

WAK-Lab FAQ

6. Februar 2019



1 Einführung

Viele haben lange darauf gewartet, in Eisenach hat sich erstmals am 11. Januar 2019 eine Gruppe von 12 technisch interessierten Menschen getroffen. Sie wollen sich auch zukünftig regelmäßig treffen. Es soll gemeinsam gebastelt, gelötet und programmiert werden.

Zunächst haben wir uns drauf verständigt diese Treffen regelmäßig in der alten Posthalterei (Abb. 1) abzuhalten. Dabei werden uns die Räumlichkeiten stundenweise überlassen.



Abbildung 1: Alte Posthalterei

Wir wollen mit einer Präsenz in sozialen Medien und mit einem Flyer (Abb. 2) zunächst andere Menschen auf unsere Idee aufmerksam machen. Das weitere Vorgehen ist von der Resonanz abhängig. Ideen über Vereinsgründung und Gemeinnützigkeit bestehen, aber wir wollen bewusst dieses Thema zunächst vertagen.



Abbildung 2: Flyer

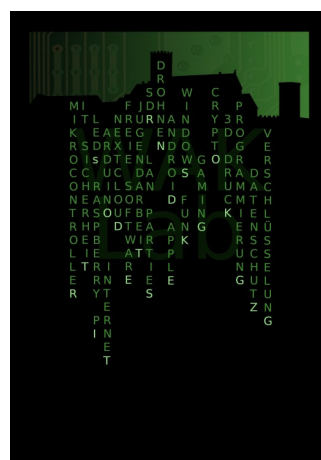


Abbildung 3: Flyer Rückseite

1.1 Finanzierung

Solange wir noch eine lose Interessensgemeinschaft sind sammeln wir bei PAYPAL <https://paypal.me/pools/c/8bkJSaWW1D> Geld für Flyer und andere notwendigen Ausgaben. Die Beteiligung ist freiwillig. Das Geld wird transparent und demokratisch ausgegeben. Eine Übersicht über die Finanzen findet ihr unter Nextcloud\Wak-Lab\Finanzen 2019.ods. Sachspenden die für den Betrieb der Infrastruktur benötigt werden, sind ebenso willkommen.

Hinweis: Es versteht sich, dass Geld- und Sachspenden nach Vereinsgründung in dessen Eigentum übergeht.

1.2 Abstimmungen

Abstimmungen werden zunächst über <http://www.strawpoll.me> durchgeführt.

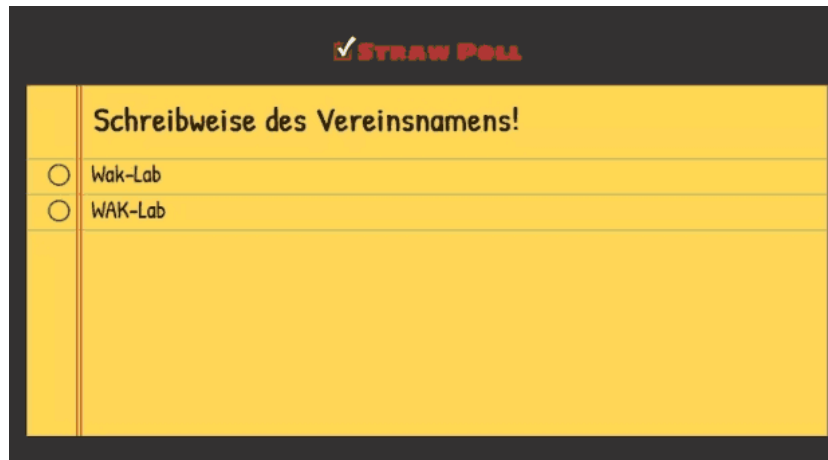


Abbildung 4: Online Abstimmung

1.3 Abstimmungen über einen Discord BOT

Parallel wird getestet ob man kurzfristige Entscheidungen innerhalb einer Stunde über einen BOT im Discord realisieren kann.

1.4 Fotos und Videos

Grundsätzlich respektieren wir, dass jeder das Recht auf sein eigenes Photo hat. Auch wenn er in einer Gruppe zu sehen ist. Daher sollten Photos immer angekündigt werden. Spontane Schnappschüsse verifiziert werden.

In der Nextcloud gibt es einen Ordner Photos. Dort werden die Photos zu Freigabezwecken zwischengelagert. Soweit nichts anderes vereinbart, werden Photos 24h lang von den Mitgliedern geprüft. Dabei werden sie gelöscht, modifiziert, in den Ordner nur für internen Gebrauch oder in den Ordner zur Veröffentlichung verschoben. Nach 24h sollen die Fotos dann frühestens veröffentlicht werden. Nicht freigegebene Fotos sollen dann auch von Medien der Nutzer gelöscht werden.

- 01__neue Fotos
- 02__Nur für Mitglieder freigegebene Fotos
- 03__Fotos die zur Veröffentlichung vorgesehen werden
- 04__Fotos zur Veröffentlichung

Innerhalb der Ordner werden die Fotos von einem Fotoadmin nach Datum in Ordner sortiert.

2 E-mail

Alle E-Mails an `info@wak-lab.org` landen bei „electribez“ auf dem Server. Auch alle anderen Mails landen bei ihm (catchall).

3 Nextcloud

Über die URL `https://nc.wak-lab.org` kommt ihr zur Verwaltungskonsole auf unserem Nextcloud Server.

Bitte beachtet, dass es sich um eine verschlüsselte „https://“ Verbindung über Port 443 handelt.

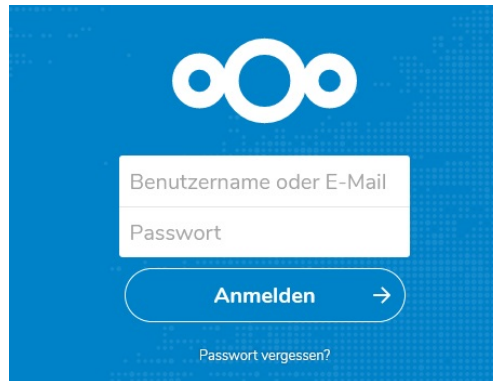


Abbildung 5: Nextcloud Login

3.1 Webinterface

Zunächst solltet ihr euer Passwort ändern, welches ihr vom Admin zugewiesen bekommen habt. Falls ihr einen Script-Blocker verwendet fügt `nc.wak-lab.org` als Ausnahme hinzu.

Ihr könnt nun über das Webinterface auf die Dateien des Wak-Lab zugreifen.

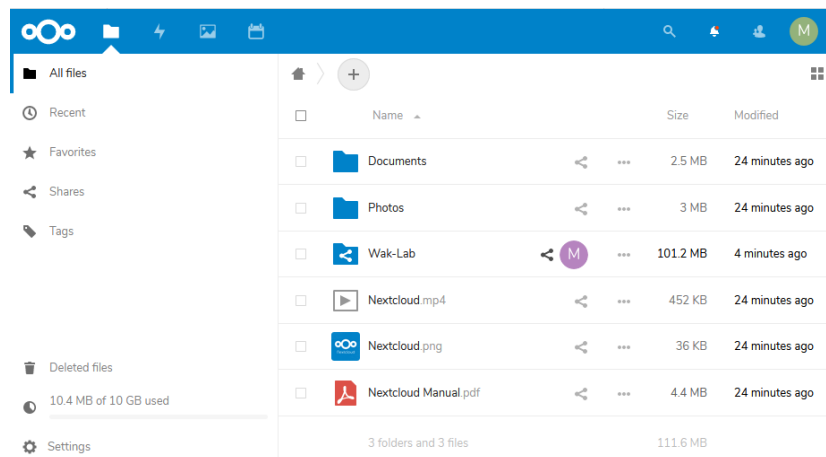


Abbildung 6: Nextcloud Web-Interface

3.1.1 Kalender

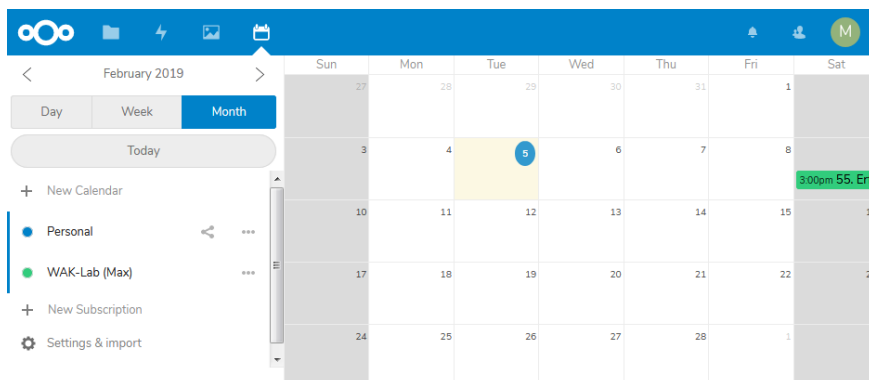


Abbildung 7: Nextcloud Kalender

Der Nextcloud Kalender lässt sich auch super einfach auf dem Endgerät deiner Wahl einbinden. Wenn du das getan hast, dann findest du 2 weitere Kalender in deiner Kalenderapp.

1. deinen persönlichen Kalender (Termine dort sind nur für dich alleine sichtbar)
2. Wak/WAK-Lab Gruppenkalender (Termine sind für Nextcloud User der Gruppe Wak/WAK-Lab Gruppe sichtbar)

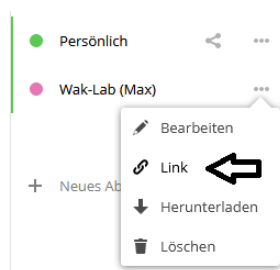


Abbildung 8: Link zum Kalender



Abbildung 9: Apple Kalender verbinden




Plattform	Kalenderintegration
 ubuntu	hier passiert das ganz automatisch sobald du unter Einstellungen/Online Konten dein Nextcloud Konto verbindest.
 xubuntu	Kalender zyklisch aus dem Netzwerk zu beziehen „wget -N -P ~/.local/share/orage https://*user*:*password*@nc.wak-lab.org/remote.php/dav/calendars/*user*/wak-lab_shared_by_Max?remote“ In Orage trägt man den Pfad ~/.local/share/orage/wak-lab_shared_by_Max?remote unter „Datei -> Tauschdaten -> Fremddateien“ ein.
 android	URL: https://nc.wak-lab.org/remote.php/dav/ Ich glaube hier wird Zusatzsoftware benötigt. Caldroid zum Beispiel. Damit habe ich allerdings keine Erfahrung. Vielleicht findet sich da noch jemand, der eine Kurzanleitung schreibt.
Apple ohne Logo weil verboten	iOS -> Einstellungen/Passwörter&Accounts -> Konto hinzufügen -> weitere -> caldav URL: https://nc.wak-lab.org/remote.php/dav/principals/users/, , deinUsername``/

Tabelle 1: Nexcloud Kalender - Client

3.2 Client Installieren

Passende Clienten findet ihr auf <https://nextcloud.com/install/>

Der Nextcloud Client stellt euch einen ständig aktualisierte Kopie des Servers in C:\Users\Username\Nextcloud zur Verfügung. Dort können wir gemeinsam an Inhalten arbeiten. Vorsicht, wenn 2 Leute an einen Thema arbeiten.

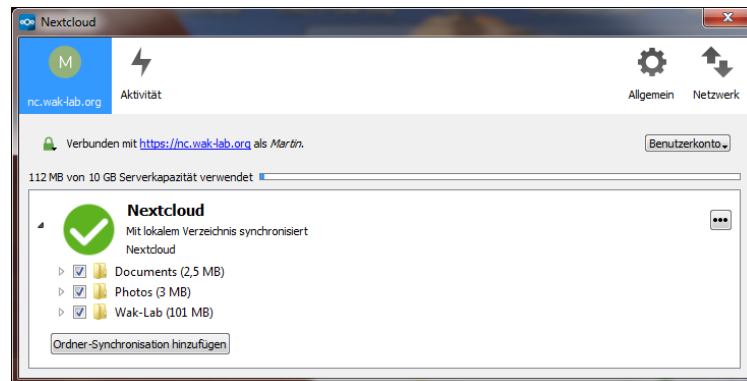


Abbildung 10: Nextcloud Windows Client

4 Discord

Wenn ihr mit Discord arbeiten wollt, benötigt ihr eine Discord Client App mit der erstellt ihr einen Discord Account. Dabei wird Nickname, Email und Passwort benötigt. Ihr könnt dann sofort loslegen und den Link zu unserem Discod Server klicken und eich mit unserem Server verbinden. URL: <https://dc.wak-lab.org> Alternativ: <https://discord.gg/FUJuq4h>

Nach der Registrierung kommt eine Bestätigungs-E-Mail in euer Postfach, das könnt ihr dann ja noch abschließen. Evtl. landet die Mail auch in eurem Spam Ordner.

Channel	Beschreibung
WAK-LAB	Wak-Lab Themen
#wak-lab_talk	
#events	
#offtopic	
#develop	
LAB TOPIC	Allgemeine Themen
#windows	
#linux	
#develop	
#funk_radio_sdr	
#3d-druck	
#smart-home	
#werkstatt	
#awareness	
ELEKTRONIK	Elektronik Themen
#elektronik	
#arduino	
#esp8266_und_co	
#raspberry-pi	
WAK-LAB(INTERN, SPÄTER E.V.)	Interne Themen
#wak-lab_verein	
#wak-lab_vorstand	
#ankündigungen	
#abstimmungen	

Tabelle 2: Liste der Channels

5 Git und SVN

Vorweg will ich kurz erklären, was es mit den Tools zur Versionskontrolle auf sich hat.

Es geht immer darum, eine lokale Kopie der Quellen von einem Server zu laden. Im einfachsten Fall bei Github als ZIP-Datei. Dies ist in den meisten Fälle ausreichend. Der Link ist jedoch für unsere Versionskontrolle interessant. Dort liegt nun das Repository die ich nenne sie jetzt mal Masterkopie.

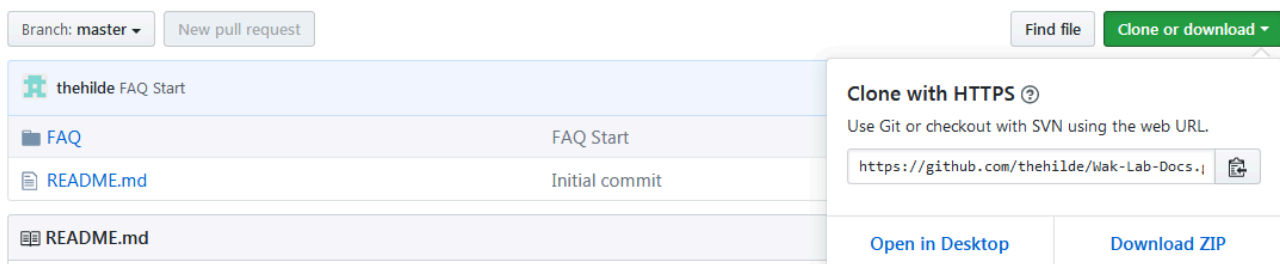


Abbildung 11: Eine lokale Kopie Clonen

Spannend wird es jedoch, wenn die Quellen sich auf dem Server weiterentwickeln oder man sogar selbst eine Verbesserung einbringen will. Die Tools GIT oder SVN managen nun die Synchronisation mit dem Server. Es kann der zeitliche Verlauf der Änderungen angezeigt und zu beliebigen Punkten zurück gesprungen werden. Nie wieder ein "Gestern ging es noch bevor das umgestrickt habe <aarg!>". Welche Änderungen habe ich vor 3 Jahren für XY rein gehackt? Und vor allem Warum? Welchen Fehler wollte ich beseitigen, welches Ticket habe ich bearbeitet? Oder war es der Kollege?

Tools wie Tortoise SVN besitzen eine Explorer Integration, arbeiten über das Kontextmenü und zeigen alle Änderungen automatisch an.

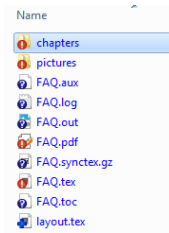


Abbildung 12: Tortoise SVN Explorer Integration

Finden Änderungen in der selben Datei aber in unterschiedlichen Bereichen statt, kann das Tool die Änderungen automatisch zusammenführen. In manchen Fällen muss dies jedoch händisch erfolgen.

Es soll noch gesagt sein, das GIT und SVN zwei verschiedene Tools sind, die etwas abweichende Philosophien haben. Zum Glück sind die beiden auf Github gut verheiratet worden.

Folgende Dateitypen sind Textbasiert und können prima verwaltet werden.

- Arduino .ino Dateien
- Quelltexte z.B. .c .cpp .h .pas .vhd
- Natürlich .html .php
- Windows .ini Dateien
- Eagle Layout Dateien und Bibliotheken ab Version 6
- STM Cube32 Dateien
- \LaTeX .tex Dateien
- Auch Word ist Prima integriert in Tortoise SVN und öffnet den Word eigenen DIF Viewer.

5.1 Ein Git Beispiel

Nach dem Login in Github können wir ein neues Projekt erstellen. (Abb. 13)

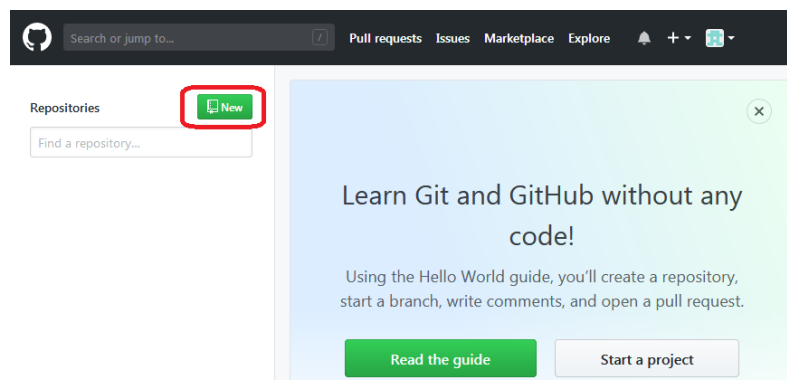



Abbildung 13: Neues Projekt

Das neue Repository bekommt einen Namen eine kurze Beschreibung. Wir legen es Public an, dann ist es auf Github kostenlos. Alternativ können wir später auch unseren Server benutzen. Wir lassen aus unserer Kurzbeschreibung gleich eine Readme Datei erstellen. Hier können wir Markdown-language verwenden um die Anzeige in github etwas zu formatieren.

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner:  gituser / Repository name: NodeMCUBlink ✓

Great repository names are short and memorable. Need inspiration? How about [reimagined-octo-couscous](#).

Description (optional): the only one "Hello, World!" program

☒ Public
Anyone can see this repository. You choose who can commit.

☐ Private
You choose who can see and commit to this repository.

☒ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None | Add a license: None ⓘ

Create repository

Abbildung 14: Repository erstellen

Zunächst möchte ich folgende Kurzbeschreibung nicht vorenthalten <https://rogerdudler.github.io/git-guide/index.de.html> dort stehen die wichtigsten git Befehle.

Wir legen uns nun lokal ein Verzeichnis an, und clonen uns das Verzeichnis vom Github Server. Den Link findet man im „Clone or download“ Menü oben rechts (Abb. 11).

```
C:\Users\SuperDave>md blinky
```

```
C:\Users\SuperDave>cd blinky
```

```
C:\Users\SuperDave\blinky>git clone https://github.com/thehilde/NodeMCUBlink.git
Cloning into 'NodeMCUBlink'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
```

```
C:\Users\SuperDave\blinky>dir
Datenträger in Laufwerk C: ist OS
Volumeseriennummer: 24F2-3FE3
```

Verzeichnis von C:\Users\SuperDave\blinky

```
30.01.2019  20:22    <DIR>          .
30.01.2019  20:22    <DIR>          ..
30.01.2019  20:22    <DIR>          NodeMCUBlink
               0 Datei(en),               0 Bytes
               3 Verzeichnis(se), 111.597.510.656 Bytes frei
```

```
C:\Users\SuperDave\blinky>cd NodeMCUBlink
```

```
C:\Users\SuperDave\blinky\NodeMCUBlink>dir
Datenträger in Laufwerk C: ist OS
Volumeseriennummer: 24F2-3FE3
```

Verzeichnis von C:\Users\SuperDave\blinky\NodeMCUBlink

```
30.01.2019  20:22    <DIR>        .
30.01.2019  20:22    <DIR>        ..
30.01.2019  20:22                54 README.md
                1 Datei(en),           54 Bytes
                2 Verzeichnis(se), 111.597.510.656 Bytes frei
```

Die README.md Datei ist gut auf unserem Rechner angekommen. Jetzt Schieben wir weiter Dateien in den Ordner und fügen sie mit „add“ dem Projekt hinzu.

Am einfachsten ist es erst einmal alle Dateien der Versionsverwaltung hinzuzufügen. Obwohl dies mit Sicherheit nicht das eleganteste ist. Da man Log Dateien, Objekt Dateien oder Binärdateien möglicherweise nicht Versionieren möchte. Einfach weil sie beim compilieren sowieso immer wieder neu erzeugt werden.

```
C:\Users\SuperDave\blinky\NodeMCUBlink>git add *
warning: LF will be replaced by CRLF in .vscode/c_cpp_properties.json.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in platformio.ini.
The file will have its original line endings in your working directory
```

In dieser Windows Installation habe ich wohl versehentlich angegeben, dass Zeilenumbrüche korrigiert werden. wir machen erst mal weiter.

```
C:\Users\SuperDave\blinky\NodeMCUBlink>dir
```

```
Datenträger in Laufwerk C: ist OS
Volumeseriennummer: 24F2-3FE3
```

Verzeichnis von C:\Users\SuperDave\blinky\NodeMCUBlink

```
30.01.2019  20:47    <DIR>        .
30.01.2019  20:47    <DIR>        ..
30.01.2019  20:47    <DIR>        .pioenvs
30.01.2019  10:44                1.624 .travis.yml
30.01.2019  20:47    <DIR>        .vscode
30.01.2019  20:47    <DIR>        include
30.01.2019  20:47    <DIR>        lib
30.01.2019  10:44                449 platformio.ini
30.01.2019  20:22                54 README.md
30.01.2019  20:47    <DIR>        src
30.01.2019  20:47    <DIR>        test
                3 Datei(en),           2.127 Bytes
                8 Verzeichnis(se), 111.564.550.144 Bytes frei
```

```
C:\Users\SuperDave\blinky\NodeMCUBlink>
```

Da wir das Verzeichnis .vscode nicht versionieren wollen, schmeißen wir rekursiv (-r) das gesamte Verzeichnis wieder heraus. Die Datei soll allerdings liegen bleiben als (-cached).

```
C:\Users\SuperDave\blinky\NodeMCUBlink>git rm -r --cached .vscode
```

jetzt committen wir das ganze in das lokale Repository

```
C:\Users\SuperDave\blinky\NodeMCUBlink>git commit -m "Ab damit ins lokale Repo."
```

```
[master a0e890c] Ab damit ins lokale Repo.  
8 files changed, 342 insertions(+)  
create mode 100644 .pioenvs/do-not-modify-files-here.url  
create mode 100644 .pioenvs/structure.hash  
create mode 100644 .travis.yml  
create mode 100644 include/README  
create mode 100644 lib/README  
create mode 100644 platformio.ini  
create mode 100644 src/main.cpp  
create mode 100644 test/README
```

```
C:\Users\SuperDave\blinky\NodeMCUBlink>
```

Jetzt steckt alles im lokalen Repository und man kann das Spiel mit add rm und commit so lange treiben bist man lokal ein lauffähiges Programm hat was man auf github mit anderen Teilen will. Nach jedem commit werden alle Änderungen am Programm archiviert und können später auch wieder hergestellt werden. Deder commit sollte mit einer Nachricht versehen werden, was man in dem Moment verändert hat. Das hilft später bei einem Changelog.

Ist man also bereit alles auf github hochzuladen verwendet man push.

```
C:\Users\SuperDave\blinky\NodeMCUBlink>git push origin master  
Enumerating objects: 23, done.  
Counting objects: 100% (23/23), done.  
Delta compression using up to 4 threads  
Compressing objects: 100% (17/17), done.  
Writing objects: 100% (22/22), 4.12 MiB | 232.00 KiB/s, done.  
Total 22 (delta 1), reused 0 (delta 0)  
remote: Resolving deltas: 100% (1/1), done.  
To https://github.com/thehilde/NodeMCUBlink.git  
08a2401..4d087a8 master -> master
```

```
C:\Users\SuperDave\blinky\NodeMCUBlink>
```

beim Commit erscheint beim ersten mal eine Passwortabfrage.

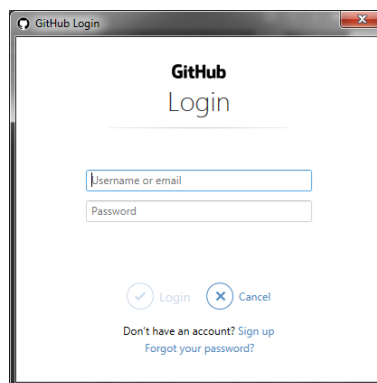


Abbildung 15: Passwortabfrage für das hochladen

Wie man jetzt auf Github sehen kann, ist alles auf Github gelandet, was wir lokal hinzugefügt haben.

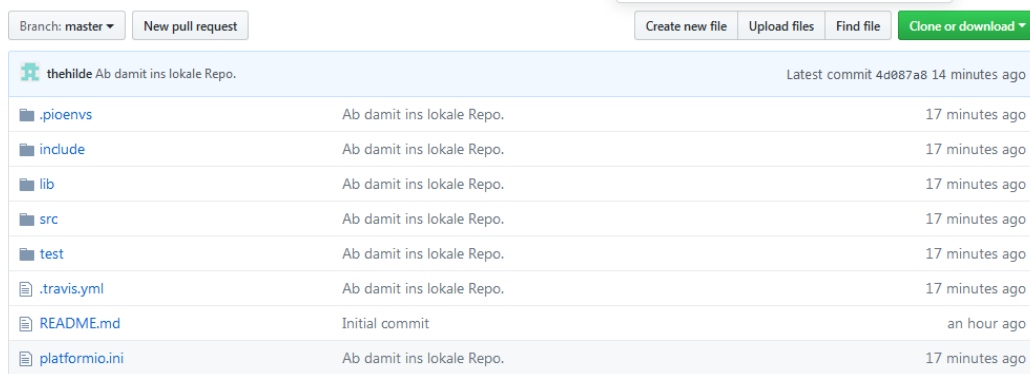


Abbildung 16: Master Branch auf GitHub

6 L^AT_EX

Latex ermöglicht es aus Textdateien formatierte PDF Dokumente zu erstellen. Ähnlich der Buchfunktion im Wikipedia. Außerdem können Ausgaben von Programmen ansprechend formatiert ausgegeben werden. Durch das Einbinden von externen Dateien können Textbausteine, Layouts oder Bilder zentral abgelegt werden. Beispielsweise können dadurch Layout, Adresse oder Logo eines Vereins einfach in allen Dokumenten gleichzeitig verändert werden. Die Dokumente müssen nur neu kompiliert werden. Ich verwende als Editor den Texmaker und als Latex Distribution Miktex. Sollte es dazu Fragen geben einfach mal an mich wenden.

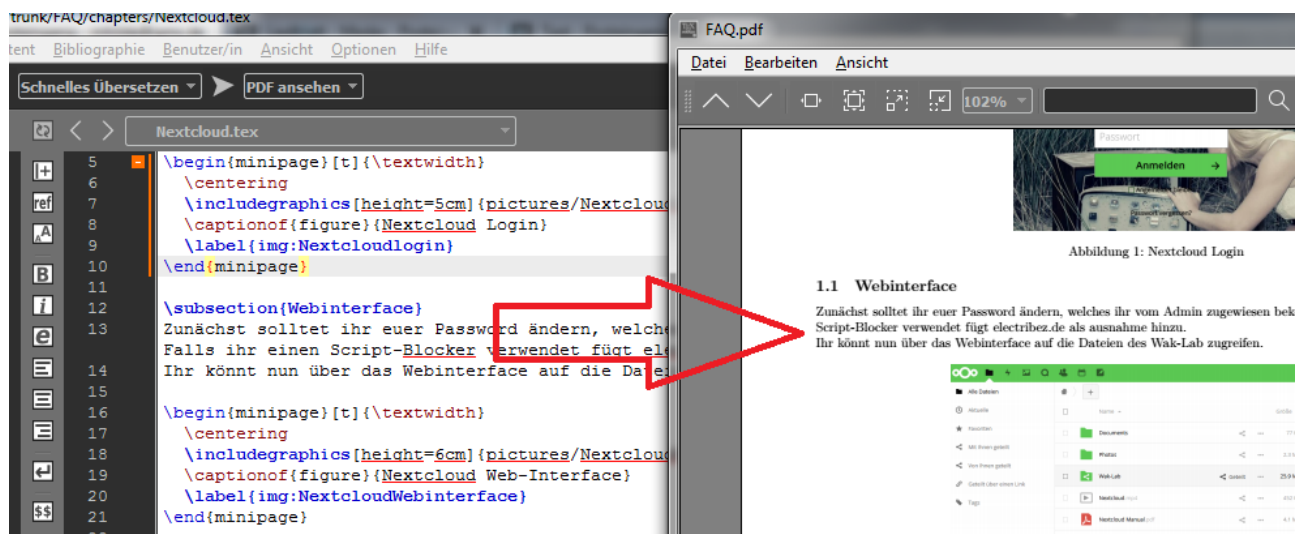


Abbildung 17: Texmaker