**Instructions for building a WiFi Throttle with JMRI Withrottle Server**

Christoph Hauenstein – Dec.24.2023.

**Thank you Geoffrey Bunza for your “A Simplified WiFi Throttle You Can Customize” blog.**

**I did a total rewrite of this example code to get extra buttons and much more. See “Features and details” below.**

**The most outstanding feature of this throttle is that it is using the JMRI Roster for selecting locos and some of their functions being automatically assigned to dedicated switches.**

**The throttle can be configured as “Single-Throttle” or “Multi-Throttle” for controlling up to 6 trains independently.**

**What you will need if you want to build this throttle:**

**DIY Keypad with 16 buttons in 4 rows and 4 columns. I designed it using the Etching service from ExpressPCB. The included .rrb file can be used with their free software to order PCB’s.**

**The included .pdf files can be used as a base for creating your own designs if you wish.**

**OLED 128x64, like SSD1306 type, with I2C controller, 4-5V.**

**Rotary Encoder with Switch and a Knob.**

**ESP32 based development board.**

**1 Toggle Switch ON/OFF/ON for direction (center off for Drive Hold).**

**1 Toggle Switch ON/OFF for Bell.**

**4 Push Buttons for Whistle, Coupler, Brake, Release Loco.**

**16 Tactile buttons for the 4x4 keypad.**

**4-5V Battery.**

**Software included in package uploaded to the ESP32 board using the Arduino IDE.**

**Computer with WiFi connection running JMRI Panel Pro or Decoder Pro.**

**In my case I loaded the JMRI Panel Pro on a Raspberry Pi with Steve Todd’s Image file. On the web: look for “JMRI Raspberry Pi as Access Point” by Steve Todd.**

**And of course you need some kind of enclosure where you can put everything in.**

All files included in the zip file are meant to be suggestions only. The wiring diagrams are of course the base of the provided sketches (source code). I reviewed everything for correctness and tested multiple times. Nevertheless no warranty can be given that everything will work correctly.

Not for commercial use.

**My throttle looks like this:**

****

**Features:**

* **Can be configured as Single-Throttle or Multi-Throttle with up to 6 throttles in one.**
* **Network ID, Password, Host IP address, Port can be defined during startup.**
* **“Dedicated” switches for functions: Whistle-short, Whistle-long, Bell, Coupler-rear, Coupler-front, Shunting, Drivehold, Fade.**

**The JMRI WiThrottle Server Roster entries determine which function number is assigned to the dedicated Switches on the throttle.**

**A matching function name in the JMRI-Roster selects the function number that will be mapped to the switch.**

**So all you need to do is: go to the JMRI Roster and put the switch-names on the function number of the locomotive. If a function is not available or not needed just don’t define it in the Roster.**

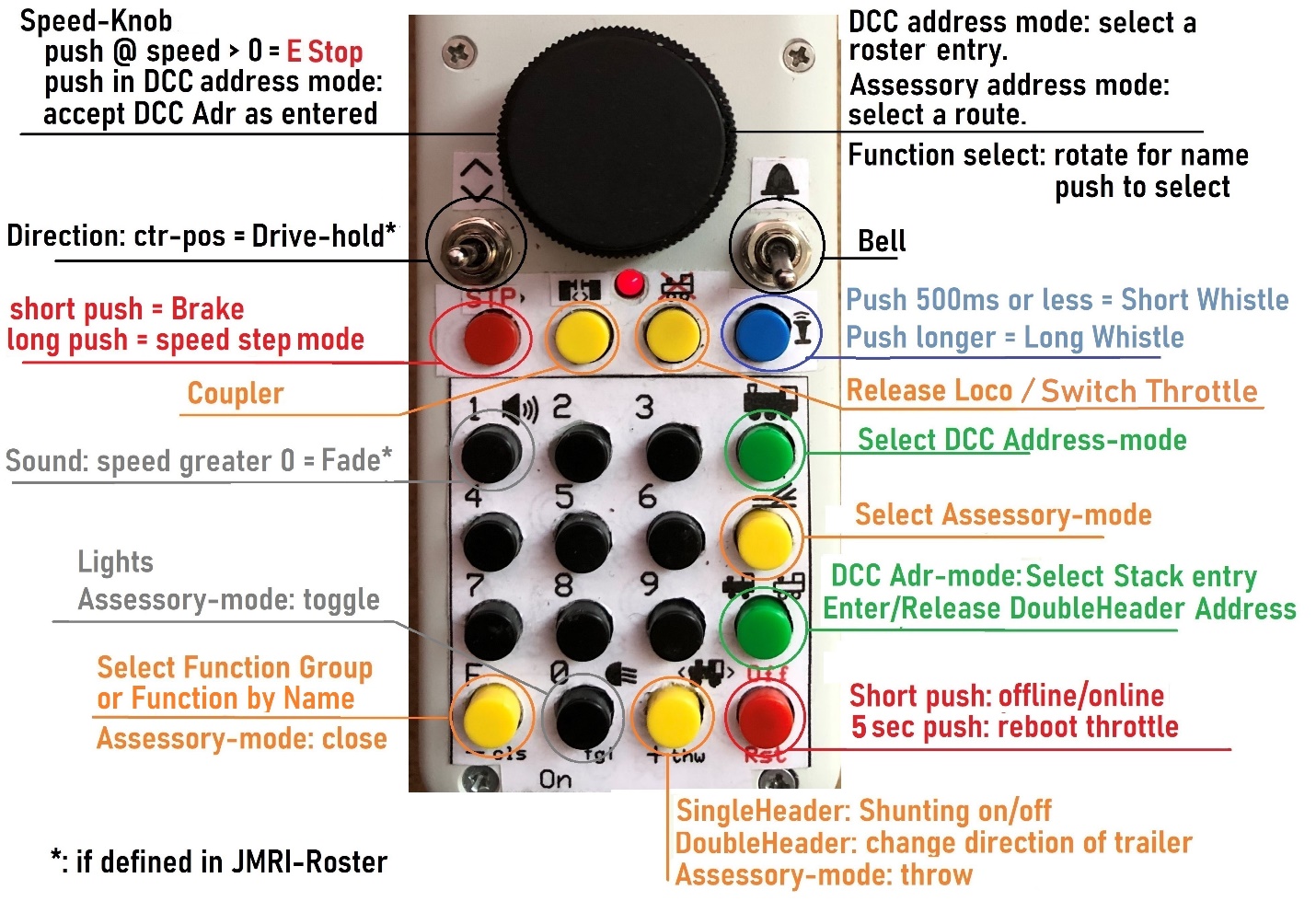
**You can also use your own switch names by changing them in the throttle program.**

* **Select a Locomotive from the roster in JMRI. Step through the roster by turning the**

**Speed Knob. Up to 100 Locos in your roster can be selected. The limit of 100 can be extended if needed.**

* **Select a Function by name using the Speed Knob.**
* **Enter a DCC address via 3x4 Numeric-Keypad.**
* **Select turnouts (stationary decoders) defined in JMRI.**
* **Select routes defined in JMRI.**
* **Display the Fast Clock** from JMRI.
* Display turns dark when not in use for 1 minute.
* Utilizing heartbeat control in JMRI.
* Support for 28 functions as supported by JMRI.
* The last 6 DCC addresses are kept in a stack beyond power off for quick access.
* Select Speed Step Mode 28 or 128.
* Monitor the battery charge.

**Functions as they are assigned to switches/buttons:**

****

**I tested my throttle:**

1) Throttle connected to JMRI on a Win10 PC running JMRI versions 4.18 and 4.20.

2) Throttle connected to a Raspberry Pi 4 Access Point running JMRI using the Image file from

Steve Todd: mstevetodd.com/rpi.

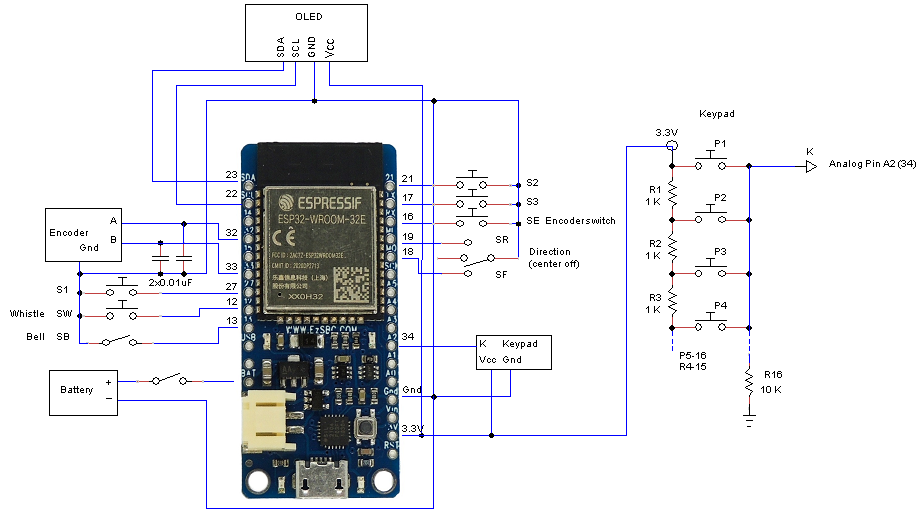
In both cases JMRI was connected via a Lenz USB (23151) adapter to Lenz LZV100 DCC controller.

Since the throttle connects to JMRI only, and doesn’t “know” what DCC system is behind JMRI it should work with any DCC syste

**My Parts list:**

|  |  |  |  |
| --- | --- | --- | --- |
| Part name | Qty | Description/Use | Source/Number |
| Enclosure | 1 | Hammond 1552C5GY | DigiKey 164-1552C5GY-ND |
| ESP32 board | 1 | ESP32\_Feather | EzSBC.com / ESP32\_Feather |
| LiPo Cell | 1 | 400mAh | EzSBC.com / Cell10-1 |
| Rotary Encoder | 1 | Bourns PEC11L | DigiKey PEC11L-4220F-S0015-ND |
| OLED 128x64 I2C | 1 |  | Adafruit, Mouser, Ebay …. |
| Toggle Switch SPST | 1 | Sub-Mini/Micro On/On | eBay |
| Toggle Switch SPDT | 1 | Sub-Mini/Micro On/Off/On | eBay |
| Tactile buttons & caps | 20 | Gikfun EK1852 | Gikfun.com #EK1852 |
| Tactile buttons | 16 | 12mm high | AllElectronics #TSW612 |
| Slide Switch | 1 | Power switch | Pololu #1408 |
| LED 3mm | 1 | Power On indicator |  |
| Knob for encoder | 1 | Cut pointer off & file smooth | AllElectronics KNB-282B |
| Wire in various colors |  | TCS Decoder wire 30-32AWG | TonysTrains.com |
| Round pin connectors |  | Mating SIP socket | eBay |
| SIP Sockets |  | To connect enclosure halves | eBay |
| Small screws & nuts |  | Flat Philips head M2 or 3/32 | FastenerExpress.com |
| JST connector | 1 | Battery to Board, 2mm pitch | Newark 38K8066 |
| Keypad make yourself | 1 | PCB designed and ordered at | ExpressPCB.com |
| Resistors 1/8W 1k | 16 | Keypad |  |
| Resistor 1/8W 10k | 1 | Keypad |  |

**Schematic:**

****

The heart of my throttle is the **ESP32 processor board**. I ordered it from EzSBC, following G. Bunza’s suggestion in his Blog on MRH). They also have the matching LiPo Cells which can be charged with the onboard charger. The board is made in the US so delivery is short and pricing is good.

To be able to access the USB connector on the board it needs to be mounted with that end looking out.

I used a Rotary Encoder for speed and loco selection. It has advantages over a potentiometer for stepping through the roster. The disadvantage is: it has no stopping points to know without looking when speed zero is reached. Therefore I included a button for issuing ‘Brake’ by setting the speed step to zero.

A suitable knob for the encoder without a pointer was hard to find. I ended up using one with a pointer

which I cut off and smoothed down with the help of a lathe.

**Keypad:**

To fit the keypad into the enclosure I designed one using a PCB service and their CAD software: ExpressPCB. When you order a prototype you get 3 boards. I got mine for around 60 dollars (shipping included), which is 20 bucks per throttle (if you build 3…. ).

In order to get everything into the enclosure I put the LiPo cell under the keypad. Therefore the keypad sits right under the inside top surface. But the rest like toggle switches and encoder reach deeper. So to accommodate that you could either wire these or like I did, make a second PCB for the encoder and switches and set it lower than the keypad.

**Encoder:**

It needs two 0.01uF or 0.1uF capacitors on A and B terminals to ground to suppress noise from bouncing switches.

**OLED:**

I had an SH1106 type handy so the Wire.lib I included calls for that. If you have an SSD1306 the code provides for that by commenting in and out the lines

**#include "SH1106Wire.h"**

**//#inlcude "SSD1306Wire.h"**

Make sure your OLED has an I2C controller.

If you use one with a different controller (driver chip) then I’m afraid you need a different library and perhaps different display commands than the ones in my sketch.

**Wiring:**

Keypad, Switches, OLED are mounted to the top half of the enclosure. The ESP32 board is mounted to the bottom half. To make the connection between both I mounted a SIP socket strip to the top and a mating male pin stripe to the bottom. See pictures at the bottom of this doc. When putting both halves of the enclosure together there are no wires that could get squeezed. I soldered the wires directly to the boards because there is no room for headers. Make sure the wires are on the pins as shown in the diagram and sketch, especially from Encoder A to ESP32 since it is used with interrupt.

**Program:**

Install Library:

**https://dl.espressif.com/dl/package\_esp32\_index.json**

which is from G.Bunza’s Blog. As mentioned before I used his example to get started with building this

throttle. His Blog is here: https://model-railroad-hobbyist.com/node/35652.

The library can be installed using the File -> Preferences in the IDE. This is from G.Bunza’s blog:

“In the field after “Additional Boards Manager URLs:” paste:

**https://dl.espressif.com/dl/package\_esp32\_index.json**  and click OK.

Go back to the top menu bar and click: Tools-> Board -> Boards Manager…

Look towards the bottom in the new Boards Manager list that opens, and find the entry labeled

“esp32 by Expessif Systems” and click on that entry. Click “Install.”

Now click on Tools-> Board-> **Esp32 Dev Module** from the board list (likely near the bottom).”

There is another library needed for the JMRI clock: TimeLib.h. I downloaded it from here:

**https://github.com/geekfactory/TimeLib**

I downloaded the zip file and installed it by going to: Sketch - Include Library - Add .ZIP Library.

This does not have to be installed if you don’t use “#define CLOCK”.

**Once the throttle is assembled:**

First the keypad needs to be configured. Download **sketch\_Keypad\_Config-V1** and upload it to the ESP32 board using the Arduino IDE. Start the serial monitor as soon as the upload is finished.

The program starts reading and displaying in SM the analog input of the keypad 10 times. With NO BUTTON PRESSED all readings should be zero.

Now press the buttons 1 thru 16, one button at a time for a couple of seconds each. Min & max values will appear in the SM once a button is released. These are the readings of a particular button. Now in the Serial Monitor enter the Button Number you pressed and click Enter. This will save the values for the particular button.

The keypad button numbers as I used them in the program are:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 13 |
| 4 | 5 | 6 | 14 |
| 7 | 8 | 9 | 15 |
| 10 | 11 | 12 | 16 |

**Note:** the keypad can be reconfigured at any time after the initial configuration as described above, but without loading the **sketch\_Keypad\_Config**, as follows:

While the throttle is ON, start the serial monitor, enter the letter C. This will start the keypad configuration doing the same steps as above.

After entering the values for all buttons the keypad-config can be ended by entering 99 in SM, or by pressing the “Brake/Stop” (S3) button.

Once the keypad is configured: from the provided library download the

**sketch\_WiFi\_Multi\_Throttle\_V …** and open it using the Arduino IDE.

**Options to check before compiling/uploading the program:**

**For the wireless connection to a JMRI WiThrottle server the throttle has a “Default”, also called “Home” setting and a “Club” setting which can be manually configured and/or selected during Setup after Power on. For the Club configuration please refer to the document MCab\_Config23xxxx.docx.**

**For the Home configuration continue reading:**

The default WiFi details provided in the **sketch\_WiFi\_Throttle\_V…** match Steve Todds Raspberry Pi image file. The default WiFi details may have to be matched up with your environment:

// WiFi Server Definitions

char\* dftssid = "RPi-JMRI";     // default local wireless network id

char\* dftpass = "rpI-jmri";     // default key

char\* dfthost = "192.168.6.1";  // default WiThrottle server IP Adr.

int dftport = 12090;

**// compile options:**

**//#define LOG**

**#define LOCKED    // Bell is defined as 'locked' in JMRI-Roster**

**#define BATLEVEL  // Battery level sensor**

**#define CLOCK     // Display JMRI clock**

float maxvolt = 4.2; // battery voltage when fully charged. Used with BATLEVEL defined.

LOG is adding statements to enable some logging in Serial Monitor which may be helpful for debugging. If enabled it still needs to be activated: in order to activate start the Serial Monitor and enter a D (or d). Logging stays active, even after power was off, until entering a D again.

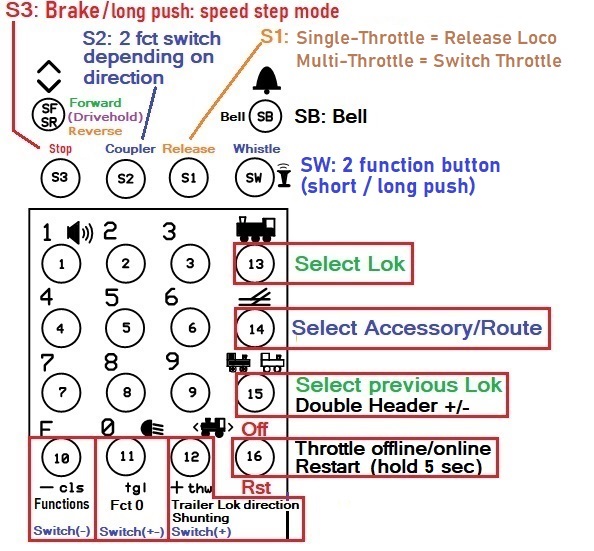
LOCKED is for the *Bell* switch. The Withrottle design works with buttons. The *Bell* usually is defined in the Roster as locked (lock = checked), and is therefore turned on by pressing a button and turned off by pressing the button again. Because I use a switch for the *Bell* the code simulates a button push each time the switch changes its position. However, the *Bell* could also be defined generally as momentary in JMRI in which case the coded simulation for button would be “in the way”. For this scenario the “LOCKED” can be removed (put // in front of #define, as in //#define), instead of painstakingly changing all JMRI-Roster entries. The switch will then match a momentary function.

In short terms:

Switch for Bell, Bell function is locked > #define LOCKED

Switch for Bell, Bell function is momentary > //#define LOCKED (commented out or line removed)

Button for Bell, Bell function is locked > //#define LOCKED (commented out or line removed)#define BATLEVEL & CLOCK can be commented out in the same way if any of these functions are not wanted.

**The image below shows the button numbers and their functions:**

**Extra Switches:**

The table below shows how I assigned the extra switches to functions. Any function that is not available in a Loco definition in JMRI will just be ignored.

All functions that are defined in JMRI and are matching one of the entries in the table will be mapped to the corresponding switch.

// JMRI-Roster mapping for common function switches >>

**char \*fctname[] = {"Whistle-short", "Whistle-long", "Bell", "s1map", "Coupler", "Coupler-front", "Shunting", "Drive-Hold", “Fade”};**

**#define numswfcs 9** // nbr of switched functions = nbr of array elements

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Element | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Switch | SW-short | SW-long | SB | S1 | S2-forward | S2-reverse | Button 12 | Direction ctr pos | Button 1(8) + Drive |
| Function | Whistle-short | Whistle-long | Bell |  | Coupler-rear | Coupler-front | Shunting | Drive-Hold | Fade |

|  |  |
| --- | --- |
| Element | 9 |
| Switch | Button 1(8) |
| Function | Sound |

Switches S1 & S3 are not mapped in my sketch.

In Single-Throttle mode S1 could be mapped - by entering a function name in element 3.

If not mapped it is used for ‘Release Locomotive’.

In Multi-Throttle mode it cannot be mapped as it is used for switching throttles.

S3 = Brake (Stop).

Element 0 is selected by a short push on Switch (Button) SW. The mapped function is “Short Whistle”.

Element 1 is selected by a long push on Switch (Button) SW. The mapped function is “Long Whistle”.

Element 2 is selected by turning toggle switch SB on. The mapped function is “Bell”.

Element 3 is selected by Switch (Button) S1. The default function is “Release Loco” (if not mapped).

Element 4 is selected by Button S2 in forward.

Element 5 is selected by Button S2 in reverse.

Element 6 is selected by Keypad Button 12. The mapped function is “Shunting”.

Element 7 is selected by the Center position of the Direction Switch. The mapped function is “Drive-Hold”.

Element 8 is selected by Button 1 while in Drive-mode. Mapped function is Sound Fade out/in.

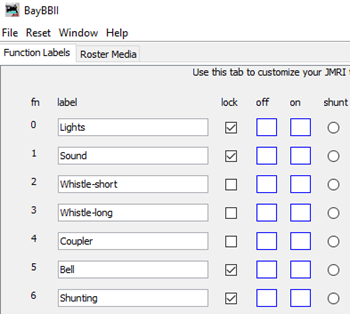
Element 9 is selected by the button for which Sound is specified in the Roster.

It is important that the function names in the sketch match the function names in the JMRI Roster.

If a function name (like i.e. “Coupler-front”, or “Drive-Hold”) has no match in the JMRI Roster the function switch has no effect except for the “Coupler-front”, see note in **bold** below. The names are case sensitive.

The sketch selects the function number as assigned in the Roster. It can be different for each Locomotive.

**Roster configuration example in JMRI: (note that “Coupler-front” is not defined for this loco, in which case the “Coupler” (rear) is addressed by element 4 instead; if there is no match for element 5 then element 4 is used in reverse as well as in forward.**



**Coupling** is a function in most European Mobile Decoders. I like to use it by programming the

push/move action also known as “coupling procedure” or “Kupplungswalzer” on one function #.

It works with Digital (aka. Electric, Remote) couplers. Or with a permanent magnet between or under the rails.

**Drive-Hold** is a function in specific LokSound decoders by ESU (and may be others, I’m not sure). If this function is defined for a Loco in the Roster the center position of the direction switch selects that function. When this function is activated the speed that is “on hold” is also displayed next to the speed setting display. Moving the direction switch away from the center position turns “Drive Hold” off.

**When you use this function make sure it is set as “Momentary” (Lock = unchecked) in the Roster.**

**Fade (Sound-Fading)** is used for instance when the train goes into a tunnel or disappears elsewhere to enhance the effect for distance. Which button is used for Fade depends on the definition in the JMRI-Roster for Sound. It can be either one of the two: F1 or F8. The Sound button controls Sound at speed 0, and Fade at any speed greater than 0.

**After reviewing the above and applying any necessary changes the sketch can now be uploaded.**

**Operation:**

The throttle comes configured as Single Throttle. It can be changed to a Multi-Throttle during start up:

**Turn the throttle on.**

**Switch number of throttles:**

When **“Press Stop for Setup”** appears on the display press **Stop** (Brake – S3 button) within 5 seconds.

If you don’t press Stop within 5 seconds **startup** continuous normally.

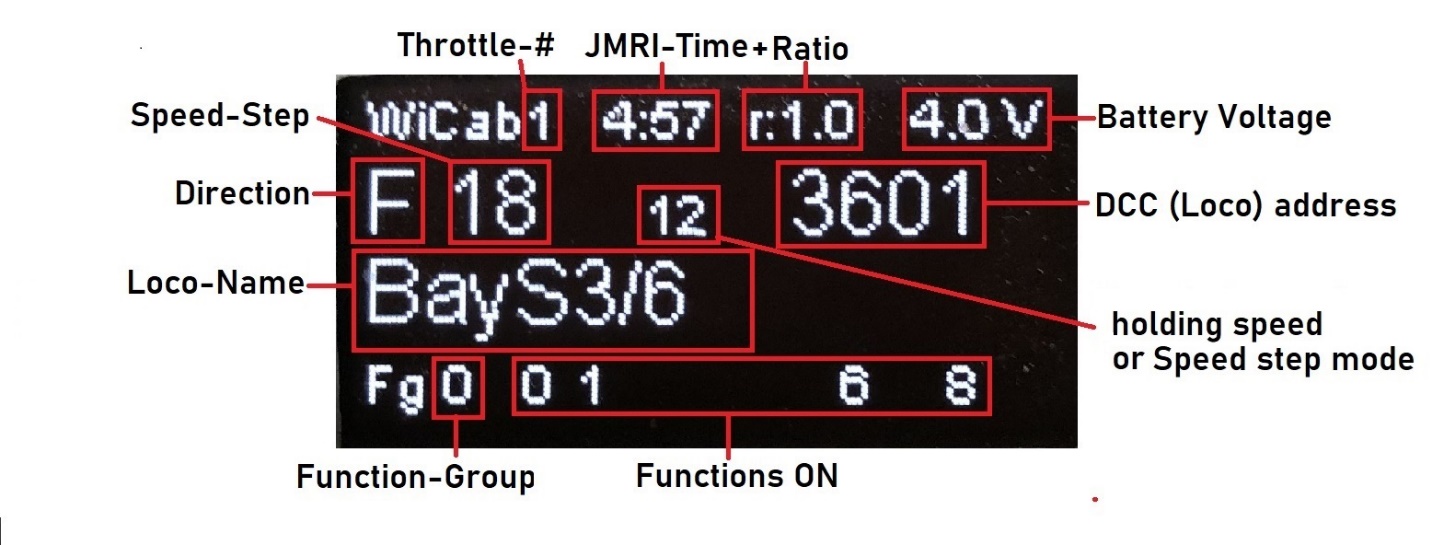
If Stop was pressed the current throttle setting is displayed. Turn Speed Knob to select a new number of throttles - up to 6. Press down on the Knob. This will change the setting to the currently displayed number of throttles and continue with normal startup.

**To switch throttles during normal work press the S1 button.**

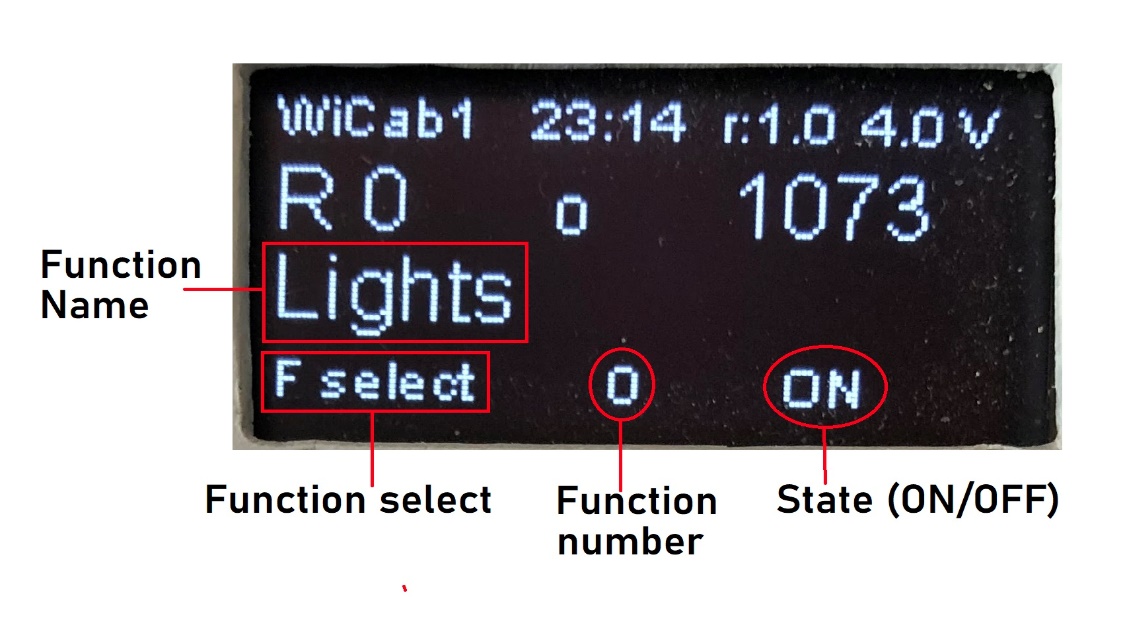
**Normal Startup:**

The throttle code tries to connect to the Local Wireless Network. If the network is not responding the code retries for 1 minute. If a connection is established within 1 minute the code tries to connect to the server. If the server is not responding the code retries for 1 minute. If the connection to the server is not established, or lost during operation, the code tries to reconnect for 2 minutes. After that it gives up and shows/stays offline.

**Loco Display:**

****

**Function select Display:**



**Notes:**

1. during the selection of a locomotive address by either going through the roster or going through the stack, the number of the roster/stack element is displayed in place of the drive-hold speed.
2. During the process of adding functions the battery charge display had been changed to showing the charge in percent.

**Select a loco:**

After power on the last used address is displayed with a “?” which means it is not activated yet.

In Multi-Throttle-Mode each throttle will display its own last used loco address.

If you want to use it press the  button (13). If not, select another address in 1 of 3 ways:

1. **Using the Roster:**

Step through the roster by rotating the speed knob until the loco you want is displayed, press the button (13), or the Speed Knob.

1. **Using the Stack:**

Press  button (15). By repeatedly pressing it steps through the throttle stack for the last 6 used addresses. Select one by pressing the  button (13), or the Speed Knob.

1. **Using the Number Keypad:**

Enter a new address. Press the button (13) ), or the Speed Knob.

To select a new address at any time press the  button (13) and proceed with one of the steps above.

**Release a Loco:**

the S1 button releases the current loco while using the device as single throttle.

Pressing button 13 in order to select a new loco also releases the current loco, in Single and Multi- throttle modes.

When the throttle is put offline by pressing the Off button (button 16) all locos that are currently active in all throttles are released.

**Recommendation: before turning off power press the Off button** in order to release all locos.

**Enter Double Header:**

Select the trailing loco first by following the steps above. Press  button (15). A plus sign appears in front of the address. Then press the  button (13) to select the Leading loco.

To change the direction of only the trailing loco press the button (12).

**Accessory decoder (aka Stationary decoder):**

Press the  button (14). Then either:

1. Use the Number Keypad to enter an accessory decoder address, or turn the speed knob for displaying turnout names. Press the Button (14) again to select the displayed turnout. Press the F button (10) for Close (-), Zero button (11) for Toggle (+-), or button (12) for Throw (+). Press button 14 to end Accessory mode.
2. If Routes are defined in JMRI: after pressing button 14 the first time in step (a) press it again. Turn the speed knob until you see the route you wish to set.

Press button 14 to set the route and end Accessory address mode.

**Select Function by Name:**

Press the F-key repeatedly until **“F select”** is displayed. Rotate Speed Knob until the desired function is displayed. Push down the Speed Knob to change the function state, or press the

F-key to cancel.

**Speed & Speed Step Mode:**

The throttle detects the speed step mode in the selected Loco automatically.

To change the speed step mode: at speed 0, press and hold the stop button until the mode changes.

Supported modes: 28 & 128.

The **Brake-button** sets the speed to zero (decoders should interpret this as Brake).

Changing the Direction while in motion also issues **Brake**.

**Drive-Hold:**

If this function is defined for a Loco in the Roster the center position of the direction switch selects that function. When this function is activated the speed that is “on hold” is also displayed next to the speed setting display. Moving the direction switch to the current direction of the Loco turns “Drive Hold” off.

**When you use this function make sure it is set as “Momentary” (Lock = unchecked) in the Roster.**

**Fade (Sound-Fading):**

Some decoders provide a function for “fading” = turn volume down to simulate tunnel or distance.

Normally Function 8 (US/AU) or Function 1 (EU) turns the sound on/off. If Fade is defined in the roster for a locomotive and the loco is driving - the Sound button accesses the function for Fade (if available) instead of F8/F1. When the loco is stopped and Fading is still on, the function 8/1 key will first turn fading off. Then another push will switch the sound again. Which of the two F-keys will control the Sound must be defined in the JMRI-Roster.

**Make sure this function - and Fade if available - are set as “Locked” in the Roster.**

**Functions:**

In normal mode with address mode off the numbers keypad can be used to turn on/off functions F0-28.

"F" = "Functions" button: Press/release F button to toggle through function groups 0 to 2 and “select”.

Functions 0 to 9 can be accessed in Fg0, F10-F19 in Fg1, and F20-F28 in Fg2. In “select” the Knob is used to step through the function-names available for the selected loco. Push the speed knob to change the current state of the particular function.

The current states of all functions (as well as the direction for a Loco) are retrieved from the server.

Function Names and Modes “momentary” or “continuous”, can be configured in JMRI by going to “Actions” – “Labels and Media” tabs.

Button (12) is also mapped to a function (“Shunting”). It too can be mapped to anything you want. It can be used while operating a loco like any other Function.

**Additional Instructions:**

**Lipo batteries** can easily be damaged. Do not squeeze or press into a tight space. Keep pointy and sharp objects away. Follow safety instructions for Lithium batteries.

The battery is charged whenever the throttle is connected to a USB port or Power-Supply and is turned on.

**Press/release the "Off/Rst" button** sends **Quit** to the server. To resume operations press it again. Pressing this button for 3+ seconds restarts (reboots) the throttle.

**The display turns dark** if there was no activity for a configured amount of seconds, default is 1 min. It can be changed in the code here: int cd = 60; // timeout for the screen saver (in seconds)

Pressing any button or turning the knob turns the display back on.

**The fast clock on JMRI** is displayed as hh:mm in 24 hour format along with the ratio on the throttle display.

JMRI is providing the clock in 1 minute intervals as a UNIX time stamp. To set the time go to the JMRI Server: Tools – Clock.

**The Throttle uses Heartbeat** if it is enabled in the server and is set to an interval of above 5.

This is from the JMRI Withrottle manual:

You can see messages sent between the throttle and JMRI by enabling DEBUG in JMRI: Add the line "**log4j.category.jmri.jmrit.withrottle=DEBUG**" to your default.lcf file and restart JMRI.

All messages (both directions) will be added to your JMRI System Console and the session.log file.

Some **DEBUG** info can also be sent by the program to the **Serial Monitor**: connect the throttle via USB with your computer and start the Serial Monitor, enter D. This will set the program into DEBUG mode.

You will see lots of messages being displayed in the SM about what’s being sent and received to and from JMRI and other info.

Entering D while in DEBUG mode ends DEBUG mode.

DEBUG mode stays on until it is ended, even past power off/on.

A letter “d” appears after the throttle name, like “WiCab1d”, to indicate when the throttle is in DEBUG mode.

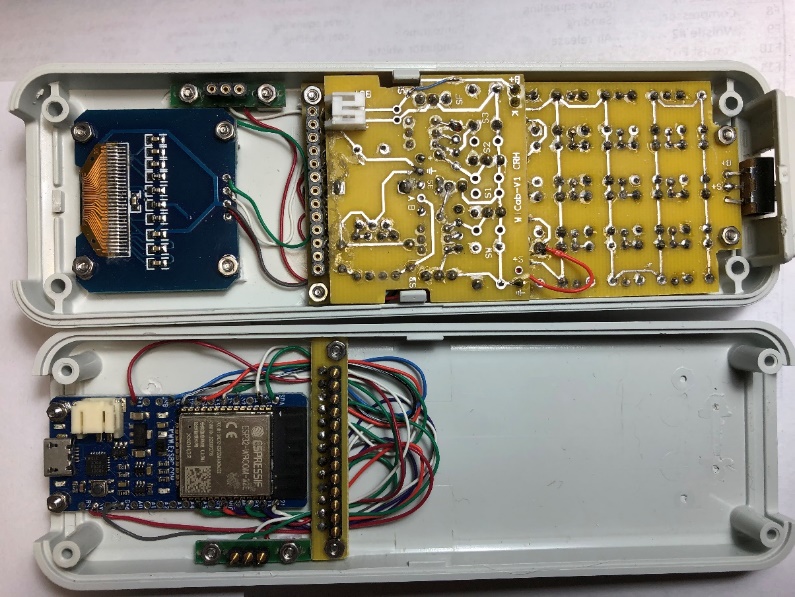
**The pictures below are from a test version. These slightly defer from the provided files for building the PCB.**

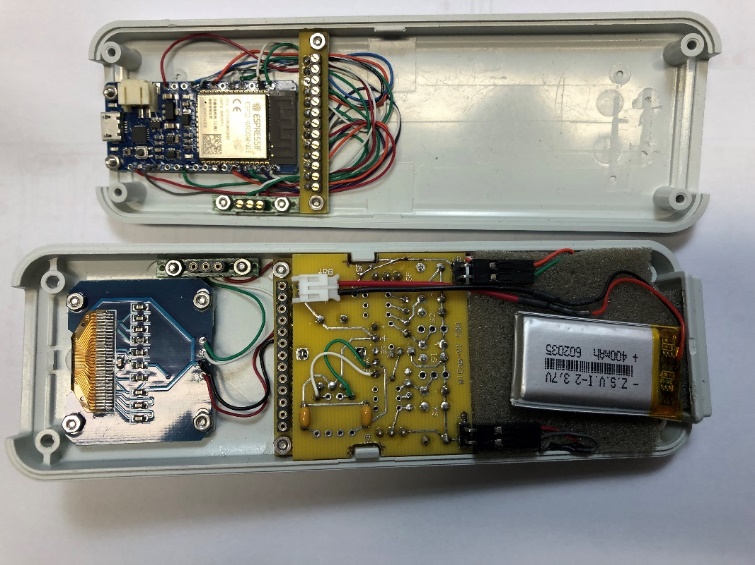
**Enclosure preparation**

The standoffs for mounting a board and clamping a cable have been milled down to make room for the internal parts. First I used a flush cutting tool, then I used a small router bit in my drill press with an x/y table to remove the rest. The electrical connection between the ESP32 board on the bottom and the

Buttons and switches as well as the OLED on the top is provided by a pair of SIP connector rows. No wires between the bottom and the top parts!

I created a drilling template with the same software that I used to develop the PCB’s. See pdf files included in the package.

****

****

**Battery sits on a sheet of foam. There is a thin sheet of styrene between foam and key pad to prevent the pointy soldering joints poking through the foam.**

****

**Access hole for USB connection**

****

**Power switch**

****

**The ESP32 can get warm during battery charging so I put holes for cooling in the bottom of the enclosure where the board is.**

**These appear to be enough to keep it from overheating.**