topview (b) Fertige Platine im Bottomview

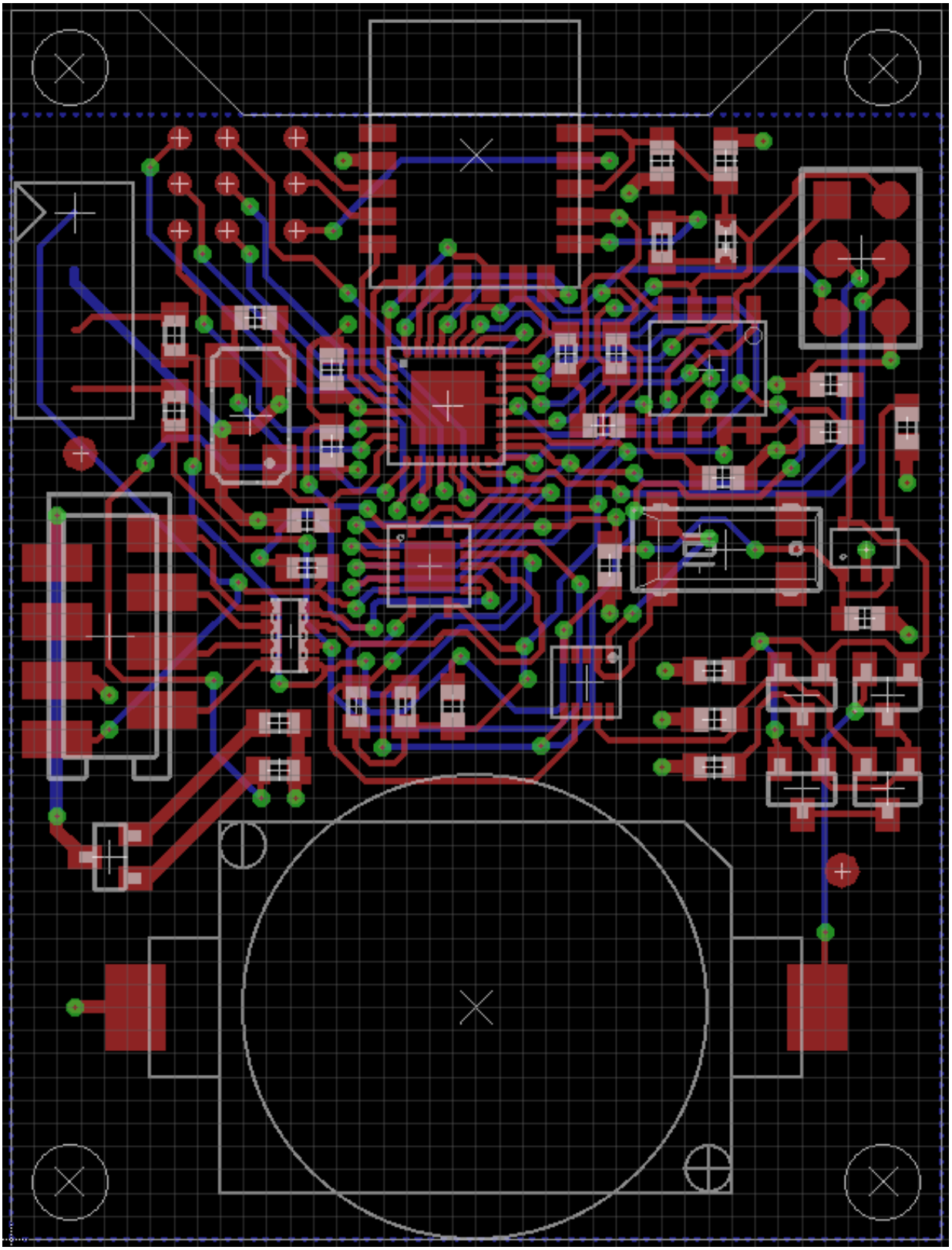


Abbildung 3.31: Das fertig korrigierte und angeschlossene Platinendesign im *Layout*-Editor von EAGLE (mit Top- und Bottom-layer und ohne Komponentenindizes)

4 Software-Analyse

4.1 Relevante Grundlagen

Im Folgenden werden die drei genutzten Kommunikationsinterfaces sowie ggf. relevante theoretische Spezifikationen erläutert, um ein ausreichendes Verständnis für die Funktionsweise des Kommunikationsverhaltens zu erlangen.

4.1.1 Serial Peripheral Interface (SPI)

Das Serial Peripheral Interface (SPI) ist ein synchrones[2] und seriell arbeitendes Kommunikationsinterface zwischen mindestens zwei on-board-Peripherie Geräten. Dabei läuft die Kommunikation im *full-duplex*[1]-Betrieb ab und arbeitet auf vier Leitungen. Die Serial Clock (SCLK) Leitung, oder auch SCK und CLK genannt, dient der Übertragung der Taktrate vom jeweiligen Master[3] ausgehend, an einen oder mehrere Slaves[3]. Spezieller: Sie entscheidet darüber, wann die zu sendenden Daten sich ändern und jeweils gelesen werden sollen. Master-Out-Slave-In (MOSI), auch SDI genannt, und Master-In-Slave-Out (MISO), auch SDO genannt, sind die zwei parallel arbeitenden Datenleitungen, die sich um die zusendenden Bytes kümmern. Die $\overline{\text{SS}}$-Leitung gibt den Betriebsstatus des Kommunikationsinterfaces mit einem jeweiligen Slave an. Bei mehr als einem angeschlossenen Slave wird der zu der Zeit aktive durch mehrere $\overline{\text{SS}}$-Leitungen separat geschaltet. Auf der Abb. 4.1, Seite 61, wird eine Verbindung zu genau einem Slave veranschaulicht [4].

[1] *full-duplex*-Betrieb heißt, dass die Kommunikation zu beiden Seiten hin gleichzeitig abläuft

[2] *synchron* bedeutet, dass die Kommunikation durch eine laufende Taktrate begleitet wird

[3] Der *Master* ist in einer kommunizierenden Verbindung (unidirektional) zweier Geräte, der Befehls- als auch Taktgeber dieser. Der *Slave* ist dabei das horchende bzw. anfragende Gerät und steht hierarchisch unter dem *Master*

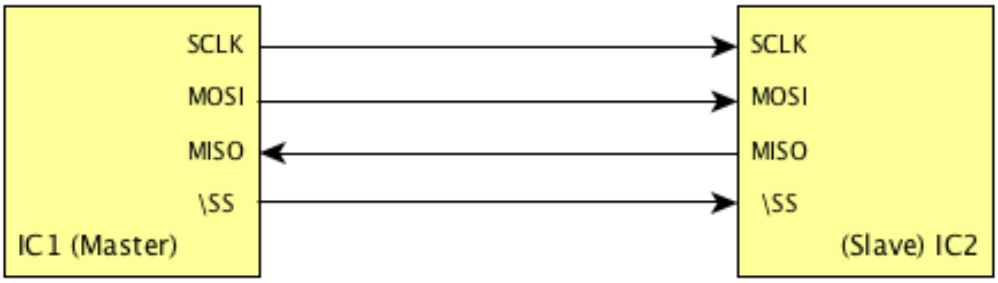


Abbildung 4.1: Zwei IC's im master/slave-Betrieb (\SS entspricht $\overline{SS}$)

4.1.2 Universal Asynchronous Receiver Transmitter (UART)

Der Universal Asynchronous Receiver Transmitter (UART) ist das am universellsten benutzte Kommunikationsinterface der in dieser Arbeit beschriebenen und arbeitet seriell, *asynchron*[4] sowie spezifisch konfigurierbar. Einstellungsmöglichkeiten, bei dem Datenformat als auch der Übertragungsgeschwindigkeit, müssen gemacht werden, um damit gemeinsam die für die gegenüberliegende Seite benötigte Synchronität herzustellen. Beide Seiten benötigen somit die gleichen Parameter, um ggf. vorliegende Daten korrekt auslesen zu können. Der erste benötigte Parameter heißt Baudrate und beschreibt die Anzahl der übertragenen Symbole pro Zeicheneinheit. Zweiter Parameter ist das Datenformat, welches über die Anzahl der Datenbits, der Parität sowie Stopbits eingestellt wird [15].

Eine RS-232-Schnittstelle eines Computers kann in Verbindung eines UART's, an z. B. einer Platine, so eingestellt werden, dass Synchronität entsteht und dadurch ein beiderseitiges Überprüfen der Funktionalität stattfinden kann.

[4] *asynchron* bedeutet, dass kein eigenes Taktsignal über die Datenleitungen übertragen wird

4.1.3 Inter-Integrated Circuit (I^2C)

I^2C ist ein weiteres Kommunikationsinterface und kann mit mindestens einem Master einen oder mehrere verbundene Slaves ansprechen. Der Unterschied zu UART und SPI besteht darin, dass die Datenkommunikation nur über eine unidirektionale Leitung, meist SDA genannt, abgewinkelt wird und dazu eine zweite Leitung, meist SCL genannt, für die Taktraten-Übertragung benutzt wird. Typischerweise nutzt man dieses Protokoll für langsame Datenkommunikationen zweier naher Peripherien in den Geschwindigkeiten 100kb/s (*Standard mode*) und 10kb/s (*low-speed mode*). Die Möglichkeit zu etwas höheren Geschwindigkeiten besteht dennoch: 400kb/s im *Fast mode*, 1Mb/s im *Fast mode plus*, und 3,4 Mb/s im *High Speed mode*. Das Protokolldesign wurde so konzipiert, dass die Ansprache eines verbundenen Slaves durch eine 7-Bit oder 10-Bit Adressierung läuft und durch die langsame Datenkommunikationsgeschwindigkeit einen sehr geringen Stromverbrauch aufweist sowie nur geringfügig Mikrocontroller-Pins nutzt.

Die Funktionsweise beschreibt sich so, dass jedes angeschlossene Gerät entweder ein Sender oder Empfänger, sowie ein Master oder Slave sein kann. Master-Geräte leiten den Datentransfer ein und geben die Taktrate vor, woraufhin alle anderen am Bus[5] hängenden Geräte zu Slaves werden. Zusammengefasst gibt es vier verschiedene potentielle Operationsmodi: Je ein Sende- sowie Empfangs-modus für den Master und den Slave [10].

[5] Ein Bus beschreibt in der Datenkommunikation eine gemeinsam genutzte Leiterbahn, an der mehrere Teilnehmer hängen

5 Software

Den Hauptteil dieses Themas beschreibt, ohne expliziter Codebeispiele oder genauer Registerbezeichnungen, die theoretische Herangehensweise und praktische Bearbeitung einer machbaren Kommunikation der Low-Energy-Platine, als Kern, zu einem Zigarettenautomaten und anderweitig einer Android Applikation. Dazu wird eine Software im Microsoft Visual Studio Community 2015 geschrieben, welche daraufhin durch den ISP und ein weiteres Hilfsprogramm, Atmel Studio 7, in den Flash des ATmega328PB geladen wird.

Hinweis: Die von mir geschriebene Android-Applikation, sowie die von der Firma Scholz System gegebene innere Automatensoftware werden wegen einer vermeintlichen Sprengung des Rahmens, für die hier erläuterte Software angepasst und funktionierend betrachtet.

5.1 Initialisierungen

Der erste Schritt beschäftigt sich damit, alle Ports des ATmega328PB zu initialisieren, dazu speziell die Datenrichtung und den *default*-Wert bzw. ob der interne *pull-up*-Widerstand des behandelten Pins aktiviert oder deaktiviert sein soll. Weiterhin wird ein Software-Fifo, der Ringpuffer, einprogrammiert, um mittels fester Puffergröße (Datenspeicher) und der ISR die empfangenen gespeicherten Bytes der UART's bei Bedarf wieder auszusenden. Wird die Puffergröße beim Speichern, ohne vorzeitigem Auslesen, überschritten, dann beginnt das Überschreiben der ersten Ringposition. Um ein zu schnelles, großes Datensenden (asynchron) und somit Datenverlust kontrollieren zu können, wurden die RTS- und CTS-Leitungen (*hardware-flow-control*) des RN4871 mit dem ATmega328PB verbunden (siehe Abb. 3.16 [Seite 40] und Abb. 3.17 [Seite 42]). Damit die CPU des ATmega328PB, durch z. B. ein einfaches UART-*polling*, nicht unnötig ausgelastet wird, ist ein Timer zu initialisieren und die Benutzung der Interrups des Mikrocontrollers von Vorteil. Letztlich kommt noch ein Watchdogtimer dazu, welcher eine Endlosschleife durch einen Mikrocontrollerreset beseitigt.

5.1.1 Timer

Für den Timer wird der Clear Timer on Compare (CTC) Modus verwendet, um bei der konstanten Frequenz von 7,3728MHz eine eigene Zählgrenze verwenden zu können, die mit der errechneten Baudrate von 115200bps (Gleichung 5.1), etwa jede Millisekunde erreicht wird. Dabei wird der kleinstmögliche Prescaler von 64 gewählt, da der Timer damit weniger Schritte braucht und der Rundungsfehler (0,0%) so klein wie möglich gehalten wird. Nach Erreichen der Zählgrenze (115 Schritte, siehe Gleichung 5.2 bei 1000 Interrups/Sekunde) wird der Interrupt getriggert, der Zähler resettet und daraufhin der Timer inkrementiert (bei einer Periodendauer von 8,6806μs und 115 Zählschritten, alle 998,3μs, Gleichung 5.3). Den Prescaler von 8 könnte man nicht nehmen, da dies nur ein 8-Bit Timer ist und ein vielfaches mehr an Schritten benötigt werden würde (922 Schritte).

$$baud = \frac{F_{CPU}}{N} = \frac{7,3728MHz}{64} = \underline{\underline{115200Hz}} \tag{5.1}$$

$$OCRx = \frac{F_{CPU}}{\frac{N}{1000}} = \frac{7,3728MHz}{\frac{64}{1000}} = \underline{\underline{115}} \tag{5.2}$$

$$T = \frac{1}{f} = \frac{1}{115200bps} = 8,6806\mu s \cdot 115 = \underline{\underline{998,3\mu s}} \tag{5.3}$$

$baud$ ist die verwendete Baudrate hinsichtlich der Quarzfrequenz (F_{CPU}), N der Prescalerwert, $OCRx$ der maximal zu erreichende Zählerwert (Interrupt flag) und T die Periodendauer

5.1.2 Ringpuffer und UART

Für die beidseitig, des gleichen Kommunikationstyps
(Rn4871 <– [UART0] –> ATmega328PB <– [UART1] –> AMP-Stecker [Zigarettenautomat]), benötigte Einrichtung der UART's werden für die Richtungen (Tx und Rx) je ein Ringpuffer initialisiert. Diese strecken sich über mehrere Funktionen, wie z. B. , ob noch auszulesende Daten verfügbar sind (rp_dataAvailble), ob im Puffer noch Platz ist (rp_spaceAvailble) oder Funktionen dieser, die Abfragen bezüglich des Ringpuffers selbst ermöglichen (rp_empty, rp_full). In der Initialisierungsfunktion der UART's wird die benutzte Baudrate (115200bps mit 0,0% Fehler bei 7,3782MHz) für das 12-Bit große Register, bestehend aus einem 8-Bit (LSB's) und einem 4-Bit (MSB's), umgewandelt. Die auf der nächsten Seite stehende Gleichung 5.4 zeigt die Rechnung für den normalen Geschwindigkeitsmodus.

$$baud_{normalspeed} = \frac{F_{CPU}}{8 \cdot baud} - 1 = \frac{7,3728MHz}{8 \cdot 115200bps} - 1 = \underline{\underline{7}} \tag{5.4}$$

$baud_{normalspeed}$ ist der errechnete und benötigte Wert für die UART-Baudratenregister, $baud$ die benutzte Baudrate und F_{CPU} die verwendete Quarzfrequenz

Da für das Command-Interface des RN4871 ASCII-Zeichen benötigt werden, wird für den Übertragungsmodus acht Bit ein Stopp-Bit und keine Parität (8N1) eingestellt, weil Zeichen zwei Byte groß sind und Übertragungsfehler anderweitig behandelt werden (siehe Kapitel „Datenkommunikationsprotokoll", auf Seite 66). Um den durch zu große Daten entstehenden Datenverlust, beim UART0, kontrollieren zu können, wird die *hardware-flow-control* mit einprogrammiert. Dabei prüft die ISR u. a. vor jedem Tx-Vorgang, ob $CTS_{\mu c}$ == *high*, also das vorherige Senden noch nicht abgeschlossen ist. Nur wenn $CTS_{\mu c}$ == *low* ist, darf der Rx-Vorgang stattfinden. Die $CTS_{\mu c}$ Zustände werden dabei vom angeschlossenen RTS_{rn4871} übermittelt. $RTS_{\mu c}$ ist bei noch genügend (> 5) Platz im Ringpuffer immer *low* und übermittelt dem CTS_{rn4871} nur bei zu großem Datenspeichern, mit einem *high*-Pegel, dass vorerst wieder ausgelesen werden muss.

5.1.3 BLE-Modul und weitere Initialisierungen

Weiterhin wurde das RN4871 für die ersten Erprobungen, mit *hardware-flow-control*, einer Abschaltung der Command-Interface-Echo-Funktion und dem Modulreboot, nachdem ein User sich *disconnected*, initialisiert. Die Echo-Funktion würde das benutzte „Datenkommunikationsprotokoll" verlängern, da ständig danach abgefragt werden müsste. Das Modulreboot hat sich bei mehreren hintereinander folgenden Verbindungstests (dem *connecting* durch die Android-Applikation), bei Wiedereinwahlen, als geschwindigkeitserbringend herausgestellt. Alle Änderungen sind im *non-volatile* Memory des Moduls gespeichert.

Da es sich nur um eine erste Machbarkeit eines Prototypen handelt, wurden nur noch die SPI-Schnittstellen initialisiert und ein Ansprechen des W25Q80DV (Flashs), mit Erfolg, getestet. Warum es aber noch keine Ansprache der RTC und Batterie, per I^2C, als auch richtiger Nutzung des Speichers, gibt, wird in Kapitel 7 erörtert (Seite 71).

5.2 Datenkommunikationsprotokoll

Um geregelte und für alle drei Seiten „verständliche”, Datenkommunikation bereitzustellen, wurde das auf der Abb. 5.1 (Seite 67) beschriebene „Datenkommunikationsprotokoll” erstellt, welches eine Abwandlung typisch benutzter ist. Denn das einfache Senden von den Daten an sich wird keine verständliche Kommunikation erbringen, wenn man sich vorher keine Gedanken darüber gemacht hat, wie Kommunikationsfehler abgefangen bzw. berichtigt werden und was dazu benötigt wird.

Das Protokoll besteht beim Senden von Daten aus den Databyte ([data]) und der restlichen Nachricht (6 Byte), in der das erste Byte das zu sendende Kommando ([*command*], siehe auch Abb. 6.1, auf Seite 69, z. B. SEND_CIG_TABLE), repräsentiert. Auf der Gegenseite wird dieses Kommando für die Zustandsmaschine verwendet. Das zweite Byte ([*order*]) zählt die in Folge gesendete Anzahl der gleichen Nachricht mit, um ein Mehrfachversenden unterscheiden zu können. Das dritte und vierte Byte ([*length*]) liefert die Datenlänge und die letzten zwei Byte ([*checksum*]) die Checksumme, welche wie folgt erläutert wird.

[1 byte : command] [1 byte : order] [2 byte : length] [X byte : data] [2 byte : checksum]

Die Prüfung der Richtigkeit einer Nachricht wird vorerst mithilfe einer einfachen Checksumme getätigt, welche alle, bis auf die eigenen Werte, addiert und mitsendet. Die andere Seite erhält die Nachricht und rechnet sich aus den erhaltenen Daten ebenfalls eine Checksumme zusammen, die bei Fehlerlosigkeit somit genau den gleichen Wert der erhaltenen Checksumme haben muss.

[0] Die

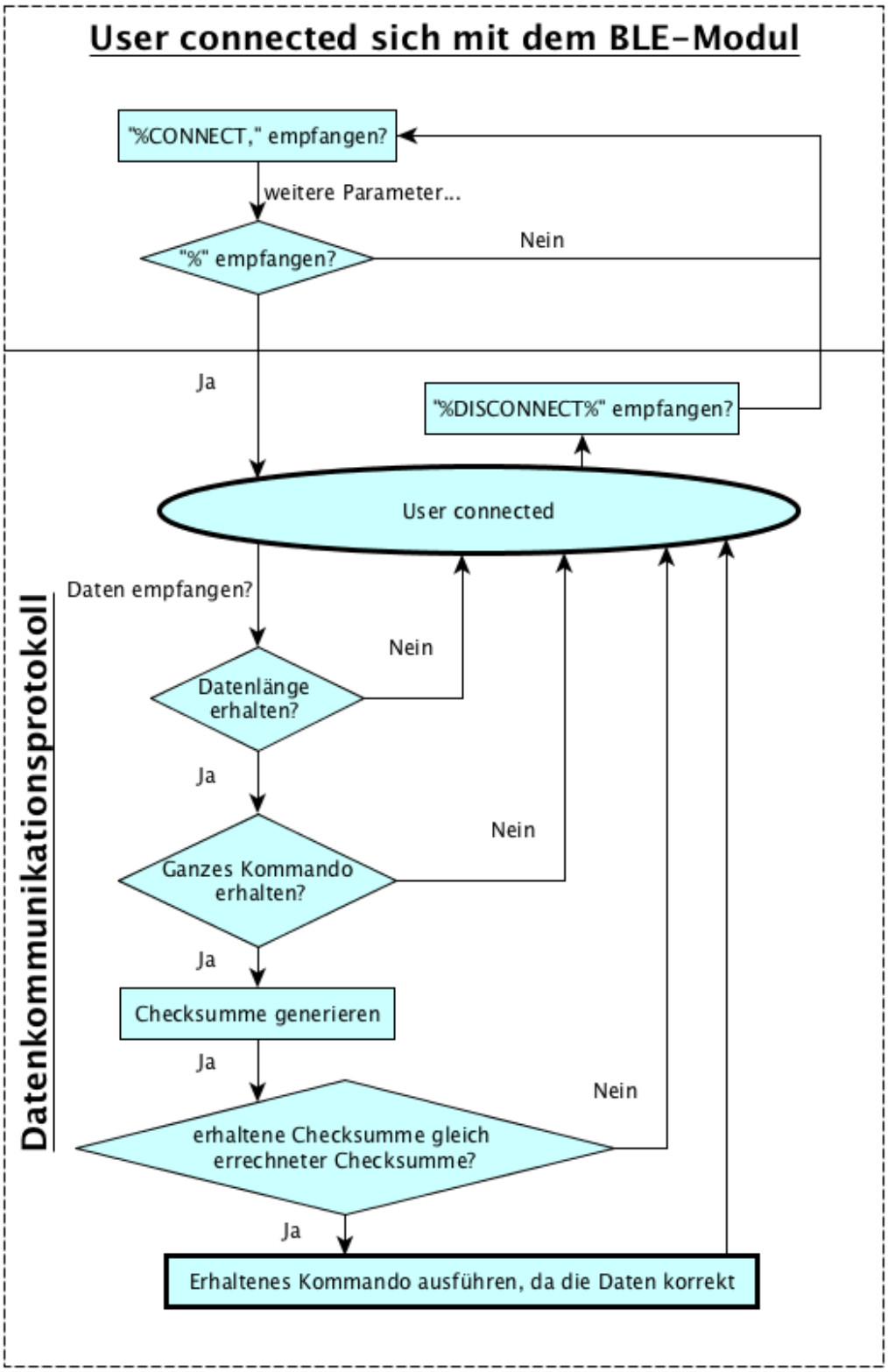


Abbildung 5.1: Ein Flussdiagramm für den Erhalt der Daten einer Android-Applikation über den UART0, die wie beschrieben verarbeitet werden. *(BLE-Antworten auf Zustandsänderungen werden von zwei Prozent-Zeichen eingeschlossen)*

6 Endanalyse

6.1 Praktische Erprobung mit einem Zigarettenautomaten und einer Android-Applikation

Der letzte Teil dieser Arbeit ist die thematische Auseinandersetzung mit der praktischen Machbarkeit eines Kaufabschlusses an einem herkömmlichen Zigarettenautomaten, durch eine funktionierende Android-Applikation und die entwickelte Low-Energy-Platine.

6.1.1 Prozedere

Das Miteinander der o. g. Systeme wird sich durch die geschriebene Software vorerst soweit erstrecken, dass ein *user* der Applikation, durch eine erstellte Bluetoothverbindung mit einem Zigarettenautomaten (Low-Energy-Platine), eine Zigarettenschachtel kaufen kann und die gewählte daraufhin erhält. Genauer werden dem *user* nach erfolgreicher Verbindung und „einfacher" Bestätigung des Alters die verfügbaren Zigarettenmarken übertragen, woraufhin der Erhalt (testweise, ohne Budget) möglich sein wird. Das auf der nächsten Seite zu sehende Flussdiagramm (Abb. 6.1) beschreibt die Verfahrensweise Punkt für Punkt, um eine Zigarettenschachtel durch die Applikation erhalten zu können. Dabei werden Funktionskommandos angegeben, dessen Bedeutungen sich wie folgt beschreiben lassen:

- SEND_CIG_EIGTHEEN_VERIFICATION: Der Benutzer hat sein Alter durch eine einfache Bestätigung von „Ja" getätigt, woraufhin das Handy diese an den Automaten sendet
- REQ_CIG_TABLE: Die Applikation sendet daraufhin automatisch eine Anfrage über die verfügbaren Zigarettenmarken an den Automaten
- SEND_CIG_TABLE: Der Automat sendet alle erhältlichen Marken mit der Marken-ID, dem Preis und der verfügbaren Anzahl an die verbundene Applikation
- SEND_CIG_COMPLETE: Die Applikation sendet die vom User verlangte Schachtel (ID) nach Kaufbestätigung an den Automaten

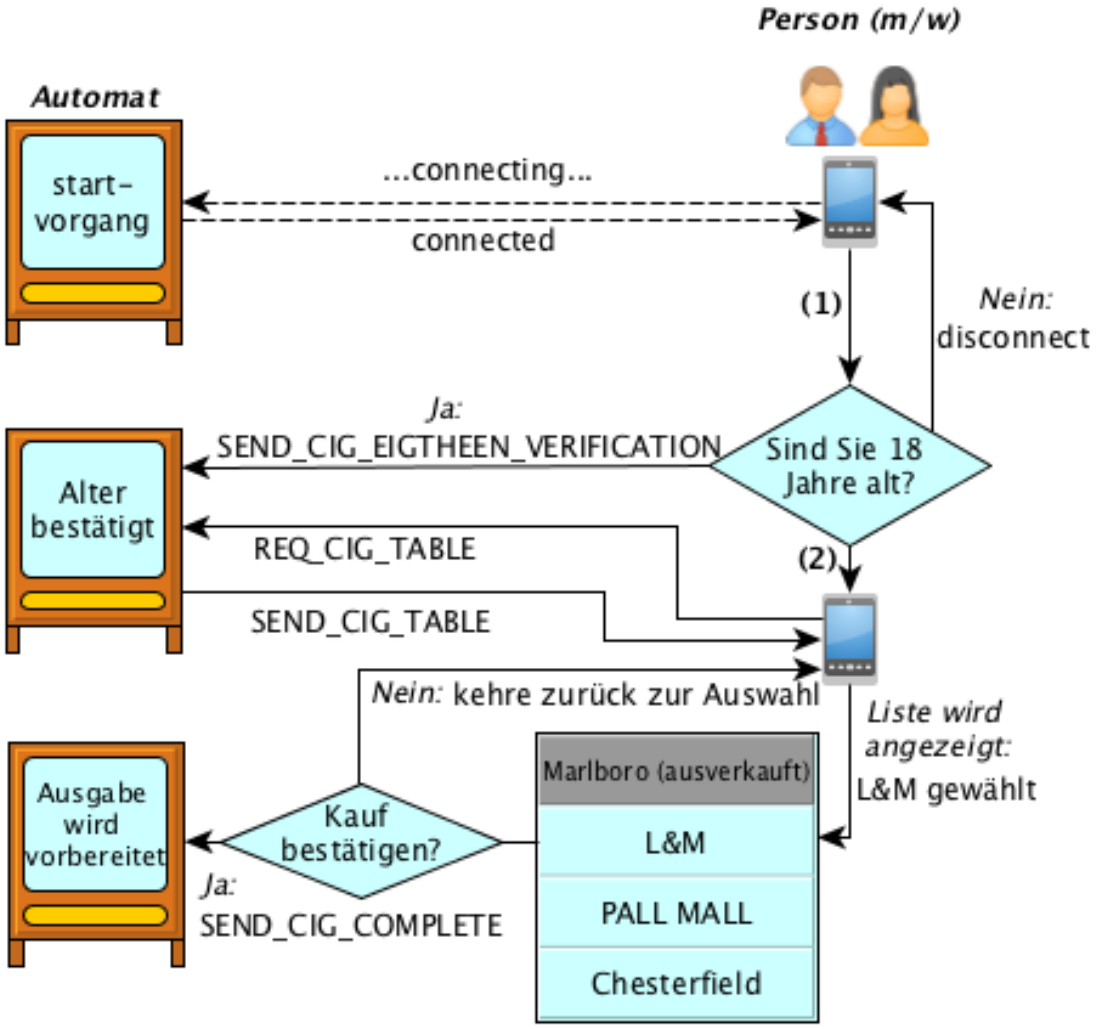


Abbildung 6.1: Bildlich erklärtes Prozedere der Machbarkeit eines Zigarettenschachtel-Kaufes, durch eine Android-Applikation in Verbindung der Low-Energy-Platine und dessen Master, dem Zigarettenautomaten. (1) steht für die folgende Anzeige der Altersverifikation, (2) für die nach Erhalt gezeigte Zigarettenmarkenliste und die großen frei stehenden Zweigbeschreibungen (SEND_X, REQ_Y) für die Kommandos (erstes Byte [*command*], des Datenkommunikationsprotokolls)

6.2 Prototyp (Fehlerdiagnose)

Wie unten zu sehen ist, teilen sich der ISP und die AMP- und Micromatch-Stecker die gleichen I/O-Pins des ATmega328PB. Ersterer Stecker kann über UART und die anderen beiden über SPI kommunizieren. Das heißt konkret, dass drei verschiedene Bauteile, mit zwei unterschiedlichen Kommunikationsarten und nur einer gemeinsamen Beschaltung, am Mikrocontroller auskommen müssen. Bei dem ersten Durchstöbern des Datenblattes arbeitete ich, nach dem Finden des USARTSPI-Anwendungsfalles, weiter und bestellte letztlich die Platine. Erst daraufhin fiel auf, dass dieser Anwendungsfall nur im Master-SPI-Modus verfügbar wäre und nicht im benötigten, herkömmlichen, Slave-SPI, woraufhin der Master nur als SPI-Kommunikationsinterface verfügbar wäre. Dies ist für unseren Fall nicht möglich, da im außenstehenden Gerät die darauf angeschlossene Platine das SPI-Interface bereits als Master nutzt. Das verbundene Schema ist in dem vorhandenen Layout (siehe Abb. 6.2) zu sehen, das für eine relative Betrachtung angefertigt wurde und nicht den genauen Aufbau mit allen Sekundärkomponenten angibt. Dabei wird, wegen völliger Selbstständigkeit (fehlender Slave Select-Leitung) und deshalb fehlendem Belangen, die ISP-Schnittstelle weggelassen.

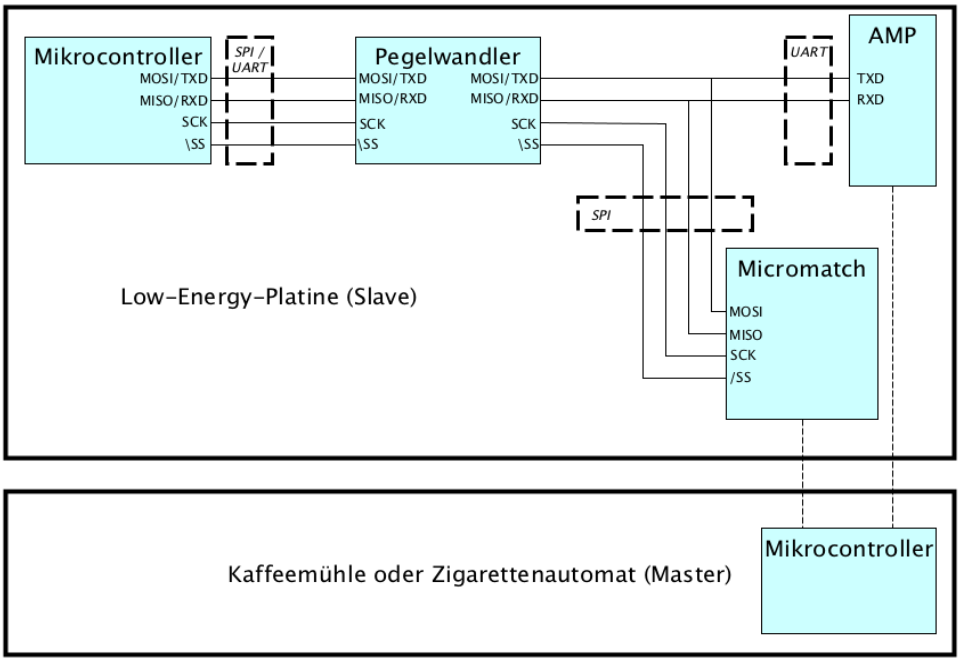


Abbildung 6.2: Fehlerdiagnose: Mehrfachübergreifende Kommunikationsinterfaces auf einer gemeinsamen Beschaltung am Mikrocontroller (Low-Energy-Platine) sowie die relative zukünftige Beschaltung an eine Kaffeemühle oder einen Zigarettenautomaten

7 Ausblick auf die Zukunft und mögliche Problembehebungen

Die Bluetooth-Low-Energy-Platine bietet, bis auf die Hauptkomponenten selbst, noch kein wirkliches Low-Energy-Verhalten, trotz vieler großmütiger Anspielungen. Dies liegt vor allem daran, dass es sich für eine erste Machbarkeit noch nicht auszahlt, bereits Standby-Eigenschaften jedes Bauteiles zu aktivieren, wenn noch nicht klar ist, was letztlich in das Endprodukt fließen soll. Deshalb fiel ebenfalls die dazugehörige Werksbetrieb-Stromanalyse weg. Der produzierte Prototyp muss noch eine Reihe an Änderungen durchlaufen, um es aus der Testphase in die Verkaufsproduktion zu schaffen. Vor allem das Problem des falschen USARTSPI-Modus muss entweder durch den Tausch des Mikrocontrollers (*worst-case)* oder dem Umstecken jeder Hauptkomponente beseitigt werden. Die einzige Möglichkeit hierzu wäre, den AMP-Stecker an UART0 des vorher benutzten BLE-Moduls zu schließen und dann mit dem vorhandenen SPI-Master-Modus das Modul und den Flash zu verbinden (UART1/SPI0). Übrig bleibt der SPI1 für den Micromatch-Stecker. Da die Testphasen einer einfachen Machbarkeit an einem Zigarettenautomaten abgeschlossen sind, besteht die zukünftige Aufgabe darin, das Gleiche mit einer Kaffeemühle zu verwirklichen. Erst daraufhin wird sich näher mit dem eingebauten Flash, der RTC und der Batterie befasst, sowie den vielfältig abrundenden Einstellungsmöglichkeiten jeder Hauptkomponente (Standby-Modus, Passwort zum Verbinden per BLE, verschlüsselte Anzeige für suchende Geräte, Anzahl gleichzeitig verbundener Teilnehmer, Bestell-Warteschlangen etc.).

8 Fazit

Das Ergebnis dieser Arbeit ist eine solide Hard- und Softwarelösung einer optional, in Zigarettenautomaten oder Kaffeemühlen, einsetzbaren Bluetooth-Low-Energy-Platine geworden. Durch den Einsatz ergeben sich Möglichkeiten der mobileren Verwendbarkeit der Platine mit Smartphones oder Smartwatches, die über die aktuellste Bluetooth-Version 4.2 (Stand: Januar 2017) verfügen, bis hin zum zukünftigen Low-Energy-Verhalten, mittels Standby-Struktur, oder z. B. der Anzeige einer Uhrzeit bei einer Kaffeemühle.

Trotzdessen, dass noch viel Arbeit anstehen wird, ist das Grundgerüst in Hard- und vor allem Software da, um in der nächsten Zeit ein funktionsfähiges Produkt zu erschaffen. Auch wenn in der Arbeit hauptsächlich die Hardware erläutert wird, floss ein ebenso großer Teil an Zeit in die Programmierung des Mikrocontrollers und BLE-Moduls bzw. derer Kommunikation miteinander. Mein im siebten Semester abgeleistetes Praxissemester bereitete die Android-Applikation vor, die meiner Meinung nach, ohne Vorkenntnisse nicht stressfrei parallel zur Platine machbar wäre. Ebenso wurde mir die Ansprache des Zigarettenautomaten in Richtung Low-Energy-Platine abgenommen, da es andernfalls eine Hürde wäre, sich in dem fremden Programmcode zurechtzufinden. Im Großen und Ganzen besteht das finale Konzept dieser Arbeit also aus mehreren, miteinander fungierenden, Stützen und wäre vielleicht in dem Ausmaß eines Buches unterzubringen.

Motivierend war nach den ersten Wochen das fertig entwickelte Platinenlayout, die kurze Zeit später in der Hand haltende Platine und gegen Ende hin, als Motivationshöhepunkt, die „Bestellung" einer Zigarettenschachtel mit der Android-Applikation, woraufhin der Automat diese mit einer Ausgabe lautstark erwiderte. Dies alles mit so großer Euphorie, obwohl ich gar kein Raucher bin.

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

BLE Bluetooth-Low-Energy

UART Universal Asynchronous Receiver Transmitter

SPI Serial Peripheral Interface

I^{2}C Inter-Integrated Circuit

RTC Real-Time-Clock

Fet Feldeffekttransistor

ppm parts per million

BR Basic Rate

SMD Surface-mounted device

DRAM Dynamic Random-Access Memory

SRAM Static Random-Access Memory

IC Integrated circuit

MISFET Metall-Isolator-Feldeffekttransistors

GAP Generic Access Profile

GATT Generic Attribute Profile

SCLK Serial Clock

MISO Master-In-Slave-Out

MOSI Master-Out-Slave-In

SS Slave Select

GUI Graphical User Interface

UUID Universally Unique Identifier

SOIC Small-Outline Package

VSOP Very Small-Outline Package

LDO Low-Dropout Regulator

ISP In-System-Programmierung

GPIO General-purpose Input/Output

ISR Interrupt Service Routine

CTC Clear Timer on Compare

Literaturverzeichnis

[1] ABRACON, Corporation: *CERAMIC SURFACE MOUNT LOW PROFILE QUARTZ CRYSTALS*. 01.18.2013. – http://www.mouser.com/ds/2/3/ABMM2-40161.pdf [Abgerufen: 10. Januar 2017]

[2] ATMEL: *ATmega328PB*. – http://www.atmel.com/Images/Atmel-42397-8-bit-AVR-Microcontroller-ATmega328PB_Datasheet.pdf [Abgerufen: 01. Januar 2017]

[3] BERNSTEIN, Herbert: *Grundlagen der Hard- und Software der Mikrocontroller ATtiny2313, ATtiny26 und ATmega32*. Springer Vieweg, 2015. – ISBN 978-3-658-02812-1

[4] BYTEPARADIGM: *[Online] Introduction to I2C and SPI protocols*. – http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/ [Abgerufen: 12. Dezember 2016]

[5] GÖBEL, Holger: *Einführung in die Halbleiter-Schaltungstechnik - 5 Auflage.* Springer Vieweg, 2014. – ISBN 978-3-642-53869-8

[6] HEYDON, Robin: *Bluetooth-Low-Energy The Developer's Handbook*. RR Donnelley, 2012. – ISBN 978-0-13-288836-3

[7] INFINEON: *IRLML6402*. 14.12.2011. – http://www.infineon.com/dgdl/irlml6402.pdf?fileId=5546d462533600a401535668c9822638 [Abgerufen: 15. Januar 2017]

[8] INFINEON: *IRLML0040TRPbF*. 29.02.2012. – http://www.infineon.com/dgdl/irlml0040pbf.pdf?fileId=5546d462533600a401535664894825e4 [Abgerufen: 11. Januar 2017]

[9] MICROCHIP: *Bluetooth ® 4.2 Low Energy Module*. 2016. – http://ww1.microchip.com/downloads/en/DeviceDoc/50002489A.pdf [Abgerufen: 02. November 2016]

[10] NXP: *I2C-bus specification and user manual*. 04. April 2014. – http://www.nxp.com/documents/user_manual/UM10204.pdf [Abgerufen: 12. Dezember 2016]

[11] NXP: *PCF8563 Real-time clock/calendar*. 2015. – http://www.nxp.com/documents/data_sheet/PCF8563.pdf [Abgerufen: 04. Januar 2017]

[12] OSRAM: *CHIPLED 0603*. 2014. – http://www.osram-os.com/Graphics/XPic9/00128792_0.pdf [Abgerufen: 10. Januar 2017]

[13] SAUTER, Martin: *Grundkurs Mobile Kommunikationssysteme - 6 Auflage.* Springer Vieweg, 2015. – ISBN 978-3-658-08341-0

[14] STINY, Leonhard: *Aktive elektronische Bauelemente - 3 Auflage.* Springer Fachmedien Wiesbaden, 2016. – ISBN 978-3-658-14386-2

[15] TESSEL: *[Online] A Web Developer's Guide to Communication Protocols (SPI, I2C, UART, GPIO)*. 22. Januar 2015. – https://tessel.io/blog/108840925797/a-web-developers-guide-to-communication-protocols4#gpio [Abgerufen: 13. Dezember 2016]

[16] TEXAS, Instruments: *TXB0104 4-Bit Bidirectional Voltage-level Translator With Automatic Direction Sensing and ±15-kV ESD Protection*. 2014. – http://www.ti.com/lit/ds/symlink/txb0104.pdf [Abgerufen: 03. Januar 2017]

[17] TEXAS, Instruments: *LP3985 Micropower,150-mA Low-Noise Ultra-Low-Dropout CMOS Voltage Regulator*. 2015. – http://www.ti.com/lit/ds/symlink/lp3985.pdf [Abgerufen: 03. Januar 2017]

[18] ULRICH TIETZE, Christoph S.: *Halbleiter Schaltungstechnik - 11 Auflage.* Springer, 1999. – ISBN 3-540-64192-0

[19] WINBOND: *3V 8M-Bit SERIAL FLASH MEMORY WITH DUAL AND QUAD SPI*. 11. Februar 2015. – https://www.winbond.com/resource-files/w25q80dv_revf_02112015.pdf [Abgerufen: 23. Dezember 2016]

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 6. Februar 2017	
Ort, Datum	Unterschrift