ZReach Z-Wave Controller Reference Design

# Introduction

## Features

* Public GitHub repository – Open-Source Repository – MIT License
* KiCAD schematic & PCB layout
  + Easily imported into other tools like Altium
  + Includes gerbers for immediate production
* Roughly hockey puck in size with an antenna poking out of the center 

Not the real thing - but similar with USB cable

* EFR32ZG23 based +20dBm & CP2102N USB->UART
* Firmware is the binary SerialAPI in the SDK
* Documentation with full details for customization and replication
* Ideally pre-certified by the Alliance and used as a test case for the SerialAPI controllers & CTT
* Development budget: $30k
  + Schematic, PCB layout, 10 protos, functional testing, RF range testing, Documentation
  + Target Assembled/tested kit cost less than $50 (not including enclosure)
    - Target BOM cost under $10@10K
* White Paper on Antenna Best Practices with simulation results from antenna expert
* Development Time Line
  + 10 weeks from agreement to delivered prototypes

# Theory of Operation

## Z-Wave MCU

The UART Rx/Tx pins are connected to the CP2102 thru 100 ohm resistors to allow the Tag-Connect debug cable to drive the pins during manufacturing. RailTest is needed to send/receive UART commands during crystal calibration.

## Antenna Filter and Match

## USB-C Interface

## Power Supplies

USB-C provides +5V power in the typical application of a controller. The +5V is reduced to 3.3V via an LDO to power the on-chip switching regulator on the ZG23. The radio is powered from +3.3V to achieve a transmit power of +20dBm to the PAVDD pin.

When prototyping a low-power end device, the battery terminals can be populated and connected to a CR123A or similar 3V battery. No reverse battery protection is provided so the connector is polarized and the battery holder must ensure the polarization. The LDO and USB devices are not populated when developing a battery powered device.

## PCB Ground Plane

A very important part of an IoT radio is the ground plane. Ideally the ground plane is perpendicular to the antenna and is half-wavelength. At 868MHz the wavelength is 34.5cm and 920Mhz is 32.6cm. The exact dimension isn’t that critical as long as it’s close. The ½ wavelength of 17cm but a square is more cost effective than a disk, thus divide by the diagonal (1.41) to get a 12cm on a side or roughly 5”. The ground plane fills the entire bottom side of the PCB to ensure there are no eddy currents or blockages. The next layer is used to distribute power. The upper two layers are used for interconnection of the ICs. All layers are flood-filled with ground as much of the area is open to provide a large ground plane for ideal RF performance. A smaller PCB size can be easily prototyped by simply making the PCB smaller with the potential of a loss in RF performance.

# Reference Documents

1. EFR32ZG23 Datasheet
2. EFR32xG23 reference manual
3. Z-Reach github repository
   1. See docs directory for this document and the Z-Reach datasheet
4. [CP2102N](https://www.silabs.com/documents/public/data-sheets/cp2102n-datasheet.pdf) Datasheet

# Journal

Details of the development and timeline are described here in reverse chronological order.

## 2024-02-20 – Schematic capture

Connected the USB UART to the same pins as the debug header but thru 100 ohm resistors. I could have put the USB UART to a different set of pins but then the firmware could NOT use the standard SerialAPI and would have to be customized with each release. Didn’t think that was a good idea. The series resistors are to provide some isolation when BOTH USB and the debugger are connected. The main issue is that during production, RailTest has to be downloaded and then commanded via the debug connector to the UART to Tx ON for crystal calibration. But since USB is also connected, it is also driving the UART pins. Simply adding some series resistors will provide sufficient isolation for the short time during production. Even during debug it’s not a big deal. During normal operation the resistors will help with EMI. UART speed is only 115.2K baud so the resistors will have no impact on that. The other option is to use a switch but that would be overkill and just 1 more place for things to fail and increase cost.

USB pins require ESD protection as described here: <https://www.digikey.com/en/articles/why-usb-type-c-circuit-protection-is-vital> but this is overkill IMHO. There are many triple TVS diodes but they do not seem to have a common pinout. Prices for these triple diodes are around 30 cents. Only need USB2 level ESD as we do not need to support USB3 which has tighter restrictions on the capacitance of the TVS diode. This [article](https://www.electronicproducts.com/how-to-protect-usb-type-c-connectors-from-esd-and-overtemperature/) has a good description of requirements. Want to have something with lots of 2nd sources and not be locked into a single part. KiCAD doesn’t have any diodes in 0402, mostly SOD-xxx. The CP2102 recommends SP0503BAHTG which is in the KiCAD library and digikey has 114K of them so it is easier to use this part.

The Tag-Connect debug header VAEM is tied directly to the 3.3V from the LDO. This is normally not a problem but could result in both the WSTK and the LDO driving the 3.3V pins. Since they are both at the same voltage it usually doesn’t matter. The WSTK should normally NOT drive the VAEM pin (slide the switch to USB or battery). The only time you would drive the VAEM pin is if you are debugging a battery powered device and want to measure the power which would be done using the AEM in SSv5. But in that case, the LDO would not be installed. The only question is during programming will the LDO short 3V3 to GND? That would require the USB to also be plugged in or powered via a test point which is acceptable but makes things more difficult. However, a functional test of the USB would require the cable being plugged in so maybe it is OK.

There is no QWIIC connector symbol or footprint in KiCad. I copied the ones from the sparkfun repo but I didn’t want to import the entire thing so I created project specific libraries which will also hold some other project specific stuff like the Z-Wave logo and things.

## 2024-02-16 – Project Start

Wrote the datasheet. Started Schematic capture and Theory of Operation (ToO) development.

Lots of LDOs to choose from. Diodes Inc [**AP2125K-3.3TRG1**](https://www.digikey.com/en/products/detail/diodes-incorporated/AP2125K-3-3TRG1/4470777) is 10 cents and will be the initial LDO to use (Digikey has 45K in stock). The TI [**TPS7A0333PDBVR**](https://www.digikey.com/en/products/detail/texas-instruments/TPS7A0333PDBVR/12165108) LDO I’ve used before is 33 cents but does have better Iq, temperature range and Vdropmax but the Diodes Inc is fine.

For the USB-C interface, I used this example: <https://hackaday.com/2023/08/07/all-about-usb-c-example-circuits/>. The classic question is what to connect the shield of the connector to? The Shield is the outside layer of the cable thus it can act as an unintentional antenna and then fail FCC. The example wires it to GND but that is likely to radiate more than leaving it unconnected. Nice discussion [here](https://electrical.codidact.com/posts/279876#:~:text=%E2%88%920-,Connect%20the%20shield%20directly%20to%20ground%20plane,to%20the%20PCB%20ground%20plane.) which mostly recommends connecting SHIELD to GND. There are 2 purposes for this connection – EMI and ESD. For ESD it makes sense to have a solid connection to have the entire board remain at the same potential which might shoot up thousands of volts in an ESD event. But I am more concerned about EMI which in an abundance of caution I put a small ferrite bead leading to the shield to reduce any EMI. Can replace this with a zero ohm resistor later, or leave it open or insert a capacitor or even just short the pads out. I’m not too worried about ESD as ZReach is pretty much plugged in once and left there for the duration.

## 2024-02-07 – Create Repository

Created the Repo, this file and started the hardware design in KiCAD.