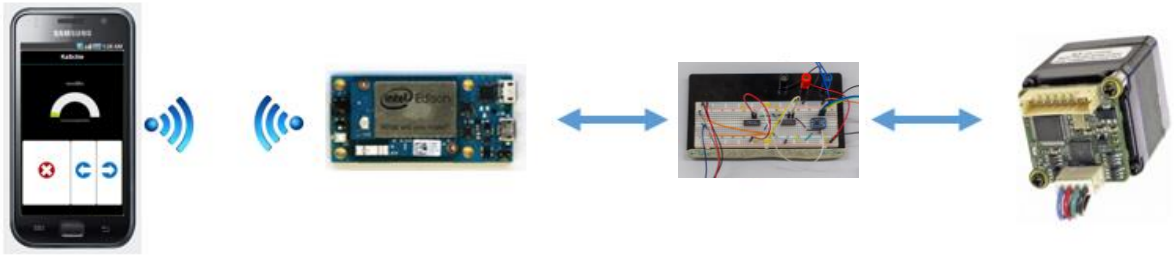
Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Steuerung eines Schrittmotors mit Intel Edison über eine App

Gruppennummer 5 und 6



Informationstechnik Labor Sommersemester 2016

Prof. J. Walter

Gruppenmitglieder:

Tobias Mildenberger	*41294*
Thomas Bertram	*40887*
Christian Biesenthal	*41297*
Johanna Funk	*40069*

Inhalt

Problemstellung

Im Rahmen des Internet of Things (IoT) gibt es noch keine preisgünstige Lösung um Schrittmotoren über ein kabelloses Netzwerk (WLAN) zu steuern.

Stand der Technik

Auf dem Markt gibt es noch keine appgesteuerten Kamera-Schlitten. Allerdings gibt es ein Kickstarter-Projekt, das T-set von Axsy, siehe Abb. 1. Dieses ist mit 3 Motoren ausgestattet und wird über das Smartphone mit Hilfe einer separaten Steuereinheit bedient. Preislich liegt das Projekt bei 1150 Euro.

Abbildung 1 Appgesteurtes T-set von Axsy

Es sind außerdem App-gesteuerte Kamera-Drehteller oder Panoramaköpfe erhältlich, siehe Abb. 2. Diese liegen preislich bei ca. 300 Euro.

Abbildung 2 App-gesteuerter Panoramakopf und Drehteller

Aufgabenstellung

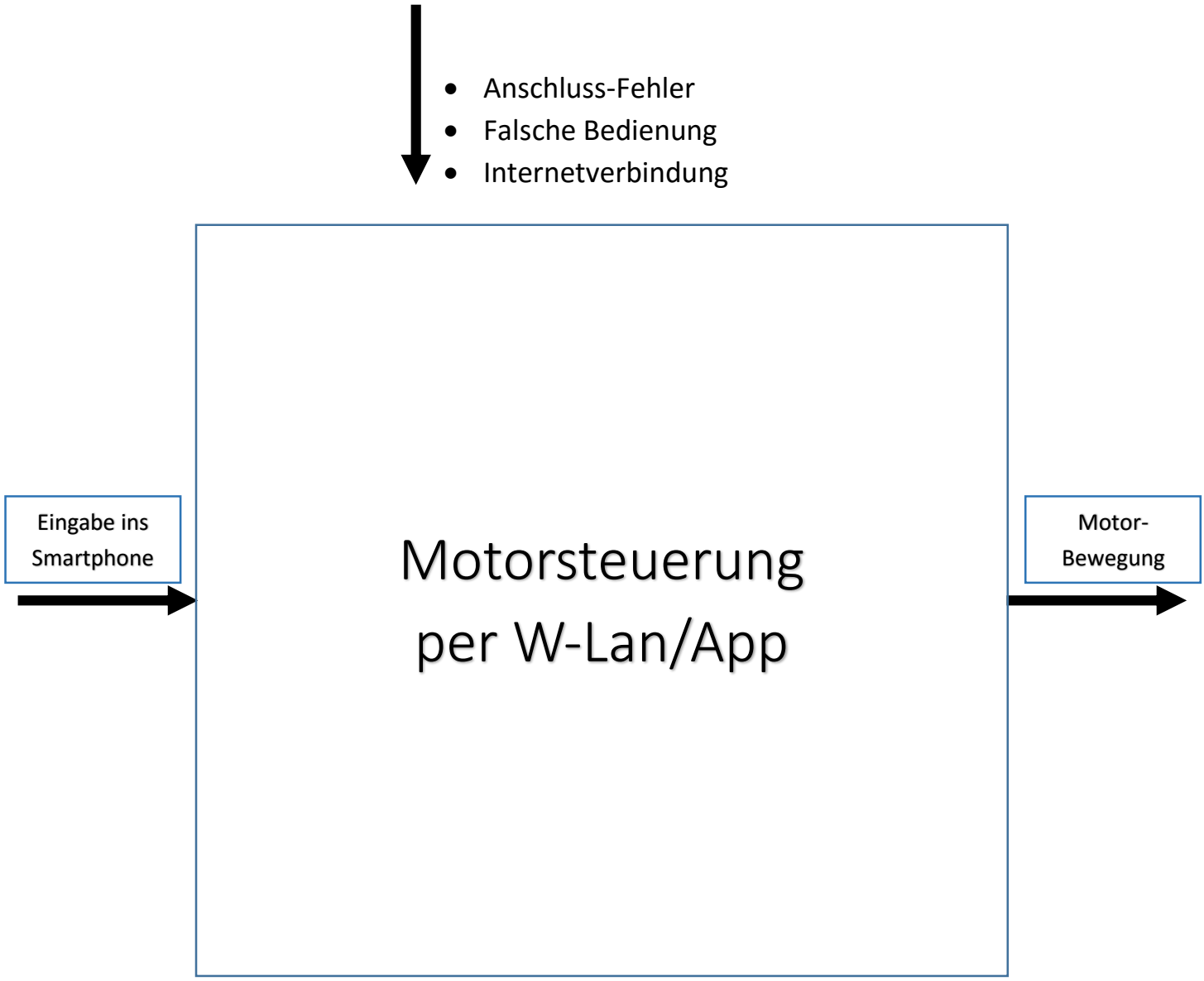
Es soll eine App geschrieben werden, mit der ein Schrittmotor kabellos angesteuert werden kann. Dazu soll das Ein-Chip-System, Intel Edison, ein WLAN erstellen, in das sich diese App anmelden kann und Signale von dieser an den Schrittmotor weiterleiten.

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Anforderungsliste

Prof. Dr.-Ing. Peter Weber Konstruktion und Produktmanagement Hochschule Karlsruhe Technik und Wirtschaft				Anforderungsliste für **Ansteuerung eines Schrittmotors per WLAN**				**Auftrag:** **Formblatt-AnfListe** **Auftragsnummer** SS 2016
Organisations-Daten		**Prozess-Daten**		**Anforderungen**	**Wert – Daten**			
Nummer	Name	Art	Phase		Mindest-Erfüllung	SOLL-Erfüllung	Ideal-Erfüllung	Einheit
				Physikalisch-Technische Funktion				
F01		F		Einstellbare Geschwindigkeit				
F02		F		Einstellbare Verfahr-Richtung	Links, rechts	Links, Rechts	Links, Rechts	-
F03		J/N		Stop-Funktion				
F04		J/N		Start-Funktion				
F05		F		Kompatibilität (An = Android, Ap = Apple, M = Microsoft)	An	An, Ap	An, Ap, M	
F06		J/N		Loop-Fahrt-Funktion zw. zwei Positionen				
F07		J/N		Virtueller Anschlag				
F08		F		Verfahrgenauigkeit	+-1	+-0,5	+-0,1	mm
F09		W		Geräuscharmer Betrieb				
F10		W		Zustandsanzeige (L = Loop, Ans = Anschlag, a = aktiv, i = inaktiv)	a, i	a, i, Ans	a, i , Ans, L	
				Technologie				
T01		J/N		W-Lan Steuerung				
T02		J/N		Intel Edison				
T03		J/N		Schrittmotor Trinamic PD1021				
T04		J/N		Intel XDK				
				Wirtschaftlichkeit				
W01		F		Kostengünstig	<500	<400	<250	Euro
				Mensch-Produkt-Beziehungen				
M01		W		Intuitives Nutzerdesign				
M02		W						
M03		W						
Anforderungsarten: J/N - Ja/Nein; F - Forderung; W - Wunsch; ***Konstruktionsphase:*** P - Prinzip; K - Konzept; E - Entwurf; A - Ausarbeitung								
Ersetzt Ausgabe vom **14.04.2016** Version: **6** Name:							**Ausgabe:** **14.04.2016** **Version:** **7** **Blatt 1 von 1**	

Blackbox


Anschluss-Fehler
Falsche Bedienung
Internetverbindung
Eingabe ins Smartphone
Motorsteuerung per W-Lan/App
Motor-Bewegung

Blockschaltbild

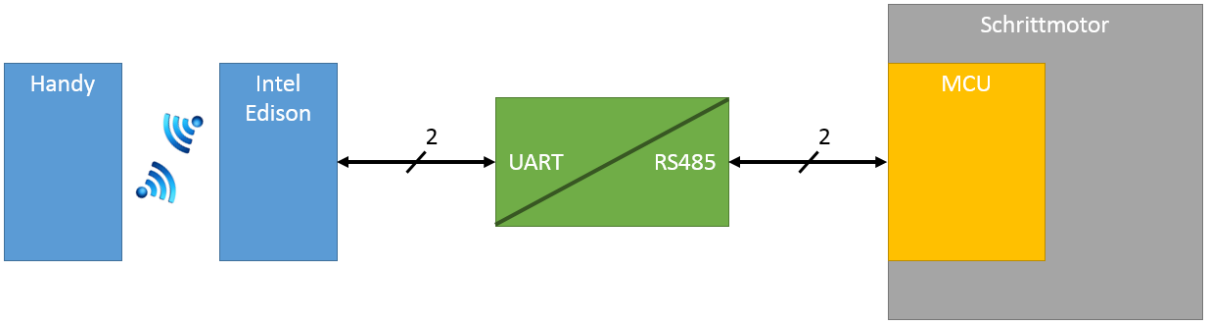


Abbildung 3 Blockschaltbild

Über eine App auf dem Handy soll ein Schrittmotor angesteuert werden. Das Smartphone soll über WLAN mit dem Intel Edison kommunizieren. Dieser soll die Informationen verarbeiteten und in für den Schrittmotor verständliche Befehle übersetzen. Der Motor kann nur über RS485 kommunizieren. Deshalb muss das UART Signal des Edison in das Differenzielle RS485 umgewandelt werden.

Zeitplan

Informationstechnik Labor Fakultät MMT Hochschule Karlsruhe Technik und Wirtschaft		**Zeitplan** für **Titel**							**MTB732** Sommersemester 2016 Datum
Nr.	**Aktivitäten**	2016							**Jahr**
		März			April			Mai	**Monat**
		11	12	13	14	15	16	17	18	**Kalenderwoche**
1	Projektdefinition erstellen	X								
2	Recherche	X	X							
3	Anforderungsliste		X							
4	Blackbox		X							
5	Aufbau und Testen			X	X	X				
6	Edison programmieren					X	X	X		
7	Dokumentation	X	X	X	X	X	X	X		
8	Abschlusspräsentation								X	P
		Meilensteine: 1 2							Präsentation: P	

Stückliste

Bauteilbezeichnung	**Erläuterung**
USB-RS485 Converter	Verbindung Computer zu Schrittmotor
Schrittmotor TRINAMIC 1021 TMCL	Vorhanden
Schnittstellen IC MC3486 und MC3487	Vorhanden
Intel Edison	Vorhanden
Level Shifter	Von 5V auf 1,8V für UART Verbindung zu Edison
Breadboard	Zum Aufstecken der Schnittstelle RS485-UART
Jumperwire	Zum Aufstecken aufs Breadboard

Aufbau von Webserver und App

Der Aufbau lässt sich aufteilen in Webserver und App, was zwei unterschiedliche Programme sind.

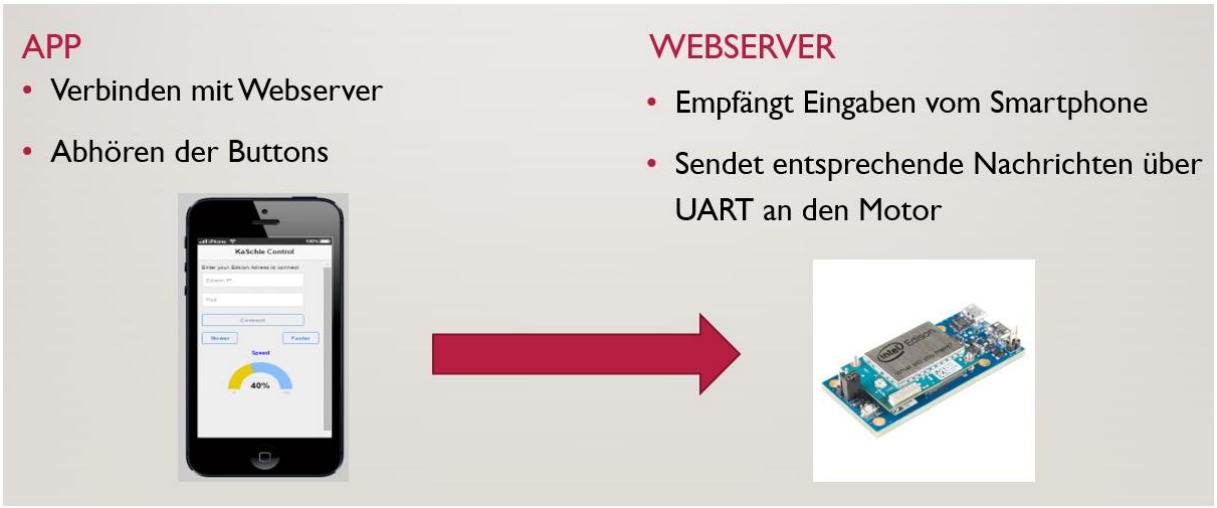


Abbildung 4 Übersicht Kommunikation App und Webserver

App

Die App ist in HTML5 und JavaScript geschrieben. Sie kann auf nahezu jedem beliebigen Smartphone oder Tablet betrieben werden. Damit Webserver und App miteinander kommunizieren können, wird das von Sparkfun heruntergeladene socket.io für Client und Server verwendet. „socketio.js“ wird dazu in den lib-Ordner der App kopiert und eingebunden.

Das Aussehen (Body) der App befindet sich in index.html. Dieser Code wurde mit dem App-Designer erzeugt und ist deshalb sehr umfangreich.

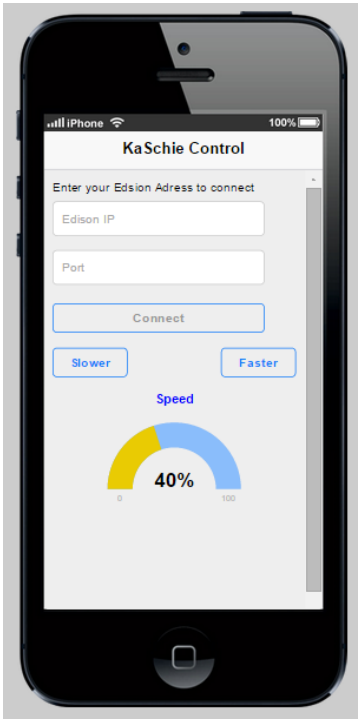


Abbildung 5 Body der Motorsteuerungs-App

In index_user_scripts.js befinden sich die wesentlichen Funktionen der App. Im Prinzip hört die App nur ab, welche Buttons vom Nutzer in der App betätigt wurden, und sendet eine entsprechende Nachricht an den Webserver.

```
 /* graphic button  #bt_left */
$(document).on("click", "#bt_left", function(evt)
{
    window.socket.emit("bt_left");
});

    /* graphic button  #bt_stop */
$(document).on("click", "#bt_stop", function(evt)
{
    window.socket.emit("bt_stop");
});

    /* graphic button  Right */
$(document).on("click", "#bt_right", function(evt)
{
    window.socket.emit("bt_right");
});
```

Abbildung 6 Abhören der Buttons und Senden an Webserver

Die Geschwindigkeitsregelung des Motors lässt sich Schrittweise um je 5% erhöhen. Dazu wurde ein JustGauge verwendet.

```
    /* button  Slower */
$(document).on("click", "#bt_slow", function(evt)
{
    if(speedHolder > 0)
   {
    speedHolder = speedHolder-5;
    var SpeedDis = window.Gauges.getById('speed_gage');
    SpeedDis.refresh(speedHolder);
    window.socket.emit("bt_slow");
   }


});

    /* button  Faster */
$(document).on("click", "#bt_fast", function(evt)
{

    if(speedHolder < 100)
   {
    speedHolder = speedHolder+5;
    var SpeedDis = window.Gauges.getById('speed_gage');
    SpeedDis.refresh(speedHolder);
    window.socket.emit("bt_fast");
   }
});
```

Abbildung 7 Verwendung des JustGauge

Um sich mit dem Webserver zu verbinden, muss die IP des Intel Edison eingegeben werden. Der Port 1111 ist für die Anwendung reserviert. Die Buttons zur Bedienung des Motors erscheinen erst, wenn die App sich erfolgreich mit dem Webserver verbunden hat.

```
    var ip = document.getElementById("ip").value;
var port = document.getElementById("port").value

window.socket = io.connect("http://" + ip + ":" + port);

window.socket.on("connect_error", function() {
    window.socket.disconnect();
   document.getElementById("bt_left").style.display = "none";
    document.getElementById("bt_stop").style.display = "none";
    document.getElementById("bt_right").style.display = "none";
    alert("Could not connect");
});

window.socket.on("connect", function() {
    document.getElementById("bt_left").style.display = "inherit";
    document.getElementById("bt_stop").style.display = "inherit";
    document.getElementById("bt_right").style.display = "inherit";
    alert("Connection established!")

});
```

Abbildung 8 Verbinden mit dem Webserver

Webserver

Der Webserver läuft auf dem Intel Edison und ist in JavaScript programmiert. Das ermöglicht den Zugriff von unterschiedlichen Geräten wie z.B. Tablet oder PC (die App muss auf dem jeweiligen Gerät installiert sein). Über Befehle der App werden hinterlegte Funktionen auf dem Server aktiviert und Befehlssätze per UART verschickt. Dazu wird auch wie bei der App die „socketio.js“ verwendet.

Die Zugangsdaten für den Intel Edison lauten:

IP Adresse: 192.168.0.134

Port (App): 1111

Port(Intel XDK): 58888

Nutzername: root

Passwort: 04EDI04EDI

In der „main.js“ befinden sich die wichtigsten Funktionen sowie die Befehlssätze. Hier ist auch die verwendete UART Schnittstelle deklariert. Diese verwendet **9600 als Baudrate**.

Nach Herstellen einer Verbindung zwischen App und Server können Befehle gesendet werden. Die Buttons Links und Rechts verwenden per Default den Geschwindigkeitswert 100. Durch die Buttons „slower“ und „faster“ wird eine switch/case Funktion aufgerufen die den Wert um jeweils 25 (entspricht 5%) ändert. Umgesetzt wird die neue Geschwindigkeit erst nach erneutem Betätigen der Buttons Links/Rechts.

```
var msg_left7_5 = "01020000000004B0B7";
var msg_left10 = "010200000000005A0A8";
var msg_left12_5 = "01020000000006C0C9";
var msg_left15 = "010200000000081924";
var msg_left17_5 = "01020000000009B8C4";
var msg_left20 = "0102000000000BA9B7";
var msg_left22_5 = "0102000000000DFFF";
var msg_left25 = "01020000000010CBDE";
var msg_left27_5 = "01020000000014273E";
var msg_left30 = "010200000000182F4A";
var msg_left32_5 = "0102000000001D0626";
var msg_left35 = "01020000000022D4F9";
var msg_left37_5 = "01020000000029CBF7";
var msg_left40 = "01020000000032275C";
var msg_left42_5 = "0102000000003C2F6E";
var msg_left45 = "010200000000483883";
var msg_left47_5 = "01020000000056AA3";
var msg_left50 = "010200000000067FF69";
var msg_left52_5 = "0102000000007CCB4A";
```

Abbildung 8 Auszug der UART Befehlssätze

```
switch(speed){

        case 0:
        serialPort.write(hex2str(msg_left0), function(err, results)
        {
        })
        break;

        case 25:
        serialPort.write(hex2str(msg_left2_5), function(err, results)
        {
        })
        break;

        case 50:
        serialPort.write(hex2str(msg_left5), function(err, results)
        {
        })
        break;

        case 75 :
        serialPort.write(hex2str(msg_left7_5), function(err, results)
        {
        })
        break;

        case 100:
        serialPort.write(hex2str(msg_left10), function(err, results)
        {
        })
        break;
```

Abbildung 9 Auszug aus der switch/case Funktion

Eine weitere auswählbare Funktion lässt den Motor eine eingestellte Zeit lang in eine Richtung fahren und anschließend automatisch umkehren. Es kann zwischen drei Geschwindigkeiten ausgewählt werden.

```
  socket.on('bt_long', function() {


      serialPort.write(hex2str(msg_right10), function(err, results){
                });
      sleep(3000);
      serialPort.write(hex2str(msg_left10), function(err, results){
                });


});
```

Abbildung 10 Automatische Fahrfunktion

Verbindung Intel Edison zum Motor

Der Schrittmotor TMCM-1021 von Trinamic kommuniziert nur über eine RS485 Schnittstelle, da der Intel Edison nicht über diese verfügt, musste eine Signalumwandlung erstellt werden. Hierzu wurden die Bausteine MC3486L (Receiver) und MC3487L (Driver) verwendet um das differentielle RS485 Signal in ein UART Signal zu wandeln.

Vorgehensweise

Zu Beginn wurde versucht den Motor direkt über einen USB-RS485 Adapter anzusteuern. Dies wurde mithilfe der Software TMCL-IDE realisiert, die von Trinamic zur Verfügung gestellt wird.

Im nächsten Schritt wurde mit Eagle ein Schaltplan erstellt und auf dem Breadboard aufgebaut. Auf dem Breadboard befindet sich der UART zu RS485 Adapter (siehe Schaltplan). Über einen FTDI war es nun möglich das RS485 Signal am UART-Ausgang mitzuhören. Dadurch konnte der Adapter auf dem Board getestet werden.

Daraufhin mussten die Signale, die den Motor steuern identifiziert werden. Diese wurden ebenfalls über den FTDI ermittelt. Nachdem einige Befehlssätze bekannt waren, wurde versucht den Motor direkt über den FTDI ohne die TMCL-DIE Software anzusteuern.

BEFEHL	ADRESSE	INSTRUCTION	TYPE	MOTOR/BANK	BYTE 4	3	2	1	CHECKSUM
ROR	01	01	00	00	0 - 1000000 in Hex				errechnen
ROL	01	02	00	00	0 - 1000000 in Hex				errechnen
MST	01	03	00	00	00	00	00	00	04

Tabelle1: Motorbefehle

Nach erfolgreichen Versuchen ist ein Excel File erstellt worden, in dem alle nötigen Befehle definiert wurden. Die Befehle bestehen aus Hexadezimalzahlen, sowie einem Kontrollbyte am Ende des Befehlssatzes. Nun mussten alle Befehle in das Java-Programm eingefügt werden.

Damit der Edison über UART kommunizieren kann wurde noch ein LevelShifter auf dem Breadboard hinzugefügt. Dieser hebt die 1,8V des Edison auf die 5V von UART.

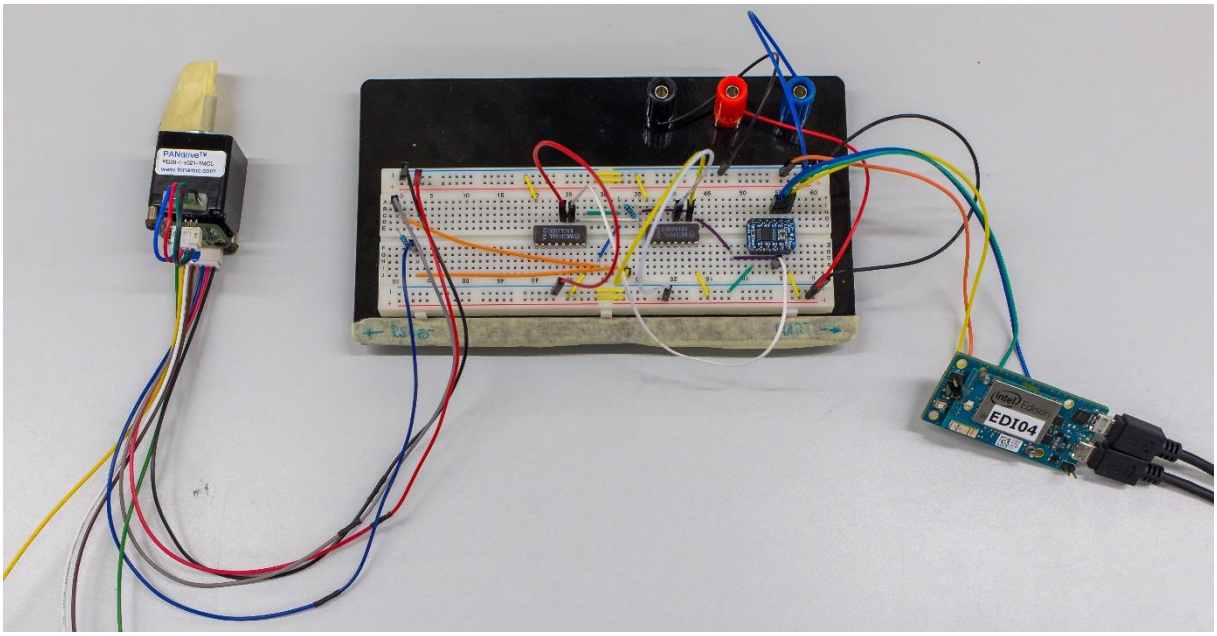


Abbildung 9 Kompletter Aufbau Edison zu Schrittmotor

Nach ersten Tests hat die Kommunikation jedoch nicht funktioniert. Bei der Analsyse der Signale wurde festgestellt, dass der Intel Edison Strings anstelle von Hexadezimalzahlen sendet. Daraufhin wurde ein zusätzliches Programm geschrieben, dass die Hexadezimalzahlen in Strings umwandelt.

Festgestellt wurde auch, dass durch das „Abhören“ der Befehle das Signal gestört wurde. Dies wurde mit Hilfe eines Oszilloskops herausgefunden.

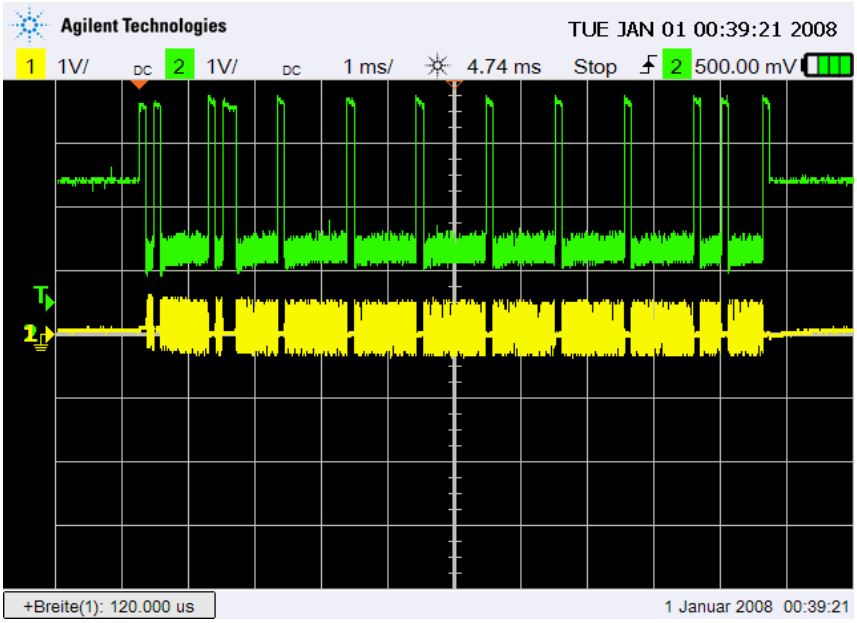


Abbildung 10 Gestörtes Signal

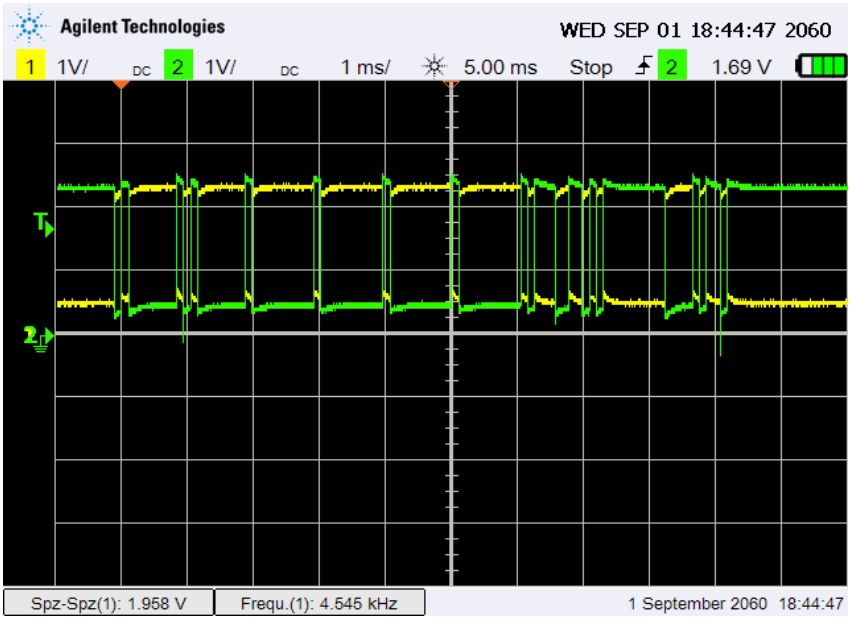


Abbildung 11 Gutes Signal

Mit einem Logic Analyser von Saleae konnten wir störungsfrei an der RS485 Schnittstelle das Signal abgreifen und auswerten.

Abbildung 12 Saleae Logic Analyser - Kommunikation Edison - Schrittmotor

Am Ende ist es möglich den Motor über eine App zu steuern.

Schaltplan

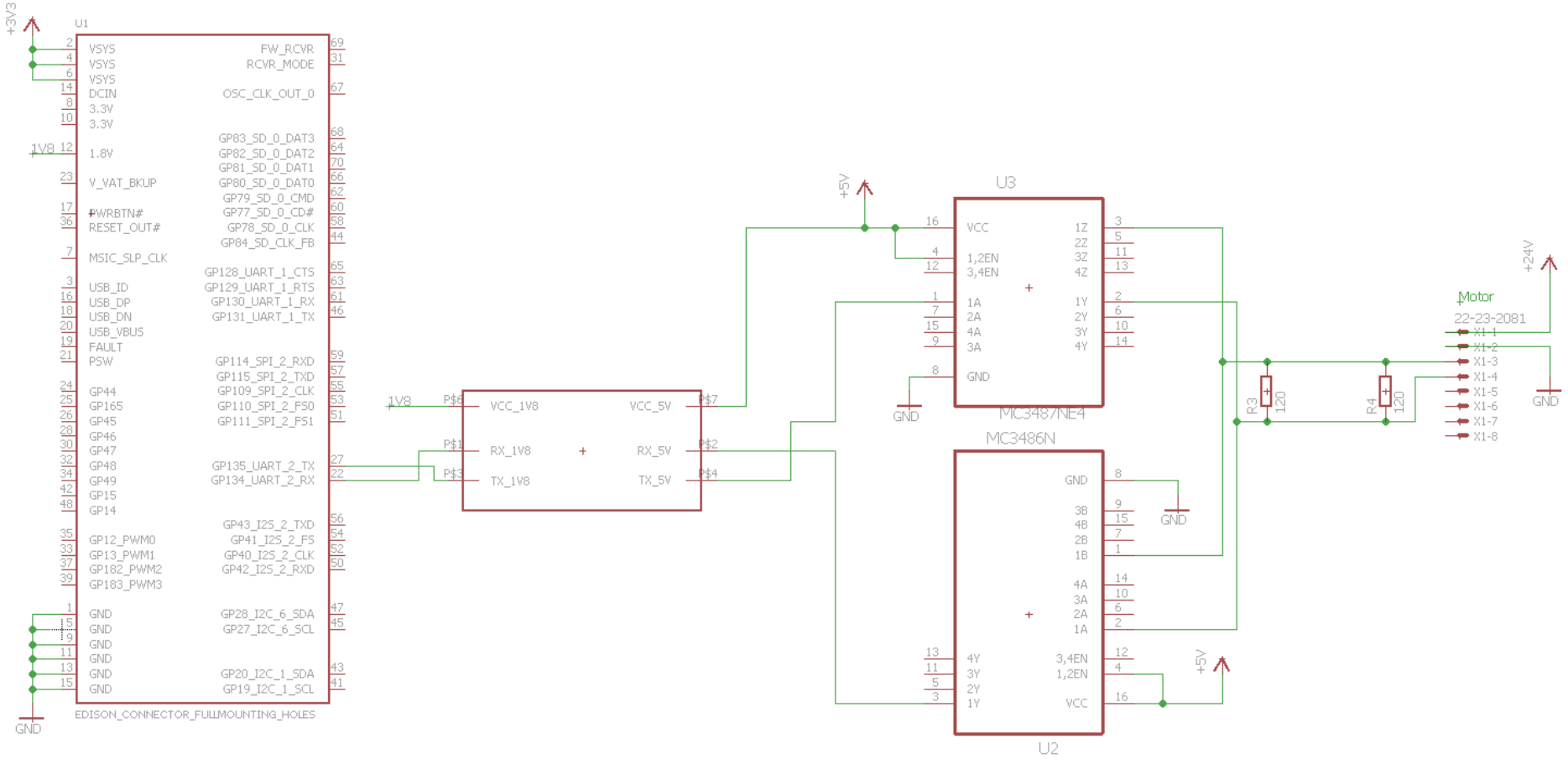


Abbildung 13 Schaltplan