Delivery 3 of AADL Model of The Smart Home System

Milestone of our project

Delivery Date	Milestones
Mar 13, 2017	Project proposal
Mar 20, 2017	Accomplishing the subcomponents and connections of implementation of the smart home system
Apr 03, 2017	Accomplishing the definition and implementation of the main control router model
Apr 10, 2017	Accomplishing the definition and implementation of the remote server and clients model
Apr 17, 2017	Accomplishing the definition and implementation of the zigbee controller and devices module
Apr 24, 2017	Identifying and Adding related modes and flows, which include nominal and error flows
May 01, 2017	Adding error and nominal behavior for error model

The overview of our project

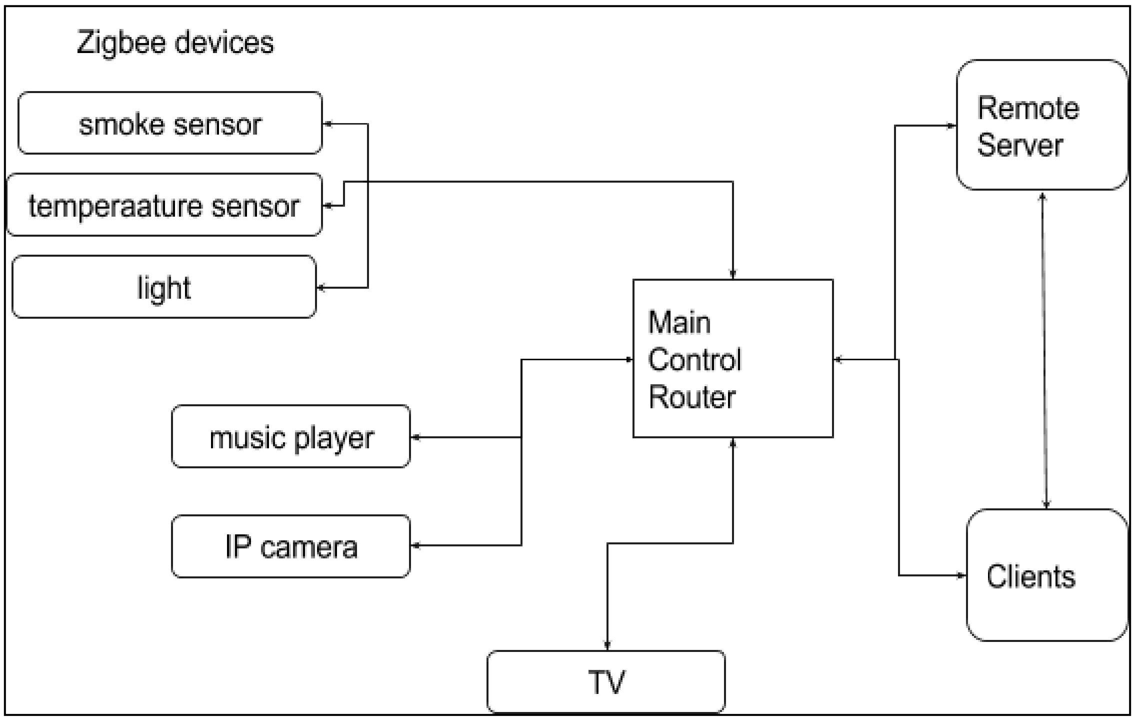

Figure 1. The Overview of Smart Home System

Overview

In this delivery, according to our milestone, we accomplished the definition and implementation of the remote server and the clients module.

For remote server, it consists two processes: client controller and router controller. The client controller receiver control messges from client and also send feadback information to the client. The router controller send the user's commands to the main control router and also receive messages from the main control router.

For clients module, it consists a process called client control unit. The function of this component is send user's commands to the remote server or main control router. Meanwhile, it also receive message from the server or main control router and show the information to user.

The Remote Server Mode

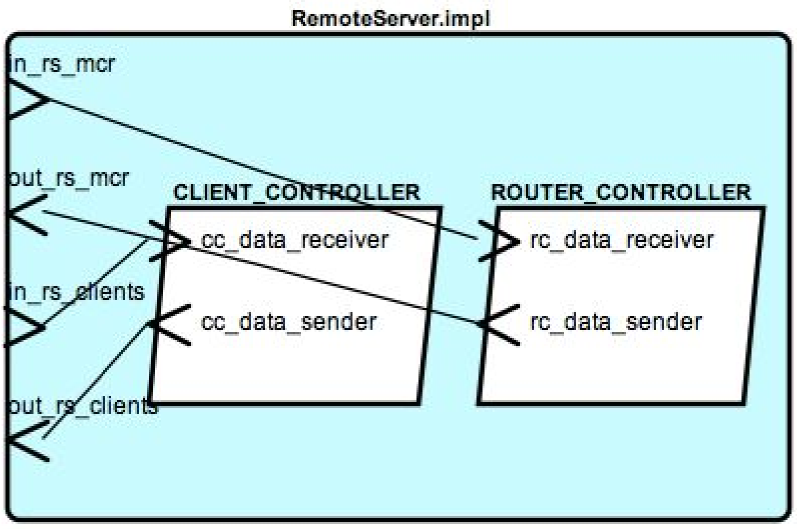

Figure 2. Remote server implementation

```
system RemoteServer
       features
              in_rs_mcr:      in event port;
              out_rs_mcr:     out event port;
              in_rs_clients:  in event port;
              out_rs_clients: out event port;
end RemoteServer;

system implementation RemoteServer.impl
       subcomponents
              CLIENT_CONTROLLER: process Client_Controller;
              ROUTER_CONTROLLER: process Router_Controller;
       connections
              c1: port in_rs_mcr -> ROUTER_CONTROLLER.rc_data_receiver;
              c2: port ROUTER_CONTROLLER.rc_data_sender -> out_rs_mcr;
              c3: port in_rs_clients -> CLIENT_CONTROLLER.cc_data_receiver;
              c4: port CLIENT_CONTROLLER.cc_data_sender -> out_rs_clients;
end RemoteServer.impl;
```

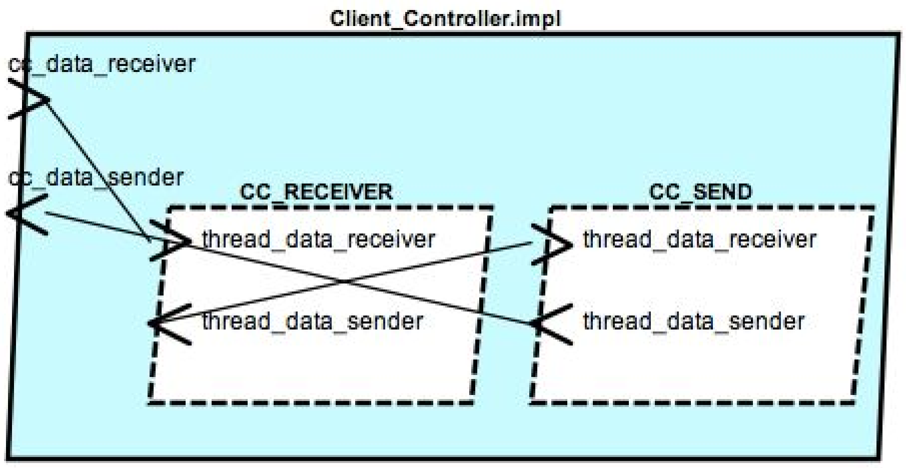

Figure 3. Client controller implementation

```
process Client_Controller
       features
              cc_data_receiver: in event port;
              cc_data_sender :  out event port;
end Client_Controller;


process implementation Client_Controller.impl
       subcomponents
              CC_RECEIVER : thread Cc_Receiver;
              CC_SEND     : thread Cc_Sender;
       connections
              c1: port cc_data_receiver -> CC_RECEIVER.thread_data_receiver;
              c2: port CC_SEND.thread_data_sender -> cc_data_sender;
              c3: port CC_RECEIVER.thread_data_sender -> CC_SEND.thread_data_receiver;
end Client_Controller.impl;

thread Cc_Receiver
       features
              thread_data_receiver:  in event port;
              thread_data_sender:    out event port;
end Cc_Receiver;

thread Cc_Sender
       features
              thread_data_receiver:  in event port;
              thread_data_sender:    out event port;
end Cc_Sender;
```

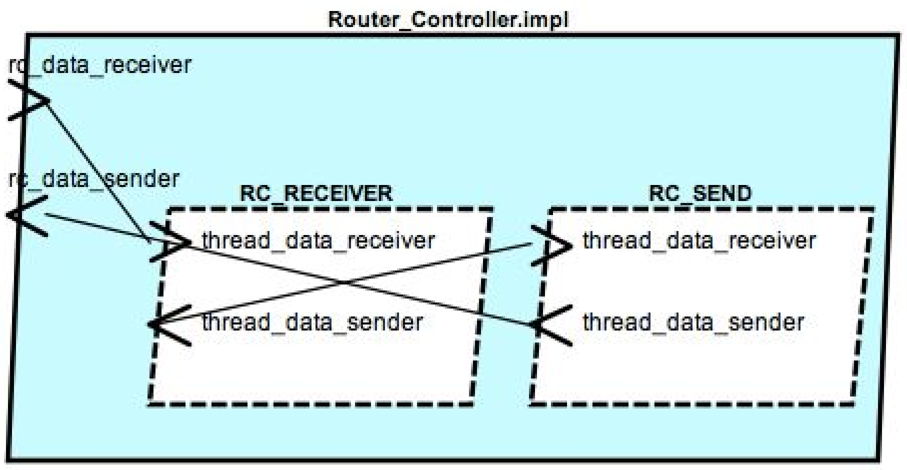

Figure 4. Router controller implementation

```
process Router_Controller
       features
              rc_data_receiver: in event port;
              rc_data_sender :  out event port;
end Router_Controller;

process implementation Router_Controller.impl
       subcomponents
              RC_RECEIVER : thread Rc_Receiver;
              RC_SEND     : thread Rc_Sender;
       connections
              c1: port rc_data_receiver -> RC_RECEIVER.thread_data_receiver;
              c2: port RC_SEND.thread_data_sender -> rc_data_sender;
              c3: port RC_RECEIVER.thread_data_sender -> RC_SEND.thread_data_receiver;
end Router_Controller.impl;

thread Rc_Receiver
       features
              thread_data_receiver:  in event port;
              thread_data_sender:    out event port;
end Rc_Receiver;

thread Rc_Sender
       features
              thread_data_receiver:  in event port;
              thread_data_sender:    out event port;
end Rc_Sender;


end RemoteServer;
```

The Clients Mode

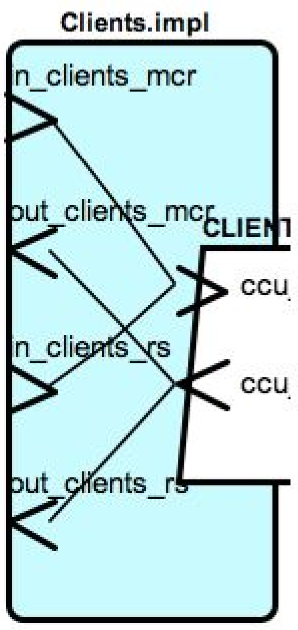

Figure 5. The clients implementation

```
system Clients
       features
              in_clients_mcr:  in event port;
              out_clients_mcr: out event port;
              in_clients_rs:   in event port;
              out_clients_rs:  out event port;
end Clients;

system implementation Clients.impl
       subcomponents
              CLIENT_CONTROL_UNIT : process Clients_Control_Unit;
       connections
              c1: port CLIENT_CONTROL_UNIT.ccu_data_sender -> out_clients_mcr;
              c2: port CLIENT_CONTROL_UNIT.ccu_data_sender -> out_clients_rs;
              c3: port in_clients_mcr -> CLIENT_CONTROL_UNIT.ccu_data_receiver;
              c4: port in_clients_rs -> CLIENT_CONTROL_UNIT.ccu_data_receiver;

end Clients.impl;
```

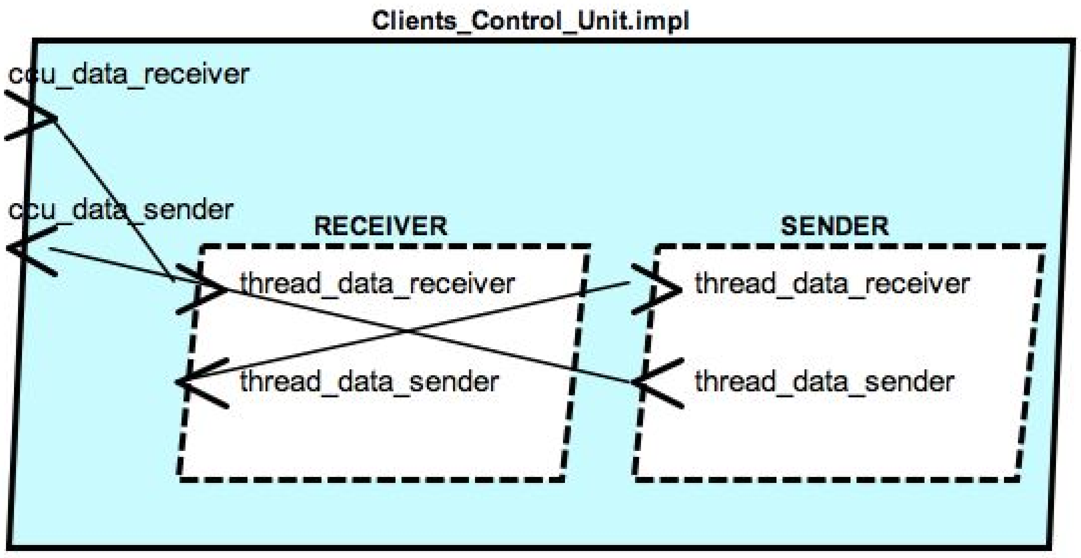

Figure 6. Client control unit implementation

```
process Clients_Control_Unit
	features
		ccu_data_receiver:  in event port;
		ccu_data_sender:    out event port;
end Clients_Control_Unit;

process implementation Clients_Control_Unit.impl
	subcomponents
		RECEIVER : thread Receiver;
		SENDER : thread Sender;
connections
		c1: port ccu_data_receiver -> RECEIVER.thread_data_receiver;
		c2: port SENDER.thread_data_sender -> ccu_data_sender;
		c3: port RECEIVER.thread_data_sender -> SENDER.thread_data_receiver;
end Clients_Control_Unit.impl;

thread Receiver
	features
		thread_data_receiver:  in event port;
		thread_data_sender:    out event port;
end Receiver;

thread Sender
	features
		thread_data_receiver:  in event port;
		thread_data_sender:    out event port;
end Sender;

end Clients;
```