**International Journal of Electrical and Computer Engineering (IJECE)**
Vol. 7, No. 2, April 2017, pp. 1012~1022
ISSN: 2088-8708, DOI: 10.11591/ijece.v7i2.pp1012-1022 

# Design of Pervasive Discovery, Service and Control for Smart Home Appliances: An Integration of Raspberry Pi, UPnP Protocols and Xbee

**Sabriansyah Rizqika Akbar, Maystia Tri Handono, Achmad Basuki**
Faculty of Computer Science, University Of Brawijaya, Indonesia

**Article Info**

***Article history:***

Received Sep 16, 2016
Revised Jan 6, 2017
Accepted Jan 20, 2017

***Keyword:***

Pervasive discovery
Service discovery
Smart home
UPnP
Xbee

**ABSTRACT**

Pervasive technology is an important feature in smart home appliances control. With pervasive technology, the user is able to discover and control every device and each service without initialization configuration and setup. Since single-board computer often used in smart home appliances, combining pervasive technology and microcomputer/single-board computer will be important to be applied and make a possibility to create a smart home system based on the requirement of it users that will be beneficial for the smart home users and the developers. This paper proposed a design of pervasive discovery, service, and control system for smart home appliances by integrating Raspberry Pi, UPnP protocols, and Xbee that able to control an RGB LED services such as switching, dimming, change color and read a temperature sensor as an example in smart home appliances. This paper enriched the raspberry Pi GPIO function to be able to control via TCP/IP network with UPnP protocol and receive information from a temperature sensor node via Xbee communication. Service control time is measured with UPnP round trip time by subtracting HTTP response arrival with HTTP request time. GPIO processing time measured at the application level by counting a timer that starts before GPIO process and ended after GPIO successfully executed.



***Corresponding Author:***

Sabriansyah Rizqika Akbar
Faculty of Computer Science,
University Of Brawijaya,
Indonesia.
Email: sabrian@ub.ac.id

## 1. INTRODUCTION

The Smart home system is one of the various applications in the Internet of Things (IOT) field. IOT contribute as a supporting technology in the everyday life of its users and has covered a variety of areas such as transport and logistics, health, smart environment, as well as the domain of personal and social [1], [2]. Internet Society defines IOT as a scenario where network and computing capabilities were implemented in a wide range of objects such as sensors and everyday equipment that is able to produce, exchange and process data with the minimum human intervention [3]. IOT has also been applied to Smart Home technologies to improve the quality of human life by implanting objects (things) that are able to communicate with each other and elaborating the information from the environment. One of the main issues in a smart home system is people have different needs. With the complex and unclear requirement, the smart home development that not following functional classifications by it users will be very costly and refused by the users [4]. Since the commercial smart home system is usually expensive, it is possible now to build a smart home system with various kind of single-board computer known as Do-it-Yourself (DIY) home automation.With DiY home

automation, it is possible to create a smart home system based on the requirement of it users and will be beneficial for the smart home users and the developers [5].

Smart home technologies are now leaving interactions with conventional and manually configured electronic equipment by connecting all the electronic devices in the home through the smart or intelligent devices. Smart home appliances and it services will be automatically discovered by a pervasive technology that now is considered as essential features of a smart home system to face the user friendliness feature as one of the main challenges in smart home appliances [6]. Several types of research had been conducted to add pervasive features in the smart home such as pervasive surveillance system [7], sensor [8], and healthcare [9]. Pervasive technology will make users not need to be bothered with the smart home appliances configuration and setup of the control mechanism. The Pervasive system will handle all the device interaction with emphasis to the three main components, the registration of the service, the service discovery and interaction [10]. In the TCP/IP network, There are several numbers of protocol that able to recognize device pervasively such as Jini (now called Apache River) [11], UPnP [12], Alljoyn [13], etc. In this paper, the system use UPnP since it has auto IP and use a standard protocol such as HTTP, XML, and SOAP for device discovery, device description, and control. UPnP protocol is well designed for small computing environments such as a small home or office networks [10]. There is also possibility smart home system communication interface is using other communication protocols outside TCP/IP network, such as Xbee [14], Bluetooth [15], etc. Xbee is now considered as communication protocol enhancement from 802.15.4 protocol that has essential features such as fast and reliable communication, automatic channel select, and addressing [16].

This paper focuses on the design of a pervasive system in the smart home by integrating pervasive protocol in TCP/IP network and a sensor network to be able to recognize by the smart home user with minimum human intervention and provide control to the smart home appliances. This paper presents a pervasive system design for home appliance by integrating microcomputer device Raspberry Pi with UPnP protocol to control a smart home appliance with smart lamp features and sensor reading as an example. The smart lamp control provided via TCP/IP network. The sensor data will send via Xbee communication and will be pervasively connected to the user that request sensor information. We integrate a GUPnP [17] (UPnP framework based on C Programming) with Raspberry Pi GPIO (General Purpose Input Output) [18] that able to switch the lamp, changing color by RGB, dimmer functions and read temperature sensor.

## 2. RESEARCH METHOD

UPnP is one of the popular pervasive protocol built on top of UDP, TCP/IP, HTTP, and XML. UPnP made the communication standard between a controller (control point) and the devices. UPnP Protocol described five-step of communication consist of discovery, description, control, and eventing. The discovery and description process made devices advertised their existence to the control point via a network and made control point able to recognize the device description and service information. Control and eventing process regulate how the control point sent control data to the device based on the information in description process and received an updated status for each service to any control point that subscribed to the service provided [12]. and service description UPnP Standard [19] in XML format. The framework of device description is written in XML format and presented in Figure 1.

```
1.  . . . . . .
2.      <?xml version="1.0"?>
3.      <root xmlns="...">
4.      ...
5.        <specVersion>
6.        <device>
7.          ...
8.          <serviceList>
9.            <service>
10.             <serviceType> …… </serviceType>
11.             <serviceId> …… </serviceId>
12.             <SCPDURL> …… </SCPDURL>
13.             <controlURL> …… </controlURL>
14.             <eventSubURL> …… </eventSubURL>
15.           </service>
16.         </serviceList>
        </device>
    </root>
. . . . . .
```

Figure 1. Device Description Template

The device & service list description read service description files that specifically explained the service action. The service description files are written in XML format showed in Figure 2, we designed the service description files (Following UPnP Standard [20]) for each service which is switching, dimming, change RGB colors and read the sensor temperature.

```
1.  <?xml version="1.0"?>
2.  <scpd xmlns="urn:schemas-upnp-org:service-1-0">
3.    <specVersion>
4.      <major>1</major>
5.      <minor>0</minor>
6.    </specVersion>
7.    ….
8.  </scpd>
9.  <scpd>
10.    …
11.   <actionList>
12.     <action>
13.       <name> … </name>
14.       <argumentList>…</argumentList>
15.     </action>
16.      ……
      </actionList>
    <scpd>
```

Figure 2. Service Description Template

### 2.1. System Design

We propose our system design presented in Figure 3. Our System focuses on the system device development based on Raspberry Pi single-board computer system that able to found and control pervasively by UPnP Client or known as the control point. We use Generic UPnP Client such as UPnP Spy developed by Intel to control all of our system features such as switching lamp, dimming, change color, and read sensor information via Xbee modules. The RGB LED put in GPIO pin 4 for the Red Color and use 68 Ohm Resistor, the blue color put in GPIO pin 5, while Green Color is in GPIO 6 along with 5.6 Ohm Resistor. The RGB led [21] have a 1.9 V forward voltage for the red color, 3.3 V for the blue color, and 3.2 V for the green color with 20 mA forward current test condition. The GPIO voltage are 3.3V [18] and to make the RGB led live longer we need to limit the current flow from GPIO to RGB led. The resistor measurement is using Ohm's Law. Since 70 Ohm and 5 Ohm resistor are not able to find in the market, we use the closest value which is 68 Ohm and 5.6 Ohm.

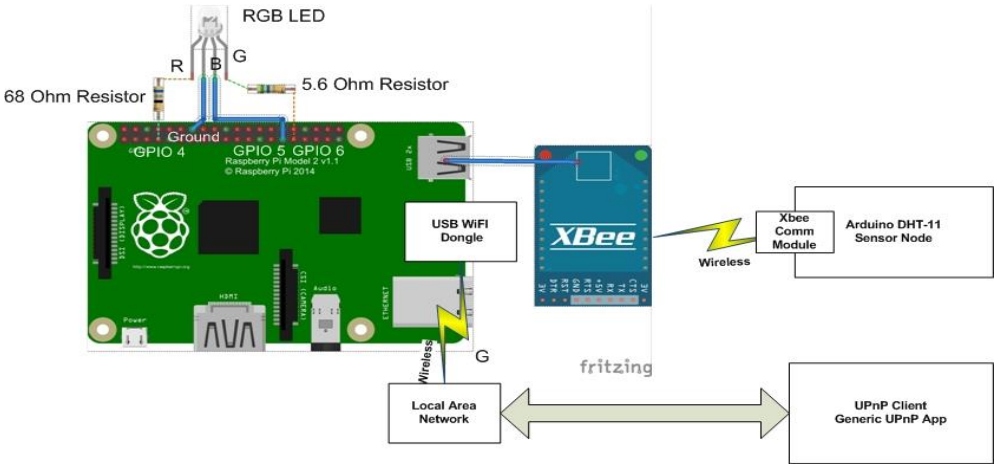


Figure 3. System Design

Our system also equipped with Xbee USB connected to a USB port in Raspberry Pi to read the temperature sensor value wirelessly from a sensor node. We use 1AAA PAN ID (Personal Area Network ID)

in Xbee network and communicate each other using broadcast method. The Device system connected with Wireless local area network using USB WI-FI dongle and located in the same network with the control point. The sensor DHT 11 will be able to be recognized by the UPnP control point.

There are two main parts of the system software. The first part is UpnP modules (UPnP handler) that contain an advertising service, control and event function for UPnP protocol, and the second part is the hardware modules (Hardware Handler) containing GPIO function to manage the smart lamp & temperature sensor features. The device description is filled with device and service list entities shown in Table and Table 2. The device description listed the device type, friendly name, manufacturer and model name. For the service list, we define there is four main service list represent smart home control that is switching (binary output on and off), Dimmer (analog output control), RGB (analog output control) and temperature sensor information (analog input).

Table 1. Device Description

| Elemen | Tipe Data | Nama |
|---|---|---|
| deviceType | Single URI | urn:schemas-upnp-org:device:BinaryLight:1 |
| friendlyName | String | LUPnP |
| manufacturer | String | Computer Engineering FILKOM UB |
| modelName | String | MTH Smart Lamp Version 1.0 |
| UDN | Single URI | uuid:cc93d8e6-6b8b-4f60-87ca-228c36b5b0e8 |

Table 2. Switching, Dimming, Color Change, and Temperature Reading Service List

| Service Name | Element | Value |
|---|---|---|
| Switching Service | serviceType | urn:schemas-upnp-org:service:SwitchPower:1 |
| | serviceID | urn:upnp-org:serviceId:SwitchPower:1 |
| | SCPDURL | Resource/SwitchPower1.xml |
| | controlURL | Resource/SwitchPower/Control |
| | eventSubURL | Resource/SwitchPower/Event |
| Dimming Service | serviceType | urn:schemas-upnp-org:service:Dimming:1 |
| | serviceID | urn:upnp-org:serviceId:Dimming:1 |
| | SCPDURL | Resource/Dimming1.xml |
| | controlURL | Resource/Dimming/Control |
| | eventSubURL | Resource/Dimming/Event |
| Change Color | serviceType | urn:schemas-upnp-org:service:ColorChange:1 |
| Service | serviceID | urn:upnp-org:serviceId:ColorChange:1 |
| | SCPDURL | Resource/ColorChange1.xml |
| | controlURL | Resource/ColorChange/Control |
| | eventSubURL | Resource/ColorChange/Event |
| Sensor Status Service | Service Type | urn:schemas-upnp-org:service:SensorLm:1 |
| | Service ID | urn:upnp-org:serviceId:SensorLm:1 |
| | SCPDURL | Resource/SensorLm1.xml |
| | controlURL | Resource/SensorLm/Control |
| | eventSubURL | Resource/SensorLm/Event |

### 2.2. Switching, Dimming, and Change Color Service Structure

Switching service list is created for switching purpose. we provide a Boolean data type since it has only two action which is switch on and switch off, details for this services presented in Table. setTarget action list used when there is an action from control point to change the switch status off to on or vice versa. GetStatus action list handled when there is a request from control point to ask the current switch status. Switching services had a status and target variables wich default set to 0 (turn off) to handle the data shown in Table 4.

Next Service created is dimming functions. This action list defined LoadLevelTarget and LoadLevelStatus action list. This Action List will be able to handle the Pulse Width Modulator input which has a 0 – 255 value and gives a status to control point. To control dimming function in RGB Led, we use Pulse Width Modulation technique and create it as a function in Raspberry Pi with 8-bit value (0-255) and convert it to duty cycle 0-100%. The equation is shown in Equation :

*Equation 1 Pulse Width Modulation Equation*

$$PWM = \frac{(Value - Input_{\min}) \times (Output_{\max} - Output_{\min})}{(Input_{\max} - Input_{\min})} + Output_{\min}$$

$PWM =$ Pulse width modulation in percentage

$Value =$ Decimal input value from control point (minimal 0 – max 255)
$Input_{\min} = Minimum\ V$alue in 8 bit decimal (default 0)
$Input_{\max} = Maximum\ Value\ in\ 8\ bit\ decimal\ (default\ 255)$
$Output_{\min} = Ouput\ minimum\ duty\ cycle\ in\ percentage\ (default\ 0\%)$
$Output_{\max} = Output\ maxmimum\ duty\ cycle\ in\ percentage\ (default\ 100\%)$

Details dimming service list and the data type presented in Table and Table 6 SetLoadLevelTarget action used when there is a control point want to change the lamp dimmer. GetLoadLevelStatus action used when control point requests the current status for lamp dimmer.

Change Color Services list and data type as stated in Table 7 and Table 8, consist of two action list which is setColorChangeTarget and GetColorChange Status. SetColorChangeTarget used when control point sends a value to change the lamp color. Control point needs to give three value to change the lamp color which is red value, a green value, and the blue value. SetColorChangeTarget will bring the three color value to the device and GetColorChangeStatus will inform control point information about the current color lamp. Red, green and blue value have a 1 Byte data type which is 0-255. For example, we want to change the Lamp color to white, the control point must be sent 255 value for red, 255 value for green, and 255 value for blue. Since the value is 1 Byte, we declare the datatype for ColorRedTarget, ColorGreenTarget, and ColorBlueTarget with ui1 that has 1 Byte data from 0 to 255.

Table 3. Switching Service List

| ActionList | ArgumentList | | |
|---|---|---|---|
| | Name | Related StateVariable | Direction |
| SetTarget | newTargetValue | Target | in |
| GetStatus | ResultStatus | Status | out |

Table 4. Switching Data Type

| serviceStateTable | Data Type | Default value | event |
|---|---|---|---|
| Target | boolean | 0 | No |
| Status | boolean | 0 | Yes |

Table 5. Dimming Service List

| actionList | argumentList | | |
|---|---|---|---|
| | Nama | relatedStateVariable | direction |
| SetLoadLevelTarget | newLoadlevelTarget | LoadLevelTarget | in |
| GetLoadLevelStatus | retLoadlevelStatus | LoadLevelStatus | out |

Table 6. Dimming Data Type

| serviceStateTable | Tipe data | Default value | Allowed value | | event |
|---|---|---|---|---|---|
| | | | Minimum | maximum | |
| LoadLevelTarget | ui1 | 0 | 0 | 255 | No |
| LoadLevelStatus | ui1 | 0 | 0 | 255 | Yes |

Table 7. Change Color Service List

| actionList | argumentList | | |
|---|---|---|---|
| | Nama | relatedStateVariable | direction |
| SetColorChangeTarget | newRedTarget | ColorRedTarget | in |
| | newGreenTarget | ColorGreenTarget | in |
| | newBlueTarget | ColorBlueTarget | in |
| GetColorChangeStatus | RedStatus | ColorRedStatus | out |
| | GreenStatus | ColorGreenStatus | out |
| | BlueStatus | ColorBlueStatus | out |

Table 8. Change Color Data Type

| serviceStateTable | Tipe data | Default value | Allowed value | | event |
|---|---|---|---|---|---|
| | | | Minimum | maximum | |
| ColorRedTarget | ui1 | 0 | 0 | 255 | No |
| ColorGreenTarget | ui1 | 0 | 0 | 255 | No |
| ColorBlueTarget | ui1 | 0 | 0 | 255 | No |
| ColorRedStatus | ui1 | 0 | 0 | 255 | Yes |
| ColorGreenStatus | ui1 | 0 | 0 | 255 | Yes |
| ColorBlueStatus | ui1 | 0 | 0 | 255 | Yes |

### 2.3. Temperature Sensor Service Structure

We designed sensor action list only have one GetTempStatus action list since sensor only able to bring its information to the control point. We use DHT 11 [22] Temperature sensor for the prototype and DHT 11 sensor have measurement range between 0 – 50 degree Celsius . GetTempStatus will inform control point when the control point requests the current temperature measurement. Sensor service lists and the data type presented in Table 9 and Table 10.

Table 9. Sensor Service List

| actionList | argumentList | | |
|---|---|---|---|
| | Name | relatedStateVariable | Direction |
| GetTempStatus | ResultStatus | Status | Out |

Table 10. Sensor Data Type

| serviceStateTable | Tipe data | Default value | Allowed value | | event |
|---|---|---|---|---|---|
| | | | Minimum | maximum | |
| StatusTemp | ui1 | 0 | 0 | 50 | No |

The Sensor will communicate with a device using Xbee communication protocol based on Zigbee. When sensor finished data acquisition, the sensor will send the data to the device and the device will keep the data inside a file. When Control point asks to the current temperature sensor, the device will read the current data from the file and inform current temperature sensor to control point.

### 2.4. System Implementation

We use GUPnP framework [17] for implementing UPnP protocol to Raspberry Pi Devices and since GUPnP is an event triggered framework, we create a model based on State Chart diagram shown in Table 4. Inside the loop, we declare there is 5 state which are Idle State, Dimming Control State, Switch Control State, Color RGB Control and Sensor Status. Idle State created to make system waiting for an event to be called by Control Point. Idle State will move to Switch Control State if there is event on_set_target and call a GUPnP_service_action_get(target) to get the value based on what control point needed. Switchcontrol(target) is an action to pass the target value to hardware handler part filled with an application to activate the GPIO in Raspberry Pi. After On_set_target event finished, Gupnp_service_notify will be called and the Raspberry Pi will send a notification to the control point declared that the Switch has been turned on. While control point received the notification, the switch control state turned back to idle state. The System will turn from idle state to dimming control state, if there is an event on_set_load_level_target() function was called. This event occurred if there is an action request from the control point that wants to change the lamp dimmer, gupnp_service_action_get (newLoadlevelTarget) will execute to get the value of dimmer value and execute dimmingControl (loadlevel) to give a control command to change the GPIO value in Raspberry Pi. The dimmer value will be filled in all Red, Green, Blue parameters and sent it to GPIO as shown in Table 5.

Color RGB Control state is triggered if there is a system event called on_set_color_target. This event will get Red, Green, and Blue value and pass it to colorcontrol() function. The colorcontrol() function is a hardware handler function to proceed Red, Green, and Blue value to GPIO. The Color RGB control state will be changed back to idle state after UPnP handler application sent gupnp_service_notify() to control point. Sensor reading state triggered when a control point requests a sensor information. Control point request will recognize from on_get_temp_status() function and the hardware handler will read a text file and get the newest line in the file which is filled with a temperature data from the sensor node. Sensor reading state will back to idle state when gupnp_service_action_return(action) function is sent the sensor value to control point.

Figure 5 present the hardware handler flowchart contain switch GPIO control, Dimmer GPIO control, Change Color GPIO control, and sensor reading. It described a Raspberry Pi function after receiving a service request from the control point and then executed the GPIO. The switching GPIO flowchart described a process after the Raspberry Pi received UPnP message contain change the switching status. If the Boolean switching status is changed to 1 the GPIO will turn on the lamp and if the switching status is change to 0 GPIO will turn off the lamp. The dimmer flowchart described the process after the Raspberry Pi received a change dimming value (0-255) from the UPnP message. The dimmer value will setup the PWM duty cycle and write the PWM value to GPIO and setup the brightness level in the lamp. The change color flowchart described a process after the Raspberry Pi received a change value in RGB from the UPnP message (0-255 RGB value) and write to the GPIO process. Sensor reading flowchart described that the Raspberry Pi received a sensor value periodically and write it to the Sensor.txt file, if control point requests the sensor value, the Raspberry Pi will read the newest line to the txt files. For all services, Raspberry Pi will send a notification feedback to the UPnP control point to make the user know that the change or request for each service has successfully proceeded.

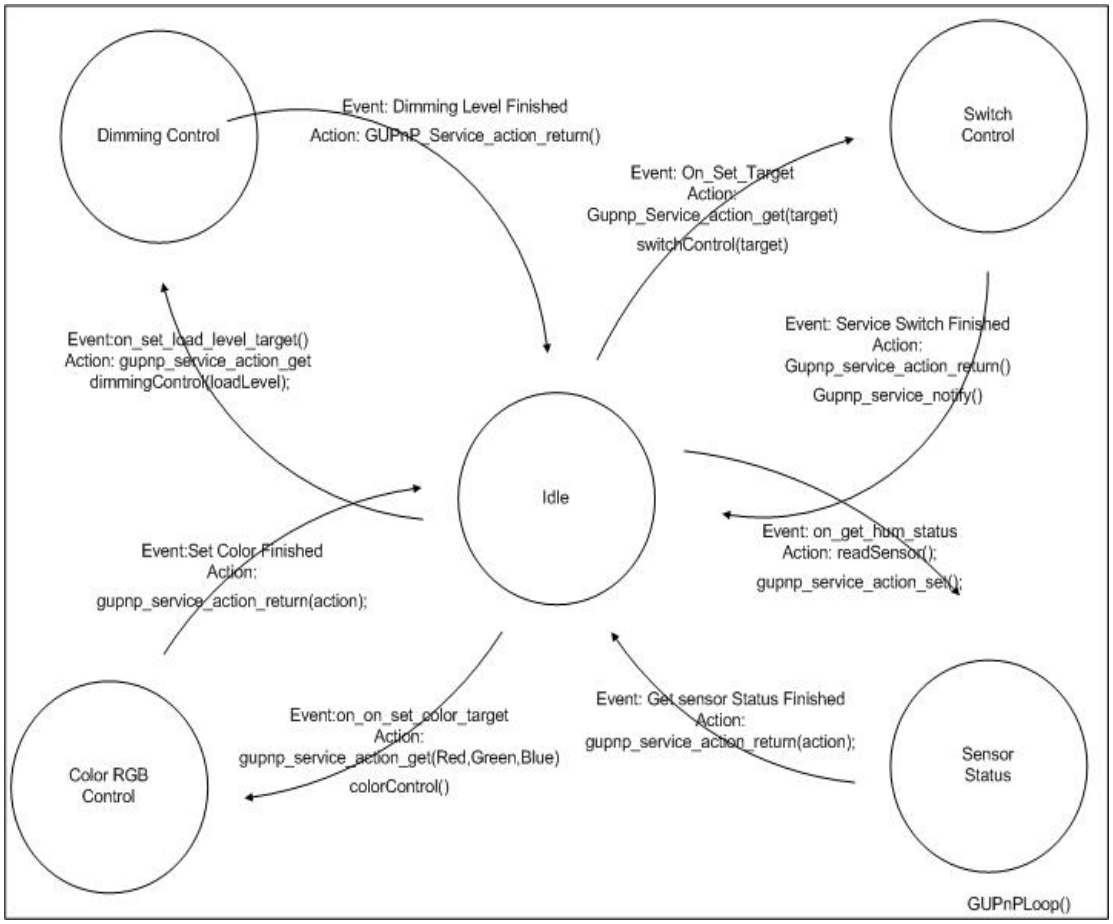


Figure 4. StateChart Diagram Control services For UPnP Handler Application

This paper conducted several validation scenarios, which is functional testing to test whether Raspberry Pi is able to find and control pervasively by control point device. After the Raspberry Pi turned on, the raspberry will broadcast its presence to the network. When the broadcast message are received by the control point, the control point will recognize the device and service description which have the switch, dimming, color RGB, and sensor reading services. Non-functional test conducted to know the performance in round-trip time and GPIO processing in the Raspberry Pi. For response time test, we measure time consumed when Raspberry Pi is processing control via UPnP protocols and forward to GPIO. We use general control point UPnP Spy created by Intel to find and control the Raspberry Pi GPIO, and measure the service response time using Wireshark packet sniffer.

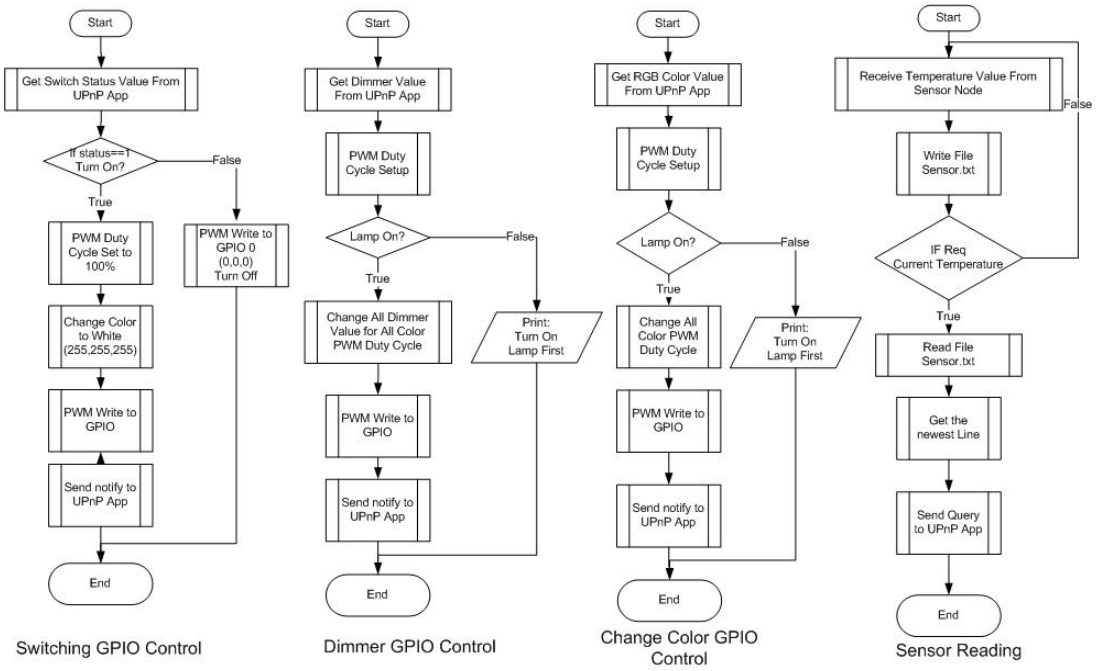


Figure 5. Hardware Handler Application

## 3. RESULTS AND ANALYSIS

The smart home functions are classified into entertainment, home security, and home automation. In home automation function, a smart home user is having attitude negatively related to cost and want to have a simple system that able to control the appliances [4]. The attitude negatively related to cost is caused by commercial smart home system that usually expensive, and since our system is built on top of single-board computer that has a DiY features, it is possible to create a smart home system based on the user requirement so user and developer are able to adjust the cost of the system based on what user really need. This paper also able to enrich the features of the smart home in automation functions wich makes the smart home control simpler since the appliances will be able to recognize pervasively by the user control device, so the user will be able to focus on the control rather than configuring IP address for each of the appliance devices. The device is able to recognize pervasively by the UPnP Spy application shown in Figure 6.

| Name | Value |
|---|---|
| Base URL | http://192.168.137.87:49789/ |
| Device icon | None |
| Device URN | urn:schemas-upnp-org:device:BinaryLight:1 |
| Embedded devices | 0 |
| Expiration timeout | 1800 |
| Friendly name | Pervasive Smart Home |
| Has presentation | False |
| Interface to host | 192.168.137.123 |
| Manufacturer | Computer Engineering UB |
| Manufacturer URL | |
| Model description | |
| Model name | Raspberry Pi GPIO, UPnP, Zigbee |
| Model number | |
| Presentation URL | |
| Product code | |
| Proprietary type | |
| Remote endpoint | 192.168.137.87:49789 |
| Serial number | |
| Services | 5 |
| Standard type | BinaryLight |
| Unique device name | cc93d8e6-6b8b-4f60-87ca-228c36b5b0e8 |
| Version | 1.0 |

Figure 1. UPnP Spy Generic Control Point

### 3.1. Round-Trip Time Performance

Service response time is measured by marking Round Trip Time Packet from HTTP request sent by UPnP Spy control point application and HTTP response from the Raspberry Pi. Round Trip Time is measured by subtracting time in HTTP response with HTTP request. An Example of service control packet and Round Trip Time presented in Table 1. Packet number 1554 identified as HTTP Request packet with POST method sent by UPnP Spy control point to remotely switch the GPIO in Raspberry Pi. Packet number 1560 identified as HTTP Response with 200 OK Response Code from Raspberry Pi. Time difference between packet is calculated as Round Trip Time in (ms) as stated in Table 1. Round-trip time measurement is conducted in every service in the system which is switching, dimming, RGB, and sensor reading services.

Table 1. Service Control Packet

| Packet Number | Time | Source IP | Destination IP | Protocol | Length | Info | Round Trip Time (ms) |
|---|---|---|---|---|---|---|---|
| 1554 | 273.100403 | 192.168.137.123 | 192.168.137.87 | HTTP/XML | 607 | POST /Resource/SwitchPower/Control HTTP/1.1 | |
| 1560 | 273.161113 | 192.168.137.87 | 192.168.137.123 | HTTP/XML | 325 | HTTP/1.1 200 OK | 0.06071 |

Figure *2* present each service round trip time. The average service round trip time for switching services is 0.1208056 ms, dimmer service is 0.072821 ms, change color service is 0.102671 ms, and 0.620716 ms for sensor services.

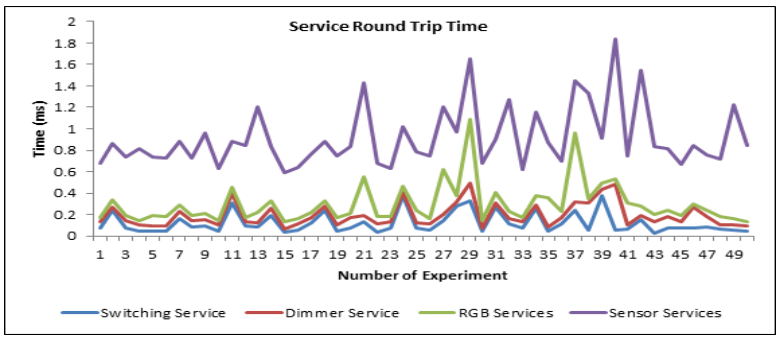


Figure 2. Service Response Time

### 3.2. General Purpose Input Output Processing Time

The GPIO processing time measured in an application based on clock timer function shown in Figure 3. When control point call a dimmer function, function on_set_load_level_target() is called and the clock start to count. Dimmingcontrol() function will execute the GPIO processing at raspberry Pi and right after the UPnP handler sent service_action_return() is called the timer will stop and print the time spent by the services. GPIO processing time measure all lamp services in switching, dimming and RGB services.

```
G_MODULE_EXPORT

void on_set_load_level_target(GUPnPService *service,
GUPnPServiceAction *action, gpointer user_data){

double timeSpent;

begin = clock();//Time begin

gupnp_service_action_get(action, "newLoadlevelTarget",
G_TYPE_UINT, &loadLevel, NULL);
```

Figure 3. GPIO Processing Measurement Metho

GPIO processing time result presented in Figure 9 showed that switching services average processing time is 0.00090878 ms, dimming service 0.00122508 ms, and RGB switching service 0.00154826.

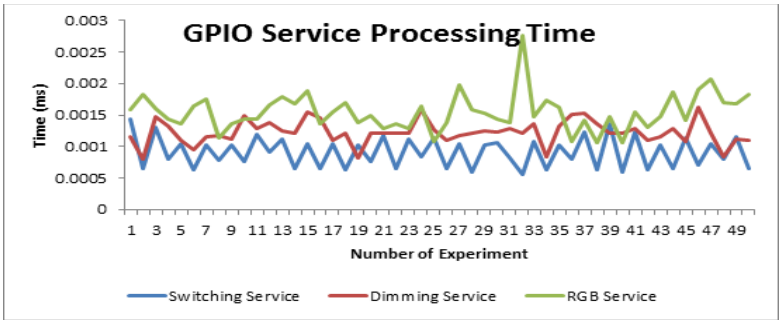


Figure 4. GPIO Service Processing Time

## 4. CONCLUSION

This paper presented a smart home system design with a single-board computer system that enriching automation features by adding pervasive device and service discovery protocol. Our system combined UPnP protocol, Raspberry Pi and Xbee to create a pervasive system device in smart home appliance with RGB LED control and sensor reading as an applications example. The system uses UPnP protocol to make device pervasively recognized by the control point, and able to control the RGB LED features such as switching, dimming, and change color via GPIO control in Raspberry Pi. Control point also able to find and read the temperature sensor data via Xbee communication connected to a sensor system with DHT 11 as a temperature sensor. UPnP Processing time is measured with UPnP round trip time by subtracting HTTP response arrival with HTTP request time. GPIO processing time measured at the application level by counting a timer that starts before GPIO process and ended after GPIO successfully executed. The Result showed that in a wireless network, the average service round trip time for switching services is 0.1208056 ms, dimmer service is 0.072821ms, change color service is 0.102671 ms, and 0.620716 ms for sensor services. GPIO processing time for switching services is 0.00090878 ms, dimming service 0.00122508, RGB switching service 0.00154826 ms, and sensor reading is 0.00099738 ms.

## REFERENCES

[1] L. Atzori, *et al.*, "The Internet of Things: A survey," *Computer Networks*, vol/issue: IV(54), pp. 2787-2805, 2010.
[2] C. Ryu and C. W. Hur, "A Monitoring System for Integrated Management of IoT-based Home Network," *International Journal of Electrical and Computer Engineering (IJECE)*, 2016.
[3] K. Rose, *et al.*, "The Internet of Things: An Overview Understanding the Issues and Challenges of a More Connected World," *The Internet Society (ISOC)*, 2015.
[4] T. Luor, H.-P. Lu, H. Yu e Y. Lu, "Exploring the critical quality attributes and models of smart homes," *Maturitas*, vol. 82, pp. 377-386, 2015.
[5] V. Vujovic and M. Maksimovic, "Raspberry Pi as a Sensor Web node for home automation," *Computers and Electrical Engineering*, vol. 44, pp. 153-171, 2015.
[6] T. Adiono, *et al.*, "Rapid Prototyping Methodology of Lightweight Electronic Drivers for Smart Home Appliances," *International Journal of Electrical and Computer Engineering (IJECE)*, 2016.
[7] A. Longheu, *et al.*, "An Intelligent and Pervasive Surveillance System for Home Security," *International Journal of Computers Communications & Control*, pp. 312-324, 2012.
[8] G. Stratogiannis, *et al.*, "User and home appliances pervasive interaction in a sensor driven smart home environment: The SandS approach," em *Semantic and Social Media Adaptation and Personalization (SMAP)*, Trento, 2015.
[9] H. Medjahed, *et al.*, "A pervasive multi-sensor data fusion for smart home healthcare monitoring," em *Fuzzy Systems (FUZZ)*, Taipei, 2011.
[10] N. Bhatti, *et al.*, "Service discovery protocols in Pervasive Computing: A review," em *Multi-Topic Conference (INMIC), 2014 IEEE 17th International* , Karachi, 2014.
[11] Apache Software Foundation, "Jini Architecture Specification," [Online]. Available: https://river.apache.org/about.html. [Accessed 05 May 2016].
[12] Open Connectivity Foundation, "About UPnP," [Online]. Available: http://openconnectivity.org/upnp. [Accessed 2 May 2016].

[13] A. Alliance, "AllJoyn Documentation," [Online]. Available: https://allseenalliance.org/framework/documentation. [Accessed 2 May 2016].
[14] S. P. Lim and G. H. Yeap, "Centralised Smart Home Control System via XBee," em *IEEE Colloquium on Humanities, Science and Engineering Research (CHUSER)*, Penang, 2011.
[15] Z. Yufeng and J. Ruqiao, "Design and Realization of the Smart Home Control System Based on the Bluetooth," em *Intelligent Transportation, Big Data and Smart City (ICITBS)*, Halong Bay.
[16] Digi, "Xbee," Digi, [Online]. Available: http://www.digi.com/lp/xbee. [Accessed 10 August 2016].
[17] Gnome Developer, "GUPNP Reference Manual," 2009. [Online]. Available: https://developer.gnome.org/gupnp/unstable/. [Accessed 9 August 2014].
[18] Raspberry Pi Foundation, "GPIO: Raspberry Pi Models A and B An introduction to GPIO and physical computing on the Raspberry Pi," Raspberry Pi Foundation, [Online]. Available: https://www.raspberrypi.org/documentation/usage/gpio/README.md. [Accessed 22 04 2016].
[19] Open Connectivity Foundation, "BinaryLight:1 Device Template Version 1.01," Open Connectivity Foundation UPnP, 2003.
[20] Open Connectivity Foundation, "SwitchPower:1 Service Template Version 1.02," Open Connectivity Foundation UPnP, 2011.
[21] Kingbright, "T-1 3/4 (5mm) FULL COLOR LED LAMP DATA SHEET," Kingbright, 2010.
[22] Sunrom Technologies, "DHT-11 Humidity and Temperature Sensor," Sunrom Technologies, 2012.