OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

Balancing Plate

An der Fakultät für Informatik und Mathematik der
Ostbayerischen Technischen Hochschule Regensburg
im Studiengang
Technische Informatik

eingereichte

Projektdokumentation (Datenverarbeitung in der Technik)

Name: Michael Braun
Matrikelnummer: 3113161
Name: Korbinian Federholzner
Matrikelnummer: 3114621
Name: Philipp Kramer
Matrikelnummer: 1234456
Name: Patrick Lesch
Matrikelnummer: 12344556
Name: Michael Schmidt
Matrikelnummer: 2907322

Erstgutachter: Prof. Dr. Richard Roth
Zweitgutachter: Hr. Matthias Altmann

Abgabedatum: 10.2.2020

Inhaltsverzeichnis

1. Einführung/ Projektkontext

Der Grundaufbau des Prototyps ist eine Plexiglasplatte, auf der sich eine Kugel befindet. Diese Kugel wird durch Druck auf eine resistive Touch-Folie (Wertbestimmung durch Druck) erkannt und die zugehörigen Daten werden an einen Raspberry Pi 4 übermittelt. Dieser berechnet dann den Offset zum gewünschten Stellpunkt auf der Platte und mit Hilfe eines PID-Reglers die neuen Stellwerte für die Servomotoren. Diese Daten werden an einen XMC 4500 übermittelt, welcher die zuvor ermittelten Werte in PWM-Signale übersetzt und diese dann an die zuständigen Servomotoren übergibt. Diese bewegen die Plexiglasplatte, auf der sich die Touch-Folie befindet, um so die Kugel zu balancieren. Die Versuchsapparatur soll selbstständig in der Lage sein eine Kugel auf einer Platte in eine vorher festgelegte Position, meist die Mitte der Platte, zu manövrieren. Dies soll durch Heben bzw. Senken der Plexiglasplatte mittels dreier Servomotoren geschehen. Optional ist dabei die manuelle Steuerung der Platte mittels einer App, welche über eine WiFi-Schnittstelle mit dem Raspberry Pi in Verbindung stehen soll. Diese App soll dabei das im Handy verbaute Gyroskop verwenden, um den Neigungswinkel des Geräts zu erfassen und die Platte dementsprechend zu neigen. Auch eine GUI sollte in der App vorhanden sein.
Im Gegensatz zum Lastenheft, dass kurz nach dem Start abgegeben werden musste, haben sich im Laufe des Projektes einige Änderungen ergeben. Dabei wurde die Steuerung mittels einer iOS-App nunmehr als Pflichtmodul angesehen. Außerdem wurde das Auslesen der Potentiometer der Servomotoren aus Zeitgründen fallengelassen.

2. Umsetzung

Um eine konsistente Abarbeitung zu gewährleisten wurde das Projekt in verschieden Module/Arbeitspakete geteilt.

Name	Aufgabe 1	Aufgabe 2	Aufgabe 3
Michael Braun	Physischer Aufbau	Reglerentwicklung	PWM-Signale
Korbinian Federholzner	Ansteuerung mit App	UART Schnittstelle	Touch-Folie
Philipp Kramer	Reglerentwicklung	Modellerstellung Matlab	Touch-Folie
Patrick Lesch	Ansteuerung mit App	WiFi Schnittstelle	Custom Kernel
Michael Schmidt	Physischer Aufbau	UART Schnittstelle	PWM-Signale

Diese werden immer von mindestens zwei Projektteilnehmern abgearbeitet. Außerdem wurde auf eine in sich geschlossene Aufteilung geachtet, um bei aneinandergrenzenden Modulen immer einen Ansprechpartner zu haben. Um durch eine gegenseitige Kontrolle eine funktionierende Version sicherzustellen und um eine Qualitätskontrolle durchzuführen, wurden in dem Git-Repository die Funktion push-Requests aktiviert.

2.1. Physicher Aufbau

Von Michael Braun und Michael Schmidt

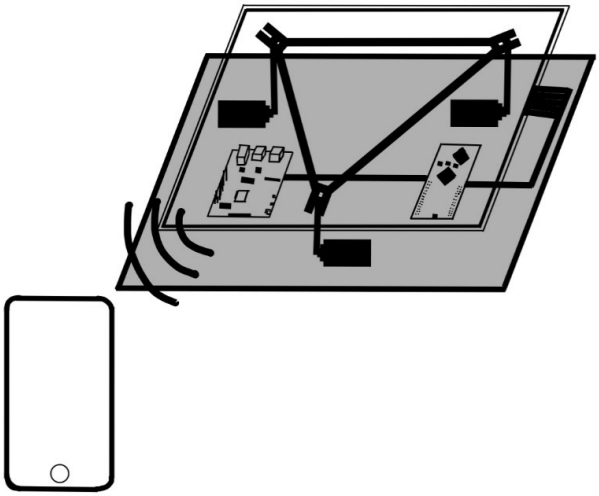

Abbildung 1: Skizzierter Aufbau des Prototypen

In 1 ist eine Skizze des physischen Aufbaus aus dem Lastenheft zum Start des Projektes zu sehen. Dabei wurde lediglich graphisch eine mögliche Vorversion des späteren Prototyps gezeichnet. Im Gegensatz zu 1 wurde eine sechseckige Grundplatte mit

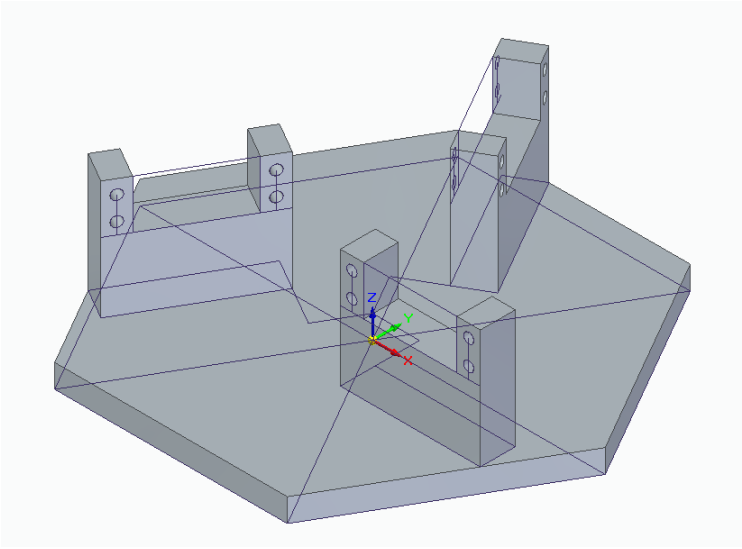

Abbildung 2: Grundplatte des Prototypen

den Halterungen für die drei Servomotoren mittels Solid Edge 2020 konstruiert und mittels eines 3D-Druckers gefertigt. Diese ist in 2 zu sehen

Der Druck der Grundplatte dauerte dabei etwa zehn Stunden, was unter anderem an den Ausmaßen der Platte (Durchmesser von 20 cm) liegt. Nachdem die Servomotoren in den Halterungen befestigt wurden, hat man die Servo-Hebel und Gabelköpfen verbunden, dazu wurden die Servo-Hebel aufgebohrt. Nach dem Zusammenbau hat man festgestellt, dass zwischen den Hebeln und dem Gabelkopf noch zu viel Spielraum war, der die Regelung beeinträchtigen könnte. Deshalb wurden Unterlegescheiben zur Verringerung des Spiels eingesetzt. Die Verbindung zur Platte wird über ein Dreieckskonstrukt 3, das ebenfalls aus dem 3D-Drucker kam, hergestellt. Da-

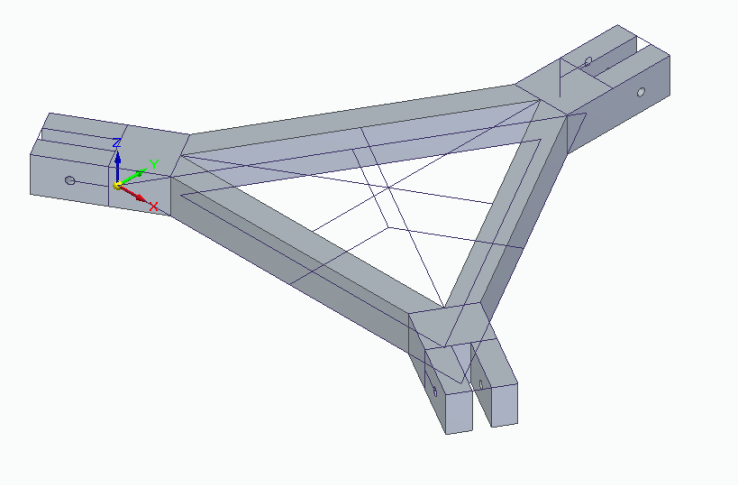

Abbildung 3: Dreiecksverbindung zur Touch-Folie

bei war vor allem die korrekte Länge zum Mittelpunkt des Dreiecks, respektive der Grundplatte, eine wichtige Größe, da eine falsche Länge nicht mehr die senkrechte Ausrichtung des Gabelkopfes zu der Grundplatte zur Folge gehabt hätte. Außerdem mussten die Kugellager passend in den Gabelkopf der Dreiecksverbindung eingefügt werden, um eine seitliche Drehung dieser zu verhindern. Der 3D-Druck dieser Verbindung dauerte in etwa sechs Stunden. Die Konstruktion dieser beiden Bauteile nahm in etwa fünfzehn Stunden in Anspruch. Diese Dreieckskonstruktion liegt genau über dem markierten Dreieck der Grundplatte. Die Touch-Folie befindet sich dabei auf einer Plexiglasplatte, welche mittels Silikon mit der Dreiecksverbindung zusammengefügt wurde. Außerdem wurden vor dem Zusammenbau der Servomotoren mit den Servo-Hebeln, alle Servomotoren in die Neutralstellung gesetzt (vgl. PWM-Signale), um die Regelung ebenfalls von einem neutralen Punkt aus zu starten. Dabei traten motorspezifische Unterschiede auf die mittels der PWM-Werte ausgeglichen werden müssen.Bei der Verkabelung wurden sowohl der XMC und die drei Servomotoren mittels eines gelöteten Teilstücks an die gleiche Masse gelegt. Auch die Betriebsspannung der Motoren wurde so gehandhabt. Zum Fazit: Die Tei-

le aus dem 3D-Drucker waren insgesamt zufriedenstellend, lediglich die Löcher für die Schrauben an der Dreiecksverbindung mussten nachträglich aufgebohrt werden und an einem Gabelkopf der Dreiecksverbindung blieben Teile an der Grundplatte des Druckers hängen. Glücklicherweise waren die Servomotoren ausreichend stark um die Platte ohne Probleme zu bewegen.

2.2. Software-Architektur

2.3. Touch-Folie

Von Korbinian Federholzner und Philipp Kramer

2.3.1. Funktion der Touch Folie

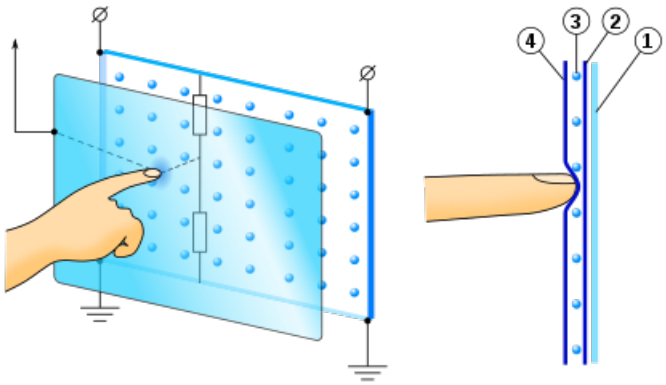

Abbildung 4: Funktion des Touch Panels

Zum Auslesen der Sensorwerte wird ein 5-Wire Rezessives Touchpanael verwendet. Die Variante mit 5 Kabeln hat den Vorteil, dass für das Auslesen ein 5tes Kabel verwendet wird, welches ein Analoges Signal liefert. Prinzipiell ist das Panel ein Spannungsteiler, bei 4 sieht man dass es zwei Schichten hat, welche durch Druck zusammengefügt werden, dadurch ergibt sich immer ein anderer Wiederstand je nach dem auf welcher Position man sich auf dem Panel befindet. Der Analoge Spannungswert ist also durch diesen Spannungsteiler bestimmt. Damit Sich die Beiden Folien berühren können muss ein Mindestruck von $1.00 Newton$ gegeben sein. Im Bild 4 sieht man dass auf der Oberen Seite Spannung und der unteren Masse anliegt. Mit dieser Konstellation kann man nur eine Richtung messen also z.B. die X-Koordinate. Um nun auch die Y-Koordinate messen zu können müssen die Pins so umgeschalten werden, dass Auf der Linken Seite Spannung und auf der Rechten

Seite Masse anliegt. Dadurch ändert wird der Wiederstand der anderen Seite nun als Spannungsteiler gemessen.

2.3.2. Implementierung

Dadurch, dass die Pins umgeschaltet werden müssen, wurden die GPIO Pins des XMC verwendet. Da das Panel nur mit einer Frequenz von 50 Hz geschalten werden kann, wurde ein Timer Interrupt definiert. Dieses Interrupt wurde alle 20 ms Ausgelöst. Die Interrupt Service Routine startet dann die Messung für den Analog Digital Converter. Der Analoge Pin des Panels ist an einem Analog fähigen Pin des XMCs angeschlossen, von welchem mittels des ADCs gemessen wird. Der ADC löst sobald er mit dem Messen fertig ist auch eine Interrupt Service Routine aus, in dieser wird der gemessene Wert gespeichert und dann die GPIO Pins umgeschalten damit die andere Koordinate gemessen werden kann. Sind beide Werte vorhanden, so werden die beiden Daten in einem UART Frame verpackt und dann ein UART-Transmit ausgelöst.

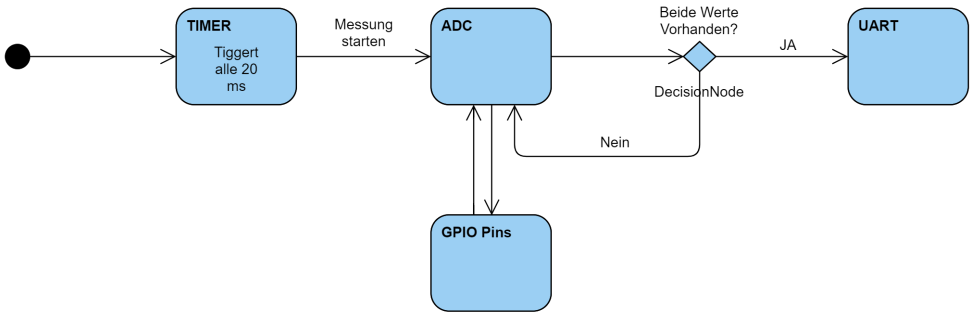

Abbildung 5: UML Activity Diagramm, der Interrupts

2.3.3. Probleme

Ein Problem im Zuge der Implementierung der Touch-Platte, war die Funktion der ADC-App im Dave4, da bei diesem keine gute Dokumentation vorhanden war, so war es z.B. schwer herrauszufinden wie die Funktion zum starten der ADC Messung hieß. Ein görßeres Problem war der Messfehler des Touch Panels, bei welchem die Platte einen Spannungsabfall je nach Pin hatte. Bsp: Falls man der X-Achse entlang messen wollte und dabei einen Konstaten Y-Wert beibehielt, so wäre das Erwartete Verhalten, dass der X-Wert gleich bleiben würde. Jedoch stieg der Wert bei messen entlag der X-Achse konstant an, das selbe war auch auf der Y-Achse. Um das dieses verhalten der Pins zu eleminieren, wurde versucht das Ganze mittels einer H-Brücke

zu lösen, jedoch war hier das selbe Problem, über der H-Brücke fiel auch Spannung ab, da 2 Pins immer fest waren und nur 2 Pins über die Brücke geschalten waren, gab es wieder ein ähnliches Szenario. Die nächste Überlegung, war die Verwendung von Relais. Leider sind Relais die schnell genug Schalten können relativ Teuer und diese haben einen höheren Verschleiß. Die Lösung des Problems war, Wiederstände zwischen die Pins des XMCs und der Platte zu schalten. Diese Wiederstände minimieren den Fehler, dadruch war der Fehler vernachlässigbar und konnte durch den Regeler ausgeglichen werden.

2.4. Programmierung Raspberry Pi

2.5. UART Schnittstelle

Von Korbinian Federholzner und Michael Schmidt

2.5.1. Funktion

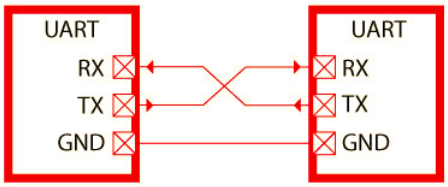

Abbildung 6: Grundaufbau der UART-Verbindung
bilder

Zur Datenübertragung zwischen dem Raspberry Pi und dem XMC Mikrocontroller wurde das Universal Asynchronous Receiver Transmitter-Protokoll (UART) verwendet. Dieses ist für schnelle Bitübertragungsraten, in unserem Fall eine Baud-Rate von 19200 (Bits pro Sekunde), über eine einzelne Leitung verantwortlich. In unserem Fall ist die Übertragung im Modus Full-Duplex und erfordert somit zwei Leitungen, die sich kreuzen 6.

Grundsätzlich erfolgt die Datenübertragung aufgrund der Einfachheit in Strings, welche jeweils nach dem Empfang dekodiert werden. Dabei enthält ein UART-Paket genau ein Zeichen der Zeichenfolge. Ein solches Beispielpaket ist in 7 zu sehen.Die Übertragung erfolgt dabei auf dem Raspberry Pi über die General Purpose In/Out Pins 14 und 15. Dabei ist der Pin 14 der Sender und der Pin 15 der Empfänger von den Nachrichten des XMC. Auf dem XMC ist die Pinbelegung variabel und somit

flexibler im Gegensatz zum Raspberry Pi. Hier wurden die beiden Pins 1.4 und 1.5 gewählt. Hier fungiert Pin 1.5 als Sender und Pin 1.4 als Empfänger. Zum Testen wurde vom Raspberry Pi ein vordefinierter String, in diesem Fall „$123456789!",, an den XMC gesendet. Dieser liest die ankommenden Daten und sendet sie zurück an den Raspberry Pi, welcher sie auf dem Bildschirm ausgibt. Dadurch ließen sich die nachfolgenden Fehler in der Implementierung und Datenverarbeitung des XMC besser nachvollziehen. Am Anfang des Projektes traten diverse Schwierigkeiten beim Testen der Implementierung auf. Zum Beispiel war das Flashen des Programmes auf den XMC unmöglich, da ein Fehler generiert wurde, da keine Verbindung zu dem auf dem XMC integrierten Debugger hergestellt werden konnte. Nach einem Test mit einem anderem XMC gleicher Bauart wurde festgestellt, dass unser erster XMC lediglich defekt war. Nach einem Tausch des verwendeten XMC konnte die Programmierung wieder fortfahren, allerdings wurden durch diesen Fehler insgesamt ca. 10 Stunden Arbeitszeit in die Fehlersuche investiert. Dieses Problem behinderte auch das Arbeitspaket „PWM-Signale". Ein weiteres Problem, auf das wir gestoßen sind, war ein Fehler in der Interrupt-Funktion. Auf dem XMC lassen sich unter Dave4 in den zur Verfügung gestellten APPs diverse Empfangs- und Sendeoptionen festlegen. Darunter war auch das Empfangen mittels eines Interrupts. Dieses Interrupt wurde jedoch beim Testen nicht aufgerufen und somit die ankommenden Daten nicht aus dem Speicher ausgelesen. Um dieses Problem zu umgehen wurde mittels der Dave4-APP NVIC/Interrupt ein Hardwareinterrupt konfiguriert, welches bei einer Änderung des Buses von logisch 0 auf logisch 1 ausgelöst wird und die ankommenden Daten verarbeitet. Dabei traten jedoch weitere Probleme auf, wie z.B. die ankommenden Daten waren nicht vollständig, da das Interrupt zu schnell ausgelöst wurde und die verbleibenden Daten nicht in den Buffer geschrieben wurden, oder das erste Zeichen wurden als „0" interpretiert. Dies hatte folgenden Grund: Das Interrupt wurde zu langsam geöffnet und im Buffer kam es zu einem Overflow, sodass die ersten Werte bereits wieder mit „0" überschrieben wurde, welche das UART-Protokoll als Byte-Stream-Ende interpretiert und somit nicht korrekte Daten liefert. Nach intensiver Suche in der integrierten Dokumentation in Dave 4, welche jedoch erst nach einigem Suchen gefunden wurde, stießen wir auf eine Funktion: „UART_StartRecieveIRQ" , deren Implementierung anscheinend erforderlich war, um mittels der in der UART-APP vorprogrammierten Interrupt-Option die ankommenden Daten auszulesen . Diese Funktion setzt einen Interrupt-Request und muss kontinuierlich nach dem erstmaligen Empfangen ausgeführt werden, um eine fehlerfreie Abarbeitung zu gewährleisten. Somit wird sie nach jedem Empfangsvorgang auf dem XMC in der Interrupt-Routine neu aufgerufen.

2.5.2. Datenübertragungsformate

Von Korbinian Federholzner und Michael Schmidt
Um eine eindeutige Übertragung zu gewährleisten, wurde sich auf ein standardisiertes Format festgelegt. Dabei wurden eine bestimmte Anzahl Pakete in einer bestimmten Reihenfolge gesendet, wobei pro Paket immer ein Zeichen des Strings übertragen wird, wie in 7verdeutlicht wird. Diese Zeichen werden dann aneinander-

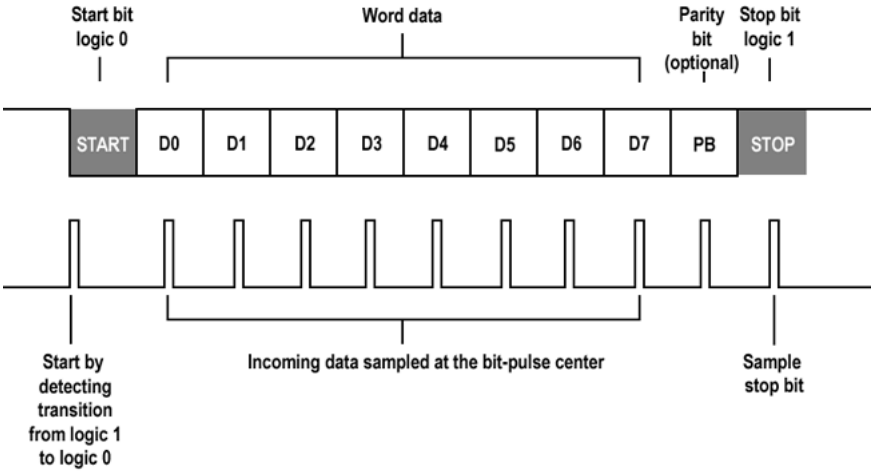

Abbildung 7: Aufbau eines UART-Frames[**?**]

gereiht und ergeben die unten beschriebenen Formate. Dabei hat die Kommunikation, bei der der Raspberry Pi der Sender und der XMC 4500 der Empfänger ist folgendes Format 8. Übertragen werden hierbei die aktuellen Werte für die PWM-

Byte 0 Startbyte: 0x24	Datenbyte 1-3	Datenbyte 4-6	Datenbyte 7-9	Byte 10 Stoppbyte: 0x21

Abbildung 8: Aufbau eines Datenpakets mit Servowerten

Signalgenerierung, deren Werte sich zwischen 450 und 1050 befinden. Dies ist durch die maximale Auslenkung der Servomotoren begründet. Um weitere Bytes zu sparen wird nur der Offset von dem Grundwert 450 übermittelt. Dieser wird in einer nachfolgenden Funktion wieder dekodiert und in die jeweiligen Variablen gespeichert, auf die die PWM-Signalgenerierung zugreift. Dabei befinden sich in den ersten drei Zeichen des Datenteils (Byte 1-3) die Werte für den ersten Servomotor., in den zweiten drei (Byte 4-6) die Werte für den zweiten Servomotor und in den letzten drei

(Byte 6-9) die Werte für den dritten Servomotor, wie in der Abbildung 6 dargestellt. Das Startbyte 0, „0x24" und das Stoppbyte 10 „0x21" dienen dabei der Erkennung der feststehenden Feldlänge, um eine korrekte und vollständige Übermittlung zu gewährleisten. Die Kommunikation zwischen XMC 4500 und dem Raspberry Pi erfolgt über mehrere Formate, da wir mehrere unterschiedliche Nachrichten auf dem Raspberry Pi verarbeiten müssen. Zum einen wären da die Übertragung der X- und Y-Koordinaten der Touch-Folie und zum anderen die Rückgabewerte der Potentiometer aus den Servomotoren. Um zwischen den beiden Nachrichtentypen, die unterschiedliche Längen und Formate haben zu unterscheiden haben wir eine Nachrichten-Identifikationsnummer (Message-ID) eingeführt, welche vor den jeweiligen Dekodierungsfunktionen überprüft wird. Lediglich der Datenteil unterscheidet sich zwischen den beiden Typen. Da sie als String übertragen werden, ist das System problemlos auf bis zu 100 unterschiedliche Nachrichtenformate erweiterbar. Um das bestehende System zu erweitern muss lediglich eine zugehörige Dekodierungsfunktion auf dem Raspberry Pi programmiert werden und ein korrekter Sendevorgang auf dem XMC sichergestellt werden. Die Nachricht mit den Werten der Touch-Folie hat die Message-ID „00" und wird wie folgt übertragen 9. Auch hier sind Start

Byte 0 Startbyte: 0x24	Byte 1-2 Message-ID	Byte 3 Trennbyte: 0x7C	Datenbyte 4-7	Byte 8 Trennbyte 0x20	Datenbyte 9-12	Byte 13 Stoppbyte: 0x21

Abbildung 9: Aufbau eines Datenpakets mit den Werten der Touch-Folie

und Ende der Nachricht wieder mit einem Zeichen, zum Erkennen der korrekten Feldlänge, markiert. Dabei werden die Werte der Touch-Folie, die bereits in X-, bzw. Y-Koordinaten umgerechnet wurden übertragen. Diese sind in den Datenbytes 4-7 (X-Koordinate) und Datenbytes 9-12 (Y-Koordinate) kodiert. Zwischen den einzelnen Koordinatenwerten und der Message-ID wurde noch ein Trennbyte eingeführt, da die Koordinaten unterschiedliche Länge haben können (3 oder 4 Zeichen). Auf dem Raspberry Pi werden diese wieder in integer-Werte umgerechnet und als Koordinaten an die Regelung übergeben.
Allerdings kamen die Formate mit Message-IDs nicht mehr zum Einsatz, da das Auslesen der Servowerte zeitlich nicht mehr umgesetzt werden konnte. Daher fallen diese Teilstücke und das nachfolgende Trennbyte aus dem Format für die Übertragung der Touch-Plattenwerte heraus. Der Code ist dennoch im Git-Repository vorhanden und diese Leitung kann dadurch erweitert werden um multiple Datenformate zu unterstützen.

2.5.3. CRC-Check

Da der Austausch der mit UART immer wieder Bit-Errors hatte, gab es immer wieder Störungen an den Servos. Um die Bitfehler zu erkennen wurde ein CRC-Check implementiert, welcher ein Generatorpolynom über das gesamte Packet ermittelt. Für die Implementierung wurde eine CRC-Tabelle benutzt, da diese den Vorteil hat, dass der CRC-Betrag nicht komplett berechnet werden muss, sondern nur ein Teil. Über diesen Anteil kann der Wert in der Tabelle nachgeschlagen werden und dieser als CRC-Wert verwendet werden. Der Vorteil einer solchen Tabelle ist die Performance, da die komplette Rechnung aufwändiger wäre und mehr Performance ziehen würde. Der CRC-Check ist momemtan nur in der Richtung vom XMC zum Raspberry Pi implementiert, da nur dort die Bit-Errors auftraten. Vom Touchpanel des XMCs zum PI wurde nicht unbedingt ein CRC benötigt, da auf dem PI ein Kalman-Filter davorgeschalten wurde der diese Fehler ausgleicht. Der CRC könnte hier in Zukunft noch implementiert werden.

2.5.4. Probleme

Die größten Probleme beim UART war die Dave4 Dokumentation des XMCs. Online findet man ein sehr detailiertes Handbuch zum XMC4500, jedoch nahezu keine Dokumentation wie die enthaltenen Dave4 Apps funktionieren. Die OTH Rechner haben bei der Dave4 IDE keine Help Manuals, was das Ganze noch erschwert. Die Lösung dieses Problems war, die Help Manuals auf Dave4 manuell nachzuinstallieren. In diesem git es sehr viele nützliche Informationen wie UART eigentlich mittels der App funktioniert und welche Bedeutung die Funktionen genau haben.

2.5.5. Ausbick

Folgende Punkte waren aufgrund von Zeitmangel nicht mehr funktionsfähig zu implementieren. Der CRC-Check ist momentan aus zeitlichen Gründen in nur eine Richtung implementiert. Es ist zwar nicht wegen des Kalman-Filters nicht unbedingt nötig,aber es wäre trozdem nicht schlecht in Richtung vom XMC zum PI einen CRC-Check zu haben. Auf dem Branch XMC/HDLCframing ist eine Version des UART Frames die sich an dem HDLC Format orientiert. Diese ist warscheinlich stabiler als die momentane. Der Code auf dem Branch ist so gut wie fertig, jedoch war das Framing welches in der Endversion ist, schon gut genug getestet, sodass dieser ungetestet Code nicht nötig war.

2.6. Reglerentwicklung

Von Michael Braun und Philipp Kramer

2.7. PWM-Signale

Von Michael Braun und Michael Schmidt

2.7.1. Algemeines

Die Ansteuerung der Servomotoren erfolgt über ein 50 Hz PWM-Signal, welches mittels der Funktion „PWM_SetDutyCycle" auf die jeweils aktuellen Werte geändert wird. Diese Funktion wandelt automatisch die Werte die im Promillebereich, im Format $0 - 10000$, angegeben werden, in die richtige Länge des Hochanteils des PWM-Signals um. Diese Funktionen werden von den Dave 4- eigenen APPs zu Verfügung gestellt. Dabei sind durch die Servomotoren einige Werte vorgegeben:

Bezeichnung	Wert in ms (zu 20ms/50Hz)	Wert in Prozent	Wert in Promille
Minimalstellung	0,9	4,5	450
Neutralstellung	1,5	7,5	750
Maximalstellung	2,1	10,5	1050

Sollten die übertragenen Werte den Maximalwert von 1050 überschreiten, werden sie lediglich auf den Maximalwert gesetzt, um eine korrekte Ansteuerung der Servomotoren zu gewährleisten. Diese Werte werden kontinuierlich von der Regelung verändert und über die Pins 3.0, 3.3 und 3.4 des XMC an die Servomotoren ausgegeben. Dies wird in einem Interrupt abgearbeitet, das im 50 Hz Takt aufgerufen wird. Dabei sind jedoch noch motorenspezifische Unterschiede aufgetreten, die in Tabelle 3 aufgeführt werde. Diese sind nötig, um die Platte initial in eine ebene Nullstellung zu positionieren.

Servomotor	Offset
1	35
2	15
3	5

2.7.2. Probleme und Ausblick

Jedoch wird die Auslenkung der Hebel durch die Aufhängung der Motoren begrenzt, da ab einem bestimmten Wert die Platte an diese stößt. Dadurch ergeben sich folgende Einschränkungen:

Servomotor	Minimalwert	Maximalwert
1	600	1050
2	490	950
3	490	950

Der Maximalwert ergibt sich dabei aus dem Umstand, dass die Hebel in der Verbindung zwischen Motor und Dreiecksverbindung überdrehen und dadurch die Platte eher absenken als sie auf ihre maximale Auslenkung zu bringen. Dieses Problem

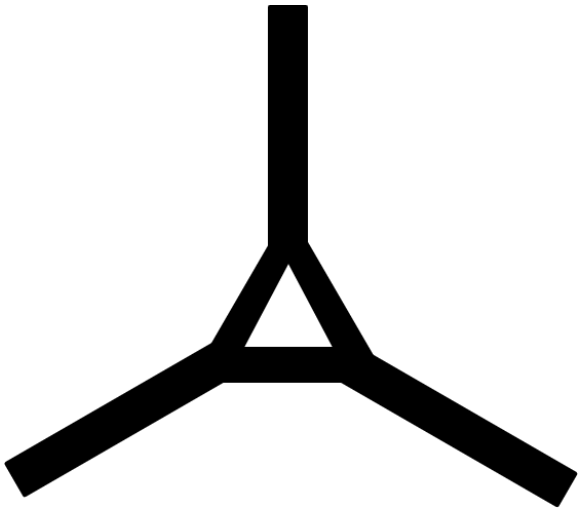

Abbildung 10: Aufbau einer alternativen Dreiecksverbindung mit größerem Wirkungsradius

könnte dabei in Zukunft mittels einer neuen Dreiecksverbindung gelöst werden, die auch niedrigere Auslenkungen erlaubt 10. Leider kamen wir aufgrund von Zeitdruck nicht mehr dazu diese neue Verbindung zu drucken. Allerdings wäre auch noch die Befestigung an der Platte ein Problem, da die alte Verbindung bereits mittels Silikon befestigt und eine Beschädigung beim Ablösen dieser zu befürchten war. Glücklicherweise hat der Regler solche extremen Ausschläge auch gar nicht zu handhaben, da die Regelung für unsere Zwecke ausreichend Spielraum zum balancieren hat.

2.7.3. Fazit

Abgesehen von Anfangsschwierigkeiten mit dem Flashen des XMCs, war die Ansteuerung über PWM-Signale mit den internen Dave 4 APPs sehr einfach zu lösen, wobei natürlich auch das verspätete Auffinden der Dokumentation dieser APPsmit hineingespielt hat. Lediglich die Berechnung bzw. das Ausprobieren der mechanischen Einschränkungen und die korrekte Übergabe der Werte durch UART stellten eine kleine Herausforderung dar.

2.8. Threading auf dem Raspberry

Von Korbinian Federholzner

Das erhalten der UDP-Packete und der UART-Frames, wurde in Blocking-Loops implementiert, dadurch wäre die Anwendung die ganze Zeit durch diese Loops blockiert. Um dieses Verhalten zu umgehen wurden alle Listening-Loops in eigene Threads verschoben. Da die Anwendung zentral von einem Button gesteuert wird, welcher zwischen den Modus zum Regeln und dem Modus zur manuelle Steuerung mit der App wechselt, gesteuert wird, befindet sich die Ansteuerung des Buttons im main Thread. Für den Regeler und das manuelle Ansteuern gibt es auch wieder zwei Threads welche je nach Zustand des Buttons auf aktiv gesetzt werden. Um den Zugriff auf die gelesenen Daten von UART und UDP sicher zu gewährleisten, wurde mittels Mutex der Zugriff auf die Daten abgesichert. Dabei kann nur entweder UART schreiben oder der Benutzer die Daten abfragen, niemals beide gleichzeitig.

2.9. UDP-Socket

Von Patrick Lesch

2.10. iOS-Applikation

Von Patrick Lesch

2.11. Zusammenführung der einzelnen Komponenten

Diverse Probleme gab es beim Merge-Vorgang auf Gitlab mit den durch Dave4 automatisch erstellten Files. Diese erfassen die konfigurierten Einstellungen der XMC

APPs und überführen diese in ausführbaren Code. Dieses Problem wurde dadurch gelöst, dass man die Einstellungen auf einem Branch durchgeführt hat und lediglich die selbstgeschriebenen Code-Files zusammengeführt hat.

3. Benutzerhandbuch

3.1. Starten des Prototyps

Schritt 1:

3.2. Flashen des Custom-Kernels

Ohh nein. Wir haben den Raspberry getötet. Was sollen wir tun?! (Alle Kommandos sind im folgenden zwischen doppelten Anführungsstrichen: „command")

1. Raspbian Buster Lite herunterladen. (Wer braucht schon eine grafische Oberfläche) https://www.raspberrypi.org/downloads/raspbian/

2. Auf die SD-Karte flashen (z.B. mit BalenaEtcher unter Windows)
https://www.balena.io/etcher/

3. Raspberry-Pi booten und einrichten:
3.1 Login: „pi" — Passwort: „raspberry" (Achtung! Englisches Layout standardmäßig –> y == z und umgekehrt)
3.1.x1 Nach dem Bootvorgang ist der Bildschirm schwarz:
3.1.x2 SD-Karte in PC stecken und in der Config.txt unter [pi4] „dtoverlay=vc4-fkms-v3d" mit „#" auskommentieren. (Das erzwingt Legacy Treiber.) Falls das nichts bringt - google is your friend.
3.2 Config anpassen: „sudo raspi-config"
3.2.1 WLAN einrichten: Network Options (Dort dem Dialog folgen und die Verbindung einrichten)
[Verbindung prüfen. z.B. mit „ping google.com" oder mit „ifconfig"]
3.2.2 SSH einschalten: Interfacing Options –> SSH
3.2.3 Neustart: „sudo reboot"

4. Preempt-RT Kernel flashen:
4.1 Dem Tutorial hier folgen und Kernel selber kompilieren
https://lemariva.com/blog/2019/09/raspberry-pi-4b-preempt-rt-kernel-419y-performance-test oder den fertigen aus dem Git-Repository nehmen [Für den Raspberry Pi 4B „kernel_4_19_59-rt23-v7l+" nehmen (siehe Tutorial-Link)] https://github.com/lemariva/RT-Tools-RPi/tree/master/preempt-rt

4.2 Den Kernel an den Raspberry senden: „scp rt-kernel.tgz pi@<ipaddress>:/tmp”
4.3 Am Pi folgende Kommandos ausführen um den Kernel zu entpacken und einzurichten:

- „cd /tmp”
- „tar xzf rt-kernel.tgz”
- „cd boot”
- „sudo cp -rd * /boot/”
- „cd ../lib”
- „sudo cp -dr * /lib/”
- „cd ../overlays”
- „sudo cp -d * /boot/overlays”
- „cd ..”
- „sudo cp -d bcm* /boot/”

4.4 Config.txt bearbeiten um den neuen Kernel zu nutzen:
4.4.1 Mit nano öffnen: „sudo nano config.txt”
4.4.2 An das Ende folgendes einfügen: ”kernel=kernel7_rt.img”
4.4.3 Reboot („sudo reboot”)
4.4.4 Prüfen ob der Kernel funktioniert: „uname -r” sollte „4.19.59-rt23-v7l+” ausgeben. (Die Version die man kompiliert hat)

5. UART einschalten:
5.1 Config.txt mit nano öffnen: „sudo nano config.txt”
5.2 An das Ende folgendes einfügen: „enable_uart=1”
5.3 Reboot („sudo reboot”)

6. Kleines Raspberry Pi Display des Hardware-Kartons rotieren:
6.1 Config.txt mit nano öffnen: „sudo nano config.txt”
6.2 An das Ende folgendes einfügen: „display_rotate=2”
6.3 Reboot („sudo reboot”)

7. Software installieren / updaten:
7.1 „sudo apt update”

7.2 „sudo apt install pip”
7.3 QR installieren: „pip install qrcode[pil]” (https://github.com/lincolnloop/python-qrcode)

8. Befehle zur Bedienung:
8.1 Befehl um QR der eigenen IP zu generieren (die inneren ¨ gehören zum Befehl): „hostname -I — awk ’print $1’ — qr && hostname -I — awk ’print $1’ — xargs echo „BalancingPlate IP: $1” ” (Ein QR Code sowie der Text „BalancingPlate IP: <RaspberryPi-IP>”) sollte ausgegeben werden.)
9. Ab hier dann Korbi?

3.3. Bekannte Probleme

4. Fazit

Anhang

A. Abkürzungsverzeichnis

B. Abbildungsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

C. Bestelliste

Produktname	Produkt-Nr.	Kosten	Stück	Insegsamte Kosten	Link
Servomotoren	209913 - 62	25,90 €	3	77,70 €	https://www.conrad.de/de/p/hitec-standard-servo-hs-5485hb-digital-servo-getriebe-material-karbonite-stecksystem-jr-209913.html
Servohebel	1530677 - VQ	6,99 €	3	20,97 €	https://www.conrad.de/de/p/reely-alu-servohebel-geklemmt-47-mm-passend-fuer-futaba-servohebelkranz-anzahl-bohrungen-5-1530677.html
Gabelkopf	239186 - 62	7,55 €	1	7,55 €	https://www.conrad.de/de/p/modelcraft-aluminium-gabelkopf-mit-innengewinde-m3-5-st-239186.html
Kugelkopf	221868 - 62	6,71 €	3	20,13 €	https://www.conrad.de/de/p/modelcraft-stahl-kugelkopf-mit-aussengewinde-m3-1-st-221868.html
Bildschirm		43,99 €	1	43,99 €	https://www.conrad.de/de/p/joy-it-rb-lcd5-touchscreen-modul-12-7-cm-5-zoll-800-x-480-pixel-passend-fuer-raspberry-pi-inkl-touchpen-1503822.html
Touch-Folie	710-5240	99,73	1	99,73 €	https://de.rs-online.com/web/p/touchscreen-sensoren/7105240/
			mt:	226,08 €	

Abbildung 11: Bestellliste und Kosten des Projektes