Berliner Hochschule für Technik

Fachbereich VII - Elektrotechnik – Mechatronik – Optometrie
Studiengang Elektromobilität
Wintersemester 2023/24

Autonomes Fahren und intelligente Sensoren (WiSe 23/24)
Dokumentation Projekt „Racing Autonomous Technology (RAT)“

Me Mouse.mp3

Name: Nico Hinrichs (930836), Marcus Stake (929605)

Betreuer: Prof. Dipl.-Ing. Koshan Mahdi

Studienfach: Elektromobilität

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.

1 Einleitung

1.1 Erläuterung

Micromouse ist ein Robotik Wettbewerb, in dem sich kleine, autonome Roboterfahrzeuge in einem Labyrinth zurechtfinden müssen. Die Micromouse-Roboterfahrzeuge sind vollständig autonom. Sie bewegen sich ohne Hilfe von außen durch das Labyrinth. Es ist dem Entwickler freigestellt, wie das Fahrzeug die Wände des Labyrinthes erkennt; verbreitet sind vorn am Fahrzeug angebrachte und schräg nach außen ausgerichtete Infrarot-Abstandssensoren. Unsere Micromouse soll als Einstieg in diese Thematik dienen.

1.2 Aufgaben und Ziele

Unsere sogenannte MicroRat soll als Teil des Modules „Projekt Steuergeräteentwicklung (SoSe24)“ von uns entworfen werden. Dabei liegt der Fokus der Arbeit in der Planung, Erstellung sowie Benutzer Freundlichkeit, damit kommende Studierende auf der Grundlage unsere Plattform eigene Projekte umsetzen kann. Die Implementierung eines Pfadfindungsalgorithmus ist nicht vorgesehen.

Das Platinen design steht im Rahmen des Modules an wichtigster stelle. Hierfür benutzen wir das Programm KiCad sowie Fusion 360. Der XMC 1402 ist dabei unser Prozessor, der für die Ansteuerung der einzelnen Bauteile verantwortlich ist.

Die Ziele sind eine funktionierende Platine unter Berücksichtigung der Designregeln zu designen und diese in Betrieb nehmen zu können. Ein simpler Code wird eingebettet, um die funktionierende Ansteuerung zu demonstrieren.

1.3 Anforderungen

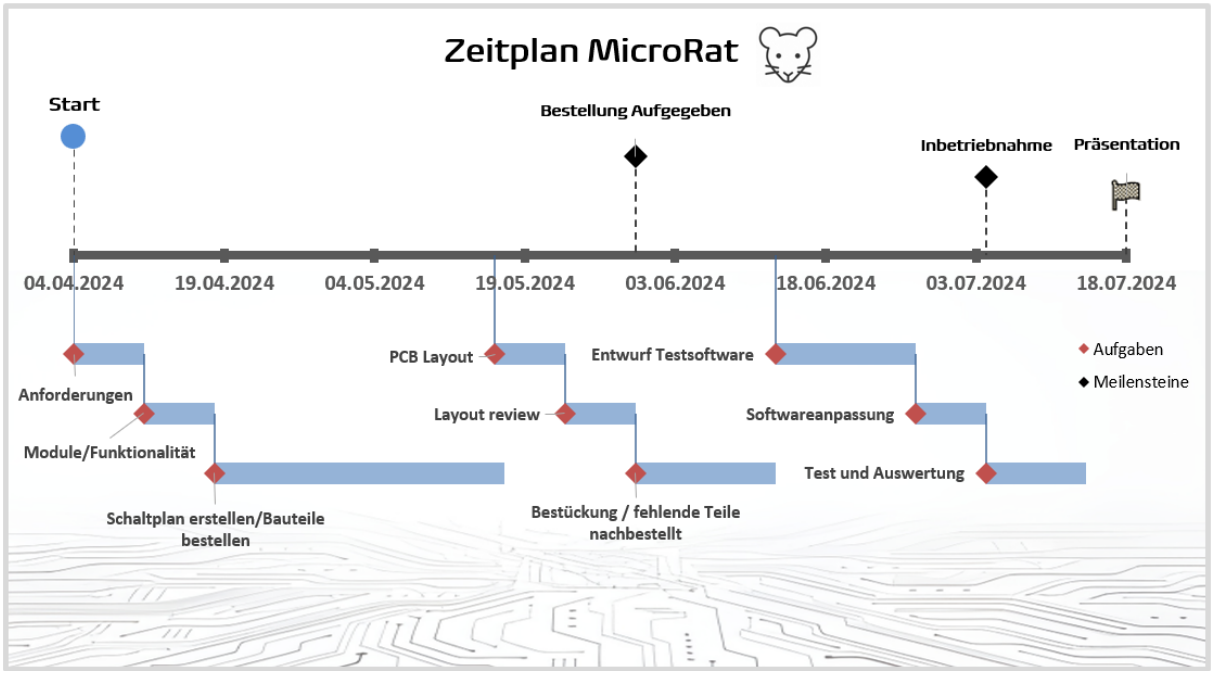
- Die MicroRat kann alle verbauten Sensoren Ansteuern sowie auslesen
- Die Größe des Projektes wird auf ein Minimum reduziert
- Das Debuggen und Auslesen (UART) von Daten wird über eigene Stecker möglich sein
- Die MicroRat ist erweiterbar, sodass in der Zukunft mehr Sensoren implementiert werden können

1.4 Struktur der Arbeit

Die Dokumentation wird mithilfe eines selbst erstellten Zeitplanes, entsprechenden Diagrammen und kommentiertem Code unterstützt, um so die Funktions- und Herangehensweise besser erklären zu können. Alle Testdaten sowie Beweise sind im Abschnitt 5 dieser Dokumentation zu finden.

2 Zeitplan und Aufgabenverteilung

2.1 Zeitplan und Meilensteine



Aufgaben

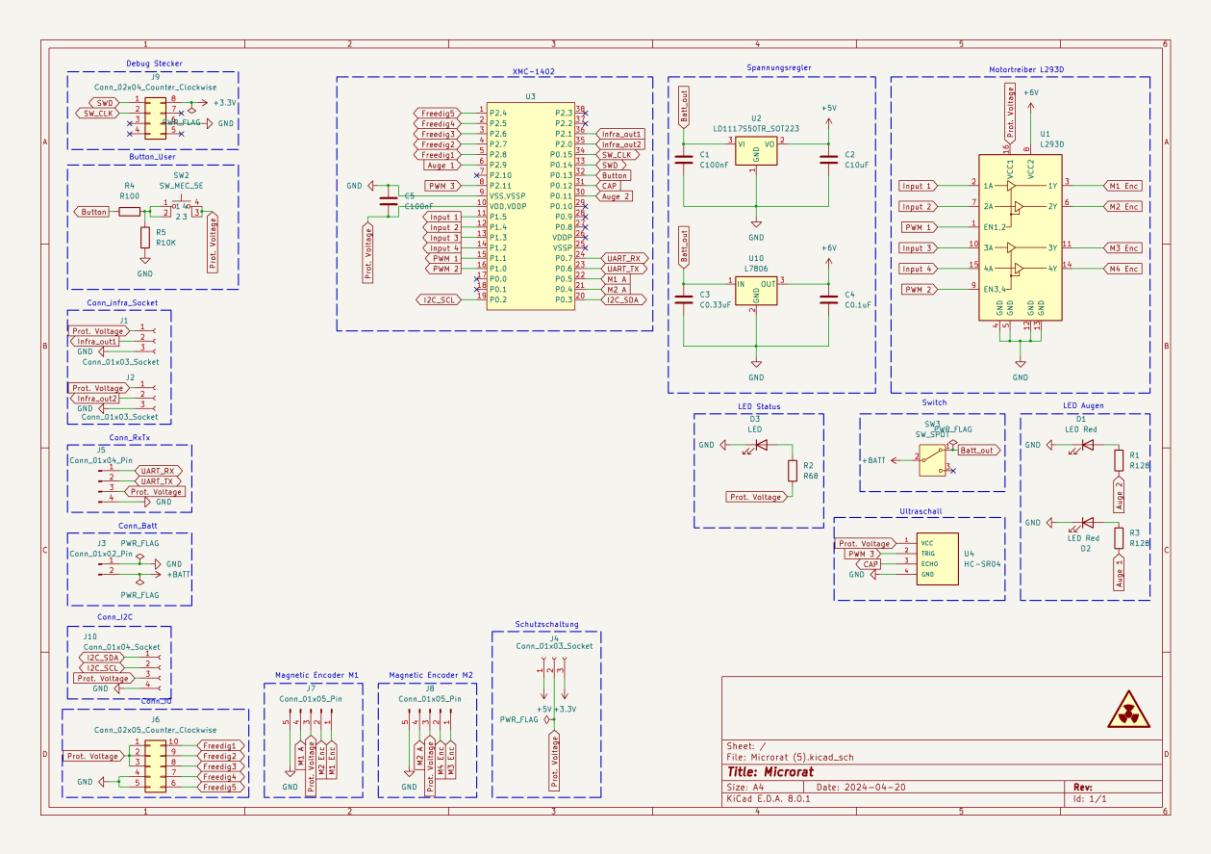
Start	Ende	Dauer	Bezeichnung
04.04.2024	11.04.2024	7	Anforderungen
11.04.2024	18.04.2024	7	Module/Funktionalität
18.04.2024	16.05.2024	29	Schaltplan erstellen/Bauteile bestellen
16.05.2024	23.05.2024	7	PCB Layout
23.05.2024	30.05.2024	7	Layout review
30.05.2024	13.06.2024	14	Bestückung / fehlende Teile nachbestellt
13.06.2024	27.06.2024	14	Entwurf Testsoftware
27.06.2024	04.07.2024	7	Softwareanpassung
04.07.2024	14.07.2024	10	Test und Auswertung

3 Funktionsbeschreibung

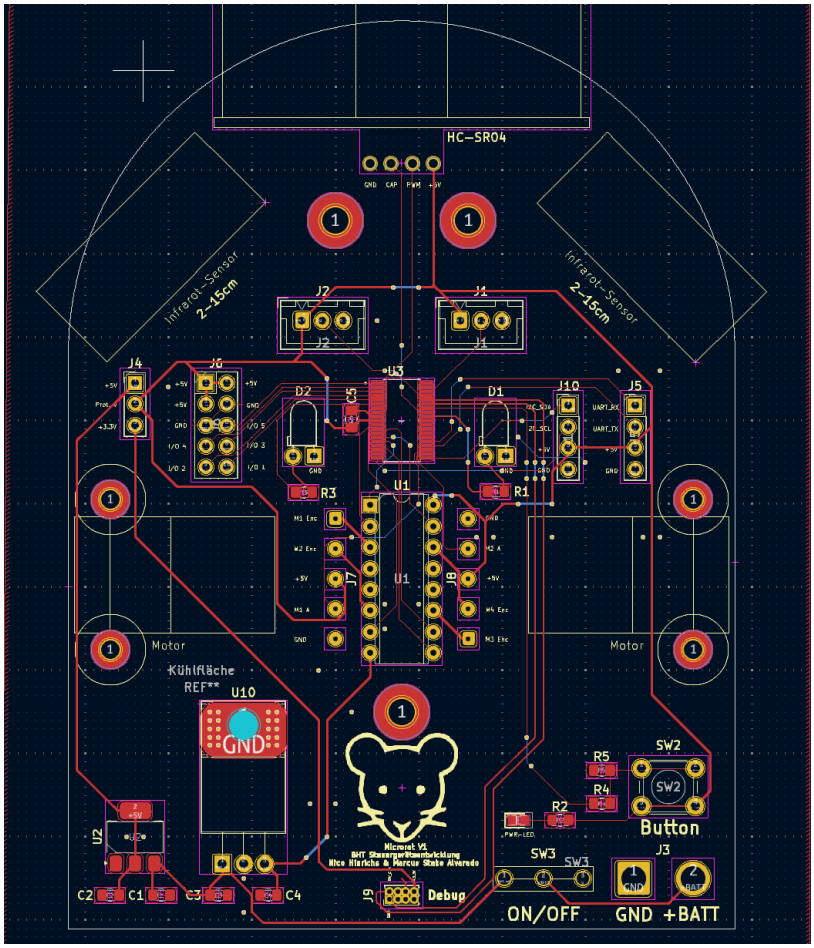
3.1 KiCad

KiCad ist ein freies ECAD-Programmpaket zur Entwicklung von Leiterplatten in der Elektronik.

3.1.1 Schaltplan



3.1.2 PCB-Layout

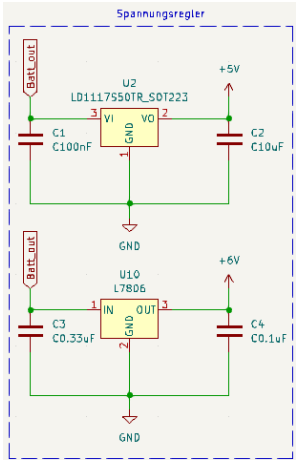


Laut des Wettbewerbes für Micromouse gibt es diverse Größen Kategorien, in der bestimmten Maßen vorgesehen sind. Wir haben

3.2 Spannungsversorgung mit Schutzschaltung

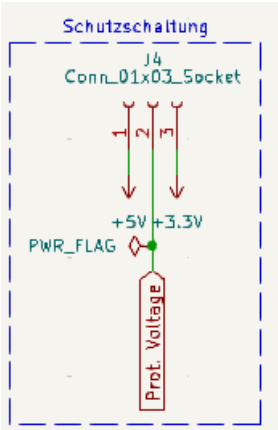
Für unser Projekt haben wir als Energiequelle einen selbsterstellten 2S1P Lithium-Ionen-Akku gewählt. Diese Entscheidung beruht auf mehreren technischen Überlegungen. Erstens liefert der 2S1P-Akku eine Nennspannung von 7.2V, die ideal für unsere elektronischen Komponenten ist und uns ermöglicht, aufwendige Spannungswandler zu vermeiden. Zweitens ermöglicht die hohe Energiedichte des Lithium-Ionen-Akkus, viel Energie in einem kompakten und leichten Paket zu speichern, was die Tragbarkeit unseres Systems verbessert.

3.2.1 Der LDO (Low-dropout regulator)



Für unser Projekt haben wir uns entschieden, Low Dropout (LDO) Spannungsregler zu verwenden. Diese Entscheidung beruht auf der Fähigkeit von LDO-Reglern, eine stabile Ausgangsspannung auch bei geringen Eingangsspannungsdifferenzen zu liefern. Der für das Board ausgewählte LDO bietet einen fixed output mit 5V und 800mA was für unsere MicroRat mehr als genug ist für die vorhandene Elektronik. Da wir die Motoren mit optimalem Wirkungsgrad betreiben wollen (bei 6V), entschieden wir uns ebenfalls einen separaten LDO für die Motoren zu nutzen. Dieser hat ebenfalls einen fixed output aber von 6V und 1,5A.

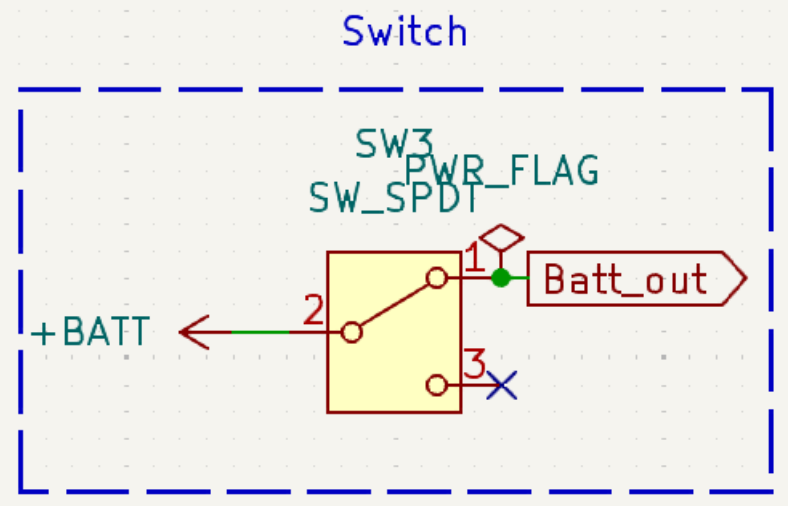
3.2.2 Jumper-Pin-Konfiguration



Im Rahmen vieler Projekte haben wir uns intensiv mit dem XMC 4700 Mikrocontroller auseinandergesetzt. Dabei sind wir auf die effiziente und vielseitige Methode gestoßen, mittels dreipoliger Pin-Header und Jumpers die Spannungen zu wechseln. Diese Methode hat uns inspiriert, eine ähnliche Konfiguration als Schutzvorkehrung in unserem eigenen Design zu implementieren. Mittels eines Jumpers ist der Bediener in der Lage die Spannungsebene zu wechseln von 3.3V (Debug) zu 5V zu wechseln. Mit dieser Methode haben wir eine

zusätzliche Schutzvorkehrung, sodass niemals beide Spannungen auf dem Board anliegen und somit potenzielle Fehler vermieden werden.

3.2.3 Schutzschalter

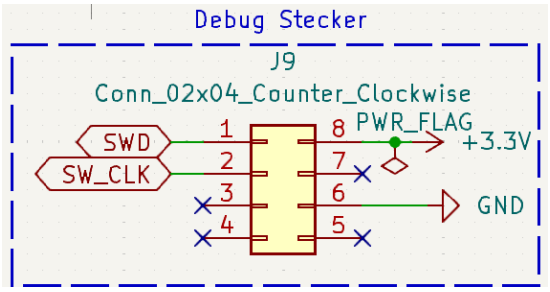


Um die Sicherheit und Kontrolle über die Spannungsversorgung weiter zu erhöhen, haben wir einen physischen Schalter am Versorgungseingang unseres Systems eingebaut. Diese Maßnahme bietet mehrere Vorteile. Erstens ermöglicht der Schalter eine einfache Kontrolle der Stromzufuhr, sodass die Stromzufuhr zum gesamten System schnell und einfach ein- und ausgeschaltet werden kann. Dies ist besonders nützlich während der Wartung oder bei Konfigurationsänderungen, da die Spannungsversorgung sicher unterbrochen werden kann, bevor Änderungen vorgenommen werden. Zweitens verhindert der Schalter ungewollte Stromzufuhr, indem er sicherstellt, dass das System nur dann mit Strom versorgt wird, wenn es tatsächlich benötigt wird. Dadurch wird das Risiko von Schäden durch versehentliche Kurzschlüsse oder Überspannungen, die auftreten können, wenn das System unerwartet eingeschaltet wird, reduziert. Der Schalter dient als zusätzliche Sicherheitsebene neben der Jumper-Konfiguration. Während die Jumper es ermöglichen, die Betriebsspannung gezielt auszuwählen, stellt der Schalter sicher, dass die Stromzufuhr nur dann erfolgt, wenn alle Konfigurationen korrekt vorgenommen wurden. Dies minimiert das Risiko, dass falsche Spannungen an die Schaltungsteile angelegt werden. Darüber hinaus erhöht der Schalter die Benutzerfreundlichkeit unseres Systems. Anwender können das System bequem und sicher ausschalten, ohne die Stromquelle direkt trennen zu müssen. Dies ist besonders praktisch in Umgebungen, in denen das Gerät regelmäßig ein- und ausgeschaltet werden muss.

3.3 Schnittstellen und Treiber

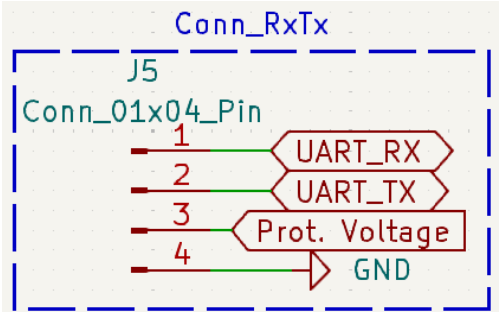
3.3.1 Der Debuganschluss

Die MircoRat hat mehrere Schnittstellen, um mit diesen interagieren zu können. Der Debug Stecker befindet sich neben dem An-Schalter und ist für das Programmieren des XMC-1402 zuständig. Über diese Schnittstelle kann ein Debugger über ein Flachbandkabel mit dem XMC4200-Prozessor verbunden werden. Der 2x8 Pin-Header, der als Anschluss am Steuergerät dient, ist möglichst nah am Mikrocontroller platziert, um Störungen durch die Leiterbahnen zu minimieren. Das Flashen des Boards erfolgt dann über SWD und SW_CLK.



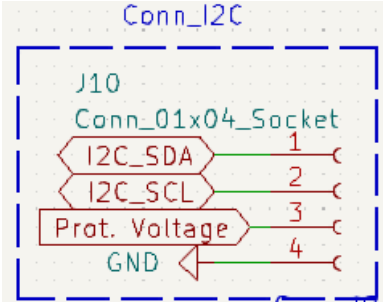
3.3.2 UART-Kommunikation

Wir haben vier Pin-Header als Schnittstelle integriert, um eine UART-Kommunikation zu ermöglichen. Diese Konfiguration erlaubt es uns, verschiedene Sensoren auszulesen und Daten effizient zu übertragen. Durch die Verwendung dieser Pin-Header wird eine zuverlässige und einfache Verbindung zwischen den Komponenten gewährleistet, was die Integration und den Austausch von Informationen innerhalb des Systems erleichtert.



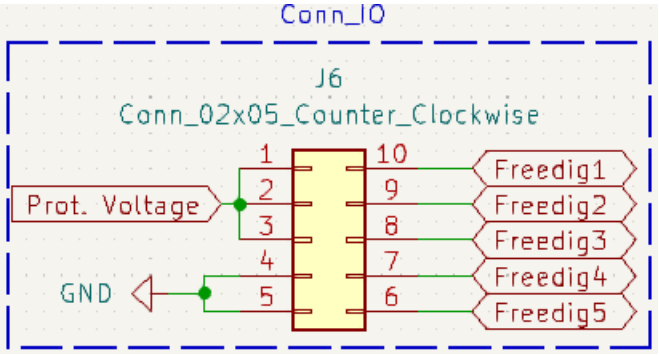
3.3.3 I²C-Kommunikation

Ähnlich wie bei der UART-Schnittstelle haben wir auf unserem Board eine I2C-Schnittstelle mit vier Pin-Header realisiert. Zwei Pins sind für die Stromversorgung zuständig, während die anderen beiden für SCL und SDA verwendet werden. Die Bereitstellung dieser Schnittstelle war uns besonders wichtig, da viele Sensoren über I2C kommunizieren und wir unseren Nutzern diese Option ebenfalls anbieten wollten.



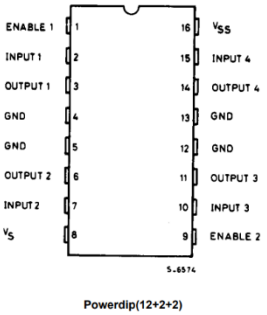
3.3.4 Extra DIGITAL_IO Pins

Im Rahmen der Erweiterung unseres Projekts möchten wir den zukünftigen Nutzern zusätzlich fünf freie digitale IO-Pins zur Verfügung stellen. Diese Pins bieten die Möglichkeit, verschiedene externe Komponenten anzuschließen und erweitern somit die Flexibilität des Systems. Es ist jedoch wichtig zu beachten, dass diese Pins ausschließlich als Eingänge genutzt werden können. Sie eignen sich ideal, um Sensordaten zu erfassen oder Statusinformationen von externen Geräten zu lesen.



3.3.5 Motortreiber

Um unsere Motoren ansteuern zu können, benutzen wir den L293D Motorcontroller. Dieser besitzt zwei unabhängige H-Brücken, um beide Motoren separat zu bedienen.



Die Pin-Belegung sieht wie folgt aus:

Pin	**Funktion**

Enable 1/2	PWM für Motoren. Duty Cycle = Speed
Input 1/2/3/4	Richtungssteuerung
Output 1/2/3/4	Anschlüsse Motor (1&2 / 3&4) gehören zsm.
GND	Ground
Vs	6V Motorspannung
Vss	5V Logikspannung

Legt man Anhand des folgenden Truth-Table die Spannungen an den Inputs entsprechend an, kann der Motor rückwärts sowie vorwärts fahren. Die Outputs gehen an die Motoren. Wird an den Enable Pins ein PWM angelegt von maximal 5 KHz, kann mit dem Dutycycle die Geschwindigkeit eingestellt werden, mit der sich die Motoren drehen.

INPUT 1	INPUT 2	ENABLE 1,2	RESULT
0	0	1	STOP
0	1	1	ANTI-CLOCKWISE
1	0	1	CLOCKWISE
1	1	1	STOP

3.4 Hardware

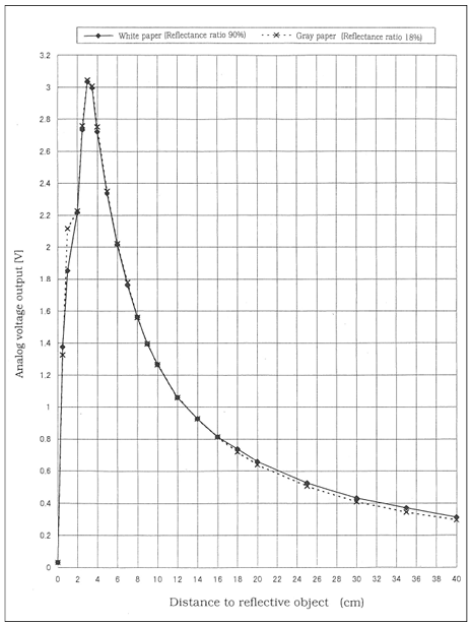
3.4.1 Infrarotsensor

Die analogen Infrarotsensoren von Sharp nutzten eine Infrarot-LED, um kontinuierlich Infrarotlicht auszusenden. Dieses Licht wird von Objekten reflektiert, und ein Fotodetektor im Sensor wandelt die reflektierte Intensität in ein analoges Signal um. Die Signalstärke korreliert mit der Entfernung zum Objekt. Die Infrarotsensoren wurden integriert, um der MicroRat eine erweiterte Wahrnehmungsfähigkeit zu verleihen. Diese Infrarotsensoren dienen dazu, die

1: Signal (analog)
2: GND
3: VCC (5V)

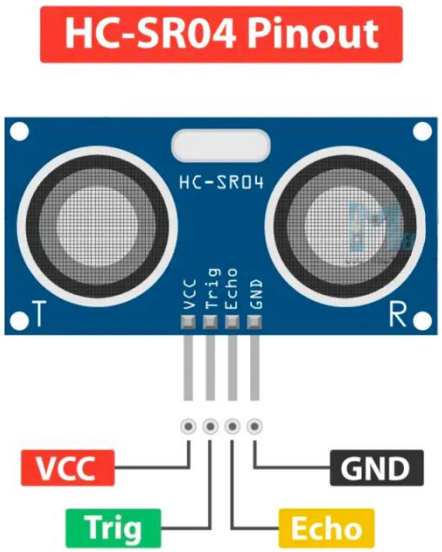
Umgebung der zu erfassen. In einem aufgebauten Labyrinth könnte so die Kollision mit den Wänden vermieden werden.

Im unteren Bild erkennt man den Zusammenhang zwischen der Ausgangsspannung und der Entfernung in cm. Im Abschnitt Dummy-Code ist ein Beispiel-Code geschrieben, um die Entfernung in Ticks zu bekommen und diese per Uart ausgeben zu lassen.

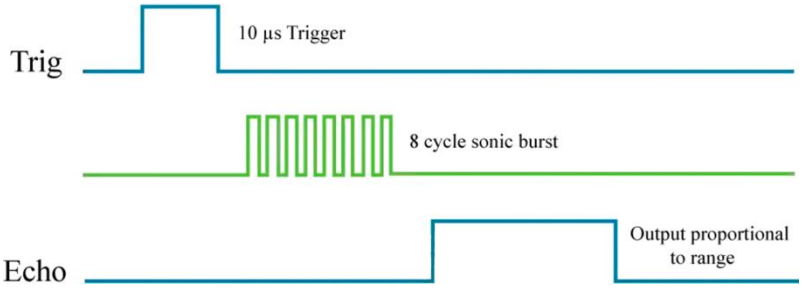


3.4.2 Ultraschallsensor

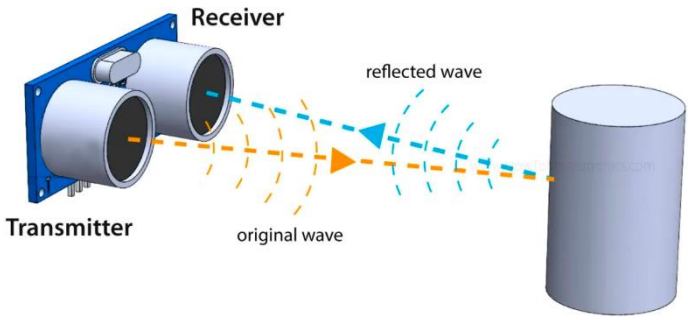
Der Ultraschallsensor und sein zugehöriges Pin-Layout sehen wie folgt aus:



Im VCC und GND-Pin sind entsprechend vom µC +5V und GND verbunden. Im Trig Pin wird vom µC ein Impuls gegeben, sodass der Sensor anfängt, den Abstand mittels Ultraschalls zu ermitteln. Der Abstand wird dann mit dem Echo Pin rausgegeben.



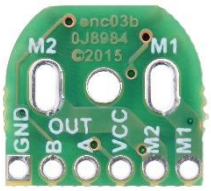
Mit dem Zeitablauf kann man die Funktion des Sensors leicht ablesen. Zuerst muss ein High Signal am Trig Pin mit einer Länge von 10us gesetzt werden. Dies wird optimalerweise mit einem PWM-Signal erzeugt. Nach einer sehr kurzen Zeit sendet der Ultraschall Sensor einen 8 cycle sonic burst aus. Diese sind im folgenden Bild dargestellt.



Die Schallwelle wird vom Receiver aufgenommen. Der entsprechende Abstand wird dann mit der Länge des High Signal am Echo Pin symbolisiert. Ein kompletter Zyklus ist abgeschlossen. Wenn man das PWM-Signal mithilfe des duty-cycles so einstellt, dass alle 60ms ein Impuls mit der Länge 10us ausgesendet wird, kann man die maximale Geschwindigkeit des Sensors ausnutzen. Die Maximale Distanz des Sensors beträgt ca. 4m was einer High-time am Echo Pin von 38ms entspricht. Falls sich kein Objekt in dieser Reichweite befindet, wird dennoch ein Signal von 38ms gesendet.

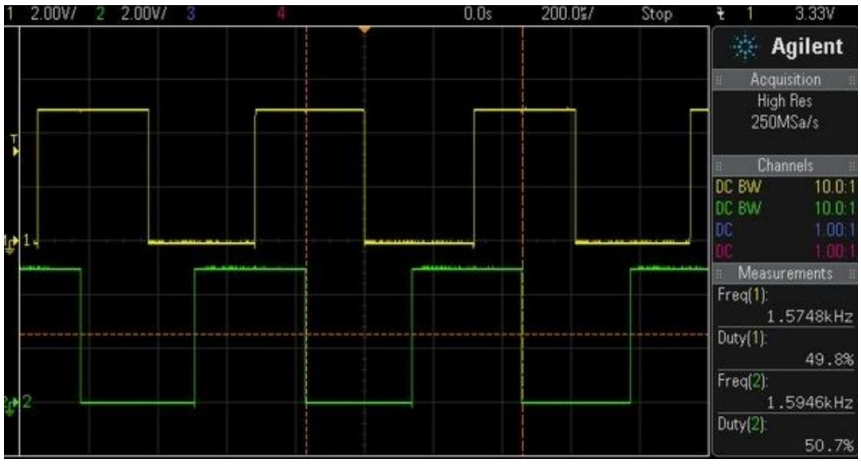
3.4.3 Magnetic Encoder

Mit dem Magnetic Encoder können wir anhand der Signale von den Hallsensoren ermitteln, wie schnell sich der Motor dreht. Dieser sieht wie folgt aus:



Am VCC und GND-Pads wird die Betriebsspannung des Encoders angelegt, um die Hallsensoren auf der Rückseite betrieben zu können. Da wir ein Hallsensor paar in einem 45º Winkel zueinander angeordnet haben, kann die Drehrichtung bestimmt werden, da ein Hallsensor früher auslösen wird. In unserem Falle benutzten wir wegen fehlender POSIF App des XMC jeweils pro Motor nur den Ausgang A. Die Ticks werden über ein internen Interrupt gezählt. Die Bestimmung der Richtung kann zudem über die Inputs des Motortreibers bestimmt werden. Zuletzt gehen die Pads M1 und M2 an den Motor, um eine Spannung anzulegen.

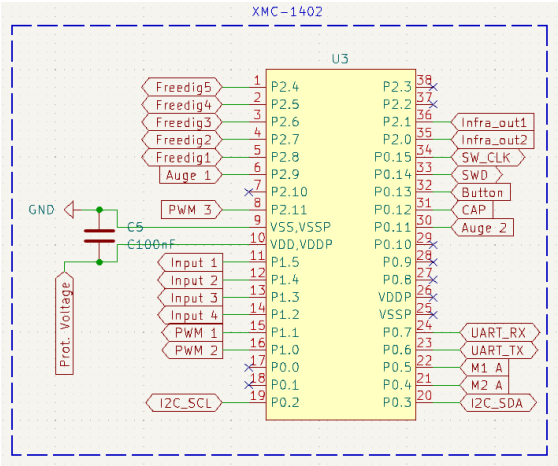
Auf dem Bild erkennt man, dass der Encoder auf die Motorwelle aufgesteckt wird. Die Kontakte des Motors werden über die großen Aussparungen (Bild bla) durchgeführt, um so noch die notwendige Verbindung herzustellen. Am Ende ist ein sich drehender Magnet dicht über den Hallsensoren platziert. Da der Magnet direkt mit der Motorwelle verbunden ist, kann präzise die Umdrehungszahl bestimmt werden.



Über den Screenshot des Oszilloskop erkennt man das versetzte auslösen der Hallsensoren gut.

3.4.4 XMC1402-T038X0128 AA

Für unsere Anwendung haben wir uns für den Microcontroller XMC1402-T038X0128 AA von Infineon entschieden. Diese Wahl basiert auf unseren bisherigen Erfahrungen mit Infineon-Microcontrollern und deren Entwicklungsumgebung. Nach mehreren Tests mit der DAVE IDE Umgebung haben wir festgestellt, dass alle benötigten APPs im Prozessor vorhanden sind und somit dieser Prozessor gut für unser Steuergerät geeignet ist.

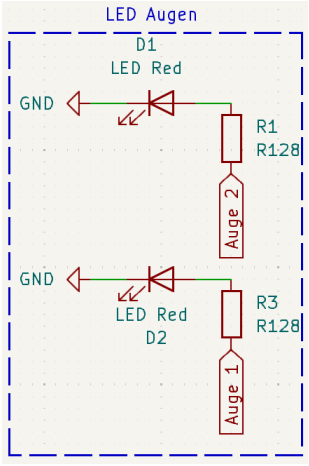


Es ist wichtig zu beachten das bei der Pin-Belegung alle APPs an der richtigen Stelle sich befinden das DAVE nicht jede APP auf jedes Pin zuweisen kann. Eine genaue Planung ist deswegen essenziell.

An den Versorgungspins des Mikrocontrollers haben wir einen Kondensator parallelgeschaltet, um die Stromversorgung zu stabilisieren. Der Kondensator filtert Spannungsschwankungen und liefert kurzfristig Energie, wodurch Rauschen reduziert, und die Betriebssicherheit erhöht wird. Dies sorgt für einen stabilen Betrieb des Mikrocontrollers und verbessert die Gesamtleistung des Systems.

3.4.5 LEDs (GPIO und Power-LED)

Die zwei LEDs (AUGE 1 und AUGE 2) sind als Eingänge als auch Ausgänge definiert. Ihr Leuchtzustand ist rein von dem Programm auf dem XMC abhängig und somit vielseitig verwendbar wie z.B. als Feedback Methode im Dummycode. Die Power-LED ist direkt mit dem 5V Spannungsausgang des LDOs verbunden.

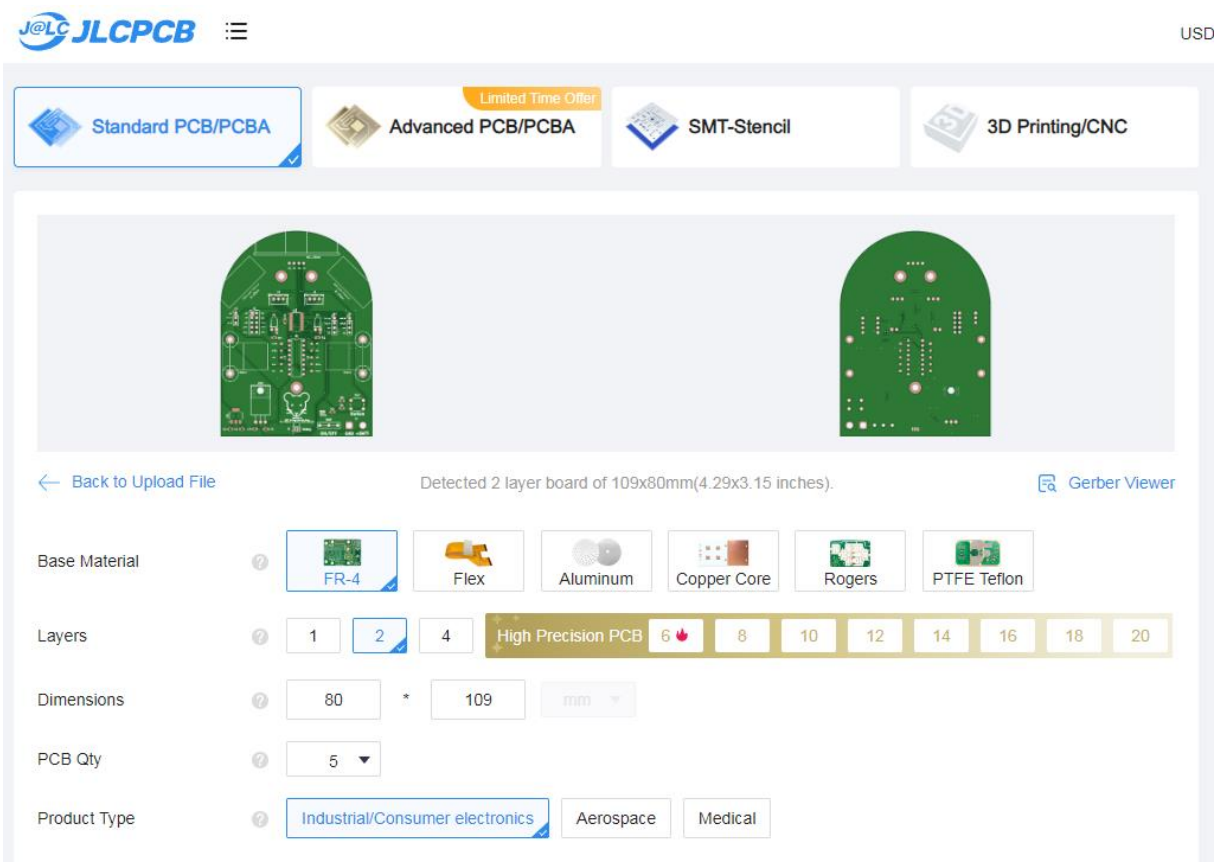


4 Platinenherstellung

Unsere Platine wurde von JLPCB hergestellt, dazu musste die gerber Datei exportiert und auf ihre Webseite hochgeladen werden. Auf KiCAD lassen sich die gerber Dateien unter dem folgenden Menü erzeugen:

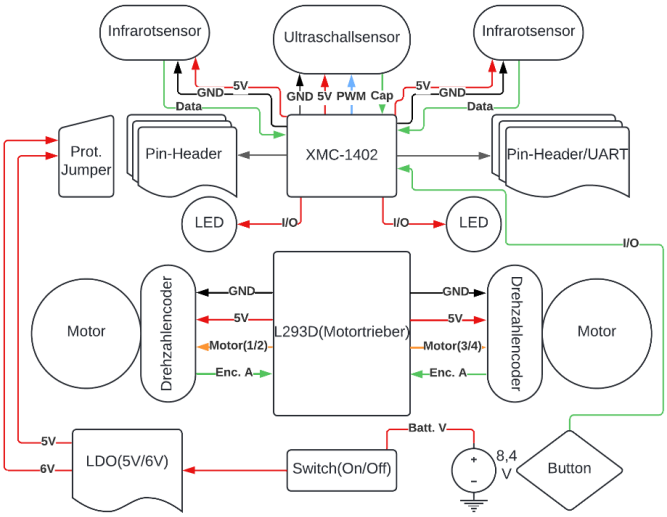
File > Fabrication Outputs > Gerbers(.gbr)

Unter dem Verzeichnis „Export-gerber“ sind die gerber Dateien von den ausgewählten Lagen zu finden. Dieses Verzeichnis soll dann auf die Webseite von JLPCB hochgeladen werden.



Die Kenngrößen wie die Maße und Anzahl an Lagen sind automatisch übernommen, allerdings sind weitere Konfiguration wie die Farbe, Platinen-dicke und Oberflächen-Material möglich. Da wir eine gemeinsame Bestellung für den Kurs gemacht haben, sind diese Weiteren Konfigurationen nicht von uns gewählt worden.

5 Blockdiagramme

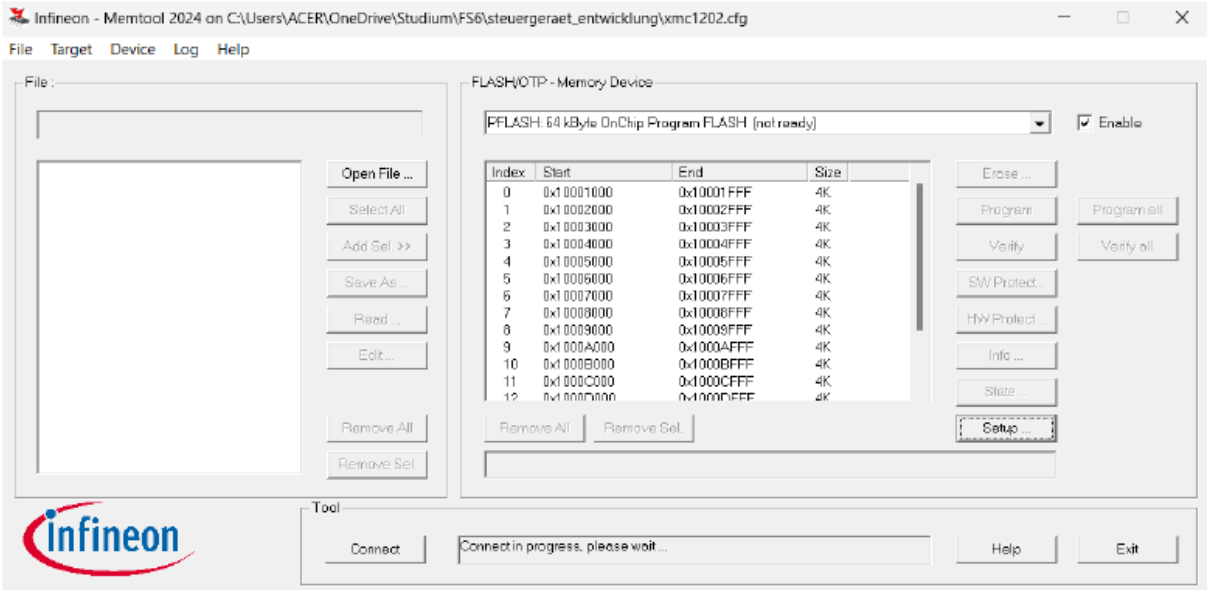


6 Inbetriebnahme und Tests

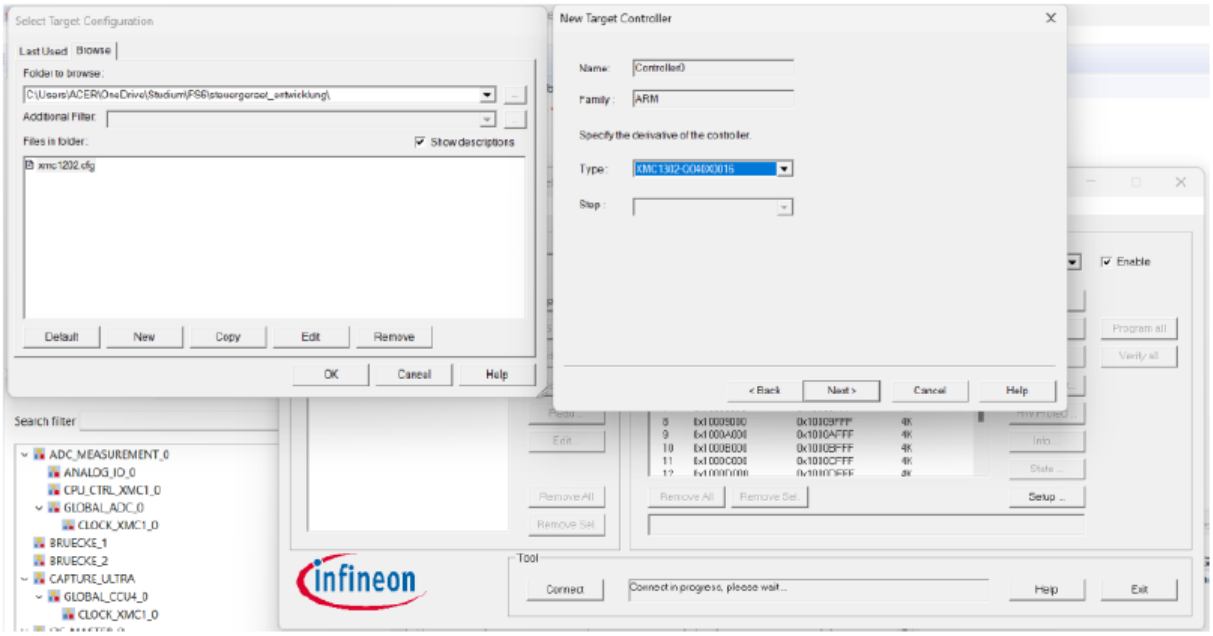
Für die Inbetriebnahme des Steuergeräts ist es wichtig zuallererst die Spannungsversorgung zu erstellen. Zunächst ist der LDO zu verlöten, danach wird die Batteriespannung am Eingang angelegt. Anschließend wird die geregelte Spannung von 5V an allen relevanten Stellen, wie am Ausgang, am Prozessor und an den Sensoranschlüssen, überprüft.

Nach der vollständigen Bestückung der Platine und erfolgreicher Prüfung der Spannungsversorgung kann der Prozessor in Betrieb genommen werden. Da die XMC-Prozessoren von Beginn an nicht programmierbar sind, muss zunächst der Boot Mode Index (BMI) geändert werden. Der erste Schritt hierbei ist die Überprüfung der Funktionalität des Prozessors über die serielle Schnittstelle, wofür ein USB-auf-UART-Adapter erforderlich ist.

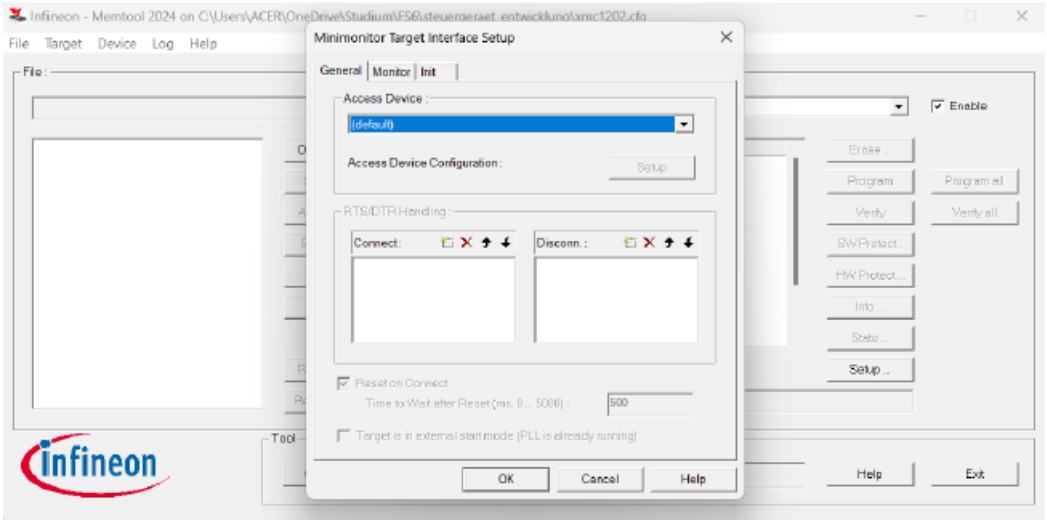
Der TX-Pin wird an Pin 0.14 und der RX-Pin an Pin 0.15 angeschlossen. Zudem sind die Pins für die Spannungsversorgung ordnungsgemäß zu verbinden. Für die Kommunikation kann ein serieller Terminal, wie beispielsweise hterm, verwendet werden. Der Prozessor sollte auf die Eingabe 0x006C mit der Antwort 0x5D reagieren, um die Funktionsfähigkeit zu bestätigen. Nach erfolgreicher Überprüfung kann der Boot Mode Index mit dem Infineon Memtool von „ASC_BSL“ auf „User Mode (Debug) SWD 0“ umkonfiguriert werden.



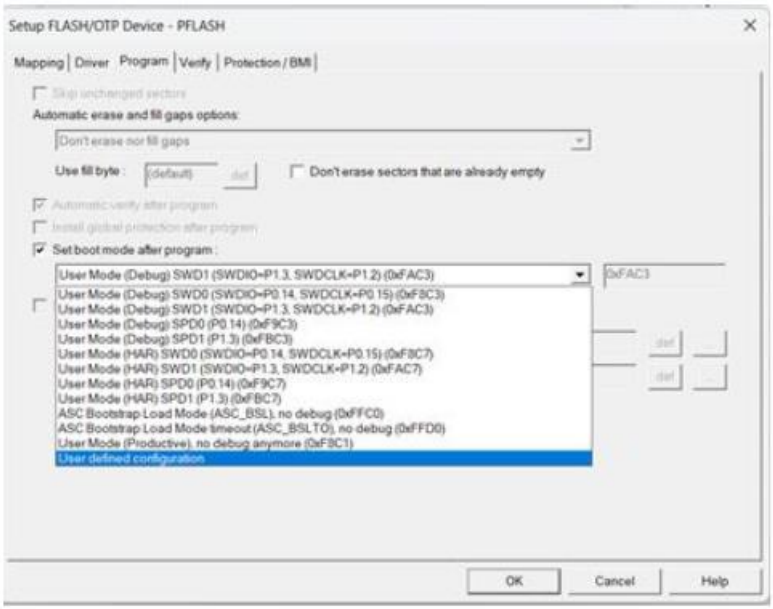
Zunächst ist es erforderlich, mit dieser Software eine .cfg-Konfigurationsdatei zu erstellen, die anschließend in den Prozessor geladen wird. Die Erstellung der Konfigurationsdatei erfolgt über das Menü: Target > Change > New > Create a new target configuration. Wählen Sie die ARM-Familie aus und suchen Sie den entsprechenden Prozessor. Achten Sie darauf, die korrekte Speichergröße auszuwählen.



Nach der Erstellung dieser Konfigurationsdatei soll nun eine Verbindung zu dem Prozessor hergestellt werden, dazu soll die richtige COM-Port unter Minimotor Target Interface Support ausgewählt werden, wichtig dabei ist die Baudrate von 19200 Baud.



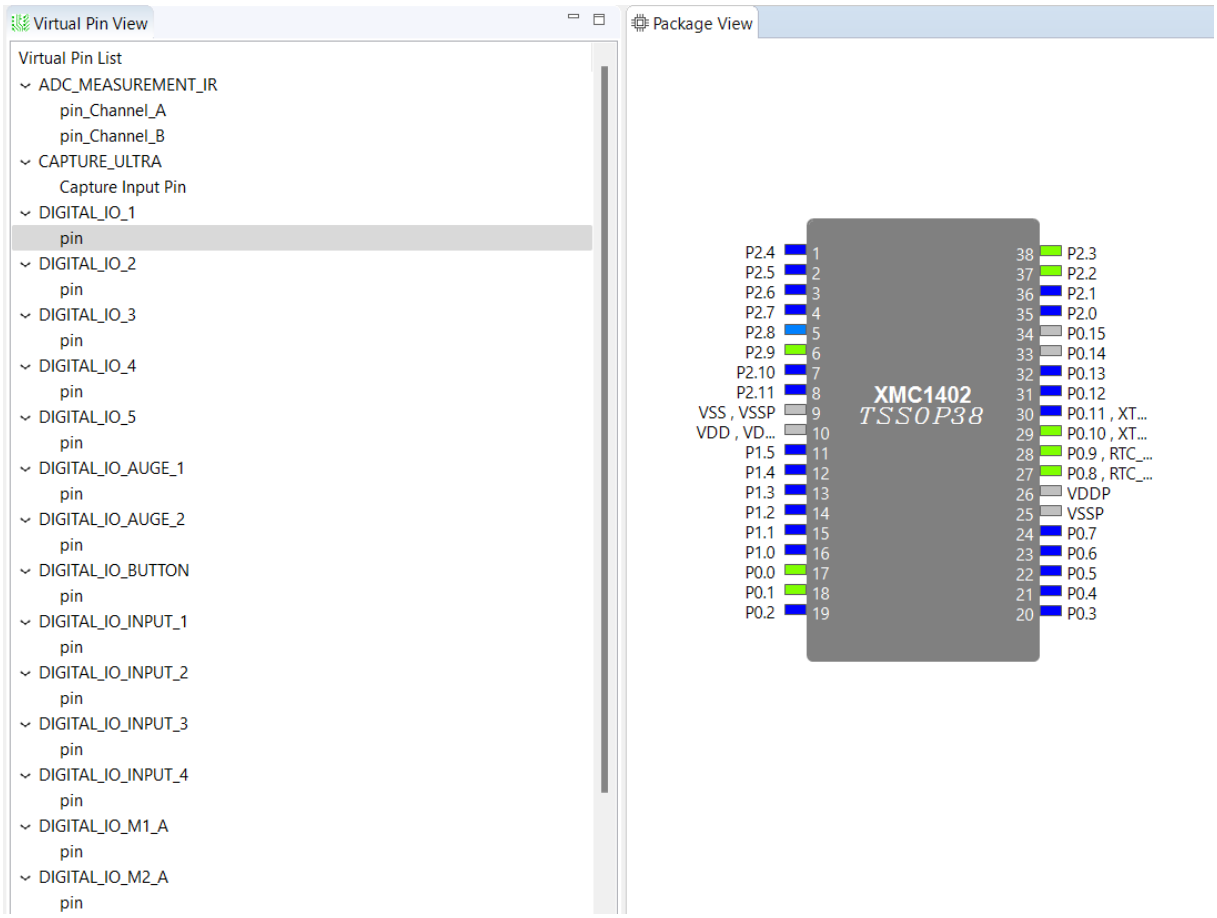
Nach erfolgreicher Verbindung zu dem Prozessor kann nun der Boot Mode Index auf „User Mode (Debug) SWD0“ geändert werden.



Nach Änderung des Boot Mode Indexes kann der Prozessor auf Dave wie ein Evaluation Kit programmiert werden und bei Abtrennen des Debuggers und korrekter Versorgung wird der Programmcode abgespielt.

6.1 Demo-Software

Um alle Funktionalitäten der MicroRat zu überprüfen haben wurde ein Demo-Programm erstellt, dass Programm liest Messwerten von den Sensoren und gibt die alle 100ms über die serielle Schnittstelle aus. Außerdem lassen Motoren des Boards individuell ansteuern.



Die Software umfasst verschiedene Aspekte und soll für Neubediener auskommentierbar sein. Unser Demo Programm soll grundsätzlich alle befindlichen Sensoren periodisch auslesen (100ms) und via UART an den Bediener senden, um die Daten auswerten zu können. Zudem soll auch demonstriert werden das die Räder individuell ansteuerbar sind.

Dummy Code:

```c
#include "DAVE.h"
#include "Dummy.h"

int button_status = 0;


int main(void)
{
    DAVE_Init();           /* Initialization of DAVE APPs  */

    while(true){
    button_status = DIGITAL_IO_GetInput(&DIGITAL_IO_BUTTON);
        if(button_status == 1){
            Dummycode_Init();
            button_status = 0;
        }
    }
}
```

```c
#include "Dave.h"
#include "Dummy.h"
#include <stdio.h>

#define PERIODIC_READ 100000U

uint8_t UART_String[100];
uint32_t Timer_100ms;
uint32_t Capture_t;

XMC_VADC_RESULT_SIZE_t ADC_Wert_IR_L;
XMC_VADC_RESULT_SIZE_t ADC_Wert_IR_R;

float captured_time_us;
float distanz_ultra;
int IR_L, IR_R;

void Sensoren_Auslesen_100ms(void){
    //IR
    ADC_Wert_IR_R = ADC_MEASUREMENT_GetResult(&ADC_MEASUREMENT_Channel_A);
    ADC_Wert_IR_L = ADC_MEASUREMENT_GetResult(&ADC_MEASUREMENT_Channel_B);
    IR_R = ADC_Wert_IR_R;
    IR_L = ADC_Wert_IR_L;
    //ULTRASCHALL
    CAPTURE_GetCapturedTime(&CAPTURE_ULTRA, &Capture_t);
    captured_time_us = ((float)Capture_t * 333.33)/1000;
    distanz_ultra = captured_time_us /58;
    //UART-Transmit
    sprintf((char*)UART_String, " Ultraschall: %.2fcm IR_R: %d IR_L: %d\n\r", distanz_ultra, IR_R, IR_L);
    UART_Transmit(&UART_COM, UART_String, sizeof(UART_String));

    DIGITAL_IO_ToggleOutput(&DIGITAL_IO_AUGE_1);
    DIGITAL_IO_ToggleOutput(&DIGITAL_IO_AUGE_2);
}

void Dummycode_Init(void)
{
    Timer_100ms = SYSTIMER_CreateTimer(PERIODIC_READ,SYSTIMER_MODE_PERIODIC,(void*)Sensoren_Auslesen_100ms,NULL);
    SYSTIMER_StartTimer(Timer_100ms);

    DIGITAL_IO_SetOutputHigh(&DIGITAL_IO_INPUT_1);
    DIGITAL_IO_SetOutputLow(&DIGITAL_IO_INPUT_2);
    DIGITAL_IO_SetOutputLow(&DIGITAL_IO_INPUT_3);
    DIGITAL_IO_SetOutputHigh(&DIGITAL_IO_INPUT_4);

    PWM_Start(&PWM_1);
    PWM_Start(&PWM_2);
}
```

Der hier dargestellte Dummycode funktioniert folgendermaßen:

Wenn der Knopf auf der MicroRat gedrückt wird, aktiviert die H-Brücke die Motoren, sodass die MicroRat mit einem Duty-Cycle von 40 % vorwärtsfährt. Gleichzeitig startet ein Timer, der alle 100 ms die ADC-Werte der Infrarotsensoren und die Entfernungswerte des Ultraschallsensors erfasst. Diese Daten werden gespeichert und in eine formatierte Zeichenkette

umgewandelt, die über UART an den Bediener gesendet wird. So erhalten wir alle 100 ms aktuelle Informationen über die Umgebung und die Fahrzeugbewegung. Als visuellen Feedback werden auch die zwei LEDs periodisch an und ausgeschaltet.

7 Messung und Auswertung

UART-Bilder machen für Beweis. PWM nochmal mit Oszi messen für Ultraschall, Motortreiber usw.

Drehzahlencoder per uart ausgeben, dass hochgezählt wird

8 Schlussfolgerung

Zusammenfassend lässt sich feststellen, dass das MicroRat-Projekt erfolgreich die Grundlagen für die Entwicklung einer innovativen Hardware geschaffen hat, die an einer Micromouse orientiert ist. Im Rahmen des Faches Steuergeräteentwicklung wurden wesentliche technische Aspekte, wie die Systemarchitektur, Kommunikationsprotokolle und die Integration von Steuerungssystemen, umfassend behandelt.
Die Erkenntnisse aus diesem Projekt bieten zukünftigen Studenten wertvolle Einblicke in die praktischen Herausforderungen und Lösungsansätze der Robotik.
Darüber hinaus legt das Projekt den Grundstein für weiterführende Entwicklungen, sei es in Form von Optimierungen, Erweiterungen oder neuen Anwendungsbereichen. Die dokumentierten Ergebnisse und Erfahrungen werden als wertvolle Ressource dienen, um die Innovationskraft im Bereich der Embedded Systems und der autonomen Systeme weiter voranzutreiben. Insgesamt stellt das MicroRat-Projekt eine bedeutende Lehr- und Lernplattform dar, die zukünftige Studienjahrgängen hoffentlich inspirieren und unterstützen wird.

9 Verbesserungsvorschläge

Nach Beendigung unseres Projektes sind uns während und nach der Bearbeitungszeit ein paar Fehler bzw. Verbesserungen aufgefallen, die wir hier nochmal auflisten wollten. Jeder angesprochene Fehler ist sowohl in der Excel Teilliste sowie in dem Schaltplan und Layout bereits behoben.

Debug Stecker: Während der Bestückung unserer Platine fiel uns auf, dass wir den ein zu kleinen Footprint des Debug Steckers verwendet hatten. Dieser hat die Maße …. Und sollte die Maße … haben. Der Footprint sollte bei einer Verbesserten Version überprüft und geändert werden, sodass man problemlos ein Pin-Header auflöten kann.

Motorauswahl: Da wir für die Erfassung der Geschwindigkeit einen Drehzahlencoder verbauen wollen, ist die Auswahl eines passenden Motors wichtig. Hier wird bei unseren Micro Metal Gearmotor zwischen einem Motor mit und ohne erweiterter Motorwelle unterschieden. Wir haben Motoren ohne jene Erweiterung bestellt, sodass das Anbringen des Drehzahlencoder nicht möglich war.

XMC Pin Belegung: Hier ist nochmal wichtig zu erwähnen, dass die Pin-Belegung des Microcontrollers gründlichst durchdacht und überprüft werden muss, da eine spätere Änderung der Leiterbahnen mit z.B. Kupferdraht sehr schwer bis gar nicht möglich ist. In unserem Falle haben wir ein paar Digital I/O’s als Input deklariert, obwohl sie ein Input/Output sein sollten. Dass hatte zur Folge, dass einige Pins nicht die vorgesehene Funktion ausüben können, da der bereits verwendete Pin nicht beide Optionen zulässt. So waren wir gezwungen, einen freien Pin mit unserem ursprünglich gedachten Output zu verbinden, um die Funktion aufrecht zu erhalten.