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

Article Info

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.

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].

Copyright © 2018 Institute of Advanced Engineering and Science. All rights reserved.

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

Journal homepage: http://journal.portalgaruda.org/index.php/EEI/index
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].

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

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](image)
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](image)

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 ‘middlesmen’ 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 UART 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.
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.

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.
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.

![Flowchart of curtain control](image)

Figure 7. Flowchart of curtain control

2.8. Android Application 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.

![GUI of 'close' and 'open' the curtain control](image)

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

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

Figure 10. Keil uVision 5 environment
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 millisecond. The accuracy of 28BYJ-47 stepper motor movement is 5.625°/64, meaning that in every step there is an 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

```c
// GPIO Init
GPIO_InitTypeDef GPIO_InitStructure; // Initialization Structure
RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOC, ENABLE); // Enable GPIOC clock
// GPIO Configuration
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6 | GPIO_Pin_7; // Initialization Structure
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
GPIO_Init(GPIOC, &GPIO_InitStructure);
RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOB, ENABLE); // Enable GPIOC clock
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 | GPIO_Pin_1;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
GPIO_Init(GPIOB, &GPIO_InitStructure);
```

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

```c
software design for the command of logic combination on the stepper motor
for(loop=0;loop<511;loop++)
{    // Initialization Structure
    GPIO_WriteBit(GPIOC, GPIO_Pin_6, Bit_RESET);
    GPIO_WriteBit(GPIOC, GPIO_Pin_7, Bit_RESET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_0, Bit_RESET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_1, Bit_SET);
    Delay(1);
    GPIO_WriteBit(GPIOC, GPIO_Pin_6, Bit_RESET);
    GPIO_WriteBit(GPIOC, GPIO_Pin_7, Bit_RESET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_0, Bit_SET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_1, Bit_SET);
    Delay(1);
    GPIO_WriteBit(GPIOC, GPIO_Pin_6, Bit_RESET);
    GPIO_WriteBit(GPIOC, GPIO_Pin_7, Bit_SET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_0, Bit_RESET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_1, Bit_RESET);
    Delay(1);
    GPIO_WriteBit(GPIOC, GPIO_Pin_6, Bit_RESET);
    GPIO_WriteBit(GPIOC, GPIO_Pin_7, Bit_RESET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_0, Bit_RESET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_1, Bit_RESET);
    Delay(1);
    GPIO_WriteBit(GPIOC, GPIO_Pin_6, Bit_SET);
    GPIO_WriteBit(GPIOC, GPIO_Pin_7, Bit_SET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_0, Bit_RESET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_1, Bit_RESET);
    Delay(1);
    GPIO_WriteBit(GPIOC, GPIO_Pin_6, Bit_SET);
    GPIO_WriteBit(GPIOC, GPIO_Pin_7, Bit_SET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_0, Bit_RESET);
    GPIO_WriteBit(GPIOB, GPIO_Pin_1, Bit_RESET);
    Delay(1);
    laststatus=0;
}
```
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](image)

3.4. Functional Test

The device performance is evaluated by using miniature curtain with vertical orientation to open and close the curtain in Figure 12. The process that must be carried out is precisely the 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 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](image)
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

<table>
<thead>
<tr>
<th>Input voltage</th>
<th>Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 V DC</td>
<td>42.1 mA</td>
</tr>
</tbody>
</table>

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


Curtain Control Systems Development on Mesh Wireless Network of the Smart Home (Trio Adiono)


