Making Microgrids Plug & Play

DESIGN DOCUMENT

Team Number: sdmay23 - 36
Client: Nick David
Advisor: Mathew Wymore
Team Members/Roles
Andrew Frank - Distributed Systems Architect
Christian Pinta - API Research and Programmer
Austin Thoreson - Systems and Hardware Engineer
Ben Eder - Software Developer
Saketh Jonnadula - Software Developer
Team Email: sdmay23-36@iastate.edu
Team Website: https://sdmay23-36.sd.ece.iastate.edu

Revised: 04.25.23 / v2.0

Table of Contents

Definitions

Abbreviation/Symbol	Definition
ISU EPRC	Iowa State University Electric Power Research Center
OMG DDS	Object Management Group Data Distribution Service
TMS	Tactical Microgrid Standard
RPi	Raspberry Pi
SunSpec Modbus	Open communication standard for Distributed Energy Resource systems
OutBack Power	Hardware device supplier
AXS Port	OutBack Power device to translate between TCP/Modbus over ethernet to proprietary OutBack Power communication
Microgrid / Pallet	A self-sufficient local electrical grid. In this document, refers to the PowerPallet Mobile design from the ISU EPRC

Table 0: Definitions

List of Figures and Tables

1 Introduction

1.1 Problem Statement

Everyone needs electricity. Microgrid pallets provide a mobile, modular power solution that can be applied to many use cases. Choice use cases are natural disaster relief, tactical deployment for military operations, or replacing gasoline-powered generators. Currently, connecting these pallets is time-consuming and requires technical expertise specific to these devices.

The goal of our project is to simplify the connection and configuration of pallets so that minimal to no training at all is required to use them. An extension of the project is to allow communication from the pallets to any connected electric grid and provide additional services.

1.2 Intended Users and Uses

The main use case focus for this project is on the military operation deployment scenario. Military personnel will have limited time and training to learn how to use the Microgrid pallets, and some untrained personnel may need to operate the equipment also to set up temporary camps. Because of this, the pallets need to be easy to modify, repair, and set up.

Outside of the military, the microgrids may need to be easily used by various disaster relief groups in order to provide power to those in need. The microgrids need to not only be simple to use but also be able to provide power quickly

Another user group is anyone in the general public who may need power for any reason. Whether it's to charge their phone or power a festival, anyone who needs power should be able to receive it simply by using one of these microgrids.

1.3 Requirements & Constraints

Functional Requirements -

- Facilitate communication between two or more Microgrid pallets through TCP/IP
- Send configuration commands to onboard OutBack Inverter and system
- Arbitrate leader/follower designation between pallets automatically
- Provide a display to view system status and major settings

Resource Requirements -

- Accept communication information following SunSpec Modbus from a connected grid
    - Possibly use this information to advise configuration settings
- Microcontroller to facilitate above communication
- Communication from other grids/controllers

Physical Requirements -

- Controller should fit on top of the pallet and connect to the existing RJ45 ports and OutBack AXS converter
- The display should fit alongside or replace MATE3 controller/display

Aesthetic Requirements -

- The interface should be simple with little to no interaction needed by the end user

User Experiential Requirements -

- Users can attach Microgrid pallets together without additional configuration.

Constraints -

- Fixed hardware for the existing system(s)
    - Some configurations unavailable according to safety regulation by OutBack
- Proprietary communication protocols with some existing systems

1.4 Engineering Standards

Tactical Microgrid Standard (TMS) - Draft, Official standard released March 2023

Some of the primary applications of these units are in disaster relief and military applications. TMS is meant to be the standard for Department of Defense (DoD) and industry needs/applications. TMS is meant to provide simple setup, be efficient, with resilient generation and distribution, all while being an open architecture.

SunSpec Modbus

Open communication standard that standardizes parameters and settings for monitoring and controlling distributed energy resources (DER). This protocol enables communication amongst different devices from multiple vendors within the same device/grid. Sunspec is made up of various IEEE protocols standardizing signal-to-noise ratios, and other aspects of network communications.

OMG- DDS

Object Management Group Data Distribution Service (OMG DDS) is a standard for publish-subscribe style communications commonly used for distributed networks. It is meant for real time or embedded systems, which makes it ideal for a microgrid system that needs to meet deadlines.

TCP/IP communication

To communicate with the pallets (which use a proprietary communication protocol) we send TCP/IP packets containing data which conforms to the SunSpec standard. An OutBack hardware device converts from their proprietary communication to TCP/IP, so to implement high level functions like leader/follower arbitration we will need to communicate with this standard.

IEEE 1547

This standard outlines requirements for interconnection and interoperability of distributed energy resources (DERs) and the utility electric power system (EPS). It provides requirements relevant to performance, operation, testing, safety, and maintenance of interconnection. Our use of this standard for this project is specifically related to the use of Freq/Watt compliance and its potential uses in AC source synchronization.

1.5 Design Evolution Since 491

- Communication following the TMS

1.6 Security Concerns and Countermeasures

Our project has a research focus on what is possible with the provided hardware, and did not have major security concerns. Eventually the design should be modified to conform to the TMS, which takes network security concerns into account.

1.7 Other Resource Requirements

Identify the other resources aside from financial (such as parts and materials) required to complete the project.

- Semi-Regular access to micro-grid pallets
- Raspberry Pi's (2x)
- Documentation on API's and associated tech
    - Microgrids
    - Sunspec
    - Tactical Microgrid Standard
    - OMG DDS

2 Design

2.1 Design Context

2.1.1 Broader Context

Our design would potentially have a broader impact on communities in disaster relief situations, and other portable power applications. If the setup of the microgrids is more accessible, the deployment should be quicker and thus further reaching in terms of scale of the networks. With larger networks/more deployments, more people in need would be able to get the help they need. With our automation of microgrid setup and coordination, they will be able to be used by people with a variety of backgrounds and serve higher-power requirement applications.

Relevant considerations related to our project:

Area	**Description**	**Our Project**
Public health, safety, and welfare	How does your project affect the general well-being of various stakeholder groups? These groups may be direct users or may be indirectly affected (e.g., solution is implemented in their communities)	Increasing availability/ease of use of emergency deployment equipment by providing portable power. Increasing availability/ease of use of renewable energy deployment equipment by providing portable power.

Global, cultural, and social	How well does your project reflect the values, practices, and aims of the cultural groups it affects? Groups may include but are not limited to specific communities, nations, professions, workplaces, and ethnic cultures.	The microgrids are much more environmentally friendly compared to generators as they are rechargeable and do not emit harmful gasses. The batteries used are also recycled. These microgrid pallets can act as distributed storage devices on an energy grid. When they are fully developed it may encourage moving to a distributed grid model rather than the centralized utility grid in use today. The Tactical Microgrid Standard (TMS) used as a framework for this project is in the draft stages right now, however our successful implementation may improve its development.
Environmental	What environmental impact might your project have? This can include indirect effects, such as deforestation or unsustainable practices related to materials manufacture or procurement.	The microgrids are designed for use with renewable power sources such as wind and solar, so they may increase the use of renewable energy in military operations. They use reclaimed batteries for energy storage which reduces electronic waste.
Economic	What economic impact might your project have? This can include the financial viability of your product within your team or company, cost to consumers, or broader economic effects on communities, markets, nations, and other groups.	The product itself is quite expensive, our work shouldn't affect cost much compared to the cost of existing hardware.

Table 2.1.1: Project Impact Context

2.1.2 Prior Work/Solutions

There are various products on the market that provide similar solutions in terms of generating and storing energy from renewable sources. Very few of these products are compliant to the TMS standard, which is a goal for our project.

Pallets like this exist and the specific pallets we are working on, do function. The problem is that those pallets are not TMS compliant and they have complex interfaces that require previous experience with electricity in order to operate them. The goal of our project is to automate a lot of

the pallet setup and create a simple, user friendly interface to allow anyone to use the pallets if they wanted to.

What our project aims to achieve is building upon these existing solutions by making them easier to configure. The interconnectivity, portability, and thus scalability of these packs/microgrids is what sets our project apart. Many solutions out there either do not support this connectivity or it is difficult to achieve.

2.1.3 Technical Complexity

The existing system contains an inverter/charger and energy storage unit with a user interface that is networked with a proprietary communications protocol developed by Outback Power.. Our design will require connection to this network and to a typical TCP/IP ethernet network, and coordination between the two.

The software we are creating must handle many different tasks, and we have divided it into several modules. One to handle user input and a graphical display, several to handle communications over the TMS network, and one to translate between user input, TMS communications, and communications over the Outback proprietary network.

Communication with proprietary hardware and software protocols is difficult in and of itself. We don't have access to proprietary information for troubleshooting and/or expanding upon device functionality.

In order to implement anything, we need a strong understanding of the system and standards that are required. It takes a lot of time to sift through and digest the documentation before we can begin making informed design decisions.

2.2 Pre-Existing Design

2.2.1 Initial State of the Project

When we started the project there was already a platform built for us on the basis of designing microgrids to be used for disaster recovery. These microgrids were built and designed to be around the size of a pallet and deliver between 8-80 kW for a duration of 10+ hrs. They were meant to be used for rapid deployment and be scalable for various applications. This previous work was done by Nicholas David, a part of Iowa State's Electric Power Research Center, in partnership with Iowa Economic Development, Iowa Army National Guard, and PowerFilm Solar.

2.2.2 Hardware Description

The devices on the pallets we were asked to continue to develop were a collection of products from Outback Power Systems. These devices were configured to implement the functionality described above. The figure below shows a high-level diagram of the pre-existing platform. This section will describe the function and role of each component in the pre-existing system.

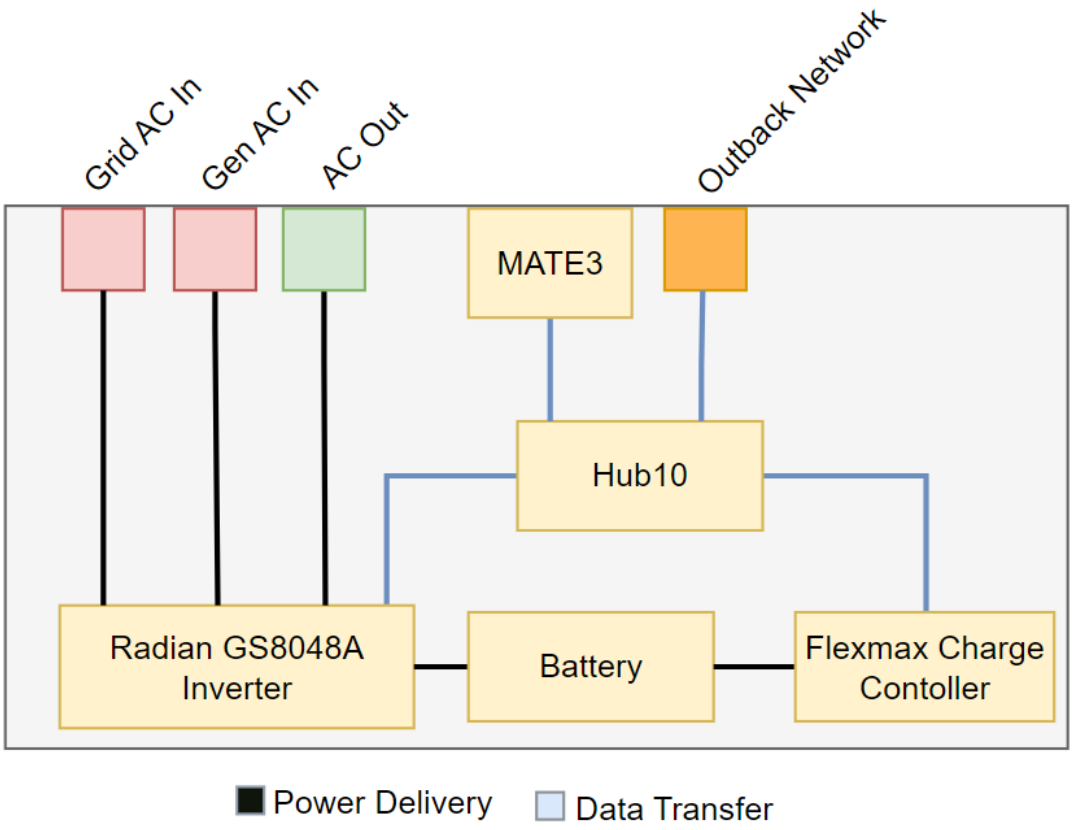

Figure 2.2.2.1: Devices of a PowerPallet (Simplified)

OutBack MATE3

The MATE3 is a configuration interface made by OutBack to control its devices. On this device, a user or installer can change various configurations allowing the system to meet the needs of the application or deployment. Our goal is to eventually remove the MATE3 completely, as it is mutually exclusive to the AXS Port, which allows us to make configurations autonomously.

OutBack Hub10

The Hub10 is a communication device that operates on an OutBack proprietary protocol. This protocol uses RJ45 terminated connections to the various devices it communicates with. While it does use RJ45 connections, it does not use a standard ethernet protocol and instead uses a proprietary protocol. The Hub10 requires specific devices to be plugged into static ports based on their role. It allows all OutBack devices on the pallet to communicate and be configurable with the AXS port or MATE3. The Hub10 is the traditional device used to enable “stacking” of parallel units. The diagram below shows the port assignment of the Hub10.

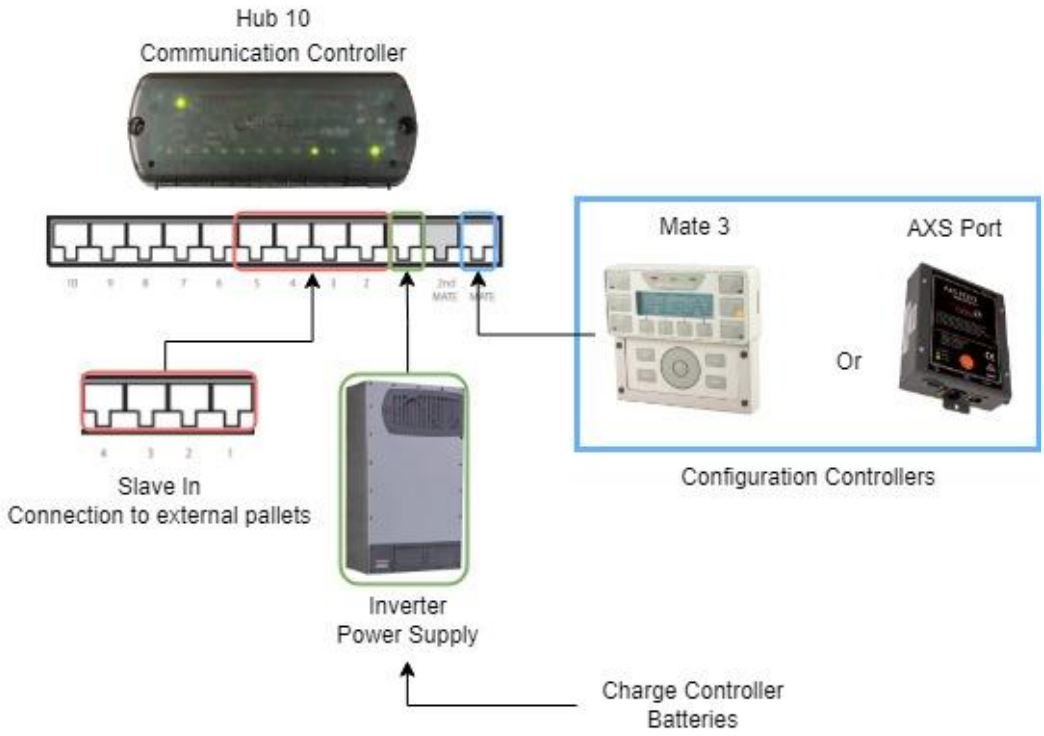

Figure 2.2.2.2: Hub 10 Ports

OutBack Radian GS8048A Inverter

The Radian is the inverter, and the main configurable unit onboard a pallet. The radian can be configured via the MATE3 or an AXS Port The radian has various configurations and modes that enable a wide range of functionality such as supplying AC power to a load from a DC source, buying and selling power from a utility grid, or even offsetting grid use with power from the connected batteries. The behavior of the inverter comes directly from the proprietary software and configurations developed by Outback.

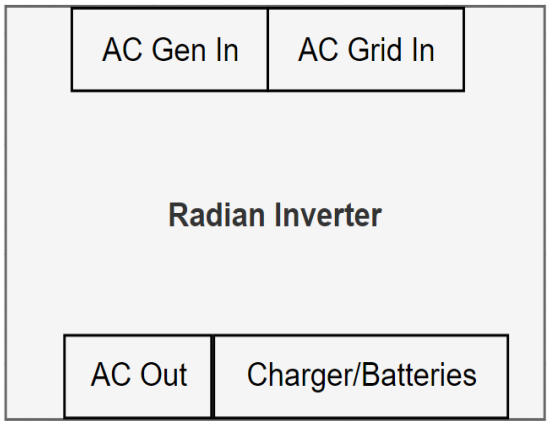

Figure 2.2.2.3: Physical Interfaces of the Radian (Simplified)

The Gen/Grid AC inputs are meant to be connected to the utility grid or a generator. They are individual, but equivalent inputs; Only one can be used at a time. They may be used interchangeably or referred to as the AC input throughout the document. The inputs are meant to accept 240VAC sources. This input power can be used to charge the batteries, power loads, and/or support the inverter during periods of high utilization. All of these functions are controlled by numerous settings and variables within the MATE3 interface.

The AC Out is simply the output where the Radian will supply power to connected loads. The Inverter comes equipped with a transfer relay that, when used, makes one of the AC inputs become electrically common with the output. This allows the AC input to be distributed to, and supply power to the loads. In order for the transfer relay to be used, the AC input must be accepted by the inverter. This AC input acceptance requires the AC source to fall within specific thresholds of voltage, frequency and should be relatively clean. If the source is acceptable and the Radian is configured to connect to it, the Radian will synchronize its own inverter with that source, and enable its transfer relay. At this point, the input and output are electrically common, and the inverter is basically passing through the input to the output.

The radian has many different input modes to enable various types of functionality with respect to the AC inputs. These input modes allow the radian to prioritize the use of battery power over the inputs, become grid interactive (buying/selling power or supporting the grid), reject any AC sources, and more. There are tons of applications using the various input modes, more information can be found via Outbacks documentation.

Inverter Stacking

One of the major selling points of these inverters, and the power pallet design is the ability to connect them in parallel in what Outback calls stacking configurations. This configuration allows the inverters to be connected in parallel via their proprietary hardware and interfaces. With this proprietary interconnection between inverters, the individual inverters are able to synchronize their outputs and supply larger amounts of available power to loads according to various settings.

The MATE3 tied to each individual inverter needs to be configured for that specific inverter and its role and the connected system, and the Hub10s on each pallet are used to coordinate and drive configurations to other pallets.

Outback uses the terminology of “master” to describe the unit that is driving the configurations and “slave” to describe the unit(s) that follow configurations commanded by the master. In this document, we will be using the terms leader and follower to denote this dynamic. The communication between a leader and the followers is meant to be handled by the Hub10. The Hub10 was designed to use static ports for the various connected devices, i.e. the leader is plugged into port 1, the AXS port/mate is plugged into another specific port, and followers are connected to the other ports.

While the Hub10 functioning in this way is great if the system is used and installed in a static manner, the “Power Pallet” platform we are working with is meant to be deployed. This means that those deploying the pallets will need to have a bit of experience with the systems and know not only how to physically connect the pallets to each other and the loads, but also how to configure them via the Mate3.

2.2.3 Use and Deployment of Pre-Existing Platform

This section will briefly describe the processes necessary to configure and deploy the original Power Pallet. Note that this section will not describe the process of pallet relocation as that is outside the scope of this project. It also assumes that physical connections regarding distribution to the loads are handled by the end user. These pallets use the L14-30 interface for power delivery. This standard is common across generators and other recreational power applications. This section also neglects the need for opening and closing physical breakers as their application and locations are not known by the reader.

For a single inverter, the configuration process is as follows.

1. Move RJ45 switch to “Leader” position
2. On the Mate3
    a. Go to Settings/Inverter/Stacking and set Port 1 as Leader
    b. Go to Mode and set as Backup
3. Switch inverter to On

For a single pallet, the process is pretty simple, but it is also important to note the need for passwords for the Mate3 as well as knowledge of menus and how to use the device.

For parallel pallets, the configuration steps are a bit more involved

1. Connect a cable from the AC load, Gen, or Grid plugs to a distribution panel
2. Connect RJ45 cable from follower out on each pallet to slave in on the leader pallet
3. Configure one pallet for “Leader” mode, and other pallets for “Follower” mode
    a. On leader pallet move RJ45 switch to “Leader” position
    b. On follower pallet(s) move RJ45 switch to “Follower” position
    c. On the Mate 3 of the leader pallet, go to Settings/Inverter/Stacking and set Port 1 as “Leader” and Port 2, 3, etc. as “Follower”.
    d. On MATE3s, go to Settings/Inverter/Power Share and set Follower Ports as sequence of 1, 3, 5, etc. and Leader Port 1 as the highest number in the sequence. The sequence increments by two because there are two power modules inside each Radian inverter. In this way, all pallets are continuously active and share load. Setting the master’s Power Share Level to 0 makes the followers disabled until the load increases to require their operation.
4. Turn on each pallet and couple their outputs.
    a. Flip the switch of each inverter to ON.
    b. Turn on load breakers on each pallet.
    c. Check at the distribution panel that all pallets are in phase with each other, then turn on load breakers. If they are out of phase, a large AC current will result and there will be an audible hum before the units fault.

As you can clearly see, the configuration of 2 or more of these pallets can be quite involved.

2.2.4 How can we Simplify Configuration?

The goal of our project is to simplify the process required to physically interconnect these pallets. To simplify the configuration we first need to understand what exactly makes the configuration

complex in the first place. The answer we found to this question came from one specific mechanic, the need for a user to choose a single pallet as a leader. This choice increases the complexity in the following two areas.

1. Physical connections of the proprietary network via the HUB 10

Since a user needs to choose a pallet to be configured as a leader, and the HUB10 has static ports set up for network connections, the user must physically connect the follower pallets to specific ports on the Hub 10 of the leader. The other area that we aim to improve upon is the following.

2. Manual, role-dependent configurations via the Mate 3

Again, since the user needs to assign a pallet as the leader, the follower pallets need to be configured accordingly via the Mate 3. These configurations need to be done manually, on each pallet before they can be used.

At a high level, our initial design plan was to focus on solving the problems outlined above. We needed to figure out how to drive configurations automatically and work around the static port assignments of the HUB 10. If we were able to do those things, the user would only need to connect the pallets to the loads and run a network cable across them.

2.3 Evolution since 491

Overview

At the end of 491, our design consisted of using a Raspberry Pi and an AXS Port to replace the Mate3 entirely. The AXS Port allowed us to make some configuration changes to the Outback Hardware connected to the Hub 10, using the Raspberry Pi. The Raspberry Pi's were also connected to each other via network switches which allowed pallets to communicate with one another.

The Rpis were meant to use this communication to automatically elect a leader, and drive the configurations to set up the individual pallets as a leader or a follower based on the manual process. Near the end of CPRE 491, we learned that the variables allowing us to change the stacking configurations of the inverter were unable to be configured through the AXS port interface. This roadblock created many new obstacles that we would spend the next semester trying to solve. The major issue here being AC Source synchronization of multiple parallel pallets.

2.3.1 Software Evolution

Our overall software design has matured since CPRE 491. While we had a rough approximation of what each section of our software would look like, many of the specifics were not established. At the end of 491 we had software to read and write values from the AXS port using SunSpec and an algorithm design for leader selection.

Since then we have integrated the CycloneDDS library for network communication and implemented a main control program to manage the network and SunSpec communication. All of the network communication programming was implemented from the design stage this semester. Additionally, high-level functions were created to configure sets of parameters at once, called pallet profiles, and a mock pallet object allowed us to test the software independently from any hardware changes.

2.3.2 AC Source Synchronization Problem

Traditionally this problem would be solved via the Mate3 and Hub10, Outback had procedures in place to handle this problem, but since we were eliminating the Mate3 and working around the Hub10 these were now problems we needed to address.

AC Source synchronization describes the need for two AC sources connected in parallel to match in voltage, frequency, and phase. If two sources are not synchronized, large currents will occur as the source tries to overpower each other and level out. These large currents can cause damage to the inverter and connected loads; Luckily Outback does a good job of protecting their inverters against these types of events. Much of 492 was spent looking for and testing ways we could achieve this synchronization. None of the members of the team had much experience working with power systems such as this, nor had taken classes related to such systems.

Details surrounding the various paths taken and tested will be outlined in Appendix II and in section 4.4.4. It is important to note that the methods we used and the results we found are not approved by the Outback as their products were used in ways they did not intend nor test, and could result in damage to loads, the inverters, or those working with the system.

2.4 Design Implementation

2.4.1 Communications and Hardware Setup

This section describes our contributions to the microgrid pallet design in an attempt to make deployment and configuration simpler. This section assumes the reader has a solid understanding of the existing system described in section 4.2. Here we will provide a brief description of each addition to the pallet and its function. The figure below shows the current state of the pallets with our added hardware.

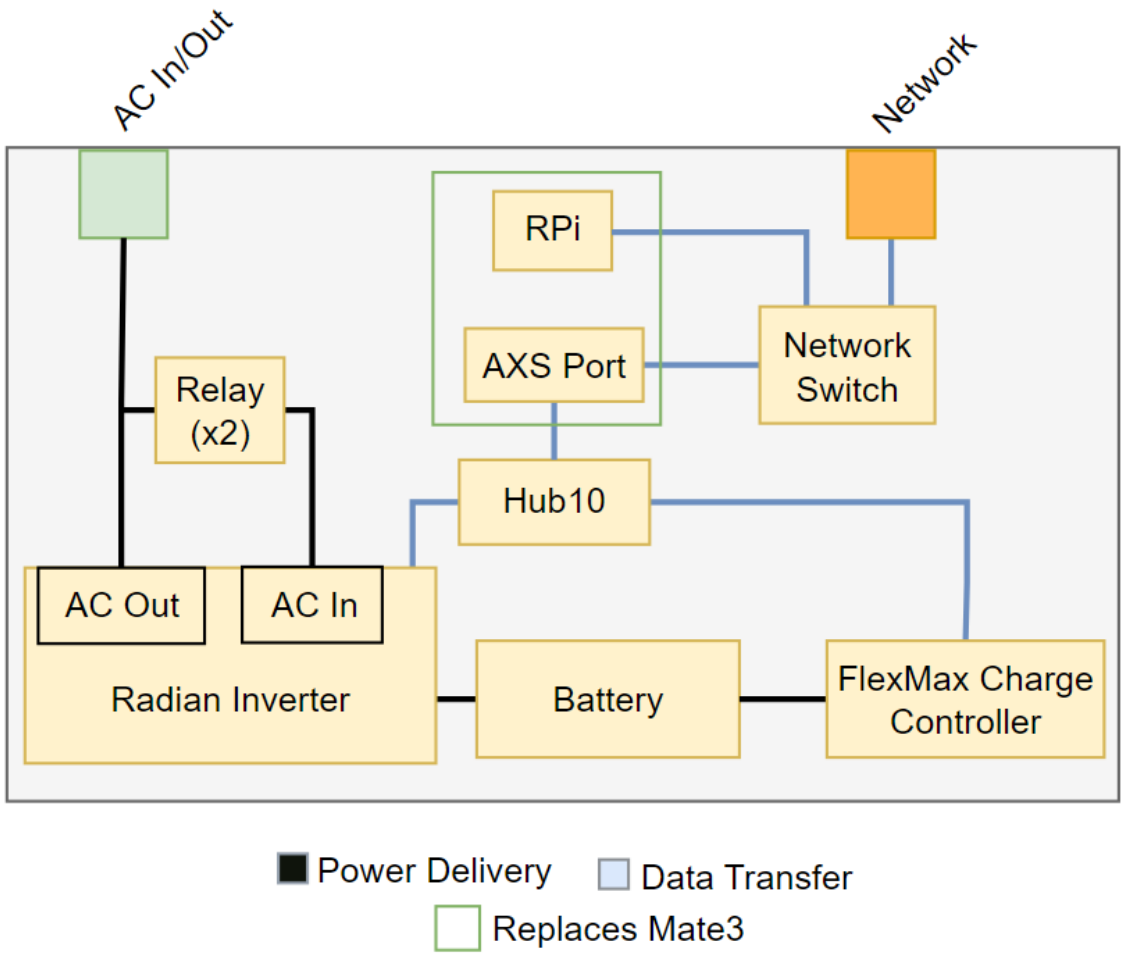

Figure 2.4.1: Connections of a PowerPallet

Raspberry Pi

The Raspberry Pi serves as our embedded computational unit. It runs the control software we designed and participates in a distributed network to operate the home pallet. The Raspberry Pi is also responsible for configuration changes on its home pallet.

Ethernet Switch

An ethernet switch on the pallet connects the AXS Port and Raspberry Pi to the larger network of pallets. Allows for traditional TCP/IP communication across pallets, breaking us out of the need to communicate through static ports on the Hub 10.

OutBack AXS Port

The onboard AXS Port serves as a network target for the Raspberry Pi and converts SunSpec configuration messages into a format that is understood by OutBack devices (Modbus). The AXS port uses these messages to drive configurations and read data from any necessary devices

connected to the Hub 10. The AXS Port and Mate3 share a port on the Hub10, so in this instance, the Mate3 is eliminated from the system.

Power Relays

Power relays are being used to provide a path from the output to the input of the inverter. The relays can be triggered via a 3.3v signal from a microcontroller and/or the Raspberry Pi. This connection allows AC output to be bridged and accepted as an AC input via the Radian. The use of this connection will become more apparent in later sections. Each pallet needs a relay for each leg or phase, in this case we have two 120v legs so we use 2 relays for each pallet.

2.4.2 Software Implementation

Overview

The Raspberry Pi software must serve multiple tasks: report status information over the microgrid network, monitor failure states, check for proper physical connections before startup, write configuration changes to the pallet, and output up-to-date information to operators. This is accomplished through a combination of libraries, custom abstraction modules, and a main control loop.

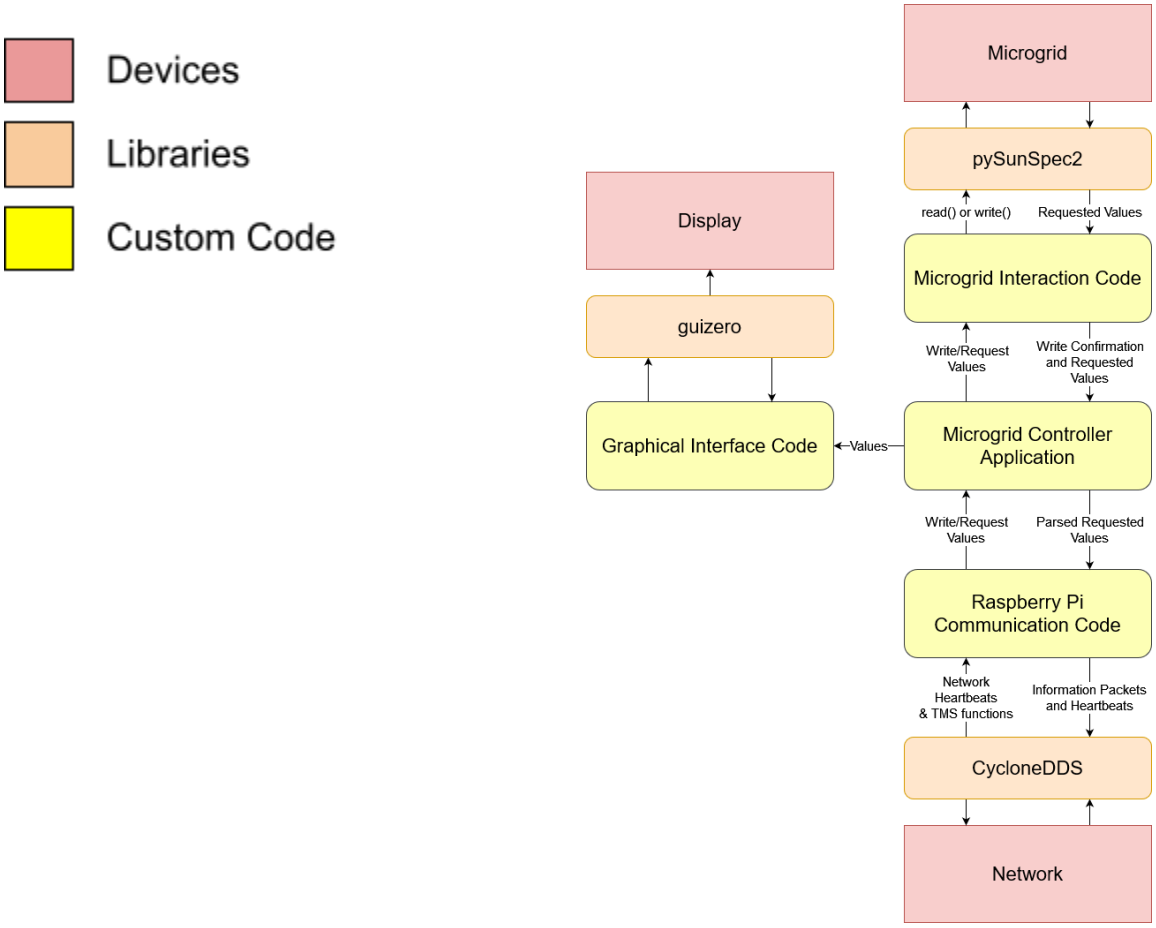

Figure 2.4.2.1: Software Module Outline

Microgrid Interaction Code

For tasks requiring the reading and writing of variables used by devices attached to the Microgrid, we used a Python library named pySunSpec2. pySunSpec2 was created by The SunSpec Alliance to provide ways to standardize communication between Distributed Energy Resource components and smart grid applications. By utilizing this library, we can simply communicate with all devices attached to the inverter and Hub10 through the AXS port. The Microgrid Interaction Code is a class that handles and simplifies this communication even more for easy use by other developers on the team.

The Microgrid Interaction code will not only allow the reading and writing of various values, but also verify that the changes made have actually occurred. It does this by then reading the value back from the system after it is written and throwing an error message if it is not the same.

Raspberry Pi Communication Code

For tasks requiring synchronization or messages across the Raspberry Pi network the CycloneDDS library is used. This is a Python library for OMG DDS, and was initially chosen since the TMS uses DDS. While inspired by the TMS, the messages we send are simplified and unique. The Raspberry Pi Communication Code defines several classes that provide simple management of the election, heartbeat, and sensing processes. Receiving of these messages is all done in the main control loop through polling, and directed to the appropriate management object.

In our implementation, there are three Topics for communication (these can be thought of as communication channels). Each Topic has its own data type including the sending and receiving identification numbers and additional data depending on the purpose.

The Election topic is used to determine a leader among pallets on the network, and includes a single integer in addition to send/receive IDs to determine the message type. The election process follows the Bully algorithm, in which the highest value ID is awarded leadership.

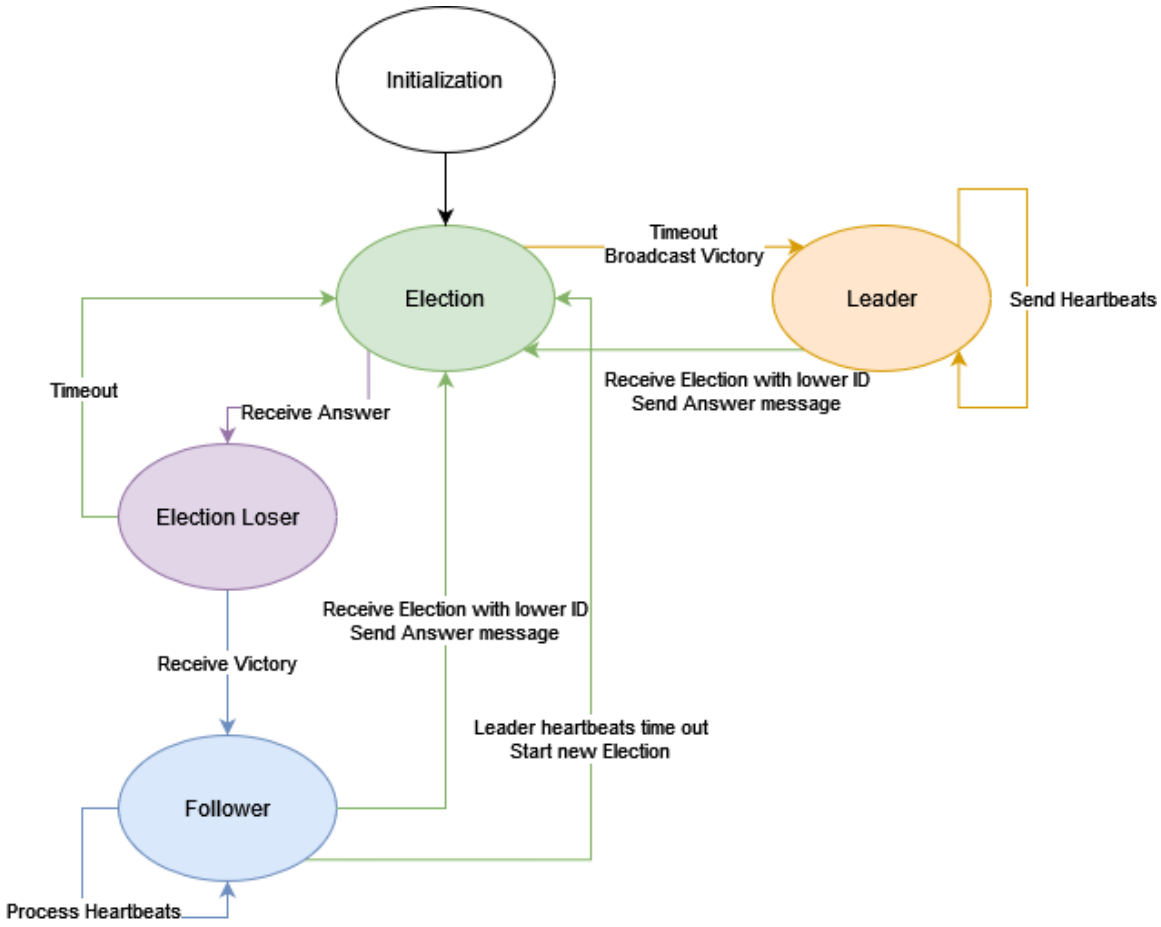

Figure 2.4.2.2: Election State Machine

A new election is run every time a pallet joins the network, or if the current leader fails to heartbeat.

The Sensing Topic is used to control the startup procedure of the pallets. It consists of control messages while the main state machine gathers data and determines if the home pallet has suitable physical connections for a leader / follower designation. We were unable to determine the exact configuration sequence to have the pallets match phase, however below is a flow chart showing a possible implementation. More details about this specific implementation can be found in Appendix II and IV.

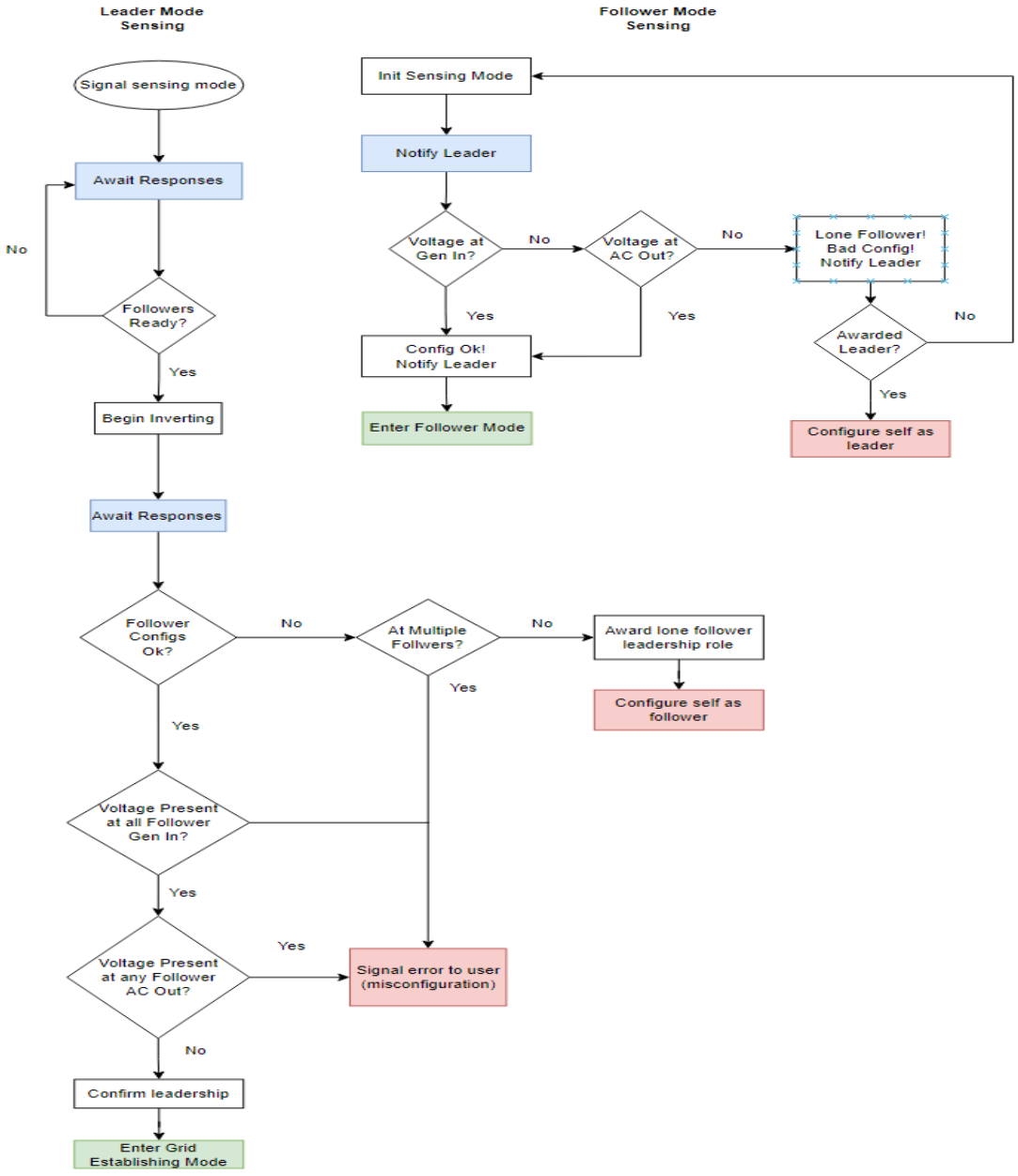

Figure 2.4.2.3: Example Configuration Flow

The Heartbeat Topic is used to periodically read key parameters from the home pallet and send this data to other pallets. This serves the purpose of keeping updated information across the whole network, and keeping each pallet aware of others' connections. The leader pallet receives heartbeats from every follower pallet, and sends heartbeats to every follower pallet. Each follower only sends and receives heartbeats from the leader. The key parameters are read and heartbeat sent from a separate thread than the main control application.

If an expected heartbeat is not received for 10 seconds, a flag is set to indicate that the pallet has gone offline or is disconnected. If the disconnected pallet is the leader, a new election will begin.

Graphical Interface Code

The Graphical Interface Code controles the microgrids display and is run from the Raspberry Pi. It fetches values from the microgrid and displays them for users to better monitor the microgrid when it is being operated. It also serves as a way to notify the users if an error happened. To do this it uses the GuiZero library to create and update the screen.

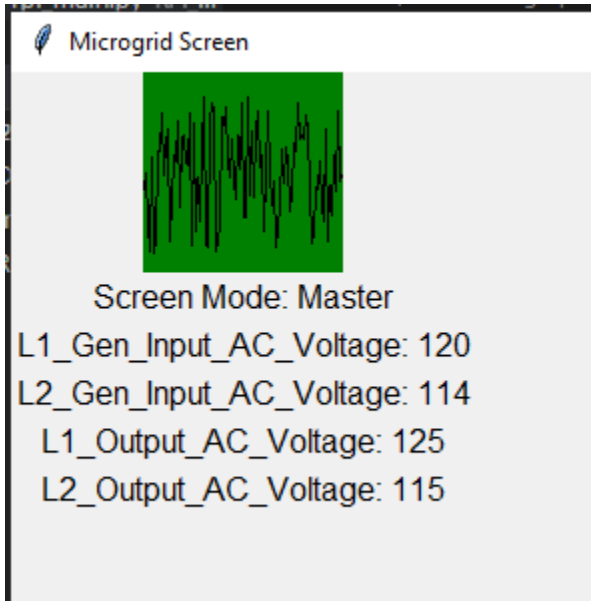

Figure 2.4.2.4: GUI display

Microgrid Controller Application

The Microgrid Controller Application is the entry point for the Raspberry Pi software. It performs high-level management of read and write operations to the home pallet and network. It can be divided generally into a setup phase and a loop phase.

In the setup phase, the control loop prepares communication objects and global variables. First, a unique ID is made. This can be from either the MAC or IP address of the Raspberry Pi, or a random number. The random number is meant for testing, and should not be used in deployment. The AXS Port device is created and scanned to obtain initial values, and the home pallet is put into a safe configuration. After these steps, a connection to the CycloneDDS network is made and the Topics for sensing, elections, and heartbeats are created. An initial election is run before entering the loop phase.

The loop phase contains polling code that runs each loop and a state machine that acts based on the current state. Every pass through the loop, the election Topic is checked for new messages. Any new election will activate the ELECTION. The heartbeat Topic is also checked for new messages, and all heartbeat flags are evaluated for a heartbeat failure.

The state machine is meant to read various points on the pallet to confirm it is in an acceptable physical state, then write configuration values based on its leader or follower status. The exact process for this requires more testing to achieve acceptable phase matching, however the sensing

procedure in Figure 2.3.2.3 is currently implemented in software as an example. The loop phase can be modeled as the composite state machine in Figure 2.4.2.5, with relatively stable states being ERROR, LEADER, and FOLLOWER.

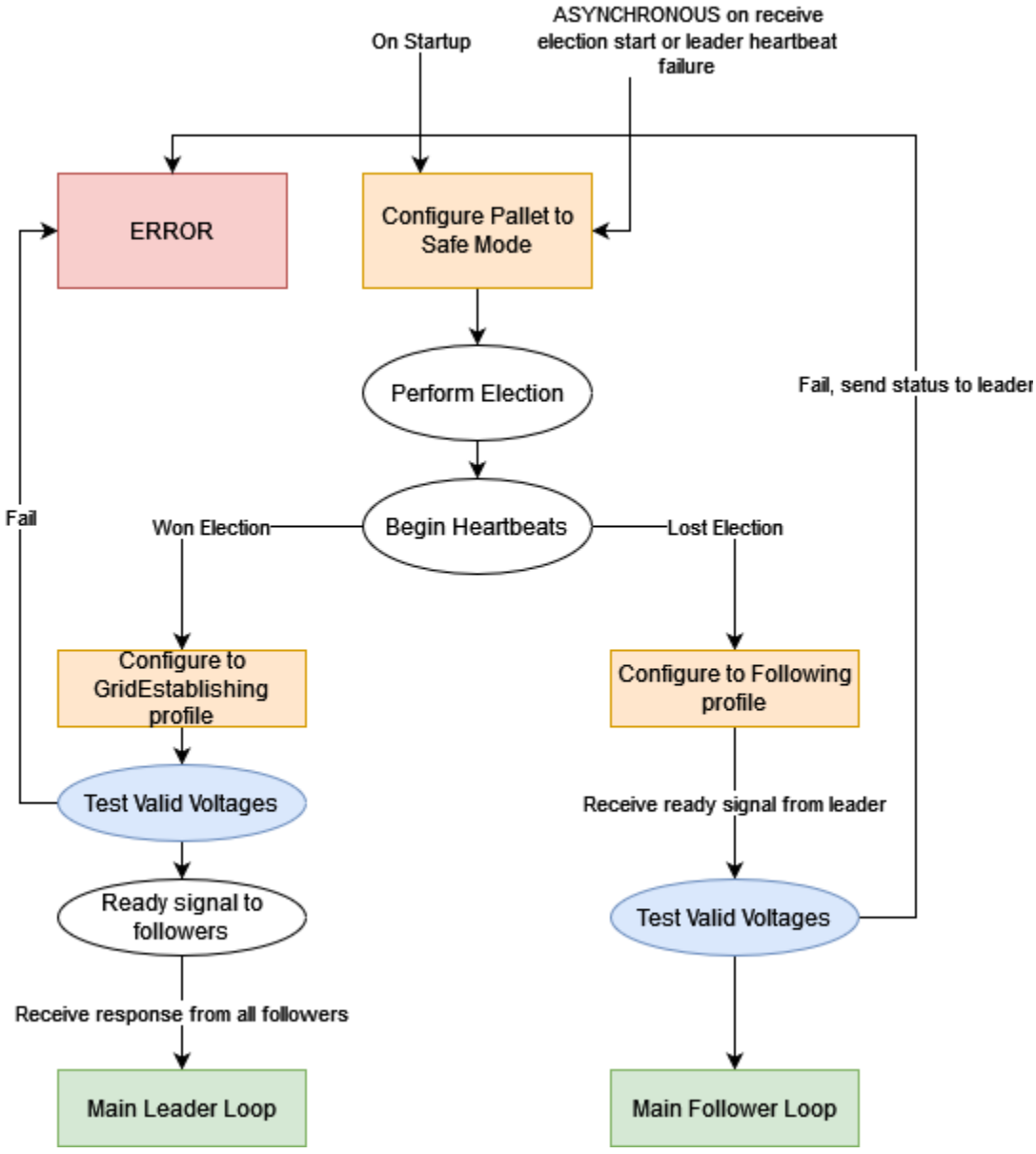

Figure 2.4.2.5: State Machine Representation of Control Application

2.4.3 Network Configuration

For successful communication over the network without a router, the AXS Ports require configuration with a static IP address. Each AXS Port in the network must have DHCP turned off and a static IP address set. The default address is 192.168.0.64, and it is recommended that this address is incremented for each AXS Port.

Each Raspberry Pi on the network (or laptop for testing) that needs to interface with an AXS Port also requires configuration with a static IP address. An example configuration is shown below, and the addresses for these devices should range from 192.168.0.2 - 192.168.0.63, inclusive.

```
43  # static IP configuration:
44  interface eth0
45  static ip_address=192.168.0.3/24
46  #static ip6_address=fd51:42f8:caae:d92e::ff/64
47  static routers=192.168.0.1
48  static domain_name_servers=192.168.0.1
```

Figure 2.4.3.1: Example Static IP Configuration

Since each Raspberry Pi is meant to configure the pallet it is physically on, the static IP address for the AXS Port should be saved as the environment variable “AXS_IP_ADDR”. This is retrieved in the main software loop to establish pallet communication.

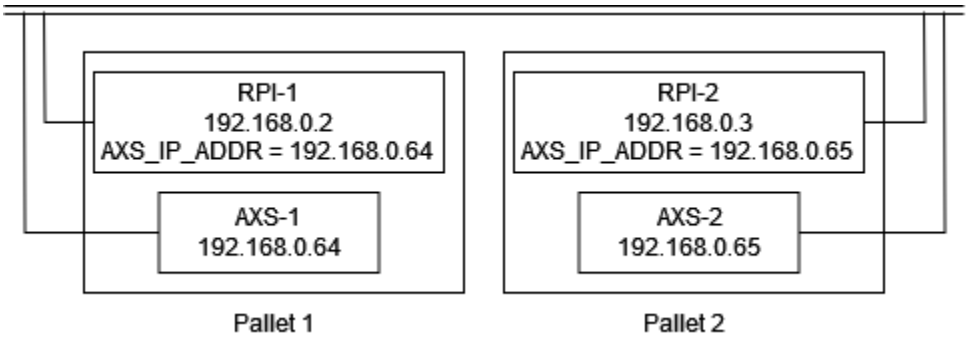

Figure 2.4.3.2: Network Configuration for 2 Pallets

2.4.4 Hardware Implementation

We know we are able to drive configurations to various devices on our pallet, but what configurations do we need to change? This depends on how exactly the pallets are connected to one another and the loads. Unfortunately, we were unable to find a configuration that we were confident would satisfy all the requirements of our client, but we were able to get close. In this section we will describe the setup of the design closest to meeting our client's needs.

The most difficult part of the hardware configuration was figuring out how to ensure the AC outputs of the pallets would be synchronized and could then be distributed to the loads. The Outback Radian GS8048a has the capability to synchronize outputs across various pallets when configured in the ways specified by Outback. These configurations are referenced in various places across section 4.2. Since we were straying far from those configurations we had to resort to other methods and figure out how to make it work ourselves. Extensive documentation surrounding different configurations we tried and tested are expanded upon in Appendix II.

The most promising solution to the problem of AC source synchronization we found was using the GridTied input mode of the Radian inverter. The traditional use of the Grid-Tied input mode is to

use an AC input to charge the batteries and supply power to the connected loads. This mode also can be used to interact with the utility grid to buy and sell power.

We found that when using this mode, the inverter would synchronize itself with the AC input, before it would accept it. The inverter would then enable its transfer relay and pass the AC input to its AC output terminals.

So far, this isn't particularly interesting, but we found if we then enabled inverting and physically removed the AC input, the inverter would quickly begin inverting while preserving the characteristics of the lost AC input. At this point, we were able to connect the output of the inverter to the source it originally used as its input and successfully power some small loads.

The key idea here is that we can use this behavior to pseudo-synchronize two ac sources. If we were to use the pallet elected as the leader to begin inverting and establish a "grid", the characteristics of that grid could be used by the follower for synchronization. Oscilloscope traces in figures show the initial phase shift and transient currents across two pallets directly before and after this process.

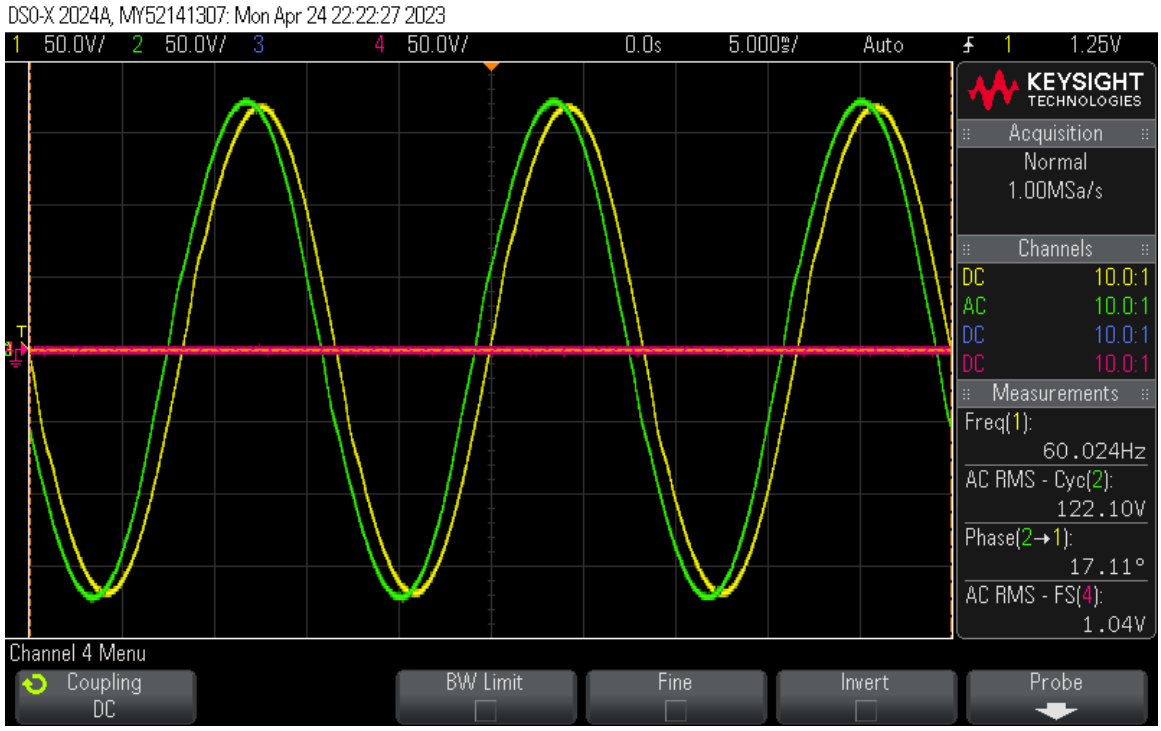

Figure 2.4.4.1: Pre-Synchronization Phase Shift (17.11°)

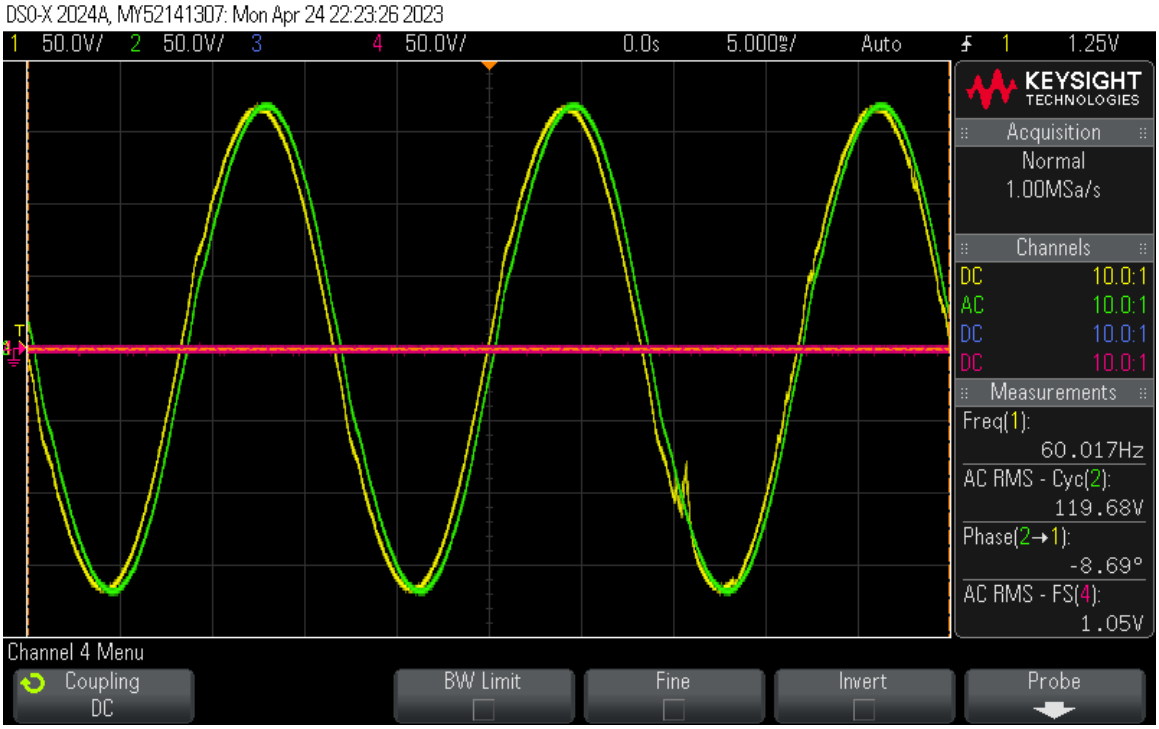

Figure 2.4.4.2: Post-Synchronization Phase shift (-8.69°)

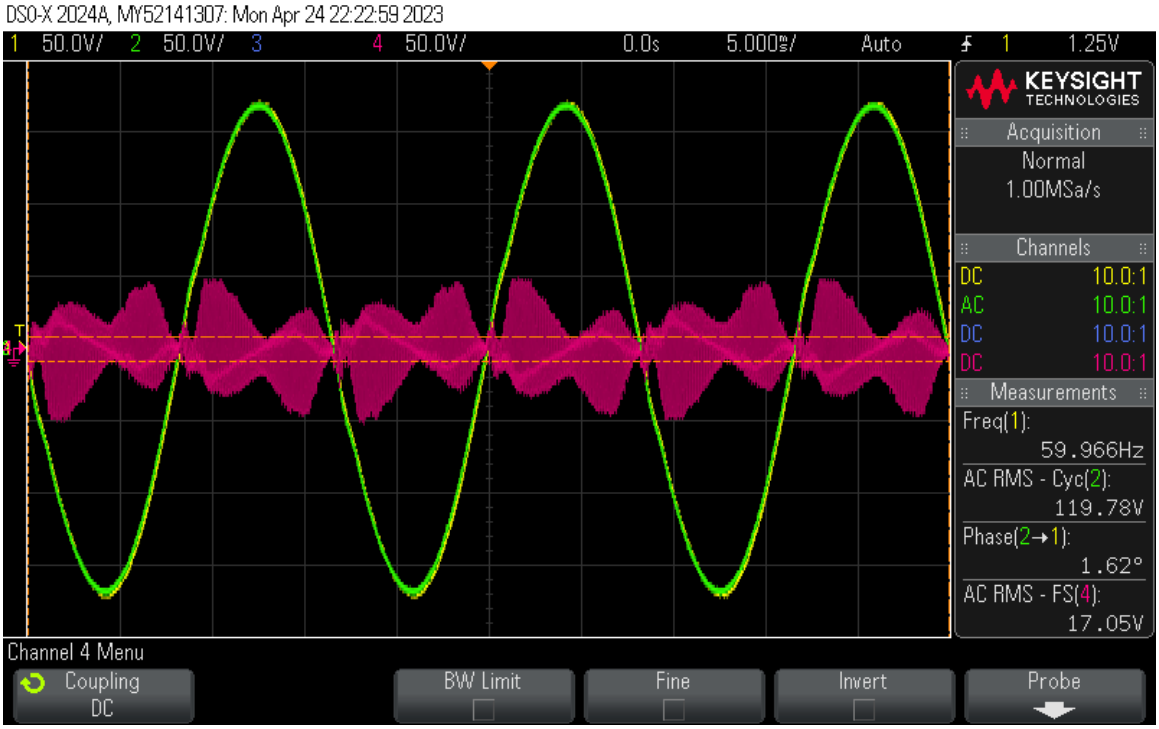

Figure 2.4.4.3: Post-Synchronization Transient Currents

In theory, this is a great idea, but in practice, we ran into some issues regarding phase loss/drifting over time. It is also important to note that we were limited in our ability to thoroughly test this solution with more than two pallets, and across more diverse load characteristics. More details surrounding these issues can be found in section 5.7, Hardware Testing.

With a method to somewhat synchronize followers with a leader figured out, we needed to be able to do so autonomously. Recall that we are electing a leader, so our hardware configurations ideally can handle the AC source synchronization in a similar manner. This could have been done by connecting all of the AC inputs and Outputs across the inverters and using relays to route power depending on the pallet's role, but we found that it was sufficient to simply provide a path from the AC output to an input on a given inverter. We then use a set of relays to isolate the terminals when necessary. The figure below shows a simplified diagram of the physical connections.

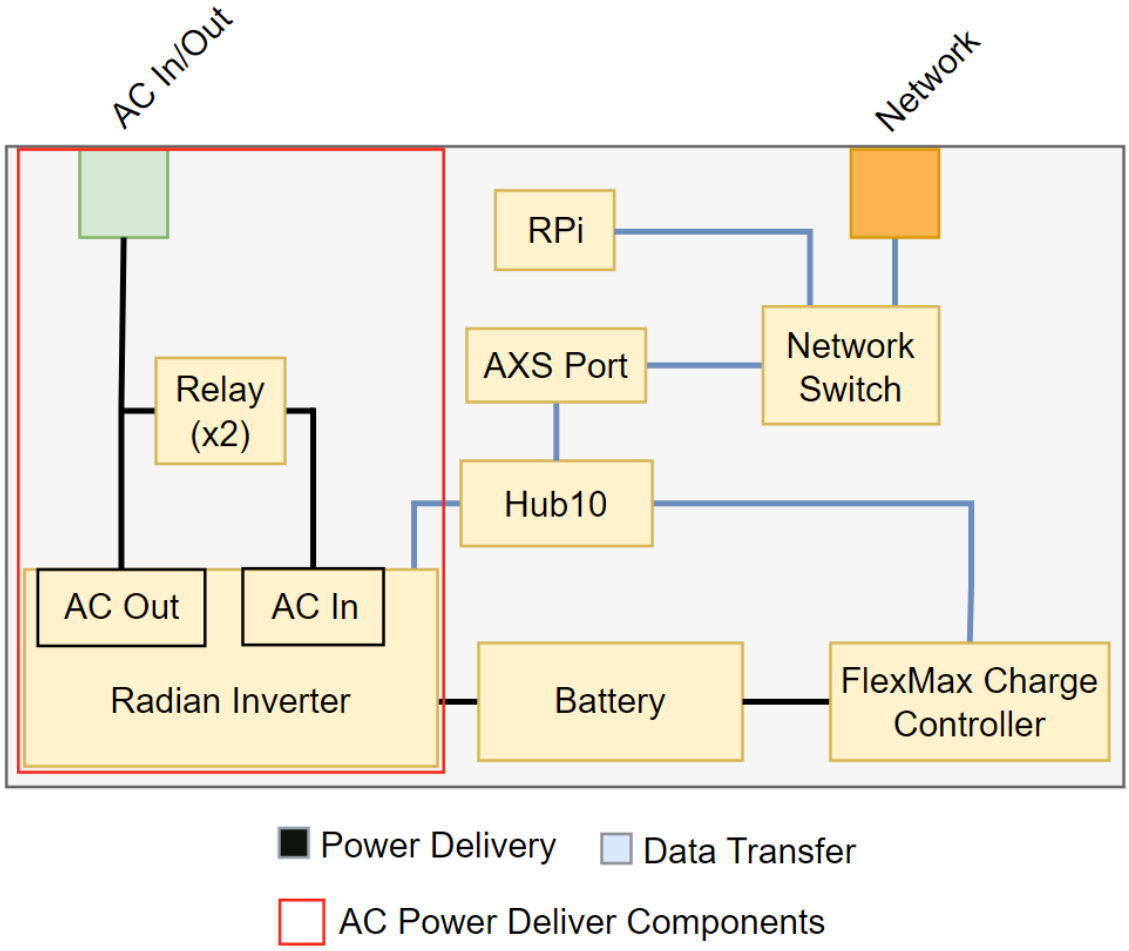

Figure 2.4.4.4: Simplified Diagram of Internal Pallet Connections

These relays allow us to simply connect all of our outputs together to a distribution panel. Once the leader begins inverting, the followers can close their relays allowing the leader's output to be seen at the input of the followers.

The relays are necessary as the connection between AC input and AC output causes problems in a couple different situations. On startup and when dropping AC inputs, the Inverters seem to check for shorts across the input and output terminals and will fault if connections exist across them. A secondary need for these relays is the behavior of the inverter when dropping the AC inputs via a configuration interface (Mate3 or the AXS port). When the input is dropped through the Mate3 the output from the inverter is shifted around 40 degrees, this is a much higher shift than the -6 to -8 degrees measured when the relays are used to isolate the input from the output. This 40-degree shift causes large currents and would trip breakers and or fault inverters causing them to stop inverting.

2.4.5 Hardware Configuration

The inverter configuration changes needed to support this functionality are quite minimal. For the leader, they really just need to start inverting, and potentially signal to the followers that they

should be prepared to synchronize. The followers control flow is a little bit more involved. The leader and followers would follow the steps outlined in the flowchart below.

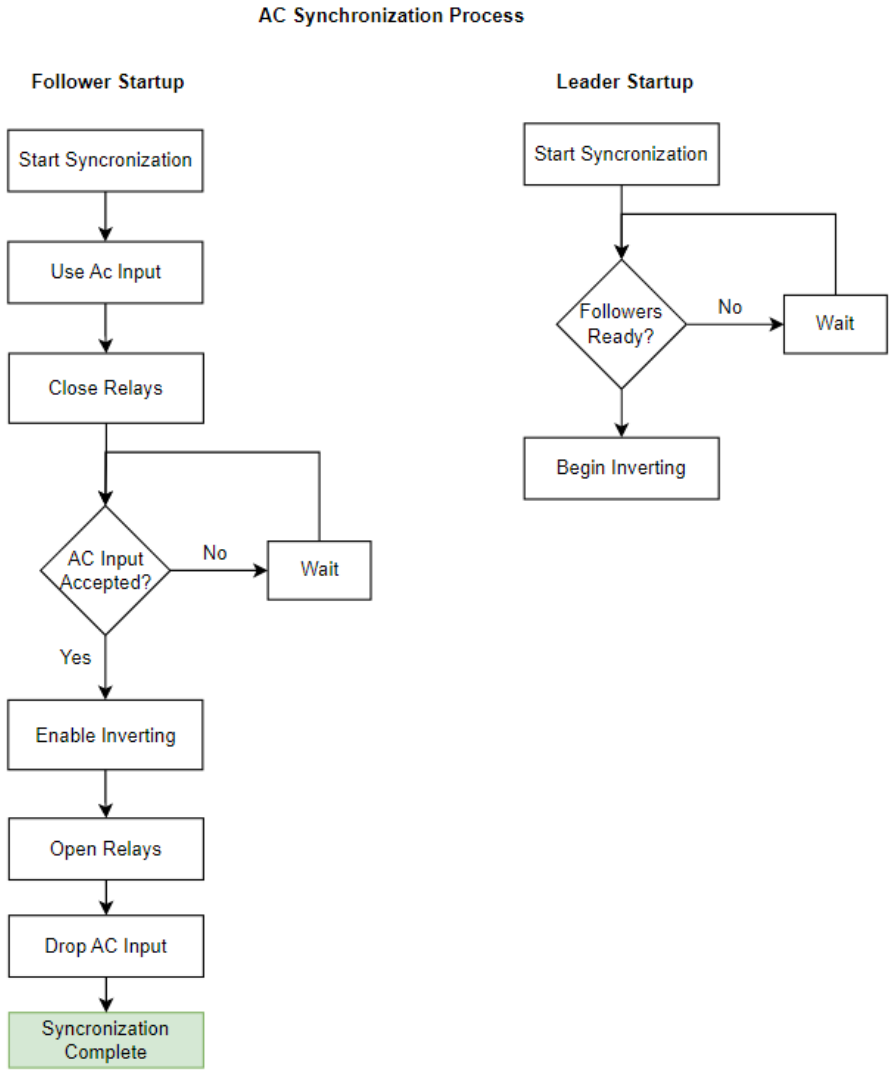

Figure 2.4.5: AC Synchronization Process

The results of manually testing this flow will be presented in the testing section. There you will be able to see additional testing results when this logic was used across the two available pallets.

Configuration Modes/Profiles

Each pallet, irrespective of its role, should be able to support/change the following configurations. The pallets should always default to safe mode in the event of any faults or errors present within the inverter. Please note that these configuration profiles are specifically for the connected outback devices and do not include the states necessary for the AC relays.

Additional functionality regarding the use of the AXS port and Outback blocks is available in Appendix IV. It outlines using similar methods for sensing physical configurations.

Safe Mode (Default)

This mode disables inverting and accepting of any AC sources and puts the inverters in a "safe" configuration. The inverters should put themselves in a similar state in the occurrence of a major fault, but we need to make sure that we are also supporting this behavior. Note: Inverters moving from any one configuration to another should always be configured or checked against the safe state before moving on to other configurations. This configuration should serve as an anchor that will undo any configurations driven in any other modes.

OutBack System Control Block (DID = 64120)

Start	Name	Description	Desired Value
7	OB_Inverter_AC_Drop_Use	1=Use, 2=Drop	2 (Drop)
8	OB_Set_Inverter_Mode	1=Off, 2=Search, 3=On	1 (Off)
9	OB_Grid_Tie_Mode	1=Enable, 2=Disable	1 (Enable)

Radian Inverter Configuration Block (DID = 64116)

Start	Name	Description	Desired Value
21	GSconfig_AC_Input_Select_Priority	0=Grid, 1=Gen	1 (Gen)
23	GSconfig_Gen_AC_Input_Current_Limit	Gen AC current limit	5
25	GSconfig_Charger_Operating_Mode	0=Disable, 1=Enable	0 (Disabled)
26	GSconfig_AC_Coupled	1=Yes(not impl.), 0=No	0
32	GSconfig_Gen_Input_Mode	0=Gen, 1=Support, 2=Grid Tied, ...	**2**
37	GSconfig_AC_Output_Voltage	AC output voltage	240

Grid Establishing/Leader (Initial Configuration)

Configuration used for the microgrid that will be driving the other grids. This mode will begin inverting, and its output will be used to set the phase for the other connected pallets. Only one grid in the system should be using this mode.

OutBack System Control Block (DID = 64120)

Start	Name	Description	Desired Value
7	OB_Inverter_AC_Drop_Use	1=Use, 2=Drop	2 (Drop)
8	OB_Set_Inverter_Mode	1=Off, 2=Search, 3=On	3 (On)

Follower (Initial Configuration)

This configuration is used by all microgrids not establishing the grid. The mode will use its AC inputs and run in a grid-tied mode. It uses this AC input to synchronize itself with the "grid". The array of following pallets get their phase from the grid establishing pallet. They work together to support the load using the leader as a source of truth for phase and voltage.

OutBack System Control Block (DID = 64120)

Start	Name	Description	Desired Value
7	OB_Inverter_AC_Drop_Use	1=Use, 2=Drop	1 (Use)

Radian Inverter Configuration Block (DID = 64116)

Start	Name	Description	Desired Value
26	GSconfig_AC_Coupled	1=Yes(not impl.), 0=No	0
32	GSconfig_Gen_Input_Mode	0=Gen, 1=Support, 2=Grid Tied, ...	2

Using the configuration parameters above, and driving them through the AXS port should allow the desired result. While we have not been able to test this using the Raspberry Pi and AXS port, we were able to do testing while manually driving these configurations through the Mate3

3 Testing

3.1 Software Testing

3.1.1 Leader Selection

The leader selection process can be done completely in software since it only requires the use of CycloneDDS on a network. To simulate multiple pallets running on a network, the election program was run in multiple terminals simultaneously.

The first tests conducted were between two processes in which ID values were assigned manually and the messages were monitored through print statements and compared to and hand-calculated expectation of the messaging order. Additional processes were added after success was observed.

Edge cases were tested by interrupting and canceling some processes during various points in the election process to ensure it never gets locked. This is an important feature of the algorithm since we must be able to recover from an unreliable network.

After confirming success through multiple terminals, the same code was run on the Raspberry Pis over a local network with random ID values to test that the election still worked.

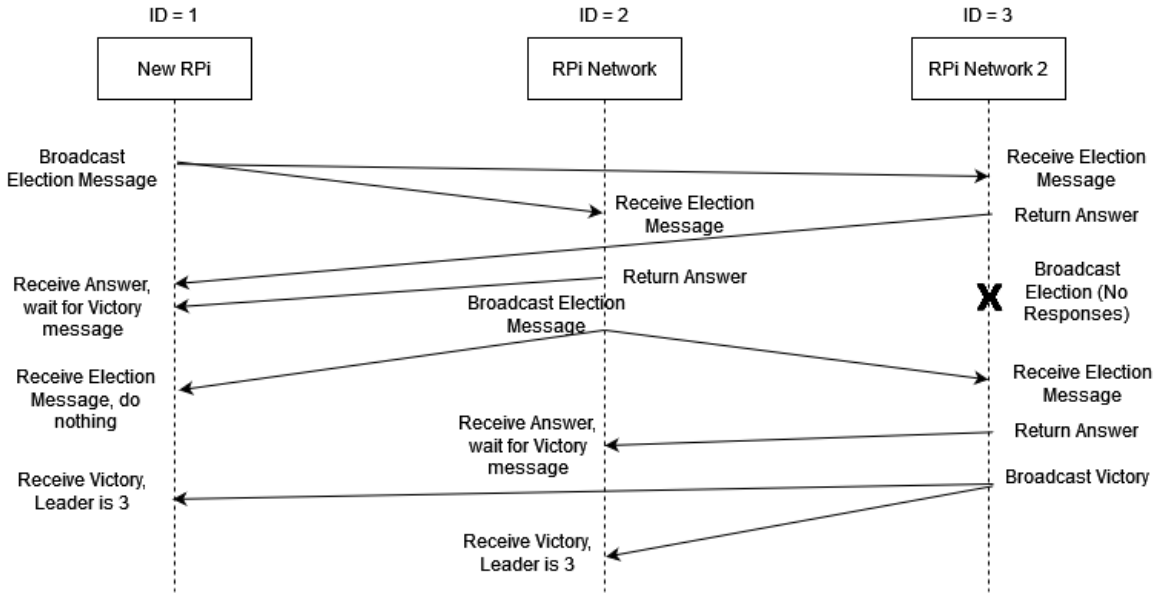

Figure 3.1.1: Example Hand Calculated Election

3.1.2 SunSpec Communication

In order to test the communication between our code, pySunSpec2, and the AXS port, Visual Studio Code, the Python Visual Studio Code extension, and various test scripts were used. By stepping through test scripts that scanned for devices, read values from those devices, and wrote over said values, we were able to analyze exactly how pySunSpec2 organized the data read from those devices.

The most common utilization of these tools was to set a breakpoint after pySunSpec2 scanned the AXS port. At this point, we could see all the variables that pySunSpec2 could use and manipulate as well as their current values.

3.1.3 Configuration State Machine

To isolate the configuration state machine in our software from possible errors in our SunSpec communication, a mock object of the AXS port device was created. This object imitated SunSpec communication by changing its own values, reading back placeholders, and responding to method calls in the same way the AXS port would.

The state machine was tested by priming this mock pallet with values to simulate a physical configuration and using it as our device in the configuration process. Using this method we confirmed that the state machine acted as expected when using 3 or more pallets.

3.1.4 Heartbeating

Heartbeating occurs mostly over the network but also requires reading status values from the pallet. To test this function, we used the mock pallet object to simulate reading status values and ran the heartbeating code in multiple terminals while printing each time a heartbeat was sent or received, and the connection status for monitored pallets.

Initial tests showed that heartbeat threads were being duplicated, resulting in threads with no reference that could not be canceled.

After success was shown with two processes, additional terminals were added to observe heartbeat behavior with many "pallets". The heartbeats were started and ended randomly to make sure all timers were working properly.

Once success was shown on the terminals, the same code was run on the Raspberry Pis over a local network to test that the heart beating still worked.

3.1.5 Display GUI

During the testing phase of the microgrid project, manual testing was performed using Visual Studio Code to verify the values displayed on GUI are the expected outputs. The manual testing was focused on ensuring that the GUI was responsive and displayed the correct information from the microgrid system.

3.2 Hardware Testing

Overview

Extensive testing was performed to reach the design decisions made for inverter configuration and interconnections of the pallets. More information about this process can be found in Appendix II.

The biggest issue that we needed to address was regarding the synchronization of AC inputs. Just to reiterate, this synchronization refers to the voltage, frequency, and phase of the sources. Any sort of mismatch across those 3 factors leads to large and potentially damaging inrush currents in the system. In order to monitor these characteristics we used an oscilloscope with 10x probes to observe the 120v waveforms across our AC sources, this gave us a view of the frequency, voltage, and phase shift of the ac source. It should be noted that the AC sources were actually 240V, we measured from one of the hot legs from each source (the legs that were supposed to be in phase), to a common neutral. We also used a FLUKE 80i-600A AC Current Clamp/Probe. This current clamp is meant to output 1mA/A with a working voltage of 750V rms max. In traces that show current, the two legs of voltage are electrically common, taken from two different places on essentially the same wire and thus items such as phase can be ignored. In the traces containing zero current, the two legs being measured are electrically isolated, one coming from each inverter.

3.2.1 Configuration Testing

Once we were able to see and understand what was going on with the power being delivered we began doing various tests across multiple physical connections, and inverter configurations. Appendix II briefly discusses some of the configurations that were not successful, and our findings. Here we will focus on our testing specifically for our choice in the final design.

Using the Gridtied AC Input Mode for Source Synchronization

The Gridtied input mode was what we ended up going with in the end, as it seemed to have the most promise. We landed on this mode while testing the sell functionality of the Grid Tied mode (more information can be found in Appendix II). For some reason, this was the only input mode that actually accepted the AC input from the other inverter. This was strange, as the Generator input mode specifically was supposed to have the most relaxed AC input parameters, but even it wouldn't accept the source. Nevertheless, once we were able to get an AC source accepted, we noticed that when that source was manually removed, the follower inverter would begin inverting with similar characteristics. The figure below shows a couple of waveforms captured shortly after the source was disconnected through different attempts.

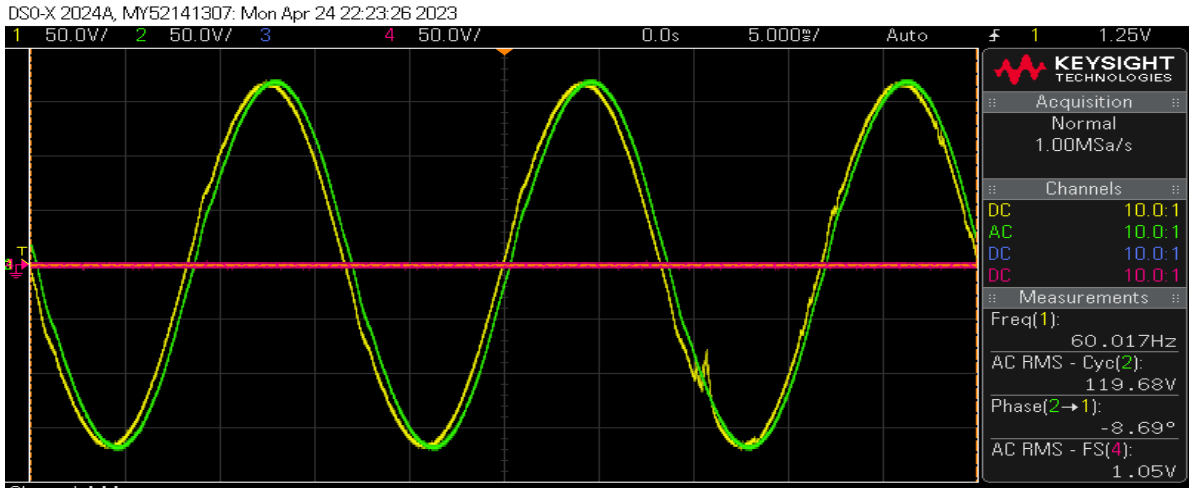

Figure 3.2.1.1: Capture immediately after source disconnection

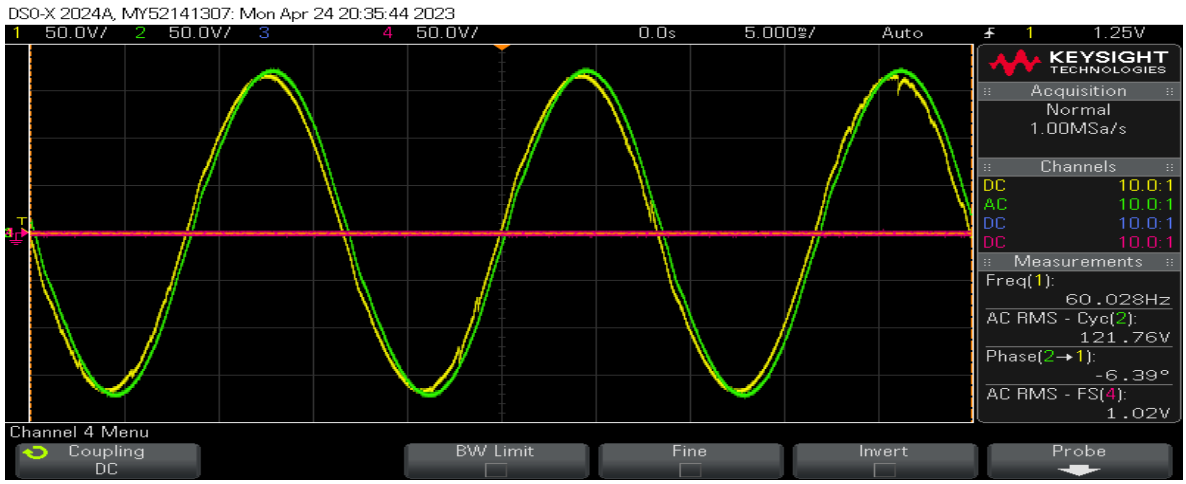

Figure 3.2.1.2: Capture 5 mins after source disconnection on a different test

As you can see, the waveforms are pretty close to each other, although not perfectly synchronized. The next figure shows the initial currents present with a parallel connection established across them.

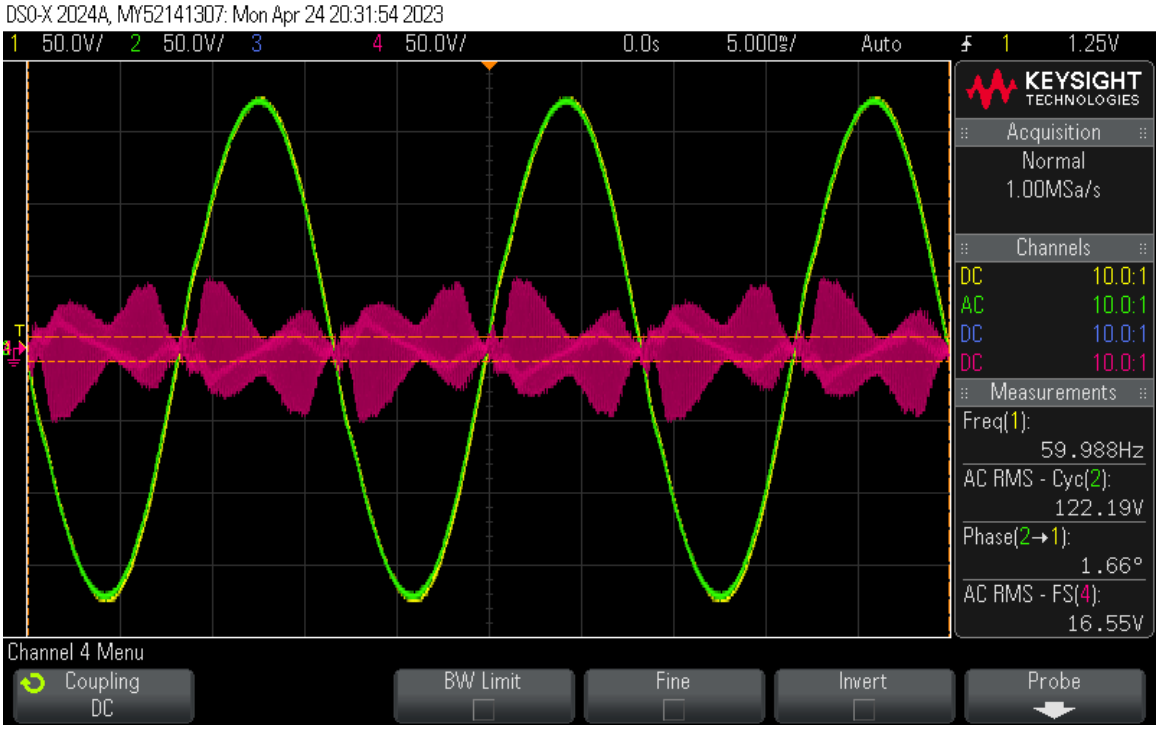

Figure 3.2.1.3: Initial currents present after synchronization

At this point, both inverters were showing very low power being delivered or transferred, not much above the 70-80w that is present whenever inverting is traditionally enabled.

3.2.2 Phase Drift

After seeing the results of the previous section we want to study the behavior of the sources over time. We specifically wanted to observe the phase shift and the currents it would cause over time. We ran 3 tests specifically with the goal of measuring these shifts under different conditions. The first condition was observing phase shifts between the pallets when they were not connected.

We initially connected the leader and performed our synchronization algorithm manually through the Mate3 and using a DC power supply to drive around 3.3v to the relays. After synchronization was complete we then disconnected the leader from the follower and captured traces every 5 minutes for a total of 30 mins.

Below in figure 3.2.2.1, there is a graph outlining the observed results, followed by an animation of the traces over time. Please note that any animations may not work correctly depending on how you are viewing this report. If that is the case, all relevant information can be gathered from the provided graphs and other figures. A live version of this document with animations enabled can be found here.

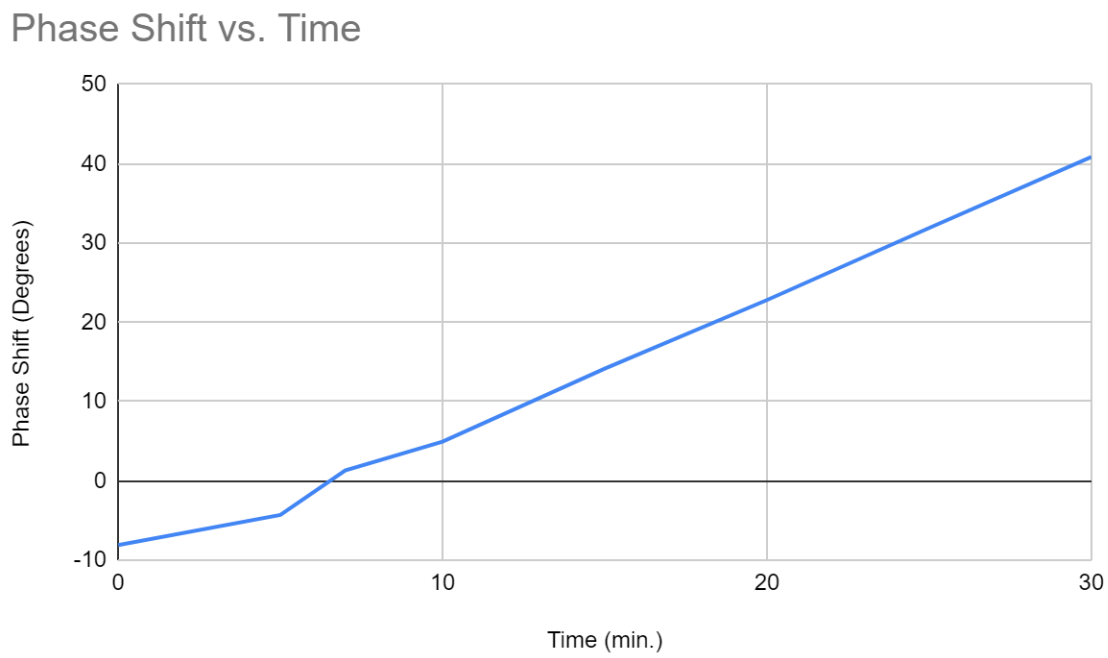

Figure 3.2.2.1: Phase Shift vs. Time Graph

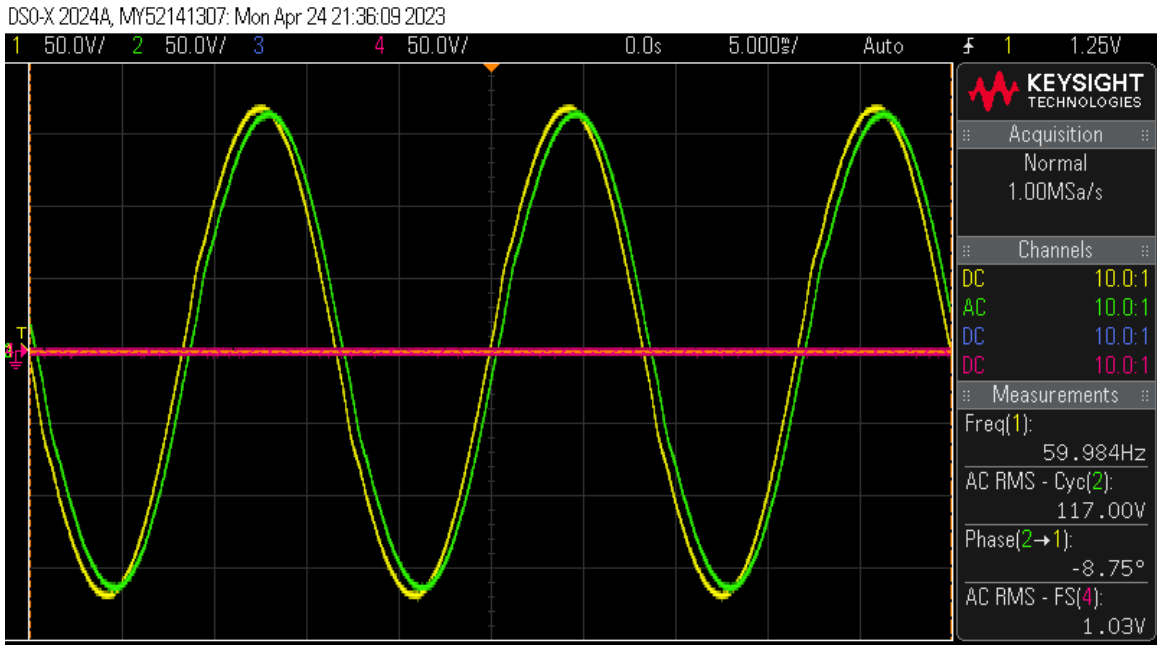

Figure 3.2.2.2: Phase shifting animation

We then restarted the process, but this time we wanted to also capture traces of the currents occurring between the sources. In order to do this we would leave the units disconnected, and every 5 mins reconnect the pallets in parallel to capture a trace of the resulting current waveforms. The figure below is another animation highlighting the changes in currents as the ac source waveforms shifted.

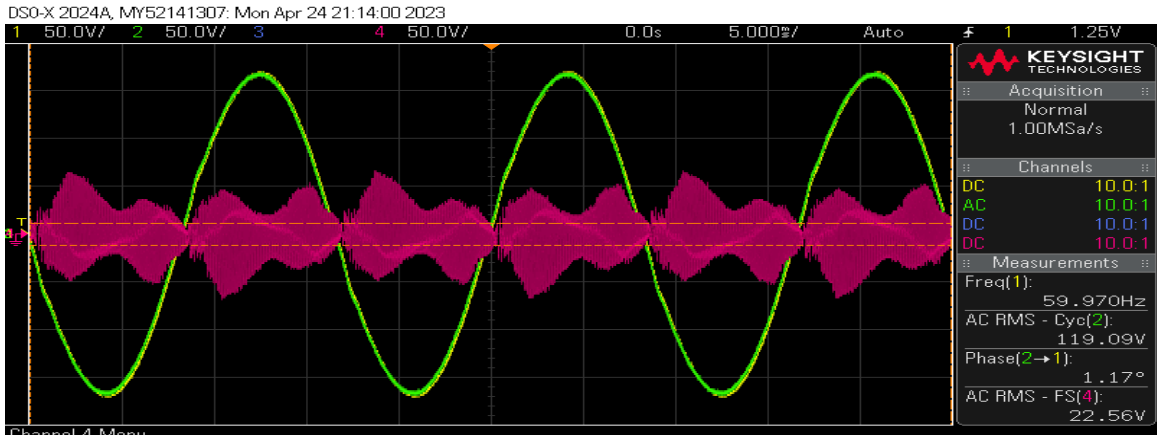

Figure 3.2.2.3: Transient currents occurring with phase shifts (animated)

Interestingly enough, this specific test was cut short. At the 20 min mark, after attempting to capture a trace of the current, the sources seemingly detected the large currents and adjusted their inverters to resynchronize with each other almost exactly as they had through our synchronization process. In this instance the sources went from 42 to -8.57 degrees out of phase. The traces below show the phase shifts and currents before and after this shifting occurred. At the time when this happened, we did not have AC coupling nor any grid interface protection settings enabled and were quite surprised to see this behavior.

Figure 3.2.2.4: Voltage traces immediately before and after source corrections

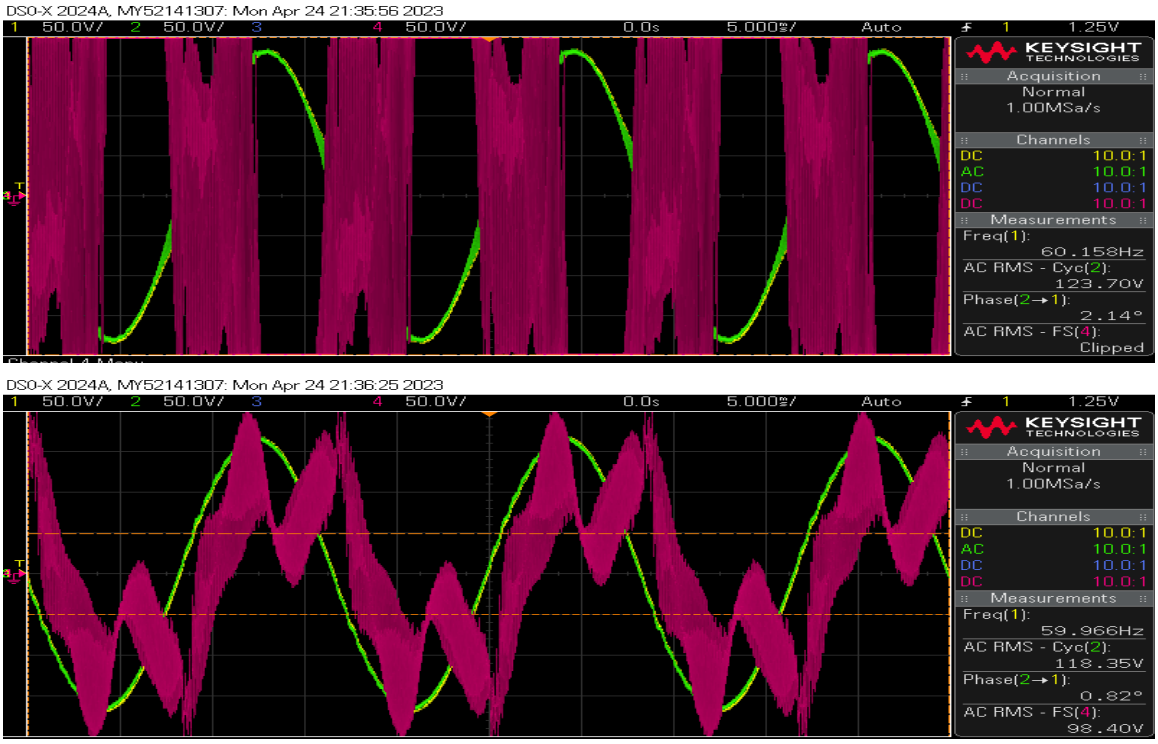

Figure 3.2.2.5: Current traces immediately before and after source resynchronization

One last test we wanted to see was if/how the behavior of these sources would change if we left them connected for the duration of the test. We wanted to know if the currents would eventually become too great, and trip breakers, or if the inverters would fault. Fortunately neither of these things happened.

What we observed was the sources would initially drift, to an offset of just around 12 to 13 degrees, but would then hold steady around that mark. The graph below highlights the difference in phase shifting in the two different scenarios we tested.

As the test progressed, we noticed that one inverter would show it was absorbing power and charging the batteries while the other was delivering it. This behavior was very similar to what we had seen when using the selling feature of the grid tied input mode. We did verify that the pallet delivering power was not “selling”, and both chargers were disabled through their respective configuration controllers.

The animations below show the shifts of voltage and currents over the course of our 30 minute testing period.

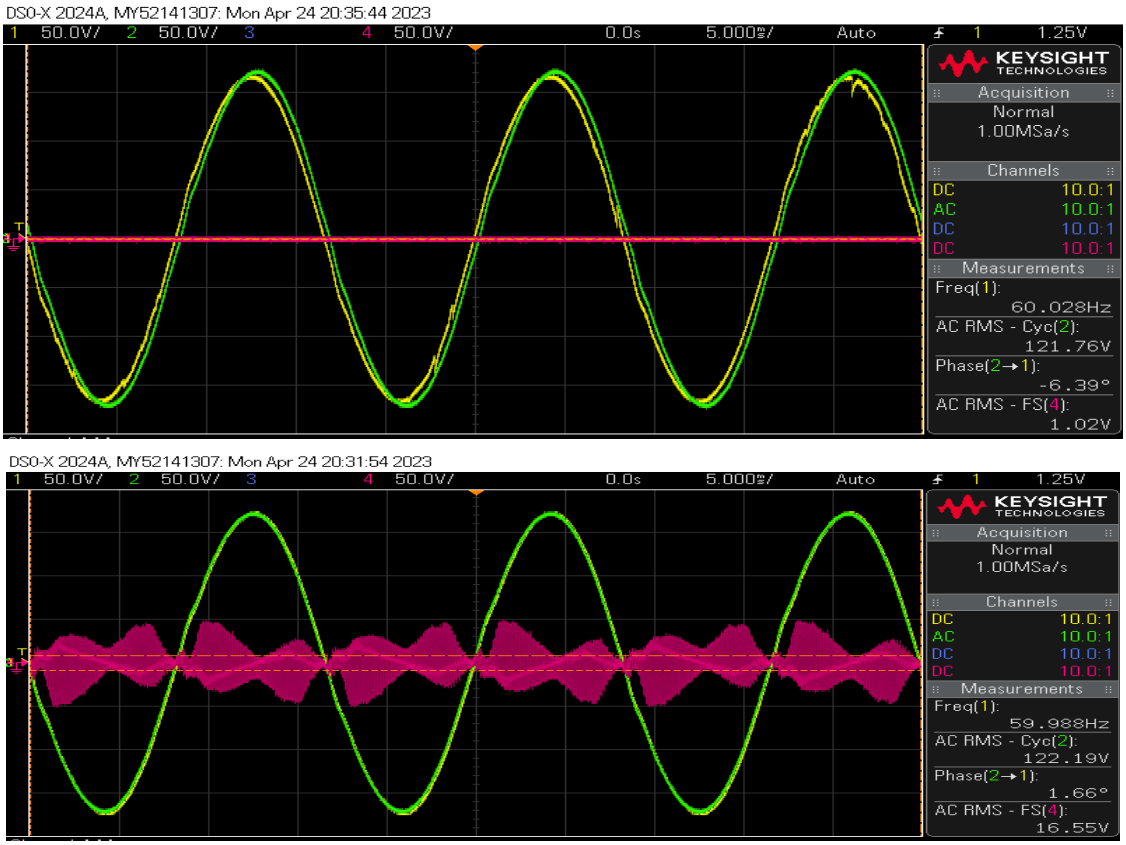

Figure 3.2.2.6: Voltage and current traces of sources connected while testing (animated)

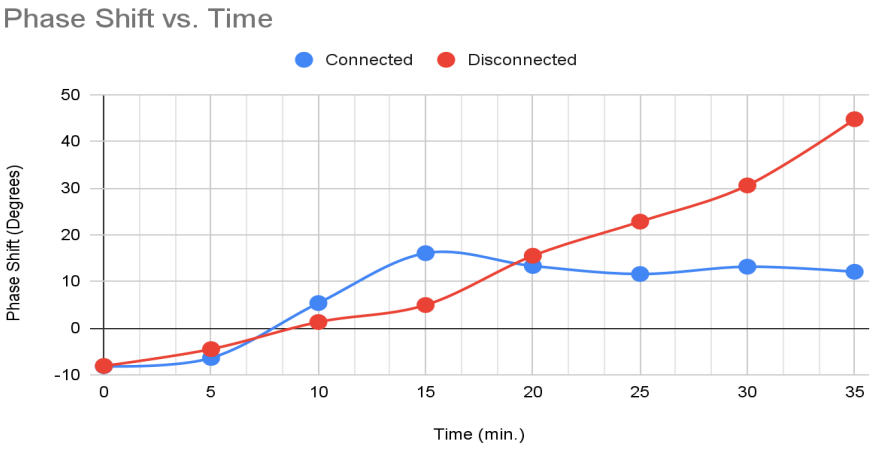

Figure 3.2.2.7: Phase Shift vs. Time across sources that were connected vs. disconnected

The figure above shows the difference in phase shifts between the two trials. If the inverters were left disconnected it is reasonable to assume that the two sources would continue to drift away from each other; It was very promising to see that when connected the sources seem to stay somewhat close to one another. It would be interesting to see this behavior over an extended period of time.

The figures below show traces of the current at the end of each test, the top is with the outputs disconnected, and at the bottom they are connected.

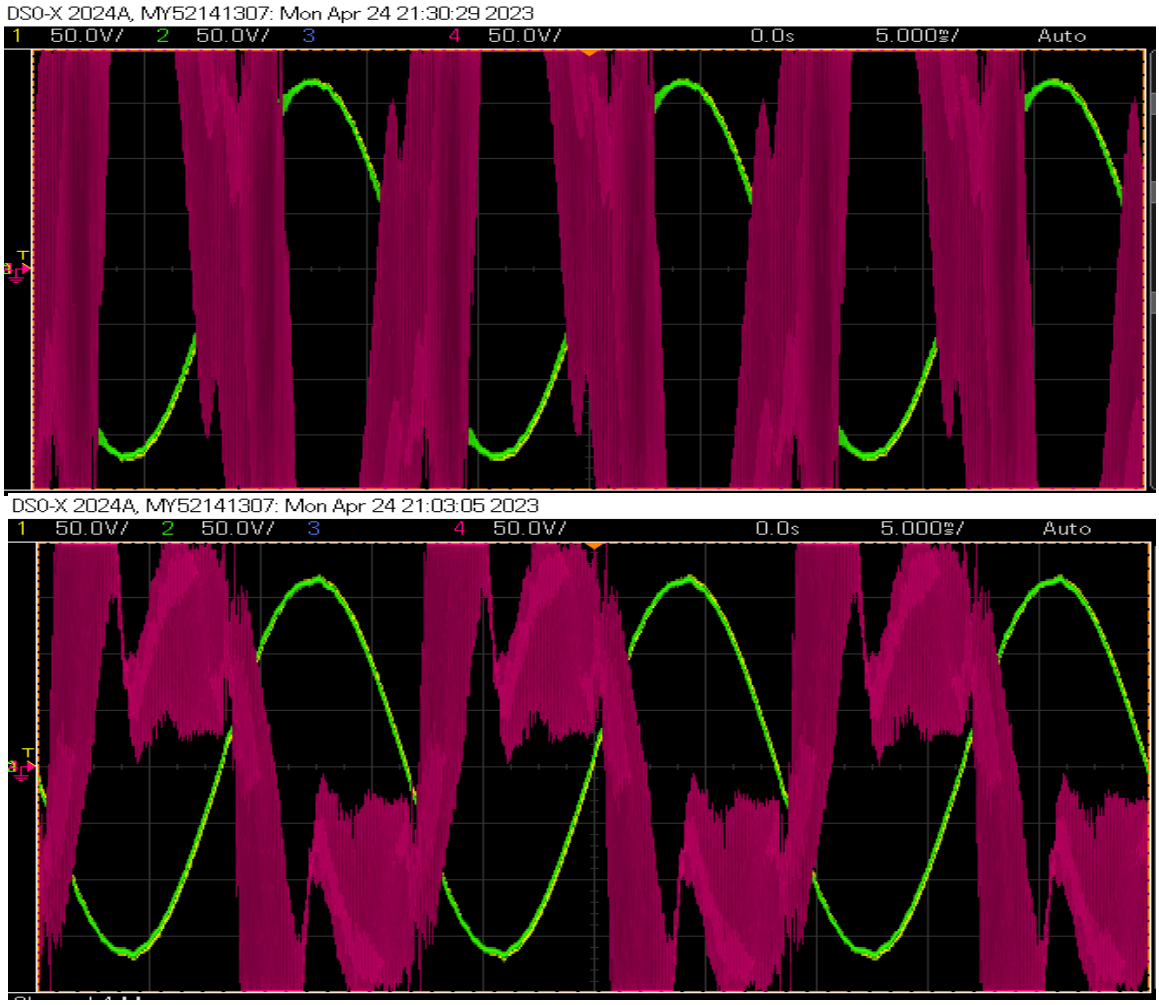

Figure 3.2.2.8: Current traces at end of phase shift testing

Trying to Understand the Results

Upon further research it seems that two connected parallel sources should aim for acceptable voltage, frequency, and phase angle differences of typically +/- 5 percent, +/- 2 Hertz, and +/- 5 degrees, respectively. Here we are a bit out of range of the acceptable phase angle, but further testing is needed to see what effects this would have on our loads. We did use this method to deliver power to some small, completely resistive loads ~1.3kW, and didn't seem to have any problems. We believe that issues could arise when trying to power larger inductive loads but did not have a chance to do any testing during this time.

Also, within our testing it is important to note that our batteries across the inverters were not physically connected nor at full and/or equal charges. We are not sure if these variables would have adverse effects on the testing/results, but there is a high probability that it could have some effect. It is possible that this could solve issues regarding charging across pallets and reduce the currents

that are present. We want to reiterate that our group is not well versed in the realm of power systems or electrical engineering, but we feel there is a lot of potential here for students that are.

4 References

Technical References: Nick David(Client), Mathew Wymore(Advisor)

M. Bush, J. Corman, and B. Fox, “SunSpec Energy Storage Models.” .

“SunSpec DER Information Model Specification.” SunSpec Alliance, San Jose.

“SunSpec Device Information Model Specification.” SunSpec Alliance, San Jose.

“SunSpec Modbus IEEE 1547-2018 Profile Specification and Implementation Guide.” SunSpec Alliance, San Jose.

“SunSpec Technology Overview.” SunSpec Alliance, San Jose.

Appendix I: Operation Manual

Before working with any of the Raspberry Pis, the file RPI_IMAGE.img must be flashed onto their SD card. This can be done in a variety of ways, but we use Win32DiskImager.

Network Setup

Create a static IP network plan where each device has a unique address. Devices on the network will consist of Raspberry Pis, AXS Ports, and possibly laptops for debugging. Assign each RPi/Laptop an address between 192.168.0.2 and 192.168.0.63. Assign each AXS Port an address between 192.168.0.64 and 192.168.0.255. For this manual, Device A will refer to devices physically located on a pallet notated as Pallet A.

Static IPs for RPis

1) On RPi A, open the file /etc/dhcpcd.conf. Identify the following text block, and replace RP_ASSIGNED_IP with the static IP address of this Raspberry Pi:

```
# static IP configuration:
interface eth0
static ip_address=RP_ASSIGNED_IP/24
static routers=192.168.0.1
static domain_name_servers=192.168.0.1
```

2) Reboot the Raspberry Pi.

Static IPs for AXS Ports

1) After the static IP has been configured on RPi A, connect AXS Port A to port 1 of the Hub10 on Pallet A. Form a local network with only the AXS Port and RPi A.
2) Open a terminal and navigate to ~/sdmay23-36/RPI_CODE
3) Start a Python shell by typing "python3", execute the following commands replacing AXS_IP_ADDRESS with the static IP address of AXS Port A:

```
import sunspec2.modbus.client as client
from microgrid_interaction import *
SetIPAddress("AXS_IP_ADDRESS")
exit()
```

Running the Automated Software

1) Open a terminal on the Raspberry Pi and navigate to ~/sdmay23_36/
2) In the file rpi_main.py, modify the LIVE_INVERTER constant to equal True, unless debugging software.
3) Execute the program with the command "python3 rpi_main.py"

Operation Instructions for Synchronization Via Grid Tied Mode

Initial configurations of the pallets

1) Both pallets operate in the Grid Tied mode
2) Relays should be initially open
3) AC coupling should be disabled on both pallets
4) Neither pallet should be inverting
5) Follower pallet should be setup to use AC inputs
    a) Accessed via AC input hotkey on the Mate3

Synchronization Steps

1) Ensure AC output of leader and follower pallets are connected in parallel
2) Close relays for battery connections on both pallets
3) On the leader, close the relay for the AC output
4) On the leader pallet enable inverting via the switch and/or inverter menu
    a) Accessed via the inverter hotkey on the Mate3
5) Observe LED for AC output on both the leader and the follower, ensure it is on
6) On the follower, close the relays labeled Gen Ac Input, and AC Output
7) On the follower pallet, supply 3.3 - 24v DC to both relays to close them, providing a path from the AC input to the output terminals.
    a) For now use a DC power supply
8) Wait for the follower to accept its AC input from the leader (the AC input light on the Mate3 will stop blinking)
    a) At this point, power should be passing through the transfer relay of the follower
9) Enable inverting on the follower pallet, the same way you did in step 4
10) Remove the power supply from the relays on the follower pallet, this causes them to open and isolates the followers AC input from its the output
11) On the follower, drop the AC input from the AC input menu, this opens the internal transfer relay, allowing the inverter to begin inverting.
12) The follower should now be supplying power in parallel to the leader

Appendix II: Previous Designs

Upon discovering that we couldn't modify the Radian stacking order from the AXS Port, a temporary design was made that would detect the physical setup of a pallet and apply configuration parameters accordingly. A state machine and outline of the hardware is shown below.

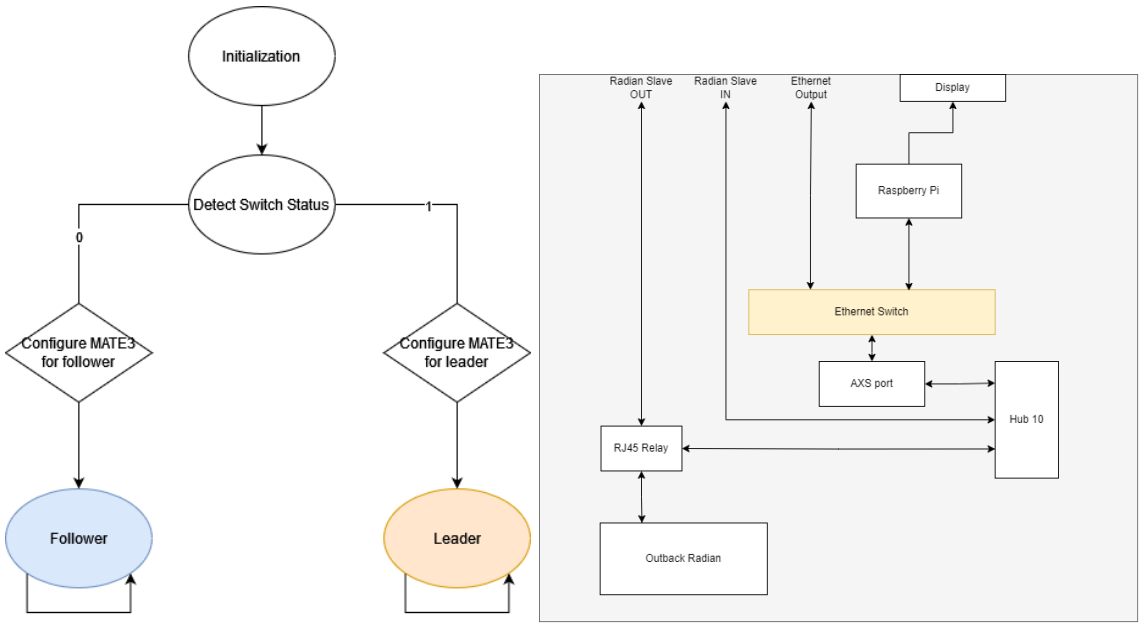

Figure A2.1: Hardware Sensing Design

Previous Hardware Configuration Directions

Support/Grid Zero Input Modes

The first approach we explored was using a function provided by Outback called Offsetting. This offsetting function is meant to allow an inverter to connect to an AC source, and offset the power going to the loads with access DC power. This function is available across multiple AC input modes, but we were not able to get the inverters to accept an AC input in any of these modes. We are not sure why this is the case, but some of the options provided in the menus look like they are designed for sources around 120v not the 240v we are using.

Also, if we were to use this approach, it would require a minimum of 5+ amps to be drawn continuously from the leader for each follower in the Support input mode, and that assumes the inverter can limit current going through the transfer relay. From our experience, any time we were to exceed current limits we set, we found that a fault would occur, but no excess power was ever supplied by the batteries to offset the load. It is very possible that we had incorrect configurations, but we decided there were other more promising avenues to explore.

AC Coupling

The radians come equipped with an AC coupling option. This option is designed to increase the frequency of the inverter's output to drive other connected inverters to adjust their outputs. Any

other inverters that may be connected to the output of our inverter that are IEEE 1547 compliant, should then adjust/shift their frequency based on their freq/watt compliance settings.

The radian inverters we are using do in fact follow the IEEE 1547 standard for inverters, not only that, they also support specific grid interface protection settings including freq/watt adjustments. Even with all of this, unfortunately, we were unable to get the outputs of the inverters to sync up. The figures below show some traces captured while the inverters attempted to synchronize and couple, but in the end, were unable to do so successfully.

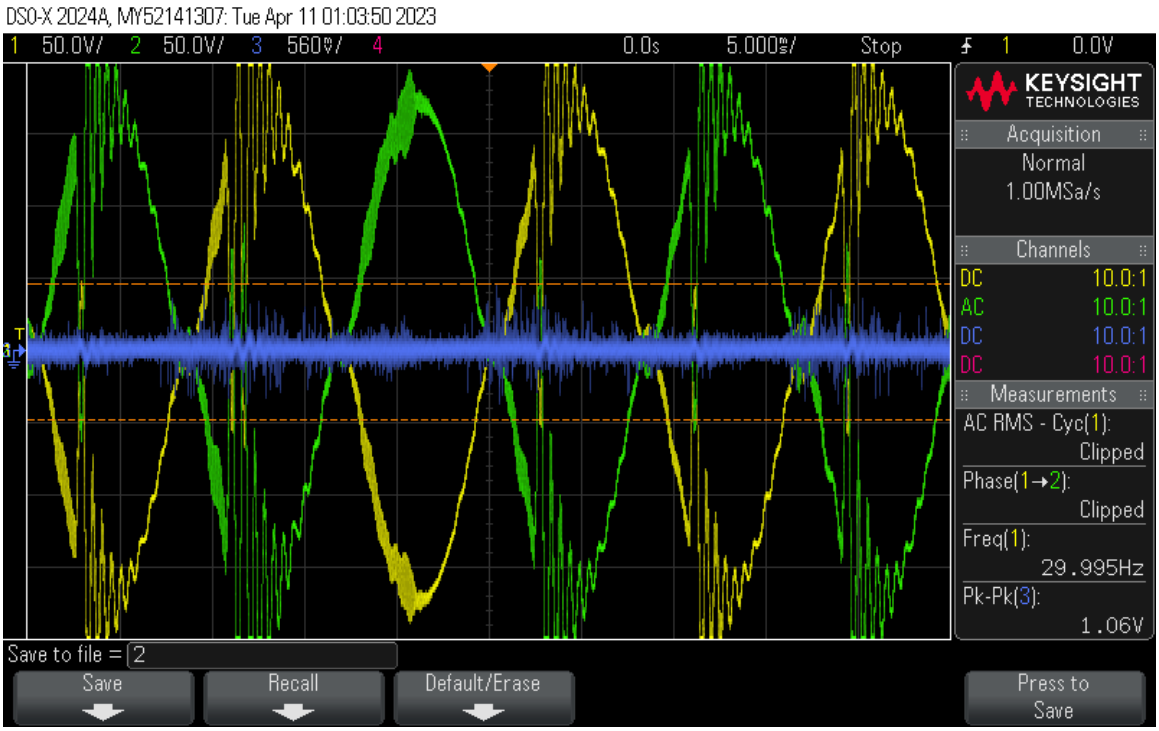

Figure A2.2: AC Coupling Attempt 1

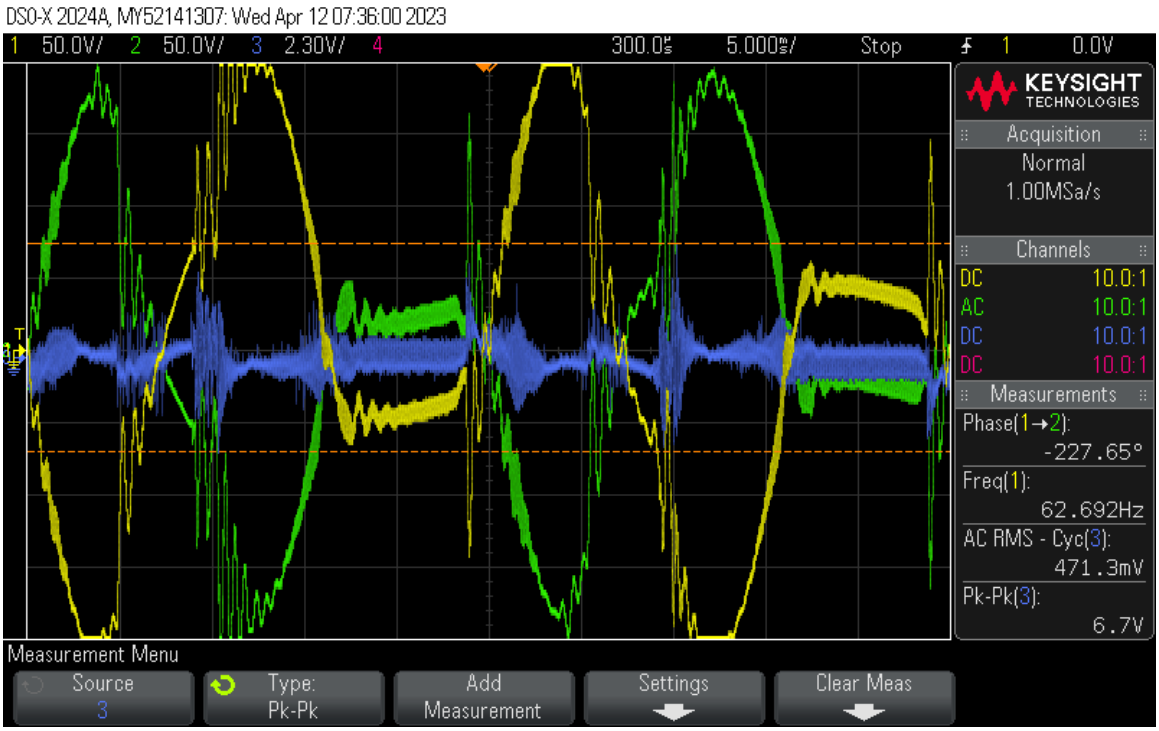

Figure 2.3: AC Coupling Attempt 2

Outbacks documentation in their AC coupling application note states that the inverter with AC coupling enabled shifts its frequency and other connected devices that are IEEE 1547 compliant should correct their frequency to synchronize or isolate themselves from the AC source altogether if they are unable. In our testing, we enabled AC coupling on each pallet individually, both pallets together, with and without freq/watt compliance enabled to little avail.

There were a few times in which we were able to get a semi-stable waveform after the inverters fought each other for a few seconds. Even though the voltages looked okay on the oscilloscope the resulting transient currents would cause breakers to trip between the pallets. We were unable to replicate this behavior reliably and thus have no traces to show.

Gridtied AC Input Mode/Selling Function

This configuration uses the Gridtied AC input mode, similar to what we were for our final design. The difference here is the use of the Sell function. The sell function works by essentially supplying power back through its AC input and is only available when using the Gridtied input mode. Using this mode we were able to use a follower to sell power back to the leader. This was done without any loads present and the sold power was used to charge the batteries of the Leader. This configuration was initially promising, as we could pretty much use the configuration we settled for in our final design, and the physical connections required would be quite simple.

The problems we encountered in the mode was the inability to effectively limit how much power was being sold. There are settings available to limit the sell current, but without loads to use the power, it would always just be sold to the leader and used to charge its batteries. In our testing, we disabled the charger on the leader through the Mate3 and inverter settings, but the power was still

being used to charge the batteries at the sell current limit of the follower pallet. We believe the charging may be a feature of the system to absorb excess power and direct it to the batteries/charger to deal with excess current, as we saw similar behavior when testing our final design. The issue here is the potential complexity of coordinating sell currents and statuses for followers. We also had no way to test the behavior of the system with multiple followers selling in unison. Given the time we had left in the semester, we decided this would probably not be a feasible approach for us.

Appendix III: Areas of Concern and Future Work

Unfortunately we don't feel like we had enough time or expertise to do sufficient research and testing for our final design. The makeup of our group consisted of software and computer engineering students, and thus we lacked some of the knowledge necessary for a more thorough job in terms of determining, testing and validating our final design choices.

We believe there is room for more work to be done in testing our final design and finishing the implementation. We were unable to make all of the necessary configurations to the Outback system through the AXS port for the Inverter. We currently can not change certain values through the AXS port. We are still in the process of troubleshooting the problem, but are unsure if it will be resolved. Due to this, we were unable to fully implement and test our final design as one integrated unit including controlling the relays, and driving all the necessary configurations.

We would also like to see more testing done trying to deliver power to various different loads. After a discussion with our client, it seemed like the power we were able to produce would be usable but we never got a chance to use it. More testing could also be done to see how battery voltages, and/or a common battery bank would affect the inverter outputs and behavior, it is possible that those things could address some of the concerns we had while doing our testing.

It should also be noted that we only had access to two pallets. In order to do more realistic testing, it would make sense to have access to a larger array of devices so we could study their behavior as a group in more detail.

These pallets are quite complex. We are almost certain it is possible to achieve what we're looking to do with the hardware provided, but we may have missed the exact combination of variables and/or input modes necessary to do so.

Specific Known Issues:

- Unable to read/write to SunSpec Model 64120 ("OutBack System Control Block")
- Phase Drifting

Appendix IV: Sensing Hardware Connections for Driving Configurations

This section expands upon the use of the various outback variables and configuration parameters available through the AXS port to determine the interconnections of a group of connected microgrids. This documentation was created to aid developers in the implementation of using those variables to determine the connection of pallets, and drive configurations accordingly. This implementation is based on sensing connections that are not necessarily present in our final design, but the main idea of sensing configurations could be useful for other potential applications, such as ensuring safe connection between pallets, or doing automated diagnostics tests.

Configuration Modes/Profiles

Each pallet, irrespective of its role, should be able to support/change the following configurations. The pallets should always default to safe mode in the event of any faults or errors present within the inverter.

Safe Mode (Default)

This mode disables inverting and accepting of AC source,and puts the inverters in a “safe” mode. The inverters should put themselves in this state in the occurrence of a major fault, but we need to make sure that we are also supporting this behavior. Note: Inverters moving from any one configuration to another should always be configured or checked against the safe state before moving on to other configurations. This configuration will serve as an anchor that will undo any configurations driven in any other modes.

OutBack System Control Block (DID = 64120)

Start	Name	Description	Desired Value
7	OB_Inverter_AC_Drop_Use	1=Use, 2=Drop	2 (Drop)
8	OB_Set_Inverter_Mode	1=Off, 2=Search, 3=On	1 (Off)
9	OB_Grid_Tie_Mode	1=Enable, 2=Disable	2 (Disable)

Radian Inverter Configuration Block (DID = 64116)

Start	Name	Description	Desired Value
21	GSconfig_AC_Input_Select_Priority	0=Grid, 1=Gen	1 (Gen)
23	GSconfig_Gen_AC_Input_Current_Limit	Gen AC current limit	5
25	GSconfig_Charger_Operating_Mode	0=Disable, 1=Enable	0 (Disabled)
26	GSconfig_AC_Coupled	1=Yes(not impl.), 0=No	0
32	GSconfig_Gen_Input_Mode	0=Gen, 1=Support,	**1** or 6 (Test 6)

		6=Grid Zero	
37	GSconfig_AC_Output_Voltage	AC output voltage	240

Grid Establishing

Configuration used for the microgrid that will be driving the other grids. This mode will begin inverting, and its output will be used to set the phase for the other connected pallets. Only one grid in the system should be using this mode.

OutBack System Control Block (DID = 64120)

Start	Name	Description	Desired Value
7	OB_Inverter_AC_Drop_Use	1=Use, 2=Drop	2 (Drop)
8	OB_Set_Inverter_Mode	1=Off, 2=Search, 3=On	3 (On)

Following

This configuration is used by all microgrids not establishing the grid. The mode will use its AC inputs and run in a support mode, in parallel with all other following pallets. This array of pallets get their phase from the grid established pallet. They work together to support the load using the grid establisher as a source of truth for phase and voltage.

OutBack System Control Block (DID = 64120)

Start	Name	Description	Desired Value
7	OB_Inverter_AC_Drop_Use	1=Use, 2=Drop	1 (Use)

Radian Inverter Configuration Block (DID = 64116)

Start	Name	Description	Desired Value
26	GSconfig_AC_Coupled	1=Yes(not impl.), 0=No	1
32	GSconfig_Gen_Input_Mode	0=Gen, 1=Support, 6=Grid Zero, …	1 or 6 (Test 6)

Sensing

Relevant values for sensing will depend on whether or not a respective pallet is a leader or a follower. In this mode, the leader will need to change its configurations to begin inverting and should also read the voltage across its own ac input. The followers do not need any configuration changes, but will need to be reading in live data from their respective inverter.

Leader Configuration Changes simply tells the leader it should begin inverting.

OutBack System Control Block (DID = 64120)

Start	Name	Description	Desired Value
8	OB_Set_Inverter_Mode	1=Off, 2=Search, 3=On	3 (On)

Before inverting the leader should also ensure zero voltage across the blocks listed below. The following table will be used by the followers to determine if it is in a safe configuration, this data will also be re

Split Phase Radian Inverter Real Time Block (DID = 64115) (read-only)

Start	Name	Req. Follower Value	Req. Leader Value
13	GS_Split_L1_Gen_Input_AC _Voltage	0 (Volts)	0 (Volts)
14	GS_Split_L1_Output_AC _Voltage	0 (Volts)	120 (Volts)
20	GS_Split_L2_Gen_Input_AC _Voltage	120 (Volts)	0 (Volts)
21	GS_Split_L2_Output_AC _Voltage	0 (Volts)	120 (Volts)
22	GS_Split_Inverter_Operating _mode	0 (Off)	1 (On)

Connection Determination

In this part of the process, the elected leader, in conjunction with the followers, will work together to determine the current state of physical connection between the pallets. Based on these physical connections the leadership position will be transferred to the pallet that is physically connected as the leader. This process allows a user to simply make the appropriate connections across inverters without requiring manual configuration changes to be made based on those connections. If the correct ports are connected across the inverters, as specified per the operating instructions, the microgrids will autonomously configure themselves, and be ready to distribute power.

Sensing

The sensing itself will initially be coordinated by the elected leader. The leader will begin inverting, and request the followers to check voltages across the appropriate ports. If each follower sees voltage across its respective AC Gen In port, the leader will know its physical connections to the followers are correct, and that it is the “grid-establishing” leader. A simplified diagram showing this connection is below.

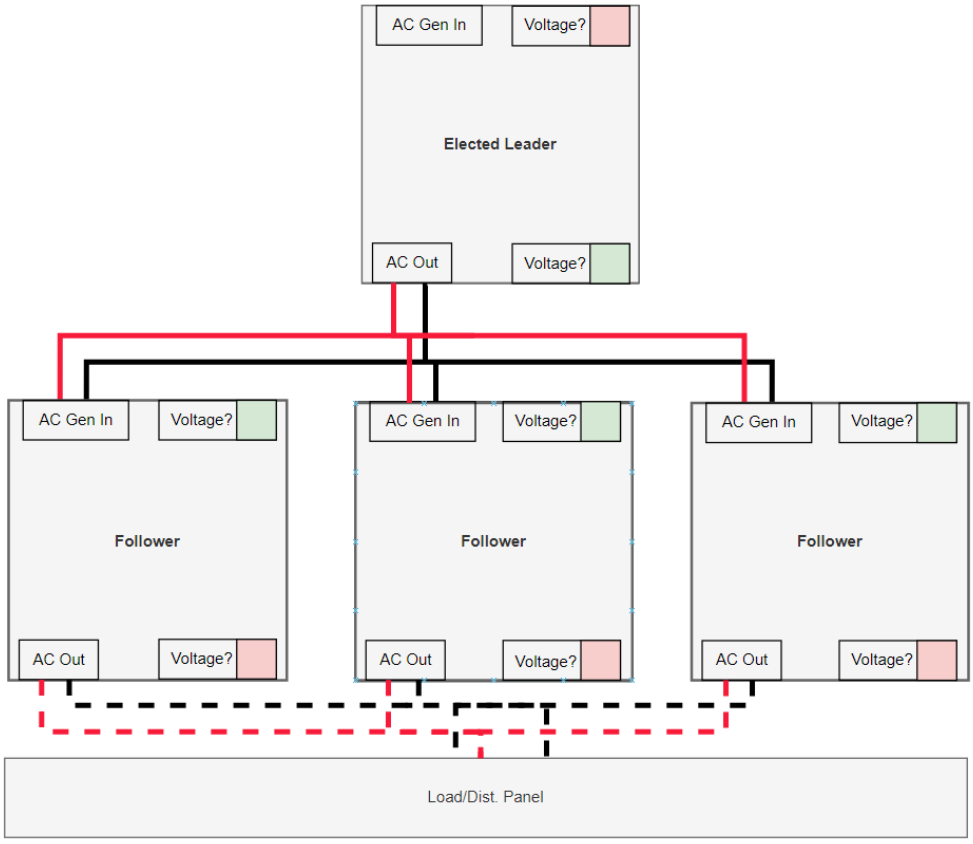

Figure A4.1: Sensing 1

Determining the Grid Establishing Pallet

In order for a pallet to be classified as “grid establishing” the pallet’s AC output will be connected to the AC Gen Input of all other connected pallets. It is integral that the AC input of this pallet is electrically isolated from its own output. This specific configuration allows the “grid-establishing” pallet to set the phase and voltages for the followers. Without a common phase across the followers, large currents will occur and can damage loads and other devices on the grid.

If the elected leader determines it is not the grid establishing pallet it will use the voltage data from the followers to determine which pallet it should transfer leadership to. This decision can be made by simply determining which follower pallet saw no voltage across its output. In the diagram below, the pallet labeled “Elected Leader” begins inverting. Voltage can then be seen across the AC outputs of the followers that are in the correct configuration while the top follower detects no voltage across any of its ports.

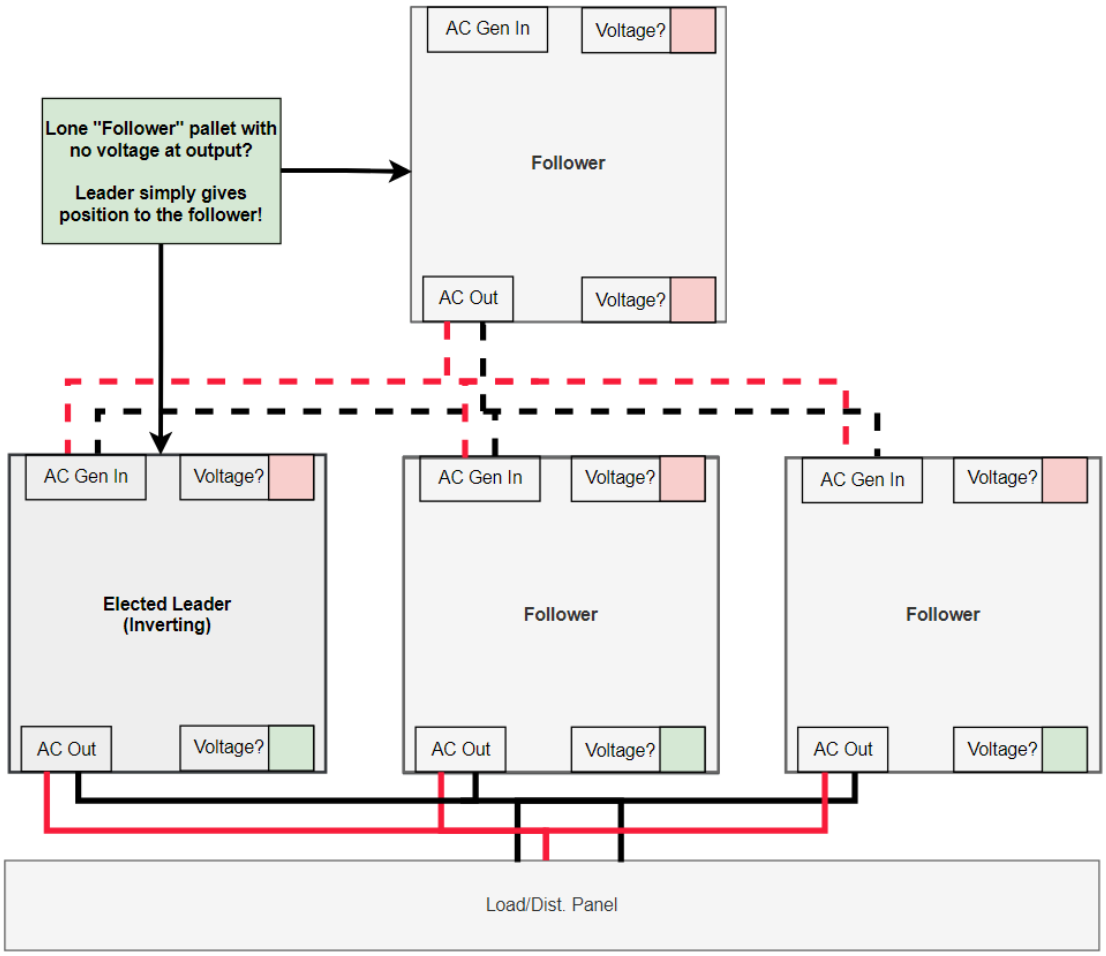

Figure A4.2: Sensing 2

While sensing, a pallet that sees no voltage across either port will consider itself a lone follower, and relay that back to the leader. If after all sensing is complete, and there is only a single lone follower, the elected leader will promote the lone follower to leader and the sensing process will start again. A diagram outlining the logic for the sensing algorithm is shown below.

Detecting Misconfiguration Errors

Another feature of this sensing process is the ability to detect potential misconfiguration errors. A partially connected system would be one error mode that our algorithm could detect. In order for the system to be considered valid, all followers must reach a consensus on whether or not the configuration is valid. The only time a configuration would be deemed valid is if the grid-establishing pallet was inverting and every follower in the system could sense the same voltage +- 5% across each leg at their AC Gen In terminals, and zero voltage across all other terminals. Under these conditions we can ensure that input and output terminals of each individual pallet are electrically isolated, thus meaning the only path for current to flow is through

the transfer relay of each follower. Consider the case presented in the diagram below, where a user may try to use two pallets as leaders.

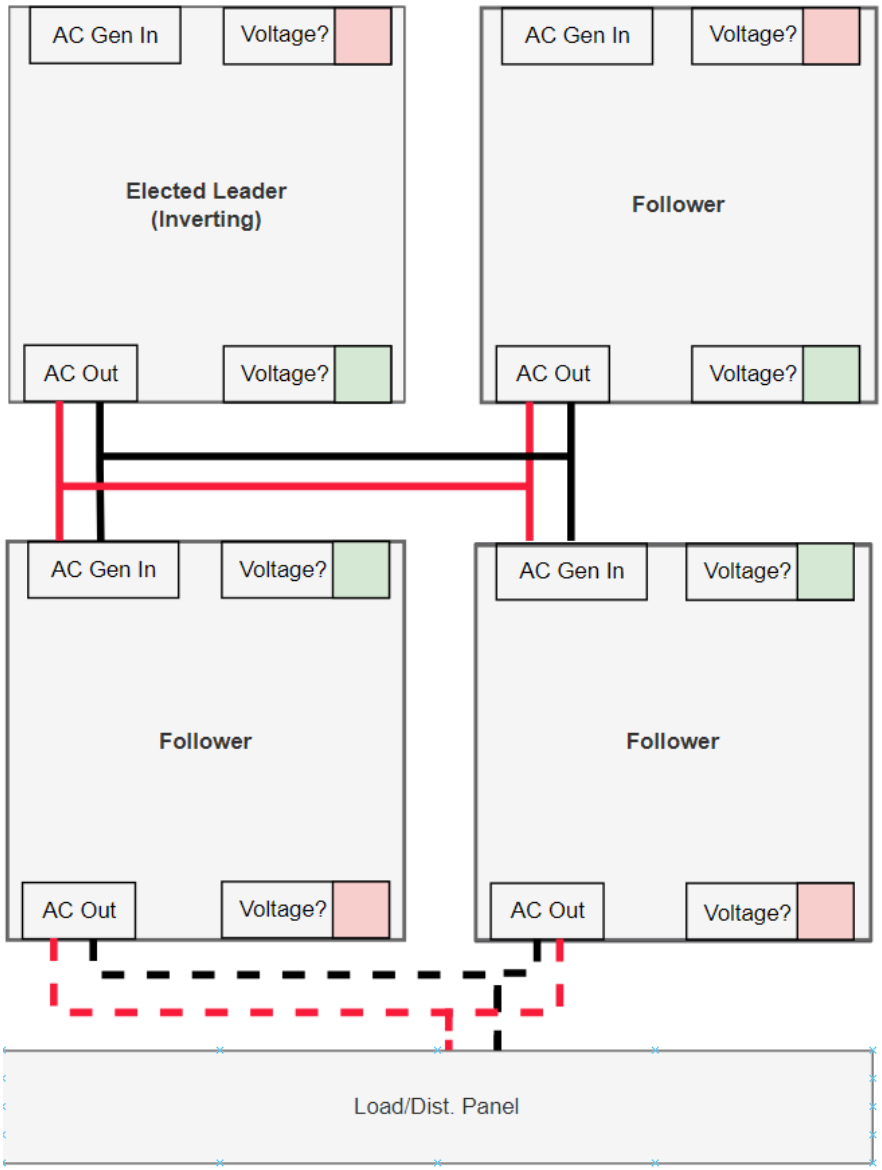

Figure A4.3: Misconfiguration Case

If one of the pallets on top was elected leader and began inverting as part of the sensing process, the follower connected in parallel would detect voltage across its output and signal that back to the leader. Other followers connected in the system would detect voltage across the Gen In terminals, and would also report that to the leader. Once the leader has all relevant information from the followers, it will deduct that some followers are improperly connected.

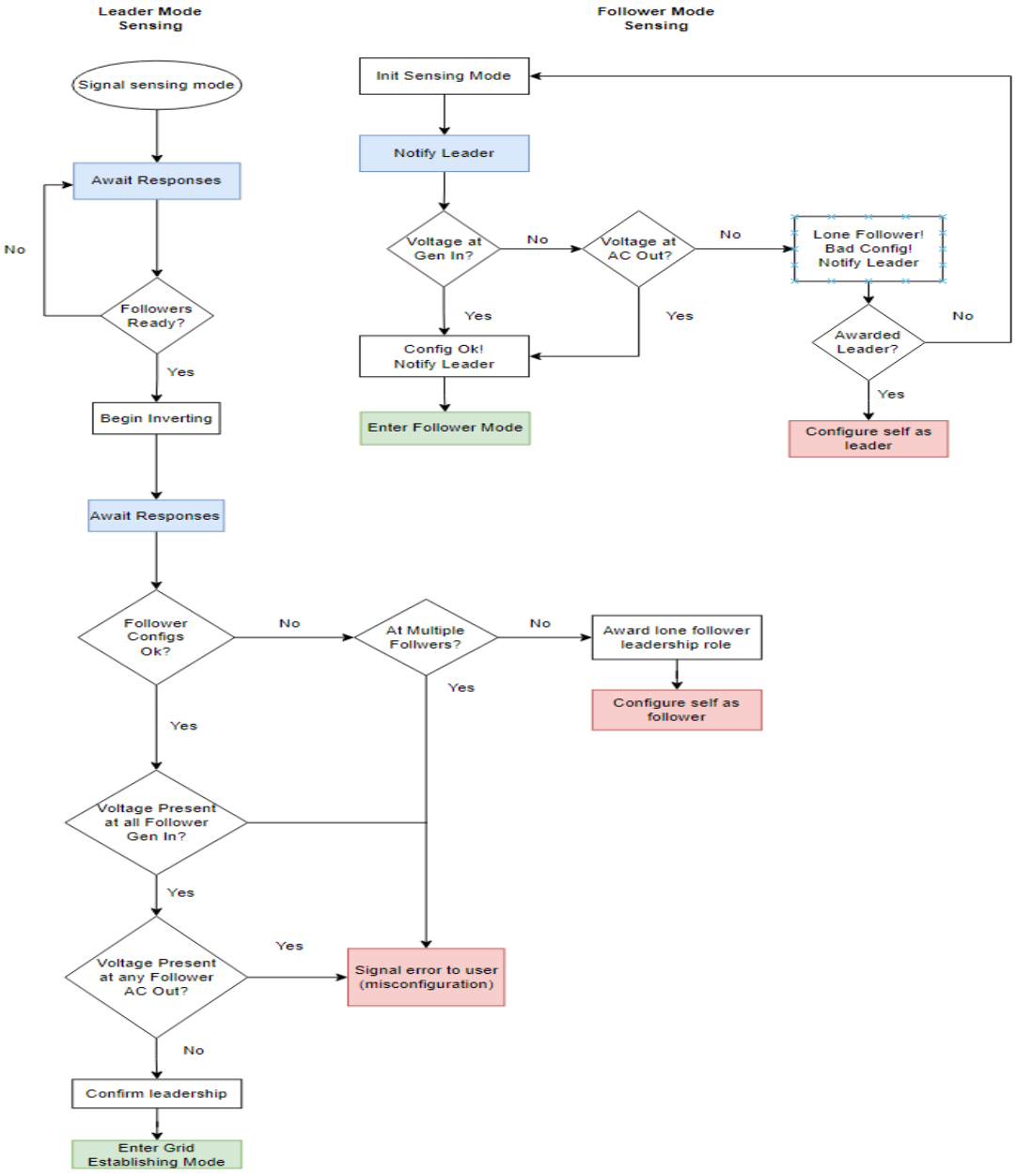

Figure A4.4: Sensing Flow Chart

Follower Pallets

After it has begun inverting, this voltage should be seen at the AC outputs of all the other inverters except for the true leader. The original elected leader will hand off leadership to the true leader. The true leader will repeat this process to verify.

If an inappropriate configuration is found (i.e. more than one inverter detects no voltage at its output) the leader will notify the other inverters and the user of the issue, and command a safe configuration to all connected devices.

By default, the first attempt of sensing will begin with just the leader inverting. If **all** the followers report back that they all can see the voltage at their Gen AC Input terminals, the leader can determine that it should be pallet to “establish” the grid. The leader then notifies all followers that it has found the grid establishing pallet, waits for acknowledgement that the followers are in the “Following” configuration mode, and begins inverting.

If the elected leader is deemed as not the “grid-establishing” pallet, it will then command each follower one by one, to attempt the same test until the “grid-establishing” pallet is determined.