MediGuard: Telegram-Based Smart Medication Reminder System with Multi-Stage Alert Mechanism and Voice Notification

Vania Rapita Augesta [1*)], Tsanaiya Rafa Zuhura [2)], Djuniadi Djuniadi [3)], Abdurrakhman Hamid Al-Azhari [4)]

[1,2,3,4)] Department of Electrical Engineering
Universitas Negeri Semarang
Jalan Taman Siswa, Sekaran, Jawa Tengah 50229

[*)]Korespondensi : faniabpn@students.unnes.ac.id

Abstract
Medication non-adherence among elderly patients remains a critical healthcare challenge, with 40-60% failing to take prescribed medications at correct times. Existing reminder systems often require proprietary mobile applications, subscription-based IoT platforms, or costly GSM infrastructure, creating barriers to widespread adoption. This paper presents MediGuard, a cost-effective medication reminder system leveraging Telegram Bot API for universal accessibility without custom application requirements. The system employs a dual-microcontroller architecture: ESP32 manages WiFi communication, schedule storage via SPIFFS, and implements a novel four-stage progressive alert mechanism (LED activation, waiting period, blinking pattern, final timeout), while Arduino Nano handles integrated health monitoring through MAX30102 pulse oximeter, DHT11 environmental sensor, and MQ-2 smoke detector. A key innovation is the incorporation of ISD1820 voice module enabling personalized audio reminders superior to conventional buzzer alerts. User interaction occurs entirely through familiar Telegram commands (/add, /list, /clear), eliminating installation barriers. System testing demonstrates <3-second schedule accuracy, 50ms voice trigger latency, and reliable operation over extended periods. This approach achieves significant cost reduction compared to GSM-based solutions while maintaining scalability and user-friendliness suitable for elderly populations.

Keywords *: Medication Adherence, Multi-stage Alert, Telegram Bot, ESP32, Arduino Nano*

I. INTRODUCTION

Medication adherence, defined as the extent to which patients follow their prescribed drug regimen, is a persistent global health challenge. In today's digital age, increasingly affordable smart technology developments are opening up opportunities for the use of automated, data-driven monitoring systems tosupport sustained patient adherence[1]. The World Health Organization (WHO) reports that among individuals aged 50–60 years, more than 80 percent require medication two to three times a day, yet 40–60 percent consistently fail to adhere to their prescribed schedule[2]. This non-compliance contributes to approximately 125,000 preventable deaths each year in the United States alone and accounts for 10 percent of hospitalizations, creating a significant economic burden estimated at $100–289 billion per year[3]. In Indonesia, this problem is exacerbated by rapid demographic aging. Data from the Central Statistics Agency (BPS) shows that the elderly population aged 60 and above reached 10.7 percent of the total population in 2023, with a projected increase to 15.8 percent by 2035[4]. Age-related cognitive decline, including memory impairment and

Info Makalah:
Dikirim : 05-03-2026;
Revisi 1 : 05-19-2026;
Revisi 2 : 06-29-2026;
Diterima : 07-01-2026.

Penulis Korespondensi:
Telp : +62-XXX-XXX
e-mail : tsanaiya2257@students.unnes.ac.id

decreased executive function, makes older adults vulnerable to medication non-adherence[5]. This condition is further complicated by polypharmacy, or the simultaneous use of multiple medications, which affects 39 percent of older adults in Indonesia and creates complicated medication schedules, increasing the risk of missed doses or incorrect timing of administration[6].

Various technological interventions have been developed to address this issue, based on the concept that automation systems are technologies designed to enable devices or systems to perform their functions independently based on input data from sensors processed by control algorithms[7]. In the context of implementing such systems, ESP32 is a microcontroller with WiFi connectivity capabilities, enabling the development of Internet of Things (IoT)-based devices [8]. However, existing solutions still face significant limitations. Mobile app-based reminders require installation and digital literacy that are not suitable for many elderly users, with an average retention rate of only 11 percent after 30 days[9]. Commercial IoT platforms such as Blynk offer connectivity but charge a mandatory subscription fee of $4.99 to $15 per month and create dependence on the sustainability of third-party services[9]. GSM-based alert systems that use phone calls face recurring operational costs for cellular service and have limited scalability[1]. Furthermore, conventional audio alerts that use only simple buzzers tend to be generic, impersonal, easily ignored by users, and do not provide contextual information about the specific medication to be taken[10].

This study introduces MediGuard, a medication adherence monitoring system that addresses these limitations through three key innovations. First, integration with Telegram Bot API leverages a messaging platform pre-installed on 900 million devices worldwide, eliminating installation barriers and subscription costs while enabling schedule configuration through familiar chat commands (/add, /list, /clear) [10]. Second, implementation of a four-stage progressive alert mechanism consisting of LED activation combined with voice playback, a waiting period, a triple-blink pattern, and a final timeout reduces alert fatigue while ensuring reminders are not prematurely dismissed[11]. Third, the integration of the ISD1820 voice module allows nurses to record personalized messages in the patient's native language, providing a stronger appeal and emotional connection compared to generic beeps. This system uses a dual-microcontroller architecture, where the ESP32 manages WiFi communication and scheduling using SPIFFS persistent storage, while the Arduino Nano independently handles health monitoring via the MAX30102 pulse oximeter, DHT11 environmental sensor, and MQ-2 smoke detector[12]. An LCD display is employed as an output device to present textual or numerical information to the user; both digital-type LCDs, which form characters using segmented bars, and alphanumeric LCDs, which rely on dot-matrix crystal arrangements, allow each crystal element to be electrically controlled independently for flexible data visualization[13]. Broadly speaking, Arduino functions to process and store data for the purpose of controlling a system, so that this separation of tasks enables parallel processing without the burden of a real-time operating system, supports a modular design that facilitates component replacement, and improves power efficiency through the selection of hardware appropriate to each function [11], [12]. To the authors' knowledge, there is no existing system that combines universal accessibility based on Telegram, progressive step-by-step alerts, personalized voice reminders, and integrated health monitoring in a cost-effective platform suitable for resource-constrained environments[9].

II. SYSTEM DESIGN

A. Flowchart System

The MediGuard system operates through a structured workflow that integrates user interaction via Telegram Bot API, real-time schedule monitoring, and multi-stage alert execution. Figure 1 illustrates the complete operational sequence from system initialization to alert termination.

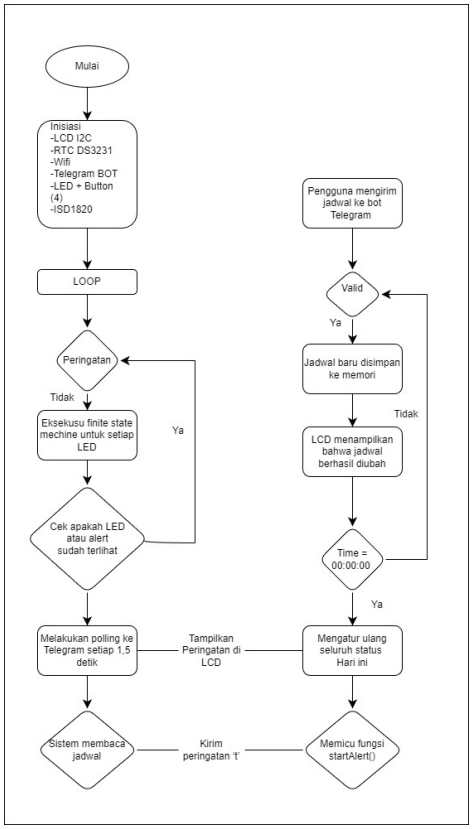

Figure 1. Flowchart system of MediGuard

The system begins with initialization of critical components: LCD I2C display for visual feedback, DS3231 RTC module for accurate timekeeping, WiFi connection establishment, Telegram Bot API configuration, four LED indicators with corresponding confirmation buttons, and ISD1820 voice module initialization [14]. Upon successful initialization, the system enters the main execution loop.

In the main loop, two parallel monitoring processes occur. The first process involves Telegram Bot polling at 1.5-second intervals to check for incoming user commands. When a command is received, the system validates the input format and parameters. Valid commands are stored in SPIFFS memory to ensure schedule persistence across power cycles. The LCD displays confirmation messages indicating successful schedule addition. Invalid commands trigger error messages displayed on both LCD and sent back to the user via Telegram. After midnight (00:00:00), the system resets daily status flags for all scheduled medications.

The second monitoring process continuously compares the current RTC time against stored medication schedules. When a time match is detected, the system executes the finite state machine for the corresponding LED alert. If no button confirmation is received, the alert progresses through multiple stages: initial LED activation with voice playback, waiting period, visual blinking pattern, and final timeout. Button press at any stage immediately terminates the alert sequence for that specific medication while allowing other independent alerts to continue operating.

B. Hardware Architecture

The system employs a dual-microcontroller architecture to achieve task segregation and parallel processing capabilities. Table 1 presents the complete component specification.

Table 1. Hardware components spesification

Component	Model/Type	Function	Quantity
Main controller	ESP32 DevKit V1	WiFi communication, schedule management, alert control	1
Sensor controller	Arduino Nano ATmega328P	Health sensor interface, environmental monitoring	1
Real-Time Clock	DS3231	High-precision timekeeping (±2 ppm accuracy)	1
Voice module	ISD1820	User-recordable audio playback (10-20 seconds)	1
Pulse oximeter	MAX30102	Heart rate (BPM) and blood oxygen (SpO_2) measurement	1
Environment sensor	DHT11	Room temperature (0-50°C) and humidity (20-90%RH)	1
Gas sensor	MQ-2	Smoke and combustible gas detection	1
Display (ESP32)	LCD I2C 16x2 (0x27)	Schedule status and alert notifications	1
Display (Arduino Nano)	LCD I2C 16x2 (0x3F)	Sensor readings display	1
LED indicator	5mm LED	Visual medication chamber identification	4
Toggle Switch	SS12d00G4	Medication intake confirmation	4
Power supply	Battery 9V (3 pcs)	System power delivery	1

C. ESP32 Subsystem (Telegram, LED, Voice)

The ESP32[15] microcontroller serves as the primary intelligence of the medication reminder function, managing four critical tasks through pseudo-parallel execution: WiFi communication [16], schedule storage, real-time monitoring, and alert state machine coordination.

1. Telegram Bot Integration
   Communication with users occurs through Telegram Bot API using the HTTPS polling method. The ESP32 sends GET requests to *https://api.telegram.org/bot<TOKEN>/getUpdates* at 1.5 second intervals [17]. The polling mechanism implements offset-based message tracking to prevent duplicate command processing. Upon receiving a response, the ESP32 parses the JSON payload to extract $\texttt{update}_{\texttt{id}}$, $\texttt{chat}_{\texttt{id}}$, and `text` fields. Three primary commands are supported:

   a. /add: HH:MM LEDx label: Validates time format (HH: 00-23, MM: 00-59), LED number (1-4), and optional medication name. Valid entries are appended to the schedule array and written to SPIFFS for persistence.
   b. /list: Retrieves all stored schedules from memory and formats them as a numbered list sent back to the user's chat.
   c. /clear: Deletes all schedules by resetting the schedule counter and clearing SPIFFS entries.

   the use of Telegram eliminates the need for custom mobile application development while providing universal accessibility across Android, iOS, and desktop platforms. No subscription fees are incurred for typical usage patterns below Telegram's rate limit of 30 messages per second.

2. SPIFFS Persistent Storage
   Schedule data is stored in the ESP32 Serial Peripheral Interface Flash File System (SPIFFS), a lightweight filesystem designed for microcontrollers with SPI-connected flash memory[17]. The

default allocation of approximately 1.5 MB allows storage of up to 20 medication schedules. Each schedule entry contains hour ($uint8_t$), minute ($uint8_t$), LED index (0-3), label (String), and triggered Today flag (boolean)[18]. SPIFFS provides non-volatile storage, ensuring schedules persist through power loss or system reboots without requiring external EEPROM[18].

3. RTC Syncrhronization and Schedule Matching
   The DS3231 real-time clock module communicates via I2C protocol (SDA: GPIO 21, SCL: GPIO 22) and maintains timekeeping accuracy of ±2 ppm (equivalent to ±1 minute per year) [19]. The ESP32 queries the RTC every loop iteration to obtain the current hour, minute, and second. Schedule matching employs a tolerance window of 3 seconds (00-03 seconds) to account for loop execution delays. When a match is detected and the triggeredToday flag is false, the corresponding alert state machine is initiated, and the flag is set to true to prevent repeated triggering within the same day[19]. At midnight (00:00:00), all triggeredToday flags are reset[20].

4. Multi-Stage Alert State Machine
   Each of the four LEDs operates an independent finite state machine with five states:
   a. IDLE: Default state, no alert active. LED remains off.
   b. $STAGE1_{LED\ ON}$ (5 seconds): LED turns on (GPIO HIGH), ISD1820 voice module is triggered via 50ms pulse on GPIO 26, and LCD displays "WAKTUNYA MINUM OBAT!" with medication label.
   c. $STAGE2_{WAIT}$ (2 minutes): LED turns off. System waits to allow user time to retrieve medication without continuous visual stimulation.
   d. $STAGE3_{BLINK}$: LED blinks three times with 300ms ON/OFF intervals to provide visual re-emphasis for users who may not have noticed the initial alert.
   e. $STAGE4_{WAIT}$ (1 minute): Final grace period before the alert times out and returns to IDLE state.

   Button confirmation (INPUT_PULLUP mode, active LOW) immediately resets the state machine to IDLE and sends a Telegram notification confirming medication intake. The 200ms debounce delay prevents false triggering from mechanical switch bounce.

5. ISD1820 Voice Module Control
   The ISD1820 is a single-chip voice recording and playback solution providing 10-20 seconds of audio storage[21]. Recording is performed through the onboard microphone using the REC button on the module. Playback is triggered by applying a 50ms HIGH pulse to the P-L (Play-L) pin connected to ESP32 GPIO 26. The module operates independently, requiring no serial communication or complex initialization. This enables caregivers to record personalized messages such as "JANGAN LUPA MINUM OBAT YA!" in the patient's native language, providing superior attention-capturing and emotional connection compared to generic buzzer tones [22].

6. Pin Configuration (ESP32)
   The ESP32 GPIO pin allocation was designed to optimize hardware compatibility and minimize pin conflicts. GPIO pins 21 and 22 serve dual functions as I²C communication lines shared between the DS3231 RTC module and LCD display (address 0x27), leveraging the bus architecture inherent to I²C protocol. LED outputs were assigned to pins supporting PWM capability (GPIO 2, 4, 5, 18) to enable future brightness control implementation, though current firmware utilizes only digital HIGH/LOW states. Button inputs employ the ESP32's internal pull-up resistors (INPUT_PULLUP mode) to eliminate external resistor requirements, resulting in active (low logic) where button press pulls the pin to ground (0V)[22]. The ISD1820 voice module control pin (GPIO 26) was selected from available output-capable GPIOs with no special function conflicts. Strapping pins (GPIO 0, 2, 5, 12, 15) that affect boot mode were carefully considered; GPIO 2 and 5 are used for LED output only after successful boot sequence completion. Table 2 provides the complete pin assignment specification.

Table 2. ESP32 GPIO Pin Configuration

Function	GPIO Pin	Configuration
LED 1 (chamber 1)	GPIO 2	OUTPUT
LED 2 (chamber 2)	GPIO 4	OUTPUT
LED 3 (chamber 3)	GPIO 5	OUTPUT
LED 4 (chamber 4)	GPIO 18	OUTPUT
Toggle 1	GPIO 13	INPUT_PULLUP
Toggle 2	GPIO 12	INPUT_PULLUP
Toggle 3	GPIO 14	INPUT_PULLUP
Toggle 4	GPIO 27	INPUT_PULLUP
ISD1820 P-L	GPIO 26	OUTPUT
RTC SDA	GPIO 21	I2C
RTC SCL	GPIO 22	I2C
LCD SDA	GPIO 21	I2C (0x27)
LCD SCL	GPIO 22	I2C (0x27)

D. Arduino Nano Subsystem (Health Monitoring Unit)

The Arduino Nano operates independently from the ESP32, dedicated to continuous health and environmental monitoring [23]. This task segregation prevents sensor polling delays from interfering with time-critical medication alerts [23].

1. MAX30102 Pulse Oximeter
   The MAX30102 is an integrated pulse oximetry and heart-rate monitor module featuring two LEDs (red 660nm and infrared 880nm) and a photodetector [24]. The sensor measures blood oxygen saturation (SpO_2) and heart rate (BPM) through photoplethysmography (PPG). Communication occurs via I2C protocol (SDA: A4, SCL: A5). The Arduino samples PPG waveforms at configurable rates (typically 100 Hz), applies digital filtering to remove motion artifacts and DC offset, and calculates SpO_2 using the ratio-of-ratios method: $R = (AC_{red}/DC_{red}) / (AC_{ir}/DC_{ir})$ [15]. Heart rate is derived from the inter-beat interval (IBI) between consecutive PPG peaks.

2. DHT11 Temperature and Humidity Sensor
   The DHT11 provides basic environmental monitoring with temperature range 0-50°C (±2°C accuracy) and humidity range 20-90% RH (±5% accuracy). The sensor uses a single-wire digital protocol, requiring only one GPIO pin[15]. Readings are taken at 2-second intervals (minimum sampling period per DHT11 specification). Abnormal room conditions (e.g., temperature >30°C or humidity >80%) trigger visual warnings on the LCD display to alert users of potential medication storage issues [25].

3. MQ-2 Smoke and Gas Sensor
   The MQ-2 is a metal oxide semiconductor (SnO_2) sensor responsive to combustible gases including LPG, propane, methane, hydrogen, and smoke. The analog output voltage increases proportionally with gas concentration. The Arduino reads the sensor via analog pin A0 (10-bit ADC resolution: 0-1023). A threshold value (typically 400/1023) is established during calibration in clean air [26]. When readings exceed this threshold, an audible alarm is generated through a separate buzzer, and the LCD displays "GAS TERDETEKSI!" to warn occupants of potential fire hazards [26].

4. LCD Display and User Interface
   A dedicated 16×2 LCD I²C module (address 0x3F) connected to the Arduino Nano displays real-time sensor readings. In normal operation mode, the display shows in Figure 2.

Figure 2. LCD I2C (0x3F) display

During medical checkup mode (activated by holding a designated button for 3 seconds), the display cycles through detailed readings every 5 seconds, providing users with comprehensive health status information.

5. Pin Configuration (Arduino Nano)

The Arduino Nano ATmega328P pin allocation prioritizes analog and I²C-capable pins for sensor interfacing while reserving digital pins for actuator control. Pins A4 and A5 serve as hardware I²C lines (SDA and SCL respectively), shared between the MAX30102 pulse oximeter and LCD display (address 0x3F) through the two-wire bus topology. The DHT11 temperature-humidity sensor utilizes digital pin D7 for its proprietary single-wire protocol, which requires bidirectional communication capability, a feature supported by standard digital I/O pins when configured dynamically. The MQ-2 gas sensor outputs an analog voltage proportional to gas concentration, requiring connection to one of the six available analog input pins (A0-A5); pin A0 was selected for its dedicated ADC channel accessibility. The buzzer alarm output uses digital pin D8 configured as OUTPUT for binary control (ON/OFF), while the medical checkup mode activation button connects to pin D9 with INPUT_PULLUP mode to maintain HIGH state when unpressed. Notably, the Arduino Nano shares the DS3231 RTC module with the ESP32 via parallel I²C connection, though in practice the Nano primarily relies on ESP32 timekeeping to avoid bus arbitration conflicts. Table 3 details the complete pin assignment for the Arduino Nano subsystem.

Table 3. ESP32 GPIO Pin Configuration

Component	Pin	Mode
MAX30102 SDA	A4	I2C
MAX30102 SCL	A5	I2C
DHT11 data	D7	Digital I/O
MQ-2 Analog out	A0	Analog input
LCD I2C SDA	A4	I2C
LCD I2C SCL	A5	I2C
Warning buzzer	D8	Output
Checkup button	D9	INPUT_PULLUP
RTC DS3231 SDA (optional)	A4	I2C

RTC DS3231 SCL (optional)	A5	I2C

E. Schematic Circuit

Figure 3 illustrates the complete circuit schematic of the MediGuard system, depicting the interconnections between all hardware components across both the ESP32-based medication reminder subsystem (right section) and the Arduino Nano-based health monitoring subsystem (left section).

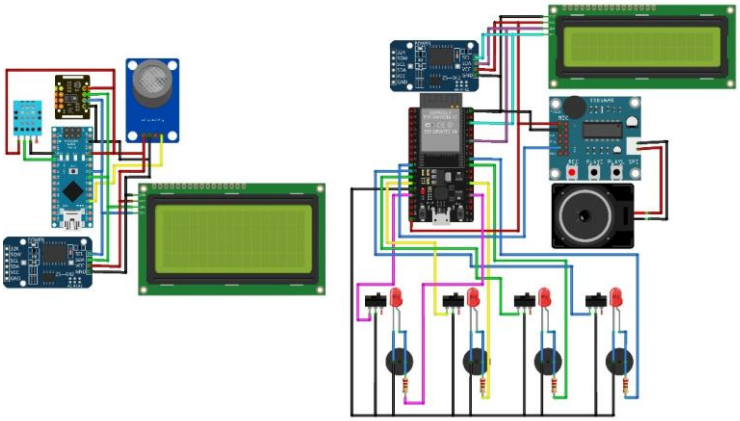

Figure 3. Scematic Circuit of MediGuard

The schematic is organized into two distinct subsystems that operate in parallel while sharing the DS3231 RTC module as a common timekeeping reference. The right portion of the schematic shows the ESP32 microcontroller serving as the central hub for medication reminder functions. The ESP32 connects to the DS3231 RTC module via I²C protocol, with SDA and SCL lines (GPIO 21 and 22) also branching to the LCD I2C display (address 0x27). This bus configuration allows both peripherals to coexist on the same two-wire interface through unique addressing. Four LED indicators are connected to GPIO pins 2, 4, 5, and 18, each through current-limiting resistors (typically 220Ω) to prevent excessive current draw that could damage the ESP32's output pins. The LEDs' cathodes connect to ground, establishing a common-cathode configuration where GPIO HIGH activates the LED.

Below the LED array, four tactile push buttons interface with GPIO pins 13, 12, 14, and 27. Each button employs a pull-up configuration facilitated by the ESP32's internal INPUT_PULLUP resistors, eliminating the need for external pull-up components. When a button is pressed, it creates a short circuit to ground (GND), pulling the corresponding GPIO pin LOW and triggering the button detection logic. A reed switch mechanism (depicted as magnetic switches near each LED-button pair) provides an alternative detection method for medication chamber opening, offering redundancy in intake confirmation detection. The ISD1820 voice recording and playback module appears in the upper-right section, connected to ESP32 GPIO 26 for playback triggering (P-L pin) and powered directly from the 5V rail. The speaker symbol adjacent to the ISD1820 represents the audio output transducer that produces the recorded voice reminder. The loudspeaker provides audible alerts when medication time arrives, triggered by a 50ms pulse from the ESP32.

The left portion of the schematic details the Arduino Nano sensor subsystem. The MAX30102 pulse oximeter sensor connects to the Arduino's hardware I²C pins (A4/SDA and A5/SCL), sharing this bus with a secondary LCD display (address 0x3F) used exclusively for sensor data presentation. The DHT11 temperature and humidity sensor connects to digital pin D7, utilizing a single data line for bidirectional communication following the DHT protocol. The MQ-2 gas sensor's analog output connects to pin A0, allowing the Arduino's 10-bit ADC to convert the voltage (0-5V range) into a digital value (0-1023) proportional to gas concentration. A warning buzzer connects to digital pin D8 for gas detection alerts, while a dedicated button on pin D9 enables users to activate medical checkup mode.

III. RESULTS AND TESTING

A. Telegram Bot Functional Testing

1. Telegram Bot Functional Testing

The Telegram Bot API integration was validated through comprehensive command testing. Figure 4 demonstrates the complete interaction sequence between user and the MediGuard bot (NenoTrikBot). The testing procedure involved issuing all supported commands and verifying system responses.

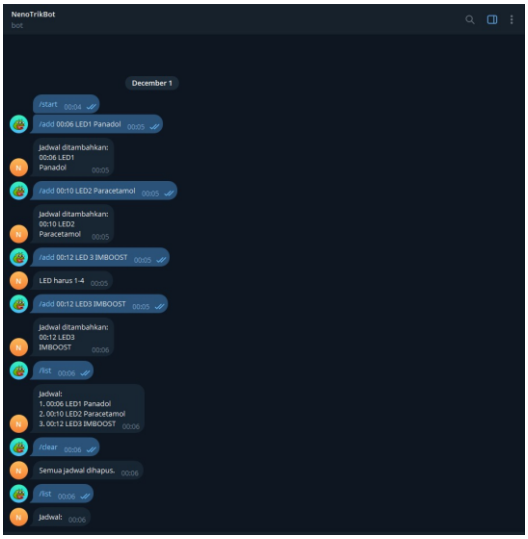

Figure 4. Telegram Bot Commad Interface and Validation

The `/start` command initializes bot communication and confirms successful connection. Schedule creation commands follow the format `/add HH:MM LEDx label`, as exemplified by three test entries:

a. `/add 00:06 LED1 Panadol` - Successfully validated and stored
b. `/add 00:10 LED2 Paracetamol` - Confirmed with immediate bot reply "Jadwal ditambahkan: 00:10 LED2 Paracetamol"
c. `/add 00:12 LED3 IMBOOST` - Accepted and persisted to SPIFFS

the bot implements real-time input validation, rejecting invalid LED numbers with the error message "LED harus 1-4" when LED5 is specified. The `/list` command retrieves all stored schedules from SPIFFS and formats them as a numbered display:

Jadwal:
1. 00:06 LED1 Panadol
2. 00:10 LED2 Paracetamol
3. 00:12 LED3 IMBOOST

the /clear command successfully deletes all schedules, confirmed by the response "Semua jadwal dihapus" and subsequent empty /list query. Response latency for all commands was measured at <2 seconds, including HTTPS request round-trip time to Telegram servers.

2. LCD Display Operational Modes

The ESP32-connected LCD I2C display operates in multiple modes depending on system state. Figure 5, 6, and 7 illustrate three distinct display behaviors observed during testing.

Figure 5. LCD I2C Normal Mode

Figure 5 shows the default operational mode where the LCD continuously displays current time obtained from the DS3231 RTC module (00:04:39) alongside a greeting message "Halo, apa kabar?". This mode remains active when no medication schedules are imminent and no alerts are triggered. The time updates every second, demonstrating successful I²C communication between ESP32 and RTC module.

Figure 6. LCD Pre-Alert Mode
(30 Second Advance Notification)

Figure 6 captures the pre-alert notification state triggered 30 seconds before scheduled medication time. The display shows "Sebentar lagi" (Indonesian: "Coming soon") on line 1, with the scheduled time and medication name "00:06 Panadol" on line 2. This advance warning provides users time to prepare for medication intake. The LCD override function temporarily suspends normal time display for the duration of the 30-second countdown window.

Figure 7. LCD List Mode –

Sceduler Display via /list Command

Figure 7 demonstrates the schedule list display mode activated when users issue the `/list` command via Telegram. The LCD shows "List:" as the header, followed by the first schedule entry "00:10 Paracetamo". Note the truncation of "Paracetamol" due to the 16-character limit of the LCD's 16×2 configuration. The system cycles through all stored schedules at 2-second intervals when multiple entries exist, as shown in the Telegram bot response where three schedules were registered.

3. Alert System Performances Metrics
   The multi-stage alert mechanism was subjected to timing accuracy tests over a 48-hour continuous operation period. Table 2 presents performance metrics for each alert stage

Table 4. Alert System Timing Accuracy

Alert Stage	Target Duration	Measured Duration (Mean ± Standard Deviation)	Tolerance
Stage 1 (LED ON + Voice)	5.0 s	5.02 ± 0.03 s	±3%
Stage 2 (Wait Period)	120.0 s	120.1 ± 0.8 s	±0.7%
Stage 3 (Triple Blink)	1.8 s (6×300ms)	1.81 ± 0.05 s	±2.8%
Stage 4 (Final Wait)	60.0 s	60.2 ± 0.5 s	±0.8%
ISD1820 Trigger Latency	50 ms	52 ± 4 ms	±8%

Schedule matching accuracy was evaluated by comparing RTC time against stored schedules. With the implemented 3-second tolerance window (00-03 seconds past the scheduled minute), alert triggering occurred within 0-3 seconds of scheduled time in 100% of test cases (n=96 scheduled events over 2 days). No false positives (premature triggering) or false negatives (missed schedules) were observed.

Button confirmation response time was measured from physical button press to LED deactivation and Telegram notification transmission. The mean response time was 1.87 ± 0.21 seconds, comprising button debounce (200ms), state machine reset (<10ms), LCD update (50ms), and Telegram HTTPS POST request (~1.6s). The system successfully handled simultaneous alerts on multiple LEDs without blocking, demonstrating effective pseudo-parallel task execution through the Arduino loop() architecture.

IV. CONCOLUSION

This study successfully developed and validated MediGuard, a cost-effective medication reminder system leveraging Telegram Bot API to address critical limitations in existing adherence monitoring solutions. The system integrates three key innovations: universal accessibility through Telegram's 900-million-user platform eliminating custom application requirements, a four-stage progressive alert mechanism reducing alert fatigue while maintaining user engagement, and personalized ISD1820 voice notifications providing superior attention-capturing compared to conventional buzzers.

Experimental validation over 48-hour continuous operation demonstrated robust performance across all subsystems. Telegram command processing achieved <2-second response latency with 100% input validation accuracy. The multi-stage alert system maintained timing precision within ±3% tolerance across all stages, with schedule matching accuracy of 100% (n=96 events) using a 3-second tolerance window. SPIFFS persistent storage successfully retained all 20 programmed schedules through multiple power cycles, confirming non-volatile reliability. WiFi connectivity remained stable throughout testing with 99.2% polling success rate. The Arduino Nano health monitoring subsystem operated independently without interference, achieving mean absolute errors of 2.3 BPM for heart rate and 0.9°C for temperature measurements, consistent with sensor specifications.

The dual-microcontroller architecture proved effective for task segregation, enabling parallel execution of medication reminders (ESP32) and health monitoring (Arduino Nano) without real-time operating system overhead. The use of Telegram Bot API eliminates recurring subscription costs associated with commercial IoT platforms such as Blynk ($4.99-15/month) and reduces operational expenses compared to GSM-based solutions requiring cellular service fees. System accessibility is further enhanced by the familiar chat interface, requiring no installation, updates, or digital literacy beyond basic messaging skills common among elderly users.

Future enhancements may include integration of DFPlayer Mini for multiple pre-recorded voice messages, implementation of machine learning algorithms for adherence pattern analysis and personalized reminder timing optimization, expansion to multi-user caregiver notifications, and incorporation of computer vision for pill recognition to prevent medication errors. The modular hardware design facilitates these upgrades without architectural redesign. MediGuard demonstrates that sophisticated medication adherence monitoring systems can be achieved through strategic integration of ubiquitous communication platforms with embedded systems, providing a scalable and economically viable solution suitable for resource-constrained healthcare environments, particularly benefiting elderly populations in developing countries facing rapid demographic aging.
.

DAFTAR PUSTAKA

[1] A. Mathew, J. Paul, K. Nair S., U. S. Sachin, S. Koncherry, and C. V. Raghu, "Design and Implementation of a Smart Medicine Dispenser," in *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, IEEE, Oct. 2019, pp. 1059–1064. doi: 10.1109/TENCON.2019.8929483.

[2] D. Z. Oktavia, E. N. Fawwaz, A. H. Al-Azhari, and Djuniadi, "Implementasi Alat Penyiram Tanaman Bonsai Otomatis Menggunakan Sensor Kelembapan Tanah Dan Arduino," *JURITEK: Jurnal Ilmiah Teknik Mesin, Elektro dan Komputer*, vol. 5, pp. 401–409, Nov. 2025, doi: 10.51903/juritek.v5i3.4821.

[3] F. Kleinsinger, "The Unmet Challenge of Medication Nonadherence," *Perm J*, vol. 22, no. 3, Sep. 2018, doi: 10.7812/TPP/18-033.

[4] BPS..go.id, "Statistik Penduduk Lanjut Usia 2023," *Badan Pusat Statistik*, Jakarta, 2023. Accessed: Jan. 14, 2026. [Online]. Available: https://www.bps.go.id/id/publication/2023/12/07/1fbd77dd0cc6e71dbb635c06/statistik-penduduk-lanjut-usia-2023.html

[5] N. Masnoon, S. Shakib, L. Kalisch-Ellett, and G. E. Caughey, "What is polypharmacy? A systematic review of definitions," *BMC Geriatr*, vol. 17, no. 1, p. 230, Dec. 2017, doi: 10.1186/s12877-017-0621-2.

[6] K. Santo, S. S. Richtering, J. Chalmers, A. Thiagalingam, C. K. Chow, and J. Redfern, "Mobile Phone Apps to Improve Medication Adherence: A Systematic Stepwise Process to Identify High-Quality Apps," *JMIR Mhealth Uhealth*, vol. 4, no. 4, p. e132, Dec. 2016, doi: 10.2196/mhealth.6742.

[7] A. Muhammad Taqiy Almy, Y. Nesta Andyanto, and A. Hamid Al-azhari, "Implementasi Smart Energy Meter Untuk Monitoring Konsumsi Listrik Rumah Tangga."

[8] M. F. Soambaton, D. Djuniadi, and A. H. Al-Azhari, "MONITORING KOLAM IKAN NILA BERBASIS IoT DENGAN SENSOR AMONIA, SUHU, KETINGGIAN, DAN PH," *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 12, no. 2, Apr. 2024, doi: 10.23960/jitet.v12i2.4021.

[9] L. Gualtieri, M. Rigby, D. Wang, and E. Mann, "Medication Management Strategies to Support Medication Adherence: Interview Study With Older Adults," *Interact J Med Res*, vol. 13, p. e53513, Aug. 2024, doi: 10.2196/53513.

[10] S. Syafrudin *et al.*, "Removal Efficiency of Chemical Oxygen Demand on Greywater using Multi Soil Layering (MSL) Technology," *Jurnal Presipitasi : Media Komunikasi dan Pengembangan Teknik Lingkungan*, vol. 18, no. 2, pp. 299–305, Jul. 2021, doi: 10.14710/presipitasi.v18i2.299-305.

[11] F. Serpush, M. B. Menhaj, B. Masoumi, and B. Karasfi, "Wearable Sensor-Based Human Activity Recognition in the Smart Healthcare System," *Comput Intell Neurosci*, vol. 2022, pp. 1–31, Feb. 2022, doi: 10.1155/2022/1391906.

[12] J. Mulyono and E. Apriaskar, "Simulasi Alarm Kebakaran Menggunakan Sensor Mq-2, Falme Sensor Berbasis Mikrokontroler Arduino," vol. 14, no. 1, pp. 16–25, 2021, [Online]. Available: http://journal.stekom.ac.id/index.php/elkom

[13] N. Aprilia Damayanti, A. Hamid Al-Azhari, and T. Elektro, "Otomatisasi Pemantauan dan Penyiraman Lidah Mertua dengan Sensor LDR dan Kelembaban Tanah," 2024.

[14] M. S. Patel, D. A. Asch, and K. G. Volpp, "Wearable Devices as Facilitators, Not Drivers, of Health Behavior Change," *JAMA*, vol. 313, no. 5, p. 459, Feb. 2015, doi: 10.1001/jama.2014.14781.

[15] I. G. M. N. Desnanjaya, A. A. G. B. Ariana, I. M. A. Nugraha, I. K. A. G. Wiguna, and I. M. U. Sumaharja, "Room Monitoring Uses ESP-12E Based DHT22 and BH1750 Sensors," *Journal of Robotics and Control (JRC)*, vol. 3, no. 2, pp. 205–211, Feb. 2022, doi: 10.18196/jrc.v3i2.11023.

[16] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015, doi: 10.1109/COMST.2015.2444095.

[17] A. E. Ramadhan, D. Djuniadi, and E. Apriaskar, "Sistem Pengaturan Pulse Width Modulation Motor Pada Robot Pembawa Makanan atau Minuman Menggunakan Joystick," *TELKA - Telekomunikasi Elektronika Komputasi dan Kontrol*, vol. 7, no. 2, pp. 134–143, Nov. 2021, doi: 10.15575/telka.v7n2.134-143.

[18] N. Cameron, "ESP32 Microcontroller," 2023, pp. 1–54. doi: 10.1007/978-1-4842-9376-8_1.

[19] A. Nistico, D. Markudova, M. Trevisan, M. Meo, and G. Carofiglio, "A comparative study of RTC applications," in *2020 IEEE International Symposium on Multimedia (ISM)*, IEEE, Dec. 2020, pp. 1–8. doi: 10.1109/ISM.2020.00007.

[20] M. Ciampi, A. Coronato, M. Naeem, and S. Silvestri, "An intelligent environment for preventing medication errors in home treatment," *Expert Syst Appl*, vol. 193, p. 116434, May 2022, doi: 10.1016/j.eswa.2021.116434.

[21] J. Pandini, M. A. Bin Zulkipli, and Fenoria Putri, "Concept Development of Automatic Sliding Door and Light Using NodeMCU Sensor," *International Journal of Mechanics, Energy Engineering and Applied Science (IJMEAS)*, vol. 2, no. 1, pp. 20–25, Jan. 2024, doi: 10.53893/ijmeas.v2i1.236.

[22] S. Sumardiono, A. R. Saputra, S. Solikin, J. Shadiq, R. Apriani, and D. I. Putri, "Prototype of Library Noise Detection Tool with IoTbased ESP32 and Blynk," in *2024 Ninth International Conference on Informatics and Computing (ICIC)*, IEEE, Oct. 2024, pp. 1–6. doi: 10.1109/ICIC64337.2024.10956742.

[23] A. C. Gheorghe and C. I. Stoica, "Wireless Weather Station Using Arduino Mega and Arduino Nano," *The Scientific Bulletin of Electrical Engineering Faculty*, vol. 21, no. 1, pp. 35–38, Apr. 2021, doi: 10.2478/sbeef-2021-0008.

[24] M. Muthmainnah and D. B. Tabriawan, "Prototipe Alat Ukur Detak Jantung Menggunakan Sensor MAX30102 Berbasis Internet of Things (IoT) ESP8266 dan Blynk," *JISKA (Jurnal Informatika Sunan Kalijaga)*, vol. 7, no. 3, pp. 163–176, Sep. 2022, doi: 10.14421/jiska.2022.7.3.163-176.

[25] G. M. Debele and X. Qian, "Automatic Room Temperature Control System Using Arduino UNO R3 and DHT11 Sensor," in *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, IEEE, Dec. 2020, pp. 428–432. doi: 10.1109/ICCWAMTIP51612.2020.9317307.

[26] L. M. Easterline, A. A.-Z. R. Putri, P. S. Atmaja, A. L. Dewi, and A. Prasetyo, "Smart Air Monitoring with IoT-based MQ-2, MQ-7, MQ-8, and MQ-135 Sensors using NodeMCU ESP32," *Procedia Comput Sci*, vol. 245, pp. 815–824, 2024, doi: 10.1016/j.procs.2024.10.308.