BACHELOR THESIS
Remote Light Control Demonstration Device
Bachelor Thesis Assignment

Gia Thinh Vo

Study Programme: B2649 Electrical Engineering
Study Branch: 2612R003 Applied Electronics
Title: Remote Light Control Demonstration Device
Demonstrační zařízení pro dálkové ovládání osvětlení

The thesis language: English

Description:
1. Analyse possibilities of remote light control.
2. Select and design a suitable system for implementing selected remote control modes.
3. Implement selected methods according to supervisor.
4. Practically verify the functionality of the designed and implemented methods.

References:

Extent and terms of a thesis are specified in directions for its elaboration that are opened to the public on the web sites of the faculty.

Supervisor: Ing. Libor Štěpánek, Ph.D.

Date of issue: 01.09.2018
Date of submission: 30.04.2019

doc. Ing. Petr Palacky, Ph.D.
Head of Department

prof. Ing. Pavel Brandštetter, CSc.
Dean
Author Declaration

I hereby declare that this bachelor’s thesis was written by myself, including annexes, independently under the supervision of Ing. Libor Štěpanec, Ph.D. I have mentioned all the quotes in the references I have drawn upon.

Ostrava, April 29, 2019
Date of submission

[Signature]
Student’s signature
Abstract

The main objective of the Bachelor Thesis is to design and implement a remote light control demonstration based on the Arduino IDE programming language and using a microcontroller ESP32 which combined dual-core processor used for Wi-Fi and Bluetooth. The main assignment of this method to control the light via the Internet network. Therefore, these devices are controlled for a long distance which it is not necessary to come to the physical switches to turn on or off devices. The first part of the bachelor thesis is a description of theory to analyze possibilities of remote light control, a brief description of the Arduino IDE, ESP32 chip, MQTT protocol and PWM method to control the brightness of lights. The second part is focused on practical work, to create the environment connection via the Internet, compiling and design PCB circuit board according to the given assignment. The main purpose is to make the demonstration of remote light control on-off using the Internet network. This aspect belongs to the Internet of Things (IoT) via the Internet any basic things can be connected and become smarter.

Key words
Arduino IDE, Light, IoT, Remote Control, Wi-Fi, Internet, ESP32 Control.
Acknowledgement

First and foremost, I would like to express my deep and sincere gratitude to my supervisor Ing. Libor Štěpanec, Ph.D., for his patient guidance. His office always comfy and welcoming to me. When the time was not supporting to me, with his assistance, valuable pieces of advice and encouragement steering me to help me recognize correct the direction and spent my concentrate to my bachelor work process. The simple words cannot express my thanks for him helping me get more professionalism and personality. Student life is represented as a heyday of youth, I do appreciate and would keep the working time with him in my memory.

I would like to express my gratefulness to Ing. Václav Sládeček, Ph.D., for the kindly help and guidance during my experiments and finding out hardware solutions of the light control device.

My grateful thanks are also extended to Ing. Jan Strossa, for his dedicated help in doing the PCB part, his useful support and advice to help me broaden my horizon and enrich my knowledge about how to drilling and soldering electronic components on PCB.

I would also like to extend my thanks to Phan Dao, Msc, Ph.D who is established the affiliate program between Ton Duc Thang University and VSB - Technical University of Ostrava, doc. Ing. Martin Kuchař, Ph.D., Ing. Aleš Havel, Ph.D. and all other lecturers in VSB - Technical University of Ostrava, for support and knowledge that I received during my study in Czech Republic.

I place on record, my sincere thank you to Mr. Phu Quoc Ma, for supporting effective advice to find the way how to write the thesis more completely. I would also like to thank Ms. Jana Bogdova, for her advice and assistance in keeping my progress on schedule.

Finally, I must express my deep special gratitude and appreciation to my parents, my siblings, my girlfriend and my intimate friend and some friends of mine, for the powerful support, care and encouragement me all the time throughout my years of study. This accomplishment would not be practicable without them, those persons I love from Czech Republic and Viet Nam.

Ostrava, April 29, 2019

Thinh Gia Vo
# Table Of Contents

Bachelor Thesis Assignment.......................................................................................................................... i  
Author Declaration........................................................................................................................................... ii  
Abstract ........................................................................................................................................................ iii  
Acknowledgement........................................................................................................................................ iv  
Table Of Contents .......................................................................................................................................... v  
List Of Figures ............................................................................................................................................ vii  
List Of Tables............................................................................................................................................. viii  
List Of Terminology Abbreviation .............................................................................................................. ix  
1. Introduction .......................................................................................................................................... 10  
2. Theoretical Part To Control Light Via The Internet........................................................................... 10  
   2.1. Wireless Communication ................................................................................................................. 10  
   2.2. Arduino IDE Introduction ................................................................................................................. 12  
       2.2.1. The Properties Of Arduino IDE ............................................................................................... 12  
       2.2.2. Arduino Programing Language ............................................................................................... 13  
       2.2.3. The Arduino Serial Monitor .................................................................................................. 13  
   2.3. ESP32 Introduction ......................................................................................................................... 15  
       2.3.1. ESP32 Board ............................................................................................................................ 15  
       2.3.2. Features Of ESP32 DEVKIT V1 ............................................................................................... 16  
       2.3.3. Specifications: ........................................................................................................................ 17  
       2.3.4. ESP32 DEVKIT V1 Pinout (ESP-WROOM-32 chip) .................................................................. 18  
       2.3.5. Absolute Maximum Ratings ................................................................................................... 18  
       2.3.6. Recommended Operating Conditions ...................................................................................... 19  
   2.4. Programming Environment For ESP32 ....................................................................................... 19  
       Programming Languages .................................................................................................................. 19  
   2.5. Brief Introduction Of MQTT Protocol .......................................................................................... 19  
       2.5.1. What Is The MQTT Protocol? ................................................................................................. 19  
       2.5.2. MQTT – Publish/Subscribe ................................................................................................... 19  
       2.5.3. MQTT – Broker ..................................................................................................................... 20  
   2.6. PWM Control Method .................................................................................................................... 21  
3. Practical Part To Control The Light .................................................................................................... 22
3.1. Simulation Of Connection Between Devices ................................................................. 22
3.2. Experimental LED Control Version ................................................................................ 23
  3.2.1. ESP32 Network Connection With The Local Wi-Fi Network .................................. 23
  3.2.2. ESP32: Data Publishing/Subscribing To MQTT Topic ............................................. 24
  3.2.3. Switching And Dimming The LED Via The MQTT-Broker ...................................... 27
3.3. Light Control Device ..................................................................................................... 30
  3.3.1. The Schematic Connection ....................................................................................... 31
  3.3.2. 3D Layout Version ................................................................................................. 33
  3.3.3. PCB Version ........................................................................................................... 34
4. Conclusion .......................................................................................................................... 35
5. References .......................................................................................................................... 36
Appendix One - Arduino IDE Programming Environment .................................................. 38
  Preparing Arduino IDE Environment for ESP32 ................................................................. 38
  Installing Programming Environment In Arduino IDE ....................................................... 38
  Installing The Windows USB Driver .................................................................................... 41
  Installing The ESP32 Board With Arduino IDE ................................................................. 42
Appendix Two – MQTT-Broker .............................................................................................. 45
  Creating MQTT – Broker .................................................................................................... 45
  Creating New Instance ........................................................................................................ 46
  Creating User Access And ACL Rules ............................................................................... 49
Appendix Three – Source Code Arduino IDE ....................................................................... 50
  Full Source Code: .............................................................................................................. 50
List Of Figures

Figure 1: IoT applications aspects [4]......................................................................................11
Figure 2: The interface of Arduino IDE Software ............................................................................13
Figure 3: The Arduino serial monitor window.................................................................................13
Figure 4: Example code and open the Arduino Serial Monitor.........................................................14
Figure 5: The Arduino Serial Monitor send information on the monitor window of the computer....14
Figure 6: ESP32 board combines Wi-Fi and Bluetooth wireless capabilities [8].................................15
Figure 7: Different types of ESP32 model board [8].........................................................................15
Figure 8: ESP32 DEVKIT V1 DOIT with ESP-WROOM 32 chip [7]......................................................16
Figure 9: The features of ESP32 DEVKIT V1 DOIT [8].....................................................................16
Figure 10: The buttons on-board of ESP32 DEVKIT V1 DOIT [8]......................................................17
Figure 11: Pinout of ESP-WROOM-32 with 30 pins version [7]............................................................18
Figure 12: Publish and Subscribe in the MQTT protocol system......................................................20
Figure 13: Devices or Clients connected via MQTT-Broker with Publish and Subscribe.................20
Figure 14: Duty Cycle 25%, 50% and 75% [11]..................................................................................21
Figure 15: Block diagram remote light control via MQTT protocol using the ESP32 as a client........22
Figure 16: Control LED using ESP32 via MQTT-Broker.................................................................23
Figure 17: Adding the library for Publishing and Subscribing to MQTT Topic via Arduino IDE library manager...................................................................................................................................25
Figure 18: Publishing and Subscribing to the topic of MQTT cloud.....................................................26
Figure 19: Websocket MQTT-Broker sends and receives the message..............................................27
Figure 20: Sending and Receiving a message by MQTT-Broker........................................................27
Figure 21: Switching information send back to Serial Monitor of Arduino IDE...............................28
Figure 22: PWM method to control the brightness of LED.................................................................28
Figure 23: Brightness control by sending a message from the broker .................................................29
Figure 24: 10% brightness of LED.....................................................................................................29
Figure 25: 75% brightness of LED.....................................................................................................29
Figure 26: 3D Top view layout of ESP32 control lights.....................................................................33
Figure 27: 3D layout of ESP32 control lights......................................................................................33
Figure 28: PCB layout version without the components.................................................................34
Figure 29: The final circuit board with the components.................................................................34
Figure 30: The main page for download the Arduino IDE.................................................................38
Figure 31: The main website to download GIT ...................................................................................39
Figure 32: Download GIT version for set up window.....................................................................39
Figure 33: The GIT GUI application interface..........................................................40
Figure 34: Source Location and Target Directory.............................................................................40
Figure 35: Git Bash window.............................................................................................................41
Figure 36: The computer manager with missing the CP2102 driver.................................................41
Figure 37: Added the URL of ESP32 to Arduino IDE........................................................................42
Figure 38: Loading ESP32 Board from Boards Manager .................................................................43
Figure 39: Install ESP32 by Espressif Systems ........................................................................43
Figure 40: Selection of the ESP32 Board to programming ..........................................................44
Figure 41: The interface free hosted message broker for the IoT ..............................................45
Figure 42: Cute Cat with a free broker for IoT solution. ...............................................................45
Figure 43: Fill in the email to create a new account cloudmqtt.com ........................................46
Figure 44: Select the name of the instance and add tags for separate instance between projects. ...46
Figure 45: Select a region and data centre for instances .................................................................47
Figure 46: Checking information about free MQTT-broker ...........................................................47
Figure 47: The interface of the new instance consist of information light control project ............48
Figure 48: The instance details information for MQTT-Broker. ....................................................48
Figure 49: Create user access for MQTT-Broker .........................................................................49
Figure 50: Create the ACLs rules, topic and pattern ..................................................................49

List Of Tables

Table 1: The specifications of ESP32 DEVKIT V1 DOIT ............................................................17
Table 2: Absolute Maximum Ratings [9] .....................................................................................18
Table 3: Recommended Operating Conditions [9] .................................................................19
Table 4: List Of Components ......................................................................................................32
## List Of Terminology Abbreviation

<table>
<thead>
<tr>
<th>Terminology Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</td>
<td>Access Point</td>
</tr>
<tr>
<td>ACL</td>
<td>Access Control List</td>
</tr>
<tr>
<td>ADC</td>
<td>Analog to Digital converter</td>
</tr>
<tr>
<td>AVR</td>
<td>Automatic Voltage Regulator</td>
</tr>
<tr>
<td>CAN</td>
<td>Controller Area Network</td>
</tr>
<tr>
<td>DAC</td>
<td>Digital to Analog converter</td>
</tr>
<tr>
<td>DEVKIT</td>
<td>Development Kit</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>IC</td>
<td>Integrated Circuit</td>
</tr>
<tr>
<td>I²C</td>
<td>Inter-Integrated Circuit</td>
</tr>
<tr>
<td>I²S</td>
<td>Integrated-Inter-IC Sound</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development</td>
</tr>
<tr>
<td>IDF</td>
<td>IoT Development Framework</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode</td>
</tr>
<tr>
<td>MCU</td>
<td>Multi Control Unit</td>
</tr>
<tr>
<td>MQTT</td>
<td>Message Queuing Telemetry</td>
</tr>
<tr>
<td>M2M</td>
<td>Machine to Machine</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed Circuit Boards</td>
</tr>
<tr>
<td>PIC</td>
<td>Programmable Intelligent Computer</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
</tr>
<tr>
<td>RMII</td>
<td>Reduces Media-Independent Interface</td>
</tr>
<tr>
<td>SPI</td>
<td>Serial Peripheral Interface</td>
</tr>
<tr>
<td>STA</td>
<td>Station</td>
</tr>
<tr>
<td>UART</td>
<td>Universal Asynchronous Receiver/Transmitter</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
</tbody>
</table>
1. Introduction

Stepping into the 21st century, along with the ongoing development of science and technology, followed by the rapid development of industries applied the new technologies to meet the market demand or human life. Access and control devices from several remote locations are one of the major reasons against waste energy. A good case in this point to explain why is when people leave home and remember that they forgot to turn off some devices like lights, television or the air conditioner they have to return to their house and switch them off, this issue exerts to energy loss and waste their time. Meanwhile, with a host of communication methods such as using Wi-Fi, Bluetooth connection, Radio-Frequency or using a sensor to connect with these devices, and they can be controlled, or another way the ongoing increase of the Internet, the human being can use a web server or an android application to remote control these devices. Analyse possibilities with these methods are mentioned above are wireless controllers and as know as the advantages of them which is not necessary to come to the physical switches to turn them on or off. Furthermore, The Bachelor Thesis has been writing about using the ESP32 dual-core processor which is not used only to remote light control or brightness of lamps but also it has great potential, it can be developed to establish as a smart-home system consist of whole applications that human being can use the remote control to administering, controlling devices before they come home such as check the room temperature and turn on the air-conditioner and send data to smart devices to the users, turn on the heater in the living room of their house, switching and controlling the speed of motor or electric fan.

2. Theoretical Part To Control Light Via The Internet

2.1. Wireless Communication

When it comes to Wi-Fi, Bluetooth or Radio-Frequency control devices have the same disadvantages. Using the normal Wi-Fi for the home which is not too far to connect to the router for the distances about several dozen meters to about one hundred meters is the limit maximum range [1]. In other way use Bluetooth to control devices, with this method Bluetooth have very short distance, maximum about 10 meters [2] to Bluetooth devices connect to the chip or nodemcu using for emission Bluetooth waves. Radio-Frequency (RF) remote controls are very common in wide range application and the most popular use in a remote of television with distance can transmit up to 30 meters [3]. For these methods with longer the limit range, they can be lost connection or interferences by the surrounding environment.

Comparison with three method Wi-Fi, Bluetooth and Radio-Frequency can be seen that Bluetooth has the shortest distance than another and Wi-Fi communicate distance farthest in three of them. But there is an issue need to take it into account whether can be control devices in over the world with any location in any country. Without limit range operation, this problem is solved by using the Internet network to connect with devices which you want to control from anywhere in the world.
In recent years a new concept in the technology industry that is mentioned most is "Internet of Things". The Internet of Things seems to be facing a turning point to go to the next stage for a modern, civilized world. Enterprises are tending to apply IoT technology products to produce more and more because of the potential creative market and lower production costs.

Two of the typical practical applications in life, the IoT are using in a wide range of daily life of human or enterprises such as:

**Smart-home:** smart-home is an aspect related to IoT which is the most searched on the Internet. Smart-home which a lot of automatic features such as turning on the air conditioner when you get home or turning off lights as soon as you leave the house, unlocking gate or door by using sound (human voice) or remote control and there are many features can help enhances the quality of life when using smart home.

**Wearable’s:** there are mega-trend topics in today’s date which the manufacturers are racing to create smart products to provide for consumption wearable of people. Devices are installed with many different types of sensors and software to received data and transfer data and inform the date are received for the user. Many example products are smartwatch, healthy watches or smart glasses. [4]

In all generally, from many benefits aspects of using IoT method to control devices via the Internet with long distance contain a lot of interference and low bandwidth communication between some devices (M2M). In this assignment, using the IoT solution to solve the problem control lights via the Internet for long distances connection.
2.2. Arduino IDE Introduction

AVR controllers and PIC microcontrollers are increasingly common and more complete but can say the appearance of Arduino in 2005, in Italy opened up a new direction for microcontrollers. The appearance of Arduino has supported people a lot in programming and design, especially for those who started to learn about microcontrollers without too much knowledge, deep understanding of physics and electronics. According to the main website of Arduino. Arduino is using the electronics platform as an open-source base on easy to use hardware and software. Arduino is an IDE with the built-in editor, compiler, programmer and it comes with firmware with the boot loader, built-in library kits and easy integration. The language is used C / C++. All are open source and contributed, developed daily by the community. About Arduino have many parts as Arduino Boards, Arduino programming language (based on Wiring) and Arduino Software IDE (based on Processing). Arduino Boards used to read inputs-light on sensors, finger buttons, messages so on and transfer them to an output to operate the motor, turn-on LED (Lamps) or even publishing and received data online. [5]

2.2.1. The Properties Of Arduino IDE

- Arduino hides the complexity of electronics with simple concepts, obscuring the complexity of the software with brief procedures. Setting up the output for 1 MCU by setting the register is so complicated that the professional must also open the data sheet to watch, but for Arduino, just call a function.
- Because of its popularity and ease of use, with built-in libraries. You only need to pay attention to the product features but ignore the sub-sections (protocol, data sheet), so you can help newbie not easily access and make great products without having to know much about electronics as mentioned above.
- Because not pay much attention to how the activities of the module work, so most users will be hard to handle when there are issues arising out of the library.
- The prototype module made for Arduino does not have high durability, the goal is to simplify the process of making products.
- The projects for Arduino are open source, you can easily get it and put it into production with good quality and learn a lot from how to design the program of the masters.
- IDE designed well, can easily integrate many types of the compiler, a variety of hardware without reducing performance. Example: Original Arduino for AVR, but there are many versions for STM32, PIC32, ESP8266, ESP32 and so on. Making the most of the available libraries.
2.2.2. Arduino Programming Language

The Arduino programming language has compiled on the Arduino Software Integrated Development Environment (IDE) and can be downloaded from a website https://www.arduino.cc/. This web is the main Arduino community or even install directly from Microsoft Store of Microsoft Window. The interface of Arduino IDE:

![Arduino IDE Software Interface](image)

*Figure 2: The interface of Arduino IDE Software*

2.2.3. The Arduino Serial Monitor

![Arduino Serial Monitor Window](image)

*Figure 3: The Arduino serial monitor window*
The Arduino Serial Monitor allows the user to add the facility to control devices from the computer by using a cable to connect between the computer and Arduino boards or some development boards which are combined a microcontroller use the Arduino platform to programming. The Serial Monitor lets compiler send and receive the data text message, helpful for debugging and also control Arduino from a keyboard [6]. A good example to this point, a compiler write statement code for ESP32 and uploaded the sketch onto the Arduino click on the right button is in a red square which is shown below to turn to Arduino Serial Monitor window:

![Figure 4: Example code and open the Arduino Serial Monitor](image)

![Figure 5: The Arduino Serial Monitor send information on the monitor window of the computer](image)
This process is illustrated in Figure 5 where the ESP32 is connected with the local Wi-Fi and showing its local IP via the computer screen using the Serial Monitor.

2.3. ESP32 Introduction

2.3.1. ESP32 Board

ESP32 is a developed version of ESP8266 designed by Espressif Systems with a lot of features and combined dual chips Wi-Fi and Bluetooth wireless capabilities.

![ESP32 board combines Wi-Fi and Bluetooth wireless capabilities](image)

Figure 6: ESP32 board combines Wi-Fi and Bluetooth wireless capabilities [8]

There are various development boards, the website http://esp32.net show where each ESP32 chip and development board are listed. There is some example of model boards which use chip ESP32 and in this assignment is used the ESP32 DEVKIT V1 board that is a basic kit board with ESP-WROOM 32 chip.

![Different types of ESP32 model board](image)

Figure 7: Different types of ESP32 model board [8]
2.3.2. Features Of ESP32 DEVKIT V1

The operating voltage of ESP-WROOM-32 chip is 3.3 [V] but the ESP32 DEVKIT V1 DOIT is possible to supply by voltage in range 7 – 12 volts via the “VIN” pin. The communication between computer and ESP32 Board is a CP2102 chip as a USB Drive and to program it without the need of FTDI programmer.
There are two on-board buttons: the ENABLE and the BOOT button. The button ENABLE when press on it has a function to reboot the ESP32 and hold down the BOOT button and then press the ENABLE the ESP32 reboots in programming mode.

2.3.3. Specifications:

- Integrated Wi-Fi and Bluetooth.
- Running with 32-bit programs.
- ESP32 has 512 KB RAM and the clock frequency can be reached at 240 MHz.
- This particular board has 30 pins (another new version of ESP32 board with 36 pins).
- There are a host of peripherals available such as capacitive touch, DACs, ADCs, UART, SPI, I2C…

<table>
<thead>
<tr>
<th>Specifications of ESP32 DEVKIT V1 DOIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. cores</td>
</tr>
<tr>
<td>Wi-Fi</td>
</tr>
<tr>
<td>Bluetooth</td>
</tr>
<tr>
<td>Clock Frequency</td>
</tr>
<tr>
<td>Architecture</td>
</tr>
<tr>
<td>Pins</td>
</tr>
<tr>
<td>RAM</td>
</tr>
<tr>
<td>Communication Protocols</td>
</tr>
<tr>
<td>Analog Input Pins (ADC)</td>
</tr>
<tr>
<td>ADC Resolution</td>
</tr>
<tr>
<td>Analog Outputs Pins (DAC)</td>
</tr>
<tr>
<td>DAC Resolution</td>
</tr>
</tbody>
</table>
2.3.4. ESP32 DEVKIT V1 Pinout (ESP-WROOM-32 chip)

The figure below shows the board GPIOs and their functionalities of ESP32 with 30 pins version.

![ESP32 DEVKIT V1 - DOIT](image)

*Figure 11: Pinout of ESP-WROOM-32 with 30 pins version [7]*

Notice: From data sheet esp-wroom-32

* Pins SCK/CLK, SDO/SD0, SDI/SD1, SHD/SD2, SWP/SD3 and SCS/CMD, namely, GPIO6 to GPIO11 are connected to the integrated SPI flash integrated on the module and are not recommended for other uses.

2.3.5. Absolute Maximum Ratings

*Table 2: Absolute Maximum Ratings [9]*

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDD33</td>
<td>Power supply volatge</td>
<td>-0.3</td>
<td>3.6</td>
<td>V</td>
</tr>
<tr>
<td>I&lt;sub&gt;output&lt;/sub&gt;</td>
<td>Cumulative IO output current</td>
<td>-</td>
<td>1.100</td>
<td>mA</td>
</tr>
<tr>
<td>T&lt;sub&gt;store&lt;/sub&gt;</td>
<td>Storage temperature</td>
<td>-40</td>
<td>150</td>
<td>°C</td>
</tr>
</tbody>
</table>


2.3.6. Recommended Operating Conditions

Table 3: Recommended Operating Conditions [9]

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typical</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDD33</td>
<td>Power supply voltage</td>
<td>2.7</td>
<td>3.3</td>
<td>3.6</td>
<td>V</td>
</tr>
<tr>
<td>I_{VDD}</td>
<td>Cumulative IO output current</td>
<td>0.5</td>
<td>-</td>
<td>-</td>
<td>mA</td>
</tr>
<tr>
<td>T</td>
<td>Storage temperature</td>
<td>-40</td>
<td>-</td>
<td>85</td>
<td>°C</td>
</tr>
</tbody>
</table>

2.4. Programming Environment For ESP32

Programming Languages

The programming environments of ESP32 are quite diverse. It can be programmed in Arduino IDE, Espressif IDF (IoT Development Framework), Micropython, JavaScript, LUA and so on. Throughout this bachelor thesis will be concentrating mainly on programming the ESP32 with the Arduino IDE. In the Appendix One - Arduino IDE Programming Environment have the guide on how to establish the programming Arduino IDE Environment for ESP32.

2.5. Brief Introduction Of MQTT Protocol

2.5.1. What Is The MQTT Protocol?

MQTT or Message Queuing Telemetry Transport is a lightweight or simple messaging protocol send by Publish and Subscribe used for devices of the Internet of Thing with low bandwidth and plays a significant role in the IoT, high reliability and ability to be used in the network which is not stable. Because of the low bandwidth so this issue leads to the environment of connection has a high latency period, so this protocol is applied in a wide range of M2M (Machine to Machine) communication.

2.5.2. MQTT – Publish/Subscribe

The MQTT protocol is established by 2 main elements Publish and Subscribe system which a device can send (publish) a message on a topic or received a message from a subscribe channel to a particular topic. In a system uses MQTT protocol has some nodes station (also-called MQTT clients or clients) connected with MQTT server (Broker) that means MQTT Client does not connect directly to each other, all data packets are sent through MQTT Broker. Each client will register many topic channels such as topic/control/light1 or topic/control/light2 this process called Subscribe and client will receive data from any nodes station send to a channel which user registered so this process called Publish.

In Publish and Subscribe in the MQTT protocol system is shown in the Figure below, device 1 publishes on a topic and device 2 subscribe on the same topic, it leads to the device 2 received messages from the topic which device 1 is publishing in. Messages in the MQTT are data or information that user want to exchange.
between devices together. MQTT-topics also play an important role in the MQTT protocol system. Topics are addresses where users want to define publish the message or register for an incoming message from subscribing.

Figure 12: Publish and Subscribe in the MQTT protocol system

2.5.3. MQTT – Broker

Receiving, filtering all messages, deciding which client is interested in them and publishing messages to the subscribed clients, whole these functions made by MQTT - Broker. Free MQTT cloud broker on the Internet and two websites of the most popular are https://www.cloudmqtt.com/ and http://www.thingstud.io/ with large quantity customers around the world.
2.6. PWM Control Method

Pulse Width Modulation Method:

The technique Pulse Width Modulation or also-called PWM is a method for generating analogue results using the digital signal. Two main elements of PWM method are a duty cycle and a frequency which define its behaviour. The duty cycle illustrates the volume of time when the signal is in a state on (high) which is the proportion percentage of the whole time to accomplish one period. The frequency verifies the speed of the PWM method to complete one cycle. For example, in this case, the frequency given 1000 Hz would be 1000 cycles for one second. Addition to this, the frequency also determines how fast PWM switches between two status on (high) state and off (low) state. In a fast speed enough, by cycling the digital signal switch on and off and in a certain duty cycle, the behaviour of the output will show a constant value of the voltage for an analogue signal when supply power to devices [10].

![Figure 14: Duty Cycle 25%, 50% and 75%](image)

The digital signal give half of the cycle for the state on and one half for state off. This issue would be known as the signal has the duty cycle of 50% and that is the ideal square waveform. The duty cycle can be lower or higher than 50%. If it spends more than 50% that means it takes the time for the state on longer than state off, but inverse if lower than 50% the duty cycle spends the time for state off more than the state on.
3. Practical Part To Control The Light

To implement the practical part, it is important to recognize that the IoT solution should be used for the MQTT protocol. Thereby, creating an MQTT-Broker has been regarded as integral elements where plays a role as a place to receive and transform the data from its clients and establish some integrated parts of the broker. In Appendix Two, there is the detailed guideline of how to create a new MQTT-Broker on the CloudMQTT webpage with 5 free users accesses. The guideline to create a new instance for user access projects.

3.1. Simulation Of Connection Between Devices

As mentioned in the Introduction, bachelor thesis using IoT solution connect the ESP32 as a client to control the light by MQTT protocol.

Figure 15: Block diagram remote light control via MQTT protocol using the ESP32 as a client

Block diagram above provides information about the basic principle operating of remote light control. From the left to the right we have 4 lights with different states; the light 1 and 3 are on and shining, the light 2 is off and for the light 4 there is used PWM method to control the brightness of light 4. In this case, for the simple example, the MQTT-Broker has the topic Topiclight1. Devices such as PC or laptop, tablet, mobile phone that publish “on” and “off” message to that topic and then from MQTT-Broker send the statement “on” or “off” message to ESP32 which control the lights being on state or off state, is subscribed to the same topic on MQTT-Broker.
3.2. Experimental LED Control Version

Figure 16: Control LED using ESP32 via MQTT-Broker

Components:
4 resistors 220 Ω
Wires and micro-cable
ESP32 DEVKIT V1
3 LED 5mm for turn on and off
1 LED 5mm for dimming by PWM method.

3.2.1. ESP32 Network Connection With The Local Wi-Fi Network

After preparing the environment for connection of devices, moving to compile code for ESP32 to connect to the Internet, at first the ESP32 need connected to local Wi-Fi network and set up the ESP32 as a station mode (STA) and simultaneously, it is provided with the name of the network SSID and Wi-Fi network. When it is connected successfully into the Wi-Fi network then the ESP32 plays a role as a client to receive the statement send by a topic in the broker which was published before. The Wi-Fi Library need to be included which will allow the ESP32 to connect to the network and then declare the name of SSID and its password for the Wi-Fi network.
Source Code:
#include <WiFi.h>
#define ssid "ESP32" ▶ replace the user name of network.
#define password "123456789" ▶ replace the user password of network

void setup() {

//Establish serial data transmission at a speed of 115,200 bits /s
Serial.begin(115200);
setup_wifi();
void setup_wifi() {
  delay(100);
  Serial.println();
//serial print line the status of Wi-Fi connection
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password); //establish the ESP32 in the station state and connect to Wi-Fi which is defined
  while (WiFi.status() != WL_CONNECTED) { //situation the ESP32 cannot connect to Wi-Fi
    delay(500);
    Serial.print("");
  }
  Serial.println(""");

  Serial.println("WiFi connected"); //seriral print line the status Wi-Fi is connected
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP()); //serial print line the local IP address
}

void loop() {
}

In the Serial Monitor appears the connection state of ESP32 with the local Wi-Fi and showing the local IP address of it as mentioned Figure 5 above in the Arduino Serial Monitor part.

3.2.2. ESP32: Data Publishing/Subscribing To MQTT Topic

To publish a message to a MQTT topic at first need to create MQTT-Broker which is established and hosted on CloudMQTT as mentioned in the Creating MQTT-Broker part and the Create A Broker Instance. From the instance information page similar to shown in Figure 48: The instance details information for MQTT-Broker. The server, the user, the password and the port plays a significant role to compile on the ESP32 code.
Next, it is necessary to add the Arduino library to use publishing on ESP32 which is called PubSubClient. This library uses for Subscribing message also.

![Figure 17: Adding the library for Publishing and Subscribing to MQTT Topic via Arduino IDE library manager.](image)

**Source Code:**

```cpp
// The library declaration for Wi-Fi connection and PubSubclient for Publishing and Subscribing Message
#include <WiFi.h>
#include <PubSubClient.h>

// Wi-Fi network information inluclude user name (SSID) and its password
#define ssid "ESP32"
#define password "123456789"

//MQTT-Broker (server)
#define mqtt_server "m16.cloudmqtt.com"
const uint16_t mqtt_port = 18227; //Port CloudMQTT port of broker

//Declare the user access user and password
#define mqtt_user "ESP32"
#define mqtt_pwd "12345678"

//Defined the Wi-Fi client
```
WiFiClient espClient;
PubSubClient client(espClient);
void reconnect() {
  // Waiting for ESP32 connected to MQTT-Borker
  while (!client.connected()) {
    Serial.println("Attempting MQTT connection...");
    // Execute the connection of ESP32 with MQTT user and password
    if (client.connect("ESP32Client", mqtt_user, mqtt_pwd)) {
      Serial.println("ESP connected to MQTT");
      // Announce information about published on MQTT-Borker
      client.publish(mqtt_topic_pub_light1, "ESP controlled light 1");
      // ... and received this information
      client.subscribe(mqtt_topic_sub_light1);
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" Try again in 5 seconds");
      // wait 5s
      delay(5000);
    }
  }
}

![Websocket](image)

**Figure 18:** Publishing and Subscribing to the topic of MQTT cloud
Uploading the code from the sketch, then logs in on the MQTT-Broker. When the code uploaded, the ESP32 subscribes the same topic of the broker which is published. Besides, the broker announces that ESP32 is connected and shown the topics which ESP32 are subscribing.

3.2.3. Switching And Dimming The LED Via The MQTT-Broker

Figure 19: Websocket MQTT-Broker sends and receives the message.

The instance project on the category of the cloudmqtt webpage chooses the WEBSOCKET UI will appear information about connecting to a broker (server cloudmqtt) marked on the red box and then upload the sketch from the computer. The information about publishing and subscribing of the project in the code compiled on Arduino IDE will be informed in Received messages box of the broker.

Publishing And Subscribing Messages And Switching The LED States

Figure 20: Sending and Receiving a message by MQTT-Broker
Turning on and off the LED via the Internet on WebSocket, the Send message red box correctly the name of the topic which creates to publish and on the source code of Arduino IDE. Import message 1 statement the LED will be turned on and send a message 0 for turn off the LED. Bachelor thesis use 4 topics for 4 LED or real light consist of 1 Topicdimmer for turn on, off and control the brightness of the LED, and 4 topics are is Topiclight1, Topiclight2, Topiclight3, they are used for normal turn on and off the load (light bulb) which are signalized by the LED.

![Figure 21: Switching information send back to Serial Monitor of Arduino IDE](image)

When sent a message 1 or 0 of the Topiclight1 from the broker, the Received messages appear the name of the topic which is chosen. LED is switched on for message 1 and off for message 0 and similarly with LED 2 and 3, this information is sent to the Serial Monitor of the Arduino IDE which is shown in Figure of Switching information send back to Serial Monitor of Arduino IDE.

### PWM Method For LED Brightness Control

```cpp
void Control_PWM_Monitoring()
{
  if(Serial.available() > 0)
  {
    String doc = Serial.readString();
    PWM = doc.toInt();
    PWM = map(PWM, 0,100,0,255);
    Serial.println(PWM);
    ledcWrite(ledChannel, PWM);
  }
}
```

![Figure 22: PWM method to control the brightness of LED](image)
Figure of PWM method to control the brightness of LED provides information about the brightness of the LED which is set to 45% and 75% by using pulse width modulation to change the width of the duty cycle with used 8-bit resolution in the control unit. Select the Topicdimmer and send number from 1 to 100 to turn control the brightness of LED brightness by sending messages directly from the broker.

![Send message](image1)

**Figure 23: Brightness control by sending a message from the broker**

![Received messages](image2)

**Figure 24: 10% brightness of LED**

**Figure 25: 75% brightness of LED**
3.3. Light Control Device

It is evident that there are some differences between the experimental and the reality version. In the experimental version, the purpose just targets at controlling the on and off states of LED or its brightness. To be able to bring light control device to reality, there are some complicated parts such as the power supply for the whole circuit board, relay part, diver and the ESP32. The majoring components consist of relays, IC LM317T TO220 (voltage regulator), driver L298N and playing the most significant role is ESP32-DEVKIT V1 which is the kit with chip to control whole the board.

With ESP32 DEVKIT V1 part (see on The Schematic Connection) required the power supply in the range from 7 to 12 volts. ESP32 DEVKIT DOIT can operate on the power around 6 to 20 volts but if beyond the threshold is 12 volts the device can overheat and damage [7]. Both sides of ESP32 has added 2 lines of extended pinouts according to the connection pins of ESP32 DEVKIT V1 and 2 lines of extended pins for power supply and ground which are used for the measurement in the other potential development projects.

The component L298N is used as a driver which plays a significant outstanding contribution of the connection between ESP32 and the relay control devices. The operating voltage for driver L298N up to 46 volts and total DC current up to 4 A. The river has 4 inputs and 4 outputs. Outputs are connected to three relays to control lights in on-off mode and 1 output is used to control the brightness by PWM. In the block of regulator, there is used the voltage regulator LM317T which has output voltage in range 1.2 volts to 37 volts that are solved the issue about the power supply to operate the circuit include the ESP32, driver and triggered the electromechanical relays.

The relay is triggered by operating voltage in the range about 3 volts to 48 volts. The electromechanical relays control devices which are connected with their LED to inform which relay is in a state on or off, when the lights switch on or off. The basic operating principle of lights control via the Internet is when sending a statement to change the status on or off of the light directly from the MQTT-Broker, ESP32 has the obligation to receive the signal and activate the statement to turn on or turn off the relays and then the lights are switched on or in reverse. In addition to this, the relays are not can be only used to control the lights but also they can be used to control electric fans, motor or to trigger some other electrical devices. The code for establishing communication environment between ESP32 and MQTT-Broker on Arduino IDE Program is shown in Appendix Three – Source Code Arduino IDE.
3.3.1. The Schematic Connection
### Table 4: List Of Components

<table>
<thead>
<tr>
<th>Components</th>
<th>Value</th>
<th>Unit</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESP32-DEVKIT V1</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>Relay connectors 5mm</td>
<td>-</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>PWM Connector 5mm</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>Relay 5-pin</td>
<td>-</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>DC power jack</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>LM317T TO220</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>Driver L298N</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>Resistor</td>
<td>220</td>
<td>Ω</td>
<td>4</td>
</tr>
<tr>
<td>Resistor</td>
<td>560</td>
<td>Ω</td>
<td>1</td>
</tr>
<tr>
<td>Resistor</td>
<td>1</td>
<td>kΩ</td>
<td>1</td>
</tr>
<tr>
<td>LED 3mm</td>
<td>-</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>Capacitor</td>
<td>0.1</td>
<td>µF</td>
<td>3</td>
</tr>
<tr>
<td>Capacitor</td>
<td>1</td>
<td>µF</td>
<td>1</td>
</tr>
<tr>
<td>Capacitor</td>
<td>10</td>
<td>µF</td>
<td>1</td>
</tr>
<tr>
<td>Capacitor</td>
<td>220</td>
<td>µF</td>
<td>6</td>
</tr>
<tr>
<td>Diode 1N4007</td>
<td>-</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>Pin Header 1×2</td>
<td>-</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>Pin Header 1×3</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>Pin Header 1×6</td>
<td>-</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>Pin Header 1×15</td>
<td>-</td>
<td>-</td>
<td>2</td>
</tr>
</tbody>
</table>
3.3.2. 3D Layout Version

Figure 26: 3D Top view layout of ESP32 control lights

Figure 27: 3D layout of ESP32 control lights
3.3.3. PCB Version

The PCB Layout Without The Components

![PCB layout version without the components](image)

Figure 28: PCB layout version without the components

The Final Circuit Board With The Components.

![The final circuit board with the components](image)

Figure 29: The final circuit board with the components
4. Conclusion

In conclusion, technology is ever-changing, the normal connection method which is especially physical switching devices should be replaced by using the Internet control that is a mega-trend to supporting and providing a convenient for the consumer. The bachelor thesis concentrates on the basic analyze the possibilities of remote control from much state-of-the-art communication such as local Wi-Fi, the Internet, Bluetooth, Radio-Frequency, sensor and so on. It is in fact that the control distances of these methods, Bluetooth has very short distance which is in the range of about more than 10 meters, the local Wi-Fi is also stopping at the limit about several dozen meters or even a hundred-meter. But via the Internet, the control devices can be connected from very long distances; it could be connected from any country or wherever has the Internet connection. Throughout the analysis, to be able to control from long distances, using the IoT solution could be taken into account with its influence in many fields and its multi-function as an ultimate method via the Internet to connection control devices and drive them to become smart things.

As far as I am concerned, the IoT solution should be selected for implementing the remote control modes. The remote light control by directly sending data of the broker from a mobile phone or from browser from PC for active the ESP32 as a client to switching states of three lamps, another lamp is controlled by PWM method but changing the values of brightness directly from MQTT-Broker also. But before that, the user should establish a free broker from some sources on the Internet or can create for their own. Installing the Arduino IDE and compiling code for ESP32 kit or some types of ESP32 chip version. Publishing and Subscribing the topic on the MQTT-Broker for sending, receiving message information and store data.

MQTT Protocol is instead for HTTP, simply because of its lightweight sending message between machine to machine. For research or study can use free MQTT-Broker in many webpages. ESP32 consists dual-core processor for Wi-Fi and Bluetooth which are useful for control electronic devices and develop to connecting to the Internet to sending data to the cloud, server for long distance and supporting for some devices which is not necessary to control from an extended distance. The bachelor thesis is not only stopping at the remote light control but it also could be extended to establish the smart home system to control some other devices such as switching the motor, the air conditioner, electric fan, adjust the level of the heater, speed of motor thanks to the PWM method or perform the weather station to inform users about the house temperature.

I have been gained enormous benefits from bachelor thesis. First of all, I need to find out the possibilities of remote control and select the optimal way to implement them. Then, choose the program platform to compile the code which I choose is the Arduino IDE simply because of the Arduino IDE have an available built-in library and it is also as open-source as I mentioned in the Arduino IDE Introduction and thanks to this case, the programmer can easily to use it for their project. In the bachelor thesis process, I can enrich my knowledge about draw the schematic and makes the PCB layout of the circuit by using Eagle Autodesk software. I have an opportunity to approach new experiences of how to drill the holes and soldering the electronics components on the circuit board. Likewise, from the selection of the software or platform to program, it contributes to reap the skills to improve the knowledge.
5. References


Appendix One - Arduino IDE Programming Environment

Preparing Arduino IDE Environment for ESP32

Making sure that Arduino IDE installed in the computer is the latest version before starting to the installation procedure. Just like the ESP8266, the ESP32 is also currently being integrated with the Arduino IDE which allows to program ESP32 using the Arduino IDE and its programming language.

Installing Programming Environment In Arduino IDE

To communicate between ESP32 and Arduino IDE take several simple following steps:
Step 1: Go to the main website of Arduino https://www.arduino.cc/
Software ► Downloads ► choose the correct version for Windows (Windows, Windows app store Mac OS X, Linux). Download and install the Arduino IDE.

Download the Arduino IDE

Figure 30: The main page for download the Arduino IDE

Choose correct git version for set-up windows 32-bit or 64-bit.
Step 3: Launch the GIT GUI when install GIT has done, search for “GIT GUI” and open it, do not open GIT bash.

![Git GUI application interface](image)

*Figure 33: The GIT GUI application interface*

Step 4: Under Source Location paste the link [https://github.com/espressif/arduino-esp32.git](https://github.com/espressif/arduino-esp32.git) and under Target Directory paste the link that is the location of ESP32 in Arduino so before do that the ESP32 must create a folder [ARDUINO_SKETCHBOOK_DIR]/hardware/espressif/esp32 with the [ARDUINO_SKETCHBOOK_DIR] can be seen in File ➤ Preferences on the Arduino IDE

![Source Location and Target Directory](image)

*Figure 34: Source Location and Target Directory*
Step 5: Open Git Bash and open it and get the following window.

![Git Bash window](image)

*Figure 35: Git Bash window*

Type “cd” and paste the Target Directory in here then press enter. After that, paste git submodule update --init --recursive and press enter again and close the window.

**Installing The Windows USB Driver**

Connecting between computer and ESP32 via micro USB cable does not know what driver is used for connecting them. This can be solved in the computer manager.

![Computer manager with missing CP2102 driver](image)

*Figure 36: The computer manager with missing the CP2102 driver*
The computer does not recognize CP2102 USB to UART Bridge Controller which need to reach this link https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers to install the CP2102 to download and install USB Driver CP2102.

**Installing The ESP32 Board With Arduino IDE**

Follow the steps below to install the ESP32 board:

Step 1: Open the Arduino Windows go to File ▶ Preferences 
Filling https://dl.espressif.com/dl/package_esp32_index.json into the “Additional Board Manager URLs” as shown in the figure below and then press “OK”. If Arduino added ESP8266 board URL before, just use a comma to separate the URLs of ESP32 and ESP8266 as the following figure.

![Figure 37: Added the URL of ESP32 to Arduino IDE](image)
Step 2: To open board manager go to Tools ► Board ► Boards Manager…is shown below:

![Image of Boards Manager window](image1.png)

*Figure 38: Loading ESP32 Board from Boards Manager*

Step 3: Install ESP32 by Espressif Systems

![Image of Boards Manager window](image2.png)

*Figure 39: Install ESP32 by Espressif Systems*

In the Boards Manager, searching window fills ESP32 and choose ESP32 by Espressif Systems and then click Install button to set up library boards types of ESP32 on Arduino IDE.
Step 4: Correct selection board for ESP32:

![Figure 40: Selection of the ESP32 Board to programming](image)

After installation Esp32 Board. Before starting to programming on ESP32, plug the ESP32 DOIT DEVKIT V1 to the computer by USB cable then open the Arduino IDE and select correctly the board version which types of ESP32 models the user using. After this step, the ESP32 is ready to program directly Arduino IDE with Arduino style coding.
Appendix Two – MQTT-Broker

Creating MQTT – Broker


![The interface free hosted message broker for the IoT](image)

*Figure 41: The interface free hosted message broker for the IoT*

From the interface of cloudmqtt press “Get managed IoT broker today” to register a new account. Then, from Plans & Pricing showing on the figure below roll down and choose free version broker Cute Cat for study or research. It is very easy to create an account on cloudmqtt.com, the user can register by email, sign up with Google or GitHub account.

![Cute Cat with a free broker for IoT solution.](image)

*Figure 42: Cute Cat with a free broker for IoT solution.*
Figure 43: Fill in the email to create a new account cloudmqtt.com

Creating New Instance

Step 1: Create a new instance in the Cute Cat plan for free. Fill any name to illustrate instance after filled all press “Select Region” to move to step 2.

Figure 44: Select the name of the instance and add tags for separate instance between projects.

Step 2: Select a region and data centre. The data centre has two main regions are US-East-1 (Northern Virginia) and EU-West-1 (Ireland). Then press the “Review” button to go to step 3.
Step 3: Checking correctly information before confirm to create the new instance. If there is some mistake or user want to rewrite information can turn back to the previous steps and if not just press “Create instance”.

Figure 45: Select a region and data centre for instances

Figure 46: Checking information about free MQTT-broker
The new window appears after to create a new instance contain all information about the user’s project. Inside the ESP32-MQTT-Broker has details information such as Server, User and Password, Port and SSL Port, Websocket Port (using for TLS only) and Connection limit (in the Cute Cat free plan MQTT-Broker the connection limit is 5) which is shown in Figure of Instance info.

After installing the connection environment between ESP32 and Arduino IDE and the preparation steps for editing the code directly to ESP32 on Arduino IDE, moving to the practical part with a experimental version using LED for testing the code and communication between clients (Laptop, Cellphone, ESP32) via MQTT-Broker to control LED before make the practical version for controlling lights.

![Instance info](image1)

**Figure 47: The interface of the new instance consist of information light control project**

![Instance details](image2)

**Figure 48: The instance details information for MQTT-Broker.**
Creating User Access And ACL Rules

To create the User Access and the ACL rules for the hosted MQTT Instance click the left button USER & ACL as shown below and filled the user name and the password then click Add in the green button.

![Figure 49: Create user access for MQTT-Broker](image)

![Figure 50: Create the ACLs rules, topic and pattern](image)

The topic ACLs is used for a given user and meanwhile, for all users, the pattern is recommended. When choosing the pattern or topic need to tick check the Read Access, Write Access or both to allow the user can approach MQTT-Broker to control or establish topics for devices.
Appendix Three – Source Code Arduino IDE

Full Source Code:

```c
#include <WiFi.h>
#include <PubSubClient.h>
#include <string.h>
// Wi-Fi information
#define ssid "ESP32"
#define password "123456789"

//declare MQTT-Broker
#define mqtt_server "m16.cloudmqtt.com"

//light1
#define mqtt_topic_pub_light1 "Topiclight1"
#define mqtt_topic_sub_light1 "Topiclight1"

//light2
#define mqtt_topic_pub_light2 "Topiclight2"
#define mqtt_topic_sub_light2 "Topiclight2"

//light3
#define mqtt_topic_pub_light3 "Topiclight3"
#define mqtt_topic_sub_light3 "Topiclight3"

//light4
#define mqtt_topic_pub_dimmer "Topicdimmer"
#define mqtt_topic_sub_dimmer "Topicdimmer"

#define mqtt_user "ESP32"
#define mqtt_pwd "12345678"
int PWM = 0;
String doc;
const uint16_t mqtt_port = 18227; //Port của CloudMQTT
const int Fhz = 5000;
const int ledChannel = 0;
const int resolution = 8;

WiFiClient espClient;
PubSubClient client(espClient);

long lastMsg = 0;
char msg[50];
int value = 0;
char receivedChar;
String test;
char* Test2;
String Input1;
int    Input2;

void setup() { 
    Serial.begin(115200);
    setup_wifi();
    client.setServer(mqtt_server, mqtt_port);
    client.setCallback(callback);
    //setup pinmode
    pinMode(Light_1, OUTPUT);
    pinMode(Light_2, OUTPUT);
    pinMode(Light_3, OUTPUT);
    pinMode(Dimmer, OUTPUT);
    // initial condition state
    digitalWrite(Light_1, LOW);
    digitalWrite(Light_2, LOW);
    digitalWrite(Light_3, LOW);
    digitalWrite(Dimmer, LOW);
    ledeSetup(ledChannel, Fhz, resolution);
    ledeAttachPin(Dimmer, ledChannel);
}

// Wi-Fi function to communicate
```
void setup_wifi() {
  delay(100);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println(" ");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

// Callback function to receive data
void callback(char* topic, byte* payload, unsigned int length) {
  String receivedChar = (String)receivedChar;
  Serial.println(receivedChar);
  Input1 = Input1 + receivedChar;
}

//light1
if (strcmp(topic, "Topiclight1") == 0) {
  Serial.println("Light 1 here");
  if (receivedChar == '1') {
    digitalWrite(Light_1, HIGH);
    Serial.println("Light 1 is On");
  } else if (receivedChar == '0') {
    digitalWrite(Light_1, LOW);
    Serial.println("Light 1 is Off");
  }
}

//light2
else if (strcmp(topic, "Topiclight2") == 0) {
  Serial.println("Light 2 here");
  if (receivedChar == '1') {
    digitalWrite(Light_2, HIGH);
    Serial.println("Light 2 is On");
  } else if (receivedChar == '0') {
    digitalWrite(Light_2, LOW);
    Serial.println("Light 2 is Off");
  }
}

//light3
else if (strcmp(topic, "Topiclight3") == 0) {
  Serial.println("Light 3 here");
  if (receivedChar == '1') {
    digitalWrite(Light_3, HIGH);
    Serial.println("Light 3 is On");
  } else if (receivedChar == '0') {
    digitalWrite(Light_3, LOW);
    Serial.println("Light 3 is Off");
  }
}

//light4 dimmer
else if (strcmp(topic, "Topicdimmer") == 0) {
  PWM = Input1.toInt();
  if (PWM >= 100) {
    PWM = 100;
    Serial.println(PWM);
  }
  PWM = map(PWM, 0, 100, 0, 255); //range 0-255
  ledcWrite(ledChannel, PWM);
  Serial.println(PWM);
}

else {
  Serial.println("Incorrect topic");
}

// Serial.println();
// reconnect function to execute reconnection
// when lost communication between esp32 and
// MQTT-Broker
void reconnect() {
  // wait to connecting
  while (!client.connected()) {
    Serial.println("Attempting MQTT connection...");
    // Execute connection with broker use user
    name and pass
    if (client.connect("ESP32Client", mqtt_user,
                   mqtt_pwd)) {
      Serial.println("ESP connected to MQTT");
    // Publish inform when connected
      client.publish(mqtt_topic_pub_light1, "ESP
controlled light 1");

    // ... receive inform subscribe
      client.subscribe(mqtt_topic_sub_light1);
      client.publish(mqtt_topic_pub_light2, "ESP
controlled light 2");

      client.subscribe(mqtt_topic_sub_light2);
      client.publish(mqtt_topic_pub_light3, "ESP
controlled light 3");

      client.subscribe(mqtt_topic_sub_light3);
      client.publish(mqtt_topic_pub_dimmer,
      "ESP controlled dimmer 4");

      client.subscribe(mqtt_topic_sub_dimmer);
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" Try again in 5 seconds");
      // wait 5s
      delay(5000);
    }
  }
}

void Control_PWM_Monitoring() {
  if(Serial.available() > 0) {
    String doc = Serial.readString();
    PWM = doc.toInt();
    PWM = map(PWM, 0,100,0,255);
    Serial.println(PWM);
    ledcWrite(ledChannel, PWM);
  }
}

void loop() {
  // check connection
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  if(Serial.available() > 0) {
    while(Serial.available()) {
      doc = Serial.readString();
      Serial.println(doc);
      PWM = doc.toInt();
      PWM = map(PWM, 0,100,0,255);
      Serial.println(PWM);
      ledcWrite(ledChannel, PWM);
    }
  }
}