**Bulletin of Electrical Engineering and Informatics**
Vol. 7, No. 4, December 2018, pp. 615~625
ISSN: 2302-9285, DOI: 10.11591/eei.v7i4.1199 

# Curtain Control Systems Development on Mesh Wireless Network of the Smart Home

**Trio Adiono, Sinantya Feranti Anindya, Syifaul Fuada, Maulana Yusuf Fathany**
University Center of Excellence on Microelectronics, Institut Teknologi Bandung,
IC Design laboratory, PAU Building 4th floor, ITB Campus, Jln. Tamansari No.126, Bandung (40132), Indonesia

**Article Info**

***Article history:***

Received Apr 15, 2018
Revised May 20, 2018
Accepted Nov 11, 2018

***Keywords:***

Bluetooth
Curtain control
Smart home
Wireless Sensor Network
Zigbee

**ABSTRACT**

In this paper, a curtain controller for smart home is presented. The aim of this work is to develop an end device in smart home system that will support power conservation function indirectly, specifically a curtain open/close controller. To achieve this, the 28BYJ-48 stepper motor is used as actuator with the assistance of ULN2003A driver. The motor is controlled using STM32L100RCT6 microcontroller, which is chosen due to its low power consumption. The microcontroller controls the motor's direction by using a pulse width modulation logic signals emitted from four GPIO pins, which works based on data transmitted from central host through ZigBee protocol on Mesh network. Meanwhile, from the user's side, the control is done by using Android-based application, which is connected to central host through Bluetooth. Based on the testing conducted on a miniature curtain, the curtain can be controlled wirelessly through the Android application. Furthermore, the device consumes power 210.5 mW for idle condition and 1,586 mW for process condition. The amount of the consumed power makes it suitable for low-power operation and in alignment with the smart home system's overall aim for power conservation in the wireless sensor network-based smart home system.



***Corresponding Author:***

Trio Adiono,
Institut Teknologi Bandung ,
University Center of Excellence on Microelectronics,
Jln. Tamansari No.126, Bandung,Indonesia
E-mail: tadiono@stei.itb.ac.id

## 1. INTRODUCTION

A smart home system is a system that consists control and management functions within a home environment intended for enhancing living quality of its inhabitants through functions such as home appliances control and monitoring and camera-based security. Smart home system generally utilize technology such as Internet-of-Things and wireless sensor network to enable functions such as remote controlling and monitoring, thus enhancing convenience for a house's inhabitant. Smart home system can generally be categorized into at least one of four types based on its primary function, namely energy, security, entertainment, and health care [1].

One of the most common applications of smart home system is for energy management purposes, such as power monitoring and energy conservation. These functions are supported by including devices for whether direct control and monitoring such as smart plug [2] and monitoring system [3], or indirect control by manipulating the environment. One of the means for the indirect control is by controlling the curtains or blinds, which serves to regulate lighting and heat within the house. A curtain serves as cover against sun, to manage air circulation, and decorative purposes. In general, a curtain can be classified as decorative element or fully-operable item for managing the room. The fully-operable curtain itself consists several types, such as roller blind, Roman blind, horizontal blind, vertical blind, and wooden blind [4].

Within some existing smart home systems, the controller integrated into the system is usually used for controlling horizontal blind Figure 1a and vertical blind Figure 1b curtains. Horizontal blind curtains are normally used within office perimeter, while residential area primarily uses vertical blind curtains. The blinds consist fin-like structure made from materials such as aluminum, wood, or textile materials such as cotton. The curtain is intended to assist with power conservation tasks albeit indirectly, such as temperature control and managing when lamps should be activated. There are several methods to control curtain through smart home system, such as wirelessly using radio frequency [5][6], SMS gateway [7], Bluetooth [8], or fully automated without remote control [9].

In the previous work, a prototype for wireless connectivity-based horizontal blind control has been established [10-11]. The curtain controller itself is one of the end-nodes of a wireless smart home structure as depicted in Figure 2, which consists an outdoor sub-system and indoor sub-system. The outdoor sub-system consists elements those cannot be accessed through WSN and requires Internet connection such as cloud server and mobile application, while the indoor sub-system consists devices within the WSN. Further explanation on the WSN is detailed on [12] and [13]. Aside from the sub-systems categorization, the elements of the proposed smart home network are divided into four types: end devices, nodes, host, and user interface. Beside the curtain controller, there are several other types of end devices developed for the proposed smart home system, namely temperature and humidity sensor [14], RGB LED lamp [15], power switcher [16], door lock [17], fan controller [18], and infrared remote controller [19].

(a) (b)

Figure 1. The type of curtain: (a) horizontal blind; (b) vertical blind.
The illustration is obtained from http://nazmagorden.yukbisnis.com

In this paper, the extension of previous work for curtain control is presented. The end result of this work is intended to be ready for implementation for real-life use case, in which an actual curtain can be controlled wirelessly to slide up or down using mobile device. To that end, there are several things those need to be considered: a) basic control function support, b) efficient (low-power and low-cost) hardware and software design, c) flexibility and mobility of device for ease of implementation.

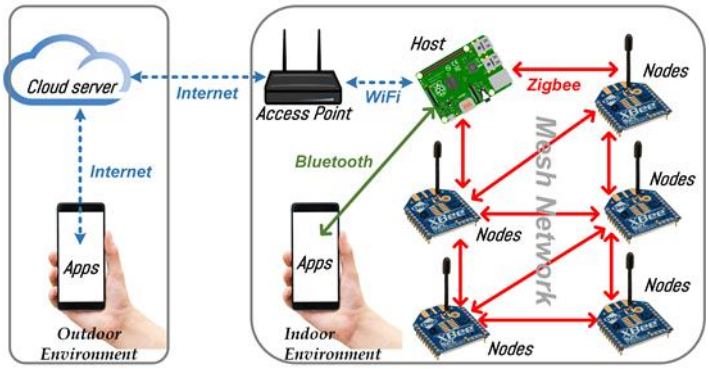



Figure 2. Proposed WSN system, edited from [12] and [13] with permission

## 2. METHODOLOGY

### 2.1. System Specification

The architecture of the device is depicted in Figure 3. The designed device uses stepper motor for controlling the curtain; as such, digital output control is required. To support such task, the microcontroller needs to be able to support low-power operation and have the following interfaces: at least four ports of general purpose input/output (GPIO), timer, and USART. The controller itself will be powered using 5 $V_{DC}$ voltage through DC adapter. From the user's side, the device is a box-shaped object which will be placed statically (not a mobile device). The device is also equipped with several other interfaces, such as switch, reset and mode buttons, and micro-USB port. Figure 3 shows the control system architecture of the curtain in which the designed scenario is employed for an indoor environment purpose.

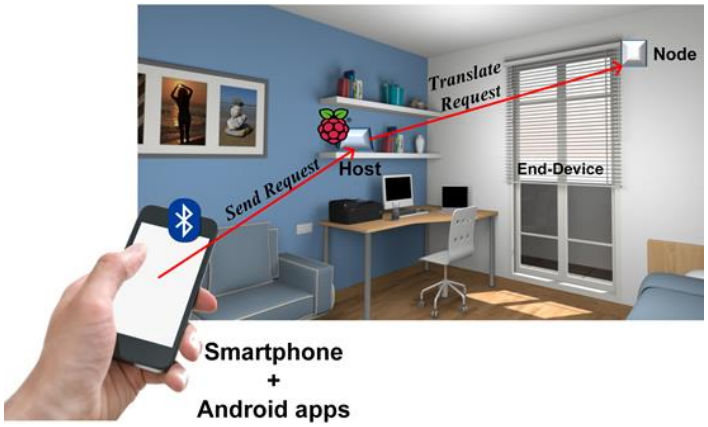



Figure 3. Architecture of the curtain control system

### 2.2. WSN Design

As previously stated, the curtain controller is part of a WSN designed for a smart home system. The WSN is designed using DigiMesh scheme from ZigBee technology. This scheme differs from the standard ZigBee nodes scheme that requires role initialization in each node. The ZigBee node consists one static coordinator that must not enter sleep mode, several routers those serve as 'middlemen' for managing devices' data, and the end devices. Each end device has been programmed to send data to the host address.

In this work, Raspberry Pi 2 is utilized as the host of the WSN. The host serves as WSN gateway where all information and data communication between user and the WSN. Raspberry Pi 2 has four USB sockets and USART GPIO, which are used for connecting the Wi-Fi and Bluetooth modules (USB), as well as Zigbee module (UART). While ideally the host is used for managerial purposes within smart home system's platform layer, in this work the host's function is limited as one of the bridges between the WSN and the user. The user communicates with the host through Bluetooth and/or Wi-Fi, while the host will process the command and relay the result to the WSN via ZigBee.

### 2.3. Processor Selection

Based on the established specification, STM32L100RCT6 microcontroller chip is used as the device's processor due to its low power consumption and cost. The processor is designed using HSI internal clock Figure 4 to minimize external clock (oscillator crystal) usage. The value of the internal clock frequency is 16 MHz.

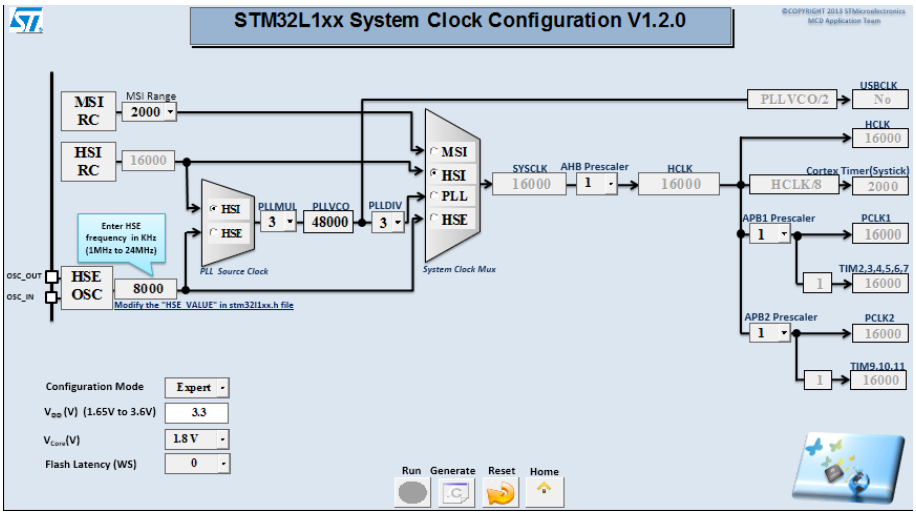



Figure 4. Internal clock configuration of STM32L100RCT6 [16]

### 2.4. Network Devices Selection

In this work, ZigBee XBP9B-DMWT-012 is utilized as communication technology for the WSN's end devices and nodes. The ZigBee technology is selected for its low power, flexibility, and ease of implementation. ZigBee communicates using serial communication interface with the main processor. To ensure extensive range coverage, ZigBee device can act as node in idle state, which in turn also allows data hopping.

### 2.5. Hardware Design

The controller uses 4-wire unipolar stepper motor 28BYJ-48 as actuator, while the driver uses ULN2003A integrated circuit. The intrinsic circuit of IC ULN2003A of the Darlington circuit for current amplifier. The driver circuit is connected with the STM32L100RCT6 microcontroller using four GPIO ports, and the stepper motor is controlled using PWM method by utilizing the logic signals. The hardware structure for the curtain controller is depicted in Figure 5. The driver controls the stepper motor based on PWM, which will turn the motor clockwise (CW) or counter-clockwise (CCW). The direction of the motor is utilized to open or close the curtain.

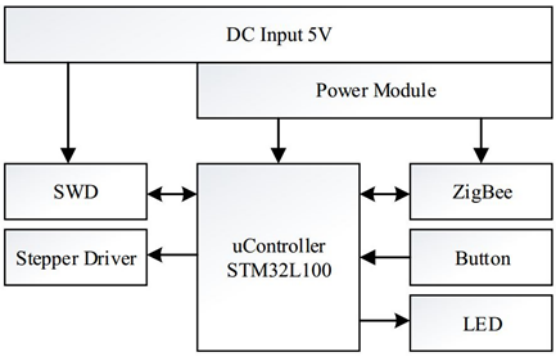



Figure 5. Block diagram of curtain control hardware

### 2.6. Protocol Design

The communication protocol for the device is designed base on the work reported in [21] and [22]. In general, the data packet transmitted in the WSN consists header with 3 bytes length, 2 bytes of device address, packet initialization byte in hexadecimal form, device-specific data payload, and 1 byte for checksum. For the curtain controller, the data payload consists 1 byte of 0 – 100% for the "control" command as shown in Figure 6.

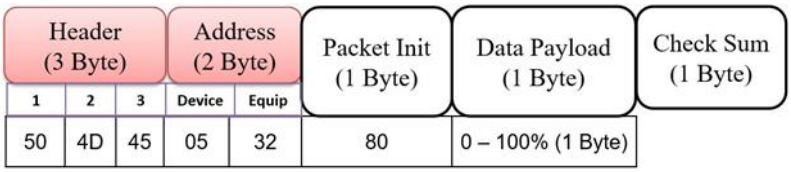



Figure 6. Structure of packet data for curtain device

### 2.7. Software Design

The flowchart design is based on previous work. A few adjustments are made, such as command execution using logic combination of four GPIO ports to control the motor rotation direction. The final flowchart is depicted in Figure 7.

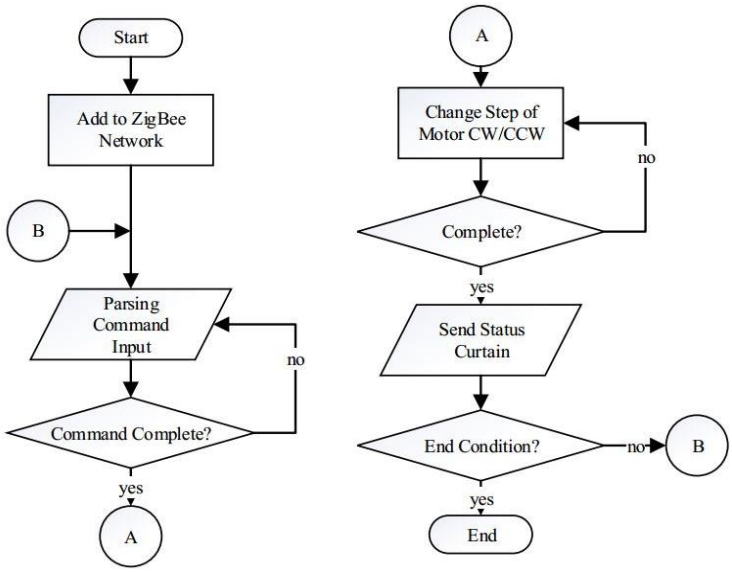



Figure 7. Flowchart of curtain control

### 2.8. Android Aplication Design

The graphical user interface to control the curtain is part of the Android-based mobile application presented in [23] and [24]. The view of the curtain control section is depicted in Figure 8, while the workflow of the control from the application is illustrated in Figure 9.

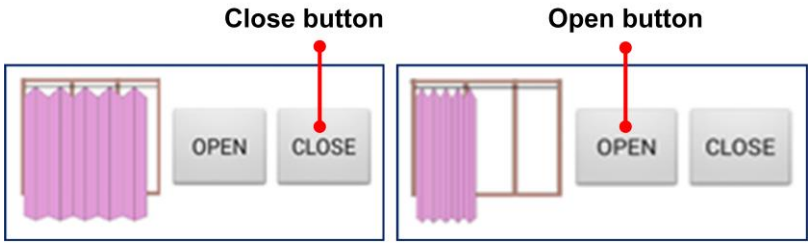



Figure 8. The GUI of ‘close’ and ‘open’ the curtain control, from [23] and [24]

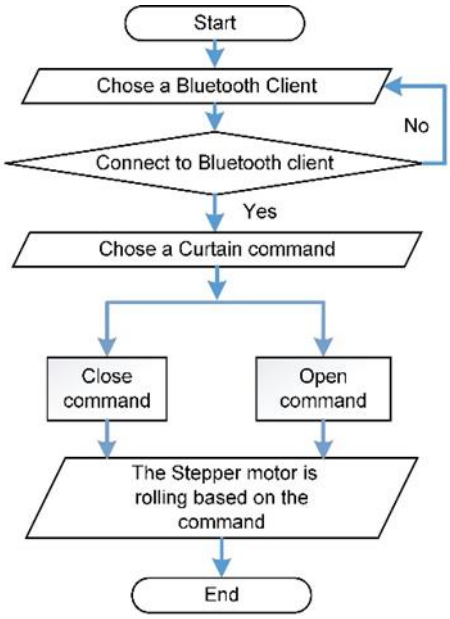



Figure 9. Flowchart of an Android application for controlling the Curtain device based on Bluetooth connection, from [23] and [24]

## 3. RESULT AND ANALYIS

### 3.1. Software Implementation of the SH

The software for the microcontroller is developed using *Keil uVision 5* as shown in Figure 10 and *Atollic TrueSTUDIO for ARM 5.4.1*. To support the task to be implemented, peripheral driver *STM32L1xx_StdPeriph_Driver* is used. The driver is then configured in the main code. Based on the specification from previous section, USART is the primary peripheral to be used for the network communication. As such, additional files in the form of *stm32lxx_usart.c* and *stm32lxx_usart.h* are added to the project. To avoid disturbance to the main computation process, a separate USART handler is required to overtake computation process when interruption in the form of incoming serial data happens. As such, the USART is configured as depicted in Table 1. While the interrupt handler for USART communication requires separate program function declaration. As such, the *USART_IRQHandler* function is written as depicted in Table 2.

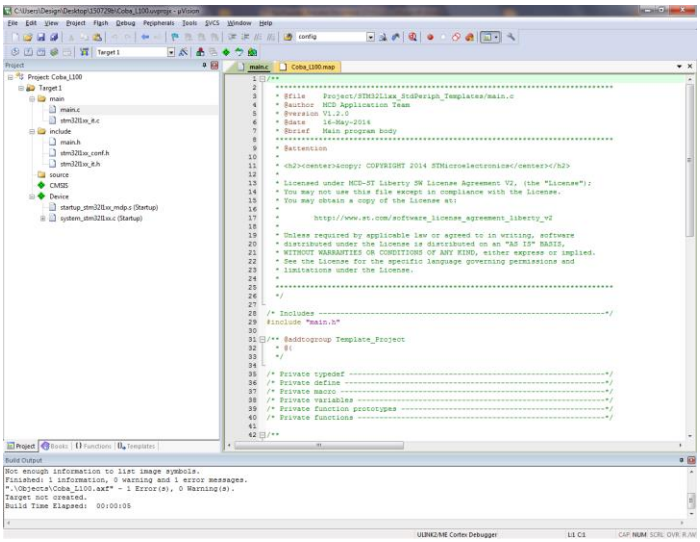


Figure 10. Keil uVision 5 environment

Table 1. Source code for USART configuration

Source code for USART configuration

```
void USART3_Configure()
{
NVIC_InitTypeDef NVIC_InitStructure;
GPIO_InitTypeDef GPIO_InitStructure;
USART_InitTypeDef USART_InitStructure;
/* GPIOC & USART3 clock enable */
RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOC, ENABLE);
RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART3, ENABLE);
/* Enable USART1 IRQ Channel (Vector Interrupt)*/
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_InitStructure.NVIC_IRQChannel = USART3_IRQn;
NVIC_Init(&NVIC_InitStructure);
/* Configure USART Tx & Rx as alternate function */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10 | GPIO_Pin_11;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_40MHz;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
GPIO_Init(GPIOC, &GPIO_InitStructure);
/* Mux out USART1 Tx & Rx */
GPIO_PinAFConfig(GPIOC, GPIO_PinSource10, GPIO_AF_USART3); // TX
GPIO_PinAFConfig(GPIOC, GPIO_PinSource11, GPIO_AF_USART3); // RX
/* USART_InitStruct members default value */
USART_InitStructure.USART_BaudRate = 9600;
USART_InitStructure.USART_WordLength = USART_WordLength_8b;
USART_InitStructure.USART_StopBits = USART_StopBits_1;
USART_InitStructure.USART_Parity = USART_Parity_No ;
USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
USART_InitStructure.USART_HardwareFlowControl =
USART_HardwareFlowControl_None;
/* USART configuration */
USART_Init(USART3, &USART_InitStructure);
/* Enable USART */
USART_Cmd(USART3, ENABLE);
/* Enable Receiver Interrupt */
USART_ITConfig(USART3, USART_IT_RXNE, ENABLE);
}
```

Table 2. Source code for USART handler function

Source code for USART handler function

```
void USART3_IRQHandler(void)
{
volatile unsigned int IIR;
IIR = USART3->SR;
if (IIR & USART_FLAG_RXNE) { // read interrupt
USART3->SR &= ~USART_FLAG_RXNE; // clear interrupt
}
DataUsart3 = USART3->DR;
//GPIO_ToggleBits(GPIOC, GPIO_Pin_9); // green led
}
```

### 3.2. Software Implementation of the Curtain Control

The software implementation of the curtain controller is based on the I/O logic combination to activate the driver of the stepper motor. To set the speed of stepper motor, a delay function is added for every change of step. The delay value is based on the SysTick handler's value, which by default is set to 1 milisecond. The accuracy of 28BYJ-47 stepper motor movement is 5.625°/64, meaning that in every step there is angle change of 0.087890625°. As such, 4096 steps are required for one full rotation. To allow this, four GPIO pins are used for controlling the stepper motor, namely pins 6, 7, 0, and 1. In this work, one run of the procedure will move the motor by 8 steps. As such, 512 runs (loops) will be required for the motor to perform one full rotation. The GPIO configuration is depicted in Table 3, while the configuration of the GPIO logic combination is depicted in Table 4.

Table 3. Source Code for Configuration of the GPIO

| Source Code for Configuration of the GPIO |
|---|
| // GPIO Init<br>GPIO_InitTypeDef GPIO_InitStructure; // Initialization Structure<br>RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOC, ENABLE); // Enable GPIOC clock<br>// GPIO Configuration<br>GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 \| GPIO_Pin_7;<br>GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;<br>GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;<br>GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;<br>GPIO_Init(GPIOC, &GPIO_InitStructure);<br>RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOB, ENABLE); // Enable GPIOC clock<br>GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 \| GPIO_Pin_1 ;<br>GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;<br>GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;<br>GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;<br>GPIO_Init(GPIOB,&GPIO_InitStructure); |

Table 4. The software design for the command of logic combination on the stepper motor

| software design for the command of logic combination on the stepper motor |
|---|
| for(loop=0;loop<511;loop++){<br>GPIO_WriteBit(GPIOC, GPIO_Pin_6, Bit_RESET);<br>GPIO_WriteBit(GPIOC, GPIO_Pin_7, Bit_RESET);<br>GPIO_WriteBit(GPIOB, GPIO_Pin_0, Bit_RESET);<br>GPIO_WriteBit(GPIOB, GPIO_Pin_1, Bit_SET);<br>Delay(1);<br>GPIO_WriteBit(GPIOC, GPIO_Pin_6, Bit_RESET);<br>GPIO_WriteBit(GPIOC, GPIO_Pin_7, Bit_RESET);<br>GPIO_WriteBit(GPIOB, GPIO_Pin_0, Bit_SET);<br>GPIO_WriteBit(GPIOB, GPIO_Pin_1, Bit_SET);<br>Delay(1);<br>GPIO_WriteBit(GPIOC, GPIO_Pin_6, Bit_RESET);<br>GPIO_WriteBit(GPIOC, GPIO_Pin_7, Bit_RESET);<br>GPIO_WriteBit(GPIOB, GPIO_Pin_0, Bit_SET);<br>GPIO_WriteBit(GPIOB, GPIO_Pin_1, Bit_RESET);<br>Delay(1);<br>GPIO_WriteBit(GPIOC, GPIO_Pin_6, Bit_RESET);<br>GPIO_WriteBit(GPIOC, GPIO_Pin_7, Bit_SET);<br>GPIO_WriteBit(GPIOB, GPIO_Pin_0, Bit_SET);<br>GPIO_WriteBit(GPIOB, GPIO_Pin_1, Bit_RESET);<br>Delay(1);<br>GPIO_WriteBit(GPIOC, GPIO_Pin_6, Bit_RESET);<br>GPIO_WriteBit(GPIOC, GPIO_Pin_7, Bit_SET);<br>GPIO_WriteBit(GPIOB, GPIO_Pin_0, Bit_RESET);<br>GPIO_WriteBit(GPIOB, GPIO_Pin_1, Bit_RESET);<br>Delay(1);<br>GPIO_WriteBit(GPIOC, GPIO_Pin_6, Bit_SET);<br>GPIO_WriteBit(GPIOC, GPIO_Pin_7, Bit_SET);<br>GPIO_WriteBit(GPIOB, GPIO_Pin_0, Bit_RESET);<br>GPIO_WriteBit(GPIOB, GPIO_Pin_1, Bit_RESET);<br>Delay(1);<br>GPIO_WriteBit(GPIOC, GPIO_Pin_6, Bit_SET);<br>GPIO_WriteBit(GPIOC, GPIO_Pin_7, Bit_RESET);<br>GPIO_WriteBit(GPIOB, GPIO_Pin_0, Bit_RESET);<br>GPIO_WriteBit(GPIOB, GPIO_Pin_1, Bit_RESET);<br>Delay(1);<br>GPIO_WriteBit(GPIOC, GPIO_Pin_6, Bit_SET);<br>GPIO_WriteBit(GPIOC, GPIO_Pin_7, Bit_RESET);<br>GPIO_WriteBit(GPIOB, GPIO_Pin_0, Bit_RESET);<br>GPIO_WriteBit(GPIOB, GPIO_Pin_1, Bit_SET);<br>Delay(1);<br>laststatus=0;<br>} |

### 3.3. Hardware Implementation

The combined PCB design for the curtain controller is depicted in Figure 11. The device consists two printed circuit boards, each with two layers (top and bottom). After the assembly of the components, the boards are mounted and packed into the final product.

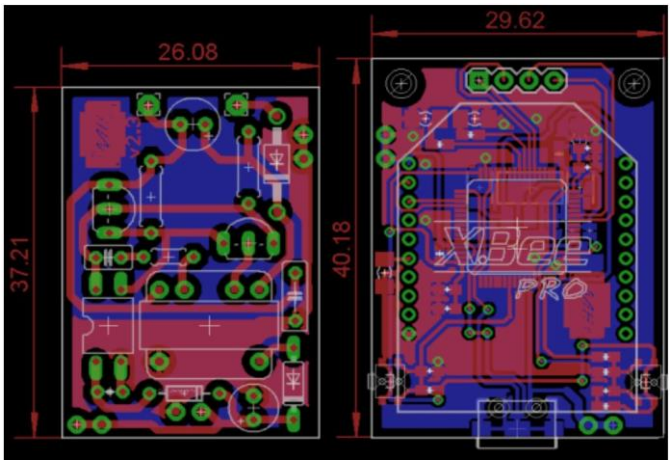
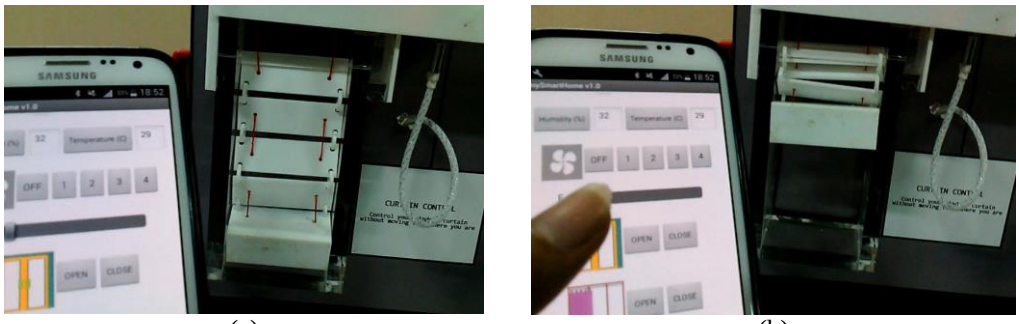



Figure 11. The PCB design for the curtain controller

### 3.4. Functional Test

The device performance is evaluated by using miniature curtain with vertical orientation to open and close the curtain Figure 12. The process that must be carried out is precisely same with the illustration in Figure 3. An initial setup is done by connecting the user's smartphone to the host via Bluetooth. Later, the user must give a specific command to control the curtain by opening or closing. The host will translate its request and it will send a command to the node. Based on the testing conducted, the controller works as intended to open and close the curtain as visualized in Figure 13.


Figure 12. Setup for functional test result of curtain device



(a) (b)

Figure 13. A photograph of the functional test for curtain control: (a) to close the curtain; (b) to open the curtain

Table 5. Power Measurement of Curtain Controller

| Input voltage | Current | |
|---|---|---|
| | *Idle condition* | *Process condition* |
| 5 $V_{DC}$ | 42.1 mA | 317.2 mA |

## 4. CONCLUSION

In this work, a prototype curtain controller for a smart home system is designed well. The device utilizes STM32L100RCT6 microcontroller as a processor, which controls the 28BYJ-48 stepper motor using IC ULN2003A as a driver. The control is done by utilizing PWM to control the motor's direction (CW and CCW), which is then applied to open or close the curtain. Based on the testing conducted, the device works as intended, with 210.5 mW for idle condition and 1,586 mW for process condition of the power consumption. This proves the device is suitable for low-power smart home system on WSN architecture. We also design the Android application as a mobile user interface.

In the future, the device will be upgraded so it can support partial opening/closing for the curtain (e.g. have the curtain only opened by one-third). In addition, the curtain operation will be integrated with other devices such as the lamp and fan to ensure stable room environment.

## ACKNOWLEDGEMENTS

This work was supported by the KEMRISTEK-DIKTI (Number 009/SP2H/LT/DRPM/IV/2017)

## REFERENCES

[1] Mendes TDP, Godina R, Rodrigues EMG, Matias JCO, Catalão JPS. Smart Home Communication Technologies and Applications: Wireless Protocol Assessment for Home Area Network Resources. Energies. 2015; 8(7): 7279-7311.
[2] Chennakesavan C. Implementation of Cloud Connected Smart Plug with Energy Monitoring System. Indonesian Journal of Electrical Engineering and Computer Science (IJEECS). 2017; 8(3): 702-704.
[3] Ryu CS, Hur CW. Monitoring System for Integrated Management of IoT-based Home Network. International Journal of Electrical and Computer Engineering (IJECE). 2016; 6(1): 375-380.
[4] Anonymous. Get to know Blind; Types of Elegant Curtain Cover with Various Variants (In Indonesia) [Internet]. January 4, 2016 [January 8, 2018]. Available from: http://www.ianinterior.co.id/mengenal-blind-jenis-tirai-penutup-nan-elegan-dengan-berbagai-varian/.
[5] Gunawan TS, Yaldi IRH, Kartiwi M, Mansor H. Performance Evaluation of Smart Home System using Internet of Things. International Journal of Electrical and Computer Engineering (IJECE). 2018; 8(1): 400-411.
[6] Warjono S, Suwinardi, Sugiharto E, Ansori MI, Andewi H V. Curtain Controller with Radio Waves (In Indonesia). Jurnal Teknik Elektro Terapan (JTET). 2013; 2(3): 136-142.
[7] Naeli M, Suleman. The prototype of Lamp and Curtains Controller Using Atmega8 Microcontroller Based on SMS Gateway (In Indonesia). Jurnal Bianglala Informatika. 2014; 2(2): 63-69.
[8] Shuvo RR. Smart curtain control system using smart phone via Bluetooth. B.Sc Thesis. Dhaka: Electrical and Telecommunication Engineering, Faculty of Science and Engineering, East West University; 2016.
[9] Ahmad AN, Dharmawan A. The prototype of Automation System Open Curtain Cover Based on Light Dependent Resistor (In Indonesia). International Journal of Electronics and Instrumentation System. 2011; 1(2): 21-34.
[10] Adiono T, Fathany MY, Putra RVW, Afifah K, Santriaji MH, Lawu BL, Fuada S. Live Demonstration: MINDS-Meshed and Internet Networked Devices System for Smart Home. Proceedings of the 13th IEEE Asia Pacific Conference on Circuits and Systems (APCCAS). Jeju. 2016: 736-737. DOI: 10.1109/APCCAS.2016.7804031.

[11] Adiono T, Putra RVW, Fathany MY, Lawu BL, Afifah K, Santriaji MH, Fuada S. Rapid prototyping methodology of lightweight electronic drivers for smart home appliances. International Journal of Electrical and Computer Engineering (IJECE). 2016; 6(5): 2114-2124.
[12] Adiono T, Putra RVW, Fathany MY, Lawu BL, Afifah K, Santriaji MH, Fuada S. Prototyping design of electronic end-devices for smart home applications. Proceedings of the IEEE Region 10 Symposium (TENSYMP). 2016: 1-5. DOI: 10.1109/ TENCONSpring.2016.7519415.
[13] Fathany MY. Design and Implementation of Smart Home System Infrastructure (In Indonesia). M.T. Thesis. Bandung: Department of Electrical Engineering, School of Electrical Engineering and Informatics, Institut Teknologi Bandung; 2016.
[14] Fathany MY, Adiono T. Wireless protocol design for smart home on mesh wireless sensor network. Proceedings of International Symposium of Intelligent Signal Processing and Communication Systems (ISPACS). 2015: 462-467.
[15] Adiono T, Fathany MY, Fuada S, Purwanda IG, Anindya SF. A Portable Node of Humidity and Temperature Sensor for Indoor Environment Monitoring. Proceedings of the 3rd International Conference on Intelligent Green Building and Smart Grid (IGBSG), 2018: 1-5. DOI: 10.1109/IGBSG.2018.8393575.
[16] Adiono T, Fathany MY, Anindya SF, Fuada S, Purwanda IG. Wireless Control for RGB Lamp End-device: Design and Implementation. Proceedings of the IEEE Ten Regional Conference (TENCON), 2018.
[17] Adiono T. Fathany MY, Anindya SF, Fuada S, Purwanda IG. Using a smart plug based on consumer electronics to support low-power smart home. Unpublished.
[18] Lawu BL, Fathany MY, Afifah K, Santriaji MH, Putra RVW, Fuada S, Adiono T. Prototyping design of mechanical based end-devices for smart home applications. Proceedings of 2016 4th International Conference on Information and Communication Technology (IcoICT). 2016: 1-5. DOI: 10.1109/IcoICT.2016.7571927.
[19] Adiono T, Fathany MY, Anindya SF, Fuada S, Purwanda IG. Development of Wireless Fan's Speed Control using Smart Home for Smart Home Prototype. Unpublished.
[20] Adiono T, Tandiawan B, Fathany MY, Adijarto W, Fuada S. Prototyping Design of IR Remote Controller for Smart Home Applications. Proceedings of IEEE Region 10 Conference (TENCON). 2017: 1304-1308. DOI: 10.1109/TENCON.2017.8228059.
[21] Adiono T, Marthensa R, Muttaqin R, Fuada S, Harimurti S, Adijarto W. Design of database and secure communication protocols for Internet-of-Things-based smart home system. Proceedings of IEEE Region 10 Conference (TENCON). 2017: 1273-1278. DOI: 10.1109/TENCON.2017.8228053.
[22] Adiono T, Tandiawan B, Fuada S. Device Protocol Design for Security on Internet of Things based Smart Home. International Journal of Online Engineering. 2018; 14(7): 161-170.
[23] Afifah K, Fuada S, Putra RVW, Adiono T, Fathany MY. Design of Low Power Mobile Application for Smart Home. Proceedings of International Symposium on Electronics and Smart Devices (ISESD). 2016: 127-131. DOI: 10.1109/ISESD.2016.7886705.
[24] Adiono T, Anindya SF, Fuada S, Afifah K, Purwanda IG. Efficient Android Software Development using MIT App Inventor 2. Unpublished.