

Implementation of greenhouse service control protocol using Python on Raspberry PI

Wook Hyun*, Mi Young Huh**, Ju Young Park*

* ETRI (Electronics and Telecommunications Research Institute), Korea

whyun@etri.re.kr, myhuh@etri.re.kr, jypark@etri.re.kr

Abstract— ICT technology is widely applied to various industries including agriculture. In the agricultural sector, most of the devices and greenhouses still use very old serial communication technologies. Nowadays, many ICT technologies such as ZigBee, Bluetooth and the Internet are spreading but still not interoperable. A set of standards is under development in Korea to ensure interoperability between different vendors. In particular, core interoperability can be provided by standardizing the LCP protocol independent of the underlying network infrastructure. In this paper, we designed service control protocol based on LCP protocol and implemented it using Python. We have deployed the implementation to Raspberry PI with actual physical devices.

Keywords—smart greenhouse, protocol, implementation, python

I. INTRODUCTION

ICT technology is applied to various industries beyond information and communication services, and agriculture is no exception. In the field of agriculture, communication technology is applied to facility automation and optimal growth environment control to acquire environmental information from sensors and control various actuators in the facility. Most of the devices use the Modbus protocol for the RS485 environment, which was created 30 years ago. Recently, ZigBee, Bluetooth, and IP based communication methods are gradually being applied. However, most vendors use their own protocols or arbitrary extensions to Modbus, it is difficult to seamlessly exchange or interwork with heterogeneous protocols that are tailored to each communication environment. To solve this problem, TTA, a Korean domestic SDO, has developed more than 20 related standards, and in 2018, TTA is developing a lightweight control protocol (LCP) that operates independently from the lower physical layer. We implemented a service control function (SCP) to control the smart greenhouse using the LCP protocol [1], and deployed it on Raspberry Pi. In addition, we developed a server to control sensor nodes through the web at remote sites. To increase the portability, we make the server as a docker container. This paper describes the experience of developing operation procedures of LCP-based service control protocol on Raspberry PI.

II. SMART GREENHOUSE CONTROL PROTOCOL

The following Figure 1 shows the interworking relationship between SCP and LCP. Since the LCP operates independently of the lower physical communication interface, an adaptation layer is required to put it on the IP layer.

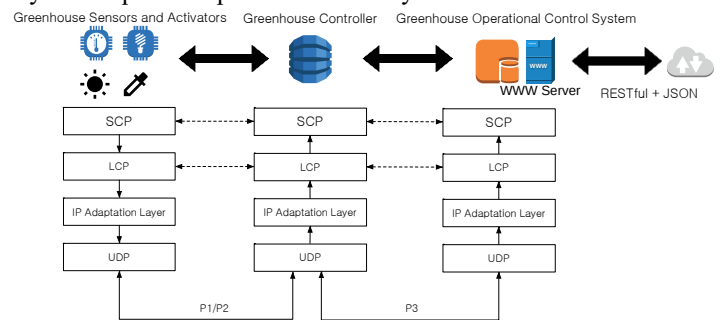


Figure 1. Architectural model of greenhouse control

A node can attach various kinds of devices at the same time. It mainly provides communication function for control and has an adaptation layer suitable for physical communication environment as shown in the Figure 1.

A. LCP (Light-weight Control Protocol)

Using the concept of CoAP [2] protocol, LCP, which is a protocol with a common concept that is not limited to the Internet environment, routes messages using the identifier of the node so as to be applicable to various lower physical communication environments. The specification of the LCP message is as follows.

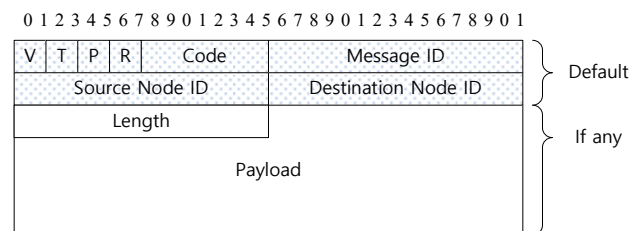


Figure 2. LCP Protocol Format [x]

In the payload of this message, SCP control message will be embedded as a text.

B. SCP (Service Control Protocol)

The SCP protocol is used to control sensor/actuator devices of smart greenhouse between actuator/sensor node and

controller node. The sensor node maintains several information such as device identifier, sensor value, type of sensor values, and the actuator node maintains device status and receive control command from controller node. The controller needs to access these information, and it is necessary to be able to control and retrieve information from multiple target devices in one message, since there can be several devices (sensors, actuators) in a single node. In addition, even if several sensors are attached to one sensor node, it is necessary to selectively read the desired value.

The main functionalities of SCP protocol in this interface are as follows;

- Acquisition of sensor value
- Acquisition of actuator status
- Activation/Deactivation of actuator

In this section, we describes those procedures in detail.

1) Acquisition of sensor value

This primitive procedure is for fetching the status of sensor devices. Firstly, the controller sends the LCP GET message to the sensor node, and put a target ID into the payload. The target ID specifies the object to be manipulated or queried. If it is needed to obtain information of all the sensors attached to the node, it uses "SENSDEV/STATUS" as the Target ID. In the case of checking only the measurement value of the specific sensor, the target ID will be like "SENSDEV/STATUS/DeviceID=<device_id>". On receiving this request message, the sensor node transmits LCP 2.00 OK message in response to the request, and its payload contains <SensingStatus> information as specified format in table 1.

TABLE 1. FORMAT OF <SENSINGSTATUS> ELEMENT

Name	Repeat	M/O	Type	
NodeID	1	M	number	
SensDevStatus	1	M	-	
DeviceID	N	1	M	number
SensingValue		1	M	number
ValueType		1	O	0: Integer, 1: float

2) Acquisition of actuator status

Unlike sensor devices, the controller should know current status of actuator devices, such as openness of windows, speed of fan, etc. This procedure provides a way to get the status of the actuator by using LCP GET message. When a controller node needs to get the status of actuator, it sends LCP GET message with a payload containing the target ID. If the controller wants to obtain status information for all actuator devices attached to the node, the Target ID should be like "ACTDEV/STATUS". When the controller wants to check only the status of a specific device, it uses "ACTDEV/STATUS/DeviceID=<device_id>" as a target ID. On receiving this request message, the actuator node respond with LCP 2.00 OK message, and its payload contains the status of actuator devices. However, there are two types of actuator in greenhouse; switching type and retractable type. Table 2 and 3 shows the format of the status of the actuator devices.

TABLE 2. FORMAT OF <RETRACTACTUATORSTATUS> ELEMENT

Name	Repeat	M/O	Type	
NodeID	1	M	number	
ActDevStatus	1	M	-	
DeviceID	N	1	M	number
OperationStatus		1	M	OPEN/OPENING/ CLOSED/CLOSING
OperationPosition		1	O	Openness in percent
OperationSpeed		1	O	Operating speed in %/sec

TABLE 3. FORMAT OF <RETRACTACTUATORSTATUS> ELEMENT

Name	Repeat	M/O	Type	
NodeID	1	M	number	
ActDevStatus	1	M	-	
DeviceID	N	1	M	number
OperationStatus		1	M	ON/OFF
OperationDuration		1	O	Duration of operation express by “PnYnMn DTnHnMnS” format
OperationTime		1	O	Time of operation as e xpress by ”hh:mm:ss”

3) Activation/Deactivation of actuator

The controller sends an LCP PUT message to the actuator node to the actuator node with a payload specified in the format of table 2 and 3, and the node responds with 2.00 OK response to the controller node.

III. IMPLEMENTATION OF PROTOCOL STACK WITH PYTHON

We have designed the inheritance model as following Figure 3. Since we use IP networking for transport the signalling messages to each node, each node should have IPAL (IP Adaptation Layer).

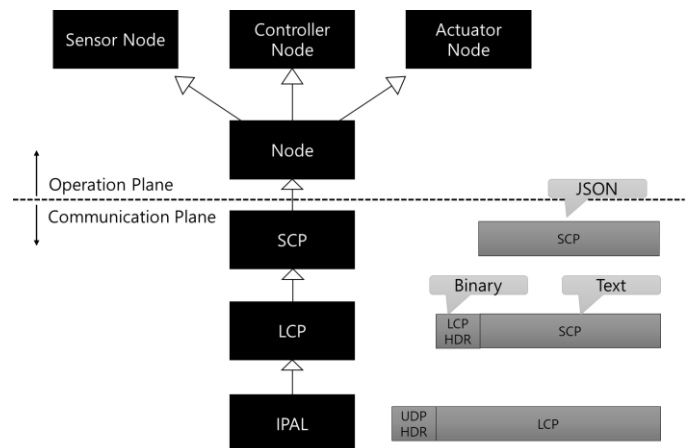


Figure 3. Inheritance design model of protocol stacks

Since LCP needs to receive message from IPAL, LCP class inherits from IPAL class, and SCP class also inherits from LCP class. The protocol stack is mainly composed of operation plane and communication plane. The operation plane is for component that running as a separate process and

the communication plane is a combination of submodules for sending and receiving message as well as parsing/generation. As shown in Figure 3, each message is embedded into the payload of each layer by appending message headers.

A. IPAL

The LCP protocol is independent of underlying network modules, and it uses just node identifier for routing the control messages, and it also assumes that the actual transmission will be done underlying adaptation layer. In our implementations, we implemented IPAL (IP Adaptation Layer) to deliver the messages through Internet. The main role of this layer is to maintain the mapping between IP address/port and LCP identifier. On receiving message from unknown node, it create a local record for later usage.

B. LCP and SCP

LCP message can have both binary and text within its payload, and the SCP message is encoded as a JSON message for extensibility. However, the size of text is rather bigger than the binary message. When the standardization on the parameter is finalized, these parameters can be encoded as a binary format. For the time being, we will use JSON encoding to convey the information.

IV. DEPLOYMENT ON RASPBERRY PI

Since Raspberry PI is a tiny Linux machine, it is possible to run python script without any modification. The following image shows an actual operation of sensor node.

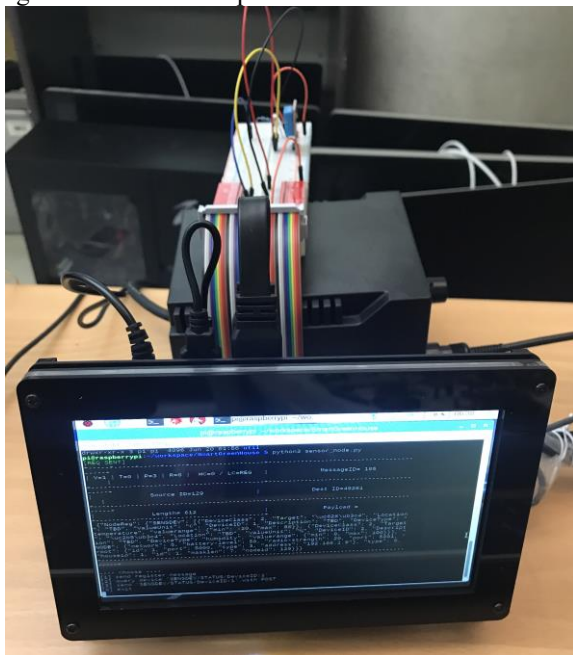


Figure 4. Deployment of sensor node on Raspberry PI

There are two sensors (temperature and humidity) are attached into this node. On bootstrapping, the node register itself to the controller node by sending LCP REG message.

After successful registration, the sensor node sends measured values within the LCP PUT message periodically. It is also possible to manual probing from the controller node by sending LCP GET message to the sensor node. For convenient debugging, the sensor node has operation menu to initiate a specific behaviour in the CLI terminal. The controller gathers the information into a docker-based smart greenhouse data analysis platform [3] for further analysis.

V. CONCLUSIONS

In this paper, we designed SCP, which is a protocol for controlling real service, using LCP protocol which can operate independently in lower physical layer among various methods for controlling Smart Greenhouse. In this process, the metadata related to the status information and control information necessary for controlling each service are defined. In addition, the designed protocol was implemented in python and installed in Raspberry PI to confirm the actual operation.

ACKNOWLEDGMENT

This work was supported by Korea Institute of Planning and Evaluation for Technology in Food, Agriculture, Forestry and Fisheries(IPET) through Advanced Production Technology Development Program Program(or Project), funded by Ministry of Agriculture, Food and Rural Affairs(MAFRA)(317015-6)..

REFERENCES

- [1] J.Y.Park et al, *LCP for Managing Smart Greenhouse*, ICTC2018
- [2] CoAP, The Constrained Application Protocol(CoAP), IETF RFC7959
- [3] W. Hyun et.al, *Implementation of docker-based smart greenhouse data analysis platform*, ICTC2018



Wook Hyun is a research staff member with ETRI (Electronics and Telecommunications Research Institutes) since 2000. He has received M.S. degree in Information Communication Engineering from Chungnam National University, Korea in 2000. His research interests include VoIP, SIP, NGN, P2P, overlay networking, digital signage, AI, Blockchain and smartfarm.



Mi Young Huh is a research staff member with ETRI (Electronics and Telecommunications Research Institutes) since 1990. She has received M.S. degree in Information Communication Engineering from Chung Nam National University, Korea in 2004. Her research interests include VoIP, SIP, IPTV, and Digital Signage, Smartfarm.



Ju Young Park is working for ETRI from when he has received his PH.D degree in 2001 from Chungnam National University. Thereafter, he took project editorships both in ITU-T and ISO/IEC/JTC1, and he also has developed three International Standards (IS). His major research areas are smart work, Multicast, QoS protocol and architecture. He also has great concerns on mobile communication and IOT.