Technische Hochschule **Rosenheim**

Fakultät für
Fakultät für Ingenieurwissenschaften

Studiengang
Maschinenbau

Entwickeln und Bewerten einer Abstandsregelung mithilfe eines Smartphones als Sensor

Developing and evaluating a distance control using a smartphone as a sensor

Bachelor Thesis

von

Leon Hilpoltsteiner

Datum der Abgabe: 01.03.2021

Erstprüfer: Prof. Dr. Peter Zentgraf

Zweitprüfer: Prof. Dr. Ing. Frank King

Inhaltsverzeichnis

1 Einführung

1.1 Motivation

Regler begegnen uns nicht nur im industriellen Umfeld. Auch in der im täglichen Leben genutzten Technologie und letztendlich in der uns umgebenden Umwelt erleben wir sie, ohne sie bewusst wahrzunehmen. Dabei erfüllt der Mensch selbst permanent die Funktionen eines Reglers bzw. einer Vielzahl von Reglern. Dies beginnt bei einfachen Vorgängen wie dem Wegziehen der Hand nach der Berührung einer heißen Oberfläche und reicht bis zu komplexeren Aufgaben wie der Abstandsregelung zu einem vorausfahrenden Fahrzeug durch konstante visuelle Erfassung und Verarbeitung sowie dem ausgleichenden Eingreifen durch Beschleunigen oder Bremsen. Wie hoch die Komplexität der ablaufenden Regelungsvorgänge ist, lässt sich an dem Aufwand ermessen den die Systementwickler in Abstandsregelsysteme stecken müssen, um diese auf das gleiche Niveau wie das eines aufmerksamen und gut ausgebildeten Fahrers zu bringen.

Um bei dem Beispiel des Fahrzeuglenkers zu bleiben: Stellt der Fahrer fest, dass der Abstand zum vorausfahrenden Fahrzeug zu gering geworden ist, wird der Autofahrer bremsen und hat somit die Funktion eines Reglers ausgeführt. Der Autofahrer ist also Teil des Regelkreises geworden, dieser steht für einen in sich geschlossenen Wirkungsablauf mit Einfluss auf eine physikalische Größe (Geschwindigkeit).

Ein Beispiel mit einem Menschen als Teil eines Regelkreises soll in der vorliegenden Arbeit genutzt werden, um Neulingen bezüglich Regelungstechnik den Einstieg in das Gebiet zu erleichtern. Regelungstechnik wird oft als sperrig und abstrakt wahrgenommen. Mithilfe eines einfachen Experimentes soll das Prinzip erlebbar, sowie messtechnisch erfasst und visualisiert werden. Für regelungstechnische Experimente werden normalerweise teure Hardware und komplexe Versuchsaufbauten benötigt. Abweichend hiervon sollen für dieses Experiment bewusst nur minimale Materialanforderungen an den Probanden bestehen. Da praktisch jeder Studierende ein Smartphone nutzt, sollen die darin verbauten Sensoren mit Hilfe einer zu entwickelnden Applikation für den Versuchsaufbau genutzt werden. Der Mensch wird somit zum Teil des Regelkreises und nutzt aktiv eine Technologie seines täglichen Lebens.

1.2 Zielsetzung

Es soll in dieser Arbeit also ein Experiment erstellt werden, dass den Bediener aktiv in den Regelkreis einbindet. Hierzu muss ein Regelkreis bestimmt werden, mit allen nötigen Ein- und Ausgangsgrößen. Um dies zu verwirklichen, wird sich eines Smartphones und dessen Funktionen bedient. Um diese Funktionen anzusprechen, soll weiterhin eine Applikation für ein Smartphone entwickelt werden. Der Bediener der Applikation soll als Regler fungieren, also muss der Bediener aktiv auf den Regelkreis einwirken. Wenn man die Funktionen eines Smartphones betrachtet, fällt einem direkt auf, dass dieses viele verschiedene Sensoren in sich verbaut hat. Sensoren sind ein integraler Bestandteil der Mess- und Regelungstechnik. Also bietet es sich direkt an Gebrauch von diesen Sensoren zu machen, um das Experiment zu verwirklichen. Nach Durchführung des Experimentes sollen zusätzlich, so viele Daten wie möglich vorliegen. Dies dient zum einen der Auswertung des Experimentes, aber auch um den Studenten zu helfen Zusammenhänge zu verstehen. Die Applikation soll jedem Studenten zugänglich und leicht zu nutzen sein. Das Gleiche gilt für das Experiment, es darf nicht zu lang oder zu kompliziert sein, sondern muss leicht verständlich sein, so dass auch jemand der wenig Ahnung von Regelungstechnik hat alles Geschehene nachvollziehen kann.

2 Grundlagen

2.1 Verwendung eines Smartphones für das Experiment

Für die Verwendung eines Smartphones gibt es mehrere Gründe. Zum einem steht so fest, dass das Experiment keine hohen Hardwareanforderungen stellt. So wird gewährleistet, dass das Experiment an den meisten Orten, also auch in einem Hörsaal, durchgeführt werden kann. Auch sind in einem Smartphone viele Sensoren verbaut, die für das Experiment genutzt werden können. Sensoren sind ein notwendiger Bestandteil der Regelungstechnik. Es liegt also nahe sich dieser Sensoren zu bedienen. Moderne Smartphones besitzen mehrere Sensoren gleichzeitig: Beschleunigungssensoren, ein Gyroskop, ein Magnetometer, Feuchtigkeitssensoren oder GPS-Sensoren. Nun musste festgelegt werden, welcher dieser Sensoren sich anbietet, um in einem Experiment verwendet zu werden. Dazu muss ein einfacher Regelkreis betrachtet werden.

2.2 Der Regelkreis

Zuerst wird ein einfacher Regelkreis betrachtet: Es gibt einen Eingang, dann folgt der Regler $\boldsymbol{G_R}$, nun die Regelstrecke $\boldsymbol{G_S}$ und schließlich der Ausgang. In den Eingang wird ein Sollwert **W** gegeben, am Ausgang wird die sogenannte Regelgröße **Y** ausgegeben. Kurz vor dem Ausgang, wird der Istwert nochmal abgegriffen und zurück bis zum Eingang geführt. Wird dieser nun von dem Sollwert abgezogen spricht man von negativer Rückkopplung. Die Summe beider Werte wird Regeldifferenz **E** genannt. Diese Regeldifferenz **E** wird an den Regler $\boldsymbol{G_R}$ weitergeleitet, dieser gibt die Stellgröße **U** aus. Diese Stellgröße **U** wirkt auf die Regelstrecke $\boldsymbol{G_S}$ und schließlich erhält man die Regelgröße **Y**. Dieser Regelkreis ist in **Abbildung 2.1** zu sehen.

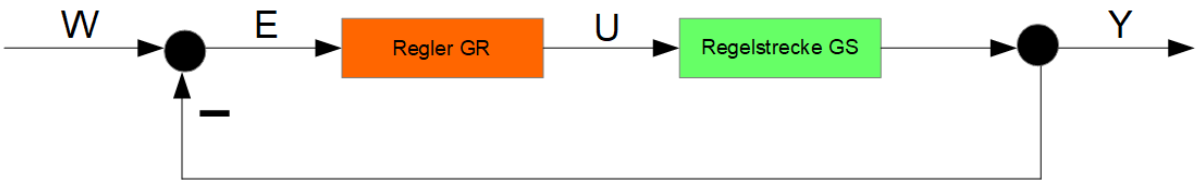


Abbildung 2.1 Allgemeiner Regelkreis

Nun muss festgelegt werden, wie der Proband und das Smartphone in den Regelkreis eingebaut werden. Am meisten Einfluss hat der Proband, wenn er als Regler auftritt. So nimmt der Proband aktiv Einfluss auf die Stellgröße **U** und damit auf die Regelgröße **Y**. Außerdem lässt es den Probanden leichter verstehen, welche Funktion der Regler in einem Regelkreis ausführt. Somit steht fest, dass das Smartphone, beziehungsweise dessen Sensoren, in die Regelstrecke integriert werden müssen. Nun kristallisiert sich heraus, welcher Sensor des Smartphones benutzt werden sollte. Temperaturänderungen sind gut wahrnehmbar aber im Allgemeinen das Ergebnis eher langsam ablaufender Prozesse. Der Mensch kann jedoch Kräfte und somit Beschleunigungsvorgänge, die er über seine Muskelgruppen auslöst, sehr gut interpretieren und steuern. Somit bieten sich die Beschleunigungssensoren des Smartphones für die Durchführung des Experimentes an. So entstand die grundlegende Idee für das Experiment: Der Proband erhält einen Sollwert für eine zurückgelegte Distanz, ausgegeben durch das Smartphone Display. Nun muss der Proband eine Beschleunigung auf das Smartphone auswirken, diese Beschleunigung wird mithilfe der Sensoren gemessen. Mithilfe der

Beschleunigungsdaten kann die zurückgelegte Strecke berechnet werden. Auch diese Information wird durch das Smartphone Display geäußert. Der Proband, der hier die Rolle des Reglers übernimmt, muss nun versuchen den bereitgestellten Sollwert so gut wie möglich einzuhalten.

Mit dieser Information kann bestimmt werden, was diese ganzen Größen für das Experiment bedeuten. Begonnen wird mit dem Sollwert **W**: Dieser gibt vor zu welchem Zeitpunkt, was für eine Distanz zurückgelegt sein sollte. Also ein typisches Strecke-Zeit-Diagramm, welches auf dem Smartphonebildschirm angezeigt wird. Die Regelgröße **Y** gibt also an wie weit der Proband das Smartphone bewegt hat. Mit der negativen Rückkopplung resultiert die Regeldifferenz **E**, also die Differenz zwischen Sollwert **W** und Regelgröße **Y**. In diesem Fall um wieviel zu kurz oder zu weit der Proband das Smartphone bewegt hat. Da der Regler $\boldsymbol{G_R}$ der Proband ist, geht die Regeldifferenz **E** an ihn, erkennen soll er das dann am Smartphonebildschirm. Die Stellgröße **U** ist also die Beschleunigung, mit der der Proband das Smartphone beschleunigt oder abbremst. Diese geht über die Regelstrecke $\boldsymbol{G_S}$, einem zweifachen Integrator, im Laplace-Bereich $\frac{1}{s^2}$ und resultiert schließlich wieder in der Regelgröße **Y**. Da nun alle Kenngrößen des Regelkreises definiert sind, sollte es möglich sein den Regler zu identifizieren und dessen Übertragungsfunktion aufzustellen. Der Proband erhält also nach dem Experiment noch genaue Informationen zu seinem Verhalten. Der bestimmte Regelkreis ist in **Abbildung 2.2** zu sehen.

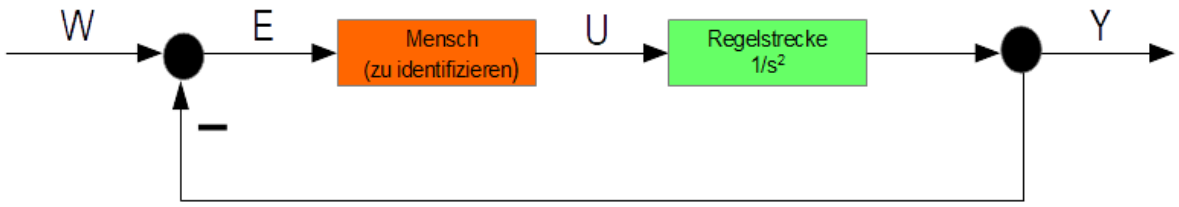


Abbildung 2.2 Bestimmter Regelkreis

2.3 Verwendung der Beschleunigungssensoren des Smartphones

Die Beschleunigungssensoren nehmen Daten entlang der x-, y- und z-Achse des Smartphones auf. Dabei sind die Achsen wie in **Abbildung 2.3** angeordnet. [1]

Bei Stillstand beträgt die Beschleunigung der x- und y-Achse 0 $\frac{m}{s^2}$, die der Z-Achse hingegen beträgt 9.81 $\frac{m}{s^2}$. Dies entspricht der Erdbeschleunigung. Während der Entwicklung dieses Projekts wurde sich bemüht alle drei Beschleunigungsrichtungen für die Berechnung der zurückgelegten Distanz zu nutzen, sowie weitere Sensoren. So sollte es möglich sein, sich mit dem Smartphone in der Hand zu bewegen und eine Strecke zu berechnen. Allerdings stellte sich dies als zu schwierig heraus und konnte nicht verwirklicht werden. So wurde sich darauf festgelegt, nur die Beschleunigung entlang der y-Achse zu nutzen. Die Applikation soll also so ausgelegt werden, dass eine Linearbewegung entlang der y-Achse ausgeführt werden soll. Die Beschleunigungswerte die in y-Richtung aufgenommen wurden, sollen dann Rückschluss auf die zurückgelegte Strecke geben. Diese Berechnung setzt voraus, dass das Smartphone nicht merklich gekippt oder gedreht wird.

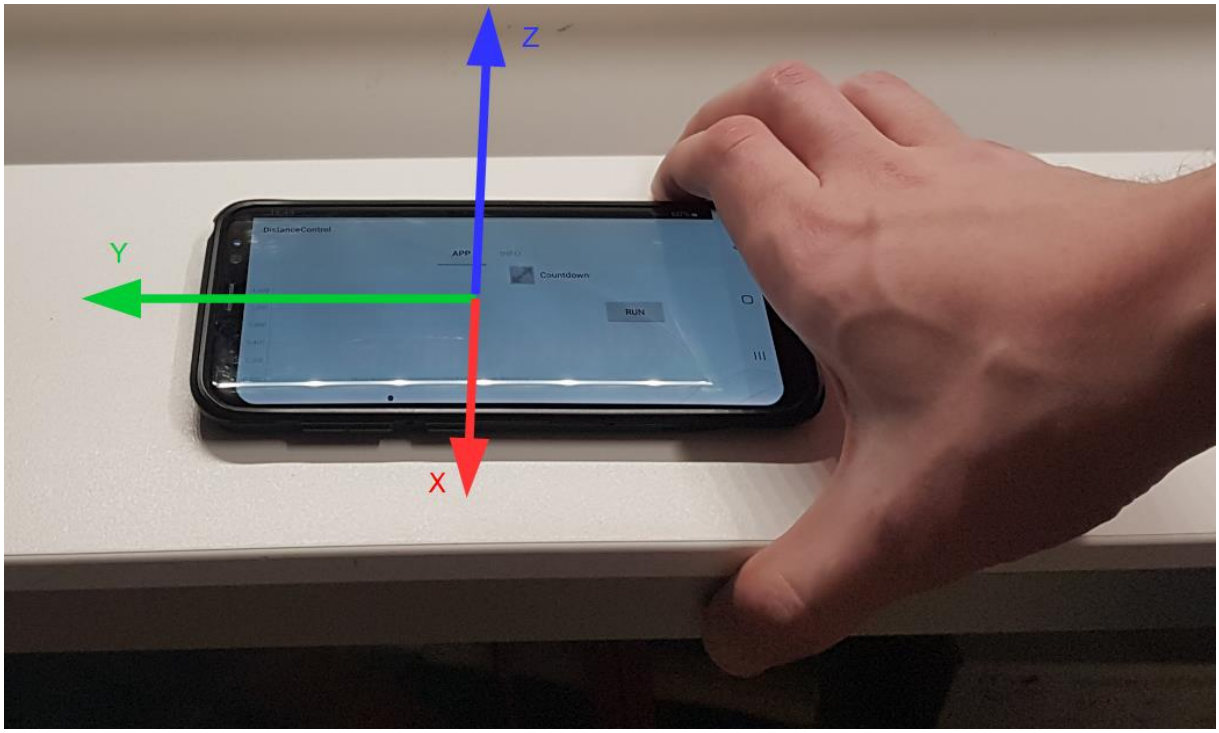


Abbildung 2.3 Achsen des Smartphones

[1] Mathworks: Accelerometer. https://www.mathworks.com/help/supportpkg/android/ref/accelerometer.html 28.02.2021 23 Uhr

2.4 Der PID-Regler

Einer der prominentesten Regler ist der sogenannte PID-Regler. Der Name PID setzt sich aus mehreren Namen zusammen: Proportional P, Integral I, Differenzial D. Der PID-Regler berechnet aus der Regeldifferenz E, die Stellgröße U. Hierbei wird die Stellgröße U aus proportionalen, integralen und differenzialen Anteilen ($k_P, k_{I,}\ k_D$) der Regeldifferenz E zusammengesetzt[2].

Übertragungsfunktion PID-Regler[3]:

$$G_{PID}(s) = k_P + \frac{k_I}{s} + k_D * \frac{s}{1 + T_1 * s} \qquad (2.1)$$

Nun wird angenommen, dass der Mensch ein ähnliches Verhalten wie ein PID-Regler aufweist. Der Proband besitzt also die Übertragungsfunktion eines PID-Reglers. Diese Übertragungsfunktion lässt sich mit den nötigen Daten berechnen. So kann dem Probanden später seine Übertragungsfunktion vorgelegt werden.

2.5 Simulink Modell für das Smartphone und Anforderungen an dieses

Schon früh war klar, dass das Projekt mithilfe von MATLAB umgesetzt wird. MATLAB eignet sich hervorragend für numerische Simulationen und Datenerfassung. Auch enthält MATLAB die Software Simulink, mit der sich Simulationsmodelle, grafisch erstellen lassen. Hinzu kommt, dass MATLAB sogenannte Toolboxes anbietet. Diese bieten dem Probanden erweiterte Möglichkeiten, bei der Erstellung von Modellen. Ergänzend dazu gibt es ein Simulink Support Package für Android Geräte. Dieses enthält Blöcke, um die Sensoren von Android Geräten anzusprechen oder Daten zu übertragen. In Simulink erstellte Modelle, lassen sich als .apk Datei auf das Smartphone übertragen, auf dem sie dann installiert werden können. Damit lagen alle notwendigen Ressourcen vor, um eine Applikation zu erstellen die die folgenden Aufgaben löst:

- Plot eines Sollwertverlaufes **W**
- Erfassen von Sensordaten, um die Regelgröße **Y** zu berechnen und zu plotten
- Übertragung von **W** und **Y** vom Smartphone an den PC.

[2] Serge Zacher, Manfred Reuter: Regelungstechnik für Ingenieure 14. Auflage, Springer, Seite 135
[3] Zentgraf, Peter: Vorlesungsskript Regelungstechnik 1

2.6 Simulink Modell für dem PC und Anforderungen an dieses

Nun müssen die Daten, die den PC erreichen weiterverarbeitet werden. Dazu braucht man ein weiteres Simulink Modell, dass die Daten erstmals empfängt. Daraus entstand die Idee, dass man die Regelgröße **Y** auch in Echtzeit am PC anzeigen kann. Für die Auswertung eignet sich ein MATLAB-Skript, da hier die verschiedenen Größen verrechnet und geplottet werden können. So lässt sich leicht die Regeldifferenz **E** berechnen. Die Daten sollten also ein Simulink-Modell am PC durchlaufen und dann einem MATLAB Skript bereitgestellt werden, welches die Auswertung vornimmt.

Dabei sollte das Simulink-Modell folgende Aufgaben lösen:

- Empfangen der Daten
- Nachbau des Sollwertverlaufes **W** und Echtzeitplotten der Regelgröße **Y**
- Abspeichern der übertragenen Daten in den MATLAB Workspace.

Das MATLAB Skript sollte können:

- Starten des Simulink-Modells
- Berechnen der Regeldifferenz **E**
- Berechnen des fit-Wertes von **Y** und **W**
- Plots von **W**, **Y**, **E** und **U** erstellen
- Zurechtlegen der Daten für die Identifikation von **GR**
- Identifikation von **GR**
- Auswerten und plotten der Ergebnisse

2.7 Verwendete Programme

2.7.1 MATLAB

MATLAB dient der numerischen Lösung von Problemen. Es ist ein weit verbreitetes Programm an Hochschulen, auch an der TH Rosenheim wird der Umgang mit MATLAB gelehrt. Mit MATLAB kann man Skripte, sogenannte „m-files“ erstellen. Diese berechnen oder plotten nach Ausführung, gewünschte Ergebnisse. Im Zuge dieser Arbeit wurden verschiedene „m-files“ erstellt, die der Ausführung des Experimentes dienen.

2.7.2 Simulink

Mit Simulink lassen sich Simulationsmodelle grafisch erstellen. Dabei werden dem Nutzer Blöcke bereitgestellt, die dieser nutzen kann, um Ablaufpläne zu erstellen. Es können zusätzlich „m-files“ in Simulink Modelle eingebunden werden. Zwischen MATLAB und Simulink lassen sich Daten austauschen, so kann sich beider Programme bedient werden, um am gleichen Projekt zu arbeiten. Für diese Arbeit wurden mehrere Modelle in Simulink erstellt.

2.7.3 Simulink Support Package for Android Devices

MATLAB bietet die Installation von sogenannten Support Packages an. Diese enthalten meist zusätzliche Funktionen oder Blöcke, die in Simulink Modellen untergebracht werden können. Zu diesen gehört auch das „Simulink Support Package for Android Devices“. Dieses enthält Blöcke die die Sensoren des Smartphones ansprechen oder Daten vom Smartphone an andere Geräte übertragen. Auch wird es benötigt, um Code zu generieren der dann in der Form einer Applikation auf das Endgerät übertragen wird.

2.7.4 pzMove

Die Software „pzMove“ wird von der TH Rosenheim angeboten. Mit dieser lassen sich Regelstrecken identifizieren, analysieren und Reglerparameter auslegen. Sie bietet auch eine graphische Oberfläche, über die Pole oder Nullstellen angezeigt werden können.

2.7.5 Android Studio

Die Software „Android Studio“ wird von Google bereitgestellt, um Nutzern zu erlauben eigene Applikationen in einer einsteigerfreundlichen Umgebung zu erstellen. Zum einen benötigt Simulink diese Software, um den Code zu generieren und zum anderen ist es mit „Android Studio“ möglich den generierten Projektordner zu öffnen, um so die Applikation im Nachhinein noch anzupassen.

2.7.6 Arduino DIE

Die open-source Software ermöglicht es Nutzern Code in C++ zu schreiben, zu kompilieren und auf Arduino Hardware aufzuspielen. Diese Software wurde benutzt um den Mikrokontroller, der für die Datenübertragung genutzt wird, einzurichten.

3 Simulink-Modell für das Smartphone

3.1 Einschränkungen von Simulink Modellen für Smartphones

Um eine Applikation für ein Smartphone zu erstellen, bietet Simulink eine sehr komfortable Lösung an. Der Nutzer erstellt ein Modell, verbindet sein Smartphone per USB mit dem PC und kann dann direkt die Applikation generieren und installieren lassen. Das bedeutet das jeglicher Code aus dem Modell bezogen wird und der Nutzer selbst keine großen Kenntnisse von Programmiersprachen besitzen muss. Allerdings gibt es für Simulink-Modelle am Smartphone einige Einschränkungen, die am PC nicht vorliegen.

Das größte Problem ist, dass das Simulationsmodell direkt ausgeführt wird, sobald man die Applikation startet. Auch können bestimmte Berechnungen nicht durchgeführt werden, weil die Applikation keine alten Werte abspeichern kann. So können keine Vektoren mit Zeitwerten erstellt werden. Dies sind zwei bedeutende Probleme, da diese Funktionen für das Experiment essenziell sind. Zum einen muss der Proband bestimmen können, wann die Applikation ausgeführt werden soll. Weiterhin werden Vektoren benötigt, um weitere Berechnungen vorzunehmen. Auch sind nicht alle Funktionen der PC-Version für das Smartphone verfügbar. So kann der Befehl „plot“ auf dem Smartphone nicht ausgeführt werden. Die einzige Möglichkeit Daten zu visualisieren besteht darin, ein Scope zu verwenden. Das Problem hierbei ist allerdings, dass ein Scope immer nur Daten zum derzeitigen Simulationszeitpunkt ausgibt. So kann nicht zuerst der Sollwert **W** und danach die Regelgröße **Y** geplottet werden. Die Signale werden zeitgleich vom Scope ausgegeben.

Das Simulink Modell welches erstellt wurde, um als Applikation auf dem Smartphone ausgeführt zu werden, soll in diesem Kapitel betrachtet werden. Um dem Leser ein besseres Verständnis zu ermöglichen, wird sich im Text mithilfe von Zahlen auf die Abbildungen bezogen. Als Beispiel: Wird eine Passage mit ① beendet, wird der in der Abbildung mit ① markierte Bereich in der Passage besprochen. Weiterhin empfiehlt es sich zusätzlich, das Kapitel 9.1 im Anhang zu lesen. Dort werden die verwendeten Simulink Blöcke erklärt.

3.2 Lösung der Einschränkungen des Simulink-Modells

Um dem Probanden Kontrolle über das Modell zu geben und die Abläufe des Modells zeitlich zu regeln wird sich eines einfachen Aufbaus bedient. Dieser Aufbau wird in **Abbildung 3.1** dargestellt. Hierzu soll ein „Enabled Subsystem“ nur für eine bestimme Zeitspanne aktiviert werden. Erreicht wird das indem ein Eingangssignal den Wert 1 annimmt, dieses Signal wird aufgeteilt und einmal verzögert ①. Nun treffen die Signale auf zwei logische Bedingungen. Zuerst hat die Ausgabe von dem „NOT“ ② den Wert „1“, da das Signal noch verzögert ist. Somit ist die weiter unten liegende Bedingung „AND“ ③ erfüllt, da beide eintreffende Signale den Wert „1“ haben. Somit wird das „Enabled Subsystem“ ④ aktiviert, aber auch nur solange bis das verzögerte Signal auf das „NOT“ trifft. Nun gibt der „AND“ Block den Wert „0“ aus und das Subsystem ist wieder inaktiv. Nun kann das Modell mit logischen Signalen gezielt „Enabled Subsystems“ für eine gewünschte Zeit aktivieren.

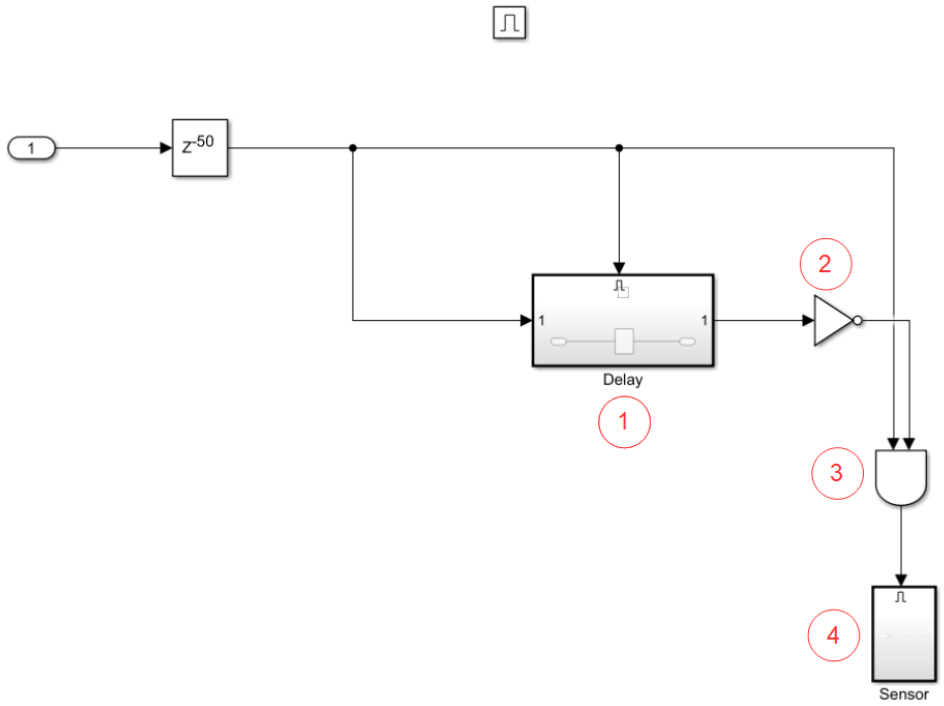


Abbildung 3.1 Delay Aufbau

Da Sollwert **W** und Regelgröße **Y** zeitgleich ausgegeben werden, wurde sich dazu entschlossen den Sollwert zweimal zu plotten. Zuerst wird dem Probanden nur der Sollwertverlauf gezeigt, währenddessen sind die Sensoren deaktiviert. So hat der Proband Zeit, sich schonmal den Sollwertverlauf einzuprägen. Nach einer kurzen Pause wird der Sollwert ein zweites Mal ausgegeben, diesmal sind die Sensoren aber aktiviert. Nun muss der Proband versuchen die Regeldifferenz so niedrig wie möglich zu halten. Genau wie ein echter Regler.

Um Signalverläufe abzuspeichern, war die einzige Lösung Signale an den PC zu übertragen und sie dort als Vektoren zu speichern. Es besteht zwar die Möglichkeit Datensätze auf dem Smartphone zu speichern, allerdings kann die Applikation, während sie ausgeführt wird nicht auf sie zugreifen. Da die Ausführung der Applikation einer Simulation am PC gleicht, können die Daten erst danach aufgearbeitet werden. Zur Ausarbeitung ist MATLAB nötig, als Lösung bleibt also nur die Daten zuerst zu übertragen.

3.3 Struktur des Simulink Modells

Die Hauptansicht des Simulink Modells ist in **Abbildung 3.2** zu sehen.

Im Bereich „App Build“ ① befindet sich der in Abschnitt 2 angesprochenen Button Block „Run“. Dieser aktiviert bei Betätigung in der Applikation die Subsyteme „Calibration“ und „Main Body“. Somit setzt dieser Button alles in Bewegung, seine Betätigung kann also als Ausführung der Applikation betrachtet werden. Im Subsystem „Main Body“ entsteht der Sollwertverlauf **W** und die Regelgröße **Y**. Diese erreichen zum Schluss die UDP Send Blöcke „W Send“ und „Y Send“. Diese sorgen dafür, dass das Smartphone beide Datensätze an den PC überträgt.

Im Bereich „Data Store Memory“ ③ befinden sich drei Data Store Memory Blöcke. Jeder einzelne speichert einen Wert ab, diese drei Werte werden in anderen Subsystemen berechnet und dann in diese Blöcke geschrieben. Der Block „Acceleration“ speichert die für den Sollwert benötigte, zufällig generierte Beschleunigung ab. Der Block „Velocity“ speichert die für den Sollwert benötigte, zufällig generierte, maximale Geschwindigkeit ab. Diese Werte werden in Kapitel 3.4.1 genauer erklärt. Schließlich speichert der Block „Calibration“ noch einen Wert ab, der in Kapitel 3.3.1 näher erläutert wird.

Im Bereich „Data Transfer“ ② werden die soeben erläuterten Werte „Acceleration“ und „Velocity“ ausgelesen und über UDP an das Modell auf dem PC übertragen. Da eine exakte Nachbildung des Sollwertprofils, das auf dem Smartphone erscheint, auf dem PC zu sehen sein soll, benötigt der PC dieselben Daten, die die Applikation für das Sollwertprofil verwendet hat.

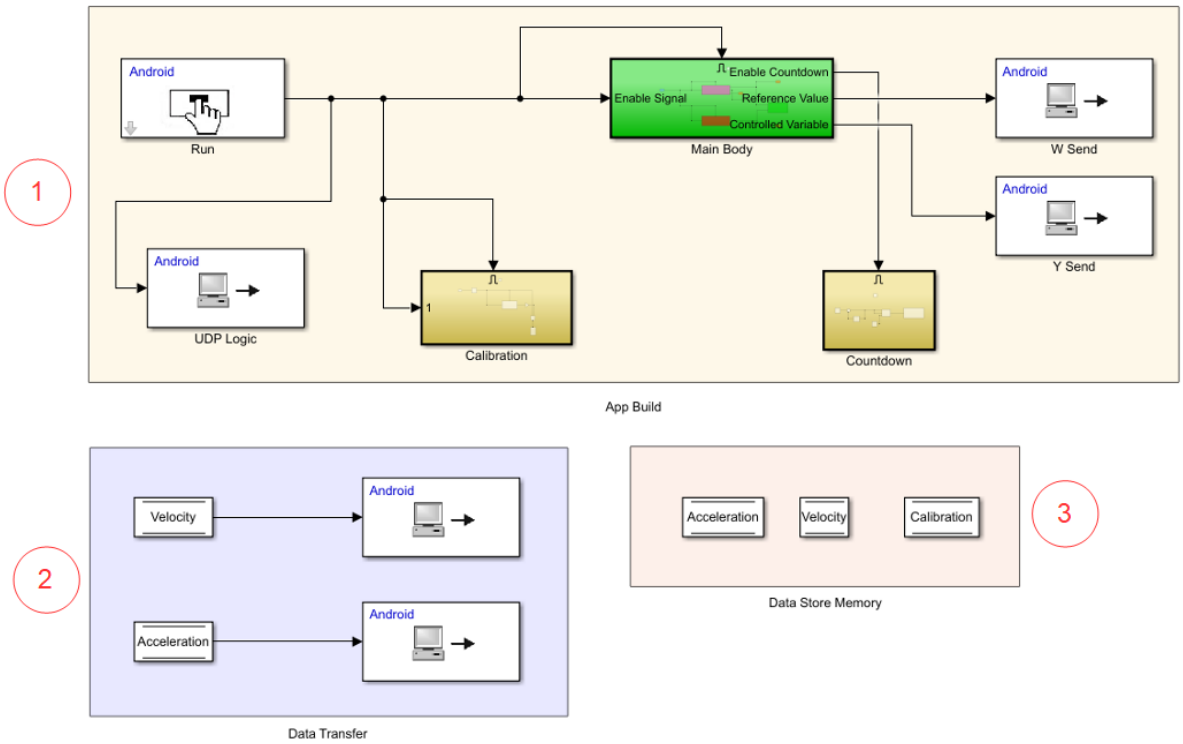


Abbildung 3.2 Hauptansicht

3.3.1 Kalibrierung des Smartphones

Da Sensoren in zwei verschiedenen Smartphones nie das Gleiche messen werden, ist es nötig zumindest eine kurze Kalibrierungsphase in den Prozess einzubauen. Ein beliebiges Smartphone etwa misst im Stillstand in y-Richtung ständig einen Wert von acc_y = - 0.09. Um diesen Fehler hinreichend zu kompensieren, wird im Subsystem „Calibration“ über einen Zeitraum von 3 Sekunden, ein Mittelwert von allen Beschleunigungen in y-Richtung bestimmt. Dieser wird später von den gemessenen Daten subtrahiert. Dies geschieht zu Beginn des Experimentes, während das Smartphone ruhig auf dem Tisch liegt. Hier wird zum ersten Mal der in Kapitel 3.2 beschriebene Aufbau, um ein „Enabled Subsystem“ nur für einen bestimmten Zeitraum zu aktivieren, benutzt. Das Subsystem „Sensor“ ist in **Abbildung 3.3** zu sehen.

Im Subsystem „Sensor“ befindet sich der Block „Accelerometer“, dieser spricht die Beschleunigungssensoren im Smartphone an. Hier werden nur die ausgegebenen y-Werte benötigt. Der Lowpass Filter lässt nur bestimmte Frequenzen durch und dämpft jene Frequenzen die über der angegebenen Grenzfrequenz liegen. So entstehen bei der Beschleunigungsmessung keine großen Schwankungen der gemessenen Werte. Nun müssen sie nur noch als „double“ ausgegeben werden und deren Mittelwert bestimmt werden. Dies übernehmen die beiden Blöcke „Cast to double“ und „Moving Average“. Der Block „Calibration“ speichert den Wert in einem „Data Store Memory“ Block ab.

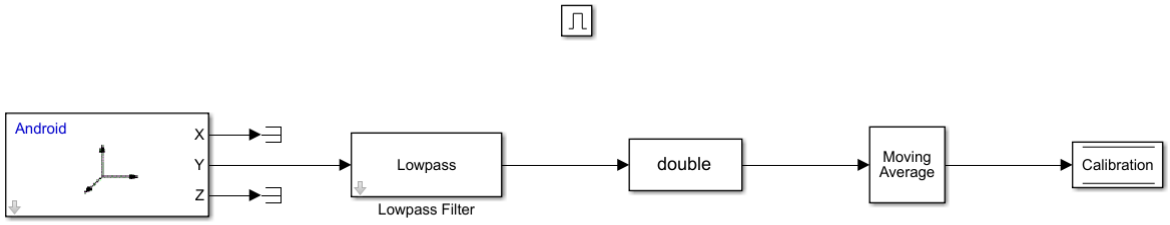


Abbildung 3.3 Subsystem Sensor

3.3.2 Erstellen eines Countdowns

Das Subsystem „Countdown“ soll, wie der Name schon sagt, in der Applikation dem Probanden einen Countdown anzeigen. Der Ablauf dieses Countdowns signalisiert dem Probanden, ab wann er das Smartphone bewegen soll. Dazu ist ein simpler Aufbau aus Delay, Sum und Switch Blöcken nötig. Im Delay Block ist die „Initial Condition“ mit „7“ und die „Delay length“ mit „1“ festgelegt. Somit wird vom Startwert 7 bis auf 0 runtergezählt, erreicht der Wert 0 wird der „Switch“ umgelegt und es wird nun konstant 0 ausgegeben. Der Block „Data Display“ erstellt einen Bereich in der Applikation, der dort den derzeitigen Wert des Countdowns ausgibt. Das Subsystem „Countdown“ ist in **Abbildung 3.4** zu sehen.

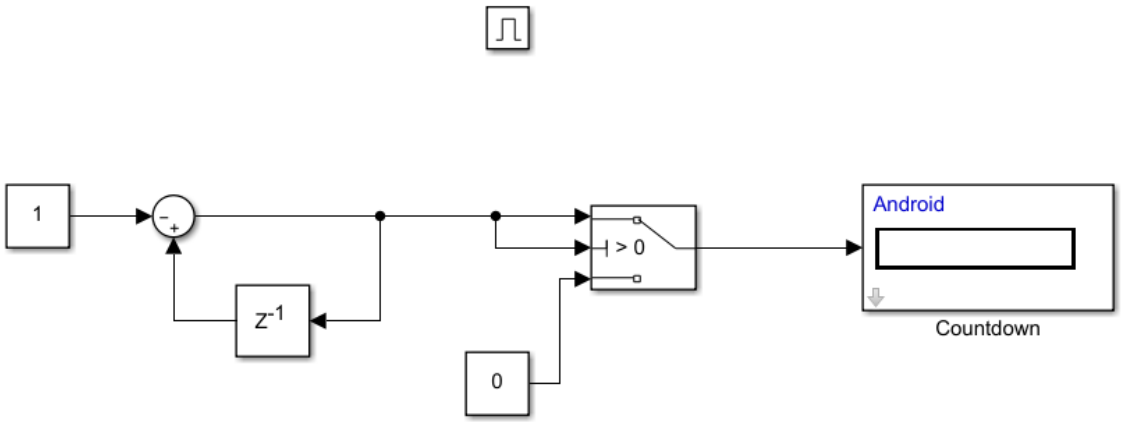


Abbildung 3.4 Subsystem Countdown

3.4 Simulieren und Messen der notwendigen Daten

Im Folgenden soll das Subsystem „Main Body“ betrachtet werden. In diesem Subsystem wird zuerst der Sollwertverlauf **W** simuliert und anschließend die Regelgröße **Y** gemessen. Die Ansicht des Subsystems ist in **Abbildung 3.5** zu sehen.

Das „Enable Signal“ teilt sich auf und aktiviert zwei Subsysteme: „Reference Value 1“ ① und „Controlled Variable 1“ ②. Im Subsystem „Reference Value 1“ entsteht der Sollwertverlauf **W**, während das Subsystem „Controlled Variable 1“ dafür zuständig ist die Regelgröße **Y** aufzuzeichnen. Das Subsystem „Scope“ ③ erhält von den beiden anderen Subsystemen Daten und plottet diese in der Applikation.

Der Ausgang „Enable Countdown“ ist dafür da, um im Bereich „App Build“ das Subsystem „Countdown“ zu aktivieren“. Die beiden Ausgänge „Reference Value“ und „Controlled Variable“, geben den Sollwert **W** und die Regelgröße **Y** an zwei UDP Send Blöcke weiter. Diese senden die Daten per UDP an das Modell auf dem PC.

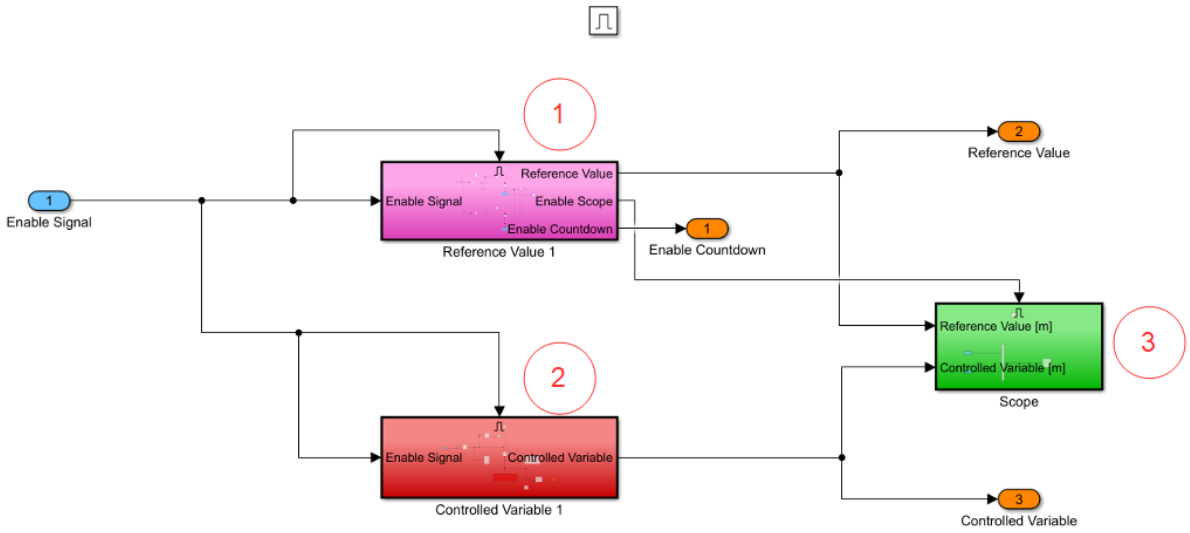


Abbildung 3.5 Subsystem Main Body

3.4.1 Entstehung des gewünschten Verlaufs der Smartphoneposition (Sollwert)

In dem Subsystem „Reference Value 1“ entsteht der Sollwert, der in der Applikation geplottet wird. Das im Folgenden besprochene Subsystem ist in **Abbildung 3.6**, **Abbildung 3.7** und **Abbildung 3.8** zu sehen.

Das Signal „Enable Signal“ mit dem Wert „1“ erreicht zuerst ein weiteres Subsystem namens „Random Values“ ①. Hier werden zwei zufällige Werte generiert, die abgespeichert werden, um sie im weiteren Verlauf zu nutzen. Ziel ist dabei, dass jedes Mal ein neuer, zufälliger Sollwertverlauf entsteht. Die zufälligen Werte repräsentieren Geschwindigkeit und Beschleunigung, da sie bei jeder neuen Ausführung einen anderen Wert annehmen, wird auch jedes Mal ein einzigartiger Sollwertverlauf entstehen. Nun teilt sich das Signal und trifft auf zwei Verzögerungen. Betrachtet wird zuerst der obere Teil. Der Delay Block ② bewirkt, dass der Sollwertverlauf erst nach 4 Sekunden geplottet wird. Dies soll der Applikation genug Zeit geben, alle nötigen Werte festzulegen, also Kalibrierung, Beschleunigung und Geschwindigkeit. Nach 4 Sekunden dann geht das Signal weiter. Der Ausgang „3“ ③ betätigt nun im Bereich „App Build“ das Subsystem „Countdown“. In der Applikation wird also ab jetzt von 7 runtergezählt. Nun trifft das Signal auf den bereits bekannten Delay Aufbau aus Kapitel 3.2 ④. Delay 1 enthält eine Signalverzögerung von 5 Sekunden. Das „NOT“ nach der Verzögerung bewirkt wieder, dass das Subsystem „Reference Value 1“ nur für 5 Sekunden betätigt ist. Auch wird das Signal zweimal abgegriffen und einmal zu einem Switch Block und zu einmal zu einem Ausgang 2 weitergeleitet ⑥. Da der Sollwert zweimal ausgegeben werden soll, befinden sich hier zwei identische Subsysteme „Reference Value 1“ und „Reference Value 2“. Der Switch ⑦ erhält die Info welches System gerade Daten generiert und leitet nur die Daten des aktiven Systems weiter. Ein inaktives System gibt kontinuierlich den Wert „0“ aus, deswegen ist es wichtig zu unterscheiden. Ausgang 2 leitet entweder den Wert „0“ oder „1“ weiter. Ist die Ausgabe gleich „0“ ist das Scope, das in der Applikation zu sehen ist, inaktiv. Sprich es werden keine Daten angezeigt. Würde es diese Kontrolle nicht geben, würde das Scope in dieser Zeit kontinuierlich den Wert 0 ausgeben. Der Wert „1“ aktiviert das Scope und es zeichnet die Daten auf, die es erhält. Da hier überprüft wird, ob eines der Subsysteme aktiviert ist, ist also das Scope nur aktiv, wenn die Sollwerte generiert werden. Schließlich aktiviert das Signal das Subsystem „Reference Value 1“ ⑤. Hier entsteht nun der Sollwertverlauf. Der Ausgang „Reference Value, gibt das simulierte Signal an überliegende Systeme aus ⑧.

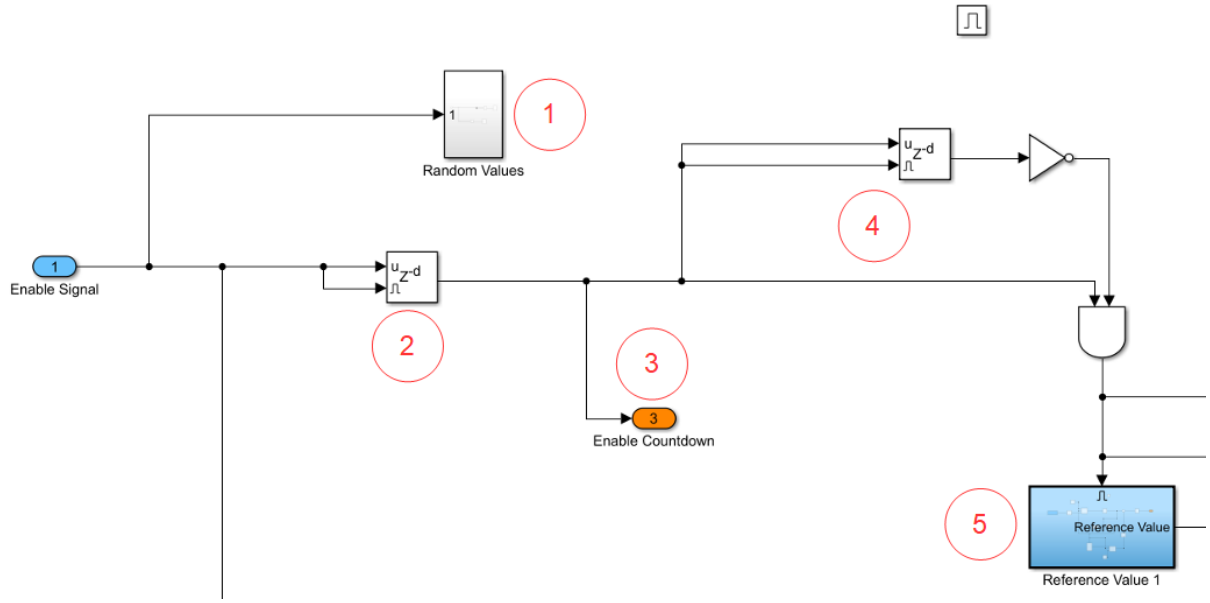


Abbildung 3.6 Subsystem Reference Value Oberer Zweig 1

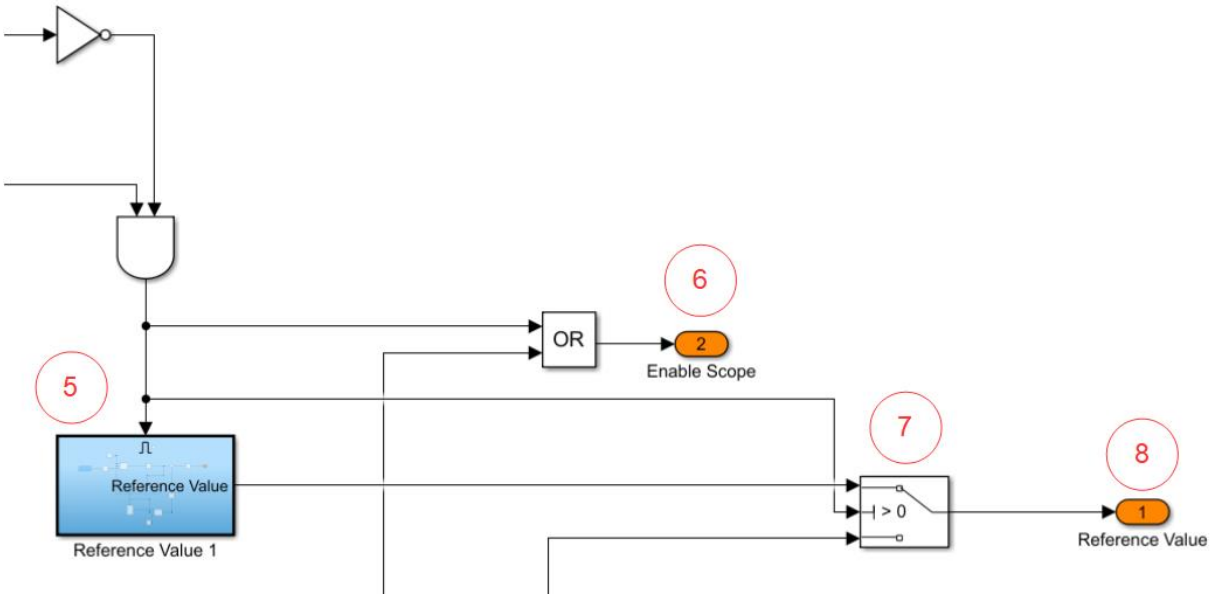


Abbildung 3.7 Subsystem Reference Value Oberer Zweig 2

Der andere Zweig funktioniert identisch, nur dass das Delay ⑨ auf 11 Sekunden gestellt ist. So entsteht auch der Wert für den Countdown. Da bei dem zweiten Plot des Sollwertverlaufs ⑩ der Proband das Smartphone bewegen soll, ist die Differenz der beiden Delays also der Anfangswert für den Countdown: Delay 2 – Delay 1 = 11 Sekunden – 4 Sekunden = 7 Sekunden. Nun werden sukzessiv die beiden Sollwertsignale mithilfe des Switches an den Scope Block weitergeleitet. In der Applikation erschient also nach 4 Sekunden der erste Sollwertverlauf und nach weiteren 7 Sekunden erscheint der zweite Sollwertverlauf.

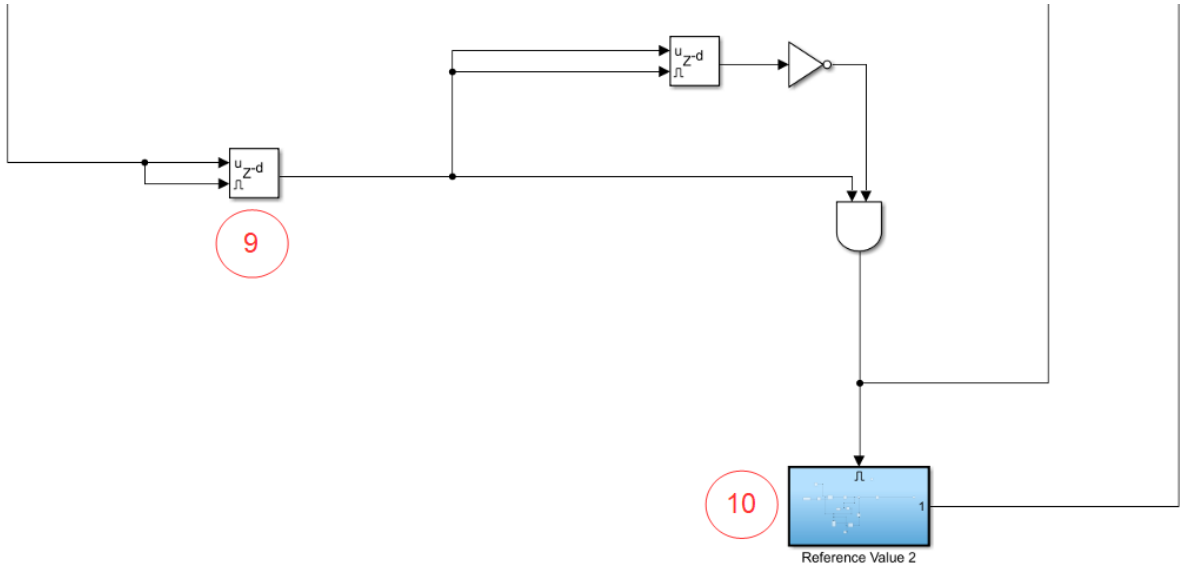


Abbildung 3.8 Subsystem Reference Value Unterer Zweig

Im folgenden Abschnitt wird nur das Subsystem „Reference Value 1“ betrachtet, da „Reference Value 2“ identisch ist. In diesem Subsystem wird der Sollwertverlauf generiert, zu sehen ist das Subsystem in **Abbildung 3.9**. Für den Sollwert wurde sich überlegt, dass er aus 5 Phasen bestehen soll: Stillstand, um dem Probanden etwas Zeit zu geben, Beschleunigung, konstante Geschwindigkeit, Abbremsen und schließlich wieder Stillstand. Um dies zu erreichen soll erst eine Beschleunigung wirken, bis eine bestimmte Geschwindigkeit erreicht wurde, nach kurzer Zeit soll dann mit dem negierten Wert der Beschleunigung gebremst werden, bis das Smartphone wieder stillsteht. Nicht zu vergessen dabei ist, dass zwei zufällige Größen vorliegen: ein Wert für die Beschleunigung und ein Wert für die maximale Geschwindigkeit, die erreicht wird. Der Block „Acceleration“ ① gibt diese zufällige Beschleunigung aus. Nun folgt nur noch eine Verzögerung von 0.5 Sekunden, also der Stillstand am Anfang des Sollwertverlaufs. Die Beschleunigung trifft nun auf einen Switch ②, der entweder die Beschleunigung weitergibt, falls der Wert der durch Integrieren der Beschleunigung ③ entsteht, also eine Geschwindigkeit, kleiner ist als der zuvor zufällig generierte Wert für maximale Geschwindigkeit. Dies wird hier ausgelesen vom Block „Velocity“ ④. Erreicht das Integral der Beschleunigung diesen Wert, ist die Bedingung erfüllt, der Switch stellt um und es wird der Wert „0“ weitergeleitet. Ist die Bedingung erfüllt wird weiterhin das darunter liegende Subsystem „Deacceleration“ ⑤ aktiviert. Dieses enthält die Beschleunigung erhalten von „Acceleration“ nur negativ. Auch hier befindet sich der bereits bekannte Switch Aufbau. Allerdings wird hier überprüft, ob die resultierende Geschwindigkeit den negativen Wert von der maximalen Geschwindigkeit unterschreitet. Ist dies der Fall, schaltet der Switch um und die Ausgabe wechselt auf konstant 0. Diese wird nun auch integriert und um 0.5 Sekunden verzögert, dies bedeutet, dass erst nach 0.5 Sekunden nach Erreichen der maximalen Geschwindigkeit gebremst wird. Das Subsystem „Deacceleration“ ist in **Abbildung 3.10** aufgezeigt. Nun werden beide Signale summiert ⑦, so entsteht ein Geschwindigkeitsverlauf, der nun nur noch integriert werden muss, um eine Strecke zu ergeben ⑧. Damit ist der Sollwertverlauf vollständig definiert und kann an das Scope in der Applikation weitergegeben werden.

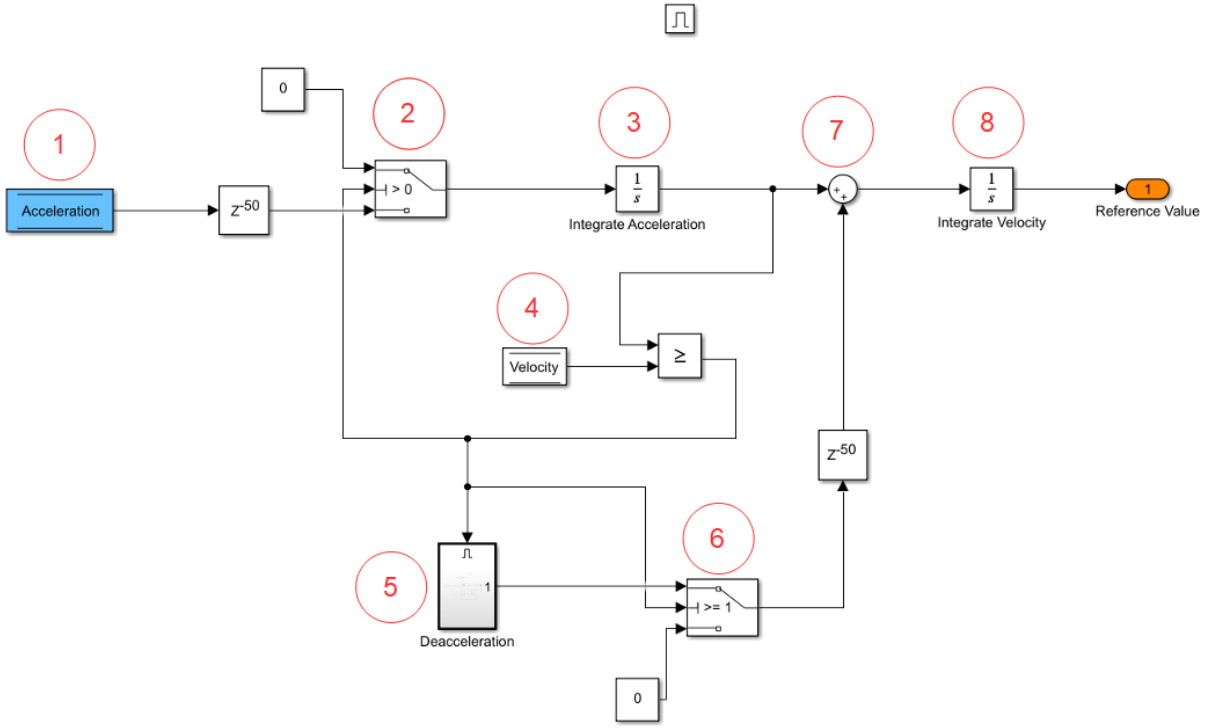


Abbildung 3.9 Subsystem Reference Value

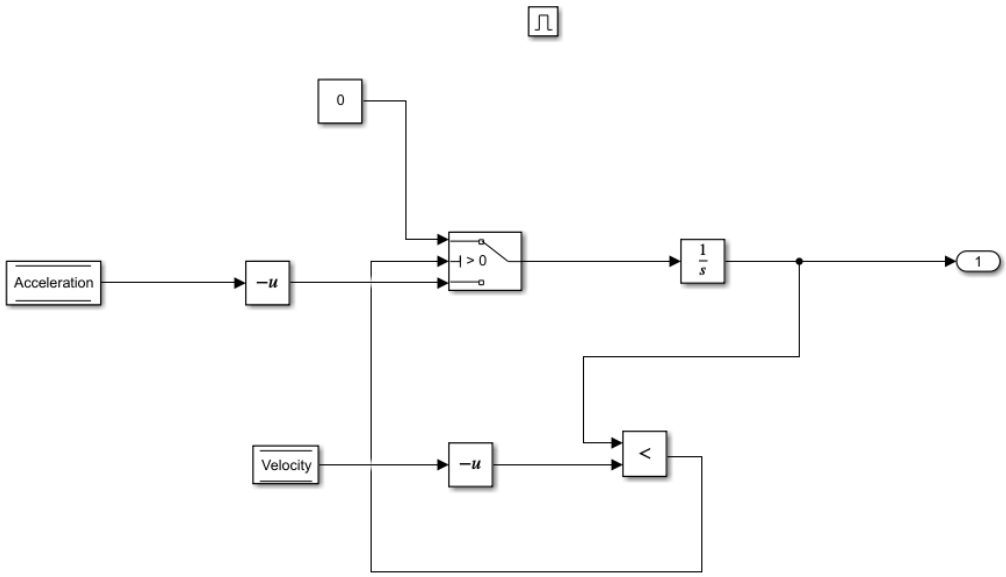


Abbildung 3.10 Subsystem Deacceleration

Aus diesem Aufbau resultieren die in **Abbildung 3.11** aufgezeigten Beschleunigungs-, Geschwindigkeits- und Streckenverläufe.

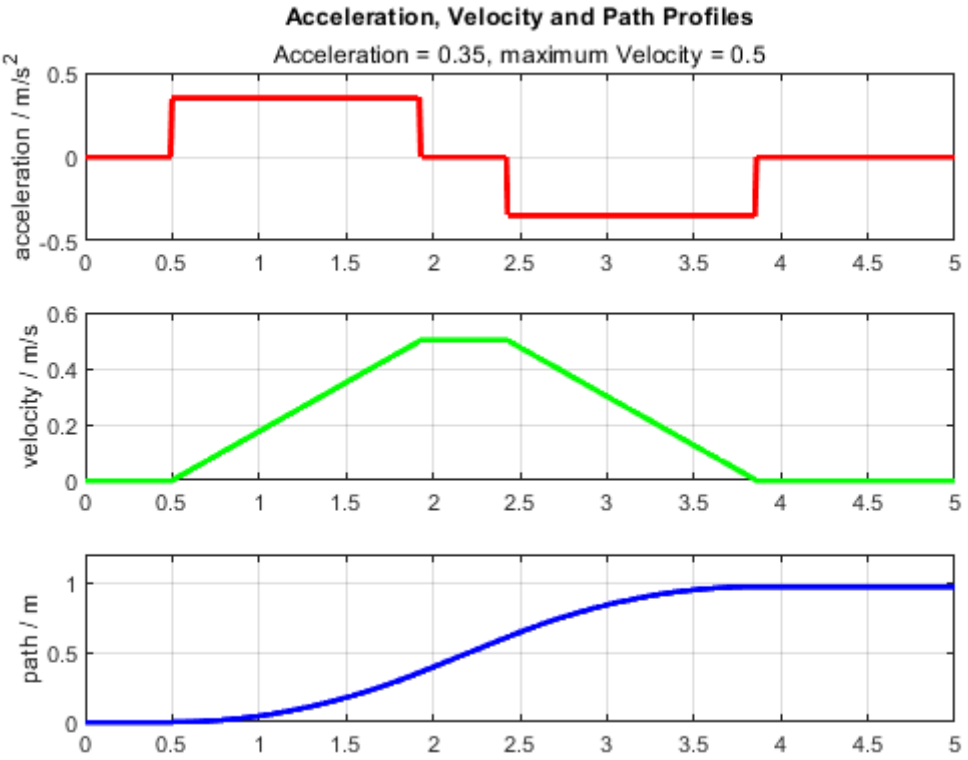


Abbildung 3.11 Verläufe

Je nach zufällig generiertem Wert für Beschleunigung und maximale Geschwindigkeit, entstehen dabei andere Verläufe. Für die obige Abbildung waren für die Beschleunigung der Wert 0.35 $\frac{m}{s^2}$ und für die maximale Geschwindigkeit der Wert 0.5 $\frac{m}{s}$ benutzt worden.

3.4.2 Tatsächlicher Verlauf der Smartphoneposition (Regelgröße)

In diesem Subsystem werden die Sensoren des Smartphones angesprochen. Die von den Sensoren gelieferten Beschleunigungsdaten werden dann in eine zurückgelegte Strecke umgerechnet. Hier entsteht also die Regelgröße **Y**. Das angesprochene Subsystem ist in **Abbildung 3.12** zu sehen.

Das Enable Signal trifft zuerst auf einen Delay von 11 Sekunden ①. Wie in Kapitel 3.4.1 beschrieben, vergehen diese 11 Sekunden bevor der zweite Sollwert geplottet wird. Da die beiden Signale simultan ausgegeben werden sollen, müssen diese 11 Sekunden zuerst vergehen, bevor die Sensoren ausgelesen werden. Nun verzweigt sich das Signal. Zum einen aktiviert es das Subsystem „Sound“ ②. In diesem befindet sich der bekannte Delay Aufbau aus Kapitel 3.2. Für eine Sekunde wird eine im Block „Audio File Read“ hinterlege wav-Datei abgespielt. Dazu werden mit dem Block „Audio Playback“ die Lautsprecher des Smartphones angesprochen. Der Sound soll dem Probanden signalisieren, dass jetzt die Sensoren angeschaltet werden und er das Smartphone bewegen soll. Weiterhin trifft das „Enable Signal“ wieder auf den bekannten Aufbau mit „NOT“ und „AND“ Blöcken ③. Diese gewährleisten, dass die Sensoren nur für genau 5 Sekunden aktiviert sind. Nun trifft das Signal auf das Subsystem „Sensor“ ④. Zuvor wird aber noch das binäre „Enable Signal“ abgegriffen und per UDP an den PC übertragen ⑤. Grund hierfür ist, dass das PC-Modell wissen muss, wann das Smartphone Daten aufzeichnet. Da das „Enable Signal“ den Sensor an- und ausschaltet ist es perfekt geeignet, um dem Modell diese Info zu liefern. Auch stellt das „Enable Signal“ einen Switch ⑥ um, sind die Sensoren aktiviert wird der im Subsystem „Sensor“ errechnete Wert ausgegeben, andernfalls der Wert „0“.

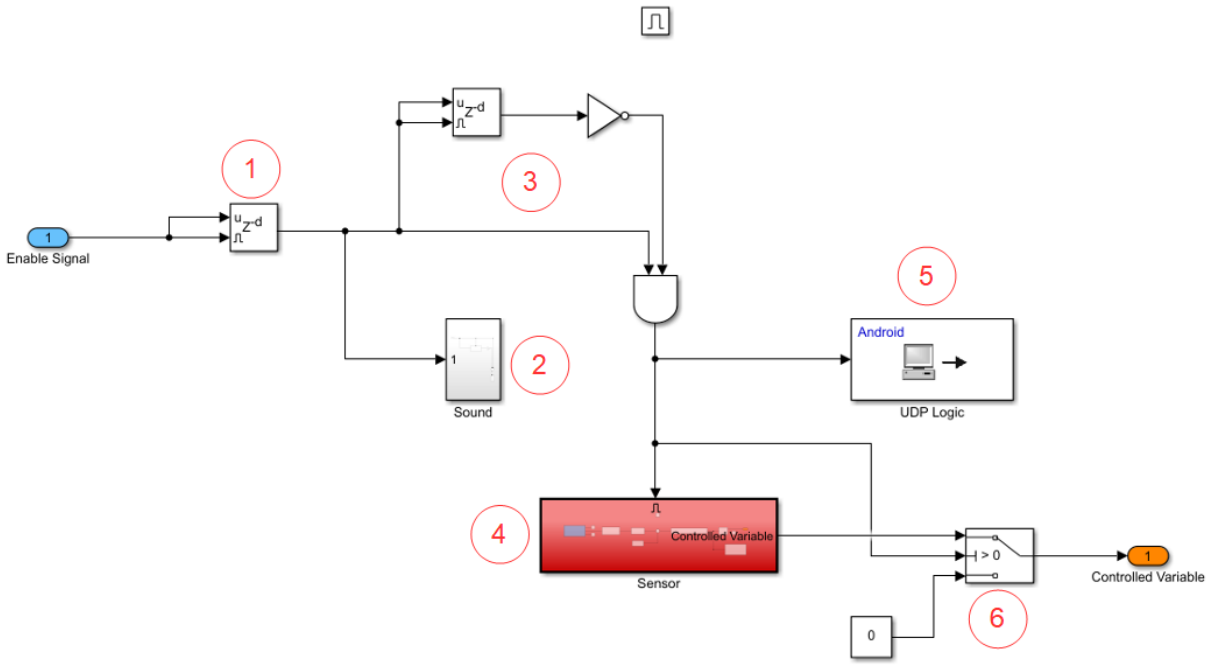


Abbildung 3.12 Subsystem Controlled Variable

Im Subsystem „Sensor“, welches in **Abbildung 3.13** zu sehen ist, werden nun die Sensoren ausgelesen. Nach Aktivierung des Subsystems gibt der Block „Accelerometer“ ① die gemessenen Beschleunigungsdaten in X-, Y- und Z-Richtung aus, da für die simple Methode nur die Daten in Y-Richtung benötigt werden, treffen die Ausgänge für X und Z nun auf Terminatoren. Die Y-Daten durchlaufen nun einen Lowpass Filter ②. Dieser glättet die Daten, so dass diese nicht so viele Ausreißer beinhalten. Der Block „Cast to Double“ wandelt die Daten in das Format Double um, da dieses bei einer Integration vorausgesetzt wird. Nun wird die Differenz aus gemessenen Daten und dem in dem Kapitel 3.3.1 berechnetet Mittelwert gebildet ③. Dieser Wert wird nun in dem Subsystem „Round Data“ gerundet ④. Schließlich wird die Beschleunigung zweifach integriert, und zwar mithilfe des Blocks „Integrator, Second order“ ⑤. Die Geschwindigkeit, die durch einmalige Integration entsteht, bei diesem Block gekennzeichnet durch den Ausgang „dx“, ist für dieses Experiment nicht von Bedeutung und endet in einem Terminator. Der Integrator berechnet nun für 5 Sekunden die zurückgelegte Strecke und gibt sie am Ausgang 1 aus. Zusätzlich wird die gemessene Beschleunigung per UDP an den PC übertragen ⑥.

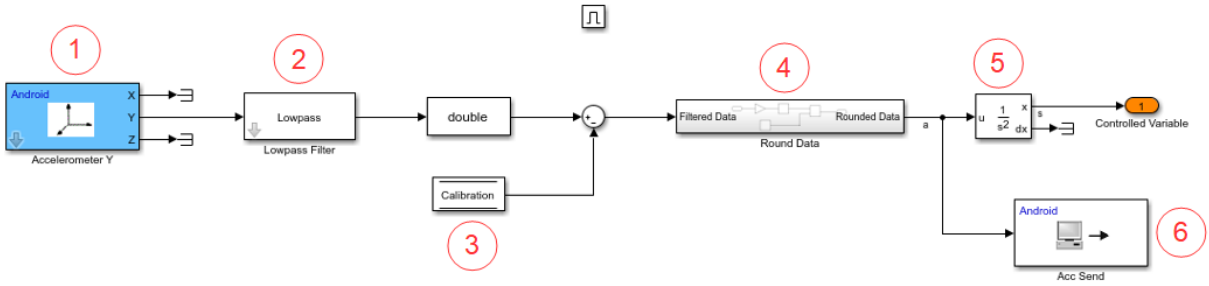


Abbildung 3.13 Subsystem Sensor

3.4.3 Steuerung der Anzeige von Sollwert und Regelgröße

Der Sollwert **W** und die Regelgröße **Y** erreichen schließlich das Subsystem „Scope“, siehe dazu **Abbildung 3.14**. Die beiden Eingänge geben die Werte an einen Mux Block ①. Dieser vereint die Daten und plottet sie weiterhin in einem Scope Block ②. Die Namen der Eingänge werden in der Applikation in der Legende für die Plots benutzt. Auch der Name des Scopes wird in der Applikation angezeigt. Hier wird der Namen als kurze Info für den Probanden zu gebraucht. Aktiviert wird dieses Subsystem zweimal, und zwar jedes Mal, wenn das Subsystem „Reference Value“ aktiviert ist. So erreicht man, dass das Scope nur dann Daten aufzeichnet, wenn dies für den Probanden von Relevanz ist. Würde man das Scope einfach so platzieren, dann würde es ab dem Zeitpunkt, ab dem die Applikation gestartet wurde, die Eingänge 1 und 2 plotten. In diesem Fall haben die beiden Eingänge die ganze Zeit den Wert „0“, solange der Button in der Applikation nicht betätigt wird. Mit den „Enabled Subsystems“ und ein paar Tricks konnte aber erreicht werden, dass das Scope vom Anwender über einen Button gesteuert werden kann.

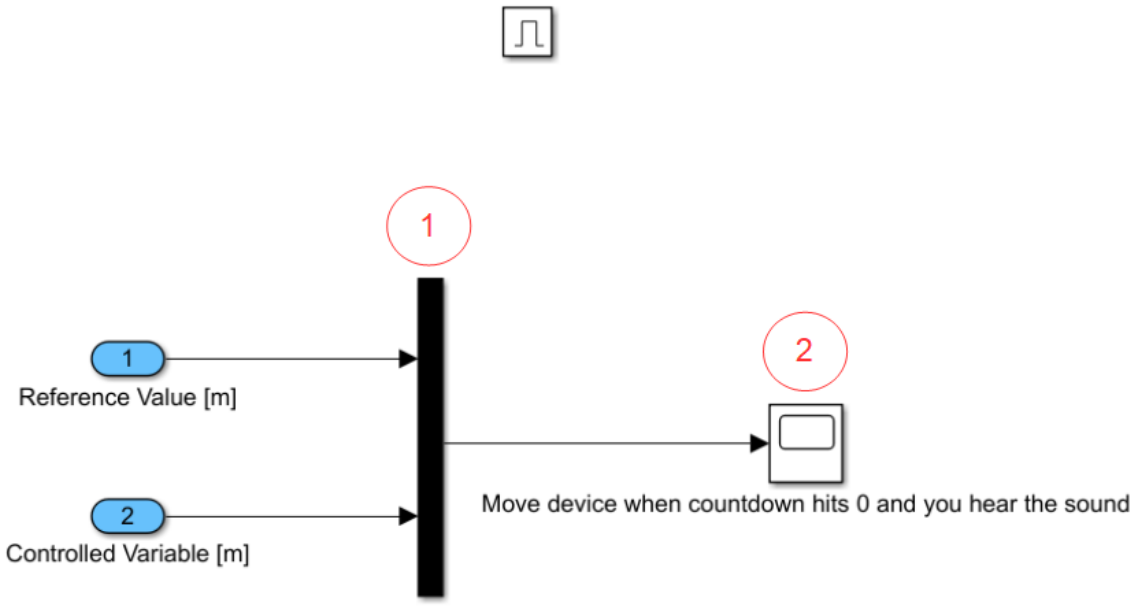


Abbildung 3.14 Subsystem Scope

3.4.4 Übertragene Daten

Um die gemessenen Daten weiterzuverarbeiten, müssen diese an den PC übertragen werden. Benötigt werden:

- Der Sollwertverlauf **W**
- Der Regelgrößenverlauf **Y**
- Die zufällig generierten Daten „Acceleration“ und „Velocity“
- Die gemessene Beschleunigung
- Logische Größen, um dem PC-Modell mitzuteilen, wann das Smartphone Daten sendet

3.5 Ausführung der Applikation auf dem Smartphone

In den folgenden Absätzen soll die fertige Applikation gezeigt und erklärt werden. Nach Installation der Applikation befindet sich ein Shortcut im Hauptmenü des Smartphones. Wird die Applikation ausgeführt, erscheint die Hauptansicht, gezeigt in **Abbildung 3.15**.

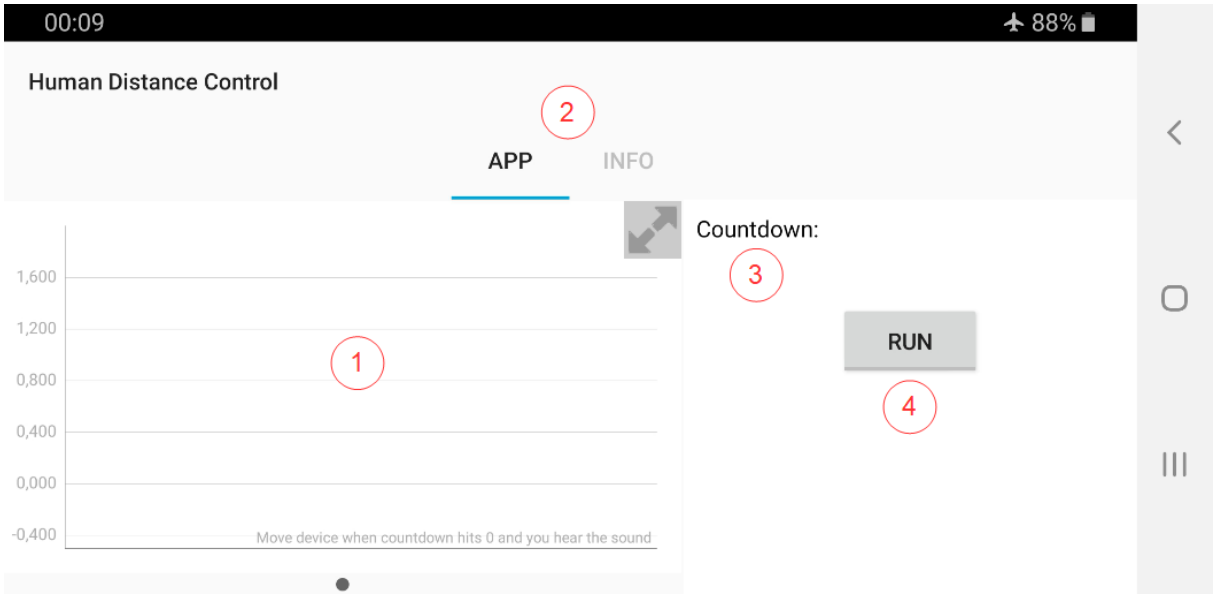


Abbildung 3.15 Hauptansicht der Applikation

Die Applikation besteht aus mehreren Objekten. Das Wichtigste Objekt ist dabei das Scope ①. Dieses gibt die Regelgröße **Y** und den Sollwertverlauf **W** aus. Im oberen Bereich der Applikation, kann der Proband zwischen den Tabs „App“ und „Info“ wechseln ②. Der Info Tab liefert zusätzliche Informationen, wie eine Bedienungsanleitung und Links zu weiteren Ressourcen. Nach dem Start der Applikation wird der Countdown gestartet und dem Probanden mitgeteilt ③. Der Button „RUN“ startet das Experiment ④. Die Applikation während und nach dem Experiment ist in **Abbildung 3.16** und **Abbildung 3.17** zu sehen.

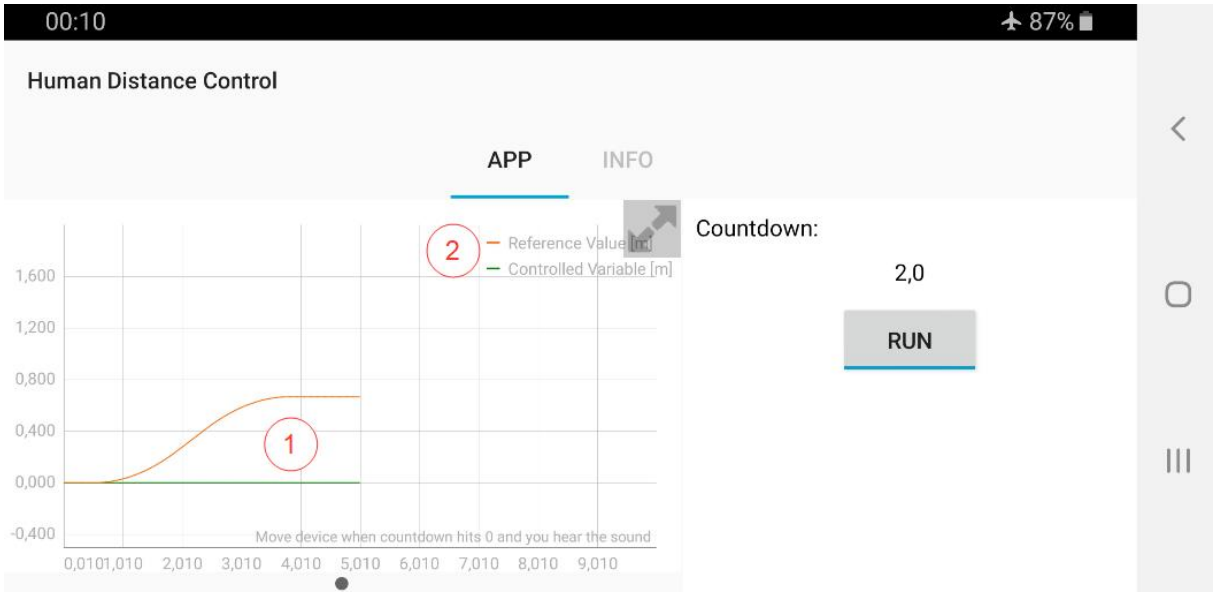


Abbildung 3.16 Applikation kurz nach Ausführung

Nach Betätigung des Buttons „RUN“ werden die Zufallsgrößen generiert und daraus wird ein zufälliger Sollwertverlauf berechnet ①. Es erscheint auch eine Legende, die dem Probanden mitteilt, welches Signal welche Farbe hat ②. Die Signale verlaufen zeitgleich.

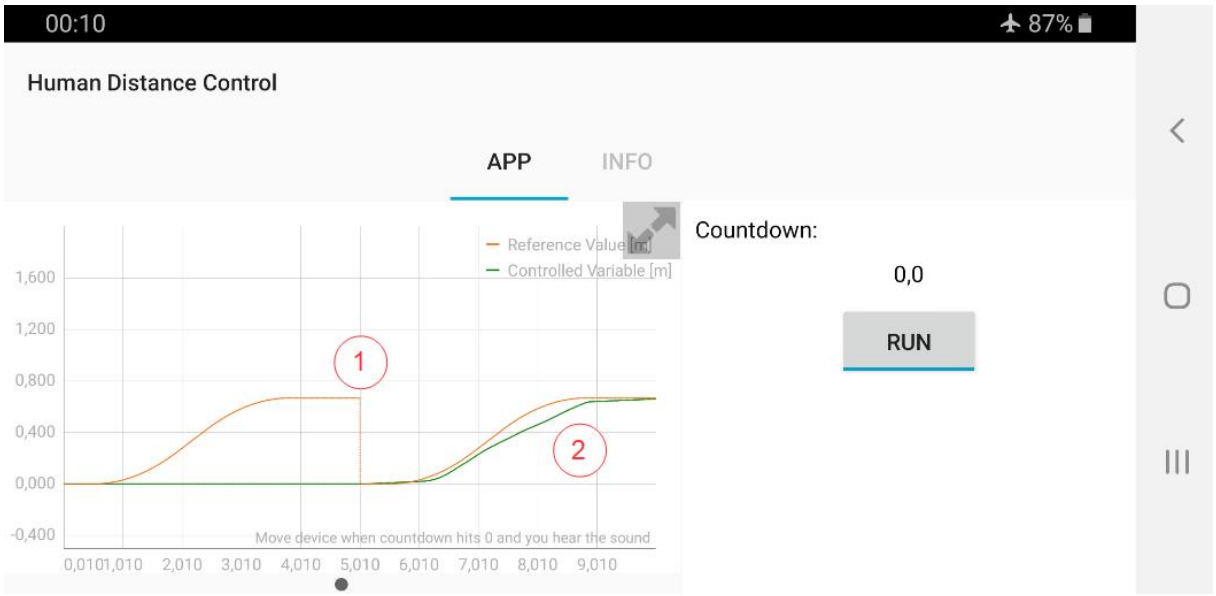


Abbildung 3.17 Applikation nach der Ausführung

Nach den ersten 5 Sekunden stoppt die Ausführung für 2 Sekunden, sodass sich der Proband den Sollwertverlauf einprägen kann ①. Nun wird der Beschleunigungssensor zusätzlich aktiviert und zeichnet die zurückgelte Distanz auf ②.

Wechselt der Proband nun auf den Infotab erscheint eine kurze Anleitung, die die Nutzung der Applikation erklärt. Auch verweisen Hyperlinks auf verschiedene Seiten, auf denen zusätzliche Dateien bezogen werden können, darunter das MATLAB Skript, das Simulink Modell und der Code für den Mikrokontroller. So kann jeder der will auch das ganze Experiment erleben. Der Infotab wurde zusätzlich mit Android Studio erstellt. **Abbildung 3.18** zeigt den Infotab.

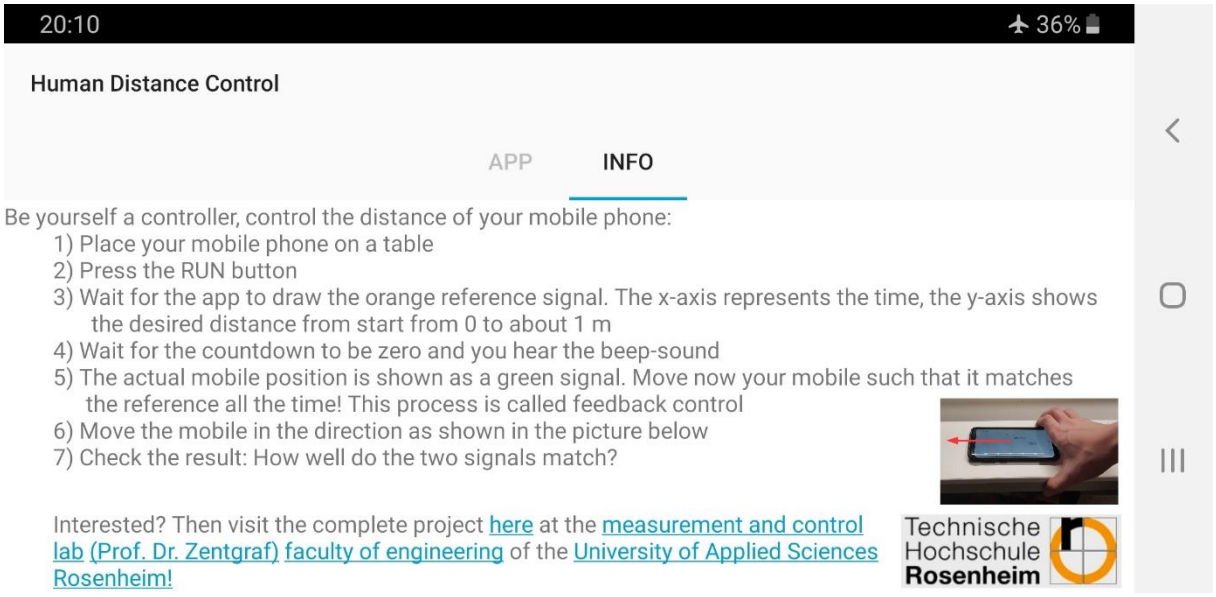


Abbildung 3.18 Infotab der Applikation

4 Datenübertragung

4.1 Bedeutung der Datenübertragung

Um die Daten von dem Smartphone auf den PC zu übertragen, wird noch eine Schnittstelle benötigt. MATLAB bietet dabei verschiedene Möglichkeiten an, es stellte sich aber schnell heraus, dass nur eine davon für dieses Experiment in Frage kommt, und zwar die Übertragung per WLAN. Die anderen Möglichkeiten wie etwa Bluetooth, funktionierten entweder nicht wie gewollt oder waren für andere Anwendungen ausgelegt. Es wird dabei vorausgesetzt, dass sich Smartphone und PC im selben Netzwerk befinden. Das Smartphone funktioniert dabei als Host, während der PC als Klient agiert. Als Netzwerkprotokoll wird das User Datagram Protocol, kurz UDP, benutzt. Nach ersten Versuchen stellte sich heraus, dass ein normales Heimnetzwerk nicht benutzt werden kann. Es ergaben sich zu hohen Latenzen, sprich die Regelgröße wurde vom PC mit zu großer Verzögerung empfangen, als dass man dies als Echtzeitübertragung ansehen konnte. Als Alternative wird ein Mikrocontroller namens Heltec ESP32 benutzt. Dieser wurde so programmiert, dass er als Access Point fungieren kann. Sobald PC und Smartphone mit dem Mikrokontroller verbunden sind, kann die Datenübertragung beginnen.

4.2 Netzwerkprotokoll UDP

Das User Datagram Protocol, auch UDP genannt, ist ein verbindungsloses Netzwerkprotokoll[4]. Dies ist von Vorteil, da man keine IP-Adressen für Sender und Empfänger festlegen muss. Stattdessen bedient sich UDP, ausschließlich der sogenannten IP-Ports. Soll also eine Größe, die in der Applikation entstanden ist, an einer bestimmten Stelle im PC-Modell auftauchen, reicht es, wenn der Block der Daten sendet und der Block der Daten empfängt, auf denselben IP-Port Parameter festgestellt sind. Da das Netzwerk zufällige IP-Adressen verteilt, müsste zum Beispiel für TCP extra die verteilte IP-Adresse ausgelesen werden und dann im Modell für PC oder Smartphone nachgetragen werden. Nachteile von UDP sind unter anderem, dass das Netzwerkprotokoll ungeschützt und ein nicht-zuverlässiges Netzwerkprotokoll ist. Da aber nur Daten in sehr kleinem Maße, übertragen werden, ist dies nicht von Bedeutung. Weiterhin sind diese Daten für Außenstehende von kompletter Bedeutungslosigkeit, also ist die Sicherheit auch nicht weiter wichtig.

[4] Wikipedia: UDP. https://de.wikipedia.org/wiki/User_Datagram_Protocol 28.02.2021 23 Uhr

4.3 Der Mikrokontroller als Access Point

Als Access Point wird ein Heltec ESP32 eingesetzt. Hierbei handelt es sich um einen Mikrokontroller mit integriertem OLED Display. Der Access Point ermöglicht eine Übertragung der Daten vom Smartphone auf den PC mit sehr geringen Latenzzeiten. Nach dem Start zeigt das Display kurz das Logo der TH-Rosenheim und führt den Benutzer durch die Anmeldung des Smartphones im Netzwerk Der Programmcode für den Microcontroller wurde in der Programmierumgebung Arduino IDE in C++ geschrieben.

```
#include "heltec.h"
#include "WiFi.h"
#include "images.h"
```

Zuerst werden die Bibliotheken geladen, diese enthalten erweiterte Befehle die nötig sind, um den Mikrocontroller einzurichten. „heltec.h“ enthält Befehle für die Ausführung des Displays, „Wifi.h“ wird benötigt um den Mikrocontroller als Access Point einzurichten und „images.h“ enthält Informationen, um das TH Rosenheim Logo anzuzeigen.

```
const char* ssid     = "DC AP";
const char* password = "12345678";
```

Hier wird Name des Netzwerks und das dazugehörige Password festgelegt. DC AP soll kurz für „Distance Control Access Point“ stehen.

```
void logo(){
  Heltec.display -> clear();
  Heltec.display -> drawXbm(0,5,logo_width,logo_height,(const unsigned char *)logo_bits);
  Heltec.display -> display();
}
```

Diese Befehlszeilen werden später aufgerufen, um bei Start des Mikrocontrollers kurz das TH Rosenheim Logo anzuzeigen. Dazu wurde das Logo als Bitmap-Datei abgespeichert. Danach wurde die Datei mit einem hex-Editor ausgelesen. Die daraus resultierenden Hexadezimalwerte wurden daraufhin in der Datei „images.h“ hinterlegt. In dieser sind auch die Werte für Höhe und Breite des Logos hinterlegt. Der Befehl „drawXbm“ bezieht aus dieser Datei also die nötigen Informationen und erstellt so das Logo der TH Rosenheim.

Nun folgt der Abschnitt void setup(), dieser enthält Befehle die einmalig, bei Start des Mikrocontrollers ausgeführt werden.

```
void setup() {

 logo();
 delay(2000);
 Heltec.display->clear();
```

Die ersten Zeilen rufen den Befehl „logo“ auf, es erscheint also für 2 Sekunden das Logo der TH Rosenheim und dann wird das Display geleert.

```
 Heltec.display -> drawString(0, 0,"SSID:");
 Heltec.display -> drawString(30, 0,(String)(ssid));
 Heltec.display -> drawString(0, 10,"PW:");
 Heltec.display -> drawString(30, 10,(String)(password));
 Heltec.display -> drawString(0, 20,"Number of devices");
 Heltec.display -> drawString(0, 30,"connected:");
 Heltec.display -> drawString(60, 30,(String)(WiFi.softAPgetStationNum()));
 Heltec.display -> drawString(0, 40,"Connect only 2 devices");
 Heltec.display -> display();
```

Diese Befehlszeilen dienen dazu, im Display interessante Informationen anzuzeigen. Die ersten zwei Zeilen geben im Display den Netzwerknamen aus, Zeile 3 und 4 zusätzlich noch das Passwort. Zeilen 5-7 geben aus, wie viele Geräte derzeit mit dem Netzwerk verbunden sind. Diese Information ist wichtig, da sich im Idealfall nur zwei Geräte verbinden sollten. Nämlich der Host und der Klient. Die Zeile 8 lässt nur einen kurzen Text erscheinen, der den Nutzer auf diesen Umstand hinweisen soll.

```
WiFi.softAP(ssid, password);
IPAddress IP = WiFi.softAPIP();
server.begin();
```

Die letzten 3 Zeilen richten nun den Access Point ein. Der Mikrocontroller bezieht die Informationen für SSID und Passwort aus den zuvor eingerichteten Strings. Die zweite Zeile weist dem

Mikrocontroller nun eine IP-Adresse zu. Die letzte Zeile startet einen Webserver, mit dem sich die Geräte kurz verbinden. Von diesem Umstand wird Gebrauch gemacht, um die Information zu erhalten, wenn sich ein neues Gerät mit dem Mikrocontroller verbindet.

Nun folgt der Abschnitt void loop(), diese Befehlszeilen werden nach Start des Mikrocontrollers immer wieder aufs Neue ausgeführt.

```
void loop() {
WiFiClient client = server.available();
```

Die erste Zeile lässt den Webserver auf neue Klienten warten.

```
 if (client) {
    Heltec.display -> drawString(0, 50,"Client connected");
    Heltec.display -> display();
    delay(2000);
    client.stop();
     }
```

Sobald sich ein neuer Klient verbindet erscheint auf dem Display kurz die Meldung: „Client connected“. Daraufhin wird der Klient vom Webserver getrennt, bleibt aber mit dem Mikrocontroller verbunden.

```
  if (WiFi.softAPgetStationNum() == 2) {
    Heltec.display -> clear();
      Heltec.display -> drawString(0, 0,"SSID:");
      Heltec.display -> drawString(30, 0,(String)(ssid));
      Heltec.display -> drawString(0, 10,"PW:");
      Heltec.display -> drawString(30, 10,(String)(password));
      Heltec.display -> drawString(0, 20,"Number of devices");
      Heltec.display -> drawString(0, 30,"connected:");
      Heltec.display -> drawString(60, 30,(String)(WiFi.softAPgetStationNum()));
      Heltec.display -> drawString(0, 40,"Ready for start");
      Heltec.display -> display();
```

```
    client.stop();
  }
```

Dieser Abschnitt bezieht zuerst die Nummer der verbundenen Geräte, sollten genau zwei verbunden sein, dann wird dieser Abschnitt ausgeführt. Die Zeilen 1-8 sorgen dafür, dass die üblichen Informationen, also Netzwerkname und Passwort, auf dem Display angezeigt werden. Die Zeile 9 gibt dann schließlich die Anzahl der verbundenen Geräte aus. Zeile 10 lässt dann auf dem Display erscheinen: „Ready for Start“. So wird dem Nutzer mitgeteilt, dass genau die richtige Anzahl an Geräten sich im Netzwerk befindet und mit dem Experiment begonnen werden kann.

```
else {
  Heltec.display -> clear();
    Heltec.display -> drawString(0, 0,"SSID:");
    Heltec.display -> drawString(30, 0,(String)(ssid));
    Heltec.display -> drawString(0, 10,"PW:");
    Heltec.display -> drawString(30, 10,(String)(password));
    Heltec.display -> drawString(0, 20,"Number of devices");
    Heltec.display -> drawString(0, 30,"connected:");
    Heltec.display -> drawString(60, 30,(String)(WiFi.softAPgetStationNum()));
    Heltec.display -> drawString(0, 40,"Connect only 2 devices");
    Heltec.display -> display();
    client.stop();
```

Dieser Abschnitt wird ausgeführt, falls eben weniger oder mehr als zwei Geräte verbunden sind. In diesem Fall werden wie gewohnt alle anderen Informationen angezeigt. Allerdings erscheint diesmal auf dem Display die Aussage: „Connect only 2 devices“. So wird dem Nutzer mitgeteilt, dass sich zurzeit zu wenige oder zu viele Geräte im Netzwerk befinden. So weiß der Nutzer, dass das Experiment noch nicht ausgeführt werden sollte.

Der obige Code muss nur noch kompiliert und auf den Mikrocontroller aufgespielt werden. Sobald der Mikrocontroller angeschaltet wurde, beginnt die Ausführung des Codes so wie oben erklärt. Die nötigen Geräte können sich mit dem bereitgestellten Passwort in das Netzwerk einwählen. Sobald zwei Geräte verbunden sind kann mit dem Experiment begonnen werden.

4.4 Der Mikrocontroller im Einsatz

In **Abbildung 4.1** wird der Mikrokontroller kurz nach dem Start gezeigt. In **Abbildung 4.2** ist der Mikrokontroller einsatzbereit.

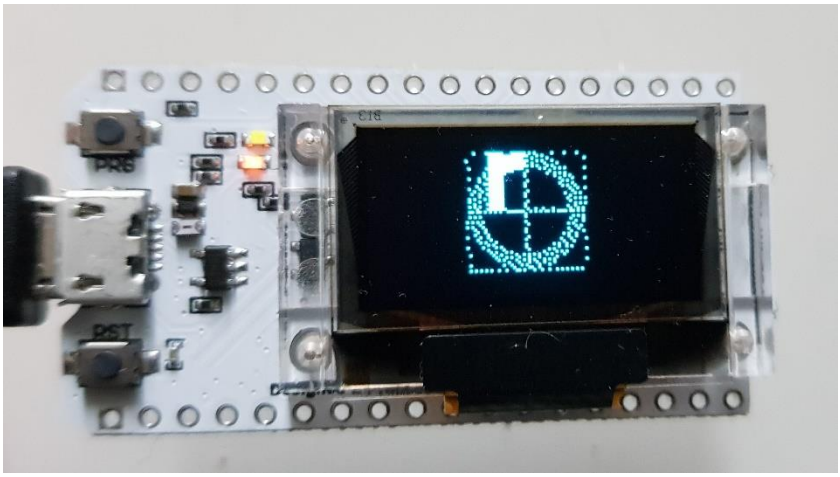


Abbildung 4.1 Mikrocontroller Start

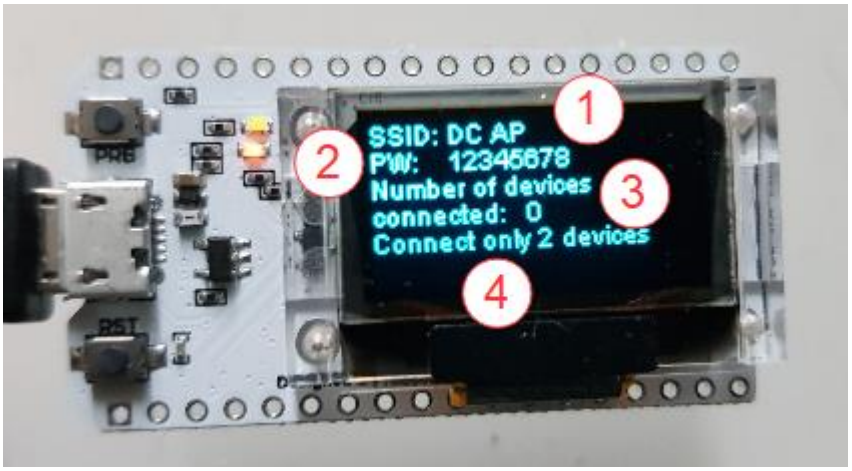


Abbildung 4.2 Angeschalteter Mikrocontroller

① Netzwerkname des Mikrocontrollers

② Passwort des Netzwerkes

③ Anzahl der verbundenen Geräte

④ Aufforderung nur zwei Geräte zu verbinden

5 PC-Modell zur Darstellung und Verarbeitung der Messdaten

5.1 Voreinstellungen des PC-Modells

Das PC-Modell wurde auch vollständig in Simulink erstellt. Mithilfe von UDP-Blöcken, bezieht das Modell Daten von dem Smartphone und verarbeitet diese weiter. Gestartet wird das Modell durch ein MATLAB Skript. Mithilfe von Voreinstellungen wurde das Modell soweit verlangsamt, dass es in Echtzeit abläuft. Dazu wurde in den Simulationseinstellungen das sogenannte „Pacing“ aktiviert und so eingestellt, dass die Simulation in Echtzeit abläuft. So läuft die Simulation nicht in einem Bruchteil einer Sekunde ab. Dies ermöglicht den Empfang von Echtzeitdaten, sowie den Echtzeitplot der Regelgröße **Y**. Die Einstellung ist in **Abbildung 5.1** zu sehen.

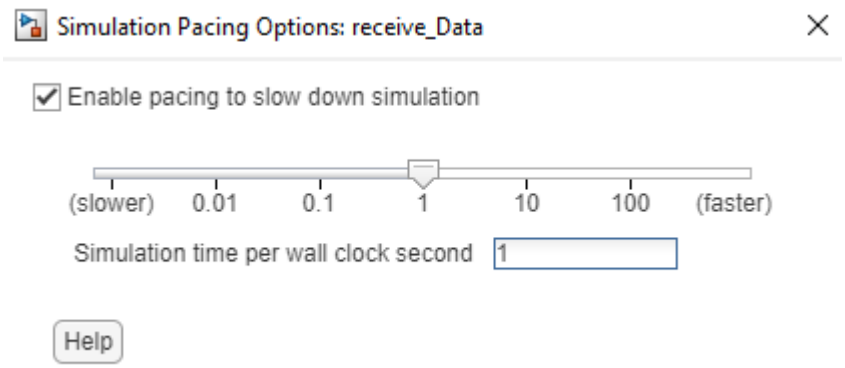


Abbildung 5.1 Pacing Optionen

5.2 Struktur des PC-Modells

Die Hauptansicht des Simulink Modells für den PC ist in **Abbildung 5.2** zu sehen.

Im Bereich „Receive Data“ befinden sich 2 „UDP Receive“ Blöcke. Diese erhalten Daten vom Smartphone und speichern diese in „Data Store Write Blöcken“ ab. Bei den Daten handelt es sich um die Größen „Acceleration“ und „Velocity“, die die Applikation verwendet hat, um den Sollwertverlauf zu erstellen. Die Daten werden benötigt, um denselben Sollwertverlauf am PC wie am Smartphone zu erstellen. Da der Size Output wieder nicht benötigt wird, endet diese Info in Terminatoren. ①

Die beiden Größen „Acceleration“ und „Velocity“ werden im Bereich „Data Store Memory“ abgespeichert. ②

Im Bereich „Plot Reference Value“ entsteht der Sollwertverlauf. Der Block „UDP Logic 1“ gibt den Wert „1“ aus, sobald der Button „Run“ in der Applikation betätigt wurde. Nach einer Verzögerung von 4 Sekunden wird dann das Subsystem „Plot Reference Value“ aktiviert. In diesem befindet sich eine MATLAB Function, die mit den Werten aus dem Bereich „Receive Data“ den Sollwertverlauf in eine MATLAB Figure plottet. ③

Im Bereich „Plot real time data” entsteht in Echtzeit der Verlauf der Regelgröße **Y**. Der UDP Receive Block „UDP Data Y“ erhält vom Smartphone die berechnete, zurückgelegte Strecke. Der UDP Receive Block „UDP Logic 2“ gibt den Wert „1“ aus, sobald das Smartphone beginnt Daten aufzuzeichnen, sprich sobald die Sensoren aktiviert wurden. So wird das Subsystem erst aktiviert, wenn die Applikation soweit ist. In dem Subsystem befindet sich eine Level-2 MATLAB S-Function. Diese plottet die erhaltene Regelgröße **Y** in dieselbe MATLAB Figure, die die Function aus dem Bereich „Plot Reference Value“ erstellt hat. Dies geschieht in Echtzeit. Also derselbe Regelgrößenverlauf, der auf dem Smartphonebildschirm ausgegeben wird, wird simultan in einer Figure auf dem PC geplottet. ④

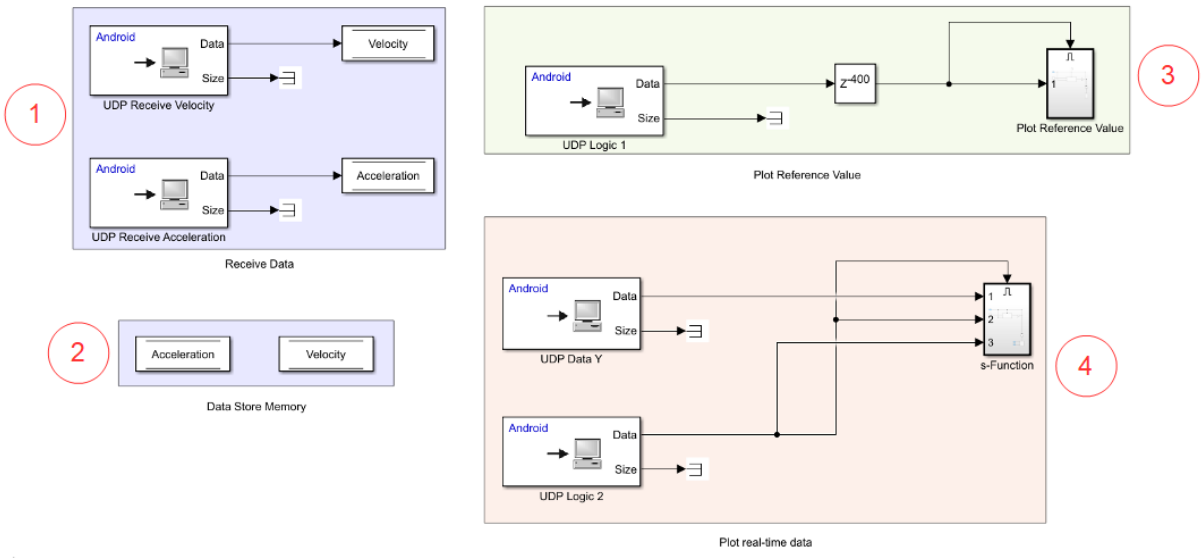


Abbildung 5.2 Hauptansicht PC-Modell

5.2.1 Plot des Sollwertes vor Beginn des Experimentes

Im Subsystem „Plot Reference Value“ wird der Sollwertverlauf, den die Applikation generiert hat, nachgebildet. Zu sehen ist das Subsystem in **Abbildung 5.3**.

In diesem Subsystem befindet sich der bereits bekannte Aufbau, um Subsysteme nur kurz zu aktivieren. Nach dem das Eingangssignal auf „1“ schaltet, wird ein Delay von 0.1 Sekunden aktiviert. Noch ist die Bedingung „AND“ erfüllt, da das Signal um 0.1 Sekunden verzögert wurde und NOT noch „1“ ausgibt. Das Subsystem „Function Plot“ wird also aktiviert. Nach diesen 0.1 Sekunden allerdings, ist die Bedingung nicht mehr erfüllt, da das verzögerte Signal auf den NOT Block trifft und dieser jetzt „0“ ausgibt. Die AND Bedingung ist also nicht mehr erfüllt und das Subsystem wird wieder abgeschaltet. Grund hierfür ist, dass die MATLAB Function, die sich im Subsystem „Function Plot“ befindet, einer Sample Time unterliegt. Das bedeutet für das Modell, dass diese Function entsprechend der Sample Time, über die Gänze der Simulation, neu ausgeführt wird. Da der Plot des Sollwertverlaufes aber nicht aktualisiert werden soll, reicht es aus, wenn die Funktion nur einmal angesprochen wird. Deswegen ist die Zeit, in der das Subsystem aktiv ist, auch so gering. Des Weiteren würde die Simulationsausführung deutlich darunter leiden, wenn der Sollwert die ganze Zeit neu berechnet und in eine Figure geplottet werden sollte.

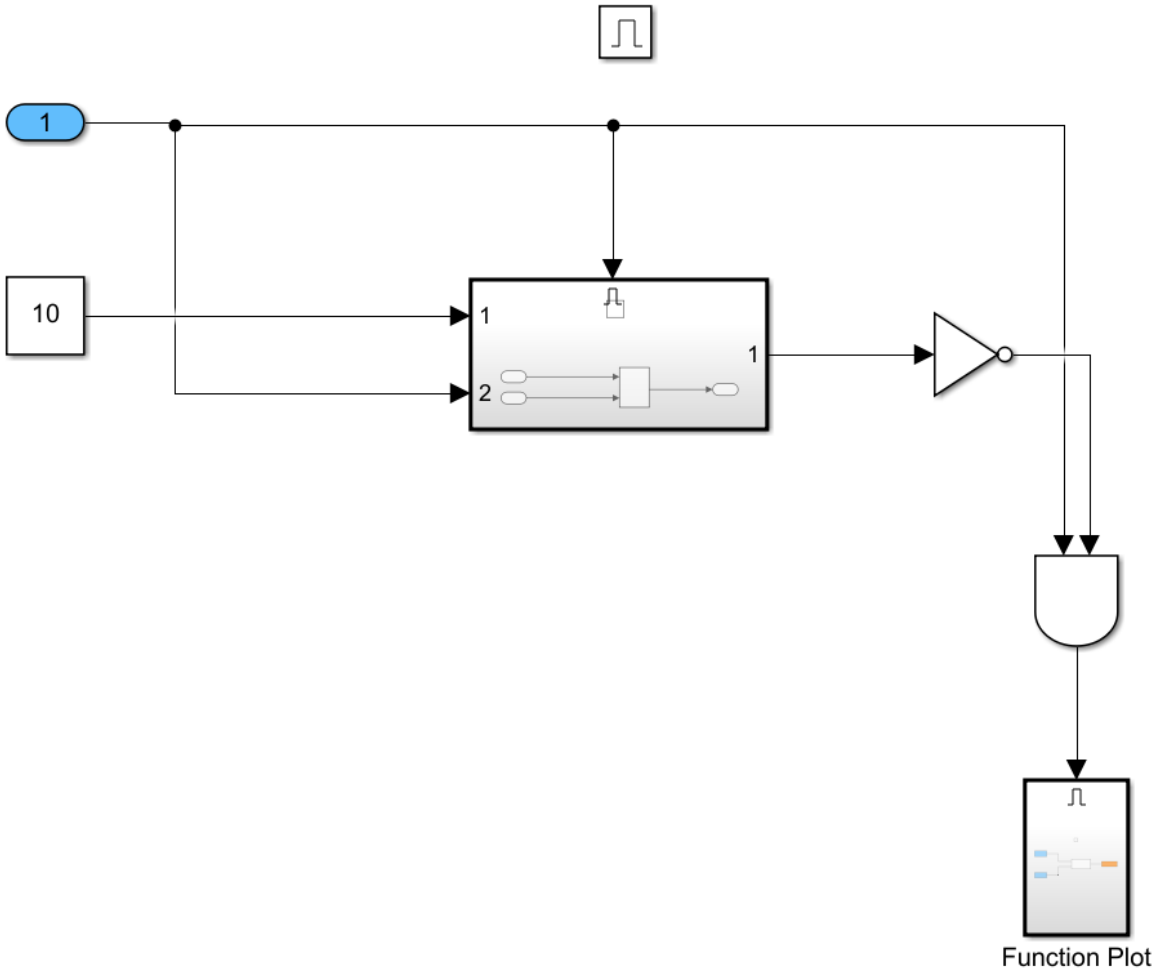


Abbildung 5.3 Subsystem Plot Reference Value

In dem Subsystem „Function Plot“ befindet sich die MATLAB Function plot_reference [9.2.6], die den Sollwertverlauf berechnet und plottet. Dieses Subsystem ist in **Abbildung 5.4** zu sehen. Dazu bezieht sie die zwei Größen, die das Smartphone liefert, aus den Data Store Read Blöcken „Acceleration“ und „Velocity“. In diesem Fall ist „Acceleration“ die Beschleunigung, mit der das Smartphone beschleunigt bzw. abgebremst werden soll. „Velocity“ ist dabei die maximale Geschwindigkeit, die bei dem Vorgang erreicht werden soll. Der berechnete Sollwertverlauf, wird daraufhin von der Function ausgeben und im Workspace von MATLAB unter dem Namen „ref“ abgespeichert. Da die beiden Eingangsgrößen „Acceleration“ und „Velocity“ schon vor Beginn des Experimentes vom Smartphone an den PC übertragen werden, wird der Sollwertverlauf sehr frühzeitig berechnet und dargestellt. Der Proband sieht den Verlauf schon einige Zeit, bevor er das Smartphone bewegen soll.

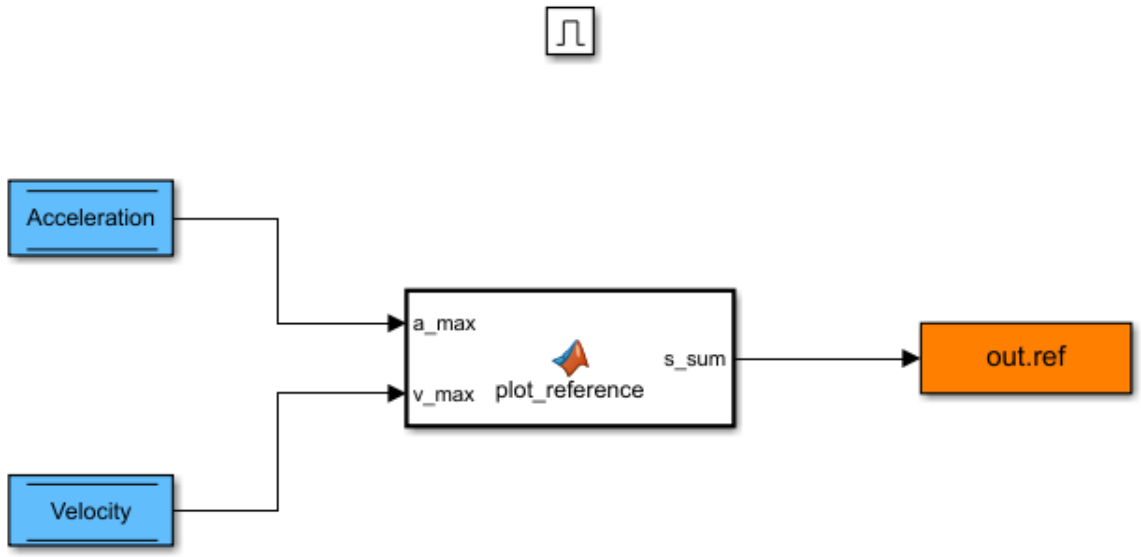


Abbildung 5.4 Subsystem Function Plot

Nun soll kurz erläutert werden, wie die Function den Sollwert berechnet.

function s_sum = plot_reference(a_max, v_max)

Ein- und Ausgabe der Function, “a_max” erhält die Function von dem Data Store Read Block „Acceleration “ und “v_max” von dem Data Store Read Block „Velocity”. Ausgegeben wird der Sollwertverlauf unter s_sum.

Zuerst legt die Function die Anfangsbedingungen fest. Diese erhält sie von den Funktionseingängen „a_max“ und „v_max“. Auch wird der Zeitvektor bestimmt, dieser geht von 0 bis 5 Sekunden mit einer Schrittweite von 0.01 Sekunden. Diese Schrittweite entspricht der Abtastzeit der Sensoren des Smartphones. So wird gewährleistet, dass bei späterer Verrechnung von Simulationsdaten und echten Daten die Vektoren auch dieselbe Größe haben. Um dies auch für die Beschleunigungen zu erreichen, wird zuerst ein Vektor derselben Größe wie der Zeitvektor angelegt und mit Nullen gefüllt, dann wird der Wert von „a_max“ für den Beschleunigungsvorgang addiert und für den Abbremsvorgang subtrahiert. Der Eingang v_max wird zusätzlich negiert. Dieser negative Wert ist nötig, um den Abbremsvorgang zu definieren. Die Geschwindigkeitsvektoren entstehen nach der bekannten Formel, Strecke multipliziert mit Zeit. Dabei entstehen die Graphen abgebildet in **Abbildung 5.5** und **Abbildung 5.6**.

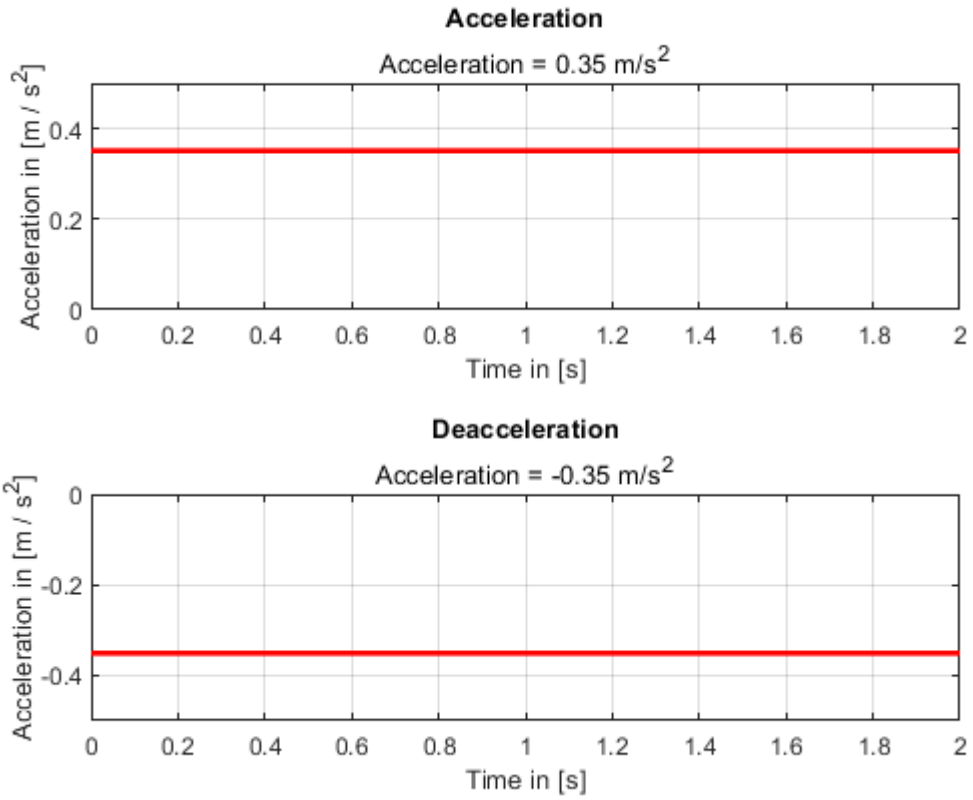


Abbildung 5.5 Anfänglicher Beschleunigungsverlauf

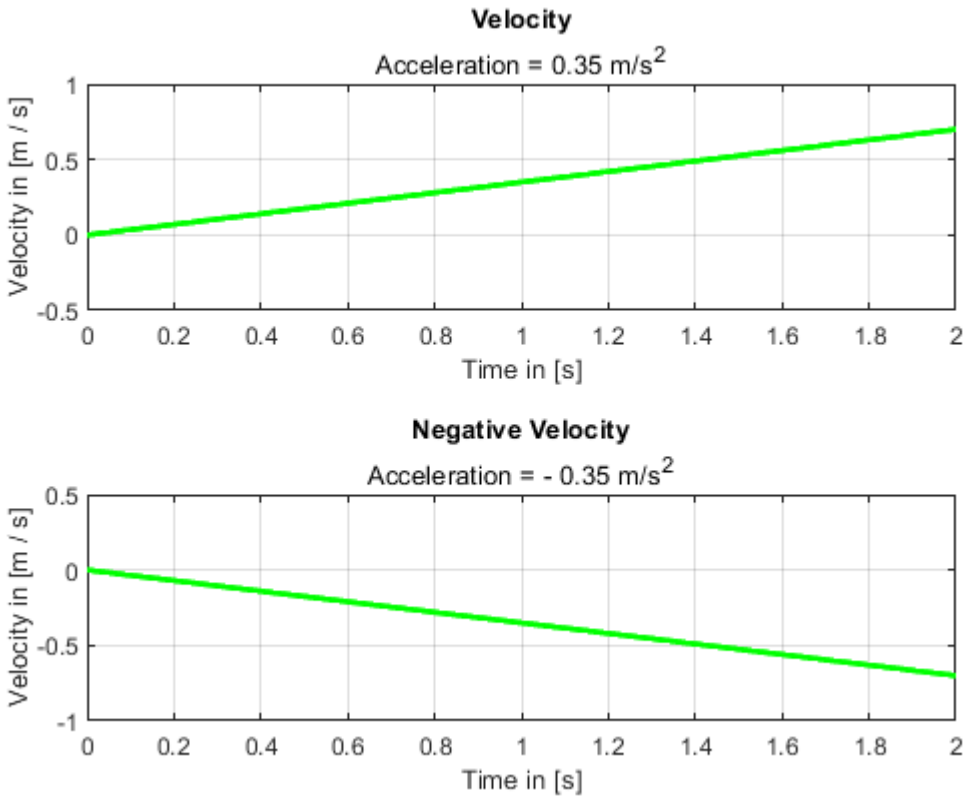


Abbildung 5.6 Anfänglicher Geschwindigkeitsverlauf

Nun werden die Beschleunigungsvektoren angepasst. Wie im Simulink Modell, wird nur solange beschleunigt, bis eine gewisse Geschwindigkeit erreicht wird. Entsprechend werden also Beschleunigungs- und Geschwindigkeitsvektor in eine for-Schleife gehängt. Sobald der Geschwindigkeitsvektor an einer Stelle, die maximale Geschwindigkeit (v_max) überschreitet, wird an derselben und an allen nachfolgenden Stellen des Beschleunigungsvektors, die Beschleunigung zu 0. Dasselbe geschieht auch mit dem Beschleunigungsvektor für den Bremsvorgang, nur das hier verglichen wird, ob die zuvor negierte Geschwindigkeit unterschritten wird. Es folgen die angepassten Beschleunigungsvektoren. Diese werden nun integriert, um die zugehörigen Geschwindigkeitsvektoren zu erhalten. Das Ergebnis ist in **Abbildung 5.7** und **Abbildung 5.8** zu sehen.

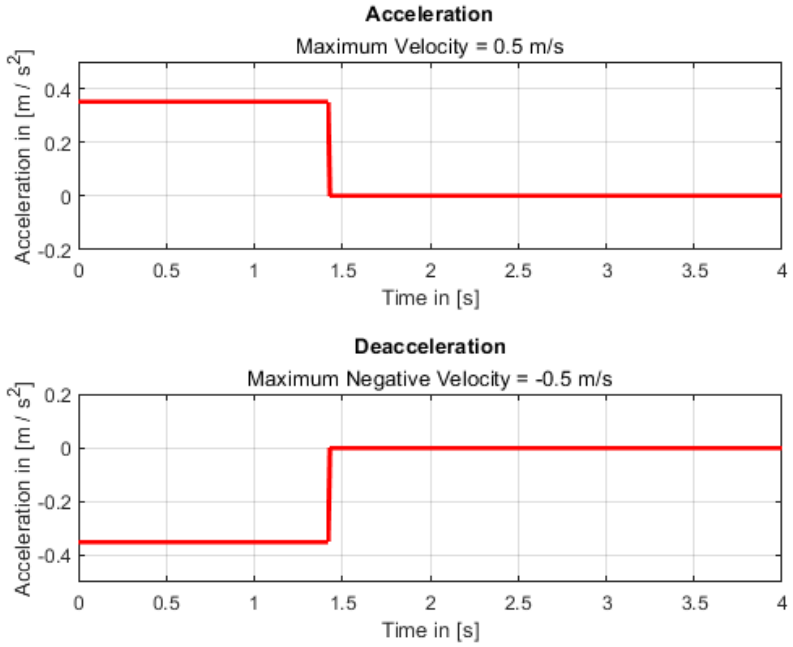


Abbildung 5.7 Begrenzter Beschleunigungsverlauf

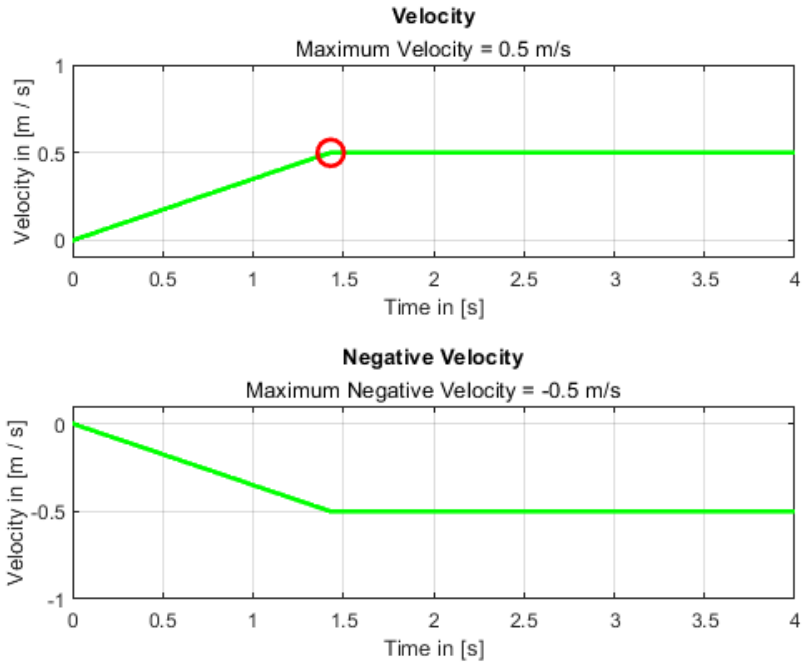


Abbildung 5.8 Begrenzter Geschwindigkeitsverlauf

Wie zu erkennen ist, überschreiten die Geschwindigkeitsverläufe nie den zuvor festgelegten Maximalwert bzw. Minimalwert. Als Nächstes wird ein Indexwert ermittelt, dieser repräsentiert die Stelle im Vektor an der das Erste Mal die Höchstgeschwindigkeit erreicht wird. Dieser Wert wird benötigt, um den Geschwindigkeitsanteil für den Bremsvorgang zu verzögern. Der Indexwert wurde in **Abbildung 5.8** mit einem roten Kreis markiert.

Um den finale Geschwindigkeitsvektor zu erstellen muss der Geschwindigkeitsvektor für den Bremsvorgang noch verzögert werden. Dafür wird der Wert d0 berechnet, dieser setzt sich zusammen aus dem Index Wert und 50, da 50/100 = 0.5 Sekunden entspricht. Die 0.5 Sekunden wurden zuvor im Simulink Modell für das Smartphone festgelegt. In diesem Beispiel berechnet sich d0 zu d0 = 50 +144 = 194. Das bedeutet, dass nach 1.44 Sekunden die maximale Geschwindigkeit erreicht wurde und dass nach 1.94 Sekunden gebremst wird. Der Geschwindigkeitsvektor wird also mit Nullen gefüllt, und zwar von 1 bis d0. Danach werden die originalen Werte eingefüllt, da der Vektor aber die gleiche Größe behalten muss, wird noch die Länge des originalen Vektors um Indexwert gekürzt. In **Abbildung 5.9** sind 2 Graphen zu sehen. Der obere Plot zeigt den verzögerten, negativen Geschwindigkeitsverlauf. Der untere Plot stellt die Summe der beiden Geschwindigkeitsverläufe dar. Integriert man diese Summe, erhält man den Sollwertverlauf.

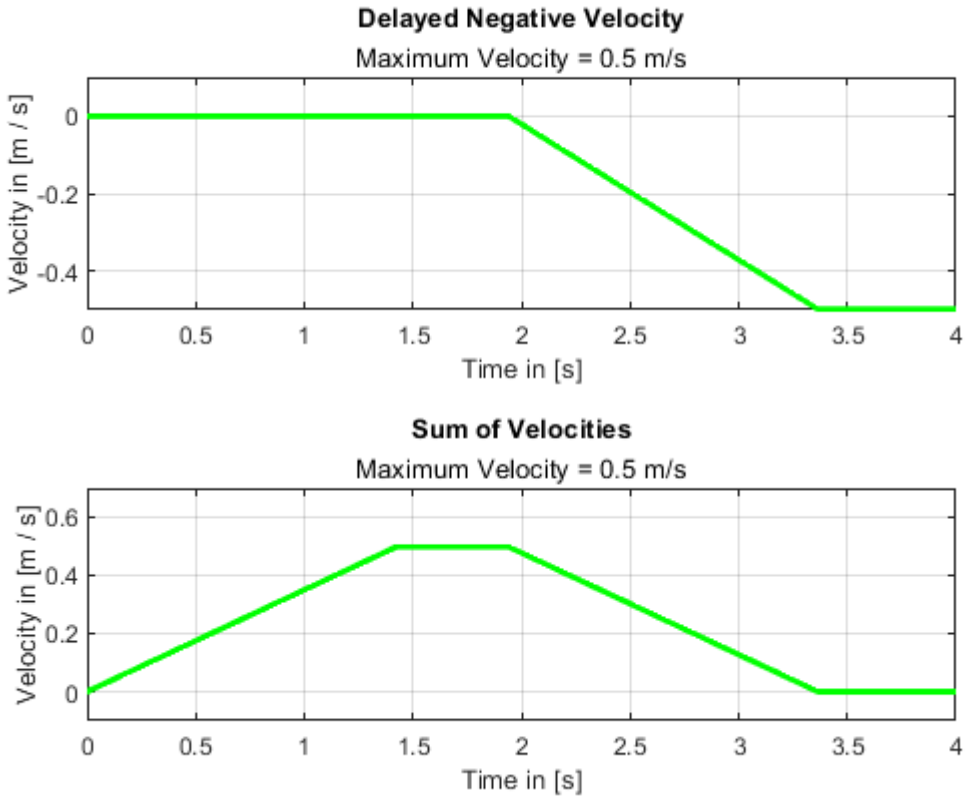


Abbildung 5.9 Geschwindigkeitsverläufe

Da in der Applikation noch 0.5 Sekunden Stillstand herrscht, bevor das Smartphone bewegt wird, wird der Streckenverlauf um 0.5 Sekunden durch die letzte Zeile verzögert. Damit liegt der finale Streckenverlauf vor und kann geplottet werden. Mit **Abbildung 5.10** folgen die beiden Graphen. Der obere Sollwertverlauf entsteht durch Integration der Summe der Geschwindigkeitsverläufe, der untere ist derselbe Plot nur um 0.5 Sekunden verzögert.

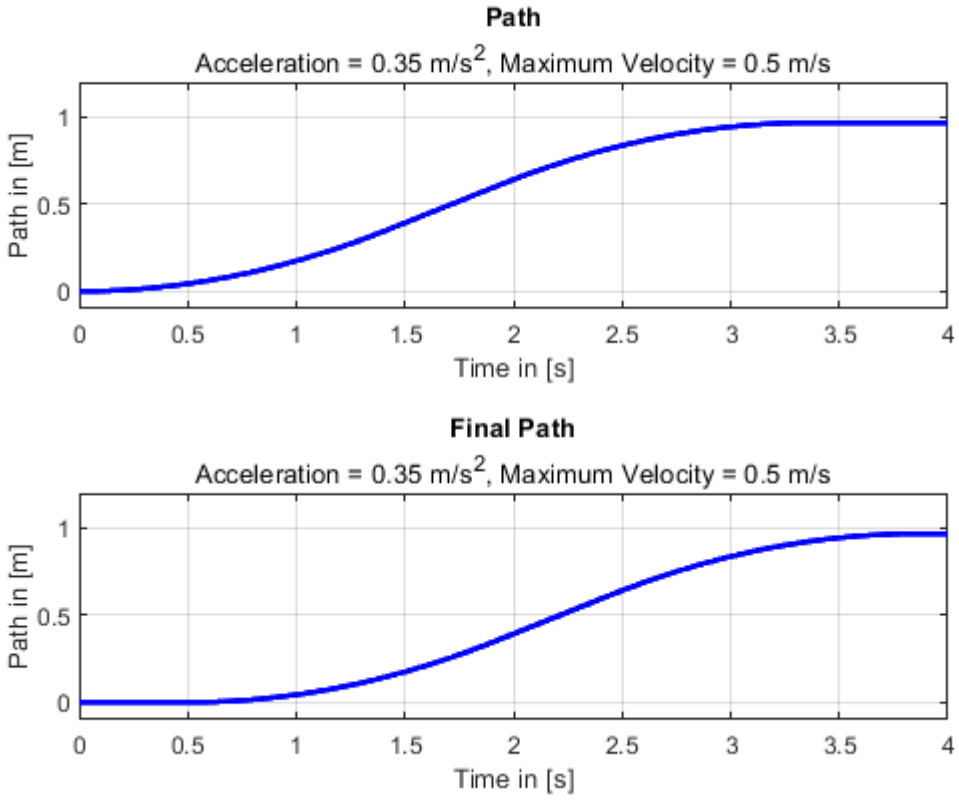


Abbildung 5.10 Finale Sollwertverläufe

Im letzten Abschnitt der Function wird zuerst die Figure erstellt. In diese wird nun der Streckenverlauf geplottet. Beschriftungen, Titel und Achsengrenzen werden noch festgelegt.
Der Befehl schaltet auch die Eigenschaft „Double Buffer“ in der Figure ein. Da die Regelgröße in Echtzeit in diese Figure geplottet werden soll, hilft dieser Befehl enorm. Er verbessert nämlich die Performance bei einer Echtzeitaufzeichnung. Der Sollwertverlauf **W** wird nun geplottet und die Figure wartet auf die Aufnahme der Regelgröße **Y**.

5.2.2 Echtzeitplot der Regelgröße

Im Bereich „Plot real time data“, zu sehen in **Abbildung 5.11**, wird die Regelgröße Y in Echtzeit geplottet. Dazu empfängt das Modell zwei Datenströme vom Smartphone. „UDP Data Y“ erhält die gemessene, zurückgelegte Strecke, also die Regelgröße Y ①. Damit Daten von diesem Block auch nur dann in der zuvor angelegten Figure aufgezeichnet werden, behilft sich das Modell mit einem zweiten UDP Receive Block. „UDP Logic 2“ ② gibt zuerst „0“ aus, sobald aber die Sensoren anfangen Daten aufzunehmen, schaltet die Ausgabe auf „1“ und aktiviert für 5 Sekunden das Subsystem „Level 2 S-Function“ ③.

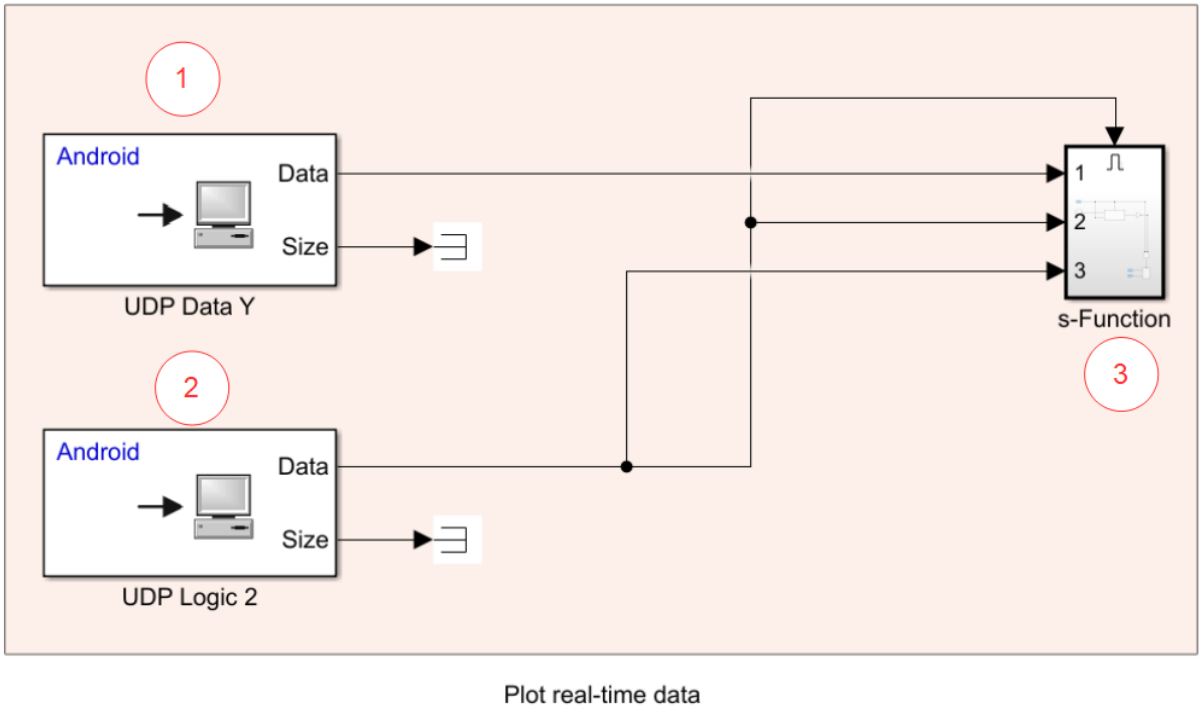


Abbildung 5.11 Bereich Plot real-time data

In **Abbildung 5.12** ist das Subsystem zu sehen, welches nun für 5 Sekunden aktiviert ist.

In diesem Subsystem befindet sich die Level 2 S-Function ①. Diese hat 2 Eingänge und 2 Ausgänge. Ihr Zweck ist, in die bereits erstellte Figure, die Regelgröße Y zu plotten. Auch garantiert eine Level 2 S-Function eine bessere Synchronität mit der Simulation, bei einer Ausführung. Zuerst sollen die zwei Eingänge betrachtet werden. Eingang 1, hier genannte „Controlled Variable“ stellt die am Smartphone aufgezeichnete Regelgröße bereit ②. Eingang 2 erhält zu den Werten von Eingang 1 Zeitwerte ③. Die Zeitwerte stellt ein simpler Aufbau bereit. Der Aufbau beginnt mit dem Anfangswert „0“, hier hinterlegt im Delay Block. Zu jedem Zeitschritt bei der Ausführung der Simulation wird der Wert 0.01 addiert. Dieser Wert entspricht der Abtastzeit der Sensoren des Smartphones. So erhält die Funktion zu jeder Zeit einen Wert für die Regelgröße Y und einen dazugehörigen Zeitwert. Da sich diese Blöcke in einem Enabled Subsystem befinden, beginnt der Zähler auch erst zu zählen, wenn Daten übertragen werden. Die zwei Ausgänge der S-Function enthalten die Regelgröße Y und die dazugehörigen Zeitwerte. Diese werden nun im MATLAB Workspace unter den Namen „path“ und „time“ abgespeichert ④.

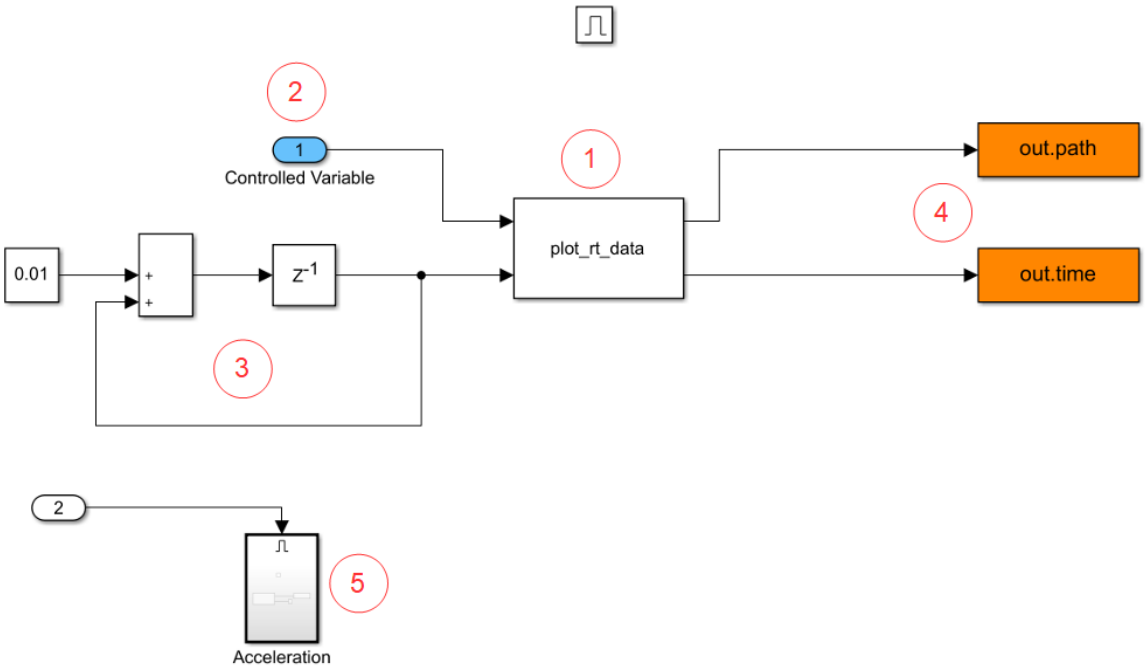


Abbildung 5.12 Subsystem s-Function

Die Level 2 S-Function verarbeitet nun diese Informationen und plottet diese in die Figure. Um dies besser zu verstehen, wird nun der benutzte Code betrachtet. Level 2 S-Functions arbeiten mit sogenannten DWork Vektoren, diese speichern zu jedem Zeitpunkt Werte ab und verarbeiten diese auch innerhalb der Funktion[5]. Dies geschieht wieder in Zeitabständen von 0.01 Sekunden. Der tatsächliche Code ist um einiges länger, aber da es sich dabei um Einstellungen handelt, werden hier nur die wichtigsten Teile betrachtet.

[5] MATHWORKS: Write Level 2 MATLAB S-Functions. https://www.mathworks.com/help/simulink/sfg/writing-level-2-matlab-s-functions.html 28.02.2021 23 Uhr

```
Function Start(block)

block.Dwork(1).Data = 0;
block.Dwork(2).Data = 0;
```

Hier werden die Anfangsbedingungen des Blockes festgelegt. Da sich das Smartphone zum Zeitpunkt t=0 bei 0 Metern befindet, wird der Wert 0 in die beiden Dwork Vektoren geschrieben.

```
Function Update(block)

block.Dwork(1).Data = block.InputPort(1).Data;
block.Dwork(2).Data = block.InputPort(2).Data;

persistent controlPlot
hold on
controlPlot = plot(0,0,'g.');
set(controlPlot,'Xdata',[block.Dwork(2).Data],'Ydata',[block.Dwork(1).Data]
);
drawnow nocallbacks
```

In diesem Abschnitt werden die Dwork Vektoren, zu jedem Zeitpunkt der Simulation, aktualisiert. Die Werte beziehen sie dabei von den beiden Eingängen der Funktion. Nun folgt der Teil, der das Plotten übernimmt. Zuerst wird ein grüner Punkt für die Werte y=0 und x=0 eingezeichnet, also der Startposition des Smartphones. Auch wird dieser Plot als Vektor „controlPlot“ abgespeichert. Die nächste Zeile bedient sich des zuvor abgespeicherten Vektors. Da das Einzeichnen von Plots sich stark auf die Performance von der Simulation auswirken kann, wird sich hier eines Tricks bedient. Normalerweise würde der Plot jedes Mal in aller Einzelheit neu gezeichnet werden. Das bedeutet in diesem Fall, dass 5/0.01 = 500-mal der Plot gezeichnet wird. Daraus resultiert eine immer langsamer werdende Simulation. Um dies zu umgehen wird der Vektor des Plots aktualisiert, und zwar um die beiden Dwork Vektoren, die wiederum durch die Eingänge aktualisiert werden. Die beiden Eigenschaften des Vektors „controlPlot“, Xdata und Ydata erhalten den aktuellen Zeitwert und den aktuellen Wert der Regelgröße Y. Somit werden insgesamt 500 grüne Punkte eingezeichnet, die in ihrer Gesamtheit den Verlauf der Regelgröße **Y** darstellen. Der Befehl „drawnow“ aktualisiert die Figure und der Befehl „hold on“ behält alle vorherig eingezeichneten Punkte bei. Ein vollständig gezeichneter Regelgrößenverlauf Y, ist in **Abbildung 5.13** zu sehen. Hier wurde auch zuerst der Sollwertverlauf wie in Kapitel 5.2.1 beschrieben geplottet. Die MATLAB Function erlaubt keine Legende in der Figure, deswegen fehlt diese hier.

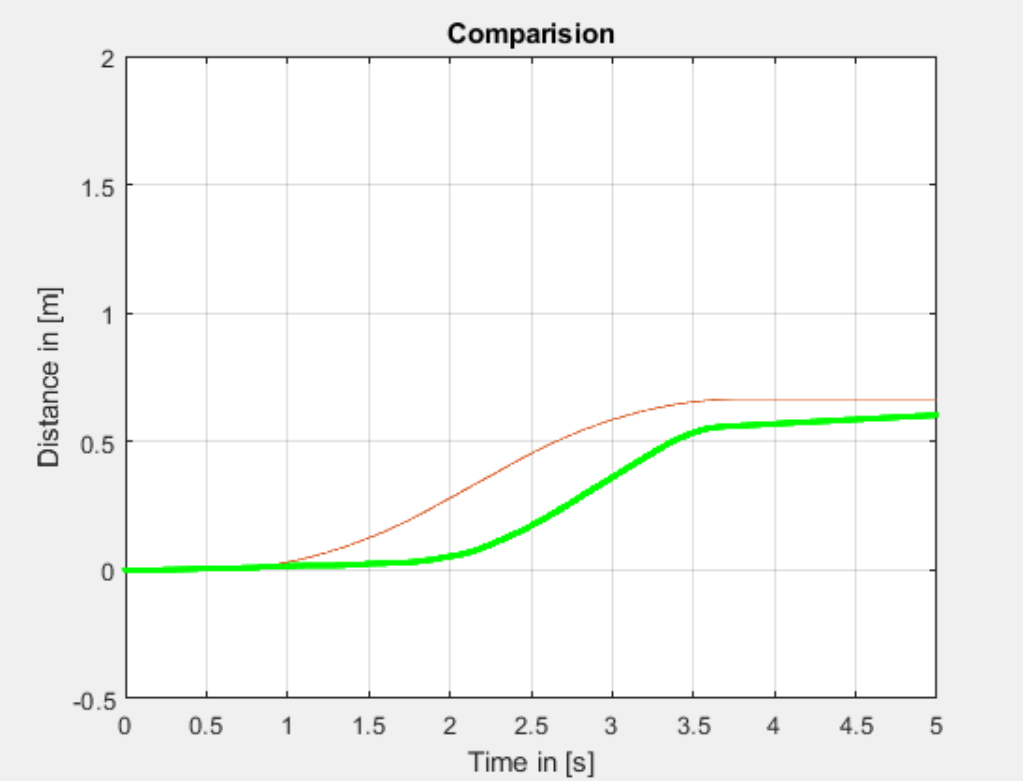


Abbildung 5.13 Echtzeitplot der Regelgröße

```
function Outputs(block)

block.OutputPort(1).Data = block.Dwork(1).Data;
block.OutputPort(2).Data = block.Dwork(2).Data;
```

Zuletzt werden hier noch die Ausgänge erläutert. Diese erhalten von den DWork Vektoren die aktuellen Werte. Die Ausgänge der Funktion leiten diese nun an zwei „to Workspace“ Blöcke weiter. Diese speichern den Regelgrößenverlauf und die dazugehörigen Zeitwerte im Workspace von MATLAB, unter den Namen „path“ und „time“, ab.

Nun liegen alle benötigten Daten vor, um mit der Regleridentifikation fortzufahren. MATLAB hat nun Kenntnis über den Sollwertverlauf W, den Regelgrößenverlauf Y und schließlich noch die dazugehörigen Zeitwerte. Diese 3 Vektoren zu einer Matrix vereint liefern die nötige .mat Datei, die das Programm pzMove benötigt, um eine Übertragungsfunktion zu berechnen.

Abschließend folgt mit **Abbildung 5.14** eine kurze Übersicht, wie der Informationsfluss während des Experimentes besteht.

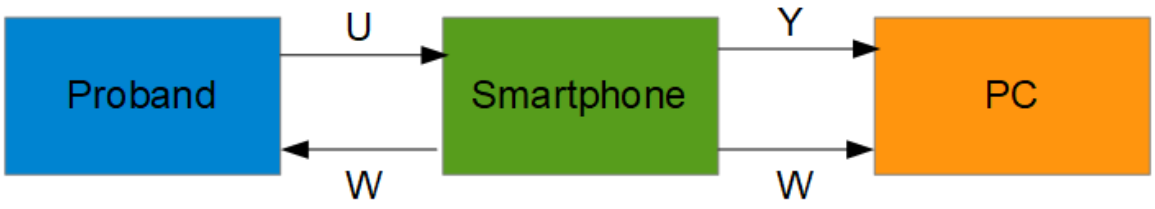


Abbildung 5.14 Informationsfluss

6 Regleridentifikation

6.1 Bereitstellung der nötigen Daten

Da nun alle Daten übertragen wurden, kann mit der Regleridentifikation begonnen werden. Hierzu müssen zuerst die bereitgestellten Daten aus dem Workspace ausgelesen und vorbereitet werden. Für diese Aufgabe steht ein weiteres MATLAB Skript [9.2.1] bereit, dass nach der Simulation ausgeführt werden sollte. Im Weiteren werden die Vorgänge betrachtet, die das Skript vornimmt.

```
%% Simulationsdaten auslesen

W = out.ref.Data;
Y = out.path.Data;
T = out.time.Data;
```

Diese drei Zeilen lesen die Ausgabe der Simulation aus und schreiben die Werte in Vektoren. Hierbei handelts es sich um den Sollwert W, die Regelgröße Y und den dazugehörigen Zeitvektor T. Diese wurden zuvor in den MATLAB Workspace gespeichert, siehe dazu Kapitel 5.2.1 und Kapitel 5.2.2. Da die Vektoren nach der Auslesung, sich noch als verschiedene Formate vorfinden, müssen diese noch formatiert werden. Der Sollwert W wird als 1x501x5 double ausgegeben. Der Zeitvektor hat in den ersten zwei Zeilen den Wert „0“. Grund hierfür ist eine minimale Verzögerung bei der Ausführung der S-Function.

```
%% Sollwert formatieren

W = squeeze(W);
W = W(:,1);
```

Der Befehl „squeeze(W)“ gibt den Vektor des Sollwertes W aus, allerdings entfernt er dabei Dimensionen der Länge 1. Der Vektor befindet sich also jetzt im Format 501x5. Die nächste Zeile liest nun die erste Spalte aus und speichert diese als „W“. Dabei ist es egal welche Zeile ausgelesen wird, da sich in den fünf Zeilen die gleichen Werte befinden.

```
%% Zeitvektor formatieren

T(1,:) = [];
tmax = max(T);
tmax = tmax + 0.01;
T = [T;tmax];
```

Als nächstes muss der Zeitvektor angepasst werden. Da sich in den ersten beiden Zeilen des Vektors T der Wert „0“ befindet, wird zuerst die erste Zeile gelöscht. Um die Löschung einer Zeile auszugleichen, wird nun der maximale Wert des Vektors ermittelt und auf diesen wird der Wert „0.01“, also die Abtastzeit der Sensoren, addiert. Dieser Wert wird dem Vektor T nun als letzte Zeile

hinzugefügt. Da die Vektoren unterschiedliche Länge besitzen, müssen diese noch auf dieselbe Länge gebracht werden.

```
%% Länge der Vektoren bestimmen und aufeinander abstimmen


minLength = min(length(Y), length(W));

Y = Y(1:minLength);
W = W(1:minLength);
T = T(1:minLength);
```

Die erste Zeile dieses Abschnittes ermittelt, welcher Vektor am kürzesten ist. Dieser Wert wird nun verwendet, um alle drei Vektoren auf dieselbe Länge zu bringen. Nun sind die benötigten Vektoren alle im gleichen Format und können nun verwendet werden, um den Regler zu identifizieren.

```
%% Fit-Wert berechnen

fit = 100*(1-norm(W-Y)/norm(W-mean(W)));
```

Diese Zeile berechnet noch den Fit-Wert. Der Fit-Wert gibt an, wie gut die gemessenen Daten der Regelgröße Y, zu denen des Sollwertes W passen. Der Wert wird prozentual ausgegeben, dass bedeutet er nimmt einen Wert zwischen 0 und 100 an, wobei 100 der bestmögliche Fall ist, hierbei gilt Y=W. Für die Berechnung des Fit-Wertes wurde folgende Formel verwendet:

$$fit = 100 * \frac{1 - |W - Y|}{|W - \bar{W}|} \qquad (6.1)$$

Nun müssen nur noch zwei simple Vorgänge erfolgen, dann können die vorzeitigen Ergebnisse des Experimentes geplottet werden.

```
%% Bestimmung Vektoren

diff = W - Y;
mat = [T W Y];
```

Die erste Zeile berechnet die Differenz von Sollwert **W** und Regelgröße **Y**. Die zweite Zeile erstellt eine Matrix aus den folgenden Vektoren: Zeit **T**, Sollwert **W** und Regelgröße **Y**. Diese Matrix wird benötigt, um die Regleridentifikation durchzuführen. Die nächsten Befehlszeilen plotten den Sollwertverlauf W, den Regelgrößenverlauf Y, deren Differenz E und schließlich noch die aufgenommenen Beschleunigungswerte. Ein Beispiel dazu, sieht man in **Abbildung 7.3**.

6.2 Struktur der Übertragungsfunktion des Reglers

Um die Übertragungsfunktion des Reglers zu ermitteln wurden sich folgende Gedanken gemacht: der Regelkreis hat den Ausgang Regelgröße Y und den Eingang Sollwert W. Die Übertragungsfunktion des Sollwertes W ist die sogenannte Führungsübertragungsfunktion[6]:

$$G_w(s) = \frac{Y(s)}{W(s)} \qquad (6.2)$$

Die Führungsübertragungsfunktion berechnet sich auch zu:

$$G_w(s) = \frac{Y(s)}{W(s)} = \frac{G_s(s) * G_R(s)}{1 + G_s(s) * G_R(s)} \qquad (6.3)$$

Wobei $G_s(s)$ die Streckenübertragungsfunktion und $G_R(s)$ die Übertragungsfunktion des Reglers ist. Für dieses Experiment ist die Reglerübertragungsfunktion $G_R(s)$ von Bedeutung.
Dafür muss die Funktion nach $G_R(s)$ aufgelöst werden, hierbei wird $G_s(s) = 1$ angenommen.

$$G_w(s) = \frac{G_R(s)}{1 + G_R(s)} \qquad | * (1 + G_R(s))$$

$$G_w(s) * \big(1 + G_R(s)\big) = G_R(s)$$

$$G_w(s) + G_w(s) * G_R(s) = G_R(s) \qquad | - (G_w(s) * G_R(s))$$

$$G_w(s) = G_R(s) * \big(1 - G_w(s)\big) \qquad | \div (1 - G_w(s))$$

$$G_R(s) = \frac{G_w(s)}{1 - G_w(s)} \qquad (6.3)$$

Da davon ausgegangen wird, dass der Mensch ein PID-Regler ist, kann die Ordnung von $G_w(s)$ festgelegt werden. Ein PID-Regler besitzt eine Übertragungsfunktion 2. Ordnung. Da die 1 im Nenner von $G_R(s)$ den Nenner von $G_w(s)$ als Zähler und Nenner annehmen wird, kürzt sich der Nenner von $G_w(s)$ raus. Um dies besser zu verstehen folgt die Rechnung:

$$G_R(s) = \frac{\frac{Z}{N}}{1 - \frac{Z}{N}}$$

$$G_R(s) = \frac{\frac{Z}{N}}{\frac{N}{N} - \frac{Z}{N}}$$

[6] Prof. Dr. Ing. Peter Zentgraf: Vorlesungsskript Regelungstechnik 1

$$G_R(s) = \frac{\frac{Z}{N}}{\frac{N-Z}{N}}$$

$$G_R(s) = \frac{Z}{N} * \frac{N}{N-Z}$$

$$G_R(s) = \frac{Z}{N-Z}$$

Somit ist klar, dass der Zähler Z und der Nenner N von $G_w(s)$ von zweiter Ordnung sein müssen, damit auch $G_R(s)$ von zweiter Ordnung ist. Diese Information wird nun benutzt um $G_w(s)$ zu bestimmen.

6.3 Identifikation des geschlossenen Systems

6.3.1 pzMove

Zur Ermittlung von $G_w(s)$ wird sich eines Programmes namens „pz-Move“ bedient, das von der TH-Rosenheim bereitgestellt wird. Das Programm besitzt unter anderem die Fähigkeit, eine Streckenidentifikation durchzuführen. Dafür müssen dem Programm die folgenden Informationen geliefert werden: Eine Matrix mit 3 Spalten, wobei die erste Spalte Zeitwerte, die zweite Spalte die Eingangsgrößen und die dritte Spalte die Ausgangsgrößen enthält. In diesem Fall wären Spalte 2 der Sollwert **W** und Spalte 3 die Regelgröße **Y**. Die dazu benötigte Matrix wurde zuvor von dem MATLAB Skript erstellt. Auch benötigt wird der Ordnungsgrad von Nenner und Zähler der zu identifizierenden Übertragungsfunktion. Aus den bisherigen Überlegungen wurde klar, dass für Nenner und Zähler ein Ordnungsgrad von 2 angesetzt werden muss. Nachdem diese Informationen in das Programm eingespeist wurden, erhält man $G_w(s)$. Nun kann man mithilfe von $G_w(s)$ auf $G_R(s)$ zurückrechnen, und zwar mit der zuvor ermittelten Formel:

$$G_R(s) = \frac{G_w(s)}{1 - G_w(s)} \qquad (6.4)$$

6.3.2 m-File

Um diesen Vorgang später zu automatisieren, wird ein m-File, dass pzMove benutzt, um die Identifikation durchzuführen, bereitgestellt. Das m-File wird während des Experimentes aufgerufen und verwendet die zuvor erstellte Matrix „mat“, die den Sollwert **W**, die Regelgröße **Y** und die Zeitdaten **T** enthält. So muss für den Vorgang nicht erst die Matrix erstellt und in pzMove eingelesen werden. Das m-File „timeSig2lti“ [9.2.2] berechnet zuerst die Führungsübertragungsfunktion $G_w(s)$ [7] und schließlich auch $G_R(s)$, mithilfe der Function „GR2PID_add“ [9.2.4].
Die handschriftlichen Berechnungen, die nun folgen werden, werden also während des Experimentes automatisch ausgeführt.

[7] Zentgraf, Peter: Ein neues Verfahren zur Modellierung linearer Systeme. https://www.th-rosenheim.de/fileadmin/user_upload/Fakultaeten_und_Abteilungen/Fakultaet_ING/Personal/Ze/EinNeuesVerfahrenZurModellierungLinearerSysteme_Zentgraf_RZ_atp_11-12_2019.pdf

6.4 Berechnen der Reglerübertragungsfunktion

Um die Übertragungsfunktion eines PID-Reglers aufzustellen muss die für die identifizierte Führungsübertragungsfunktion $G_w(s)$ die stationäre Verstärkung $k_s = 1$ betragen. Die stationäre Verstärkung errechnet sich für $s \to 0$ und $t \to \infty$. Das bedeutet für $G_w(s)$, dass

$$G_w(s) = 1 \quad \text{für} \quad s \to 0$$

Um diesen Zustand immer zu erreichen wird überprüft ob $G_w(s) = 1$.
Falls dies nicht zutrifft wird zunächst k_s berechnet:

$$k_s = G_w(s) \quad \text{mit} \quad s \to 0$$

Nun erscheint k_s im Regelkreis als Vorfilter für den Sollwert, also berechnet sich die Führungsübertragungsfunktion mit $G_w(s)$ nun zu:

$$G_w(s) = k_s * G_{w2}(s)$$

Um die Führungsübertragungsfunktion $G_{w2}(s)$ zu erhalten, die eine stationäre Verstärkung von $k_s = 1$ besitzt, muss nun gerechnet werden:

$$G_{w2}(s) = \frac{G_w(s)}{k_s} \qquad (6.5)$$

Jetzt kann man die Reglerübertragungsfunktion mit der zuvor ermittelten Formel 6.3 berechnen:

$$G_R(s) = \frac{G_{w2}(s)}{1 - G_{w2}(s)} \qquad (6.6)$$

Die Reglerübertragungsfunktion ergibt sich nun in der Form:

$$G_R(s) = \frac{Z_2 * s^2 + Z_1 * s + Z_0}{N_2 * s^2 + N_1 * s}$$

Wobei Z_n und N_n für beliebige Werte stehen. Damit man den dominierenden Reglertypen bestimmen kann, muss diese Übertragungsfunktion noch in die additive Form der Übertragungsfunktion des PID-Reglers gebracht werden. Diese lautet wie folgt:

$$G_{PID}(s) = k_P + \frac{k_I}{s} + k_D * \frac{s}{1 + T_1 * s}$$

Für diesen Vorgang wird die zuvor erwähnte Function namens „GR2PID_add“ [9.2.4] benutzt. Dieses ermittelt aus einer Übertragungsfunktion $G_R(s)$, wenn möglich $G_{PID}(s)$ in additiver Form.
Liegt dann $G_{PID}(s)$ einmal vor, können der P-Anteil k_P, der I-Anteil k_I und der D-Anteil k_D ausgelesen werden. Der größte Wert bestimmt den Reglertypen und das Experiment ist damit abgeschlossen.

6.5 Überprüfung der Ergebnisse

Im Folgenden soll mithilfe eines zuvor aufgenommenen Datensatzes die zuvor geschilderten Vorgänge durchgeführt werden. Anschließend sollen die Ergebnisse in einer Simulation geprüft werden.

Zuerst wird die Matrix in pzMove eingespeist, dabei wurden folgende Einstellungen vorgenommen:

Set order:

Numerator: 2
Demoninator: 2

Filter solutions:

iStab = only stable

Assumed initial conditions:

iy0 = zero

Dabei gab pzMove die folgende Übertragungsfunktion aus:

$$G_w(s) = \frac{1.8855 * s^2 + 4.2625 * s + 16.316}{s^2 + 10.996 * s + 17.483}$$

Nun muss die stationäre Verstärkung k_s bestimmt werden:

$$G_w(s) = \frac{1.8855 * s^2 + 4.2625 * s + 16.316}{s^2 + 10.996 * s + 17.483} = \frac{0 * s^2 + 0 * s + 16.316}{0^2 + 0 * s + 17.483} = \frac{16.316}{17.483} \quad \text{mit } s \to 0$$

Daraus ergibt sich:

$$k_s \approx 0.9332$$

Mit k_s wird $G_{w2}(s)$ berechnet:

$$G_{w2}(s) = \frac{G_w(s)}{k_s} = \frac{\frac{1.8855 * s^2 + 4.2625 * s + 16.316}{s^2 + 10.996 * s + 17.483}}{0.9332} = \frac{1.8855 * s^2 + 4.2625 * s + 16.316}{s^2 + 10.996 * s + 17.483}$$

$$= \frac{2.0203 * s^2 + 4.5674 * s + 17.483}{s^2 + 10.996 * s + 17.483}$$

Nun kann $G_{w2}(s)$ in die zuvor ermittelte Formel $G_R(s) = \frac{G_{w2}(s)}{1-G_{w2}(s)}$ eingesetzt werden:

$$G_R(s) = \frac{\frac{2.0203 * s^2 + 4.5674 * s + 17.483}{s^2 + 10.996 * s + 17.483}}{1 - \frac{2.0203 * s^2 + 4.5674 * s + 17.483}{s^2 + 10.996 * s + 17.483}}$$

$$G_R(s) = \frac{\frac{2.0203 * s^2 + 4.5674 * s + 17.483}{s^2 + 10.996 * s + 17.483}}{\frac{s^2 + 10.996 * s + 17.483}{s^2 + 10.996 * s + 17.483} - \frac{2.0203 * s^2 + 4.5674 * s + 17.483}{s^2 + 10.996 * s + 17.483}}$$

$$G_R(s) = \frac{\frac{2.0203 * s^2 + 4.5674 * s + 17.483}{s^2 + 10.996 * s + 17.483}}{\frac{s^2 + 10.996 * s + 17.483 - (2.0203 * s^2 + 4.5674 * s + 17.483)}{s^2 + 10.996 * s + 17.483}}$$

$$= \frac{[2.0203 * s^2 + 4.5674 * s + 17.483] * (s^2 + 10.996 * s + 17.483)}{[s^2 + 10.996 * s + 17.483 - (2.0203 * s^2 + 4.5674 * s + 17.483)] * (s^2 + 10.996 * s + 17.483)}$$

$$G_R(s) = \frac{2.0203 * s^2 + 4.5674 * s + 17.483}{s^2 + 10.996 * s + 17.483 - (2.0203 * s^2 + 4.5674 * s + 17.483)}$$

$$G_R(s) = \frac{2.0203 * s^2 + 4.5674 * s + 17.483}{-1.0203 * s^2 + 6.4286 * s}$$

Dieses Ergebnis kann nun überprüft werden, indem es in die Formel

$$G_{w2}(s) = \frac{G_s(s) * G_R(s)}{1 + G_s(s) * G_R(s)} \quad mit\ G_s(s) = 1$$

Eingesetzt wird. Das Ergebnis stimmt, wenn nun wieder $G_{w2}(s)$ errechnet wird.

$$G_{w2}(s) = \frac{1 * \frac{2.0203 * s^2 + 4.5674 * s + 17.483}{-1.0203 * s^2 + 6.4286 * s}}{1 + 1 * \frac{2.0203 * s^2 + 4.5674 * s + 17.483}{-1.0203 * s^2 + 6.4286 * s}}$$

$$G_{w2}(s) = \frac{\frac{2.0203 * s^2 + 4.5674 * s + 17.483}{-1.0203 * s^2 + 6.4286 * s}}{\frac{-1.0203 * s^2 + 6.4286 * s}{-1.0203 * s^2 + 6.4286 * s} + \frac{2.0203 * s^2 + 4.5674 * s + 17.483}{-1.0203 * s^2 + 6.4286 * s}}$$

$$G_{w2}(s) = \frac{\frac{2.0203 * s^2 + 4.5674 * s + 17.483}{-1.0203 * s^2 + 6.4286 * s}}{\frac{-1.0203 * s^2 + 6.4286 * s + (2.0203 * s^2 + 4.5674 * s + 17.483)}{-1.0203 * s^2 + 6.4286 * s}}$$

$$= \frac{[2.0203 * s^2 + 4.5674 * s + 17.483] * (-1.0203 * s^2 + 6.4286 * s)}{[-1.0203 * s^2 + 6.4286 * s + (2.0203 * s^2 + 4.5674 * s + 17.483)] * (-1.0203 * s^2 + 6.4286 * s)}$$

$$G_{w2}(s) = \frac{2.0203 * s^2 + 4.5674 * s + 17.483}{-1.0203 * s^2 + 6.4286 * s + (2.0203 * s^2 + 4.5674 * s + 17.483)}$$

$$G_{w2}(s) = \frac{2.0203 * s^2 + 4.5674 * s + 17.483}{-1.0203 * s^2 + 6.4286 * s + (2.0203 * s^2 + 4.5674 * s + 17.483)}$$

$$G_{w2}(s) = \frac{2.0203 * s^2 + 4.5674 * s + 17.483}{s^2 + 10.996 * s + 17.483}$$

Es ergibt sich also wieder $G_{w2}(s)$ und die Rechnung ist soweit korrekt.

Um die Übertragungsfunktion des PID-Reglers in der additiven Form zu erhalten, wird

$$G_R(s) = \frac{2.0203 * s^2 + 4.5674 * s + 17.483}{-1.0203 * s^2 + 6.4286 * s}$$

in die Function „GR2PID_add“ eingespeist. Das Ergebnis lautet:

$$G_{PID}(s) = k_P + \frac{k_I}{s} + k_D * \frac{s}{1 + T_1 * s}$$

$$G_{PID}(s) = 1.1421 + \frac{2.7196}{s} + 0.4955 * \frac{s}{1 - 0.1587 * s}$$

Die Anteile betragen dabei $k_P = 1.1421$, $k_I = 2.7196$, $k_D = 0.4955$ und $T_1 = -0.1587$.
Der größte Wert ist bei diesem Experiment k_I, der Proband ist also ein I-Regler.
Nun kann zur Überprüfung dieses Ergebnis mit den Resultaten von dem bereitgestellten Skript „timeSig2lti“ [9.2.2] verglichen werden. Dieses berechnete:

$$G_R(s) = \frac{1.924 * s^2 + 3.722 * s + 15.46}{-s^2 + 5.547 * s}$$

$$G_{PID}(s) = 1.1733 + \frac{2.7865}{s} + 0.5584 * \frac{s}{1 - 0.1803 * s}$$

Die Anteile betragen dabei $k_P = 1.1733$, $k_I = 2.7865$, $k_D = 0.5584$ und $T_1 = -0.1803$.
Der größte Wert ist bei diesem Experiment k_I, also ist auch hier der Proband ein I-Regler.

6.6 Simulation der berechneten Übertragungsfunktion

Als letzter Beweis sollen die berechneten Ergebnisse mithilfe eines Simulink-Modells als richtig erklärt werden. Dazu wird zuerst der Sollwert an die Führungsübertragungsfunktion

$$G_w(s) = k_s * \ G_{w2}(s) = \frac{1.8855 * s^2 + 4.2625 * s + 16.316}{s^2 + 10.996 * s + 17.483}$$

übergeben. So entsteht ein simuliertes Signal der Regelgröße Y, also ySim. Dieses wird nun mit der vom Smartphone tatsächlich gemessenen Regelgröße verglichen. Ein MATLAB Skript übernimmt diese Aufgabe. Es folgen die Plots in **Abbildung 6.1**. Im oberen Fenster sieht man den Sollwert W, die Regelgröße Y und die simulierte Regelgröße ySim. Im unteren Fenster sieht man die Differenz von ySim und Y. Der zuvor berechnete fit-Wert von ySim und Y wird zusätzlich in den Titel geschrieben. Wie zu erkennen ist stimmen ySim und Y, mit einem fit-Wert von 97.53%, sehr gut überein.

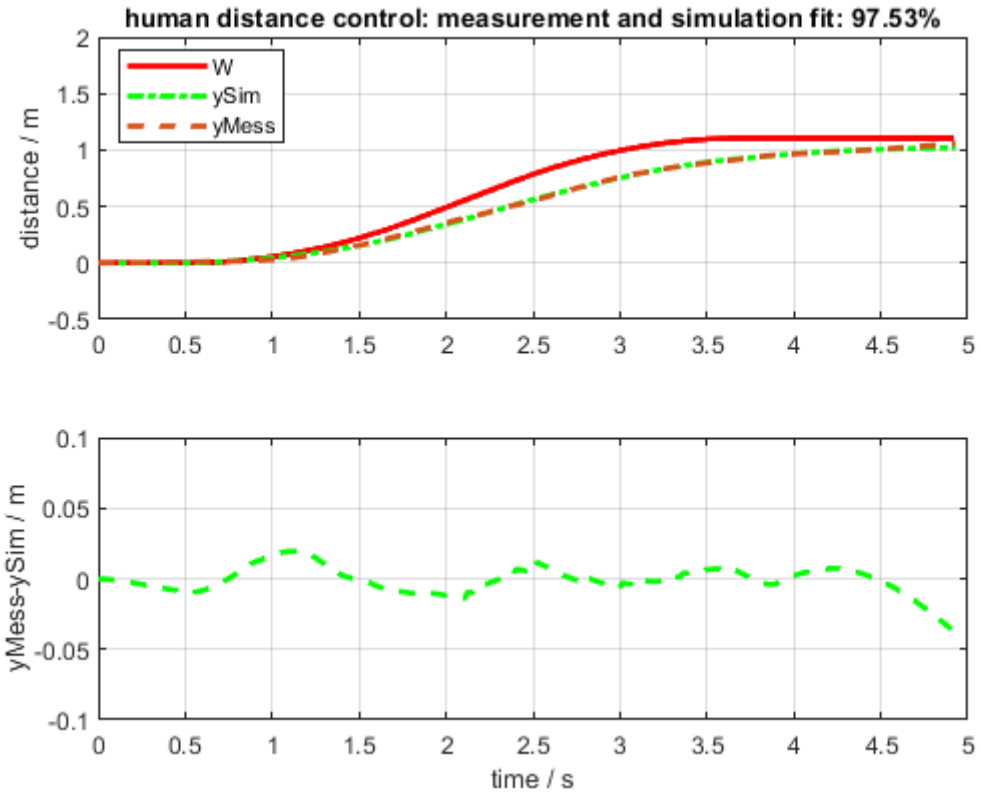


Abbildung 6.1 Simuliertes Signal der Regelgröße

Alternativ werden noch die zuvor berechneten Anteile $k_P = 1.1421$, $k_I = 2.7196$, $k_D = 0.4955$ und $T_1 = -0.1587$, in einen PID-Regler Block eingesetzt werden. Nun muss diesem nur noch die stationäre Verstärkung $k_s = 0.9332$ vorgeschaltet und eine Rückkopplung eingebaut werden. In **Abbildung 6.2** wird das Ergebnis gezeigt.

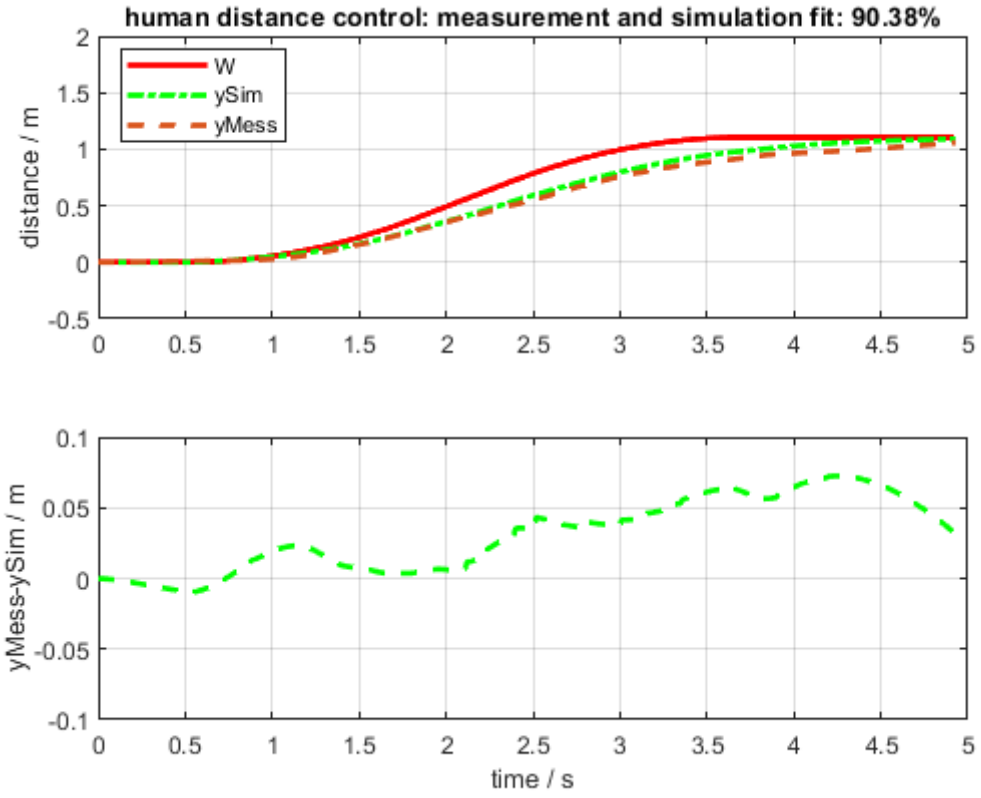


Abbildung 6.2 Simuliertes Signal der Regelgröße mithilfe eines PID-Reglers

Auch hier stimmen ySim und Y überein. Es wurde ein fit-Wert von 90.59% erreicht. Es wurde also bestätigt, dass alle Überlegungen bis hierhin korrekt sind. Das Experiment funktioniert also.

7 Durchführung des Experimentes

7.1 Versuchsaufbau

Um das Experiment leichter durchzuführen, wurde ein Versuchsaufbau konstruiert. Dieser besteht aus einem Gerüst aus Aluminiumprofilen. Dieses wurde so zusammengesetzt, dass es 4 Beine hat und so stabil steht. Das Aluminiumprofil hat mehrere Bohrungen in bestimmten Abständen, um drei Schienen anzubringen. Auf diesen Linearführungen befindet sich ein Schlitten. Auf dem Schlitten ist wiederum eine Aluschiene auf die angepasste Smartphonehalterung angeklebt ist. Auf der Schiene befindet sich ein Griff, um das Verschieben des Schlittens zu erleichtern. An den Enden der Linearführung befinden sich Stopper, damit der Schlitten nicht von den Schienen fährt. Schließlich ist noch eine Box mit einer Öffnung an dem Gerüst angebracht. In dieser Box befindet sich eine Platine, auf die der Mikrokontroller gelötet wurde. Durch die Öffnung sieht man das Display des Mikrokontrollers. Der Versuchsaufbau ist in **Abbildung 7.1** zu sehen.

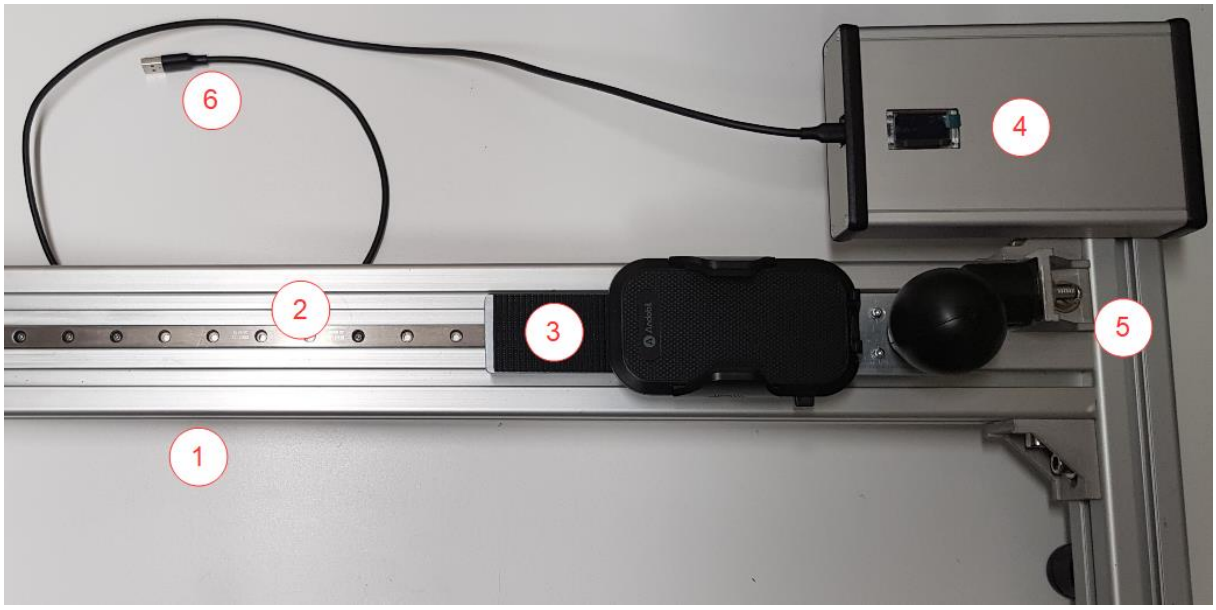


Abbildung 7.1 Versuchsaufbau

Stückliste:

- Aluminiumgerüst ①
- Linearführung ②
- Schlitten mit Smartphonehalterung und Griff ③
- Box mit eingebautem Mikrokontroller ④
- Stopper ⑤
- Stromversorgung Mikrokontroller ⑥

Um den Versuch durchzuführen, muss nur das Smartphone in die Halterung gelegt werden. Sobald die Sensoren des Smartphones aktiviert sind und die Regelgröße Y aufgezeichnet wird, sollte der Schlitten mithilfe des Griffes bewegt werden. Die Linearführung garantiert, dass sich das Smartphone während des Experimentes nicht verdreht und der Beschleunigungssensor in Y-Richtung, die volle Beschleunigung aufnimmt. Nun kann das Smartphone und der PC mit dem Mikrokontroller verbunden werden. Sobald alle Programme startbereit sind, kann mit dem Experiment begonnen werden. In **Abbildung 7.2** kann der Aufbau mit einem angebrachten Smartphone betrachtet werden.

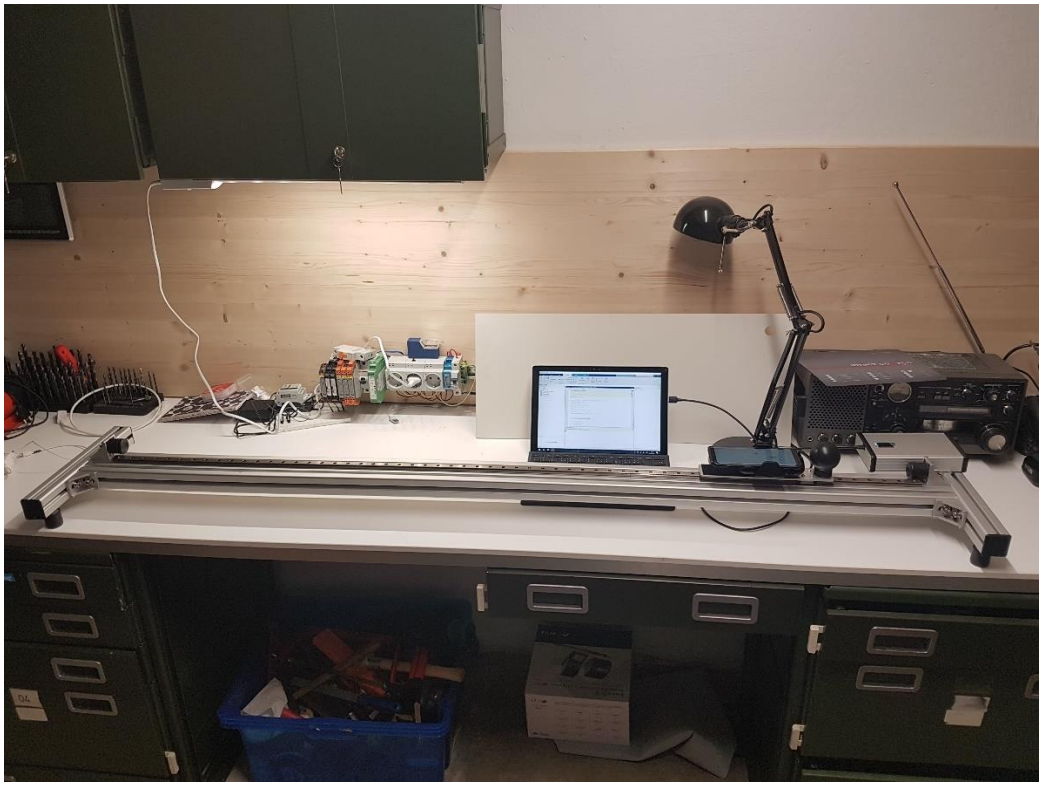


Abbildung 7.2 Versuchsaufbau mit angebrachtem Smartphone

7.2 Ablauf des Experimentes

Zuerst werden das MATLAB Skript „DistanceControl_Ident.m“ [9.2.1] und das Simulink Modell „DataTransfer_Ident.slx“ auf dem PC aufgerufen. Smartphone und PC müssen sich im selben Netzwerk befinden. Wenn möglich sollten Smartphone und PC mit dem Mikrokontroller verbunden werden. Nun wird das MATLAB Skript gestartet, dieses wiederum startet das Simulink Modell. Sobald dieses gestartet wurde, sollte der Proband nun den Sollwertverlauf **W** betrachten und versuchen diesen durch Bewegen des Smartphones nachzufahren. Dazu kann er entweder den Bildschirm des PCs oder des Smartphones nutzen. Die Simulation läuft 20 Sekunden, danach übernimmt das MATLAB Skript die Auswertung. Der Proband sieht nun die in **Abbildung 7.3** aufgeführten Plots.

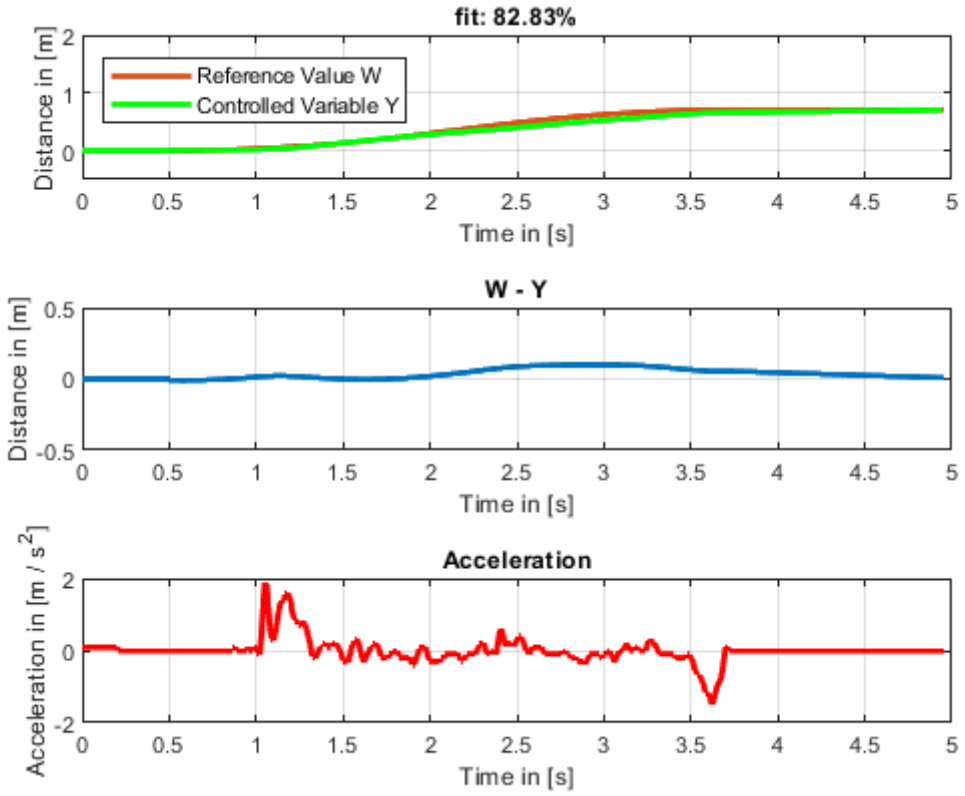


Abbildung 7.3 Auswertung des Experimentes

Zuerst öffnet sich eine Figure, die im ersten Fenster Sollwert **W** und Regelgröße **Y** plottet. Der Titel gibt den fit-Wert von Regelgröße und Sollwert an. Danach folgt die Differenz von Sollwertverlauf **W** und Regelgrößenverlauf **Y**. Das letzte Fenster gibt zusätzlich noch die aufgenommene Beschleunigung aus. Nun erfolgt die Regleridentifikation, indem das Skript die Function „timeSig2lti“ [9.2.2] aufruft. Dieses vollzieht automatisch die in Kapitel 6.5 durchgeführten Rechnungen. Also bestimmt es die Reglerübertragungsfunktion, sowie die Werte für die Reglerparameter k_P, k_I und k_D. Auch wird die simulierte Regelgröße ySim, sowie deren Abweichung zur realen Regelgröße Y berechnet und geplottet. Der Titel gibt auch die stationäre Verstärkung k_s, sowie den fit-Wert von simulierter zur realer Regelgröße an. Ein Beispiel ist in **Abbildung 7.4** zu sehen.

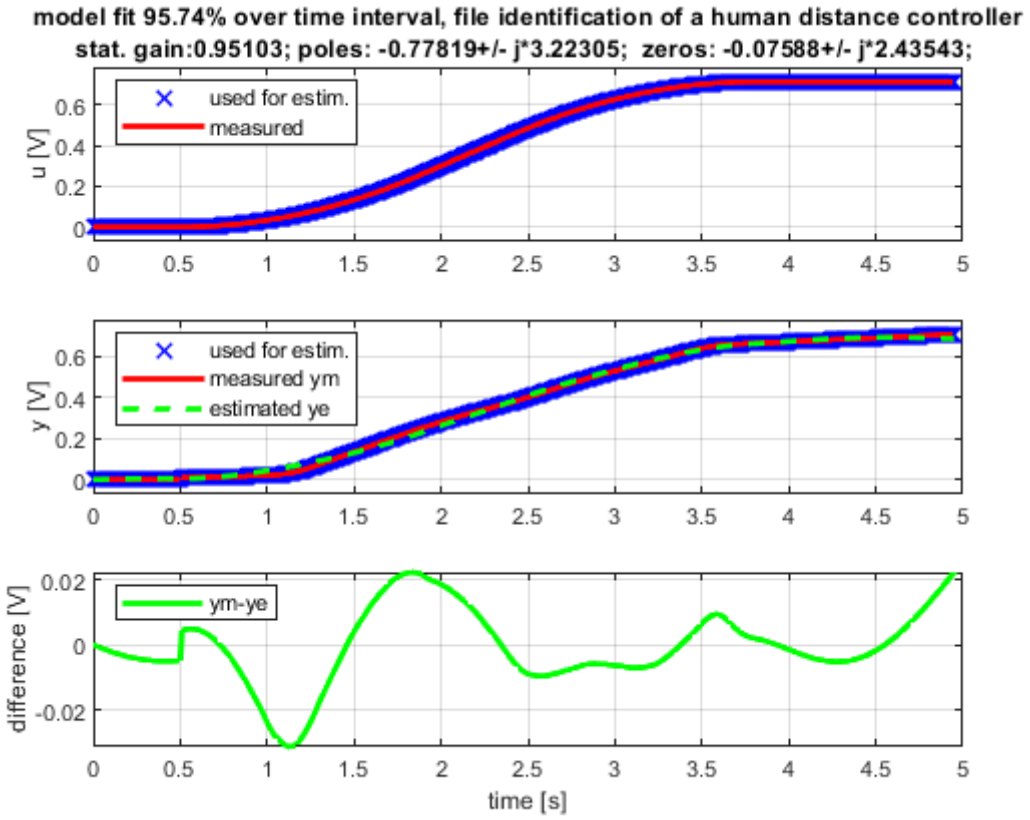


Abbildung 7.4 Plot von ySim und Y

Schließlich erhält der Proband die Parameter k_P, k_I und k_D, sowie die Übertragungsfunktion des Reglers in der Kommandozeile von MATLAB. Auch wird der größte Parameterwert benannt, damit der Proband weiß welcher Typ Regler er ist. Die finale Auswertung ist in **Abbildung 7.5** zu sehen.

```
GR2 =

  -1.452 s^2 - 0.2203 s - 8.62
  ----------------------------
         0.6678 s^2 - s


-------------------------
KP: 5.9768
KI: 8.62
KD: 5.4432
-------------------------
You are a Controller of the type I
```

Abbildung 7.5 Finale Auswertung

Der Proband hat nun alle Informationen erhalten und das Experiment ist somit vollständig ausgeführt. Je nach Qualität der Messung können die Ergebnisse variieren.

8 Zusammenfassung & Ausblick

8.1 Zusammenfassung

Im Zuge dieser Arbeit wurde ein Experiment entworfen, welches den Probanden als Regler in einen Regelkreis einsetzt. Für die Umsetzung wurden auf Softwareseite diverse Entwicklungsumgebungen und Programmpakete eingesetzt. Die Komplexität wurde für die Versuchsdurchführung in den Hintergrund verlagert, so dass der Proband sich auf die einfache Versuchsdurchführung sowie deren Interpretation konzentrieren kann und somit den Einstieg in die Welt der Regelungstechnik erleichtert.

Die Arbeit begann mit der Entwicklung der Applikation. Nach und nach stellte sich heraus, was möglich war und was anders zu lösen oder aufgrund von Limitierungen der eingesetzten Systeme unmöglich war. Da zum Beispiel eine vollständige Verarbeitung der Daten innerhalb der Smartphone Applikation nicht umzusetzen war, werden nun Daten vom Smartphone an den PC übertragen. Dies bringt neben der Lösung des konkreten Problems den Vorteil, dass der PC viel mehr Freiheiten bietet. Durch die Ergänzungen des zunächst angedachten Systems konnten die ursprünglich festgelegten Ziele alle erreicht werden. Bei der Identifikation des Reglers halfen Hinweise und Programme bereitgestellt von Herrn Prof. Dr.-Ing. Peter Zentgraf.

Das Experiment eignet sich gut, um in den ersten Vorlesungen der Regelungstechnik eingesetzt zu werden, um einen vorerst vermeintlich komplexen Regelkreis erlebbar zu machen und abstrakte Begriffe verständlich zu visualisieren.

Regelungstechnik ist in unserer täglichen Umgebung äußerst präsent und wir sind es gewohnt die Funktionen ohne weiteres Nachdenken zu nutzen. Die mathematische Darstellung ist jedoch anfangs erstmal abstrakt, dass auch Studenten den Bezug nicht intuitiv herstellen können. Die Durchführung praktischer Versuche erleichtert den Einstieg in die Thematik erheblich. Die Verwendung eines üblichen Gebrauchsgegenstandes verfestigt den Bezug zur realen Nutzung.

Der Verfasser dieser Arbeit konnte bei den Praktikumsversuchen im Fach Regelungstechnik die Vorzüge des Praxisbezugs erfahren.

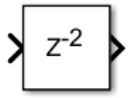
8.2 Ausblick

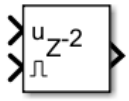
Verbessern könnte man das Experiment, indem man die Streckenberechnung, die das Smartphone vornimmt, für den dreidimensionalen Raum umsetzt. So könnte der Proband, statt das Smartphone über den Tisch zu schieben, sich einfach mit dem Smartphone in der Hand bewegen. Dies stellte sich für diese Arbeit aber als zu kompliziert heraus und nach Absprache mit Herrn Prof.-Dr. Ing. Peter Zentgraf ist es auch nicht Teil dieser Arbeit. Eine Erweiterung der Applikation, etwa dass diese mehr übernimmt als nur das Aufzeichnen und Senden von Daten, wurde versucht, da die meisten Smartphones aber sehr viel leistungsschwächer sind als ein PC und es bei Tests zu sichtbaren Einbrüchen der Performance kam, wurde sich dazu entschlossen die Applikation auf das Nötigste zu reduzieren.

9 Anhang

9.1 Häufig verwendete Simulink Blöcke

Delay und Enabled Delay



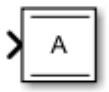


Der Delay Block verzögert das eintreffende Signal um eine bestimmte Anzahl an Simulationsschritten. Der Enabled Delay Block verzögert das eintreffende Signal nur, wenn am unteren Eingang der Wert „1“ eintrifft. Diese beiden Blöcke werden vermehrt in dem Modell für das Smartphone benutzt, um Abläufe zeitlich einzuordnen.

Enabled Subsystem

Dieses Subsystem, also alle Blöcke, die sich in diesem Subsystem befinden, werden erst ausgeführt, sobald der obere Eingang, den Wert „1“ enthält.

Data Store Write, Data Store Memory und Data Store Read

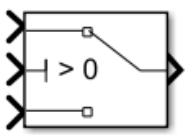


Data Store Write erhält das Eingangssignal und gibt es an den Data Store Block weiter.

Data Store Memory speichert das eingelesene Signal.

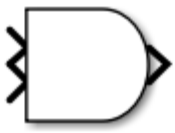
Data Store Read gibt das eingelesene Signal aus.

Switch



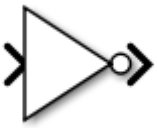
Der Switch gibt eines von zwei Signalen weiter. Das obere Signal wird weitergegeben, wenn die Bedingung in der Mitte erfüllt ist. Andernfalls wird das untere Signal weitergegeben

AND



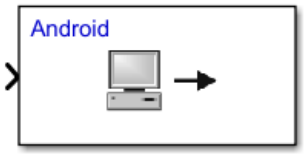
Der AND Block gibt den Wert „1“ aus, wenn seine beiden Eingänge auch den Wert „1“ erhalten.

NOT



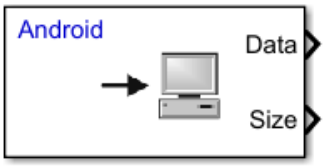
Der NOT Block gibt den Wert „1“ aus, wenn sein Eingang den Wert „0“ erhält.

UDP Send



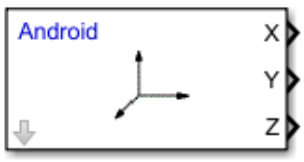
Der UDP Send Block empfängt ein eintreffendes Signal und nutzt das UDP Netzwerkprotokoll, um das Signal an ein anderes Gerät im selben Netzwerk zu übertragen.

UDP Receive



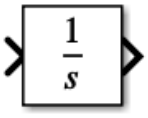
Der UDP Receive Block empfängt die Daten, die er vom UPD Send Block erhält und gibt diese bei dem Ausgang „Data“ aus. Der Ausgang „Size“ liefert nur Informationen zur Datengröße.

Accelerometer



Der Block Accelerometer spricht die Beschleunigungssensoren des Smartphones an und gibt Beschleunigungsdaten in X-,Y und Z-Richtung aus.

Integrator



Der Integrator Block integriert das eintreffende Signal.

9.2 Verwendete MATLAB Functions

9.2.1 DistanceControl_Ident

Diese Function startet die Simulation auf dem PC. Weiterhin liest sie die vom Smartphone empfangenen Daten aus, danach verarbeitet und plottet sie diese. Schließlich ruft sie noch die Function „run_ControllerIdent“ auf, welche den Regler identifiziert. Mit den berechneten Informationen von „run_ControllerIdent“ wertet sie schließlich das Experiment aus.

9.2.2 timeSig2lti

Diese Function identifiziert das geschlossene System und damit den Regler. Auch verwendet sie weitere Functions, um etwa die Regelgröße zu simulieren. Siehe dazu **Abbildung 7.4**. Die Erstellung der Function war nicht Teil dieser Arbeit und wurde von Herrn Prof. Dr.-Ing. Peter Zentgraf bereitgestellt. Sie benötigt eine dreispaltige Matrix, die Werte für ein Referenzsignal, ein Ausgangssignal und die Zeit enthalten. Interessierte können das in dem Literaturverzeichnis referenzierte PDF-Dokument „*Ein neues Verfahren zur Modellierung linearer Systeme*“ aufrufen.

9.2.3 run_ControllerIdent

Diese Function ist eine gekürzte Variante von „timeSig2lti“. Sie hat die gleiche Funktion wie „timeSig2lti“, wurde aber auf das, für dieses Experiment Nötigste, reduziert.

9.2.4 GR2PID_add

Diese Function stellt die Übertragungsfunktion eines PID-Reglers in additiver Form da. Auch diese Function war nicht Teil dieser Arbeit und wurde von Herrn Prof. Dr.-Ing. Peter Zentgraf bereitgestellt. Sie wird kurz vor Ende des Experimentes aufgerufen, um das finale Ergebnis zu berechnen.

9.2.5 plot_rt_data

Dies ist eine MATLAB Level 2 S-Function. Sie wird verwendet, um die Regelgröße **Y** in Echtzeit auf dem PC zu plotten.

9.2.6 plot_reference

Diese Function ist in dem PC-Modell verbaut. Mithilfe von übersandten Daten, bildet die Function den Sollwertverlauf, den das Smartphone generiert hat, nach und plottet diesen.

Abbildungsverzeichnis

Literaturverzeichnis

Mathworks. 2021. *Accelerometer.* [Online] 28. 02 2021. https://www.mathworks.com/help/supportpkg/android/ref/accelerometer.html.

—. Write Level-2 MATLAB S-Functions. [Online] [Zitat vom: 28. 02 2021.] https://www.mathworks.com/help/simulink/sfg/writing-level-2-matlab-s-functions.html.

Serge Zacher, Mafred Reuter. 2013. *Regelungstechnik für Ingenieure.* s.l. : Springer, 2013.

Wikipedia. User Datagram Protocol. [Online] [Zitat vom: 28. 02 2021.] https://de.wikipedia.org/wiki/User_Datagram_Protocol.

Zentgraf, Peter. 2019. *Ein neues Verfahren zur Modellierung linearer Systeme.* [PDF-Dokument] 2019.

—. 2020. *Vorlesungsskript Regelungstechnik 1.* [PDF-Dokument] 2020.

Symbolverzeichnis

Formelzeichen	Name
W	**Sollwert**
Y	**Regelgröße**
E	**Regeldifferenz**
U	**Stellgröße**
$\boldsymbol{G_S}$	**Regelstrecke**
$\boldsymbol{G_R}$	**Regler**
$\boldsymbol{G_w(s)}$	**Führungsübertragungsfunktion**
$\boldsymbol{G_{PID}(s)}$	**Übertragungsfunktion des PID-Reglers**
$\boldsymbol{k_P}$	**Proportionalanteil**
$\boldsymbol{k_I}$	**Integralanteil**
$\boldsymbol{k_D}$	**Differentialanteil**
$\boldsymbol{k_s}$	**Stationäre Verstärkung**

Name / family name

H	i	l	p	o	l	t	s	t	e	i	n	e	r											

Vorname / first name

L	e	o	n																					

Matrikelnummer / enrolment no.

8	7	0	5	4	7		

Studiengang / degree course.

M	a	s	c	h	i	n	e	n	b	a	u		B	a	c	h	e	l	o	r				

An die /To the
Technische Hochschule Rosenheim
Prüfungsamt / Exam Office
Hochschulstr. 1
83024 Rosenheim

Eigenständigkeitserklärung / *Declaration of Originality*

zur Abschlussarbeit mit folgenden Thema / *of my thesis with the following topic:*

E	n	t	w	i	c	k	e	l	n		u	n	d		B	e	w	e	r	t	e	n		
e	i	n	e	r		A	b	s	t	a	n	d	s	r	e	g	e	l	u	n	g			
m	i	t	h	i	l	f	e		e	i	n	e	s											
S	m	a	r	t	p	h	o	n	e	s		a	l	s		S	e	n	s	o	r			

Hiermit bestätige ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken (dazu zählen auch Internetquellen) entnommen sind, wurden unter Angabe der Quelle kenntlich gemacht.

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Bad Endorf, 01.03.2021
Ort, Datum / Place, Date

Leon Hilpoltsteiner
Unterschrift / Signature