Design and Implementation of an Online-Controlled Running Text Display

Hamdani, Suherman, Alamsyah Harahap

Abstract

Running text is one of the digital media to display information to the publicwith the help of LED (Light Emitting Diode). The information to be displayed should always be up to date, in other words the information displayed is as much aspossible the latest information. Therefore we need a device that can replace the information in the running text quickly and practically, one way is by using the telegram application. This system is controlled by the NodeMcu with the ESP8266 microcontroller. While the text display is displayed by 5 P10 LED modules. NodeMCU acts as a gateway connecting system and humans, also acts as a controller for the P10 LED Module. All modules in the system including the NodeMcu and P10 modules use a 5V power supply. Then only 1 power supply module is needed that can convert 220VAC (PLN) electricity to 5V DC. Users communicate with NodeNcu using an internet connection via the Telegram messaging application. Telegram requires a third-party bot that can connect nodemcu users and devices directly. The bot used is the "UniversalTelegram Bot".

Keywords: Running Text, Telegram, NodeMcu ESP 8266, Dot Matrix P10.

Hamdani[1]
[1]Electrical Engineering, Universitas Pembangunan Panca Budi, Indonesia
e-mail: hamdani.stmt@dosen.pancabudi.ac.id[1]

Suherman[2], Alamsyah Harahap[3]
[2]Electrical Engineering, Politeknik Negeri Medan, Indonesia
[3]Program Study of Electrical Engineering, Universitas Pembangunan Panca Budi, Indonesia
e-mail: suherman@polmed.ac.id[2], alamsyahharahap@gmail.com[3]

Introduction

The demand for dynamic information dissemination has led to the development of digital display systems that can present real-time updates. Running text or scrolling LED displays have become more and more popular in public places, transportation hubs, campuses, and business areas. However, most conventional systems require manual message input through local interfaces such as serial ports or SD cards, which is inefficient and limits accessibility.With the rapid development of the Internet of Things (IoT), new opportunities have emerged to enable remote and real-time control of display systems. IoT technology allows devices such as microcontrollers to communicate through the internet using standard protocols, providing flexibility and ease of use.This research focuses on the design and implementation of an online-controlled running text display system using the microcontroller. The ESP8266 was chosen for its built-in Wi-Fi capability, low cost, and compatibility with open-source development tools. The system uses a cloud server that stores and transmits messages to the display unit, allowing users to update information from anywhere.The objectives of this study are:To design a hardware and software system for an internet-based running text display.To develop a telegram application based user interface for message management.To evaluate system performance in terms of update latency, stability, and usability.

Literature Review

2.1 Traditional LED Display Systems

Early LED display systems commonly used microcontrollers such as ATmega328 and PIC16F877A, with message data stored locally in EEPROM [1], [9]. These systems were reliable for static messages but lacked flexibility for frequent updates and remote operation.

2.2 Wireless-Based Display Control

Wireless display systems evolved using Bluetooth [2], [10] and GSM [3], [11]. Bluetooth offered low-cost short-range communication but was limited to distances below 10 meters. GSM-based systems supported remote control but incurred SMS charges and exhibited message delays.

2.3 IoT-Based LED Display Systems

The introduction of Wi-Fi-enabled microcontrollers such as ESP8266 and ESP32 revolutionized remote display technologies. Research showed that ESP8266-based LED control improved accessibility via local web servers [4], [12], but was restricted by LAN network limitations. ESP32 introduced more robust capabilities including dual-core processing, higher clock speed, and better Wi-Fi performance, enabling IoT-driven signage [5], [13].

2.4 Cloud-Integrated Digital Signage

Cloud platforms allow multi-user access, real-time synchronization, and global scalability. Studies such as M. Ali et al. [6] demonstrated locally hosted web servers but lacked cloud functionality. Advanced implementations incorporated cloud messaging and MQTT-based communication to optimize digital signage networks [7], [8], [14]–[20].

Research Methodology

3.1 Hardware design

The following shows the hardware design in the form of a block diagram

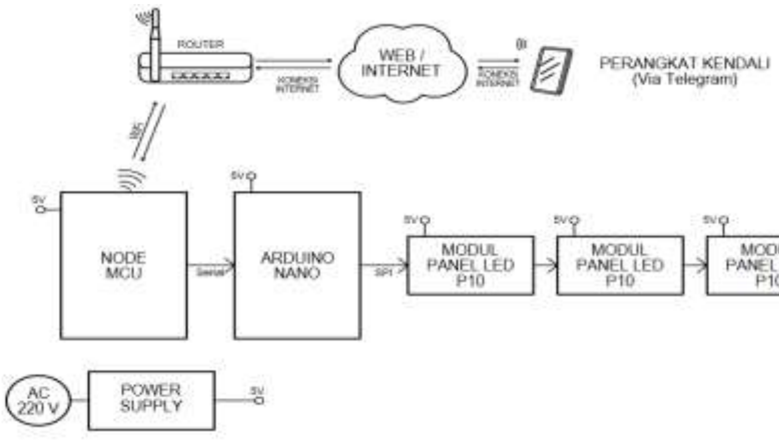


Figure 1: Block Diagram

According to the block diagram, the circuit consists of an Arduino circuit, an ESP8266 NodeMCU, a power supply, and a P10 LED module. This circuit can be seen in Figure 2. Furthermore, the circuit is connected to a printed circuit board (PCB) simply and interconnectedly.

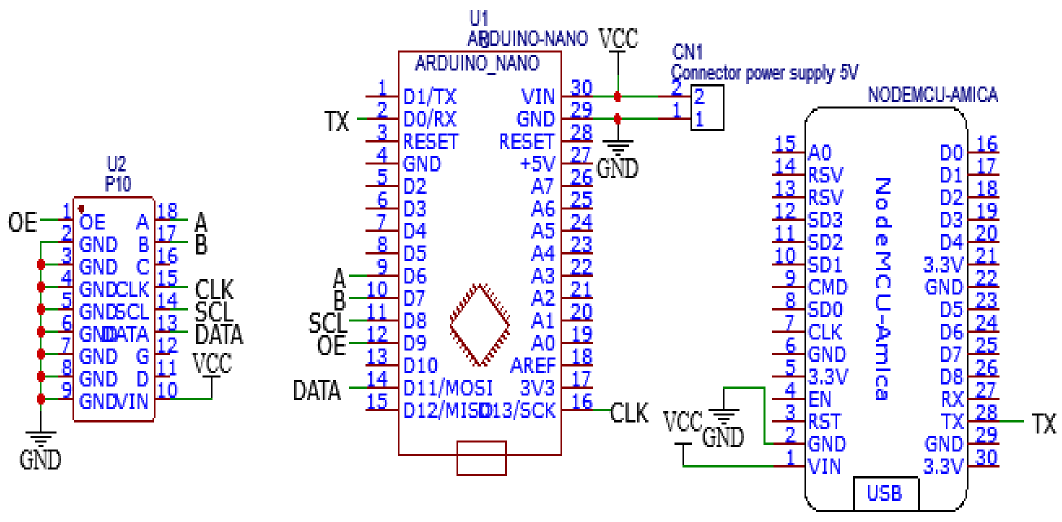


Figure 2: Circuit Diagram

In Figure 2, it can be seen that the VIN and GND symbols, which are the power supply, are connected to the Arduino and nodeMCU. The power supply is also connected to the 5V and GND pins of the P10 LED module. Then, the Arduino's D6, D7, D13, D8, D11, and D9 pins are connected to the A, B, CLK, SCL, DATA, and OE pins on the P10 LED module. The Arduino functions as the main microcontroller that controls the P10 module and as the receiver of Telegram message data from the nodeMCU. The nodeMCU then functions to receive Telegram messages and send them to the Arduino, and the P10 LED module displays running text information according to what is entered thru the Telegram application.

3.2 Program Design on Arduino

In this design, Arduino Nano is used as the microcontroller, which functions to receive data from the NodeMCU. The NodeMCU previously reads text data from Telegram via the internet network. When the Arduino Nano receives text data from the NodeMCU, it will process the text data, causing the text to be displayed on the P10 running text module.

The steps for creating a program for this system are as follows:

1. 1.The first step is to call the library for the LED module that was previously downloaded in the Arduino IDE application.
2. Include the text font library you want to use.
3. Determine the number of rows and columns of P10 LED modules used.
4. Create a serial communication initialization program to display data on the serial port, and initialize the P10 LED module program by adjusting brightness and text font type. This program is located in the void setup section.
5. Create a function program to read message data from the NodeMCU via serial communication.
6. After that, a program was created to process message data and break it down into 3 sentences.
7. Next, a function program was created to display running text information consisting of 3 sentences on the P10 LED module.

3.3 Program Design on NodeMCU

NodeMCU is used as the microcontroller, which functions to read text data from the Telegram application. This text will then be sent to Arduino Nano via serial communication for processing and display on the P10 running text module.
The steps for creating the NodeMCU program for this system are as follows:

1. The first step is to call the Wi-Fi manager, Telegram bot, and serial communication libraries that were previously downloaded in the Arduino IDE application.
2. Next, initialize the serial communication pins and create variables to store values.
3. Then, create a Wi-Fi manager program to store the token entered thru the web.
4. Next, create a program to determine the serial communication speed and a program to connect to Wi-Fi via the web. This program is located in the void setup section.
5. After that, create a program to reset Wi-Fi thru the serial monitor if you want to change Wi-Fi.
6. Then, create a program to check Telegram messages and store the information text obtained from Telegram messages, which is located in the void loop. NodeMCU will repeatedly check if there are any incoming Telegram messages.

3.4 Design of the Telegram Application for Smartphones

The initial step is to download and install the Telegram application on your mobile device. Subsequently, launch the Telegram application and select the search menu. Subsequently, input "BotFather" to establish a third-party bot that facilitates direct connectivity between users and the NodeMCU device via the internet. As illustrated in Figure 3

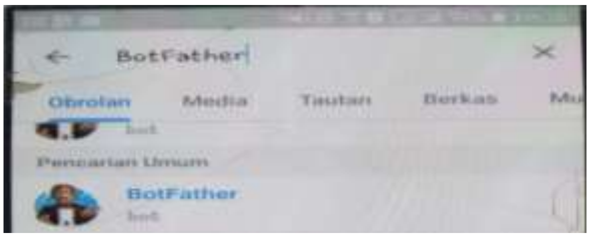
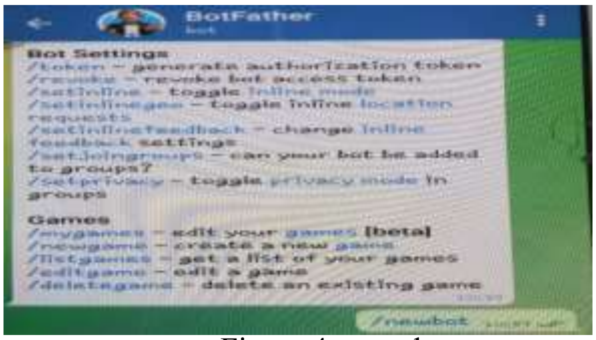


Figure 3: search telegram menu

Subsequently, pick BotFather and input /start to initiate BotFather, then enter /newbot to establish a new bot as illustrated in Figure 4


BotFather
bot

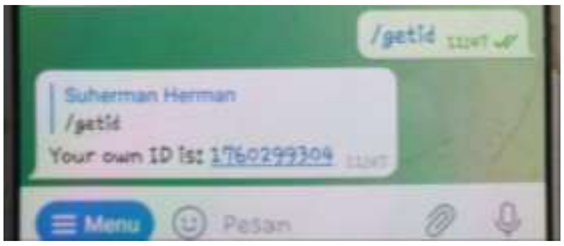
Bot Settings
/token – generate authorization token
/revoke – revoke bot access token
/setinline – toggle inline mode
/setinlinegeo – toggle inline location requests
/setinlinefeedback – change inline feedback settings
/setjoingroups – can your bot be added to groups?
/setprivacy – toggle privacy mode in groups

Games
/mygames – edit your games [beta]
/newgame – create a new game
/listgames – get a list of your games
/editgame – edit a game
/deletegame – delete an existing game

/newbot

Figure 4 : new bot

Subsequently, inscribe and input the bot's name as per your preferences, Subsequently, provide your preferred username, adhering to the specified format, ensuring that it concludes with the term "bot,"Subsequently, a screen will display confirming the successful creation of the new bot. Subsequently, insert the token from the newly generated bot into the Arduino IDE application, Subsequently, enter the bot ID in the search field. The bot ID comprises the identifying number of the newly established bot, which will subsequently be input into the Arduino IDE program.To initiate IDBot, enter /start, and to retrieve the ID of previously constructed bots, input /getid, Once the bot's ID is displayed, transfer the ID into the Arduino software to enable communication between Telegram and the nodeMCU device, as illustrated in Figure 5


/getid

Suherman Herman
/getid
Your own ID is: 1760299304

Figure 5: ID Bot was done succesfully

3.5 Design for Replacing Wi-Fi and Telegram Bot

This design is implemented if you want to replace the Wi-Fi hotspot for the nodeMCU and change to a new user or Telegram bot so that the Telegram on your new phone can send messages to the P10 running text module. This is done by making the nodeMCU an Access Point, then connecting your phone or computer to the NodeMCU to enter the new SSID, password, and Telegram token.

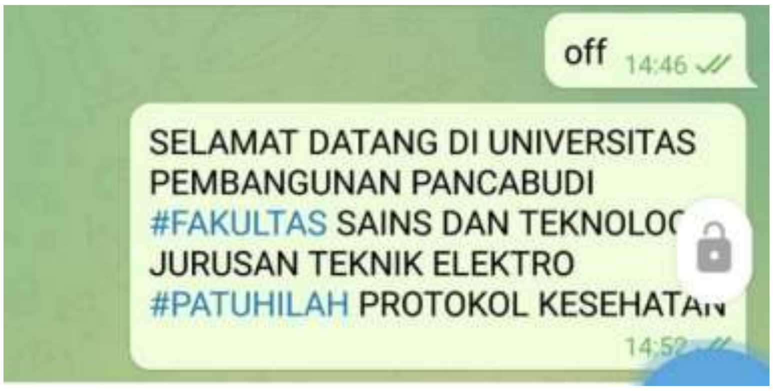
Open the Arduino application. Then select Tools, choose the COM port, and open the serial monitor, then type reset. Next, connect your computer to the Access Point of the nodeMCU.

Then open your browser and type the IP address of the Access Point nodeMCU to open the WiFi manager webpage. Next, click on "Configure WiFi" and enter the SSID, password, and bot token as desired.Then click save.

Results

4.1 Evaluating Data Transmission through the Telegram Application.

This testing was performed to ascertain whether the data transmitted via the Telegram application was received by the nodeMCU microcontroller. The testing was conducted by transmitting messages via the Telegram application to the previously established ESP8266 bot. The format of a Telegram message is "text1 # text2 # text3". The informational text will be presented in sequence, commencing with text 1, followed by text 2, and concluding with text 3. The nodeMCU's answer will subsequently appear on the serial monitor, confirming that the data has been received by the nodeMCU. The outcomes of transmitting messages via the Telegram program are illustrated in Figure 6, while the serial monitor results are depicted in Figure 7.



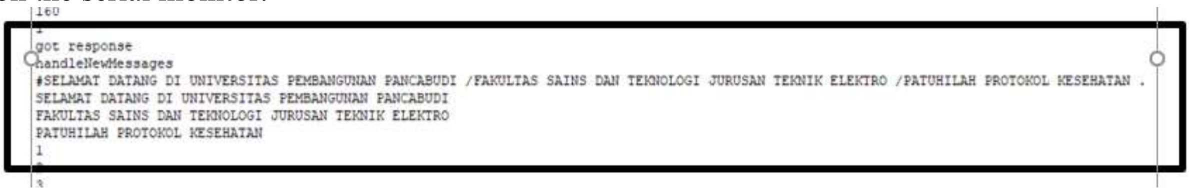
off 14:46

SELAMAT DATANG DI UNIVERSITAS PEMBANGUNAN PANCABUDI #FAKULTAS SAINS DAN TEKNOLOC JURUSAN TEKNIK ELEKTRO #PATUHILAH PROTOKOL KESEHATAN

14:52

Figure 6 : message sent format.

The test results indicate that the Telegram message reads: ". SELAMAT DATANG DI UNIVERSITAS PEMBANGUNAN PANCABUDI #FAKULTAS SAINS DAN TEKNOLOGI JURUSAN TEKNIK ELEKTRO #PATUHILAH PROTOKOL KESEHATAN " The NodeMCU serial monitor displays identical data, confirming that the message transmitted via the Telegram application has been successfully received by the NodeMCU, as evidenced on the serial monitor.



```
160
1
got response
handleNewMessages
#SELAMAT DATANG DI UNIVERSITAS PEMBANGUNAN PANCABUDI /FAKULTAS SAINS DAN TEKNOLOGI JURUSAN TEKNIK ELEKTRO /PATUHILAH PROTOKOL KESEHATAN .
SELAMAT DATANG DI UNIVERSITAS PEMBANGUNAN PANCABUDI
FAKULTAS SAINS DAN TEKNOLOGI JURUSAN TEKNIK ELEKTRO
PATUHILAH PROTOKOL KESEHATAN
1
2
3
```

Figure 7 : Serial monitor response

4.2 Testing of P10 LED Module

This testing seeks to determine whether the display output of the P10 LED module corresponds with the transmitted data. Testing was performed by transmitting a message to the Telegram application on a mobile device. then, the NodeMCU will acquire the Telegram message across the internet and transmit it to the Arduino, which will then show it on the P10 LED module. The information to be transmitted will be in textual format: "WELCOME TO PANCABUDI UNIVERSITY #FACULTY OF SCIENCE AND TECHNOLOGY DEPARTMENT OF ELECTRICAL ENGINEERING #FOLLOW HEALTH PROTOCOLS". The nodeMCU will divide this message into three texts, using "#" as the delimiter. The outcomes for text display 1 are illustrated in Figure 4.6, the outcomes for text display 2 are depicted in Figure 8, and the outcomes for text display 3 are represented in Figures 9 and 10**.**

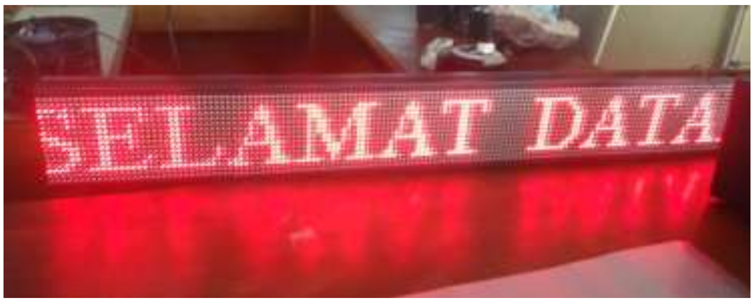


Figure 8 : shows the text display results of the first test on the P10 LED module, which is text. SELAMAT DATANG DI UNIVERSITAS PEMBANGUNAN PANCABUDI

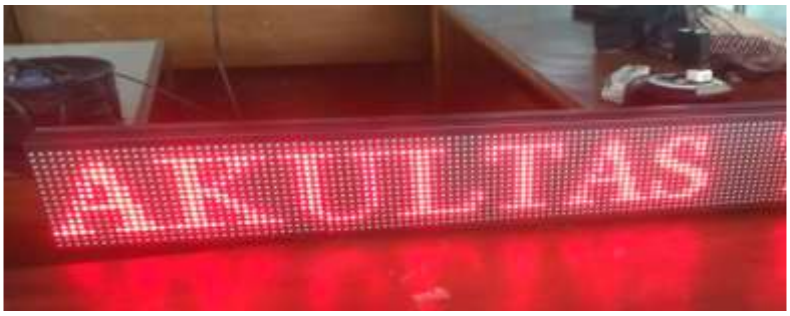


Figure 9: shows the text display results of 2 tests on the P10 LED module, which is text "FAKULTAS SAINS DAN TEKNOLOGI JURUSAN TEKNIK ELEKTRO"

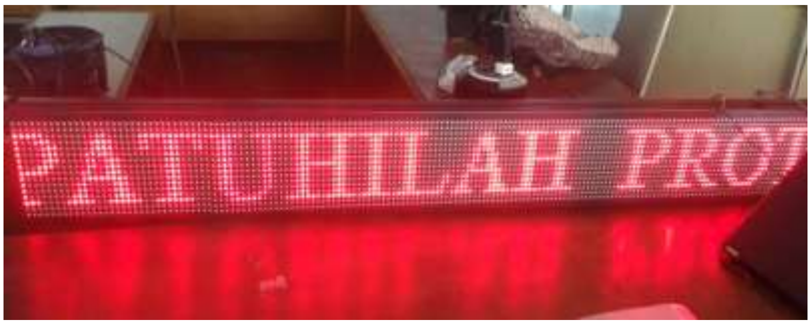


Figure 10: shows the text display results of the 2 tests on the P10 LED module, which is text. PATUHILAH PROTOKOL KESEHATAN

From the test results, it can be seen that the text displays 1, 2, and 3 are already appearing on the P10 LED module. Therefore, it can be concluded that the messages sent on the Telegram application have been received by the NodeMCU and then sent to the Arduino for processing and display on the P10 LED module.

Conclusion

The design of this DMD P10 running text display board possesses a certain allure for reading, attributed to its vibrant and attention-grabbing appearance. The DMD P10 module necessitates minimal electrical power, leading to reduced electricity expenses. It is functional and can be operated remotely.

References

[1] N. K. Suryadevara and S. R. Rao, "Design and Development of a Microcontroller Based Programmable LED Display System," IJEC, vol. 8, no. 2, pp. 45–52, 2015.

[2] R. Nair and B. Jacob, "Bluetooth Based Scrolling LED Display Using Android Device," IJERT, vol. 4, no. 5, pp. 314–318, 2015.

[3] M. A. Rahman, "GSM Based LED Scrolling Display Board," International Journal of Computer Applications, vol. 140, no. 2, pp. 1–5, 2016.

[4] R. Khatri, A. Sharma, and M. Kumar, "IoT-Based Wireless LED Display Board Using ESP8266," in Proc. ICCCA, 2019.

[5] S. Patel et al., "Wi-Fi Enabled LED Display Board Using ESP32," IEEE IoT Journal, vol. 8, no. 7, 2021.

[6] M. Ali, R. Singh, and P. Verma, "Design of an ESP8266-Based Web Controlled LED Display," in Proc. ICITEE, 2021.

[7] Y. Kim and D. Choi, "Cloud-Based Digital Signage Platform," IEEE Access, 2019.

[8] L. Wang and J. Li, "IoT-Driven Smart Display Using MQTT and ESP32," Sensors, 2021.

[9] P. Kumar and S. Jha, "PIC Microcontroller-Based LED Display," IJEEE, 2014.

[10] S. G. Rao and R. Patel, "Wireless LED Display via Bluetooth," IJSCE, 2013.

[11] N. Ahmed, "GSM Based Digital Notice Board," IJECE, 2016.

[12] A. Sen and U. Roy, “NodeMCU-Based IoT Display System,” IEEE ICACCS, 2020.
[13] H. Gupta et al., “ESP32-Based Real-Time Display,” IEEE WiSPNET, 2021.
[14] C. Park and J. Kang, “Smart Digital Signage via Cloud,” IEEE Transactions on Consumer Electronics, 2018.
[15] M. Santos et al., “MQTT-Based Signage Control,” IEEE LACS, 2020.
[16] R. Darshan and K. Mohan, “IoT Information Display System,” IJET, 2021.
[17] B. Lee and S. Kim, “Web-Controlled LED Panels,” IEEE ICIT, 2019.
[18] D. Xu and L. Yang, “Remote LED Board Monitoring,” IEEE Access, 2020.
[19] P. Chen et al., “Cloud IoT Signage Control,” ACM IoT Systems, 2021.
[20] Z. Zhao and H. Zeng, “Digital Signage Management via IoT Cloud,” Elsevier IoT Journal, 2022.