Making Microgrids Plug and Play

Andrew Frank - Distributed Systems Architect
Christian Pinta - API Research/Programmer
Austin Thoreson - Systems and Hardware Engineer
Ben Eder - Software Developer
Saketh Jonnadula - Software Developer

Client: Nick David

Advisor: Matthew Wymore

sdmay23-36

Introduction

- Microgrid Pallets developed by Iowa State University Electric Power Research Center (EPRC)
- Individually deliver 8kW, higher power applications require synchronization
- Synchronizing pallets requires a specific setup process and technical knowledge

Our Goal:

Simplify the process of connecting pallets together and automate as much as possible.

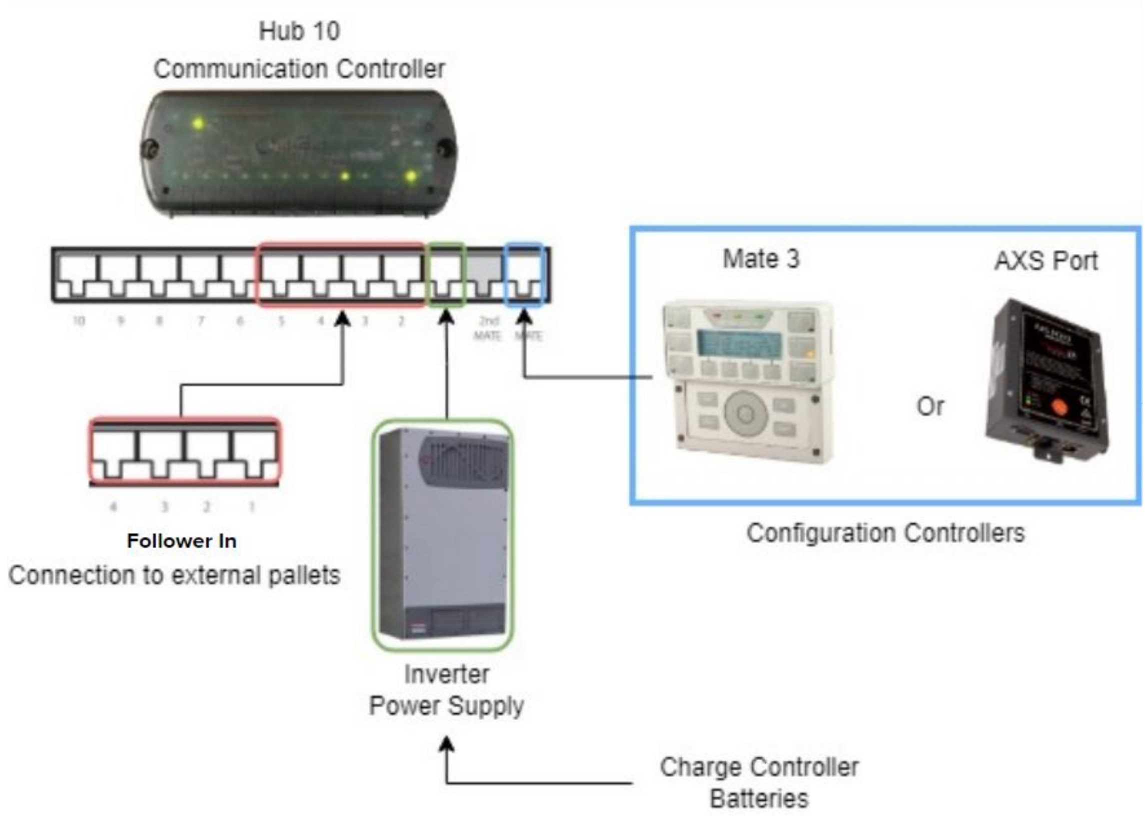
Existing Microgrid Overview

Mate 3
- Proprietary Interface
- Configures devices connected to Hub 10

AXS Port
- Interface for changing configurations
- Uses Sunspec commands via ModBus
- Translates Sunspec commands to proprietary messages

Hub 10
- Distributes proprietary communications
- Coordinates communication between pallets

Radian (Inverter)
- Delivers power from batteries
- Outputs single phase AC power at 120 or 240V



Relevant Terms

- DDS (Data distribution service)
  - CycloneDDS
- Outback Power
  - Manufacturer of the pallet power components
- Sunspec
  - pySunSpec2
  - Open Standard for the Distributed Energy Industry
  - Interacts with compliant devices (Outback Power)
- Tactical Microgrid Standard (TMS)
  - Standard created by Department of Defense for Microgrid communication
- AC Source Synchronization
  - Requirements of independent ac sources in order for them to be coupled. Must be similar in terms of Voltage, Frequency, and Phase Angle

Original Design Concerns

Problem

- Communication connections (Hub 10) are dependent upon chosen leader/follower roles

Solution

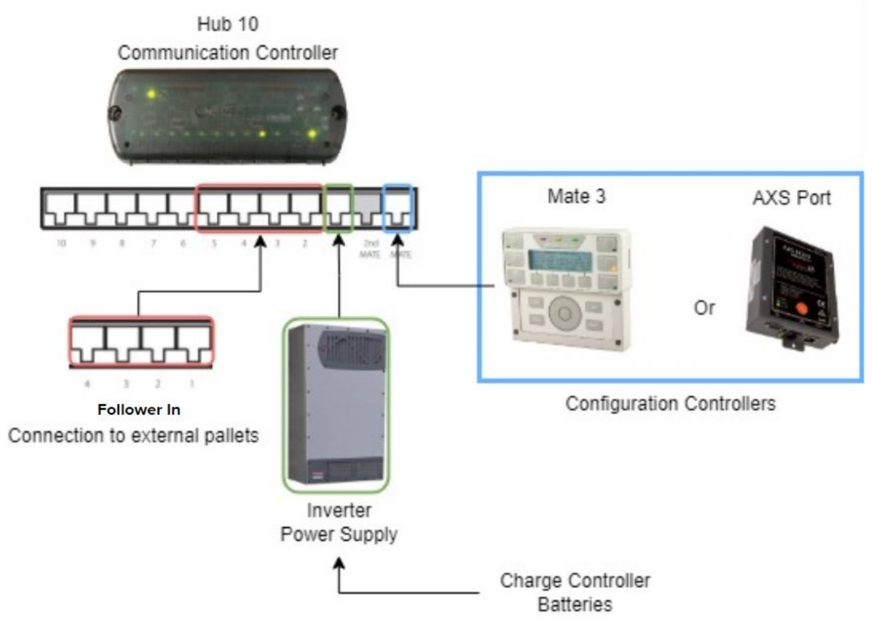
- Elect a Leader via software
- Bypass Hub10 traditional networking
    - Allows for more dynamic communication across microgrids

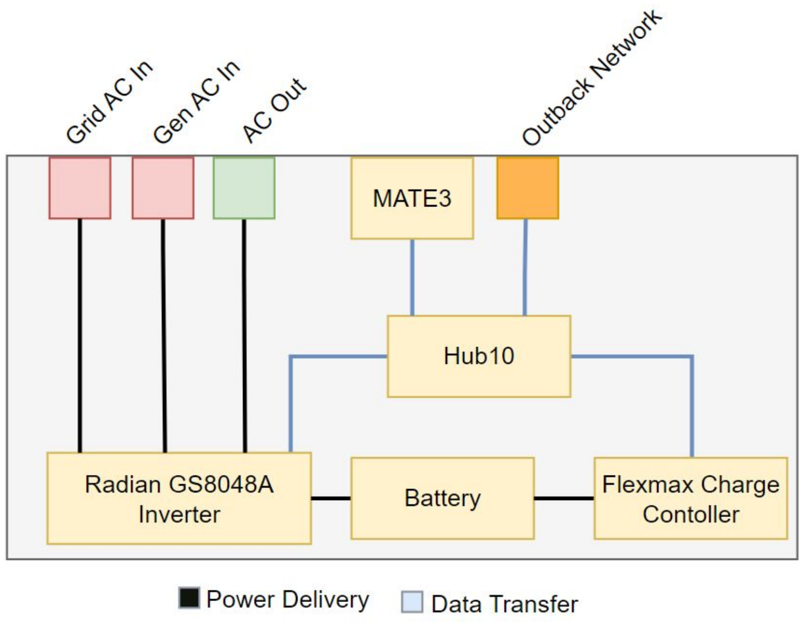
Problem

- Configuration interface (Mate3) requires prior knowledge of system

Solution

- Automatically configure system via a microcontroller and the AXS Port





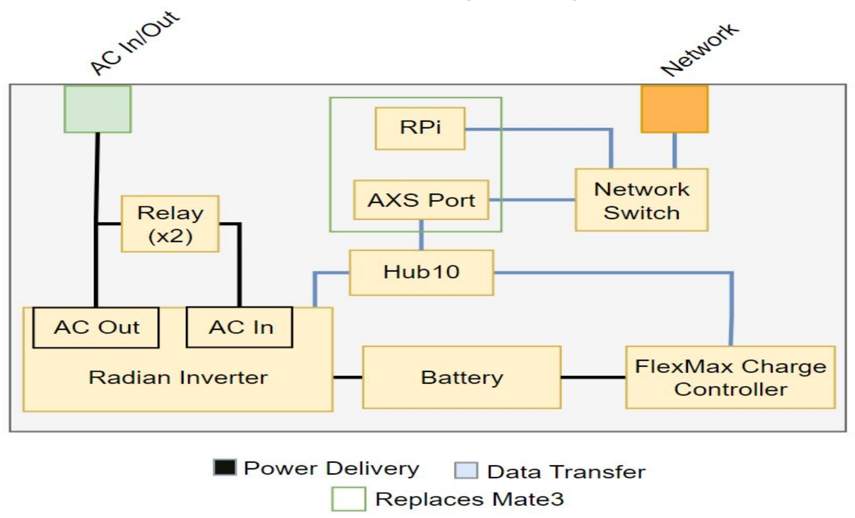
Traditional System Design

Proposed Multi-Pallet Design

New Steps

1. Couple inverters AC outputs and dist. to loads
2. Connect pallets via ethernet cable
3. Close breakers for batteries to turn on pallets
4. Pallets can now distribute power to loads

Proposed Design Diagram



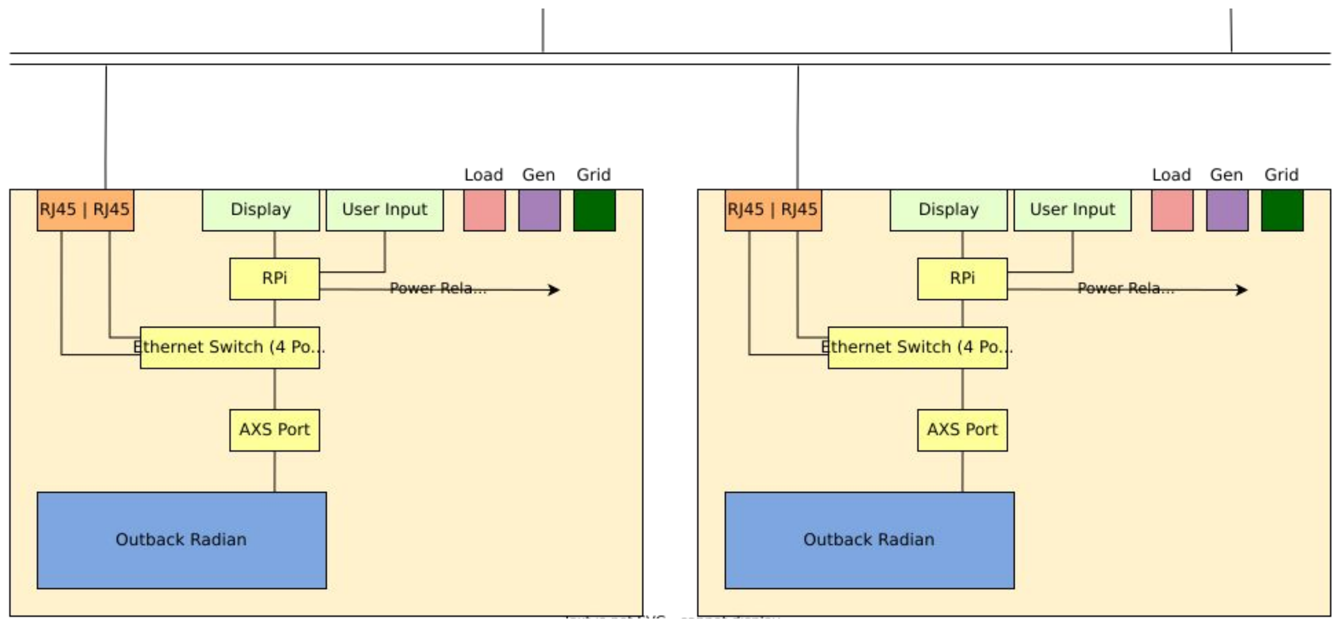
- No RJ45 switching or leader dependent routing
  - Use network switches instead
- Eliminates Mate3 and need for manual user configs
  - Uses Raspberry Pi and Axs Port

Traditional Steps

1. User selects a given pallet as a leader
2. Connect RJ45 from “follower out” port on follower(s) to “follower in” port on leader
3. On the leader pallet
   a. Switch RJ45 to Master
   b. Configure as a leader via the Mate3
      i. Settings/Inverter/Stacking and set Port 1 as leader, and other connected ports as followers
      ii. Settings/Inverter/Power Share and configure appropriate power share settings
4. On follower pallets
   a. Switch RJ45 to Follower
5. Turn on each pallet and couple their outputs
   a. Flip inverter switches to on
   b. Turn on load breakers for each pallet

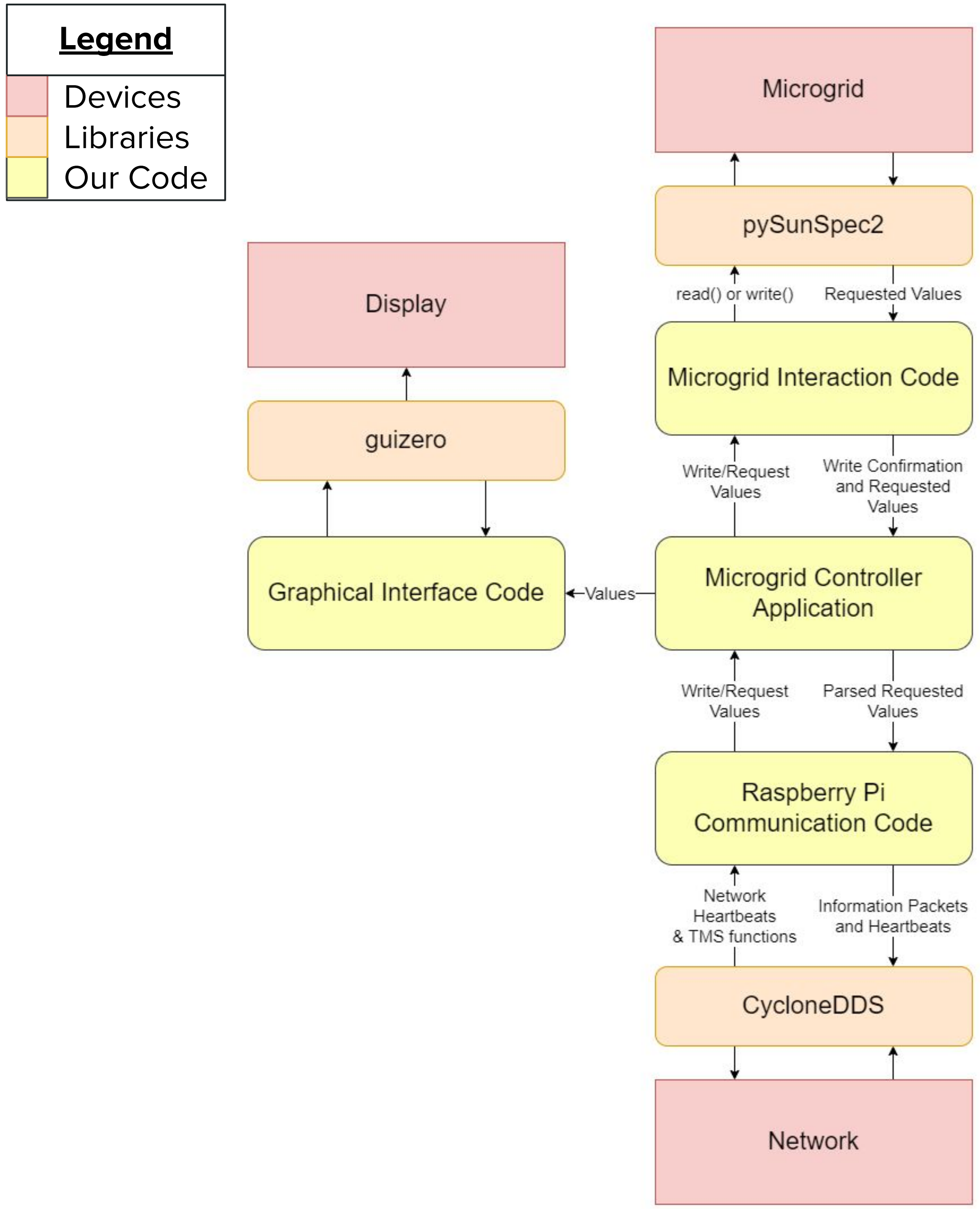
Network Design

- A Raspberry Pi is used to control/configure a pallet via the AXS port using SunSpec commands.
  - The Pi and AXS port replace the MATE 3
  - Eliminates static connections between microgrids by bypassing the HUB 10
- Pi communication occurs over a standard Ethernet connection via DDS (Data Distribution Service)
  - New grids can easily join an existing grid through an ethernet switch

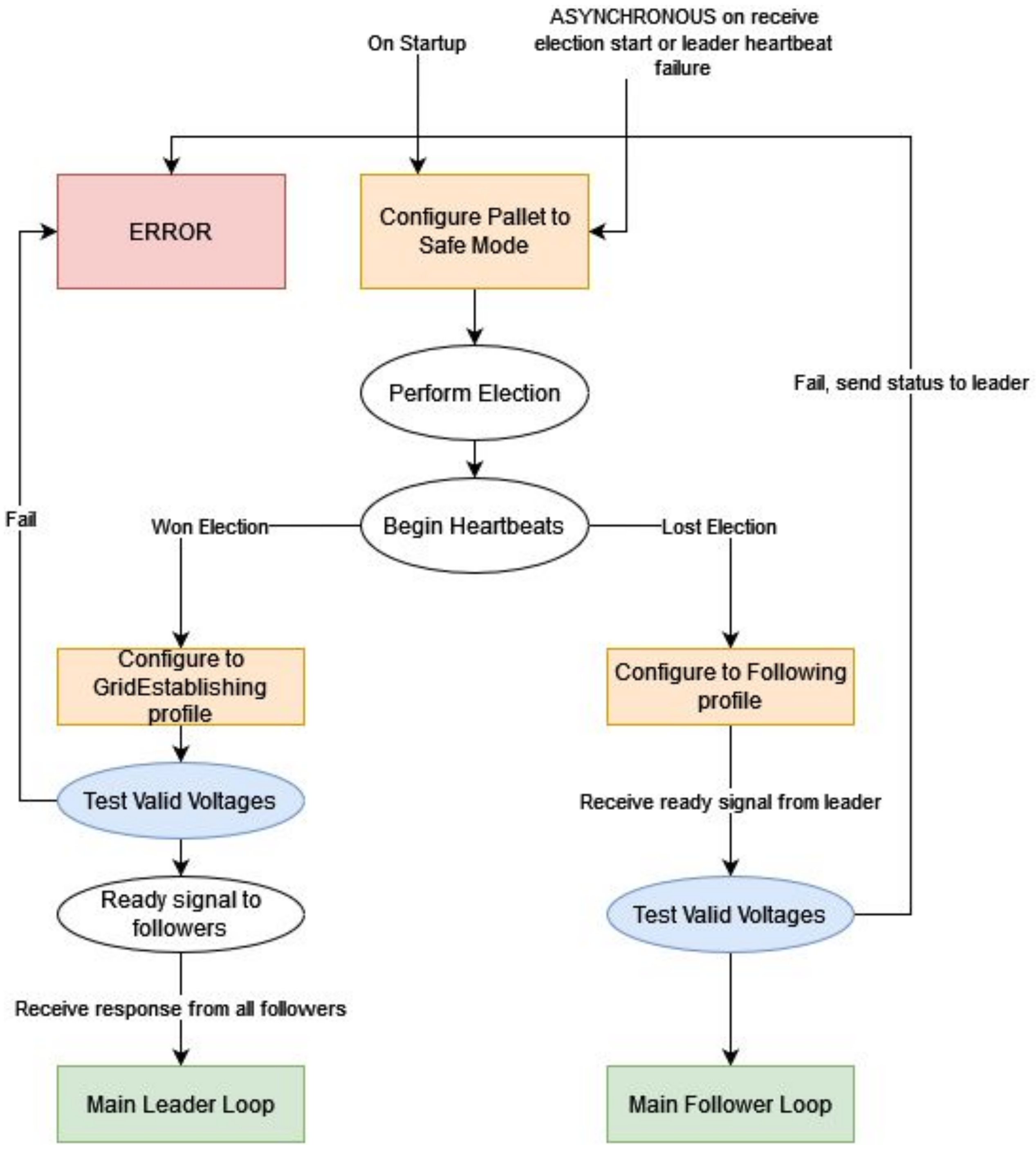


Software Module Diagram

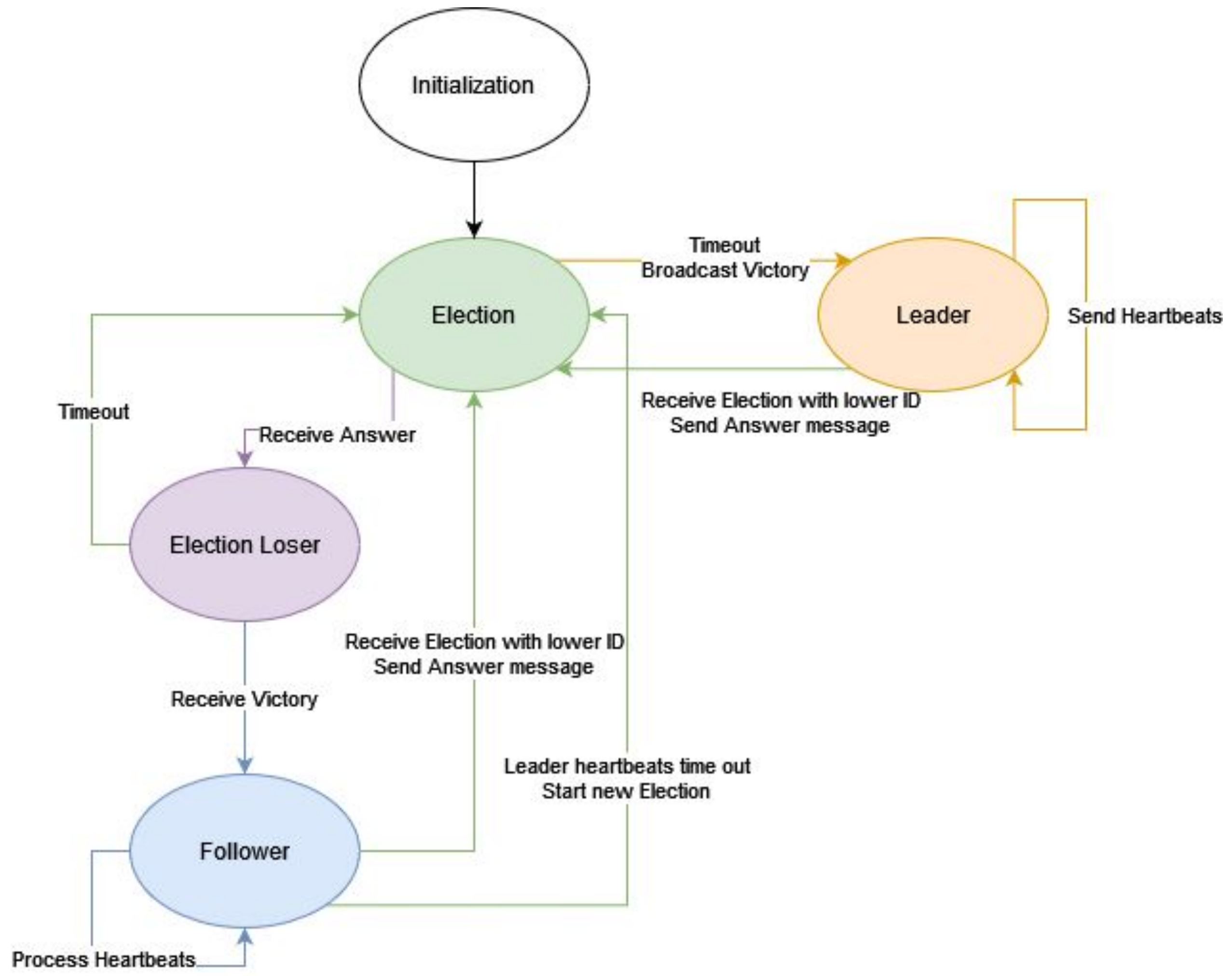
- Microgrid Interaction Code
- Microgrid Controller Application
- Raspberry Pi Communication Code
- Graphical Interface Code

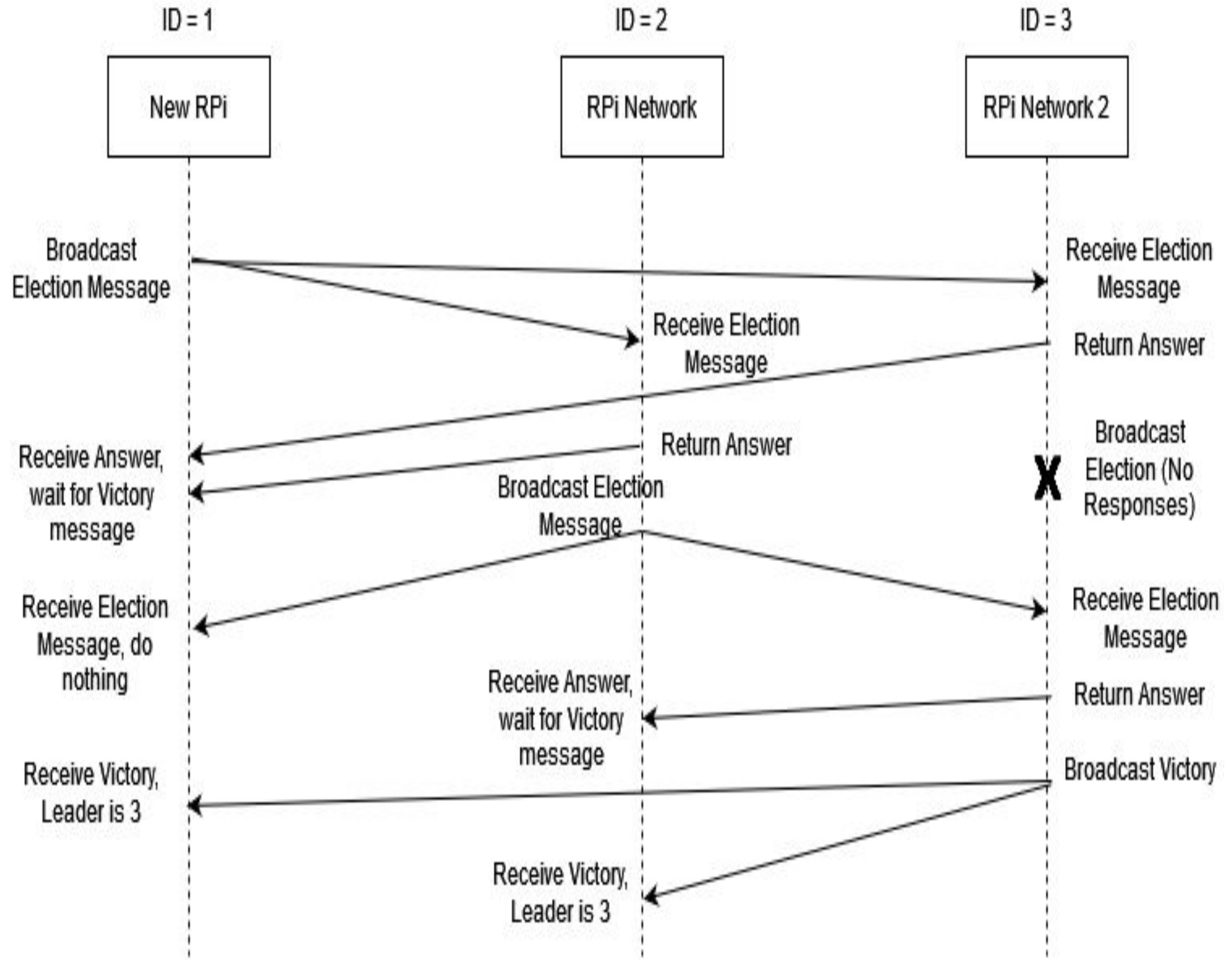


Controller Application Flow


On Startup
ASYNCHRONOUS on receive election start or leader heartbeat failure
ERROR
Configure Pallet to Safe Mode
Perform Election
Fail, send status to leader
Fail
Won Election
Begin Heartbeats
Lost Election
Configure to GridEstablishing profile
Configure to Following profile
Test Valid Voltages
Receive ready signal from leader
Ready signal to followers
Test Valid Voltages
Receive response from all followers
Main Leader Loop
Main Follower Loop

Leader Election




ID = 1
ID = 2
ID = 3
New RPi
RPi Network
RPi Network 2
Broadcast Election Message
Receive Election Message
Receive Election Message
Return Answer
Return Answer
Receive Answer, wait for Victory message
Broadcast Election (No Responses)
Broadcast Election Message
Receive Election Message, do nothing
Receive Election Message
Receive Answer, wait for Victory message
Return Answer
Broadcast Victory
Receive Victory, Leader is 3
Receive Victory, Leader is 3

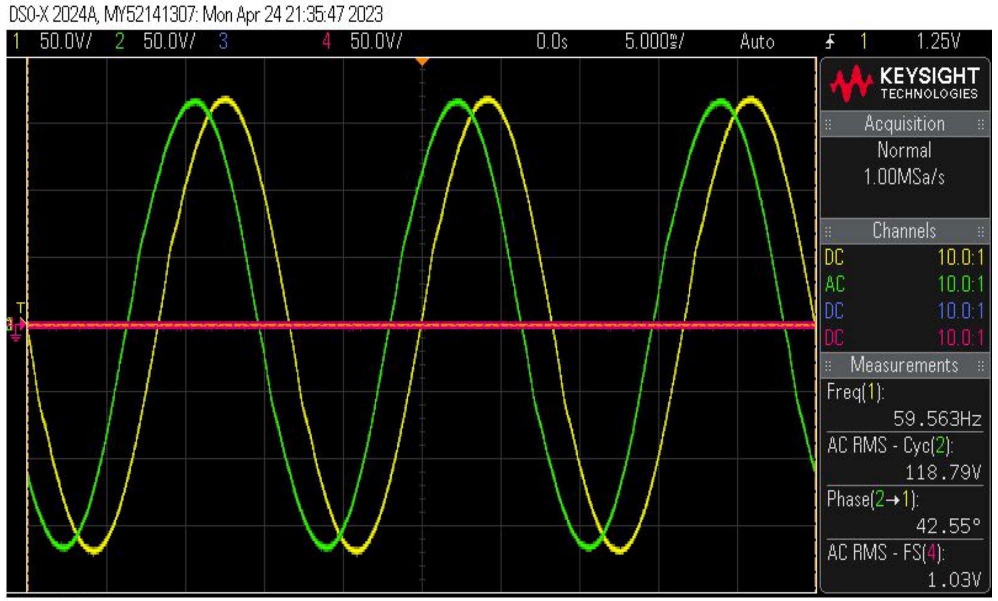
Hardware Configurations

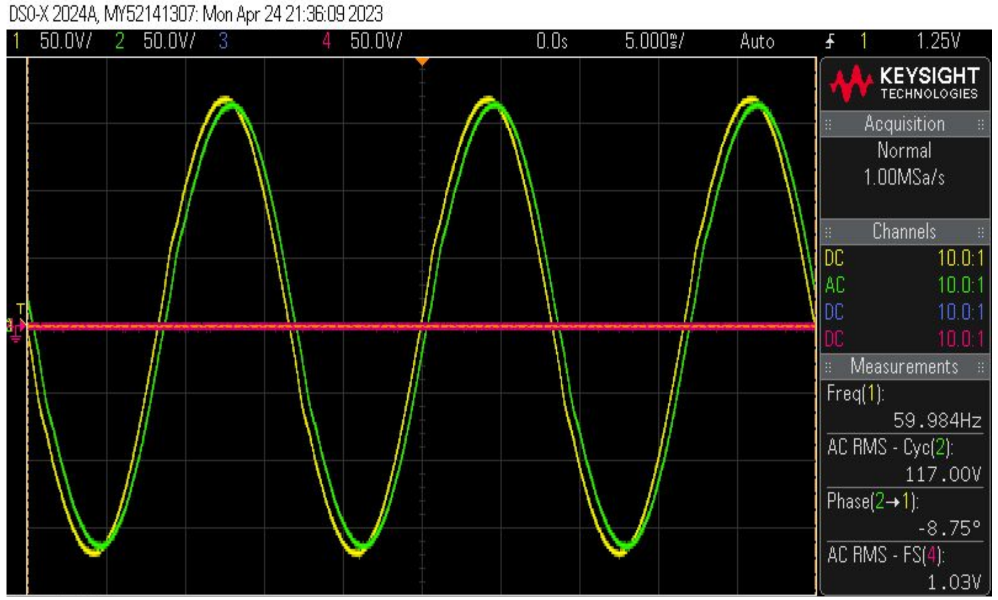
Challenge

- Hub 10 and Mate 3 were used to to coordinate AC source synchronization via “Inverter Stacking”
- The AXS port did not allow us to change these configurations
  - Deemed unsafe by manufacturer

Solution

- Find modes/variables accessible via the AXS port to achieve synchronization
- Landed on Grid Tied input mode
  - Inverter synchronizes with AC input, and allows it to pass through to the loads
  - Enable inverting and remove the AC input via added relays
  - Follower begins inverting with similar characteristics to source





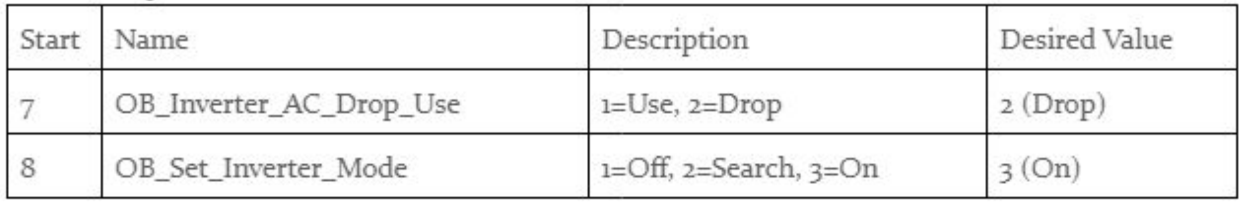
AC Sources before (top) and after (bottom) synchronization

AC Source Synchronization

- Uses relays that can be triggered by Raspberry Pi
- Provides path for Leaders power to AC input of follower
- Process can be controlled by Raspberry Pi and AXS port

Leader Configurations

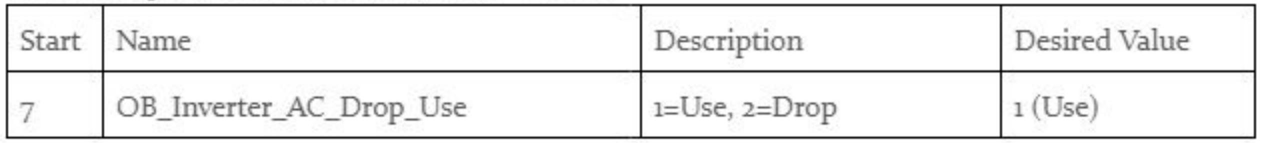
OutBack System Control Block (DID = 64120)



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)

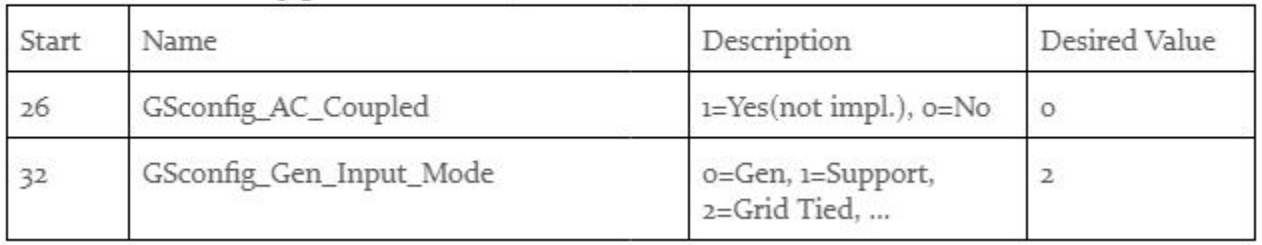
Follower Configurations

OutBack System Control Block (DID = 64120)



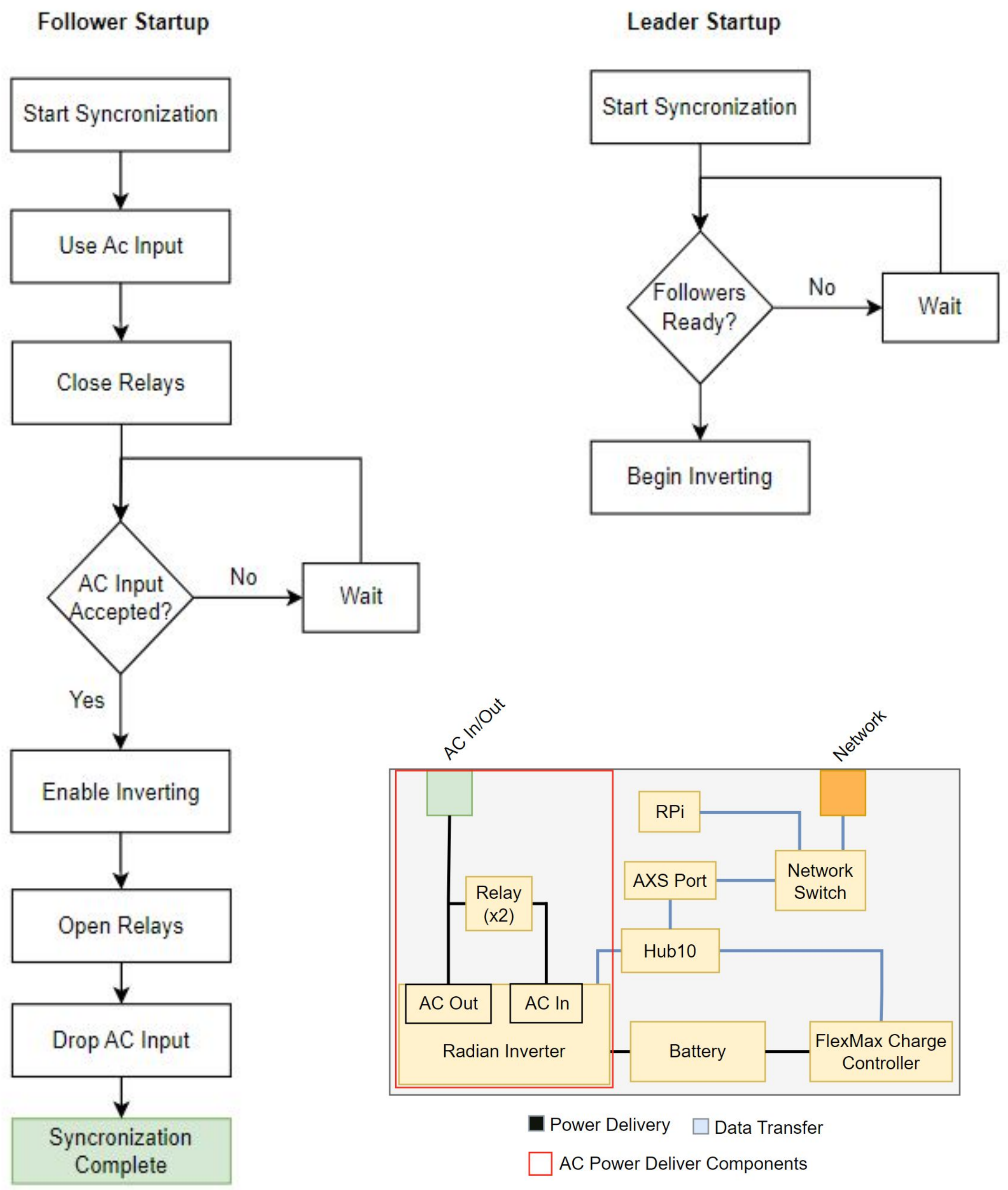
Start	Name	Description	Desired Value
7	OB_Inverter_AC_Drop_Use	1=Use, 2=Drop	1 (Use)

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

AC Synchronization Process



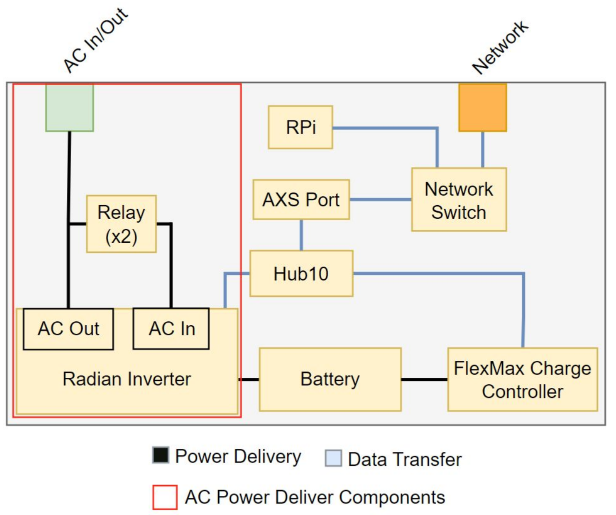
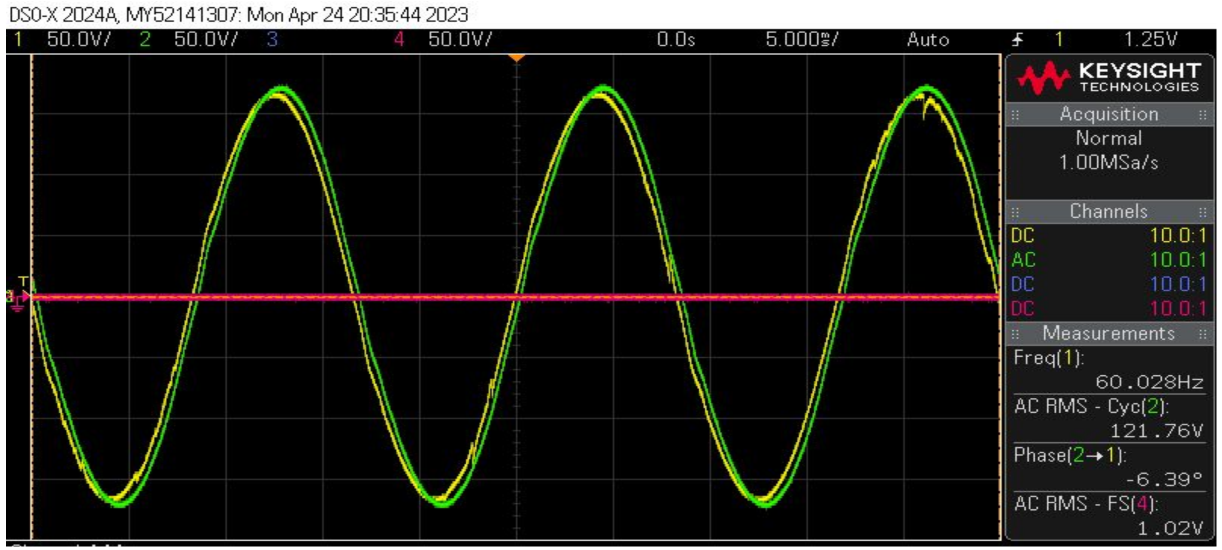


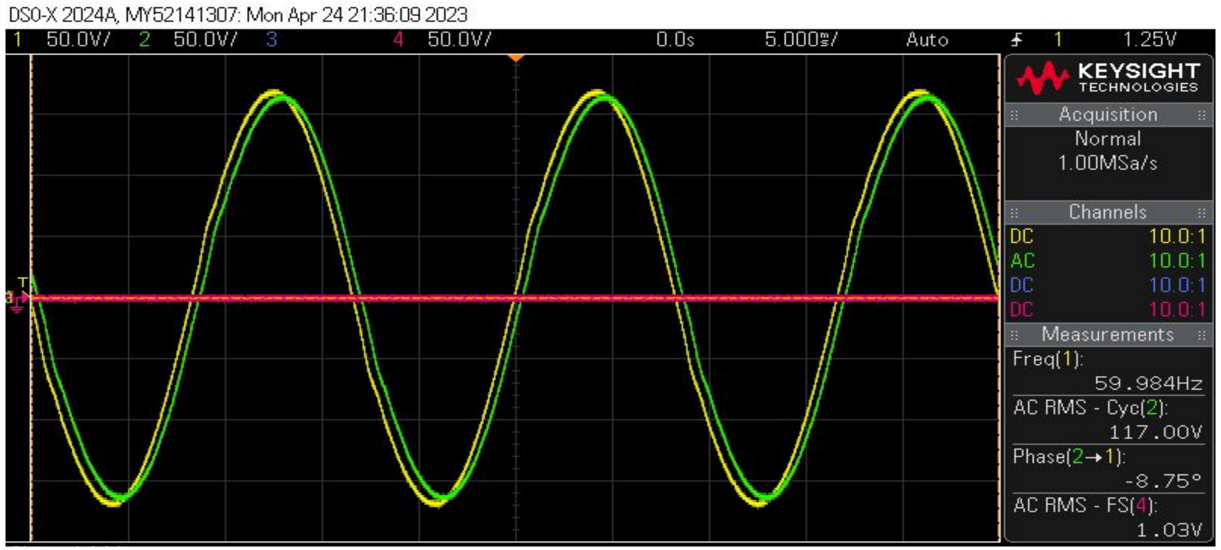
Diagram showing location of relays in the system

Hardware Testing Results



DS0-X 2024A, MY52141307: Mon Apr 24 20:35:44 2023

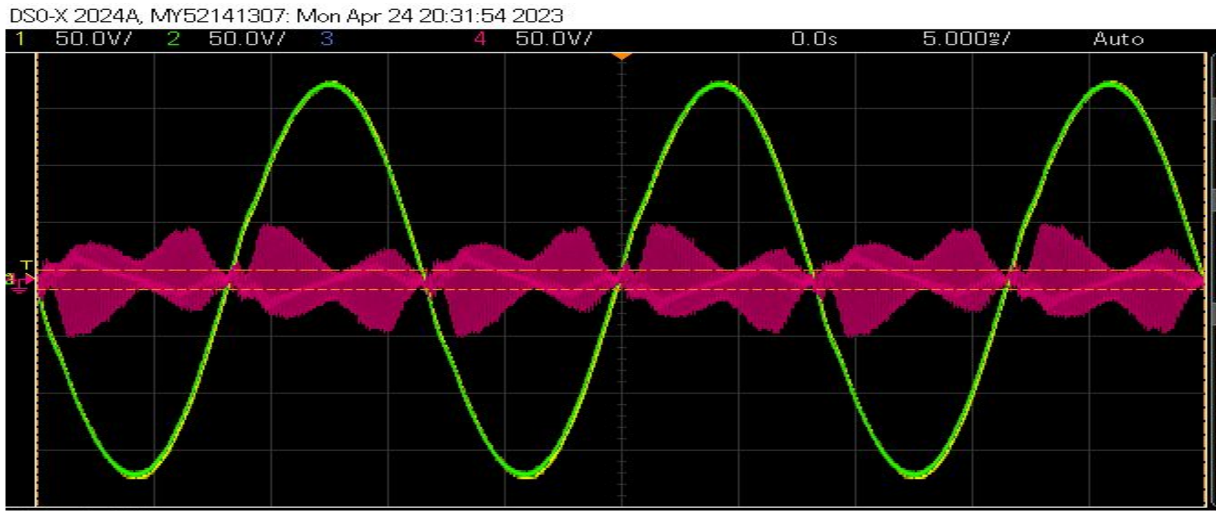
```
1  50.0V/  2  50.0V/  3    4  50.0V/    0.0s   5.000ms/   Auto   1   1.25V
Acquisition
Normal
1.00MSa/s
Channels
DC  10.0:1
AC  10.0:1
DC  10.0:1
DC  10.0:1
Measurements
Freq(1):
  60.028Hz
AC RMS - Cyc(2):
  121.76V
Phase(2→1):
  -6.39°
AC RMS - FS(4):
  1.02V
```



DS0-X 2024A, MY52141307: Mon Apr 24 21:36:09 2023

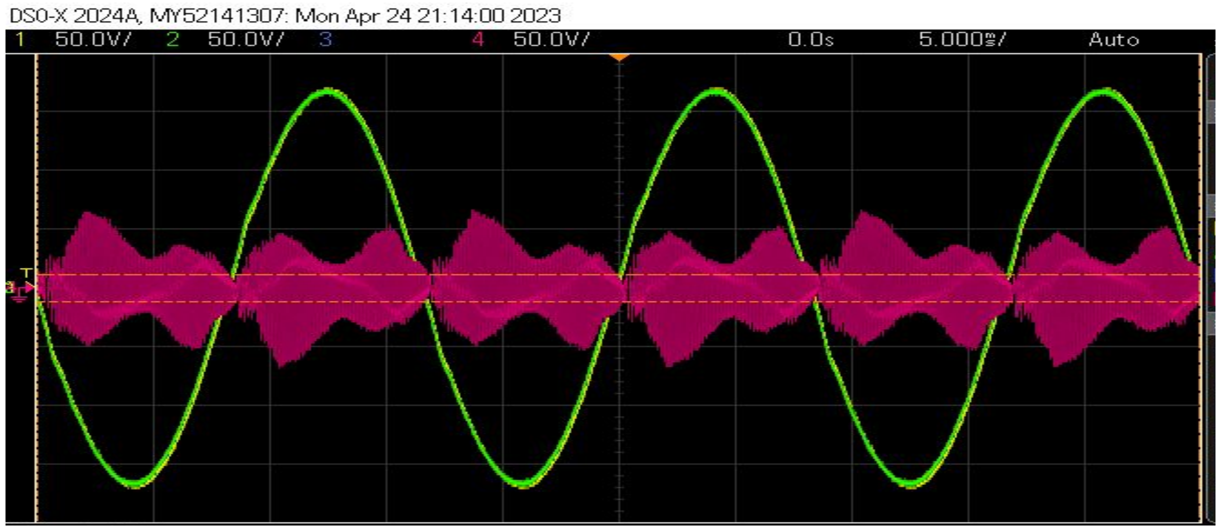
```
1  50.0V/  2  50.0V/  3    4  50.0V/    0.0s   5.000ms/   Auto   1   1.25V
Acquisition
Normal
1.00MSa/s
Channels
DC  10.0:1
AC  10.0:1
DC  10.0:1
DC  10.0:1
Measurements
Freq(1):
  59.984Hz
AC RMS - Cyc(2):
  117.00V
Phase(2→1):
  -8.75°
AC RMS - FS(4):
  1.03V
```

Phase shift characteristics of sources when coupled (left) vs uncoupled (right) for 30 mins



DS0-X 2024A, MY52141307: Mon Apr 24 20:31:54 2023

```
1  50.0V/  2  50.0V/  3    4  50.0V/    0.0s   5.000ms/   Auto
```



DS0-X 2024A, MY52141307: Mon Apr 24 21:14:00 2023

```
1  50.0V/  2  50.0V/  3    4  50.0V/    0.0s   5.000ms/   Auto
```

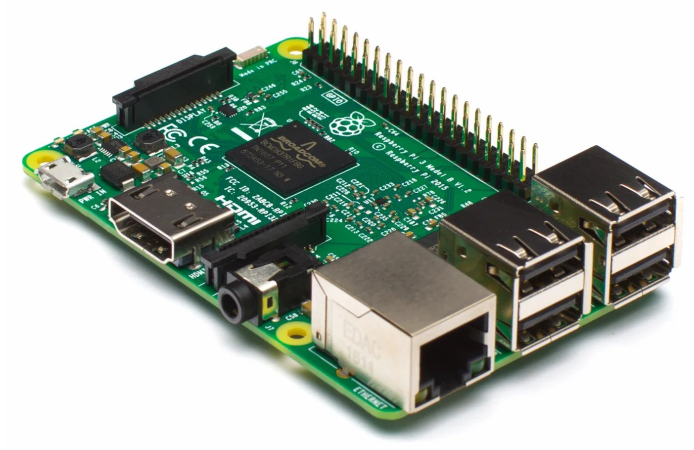
Current characteristics of sources coupled (left) vs uncoupled (right) for 30 mins

Work Accomplishments

- Microgrid Read and Write Code
  - Created missing model files for the pySunSpec2 API.
    - This allows access to parameters OutBack devices.
  - Created methods for easy interfacing with Microgrid devices
    - Allows for simpler, higher-end development.
    - Read/write packages of attributes at once
  - Network configuration that allows many devices to be connected with a single controller
    - Static IPs linked to physical raspberry pi controller

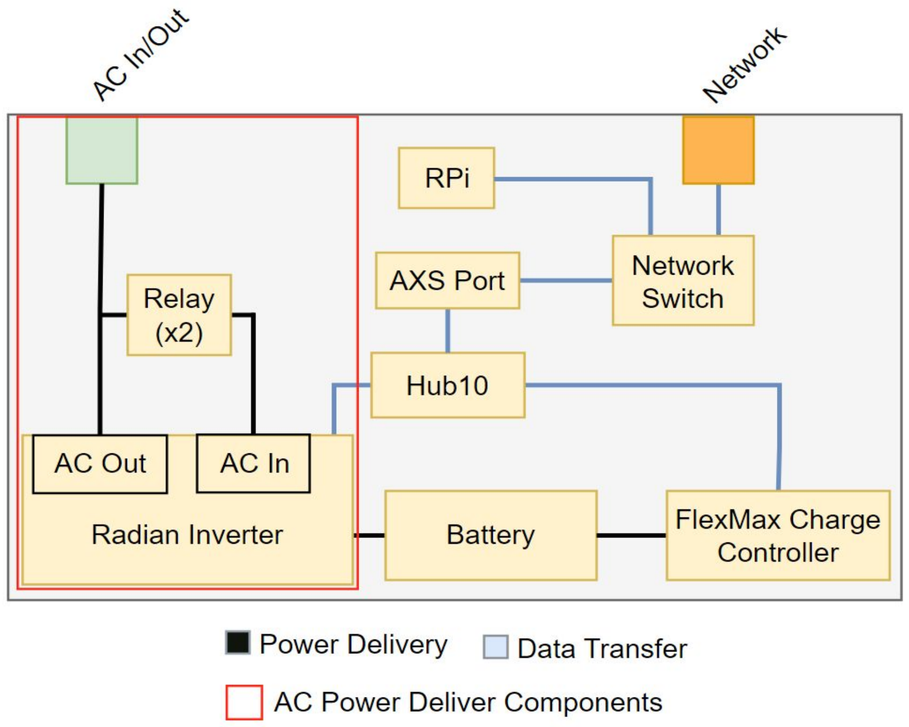
Work Accomplishments

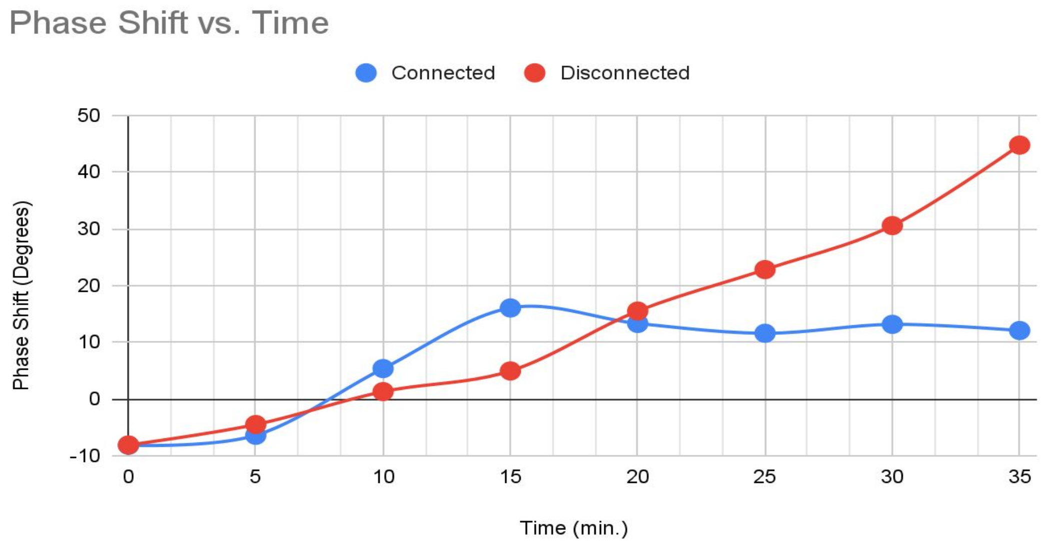
- Raspberry Pi Communication Code
  - Reaching consensus on a distributed network via election
  - Continuous monitoring and response to network events in the main controller loop
- Controller Application
  - State machine design for sensing pallet connections and configuring them accordingly
  - This can be adjusted to future configuration processes



Work Accomplishments

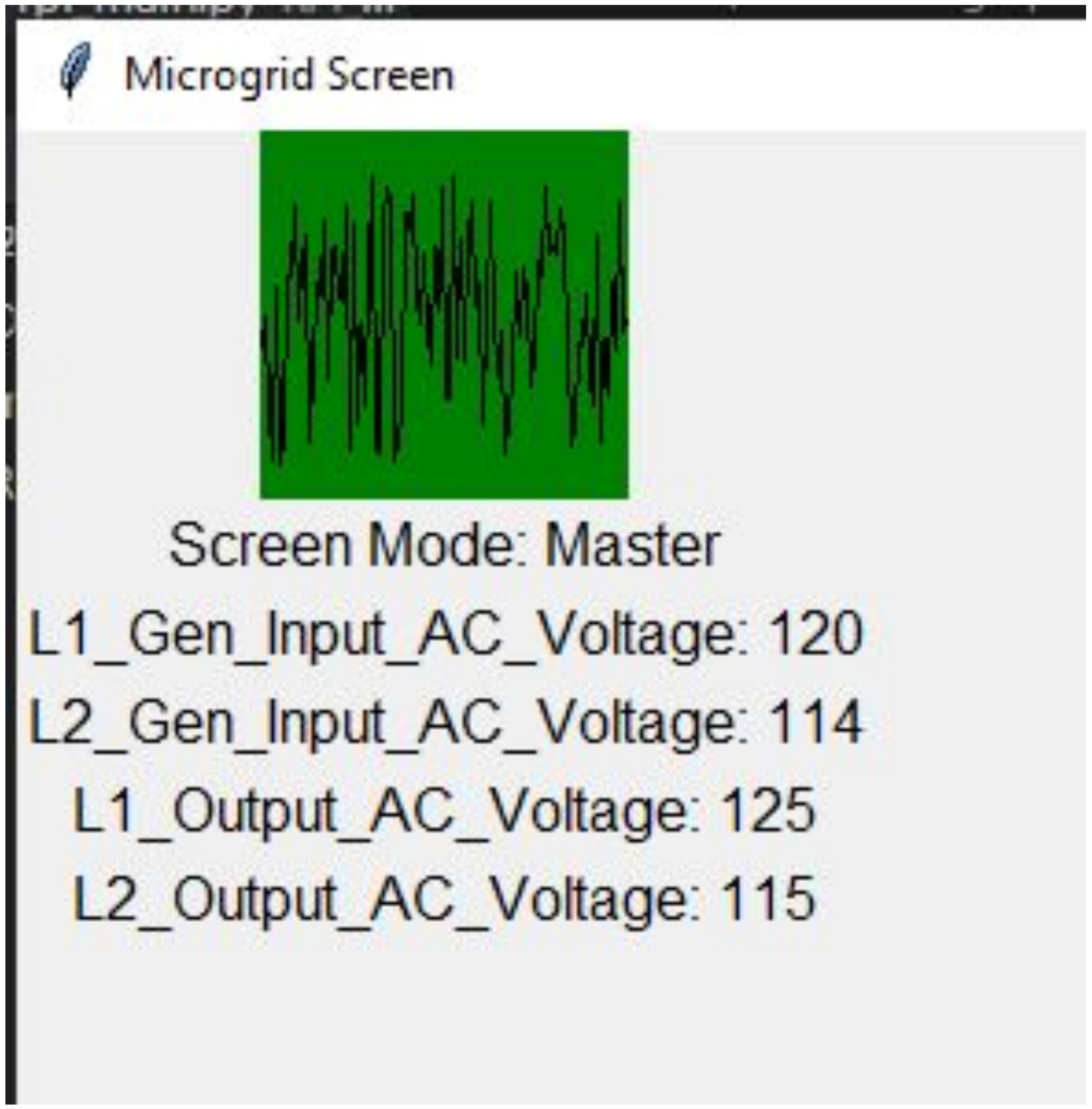
- Promising AC synchronization approach
  - Takes advantage of grid-tied input mode, and the transfer relay
  - Determined configurations necessary for software implementation
- Physical wiring of pallets
  - Simply couple the AC outputs to a load
  - On board relays take care of extra routing
- Testing of this approach
  - Phase shift characteristics of sources
    - Coupled vs uncoupled
    - Current characteristics
  - Powering small loads (<1.5kW)





Work Accomplishments

- GUI screen
  - What the current screen looks like
  - Master slave mode
  - What outputs can be shown.



Screen Mode: Master

L1_Gen_Input_AC_Voltage: 120

L2_Gen_Input_AC_Voltage: 114

L1_Output_AC_Voltage: 125

L2_Output_AC_Voltage: 115

Team Member	Role	Key Contributions
Austin Thoreson	Systems/Hardware	Hardware configurations, AC source synchronization, all things outback inverters and configuration, hardware testing, configuration testing, wiring, etc...
Christian Pinta	SunSpec API/Software	Researched SunSpec API, created necessary files and software not provided by Outback or SunSpec, developed Microgrid Interaction code, developed code to imitate AXS port
Andrew Frank	Distributed Software	Researched and implemented algorithms for leader election and heartbeating process. Implemented state machine software. Coordinated schedules and meeting times.
Ben Eder	Software engineer	Researched TMS functions, dataflow implementation and display GUI library. Implemented heartbeat process.
Saketh Jonnadula	Software Engineer	Helped research the TMS FUnctions, Worked on the GUI, Worked on the implementation for the heartbeat into the GUI

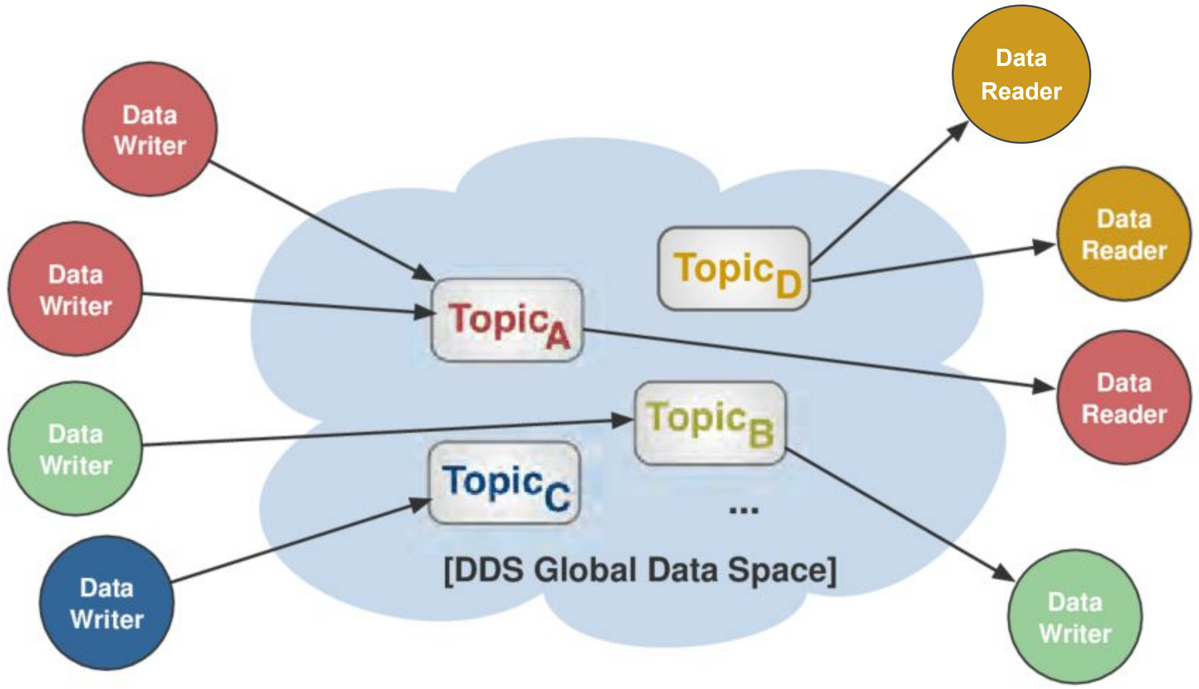
Challenges and Solutions

Challenge

- Collaboration of multiple devices over an unreliable network
- Account for devices dropping at any time

Solution

- Timer threads, heart beating
- Return to a safe configuration on unexpected changes
- In practice, more testing should be done to determine reliable timeout periods



Challenges and Solutions

Challenge

- pySunSpec2 did not include all the model files that the project needed

Solution

- Created a Python script to convert the documentation value tables into the format needed by pySunSpec2

```
{
  "desc": "Model identifier",
   "label": "Model ID",
   "mandatory": "M",
   "name": "GSconfig_DID",
   "size": 1,
   "static": "S",
   "type": "uint16",
   "value": 64116
 },
```

Challenges and Solutions

Challenge

- Our project needed to work on 2 or more pallets but we only had 2 pallets to test with

Solution

- Created a Python script to imitate having another pallet connected to allow us to test our code with as many potential pallets as we may need

Future Work

- Fix AXS port reading bugs
- Refinement of source synchronization
- Add small display to Microgrid
- Figure out how to power the network switch, Raspberry Pi, and display from within the Microgrid

Future Work: TMS Compliance

- Monitoring and management of a tactical power devices.
- Communication protocols
- Messaging patterns
- Data models
- Validation and Conformance

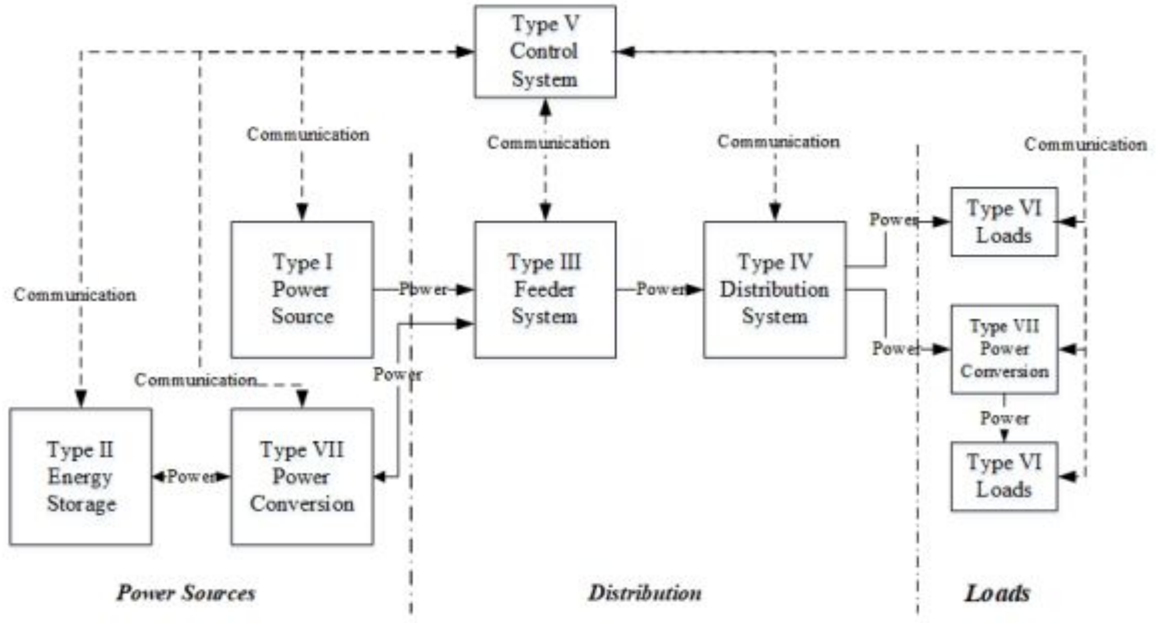


Figure 1 - Tactical Microgrid Schematic

[TMS Draft]
[TMS Public Release]

Conclusion

- Objectives Accomplished
  - Raspberry Pi Network software
  - Promising Configuration Procedure with Example Software
  - SunSpec Communication
  - Base GUI Design
- Missed Objectives
  - TMS functionality was a reach goal
    - Client agreed we should base our system on TMS at this point
  - Ideal Configuration Procedure
  - Power sources for switch, relays, raspberry pi