Projekt-Handbuch:

Datenübertragung

Danijel Stojanovic

Die Datenübertragung besteht aus drei Microcontrollern, wie auch das Einlesen von Sensordaten.

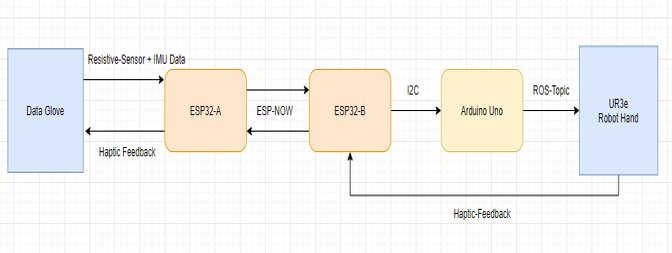


Abbildung: Datenübertragung – Prinzip

Dafür notwendig sind:

* 2 x ESP32
* Arduino UNO
* Resistiver Sensor
* IMU (MPU-9250)

Jeder Microcontroller hat seinen eigenen Programmcode, der sofort abgelaufen wird, sofern der Microcontroller eine Spannungsversorgung hat.

**Sensordatenerfassung:**

Der resistive Sensor wird mittels ADC vom ESP32-A eingelesen. Dazu ist es notwendig den an dem ADC anzustecken. Der ESP32 Microcontroller hat mehrere ADC Pins zu Verfügung.

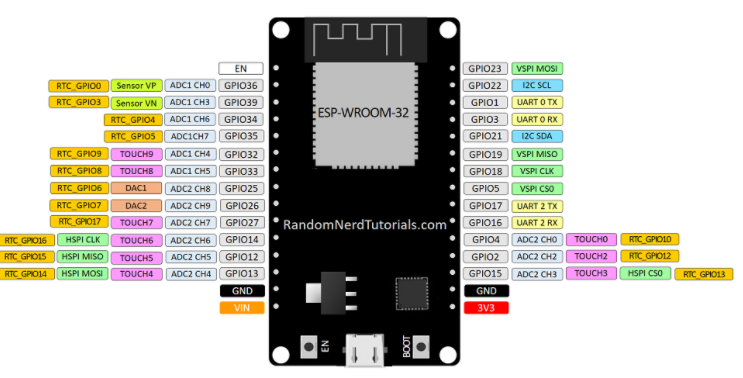


Abbildung: ESP32 – Pinbelegung

In der Abbildung sieht man welche ADC man bei dem ESP32 verwenden kann, diese müssen dann aber auch im Code geändert werden. In unserem Fall fängt es mit dem Pin „34“ an und geht so weiterführend für alle fünf Finger bis Pin „38“ weiter.

Der IMU Sensor MPU-9250 wird mittels I²C ausgelesen.

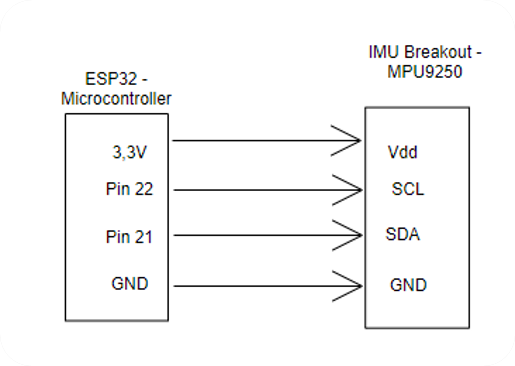


Abbildung: I²C – Pinbelegung

In der Abbildung sieht man wie der MPU-9250 mit dem ESP32-A verbunden wird. Die Hardware-technische Verdrahtung ist sehr einfach, da Vdd mit 3.3 V versorgt wird und die Pins „21“ und „22“ die Serial Clock (SCL) und die Serial Data (SDA) Verbindungen sind. Weiters wird Ground (GND) mit Ground verbunden.

Sollte im Programmcode was geändert werden müssen muss man darauf achten, dass die Bibliotheken, die dafür notwendig sind, installiert sind.

Für das Einlesen des Sensoren, wird in dem Programm eine spezielle Bibliothek gebraucht. Die Bibliothek ist die sogenannte „<MPU9250\_asukiaaa.h>“. Diese kann von hier heruntergeladen werden <https://github.com/asukiaaa/MPU9250_asukiaaa>.

**Datenübertragung:**

Wie schon vorher erwähnt besteht die Datenübertragung aus drei Microcontrollern wie auch aus drei Programmcodes.

**ESP32-A:**

Der ESP32-A liest die Sensordaten (Resistiv,IMU) ein und sendet sie über ESP-Now an den ESP32-B. Sollte beim Programm was geändert werden müssen oder neu raufgeladen, werden speziell Bibliotheken dafür benötigt.



Abbildung: ESP32-Bibliothek

Wie schon vorher erwähnt werden beim ESP32-A die „asukiaaa“ wie auch die „esp32“ Bibliothek benötigt. Bei der „esp32“ Bibliothek ist die Bibliothek für ESP-Now inkludiert.

Sollte für die Datenkommunikation ein anderer Microcontroller verwendet werden, muss die Broadcastadresse (MAC-Adresse) des jeweiligen Microcontrollers im Programm geändert werden.

**ESP32-B:**

Der ESP32-B ist für das Einlesen der Daten von ESP32-A zuständig wie auch für das Senden der Daten über I²C an den Arduino UNO. Hierbei wird auch die „esp\_now.h“ Bibliothek benötigt. Wichtig hierbei ist auch, sollte ein andere Microcontroller verwendet werden, muss die Broadcastadresse wieder ausgetauscht werden. Für die Hardwareverbindung zwischen ESP32-B und Arduino UNO ist es notwendig die I²C Pins zu verbinden.

|  |  |
| --- | --- |
| ESP32-B | Arduino UNO |
| SDA-21 | SDA-A4 |
| SCL-22 | SCL-A5 |
| GND | GND |

Diese Tabelle zeigt die Pin-Belegung von den jeweiligen Microcontrollern, für die I²C Kommunikation.

**Arduino UNO:**

Der Arduino UNO ist dafür zuständig die Daten über I²C vom ESP32-B zu empfangen und diese dann auf ROS über Rosserial zu publishen.

Dafür notwendig ist es zuerst ein System mit ROS+Rosserial installiert zu haben. Dies kann man auf einem Raspberry Pi, Virtual Box oder auf einem PC machen. Dieses System ist für ROS-Melodic gemacht worden.

Tutorials zur Installation (Dies sollte der Reihe nach abgearbeitet werden):

<http://wiki.ros.org/melodic/Installation/Ubuntu> (ROS -Installation)

<http://wiki.ros.org/ROS/Tutorials/InstallingandConfiguringROSEnvironment> (Configuring ROS-Environment)

<http://wiki.ros.org/rosserial_arduino/Tutorials/Arduino%20IDE%20Setup> (Install Rosserial) Dieses Rosserial beinhaltet die Installation von Rosserial für Arduino DIE zugleich, sollte was beim Programm Code geändert werden müssen oder neu raufgeladen werden müssen, ist dies dafür notwendig.

.