A Hardware-in-Loop Simulation of DC Microgrid using Multi-Agent Systems

Diana Rwegasira[1 2], Imed Ben Dhaou[3 6], Aron Kondoro[1 2], Amleset Kelati[1 5], Nerey Mvungi[2], Hannu Tenhunen[1 5]
[1] KTH Royal Institute of Technology in Stockholm (SWEDEN)
[2] University of Dar es Salaam (TANZANIA)
[3] College of Engineering, Qassim University (SAUDI ARABIA)
[5]University of Turku (FINLAND)
[6]University of Monastir (TUNISIA)
dianasr@kth.se, phd.imed.bendhaou@ieee.org, kondoro@kth.se, smleset@kth.se, nhmvungi@udsm.ac.tz, hannu@kth.se

***Abstract*—Smart-grid is a complex system that incorporates distributed control, communication, optimization, and management functions in addition to the legacy functions such as generation, storage, and control. The design and test of new smart-grid algorithms require an efficient simulator. Agent-based simulation platforms are the most popular tools that work well in the control and monitoring functionalities of the power electric network such as the microgrid. Most existing simulation tools necessitate either simulated or static data. In this paper, we propose a hardware-in-loop simulator for dc-microgrid. The simulator reads the power generated by the PV panels and the battery SoC using Raspberry PI. A physical agent that runs on Raspberry PI sends the real-time data to a dc-microgrid simulator that runs on a PC. As a proof of concept, we implemented a load-shedding algorithm using the proposed system.**

I. INTRODUCTION

The new revolution in the power industry systems is nowadays sustained by the integration of ICT technologies along with smart sensors and algorithms. Smart-grid integrates distributed energy resources through microgrid systems [1]. A basic solar PV-microgrid is composed of a controller, storage units, micro sources and loads as seen in Fig.1.

In the DC-microgrid systems, the control part aims at, for instance, matching supply and demand, controlling and monitoring the storage units (battery banks), and the switching between islanded and grid-connected modes. Several control techniques for the DC-microgrid have been proposed such as fuzzy control [2], and the multi-agent system, MAS.

In the literature, various control algorithm using MAS have been proposed [1]. However, the efficiency of those algorithms has been assessed using measurement. The design of contemporary dc-microgrid necessitates the integration of simulation, data-acquisition, and communication protocols.

Future electrical systems will be composed of DC voltage distribution and DC appliances. Several implementations of DC networks have been achieved and worked successfully for different purposes. The success of these applications is due to the advantages of DC microgrid networks such as efficiency, energy saving, power quality and reliability. In terms of energy saving a large percent of the AC loads can be saved by eliminating the AC-DC conversion losses in the distribution system [3]. Other advantages include easy isolation of the grid under a fault, better stability, and the needless for phase detection [4].

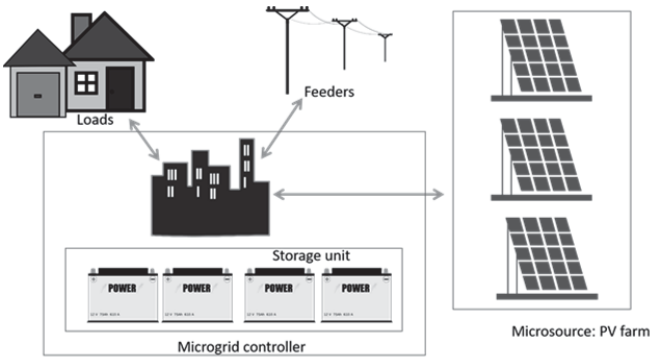

Fig. 1. Microgrid Architecture

Smart microgrid requires infrastructure in order to acquire, process, transfer and provide information extracted from the physical world. These include distributed smart nodes and smart appliances that can interact among themselves and produce an output. The work of [5] and [6] compared smart grid implementation from China, US, and Europe and came up with six technologies to adhere to. These are: (i) Real-time data management, planning and visualization technology, (ii) Integration of power grid and renewable Energies technology (iii) Power system security monitoring, fast simulation, intelligent decision-making and comprehensive defense technology, (iv) Intelligent protection and control of the power grid based on power electronics technology, (v) Energy Storage technology and application and (vi) Advanced asset management technology.

The development of the DC-microgrid needs the design and test of various techniques, algorithm, and platforms. To reduce the cost of development, simulations are the only way to go. The development of simulators for the smart grid, in general, received lots of attention both in academia and industry [7]. The authors of [8] argued that load control is the most appropriate scenario that can be used to develop a smart grid simulator.

The work described in [9] focused on agent-based simulation framework using Repast S and included load

shedding technique. While the framework is efficient, it lacks the real-time capabilities. This work focuses on describing an HIL simulation for the DC-microgrid using JADE physical agent and Repast S. The rest of the paper is organized as follows: Section II discusses related work. Section III provides motivations for load shedding framework. Section IV describes the implementation. The results are then presented in Section V. Finally, Section VI concludes the paper.

II. Related work

The efficiency and resiliency of electric power supply to serve critical and non-critical facilities are of high importance in today's life. Instead of building large electric power grids and high capacity transmission lines, an intelligent microgrid (or smart grid) can be deliberated as a promising power supply alternative and serve the purpose. A multi-agent approach to design different applications for the power system can be used to test distributed algorithms for intelligent control and monitoring of the configuration implementation into real systems[10]. In order to achieve the desired results, the realistic idea is to simulate advanced multi-agent using simulation tools with proper configuration strategies. These tools can be simulated as either a single tool, combination of two or more simulation tools or including the hardware part with simulation tools.

Previously researchers have done overwhelming work on the simulation of smart microgrid systems in the control, communications and demonstrated the advantages of simulations [7] and [8]. A comparison between four simulation tools is reported in [11] . The research by [12] proposed a two-layer framework that comprised of a control layer that with the agent-based control and a functional layer where the physical infrastructure is rivaled. In this case, the control layer can be implemented with any simulation tools such as GridLAB-D, NetLogo, and Repast while on the physical layer can be supported by other MAS tools like JADE. Also, [13] introduced the JADE and Repast framework for simulating enterprise adding value network for analyzing the management performance. The objective was to make the payment operations to be beneficial to the process with better functionalities. The results of the model explored the advantages of modeling the systems using two platforms to render the context more realistic. Likewise, [14] used the same approach and come up with the same results though for this the version used for Repast was more flexible compared to that in [13]. Also, this was specifically targeted for airport simulation for autonomous transport service. Other factors focused was scheduling and synchronization of new agents in the system, and registration of it in the environment. So far, the combination of the tools have been done with JADE and Repast and this is due to their comparison as described in Table 1 by [15] and [16].

Hardware in Loop (HiL) simulation for smart microgrid system control is another way that has recently become popular for different applications in the power systems for realization between the controller and the real system. The fact that hardware control makes IoT achievable, then the need to simulate also is more important.

TABLE I. JADE AND REPAST COMPARISONS

	JADE	**REPAST**
Distributed	Yes	No
Simulation tools	No	Yes
Scalability	Limited	High
Open source	Yes	Yes
Agent Execution	behaviors multi-thread Event-driven Asynchronous	Scheduler Single-thread, tick driven Synchronous
Interaction	Yes with FIPA ACL	Yes through Method calls and shared resources
Ontologies	Yes	No

III. The rationale for load shedding framework

Combining two MAS platforms provides flexibility and efficiency of the systems. The research done by [13] developed the framework by integrating Repast and JADE for supply chain VAN. Another work by [14] developed a framework using Repast S and JADE for airport simulations. Both works successfully but did not include a mechanism on how to integrate the framework with IoT interfaces (hardware in loop: HIL) to provide a mechanism of interfacing devices for automatic control purposes.

A solar charge controller or a regulator is an embedded system, which is responsible for regulating the DC power coming from the PV panel that goes to the battery. The block diagram of a typical solar charge controller is shown in Fig. 2.

The regulator acts as a microcontroller that implements the regulation algorithms, e.g. MPTT. The regulator engenders many functions, some of which are: metering, protection against reverse current and overcharging, load control, and temperature compensation.

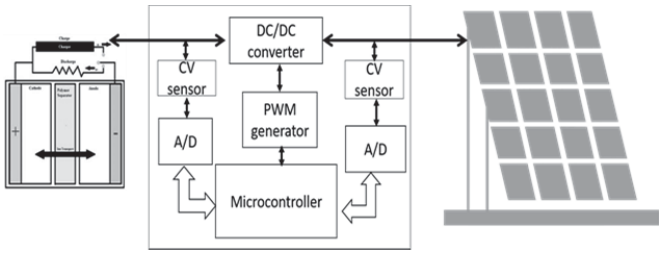

Fig. 2. Block diagram of a typical regulator

In this work, the reasons behind the use of solar charger controller are due to the following attributes: (i) to regulate the voltage from the solar panel, (ii) to monitor the voltage to the battery and (iii) to stop block reverse current at night. Furthermore, the generated power along with the battery SoC will be monitored and controlled by JADE platform while the load shedding simulation will be implemented in the Repast platform. In this manner, by using smart devices connected to the microcontrollers, data can be sent by JADE platform to the Repast S for simulation.

The integration of platforms with devices that have IoT capabilities provides a realistic communication between platforms and hence the results can be realistic, easily

visualized and interpreted. Fig. 3 describes the basic architecture of the framework with respect to the scenario. The power source (for this case is a solar panel with the charger controller) will be implemented in JADE and the Raspberry Pi will measure and send the data to the Repast engine.

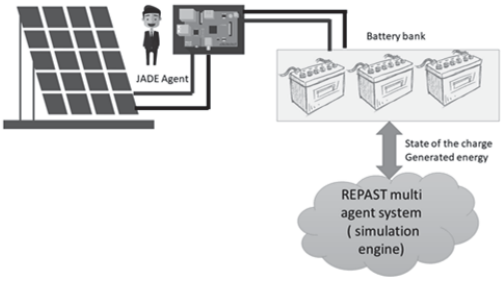

Fig. 3.Hardware-in-loop for simulating load shedding techniques

IV. HARDWARE, JADE & REPAST FRAMEWORK IMPLEMENTATION

The design and implementation of an autonomous system depend on several aspects such as control algorithm used, simulation environment, agent communication languages and security of the system. MAS has been used as a solution on accomplishing different scenario and implementations of real systems and thus, the need of the simulation framework is significant to test the visibility. To accomplish the simulation framework, the two stages where done as described in this section.

A. Platforms Interfacing (JADE and Repast)

In this paper, the JADE version used is 4.5.0 and Repast Simphony version used is 2.4 in our implementation process. Both will be operated on one PC under the same java programming language. The application of the distributed platform to run one container in the Raspberry PI and another container (the main container) on the PC have been implemented both on JADE platform. The main container is responsible for keeping the repository of all intelligent agents of the platform.

The Remote Method Invocation (RMI) has seen in Fig. 4 is the class method used for integration between JADE and Repast to allow an object running in one Java virtual machine to invoke methods on an object running in another machine. These have been done with the following steps to achieve it:

i. Create and compile an interface that specifies the methods that will have remote access

ii. Compile a class that implements remote interface in (i)

iii. Create the stub class using *rmic* tool

iv. Compile a server class that requested an object from the remote server using its hostname and the unique identifier of the object and then casts the object to the interface type from (i)

v. Then start the bootstrap *rmi* registry in the server side and execute the server class on the machine that the client named it.

vi. Execute the client class on the machine.

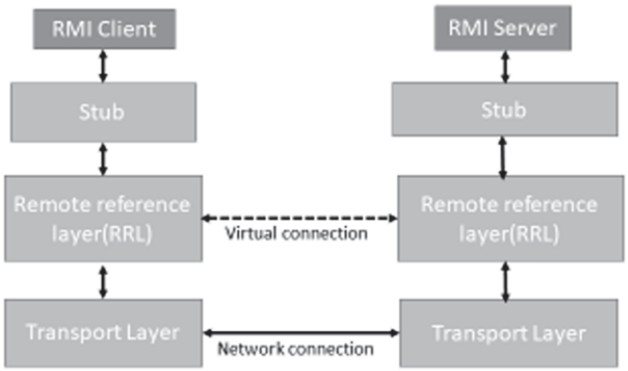

Fig. 4. Remote Method Invocation Approach

Each implemented interfaces extends the Remote class on the RMI java package. Both *PanelAgentPresentation* and *RegisterAgentToEnvironment* classes implemented class *UnicastRemoteObject* for them to be accessed remotely. Class *RegisterAgentToEnvironment* was used for registering a remote JADE agent to the Repast environment and class *PanelAgentPresentation* was used to represent JADE agent on the Repast environment.

PanelAgentPresentation contains all the methods which the Repast engine will need to inform the agent on the changing environmental parameters and obtaining values on the JADE agents. This uses a method like *schedule* for informing JADE agent on the current Repast environment.

On the JADE application, two classes and one interface were implemented. One class called *SolarAgent* extends Agent class on jade.core package (special package for JADE applications) and the other class called *AgentPresenter* which is used by the remote Repast application for remote access calls. *AgentPresenter* implements *PanelAgentInterface* the same interface implemented by the repast's PanelAgentPresentation. The function of the SolarAgent class is to read repast environment through schedule method and to act based on the information it gets from repast.

Fig. 5 describes the class diagram for the overall interactions of the hardware and the software platforms upon achieving load shedding technique whereby both will run with Java programming language.

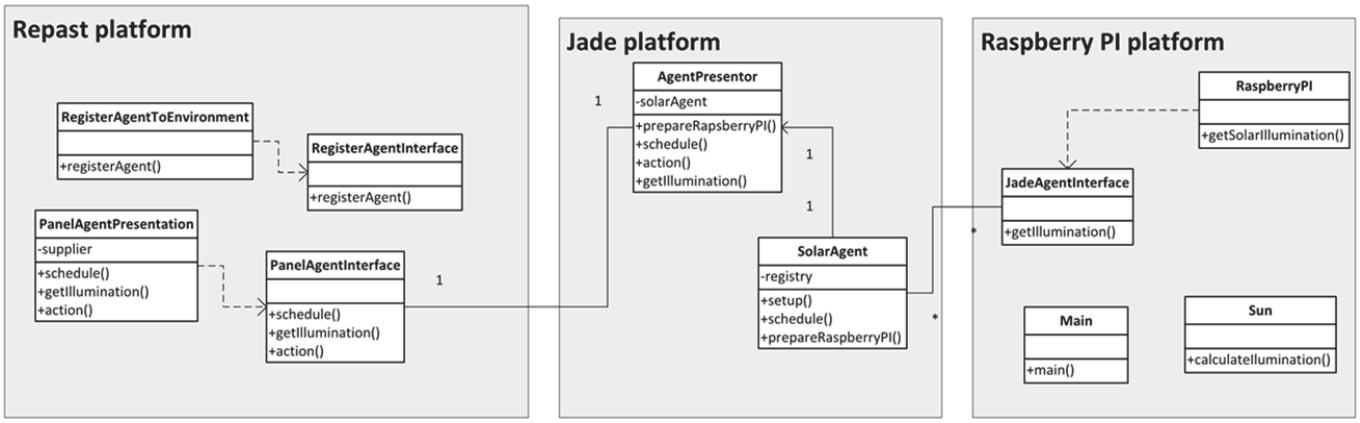

Fig. 5. Class Diagram for the framework implementation

B. *Hardware in loop interfacing*

The Raspberry PI is a small computer on a single integrated circuit that is designed specifically for embedded applications. It has evolved through several versions that feature variations in memory capacity and peripheral device support. The model has featured with a Broadcom system on a chip (SoC), an integrated ARM compatible central processing unit and on-chip graphics processing unit. Its processor speed ranges from 700 MHz to 1.2 GHz; onboard memory ranges from 256 MB to 1 GB RAM for Pi 3 version. The selected version in this work is Raspberry PI 3.

The Raspberry PI was connected by Ethernet cable with the IP address: 130.237.202.152 to get internet connection as seen in Fig. 6. It was also installed with Java Development Kit (JDK) to run Java applications as both JADE and Repast use java programming language. On this java application, agent class was implemented using JADE.jar library which acted as a remote container (responsible for receiving groups of the embedded remote agents) running on different host to support the functionality of solar panel (power source), i.e. be able to read the values from sensors. This library is found on JADE platform. The remote container has several attributes which combine both solar panel and solar charger controller. These include power control and distribution to the loads, battery charging and discharging. Three applications were installed in Raspberry PI namely *Main.java* for running the application, *Raspberry.java* extends implementation of *JadeAgentInterface* which contains the remote methods and *Sun.java* used to calculate the solar illumination intensity and report back to JADE agent.

The following is the call flow used in the process:

- When repast application starts it first creates the RMI registry of repast and then it creates an instance of *RegisterAgentToEnvironment* class and then bind this instance to the RMI registry of repast.

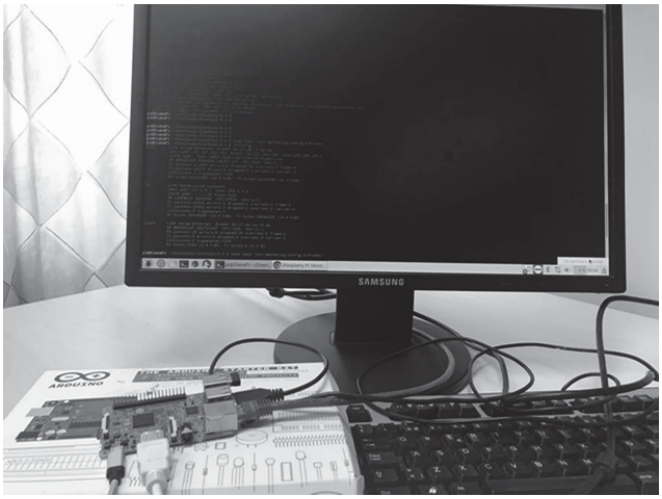

Fig. 6. Raspberry PI connectivity

- On the other hand, when JADE agent application it created its own RMI registry and then bind the *SolarAgent* instance to its registry. It then starts to look up for the *RegisterToEnvironment* instance from Repast registry and calls the method *registerAgent* to register the SolarAgent instance on Repast environment.
- The Raspberry application starts by creating it registry and then binding its Raspberry PI instance on it RMI registry.
- On each tick of the Repast simulation, it call the method schedule on JADE agent to inform on JADE agent on the time change. At the same, when the JADE agent is scheduled for the first time it looks for the Raspberry remote instance and set it up ready on the SolarAgent instance;
- When Repast simulation runs on each new tick, it schedules the *JadeAgent* and then calls the method *getSunIllumination* from Raspberry via *JadeAgent*.

V. Results and Discussion

The scenario of our problem has been demonstrated with the simulation of the loads and the power supply. Fig. 7 shows the solar agent scheduled to generate the power for the three houses. This will be changing based on the power values inserted/or generated from the PV panels.

Fig. 8 describes the communications between agents in the proposed framework. The Raspberry PI was accessed remotely by the JADE agent to collect the instantaneous value of the power (available or generated). The main container contains one agent, i.e solar for this scenario but it is possible to implement more agents based on the problem simulated.

The use of the microcontrollers with the integration and software platforms have provided the ping results on response time as seen the in Fig. 9 with maximum and minimum latency. In addition, the memory footprint in the PC that integrates the Jade and Repast occupies approximately 720KB when running with 100 ticks upon simulation.

Fig. 7. JADE Agent GUI with one instance of SolarAgent

The power consumption on the Raspberry pi was approximately 250mA when simulating the results with less than 200 ticks but with increasing tick counts the device seems to become slow on responding to the simulation.

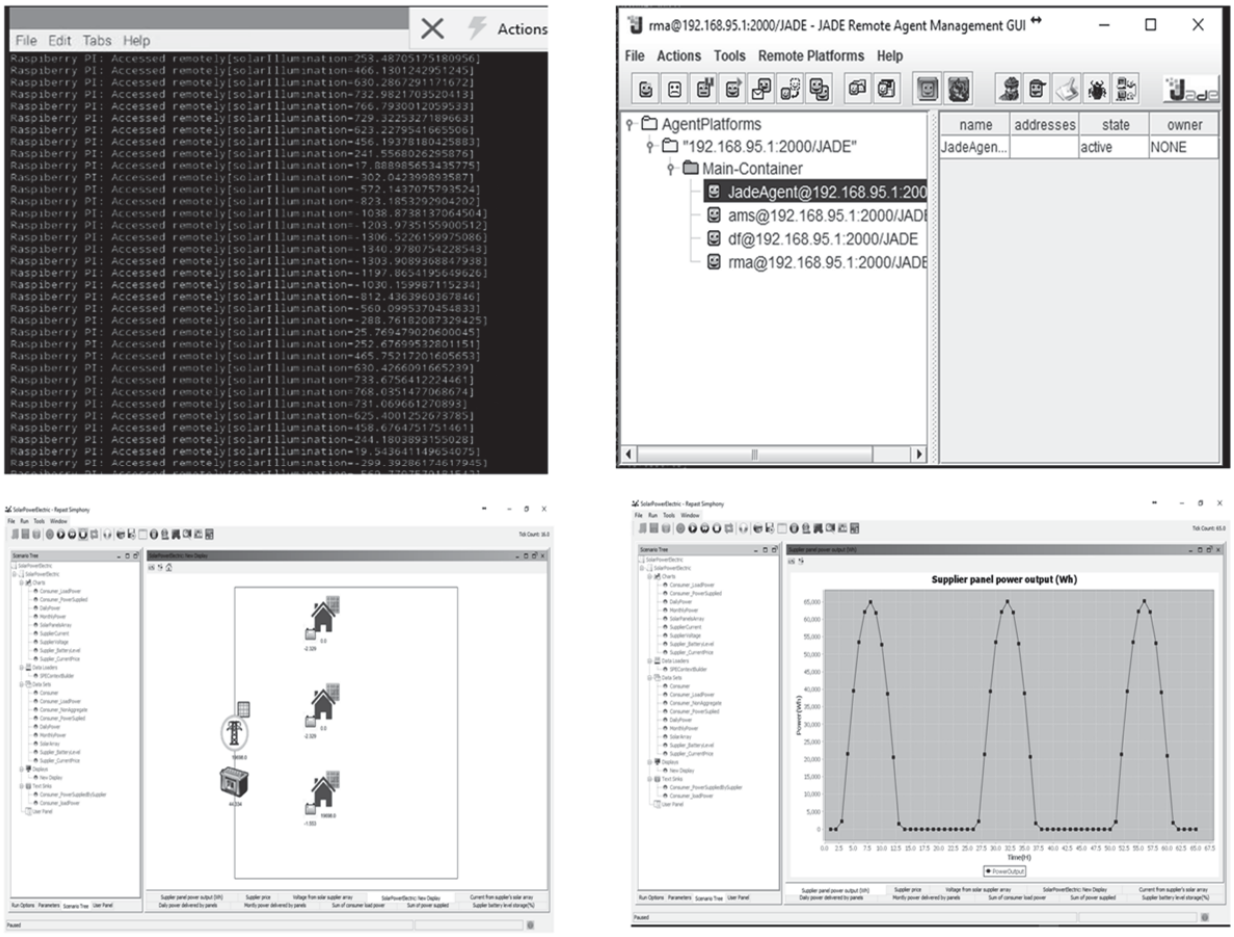

Fig. 8. (a) Raspberry PI application remotely accessed, (b) jade agent represent the solar agent and charger controller, (c) Three load agents simulations in the Repast and (d) The supplier power generated in the Repast platform

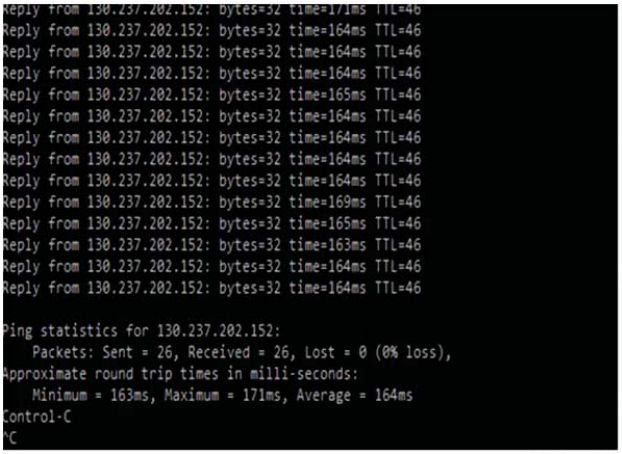

Fig. 9. Response time

The implementation of the framework can be in a distributed manner whereby the microcontrollers with Jade can be in different places with another PC running the platform and another PC running Jade and Repast for simulation. Therefore, the large-scale usage, it is easy to have different microcontrollers connected to the solar panels and charger controller and able to send data to the PC for simulation and monitoring purposes. Table II summarizes the features on the comparison of our proposed framework and the previous works.

TABLE II. COMPARISON OF PROPOSED AND EXISTING FRAMEWORK

work	Real-time data	Technologies	IoT capability	Flexibility
[12]	Yes	GridLab D	Limited	No
[13]	No	Repast, Jade	Limited	No
[14]	Yes	Repast, Jade	Limited	No
This work	Yes	Repast, Jade, Raspberry pi	Unlimited	Yes

VI. CONCLUSION

In this paper, we proposed a mixed simulation system for a DC-microgrid. The system is implemented using multi-agent systems: JADE and Repast S. The JADE agent runs on a Raspberry PI with the task of controlling and monitoring the charge of the battery. The Repast S agents are implemented on a personal computer with the task of implementing the load-shedding algorithm. The load shedding is one the control system approach used to prove the concept of the idea. The importance of this framework can be applied in any control activities such as fault detection and self-healing as long as the functional algorithm is well defined. Further research will be on the extension of the hardware device to include smart devices such as Arduino with smart sensors to control voltages, current, temperature, etc., for critical and non-critical loads.

ACKNOWLEDGMENT

This work is supported by the Swedish government through the SIDA project. Appreciation also goes to the JADE and Repast forums for the support.

REFERENCES

[1] N. Hatziargyriou, H. Asano, R. Iravani, and C. Marnay, "Microgrids," *IEEE Power and Energy Magazine*, vol. 5, no. 4, pp. 78–94, 2007.

[2] R. B. Menon, S. B. Menon, D. Srinivasan, and L. Jain, "Fuzzy logic decision-making in multi-agent systems for smart grids," in *2013 IEEE Computational Intelligence Applications in Smart Grid (CIASG)*, 2013, pp. 44–50.

[3] E. Rodriguez-Diaz, M. Savaghebi, J. C. Vasquez, and J. M. Guerrero, "An overview of low voltage DC distribution systems for residential applications," in *5th IEEE International Conference on Consumer Electronics - Berlin, ICCE-Berlin 2015*, 2016, pp. 318–322.

[4] K. Kurohane and T. Senjyu, "A Hybrid Smart AC / DC Power System," *IEEE Trans. Smart Grid*, vol. 1, no. 2, pp. 199–204, 2010.

[5] Z. R. Z. Ruihua, D. Y. Du Yumei, and Y. L. Y. Liu, "New challenges to power system planning and operation of smart grid development in China," in *2010 International Conference on Power System Technology (POWERCON)*, 2010, pp. 1–8.

[6] S. . M. Tianshu Bi, S. Liu, S. M. M. Zhenyu Huang, and N. Hadjsaid, "The implication and implementation of smart grid in China," in *IEEE 9th International Conference on power Electronics and Drive Systems*, 2011, pp. 304–309.

[7] R. Podmore and M. R. Robinson, "The role of simulators for smart grid development," *IEEE Trans. Smart Grid*, vol. 1, no. 2, pp. 205–212, 2010.

[8] J. Nutaro, "Designing power system simulators for the smart grid: Combining controls, communications, and electro-mechanical dynamics," *IEEE Power Energy Soc. Gen. Meet.*, 2011.

[9] D. Rwegasira *et al.*, "A framework for Load shedding and Demand Response in DC Microgrid using multiagent systesm," in *21st conference of FRUCT Association*, 2017, pp. 1–6.

[10] A. M. Kosek, O. Lünsdorf, S. Scherfke, O. Gehrke, and S. Rohjans, "Evaluation of smart grid control strategies in co-simulation - Integration of IPSYS and mosaik," in *Power Systems Computation Conference, PSCC 2014*, 2014, no. 4, pp. 1–7.

[11] A. Kondoro, I. Ben Dhaou, D. Rwegasira, A. Kelati, N. Shililiandumi, and N. Mvungi, "Simulation Tools for a Smart Micro-Grid : Comparison and Outlook," in *21st conference of FRUCT Association*, 2017, pp. 1–7.

[12] A. Ferreira, P. Leitão, and P. Vrba, "Simulating smart grid using a two-layer multiagent framework," in *IEEE International Conference on Industrial Technology*, 2015, pp. 2982–2987.

[13] M. J. Yoo and R. Glardon, "Combining JADE and repast for the complex simulation of enterprise value-adding networks," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5386, pp. 116–130, 2009.

[14] J. Görmer, G. Homoceanu, C. Mumme, M. Huhn, and J. P. Müller, "JREP: Extending Repast Simphony for JADE agent behavior components," *Proc. - 2011 IEEE/WIC/ACM Int. Conf. Intell. Agent Technol. IAT 2011*, vol. 2, pp. 149–154, 2011.

[15] H. L. Cardoso, "Multi-Agent Systems Software Platforms / Frameworks." FEUP-Universidade do Porto Faculdade de Engenharia, pp. 1–38, 2017.

[16] J. P. C. Lopes and H. L. Cardoso, "From Simulation to Development in MAS: A JADE-based Approach," *Int. Conf. Agents Artif. Intell. (ICAART 2015)*, pp. 1–12, 2015.