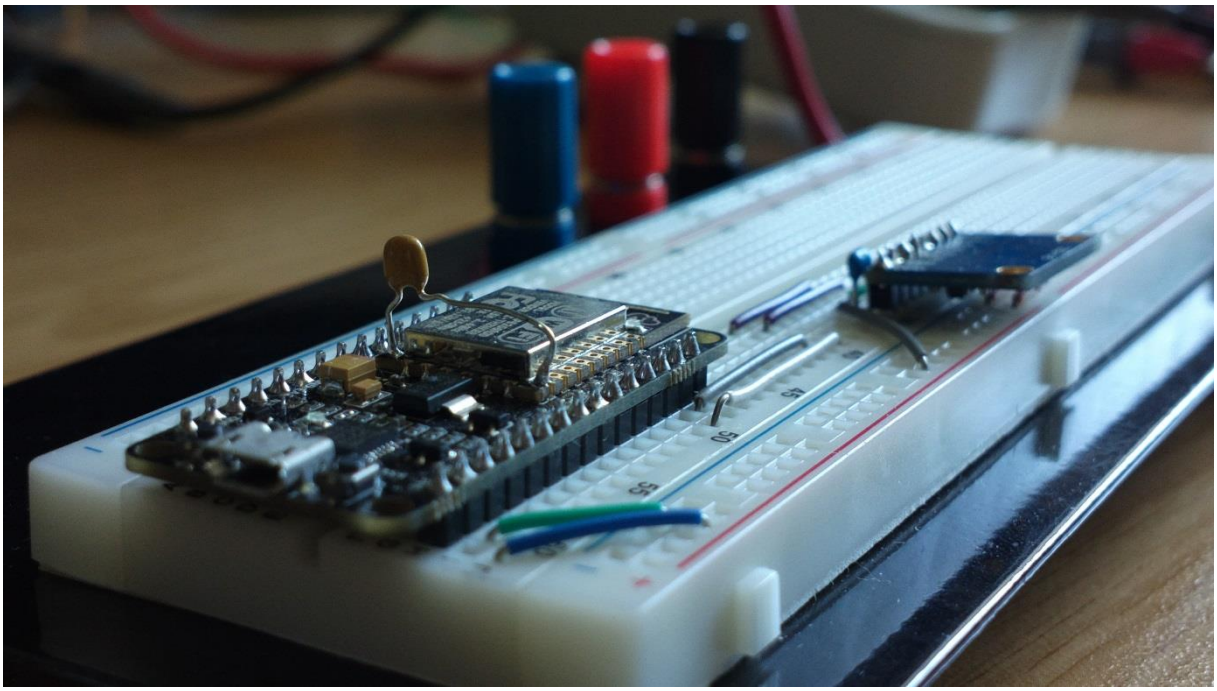
Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Beschleunigungsmessung im American Football Helm

Gruppennummer 14



Informationstechnik Labor Sommersemester 2016

Prof. J. Walter

Gruppenmitglieder:

Kay Bouché	*42892*
Benedikt Schumann	*41296*

Inhalt

Problemstellung

American Football ist ein Kontaktsport bei dem hohe Kräfte auftreten. Kürzlich hat die höchste American Football Liga in den USA erstmals zugegeben, dass das Risiko für Gehirnverletzungen durch langjähriges Ausüben des Sports drastisch erhöht wird. Der Kopf wird beim American Football eigentlich durch einen Helm geschützt. Häufig entstehen trotzdem Gehirnerschütterungen beim direkten Aufprall mit dem Gegenspieler. Die Thematik ist hoch aktuell. 2016 kam der Film „Concussion“, der genau diese Problemstellung aufgreift, in die Kinos. Wir wollen in unserer Laborstudie die auftretenden Beschleunigungen im Footballhelm erfassen und dem Spieler oder Trainer die Werte zur Verfügung stellen. Ziel ist es, bei einem schweren Aufprall die Gefahr für eine Gehirnerschütterung zu bestimmen. Spieler und Trainer können mit Hilfe der Messwerte proaktiv reagieren.

Stand der Technik

„Schockbox Sports Helmet Sensors“ bietet bereits eine Sensorik für Footballhelme an. Die Sensorik übermittelt die auftretenden Kräfte in Echtzeit an eine Smartphone-App. Die Signalübertragung erfolgt per Bluetooth. Leider werden Footballhelme bisher nur sehr selten durch eine derartige Sensorik erweitert, hauptsächlich im professionellen Umfeld. Wir wünschen uns, dass Hobbyisten unserer DIY-Anleitung folgen und eine kostengünstige Beschleunigungsmessung für ihren Footballhelm nachbauen.

Aufgabenstellung

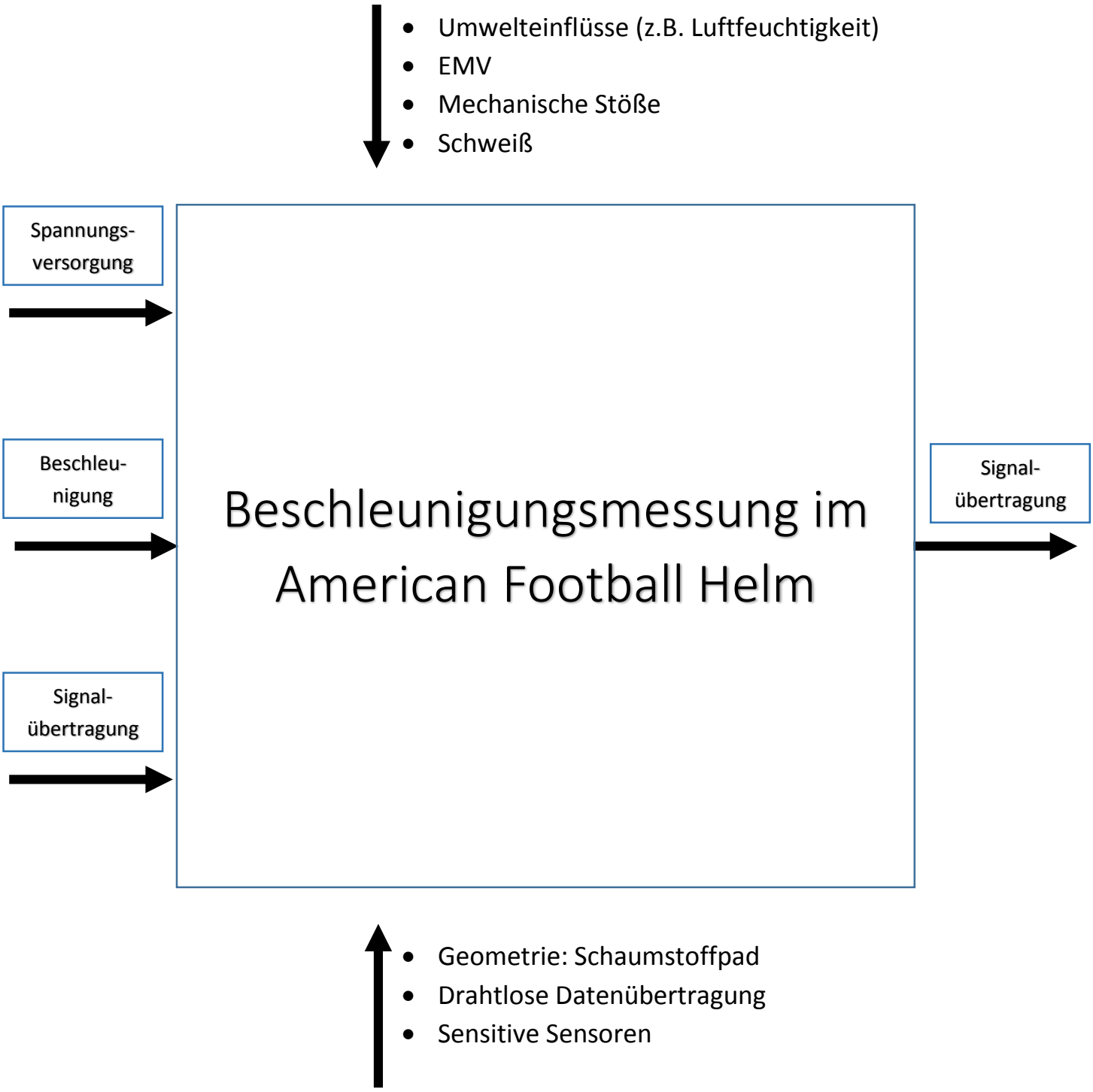
Wie bereits beschrieben, haben wir uns als Aufgabe gesteckt, eine Beschleunigungsmessung für den Footballhelm zu entwickeln.
Durch einen Achsenbewegungssensor, den MPU-6050, sollen die Beschleunigungen im Helm ermittelt werden. Die Messungen sollen lokal verarbeitet und drahtlos mittels WLAN übertragen werden. Die Signalverarbeitung und Signalübertragung erfolgt durch das Entwicklungsboard NodeMCUv2.
Das Hauptaugenmerk unserer Laborstudie wird auf die Erfassung, Übertragung und anschauliche Visualisierung von harten Aufprällen gelegt. Entscheidend ist dabei neben der Übertragung in die Cloud und Visualisierung mittels HTML-Skript, die Messwerte mit ausreichend hoher Auflösung aufzunehmen und einen einzelnen Stoß, der eine ganze Reihe von erhöhten Beschleunigung mit mehreren Peaks produziert, zuverlässig zu identifizieren und den tatsächlich Maximalwert zu finden.
Das Einbauen der Elektronik in den Helm, bzw. in die Schaumstoffpads des Helmes, ist nicht Bestandteil unseres Projektes. Beim Einbau muss berücksichtigt werden, dass hohe Kräfte beim Sport auftreten und der Schweiß des Spielers Probleme machen könnten. Der Schutz des Kopfes darf durch den Einbau in den Helm nicht verschlechtert werden. Zusätzlich werden ein platzsparender Akku und eine Ladeschaltung für den Einbau der Elektronik in den Helm benötigt.

Anforderungsliste

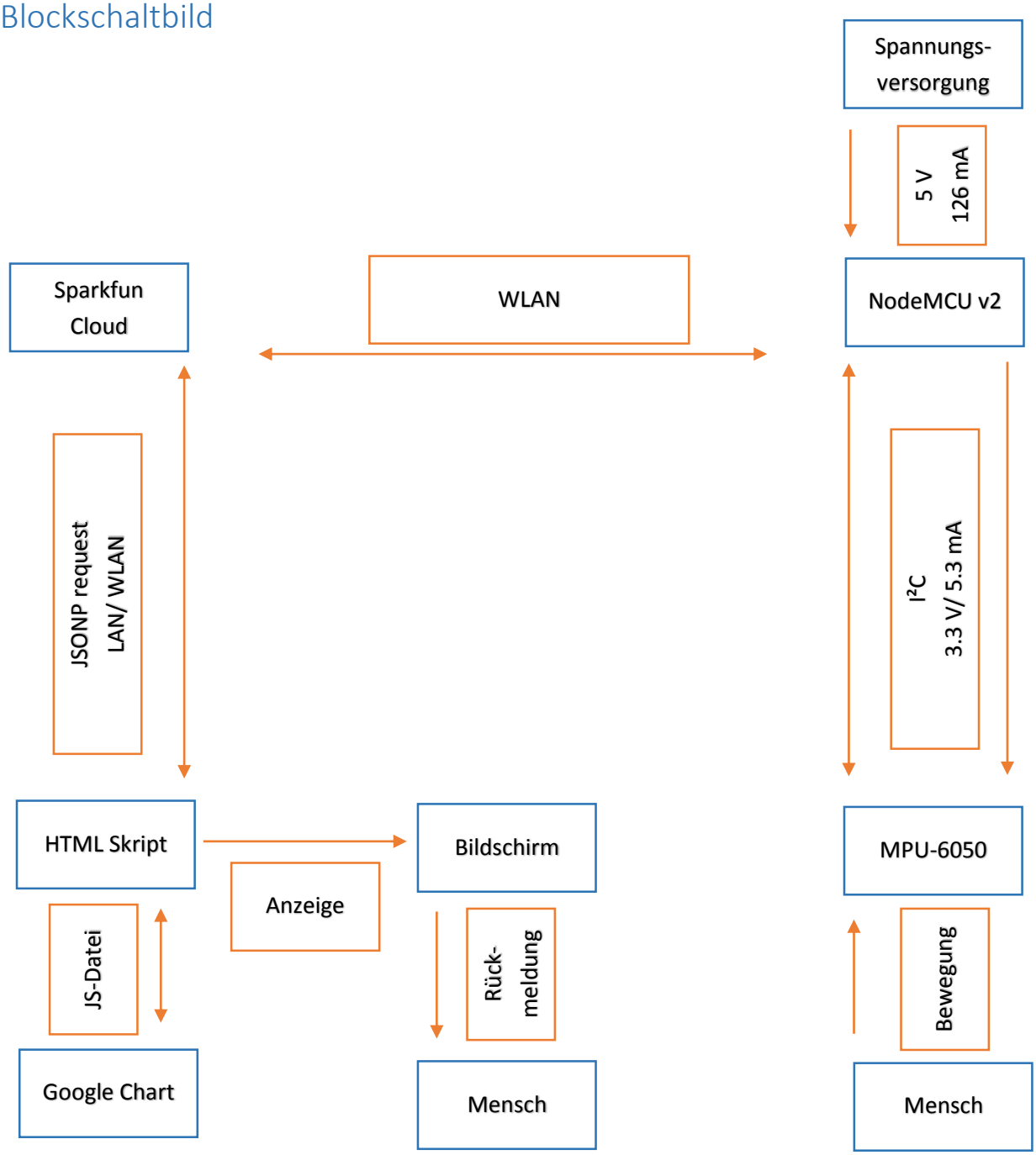
Informationstechnik Labor Fakultät MMT Hochschule Karlsruhe Technik und Wirtschaft	**Anforderungsliste** für **eine Beschleunigungsmessung im American Football Helm**	**MTB732** Sommersemester 2016
Art	**Anforderung**	**Wert/Daten**
Datenübertragung		
J/N	Drahtlose Datenübertragung zwischen µC und Computer via WLAN	
J/N	Datenübertragung zwischen Sensor und µC-Board via I²C	
Beschleunigungsmessung		
J/N	Verwendung des MPU-6050- Beschleunigungssensor	
W	Errechnung der Kräfte aus erkannter Beschleunigung	
Elektronik		
J/N	Modulare Bauweise: Trennung von Datenverarbeitung und Sensorik	
J/N	Verwendung des NodeMCU v2 µC-Board	
W	Spannungsversorgung über Akku	Verbrauch: 5 V, 130 mA
W	Geringer Energieverbrauch bzw. hohe Akkulaufzeit	20 - 40 Betriebsstunden
Mechanik		
W	Gesamte Konstruktion einbaukompatibel in den Footballhelm	
W	Dose oder Gehäuse aus Kunststoff	
W	Einbau in das Schaumstoffpad des Football Helmes	
W	Maximale Abmessungen des Aufbaus	Kleiner als Schaumstoffpad

Software		
J/N	Visuelles Feedback bei hohen Kraftmessungen via HTML (inkl. max. zulässigen Wert)	
J/N	Programmierung des µC in C++ mit der Arduino Entwicklungsumgebung	
Restriktionen		
F	Materialkosten	< 50€
F	Entwicklungsende	06.05.2016

Blackbox


Umwelteinflüsse (z.B. Luftfeuchtigkeit)
EMV
Mechanische Stöße
Schweiß
Spannungs-versorgung
Beschleu-nigung
Signal-übertragung
Beschleunigungsmessung im American Football Helm
Signal-übertragung
Geometrie: Schaumstoffpad
Drahtlose Datenübertragung
Sensitive Sensoren

Blockschaltbild



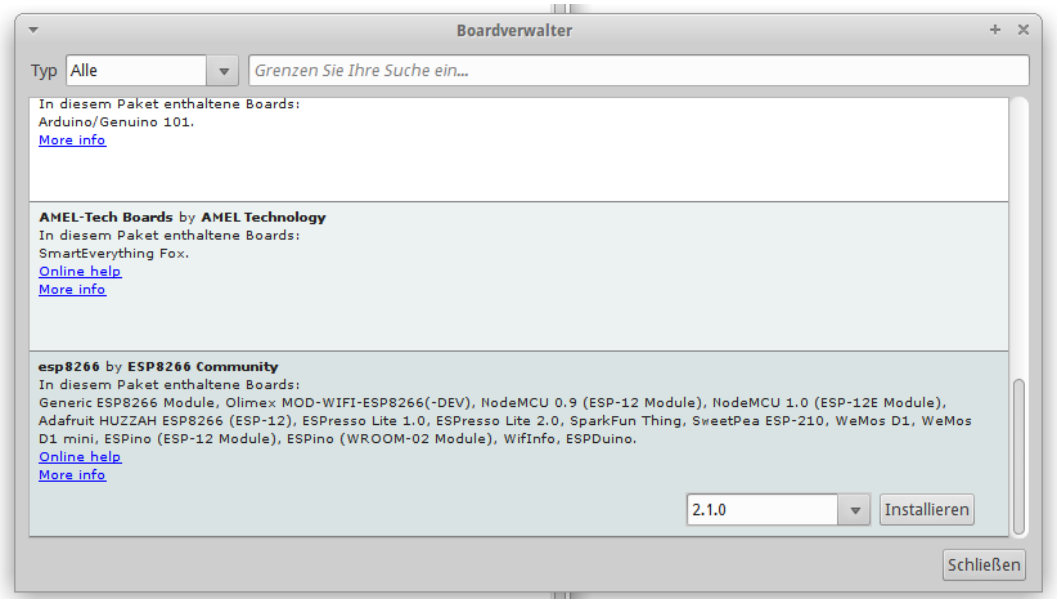
Inbetriebnahme NodeMCUv2

Wir haben uns dafür entschieden, das Projekt mit der Arduino-Entwicklungsumgebung zu realisieren. Entscheidend ist dabei das Einbinden einer passenden Bibliothek, sodass das NodeMCUv2 Entwicklungsboard von der IDE unterstützt wird.

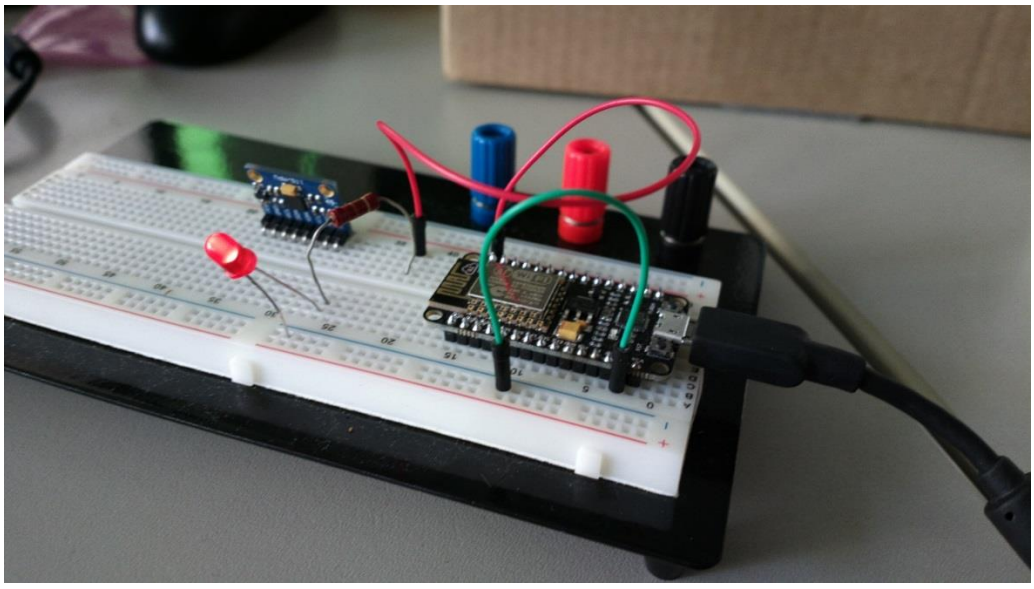
Die Bibliotheken sind unter folgendem Link zu finden:

- https://github.com/adafruit/ESP8266-Arduino

In der README.md Datei in der ersten Verzeichnisebene sind alle zur Installation notwendigen Schritte ausführlich beschrieben.



Folgendes Bild zeigt die erfolgreiche Inbetriebnahme des Boards anhand eines Blinktest mit einer LED:



Die getesteten Arduino-Einstellungen lauten wie folgt:

- Board: NoceMCU 1.0 (ESP-12E Module)
- CPU Frequency: 80Mhz
- Flash Size: 4M (1M SPIFFS)
- Upload Speed: 115200
- Programmer: AVRISP mkII

Für die Kommunikation haben wir die Wire.h (I^2C/TWI) und die ESP8266WiFi.h (WLAN) Bibliotheken eingebunden.

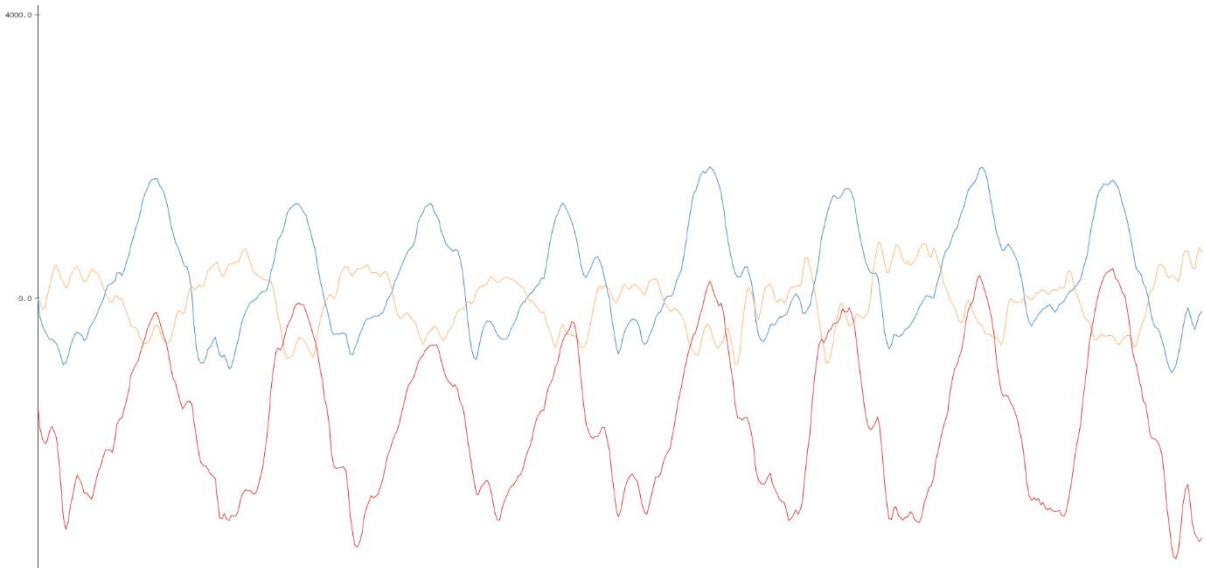
Anmerkung:

- Wie oben zu sehen, hat ein einfaches Testprogramm zum Blinken einer LED funktioniert. Beim Versuch, die WLAN Funktionen des ESP8266 zu nutzen, sind wir mit dieser Konfiguration jedoch gescheitert. Es kam zu ständigen WDT Resets (alle 5s), die das Board unbrauchbar machten. Die Ursache dafür war offensichtlich ein Einbrechen der Versorgungsspannung (erhöhte Anforderungen an Stromversorgung durch WLAN) am auf das NodeMCU v2 Board aufgelöteten ESP8266 aufgrund mangelhafter Konzeption der Zuleitungen bzw. Stützkondensatoren, denn ein 100nF Kondensator zusätzlich am Versorgungspin des ESP8266 angelötet schuf Abhilfe.

Auswertung der Messungen

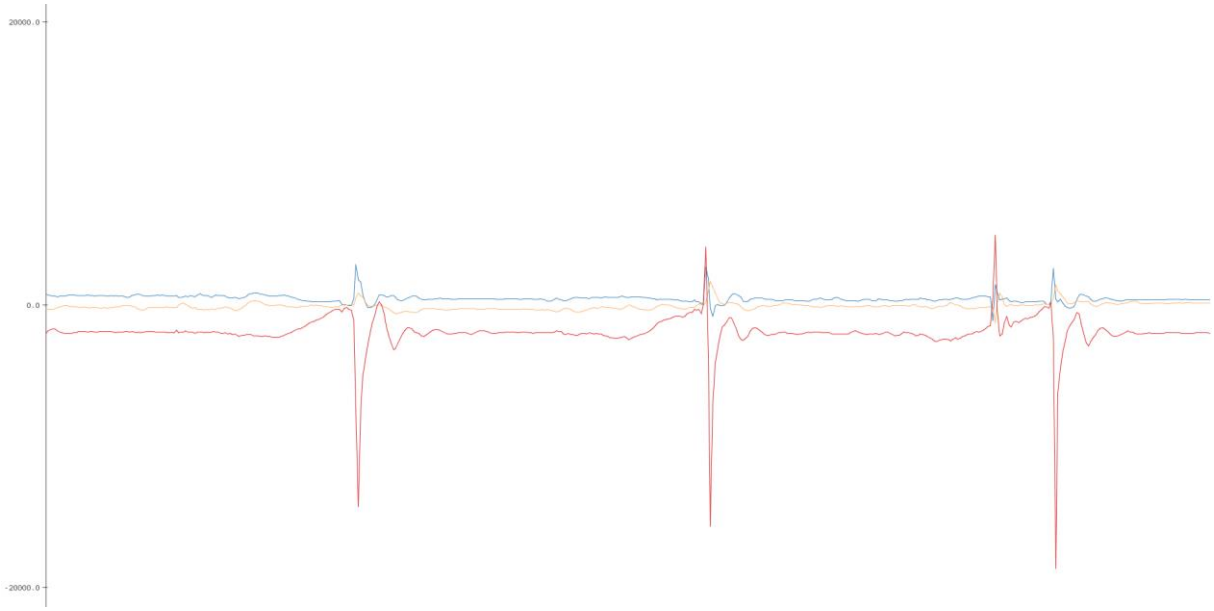
Die Messwerte des MPU werden in der MCU ausgewertet. Die MCU bildet den Betrag aller Beschleunigungen und erstellt somit eine absolute Beschleunigung aus den drei Beschleunigungsvektoren. Die ermittelte absolute Beschleunigung wird in einer float-Variable abgelegt. Für unser Projekt sind geringe Beschleunigungen nicht relevant, deshalb senden wir lediglich Messungen größer 2.5g. Diese Messungen werden wiederum in Messungen größer 8g und kleiner 8g aufgeteilt. Dies bietet Vorteile bei der Visualisierung von bedrohlichen und ungefährlichen Aufprällen. Da ein Aufprall weitere Messwert-Ausschläge zur Folge haben kann, haben wir nach Bestimmung des Maximums die Messung für 300ms unterbunden. In den folgenden Graphiken sehen Sie beispielhaft die Messung der Beschleunigungsvektoren.

Drehen ums Handgelenk:



Breadboard mit Hand auf weicher Unterlage (Federkernmatraze):

Breadboard mit Hand auf Harte Unterlage stoßen (Tisch):



An den beiden Graphen kann man sehr gut folgende Zusammenhänge aus der Systemtheorie erkennen:

- Die Eigenfrequenz des schwingungsfähigen Gesamt-Systems (Breadboard, Hand, Federkernmatratze bzw. Tisch) variiert.
- Die Dämpfung des oberen Systems ist deutlich geringer (Matratze schwingt mit).

Diese Zusammenhänge gelten selbstverständlich analog auch für die im American Football eintretenden Beschleunigungen. Ein Aufprall auf den Rasen wird z.B. im Verlauf der Beschleunigung eine von einem Zusammenstoß zweier Helme deutlich unterschiedliche Charakteristik aufweisen.

Bei dem Aufprall auf weicher Unterlage bzw. weichem Gegenstand (z.B. Rasen) entstehen im Vergleich zur harten Unterlage deutlich höhere Messwert-Ausschläge. Wie bereits erwähnt, haben wir nach Bestimmung des Maximums die Messung für 300ms unterbunden. Damit spielt die Unterlage bzw. der Aufprallgegenstand eine untergeordnete Rolle. Die maximale Beschleunigung wird in beiden Fällen korrekt erkannt.

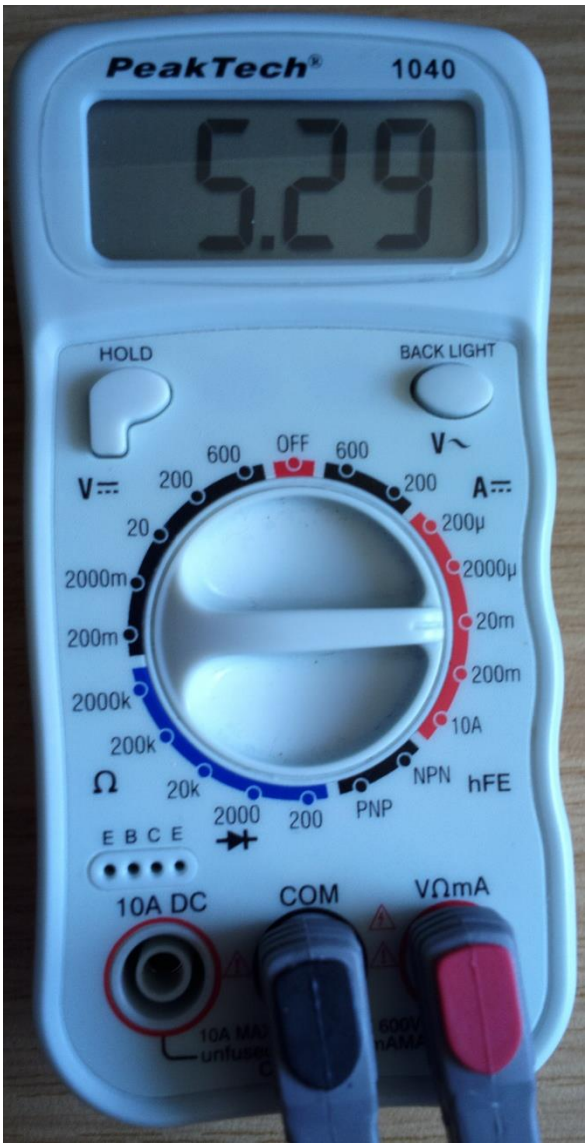
Anmerkung:

- Es muss in einer Testphase geprüft werden, ob der Messbereich von 27.7g ausreicht (davon ist auszugehen) und von jemandem mit medizinischer Qualifikation entschieden werden, ab welchem Grenzwert ein Stoß als gefährlich zu werten ist. Denn verschiedene Quellen, die wir gefunden haben, nennen sehr unterschiedliche Grenzenzwerte für die Gefährlichkeit eines Stoßes, sodass eine seriöse Entscheidung diesbezüglich durch uns nicht möglich ist. Ist dieser Wert bekannt, ist die Adaption des Programms dafür trivial.

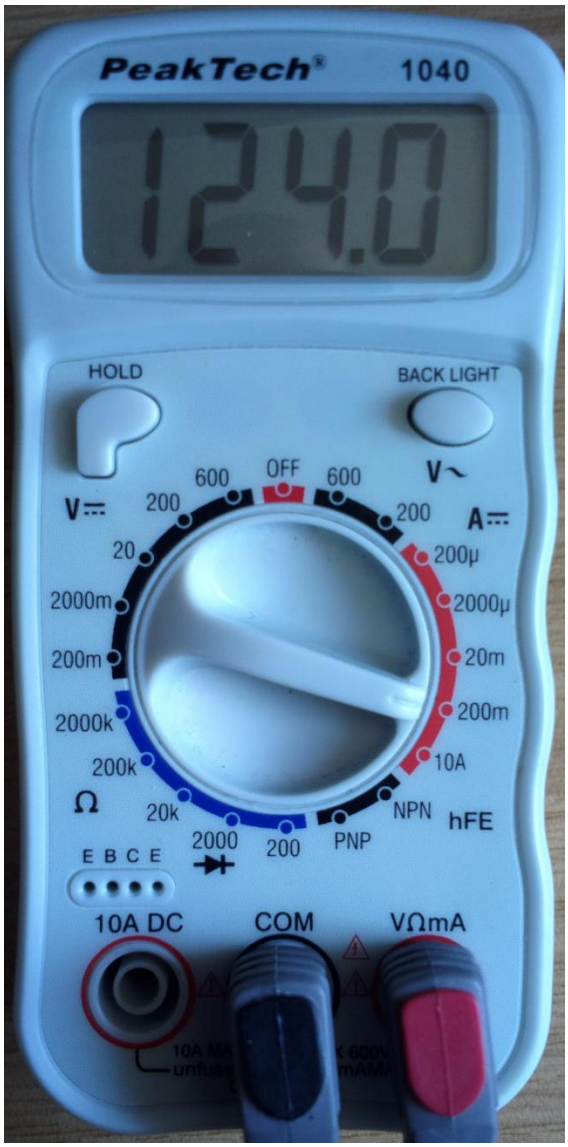
Strommessung

Um die Einheit später in den Helm einzubauen, wird ein Akku mit zusätzlicher Aufladeschaltung oder eine Batterie benötigt. Dies ist nicht mehr Teil unseres Semesterprojektes. Für eine Nachfolgegruppe haben wir als Orientierung folgende Stromverbrauchswerte in mA erfasst:

Sensor MPU-6050:



MPU-6050 + NodeMCU v2:



Der Stromverbrauch für das Gesamtsystem beträgt 124 mA. Ist ein niedrigerer Stromverbrauch gewünscht, empfehlen wir, statt des Entwicklungsboard NodeMCU v2 (auf dem ein ESP8266 aufgelötet ist), nur einen ESP8266 zu verwenden.

Hier ist im Datenblatt ein Stromverbrauch von durchschnittlich 80 mA angegeben. Gehen wir nun von einem durchschnittlichen Stromverbrauch von 85 mA (ESP8266 + MPU-6050) und einem Betriebseinsatz von 10 Stunden in der Woche aus, dann kann die Einheit z.B. durch ein 4,5 V Batteriepack mit einer Kapazität von 2300 mAh (3 x 1.5 V AA Alkali-Mangan) 18 Tage oder 27 Betriebsstunden versorgt werden.

Übertragung und Speicherung der Messung

Die ausgewerteten Messwerte werden drahtlos übertragen. Sobald der NodeMCUv2 mit einem WLAN verbunden ist, kann dieser die Messwerte übertragen. Hierfür benötigt man lediglich die SSID (WLAN-Name) und das Passwort. Das Übertragen der Messung erfolgt über Port 80.

Im nächsten Schritt legen wir die Daten auf einem Public Stream ab um sie weltweit abrufen zu können. Wir benutzen eine kostenfreie Cloud auf data.sparkfun.com. Hier wurden bei Erstellung des Streams zwei Platzhalter für Variablen namens „acc_h“ (Beschleunigung hoch) und „acc_l“ (Beschleunigung niedrig) konfiguriert. Die Messwerte über 8g werden auf „acc_h“ und die Messwerte unter 8g auf „acc_l“ abgelegt. Mit den folgenden Angaben können die abgelegten Daten abgerufen oder bearbeitet werden:

Public URL:

- http://data.sparkfun.com/streams/5JoxqobV6GIqq8jy4aYa

Public Key:

- 5JoxqobV6GIqq8jy4aYa

Private Key:

- 7B8x08klRqUEEBYkvg4g

Über die Public URL kann man die Messergebnisse weltweit einsehen. Jeder Messung wird eine Uhrzeit der Datenübertragung beigelegt.

DATA.SPARKFUN.COM

Football Helmet Senor B&S HsKA DIY-Solution for measuring hard hits in an American Football helmet based on a NodeMCU v2 Deleopement Kit with WiFi and a MPU-6050 Six-Axis (Gyro+Accelerometer) Sensor

Manage | Export to Analog.io

JSON | CSV | MySQL | PostgreSQL | Atom

TAGS

100% (50.00 of 50 MB) remaining.

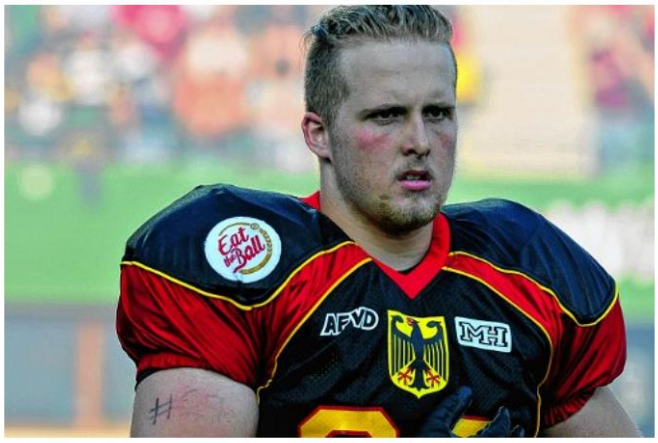
acceleration	temperature	timestamp
0.00	4.96	2016-04-29T13:16:05.950Z
0.00	3.89	2016-04-29T13:16:06.635Z
0.00	3.39	2016-04-29T13:16:05.118Z
0.00	3.03	2016-04-29T13:16:02.853Z
0.00	2.62	2016-04-29T13:15:33.250Z
0.00	5.36	2016-04-29T13:15:33.454Z
0.00	5.37	2016-04-29T13:15:01.934Z
8.65	0.00	2016-04-29T13:15:02.633Z
0.00	3.12	2016-04-29T13:10:27.588Z
0.00	4.36	2016-04-29T13:10:25.340Z
8.18	0.00	2016-04-29T13:10:22.941Z

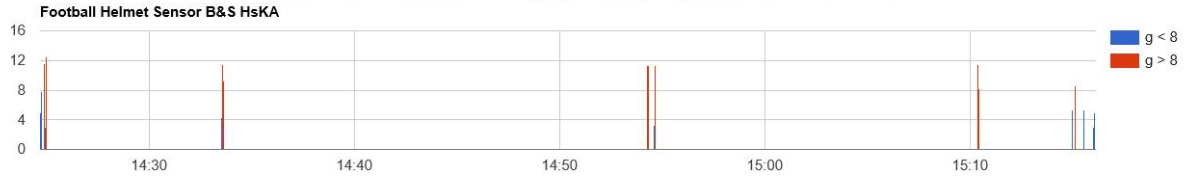
Anmerkungen:

- Nicht jede Sportstätte bietet ein umfassendes WLAN-Netz an. Hier empfehlen wir einen mobilen Hotspot mit dem Smartphone zu erstellen.
- Es können 100 Messwerte pro 15 min auf den aufgeführten Stream gepusht werden. Aus Erfahrungen aus dem Training, Spiel und Fernsehübertragungen können wir sagen, wie viele Hits maximal zu erwarten sind. Pro Spiel gibt es selten mehr als 140 Spielzüge (Offense, Defense, Special Team), wobei ein Spieler kaum über die komplette Zeit auf dem Feld steht. Bei der Annahme, dass der Spieler in einem Spielzug zwei heftige Aufpralle erleidet und ein Spiel nie unter 2h (8 x 15 min) dauert kamen wir auf einen maximalen Richtwert von 35 Hits pro15 min. (140x2/8=35), also mehr als ausreichend (~ Faktor 3).
- Ist eine Menge von Messwerten, bspw. von einem vergangenen Football-Match, nicht mehr von Interesse, da bereits begutachtet, können diese durch direktes Aufrufen der Cloud im Browser mit wenigen Klicks gelöscht werden (data.sparkfun.com/#PUBLIC_KEY -> Manage -> Clear all).

Visualisierung der Messergebnisse

Mit Hilfe von Google Chart haben wir die abgelegten Messwerte auf data.sparkfun.com visualisiert. Die erstellte HTML-Seite greift auf die abgelegten Werte über das Datenaustauschformat JSON zu. Aus diesen Werten wird eine Säulenfunktion über die Zeit erstellt. Die Messungen unter 8g sind hierbei blau und die über 8g rot. Zusätzlich haben wir uns dafür entschieden der Webseite den Titel „Football Helmet Sensor B&S HsKA“ zu geben und ein Bild von Benedikt Schumann in Footballausrüstung einzufügen. Das Bild und das Diagramm passen sich an die Größe des Browserfensters an.





Anmerkungen:

- Die HTML-Datei muss auf einem Server oder dem Gerät, an dem die Messwerte visualisiert werden sollen, abgelegt sein.
- Beim Aufruf der Seite im Internet Explorer müssen die geblockten Inhalte über ein Fenster am unteren Bildschirmrand zugelassen werden.
- Google gibt an, dass vorhandene HTML-Skripte, die auf die Google Chart Library zugreifen, bei neuen Revisionen dieser Library eventuell im Syntax angepasst werden müssen.

Features

- Erfassung von Beschleunigungen bis 16g bzw. 27.7g bei Vektorbetragsbildung
- Drahtlose Datenübertragung der Messungen
- Weltweiter Abruf der Daten von der data.sparkfun-Cloud möglich
- Visualisierung der Messungen über ein Diagramm mit „bedrohlichen“ und „ungefährlichen“ Messungen
- Konstruktion ist einbaukompatibel für den Football Helm
- Materialkosten unter 20€

Zeitplan

Informationstechnik Labor Fakultät MMT Hochschule Karlsruhe Technik und Wirtschaft		**Zeitplan** für **eine Beschleunigungs-messung im American Football Helm**							**MTB732** Sommersemester 2016 18.03.2016
Nr.	**Aktivitäten**	2016							**Jahr**
		März			April			Mai	**Monat**
		11	12	13	14	15	16	17	18	**Kalenderwoche**
1	Projektdefinition erstellen	X								
2	Produktrecherchen	X	X							
3	Bestellung der Bauteile		X							1
4	Verdrahtungsentwurf		X	X						
5	Signalübertragung			X	X	X				
6	Signalverarbeitung				X	X	X			2
7	Testphase						X	X	X	
8	Abschlusspräsentation								X	P
		Meilensteine: 1 2							Präsentation: P	

Stückliste

Bauteilbezeichnung	**Beschreibung**
NodeMCU v2	Development Kit based on ESP8266, integrates GPIO, PWM, ADC, I²C and WiFi
MPU-6050	Arduino board: 16-bit Six-Axis (Gyro + Accelerometer) Sensor with I²C
Breadboard, Jumper Wire	Electrical Connectors

Anhang

HTML-Seite

```
<html>
 <head>
  <!-- EXTERNAL LIBS-->
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
  <script src="https://www.google.com/jsapi"></script>
  <!-- EXAMPLE SCRIPT -->
  <script>
   // onload callback
   function drawChart() {
    • var public_key = '5JoxqobV6GIqq8jy4aYa';
    // JSONP request
    var jsonData = $.ajax({
     url: 'https://data.sparkfun.com/output/' + public_key + '.json',
     data: {page: 1},
     dataType: 'jsonp',
    }).done(function (results) {
     var data = new google.visualization.DataTable();
     data.addColumn('datetime', 'Time');
     data.addColumn('number', 'g < 8');
     data.addColumn('number', 'g > 8');
     $.each(results, function (i, row) {
      data.addRow([
       (new Date(row.timestamp)),
       parseFloat(row.temperature),
       parseFloat(row.acceleration)
      ]);
     });
```

```
        var chart = new google.visualization.ColumnChart($('#chart').get(0));
        chart.draw(data, {
          title: 'Football Helmet Sensor B&S HsKA'
        });
      });
    }
    // load chart lib
    google.load('visualization', '1', {
      packages: ['corechart']
    });
    // call drawChart once google charts is loaded
    google.setOnLoadCallback(drawChart);
  </script>
 </head>
 <body>
        <img src="http://www.verlagshaus-jaumann.de/media.facebook.32db13ba-d7b4-4f38-972e-b3c244890950.normalized.jpg" style="width: 50%; display: block;margin: 0 auto;" >
        <div id="chart" style="width: 100%;"></div>
 </body>
</html>
```

MCU-Nodev2

```
// Arduino-Einstellungen (getested):
/*
 * Board: NoceMCU 1.0 (ESP-12E Module)
 * CPU Frequency: 80Mhz
 * Flash Size: 4M (1M SPIFFS)
 * Upload Speed: 115200
 * Programmer: AVRISP mkII
 */

#include <Wire.h>
#include "own_MPU-6050.h"
#include <ESP8266WiFi.h>

#define G_THRESH 2.5
#define G_THRESH_RED 8.0
#define N_EXPIRE 300
#define MS_SAMPLING 1
#define httpPort 80

const char* ssid = "HIT_Repeater";
const char* password = "mechatronikwelt";
const char* host = "data.sparkfun.com";
const char* publicKey = "5JoxqobV6GIqq8jy4aYa ";
const char* privateKey = "•7B8x08klRqUEEBYkvg4g";

sensor_readings MPU;

struct two_floats
{
  float a, b;
};
// ____________________________________________________________________
void setup() {

  MPU6050_init();
```

```
  Serial.begin(115200);
  Serial.println();
  Serial.println();
  Serial.print("ESP8266 Connecting to ");
  Serial.println(ssid);  delay(MS_SAMPLING);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
// ____________________________________________________________________
void loop()
{
  static two_floats http_data;
  static float val;
  static float maximum;
  static unsigned char detected = 0;
  static unsigned int expire = 0;

  get_data();
  //print_raw_data();
  print_abs_acc();

  val = abs_acc(MPU);

  if ( expire ) { expire--; }
  else if ( val > G_THRESH ) detected = 1;

  if (detected)
  {
```

```
    if ( val > maximum ) maximum = val;
    else if ( val < G_THRESH ) { expire = N_EXPIRE; detected = 0; }
  }

  // MAXIMUM gefunden, state zählt gerade wieder runter.
  if ( expire == N_EXPIRE )
  {
    Serial.print("*** MAXIMUM: "); Serial.println(maximum);

    if (maximum > G_THRESH_RED) {http_data.a = maximum; http_data.b = 0.0;}
    else                        {http_data.a = 0; http_data.b = maximum;}
    WiFi_transmit(http_data);
    maximum = 0.0;
  }
  delay(MS_SAMPLING);
}
// ____________________________________________________________________
// Funktionen WiFi
// ____________________________________________________________________
void WiFi_transmit(two_floats& data)
{
  Serial.print("connecting to "); Serial.println(host);
  WiFiClient client;
  ESP.wdtFeed();
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }

  String url = "/input/";  url += publicKey;
  url += "?private_key=";  url += privateKey;
  url += "&acceleration="; url += data.a;
  url += "&temperature=";  url += data.b;

  Serial.print("Requesting URL: ");
  Serial.println(url);
  client.print(String("GET ") +  url +  " HTTP/1.1₩r₩n" +
               "Host: " +  host +  "₩r₩n" +
```

```
            "Connection: close₩r₩n₩r₩n");
  //delay(10);
  ESP.wdtFeed();

  // Read all the lines of the reply from server and print them to Serial
  while(client.available()){
    String line = client.readStringUntil('₩r');
    Serial.print(line);
  }

  Serial.println();
  Serial.println("closing connection");

  ESP.wdtFeed();
}

// Funktionen MPU-6050
// ____________________________________________________________________________
void get_data()
{
  Wire.beginTransmission(0x68);
  Wire.write(0x3B);
  Wire.endTransmission(false);
  Wire.requestFrom(0x68,6,true);
  MPU.AcX = Wire.read() << 8 | Wire.read();
  MPU.AcY = Wire.read() << 8 | Wire.read();
  MPU.AcZ = Wire.read() << 8 | Wire.read();
  /*MPU.Tmp = float( Wire.read() << 8 | Wire.read() ) / 340.00 +  36.53;
  MPU.GyX = Wire.read() << 8 | Wire.read();
  MPU.GyY = Wire.read() << 8 | Wire.read();
  MPU.GyZ = Wire.read() << 8 | Wire.read();*/
}

void print_raw_data()
{
  Serial.print(MPU.AcX); Serial.print(" ");
  Serial.print(MPU.AcY); Serial.print(" ");
  Serial.print(MPU.AcZ); Serial.print(" ");
```

```
  Serial.print(MPU.GyX); Serial.print(" ");
  Serial.print(MPU.GyY); Serial.print(" ");
  Serial.print(MPU.GyZ); Serial.print(" ");
  Serial.print(MPU.Tmp); Serial.println(" ");
}

void print_abs_acc()
{
  Serial.println( abs_acc(MPU) );
}

// Rohwerte zu Beschleunigung in g
float raw_to_g(int16_t& val)
{
  // + - 8g: 4096LSB/g (pg. 29)
  return float(val) / 4096.00;
}

// Betrag des Vektors der 3 Beschleunigungsvektoren
float abs_acc(sensor_readings& val)
{
  float temp =
    ( raw_to_g(val.AcX) * raw_to_g(val.AcX) ) +
    ( raw_to_g(val.AcY) * raw_to_g(val.AcY) ) +
    ( raw_to_g(val.AcZ) * raw_to_g(val.AcZ) );

  return sqrt(temp);
}
```

Header MPU-6050

```
#include <Wire.h>
#define MPU6050 0x68

struct sensor_readings
{
  int16_t AcX, AcY, AcZ, GyX, GyY, GyZ;
  float Tmp;
};

/*
 * Accelerometer output rate = 1kHz
 * Gyroscope output rate = 8kHz bzw. 1kHz wenn DLPF = enabled
 */

void write_reg(unsigned char reg, unsigned char val)
{
   Wire.begin();
   Wire.beginTransmission(MPU6050);
   Wire.write(reg);
   Wire.write(val);
   Wire.endTransmission();
}

void MPU6050_init()
{
   write_reg(0x6B, 0x00); // Aufwecken
   write_reg(0x1C, ((unsigned char) 2) << 3);   // 0 = 2g, 1 = 4g, 2 = 8g, 3 = 16g
}
```

Quellenverzeichnis

Stand der Technik:

http://www.theshockbox.com/ (02.05.2016, 12:09)

Inbetriebnahme NodeMCUv2:

https://github.com/adafruit/ESP8266-Arduino (02.05.2016, 12:17)

Inbetriebnahme MPU-6050:

http://www.theshockbox.com/helmet-sensors-technology-for-concussion-management/shockbox-helmet-sensors-research/ (02.05.2016)

Speicherung/ Visualisierung der Messwerte:

http://hpclab.blogspot.de/2015/06/esp8266-based-wifi-weather-monitoring.html (02.05.2016, 12:34)

https://developers.google.com/chart/interactive/docs/gallery/columnchart (02.05.2016, 12:37)

https://data.sparkfun.com/ (02.05.2016, 12:39)