IOWA STATE UNIVERSITY
COMPUTER AND ELECTRICAL ENGINEERING DEPARTMENT

CPRE EE 492 TEAM 25

DISTRIBUTED SNIFFER NODES FOR BATTERYLESS SENSOR NODES

DESIGN DOCUMENT

April 28, 2024

Client/Advisor- Henry Duwe

Team/Software Lead- Thomas Gaul
Scribe/Software Member- Ian Hollingworth
Software Member- Spencer Sutton
Hardware Lead- Tori Kittleson
Hardware Member- Matthew Crabb

sdmay24-25@iastate.edu
https://sdmay24-25.sd.ece.iastate.edu

Executive Summary

Problem Statement

We aim to create an open-source testbench for testing and benchmarking wireless batteryless sensor nodes that have been designed by our clients. This test bench needs to be made of modular units to allow test scalability. The units must interface with the wireless nodes to collect, record, and store information about their operation and performance.

Terminology

BOB Node: Batteryless sensor node device(s) under test

Sniffer Node: Single node of testbed

Sink Node: Central Node in network

Host: Aggregates data through the Sink node for researchers use

cc1352: radio microcontroller using in Sink and Sniffer Nodes

Msp-430: low power microcontroller used in BOB nodes

Development Standards & Practices Used

C: Microcontroller code written in standard C with standard coding practices such as commenting and library use

Python: Host computer programming written in standard C with standard coding practices

UART: Standard UART serial communication between Sync Node and Host Computer allows seamless integration to any computer (RS-232)

SPI: Communication between the various portions of the BOB Node

KiCad: Standard open-source CAD software for designing PCBs

GitLab: Industry standard version control software

TIRTOS: Texas Instruments provided a Real-Time Operating System used to handle tasks and threading within code **Low Energy Bluetooth** Bluetooth 5.2 Low Energy (IEEE 802.15.1)

Hardware Design: Consistent with TI product line

Summary of Requirements

Modularity: Start with 9 nodes, ability to scale to 1000

Simulation: Sniffers must be able to provide faux sensor data to BOB Nodes for testing

Low Cost: Low cost for Sniffer PCB

Electrical Isolation: Connecting Sniffer to BOB node causes no measurable change in BOB behavior

Open Source: Build to be well-documented, reproducible, and expandable for future groups to use and build upon

Applicable Courses from Iowa State University Curriculum

EE 201/230: Circuits I / II

EE 285: Intro to C

EE 330: Intro to VLSI

EE 465: Digital VLSI

EE 333: PCB design

EE 224/324: Signals and Systems I / II

EE 321: Communication Systems I

CPRE/EE 185: Intro to EE/CPRE

CPRE 281/288: Digital Logic & Embedded Systems

CPRE 381: Computer Structure and Assembly

CPRE 558: Real Time Systems

New Skills/Knowledge acquired that was not taught in courses

Technological Skills

- PCB design
- Soldering
- Wireless Transmission Protocols
- Python
- KiCad
- System-level Debug

Soft Skills

- Team Management
- Team Communication
- Client Negotiation
- Meeting management

Contents

1 TEAM

1.1 Team Members

Matthew Crabb, Thomas Gaul, Ian Hollingworth, Tori Kittleson, Spencer Sutton

1.2 Required Skill Sets for Your Project

Embedded Systems

PCB Design

Electrical Systems

Radio Communication Protocol

Soldering

Debugging

1.3 Skill Sets covered by the Team

Embedded Systems- All members

PCB Design- Matt, Tori, Ian, Thomas

Electrical Systems- Thomas, Matt, Tori

Soldering- All members

1.4 Project Management Style Adopted by the team

We have committed to an agile management style with goals for each week which will be assigned to each team member. Daily scrum meetings will be in the form of consistently

messaging and two to three meetings a week. We will use GitLab to for gooals and milestones.

1.5 Initial Project Management Roles

Thomas- Team lead, Technical Software

Matt- Technical hardware

Tori- Hardware lead, Technical hardware

Ian- Scribe, Technical software

Spencer- Technical software

2 INTRODUCTION

2.1 Problem Statement

We aim to create an open-source testbench for testing and benchmarking wireless batteryless sensor nodes that have been designed by our clients. This testbench needs to be made of modular units to allow test scalability. The units must interface with the wireless nodes to collect, record, and store information about their operation and performance.

2.2 Requirements & Constraints

2.2.1 Requirements

- Provide scalability in design in Sniffer code and communication and hardware for BOB and Sniffer (1 Sniffer per BOB). Goal: 9 nodes, 1000 nodes design analysis.
- Mechanical durability of BOB and Sniffer interface and stand-alone unit for testing use.
- Implement a system that is easy to move for testing.

- Set pin requirements that map to each other.
- Implement hardware for wireless mass reprogramming of BOB.
- Provide thorough documentation on the project.
- Include make file for all project documents.
- MSP430 should have a power switch for the CC1352 so that it is not always on. Need to prototype this and ensure it meets the electrical characteristics/requirements.
- Do not have a modification of BOB code for Sniffer implementation.
- Sniffers communicate with each other to pass log data from BOBs to the Host and compare data from BOB.
- Have a Host system that organizes and stores logs from Sniffers.
- Sniffers record both wireless data and wired. Wired can optionally be disconnected.

2.2.2 Constraints

- Use CC1352 microcontrollers in our Sniffer design
- Our Sniffer network must record all transmission from BOBs at a maximum rate of 1 8-byte message every 5ms.
- Sniffers only cause trivial effects on BOB's performance (Sniffers are electrically isolated from each other and have minimal effect on BOB up time).
- Have 9 Sniffer/BOB pairs fully constructed by the end of the year.
- Central Sniffer Node that cannot lose power and collects data from other Sniffer nodes. The central Sniffer Node can be (statically) selected from any Sniffer Node.
- Sniffer to Sniffer communication does not interfere with BOB communication.
- 20 Dollars or less per Sniffer node.
- Wireless Sniffer Nodes must have the ability to operate continuously for a week.

2.3 Engineering Standards

Software

- UART communication protocol: Standard way for microcontrollers to communicate with PC's, will be used to connect our Sink node to our Host PC. RS232
- Radio communication standards such as , Bluetooth 5.2 Low Energy (IEEE 802.15.1), RS-232, Proprietary 2.4 GHz
- We need these communication standards to communicate between multiple sniffer nodes sending data back to the Host

Hardware

- Standard pin header for connecting between boards
- Revision numbers
- Version control for PCB design
- Standard circuit schematic design
- Effective organization of component library

2.4 Intended Users and Uses

Our clients are Professor Duwe and his graduate assistants, with secondary users being other researchers who might want to use our testing infrastructure for their own work.

Our clients will be using this testbench to test their sensor nodes and write research papers about them, so they need it to be as versatile as possible.

Our design and software will all be open-source for other researchers and designers. They may build upon and expand our designs, so it will all need to be as modular and well-documented as possible.

3 DESIGN

3.1 Design Content

The design content can be split into four main components: Sniffer design, Host design, BOB updates, and system integration.

The main task of this project is to design a set of Sniffer Nodes, which will provide a testbed for the BOB Nodes by accurately recording data on BOB operation and by emulating an environment in which the sensor resides. Both hardware and software systems must be designed and integrated for the Sniffer Nodes.

The Sniffer hardware tasks consist of designing a PCB that contains an MCU, a wireless communication interface and an antenna, headers to connect to other boards, and a power supply system. The wireless communication system will be designed to pick up BOB communications. The headers must be carefully designed to provide all necessary monitoring without interfering with the BOB operation. The power system contains design decisions on how to power the Sniffer with a battery as well as how the power supply can be switched to a continuous power supply.

The Sniffer software will contain design decisions mainly regarding communication protocols and data aggregation and sorting. The Sniffer software will be configured to work on the MCU. The Sniffer will have to be designed to communicate with 3+ MCUs from other boards simultaneously which will require intensive software design. The Sniffers will also have to move data to the Host Node in some fashion. This will require large amounts of data to be sorted and transferred in an efficient manner. This will also present difficult design challenges.

The testbed also requires a Host Node. The Host Node will have a hardware system that it operates on and software that collects and stores data in a form that is easy for researchers to access and manipulate. Design decisions must be made on what hardware system to use and how it will communicate with and receive data from the Sniffers. A software system will also need to be designed which provides accurate, easy to access data.

In addition to designing the testbed, Dr. Duwe's research group has asked us to simplify their

current BOB design and add some new functionality to it. This task includes making design decisions on how best to simplify the board as well as designing new circuitry.

All systems mentioned above will need to be designed in such a way that they work together functionally as a whole. The hardware requires designs that can interface with each other and easily be connected. The software will require communication protocols which work with one another. The Sniffers and the BOB nodes will need to communicate but also maintain electrical isolation from each other. This leads to a situation where every design decision for each part also affects the whole and must be carefully considered.

Lastly, the systems must also mechanically interface with each other and be able to be easily set up for tests. Dr. Duwe's group is hoping to hang each BOB Node and Sniffer pair from the ceiling together. This will require design on the hardware side as well as the design of mechanical hangers.

Overall, this project contains a vast and varied amount of design content.

3.2 Design Complexity

Our project has a variety of aspects contributing to making it a complicated and challenging project for the capstone course.

To meet the requirements for Senior Design, our project must be complex. There are two main criteria which determine complexity. The first criterion asks if the design has multiple elements that each require different scientific, mathematical, and engineering principles. The second criterion asks if the design task requires solving challenging problems which have not been solved, or have not had industry standard solutions adopted yet.

Our project satisfies the first criterion but does not address problems modern enough to meet the second criterion. Our project requires the design of multiple subsystems which must work together. The main subsystems we will design are:

1. A Sniffer Node PCB
2. A simplified BOB Node PCB

3. A common header pinout which allows all PCBs (4 total) to be stacked and to communicate without interfering with each others operation
4. Software for the Host Node to recieve and organize data
5. A mechanical mounting system to hold all 4 boards together and hang them from the ceiling
6. Software for the Sniffer Nodes to listen and send communications
7. A communication scheme to route all packets from multiple Sniffer Nodes to the Host Node

Designing these subsystems will require the use of a number of mathematical and engineering principles. The hardware design for the Sniffer Node and simplified BOB node require schematic design and capture, calculations to determine system needs accurately, soldering, RF PCB layout, reverse engineering of provided reference designs, usage of test equipment, and hardware debugging. Designing the software for the Sniffer Nodes requires C programming, Python scripting, usage of multiple different communication protocols, coding using operating systems, writing software tests, and debugging software. Designing the communication scheme requires using math to plan out the order of events and usage of communication protocols. Designing the mechanical mounting system will require 3D modeling, usage of workshop tools for fabrication, and potentially 3D printing.

The project contains much complexity in the number of subsystems to design and the various engineering skills required to design each subsystem. Not directly relating to the criteria, there will also be a vast amount of complexity in implementing ten Sniffer Nodes and a Host Node which all seamlessly work together. This will require careful planning, time management, and debugging skills.

3.3 Modern Engineering Tools

We will require a handful of engineering tools to accomplish our goals and project

Code Composer Studio(CCS)- We use CCS to code and program in C for all the microcontrollers within our project.

Visual Studio Code (VS Code)- We use VS Code Python to write and run the program for collecting and compiling the data from tests.

KiCad- KiCad is used for all aspects of our PCB design. It is used for creating schematics and footprints in our parts library and the design schematic and layout for our PCBs (Printed Circuit Board)

AutoDesk Inventor- We will use Inventor (or similar CAD software) to model any of the physical components required by our project.

GitLab- We use GitLab for doing version control and project management tools

3.4 Design Context

Our project is used in the context of research by our clients, who are part of a research team led by Professor Duwe. One of our constraints is to make all aspects of our design available as open source. With open source, other batteryless sensor enthusiasts may choose to use or update our testbed design to fit with their own project needs. This research project is designed to be able to develop systems that could be applied in a wide variety of fields such as manufacturing, farming, infrastructure, and more.

Given the broad scope of this project, it creates a lot of unique constraints and requirements. We must make our design flexible to fit a wide variety of testing environments, particularly with system communication. Along with this comes documentation, and we would like to document our work clear and concisely so that someone with limited experience in this field can quickly comprehend and implement the work themselves.

Area	Description	Example
Public Health, Safety, and Welfare	During the testing process, radio signals will be sent between various MCUs on both BOB and Sniffer nodes. Additionally, the solder and batteries can have a negative effect on health	We will work to avoid interference with other people's devices during the testing process. Additionally, in the lab setting, we can contain the health effects of solder and batteries by handling them in the prescribed safe procedure, including end of life.
Global Cultural and Social	A BOB network can be used in professional and workplace settings to increase overall work efficiency.	In workplaces where collecting sensor data is important, a BOB network will automate the process, saving time for people in the workplace.
Environmental	The batteries and solder in our project can harm the environment	It will be essential to dispose of Sniffer nodes in an environmentally friendly way after they are no longer of use; the lab and University will handle proper disposal at end of life.
Economic	After the testing is complete and BOB nodes are distributed on a large scale in a practical setting, the BOB nodes can be used to reduce the cost of monitoring processes for the user. For example, a network of BOB nodes could be distributed around a factory to monitor temperature, pressure, or other relevant information to identify if everything is running accordingly.	The data from the BOB network could help improve factory cost efficiency or could be used as a safety mechanism to turn off certain processes if things go wrong. In any case, increased monitoring capabilities in many cases will reduce the burden of cost on the BOB network user.

Table 1: Project Context

3.5 Prior Work/Solutions

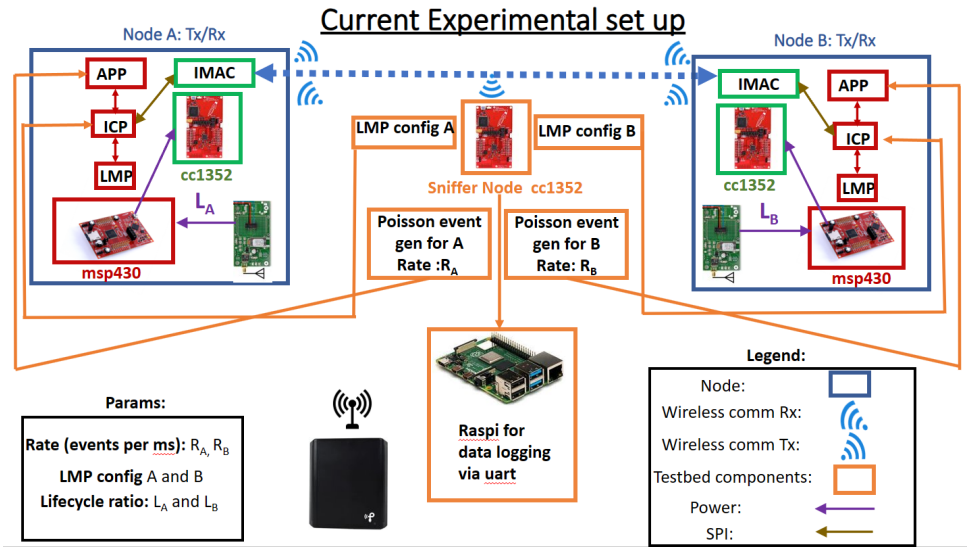


Figure 1: The current setup design the Research Assistants use

Figure 3 shows a testing setup used as a prior solution by the research team to run tests on two BOB nodes with a single Sniffer. This test works and is currently being used but we have been tasked with expanding it. Our goal is to create a setup that has each Sniffer paired with a single BOB as opposed to the current system. Additionally, we want all the communication back to the Sink node to be conducted wirelessly. The other goal is to make the wiring cleaner and completely remove the need for jumper wires. Additionally, we will be making the stack of boards more robust mechanically, resolving that challenging aspect of debugging.

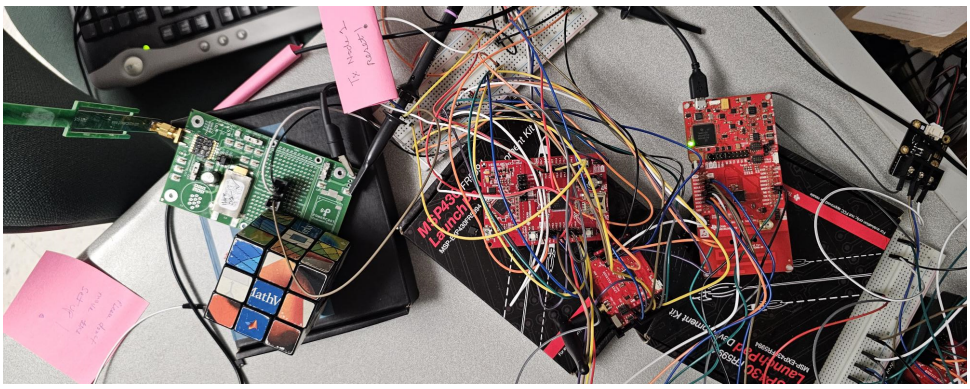


Figure 2: The current setup the Research Assistants use

3.6 Design Decisions

We decided to do communication between the Sniffer nodes with the CC1352 radio and not with an ESP8266 because of power constraints.

BOB communication will take place on a sub-1Ghz band with a custom communication protocol developed by the research team. The Sniffer communication will use 2.4Ghz Bluetooth communication.

We decided to use a load switch to control power between the MSP430 and the CC1352 BOB.

We decided that the boards would be stacked from bottom to top as Powerboard, MSP430(simplified), CC1352(radio board), and Sniffer.

3.7 Design

3.7.1 Design Visual and Description

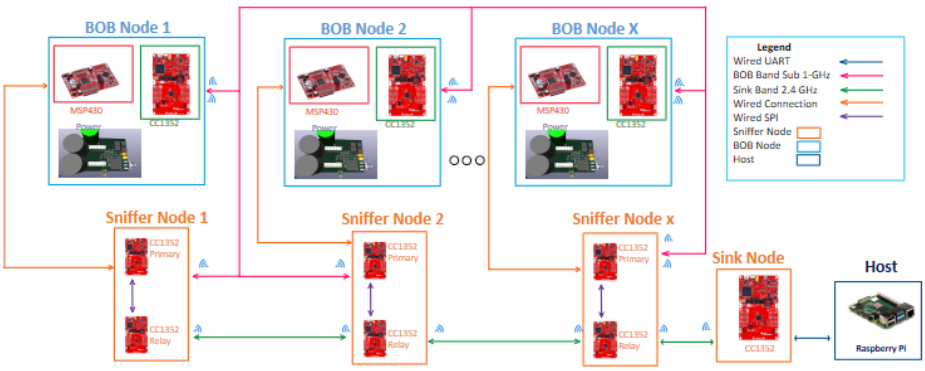


Figure 3: Our final design plan

Our design involves having the Sniffer nodes harvest all their data independently, then passing them through a predetermined route of Sniffers until the data ultimately reaches a Sink node. This Sink node would be the “test Host” of all Sniffers. We will have two CC1352s in each Sniffer to both listen and communicate effectively at the same time. The data will pass through the network and reach the Sink node, and be communicated to the Host via UART for data logging and processing. Conversely, to set up tests, the Host communicates

the setup to the Sink Sniffer via UART, and the Sink node will create a ripple distribution throughout the network until every board has received the setup and implemented it.

We took this design as it was suggested by our advisor, and it will require less power than design 0 described in Appendix II.

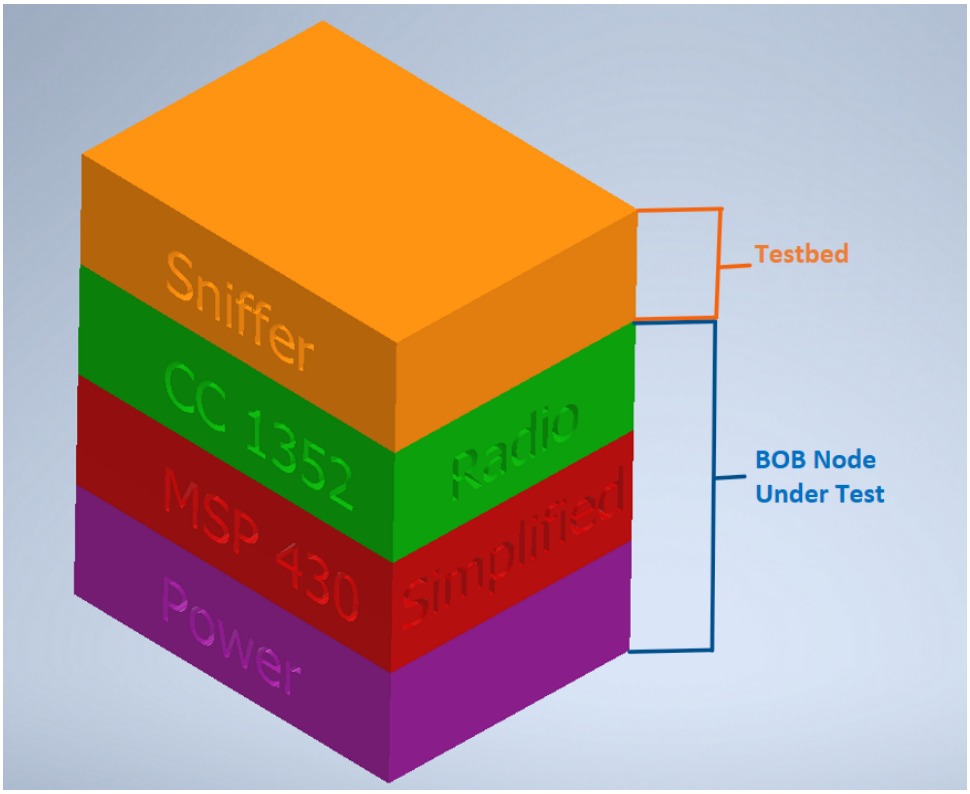


Figure 4: Our current hardware stackup plan

The above diagram gives a model for what our PCB stack up will look like.

The current design involves a 4-board stack. The bottom most board is the power harvester board. This will harvest the RF energy and supply power to the entirety of the stack. The current power harvester board is an off the shelf evaluation board. One of Dr. Duwe's graduate students, Vishak, will be designing an in-house board to replace the current revision. This will allow for more flexibility with the connector stack-up, as far as modification of connector pinout goes.

The next board in the stack is the MSP430. This board is currently an evaluation board and is being simplified down by our senior design team to condense the design and keep only the used functionality of the board. This is included as part of the BOB Node. This board is

placed between the MSP430 and the CC1352 as both boards will contain radios and there are concerns of interference between the two. The two radios that connect via SMA connector will be placed on opposite sides of the board stack to account for this.

The CC1352 or “radio board” is used for communications both between each node and to the Host, through the Sniffer testing board.

The topmost board is the Sniffer board which is the primary focus of our senior design project. The Sniffer will be probing into the main connector of the node and evaluating the performance of the BOB Node. The Sniffer will only be used for initial prototyping and implementation. When the product is distributed and functioning beyond these steps, the Sniffer board will not be included in the unit. This is why the board is placed at the top of the stack as it can easily be removed once the unit is ready to be used.

3.7.2 Functionality

One of our main considerations for our current proposed solution is deciding how to allow each of the nodes connect to both BOB and Sniffer communication bands without losing any information. The idea we landed on was having two CC1352s for each sniffer node one monitoring the BOB band (Sub 1 GHz band) and one communicating on the Sniffer band (2.4GHz band).

Another main consideration of the Sniffer Node design is how it will be powered. The requirements for the power system is that it is easy to use, can power the Sniffer for up to 7 days continuously, and that the Sniffer node can optionally be powered by a wire to ensure that it does not fail. To meet these requirements, our design is to use a battery system to power the Sniffer.

There are a number of considerations to take into account for the design of the battery system. The first is the power, voltage, and current needs of the Sniffer board. The main element on the sniffer board is the CC1352R processor. According to the datasheet, the CC1352R has its own internal regulators. This negates the need for a regulator on the chip. The CC1352R requires a supply voltage between 1.8 – 3.8V. According to measurements made in the datasheet, the maximum operating current of the chip is about 25mA. Our requirement will be that the battery system can supply 35mA to give plenty of margin. The battery system must meet all CC1352R power needs and have the capacity necessary to run the Sniffer for 7

days continuously (this requirement is estimated below).

The second consideration is cost of the system. One of the Sniffer Node requirements is to have a low cost. The client desires that the cost is low to enable more nodes to be tested on limited research funds and so that other groups can cost-effectively begin using our test system. The cost of the system includes batteries, ICs required to run them, mounting solutions, and potentially recharging systems. The goal is to reduce the cost per Sniffer. The cost also includes the cost of future use (for example if batteries must be replaced).

The third consideration is ease of use and practicality. The system should be easy for the graduate students to set up and use. For example, the system would not be easy to use if the batteries were difficult to put in and take out or if the batteries took an inordinate amount of time to charge.

The fourth consideration is sustainability. Disposal of batteries can have negative environmental effects. If possible, the environmental impact of our design should be minimized.

The next sections cover an estimation of the worst-case battery capacity requirement and several of the design ideas we generated. These will be reviewed with the client to determine the most optimal design.

3.7.3 Design of Pinouts for the Stack

The PCBs must connect in a simple, mechanically sound way. Mechanically sound means that they all stay together and that the connections between them are secure. The solution chosen to complete this task is to put all the PCBs together in one stack connected by identical headers. All TI development boards use an identically spaced header with a standardized pinout. This is shown for the CC135R development board below.

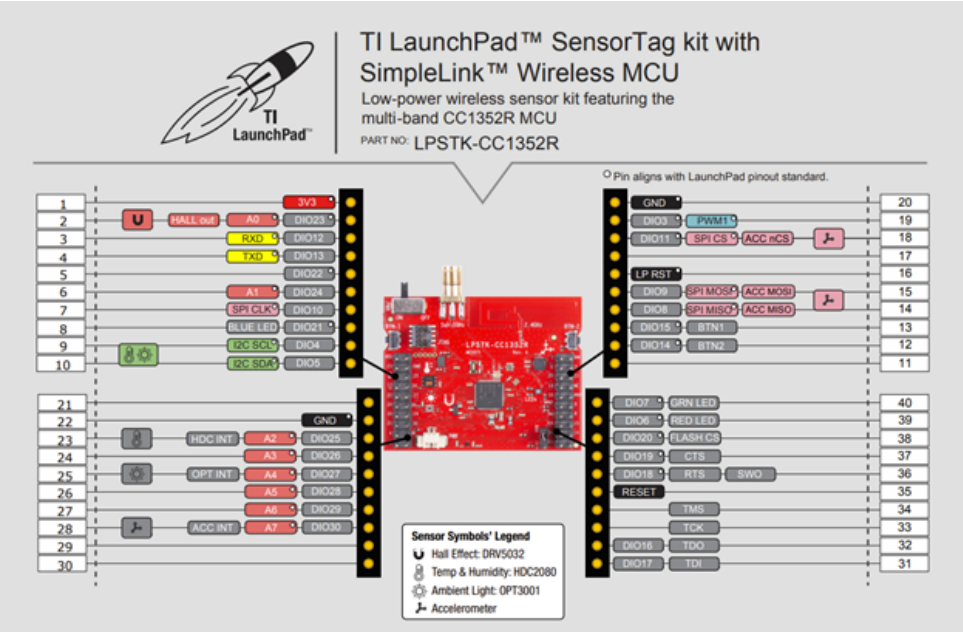
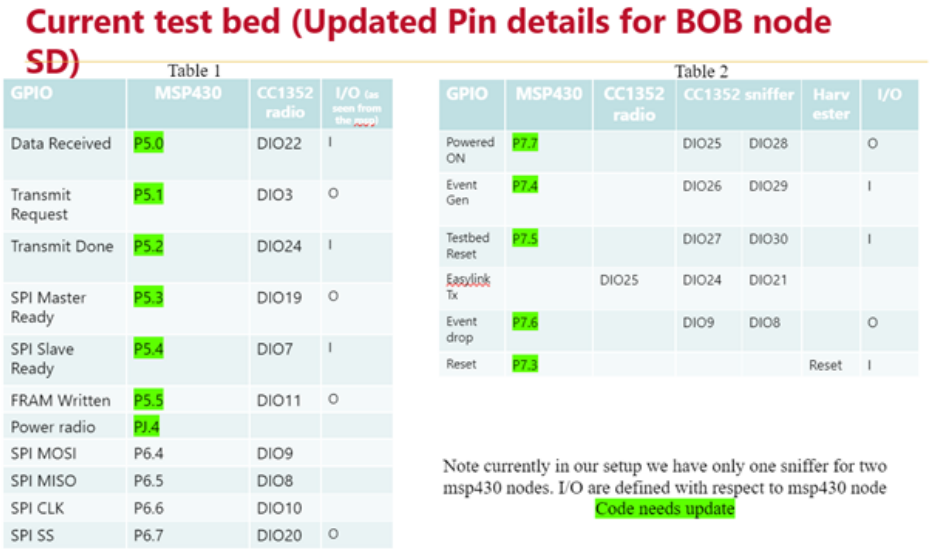


Figure 5: TI CC1352R Development Board Quick Start Guide [1]

The plan is to use the friction of the header connections to hold the stack together. However, this will likely be supplemented by additional mechanical support such as non-conducting bolts through mounting holes in the boards or a 3d printed frame.

To enable this solution, the pinout must be carefully designed to ensure the pins of all four PCBs are properly connected. The middle boards must also have a small footprint due to the antennas and capacitors sticking up or down from the radio board, Sniffer board, and harvester board. This is one of the main motivations for designing a new MSP430 board. Additionally, the Sniffer pinout and Harvester pinout needed to be made compatible with the stack. The research team gave us the details of all connections needed and we designed the pinouts for all boards.

The pinout design centered around the existing pinout of the CC1352R development board because it cannot be altered. There are a number of connections that must be made between only the MSP430 board, Harvester board, and Sniffer board and not the CC1352R board. To make these connections, the no connect pins on the CC1352R were used. However, there were not enough to meet the requirements. To address this problem, fourteen pin headers will be used as offsets to the CC1352R development board. This creates more no connects which allows the pinouts to meet the requirements as shown below.



Current test bed (Updated Pin details for BOB node SD)

Table 1

GPIO	MSP430	CC1352 radio	I/O (as seen from the ...)
Data Received	P5.0	DIO22	I
Transmit Request	P5.1	DIO3	O
Transmit Done	P5.2	DIO24	I
SPI Master Ready	P5.3	DIO19	O
SPI Slave Ready	P5.4	DIO7	I
FRAM Written	P5.5	DIO11	O
Power radio	PJ.4		
SPI MOSI	P6.4	DIO9	
SPI MISO	P6.5	DIO8	
SPI CLK	P6.6	DIO10	
SPI SS	P6.7	DIO20	O

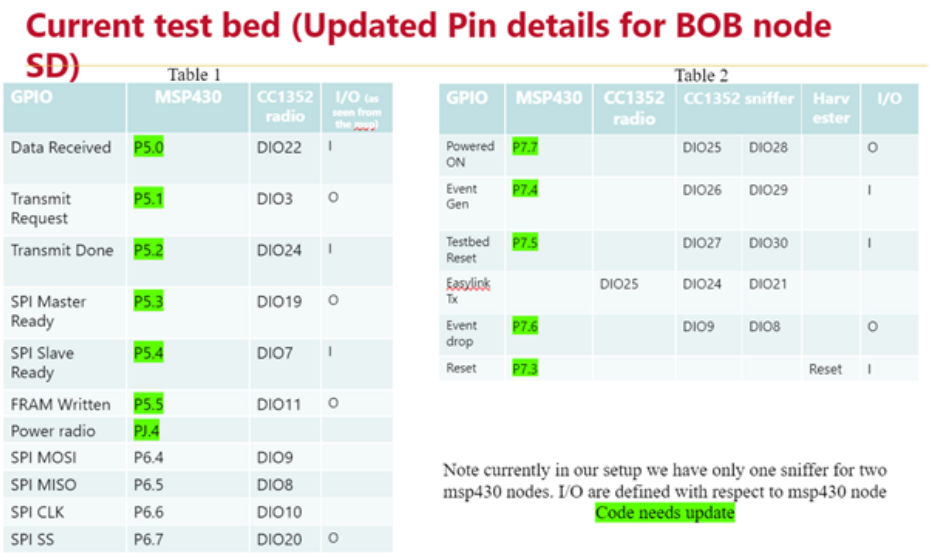
Table 2

GPIO	MSP430	CC1352 radio	CC1352 sniffer		Harvester	I/O
Powered ON	P7.7		DIO25	DIO28		O
Event Gen	P7.4		DIO26	DIO29		I
Testbed Reset	P7.5		DIO27	DIO30		I
Easylink Tx		DIO25	DIO24	DIO21		
Event drop	P7.6		DIO9	DIO8		O
Reset	P7.3				Reset	I

Note currently in our setup we have only one sniffer for two msp430 nodes. I/O are defined with respect to msp430 node
Code needs update

Figure 6: Plan to Create Extra NC Pins on the CC1352R Development Board

The pin connection requirements provided by the research team are shown in the following figure.



Current test bed (Updated Pin details for BOB node SD)

Table 1

GPIO	MSP430	CC1352 radio	I/O (as seen from the ...)
Data Received	P5.0	DIO22	I
Transmit Request	P5.1	DIO3	O
Transmit Done	P5.2	DIO24	I
SPI Master Ready	P5.3	DIO19	O
SPI Slave Ready	P5.4	DIO7	I
FRAM Written	P5.5	DIO11	O
Power radio	PJ.4		
SPI MOSI	P6.4	DIO9	
SPI MISO	P6.5	DIO8	
SPI CLK	P6.6	DIO10	
SPI SS	P6.7	DIO20	O

Table 2

GPIO	MSP430	CC1352 radio	CC1352 sniffer		Harvester	I/O
Powered ON	P7.7		DIO25	DIO28		O
Event Gen	P7.4		DIO26	DIO29		I
Testbed Reset	P7.5		DIO27	DIO30		I
Easylink Tx		DIO25	DIO24	DIO21		
Event drop	P7.6		DIO9	DIO8		O
Reset	P7.3				Reset	I

Note currently in our setup we have only one sniffer for two msp430 nodes. I/O are defined with respect to msp430 node
Code needs update

Figure 7: Pinout Requirements Provided by Research Team

The final pinouts settled on for the harvester board and the MSP430 Simplified board are shown below. The extra pins that did not have specific functionality are connected to GPIO pins on the MSP430, Simplified with jumper resistors in between. This will allow the research

team to use these pins in the future for any functionality that may be desired. This effectively makes the whole stack much more configurable, enabling easier development.

The Sniffer pinout has not been fully completed yet, as the Host to Sink communication is still being determined.

MSP Board Pinout

Pin #	Func	Pin #	Func	Pin #	Func	Pin #	Func
1	3V3 to CC	21	3V3	40	P5.4	20	GND
2	GPIO	22	GND	39	GPIO	19	P5.1
3	GPIO	23	NC	38	P6.7	18	P5.5
4	GPIO	24	GPIO	37	P3.5	17	GPIO/EN
5	P5.0	25	GPIO	36	GPIO	16	NC
6	P5.2	26	GPIO	35	GPIO	15	P6.4
7	P6.6 (SPI)	27	GPIO	34	RST_MSP	14	P6.5
8	P1.0	28	P7.3	33	P1.1	13	P1.6
9	P7.4	29	P7.5	32	P1.7	12	P2.6
10	P7.6	30	P7.7	31	P2.5	11	GPIO

Figure 8: MSP Simplified Pinout

Harvester Board Pinout

Pin #	Func	Pin #	Func	Pin #	Func	Pin #	Func
1	NC	21	3V3	40	P5.4	20	GND
2		22	GND	39		19	P5.1
3		23	NC	38	P6.7	18	P5.5
4		24		37	P3.5	17	
5	P5.0	25		36		16	NC
6	P5.2	26		35		15	P6.4
7	P6.6	27		34		14	P6.5
8	P1.0	28	P7.3	33	P1.1	13	P1.6
9	P7.4	29	P7.5	32	P1.7	12	P2.6
10	P7.6	30	P7.7	31	P2.5	11	

Figure 9: Power Harvester Pinout

3.7.4 Breakout Board Design

The Breakout Board 64 was designed to test out some of the modifications that we would be making to the MSP Simplified board. This way we could order and test the Breakout Board which can be manufactured for a much cheaper cost. The Breakout Board is a pretty simple PCB that takes the form factor of the TI MSP430 Board 65 , and includes the following: 20P Boosterpack connectors, load switch circuitry, a 2 header pin for current measurements. Mostly, we wanted to ensure that the load switch would work for our needs, the connector

spacing would match the stack up, and overall that fabrication of our board designed in KiCAD, and ordered through JLCPCB, would work as we expected. The breakdown of the estimated per board can be found here 68. It should be noted that this is only an estimate based on the components and boards that we ordered. This however does not include the cost of shipping for the components, as they were ordered through the ETG in Coover. The Breakout Board was tested and everything works as we expected, so we moved onto the MSP Simplified design. The Breakout Board is currently being used by one of the graduate researchers.

3.7.5 MSP Simplified Design

The MSP Simplified 69 was designed to replace the current development board that is being used as part of the BOB Node. The researchers wanted to minimize the board so that would fit better in our 4 layer stack-up, as well as get rid of any unused circuitry. The development board that was used is TI's MSP430fr5994. This board acts as the controller in the BOB Node setup. The MSP430 board includes an MCU, and communicates with the CC1352, controls power distribution from the Power board, and will be be monitored by the Sniffer Node that our team fully develops.

The MSP Simplified Removed the following logic from the MSP430 design: The debugger logic (we will attach separately to debugger when programming), additional LEDs, unused switches, a super capacitor, SD card, and external power connectors. The additions to the board include: change of form factor to roughly match that of the CC1352 radio board (it is actually a bit longer, as we included mounting holes), load switch circuitry to control power to the CC1352 board, mounting holes, a 90 degree debugger pin, a 2 pin connector on the power rail to potentially adjust capacitance, and DNP resistors for unused GPIOs (flexible for future use). The breakdown of the estimated cost per board can be found here: 76. It should be noted that this is only an estimate based on the components and boards that we ordered. This however does not include the cost of shipping for the components, as they were ordered through the ETG in Coover. The first revision of the board has been soldered and tested. We will order a second revision of the board during the beginning weeks of EE 492 **??** to account for a mistake found in the pinout of the 40P connector in the first revision.

Sniffer Node Power Supply Design

The first step in designing the power supply is to estimate the battery capacity need; we performed several measurements on the CC1352R development board. TI provides a tool called Energy Trace on their development boards which can accurately measure the power usage of the board. The Sniffer board will be similar to the development board so the estimations made using this tool will give a good estimate for our design.

The development board was tested under two configurations. The first was with no transmissions being sent. This represents typical operation while listening for packets and recording data. The second was transmitting every 5ms. This represents the maximum frequency at which packets would be sent while transmitting data back toward the Host.

The test results for both configurations are shown in Figures 10 and 11 below. The data includes the maximum, minimum, and average power and current draws over a 10 second test window.

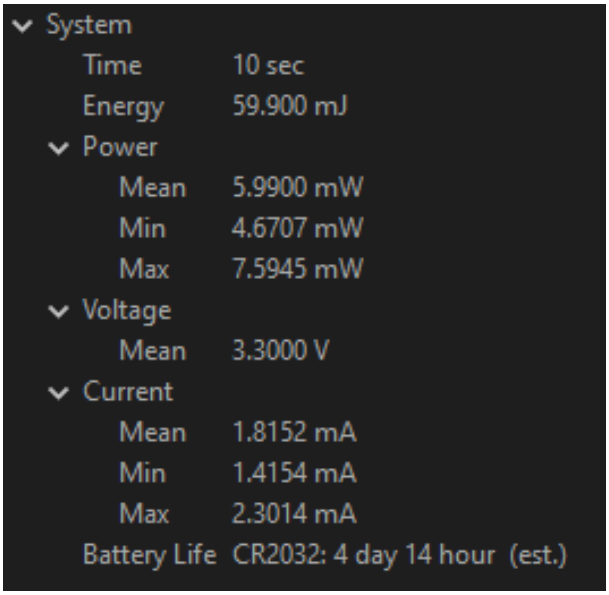


Figure 10: Power Usage Data When not Transmitting

Transmitting every 5ms is much more frequent than the we plan to in the final design. Additionally, the Sniffer will be listening and not transmitting at least 50% of the time. So for a worst case estimate of the power usage of the Sniffer, we can assume that the Sniffer is in normal operation 50% of the time and transmitting every 5ms 50% of the time. The calculations for average power usage and energy needs can be calculated as shown below.

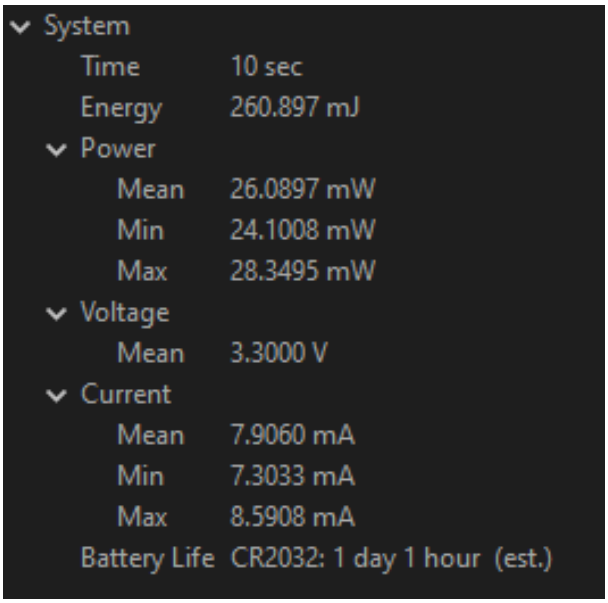


Figure 11: Power Usage Data When Transmitting Every 5ms

$$P_{avg} = 0.5(P_{normal}) + 0.5(P_{trans,5ms})$$

$$P_{avg} = 0.5(5.99) + 0.5(26.09) = 16.04mW$$

The Sniffer will need to run for up to 7 days continuously. The energy needed for 7 days of continuous operation can be estimated as follows.

$$E_{wk} = P_{avg}(7)(24)(60)(60) = 9.701kJ$$

Typically, battery capacity is stated in Ah (ampere-hours) or mAh (milliampere-hours). Ah is calculated by multiplying the number of amps by the hours of usage needed. Similarly, mAh can be calculated by substituting mA instead of A. The battery capacity needed for the Sniffer node is estimated below using the same assumptions that were made for the average power usage estimation.

$$capacity - needed = (0.5(I_{normal}) + 0.5(I_{trans,5ms}))(7)(24)$$

$$capacity - needed = ((0.5)(1.8152) + (0.5)(7.9060))(7)(24) = 816.581mAh$$

Rounding this value up, the worst-case scenario battery capacity needed to meet the require-

ments can be estimated as 820 mAh. However, the battery will not provide a constant voltage, but rather the voltage will decrease as the battery discharges. The power requirements of the Sniffer will remain the same so more current will be drawn from the battery. This results in a larger mAh need. The voltage affects can be inserted into the mAh estimate as follows.

$$capacity - needed = (\frac{P_{avg}}{V_{supplied}})(7)(24) = \frac{2695}{V_{supplied}} mAh$$

To perfectly predict the needed battery capacity, the voltage curve of the battery would need to be modeled and the capacity equation above would need to be integrated with respect to the voltage. This would be tedious so instead a minimum and maximum mAh will be calculated given the minimum and maximum voltages of the battery and judgement will be used to determine if the capacity of the battery will be high enough. Then when the first revision is done, we will perform more testing to determine if the battery needs to be larger or not.

Our team generated three main design ideas for the battery portion of the power supply. The three ideas are listed below with a rough cost estimation as well as the pros and cons of each idea. All costs are estimates based on research. All rechargeable solutions provide enough charging capacity to charge all batteries in one round of charging.

1. AA batteries which will be replaced when discharged

A typical AA or AAA battery has a voltage of 1.5V and discharges to about 1V before a strong drop-off. Typically, AA batteries have about 2500 mAh of capacity. Standard mounts for the batteries are cheap and readily available. Two AA batteries could be connected in series to provide 3V maximum and 2V minimum to the board. The minimum needed capacity for this setup would be 900 mAh and the maximum would be 1350 mAh. With a typical capacity of 2500 mAh, this setup should easily be able to meet the capacity requirements.

The cost estimation of this solution is as follows:

Item	Cost per Item	Quantity	Total Cost
AA Battery	$0.80	2	$1.60
Battery Mount	$2.00	1	$2.00

Table 2: Cost Estimation for Battery Solution 1

Total cost per board: $3.60

Pros:

- Low complexity
- Cheap

Cons:

- Not very sustainable
- Marginal cost of replacing batteries ($1.60 per board per replacement)

2. AA batteries which will be replaced when discharged

A typical AA rechargeable battery has a voltage of 1.2V and discharges to about 0.8V before a strong drop-off. Typically, these batteries have about 2000 - 2400 mAh of capacity. Standard mounts for the batteries are cheap and readily available. Three batteries could be connected in series to provide 3.6V maximum and a 2.4V supply to the board.

The minimum needed capacity for this setup would be 750 mAh and the maximum would be 1123 mAh. With a typical capacity of 2000-2500 mAh, this setup should easily be able to meet the capacity requirements.

The cost estimation of this solution is as follows:

Item	Cost per Item	Quantity	Total Cost
AA Recharg. Batt. (Amazon Basics)	$1.10	30	$33.00
Battery Mount	$2.00	10	$20.00
Battery chargers (cheap Amazon ones)	$15.00	4	$60.00

Table 3: Cost Estimation for Battery Solution 2

Total cost per board: $11.03

Pros:

- Low complexity
- Sustainable
- No replacing batteries

Cons:

- Unknown Amazon battery quality (has good ratings though)
- External chargers needed
- Batteries must be kept track of

3. Rechargeable LIPO batteries

Many LIPO batteries have voltages that range from roughly 2.8V – 3.6V and can have various different capacities. It is not difficult to find LIPO cells that have the needed voltage range and have large capacities of over 1 Ah. These batteries easily satisfy the power requirements. The downside is that they need charging circuitry and protection circuitry. They also do not have as standard of sizes which will make them more difficult to mount. Our idea is to use a LIPO on each Sniffer with a protection and or management IC. Then we would also design separate charger boards which have charging ICs. This would allow the batteries to be taken out and charged. This increases modularity.

The cost estimation of this solution is as follows:

Item	Cost per Item	Quantity	Total Cost
LIPO	$5.00	10	$50.00
Battery Mount	$3.00	10	$30.00
Protection/Management ICs	$0.50	10	$5.00
Charger ICs and parts	$1.00	10	$10.00
Charger PCB	$15.00	1	$15.00

Table 4: Cost Estimation for Battery Solution 3

Total cost per board: $11.00

Pros:

- High capacity allows for less recharging
- Sustainable
- No replacing batteries
- High voltage supplied

Cons:

- Complexity
- Longer design and testing time
- Non-standard sizes make mounting more difficult

Of these three options, our preference is number 3. However, we will review the options with our client before the final decision is made.

3.8 Technology Considerations

3.8.1 Two band-pass off

The two-band pass-off design uses the CC1352s' dual-band capabilities to switch the Sniffer between the listening to the BOBs and sending data to the next Sniffer or Host in the communication line. This process alternates the Sniffer between different bands, ensuring that the communication back to the Host does not interfere with the communication between BOBs. This is the method described above in Design 1. The advantages of this design are that we have the setup for the CC1352 completed and we know it can be done this way. The disadvantage is that it requires precise timing to ensure that the packets are not missed from the BOB nodes. It also gives challenges and throttle to the packet received and passed back to the Sink due to having less than 50% of the time that it can do it.

3.8.2 Internet Enabled Sniffer Node

This design described in Design 0 uses an ESP microcontroller such as an ESP01 to connect to a WIFI network to pass data directly back to the Host without the need for a Sink node. This design advantages are that it does not require a Sink node and does not need to have a predetermined path back to the Host allowing for a much larger test bed. The disadvantages are that it adds the complexity of a second microcontroller into the system and the complexity of connecting to the internet. The main disadvantage of it is the high energy requirements of it causing a large drain on the battery power of the Sniffer.

3.8.3 Two CC1352 Design

In this design we have two CC1352s built into the design one that transmits data back to the Sink node on the Sink node band and another that collects the communication data back from the BOB and the two communicate via UART or SPI. An advantage of this design is not having to switch between Bands and having to maintain exact timing to ensure packets are not dropped. The disadvantages are the additional cost, complexity, and power of having two CC1352s on the Sniffer.

3.8.4 Both bands running on the CC1352 Simultaneously

Late in the design phase of this semester, we found that we could use both sub-1-GHz and a 2.4GHz BLE on the CC1352s at the same time. In this design we would listen to the BOB nodes on sub-1-GHz and communicate back to the Sink with BLE. The advantage of this design is the simplicity of the design not having to deal with switching bands or a second microcontroller. The disadvantage is that we do not have a clear example to work with.

3.8.5 Single band channel filtering

This design makes use of multiple channels and filtering within a single Band for the CC1352. The design would have the CC1352 for the BOB have a small filtered range around its communication frequency and the Sniffer CC1352 would have a large area of filtering that would include the BOBs transmit frequency and would transmit outside of the BOB's CC1352s filtering range. The advantage of this design is that it only uses one band. The disadvantage of this design is that depending on the way the CC1352s filter it could disturb the BOBs and that packets could be missed.

3.9 Design Analysis

The design we landed upon was the dual CC1352 design. We chose this one as it was the one we were most confident would work.The concern with trying to do 1 cc1352 was if it could not handle two bands simultaneously and not drop a single packet. Our concern with band switching was ensuring the timing was perfect. If the timing got off much, then we would start dropping packets. We implement the duel cc1352 design because future work could switch to a single cc1352 design.

We used two AA batteries for the battery system because it was the cheapest option that could still provide enough power without being wasteful or having the danger of Lithium Ion batteries.

4 TESTING

4.1 Unit Testing

4.1.1 Software

Our software implementation is primarily interrupt-based coding, so our unit testing units are primarily chunks of interrupt activation and callbacks. Thus most of our unit testing was done within CCS using the provided debug tools, as well output monitoring using PuTTY terminals and a debugger UART connection.

UART interrupts: The Sink sniffer node uses UART to communicate back to the host computer, which runs a Python script for data logging and processing. There are several UART commands defined for the sniffer to handle, determine different functions and different paths through the interrupt handler. To test these we set breakpoints within each of these paths in the intended function and send a UART package with the package ID for each path, and tested that every defined function worked as expected.

Packet ID	**Intended Path to Set Breakpoint**
UART_SET_NODE_MESSAGES	A function that sets the node messages by sending a radio message to all the Sniffer nodes
UART_SET_NODE_LAMBDA	A function that sends the node Lambda for the test configuration to the Sniffer nodes
UART_START_EXPERIMENT	Function to send a start experiment to the Sniffer nodes via the radio
UART_RESET_EXPERIMENT	Function to send a reset experiment to the Sniffer nodes
UART_SET_NODE_CONFIG	A function that sets the node config by sending a radio message to all the Sniffer nodes
UART_HANDSHAKE	A function to complete the handshake with the Python script

Table 5: UART Unit Tests

Radio Interrupts: Each Sniffer node uses the radio unit to communicate back to the sink, as well as for listening to the BOB network. There are many radio commands possibly received, each of which needed to be tested for triggering the interrupt correctly, as well as entering the correct section of code based on command ID.. Our radio unit interrupts were largely tested using a second cc1352 radio board configured to send out arbitrary data packets at fixed rates to trigger the radio callbacks. We then monitored what state the code was in, again either with the debugger or VIA the PuTTY window.

Payload	Intended Path to Set Breakpoint
RADIO_SET_NODE_MESSAGES	A function that sets the number of events for the Sniffer node to generate.
RADIO_SET_NODE_LAMBDA	A function that sets the node Lambda for the test in the Sniffer node
RADIO_START_EXPERIMENT	A function that starts the experiment on its respective BOB node.
RADIO_RESET_EXPERIMENT	Function to reset the experiment with its BOB node
RADIO_SET_NODE_CONFIG	Function to send the config to the BOB node
RADIO_HANDSHAKE	A function to complete the Radio handshake with the Sink Sniffer
TIME_SYNC	For non-Sink nodes, a message from the Sink is needed to maintain a synchronized clock.
TYPE_ACK	Function to save the acknowledgment back to the queue to be eventually sent back to the Sink.
RAW_DATA	Function to save the raw data from BOB back to the queue to be eventually sent back to the Sink.

Table 6: Radio Tests

*Sink Sniffer Node Only will have a acknowledge version of these packets

**Sink Sniffer Node Only will have an interrupt for this aspect to save it back to the Host.

Time Interrupt: The time interruption needs to occur precisely and urgently to achieve the pass of off the roles of data collection and data aggregation back to the Sniffer node. To test this, we will had the module send a message via UART to the Python script with the exact timing of the interrupt. We used this test to make sure we can establish and maintain strict timing to ensure no packets are lost.

GPIO Interrupts: The Sniffer node collects data about the life cycle of the BOB node and it transmission patterns via GPIO pins on the Sniffer nodes. To track this and record it we have interrupts on these pins to record the timing and send data on to the Host. Similar to the UART and Radio we will test it by setting breakpoints and then manually pulling pins up and down to ensure that the interrupt is triggered.

Pin	**Record to Make**
Powered On	Function to add a power on or power down event to the radio send queue
Easylink TX	Function to add a transmission event to the radio send queue
Event Drop	Function to add an event drop event to the radio send queue

Table 7: GPIO Interrupt Tests

Queue: The final unit for our software team to test is not an interrupt-based function but is a queue that has been previously mentioned that holds a backlog of events that need to be logged by being sent back to the Host. We tested this using a timer interrupt set to the maximum rate at which packets will be added to a queue. We then added packets to a queue, counting up to the maximum storage amount required of the queue. Once that was complete, we iterated through the queue removing items from the queue at the required rate while ensuring that every item was accounted for.

4.1.2 Hardware

Below is a description of the general unit tests and process that will be taken for testing each new PCB that is part of the system. For the purposes of our senior design project, this will include 3 PCBs named as following: Breakout Board, MSP_Simplified, and Sniffer. These will be broken into their own sections to provide specific details of unit testing each board.

Step 1 of testing occurs before the board is fabricated. This includes checks that are made during the design process. The ERC, or electrical rules check, is used in KiCAD to verify correct schematic connections. This check will be performed throughout the schematic

design, and will need to be passed before the layout is generated. Next, DRC, design rule checking, will take place in the layout. In KiCAD, this will check for mismatches between the schematic and layout, as well as layout custom rules. These rules are provided by the fabrication company that you are ordering from and defined in the design tool by the designer. Design rules ensure that the board can be reliably manufactured and that it follows good design practices. These two checks are vital when designing a PCB.

Step 2 occurs after the board has been fabricated and populated. The designer visually inspects the board for defects with a microscope. Defects can be pins that do not have a solder connection to the pad, shorted pads, peaks in the solder, and burned components. Often the designer will find a number of issues in this step.

In step 3, the populated board is connected to a power supply and the supply current is measured. The power supply current is limited to a safe maximum to protect the board while allowing adequate current for normal operation. If the supply current jumps to the maximum allowed value, then there is likely a short in the board. This check serves to find major issues before the board is used and protects against damaging the board. The board software independent functionality can also be tested using power supplies, multimeters, oscilloscopes, and logic analyzers.

In step 4, the debugger is connected, and the board is programmed with any software it will run. Often, if there is an issue on the board, the debugger will not be able to successfully communicate with and program the board. This would indicate the need to go back to steps 2 and 3 to find the issue in the board.

In step 5, the software dependent elements of the board are tested by simulating the operating environment. For example, if a motion sensor was used on the board, then the board could be shaken and the data in software could be analyzed to see if motion is sensed or not. Initial tests are very basic and usually just establish communication with peripherals or basic functionality. Further testing and calibration are pursued later to verify that the elements work as expected.

Element Under Test	Testing Task	Expected Result
Solder Joints	Visually inspect the PCB with a microscope for any soldering issues.	Ensure none of the following are present: Shorts between pins, peaks of solder at connections, solder on pin but not connected to pad, component pins not aligned.
Components	Visually inspect the PCB (with a microscope for 0402 or smaller) for any issues with components or the board	Ensure none of the following are present: Leftover flux on board, burned or damaged discrete components, missing pins, melted plastic, damaged pads or traces, missing parts.
Power to ground isolation	Continuity test between power and ground	Should not see any measurable current exchange.
Power rail check	Power the 3.3V rail and GND rail with a power supply. Probe the 3V3 and GND test points with a multimeter.	The expected outputs are 3.3V and 0V, respectively. Significant margins (100s of mV) off this expected value should be evaluated further.
Header pin isolation	Run continuity checks between all neighboring pins with a multimeter.	The expected output is no beeping (this indicates connectivity) from the multimeter, when the nets are not connected on the schematic.

Table 8: Breakout Board Unit Testing

Element Under Test	Testing Task	Expected Result
EN Test for load switch	Provide 3.3V and GND power rails to the PCB with a power supply. Measure the VOUT voltage with a multimeter. Next, provide 1V on the enable pin and measure the VOUT voltage with a multimeter.	With no enable voltage provided we expect to see 0V on the output and with the enable voltage we expect approximately 3.3V (within 100mV) on the output.
Monitor Current output from the load switch	Using an ammeter, probe the open circuit provided on the board. This can be done either across the two-pin connector or the DNP resistor.	Should show very little current drawn from the load switch, as indicated on the datasheet.

Table 9: Breakout Board Unit Tests

Element Under Test	Testing Task	Expected Result
Solder Joints	Visually inspect the PCB with a microscope for any soldering issues.	Ensure none of the following are present: Shorts between pins, peaks of solder at connections, solder on pin but not connected to pad, component pins not aligned.
Components	Visually inspect the PCB (with a microscope for 0402 or smaller) for any issues with components or the board	Ensure none of the following are present: Leftover flux on board, burned or damaged discrete components, missing pins, melted plastic, damaged pads or traces, missing parts.
Power to ground isolation	Continuity test between power and ground	Should not see any measurable current exchange.

Table 10: MSP Simplified Unit Testing

Element Under Test	Testing Task	Expected Result
Power rail check	Power the 3.3V rail and GND rail with a power supply. Probe the 3V3 and GND test points with a multimeter.	The expected outputs are 3.3V and 0V, respectively. Significant margins (100s of mV) off this expected value should be evaluated further.
Header pin isolation	Run continuity checks between all neighboring pins with a multimeter.	The expected output is no beeping (this indicates connectivity) from the multimeter, when the nets are not connected on the schematic.
EN Test for load switch	Provide 3.3V and GND power rails to the PCB with a power supply. Measure the VOUT voltage with a multimeter. Next, provide 1V on the enable pin and measure the VOUT voltage with a multimeter.	With no enable voltage provided we expect to see 0V on the output and with the enable voltage we expect approximately 3.3V (within 100mV) on the output.
Hardware reset buttons	Work with Software to power up and flash the microcontroller. Depress and release the reset button (S2)	Software indicators show a successful reset, and that the device can continue normal operation after a reset.

Table 11: MSP Simplified Unit Tests

Element Under Test	Testing Task	Expected Result
Solder Joints	Visually inspect the PCB with a microscope for any soldering issues.	Ensure none of the following are present: Shorts between pins, peaks of solder at connections, solder on pin but not connected to pad, component pins not aligned.
Components	Visually inspect the PCB (with a microscope for 0402 or smaller) for any issues with components or the board	Ensure none of the following are present: Leftover flux on board, burned or damaged discrete components, missing pins, melted plastic, damaged pads or traces, missing parts.
Power to ground isolation	Continuity test between power and ground	Should not see any measurable current exchange.
Power rail check	Power the 3.3V rail and GND rail with a power supply. Probe the 3V3 and GND test points with a multimeter.	The expected outputs are 3.3V and 0V, respectively. Significant margins (100s of mV) off this expected value should be evaluated further.
Header pin isolation	Run continuity checks between all neighboring pins with a multimeter.	The expected output is no beeping (this indicates connectivity) from the multimeter, when the nets are not connected on the schematic.
Hardware reset buttons	Work with Software to power up and flash the microcontroller. Depress and release the reset button	Software indicators show a successful reset, and that the device can continue normal operation after a reset.

Table 12: Sniffer Unit Tests

A few things that we took note of while doing the unit tests for the sniffer are identifying shorts, verifying connectivity, and other debug issues that came up as software programmed

the boards.

To start, we were able to reflow solder the PCBs and then touch up and add additional components that don't reflow well, such as the SMA connectors. With reflow soldering, we came across quite a few shorts across pins. Particularly with the CC1352 MCU. These were resolved after the board was baked in the reflow oven. We applied flux, heated up a soldering iron, and ran it across the entire row of shorted pins. This was able to spread the solder across multiple pins and typically resolved the issue. If shorting still would arise, we used solder wick, which is essentially a roll of weaved copper that when heated will soak up some of the extra solder. This method was also done for the Balun Filter which also frequently had solder shorts.

After the boards were soldered we would do another full inspection and tested the power rails and placement of components. One issue that was discovered while software was getting an initialization issue while programming. We found that the crystal oscillators on REV1 of the board were soldered on in the incorrect direction. This was noticed when REV2 was soldered and programming correctly. This was noted, and the future soldered boards were double checked when components were placed.

4.1.3 Additional Hardware Unit Test Plans Added in Semester 2

Several additional test plans were added in the second semester of Senior Design as the design changed and more features were added. These are detailed below.

Testing and Tuning the Oscillator Load Capacitors

The crystal oscillator used for generating RF signals is loaded by capacitors in the CC1352R chip. The chip has a capacitor bank and the capacitor values can be adjusted in the software. If the load capacitors are not set correctly, then the maximum RF output power may not be centered at the correct frequency. Tuning the capacitors places the output power maximum at the desired frequency.

Note that a spectrum analyzer is needed for this testing.

The testing steps are laid out below (test plan based on [2] and [3]):

1. Remove C42 and place a 12pF blocking capacitor at C43
2. Connect a spectrum analyzer to J8 using a female-female coaxial cable
3. Configure the CC1352R to output 2.4GHz signals
4. Observe the output spectrum on the spectrum analyzer
5. Adjust the oscillator loading capacitors in the software and observe the output spectrum on the spectrum analyzer
6. Repeat (5) until the output spectrum is best centered in the 2.4GHz ISM band
7. Copy settings to all other CC1352R chips (will not repeat testing on every board to save time)

Testing and Tuning the 2.4GHz PCB Antenna Matching

The 2.4GHz input impedance is unknown and subject to great variation depending on its surroundings. For this reason, measurements need to be taken so that a suitable matching network can be designed, implemented, and verified. The steps below lay out the process for matching the PCB antenna.

Note that the other antennas are not included because they are assumed to have a fixed, 50 ohm impedance due to how they were manufactured.

The testing plan below is based on [4].

1. Replicate the operating conditions of the Sniffer board (PCB antenna can be greatly affected by its surroundings such as batteries, its housing, humans, etc.)
   (a) Place the battery pack nearby
   (b) Place on top of other dev board PCBs
2. Remove the DC blocking capacitor
3. Solder a semi-rigid coaxial cable to the DC blocking capacitor pad and to ground (ground connection supplied by design or created by scraping of silkscreen)
4. Calibrate the VNA using an open circuit, short circuit, and 50 ohm load
   (a) Open circuit is created by removing all matching components
   (b) Short circuit is created by placing a 0 ohm resistor at C6
   (c) 50 ohm load is created by placing two 100 ohm resistors on top of one another at C6 (two 100 ohm resistors have less ESL than a single 50 ohm resistor)

5. Perform port extension on the VNA using the short circuit load to place the reference plane at the matching network
   (a) This is done by adjusting the port extension settings on the VNA until the desired frequency marker is near the short circuit point on the Smith Chart
6. Connect the VNA to the antenna by placing a 0 ohm resistor at C5
7. Record the position of the 2.4GHz marker on the Smith Chart
8. Calculate the needed matching components given the position
   (a) This can be done using Smith Chart computer tools or calculations
   (b) Optionally perform simulations using the actual component models in ADS to get a more accurate matching prediction
9. Place matching components one at a time and adjust as necessary to get the 2.4GHz marker near the 50 ohm point on the Smith Chart
10. Unsolder the coaxial cable and replace the DC blocking capacitor
11. Place identical components on all other boards (will not repeat testing for every board to save time)

Some extra notes on testing and tuning the 2.4GHz PCB antenna matching:

1. Tuning would be much easier if a kit of 0402 capacitors and inductors were available. However, it is likely this would not be available. The backup plan is to calculate the expected values, order some part values at and around the expected values, and then to repeat the testing to tune the matching network.
2. A semi-rigid coaxial cable will be needed to solder to the board.
3. The calibration of the VNA can also be done using a precise calibration kit [3] – again, not sure if this is something we have or not.
4. Some people resolve the imaginary part mismatch using a capacitor or inductor and then resolve the real part mismatch by cutting down the antenna [3]. This will be avoided to make the results more replicatable across multiple boards. This could be done on one board if a second revision needed to be made. Then the length could be adjusted on the new revision.

Buck-Boost Regulator Testing

The testing plan for the regulator is given in Table 13. The test results can be found in the Hardware Results section in Table 19.

Element Under Test	Testing Task	Expected Result
Regulator boost functionality	Supply the regulator with a voltage between 1.8-3.3V and measure the voltage at the regulator output.	The regulator output should have a voltage of 3.3V.
Regulator buck functionality	Supply the regulator with a voltage between 3.3-5.5V and measure the voltage at the regulator output.	The regulator output should have a voltage of 3.3V.

Table 13: Regulator Unit Test

4.2 Interface Testing

4.2.1 Software

The software interface testing consisted of testing our various communication interfaces between nodes and parts of nodes. We tested them both for maximum bandwidth, as well as accuracy of data transmission, so as to be sure we would never drop data from the test system. Similar to the unit tests, we made use of CCS, several CC1352s, and a python script to test these interfaces.

UART: To record the data at the Host, all data from the test needs to be communicated from the Sink to the Host. We tested this with a timer to create a UART data rate being sent from the CC1352 Sink to the Host computer at a rate of 1 packet every 1 ms. As packets are created, it increased the counter to ensure a unique identifier for each packet. We can then go through all records and ensure all are accounted for. This was succesful at the maximum rate we calculated nodes could feasably generate packets.

Radio Transmit and Receive:

Once all the unit tests on the Sniffer PCBs were completed, then the radio interface between boards was tested. The goal of these tests were to confirm that the radio was able to send and receive and to test the quality of the RF system. To complete this testing, a program

developed by TI called SmartRF Studio was used. SmartRF Studio allows for quick and easy testing of TI's CCxxxx products. The performance of the Sniffer PCBs were bench marked against the developement boards from TI. The testing plan is listed below.

1. Determine Development Board Performance as a Benchmark
    (a) Open SmartRF Studio
    (b) Connect a development board (REF1) through a debugger to the computer
    (c) Ensure the device appears and connects in SmartRF Studio
    (d) Configure the device for transmitting in SmartRF Studio. Setup the proper packet settings. Set the number of packets to be sent to 100. Set the frequency to 2.4GHz
    (e) Connect a second development board (REF2) through a debugger to the computer
    (f) Configure the device for receiving in SmartRF Studio. Setup the proper packet settings. Set the frequency to 2.4GHz
    (g) Place the boards roughly 2 meters (6 foot 6 inches) apart with their antennas pointing towards each other
    (h) Start the receiver so that it begins listening for packets
    (i) Start the transmitter so that it sends packets
    (j) Record how many packets were received and what the RSSI (Received Signal Strength Indicator) value
    (k) Swap the board roles (receiver now transmits, transmitter now receives)
    (l) Repeat steps i-k
2. Repeat Step 1 with the Frequency set to the Sub-1GHz Band
3. Test the Sniffer PCB at 2.4GHz
    (a) Open SmartRF Studio
    (b) Connect a development board (REF2) through a debugger to the computer
    (c) Ensure the device appears and connects in SmartRF Studio
    (d) Configure the device for transmitting in SmartRF Studio. Setup the proper packet settings. Set the number of packets to be sent to 100. Set the frequency to 2.4GHz
    (e) Connect a Sniffer PCB through a debugger to the computer
    (f) Configure the device for receiving in SmartRF Studio. Setup the proper packet settings. Set the frequency to 2.4GHz
    (g) Place the boards roughly 2 meters (6 foot 6 inches) apart with their antennas pointing towards each other
    (h) Start the receiver so that it begins listening for packets

(i) Start the transmitter so that it sends packets
(j) Record how many packets were received and what the RSSI (Received Signal Strength Indicator) value
(k) Swap the board roles (receiver now transmits, transmitter now receives)
(l) Repeat steps i-k

4. Repeat Step 3 with the Frequency set to the Sub-1GHz Band

The benchmark test setup is shown in Figure 12 and the Sniffer test setup is shown in Figure 13. The test results can be found in the Hardware Results section of the design report.

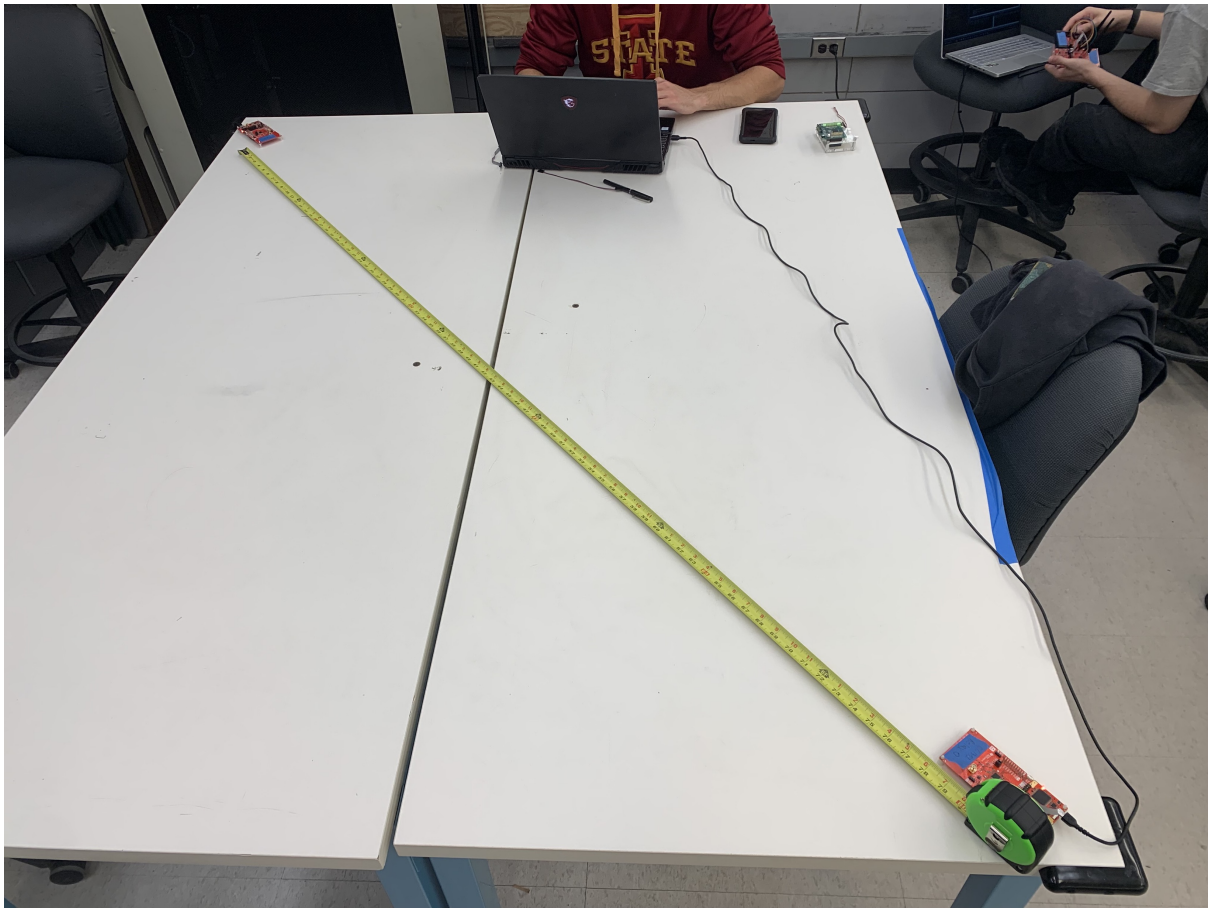


Figure 12: Radio Transmit and Receive Benchmark Test Setup

SPI Transmit: For the SPI system, we again tested the data rate. We set up a maximum sized 19 byte packet and sent it at varying frequencies, down to 1 ms between transactions. This is a substantially faster rate than we will ever feasibly be producing packets.

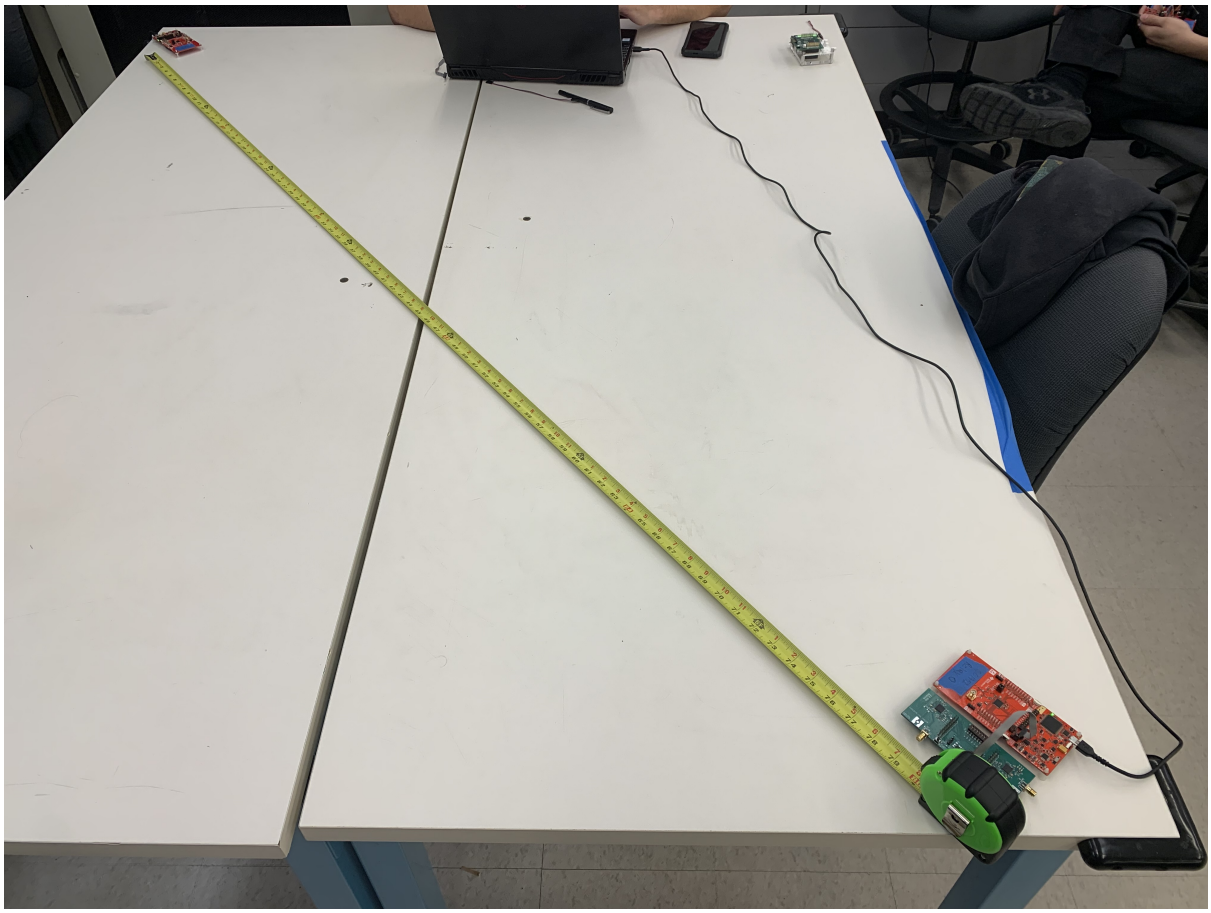


Figure 13: Sniffer Radio Transmit and Receive Test Setup

4.2.2 Hardware:

Element Under Test	Testing Task	Expected Result
Debugger and Programmability of Microcontroller	Connect PCB to power. Connect the debugger to the computer and to the PCB. Set up any necessary parameters on the computer side. Flash the microcontroller with the code.	Debugger is able to establish communication with the microcontroller. Device is successfully flashed.
Test 3V3_CC line on the CC board	When the board stack is in place and initial unit testing is complete for the MSP_Simplified, probe the 3.3V line on the CC1352 with a voltmeter.	You should read approximately 3.3V on the multimeter for the power rail on the CC1352 board.

Table 14: MSP_Simplified Interface Testing

4.3 Integration Testing

4.3.1 Software

Avoids Lockup: With all the interruptions that are occurring, there is the possibility of a lockup scenario. To test for this, we took our Sniffer node and drove the GPIO pins high and low while simultaneously having radio transmissions incoming and having the timing interrupts active. We ensured these all occur correctly via a log sent out via UART packets.

A similar testing set up will be done for the Sink Sniffer node setup, where it also has UART commands being sent to it from the Host.

4.3.2 Hardware

Stack integration- Test signals both the I/O and power ports in the main Boosterpack connector that is distributed throughout the stack. Ensure one does not impact the performance of the entire stack.

Integration between the physical boards to ensure mechanical stability throughout the entirety of the stack.

Element Under Test	Testing Task	Expected Result
Stack Integration	Probe the power port and I/O connections in the boosterpack connector. The I/O will need to be run with a series of software tests to set and then measure the recorded I/O values. Ensure that there is no loss in power droop across the stack and no effects from I/O while other boards are running.	We expect to see the programmed I/O values as well as a consistent shared power rail across the stack. Isolation is expected across GPIOs that are being written on another board.
Mechanical Stack Integration	Test mechanical durability between the physical boards to ensure mechanical stability throughout the entirety of the stack.	We expect the stack to stay intact and operable through all mechanically strenuous conditions.

Table 15: MSP_Simplified Integration Testing

4.3.3 Sub-System

MSP430: To test the Custom MSP-430 board we flashed the program that is currently running the tests onto the new board and ensured it still ran by checking UART output.

Custom Sniffer Board: Similar to the custom MSP-430, we loaded code onto our custom board and ensured all IO worked correctly.

Custom BOB stack: With the updated code sets above for the alternative pinouts, we can run our full stack doing a test with the original code base, just our pinout and our hardware, ensuring the hardware works properly together.

4.4 System Testing

For our system level tests, the main tool we created was what we called a *Faux BOB*. This Faux BOB was a CC1352 radio node that was coded to simulate data that would come from a BOB but with controlled data. Right now it is fairly simple and used mostly to test radio transmissions at set intervals, but in the future we plan to build it out with functionality to simulate BOB on-times and transmission cycles, as well as adding a larger variety of potential packets and different timings.

The use of this Faux BOB was so that when we are testing our sniffer nodes, we know exactly when and what we are supposed to be receiving, to eliminate many of the other variables that come along with a real BOB node. It ensures that we are receiving the right data consistently, and expedites testing greatly.

To test the Sniffer setup in a less-known environment, we will set our test next to the current setup running a test and comparing the logs to ensure that they get the same logged data. We can receive the same wireless data being “sniffed” from the BOBs running and be connected into the power and transmit pins with the GPIO of the Sniffer. The data logged by this will then be mapped to the current systems log to ensure that they line up properly.

4.5 Regression Testing

Our regression testing will consist exclusively of the mock BOB node that will be designed for system testing. To ensure that future iterations of our design continue to fit requirements, the mock BOB node can have a test run on it, and if all the predetermined data is recorded, it will help ensure functionality does not get broken. It will continue to use the Python script, our testbed setup, and a CC1352 development board.

4.6 Acceptance Testing

The acceptance tests consist of two parts:

We will be given a simple test setup that the research assistants run on the current setup. We will run this test on our setup and ensure that our code configures things correctly and that the code has the data output all recorded and organized in an acceptable way for the researchers.

The follow-up acceptance test will give our product setup with documentation to the research assistants and have them set up and run their test with the exclusive help of the documentation we wrote to ensure they will be able to make use of it once we graduate and move on and future research groups can use our infrastructure.

4.7 Results

4.7.1 Hardware Results

For our hardware, we performed unit tests on both of our PCBs that were manufactured. This includes the Breakout Board and the MSP Simplified. The Breakout Board passed all of the unit checks that we had listed in our test plan. The MSP Simplified did not meet all of the unit checks. We found issues with the design of the board as there was an incorrect footprint for one of the components, and the order of the pins was reversed for one of our 20P connectors. This makes it unusable in the hardware stack, and so a second revision will be ordered during the first few weeks of EE 492. This is outlined in our Gantt chart **??**.

Unfortunately, we were not able to obtain access to a spectrum analyzer and a vector network analyzer in time to perform the RF testing and tuning. As a backup plan, we performed computer simulations to test the antenna and find the matching values. This process is explained below.

First, the 2.4GHz PCB antenna was modeled in CST Studio Suite. The layer thicknesses and and physical properties were set based on the values provided by the manufacturer. The board outline, antenna design and placement, and the via placement was modeled to be identical to the Sniffer PCB. The rest of the components were left off to get the modeling done

without taking an exorbitant amount of time. The 3D model is shown in Figure 14.

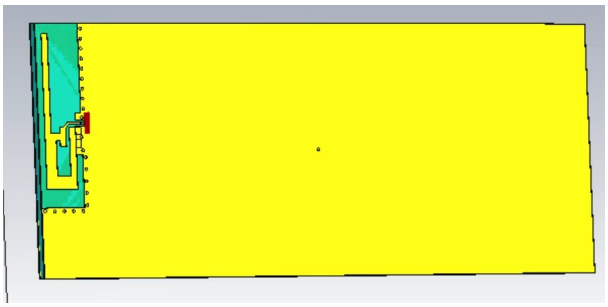


Figure 14: 3D Model of PCB with 2.4GHz Antenna in CST Studio Suite

Simulations were performed on the model to obtain the S-Parameters for the antenna. S-Parameters are a common metric used to characterize circuit operation in a simple manner at high frequencies. An distance S-Parameter exists for each pair of ports. Since the antenna only has one port, then there is only one S-Parameter called S11. S11 can be roughly thought of as the ratio of power reflected from port one to the power applied at port one. This is not an exact definition but provides some intuition about interpreting the S-Parameter.

The S11 information extracted from the model in CST Studio Suite is shown in Figure 15.

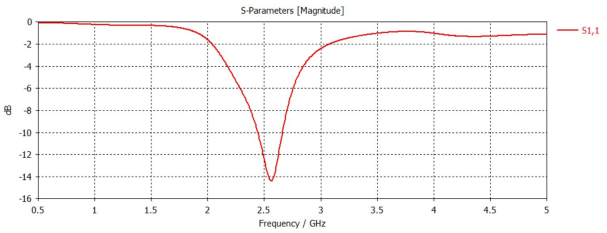


Figure 15: S11 Extracted from CST Studio Suite Model Through Simulation

The extracted S-Parameter data can be used in circuit simulators as a model of the antenna. A common and popular high frequency simulator is Keysight ADS. The S-Parameters were imported into ADS and the input impedance was extracted at 2.4GHz. Using the input impedance, rough values to be used for matching were calculated. The matching values were calculated using a MATLAB script which a team-member developed in a previous class. The script is listed in Appendix V.

Murata provides a number of passive components including several product lines for high speed systems. They provide several ADS libraries which include accurate models of their available parts. Using these models, parts close to the calculated matching values were found

and tested. Figure 16 shows the ADS circuit with the S11 antenna model and the Murata capacitor and inductor models. The Murata parts selected were:

Inductor: LQW15AN3N9D00

Capacitor: GJM1555C1HR90BB01

Both parts are available to order through DigiKey.

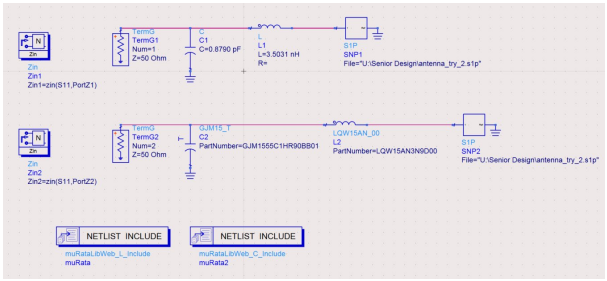


Figure 16: ADS 2.4GHz Antenna Matching Schematic

The results from ADS simulation of the antenna model and matching network are given in Figures 17 and 18.

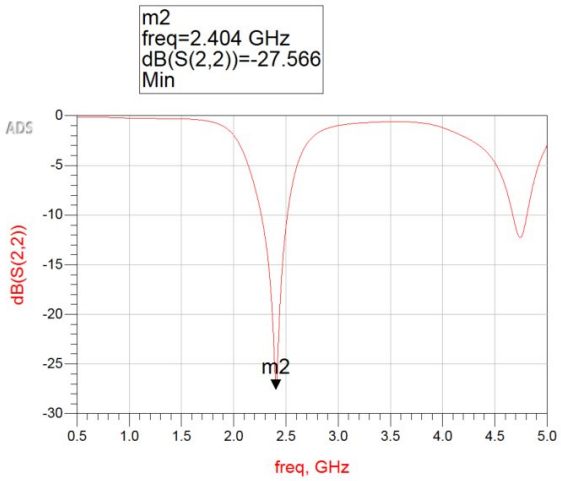


Figure 17: S11 ADS Simulation Results after Matching

The results show that the dip in the S11 plot (the point where almost all power is sent through the antenna) has been moved much closer to 2.4GHz. The results also show that at 2.4GHz, the input impedance is almost exactly 50 ohms as desired.

The drawback of these simulations is that they took too long to include all elements which needed to be modeled. In practice, any objects around the antenna can change the S11

freq	S(2,2)	Zin2
2.400 GHz	-27.138 / -80.493	50.000 + j0.001

Figure 18: Matching Results at 2.4GHz from ADS Simulation

and the matching parameters. Without the BOB PCBs, the battery pack, the mechanical housing, and the other components on the Sniffer PCB being modeled, the simulations can only supply moderately accurate results. More simulation work or physical testing may need to be done in the future to increase the output power of the antennas.

Through our testing so far, we have found a couple of important numbers we needed for decision-making and design work.

- **Worst Case Packet Send Time:** 12 ms
- **Band-Switch Time Cost:** 30 ms

The radio transmit and receive test results are shown below.

The benchmark results are given in Table 16. The 2.4GHz test results for the Sniffer are given in Table 17 and the Sub-1GHz results are given in Table 18. Note that the two development boards used were termed REF1 and REF2. Two Sniffers were used in the Sub-1GHz tests but only one in the 2.4GHz tests. These were termed Sniffer1 and Sniffer2.

Test Configuration	**Frequency Band**	**Packets Sent/Packets Received**	**RSSI (dBm)**
REF1 transmitting, REF2 receiving	2.4GHz	100/100	-52.7
REF2 transmitting, REF1 receiving	2.4GHz	100/100	-53.7
REF1 transmitting, REF2 receiving	Sub-1GHz	100/100	-31.9
REF2 transmitting, REF1 receiving	Sub-1GHz	100/100	-39.8

Table 16: Baseline Radio Test Results

Test Configuration	**Packets Sent/Packets Received**	**RSSI (dBm)**
Sniffer1 transmitting, REF2 receiving	100/100	-53.1
REF2 transmitting, Sniffer1 receiving	100/100	-53.1

Table 17: 2.4GHz Radio Transmit and Receive Test Results

Test Configuration	**Packets Sent/Packets Received**	**RSSI (dBm)**
Sniffer1 transmitting, REF2 receiving	99/100	-80.2
REF2 transmitting, Sniffer1 receiving	100/100	-80.8
Sniffer2 transmitting, REF2 receiving	100/100	-39.2
REF2 transmitting, Sniffer2 receiving	100/100	-33.4

Table 18: Sub-1GHz Radio Transmit and Receive Test Results

The test results show that at 2.4GHz the Sniffer performed at a level consistent with the benchmark. The Sub-1GHz also showed test results consistent with the benchmark but also some that were pretty far below the benchmark (-80dBm RSSI is generally considered to be a low level with mediocre to poor performance). The results show that some boards may perform very well and some may suffer from performance dropoff. The cause of the performance drop is actively being investigated. There was a soldering mistake where the wrong capacitors were placed in the RF sections on several boards. This could potentially be the cause of the performance drop.

Input Voltage	Output Voltage	Result
2V	3.38V	Passed. The voltage is good enough for the PCB to function.
4V	3.36V	Passed. The voltage is good enough for the PCB to function.

Table 19: Regulator Unit Test Results

4.7.2 Software Results

As mentioned earlier in the document, we successfully transmitted rates of SPI, UART, and radio.

Once we built up our code base and had custom hardware, we could continue our testing. Our testing found that the SPI works fine once the GPIO is updated as well as the LEDs we used in our testing. We also found that the rf design worked perfectly for communicating across multiple custom boards. Further testing is described in the demo section below.

4.7.3 Demo

For our demo, we did a 3-node testbed with a width of 2 on the first row and a depth of 1. Due to hardware constraints, we did not have more nodes, but we believe it should scale to a larger testbed relatively seamlessly. The boards functioned as desired in our test, communicating packets back to the Sink and then to the Host. Each node in the demo setup was composed of a Sniffer PCB and a Faux BOB. Future work should validate that all data is recorded back to the Sink and that no packets are missed when increasing the packet rate. As shown in Figure 19, there are two computers. The left one runs Host python code receiving data, and the computer to the right shows the SPI output of node 8 or the node of depth 1 in our demo. In our test, we just powered all boards from a power supply due to delays in getting the regulator support parts needed for using a battery pack as the main power source.

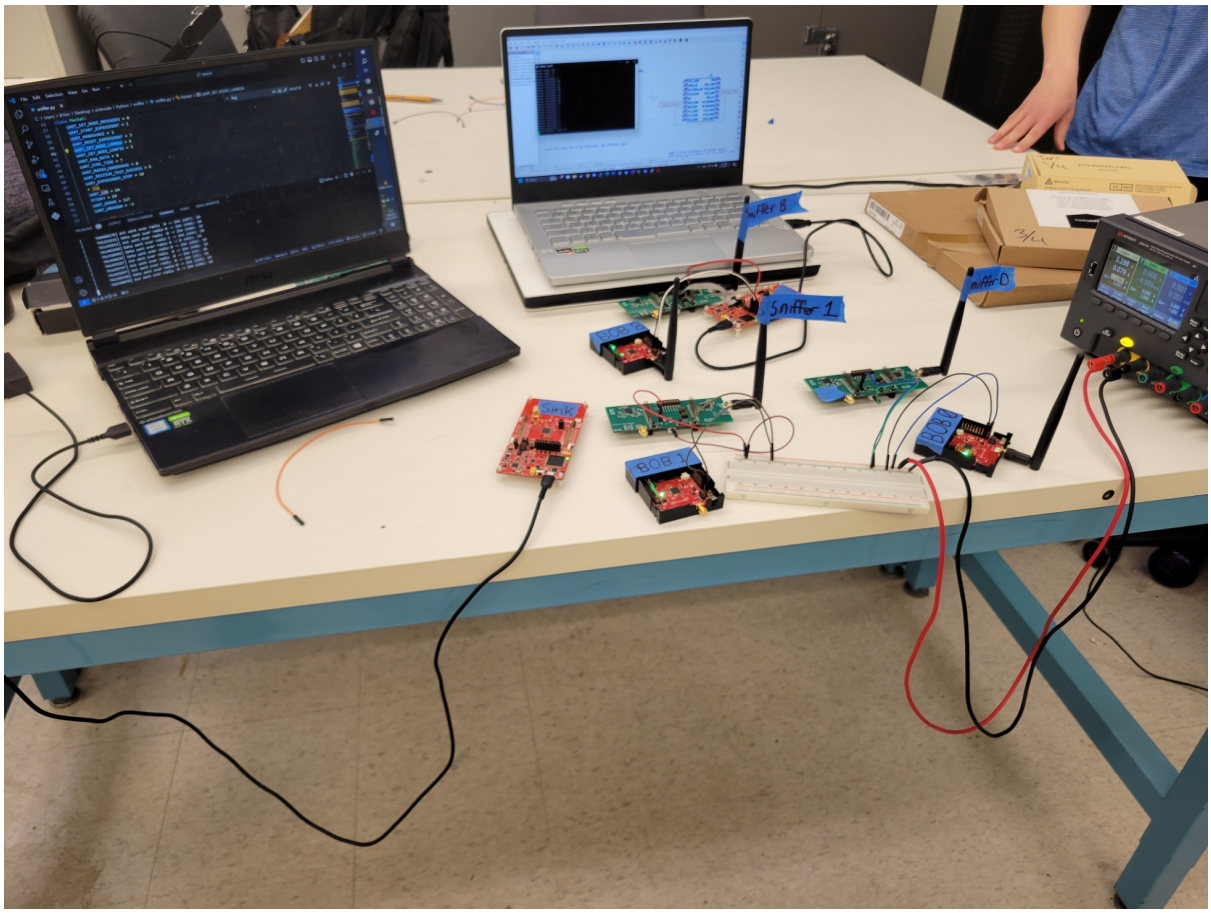


Figure 19: Demo setup

5 Implementation

Much implementation work can be seen in the design section

Our implementation plan is split into hardware and software

5.1 Hardware

1. Assemble MSP Simplified Board
2. Test MSP Simplified Board
3. Complete MSP Simplified second revision
4. Complete Sniffer schematic

5. Complete Sniffer layout
6. Assemble Sniffer PCB
7. Test Sniffer PCB
8. Complete second Sniffer revision, if needed

5.1.1 Sniffer PCB Design

Several major changes were made to the initial Sniffer PCB design from semester one. These changes were:

1. Addition of a second CC1352R processor
2. Addition of a DIP (Dual Inline Package) switch
3. Removal of the battery and charger from the Sniffer PCB
4. Changes to the power supply module

The second CC1352R processor was added to the PCB to eliminate the need for a single processor to switch between the sub-1GHz ISM band and the 2.4GHz ISM band. This was mainly implemented to simplify the software development as a risk mitigation strategy. One of the CC1352R chips is termed the "Primary" and its job is to listen to BOB nodes. The other CC1352R is termed the "Relay" and its function is to pass on information to the other Sniffer nodes.

The DIP switch was added for manual addressing of the Sniffer nodes. Each node has its own address which is used in the communication between the nodes. Originally the plan was to assign the Sniffer address in the software. However, this would make reprogramming a number of Sniffer nodes cumbersome as the address would need to be changed in the code for each one before flashing. With the DIP switch, the address can be set once and the same code can be flashed to every Sniffer node quickly and efficiently.

The removal of the battery and charger from the Sniffer PCB was done for several reasons. The first was to reduce the number of items to be designed before the first revision could be ordered. This would give more time to work out issues in the first revision. The second was to allow the use of multiple types of battery solutions with the Sniffer. This adds flexibility for future users. The third was that with a separate battery pack, it would be easier to mount the batteries on the mechanical stack.

The change to the power supply was minor. Originally the plan was to use the battery output directly as the positive voltage rail for the PCB. The reason was that the CC1352R chips have an on-board regulator that is highly efficient. A regulator would only be needed to step down a 5V input into the valid range for the CC1352R.

This plan however had several flaws. First, the I/O voltage would not be constant but would vary with battery charge. This could cause issues in interfacing with the BOB node. Second, if the battery voltage got too low, several elements (such as the indicator LEDs) may begin to fail. Third, not using a regulator at the input makes it more likely that a user will accidentally damage the board by unknowingly supplying too high of a voltage to the board. For these reasons, a buck-boost regulator was added to the battery supply path to ensure a constant PCB supply voltage of 3.3V.

The first step in the Sniffer PCB design was to design the schematic. The schematic was broken into nine sections for easier design and more readability. Figure 20 below shows how the nine sections interact with one another. The schematic for each section is described in more detail below.

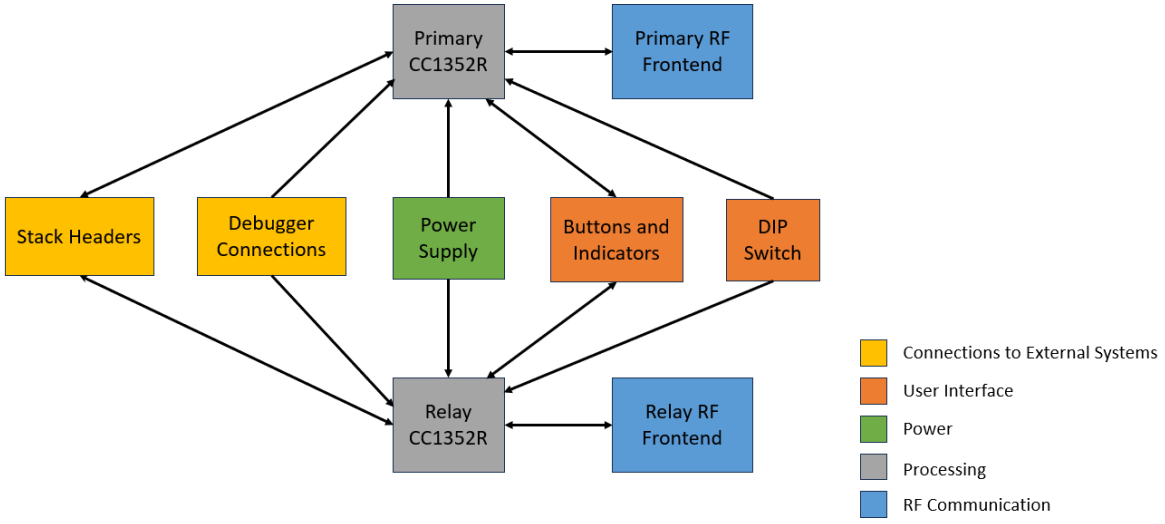


Figure 20: Sniffer PCB Schematic Sections and Relationships

Primary CC1352R Schematic:

Figure 21 shows the schematic for the Primary CC1352R section. The schematic is largely based off of reference designs and documentation available from Texas Instruments (TI).

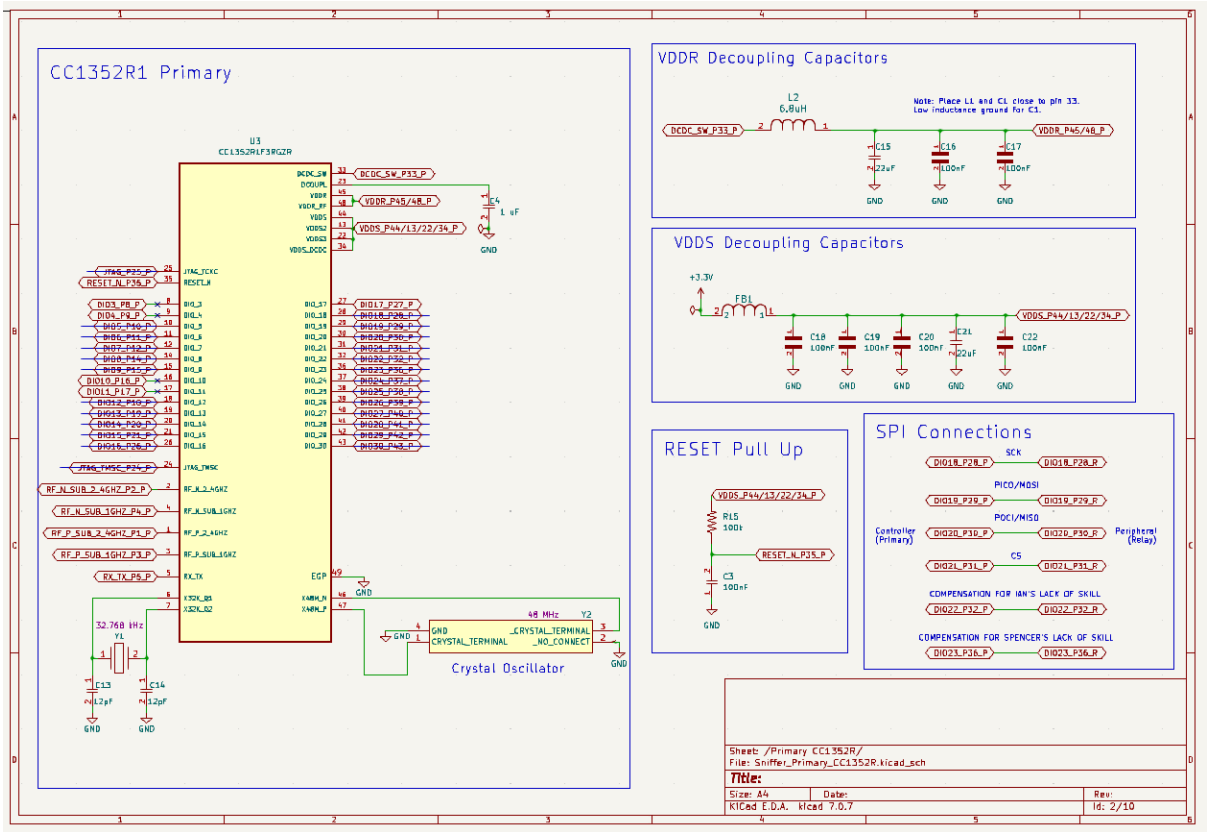


Figure 21: Primary CC1352R KiCad Schematic

There are several core element in the schematic. The first is the CC1352R chip.

The second is the decoupling capacitors for the power supplies. The decoupling capacitors eliminate noise on the supply line introduced by the switching in the CC1352R. The values and number of decoupling capacitors were based on TI recommendations. An inductor and ferrite bead are also added per TI recommendations. The ferrite bead suppresses noise signals to reduce radiated emmisions from the PCB.

The third core element is the oscillators. These provide the clock signals needed for the CC1352R to operate.

The fourth element is the pull-up resistor for the reset. This resistor pulls the reset pin up. If the pin is forced low then the CC1352R will reset.

The final element is the SPI (Serial Peripheral Interface) connections to the Relay CC1352R. These connections allow recieved information from the Primary to be transfered to the Relay to be sent down the Sniffer chain. There are the usual four SPI lines (SCK, PICO (formerly

MOSI), POCI (formerly MISO), and CS) as well as two additional lines to make the code development easier.

Relay CC1352R Schematic:

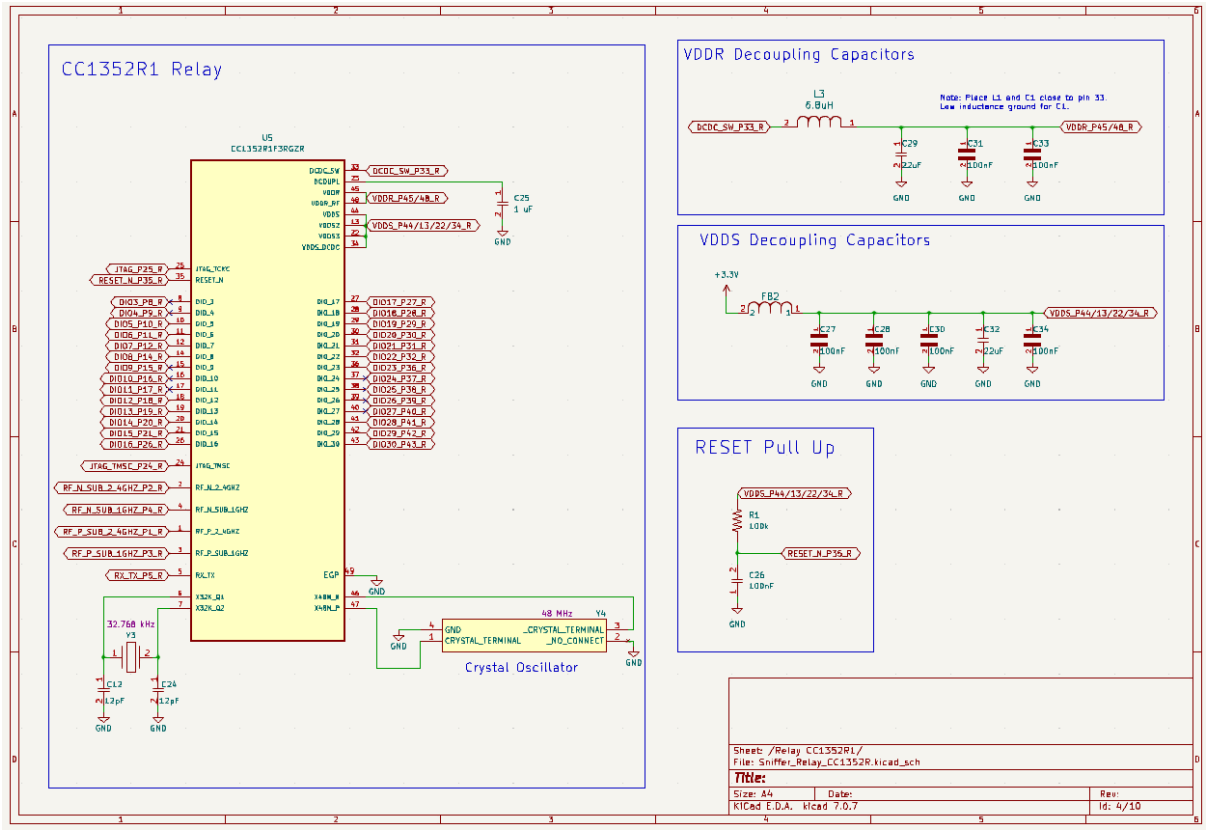


Figure 22: Relay CC1352R KiCad Schematic

Figure 22 shows the schematic for the Relay CC1352R section. The schematic is almost identical to the Primary CC1352R schematic. This is because the schematic is basically just the required support circuitry for the TI CC1352R. Note that the SPI connections are not included because they only need to be made once in the schematic.

Primary RF Frontend:

Figure 23 shows the schematic for the Primary RF frontend section. The CC1352R chip can output either unbalanced or balanced RF signals from the RF pins. The antenna requires an unbalanced input RF signal. Although it seems intuitive to use an unbalanced output from the CC1352R, a balanced output has been shown to provide better performance. ADD CITATION The balanced output signal then needs to be converted to an unbalanced signal

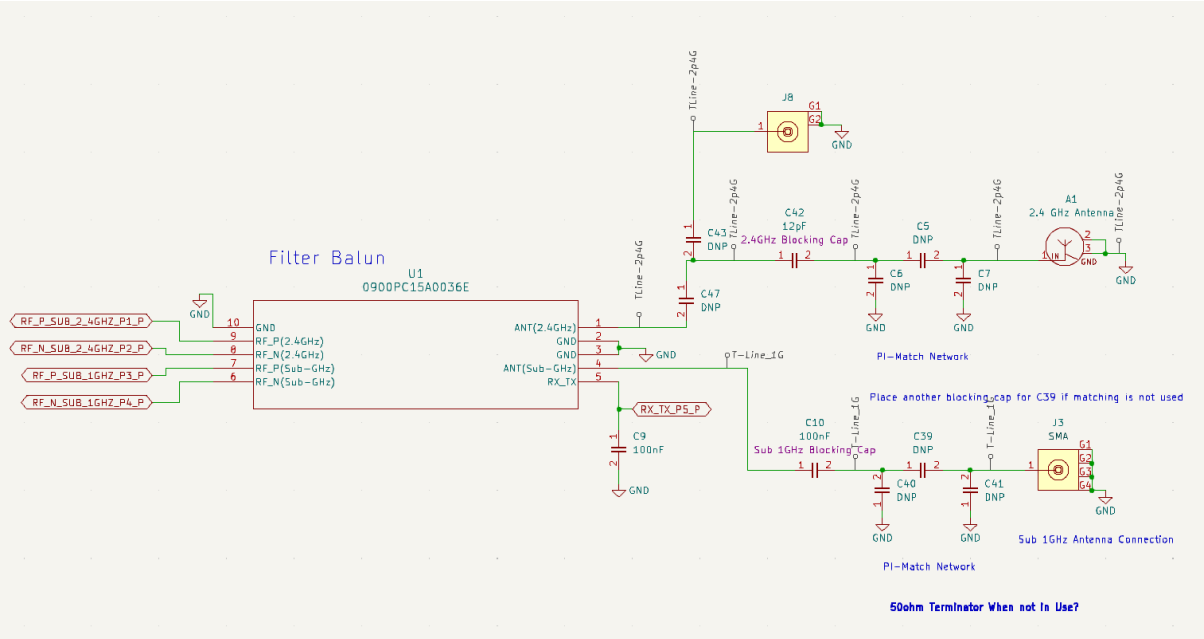


Figure 23: Primary CC1352R RF Frontend KiCad Schematic

to send and receive communications via the antenna. A circuit called a balun (standing for balanced to unbalanced) is needed to perform this operation.

The other elements needed in the frontend are a filter and a matching network. The filter suppresses unwanted signals which get picked up by the antenna. A matching network is used to eliminate impedance mismatches between circuits.

If RF waves are traveling and hit an impedance mismatch, then some of the signal is transmitted through the barrier formed by the mismatch and some is reflected back. This phenomenon is analogous to light hitting glass. If one looks through a window, they can see both the objects on the other side as well as faint reflections of the object on their same side. This is because, when the light waves hit the barrier between the air and the glass, some light is reflected back and some passes through.

Reflected waves reduce efficiency so matching networks are used to eliminate the impedance barrier. Typically, manufacturers design all components to have a 50 ohm impedance so the goal is usually to match to 50 ohms.

Conveniently, Johanson Technology offers a integrated passive component which implements the balun and the filter in one small package and is designed to be matched the CC1352R on the input and to 50 ohms on the output. This component (called the balun filter) was used to

simplify the RF design.

For the sub-1GHz channel, an off the shelf antenna was selected so an SMA connector is provided in the schematic for it to attach to. The SMA connector is matched to 50 ohms.

For the 2.4GHz channel, a PCB antenna design provided by TI was selected. This same antenna design is used in the development boards used by Dr. Duwe's research team. Using a PCB antenna saves money. A PCB antenna is not used for the sub-1GHz channel because, at the lower frequency, the antenna size would grow too large. An optional SMA connector is also included in the 2.4GHz chain. This is added so a spectrum analyzer can be attached for testing.

A Pi-matching network was added to both the 2.4GHz signal path and the sub-1GHz signal path. It is called a Pi-match network due to its shape mimicking the signal pi. The network is made using 0402 (imperial) sized components. The network was only anticipated to be needed for the 2.4GHz channel to match the PCB antenna. It was not hypothetically needed on the sub-1GHz chain because both the output of the balun filter and the SMA connector are already matched to 50 ohms. The Pi-match network is included anyway in case it is ever needed. The Pi-match network is used so that a number of different matching configurations can be used. Likely not every element would not be needed and some of the components would be left off the board.

The last parts of the schematic are the 100nF capacitor that supports the balun filter and the DC blocking capacitors. DC blocking capacitors keep any DC signal from reaching and damaging the CC1352R RF circuitry.

Relay RF Frontend:

Figure 24 shows the Relay RF frontend. The circuit is identical to the Primary RF frontend.

Power Supply:

Figure 25 shows the power supply schematic. The main element in the power supply circuit is a buck-boost regulator and its support circuitry. The buck-boost converter takes an input voltage ranging between 1.8V-5.5V and gives a 3.3V output. The input range of the converter allows it to be powered on the low end by two almost discharged, rechargeable AA batteries in series or on the high end by a 5V (typical USB voltage) input. This gives a wide range of flexibility.

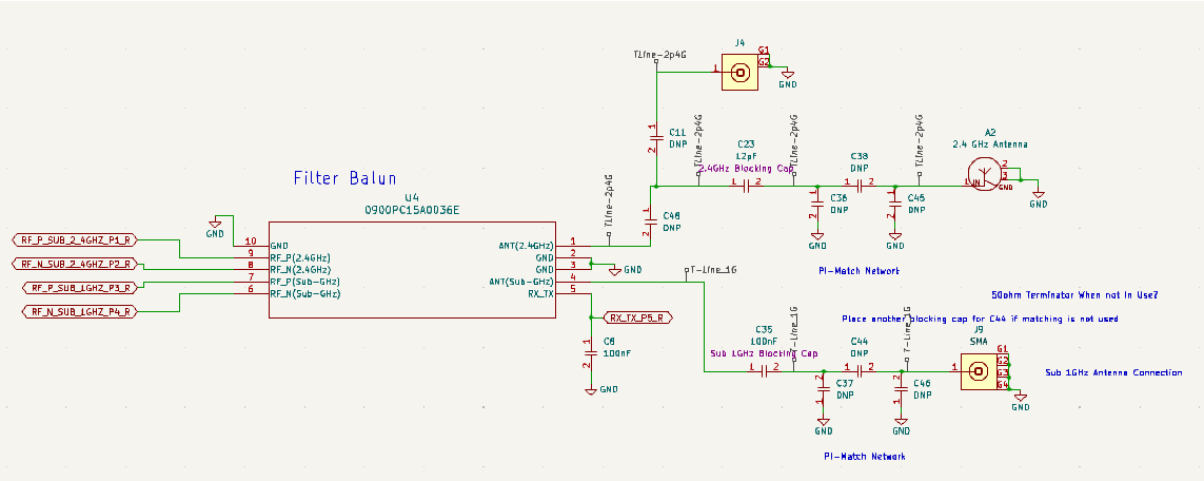


Figure 24: Relay CC1352R RF Frontend KiCad Schematic

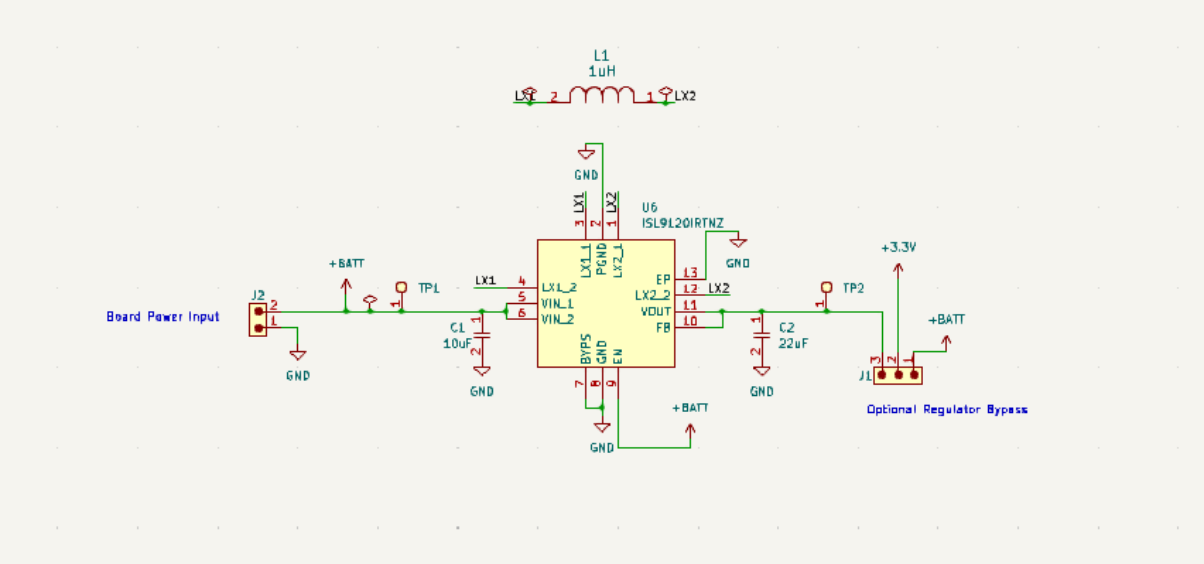


Figure 25: Power Supply KiCad Schematic

The regulator operates at a high efficiency (as high as 98%). The inductor and the capacitors are required for the regulator to operate and were chosen according to the specifications given in the regulator datasheet.

The small connector was chosen for the battery that is inexpensive and has pre-wired harnesses available at a low cost.

At the output of the regulator, a 3-pin header was added. This header allows the user to optionally connect the PCB power rail directly to the battery voltage and to bypass the

regulator. This allows the user to experiment with an ultra-low power mode or to bypass the regulator in case it has any issues.

Buttons and Indicators:

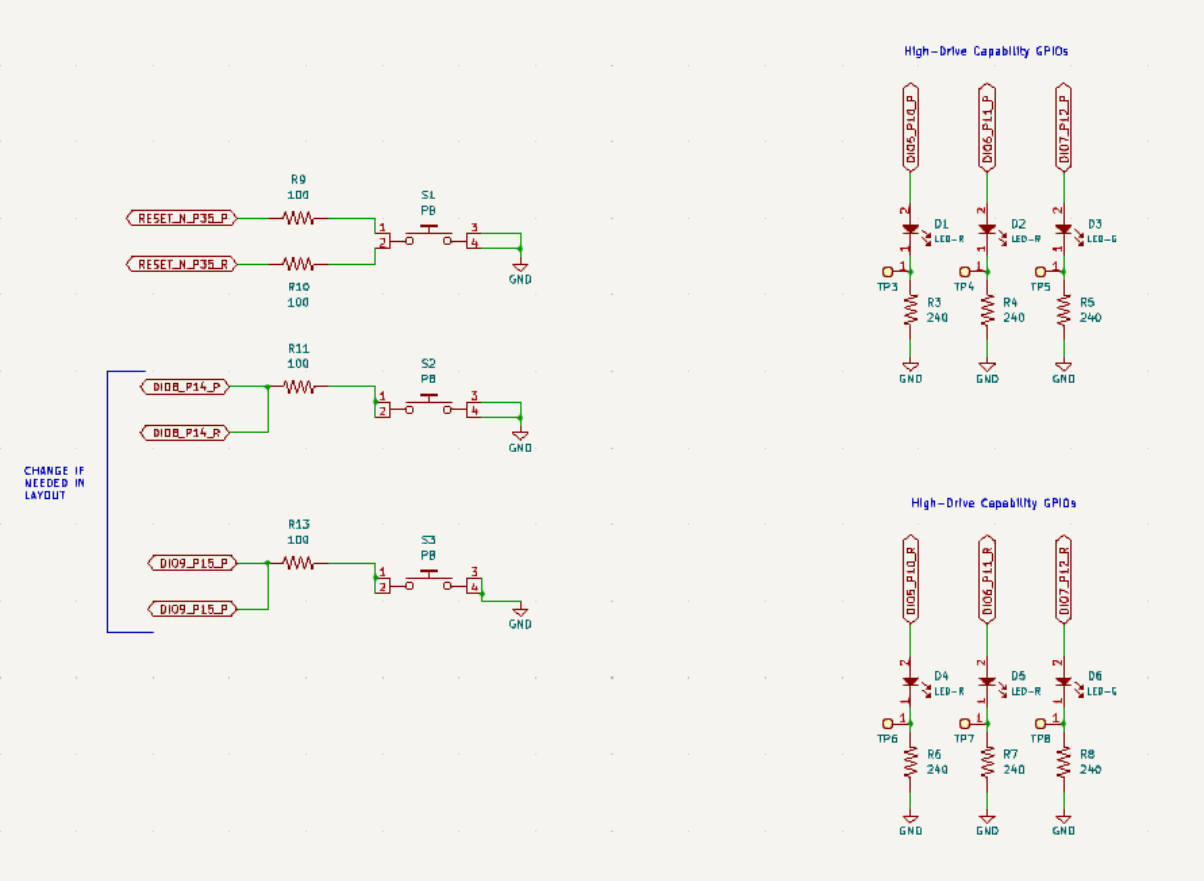


Figure 26: Buttons and Indicators KiCad Schematic

Figure 26 shows the schematic for the indicators and buttons section. Both the Primary and the Relay CC1352R have two red and one green LEDs connected to GPIO pins. Each LED has a current limiting resistor to ensure the LED maximum operating conditions are not violated. LEDs with low turn on voltages were selected so that the Sniffer PCB could optionally be used in an ultra-low power mode with the regulator bypassed. The GPIOs used to drive the LEDs were several of the high drive capable GPIOs which can source up to 8mA.

One of the buttons resets both CC1352R chips. The other two are optional buttons that can be used by the software developers for any purpose. The buttons are tied to ground with a 100 ohm current limiting resistor. This design is based on TI reference designs. The GPIO is

set to have an internal pull-up enabled (or external in the reset pin case). Then if the button is pressed, then the pin is pulled low.

Stack Headers:

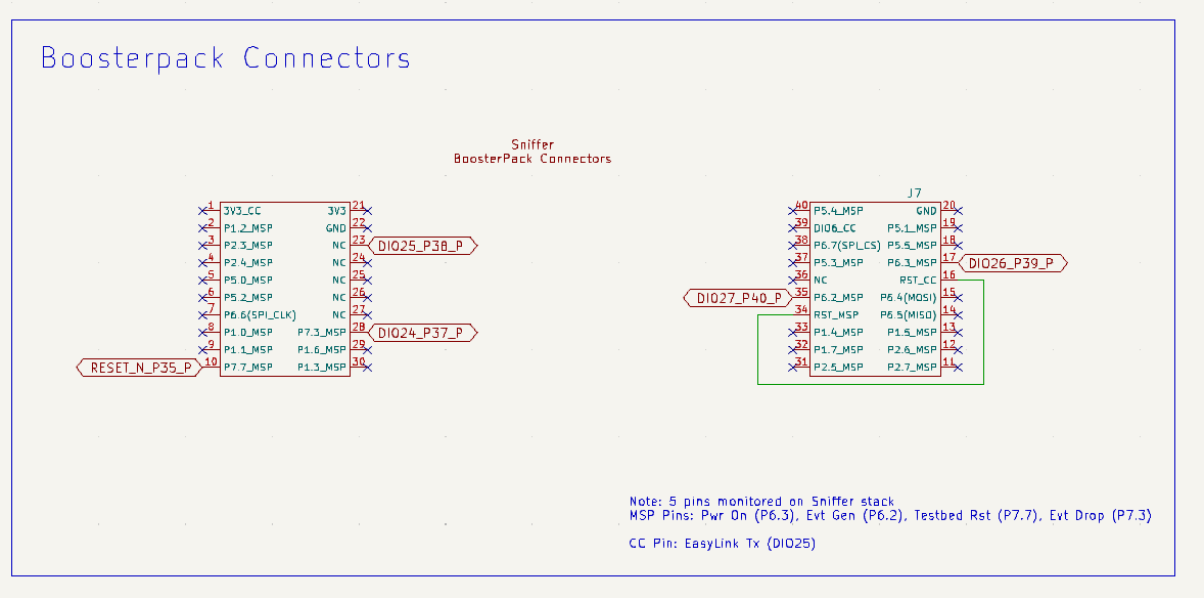


Figure 27: Stack Headers KiCad Schematic

Figure 27 shows the schematic for the stack headers section. Only a few connections are made to the BOB through the headers. These connections allow the BOB to communicate the occurrence of important events to the Sniffer during operation. Note that the power is not connected between the boards. This is to ensure isolation which is one of the requirements for the project.

Note that the grounds are also not connected. This was a mistake made in the first revision. A second revision will likely be ordered in the future to fix this mistake. For now the mistake will be fixed using jumper wires.

Debugger Connection:

Figure 28 shows the schematic for the programming section. The headers were selected to work with the standard JTAG connections that TI uses.

DIP Switch:

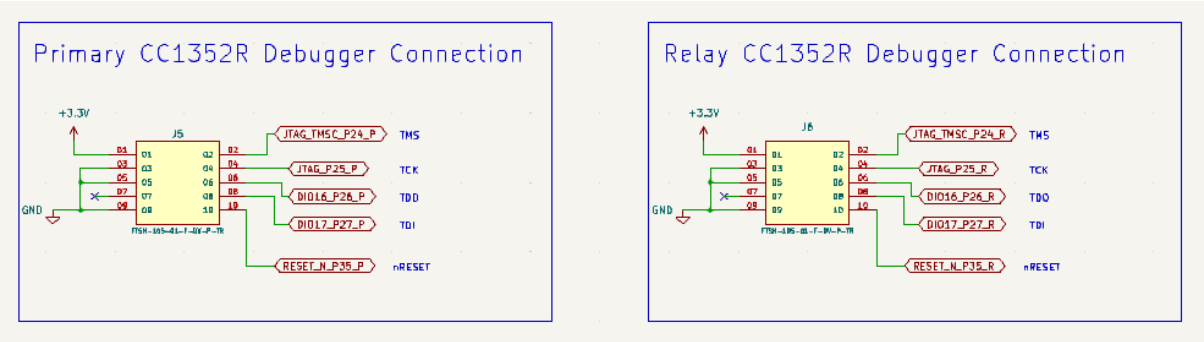


Figure 28: Programming Headers KiCad Schematic

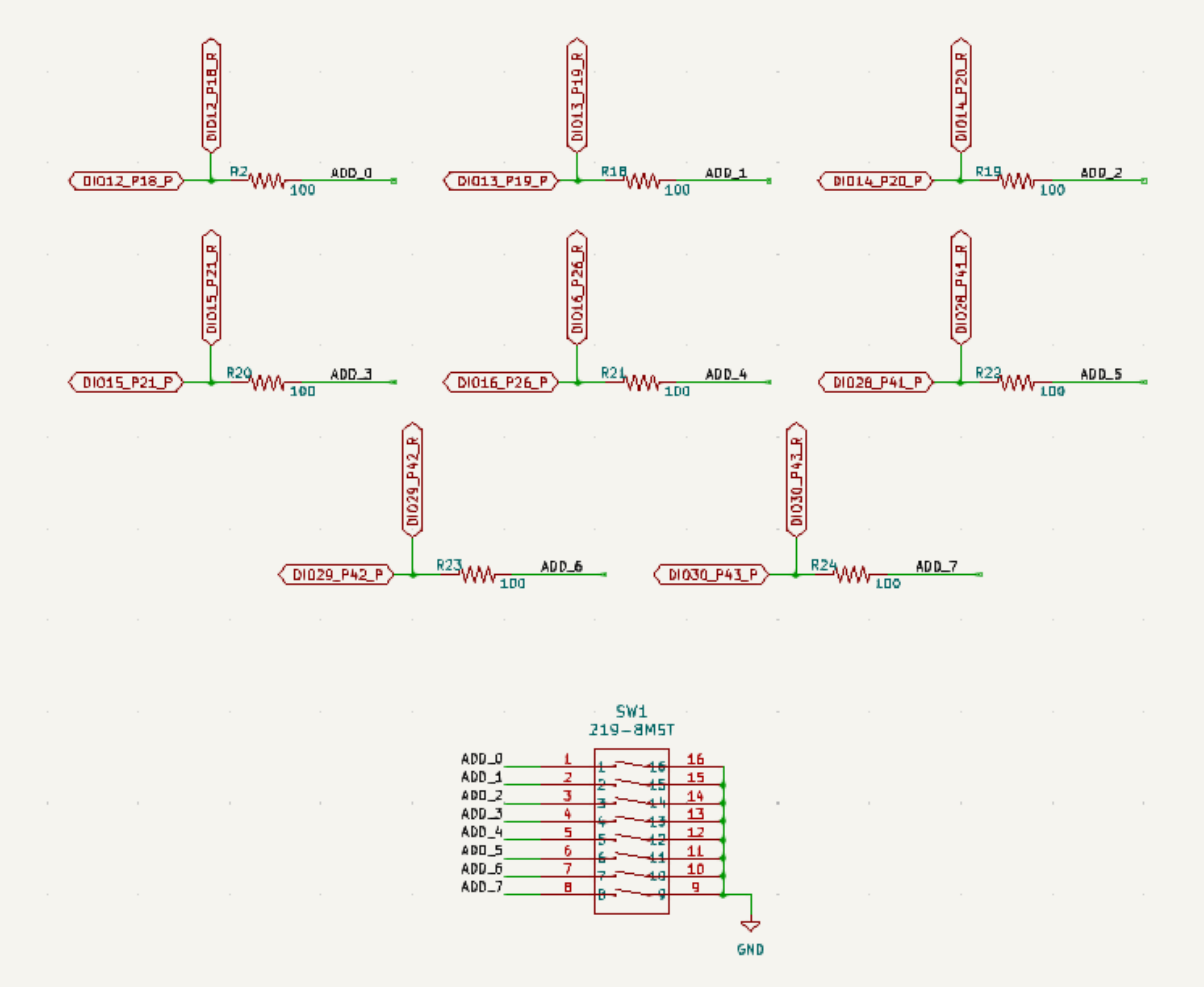


Figure 29: DIP Switch KiCad Schematic

Figure 29 shows the schematic for the DIP switch. The purpose of the DIP switch was discussed earlier. The switches are implemented in the same manner as the buttons were with the 100 ohm current limiting resistor.

Sniffer PCB Layout

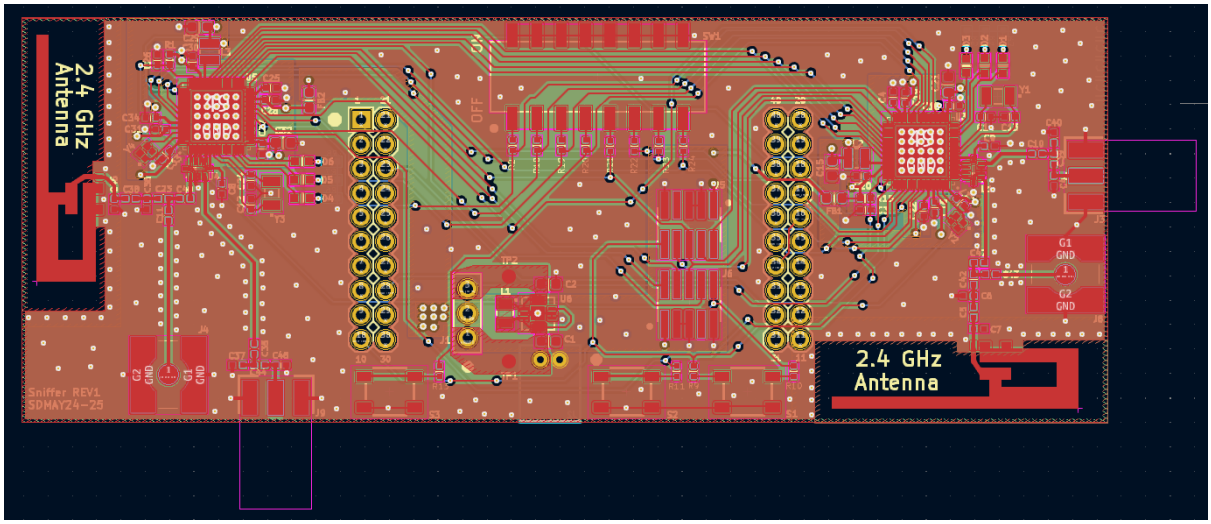


Figure 30: Sniffer PCB Layout

Figure 30 shows the layout of the Sniffer PCB. Much care was taken in the layout of the PCB to minimize negative effects.

In the RF sections, the trace widths were calculated to provide a reference impedance of 50 ohms to avoid any mismatches causing signal reflections. The matching and blocking components were placed close together to avoid having transmission line effects between them. Vias were placed frequently throughout the board to create low impedance paths to ground. The unused area of each layer was filled with a ground plane to provide shielding to reduce radiated emissions. In addition, no traces were routed under the RF sections to ensure high signal integrity.

Care was also taken in the power supply. Large power planes were used to reduce the inductance in the power paths. This was to reduce the affects of switching on the amount of noise in the power rail. Decoupling capacitors were placed as close to the chip pins as possible to eliminate inductance between them and the chip. Each decoupling capacitor was given its own ground via to ensure only a small current loop was formed to reduce radiated emissions.

To reduce cross-talk and increase signal integrity, effort was made to route traces in adjacent layers in an anti-parallel manner.

The antennas were placed outside the board to board headers so that they would not physi-

cally interfere with the antennas from the BOB node boards.

1GHz Transmission Line Calculations

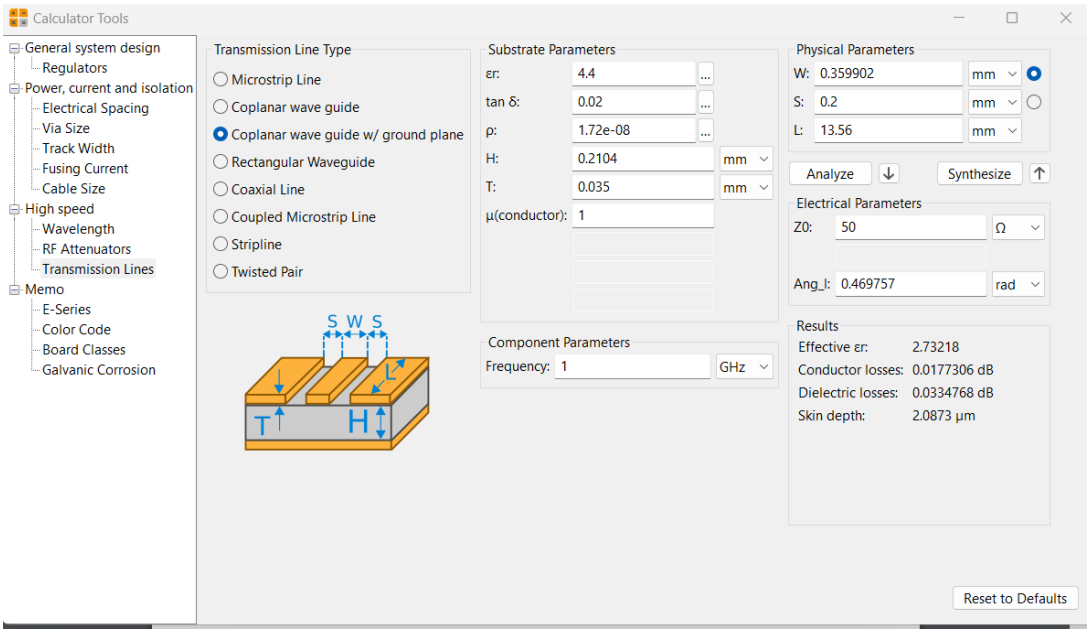


Figure 31: 1GHz Transmission Line Calculations

Figure 31 shows the calculations that were found for the co-planar wave transmission line for 1GHz antenna. Calculations were found using the built in tool to Kicad. Using the width, spacing, and length measurements of the trace we were able to approximate to as close to 50 ohms as possible.

2.4GHz Transmission Line Calculations

Figure 32 shows the calculations that were found for the co-planar wave transmission line for 2.4GHz antenna. Calculations were found using the built in tool to Kicad. Using the width, spacing, and length measurements of the trace we were able to approximate to as close to 50 ohms as possible.

5.1.2 Battery Pack

Our battery integration for the Sniffer test board is using two AA NiMH rechargeable batteries per Sniffer PCB. We use a battery case that holds two batteries in series and is wired to a

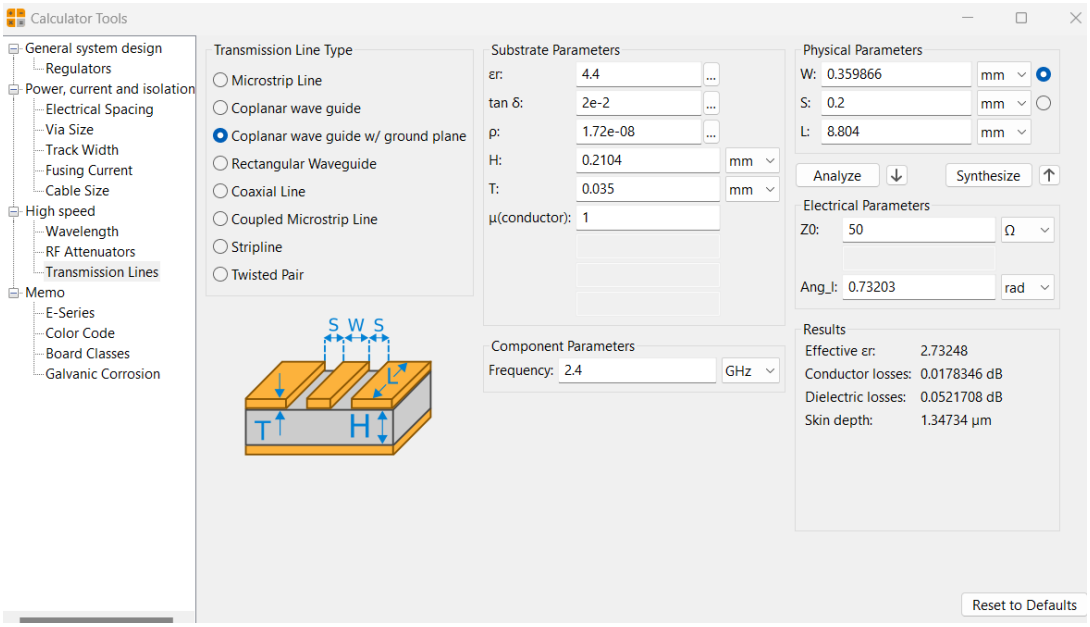


Figure 32: 2.4GHz Transmission Line Calculations

connector that will plug into the Sniffer PCB. The batteries that we have spec'd out are 1.2V with a standard charge of 2,000mAh. With two batteries connected in series that will put our operating voltage at 2.4V with a charge of 2,000mAh. It should be noted that the 2.4V does fit within the voltage range requirements of the CC1352 MCU which is 1.8 V to 3.8 V. The lower voltage was also chosen, as we it will work if disposable AA batteries are accidentally swapped in for future use by the research team or another senior design team. The disposable batteries operate at 1.4V, which when combined would offer 2.8V. Which is within a safe range for our 2 cell setup. A requirement for this project was an operation time of 1 week. With the power calculations that we ran during the 1st semester, we determined an average current draw of 7.906mA with our setup. These calculations were done using a tool provided by TI called EnergyTrace figure. 10

$$Charge(mAh) = I(mA) * Time(hours)$$

$$Time(hours) = Charge(mAh)/I(mA)$$

$$Time(hours) = 2{,}000(mAh)/7.906(mA)$$

$$Time(hours) = 253(hours) = 10.5days$$

You can see from the above calculations that we should reach our runtime requirements

and should have a 3.5 day buffer of extra time. After one week time, the AA batteries can be removed and replaced with a charged set. Then, the low-charged batteries can be placed in the chargers for the next time they will be swapped out.

Components:

- Amazon Basics Rechargeable AA NiMH 2000mAh Batteries
- PowerRowl 16 bay and 8 bay chargers
- Keystone Battery Holder AA 2 Cell 6" Leads 2463

5.1.3 Cost

The total cost of our design will be highlighted below. This includes the cost from the initial Breakout Board, MSP Simplified REV1 and REV2, Sniffer REV1, Batteries and Chargers, additional part order, and mechanical design. This put our total cost per node at roughly a hundred dollars. This does not include the cost of the CC1352 radiotag board and harvester board that the research team is implementing or covering the cost of. It should be noted that the battery cost was covered by the research team, and was not an expense of the senior design project.

Senior Design Cost Breakdown		
	Cost Per Node	Overall Cost
Breakout Board	-	$37.83
MSP_Simplified	~$31	$234.60
Sniffer Board	~$56	$611.88
Batteries and Chargers	~$12	$112.05
Additional parts	-	$29.29
Mechanical Design	~$4	$34.60
Total	~$103	$1060.25

Figure 33: Senior Design Implementation Cost

5.2 Software

5.2.1 Sink Node

The Sink node is set up to take input from the Host via UART and transfer that data to the Sniffer nodes via the Sniffer Band (2.4 GHz). It sets the packet type based on the input from the host and transmits it to either a specific Sniffer or all Sniffers. It also receives data from the Sniffer Band (2.4 GHz) and sends it back to the Host via the same UART connection.

5.2.2 Sniffer Network

The Sniffer network is done on the 2.4 GHz band. Data can originate from the Sink node and be sent out to the Sniffers, such as in the case of test configurations, test commands like start and stop, or acknowledgments. Data can also be sent from the Sniffer nodes back to the Sink node in the case of sending testing data "sniffed" from the BOB back to be logged by the host.

Shown in figure 34 the network is made up of rows of Sniffer that pass data back to the sink. The depth of the row is set by the networkDepth parameter in the radio packet. The Sniffers pass the packet deeper and deep to the network depth node. If the network depth is 0xFF then the packet is destined for all sniffers and the Sniffer will always broadcast it deeper. The row is selected by the Sink. There are up to 8 rows or 0x00 to 0x08. Each next depth is the prior address in the row plus 8. The Sniffer filters out messages without its address. Depth Parameter can be calculated as such: Depth = floor(Destination Address/8)+1 Transmitting to the sink sets the depth parameter to 0xFE, then each node takes its address and subtracts 8. If the new address is less than 0, it sets its destination to 0xFE.The initial node to transmit to can be calculated from the Sink by doing destination % 8. Each destination is 1 byte, meaning a maximum of 253 destinations in the network due to the Sink and the 0xFF for all notation(could be increased, but due to address filtering in the BOBs this would be unnecessary unless that BOB code changes

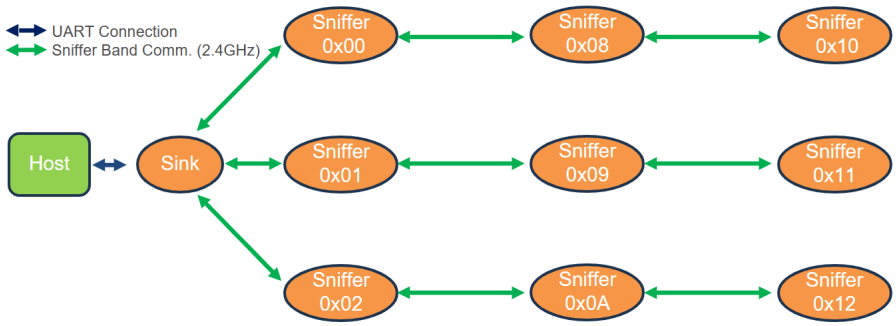


Figure 34: Sniffer Network Design

5.2.3 Sniffer Design

The Sniffer design is shown in figure 35 and is made up of two CC1352s. This design allows for one to monitor the BOB band (Sub-1-GHz) and the other to transmit data amount Sniffers on the Sniffer band (2.4 GHz). The two CC1352s communicate via SPI described below. The Primary handles all interfacing with the BOB, whereas the Relay handles Radio communications with the other Sniffers via the Sniffer network described above.

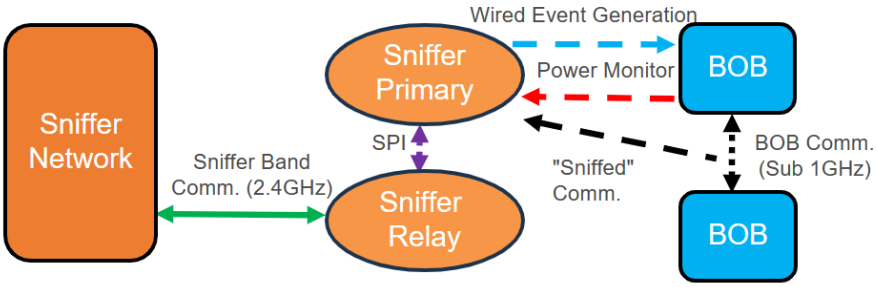


Figure 35: Sniffer Design

5.2.4 Sniffer Relay

As mentioned in prior sections, the Sniffer Relay handles communication on the Sniffer Band. It relays messages back to the Sink and deeper into the network. The Sniffer relay handles this by having two queues: one that holds messages received that need to be rebroadcasted, and the other one messages to be sent back to the Sink, which is data from the primary CC1352. These queues are together protected by a counting semaphore. It only increases the queue size once things are enqueued. It dequeues whichever queue is bigger at a given time to

decrease the chance of packet dropping due to filling the queue. In the event that a packet is dropped, it transmits a packet stating that a packet is dropped so the user knows and can respond accordingly. Another queue is used to house packets that need to be sent to the Sniffer Primary to set test configurations. There a multiple tasks or threads set to accomplish these various jobs.

5.2.5 Sniffer Primary

The Sniffer Primary handles all interactions with the BOB node sensing packet movement, lifecycles, and test configurations. Similar to the Sniffer Relay, it has queues to send data over SPI from various sources. Once again, there is a semaphore to protect these queues. Similarly, there are multiple threads to handle all these tasks.

5.2.6 SPI

A three-wire SPI protocol sends data between the Primary and Relay CC1352. The Primary acts as the SPI master, and the Relay acts as the SPI slave.

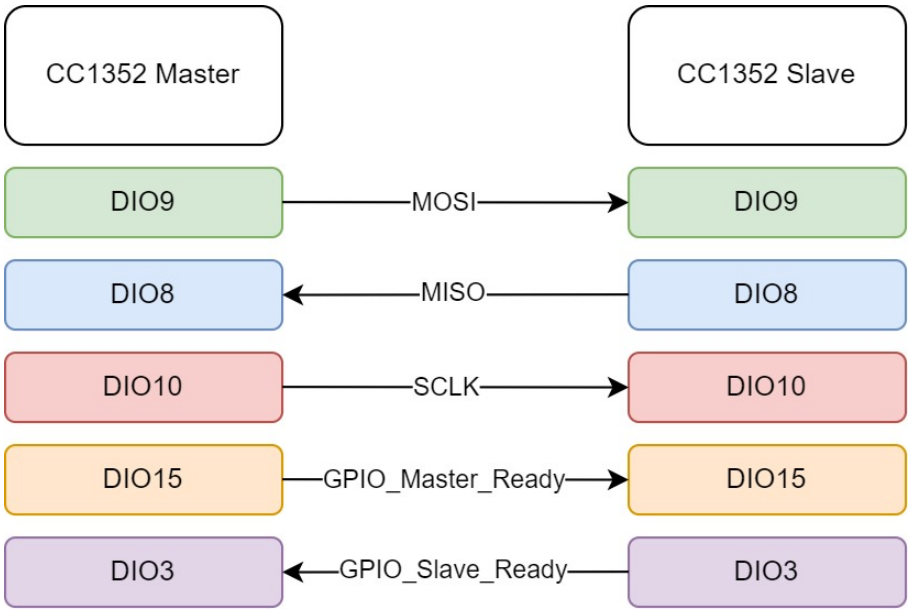


Figure 36: SPI Conncetion

In addition to the pins used for SPI, two GPIO pins are used to sync the software on both

CC1352s. The GPIO pins are named CONFIG_SPI_SLAVE_READY and CONFIG_SPI_MASTER_READY (the last two connections shown in Figure 36). The CONFIG_SPI_MASTER_READY goes high then waits for the CONFIG_SPI_SLAVE_READY to go high before opening the SPI handle. Three semaphores are used to synchronize a SPI transaction. The spiTaskSem semaphore pends until a radio packet is received. The masterSem semaphore waits for the slave to be ready which is toggled by the changing in the GPIO CONFIG_SPI_SLAVE_READY pin. The SlaveRxSem semaphore pends until the SPI transfer is complete. Figure 37 shows the steps taken in software for SPI execution. The section of the table in yellow is repeated in a while loop to keep the SPI handle open and allow for SPI transactions to transfer continuously as data packets are received via sub 1GHz radio.

SPI Order of Operations	
SPIMaster	**SPISlave**
Configure GPIO (CONFIG_SPI_MASTER_READY) as output	Configure GPIO (CONFIG_SPI_SLAVE_READY) as output
Configure GPIO(CONFIG_SPI_SLAVE_READY) as input	Configure GPIO(CONFIG_SPI_MASTER_READY) as input
Set GPIO (CONFIG_SPI_MASTER_READY) = 1	Set GPIO (CONFIG_SPI_SLAVE_READY) = 1
Wait for GPIO(CONFIG_SPI_SLAVE_READY) = 1	Wait for GPIO(CONFIG_SPI_MASTER_READY) = 1
Configure GPIO(CONFIG_SPI_SLAVE_READY) as interrupt	Create Semaphore (SlaveRxSem)
Create semaphore (masterSem)	Wait for GPIO(CONFIG_SPI_MASTER_READY) = 0
Create semaphore (spiTaskSem)	Initalize spi handle (slaveSpiHandle)
Initalize SPI handle (masterSpiHandle)	While(1){
While(1){	Set transmit buffer(transaction.txBuf) = masterTxBuffer
Wait for semphore(spiTaskSem) ** Recieved radio packet	Set receive buffer(transaction.rxBuf) = masterRxBuffer
Set GPIO (CONFIG_SPI_MASTER_READY) = 0	Perform SPI transfer (SPI_transfer())
Set data buffer(masterTxBuffer) = dequeued packet	Set GPIO(CONFIG_SPI_SLAVE_READY) = 0
Pend semaphore (masterSem) **Slave ready for transfer	Wait for semaphore (SlaveRxSem) **Slave completed transfer
Set transmit buffer(transaction.txBuf) = masterTxBuffer	Set GPIO(CONFIG_SPI_SLAVE_READY) = 1
Set receive buffer(transaction.rxBuf) = masterRxBuffer	}
Perform SPI transfer (SPI_transfer())	
}	

Figure 37: Order of Operations in Code for SPI

5.3 Mechanical

Mechanically, we have all the components mounted to the ceiling through a laser-cut plate of cast acrylic with some ceiling hangers. The mounting holes of the radio frequency energy harvester are put in the plate and connected with half in a standoff. The plate also has a spot and holes to screw the battery pack into. The rest of the stack mounts to the power harvester via their pins which provide enough force to mount all the portions together. The Stackup can be seen in Figure 38. The alternative holes are to mount the simplified MSP-430 to the plate as another option. The dimensions are shown in figure 39.

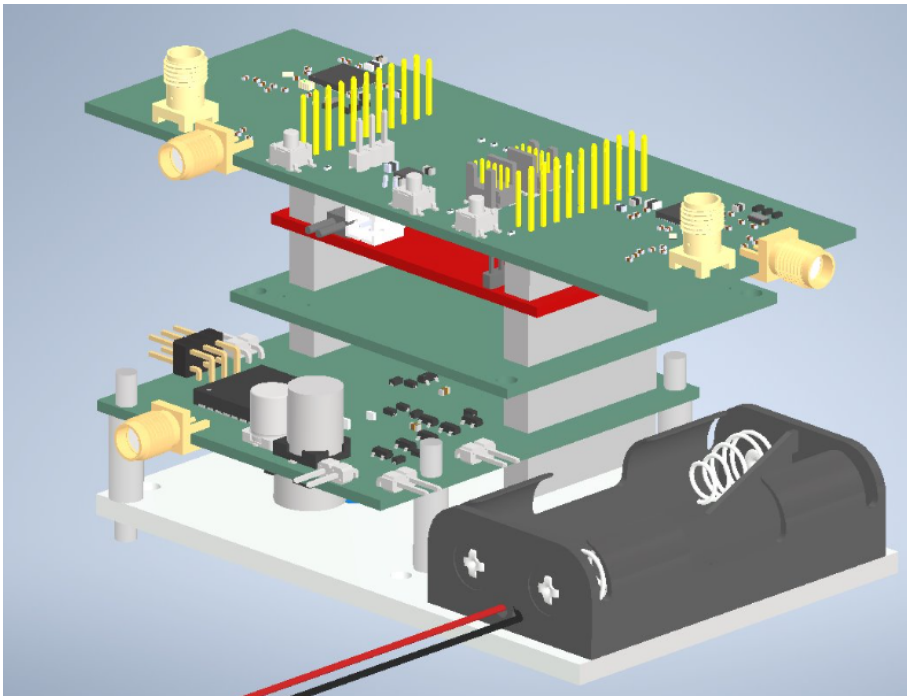


Figure 38: Sniffer Mechanical Stackup

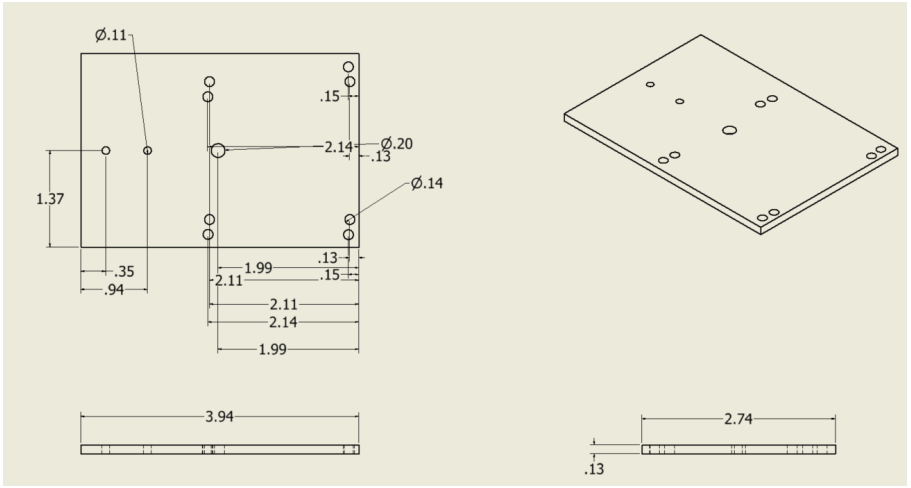


Figure 39: Sniffer Mechanical Dimensions

As per the client's request, we added a logo to the second revision of the test bed mounting plate and gave it the name of the Reel Testbed due to the fishing naming convention they have maintained throughout their research (figure 40.

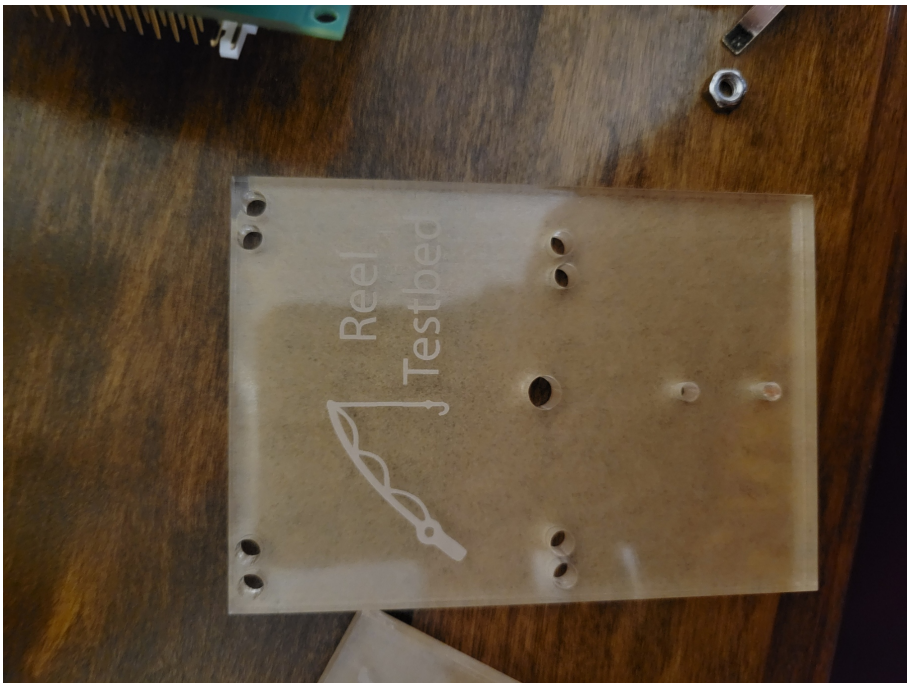


Figure 40: Reel Testbed

6 PROFESSIONALISM

6.1 Areas of Responsibility

6.2 Project Specific Professional Responsibility Areas

Work Competence - High. Work competence is the most important IEEE standard for our project. Given that Senior design is limited by assignment deadlines and graduation dates, it is important that we complete tasks in a timely manner. The success of the project is dependent on a large code base working with hardware over a scalable communication network, so it is important that our work is high quality to reduce the chances of errors in the design.

Financial Responsibility - Low. Based on the IEEE standard, there is no financial responsibility for this project, but the BOBs being tested by our design team may someday be used as a means of data collection, in which case the IEEE standard for financial responsibility would apply.

Areas of Responsibility	IEEE Definition	NSPE vs. IEEE Comparison
Work Competence	Conduct tasks with excellence, honesty, adherence to schedules, and a demonstration of professional expertise.	IEEE defintion builds on the NSPE Canon defintion for work competence by including quality and timliness as important factors.
Financial Responsibility	Provide products and services that offer tangible value and are reasonably priced.	NSPE focuses more on the importnace of loyality to an employer while IEEE focus is more on the resonable price of the product.
Communication Honesty	Present work with honesty, devoid of deception, and in a manner comprehensible to stakeholders.	NSPE states that only truthful statements should be made to the public, while IEEE defintion emphaszies the understanding of the stakeholders specifically.
Health Safety, Well Being	Reduce hazards to the safety, health, and welfare of stakeholders.	NSPE states the health and safety of the public is paramount, while IEEE focuses on the health and safety of the stakeholders.
Property Ownership	Show regard for the property, ideas, and information belonging to clients and others.	NSPE uses the same defintion as the one used for financial responsibility claiming the importance of loyalty to the employer. IEEE defintion is more specifc in th meaning of property.
Sustainability	Safeguard the environment and preserve natural resources both on a local and global scale.	The "Sustainability" area of responsiblity does not exist for NSPE.
Social Responsibility	Create goods and services that contribute positively to society and local communities.	Both defintions are similar, but NSPE includes the statment that things should be done lawfully with the end goal of making the profession more useful and improving its reputation.

Table 20: Areas of Responsibility

Communication Honesty - High. It is important that teammates are honest with each other and the clients of the project. Honesty will allow for faster progress as problems in the deign process will be clear making it easier to find solutions.

Health Safety, Well Being -N/A. There are not any direct safety concerns for anyone involved in the project.

Property Ownership -Medium. Throughout the project team members will use circuit boards

and software provided by the client. Team members should treat the equipment used with respect and avoid damaging the equipment.

Sustainability - N/A. There are no direct environmental threats from the project. The testing area will be contained to a single room, and should not have any affect on the environment.

Social Responsibility - N/A. The senior design project does not produce a good or service, so by the IEEE definition for social responsibility the standard is not applicable. It is important to note that the devices we are testing (the BOB communication network) could potentially become a product that has an affect on a community, but that is beyond the scope of the senior design project.

6.3 Most Applicable Professional Responsibility Area

The most applicable area of responsibility is work competence. Considering the constraints imposed by assignment deadlines and graduation dates for Senior design, it is crucial to ensure timely completion of tasks. The project's success relies on a substantial code base interacting with hardware across a scalable communication network, underscoring the significance of maintaining high-quality work to minimize the risk of design errors.

7 Closing Material

7.1 Conclusion

This year of work has resulted in the completion of an initial revision of the Sniffer code, three PCB designs with two revisions on one of them, and a mechanical interface for the testbed. on the hardware side we have learned a lot of radio frequency circuits and requirements. On the software, we learned about multi-threading microcontrollers, embedded debugging, semaphores, and Real-Time Operating Systems. We implemented a highly complex system with multiple hardware, software, and mechanical portions. There were portions that could have used more polishing, testing, and features, but our team is happy with what we accomplished and what we learned over the senior design experience.

7.2 Future Work

Though we achieved a lot over the course of the year, there were some features we did not complete and will be left for future teams.

7.2.1 Hardware

On the hardware side we were able to get the PCB ordered and tested, however there were some aspects of the design that we would update in a future revision of the board. Some of these include usability of the board.

While our board is functionally working, their are some "janks" that were needed to be added that make the board less of a viable product. One of these include the grounding. In attempt to meet the requirement of power isolation between the Sniffer testbed and the rest of the Bob node, we did not connect the power pin from the Bob node to the Sniffer power rail. This is because the Sniffer node is going to be powered by our attached battery pack. However, one mistake that was made was also isolating the grounds from one another. This is problematic as the grounds don't match between the Sniffer and the rest of the stack, so for GPIOs it doesn't have a common ground reference, and therefore is essentially a floating pin, which is a problem when programming. To resolve this problem we have soldered a wire going from the GND pin on the board to board connector and a GND pin on the Sniffer PCB. This will then connect the two grounds. This is a temporary fix, and would be implemented in the layout in a future revision.

Another fix that we would make with the hardware includes usability of the board. This includes things like labeling. For example, the RST button was not clearly labeled as a RST and so while we knew what it was from the schematic, it wasn't an easy reference for the Software team while using the board. Another labeling feature to include in future revisions would be labeling of the JTAG pins, and perhaps including a pin header with a latch, that way it can't be plugged into the wrong direction. This will make things easier from a programming perspective, so you don't need to have the schematic up in order to program the board.

7.2.2 Software

On the software side of things, we got the base functionality done and were left with adding features and stress testing. The main feature we needed to add was the logic for sending configurations to the BOB. The logic is there to send it from the Host to the Sniffer through the Sink, but sending it to the BOB needs to be implemented. An additional feature that needed to be added was the logic to have a relay cc1352 or slave of SPI be able to initiate transmissions to send data specifically for configuring tests. Another important feature to be added would be the event generation and recording of BOB life cycles. Much of this code could come from the original Sniffer Code in Duwe's repository.

One less crucial feature to the Sniffer is the end test logic to be added to the Sniffer primary to stop sending data back after the test has ended, allowing queues to empty. The next feature would be adding the code to have the dipswitches on the sniffer set the Sniffer number. Finally, adding code to continuously update the time of the program to maintain a consistent clock between the Sink and all of the Sniffers needs to be added.

The final section of work would be doing more extensive testing. Due to our timeline, we were never able to test this with real BOBs, so ensuring everything interfaces with it correctly needs to be tested. On top of that stress testing to find out data rates and the design robustness to bursty data is needed. To help with this building up upon our Faux BOB would be beneficial. Adding more in-depth code to have the Faux BOB mock the life cycle of the BOB better would be required, and building up the versatility to send more realistic and random packets would ensure more robust packets.

The final piece of work for this project would be implementing this testbed with a single cc1352. Implementing this would need to involve using both radios at the same time or doing band switching as described in Appendix II, section 9.2.

REFERENCES

[1] Texas Instruments, "Launch your design start@ ti launchpadtm sensortag kit with simplelinktm wireless mcu." https://www.ti.com/lit/ug/swau127a/swau127a.pdf?ts=1701604662638&ref_url=https%253A%252F%252Fwww.ti.com%252Ftool%252FLPSTK-CC1352R, 2023. Accessed: Dec. 03, 2023.

[2] "Application note cc13xx/cc26xx hardware configuration and pcb design considerations," Sep 2022.

[3] R. Feranec, "Pcb antenna - how to design, measure and tune," Aug 2023.

[4] M. Salov, "Application note antenna impedance measurement and matching," Mar 2022.

[5] Sep 2019.

8 APPENDIX I (OPERATION MANUAL)

8.1 Setting up Coding Environment

This section walks through setting up the coding environment step by step.

8.1.1 Installing Code Composer Studio (CCS) and Dependencies

The program used for the bulk of the work on the Sniffer Nodes as well as the BOB Nodes is CCS, TI's proprietary C based IDE for their micro-controllers. Navigate to TI's download page for CCS, and make sure to grab the CCSTUDIO rather than CCSUDIO-THEIA. The newest version should theoretically be fine, but the bulk of our work was done on version 12.6. When prompted, a full install for all devices will be the most versatile and fool-proof, otherwise selecting the specific MCUs wanted is also an option for saving drive space. Otherwise, the installation wizard is self-explanatory.

The next tool needing to be installed is the SimpleLink SDK platform. Download and run the SIMPLELINK-CC13X2-26X2-SDK installer which can be found on TI's website. Once this program has been run, it will add in the library files to the location specified by the user, or at the C:/ti folder if left default. Next, open CCS, and open the *Preferences* tab from within the *Window* button in the top ribbon. From there, open the *Code Composer Studio* dropdown, and finally open the *Products* window. You should be greeted by a screen that looks like **Figure 41**.

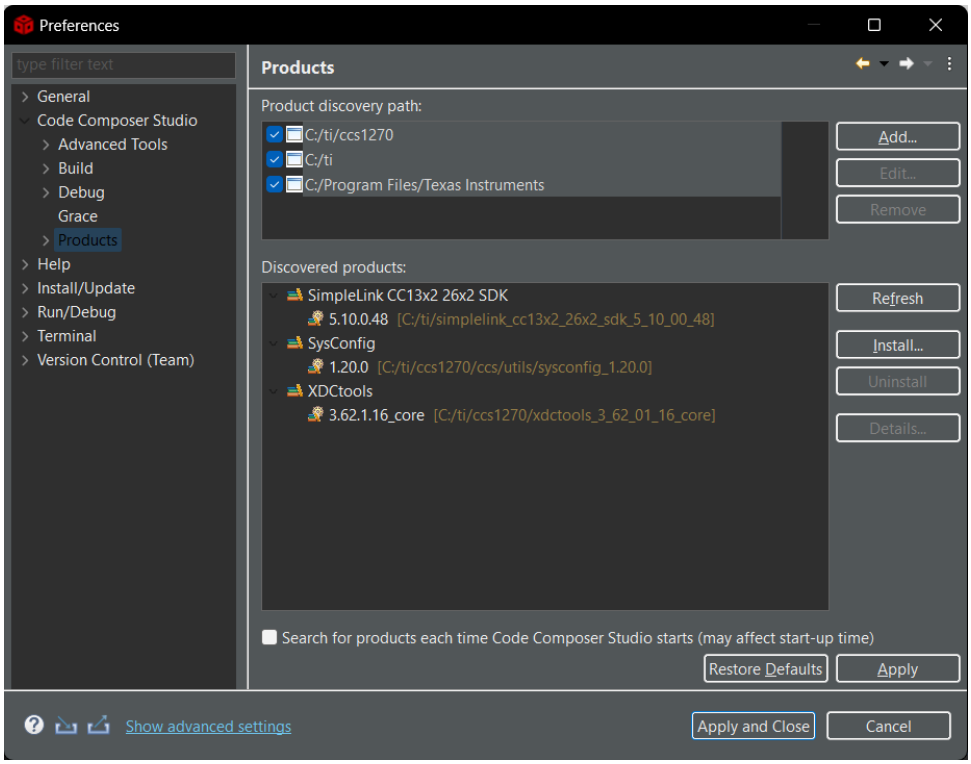


Figure 41: CCS Products Window

From here, after pressing the *Refresh* button, you should see the SimpleLink product you just installed, as well as a SysConfig product and XDCTools Product which should have been installed into the CCS folder when installed. These products should all show up immediately, but if not they can be found by clicking the *Add* button in the top section and adding the paths they are found within. Once the products are all there you can press *Apply and Close*.

Next, the projects can be downloaded from the GIT repository at

BOB Node Git Repository

If more help is needed with GIT, there are many helpful repositories online which can be used

to help.

Once they have been downloaded or cloned, they can be added to CCS using *Project -> Import CCS Project* and navigating to the local location.

8.2 Setting up and Programming Boards

This section walks through how to program all the boards in a configuration like the one in our demo. This can change with the number of Faux BOB nodes set up with the Sniffer node.

8.2.1 Setting up Faux BOB

To program the Faux BOB you need a CC1352 we used a Sensor tag but any of them would work. Ensure the active Debug program in CCS is the Faux BOB project. To program the Sensor tag, a launch pad is needed. To complete this program, plug a JTAG connected into the two JTAG ports circled in figure 42. If an LED starts flashing, it gets programmed correctly. Power this board with a power supply of 3.3V to the VDD pin after disconnecting the launchpad.

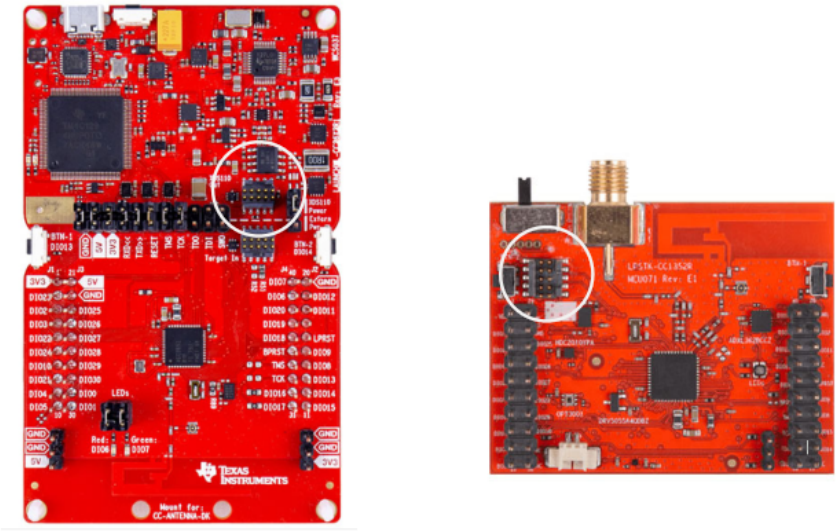


Figure 42: JTAG connection to program sensortag

8.2.2 Setting up Sniffer

To program the Sniffer, use the Launchpad again. Open up CC1352 Primary project in CCS. This time, you will be programming our custom CC1352 boards. We will start with programming the primary CC1352. To do so, use the same JTAG port and plug it into the CC1352 primary JTAG port marked in figure 43. It is essential that the Key of the JTAG connector faces out on the board.

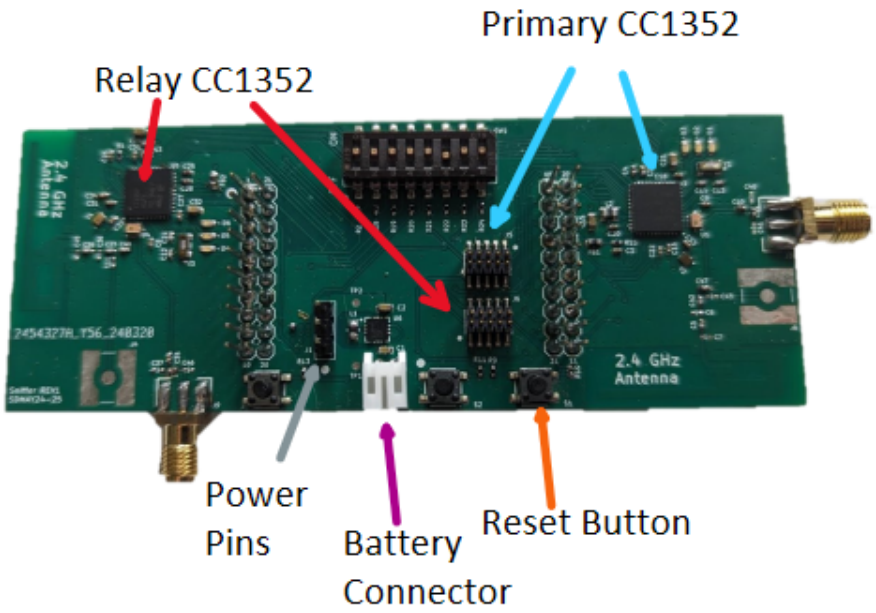


Figure 43: Important Sniffer Parts

Next, we will program the CC1352 Relay. The actual programming is similar to the primary, but first, we need to set the configuration in the project for the node address. This is done in two places in the sniffer.syscfg file under EasyLink, Recieve, Address filter and in config.h under NODE_ADDRESS in includes file shown in figure 44. This sets up the address filtering to ensure every Sniffer does not receive every packet. For this to work properly ensure that you build up each row of Sniffers fully so every packet has a path back. For instance, if you have a Sniffer of 0x1E, you need a 0x17, 0x0E, and 0x07. Build up the row for packet transfers as described in the Design and Implementation section. As explained in future work, hopefully, this can be configured with the Dipswitch. Once again, when programming, use the Relay CC1352's JTAG marked in figure 43. The JTAG key should again face toward the outside of the

board.

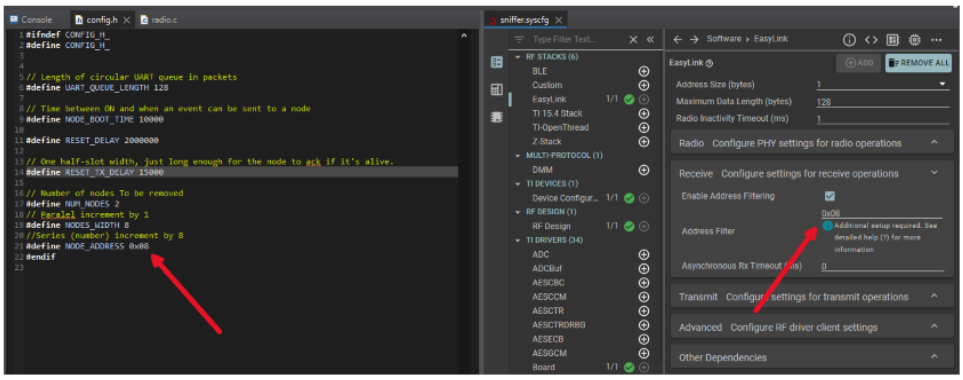


Figure 44: Relay Address Filtering Setup

Once both CC1352s are programmed it can can be powered via a power supply on the power pins or with a battery. They should be reset to ensure SPI does its handshake correctly. When the relay receives packets from the primary or more relays, it should flash an LED very quickly, showing it works properly.

8.2.3 Setting up Sink

Finally, we need to set up the Sink. You should use a Launchpad for this. Program the board with the CC1352_Sink project. Open up the Python folder with VSCode and run it with whichever port your board is connected to for UART, and you should see it do the UART handshake and receive data from the network.

8.3 Problems that came up

While integrating the custom hardware with our software, we ran into a couple of challenging debug problems. These bugs are documented here to ensure they are not experienced in further use.

8.3.1 EasyLink Init Breaking

The first issue we ran into was with the Easylink Init function. The board would program completely fine, and we could step through the code to find and interact with GPIO pins, but as soon as we hit the Easylink_init() function, the program broke. It still ran but did not continue execution. After digging around registers, we saw some timer decrements and chose to probe the oscillators. We found that the 48MHz oscillator was not soldered correctly, and fixing that solved our issue.

8.3.2 Program not run outside of programming cycle

This issue occurred where the program would run fine whenever we programmed it, but as soon as we hit the reset button or unplugged it and plugged it back in, it would not do anything. There were a lot of different reasons found on the internet, but one in particular talked about an Errata on revisions through E [5]. The development boards we had been working on all semester were revision E and had the required change implemented in the code. The boards we designed were revision F, which, at the time of writing, does not have an errata sheet after some messing around with the settings in sniffer.syscfg, we found that the fix mentioned above broke our code, so to fix it, we had to uncheck the Enable Bootloader Backdoor option in TI Devices, Device Configurations as shown in figure 45. Once that was done, the program was maintained throughout power cycles and resets.

9 APPENDIX II (PREVIOUS DESIGNS)

9.1 Design 0 (Initial Design)

9.1.1 Design Visual and Description

Our initial plan was to include an ESP8266 on the Sniffer Nodes that would connect to the ISU Wi-Fi network or a router. It would be used to communicate all test logs back to the Host and would allow us to not worry about range too much and always be able to have Sniffer nodes listening and logging data. This design was intended to be very modular, meaning

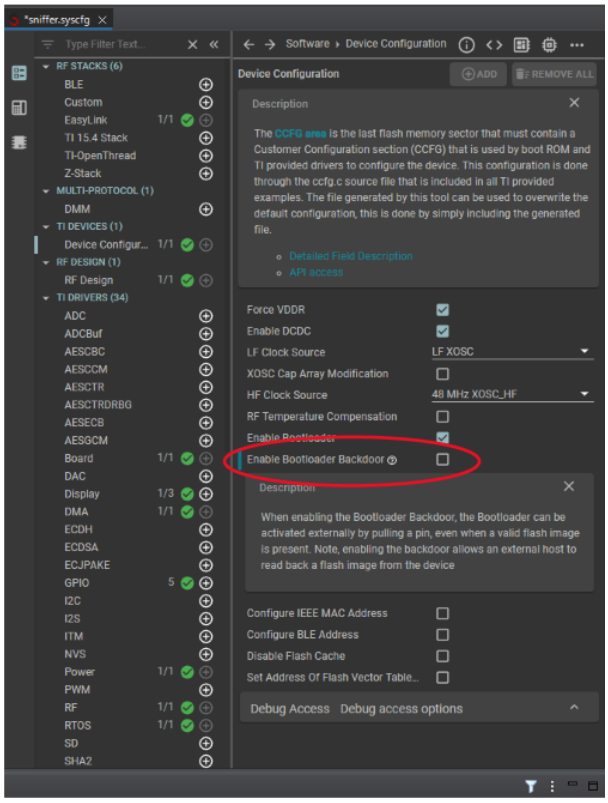


Figure 45: Disable the Backdoor Bootloader

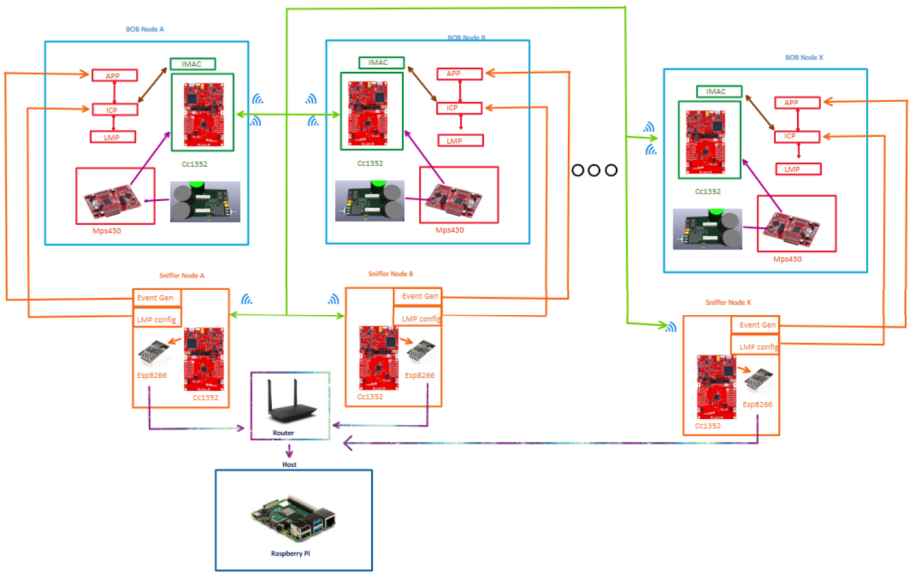


Figure 46: The initial idea we had with ESPs

that more nodes should easily be added to the system. Adding more nodes would be very straightforward; all aspects would be wireless, maintaining a clean setup. The data logging

would be on a different band than the BOBs, so no interference would be created.

9.1.2 Functionality

This initial and, since scrapped design, would function much like the research team's current design. The Sniffers would listen to wired and wireless communication from the BOB simultaneously, as well as have the functionality to create tests on the BOB boards using faux sensor data. The main difference is that instead of communicating to and from the Host wired via UART, the Sniffers would communicate using an ESP8266 to relay data back to the Host using the internet. This design would likely fill all requirements but harbored a concern of requiring too much power to reasonably be powered off a battery for a week or more. Additionally, our advisor thought adding another microcontroller and the communication work required would add too much complexity for this stage of the project.

9.2 Design 1 (Design Iteration)

9.2.1 Design Visual and Description

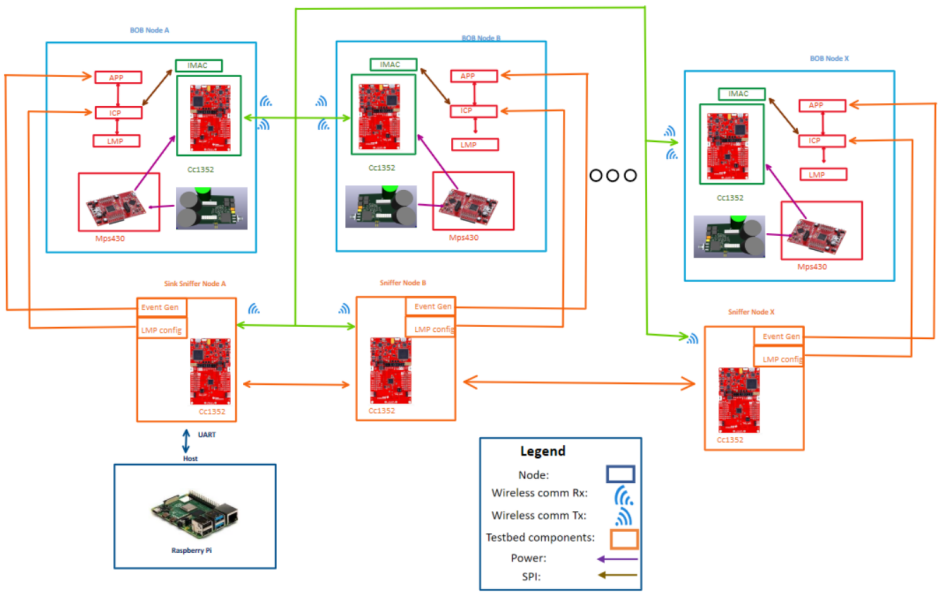


Figure 47: Our current design plan

Our current iteration involves having the Sniffer nodes harvest all their data independently,

then passing them through a predetermined route of Sniffers until the data ultimately reaches a Sink node. This Sink node would be the “test Host” of all Sniffers. We are unsure if we can use two communication bands simultaneously on the CC1352, or if we will have to create a strategic switching protocol to allow the board to both listen and communicate effectively. However, if we achieve it, the data will reach the Sink node and be communicated to the Host via UART for data logging and processing. Conversely, to set up tests, the Host communicates the setup to the Sink Sniffer via UART, and the Sink node will create a ripple distribution throughout the network until every board has received the setup and implemented it.

We took this design as it was suggested by our advisor, and it will require less power than design 0.

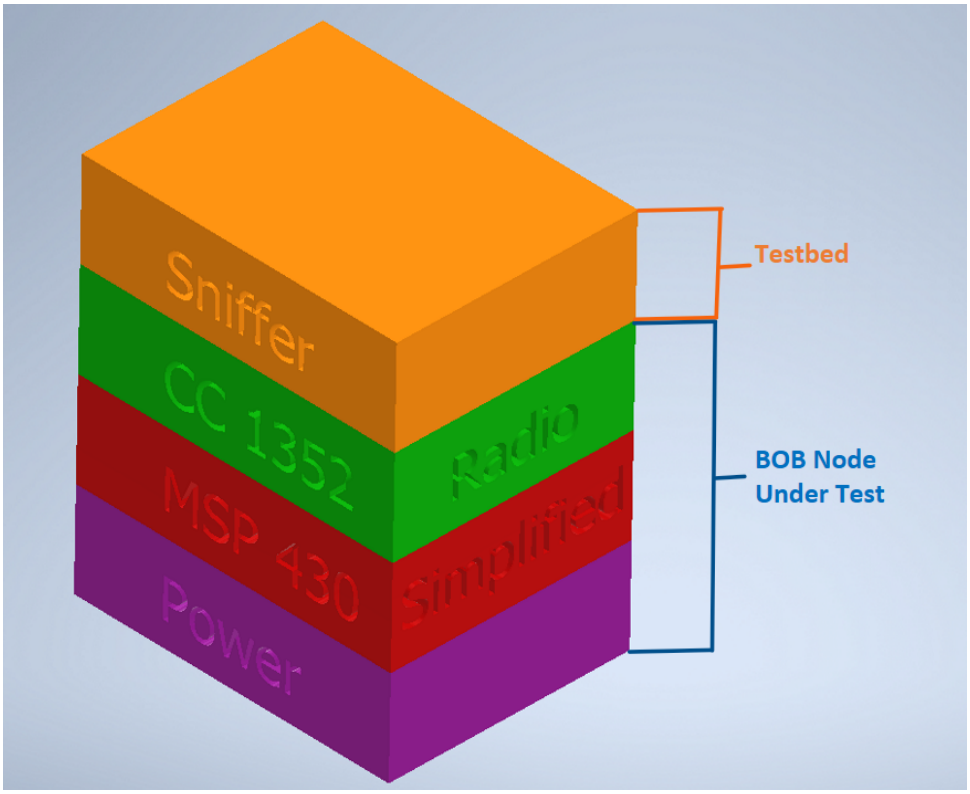


Figure 48: Our current hardware stackup plan

The above diagram gives a model for what our PCB stack up will look like.

The current design involves a 4-board stack. The bottom most board is the power harvester board. This will harvest the RF energy and supply power to the entirety of the stack. The current power harvester board is an off the shelf evaluation board. One of Dr. Duwe’s graduate students, Vishak, will be designing an in-house board to replace the current revision. This

will allow for more flexibility with the connector stack-up, as far as modification of connector pinout goes.

The next board in the stack is the MSP430. This board is currently an evaluation board and is being simplified down by our senior design team to condense the design and keep only the used functionality of the board. This is included as part of the BOB Node. This board is placed between the MSP430 and the CC1352 as both boards will contain radios and there are concerns of interference between the two. The two radios that connect via SMA connector will be placed on opposite sides of the board stack to account for this.

The CC1352 or “radio board” is used for communications both between each node and to the Host, through the Sniffer testing board.

The topmost board is the Sniffer board which is the primary focus of our senior design project. The Sniffer will be probing into the main connector of the node and evaluating the performance of the BOB Node. The Sniffer will only be used for initial prototyping and implementation. When the product is distributed and functioning beyond these steps, the Sniffer board will not be included in the unit. This is why the board is placed at the top of the stack as it can easily be removed once the unit is ready to be used.

9.2.2 Functionality

One of our main considerations for our current proposed solution is deciding how to allow each of the nodes to switch between the BOB and Sniffer communication bands without losing any information. The idea we landed on was having everything time synced relative to the Sink node, then having half of the nodes on each band at a time swapping off at regular intervals. We will have to have overlap where everything is on the BOB band for a short time so as to not miss any packets. A rough timing diagram of this system is shown in Figure 49.

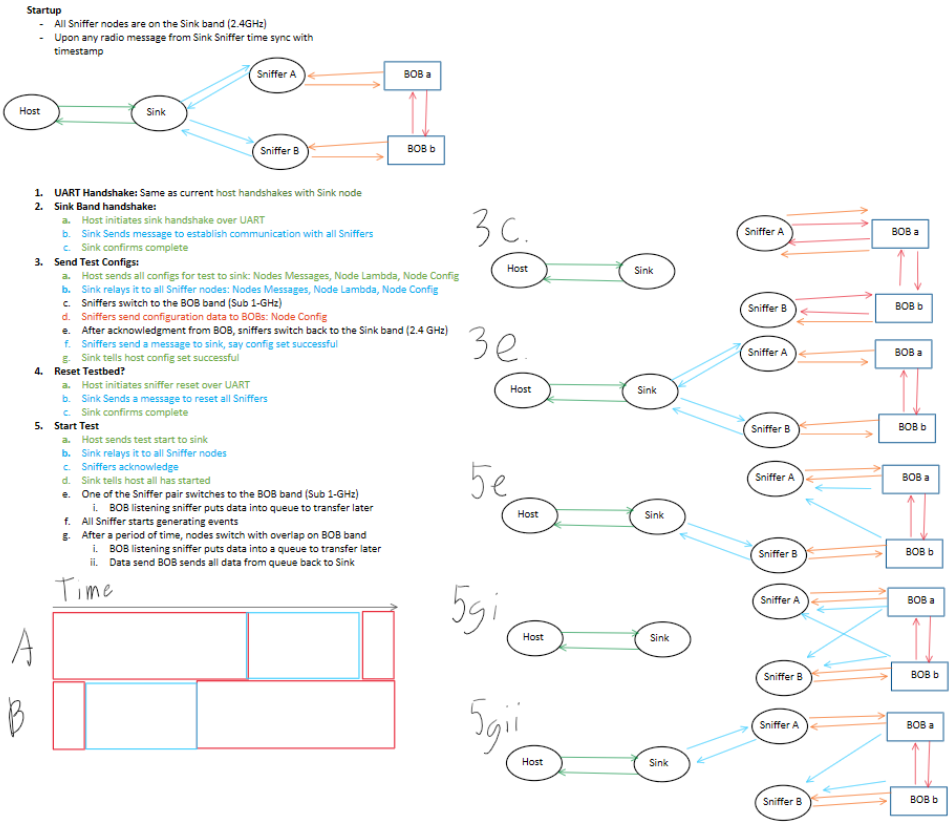


Figure 49: Our two-band pass-off

It can be seen that half of the Sniffers are listening back to the BOB nodes, while the other half are talking and listening to the other Sniffer nodes. This would happen according to the schedule shown at the bottom the image. The diagrams on the right side of the image are an example of how dataflow would work between the nodes over time.

Another important aspect of this band pass-off we decided upon is that in order to make it work without losing packets from the BOBs during pass-off time, we need to make sure that adjacent nodes don't end up on the Sniffer band at the same time. To this end, we decided upon a checkerboard pattern of band swapping, where each Sniffer has a specific data-path back to the Sink node. This simplifies the problem of switching bands, as we will always know which node is on what band based on the checkerboard we design, as everything will be reasonably synced to the Host node's time, as discussed. A diagram of this checkerboard pattern can be seen in Figure 50.

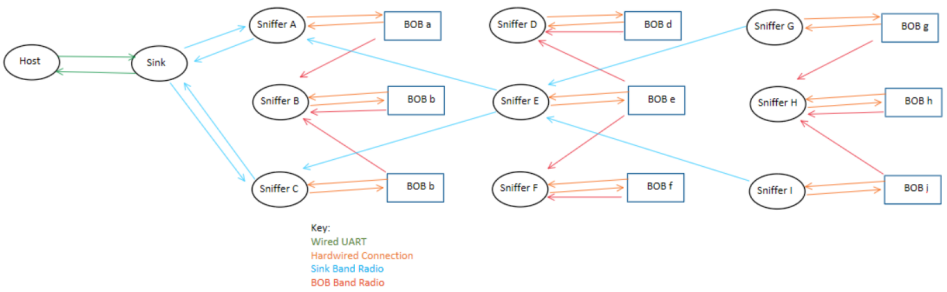


Figure 50: Checkerboard Band Swap

To ensure that we can maintain the band switching algorithm with the rates of transmission, band switching, saving, and loading to the queue under worse case load of one 8-byte transmission every 5ms, we ran some timing tests.

With some prototype testing, we found that we can send a 128-byte (maximum size) packet in 12ms, we can switch between bands (from BOB to Sink, and from Sink to BOB) in about 30 ms, and finally, can create an array-based queue of 2500 10 byte packets. In this queue, we can read or write the entire or it in under 15 ms.

With this data, we can generate a simple table (table 51 to show that we can maintain a band-switching procedure that can save all data while listening to the BOBs in the BOB band and transmit all the data back to the Sink while in the Sink band while being able to maintain the overhead of the band switching time while with the constraint of queue size. In this diagram, we are taking delays incurred by the queue as negligible and giving a 5 ms buffer to handle variation in Band Switching time. In actual implementation, we will have it run for longer on each band to make further use of our array of 2500 data packets and if we can make use of a maximum packet size we can transfer 128 bytes in on-go at the fastest rate.

Time(ms)	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
Odd																				
Odd(Bytes)	8	16	24	32	40	48	56	64	64	64	64	64	64	64			0	0	0	0
Even																				
Even(Bytes)	0	0	0	0	0	0	0	8	16	24	32	40	48	56	64	72	80	88	96	104
Time(ms)	105	110	115	120	125	130	135	140	145	150	155	160	165	170	175	180	185	190	195	200
Odd																				
Odd(Bytes)	0							8	16	24	32	40	48	56	64	72	80	88	96	104
Even																				
Even(Bytes)	112	120	128	136	144	152	160	168	168	168	168	168	168	168			40			0

Figure 51: Example Timing Schedule

Another main consideration of the Sniffer Node design is how it will be powered. The requirements for the power system is that it is easy to use, can power the Sniffer for up to 7

days continuously, and that the Sniffer node can optionally be powered by a wire to ensure that it does not fail. To meet these requirements, our design is to use a battery system to power the Sniffer and to also provide a wired power connection. When using the wired power connection, the system will either automatically switch over to the wired power option, or a switch will be available for the user to toggle between power modes.

There are a number of considerations to take into account for the design of the battery system. The first is the power, voltage, and current needs of the Sniffer board. The main element on the niffer board is the CC1352R processor. According to the datasheet, the CC1352R has its own internal regulators. This negates the need for a regulator on the chip. The CC1352R requires a supply voltage between 1.8 – 3.8V. According to measurements made in the datasheet, the maximum operating current of the chip is about 25mA. Our requirement will be that the battery system can supply 35mA to give plenty of margin. The battery system must meet all CC1352R power needs and have the capacity necessary to run the Sniffer for 7 days continuously (this requirement is estimated below).

The second consideration is cost of the system. One of the Sniffer Node requirements is to have a low cost. The client desires that the cost is low to enable more nodes to be tested on limited research funds and so that other groups can cost-effectively begin using our test system. The cost of the system includes batteries, ICs required to run them, mounting solutions, and potentially recharging systems. The goal is to reduce the cost per Sniffer. The cost also includes the cost of future use (for example if batteries must be replaced).

The third consideration is ease of use and practicality. The system should be easy for the graduate students to set up and use. For example, the system would not be easy to use if the batteries were difficult to put in and take out or if the batteries took an inordinate amount of time to charge.

The fourth consideration is sustainability. Disposal of batteries can have negative environmental effects. If possible, the environmental impact of our design should be minimized.

The next sections cover an estimation of the worst-case battery capacity requirement and several of the design ideas we generated. These will be reviewed with the client to determine the most optimal design.

10 Appendix III (Design Files)

10.1 Sniffer Board Design Files

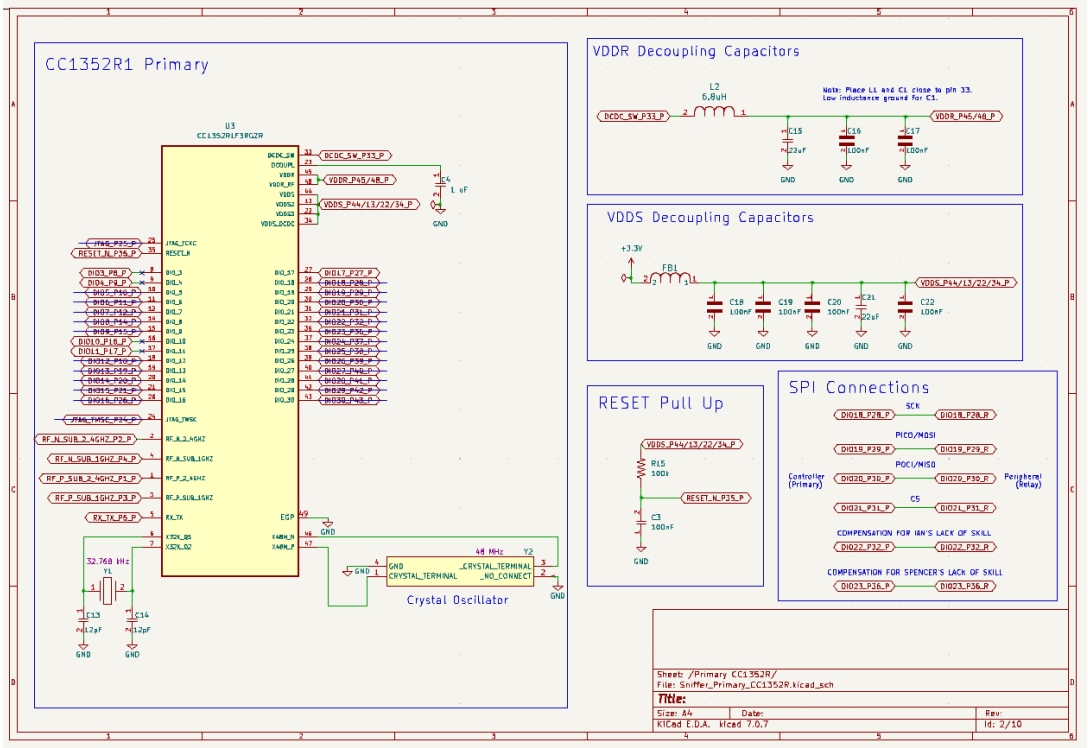


Figure 52: Sniffer Board Schematic

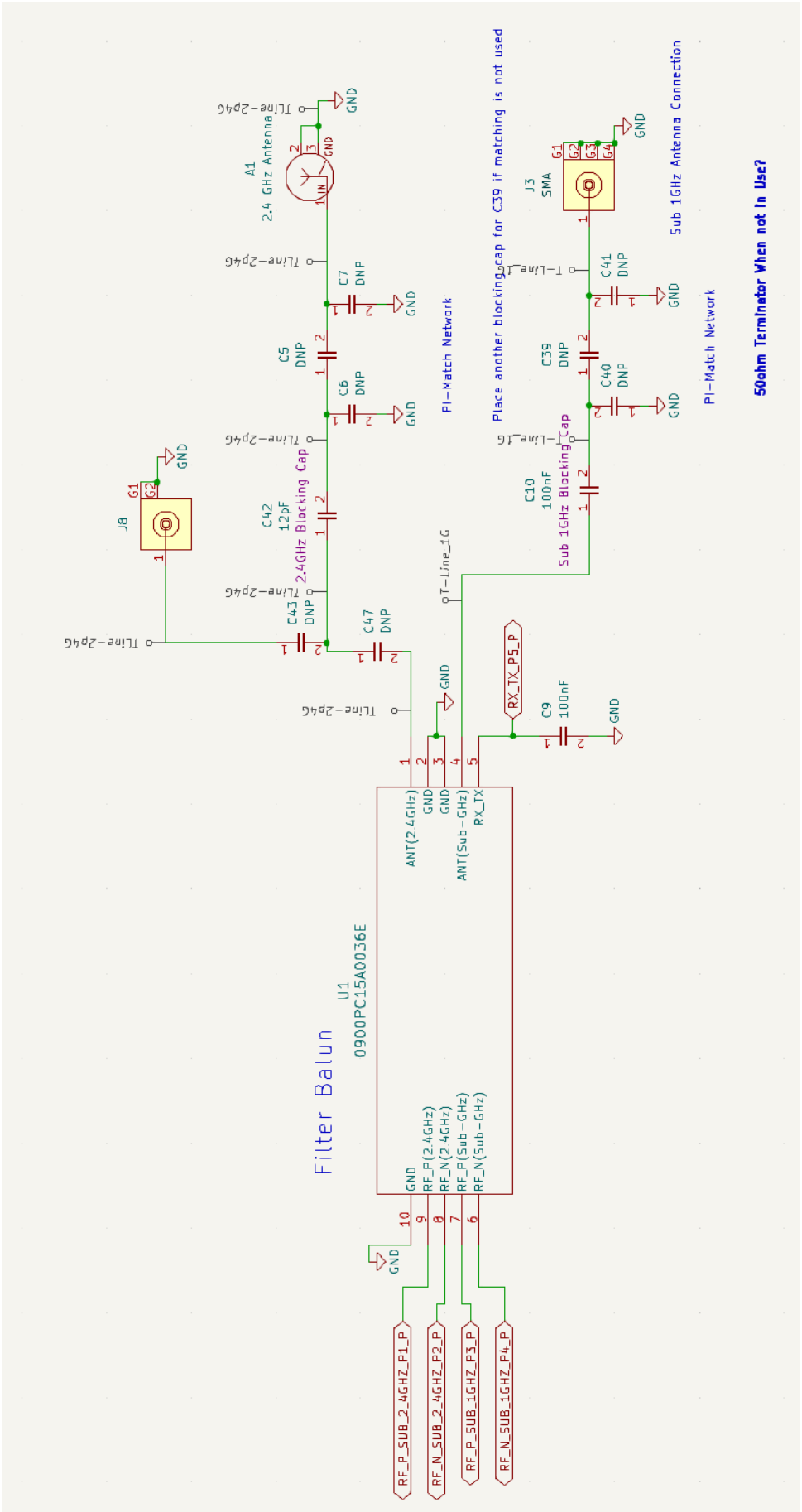


Figure 53: Sniffer Board Schematic

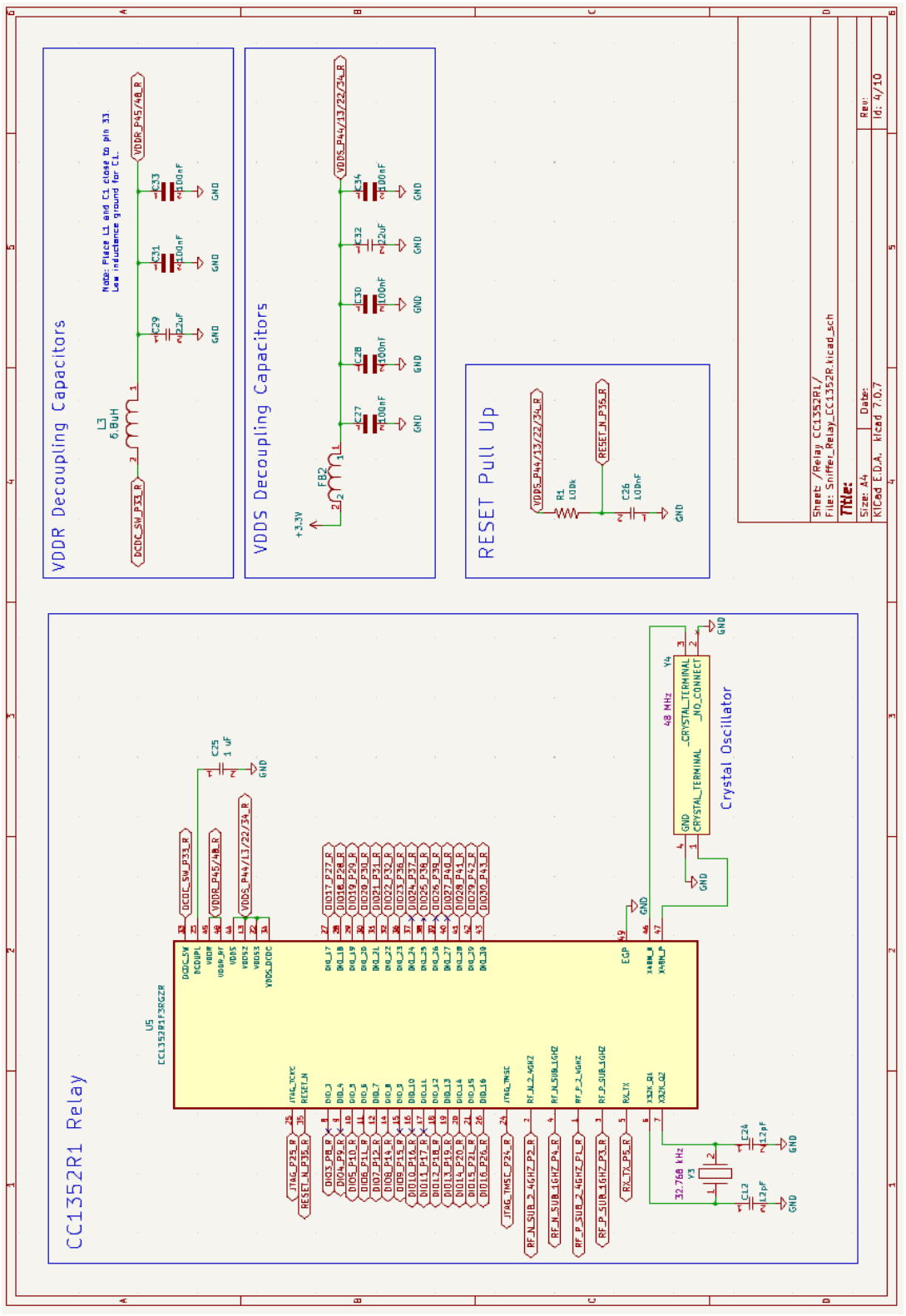


Figure 54: Sniffer Board Schematic

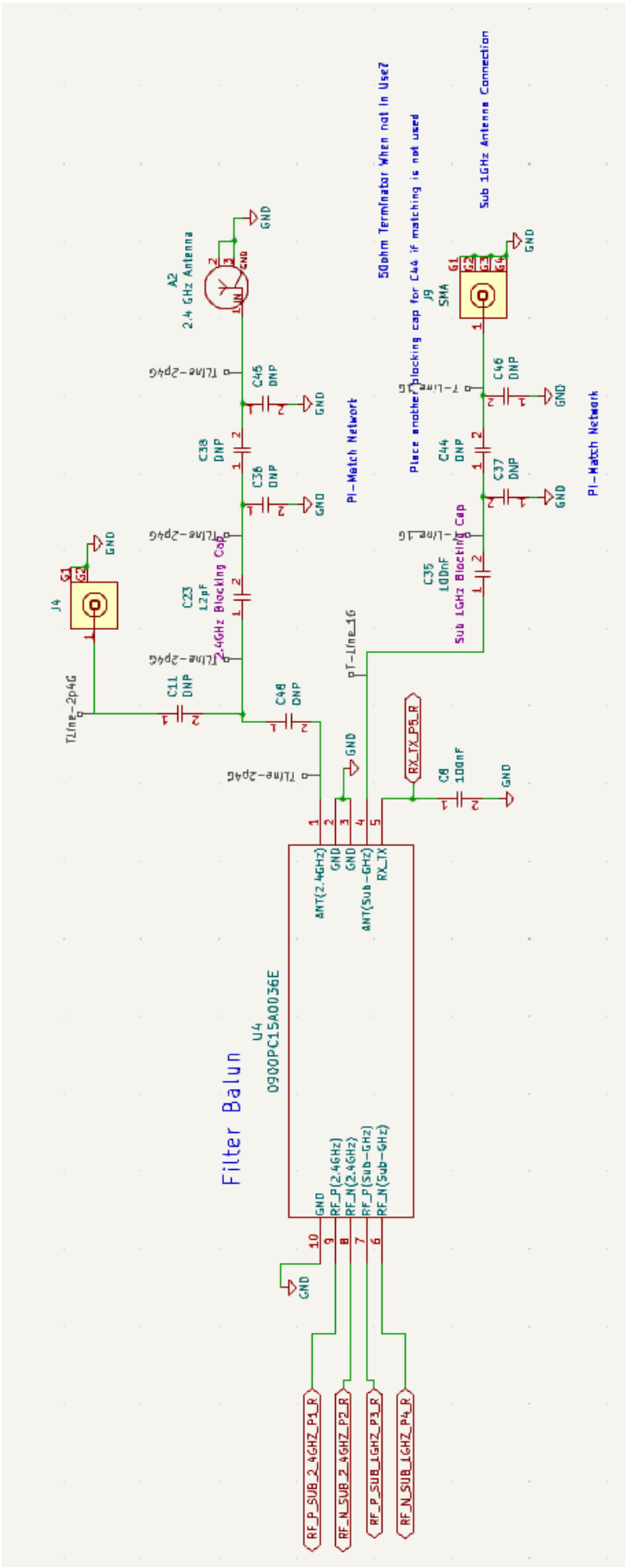


Figure 55: Sniffer Board Schematic

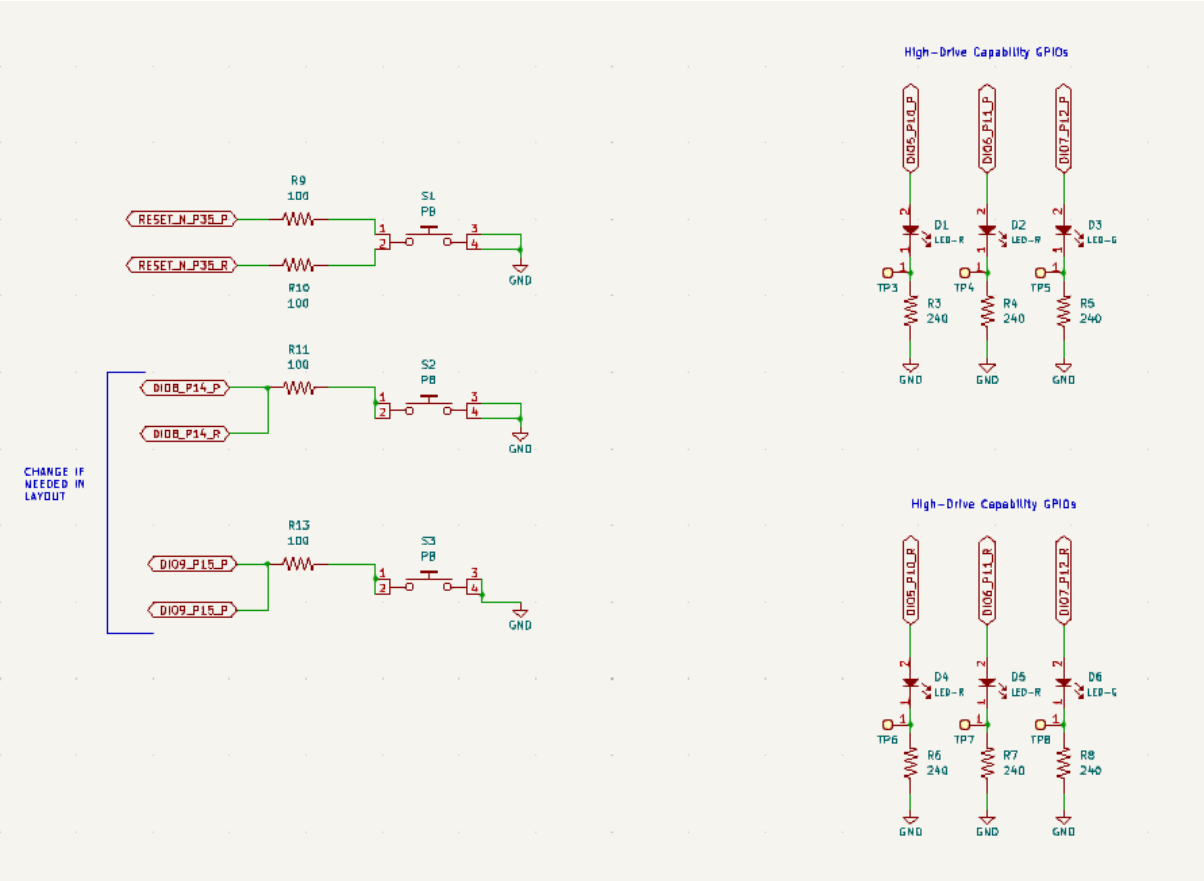


Figure 56: Sniffer Board Schematic

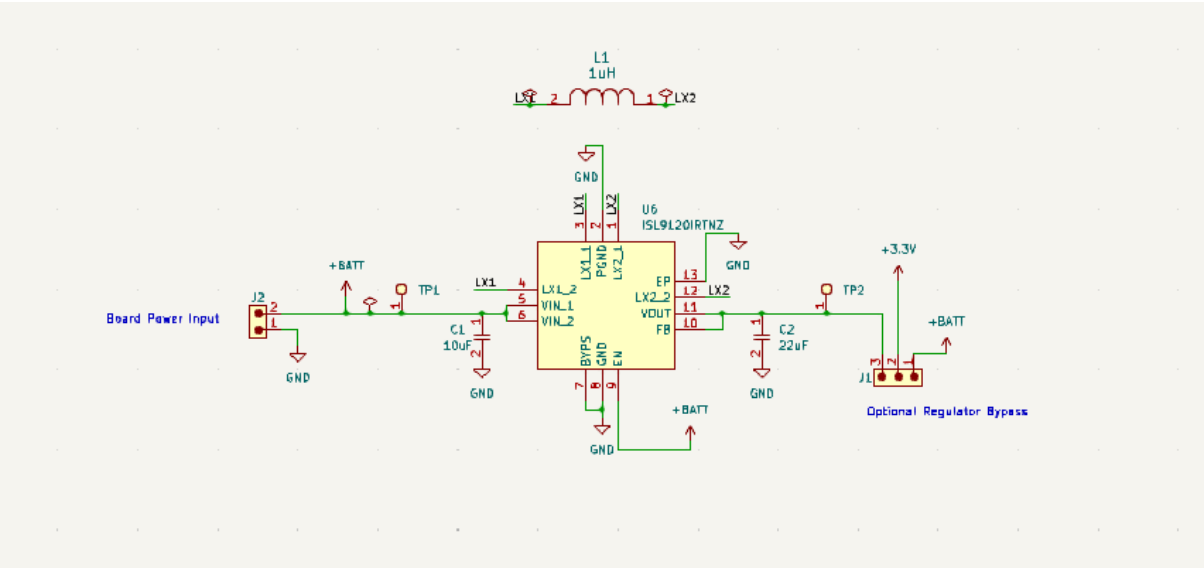


Figure 57: Sniffer Board Schematic

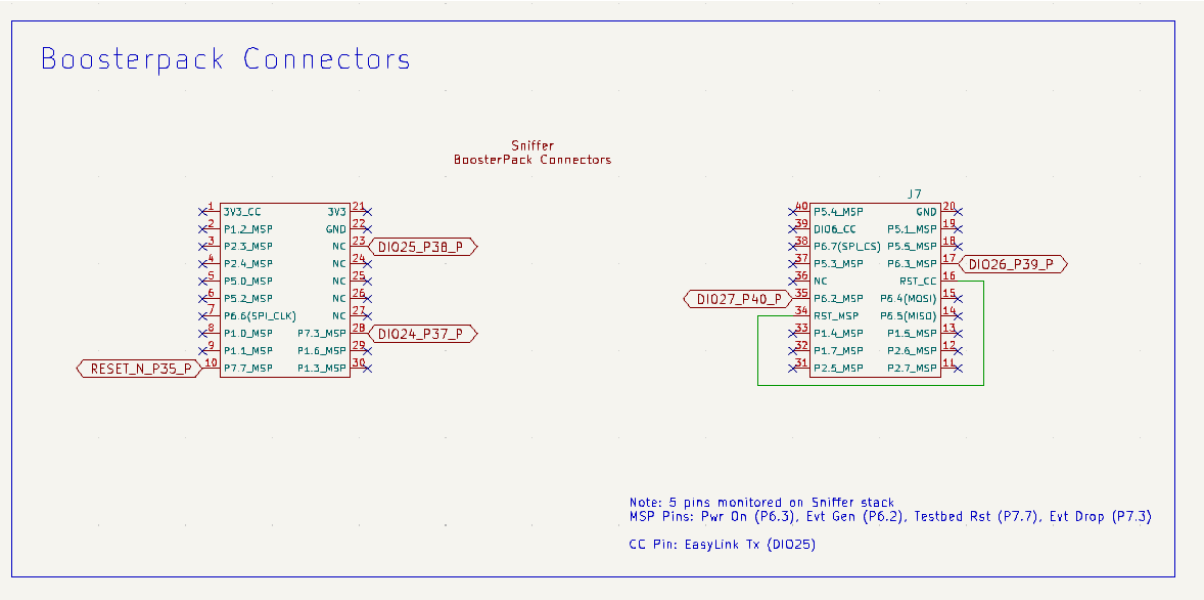


Figure 58: Sniffer Board Schematic

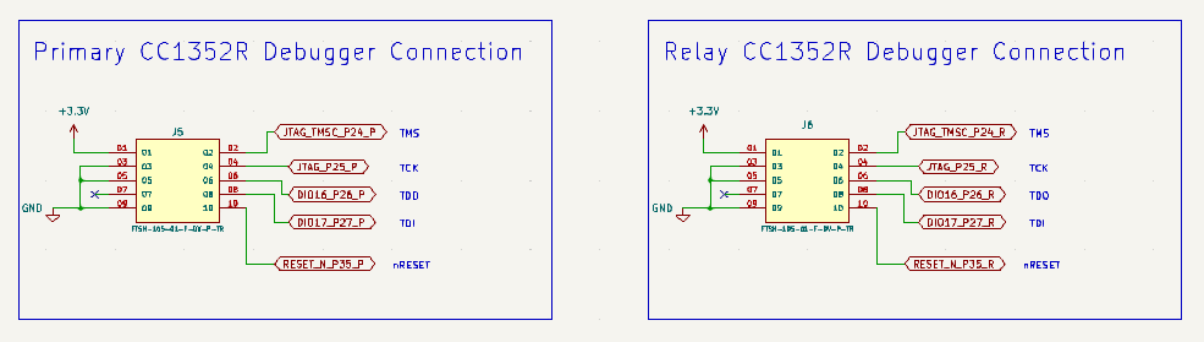


Figure 59: Sniffer Board Schematic

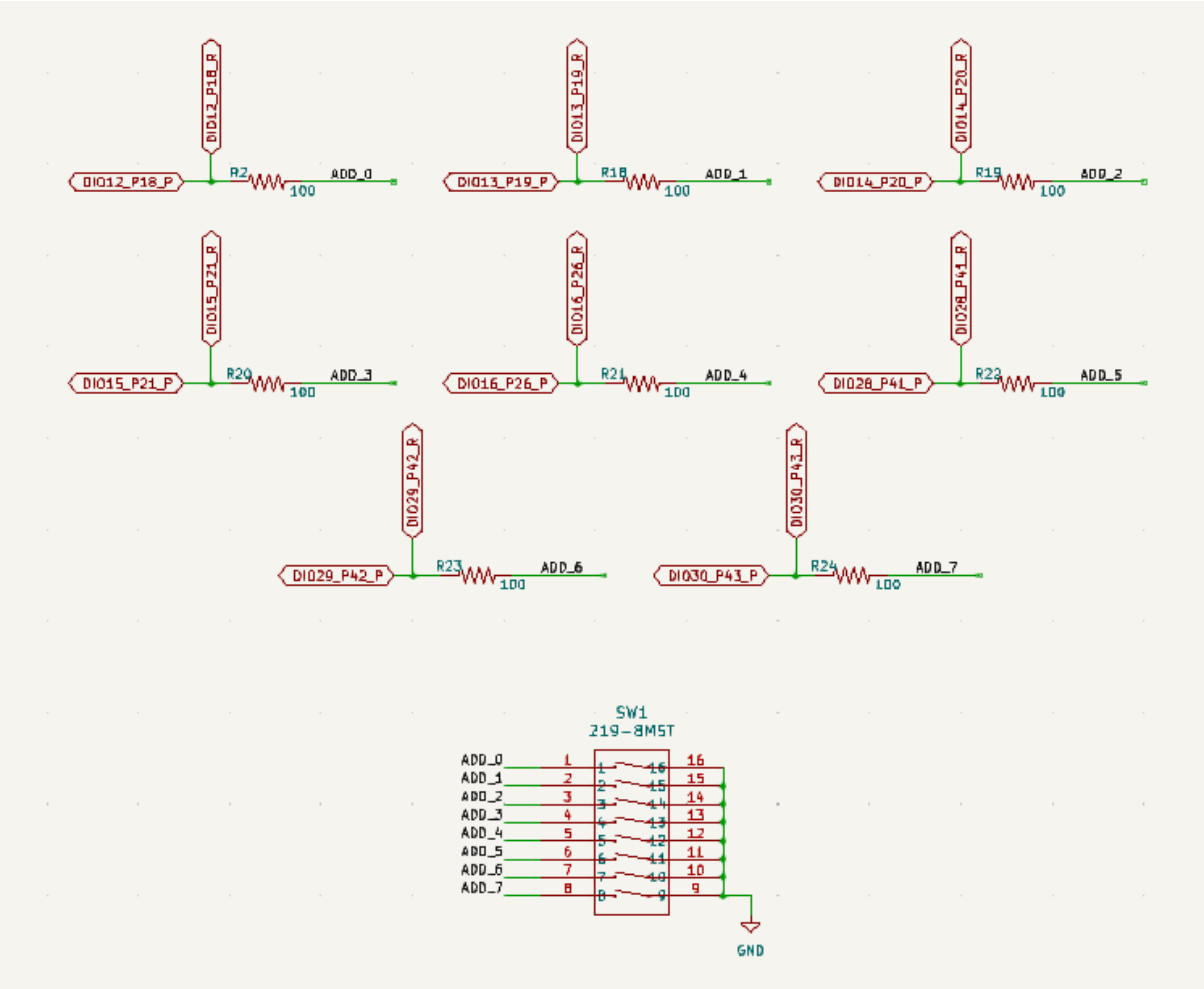


Figure 60: Sniffer Board Schematic

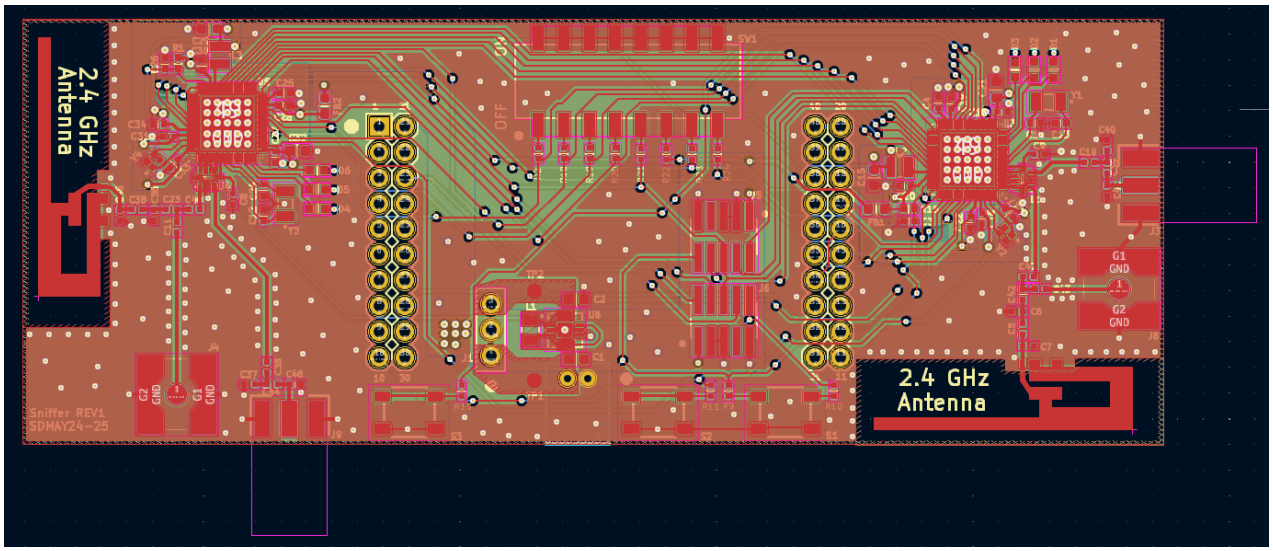


Figure 61: Sniffer Board Layout

Item #	Designator	Manufacturer	Mfg Part #	Description / Value	Package	Supplier	Link	Qty	Cost	Total Cost
1	U3,U5	Texas Instruments	CC1352R1F3RGZR	IC RF TXRX+MCU BLE 5.1 48VQFN		Digikey	https://www.d	25	7.2708	181.77
2	Y1,Y3	EPSON	FC-135 32.7680KA-AG0	CRYSTAL 32.7680KHZ 7PF SMD		Digikey	https://www.d	24	0.505	12.12
3	Y2,Y4	Murata Electronics	XRCMD48M000F1P2AR	48.0MHZ CRYSTAL UNIT +/-10PPM IN		Digikey	https://www.d	24	0.787	18.888
4	C4,C25	KYOCERA AVX	04026C105KAT2A	CAP CER 1UF 6.3V X7R 0402	0402	Digikey	https://www.d	24	0.32	7.68
5	L2,L3	TDK Corporation	MLZ2012N6R8LT000	FIXED IND 6.8UH 550MA 250MOHM SM		Digikey	https://www.d	25	0.0876	2.19
6	C15,C21,C29,C32,C2	Murata Electronics	GRM188R60J226MEA0I	CAP CER 22UF 6.3V X5R 0603	0603	Digikey	https://www.d	70	0.0608	4.256
7	C16,C17,C18,C19,C20,C22,C3,C31,C33,C27,C28,C30,C34,C26	KYOCERA AVX	KGM05AR51A104KH	CAP CER 0.1UF 10V X5R 0402	0402	Digikey	https://www.d	200	0.0062	1.24
8	FB1,FB2	Murata Electronics	BLM18HE152SN1D	FERRITE BEAD 1.5K OHM 0603 1LN	0603	Digikey	https://www.d	26	0.124	3.224
9	R15,R1	Vishay Dale	CRCW0402100KJNED	RES SMD 100K OHM 5% 1/16W 0402	0402	Digikey	https://www.d	50	0.0152	0.76
10	R9,R10,R11,R13,R2,R18,R19,R20,R21,R22,R23,R24	Vishay Dale	CRCW0402100RJNED	RES SMD 100 OHM 5% 1/16W 0402	0402	Digikey	https://www.d	150	0.0127	1.905
11	U1, U4,	Johanson Technology Ir	0900PC15A0036001E	RF Balun 862MHz ~ 928MHz, 2.4GHz ~	0805	Digikey	https://www.d	25	0.5568	13.92
12	J8, J4	TE Connectivity Linx	CONSMA001-SMD-G-T	SMA Connector Receptacle, Female So	-	Digikey	https://www.d	4	3.15	12.6
13	J3, J9	RF Solutions	CON-SMA-EDGE-S	SMA Connector Jack, Female Socket Bo	-	Digikey	https://www.d	25	2.2584	56.46
14	S1,S2,S3	C&K	PTS 647 SM50 SMTR2 LI	SWITCH TACTILE SPST-NO 0.05A 12V		Digikey	https://www.d	36	0.236	8.496
15	R3,R4,R5,R6,R7,R8	KOA Speer Electronics, I	RK73H1ETTP2400F	RES 240 OHM 1% 1/10W 0402	0402	Digikey	https://www.d	76	0.0134	1.0184
16	D1,D2,D3,D4,D5,D6	Harvatek Corporation	B1931USD-20D00081	LED RED DIFFUSED 0603 SMD	0603	Digikey	https://www.d	76	0.056	4.256
17	J2	JST Sales America Inc.	S2B-PH-K-S	Connector Header Through Hole, Right	TH	Digikey	https://www.d	12	0.136	1.632
18	L1	TDK Corporation	MLZ2012N6R8LT000	0.8 pH Shielded Multilayer Inductor 550 mA 250mOhm 0805 (2012 Metric)	0805	Digikey	https://www.d	14	0.095	1.33
19	C1	Murata Electronics	GRM188R61A106ME6	10 µF ±20% 10V Ceramic Capacitor	0603	Digikey	https://www.d	14	0.098	1.372
20	J1	Sullins Connector Solut	PRPC003SAAN-RC	CONN HEADER VERT 3POS 2.54MM		Digikey	https://www.d	12	0.076	0.912
21	J7	DIKAVS	n/a	nent conn 2x7pin. Using breakaway pin	n/a	Amazon	B3XBYL3J/ref=	5	11.99	59.95
22	J5,J6	Samtec Inc.	FTSH-105-01-F-DV-P-T	CONN HEADER SMD 10POS 1.27MM		Digikey	https://www.d	24	2.568	61.632
23	SW1	CTS Electrocomponent	219-8MST	SWITCH SLIDE DIP SPST 0.1A 20V		Digikey	https://www.d	12	0.911	10.932
	RF Test Plan Capacitors below:									
24	n/a, C9, C10, C8, C35	Kemet	CBR04C108B5GAC	0.1 pF ±0.1pF 50V Ceramic Capacitor C	0402	Digikey	https://www.d	60	0.0626	3.756
25	n/a	Murata Electronics	GJM1555C1HR50WB01	CAP CER 0.5PF 50V C0G/NP0 0402	0402	Digikey	https://www.d	20	0.123	2.46
26	n/a	Murata Electronics	GJM1555C1H1R5WB01	CAP CER 1.5PF 50V C0G/NP0 0402	0402	Digikey	https://www.d	20	0.131	2.62
27	n/a	Johanson Technology Ir	QSCF500Q2R4B1GV00	CAP CER 2.4PF 50V C0G/NP0 0402	0402	Digikey	https://www.d	20	0.163	3.26
28	n/a	Johanson Technology Ir	QSCF500Q5R6B1GV00	CAP CER 5.6PF 50V C0G/NP0 0402	0402	Digikey	https://www.d	20	0.22	4.4
29	n/a	Murata Electronics	GJM1555C1H100FB01I	10 pF ±1% 50V Ceramic Capacitor C0G	0402	Digikey	https://www.d	20	0.113	2.26
30	n/a, C42, C23	Murata Electronics	GJM1555C1H120JB01I	12 pF ±5% 50V Ceramic Capacitor C0G	0402	Digikey	https://www.d	50	0.06	3
31	n/a	Murata Electronics	GJM1555C1H180GB01	18 pF ±2% 50V Ceramic Capacitor C0G	0402	Digikey	https://www.d	20	0.12	2.4
32	n/a	Johanson Technology Ir	QSCF500Q270G1GV00	CAP CER 27PF 50V C0G/NP0 0402	0402	Digikey	https://www.d	20	0.201	4.02
33	n/a	Murata Electronics	GJM1555C1H470FB01I	CAP CER 47PF 50V C0G/NP0 0402	0402	Digikey	https://www.d	20	0.114	2.28
34	n/a	Vishay Vitramon	VJ0402D680JXXAJ	CAP CER 68PF 25V C0G/NP0 0402	0402	Digikey	https://www.d	12	0.357	4.284
35	n/a	KEMET	CBR04C101F3GAC	CAP CER 100PF 25V C0G/NP0 0402	0402	Digikey	https://www.d	12	0.306	3.672
	-									
36	n/a	AideTek	n/a	Parts Box	-	Amazon	https://www.a	1	23.99	23.99
								Total Cost		**530.945**

Figure 62: Sniffer Board BOM

Item #	Designator	Manufacturer	Mfg Part #	Description / Value	Package	Supplier	Link	Qty	Cost	Total Cost
1	n/a	Keystone Electronics	2463	BATT HOLDER AA 2 C	n/a	Digikey	https://ww	10	1.29	12.9
2	n/a	JST Sales America Inc	PHR-2	2 Rectangular Conne	n/a	Digikey	https://ww	10	0.071	0.71
3	n/a	JST Sales America Inc	SPH-002T-P0.5	CONN SOCKET 24-3	n/a	Digikey	https://ww	25	0.0516	1.29
4	n/a	Nexperia USA Inc.	NZX3V9C,133	DIODE ZENER 4V 50(	TH	Digikey	https://ww	12	0.137	1.644
5	n/a	PowerRowl		8 bay charger	n/a	Amazon	https://ww	1x8bay	13.98	13.98
6	n/a	PowerRowl		16 bay charger	n/a	Amazon	https://ww	1x16bay	23.99	23.99
7	n/a	Amazon Basics		Rechargeable AA NiM	AA	Amazon	https://ww	1x24ct	34.68	34.68
8	n/a	Amazon Basics		Rechargeable AA NiM	AA	Amazon	https://ww	1x16ct	22.86	22.86
								Total Cost:		**112.054**

Figure 63: Battery BOM

10.2 Breakout Board Design Files

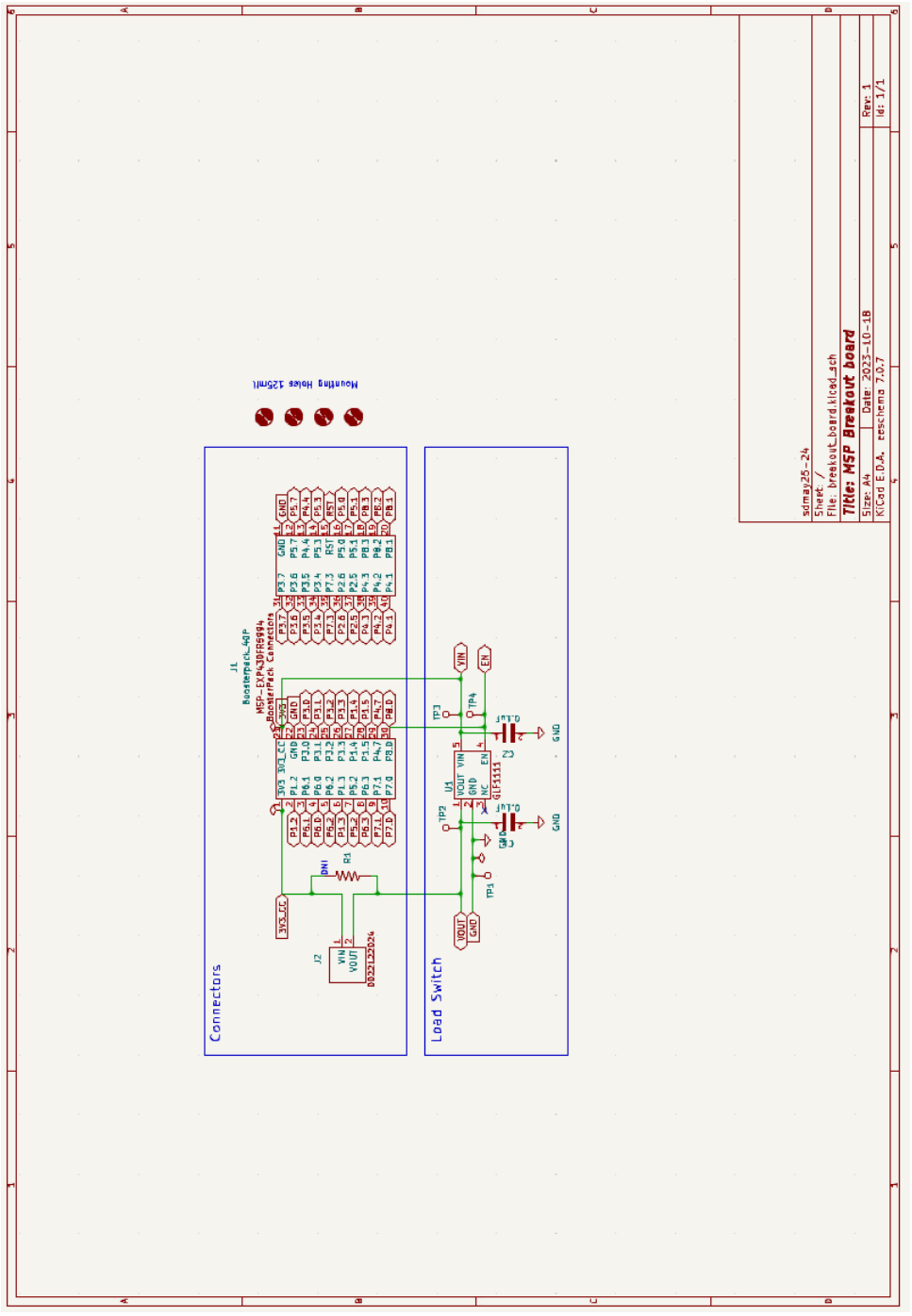


Figure 64: Breakout Board Schematic

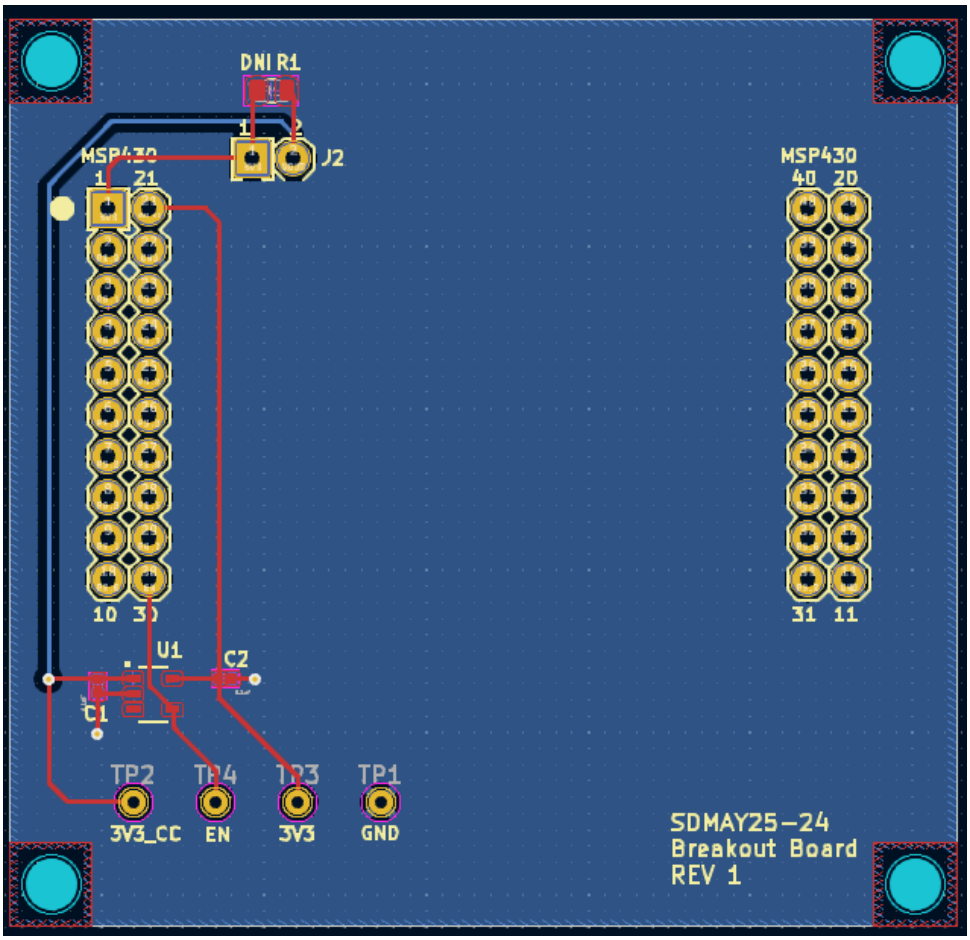


Figure 65: Breakout Board Layout

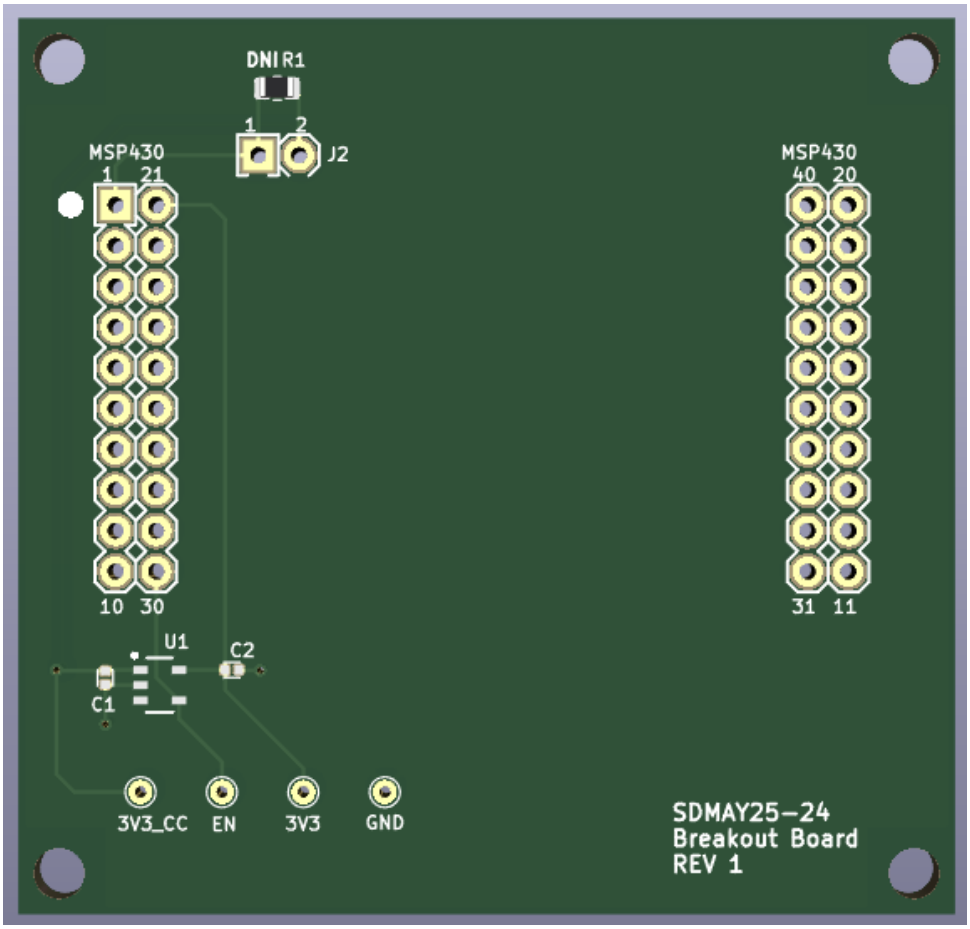


Figure 66: Breakout Board 3D Layout

SD25 2023 Breakout Board BOM

Item #	Designator	Manufacturer	Mfg Part #	Description / Value	Package	Supplier	Link	Qty	Cost	Total Cost
1	U1	GLF Integrated Power	GLF1111	Power Switch/Driver P-Channel 2A	SOT-23-5L	DigiKey	https://www.d	3	0.33	0.99
2	C1, C2	Samsung Electro-Mechanics	CL05A104KA5NNNC	CAP CER 0.1UF 25V X5R 0402	0402	DigiKey	https://www.d	10	0.01	0.1
3	J1	Samtec Inc.	SSW-110-03-G-D	CONN RCPT 20POS 0.1 GOLD PCB	-	DigiKey	https://www.d	4	3.89	15.56
4	J2	Molex	22122024	TH, Right Angle 2 position 0.100" (2.54mm)	-	DigiKey	https://www.d	2	0.77	1.54
5	R1	Stackpole Electronics	RMCF0805ZT0R00	RES 0 OHM JUMPER 1/8W 0805	0805	DigiKey	https://www.d	10	0.018	0.18
								Total Cost		18.37

Figure 67: Breakout Board BOM Order

Cost for Single Board

Item #	Designator	Manufacturer	Mfg Part #	Description / Value	Package	Supplier	Link	Qty	Cost	Total Cost
1	U1	GLF Integrated Power	GLF1111	Power Switch/Driver P-Channel 2A	SOT-23-5L	DigiKey	https://www.d	1	0.33	0.33
2	C1, C2	Samsung Electro-Mechanics	CL05A104KA5NNNC	CAP CER 0.1UF 25V X5R 0402	0402	DigiKey	https://www.d	2	0.01	0.02
3	J1	Samtec Inc.	SSW-110-03-G-D	CONN RCPT 20POS 0.1 GOLD PCB	-	DigiKey	https://www.d	2	3.89	7.78
4	J2	Molex	22122024	TH, Right Angle 2 position 0.100" (2.54mm)	-	DigiKey	https://www.d	1	0.77	0.77
5	R1	Stackpole Electronics	RMCF0805ZT0R00	RES 0 OHM JUMPER 1/8W 0805	0805	DigiKey	https://www.d	1	0.018	0.018
6	-	-	-	Board Fabrication	-	JLCPCB	-	1	3.892	3.892
								Total Cost		12.48

Figure 68: Breakout Board Cost Per Board

10.3 Simplified MSP-430 Board

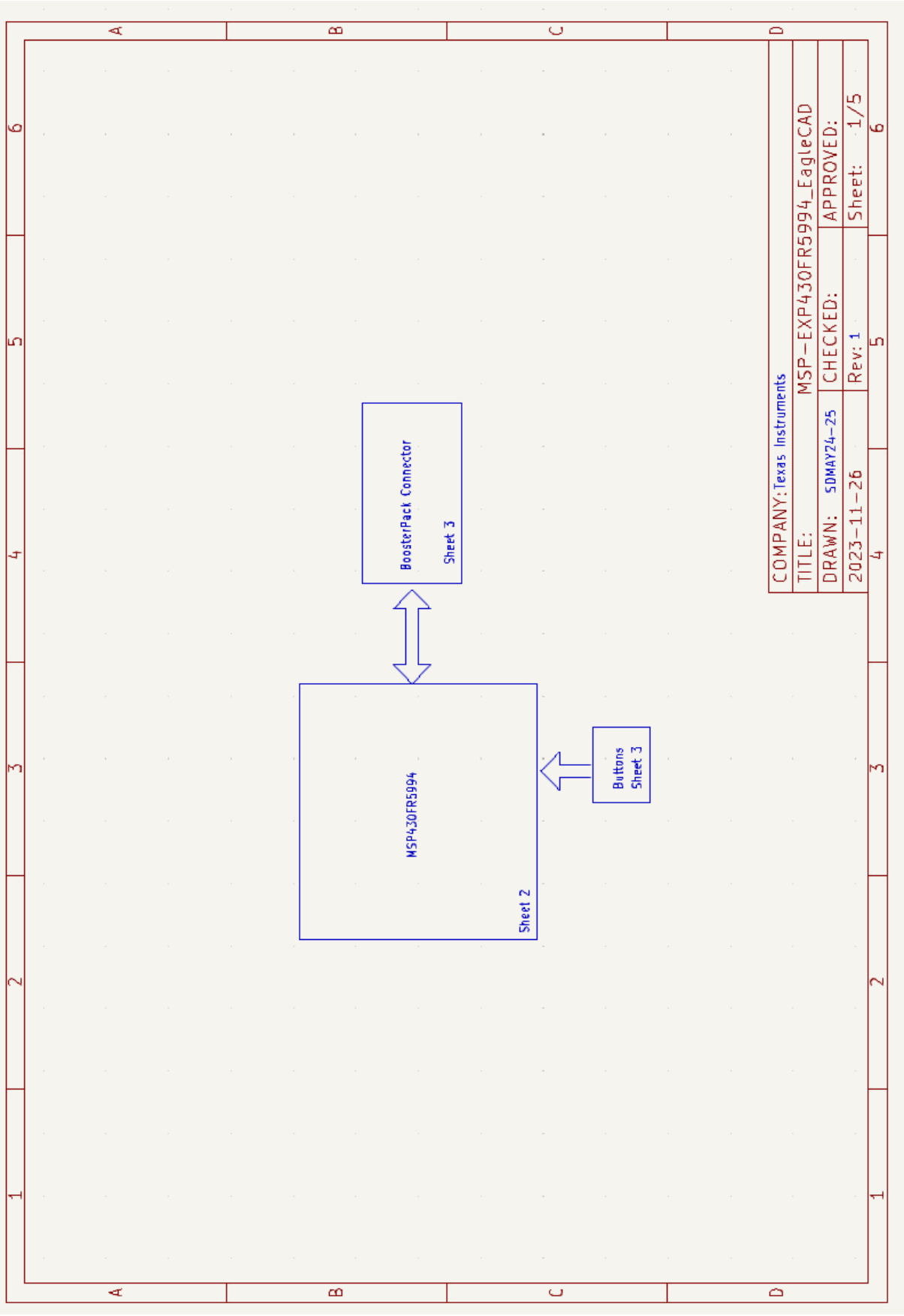


Figure 69: MSP Simplified Schematic Sheet 1

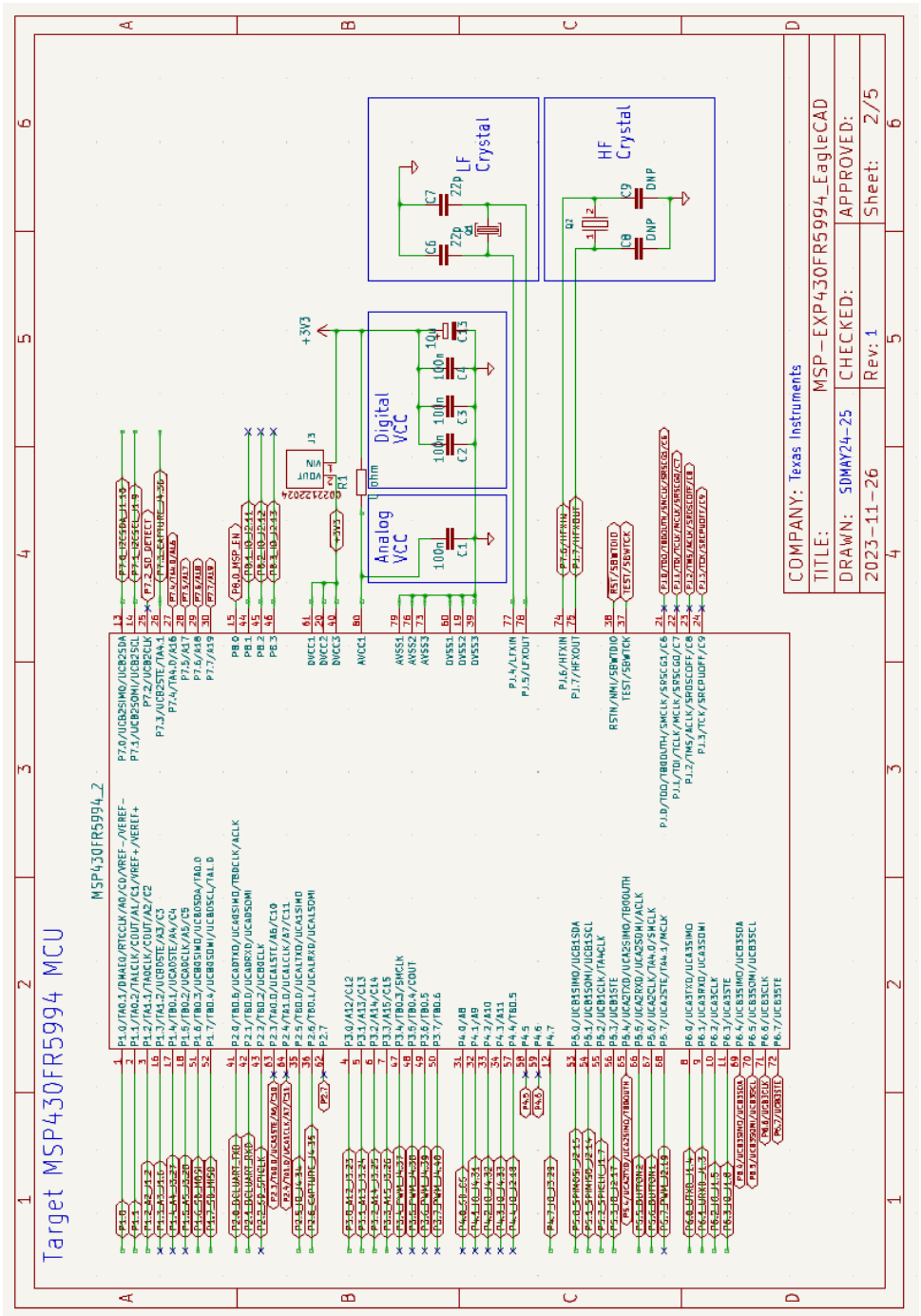


Figure 70: MSP Simplified Schematic Sheet 2

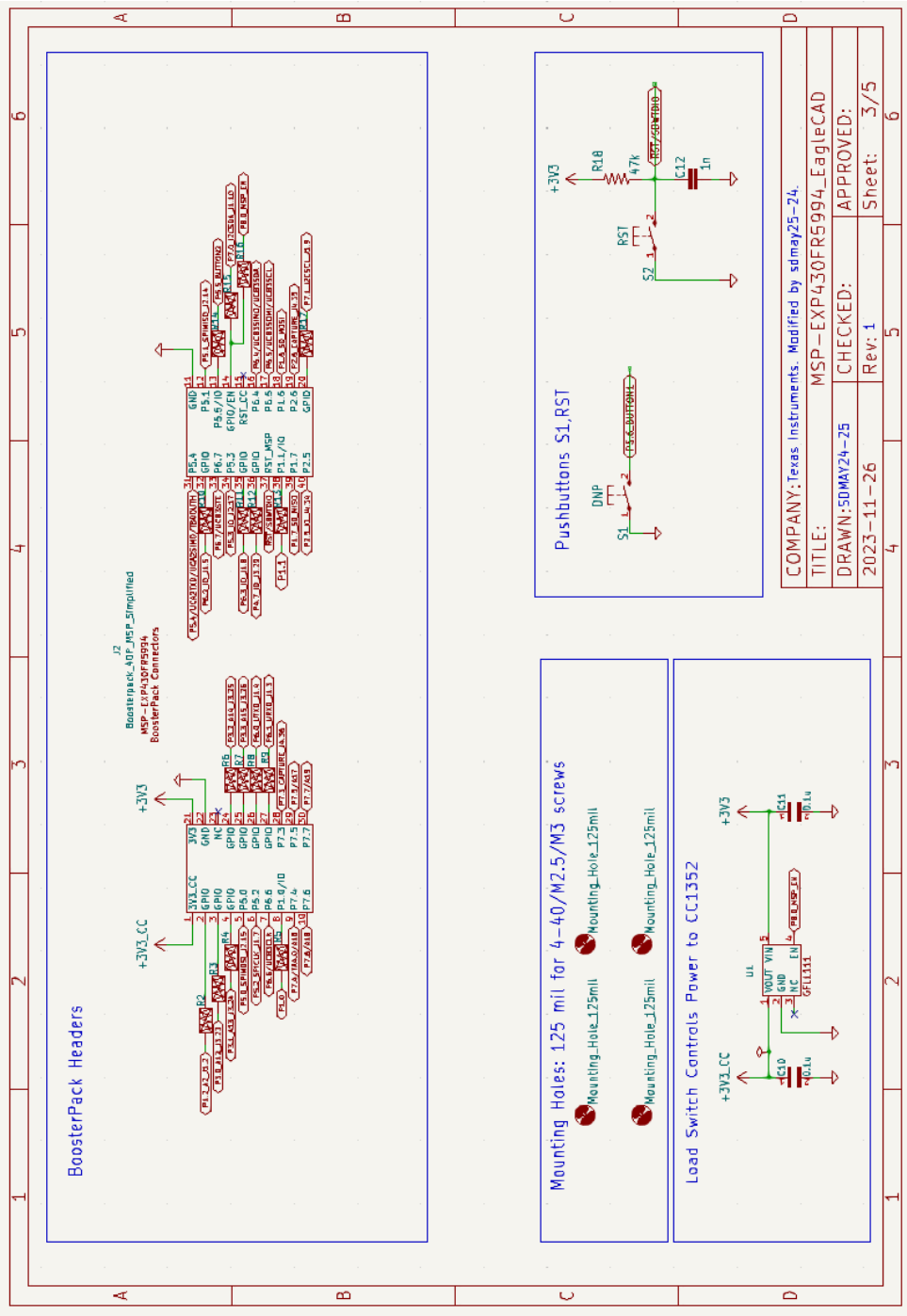


Figure 71: MSP Simplified Schematic Sheet 3

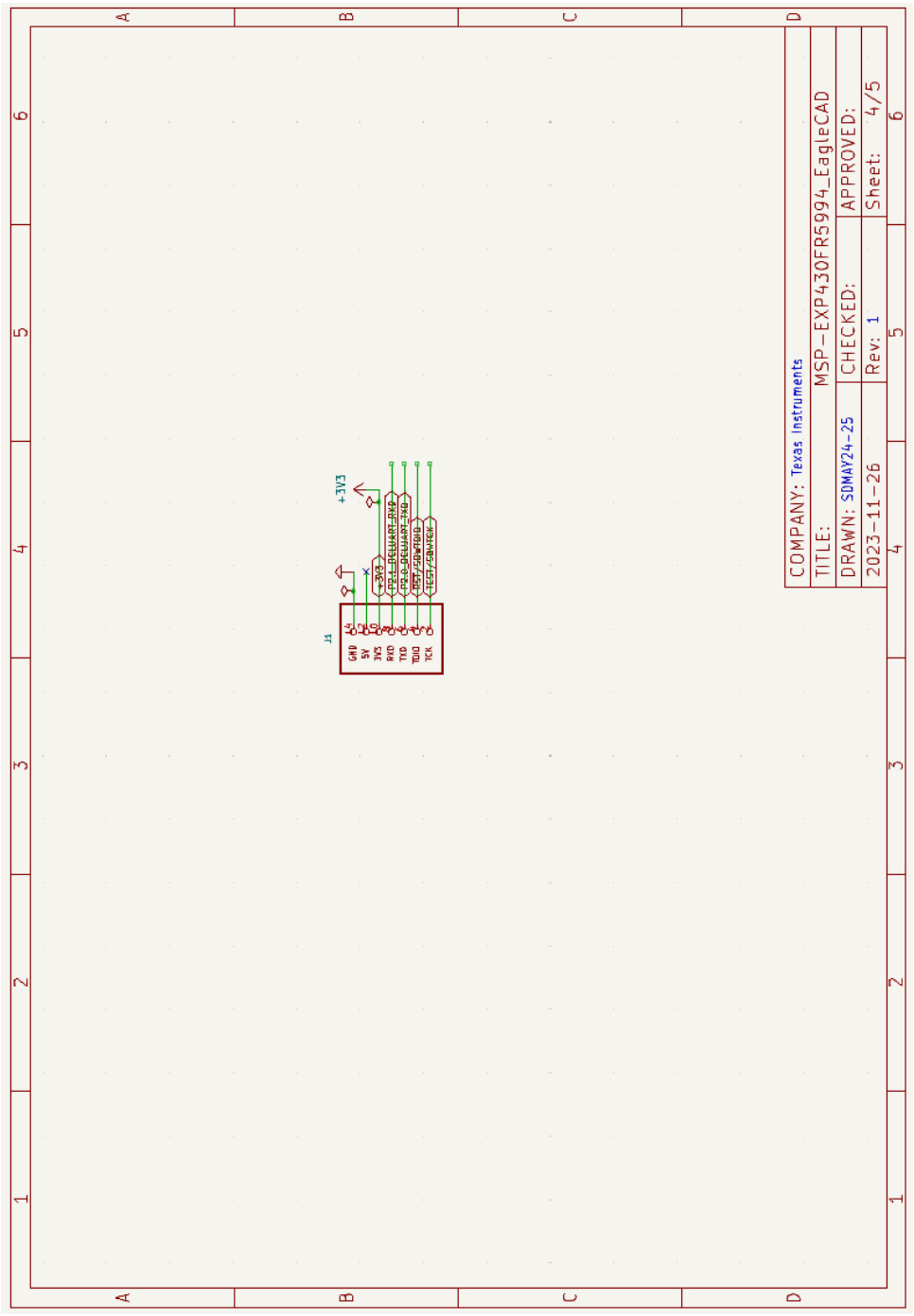


Figure 72: MSP Simplified Schematic Sheet 4

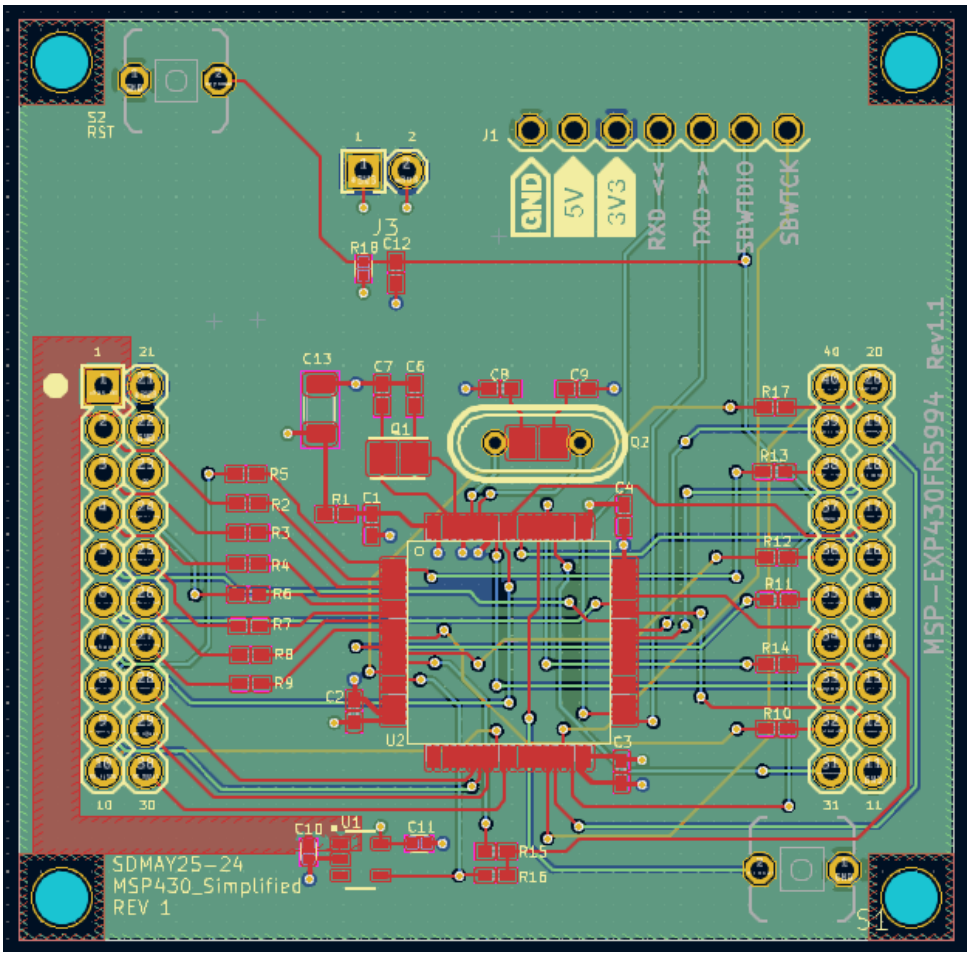


Figure 73: MSP Simplified Layout

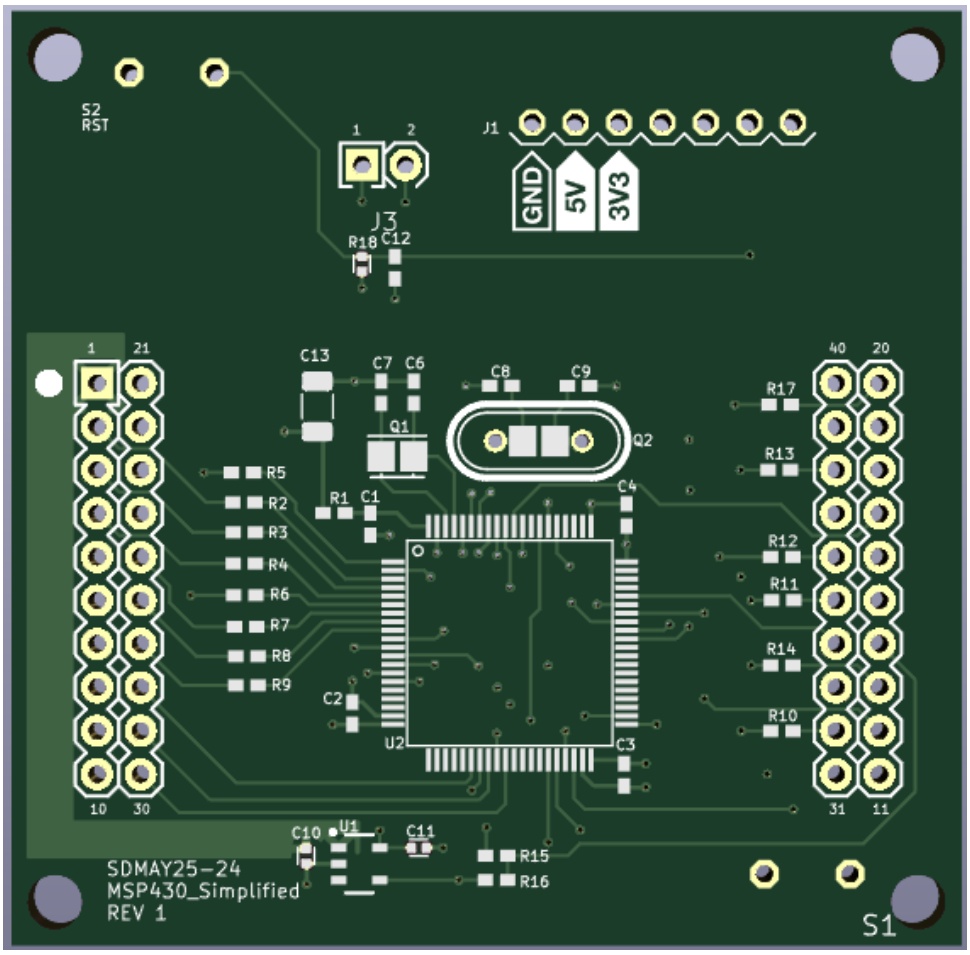


Figure 74: MSP Simplified 3D Layout

				SD25 2023 MSP_Simplified Board BOM						
Item #	Designator	Manufacturer	Mfg Part #	Description / Value	Package	Supplier	Link	Qty	Cost	Total Cost
1	U1	GLF Integrated Power	GLF1111	Power Switch/Driver P-Channel 2A	SOT-23-5L	DigiKey	https://www	12	0.33	3.96
2	C1, C2, C3, C4, C10, C11	TDK Corporation	C1005X5R1A104M050BA	CAP CER 0.1UF 10V X5R 0402	0402	DigiKey	C1005X5R1A	65	0.021	1.365
3	J2	Samtec Inc.	SSW-110-03-G-D	CONN RCPT 20POS 0.1 GOLD PCB		DigiKey	SSW-110-03	2	3.89	7.78
4	J3	Molex	22122024	TH, Right Angle 2 position 0.100" (2.54mm)	-	DigiKey	https://www	10	0.64	6.4
5	C6, C7	TDK Corporation	C1005C0G1H220J050BA	CAP CER 22PF 50V C0G 0402	0402	DigiKey	C1005C0G1H	25	0.047	1.175
6	C12	TDK Corporation	C1005X7R1H102K050BA	CAP CER 1000PF 50V X7R 0402	0402	DigiKey	C1005X7R1H	12	0.051	0.612
7	C13	Murata Electronics	GRM155R61A106ME11D	CAP CER 10UF 10V X5R 0402	0402	DigiKey	GRM155R61A	12	0.091	1.092
8	J1	Sullins Connector Soluti	PRPC007SBAN-M71RC	CONN HEADER R/A 7POS 2.54MM	-	DigiKey	PRPC007SBA	10	0.191	1.91
9	Q1	EPSON	FC-135R 32.7680KA-A0	CRYSTAL 32.7680KHZ 12.5PF SMD	-	DigiKey	FC-135R 32	2	0.7	1.4
10	R1, R2, R3, R4, R5, R6, R	YAGEO	RC0402JR-070RL	RES 0 OHM JUMPER 1/16W 0402	0402	DigiKey	RC0402JR-0	185	0.0045	0.8325
11	R18	YAGEO	RC0402FR-0747KL	RES 47K OHM 1% 1/16W 0402	0402	DigiKey	RC0402FR-0	12	0.015	0.18
12	U2	Texas Instruments	MSP430FR5994IPN	IC MCU 16BIT 256KB FRAM 80LQFP	-	Mouser	MSP430FR59	12	11.27	135.24
13	Q2	DNP								
14	-	Würth Elektronik	60900213421	JUMPER W/TEST PNT 1X2PINS 2.54MM	-	DigiKey	60900213421	10	0.31	3.1
15	S1,S2	E-Switch	TL59NF160Q	SWITCH TACTILE SPST-NO 0.05A 12V	-	DigiKey	TL59NF160Q	12	0.284	3.408
16	J2 (trying another comp	Samtec Inc.	SSW-110-23-G-D	CONN RCPT 20POS 0.1 GOLD PCB	-	DigiKey	SSW-110-23	2	5.71	11.42
	**NOTE: Qty provided above is for 10 boards								Total Cost	179.8745

Figure 75: MSP Simplified BOM Order

Cost for single board

Item #	Designator	Manufacturer	Mfg Part #	Description / Value	Package	Supplier	Link	Qty	Cost	Total Cost
1	U1	GLF Integrated Power	GLF1111	Power Switch/Driver P-Channel 2A	SOT-23-5L	DigiKey	https://www	1	0.33	0.33
2	C1, C2, C3, C4, C10, C11	TDK Corporation	C1005X5R1A104M050B4	CAP CER 0.1UF 10V X5R 0402	0402	DigiKey	C1005X5R1A	6	0.021	0.126
3	J2	Samtec Inc.	SSW-110-03-G-D	CONN RCPT 20POS 0.1 GOLD PCB		DigiKey	SSW-110-03	2	3.89	7.78
4	J3	Molex	22122024	TH, Right Angle 2 position 0.100" (2.54mm)	-	DigiKey	https://www	1	0.64	0.64
5	C6, C7	TDK Corporation	C1005C0G1H220J050BA	CAP CER 22PF 50V C0G 0402	0402	DigiKey	C1005C0G1H	2	0.047	0.094
6	C12	TDK Corporation	C1005X7R1H102K050BA	CAP CER 1000PF 50V X7R 0402	0402	DigiKey	C1005X7R1H	1	0.051	0.051
7	C13	Murata Electronics	GRM155R61A106ME11D	CAP CER 10UF 10V X5R 0402	0402	DigiKey	GRM155R61A	1	0.091	0.091
8	J1	Sullins Connector Solut	PRPC007SBAN-M71RC	CONN HEADER R/A 7POS 2.54MM	-	DigiKey	PRPC007SBA	1	0.191	0.191
9	Q1	EPSON	FC-135R 32.7680KA-A0	CRYSTAL 32.7680KHZ 12.5PF SMD	-	DigiKey	FC-135R 32	1	0.7	0.7
10	R1, R2, R3, R4, R5, R6, R	YAGEO	RC0402JR-070RL	RES 0 OHM JUMPER 1/16W 0402	0402	DigiKey	RC0402JR-0	17	0.0045	0.0765
11	R18	YAGEO	RC0402FR-0747KL	RES 47K OHM 1% 1/16W 0402	0402	DigiKey	RC0402FR-0	1	0.015	0.015
12	U2	Texas Instruments	MSP430FR5994IPN	IC MCU 16BIT 256KB FRAM 80LQFP	-	Mouser	MSP430FR59	1	11.27	11.27
13	Q2	DNP								
14	-	Würth Elektronik	609002134 21	JUMPER W/TEST PNT 1X2PINS 2.54MM	-	DigiKey	609002134 2	1	0.31	0.31
15	S1,S2	E-Switch	TL59NF160Q	SWITCH TACTILE SPST-NO 0.05A 12V	-	DigiKey	TL59NF160Q	2	0.284	0.568
16	J2 (trying another comp	Samtec Inc.	SSW-110-23-G-D	CONN RCPT 20POS 0.1 GOLD PCB	-	DigiKey	SSW-110-23	0	5.71	0
17	-	-	-	PCB Fabrication	-	JLCPCB	-	1	4.96	4.96
									Total Cost	27.2025

Figure 76: MSP Simplified Cost Per One Board

11 APPENDIX IV (TEAM CONTRACT)

Team Contract: _sdmay24-25__

TEAM MEMBERS

1. Thomas Gaul
2. Matthew Crabb
3. Spencer Sutton
4. Tori Kittleson
5. Ian Hollingworth

TEAM PROCEDURES

Regular Team Meetings

- Team meeting: Wednesdays at 7:30 pm in person at TLA.
- Client/advisor meeting: Tuesdays at 8:30 am in person at Durham 353 Conference room.
- TA meeting (Thomas, Matt, Tori): Wednesdays at 2:30 pm in Senior Design Lab.
- TA meeting (Spencer, Ian): Fridays at 1 pm in Senior Design Lab.
- Workday Monday 3:30 PM in the TLA

Communication

- Teams account for updates, reminders, and scheduling issues within the group and with the research team.
- Teams account for updates with the client/advisor.

Decision-making Policy

The majority vote for the final decision, focusing on consensus when possible. Diverted opinions can be discussed with the client/advisor for advice.

Record Keeping

Ian and Matthew will handle shared document redundancy on Teams.

PARTICIPATION EXPECTATIONS

Attendance and Punctuality

Attend when possible, communicate scheduling conflicts, catch up on missed meetings, and notify if running substantially late.

Responsibility for Assignments

Complete tasks on time, give warnings if deadlines might be missed, and avoid procrastination.

Communication with Team Members

Respond to team messages within 24 hours on weekdays, 36 hours on weekends, and a reasonable time on breaks.

Commitment to Team Decisions

Put full effort into assignments and ensure fair work distribution.

LEADERSHIP

Roles

- Thomas: Lead, Technical software
- Matt: Technical hardware
- Tori: Technical hardware
- Ian: Technical electrical systems
- Spencer: Technical software

Supporting and Guiding Work

Weekly meetings and sub-meets for team members working on project portions together.

Recognizing Contributions

Weekly meetings to discuss progress and record progress in weekly deliverables.

COLLABORATION AND INCLUSION

Team Member Skills

- Ian Hollingworth: C programming, PCB design, Digital circuit design, Verilog, Cadence
- Tori Kittleson: Hardware circuit design, Cadence, Altium PCB design
- Matthew Crabb: C Programming, analog circuit design, microwave circuit design, PCB layout, Cadence, Altium, MATLAB, PCB debugging, circuit measurement, Keysight ADS

- Thomas Gaul: Embedded Systems, hardware design, firmware experience
- Spencer Sutton: Embedded Systems, Digital Logic/microprocessors, Cadence, Physics

Encouraging Contributions

Weekly meetings for idea exchange and progress presentation.

Resolving Collaboration Issues

Bring up issues in weekly team meetings or with the team lead or Dr. Duwe if uncomfortable.

Goal-Setting, Planning, and Execution

Team Goals

- Concrete plan and presentation for project completion in the second semester.
- First revision of hardware completed and ordered for a second revision next semester, with completed testing code.

Planning and Assigning Work

Volunteer for tasks and select the most qualified individual if no one is particularly interested.

Keeping on Task

Create weekly goals and plan work on a week-by-week basis for the remainder of the semester.

CONSEQUENCES FOR NOT ADHERING TO TEAM CONTRACT

Handling Infractions

Discuss conflicts as a team and try to resolve to fit the needs of all team members.

Continued Infractions

If problems persist, discuss with the client/advisor for suggestions to adjust to the team's needs.

ACKNOWLEDGMENT

Confirmation of Understanding

1. Tori Kittleson DATE 9/8/23
2. Ian Hollingworth DATE 9/8/23
3. Matthew Crabb DATE 9/8/23
4. Thomas Gaul DATE 9/8/23
5. Spencer Sutton DATE 9/8/23

12 APPENDIX V (MATLAB SCRIPT FOR ANTENNA MATCHING)

clearvars;

f0 = 2.4*(10$\hat{9}$);

R1 = 50;

```
R2 = 34.74;
HLP = 0;
if R1 > R2
Rp = R1;
Rs = R2;
else
Rp = R2;
Rs = R1;
end
Q = sqrt(((Rp/Rs)-1));
Xs = Q*Rs;
Bp = Q/Rp;
if HLP
C = 1/(2*pi*f0*Xs);
L = 1/(2*pi*f0*Bp);
else
C = Bp/(2*pi*f0);
L = Xs/(2*pi*f0);
end
disp("Capacitance in pF");
disp(C*(10Î2));
```

```
disp("Inductance in nH");

disp(L*(10^9));
```