HTL | MÖSSINGERSTRASSE

Hard- und Softwareprojekt

„3D-Mouse“ V1.0

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 ein 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) wird mittels I²C-Bus mit einem Wifi fähigem Arduino basierten Mikrocontroller verbunden. Die Sensordaten werden am Microcontroller ausgewertet und als UDP-Netzwerkpaket an einen Computer gesendet.

Dort werden die Daten zunächst empfangen und anschließend durch ein selbst erstelltes Programm an einem beliebigen dreidimensionalen Objekt grafisch dargestellt. Die Bewegung sowie Neigung dieses Objektes soll den Sensordaten entsprechen und mit möglichst niedriger Latenz (quasi live) vom Gyrosensor übernommen werden.

Die genauen Bezeichnungen der Bauteile können der Bauteilliste, welche sich ebenfalls in dieser Dokumentation befindet, entnommen werden.

1.3. Blockschaltbild

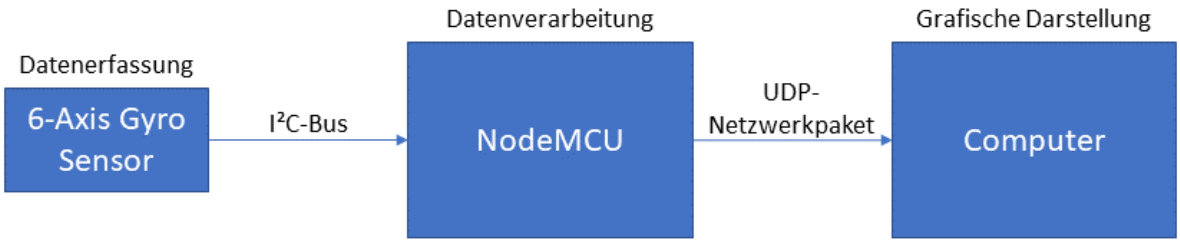

Abbildung 1: Blockschaltbild des Projektaufbaus

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) gelegt.

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

2.1.2. Sensorkalibrierung

Damit die Messwerte die erforderliche Genauigkeit erreichen wird der Sensor nach dem Starten oder beim Reset jedes Mal neu kalibriert. Dazu werden 100 Messwerte im Abstand von 10 Millisekunden erfasst. In diesem Zeitraum sollte der Sensor in der Ausgangsposition liegen und möglichst nicht bewegt werden. Anschließend wird der durchschnittliche Fehler der 100 Messwerte errechnet und als Offset gespeichert. Bei jedem Messvorgang wird der Offsetwert vom Messwert abgezogen und das Ergebnis wird an den Computer gesendet.

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 folgen einige Abbildungen aus den wichtigsten Stellen der NodeMCU Software, in der Programmiersprache C++.

```
float ax = 0.0f;
float ay = 0.0f;
float az = 0.0f;
float gx = 0.0f;
float gy = 0.0f;
float gz = 0.0f;

for (float i = 0; i < OFFSET_MEASUREMENTS; i++)
{
  ay += myIMU.readFloatAccelY();
  ax += myIMU.readFloatAccelX();
  az += myIMU.readFloatAccelZ();
  gx += myIMU.readFloatGyroX();
  gy += myIMU.readFloatGyroY();
  gz += myIMU.readFloatGyroZ();
  delay(OFFSET_DELAY);
}

ax_offset = ax/OFFSET_MEASUREMENTS;
ay_offset = ay/OFFSET_MEASUREMENTS;
az_offset = az/OFFSET_MEASUREMENTS;
gx_offset = gx/OFFSET_MEASUREMENTS;
gy_offset = gy/OFFSET_MEASUREMENTS;
gz_offset = gz/OFFSET_MEASUREMENTS;
```

Abbildung 2: Erfassung des Offsets

```
int packet = udp.parsePacket();
if(packet)
{
  int size = udp.read(packetBuffer, 3);
  int color = (packetBuffer[0] << 16) | (packetBuffer[1] << 8) | packetBuffer[2];
  RGB(color);
}
```

Abbildung 3: Steuerung der RGB-LED über Unity

```
void RGB(unsigned int color)
{
  unsigned int r = color & 0xFF0000;
  r = r >> 16;

  unsigned int g = color & 0xFF00;
  g = g >> 8;

  unsigned int b = color & 0xFF;

  analogWrite(RGB_RED, r);
  analogWrite(RGB_GREEN, g);
  analogWrite(RGB_BLUE, b);
}
```

Abbildung 4: Steuerung der RGB-LED mit hexadezimalen Farbcodes

```
float gx = myIMU.readFloatGyroX() - gx_offset;
float gy = myIMU.readFloatGyroY() - gy_offset;
float gz = myIMU.readFloatGyroZ() - gz_offset;

bool button = digitalRead(BUTTON);
udp.beginPacket(ip, port);

String data = "t ";
data += millis();
data += " a \n";
data += ax;
data += "\n";
data += ay;
data += "\n";
data += az;
data += "\ng\n";
data += gx;
data += "\n";
data += gy;
data += "\n";
data += gz;
data += "\nbtn\n";
data += button ? "1" : "0";
data += "\n";
data += "";


char data2[data.length()];
data.toCharArray(data2, data.length());

PRINT(data2);

udp.write(data2);
udp.endPacket();
```

Abbildung 5: Erfassen und Senden der Sensorwerte

2.2. Unity

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

2.2.1. Datenempfang und Verarbeitung

Die Daten, die vom Sensor ausgelesen und über das Netzwerk an den Computer übertragen werden beschreiben die Änderung der Neigung und die derzeitige Beschleunigung. Um die Neigungsdaten in einen Neigungswinkel umzuwandeln muss jeder einzelne Wert (x, y, z) integriert (summiert) werden.

Im Unity-Skript werden zunächst die vom NodeMCU versandten Daten mit einem UdpClient empfangen und in Variablen zwischengespeichert. 60 Mal in der Sekunde wird mit der von Unity zur verfuegung gestellten Methode „FixedUpdate“ die Änderung der Rotation vom Sensor zur Rotation des dargestellten 3D Objekts addiert. Des Weiteren ist noch anzumerken, dass die Rotationsdaten vom Sensor zu hoch sind. Aus diesem Grund werden die Rotationswerte durch 50 dividiert. Damit ist die Rotation des Objektes sowie die Rotation des Sensors ausreichend identisch.

2.2.2. Kontrolle der RGB-LED

Die am Mikrokontroller angeschlossene LED kann im Normalbetrieb ferngesteuert werden. Dabei wird ein neues Unity-Skript erstellt, welches in einer Subroutine 200 Mal pro Sekunde die RGB Daten (3 Kanäle, jeweils 0 bis 255 → 3 bit) an den Mikrocontroller sendet.

2.2.3. Unity Software Sourcecode

Nun folgen kurze Code Ausschnitte vom Unity Programm in der Programmiersprache C#.

```
IPEndPoint endpoint = new IPEndPoint(IPAddress.Any, 0);

byte[] receiveBytes = m_Client.Receive(ref endpoint);
{
    string data = Encoding.ASCII.GetString(receiveBytes);

    stoff = data;

    string[] split = data.Split('\n');

    float.TryParse(split[1], out ax);
    float.TryParse(split[2], out ay);
    float.TryParse(split[3], out az);
    float.TryParse(split[5], out gx);
    float.TryParse(split[6], out gy);
    float.TryParse(split[7], out gz);
    int temp = 0;

    int.TryParse(split[9], out temp);
    if (temp == 0)
        button = false;
    else
        button = true;
}
```

Abbildung 6: Empfang und zwischenspeichern der Daten

```
void FixedUpdate()
{
    transform.Rotate(new Vector3(gx / scaleFactor, gy / scaleFactor, gz / scaleFactor));
}
```

Abbildung 7: Rotieren des 3D Objektes

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. Für Input und Output wird ein Taster mit Pull-Down Widerstand, ein Reset Taster und eine RGB-LED mit Vorwiderständen eingesetzt. 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

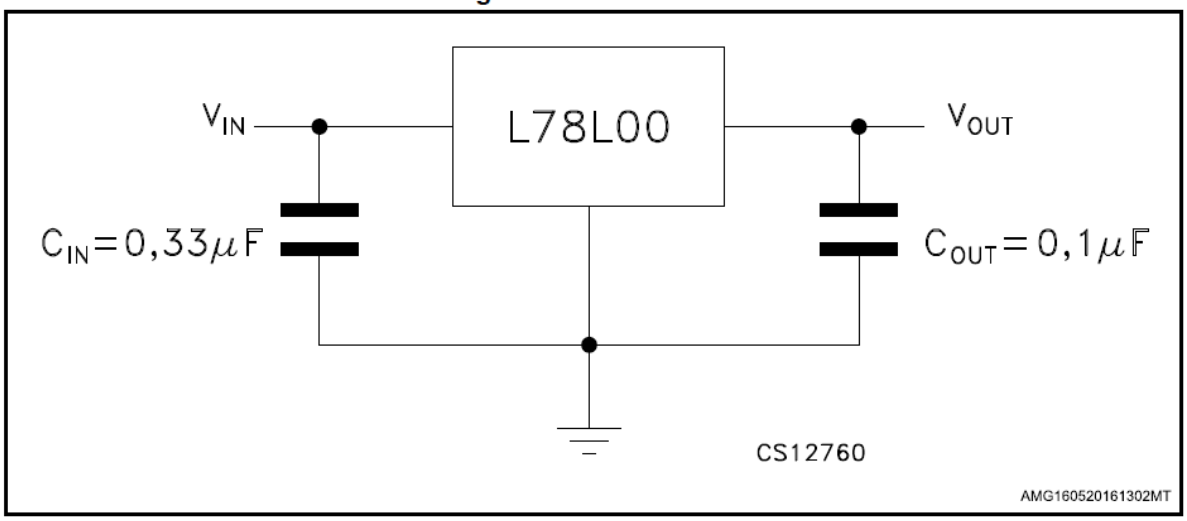

Abbildung 8: Externe Beschaltung des Spannungsreglers

Anschließend folgt noch das Pinout des Spannungsreglers laut Datenblatt.

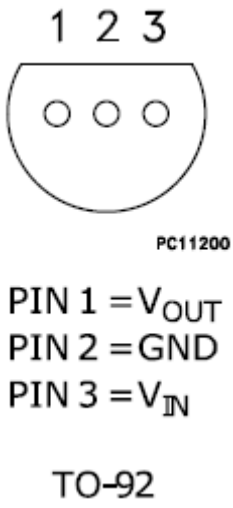

Abbildung 9: Spannungsregler (L78L33) Pinout
[Datenblattauszug]

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 (Fusion 360) gezeichnet und in der Schule gedruckt.

2.3.4. Ungeplantes Zusatzelement: RGB-LED

Gegen Ende des Projektes wurde eine weitere, anfangs ungeplante Funktion zur Demonstration der Datenübertragung zwischen einem Computer und dem NodeMCU hinzugefügt: Eine RGB-LED wurde zusätzlich verbaut, deren Farbe kann mithilfe von 3 Farbintensitätsreglern (je einem pro Farbe) im Unity Programm eingestellt werden.

Während das Gerät startet (also nachdem es eingeschalten oder zurückgesetzt wurde) dient die RGB-LED als Status LED. Am Beginn des Startvorganges leuchtet sie vorerst rot, während der Sensor kalibriert wird gelb, während die WLAN Verbindung durchgeführt wird blau und wenn alles funktioniert hat grün. Dies signalisiert dann, dass das Gerät zum Einsatz bereit ist.

2.3.1. Ungeplantes Zusatzelement: Taster und Reset Knopf

Des Weiteren wurde ein Taster mit Pulldown-Widerstand und Debounce-Kondensator eingebaut, welcher ebenfalls durch Unity auslesbar ist. Neben dem Funktionstaster wird ein externer Reset Knopf eingebaut, welcher bei Betätigung den RST Pin mit GND verbindet um den Mikrocontroller zurückzusetzen.

3. Fertigungsunterlagen

3.1. Schaltungen

Es folgt nun eine Abbildung der gesamten Schaltung des Gerätes.

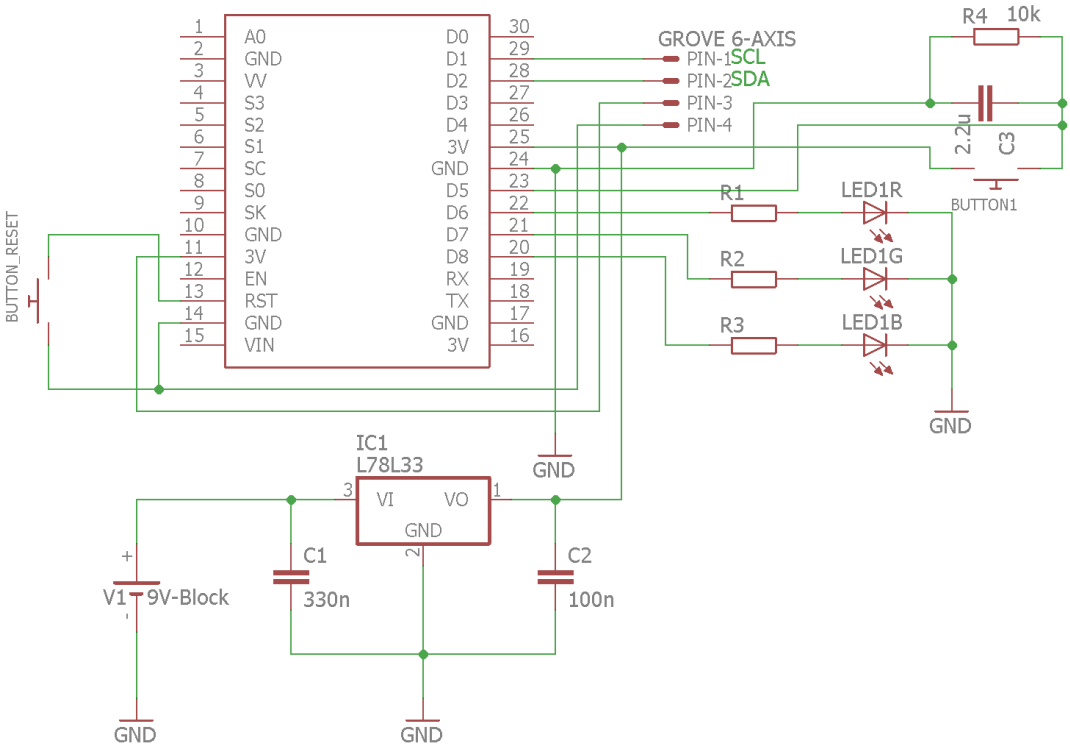

Abbildung 10: Schaltung des Spannungsreglers, NodeMCUs, Sensors, der Taster und der RGB-LED

3.2. Layout

Aufgrund der benötigten Kompaktheit wurde keine Platine angefertigt. Alle Bauteile wurden direkt an den Mikrokontroller gelötet und wenn nötig mit Heißkleber befestigt.

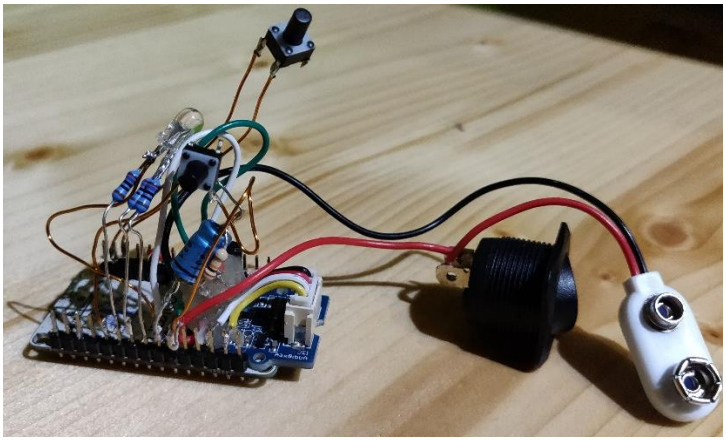

Abbildung 11: Schaltungsaufbau

3.3. Source Code

Für die Zusammenarbeit wurde das Version Control System (VCS) Git eingesetzt. Dieses ist auf der Website Github.com kostenlos implementiert.

Der komplette Source Code vom Mikrocontroller sowie das Unity Projekt, Eagle Dateien und Bilder sind in folgendem Repository zu finden:

https://github.com/NeXTormer/3D-Mouse

3.4. Bauteilliste

Hier eine Liste der verwendeten Bauteile:

Stückzahl	Bauteilkategorie	Bauteilbezeichnung/-wert
1	Mikrocontroller	NodeMCU 12E 1.0 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
1	Kondensator	2200 nF
3	Widerstand	270 Ω
1	Widerstand	10 kΩ
1	LED	RGB-LED
2	Taster	Taster

4. Inbetriebnahme und Testergebnisse

4.1. Probleme

Folgende Probleme wurden während der Entwicklungsphase überwunden:

4.1.1. 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.

4.1. Projektabschluss und Funktionstest

Um das Gerät zu verwenden muss lediglich der On-/Off-Switch umgelegt werden, anschließend verbindet sich das NodeMCU selbstständig mit dem schulischen WLAN-Netzwerk und beginnt damit, die Sensordaten an die in der Software definierte IP Adresse zu versenden.

Wird nun am passenden PC die Software gestartet, so entspricht die Neigungsposition des darin dargestellten dreidimensionalen Objekts der des Sensors.

Des Weiteren wurde eine Android Version des Programms erstellt, da dies mit Unity sehr einfach ist. Diese Version ist funktionsweise identisch mit der PC Version.

5. Projektmanagement

5.1. Produktstrukturplan

Der grobe Plan des fertigen Produktes sieht folgendermaßen aus:

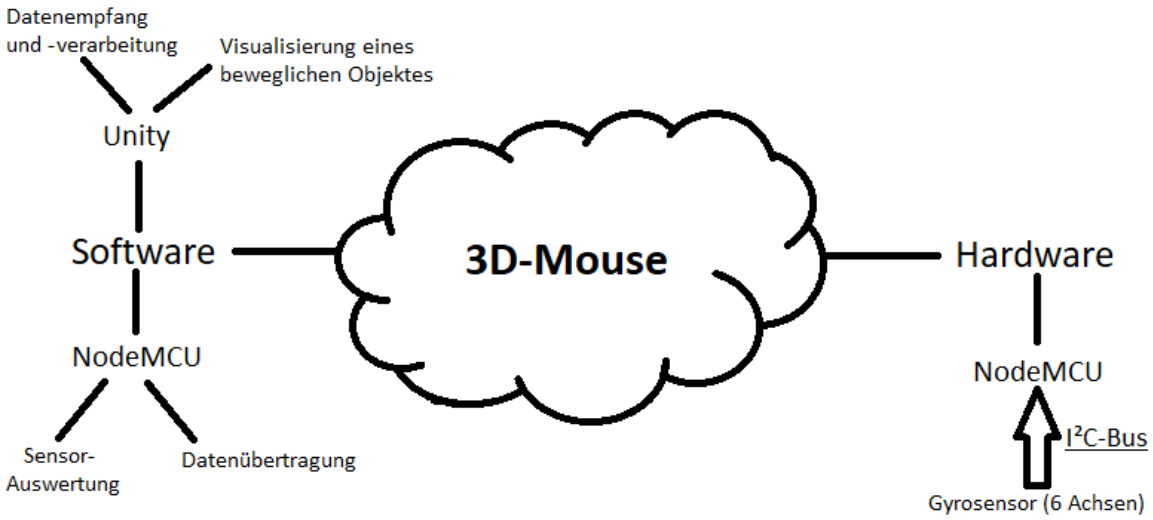

Abbildung 12: Produktstrukturplan als Mindmap

5.2. Projektstrukturplan

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

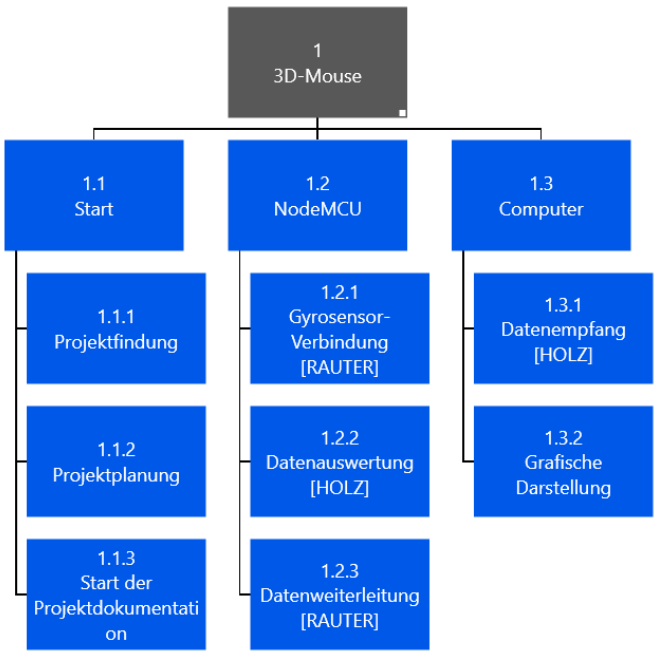

Abbildung 13: Projektstrukturplan

5.3. Ganttdiagramm

Als nächstes folgt eine Grafik des Ganttdiagramms:

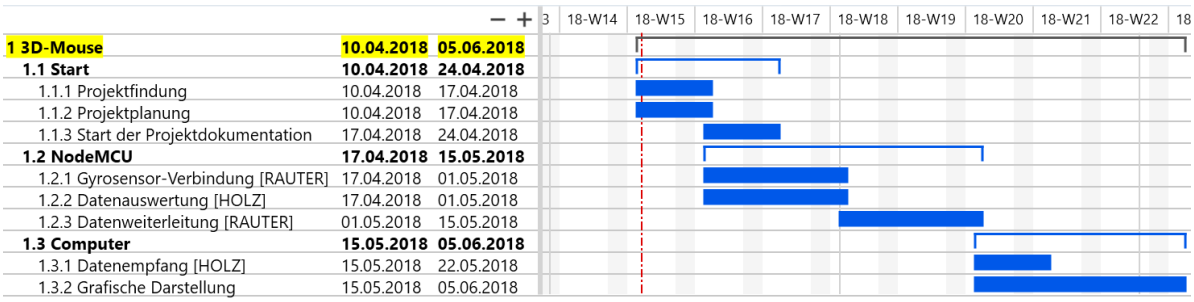

Abbildung 14: Ganttdiagramm

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)