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: 3117284
Name: Patrick Lesch
Matrikelnummer: 2917177
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.

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

Tabelle 1: Arbeisverteilung

2. Umsetzung

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

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

2.1.1. Mechanik

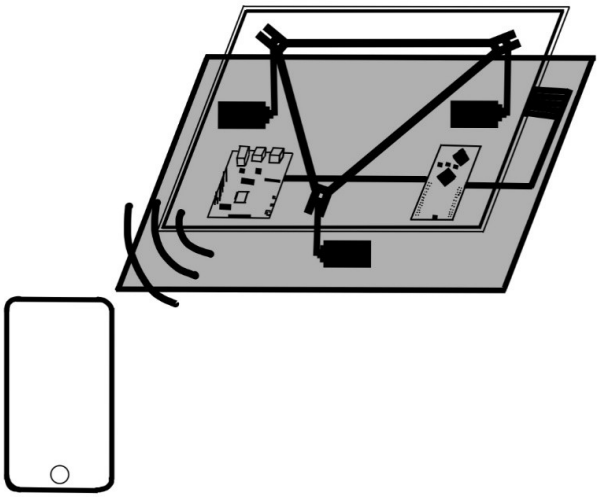


Abbildung 1: Skizzierter Aufbau des Prototypen

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 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.

Dabei war vor allem die korrekte Länge zum Mittelpunkt des Dreiecks, respektive der Grundplatte, eine wichtige Größe, da eine falsche Länge nicht mehr die senkrechte

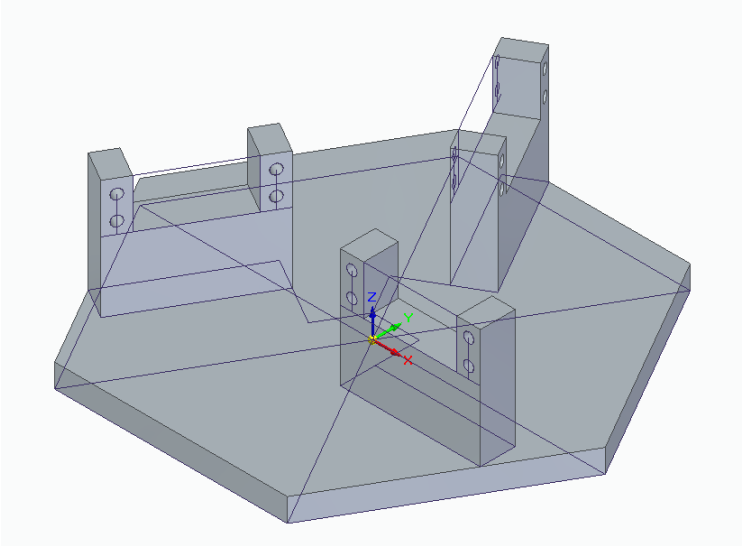


Abbildung 2: Grundplatte des Prototypen

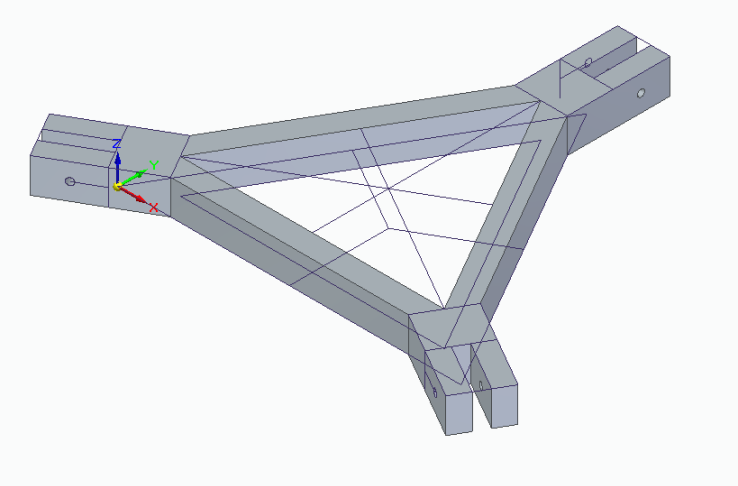


Abbildung 3: Dreiecksverbindung zur Touch-Folie

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.

2.1.2. Schaltplan und Verkabelung

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

Abbildung 4: Schaltplan des Prototyps

Die gesamte Verkabelung ist dabei in **??** zu sehen. Hier sind auch alle Pins verzeichnet, sollte sich einmal ein Kabel lösen. Dabei wurde die Stromversorgung für den XMC und den Raspberry Pi getrennt von der allgemeinen Stromversorgung für die Servomotoren gehalten, um ein Absacken der Spannung zu verhindern. Außerdem wurden sowohl die Servomotoren, als auch die UART-Verbindung auf dieselbe Erdung gelegt.

2.1.3. Fazit

Die Teile aus dem 3D-Drucker waren insgesamt sehr zufriedenstellend, lediglich die Löcher für die Schrauben an der Dreiecksverbindung mussten nachträglich aufgebohrt werden. Dasselbe galt für die Löcher in den Servohebeln. Außerdem blieben von einem Gabelkopf der Dreiecksverbindung leider nicht funktionsessentielle Teile an der Platte des Druckers hängen, somit war lediglich der ästethische Punkt beeinträchtigt. Glücklicherweise waren die Servomotoren ausreichend stark um die Platte ohne Probleme und mit einer ausreichenden Geschwindigkeit bewegen zu können. Des Weiteren wurde aufgrund von Zeitmangel die Verkabelung des Prototypen und die Halterung des Bildschirms lediglich provisorisch ausgeführt. An dieser Stelle wäre

für zukünftige Projekte noch Verbesserungspotential. Insgesamt liegt jedoch ein für unsere Zwecke einfacher und dennoch ausreichender physischer Aufbau vor.

2.2. Software-Architektur

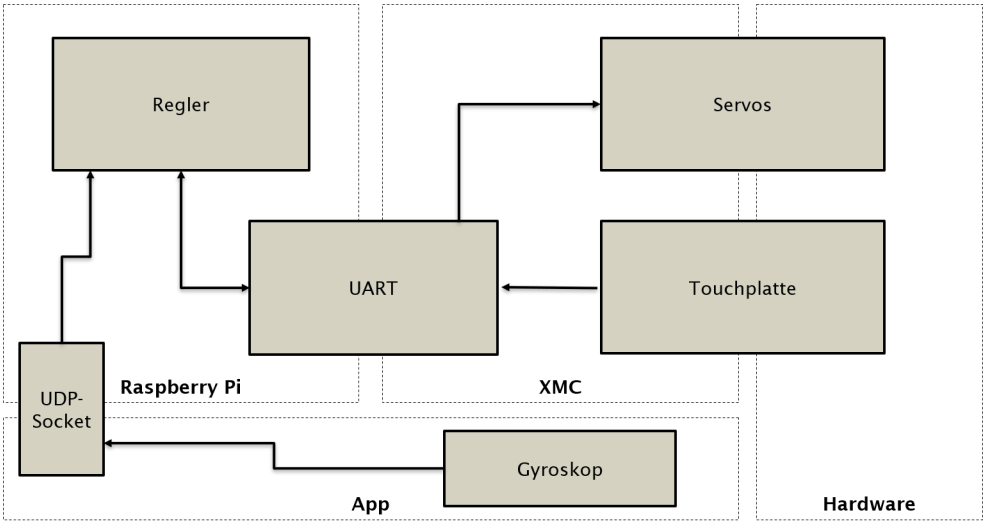


Abbildung 5: Grundaufbau der Software-Architektur

Wie in 5 zu sehen gliedert sich der Aufbau der Software in drei Teile. Zum einen der Programmteil auf dem XMC, der die eingehenden Daten verarbeitet und an die Servomotoren weiterleitet und gleichzeitig die Werte der Touch-Platte ausliest. Zum zweiten die Regelung auf dem Raspberry Pi, welche die ankommenden Werte von der Touchplatte umwandelt und an den XMC zurücksendet. Außerdem beschäftigt sich dieser Teil mit dem Threading zwischen den beiden Modi: Steuerung mittels App und selbstständige Regelung. Für die Steuerung per App müssen auch hier die ankommenden Daten verarbeitet werden. Die App bildet dabei den dritten Teil in dem die eingebauten Gyroskope des Handys ausgelesen und diese Werte an den Raspberry Pi übermittelt werden.

2.3. Touch-Folie

Von Korbinian Federholzner und Philipp Kramer

Nachdem feststand, dass dieses Projekt ein Balancing Plate sein wird, war einer der ersten Schritte die Suche nach vergleichbaren selbstbalancierenden Platten, um nachzusehen, welche Hardware diese verwenden. Hauptsächlich relevant war dafür,

wie die einzelnen Vergleichsprojekte die Position des Balles erkennen können. Eine der am beiden am meisten vertretenen Aufbauten war es, eine einfache Plastikplatte zu verbauen und den Ball mithilfe einer Kamera zu erkennen. Diese Umsetzung war aufgrund von mangelnder Kenntnis in der Bildverarbeitung und dem Ziel, das Endprodukt so simpel und kompakt wie möglich zu gestalten, allerdings nicht die letztendliche Wahl. Die Entscheidung fiel dafür eine Touch-Folie zu nutzen und mit einer ausreichend schweren Kugel per Druck die Koordinaten zu bestimmen. Da die Suche nach einer geeigneten Platte auf den deutschen Markt eingegrenzt war, war es schwerer als erwartet eine passende Touch-Platte zu finden. Endgültig fiel die Entscheidung auf das in Abb. 16 genannte Platte AMT 2517 der Apex Material Technology Corp.

2.3.1. Funktion der Touch Folie

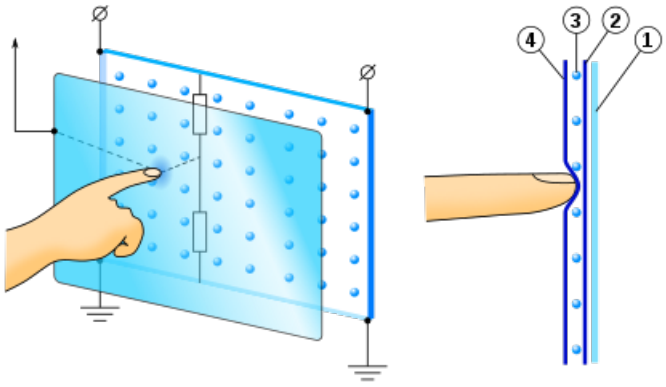


Abbildung 6: Funktion des Touch Panels

Zum Auslesen der Sensorwerte wird ein 5-Wire rezessives Touch Panel verwendet. Wie in Abb. 6 [**?**] gezeigt, besteht dieses grundsätzlich aus zwei Schichten. Wird ein Druck von 1.00 *Newton* oder mehr auf die Oberfläche des Panels ausgeübt, so berühren sich die beiden Schichten. Da das Touch Panel im Prinzip ein Spannungsteiler ist, ergeben sich, je nachdem wo gedrückt wird, unterschiedliche Widerstände, welche wiederum einen Rückschluss geben auf die Position des Druckpunktes. Der analoge Spannungswert, der durch Pin 3 zurückgegeben wird, ist also durch diesen Spannungsteiler bestimmt.
Um zu messen, wo auf der Touchplatte sich ein Druckpunkt befindet, werden zwei der fünf Pins des Panels auf Masse (GND) gesetzt und zwei auf Spannung (V_{DD}), im Fall dieses Projektes sind dies 5 *Volt*. Die Verwendung von fünf Kabeln bietet den Vorteil, dass der Pin in der Mitte, Pin 3, zum Auslesen des analogen Sensorwerts verwendet wird. Sind wie in Abb. 6 die oberen beiden Pins auf V_{DD} und die unte-

ren beiden auf GND, so lässt sich ermitteln, wo der Druckpunkt sich zwischen der oberen und der unteren Kante befindet. Bei einem kartesischen zweidimensionalen Koordinatensystem entspricht dies dem y-Wert. Da mit dieser Methode nur eine Richtung gemessen werden kann, für einen Punkt auf dem Panel allerdings sowohl ein y- als auch ein x-Wert benötigt wird, müssen die Pins so umgeschaltet werden, sodass beide Pins an der linken kurzen Seite auf V_{DD} sind und beide Pins auf der rechten Seite GND.
Die Werte der Platte liegen bei perfekten Bedingungen zwischen 0 und 4096, wobei 0 an der Kante, an der GND anliegt, ist und 4096 der maximale Wert an der V_{DD}-Seite. Bei diesem Projekt liegen die Werte, die letztlich ausgegeben werden, bei beiden Seiten zwischen 1500 und 2500. Die Verringerung der Anzahl der Messwerte ist zum Teil auf die in Kapitel 2.3.3 angeführten Probleme zurückzuführen.

2.3.2. Implementierung

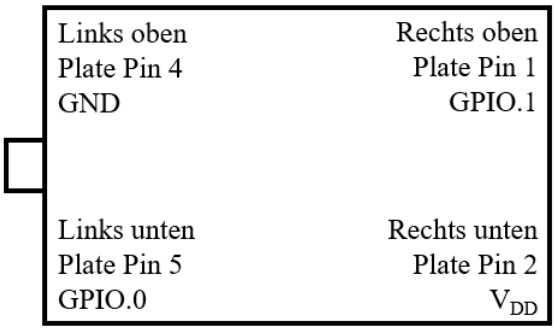


Abbildung 7: Skizze der Platte

Bei der Implementierung wird als Verbindung zur Touch Plate ein XMC4500 Mikrocontroller verwendet, welcher mit der Applikation Dave4 der Infineon AG programmiert wird. Die einzelnen Pins der Platte sind hierbei mit Pins des XMC verbunden. Da, wie in 2.3 beschrieben, die Pins an der Platte umgeschaltet werden müssen, um sowohl x-, als auch y- Koordinate auslesen zu können, werden zwei GPIO-Pins des XMC verwendet. Bei einem skizzierten Aufbau der Platte wie in Abb. 7 werden die Pins 1 und 5 mit GPIO-Pins des XMC verbunden. Pin 1 und Pin 5 müssen immer jeweils gegenteilig sein, also wenn Pin 1 high ist, ist Pin 5 low und vice versa. Der übrige Pin 3 wird ausschließlich dafür verwendet die Werte, die die Platte liefert, in je eine x-Variable und eine y-Variable zu schreiben. In diesem Projekt sind die Pins der Touch Plate wie in Tabelle 2 mit dem XMC verbunden.

Da das Panel nur mit einer Frequenz von 50 Hz geschaltet werden kann, also alle 0.02 Sekunden, ist ein Timer-Interrupt definiert. Dieser wird alle 20 ms ausgelöst,

Pin Touch Plate	Pin XMC
Pin 1	Pin 2.15
Pin 2	VDD
Pin 3	Pin 15.2
Pin 4	GND
Pin 5	Pin 2.14

Tabelle 2: Verbindungen der Pins der Touch Plate mit dem XMC

das Panel wird also maximal oft ausgelesen. Die Interrupt Service Routine (ISR) startet die Messung für den Analog Digital Converter (ADC). Dieser ADC wandelt die analoge Spannung, die per Pin 3 der Touch Plate an den XMC geleitet wird, in einen digitalen Wert um, der, wie bereits in Kapitel 2.3 beschrieben, zwischen 1500 und 2500 liegt. Sobald der ADC die Messung und Umwandlung abgeschlossen hat, startet er selbst ebenfalls eine Der ADC löst sobald er mit dem Messen fertig ist auch eine ISR aus. In dieser wird der gemessene digitale Wert je nach gemessener Koordinate in eine x- bzw. y-Variable gespeichert und daraufhin die GPIO Pins umgeschaltet, sodass die andere Koordinate ermittelt werden kann. Die GPIO-Pins sind beim Start des Programms so eingestellt, dass erst der x-Wert ausgelesen wird und daraufhin der y-Wert. Sind beide Werte vorhanden, so wird bei beiden Variablen jeweils 1500 abgezogen. Dadurch sind die weiterverwendeten nicht mehr im Bereich von 1500 bis 2500, sondern befinden sich zwischen 0 und 1000, was die darauffolgenden Berechnungen erleichtert. Nach der Umrechnung werden die beiden Daten in einem UART Frame verpackt und dann ein UART-Transmit ausgelöst. Der gesamte Ablauf ist schematisch in Abb. 8 dargestellt.

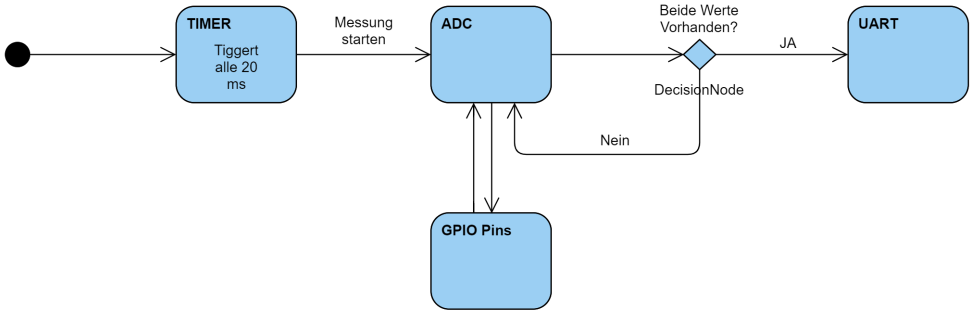


Abbildung 8: UML Aktivitätsdiagramm der Interrupts

2.3.3. Probleme

Ein Problem im Zuge der Implementierung der Touch-Platte war die Funktion der ADC-App im Dave4. Da bei dieser keine gute Dokumentation vorhanden war, war es z.B. schwer herauszufinden, wie sich eine Messung bei dem Analog Digital Converter starten lässt und wie die dafür benötigte Funktion heißt.
Eine weitere Problematik stellt der Randbereich der Platte dar. Dieser ist minimal höher als der Rest der Touch-Folie und auf diesem schwanken die Werte sehr stark. Anstatt eines erwarteten Wertes, der kleiner als 500 ist, werden dort teils Werte von über 2000 ausgegeben. Zur Behebung des Hindernisses wurde ein etwas kleinerer Innenbereich definiert und sollte ein Druckpunkt zwischen dem inneren Bereich und dem etwas erhöhten Rand sein, so wird die Platte maximal ausgelenkt, sodass sich der Ball möglichst wieder in Richtung Mitte der Platte bewegt. Das Hauptproblem, das bei der Implementierung der Platte anfiel, waren konstante Messfehler des Touch Panels. Das erwartete Verhalten ist, dass die x-Werte konstant bleiben, wenn sich die Druckpunkte parallel zur y-Achse bewegen. Dies entspricht beim endgültigen Aufbau einer Kugelbahn, die parallel zur kurzen Seite verläuft. Plausiblerweise wird das gleiche Verhalten auch bei den y-Werten erwartet, wenn die Druckpunkte parallel zur x-Achse liegen. Bei Tests stieg allerdings der konstant erwartete Wert entlang der Achsen immer weiter an. In Abb. 9 ist dieses Verhalten für die Messung des x-Werts dargestellt.

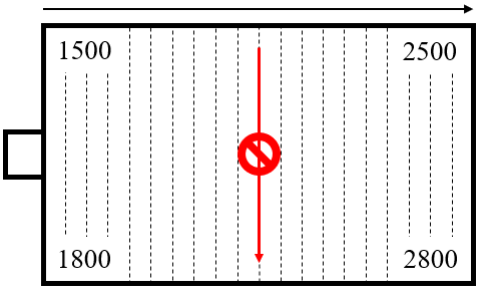


Abbildung 9: Problem der Plattenwerte am Beispiel der x-Werte

Nachdem die Touch Plate mittels eines Picoscopes auf Fehler untersucht war, dort aber alles wie erwartet funktionierte, wurden alle Pins des XMC auf eventuelle Mängel untersucht. Dort waren ebenfalls keine Fehler zu erkennen.
Der nächste Lösungsansatz war es die GPIO-Pins zu eliminieren, indem eine H-Brücke eingesetzt wird. Hintergrund dieser Idee war es, dass die GPIO-Pins des XMC keine wirklich konstante Spannung liefern können und mittels der H-Brücke der Strom nicht mehr vom Mikrocontroller, sondern direkt von der Stromversorgung

kommt. Allerdings fiel wie bei den GPIO-Pins des XMC über die H-Brücke die Spannung ab, was zu einem sehr ähnlichen Endszenario führte. Die nächste Überlegung war die Verwendung von Relais. Da Relais, die schnell genug für die Anforderungen dieses Projektes sind, sehr teuer sind und weiterhin deren Verschleiß hoch ist, wurde diese Idee letztlich ebenfalls nicht angewandt.
Die Lösung, die letztendlich im Projekt verwendet wird, ist, Widerstände von 100 *Ohm* zwischen die Pins des XMC und die der Platte zu schalten. Diese Widerstände minimieren den Fehler so weit, dass er entlang einer Achse nur noch etwa 40 ist und damit vernachlässigbar. Der minimale Fehler, der übrig bleibt, kann außerdem mithilfe des Reglers ausgeglichen werden. Mit der Kenntnis, dass dieser Ansatz den erhofften Erfolg brachte, lag der Fehler weniger an der mangelnden Konstanz der GPIO-Pins, sondern eher am zu geringen Widerstand, den die Platte liefert, was wiederum zu gelegentlichen Spannungsabbrüchen führte.
Ein anderes Problem, das im Laufe der Tests aufkam, war, dass die erste Kugel, die genutzt wurde, zu leicht war, wenn sich die Platte zu schnell bewegte. Dies war der Fall, da in dem kurzen Moment, in dem sich die Platte bewegte, der Mindestdruck von 1 *Newton* unterschritten wurde und dadurch falsche Werte an den XMC gesendet wurden. Daraufhin berechnete der Regler neue Plattenneigungswinkel, die allerdings basierend waren auf den unkorrekten Positionen. Eine geeignete Kugel war dann eine Flipperkugel, die der Inhaber der Flipperwerkstatt Regensburg zur Verfügung stellte.

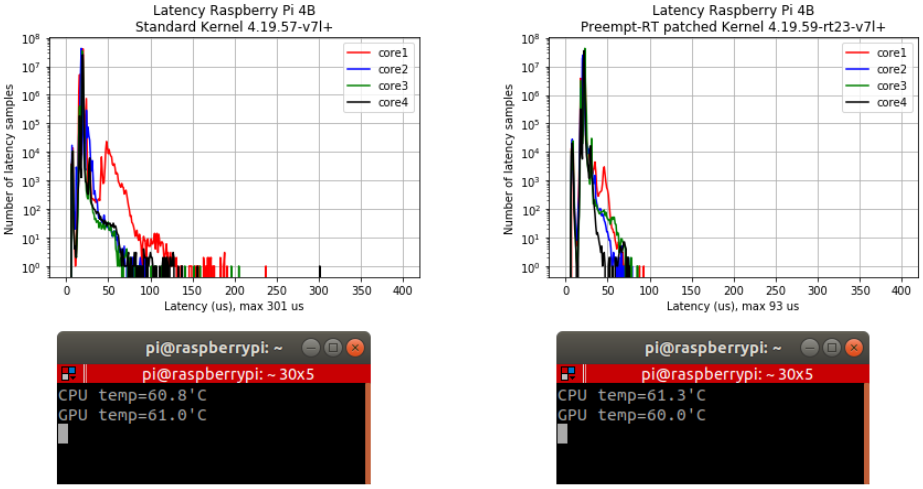
2.4. Einrichtung des Raspberry Pi

Von Patrick Lesch

2.4.1. Allgemeines

Aufgrund seiner Vielzahl von flexibel einsetzbaren Komponenten wurde der Raspberry Pi 4 als zentraler Bestandteil des Systems gewählt. Besonders die hohe Rechenleistung sowie das bereits verbaute Wlan-Modul und die einfache Bedienbarkeit waren hierbei ausschlaggebend. Als Betriebssystem wurde eine Aktuelle Version des Raspbian Buster Lite gewählt welches ein leichtgewichtiges Image ohne Grafische Oberfläche darstellt. Die Wahl viel auf dieses Image da hierfür der im anschließenden Kapitel erwähnte Preempt-RT Kernel existiert. Zudem wird keine Rechenleistung für das Darstellen von Grafiken und Animationen aufgewendet die eventuell die Geschwindigkeit von kritischen Prozessen negativ beeinflussen.

2.4.2. Echtzeit Preempt-RT Kernel



(a) Standard Raspbian Kernel (4.19.57-v7l+)

(b) Preempt-RT Raspbian Kernel (4.19.59-rt23-v7l+)

Abbildung 10: Latenzen und Temperaturen der jeweiligen Kernel

Um die Echtzeitfähigkeit des gesamten Systems zu optimieren wurde für den Raspberry Pi der Preempt-RT Kernel kompiliert und aufgespielt. Die Latenzen verbessern sich in Tests, wie in Abbildung 10 zu sehen ist, um einen Faktor von 3 während die Temperatur der CPU um nur 0.5° C steigt. Dies stellt objektiv betrachtet eine Verhältnismäßig gute Kosten-Nutzen Balance dar.

2.4.3. Ausblick

Auch wenn das Potential des Preempt-RT Kernels aktuell nicht voll ausgeschöpft wird bietet er das Potential künftige Funktionalität auf einem möglichst schnellen Level zu halten wodurch die Echtzeitfähigkeit des Systems gewährleistet wird.

2.5. UART Schnittstelle

Von Korbinian Federholzner und Michael Schmidt

2.5.1. Funktion

Zur Datenübertragung zwischen dem Raspberry Pi und dem XMC Mikrocontroller wurde das Universal Asynchronous Receiver Transmitter-Protokoll (UART) verwen-

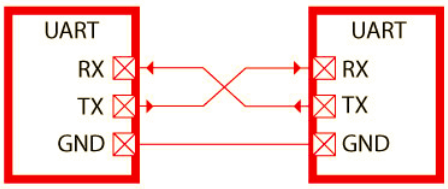


Abbildung 11: Grundaufbau der UART-Verbindung

det. 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 11 [**?**].

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 12 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.

2.5.2. Testen und erste Fehler

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 Sendeoptio-

nen 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.

2.5.3. Datenübertragungsformate

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 12 [**?**] verdeutlicht wird. Diese Zeichen werden dann aneinandergereiht 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 13. Übertragen werden hierbei die aktuellen Werte für die PWM-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.

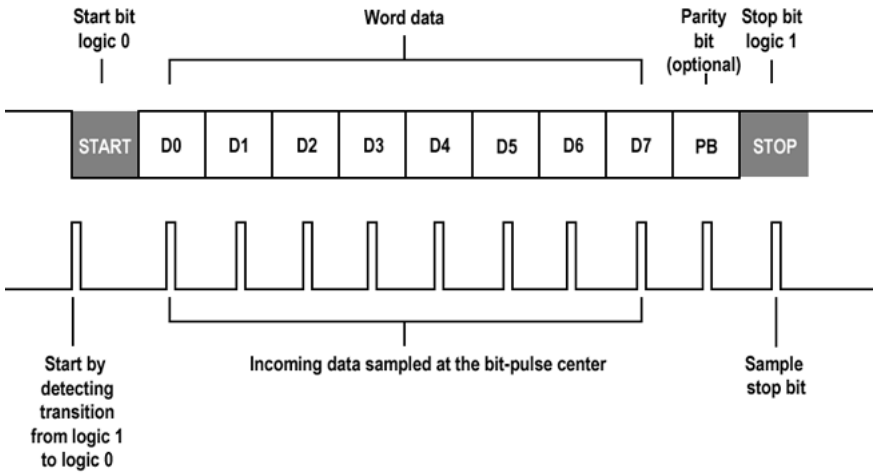


Abbildung 12: Aufbau eines UART-Frames

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

Abbildung 13: Aufbau eines Datenpakets mit Servowerten

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 14. 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 14: 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.

2.5.4. 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 Touch Panel 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.

2.5.5. Zeitliche Betrachtungen

Die Übertragungsgeschwindigkeit setzt sich hierbei zusammen aus Baudrate und in unserem Fall Message-Länge:
Baudrate/Message-Länge = Nachrichtenfrequenz
Da in unserem Fall jedoch String-Zeichen und keine einzelnen Bits übertragen werden ergibt sich folgende Frequenz für die Nachrichten vom XMC zum Raspberry Pi (Werte der Touch-Platte):
$19200/[(10 * 8) + (10 * 2)] = 192$Nachrichten pro Sekunde
Selbiges ergibt sich für die Nachrichten vom Raspberry Pi zum XMC (Servowerte). Sollte jedoch die Message-ID für die Kommunikation von XMC zu Raspberry Pi aktiviert sein ergibt sich durch die geänderte Frame-Länge folgende Rechnung:
$19200/[(13 * 8) + (10 * 2)] = 154,8$Nachrichten pro Sekunde
Sollte die UART-Verbindung, wie in den beiden Fällen angenommen, vollständig ausgelastet sein, reicht die Verbindung aus um die Kommunikation konsistent und vorallem schnell genug zu gewährleisten um eine Regelung zu ermöglichen. Dies liegt daran, dass die Servomotoren lediglich in einem 50 Hz takt arbeiten und die Verbindung somit deutlich schneller arbeitet

2.5.6. 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.

2.5.7. Ausblick

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. Allgemeines

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 könnte dabei in Zukunft mittels einer neuen Dreiecksverbindung gelöst werden, die auch niedrigere Auslenkungen erlaubt 15. 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

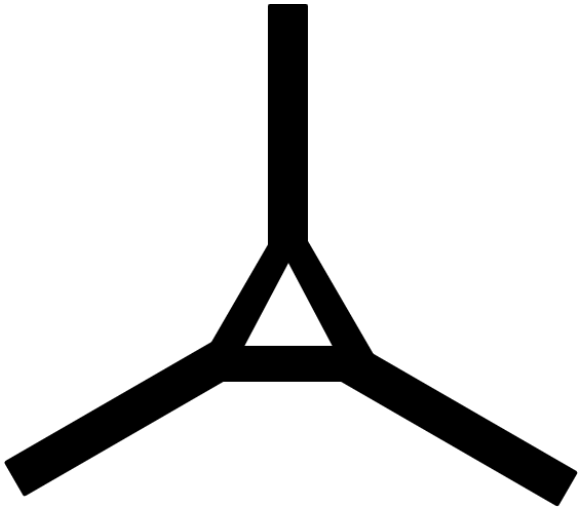


Abbildung 15: Aufbau einer alternativen Dreiecksverbindung mit größerem Wirkungsradius

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, wur-

de 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

Zum externen Empfang von Steuersignalen wurde auf dem Raspberry Pi ein UDP-Socket implementiert welcher auf eingehende Daten auf Port 1337 wartet. Dieser Port wurde festgelegt um den Kommunikationsaufbau zu vereinfachen und Störquellen zu minimieren. Durch einen festgelegten Port wird sichergestellt, dass keine ungewollten Nachrichten auf eventuell bereits belegten Ports eingehen und die Kommunikation erschweren. Zur Verbindung wird vorausgesetzt, dass sich die Geräte im selben Netzwerk befinden und dieses die Kommunikation zwischen den Geräten zulässt. Im Gegensatz zu anderen Protokollen erfüllt UDP am besten die Echtzeitanforderungen des Projekts da stets die aktuellsten Daten gesendet werden.

Zusätzlich zur iOS-Applikation können durch den UDP-Socket auch andere Geräte mit dem Raspberry Pi kommunizieren. Die Funktionalität kann so beliebig erweitert werden. Während der Recherche zu sinnvollen Echtzeitübertragungen wurde unter anderem in Erwägung gezogen die Kommunikation über Bluetooth statt Wlan stattfinden zu lassen. Dies hätte zwar potentiell geringere Latenzen aber auch einen begrenzten Raum zur Steuerung gebracht. Dank der Kommunikation über das Netzwerk kann die Platte auch aus weiter Entfernung bedient werden ohne, dass merkliche Verzögerungen eintreten. Eine weitere Überlegung stellten andere Protokolle wie Beispielsweise TCP dar. Diese waren für das Projekt aufgrund eines potentiellen Kommunikationsstaus ungeeignet, da die App dann auf eine Bestätigung des Raspberry Pi warten müsste. Somit wäre nicht gewährleistet, dass stets die aktuellsten Daten empfangen werden. Ein eigener Hotspot wurde aus Gründen der Bürokratie als Idee verworfen, da die Hochschule diesen erst speziell genehmigen müsste.

2.10. iOS-Applikation

Von Patrick Lesch

2.10.1. Allgemeines

Eine der Anforderungen an das Projekt war, dass die Platte manuell steuerbar sein sollte. Dazu wurde eine Applikation für iOS-Geräte entwickelt. Ziel der Applikation ist es die Neigung des mobilen Endgerätes an den Raspberry Pi zu übertragen. Dieser übersetzt die übermittelten Werte in Neigungswinkel mit deren Hilfe der XMC die Motoren ansteuert welche die Platte in eine Lage versetzen welche die Lage des iOS-Gerätes spiegelt.

Um eine möglichst einfache und Geräteübergreifende Installation auf verschiedenen Geräten zu gewährleisten wurde erwogen die App als Webapp zu implementieren. Dieser Ansatz wurde aufgrund von in iOS 13 von Apple eingeführten Policies verworfen. Die Abfrage zur Genehmigung des Auslesens des Gyroskops muss für verschiedene iOS Versionen aktuell unterschiedlich implementiert werden. Das hätte einen für das Projekt unverhältnismäßigen Mehraufwand bedeutet der den zeitlichen Rahmen des Projekts gesprengt hätte.

Ein weiteres Gegenargument stellte zudem die durch den Browser festgelegte Orientierung der Seite dar. Wird das Gerät gedreht während die Orientierung des Gerätes entsperrt ist dreht sich dieser entsprechend mit. Dies würde bei jedem Kippen des Gerätes den Browser drehen was wiederum die zu übertragenden X- und Y-Winkel des Gerätes drehen würde. Durch eine native Implementierung als iOS-App kann gewährleistet werden, dass die Orientierung der App immer festgesetzt ist und die Daten unabhängig von Systemweit aktivierter Orientierungssperre gleich bleibt.

2.10.2. Views

Die Applikation verfügt im wesentlichen über zwei Views. Der erste View der nach dem Splash-Screen gestartet wird ist der QR-Scanner. Wird ein QR-Code erfolgreich gescannt wird zum eigentlichen Steuerungs-View gewechselt welcher die Winkel der X- und Y-Achse, sowie die IP die mittels QR-Code eingelesen wurde anzeigt.

2.10.3. Implementierung

Umgesetzt wurde die App zum großteil in nativem Swift-Code. Für den Aufbau der UDP-Verbindung und dem einrichten des sendenden Sockets werden Apples C Libraries verwendet. Als Verbindungsart zwischen iOS-App und Raspberry Pi wurde UDP gewählt um eine Echtzeitübertragung zu garantieren. Da die App dank UDP kein Acknowldge des Raspberry Pi erwartet gibt es weder Verzögerung noch einen Datenrückstau. Gegebenenfalls bei der Übertragung verlorengegangene Daten

wurden in Kauf genommen um stets möglichst aktuelle Werte an die Platte zu liefern.

Da der Raspberry Pi, wie in Kapitel 2.9 erwähnt, auf Port 1337 lauscht sendet die App nur auf diesem ihre X- und Y-Winkel. Die Winkel werden von der iOS-Schnittstelle die das Gyroskop des Gerätes ausliest in Radiant angegeben. Vor der Übertragung werden diese aufbereitet indem sie in Grad umgerechnet und auf 3 Nachkommastellen gerundet werden. Zusätzlich werden diese Werte mit einem Faktor von 0.5 multipliziert, da sich gezeigt hat, dass die Bewegung der Platte die notwendig ist um die Kugel zu bewegen kleiner ist als die Neigung die ein Nutzer sinnvoll steuern kann. Ohne diese Ergänzende Anpassung reichten bereits minimale Bewegungen um die Platte stark zu bewegen was ein Kontrollieren der Kugel schwierig machte.

Die aufbereiteten Winkel werden zur Übertragung an die Sende-Funktion als String in dem Format „X_ACHSEN_WINKEL;Y_ACHSEN_WINKEL;" übergeben welche diesen über den festgelegten Port an die gescannte IP überträgt. Auf dem Raspberry Pi kann anhand der Semikola zwischen den einzelnen Winkeln einer Nachricht unterschieden werden. Das Aktualisierungsintervall des Gyroscops wurde aufgrund guter Ergebnisse auf 60 Hz festgelegt.

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. Im Bereich der App gestaltete sich das Zusammenführen der Komponenten dank guter Zusammenarbeit denkbar einfach und es traten an dieser Stelle keine unvorhergesehenen Fehler auf.

3. Benutzerhandbuch

3.1. Starten des Prototyps

Schritt 1: Einloggen mit Benutzername „pi" und Passwort „raspberry"
Schritt 2: Aufrufen des QR-Codes mit Hilfe des folgenden Kommandos:
(die inneren ¨ gehören zum Befehl)

„raspberry" „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.)
Schritt 3:

3.2. Flashen des Custom-Kernels und allgemeine Einrichtung

Alle Kommandos sind im folgenden zwischen doppelten Anführungsstrichen: „command"

1. Raspbian Buster Lite herunterladen. 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)
7.4. WiringPI: „sudo apt install wiringpi”
7.5. „sudo apt install cmake”

8. Starten des Programmes
8.1 Navigieren in das Verzeichnis in dem CMakeList.txt liegt
8.2 „cmake .”
8.3 „make”
8.4 danach sollte eine Executable in /bin liegen

3.3. Bekannte Probleme

- Wurde mittels QR-Code eine falsche IP gescannt muss die App neu gestartet werden

4. Fazit

Das Projekt wurde fristgemäß mit allen daran festgelegten Anforderungen abgeschlossen. Es befindet sich in einem funktionsfähigen und erweiterbarem Zustand. Dennoch besteht weiteres Verbesserungspotential. So könnte etwa die derzeit noch langsame und etwas ruckelige Regelung optimiert werden. Des weiteren könnten weitere Funktionen eingebaut werden. Wünschenswert wäre hier etwa die graphische Ausgabe der aktuellen Kugelposition in der iOS-App oder das festlegen eines Punktes auf dem iOS-Geräte Display welches die Kugel analog auf die entsprechende Position der Platte bringt. Dadurch könnten auch noch gewisse Formen nachgefahren werden (z.B. ein Unendlichkeitszeichen).
Insgesamt kann das Projekt dennoch als Erfolg betrachtet werden, da ein funktionsfähiger Prototyp mit kleineren Fehlern geschaffen wurde, der ein großes Erweiterungspotential besitzt und außerdem für Vorführungen zur verfügung steht.

Anhang

A. Abbildungsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

B. Bestellliste

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 16: Bestellliste und Kosten des Projektes

C. Stundenliste

Name	Gesamt	Aufgabe 1	Aufgabe 2	Aufgabe 3
Michael Braun	125	lirem ipso lorem	lirem ipso lorem	lirem ipso lorem
Korbinian Federholzner	128	lirem ipso lorem	lirem ipso lorem	lirem ipso lorem
Philipp Kramer	123	Touch-Folie 63	Testing 19	Reglerentwicklung 41
Patrick Lesch	125	lirem ipso lorem	lirem ipso lorem	lirem ipso lorem
Michael Schmidt	122	UART 62	Phys. Bau 40	PWM-Signale 20