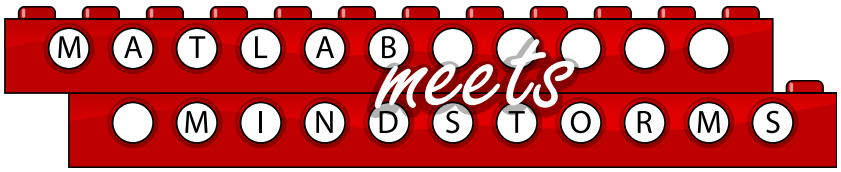
Projekt der Elektrotechnik und Informationstechnik



2. Projektversuch

- Bluetooth und Tastsensor -

15. November 2021

Inhaltsverzeichnis

1 Einleitung

In diesem Versuch wird die Ansteuerung des LEGO Mindstorms EV3 Systems über MATLAB und die Verwendung der RWTH - Mindstorms EV3 Toolbox behandelt. Die Kommunikation zwischen Computer und dem LEGO Mindstorms Programmable Brick erfolgt dabei entweder über eine drahtlose Bluetooth Verbindung oder nutzt eine drahtgebundene USB-Schnittstelle. Dafür wurde ein von LEGO spezifiziertes serielles Kommunikationsprotokoll in die Toolbox eingebettet. Während die USB-Verbindung vom Rechner direkt als serielle Kommunikation etabliert wird, muss dies für die Verwendung von Bluetooth manuell eingerichtet werden. Hierbei wird innerhalb der Bluetooth-Verbindung eine serielle Kommunikation emuliert.
Dieser Versuch erläutert das Erstellen der Verbindung mit USB oder über den Bluetooth Adapter und erklärt die Handhabung der in der RWTH - Mindstorms EV3 Toolbox zur Verfügung gestellten Funktionen zum Aufbau der seriellen Verbindung aus MATLAB heraus. Weiterhin wird anhand einer MATLAB GUI die Verwendung des Tastsensors erlernt.

2 Grundlagen

In diesem Kapitel werden die zur Versuchsdurchführung benötigten Grundlagen erläutert. Dabei wird der Tastsensor kurz vorgestellt und auf die Verbindung zwischen dem Rechner und dem EV3 Brick eingegangen. Nach einer Einführung in das verwendete Kommunikationsprotokoll werden die für diesen Versuch benötigten Befehle der RWTH - Mindstorms EV3 Toolbox erklärt.

2.1 LEGO Mindstorms EV3 Tastsensor

Bei dem LEGO Tastsensor handelt es sich um einen einfachen Schalter. Er verfügt über zwei Modi. Im Ersten gibt er die Zustände *Ein* und *Aus* und im Zweiten wie oft gedrückt wurde zurück. (Abbildung 1).

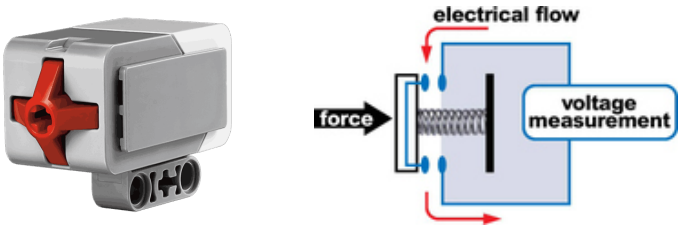


Abbildung 1: EV3 Touch Sensor (LEGO ©)

Im gedrückten Zustand ist der elektrische Stromkreis geschlossen. Wird der Tastschalter losgelassen ist der Stromkreis unterbrochen. Die benötigte Kraft zum Drücken des Schalters beträgt dabei ungefähr 0,34N.

2.2 Serielle Kommunikation

Die Kommunikation zwischen dem Rechner und dem EV3 Brick kann, wie in der Einleitung bereits erwähnt, auf zwei Arten erfolgen - drahtlos über Bluetooth oder kabelgebunden über USB. Dabei bietet die USB Verbindung einen höheren Datendurchsatz, ist aber für mobile Legomodelle aufgrund des Kabels ungeignet. Die Auswahl der zu verwendenden Schnittstelle wird beim Einrichten der Verbindung in MATLAB mit der RWTH - Mindstorms EV3 Toolbox getroffen. Die Abbildung 2 zeigt den prinzipiellen Aufbau der Kommunikationsstruktur.

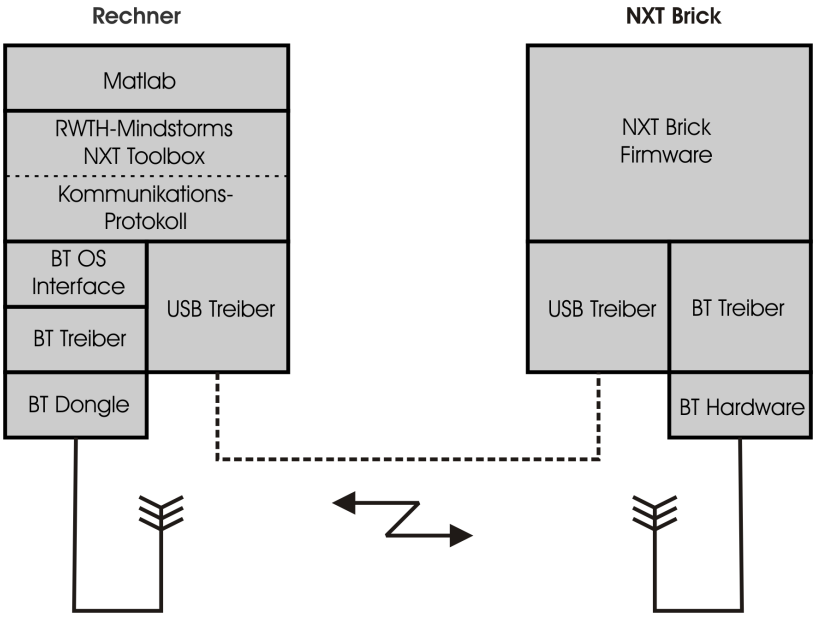


Abbildung 2: Struktur der Verbindung zwischen Computer und NXT bzw. EV3 Brick

Aufbau der Datenpakete

Das Datenpaket, welches zur Kommunikation verwendet wird, ist wie in Abbildung 3 dargestellt aufgebaut. Die Bedeutung der einzelnen Bytes wird im Folgenden erklärt.

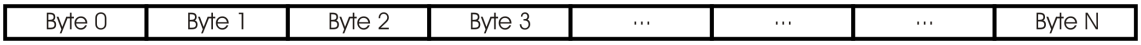


Abbildung 3: Aufbau eines Datenpaketes

Byte 0-1: Größe des Befehls **Byte 2-3:** Befehlszähler, kann beispielsweise dafür genutzt werden gesendete und empfangende Pakete zuzuordnen.

Byte 4: Nutzdaten. Diese Bytes beinhalten Zusatzinformationen für den jeweiligen Befehl (z.B. Frequenz und Länge eines Tons, der abgespielt werden soll).

- `0x00`: Direct command, reply required
- `0x01`: System command, reply required
- `0x02`: Direct command reply
- `0x04`: Direct command reply error
- `0x80`: Direct command, reply not required

- `0x81`: System command, reply not required

Bytes 5-N unterscheiden sich bei Comand- und Response-Befehlen. **Byte 5-6:** Anzahl der erwarteten Rückgabenwerte **Byte 7-N:** Befehle in HEX-Code. Für Befehle die eine Rückgabe erwarten wird der Aufbau des Responses festgelegt. **Byte 5-N(Response):** Response Buffer

Die folgenden Beispiele verdeutlichen den Aufbau der Datenpakete.

Beispiel playTone:
Byte 0: 0xF0 (Länge LSB)
Byte 1: 0x00 (Länge MSB)
Byte 2-3: beliebiger msg-counter
Byte 4: 0x80 (kein Reply notwendig)
Byte 5: 0x00 (keine LVs und GVs = keine Antwort erwartet)
Byte 6: 0x00 (keine LVs und GVs = keine Antwort erwartet)
Byte 7: 0x94 (SOUND-comand)
Byte 8: 0x01 (TONE-subcomand)
Byte 9: 0x81 (Konstante die festlegt, dass das nächste Byte einen Parameter enthält)
Byte 10: Lautstärke in Prozent
Byte 11: 0x82 (Konstante die festlegt, dass die nächsten 2 Bytes einen Parameter enthalten)
Byte 12-13: Frequenz in Hertz
Byte 14: 0x82 (Konstante die festlegt, dass die nächsten 2 Bytes einen Parameter enthalten)
Byte 15-16: Dauer in Millisekunden

batteryValue
Byte 0: 0x08 (Länge LSB)
Byte 1: 0x00 (Länge MSB)
Byte 2-3: beliebiger message-counter
Byte 4: 0x00 (Reply notwendig)
Byte 5: 0x01 Byte 6: 0x00 (Byte 5+6: Erwarte 1 global variable = 1 normaler Rückgabewert)
Byte 7: 0x81 (UI_READ-command)
Byte 8: 0x12 (GET_LBATT-subcommand)
Byte 9: 0x60 (Rückgabewert soll im ersten Byte des Response-Buffers gespeichert werden)

Antwortpaket:
Byte 0: 0x04 (Länge LSB)
Byte 1: 0x00 (Länge MSB)
Byte 2-3: selber message-counter wie im vorher gesendeten Paket
Byte 4: 0x02 (kein Error)
Byte 5: 0x64 (100% Akkuladung)

Testergebnisse während der Entwicklung des Protokolls zeigten, dass die serielle Bluetooth Kommunikation im Gegensatz zur Verbindung mit USB Nachteile beim Datenstreaming aufweist. Einerseits können große Datenpakete den internen ARM Prozessor mit kleinen Zeitdifferenzen erreichen und andererseits tritt eine 30 Millisekunden Zeitverzögerung auf, wenn der Bluecore Chipsatz zwischen dem Sende- und Empfangsmodus wechselt. Um das Problem der Zeitverzögerung des Bluecore Chipsatzes zu verringern, sollte darauf geachtet werden, dass nicht unnötige Paketantworten verlangt werden. Dann wird das häufige Umschalten des Chipsatzes zwischen Sende- und Empfangsmodus und damit die Latenzzeit verringert.

2.3 Erstellen einer Bluetoothverbindung

Während die Verbindung zwischen dem PC und dem EV3 Brick via USB nur durch Einstecken des Kabels und Eingabe der entsprechenden Befehle in MATLAB erfolgt, muss die Bluetoothverbindung zunächst durch die Software des Bluetooth Adapters etabliert werden. Im Folgenden werden die Schritte zum Aufbau dieser Verbindung für Windows und Linux erläutert.

Unter Windows 32 Bit

Zum Aufbau der Verbindung müssen die Geräte als erstes untereinander bekannt gemacht werden. Dazu öffnet man den Bluetooth Verbindungsdialog auf dem Rechner durch Doppelklick auf das Bluetooth Symbol im Systemtray (Taskleiste rechts). In dem sich öffnenden Fenster werden alle bekannten Verbindungen zwischen Rechner und Bluetooth Peripherie angezeigt (Abbildung 4).

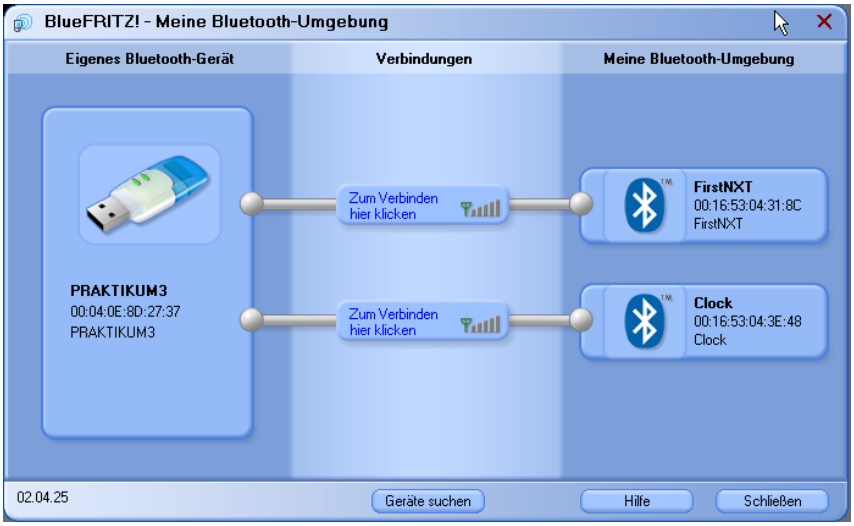


Abbildung 4: Meine Bluetooth-Umgebung

Sind noch keine Verbindungen erstellt worden, so sind die Spalten *Verbindungen* und *Meine Bluetooth-Umgebung* leer. Um den EV3 Brick in die Bluetooth-Umgebung aufzunehmen, muss man ihn zunächst suchen. Dazu ist sicherzustellen, dass der Brick eingeschaltet und Bluetooth aktiviert ist. Klicken Sie danach am PC auf die Schaltfläche *Gerät suchen*. Es öffnet sich ein Suchdialog (Abbildung 5).

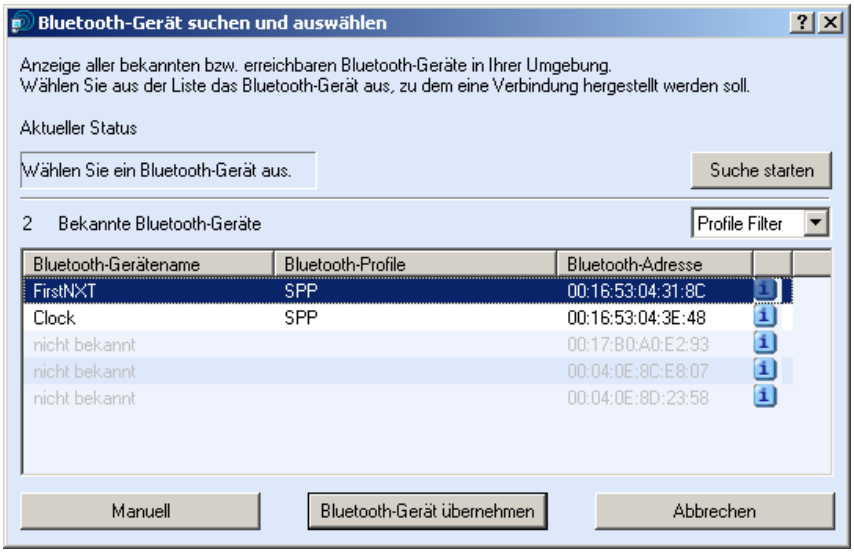


Abbildung 5: BT Suchdialog

Falls die Suche nicht automatisch beginnt oder erneut gestartet werden soll, muss die Schaltfläche *Suchen* betätigt werden. Nach einigen Sekunden werden dann alle sich in

Reichweite des Bluetooth Adapters befindlichen Geräte angezeigt. Markieren Sie den gewünschten EV3 Brick in der Liste und klicken Sie auf *Bluetooth-Gerät übernehmen*. Am Brick ertönt nun ein Signalton und es wird eine vierstellige Kennung im Display angezeigt (standardmäßig 1234). Drücken Sie den orangenen Knopf und geben Sie die Kennung in dem sich öffnenden Dialog am PC ein (Abbildung 12).

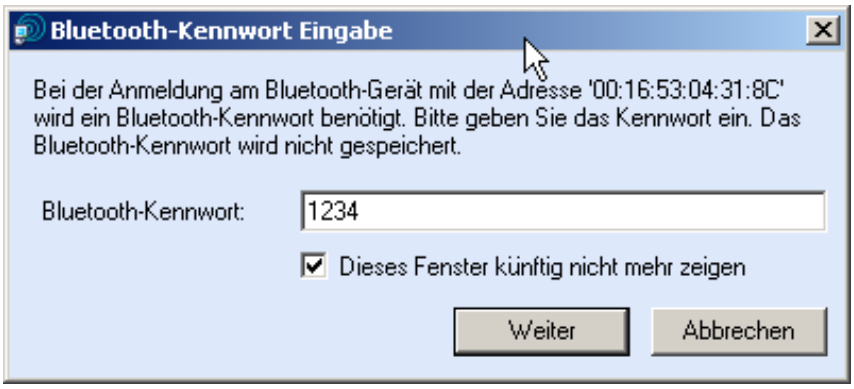


Abbildung 6: Kennworteingabe

Die Verbindung ist jetzt initialisiert und die Geräte verbunden. Am Bluetooth Adapter wird der Status *Verbunden* durch eine zweite grüne LED und am PC durch die Grünfärbung der Verbindungswege signalisiert (Abbildung 7).

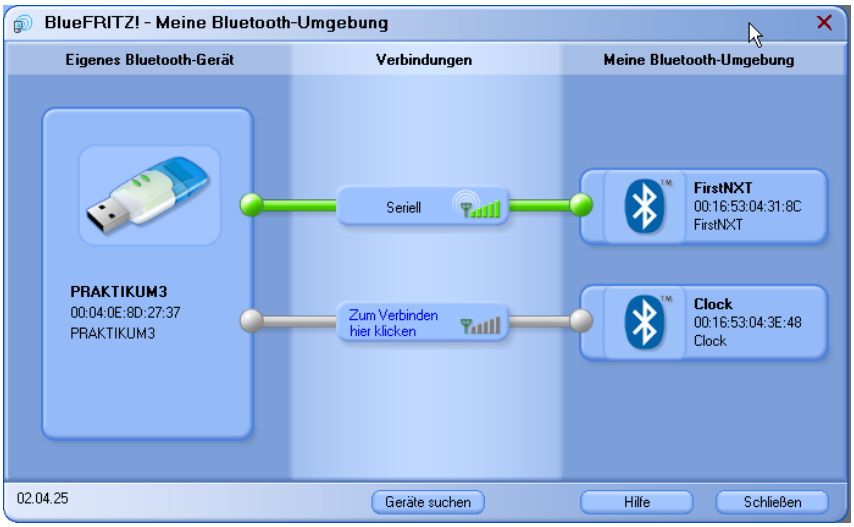


Abbildung 7: Verbindung öffnen

Diese beschriebene Prozedur ist nur für die erste Initialisierung der Verbindung notwendig. Wurden die Geräte einmal miteinander bekannt gemacht, brauchen Sie nur auf das

Verbindungssymbol klicken und im Kontextmenu *Verbinden mit* -> ***Alle*** auswählen (Abbildung 8).

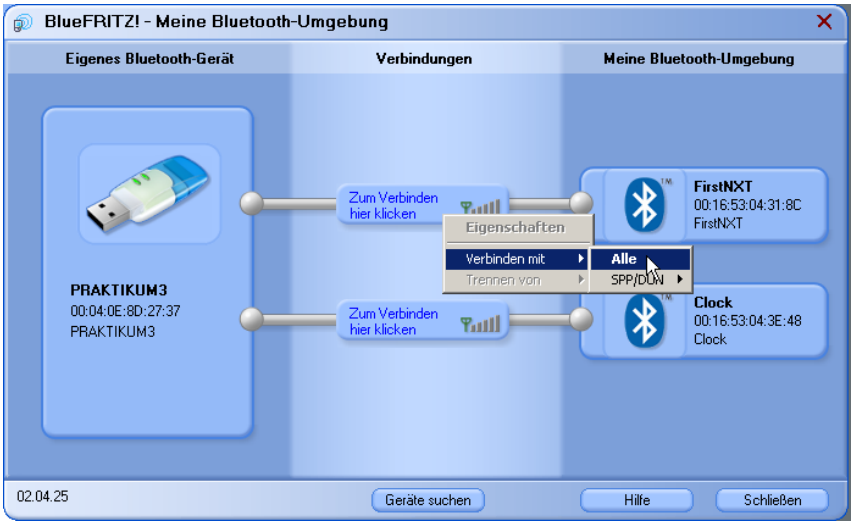


Abbildung 8: Verbindung öffnen

Falls die Verbindung nicht zustande kommt, wählen Sie im Kontextmenu nicht ***Alle***, sondern unter *SPP/DUN* den Port mit der niedrigsten Nummer aus.

Zum Trennen der Verbindung wird ähnlich vorgegangen. Klicken Sie auf das Symbol *Seriell* und in dem erscheinenden Kontextmenu auf *Trennen von* -> ***Alle***.

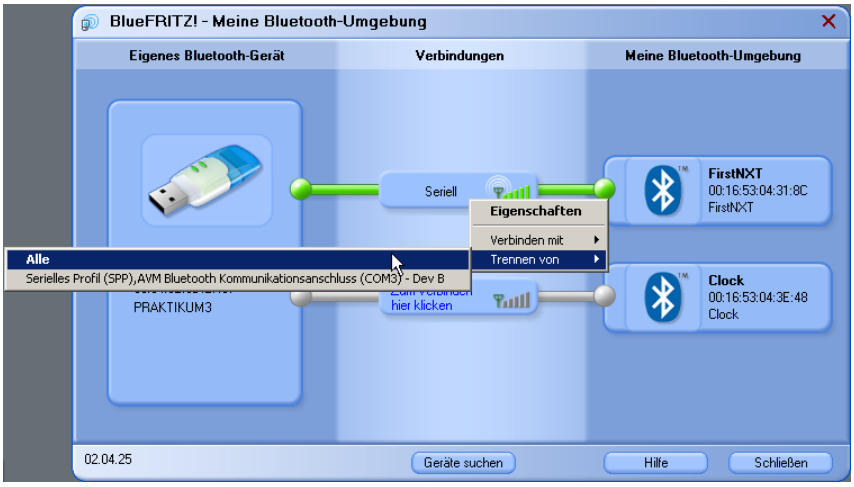


Abbildung 9: Verbindung trennen

Es ist auch möglich, den Rechner mit zwei oder mehreren EV3 Bricks zu verbinden. Dazu

müssen nach der Verbindung mit dem ersten Brick die oben beschriebenen Schritte für jeden weiteren Brick wiederholt werden.

Unter Windows 64 Bit

Zum Aufbau der Verbindung müssen die Geräte als erstes untereinander bekannt gemacht werden. Dazu öffnet man den Windows eigenen Bluetooth Verbindungsdialog auf dem Rechner durch Doppelklick auf das blaue Bluetooth Symbol im Systemtray (Taskleiste rechts). In dem sich öffnenden Fenster werden alle bekannten Verbindungen zwischen Rechner und Bluetooth Peripherie angezeigt (Abbildung 10).

Abbildung 10: Bluetooth-Manager

Sind noch keine Verbindungen erstellt worden muss der EV3 Brick erst noch hinzugefügt werden. Um den EV3 Brick in die Bluetooth-Umgebung aufzunehmen, muss man ihn zunächst suchen. Dazu ist sicherzustellen, dass der Brick eingeschaltet und Bluetooth aktiviert ist. Klicken Sie danach am PC auf die Schaltfläche *Gerät suchen*. Es öffnet sich ein Suchdialog (Abbildung 11).

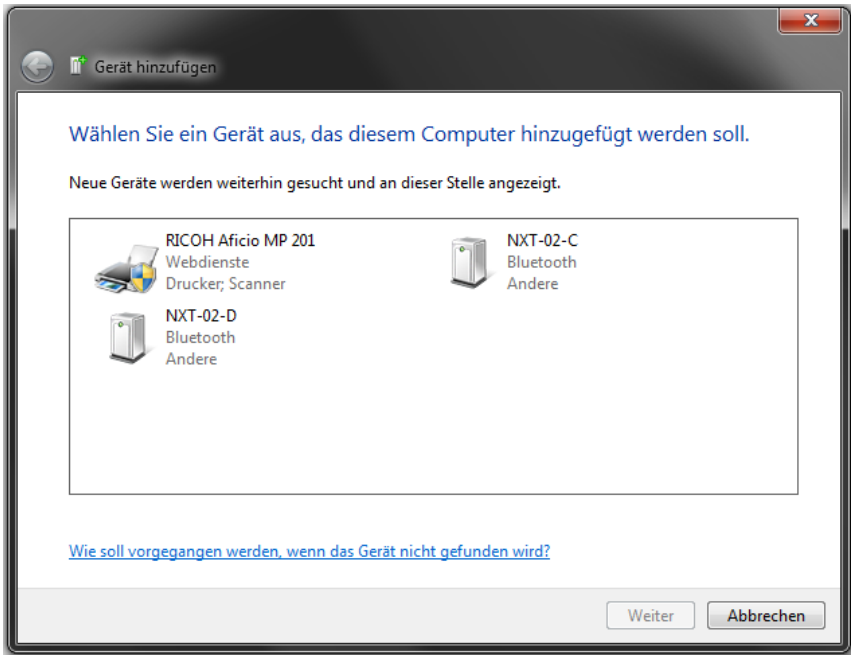


Abbildung 11: BT Suchdialog

Nach einigen Sekunden werden alle sich in Reichweite des Bluetooth Adapters befindlichen Geräte angezeigt. Markieren Sie den gewünschten EV3 Brick in der Liste und klicken Sie auf *Weiter*. Am Brick ertönt nun ein Signalton und es wird eine vierstellige Kennung im Display angezeigt (standardmäßig 1234). Drücken Sie den orangenen Knopf und geben Sie die Kennung in dem sich öffnenden Dialog am PC ein (Abbildung 12).

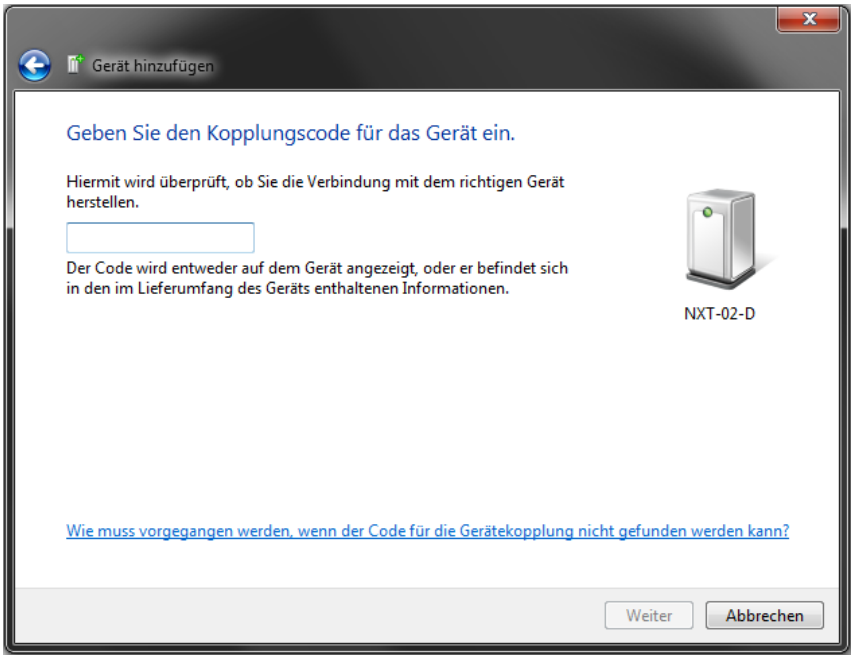


Abbildung 12: Kennworteingabe

Diese beschriebene Prozedur ist nur für die erste Initialisierung der Verbindung notwendig. Wurden die Geräte einmal miteinander bekannt gemacht, können sie den EV3 Brick sofort in Matlab verwenden.

Es ist auch möglich, den Rechner mit zwei oder mehreren EV3 Bricks zu verbinden. Dazu müssen nach der Verbindung mit dem ersten Brick die oben beschriebenen Schritte für jeden weiteren Brick wiederholt werden.

Unter Linux (Livesystem)

Das Linux Livesystem verwendet eine betriebssystemeigene Software. Wenn der Bluetooth Adapter eingesteckt und ein eingeschalteter EV3-Brick mit aktiviertem Bluetooth in der Nähe ist, kann man die Verbindung durch ausführen des *connect*-Skripts herstellen. Das Skript startet man entweder durch Doppelklick auf das *Connect EV3*-Symbol auf dem Desktop oder indem man in der Kommandozeile (im Menü unter *Applications -> Accessories -> Terminal* oder durch drücken von *Strg + Alt + T*) den Befehl

```
connect
```

eingibt.

Folgende Schritte müssen für eine erfolgreiche Verbindung in der angegebenen Reihenfolge ausgeführt werden:

1. EV3 einschalten
2. connect-Skript mit einer dieser Methoden aufrufen:
   - Bluetooth-Symbol auf dem Desktop doppelklicken
   - Kommandozeile: `connect`
3. Den Anweisungen auf dem Bildschirm folgen
   a) Warten bis der EV3 eingeschaltet ist
   b) Enter drücken, um den Scanvorgang zu starten
   c) Warten bis der Scanvorgang abgeschlossen ist
   d) Im Menü die Zahl des gewünschten EV3 ablesen und eingeben oder mit 0 erneut scannen
   e) Pairing request ("Connect") am EV3 annehmen und PIN 1234 (per default schon voreingestellt) verwenden
   f) Am PC Enter drücken, um das Pairing fortzusetzen
   g) Der Pairingversuch wird bis zum Erfolg wiederholt oder kann vom Nutzer abgebrochen werden (Fenster schliessen)
   h) Bei Erfolg wird angezeigt, an welche Geräteschnittstelle der EV3 verbunden wurde (normalerweise `/dev/rfcomm0`) sowie wie man die Verbindung nun in Matlab nutzen kann
4. Das Fenster kann als Referenz im Hintergrund geöffnet bleiben, um sich die Verknüpfung von EV3-Name und Schnittstelle zu merken.

Es ist wie unter Windows möglich, Verbindungen zu mehreren Bricks herzustellen. Hierfür muss für jede weitere Verbindung das Skript erneut ausgeführt werden und während der Ausführung jeweils der EV3-Brick gewählt werden, welcher verbunden werden soll.

Sollte die Verbindung einmal fehlerhaft sein oder der EV3 mit einem andern PC verbunden werden, kann eine bestehende Verbindung getrennt werden. Dies kann mit Hilfe des *disconnect*-Skripts ausgeführt werden. Das Skript startet man entweder durch Doppelklick auf das *Disonnect EV3*-Symbol auf dem Desktop oder indem man in der Kommandozeile den Befehl

```
disconnect
```

eingibt.

Zur korrekten Bedienung des Skripts folgende Anweisungen befolgen:

1. disconnect-Skript mit einer dieser Methoden aufrufen:

- Durchgestrichenes Bluetooth-Symbol auf dem Desktop doppelklicken
- Kommandozeile: `disconnect`

2. Den Anweisungen auf dem Bildschirm folgen
   a) In der Liste den zu trennenden EV3 anhand seiner zugewiesenen Geräteschnittstelle `/dev/rfcommX` oder MAC-Adresse identifizieren
   b) Die entsprechende Zahl eingeben und mit Enter bestätigen
   c) Sicherstellen, dass der EV3 nicht mehr in Matlab etc. verwendet wird und eventuell zuerst dort trennen

Danach kann der EV3 mit diesem oder einem anderen PC gemäss der Verbindungsanleitung neu verbunden werden.

2.4 MATLAB-Grundlagen

2.4.1 If-Anweisung

Eine *if*-Anweisung testet, ob eine vorgegebene Bedingung wahr oder falsch ist und führt entsprechend des Ergebnisses entweder einen Befehl aus oder nicht.

```
if x > 0
  y = 1;
else
  y = 0;
end
```

In diesem Beispiel wird überprüft, ob die Variable x einen Wert größer als 0 besitzt. Ist dies der Fall, wird y auf den Wert 1 gesetzt. Ist x nicht größer als 0, so bekommt y den Wert 0. Lässt man den *else*-Teil weg, so ändert y seinen Wert nur falls $x > 0$ – anderenfalls behält es den Wert, den es vor der *if*-Anweisung hatte:

```
if x > 0
  y = 1;
end
```

2.5 EV3 Grundlagen

In diesem Abschnitt werden die EV3 und die Sensor Klassen RWTH - Mindstorms EV3 Toolbox vorgestellt. Zusätzlich können weitere Erläuterungen und Hilfen zu den einzelnen Eigenschaften und Methoden der Klassen aus der MATLAB Hilfe entnommen werden.

- Klassenname: `Sensor`
  - Eigenschaften
    - Port (privat)
  - Methoden
    - connect (privat)
    - disconnect (private)
    - reset (public)
    - setProperties (public)
- Klassenname: `EV3`
  - Eigenschaften
    - Port (privat)
  - Methoden
    - connect (privat)
    - beep (public)
    - disconnect (private)
    - playTone (public)
    - setProperties (public)
    - stopAllMotors (public)
    - stopTone (public)
    - tonePlayed (public)

Dies ist eine sog. High-level class die wir als zentrale Klasse benutzen werden, um mit dem physikalischen EV3 Brick zu kommunizieren. Sie liefert ein Interface für den Brick.

2.5.1 Eigenschaften der Objekte

Die Eigenschaften beschreiben die individuelle Instanz einer Klasse (also ein Objekt). Sie enthalten Informationen über den aktuellen Zustand des Objekts. Die Parameterwerte der Eigenschaften können direkt beim Erstellen der Objekte durch den Klassenkonstruktor gesetzt werden. Die Werte der Eigenschaften können über den Eigenschaftsnamen und den dazugehörigen Parameterwert gesetzt werden.

```
Standard EV3 Objekt
 Eigenschaften
   motorA[,B,C,D]
   sensor1[,2,3,4]
   debug           - Debug modus kann ein, aus oder auf erweitert geschaltet
   batteryMode     - Modus in dem der Ladestand der Batterie angezeigt wird
   Dependent
   batteryValue    - Ladestand Batterie
   get-only
   isConnected     - ist das EV3-Objekt (der virtuelle Brick) mit einem
   commInterface   - Interface des Kommunikation-Layers

 Methoden:
   connect         - Verbindet  das EV3-Objekt and seine Motoren-/Sensor
   disconnect      - Schließt die Verbindung
   setProperties   - Setzt mehrere EV3 Eigenschaften
   Brick functions
   beep            - Spielt einen 'beep' Ton am Brick ab
   playTone        - Spielt einen mit Parametern spezifizierten Ton am Brick ab
   stopTone        - Stoppt den derzeit gespielten Ton
   tonePlayed      - testet ob derzeit ein Ton abgespielt wird.
```

Neben der Parameterzuweisung beim Erzeugen eines Objekts durch den Konstruktor, können die Werte der Eigenschaften auch durch den Punktoperator (.) oder die get- und set-Methoden gesetzt werden.
Zum Beispiel

```
handle.debug = 1
```

2.5.2 EV3 Objekt

Ein Objekt dieser Klasse nennen wird im weiteren Verlauf virtuellen Brick oder Handle, und nutzen es zum Verwalten unseres EV3.

Beim Erstellen eines EV3 Objektes ist es für die weitere Verwendung wichtig, ihm einen Bezeichner (handle) zuzuweisen.

```
handle = EV3()
```

```
>> handle = EV3()
   EV3 with properties:

       Writeable
               batteryMode:  'Percentage'
               debug:  0

       Read-only
```

```
            batteryValue:  0

      Devices
                    motorA:  [Type: Unknown]
                    motorB:  [Type: Unknown]
                    motorC:  [Type: Unknown]
                    motorD:  [Type: Unknown]
                   sensor1:  [Type: Unknown]
                   sensor2:  [Type: Unknown]
                   sensor3:  [Type: Unknown]
                   sensor4:  [Type: Unknown]
```

Der virtuelle EV3 Brick selbst besitzt wiederum virtuelle Sensor- und Motorenobjekte. Motor Objekte werden in Versuch 4, und Sensoren in Abschnitt 2.7 näher erläutert.

2.6 MATLAB - Kommunikationsfunktionen

In diesem Kapitel werden die Funktionen zur Kommunikation mit dem EV3 Brick vorgestellt. Die betriebssystem-spezifischen Unterschiede im Kommunikationsweg werden innerhalb der Funktionen behandelt und müssen vom Anwender nicht beachtet werden. Anders verhält es sich allerdings mit der Art der Verbindung - hier ergeben sich einige wenige Unterschiede, die beim Aufbau der Verbindung beachtet werden müssen.

2.6.1 Verbindung über USB

handle.connect()

Verbindet das EV3-Objekt and seine Motoren-/Sensor Objekte mit dem physikalischen Brick via Bluetooth oder USB. Die Verbindung über USB gestaltet sich als relativ einfach:

```
handle.connect('usb')
```

Die Funktion versucht eine Verbindung über USB zum EV3 Brick herzustellen, dabei wird sich der Rechner mit dem ersten gefundenen Gerät verbinden. Diese Art des Verbindungsaufbaus ist eine sehr einfache und schnelle, aber sehr eingeschränkte Möglichkeit zum Herstellen der Kommunikation. Trotzdem kann, falls kein EV3 über USB angeschlossen ist, mit der Übergabe eines anderen Parameters auch eine Bluetooth-Verbindung aufgebaut werden (siehe Kap. 2.5.2). Es ist dabei wichtig darauf zu achten, dass jeder physikalische Brick immer nur mit einem virtuellen Brick verbunden ist, sonst kann es zu unvorhersehbaren Problemen kommen. Dies gilt natürlich für Bluetooth- und USB-Verbindungen gleichermaßen.

handle.disconnect()

Mit dieser Funktion kann ein virtueller Brick von einem physikalischen getrennt werden, um die Eindeutigkeit der Verbindungen zu gewährleisten. Falls ein handle mit `delete` gelöscht wird, wird `disconnect` automatisch ausgeführt.

Debug Modus

Im Debugmodus 1, wird immer wenn ein Befehl über die Konsole an den EV3 übergeben wird, ein Feedback über den aufgerufenen Befehl in der Konsole zurückgegeben.

```
handle.debug = 1/'on'/true
```

Im Debugmodus 2 wird der Debugmodus auch für die lower layers (Tieferen Kommunikationslayers) aktiviert, sodass jedes gesendete und empfangene Bluetooth-Paket in der Konsole ausgegeben wird.

```
handle.debug = 2
```

2.6.2 Bluetooth Verbindung

Zum Erstellen einer Bluetooth Verbindung wird auf die selben Funktionen wie bei USB zurückgegriffen. Allerdings müssen der connect-Funktion mehr Informationen über die Schnittstelle mitgegeben werden. Da über den Bluetoothkanal eine serielle Schnittstelle emuliert wird, muss MATLAB ihre Spezifikationen kennen. Wie diese Details eingestellt und an die EV3 Funktionen übergeben werden, wird im Folgenden erklärt.

handle.connect()

Genau wie bei der USB-Verbindung wird dieser Befehl auch für den Verbindungsaufbau bei Bluetooth verwendet.

```
handle.connect('bt', 'serPort', '/dev/rfcomm0')
```

Wie leicht zu sehen ist, ändert sich der 1. Parameter von 'usb' zu 'bt'. Weiterhin wird in typischer Matlab Manie(sog. Matlab Input Parser), erst der 'Parametername' und dann der zu setzende 'Value' angegeben. Der Serielle Port der Bluetoothverbindung lautet bei Windows 32 Bit systemen normalerweise 'COMx' wobei die Zahl x aus der Bluettoth Umgebung ausgelesen werden muss. Unter Linux muss aus der Konsole mit der die Bluetooth Verbindug aufgebaut wurde, die Nummer X des serPort ausgelesen werden: '/dev/rfcommX'.

2.7 MATLAB - EV3 Funktionen

Tastsensor

Genau wie die EV3-Objekte dienen Sensor-Objekte dazu, die physikalischen Sensoren einfach mit einem virtuellen Sensor zu bedienen. Beim Initialisieren eines EV3 Objektes werden automatisch die nötigen Motor und Sensorobjekte erstellt.

Die Sensor-Objekte sind als Eingenschaften (properties) des EV3 handles gespeichert und können wie folgt aufgerufen werden:

```
handle.sensorX mit X aus {1,2,3,4}
```

```
>>  handle.sensor1
    Sensor with properties:

        Writeable
                 mode:  Undefined
                debug:  0

        Read-only
                value:  0.0
                 type:  Unknown
```

Undefined entspricht dem Standartwert in der Firmware des Bricks. In der Dokumentation mit `help sensor.PROPERTYNAME` zu finden.

In diesem Versuch werden wir nur einen Sensor brauchen, einen Touchsensor. Der Touchsensor lässt sich wiederum in 2 Modi betreiben:

Bumps: in diesem Modus zählt der Sensor in Value, wie oft er schon gedrückt wurde

```
handle.sensor1.mode = DeviceMode.Touch.Bumps
```

Pushed: Hier gibt der Sensor einfach den Wert 1 (ist gedrückt) oder 0 (nicht gedrückt) in Value zurück.

```
handle.sensor1.mode = DeviceMode.Touch.Pushed
```

Zum Abfragen der Sensordaten, kann man einfach die Eigenschaft value des Sensors abfragen.

```
[Wert] = handle.sensor1.value
```

Bei den weiteren Sensortypen verwendet man grundsätzlich die selbe Syntax, diese werden in den folgenden Versuchen weiter erläutert.

playTone

Die Funktion `playTone` ermöglicht dem EV3 Brick einen Signalton zu spielen. Als Funktionsparameter sind hierzu die Lautstärke in Prozent, die Tonfrequenz in Hertz und die Zeitdauer in Millisekunden anzugeben:

```
playTone(Lautstärke,Frequenz,Dauer)
```

Der mögliche Frequenzbereich liegt zwischen 250 und 10000Hz.
Beispiel: `playTone(50 ,880, 500)`. In diesem Fall wird ein Signalton bei 50% der maximalen Lautstärke und einer Frequenz 880Hz mit der Tonlänge von 500ms ausgegeben.

Hinweis: Die angegebene Zeitdauer unterbricht nicht den MATLAB Programmablauf. Sollen zwei Töne hintereinander gespielt werden, muss eine manuelle Pause durch die MATLAB Funktion `pause(secs)` angegeben werden.

2.8 Übertragungsglied Integrator

Als Integrator bezeichnet man in der Systemtheorie ein Übertragungsglied, welches integrierendes Verhalten aufweist. Das heißt, dass ein Eingangssignal ständig zeitlich integriert wird und der aktuelle Integralwert am Ausgang anliegt. Die Gleichung (1) gibt dieses Verhalten mathematisch wieder.

$$y(t) = \int_0^\tau u(t)dt \tag{1}$$

Eine charakteristische Eigenschaft von Übertragungssystemen ist die Sprungantwort. In Abbildung 13 ist die Sprungantwort $y(t)$ des Integrators auf den Einheitsprung $u(t)$ dargestellt, welche das integrierende Verhalten anschaulich verdeutlicht.

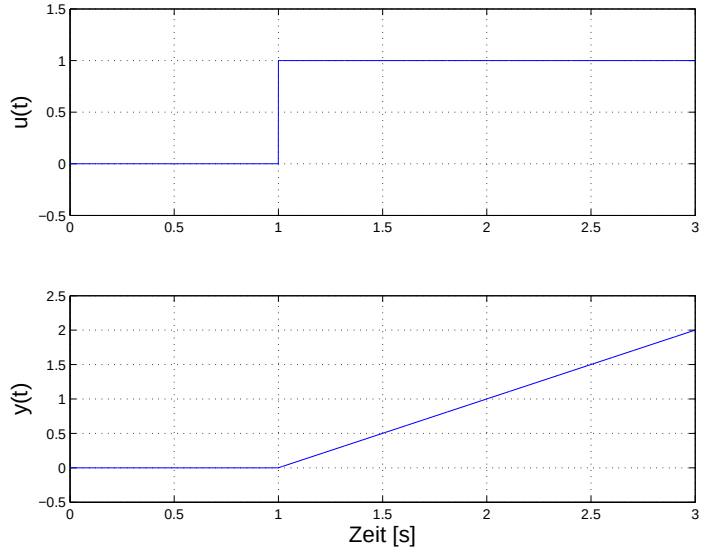


Abbildung 13: Sprungantwort eines Integrators.

Während man in der Elektronik mit z.B. Operationsverstärkern und einigen anderen Bauteilen kontinuierlich arbeitende Integratoren aufbauen kann, ist dies mit digitaler Technik nicht möglich. Hier muss man auf zeitdiskrete Integratoren ausweichen, da der digitalen Technik immer eine gewisse Taktzeit zugrunde liegt. Dazu muss das Integral in Gleichung (1) durch eine Summation ersetzt werden und ändert sich somit zu:

$$y(k) = T \cdot \sum_{i=1}^{k} u(i) = T \cdot u(1) + T \cdot u(2) + ... + T \cdot u(k) \quad . \tag{2}$$

Hierin ist k die Anzahl der Abtastschritte und T gibt die Abtastzeitkonstante an. Diese Gleichung ist durch die Abbildung 14 verdeutlicht. In dem oberen Graphen beschreibt die glatte Linie das Signal $u(t)$ vor der Abtastung - die Größe $u(k)$ gibt den abgetastete Verlauf wider (Sample and Hold). Der unter Graph zeigt die Antwort des zeitdiskreten Integrators auf die Eingangsfolge $u(k)$.
Durch Ersetzen der Zeitkonstante T mit einer allgemeinen Variablen a kann der zeitdiskrete Integrator zu einem frei skalierbaren diskreten Integrator abgeändert werden.

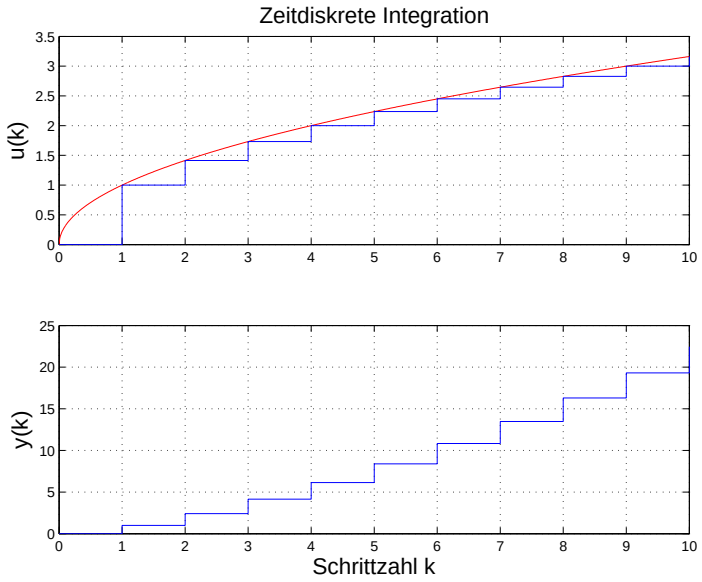


Abbildung 14: Zeitdiskrete Integration einer Wurzelfunktion.

3 Durchführung

Bei Unsicherheit über die Nutzung bestimmter Befehle kann man immer über `help BEFEHL` die Hilfe von Matlab in der Konsole aufrufen, über `doc BEFEHL` kann ma die Dokumentation durchsuchen, dies funktioniert auch für Parameter und alle EV3 bezogenen Methoden Funktionen und Eigenschaften.

3.1 Öffnen und Testen der seriellen Verbindung

Dauer: ca. 30 Minuten

a) Stellen Sie die Bluetooth Verbindung wie in Kapitel 2.3 beschrieben her. Achten Sie dabei auf das Bluetooth Zeichen im Display des Bricks (links oben).
(ca. 5min)

b) Öffnen Sie MATLAB und wechseln Sie in das Verzeichnis in dem Sie die Praktikumsdateien abgelegt haben. Öffnen Sie dort das Verzeichnis dieses Versuchs. Erstellen Sie die serielle Verbindung mit den in Kapitel 2.6 beschriebenen Funktionen - Nutzen Sie dazu die Kommandozeile.
(ca. 10min)

c) Um die erfolgreiche Verbindung zwischen Matlab und dem EV3 Brick zu testen, senden Sie den Befehl `beep`. Für die Verwendung des Befehls schauen Sie in diese Anleitung oder nutzen Sie die EV3 Toolbox Hilfe in MATLAB.
(ca. 5min)

d) Schalten Sie nun den Debug Modus der EV3 Toolbox ein, sodass die Paketinhalte wiedergegeben werden. Senden Sie den Befehl `playTone` mit verschiedenen Frequenzen und Längen und beachten Sie dabei die sich ändernden Paketinhalte. Protokollieren Sie die Befehlsfolgen für die Frequenzen 500Hz, 1500Hz bei den Längen 0,5s, 1s, 2s, jeweils für 100% und 50% der maximalen Lautstärke.
(ca. 10min)

Bei 100% Lautstärke

Frequenz\Dauer	0,5s	1,0s	2,0s
500Hz			
1500Hz			

Bei 50% Lautstärke

Frequenz\Dauer	0,5s	1,0s	2,0s
500Hz			
1500Hz			

e) Führen sie jetzt den Befehl `beep` aus. Was fällt Ihnen auf, wenn sie nun die gesendeten Bytes mit den Ergebnissen aus den Vorherigen Tabellen vergleichen?

Protokollieren sie ihre Ergebnisse kurz.
(ca. 2min)

3.2 Sinusverfolgung

Dauer: ca. 220min

Einführung

Ziel des Versuchs ist es, zwei Funktionen, *touchGetYPos.m* und *touchPlot.m*, so zu programmieren, dass sich eine Sinuskurve mit dem Tastsensor verfolgen lässt. Die Sinuskurve durchläuft das Fenster von rechts nach links. Indem man den Taster drückt, steigt der y-Wert des roten Punktes und während der Taster nicht gedrückt ist, fällt der y-Wert des roten Punktes. Am Ende erhält man eine Ausgabe über die Genauigkeit der Verfolgung und über die Schaltzustände des Tastsensors über der Zeit erhält.
Beim Ausführen der Funktion *touchStart.m* wird eine Bluetoothverbindung aufgebaut, ein Tastsensor initialisiert und die in Abbildung 15 gezeigte MATLAB GUI aufgerufen. Über diese wird der Versuch gesteuert.

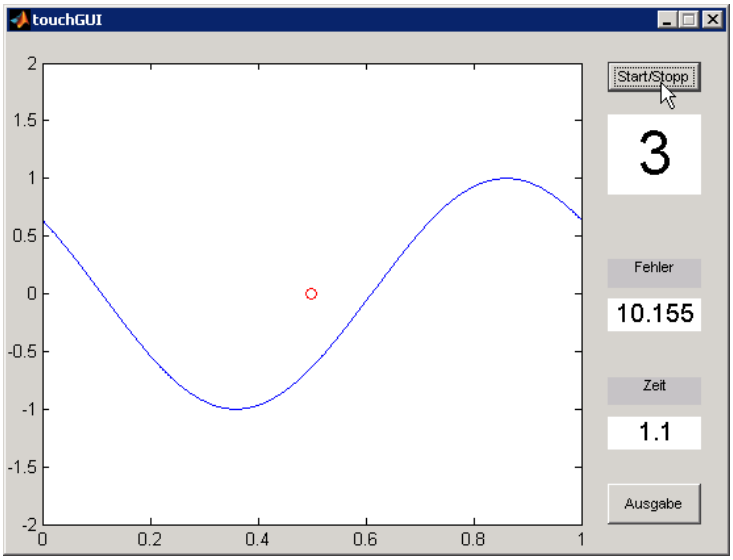


Abbildung 15: Die MATLAB GUI

Die Funktionsweise der GUI ist in Abbildung 16 dargestellt. Nach Drücken des Schalters *Start/Stopp* wird als erstes die Ausgabe initialisiert. Bei der ersten Ausführung wird eine Sinuskurve eingeblendet und ein dreisekündiger Countdown gestartet. Bei

erneuter Ausführung nach einem Programmende wird die Sinuskurve und der Fehler- und Zeitzähler zurückgesetzt und auch wieder der Countdown gestartet. Ein erneute Betätigung des *Start/Stopp* Schalters beendet den Countdown. Nach Ablauf der drei Sekunden wird die Funktion *touchGetYPos.m* aufgerufen, welche die aktuelle vertikale Position des roten Punktes berechnen soll (Aufgabe a). Nach der Ausführung dieser Funktion springt das Programm wieder zurück, verschiebt die Sinuskurve um einen Zeitschritt nach links und gibt die neue Position grafisch aus sowie auch die aktuelle Zeit und den Fehler. Wurde zwischendurch wieder der Schalter *Start/Stopp* betätigt endet die Programmausführung. Nach drei Sinusperioden wird das Programm automatisch gestoppt.

Der Schalter *Ausgabe* ruft, sofern Daten durch eine vorherige Sinusverfolgung gespeichert sind, die Funktion *touchPlot.m* auf. Diese gibt zunächst einen Graphen mit dem zeitlichen Verlauf der drei Sinusperioden aus. Wurde die Sinusverfolgung vorher nicht gestartet, wird bei Aufruf eine Fehlermeldung ausgegeben, dass noch keine Daten gespeichert sind. Eine erneute Betätigung des Tasters schließt den Graphen und führt die Funktion erneut aus. Die Bearbeitung dieser Funktion wird in Aufgabe b vorgenommen.

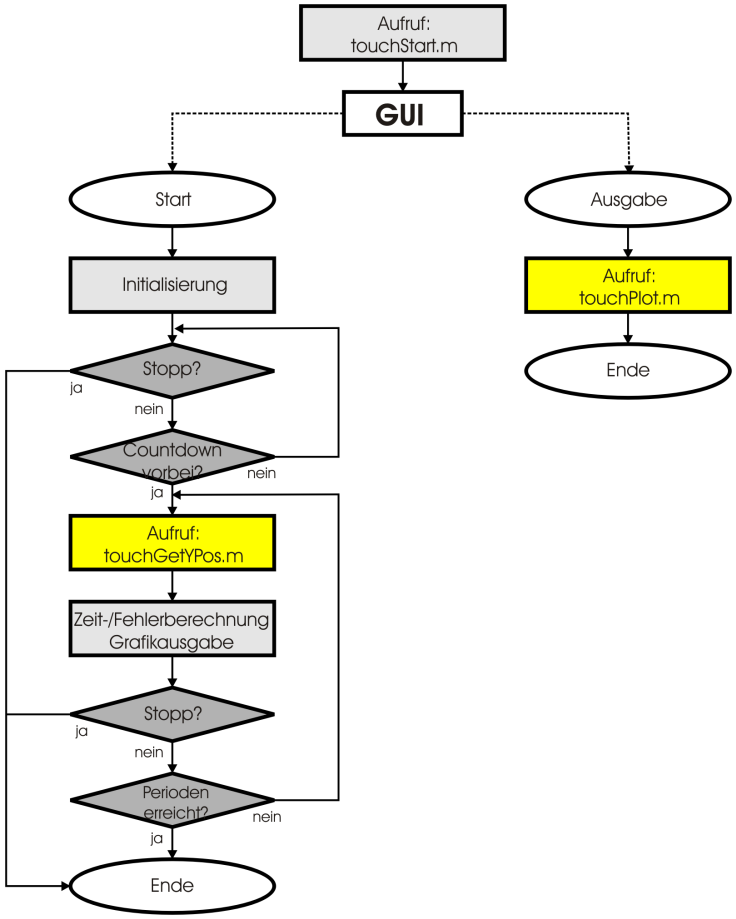


Abbildung 16: Flussdiagramm der Programmausführung

Hinweise:
Die Erstellung von Quellcode erfolgt nur an den markierten Stellen in den beiden zu bearbeitenden Dateien *touchGetYPos.m* und *touchPlot.m*:

- *touchGetYPos.m*: Hier soll die Position des roten Punktes berechnet werden (Unterpunkt b).

- *touchPlot.m*: Hier soll die Ausgabe generiert werden (Unterpunkt c).

Aufgaben

a) Nun soll der rote Punkt mit einem LEGO Tastsensor gesteuert werden. Schließen Sie dazu einen Sensor an den Port `SENSOR_1` des Bricks an. Führen Sie dann in MATLAB die Funktion *touchStart.m* aus. Es erscheint das oben beschriebene

Fenster. Öffnen Sie danach die Datei *touchGetYPos.m* und schauen Sie sich die kurze Beschreibung unter dem Funktionskopf an. Die Funktion soll den aktuellen Status des Touch-Sensors (gedrückt 1 oder nicht gedrückt 0) in der Variable *switchstate* sowie die gewünschte Y-Position des roten Punktes in der Variable *value* zurückgeben. Dazu muss der Tastsensor ausgelesen werden. Überlegen Sie sich jetzt, wie die Y-Position unter Zuhilfenahme des bereitgestellten Vektors *actualval_vector* und der Variable *cyclecount* berechnet werden kann. Der Vektor beinhaltet nacheinander die Y-Positionen vor dem jeweiligen Aufruf der Funktion und *cyclecount* die Anzahl der bisherigen Aufrufe. Ziel ist es nun, die Berechnung der Y-Position so zu gestalten, dass es durch Drücken oder Loslassen des Tasters möglich wird, mit dem roten Punkt dem Sinussignal gut zu folgen. Begrenzen Sie in der Funktion den Wertebereich der Y-Position auf [-1,5;1,5].
(ca. 110min)

Tipps:

- Die Matlab Syntax unterscheidet sich teilweise stark von der C Syntax. Nutzen Sie daher die Matlab Hilfe auch bei vermeintlich bekannten Strukturen.
- Ein-/Ausgabe-Syntax einer Funktion:
  function [out1 out2 etc.] = nameFunction(in1, in2, etc.)
- Die Initialisierung des Sensors und des EV3 Objektes ist bereits an anderer Stelle implementiert und muss **nicht** mehr erfolgen.
- Zugriff auf ein Element an der Stelle x des Vektors y durch: `y(x)`.
- Beachten Sie die Funktionsweise des zeitdiskreten Integrators: Die Schaltzustände "Taster gedrückt" und "Taster nicht gedrückt" können als Werte +1 bzw. -1 interpretiert werden. Wird zeitlich darüber integriert, erhält man einen linearen Anstieg oder Abfall.
- If-Anweisungen könnten nützlich sein.

b) Der Verlauf des Sinussignals (Sollwert) und des roten Punktes (Istwert) sowie der Sensorstatus sollen mit der MATLAB Funktion `plot` visualisiert werden. Schauen Sie sich zunächst in der Hilfe die Verwendung des Befehls an. Öffnen Sie dann die Datei *touchPlot.m* und arbeiten Sie die unten gegebenen Punkte der Reihe nach ab. Nutzen Sie dazu intensiv die MATLAB Hilfe. Die Vektoren der Signale (*nominalval_vector*, *actualval_vector*, *switchstate_vector*) werden der Funktion übergeben. Durch Betätigung von *Ausgabe* kann die Funktion aus der GUI heraus aufgerufen werden, wobei die offenen Fenster der ausgegebenen Graphen bei erneuter Betätigung von *Ausgabe* automatisch geschlossen werden. Der Aufruf der Funktion direkt aus Matlab führt zu einem Fehler, da die Übergabeparameter nur in der GUI bekannt sind (private Variablen).

1) Geben Sie das Sollsignal und das Istsignal in **einem** Graphen in verschiedenen Farben aus.
2) Beschriften Sie die Achsen mit *X-Werte* und *Y-Werte* in der Schriftgröße 12.
3) Geben Sie dem Graphen die Überschrift *Soll- und Istwert* mit der Schriftgröße 15.
4) Ändern Sie die Fenstergröße auf halbe Bildschirmgröße und platzieren Sie es genau mittig auf dem Bildschirm. (Diese Änderungen sind in der Funktion zu implementieren)
5) Fügen Sie Hilfslinien in Form eines Gitters in den Graphen ein.
6) Fügen sie eine Legende in der unteren rechten Ecke mit der Benennung der Signale ein.
7) Stellen Sie in einem weiteren Graphen den Zustand des Sensors dar.
8) Geben Sie dem Graphen die Überschrift *Schalterstatus*.
9) Passen Sie die Skalierung der Y-Achse auf -0.5 bis 1.5 und die Skalierung der X-Achse genau passend an.
10) Berechnen Sie die Anzahl des Zustandes *gedrückt* in dem Vektor *switch-state_vector* und geben Sie ihn unten links in dem Graphen aus. (Form: True-State Count: XX)

(ca. 110min)

Tipps:

- Der Befehl `get` wird nur in Unterpunkt 4) verwendet, während der Befehl `set` nicht benötigt wird.
- Suchen Sie in der Hilfe nach: `title, ylabel, xlabel, ScreenSize, grid, legend, axis, find, length, text, num2str`
- Verknüpfung mehrerer Strings in Matlab: `['string1','string2',...]`
- Die Beispiele in der Hilfe sind in der Regel sehr nützlich.