Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

"Telegram Bot Message Display Using Arduino and ESP8266"

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS

OF

Industrial Internet of Everything Laboratory
Semester VII (DJS22ICC702)

By

Group No. 3

Parshav Dedhia	**60019220120**
Devarsh Chandiwade	**60019230112**
Hiren Darji	**60019230114**
Tushar Panchal	**60019230116**

Project Guide Name

Mr. Swapnil Gharat

Department of Computer Science and Engineering
(IoT and Cyber Security with Blockchain Technology)

Academic Year 2025-2026

Mr. Swapnil Gharat

Date: _____

Declaration

We declare that any and all sources utilized in the preparation of this report have been properly cited and referenced. The ideas, concepts and research findings presented in this proposal are entirely our own, unless otherwise acknowledged and referenced. This report represents my genuine efforts to contribute to the field of Computer Science Engineering ((IoT and Cyber security with Blockchain Technology) and to advance scholarly knowledge in a meaningful and ethical manner

Parshav Dedhia	60019220120
Devarsh Chandiwade	60019230112
Hiren Darji	60019230114
Tushar Panchal	60019230116

Date:

Place:

CERTIFICATE

This is to certify that, topic entitled **"Telegram Bot Message Display Using Arduino and ESP8266"** by **Parshav Dedhia, Devarsh Chandiwade, Hiren Darji, Tushar Panchal** has been reviewed and evaluated by undersigned members, and is submitted as partial fulfillment of Industrial Internet of Everything Laboratory Semester VII (DJS22ICC702) in the Department Of Computer Science & Engineering (IoT and Cyber Security with Blockchain Technology)

_____

(Mr. Swapnil Gharat)

_____

Examiner Name & Signature

_____	_____
Dr. Narendra Shokokar	**Dr. Hari Vasudevan**
(Vice Principal & Head of Department)	(Principal)

Abstract

This mini project presents a simple IoT based system that allows users to send messages from a Telegram Bot and view them instantly on a 16x2 LCD connected to an Arduino through an ESP8266 WiFi module. The aim is to demonstrate how real time communication can be achieved between a cloud based messaging platform and embedded hardware using an easy and reliable workflow. The system listens for incoming text through the Telegram API then passes the data to the Arduino where it is shown on the serial monitor as well as on the LCD display.

The project highlights how lightweight IoT hardware can be used to create a practical notification system. The working model reflects how display boards at public places operate including railway stations where information screens update continuously based on a remotely generated message. The same concept also relates to smartwatches and IoT appliances that show alerts in real time when a message arrives from a connected device.

The setup focuses on stable communication between the ESP8266 and the Telegram servers along with proper handling of incoming messages. The LCD serves as the output unit that shows the received text clearly which allows the prototype to act as a basic live information panel. The design keeps the architecture simple so that the system remains easy to understand and reproduce for academic or industrial learning.

This mini project shows how IoT communication, cloud APIs, and microcontroller based displays can work together for real world notification systems. It also opens the scope for scaling the model into larger display networks or automated alert systems used in industries, homes, and public infrastructure.

.

Keywords: IoT, Arduino, ESP8266, Telegram Bot, LCD Display, Real Time Notification System

Table of Contents

Chapter	Contents	Page No.
i	List of Tables …….	v
ii	List of Figures ……	vi
iii	Abbreviations …….	vii
1	INTRODUCTION …….	1
2	PROPOSED SYSTEM……...	2
3	SYSTEM DESIGN AND IMPLEMENTATION……	3
4	RESULTS AND OUTPUT……	9
5	APPLICATIONS AND USE CASES……...……	12
6	ADVANTAGES AND LIMITATIONS……..	13
7	COSTING…….…….	14
8	CONCLUSION ……	15

List of Tables

Table. No.	Table Name	Page. No.
3.1	Description of each component used	5
7.1	Costing Table	5

List of Figures

Fig. No.	Diagram Name	Page No.
3.2	Circuit Diagram	4
4.1	Actual Connection Setup	9
4.2	WiFi Module Connection Established	9
4.3	Sending Message via Telegram Bot	10
4.4	Message Displayed on the Serial Monitor	
4.5	Message Displayed on the LCD Display	

Abbreviations

ESP32	Espressif Systems 32-bit Microcontroller
ESP8266	Espressif Systems
IIOE	Industrial Internet of Everything
IOT	Internet of Things
LCD	Liquid Crystal Display
I2C	Inter Integrated Circuit
WiFi	Wireless Fidelity
API	Application Programming Interface
TX	Transmit Pin
RX	Receive Pin
SDA	Serial Data
SCL	Serial Clock
GPIO	General Purpose Input Output
GND	Ground
VCC	Voltage Common Collector (Power Supply Voltage)
USB	Universal Serial Bus
SSID	Service Set Identifier
IDE	Integrated Development Environment
LCD EN	LCD Enable Pin
LCD RS	LCD Register Select Pin
ADC	Analog to Digital Converter

1. Introduction

The growth of the Internet of Things has made it possible for everyday devices to exchange information through simple and efficient communication channels. Microcontrollers paired with wireless modules now support real time data transfer which allows small systems to perform tasks that once required complex infrastructure. Messaging platforms have also opened their APIs for developers which makes it easier to build interactive systems that respond to user commands from anywhere. This mini project uses these capabilities to create a compact message display system controlled through a Telegram Bot.

The system is built around an Arduino board connected to an ESP8266 WiFi module that communicates with the Telegram servers. Whenever a user sends a message to the assigned bot the ESP8266 retrieves the text and forwards it to the Arduino. The message is then shown on the serial monitor and displayed on a 16x2 LCD which acts as the local output interface. This flow demonstrates a clear and direct link between a cloud based application and embedded hardware through standard IoT practices.

The concept relates to real world notification systems that operate in transport stations and other public environments where display boards update based on centrally generated information. It also reflects how smart devices like watches and home automation equipment show alerts that originate from cloud platforms. By replicating these ideas on a smaller scale the project allows learners to understand how real time updates are delivered to local displays through wireless communication.

The objective of this introduction is to outline the purpose and relevance of the project within the IoT domain. The system shows how cloud messaging, WiFi enabled modules, and microcontroller based displays can work together to present live information. The design remains simple enough for academic study while still representing the basic structure of modern notification and alert systems used in industry.

2. Proposed System

The proposed system focuses on creating a simple and effective method to display user generated messages on a physical LCD through a Telegram Bot interface. The user interacts with a Telegram Bot by sending any text message which serves as the primary input for the system. This message travels through the Telegram API where it is fetched by the ESP8266 WiFi module connected to the Arduino. The idea is to provide a smooth and direct communication path between a cloud based messaging platform and the hardware that handles the display. This approach demonstrates how IoT devices can respond instantly to remote commands without complex networking setups.

At the hardware level the Arduino acts as the central controller that receives the incoming message from the ESP8266. The ESP8266 is responsible for handling the WiFi connection and communicating with Telegram servers. Once the message arrives the Arduino processes it and sends the text to the 16x2 LCD module. The LCD then updates its screen and presents the message clearly to the user. This entire flow shows how cloud data can be translated into a local visual output through a lightweight IoT configuration.

The system architecture is designed to stay modular and easy to understand. The communication logic runs on the ESP8266 while the display logic remains on the Arduino which keeps the responsibilities well separated. The LCD acts as the final output device and gives the project a practical use case because the displayed information can change based on real time user commands. The components are connected through standard jumper wires and assembled on a breadboard which makes the setup flexible for further expansion.

The purpose of the proposed system is to demonstrate a reliable method for remote message display and to show how IoT hardware can perform real time notification tasks. This design reflects real applications including public information boards smart device alerts and automated messaging systems. By using Telegram as the communication platform the project ensures accessibility and ease of use which allows anyone to control the display from anywhere with an internet connection. This structure makes the system suitable for educational demonstrations and practical IoT learning.

3. System Design and Implementation

The implementation of the system integrates the ESP8266 Wi-Fi module, Telegram Bot API, Arduino Uno, and a 16×2 LCD into a single functional workflow. The ESP8266 is programmed to establish a secure Wi-Fi connection and continuously poll the Telegram servers for new messages. Once a message is received, the ESP8266 verifies the sender's chat ID for authorization, processes the text, and forwards the validated message to the Arduino through the serial communication interface. The Arduino receives the incoming data and formats the text according to the LCD's 16-character display limit, ensuring the message is properly parsed across one or two lines. The LCD is initialized through the I2C interface, allowing efficient communication with minimal wiring while maintaining a steady refresh rate.

The hardware setup is structured on a breadboard for modular testing. Power is distributed from the Arduino's 5V and GND pins to the LCD and common rails, while the ESP8266 receives power through its dedicated USB supply to ensure stable current for Wi-Fi operations. The system was tested iteratively by sending sample messages through the Telegram bot and validating text rendering on the LCD. Serial debugging was used to monitor packet reception, bot polling intervals, and string handling. The complete implementation demonstrates the interoperability between IoT messaging services and embedded hardware, resulting in a compact and real-time message display system suitable for automation, remote announcements, and educational IoT applications.

3.1 Components Used

Sr. No.	Component	Description
1	Arduino Uno	Acts as the main controller that receives processed text from the ESP8266 and displays it on the LCD. Handles serial

		communication and overall control flow.
2	ESP8266 NodeMCU WiFi Module	Connects to WiFi and communicates with the Telegram servers to fetch incoming messages. Forwards the messages to the Arduino for display.
3	16x2 LCD Display (I2C Interface)	Displays the received Telegram message on its two-line screen. Uses I2C communication to reduce wiring complexity
4	Jumper Wires (Male-to-Female)	Used to interconnect the LCD, ESP8266, and Arduino through the breadboard. Supports both signal and power wiring
5	Breadboard	Provides a non-soldering platform for arranging and routing jumper wire connections for the components.
6	Telegram Application	Serves as the interface through which the user sends messages to the Telegram Bot. Messages are retrieved by the ESP8266.
7	USB Cable	Supplies power to both the Arduino and ESP8266, and

		supports code upload and serial monitoring.
8	Arduino IDE	Used to write, compile, and upload the code to the ESP8266. Manages all required libraries and debugging tools.

Table.3.1: Description of each component

3.2 Circuit Diagram

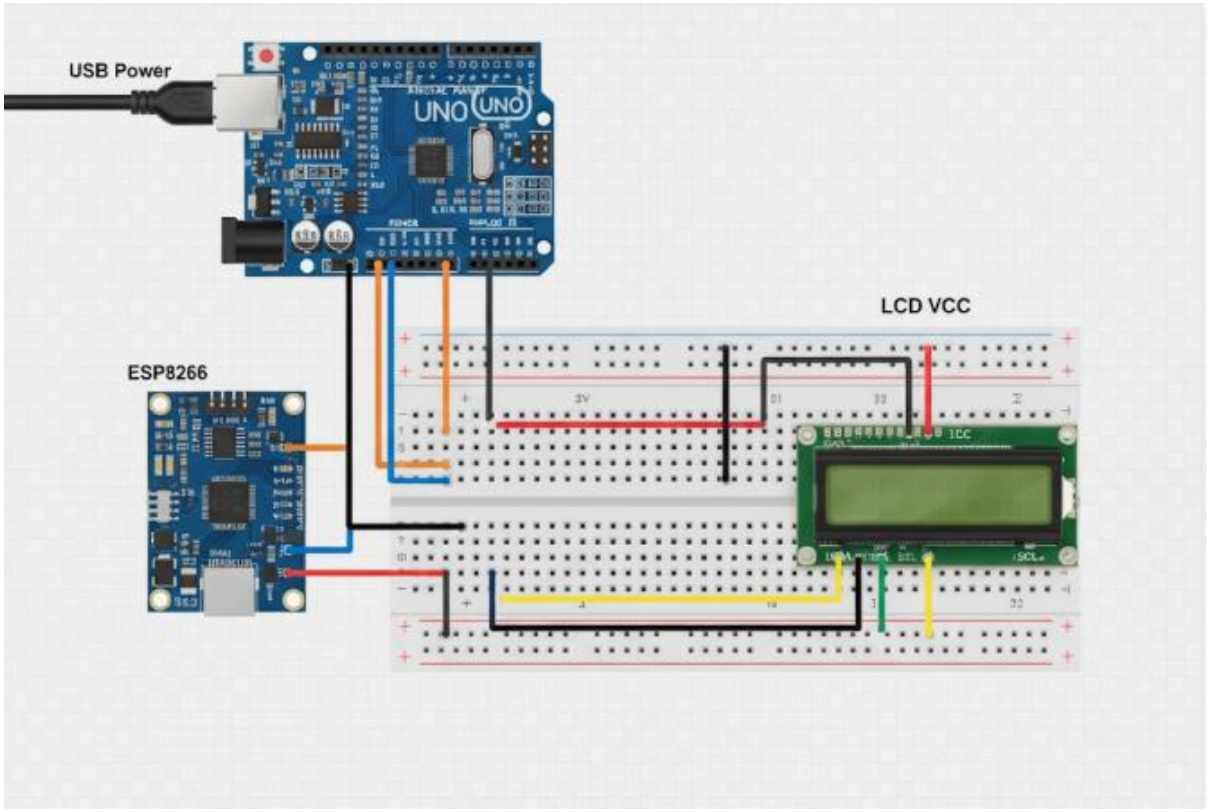

fig.3.2: Circuit Diagram

3.3 Connection

Arduino Uno Connections

1. **5V pin** connected to the breadboard positive rail to supply power to the LCD module.

2. **GND pin** connected to the breadboard ground rail to create a common ground for all components.
3. **TX (Pin 1)** connected to **ESP8266 RX** for sending formatted data to the ESP8266.
4. **RX (Pin 0)** connected to **ESP8266 TX** for receiving the incoming Telegram messages.
5. **Digital Pin 7** connected to **LCD RS** which controls the selection of command or data mode.
6. **Digital Pin 8** connected to **LCD EN** which enables the LCD to latch incoming data.
7. **Digital Pin 9** connected to **LCD D4** to support 4-bit data communication.
8. **Digital Pin 10** connected to **LCD D5** for the second data line in 4-bit mode.
9. **Digital Pin 11** connected to **LCD D6** for the third data line.
10. **Digital Pin 12** connected to **LCD D7** for the fourth data line.
11. **5V pin** connected to **LCD VCC** for providing display power.
12. **GND pin** connected to **LCD GND** to ground the LCD.
13. **Analog pin via potentiometer** connected to **LCD V0** to control the contrast of the display.

16x2 LCD Display Connections

1. **VCC pin** connected to the 5V rail to power the LCD.
2. **GND pin** connected to the common ground rail.
3. **V0 pin** connected to the potentiometer output to adjust screen contrast.
4. **RS pin** connected to **Arduino Digital Pin 7** for register selection.
5. **EN pin** connected to **Arduino Digital Pin 8** to enable the LCD.
6. **D4 pin** connected to **Arduino Digital Pin 9** for 4-bit data transmission.
7. **D5 pin** connected to **Arduino Digital Pin 10** for data line 2.
8. **D6 pin** connected to **Arduino Digital Pin 11** for data line 3.
9. **D7 pin** connected to **Arduino Digital Pin 12** for data line 4.
10. **LED+ pin** connected to 5V for backlight power.
11. **LED− pin** connected to GND for backlight grounding.

ESP8266 Connections

1. **TX pin** connected to **Arduino RX (Pin 0)** to send Telegram messages to the Arduino.
2. **RX pin** connected to **Arduino TX (Pin 1)** for receiving processed serial commands.

3. **EN pin** connected to **3.3V** to activate the ESP8266 module.
4. **RST pin** left unconnected or used for manual reset when required.
5. **VCC pin** connected to **3.3V or regulated supply** depending on the ESP8266 version.
6. **GND pin** connected to the common ground rail for stable operation

3.4 Code

```
#include <ESP8266WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

// ==== WiFi & Telegram Config ====
const char* ssid = "Galaxy A20s9257";
const char* password = "12341234";

#define BOT_TOKEN "8220531272:AAGv-S8l__6uxOaQkXIb1NLf7IsWgOPHnxA"
#define CHAT_ID   "1408487703"

// ==== LCD Config ====
LiquidCrystal_I2C lcd(0x27, 16, 2);
// SDA = D2 (GPIO4), SCL = D1 (GPIO5)

// ==== Telegram Setup ====
WiFiClientSecure client;
UniversalTelegramBot bot(BOT_TOKEN, client);

unsigned long lastTimeBotRan = 0;
int botRequestDelay = 1500;

void setup() {
  Serial.begin(115200);

  // VERY IMPORTANT FOR ESP8266 I2C:
  Wire.begin(D2, D1);    // SDA = D2, SCL = D1

  // LCD Init
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Starting...");

  // WiFi
  WiFi.begin(ssid, password);
  client.setInsecure();

  Serial.print("Connecting");
```

```
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  lcd.clear();
  lcd.print("WiFi Connected!");
  Serial.println("\nWiFi Connected!");
}

void handleNewMessages(int numNewMessages) {
  for (int i = 0; i < numNewMessages; i++) {

    String chat_id = bot.messages[i].chat_id;
    String text = bot.messages[i].text;

    Serial.println("Message received: " + text);

    if (chat_id != CHAT_ID) {
      bot.sendMessage(chat_id, " Unauthorized!", "");
      return;
    }
    lcd.clear();
    lcd.setCursor(0, 0);

    if (text.length() <= 16) {
      lcd.print(text);
    } else {
      lcd.print(text.substring(0, 16));
      lcd.setCursor(0, 1);
      lcd.print(text.substring(16, 32));
    }
    bot.sendMessage(chat_id, " Text displayed:\n" + text, "");
  }
}
void loop() {
  if (millis() - lastTimeBotRan > botRequestDelay) {

    int newMessages = bot.getUpdates(bot.last_message_received + 1);

    while (newMessages) {
      handleNewMessages(newMessages);
      newMessages = bot.getUpdates(bot.last_message_received + 1);
    }

    lastTimeBotRan = millis();
  }
}
```

4. Results and Output

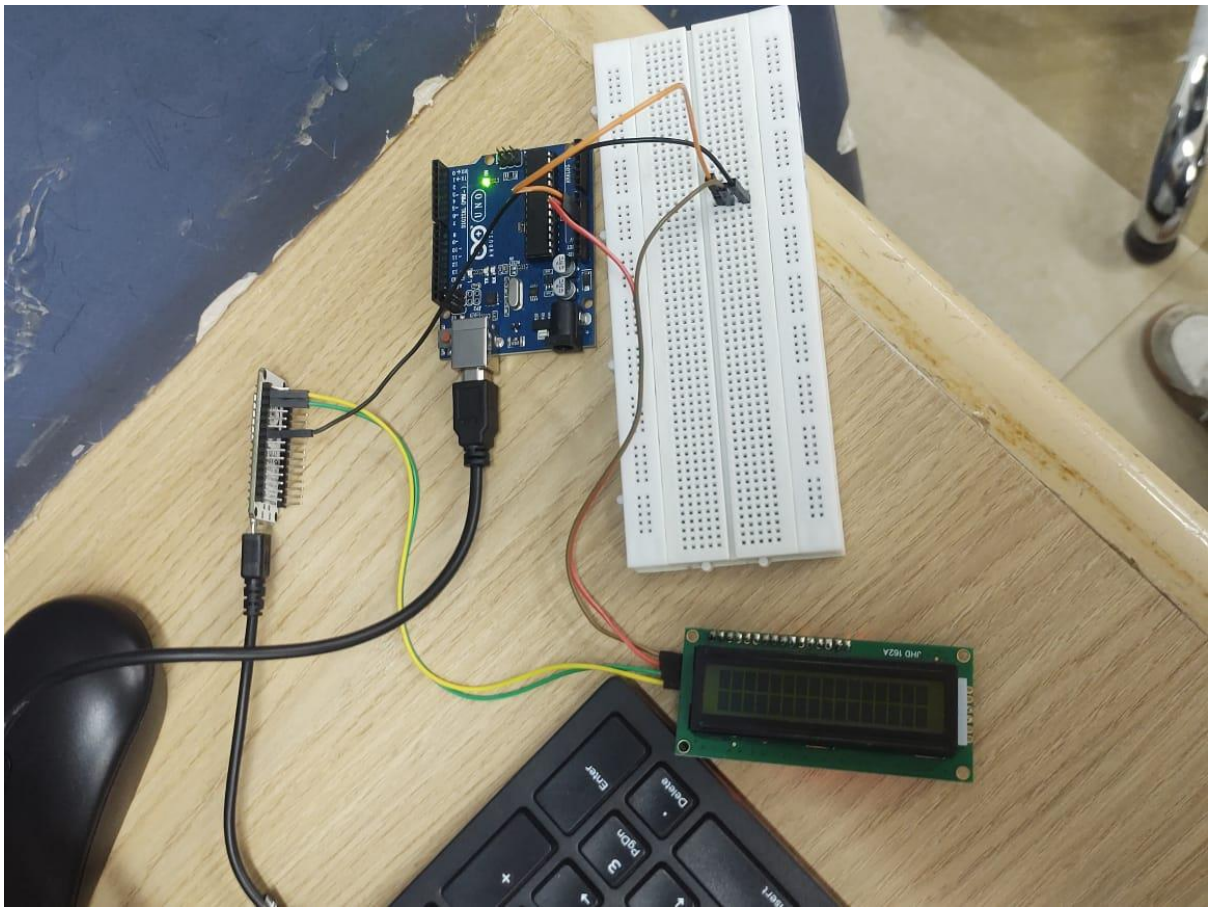

Fig:4.1. Actual Connection Setup

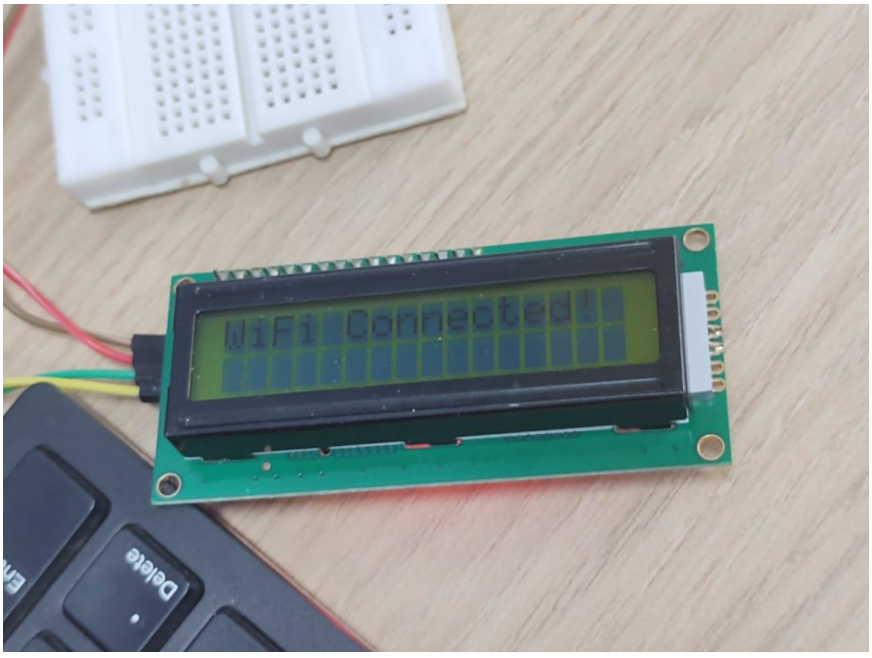

Fig:4.2. Wi-Fi Module Connection Established

Fig:4.3. Sending Message via Telegram Bot

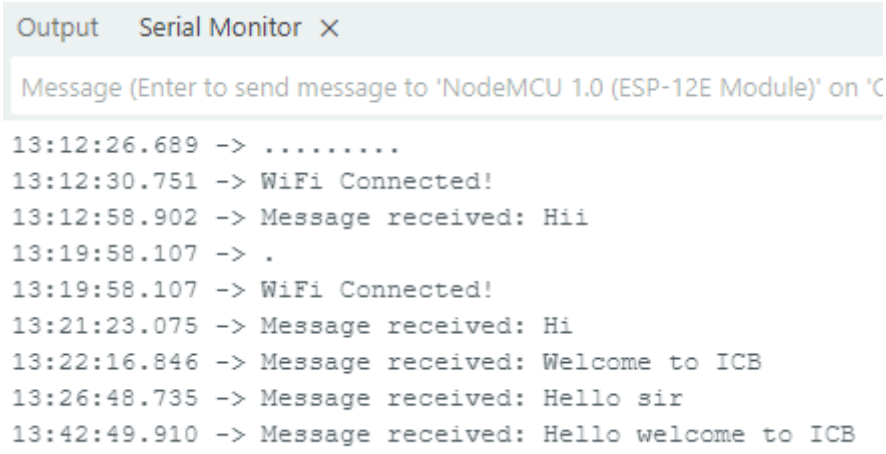

Fig:4.4. Message Displayed on the Serial Monitor

Fig:4.5. Message Displayed on the LCD Display

4.1 Video Demonstration

click here to see the project demonstration

5. Applications and Use Cases

This system can be used in real time notification environments where information must be displayed instantly without delay. One of the most direct applications is in small scale announcement systems such as classrooms, laboratories, coaching centers, or office workspaces. Staff can send updates, reminders, or alerts through the Telegram bot and the message appears immediately on the LCD display. This removes the need for manual boards and provides a quick way to broadcast short messages.

The project also fits well into public information displays where remote updates are required. Railway stations, bus depots, and small terminals often need compact scrolling or fixed display units. Using this setup the operator can send important travel information, delay messages, or emergency alerts directly from a Telegram interface. This reduces manual intervention and increases the speed at which information reaches the public.

Another strong use case is in smart home and personal automation systems. Users can send text reminders to the display placed in the living room, kitchen, or workspace. These messages can include daily tasks, medicine reminders, or family notifications. The same concept supports wearable systems where short alerts are delivered to compact screens similar to smart watch notifications. The project provides the basic logic behind how these notification mechanisms operate.

The system also has applications in industry and IoT based monitoring setups. Supervisors can send short operational instructions to field workers or display status updates on small screens near machinery. Since the communication relies on Telegram the system works from anywhere with internet access. This makes it suitable for factory floors, warehouses, and remote monitoring stations. The use of Arduino and ESP8266 makes the solution cost effective and suitable for rapid prototyping in industrial IoT projects.

6. Advantages and Limitations

6.1 Advantages

The project offers several practical advantages that make it suitable for small scale communication and IoT based notification tasks.

• Real time message delivery through Telegram which ensures quick updates.

• Low cost implementation since Arduino and ESP8266 are affordable.

• Simple wiring and programming which makes the system easy to build and maintain.

• Remote accessibility that allows users to send messages from any location.

The system also helps demonstrate key IoT communication concepts in a clear manner.

Additional benefits:

• Direct integration with existing WiFi networks without extra hardware.

• Can be expanded with sensors or automation modules.

• Supports LCD based visual feedback which improves clarity for the user.

• Uses a secure bot service which controls who can push updates.

6.2 Limitations

Even though the system works well for demonstration and small scale use it has some limitations.

• Dependence on stable WiFi for message reception.

• Limited display size of the 16x2 LCD which cannot show long messages at once.

• ESP8266 can lag when processing many requests rapidly.

• Delay caused by Telegram API rate limits.

There are also some hardware and design constraints that can affect reliability.

Further limitations:

• No onboard battery support which makes it dependent on wired power.

• LCD visibility reduces under bright sunlight.

• Serial communication between Arduino and ESP8266 may face noise issues.

• Not suitable for large display systems or high traffic environments.

7. Costing

.

Sr. No.	Component Name	Quantity	Unit Cost (INR)
1	Arduino Uno	1	₹335
2	ESP8266 WiFi Module	1	₹320
3	16x2 LCD Display	1	₹218
4	Jumper Wires (Male to Female)	1 set	₹80
5	Breadboard	1	₹150
6	Telegram (Software)	1	Free
7	USB Cable	1	₹120
8	Arduino IDE (Software)	1	Free

Table 7.1: Costing Table

The cost estimation for the components used in the project **"Telegram Bot Message Display Using Arduino and ESP8266"** is presented above. These components represent the essential hardware required for communication, processing, and message display. The total estimated cost of the prototype remains affordable, making it suitable for academic projects and IoT demonstrations. The combination of Arduino and ESP8266 delivers reliable communication with minimal investment, while the use of free software tools keeps the overall expense low.

8. Conclusion

The project successfully demonstrates how IoT communication can be implemented through a simple Telegram-controlled LCD display using Arduino and the ESP8266 WiFi module. The system shows how a wireless notification setup can receive real time messages over the internet and display them instantly on the LCD. This reflects the core idea behind smart information systems used in public spaces and personal devices.

The implementation highlights how easily available components can be combined to create a functional communication model. The use of Telegram as the messaging interface makes the system secure and accessible while the ESP8266 handles the WiFi communication with steady performance. The Arduino manages the display operations and ensures that the received text is shown correctly on the 16x2 LCD which makes the overall workflow smooth and dependable.

This mini project also builds a foundation for further extensions such as integrating sensors, adding automation triggers, or shifting to larger displays. It offers a practical demonstration of IoT concepts that can be adapted for real world applications like public information boards, personal notification devices and alert systems. The successful working of this prototype confirms that simple hardware and open source tools can deliver meaningful IoT solutions.