Ho Chi Minh City University of Technology
Faculty of Computer Science and Engineering

– CO2021 Logic Design Project – Spring 2019 –

Smart Home system based on ESP8266 NodeMCU and MQTT protocol

By
Ngo Duc Tuan - 1710364
Tran Dinh Tien -
Ngo Nguyen Thuan -
Nguyen Ngoc Anh Tuan -

Instructor: PhD. Cuong Pham Quoc

July 3, 2019

Acknowledgements

After a semester with the guidance of Dr. Cuong Pham Quoc - Department of Computer Science and Engineering, Ho Chi Minh City University of Technology, we have finally finished our product for the Logic Design Project course.

We would like to express our deep gratitude to our teachers who have accompanied during the semester. You encouraged and inspired us to implement and complete our project.

We need to say thank you to our colleagues. You gave us a lot of advices and help to complete our project.

We tried our best. But with our lack of knowledge and experiences, mistakes are inevitable. Your criticisms are much appreciate. Thank you.

Abstract

Smart home is an emerging trend that grows continuously in recent. It is composed of many essential technologies through connected, remotely control devices and network for improving the quality of human's life. Therefore, there is a lot of research and projects to apply this technology in real life. In this paper, we will illustrate a model of smart home based on remote control, observe the environment and automatically detect events. Our system is based on two main devices ESP8266 NodeMCU and Arduino Uno R3 to manage and control home in safe and convenient way.

Contents

List of Figures

1 Introduction

1.1 Product objectives

- Design a system that can simulate as a smart home.
- Simulate some fundamental functions: switch lamp, switch fan, open door.
- Control by three ways: MQTT, menu screen, button.
- Automatically detect abnormal condition and send alert to users.
- Allow to set member users and create a user hierarchy.

1.2 Report organization

After this chapter, the rest of this report is organized in the following manners:

- Chapter 2 describes the overall design of the proposed Smart Home system.
- Chapter 3 presents the design principles of the system.
- The implementation of the prototype is given in chapter 4.
- Chapter 5 presents the result.
- Finally, chapter 6 gives some concluding remarks and approaches for future works.

2 Overall Design

2.1 Platform

Our product is based on Arduino platform with ESP8266 NodeMCU and Arduino Uno R3 as the main microcontrollers. For the remote control, we use MQTT connectivity protocol for publish/subscribe messaging transport.

2.2 Overall schematic

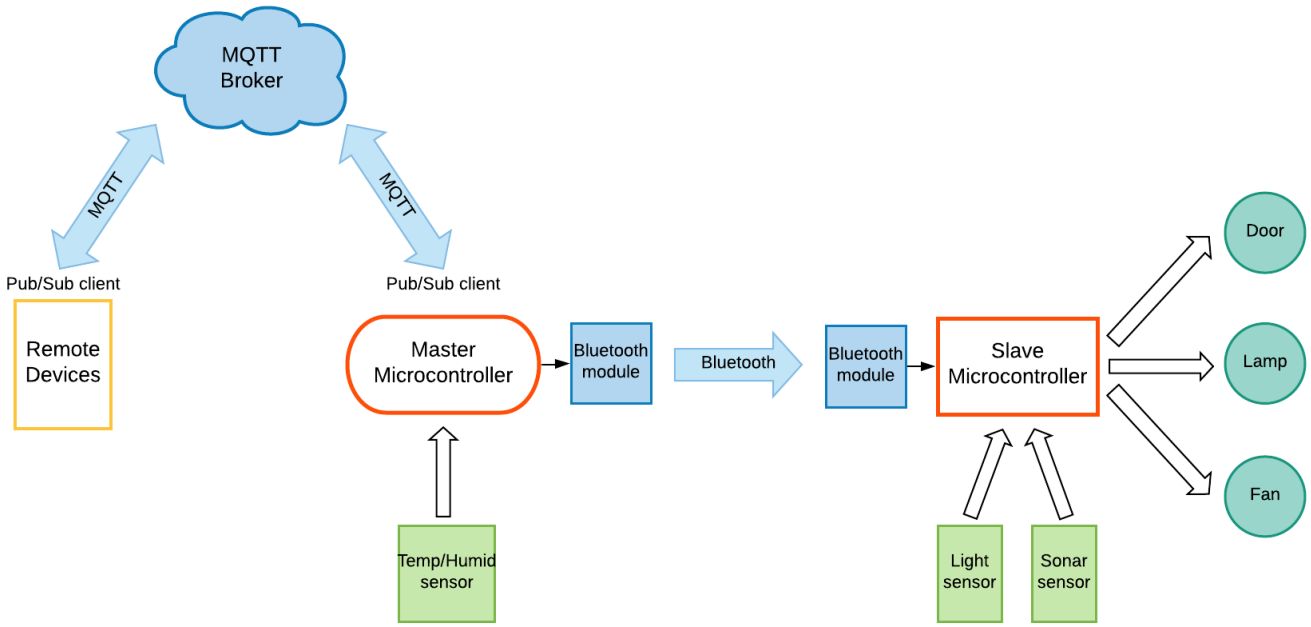

Figure 1: Overall schematic

2.3 Function

- Read temperature and humidity from sensor.

- Send information, sensor values to server.
- Log-in by RFID card or MQTT.
- Use LCD display to show information and control menu.
- Control devices in place by button or remotely by MQTT.
- Get time and date from server.
- Set alarm.
- Add/Remove member user.
- Use EEPROM so store information.
- Detect fire, intrusion and send inform to master user.

3 Design

3.1 User hierarchy

We provide two different methods to control the system: button control and MQTT control. These methods are divides into two kinds of user: master and member. Therefore, there are totally four kinds of users depends on the way they access and the log-in key.

Types of users			
No.	**Type**	**Way to access**	**Max. number**
1	Master RFID	master RFID card	1
2	Master MQTT	master MQTT client	1
3	Member RFID	member RFID card	4
4	Member MQTT	member MQTT client	4

They are ranked form the highest access priority to the lowest one with the same order in the table. This hierarchy can deal with the situation when there are more than one user try to access system at the same time. In this case, depends on the rank of users, the new user can be allowed to log-in and eject the current one or reverse.

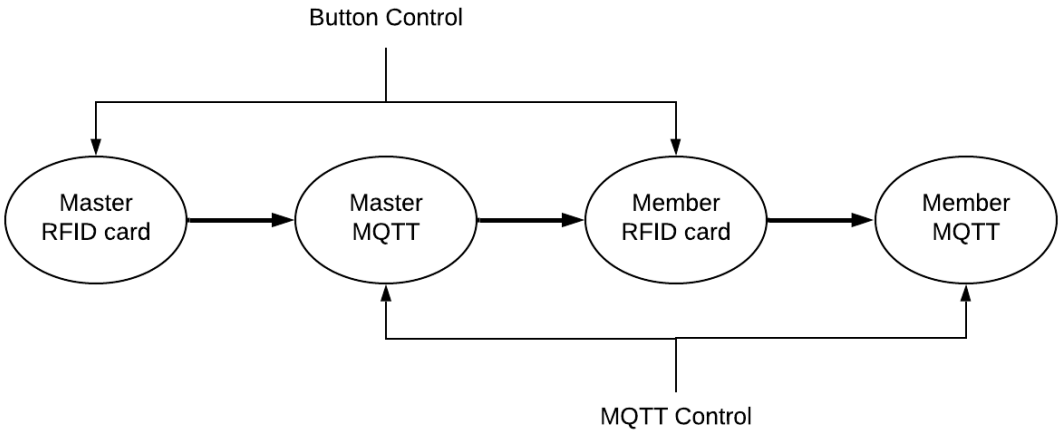

Figure 2: User hierarchy

3.2 Finite State Machine model

For controlling the main system, we design a finite state machine model with 5 main states:

- Log-in state

- Sleep Mode state
- Button Control state
- MQTT Control state
- User Manage state

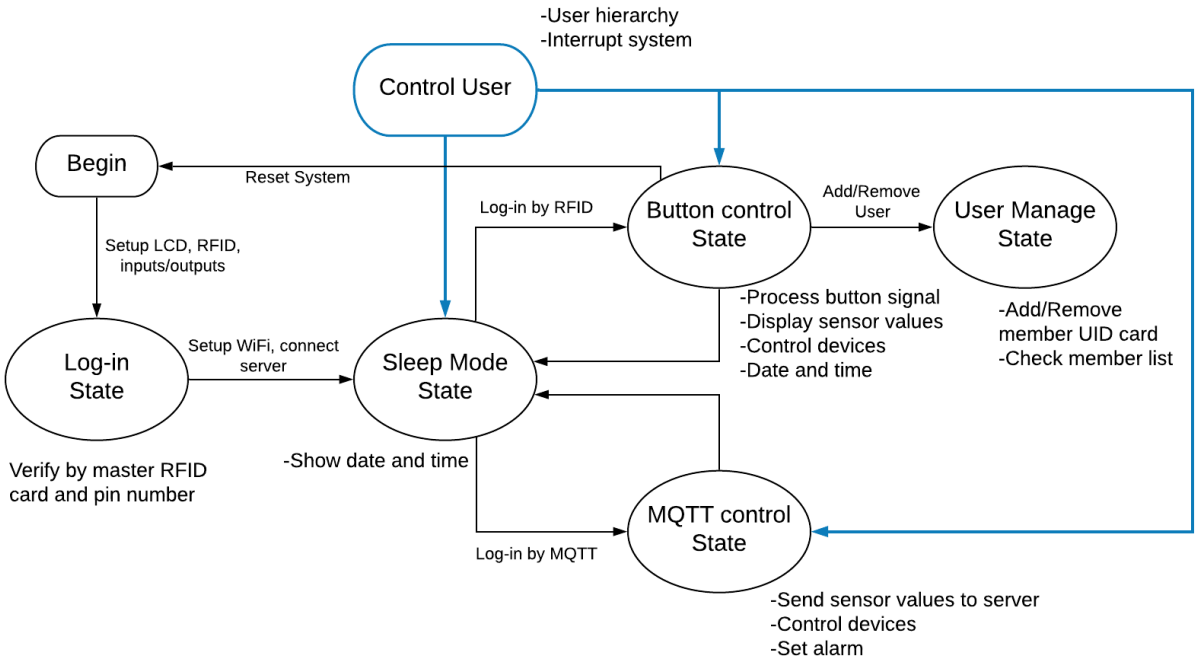

Figure 3: Finite State Machine model

1. Log-in State:
   This is the initial state when the system is power on or reset. It is used to setup the submodules and inputs/outputs. Because this is the first state, it allows only master RFID card as well as the pin number to access and change to the subsequents states. Therefore, only the master can activate the system.

2. Sleep Mode State:
   Whenever there is no interaction, the system changes to this state and it becomes a simple clock which displays time and date on the screen. If user wants to active the system, they can insert RFID card or send message through MQTT.

3. Button Control State:
   When user accesses by RFID card, the system changes to this state. Almost task are executed at this state: scroll menu, read sensor values, control devices, show time/alarm, turn off, reset system by process the signals received from two button:

Trig and Change. If the Change is pressed, the screen will be updated: scroll down or turn to the top menu. if the Trig is pressed, screen will be updated and the corresponding task will be executed simultaneously. There is a special task, when we want to add/remove member RFID card, the system will change to User Manage State and only when we access by RFID master card, it can turn to this state.

4. MQTT Control State:
   This state is used when user accesses through MQTT on smart phone or PC. Similar with the Button Control state, user can send message to do the following tasks: read sensor values, control devices, set/show alarm. The signal from two buttons has no effect in this state.

5. User Manage State:
   This state is used to check member RFID card list and add/remove member. The system provides four slot for member card and one for master card. Master card is initially set for the system and user cannot modify it. Whenever user adds or removes member card, the system also writes to EEPROM therefore the data will not be volatile during power off.

The Control User is a special mechanism. We design it to manage user access corresponding to the User Hierarchy. Whenever there are more than one user log-in to system at the same time, it will select among them then allow or deny the new access or eject the current user depends on their priority ranks.

3.3 Connect two microcontroller: Bluetooth Communication

We use Bluetooth communication to between two microcontroller. The master receives request from users, read values from environment through sensors then process and transfer signal to slave. On the other hand, the slave waits for signal from master, execute corresponding task to control peripheral devices.

To send signal from master to slave, first, we configure bluetooth module in master microcontronller as master mode and indicate it to connect to only our slave by providing address of the slave. The microcontroller interacts, transfers data and requests to bluetooth module then

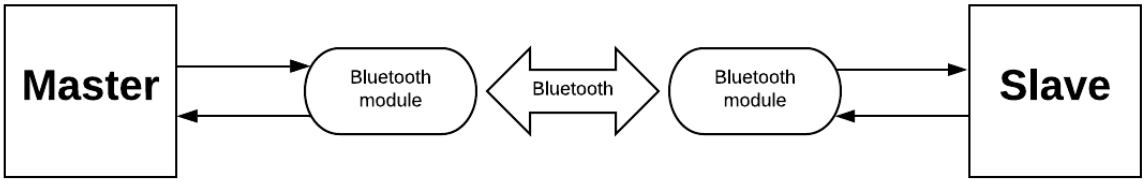

Figure 4: BLuetooth communication

3.4 Read temperature and humidity from sensor

We get temperature and humidity from environment from sensor. To avoid noise and flaw results, we take several nearest values then calculate its mean and use it as the final result.

In details, at the beginning, after setting up the sensor, it gets the values 20 times consecutively. After that, every 100ms, it will read new value as well as eject the oldest value from the array and recalculate the new mean value. During the process, if it encounters wrong value, such as the temperature is negative or abnormal high, it will bypass and remain the array. Therefore, the temperature and humidity is updated continuously and gradually to avoid noise value that can lead to wrong control.

3.5 Switch lamp and adjust the brightness according to the ambient light

There are three way to turn on the lamp:
- Using MQTT dash app on smartphone
- Using “turn on lamp” option on the menu which is displayed on the screen
- Using the button

The sensor with photoresistor can return two kind of value: analog and digital. It is connected to analog pin of Arduino R3 to get the analog value of light in surrounding environment. While LED is on, every one second, analog value will be got by using analogRead() and it is used to control the light intensity of the lamp.

3.6 Control door and fan

We use two motors to control door and fan.

For the door, its motor can rotate clockwise or counter-clockwise to open and close door. Every time it opens, it waits for 5 seconds. During that time, the infrared sensor checks if there is any obstacle or people between the door and then delay the door for 5 more seconds.

Control the fan's motor is more easier. When it receives the switch signal, it turns on or turns of the motor depends on the current state of this devices.

3.7 Fire Alert

Temperature value is used as the factor to check fire. If fire condition is true,the master will send a signal to slave by Bluetooth communication. Slave receives fire signal and execute the "Fire Detect" procedure. First, it will open the door, turn on all LED to alert danger. Then it will activate the buzzer until receive turn off signal when the fire condition is false .

3.8 Get date and time from NTP server

We use NTP server to update the date and time for the system.The name of variable that used for contacting with the NTP server is $timeClient$. To connect with the NTP server and update the time, $timeClient.begin()$ and $timeClient.update()$ is executed respectively in $voidsetup()$, after that, this process will run every 10 seconds. The time must be updated at least 1 time, the variable of time will continue to be counted base on $millis()$ variable.

Date and time are constantly updated by using two main fucntions , $timeClient.getFormattedDate()$ and $timeClient.getFormattedTime()$ (these functions will return a string included date and times, which is converted from the time in $timeClient$) . These strings will be saved to display on LCD (in the sleeping mode and in the "Time" option) and compare with the alarm time to active the alarm buzzer. This process will repeated every 500ms.

3.9 Set alarm

The users must use MQTT to set the alarm. the time must be inputted correctly for saving in the alarm memory.

Add new alarm time: 15:40ADD

Delete the existed alarm time: 15:40DEL

The alarm time will be saved or deleted in alarm memory if users inputs are correct syntax and "added" or "deleted" will be send to MQTT server. Otherwise, "wrong format" will be send to the server.

The alarm memory contain three main variables: "count" is the number of alarm times that are saved by users, "Time_Alarm" contain maximum 4 integers which are converted from the string of time to save into EEPROM, compare with the real time (a string of time: "16:45" is converted into 16×60 + 45 = 1005 to save to "Time_Alarm") and "Time_Alarm_display" consist of maximum 4 strings of time for displaying on LCD in "C.2.Alarm List" option and on $MQTTdash$ (The string "16:45" is saved in "Time_Alarm_display").

Every 1.5 second, the real time string (which is updated per 0.5 second) will be converted into integer for compairing with all elements in "Time_Alarm". if the real time is equal to any alarm times, ESP8266 will send a signal to Arduino Uno through Bluetooth to active the buzzer.

3.10 EEPROM

We use EEPROM to save the values when the board is turned off.The variable which is used to contact with EEPROM is $myEEPROM$. ESP8266 Nodemcu have an emulated EEPROM space of 512 bytes, therefore the system have 512 slots to save variables, each slot can save an integer from 0 to 255. All the Variables should be convert into integers to save to EEPROM. Two main functions are used to save and delete values in EEPROM are:

1. Save values (write to EEPROM):
   $myEEPROM.write(a,b)$ (a is the position (slot), b is the value).
   This function can be used to delete a particular slot by writing "0" into that slot.

2. Read values (read from EEPROM):
   $myEEPROM.read(a,b)$ (a is the position (slot), b is the value)

Each type of variables have the particular area in EEPROM for better management:

1. **Slot 0: (save the active state of the system):** when the system is actived, a value "1" will be saved in slot 0. Otherwise, the saved value is "0". This saved value are read from EEPROM in the $voidsetup()$ function (at the beginning of the operation), it help the system keep operate in the previous main state (actived or deactived) after users hard reset the system or run out of power.

2. **slot 10 - 130 (save the members' RFID cards):** The maximum number of members is 4, each member have 30 slots to save card's ID. Since the card's ID include both character and integer so it must be converted into integer to save into EEPROM. These values are read from EEPROM and return to original state (base on ascii table) in $voidsetup()$ and stored into an array of members' cards for checking UID.

3. **Slot 200 - 220 (save alarm times):** Each alarm times have 2 slots, the first slot is used to save hour, and the second store minute. And the "count" will be save in a particular position (220) . These alarm times are read and store in alarm memory in the beginning of program.

4. **Slot 300 - 402 (save login history):** each login histories is saved in 6 slots. five first slots are used to save time of login, the type of user is saved at the last position.

The EEPROM can be cleaned if the master choose "reset system" in option menu. The system will executed $myEEPROM.write(a,b)$ to write "0" to EEPROM from slot 0 to 299. This process will not overwrite Slot 300 - 512 (save login history).

3.11 Login history

The login history is saved in EEPROM, and this feature can only access by Master RFID. Master must push and hold both buttons in 5 seconds to access into this feature.

To save the login histories, the program need a special data structure - Circular queue. This circular queue store 10 arrays of 6 integers, the maximum number of login histories is 10. When the queue is full, The oldest data will be replaced by the new data. Each of 6 integers in the array is used for different purposes:

1. The first 3 integers: are used to save the date of the login event.The data include year (2 last digit of year, for example: 2019 ⟶ "19" will be saved), month, day respectively.

2. Next 2 integers: are used to save the time of login event. The data include hour and minute.

3. The last integer: is the type of users, There are 7 different integers:

The integer	Type of user
0	Master RFID
1	Master MQTT
2	Member RFID 001
3	Member RFID 002
4	Member RFID 003
5	Member RFID 004
6	Member MQTT

The login histories cannot be deleted by users (even Master RFID). This feature help master to check unauthorized accesses.

3.12 MQTT dash: control the system by using a smartphone

This software help the system connect with smartphones through MQTT server. It can be downloaded easily on CH play (only support for android).

To connect the software with MQTT server, users must input correct "server", "User name", "User password" and "port" from MQTT server into the software. After connect successfully, we can add some objects to the interface of MQTT dash, these boxes can be used to input and show the information received from the system (Display temperature, humidity, the status of the system and input the alarm time) or used as a button to send signals to the system (open door, turn on lamp, ...).

	Name of object	**Type of object**	**Function**	**Content**
1	Active	Button	Send	Password of user
2	Temp|Humid	Text box	Receive	Temperature and humidity
3	Set Alarm	Text box	Send/receive	Alarm time/result set time
4	Display Alarm	Text box	Receive	Alarm time
5	State	Text box	Receive	State of the system
6	Open Door	Button	Send	Open door signal
7	Switch Lamp	Button	Send	Switch lamp signal
8	Send Info	Button	Send	Read sensor signal

The "Active" button will send a special password (different with members - who have the same password). After receive master's password, the system will change to "Master MQTT" state (The system will only receive special commands from Master's smartphone).

4 Implementation

In this project, we use two different microcontrollers: ESP8266 NodeMCU and Arduino Uno R3. Then we separate the system into two parts corresponding to these two microcontrollers.

4.1 Components

4.1.1 Hardware components

Components List					
No.	**Name**	**Number**	**No.**	**Name**	**Number**
1	ESP8266 NodeMCU	1	7	L298N	1
2	Arduino Uno R3	1	8	DHT11	1
3	RFID RC522	1	9	CDS sensor	1
4	LCD_I2C	1	10	Infrared sensor	1
5	LM2596	2	11	Motor	2
6	HC-05	1	12	Button	2
7	HC-06	1	13	Super bright LED	6
8	Buzzer	1	16	Capacitor	10

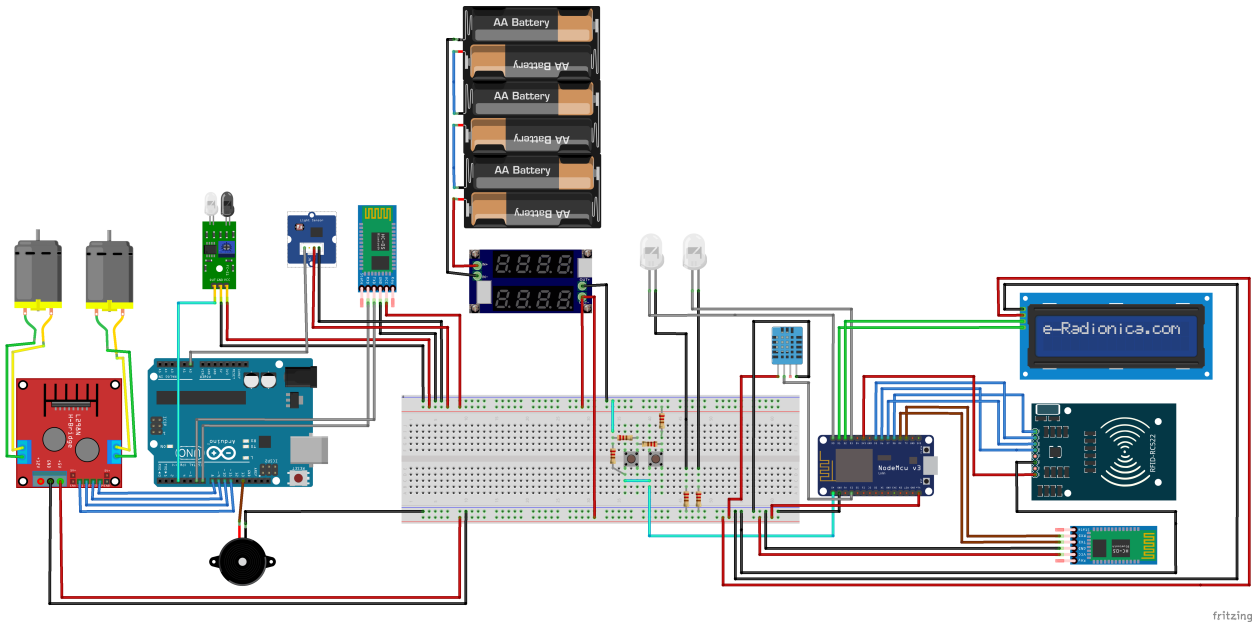

Figure 5: Connect hardware components by Fritzing

4.1.2 Software components

MQTT (Message Queuing Telemetry Transport) is an ISO standard publish-subscribe-based messaging protocol. It is designed for connections with remote locations where the network bandwidth is limited.

MQTT is the machine-to-machine protocol of the future. It is ideal for the “Internet of Things” world of connected devices. Its minimal design makes it perfect for built-in systems, mobile phones and other memory and bandwidth sensitive applications.

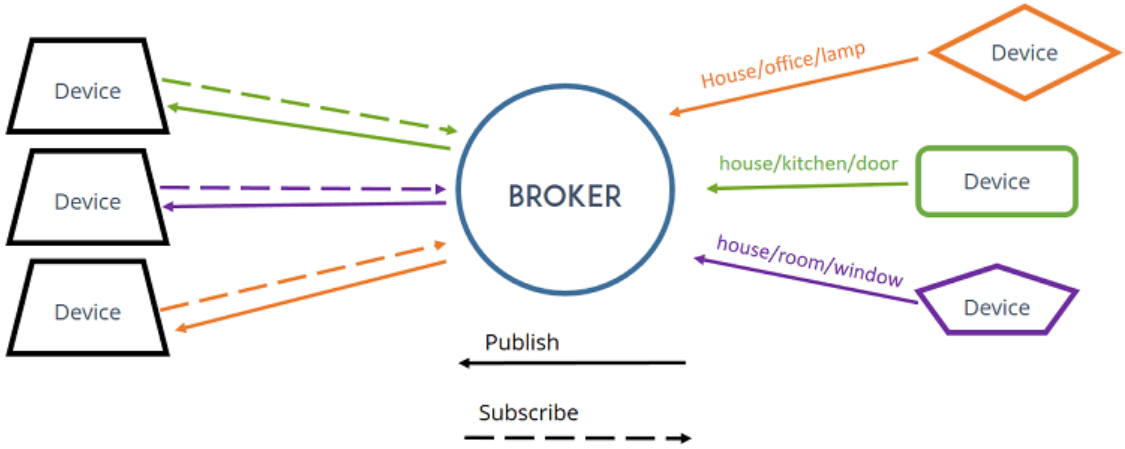

Figure 6: MQTT protocol

An MQTT system consists of clients communicating with a server, often called a "broker". A client may be either a publisher of information or a subscriber. Each client can connect to the broker. Information is organized in a hierarchy of topics. When a publisher has a new item of data to distribute, it sends a control message with the data to the connected broker. The broker then distributes the information to any clients that have subscribed to that topic. The publisher does not need to have any data on the number or locations of subscribers, and subscribers in turn do not have to be configured with any data about the publishers.

- Main Control System:
  We use module ESP8266 NodeMCU as the main microcontroller. It controls the main state of the system, process input signal from users through buttons or MQTT then respond proper information, send request to the other microcontroller to control devices.

- Devices Control System:
  Module Arduino Uno R3 is used to control peripheral devices. It receives request from the main control system and do the corresponding tasks.

To connect two parts together, we use Bluetooth communication with the master - slave model. The main control system uses module bluetooth HC-05 as the master and

the other uses HC-06 as the slave. Master transfers signal and slave waits until receive then execute task.

4.2 Main Control System

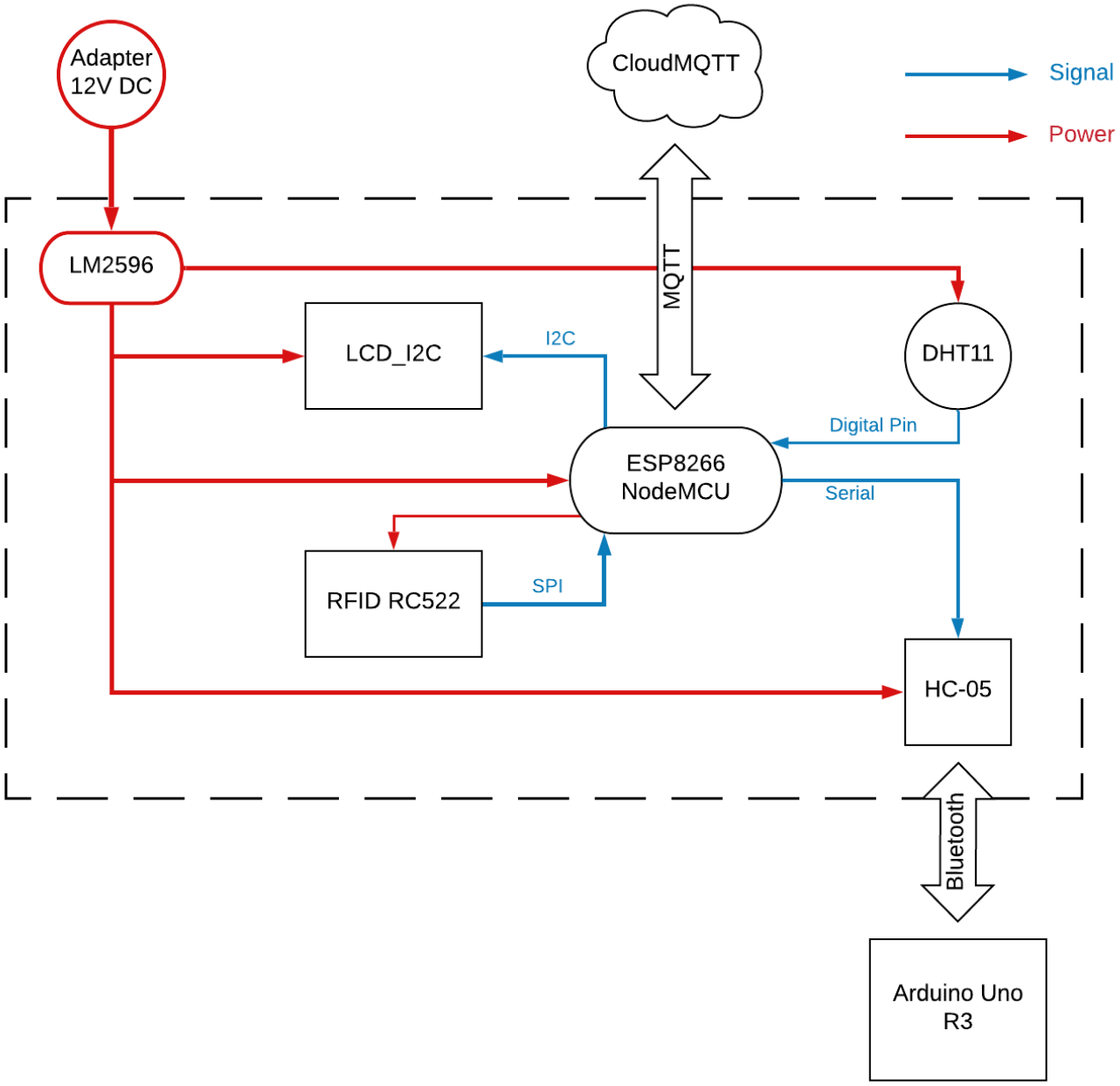

Figure 7: Main Control System

4.2.1 ESP8266 NodeMCU

NodeMCU is an open source IoT platform. It includes firmware which runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module.

We used this module as the main microcontroller in this system to connect all other submodules. Especially, it can be used to search and connect to WiFi then access to our MQTT API to transfer data and signal.

It receives requests from users through MQTT or button to process and send command and data to submodules. Reversely, it also receives data from sensors, analyze and send to the server.

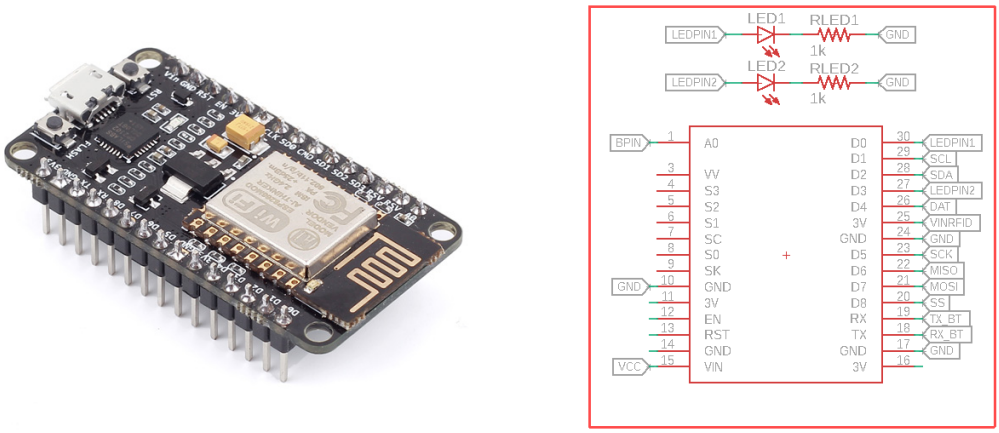

Figure 8: ESP8266 NodeMCU

4.2.2 Power Supply

Since the system contains a large number of modules, we use a 12V DC from adapter and module LM2596 to convert 12V DC to around 4.7V DC with high current then supply all module in the system.

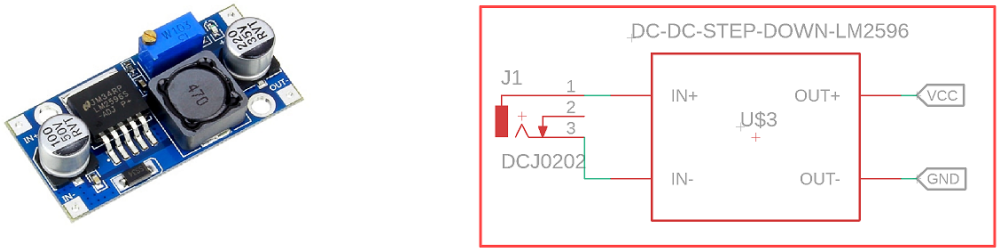

Figure 9: Module LM2596

4.2.3 LCD Display

In this system, to display value and menu, I chose the LCD display with the I2C converter to reduce the number of connecting pins. The ESP8266 Nodemcu interacts with LCD through I2C connection where the Nodemcu is master and LCD is slave.

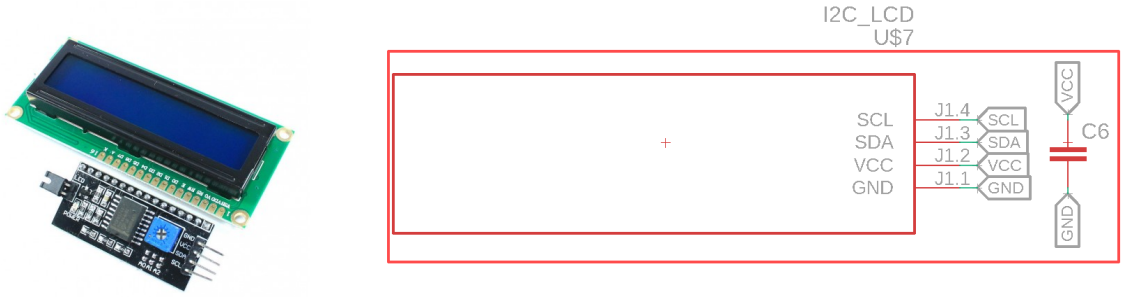

Figure 10: Module LCD

4.2.4 Button

To control the system in Button Control state, including choose method to login, switch menu, select tasks, ..., we use two buttons for two different functions. The first is for move the cursor (to scroll the menu, move up, move down) and the second is for select tasks. Two buttons are connected to analog Pin A0. By choosing different values of resistor, when we press, each button has different voltage level corresponding to analog value the microcontroller read from A0. Then it can detect which button is pressed and do the proper function.

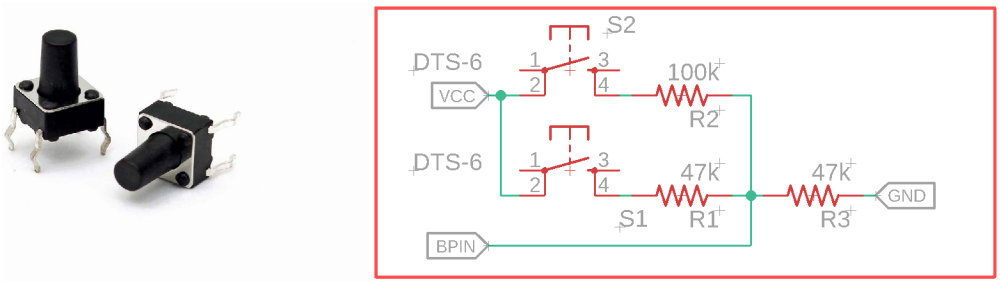

Figure 11: Push buttons

4.2.5 Sensor DHT11

With the support of DHT11 library, we can read the value of temperature and humidity directly through a digital Pin.

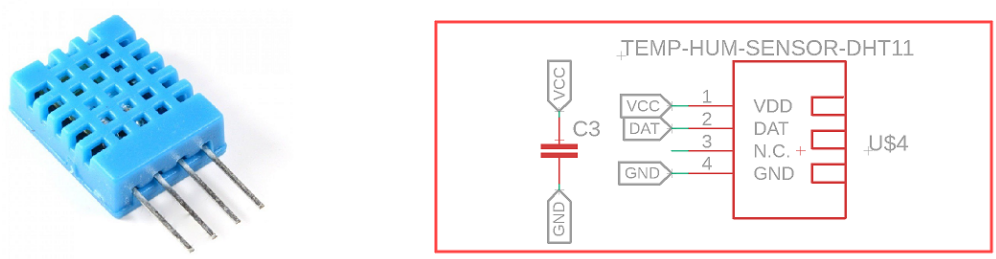

Figure 12: Module DHT11

4.2.6 RFID RC552

Module RFID interacts with Nodemcu through SPI. It is used to check the present of UID card, read or overwrite UID code. In this system, we use it to check master card and member card to login/logout system as well as add/remove member.

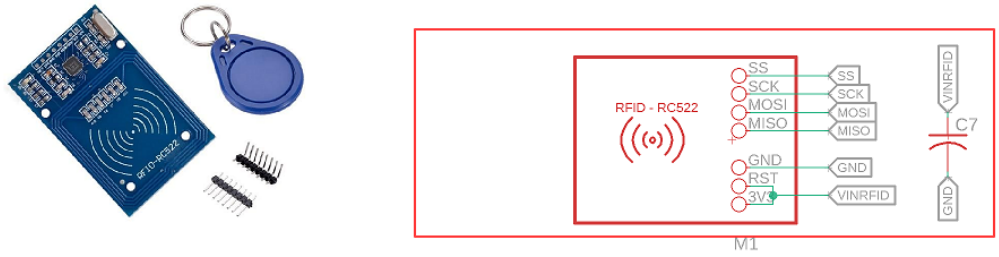

Figure 13: Module RFID RC552

4.2.7 HC-05

Module bluetooth HC-05 interacts with the NodeMCU by Serial communication. It is used to transfer data and request wirelessly between microcontrollers in a small area. We can use the built-in TX, RX pins or create a software Serial port to connect with this module.

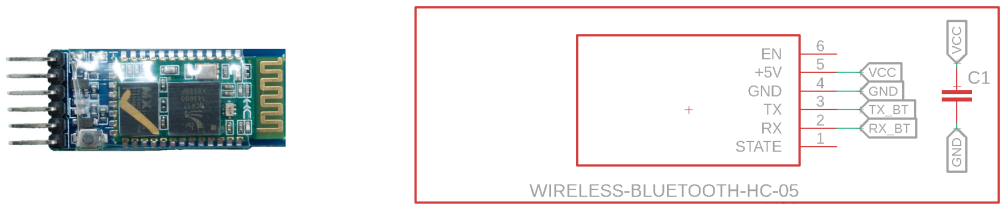

Figure 14: Module HC-05

4.3 Devices Control System

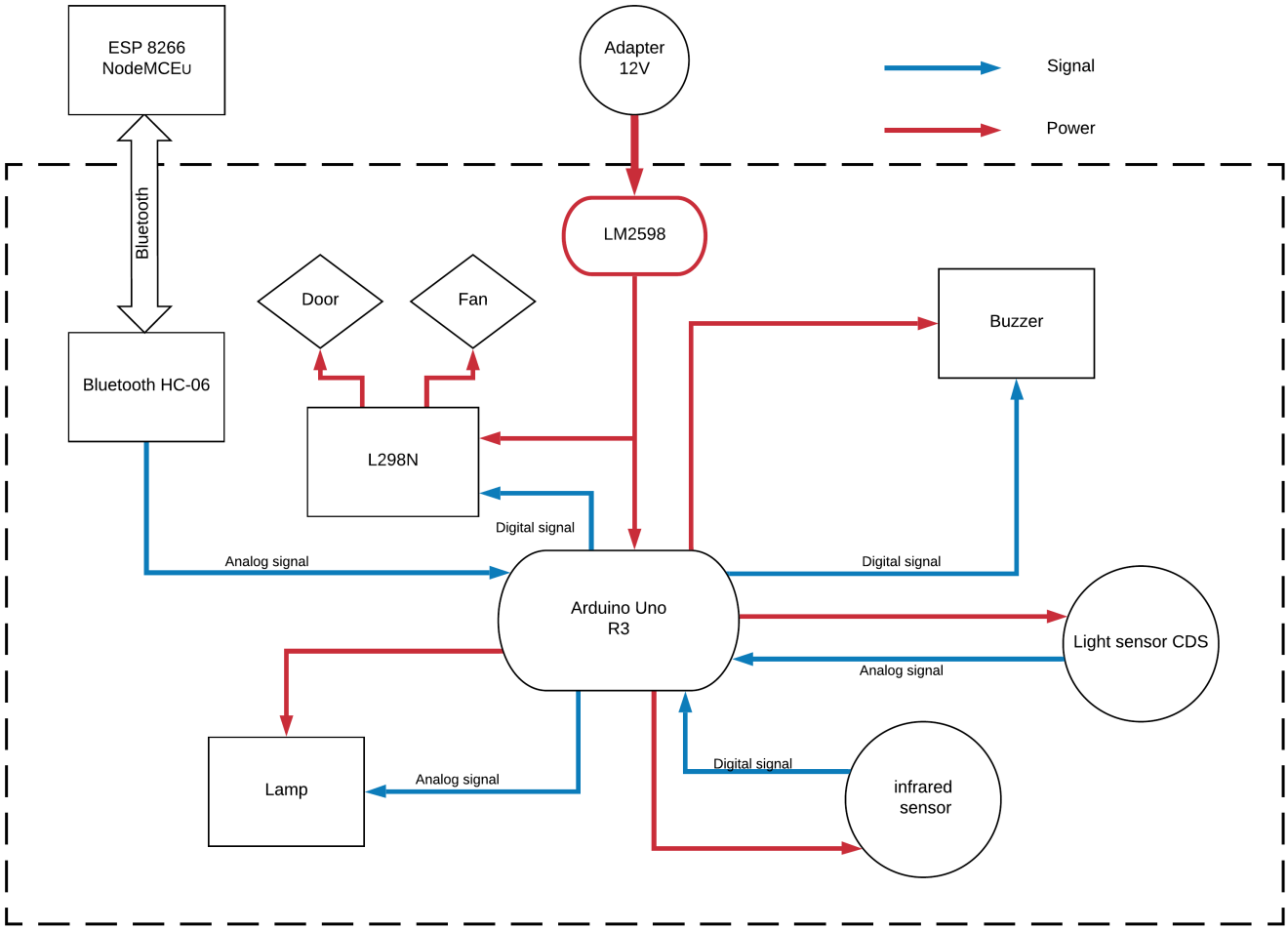

Figure 15: Devices Control System

4.3.1 Arduino R3

Arduino R3 is a microcontroller board and it is used as main controller which is connected to Stepper driver module L298N, LED, module Bluetooth, module buzzer YL-44 and orther sensor such as Temperature and Humidity sensor (DHT 11), Light Detector CDS, infrared sensor LM393. In this design, we use 17 pin, include pin 5V to supply power for all sensor and device.

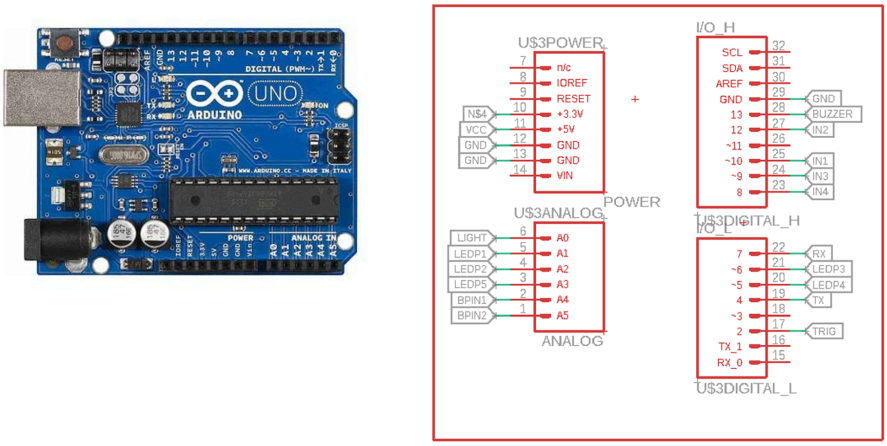

Figure 16: Arduino Uno R3

4.3.2 Infrared sensor LM393

LM393 is a very popular infrared sensor, the sensor has the ability to identify obstacles in the environment with a pair of infrared transceivers to transmit and receive infrared data. Infrared rays emitted with a certain frequency, when there is an obstacle on the transmission line of the LED, it will reflect on the infrared receiver LED, then the obstacle LED on the module will light, when there are no obstacles, LED will turn off. It is easy to get data via 1-wire communication (1-wire digital communication transfers data only). So it's data pin is connected to digital pin 3 of Arduino board and labeled as TRIG.

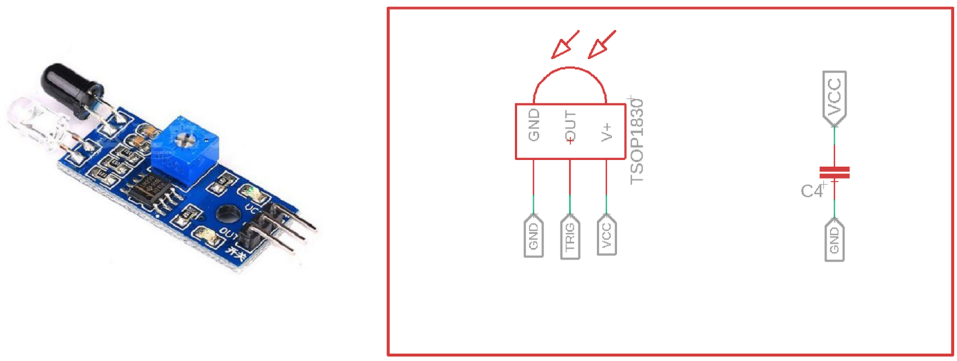

Figure 17: Infrared sensor

4.3.3 Light Detector CDS

Light Detector CDS using photoresistor is a digital sensor - the output signal is the value of Digital HIGH (5V) and LOW. At OUT terminal, the circuit returns HIGH (5V) when it is dark (low light intensity) and LOW if it is the opposite. CDS has 3 pin, VCC is connected

to pin 5V of Arduino R3 board to get power supply, data pin is connected to pin A1 of Arduino R3 board and it is labeled as “LIGHT’.

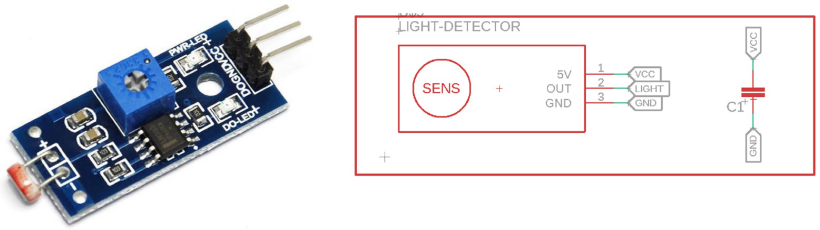

Figure 18: Light Detector CDS

4.3.4 Module buzzer YL-44

Module buzzer YL-44 has 3 pin , VCC pin is connected pin 5v to get power supply from Arduino R3 , the enable pin of buzzer is connected to the digital pin 13 of Arduino R3 and it is labeled as “BUZZER”.

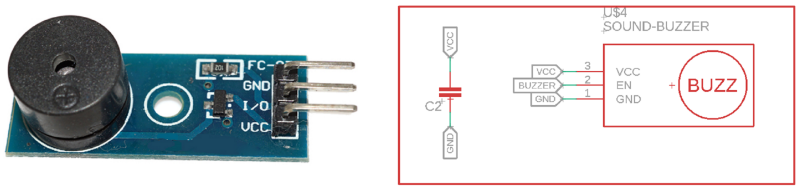

Figure 19: Module buzzer YL-44

4.3.5 Bluetooth HC-06

The HC06 Bluetooth module can only run 1 Slave mode (different from HC05 that can operate with Mater or Slave mode). This means you cannot actively connect from microcontroller to peripheral devices. The connection is: you must use a peripheral device (smartphone, laptop computer) to signal Buletooth connection that HC06 emits. After successful pairing you can send signals from the microcontroller to these peripherals, and vice versa. HC-06 communicates with the microcontroller with only 2 pins (Tx and Rx). In design, pin Tx of module Bluetooth HC-06 is connected to pin Tx on Arduino R3 and the connection of Rx pin is the same.

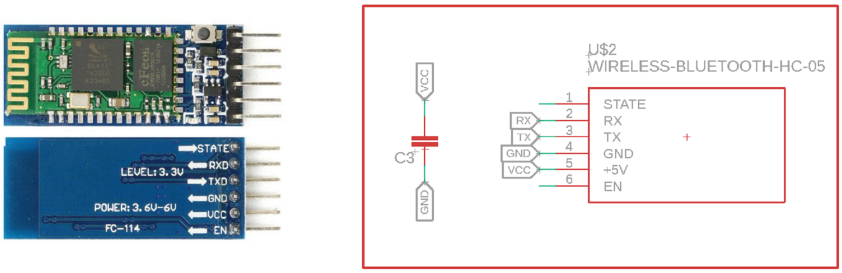

Figure 20: Module bluetooth HC-06

4.3.6 Stepper driver L298N

The L298 motor control circuit is capable of controlling 2 DC motors. The module includes 4 pins (2 pin for each motor) IN1, IN2, IN3, IN4 were connected to digital pin 11, 10, 9, 8 (PWM) of Arduino R3 which has ability to change speed of motors.

There are two levels of signal: HIGH and LOW in IN1, IN2, IN3, IN4. When IN1 is HIGH and IN2 is LOW, the motor of fan will rotate. When the signal in IN1 goes LOW, the motor stop working.

At the beginning, the signal of IN3 and IN4 are LOW, so the door is closed. When there is HIGH in IN3, the motor is activated and it is opening. After 5 seconds, the signal of IN3 go LOW while the signal of IN4 go HIGH, the motor rotates reversely to close the door and then both signals go LOW, the motor stops working.

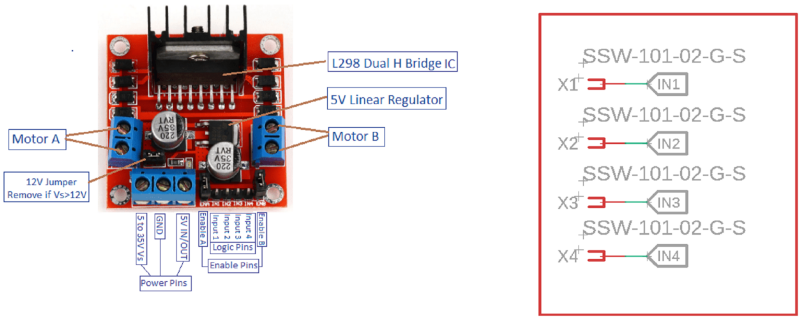

Figure 21: Stepper driver L298N

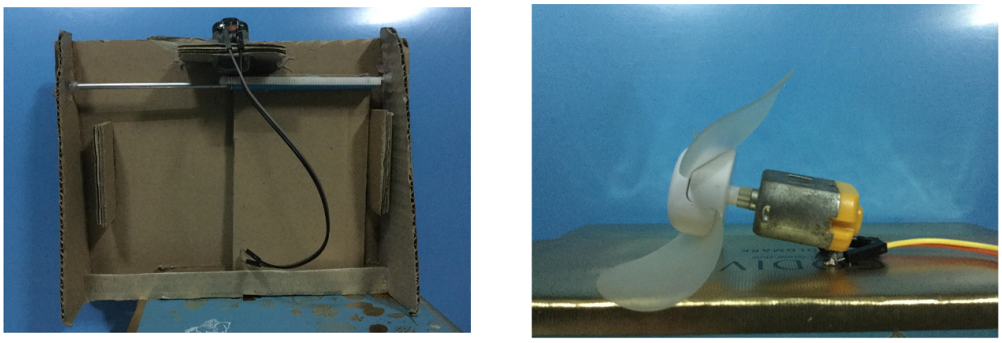

Figure 22: Door and fan motor

5 Results

5.1 Main Control System

5.1.1 Schematic

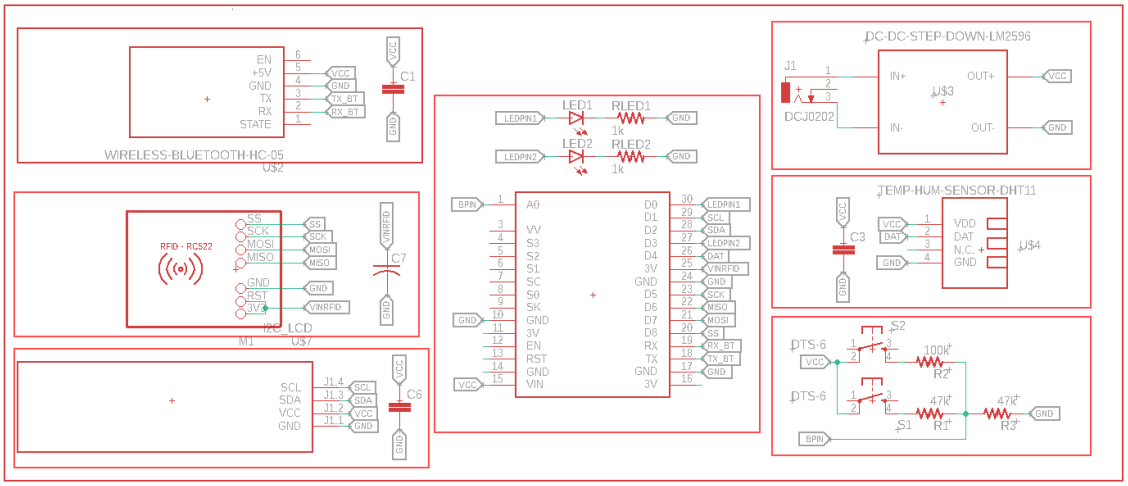

Figure 23: Schematic of Main control circuit

5.1.2 PCB layout

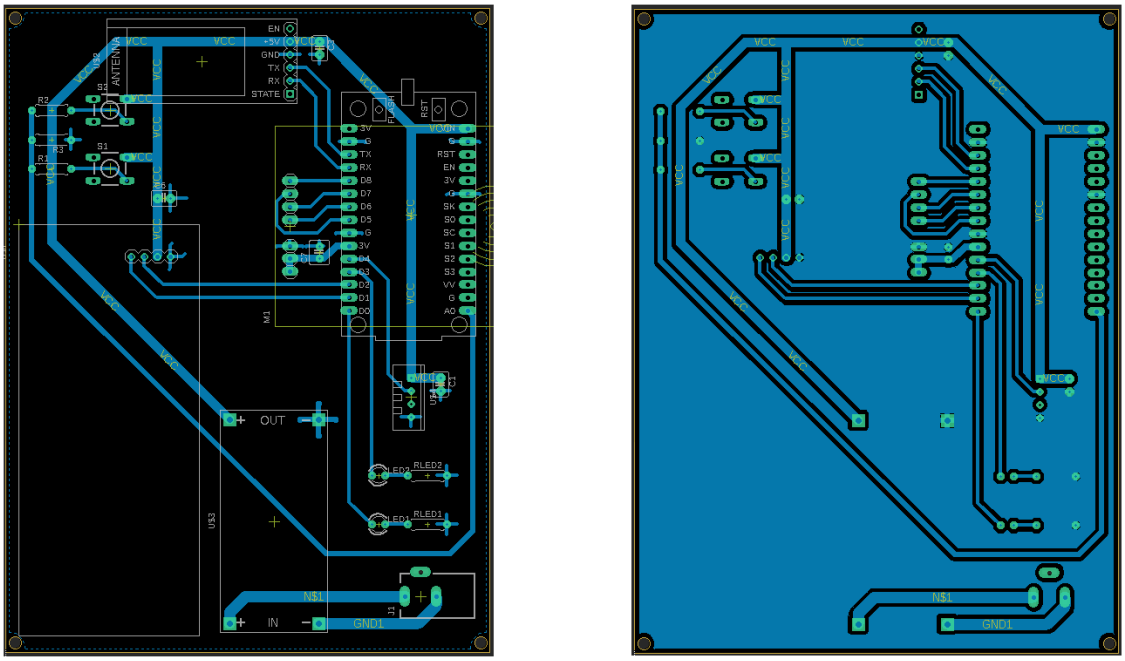

Figure 24: PCB layout of Main control circuit

5.1.3 Product

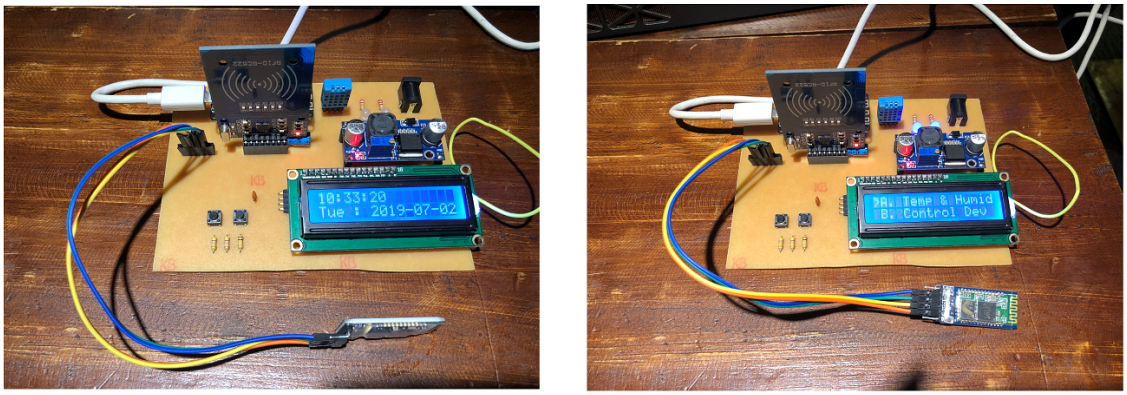

Figure 25: Main control circuit

5.2 Device Control System

5.2.1 Schematic

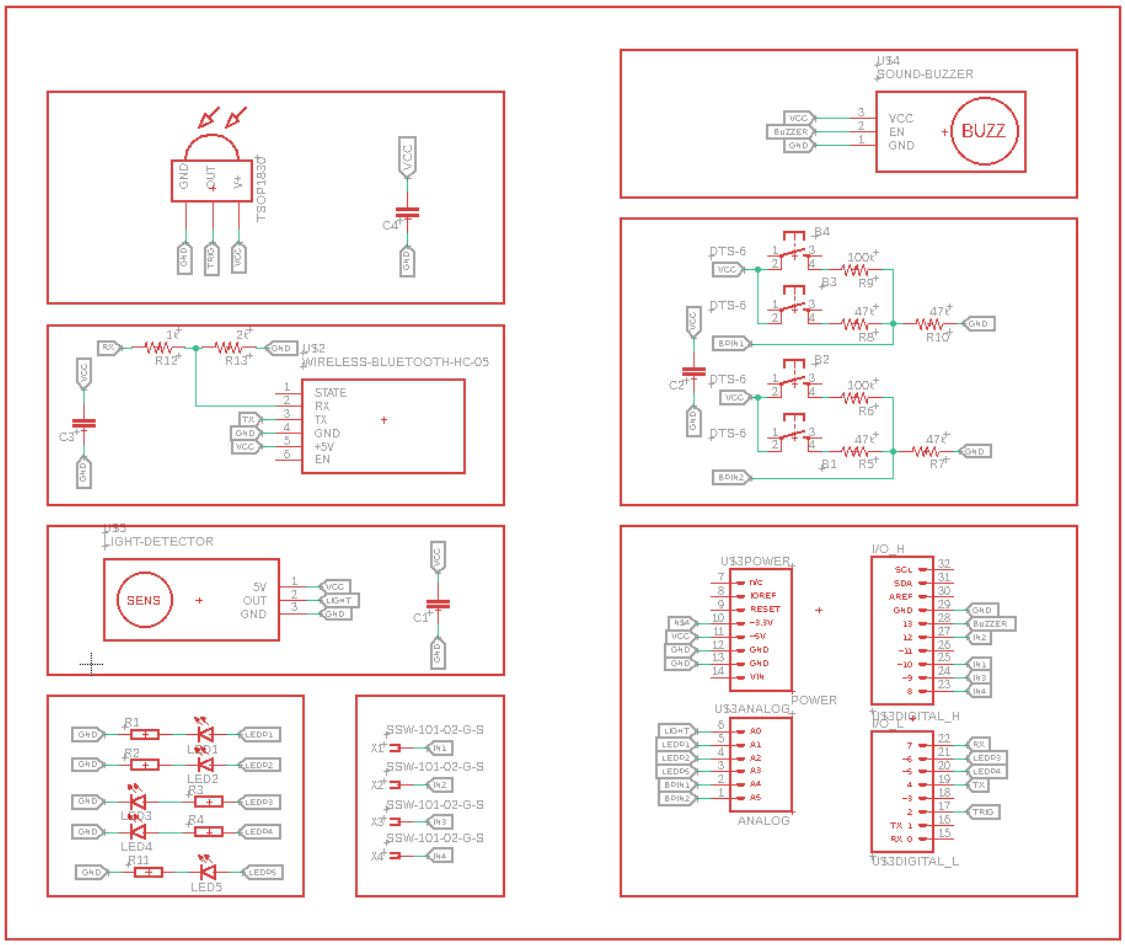

Figure 26: Schematic of Devices control circuit

5.2.2 PCB layout

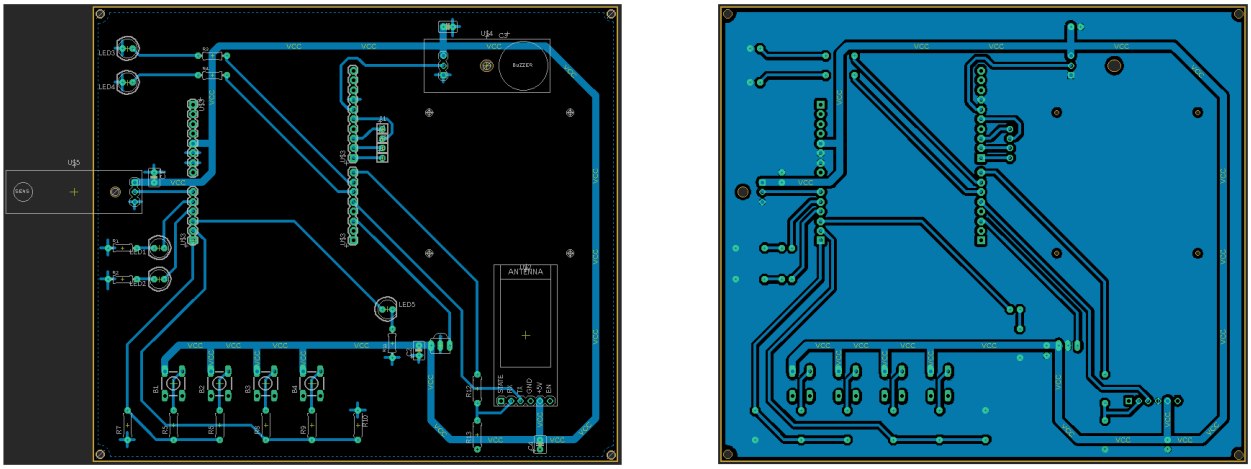

Figure 27: PCB layout of Devices control circuit

5.2.3 Product

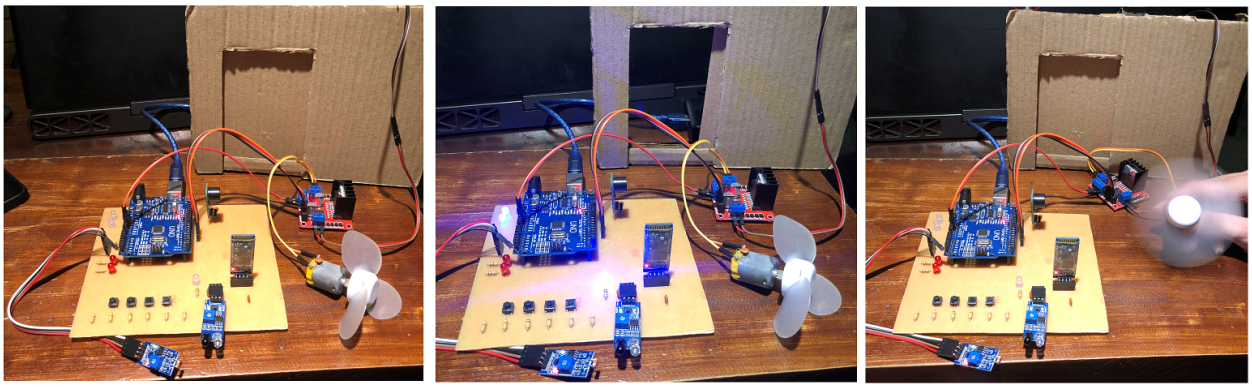

Figure 28: Devices control circuit

6 Instruction manual

1. When power on the system the first time, user needs to insert master RFID card only to activate it. After that, it will setup the connection to WiFi, to CloudMQTT then change to Sleep Mode. From now on, to access the system, user can use any valid method including: master/member RFID card, master/member MQTT.

2. When activating the system, it will change to proper state according to the way people use to access (MQTT Control Mode or Button Control Mode).

3. In MQTT Control Mode, user can use MQTT Dash , an application on Android OS, or directly access to to control and observe the system by sending message to subscribed topics.

4. In Button Control Mode, there are two buttons on the circuit: one for moving the menu, scroll down or turn to top; one for choosing the task to execute. The LCD display shows exactly what people currently do on the circuit.

5. Both control modes have following functions: display temperature and humidity, control devices (door, fan, lamp), show time and alarm.

6. Add/remove member RFID card or check member list is a special task of the system, only master RFID card can access this function. To add/remove card, user needs to insert card to module RC522 then the LCD display will show whether it is successful or failed following the reason. Every member card has an ID and user can check on the list to remove the card corresponding to the ID.

7 Discussion

7.1 Conclusion

7.1.1 Difficulties

- The first difficulty is finding ways to synchronize devices together. Two microcontrollers, Node MCU and Arduino R3, are connected wirelessly so there are some problem while two board communicating with each other.
- There exists the conflict between working state, control model and user manage.
- The model cannot work fluently if there is no Internet connection.

7.1.2 Achievements

- Can simulate almost smart house functions.
- Implement 3 ways to control devices:
  - Using button.
  - Remote control (Using MQTT app on smart phones or on MQTT cloud website).
  - Using menu option on LCD.
- Solved communication problem between two main microcontrollers by using Bluetooth connections instead of connecting via wire.
- Solved the conflict between working state and control model by designing a detail data and signal flow for each state.
- Solved conflict between user by designing a hierarchical model.
- Design and implement a PCB board to connect all devices instead of using breadboard.

7.2 Future works

1. We want to experiment more kind of sensors such as sound detector, pressure sensor, water quality sensor, chemical sensor, accelerometer sensor,....
2. When deploying a large number of sensors, we can create a wireless sensor network to observe and control in a larger area. In there sensors can share their data mutually and transfer to the main microcontroller automatically.

3. We can replay the ESP8266 NodeMCU microcontroller with the Raspberry PI as the main microprocessor to improve performance of the whole system. Besides that we also want to use more and more Arduino R3 board to control the hardware devices such as LCD and motors.

4. We want to create database and sever to collect information from sensor to analysis and calculation. Furthermore, sensors can access to get shared data or get information from sever to do their task. It can also be used as a storage to store backup data if a system crash happen.

References

1. “Comparison of esp8266 nodemcu development boards.” [Online]. Available: https://frightanic.com/iot/comparison-of-esp8266-nodemcu-development-boards/

2. “Module rfid rc522.” [Online]. Available: http://mualinhkien.vn/san-pham/317/module-rfid-rc522.html

3. “Board package for esp8266 nodemcu 12e v2.” [Online]. Available: https://github.com/esp8266/Arduino/releases/download/2.3.0/package_esp8266com

4. "API documentation" [Online]. Available: https://pubsubclient.knolleary.net/api.html

5. "ESP8266 Communication With Server and ESP8266" [Online]. Available: https://www.instructables.com/id/ESP8266-Communication-With-Server-and-ESP8266/

6. "Multiple button inputs using Arduino analog pin" [Online]. Available: https://rayshobby.net/wordpress/multiple-button-inputs-using-arduino-analog-pin/

7. "Connecting 2 Arduinos by Bluetooth using a HC-05 and a HC-06: Pair, Bind, and Link" [Online]. Available: http://www.martyncurrey.com/connecting-2-arduinos-by-bluetooth-using-a-hc-05-and-a-hc-06-pair-bind-and-link/