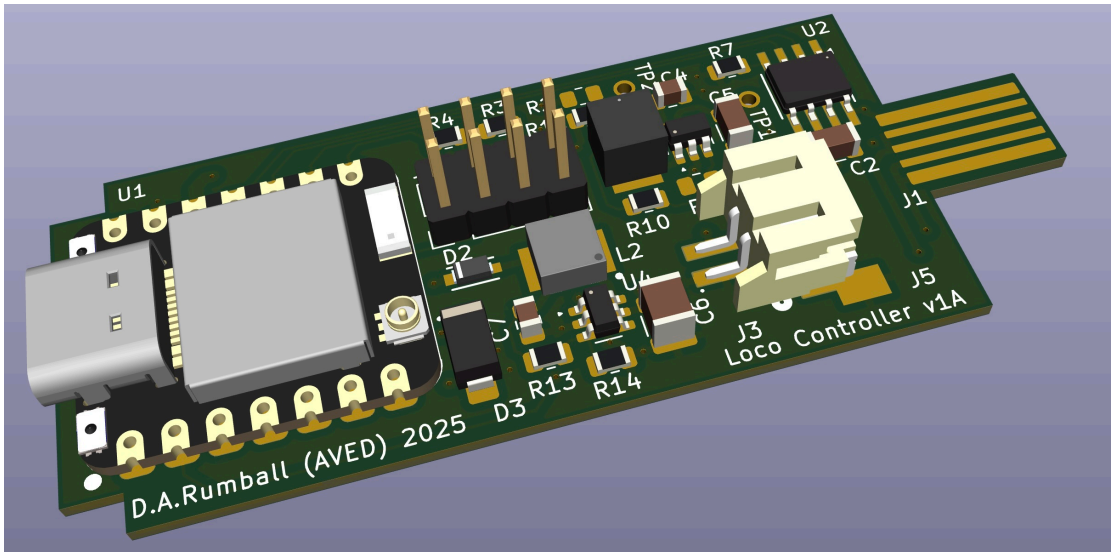
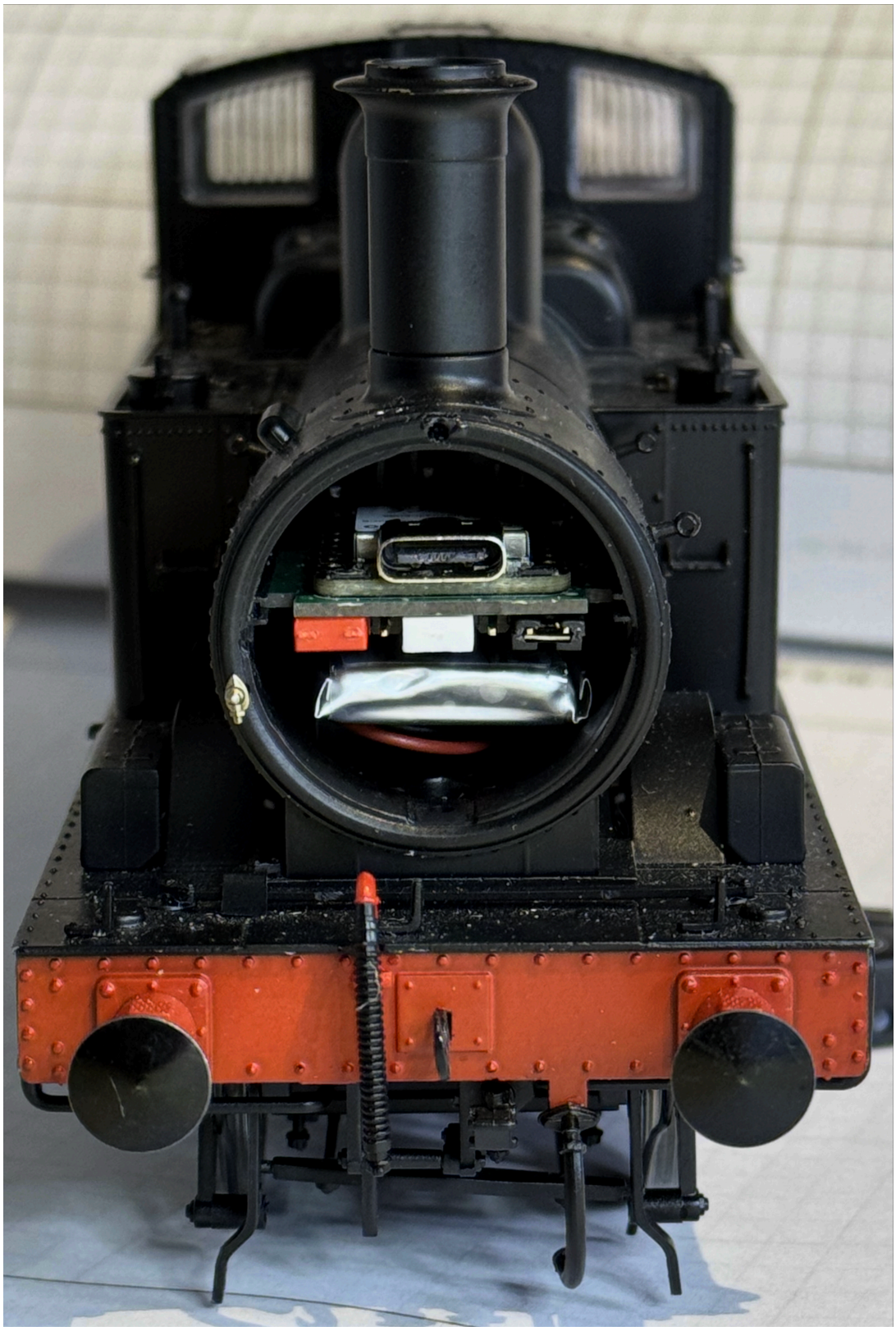
Loco Controller v1a User Notes





CONTENTS

Acknowledgements

I'd like to acknowledge and say thank you to all those in the Falmouth Society of Railway Modellers and Cornwall O Gauge Group for their encouragement and advice. Also thanks to the testers, Paul.R, Ian.W, Chris.H for their invaluable feedback to Richard.B for his help proof reading the documentation and and lastly thanks to Pocket Railway Museum for permission to work with their mobile app.

D.A.Rumball - Falmouth, UK - Oct 2025

"Every once in a while, a new technology, an old problem, and a big idea turn into an innovation" – Steve Jobs.

“Look Ma, no wires! - Anonymous”

Section 1 Introduction

The Loco Controller is designed to offer a simple upgrade of Dapol and other locomotives to battery power and wireless control. It is highly integrated, incorporating an ESP32-C6 dual core MCU (microcontroller Unit) module together with a motor driver, voltage booster, battery charger and input power conditioner all on a single double sided PCB.

The controller is designed as an exact replacement for the factory fitted DCC decoder adaptor installed in newer Dapol locomotives and installation consists of sliding out the factory PCB and replacing it with the controller. On the 58XX and B4 models the battery also slides in underneath the controller and is connected directly to the controller PCB.

For non Dapol locomotives there are a range of signals and power sources available on 2.54mm spacing pin headers including motor drive and LED outputs. In these cases though some disassembly of the locomotive body and soldering will be required.

Although designed primarily as a controller for motorised devices the device can also function as a generic wireless interface for signals, turnouts and other accessories. In such a ‘fixed’ application the unit can operate without a connected battery being powered solely by externally connected power sources and can drive multiple LED or servo outputs.

The controller implements a Wi-Fi access point and Web server that serves a page of locomotive specific controls to any standard web browser on a phone or tablet

The initial hardware and firmware release supports:-

- Connector compatible with Dapol 0 Gauge 58XX and B4 locomotives
- Aux connector for other models with power, motor, LED and servo drive
- Integral voltage boost from VBAT to 12 V for the motor at 1 A
- Integral battery charger with power in via track or USB and 2mm JST socket for a 1S Li-Po battery
- Track power input is compatible with DCC and can be up to +-50 VDC with protection against inductive spikes
- Battery voltage and USB voltage can be monitored in firmware
- Links provided to isolate track power and the battery
- Charging is indicated by firebox LEDs in Dapol models and also in UI
- Multiple levels of over-current/over-voltage/over-temperature protection
- Integral Wi-Fi access point and Web server
- Control by standard Web browsers (up to four simultaneous connections)
- Firmware is updatable via USB connection

Changelist

2025-10 - Initial release of hardware (1a) and distribution (v0.90).

Known Issues

None

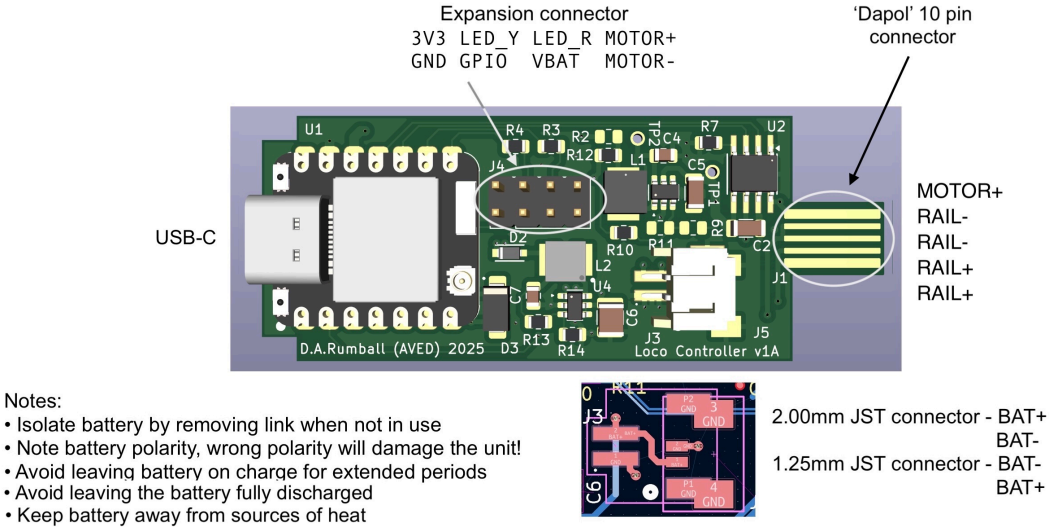
Todo List

None

Section 2 - Hardware Notes

Connections

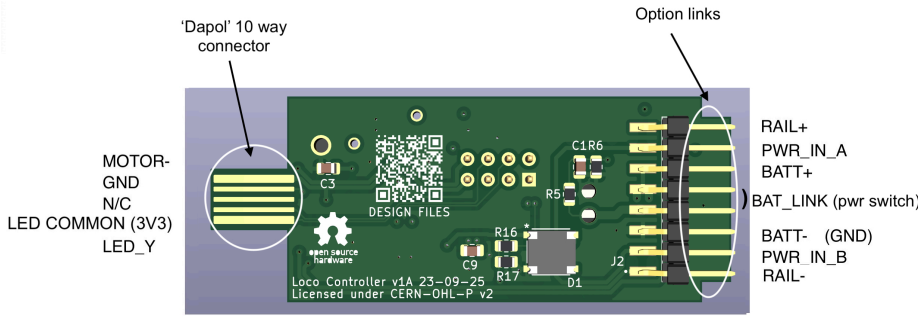
The PCB has five main connectors, a USB-C socket that can be used for charging and loading firmware, the Li-Po battery connector, an ‘edge’ style connector that plugs into a Dapol locomotive’s 10 pin internal socket, a set of 2.54mm header pins under the front of the PCB with options for powering the device and a 2.54mm header auxiliary connector that provides access to internal power, motor, LED and servo drive in models that don’t use the Dapol connector.



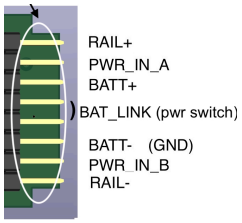
The battery is connected via a standard ‘JST’ style 2mm connector.

> CAUTION – Equipment damage. The PCB has no reverse-polarity protection. Confirm battery polarity before you connect the battery.

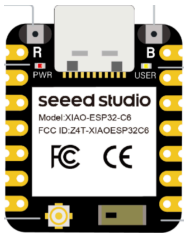
The battery negative is connected to GND on the PCB and the connection diagram above shows the polarity of the battery connector.



The links at the front of the PCB allow for the isolation of the battery which can be used as a power switch (or alternatively used for connecting an external switch) and provide for the isolation of external track power by removing the two outer links. When these links are removed the battery can still be charged via the USB connector and external power can be supplied through the front header pins.



Status LEDs



There are two LEDs on the ESP32 module which are visible from the front if the smokebox door is open. Viewed from the front on the left hand side is a yellow flashing LED that indicates normal operation of the firmware and on the right hand side is a red LED that indicates charging status :-

- When no battery is connected:
  - The red light turns on when the Type-C cable is connected and turns off after 30 seconds.
- When a battery is connected and the Type-C cable is plugged in for charging:
  - The red light flashes.
- When the battery is fully charged via the Type-C connection:
  - The red light turns off.

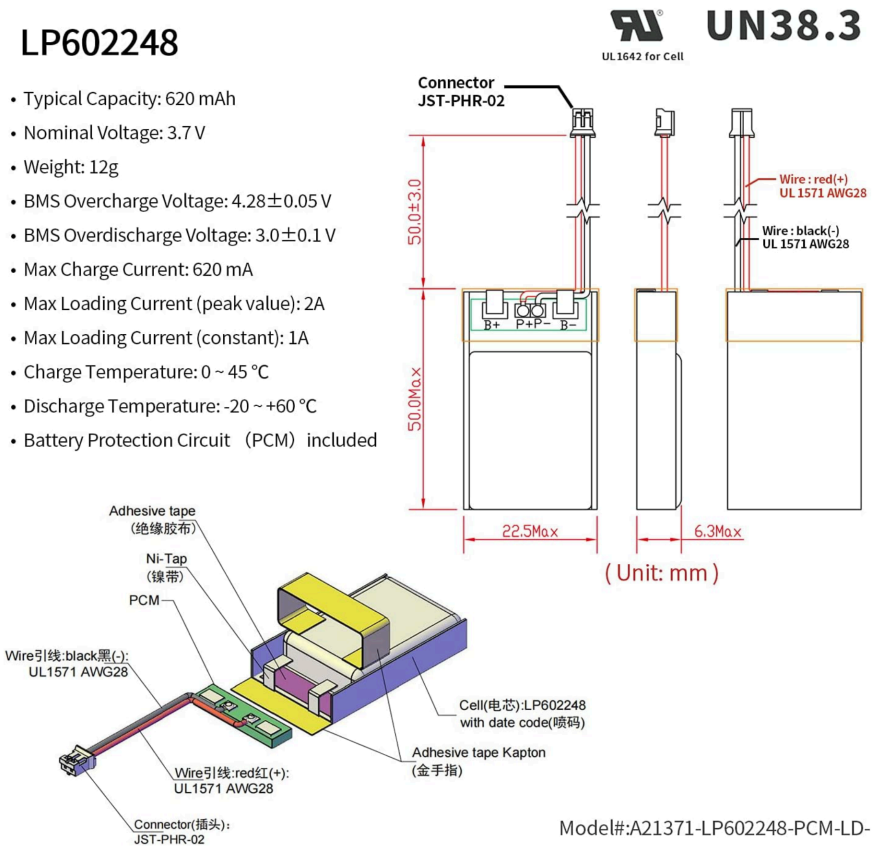
Battery Selection

Since the choice of battery is dependent on the particulars of the installation a battery is not provided with the unit.

The controller is designed to work only with a 1S Li-Po cell with a 2mm JST style connector.

> “WARNING – Battery fire or injury. If the battery cannot supply 1 A continuous or lacks internal protection, it can overheat, ignite, or fail. Use a protected 1S Li-Po rated ≥1 A continuous from a reputable manufacturer.”

An example of a suitable battery is the LP602248 from EEMB.



Battery Charging

The controller has an on board charger which provides a maximum battery charge current of approximately 100 mA, a limit set for thermal dissipation reasons in the charger and to provide safe charging for low capacity cells. The charger is designed for Li-Po cells with high-precision linear constant-current and constant-voltage charging. It can automatically complete the processes of pre-charge, fast-charge, trickle floating charge, voltage fold-back holding, resistive voltage drop compensation and recharge. For more information please see the SGM40567 data sheet included as part of the documentation.

The supply for the charger can be either via the USB-C socket (500 mA @ 5 V) or via the connections for external power. External power can be between 12 V and 50 VDC, is protected from inductive spikes above 51 V and is insensitive to polarity. The current draw will depend on the applied voltage. The buck regulator used in the power conditioning stage has a maximum input voltage of 70 V which is well above the TVS clipping voltage of approx 55 V..

The controller can switch seamlessly between internal battery power and external charging without operator intervention. Note that external power and USB power should not be applied at the same time.

With the current firmware the firebox LEDs of the 58XX and B4 locomotives will illuminate if external charging power is detected providing an easy way to check on charging status from outside the locomotive. Additionally the UI will show a ‘charging’ indicator although note that this is an indicator of external power rather than the actual charging status. However the battery voltage can be used to estimate the actual state of charge.

Battery Care

Please read Appendix 1 for more information on battery care and safety.

For long life Li-Po batteries should not be left fully charged or discharged for extended periods and ideally should be stored at approximately 50% charge (3.8 V). Always isolate the battery after use either with an external power switch or by the battery link at the front of the PCB to avoid the MCU completely draining the battery.

Although in normal operation the battery should not itself get hot, care should be taken in the location of the battery to keep it away from any heat sources. On the controller PCB these are located on the top side but as heat sinking is provided by the inner layers of the PCB the controller can become warm with use.

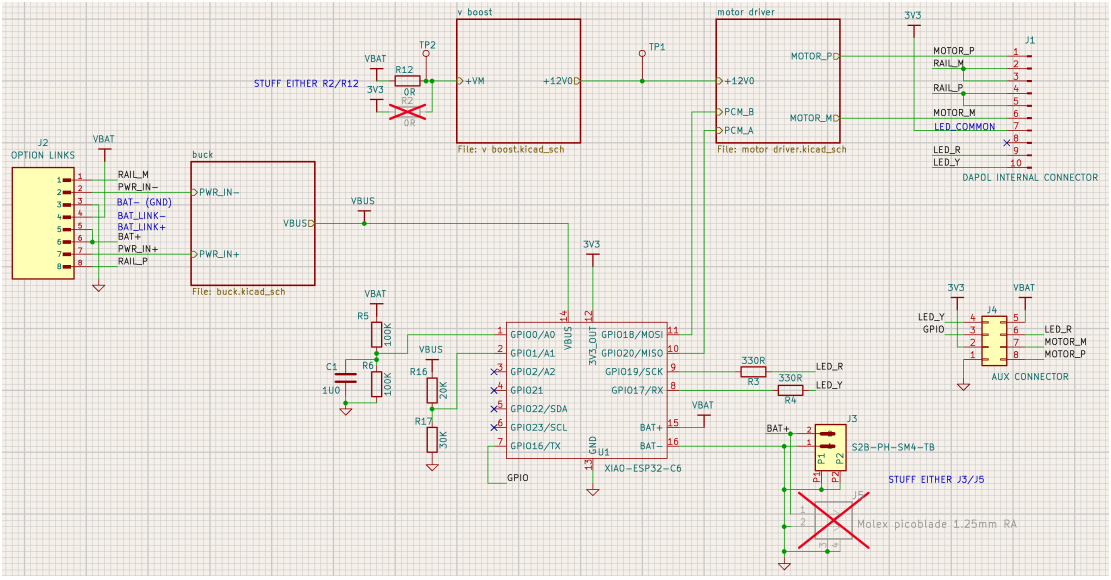
Care should also be taken to avoid compressing or bending the battery during installation as this will damage the cell.

Check your batteries regularly for any signs of damage, leakage or swelling of the cell. If you notice any issues then the battery should be disposed of safely. Do not dispose in your normal waste but rather take it to a recycling centre for safe disposal.

Circuit Description and Operation

The schematic of the controller is provided in Appendix 3 and data sheets for the silicon devices and MCU can be found in the docs folder of the distribution.

The block diagram of the controller is very simple with everything centred on the MCU and battery.



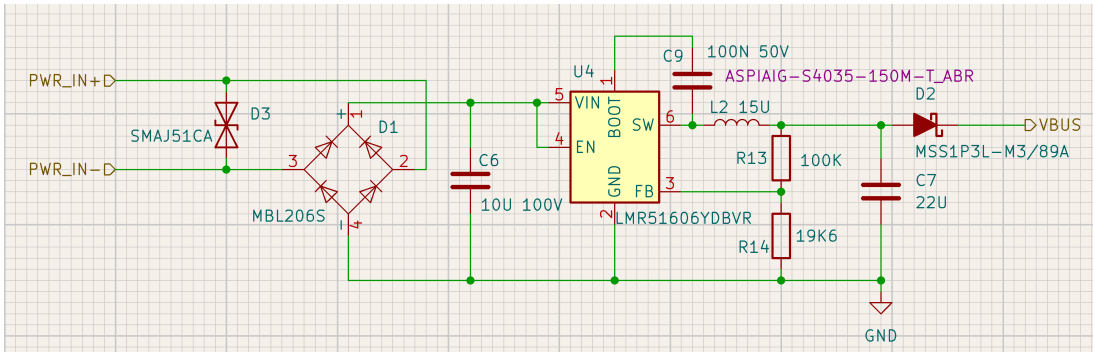
An input power conditioning stage converts the variable input voltage to a stable 5 V at a maximum current of 600 mA which is diode OR'd together with USB VBUS. The MCU uses VBUS as a source for the battery charger and as the input for a 3.3 V regulator which both powers the MCU and provides an output for other devices.

To provide a stable 12 V supply for the motor driver there is a boost stage powered either directly from the battery or the 3.3 V supply (for operation without the battery) and can supply a maximum of 1 A to the motor driver.

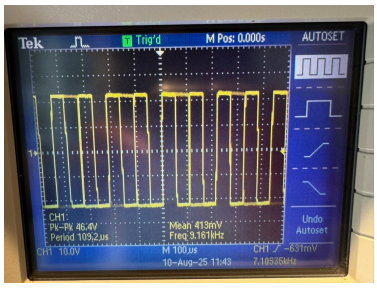
The motor driver is a standard 'H-bridge' device controlled by a pair of PWM signals from the MCU controlling both speed and direction of the motor. The motor driver is protected against under voltage, over current and over temperature conditions.

The MCU ties everything together and is based on the ESP32C6 chip, chosen for low cost and a well supported eco-system with huge amounts of third party software and learning materials. The MCU is dual core and integrates Wi-Fi/Bt radios, RAM and Flash program storage and can be programmed via the USB connection. Outputs from the MCU drive the motor controller and firebox LEDs whilst inputs allow the sensing of VBUS and battery voltage.

Input Power Stage



The input power stage is designed to tolerant a bipolar wide voltage range power source. This might be 12 VDC from a classic analogue system or up to 50 V p-p from a DCC system. Note that typically a DCC track will be driven at double its power supply voltage so an 18 VRMS AC transformer can result in a near 50 V p-p square wave on the track.On the right is a scope trace from such a system.



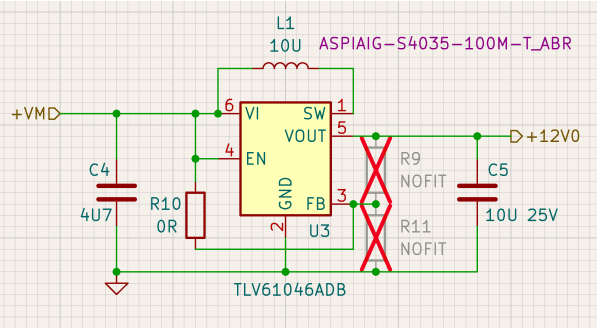
A DCC system may also exhibit over/undershoots of the digital waveform on the track due to inductive effects of the track and wiring with the potential to add several 10s of volts to the track voltage.

To protect from the inductive spikes the controller has a bi-directional TVS diode across its power input (D3) and this diode will clamp noise spikes to approx 55 V. The input power is then rectified with a full wave bridge and smoothed by C6. Note that C6 has a 100 V rating so as to minimise the derating of its capacitance with voltage.

The buck regulator chosen is the TI LMR51606, a wide-Vin (4-65 V), synchronous buck converter capable of driving up to 600 mA load current. The device has built-in protection, such as cycle-by-cycle current limit, hiccup mode short-circuit protection, and thermal shutdown in case of excessive power dissipation so is ideal for this application. R13 and R14 set the output voltage to 5 V, C7 smooths the output voltage and D2 stops any external VBUS power from USB back feeding into the regulator.

Note the current limit of 600 mA, this is well above the normal load of approx 200 mA for the battery charging and MCU giving 400 mA of headroom for any additional devices connected to the expansion connector’s 3.3 V pin.

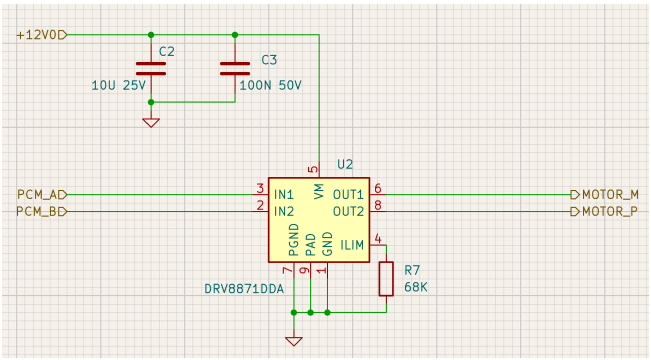
Motor Voltage Boost



The boost stage is designed to take the 3.5-4.2 V Li-Po battery voltage up to 12 V which is more suitable for the motors used in model locomotives. This stage may be alternatively driven from the 3.3 V regulated supply allowing the controller to function without a battery in ‘static’ applications such as servo drive for signals and turnouts where external power can be continuously supplied.

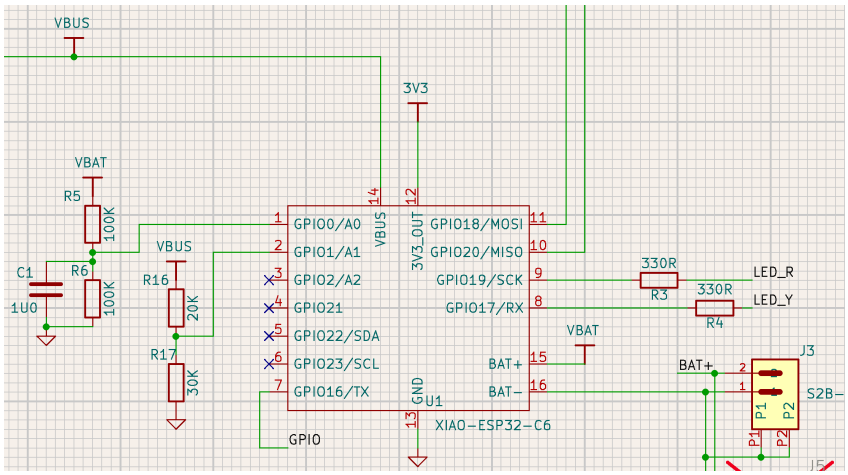
The boost regulator chosen is the TI TLV61046A which is a synchronous device with a 30 V power switch and an internal default 12 V output voltage setting. If a different output voltage is desired (in the range 6-30 V) R10 should be removed and R9/R11 fitted. The values of R9/R11 can be calculated by reference to section 8.2.2 of the TLV61046A data sheet, a copy of which is in the distribution’s docs folder. The TLV61046A has typical 980 mA switch current limit and implements output short circuit, output over voltage and over temperature shutdown.

Motor Driver



The motor driver stage uses the classic DRV8871 ‘H-bridge’ device to generate 12 V 1 A bi-directional signals to power the motor under the control of two logic level inputs from the MCU which are pulse-width modulated to control the motor’s speed and direction. Setting both inputs low forces the device into a low-power sleep mode which is used to minimise battery drain when the locomotive is stopped. The DRV8871 device has current regulation circuitry using R7 to set the current threshold to approximately 1 A. The value of R7 can be calculated by reference to section 7.3.3 of the DRV8871 data sheet, a copy of which is in the distribution’s docs folder. The device is fully protected from short-circuits, under-voltage, over-current and over temperature events. When the fault condition is removed, the device automatically resumes normal operation.

MCU



The MCU is an XIAO ESP32C6 module from Seeed Studio chosen for its compact design and on board Li-Po battery charger. The ESP32-C6 SoC contains two 32-bit RISC-V processors, a high-performance processor running up to 160 MHz, and a low-power processor running up to 20 Mhz and contains 512KB of SRAM and 4MB Flash on the chip. Aside from the normal mix of communication interfaces (I2C, UART) etc the SoC also has hardware subsystems supporting Wi-Fi 6, BLE and Bluetooth 5.0.

I/O outputs of the MCU drive the two motor controller PWM signals, two additional PWM outputs with current limiting resistors for LEDs (used in this design for firebox effects) and a spare GPIO on the expansion connector. There are four unused GPIO pins which don't have any function in the controller design but can be simply used for additional functions by soldering to the exposed module pads.

A digital GPIO input with a 5:3 resistive divider is used to sense VBUS and an analogue input with a 2:1 resistive divider is used for battery voltage monitoring.

Power steering circuity built into the module allows it to be powered either from the battery (VBAT) or USB (VBUS) with seamless change over between the two. This power is fed to an onboard 3.3 V buck regulator that derives the processors power supply and also feeds to an external module pin which feeds the expansion connector and optionally the motor driver boost converter for operation without a battery.

An onboard Li-Po battery charger based on the SGMICRO SGM40567 is powered by VBUS and connected to the VBAT pin.

The data sheets folder contains an ESP32C6 data sheet together with other documents about the module and a tutorial book covering programming the ESP32C6 in the Arduino IDE environment

Section 3 - Firmware Notes

Introduction

A feature of the Loco Controller is simple to change firmware and as such you are encouraged to modify the supplied examples to suit you own needs. The examples are all based around the Arduino which has a vast community supported ecosystem with many tutorials and libraries, and simple integration with many sensors and components. Its cross-platform IDE and C++ based programming language is very easy to learn and there are thousands of tutorial web sites and videos covering all aspects of programming and libraries.

This section will cover some of the general techniques applicable to the controller such as controlling digital and analogue GPIOs, driving motors, LEDs and servos as well as communicating via Wi-Fi. It will then go into the supplied examples in more detail showing how those techniques are applied in practice.

For more background in programming the ESP32 in the Arduino IDE see the excellent ‘Random Nerd Tutorials’ website which proved very helpful during development.

Basic Structure of the program

Arduino programs are called ‘sketches’, have an ‘.ino’ extension and are always stored in a folder with the same name as the sketch. The folder can include other files, such as a header file included in the sketch or data files that are placed in a flash based file system on the device.

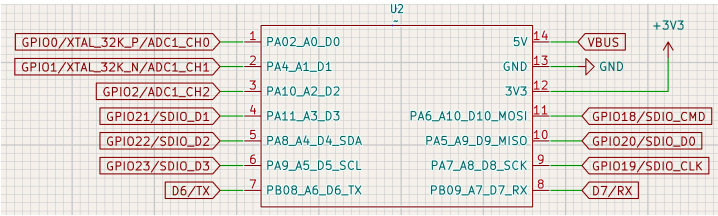
Sketches have a very simple structure with just two required functions, ‘setup()’ and ‘loop()’.

setup() - this function executes only once after the program starts. It contains things such as the mode of a pin (input or output), the baud rate of serial communication or the initialisation of a library.

loop() - code in this function is executed in a continuous loop and allows the program to respond to inputs and network requests setting outputs and sending network data as required.

The above functions are always required in an Arduino sketch, but there are no restrictions to adding further functions, headers and library files.

Pin Mapping



When using I/O functions its necessary to know the mapping between the logical ESP32 GPIO pin designation and the physical pin of the module and the diagram above shows this mapping for the XIAO ESP32C6. For example the pin designation (as used in the GPIO functions) GPIO20 maps out to pin 10 of the module.

GPIO

It's very simple to read and write GPIO pins in an Arduino program. First the function of the pin must be defined with the pinMode() function in setup():-

```
#define USER_LED          15  // GPIO pin for LED (from module schematic)
pinMode(USER_LED, OUTPUT);
```

Once defined the pin can be written or read :

```
#define LED_ON          0       // LED common is +3V3
digitalWrite(USER_LED, LED_ON); // note the use of the constant 'LED_ON' for
                                // clearer code (otherwise it would be '0')
```

```
#define VBUS        1
pinMode(VBUS, INPUT);

if (digitalRead(VBUS)) {  // do something if VBUS is present
 …
}
```

LEDs

LEDs can be driven in one of two ways, either statically as in the 'write' example above or using Pulse Width Modulation (PWM) for variable LED intensity. PWM is built into the base ESP32 Arduino library and controlled by the analogWrite() function, but a more flexible solution is the use the LEDC API. First define the pin function in setup():-

```
#define LED_Y  17
pinMode(LED_Y, OUTPUT);
```

and attach the pin to a PWM channel:-

```
#define PWM_BITS 8      // defining these constants makes it simpler to
#define PWM_FREQ 10000  // change the parameters of several PWM functions at
                        // once.

ledcAttach(LED_Y, PWM_FREQ, PWM_BITS);
```

then the LED intensity can be set with ledcWrite().

```
ledcWrite(LED_Y, 255);
```

The second parameter represents the width of the pulse and is proportional to the intensity of the LED. Note that as the LEDs are connected to 3.3 V, a pulse width of '0' would be fully on and '255' fully off.

An example of combining the GPIO and led functions is in driving the firebox LEDs of a locomotive:-

```
  if (digitalRead(VBUS)) {            // only illuminate firebox if charging

      ledcWrite(LED_Y, random(245)+10);
      ledcWrite(LED_R, random(245)+10);

   } else {                           //LEDs off

      ledcWrite(LED_Y, 255);
      ledcWrite(LED_R, 255);
   }
```

In the examples, this section is called multiple times per second to give the illusion of flickering.

Servos

Another use of PWM is in the control of digital servos. However in this case there is a servo specific library 'ESP32Servo' that can be used. This is not part of the standard ESP32 install so will have to be added to the IDE before use, see the 'Firmware Build and Flashing Guide' for details on how to do this.

First the library needs to be included and a 'Servo' instance declared:-

```
#include <ESP32Servo.h>

#define SERVOPIN   16
Servo gearboxServo;
```

and then the GPIO pin with the servo drive signal needs to be attached in setup():-

```
gearboxServo.attach(SERVOPIN);
```

Note that this should be done before any calls to ledcAttach() as the servo library always grabs the ESP32's timer0 (used to generate PWM timing) and this can conflict with the LED code.

Moving the servo just requires a call to the write() method of the object with a parameter which is the position of the servo arm in degrees:-

```
gearboxServo.write(90);
```

Motors

The final example of PWM is in the control of motor drivers. The DRV8871 motor driver used in the loco controller requires two PWM signals and these can be conveniently generated using the 'ledc' library in the same manner as the examples above.

First define the pin mode in setup() for the two control pins:-

```
#define IN1         18
#define IN2         20
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
```

and attach the pins to a PWM channel:-

```
#define PWM_BITS 8      // defining these constants makes it simpler to
#define PWM_FREQ 10000  // change the parameters of several PWM functions at
                        // once.

  ledcAttach(IN1, PWM_FREQ, PWM_BITS);
  ledcAttach(IN2, PWM_FREQ, PWM_BITS);
```

The motor direction and speed is then set by placing static signals and PWM on the control pins according to this table:-

forward	reverse
IN1 = 0	IN1 = PWM
IN2 = PWM	IN2 = 0

```
  if (direction) {  // direction is 1 for forward, 0 for reverse

    ledcWrite(IN1, 0);
    ledcWrite(IN2, motor_power);

  } else {

    ledcWrite(IN2, 0);
    ledcWrite(IN1, motor_power);

  }
```

Timers and Delays

A common function is that of a delay, to set the flash period of a LED for example. The simple way is just to use the delay() function which just waits for the number of ms in the parameter, however this is a blocking function and if used in the event loop will pause execution of the program until the delay is over which will affect the responsiveness of any user interface.

A better way is to use a timer to set a delay which can be tracked with the millis() function. The millis() function returns the number of milliseconds that have passed since the program was first started. As such it can be used to measure the passage of time by recording its present value then comparing that to a future time.

Here is an example of using millis() to toggle the state of a LED without blocking execution of the event loop (LED_TIMEOUT is a constant defining the period of flashing):-

```
  if ((millis() - lastLedTime) > LED_TIMEOUT) {

    lastLedTime = millis();

    if (ledState) {

      ledcWrite(USER_LED, 255);
      ledState--;

    } else {

      ledcWrite(USER_LED, 0);
      ledState++;

  }
```

Wi-Fi and Web Server

One of the great strengths ESP32 and Arduino development is the simplicity in using Wi-Fi and Bluetooth, it's possible to include a functioning Wi-Fi access point (AP) in your program with just a few lines of code:-

First include the library:-

```
#include <WiFi.h>
```

then define a couple of string constants for the SSID and password that devices will use to connect:-

```
const char* ssid = "Loco Controller";
const char* password = ""; // empty string disables password
```

finally instantiate and initialise the Access Point:-

```
WiFi.softAP(ssid, password);
```

That's all that is necessary.

Adding a Web Server is just as simple with similar steps, first include the libraries:-

```
#include <ESPAsyncWebServer.h>
#include <AsyncTCP.h>
```

instantiate and initialise the server on port 80:-

```
AsyncWebServer server(80);
```

lastly, start the web server in setup():-

```
 server.begin();
```

Again, that's all that is required for the server.

Typically though your program will need code to process the requests from the client and serve pages stored from the on board flash, see ESP32 Web Server using LittleFS Filesystem for more details.

Section 4 - Example Firmware

Example 1 - Semi Realistic Interface

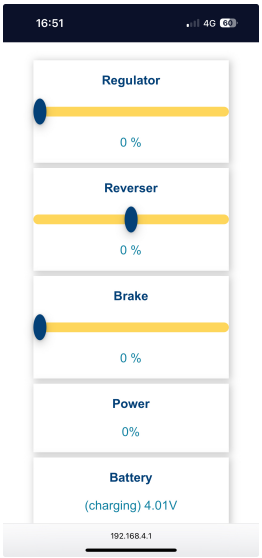
The default firmware that is shipped with the controller gives a semi-realistic set of controls implementing a very simple (and not quite correct) physics model that simulates the characteristics of a steam locomotive including momentum and friction etc.

This example is based around the use of WebSockets and derived originally from the examples on the Random Nerds site, 'ESP32 Web Server with Multiple Sliders' and 'ESP32 WebSocket Server: Control Outputs'.

The firmware instantiates a 2.4 GHz Wi-Fi access point capable of supporting four simultaneous connections together with a simple Web server appearing on port 80. This server provides a page defined by files stored in the ESP32's flash based file system.

To use the controller connect your device's Wi-Fi to the locomotives AP, which defaults to 'LocoController'. Once connected open a browser and navigate to the IP address 192.168.4.1 whereupon a page of controls will appear in the browser

The current interface offers three sliders which provide a simple emulation of a steam locomotive's controls. To startup set the Reverser to 50% and with the brake off advance the regulator to about 33% and after a short delay the locomotive should move. The firmware implements a very simple 'physics' model that takes account of momentum and friction such that if the regulator is shut off the locomotive will coast, gradually slowing down. Applying the brake will increase friction thus increasing the rate of slowing. To reverse repeat the above but set the reverser to -50%



To emulate a traditional 'direct' controller first set the regulator to 100% after which the reverser will act as an analogue control directly setting the locomotive speed.

As an aid to judging the charge state of the battery the battery voltage is displayed. Empty corresponds to approx 3.5 V and full 4.1V. When track power is detected then the battery voltage will be prepended with a 'charging' message. Note that this is a proxy for the battery charging state as it isn't directly possible to sense charging.

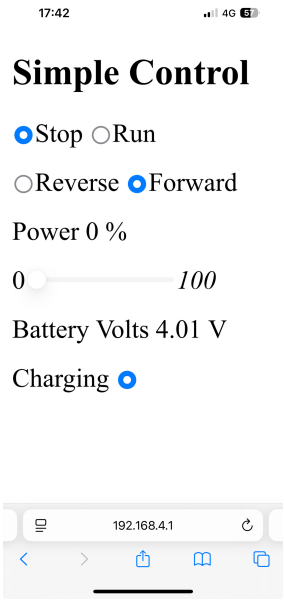
Example 2 - Simple ‘Direct’ Interface

This example implements a simple interface with a ‘direct’ control of the motor similar to a traditional analogue DC speed controller. Unlike the first example it also generates the served web page using the ‘EmbAJAX’ library rather than separate HTML, CSS and Java scripts.

EmbAJAX is a simplistic framework for creating and handling displays and controls on a web page served by an embedded device. It works by ” printing" common controls to an HTML page automatically adding AJAX code to relay control information from the client to the server and back. The programmer simply interacts with local objects while the framework takes care of keeping information in sync with the client. EmbAJAX supports an arbitrary number of pages and it provides simple but effective error handling for unreliable connections and server reboots.

In the same manner as example 1 the firmware instantiates a 2.4 GHz Wi-Fi access point capable of supporting four simultaneous connections together with a simple Web server appearing on port 80.

To use the controller connect your device’s Wi-Fi to the locomotives AP, which defaults to ‘Simple Controller’. Once connected open a browser and navigate to the IP address 192.168.4.1 whereupon a page of controls will appear in the browser



Unlike the more complex interface this example only has direction control and a power slider that directly controls the motor providing instantaneous changes in speed without any effects of momentum or friction. It does however have a ‘stop’ button which can be very useful at times!

Again as an aid to judging the charge state of the battery the battery voltage is displayed. Empty corresponds to approx 3.5 V and full 4.1 V. When track power is detected then the battery voltage will be prepended with a ‘charging’ message. Note that this is a proxy for the battery charging state as it isn’t directly possible to sense charging.

Example 3 - Pocket Railway Museum mobile App

This example works very differently to the others in that it is designed to work with the Pocket Railway Museum's IOS and Android mobile device apps. As such this example doesn't serve a web page to a browser but instead talks directly to the app using the published Pocket Railway Museum API. In the same manner as example 1 the firmware instantiates a 2.4 GHz Wi-Fi access point and to use the controller connect your device's Wi-Fi to the locomotives AP, which defaults to 'PRM Controller'. Once connected open the PRM app and follow the instructions therein.

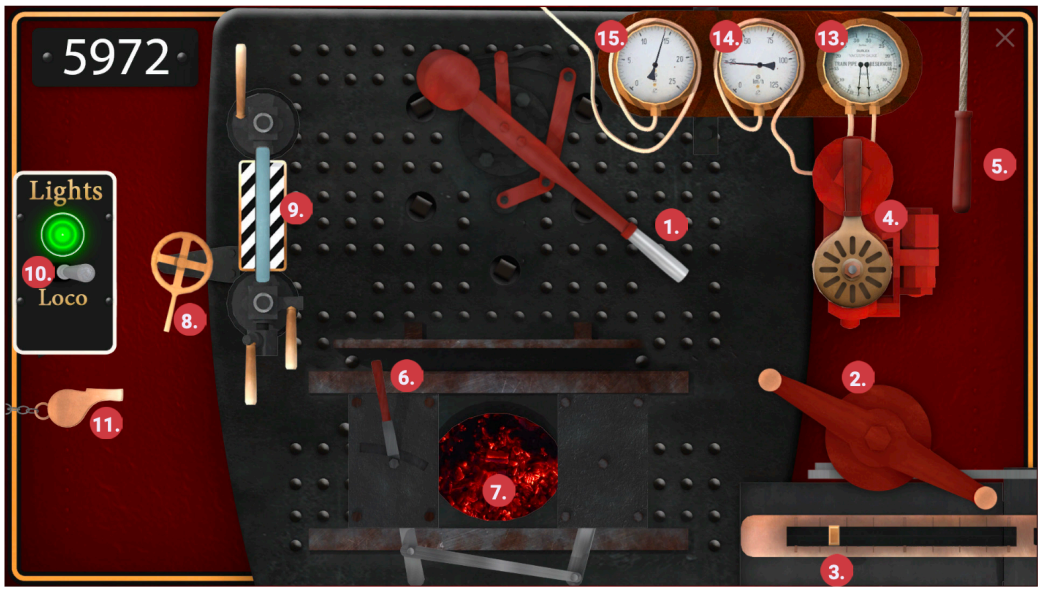
Below are the app instructions with thanks and acknowledgements to the Pocket Railway Museum.

1. How to drive the locomotive.

At Pocket Railway Museum we try to give you the most lifelike driving experience while also providing plenty of fun. Our app let's the locomotive behave like in real life. It will need time to get up to speed and also plenty of time to slow down to a halt. A locomotive is heavy, and we simulate that.

Once you get the hang of it, it's easy, but it might need some practice. Don't forget to put coal on the fire and keep your boiler filled with plenty of water, otherwise it will impact your locomotives performace.

Next is a little breakdown of all the instruments at your control. We'll also provide a video on our YouTube channel on how to control the locomotive.



1. **Regulator, used to send steam to the cylinders.**
   Open it to accelerate and close it to coast / slow down.

2. **Reverser, used to control the locomotives speed, direction and fuel consumption**
   Turn it around to change the reverser setting. (3)

3. **Reverser indicator.**
   To start driving in any direction, turn the reverser all the way to one of the outer edges. When started, crank it up towards the middle, now you go faster.

   You'll hear a change in the sound of the chuff's too, just like in real life.

   While driving, never go past the middle to the other side, or your locomotive will suddenly change direction.

   To change direction, stop the locomotive. Turn the reverser all the way to the outer edge on the other side and start driving again. To pick up speed, crank it slowly towards the middle again.

4. **Brake lever, to slow down quickly**
   This is the locomotive brake. Use it to come to a halt fast. Fast is relative here. Locomotives are heavy machines, so they need some time to come to a halt.

   Don't overshoot the station! You will get a feeling of it quickly and be able to stop it exactly where you want. Don't forget to close the regulator before braking.

5. **Steamwhistle, to make some noise**
   Just try this one out. You can tune the whistle a bit by pulling the cord less far.

6. **Firebox door**
   Open and close the firebox by sliding the doors using this handle.

7. **Firebox, for adding coal**
   Tap the fire to add coal to the fire.

8. **Injector, to add water to the boiler.**
   Open the valve to add water to the boiler.

9. **Watergauge, to check the waterlevel**
   Don't let it reach the bottom, ever.

10. **Lightswitch**
    Turns your locomotive lights on and off.

11. **Guards whistle**
    Whistle before leaving the station.

12. **Guards whistle**
    Whistle before leaving the station.

13. **Brake gauge**
    Here you can check the force on the brakes.

14. **Speed gauge, to check your speed**
    Don't speed!

15. **Pressure gauge, to check the boilerpressure**
    Keep the boiler level steady. Add more coal to the fire if it's falling.
    don't add too much or you'll smother the fire and the pressure drops.

Appendix 1 - Li-Po Battery Care

Lithium Ion Polymer (Li-Po) batteries have a very high energy density and whilst this allows long run times for models it also means that Li-Po batteries should be treated with some respect as a mistreated battery can bite back with the worst case being an uncontrolled discharge of the stored energy causing a fire.

WARNING: Li-Po batteries should always be charged with a compatible charger and should not be left on charge for extended periods.

For long battery life and safe use there are a number of simple do's and don'ts

DO

- Always isolate the battery by removing the battery link after use.
- Note the battery polarity when connecting to the controller.
- Read the data sheet for the battery.
- Keep the charge rate below 1C (ie 650 mA for a 650 mAh battery).
- Keep the battery state of charge between 20% and 80%.
- Store the battery at approx 50% charge.
- Use a battery with an internal protection circuit.
- Treat Li-Po batteries gently and avoid dropping, bending, or subjecting them to impact or pressure.

DON'T

- Leave the battery unattended during charging.
- Leave the battery on charge for extended periods.
- Leave the battery fully charged or discharged for extended periods.
- Expose the battery to sources of heat or freezing temperatures.
- Charge the battery outside the temperature range 0°C to 30°C.

Storage and Inspection

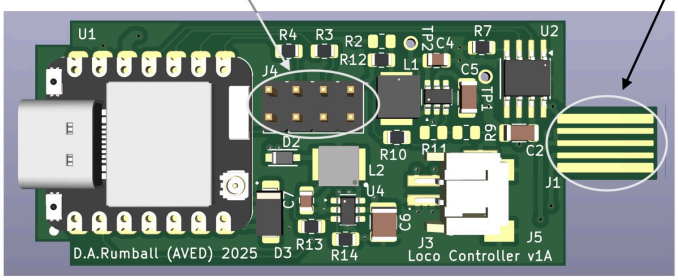
When not in use for long periods, it is recommended to discharge the Li-Po battery to about 3.8-3.9 V (about 50% of charge), remove the battery and store separately in a cool, dry enclosure free of sharp objects and any other conductive object or material.

Short circuits or damage to Li-Po batteries may not always be noticeable, check the battery for puffiness, heat, or other changes. If the battery looks damaged, remove immediately and dispose safely.

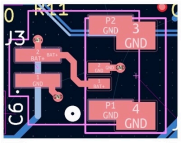
If the battery gives off an odour, generates heat, becomes discoloured or deformed, or in any way appears abnormal during use, charging, or storage, immediately remove it from your locomotive and stop using it. Do not dispose in your normal waste but rather take it to a recycling centre for safe disposal.

A good resource for more information, is 'Single Cell Li-Po Battery Care' on SparkFun.com.

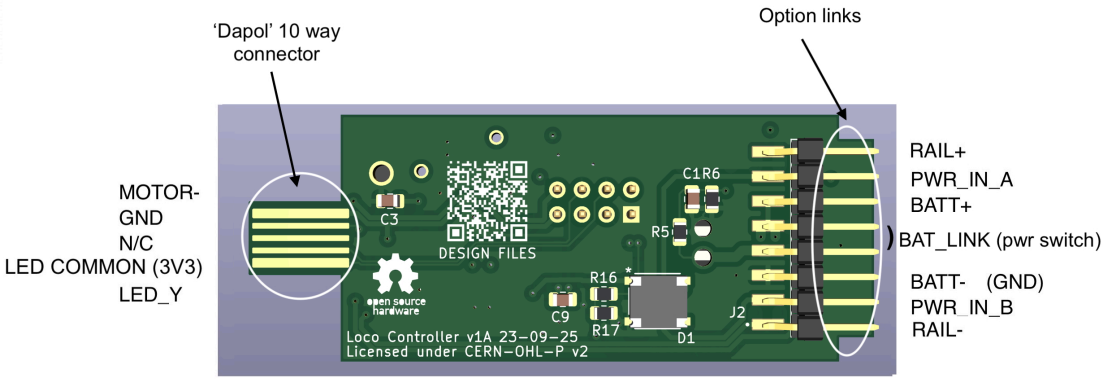
Appendix 2 - PCB Connections



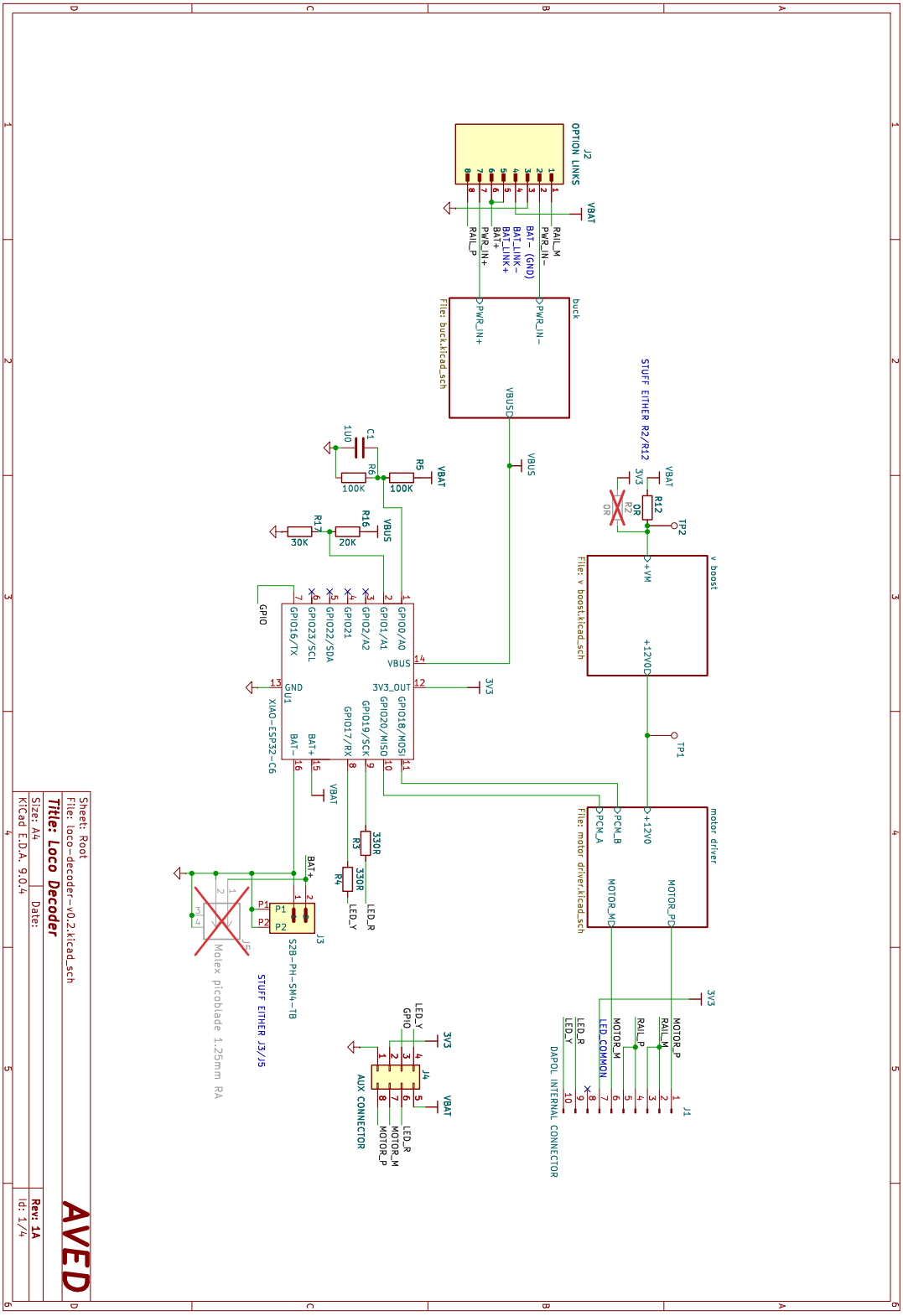
Notes:
- Isolate battery by removing link when not in use
- Note battery polarity, wrong polarity will damage the unit!
- Avoid leaving battery on charge for extended periods
- Avoid leaving the battery fully discharged
- Keep battery away from sources of heat

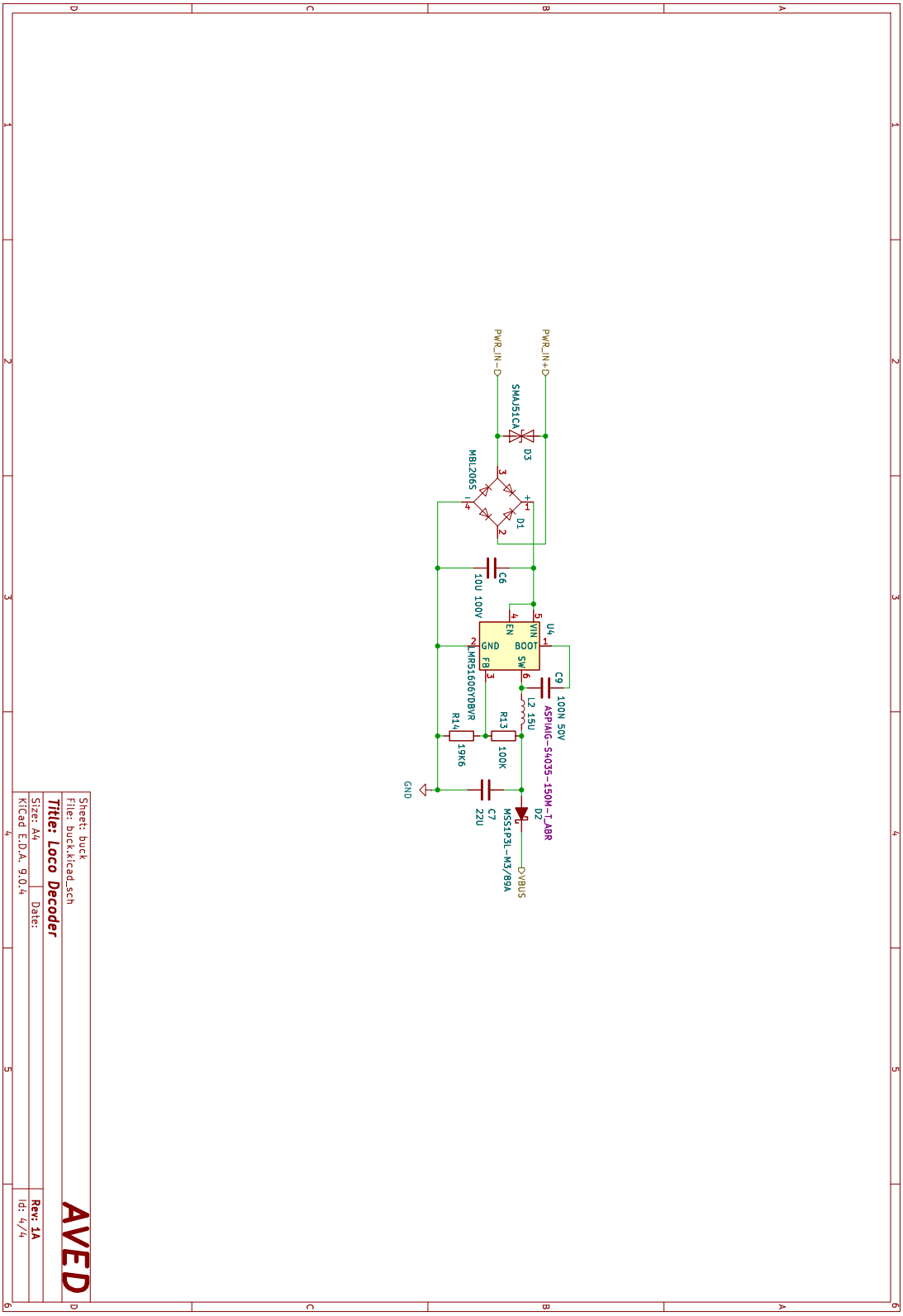


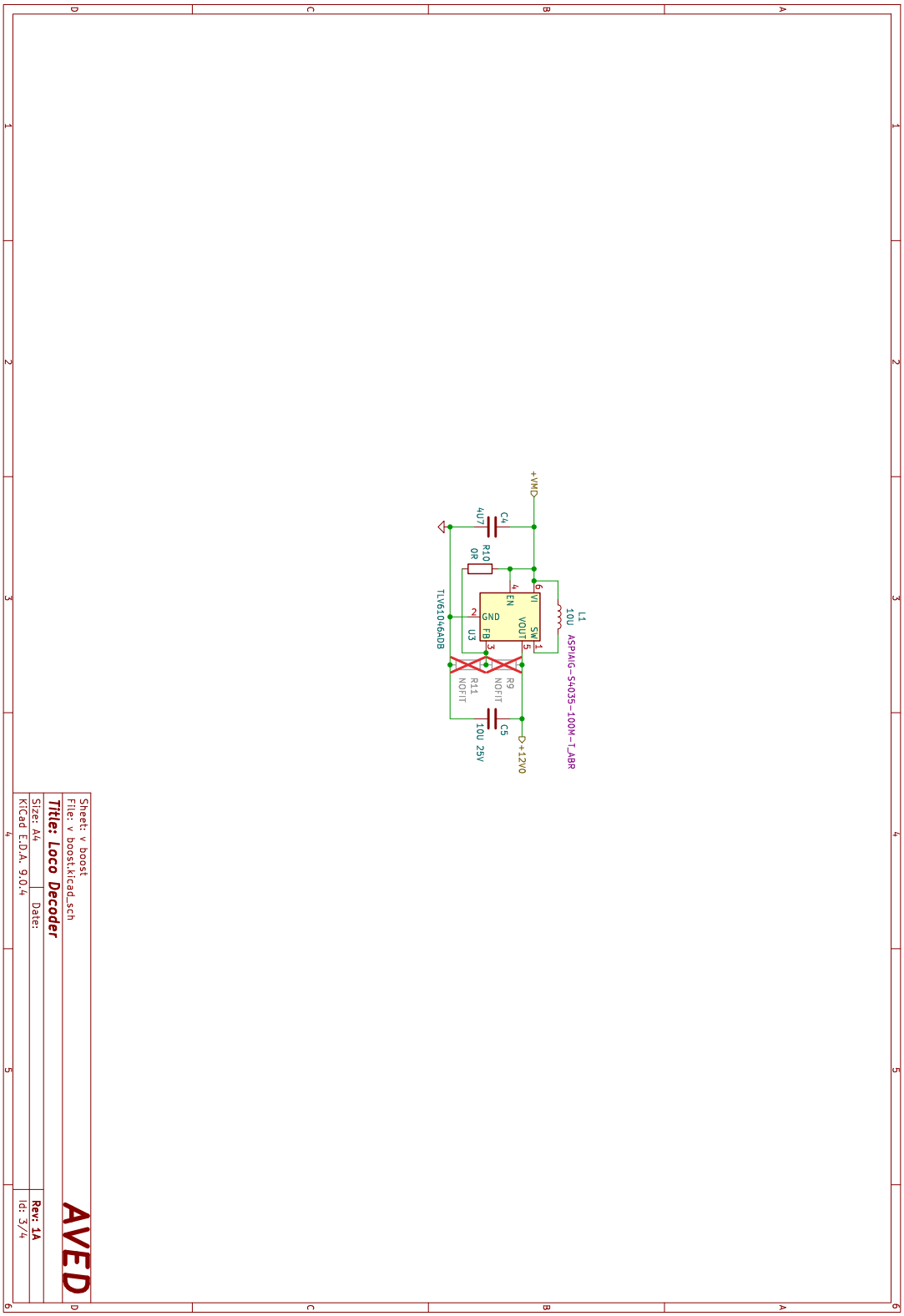
2.00mm JST connector - BAT+
BAT-
1.25mm JST connector - BAT-
BAT+

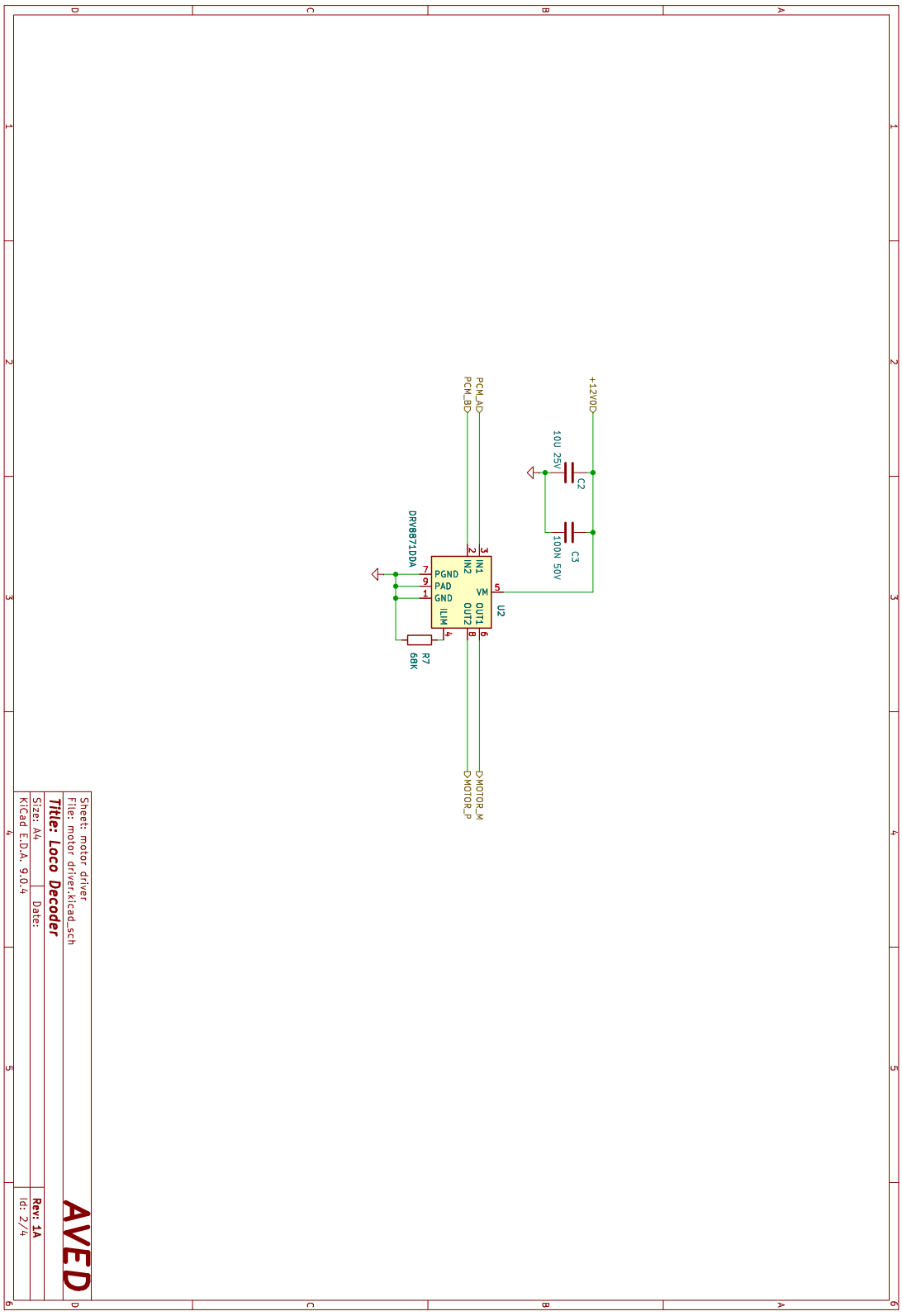


Appendix 3 - Schematics

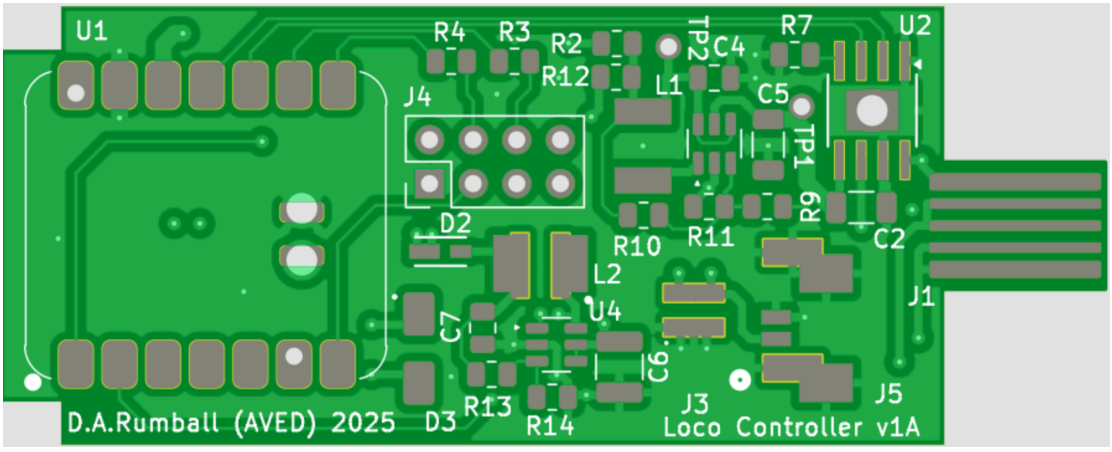



PWR_IN-
PWR_IN+
SMAJ51CA
D3
MBL2065
D1
C6
10U 100V
U4
VIN
EN
BOOT
GND
SW
FB
LMR51606YDBVR
C9
100N 50V
L2 15U
R14
19K6
R13
100K
GND
C7
22U
D2
MSSLP3L-M3/89A
VBUS
Sheet: buck
File: buck.kicad_sch
Title: Loco Decoder
Size: A4
Date:
KiCad E.D.A. 9.0.4
Rev: 1A
Id: 4/4
AVED

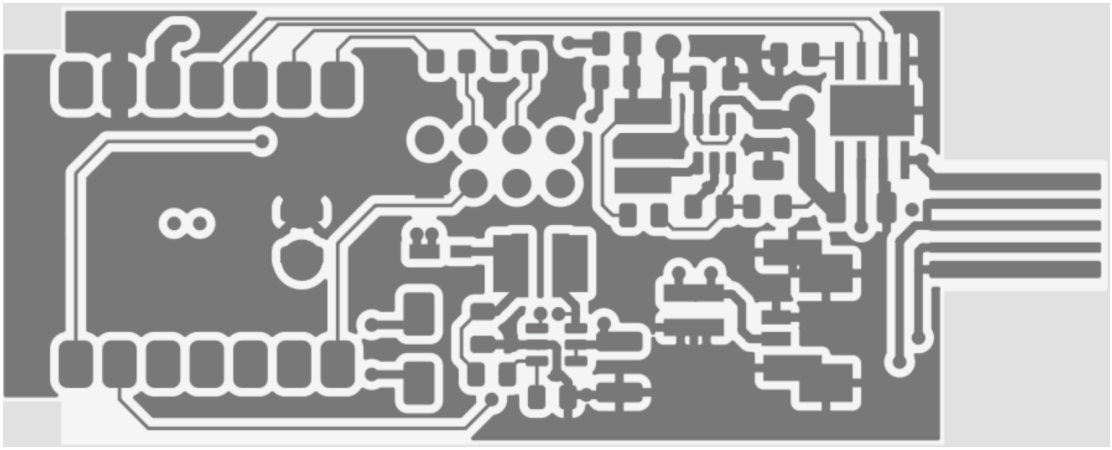

+VMD
C4
4U7
R10
0R
TLV61046ADB
U3
EN
GND
FB
VI
VOUT
SW
L1
10U
ASPIAIG-S4035-100M-TABR
R9
NOFIT
R11
NOFIT
C5
10U 25V
+12V0
Sheet: v boost
File: v boost.kicad_sch
Title: Loco Decoder
Size: A4
Date:
KiCad E.D.A. 9.0.4
Rev: 1A
Id: 3/4
AVED


PCM_AC
PCM_BC
+12VDC
10U 25V
C2
100N 50V
C3
DRV8871DDA
PGND
PAD
GND
ILIM
IN1
IN2
VM
OUT1
OUT2
U2
R7
68K
MOTOR_M
MOTOR_P
Sheet: motor driver
File: motor driver.kicad_sch
Title: Loco Decoder
Size: A4
Date:
Rev: 1A
KiCad E.D.A. 9.0.4
Id: 2/4
AVED

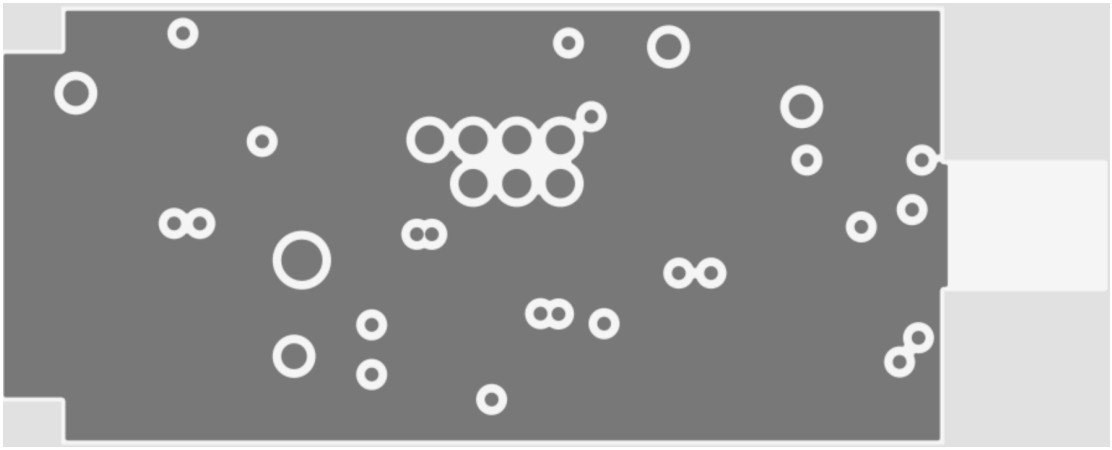
Appendix 4 - PCB Plots



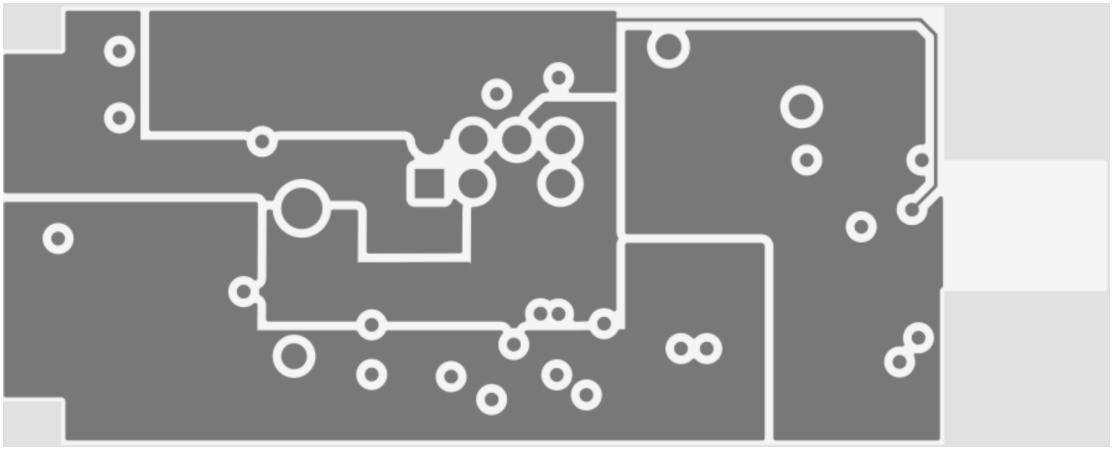
Top view



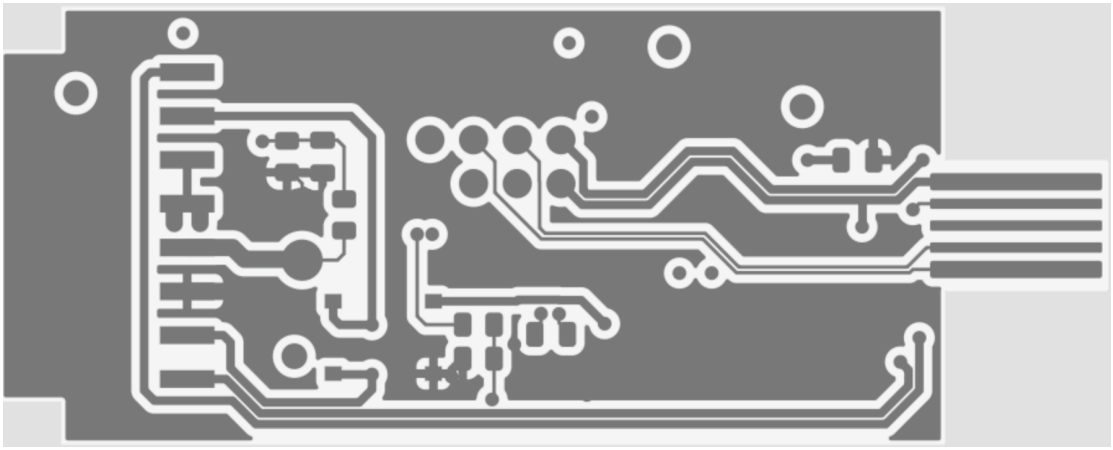
Top copper



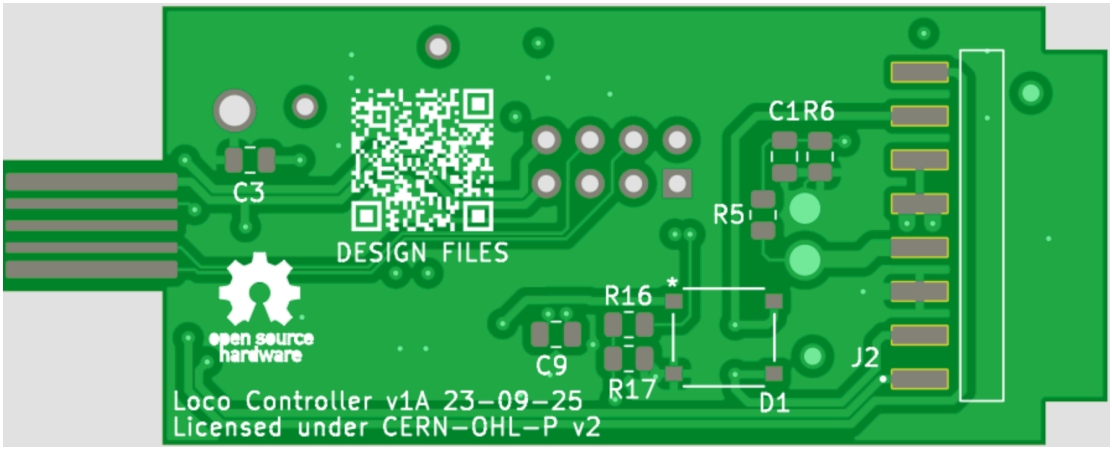
Inner copper 1



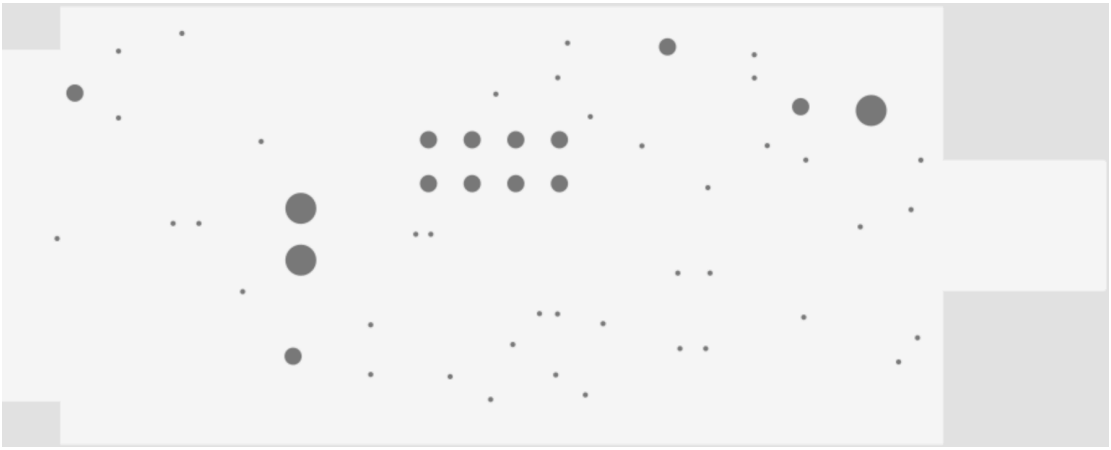
Inner copper 2



Bottom copper

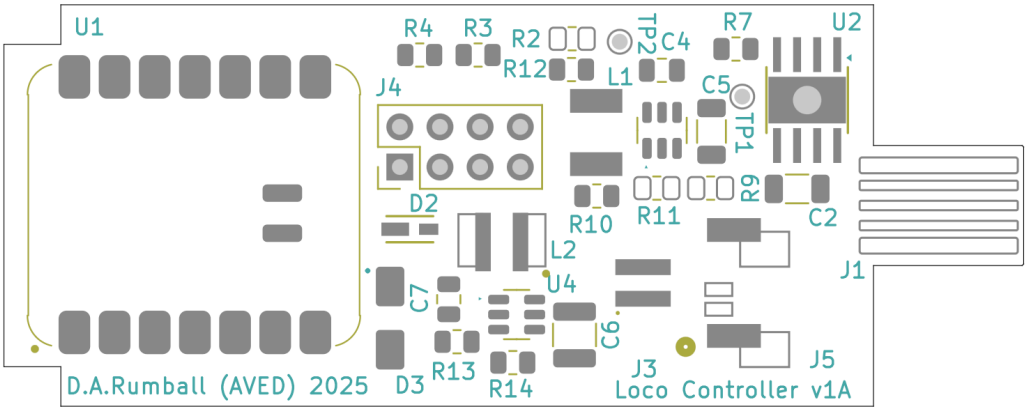

C3
DESIGN FILES
C1R6
R5
R16
C9
R17
D1
J2
open source hardware
Loco Controller v1A 23-09-25
Licensed under CERN-OHL-P v2

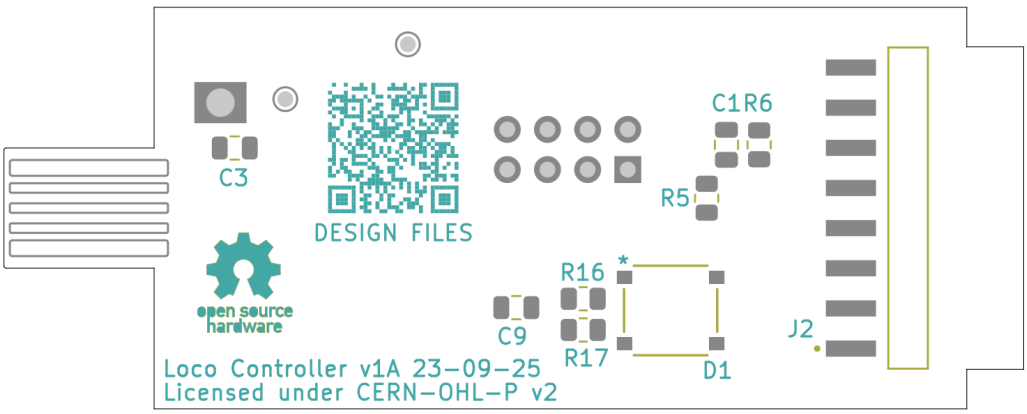
Bottom view



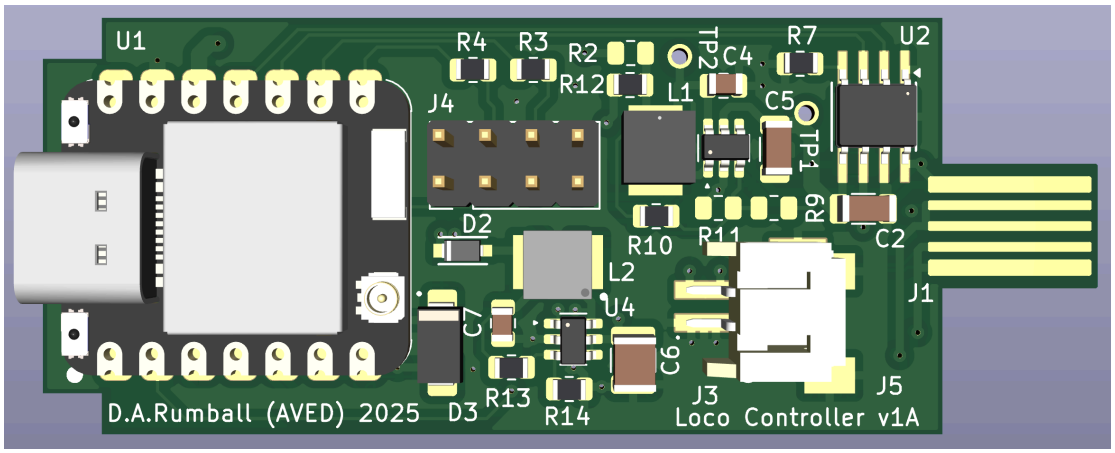
Plated drill holes

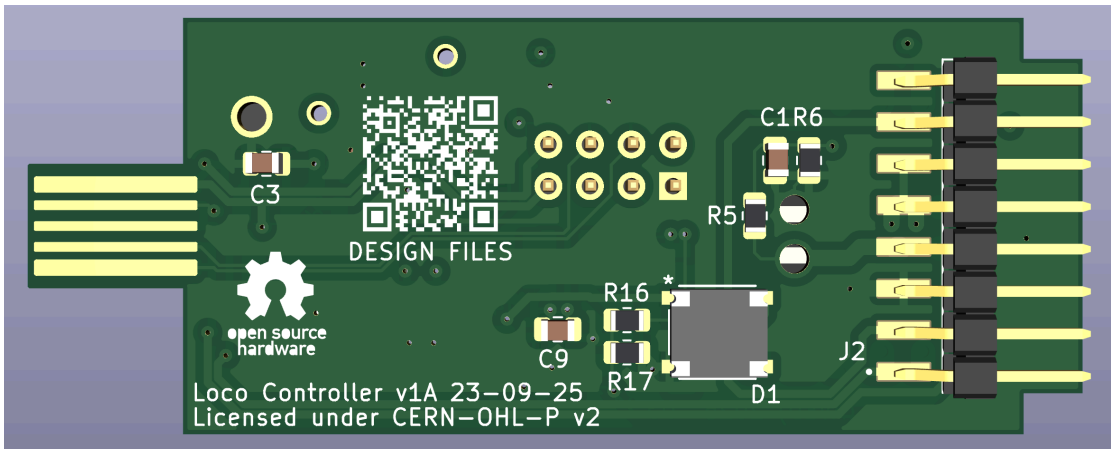
Appendix 5 - Assembly drawings





Appendix 6 - 3D renderings





Appendix 7 - BOM

bom

Designator	Footprint	Quantity	Value
C1	C_0805	1	1uF 50V X7R ±10%
C2, C5	C_1206	2	10uF 50V X5R ±10%
C3, C9	C_0805	2	100nF 50V X7R ±10%
C4	C_0805	1	25V 4.7uF X5R ±10%
C6	C_1210	1	100V 10uF X7S ±10%
C7	C_0805	1	10V 22uF X5R ±10%
D1	CD-MBL2xxS_BRN	1	CD-MBL206SL
D2	DIODE_MSP3V3-L	1	MSS1P3L-M3/89A
D3	D_SMA	1	SMAJ51CA
J2	1X8_P2.54MM_RA	1	HARWIN_M20-8890845
J3	JST_S2B-PH-SM4-TB	1	S2B-PH-SM4-TB
J4	2X4_P2.54MM_VERT	1	AUX CONNECTOR
L1	SMD,4x4mm	1	10U IND_ASPIAIG-S4035-100M-T_ABR
L2	SMD,4x4mm	1	15U IND_ASPIAIG-S4035-150M-T_ABR
R10, R12	R_0805	2	0R 125mW ±1%
R13, R5, R6	R_0805	3	100K 125mW ±1%
R14	R_0805	1	19K6 125mW ±1%
R16	R_0805	1	20K 125mW ±1%
R17	R_0805	1	30K 125mW ±1%
R3, R4	R_0805	2	330R 125mW ±1%
R7	R_0805	1	68K 125mW ±1%
U1	XIAO-ESP32C6-SMD	1	XIAO-ESP32-C6
U2	HTSOP-8	1	DRV8871DDAR
U3	SOT-23-6	1	TLV61046ADB
U4	SOT-23-6	1	LMR51606YDBVR