

Macoun

iOS (7) bewegt wieder einmal alles

Frank Jüstel

Ablauf

- Aufgabenstellung
- Lösung Xcode / Objective C
- Lösung OpenPicus IDE / C
- Lösung RobotC / C
- Und so kommt alles zusammen

Aufgabenstellung

Ein kleiner, einfacher, gewöhnlicher,
niedlicher Lego nxt Roboter soll über
das iPhone gesteuert werden



Kein Problem, weil Lego nxt 2.0
kann ja Bluetooth :-)

Ha, ha, iOS unterstützt aber
nicht irgendwelche
dahergelaufene BT-Profile :-)

Ein Übersetzer muss her!

- Übersetzung? Auf was?
- WiFi. Ok, immer eine gute Idee.
- WiFi auf differentiellen Transceiver (auch PartyLine genannt)
- Weil das kennt auch Lego unter dem Begriff:
serial communications high-speed link

Die Qual der Wahl

Arduino: Open-Source-Hardware – Golem.de

Horizon™ Suite
The Power to Empower.

golem.de HOME TICKER
TOP-THEMEN: NSA iPhone 5S Haswell Xbox One Playstation 4 Windows 8 mehr

Arduino: Open-Source-Hardware

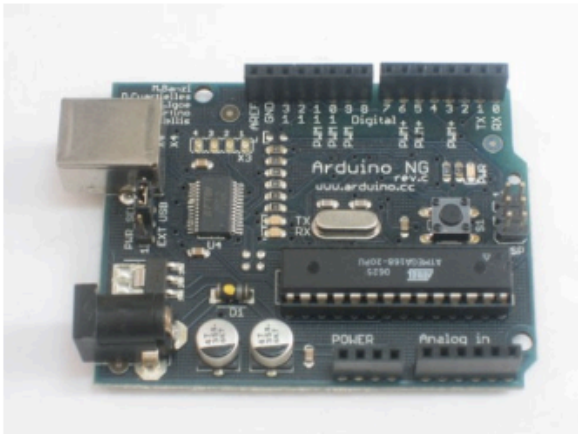
Creative-Commons-Hardware für Künstler und Bastler

Die **Firma** Smart Projects Snc vertreibt unter dem Namen Arduino ein I/O-Board mit Microcontroller samt API und IDE als Open-Source-Projekt. Die Platine darf nachgebaut und für eigene Zwecke entfremdet werden.

Datum: 27.11.2009, 15:40
Autor: Jörg Thoma
Themen: Arduino, GPL, OSHW, API, Open Source, PC-Hardware
Tellen: 0 0 0
Tools: Drucken, E-Mail, Trackback

Google-Anzeigen
Geräteübergreifend
Entwickeln von Apps für viele Geräte. Gratis-Download hier.
www.embarcadero.com/DE
IBM Analytics Lösungen
Von Datenanalysen profitieren. Auch in kleinen Schritten.
ibm.com/de/

Unter dem Namen **Arduino** vertreibt die Firma **Smart Projects Snc** im italienischen Scarmagno eine programmierbare Input-Output-Hauptplatine. Das Layout des Boards und die dazugehörige **Software** sind alle offengelegt, stehen genauer unter der Lizenz Creative Commons Attribution Share-Alike 2.5 beziehungsweise unter der GPL v2.



Ein Arduino NG (Nuova Generazione) mit ATmega168-Microcontroller

Obwohl jeder die Platine nachbauen kann, kommen die Hersteller den Bestellungen nicht hinterher. Allein in diesem Jahr haben sie Arduino und seine Abkömmlinge mehr als 60.000-mal verkauft, letztes Jahr waren es noch 34.000 Stück.

Folgen Sie uns
f t g+ RSS

Videos
Ruby Project Phoenix beim AMD Techday 2013

Verwandte Artikel
Selbstbauroboter überwacht Twitterfeeds
Bruce Perens stellt Busybox-Klage infrage

Arduino – Products

GoToWebinar JSM Arts Willkommen FlatRate Synology Dis...DiskStation Linguee Apple Google Maps YouTube Wikipedia speedio Gesicherte Tabs

Buy Download Products Learning Reference Support Blog LOG IN SIGN UP

BOARDS (Specs Compare)

Arduino Uno
Arduino Leonardo
Arduino Due
Arduino Yún
Arduino Tre

SHIELDS

Arduino GSM Shield
Arduino Ethernet Shield
Arduino WiFi Shield

KITS

The Arduino Starter Kit

ACCESSORIES

TFT LCD screen

Raspberry Pi: Doppelter Arbeitsspeicher bei gleichem Preis » t3n

Startseite Abo Jobbörse Marktplatz Werben Sponsored Post Partnerprogramm Gib uns einen Tippl Anmelden

24 39 22 Drücke die Tasten für weitere Artikel

Raspberry Pi: Doppelter Arbeitsspeicher bei gleichem Preis

Das Modell B des kleinen und überaus günstigen Rechners Raspberry Pi erhält ein kleines aber begrüßenswertes Hardware-Upgrade: Der Arbeitsspeicher des Minirechners wird von 256MB auf 512MB RAM verdoppelt.

Google-Anzeigen
Werbung für Ihre Website
Mit AdWords mehr Kunden erreichen. Heute das Angebot von 756 nutzen.
www.google.de/adwords

Raspberry Pi – Modell B jetzt für anspruchsvollere Anwendungen gewappnet
Wie der **Ankündigung des Hardware-Upgrades** auf der Raspberry Pi-Website zu entnehmen ist, haben sich viele Nutzer ein teureres Model C gewünscht, das mit zusätzlichem Arbeitsspeicher ausgestattet ist. Insbesondere für Nutzer, die den Minirechner als universell einsetzbaren Computer verwenden und mehrere größere Anwendungen parallel laufen lassen möchten, sei dies hilfreich. Ein Rechner mit mehr RAM würde zudem interessante Einsatzzwecke im Embedded-Bereich eröffnen, da hier oftmals Java zum Einsatz kommt und 256MB etwas zu wenig seien, so Raspberry-Pi-Mitgründer Eben Upton.



Der Minirechner Raspberry Pi ist mit doppeltem Arbeitsspeicher für gleiches Geld noch attraktiver geworden.

Schneller drucken
Ab € 399

hp Make it matter.

Autoren: Andreas Floemer
Datum: 15.10.2012, 15:27 Uhr
Themen: Raspberry Pi, Hardware

NEWSLETTER: TOP THEMEN DER WOCHE
E-Mail-Adresse eintragen
2x die Woche Redaktionell bearbeitet
Jetzt lesen: t3n Newsletter Nr. 373

Geräteübergreifend
www.embarcadero.com/DE
Entwickeln von Apps für viele Geräte. Gratis-Download hier.

Ankauf Wohnmobile

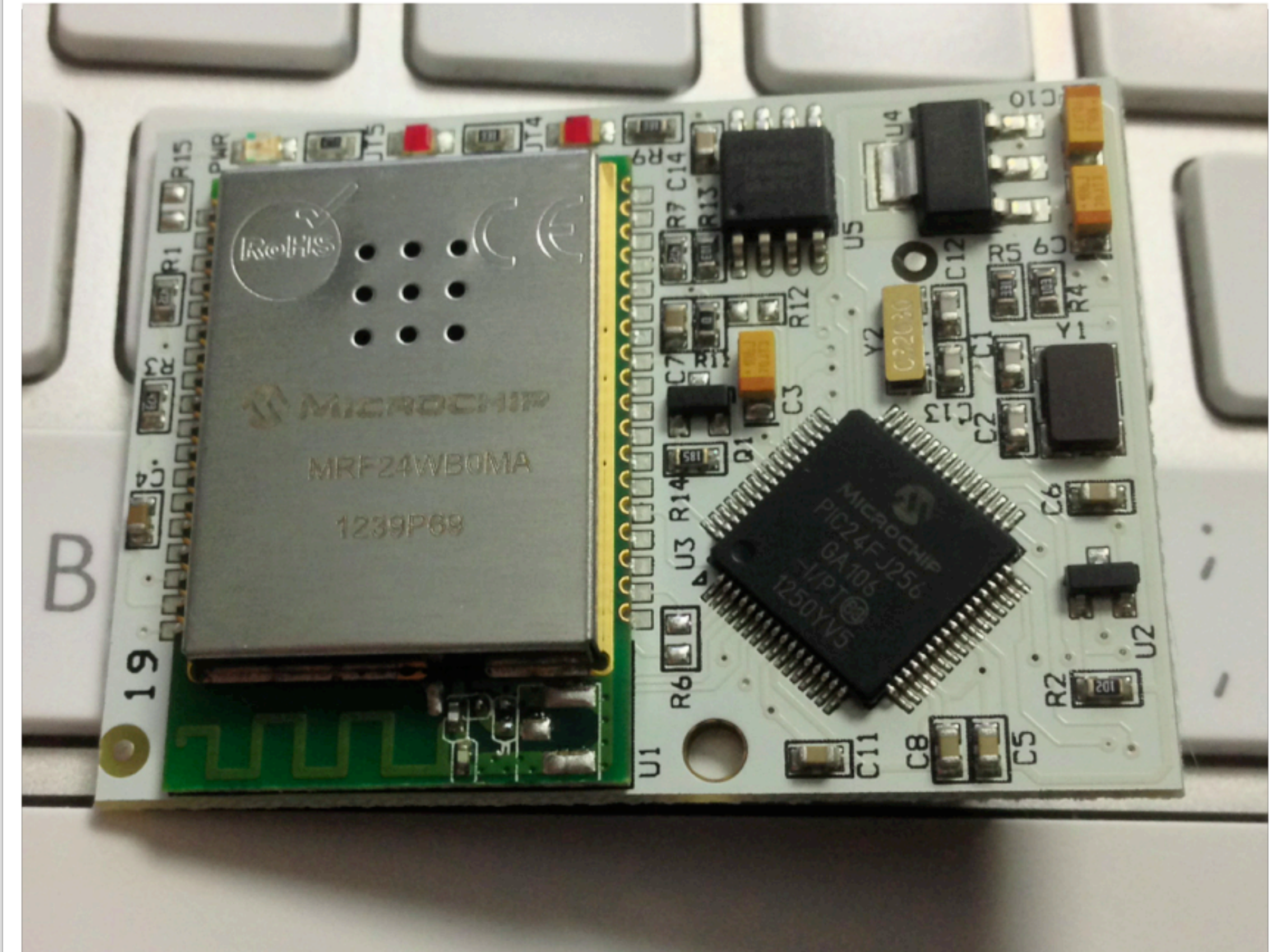
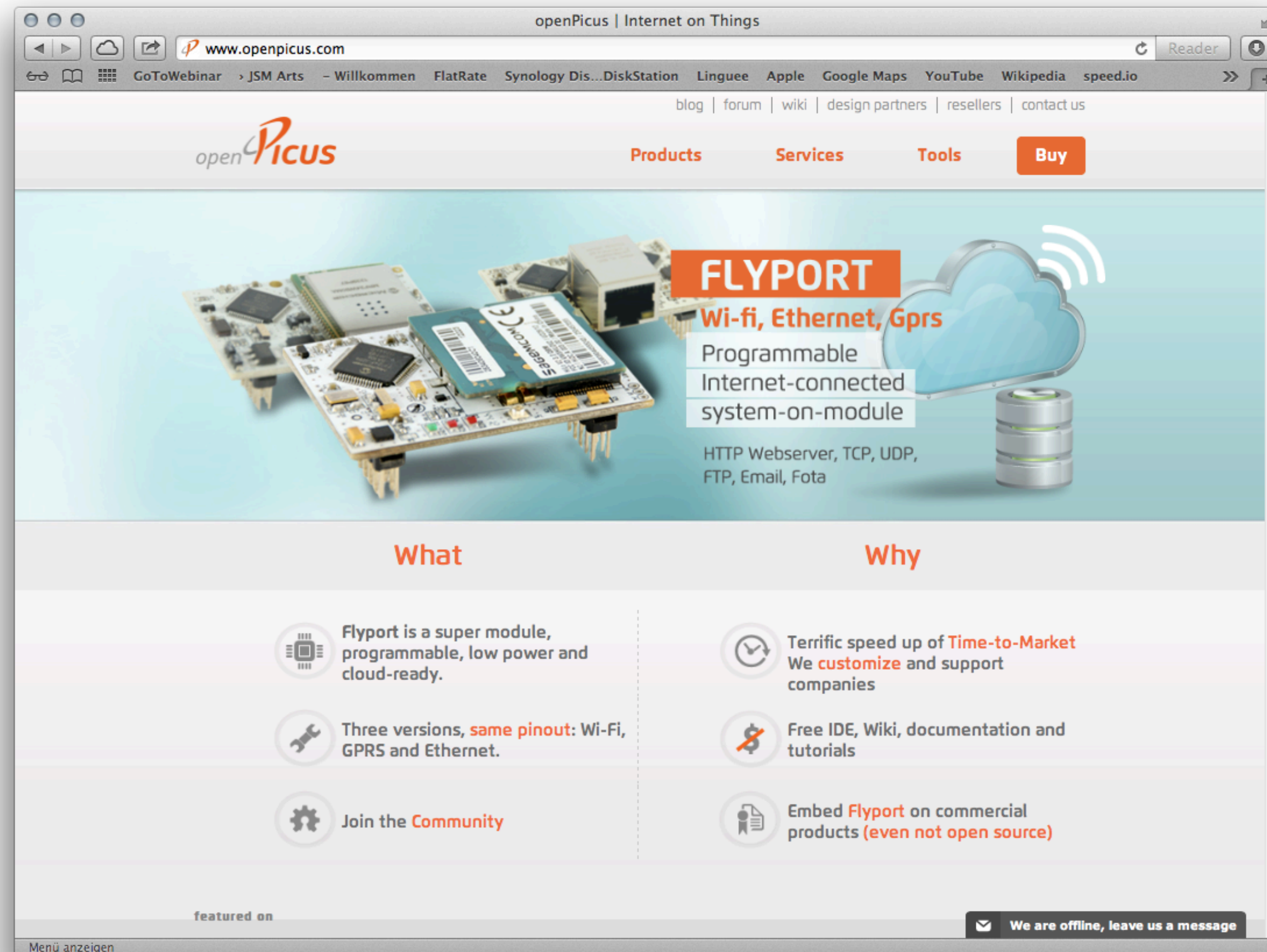
Werbung für Ihre Website
Google-Anzeigen

TechCrunch DISRUPT EUROPE 2013
28-29 October
Arena Berlin
Get Tickets Now and Save €200

Schrödinger lernt ...
HTML5, CSS3 und JavaScript
Galileo Computing

Die Plattform für mobile Entscheider
Kostenloses Besucherticket!
Aktionscode: CW3H6
Klicken und Aktionscode eintragen
6.7. Nov. 2012 COMMUNICATIONSWORLD

Meine Entscheidung:



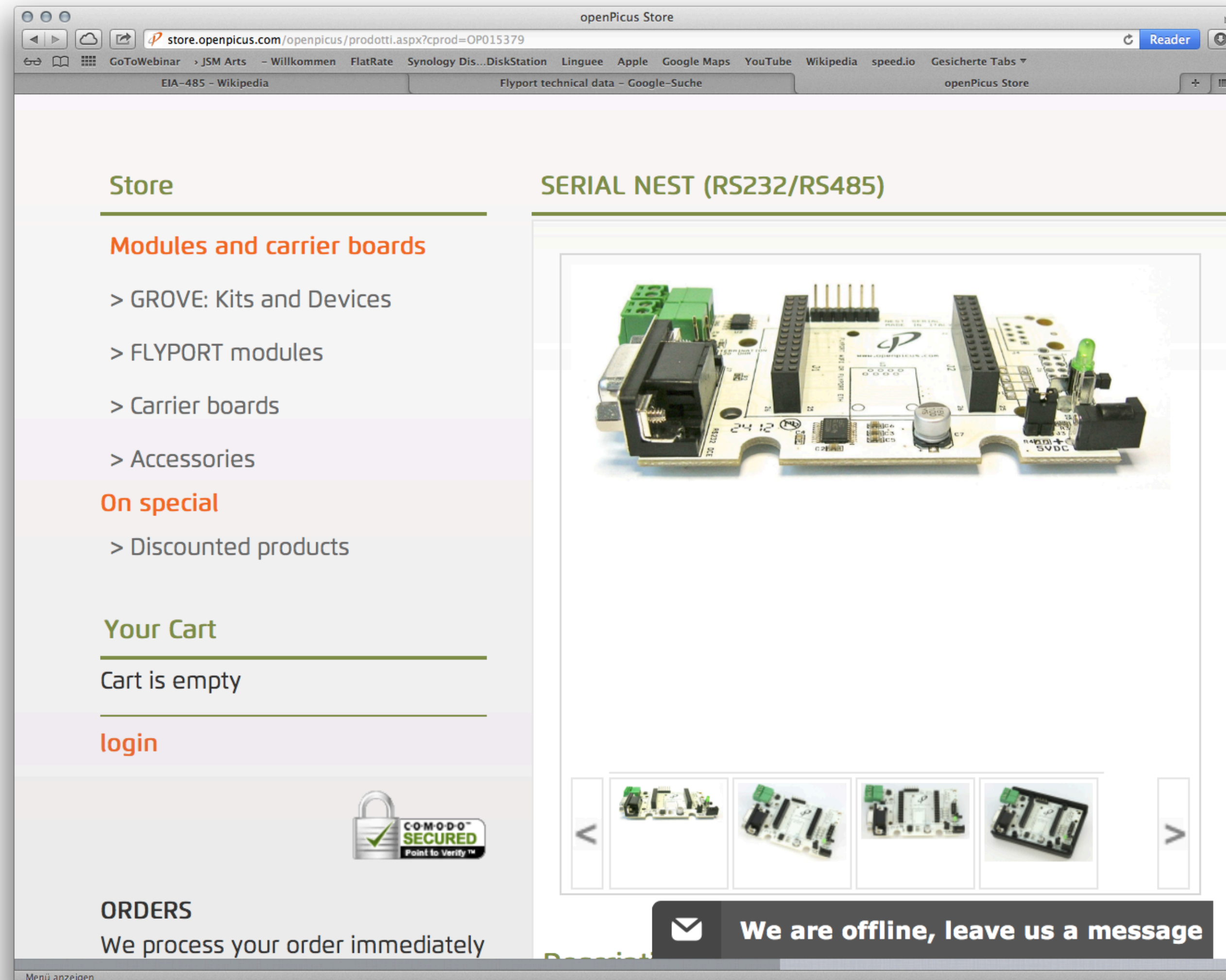
OpenPicus FlyPort

- Einfach
- OpenSource
- Preiswert (29,00 €)
- WiFi
- Grove und andere „Nester“ verfügbar
- 16 Bit, Pic24FJ256, 16K Ram 32MHz
- 802.11 b/g/n WiFi
- 5V oder 3.3V
- 18 remappable I/O's
- 4 UARTs, 2 SPIs, 1 I2C
- 11 Gramm

iPhone via WiFi to FlyPort. Gut. Und dann?

- Eine RS-485 muss her
- Auf neudeutsch ein Serial-Nest :-)

So sieht es aus:



29,00 € :-)

Also, Hardware für 58,00 €

Den Rest muss das Hirn besorgen

- FlyPort spielt HotSpot, dann also vom iPhone via WiFi über Sockets zum FlyPort (natürlich über eine abgesicherte Verbindung)
 - FlyPort setzt um auf RS-485 - RS-485 als Eingang auf Port 4 des Lego nxt Bausteins - aktivieren des High-Speed-Communication Interfaces - Einlesen der Kommandos und diese dann ganz einfach ausführen
- Fertig, und der Roboter macht, was er soll :-)

Lösung Xcode / Objective-C

Will man im Jahr 2013 noch Netzwerk-Kommunikation zu Fuß programmieren?

- Ich nicht :-)
- Für OpenSource Projekte verwende ich GCDAsyncSockets
- <https://github.com/robbiehanson/CocoaAsyncSocket>
- Einfach, übersichtlich, robust, vertrauenswürdig, gut gepflegt, sauber entwickelt, funktional, stabil. Mit einem Wort: „sexy“

Einfach und übersichtlich

```
#define ROBOT_IP_ADDRESS @"192.168.1.115"
#define ROBOT_PORT 9988

@interface RobotCommunication : NSObject

+(instancetype) sharedInstance;

-(void) connect;
-(void) disconnect;
-(BOOL) isConnected;

-(void) sendForward;
-(void) sendBackward;
-(void) sendLeft;
-(void) sendRight;

@end
```

Als Singleton kann jeder mit dem Roboter reden

```
+(instancetype) sharedInstance
{
    static RobotCommunication* sharedInstance;
    static dispatch_once_t onceToken;
    dispatch_once(&onceToken, ^{
        sharedInstance = [[self alloc] init];
    });
    return sharedInstance;
}
```


init

```
-(instancetype) init
{
    if (self = [super init]) {
        dispatch_queue_t queue = dispatch_get_global_queue (DISPATCH_QUEUE_PRIORITY_DEFAULT, 0);
        self.socket = [[GCDAsyncSocket alloc] initWithDelegate:self delegateQueue:queue];
    }
    return self;
}
```

Die socket ist der Dreh- und Angelpunkt

```
@interface RobotCommunication () <GCDAsyncSocketDelegate>
@property GCDAsyncSocket* socket;
@end

@implementation RobotCommunication

-(BOOL)isConnected
{
    return self.socket.isConnected;
}

...
```


Aufbau der Socket-Communication

```
...
-(void) connect
{
    if (!self.isConnected) {
        NSError* error;
        [self.socket connectToHost:ROBOT_IP_ADDRESS onPort:ROBOT_PORT error:&error];
        [error dump];
    }
}

-(void)disconnect
{
    if (self.isConnected) {
        [self.socket disconnect];
    }
}
```

[error dump]; // Hä?

```
@interface NSError (JSM)
-(void) dump;
@end

@implementation NSError (JSM)
-(void)dump
{
    NSLog(@"Error: %@", self);
}
@end
```

Also, connect und disconnect

```
...
-(void) connect
{
    if (!self.isConnected) {
        NSError* error;
        [self.socket connectToHost:ROBOT_IP_ADDRESS onPort:ROBOT_PORT error:&error];
        [error dump];
    }
}

-(void)disconnect
{
    if (self.isConnected) {
        [self.socket disconnect];
    }
}
```

Beim connect werden wir zurückgerufen

```
-(void)socket:(GCDAsyncSocket *)sock didConnectToHost:(NSString *)host port:
(uint16_t)port
{
    NSLog(@"%s", __PRETTY_FUNCTION__);
    NSDictionary* message = @{@"type" : @"init", @"config" : @"none"};

    NSData* messageJSON = [NSJSONSerialization dataWithJSONObject:message options:0
error:nil];

    [self.socket writeData:messageJSON withTimeout:-1 tag:DataWriteTagInitSocket];

    [[NSNotificationCenter defaultCenter] postNotificationName:@"socketDidConnect"
object:self.socket];
}
```

Auch der diconnect meldet sich noch einmal

```
-(void)socketDidDisconnect:(GCDAsyncSocket *)sock withError:(NSError *)err
{
    NSLog(@"%s", __PRETTY_FUNCTION__);
    [[NSNotificationCenter defaultCenter] postNotificationName:@"socketDidDisconnect"
object:self.socket];
}
```


Vorwärts.....

```
-(void) sendForward
{
    if (!self.isConnected) return;

    NSDictionary* message = @{@"type" : @"direction", @"direction" : @"forward"};

    NSData* messageJSON = [NSJSONSerialization dataWithJSONObject:message options:0
error:nil];

    [self.socket writeData:messageJSON withTimeout:-1 tag:DataWriteTagForward];
}
```

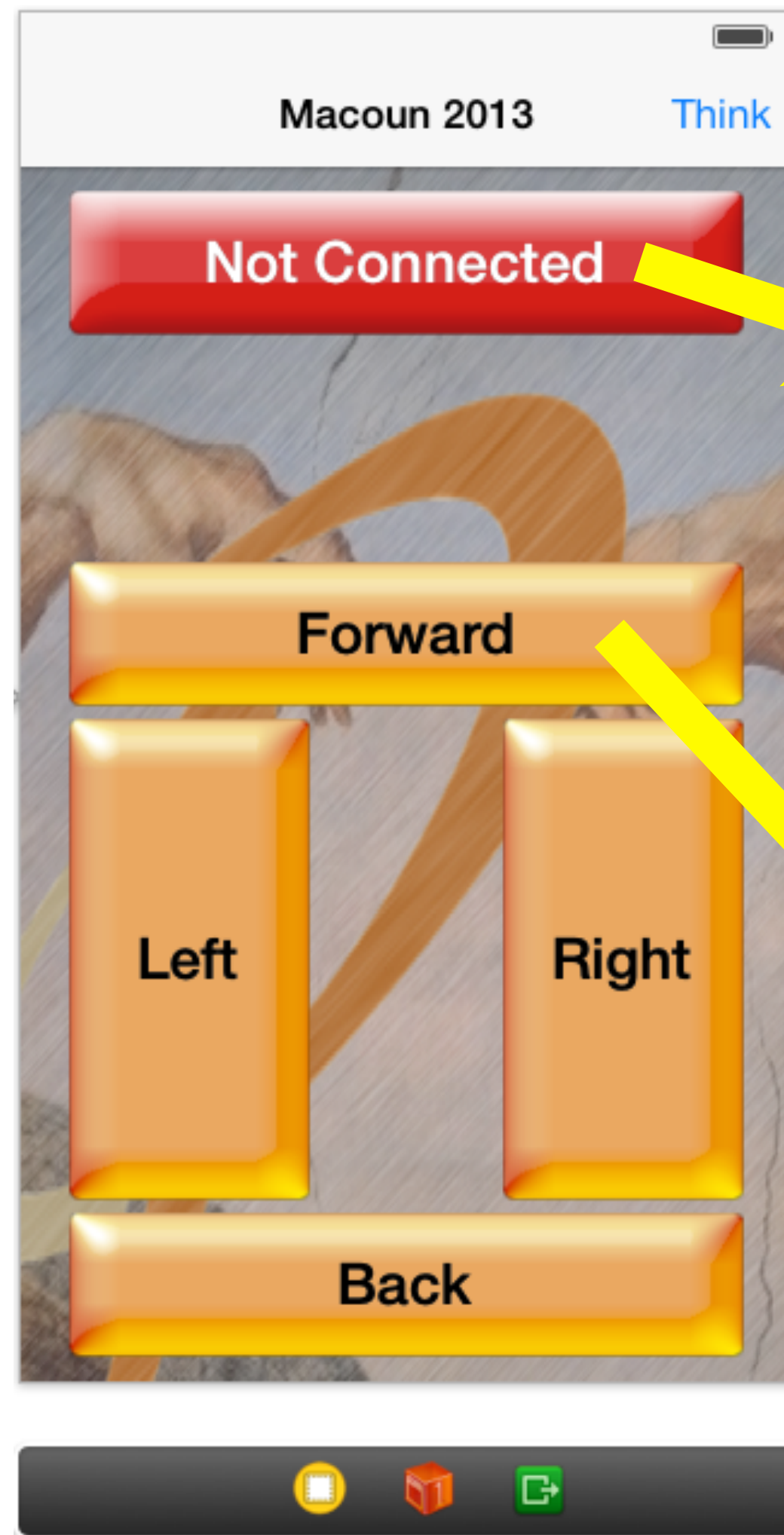
Jedem Kommando seinen Tag

```
// Wir sind höflich und geben jedem Send einen Tag mit auf den Weg
enum {
    DataWriteTagInitSocket,
    DataWriteTagBackward,
    DataWriteTagLeft,
    DataWriteTagRight,
    DataWriteTagForward,
};
typedef long DataWriteTag;
```

Wenn etwas geschrieben wurde sagt man uns bescheid

```
-(void)socket:(GCDAsyncSocket *)sock didWriteDataWithTag:(long)tag
{
    NSLog(@"didWriteDataWithTag %ld", tag);
}
```


Die GUI ist dann schon fast geschenkt



```
- (IBAction)connectionButtonTouchUpInside:(id)sender
{
    self.connectionButtonOutlet.enabled = NO;
    if (![RobotCommunication sharedInstance].isConnected) {
        [[RobotCommunication sharedInstance] connect];
    } else {
        [[RobotCommunication sharedInstance] disconnect];
    }
}
```

```
- (IBAction)forwardButtonTouchUpInside:(id)sender
{
    [[RobotCommunication sharedInstance] sendForward];
}
```

Ich mag Observer mit Blöcken

```
[[NSNotificationCenter defaultCenter] addObserverForName:@"socketDidConnect"
object:nil queue:nil usingBlock:^(NSNotification *note)
{
    dispatch_async(dispatch_get_main_queue(), ^{
        self.connectionButtonOutlet.selected = YES;
        self.connectionButtonOutlet.enabled = YES;
        [self enableDirectionButtons:YES];
    });
}];
```

Demo

Lösung OpenPicus IDE

leider unter Windows :-(

Bevor wir zu Windows
wechseln...

...an dieser Stelle einen herzlichen
Dank von mir an Euch :-)

„Hallo und herzlich willkommen liebe
Freunde der Programmierung von iPad,
iPhone & Co. ...“

Downloads pro Datei, pro Tag

Zeigt an, wie oft die Dateien in den letzten Tagen heruntergeladen wurden.

	Freitag, 04.10.13	Donnerstag, 03.10.13	Mittwoch, 02.10.13	Dienstag, 01.10.13	Montag, 30.09.13	Sonntag, 29.09.13	Samstag, 28.09.13
me	49	315	424	553	379	469	503
ure-Talk-023 (06.08.2013) 06.09.13	21	75	103	82	59	107	86
lk-022 (23.08.2013) 23.08.13	4	25	21	31	31	30	
lk-021 (02.08.2013) 02.08.13	3	20	12	15	10	22	
IL-Parser.mp4	3	13	4	14	9	15	
rcodesanner.mp4	1	9	8	14	7	10	
yValueCoding-Werbung-	3	8	9	12	7	8	
KeyValueObserving.mp4	1	8	13	11	5	7	
nPinchUndRotate.mp4	0	9	12	12	5	8	
xtFieldUndDatePicker.mp4	0	7	13	13	6	8	
rRettendeStrohalm.mp4	0	8	8	12	8	11	
coun-2012-CollectionViews.mp4	0	7	13	13	6	9	
UnwiderstehlicheApp.mp4	0	5	9	10	6	7	
pc034-MailComposer.mp4	0	4	11	11	5	5	
edicate.mp4	0	5	11	9	5	5	
O.mp4	1	3	3	9	6	7	
dioPlayer.mp4	0	4	4	9	6	5	
oCoder.mp4	0	4	5	8	6	6	
tureBrowser vierter Teil.mp4	0	3	3	8	6	6	6
pc026-PictureBrowser dritter Teil.mp4	0	4	4	7	5	5	6
pc027-PictureBrowser zweiter Teil.mp4	0	2	6	7	5	5	7
pc026-PictureBrowser erster Teil.mp4	0	3	6	8	7	6	9
pc025-SettingsBundle.mp4							

04.10.13
49

03.10.13
315

02.10.13
425

01.10.13
553

30.09.13
379

29.09.13
469

28.09.13
503

herzlichen DANK dafür

Zum Thema: OpenPicus IDE

```
#include "taskFlyport.h"
#include "RS485Helper.h"
#include "RS232Helper.h"
#include "cJSON.h"

#define DE_485    p2
#define RE_485    p17
#define TX_485    p5
#define RX_485    p7

#define TX_232    p4
#define RX_232    p6
#define CTS_232   p11
#define RTS_232   p9

const int port485 = 2;
const int port232 = 3;
```

Wie schön, nur ein Task

```
void FlyportTask()
{
    // Pins mappen, damit sich das Serail Nest auch richtig wohl fühlt
    IOInit(p1, out);      // pin 3
    IOInit(p3, out);      // pin 4
    IOInit(p8, out);      // pin 5
    IOInit(p10, out);     // pin 6
    IOPut(p1, off);
    IOPut(p3, off);
    IOPut(p8, off);
    IOPut(p10, off);
    IOInit(p12, indown);  // pin 7
    IOInit(p14, indown);  // pin 8
    ...
}
```

Initialisieren der RS-485

Leute, da werden Erinnerungen wach, oder?

```
// Initialize RS485
RS485Off(port485);

RS485Remap(port485, TX_485, RX_485, DE_485, RE_485);
RS485Init(port485, 19200);
RS485SetParam(port485, RS485_STOP_BITS, RS485_ONE_STOP);
RS485SetParam(port485, RS485_DATA_PARITY, RS485_8BITS_PARITY_NONE);

RS485On(port485);
```

Aufbau der Socket „zu Fuß“

```
// Verbindungsaufbau
if (serverSocket == INVALID_SOCKET) {

    // ServerSocket auf Port 9988 anbieten
    UARTWrite(1, "setup serverSocket...\r\n");
    serverSocket = TCPServerOpen("9988");

    // Warten, bis alles ok ist
    UARTWrite(1, "setup serverSocket waiting...\r\n");
    while (serverSocket == INVALID_SOCKET) vTaskDelay (1);

    UARTWrite(1, "serverSocket ready...\r\n");
```

...

Socket Verbunden? Dann lesen wir mal...

```
} else {  
  
    int rxlen = TCPRxLen (serverSocket);  
    if (rxlen > 0 && rxlen < 255) {  
        char readBuffer[256];  
        TCPRead (serverSocket, readBuffer, rxlen);  
  
        cJSON* json = cJSON_Parse (readBuffer);
```

`cJSON* json = cJSON_Parse (readBuffer);`

16K Ram hin oder her, wir gönnen uns den Luxus!

<https://github.com/kbranigan/cJSON>

JSON macht es deutlich einfacher

```
char* type = cJSON_GetObjectItem (json, "type")->valuestring;
if (!strcmp (type, "direction")) {
    char* direction = cJSON_GetObjectItem (json, "direction")->valuestring;
    sprintf (msgBuffer, "%c", direction[0]);
    RS485Write (port485, msgBuffer);          // Hier geht's dann schon zum Robot
}
```

// Zur Erinnerung: So senden wir aus Objective-C heraus

```
-(void) sendForward
{
    NSDictionary* message = @{@"type" : @"direction", @"direction" : @"forward"};
    NSData* messageJSON = [NSJSONSerialization dataWithJSONObject:message options:0 error:nil];
    [self.socket writeData:messageJSON withTimeout:-1 tag:DataWriteTagForward];
}
```

Demo

(Nicht erschrecken, jetzt kommt Windows)

Lösung RobotC / C

RobotC


- Nicht kostenfrei, jedoch bezahlbar:
49,- USD für 365 Tage / 79,- USD für immer
- In amerikanischen Klassenzimmern sehr beliebt
- „Echtes“ C, String-Library, Collections, nett gemacht
- Tolle Hilfen, viele Beispiele, große Community

RobotC

ROBOTC.net :: Download ROBOTC for LEGO MINDSTORMS. Start programming your NXT robotics projects now!

www.robotc.net/download/nxt/Reader

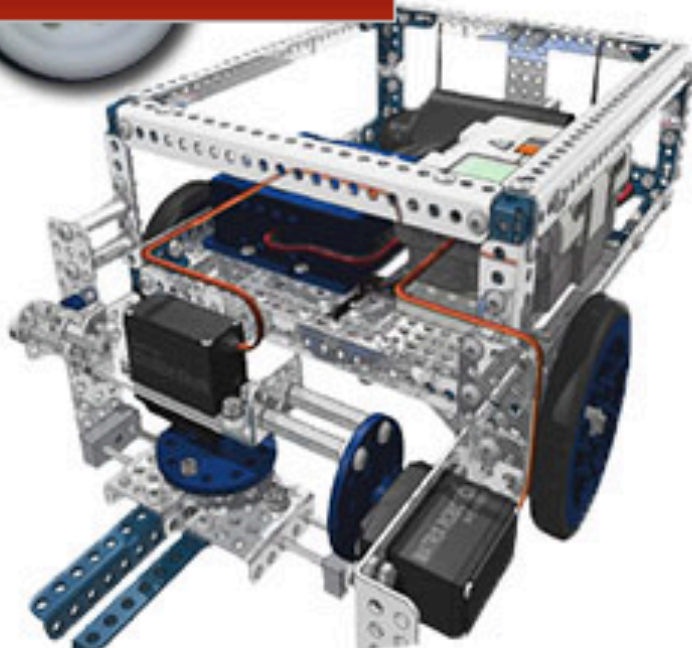

GoToWebinar JSM Arts - Willkommen FlatRate Synology Dis...DiskStation Linguee Apple Google Maps YouTube Wikipedia speed.io Gesicherte Tabs

 **ROBOTC**






Download Support Education Community Purchase Search... Go

ROBOTC for MINDSTORMS

Platforms: NXT, TETRIX, MATRIX
Version: 3.62
Date Posted: Sept 6, 2013



- About ROBOTC
- ROBOTC 3.0 for MINDSTORMS
- ROBOTC 3.0 for CORTEX & PIC
- ROBOTC 4.0 for VEX Robotics - VEX IQ
- ROBOTC 3.0 for Arduino
- Robot Virtual Worlds
- ROBOTC 2.0 for RCX
- ROBOTC 2.0 for IFI VEX



Free 30-day Full Trial!

Note: You do not need to deactivate your license in order to upgrade for ROBOTC versions 3.0 and above

Download version 3.62
(includes free trial)

ROBOTC 3.0 requires a ROBOTC 3.0 license. If you have a ROBOTC 2.0 license (or older), you can download ROBOTC 2.0 in the "Optional Downloads" section below.

Already have a building license?
[How to install the Building License file »](#)

Note: The NXT LEGO USB Device Driver is included in the ROBOTC installer.

Menü für „http://www.robotc.net/download“ anzeigen

High-Speed initialisieren

```
nxtEnableHSPort ();  
  
nxtSetHSBaudRate (19200);  
  
nxtHS_Mode = hsRawMode;
```

Wir warten auf ein Zeichen

```
char direction;

// so lange, bis die orange Taste gedrückt wird
while (nNxtButtonPressed != kEnterButton) {

    // warten bis zeichen an der RS485 anliegen
    while (!nxtGetAvailHSBytes() && nNxtButtonPressed != kEnterButton){
        wait10Msec (1);
    }

    // Byte in direction lesen
    nxtReadRawHS(direction, 1);

    ...
}
```


Ein Byte gibt die Richtung an :-)

```
switch (direction) {  
  case 'f':  
    motor[motorC] = 50; // So einfach ist das in RobotC  
    motor[motorB] = 50; // Alle „Lego-Ports“ haben eine Bezeichnung  
    wait1Msec(2000);  
    motor[motorC] = 0;  
    motor[motorB] = 0;  
    break;  
  
  case 'b':  
    motor[motorC] = -50;  
    motor[motorB] = -50;  
    wait1Msec(2000);  
    motor[motorC] = 0;  
    motor[motorB] = 0;  
    break;  
}
```

...

Demo

(Nicht erschrecken, jetzt kommt schon wieder Windows)

Und so kommt alles zusammen

Hardware RS-485

- Pin-Belegung am sagenhaften Port 4 vom Lego-Brick nxt 2.0

Pin 1 - weiß - analog in (0-5V) oder Spannungsversorgung für Sensoren

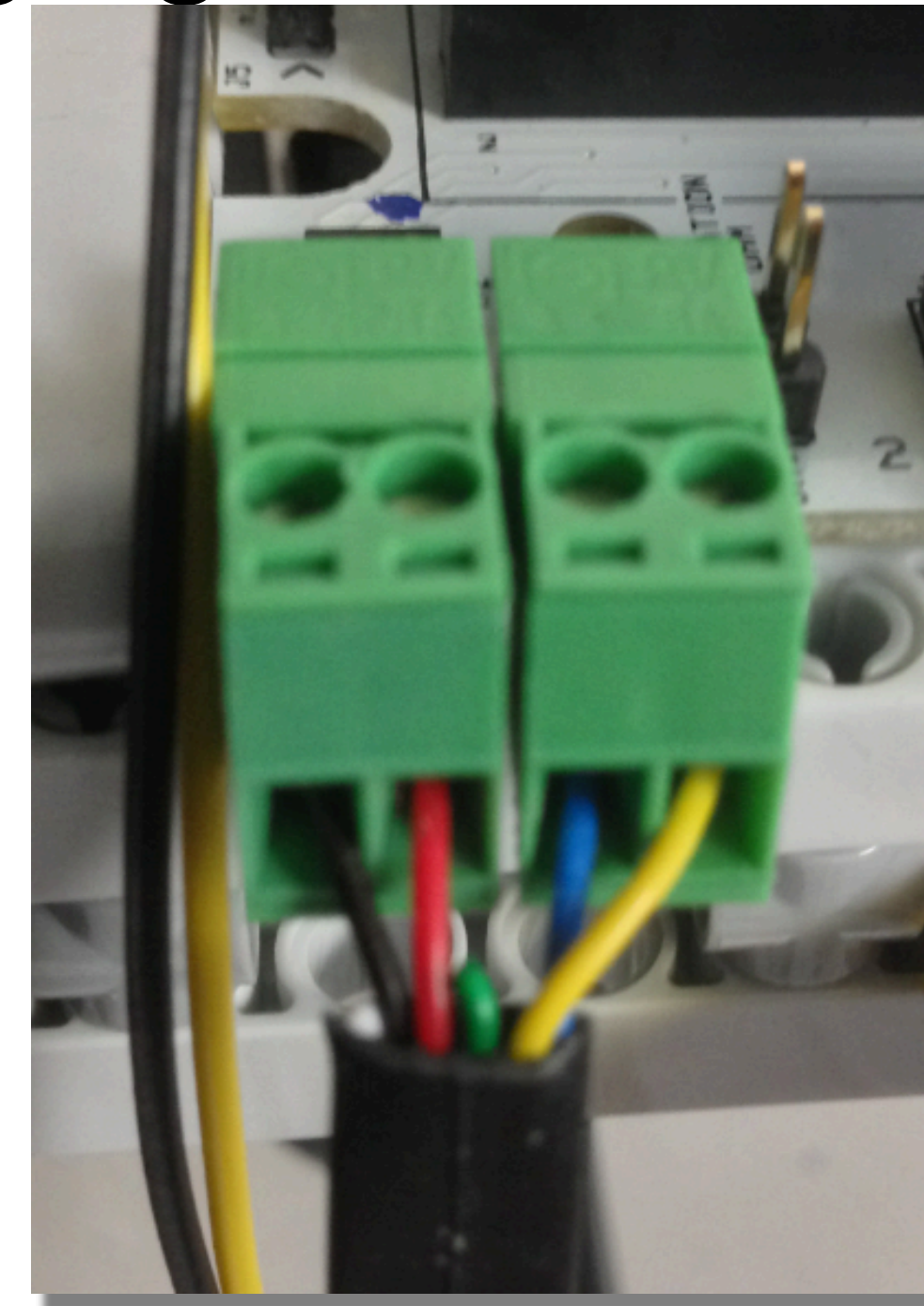
Pin 2 - schwarz - Masse

Pin 3 - rot - Masse

Pin 4 - grün - 4,3 V

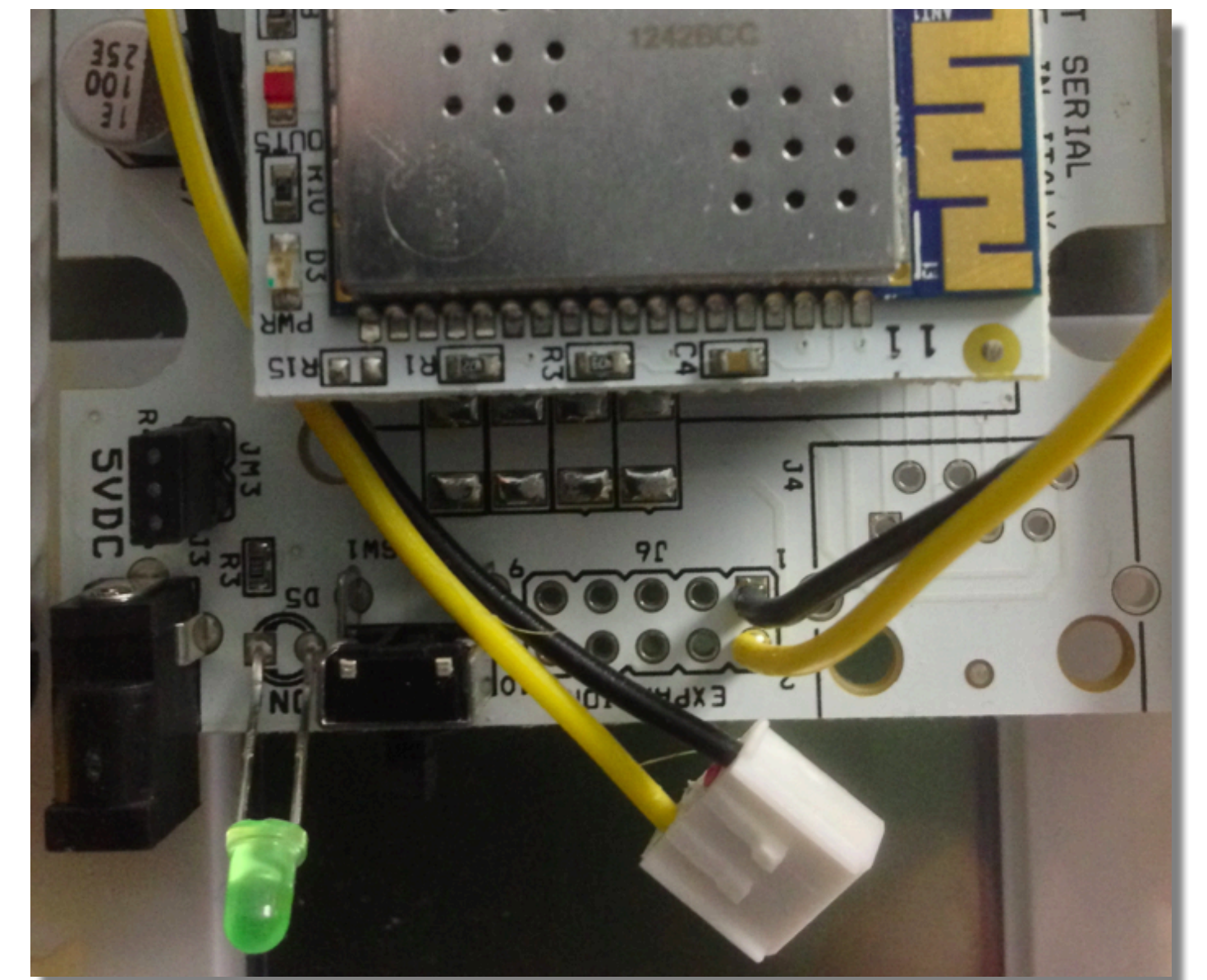
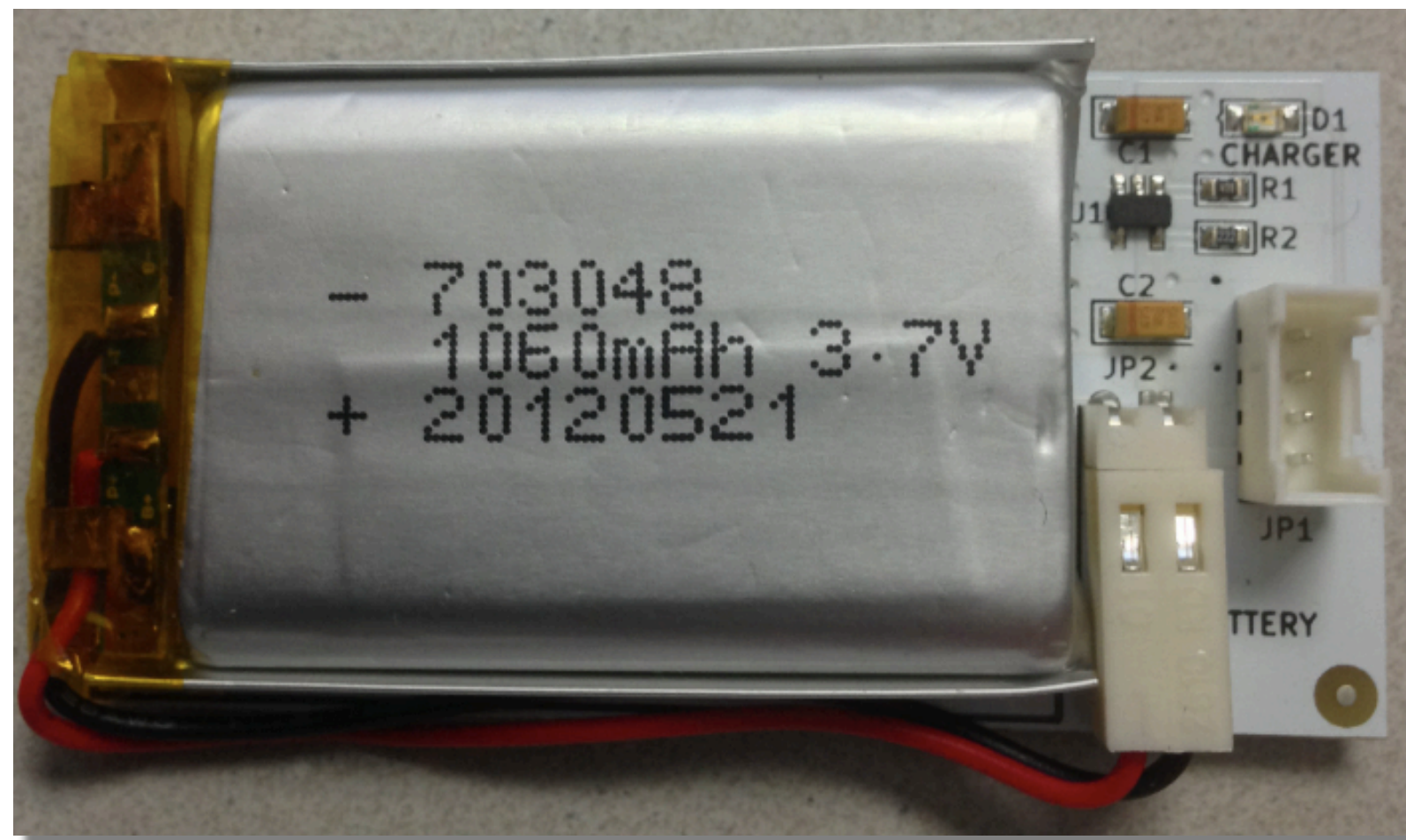
Pin 5 - gelb - I2C, SCL, oder RS-485-A

Pin 6 - blau - I2C, SDA, oder RS-485-B



Noch was schickes...

- Der Roboter am langen Kabel auf der Bühne wäre ja schrecklich
- „Autonome“ Stromversorgung ist angesagt
- Noch einmal 9,98 € investiert und das hier erworben:



„Ich will den Quelltext...“

Der steht in 15 Minuten auf

<http://jsmarts.de>

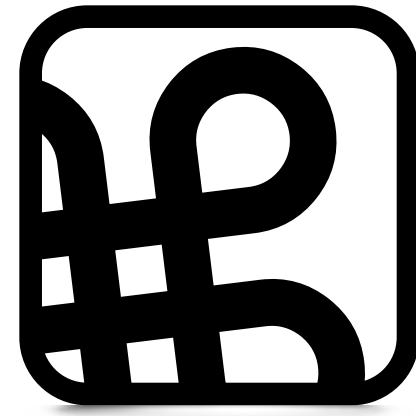
zur Verfügung :-)

Na, dann wollen wir das Ding
auch mal im Einsatz sehen :-)

Fragen?

Ich wünsche Euch zwei
spannende Tage hier auf der
Macoun 2013

herzlichst
Euer Franky



Macoun