wiFred

WiFi throttle for model railroads using the wiThrottle protocol

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
0.3	6-27-2021	Added v0.6 description	HR

Contents

List of Figures

List of Tables

Abstract

This document describes the usage and configuration of the wiFred - a very simple wireless throttle for model railroads to connect to wiThrottle servers like JMRI. It also contains schematics and BOMs for the device - for both LiPo battery versions in active development - as well as programming instructions and assembly tips, and also an overview of options for the server side of things.

The most recent version of this document can be found at https://newheiko.github.io/wiFred, https://github.com/newHeiko/-wiFred/raw/master/documentation/docu.pdf and https://github.com/newHeiko/wiFred/blob/master/documentation/docu.adoc.

If you want to know more about the development history of the wiFred, skip ahead to Section 6 - otherwise read on with Section 1.

1 Quickstart Guide

1.1 First startup

1. Charge device with a micro USB charger until charging LED switches to green
2. Turn on device with charger still connected (calibrates low battery threshold)
3. Wait for red LED on top to stop flashing and stay lit
4. Use any WiFi device to search for and connect to network wiFred-configXXXX
5. Connect to http://config.local and configure device

More information can be found at https://newHeiko.github.io/wiFred.

Red LED	**Green LED (Left)**	**Green LED (Right)**	**Status**
Slow Blinking (0.5 Hz)	Off	Off	Trying to connect to WiFi network
Fast Blinking (2 Hz)	Off	Off	Successful WiFi connection, trying to connect to wiThrottle server and acquire locos
Off	Off	On	Regular operation, forward direction
Off	On	Off	Regular operation, reverse direction
Off	Flashing	On	Emergency stop, forward direction. Also happens when switching direction with speed potentiometer not at zero
Off	On	Flashing	Emergency stop, reverse direction. Also happens when switching direction with speed potentiometer not at zero
Off	Off	Blinking	Battery low, regular operation, forward direction
Off	Blinking	Off	Battery low, regular operation, reverse direction
Off	Flashing	Blinking	Battery low, Emergency stop, forward direction
Off	Blinking	Flashing	Battery low, Emergency stop, reverse direction
Short flashes	Off	Off	Throttle in low-power mode
Off	Off	Off	Battery empty or no battery inserted
On	Off	Off	No connection to existing WiFi network. Created internal configuration WiFi network
On	On	On	Configuration mode enabled while connected to existing WiFi network. All locos emergency stop to avoid runaways. Push SHIFT + ESTOP again to exit configuration mode
To recover from an emergency stop, turn speed potentiometer to zero to re-gain control.			

Table 1: LED patterns and their meaning on the wiFred

2 Usage

2.1 Operating locos

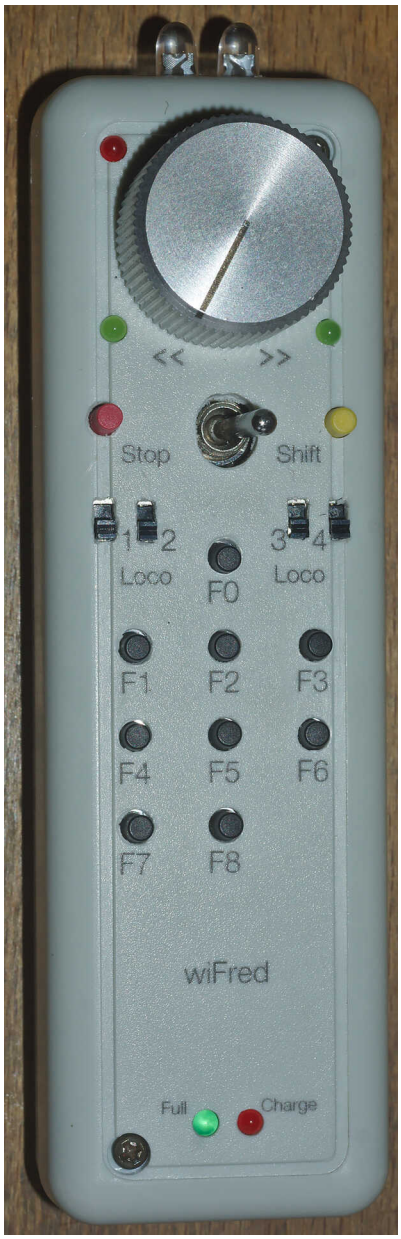


Figure 1: Controls and features of the wiFred throttle

Figure 1 shows the controls of the wireless throttle. They consist of the following:

- Four loco selection switches (loco 1 on the left, loco 4 on the right, move towards speed potentiometer to enable)
- Speed potentiometer (Counter-clockwise endstop: Stop, clockwise endstop: Full speed)
- Direction switch - move right for forward movement, left for reverse movement
- Black function keys F0 to F8
- Yellow shift key to trigger F9-F16 and turn on flashlight function

- Red emergency stop key
- Two green direction indicator LEDs next to speed potentiometer
- Red status LED next to speed potentiometer
- Red charging indicator LED at lower end of device - lit while charging
- Green fully charged indicator LED at lower end of device - lit when fully charged as long as charger still connected

As soon as any of the loco selection switches is moved into the "enabled" position, the throttle will boot up and try to connect to a wireless network. When all four loco selection switches are "disabled", the throttle will disconnect from the wireless network after a grace period of five seconds. The device will then go into low power mode, in which the battery will last for more than a year.

If no connection to the network configured into the device can be established within 60 seconds, the throttle will create it's own wireless network named **wiFred-config** plus four hex digits taken from the MAC address of the throttle WiFi interface, for example **wiFred-config0CAC**, to enable configuration as described in Section 3.

Four different locos with long DCC addresses can be assigned to the four loco selection switches. Commands derived from the speed potentiometer, the direction switch and the function keys will be transmitted to all selected locos (near) simultaneously, with a certain translation table enabling some locos to go backwards when others go forwards and also limiting function keys to some of the four locos only - this is described in more detail in Section 3.5 and Section 3.6.

Pushing the red emergency stop key will cause the throttle to send an emergency stop signal to all four locos attached. After an emergency stop, turn the speed potentiometer to zero to re-enable control of the locos.

Pushing the red emergency stop key while holding down the shift key will place the device into configuration mode (as well as issueing an emergency stop to all attached locos). See Section 3 for more details on how to access the throttle to do the configuration.

Any change in the loco selection switches will cause the throttle to send an emergency stop command to all attached locos. This makes sure that any loco that is deselected will stop on the layout and avoids newly selected locos suddenly taking off at speed. The same is true for a change in the direction switch, to avoid high-speed reverse maneuvers. Turn the speed potentiometer to zero to re-enable control of the locos.

When the battery is low, the device will not re-activate before charging the batteries, but continue operating for approximately an hour if active. When the battery is empty, it will disconnect and enter low power mode. Expected runtime is around 20 hours of full time operations, more if the throttle is placed in low power mode when the locos are not running.

During startup and operation, the LEDs will show the patterns explained in Table 1.

2.2 Charging the wiFred

The wiFred can be charged through the Micro-USB connector at the lower end of the device. Maximum charging current is approximately 400 mA and the device does not communicate with the USB host, so technically there is no guarantee that charging from a USB cable will work, but most chargers, computer ports or power banks do not check the current before powering up.

As long as the battery is being charged, the red charging indicator LED will be lit. When the battery is fully charged, the green charged indicator LED will be lit as long as the charger is still connected. Expected charging time is around five to six hours for a full charge.

Even while charging, the device can still be operated (particularly helpful with a power bank) but since the operating current will come out of the battery, the battery will never be fully charged.

If both charging status LEDs light up when a charging cable is connected, probably the internal connection to the battery is faulty.

3 Configuring the wiFred

Before using the device, it must be configured. At the very least, the General Configuration page Figure 2 has to be submitted once to be saved to non-volatile memory. If no valid configuration is detected at startup, the device will start with a default configuration with no locos enabled and no WiFi settings, so it won't be able to connect to any WiFi network.

Important
After entering any kind of text (names, numbers...) into text fields, the corresponding "Save" button has to be pressed to submit the changes to the wiFred.

3.1 Entering configuration mode

There are two ways to enter configuration mode:

1. Power up the throttle/select a loco when the configured WiFi network is not in range (or when there is no valid configuration - the first startup of a new throttle will fall into this category)
2. Press SHIFT and ESTOP together when the throttle is connected

In the first case, the throttle will create a wireless network named **wiFred-config** plus four hex digits taken from the MAC address of the throttle WiFi interface, for example **wiFred-config0CAC** and announce its presence under the name **config.local** as well as creating a captive portal. Any WiFi device with a web browser can connect to that network and open a web browser to point to **http://192.168.4.1** or **http://config.local**.

Note
This has been tested with Mozilla Firefox and Opera on Linux with Avahi (a Zeroconf implementation), FOSS Browser on Android 9/10 and Safari on iOS 13/14.

In the second case, the throttle will only announce its presence under the name **config.local** using the Bonjour/Zeroconf-protocol. Any device on the same WiFi network with Bonjour/Zeroconf can use a web browser to access the configuration at **http://config.local**. See Section 4.7 for an explanation what is required to have your device read Bonjour/Zeroconf announcements.

Note
If the IP address or the name of the throttle during normal operation is known, the configuration pages can also be accessed by pointing a web browser to it at any time while it is connected. This is mostly untested and therefore not recommended while the throttle is running locos.

Note
This has been tested with Mozilla Firefox and Opera on Linux with Avahi (a Zeroconf implementation) and BonjourBrowser on Android 10.

Figure 2 shows the first page you will see when you point a web browser at your wiFred throttle. It is divided into multiple sections explained in the following chapters.

Figure 2: Screenshot of wiFred main configuration page

3.2 General configuration

In the "General configuration" section there is only one configuration option: The throttle name. This is a free-form identification string of the throttle. It shows up in the wiThrottle window of JMRI as shown in Figure 7 and can be used to identify the throttle during configuration.

Note
The wiFred also announces its presence on the WiFi network through Bonjour/Zeroconf using a sanitized version of the name, i.e. a throttle called "Heiko Prototype 2-2" will announce its presence as **heikoprototype22.local** when not in configuration mode.

3.3 WiFi configuration

The "WiFi configuration" section shows a list of configured WiFi networks. The wiFred will connect to any network in this list, choosing the strongest one if multiple configured networks are in range.

Existing entries can be removed by clicking on the "Remove SSID" button in the line of the network that shall be removed.

New entries can be added either by manually entering the SSID and PSK[1] if required and clicking the "Manually add network" button or by clicking on the "Scan for networks" link which takes the user to the page shown in Figure 3.

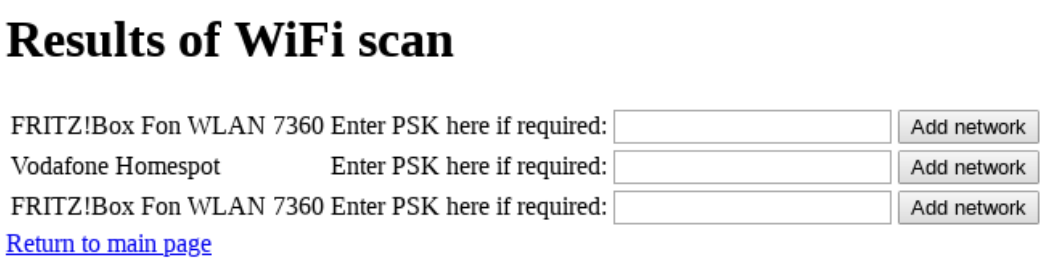


Figure 3: Screenshot of wiFred "Scan for WiFi"-page

This page will take a few seconds to load, since the scan for networks has to be completed first. It shows all the networks found during the scan. Networks can be added to the list by clicking the "Add network" button, after entering the PSK[1] in the field next to it.

Note
The wiFred does not support WPS and it won't accept multiple networks with the same SSID but different PSKs. More details regarding the network requirements can be found in Section 4.

The new WiFi configuration will not be activated until the wiFred is restarted, either through a power-cycle or by clicking on the "Restart wiFred to enable new WiFi-settings" link on the configuration page.

3.4 Loco server configuration

Following the WiFi configuration, the section "Loco server configuration" allows configuring the wiThrottle server to which the wiFred shall connect. The default setting - automatically detect server - works well if there is only one wiThrottle server on the network. It will connect to any server announcing its presence on port 12090 through Zeroconf/Bonjour, the result of the Zeroconf/Bonjour-search will be shown here when the wiFred has automatically discovered a server.

3.5 Loco configuration

Following the "Loco server configuration", there are four identical sections assigned to the four different locomotives which can be controlled with this throttle. Each section consists of the following settings:

[1] Pre-Shared Key, often just called password

DCC address Can be a short address between 1 and 127 (also used for consists) or a long address between 0 and 10239.

Note
Short addresses between 1 and 127 are not the same as long addresses between 1 and 127, and many DCC systems do not support all those addresses.

If this is set to -1, the corresponding loco is disabled.

Long address? Checkbox to change the behaviour of the DCC address input field described above.

Direction This has three options:

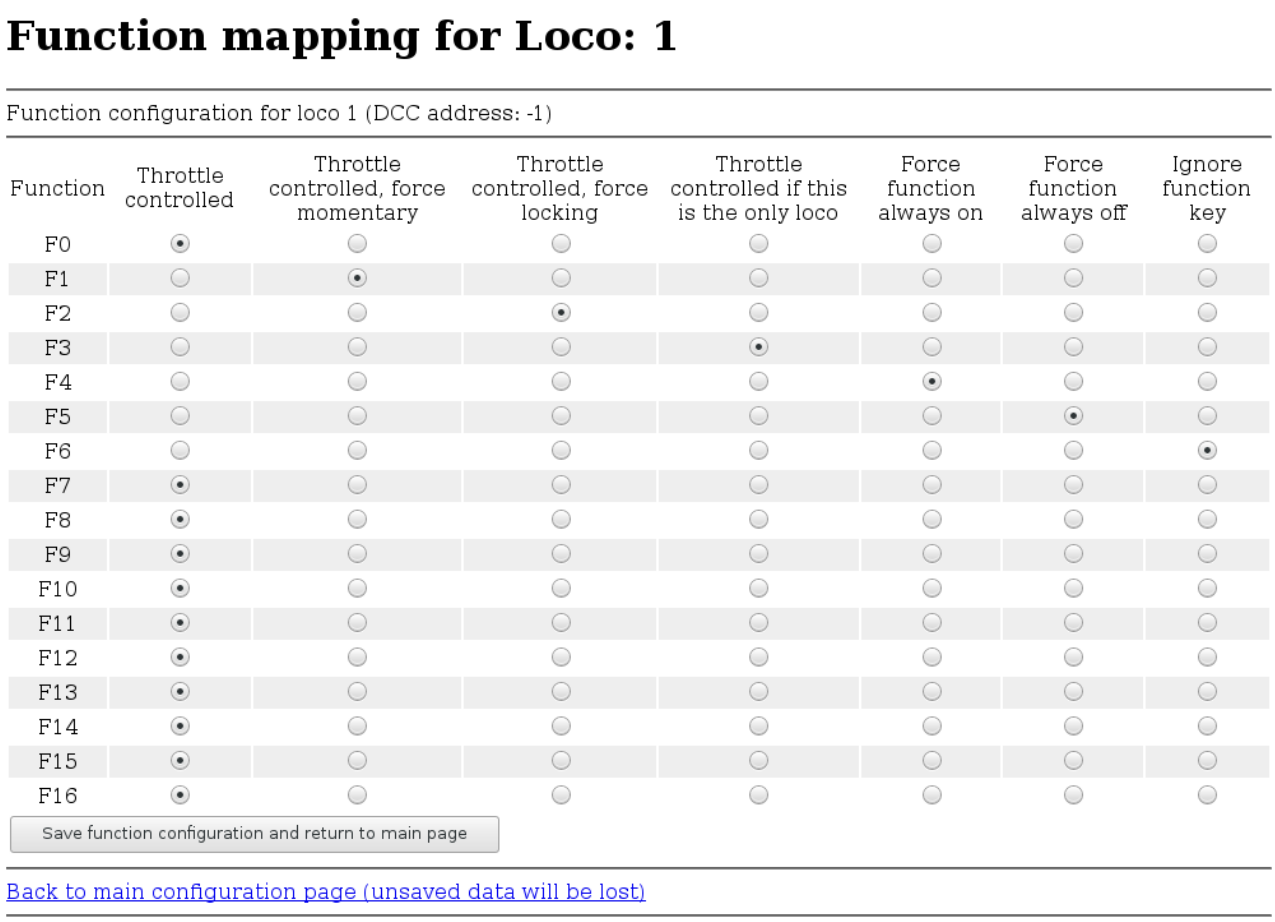
- Normal: Loco will travel forwards if wiFred is set to forwards and vice versa
- Reverse: Loco will travel backwards if wiFred is set to forwards and vice versa
- Don't change: Loco will continue to travel in the same direction it was set to when it was activated and switch direction when the wiFred switches direction

Function mapping Link to the function mapping subpage for the corresponding loco, as described in Section 3.6. Clicking this link will lose all information entered on the current page and take the web browser to a different subpage.

Important
Reminder: Changes are saved using the "Save loco config" button which may look different in different web browsers (firefox shown).

3.6 DCC function configuration

By default, if a function key is pressed, the throttle will send the appropriate commands to every loco under control. Under certain circumstances, this may not be desired - the obvious example being a loco in the middle of a multi-unit consist, which should not have lights or ditchlights. So this page - shown in Figure 4 - offers the option to chose between three different settings for every function on each of the four locomotives (one page per locomotive):



Function mapping for Loco: 1

Function configuration for loco 1 (DCC address: -1)

Function	Throttle controlled	Throttle controlled, force momentary	Throttle controlled, force locking	Throttle controlled if this is the only loco	Force function always on	Force function always off	Ignore function key
F0	◉	○	○	○	○	○	○
F1	○	◉	○	○	○	○	○
F2	○	○	◉	○	○	○	○
F3	○	○	○	◉	○	○	○
F4	○	○	○	○	◉	○	○
F5	○	○	○	○	○	◉	○
F6	○	○	○	○	○	○	◉
F7	◉	○	○	○	○	○	○
F8	◉	○	○	○	○	○	○
F9	◉	○	○	○	○	○	○
F10	◉	○	○	○	○	○	○
F11	◉	○	○	○	○	○	○
F12	◉	○	○	○	○	○	○
F13	◉	○	○	○	○	○	○
F14	◉	○	○	○	○	○	○
F15	◉	○	○	○	○	○	○
F16	◉	○	○	○	○	○	○

Save function configuration and return to main page

Back to main configuration page (unsaved data will be lost)

Figure 4: Screenshot of wiFred function handling config page

Throttle controlled When the first loco is enabled by moving the selection switch to the "selected" position, the current status of the function is queried and saved. When selecting the next loco, the function status is set to be the same. Afterwards, key presses are handed through to the loco. Whether the function will be momentary[2] or locking[3] will be determined by the server, in the case of JMRI by the roster entry.

Throttle controlled, force momentary Function key will be handed through to the loco as a momentary[2] function.

Throttle controlled, force locking As described for **Throttle controlled** but the function will be forced to be locking[3] regardless of the server settings[4].

Throttle controlled if this is the only loco As described for **Throttle controlled, force locking** but the function key will only be toggling the function if no other loco is enabled. This allows for i.e. turning headlights and rearlights on and off on individual locos before assembling a consist.

Force function always on The function will be forced on when the loco is activated and kept on regardless of key presses[5].

Force function always off The function will be forced off when the loco is activated and kept off regardless of key presses.

[2] Momentary functions are only activated for as long as the function key is pressed - like a whistle or horn that sounds just as long as the key is pressed.

[3] Locking functions are toggled with every press of the function key - like pressing F0 once typically turns the headlight on, pressing F0 again turns it off.

[4] The JMRI setting "F2 always momentary" will override this.

[5] This will also turn momentary functions permanently on.

Ingore function key No function commands will be sent to this loco.

Important
Reminder: Changes are saved using the "Save function configuration" button which may look different in different web browsers (firefox shown).

3.7 wiFred status

The "wiFred status" section shows the current battery voltage, as measured by the wiFred. This is updated on reloading the page, not continuosly.

It also shows the current firmware version(s).

3.8 wiFred system

The "wiFred system" section consists of two links:

- Reset wiFred to factory defaults - which leads to a confirmation page shown in Figure 5 to reset all configuration data to factory defaults as on a new wiFred.
- Update wiFred firmware - which leads to a firmware update page shown in Figure 6 to update the wiFred firmware of the ESP8266 / ESP32-S2. Find the .bin-file from the arduino build folder or from the [Github repository for wiFred], click on "Browse", navigate to the .bin-file and finally initiate the update with a click on "Update Firmware" - which will take a while.

Note
It may be necessary to power down the wiFred entirely for a firmware update to restart properly.

Figure 5: Screenshot of wiFred configuration reset page

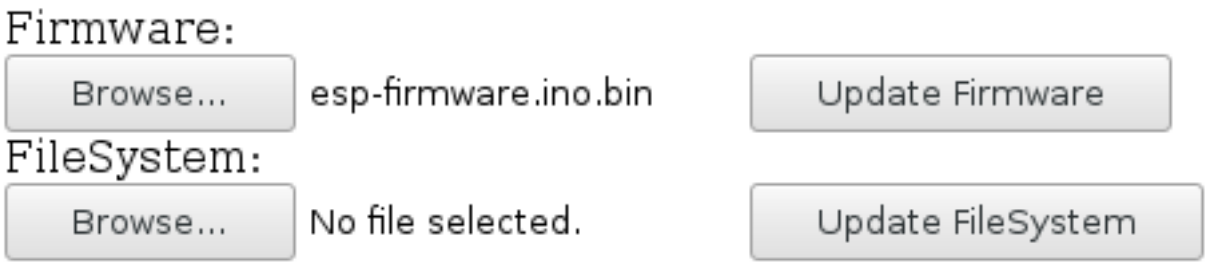


Figure 6: Screenshot of wiFred firmware update page

4 Options for server setup

4.1 Components required to get the wiFred running

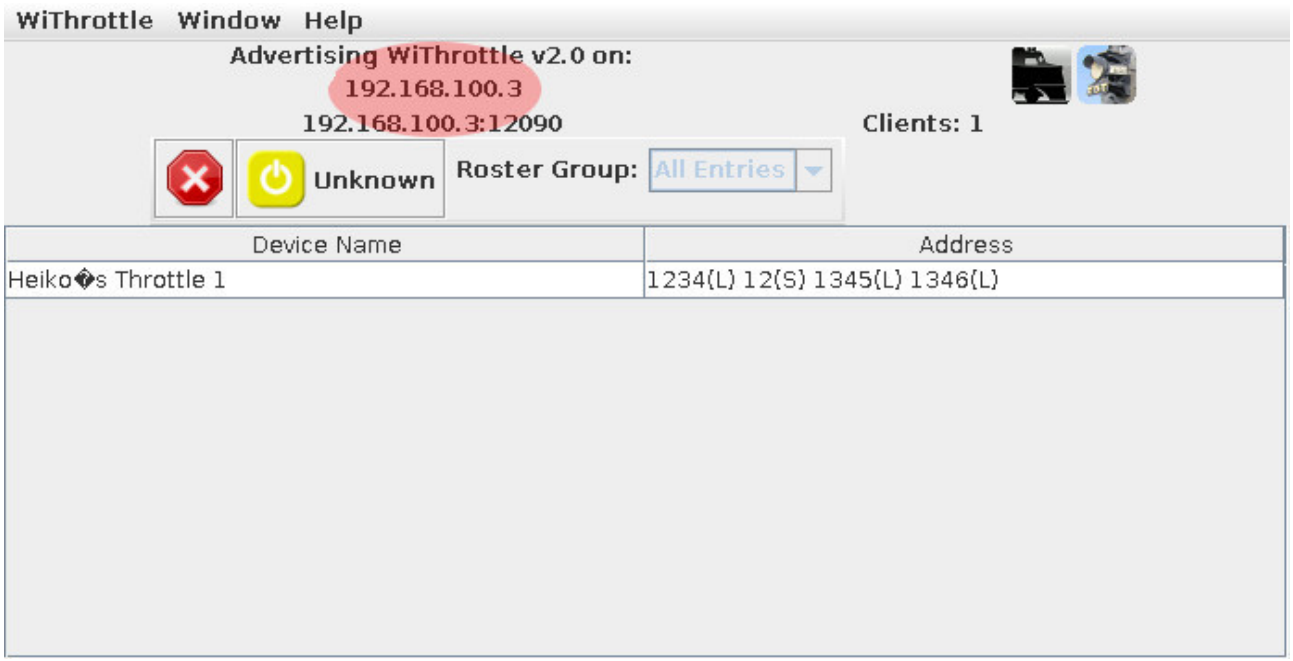


Figure 7: Screenshot of wiThrottle screen showing one throttle connected

Figure 8 shows the connections between the devices required to run trains using the wiFred.

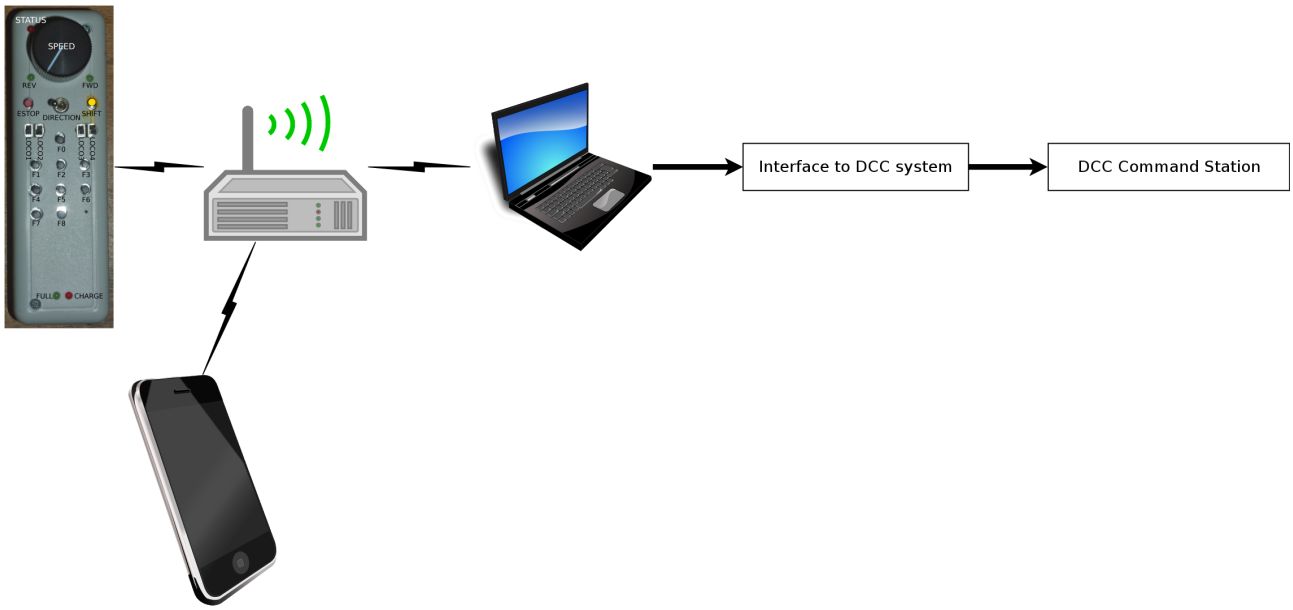


Figure 8: Overview of devices required to run trains with the wiFred

The symbols in Figure 8 symbolize the following parts:

1. An IEEE 802.11b/g/n 2.4 GHz WiFi access point described in detail in Section 4.4
2. A PC or laptop computer with Windows, Linux or MacOS to run the JMRI server described in detail in Section 4.5
3. A way to connect the JMRI server to the model railroading layout described in detail in Section 4.6
4. A device with a web browser connected to the same network as the wiFred to configure it - can be the same physical device as the one running the server, if it meets the requirements in Section 4.7

Multiple options for every step or combining these steps are described in the following sections.

Note
Basically, if a layout is set up to run trains with a smartphone running wiThrottle or EngineDriver, a wiFred should work with no changes to the layout configuration. If a layout is set up in a way that trains can be run from a JMRI screen throttle on a computer, only a WiFi connection to the JMRI computer needs to be added.

4.2 Out-of-the-box server-side options

A pretty much out-of-the-box solution for a Raspberry Pi is provided by [Steve Todd], which creates its own WiFi network, auto-detects multiple options to interface to a DCC layout and starts a wiThrottle server. It has been tested in the JMRI 4.16 and JMRI 4.20 versions to work with the wiFred, connecting to a Z21 black through both an RRCircuits LocoBuffer USB and a Digitrax PR3 via Loconet.

Although untested so far, adding a [Digitrax LNWI] to a Digitrax system or an [MRC Prodigy WiFi] to an MRC system should allow the wiFred to run locos out-of-the-box as well.

4.3 Step by step instructions for a Windows computer

This has been tested on a Windows 7 64Bit laptop computer with a 2.4GHz WiFi card inside the PC.

4.3.1 Installation

1. Install HostedNetworkStarter
2. Install DHCP server - download and extract all the files from the zip file to somewhere on your harddrive, for example C:\DHCPServer
3. Install a JDK, version 8 and 11 have been tested. For example, Version OpenJDK 11 (LTS), JVM HotSpot. Choose the 64bit version for most modern hardware, 32bit only if you are running a 32bit operating system. Easiest option: MSI file, download and install.
4. Install JMRI - versions tested to run with the wiFred include 4.14, 4.16, 4.18 and 4.20. Most recent production version recommended.

Note
Windows 10 provides a Hotspot feature that will work in place of HostedNetworkStarter and DHCP server, so you can jump right into installing JDK and JMRI.

4.3.2 Configuration

1. Start HostedNetworkStarter from the start menu, enter a Network Name and Network Key, then hit the Start button. Note the "Hosted Network Connection Name" for the next step
2. Start the DHCP server wizard from C:\DHCPServer\dhcpwiz.exe, select the network with the name that's the same as the "Hosted Network Connection Name" from the step before, hit "Next" a few times (deselecting all additional supported protocols), Write INI file, Start Service and Configure Firewall Exceptions
3. Start JMRI using the DecoderPro icon on the desktop, setup your layout connection, test if you can run a loco with a JMRI throttle
4. Within JMRI, start the WiThrottle Server from the Actions menu. If a firewall popup comes up, allow all.
5. Within JMRI, edit the Preferences from the Edit menu, choose WiThrottle on the left pane, click the "Start automatically with application" checkbox. All the Allowed Controls can be disabled.

4.3.3 Running

1. Start HostedNetworkStarter from the start menu, enter a Network Name and Network Key, then hit the Start button.
2. Start JMRI using the DecoderPro icon on the desktop

4.4 WiFi access point requirements

The wiFred will work on basically any 2.4GHz IEEE802.11b or -g WiFi network that will accept new devices and provide them with an IPv4 address through a DHCP server. It supports open networks or WPA2 encryption, though not WPS.

It has successfully been tested with a FRITZ!Box, a random cable router, multiple public WiFi networks and several software hotspots including Linux hostapd both on a first generation Intel Atom netbook and a Raspberry Pi, the Windows 10 Hotspot and a Windows 7 software as described in Section 4.3.

Note
Some WiFi access points and routers offer a feature called "Wireless Isolation" or similar that will forbid connected devices from connecting to each other. This feature should be disabled for wiFreds to find their server and to be configured.

4.5 JMRI server requirements

Any PC that can run at least Java 8 will be able to run the JMRI server. Even old 32Bit first generation netbook computers like the 2008 Eee-PC have more than enough computing power for that task. There are JMRI versions for Windows, Linux or MacOS available, so any of these operating systems should be fine.

4.6 Layout connection options

[JMRI] supports a multitude of different connections to the layout as described in [Supported Hardware]. In general, if a JMRI installation can run a loco on the layout, it's a simple matter of connecting the JMRI computer to a WiFi access point (if not already connected) and starting the wiThrottle server from the Actions-menu inside DecoderPro.

The wiFred has been tested to work with a LocoBufferUSB or a Digitrax PR3 connected via Loconet to an Intellibox, a Z21 black, a DCS 51 Zephyr xtra or a Minibox 2 DIY DCC central station. It has also been used to run a loco on a SPROG programmer.

4.7 Computer or smartphone to configure wiFred

Webbrowser, Zeroconf. Avahi. Bonjour (iTunes?). MacOS out of the box? iOS? Android?

For initial configuration of the wiFred, most of the devices mentioned above can be omitted. As shown in Figure 9, only a WiFi capable device with a web browser is required.

Figure 9: For initial configuration, the requirements are very small

Tested with iOS 13 and iOS 14, Android 9 and 10 as well as Firefox on Linux, the initial configuration page is automatically loaded when the device is connected to the **wiFred-configXXXX**-network.

For regular configuration, once the wiFred has successfully connected to a WiFi network and put into configuration mode, Bonjour/Zeroconf/mDNS support is required to find the device announcing itself as http://config.local. This service is provided on Linux by a software package called avahi, should work out of the box on iOS and MacOS and can be installed on Windows with iTunes. For mobile operating systems, there are apps called Discovery for iOS and BonjourBrowser for Android which not only translate http://config.local to point to the configuration website of the one wiFred on the network that is in configuration mode but also browse all wiFreds currently announcing their presence. Figure 10 shows a screenshot of the Android version. Clicking on a device will open a browser to its configuration website.

Note
Configuring a wiFred without putting it into configuration mode first is largely untested and may produce undesired results, particularly if you are running a train with it at the same time.

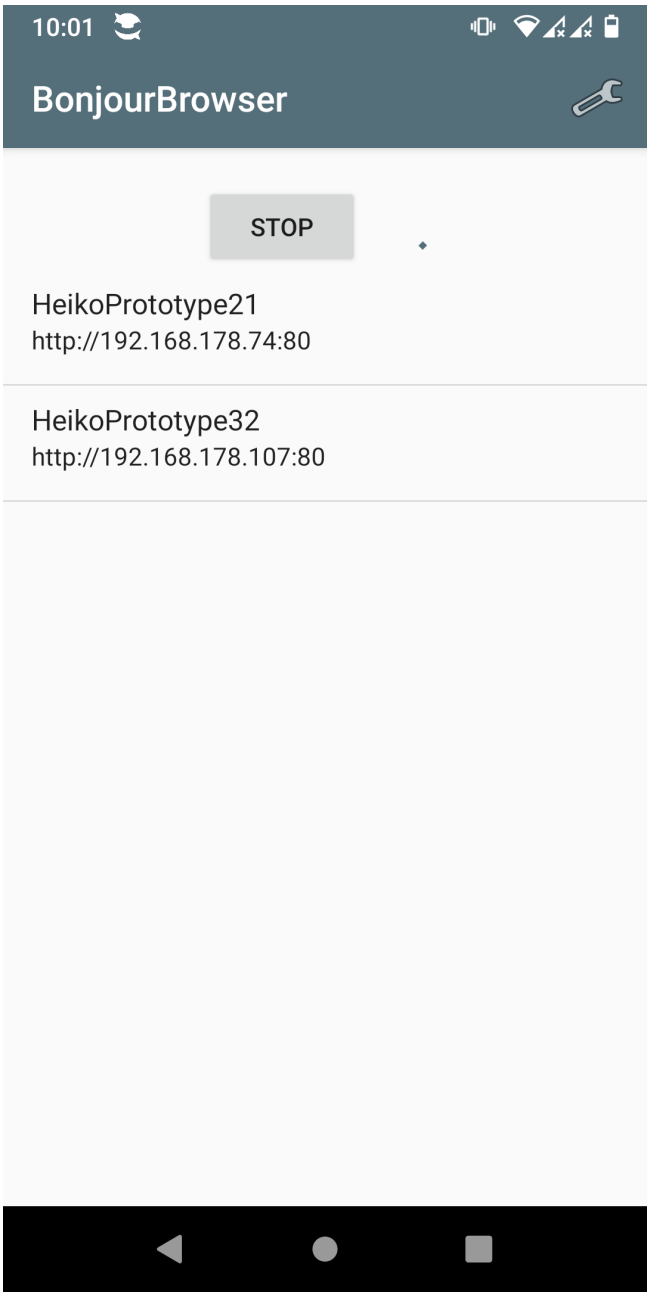


Figure 10: User interface of BonjourBrowser, showing two wiFreds active on the network.

5 Hardware description

5.1 From revision 0.6 onwards

The wiFred hardware is centered around an ESP32-S2-WROOM module that controls the LEDs and reads the switches, push buttons and speed potentiometer. It gets its power from a single cell LiPo battery through a 3.3V low drop linear voltage regulator that is activated directly from the loco selection switches combined through a set of diodes. An RC lowpass will delay turning off the wiFred by a couple of seconds.

Optionally, two white 5 mm-LEDs protruding from the top of the PCB can be installed to serve as a flashlight. They are driven by a constant-current source directly from the battery and enabled when pushing the yellow SHIFT key.

The ESP32-S2 also monitors the battery voltage, going into sleep mode when the battery voltage drops too low. This should drop power consumption to a reasonably safe value, even though not as low as it can go when all the loco selection switches are off. Charging the battery can be done through the integrated USB port with two LEDs showing the charging status.

The schematic is split into several pages and can be found in Figure 11 to Figure 13. It has been created with kicad and is available on the [Github repository for wiFred] along with the PCB design.

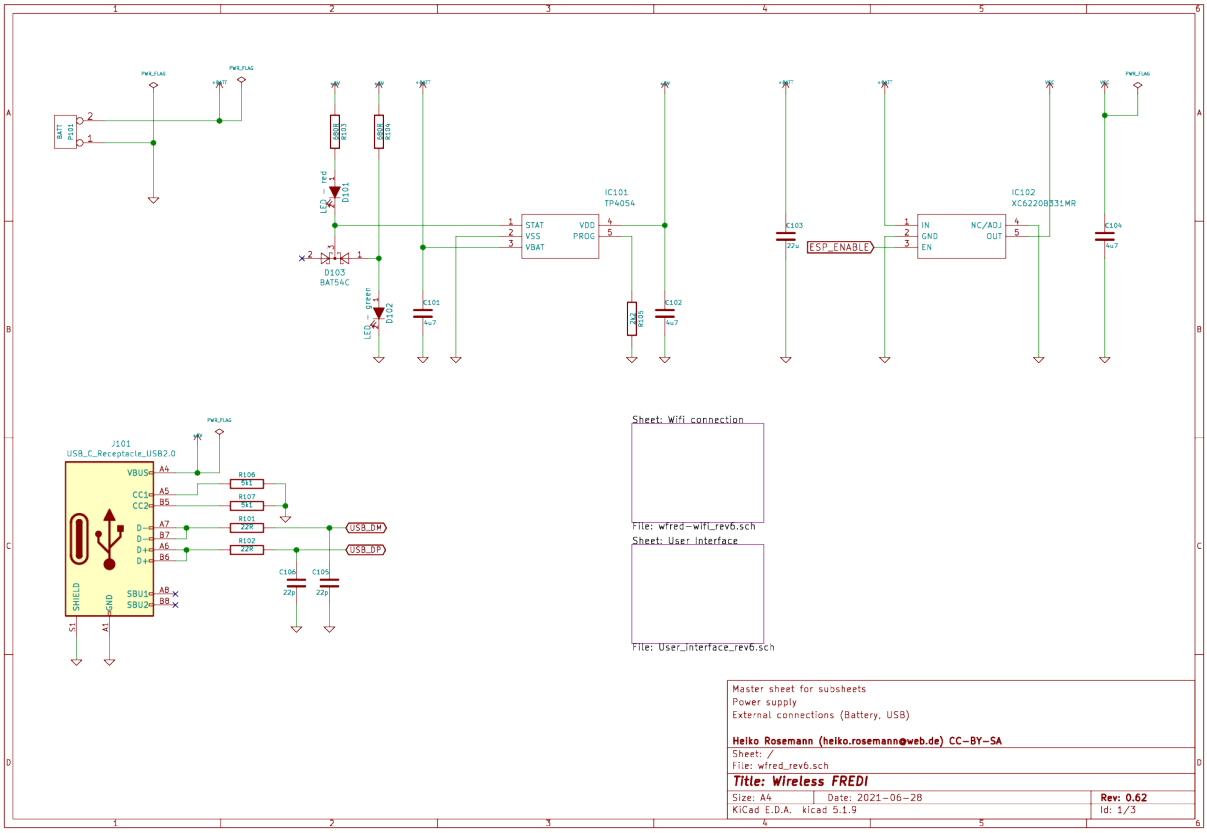


Figure 11: Master schematic sheet with battery connector, charging circuit and power supply

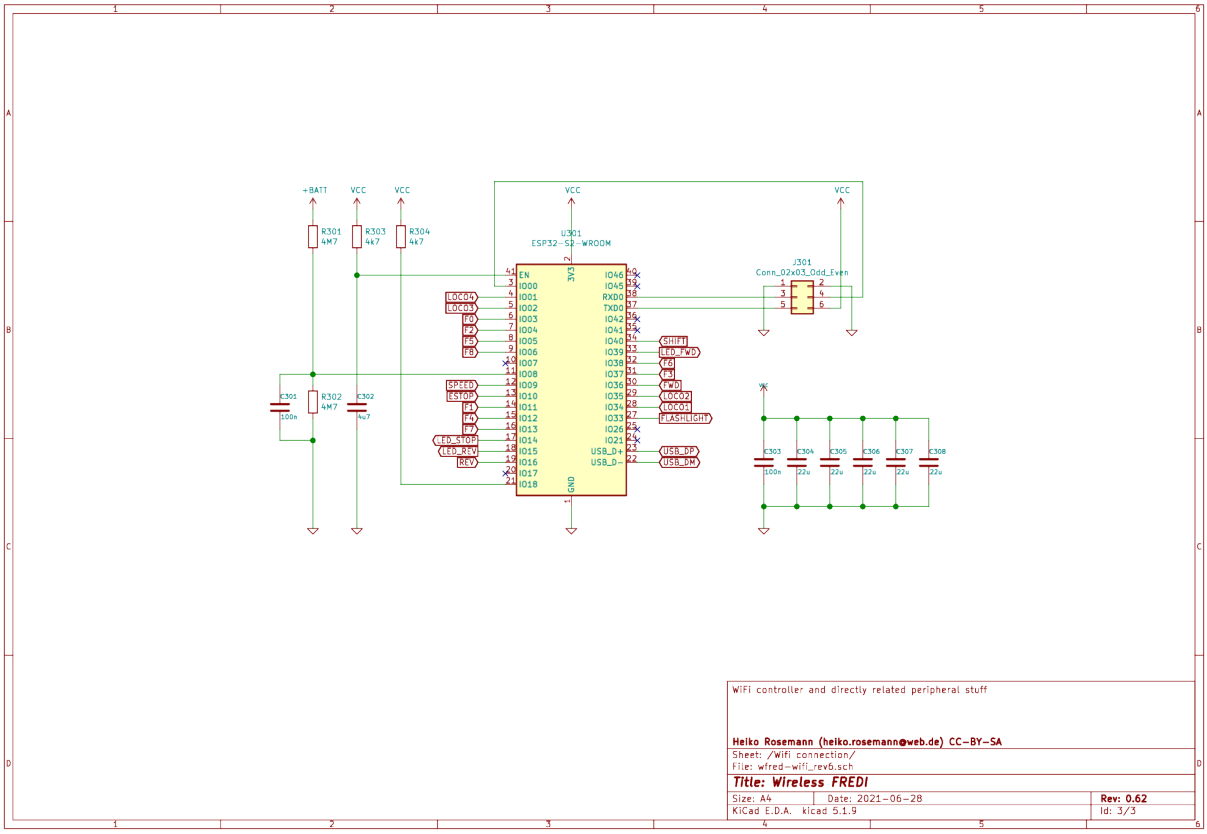


Figure 12: Schematic sheet including ESP32-S2-WROOM for WiFi connection with bootloader enabling jumper and connection to programming cable

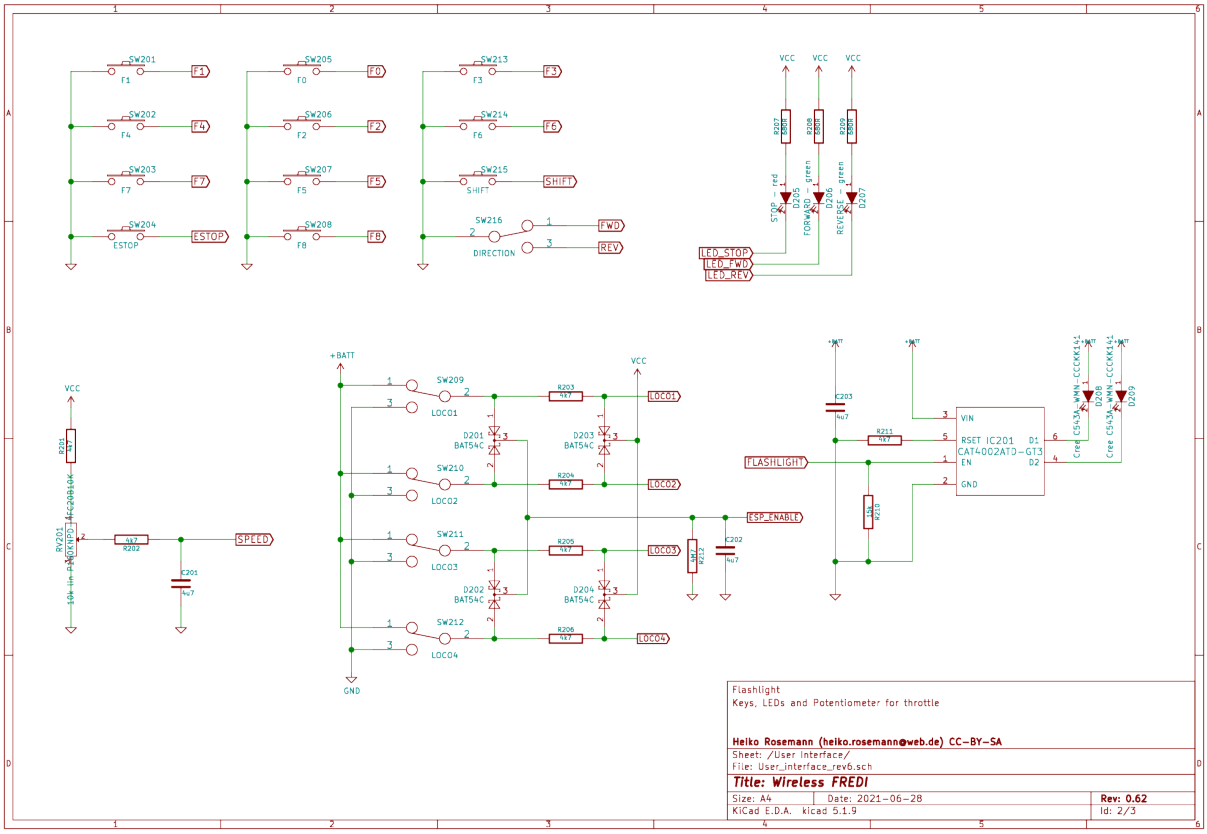


Figure 13: Schematic sheet for user interface (Keys, switches, LEDs, poti and flashlight)

5.2 Writing the firmware to the ESP32-S2-WROOM

To place the ESP32-S2 into firmware flashing mode, a jumper has to be installed on the BOOT pins as shown in Figure 14 when the device is powered up. After entering boot mode, the ESP32-S2 will register as a USB CDC device when connected to a PC and can be programmed through it. Alternately, the serial port at the TXD/RXD pins can be used. The serial port needs to be at 3.3 V-levels like from an FTDI232-device run at 3.3 V.

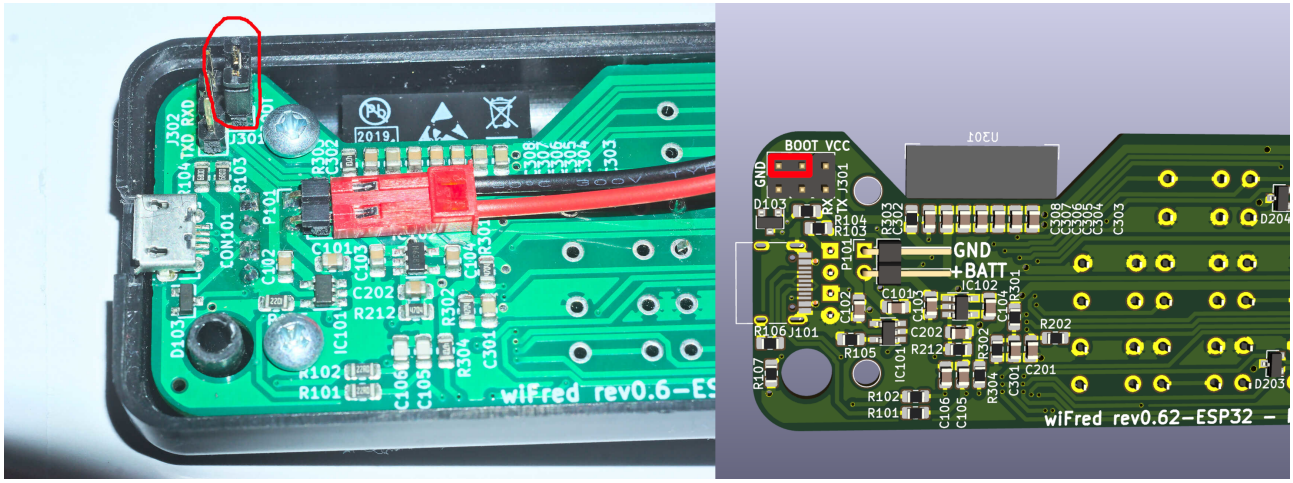


Figure 14: Display of boot jumper - left from 0.6 prototype, right from 0.62 rendering

All files in the **software/esp-firmware**-subdirectory of the [Github repository for wiFred] need to be placed in a folder, then the main sketch **esp-firmware.ino** needs to be opened with the Arduino IDE. Settings for the Arduino IDE can be found inside the main file, programming the device should work using the **Upload**-button in the **Sketch**-menu. An alternate solution using a pre-compiled firmware and esptool.py may be documented in the future.

5.3 Hints for building revision 0.6

All the SMD parts and connectors in Table 2 are located on the bottom side, so they can easily be installed by an automated PCB assembly service. Gerber files and location files will be made available on the [Github repository for wiFred]. They are no smaller than 0805, though, so it should be possible for experienced electronics hobbyists to hand-solder them as well, although the USB-C-connector has a very small pitch.

Then - as shown in Table 3 - there is only the ESP32-S2-WROOM module on the top of the PCB and all the switches, push buttons, potentiometer and LEDs. After installing the ESP32-S2-WROOM, the direction switch should be screwed into the PCB with an 8 mm hex nut first, then attached to it's pads using the cutoffs from the LEDs, and the speed potentiometer should be screwed into the PCB with a 10 mm hex nut before soldering its leads.

For all other switches, push buttons and LEDs, it is recommended to drill the holes in the housing first, then place the parts on the PCB and screw the PCB into the housing before soldering the parts. Holes for the pushbutton switches should be drilled at 3.5 mm diameter. Holes for the LEDs should be drilled at 3 mm diameter and holes for the speed potentiometer at 8 mm, for the direction switch at 6.5 mm diameter. The cutouts for the loco selection switches are best drilled at 1.5 mm and extended to fit with a jigsaw or a sharp hobby knife and a file. The easiest approach is a laser engraved housing - PDF and DXF files of the markings are available on the [Github repository for wiFred] as well.

To form a complete BOM, also include the parts listed in Table 4 which are not soldered to the PCB but used in assembly later on.

After assembling the PCB with all the components, the holes and cutouts in the enclosure most likely will have to be reworked / extended to actually fit the PCB, then the PCB can be screwed into the enclosure with four screws. Afterwards the battery should be connected to the BATT connector making sure the orientation is correct as shown in Figure 15 and printed on the PCB, then the battery should be glued to the bottom of the enclosure with double-sided tape so it does not collide with any parts on the PCB, particularly the pin headers and the direction switch.

Designator	Package	Designation
C106, C105	C_0805_2012Metric_Pad1.15x1.40mm_HandSolder	22p
C201, C302, C102, C203, C202, C104, C101	C_0805_2012Metric_Pad1.15x1.40mm_HandSolder	4u7
C303, C301	C_0805_2012Metric_Pad1.15x1.40mm_HandSolder	100n
C308, C307, C306, C305, C103, C304	C_0805_2012Metric_Pad1.15x1.40mm_HandSolder	22u
D103, D204, D203, D202, D201	SOT-23	BAT54C
IC101	SOT-23-5_HandSoldering	TP4054
IC102	SOT-23-5_HandSoldering	XC6220B331MR
IC201	SOT-23-6_Handsoldering	CAT4002ATD-GT3
J101	USB_C_Receptacle_HRO_TYPE-C-31-M-12	USB_C_Receptacle_USB2.0
J301	PinHeader_2x03_P2.54mm_Vertical	Conn_02x03_Odd_Even
P101	PinHeader_1x02_P2.54mm_Horizontal	BATT
R102, R101	R_0805_2012Metric_Pad1.15x1.40mm_HandSolder	22R
R103, R209, R208, R207, R104	R_0805_2012Metric_Pad1.15x1.40mm_HandSolder	680R
R105	R_0805_2012Metric_Pad1.15x1.40mm_HandSolder	2k2
R107, R106	R_0805_2012Metric_Pad1.15x1.40mm_HandSolder	5k1
R212, R302, R301	R_0805_2012Metric_Pad1.15x1.40mm_HandSolder	4M7
R303, R304, R206, R211, R205, R204, R203, R202, R201	R_0805_2012Metric_Pad1.15x1.40mm_HandSolder	4k7

Table 2: List of bottom side components - SMDs and pin headers - for the wiFred PCB from revision 0.6

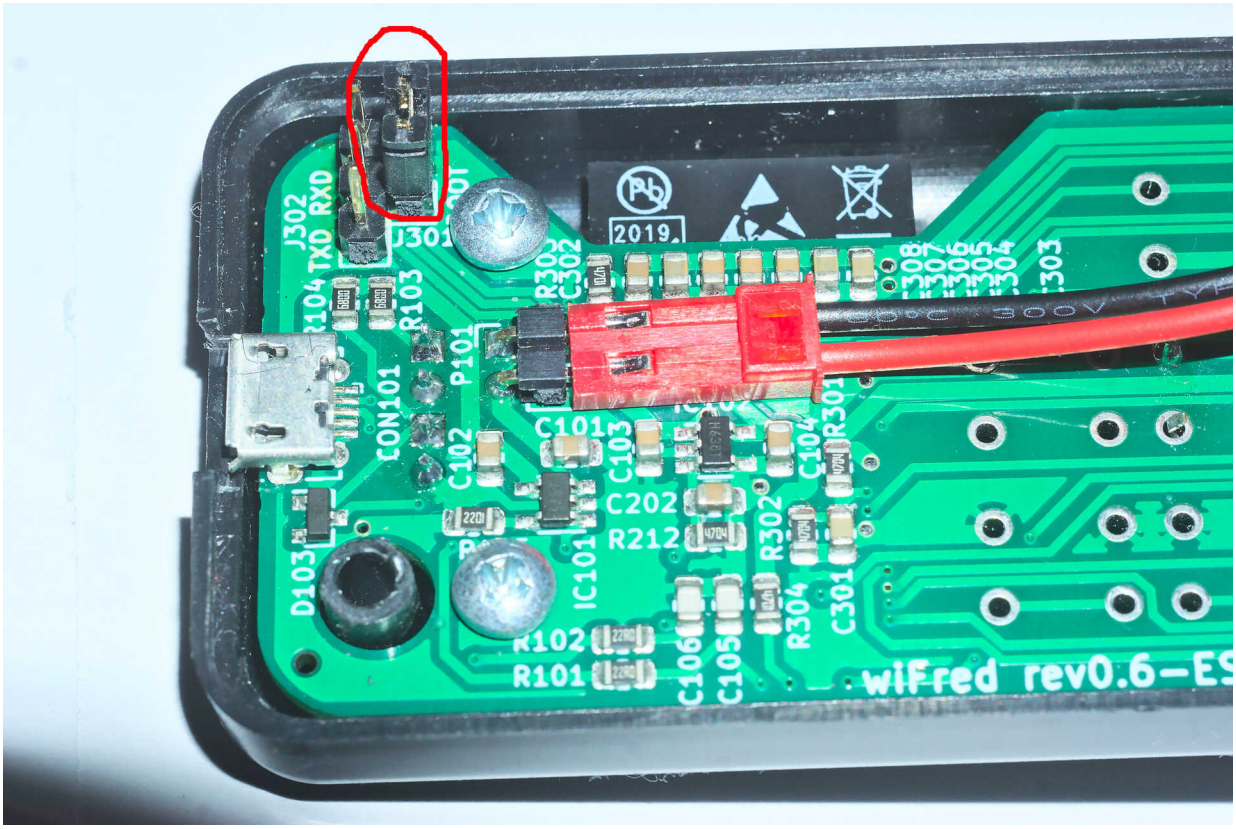


Figure 15: Connection of battery to P101 on revision 0.6 and up - black wire is GND, red wire is positve

Designator	Package	Designation
D101	LED_D3.0mm	LED - red
D102	LED_D3.0mm	LED - green
D205	LED_D3.0mm	STOP - red
D206	LED_D3.0mm	FORWARD - green
D207	LED_D3.0mm	REVERSE - green
D208	LED_D5.0mm_Horicontal_FLIPPED_O1.27mm	Cree C543A-WMN-CCCKK141
D209	LED_D5.0mm_Horizontal_O1.27mm_Z3.0mm_Clear	Cree C543A-WMN-CCCKK141
RV201	P160KNPD	10k lin P160KNPD-4FC20B10K
SW201	SW_PUSH_6mm_H9.5mm	F1
SW202	SW_PUSH_6mm_H9.5mm	F4
SW203	SW_PUSH_6mm_H9.5mm	F7
SW204	SW_PUSH_6mm_H9.5mm	ESTOP
SW205	SW_PUSH_6mm_H9.5mm	F0
SW206	SW_PUSH_6mm_H9.5mm	F2
SW207	SW_PUSH_6mm_H9.5mm	F5
SW208	SW_PUSH_6mm_H9.5mm	F8
SW209	OS102011MS2Q	LOCO1
SW210	OS102011MS2Q	LOCO2
SW211	OS102011MS2Q	LOCO3
SW212	OS102011MS2Q	LOCO4
SW213	SW_PUSH_6mm_H9.5mm	F3
SW214	SW_PUSH_6mm_H9.5mm	F6
SW215	SW_PUSH_6mm_H9.5mm	SHIFT
SW216	100SP1T1B1M1QEH	DIRECTION
U301	ESP32-S2-WROVER-HandSoldering	ESP32-S2-WROOM

Table 3: List of top side components - ESP32-S2-WROOM and user interface - for the wiFred PCB from revision 0.6

Designator	Package	Designation
B1	Battery	Lithium battery 1700mAh
H1a	Housing black	Strapubox 2090
or H1b	Housing white	Strapubox 2090
J1	Jumper	
K1a	Potentiometer Knob silver	24mm
or K1b	Potentiometer Knob black	24mm
P1	PCB	124mm x 35mm x 1.6mm
S1, S2, S3, S4	Mounting Screws	2,9mm x 6,5mm

Table 4: List of components for the wiFred from revision 0.6 excluding electronic parts to solder to PCB

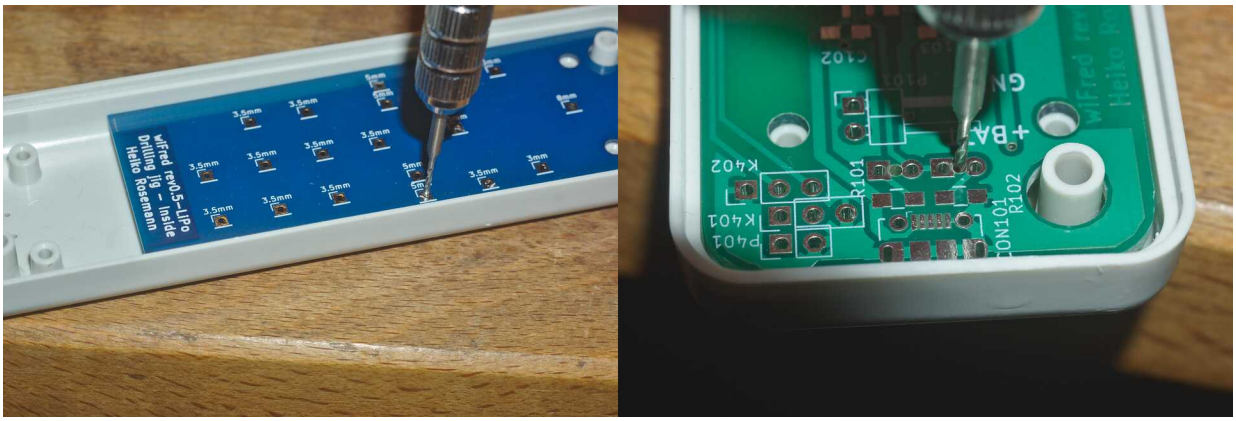


Figure 16: Using the original PCB and drilling jig to transfer the positions of the holes to the housing - better results will be achieved when the PCBs are screwed in position

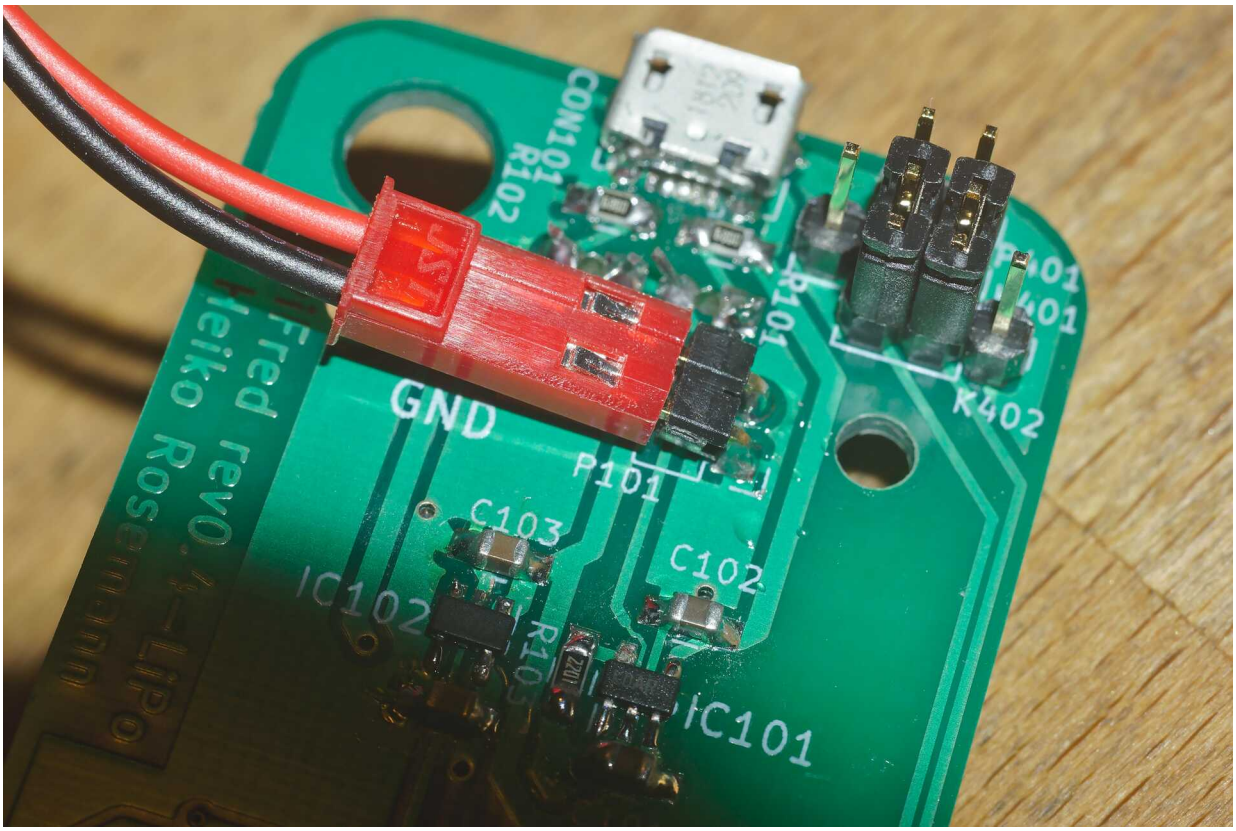


Figure 17: Connection of battery to P101 on revision 0.4 - black wire is GND, red wire is positive

5.4 Up to revision 0.5

The wiFred hardware is centered around an ESP8266 for the WiFi connection. The ESP8266 communicates through its serial port with an ATMega 328P microcontroller which manages the power, controls the LEDs, reads the loco selection switches, speed potentiometer, direction switch and pushbutton switches for functions and emergency stop. The communication goes through a 2x3 pin header which enables the user to connect a programming cable to the same serial port if removing the jumpers.

Optionally, two white 5 mm-LEDs protruding from the top of the PCB can be installed to serve as a flashlight. They are driven by a constant-current source directly from the battery and enabled when pushing the yellow SHIFT key.

The wiFred is powered by a single cell LiPo battery. The ATMega 328P is connected directly to the LiPo cell, going into sleep mode when no loco selection switch is active, thereby reducing the power consumption to less than 1 μA. The ESP8266 is powered by a low-drop linear voltage regulator with an output voltage of 3 V which is disabled by the ATMega 328P when the device goes into standby.

The schematic is split into several pages and can be found in Figure 18 to Figure 21. It has been created with kicad and is available on the [Github repository for wiFred] along with the PCB design.

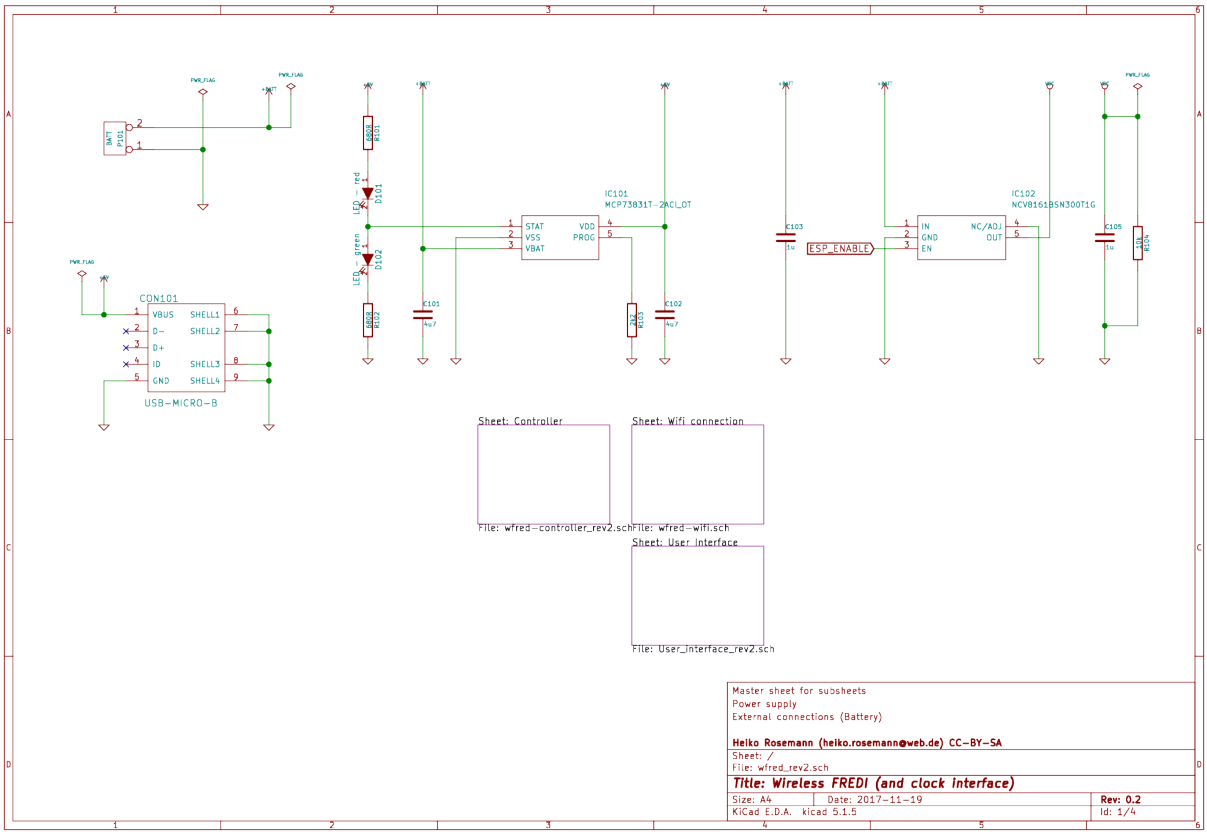


Figure 18: Master schematic sheet with battery connector, charging circuit and power supply

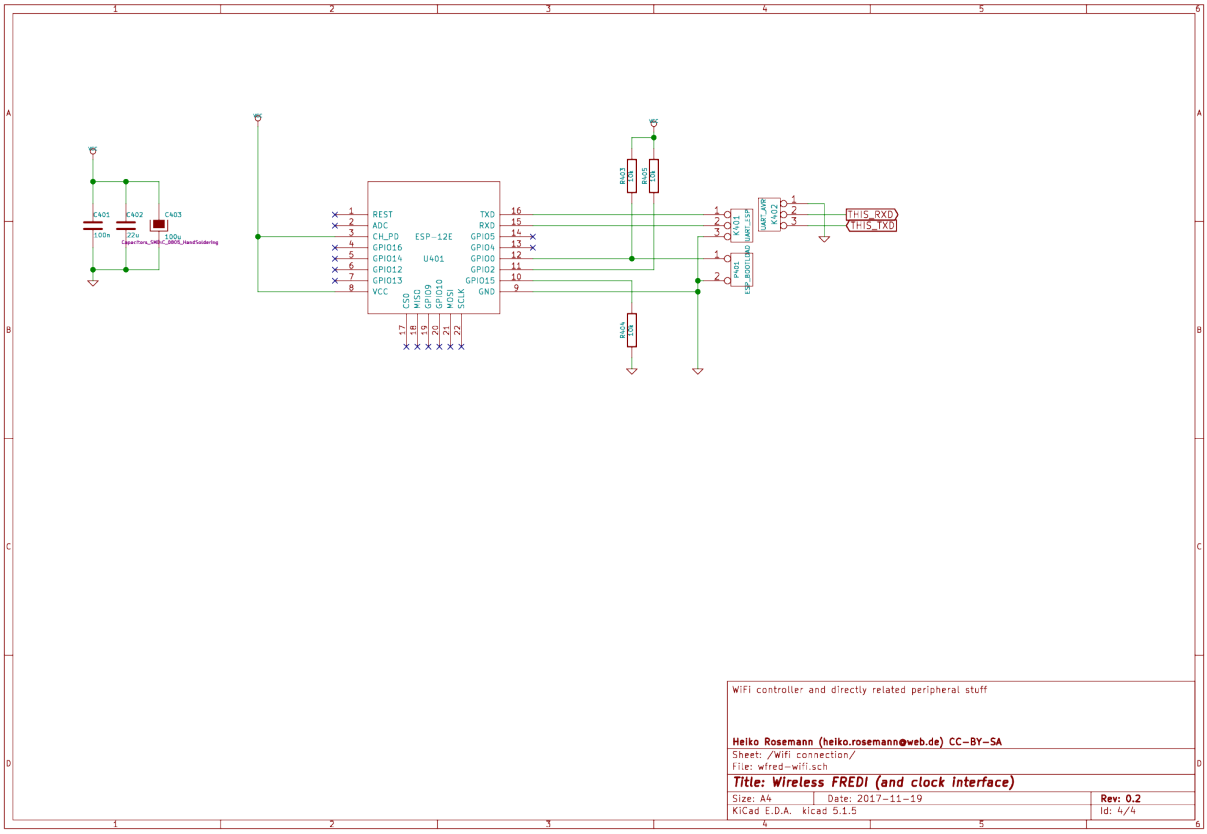


Figure 19: Schematic sheet including ESP8266 for WiFi connection with bootloader enabling jumper and connection to programming cable

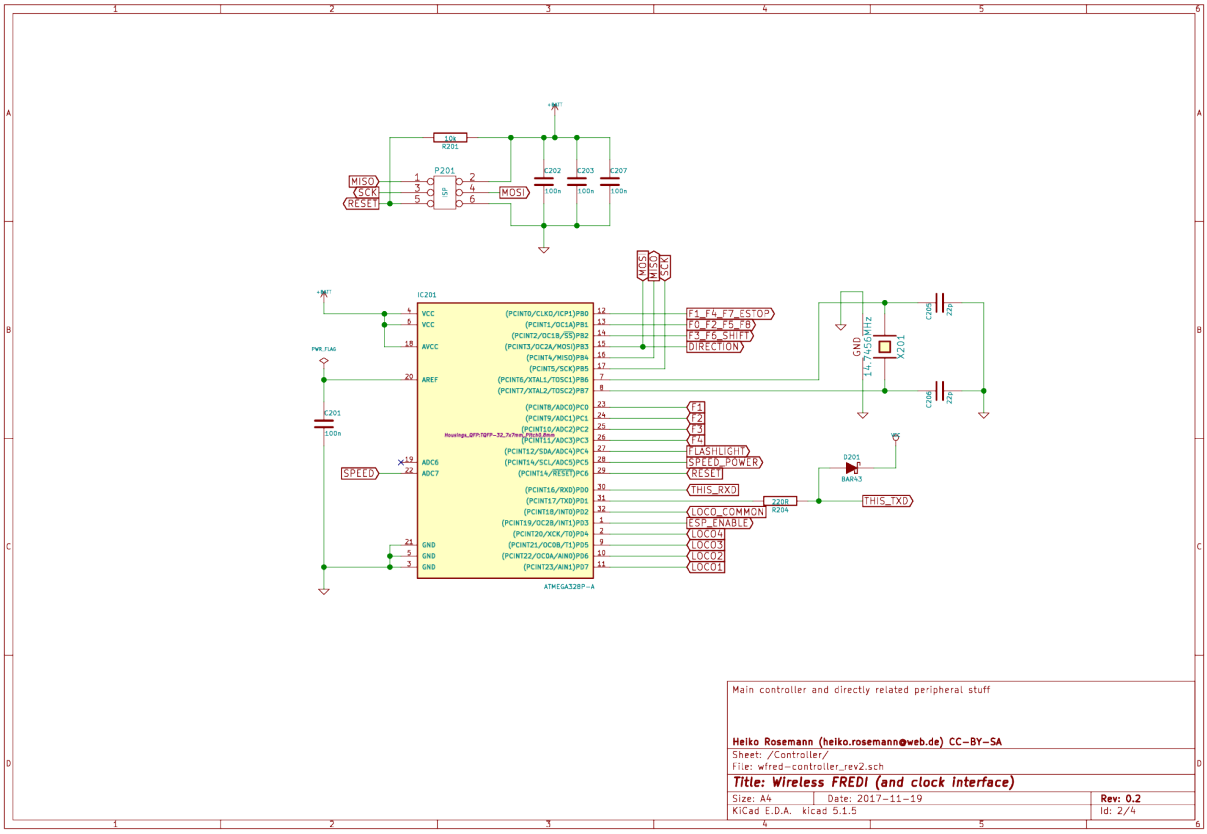


Figure 20: Schematic sheet including ATMega 328P along with crystal and in system programming header

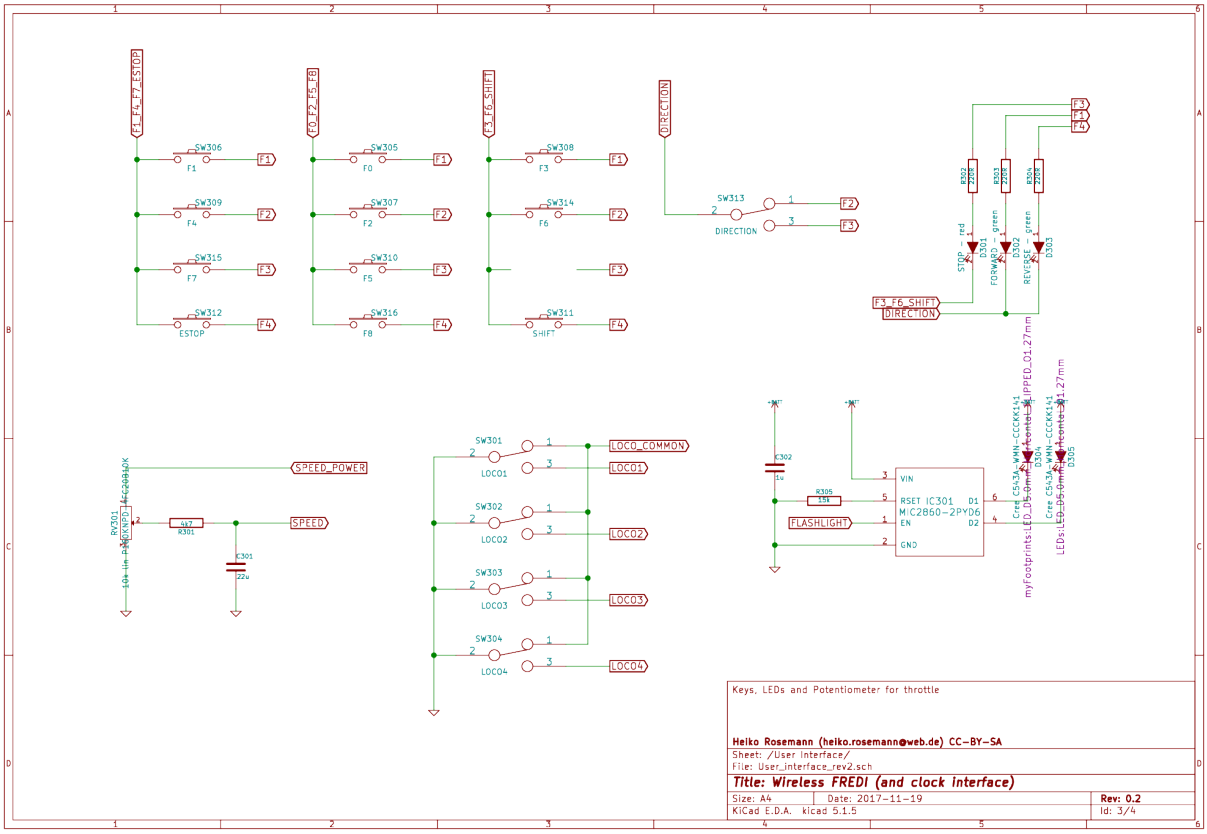


Figure 21: Schematic sheet including pushbutton switches, loco selection switches, direction switch, speed potentiometer and flashlight LEDs with controller

5.5 Hints for building revisions up to 0.5

The PCB has holes in the center of the LED footprints to enable transferring their positions to a StrapuBox housing with a sharp needle or to drill pilot holes with a 1 mm drill. For all other holes, there is a drill jig available which also allows the drilling of pilot holes for the pushbutton switches, the direction control switch and the speed potentiometer. Figure 16 shows the process and its results. Holes for the pushbutton switches should be drilled at 3.5 mm diameter. Holes for the LEDs should be drilled at 3 mm diameter and holes for the speed potentiometer at 8 mm, for the direction switch at 6.5 mm diameter. The cutouts for the loco selection switches are best drilled at 1.5 mm and extended to fit when the PCB is assembled with a jigsaw or a sharp hobby knife and a file.

The remaining assembly is a basic exercise in installing all the components to the PCB, listed in Table 5. From assembling the prototypes, the suggested order of installing the components is as follows:

1. IC101, IC102, IC201 (note: Rotate PCB so Designator is right side up, then Pin 1 is on top left) and IC301
2. X201 and D201
3. USB connector CON101
4. Capacitors and Resistors in 0805 size (first those on the same side as the items before)
5. U401
6. Capacitors and Resistors not installed before - that is R403, R404, R405, C401, C402 and C403

Designator	Package	Designation
C102, C101	C_0805_HandSoldering	4u7
C105, C103, C302	C_0805_HandSoldering	1u
C206, C205	C_0805_HandSoldering	22p
C401, C203, C202, C201, C207	C_0805_HandSoldering	100n
C402, C301	C_0805_HandSoldering	22u
C403	C_0805_HandSoldering	100u
CON101	USB_Micro-B_Molex-105017-0001	USB-MICRO-B
D101	LED_D3.0mm	LED - red
D102	LED_D3.0mm	LED - green
D201	SOT-23_Handsoldering	BAR43
D301	LED_D3.0mm	STOP - red
D302	LED_D3.0mm	FORWARD - green
D303	LED_D3.0mm	REVERSE - green
D303, D302, D301, D101, D102	LED Spacer	3mm
D304	LED_D5.0mm_Horicontal_FLIPPED_O1.27mm	LED white
D305	LED_D5.0mm_Horicontal_O1.27mm	LED white
IC101	SOT95P270X145-5N	MCP73831T-2ACI_OT
IC102	SOT95P275X110-5N	NCV8161BSN300T1G
IC201	TQFP-32_7x7mm_Pitch0.8mm	ATMEGA328P-A
IC301	SOT-23-6_Handsoldering	MIC2860-2PYD6
K401	Pin_Header_Straight_1x03_Pitch2.54mm	UART_ESP
K402	Pin_Header_Straight_1x03_Pitch2.54mm	UART_AVR
P1	PCB	124mm x 35mm x 1.6mm
P101	Pin_Header_Angled_1x02_Pitch2.54mm	BATT
P201	Pin_Header_Straight_2x03_Pitch2.54mm_SMD	ISP
P401	Pin_Header_Straight_1x02_Pitch2.54mm	ESP_BOOTLOAD
R101, R102	C_0805_HandSoldering	680R
R103	C_0805_HandSoldering	2k2
R301	C_0805_HandSoldering	4k7
R304, R303, R302, R204	C_0805_HandSoldering	220R
R305	C_0805_HandSoldering	15k
R405, R404, R403, R201, R104	C_0805_HandSoldering	10k
RV301	P160KNPD	10k lin P160KNPD-4FC20B10K
SW301	OS102011MS2Q	LOCO1
SW302	OS102011MS2Q	LOCO2
SW303	OS102011MS2Q	LOCO3
SW304	OS102011MS2Q	LOCO4
SW305	SW_SPST_PTS645	F0
SW306	SW_SPST_PTS645	F1
SW307	SW_SPST_PTS645	F2
SW308	SW_SPST_PTS645	F3
SW309	SW_SPST_PTS645	F4
SW310	SW_SPST_PTS645	F5
SW311	SW_SPST_PTS645	SHIFT
SW312	SW_SPST_PTS645	ESTOP
SW313	100SP1T1B1M1QEH	DIRECTION
SW314	SW_SPST_PTS645	F6
SW315	SW_SPST_PTS645	F7
SW316	SW_SPST_PTS645	F8
U401	ESP-12E_SMD	ESP-12E
X201	Crystal_SMD_TXC_7M-4pin_3.2x2.5mm_HandSoldering	14.7456MHz

Table 5: List of components for the wiFred PCB up to revision 0.5

7. Pushbutton switches SW305 to SW312 and SW314 to SW316 - taking care to place the red one at SW312 and the yellow one at SW311
8. Pin headers K401, K402 and P401 (correct alignment of K401 and K402 can be assured by adding a jumper before soldering)
9. Pin headers P101 and P201
10. Loco selection switches SW301 to SW304
11. LEDs D101, D102 and D301 to D303 with 3mm spacers to the PCB - making sure the Anode (long pin) is aligned with the square pad on all of them
12. LEDs D304 and D305 - making sure the Anode (long pin) is aligned with the square pad on both, they can be installed on top or bottom of the PCB as desired
13. Direction switch SW313 (screwed into the PCB with an 8 mm hex nut first, then attached to it's pads using the cutoffs from D301, D302 and D303) and Speed potentiometer RV301 (screwed into the PCB with a 10 mm hex nut first)

To form a complete BOM, also include the parts listed in Table 6 which are not soldered to the PCB but used in assembly later on.

Designator	**Package**	**Designation**
B1	Battery	Lithium battery 1700mAh
H1a	Housing black	Strapubox 2090
or H1b	Housing white	Strapubox 2090
J1, J2	Jumper	
K1a	Potentiometer Knob silver	24mm
or K1b	Potentiometer Knob black	24mm
P1	PCB	124mm x 35mm x 1.6mm
S1, S2, S3, S4	Mounting Screws	2,9mm x 6,5mm

Table 6: List of components for the wiFred up to revision 0.5 excluding electronic parts to solder to PCB

After assembling the PCB with all the components, the holes and cutouts in the enclosure most likely will have to be reworked / extended to actually fit the PCB, then the PCB can be screwed into the enclosure with four screws. Afterwards the battery should be connected to P101 making sure the orientation is correct as shown in Figure 17 and printed on the PCB, then the battery should be glued to the bottom of the enclosure with double-sided tape so it does not collide with any parts on the PCB, particularly P101 and SW313. Finally, both the ATMega 328P and the ESP8266 will need to be programmed with firmware as described in Section 5.6.

5.6 Writing the firmware to revision 0.5 and older wiFreds

5.6.1 AVR firmware

The ATMega 328P is programmed using the regular AVR ISP connection on P201. Pin 1 - GND - is towards the PCB edge, as shown in Figure 22. An ISP dongle with either automatic voltage selection or 3.3 V supply voltage should be used to avoid placing too high voltage on the ESP8266, which can only support 3.3 V power. The firmware for the ATMega 328P can be found in the **software/avr-firmware**-subdirectory of the [Github repository for wiFred] with both a precompiled hexfile and all source code including a Makefile to recompile as needed. After writing the firmware file and the eeprom file, also the fuse bits need to be set properly as detailed in the **main.c**-file.

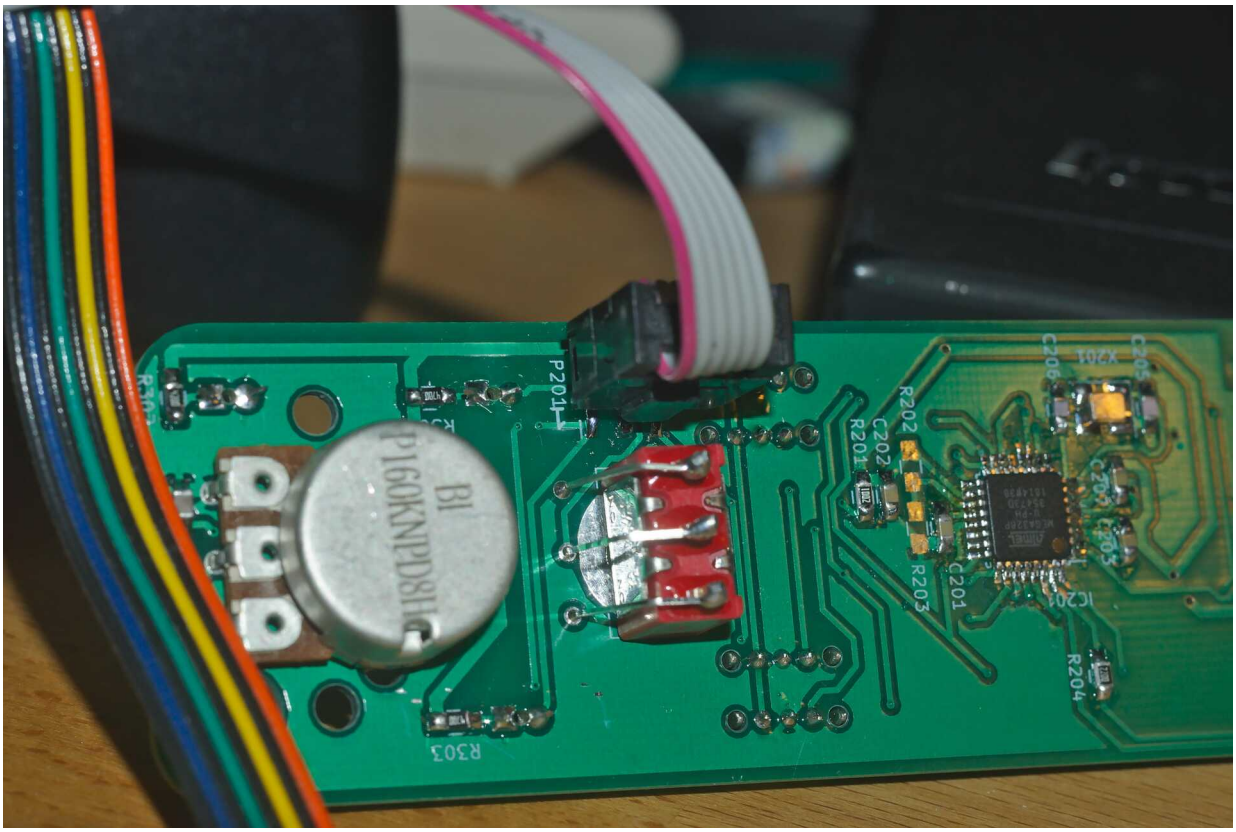


Figure 22: Programming connection for ATMega_328P - Pin 1 on purple cable

5.6.2 ESP8266 firmware

The ESP8266 is programmed using the Arduino IDE connected via a serial or USB-to-serial port to the K401 header as shown in Figure 23. The serial port needs to be at 3.3 V-levels like from an FTDI232-device run at 3.3 V. To program the ESP8266, first the ATMega 328P has to be programmed, a battery has to be connected and reasonably charged and one of the loco selection switches needs to be moved to the "enabled" position to power up the ESP8266.

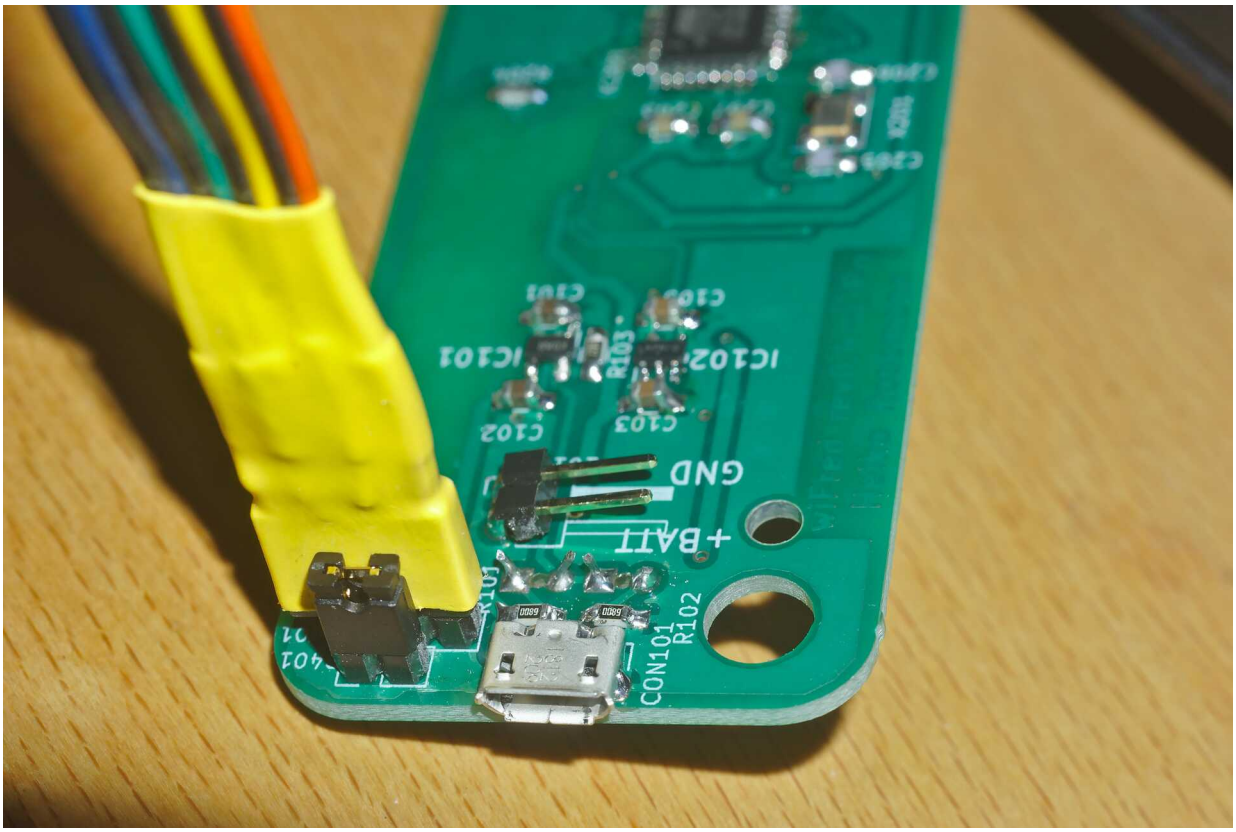


Figure 23: Programming connection for ESP8266 - GND on orange wire, then TXD of programming cable (RXD of ESP8266), then RXD of programming cable (TXD of ESP8266) - also note the jumper on P401

All files in the **software/esp-firmware**-subdirectory of the [Github repository for wiFred] need to be placed in a folder, then the main sketch **arduino_main_sketch.ino.ino** needs to be opened with the Arduino IDE. Settings for the Arduino IDE can be found inside the main file, programming the device should work using the **Upload**-button in the **Sketch**-menu.

To put the ESP8266 into programming mode, a jumper needs to be placed across the P401 header before powering up the ESP8266 by enabling one of the loco selection switches to start the device in programming mode. The red STOP LED should start flashing and the bootloader should show some results on the serial port and during download the LED on the ESP8266 module should flash as well.

After programming, two jumpers need to be placed between the K401 and K402 pin headers to re-enable communication between the ESP8266 and the ATMega 328P as shown in Figure 24.

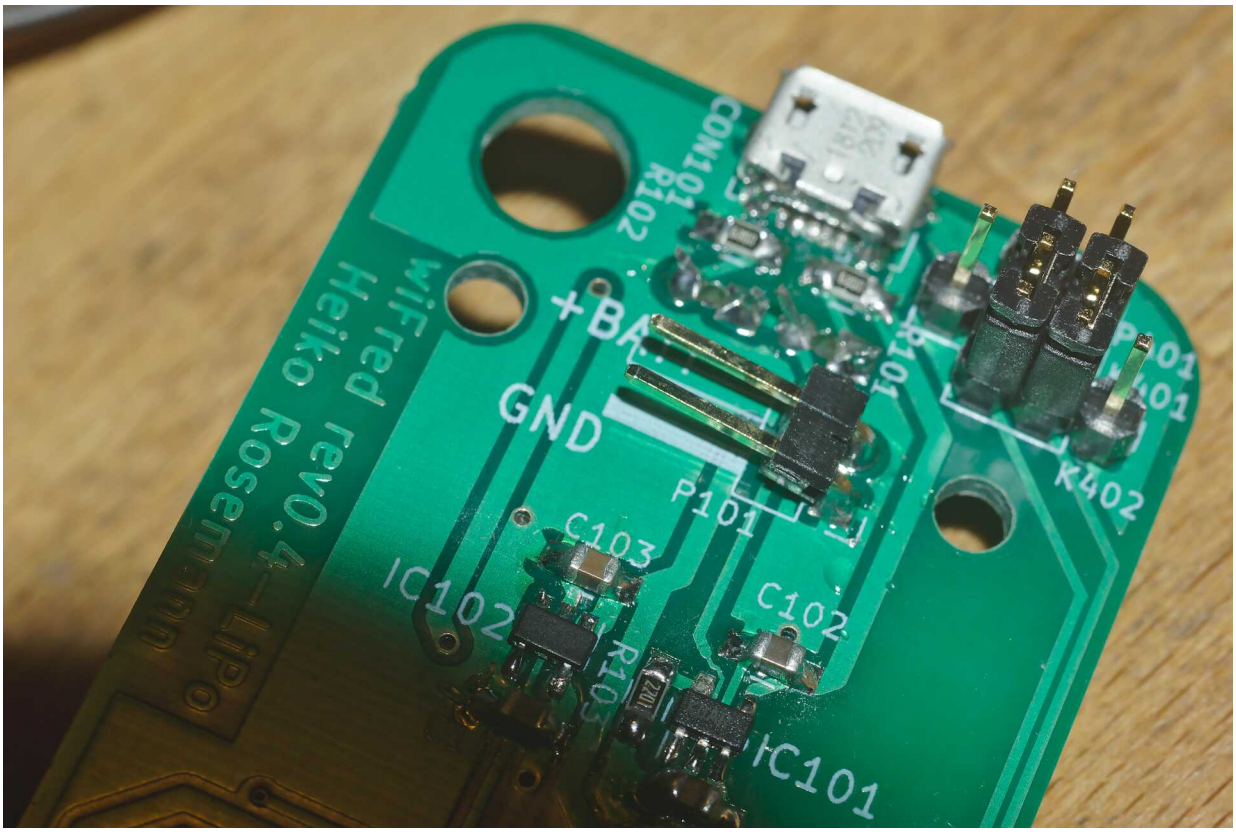


Figure 24: Communication jumpers for connecting the ESP8266 and the ATMega 328P

5.7 AA battery first prototype

The wiFred prototype hardware is centered around an ESP8266 for the WiFi connection. The ESP8266 also reads the loco selection switches and the battery voltage and communicates through it's serial port with an ATMega 328P microcontroller which controls the LEDs, reads the speed potentiometer, direction switch and pushbutton switches for functions and emergency stop. The communication goes through a 2x3 pin header which enables the user to connect a programming cable to the same serial port if removing the jumpers.

The wiFred prototype is powered by two AA size battery cells connected to a step-up converter creating 3.3 V for the entire device. As soon as a pair of batteries is inserted into the battery compartment as the symbols inside the battery compartment show, the throttle will boot up and try to connect to a wireless network. The throttle will not be damaged if batteries are inserted wrongly, but it will not work either. Use NiMH- or primary AA cells with 1.2 V to 1.5 V nominal voltage, low self discharge NiMH cells like Eneloop or similar are recommended.

The schematic is split into several pages and can be found in Figure 25 to Figure 28. It has been created with kicad and is available on the [Github repository for wiFred] along with the PCB design.

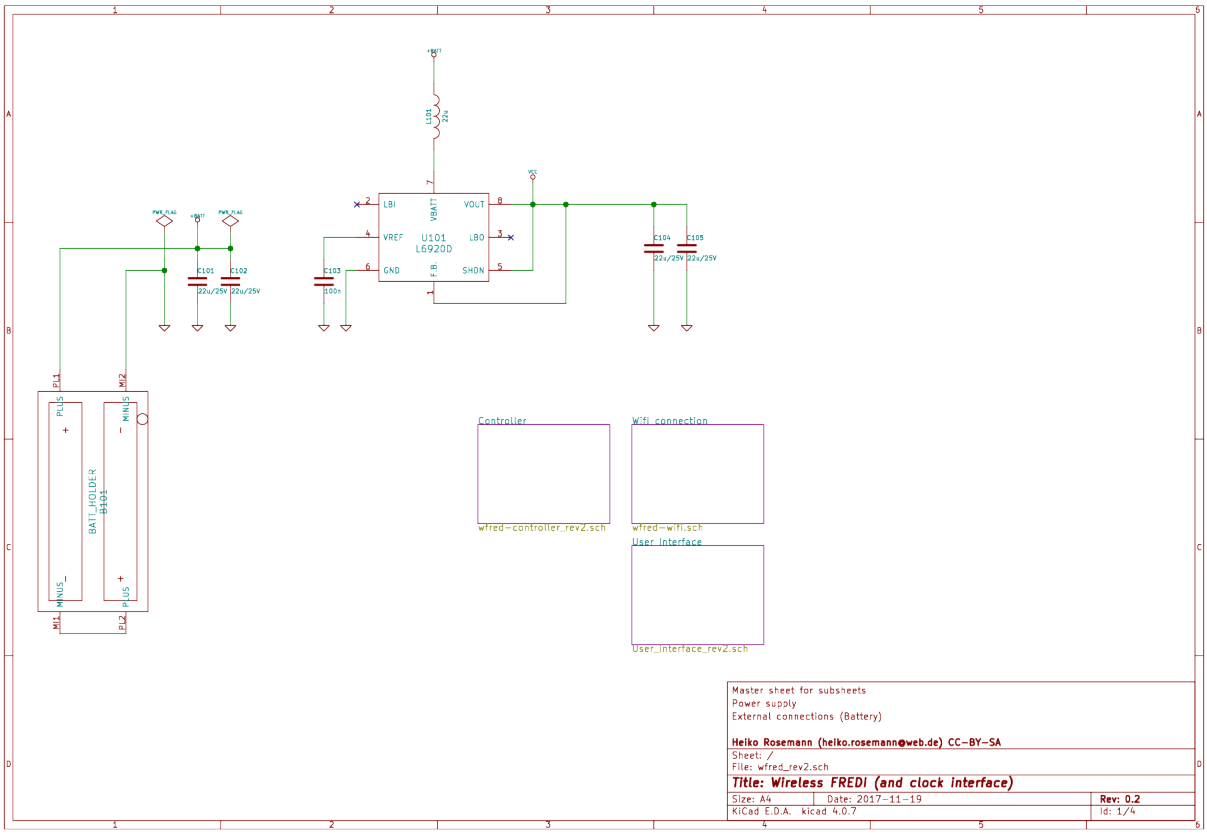


Figure 25: Master schematic sheet with batteries and power supply

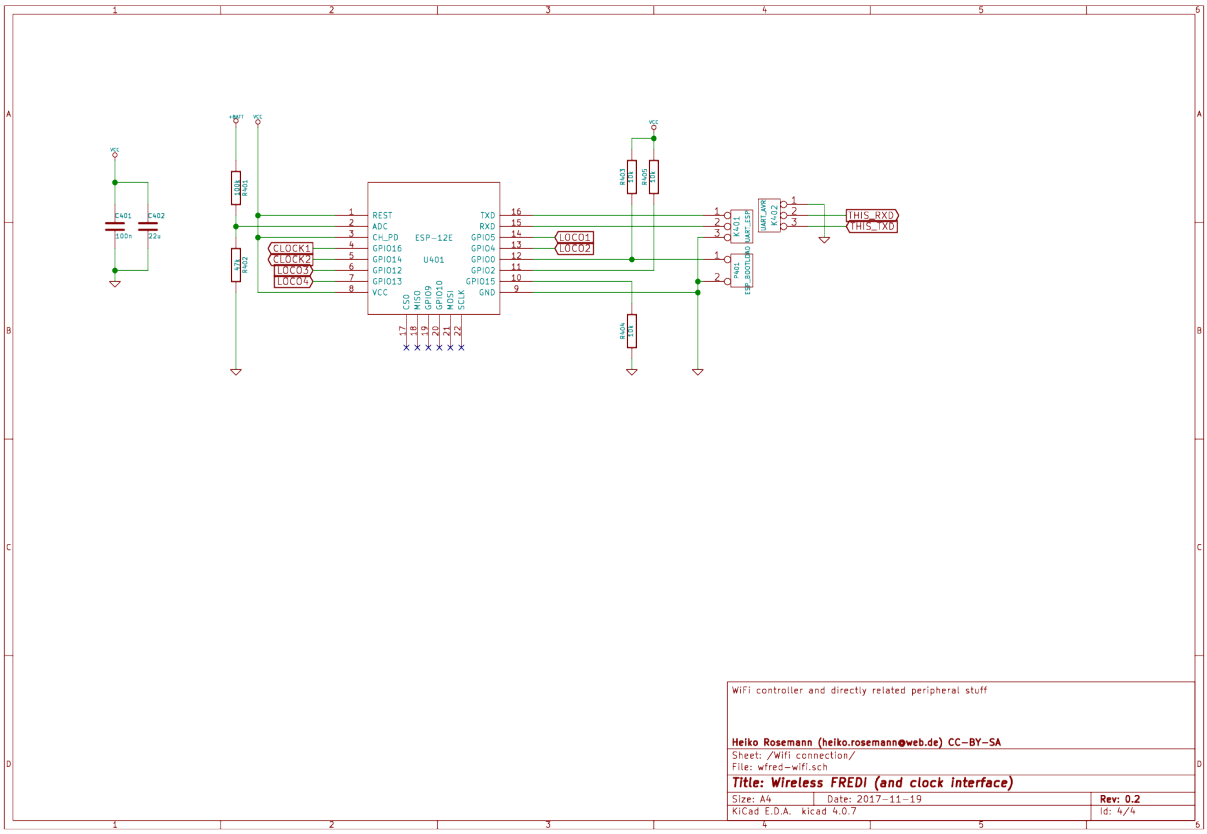


Figure 26: Schematic sheet including ESP8266 for WiFi connection with bootloader enabling jumper and connection to programming cable

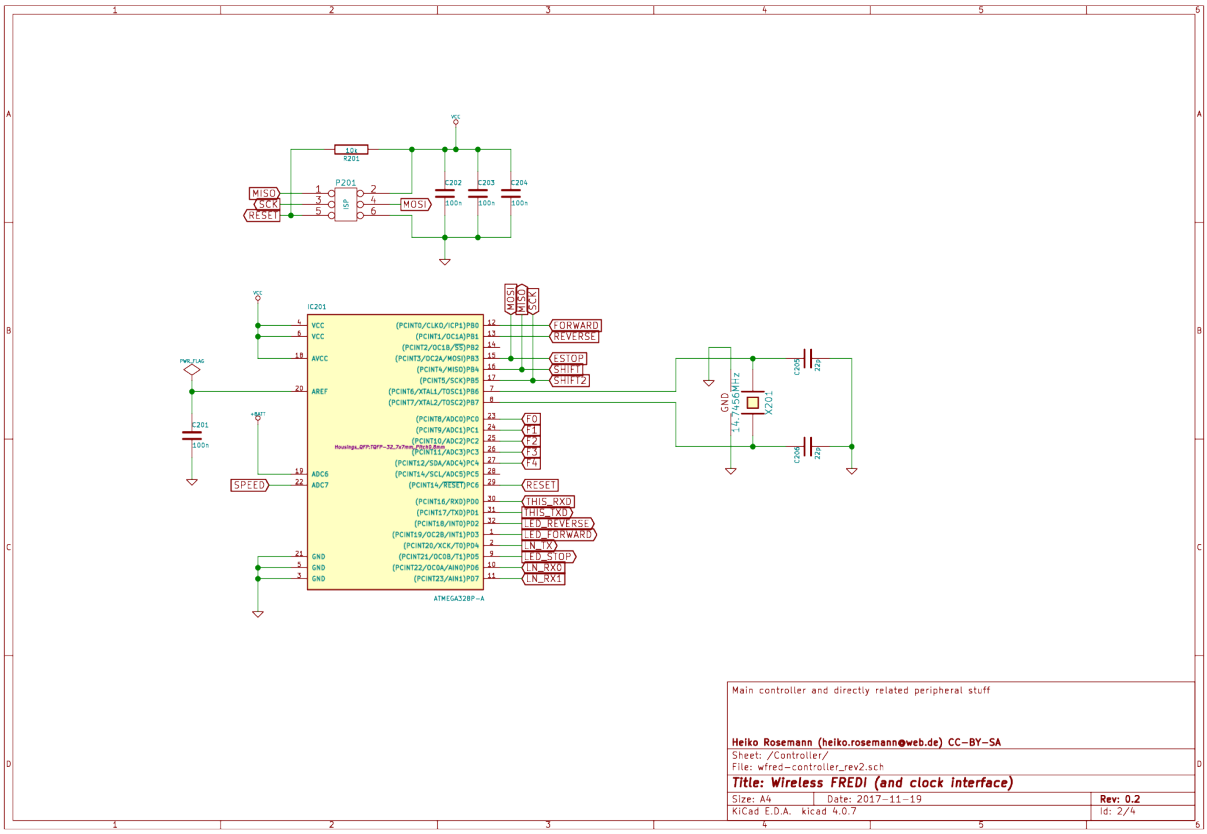


Figure 27: Schematic sheet including ATMega 328P along with crystal and in system programming header

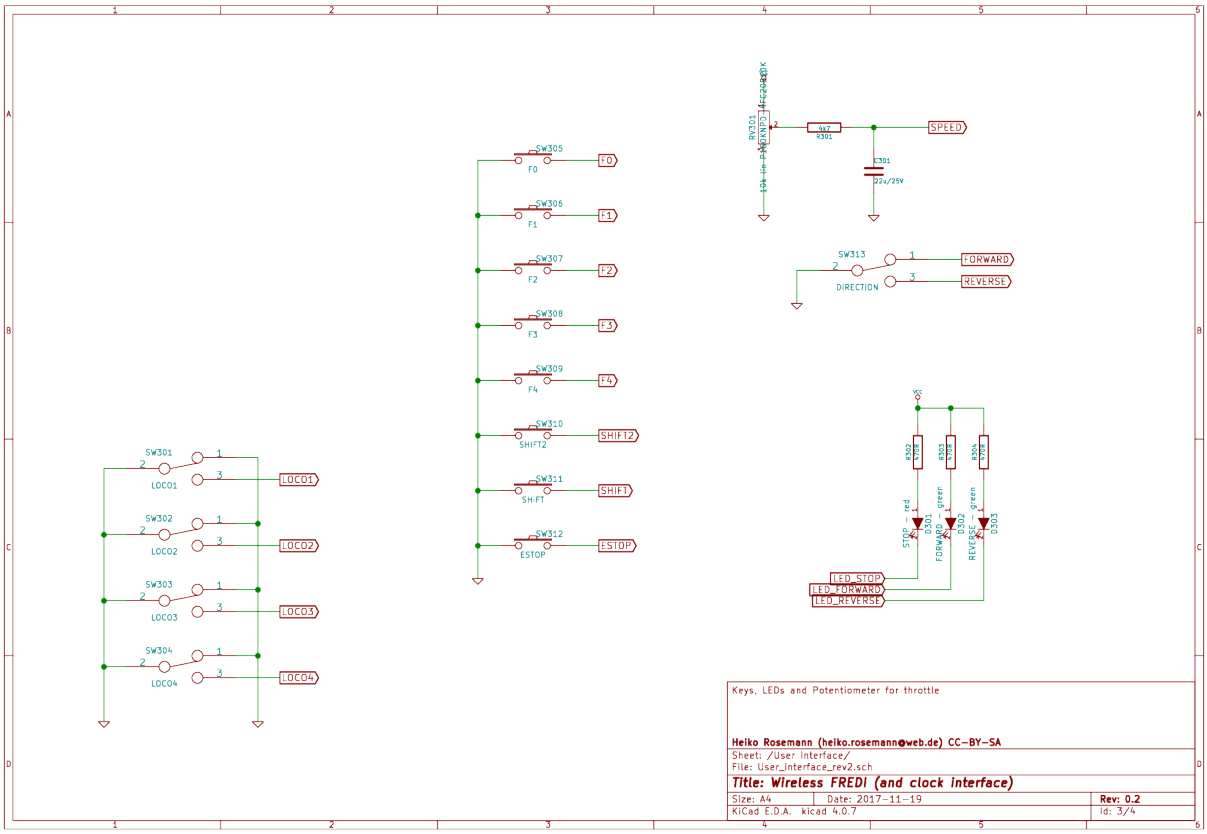


Figure 28: Schematic sheet including pushbutton switches, loco selection switches, direction switch and speed potentiometer

5.8 Hints for building the wiFred AA battery prototype

The prototype PCB has holes in the center of the pushbutton switch footprints and LED footprints to enable transferring their positions to a StrapuBox housing with a sharp needle or similar, and the position of the loco selection switches can also be transferred to the housing by marking it through the non-copper holes at their ends. Refer to Section 5.5 for pictures and more detail, the differences between the AA battery prototype and the LiPo battery revisions are minimal in this aspect.

The remaining assembly is a basic exercise in installing all the components to the PCB, listed in Table 7.

After assembling the PCB with all the components and drilling and cutting the holes and cutouts into the housing, there are few steps left. First, a few protrusions inside the housing need to be removed so the PCB fits properly. Figure 29 shows how they can be removed easily, remains may be cut off with a hobby knife. Second, new PCB mounting pads need to be installed as shown in Figure 30. For the prototype, Forex PVC foam was used, cut with a pair of scissors and glued to the housing with superglue, making sure not to be in the way of any components on the PCB, but any kind of easily worked upon material with a thickness of 3 mm can be used, as long as it will take self-driving screws (prototype uses 2.9 mm by 6.5 mm DIN 7981 screws). Third, the two shift keys need yellow paint on the top and the emergency stop key needs red paint - either any kind of paint or a paint marker like Edding 751 will do. Finally, both the ESP8266 and the ATMega 328P will need to be programmed as described in Section 5.6.

Designator	Package	Designation
B101	KEYSTONE1013	BATT_HOLDER
C206, C205	C_0805_HandSoldering	22p
C301, C105, C104, C102, C101, C402	C_0805_HandSoldering	22u
C401, C204, C203, C202, C201, C103	C_0805_HandSoldering	100n
D301	LED_D3.0mm	STOP - red
D302	LED_D3.0mm	FORWARD - green
D303	LED_D3.0mm	REVERSE - green
IC201	TQFP-32_7x7mm_Pitch0.8mm	ATMEGA328P-A
K401	Pin_Header_Straight_1x03_Pitch2.54mm	UART_ESP
K402	Pin_Header_Straight_1x03_Pitch2.54mm	UART_AVR
L101	L_2424_HandSoldering	22u
P201	Pin_Header_Straight_2x03_Pitch2.54mm_SMD	ISP
P401	Pin_Header_Straight_1x02_Pitch2.54mm	ESP_BOOTLOAD
R301	C_0805_HandSoldering	4k7
R304, R303, R302	C_0805_HandSoldering	470R
R401	C_0805_HandSoldering	100k
R402	C_0805_HandSoldering	47k
R405, R404, R403, R201	C_0805_HandSoldering	10k
RV301	P160KNPD	10k lin P160KNPD-4FC20B10K
SW301	OS102011MS2Q	LOCO1
SW302	OS102011MS2Q	LOCO2
SW303	OS102011MS2Q	LOCO3
SW304	OS102011MS2Q	LOCO4
SW305	KSC621G	F0
SW306	KSC621G	F1
SW307	KSC621G	F2
SW308	KSC621G	F3
SW309	KSC621G	F4
SW310	KSC621G	SHIFT2
SW311	KSC621G	SHIFT
SW312	KSC621G	ESTOP
SW313	100SP1T1B1M1QEH	DIRECTION
U101	TSSOP-8_4.4x3mm_Pitch0.65mm	L6920D
U401	ESP-12E_SMD	ESP-12E
X201	Crystal_SMD_TXC_7M-4pin_3.2x2.5mm_HandSoldering	14.7456MHz
	Housing StrapuBox 6090	
	Two Jumpers, 2.54mm	
	Potentiometer Knob, 21 mm	
	Three fastening screws, 2.9 mm dia x 6.5 mm	

Table 7: List of components for the AA battery prototype wiFred

Figure 29: Removing protrusions inside the housing so the PCB fits

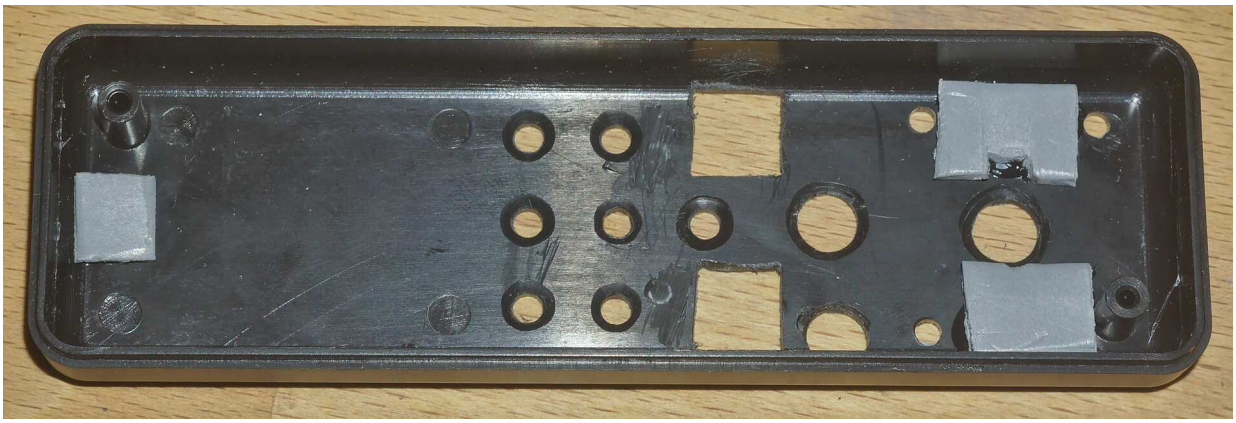


Figure 30: New PCB mounting pads made from 3 mm thick Forex PVC

6 Background for wiFred development

As of the writing of this document, [JMRI] has a long track record of offering a server for using smartphones as wireless model railroad throttles, along with apps like [wiThrottle][6] and [EngineDriver]. This server will enable WiFi throttles to control locos any model railroading layout to which JMRI can build a connection [Supported Hardware]. In addition, [Digitrax LNWI] and [MRC Prodigy WiFi] offer specific hardware solutions to enable the connection of the abovementioned smartphone apps to their DCC systems through a WiFi network.

[6] wiThrottle is also the name JMRI uses for the protocol and the server.

The [Fremo] is a European modular model railroading club whose unique requirements on its DCC throttles led to the creation of the throttles [FRED] and [FREDI] - a series of LocoNet-throttles which started their life as hobbyist projects with large numbers in circulation but were also commercially available from [Uhlenbrock].

6.1 Specification wishlist

In modular railroading events, particularly of the Fremo-americaN-group, multiple people have evaluated the smartphone throttle solutions and found them lacking a nice, haptical feedback. But the idea of wireless control without locking into a specific vendor and their necessarily expensive equipment found great approval. So a wishlist was compiled to define the requirements for a wireless throttle:

- Same form factor as the [FRED] with similar controls
- Option to control at least two, better four locomotives for double/triple traction (similar to the double FRED)
- Battery runtime of at least six hours
- Exchangeable batteries, so when the battery runs down, they can be quickly exchanged for a charged set or cheap primary cells
- Easy configuration, but not too easy to prevent operators from accidentally selecting other locomotives
- As little change to the existing Fremo Loconet network as possible
- Use of wiThrottle protocol, so the server side of the communication can be assumed to work and does not have to be developed as well

6.2 Development history

The first prototype versions of the wiFred were built to run from two AA cells, either dry batteries or rechargeable NiMH cells. As described in Section 5.7 this required some special adaptations of the housing to fit all components. Even then, experience with the prototypes showed the battery compartment cover did not really fit and easily broke when trying to open and close the battery compartment.

So the next versions were built around an integrated lithium battery, losing the ability to exchange empty batteries, but with increased runtime and proper fit into the housing. Recharging of the second generation is done through a Micro USB connector, so a powerbank can extend the runtime of the device when the internal battery is exhausted. Also, the loco selection switches act as more of a power switch than they did with the first prototypes, reducing power consumption to a negligible amount when all locos are deselected.

The latest prototype will switch from a double processor implementation to a single, new ESP32-S2 processor, saving space, reducing the number of components, placing all SMD components on one side of the PCB and enabling direct USB firmware download. It will not contain any new features over the earlier LiPo battery versions, but mainly be easier to build for the hobbyist.

6.3 Wireless clock

During the development of this wiFred another topic came up in the americaN group of the Fremo, namely wireless clocks with adjustable clock rate for Timetable & Trainorder operations. This led to the spinoff of the [wiClock] project.

7 References

7.1 References

[1] [Github repository for wiFred] newHeiko/wiFred: wiThrottle-compatible hardware controller, https://github.com/newHeiko/wiFred

[2] [JMRI] JMRI: A Java Model Railroad Interface, https://www.jmri.org

[3] [Supported Hardware] JMRI: Hardware Support, https://www.jmri.org/help/en/html/hardware/-index.shtml

[4] [wiThrottle] WiThrottle, https://www.withrottle.com/html/home.html

[5] [EngineDriver] Home | EngineDriver, https://enginedriver.mstevetodd.com/

[6] [Fremo] Home - FREMO - Freundeskreis Europäischer Modelleisenbahner e.V., https://www.fremo-net.eu/en/home/

[7] [FRED] FRED, http://fremodcc.sourceforge.net/diy/fred/fred_e.html

[8] [FREDI] FREDI, http://fremodcc.sourceforge.net/diy/fred2/fredi_d.html (German only)

[9] [Uhlenbrock] Uhlenbrock | FRED, der Handregler für die Intellibox, https://uhlenbrock.de/de_DE/-produkte/prodarch/I62AD172-001.htm!ArcEntryInfo=0004.41.I62AD172

[10] [Digitrax LNWI] LocoNet WiFi interface, https://www.digitrax.com/products/wireless/lnwi/

[11] [MRC Prodigy WiFi] Prodigy WiFi, https://www.modelrectifier.com/Prodigy-WiFi-s/332.htm

[12] [Steve Todd] JMRI RaspberryPi as Access Point | M Steve Todd, https://mstevetodd.com/rpi

[13] [wiClock] wiClock, a WiFi-JMRI-Clock, found at https://github.com/newHeiko/WiFi-JMRI-Clock or its documentation at https://newHeiko.github.io/WiFi-JMRI-Clock