OpenPLC61850: An IEC 61850 compatible OpenPLC for Smart Grid Research

Muhammad M. Roomi^{*}, Wen Shei Ong^{*}, Daisuke Mashima^{*}, Suhail S. M. Hussain[†]
^{*}Illinois at Singapore Pte. Ltd., Singapore

roomi.s@adsc-create.edu.sg, wenshei.ong@adsc-create.edu.sg, daisuke.m@adsc-create.edu.sg

[†]School of Computing, National University of Singapore, Singapore
suhail@comp.nus.edu.sg

Abstract: OpenPLC is an open-source software that complies with IEC 61131-3 international standard and has been widely used by researchers to validate PLC logic execution. However, the software's capability is limited as it is solely restricted to Modbus and DNP3 protocols. With the integration of intelligent electronic devices (IEDs) into modernized power grid based on IEC 61850 standards, it is necessary to incorporate functionalities that enable PLCs to communicate with IEDs. Therefore, in this paper we develop OpenPLC61850, which is an extended version of OpenPLC and is compatible with the IEC 61850 protocol. The architecture and the procedure involved with PLC logic execution, IED communication, and SCADA connectivity are explained in this paper. As the proposed software is also an open source, we believe that this would be helpful for researchers for conducting smart grid research.

Keywords—IEC 61850, OpenPLC, OpenPLC61850, Communication mapping

I. INTRODUCTION

Industrial automation has increased productivity by manifold, due to the surging demand for premium quality products and services. This has rendered industrial automation a critical component of the production processes of many companies [1]. The automated process is implemented by supervisory control and data acquisition (SCADA), programmable logic controllers (PLC), and distributed control systems (DCS). To understand the operation of PLC, knowledge of the device's hardware and firmware is essential. However, this information is limited by the published disclosures of product vendors. Therefore, in [2] open source software (OpenPLC) was introduced. It is a fully functional open source logic controller that provides a low cost industrial solution for automation and research. However, the software is limited to Modbus and DNP3 protocols only. With recent advances in the industrial control systems and the evolution of digitalization in power grids, the integration of internet of things (IoT) devices necessitate advance protocols. One such popular and widely accepted standard for securing power grid is IEC 61850, which follows different protocols. Therefore, in this paper we propose a software named OpenPLC61850. The advantages of the proposed software are its open source specification and support for manufacturing message

specification (MMS) protocol of IEC 61850. Therefore, the new version of OpenPLC can be utilized for research and automation experiments that involve intelligent electronics devices (IEDs).

This paper is structured as follows: Architecture of existing OpenPLC software followed by the proposed OpenPLC61850 architecture, memory mapping between OpenPLC and IEC 61850, implementation of the proposed software, and link to the OpenPLC61850 website.

II. ARCHITECTURE: OPENPLC AND OPENPLC61850

A. OpenPLC:

OpenPLC is an open-source Programmable Logic Controller. The detailed information about the software can be found in [3], [4]. OpenPLC contains 2 main components: 1) PLC runtime which runs the PLC program and servers/clients for communication in different protocols; 2) Web server, which runs a web application for users to control the PLC runtime. The architecture of the OpenPLC is demonstrated in Fig. 1.

B. OpenPLC61850:

OpenPLC61850 is an extension of OpenPLC by Thiago Alves. OpenPLC61850 supports the IEC 61850 protocol using *libiec61850* [5]. The architecture of OpenPLC61850 is depicted in Fig. 2. IEC 61850 support in OpenPLC is implemented using 2 new subcomponents within OpenPLC61850:

- IEC 61850 Server
- IEC 61850 Client

Both the server and client run on separate threads within

Fig. 1. OpenPLC architecture

Fig. 2. OpenPLC61850 architecture

OpenPLC61850 and use `libiec61850`. Additionally, a tool was developed in order to make the IEC 61850 support possible, which is the *mapper tool*. The detailed description of the tools and the subcomponents are explained in Section III B and Section IV, respectively.

C. Terminologies Used:

- OpenPLC61850
 - The multithreaded runtime that runs the PLC program and multiple servers/clients for different protocols (e.g. Modbus, IEC 61850) on different threads
 - Written in C++
 - Files for OpenPLC61850 are contained within `webserver/core`
 - Compilation of OpenPLC61850 is done using: `webserver/scripts/compile_program.sh`
- PLC program
 - The PLC logic; contained within the ST file output from OpenPLC Editor
 - Written in Structured Text (ST)
 - The ST files are found in `webserver/st_files`. The currently active ST file is written in `webserver/active_program`
- Web Server
 - The web server which runs a web application to control OpenPLC61850
 - Written in Python
 - Files can be found in `webserver`

III. MAPPING BETWEEN OPENPLC AND IEC 61850

The main contribution of this paper is the mapping between OpenPLC and IEC 61850, in order to successfully execute the software. In this section, the description of the mapping is detailed. Fig. 3 shows the component design and interaction of OpenPLC61850 with the IEC 61850 server and client, excluding the servers for other protocols.

A. PLC Program and Memory

The PLC program reads input values and provides the logic outputs to the PLC memory. For more information on PLC memory address, please refer to [6]. This memory is implemented in the form of arrays of different data types. There are the following arrays (defined in `glueVars.cpp`), as depicted in Fig. 4.

Fig. 3. Interaction architecture between OpenPLC and IEC 61850

```
//Internal buffers for I/O and memory. These buffers are defined in the
//auto-generated glueVars.cpp file
#define BUFFER_SIZE 1024

//Booleans
IEC_BOOL *bool_input[BUFFER_SIZE][8];
IEC_BOOL *bool_output[BUFFER_SIZE][8];

//Bytes
IEC_BYTE *byte_input[BUFFER_SIZE];
IEC_BYTE *byte_output[BUFFER_SIZE];

//Analog I/O
IEC_UINT *int_input[BUFFER_SIZE];
IEC_UINT *int_output[BUFFER_SIZE];

//Memory
IEC_UINT *int_memory[BUFFER_SIZE];
IEC_DINT *dint_memory[BUFFER_SIZE];
IEC_LINT *lint_memory[BUFFER_SIZE];

//Special Functions
IEC_LINT *special_functions[BUFFER_SIZE];
```

Fig. 4. Arrays of different data types

In order for the PLC program to work with IEC 61850, we need to pass the values of the IEC 61850 data attributes to the PLC program so that it can execute its logic. Thus, to achieve IEC 61850 support, the values of the IEC 61850 data attributes is passed to the PLC program and vice versa by reading from/writing into these arrays.

The reading and writing actions are performed using functions defined in `iec61850_rw.cpp`. These functions are used by both the IEC 61850 server and the client to read from/write into the PLC memory. In order to determine the memory that has to be accessed, we need to assign an address to each data attribute. This is done by using the mapper tool which is described in the next subsection.

B. Mapper Tool

This tool is used for the IEC 61850 server and client. Both the server and client rely on the tool output to support IEC 61850. This tool is found at `utils/iec61850_mapper_src`. The mapper tool assigns PLC memory addresses to the IEC 61850 data attributes. This is achieved by providing 2 kinds of inputs:

- ST file (PLC program, exported from OpenPLC Editor)
- SCL file(s)

A sample of the structure is depicted in Fig. 9 (Section V B). One of the advantages of the tool is its automated execution while compiling the OpenPLC61850, thereby, rendering manual intervention unnecessary.

C. Input

For the ST file, the tool extracts a mapping of PLC program variables into the PLC memory address. The tool uses *regex* matching to determine the lines for parsing. For example, in the snippet of the ST file in Fig. 5, *line_cb_0* is mapped to *%QX00.0*, and *line_cb_1* is mapped to *%QX00.1*, etc.

For the SCL files, the tool parses the *Private* elements in the SCL files to obtain a mapping of the PLC program variables with the IEC 61850 data attributes. The tool uses an XML parser, *pugixml*, to traverse the SCL file. Once all the input files have been processed, the mapper tool will combine the 2 mappings to create a new mapping of IEC 61850 data attributes with the PLC memory addresses. This new mapping will be written into the output file.

D. Output

The mapper tool provides different outputs when used for the server and the client. When the OpenPLC61850 is compiled, the tool is run by the compilation script twice; once for the server and once for the client. The outputs will be contained in separate files (*iecserver.map* and *iecclient.map*).

The mapping between OpenPLC and IEC 61850 server is depicted in Fig. 6. It contains the mapping of IEC 61850 data attributes with the PLC memory addresses. CONTROL and MONITOR refers to whether the subsequent data attribute is operable or not.

Similarly, the mapping between OpenPLC and IEC 61850 client is illustrated in Fig. 7. In addition to the mapping of IEC 61850 data attributes with the PLC memory addresses, it also includes the IP address, available report control blocks (RCB), and the operable IEC 61850 data objects of each IED.

IV. OPENPLC61850 SUBCOMPONENTS

A. IEC 61850 Server

The server is the component which enables IEC 61850 clients to connect to OpenPLC61850. The server is responsible for allowing clients such as an IEC 61850 SCADA system to connect with the OpenPLC61850 and issue read/operate commands. The server is implemented in *iec61850server.cpp*. The server is created using *libiec61850*. *libiec61850* provides a model generator, *genmodel.jar* (requires JRE), that takes in a SCL file and creates a static model of the IEC 61850 server. This model generator is automatically run when

```
WaitStatus : BOOL := true;
pass_code : BOOL;
END_VAR
VAR
  line_cb_0 AT %QX00.0 : BOOL;
  line_cb_1 AT %QX00.1 : BOOL;
  line_cb_2 AT %QX00.2 : BOOL;
  line_cb_3 AT %QX00.3 : BOOL;
  line_cb_4 AT %QX00.4 : BOOL;
  line_cb_5 AT %QX00.5 : BOOL;
  line_cb_6 AT %QX00.6 : BOOL;
  line_cb_7 AT %QX00.7 : BOOL;
  line_cb_8 AT %QX00.8 : BOOL;
  line_cb_9 AT %QX00.9 : BOOL;
```

Fig. 5. Mapping to PLC memory

```
MONITOR OpenPLCplusLogicalDevice/GGIO33.AnIn1.mag.f %MD414
MONITOR OpenPLCplusLogicalDevice/GGIO34.AnIn1.mag.f %MD415
MONITOR OpenPLCplusLogicalDevice/GGIO35.AnIn1.mag.f %MD416
MONITOR OpenPLCplusLogicalDevice/GGIO36.AnIn1.mag.f %MD417
MONITOR OpenPLCplusLogicalDevice/GGIO37.SPC.stVal %QX00.0
CONTROL OpenPLCplusLogicalDevice/GGIO37.SPC.Oper.ctlVal %QX00.0
MONITOR OpenPLCplusLogicalDevice/GGIO38.SPC.stVal %QX00.1
CONTROL OpenPLCplusLogicalDevice/GGIO38.SPC.Oper.ctlVal %QX00.1
MONITOR OpenPLCplusLogicalDevice/GGIO39.SPC.stVal %QX00.2
CONTROL OpenPLCplusLogicalDevice/GGIO39.SPC.Oper.ctlVal %QX00.2
```

Fig. 6. Mapping to IEC 61850 server

```
192.168.170.159 IP Address
IEDServer09LogicalDevice/LLN0.RP.Measurements01 IEDServer09LogicalDevice/LLN0Measurements
IEDServer09LogicalDevice/LLN0.RP.Measurements02 IEDServer09LogicalDevice/LLN0Measurements
IEDServer09LogicalDevice/LLN0.RP.Measurements03 IEDServer09LogicalDevice/LLN0Measurements
IEDServer09LogicalDevice/LLN0.RP.Measurements04 IEDServer09LogicalDevice/LLN0Measurements
IEDServer09LogicalDevice/LLN0.RP.Measurements05 IEDServer09LogicalDevice/LLN0Measurements
IEDServer09LogicalDevice/LLN0.RP.Controls01 IEDServer09LogicalDevice/LLN0Controls
IEDServer09LogicalDevice/LLN0.RP.Controls02 IEDServer09LogicalDevice/LLN0Controls
IEDServer09LogicalDevice/LLN0.RP.Controls03 IEDServer09LogicalDevice/LLN0Controls
IEDServer09LogicalDevice/LLN0.RP.Controls04 IEDServer09LogicalDevice/LLN0Controls
IEDServer09LogicalDevice/LLN0.RP.Controls05 IEDServer09LogicalDevice/LLN0Controls
IEDServer09LogicalDevice/GGIO3.SPC1.Oper.ctlVal %QX00.9 Operable data objects
MONITOR IEDServer08LogicalDevice/GGIO1.AnIn1.mag.f %XD0
MONITOR IEDServer08LogicalDevice/GGIO2.AnIn1.mag.f %XD000
MONITOR IEDServer08LogicalDevice/GGIO3.SPC1.stVal %QX00.0
CONTROL IEDServer08LogicalDevice/GGIO3.SPC1.Oper.ctlVal %QX00.0
MONITOR IEDServer10LogicalDevice/GGIO1.AnIn1.mag.f %XD10
MONITOR IEDServer10LogicalDevice/GGIO2.AnIn1.mag.f %XD410
MONITOR IEDServer10LogicalDevice/GGIO3.SPC1.stVal %QX00.10
CONTROL IEDServer10LogicalDevice/GGIO3.SPC1.Oper.ctlVal %QX00.10
```

Fig. 7. Mapping to IEC 61850 client

compiling OpenPLC61850. The model generator produces 2 files: *static_model.h* and *static_model.c*. *static_model.c* contains the server model which includes the IEC 61850 objects and any reports and datasets as configured in the SCL file.

The *libiec61850* API is then used to create and run the IEC 61850 server instance with the methods *IedServer_create(&iedModel)* and *IedServer_start(iedServer,port)*. Once the server is started, the IEC 61850 server is able to handle incoming client connections and its data points and capabilities are as configured in the server's SCL file.

Updating Server Model Values

During the server runtime, the main action of the server is to update the values of its IEC 61850 data attributes in the server model. The server performs this by calling the function *update_server()* in the main loop. Before the server starts, it will process the mappings in *iecserver.map* by calling *process_server_mappings()*. The server stores the string references to the data attributes in a vector. Each time the function *update_server()* is called, it will iterate through all the references in the vector, extract the values from the corresponding PLC memory address and update the data attribute in the server model. As mentioned earlier, reading from/writing into the PLC memory is performed using the functions in *iec61850_rw.cpp*.

Control handling

The server is also capable of accepting "operate" commands on data objects that can be operated on by a SCADA system. The server calls the function *set_control_handlers()* which sets up control

handlers for each operable data object in the server. The control handler is a callback function which is called when a control command is received. It is defined as `controlHandler()`. The server determines which data objects are operable by iterating through a vector of operable data objects. These objects are obtained by processing `iecserver.map`. When the server receives a control from a SCADA system, the IEC 61850 server will call the control handler for that particular data object, which writes the value of the control command into the PLC memory.

B. IEC 61850 Client

The client is the component that enables OpenPLC61850 connection with IEDs. The client is responsible for updating the PLC program with the latest values from the IEDs. It accomplishes this task by receiving reports from the IEDs rather than utilizing the read commands. The client is implemented in `iec61850client.cpp`.

Firstly, the client component processes the mappings by calling `process_client_mappings()`. This function reads the mapper tool output `iecclient.map` and processes it. From the file, it creates a vector of `IED` structures (Fig. 8), which contains the IP address, RCBs and corresponding datasets, as well as the operable data objects for each IED.

The main function of the IEC 61850 client component is to receive and process reports from the IEDs. The client uses the `Libiec61850` API to connect to the IEDs and receive reports. To receive IEC 61850 MMS reports, the client primarily needs to establish a TCP connection to the IED. Once connection has been established, the client will install report handlers to receive reports. Report handlers are installed for every report in each IED. The report handler is a callback function that is called when the client receives a report, similar to the control handlers in the server component. This function is defined in `reportCallbackFunction()`. The report handlers are installed by calling the function `IedConnection_installReportHandler()` from the `Libiec61850` API.

When the client receives a report, it writes into the PLC memory by referring to those IEC 61850 data attributes mapped onto the PLC memory addresses obtained from `iecclient.map`. Writing to the PLC memory is done using the functions in `iec61850_rw.cpp`. Another function of the client is to send control commands to the IEDs. In the main loop of the client component, the function `checkControlChanges()` is called. This function checks the values at the PLC memory addresses for every operable data object for each IED. If the value at a PLC address is different from the data object value, it

will use the `Libiec61850` API to issue a control command to the respective IED using `sendOperateCommand()`.

Limitations

Currently, the client can only check control changes and issue operate commands for BOOLEAN data attributes in `checkControlChanges()`. Additionally, the current version of OpenPLC61850 supports only the MMS protocol in IEC 61850, and the query is done periodically using report control blocks.

V. IMPLEMENTING PROCEDURE FOR OPENPLC61850

In this section, the procedure to implement the OpenPLC61850 software is explained. The steps comprise:

1. Install OpenPLC61850
2. Add SCL files
3. Upload ST file
4. Start OpenPLC61850

A. Install OpenPLC61850

Installation process of OpenPLC61850 is similar to that of OpenPLC. Currently, only the Linux installation is supported, as the installation scripts have not been updated for other platforms. The instructions for installation in Linux are provided in [7].

B. Add SCL files

The use of IEC 61850 SCL files is an additional feature to the OpenPLC to support IEC 61850. There are 2 kinds of SCL files that will be used:

- 1 SCL file for the IEC 61850 server (stored in `webserver/scl_server_files`)
- 1 or more SCL files from the IEDs that the IEC 61850 client will connect to (stored in `webserver/scl_client_files`)

Some prior configuration needs to be performed. In the SCL XML schema, it allows for a `Private` element that can be used by vendors for proprietary uses. We use this `Private` element to assign PLC program variables to the IEC 61850 data attributes. This `Private` element in the SCL files is the vital component for OpenPLC61850 to support IEC 61850.

For example, in the SCL file snippet (Fig. 9), we assign the data attribute `stVal` of a particular `SPC` data object. The `Private` element should have a `Property` child element as shown. This `Property` element should have the following attributes:

- Name
 - `sMonitoringVar`, for data attributes whose values will be updated to and from PLC program (e.g line loading)
 - `sControlVar`, for data attributes that can be operated (typically circuit breakers)
- Value
 - PLC program variable to be assigned (e.g. `line_cb_15`)

```
struct ied_t {
    std::string ipaddr;
    std::vector<std::vector<pair<std::string, std::string>>> rcb_dataset_list; //report -> dataset
    std::unordered_map<std::string, IEC_BOOL> controlWatch; //iec61850 da -> ctl value
};
typedef struct ied_t IED;
```

Fig. 8. IED structures


```

<DOType id="SPC_16" cdc="SPC">
  <DA name="stVal" bType="BOOLEAN" fc="ST" dchg="true">
    <Private name="stVal">
      <Property Name="sMonitoringVar" Value="line_cb_15"/>
    </Private>
  </DA>
  <DA name="q" bType="Quality" fc="ST" qchg="true" />
  <DA name="Oper" type="SPCOperate_1" bType="Struct" fc="CO">
    <Private name="Oper">
      <Private name="ctlVal">
        <Property Name="sControlVar" Value="line_cb_15"/>
      </Private>
    </Private>
  </DA>
  <DA name="ctlModel" type="CtlModelKind" bType="Enum" fc="CF" />
  <DA name="t" bType="Timestamp" fc="ST" />
</DOType>

```

Fig. 13. Using nested Private Elements

Note that for data attributes that are structures, such as *Oper*, the nested Private elements are required to refer to the actual data attribute within the structure, as shown in Fig. 9. Also, multiple data attributes with the same PLC program variable may be assigned (*line_cb_15* is assigned to *stVal* and *Oper.ctlVal*).

This *Private* element will be parsed by the mapper tool that was explained in Section III B. Notably this tool does not check data types, so it is crucial to ensure that the data type for each IEC 61850 data attribute can be used for the PLC program variable that has been assigned.

After the SCL files are prepared, they have to be copied into their respective folders. Currently, the web application lacks a user interface to add the SCL files, hence requiring manual copying. Also, the updating of filename of the *webserver/active_scl* with the server SCL file is necessary.

C. Upload ST file

Before uploading the ST file, ensure that SCL files are up to date as OpenPLC61850 is recompiled when changing PLC programs. The ST file contains the PLC program logic. We use OpenPLC Editor to generate the ST file. OpenPLC Editor enables the programming of the PLC logic in IEC 61131 languages, such as structured text (ST) and function block diagram (FBD). A guide for using the OpenPLC Editor can be found in [8]. Once the programming for PLC logic is completed, a ST file can be exported. Once exported, it can be uploaded to the OpenPLC61850 using the web application (localhost:8080). The step-by-step procedure is demonstrated by Fig. 10 - Fig. 14.

D. Start OpenPLC61850

Once the program is compiled, navigating to the dashboard and pressing 'Start PLC' will execute the OpenPLC61850 run, as shown in Fig. 15.

Once the PLC is started, the PLC initiates a connection with each IED. The successful connection between OpenPLC and IEDs is depicted in Fig. 16 under the 'Runtime Logs'. Subsequently, the MMS communication between the OpenPLC and the IEDs is illustrated in Fig. 17.

Fig. 9. Step 1: OpenPLC login (User: openplc, Password: openplc)

Fig. 10. Step 2: Chose 'Programs'

Fig. 11. Step 3: 'Choose' and 'Upload' program file

Fig. 12. Step 4: Enter program details and upload

Fig. 14. Compilation of program

Fig. 15. Starting OpenPLC61850

Fig. 16. Initiation of IEDs through OpenPLC

Fig. 17. MMS communication between OpenPLC and IED

VI. URL

Please refer to the following website for the demo and the GitHub repository for detailed information of OpenPLC61850:

<http://www.illinois.adsc.com.sg/OpenPLC61850/>
<https://github.com/smartgridadsc/OpenPLC61850>

VII. CONCLUSION

In this paper, we have developed a new open source software named OpenPLC61850, which is an extension of the existing OpenPLC software. In addition to the functionalities provided by OpenPLC, we have extended the features to support IEC 61850 MMS protocol. The mapping between the IED attributes and PLC memory and implementation details were explained in this paper. The developed OpenPLC61850 software can be validated by conducting simulation studies with any grid model. As a part of future work, we are working on including additional protocols of IEC 61850 such as Generic object oriented substation events (GOOSE) and Sampled value (SV) protocols.

ACKNOWLEDGEMENT

This research is supported in part by the National Research Foundation, Singapore, Singapore University of Technology and Design under its National Satellite of Excellence in Design Science and Technology for Secure Critical Infrastructure Grant (NSoE_DeST-SCI2019-0005), and in part by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

REFERENCES

- [1] "Plant Automation Technology," [Online].
- [2] T. Alves and T. Morris, "OpenPLC: An IEC 61,131–3 compliant open source industrial controller for cyber security research," *Computers & Security*, vol. 78, pp. 364-379, 2018.
- [3] T. Alves, "OpenPLC," [Online]. Available: <https://www.openplcproject.com/>.
- [4] T. Alves, "GitHub," [Online]. Available: https://github.com/thiagoralves/OpenPLC_v3.
- [5] "libIEC61850," [Online]. Available: <https://libiec61850.com/libiec61850/>.
- [6] T. Alves, "OpenPLC," [Online]. Available: <https://www.openplcproject.com/reference/Modbus/>.
- [7] T. Alves, "OpenPLC," [Online]. Available: <https://www.openplcproject.com/runtime/linux/>.
- [8] T. Alves, "OpenPLC," [Online]. Available: <https://www.openplcproject.com/reference/basics/first-project>.
- [9] [Online]. Available: <https://www.7-zip.org/download.html>.
- [10] M. M. Roomi, P. B. Biswas, D. Mashima, Y. Fan and E.-C. Chang, "False Data Injection Cyber Range of Modernized Substation System," in *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, 2020.
- [11] T. M. Thiago Alves, "OpenPLC: An IEC 61,131–3 compliant open source industrial controller for cyber security research," *Computers & Security*, vol. 78, pp. 364-379, 2018.