

**Bachelorarbeit**

Vorgelegt von

Ngahan nyonse Francois Brice

am. 24.Mai.2020

Studiengang Biomedical Engineering Bachelor of Ingenieur

**Kommunikationsprotokoll zwischen einem Computer und einer**

**Mikrocontroller Des Typs Pyboard**

Fachbereich 9 – Medizintechnik & Technomathematik

Fachhochschule Aachen, Campus Jülich

Diese Arbeit ist von mir selbständig angefertigt und verfasst. Es sind keine anderen als

die angegebenen Quellen und Hilfsmittel benutzt worden.

Unterschrift…...……….…………………….

1. **Prüfer:** Prof. Dr. -Ing Torsten Wagner
2. **Prüfer:** Prof. Dr.

# Danksagung

An dieser Stelle bedanke ich mich bei allen, die mich während der Bearbeitung der Bachelorarbeit unterstützt haben.

Insbesondere bedanke ich mich bei meinem Betreuer **Herrn Prof. Dr. Ing- Torsten Wagner** für sein Hilfreiche Unterstützung und seine Anregungen bei der Bearbeitung. Ferner möchte ich Herr \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ danken für seinen Zuspruch und Rat.

Zusätzlich möchte ich mich ganz herzlich bei meinem Freund **Donald Kapyouo** bedanken für seine stetige Anteilnahme an meiner Arbeit.

Darüber hinaus bedanke ich mich bei meiner Gast Familie **Tayoumo** und meiner ganzen Familie, die immer für mich da sind, bei meinen Eltern für ihre Unterstützung und ihr guter Rat.

# Inhaltsverzeichnis

[Danksagung III](#_Toc44755920)

[Inhaltsverzeichnis IV](#_Toc44755921)

[Abbildungsverzeichnis V](#_Toc44755922)

[Tabellenverzeichnis V](#_Toc44755923)

[Abkürzungsverzeichnis VI](#_Toc44755924)

[1 Einleitung 1](#_Toc44755925)

[2 Grundlagen und Definition 2](#_Toc44755926)

[2.1 Mikrocontroller 2](#_Toc44755927)

[2.1.1 Eigenschaften eines Mikrocontrollers 2](#_Toc44755928)

[2.1.2 Einsetzung von Mikrocontroller 2](#_Toc44755929)

[2.1.3 Funktionsweise Mikrocontroller 3](#_Toc44755930)

[2.2 Mikroprozessor 5](#_Toc44755931)

[2.3 Mikropython (µC) 6](#_Toc44755932)

[2.3.1 Mikropython Devices 7](#_Toc44755933)

[2.4 Der Pyboard 10](#_Toc44755934)

[ Definition des Pyboard & Hardware. 10](#_Toc44755935)

[(siehe 2.2.2 i) 10](#_Toc44755936)

[3 Module und Klassen 13](#_Toc44755937)

[3.1 Das Module Pyserial & Class Serial 13](#_Toc44755938)

[3.2 Historische Einordnung 15](#_Toc44755939)

[3.3 Quellenlage 15](#_Toc44755940)

[4 Hauptteil 2: Grundlagen und Methodik 16](#_Toc44755941)

[4.1 Theoretischer Bezugsrahmen 16](#_Toc44755942)

[4.2 Methodische Vorgehensweise 16](#_Toc44755943)

[5 Hauptteil 3: Empirische Untersuchung 17](#_Toc44755944)

[5.1 Analysen 17](#_Toc44755945)

[5.2 Interpretation der Ergebnisse 17](#_Toc44755946)

[6 Zusammenfassung 18](#_Toc44755947)

[Literaturverzeichnis VII](#_Toc44755948)

[Anhänge IX](#_Toc44755949)

[Anhang 1 IX](#_Toc44755950)

[Anhang 2 IX](#_Toc44755951)

[Eidesstattliche Versicherung X](#_Toc44755952)

[Stichwortverzeichnis XI](#_Toc44755953)

# Abbildungsverzeichnis

Abbildung 1: Aufbau eines Mikrocontroller 4

Abbildung 2 Erläuterung der Begriffe Mikroprozessor, Mikrorechner-/System 6

Abbildung 3 Das erste und Original Pyboard 7

Abbildung 4 BBC Micro: Bit 9

Abbildung 5 Die Adafruit-Feder HUZZAH-Tafel (Bildnachweis: Adafruit). 9

Abbildung 6: Das Espressif ESP32 Development Board (Bildnachweis: Adafruit). 10

# Tabellenverzeichnis

Tabelle 1: Beispieltabelle mit einer besonders langen Tabellenüberschrift, um den Umbruch des Textes darzustellen **Fehler! Textmarke nicht definiert.**

# Abkürzungsverzeichnis

Enthält die weniger gebräuchlichen Abkürzungen und fachsprachlichen Abkürzungen in alphabetischer Reihenfolge mit Auflösung. Ein Abkürzungsverzeichnis ist bei dem häufigeren Gebrauch von Abkürzungen üblich.

Diese Abkürzungen gehören nicht in das Abkürzungsverzeichnis, weil allgemein üblich:

# Einleitung

Damit mein Mobiltelefon funktioniert, muss es mit einem Mikrocontroller ausgestattet sein, der das gesamte Gerät steuern kann. Und dies gilt für fast alle elektronischen Geräte, die wir in unserem Alltag benutzen. Seit dem Aufkommen neuer Technologien in der Welt, vieles hat sich geändert. Heutzutage sind elektronische Geräte, die früher groß waren, sehr klein geworden, und der Mikrocontroller, der zu ihrer Herstellung verwendet wurde, wird Tag für Tag extrem klein wie zum Beispiel das Pyboard. Apropos Pyboard, dies wird das wichtigste Element meiner Arbeit sein. Das Pyboard ist ein Mikrochip, der zur Familie der Mikrocontroller gehört und mit Mikropython-Programmierung arbeitet.

(1) Diese Arbeit beschäftigt sich mit der asynchronen Kommunikation, und Gegenstand dieser Untersuchung ist das Kommunikationsprotokoll zwischen einem µC vom Typ "Pyboard" und einem Computer. In dieser Arbeit wird ein µPy-basiertes Board programmiert, damit die Analogen Ein- Ausgänge angesteuert werden können.

(2) Dann muss eine Kommunikation zwischen PC und Mikrocontroller über USB aufgebaut werden.

Ansatzpunkt für diese Arbeit ist, dass die ersten und zweiten oben genannten Punkte voneinander entkoppelt werden können, so dass sie asynchron arbeiten können. Das bedeutet, dass µC auch während der Messung weiterhin Befehle empfangen und Daten senden muss, und solange ein Befehl empfangen wird, darf er die Messungen nicht verändern.

Ziel dieser Arbeit ist es, eine Ausarbeitung eines Kommunikationsprotokoll, also eine Reihe von Befehlen die an den Mikrocontroller verschickt werden können, die diesen verstehen und abarbeiten, ohne dass irgendwelche Befehle gestört oder unterbrochen wird.

Damit dieses Projekt erfolgreich durchführen zu können, muss ich das Module "Serial" für die Kommunikation zwischen meinem Computer und meinem Pyboard verwenden. Und um mehrere Befehlen parallel verarbeiten zu können, ohne sich gegenseitig zu unterbrechen, muss ich das Uasyncio-Modul verwenden.

# Grundlagen und Definition

## Mikrocontroller

Mikrocontroller sind spezielle Mikrocomputer auf einem Chip, die an bestimmte Anwendungen angepasst sind. Dabei handelt es sich in der Regel um Steuer- oder Kommunikationsaufgaben, die einmal programmiert und dann während der Lebensdauer des Mikrocontrollers auf diesem ausgeführt werden. Die Anwendungsbereiche sind sehr breit gefächert. Mikrocontroller arbeiten normalerweise unsichtbar in einer Vielzahl von Geräten und Systemen, die uns in unserem Alltag umgeben.

### Eigenschaften eines Mikrocontrollers

Ein Mikrocontroller hat die folgenden typischen Merkmale:

* Besitz einen Programmspeicher (früher ROM, EPROM, EEPROM, heute FLASCH)
* Ein Datenspeicher (RAM)
* Eine Verarbeitungseinheit (CPU)
* digitale Eingabe- / Ausgabe- Ports (GPIO), oft analoge Eingabegeräte (ADC)
* einen oder mehrere Zeitgeber
* Kommunikationsbausteine (COM, UART,…)

### Einsetzung von Mikrocontroller

Mikrocontroller sind häufig in eingebetteten Systemen zu finden. Eingebettete Systeme sind in einen technischen Kontext integrierte Computer/elektronische Rechner (embedded = eingebettet). Die Aufgabe des Bordcomputers ist es, das System zu steuern, zu regeln oder zu überwachen. Es kann auch zur Daten- und Signalverarbeitung verwendet werden und verantwortlich sein. Zum Beispiel Ver.-/Entschlüsselung, Decodierung oder Filterung.[[1]](#footnote-1)

Obwohl Mikrocontroller in fast allen Bereichen unserer Gesellschaft vorkommen, werde ich die wichtigsten Bereiche auflisten, in denen wir mit Mikrocontrollern zu tun haben

Hier sind einige Beispiele für Aufgaben, für die Mikrocontroller eingesetzt werden können:

* In Geräten der Medizintechnik
* Steuerung von Infusionsgeräten,
* Kardiovaskulären Monitoren,
* Atemschutzgeräte,
* Dialysegeräte,
* Im Haushalt
* Fernsehers,
* Staubsaugers,
* Waschmaschine,
* Kaffeemaschine,
* Etc…
* In Kraftfahrzeugs Technik
* Das Motorsteuergerät,
* Das Stabilitätsprogramm,
* Verschiedene Assistenten, z.B. beim Bremsen,
* Antriebskontrolle,
* In der Automatisierungstechnik
* Prozesssteuerung und -regelung,
* Materialflusssteuerung,
* Produktionsanlagen,
* Das Überwachen von Prozessen

### Funktionsweise Mikrocontroller

Die Funktionalitäten der Mikrocontroller sind vielfältig, aber ich werde mich auf die wichtigsten konzentrieren.

Der Mikrocontroller besteht aus:

* Einer Zentraleinheit:

ist eine zentrale Verarbeitungseinheit (CPU) der Teil eines Computersystems, der die anderen Teile des Systems steuert und verwaltet.

* Einem SRAM:

einen Informationsspeicher, der beschreibbar, löschbar und wieder lesbar ist

* Einem EEPROM:

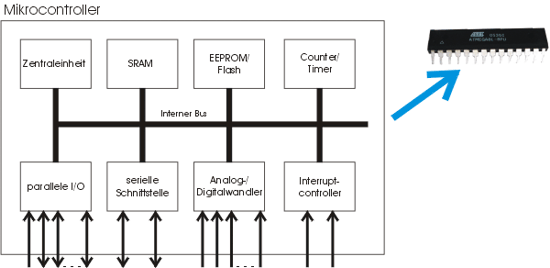
können jederzeit beliebig oft beschrieben und entfernt werden und bewahren ihren Zustand auch nach Ausschalten der Versorgungsspannung.

* Einem Flash:

sind sehr performante EEPROMs, mit denen der Controller in das Zielsystem programmierbar ist.

* Seriellen Schnittstelle:

ist eine Beschreibung für eine Datenübertragungsstrecke von zwei Geräten, bei der die Einzelbits nacheinander transportiert werden.

* Zeitgeber.[[2]](#footnote-2)
* Abbildung 1: Aufbau eines Mikrocontroller

Aus Peripheriebaustein können z.B. sein:

* Digitale Ein- und Ausgabebausteine:

Die werden auch Ports genannt und dienen zum Lesen oder Erzeugen von Ein-/Aus-Zuständen, Schaltvorgängen, Impulse oder Frequenzen.

* Komparator:

Die können " irgendwelche" Spannungen miteinander vergleichen.

* A/D-Wandler, D/A-Wandler:

kann einen "beliebigen" analogen Spannungswert in einen Zahlenwert umwandeln und als ganzzahlige Zahl für die weitere Verarbeitung bereitstellen.

* Zähler(„Timer“) -Baustein:

für zeitabhängige Steuerungen, die notwendig sind, wenn man beispielsweise die Uhrzeit ausgeben möchte

* Sender- / Empfänger-Baustein:

man kann größere Datenmengen übertragen, typische Schnittstellenbausteine sind UART, SPI, IÂ²C-Bus oder CAN-Bus.

* Interrupt Controller-Baustein:

dienen dazu, kurze Reaktionszeiten des Mikrocontrollers zu erhalten oder laufende Programme durch ein Interrupt zu unterbrechen.[[3]](#footnote-3)

## Mikroprozessor

heutzutage kennen viele Leute nicht den Unterschied zwischen einem Mikrocontroller und einem Mikroprozessor, deshalb wird es für mich wichtig sein, zu definieren, was ein Mikroprozessor ist.

Ein Mikroprozessor ist die zentrale Steuereinheit (CPU) eines Datenverarbeitungssystems, die heute in der Regel zusammen mit anderen Bauelementen auf einem einzigen Mikrochip eingebaut ist. Sie besteht in der Regel aus einer Bedieneinheit und einer arithmetischen Einheit, auch Verfahrenskern genannt, sowie einer Schnittstelle zur Außenwelt. Je nach Bedarf und Leistung können weitere Verarbeitungskomponenten wie Cache-Speicher und virtuelle Speicherverwaltung hinzugefügt werden. Die Tätigkeit eines Mikroprozessors ist die Bearbeitung eines Programms, das aus einer Folge von Befehlen zur Bearbeitung einer Anwendung besteht. Zu diesem Zweck muss der Mikroprozessor auch alle anderen Bestandteile der Datenverarbeitungseinrichtung steuern, wie z.B. Speicher und Ein-/Ausgabeschnittstellen4.[[4]](#footnote-4)4

* Ein Mikroprozessorsystem ist ein technisches System, das einen Mikroprozessor enthält. Dabei muss es sich nicht um einen Rechner oder eine Kaffeemaschine handeln, die von einem Mikroprozessor geregelt wird, sondern um ein Mikroprozessorsystem.
* Ein Mikrocomputer ist ein Computer oder eine Maschine, deren zentrale Verarbeitungseinheit aus einem oder mehreren Mikroprozessoren besteht. Zusätzlich zu dem/den Mikroprozessor(en) enthält ein Mikrocomputer einen Speicher, Ein-/Ausgabeschnittstellen und ein Anschlußsystem.
* Ein Mikrocomputersystem ist ein Mikrocomputer mit angeschlossener Peripherie an den Ein-/Ausgabeschnittstellen, z.B. einem Keyboard, einer Maus, einem Computerbildschirm, einem Printer oder ähnliches.

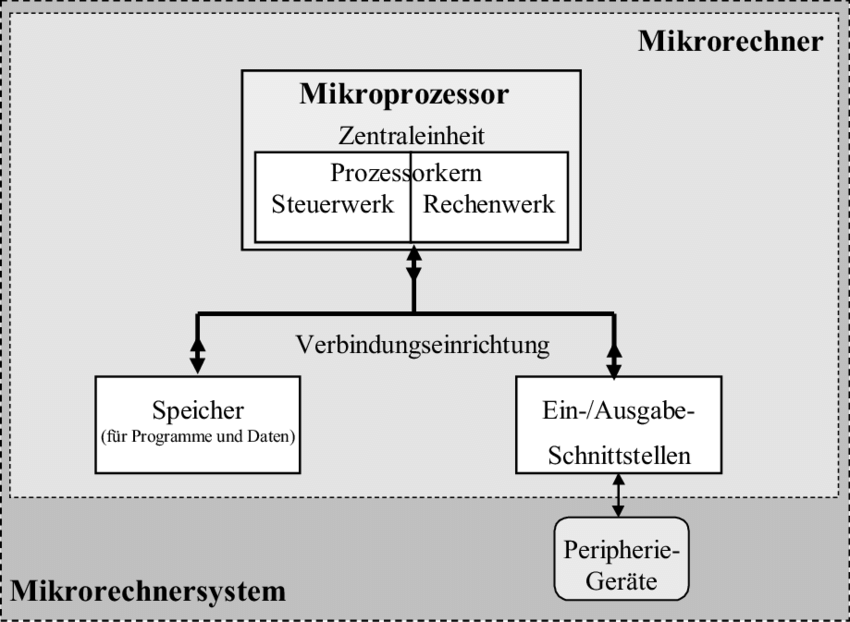
x

Abbildung 2 Erläuterung der Begriffe Mikroprozessor, Mikrorechner-/System[[5]](#footnote-5)6

### 2.3 Mikropython (µC)

µC ist eine Neuimplementierung von Python 3 für Mikrocontroller und eingebettete Systeme. Es ist auch eine erstaunliche Ingenieursleistung, um eine Neuimplementierung von Python zu haben, die unter solchen eingeschränkten Bedingungen funktioniert. es bringt eine der schönsten, leicht erlernbarsten und ausdrucksstärksten Programmiersprachen in die Welt der eingebetteten Entwicklung.[[6]](#footnote-6)5

Mikropython ist vollgepackt mit fortschrittlichen Funktionen wie interaktive Eingabeaufforderungen, Ganzzahlen beliebiger Genauigkeit, Schließungen, Listenverständnis, Generatoren, Ausnahmebehandlung und mehr. Dennoch ist er so kompakt, dass er mit nur 256k Code und 16k RAM passt und funktioniert.

Mikropython versucht, so kompatibel wie möglich zu regulärem Python zu sein, um Ihnen die einfache Übertragung von Code vom Desktop auf einen Mikrocontroller oder ein eingebettetes System zu ermöglichen.[[7]](#footnote-7)7

### 2.3.1 Mikropython Devices

Ich kann nicht über Mikropython sprechen, ohne die 4 Hauptmikrocontroller zu erwähnen, mit denen er arbeiten könnte.

Ich würde natürlich mit dem Hauptmikrocontroller beginnen, der unser Arbeitswerkzeug ist.

1. **Der Pyboard**

Das Pyboard ist ein Gerät, das für Mikropython entwickelt und gebaut wurde (siehe Abbildung 3).

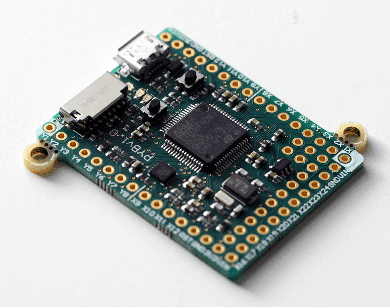


Abbildung 3 Das erste und Original Pyboard

Das Board wird über ein Mikro-USB-Kabel mit dem PC verbunden. Diese Verbindung bietet zwei Möglichkeiten, mit den Geräten zu interagieren: als USB-Flash-Laufwerk und als seriell-basiertes Python REPL[[8]](#footnote-8)8.

Die Platine enthält ein kleines Dateisystem als Teil ihres Flash-Speichers (on Chip). Wenn wir auf das Pyboard als Flash-Speicher zugreifen, können wir Dateien auf und vom Dateisystem des Boards kopieren. Wenn wir ein Python-Skript namens main.py auf dem Dateisystem des Boards erstellen, führt Mikropython dieses Skript beim Start aus. Folglich kann das Skript auf dem Board ausgeführt werden, ohne dass es mit einem PC verbunden sein muss.

Genau wie normales Python können wir die anderen Pyboard-Dateien kopieren, sie in main.py importieren und Ihren Code in entsprechende Module organisieren.

Einmal physikalisch verbunden, ist es auch möglich, jedes serielle Programm zu verwenden, um sich mit dem Pyboard zu verbinden und Mikropython REPL-Prompt zu erhalten. Und um sie sofort auszuführen und auszuwerten, müssen wir Befehle in das REPL eingeben.

* **Der Hardware**

Das Pyboard ist in vielerlei Hinsicht das leistungsfähigste Board, das in dieser Arbeit verwendet wird. Es verfügt über einen STM32F405RG Mikrocontroller, eine 168 MHz Cortex M4 CPU mit Hardware-Fließkomma, 1.024 KB Flash-ROM und 192 KB RAM.

Die Verbindung mit Peripheriegeräten und anderen Komponenten erfolgt über die GPIO-Pins, die um drei Kanten der Platine herum verlaufen.

Zusätzlich zum Mikro-USB-Steckplatz kann das Pyboard mit Batterien oder einer anderen Stromquelle betrieben werden. Schließen Sie die positive Leitung der Stromversorgung an den mit VIN gekennzeichneten Port an und erden Sie sie an GND. Die Eingangsspannung muss zwischen 3,6V und 10V liegen.[[9]](#footnote-9)9

Zu den weiteren Peripheriegeräten gehören ein Bluetooth-Modul, Prototyp-Skins (auf denen Testschaltungen aufgebaut werden können) und verschiedene Arten von Servomotoren zur Herstellung von Dingen, die sich bewegen.

Es war für mich ein konkretes Problem, zu erklären, dass Pyboard unser Hauptmaterial ist. Ich werde einfach versuchen, die anderen Geräte aufzulisten, ohne viele Details zu nennen.

1. **BBC Micro: Bit**

Das BBC Micro: Bit ist ein einfaches und dennoch leistungsstarkes Rechengerät für Programmieranfänger. Das Gerät wurde von der BBC (British Broadcasting Corporation) entwickelt, um die digitale Kreativität zu fördern. Es ist klein, billig und einfach zu bedienen.



Abbildung 4 BBC Micro: Bit[[10]](#footnote-10)10

1. **ESP8266**

ESP8266 ist ein beliebtes WiFi-fähiges System-on-Chip (SoC) von Expressif Systems.

Es gibt eine Vielzahl von Modulen und Platinen aus verschiedenen Quellen, die den ESP8266-Chip tragen.

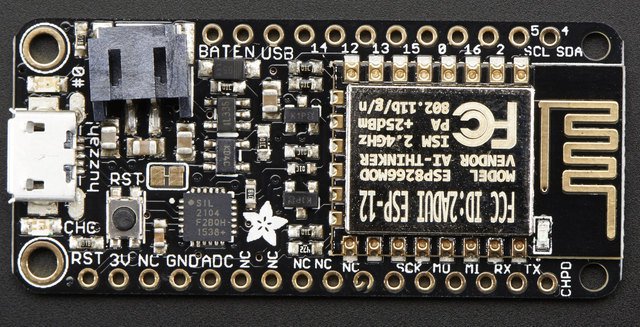


Abbildung 5 Die Adafruit-Feder HUZZAH-Tafel (Bildnachweis: Adafruit).

1. **ESP32**

Wie das ESP8266, das ESP32 ist ein beliebtes WiFi- und Bluetooth-fähiges System-on-Chip (SoC) von Expressif Systems.

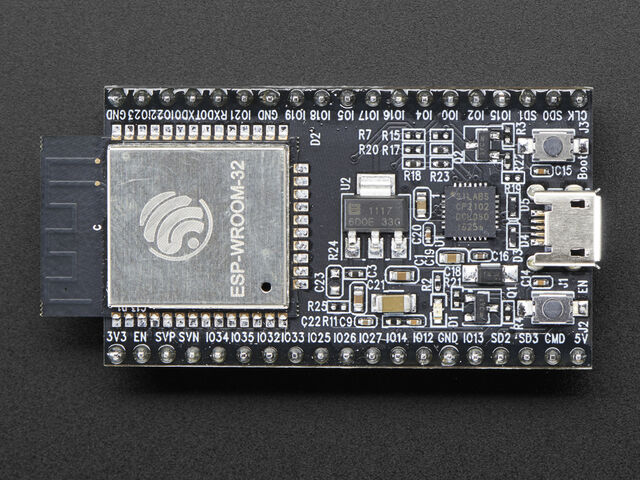


Abbildung 6: Das Espressif ESP32 Development Board (Bildnachweis: Adafruit).

## 2.4 Der Pyboard

## Definition des Pyboard & Hardware.

## (siehe 2.2.2 i)

* **Entwickler-Setup**

Im Pyboard-paket[[11]](#footnote-11)11 ist Mikropython bereits auf das Gerät geflasht. Es kann jedoch sein, dass dies nicht die neueste Version ist, die für das Board verfügbar ist, daher ist es immer eine gute Idee, das Gerät neu zu flashen. Es ist kein schwieriger Prozess, und es stellt sicher, dass Sie Zugang zu den neuesten Fehlerbehebungen und Updates erhalten.

Laden Sie zunächst die neueste Firmware von der Mikropython-Website (https://micropython.org/download/) herunter. Zweitens müssen Sie alles von Ihrem Pyboard trennen und den DFU-Pin (Device Firmware Update) mit dem 3,3-V-Pin verbinden. Sie liegen direkt nebeneinander, und wenn Sie eine Lupe haben, sind sie auch auf der Rückseite der Platine beschriftet. Sie sind auf dem Foto auf der Vorderseite der Platine hervorgehoben.

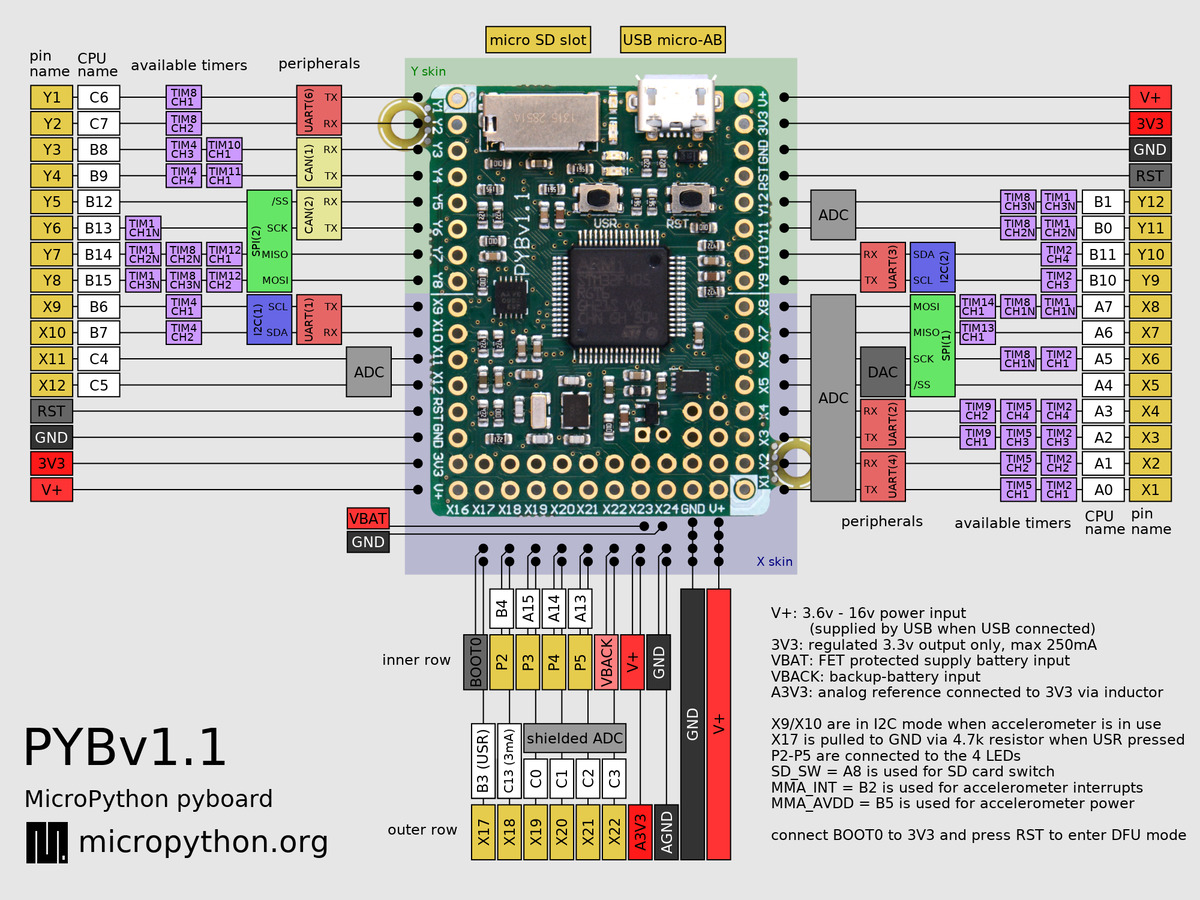


Abbildung 7 Pin-belegung für PYBv1.1

Die Verbindung der Pins lässt sich am besten mit einem Stecker-zu-Stecker-Brückendraht herstellen.

Für die Aktualisierung der Firmware auf dem Pyboard ist ein DFU-Dienstprogramm erforderlich. Es gibt zwei Optionen: dfu-util oder pydfu. Das pydfu-Skript kann von Mikropython‘s Github-Repository (https://github.com/micropython/micropython/blob/master/tools/pydfu.py) heruntergeladen werden.

Verbinden Sie das Pyboard über das USB-Kabel mit Ihrem Computer. Geben Sie für dfu-util den folgenden Befehl ein:

" sudo dfu - util --alt 0 -D firmware.dfu".

Wir müssen die Datei „Firmware.dfu“ durch die neueste Datei ersetzen, die Sie von der Mikropython-Website heruntergeladen haben.

um pydfu zum Aktualisieren der Firmware zu verwenden, verwenden Sie den folgenden Befehl:

"s sudo python pydfu.py -u firmware.dfu"

Nachdem die Firmware aktualisiert wurde, trennen Sie das Pyboard von Ihrem Computer und entfernen Sie das Überbrückungskabel zwischen den DFU- und 3,3-V-Pins.

Wenn das Gerät als abnehmbares USB-Flash-Laufwerk erscheint, wird es zunächst drei Dateien enthalten:

**boot.py**

Wird beim Start des Geräts ausgeführt und richtet verschiedene Konfigurationsoptionen ein.

**main.py**

Das Hauptskript, das Ihren Code enthält. Es wird unmittelbar nach boot.py ausführt.

**README.txt**

Enthält grundlegende Informationen über das Pyboard.

Öffnen Sie die Datei main.py in Ihrem Texteditor. Ersetzen Sie den Code-Kommentar, den Sie darin finden, durch den folgenden Codeschnipsel:

"import pyb"

"Pub.LED(4). on()"

Das Pub-Modul enthält alle Funktionen und Klassen, die für die Arbeit mit der Hardware des Pyboards erforderlich sind. Das obige Skript schaltet einfach LED4 (die blaue LED) ein.

# Module und Klassen

Für die gute Durchführung unserer Arbeit benötigen wir eine Menge Module und Klassen, die die Ausführung der Aufgaben ermöglichen. Es wäre mir wichtig, zunächst jedes Modul und jede Klasse zu definieren und zu erklären, um ein gutes Verständnis der Arbeit zu ermöglichen.

## Das Module Pyserial & Class Serial

Dieses Modul enthält den Zugang für die serielle Schnittstelle. Es stellt Hintergrundsysteme für Python zur Verfügung, die unter Windows, OSX, Linux, BSD (möglicherweise jedes POSIX-kompatible System) und IronPython laufen. Das Modul mit dem Namen "Serial" wählt automatisch das entsprechende Hintergrundsystem aus.

* **Eigenschaften**

Gleiche klassenbasierte Schnittstelle auf allen unterstützten Plattformen.

Zugriff auf die Port-Einstellungen über Python-Eigenschaften.

Unterstützung für verschiedene Bytegrößen, Stoppbits, Parität und Flusskontrolle mit RTS/CTS und/oder Xon/Xoff.

Arbeiten mit oder ohne Empfangs-Timeout.

Datei-ähnliche API mit "Lesen" und "Schreiben" ("readline" etc. ebenfalls unterstützt).

Die Dateien in diesem Paket sind 100% reine Python-Dateien.

Der Port ist für binäre Übertragung eingerichtet. Kein NULL-Byte-Stripping, CR-LF-Übersetzung usw. (die oft für POSIX aktiviert sind.) Dies macht dieses Modul universell einsetzbar.

Kompatibel mit der IO-Bibliothek.

RFC 2217-Client (experimentell), Server in den Beispielen angegeben.

* **Erfordernisse**

Python 2.7 oder Python 3.4 und neuer.

* **Installation**

Dadurch wird ein Paket installiert, das von Python verwendet werden kann („import serial“).

PySerial kann von „PyPI“ direkt installiert werden: („ python -m pip install pyserial „).

* **Class Serial**

Parameter:

1. **Port:**

Der Port wird sofort bei der Objekt Erstellung geöffnet, wenn ein Port angegeben wird. Er wird nicht geöffnet, wenn Port None ist und ein nachfolgender Aufruf von open () erforderlich ist.

Port ist ein Gerätename: abhängig vom Betriebssystem. z.B. /dev/ttyUSB0 unter GNU/Linux oder COM3 unter Windows.

1. **Baudrate(int):** Einheit der Kommunikationstechnik, die die Gehgeschwindigkeit darstellt.

Der Parameter Baudrate kann einer der Standardwerte sein: 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, 19200, 38400, 57600, 115200. Diese werden auf allen Plattformen gut unterstützt.

1. **Bytesize:** Anzahl der Datenbits.

Mögliche Werte: „FIVEBITS, SIXBITS, SEVENBITS, EIGHTBITS“

1. **Parity**: Aktivieren der Paritätskontrolle.

Mögliche Werte: PARITY\_NONE, PARITY\_EVEN, PARITY\_ODD.

1. **Stopbits:** Anzahl der Stoppbits.

Mögliche Werte: STOPBITS\_ONE, STOPBITS\_ONE\_POINT\_FIVE.

1. **Timeout(float):** Legt einen Lese-Timeout-Wert fest.
2. **Write\_timeout(float):** Legt einen Schreib-Timeout-Wert fest.
3. **rtscts(bool):** Hardware (RTS/CTS) -Flusskontrolle aktivieren.
4. **Open ():** Port öffnen.

Der Zustand von rts und dtr wird angewendet.

1. **Close ():** Port schließen.
2. **Read (Size=?):**

Size: Anzahl von Bytes zu lesen.

Liest Size Bytes von der seriellen Schnittstelle (Port). Wenn ein Timeout eingestellt ist, kann es vorkommen, dass weniger Zeichen als angefordert zurückgegeben werden. Ohne Timeout blockiert es, bis die angeforderte Anzahl von Bytes gelesen wurde.

1. **Write(data):** Data zu senden, und gibt die Anzahl der geschriebenen Bytes zurück

Schreibt der Byte-Daten an den Port.

## Historische Einordnung

## Quellenlage

# Hauptteil 2: Grundlagen und Methodik

## Theoretischer Bezugsrahmen

## Methodische Vorgehensweise

# Hauptteil 3: Empirische Untersuchung

## Analysen

## Interpretation der Ergebnisse

# Zusammenfassung

Hier werden die wesentlichen Ergebnisse der Arbeit zusammengefasst. Insbesondere wird hier der Bogen zu der in der Einleitung formulierten Forschungshypothese geschlagen. Hier können auch die Ergebnisse kritisch betrachtet und in den Zusammenhang der Forschungsentwicklung eingeordnet werden (Ausblick/Desiderata).

Textbausteine/Formulierungsvorschläge für das Fazit können Sie hier herunterladen:  
<http://lektorat-korrekturlesen.de/wp-content/uploads/2017/02/Textbausteine-wiss.-Arbeit-1.docx>

# Literaturverzeichnis

Überschrift nicht nummeriert.

Das Literaturverzeichnis folgt in der Hausarbeit nach dem Textteil. Es enthält sämtliche in der Hausarbeit benutzte Literatur, d. h. die Texte, die zur Stützung der Argumentation entweder wörtlich oder sinngemäß zitiert werden. Alibi-Nennungen von nicht herangezogener Literatur, um Gelehrsamkeit zu demonstrieren, gelten als verpönt.

Die im Beispielverzeichnis angeführten Titel beziehen sich auf aktuelle Literatur zum wissenschaftlichen Arbeiten. Zitierstil: Citavi-Basis-Stil.

**Brink**, Alfred (2013): Anfertigung wissenschaftlicher Arbeiten. Ein prozessorientierter Leitfaden zur Erstellung von Bachelor-, Master- und Diplomarbeiten. 5., überarbeitete und aktualisierte Aufl. 2013. Dordrecht: Springer.   
Online verfügbar unter http://gbv.eblib.com/patron/FullRecord.aspx?p=1317754.

**Eco**, Umberto; Schick, Walter (2010): Wie man eine wissenschaftliche Abschlußarbeit schreibt. Doktor-, Diplom- und Magisterarbeit in den Geistes- und Sozialwissenschaften. 13., unveränd. Aufl. der dt. Ausg. Wien: Facultas Univ.-Verl. (UTB Schlüsselkompetenzen, 1512).   
Online verfügbar unter http://www.gbv.de/dms/faz-rez/FD120010510877043.pdf.

**Esselborn-Krumbiegel**, Helga (2014): Richtig wissenschaftlich schreiben. Wissenschaftssprache in Regeln und Übungen. 3., durchges. Aufl. Paderborn: Schöningh (Uni-Tipps, 3429).   
Online verfügbar unter http://www.utb-studi-e-book.de/9783838541570.

**Hirsch-Weber**, Andreas; **Scherer**, Stefan (2015): Wissenschaftliches Schreiben und Abschlussarbeit in Naturwissenschaften und Ingenieurwissenschaften. Grundlagen – Praxisbeispiele – Übungen. 1. Aufl. Stuttgart, Stuttgart: UTB.

**Karmasin**, Matthias; **Ribing**, Rainer (2014): Die Gestaltung wissenschaftlicher Arbeiten. Ein Leitfaden für Seminararbeiten, Bachelor-, Master- und Magisterarbeiten sowie Dissertationen. 8., aktualisierte Aufl. Wien: facultas.wuv (UTB Schlüsselkompetenzen, 2774).   
Online verfügbar unter http://www.utb-studi-e-book.de/9783838542591.

**Kornmeier**, Martin (2013): Wissenschaftlich schreiben leicht gemacht. Für Bachelor, Master und Dissertation. 6., aktualisierte Aufl. Bern, Stuttgart: Haupt; UTB (utb-studi-e-book, 3154).   
Online verfügbar unter http://www.utb-studi-e-book.de/9783838540733.

**Manschwetus, Uwe (2016):** Ratgeber wissenschaftliches Arbeiten. Leicht verständliche Anleitung für das Schreiben wissenschaftlicher Texte im Studium. Lüneburg: Thurm Wissenschaftsverlag.

**Oehlrich**, Marcus (2015): Wissenschaftliches Arbeiten und Schreiben. Schritt für Schritt zur Bachelor- und Master-Thesis in den Wirtschaftswissenschaften. Berlin: Springer Gabler. Online verfügbar unter http://dx.doi.org/10.1007/978-3-662-44099-5.

**Oertner**, Monika; **St. John**, Ilona; **Thelen**, Gabriele (2014): Wissenschaftlich schreiben. Ein Praxisbuch für Schreibtrainer und Studierende. Paderborn, Stuttgart: Fink; UTB (UTB Schlüsselkompetenzen, 8569).   
Online verfügbar unter http://www.utb-studi-e-book.de/9783838585697.

**Sesink**, Werner (2012): Einführung in das wissenschaftliche Arbeiten. Inklusive E-Learning, Web-Recherche, digitale Präsentation u. a. 9., aktualisierte Aufl. München: Oldenbourg.

**Weber**, Daniela (2014): Die erfolgreiche Abschlussarbeit für Dummies. 2., aktualisierte Aufl. Weinheim: Wiley-VCH

**Weber**, Daniela (2015): Wissenschaftliches Arbeiten für Wirtschaftswissenschaftler. Hoboken: Wiley.   
Online verfügbar unter http://gbv.eblib.com/patron/FullRecord.aspx?p=2059125.

**Wolfsberger**, Judith (2010): Frei geschrieben. Mut, Freiheit & Strategie für wissenschaftliche Abschlussarbeiten. 3. Aufl. Wien: Böhlau (UTB Schlüsselkompetenzen, 3218).

# Anhänge

Überschrift nicht nummeriert

## Anhang 1

## Anhang 2

# Eidesstattliche Versicherung

Überschrift nicht nummeriert.

Wissenschaftliche Arbeiten müssen selbstständig erbracht und verfasst werden.

Die meisten Prüfungsordnungen schreiben einen Mustertext vor, der am Ende der Arbeit beigeheftet wird. Er ist zu datieren und handschriftlich zu unterschreiben.

Kann man nach Abgabe einer wissenschaftlichen Arbeit mit eidesstattlicher Erklärung einen Verstoß nachweisen, droht nicht nur die Aberkennung der Prüfungsleistung, sondern auch ein Verfahren wegen Täuschung bzw. Betrug (§ 156 StGB).

Beispiel für eine eidesstattliche Versicherung:

*„Hiermit erkläre ich, (Name), an Eides statt, dass ich die vorliegende Bachelorarbeit bzw. Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.“*

Ort, Datum, Unterschrift

# Stichwortverzeichnis

Überschrift nicht nummeriert.

Das Stichwortverzeichnis ist bei der Abgabe von Abschlussarbeiten eher nicht üblich. Es wird meist bei der Druckfassung ergänzt.

**Brink**, Alfred VII

**Eco**, Umberto VII

**Esselborn-Krumbiegel**, Helga VII

**Hirsch-Weber**, Andreas VII

**Karmasin**, Matthias VII

**Kornmeier**, Martin VII

**Manschwetus, Uwe** VIII

**Oehlrich**, Marcus VIII

**Oertner**, Monika VIII

**Ribing**, Rainer VII

**Scherer**, Stefan VII

**Sesink**, Werner VIII

**St. John**, Ilona VIII

**Thelen**, Gabriele VIII

**Weber**, Daniela (2014) VIII

**Weber**, Daniela (2015) VIII

**Wolfsberger**, Judith VIII

1. (Laser & Co. Solutions GmbH, 2020) [↑](#footnote-ref-1)
2. (Laser & Co. Solutions GmbH, 2020) [↑](#footnote-ref-2)
3. (Laser & Co. Solutions GmbH, 2020) [↑](#footnote-ref-3)
4. 4 (Uwe Brinkschulte, 2010) [↑](#footnote-ref-4)
5. 6 (George, 2014) [↑](#footnote-ref-5)
6. 5 (Tollervey, Programming with Micropython , 2017) [↑](#footnote-ref-6)
7. 7 (George, 2014) [↑](#footnote-ref-7)
8. 8 REPL steht für Read Evaluate Print Loop (Print, Lesen und Auswerten) und ist der Name der interaktiven Mikropython-Eingabeaufforderung, auf die Sie über WiPy zugreifen können. [↑](#footnote-ref-8)
9. 9 (Tollervey, Programming with Micropython , 2017) [↑](#footnote-ref-9)
10. 10 (CO, 2020) [↑](#footnote-ref-10)
11. 11 (Tollervey, Programming with Micropython, 2017. Seite17-19) [↑](#footnote-ref-11)