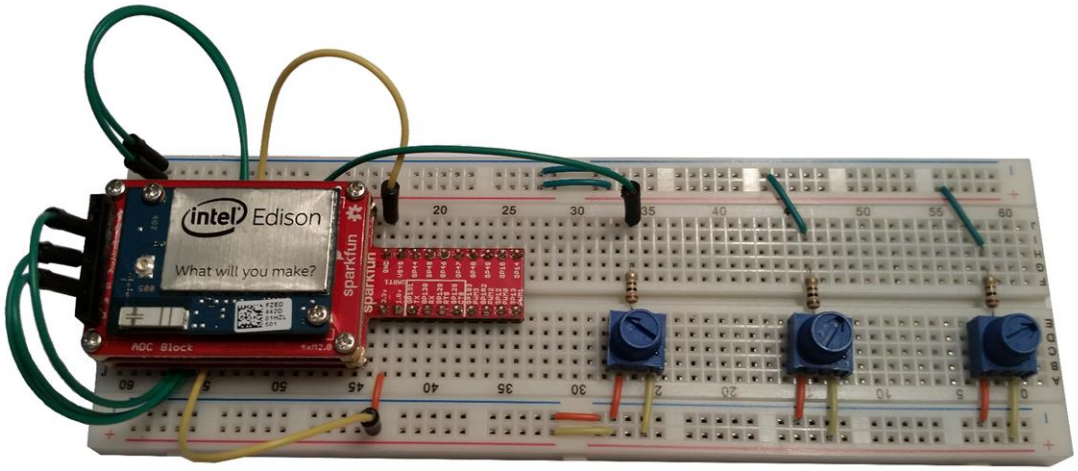
Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Edison Prothese

Gruppennummer 13



Informationstechnik Labor Sommersemester 2016

Prof. J. Walter

Gruppenmitglieder:

Lucas Ertl *43974*

Alexander Schemel *43139*

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

Problemstellung

Im Rahmen des Hybride-Integration Labors dieses Semesters wird eine Auswerteelektronik für einen kapazitiven Winkelsensor erstellt, die später in einer Handprothese zum Einsatz kommen soll. Beim Ausgangssignal der Schaltung handelt es sich um ein analoges Spannungssignal im Bereich zwischen 0 und 5V.

Zur einfachen Messung der Winkelstellung in der Praxis benötigt es eine Möglichkeit zur Umrechnung des Spannungssignals in den zugehörigen Winkel sowie zur Anzeige des aktuellen Winkels.

Stand der Technik

Als µ-Controller Plattform würde sich besonders der Intel Edison eignen, da hier im Vergleich zu den meisten anderen Alternativen bereits ein integriertes Bluetooth-Modul vorhanden ist. Ein Breakout-Board mit einem Analog-Digital-Umsetzer stellt die notwendigen analogen Input-Pins zu Verfügung.

Zusätzlich kann die Entwicklungsumgebung mehr oder weniger frei gewählt werden. Wir haben uns hier für die Intel XDK IoT Edition entschieden, da diese bereits Templates zur einfachen Nutzung des Bluetooth-Moduls beinhaltet und sowohl Programme zum Einsatz auf dem Edison, als auch Apps zum Einsatz auf allen gängigen Smartphone-Betriebssystemen erstellt werden können. Alternativ könnten aber auch die Arduino IDE, Eclipse oder einige andere IDEs zur Realisierung des Projektes verwendet werden.

Aufgabenstellung

Für die Umrechnung und Darstellung der Sensorsignale als die der Spannung entsprechenden Winkelwerte soll das analoge Signal in einem µ-Controller umgerechnet und über Bluetooth an ein Smartphone übertragen werden.

Da die Auswerteelektronik erst noch gefertigt werden muss, wird der ausgegebene Spannungswert anfangs durch ein mittels Drehpotentiometer simuliertes Signal ersetzt. Dieses simulierte Sensorsignal soll über den analogen Eingang eines µ-Controller eingelesen werden, dort zu einem Winkel umgerechnet und anschließend über das integrierte Bluetooth-Modul einer Smartphone-App zur Verfügung gestellt werden.

In der Smartphone-App sollen neben den Winkelwerten auch die Spannungswerte als Zahlen angezeigt werden. Zusätzlich soll eine optische Darstellung des aktuell ausgelesenen Winkels implementiert werden.

Anschließend könnte die Anordnung um 2 weitere Potentiometer zur Simulation von 2 weiteren Winkelsensoren erweitert werden, damit ein vollständiger Finger mit 3 Gelenken und deren Winkelstellungen in der App angezeigt werden kann. Nach erfolgreichem Test mit den Drehpotentiometern und erfolgreicher Fertigung der Auswerteelektronik im Hybride Integration Labor kann die Umrechnung der am ADC anliegenden Spannungen in Winkel im Programmcode an die echten Werte des Sensors angepasst werden.

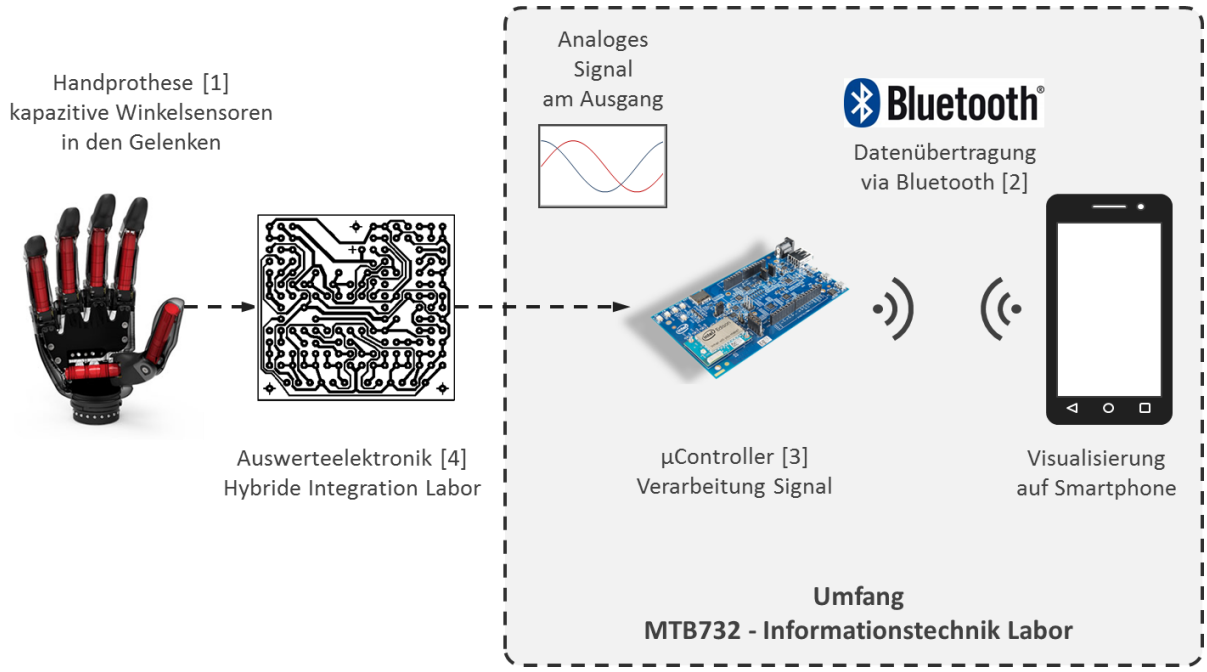


Abbildung 1: Grundidee bzw. Prinzipskizze

MindMap

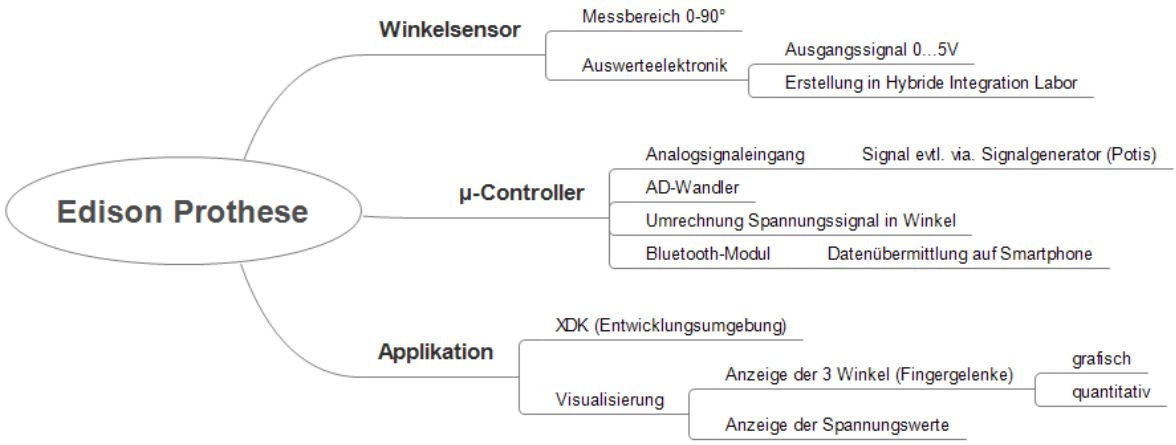


Abbildung 2: Brainstorming im Zuge der Aufgabendefinition

Anforderungsliste

Hochschule Karlsruhe Technik und Wirschaft MTB732 - Informationstechnik Labor		**Anforderungsliste** Edison Prothese	Prof. Dipl.-Ing. Jürgen Walter			
Organisation	**Prozess**	**Anforderungen**	**Wert - Daten**			
Nr.	Art		Mindest-Erfüllung	Soll-Erfüllung	Ideal-Erfüllung	Einheit
Physikalisch-Technische Funktion						
F01	J/N	kabellose Signalübertragung zw. µC und Smartphone mit Android	-	-	-	-
F02	F	Einlesen von Sensorwerten in Form analoger Spannungen, 0-5V	1	1	3	Signale
F03	W	Einlesen von Sensorwerten in Form frequenzmodulierter Signale Frequenzbereich von 10 bis 100kHz	-	-	-	-
Technologie						
T01	J/N	µController: Intel-Edison / XDK	-	-	-	-
T02	J/N	Blueooth Low-Energy, Edison, Android	-	-	-	-
T03	F	Bluetooth Version	?	4.0	4.1	-
Mensch-Produkt-Beziehung						
M01	F	Simulation von Sensorwerten mittels manueller Betätigung (Poti's)	1	1	3	St.
M02	J/N	Anzeige der Spannung der Eingangssignale auf Smartphone-App	-	-	-	-
M03	W	Anzeige der Frequenz der Eingangssignale auf Smartphone-App	-	-	-	-
M04	J/N	Animation der Fingerglieder auf Smartphone-App	-	-	-	-
Anforderungsarten: J/N - Ja/Nein; F - Forderung; W - Wunsch; **Konstruktionsphase**: P - Prinzip; K - Konzept; E - Entwurf; A - Ausarbeitung						
Ausgabe: **29.04.2016** Version: **2** Blatt: **1** von **1**		**Bemerkungen:**	***22.04.2016, J.Walter*** Datum, Unterschrift			

Blackbox

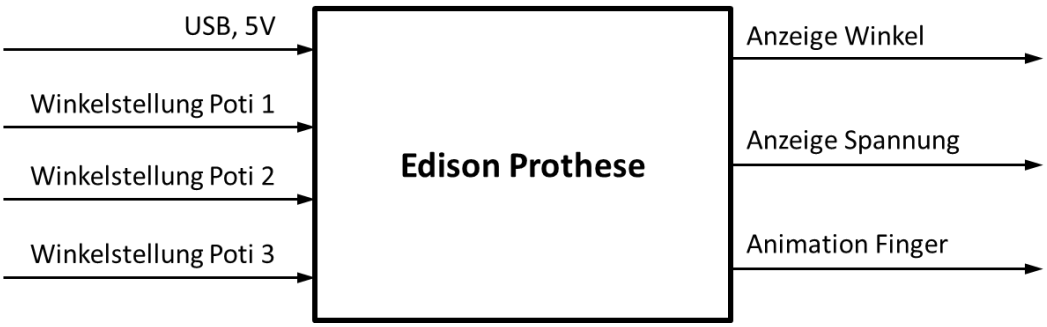


Abbildung 3: Blackbox des Systems

Zeitplan

Nr.	Zeitplan: MTB732 - Informationstechnik Labor Lucas Ertl, Alexander Schemel	Bearbeitungs-Status 0-9	Jahr	2016									
			Monat	März				April				Mai	
			Kalenderwoche	10	11	12	13	14	15	16	17	18	19
	Edison Prothese												
1	Problemstellung, Stand der Technik, Aufgabenstellung; BlackBox, AFL, Funktionsstruktur	1											
2	Recherche und Einarbeitung, Edison und XDK	1											
3	Einarbeitung Java Script	1											
4	Programmierung Signalumwandlung (Edison)	1											
5	Erstellung Anzeigeapplikation (XDK)	1											
6	Test	1											
7	Erstellung Dokumentation und Präsentation	1											
8	**Termin: Präsenation**	1										06.05	
9	**Termin: Abgabe**	1										06.05	

Abbildung 4: Zeitplanung

Stückliste

Tabelle 1: Stückliste

Bauteilbezeichnung	**Erläuterung**
Intel® Edison	Single-Board-Computer
ADC Breakout-Board	Sparkfun; ADS1015 ADC von TI 12-Bit Delta-Sigma Convertor mit Analog Multiplexer, 4 Channel
GPIO Breakout-Board	Sparkfun GPIO Block
Base Breakout-Board	Sparkfun Base Block
Potentiometer	3x10k
Widerstand	3x100Ohm
LM331N	Spannung-Frequenz-Wandler PMIC - U/F-Wandler Texas Instruments LM331N/NOPB Spannung zu Frequenz 100 kHz PDIP-8

Intel® Edison

Der Intel® Edison ist ein Single-Board-Computer, der durch seine sehr kleine Bauweise, sowie seine Leistungsfähigkeit und Langlebigkeit ideal für Projekte geeignet ist die viel Rechenleistung benötigen, aber mit wenig Energiebedarf und geringem Bauraum auskommen müssen.

Der µComputer ist mit einem High-Speed-Prozessor und integrierten Arbeitsspeicher-, Speicher-, WiFi und Bluetooth® 4.0 Low Energy (BLE) Bausteinen ausgestattet. BLE ermöglicht, im Vergleich zum „klassischen“ Bluetooth, eine Funkübertragung mit geringerem Stromverbrauch bei einer Reichweite von ca. 10m. Über den 70-Pin-Anschluss kann der Edison mit zusätzlichen, je nach Hersteller teils auch miteinander kombinierbaren Breakout-Boards erweitert werden.

Intel® XDK IoT Edition oder Eclipse können als Open-Source-Software-Entwicklungsumgebungen genutzt werden. Programmiersprachen die unterstützt werden sind C++, JavaScript, Python und HTML5. Wir haben unter Intel® XDK mit JavaScript und HTML5 gearbeitet.

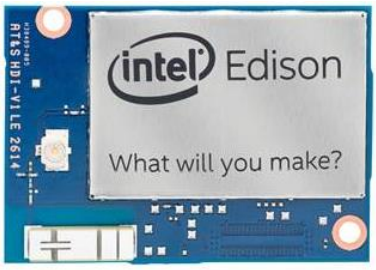


Abbildung 5: Intel® Edison [10]

Die technischen Daten auf einen Blick

Abmessungen:	35.5 × 25.0 × 3.9 mm
Arbeitstemperatur:	0 … 40°C
Prozessor:	22 nm Intel SoC – bestehend aus einem Intel Atom Dual-Core CPU mit 500 MHz und einem Intel Quark Microcontroller mit 32-Bit und 100 MHz
Arbeitsspeicher:	1 GB LPDDR3
Flashspeicher:	4 GB eMMC
WiFi:	Broadcom 43340 802.11 a/b/g/n, Dual-band (2.4 and 5 GHz)
Bluetooth:	Bluetooth 4.0 (LE)
Firmware / Software:	CPU OS: Yocto Linux v1.6 Microcontroller OS: RTOS

SparkFun-Module für den Intel® Edison

Base Block

Schnittstellen auf dem Base-Block ermöglichen den Anschluss externer Peripheriegeräte an den Intel® Edison. Der Base-Block verfügt über einen Micro-USB-AB Anschluss der USB-OTG (On-the-go) unterstützt. Dieser ermöglicht den Zugriff auf den Edison OTG Port, der angeschlossene OTG-Geräte bzw. den Edison mit Energie versorgen kann. Der Micro-USB-B-Port (Console) versorgt den Edison und andere evtl. verwendete Blöcke mit einer Spannung von 4±0.1V und ermöglicht den Zugriff auf den Edison Stack. Über den Power-Button kann der Edison in einen Sleep-Zustand versetzt oder komplett ausgeschaltet werden. Sobald an einem der Sparkfun-Module Energie anliegt leuchtet die Power-LED. Die Daten-LED's signalisieren, dass die Console gerade aktiv ist. Über die LED-Jumper können die LEDs zu- oder abgeschaltet werden um Energie zu sparen. Der 70-Pin- Anschluss (Expansion Header) übermittelt die Signale und ist die zentrale Schnittstelle zwischen dem Edison und den Sparkfun-Modulen. Um den Base Block zu nutzen, muss dieser auf der Rückseite des Edison oder auf einen vorhandenen Sparkfun-Stack aufgesteckt werden.

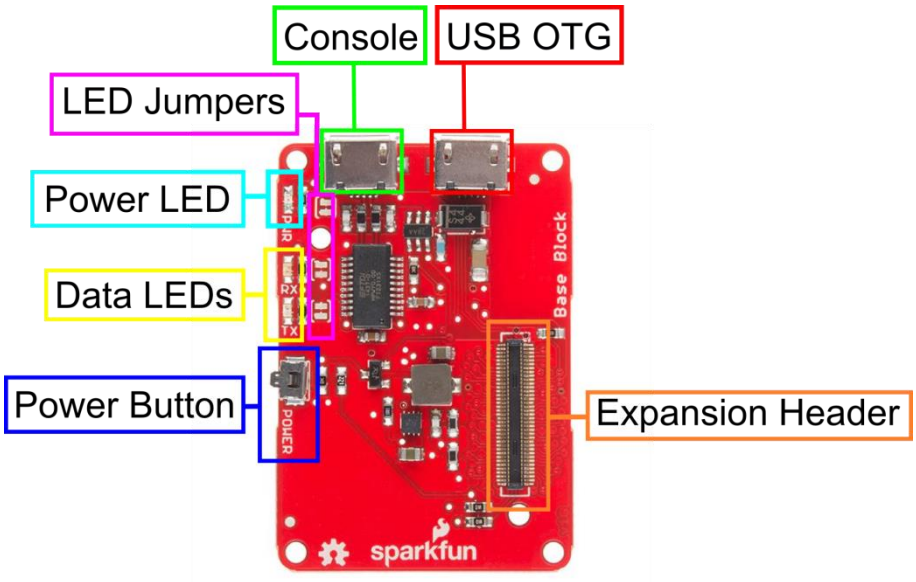


Abbildung 6: Base-Block [11]

GPIO Block

Der GPIO Block ist ein einfaches Breakout-Board, dass den Zugriff auf alle grundlegenden GPIO, PWM und UART2 Pins ermöglicht. Des Weiteren stellt der Block die auf dem Edison zur Verfügung stehenden Spannungspotentiale von 3.3V, 1.8V, VSYS und GND auf dem Bread-Board zur Verfügung. Über den Jumper Level-Select wird die Referenzspannung auf 3.3V oder VSYS eingestellt. Die Power Pins bieten Zugriff auf die Power Pins des Edison, bestehend aus GND (Pin für Erdung), VSYS mit 4.0-4.1V, einem Pin mit 1.8V und einem Pin mit 3.3V die von der internen Spannungsversorgung des Edison zur Verfügung gestellt werden. Die GPIO UART1 Pins können wie der Name schon sagt als UART (Universal Asynchronous Receiver Transmitter - elektronische Schaltung zur Realisierung digitaler serieller Schnittstellen) genutzt werden und die GPIO PWM Pins zur Erzeugung einer Pulsweitenmodulation. Die General GPIO Pins stehen zur freien Verwendung zur Verfügung. Der 70-Pin- Anschluss (Expansion Header) bildet wieder die zentrale Schnittstelle zwischen dem Edison und den Sparkfun-Modulen.

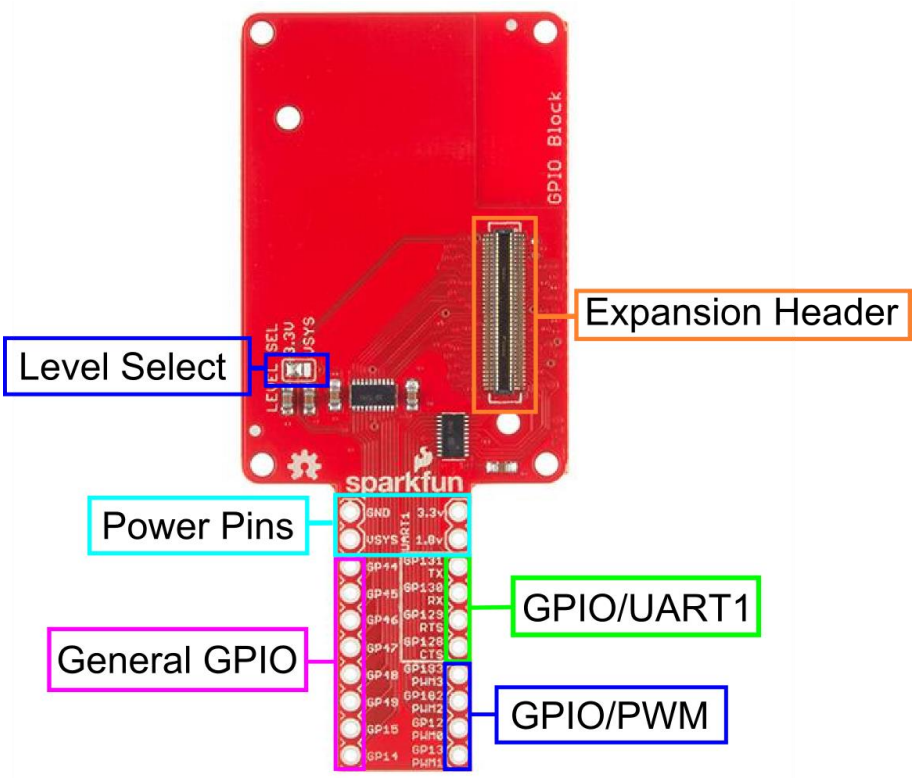


Abbildung 7: GPIO Block [12]

ADC Block

Der ADC Block stellt dem I2C-Bus des Edison einen AD-Wandler zur Verfügung. Jumper ermöglichen die Auswahl der I2C Slave Adresse, sodass vier Blöcke unter einem Edison auf einem Stack verwendet werden können. Die Abtastrate beträgt 2.2kHz. Es stehen vier Signaleingänge und eine Referenzspannung von 3.3V (150mA) zur Verfügung. Die Referenzspannung wird on-Board erzeugt und kann bspw. Potentiometer versorgen. 3.3V sollten an diesen Pins in keinem Fall überschritten werden.

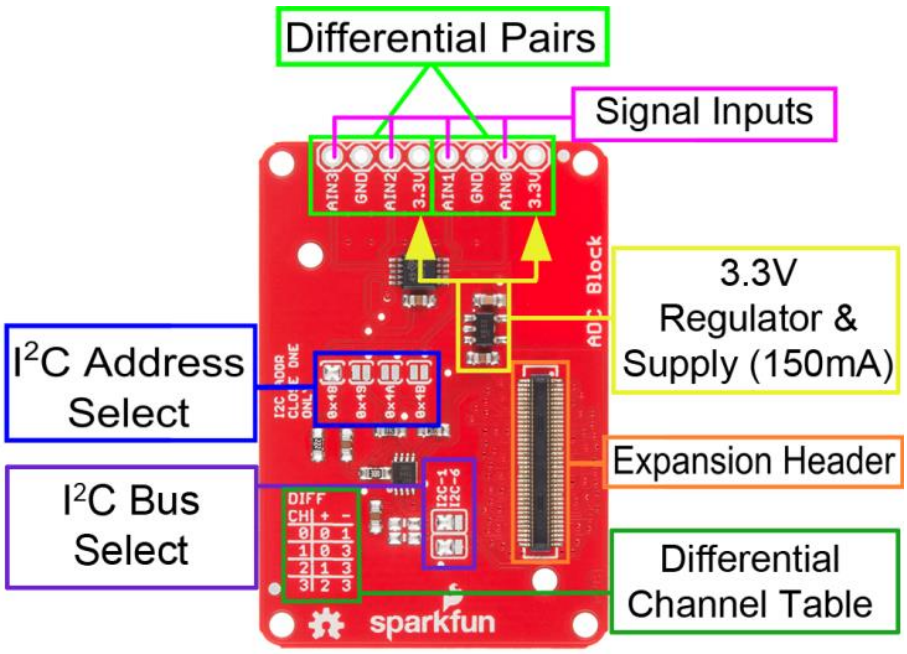


Abbildung 8: ADC Block [13]

System

Bevor die Bluetooth-Verbindung des Intel Edison genutzt werden kann, muss diese aktiviert werden. Wird der Edison von der Stromversorgung getrennt, muss dieser Schritt erneut durchgeführt werden. Folgende 3 Zeilen müssen in die Konsole des Edison (Serial oder SSH) eingegeben werden.

rfkill unblock bluetooth

killall bluetoothd

hciconfig hci0 up

In der main.js auf dem Edison wird nach Start der Anwendung sofort das Advertising der Bluetooth-Verbindung gestartet, andere BLE-kompatible Geräte können den Edison jetzt unter dem in den Voreinstellungen der main.js definierten Namen sehen und eine Verbindung mit ihm herstellen. Hierzu wird mit den Funktionen des bleno-Moduls zuerst die BLE Characteristic definiert und anschließend mehrere bleno Event-Handler registriert. Diese führen verschiedene Funktionen aus, wenn bleno bezogene Events eintreffen. Startet das Advertising nicht bei Start des Programms automatisch, müssen wahrscheinlich die 3 oben angegebenen Zeilen erneut in die Konsole eingegeben werden.

Im Zweiten Teil wird die read.ADC-Funktion deklariert. Mit dieser kann dann durch Übergabe eines der 4-Channel als Argument ein Spannungswert am ADC eingelesen werden.

Im Hauptteil des Codes wird eine Endlosschleife ausgeführt, die über die setTimeout()-Direktive am Ende in 50ms Abständen eine Funktion ausführt. Die Funktion weist die an dem ADC anliegenden Spannungen, einer Zwischenspeichervariablen zu. Die Spannungswerte werden nach der dritten Nachkommastelle abgeschnitten und für eine einfachere Übertragung mit 1000 multipliziert.

```
function periodicActivity()
{
        var bt_send_value=((adc.readADC(0)).toFixed(3))*1000;
```

Anschließend wird die Ganzzahl in 2 Bytes zerlegt. Zuerst wird die Zahl durch 255 geteilt und das Ergebnis in das erste Byte gespeichert. Dann wird mittels Modulo der Rest bestimmt und in das zweite Byte gespeichert.

```
        var fak=Math.floor((bt_send_value)/255);
        var rst=Math.round((bt_send_value)%255);
        var buf = new Buffer([fak,rst]);
```

Im BLE-Central-Gerät können die Bytes dann wieder einfach zusammengerechnet werden um den ursprünglichen float-Wert wieder zu erhalten. Der Vollständige Buffer für alle 3 Fingerglieder enthält

demnach 6 Bytes und wird analog zum Code-Snippet welches die Werte für 1 Fingerglied versendet, implementiert. Das Versenden findet statt, indem der Buffer an die sendNotification-Methode unserer Controller-Characteristic übergeben wird.

```
        controllerCharacteristic.sendNotification(buf);
        setTimeout(periodicActivity, 50);
}
periodicActivity();
```

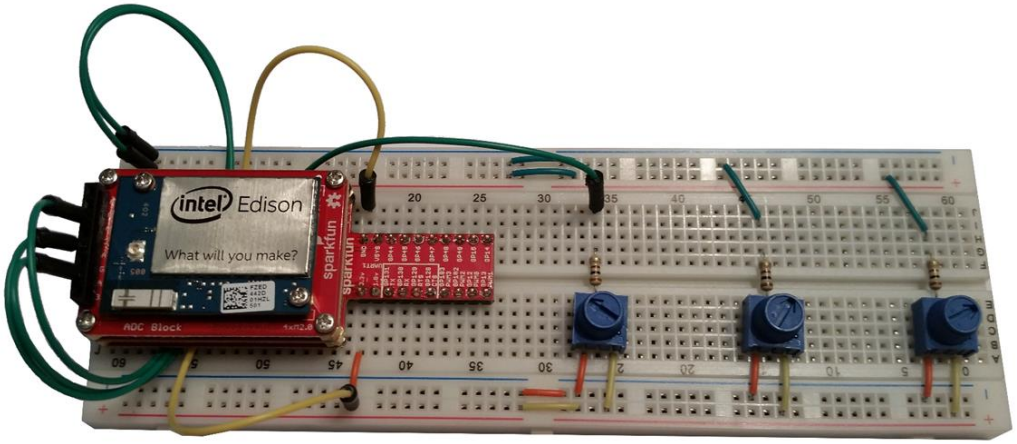


Abbildung 9: Aufbau - Edison mit Sparkfun Blocks und Potis auf Bread-Board

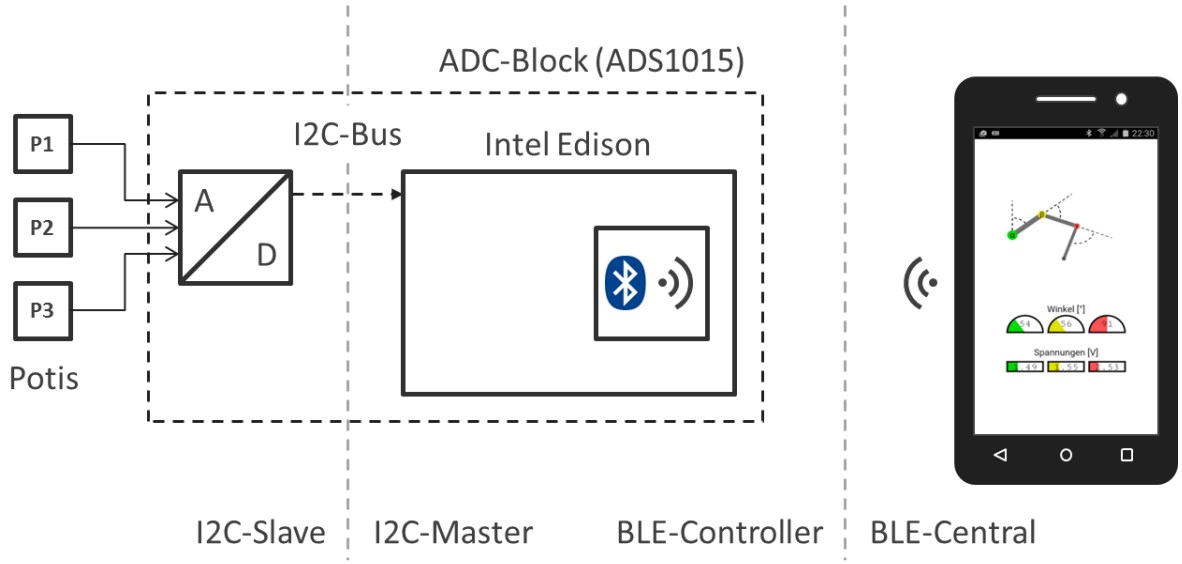


Abbildung 10: Blockschaltbild des Systems

Smartphone Applikation für Android

Bevor mit der Programmierung der App begonnen werden kann, muss zuerst im Project-Tab von XDK unter Plugin-Management das Third-Party-Plugin „cordova-plugin-ble-central“ eingebunden werden. Dieses ermöglicht auf dem Smartphone die Nutzung des Bluetooth-Moduls als Bluetooth-Low-Energy Central-Device. Das Gegenstück zum BLE-Peripheral Device, welches mit Hilfe des bleno-Moduls in der main.js auf den Edison erstellt wurde.

In der index.html der Smartphone-App wird das Layout der Oberfläche angelegt. In unserem Fall besteht das Layout aus Header, Überschrift, einem Connect-Button zum Herstellen der BLE-Verbindung mit dem Edison, sowie einer zentrierten Canvas-Zeichenfläche zur Anzeige der verschiedenen Animationen. Zusätzlich werden hier die verschiedenen benötigten JavaScript-files eingebunden. Neben dem Main-Code in der Datei app.js, gehört in unserem Fall dazu noch die jQuery-Bibliothek als Schnittstelle zwischen html-Elementen in index.html und JavaScript-Events in app.js, sowie die Cordova-Bibliothek zur App-Programmierung für verschiedene mobile Endgeräte und mehrere Betriebssysteme. In unserem Fall wurde die App jedoch nur auf verschiedenen Android-Geräten getestet.

In der app.js der Smartphone-App befindet sich der mit den Elementen in der index.html verknüpfte JavaScript-Code. Hier wird zuerst mit Hilfe der Plugin-Funktionen eine BLE-Verbindung zum Edison hergestellt, sobald der Connect-Button gedrückt wurde. Die Verbindung funktioniert „onNotify“ – jedes Mal wenn eine BLE-Notification vom Edison auf dem Smartphone eintrifft, wird eine Callback-Funktion aufgerufen. In dieser Funktion werden zuerst die Ganzzahlen, die als Byte-Stream empfangen wurden in einer Variablen zwischengespeichert.

```
var dir = new Uint8Array(data);
```

Anschließend wird der Byte-Stream wieder in die ursprünglichen Spannungswerte umgerechnet das erste Byte mit 255 multipliziert und das zweite Byte addiert wird.

```
var value=(dir[0]*255)+dir[1];
```

Dann wird die Ganzzahl mit 1000 dividiert, wieder als Gleitkommazahl gespeichert und der Funktion updatefingerPos() übergeben.

```
var floatvalue=value/1000;
window.app.GUI.updatefingerPos (floatvalue);}
```

Für die nachfolgenden Bytes der beiden anderen Fingerglieder wird analog vorgegangen

Das nachfolgende Code-Snippet zeigt die Funktion updatefingerPos, die die Eigenschaften des Objekts Finger für die jeweiligen Fingerglieder aktualisiert. Hier werden auch die Werte für den (Anzeige-)Winkel und den Winkel in Grad aus dem übergebenen Spannungswert berechnet. Auch hier wird für die andern beiden Fingerglieder analog vorgegangen.

```
GUI.prototype.updatefingerPos = function(current_voltage)
{
        this._finger.voltage =current_voltage;
        this._finger.angle_deg =current_voltage/5*180;
        this._finger.angle = 1.5*Math.PI + (current_voltage*Math.PI/5);

};
```

Alle 50 ms zeichnet eine draw-Funktion die aktuellen Werte der Objekteigenschaften, welche die Winkelstellungen der 3 Fingerglieder enthalten, in die Canvas-Zeichenfläche auf dem Display. Auf die Canvas-Funktionen wird im Folgenden noch genauer eingegangen.

jQuery

JQuery ist eine weit verbreitete JavaScript-Bibliothek bzw. ein Framework. Es stellt dem Programmierer Funktionen zur Manipulation und Navigation des DOM's (Document Object Model=Abbild eines HTML-Dokuments im Arbeitsspeicher) zur Verfügung, was u.a. den Umgang mit Events und Animationen mit HTML vereinfacht. Außerdem kann mit JQuery der JavaScript Quellcode erheblich reduziert und damit Zeit eingespart werden. So können bspw. Animationen erstellt, oder Elemente manipuliert werden – sprich Position, Farbe, Größe etc.

Cordova BLE Central Plugin

Das Cordova BLE-Plugin ermöglicht die Kommunikation zwischen einem Smartphone und dem Bluetooht Low Energy (BLE) Peripheriegerät, in unserem Fall der Bluetooth-Baustein auf dem Edison. Es bietet eine JavaScript Programmierschnittstelle für Anwendungen (API - Application Programming Interface) u.a. für Android. Mit Hilfe des Plugins kann eine Verbindung zwischen dem Edison und dem Smartphone hergestellt werden, sodass Daten gelesen und geschrieben werden können.

HTML5 Canvas

HTML5 Canvas ist – ähnlich wie OpenGL – eine Art Bibliothek/Container bestehend aus Elementen mit Eigenschaften und Methoden. Ein „canvas"-Element ist ein rechteckiger Bereich bestehend aus in Höhen- und Breitenkoordinaten mit der Einheit Pixel. Der Ursprung liegt in der oberen linken Ecke und hat die Koordinaten (0,0). Mit Script-Sprachen wie bspw. JavaScript können dynamische Grafiken innerhalb des Bereichs erzeugt werden. Canvas beinhaltet verschiedene Methoden (drawImage, scale, rotate, translate) und Attribute (lineWidth, fillStyle), um Grafiken wie Pfade, Boxen, Kreise, Text und Bilder zu zeichnen und die gezeichneten Objekte zu skalieren, verschieben und rotieren.

Nachfolgend sind einige Befehle und Auszüge aus dem Quellcode aufgeführt und erläutert:

```
<div style="margin:auto; width:260px; height:540px;">
    <canvas id="finger_canvas" width="240" height="520"
            style="background:#ffffff;
                   margin-left:auto;
                   margin-right:auto;
                   display:block;">
    </canvas>
</div>
```

In diesen Code-Zeilen wird eine neue Zeichenfläche (canvas) mit den Maßen 260x540 Pixel erstellt. Die Hintergrundfarbe soll weiß sein und der Seitenabstand zum Displayrand erfolgt automatisch (zentriert).

```
canvas.getContext("2d").lineTo(x,y);
```

Erzeugt eine Linie zu den angegeben Koordinaten von x,y.

```
canvas.getContext("2d").moveTo(x,y);
```

Verschiebt die aktuelle Arbeitsposition nach x,y.

```
canvas.getContext("2d").arc(x,y,10,0,2*Math.PI);
```

Mit dieser Direktive lässt sich eine Kreisform zeichnen. Die Argumente sind die x-, y-Position des Mittelpunkts, der Radius, der Anfangswinkel und der Endwinkel.

```
canvas.getContext("2d").translate(x,y);
```

Mittels translate wird das Koordinatensystem der Zeichenfläche verschoben.

```
canvas.getContext("2d").ctx.save();
```

Speichert den Zustand und die Attribute des aktuellen Zeichenfeldes.

```
canvas.getContext("2d").restore();
```

Stellt den zuvor gespeicherten Zustand und die Attribute wieder her.

Grafische Oberfläche und Bedienung

Die Android-Applikation wurde mit HTML5 und JavaScript unter Intel® XDK erstellt. Die Verbindung zwischen Smartphone-App und Edison wird über den Connect-Button hergestellt. Die App animiert die drei Fingerglieder bzw. Fingergelenke über drei farblich kenntlich gemachte Punkte die mit Geraden verbunden sind. Die Winkel α, β und γ werden bei Auslenkung gestrichelt dargestellt. Am unteren Bildrand werden die Zahlenwerte der jeweiligen Winkel in Grad in drei Halbkreisen ausgegeben und die Winkel in der entsprechenden Farbe dargestellt. Direkt darunter sind die Spannungen in Volt aufgeführt, ebenfalls in der passenden Farbe.

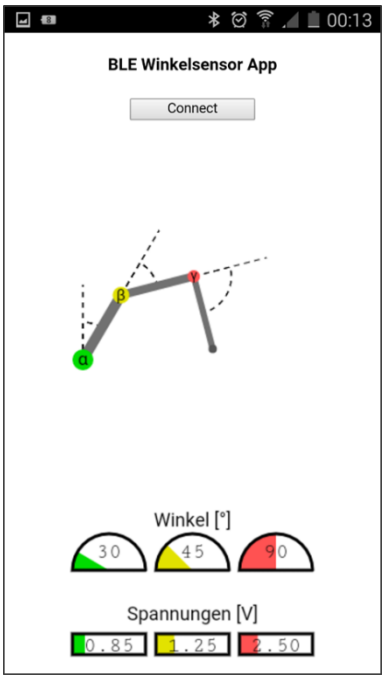


Abbildung 11: aktuelle Oberfläche der App

Fazit

- Error-Handling wünschenswert
- Erweiterung zur Verarbeitung frequenzmodulierter Signale mit LM331 Baustein

  ➔ Bisher noch nicht getestet
- Messungen mit fertiger Schaltung aus hybride Labor durchführen

Quellen

[01] http://www.intel.com/content/www/us/en/do-it-yourself/edison.html
Abgerufen am 2016-04-30; Intel® Edison - Product Page

[02] http://www.intel.com/content/dam/support/us/en/documents/edison/sb/edison_pb_331179002.pdf
Abgerufen am 2016-04-30; Intel® Edison - Data-Sheet

[03] https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-for-edison-experiment-guide
Abgerufen am 2016-04-30; SparkFun - Inventor's Kit for Edison Experiment Guide

[04] https://www.sparkfun.com/products/13024
Abgerufen am 2016-04-30; SparkFun - Intel® Edison

[05] https://www.sparkfun.com/products/13045
Abgerufen am 2016-04-30; SparkFun - Block for Intel® Edison - Base

[06] https://www.sparkfun.com/products/13038
Abgerufen am 2016-04-30; SparkFun - Block for Intel® Edison - GPIO

[07] https://www.sparkfun.com/products/13327
Abgerufen am 2016-04-30; SparkFun - Block for Intel® Edison - ADC

[08] https://learn.sparkfun.com/tutorials/general-guide-to-sparkfun-blocks-for-intel-edison
Abgerufen am 2016-04-30; General Guide to SparkFun Blocks for Intel® Edison

[09] http://www.elektronik-kompendium.de/sites/kom/1805171.htm
Abgerufen am 2016-04-30; Bluetooth Low Energy / Bluetooth Smart (4.0/4.1/4.2)

[10] http://www.intel.com/buy/us/en/product/emergingtechnologies/intel-edison-compute-module-iot-463633
Abgerufen am 2016-05-01; Abbildung Intel® Edison

[11] https://cdn.sparkfun.com/assets/learn_tutorials/2/8/0/BaseAnnotated.png
Abgerufen am 2016-05-01; Abbildung Sparkfun Base Block

[12] https://cdn.sparkfun.com/assets/learn_tutorials/2/8/6/GPIOBlockAnnotated.png
Abgerufen am 2016-05-01; Abbildung Sparkfun GPIO Block

[13] https://cdn.sparkfun.com/assets/learn_tutorials/4/4/0/Edison_ADC_New2.png
Abgerufen am 2016-05-01; Abbildung Sparkfun ADC Block

[14] http://www.w3schools.com/tags/ref_canvas.asp
Abgerufen am 2016-05-03; HTML5 Canvas Reference

[15] https://html.spec.whatwg.org/multipage/scripting.html#the-canvas-element
Abgerufen am 2016-05-03; HTML5 Canvas Element auf HTML-Standard

[16] https://github.com/sandeepmistry/bleno
Abgerufen am 2016-05-04; Bleno-Modul

[17] http://iotdk.intel.com/docs/master/mraa/
Abgerufen am 2016-05-04; Mraa-Modul

[18] http://api.jquery.com/
Abgerufen am 2016-05-04; jQuery Bibliothek

[19] https://github.com/don/cordova-plugin-ble-central
Abgerufen am 2016-05-04; Cordova BLE Central Plugin

Anhang

Code

Edison-Code

Package.JSON

```
{
  "name": "blankapp",
  "description": "",
  "version": "0.0.0",
  "main": "main.js",
  "engines": {
    "node": ">=0.10.0"
  },
  "dependencies": {
      "async": "1.5.0",
      "bleno": "0.3.3"
  }
}
```

Main.JS

```
//-----------------
//Voreinstellungen
//-----------------

//Einbindung der JavaScript Hint-Funktionalitäten
/*jslint node:true, vars:true, bitwise:true, unparam:true */
/*jshint unused:true */

//Variablenzuweisung der in package.json eingebundenen Module
//MRAA zur Nutzung des I2C-Bus für die Kommunikation mit dem Sparkfun-ADC-Block
var mraa = require('mraa');
//BLENO zur Nutzung der Bluetooth Low Energy Funktionsbibliothek
var bleno = require('bleno');

//Variablenzuweisung des ADC-Blocks als I2C-Slave 1
var adc = new mraa.I2c(1);
adc.address(0x48);


//--------------------------------
//bleno Controller Characteristic
//--------------------------------

//Deklaration der globalen Variable für die Controller Characterist
var controllerCharacteristic;

//Deklaration einer Variable für BLE-Gerätename, ID, Service und Characteristic des Intel Edison
var edison = {
    name: "jarvis",
    deviceId: null,
    service: "12ab",
    characteristic: "34cd"
};

//Definition der Controller Characteristic, bei der sich BLE-Central-Geräte registrieren können
controllerCharacteristic = new bleno.Characteristic(
    {
    value: null,
    uuid: edison.characteristic,
    properties: ['notify'],

    //Eventhandler für erfolgreiche Registrierung
    onSubscribe: function(maxValueSize, updateValueCallback)
    {
        console.log("Geraet registriert");
        //Aktualisierung des zu Versenden Wertes
        this._updateValueCallback = updateValueCallback;
    },

    //Eventhandler für Aufhebung einer Registrierung
    onUnsubscribe: function()
    {
        console.log("Registrierung aufgehoben");
        this._updateValueCallback = null;
    },
});
```

```
//Deklaration der Objekteigenschaft updateValueCallback
//als Zwischenspeicher für den als Notification zu versendenden Wert
controllerCharacteristic._updateValueCallback = null;

//Deklaration der Funktion zum Versenden eines Wertes als BLE-Notification
controllerCharacteristic.sendNotification = function(buf) {
    if (this._updateValueCallback !== null) {
        this._updateValueCallback(buf);
    }
};


//--------------------------------------
//Registrierung der bleno Event-Handler
//--------------------------------------

//bleno-Event stateChange - tritt ein wenn Bluetooth-Modul von bleno
//angesprochen werden kann, oder nicht mehr erreichbar ist
bleno.on('stateChange', function(state)
{
    console.log('Status Aenderung: ' + state);

    //Wenn das Modul angesprochen werden kan
    if (state === 'poweredOn')
    {
        //Beginnt der Edison kontinuierlich seinen Namen und Service-Information für Verbindungen zur
Verfügung zu stellen
        bleno.startAdvertising(edison.name,[edison.service]);
    }

    else
    {
        //Kann das Modul nicht (mehr) erreicht werden, wird das oben beschriebene Advertising wieder
beendet
        bleno.stopAdvertising();
    }
});

//bleno-Event accept - tritt bei erfolgreicher Verbindung auf
//Konsolenausgabe bei erfolgreich hergestellter Verbindung mit einem BLE-Central-Device
bleno.on('accept', function(clientAddress)
{
    console.log("Verbunden mit Adresse: " + clientAddress);
});

//bleno-Event disconnect - tritt bei Verbindungsabbruch auf
//Konsolenausgabe bei Verbindungsabbruch
bleno.on('disconnect', function(clientAddress)
{
    console.log("Verbindungsabbruch von Adresse: " + clientAddress);
});

//bleno-Event advertisingStart - tritt auf wenn Advertisting-Vorgang erfolgreich gestartet wurde
//Bei Beginn des Advertisings wird eine neuer Service mit einer Characteristic erstellt
bleno.on('advertisingStart', function(error)
{
    if (error)
    {
        console.log("Advertising Fehlerr:" + error);
    }

    else
    {
        console.log("Advertising gestartet");
        bleno.setServices([

            //Definition eines neuen Primary-Services
            new bleno.PrimaryService(
            {
                uuid: edison.service,
                characteristics: [controllerCharacteristic]
            })
        ]);
    }
});


//-------------------------------------------------------------------------------------------
//Deklaration der readADC-Methode zum Einlesen von Spannungen am Sparkfun ADC-Block (TI ADS1015)
//-------------------------------------------------------------------------------------------

adc.readADC = function(channel)
{

    //Erstellung des Konfigurations-Registers für den ADC
    var config = 0;                 // Bits     Beschreibung
    config |= 1 << 15;              // [15]     Einzelne Umwandlung starten
    config |= 1 << 14;              // [14]     Non-Differential ADC
```

```
    config |= channel << 12;        // [13:12]  Channel wählen
    config |= 1 << 9;               // [11:9]   Messbereich einstellen
    config |= 1 << 8;               // [8]      Power-down, single-shot mode
    config |= 4 << 5;               // [7:5]    1600 Abtastungen pro Sekunde
    config &= ~(1 << 4);            // [4]      Traditional comparator
    config &= ~(1 << 3);            // [3]      Active low comparator polarity
    config &= ~(1 << 2);            // [2]      Non-latching comparator
    config |= 3;                    // [1:0]    Disable comparator

    //Konfigurationsregister auf den ADC schreiben und Einlesevorgang starten
    this.writeWordFlip(0x01, config);

    //Warten bis der Einlesevorgang beendet wurde
    while (!(this.readWordFlip(0x01) & 0x8000)) {}

    //Spannungswert aus dem Umwandlungsregister auslesen und um 4 Bytes shiften
    var voltage = (adc.readWordFlip(0x00) >> 4);

    //Spannungswert auf wahren Wert in 0.002 Volt Auflösung umrechnen
    voltage = 0.002 * voltage;

    //Spannungswert zurückgeben
    return voltage;
};

//Da der ADS1015 das LSB zuerst erwartet, wird die Reihenfolge der Bytes beim Schreibvorgang umgekehrt
adc.writeWordFlip = function(reg, data)
{
    var buf = ((data & 0xff) << 8) | ((data & 0xff00) >> 8);
    return this.writeWordReg(reg, buf);
};

//Da der ADS1015 auch das LSB zuerst ausgibt, wird die Reihenfolge der Bytes auch beim Lesevorgang
umgekehrt
adc.readWordFlip = function(reg)
{
    var buf = adc.readWordReg(reg);
    return ((buf & 0xff) << 8) | ((buf & 0xff00) >> 8);
};


//-------------------------------------------------------------------------------------------------------------
-----
//Endlosschleife in Form einer mittels setTimeout erzeugten, periodisch in 50ms Abständen ausgeführten
Funktion
//-------------------------------------------------------------------------------------------------------------
-----

function periodicActivity()
{

        //Zuweisung der am ADC anliegenden Spannungen an Zwischenspeichervariablen mittels der Methode
readADC
        //Die Spannungswerte werden mit 3 Nachkommastellen abgespeichert und zur anschließenden
vereinfachten Übertragung
        //als Byte-Stream mit 1000 multipliziert um eine Ganzzahl zu erhalten,
        //die alle Stellen der float-Variable mit 3 Nachkommastellen enthält.
        //Umrechnung funktioniert also nur für den hier ausreichenden Wertebereich von 0.000-65.025(!)

        //Einlesen und Umwandlung in Ganzzahl von ADC-Channel 0 - Sensor Fingerglied 1
        var bt_send_value_1=((adc.readADC(0)).toFixed(3))*1000;
        //Einlesen und Umwandlung in Ganzzahl von ADC-Channel 1 - Sensor Fingerglied 2
        var bt_send_value_2=((adc.readADC(1)).toFixed(3))*1000;
        //Einlesen und Umwandlung in Ganzzahl von ADC-Channel 2 - Sensor Fingerglied 3
        var bt_send_value_3=((adc.readADC(2)).toFixed(3))*1000;

        //Anschließend wird die Ganzzahl in 2 Bytes zerlegt, indem sie einmal durch 255 geteilt wird, und
das Ergebnis
        //ins erste Byte gespeichert, und anschließend mit mod der Rest bestimmt und ins zweite Byte
gespeichert wird
        //Im BLE-Central-Gerät können die Bytes dann wieder einfach zusammengerechnet werden um den
ursprünglichen float-Wert
        //zurückzuerhalten

        //Umrechnung des Spannungswerts für Fingerglied 1 und speichern in Zwischenspeichervariablen für
Byte 0 und 1
        var fak_1=Math.floor((bt_send_value_1)/255);
        var rst_1=Math.round((bt_send_value_1)%255);
        //Umrechnung des Spannungswerts für Fingerglied 2 und speichern in Zwischenspeichervariablen für
Byte 2 und 3
        var fak_2=Math.floor((bt_send_value_2)/255);
        var rst_2=Math.round((bt_send_value_2)%255);
        //Umrechnung des Spannungswerts für Fingerglied 3 und speichern in Zwischenspeichervariablen für
Byte 4 und 5
        var fak_3=Math.floor((bt_send_value_3)/255);
        var rst_3=Math.round((bt_send_value_3)%255);

        //Zwischenspeichervariable von Typ Buffer, mit den 6 Bytes die die 3 Spannungswerte enthalten
        var buf = new Buffer([fak_1,rst_1,fak_2,rst_2,fak_3,rst_3]);
```

```
        //Versenden des Buffers als BLE-Notification
        controllerCharacteristic.sendNotification(buf);
        //50ms warten und anschließend die Funktion periodicActivity erneut ausführen
        setTimeout(periodicActivity, 50);
}

//Aufruf der oben definierten Endloschleifenfuktion
periodicActivity();
```

Smartphone-App Code

Index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>BLE Sensor</title>
    <meta http-equiv="Content-type" content="text/html; charset=utf-8">
    <meta name="viewport" content="width=device-width, minimum-scale=1, initial-scale=1, user-scalable=no">
    <style>
        @-ms-viewport { width: 100vw ; min-zoom: 100% ; zoom: 100% ; }  @viewport { width: 100vw ; min-
zoom: 100% zoom: 100% ; }
        @-ms-viewport { user-zoom: fixed ; min-zoom: 100% ; }           @viewport { user-zoom: fixed ; min-
zoom: 100% ; }
    </style>
</head>
<body>

    <!-- Überschrift und Connect-Button -->
    <h4 style="text-align:center;">BLE Winkelsensor App</h4>
    <div id="connection" style="text-align:center;">
        <button id="connect_ble" style="width:35%;">Connect</button>
    </div>

    <!-- Anlegen einer Canvas-Zeichenfläche mit den Maßen 260x540 Pixel -->
    <!-- Hintergrundfarbe: weiß, Seitenabstand zum Displayrand automatisch -->
    <div style="margin:auto; width:260px; height:540px;">
        <canvas id="finger_canvas" width="240" height="520"
                style="background:#ffffff;
                       margin-left:auto;
                       margin-right:auto;
                       display:block;">
        </canvas>
    </div>

    <!-- Laden der benötigten JavaScript files  -->
    <!-- Cordova-Bibliothek -->
    <script type="text/javascript" src="cordova.js"></script>
    <!-- jQuery-Bibliothek  -->
    <script type="text/javascript" src="lib/jquery/jquery-2.2.2.js"></script>
    <!-- app.js-File mit JavaScript-Programmcode  -->
    <script type="text/javascript" src="js/app.js"></script>
</body>
</html>
```

App.js

```
//-----------------
//Voreinstellungen
//-----------------

//JavaScript Code-Hints für XDK einbinden
/*jslint unparam: true */
/*jshint strict: true, -W097, unused:false,  undef:true, devel:true */
/*global window, document, d3, $, io, navigator, setTimeout */
/*global ble*/

//strict mode aktivieren um schlechte Syntax als Fehler zu erkennen (z.B. undeklarierte Variablen)
//Sorgt für zusätzliche Sicherheit von JavaScript-Code
"use strict" ;

//Bluetooth Low Energy Service und Characteristic Informationen für Edison definieren
window.edison =
{
    name: "jarvis",
    deviceId: null,
    service: "12ab",
    characteristic: "34cd"
};
```

```
//--------------------------------------------------------------------------
//GUI Class  - Übergeordnete Klasse für alle Anzeigefunktionen der  App
//--------------------------------------------------------------------------

//GUI Konstruktor
function GUI(canvas)
{

    //Canvas Element der Klasse zuordnen
    this._canvas = canvas;
    //Abkürzung der getContext("2d) Methode als "ctx" festlegen
    this._ctx = this._canvas.getContext("2d");

    //Initialisierung des Thread-Objekts und des Finger-Objekts mit allen Anfangseigenschaften
    this._GUIThread = null;
    this._finger =
    {
        //Ausgangspunkt des ersten Fingerglieds
        x: 15,
        y: 230,
        //Länge der Fingerglieder
        radius: 72,
        visible: false,
        //Anfangswinkel
        startangle: Math.PI,
        //Winkelstellungen der 3 Fingerglieder in Bogenmaß
        angle_1: 0,
        angle_2: 0,
        angle_3: 0,
        //Winkelstellungen der 3 Fingerglieder in Gradmaß
        angle_deg_1: 0,
        angle_deg_2: 0,
        angle_deg_3: 0,
        //Spannugen an den 3 Sensorelementen
        voltage_1: 0,
        voltage_2: 0,
        voltage_3: 0,

    };
}

//Funktion zum Aktualisieren der Eigenschaften des Objekts Finger für das Erste Fingerglied
GUI.prototype.updatefingerPos1 = function(current_voltage)
{
        this._finger.voltage_1 =current_voltage;
        this._finger.angle_deg_1 =current_voltage/5*180;
        this._finger.angle_1 = 1.5*Math.PI + (current_voltage*Math.PI/5);
};

//Funktion zum Aktualisieren der Eigenschaften des Objekts Finger für das Erste Fingerglied
GUI.prototype.updatefingerPos2 = function(current_voltage)
{
        this._finger.voltage_2 =current_voltage;
        this._finger.angle_deg_2 =current_voltage/5*180;
        this._finger.angle_2 = 2*Math.PI + (current_voltage*Math.PI/5);
};

//Funktion zum Aktualisieren der Eigenschaften des Objekts Finger für das Erste Fingerglied
GUI.prototype.updatefingerPos3 = function(current_voltage)
{
        this._finger.voltage_3 =current_voltage;
        this._finger.angle_deg_3 =current_voltage/5*180;
        this._finger.angle_3 = 2*Math.PI + (current_voltage*Math.PI/5);
};

//Funktion zum Zeichnen der Fingeranimation auf dem HTML-Canvas
GUI.prototype.drawfinger = function()
{

//-----------------------------------------------------------------------------
//1. Anzeigelayer (von unten nach oben): Gestrichelte Winkelanzeigen einzeichnen
//-----------------------------------------------------------------------------

    //Zeichencursor auf Ausgangspunkt bewegen - Untererster Punkt des ersten Fingerglieds
    //Leere Ausgangskonfiguration abspeichern, zum späteren Laden des Ausgangszustandes
    this._ctx.save();
    //Neuen Zeichenpfad beginnen
    this._ctx.beginPath();
    //Ausgangspunkt definieren
    x = this._finger.x;
    y = this._finger.y;
    //Cursor zum Ausgangspunkt bewegen (ohne dabei zu zeichnen)
    this._ctx.moveTo(x,y);

    //Winkel im Hintergrund für Fingerglied 1
    //Kreisbogen vom Ausgangspunkt mit Start-Winkel 1,5 Pi und dem ersten Messwert als Endwinkel
    this._ctx.arc(x,y,this._finger.radius/2,1.5*Math.PI,this._finger.angle_1);
    //Linientyp: gestrichelt
```

```
    this._ctx.setLineDash([5]);
    //Cursor auf Ausgangspunkt zurücksetzen
    this._ctx.moveTo(x,y);
    //Endpunkt der gestrichelten Linie definieren
    x = this._finger.x;
    y = this._finger.y-this._finger.radius;
    //Linie zeichnen
    this._ctx.lineTo(x,y);
    //Linienende rechteckig abgeschnitten
    this._ctx.lineCap="square";
    //Linienstärke: 1 Pixel
    this._ctx.lineWidth=1;
    //Linienfarbe: schwarz
    this._ctx.strokeStyle="#000000";

    //Winkel im Hintergrund für Fingerglied 2
    //Ausgangspunkt für zweites Fingerglied definieren
    x = this._finger.x + this._finger.radius*(Math.cos(this._finger.angle_1));
    y = this._finger.y + this._finger.radius*(Math.sin(this._finger.angle_1));
    //Cursor zum Ausgangspunkt bewegen
    this._ctx.moveTo(x,y);
    //Kreisbogen zeichnen
    this._ctx.arc(x,y,this._finger.radius/2,this._finger.angle_1,this._finger.angle_1+this._finger.angle_2-
2*Math.PI);
    //Cursor zurück auf Ausgangspunkt
    this._ctx.moveTo(x,y);
    //Endpunkt der gestrichelten Linie definieren
    x = this._finger.x + 2*this._finger.radius*(Math.cos(this._finger.angle_1));
    y = this._finger.y + 2*this._finger.radius*(Math.sin(this._finger.angle_1));
    //Linie zeichnen
    this._ctx.lineTo(x,y);
    //Linienende rechteckig abgeschnitten
    this._ctx.lineCap="square";
    //Linienstärke: 1 Pixel
    this._ctx.lineWidth=1;
    //Linienfarbe: schwarz
    this._ctx.strokeStyle="#000000";
    //Pfad zeichnen
    this._ctx.stroke();

    //Winkel im Hintergrund für Fingerglied 3, analog zu 1 und 2
    x = this._finger.x +
this._finger.radius*(Math.cos(this._finger.angle_1))+this._finger.radius*(Math.cos(this._finger.angle_2+thi
s._finger.angle_1));
    y = this._finger.y +
this._finger.radius*(Math.sin(this._finger.angle_1))+this._finger.radius*(Math.sin(this._finger.angle_2+thi
s._finger.angle_1));
    this._ctx.moveTo(x,y);
    this._ctx.arc(x,y,this._finger.radius/2,this._finger.angle_1+this._finger.angle_2-
2*Math.PI,this._finger.angle_1+this._finger.angle_2-4*Math.PI+this._finger.angle_3);
    this._ctx.moveTo(x,y);
    x = this._finger.x + 2*this._finger.radius*(Math.cos(this._finger.angle_2+this._finger.angle_1))+
this._finger.radius*(Math.cos(this._finger.angle_1));
    y = this._finger.y + 2*this._finger.radius*(Math.sin(this._finger.angle_2+this._finger.angle_1))+
this._finger.radius*(Math.sin(this._finger.angle_1));
    this._ctx.lineTo(x,y);
    this._ctx.lineCap="square";
    this._ctx.lineWidth=1;
    this._ctx.strokeStyle="#000000";
    this._ctx.stroke();
    this._ctx.restore();

    //Zeichencursor zurücksetzen auf Ausgangspunkt des ersten Fingerglieds
    this._ctx.beginPath();
    var x = this._finger.x;
    var y = this._finger.y;
    this._ctx.moveTo(x,y);


//-----------------------------------------------------
//2. Anzeigelayer: Einzelne Fingerglieder einzeichnen
//-----------------------------------------------------

    //Endpunkt für erstes Fingerglied definieren
    x = this._finger.x + this._finger.radius*(Math.cos(this._finger.angle_1));
    y = this._finger.y + this._finger.radius*(Math.sin(this._finger.angle_1));
    //Linie zeichnen
    this._ctx.lineTo(x,y);
    //Linienende bzw. Linienübergang als Halbkreis bzw. abgerundet
    this._ctx.lineCap="round";
    //Linienstärke: 10 Pixel
    this._ctx.lineWidth=10;
    //Linienfarbe: dunkelgrau
    this._ctx.strokeStyle="#717171";
    //Pfad zeichnen
    this._ctx.stroke();

    //Ausgangskonfiguration speichern - um folgende Translation zurückzusetzen
    this._ctx.save();
```

```
    //Translation des Koordinatensystem auf Endpunkt des 1. Fingerglieds
    this._ctx.translate(x,y);
    //Endpunkt für zweites Fingerglied definieren
    x = this._finger.radius*(Math.cos(this._finger.angle_2+this._finger.angle_1));
    y = this._finger.radius*(Math.sin(this._finger.angle_2+this._finger.angle_1));
    //Linie zeichnen
    this._ctx.lineTo(x,y);
    //Linienende: Abgerundet
    this._ctx.lineCap="round";
    //Linienstärke: 8 Pixel
    this._ctx.lineWidth=8;
    //Linienfarbe: dunkelgrau
    this._ctx.strokeStyle="#717171";
    //Pfad zeichnen
    this._ctx.stroke();

    //Translation des Koordinatensystems auf Endpunkt des 2. Fingerglieds
    this._ctx.translate(x,y);
    //3. Fingerlied analog zu 1 und 2, Linienstärke 6
    x = this._finger.radius*(Math.cos(this._finger.angle_3+this._finger.angle_2+this._finger.angle_1));
    y = this._finger.radius*(Math.sin(this._finger.angle_3+this._finger.angle_2+this._finger.angle_1));
    this._ctx.lineTo(x,y);
    this._ctx.lineCap="round";
    this._ctx.lineWidth=6;
    this._ctx.strokeStyle="#717171";
    this._ctx.stroke();
    this._ctx.restore();


//-----------------------------------------------------------------
//3. Anzeigelayer: Farbige Kreise in Fingergelenken einzeichnen
//-----------------------------------------------------------------

    //Farbiger Kreis am Ausgangspunkt des 1. Fingerglieds
    //Neuen Pfad beginnen
    this._ctx.beginPath();
    //Mittelpunkt des Kreises definieren
    x = this._finger.x;
    y = this._finger.y;
    //Cursor auf Mittelpunkt setzen
    this._ctx.moveTo(x,y);
    //Vollkreis zeichnen mit Radius 10 Pixel
    this._ctx.arc(x,y,10,0,2*Math.PI);
    //Füllfarbe: grün
    this._ctx.fillStyle="#00db00";
    //gezeichneten Kreis ausfüllen
    this._ctx.fill();

    //Farbiger Kreis am Ausgangspunkt des 2. Fingerglieds
    this._ctx.beginPath();
    x = this._finger.x + this._finger.radius*(Math.cos(this._finger.angle_1));
    y = this._finger.y + this._finger.radius*(Math.sin(this._finger.angle_1));
    this._ctx.moveTo(x,y);
    this._ctx.arc(x,y,8,0,2*Math.PI);
    this._ctx.fillStyle="#e2e200";
    this._ctx.fill();
    this._ctx.save();
    this._ctx.translate(x,y);

    //Farbiger Kreis am Ausgangspunkt des 3. Fingerglieds
    this._ctx.beginPath();
    x = this._finger.radius*(Math.cos(this._finger.angle_2+this._finger.angle_1));
    y = this._finger.radius*(Math.sin(this._finger.angle_2+this._finger.angle_1));
    this._ctx.moveTo(x,y);
    this._ctx.arc(x,y,6,0,2*Math.PI);
    this._ctx.fillStyle="#ff5252";
    this._ctx.fill();
    this._ctx.translate(x,y);

    //Schwarzgrauer Kreis am Endpunkt des 3. Fingerglieds
    this._ctx.beginPath();
    x = this._finger.radius*(Math.cos(this._finger.angle_3+this._finger.angle_2+this._finger.angle_1));
    y = this._finger.radius*(Math.sin(this._finger.angle_3+this._finger.angle_2+this._finger.angle_1));
    this._ctx.moveTo(x,y);
    this._ctx.arc(x,y,4,0,2*Math.PI);
    this._ctx.fillStyle="#555555";
    this._ctx.fill();
    this._ctx.translate(x,y);
    this._ctx.restore();


//-----------------------------------------------------------------------
//4. Anzeigelayer: Beschriftungen innerhalb der farbigen Kreise einfügen
//-----------------------------------------------------------------------

    this._ctx.save();
    //Textfarbe: schwarz
    this._ctx.fillStyle="#000000";
```

```
    //Beschriftung alpha
    //Texgröße 17 Pixel, Schriftart Arial
    this._ctx.font="17px Arial";
    //Textausrichtung in vertikaler Richtung mittig
    this._ctx.textBaseline="middle";
    //Textausrichtung in horizontaler Richtung mittig
    this._ctx.textAlign="center";
    //alpha auf Ausgangspunkt von Fingerglied 1 schreiben
    this._ctx.fillText("\u03B1",this._finger.x,this._finger.y);

    //Beschriftung beta
    x = this._finger.x + this._finger.radius*(Math.cos(this._finger.angle_1));
    y = this._finger.y + this._finger.radius*(Math.sin(this._finger.angle_1));
     this._ctx.font="15px Arial";
    this._ctx.fillText("\u03B2",x,y);

    //Beschriftung gamma
    x = this._finger.x +
this._finger.radius*(Math.cos(this._finger.angle_1))+this._finger.radius*(Math.cos(this._finger.angle_2+thi
s._finger.angle_1));
    y = this._finger.y +
this._finger.radius*(Math.sin(this._finger.angle_1))+this._finger.radius*(Math.sin(this._finger.angle_2+thi
s._finger.angle_1));
     this._ctx.font="13px Arial";
    this._ctx.fillText("\u03B3",x,y);
    this._ctx.restore();


//-------------------------------------------------------------
//5. Anzeigelayer: Balkenanzeigen für Spannungen einzeichnen
//-------------------------------------------------------------

    this._ctx.save();
    //Liniestärke: 3 Pixel
    this._ctx.lineWidth=3;
    //Linienfarbe: schwarz
    this._ctx.strokeStyle="#000000";

    //Balkenanzeige für Spannungswert an Fingerglied 1
    //Ausfüllfarbe: grün
    this._ctx.fillStyle="#00db00";
    //Rechteck einzeichnen mit variabler rechter Seite, abhängig vom Spannungswert
    this._ctx.fillRect(5,493,70/5*this._finger.voltage_1,20);

    //Balkenanzeige für Spannungswert an Fingerglied 2
    //Ausfüllfarbe: gelb
    this._ctx.fillStyle="#e2e200";
    //Rechteck einzeichnen mit variabler rechter Seite, abhängig vom Spannungswert
    this._ctx.fillRect(85,493,70/5*this._finger.voltage_2,20);

    //Balkenanzeige für Spannungswert an Fingerglied 3
    //Ausfüllfarbe: rot
    this._ctx.fillStyle="#ff5252";
    //Rechteck einzeichnen mit variabler rechter Seite, abhängig vom Spannungswert
    this._ctx.fillRect(165,493,70/10*this._finger.voltage_3,20);

    //Schwarze Rahmen um Balkenanzeigen einzeichnen
    this._ctx.strokeRect(5,493,70,20);
    this._ctx.strokeRect(85,493,70,20);
    this._ctx.strokeRect(165,493,70,20);
    this._ctx.restore();


//-------------------------------------------------------------
//6. Anzeigelayer: Halbkreisanzeigen für Winkel einzeichnen
//-------------------------------------------------------------

    //Ausgangskonfig speichern
    this._ctx.save();

    //Halbkreisanzeige für Winkel an Fingerglied 1
    //Neuen Pfad beginnen
    this._ctx.beginPath();
    //Cursor Auf Mittelpunkt des ersten Halbkreises bewegen
    this._ctx.moveTo(40,432);
    //Kreissegment zeichnen mit Endwinkel abhängig von eingelesenem Wert
    this._ctx.arc(40,432,35,Math.PI,this._finger.angle_1-Math.PI/2);
    //Zeichenpfad schließen
    this._ctx.closePath();
    //Ausfüllfarbe: grün
    this._ctx.fillStyle="#00db00";
    //Gezeichnetes Kreissegment ausfüllen
    this._ctx.fill();

    //Halbkreisanzeige für Winkel an Fingerglied 2
    this._ctx.beginPath();
    this._ctx.moveTo(120,432);
    this._ctx.arc(120,432,35,Math.PI,this._finger.angle_2-Math.PI);
    this._ctx.closePath();
```

```
    this._ctx.fillStyle="#e2e200";
    this._ctx.fill();

    //Halbkreisanzeige für Winkel an Fingerglied 3
    this._ctx.beginPath();
    this._ctx.moveTo(200,432);
    this._ctx.arc(200,432,35,Math.PI,this._finger.angle_3-Math.PI);
    this._ctx.closePath();
    this._ctx.fillStyle="#ff5252";
    this._ctx.fill();
    this._ctx.restore();

    //Schwarze Rahmen um Halbkreisanzeigen einzeichnen
    this._ctx.save();
    //Linienstärke: 3 Pixel
    this._ctx.lineWidth=3;
    //Linienfarbe: schwarz
    this._ctx.strokeStyle="#000000";
    this._ctx.beginPath();
    //Halbkreis zeichnen
    this._ctx.arc(40,432,35,Math.PI,2*Math.PI);
    this._ctx.closePath();
    this._ctx.stroke();
    this._ctx.beginPath();
    this._ctx.arc(120,432,35,Math.PI,2*Math.PI);
    this._ctx.closePath();
    this._ctx.stroke();
    this._ctx.beginPath();
    this._ctx.arc(200,432,35,Math.PI,2*Math.PI);
    this._ctx.closePath();
    this._ctx.stroke();
    this._ctx.restore();


//-----------------------------------------------------------------------------------------
//7. Anzeigelayer: Zahlenwerte mit Einheit in Balken- und Kreissegmentanzeigen einzeichnen
//-----------------------------------------------------------------------------------------

    this._ctx.save();
    this._ctx.textAlign="center";
    this._ctx.fillStyle="#000000";

    //Zahlenwerte für Winkel in Kreissegmentanzeigen
    this._ctx.font="18px Arial";
    //Überschrift schreiben
    this._ctx.fillText("Winkel [°]",120,390);
    this._ctx.font="20px Courier";
    //Werte in Mitte der Anzeigen schreiben
    this._ctx.fillText(this._finger.angle_deg_1.toFixed(0),40,420);
    this._ctx.fillText(this._finger.angle_deg_2.toFixed(0),120,420);
    this._ctx.fillText(this._finger.angle_deg_3.toFixed(0),200,420);

    //Zahlenwerte für Spannungen in Balkenanzeigen
    this._ctx.font="18px Arial";
    //Überschrift schreiben
    this._ctx.fillText("Spannungen [V]",120,480);
    this._ctx.font="20px Courier";
    //Werte in Mitte der Anzeigen schreiben
    this._ctx.fillText(this._finger.voltage_1.toFixed(2),40,510);
    this._ctx.fillText(this._finger.voltage_2.toFixed(2),120,510);
    this._ctx.fillText(this._finger.voltage_3.toFixed(2),200,510);
    this._ctx.restore();

};



//-----------------------------------------------------
//Prototypen-Deklaration der Zeichen-Thread Methoden
//-----------------------------------------------------

//Methode Zeichnen des Fingers und anschließendem Leeren des Canvas,
//damit mit aktuellen Werten erneut gezeichnet werden kann
GUI.prototype.draw = function()
{
    //Sicherheitsabfrage ob getContext("2d") definiert wurde
    if (typeof this._ctx != 'undefined')
    {
        //Vollständige Canvas-Fläche löschen
        this._ctx.clearRect(0, 0, this._canvas.width, this._canvas.height);
        //Wenn die visible-Eigenschaft des Finger-Objekts durch die Start-Methode auf true gesetzt wurde
        if (this._finger.visible)
        {
            //Finger mit aktuellen Werten erneut zeichnen
            this.drawfinger();
        }
    }
};
```

```
//Methode zum Starten des GUI-Threads
GUI.prototype.start = function()
{
    var that = this;
    //visible-Eigenschaft des Finger-Objekts auf true setzen
    this._finger.visible = true;
    //Im GUI-Thread wird die draw-Methode in 50ms Abständen ausgeführt um eine flüssige Animation zu
erhalten
    this._GUIThread = window.setInterval(function()
    {
        that.draw();
    }, 50);
};

//Methode zum Stoppen des GUI-Threads um die Animation anzuhalten
GUI.prototype.stop = function()
{
    //Rücksetzen der visible-Eigenschaft des Finger-Objekts auf false
    this.finger.visible = false;
    //Letztes, Einmaliges Auffrufen der Zeichenmethode um Canvas zu leeren
    this.draw();
    //Intervallausführung des GUI-Threads stoppen
    window.clearInterval(this.GUIThread);
};


//-------------------------------------------------------------------------------------------------------------
------------
//Globales app-Objekt für den Empfang von Werten über BLE sowie der Zurückrechnung der Byte-Stream-Werte in
float-Werte
//-------------------------------------------------------------------------------------------------------------
------------

window.app =
{

    //Initialisierung des GUI Objekts
    GUI: null,

    //Initialisierungsmethode
    initialize: function()
    {
        //Erstellt eine neue GUI-Instanz und weist diese dem GUI-Objekt zu
        this.GUI = new GUI($('#finger_canvas')[0]);
        //Ruft die unten deklarierte bindEvents-Methode auf
        this.bindEvents();
    },


    //Verbindet Event(s) mit Element(en) in index.html
    bindEvents: function()
    {
        var that = this;
        //Eventhandler zum Ausführen der connect-Methode
        //mittels jQuery-.on-Methode für click-Event auf connect_ble-Button
        $('#connect_ble').on('click', this.connect);
    },



//------------------------------------------------------------
//Implementierung der BLE-Central Cordova-Plugin Funktionen
//------------------------------------------------------------

    //connect-Methode - Scannt nach BLE-Geräten, Startet onDiscoverDevice-Methode
    //nach erfolgreichem Auffinden eines kompatiblen Geräts
    connect: function()
    {
        var that = this;
        //Sucht uneingeschränkt nach allen BLE-Services für 5 Sekunden
        ble.scan([],5,window.app.onDiscoverDevice);
    },



    //onDiscoverDevice-Methode - Wird bei Auffinden eines kompatiblen BLE-Geräts aufgerufen
    onDiscoverDevice: function(device)
    {
        var that;
        //Wenn das Gefundene Gerät den in den Voreinstellungen definierten Namen hat
        if (device.name == window.edison.name)
        {
            //Wird die ID des Geräts unserem Edison-Objekt zugewiesen
            window.edison.deviceId = device.id;
            //Und versucht eine Verbindung zu dieser ID herzustellen
            //Bei Erfolg wird die onConnect-Methode aufgerufen
            ble.connect(window.edison.deviceId,
                        window.app.onConnect);
        }
```

```
    },


    //onConnect-Methode - startet GUI und registriert das onNotify-Callback, das jedes mal aufgerufen wird,
    //wenn sich der Wert einer BLE-Characteristic ändert
    onConnect: function()
    {
        window.app.GUI.start();
        ble.startNotification(window.edison.deviceId,
                              window.edison.service,
                              window.edison.characteristic,
                              window.app.onNotify);
    },


    //onNotify-Methode - Wird als Callback jedes mal ausgeführt wenn sich eine Bluetooth-Characteristic
ändert
    onNotify: function(data)
    {
        //Zwischenspeicher vom Typ Uint8Array für die als Byte-Stream empfangenen Daten
        var dir = new Uint8Array(data);

        //Zusammensetzen des Spannungswertes für Fingerglied 1 aus Byte 0 und 1
        //analog zur manuellen Codierung von float-Werten zu Byte-Stream für begrenzte Wertebereiche im
Edison-Code
        var value=(dir[0]*255)+dir[1];
        //Kommastelle einfügen
        var floatvalue=value/1000;
        //Aufruf der updatefingerPos-Methode zum Aktualisieren der Objekteigenschaften des ersten
Fingerglieds
        window.app.GUI.updatefingerPos1(floatvalue);

        //Zusammensetzen des Spannungswertes für Fingerglied 2 aus den Byte 2 und 3
        value=(dir[2]*255)+dir[3];
        //Kommastelle einfügen
        floatvalue=value/1000;
        //Aufruf der updatefingerPos-Methode zum Aktualisieren der Objekteigenschaften des zweiten
Fingerglieds
        window.app.GUI.updatefingerPos2(floatvalue);

        //Zusammensetzen des Spannungswertes für Fingerglied 3 aus Byte 4 und 5
        value=(dir[4]*255)+dir[5];
        //Kommastelle einfügen
        floatvalue=value/1000;
        //Aufruf der updatefingerPos-Methode zum Aktualisieren der Objekteigenschaften des dritten
Fingerglieds
        window.app.GUI.updatefingerPos3(floatvalue);
    },
};

// jQuery(document).ready()-Methode um JavaScript jQuery-Callbacks den HTML-Elementen in index.html
zuzuweisen
$(function()
{
    // app-Objekt initialisieren und jQuery-Callbacks zuweisen (durch Definition mit vorangestelltem $)
    window.app.initialize();
});
```