IMPLEMENTATION OF VOLTAGE REGULATION CONTROL SCHEME IN DISTRIBUTED COMPUTING PLATFORM

By

Diego Begara Bretón

Senior Thesis in Electrical Engineering

University of Illinois at Urbana-Champaign

Advisor: Alejandro Domínguez-García

May 2020

Abstract

Modern networks are built following a centralized architecture that relies on a constant communication between each node and the controller node. That node will then regulate the network. The approach we are taking throughout this project is implementing a distributed architecture. In this architecture we eliminate the controller node. We will substitute it by simpler controllers, Arduino boards, in each of the nodes. These Arduinos will be in charge of computation and communication with one another. The algorithms that regulate the network will be run in the Arduinos. They will work as a neuronal network in a collaborative structure.

This part of the project is based on the implementation of the voltage control algorithm into the Arduino code that will be implemented in the network. This code is tested with the Typhoon HIL simulator in one of the ECE laboratories. The goal is to code the algorithm so that it can work with any possible network we want to simulate. As complementary goals I will try to improve the robustness of the algorithms and improve functions in order to shorten the computational time.

For testing we have implemented the code into a 6-node example. Comparing the results obtained we can assure that the best approach for this case is implementing a distributed architecture which will give a more reliable faster recovery time.

Subject Keywords: Arduino, Centralized, Control, Distributed, Microgrid, Typhoon, Voltage.

Acknowledgments

I would like to thank Professor Dominguez and the University of Illinois at Urbana-Champaign (UIUC) very much for giving me the opportunity to work in this project that will continue and will make an impact on the way we control microgrids. After this project I will be able to say how much I learned and how I was able to help.

I would also like to thank Olaoluwapo Ajala for his help throughout this project, since without his help this would have not been possible. Thanks to his knowledge and patience I have been able to be here today presenting this paper.

Finally, I would like to thank the rest of team involved in the project: A. Mayoral, P. Samper, N. Siddartha and M. Zholbaryssov.

Once an Illini, always an Illini…

Contents

1. Introduction

1.1. Problem Description

The US energy Information Administration states that the power system consists of more than 7,300 power plants, nearly 160,000 miles of high-voltage power lines, and millions of low-voltage power lines and distribution transformers, which connect 145 million customers [1]. These numbers indicate the magnitude of the power delivery and consumption network we have nowadays. But, in order to understand the real size of the grid, we need to dive into the economics. To set the grounds, the average expenditure in electricity per month on American houses is around 117.65$ [2]. That makes a total profit of $15.75 billion per month just considering housing. With money comes improvement. The electrical market should focus their resources into providing the most reliable and affordable form of electricity. The main goal is to create a network as robust as possible so as to be able to withstand all the possible variations in consumption and generation our modern, electrical society can place. The second goal is to make the network more affordable, not only maximizing cash input, like electric bills. But also minimize losses, that come from blackouts and replacement of hardware.

Currently, we have an extremely complex and secure network, but there is plenty of room for improvement. On average, as shown by S. Massoud Amin [3], there is an average of 92 minutes of power outage per year in the Midwest, and 214 minutes in the Northeast. This data shows how reliable the grid is, since this quantity accounts for a mere 0.03% of the year. These number are relatable, but if you take into consideration the number introduced previously, the amount of money that is not being input is around thirty million dollars every month. This implies that there is plenty of room for. From the premise that more reliable means more advanced and more money, there needs to be a desire for improvement towards the most reliable and economic electric system possible.

According to the US DOE Smart Grid initiative and the Electricity Networks of the future, a grand transformation is taking place in the power distribution systems department. This trend is triggered by the rise in variable generation. The variable generation is referred to the implementation of renewable-based power plants like solar or wind farms. These plants are characterized by variable power function curves. An example of this type of curves is represented in Figure 1, shows the power output according to the time of the day in a region in the south of Russia. The second element is the inclusion of Variable Loads or VL. This concept is closely related to devices with the ability to store power, which is the case of plug-

in vehicles (PHEVs) or battery storage. As can be seen in Figure 2, these devices demand variable power from the network for an interval of time. This will put more stress in the network and will make it more susceptible to incur in failure conditions. These variable generation and consumption devices are referred to as distributed energy resources or DER [4]. The use of DERs has known advantages such as a constant power factor and disconnection when a fault takes place.

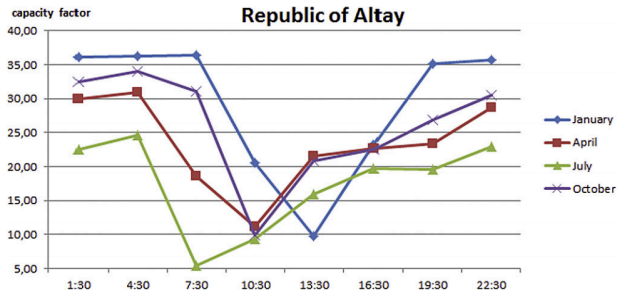


Figure 1:Daily Wind Power Output [5]

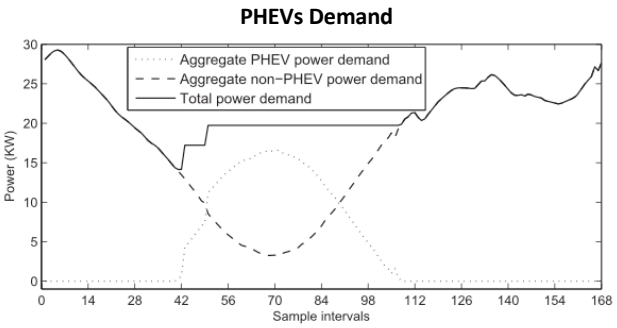


Figure 2: Power Demand from PHEVs [6]

Due to these new concepts in the electrical field, there is a need for change. The traditional network concept is based on a linear structure, as can be seen in Figure 3. It is based on the concept of generation vs load. The process begins by taking into account generation from power plants, such as coal power plants or wind farms. After that, the power is transported to the costumers. And finally, on the other side, there are loads in the which the power will be consumed. These cover from residential areas to hospitals or factories. It is a safe to say that this has proven to be a very useful and reliable take on the depicture of networks in the past. But, as our technology has advanced, the way we arrange and use also has to do so.

The electrical arrange we have nowadays is based on a non-linear structure. The concept of generation and consumption is no longer a direct partition. We can see this concept in the right half of Figure 3. The use of DERs has turned the network into a radial bidirectional scheme. For example, the implementation of DERs has led to the use of batteries that are considered as VLs that take power for future injection into the network. Another use in the which VLs are considered is in the charging of PHEVs. That, as seen in Figure 2, is characterized by a variable power curve. Second, there are houses with domestic CHP systems (Combined Heat & Power). These can be modeled as a load that will draw not only depending on the time of the day, but also on the power input from the sun. this raises the possibility for homes to draw fewer power from the grid as well as generating and selling power to the grid. This will introduce “new generator elements” into the grid´s scheme.

This prompts us to consider the grid as a neuronal network in the which each cell has similar functions and responsibilities. But the key conclusion is that they need to work with each other so as to execute their purpose and keep the mesh working in the optimal operation conditions. We have described the behavior of the system and the reason for the which there needs to be a change. Now a control architecture has to be defined. This control scheme will be in charge of the monitoring and mending of the system. There are two possible control architectures: centralized or distributed.

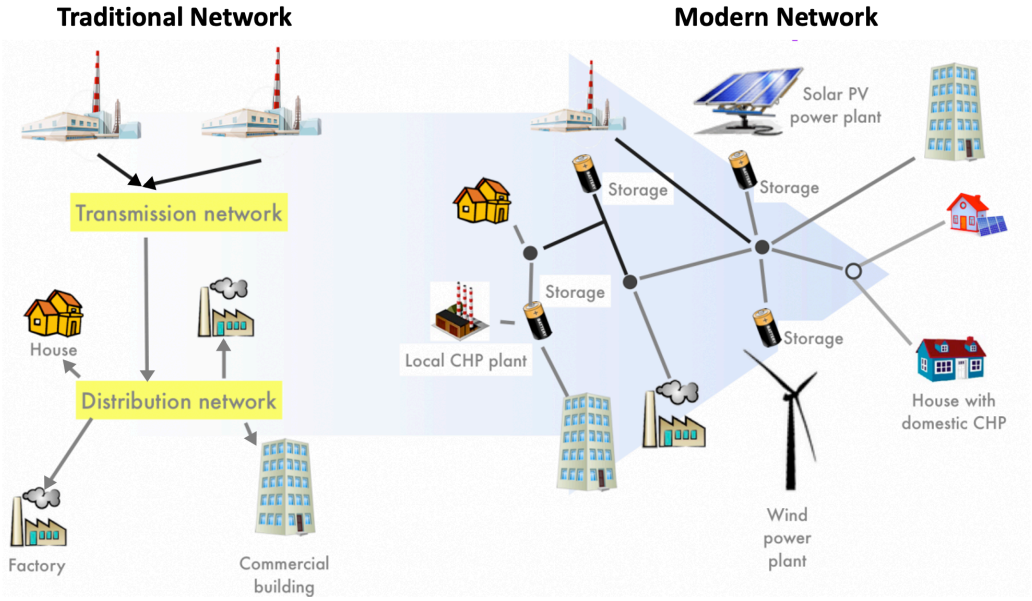


Figure 3: Traditional vs Modern Network [7]

In order to study the impact of these new technologies, an architectural model has to be chosen. We are going to be studying Microgrids. We are choosing the study of microgrids over a traditional generation-transportation-consumption network due to the concepts we are going to be looking at. This is due to the inability of a traditional network to incorporate DERs in the calculations that are processed.

1.2. MicroGrids

The MicroGrid concept is stablished around the concept that modern electrical installations are not linearly built. Microgrids are defined as a cluster of loads and micro-sources operating as a singular controllable system that provides power as well as heat to its local area [8]. We can picture a microgrid as a fragment of the system that takes into account a small area. One example of the same could be a small village that counts with PV installations, wind farms, PHEVs and other installations which can be relevant for the study of electrical networks. The implementation of microgrids has some objectives, such as. improving reliability, minimizing losses or avoiding blackouts.

Microgrids have two operating conditions, islanded and grid-connected:

- Grid-Connected: When the system is working with all its DERs and resources connected.
- Islanded: Once there is a failure in the system that leads to a disconnection of a part of the microgrid, the isolated area enters Islanded mode. In this mode, there needs to be at least one DER in order to allow for the system to control itself, confronting voltage and frequency deviation.

Regardless of the mode the microgrid is operating in, it consists of two layers, physical and electronic

- Physical-Layer: the physical layer is referred to the microgrid, which in the case that we are going to be studying is going to be carried out by the Virtual HIL Device from Typhoon (C-HIL Testbed)[3.1.1] and the Texas Instruments MSP-EXP432e401. These two devices will work collaboratively so as to simulate the behavior of the network since testing these algorithms in a real-life microgrid can be very expensive. Apart from the economic, the implementation of experimental algorithms into the microgrid might result in a major failure that can infer in the microgrid not performing as expected in the future.

- Cyber-Layer: the cyber layer of a microgrid is based on the computational states. It is the layer in charge of computing the algorithms that need to be run in order to evaluate the state of the grid and compute the changes that need to be done in the DERs so that the grid goes back to the rated values. In our case, the cyber layer is described by an Arduino [3.1.2] and the codes that are implemented.

The implementation of microgrids for research purposes is not the best approach for several reasons. The first reason is the lack of normalized models, since the number of physical microgrids built for research is scarce. Therefore, the implementation of new control software would have rigorous previous testing of the microgrid in order to normalize it. Second, after the network is normalized and ready to be tested, the implementation of erroneous algorithms or conditions could result in a grid malfunction and even a condition violation that could damage the measuring instruments and devices. From these statements we can infer how the safest way to do research with microgrids is the utilization of simulated microgrids.

Throughout this project we are going to be using a network simulator, the Typhoon HIL device, that is described in [3.1.1].

1.3. Centralized Approach

This approach is based on the hierarchy of one node over the rest. This node is what we will denote as the controller. The centralized architecture is based on three major phases that can be seen in Figure 4.

1. Communication: The first phase is the communication between nodes and controller. It will have as purpose the broadcast of state variables that will describe the operating conditions of the network. Some of the information that is transmitted can be the voltage level, frequency deviation or operational state (to check if there if any node is down).

2. Processing: The second phase is the interpretation of the information received from the nodes. Once the information is obtained, it will be compared with the reference values. In case any irregularity is detected, the computational stage starts. In this stage several algorithms are conducted in order to determine the optimal conditions that will make the network go back to an operational state.

3. Broadcast: Once the computational phase is over, we obtain as results the changes that need to be made to the DERs´ setpoints in order to obtain the desired conditions in the microgrid. These can be from a change in the DER´s reactive or active power input, to the decision to put one of the loads down. These changes are broadcasted to the nodes, who will proceed to follow the directions from the controller.

- Disadvantages: Due to the implementation of DERs, the probability of node failures increases, and with that increase comes the vulnerability of the controlling node. If the controlling nodes falls due to a failure in the network, the system will remain uncontrolled until that node is repaired. That is a very costly operation and makes the network vulnerable to minor failures like overvoltage or frequency deviation. These faults can result in very costly outages or power blackouts.

- Advantages: The main advantage of centralized over distributed systems is the decrease in difficulty of implementation of algorithms in the control architectures. In a centralized architecture every node communicates with the controller device in order to compute the system state. This results in a smaller computation scheme. The algorithms that need to be implemented in order to restore normality are going to be easier and therefore will require less time to compute than those of the distributed architecture. Another advantage is the ease of optimization procedures. All the information required for these calculations is available at the controlling node. On the contrary, distributed architectures don't have that information at every controller, therefore making it a more complex calculation. This results in an easier and faster control.

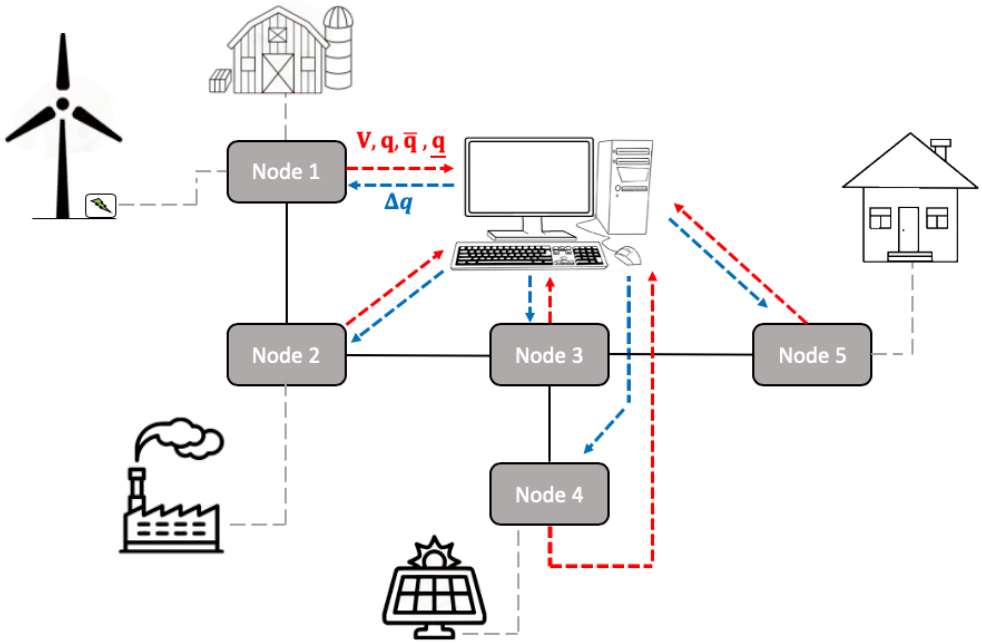


Figure 4:Centralized Network Diagram

1.4. Distributed Approach

This approach is based on the collaboration between nodes. We get rid of the controller node that is in charge of the computation. in contrary to the centralized approach, there will be a control module on each of the nodes. This might result in a more expensive overall installation, but in the future, the cost and reliability of the network will make up for the initial investment. This architecture is based on these three phases:

1. Node check: This is the recursive phase in the which the node will stay until there is the need to proceed to phase 2, the controller in the node will evaluate the information about the behavior of the same in order to determine whether it operating according to its rated values or if it is experiencing any form of malfunction. if the node finds a violation of the rated values (overvoltage, frequency violation…) it will enter the computational phase of the control.

2. Processing: In this phase the node´s controller will implement the algorithms in order to determine the alterations that need to be done. These changes will be analyzed in regard to the available resources in the DERs. If the node is able to fulfil the request by itself, it will proceed to do so. In the case that the amount required of reactive or active power, or any measure required, is superior to the amount available, the node will send a flag to its neighbors. This flag will initiate phase 3 in the which the nodes will collaborate to supply the resources needed.

3. Collaborative Processing: when one of the nodes cannot provide the required amount, the remaining has to be delivered by the rest of the nodes in the network. If the calculation was through a centralized algorithm in the which one node had all of the information, this would be a very easy computation. in a distributed network, the node will only have its own information and that of its neighbors. In order to obtain the amount each of the nodes has to provide, the nodes will perform the distributed algorithm of ratio consensus, which is described in [Appendix C]. Once the algorithm is finished and the controllers set the nodes to the appropriate conditions, the system will go back to phase 1.

- Disadvantages: The increased complexity over centralized controls makes this approach more time consuming and computationally difficult.

- Advantages: The main advantage of distributed over centralized systems is the low communication overhead and fast control response. This is related with the speed of the system. Distributed algorithms will have a faster recovery time after a fault. This can be seen in [5.4.2.3]. The second advantage distributed algorithms have over centralized architectures is the increased reliability, which can be measured in two manners. The first one is the fewer rebounds the signal gives after a fault is detected compared to the oscillatory response from a centralized system. On second stance, the system is less vulnerable to single failures. When the controller of a centralized network fails, the network stays unregulated. However, in a distributed network, when a node fails, the neighboring nodes will account for it regulating the network.

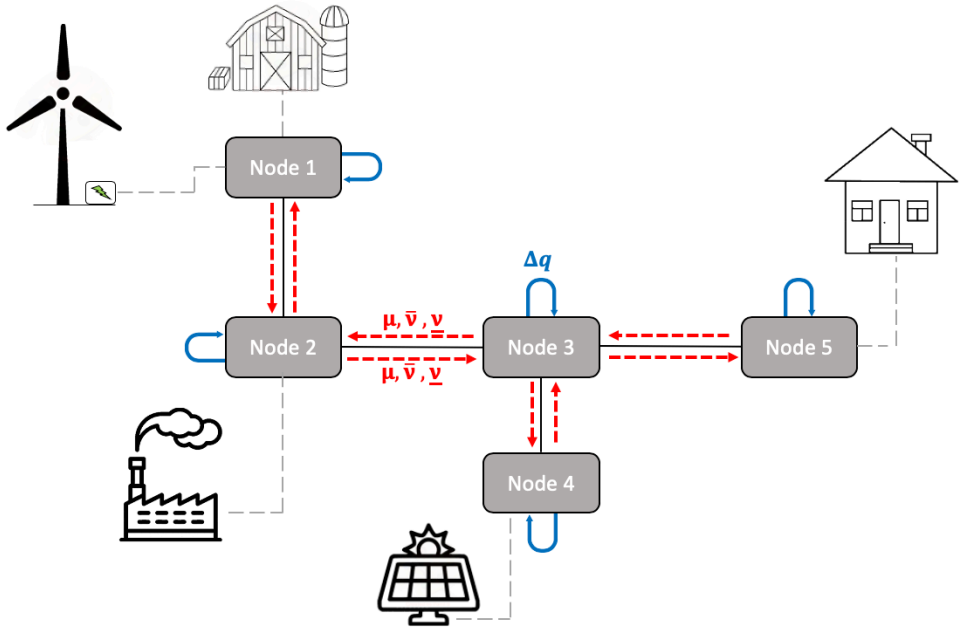


Figure 5: Distributed Network Diagram

2. Depiction of the project

2.1. Justification

This project is a result of the voltage control algorithms described in [4]. In this paper the authors introduce a set of algorithms for the voltage control in distribute microgrid. This comes from the desire to change the control architecture of microgrids from centralized to distributed.

Our team is from the University of Illinois at Urbana Champaign (UIUC). And we are developing a distributed microgrid control architecture. This scheme is composed of a number of control algorithms that aim towards the simulation and control of a microgrid, in this case a simulated one. These algorithms not only include control paths, such as voltage control, frequency control or optimal power flow. But also include every step in the regulation of grids, from wireless communication in between nodes to synchronization and initialization of procedures.

This project is based on the implementation of the algorithm for voltage control into C++. This is due to the need to create a library with all the control algorithms a network will need to implement in order to ensure the proper performance of the microgrid. This code will later be used by Arduinos in order to control a simulated grid by the virtual HIL simulator, as explained in [3.1].

2.2. Objectives

There are three objectives for this project, organized from top to bottom according to priority and importance.

1. Construction of the Arduino code: this step is the coding of the algorithms in C++ following object-oriented programing technics. The code is defined in [5.2.3] and consists of several layer of code.

2. Reinforce the Ratio Consensus Algorithm: The ratio consensus code has been written several times during the life of the project, each of the iteration has had the objective of increasing its efficiency and minimizing the processing time. It is described in [Appendix C].

3. Improve iteration precision: In the Ratio Consensus algorithm, the precision of the same is affected by the number of iterations that are run. The adequate number of iterations can be set experimentally, but a way to automize this project in order to stop the number of iterations at the optimal point, the MaxMin consensus.

2.3. Methodology

The methodology of the research project is divided into steps:

1. Review previous work about centralized and distributed voltage control algorithms.
2. Understand the concept of microgrid, the voltage control algorithms and the instruments.
3. Create flowcharts and class diagrams for the centralized and distributed codes.
4. Code the algorithms into C++ and test them in typhoon.
5. Study the results and come up with the conclusion of the project.

3. Description of the Technology

3.1. The Laboratory

The project is taking place in one of the laboratories of the University of Illinois at Urbana-Champaign. The material in this room can be divided into three categories.

1. Network simulation: The first step towards creating a virtual network is the definition of the topology that the system will have. This process is developed by one of the team members, A. Mayoral. The implementation involves the creation of a model of the network in a program named Typhoon that will be uploaded into the network first simulator, the Typhoon HIL.

2. Algorithm creation and testing: The second step into simulating the network is writing and testing the code that will implement the algorithms that the controllers will have to apply. This is done in two high power computers which are used by the team.

3. Code storage and communication: After the code is written and thoroughly tested for errors, it is stored into the memory of the Arduinos [3.1.2]. These Arduinos will act as nodes. Each Arduino can simulate multiple nodes. These Arduinos will communicate one with each other according to the structure and behavior set by the grid simulator.

3.1.1. Typhoon HIL Hardware

The Brain of the project is the C-HIL testbed. It will simulate the physical behavior of the network. It consists of three ultra-high-fidelity real-time simulation devices as seen in Figure 6. One Typhoon HIL 402 and two Typhoon HIL 603s [9]. These will allow us to simulate full-scale networks in order to analyze the behavior. It will simulate responses at step sizes as low as 0.5µs PWM sampling of 20 ns [9]. this allows for very high levels of fidelity. The computational power available through these network simulators gives the team the ability to emulate a large number of nodes. These devices are responsible for the high-end simulation and behavior of the network in the which the system evolves over time. They also store pre-defined information about each node. This information will later be transmitted to the lower-level controllers.

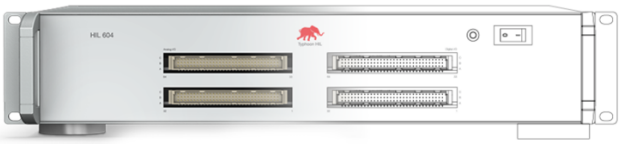


Figure 6: Typhoon HIL Device [10]

3.1.2. Lower-Level Controller Hardware

On the hierarch of the network. The first piece is the C-HIL bench. Below that, there are a sequence of levels that rely one on the orders to use the information from the C_HIL. The second layer are the Lower-level Controllers. The role of this controller is being performed by Arduinos. This layer of control is in charge of performing lower level DER control schemes. These are the ones that focus on issues like voltage and frequency violation; as well as oscillator control.

- Topology: The Arduinos modules, described in Figure 7, are composed of three layers. The first one is the board that will be the main component. The board is in charge of storing the code and implementing it with the inputs received from the C_HIL devices. The second layer is the Ethernet shield. The third layer is the XBee (RF) module.

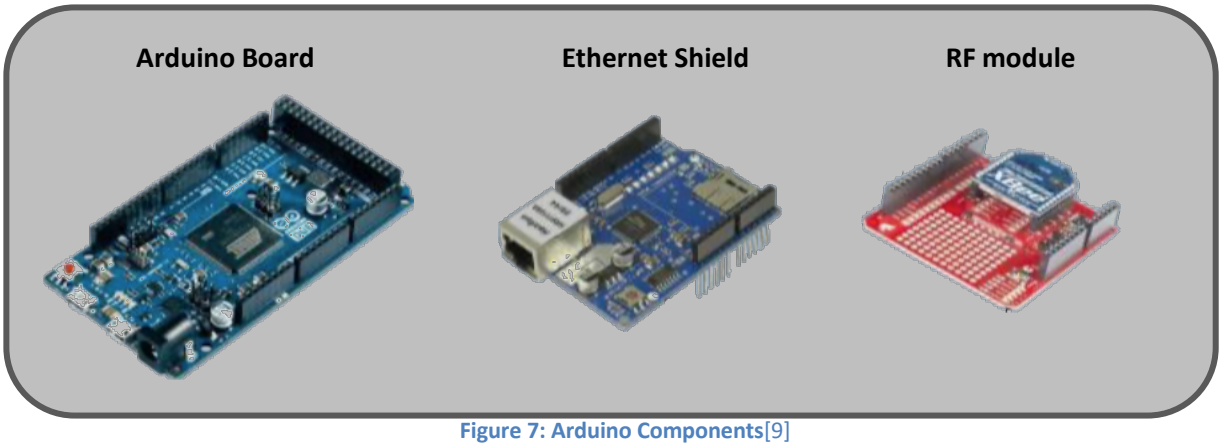


Figure 7: Arduino Components[9]

- Behavior: The first and most important task is the communication in between one another. It is carried out by the XBee module and the ethernet. The XBee module is in charge of the wireless communication in between the Arduinos themselves. The Arduinos need to have bidirectional communication with one another in order to implement a distributed architecture. The Ethernet Shield is responsible for wired communications with the computers. Its purpose is to upload the code from the computers into the Arduinos and the display of information back in the computers after the computations are completed in the board.

4. Literature Review

The most important source of information for this project has been the paper by Professor Dominguez[4]. This paper has been the source of knowledge for this project. Thanks to it I have learned about distributed controls in microgrids and overall basic knowledge in the modern paradigm of electric grids and how to manage them. Apart from that, this paper is the one in the which this thesis is based on. This is because the algorithms that are being implemented are extracted from this paper.

5. Description of Research Results

5.1. Voltage control Implementation

For a grid to work in optimal conditions there needs to be a limit to the deviation in voltage. Voltage is set to a Voltage Reference in the DERs and the limits will be set at ± 5%. This reference value is considered to be 1pu since the V_{ref} that will be chosen will the base value. Voltage will tend to match the reference value, but this value can be altered. This deviation in voltage comes from variations in reactive and active power input and consumption. The most common scenario in the which we encounter this form of failure is the failure of a node. This can be given as result of a generator plant falling or the disconnection of load. It can lead to a variation in the power scheme in the grid and will result in a variation in the voltage profile, therefore incurring in a violation of the limits. Another scenario in the which we can come across a variation in voltage is the implementation of DERs in microgrids. These resources are characterized for their variable input/output power curves as can be seen in Figure 1 and Figure 2.

The variation of active and reactive power as a function of time r in PQ buses is defined in [4] as ΔS[r+1]= S[r+1] -S[r]. from this expression we can define the expression for the computation of power flow in:

$$\begin{bmatrix}\Delta \mathbf{P}[\mathbf{r}] \\ \Delta \mathbf{Q}[\mathbf{r}]\end{bmatrix} = \begin{bmatrix}\mathbf{H} & \mathbf{N} \\ \mathbf{K} & \mathbf{L}\end{bmatrix} * \begin{bmatrix}\Delta \boldsymbol{\theta}[\boldsymbol{r}] \\ \Delta \boldsymbol{V}[\boldsymbol{r}]\end{bmatrix}$$

From where we obtain:

(1)

$$\boldsymbol{H} = \left[\frac{\partial \boldsymbol{Pi}}{\partial \boldsymbol{\theta j}}\right], \boldsymbol{N} = \left[\frac{\partial \boldsymbol{Pi}}{\partial \boldsymbol{Vj}}\right], \boldsymbol{K} = \left[\frac{\partial \boldsymbol{Qi}}{\partial \boldsymbol{\theta j}}\right], \boldsymbol{L} = \left[\frac{\partial \boldsymbol{Qi}}{\partial \boldsymbol{Vj}}\right]$$

These equations describe the variation of active and reactive power in the system due to change in V and θ. We will be operating under the Decoupled assumption, in the which it is stated that the variations in voltage due to a change in active power injection is neglectable, we will be studying the system on the changes in reactive power as a means to change the voltage profile.

For the computation of the first stage control, we need to obtain the system´s behavior, which is described by the sensitivity matrix.

We will start from the Power Flow equations, where i & k are the nodes at both ends of the line, $\mathcal{N}$i is the set of neighbors to node i and B and G come from the Y_{bus} Matrix as $Y_{ik}=G_{ik}+jB_{ik}$.

$$\mathbf{P_i} = \sum_{\mathbf{k}\in\boldsymbol{\mathcal{N}_i}} \mathbf{V_i V_k}[\mathbf{G_{ik}}\boldsymbol{cos}(\boldsymbol{\theta_i} - \boldsymbol{\theta_k}) + \mathbf{B_{ik}}\boldsymbol{sin}(\boldsymbol{\theta_i} - \boldsymbol{\theta_k})]$$

$$\mathbf{Q_i} = \sum_{\mathbf{k}\in\boldsymbol{\mathcal{N}_i}} \mathbf{V_i V_k}[\mathbf{G_{ik}}\mathbf{sin}(\boldsymbol{\theta_i} - \boldsymbol{\theta_k}) - \mathbf{B_{ik}}\mathbf{cos}(\boldsymbol{\theta_i} - \boldsymbol{\theta_k})]$$

And from the infinitesimal assumption that all of the angles will be near to zero, we can approximate $\cos(\Delta\theta)\approx 1$ and $\sin(\Delta\theta)\approx \Delta\theta$, obtaining the simplified power flow equations:

(2)

$$\mathbf{P_i} = \sum_{\mathbf{k}\in\mathcal{N}_\mathbf{i}} \mathbf{V_i V_k}[\mathbf{G_{ik}} + \mathbf{B_{ik}}(\boldsymbol{\theta_i} - \boldsymbol{\theta_k})]$$

(3)

$$\mathbf{Q_i} = \sum_{\mathbf{k}\in\mathcal{N}_\mathbf{i}} \mathbf{V_i V_k}[\mathbf{G_{ik}}(\boldsymbol{\theta_i} - \boldsymbol{\theta_k}) - \mathbf{B_{ik}}]$$

From the use of (1), differentiating the equations (2) and (3), we obtain the equation that will give us the system parameters H, N, K and L.

For i≠k

(4)

$$\mathbf{H} = \frac{\partial \mathbf{P_i}}{\partial \boldsymbol{\theta_k}} = -\mathbf{V_i V_k B_{ik}} \qquad \mathbf{N} = \frac{\partial \mathbf{P_i}}{\partial \mathbf{V_k}} = \mathbf{V_i}[\mathbf{G_{ik}} + \mathbf{B_{ik}}(\boldsymbol{\theta_i} - \boldsymbol{\theta_k})]$$

$$\boldsymbol{K} = \frac{\partial \boldsymbol{Q_i}}{\partial \boldsymbol{\theta_k}} = -\boldsymbol{V_i V_k G_{ik}} \qquad \mathbf{L} = \frac{\partial \mathbf{Q_i}}{\partial \mathbf{V_k}} = \mathbf{V_i}[\mathbf{G_{ik}}(\boldsymbol{\theta_i} - \boldsymbol{\theta_k}) - \mathbf{B_{ik}}]$$

For i=k:

(5)

$$\boldsymbol{H} = \frac{\partial \boldsymbol{P_i}}{\partial \boldsymbol{\theta_k}} = \sum_{\boldsymbol{i}\neq\boldsymbol{k}} \boldsymbol{V_i V_k B_{ik}} \qquad \boldsymbol{N} = \frac{\partial \boldsymbol{P_i}}{\partial \boldsymbol{V_k}} = \sum_{\boldsymbol{k}\in\mathcal{N}_\boldsymbol{i}} \boldsymbol{V_i}[\boldsymbol{G_{ik}} + \boldsymbol{B_{ik}}(\boldsymbol{\theta_i} - \boldsymbol{\theta_k})]$$

$$\boldsymbol{K} = \frac{\partial \boldsymbol{Q_i}}{\partial \boldsymbol{\theta_k}} = \sum_{\boldsymbol{i}\neq\boldsymbol{k}} \boldsymbol{V_i V_k G_{ik}} \qquad \boldsymbol{L} = \frac{\partial \boldsymbol{Q_i}}{\partial \boldsymbol{V_k}} = \sum_{\boldsymbol{k}\in\mathcal{N}_\boldsymbol{i}} \boldsymbol{V_i}[\boldsymbol{G_{ik}}(\boldsymbol{\theta_i} - \boldsymbol{\theta_k}) - \boldsymbol{B_{ik}}]$$

For the computation of the sensitivity matrix, we will need to compute the matrices H,K,N,L. for the construction there will be used in (4) for the elements in the diagonal, and in (5) for the out-diagonal elements.

Once we have H,K,N,L, the computation of the Sensitivity matrix will be obtained from the expression :

(6)

$$S=(L-KH^{-1}N)^{-1}$$

The voltage levels will be monitored by the controller. The reference values for the control will be stablished in accordance with the base voltage and a percentage of security chosen beforehand to ensure the proper functioning of the system. The voltage limits will be stablished as V=Vref ± Vref*(%/100) as can be explained in Figure 8.

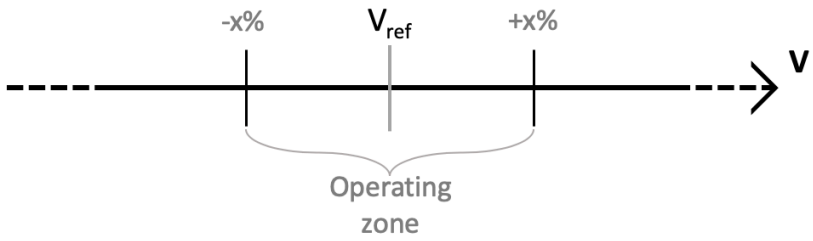


Figure 8: Voltage limits

The controller is going to be recurrently evaluating the voltage level in order to prevent voltage violations. If the controller detects a violation, it will start the implementation of the Voltage Control algorithm. The Voltage Control algorithm consists of two stages:

- 1st Stage: The first stage is the self-adjustment step. In it the node in the which there has been a failure is in charge of its own regulation. The controller will compute the algorithms that will obtain as a result the target value for the reactive power in the DER. The controller will evaluate the quantity obtained through its calculation with the level of available resources from the node.

  - If the node can supply the change in reactive power needed for the implementation of the first stage, it will do so, mitigating the failure and restoring the normal operation conditions. This will bring the voltage profile back to the operating zone shown in Figure 8.

  - If the node cannot supply the change in reactive power needed, the system will initiate the second stage control. First the faulty node will set its reactive power to the limit ($\overline{q} or \underline{q}$) depending on the nature of the failure (overvoltage or undervoltage). Second, the computation of the second stage algorithms will commence.

- 2nd Stage: The second stage is the collaborative step in the Voltage control algorithm. It is described as a request for help from the faulty node to its neighbors. When the faulty node cannot provide the required amount of reactive power that the network needs in order to mitigate the problem, the system starts the collaborative stage. In it all the neighboring nodes will help injecting/consuming reactive power in order to restore the voltage profile. Once the controller asks for help from the neighboring nodes, the second stage takes off.

5.1.1. Flowchart

Figure 9 shows the flowchart that describes the steps that take place in the voltage control algorithm is:

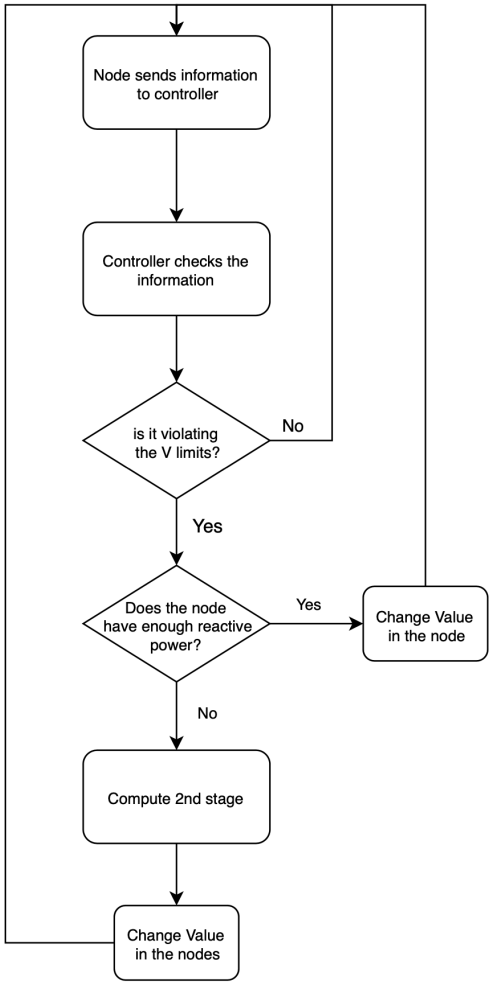


Figure 9: Voltage Control Flowchart

5.2. Distributed Approach

5.2.1. state Diagram

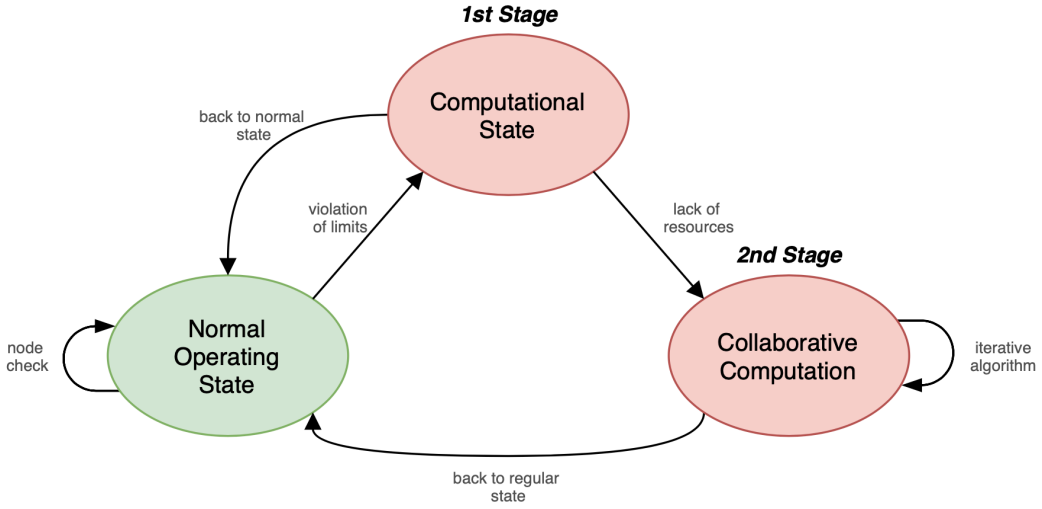


Figure 10: Distributed Actuation diagram

5.2.2. Algorithm

5.2.2.1. First Stage

Once the controller detects a voltage violation, the system will start computing the first stage controlling algorithm. In the first stage the algorithms will try do an auto-reparation strategy. The required reactive power needed by the node will be supplied by itself in order to minimize distributed approaches and increase efficiency and increase the speed of response.

First, the reactive power needed is calculated. It is a function of the difference between the voltage level and the limit in the DER ; the participation factor ($\boldsymbol{\alpha}$) and the node´s sensitivity (s_{jj})computed with (6).

- Over-Voltage:

(7)

$$\Delta \mathbf{q_k}[\mathbf{r}] = \frac{\boldsymbol{\alpha}}{\mathbf{Sjj}}(\underline{\mathbf{V_j}} - \mathbf{V_j}[\mathbf{r}])$$

- Under-Voltage:

(8)

$$\Delta \mathbf{q_k}[\mathbf{r}] = \frac{\boldsymbol{\alpha}}{\mathbf{Sjj}}(\overline{\mathbf{V_j}} - \mathbf{V_j}[\mathbf{r}])$$

Once the required reactive power is computed, the amount of remaining resources has to be taken into account. If there is enough the system will change the node´s reactive power target level and the control algorithm will have finished. If one of the nodes doesn´t have enough reactive power, the faulty node will be categorized as a saturated node and all the nodes in the system will enter the Second Stage part of the algorithm.

5.2.2.2. Second Stage

This is the collaborative part of the algorithm. In contrast to the centralized voltage control algorithm in the which you have all the information from all the nodes, this algorithm is built following a distributed architecture which makes this step more complex.

Before the start of the algorithm, there needs to be an initialization. We have to initialize ν and μ, which are going to be the auxiliary variables for the implementation of the ratio consensus in the nodes so as to obtain the required amount to rise or decrease in reactive power in order to mitigate the fault. For the explanation of the ratio consensus we need to define the concept of the reactive power target as [4] $\widehat{q_j}[r+1] = q_j[r] + \Delta q_j[r]$.

The first stage is the initialization of the auxiliary variables. This initialization is characterized by the state of the node.

- Over-saturated:

$$\begin{cases} \mu[0] = \widehat{q_j}[r+1] - \overline{q_j} \\ \overline{\nu}[0] = 0 \\ \underline{\nu}[0] = \underline{q_j} - \widehat{q_j}[r+1] \end{cases}$$

- Under-saturated:

$$\begin{cases} \mu[0] = \widehat{q_j}[r+1] - \underline{q_j} \\ \overline{\nu}[0] = \overline{q_j} - \widehat{q_j}[r+1] \\ \underline{\nu}[0] = 0 \end{cases}$$

- Between q limits:

$$\begin{cases} \mu[0] = 0 \\ \overline{\nu}[0] = \overline{q_j} - \widehat{q_j}[r+1] \\ \underline{\nu}[0] = \underline{q_j} - \widehat{q_j}[r+1] \end{cases}$$

Once the variables are initialized, the system will compute ratio consensus in order to find the quantity that the DER have to change their q level in order to help the faulty node. The ratio consensus algorithm is explained in [Appendix C]. We will compute two ratio consensus algorithms. One with the upper nu and one with the lower nu. Once we reach consensus, depending on the value of μ one of the results will be chosen. The reactive power variation due to the second stage algorithm is defined by η. The ratio consensus function will be denoted as RC(y,z) The formulation for the ratio consensus is:

$$\overline{\eta} = RC(\mu, \overline{\nu})$$

$$\underline{\eta} = RC(\mu, \underline{\nu})$$

The criteria for the election of η is based on the sign of μ with k being the final iteration number:

$$\eta = \begin{cases} \overline{\eta} & \mu_j[k] > 0 \\ \underline{\eta} & \mu_j[k] < 0 \end{cases}$$

On the last step of the Voltage Control algorithm, the limits need to be evaluated. The target values for the nodes will be decided as follows[4]:

$$q_j[r+1]\begin{cases} \overline{q}_j, & \widehat{q_j}[r+1] + \eta_j > \overline{q}_j \\ \underline{q_j}, & \widehat{q_j}[r+1] + \eta_j < \underline{q_j} \\ \widehat{q_j}[r+1] + \eta_j, & otherwise \end{cases}$$

5.2.3. Description of the Code

The code has been written in C++ in the Sublime Text editor and the content is divided in two libraries, OGraph and OAgent. OGraph is in charge of defining the image of the network, assigning neighbors and node parameters. The objects that are defined in OGraph are the OLocalVertex, which stores the local parameters from the node, and the ORemoteVertex, which stores the parameters obtained from the broadcasted information packages that are transmitted from one Arduino to the order in order to implement Ratio Consensus. The class diagrams that define ORemoteVertex and OLocalVertex are in [Appendix A].

The code is divided in several function inside an Arduino library, which are shown in [Appendix A] and are divided as follows:

- voltageControl: The core of the algorithm. From it we will be calling the rest of the functions.
- initializeVoltageControl: The initialization of the parameters in the OLocalVertex for the voltage control algorithm.
- IsOver/UnderVoltage: These two functions will check if the node is experiencing voltage violation over or under the limit.
- firstStageControl: The implementation of the First Stage of the algorithm. The operation that are computed are specified in [5.2.2].
- shareFlag: This function will broadcast whether the nodes need to implement the Second Stage voltage control or not.
- initializeVariablesSecStage: the initialization of the values for the auxiliary variables for the second stage ratio consensus computation.
- secondStageControl: The implementation of the Second Stage algorithm. The collaborative algorithm that is computed in [5.2.2].

Finally, the code returns the new reactive power setpoint for the node. this number will be passed to the Typhoon HIL device.

5.2.4. Flowchart

Figure 10 shows the flowchart that describes the steps that the code takes in order to implement the Distributed Voltage Control algorithm.

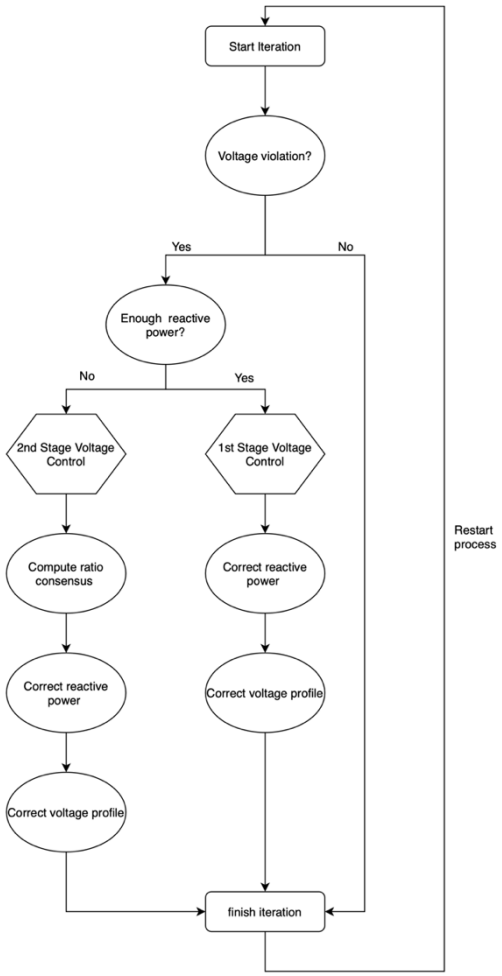


Figure 11: Distributed Voltage Control Code Flowchart

5.3. Centralized Approach

5.3.1. Actuation Diagram

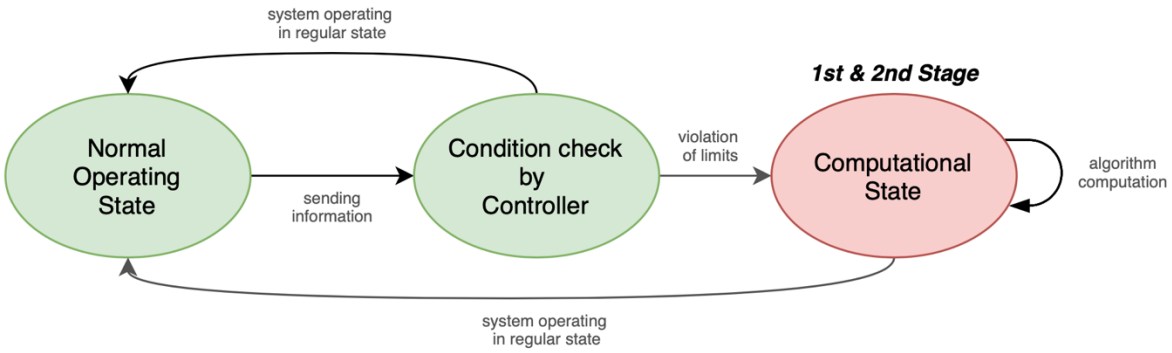


Figure 12: Centralized Actuation diagram

5.3.2. Algorithm

The information from the nodes is going to be introduced In the Arduino, the which will compute testing until it detects a voltage violation. Once the violation is found, the Arduino will implement the Voltage control algorithm in the Centralized approach.

Since there is only one Arduino in charge of the computation, the times will be shorter since the communication in between controllers is eliminated.

The control will start by the implementation of the first stage.

5.3.2.1. First Stage

The implementation of the first stage of the algorithm is the one where the failure will try to be fixed within the faulty node. the algorithm that is going to be implemented for the first stage is the same as for the distributed approach. The equations that are going to be implemented are the same as with the distributed network. We will use (7) for the over-voltage case and (8) for the under-voltage. If the node is not undergoing any fault, the reactive power variation in this node will be zero (**ρ**j=0). After obtaining the q_{target} for each of the nodes, the algorithm will compare the value with the reactive power reserve. In case the reactive power needed exceeds the limits of the DER, the. Remaining power will be added to the reactive power that needs to be supplied by the second stage.

$$\hat{q_j}[r+1] = q_j[r] + \rho_j[r]$$

$$q_{sec} = \sum_{\hat{q}>\overline{q}} \overline{q} - \hat{q} + \sum_{\hat{q}<\underline{q}} \hat{q} - \underline{q}$$

5.3.2.2. Second Stage

This is the collaborative part of the algorithm in the which the neighboring nodes will vary their reactive power level in order to fix the failure that the first stage could not fix.

The lack of need of communication makes the code simpler. There is no need to implement ratio consensus. The required reactive power variation in the second stage is computed proportionally to the reserve the node has:

$$\Delta q_{sec}^{j} = q_{sec} * \frac{q_{reserve}^{j}}{q_{reserve}}$$

Once the variation of reactive power in the second stage is computed, there needs to be another check of reactive power limit violation for the final q_{target}. Once the second stage is over, the reactive power level is communicated to the typhoon HIL device which will change the setpoints in order to get the system back to its normal operating state.

5.3.3. Description of the Code

The code is implemented directly Into the Arduino code due to the simplicity of the algorithm. The code is shown in [Appendix B].

The code is divided in:

- First Stage: The node tries to fix the failure by itself.
- Reserve Q check: The Reserve of Q is computed.
- Second Stage: the collaborative algorithm is computed with all the information that has already been recovered by the Arduino.

5.3.4. Flowchart

Figure 13 shows the flowchart that describes the steps that the Arduino code takes in order to implement the Centralized Voltage Control algorithm.

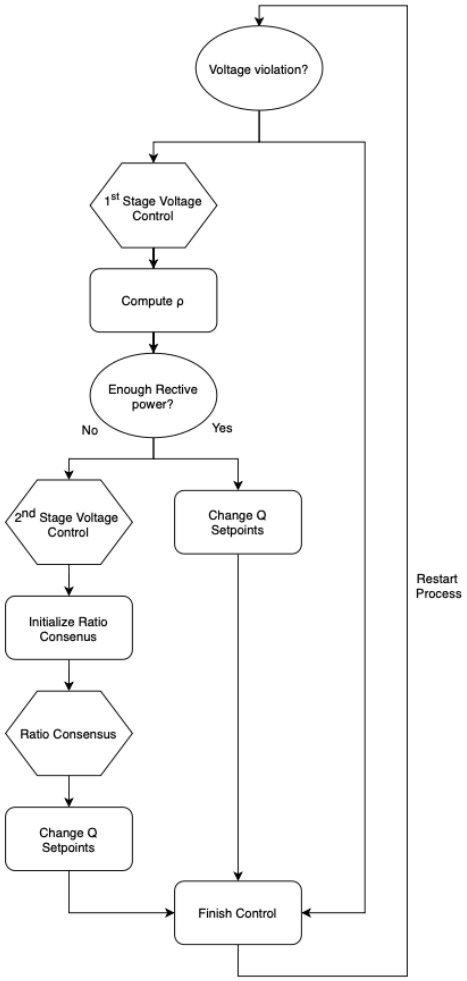


Figure 13: Centralized Voltage Control Code Flowchart

5.4. 6-node Example

5.4.1. Layout

The example that is going to be studied is a six-node configuration in radial layout. The layout of the system is as it is shown in Figure 14. The system consists of three generators (1,2,3) and three load (4,5,6) buses. This model will be implemented into the virtual HIL in order to compute the simulation. Each of the nodes in the system is modeled by an Arduino board. That board has access to the libraries that are uploaded from the Computer.

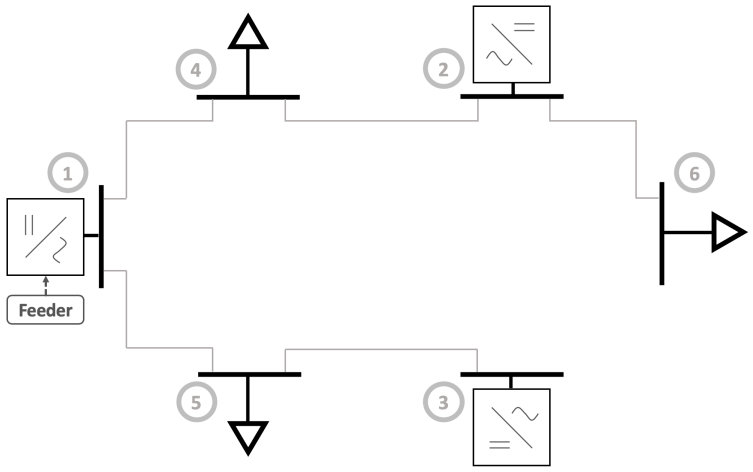


Figure 14: 6 Node Radial Model

The system is characterized by its line parameters, which are shown in Table 1 describing the resistance and impedance in between lines as well as the rated power at the which they operate. The nodes are characterized by their rated power which is the same for all and will be the voltage reference (V_{ref}) and the rated power at the which they operate. This is shown in Table 2.

Table 1: Line Parameters

Bus		Parameters		
from	to	R(pu)	X(pu)	S (KVA)
1	4	2.34592	3.04623	200
1	5	2.73691	2.35838	200
4	2	2.22862	3.04623	200
2	6	2.34592	2.35838	200
5	3	2.22862	4.42195	200

Table 2: Bus Parameters

Bus	V_{rated} (V)	S_{rated} (KVA)
1	277	200
2	277	200
3	277	200
4	277	6.6
5	277	6.6
6	277	6.6

From the values described in the previous tables we can define the power limits for the DERs. The values for the Reactive and Active power will be inferred from the P-Q diagram from Figure 15. From this diagram and using (9) , we can infer P and Q. There are many different criteria so as to arrive at a viable solution for the values. The criteria for the shape of the limits should be chosen accordingly to the role that the DER plays in the grid. With this and the grid experimental data we could make a decision in accordance with the reactive and active power that the DER is going to require. This technic maximizes the performance and minimizes the shortages in resources. Since for these simulations we do not have experimental data or operational data from the DERs, we will be choosing the least restricting criteria. The least restricting case is obtained following the square paradigm. In this scenario, the limits will be the same. For active and reactive power.

(9)

$$S^2=\sqrt{P^2+Q^2}$$

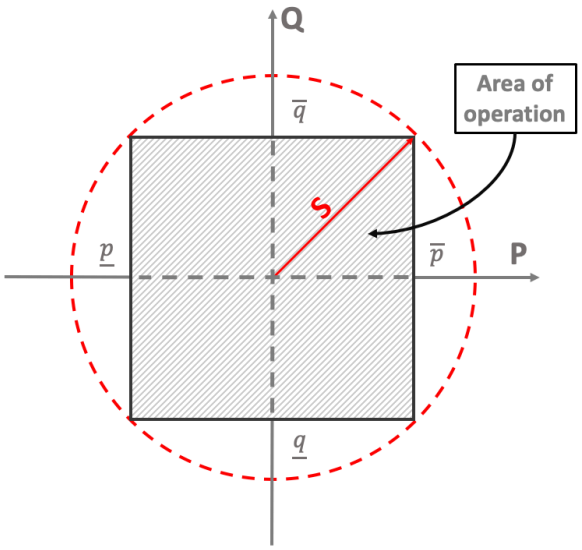


Figure 15: P-Q Diagram

From the parameters in Table 1 and Table 2; and the equation (6) we obtain the sensitivity matrix:

$$S_{ij}=\begin{bmatrix} -0.225736 & -0.005353 & 0.018581 & -2.96672 & 0.008358 & 3.51653 \\ -0.033996 & -0.224693 & 0.021049 & 0.001518 & 0.000087 & -2.96511 \\ 0.003489 & 0.003292 & -0.258852 & 1.84111 & -2.51039 & -2.17775 \\ -2.96534 & 0.00153 & 1.84085 & -0.225909 & 0.007286 & -0.032548 \\ 0.007543 & 0.000079 & -2.51089 & 0.043151 & -0.258583 & -0.051291 \\ 3.51514 & -2.96532 & -2.17847 & -0.003869 & -0.008873 & -0.227252 \end{bmatrix}$$

5.4.2. Centralized VS Distributed Results

The main difference when comparing these two approaches is the communication scheme they use. The centralized architecture is based on passing the results from the typhoon device into the Arduino, which will implement the algorithms by itself and after the results are obtained, they are broadcasted to the nodes. This can be seen in Figure 1. On the other side, the distributed algorithm is based on the communication in between Arduinos, especially for the Ratio Consensus algorithm [Appendix C]. Therefore, the structure is based on three Arduinos communicated in between each other in a triangular arrangement. Each of the Arduinos obtains the information from one of the nodes, then computes the singular and distributed algorithms. Once the Arduino obtains the new setpoints for the nodes, they broadcast it and proceed to the next iteration. This can be seen in Figure 17.

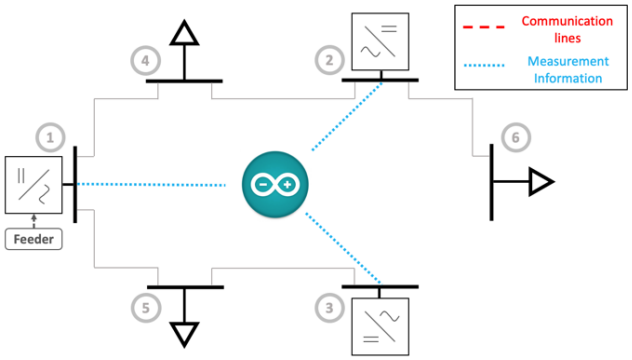


Figure 16: Centralized Model

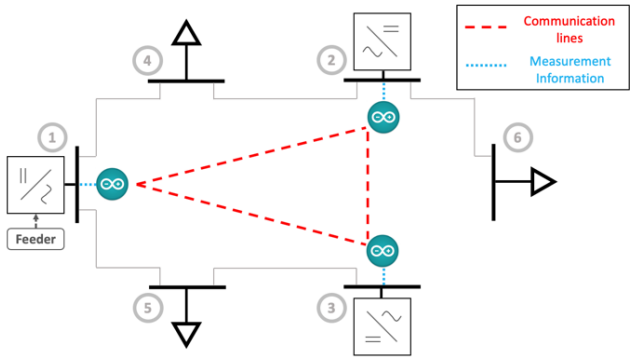


Figure 17: Distributed Model

5.4.2.1. Simulation

We have tested a simulation that is going to test the voltage control algorithm in two means:

4. Black-start: the microgrid will implement a black-start. This process consists on starting with all of the elements disconnected. Each of the elements will be connected in 30 seconds interval until the grid is fully connected.

5. Variable loads: the implementation will test the voltage control once the system is operating in fully connected mode. The curve the loads are going to follow is expressed in Figure 18.

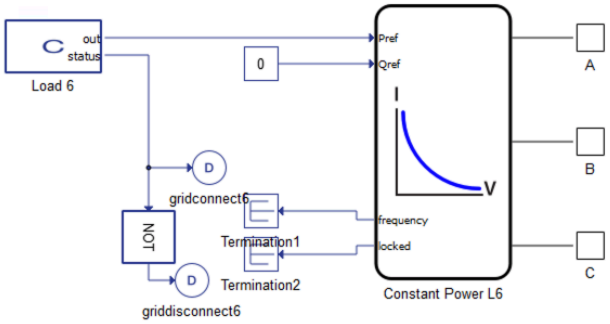
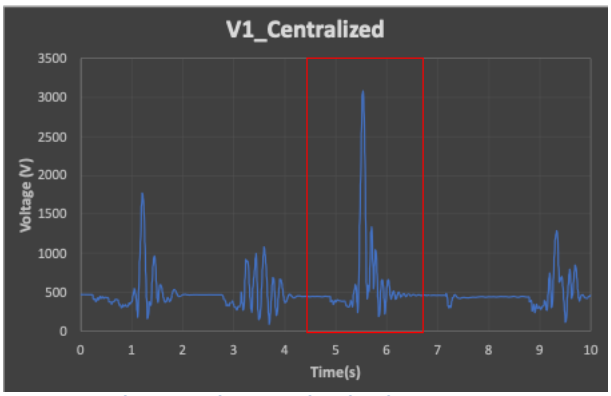


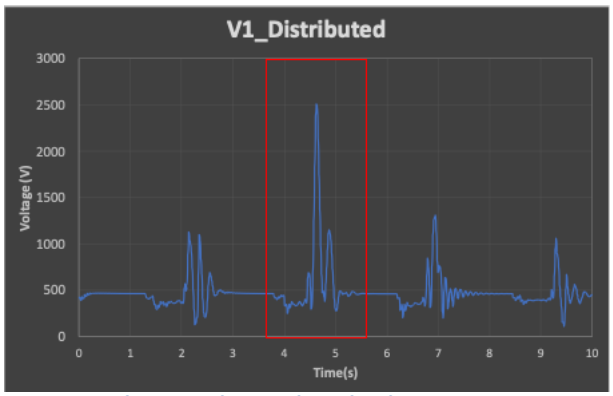
Figure 18: Typhoon Model of a Variable Load

5.4.2.2. Results

The results are obtained through the Typhoon HIL software and exported into an excel sheet. The captures are going to be displaying the reference of Voltage in function of the time. Graph 1 corresponds to a 10 second capture in the centralized model. Graph 2 corresponds to a 10 second capture in the distributed model. The capture speed in both of the models is 2ksps (kilo-samples per second). This trigger speed has been chosen due to the easier computational interpretation and least loss of data from the Typhoon model.

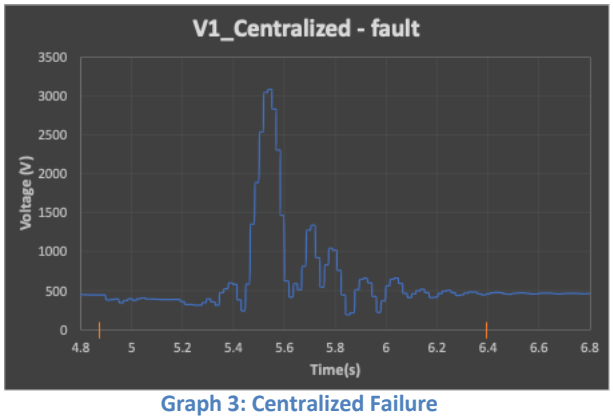


Graph 1: 6-node Centralized Voltage Capture

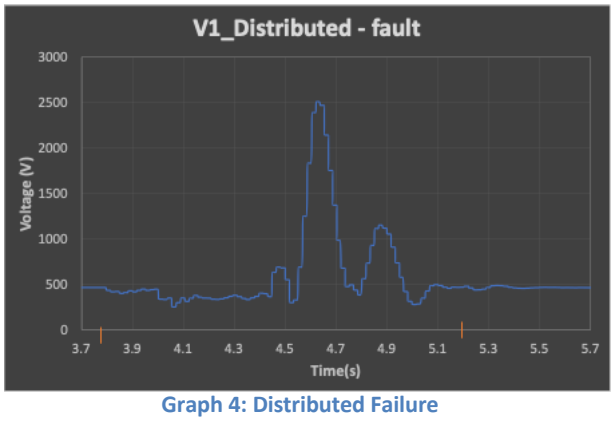


Graph 2: 6-node Distributed Voltage Capture

From the captures that can be seen in Graph 1 and Graph 2 we can isolate a particular failure for each of the distributions. The graphs that the describe the failures are shown below.



Graph 3: Centralized Failure



Graph 4: Distributed Failure

5.4.2.3. Interpretation of the results

The results obtained in Graph 1 and Graph 2 can be analyzed appealing to two main criteria, speed and stability of the response.

- Speed: The speed of the response will be evaluating the interval of time between the voltage violation and when the signal is inside the voltage limits.
  t_{cent}= 6.4 – 4.91 = 1.49 seconds
  t_{dist}= 5.53 - 3.8 =1.73 seconds
  From these results we can assure that the centralized control obtains a faster failure suppression.

- Stability: The stability of the response will take into account the damage that the voltage control can produce in the system in a worst-case scenario. The worst-case scenario for this test will be the faults that have voltage violations of up to 2500 Volts instantaneously.

    - Centralized: From Graph 3 we can observe that the response is more uniform. We can also notice how the curve has several rebounds which are up to 1500 Volts which might infer in damaging the instruments.

    - Distributed: From Graph 4 we can observe that the response follows an unusual and uneven form with two humps. After the humps, the systems goes straight into normal operational conditions.

From the analysis of both models there are two clear conclusions:

1. A centralized Control Architecture provides a faster response to a voltage violation.

2. A distributed Control Architecture provides a more stable response to a voltage violation. This comes from the fact that the recovering time after lowering the voltage is smaller. In the Distributed fault graph we can tell how after the second hump, the system quickly tends to the voltage reference. The centralized fault response has more humps that oscillate around the voltage reference, therefore making it less stable and more prompt to failure.

5.4.2.4. Conclusions of the 6-node Example

From the analysis of the results obtained by testing the Voltage Control Code in the six-node radial example we can infer two main results. The first one is that the Centralized Control is faster in overall response. The second one is that the Distributed Control is more stable and has a faster response after peak failure.

The implementation of one control or the other is a difficult matter, since both have numerous advantages and also disadvantages.

For the purpose of this project, the best architecture is a distributed one, since the versatility, speed and reliability makes it the best approach.

6. Future Steps

The future steps for this project can be divided into two subcategories:

- Short-term: The short-term goals for this project rely on the work that the team of the voltage control algorithms will be able to do throughout the fall semester 2020. These goals are two. The first one is the improvement in reliability and speed for the algorithms in order to narrow the gap with the centralized voltage control algorithm. The second is the implementation of the voltage control code into higher order networks. The best example for this is the Banshee network shown in Figure 19. Compared to the 6-node example, shown in Figure 14, is a leap in complexity.

- Long-term: The long-term goals for this project is the proposition of distributed microgrids control. From voltage control, to optimal dispatchment problems. This proposition will change the way we manage microgrids. And with the help of Typhoon, this research project will be something that will change the industry.

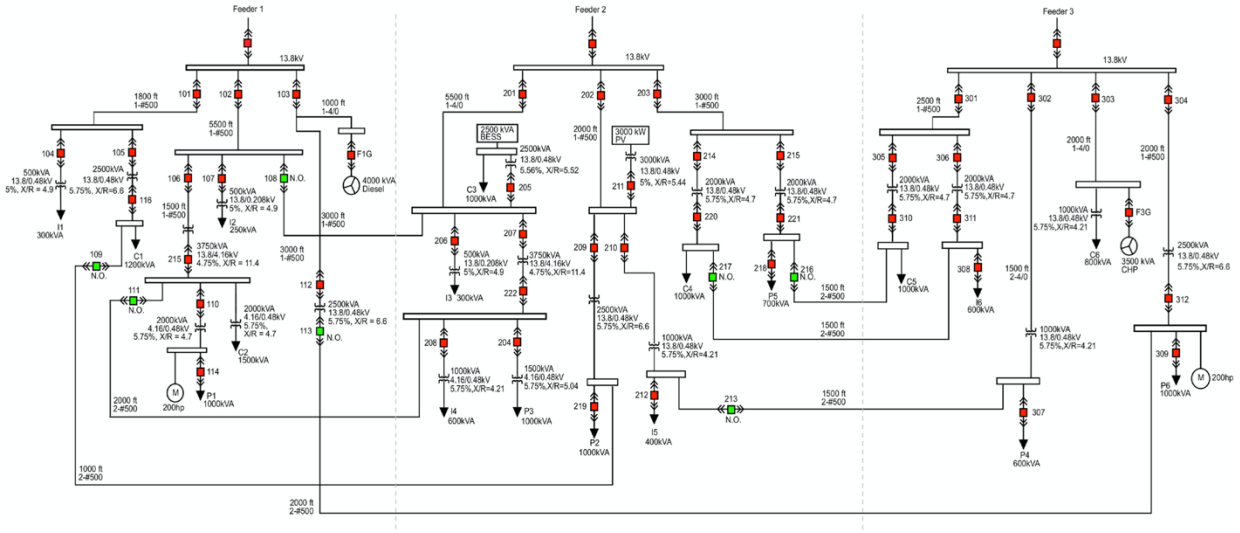


Figure 19: Banshee Network Model

7. Conclusion

We live in an energy hungry modern reality in the which power needs to be understood as a non-linear concept. This can be seen in Figure 3 and has made us change our train of thought. As the concept of power input and output has been reshaped, our way to control it has to do so too. We need to be able to control the network, and the best method to do it is the implementation of Microgrids. Microgrids are clusters of DERs (Distributed Energy Resources) that need to be controlled.

In order to create the control algorithms, test them and obtain results there needs to be a physical environment. Using a physical microgrid has many disadvantages. There are not many normalized microgrids from the which we could obtain contrasted results. They are less durable. Therefore, if there was an error in the control code that led the network to critical operating condition could damage it, incurring in an expensive repair process. For these reasons, for this project and all the ones that will study these algorithms and architectures, the team will be using simulated microgrids. They can be controlled following two different architectures, distributed or centralized. Throughout this paper we have studied the implementation of the voltage control algorithm for both control architectures. This has allowed us to compare both of them. However, the main goal was developing the distributed algorithm, since it is going to be implemented in further studies.

As we have seen throughout the result section [5.4], the best approach for this project in implementing voltage control in microgrids is following a distributed architecture. This comes from the fact that the control is going to have a more stable, less oscillating and with faster recover time response. Apart from the analytical results, the implementation of a distributed architecture is also better in terms of overall reliability. The superior reliability is due to the fact that a centralized approach has a single point of vulnerability, which makes the system very susceptible to failure. If the controller fails, the microgrid is unregulated until the controller is repaired. However, if we implement a distributed architecture, in case one of the controllers fails, the controllers from the neighboring DERs will supply for the fallen controller, resulting in a regulated and safe system.

However, in the future, the best option for the implementation of control would be a hybrid in between both architectures in the which we could obtain the speed of Centralized computing with the reliability and faster recovery of distributed controls.

Appendix A. Distributed Voltage Control Code

Classes

Figure 20shows the class diagrams of OLocalVertex and ORemoteVertex.

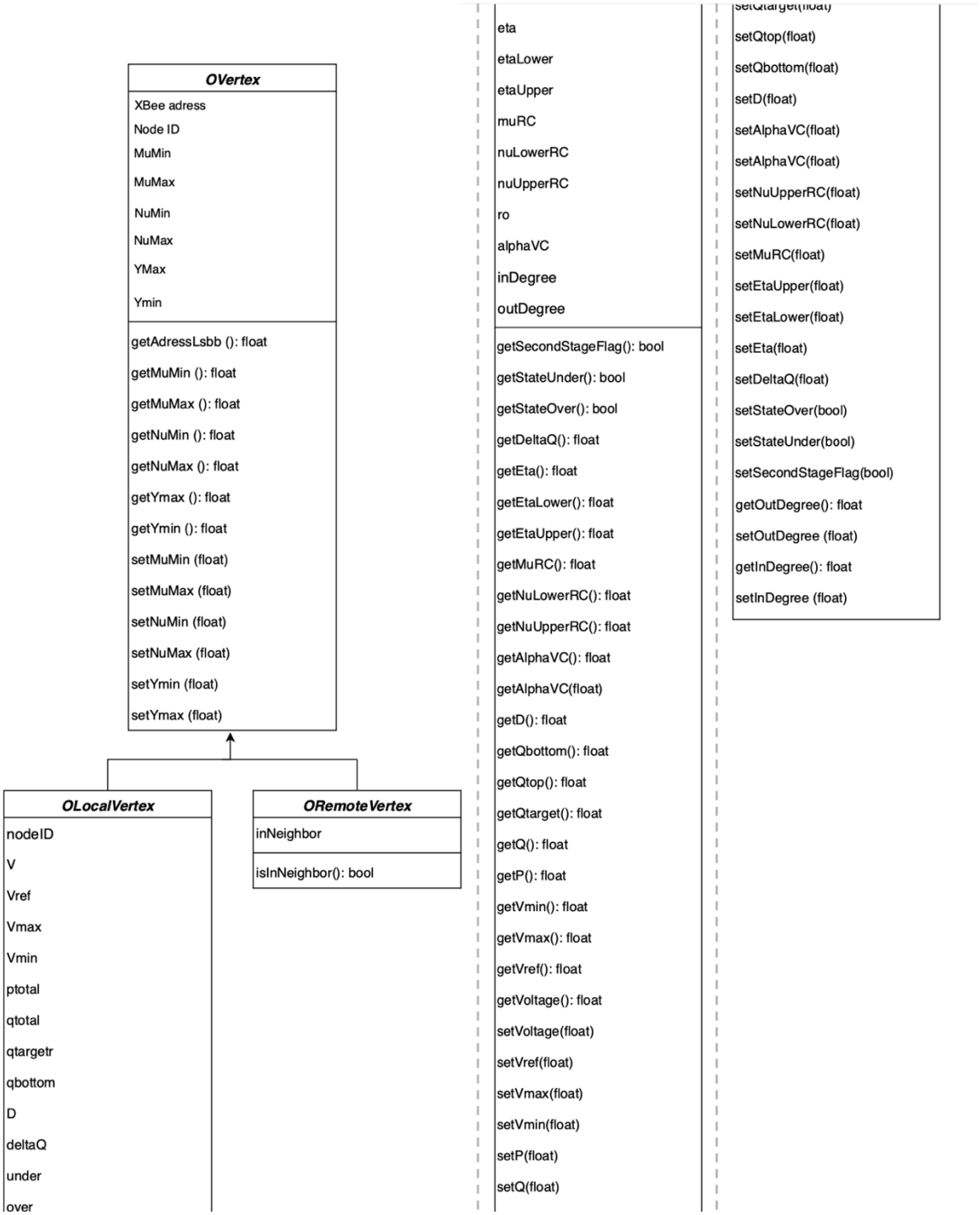


Figure 20: Class Diagram

Functions

- Voltage control body

```
loat OAgent::voltageControl_dist( float diffV, float Vref, float secPercentage, float
p, float q, float qtop, float qbottom, float Sij , float alphaVC, uint8_t iterations,
uint16_t period )  //it is going to give back the q required to rise or lower
{
    OLocalVertex * s = _G->getLocalVertex();
    _initializeVoltageControl( s, diffV, Vref ,secPercentage ,p, q, qtop, qbottom,
Sij, alphaVC );

    isUnderVoltage(s);
    isOverVoltage(s);
    firstStageControl(s);
    shareFlag( s, 9, period );

    if(s->getSecondStageFlag())
    {
        secondStageControl( s, iterations, period );

        if((s->getQtarget()+s->getEta()) > s->getQtop())
        {
            s->setDeltaQ(s->getQtop() - s->getQ());
            s->setQ(s->getQtop());

        }else if((s->getQtarget()+s->getEta()) < s->getQbottom())
        {
            s->setDeltaQ(s->getQtop() - s->getQ());
            s->setQ(s->getQbottom());

        }else
        {
            s->setDeltaQ( (s->getQtarget() + s->getEta()) - (s->getQ()) );
            s->setQ( s->getQtarget() + s->getEta());

        }

    }else{
        s->setDeltaQ( (s->getQtarget()) - (s->getQ()) );
        s->setQ(s->getQtarget());
    }

    return (s->getDeltaQ() + s->getQ());
}
```

- Voltage Control Initialization

```
void OAgent::_initializeVoltageControl( OLocalVertex * s, float diffV, float Vref,
float secPercentage, float p, float q, float qtop, float qbottom, float Sij, float
alphaVC )
{
    _G->clearAllStates();

    s->setVoltage(Vref + diffV);
    s->setVref(Vref);
    s->setVmax(1.05);
    s->setVmin(0.95);
    s->setP(p);
    s->setQ(q);
    s->setQtop(qtop);
    s->setQbottom(qbottom);
    s->setD(1/Sij);
    s->setAlphaVC(alphaVC);
    s->setSecondStageFlag(false);
}
```

- Over/Under Voltage check

```
void OAgent::isOverVoltage(OLocalVertex * s)
{
    if(s->getVoltage() > s->getVmax())
    {
        uint8_t ID = s->getID();
        s->setStateOver(true);

    }
    if(s->getVoltage() <= s->getVmax())
    {
        uint8_t ID = s->getID();
        s->setStateOver(false);
    }
}

void OAgent::isUnderVoltage(OLocalVertex * s)
{
    if(s->getVoltage() < s->getVmin())
    {
        uint8_t ID = s->getID();
        s->setStateUnder(true);

    }
    if(s->getVoltage() >= s->getVmin())
    {
        uint8_t ID = s->getID();
        s->setStateUnder(false);
    }
}
```

- First Stage

```
void OAgent::firstStageControl( OLocalVertex * s )
{

    Serial<<"*****************1st Stage*****************"<<endl;

    if(s->getStateOver())
    {
        Serial<<"1st stage---Over"<<endl;
        s->setRo(s->getD() * s->getAlphaVC() *(s->getVmax() - s->getVoltage()));
        s->setQtarget(s->getQ()+s->getRo());

        Serial<<"ΔQ in 1st stage is: "<<s->getRo()<<endl;
        Serial<<"Q_target for node "<<s->getID()<<" is: "<<s->getQtarget()<<endl;

        if(s->getQtarget() < s->getQbottom())
        {
            s->setSecondStageFlag(true);
        }else if(s->getQtarget() > s->getQtop())
        {
            s->setSecondStageFlag(true);
        }
    } else if(s->getStateUnder())
    {
        Serial<<"1st stage---Under"<<endl;
        s->setRo (s->getD() * s->getAlphaVC() *( s->getVmin() - s->getVoltage() ));
        s->setQtarget(s->getQ()+s->getRo());

        Serial<<"ΔQ in 1st stage is: "<<s->getRo()<<endl;
        Serial<<"Q_target for node "<<s->getID()<<" is: "<<s->getQtarget()<<endl;

        if(s->getQtarget() > s->getQtop())
        {
            s->setSecondStageFlag(true);

        } else if(s->getQtarget() < s->getQbottom())
        {
            s->setSecondStageFlag(true);
        }
    }else
    {
        s->setSecondStageFlag(false);
        s->setRo (float (0));
        s->setDeltaQ(float (0));
        s->setQtarget(s->getQ());
    }
}
```

- Broadcast Second Stage flag

```
 void OAgent::shareFlag( OLocalVertex * s, uint8_t iterations, uint16_t period)
{
    float Dout = float(s->getOutDegree() + 1);
   _initializeFairSplitting_RSL(s,0,0);
    unsigned long start;
    bool txDone;
    uint16_t txTime;
    int iter=0;
    int node_check[NUM_REMOTE_VERTICES];
    uint32_t aLsb;

    for(int i=0; i < NUM_REMOTE_VERTICES; i++)
    {
        node_check[i] = 0;
    }
    int frame = 30;

    do
    {
        srand(analogRead(0));
        txTime =  (rand() % (period - 2*frame)) + frame;
        txDone = false;
        start = millis();

        uint8_t i;
        while(uint16_t(millis()-start) < period)
        {
            if(_fairSplitPacketAvailable())
            {
                aLsb = _rx->getRemoteAddress64().getLsb();
                if(_G->isInNeighbor(aLsb,i))
                {
                    if(getSecondStageFlagfromPackage(s))
                    {
                        s->setSecondStageFlag(true);
                    }
                    uint8_t neighborID = _getNeighborIDFromPacket();
                    uint8_t nodeID = s->getID();
                    node_check[neighborID -1] = 1;
                }
            }
            if((int((millis() - start)) >= txTime) && !txDone) {
                txDone = true;
                _broadcastFairSplitPacket_RSL(s);
            }
        }
        if(!_quiet) {

            delay(10);
        } else {
            delay(25);
        }
         for(int j=0;j < NUM_REMOTE_VERTICES; j++)
```

```
        {
            if(node_check[j] == 0 && node_counter[j] >= 0)
                node_counter[j] += 1;
            else if(node_check[j] == 1 )
                node_counter[j] = 0;

            if(node_counter[j] >= int(iterations/2) )
            {
                s->setStatus(j+1, 1);
                s->decrementInDegree();
                uint8_t dout = s->getOutDegree();
                s->setOutDegree(dout - 1);
                node_counter[j] = -1;
            }
            node_check[j] = 0;
        }
        iter++;
    }while(iter < iterations);
    _buffer[1] = s->getOutDegree();
    _buffer[2] = _G->getN() - 1;
}

bool OAgent::getSecondStageFlagfromPackage(OLocalVertex * s)
{
    uint8_t ptr = 16;
    long Flag = _getUint32_tFromPacket(ptr);
    if(s->getSecondStageFlag())
    {
       return true;
    }else
    {
        if(Flag==0)
        {
            return false;
        }else
        {
            return true;
        }
    }
}
```

- Second Stage variables initialization

```
void OAgent::_initializeVariablesSecStage(OLocalVertex * s)
{
    if((s->getQtarget()) > (s->getQtop()))
    {
        s->setMuRC( s->getQtarget() - s->getQtop());
        s->setNuUpperRC(float (0));
        s->setNuLowerRC(s->getQbottom() - s->getQtarget());
    }else if ((s->getQtarget()) < (s->getQbottom()))
    {
        s->setMuRC( s->getQtarget() - s->getQbottom());
        s->setNuUpperRC(s->getQtop() - s->getQtarget());
        s->setNuLowerRC(float (0));
    }else
    {
        s->setMuRC(float(0));
        s->setNuUpperRC(s->getQtop()-s->getQtarget());
        s->setNuLowerRC(s->getQbottom()-s->getQtarget());
    }
}
```

- Second Stage

```
void OAgent::secondStageControl( OLocalVertex * s, uint8_t iterations, uint16_t period
)
{

    Serial<<"******************2nd Stage******************"<<endl;

    _initializeVariablesSecStage(s);

    s->setEtaLower(fairSplitRatioConsensus_RSL(                         s->getMuRC(),s-
>getNuLowerRC(),iterations,period ));
    s->setEtaUpper(fairSplitRatioConsensus_RSL(                         s->getMuRC(),s-
>getNuUpperRC(),iterations,period ));
    if( s->getMuRC() < 0 )
    {
        s->setEta(s->getEtaLower());

    }else if( s->getMuRC() > 0 )
    {
        s->setEta(s->getEtaUpper());

    }
}
```

Appendix B. Centralized Voltage Control Code

- First Stage

```
for(i=0; i<3; i++)
{
    if(V[i] > Vmax)
    {
        ro[i] = alphaVC/Sij[i]*(Vmax - V[i]);

        if((Q[i]+ ro[i]) > qtop)
        {
            Q_secondary=Q_secondary + ( Q[i]+ ro[i] - qtop );
            ro[i] = ro[i] - Q_secondary;
        }

    }else if(V[i] < Vmin)
    {
        ro[i] = alphaVC/Sij[i]*(Vmin - V[i]);

        if((Q[i]+ ro[i]) < qbottom)
        {
            Q_secondary=Q_secondary + ( Q[i]+ ro[i] - qbottom );
            ro[i] = ro[i] - Q_secondary;
        }
    }else
    {
        ro[i] = 0;
    }
}
```

- Reserve of reactive power computation

```
for(i=0;i<3;i++)
{
    QreserveRise[i]= qtop - (Q[i] + ro[i]);
    QreserveLower[i]= qbottom  - (Q[i] + ro[i]);
    QreserveUp = QreserveUp + QreserveRise[i];
    QreserveDown = QreserveDown + QreserveLower[i];
}
```

- Second Stage

```
if (Q_secondary != 0){
    if(Q_secondary > 0){
        for(i=0;i<3;i++){
            deltaQsec[i] = Q_secondary * (QreserveRise[i]/QreserveUp);
        }
    }else{
        for(i=0;i<3;i++){
            deltaQsec[i] = Q_secondary * (QreserveLower[i]/QreserveDown);
        }
    }
}
```

- New reactive power level

```
for(i=0;i<3;i++)
{
    Qnew[i] = Q[i] + ro[i] + deltaQsec[i];
    if(Qnew[i] > qtop)
    {
      Qnew[i]=qtop;
    }else if(Qnew[i] < qbottom)
    {
      Qnew[i]=qbottom;
    }
}
```

Appendix C. Ratio Consensus

Algorithm

For the implementation of the Second Stage in the Voltage Control algorithm there needs to be collaboration in between the neighboring nodes. This comes as a consequence that the target node does not have enough reactive power available so as to manage the fault by itself. For the system to go back to normal, the remaining reactive power needed is provided by the remaining neighbors.

the partitions of the reactive power remaining have to be chosen regarding the reactive power levels and limits. This process can be executed following a centralized approach. This process is based on the computing of the average of the raising or lowering potential of each DER. However, we want to develop a distributed algorithm. This algorithm is the fair split ratio consensus.

In the algorithm, the desired output for each operation is going to be the ratio in between the state variables μj[k] and νj[k], which will be denoted by γj[k]=(yj[k] / zj[k]). After each iterations, the state variables need to be updated as follows from [11]:

$$y_j[k+1] = \sum_{i \in \{\mathcal{N}_i^-\} \cup \{i\}} \frac{1}{1+\mathcal{D}_j^+} y_j[k]$$

$$z_j[k+1] = \sum_{i \in \{\mathcal{N}_i^-\} \cup \{i\}} \frac{1}{1+\mathcal{D}_j^+} z_j[k]$$

This algorithm shows that the value at each iteration is going to be updated according the value of μj[k] and νj[k] in iteration k from the neighboring nodes, denoted by $\mathcal{N}_j^-$, divided by the weight of the node, which is the outdegree of node i $\mathcal{D}_i^+$. The outdegree is the number of nodes that are part of the set of neighbors of node i.

```
if(_G->isInNeighbor(aLsb,i)){
        float inMu = _getMuFromPacket();
        float inSigma = _getSigmaFromPacket();
        uint8_t neighborID = _getNeighborIDFromPacket();
        uint8_t nodeID = s->getID();

        float Mudiff = inMu - s->getNuMin(i);
        float sigdiff =  inSigma - s->getTau(i);

        inY += Mudiff;
        s->setNuMin(i,inMu);
        inZ += sigdiff;
        s->setTau(i,inSigma);

        node_check[neighborID -1] = 1;
}
       s->setYMin((s->getYMin()/Dout) + inY);
       s->setMuMin(s->getMuMin() + (s->getYMin()/Dout));
       s->setZ((s->getZ()/Dout) + inZ);
       s->addToSigma(s->getZ()/Dout);
```

The values of γj[k] from each node will converge to a common value, since as we mentioned, we are computing the average of the values. The value of γj[k] at the last iteration will asymptotically approach to:

$$\gamma = \lim_{k\to\infty} \gamma_i[\mathrm{k}] = \frac{\sum_{j=1}^{n} y_i[0]}{\sum_{j=1}^{n} \mathrm{z}_i[0]}, \qquad i \in \mathcal{V}_c$$

By computing this algorithm paired to a correct initialization of the state variables, we will obtain enough information about the system so that we can compute the weighted distribution of reactive power addition or consumption needed so as to supply the required amount that will make the network go back to its operating conditions.

In the ideal case for the algorithm, the number of iterations would be infinite, but in a real simulation, the number of iterations that the algorithm runs are set by various criteria. The criterion that will be used in this project is the Max-min Consensus.

Code

The structure of the code is as follows. First we have to define which of the nodes is going to be the leader. The leader node works the same way as the non-leader nodes. The only difference is that the leader node will be set as time Reference for the collaborative algorithm. Once the leader is set and the nodes have synced, the ratio consensus algorithm starts. The first stage of the code is the initialization of the nodes, which is done in the function “fairSplitRatioConsensus_RSL”. Once initialized, the algorithm will differentiate in between the leader node and the non-leaders, calling different functions (leaderFairSplitRatioConsensus_RSL or nonleaderFairSplitRatioConsensus_RSL). In the second stage, the nodes will start by sharing their time through the XBee protocol. The new time for all of the Arduinos will be the one that the leader has in order to sync all of the nodes. Once the nodes are synced, they will enter the body of the algorithm, that is the function “ratiomaxminConsensus”. Once in the ratio maxmin consensus, the Arduino will develop the algorithm describe previously. The result (ɣ) will be sent to the Arduino and the ratio consensus will be finished.

Figures, Tables and Graphs

Figures

Tables

Graphs

References

[1] "Detail @ Www.Eia.Gov." [Online]. Available: http://www.eia.gov/todayinenergy/detail.cfm?id=25372.

[2] N. Carolina, "2018 Average Monthly Bill- Residential 2018 Average Monthly Bill- Residential," pp. 101–102, 2018.

[3] S. M. Amin, "U.S. Electrical Grid Gets Less Reliable," *IEEE Spectr.*, 2011.

[4] B. A. Robbins, C. N. Hadjicostis, and A. D. Domínguez-García, "A two-stage distributed architecture for voltage control in power distribution systems," *IEEE Trans. Power Syst.*, vol. 28, no. 2, pp. 1470–1482, 2013.

[5] B. V. Ermolenko, G. V. Ermolenko, Y. A. Fetisova, and L. N. Proskuryakova, "Wind and solar PV technical potentials: Measurement methodology and assessments for Russia," *Energy*, vol. 137, no. April, pp. 1001–1012, 2017.

[6] Y. Mou, H. Xing, Z. Lin, and M. Fu, "Decentralized optimal demand-side management for PHEV charging in a smart grid," *IEEE Trans. Smart Grid*, vol. 6, no. 2, pp. 726–736, 2015.

[7] KSFR, "Retake Our Democracy Host Paul Gibson Weighs in on the Democratization of Energy | KSFR." [Online]. Available: https://www.ksfr.org/post/retake-our-democracy-host-paul-gibson-weighs-democratization-energy. [Accessed: 17-Apr-2020].

[8] R. H. Lasseter, "MicroGrids: A Conceptual Solution," *2002 IEEE Power Eng. Soc. Winter Meet. Conf. Proc. (Cat. No.02CH37309)*, vol. 1, pp. 305–308, 2002.

[9] O. Azofeifa *et al.*, "Controller Hardware-in-the-Loop Testbed for Distributed Coordination and Control Architectures," 2020.

[10] "Virtual HIL Device - Typhoon HIL." [Online]. Available: https://www.typhoon-hil.com/products/virtual-hil-device/. [Accessed: 08-Apr-2020].

[11] S. T. Cady, S. Member, A. D. Domínguez-garcía, C. N. Hadjicostis, and S. Member, "A Distributed Frequency Regulation Architecture for Islanded Inertialess AC Microgrids," vol. 23, no. 5, pp. 1717–1735, 2015.