Titel

Team 1: Eggs & Beacon

*Tobias Gilgenreiner, Felix Dollinger, Marco Stoiber,*

*Maximilian Kritzenthaler, Eduard Schröder*

Untertitel

# Robonova

In diesem Kapitel geht es um den Robonova Roboter von HiTec. Es werden alle Hardwarebestandteile des Robonova sowie benötigte Hardwaretools vorgestellt. Anschließend wird erklärt, wie man eine Entwicklungsumgebung für den Robonova auf dem Computer aufsetzt und diese nutzt. Danach wird die selbst entwickelte Treibersoftware vorgestellt. Zuletzt wird beschrieben wie man die Firmware auf den Servomotoren neu programmieren kann.

## Hardwareübersicht

In der Sammelkiste des Robonova ist eine große Sammlung an Bauteilen und ähnlichem von vergangenen Projekten. Im Folgenden wird, die für dieses Projekt benötige, Hardware vorgestellt.

### Robonova Backplane

### 

**Legende:**

Grün – Debug Header

Lila – UART Header BLE- /WIFI-Modul

Blau – GND

Rot – VCC

Orange - Signal

Die orginale Backplane des Robonova wurde durch eine Custom-Backplane ersetzt. Sie wurde von Florian Laufenböck entwickelt.

Auf dem Board ist ein 32bit ARM-Cortex Microcontroller STM32F405RG verbaut. An diesem ist ein MPU-6050 über I2C und ein RN4020 BLE-Modul über UART angebunden. Die UART Leitungen des BLE-Moduls führen zu dem lila Header. Somit kann entweder das BLE-Modul (Achtung: gekreuzte RX und TX Leitungen) oder ein anders Modul an der UART Instanz genutzt werden. Für die Servomotoren werden 4 UART Instanzen des Microcontroller genutzt. Jede Instanz führt zu einem Header an welchem je 5 Servomotoren angeschlossen werden können.

Aktuell sind 2 Backplanes bestückt. Davon funktioniert bei einer Platine nur eine „Servomotor UART Header“ und bei der anderen gar keine.

### Robonova Servomotoren

Ein Bild, das sitzend, Tisch, Kabel, Paar enthält.

Automatisch generierte Beschreibung

Für die Gelenke werden HiTec HSR-8498HB Servomotoren genutzt. Diese bieten ca. 190° Drehwinkel. Die orginalen Steuerchips wurden durch Atmega 328P ersetzt. Für den detaillierten Aufbau im Servomotor siehe Dokumentation Servomotoren. Die Chips können mit einem bestimmten Aufbau über UART programmiert werden, somit müssen sie nicht abgelötet werden. Mehr dazu im Kapitel.

### WIFI Modul

Ein Bild, das Elektronik, Schaltkreis, Fern enthält.

Automatisch generierte Beschreibung

Zur Kommunikation mit dem Backend wurde ein ESP8266 Modul von JOY-IT verwendet. Das Modul wurde für eine unidirektionale Kommunikation zum Robonova programmiert. Somit wurde das WiFi Modul an den UART Header angeschlossen, welcher für das BLE-Modul gedacht war.

### Jlink Segger Debugger

Ein Bild, das Flasche, sitzend, Oberfläche, Tisch enthält.

Automatisch generierte Beschreibung

Mit dem j-link Segger Debugger kann man per SWD Interface den Microcontroller flashen und debuggen. Dazu schließt man ihn an dem vorgesehenen Header auf der Backplane an (auf schwarze Markierung achten).

Es gibt 2 Debugger, der Debugger mit den offenen Leitungen scheint nicht zu funktionieren. Der zweite Debugger funktioniert, jedoch muss man drauf achten, dass das Kabel zwischen Debugger und Microcontroller gerade liegt, da es scheinbar einen Wackelkontakt hat.

### FTDI FT232RL USB-UART Modul

Ein Bild, das Elektronik, Schaltkreis enthält.

Automatisch generierte Beschreibung

Dieses Modul ist Teil der Toolchain für das Servomotor flashen. Es ermöglicht die Ausgabe von UART Nachrichten über den PC. Dazu schließt man es mit einem Mini-USB Kabel am PC an. Auf dem PC verbindet man sich mit der seriellen Schnittstelle (über das Terminal, Putty, CoolTerm oder ähnlichem) und kann damit Nachrichten senden und empfangen. Man kann zwischen einem Signalpegel von 5V und 3,3V entscheiden.

### Converter Platine

Ein Bild, das Elektronik, Schaltkreis enthält.

Automatisch generierte Beschreibung

Die Converter Platine wird ebenfalls zum flashen der Servomotoren benötigt. Bauartbedingt nutzt der Servomotor ein invertiertes Signal. Deswegen ist auf der Platine ein HEX Inverter von ST verbaut. Dieser invertiert das eingehende Signal und gibt es an die Signalleitung des Servomotors weiter. Zusätzlich gibt es noch einen Header, um eine Stromversorgung für die Servomotoren anzuschließen.

Die Platine funktioniert nicht. Entweder sind Lötstellen defekt oder der HEX Inverter ist kaputt.

## Aufbau

Dieser Abschnitt beschreibt den Hardwareaufbau bezogen auf den Robonova.

Für dieses Projekt werden die Servomotoren und das WiFi-Modul genutzt.

Im Robonova sind 16 Servomotoren verbaut. Je 3 Motoren pro Arm und je 5 Motoren pro Bein. Die große Anzahl ist notwendig, da sich die Servomotoren nur in einer Achse bewegen können. Das menschliche Schulter-, Hüft- und Fußgelenk kann sich aber in mehrere Achsen bewegen, was mit 2 Servomotoren bestmöglich repliziert werden kann.

Durch den Ausfall der UART Header auf einer Backplane können nicht alle Motoren angesteuert werden. Mit einem UART-Splitter werden nur die 6 Servomotoren der Arme an dem einzig funktionierenden UART Header der zweiten Backplane angeschlossen.

Das WiFi-Modul ist an dem UART Header des BLE-Moduls angeschlossen.

## Entwicklungsumgebung

In diesem Abschnitt wird erklärt wie eine neue Entwicklungsumgebung erstellt werden kann.

In den älteren Dokumentationen ist beschrieben wie man eine Entwicklungsumgebung auf Windows mittels STM32 CubeMX und STM32 Cube Workbench erstellt. In diesem Projekt wurde statt der Eclipse basierenden Cube Workbench VS Studio Code auf MacOS genutzt.

### Downloads und Installationen

Auf MacOS muss man Xcode installieren, da mit diesem der C-Compiler und das make Tool installiert werden. Xcode kann im Appstore installiert werden

Des Weiteren muss man die ARM GCC Toolchain installieren. Diese kann unter <https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads> heruntergeladen werden.

Für den Debugger wird das J-Link Segger Software Package benötigt. Unter <https://www.segger.com/downloads/jlink/> kann man dieses runterladen.

Zur Konfiguration des Basiscode wird STM32 CubeMX benötigt. Es kann unter <https://www.st.com/en/development-tools/stm32cubemx.html> heruntergeladen werden.

Zuletzt benötigt man Visual Studio Code und die Cortex-Debug Erweiterung. VS Code kann unter <https://code.visualstudio.com/download> heruntergeladen werden, die Erweiterung kann in VS Code über den Extensions Marketplace installiert werden.

### Projekt erstellen

Um ein neues Projekt zu erstellen startet man STM32 CubeMX und wählt „New Project from MCU“. In der folgenden Liste wählt man „STM32F405RG“. Da VS Code keinen automatischen make Prozess hat muss in CubeMX unter Project Manager → Project → Toolchain/IDE „makefile“ ausgewählt werden. Für den richtigen Debugmodus muss unter Pinout & Configuration → System Core → SYS → debug „JTAG (4 pins) gewählt werden. Nach diesen Schritten ist die Basiskonfiguration abgeschlossen und das CubeMX Project kann gespeichert und generiert werden.

Die von CubeMX generierte Umgebung wird mit Visual Studio Code geöffnet. Dort kann entweder ein Task für den build-Prozess erstellt werden oder händisch in der Konsole mit „make“ der Code kompiliert werden. Als nächstes muss unter Run → Open configuration eine „cortex-debug“ Konfiguration erstellt werden. Hier sind das richtige Interface (hier SWD), das Device (STM32F405RG), die Executable (.elf file des build-Prozess) und der Servertype (jlink) wichtig. Unter <https://github.com/Marus/cortex-debug/wiki> sind einige Beispiele gegeben wie die Konfiguration angepasst werden kann.

Nach diesen Schritten sollte die Toolchain stehen. Mit CubeMX kann der Microcontroller und dessen Modulinstanzen konfiguriert werden und der Code generiert werden. In VS Code kann Nutzer-Code ergänzt werden, aus dem Projektcode eine Executable erstellt werden und der Microcontroller programmiert und gedebugged werden.

## Software

In diesem Abschnitt wird die gesamte Software des Robonova erklärt. Es wird gestartet mit der Konfiguration des Microcontroller und dessen Modulinstanzen. Anschließend werden die selbstgeschriebenen Treiber für die Servomotoren und das WiFi Modul vorgestellt.

### Microcontroller Konfiguration

Allgemein gesehen wurden für dieses Projekt nur zwei Arten an Konfigurationen gemacht. Es wurden alle Peripheriegeräte über UART angeschlossen. Somit ergibt sich eine Konfiguration für die UART Instanz des WiFi Modul und ähnliche Konfigurationen für die UART Instanzen für die Servomotoren.

#### WiFi UART

Das WiFi Modul ist an dem Header angeschlossen welcher mit den USART1 Pins des Microcontrollers verbunden ist.

Das WiFi Modul sendet Nachrichten mit einer Baudrate von 115200 Bit/s, einer 8 Bit Wordlength, einem Stopbit und sendet keine Paritätsinformation.

Der Treiber für das WiFi Modul nutzt DMA um Nachrichten zu empfangen und in einem Buffer zu speichern. Dafür muss der RX DMA konfiguriert werden und das globale Interrupt des DMA aktiviert werden. Da es sich um den empfangenden DMA handelt wurde Peripherie zu Speicher als Richtung gewählt. Der DMA wird im normalen Modus verwendet und es wird eine Datenbreite von einem Byte gewählt.

#### Servomotor UART

Die UART Instanzen 2, 3, 4 und 6 führen jeweils zu einem der Servomotor Header auf der Backplane. Will man diese nutzen müssen die Instanzen konfiguriert werden.

Das Protokoll der Servomotoren ist so ausgelegt, dass eine Baudrate von 19200 Bit/s, eine 8 Bit Wordlength, 2 Stopbits und keine Paritätsinformationen benötigt werden.

Der Servomotor Treiber benötigt die Aktivierung der globalen Interrupts.

### Servomotor Treiber

Der Servomotor Treiber wurde im Rahmen eines vorherigen Projekts von einem anderen Team geschrieben.

Die Implementierung ist dreigeteilt. Auf der abstrakten Schicht gibt es die Konfiguration und das ServoInterface, auf der Hardwareschicht ist der ServoDriver.

Im ServoDriver ist alles was auf Hardware Abstraction Layer (HAL) geschieht implementiert. Dabei werden die von ST gebotenen HAL\_UART Treiber genutzt. Der ServoDriver kümmert sich um die HAL\_UART Callbacks und das Senden und Empfangen von Nachrichten mit HAL\_UART Interrupt Funktionen.

Im ServoInterface werden die Funktionen zur Servomotoren-Ansteuerung zu implementiert. Die Funktionen erstellen Nachrichten gemäß dem ServoMotor Protokoll und sendet diese mit den ServoDriver Funktionen. Das ServoInterface bietet Funktionen, um den Servomotor zu aktivieren, seine ID auszulesen, die Kommunikation zu zurückzusetzten und dessen Position setzten und auszulesen.

Das ServoInterface benötigt Informationen an welchen Servomotor die Nachricht gesendet werden soll. Dazu muss es wissen an welcher UART Instanz der Servomotor angeschlossen ist und welche ID der Servomotor hat. Diese Information ist in einer Struktur gespeichert, welche jedem Robonova Gelenk die Informationen zuordnet. Zusätzlich sind defines implementiert welche den Code leichter lesbar macht, da jedem Gelenk ein Name gegeben wurde, anstatt nur Nummern zu verwenden.

### WiFi Treiber

Der WiFi Treiber besteht aus nur einer Funktion, da nur das Empfangen von Daten benötigt und implementiert wurde. In dieser Funktion wird die Nachricht aus dem DMA Buffer gelesen und gemäß des Nachrichtenschemas die Informationen in einer globalen Variablen gespeichert.

# Servomotoren

Jeder Servomotor ist so programmiert, dass er passend zum Protokoll welches, in der Servomotor Dokumentation beschrieben ist, reagiert. Zusätzlich besitzt jeder Servomotor eine Eindeutige ID, dadurch kann er erkennen ob die Nachricht auf dem UART-Bus an ihn gesendet wurde.

## Programmieren

Dieses Kapitel basiert auf theoretischem Wissen. Durch die defekte Converter Platine konnte leider nie ein Servomotor in Praxis programmiert werden.

Allgemein werden Microcontroller über Evaluationsboards oder ähnlichem auf denen ein Sockel ist oder der Controller fest verlötet ist programmiert. Damit die Atmega Chips nicht für jeden Programmiervorgang aus dem Motor ausgelötet werden müssen wurde von Florian Laufenböck ein eigener Bootloader entwickelt. Dieser ermöglicht es über UART die Servomotoren zu programmieren.

Dazu wird das USB-UART Modul und die Converter Platine benötigt. Das USB-UART Modul wird per USB an dem PC angeschlossen und die RX, TX und GND Pins mit den passenden Pins auf der Converter Platine verbunden. Nach Anschluss einer externen Stromversorgung sollte man mit zB. CoolTerm Nachrichten an die Signalleitung des Servomotor Header senden können. In CoolTerm müssen folgende Einstellungen getroffen werden: Baudrate: 19200; Data Bits : 8; Stop Bits: 2; Parity: none; transmit character delay: 10ms.

Zum programmieren des Servomotors sollte man diese Schritte ausführen:

1. Nur EINEN Servomotor auf der Converter Platine anstecken
2. Die Stromversorgung anschalten
3. Sofort innerhalb 1s beliebig viele 's' verschicken.
4. Mit einer beliebigen Nachricht testen, ob man im Bootloader ist(z.B.0x420x00). Liest man 0x42 zurück, passt alles. Ansonsten die Kommunikation mit 0x00 0x00 0x00 0x00 zurücksetzen und diesen Schritt wiederholen.
5. Mit der Nachricht 0x70 0x00 den Flash-Modus erreichen. Überprüfen ob 0x70 zurückgelesen werden konnte.
6. Die gesamte hex-Datei kopieren(vom ersten bis zum abschließenden FF) kopieren und per Copy-Paste im Text-Modus der seriellen Konsole einfügen. Die Übertragung kann durchaus etwas länger dauern, da nach jedem Zeichen 10ms gewartet wird.
7. Ist die Übertragung abgeschlossen, kann durch die Nachricht 0x69 0x00 überprüft werden, ob alles gut gegangen ist. Wird der Wert 0x01 zurückgelesen, passt alles. Wird 0xFF zurückgelesen, sollte man den Aufbau überprüfen und zu Punkt 5) zurückkehren.
8. Den Bootloader mit dem Befehl 0x71 0x00 verlassen. Sicherstellen, dass der Wert 0x71 zurückgelesen werden konnte.

Detailliertere Informationen zum Bootloader kann man in der Servomotor Dokumentation finden.

Stundentabelle Felix Dollinger:

|  |  |
| --- | --- |
| **Aktivität** | **Stunden** |
| Wöchentliche Meetings | 35 |
| Sprints | 30 |
| Entwicklungsumgebung erstellen | 10 |
| Einlesen Robonova und Servomotor Ansteuerung | 7 |
| Robonova programmieren und erste Servomotor Ansteuerung | 2 |
| Servomotor Treiber anpassen | 1 |
| Einlesen und Aufbau für das Servomotor flashen erstellen/nachvollziehen | 5 |
| Vergebliche Versuche die Servomotoren zu flashen und Fehlersuche | 20 |
| Treiber für WiFi Modul erstellen | 5 |
| Fehlersuche aufgrund von (teil)defekter Hardware) | 10 |
| Dokumentation erstellen | 15 |