

REVERSE ENGINEERING DER PHYBOX (UND ERSETZEN DES SYSTEMS)

VON

JULIAN NOEL BAUMANN

2023-04-20

Dieses Werk einschließlich seiner Teile ist **urheberrechtlich geschützt**. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Autors unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Einspeicherung und Verarbeitung in elektronischen Systemen.

Eidesstattliche Erklärung

Ich, Julian Noel Baumann, versichere hiermit, dass ich mein **Fachprojekt** mit dem Thema

Reverse Engineering der Phybox (und ersetzen des Systems)

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Rosenheim, den 2023-04-20

JULIAN NOEL BAUMANN

Open Source

Alles in diesem Projekt, einschließlich der Quelldateien dieser Dokumentation, ist als Open Source Projekt unter <https://github.com/julian-baumann/open-phybox> verfügbar.



Inhaltsverzeichnis

Abbildungsverzeichnis	I
Glossar	II
1 Einführung	1
1.1 Problemstellung	1
1.2 Zielsetzung des Projektes	2
1.3 Herangehensweisen	2
2 Reverse Engineering des Phybox Protokolls	3
3 Entwickeln einer neuen Phybox	7
3.1 Semantik der Open Phybox	9
3.2 Bluetooth Low Energy Kommunikation	10

Abbildungsverzeichnis

1 Ausschnitt aus der offiziellen Phybox Dokumentation der IBK electronic+informatic GmbH (IBK)	5
2 Ausschnitt aus der offiziellen Phybox Dokumentation der IBK	8



Glossar

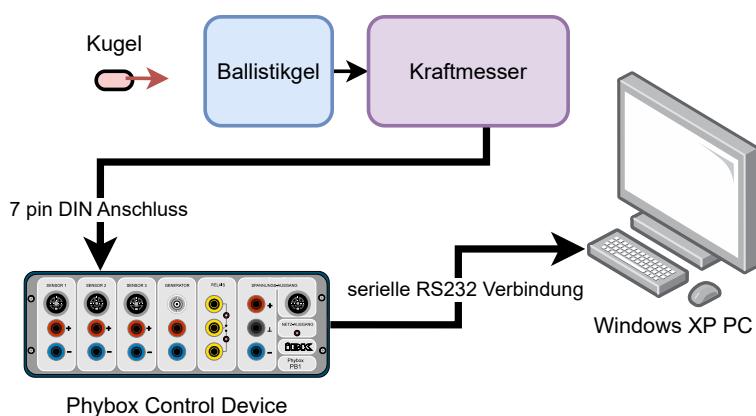
PC	Personal Computer
IBK	IBK electronic+informatic GmbH
ESP32	Microcontroller, entwickelt von Espressif Systems
ADC	Analog-to-Digital Converter
BLE	Bluetooth Low Energy



1 Einführung

1.1 Problemstellung

Zusammengefasst geht es in diesem Projekt darum, alte 90-er Jahre Technik eines Physikexperiments zu ersetzen. Das gewünschte Experiment dient dazu, die Kraft eines abgefeuerten Geschosses aus einer Pistole zu bestimmen.



Wie in der obigen Grafik dargestellt, trifft ein Geschoss auf ein ballistisches Gel, dessen Aufprall dann von einem Kraftmesser ausgelesen wird, dass das Signal an das Steuergerät "Phybox PB1" weiterleitet. Dieses Gerät verarbeitet das Signal des Messgeräts und sendet es über eine serielle RS-232-Verbindung an einen Windows XP-Computer. Auf dem PC läuft eine proprietäre Software namens "CATTSoft", die den gemessenen Stoß als Liniendiagramm auf dem Display anzeigt. Die Software wurde von denselben Leuten geschrieben, die auch das Phybox PB1-Gerät entwickelt haben, nämlich die IBK.

Das Problem bei diesem Versuchsaufbau ist, dass diese proprietäre Software nur auf einem Personal Computer (PC) mit Windows 3.1, 95, 98 oder Windows XP verfügbar ist. Die FOSBOS ROSENHEIM hat seit einigen Jahren Windows 10 PCs in allen Klassenzimmern im Einsatz und die Schüler verwenden hauptsächlich iPad-Geräte. Mit diesen neueren Geräten ist es möglich, das beschriebene Diagramm auf einen großen Bildschirm zu projizieren, der für alle Schüler sichtbar ist und auch die Möglichkeit bietet, die gemessenen Daten weiterzuverarbeiten.



1 Einführung

1.2 Zielsetzung des Projektes

Das **ultimative Ziel** dieses Projekts ist es, eine neue Software zu schreiben, die kompatibel mit dem Messgerät und dem Versuchsaufbau bleibt und die diese gemessenen Auswirkungen als Linendiagramm anzeigt und auf einem modernen Windows PC, iPadOS oder macOS Gerät läuft. Folglich wurden schon sehr früh in der Projektphase folgende Ziele definiert:

1. Einfache Benutzeroberfläche mit Null-Konfigurationsaufwand für den Nutzer
2. Moderne iPadOS und Windows App
3. Messwerte sollen über Bluetooth Low Energy (BLE) an das Endgerät geschickt werden
4. Kompatibel mit den originalen Phybox Messgeräten

1.3 Herangehensweisen

Um dieses Ziel zu erfüllen, wurde mit zwei verschiedene Ansätzen experimentiert:

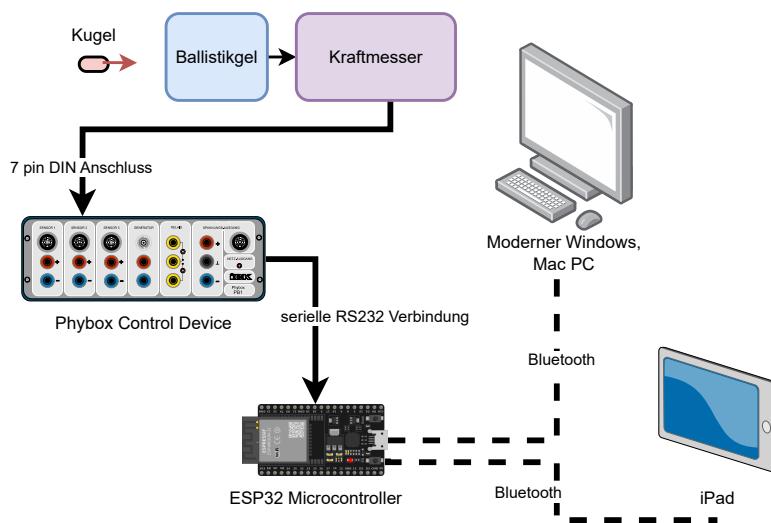
1. Reverse Engineering der seriellen Schnittstelle der Phybox
2. Ersetzen der kompletten Phybox durch einen ESP32

Letztendlich wurde sich für den letzteren Ansatz entschieden, da dieser mit weniger Aufwand verbunden ist und deutlich stabiliere Ergebnisse erzielen konnte.

Anschließend wird nun zuerst der erste Ansatz beschrieben, danach der zweite und zum Schluss wird noch die Entwicklung der iPadOS App angeschnitten.



2 Reverse Engineering des Phybox Protokolls



Bei diesem Ansatz soll ein ESP32 Mikrocontroller zwischengeschaltet werden, der die gemessenen und verarbeiteten Daten der Phybox über BLE an das gewünschte Endgerät weitergibt. Hierbei ist die Idee, dass der ESP32 die Daten der Phybox über die RS232 SCHNITTSTELLE entgegennimmt und sich so verhält wie die proprietäre CATTSoft Software. Somit wirkt es für die Phybox so, als wäre an ihr immer noch ein alter Windows XP PC angeschlossen. Es braucht somit keinerlei Modifizierungen an der Phybox selbst. Diese misst wie gewohnt das analoge Signal des Kraftmessers über den 7 PIN DIN ANSCHLUSS aus, bereitet diese Digital auf und schickt diese Messwerte über als digitales Signal über die RS232 SCHNITTSTELLE an den ESP32.

Das Problem ist, dass dieses serielle Protokoll der Phybox weder ein standardisiertes Protokoll ist, noch wurde je eine Entwicklerdokumentation dafür veröffentlicht. Eine Möglichkeit um dieses Problem zu lösen, bietet REVERSE ENGINEERING.

Bei (fast) jedem seriellen Protokoll werden im Grunde nur Datenbits übertragen. Wenn es uns nun gelingt herauszufinden, welche Datenpakete die Phybox bei einer bestimmten Anweisung erwartet, ist es uns folglich auch möglich ihr vorzugaukeln, dass am anderen Ende der seriellen Verbindung das CATTSoft Programm die Befehle sendet und entgegennimmt.

Um Herauszufinden, welche Datenpakete welche Bedeutung haben und auf welche Datenpakete wir antworten müssen, um die gewollten Daten zu kriegen, wurde eine virtuelle Maschine aufgesetzt, in der ein Windows XP System mit der installierten CATTSoft Software sitzt. Die seriellen Signale werden nun durch eine Sniffer-Software überwacht, bevor sie in die virtuelle Maschine weitergeleitet werden. Dies ermöglicht es uns genau nachzuvollziehen zu können, welche Daten von der Phybox gesendet werden, welche von der CATTSoft Software gesendet werden und wie die beiden Parteien miteinander interagieren.

REVERSE ENGINEERING DER PHYBOX

(und ersetzen des Systems)



2 Reverse Engineering des Phybox Protokolls

The screenshot displays the Free Device Monitoring Studio interface. On the left, there's a navigation bar with icons for File, Edit, View, Sessions, Bridge, Tools, Window, Help, and a toolbar with various functions like Open, Save, Print, and Export. Below the toolbar, tabs include USB Serial Port (COM1) - Data View, USB Serial Port (COM1) - Packet View, and USB Serial Port (COM1) - Console View. The main area shows two panes: 'Reads' and 'Writes'. The 'Reads' pane lists several hex dump entries, each with a timestamp, ID, and data. The 'Writes' pane also lists hex dump entries. A 'MODBUS Send' component is open on the right, showing session details like Mode (ASCII Mode), Address (0), Function (REQUESTS), and Result length. Below it is a terminal window with the instruction 'Status: Create a terminal session and select it in Session list box.' At the bottom, there are tabs for Selected Packet and MODBUS Send, and a Sessions list table with columns for Source, Type, Device, Processing, Start, Length, and State (which is 'Empty').

In dem obigen Screenshot ist die FREE DEVICE MONITORING SOFTWARE¹ zu sehen, mit der die serielle Verbindung zwischen der Phybox und dem CATTSoft Programm mitverfolgt wurde. In Rot werden alle Befehle visualisiert, die von der Phybox an das Computerprogramm gesendet werden, darunter in Blau sieht man alle Befehle welche die Software erwidert. Alle Befehle werden als Hexadezimalzahlen angezeigt. Diese muss man manuell vergleichen um zu sehen, ob sich bestimmte Muster ergeben, aus denen man Analysieren kann, was diese Bitfolgen repräsentieren sollen.

Allerdings zeigt dieses Programm die Read und Write Pakete getrennt voneinander und ziemlich unübersichtlich für unsere Zwecke da. Zum Glück erlaubt dieses Programm, die aufgezeichnete Kommunikation als Text-Datei zu exportieren. Als Nächstes wurde ein kleines Programm geschrieben, dass diese Dateien lesen und auswerten kann. Diese kleine Software zeigt die Read und Write Pakete chronologisch untereinander an, um den Verlauf der Kommunikation besser zu visualisieren. Außerdem erlaubt die Software auch mehrere Aufzeichnungen nebeneinander anzuzeigen und gemeinsame Pakete auszuwählen, die wiederum zum Synchronisieren der einzelnen Aufzeichnungen nützen. Dies erlaubt es uns die Kommunikation aus verschiedenen Aufzeichnungen zu vergleichen um bestimmte Muster besser erkennen zu können.

¹<https://freeserialanalyzer.com>

REVERSE ENGINEERING DER PHYBOX

(und ersetzen des Systems)



2 Reverse Engineering des Phybox Protokolls

Screenshot der Software:

The screenshot shows a list of network packets captured by the protocol analyzer. Each packet is represented by a card-like structure containing the following information:

- Timestamp:** The time when the packet was captured.
- ID:** A unique identifier for the packet.
- Raw Data:** The raw hex data of the packet, shown in two columns: bytes and ASCII.
- Details:** A small pane showing detailed information about the packet, such as source and destination MAC addresses, port numbers, and protocols.

The list includes entries from various dates, such as 2022-12-28 and 2023-01-04, with IDs ranging from 006806 to 07673. The raw data section shows typical network traffic patterns like ARP requests and responses.

Abbildung 1: Ausschnitt aus der offiziellen Phybox Dokumentation der IBK

Diese Software ist als Website unter <https://tools.juba.dev/protocol-analyzer> zu finden. (Source Code: <https://github.com/julian-baumann/serial-protocol-analyzer>)

Trotzdem ist die Analyse des Protokolls immer noch sehr mühsam. Man vergleicht hunderte dieser Datenpakete, um Muster zu finden, um herauszufinden, was diese wohl bedeuten. Es reicht nicht, einfach dieselben Pakete zurückzusenden. Es ist auch wichtig, dass diese Pakete zur richtigen Zeit und in der richtigen Reihenfolge gesendet werden. Oft kann nicht genau gesagt werden, ob die Antworten der CATTSoft Software überhaupt mit den Paketen der Phybox zusammenhängen, oder ob diese nur nach einer zeitlichen Periode gesendet werden. Auch ist oft unschlüssig, was diese Pakete bedeuten. Vielleicht wird anfangs die Seriennummer übertragen, vielleicht werden aber auch wichtige Konfigurationen an die Phybox gesendet, die Zeitabhängig sind und sich in Zukunft ändern.

REVERSE ENGINEERING DER PHYBOX

(und ersetzen des Systems)



2 Reverse Engineering des Phybox Protokolls

Dennoch gelang es einige Komponenten der Kommunikation teilweise zu entschlüsseln und durch ein C# Programm nachzubilden. Zum Beispiel sendet die Phybox mit der Baud-Rate 57600 kontinuierlich die folgenden Bytes:

```
0x50 0x68 0x79 0x62 0x6F 0x78 0x20 0x50 0x42 0x31
```

Diese repräsentieren die ASCII Zeichenkette "Phybox PB1", welche von der Phybox solange gesendet werden, bis dann das andere Ende mit folgender Bytefolge antwortet:

```
0x8B 0x0F 0x17 0x23 0x41
```

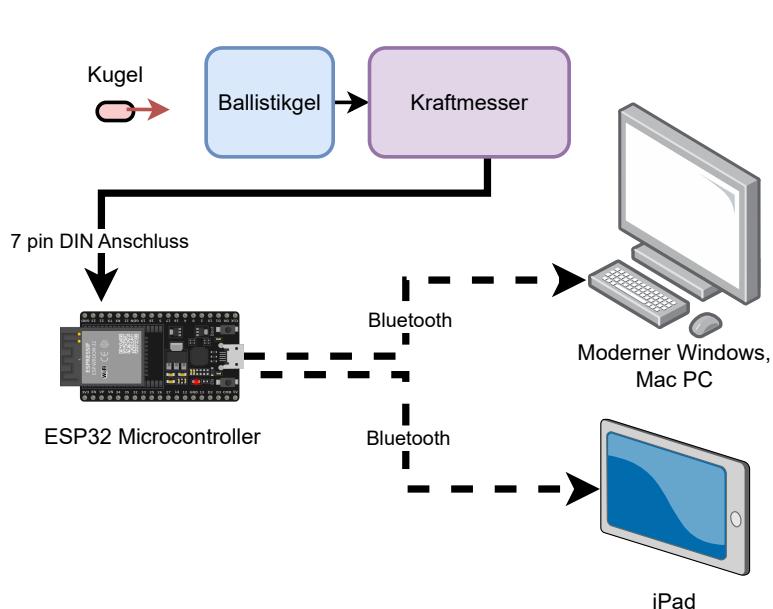
Diese repräsentieren jedoch keine logische ASCII Zeichenkette mehr, noch konnte ermittelt werden, was diese Bytefolge bedeutet. Die Wahrscheinlichkeit ist gering, aber es ist natürlich auch möglich, dass diese Bytefolge einfach zufällig gewählt wurde. Nachdem der Client diese Bytes an die Phybox gesendet hat, hört die Phybox auf, kontinuierlich "Phybox PB1" zu senden und initiiert den Kommunikationsfluss. Es scheint auch, als würde ein Timer laufen, der abwechselnd vom Client und der Phybox aktualisiert wird, um sicherzustellen, dass der gegenteilige Partner noch aktiv an der Kommunikation teilnimmt.

Nach einiger Zeit wurde klar, dass diese Art eher sehr mühsam ist und zu ungenügenden Ergebnissen zielt, da nicht genau ermittelt werden konnte, was die einzelnen Bytefolgen bedeuten sollen. Mit mehr Zeit wäre dieser Ansatz vielleicht durchaus sinnvoll und umsetzbar, jedoch war dieser für dieses Projekt viel zu Zeitaufwändig.

Aus diesem Grund wurde ein anderer Ansatz gewählt, der im folgenden beschrieben wird.



3 Entwickeln einer neuen Phybox



Anstatt das digitale serielle Signal der Phybox abzugreifen, wird in diesem Ansatz gleich das analoge Signal des Messgeräts genutzt. Der Kraftmesser wird hierbei direkt über den 7 Pin DIN Anschluss an den ESP32 angeschlossen. Dieser misst das analoge Signal durch einen Analog-to-Digital Converter (ADC). Mit dieser Methode ist es uns also möglich, die Phybox komplett zu ersetzen und ein neues System einzusetzen, dass die Messung selbst interpretiert. Der glückliche Vorteil hierbei ist, dass das Messgerät ein analoges Signal ausgibt, das in der offiziellen Dokumentation detailliert beschrieben wird. Mit dem ESP32 können wir dann kontinuierlich eine Messung vornehmen und die Ergebnisse über BLE an das Endgerät schicken.



3 Entwickeln einer neuen Phybox

IBK
electronic + informatic

5.3 Kraft-Sensor

Bestellbezeichnung: DKS Kraft-Sensor

Der Kraftsensor beruht auf dem Biegebalken-Prinzip. Die Last wird mit einem kleinen Haken an den 10mm breiten und 30mm aus dem Gehäuse ragenden Federstahl-Streifen angebracht und über die Dehnung dieses Streifens elektronisch erfaßt. Über Gewindewürfel und Stativstange ist die universelle Befestigung des Kraftsensors gewährleistet. Der Anschluß erfolgt über ein siebenpoliges DIN-Kabel an einem der Sensor-Eingänge der Phybox.

Anwendungsgebiete

Mechanikversuche z.B. Hooksches Gesetz, Messung der Federkräfte bei einem Federpendel (F-t Diagramm), Dehnung von Drähten (elastische, plastische Verformung) etc.

Technische Daten

Bereich: 0 ... +10N,
Auflösung: 0,01N,
Ausgang: 1V/N +2%

Buchsenbelegung:

Pin-Nr. Belegung

1	+ Ausgangssignal (1V/N)
2	Masse
3	nicht belegt
4	Versorgung (-15V)
5	Versorgung (+15V)
6	- Ausgangssignal (Masse)
7	nicht belegt

Frontansicht DIN-Buchse

The diagram shows a circular 7-pin DIN connector. The pins are numbered as follows: Pin 1 is at the bottom, Pin 2 is at the top, Pin 3 is at the bottom-left, Pin 4 is at the top-left, Pin 5 is at the top-right, Pin 6 is at the bottom-right, and Pin 7 is at the top-right, overlapping with Pin 5. Each pin has a small square symbol next to it.

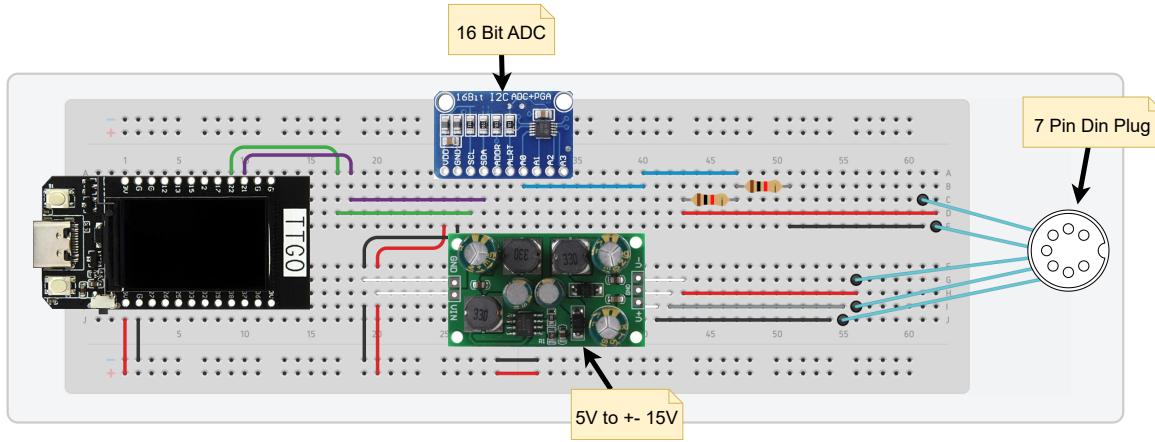
Abbildung 2: Ausschnitt aus der offiziellen Phybox Dokumentation der IBK

In der obigen Abbildung ist zu erkennen, dass in der Dokumentation nicht nur beschrieben steht, wie genau der Sensor funktioniert, sondern auch die Belegung des 7 Pin DIN Steckers offengelegt wird. Es kann außerdem herausgelesen werden, dass der Kraft-Sensor eine Stromversorgung von $\pm 15V$ benötigt. Das Messgerät gibt folglich je nachdem wie stark der Federstahl-Streifen nach vorne oder hinten verbogen wird eine Spannung von $\pm 15V$ zurück. Ein Volt repräsentiert hierbei ein Newton. In der Dokumentation kann aus den technischen Daten jedoch herausgelesen werden, dass der Messbereich nur von 0 bis +10N definiert ist. Dies ist durchaus nachvollziehbar, denn eine negative Messung macht in den seltensten Fällen Sinn und in den meisten Fällen wird eine Dehnung des Federstahl-Streifens in nur eine Richtung gebraucht. Dennoch wurde durch Experimente am Oszilloskop herausgefunden, dass das Messgerät aus technischer Sicht dennoch eine $\pm 15V$ Spannung liefert, auch wenn in der Software dann nur der Messbereich von 0 bis +10V berücksichtigt wird.



3 Entwickeln einer neuen Phybox

3.1 Semantik der Open Phybox



Wie schon aus der IBK Dokumentation entnommen werden kann, benötigt der Kraftmesser eine Eingangsspannung von $\pm 15V$. Um dies zu gewährleisten wird die 5V Ausgangsspannung des ESP32 durch einen BUCK CONVERTER auf die gewünschten $\pm 15V$ gebracht. Diese Spannung wird nun über den 7 PIN DIN Stecker an das Messgerät weitergegeben, welches wiederum ebenfalls ein analoges Messsignal zwischen $\pm 15V$ zurückliefert. Die maximale Messspannung des ADCs entspricht der gegebenen Eingangsspannung von 5V. Deswegen wird das Messsignal durch einen Spannungsteiler auf ungefähr $\pm 3V$ runtergebrochen. Der externe ADC gibt die gemessene Spannung als digitales Signal über eine I²C Verbindung an den ESP32 weiter.

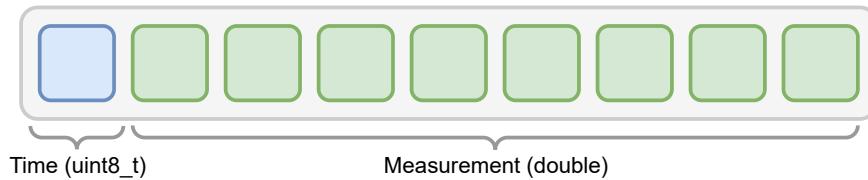
Der ESP32 führt alle 10 Millisekunden eine analoge Messung mittels ADC aus. Diese Messung wird präzise über einen System-Interrupt ausgeführt. Diese Messdaten werden zusammen mit dem Zeitpunkt der Messung in einer Liste gespeichert, die jede Sekunde über BLE an das Endgerät geschickt wird.



3 Entwickeln einer neuen Phybox

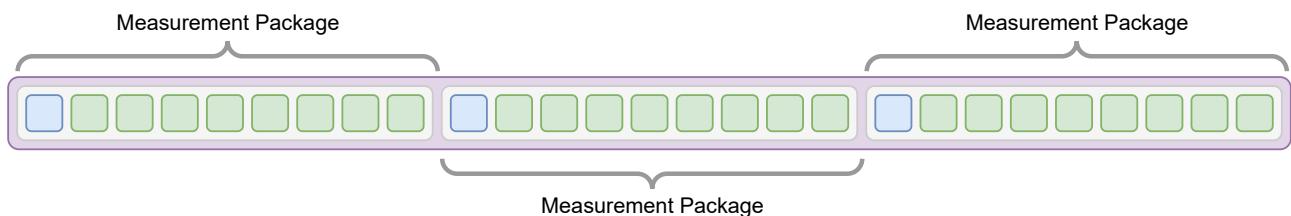
3.2 Bluetooth Low Energy Kommunikation

Die Bluetoothkommunikation gestaltet sich ziemlich einfach. Dies wurde mit Absicht so implementiert, um soviel Daten wie möglich an mehrere Geräte so schnell wie möglich zu senden. Eine Messung beinhaltet wie zuvor schon erwähnt den Zeitpunkt der Messung und die Messung an sich. Diese ist ein DOUBLE, dessen Wert die Spannung in Volt widerspiegelt. Die folgende Abbildung verdeutlicht dies noch einmal.



Wie aus der obigen Abbildung zu entnehmen ist, ist TIME ein 8-Bit unsigned Integer. Dieser enthält nicht den vollen Zeitpunkt der Messung, sondern nur die Differenz zur letzten Messung. Diese sollte im Idealfall immer 10 MILLISEKUNDEN sein, da der Messinterrupt immer im 10 Millisekudentakt feuern sollte. Trotzdem enthält jede Messung diese Zeitdifferenz um noch ein mal sicherstellen zu können, dass die Messungen richtig interpretiert werden können.

Da alle 10 MILLISEKUNDEN eine Messung aufgenommen wird, diese aber nur JEDER SEKUNDE über BLE übertragen werden, sammeln sich diese Messungen an und werden dann anschließend alle gemeinsam in einem Schwung übertragen.



Die Messungen werden einfach aneinander Konkateniert. Die Daten, die über Bluetooth gesendet werden bestehen, also aus einem ARRAY, indem sich ein STRUCT mit einem 8-BIT UNSIGNED INTEGER und einem DOUBLE befindet.