Hard- und Softwareprojekt

„3D-Mouse“ V0.1

Felix Holz, Yannik Rauter

Inhaltsverzeichnis

1. Aufgabenstellung

1.1. Ziel des Projekts

Es soll eine grafische Visualisierung eines beweglichen Objektes in Unity erstellt werden. Dazu wird Gyrosensor, welcher an ein Arduino NodeMCU angeschlossen ist in der realen Welt bewegt, die Neigung sowie Position sollen dann in Unity live wiedergegeben werden.

1.2. Detaillierte Funktionsbeschreibung

Ein 6-Achsen Gyrosensor (3-Wege Neigung, 3-Wege Beschleunigung) [Bauteil: Grove 6-Axis Accelerometer & Gyroscope v1.0] wird mittels I²C-Bus mit einem Wifi fähigem Arduino basierten Mikrocontroller [Bauteil: NodeMCU mit ESP8266] verbunden. Die Sensordaten werden am Microcontroller ausgewertet und als UDP-Netzwerkpaket an einen Computer gesendet.

Dort werden diese zunächst empfangen und anschließend in einem selbst erstellten Programm [Software: Unity] ein beliebiges Objekt grafisch in einem dreidimensionalen Raum dargestellt. Die Bewegung sowie Neigung dieses Objektes soll den Sensorddaten entsprechen und mit möglichst niedriger Latenz (quasi live) vom Gyrosensor übernommen werden.

1.3. Blockschaltbild

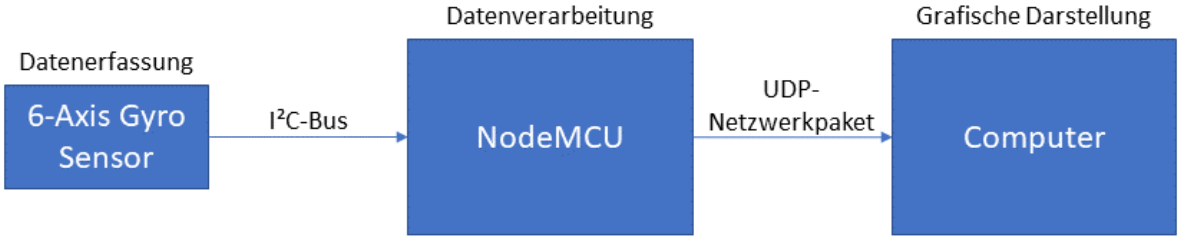

Abbildung 1: Blockschaltbild des Projektaufbaus

2. Entwurf

2.1. NodeMCU

Begonnen wird mit der Arbeit am NodeMCU.

2.1.1. Sensordatenerfassung

Zunächst wird der 6-Achsen Gyrosensor per I^2C-Bus mit dem NodeMCU verbunden. Dazu wird der Sensor nicht nur mit 3,3V und GND verbunden, sondern auch die SDA- und SCL-Leitungen auf die korrekten Standard-I^2C-Pins des Arduino Bausteins (SDA = GPIO4 = D2 und SCL = GPIO5 = D1).

Später wird in der Software die Kommunikation mittels I^2C-Protokoll über eine Methode in der Library des Sensors gestartet.

2.1.2. Sensordatenverarbeitung

Da der Sensor so genannte „Raw“-Werte für den Neigungswinkel der drei Achsen ausgibt, müssen diese zunächst in brauchbare Winkelwerte umgewandelt werden. Dazu werden die Werte über einen Zeitraum hinweg gemessen und anschließend der Mittelwert gebildet. Wichtig ist ebenso, dass der Sensor während der ersten beiden Sekunden nach dem Reset des NodeMCU still und flach am Tisch liegt, da zu diesem Zeitpunkt die Referenzwerte kalibriert werden.

2.1.3. Datenweiterleitung zum Computer

Zum Schluss werden die Daten über ein UDP-Netzwerkpaket zum Computer gesendet, um dort grafisch dargestellt zu werden. Hierfür wird ebenfalls eine Library (WifiUDP) benötigt, welche das erstellen und versenden (optional auch noch das Empfangen) von Netzwerkdatenpaketen ermöglicht. Die vorbereiteten Sensordaten werden nun an die IP-Adresse des Computers gesendet, dies geschieht mehrmals in der Sekunde. Für die Datenübertragung wird der Port 8888 verwendet.

2.1.4. NodeMCU Software Sourcecode

Es folgt nun eine Abbildung des Programmcodes der Mikrocontrollersoftware.

```
#include "SparkFunLSM6DS3.h"
#include "Wire.h"
#include "SPI.h"
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>

LSM6DS3 myIMU( I2C_MODE, 0x6A );

const char* ssid = "htl-IoT";
const char* password = "iot..2015";

WiFiUDP Udp;
unsigned int port = 8888;
char incomingPacket[255];

float ax_offset = 0;
float ay_offset = 0;
float az_offset = 0;
float gx_offset = 0;
float gy_offset = 0;
float gz_offset = 0;

void setup()
{
  myIMU.begin();

  Serial.begin(115200);
  delay(1000);
  Serial.println("Processor came out of reset.\n");

  Serial.printf("Connecting to %s ", ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
  Serial.println(" connected");

  Udp.begin(8888);
  Serial.printf("UDP Server Started");

  Serial.println("reading offset...");

  float ay = myIMU.readFloatAccelY();
  float ax = myIMU.readFloatAccelX();
  float az = myIMU.readFloatAccelZ();
  float gx = myIMU.readFloatGyroX();
  float gy = myIMU.readFloatGyroY();
  float gz = myIMU.readFloatGyroZ();

  float tries = 300.0f;
```

Abbildung 2: NodeMCU Code

2.2. Unity

Nachdem die Mikrocontrollersoftware funktionsfähig ist wird mit der Arbeit an der Software in Unity fortgesetzt.

2.2.1. Datenempfang und grafische Darstellung

Im Unity-Softwarebaustein werden zunächst die vom NodeMCU versandten Daten empfangen und zwischengespeichert. Anschließend wird in der virtuellen Umgebung ein dreidimensionales Objekt grafisch dargestellt. Dessen Rotationsbewegung sowie Neigungsposition ist von den empfangenen Werten abhängig.

2.2.2. Unity Software Sourcecode

Nun folgt eine Abbildung des Programmcodes der Unity Software.

```
#include "SparkFunLSM6DS3.h"
#include "Wire.h"
#include "SPI.h"
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>

LSM6DS3 myIMU( I2C_MODE, 0x6A );

const char* ssid = "htl-IoT";
const char* password = "iot..2015";

WiFiUDP Udp;
unsigned int port = 8888;
char incomingPacket[255];

float ax_offset = 0;
float ay_offset = 0;
float az_offset = 0;
float gx_offset = 0;
float gy_offset = 0;
float gz_offset = 0;
```

Abbildung 3: Unity Code

2.3. Hardwareaufbau

Abschließend werden die einzelnen Bauteile final zusammengesetzt.

2.3.1. Zusammenbau der Komponenten

Um ein kompakteres und stabileres Gesamtpaket zu erhalten wird der verwendete Sensor mithilfe einer Heißklebepistole an die Unterseite des NodeMCUs geklebt. Des Weiteren wird eine 9-V-Block Batterie in Verbindung mit einem Spannungsregler (L78L33) und dessen externer Beschaltung (je ein Kondensator am Ein- und Ausgang) verwendet. Der gesamte Schaltungsaufbau ist in einer Abbildung später in der Dokumentation zu finden.

2.3.2. Probleme bei der Spannungsreglerschaltung

Beim erstmaligen Aufbau der Spannungsreglerschaltung wurden der Ein- und Ausgangspin des Spannungsreglers vertauscht, da aus dem Datenblatt auf den ersten Blick nicht eindeutig erkennbar war, ob die Pins in der Abbildung des Gehäuses in das Bild hinein oder aus dem Bild hinaus zeigen. Als die Kondensatoren dazu geschalten und die Schaltung getestet wurden, funktionierte diese nicht. Nachdem die Beschaltung der Ein- und Ausgangspins allerdings vertauscht wurde war die Schaltung funktionsfähig, die 9 V der Batterie werden erfolgreich auf 3,3 V für den NodeMCU herabgeregelt.

Der Aufbau der externen Beschaltung des Spannungsreglers befindet sich in der nachfolgenden Abbildung.

Figure 3: Test circuits

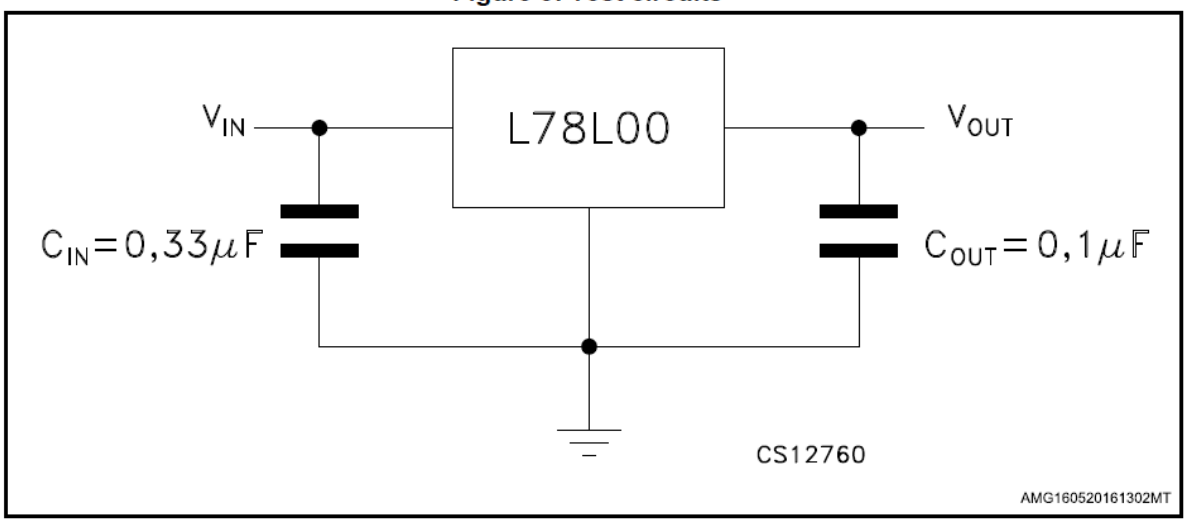

Abbildung 4: Externe Beschaltung des Spannungsreglers

Anschließend folgt noch das Pinout des Spannungsreglers laut Datenblatt.

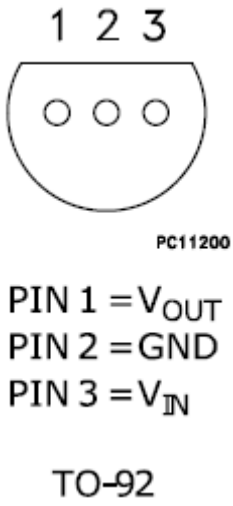

Abbildung 5: Spannungsregler (L78L33) Pinout [Datenblattauszug]

2.3.3. Einbau in das Gehäuse

Damit das Gerät handlicher und attraktiver wird werden die ohnehin schon kompakten Komponenten in ein Gehäuse eingebaut, welches mit einem 3D-Drucker gedruckt wurde. Diese Zusatzaufgabe wurde erst kurz vor der Fertigstellung des Projektes begonnen, das Gehäuse wurde daraufhin in SOFTWARE gezeichnet und in der Schule gedruckt. Auch eine Status-LED wird noch verbaut, um auf Fehler in der WLAN-Verbindung hinweisen zu können.

3. Fertigungsunterlagen

3.1. Schaltungen

Schaltung des Sensors, NodeMCU und Batterie

3.2. Layout

NONE

3.3. Bauteilliste

Hier eine Liste der verwendeten Bauteile:

Stückzahl	Bauteilkategorie	Bauteilbezeichnung/-wert
1	Mikrocontroller	NodeMCU ESP8266
1	Neigungs- & Beschleunigungssensor	Grove 6-Axis Accelerometer & Gyroscope v1.0
1	Spannungsregler	L78L33
1	Mobile Spannungsversorgung (Batterie)	9-V-Block
1	Batterieclip	Passend für 9-V-Block
1	Kondensator	100 nF
1	Kondensator	330 nF

4. Inbetriebnahme und Testergebnisse

4.1. Überschrift

Platzhaltertext

4.2. Probleme

Folgende Probleme wurden während der Entwicklungsphase überwunden:

4.2.1. Sensordatenempfang nicht möglich

Anfangs gab es Probleme beim Empfang der Sensordaten. Dieser Fehler wurde durch Überprüfen der Verkabelung, des Sensors, des I^2C-Protokolls mittels Saleae Logic Analyzer sowie der Software behoben. Eine Zeile des NodeMCU Programms beinhaltete die falschen I^2C Pins, weshalb die Sensordaten nicht ausgelesen werden konnten.

5. Projektmanagement

5.1. Produktstrukturplan

Der grobe Plan des fertigen Produktes sieht folgendermaßen aus:

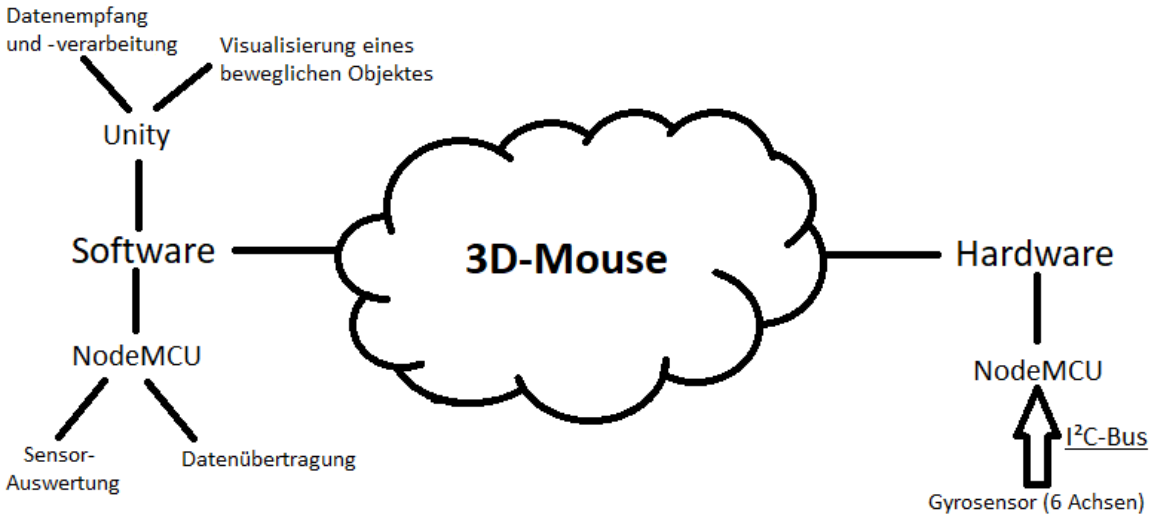

Abbildung 6: Produktstrukturplan als Mindmap

5.2. Projektstrukturplan

Ein Plan der Struktur des Projektes ist in Abbildung 3 zu sehen.

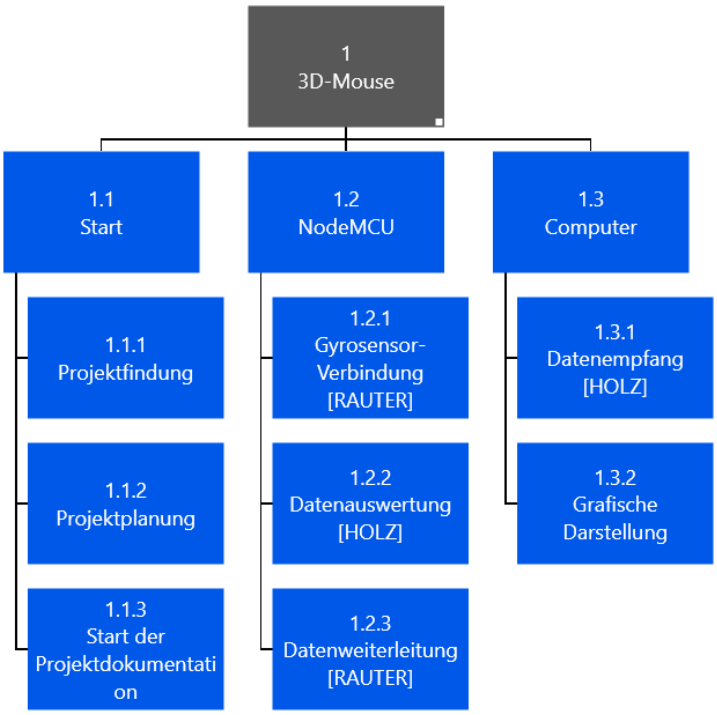

Abbildung 7: Projektstrukturplan

5.3. Ganttdiagramm

Als nächstes folgt eine Grafik des Ganttdiagramms:

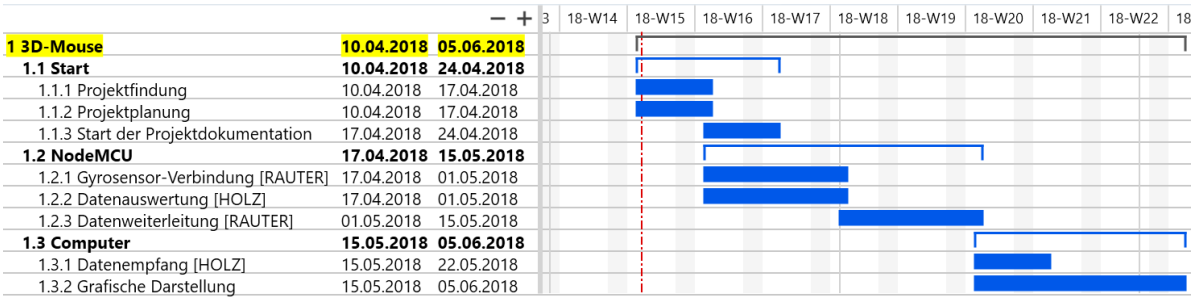

Abbildung 8: Ganttdiagramm

5.4. Arbeitspakete

Die Arbeitspakete wurden laut der unten ersichtlichen Tabelle aufgeteilt:

Vorgangs-Nr. (Arbeitspaket)	Beschreibung, Ziel, Arbeitsinhalt	Dauer (in Stunden)	Beteiligte Schüler
1.1.1	Projektfindung (Einigung auf ein Projekt)	1	Beide
1.1.2	Projektplanung (Grober Ablauf)	1	Beide
1.1.3	Start der Projektdokumentation	1	Beide
1.2.1	Verbindung des Gyrosensors mit NodeMCU	2	Rauter
1.2.2	Auswertung der Daten des Gyrosensors	2	Holz
1.2.3	Weiterleitung der Daten zum Computer	2	Rauter
1.3.1	Empfang der Gyrosensor-Daten	2	Holz
1.3.2	Grafische Darstellung in Unity	4	Beide

Tabelle 1: Arbeitspakete (Dauer und Aufteilung)