Ermittlung der Laufzeiten von BLE (Bluetooth LE) … und OTG

Andreas D., März 2020

Aufgrund des doch sehr schwachen Ergebnisses in einer von uns vergebenen Bachelorarbeit soll hier am Beispiel BLE (OTG würde technisch ähnlich funktionieren) kurz skizziert werden, wie wir uns in etwa die Vorgehensweise zur Ermittlung von Bluetooth LE Laufzeiten vorgestellt hatten.

Der den Messungen zugrunde liegende Code stammt von Brian.

Aufgabenstellung

Als vorbereitende Maßnahme zur erfolgreichen Umsetzung des acaPendulum-Projektes mit einem externen Taster sollte in der Abschlussarbeit zunächst die Laufzeit von BLE zwischen einem (Achtung: Richtung!) Microcontroller (z.B. Arduino, ESP32) und einer App auf einem Tablet-PC ermittelt werden. Als triviale, praktische Aufgabenstellung war die Realisierung eines **Reaktionstesters** als App auf einem Tablet in Verbindung mit einem externen Taster vorgesehen, der an einem uC angeschlossen ist (s. Abbildung 1). Sobald in der App durch eine Zufallszeit eine "quadratische Lampe" als Farbfläche von dunkel nach rot geschaltet wird, sieht ein menschlicher Beobachter/Proband das mit seinen Augen, und er soll dann möglichst schnell den Taster betätigen. Ist das geschehen, wird dieses Ereignis als Datum **drahtlos über BLE** vom uC an das Tablet übertragen. Die App registriert das Eintreffen des Datums und ermittelt die vergangene Zeit seit der Farbumschaltung von Dunkel auf Rot. Die vergangene Zeit ist die Reaktionszeit des Probanden.

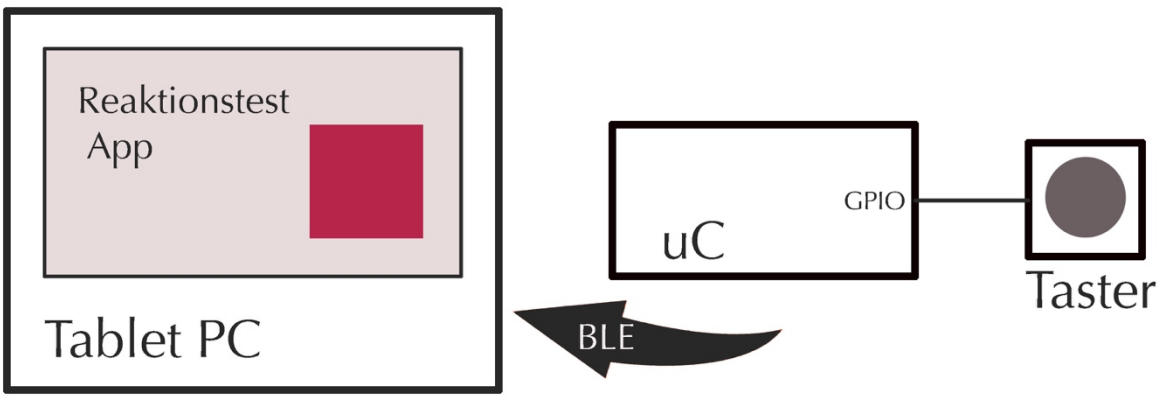

Abbildung 1 Skizzierung der Aufgabenstellung: ein Reaktionstester mittels BLE

Ergibt beispielsweise die Zeitmessung in der App, dass der Proband den Taster 200ms nach dem Aufleuchten der Lampe gedrückt hat, dann muss von diesem Wert noch die BLE-Laufzeit zwischen uC und App abgezogen werden. Wie groß die BLE-Laufzeit ist und ob sie starke Streuung aufweist oder sehr stabil ist, ist zu Beginn der Betrachtungen nicht bekannt und muss erst ermittelt werden.

Erste Überlegung zur Messung der BLE-Laufzeit: der BLE-Weg HIN plus dem Weg ZURÜCK geteilt durch 2

(Es handelt sich hierbei um die erste und einzige selbstständige Überlegung des Studierenden, die er dann auch programmtechnisch im Rahmen seiner Arbeit weiter verfolgt hat.)

Ein erster, für sich alleine betrachtet leider unbrauchbarer Ansatz zur Bestimmung der BLE-Laufzeit vom uC zur App ist, von der App ein BLE-Datum an den uC zu schicken, und dieser schickt das empfangene BLE-Datum über den gleichen Weg zurück an die App (s. Abbildung 2). Die hierfür benötigte Gesamtzeit würde durch zwei geteilt, dann hätte man die Zeit für einen Weg. Dieser Ansatz ist deshalb unbrauchbar, weil aus ihm nicht erkennbar ist, ob z.B. aufgrund der Hardware-Spezifika der BLE-Übertragungsweg **von der App zum uC** genau so lange dauert wie der Weg **vom uC zur App**. Diese Unterscheidung ist aber wichtig, weil eigentlich nur die Daten-Übertragungszeit vom uC zur App interessant ist. **Man möchte nur wissen, wie lange es über BLE dauert, bis die App mitbekommt, dass der Taster gedrückt wurde.** Die BLE-Laufzeit des anderen Weges, von der App zum uC, ist für die vorliegende Aufgabenstellung nicht von Bedeutung. (Das wurde dem Studierenden vor Beginn seiner Arbeit auch schon sehr deutlich gesagt, aber trotzdem hat er in den folgenden Wochen nur diesen einen Ansatz verfolgt. Für eine Bachelorarbeit ist das eigentlich zu wenig, denn der Sinn seiner Arbeit lag darin, sich etwas anderes auszudenken :-(

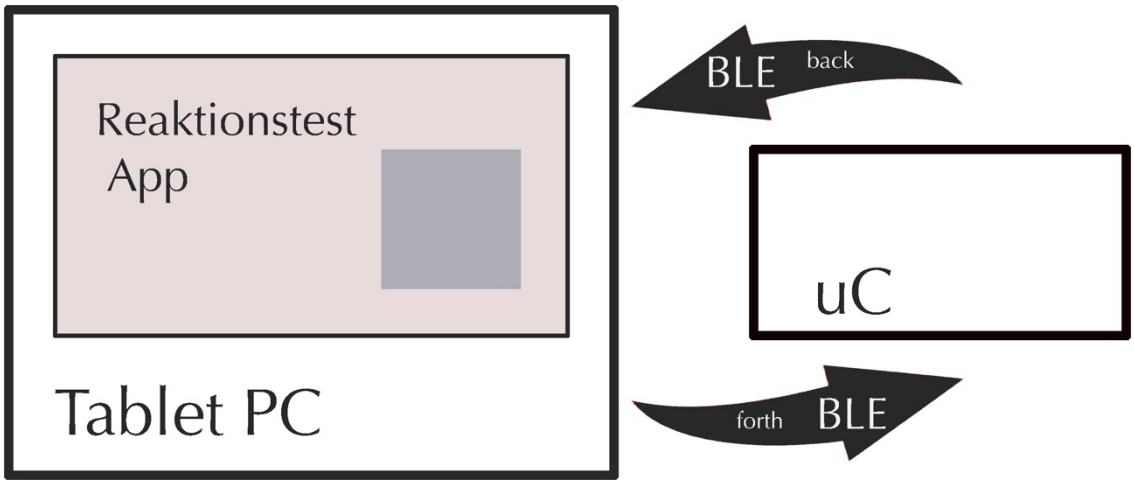

Abbildung 2 Kein wirklich brauchbarer Ansatz: BLE HIN und ZURÜCK geteilt durch 2

Die Realisierung dieser Hin-Zurück-Messmethode (namens BLE2) hat mit dem Code von Brian für 100 Messungen ergeben:
min: 107 ms (=schnellste Übertragungszeit)
max: 245 ms (=langsamste ÜZ)
av: 132 ms (=Durchschnitt)

Für den einfachen Weg wäre die durchschnittliche Übertragungszeit demnach BLE2 = 66 ms.

Verbesserter Ansatz: ein definierter Startpunkt

Um es gleich vorweg zu sagen: Trotz unserer eigentlich sinnvollen Vorüberlegung zur Emittlung der BLE-Laufzeit ist dieser Ansatz völlig gescheitert und hat sogar laut des Studierenden, der ihn, von uns übernommen, nachträglich noch versucht hat als Code zu realisieren, zu **negativen** BLE-Laufzeiten geführt; das hat den Studierenden aber nicht weiter gestört. Dass da etwas nicht stimmen kann, hat er gar nicht in Erwägung gezogen. "Warum, bitte" fragte er uns "soll es keine negativen Laufzeiten geben?" :-(

Um in einer (vermeintlich) verbesserten Messmethode die BLE-Laufzeit vom uC zur App zu ermitteln, könnte man - so unsere Überlegung - folgendermaßen vorgehen (s. Abbildung 3): Es wird ein **definierter Startpunkt für die Zeitmessung** installiert. Hierfür wird über der angehenden "App-Lampe" ein realer Sensor (Photodiode) montiert, der mit einem GPIO des uC verbunden ist. Sobald die Lampe in der App von dunkel nach rot schaltet, wird dies vom uC über den Sensor in vernachlässigbarer Zeit (ein paar Mikrosekunden) registriert, und der uC sendet über BLE ein Datum an die App. Die vergangene Zeit zwischen Rot-Aufleuchten der App-Lampe und Eintreffen des BLE-Datums in der App ist die BLE-Laufzeit. Soweit plausibel. Aber leider hat sich dieser Ansatz entgegen allen Erwartungen ebenfalls als unbrauchbar erwiesen. **Voraussetzung für das Funktionieren dieser Vorgehensweise ist nämlich, dass das Schalten der Lampe von dunkel nach rot auf dem Grafikdisplay des Tablets in "vernachlässigbar" kurzer Zeit - oder in zumindest <u>bekannter</u> Zeit - erfolgt.** Das ist aber leider nicht so, wie durch eine andere Messmethode festgestellt wurde[1]. **Schwer zu glauben, aber das Schalten der Lampe von dunkel nach rot dauert (bei dem verwendeten Huawei-Tablet) länger als die Daten-Übertragung über BLE!** Wie lange das genau dauert, ist nicht bekannt und kann - aufgrund des Fehlens einer zeitpräzisen Synchronisationseinrichtung - auch nicht über einen geeigneten Messaufbau ermittelt werden. Deshalb ist dieser Messaufbau aus Abbildung 3 ebenfalls nicht geeignet, um die BLE-Laufzeit vom uC zur App zu ermitteln. Aber ein Versuch war es wert ...

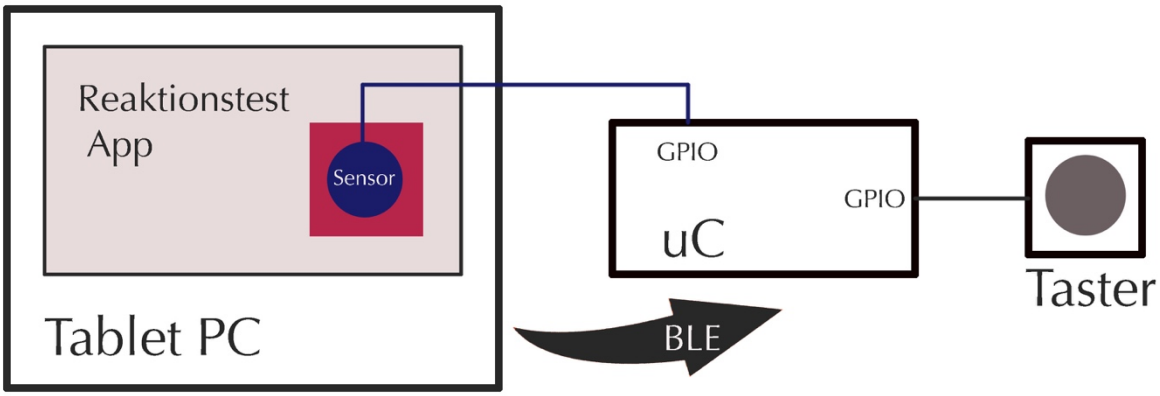

Abbildung 3 Gescheiterter Messaufbau: Ein Sensor am GPIO des uC erkennt den App-Lampenwechsel von Dunkel nach Rot als Startpunkt für die Übertragung des BLE-Datums (so die Theorie)

[1] Hierzu wird im Code der App unmittelbar hintereinander die Lampe von Dunkel nach Rot geschaltet. Dieses Umschalten registriert der Sensor am uC (jedoch nicht sofort, sondern erst nach einiger Zeit, wenn sich nämlich das Umschalten auch auf dem Display auswirkt) und das BLE-Datum wird auf den Weg geschickt. Es hat sich in diesem Messaufbau gezeigt, dass das BLE-Datum schon mehrere Millisekunden früher im uC angekommen ist, bevor der Sensor überhaupt das Umschalten von Dunkel nach Rot registrieren konnte. Eigentlich hätte es - der Erwartung nach - genau andersherum sein sollen: erst registriert der Sensor am uC das Schalten der Lampe und setzt damit den Startpunkt für die Zeitmessung im uC, und erst dann trifft das BLE-Datum ein.

Wie oben schon gesagt, hat dieser (in der Erwartung vermeintlich bessere) Aufbau ergeben, dass die Übertragung eines BLE-Datums von der App zum uC schneller ist als das grafische Umschalten der App-Lampe von Dunkel nach Rot. Bei der Zeitmessung ergeben sich deshalb - wen wundert's - "negative" Laufzeiten.

(Nebenbei: In dem Aufbau des Studierenden gab es die "negativen Laufzeiten" aber nur manchmal bis selten - deshalb haben sie ihn "als statistische Ausreißer" auch weiter nicht interessiert. Bei genauerer Betrachtung seines Codes hat sich jedoch gezeigt, dass dieser elementare Fehler aufwies. :-(

Der nächste Schritt: die Verwendung zweier zeitunkritischer uC

Ein allgemeiner Versuch zur Ermittlung zumindest der BLE-Laufzeit beim uC liegt in der Verwendung zweier zeitunkritischer uC, die über die Verbindung zweier GPIO einen definierten Startzeitpunkt für die Messung erhalten (s. Abbildung 4). Hier sendet uC-1 ein BLE-Datum an uC-2 und schaltet zeitgleich seinen GPIO. Den Startpunkt der Übertragung erfährt uC-2 ebenfalls über seinen GPIO und setzt einen Timestamp. Sobald das Datum über BLE eintrifft, kann über die Zeitdifferenz die BLE-Laufzeit ermittelt werden. Um sicherzustellen, dass der Rückweg (back) durchschnittlich genauso schnell ist wie der Hinweg (forth), kann diese Messung problemlos in zwei Richtungen betrieben werden.

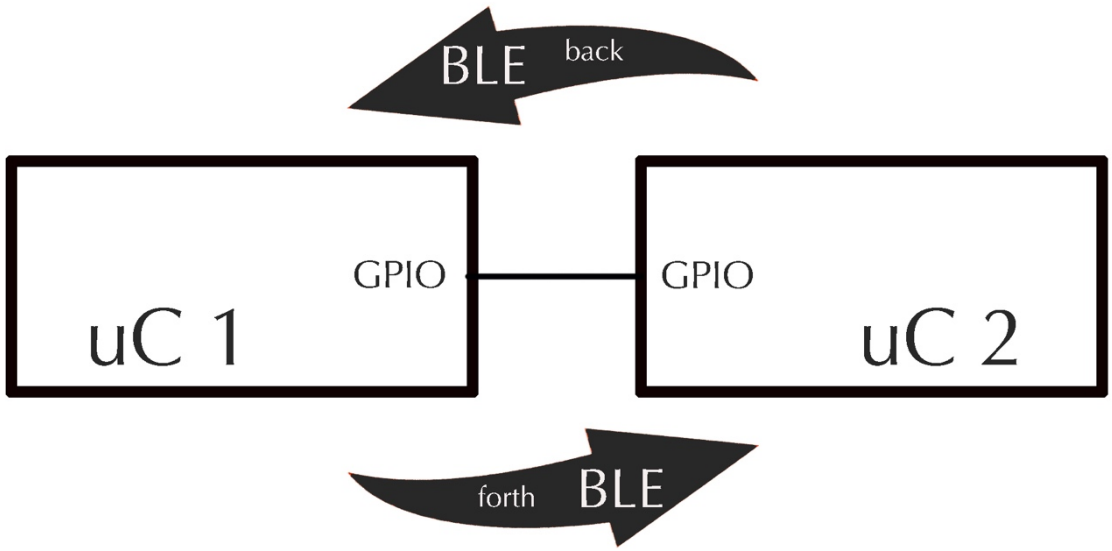

Abbildung 4 Die Verwendung zweier Microcontroller als Notlösung

Die Realisierung dieser Messmethode (namens BLE4) hat für 100 Messungen ergeben:
min: 20 ms (=schnellste Übertragungszeit)
max: 58 ms (=langsamste ÜZ)
av: 28 ms (=Durchschnitt)

Für den einfachen Weg wäre demnach BLE4 = 14 ms. (Schnell ist das nicht gerade.)

Aber es handelt sich hierbei natürlich nur um die BLE-Laufzeit zwischen zwei uC, und nicht um die möglicherweise hardwarebedingt unterschiedliche BLE-Laufzeit von einem uC an eine Tablet-App. Deshalb ist noch ein weiterer Schritt erforderlich. Die Zeitmessungen aus dem Messaufbau von Abbildung 4 (BLE4) müssen mit denen aus Abbildung 2 (BLE2)

verglichen werden. Sind die Laufzeiten für einen Weg in etwa gleich, dann gibt es keinen Unterschied zwischen Hin und Zurück, und das heißt, die BLE-Verarbeitungzeit in einem Tablet ist identisch mit der in einem uC. Sind die ermittelten Laufzeiten aus den Aufbauten aus Abbildung 2 und Abbildung 4 hingegen unterschiedlich, dann repräsentiert dieser Unterschied die langsamere oder schnellere BLE-Verarbeitungszeit in der App (BLE-App).

Zwischenergebnis:

BLE-App = BLE2 - BLE4 = 66ms - 14ms = 52ms

Der ermittelte "BLE-App-Verlust" von 52ms (hier: Huawei-Tablet) gegenüber BLE-uC betrifft aber nur den Empfang von BLE in der App, nicht das Senden, denn das geht, wie aus den Messungen vom Aufbau aus Abbildung 4 erkennbar war: zwar nicht gerade sehr schnell, aber schneller. **Eine genaue, d.h. analytisch begründete Angabe kann aber systembedingt nicht gemacht werden.**

Versuch zur empirischen Ermittlung der Prozesszeit für den Zustandswechsel der App-Lampe

Im vorangegangen Abschnitt wurde gesagt, dass eine genaue, d.h. analytische Aussage über die Prozesszeit des App-Lampenwechsels nicht gemacht werden kann. Deshalb soll hier nun empirisch im Sinne einer Annäherung vorgegangen werden (s. Abbildung 5).

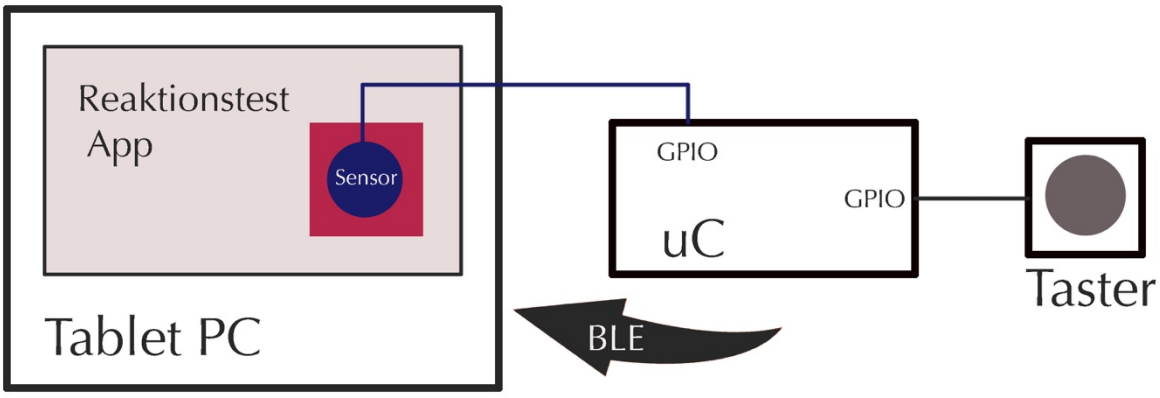

Abbildung 5 Empirsche Ermittlung der App Grafik-Prozesszeit

In diesem Aufbau sendet der uC ein BLE-Datum (BLE5) an die App erst dann, wenn der Sensor das Umschalten der Lampe von dunkel nach rot registriert.

Die Realisierung dieser Messmethode (BLE5) hat für 100 Messungen ergeben:
min: 83 ms (=schnellste Übertragungszeit)
max: 127 ms (=langsamste ÜZ)
av: 105 ms (=Durchschnitt)

Diese BLE5-Zeiten stehen für die zeitlichen Differenzen zwischen "App schaltet Lampe ein" und "BLE-Datum trifft ein". Von diesem Durchschnittswert muss nun noch die oben

ermittelte BLE-App Zeit abgezogen werden, und man hat auf empirischem Weg die durchschnittliche Grafikprozesszeit für das Schalten der App-Lampe.

GraficProcTime = BLE5 - BLE-App = 105ms - 52ms = 53ms

Dieses Ergebnis von durchschnittlich 53 ms Grafikprozesszeit verhält sich plausibel mit der oben gemachten Erfahrung, dass die grafischen Änderungen in der App durchschnittlich "ein klein wenig länger" dauern als der Empfang von Bluetooth LE in einer Tablet-App (hier: Huawei); es kam dabei wie erwähnt zu "negativen" Laufzeiten.

Endergebnis zur BLE-Laufzeitmessung in der Bedeutung für den Reaktionstester und das acaPendulum-Projekt

Bei einem mittels einer App (hier: auf Huawei-Tablet) realisierten Reaktionstester in Verbindung mit einem externen Taster müssen von der ermittelten Reaktionszeit des Probanden - durchschnittlich - 105 ms (=BLE5) abgezogen werden. Dennoch bleibt, dass bei einer Streuung der BLE-Zeit von +/-22ms (s. BLE5 min und max) nicht quantitativ exakt die Reaktionszeit eines Probanden gemessen werden kann. Brians Reaktionszeit beispielsweise beträgt durchschnittlich flotte 140 ms, meine hingegen über 250 ms. Unschwer erkennbar wäre die genannte BLE-Streuung für eine verlässliche Reaktionszeit-Aussage zu groß. (Nebenbei: Wenn sich die Schülerpraktikanten im PSE-Labor zweimal im Jahr mit Brian im Kampf um die bessere Reaktionszeit messen, dann geht es um 1 ms, wer der Sieger ist :-)

Was bleibt, wäre eine qualitative Gruppierung der BLE-Messergebnisse für Reaktionszeiten, wie z.B.

120 - 169	sehr gut
170 - 219	gut
220 - 269	mittel
270 - 319	langsam
320 - ...	zu langsam, Proband ist evtl. krank oder entwicklungsverzögert u.Ä.

Eine exakte Reaktionszeitmessung in Verbindung mit einem physikalischen Taster, den der Proband betätigt, ist nur möglich unter Verwendung eines EZ-fähigen Microcontrollers, oder eventuell eines Raspberry Pi. Eine App in der hier verwendeten Weise mit einem externen Taster, der an einem uC angeschlossen ist, ist nicht geeignet.

Aber wenn doch eine App plus externer Taster für die Reaktionszeitmessung verwendet werden soll, dann läge die Lösung vielleicht in der Installation einer **Uhrensynchronisation (htp)** zwischen uC und Android-App. (Martin von Löwis sagt, das Thema sei nicht leicht, aber machbar ...)

Wie oben schon gesagt, sollten die mit der Bachelorarbeit angestrebten Erkenntnisse zu den BLE-Laufzeiten zwischen einem uC und einer Android-App insbesondere dazu dienen, herauszufinden, ob das acaPendulum-Projekt als App in der Verwendung mit einem externen Taster sinnvoll realisiert werden kann.

Ich denke, ja das physikalische acaPendulum-Projekt kann in der Simulation als App in Verbindung mit einem externen, physikalischen Taster realisiert werden. Auch mit den ziemlich weit streuenden Zeitwerten in der Verwendung für BLE kann es das, denn anders als

beim Reaktionstester geht es bei acaPendulum um das mehr oder weniger gemächliche, kontinuierliche Zuführen von "Energie" auf ein schwingendes, simuliertes Pendel, um dieses entgegen der auftretenden Verluste am Schwingen zu halten. Es handelt sich hierbei um relativ langsame Eingriffe, nicht um schnelle Reaktionszeiten. Zudem geht es beim acaPendulum-Projekt nur sekundär um menschliche Aktionszeiten in der Betätigung eines Tasters, sondern vielmehr um das kognitive Verständnis der Aufgabenstellung und um ein gesundes Wahrnehmen, wann man "in etwa" den Taster drücken muss, und wann man ihn "in etwa" wieder loslassen sollte, damit das Pendel gut schwingen kann. Die hierbei anfallenden Zeitspannen liegen - geschätzt - ohnehin im Bereich mehrerer hundert Millisekunden.