IMPLEMENTATION OF A NOVEL COOPERATIVE PROTOCOL FOR DISTRIBUTED VOLTAGE CONTROL IN ACTIVE DISTRIBUTION NETWORKS

RABBIA QAMAR

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE OF COMPUTER ENGINEERING

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

MAY 2018

Abstract

Microgrids are small localized grids that help to integrate many renewable-energy sources into the main electric grid. Microgrids can also operate separately from the main electric grid during faults to enhance the customers reliability. For a successful integration of microgrids we need to control the voltage at the distributed generation units in order to achieve the required sharing of reactive power. For this purpose a multiagent based distributed control scheme is implemented in this thesis. The objective of this thesis is to design and implement a multiagent system for the microgrid that has distributed battery energy storage systems (BESS) and renewable distributed generation (DG) units. The proposed multiagent system has been designed to coordinate among distributed generation (DG) units to control voltage. Multiagent system is composed of multiple agents that communicate to solve problems. The proposed multiagent system for the control of microgrid has been implemented on Texas Instruments Tiva-C controller boards. The real time simulator Opal-RT has been used to create a microgrid model. Hardware testing is done in real time.

Acknowledgements

I would first like to thank ALLAH ALMIGHTY who showered His endless blessings on me to carry out my work in an effective manner. I would like to thank my supervisor Prof. Mokhtar Aboelaze for his thought-provoking suggestions throughout the entire course of this research. His patience, trust and encouragement have helped to keep my enthusiasm high through some difficult times. I have been lucky to have a supervisor who is always there to answer my questions. I would also like to thank Prof. Hany Farag for his absolute support to this research. His enthusiasm and integral view on research and his mission for providing only high-quality work, have made a deep impression on me. I would like to express my gratitude from my heart to my parents whose love and prayers are always with me during the whole period of this master. My special thanks goes to my husband for such an invaluable support to complete my work. Finally I would like to thanks my daughters whose smile and love always refresh and energize me throughout my research work.

Table of Contents

List of Tables

List of Figures

Abbreviations

DG	Distributed Generation
BESS	Battery Energy Storage System
RE	Renewable Energy
VSC	oltage Source Converter
PV	Photovoltaic
CESS	Composite Energy Storage System
FC	Fuel Cell
CB	Circuit Breaker
DSP	Digital Signal Processing
SIMD	Single Instruction Multiple Data
NVIC	Nested Interrupt Controller
NMI	Non-maskable Interrupt
ISR	Interrupt Service Routine
MPU	Memory Protection Unit

RTC	Real-time Clock
ADC	Analog to Digital Converter
UART	Universal Asynchronous Receiver Transmitter
SSI	Synchronous Serial Interface
IC	Inter Integrated Circuit
CAN	Controller Area Network
FIFO	First In First Out
DMA	Direct Memory Access
FPGA	Field Programmable Gate Array
SoC	State Of Charge
BSA	Battery Of Storage Agent
HIL	Hardware in Loop

Chapter 1

Introduction

The whole world is promoting the renewable energy (RE) sources to meet the energy demand. The renewable energy sources do not pollute the environment and also do not drain with continuous usage [1]-[7]. Renewable energy sources e.g wind, solar and biofuels are used in distribution networks to supply secure energy with new challenges of operation and planning [8]-[10]. The USA is generating 22.9 percent of its electrical energy by using renewable energy sources [11]. Rapid growth in the use of renewable energy sources has the benefit of reducing greenhouse gas emissions due to conventional energy generation. Due to the above mention properties renewable energy sources are considered to be good alternative to conventional sources for power generation. [12],[13].

The electromechanical generators have been used for the generation of electricity for decades. The engines of these electromechanical generators fueled by chemical combustion or nuclear fission. Conventional power plants powered by fossil fuel have an overall

efficiency from source to user of 30 percent due to low efficiency of thermodynamic cycles , transmission and distribution losses while renewable energy sources will have an estimated better efficiency of 70 percent [14]. According to Environmental Investigation Agency (EIA) [14] International Energy Statistics 2010 data shows that 63 percent of the electricity generated in the United States comes from fossil fuel combustion whereas more than 70 percent electricity in china comes from fossil fuel [14]. Conventional power grid has centralized sources of power generation. Flow of energy is unidirectional from the sources to the customers. Real-time monitoring and control is used in generation and transmission system only sometimes it is also used in the distribution system. Conventional systems are not flexible, it is hard to produce electricity by another sources or to provide new services based on the demands of the users. The conventional power systems have fulfilled the electricity demands in the past. However, there is a need of electric energy system that is combined with advanced, intelligent technologies and renewable energy sources to drive the global economy [15]. To get the energy system with above mentioned properties a modernized and upgraded electricity grid called smart grid has been introduced in market. Smart grid uses advanced information and communication technologies to optimize its operation. We are going to discuss smart grid in next section.

1.1 Smart Grid

The smart grid is a safe, secure, reliable, resilient, efficient, and sustainable electric system for electricity generation, transmission, distribution, and consumption [16],[17].

- Smart grid allows the integration of renewable energy resources to address global climate change.
- Smart grid provides better energy conservation through active participation of customers.
- Smart grid enables two-way secure communication between generation,transmission and distribution units.
- smart grid reduces transmission losses and cost of electricity by optimized energy flow.
- Smart grid can work with existing grids to improve their utilization.
- Smart grid motivates integration of electric vehicles that reduce usage of hydrocarbon fuels.

Distributed generation and energy storage in smart grids eliminate system expansion and lowers the cost of energy. Smart grid introduces communication and control to the whole energy system to promote safety and operational flexibility [18]. Cost management is the key component before implementation of the smart grid then testing and verification before placement. New functionality of smart grid can be implemented on existing infrastructure of electric system. It is not feasible to implement all the features of smart grid together because cost justification for the investors is a major concern. The property of smart grid to repair or stop faulty equipment from service and reroute energy supply to maintain

power to the consumers is known as self-healing. Distributed generation and energy storage system can be interconnected at any time. Machine learning technology can be used in the smart grid to predict the weather impact and reconfigure the system according to the situation [15]. It is an interactive system that provide useful parameters of the energy system to the operators and customers to maintain optimal management. Optimization of electricity flow is possible due to real time readings of main components and control equipment used to reroute energy paths. Smart grid provides cyber-security and end to end two-way communication through the system. Smart grid has changed the conventional concept of energy management and operations of energy system. Smart grid uses intelligent equipment at generation, transmission, distribution, and consumption level. To make the operation of these components effective smart grid is integrated with control technologies, data management, diagnostic analysis, and work management [15].

1.2 Why Smart Grid?

The challenging issues of conventional power system that motivate developing and implementing the smart grid are highlighted by the electric power industry stakeholders. Infrastructure of existing power systems is dated back to the 1950 or even earlier. Due to lack of investment and increasing load on power grids due to increase in electricity demand throughout the world these infrastructures need lot of maintenance work. Smart grid can reduce CO_2 emission by adopting renewable energy sources . Increasing distance between generation and load sites is causing transmission losses. Smart grid can also provide local

generation system. The central generation system working in parallel with lots of decentralized and distributed generation systems in smart grid. Renewable energy sources like wind and solar provide infrequent energy that cannot be handled by conventional power grids. Smart grid is needed to balance the intermittent behavior of renewable energy sources. Electric vehicles, smart homes and smart buildings are new electricity consumption models. Regulatory pressures and electricity cost is increasing. Trade of energy is increasing due to split charges of utilities. Clear information of consumption and pricing of electricity should be available for consumers. The major concern of the regulatory authorities is less electricity cost. To meet the increase in energy demands secure supply is needed. Information and communication technologies should be used to handle operational challenges. Investment in power grid infrastructure is important to maintain efficient and reliable transmission and distribution [19],[20].

1.3 Microgrid

The microgrid consists of number(may be small number) of loads and microsources that operate as a single controllable system and provides power to local consumers. For the power system, microgrid is a controllable cell that can be controlled as single dispatchable load that gives fast response to the transmission system. Microgrid can be designed according to the needs of the customers. Microgrids enhance reliability, reduce feeder losses, support local voltages, increase efficiency by using waste heat, provide power supply without suspension [21]. The microgrid consists of generators, consumers, energy storage devices and

hybrid devices like electric vehicles that can produce and consume energy. Microgrid is a cluster of distributed generators (DGs) that is interfaced with the main power grid through voltage source converter (VSC) [22]. Distributed generation (DG) technologies in microgrid are getting more importance and has become major part of the microgrid. Production of energy by using DG technologies has very little impact on environment. Placement of these technologies are easier and their efficiency is higher [23]. Implementation of microgrid systems has benefits for both the user and the electricity provider. The microgrid while connected to the grid can improve network quality, reduce emissions, and can reduce the electricity cost for the users. Implementation of distributed generation systems of microgrids can reduce the power flow on transmission and distribution lines of electric utility provider that reduce losses and costs for additional power. In case of errors, microgrids help to repair the network, bypass the dead-end to meet the electricity needs and reduce network load [24]. Many countries has developed microgrids due to its advantages like good power quality, more environmentally friendly and better economic potential because it can utilize the waste heat from the engine generator [25]. Microgrid also has renewable energy sources as an alternative generation system that requires operating mechanism and sophisticated control system to make it reliable and efficient system [25]. DGs work as the main power supply in microgrid. The dynamic behavior of microgrids is due to the dynamic characteristics of DGs. There are two ways to connect the DGs with microgrid either through inverter interface or directly. PV, small direct-drive wind turbines, micro gas turbines, batteries, flywheel energy storage and super capacitor are inverter interface

DGs that can be connected to the microgrid through inverters. While double-fed induction generator, diesel generator, and small hydro units can be connected directly to the microgrid. Renewable energy sources are widely used in the building of microgrid. Usage of inverter interfaced DGs make the operating characteristics of microgrid quite different from the traditional grid.

1.3.1 Operating Characteristics of Microgrid

There are two operating modes of microgrid grid-connected mode and islanded mode. Grid-connected mode is when microgrid is connected to the main power grid. In grid-connected mode the power flow of microgrid is bidirectional. In islanded mode microgrid works standalone and meet the demand of power supply load [26]. Microgrid structure is diversified by different DGs. Characteristics of microgrid stability problems depends on varieties of operation modes. Different control strategies for DGs has different effect on voltage and frequency regulation of DGs in the microgrid. The output impedance and over-current capability of inverter interfaced DGs are very small as compared to synchronous generators. The protection response time of these DGs are much faster than synchronous generators. Especially for the transient stability analysis of microgrid protection response time is the most important characteristic of inverter interfaced DGs. In case of faults DGs manage to fulfill the energy demands. The stability is maintained through energy storage DGs [27]. The inverter interface DGs have fast response time than traditional DGs [28].

The control system of a microgrid is designed to safely operate the system in grid-

connected mode and islanded mode. This system may be based on a central controller or embedded as autonomous parts of each distributed generator. When the utility is disconnected, the control system must control the local voltage and frequency, provide (or absorb) the instantaneous real power difference between generation and loads and provide the difference between generated reactive power and the actual reactive power consumed by the load and protect the internal microgrid [29].

A multi-agent control scheme in order to control the voltage at distributed generation units is proposed in this thesis. The purpose of this distributed control scheme is to achieve the required reactive power sharing in distribution networks. We used Tiva-C microcontroller boards for the implementation, and testing has been done in real time on hardware.

1.4 Thesis Organization

The remaining of the thesis is organized into five chapters. First, a literature review of the multiagent system is presented in Chapter 2. In Chapter 3, we illustrate the implentation platform for our work the Tiva-C series microcontrollers and OPAL-RT technologies. In Chapter 4, we demonstrate the implementation of distributed control system for islanded microgrid and communication between distributed generation agent by using Tiva-C microcontroller boards. Conclusion and the future work is described in Chapter 5.

Chapter 2

Multiagent System

For the past years researchers are taking more interest in multiagent systems (MASs). Applications like air traffic control, process control, manufacturing, electronic commerce, patient monitoring, or games are examples of those multiagent system. The understanding, design, and implementation of complex distributed and concurrent software are the main reason for the popularity of multiagent systems. The advancement in internet computing has brought more attention towards multiagent technology. It provides infrastructure in which autonomous agents interact with each other [30]. Agent-based computation is considered a new paradigm of information and communication technology [31]-[34].

2.1 What is an Agent?

An agent is a computational system who act autonomously and intelligently according to the environment in which it is situated [31]. An agent may have other agents in his

environment. At first, an agent observes its environment then according to its knowledge and the state of its environment it executes action. Agents have partial knowledge of their environments. Autonomous agents are the computational agents that have some kind of control over their actions. They can make decisions without intervention of any other agent. Rational agents operate according to the state of the environment and do the best for themselves. The computational complexity and limited computing resources can bound the rationality of the agent. An agent has some important properties. An agent is designed by the user to fulfill specific purpose. They have control over their internal state and behavior. Fixed rules of reasoning, planning, and learning capability make an agent intelligent. Agents can adjust their behavior according to any change in environment without interference of designer. Mobility is the property of an agent for special applications. An agent can transport to access remote resources or to interact with other agents in another environment[31].

2.2 What is Multiagent System?

A network of problem-solving entities that work together to find answers to problems without knowing each other capabilities is called a multiagent system. Multiagent systems are incorporated with multiple agents and mechanism for their coordination. Multiagent systems can be classified according to cooperation types as shown in Figure 2.1. Three ways of cooperation within a multiagent system can be invoked.

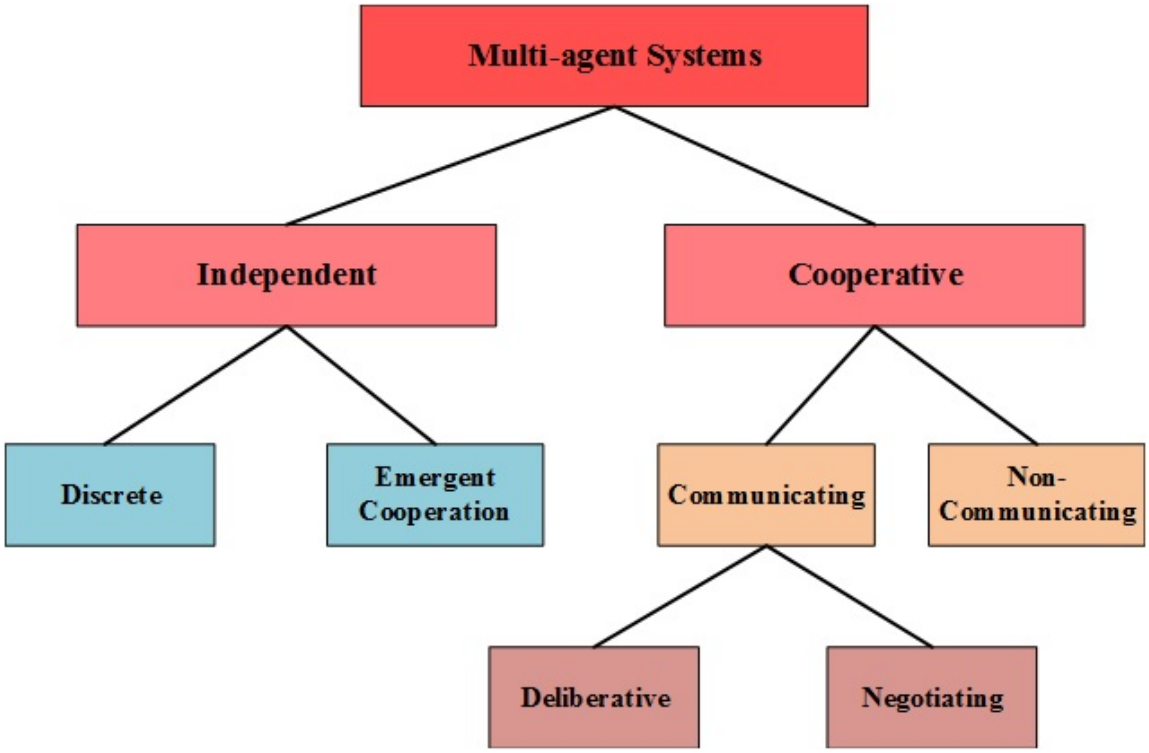


Figure 2.1: Multiagent System Cooperation types

- Agents are designed to cooperate with other agent.
- Each agent can adapt the cooperation property.
- By evolutionary process cooperation can be evolved among agents.

The classification of multiagent systems can be explained through types of cooperation. When each agent of the system works independently and fulfill its goals without any coordination, this type of system is called independent multiagent system. A discrete multiagent system comes under independent multiagent system that consists of agents who have assigned different goals with no relation to one another. There is no cooperation in discrete multiagent system. Agents of independent multiagent system with emergent cooperation pursue their goals without knowing the existence of other agents in the same environment. They can affect each other indirectly because they all are part of the same environment. The individual agent does not explicitly cooperate but cooperation emerges due the environmental effects. The agent can receive sensory inputs of one or more other agents in the same environment. The agent can change the sensory inputs of another agent unintentionally. Cooperative multiagent system is further categorized according to agents homogeneity and heterogeneity. Also depends on the communication among them. There are four categories of cooperative multiagent system. Homogeneous non-communicating multiagent system, Homogeneous communicating multiagent system, Heterogeneous non-communicating multiagent system and Heterogeneous non-communicating multiagent system;

- Homogeneous non-communicating multiagent system consist of identical structured

agents. All agents have the same internal structure, goals, actions, and domain knowledge. All agents select their actions by following the same procedure. Each agent has its individual sensory inputs and list of actions. They take decision about their actions independently and produce different outputs. Having different inputs and actions for every individual agent are the necessary conditions for homogeneous non-communicating multiagent system.

- Heterogeneous non-communicating multiagent system composed of heterogeneous agents. These agents have different goals, domain knowledge and actions. In designing heterogeneous non-communicating multiagent system one of the most important issue to consider is whether the different agents will be benevolent or competitive. Despite having their different goals if agents are willing to help each other to achieve their goals it means that agents are benevolent. If they are not willing to help each other then they are competitive. Figure 2.2 describes the non-communicating multiagent system.

- In homogeneous communicating multiagent system homogeneous agents can communicate directly with each other as we can see in Figure 2.3. But communication raises some issues in multiagent system. Like if agents can only sense each other presence or they can share information. Based upon their observation in the multiagent system agents can learn what and when to communicate with each other.

- In the heterogeneous communicating multiagent system the agents with different

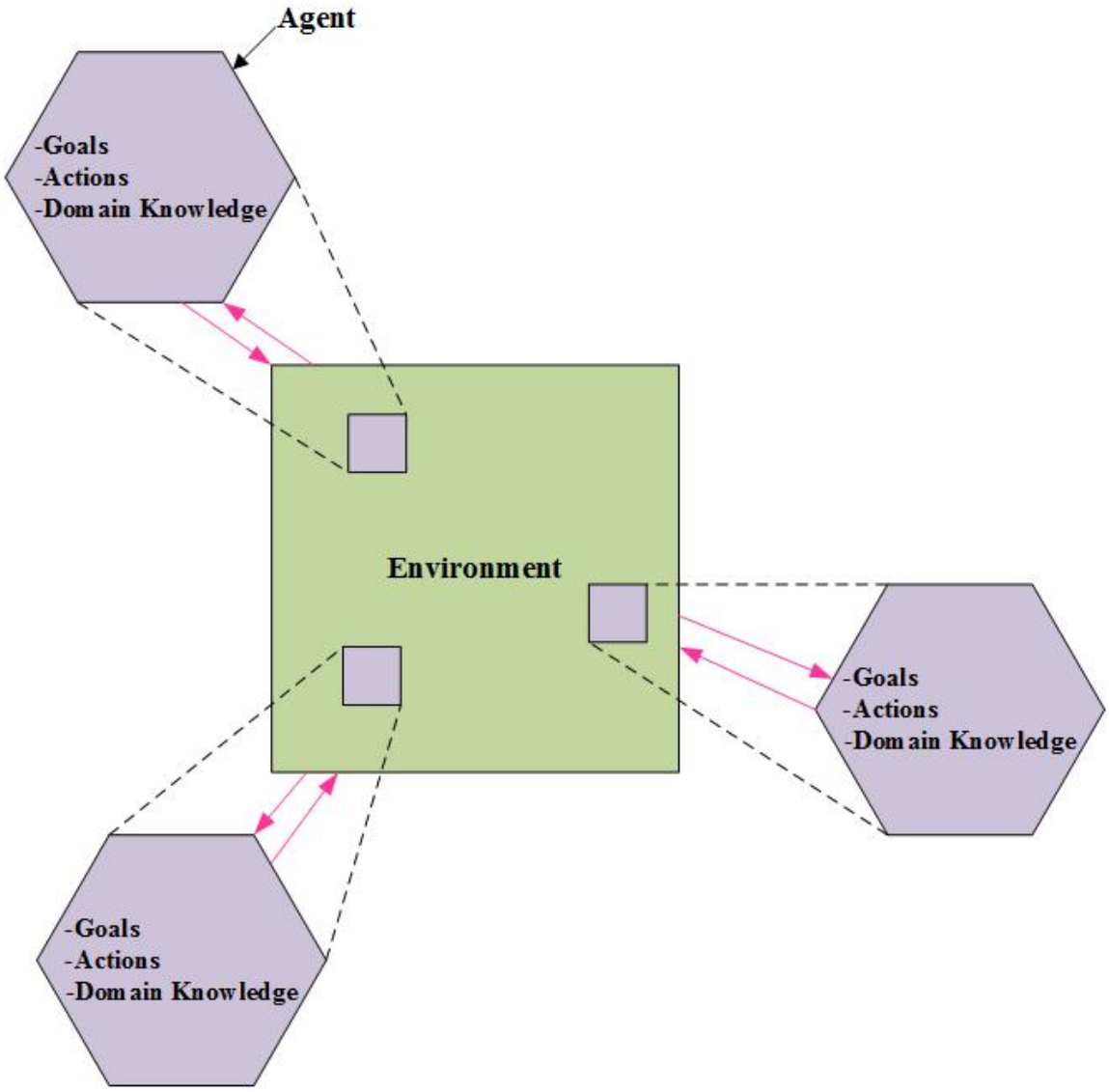


Figure 2.2: Non-Communicating Multiagent System

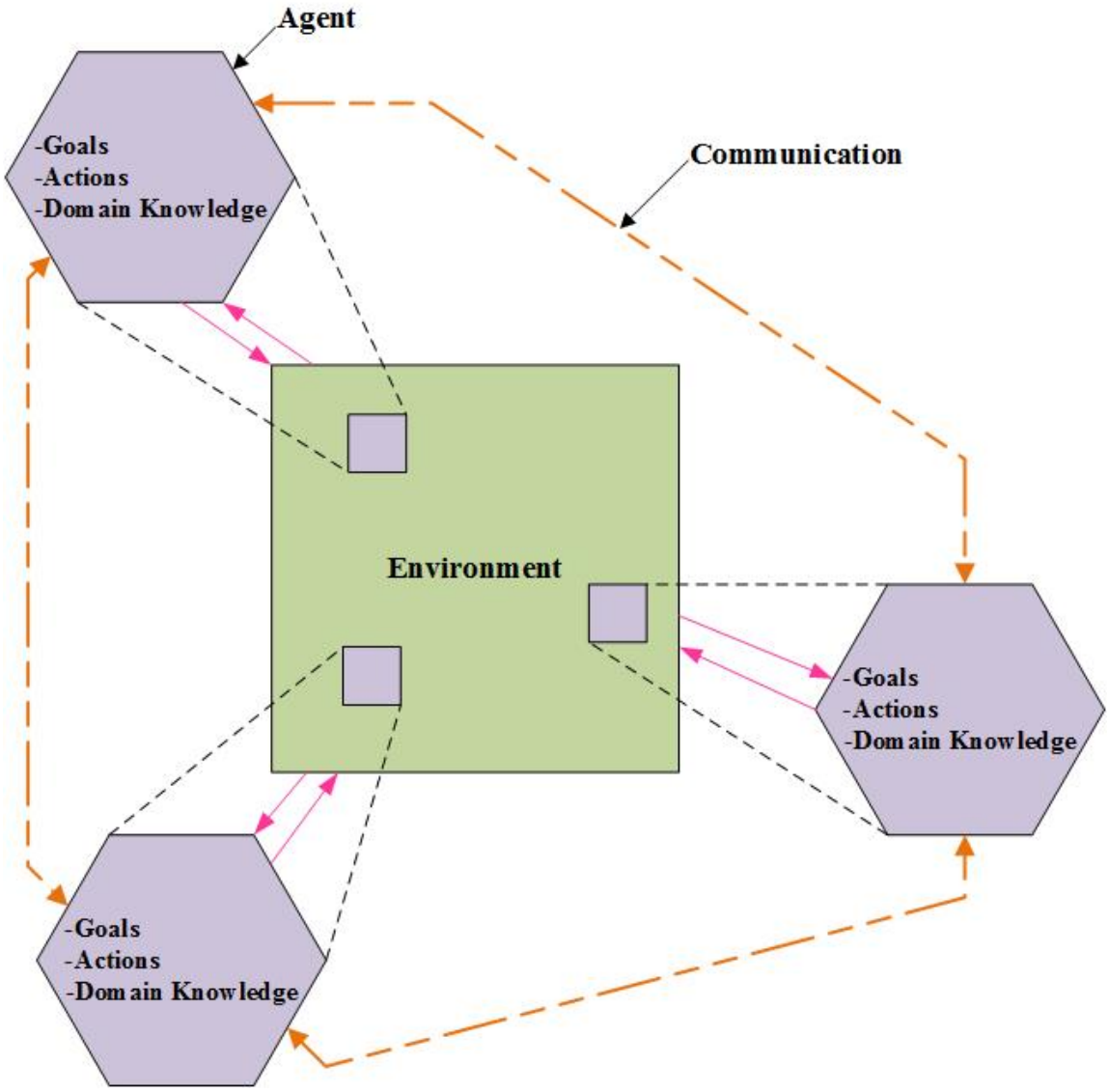


Figure 2.3: Communicating Multiagent System

goals, actions and domain knowledge can communicate directly with each other. Communication issues are same as we discussed for homogeneous communicating multiagent system, but heterogeneity make communication more challenging. Communication protocols and theories of commitment are major concern for this type of multiagent system. Communication becomes more complicated due to benevolent or competitive behavior of agents.

Agents can communicate with each other to complete assigned task and cooperate for assigned duration of time. If they agree to do so they make commitment to each other. It can be internal commitment when an agent specified itself to perform some task. It can be social commitment when an agent makes commitment to other agent or it can be collective commitment when agent agree to fill certain role. To pursue a given goal an agent commit to another agent and perform accordingly regardless of its self-interest [35],[36].

2.3 Applications of Multiagent System

For real-time control and management of microgrid the multiagent system was proposed in [37]. The microgrid was created in real-time digital simulator (RTDS) [38] by using electrical components of model libraries. Schematic diagram of the microgrid is shown in Figure 2.4 [37]. The multiagent system was developed in Java Agent Development Framework (JADE) [39]. JADE is a Java based software framework Foundation for Intelligent Physical Agents (FIPA) [40] compliant open source platform. It is used for the development of

intelligent agent and developed by Telecom Italia [39]. Interface between JADE and RTDS was developed by using TCP/IP protocol. The proposed multiagent system is composed of distributed generator agents (DG Agent), load agents (Load Agent), a renewable energy source agent (RES Agent), a storage system agent (Storage Agent), a microgrid manager agent (MGM Agent), a schedule coordinator agent (SC Agent), a demand side management agent (DSM Agent). Administrative agents were included in this multiagent system. Distributed database to store data and information of all agents was also included in this multiagent system.

This intelligent multiagent system was proposed for management of generation scheduling functions and demand side management of microgrid. Two-stage generation scheduling system is composed of day-ahead and real-time scheduling was used for this work. The day-ahead scheduling finds out hourly power settings of distributed energy resources (DERs) from an energy market. Day-ahead scheduling is done for short term generation scheduling. Generator agents get the power scheduling information and update the generators. The real-time scheduling updates the power settings of the distributed energy resources according to the results of the day-ahead scheduling and feedback from real-time operation of the microgrid in real-time digital simulator (RTDS). Load shifting is done by a demand side management agent before the day-ahead scheduling. Whenever it is required, load curtailing in real-time is performed by a demand side management agent. The distributed multiagent system provides a common communication interface for all agents of microgrid for the autonomous control actions and decision making. The multiagent system used

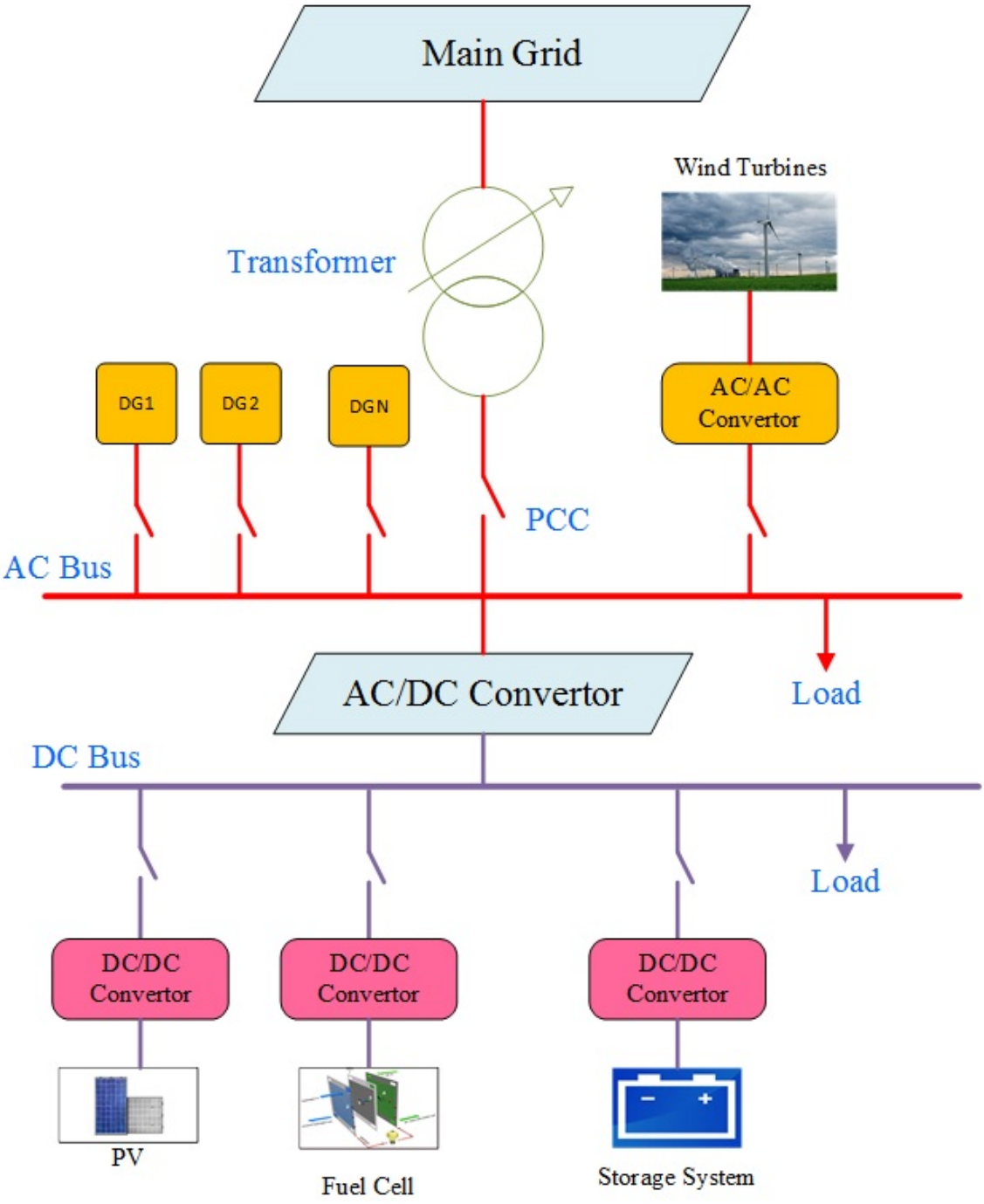


Figure 2.4: Schematic diagram of microgrid

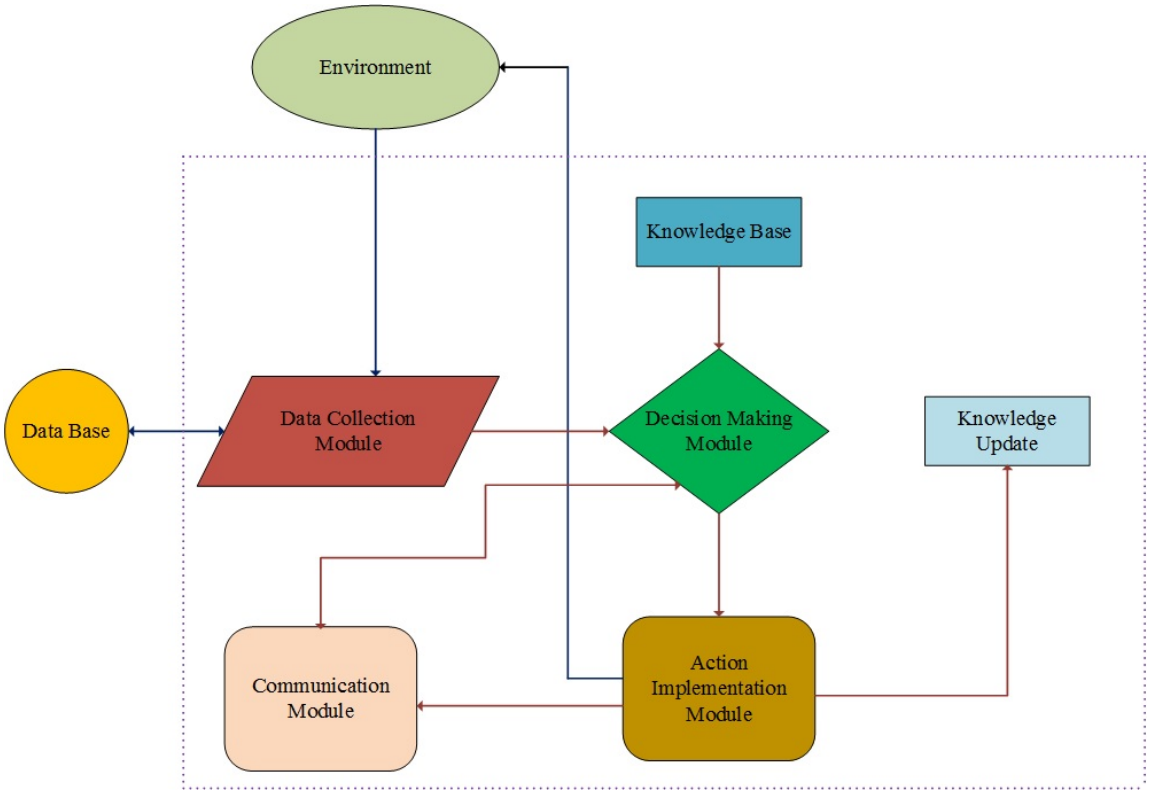


Figure 2.5: Architecture of an intelligent agent

the real-time digital simulator for the real-time operation without violating any technical constraints during the microgrid operation. Computational intelligence techniques and mathematical tools were used for the implementation of the decision making modules of the intelligent agents. The architecture of an intelligent agent in this multigent system is shown in Figure 2.5 [37]. The proposed multiagent system maximizes the power production of local distributed generators, minimizes the operational cost of the microgrid, and optimizes the power exchange between the main power grid and the microgrid subject to

system constraints and constraints of distributed energy resources. The two stage scheduling is a useful tool for management of a microgrid. The simulation studies have shown that the operational strategy is able to tackle both economical and technical objectives of the operation [37].

Another control system for microgrid was proposed in [41]. Distributed and cooperative control architecture was developed by using the multiagent system technology to control the distributed resources within a microgrid. The author used Matlab to create a hybrid microgrid model. Schematic configuration of this microgrid model is shown in Figure 2.6 [41]. The modeled microgrid is composed of photovoltaics (PV) system, fuel cell (FC), Composite Energy Storage System(CESS) and three distributed generators operating with diesel, biodiesel, and natural gas. Each distributed energy resource of this microgrid was connected to its own microcontroller. Foundation for intelligent physical agent (FIPA) complaint multiagent platform JADE was used for the implementation of the proposed multiagent system. The proposed multiagent system consists of photovoltaic (PV) agent, Fuel cell (FC) agent, composite energy storage system(CESS) agent, power converter building block (PCBB) agent, distributed generators (DG) agents, load agents, and circuit breakers (CB) agent. Each agent in the multiagent system can monitor and control the corresponding components also can communicate with other agents. There is no central control agent. Each source or load makes decisions locally and provide a distributed, scalable, and robust control for the microgrid. This intelligent control system built with multiagent technique maximizes the power production output of local distributes generators and optimizes power

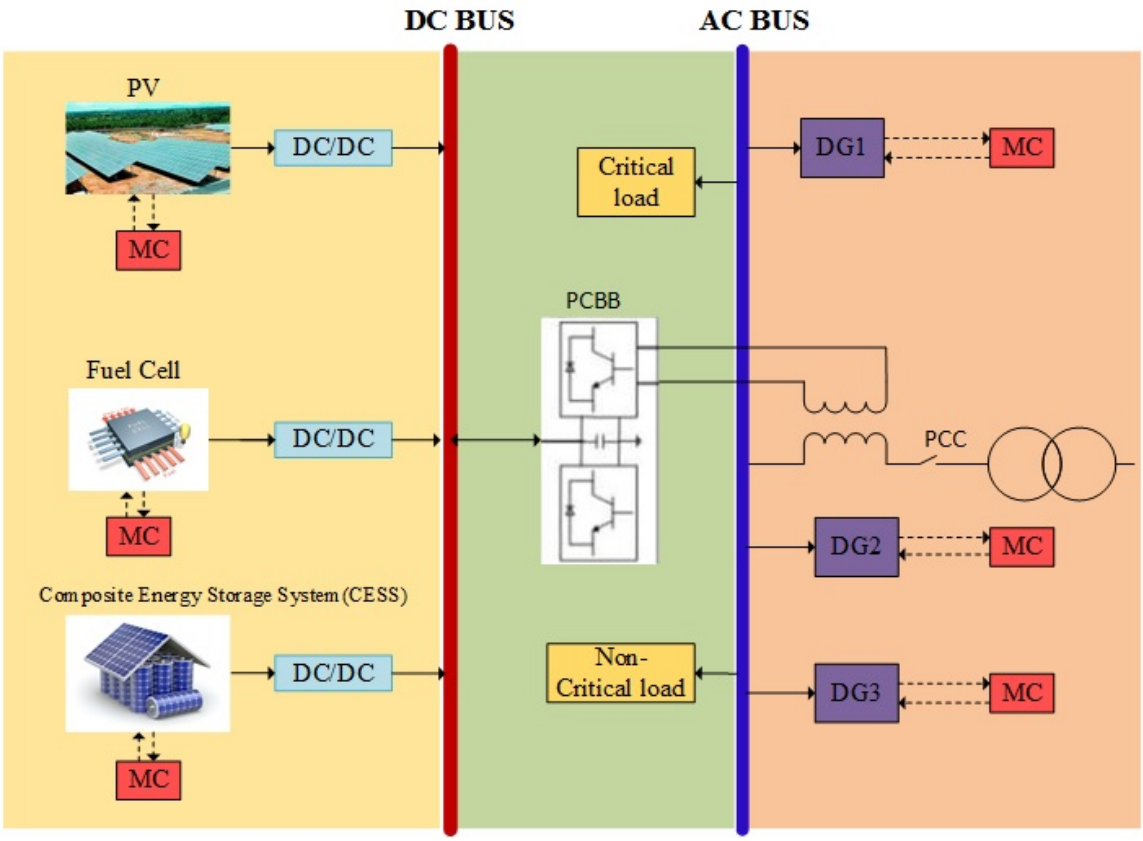


Figure 2.6: Configuration of hybrid microgrid

exchanges between the microgrid and the main power grid.The results of case studies of a hybrid microgrid show that proposed multiagent system based control architecture can provide successful performance for the real-time control of microgrids [41].

To provide remote real time monitoring for premature babies in hospital, the multi-agent monitoring system monitors vital signals of premature babies in an incubator was proposed in [42]. This multiagent system was integrated with distributed agents like data capture agent, data publisher agent, and a notification agent. For the proper growth of the newborns in their very early age, it is important to keep them safe from airborne infectious diseases and provide them with appropriate environmental conditions. The main parameters for the safety and the normal growth of the infant in a neonatal incubator are humidity, the baby's heart rate, and the baby's temperature. The proposed multiagent based NeonatalIincubator Monitoring System (NIMS) dynamically collects the information from incubators sensors and make it accessible for the medical staff and the babies parents through a mobile phone application. In case of abnormal values and the need of urgent interaction notification is also received through mobile phone application. The proposed neonatal incubator monitoring system (NIMS) consists of a humidity sensor, a body temperature sensor and a heart rate sensor connected to a data hub device. The sensed data is preprocessed by the data hub agent. The data hub is connected to the local medical server of the hospital through Ethernet to minimize the effect of wireless signals around the premature baby. Incoming data can be continuously listen through local router that is externally powered. The medical server is a secure server inside the hospital provided with

a distributed software component. The medical server, works as the publisher agent, will receive the sensed data from the Arduino data hub, using user datagram protocol (UDP) socket programming and Constraint Application Protocol (CoAP) protocol [43]. Who publishes the data into the hospital information system and makes this data available over the Internet by using the CoAP protocol. The server also notifies the notification agent according to sensed data. The Rest architecture was used for real monitoring application because it is a stateless protocol. The CoAP uses the Rest technique and access resources through URI. The server is responsible of creating the resources while the client can GET, PUT, POST or DELETE a resource. The CoAP is faster and requires less resource usage at network side as well as at the node (server/client) side [43]. CoAP enables IP multicasting and provides retransmission mechanisms that compensate UDP unreliability [44]. That is why CoAP is the most adopted protocol for the Internet of Things (IoT) devices. The CoAP protocol for data publishing was used in the proposed distributed neonatal incubator monitoring system (NIMS) because it reduces mobile phone battery consumption and provided real time monitoring with the minimum delay. CoAP can transfer any types of messages from simple value to embedded videos. It permits to integrate any sensor in the network like web camera, audio recording, localization, etc. The medical server agent was implemented in Java. For the CoAP library , jcoap [45] was used. A data server listening and receiving UDP packets from Arduino data hub and generating alarm events for notification agent were done by CoAP. Data publisher agent create and manage resources listen from CoAP server. The proposed system implemented and tested in a hospital in Lebanon

[42].

2.4 Chapter Summary

This chapter describes multiagent system and their types. Multiagent system consists of homogeneous or heterogeneous agents. Cooperation between agents classifies multiagent system in different types. Multiagent technology is useful to decentralize the control system of power grid. Previous work about the multiagent system is discussed in this chapter. The next chapter discusses the implementation platform used in this thesis.

Chapter 3

Implementation Platform

3.1 Tiva-C Microcontroller

We will use Texas Instruments Tiva-C Series microcontroller TM4C123GH6PM for implementation of proposed multiagent system in this thesis. Tiva-C Series microcontrollers are the most widely used in the market for embedded system applications. Tiva-C series microcontroller used for the implementation of our multiagent system is based on ARM Cortex-M processor architecture. ARM Cortex-M processor architecture has become popular very quickly in industry. Cortex-M based products provide highly connected, low cost and easy to use 32-bit microcontrollers. The new series of Tiva-C microcontrollers has improved performance and refined features to a new level of quality. High level connectivity and sensor aggregation of Tiva-C Series microcontrollers make them perfect for connected applications like home, building and industrial automation. The Texas Instruments 32-bit microcontroller has more effective processing performance per clock cycle, integrated

mixed-signal circuits, flash memory and power consumption as compared to the other 32-bit microcontrollers. These qualities has increased the scope of applications that were not considered practical previously [46].

3.2 Architecture

Tiva-C series microcontroller provides high performance and advanced integration for 32-bit applications. These microcontroller devices provide cost effective solutions for control processing and connectivity applications like gaming equipment, home and commercial site monitoring and control, motion control, medical instruments, test and measurement equipment, factory automation, fire and security, smart grid solutions, intelligent lighting control, transportation. We can see the TM4C123GH6PM microcontroller in Figure.3.1. The TM4C123GH6PM microcontroller combines complex integration and high performance with the following features.

3.2.1 ARM Cortex-M4

The Cortex-M4F processor is ideal for embedded applications. It is build with a 3-stage pipeline Harvard architecture. Harvard architecture is a computer architecture that has separate areas for storage and signal pathways of instructions and data. It can easily operate on up to 32 bits wide data. The Cortex-M4 Thumb-2 instructions include arithmetic, logical, bit, branch and data movement operations, multiplication, bitfield manipulation, conditional prefixes and operates on 8, 16 and 32-bits of data. Thumb-2R instruction set

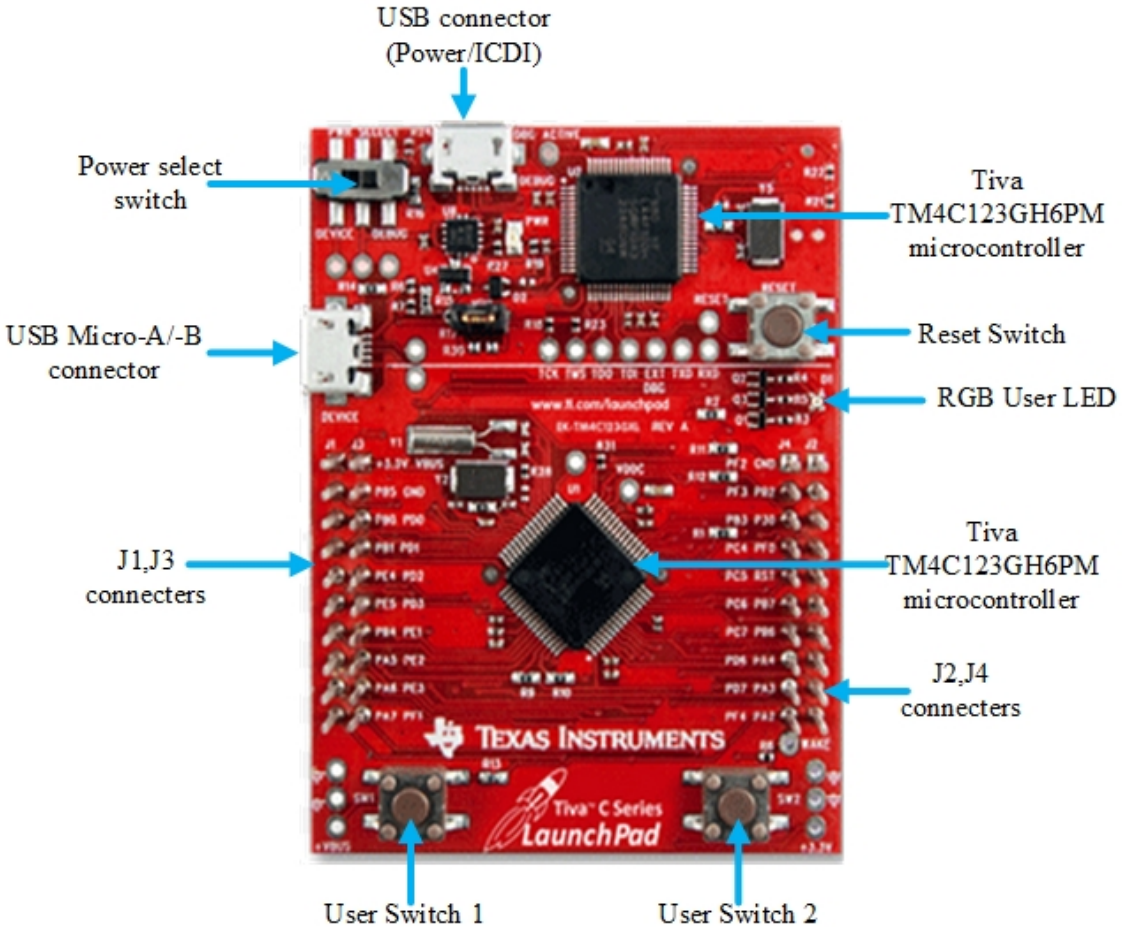


Figure 3.1: Tiva EK-TM4C123GXL LaunchPad

is a high-density, power-efficient instruction set useful for general-purpose data processing and control operations. The Cortex-M4 core has digital signal processing (DSP) extensions and single instruction multiple data (SIMD) instructions. It also has single-cycle 32-bit or dual 16-bit multiply-accumulate instructions and saturating arithmetic instructions in digital signal processing unit. SIMD can be optimized for arithmetic operations of large data. It can perform four 8-bit or two 16-bit arithmetic operations in a single cycle. The hardware divide logic produces a result in between 2 and 12 clock cycles. The single precision floating point instructions are compliant with the IEEE 754 standard. floating-point unit has simplified the implementation and programming of floating-point routines [46]. The Cortex-M4 processor is incorporated with a nested interrupt controller (NVIC) that provides industry-leading interrupt performance. The nested interrupt controller (NVIC) includes a non-maskable interrupt (NMI) and provides eight interrupt priority levels. The tight integration of the processor core and NVIC has reduced interrupt latency by providing fast execution of interrupt service routines (ISRs). For low-power designs the NVIC coordinates with the sleep modes and also Deep-sleep mode in which whole device powered down quickly. The cortex-M4 memory protection unit (MPU) divides the memory map into several regions and defines the location, size, access permissions, and memory attributes of each region. The MPU supports independent attribute settings for each region, overlapping regions, and export of memory attributes to the system. The Cortex-M4 MPU memory map is unified, meaning that instruction accesses and data accesses have the same region settings.

3.2.2 Power Saving Mode

The power consumption of the system is reduced due to the hibernation module. When the system is idle hibernation module shut down the system power only hibernation module is powered on.The system power can be restored by external signal or by using built in real time clock. The hibernation module is incorporated with a 32-kHz oscillator circuit, a supporting real-time clock (RTC) module, a battery monitor circuit and sixteen 32-bit words of backup battery SRAM. All GPIO pins state can be saved during hibernation.

3.2.3 Analog-to Digital Converter

Tiva-C Series microcontrollers have high-quality and high-resolution analog-to-digital converters (ADCs). The TM4C123GH6PM microcontroller has two ADC modules each of one is comprised of 12-bit precision,12 input channels, single-ended and differential input support and internal temperature sensor. The sample rate of 1 MSPS supports both the full resolution and accuracy of the ADCs as compared to other microcontrollers that degrade the quality of the readings to achieve higher sample rates.

3.2.4 Memory

The TM4C123GH6PM microcontroller includes 32 KB of bit-banded SRAM, internal ROM, 256 KB of Flash memory, and 2KB of EEPROM. Flash memory is organized in 1-KB independently erasable blocks to make its programming simple. Flash memory can

be protected in a set of 2-KB block. Reprogrammable feature of flash memory vanishes the fear of wearing out the memory from re-flashing for data collection, configuration parameters or program modifications. EEPROM is used to store long-term variables that need to survive power outages and dead batteries. The integrated memory allows for the execution of code while writing values to nonvolatile memory. The micro direct access memory controller can transfer data to and from the on-chip SRAM. Flash memory and ROM are located on a separate internal bus that why it is not possible to transfer data from the Flash memory or ROM with the micro direct access memory controller.

3.2.5 Communication Feasibility

The TM4C123GH6PM microcontroller does have eight Universal Asynchronous Receiver/-Transmitter (UART), four Synchronous Serial Interfaces (SSI), six Inter-Integrated Circuit (IC), two Controller Area Network (CAN) and USB 2.0 modules. The UART module is encapsulated with a serial IR (SIR) encoder/decoder block that can be connected to an infrared transceiver to implement an IrDA SIR physical layer. SSI module can communicate with other devices that have either Freescale SPI, MICROWIRE, or Texas Instruments synchronous serial interfaces. The transmit and receive paths of SSI module are buffered with internal FIFO memories that can store eight 16-bit values independently in both modes. The UART and SSI also support the micro direct access memory(DMA) interface [47].

3.3 Real Time Simulator

Simulation has played a critical role in the successful development of a large number of applications like the layout of transmission lines in large scale power systems and the optimization of motor drives in transportation. Electrical and power systems have been using simulation tools since the early twentieth century. The evolution of simulation tools gradually improved with the evolution of computing technologies. Analog, hybrid and then digital simulators have come in the market. With the passage of time performance of simulation tools dramatically increased while their cost has decreased. Nowadays affordable and high performance simulation tools are available for researchers and engineers. Opal-RT technologies has designed the real-time simulator in reference to embedded systems. Embedded systems are electronic devices designed to interface with the real world and provide control, interaction, and convenience of some form. In a real-time system, the embedded device is given a predetermined amount of time to read input signals, to perform all necessary calculations and to write all outputs. The amount of time is known as step size. Fixed-step solver solve the model at regular time intervals [48].

3.3.1 Opal-RT OP5600

Opal-RT OP5600 is a complete simulation system invoked with Spartan 3 or vertex 6 FPGA platform. It has powerful real-time target computer with 12 Intel processor cores of 3.3 GHz and Linux REDHAT operating system. It comes with Spartan 3 or vertex 6

Figure 3.2: Opal-RT OP5600 Real Time Simulator [49]

FPGA. It has DB37, RJ45, and mini-BNC connectors. These standard connectors allow quick connections for monitoring. It has up to 4 PCI or PCIe connector slots. we can see the Opal-RT OP500 simulator in figure 3.2.The black box is divided into two parts. Lower part has target computer that includes ATX motherboard, 6 DRAM connectors, 250 Mb Hard Drive and PCI slots. The upper part is composed of front end processor (Spartan 3 or vertex 6 FPGA board) and analog and digital I/O mezzanines boards. These mezzanines boards include 32 digital outputs, 32 digital inputs, 16 analog outputs and 16 analog inputs. The PCIe synchronization board is used to transmit and receive data between the target and the front-end processor.

The RJ45 and mini-BNC connectors on the front of OP5600 are used for monitoring

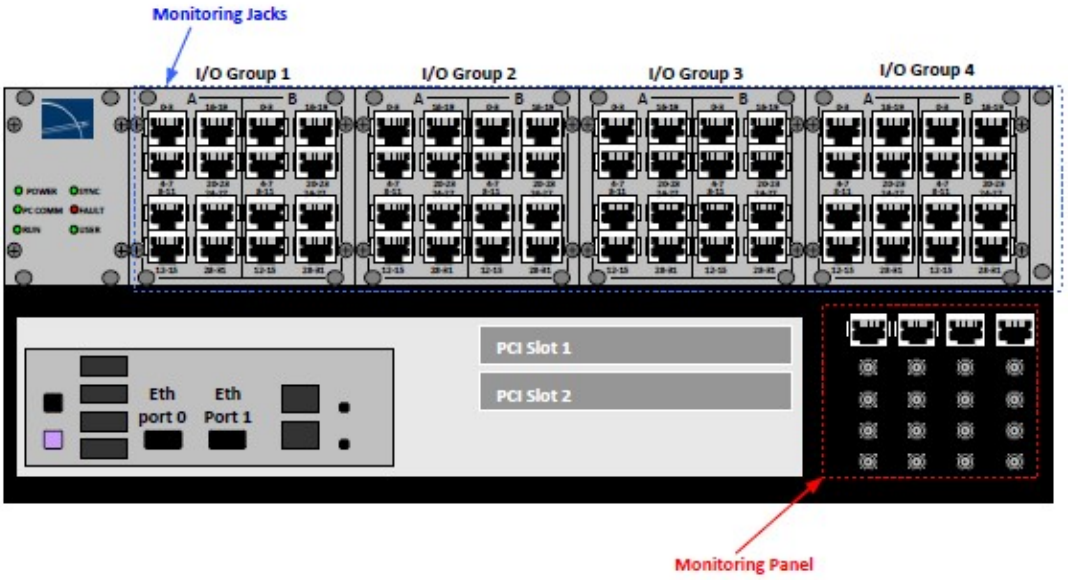


Figure 3.3: Front view of OP5600 Simulator

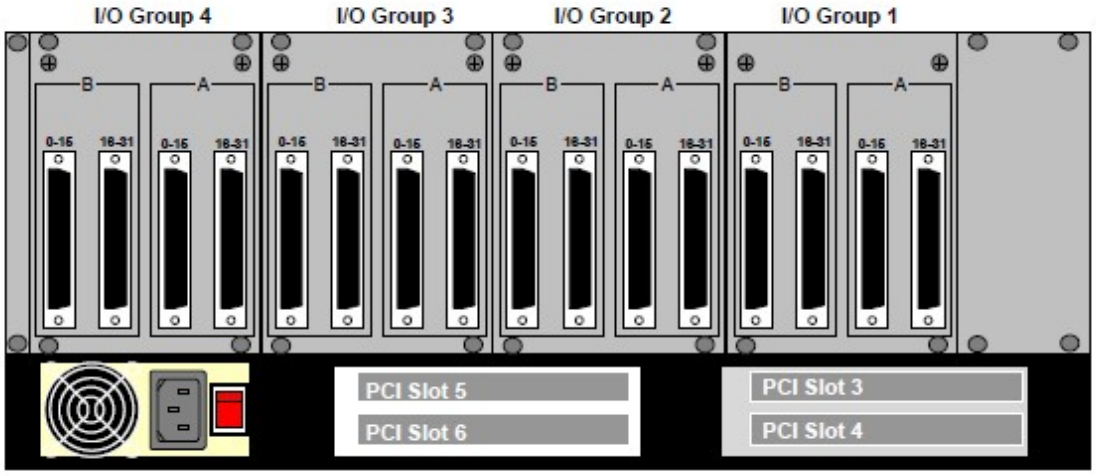


Figure 3.4: Rear view of OP5600 Simulator

purpose only. One RJ45 connector drives four signals. Mini-BNC connectors can be connected with cable to the monitoring device like oscilloscope. Figure.3.3 shows the front view of OP5600. The DB37 connectors at the rear side of OP5600 simulator are used to connect external hardware to the OP5600.One DB37 connector drive 16 signals. There are 16 differential analog input channels. Each channel has one 16-bit analog to digital converter(ADC). Each ADC can be sample at 500KS/s. All channels can be sampled simultaneously. Analog output module has 16 single ended output channel that are short circuit protected. Each channel contains digital to analog converter that can be sample at 1MS/s. Total throughput of ADC and DAC is 8MS/s. DB37 connectors at rear side of OP5600 simulator present 32 digital input and 32 digital output channels. Optically isolated digital inputs provide over-voltage protection. Digital outputs are short circuit

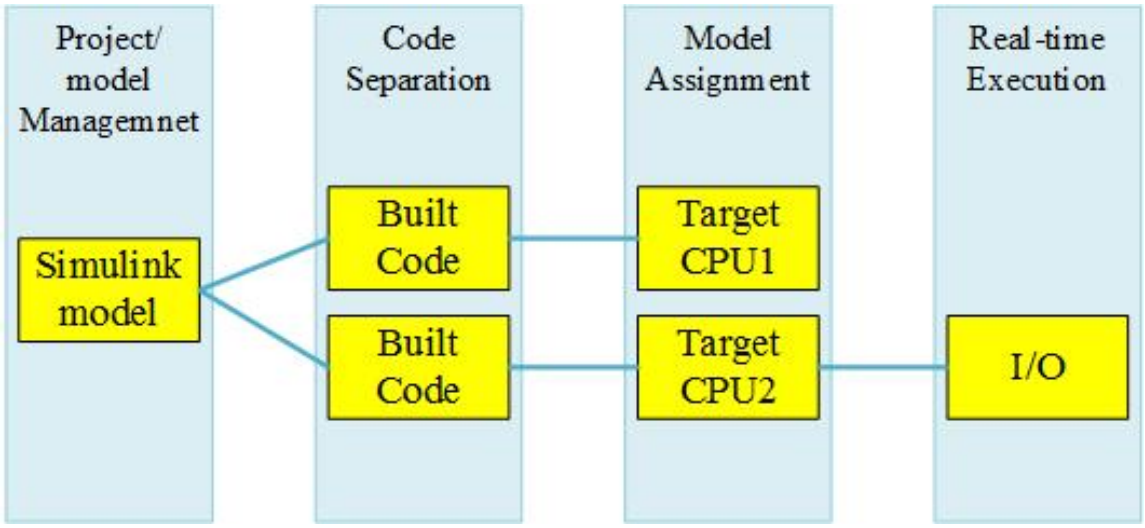


Figure 3.5: System architecture of RT-LAB

protected by auto-resettable fuse [49].

3.3.2 RT-LAB

RT-LAB provides the complete technology framework for real time simulation of complex plant models for rapid control prototyping and HIL (Hardware in loop) applications. RT-LAB software follow specific steps to run the model as described in figure 3.5. First step is link the Simulink model from Matlab. Second step is build the model. Build configuration of the model is prepared before building model for real time. Third step is compiling the model. Target node on which the model will be compile must be selected before compiling. Next step is load the model and run the real time simulation. RT-LAB software window is shown in figure 3.6.

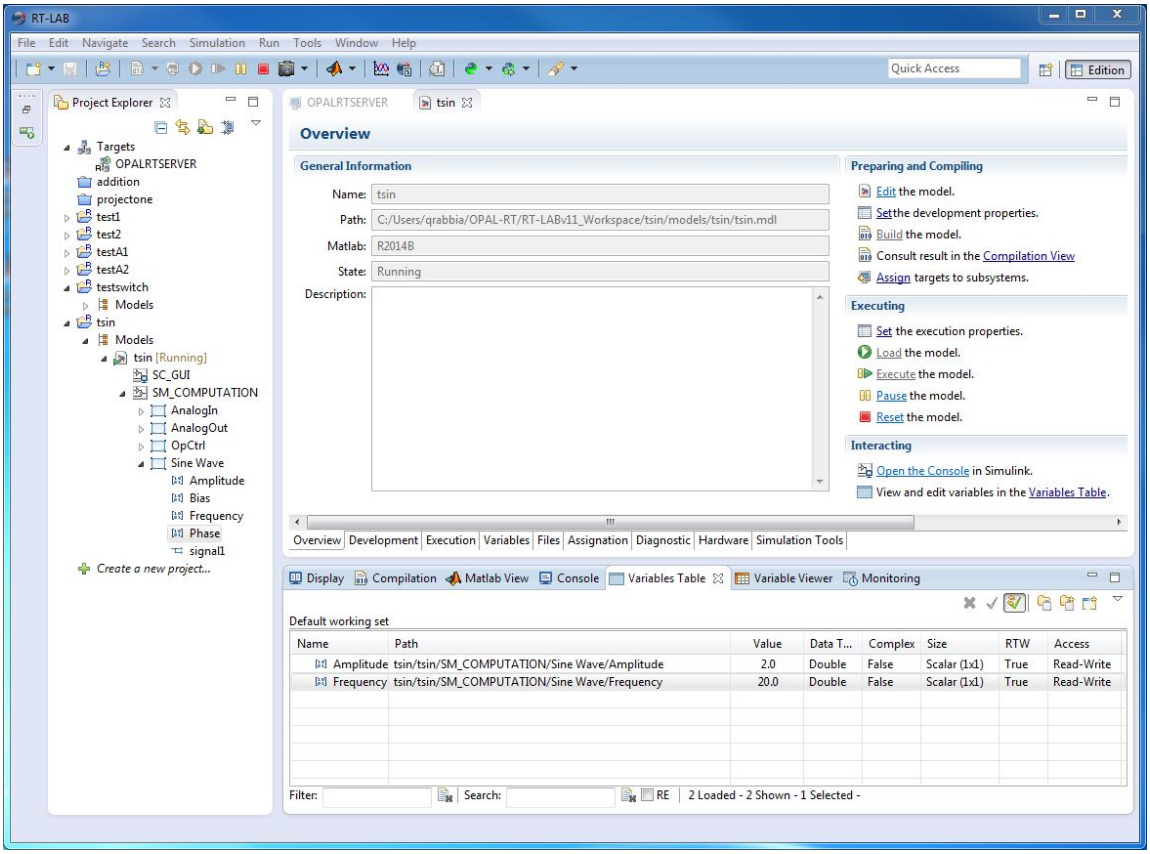


Figure 3.6: RT-LAB software window

There are two simulation mode in RT-LAB software synchronized and hardware synchronized. Software synchronized: The model is synchronized on the internal RTOS timer and run in real time. When executing a distributed simulation only one computation node is synchronized. Other subsystems are synchronized with communication link. Operating system use the CPU clock as a reference for synchronization. The models without Opal-RT I/Os can be test and some PCI I/Os can be run. Hardware synchronized: The external hardware timer is uses to synchronize and run the model in real-time. To specify and configure the location of external clock a specific block is inserted in the model. Hardware synchronized mode is used when Opal-RT I/Os are driven by the model and signals are physically input to simulator and output from the simulator. Simulation is synchronized with I/O board clock

3.3.3 Testing

We built a simple Simulink model to test the functionality of the Opal-RT. Model is built in matlab by using Simulink library components. The model is grouped in two subsystems Computation subsystem and GUI subsystem as shown in figure 3.7. RT-LAB distinguish computation subsystem and GUI subsystem. The computation subsystem is assigned and executes on one CPU core of the real time target while GUI subsystem is displayed on the Host PC as shown in figure 3.10. The data between two subsystems is exchanged asynchronously through the TCP/IP link. The computation block can be split into more than one subsystem. The communication between two computation subsystems will be

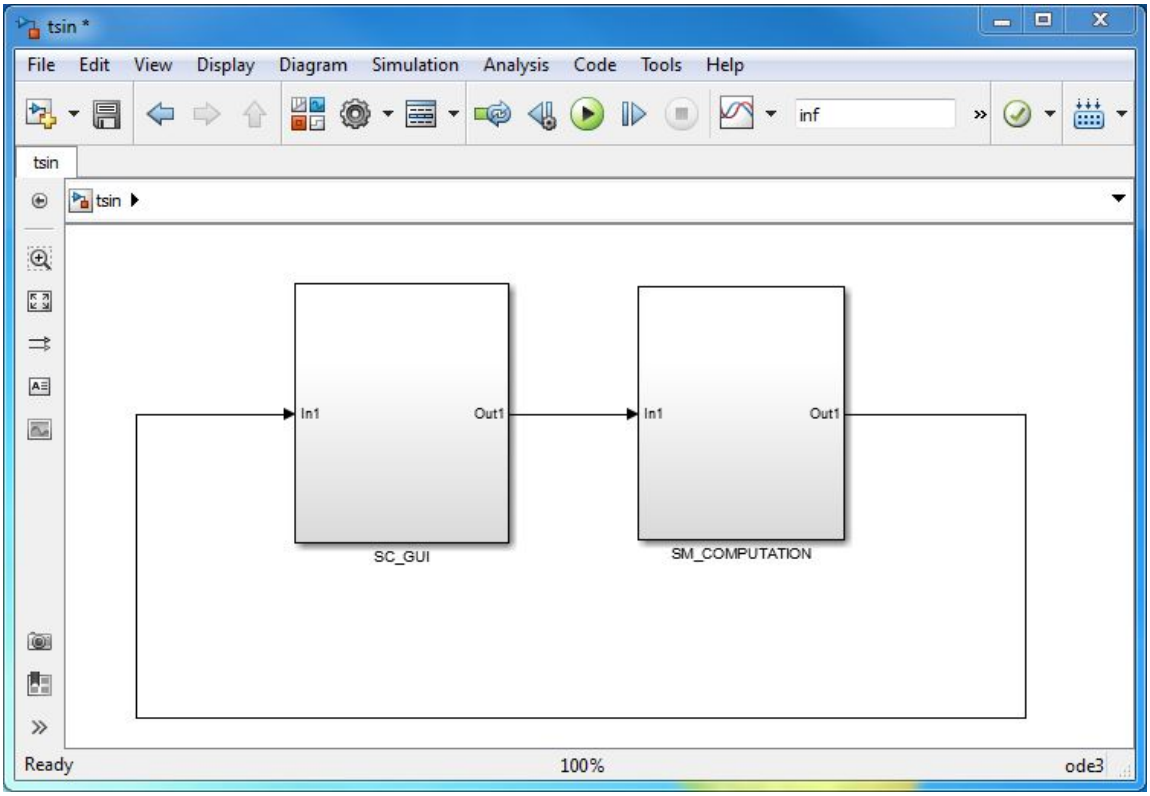


Figure 3.7: Subsystems of the model

done synchronously through shared memory.

All inputs of the subsystems are connected with Opcomm block as shown in figure 3.8 and figure 3.9. All inputs must go through Opcomm block before any operations is done on the signals. Opcomm block is in the RT-LAB library. It is responsible for communication between computation subsystems and GUI subsystems.

We run the model on real time simulator and see the output waveform on oscilloscope as shown in figure 3.11. The output wave form can also be seen on GUI subsystem scope

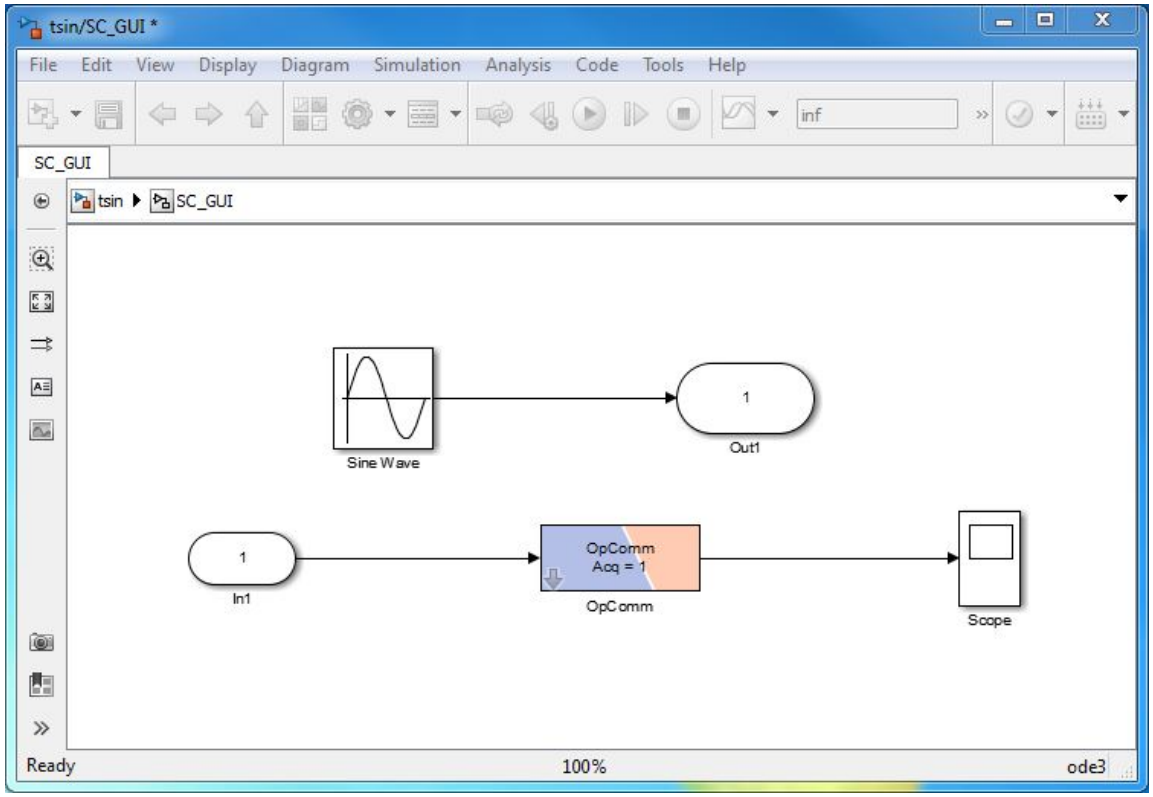


Figure 3.8: GUI Subsystems of the model

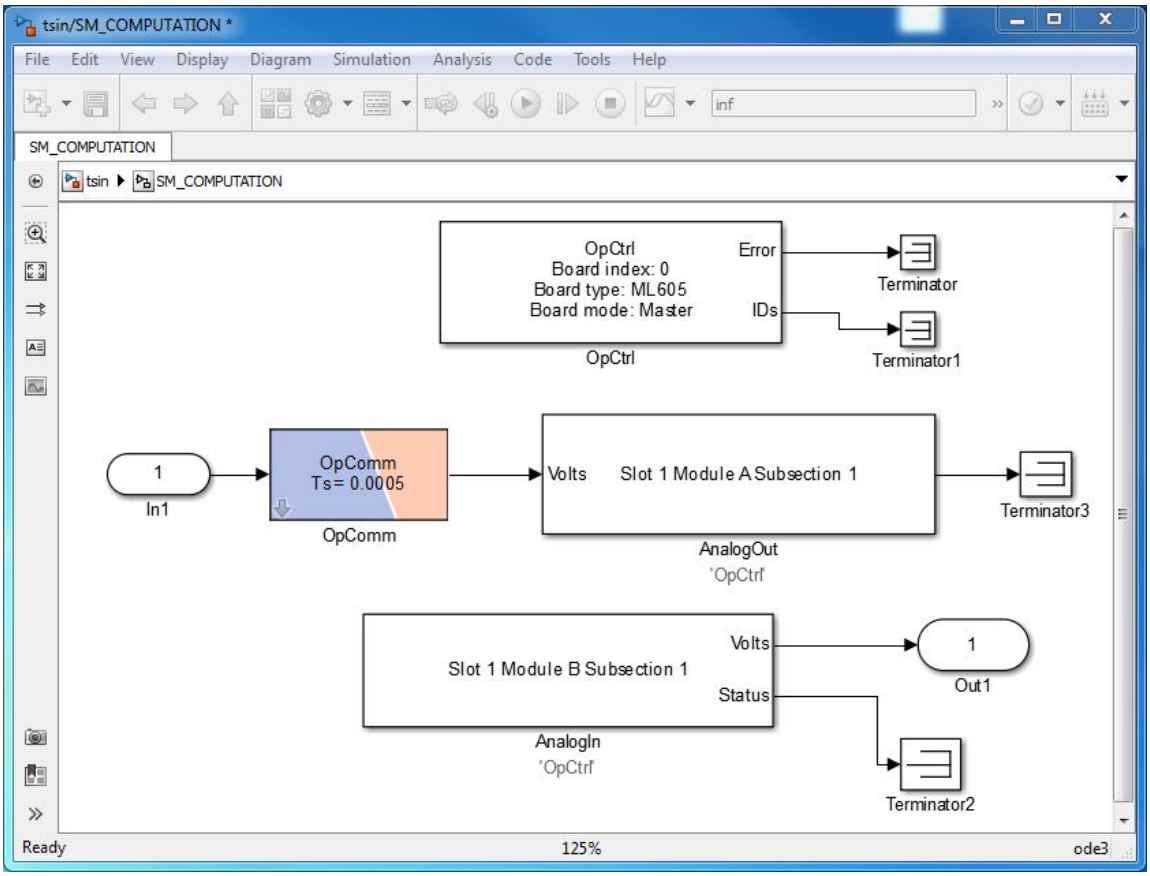


Figure 3.9: Computation Subsystems of the model

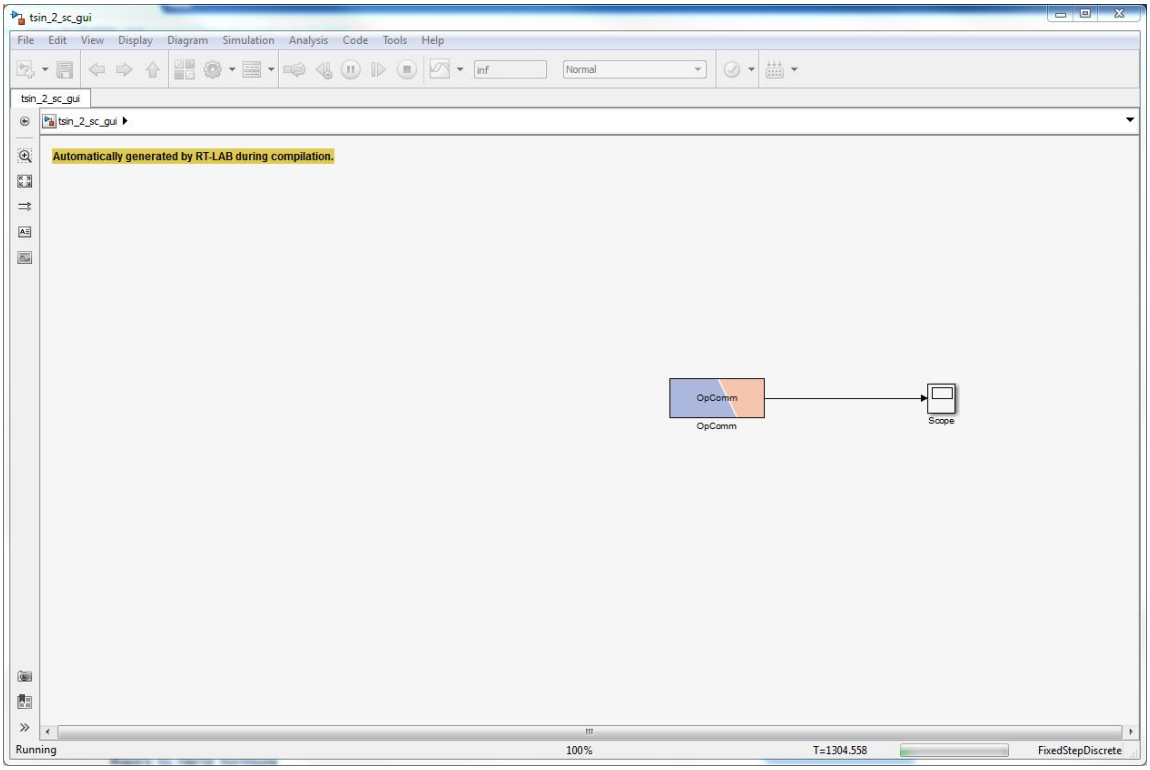


Figure 3.10: GUI subsystem launched by RT-LAB

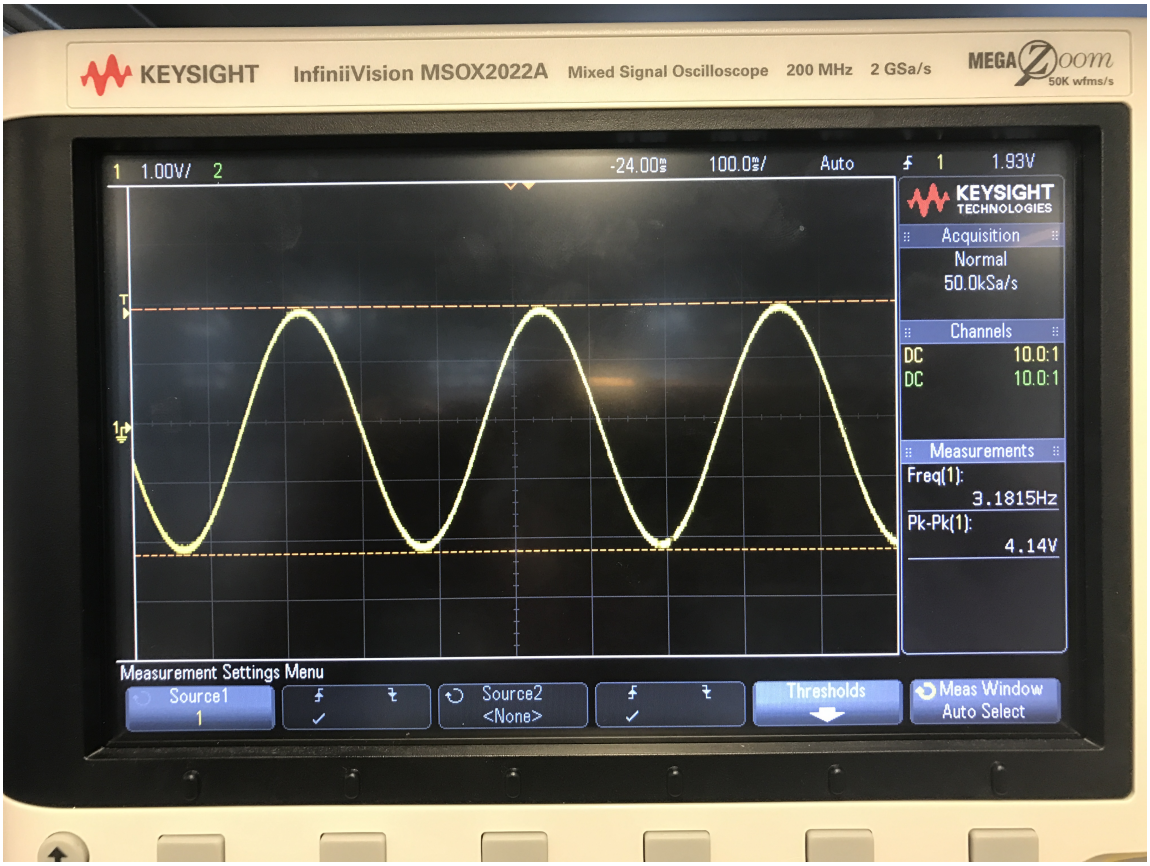


Figure 3.11: Output at oscilloscope

launched by RT-LAB as shown in figure 3.12.

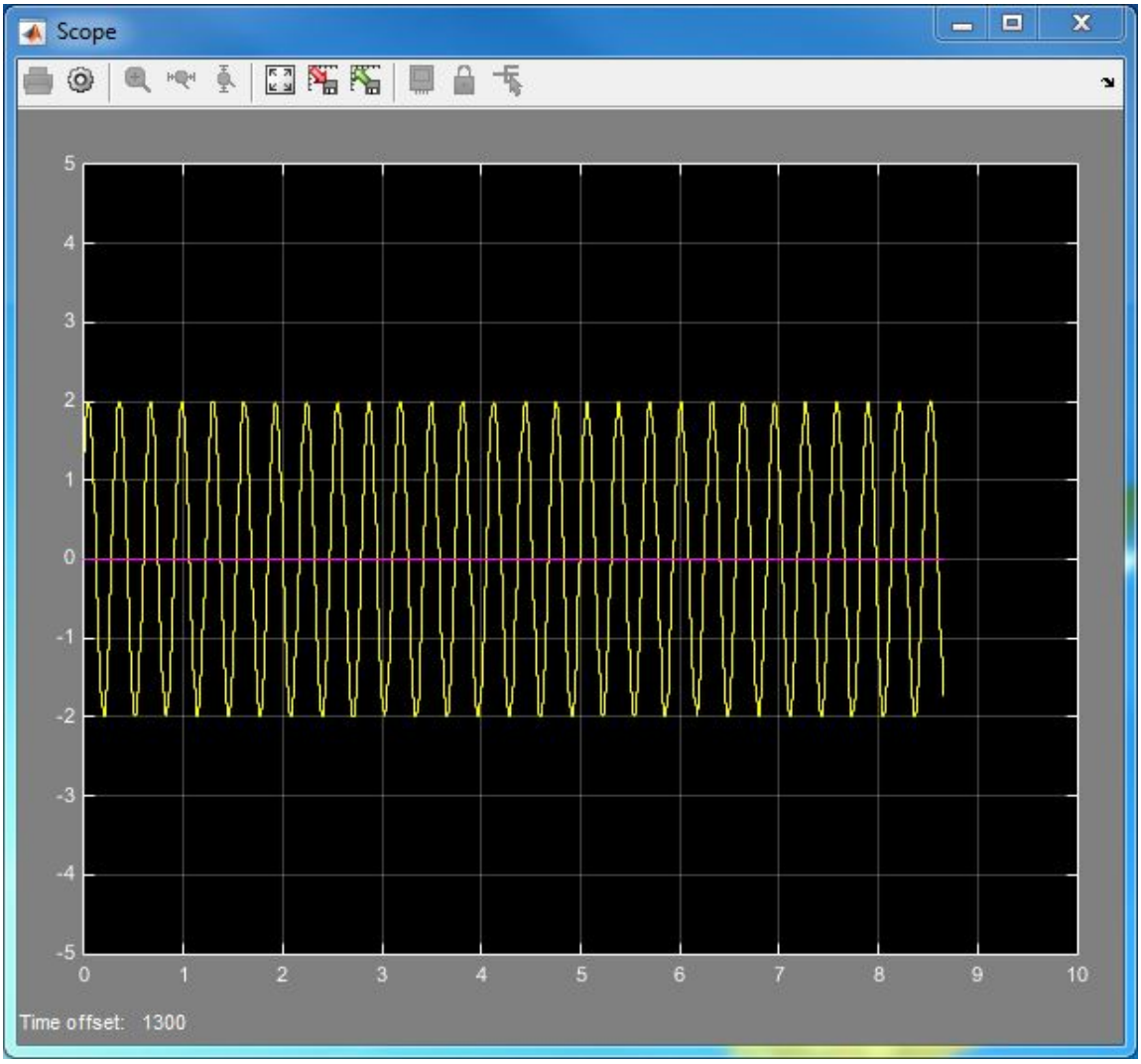


Figure 3.12: Output at GUI scope launched by RT-LAB

3.4 Chapter Summary

This chapter discusses how Tiva-C microcontroller is useful for embedded applications and describes its features in detail. Real time simulator Opal-RT is used to built the microgrid model in this thesis is described in this chapter. Features of Opal-RT 56000 and RT-LAB software are illustrated. In the next chapter, multiagent system implementation and results will be discussed.

Chapter 4

System Implementation

4.1 Model of the design

For the implementation of our system we built the microgrid model in Matlab Simulink.The microgrid model as shown in Figure 4.1 and is composed of 39 buses [50]. It has two photovoltaic (PV) units. The capacity limit of each PV is 1.5 MVA. There are two Distributed Battery Energy Storage Systems (BESS) with capacity of 300 kWh. Maximum and minimum State of charge (SoC) limits of each BESS are 90 percent and 10 percent. The upper and lower voltage boundaries of battery storage agent (BSA) are 1.05 per unit and 0.95 per unit respectively. Voltage limit for the distribution system is 6 percent. As we can see in Figure 4.1 there are two zones. Two agents will control these two zones. Each bus has voltage between the limit of 0.95 per unit to 1.05 per unit. If there is a change in voltage of any bus out of the given limit. It means there is a violation and the agent of the associated bus is responsible to mitigate the voltage [50].

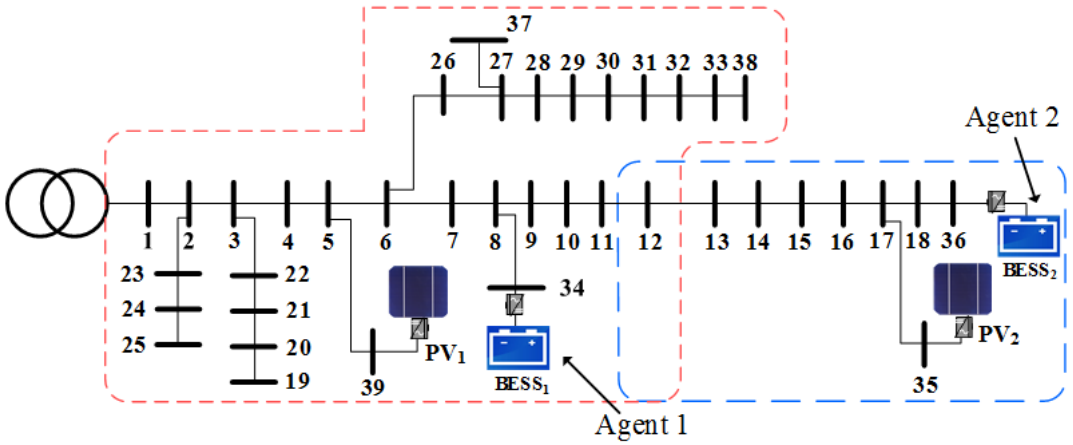


Figure 4.1: A microgrid consists of 39 buses, two energy storage systems and two solar power generation units

The microgrid model is loaded in the real time simulator Opal-RT. Simulation is running on Opal-RT and generating the required outputs for agent1 and agent 2. The data is transferred from Opal-RT to the agents at each time step t as we can see in Figure 4.2. Received data is composed of following values.

- 1) V_i=Voltage of the battery
- 2) V_j= voltage of the solar power bus
- 3) V_{pzc}=Voltage of common node between two zones(point of zone coupling).
- 4) P=Active power of battery
- 5) Q= Reactive power of battery

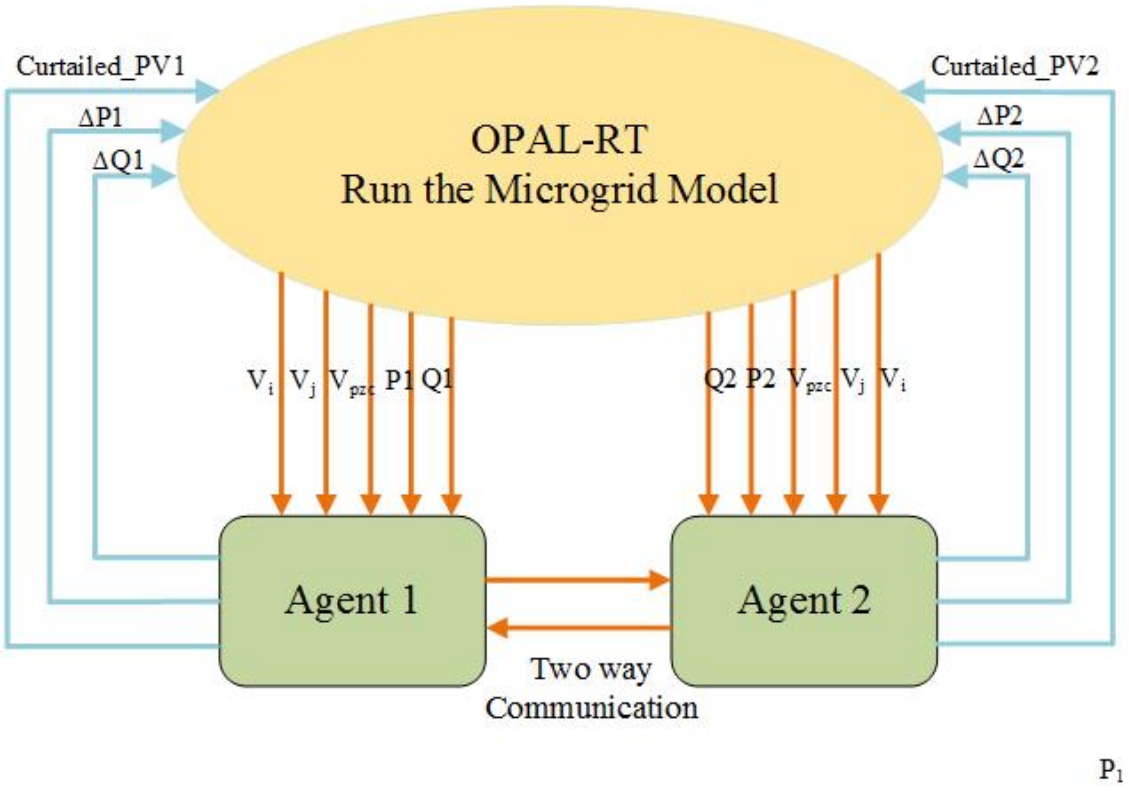


Figure 4.2: Multiagent Framework using Opal-RT OP5600 Real Time Simulator

After receiving data from Opal-RT each agent will perform the proposed algorithm. We can see the flowchart of algorithm performed by agents in Figure 4.3. After running the algorithm each agent will send back three pieces of information to the microgrid modeled in Opal-RT. These three values are ΔP (change in active power), ΔQ (change in reactive power) and change in photovoltaic voltage (Curtailed-PV). To produce the output at any time instant t the proposed algorithm will use the received data at time t and $t-1$. Each agent will save the received data at any time instant to use it to produce outputs for the next time step. After receiving the data from Opal-RT each agent will initialize the output solution ΔP, ΔQ and Curtailed-PV. Then it will estimate the received data for next time step t+1 according to following equation

$$data(t+1) = data(t) + (data(t) - data(t-1)) \tag{1}$$

The violation will be checked by each agent for estimated values of voltage at time $t+1$ by using equation (2).

$$\Delta V(t+1) = \begin{cases} \max(V(t+1) - V_{ub}) - \epsilon, & \forall V > V_{ub} \\ \min(V(t+1) - V_{lb}) + \epsilon, & \forall V < V_{lb} \end{cases} \tag{2}$$

If there is a violation in voltage, the agent will calculate the value of ΔP and ΔQ in the next time step to find the solution by using equations (3) and (4).

$$\Delta Q(t+1) = (P(t+1) + \Delta P(t+1)) \times \tan(\cos^{-1}(pf_{min})) - Q(t+1) \tag{3}$$

In equation (3) pf_{min} is the minimum power factor of the battery.

$$\Delta P(t+1) = \frac{(V(t+1)\times\Delta V(t+1))-(P(t+1)+\Delta P(t+1))\times\tan(\cos^{-1}())-Q(t+1))\times(\frac{\partial V}{\partial Q}\times V(t+1))+\sum_{n=i}^{j-1} x}{(\frac{\partial V}{\partial P}\times V(t+1)+\sum_{n=i}^{j-1} r)+(\frac{\partial V}{\partial Q}\times V(t+1))+\sum_{n=i}^{j-1} x)\times\tan(\cos^{-1}(pf_{min})} \quad (4)$$

r and x are representing resistance and reactance of the bus in equation (4). If the agent fails to find the solution for voltage violation. The rate of point of zone coupling voltage will change according to equation (5).

$$\frac{\partial V_{PZC}}{\partial V} = \frac{V_{PZC}(t) - V_{PZC}(t-1)}{V(t) - V(t-1)} \quad (5)$$

The agent will send the request to the neighboring agent to adjust the point of zone coupling voltage to a certain limit. The agent will send the following three pieces of information with the request.

- Agent ID (Identification number of sending agent)
- $V_{PZC}^{limit}(t+1)$ (limit of point of zone coupling voltage)
- $\Delta V_{PZC}(t+1)$ (agent's contribution towards point of zone coupling voltage limit)

The point of zone coupling voltage limit is calculated by equation (6) and contribution of agent towards this voltage limit $\Delta V_{PZC}(t+1)$ is calculated by equation (7). The agent is sending its contribution to maintain point of zone coupling voltage limit with request message to inform the neighbor that does not contribute the same value.

$$V_{PZC}^{limit}(t+1) = V_{PZC}^{pre}(t+1) + \Delta V_{PZC}(t+1) \quad (6)$$

$$\Delta V_{PZC}(t+1) = \begin{cases} \frac{\partial V_{PZC}}{\partial V} \times (V_{ub} - V_{PZC}^{pre}(t+1)) - \epsilon, & \forall V > V_{ub} \\ \\ \frac{\partial V_{PZC}}{\partial V} \times (V_{PZC}^{pre}(t+1) - V_{lb}) + \epsilon, & \forall V < V_{lb} \end{cases} \tag{7}$$

The neighboring agent will receive and process the request to find the solution and reply back with either good or no-good message. If the sending agent receives the good message, both agents will send their output solution to Opal-RT. If the agent receives no-good message from its neighbor it will curtail solar power by calculating Curtailed-PV and send the output solution to the Opal-RT. The solar power curtailment depends upon the active power sensitivity of the bus and estimated voltage at next time step. It is illustrated in equation (8).

$$\Delta P_{PV}(t+1) = \begin{cases} S_i \times (V_{ub} - V^{pre}(t+1)) & \forall V > V_{ub} \\ \\ S_i \times (V^{pre}(t+1) - V_{lb}) & \forall V < V_{lb} \end{cases} \tag{8}$$

The agents are implemented on Tiva-C microcontroller TM4C123GH6PM. We have used Code Composer Studio for writing the code and uploading the configuration file to the target microcontroller after compiling and debugging. Figure 4.4 shows the flow chart of the design for agent. Opal-RT Simulator is generating the analogue data. The range of the data is 0.95 per unit to 1.05 per unit. The A/D (analogue to digital converter) Unit will convert the data into digital. We have implemented the analogue to digital converter in Tiva-C microcontroller for this purpose. Communication between the agents is done by using the Synchronous Serial Interface (SSI). Each agent will estimate the data for next

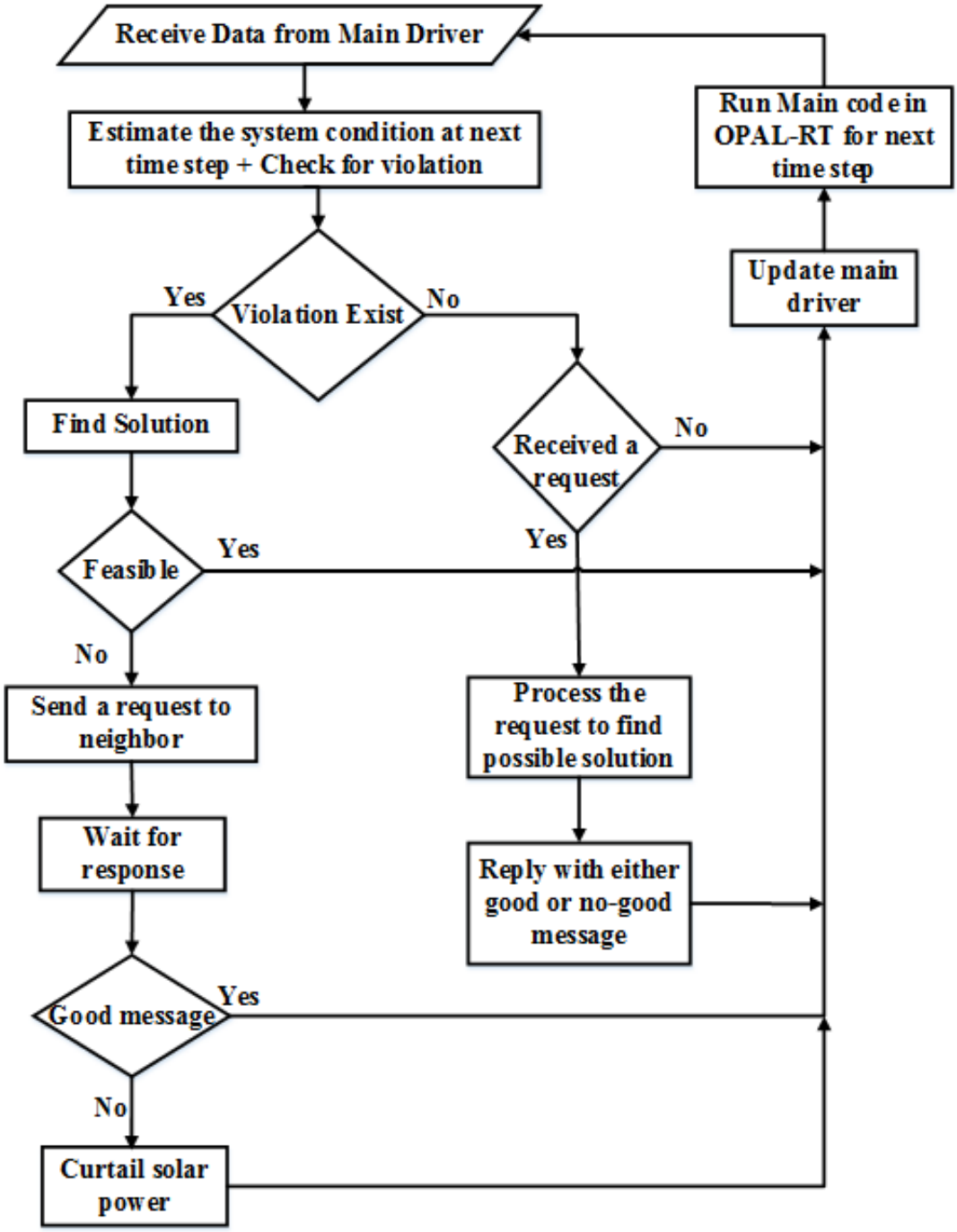


Figure 4.3: Multi-agent flowchart

time step and check for violation. There are three scenarios.

- No violation. The agent 1 will send message to agent 2 and update agent 2 that "I have no violation" and wait for reply. If agent 1 will receive the same reply from the agent 2 it will update the Opal-RT.

- Violation occur at agent 1. The agent 1 will find the solution. There are two possibilities either the solution is feasible or not.

  - Feasible solution. If the agent 1 finds the feasible solution. It will tell agent 2 "I do not need your help" and wait for response. If the agent 2 reply back with same answer the agent 1 will update the outputs.

  - No-feasible solution. In case the agent 1 cannot find the feasible solution. It will send the request to the agent 2 and wait for response. The agent 2 will process the request and reply with either good message or no-good message. Good-message means agent 2 can help agent 1. While receiving no-good message the agent 1 will downsize the solar power voltage and update the Opal-RT.

- Violation at both agents. when the agent 1 has the violation it will send the request to agent 2 and wait for response. If the agent 2 also has the violation it will reply back " I cannot help you". after receiving this response the agent 1 will update the outputs with the existing solution.

In these three scenarios agents are sending and receiving messages through the syn-

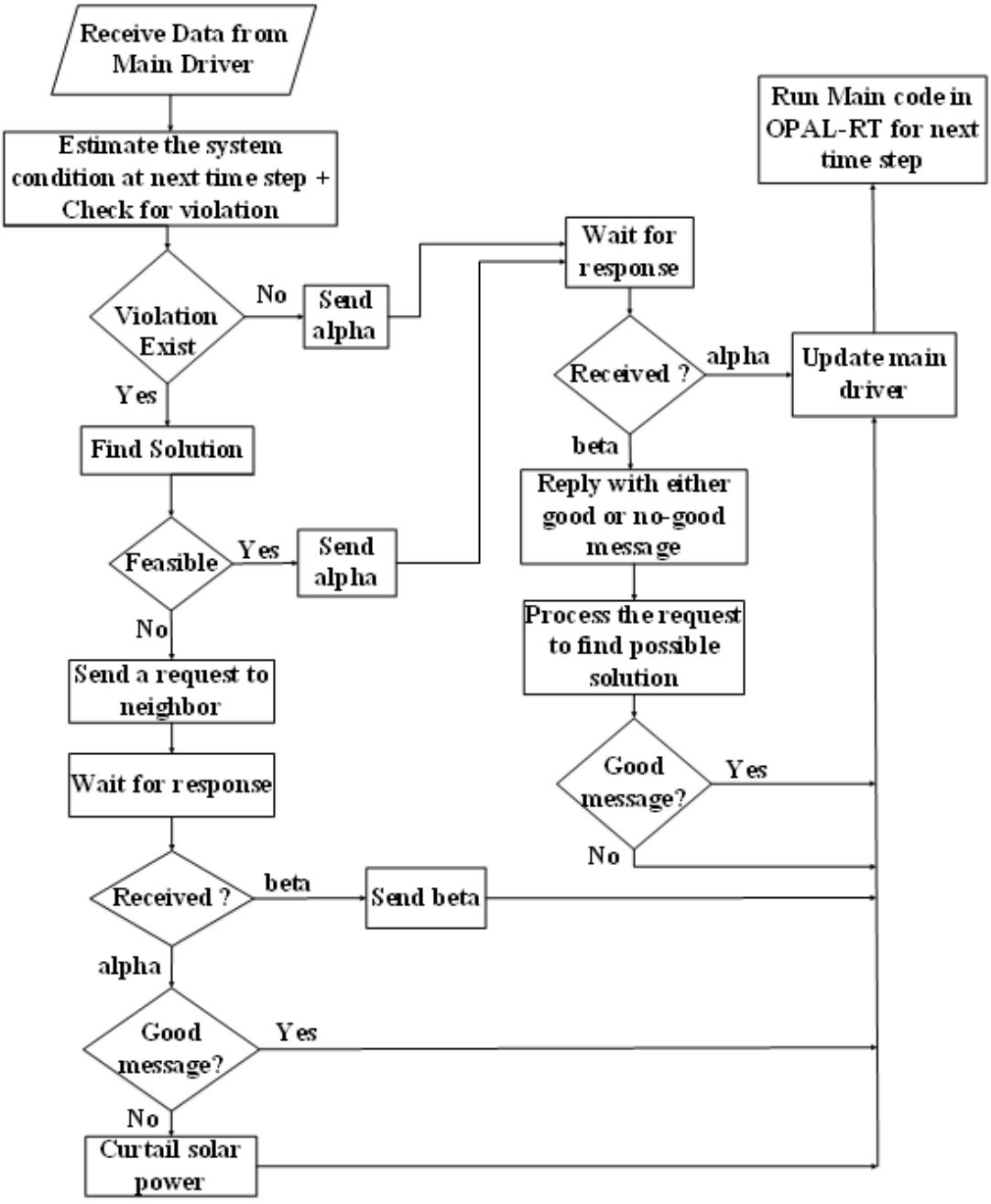


Figure 4.4: Flowchart for two agents

chronous serial interface (SSI). As we can see in flowchart of Figure 4.4 there are six paths of communication between two agents. Synchronization and computation time are the main factors for the communication between two agents. Computation time is the time taken by the agent to send and receive data from other agent. In our design we have implemented the two agents on two Tiva-C microcontrollers.

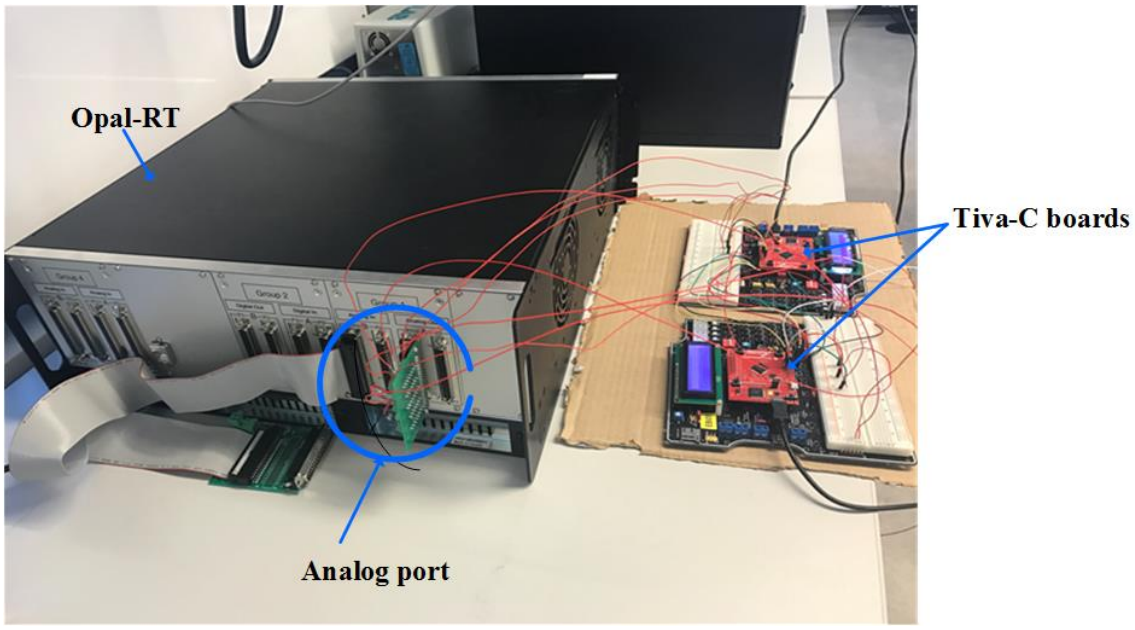


Figure 4.5: Experimental Setup of our work

4.2 Experimental Setup

Figure 4.5 is showing the hardware setup for our work. As we can see in the picture two Tiva-C microcontroller TM4C123GH6PM boards are connected to the Opal-RT OP5600 real time simulator. The two Tiva-C boards are implementing two agents of our design. Physical wires are used to connect the analog port of Opal-RT with GPIOs of Tiva-C boards. GPIOs of Tiva-C are 5 volt tolerant. Analog port has positive and negative channel. Positive channels are connected to the GPIOs of Tiva-C boards. We short the negative channels and connected to ground of the Tiva-C boards. Two SSI modules are using in this system for sending and receiving messages between two agents. The SSI

modules are working in master/slave mode. The SSI modules clock (Clk), receive (Rx), transmit (Tx) and serial frame (Fss) pins are connected with wires. A signal is generated by agent 2 to start the communication and write on GPIO. This GPIO is connected to the GPIO of agent 1 with physical connection and agent 1 read the signal from GPIO.

4.2.1 Challenges Faced

We faced a lot of challenges during the implementation of this design. Some of them are describing here.

- We were converting the inputs into fixed point unit. In fixed point conversion selection of scaling factor to get nearest results was a challenge. There was chance to loose precision during calculations. So we want to use floating point unit. Floating point unit was not working due to limited stack size. We upgrade the code composer license and increased the stack size to enable the floating point unit to work.

- After connecting the SSI ports for both boards manually with wires we were not receiving anything at SSI slave module. We created SSI interrupt to receive data through SSI slave module.

- We used the timer interrupt to synchronize the communication and processing for each time step. Setting the frequency of the timer interrupt according to frequency of the system was a real challenge.

- To receive exact values of inputs from Opal-RT to Tiva-C was a real challenge. We

used analog to digital converter in microcontroller. There was a lot fluctuation in the receiving values. We tested the ports of ADC channels of Tiva-C boards and figured out some of them were not working properly. After testing all ADC channels we used the correct ports to get the right range of input values.

4.2.2 Testing of Experimental Setup

We tested the experimental setup by running the sine wave model. Figure 4.6 is showing the setup for testing. Tiva-C microcontroller read the values of sine wave. We can see the values for sine wave in memory browser of Code composer window as shown in Figure 4.7. The output of this test is also shown in Figure 3.11.

4.3 Results

We have calculated the computation time for each communication path between two agents while running in real time. Computation time for each path has been illustrated in table.4.1. Sending alpha by the agent to the other agent means that it has no violation. While sending beta means there is a violation and it is sending request. Alpha and Beta can be any constant value to express the agent's situation to the other agent.

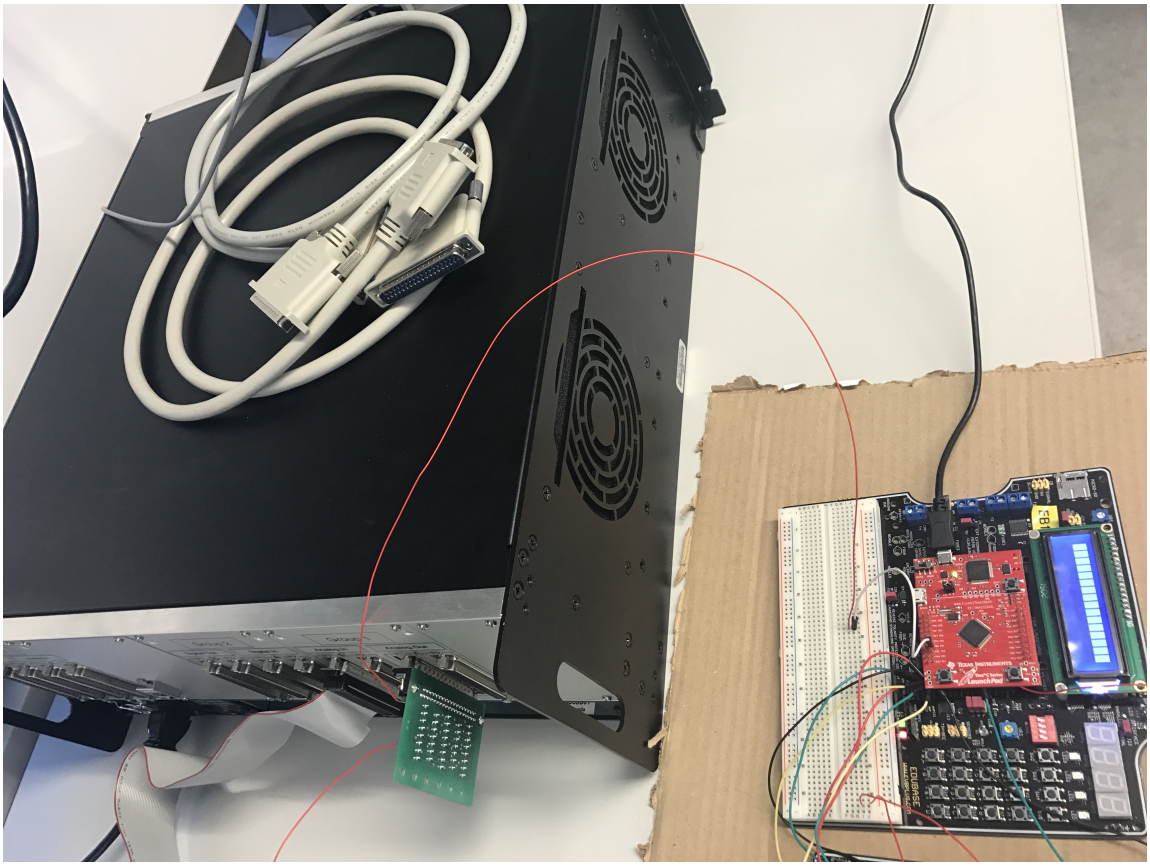


Figure 4.6: Testing of Experimental Setup

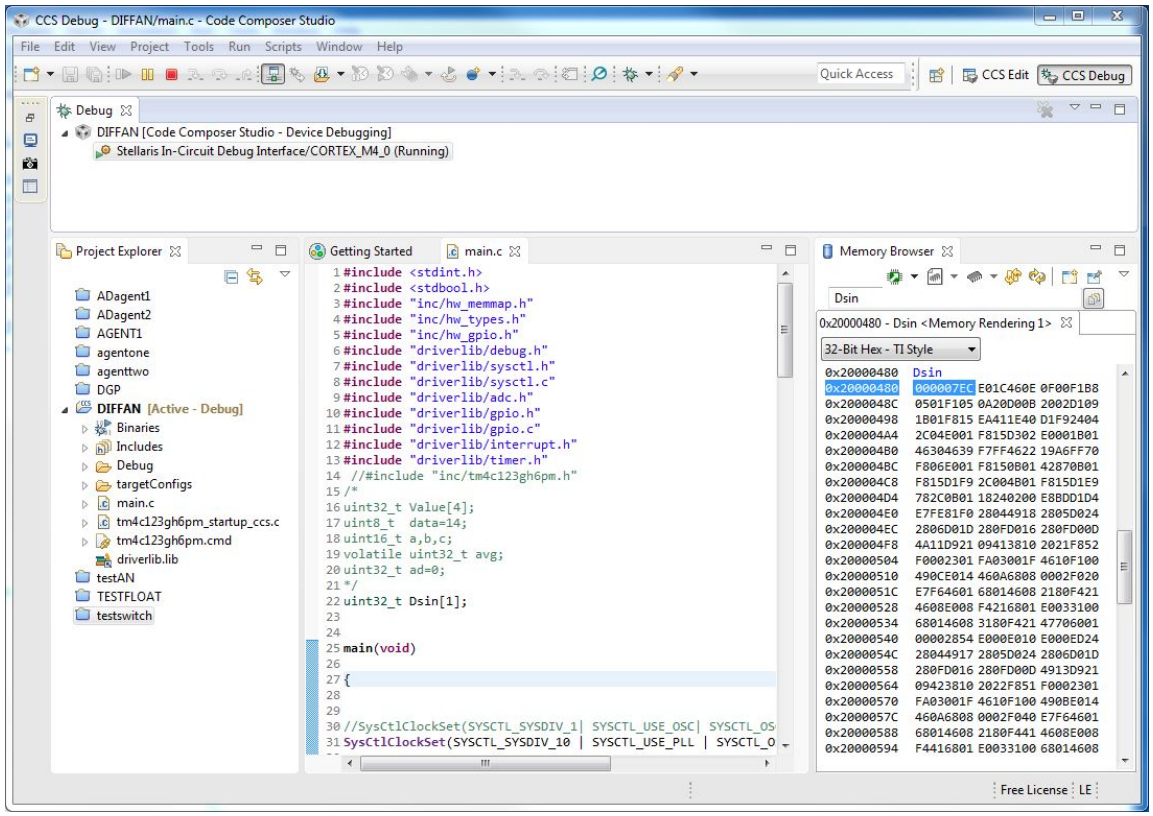


Figure 4.7: Reading of sine wave by Tiva-C

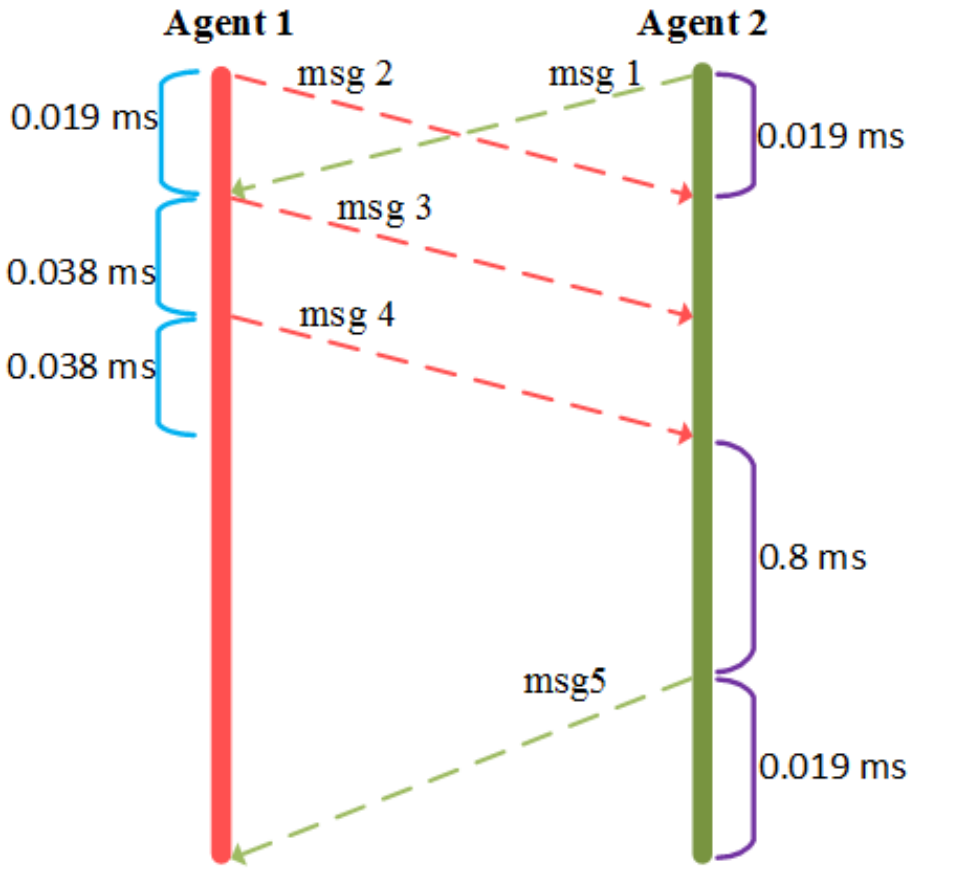
1	Sending and receiving Alpha	0.008 ms
2	After finding solution itself Sending and receiving Alpha	1.1 ms
3	Sending request, Receiving Alpha and No-good message	3 ms
4	Sending request, Receiving Alpha and good message	2.4 ms
5	Sending Alpha, Receiving request and reply No-good message	1.4 ms
6	Sending Alpha, Receiving request and reply good message	0.8 ms

Table 4.1: Computation time for communication paths between two agents

CASE Study 1: We conducted the case study to calculate execution time for communication between two agents. In this case study agent 1 has a violation and agent 2 does not have violation. At first both agents sent message to update each other either violation exist or not. Agent 1 could not find the feasible solution by itself so he requested agent 2 for help. Agent 1 sent two more piece of information to agent 2 as we can see transmission in Figure 4.8. Agent 2 received the messages and processed the request from

agent 1. Agent 2 reply back with good message to mitigate the voltage violation as shown in Figure 4.8.

CASE Study 2: This case study is to show the behavior of the agent when it detects the violation at any bus. In this case study agent 2 detects the violation in the voltage of bus 35 near to PV. Voltage upper bound is for each bus is 1.05 p.u. We read the values of bus voltage, active power of battery, reactive power of battery and power generation of PV for 11 time steps. As we can see in Figure 4.9 at time step t=2 violation occur and voltage of bus is 1.056 p.u. BESS active power , BESS reactive power and PV power generation at time step t=2 can be seen in Figure 4.10, Figure 4.11 and Figure 4.12 respectively. Agent 2 find the feasible solution and by applying new values of active power 0.199 MW and reactive power 0.102 MVAR at time step t=3 the bus voltage drops to 1.048 p.u. Another violation is detected by agent 2 at t=4. Similarly by finding the feasible solution and executing the decision variables the voltage violation mitigates at t=5. At time step t=6 violation occurred again and the bus voltage is 1.06 p.u as we can see in Figure 4.6. The agent 2 could not find the feasible solution by itself and requested agent 1 for help to mitigate the violation. Agent 1 replied with no-good message. So agent 2 curtailed the solar power by 0.0012 MW to remove the violation as we can see in Figure 4.12 at time step t=7. After curtailing solar power from 0.94 MW to 0.93 MW and applying new values of active and reactive power of battery 0.62 MW and 0.32 MVAR respectively the bus voltage drops to 1.032 p.u at time step t=7 . The same behavior is repeated and can be seen at time steps t=9 and t=11 in Figure 4.9 and related curtailed solar power in Figure 4.12.



Message 1 = no violation, agent 2 updating agent 1
Message 2 = Violation, agent 1 updating agent 2
Message 3 = piece of information for help
Message 4 = piece of information for help
Message 5 = good message

Figure 4.8: Exchange of messages between agents

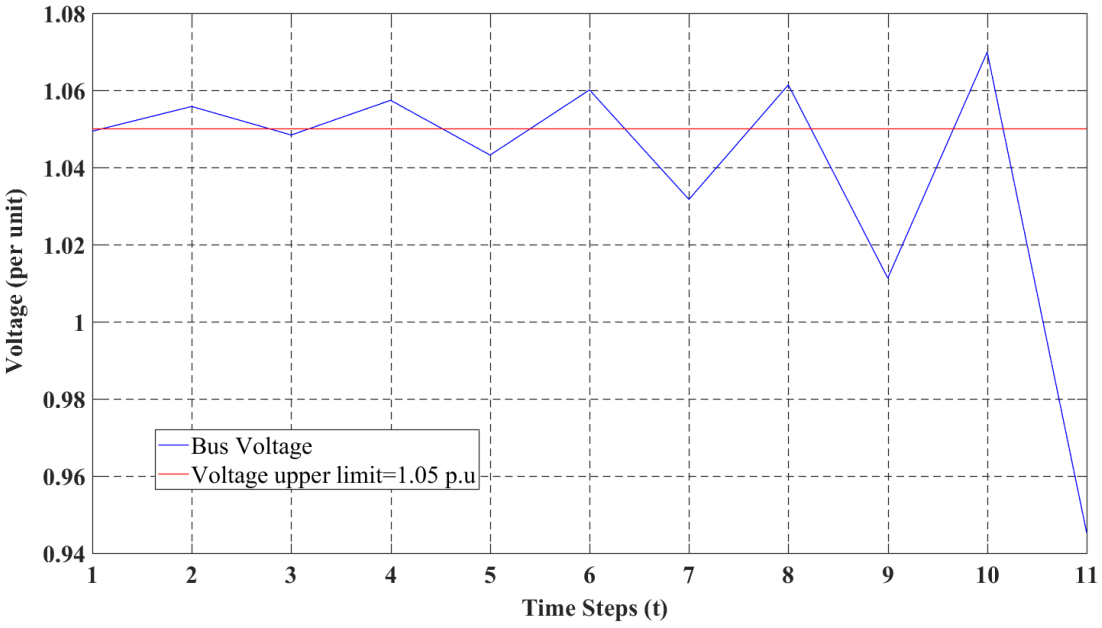


Figure 4.9: Bus Voltage associated with PV

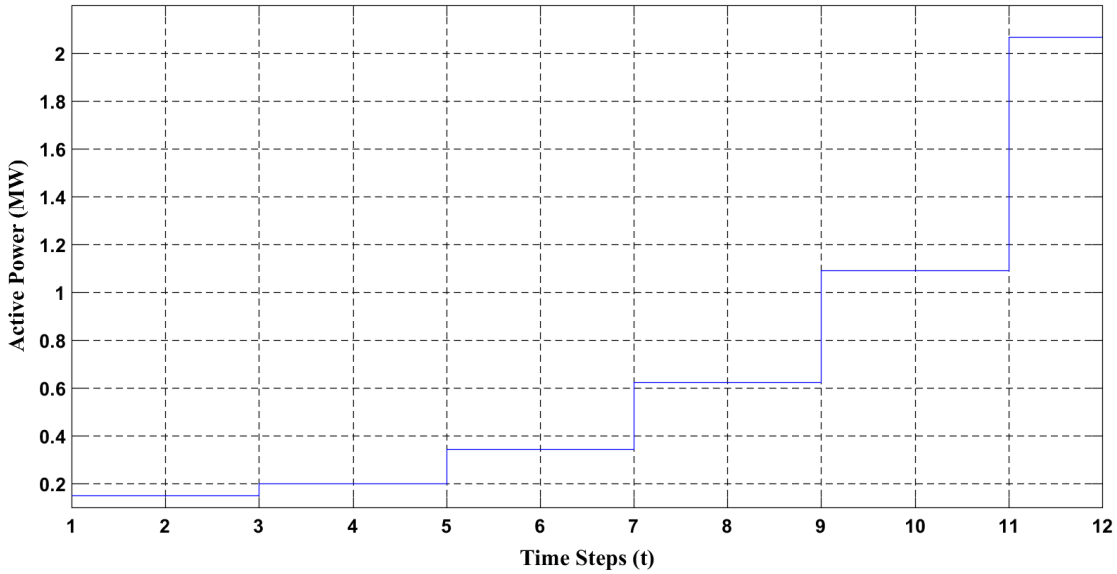


Figure 4.10: Active power of Battery

4.4 Chapter Summary

In this chapter, the design of microgrid model, design of multiagent system and implementation of multiagent system is illustrated. Experimental setup and testing of experimental setup is discussed. This chapter presented the results by describing the case studies.

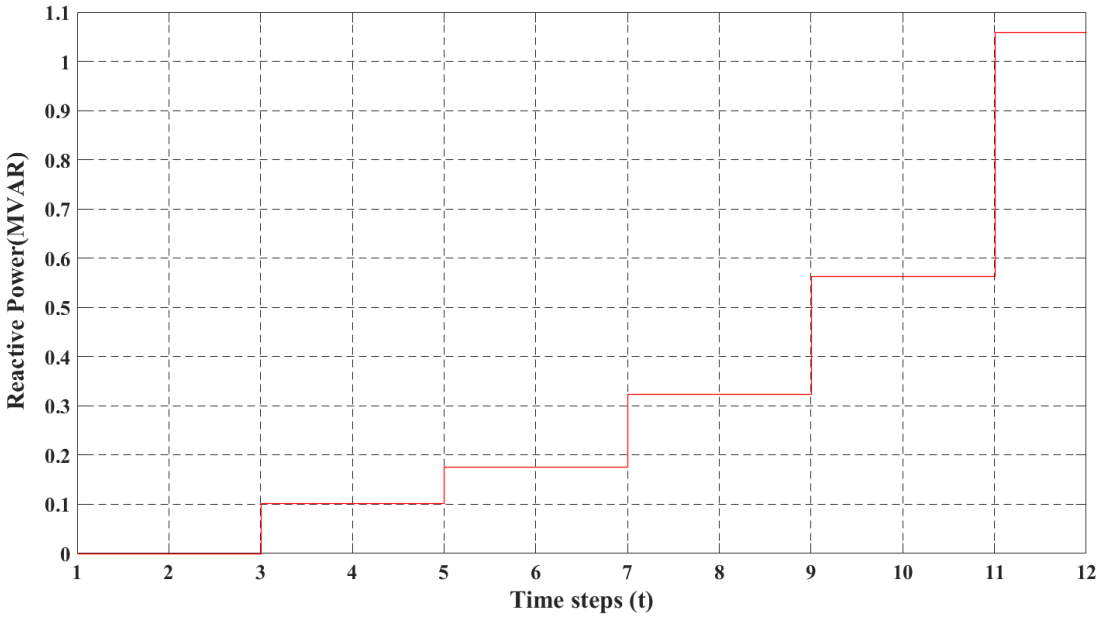


Figure 4.11: Reactive Power of Battery

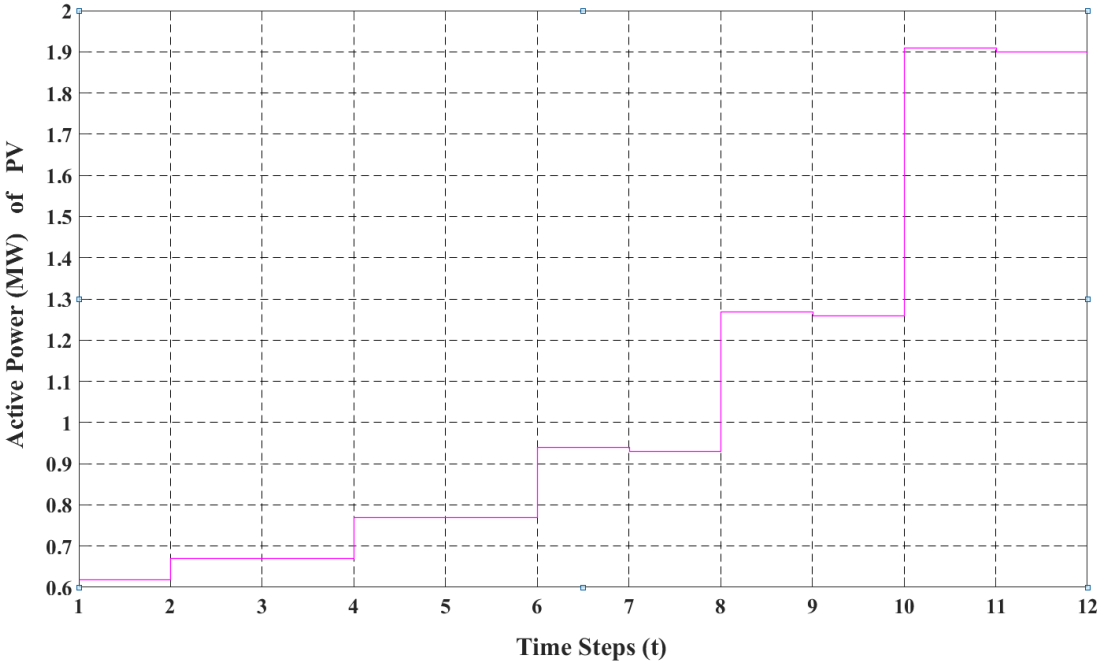


Figure 4.12: Power generation of PV system

Chapter 5

Conclusion and Future work

The contribution of this thesis is to implement a multiagent system for distributed voltage control system application for an islanded microgrid. This technique helps to solve voltage regulation problem in a multiagent environment. The proposed method effectively mitigates the challenges of voltage regulation in droop-controlled islanded microgrid and eliminates the need for a central controller. In this multiagent control system agents communicate with each other to mitigate the voltage regulation problem. Tiva-C TM4C123GH6PM microcontroller has been used for the design of this control system. Tiva-C microcontrollers are easy to program. Tiva-C microcontrollers provide fast processing and reduce memory requirements, they are integrated with many features that are useful to implement embedded applications. To implement this multiagent control system we learned and studied the Opal-RT technologies real time simulator. Opal-RT real time simulator is used to create micorgrid model. Real time simulator is interfaced with two Tiva-C microcontrollers that

are working as two agent in the proposed multiagent design. Physical wires are used to connect Opal-RT analog ports with Tiva-C micontrollers ports. Different models were run to test the experimental setup. The results of case studies validate the experimental setup for multiagent control system.

5.1 Future work

- We implemented the design with two agents. It can be expand to n number of agents.
- The proposed multiagent system can be used for potential application of smart grid like self healing, cyber-physical security , demand response.
- Further testing of communication aspects like latency, message delay, lost of message and medium of communication.

Bibliography

[1] RE, Renewable Energy. [Online]. Available: https://en.wikipedia.org/wiki/Renewable energy.

[2] S. R. Bull, "Renewable energy today and tomorrow, *Proc. IEEE*, vol. 89, no. 8, pp. 1216-1226, Aug. 2001.

[3] B. Wu, Y. Lang, N. Zagari, and S. Kouro, "Power Conversion and Control of Wind Energy Systems," *Piscataway, NJ, USA: Wiley*, 2011.

[4] M. G. Simoes , F. A. Farret, "Renewable Energy Systems," *Boca Raton, FL, USA: CRC*, 2004.

[5] T. Teodorescu, M. Liserre, and P. Rodriguez, "Grid Converters for Photovoltaic and Wind Power Systems," *Piscataway, NJ,USA: Wiley*, 2011.

[6] F. A. Farret, M. G. Simoes, " Integration of Alternate Sources of Energy," *Piscataway, NJ, USA: Wiley*, 2011.

[7] H. Abu-Rub, M. Malinowski, and K. Al-Haddad, " Power Electronics for Renewable Energy Systems," *Transportation and Industrial Applications. Hoboken, NJ, USA: Wiley*, 2014.

[8] E. J. Coster, J. M. A. Myrzik, B. Kruimer, and W. L. Kling, "Integration issues of distributed generation in distribution grids," *Proc. IEEE*, vol. 99, no. 1, pp. 28-39, Jan. 2011.

[9] H. Al Haj Hassan, A. Pelov, and L. Nuaymi, "Integrating cellular networks, smart grid, and renewable energy: Analysis, architecture, and challenges," *IEEE Access*, vol. 3, pp. 2755-2770, 2015.

[10] P. Mahat, Z. Chen, B. Bak-Jensen, and C. L. Bak, "A simple adaptive overcurrent protection of distribution systems with distributed generation," *IEEE Trans. Smart Grid*, vol. 2, no. 3, pp. 428-437, Sept. 2011

[11] Renewable Energy Hits a Milestone, Fortune, p. 15, Aug. 2017.

[12] R. Karki, P. Hu, and R. Billinton, "A Simplified Wind Power Generation Model for Reliability Evaluation, *IEEE Tran. Energy Conversion*, vol. 21, pp. 533-540, June 2006.

[13] H.R. Seo, G.H. Kim, S.Y. Kim, N. Kim, H.G. Lee, C. Hwang, M. Park, and I.K. Yu, " Power quality control strategy for grid-connected renewable energy sources

using PV array and supercapacitor," *International Conference on Electrical Machines and Systems (ICEMS)*, Dec. 2010.

[14] X.Yu, C.Cecati, T.Dillon, "The New Frontier of Smart Grids," *IEEE Industrial Electronics Magazine* , Sept. 2011.

[15] H.GHARAVI, R.GHAFURIAN, "Smart Grid: The Electric Energy System of the Future," *Proceedings of the IEEE*, vol. 99, no. 6, June. 2011.

[16] Smart Grid. [Online]. Available: https:// en.wikipedia.org/wiki/Smart grid.

[17] J. Wang, A. Q. Huang, W. Sung, Y. Liu, and B. J. Baliga, "Smart grid technologies, *IEEE Ind. Electron. Mag.*, vol. 3, no. 2, pp. 16-23, Jun. 2009.

[18] B.K. Bose, "Power Electronics, Smart Grid and Renewable Energy Systems," *Proceedings of the IEEE*, vol. 105, no. 11, November. 2017.

[19] ME. El-Hawary, "The smart grid state of the art and future trends," *Electric Power Components and Systems*, vol. 42, no.3-4, 2014.

[20] Department of Energy, Request for information,*available at: http://www.gpo.gov/fdsys/pkg/fr*,2010

[21] R.H.Lasseter, "Microgrids," *IEEE* , 2002.

[22] C.H.Cho, J.H.Jeon, J.Y.Kim, S.Kwon, K.Park and S.Kim, "Active synchronizing control a microgrid," *IEEE Trans. Power Electron*, vol. 26, no.12, 2011.

[23] H.B.Puttgen, P.R.MacGregor and F.C.Lambert, "Distributed generation: semantic hype or the dawn of a new era?," *IEEE Power and Energy Magazine*, vol.1, no.1, pp. 22-29, Jan-Feb 2003.

[24] Anestis. G. Anastasiadis, Antonis. G. Tsikalakis, Dan Nikos. D.Hatziargyriou, "Environmental Benefits From DG Operation When Network Losses are Taken Into Account," *Distress Conference Nicosia Cyprus*, Dec. 2009

[25] N. Hatziargyriou, H. Asano, R.Iravani, D.C. Marnay, "Microgrid, An Overview of Ongoing Research, Development, and Demonstration Projects," *IEEE Power and Energy Magazine*, July/August 2007.

[26] B.S. Hartono, R. Setiabudy, "Review of microgrid technology, *QiR (Quality in Research)*, 2013.

[27] Z. Shuai , Y. Sun , Z.John Shen ,W. Tian , C.Tu , Y. Li , X. Yin, "Microgrid stability: Classification and a review," *Renewable and Sustainable Energy Reviews*, pp.167-179, 2016.

[28] Wang Dan. "Modeling and stability simulation of distributed generation system," *PhD thesis. Tianjin, China: Tianjin University*, 2008.

[29] B.Kroposki, T.Basso,R.DeBlasio, "Microgrid Standards and Technologies," *IEEE*, August 2008.

[30] Michael Schumacher, "Objective Coordination in Multi-Agent System Engineering," *www.springer.com*, pp 9-32, 2001.

[31] L.Monostori,J.Vncza,S.R.T.Kumara, "Agent-Based Systems for Manufacturing," *Annals of the CIRP*, vol. 55, no. 2, 2006.

[32] M. Luck, P. McBurney, O. Shehory, " Agent Technology: Computing as Interaction. A Roadmap for Agent-Based Computing, "*AgentLink III, http://www.agentlink.org/roadmap/al3rm.pdf* , 2005.

[33] P. Wooldridge, "An Introduction to Multiagent Systems," *Addison-Wesley, Reading, MA*, 2000.

[34] G. Weiss, " Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence," *The MIT Press*, 1999.

[35] Dr.Mevludin Glavic, "Agents and Multi-Agent Systems: A Short Introduction for Power Engineers," *Technical Report*, May 2006.

[36] P.Stone, M.Veloso, "Multiagent Systems: A Survey from a Machine Learning Perspective," *Autonomous Robots*, vol. 8, no. 3, pp. 345-383, June 2000.

[37] T.Logenthiran, D.Srinivasan, AM.Khambadkone, HN.Aung, "Multiagent System for Real-Time Operation of a Microgrid in Real-Time Digital Simulator," *IEEE Transaction on Smart Grid*, vol. 3, no. 2,June 2012.

[38] RTDS, Real-Time Digital Simulator [Online]. Available: http://www.rtds.com.

[39] Java Agent Development Environment (JADE) [Online]. Available: http://jade.tilab.com.

[40] FIPA, The Foundation for Intelligent Physical Agents standards [Online]. Available: http://www.fipa.org.

[41] T.Logenthiran, RT. Naayagi,WL.Woo, VT.Phan,K.Abidi, "Intelligent Control System for Microgrids Using Multiagent System," *IEEE Journal of ESTPE*, vol. 3, no. 4, Dec 2015.

[42] L.Nachabe,M.Girod-Genet,B.ElHassan,J.Jammas, "M-health application for Neonatal Incubator signals monitoring through a CoAP-based multi-agent system," *IEEE International Conference on Advances in Biomedical Engineering (ICABME)*, 2015.

[43] Z.Shelby , K.Hartke,C. Bormann, "Constrained Application Protocol (CoAP)," *draft-ietf-core-coap-18*, June 2013.

[44] M Kovatsch, S.Duquennoy,A.Dunkels, "A low-power CoAP for Contiki," *In the proceeding of IEEE 8th International Conference Mobile Adhoc and Sensor Systems (MASS)*, 2011.

[45] https://github.com/dapaulid/JCoAP.

[46] M.Morales, "An Introduction to the Tiva C Series Platform of Microcontrollers," *Texas Instruments*, April 2013.

[47] Texas Instruments, "Tiva TM4C123GH6PM Microcontroller Data sheet," , 2014.

[48] J.Blanger, P.Venne and J.-N.Paquin, "The What, Where and Why of Real-Time Simulation," *Opal-RT Technologies*.

[49] OPAL-RT Technologies, "OP5600 The Real-Time Simulator User Guide," *www.Opal-RT.com*, 2011.

[50] N. El-Taweel,H. E. Z. Farag," Multi-agent Coordination of Distributed Energy Storage Systems for Mitigating Renewable Energy Resources High Ramp-Rate Issues, IEEE Electrical Power and Energy Conference (EPEC), pp. 1-6, 2017.

Appendix A

Code for Agent One and Two

A.1 Agent One

```
#include <stdint.h>
#include <stdbool.h>
#include <math.h>
#include <stdio.h>
#include<stdlib.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/hw_ints.h"
#include "inc/hw_ssi.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/timer.h"
#include "driverlib/ssi.h"
#include "driverlib/pin_map.h"

volatile int upper_flag=0;
volatile int flag;
volatile int flags=0;
volatile float S=0;
volatile float Sm=0;
volatile float S1=0;
volatile float Sm1=0;
float Vpzc_limit_recieved;
float Delta_Vpzc_recieved;
float Vj_found_recieved;
uint32_t RecData[10];
uint32_t RData;
uint32_t RecAB;
uint32_t RecGNG;
uint32_t SData1,SData2;
uint32_t Data1 ,Data2,Data3,Data4;
uint32_t D1,D2,D3,D4;
```

```
uint32_t RD1,RD2;

volatile int head=0;
volatile int tail=0;
uint32_t Value[6];

void main(void)
{
  uint32_t ui32Period;
    uint32_t alpha;
    alpha=1;
    uint32_t beta;
    beta=3;
    uint32_t good;
    good=1;
    uint32_t nogood;
    nogood=2;
  uint8_t pindata;
    int loc=0;
    float Vj[3]={};
    float Vi[3]={};
    float Vpzc1[3]={};
    float V[3][3]={{Vj},{Vi},{Vpzc1}};
    float P[2]={};
    float Q[2]={};
    float V_upper=1.05;
    float e=0.005;
    float minPF=0.89;
    float Smax=0.5;
    float P_si=0.068885+0.012453;
    float Q_si=0.056928+0.012453;
    float R_ij=0.008026+0.004558+0.012453;
    float X_ij=0.010716+0.003574+0.012453;
    float R_ipzc1=0.0459;
    float X_ipzc1=0.045;
    float PV_senstivity=0.078;

   SysCtlClockSet(SYSCTL_SYSDIV_4|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
   GPIOPinTypeGPIOInput(GPIO_PORTB_BASE, GPIO_PIN_1);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);

   GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_5);
   GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_1);
   GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_0);
   GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_3);
   GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_2);
   GPIOPinTypeADC(GPIO_PORTB_BASE, GPIO_PIN_4);

   while(!SysCtlPeripheralReady(SYSCTL_PERIPH_ADC0))
   {

    }

   ADCSequenceDisable(ADC0_BASE, 1);
   ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);

   ADCSequenceStepConfigure(ADC0_BASE,1, 0, ADC_CTL_CH3);
   ADCSequenceStepConfigure(ADC0_BASE,1, 1, ADC_CTL_CH2);
```

```
    ADCSequenceStepConfigure(ADC0_BASE,1, 2, ADC_CTL_CH0);
    ADCSequenceStepConfigure(ADC0_BASE,1, 3, ADC_CTL_CH10);
    ADCSequenceStepConfigure(ADC0_BASE,1, 4, ADC_CTL_CH8 | ADC_CTL_IE | ADC_CTL_END);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI0);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI1);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI3);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);

     ui32Period = (SysCtlClockGet() /100000) / 2;
     TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);

     IntEnable(INT_TIMER0A);
     TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
     IntMasterEnable();
     TimerEnable(TIMER0_BASE, TIMER_A);
//Enable SSI3 module
     GPIOPinConfigure(GPIO_PD0_SSI3CLK);
     GPIOPinConfigure(GPIO_PD1_SSI3FSS);
   GPIOPinConfigure(GPIO_PD2_SSI3RX);
     GPIOPinConfigure(GPIO_PD3_SSI3TX);
     GPIOPinTypeSSI(GPIO_PORTD_BASE,GPIO_PIN_3|GPIO_PIN_2|GPIO_PIN_1|GPIO_PIN_0);
// Configure the SSI0.
SSIConfigSetExpClk(SSI3_BASE, SysCtlClockGet(),
SSI_FRF_MOTO_MODE_0, SSI_MODE_MASTER, 1000000, 16);
     SSIEnable(SSI3_BASE);
//Enable SSI0 module
     GPIOPinConfigure(GPIO_PA2_SSI0CLK);
     GPIOPinConfigure(GPIO_PA3_SSI0FSS);
     GPIOPinConfigure(GPIO_PA5_SSI0TX);
     GPIOPinConfigure(GPIO_PA4_SSI0RX);
     GPIOPinTypeSSI(GPIO_PORTA_BASE,GPIO_PIN_5|GPIO_PIN_4|GPIO_PIN_3|GPIO_PIN_2);
// Configure the SSI0.
SSIConfigSetExpClk(SSI0_BASE, SysCtlClockGet(),
SSI_FRF_MOTO_MODE_0, SSI_MODE_MASTER, 1000000, 16);
     SSIEnable(SSI0_BASE);
 // Enable the SSI1 module.
     GPIOPinConfigure(GPIO_PF2_SSI1CLK);
     GPIOPinConfigure(GPIO_PF3_SSI1FSS);
  HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
    HWREG(GPIO_PORTF_BASE + GPIO_O_CR) = 0x1;
     GPIOPinConfigure(GPIO_PF0_SSI1RX);
     GPIOPinConfigure(GPIO_PF1_SSI1TX);
     GPIOPinTypeSSI(GPIO_PORTF_BASE,GPIO_PIN_3|GPIO_PIN_2|GPIO_PIN_1|GPIO_PIN_0);
// Configure the SSI1.
SSIConfigSetExpClk(SSI1_BASE, SysCtlClockGet(),
SSI_FRF_MOTO_MODE_0, SSI_MODE_SLAVE, 10000, 16);
     SSIEnable(SSI1_BASE);
     SSIIntEnable(SSI1_BASE, SSI_RXTO);
     IntEnable(INT_SSI1);

     while(SSIDataGetNonBlocking(SSI0_BASE, &RecData[0]))
```

```
        {
          }
    //-------------------------------------------------------------------------------
      //Initializing
      float DP=0;
      float DQ=0;
      float Curtailed_PV=0;
      float DVj=0;
      float DP_nominator=0;
      float DP_denominator=0;
      float Vpzc_limit=0;
      float Delta_Vpzc=0;
      float P_max=0;
      float Q_max=0;
      float DVj_found=0;
      float Vj_found=0;
      void *P3,*P1,*P2;

      ADCSequenceEnable(ADC0_BASE, 1);
      ADCIntClear(ADC0_BASE, 1);

    pindata=GPIOPinRead(GPIO_PORTB_BASE, GPIO_PIN_1);
    while (pindata != 2) {
     pindata=GPIOPinRead(GPIO_PORTB_BASE, GPIO_PIN_1);
    }

while(1)
 {
      while(flag==0){}
      flag=0;
 //-------------------------------------------------------------------------------
         ADCProcessorTrigger(ADC0_BASE, 1);
         while (!ADCIntStatus(ADC0_BASE, 1, false))
          {
          }
          ADCIntClear(ADC0_BASE, 1);
          ADCSequenceDataGet(ADC0_BASE, 1, Value);

Vj[1]=Vj[0];
Vi[1]=Vi[0];
Vpzc1[1]=Vpzc1[0];

Vj[1]=Value[0] & 0xFFFF;

Vi[1]=Value[1] & 0xFFFF;

Vpzc1[1]=Value[2]& 0xFFFF;

P[0]=Value[3]& 0xFFFF;

Q[0]=Value[4]& 0xFFFF;

//-------------------------------------------------------------------------
// estimated PV, BESS and PZC voltage at next time step t+1

  Vj[2]= Vj[1]+(Vj[1]-Vj[0]);
  Vi[2]= Vi[1]+(Vi[1]-Vi[0]);
  Vpzc1[2]=Vpzc1[1]+(Vpzc1[1]-Vpzc1[0]);
//-------------------------------------------------------------------------
//###Code: Estimating Next Step paramtes and Checking for Violation###
```

```
  int c;
    int n;
    float Violation_upper[3]={0};

    for(n=0;n<=2;n++)
    {

      V[n][2]=V[n][1]+(V[n][1]-V[n][0]);

    }

   float minimum= Violation_upper[0];

    for(n=0; n<=2; n++)
    {
       if(V[n][2] > V_upper)
       {
           upper_flag=1;
             Violation_upper[n]=V_upper-V[n][2]-e;
         }
       }
 if(upper_flag==0)
  {
  SSIDataPut(SSI0_BASE,alpha);
  while(SSIBusy(SSI0_BASE))
    {
      }

    while(head==tail){}
    RecAB=RecData[head];
    head=(head+1)%10;

   if(RecAB==1)
     {
// update
    SSIDataPut(SSI3_BASE,DP);
       while(SSIBusy(SSI3_BASE))
        {
       }
       SSIDataPut(SSI3_BASE,DQ);
       while(SSIBusy(SSI3_BASE))
        {
         }
     SSIDataPut(SSI3_BASE,Curtailed_PV);
     while(SSIBusy(SSI3_BASE))
       {
        }
     }
   }
     else if(RecAB==3)
     {
//### Code : When agent recive a request from adjacent Neighbour###
     while(head==tail){}
     D1=RecData[head];
     head=(head+1)%10;

     while(head==tail){}
     D2=RecData[head];
```

```
      head=(head+1)%10;

      while(head==tail){}
      D3=RecData[head];
      head=(head+1)%10;

      while(head==tail){}
      D4=RecData[head];
      head=(head+1)%10;
      RD1=(D1<<16)|D2;
      RD2=(D3<<16)|D4;

      float V_upper_pzc=0;
   Vpzc_limit_recieved=*((float*)&RD1);
      Delta_Vpzc_recieved=*((float*)&RD2);
   V_upper_pzc=Vpzc_limit_recieved;
   Vpzc1[2]=Vpzc1[1]+(Vpzc1[1]-Vpzc1[0]);
   DVj=V_upper_pzc-e-Vpzc1[2]+Delta_Vpzc_recieved;
DP_nominator=
(Vpzc1[2]*DVj)-((P[0]*tan(acos(minPF))-Q[0])*((V[1][2]*Q_si)-X_ipzc1));
DP_denominator=
(V[1][2]*P_si)-(R_ipzc1)+(tan(acos(minPF))*((V[1][2]*Q_si)-X_ipzc1));
DP=DP_nominator/DP_denominator;
DQ=((P[0]+DP)*tan(acos(minPF)))-Q[0];

      S1=pow((DP+P[0]),2)+pow((DQ+Q[0]),2);
      Sm1=pow(Smax,2);
   if( S1 > Sm1)
    {
            P_max=sqrt(pow(Smax,2)/(1+pow((tan(acos(minPF))),2)));
           Q_max=P_max*tan(acos(minPF));
           DP=-P_max-P[0];
           DQ=-Q_max-Q[0];
DVj_found=
(1/Vpzc1[2])*((V[1][2]*((P_si*DP)+(Q_si*DQ)))-(DP*R_ipzc1)-(DQ*X_ipzc1));
Vj_found=Vpzc1[2]+DVj_found+Delta_Vpzc_recieved;

            void *P4;
            P4=&Vj_found;
            SData1=*((uint32_t *)P4);
            Data1=(SData1>>16)&0x0000FFFF;
            Data2=SData1 & 0x0000FFFF;

      SSIDataPut(SSI0_BASE,nogood);
      while(SSIBusy(SSI0_BASE))
       {
        }
      SSIDataPut(SSI0_BASE,Data1);
      while(SSIBusy(SSI0_BASE))
        {
         }
      SSIDataPut(SSI0_BASE,Data2);
      while(SSIBusy(SSI0_BASE))
       {
          }
//update
       SSIDataPut(SSI3_BASE,DP);
       while(SSIBusy(SSI3_BASE))
```

```
        {
        }
        SSIDataPut(SSI3_BASE,DQ);
        while(SSIBusy(SSI3_BASE))
         {
          }
        SSIDataPut(SSI3_BASE,Curtailed_PV);
        while(SSIBusy(SSI3_BASE))
         {
          }
       }
//-------------------------------------------------
     else if( S1 < Sm1)
      {
       SSIDataPut(SSI0_BASE,good);
       while(SSIBusy(SSI0_BASE))
           {
          }
 //update
       SSIDataPut(SSI3_BASE,DP);
       while(SSIBusy(SSI3_BASE))
        {
       }
       SSIDataPut(SSI3_BASE,DQ);
       while(SSIBusy(SSI3_BASE))
        {
         }
       SSIDataPut(SSI3_BASE,Curtailed_PV);
       while(SSIBusy(SSI3_BASE))
        {
         }
      }

     }
  }
if(upper_flag==1)
//###  Code: Calculating Delta P "DP" and Delta Q "DQ" ###
 {

     minimum= Violation_upper[0];
     for(c=1; c<=2; c++)
      {
        if (Violation_upper[c] < minimum)
         {
           minimum  = Violation_upper[c];
           loc = c;
         }
       }

DVj=V_upper-e-V[loc][2];
DP_nominator=
((V[loc][2]*DVj)-((P[0]*tan(acos(minPF))-Q[0])*((V[1][2]*Q_si)-X_ij)));
DP_denominator=
(V[1][2]*P_si)-(R_ij)+(tan(acos(minPF))*((V[1][2]*Q_si)-X_ij));
DP=DP_nominator/DP_denominator;
```

```
DQ=((P[0]+DP)*tan(acos(minPF)))-Q[0];
//-------------------------------------------------------------------
//### Code: Checking feasability , if feasible apply to driver,
else find possible solution and send request to neighbor agent ###

     float Sen_pzc=0;
     S=pow((DP+P[0]),2)+pow((DQ+Q[0]),2);
   Sm=pow(Smax,2);

     if (S < Sm)
      {
     SSIDataPut(SSI0_BASE,alpha);
     while(SSIBusy(SSI0_BASE))
       {
        }
     while(head==tail){}
     RecAB=RecData[head];
     head=(head+1)%10;
     if(RecAB==1)
      {
//update
       SSIDataPut(SSI3_BASE,DP);
       while(SSIBusy(SSI3_BASE))
        {
       }
       SSIDataPut(SSI3_BASE,DQ);
       while(SSIBusy(SSI3_BASE))
        {
         }
      SSIDataPut(SSI3_BASE,Curtailed_PV);
      while(SSIBusy(SSI3_BASE))
        {
         }

      }
     else if(RecAB==3)
      {
//### Code: When agent recive a request from adjacent Neighbour ####
     while(head==tail){}
     D1=RecData[head];
     head=(head+1)%10;

     while(head==tail){}
     D2=RecData[head];
     head=(head+1)%10;

     while(head==tail){}
     D3=RecData[head];
     head=(head+1)%10;

   while(head==tail){}
   D4=RecData[head];
   head=(head+1)%10;

   RD1=(D1<<16)|D2;
   RD2=(D3<<16)|D4;

    float V_upper_pzc=0;
```

```
   Vpzc_limit_recieved=*((float *)&RD1);
   Delta_Vpzc_recieved=*((float *)&RD2);
        V_upper_pzc=Vpzc_limit_recieved;
        Vpzc1[2]=Vpzc1[1]+(Vpzc1[1]-Vpzc1[0]);
DVj=V_upper_pzc-e-Vpzc1[2]+Delta_Vpzc_recieved;
DP_nominator=
 (Vpzc1[2]*DVj)-((P[0]*tan(acos(minPF))-Q[0])*((V[1][2]*Q_si)-X_ipzc1));
DP_denominator=
 (V[1][2]*P_si)-(R_ipzc1)+(tan(acos(minPF))*((V[1][2]*Q_si)-X_ipzc1));
DP=DP_nominator/DP_denominator;
DQ=((P[0]+DP)*tan(acos(minPF)))-Q[0];

        S1=pow((DP+P[0]),2)+pow((DQ+Q[0]),2);
        Sm1=pow(Smax,2);
      if( S1 > Sm1)
        {
P_max=sqrt(pow(Smax,2)/(1+pow((tan(acos(minPF))),2)));
Q_max=P_max*tan(acos(minPF));
DP=-P_max-P[0];
DQ=-Q_max-Q[0];
DVj_found=
(1/Vpzc1[2])*((V[1][2]*((P_si*DP)+(Q_si*DQ)))-(DP*R_ipzc1)-(DQ*X_ipzc1));
Vj_found=Vpzc1[2]+DVj_found+Delta_Vpzc_recieved;

      void *P4;
      P4=&Vj_found;
      SData1=*((uint32_t *)P4);
           Data1=(SData1>>16)&0x0000FFFF;
           Data2=SData1 & 0x0000FFFF;

        SSIDataPut(SSI0_BASE,nogood);
        while(SSIBusy(SSI0_BASE))
         {
          }
        SSIDataPut(SSI0_BASE,Data1);
        while(SSIBusy(SSI0_BASE))
         {
            }
        SSIDataPut(SSI0_BASE,Data2);
        while(SSIBusy(SSI0_BASE))
         {
            }
//update
        SSIDataPut(SSI3_BASE,DP);
        while(SSIBusy(SSI3_BASE))
         {
        }
        SSIDataPut(SSI3_BASE,DQ);
        while(SSIBusy(SSI3_BASE))
         {
          }
       SSIDataPut(SSI3_BASE,Curtailed_PV);
       while(SSIBusy(SSI3_BASE))
         {
          }
//-------------------------------------------------------------------------
   else if( S1 < Sm1)
```

```
      {
     SSIDataPut(SSI0_BASE,good);
     while(SSIBusy(SSI0_BASE))
      {
        }
//update
       SSIDataPut(SSI3_BASE,DP);
       while(SSIBusy(SSI3_BASE))
        {
       }
       SSIDataPut(SSI3_BASE,DQ);
       while(SSIBusy(SSI3_BASE))
         {
          }
      SSIDataPut(SSI3_BASE,Curtailed_PV);
      while(SSIBusy(SSI3_BASE))
         {
          }
      }
       }
      }
     else if (S > Sm)
      {
 //    means not feasible solution
 P_max=sqrt(pow(Smax,2)/(1+pow((tan(acos(minPF))),2)));
 Q_max=P_max*tan(acos(minPF));
DP=-P_max-P[0];
DQ=-Q_max-Q[0];
DVj_found=
(1/V[loc][2])*((V[1][2]*((P_si*DP)+(Q_si*DQ)))-(DP*R_ij)-(DQ*X_ij));
Vj_found=V[loc][2]+DVj_found;
Sen_pzc=(Vpzc1[1]-Vpzc1[0])/(Vj[1]-Vj[0]);
Vpzc_limit=Vpzc1[1]+(V_upper-e-Vj[1])*Sen_pzc;
Delta_Vpzc=Sen_pzc*(DVj_found);

     void *P1,*P2;
     P1=&Vpzc_limit;
     P2=&Delta_Vpzc;
   SData1=*((uint32_t*)P1);
     SData2=*((uint32_t*)P2);
     Data1=(SData1>>16) & 0x0000FFFF;
     Data2=SData1 & 0x0000FFFF;
     Data3=(SData2>>16) & 0x0000FFFF;
     Data4=SData2 & 0x0000FFFF;

   SSIDataPut(SSI0_BASE,beta);
   while(SSIBusy(SSI0_BASE))
      {
       }

   SSIDataPut(SSI0_BASE,Data1);
   while(SSIBusy(SSI0_BASE))
      {
       }

   SSIDataPut(SSI0_BASE,Data2);
   while(SSIBusy(SSI0_BASE))
```

```
      {
       }
   SSIDataPut(SSI0_BASE,Data3);
      while(SSIBusy(SSI0_BASE))
       {
        }
      SSIDataPut(SSI0_BASE,Data4);
      while(SSIBusy(SSI0_BASE))
       {
        }
      while(head==tail){}
      RecAB=RecData[head];
      head=(head+1)%10;
      if(RecAB==1)
       {
      while(head==tail){}
      RecGNG=RecData[head];
      head=(head+1)%10;
        if(RecGNG==1)
        {
//update
         SSIDataPut(SSI3_BASE,DP);
         while(SSIBusy(SSI3_BASE))
          {
         }
         SSIDataPut(SSI3_BASE,DQ);
         while(SSIBusy(SSI3_BASE))
          {
           }
        SSIDataPut(SSI3_BASE,Curtailed_PV);
        while(SSIBusy(SSI3_BASE))
          {
           }
         }
         else  if(RecGNG==2)
          {
         while(head==tail){}
         D1=RecData[head];
         head=(head+1)%10;
         while(head==tail){}
         D2=RecData[head];
         head=(head+1)%10;
         RD1=(D1<<16)|D2;
//#### Code: Curtailment of PV Power when an agent recive a no-good msg.####
   float Senstivity_PV_pzc1=0;
   float PV_volt_affect_by_negihbours_action=0;
   float Vj_found_recieved;

Vj_found_recieved=*((float*)&RD1);
Senstivity_PV_pzc1=(Vj[1]-Vj[0])/(Vpzc1[1]-Vpzc1[0]);
PV_volt_affect_by_negihbours_action=
Senstivity_PV_pzc1*(Vj_found_recieved-Vpzc1[2]);
Curtailed_PV=
(V_upper-(Vj[2]+PV_volt_affect_by_negihbours_action)-e)*PV_senstivity;
//-------------------------------------------------------------------------
```

```
//update
        SSIDataPut(SSI3_BASE,DP);
        while(SSIBusy(SSI3_BASE))
          {
         }
        SSIDataPut(SSI3_BASE,DQ);
        while(SSIBusy(SSI3_BASE))
          {
           }
        SSIDataPut(SSI3_BASE, Curtailed_PV);
        while(SSIBusy(SSI3_BASE))
          {
           }

        }
      }
     else  if(RecAB==3)
       {
       SSIDataPut(SSI0_BASE, beta);
       while(SSIBusy(SSI0_BASE))
        {
         }
//update
     SSIDataPut(SSI3_BASE,DP);
          while(SSIBusy(SSI3_BASE))
             {
           }
          SSIDataPut(SSI3_BASE,DQ);
          while(SSIBusy(SSI3_BASE))
            {
             }
          SSIDataPut(SSI3_BASE, Curtailed_PV);
          while(SSIBusy(SSI3_BASE))
            {
             }

          }

       }
      }
//-------------------------------------------------------------------------
  }
}

void Timer0IntHandler(void)
  {
  // Clear the timer interrupt
  TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
  flag=1;
   }

void SSI1IntHandler(void)
  {
   // Clear interrupts.
    SSIIntClear(SSI1_BASE,SSI_RXTO);
    SSIDataGet(SSI1_BASE, &RData);
    RecData[tail]=RData;
```

```
    tail=(tail+1)%10;
    }
```

A.2 Agent Two

```
#include <stdint.h>
#include <stdbool.h>
#include <math.h>
#include <stdio.h>
#include<stdlib.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/hw_ints.h"
#include "inc/hw_ssi.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/timer.h"
#include "driverlib/ssi.h"
#include "driverlib/pin_map.h"

volatile int upper_flag=0;
volatile int flag;
volatile int flags=0;
volatile float S=0;
volatile float Sm=0;
volatile float S1=0;
volatile float Sm1=0;
float Vpzc_limit_recieved;
float Delta_Vpzc_recieved;
float Vj_found_recieved;
uint32_t RecData[10];
uint32_t RData;
uint32_t RecAB;
uint32_t RecGNG;
uint32_t SData1,SData2;
uint32_t Data1 ,Data2,Data3,Data4;
uint32_t D1,D2,D3,D4;
uint32_t RD1,RD2;

volatile int head=0;
volatile int tail=0;
uint32_t Value[6];


void main(void)
{
  uint32_t ui32Period;
    uint32_t alpha;
    alpha=1;
    uint32_t beta;
    beta=3;
    uint32_t good;
    good=1;
    uint32_t nogood;
    nogood=2;
  uint8_t pindata;
    int loc=0;
    float Vj[3]={1.0453,1.0529, 0};
    float Vi[3]={1.0209,1.0278, 0};
```

```
    float Vpzc1[3]={1.0308, 1.0361,0};
    float V[3][3]={{Vj},{Vi},{Vpzc1}};
    float P[2]={-0.15-0.125,0};
    float Q[2]={-0.141,0};
    float V_upper=1.05;
    float e=0.005;
    float minPF=0.89;
    float Smax=0.5;
    float P_si=0.018992+0.012453;
  float Q_si=0.013946+0.012453;
  float R_ij=0.02848;
  float X_ij=0.024782;
  float X_ipzc1=0.022845;
  float PV_senstivity=0.078;

   SysCtlClockSet(SYSCTL_SYSDIV_4|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
   GPIOPinTypeGPIOInput(GPIO_PORTB_BASE, GPIO_PIN_1);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);

   GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_5);
   GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_1);
   GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_0);
   GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_3);
   GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_2);
   GPIOPinTypeADC(GPIO_PORTB_BASE, GPIO_PIN_4);

   while(!SysCtlPeripheralReady(SYSCTL_PERIPH_ADC0))
   {

    }

   ADCSequenceDisable(ADC0_BASE, 1);
   ADCSequenceConfigure(ADC0_BASE, 1, ADC_TRIGGER_PROCESSOR, 0);

   ADCSequenceStepConfigure(ADC0_BASE,1, 0, ADC_CTL_CH3);
   ADCSequenceStepConfigure(ADC0_BASE,1, 1, ADC_CTL_CH2);
   ADCSequenceStepConfigure(ADC0_BASE,1, 2, ADC_CTL_CH0);
   ADCSequenceStepConfigure(ADC0_BASE,1, 3, ADC_CTL_CH10);
   ADCSequenceStepConfigure(ADC0_BASE,1, 4, ADC_CTL_CH8 | ADC_CTL_IE | ADC_CTL_END);

   SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI0);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI1);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_SSI3);
   SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);

   SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
   TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);

    ui32Period = (SysCtlClockGet() /100000) / 2;
    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);

    IntEnable(INT_TIMER0A);
    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
    IntMasterEnable();
    TimerEnable(TIMER0_BASE, TIMER_A);
//Enable SSI3 module
```

```
    GPIOPinConfigure(GPIO_PD0_SSI3CLK);
    GPIOPinConfigure(GPIO_PD1_SSI3FSS);
  GPIOPinConfigure(GPIO_PD2_SSI3RX);
    GPIOPinConfigure(GPIO_PD3_SSI3TX);
    GPIOPinTypeSSI(GPIO_PORTD_BASE,GPIO_PIN_3|GPIO_PIN_2|GPIO_PIN_1|GPIO_PIN_0);
  // Configure the SSI0.
SSIConfigSetExpClk(SSI3_BASE,SysCtlClockGet(),
SSI_FRF_MOTO_MODE_0, SSI_MODE_MASTER,1000000,16);
    SSIEnable(SSI3_BASE);
//Enable SSI0 module
    GPIOPinConfigure(GPIO_PA2_SSI0CLK);
    GPIOPinConfigure(GPIO_PA3_SSI0FSS);
    GPIOPinConfigure(GPIO_PA5_SSI0TX);
    GPIOPinConfigure(GPIO_PA4_SSI0RX);
    GPIOPinTypeSSI(GPIO_PORTA_BASE,GPIO_PIN_5|GPIO_PIN_4|GPIO_PIN_3|GPIO_PIN_2);
  // Configure the SSI0.
SSIConfigSetExpClk(SSI0_BASE,SysCtlClockGet(),
SSI_FRF_MOTO_MODE_0,SSI_MODE_MASTER,1000000,16);
    SSIEnable(SSI0_BASE);
 // Enable the SSI1 module.
    GPIOPinConfigure(GPIO_PF2_SSI1CLK);
    GPIOPinConfigure(GPIO_PF3_SSI1FSS);
  HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
    HWREG(GPIO_PORTF_BASE + GPIO_O_CR) = 0x1;
    GPIOPinConfigure(GPIO_PF0_SSI1RX);
    GPIOPinConfigure(GPIO_PF1_SSI1TX);
    GPIOPinTypeSSI(GPIO_PORTF_BASE,GPIO_PIN_3|GPIO_PIN_2|GPIO_PIN_1|GPIO_PIN_0);
// Configure the SSI1.
    SSIConfigSetExpClk(SSI1_BASE, SysCtlClockGet(),
SSI_FRF_MOTO_MODE_0, SSI_MODE_SLAVE, 10000, 16);
    SSIEnable(SSI1_BASE);
    SSIIntEnable(SSI1_BASE, SSI_RXTO);
    IntEnable(INT_SSI1);

    while(SSIDataGetNonBlocking(SSI0_BASE, &RecData[0]))
        {
          }
   //-------------------------------------------------------------------------
    //Initializing
    float DP=0;
    float DQ=0;
    float Curtailed_PV=0;
    float DVj=0;
    float DP_nominator=0;
    float DP_denominator=0;
    float Vpzc_limit=0;
    float Delta_Vpzc=0;
    float P_max=0;
    float Q_max=0;
    float DVj_found=0;
    float Vj_found=0;
    void *P3,*P1,*P2;

    ADCSequenceEnable(ADC0_BASE, 1);
    ADCIntClear(ADC0_BASE, 1);

   pindata=GPIOPinRead(GPIO_PORTB_BASE, GPIO_PIN_1);
```

```
    while (pindata != 2) {
     pindata=GPIOPinRead(GPIO_PORTB_BASE, GPIO_PIN_1);
    }

while(1)
 {
      while(flag==0){}
      flag=0;
 //-------------------------------------------------------------------------------------
        ADCProcessorTrigger(ADC0_BASE, 1);
        while (!ADCIntStatus(ADC0_BASE, 1, false))
         {
         }
         ADCIntClear(ADC0_BASE, 1);
         ADCSequenceDataGet(ADC0_BASE, 1, Value);

Vj[1]=Vj[0];
Vi[1]=Vi[0];
Vpzc1[1]=Vpzc1[0];

Vj[1]=Value[0] & 0xFFFF;

Vi[1]=Value[1] & 0xFFFF;

Vpzc1[1]=Value[2]& 0xFFFF;

P[0]=Value[3]& 0xFFFF;

Q[0]=Value[4]& 0xFFFF;

//-------------------------------------------------------------------------------
  // estimated PV,BESS and PZC voltage at next time step t+1

    Vj[2]= Vj[1]+(Vj[1]-Vj[0]);
    Vi[2]= Vi[1]+(Vi[1]-Vi[0]);
    Vpzc1[2]=Vpzc1[1]+(Vpzc1[1]-Vpzc1[0]);
//-------------------------------------------------------------------------------
//### Code: Estimating Next Step paramtes and Checking for Violation ###
  int c;
    int n;
    float Violation_upper[3]={0};

    for(n=0;n<=2;n++)
    {

      V[n][2]=V[n][1]+(V[n][1]-V[n][0]);

    }

   float minimum= Violation_upper[0];

    for(n=0; n<=2; n++)
    {
       if(V[n][2]>V_upper)
       {
           upper_flag=1;
             Violation_upper[n]=V_upper-V[n][2]-e;
       }
       }
 if(upper_flag==0)
```

```
   {
   SSIDataPut(SSI0_BASE, alpha);
   while(SSIBusy(SSI0_BASE))
     {
       }

     while(head==tail){}
     RecAB=RecData[head];
     head=(head+1)%10;

    if(RecAB==1)
      {
//  update
     SSIDataPut(SSI3_BASE,DP);
       while(SSIBusy(SSI3_BASE))
        {
       }
       SSIDataPut(SSI3_BASE,DQ);
       while(SSIBusy(SSI3_BASE))
        {
         }
      SSIDataPut(SSI3_BASE, Curtailed_PV);
      while(SSIBusy(SSI3_BASE))
        {
         }
      }
    }
      else  if(RecAB==3)
      {
//### Code : When agent recive a request from adjacent Neighbour  ###
      while(head==tail){}
      D1=RecData[head];
      head=(head+1)%10;

      while(head==tail){}
      D2=RecData[head];
      head=(head+1)%10;

      while(head==tail){}
      D3=RecData[head];
      head=(head+1)%10;

      while(head==tail){}
      D4=RecData[head];
      head=(head+1)%10;
      RD1=(D1<<16)|D2;
      RD2=(D3<<16)|D4;

     float  V_upper_pzc=0;
   Vpzc_limit_recieved=*((float*)&RD1);
     Delta_Vpzc_recieved=*((float*)&RD2);
   V_upper_pzc=Vpzc_limit_recieved;
   Vpzc1[2]=Vpzc1[1]+(Vpzc1[1]-Vpzc1[0]);
   DVj=V_upper_pzc-e-Vpzc1[2]+Delta_Vpzc_recieved;

DP_nominator=
(Vpzc1[2]*DVj)-((P[0]*tan(acos(minPF))-Q[0])*((V[1][2]*Q_si)-X_ipzc1));
 DP_denominator=
```

```
(V[1][2]*P_si)-(R_ipzc1)+(tan(acos(minPF))*((V[1][2]*Q_si)-X_ipzc1));
  DP=DP_nominator/DP_denominator;
  DQ=((P[0]+DP)*tan(acos(minPF)))-Q[0];

     S1=pow((DP+P[0]),2)+pow((DQ+Q[0]),2);
     Sm1=pow(Smax,2);
   if( S1 > Sm1)
    {
          P_max=sqrt(pow(Smax,2)/(1+pow((tan(acos(minPF))),2)));
        Q_max=P_max*tan(acos(minPF));
        DP=-P_max-P[0];
      DQ=-Q_max-Q[0];
DVj_found=
(1/Vpzc1[2])*((V[1][2]*((P_si*DP)+(Q_si*DQ)))-(DP*R_ipzc1)-(DQ*X_ipzc1));
      Vj_found=Vpzc1[2]+DVj_found+Delta_Vpzc_recieved;

       void *P4;
       P4=&Vj_found;
       SData1=*((uint32_t *)P4);
            Data1=(SData1>>16)&0x0000FFFF;
            Data2=SData1 & 0x0000FFFF;

     SSIDataPut(SSI0_BASE,nogood);
     while(SSIBusy(SSI0_BASE))
      {
       }
     SSIDataPut(SSI0_BASE,Data1);
     while(SSIBusy(SSI0_BASE))
       {
        }
     SSIDataPut(SSI0_BASE,Data2);
     while(SSIBusy(SSI0_BASE))
      {
        }
//update
       SSIDataPut(SSI3_BASE,DP);
       while(SSIBusy(SSI3_BASE))
        {
       }
       SSIDataPut(SSI3_BASE,DQ);
       while(SSIBusy(SSI3_BASE))
        {
         }
       SSIDataPut(SSI3_BASE,Curtailed_PV);
       while(SSIBusy(SSI3_BASE))
        {
         }
      }
//-------------------------------------------------------------------------
     else if( S1 < Sm1)
      {
       SSIDataPut(SSI0_BASE,good);
       while(SSIBusy(SSI0_BASE))
            {
           }
 //update
```

```
        SSIDataPut(SSI3_BASE,DP);
        while(SSIBusy(SSI3_BASE))
         {
        }
        SSIDataPut(SSI3_BASE,DQ);
        while(SSIBusy(SSI3_BASE))
         {
          }
        SSIDataPut(SSI3_BASE,Curtailed_PV);
        while(SSIBusy(SSI3_BASE))
         {
          }
       }

     }
   }
if(upper_flag==1)
//### Code: Calculating Delta P "DP" and Delta Q "DQ"  ###
 {

     minimum= Violation_upper[0];
     for(c=1; c<=2; c++)
      {
        if (Violation_upper[c] < minimum)
         {
           minimum  = Violation_upper[c];
           loc = c;
         }
       }

   DVj=V_upper-e-V[loc][2];
DP_nominator=
((V[loc][2]*DVj)-((P[0]*tan(acos(minPF))-Q[0])*((V[1][2]*Q_si)-X_ij)));
 DP_denominator=
(V[1][2]*P_si)-(R_ij)+(tan(acos(minPF))*((V[1][2]*Q_si)-X_ij));
   DP=DP_nominator/DP_denominator;
   DQ=((P[0]+DP)*tan(acos(minPF)))-Q[0];
//-------------------------------------------------------------------------------
//### Code: Checking feasability , if feasible apply to driver,
 else find possible solution and send request to neighbor agent ###

     float Sen_pzc=0;
     S=pow((DP+P[0]),2)+pow((DQ+Q[0]),2);
   Sm=pow(Smax,2);

     if (S < Sm)
      {
     SSIDataPut(SSI0_BASE,alpha);
     while(SSIBusy(SSI0_BASE))
       {
        }
     while(head==tail){}
     RecAB=RecData[head];
     head=(head+1)%10;
     if(RecAB==1)
      {
```

```
//update
        SSIDataPut(SSI3_BASE,DP);
        while(SSIBusy(SSI3_BASE))
          {
        }
        SSIDataPut(SSI3_BASE,DQ);
        while(SSIBusy(SSI3_BASE))
          {
            }
       SSIDataPut(SSI3_BASE,Curtailed_PV);
       while(SSIBusy(SSI3_BASE))
          {
            }

       }
      else  if(RecAB==3)
       {
//### Code: When agent recive a request from adjacent Neighbour  ###
      while(head==tail){}
      D1=RecData[head];
      head=(head+1)%10;

      while(head==tail){}
      D2=RecData[head];
      head=(head+1)%10;

      while(head==tail){}
      D3=RecData[head];
      head=(head+1)%10;

   while(head==tail){}
   D4=RecData[head];
   head=(head+1)%10;

   RD1=(D1<<16)|D2;
   RD2=(D3<<16)|D4;

     float  V_upper_pzc=0;
     Vpzc_limit_recieved=*((float  *)&RD1);
     Delta_Vpzc_recieved=*((float  *)&RD2);
     V_upper_pzc=Vpzc_limit_recieved;
     Vpzc1[2]=Vpzc1[1]+(Vpzc1[1]-Vpzc1[0]);
     DVj=V_upper_pzc-e-Vpzc1[2]+Delta_Vpzc_recieved;
 DP_nominator=
(Vpzc1[2]*DVj)-((P[0]*tan(acos(minPF))-Q[0])*((V[1][2]*Q_si)-X_ipzc1));
 DP_denominator=
 (V[1][2]*P_si)-(R_ipzc1)+(tan(acos(minPF))*((V[1][2]*Q_si)-X_ipzc1));
    DP=DP_nominator/DP_denominator;
    DQ=((P[0]+DP)*tan(acos(minPF)))-Q[0];

        S1=pow((DP+P[0]),2)+pow((DQ+Q[0]),2);
        Sm1=pow(Smax,2);
     if( S1 > Sm1)
        {
           P_max=sqrt(pow(Smax,2)/(1+pow((tan(acos(minPF))),2)));
           Q_max=P_max*tan(acos(minPF));
     DP=-P_max-P[0];
     DQ=-Q_max-Q[0];
```

```
DVj_found=
(1/Vpzc1[2])*((V[1][2]*((P_si*DP)+(Q_si*DQ)))-(DP*R_ipzc1)-(DQ*X_ipzc1));
     Vj_found=Vpzc1[2]+DVj_found+Delta_Vpzc_recieved;

     void *P4;
     P4=&Vj_found;
     SData1=*((uint32_t *)P4);
          Data1=(SData1>>16)&0x0000FFFF;
          Data2=SData1 & 0x0000FFFF;

       SSIDataPut(SSI0_BASE,nogood);
       while(SSIBusy(SSI0_BASE))
        {
         }
       SSIDataPut(SSI0_BASE,Data1);
       while(SSIBusy(SSI0_BASE))
        {
           }
       SSIDataPut(SSI0_BASE,Data2);
       while(SSIBusy(SSI0_BASE))
        {
           }
//update
       SSIDataPut(SSI3_BASE,DP);
       while(SSIBusy(SSI3_BASE))
        {
       }
       SSIDataPut(SSI3_BASE,DQ);
       while(SSIBusy(SSI3_BASE))
        {
         }
      SSIDataPut(SSI3_BASE,Curtailed_PV);
      while(SSIBusy(SSI3_BASE))
        {
         }
//-------------------------------------------------------------
  else if( S1 < Sm1)
      {
     SSIDataPut(SSI0_BASE,good);
     while(SSIBusy(SSI0_BASE))
      {
         }
//update
       SSIDataPut(SSI3_BASE,DP);
       while(SSIBusy(SSI3_BASE))
        {
       }
       SSIDataPut(SSI3_BASE,DQ);
       while(SSIBusy(SSI3_BASE))
          {
           }
      SSIDataPut(SSI3_BASE,Curtailed_PV);
      while(SSIBusy(SSI3_BASE))
          {
           }
      }
```

```
        }
       }
      else if (S > Sm)
       {
       //   means not feasible solution
      P_max=sqrt(pow(Smax,2)/(1+pow((tan(acos(minPF))),2)));
      Q_max=P_max*tan(acos(minPF));
   DP=-P_max-P[0];
   DQ=-Q_max-Q[0];
DVj_found=
(1/V[loc][2])*((V[1][2]*((P_si*DP)+(Q_si*DQ)))-(DP*R_ij)-(DQ*X_ij));
    Vj_found=V[loc][2]+DVj_found;
    Sen_pzc=(Vpzc1[1]-Vpzc1[0])/(Vj[1]-Vj[0]);
    Vpzc_limit=Vpzc1[1]+(V_upper-e-Vj[1])*Sen_pzc;
    Delta_Vpzc=Sen_pzc*(DVj_found);

      void *P1,*P2;
      P1=&Vpzc_limit;
      P2=&Delta_Vpzc;
    SData1=*((uint32_t*)P1);
      SData2=*((uint32_t*)P2);
      Data1=(SData1>>16) & 0x0000FFFF;
      Data2=SData1 & 0x0000FFFF;
      Data3=(SData2>>16) & 0x0000FFFF;
      Data4=SData2 & 0x0000FFFF;

    SSIDataPut(SSI0_BASE,beta);
    while(SSIBusy(SSI0_BASE))
       {
        }

    SSIDataPut(SSI0_BASE,Data1);
    while(SSIBusy(SSI0_BASE))
       {
        }

    SSIDataPut(SSI0_BASE,Data2);
    while(SSIBusy(SSI0_BASE))
       {
        }

    SSIDataPut(SSI0_BASE,Data3);
      while(SSIBusy(SSI0_BASE))
       {
        }
      SSIDataPut(SSI0_BASE,Data4);
      while(SSIBusy(SSI0_BASE))
       {
        }

      while(head==tail){}
      RecAB=RecData[head];
      head=(head+1)%10;
      if(RecAB==1)
       {
      while(head==tail){}
      RecGNG=RecData[head];
      head=(head+1)%10;
```

```
      if(RecGNG==1)
      {
//update
       SSIDataPut(SSI3_BASE,DP);
       while(SSIBusy(SSI3_BASE))
        {
       }
       SSIDataPut(SSI3_BASE,DQ);
       while(SSIBusy(SSI3_BASE))
        {
         }
      SSIDataPut(SSI3_BASE,Curtailed_PV);
      while(SSIBusy(SSI3_BASE))
        {
         }
       }
       else  if(RecGNG==2)
        {
       while(head==tail){}
       D1=RecData[head];
       head=(head+1)%10;
       while(head==tail){}
       D2=RecData[head];
       head=(head+1)%10;
       RD1=(D1<<16)|D2;
//###Code: Curtailment of PV Power when an agent recive a no-good msg.###
     float Senstivity_PV_pzc1=0;
     float PV_volt_affect_by_negihbours_action=0;
     float Vj_found_recieved;

     Vj_found_recieved=*((float*)&RD1);
     Senstivity_PV_pzc1=(Vj[1]-Vj[0])/(Vpzc1[1]-Vpzc1[0]);
PV_volt_affect_by_negihbours_action=
Senstivity_PV_pzc1*(Vj_found_recieved-Vpzc1[2]);
 Curtailed_PV=
(V_upper-(Vj[2]+PV_volt_affect_by_negihbours_action)-e)*PV_senstivity;
//-------------------------------------------------------------------------
//update
       SSIDataPut(SSI3_BASE,DP);
       while(SSIBusy(SSI3_BASE))
         {
        }
       SSIDataPut(SSI3_BASE,DQ);
       while(SSIBusy(SSI3_BASE))
         {
          }
       SSIDataPut(SSI3_BASE,Curtailed_PV);
       while(SSIBusy(SSI3_BASE))
         {
          }

       }
     }
    else  if(RecAB==3)
      {
      SSIDataPut(SSI0_BASE,beta);
```

```
        while(SSIBusy(SSI0_BASE))
         {
          }
//update
     SSIDataPut(SSI3_BASE,DP);
          while(SSIBusy(SSI3_BASE))
             {
            }
          SSIDataPut(SSI3_BASE,DQ);
          while(SSIBusy(SSI3_BASE))
            {
             }
          SSIDataPut(SSI3_BASE,Curtailed_PV);
          while(SSIBusy(SSI3_BASE))
            {
             }

          }

       }
      }
//-------------------------------------------------------------------------------
  }
}

void Timer0IntHandler(void)
  {
  // Clear the timer interrupt
  TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
  flag=1;
   }

void SSI1IntHandler(void)
  {
   // Clear interrupts.
    SSIIntClear(SSI1_BASE,SSI_RXTO);
    SSIDataGet(SSI1_BASE, &RData);
    RecData[tail]=RData;
    tail=(tail+1)%10;
    }
```