Cryptonode

ECE 191 Group H

Sponsored by: 96Boards

Advisor: Robert Wolff

Team Members: Andrew Betts, Chencheng Ning, Odeh Saba, Richard Nguyen

Table of Contents:

Background

* 1. Statement of Work

1. Design Approach
   1. Research
2. Project Overview
   1. Ethics and Safety
   2. Crypto functions
   3. Hardware
      1. Battery Charger
      2. Screen
   4. Enclosure
3. Implementation
   1. Software
   2. Hardware
4. Budget
5. Conclusion - left: software

References

Appendix A - Hardware

Appendix B - Code Samples

Introduction

* 1. Background
     1. Cryptonode on 96Boards was designed as an inclusive product to lower barrier of entry to exchange of crypto-tokens (commonly referred to in the financial sense as cryptocurrency or cryptocoins). The aim of the project is not only to implement these features, but also, to do so on an ARM platform (Advanced [Acorn] RISC [Reduced Instruction Set] Machine).
  2. Statement of Work
     1. The Cryptonode project seeks to meet the above goals stated in the background. Namely to implement a platform over which the core functions of a cryptocurrency can be used. We take the “core functions” to be as follows:
        + Wallet
          1. Allows a place of secure storage, and exchange, of cryptocurrency
        + Miner/Generator
          1. Allows user to produce more cryptocurrencies through different methods
        + Node
          1. Allows user to host a part of the blockchain network.
     2. Hardware
        + Screen
          1. A serial display should be able to display pertinent information about the user such as current fiat exchange rates, and wallet balances
        + Power
          1. Able to power arm platform on which our functionality will be running
          2. Able to be ran off of battery
          3. Able to run off of Wall Power
        + Case
          1. Packages the above hardware into a discrete unit while still providing access to relevant ports on the device.

1. Design Approach
   1. Research
      1. In order to implement a crypto box on Linux, specifically the Dragonboard 410C, we first needed to investigate the operation of cryptocurrencies. Cryptocurrencies operate on a distributed node network in which each node views and verifies transactions from address to address. Once our understanding of the mechanism behind cryptocurrencies was established, we could then move to implementation in code. Implementation was dependent upon which library to use as reimplementation of existing open source code libraries would lead to necessary development times. After additional research and some recommendations we settled on using SmartCash as our choice of cryptocurrency for the project. SmartCash implements much of the code used for Bitcoin development and made sense for development. We settled for using the common Electrum wallet to control our cryptocurrency operation. Once settled on how the software would be programmed, we then shifted our attention to hardware in the form of the display and battery. The display and batteries interface to the board while the case would house all three components while still allowing interfacing to the hardware ports.
      2. As suggested, a 12V 2A input is desired for the dragonboard 410C. Considering the size and capacity of all kinds of batteries, lithium polymer batteries are chosen for their compact size and higher energy density. To balance size and efficiency, two LiPo batteries in series working with a high efficiency boost converter is the best way to support the device with a decent amount of battery life. Charger and battery protection are selected based on 2S configuration and the circuits follow typical applications on datasheet with certain modifications to meet specifications. The boost converter is a switching regulator type. It is chosen for high efficiency and low heat generation. The important thing about switching regulator is layout. Proper layout must be implemented for proper operation, such as short power loops, and short distances from capacitors to the IC. To shorten charging duration, since batteries have fairly big capacity, a switch is designed with MOSFETs to give priority to DC input to power the dragonboard while charging battery pack. The functionality is achieved by monitoring two power source voltages to turn the monitoring PMOS on or off. The monitoring PMOS will then turn the controlling NMOS on or off. And the NMOS controls the operation of the final conduction PMOS. the reason for three transistors is that two transistors design is not stable as load increases, and generates a large amount of heat. To achieve uninterrupted switching between two sources, a large capacitor is placed at the output of the switch to act as a backup battery during switching.
2. Project Overview
   1. Ethics and Safety
      1. Caution must be taken when handling LiPo batteries due to its nature. Avoid any physical damage to battery packs.
      2. Team Members agreed upon a set code of conduct before start of work. Among these, punctuality, consistency and communication were emphasized.
   2. Crypto Functions
      1. Crypto Operations mainly operate off of the Electrum-Smart Wallet. This program contains many functions useful for sending and receiving cryptocurrencies. Electrum-Smart typically contains a py-Qt5 gui as well as cmdline functionality but on our specific arm board, only the cmdline tools work. This is fine as our main operation case will be a cold storage wallet. A cold storage wallet consists of a computer generating transactions, transferring the transactions to an offline device, having the transactions signed offline, and then broadcasting these transactions to the network on the online device. The use of such functionality is to protect our wallet from ever seeing the internet and reducing the chance of theft. As the device will operate headless, the device needs to recognize the storage device containing transactions, open our wallet and sign the transactions. In debian linux, storage mounting is handles by the UDEV service. As such we use a combination of UDEV and SYSTEMD to recognize the storage mount. We use the aforementioned services to call Electrum-Smart to sign the files. It is important to note that the wallet on the device needs to match the public wallet on the desktop in order for the device to sign.
   3. Hardware
      1. Power System
         * AC/DC wall mount adapter provides 9V 5.5A maximum output to support power system and the dragonboard.
         * Power system contains five major parts.
           1. 2S 6000mAh LiPo battery pack
           2. 2S LiPo/Li-ion battery pack charger
           3. 2S LiPo/Li-ion battery pack protection with balancing function
           4. MOSFET switch for DC input priority
           5. Step up switching regulator
         * Charger utilizes a BQ2057W advanced linear charge management IC. Charging current up to 2A and is easily programmable by changing the Rsns in charger circuit.
         * Battery protection utilizes a BQ29200 PMIC. the circuit discharge higher voltage battery cell at 7mA until 0V difference between two battery cells. Activation threshold is 30mV difference between the two cells.
         * MOSFET switch constantly measures voltage difference between battery pack output port and DC input port with a PMOS. When VDCin > VBAT, all MOSFETs are off to disconnect battery from dragonboard. When VDCin < VBAT, all MOSFET are on to allow current flow from battery pack to switching regulator. A large capacitor is placed at the output of the switch to achieve uninterrupted switching.
         * Step up switching regulator utilizes a LM3481 controller. It regulates a 12V output with 2A maximum output current.
      2. Screen
         * SunFounder 20x4 LCD display is connected to a LCM1602 I2C controller for simple pinouts.
         * DragonBoard 410c has a 1.8V and a 5V pin on the low speed expansion header. The display requires at least 2.5V so it will be powered by the 5V pin.
         * I2C signals from the board are 1.8V, so to interface with the display we use a level shifter to step up the signals to 5V. The 1.8V pin is connected to the low voltage input of the level shifter and the 5V pin to the high voltage. By connecting the SDA and SCL pins from the board to the low voltage side and the same pins from the display to the high voltage side, signals will be safely converted from 1.8V to 5V and vice versa.
         * On Debian we install the libmraa and libupm libraries to interface with the display from our code. Attempts to use the Python bindings proved to be unsuccessful, so we wrote a script in C++ to fetch string inputs from the terminal and print them to the display.
         * Simple lines of code were added to the wallet and stock ticker scripts to print relevant information to the terminal and call the C++ program to print it to display. Python functions were also made to split the information into four strings for each of the four rows of the display. We also implemented a scrolling function for strings longer than 20 characters for the last row which is reserved for the stock ticker.
         * Start-up script was written to immediately run the electrum wallet daemon script when the board boots. This is done so that wallet balance is immediately printed to the display upon boot.
   4. Enclosure
      1. Prototype Version 1: Minimal casing designed to house only the dragonboard itself. The intended function was to protect the board while keeping all complicated components hidden from the user. All ports required for the functionality of the project were still made accessible through the casing.
      2. Prototype Version 2: The second iteration was designed with our intended secondary components in mind. This design sought to be more encompassing of what our final image for the project intended to be by including housing for the lcd and space above the dragon boards housing for the necessary mezzanine. The protective design was improved however the size increased more than we would like. This was inevitable considering the size of our component.
      3. Final Design: The final design was molded with every aspect of the overall project taken into consideration. As with prototype two it includes housing for the dragonboard and the addition of a mezzanine above it. It has a flush inlet for the lcd screen and a comfortable amount of space in the housing to fit each component. The final iteration also added housing for the Li-Po batteries and the power circuit. During design the possible orientation of the components would heavily influence the size and dimensions of the enclosure. In order to keep the design as user friendly and ideal as possible the final design incorporates the batteries in their lowest profile on the sides of the case making it only marginally wider and keeping its dimensions at a desirable ratio for the intended product.
3. Implementation
   1. Software
      1. Implementation of the software ended up being split into three distinct but intertwined part. The first and most critical part is the functioning of the wallet. The wallet runs on a SmartCash port of the Electrum wallet, typically used for bitcoin. This library runs on a combination of python and bash functions. This software suite allows us to call functions to sign and send transactions. Snippets of such can be found in the coding appendix. We are able to interface with the library through its native python code through editing its library files or instead writing our own scripts and importing and calling its functions.The USB signing takes advantage of this fact and is implemented mostly in BASH, linux command line, code and calls the operation of python scripts which interface as mentioned above. The usb signing scripts also use the Linux system operations such as UDEV and SYSTEMD to avoid thread locking and implementation while waiting for the USB to be accessible. The Cryptostock ticker pulls data from Binance, a cryptocurrency exchange, to show current exchange prices between fiat and between different currencies and is implemented in standard python libraries, json and requests. The wallet and ticker are able to be displayed on the serial display powered by our i2c. In order to drive the gpio ports on the board, we use the libmraa and upm libraries to modulate our gpio pins and send data packets containing our strings.

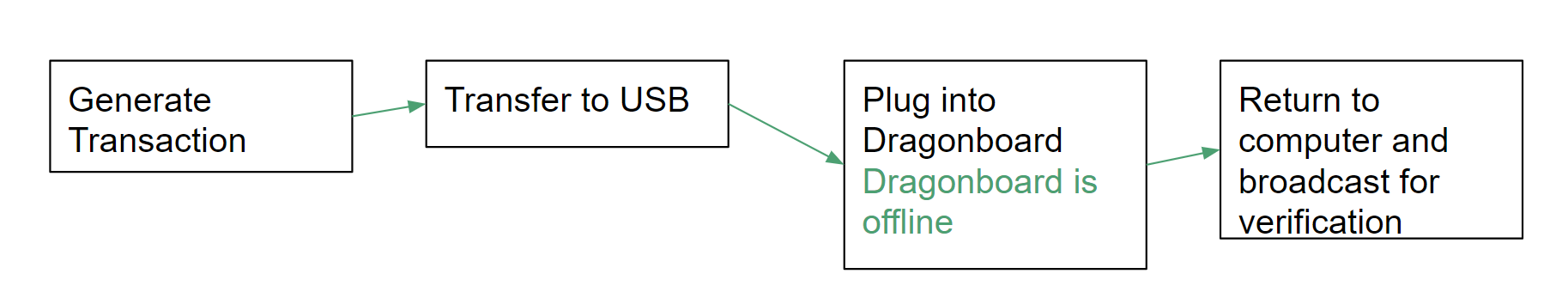


Fig 1. Flowchart

* 1. Hardware
     1. Power System

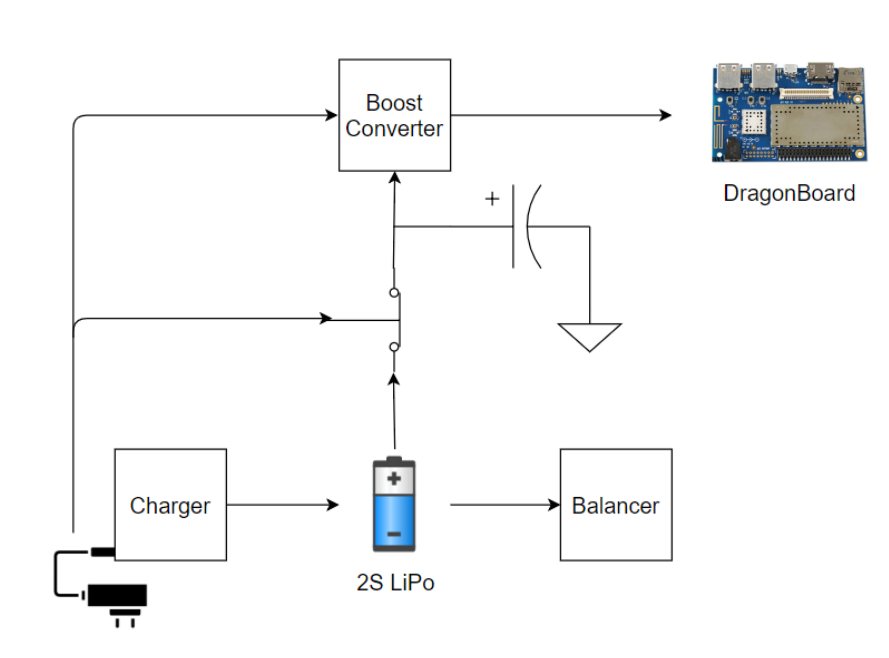


Fig 2. Block diagram of power system

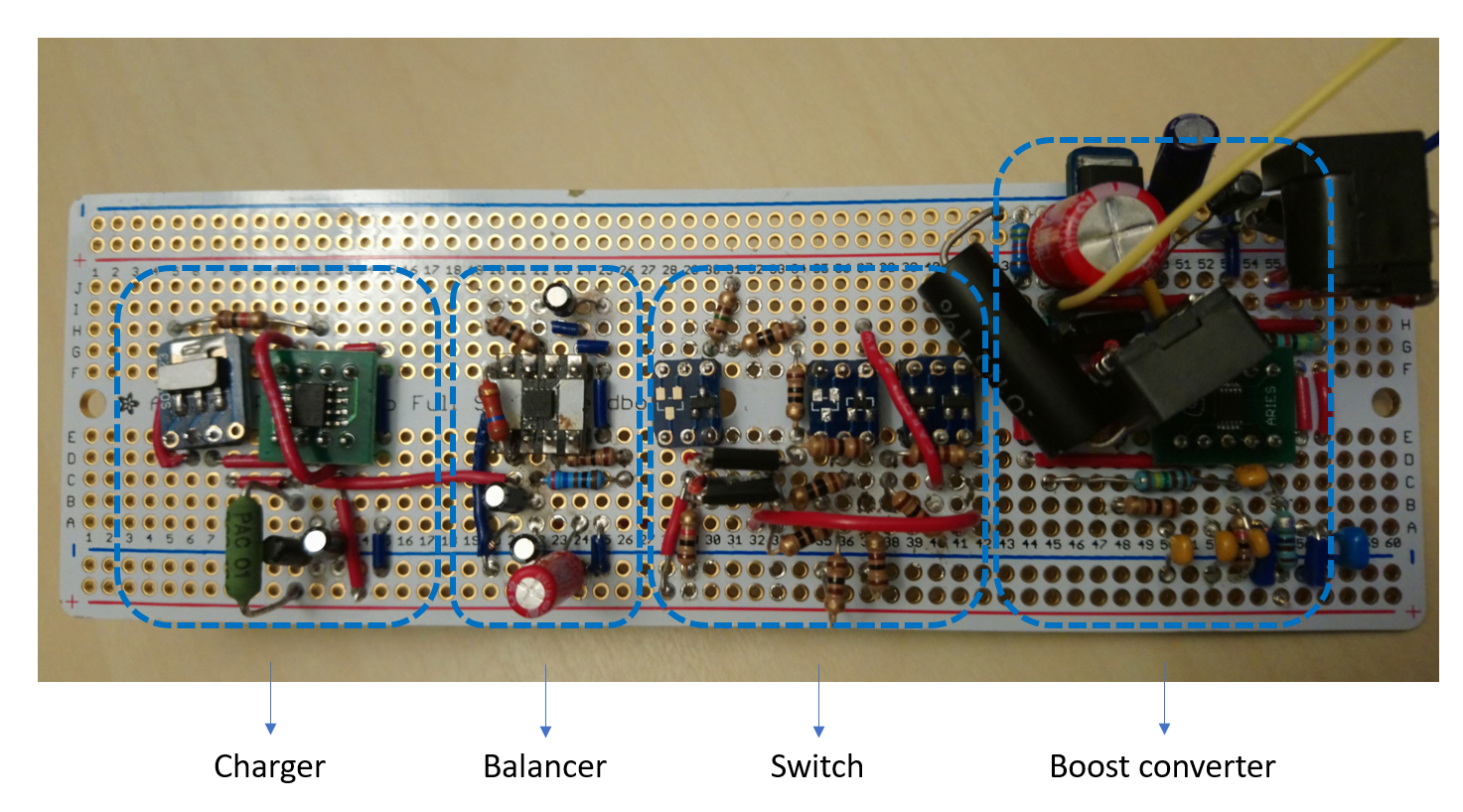


Fig 3. Implementation of power system

* + - * Charger - BQ2057W

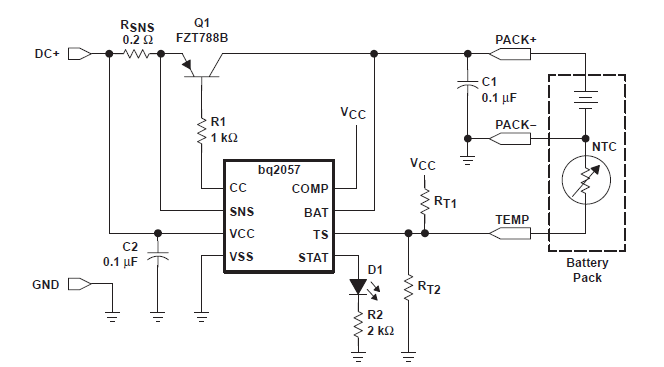


Fig 4. Charger schematic

Rsns\_min = Vsns / 2A. Heatsink required. Dissipation equation:

PD = (VI - VSNS - VBAT) \* IREG

where VI is charger supply voltage, VSNS is voltage across current sensing resistor, VBAT is battery pack voltage, IREG is desired charging current. Typical PD for 2A charging current is 5.76W at initial stage.

* + - * Battery protection with balancing - BQ29200

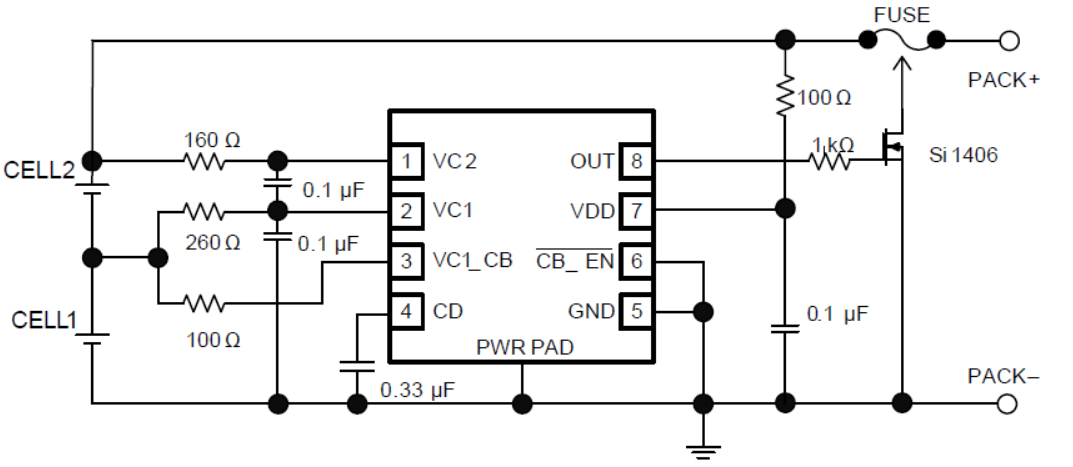


Fig 5. Protection/balancer schematic

Balancing current equation:

ICB1 = VC1 / (RCB + RCB1)

ICB2 = (VC2 - VC1) / [(RCB + RVD)+RCB2]

where VC1 is bottom cell voltage, VC2 is battery pack voltage, RCB is resistor connected between top of bottom cell and VC1\_CB, RCB1 is resistor connected between top of bottom cell and VC1, RCB2 is resistor connected between top of top cell and VC2, RVD is resistor connected between top of top cell and VDD.

* + - * Boost converter - LM3481

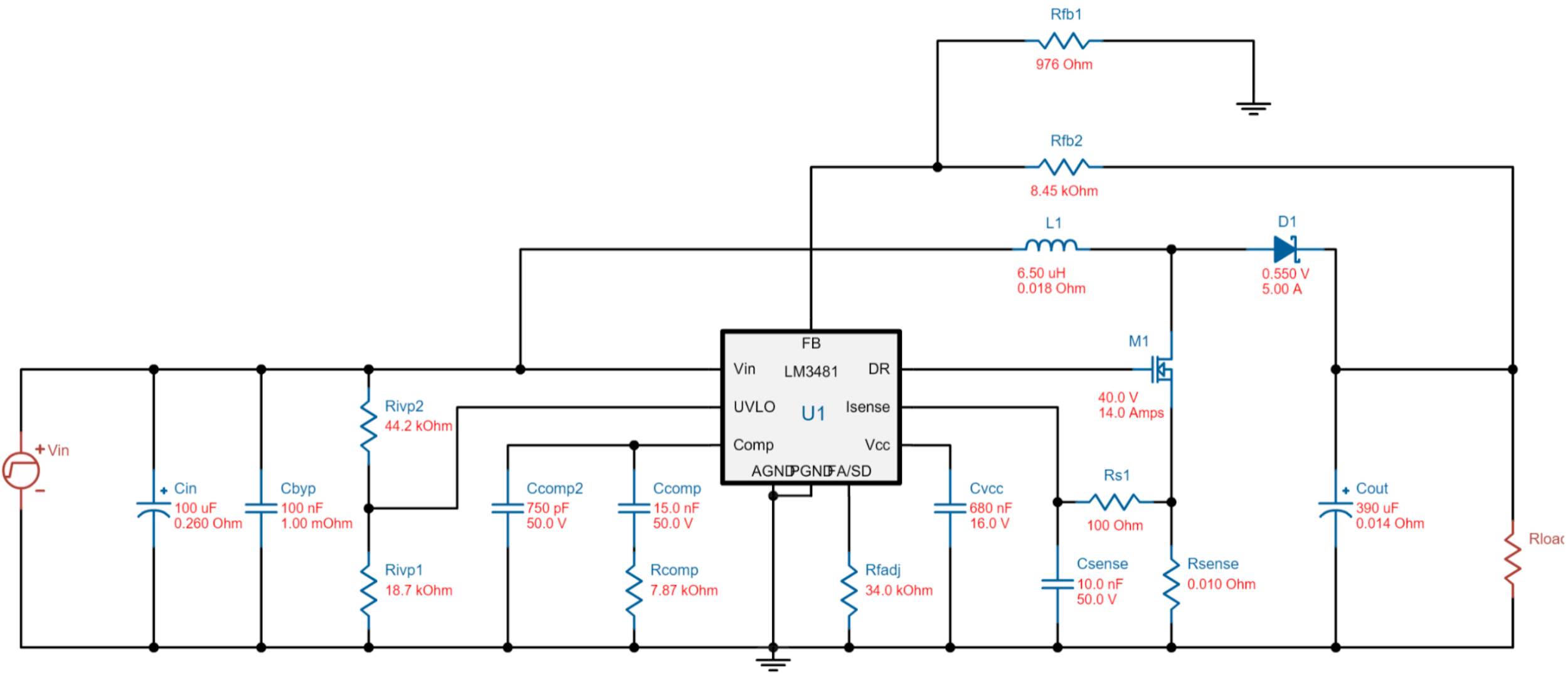


Fig 6. Boost converter schematic

Input range is designed to be 5V to 9V. Output is regulated to 12V with 2A maximum output current.

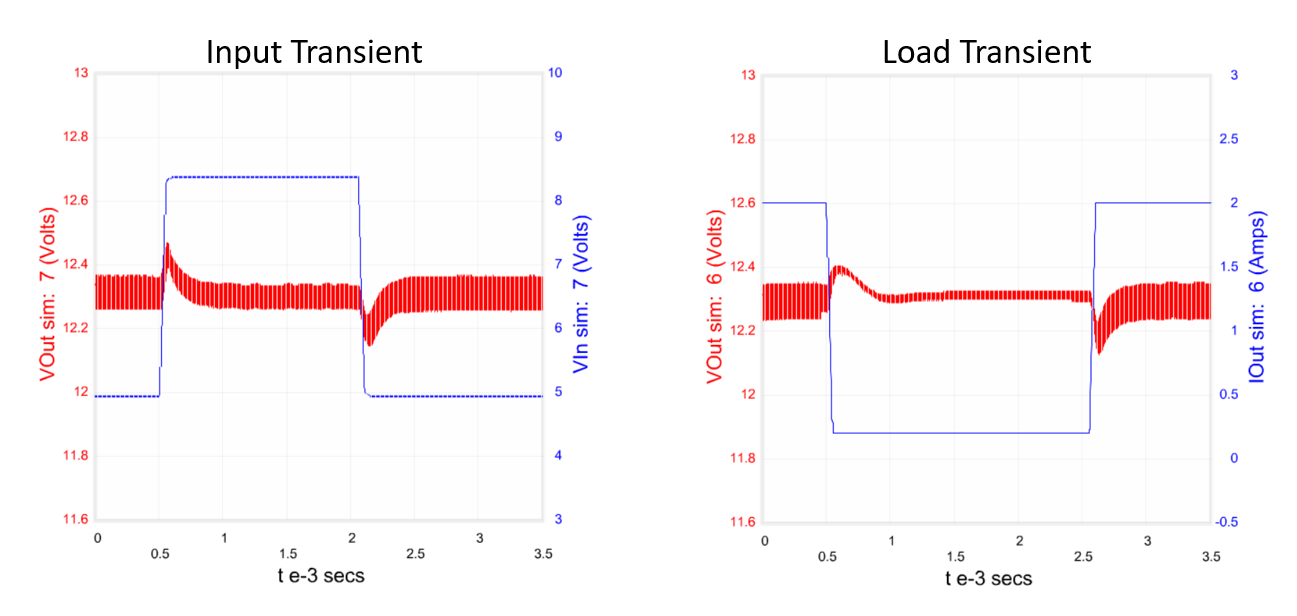


Fig 7. Converter simulation

Simulation results show that the boost converter is able to produce reliable 12V output within designed operating range.

* + - * MOSFET switch

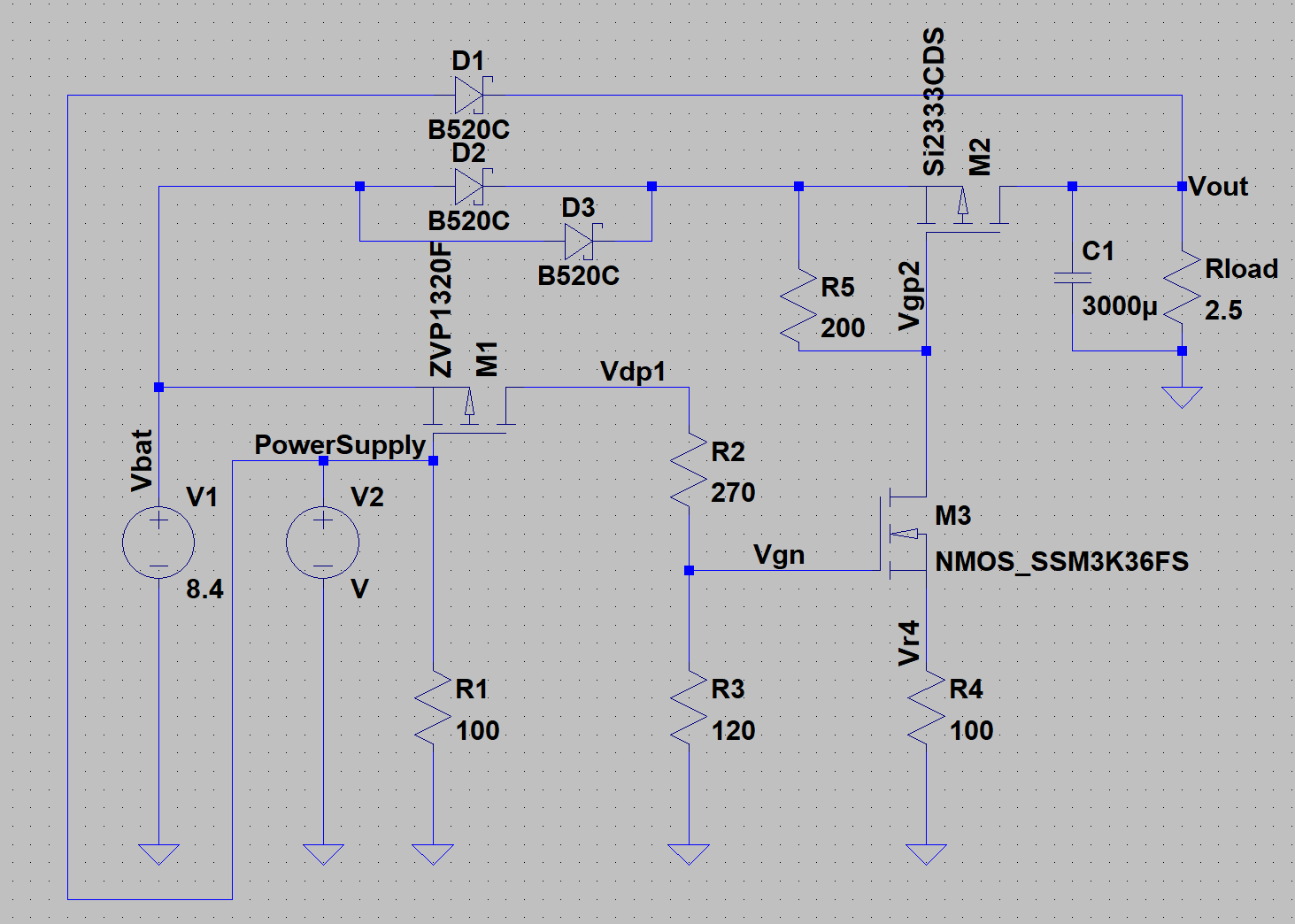


Fig 8. Switch schematic

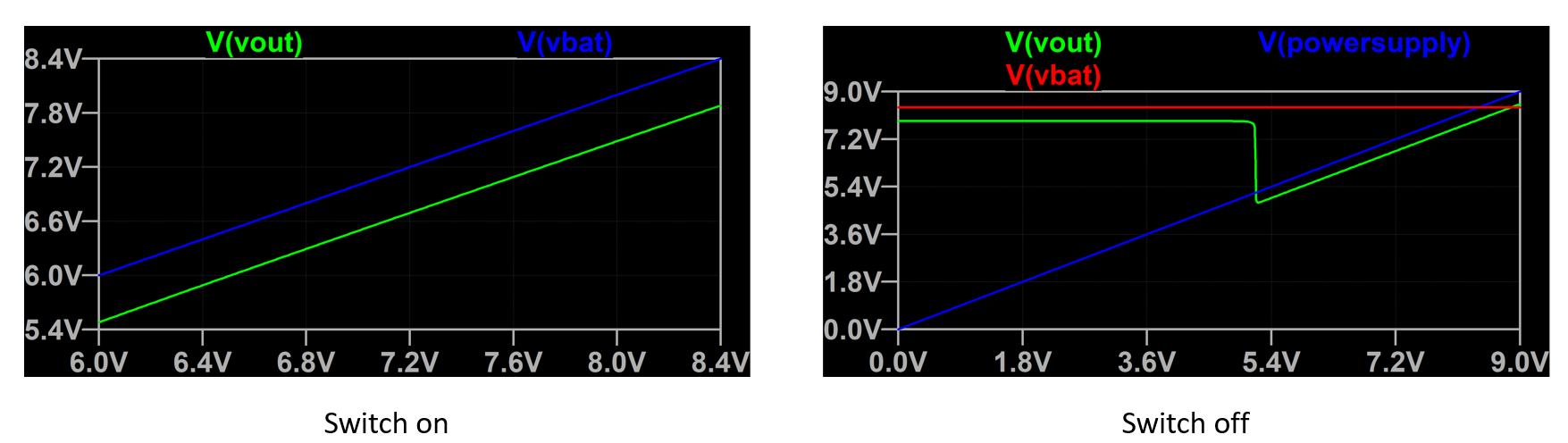


Fig 9. Switch simulation

On the left side, the switch is on so that current can flow from battery to device and the output curve follows battery voltage. On the right side, the switch is off so that battery is disconnected from device when input voltage is higher than about 5.3V. Since DC input is designed to be 9V, so as the diagram shows, output will not see battery and follow DC input. Operating point simulations show that the switch is at least 87% efficient and the standby power is 128mW.

* 1. Display

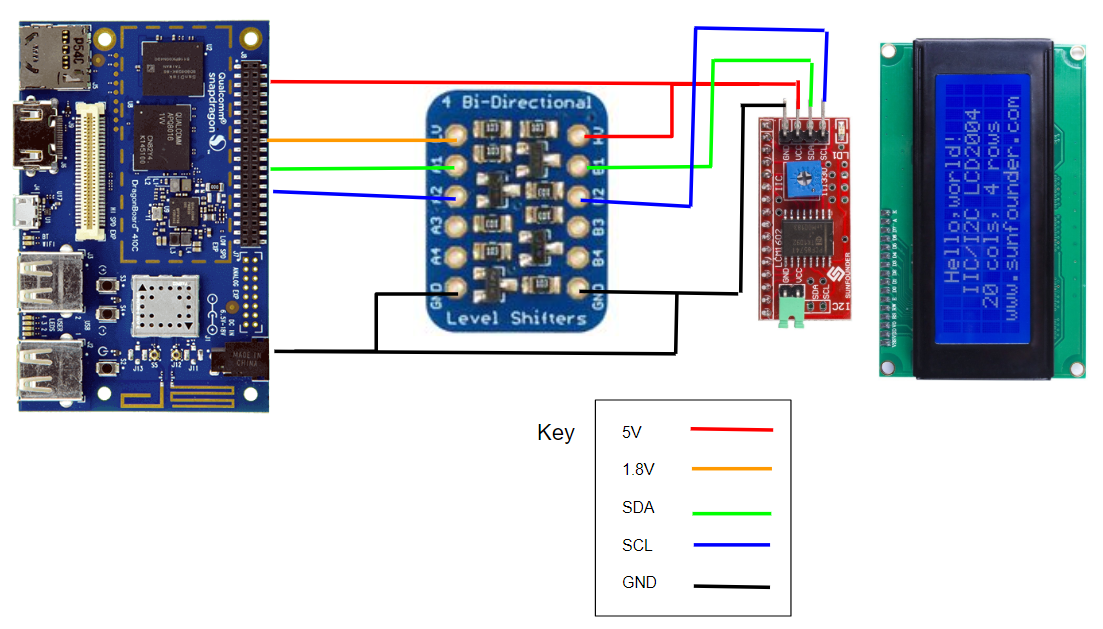


Fig 10. Wiring diagram for level shifter and LCD display

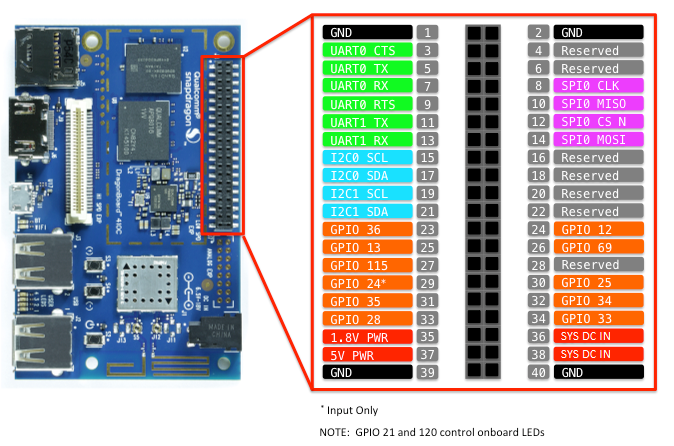


Fig 11. DragonBoard 410c low speed expansion pinout



Fig 12. Stock ticker running on Dragonboard w/ LCD display

* 1. Enclosure

Prototype 1:



Fig 13. Case prototype 1

Prototype 2:



Fig 14. Case bottom

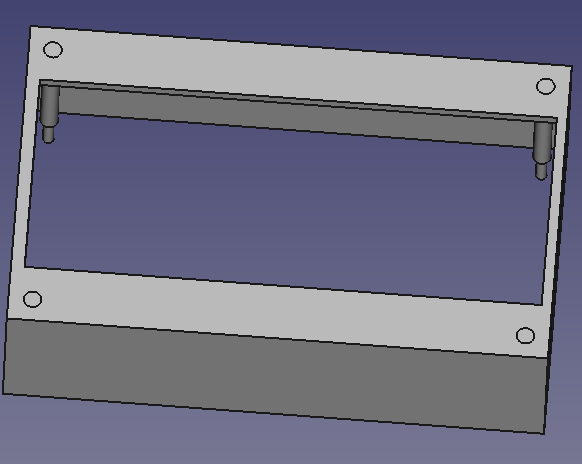


Fig 15. Case top

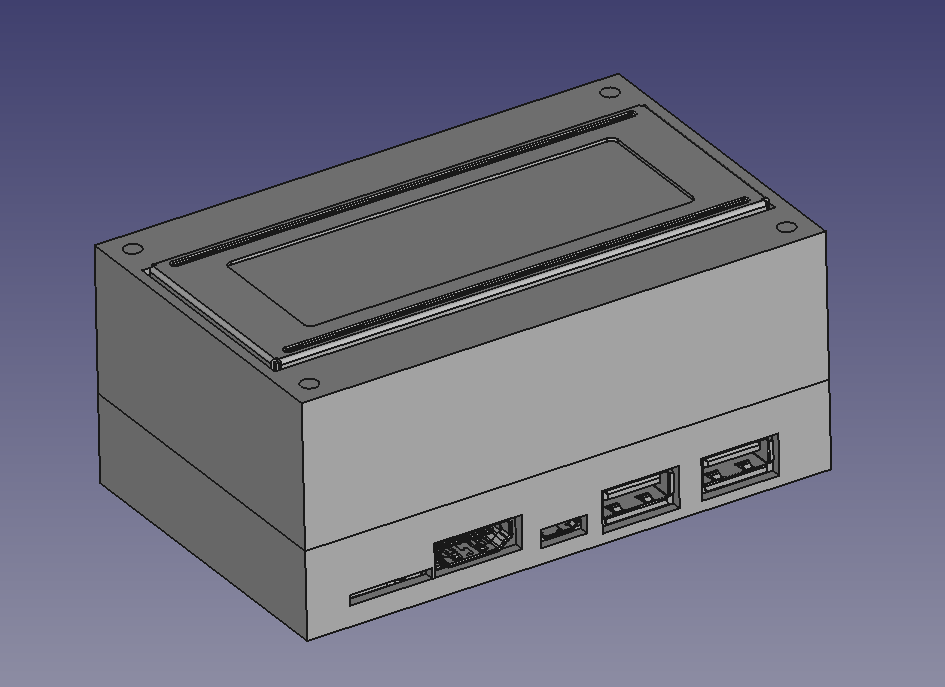


Fig 16. Case prototype 2

Final Design:



Fig 17. 3D printed case external

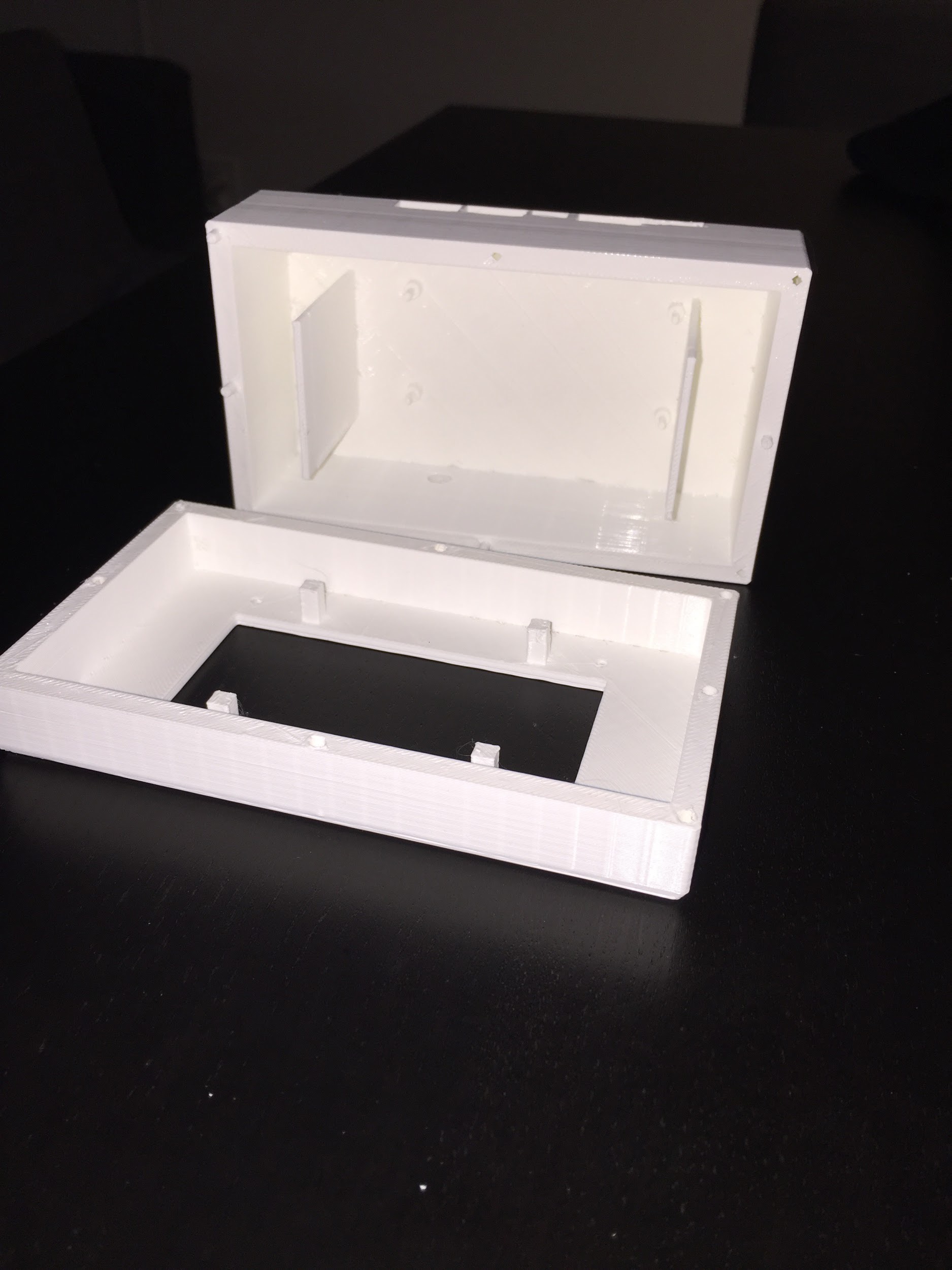
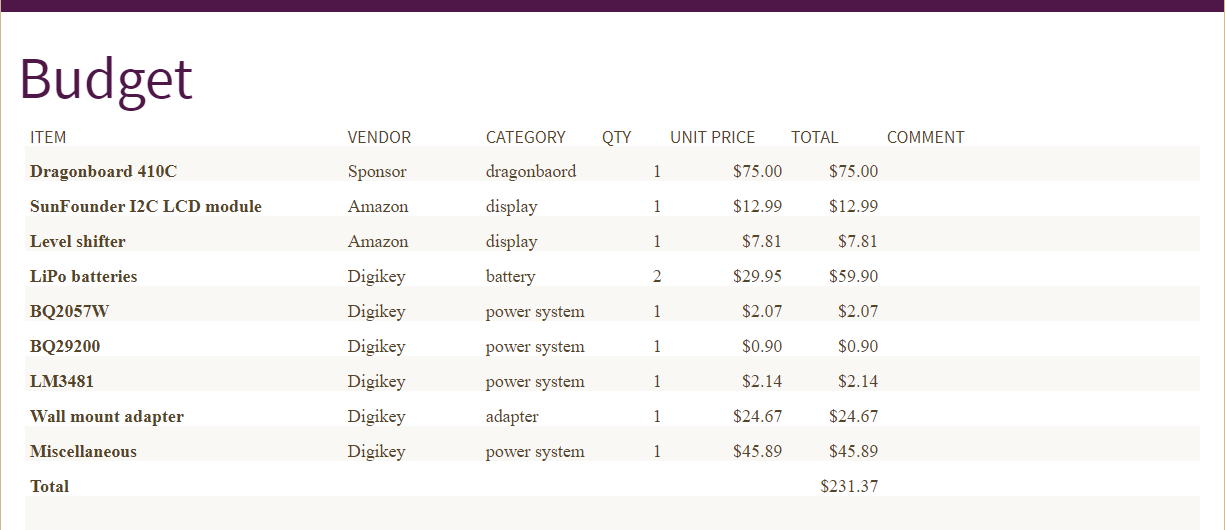


Fig 18. 3D printed case internal

1. Budget



1. Conclusion

In total, the functions implemented on the software side, include a functioning cryptocurrency wallet, capable of holding and exchanging SmartCash, a cryptocurrency, and interfacing with blockchain. With this core functionality completed, we then extended operations to a Cryptocurrency price conversion display for users to visualize the value of investments and a cold storage, offline, way to approve cryptocurrency transactions. These crypto functions achieve much of our original goal of the project but in the future work should be made in the hosting of a cryptocurrency nodes, and research will be pursued on mining functionality on an arm processor.

As for hardware, a LCD screen is successfully attached to the Dragonboard to display information including cryptocurrency conversion rates as well as the wallet balance of the user. The inclusion of these functions on a portable device means that the important information can be provided to the user at all times. To customize the conversion rates printed on the display, the user can edit the scripts display the conversion rates of their desired cryptocurrencies. The power system is able to support the device to run off of batteries and wall power. Charger and balancer are working as designed except the discharging current is a little lower than designed value. Possible reason could be due to breadboard setup which is not secure or accurate compare to PCB layout. Boost converter is at least working as designed under a light load. A PCB version of the power system is much prefered due to sensitivity of the circuit. Overall the system is performing well under normal condition, but does need more tests under other extreme conditions to ensure its functionality and stability.

The final product accomplishes many of the goals set forth by our mentor at the beginning of the quarter. We have managed to create a neatly packaged device that encmpasses all the hardware necessary to provide the services it promises to bring. An informative display shows all the relevant data keeping the user connected to their currency and the device. We also managed to accomplish stretch goals including the integrated mobile powersource and and power circuit. The entire system is integrated on an ARM device, as requested, allowing the device to run at lower power while generating less heat. As a whole the product is still in prototype phase but has come along nicely and meets the requirements of our mentor. Considering the nature of the desired product presented to us at the beginning of the quarter we ran into many hurdles on the way but having overcome them we have learned a great deal about the topics of Crypto Currency, Software Development, and more.

References

1. <https://github.com/SmartCash/electrum-smart>
2. <https://en.bitcoin.it/wiki/API_reference_(JSON-RPC)>
3. <http://docs.electrum.org/en/latest/coldstorage_cmdline.html?highlight=broadcast>
4. <https://github.com/96boards/documentation/tree/master/consumer/guides/mraa/i2c>
5. <https://www.96boards.org/documentation/consumer/dragonboard410c/hardware-docs/>
6. <http://www.righto.com/2014/02/bitcoins-hard-way-using-raw-bitcoin.html>

Appendix A - Hardware

* Power System
  + Battery - SparkFun electronics PRT-13856
  + Charger - BQ2057W
  + Battery protection with balancing - BQ29200
  + Boost converter - LM3481
  + MOSFET switch
* Display
  + 20x4 LCD display - SunFounder I2C Serial 2004 w/ LCM1602 controller
  + Level shifter - Adafruit ADA757
* Enclosure
  + 3D printed enclosure

Appendix B - Code samples

* Electrum Signing of USB

#!/bin/bash  
echo "Started Signing of Smartcash" > /home/linaro/SmartCash.log  
date >> /home/linaro/SmartCash.log  
i=20  
j=1  
while [ $i -gt 0 ]  
do  
 if [ -d /media/linaro/EB40-B1B7 ] && [ $i -lt 15 ]; then # Directory of USB  
 echo "found drive" >> /home/linaro/SmartCash.log  
 break  
 fi  
 echo $i  
 i=$(($i-$j))  
 sleep 1  
done  
for file in /media/linaro/EB40-B1B7/\*.txn  
do  
   
 cat $file | sudo -u linaro ./electrum-smart signtransaction - > "$file"".signed"  
 echo "Signed ""$file" >> /home/linaro/SmartCash.log  
done

* DEPENDENCY LIST

su root  
apt-get update  
apt-get upgrade  
apt-get dist-upgrade  
apt-get install sudo  
usermod -aG username sudo  
su username  
sudo apt-get install dkms build-essential #Install linux build dependencies for  
sudo apt-get install git  
cd ~/  
git clone https://github.com/smartcash/electrum-smartcash/  
sudo apt-get install python3-pip  
sudo apt-get install libusb1.0 libusb-dev libudev-dev g++  
pip3 install .[full] #For dragon board operation instead use the setup.py install method  
sudo apt-get install pyqt5-dev-tools  
pyrcc5 icons.qrc -o gui/qt/icons\_rc.py  
sudo apt-get install protobuf-compiler  
protoc --proto\_path=lib/ --python\_out=lib/ lib/paymentrequest.proto  
./electrum-smart create > secretseed.log  
cd ~  
git clone https://github.com/96boards/projects/crypto-node  
#Follow install instructions from the github