BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Sistem

Pada tahap ini, sistem pengendalian kualitas proses pembakaran batu bata mulai direalisasikan melalui integrasi antara perangkat keras dan perangkat lunak. Implementasi sistem dilakukan untuk memastikan bahwa alat dapat berfungsi secara optimal sesuai dengan rancangan yang telah dibuat pada bab sebelumnya.

Sistem ini bertujuan untuk memantau suhu dalam ruang pembakaran secara berkala menggunakan sensor, kemudian mengirimkan data tersebut kepada pengguna melalui aplikasi Telegram. Dengan cara ini, operator dapat mengetahui kondisi suhu pembakaran secara real-time tanpa harus memantau secara manual secara terus-menerus. Proses implementasi dilakukan melalui dua tahapan utama, yaitu perancangan perangkat keras dan pengembangan perangkat lunak.

4.1.1 Perangkat Keras (Hardware)

Perangkat keras dalam sistem ini merupakan bagian penting yang bertanggung jawab terhadap proses pengambilan data suhu serta pengiriman data ke server. Komponen-komponen elektronik yang digunakan dirakit menjadi satu kesatuan sistem yang saling terhubung. Adapun daftar komponen perangkat keras yang digunakan dapat dilihat pada tabel berikut:

Tabel 4. 1 Perangkat Keras

No	Komponen	Fungsi
1	Arduino Uno	Pengontrol Utama yang menangani proses pengolahan data
2	ESP8266 NodeMCU	Penghubung yang menangani komunikasi Wi-fi, *Bot Telegram* dan Blynk

4	Sensor Termokopel	Mengukur suhu di dalam ruang pembakaran.
5	Power Supply 5V	Memberi daya ke keseluruhan sistem
6	Kabel Jumper	Sebagai penghubung antar komponen pada breadboard.
7	Box Pelindung	Melindungi rangkaian dari panas dan gangguan luar.
8	LCD I2C	Sebagai tampilan untuk *monitoring* secara *real-time*
9	Buzzer	Sebagai indikator penanda dengan bunyi beep
10	LED	Sebagai indikator penanda dengan visual warna

Perakitan dilakukan pada papan breadboard untuk menyusun hubungan antar komponen secara sementara sebelum diimplementasikan secara permanen. Sensor DHT22 diposisikan agar menghadap langsung ke dalam tungku pembakaran untuk menangkap suhu dengan lebih akurat, sedangkan ESP8266 akan memproses data dan mengirimkan hasilnya ke *Bot Telegram* melalui jaringan *internet*.

4.1.2 Perangkat Lunak (*Software*)

Perangkat lunak berperan dalam mengatur logika kerja sistem, mulai dari pembacaan sensor, pengolahan data, hingga pengiriman notifikasi ke Telegram. Pengembangan dilakukan menggunakan Arduino IDE dengan bahasa pemrograman C/C++ yang sesuai dengan platform ESP8266.

Berikut adalah perangkat lunak dan *Library* yang digunakan dalam implementasi:

1. Perangkat Lunak Utama

   Perangkat lunak utama yang digunakan dalam pengembangan sistem adalah Arduino IDE, Merupakan lingkungan pengembangan (*Integrated Development Environment*) yang digunakan untuk menulis, mengompilasi, dan mengunggah

program ke ESP8266 NodeMCU dan Arduino UNO. Arduino IDE mendukung banyak *board mikrokontroler* serta memudahkan manajemen *Library* dan komunikasi *serial* saat *debugging*.

2. *Library* Pendukung

   Agar sistem dapat berjalan sesuai fungsinya, digunakan beberapa *Library* pendukung berikut:

   a. *Library* ESP8266WiFi.h : Berfungsi mengelola koneksi mikrokontroler ke jaringan *Wi-Fi*. *Library* ini menyediakan fungsi untuk menyambungkan ke *SSID* tertentu, memeriksa status jaringan, dan menangani *IP*.

   b. *Library* WiFiClientSecure.h : Berfungsi mendukung komunikasi dengan *server* eksternal menggunakan protokol *HTTPS*. *Library* ini penting untuk menjamin keamanan saat berinteraksi dengan *API Telegram*.

   c. *Library* UniversalTelegramBot.h : Memfasilitasi komunikasi dua arah antara *mikrokontroler* dan pengguna Telegram. *Library* ini memungkinkan sistem mengirim dan menerima pesan melalui *bot*, serta memproses perintah dari pengguna.

   d. SPI.h : Digunakan untuk mengatur komunikasi SPI antara *mikrokontroler* dan *modul MAX6675*.

   e. Adafruit_MAX31855.h : Menyediakan fungsi untuk membaca data suhu dari *sensor termokopel* yang terhubung melalui *modul MAX6675*.

   f. LiquidCrystal_I2C.h : Digunakan untuk mengatur komunikasi antara *mikrokontroler* dan *LCD 16x2* melalui antarmuka *I2C*, sehingga informasi suhu atau status sistem dapat ditampilkan secara *real-time.*

g. SoftwareSerial.h : bila ingin mengalokasikan *pin serial* terpisah untuk komunikasi Arduino ↔ ESP8266 sehingga USB tetap untuk *debugging*.

h. BlynkSimpleEsp8266.h: Digunakan untuk menghubungkan *mikrokontroler ESP8266* ke *platform Blynk IoT. Library* ini menyediakan fungsi untuk mengirim dan menerima data melalui *Virtual Pin*, mengatur *event*, serta mengintegrasikan *sensor/aktuator* dengan *dashboard Blynk* secara *real-time*.

3. Perangkat Lunak Pendukung Dokumentasi Sistem

Selain pengembangan sistem inti, beberapa aplikasi tambahan digunakan untuk membuat dokumentasi visual guna menjelaskan struktur dan alur sistem secara lebih jelas, yaitu:

a. Fritzing : Digunakan untuk membuat skema rangkaian *elektronik* dari seluruh komponen *hardware* yang digunakan. Melalui *Fritzing*, hubungan antar komponen seperti ESP8266, *sensor Termokopel*, dan *power supply* dapat divisualisasikan dalam bentuk *layout* yang mudah dipahami, baik dalam *mode breadboard* maupun *schematic*.

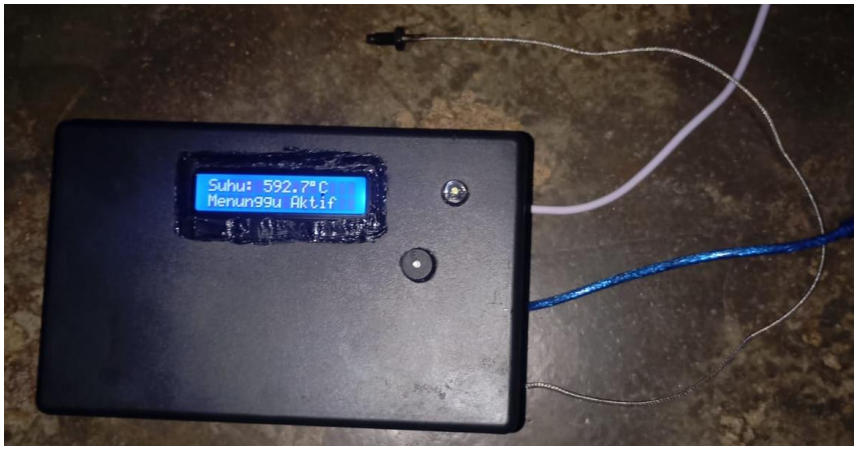
b. Microsoft Visio : Aplikasi ini digunakan untuk menggambarkan alur kerja sistem (*flowchart*), mulai dari pembacaan suhu oleh sensor, pemeriksaan ambang batas suhu, hingga pengiriman data atau notifikasi ke Telegram. Dengan adanya *flowchart*, proses logika sistem dapat dijelaskan secara sistematis dan rapi.

Penggunaan *Fritzing* dan *Microsoft Visio* membantu memberikan gambaran yang lebih *komprehensif* terhadap rancangan sistem, baik dari sisi fisik (rangkaian)

maupun logika (alur kerja), sehingga mempermudah proses implementasi maupun penjelasan dalam laporan.

4.2 Rangkaian Sistem

Rangkaian sistem pada penelitian ini dirancang untuk menghubungkan semua komponen *hardware* sehingga bekerja secara terintegrasi dalam pengendalian kualitas proses pembakaran batu bata. Arsitektur yang digunakan memisahkan tugas: *Arduino Uno* berperan sebagai unit akuisisi dan pengolahan data (membaca *sensor termokopel*, menampilkan ke *LCD*, mengendalikan indikator), sedangkan ESP8266 bertugas sebagai modul komunikasi yang mengirim data/notifikasi ke *Bot Telegram*. Perancangan rangkaian dilakukan dalam dua tahap, yaitu pembuatan desain rangkaian secara virtual menggunakan Fritzing, kemudian perakitan rangkaian fisik di lapangan.



Gambar 4. 1 Rangkaian Keseluruhan Sistem

Penjelasan Fungsi Masing-Masing Modul:

1. Sensor Termokopel + Modul Amplifier (MAX6675)

   Digunakan untuk mengukur suhu pada rongga pembakaran. *Termokopel tipe K* (*probe*) menghasilkan sinyal *millivolt* yang kemudian diperkuat dan dikonversi

ke data digital oleh *modul MAX6675* agar dapat dibaca oleh Arduino melalui antarmuka *SPI*.

2. Arduino Uno

   Berfungsi sebagai *mikrokontroler* utama yang: membaca nilai suhu dari modul *termokopel* (*SPI*), memproses logika ambang suhu (ambang_min dan ambang_max), menampilkan data ke *LCD I2C*, mengendalikan *indikator* lokal (*buzzer* dan *LED*), serta mengirimkan data ringkas melalui komunikasi serial ke ESP8266.

3. ESP8266 (*NodeMCU* sebagai modul *Wi-Fi*)

   Berperan sebagai modul komunikasi *nirkabel*. ESP menerima data dari Arduino via *UART* (*serial*) lalu menghubungkan ke jaringan *Wi-Fi* dan mengirim pesan/notifikasi ke *Bot Telegram* menggunakan koneksi *HTTPS*.

4. LCD I2C (16×2)

   Menampilkan nilai suhu saat ini, status proses (NORMAL / WARNING), dan informasi waktu pembacaan secara lokal sehingga operator dapat langsung melihat kondisi tanpa membuka Telegram.

5. Buzzer (indikator lokal)

   Memberi sinyal suara/*visual* ketika kondisi suhu berada di luar ambang batas agar *operator* di lokasi segera melakukan tindakan.

6. Power Supply (5V dan 3.3V regulator)

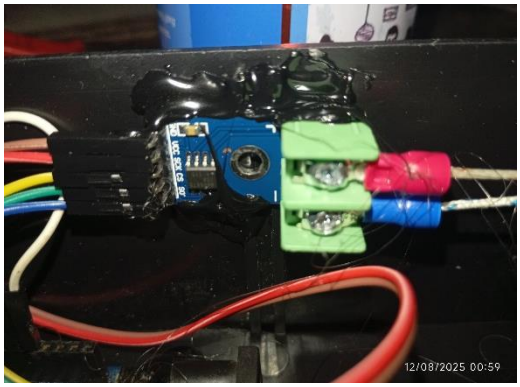
   Menyediakan catu daya 5V untuk *Arduino* dan *LCD* serta catu daya 3.3V yang stabil untuk *ESP8266*. *Ground* semua modul dihubungkan menjadi *common GND*.

7. Breadboard & Kabel Jumper

Digunakan untuk perakitan *prototipe*; kabel *jumper* mempermudah koneksi antar komponen sebelum dirancang *PCB* permanen.

4.2.1 Rangkaian Sensor Termokopel + Modul MAX6675

Modul MAX6675 mengkonversi sinyal dari termokopel ke format digital yang dapat dibaca Arduino melalui SPI. Penempatan probe termokopel harus di lokasi yang representatif terhadap kondisi pembakaran (tidak langsung bersentuhan dengan api jika probe tidak tahan temperatur ekstrim sesuaikan tipe probe).



Gambar 4. 2 Rangkaian MAX6675

Keterangan koneksi:

a. VCC (MAX6675) → 5V Arduino (atau sesuai spesifikasi modul)

b. GND → GND Arduino

c. SCK → D13 (SCK) Arduino

d. SO / MISO → D12 (MISO) Arduino

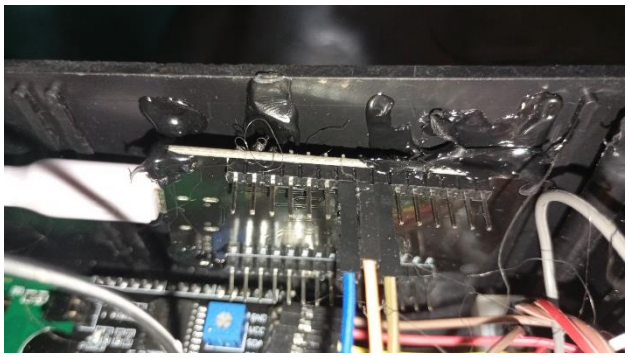
e. CS → D10 (SS / Chip Select) Arduino

4.2.2 Rangkaian Komunikasi Arduino ↔ ESP8266

Komunikasi antara Arduino Uno dan ESP8266 NodeMCU dilakukan menggunakan antarmuka *UART* (Serial). Arduino bertindak sebagai pengirim data hasil pembacaan suhu dan status sistem, sedangkan ESP8266 menerima data

tersebut untuk diproses lebih lanjut dan dikirimkan ke *Bot Telegram* melalui koneksi *Wi-Fi*.

Agar komunikasi berjalan dengan baik, diperlukan perhatian terhadap perbedaan level logika: Arduino Uno bekerja pada tegangan logika 5V, sedangkan ESP8266 bekerja pada 3,3V. Oleh karena itu, pada jalur TX Arduino → RX ESP8266 digunakan pembagi tegangan (*voltage divider*) untuk menurunkan sinyal dari 5V menjadi 3,3V. Jalur TX ESP8266 → RX Arduino dapat langsung dihubungkan karena sinyal 3,3 V masih dapat dibaca oleh Arduino sebagai logika HIGH.



Gambar 4. 3 Rangkaian ESP8266

Keterangan koneksi:

a. TX (D3) → D5 (ESP8266)

b. RX (D2) ← D6 (ESP8266)

c. GND (Arduino) ↔ GND (ESP8266)

Prinsip kerja rangkaian:

a. *Arduino* membaca suhu dari *sensor termokopel* dan memproses logika ambang batas.

b. Data suhu dan status dikirim melalui komunikasi *serial* ke *ESP8266*.

c. *ESP8266* mem-*parsing* data yang diterima dan mengirimkannya ke *Bot*

Telegram menggunakan *protokol HTTPS*.

d. Sistem ini memungkinkan pengiriman pesan otomatis tanpa interaksi *manual* dari *operator* di lokasi.

Catatan Teknis:

a. Untuk menghindari gangguan komunikasi, kecepatan *baud rate* disesuaikan 9600 agar stabil.

b. Jika digunakan *pin digital* biasa untuk komunikasi serial tambahan, maka pada Arduino diperlukan *Library SoftwareSerial* untuk mendefinisikan pin RX/TX khusus, sehingga port serial bawaan tetap bisa digunakan untuk *debugging*.

4.2.3 Rangkaian Rangkaian LCD I2C (16×2)

LCD 16×2 dengan modul antarmuka *I2C* digunakan untuk menampilkan informasi suhu pembakaran, status kondisi (NORMAL / WARNING), serta waktu pembacaan data secara *real-time*. Penggunaan antarmuka *I2C* meminimalkan jumlah pin yang dibutuhkan karena hanya menggunakan dua jalur komunikasi data (*SDA* dan *SCL*), sehingga *pin Arduino* lainnya tetap tersedia untuk sensor dan modul lain.

Gambar 4. 4 Rangkaian LCD I2C

Keterangan koneksi servo 1:

a. VCC (LCD I2C) → 5 V Arduino

b. GND (LCD I2C) → GND Arduino

c. SDA (LCD I2C) → A4 (SDA) Arduino Uno

d. SCL (LCD I2C) → A5 (SCL) Arduino Uno

Prinsip kerja rangkaian:

a. Arduino membaca nilai suhu dari *sensor termokopel* melalui modul *MAX6675*.
b. Data yang diperoleh diolah dan ditampilkan pada *LCD 16×2*, mencakup:
c. Baris 1: Nilai suhu saat ini dalam °C
d. Baris 2: Status proses pembakaran (NORMAL / WARNING) dan waktu pembacaan
e. Informasi ini memungkinkan operator memantau kondisi tungku pembakaran tanpa harus mengakses Telegram, sehingga respons terhadap perubahan suhu bisa lebih cepat.

Catatan teknis:

a. *Library* LiquidCrystal_I2C.h digunakan untuk memudahkan komunikasi antara Arduino dan LCD melalui protokol I2C.
b. Alamat *I2C LCD* biasanya *0x27* atau *0x3F* tergantung modul yang digunakan, dan harus disesuaikan pada kode program.
c. Kontras tampilan LCD dapat diatur menggunakan *potensiometer* yang terdapat pada modul *I2C*.

4.2.4 Rangkaian Indikator

Buzzer dan *LED* digunakan sebagai *indikator* lokal yang memberikan peringatan langsung kepada *operator* di lokasi ketika suhu pembakaran berada di luar batas aman. LED berfungsi sebagai penanda visual, sedangkan *buzzer* berfungsi sebagai penanda suara. Kombinasi keduanya memastikan bahwa

peringatan dapat terdeteksi dengan cepat meskipun operator tidak melihat layar *LCD.*

Gambar 4. 5 Rangkaian indikator

Keterangan koneksi:

a. Buzzer (+) → Pin Digital D8 Arduino
b. Buzzer (–) → GND Arduino
c. LED Anoda (+) → Pin Digital D7 Arduino (melalui resistor 220 Ω)
d. LED Katoda (–) → GND Arduino

Prinsip kerja rangkaian:

a. Arduino membaca suhu dari *sensor termokopel*.
b. Jika suhu melebihi ambang batas maksimum atau turun di bawah ambang batas minimum, Arduino mengaktifkan *buzzer* dan menyalakan *LED indikator* sebagai tanda peringatan.
c. Jika suhu kembali ke kondisi normal, *buzzer* dan *LED otomatis* dimatikan.

Catatan teknis:

a. *Resistor seri* pada *LED* diperlukan untuk membatasi arus dan mencegah kerusakan *LED*.
b. *Buzzer* yang digunakan adalah tipe aktif, sehingga dapat langsung dikendalikan dengan logika *HIGH/LOW* dari pin digital Arduino.

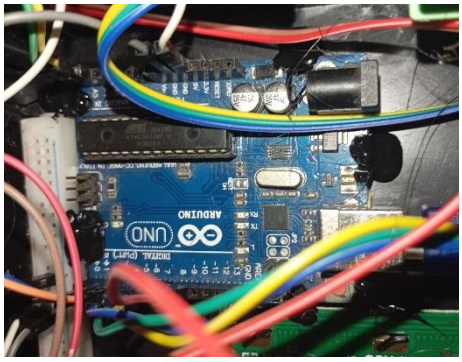
c. Untuk menghindari gangguan suara yang terus-menerus, logika program dapat mengatur *buzzer* agar berbunyi dengan pola tertentu (*beep*).

4.2.5 Diagram Alur Data Sistem

Diagram alur data sistem ini menjelaskan diagram alur data (*data flow*) sistem pengendalian proses pembakaran batu bata yang menggabungkan sensor *termokopel + modul MAX6675, Arduino Uno, LCD I2C, buzzer & LED indikator*, serta *modul ESP8266 (NodeMCU)* yang mengirim notifikasi *ke Bot Telegram*.

a. Ringkasan Alur

1. Pembacaan Sensor: *Probe termokopel* mengirimkan sinyal *millivolt* ke *modul MAX6675*.
2. Konversi & Akuisisi: *MAX6675* mengkonversi menjadi data *digital*, lalu *Arduino* membaca nilai suhu melalui *SPI*.
3. Proses & Keputusan: *Arduino* memproses data, membandingkan dengan *ambang_min* dan *ambang_max.*
4. Output Lokal:
    a. Tampilkan nilai suhu, status, dan waktu pada *LCD I2C*.
    b. Jika kondisi di luar ambang, aktifkan *LED & buzzer*.
5. Pengiriman Data: Arduino mengirim data ringkas (*TEMP:350;STATUS: WARNING;TIME:10:23) via serial UART ke ESP8266*.
6. Komunikasi Jarak Jauh: *ESP8266* mem-*parsing* pesan *serial* dan mengirim notifikasi ke *Bot Telegram* menggunakan *HTTPS*.
7. Umpan Balik Pengguna: *User* dapat meminta status *via bot; ESP8266* merespons dengan mengirimkan data terakhir yang diterima dari *Arduino*.



Gambar 4. 6 rangkaian Arduino UNO

b. Catatan Implementasi

1. Pastikan *common ground* antara Arduino dan ESP8266.
2. Gunakan format pesan yang mudah *diparse* dan tahan terhadap *noise* (sertakan *checksum* jika perlu).
3. Atur kebijakan pengiriman untuk menghindari *spam* ke *Bot Telegram*.

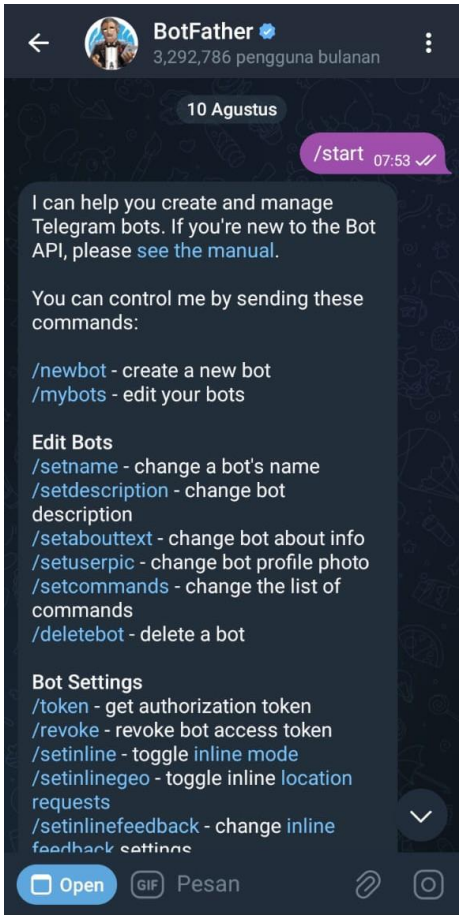
4.3 Integrasi *Bot Telegram* dan *Blynk*

Dalam sistem pengendalian proses pembakaran batu bata ini, *Bot Telegram* digunakan sebagai media komunikasi yang memungkinkan pengguna memantau kondisi suhu dan status proses secara real-time. Bot dapat mengirimkan notifikasi atau merespons perintah tertentu dari pengguna, seperti permintaan data suhu terkini.

4.3.1 Pembuatan *Bot Telegram*

Bot Telegram dibuat menggunakan aplikasi Telegram dengan bantuan bot resmi bernama @BotFather. Berikut langkah-langkah pembuatannya:

1. Buka aplikasi Telegram dan cari bot @BotFather.



Gambar 4. 7 Buka Botfather

2. Ketik perintah /newbot pada ruang percakapan dengan @BotFather untuk memulai proses pembuatan bot baru. Setelah perintah dikirimkan, @BotFather akan memberikan instruksi lanjutan untuk menetapkan *display name* atau nama tampilan bot.
3. Masukkan nama bot sesuai dengan tujuan penggunaan sistem, misalnya BatuBataBot. Nama bot berfungsi sebagai identitas publik yang akan terlihat oleh pengguna ketika membuka profil bot di Telegram. Nama ini tidak harus bersifat unik, namun disarankan menggunakan nama yang deskriptif, jelas, dan mudah diingat sehingga pengguna dapat memahami fungsi bot tersebut.

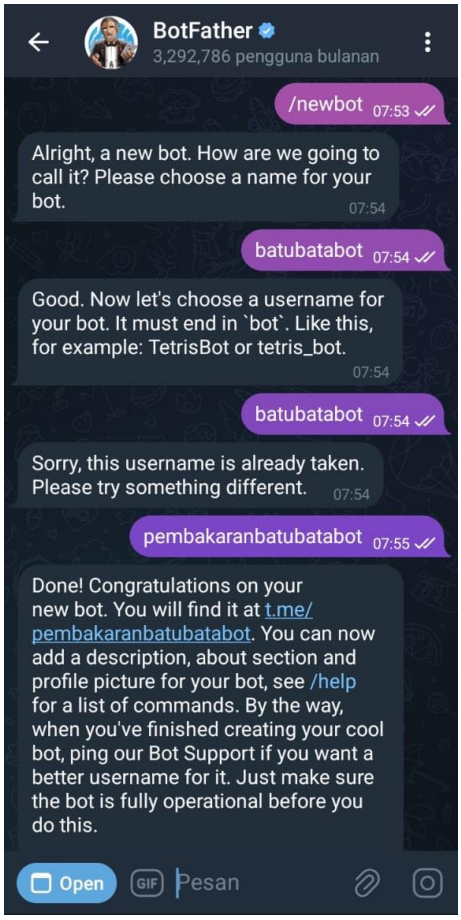
4. Setelah nama bot dimasukkan, @BotFather akan meminta penetapan *username bot*. *Username* ini bersifat unik di seluruh *platform* Telegram dan wajib diakhiri dengan kata “bot”. Contoh: pembakaranbatubatabot. Apabila *username* yang dimasukkan telah digunakan oleh pihak lain, @BotFather akan meminta untuk memilih username yang berbeda. Disarankan menggunakan kombinasi kata yang relevan dengan fungsi bot dan mudah diketik oleh pengguna.
5. Setelah *username* diterima, @BotFather akan memberikan pesan berisi informasi bot, termasuk *Token API*. Token ini merupakan rangkaian karakter unik yang berfungsi sebagai *kredensial autentikasi* untuk mengakses dan mengendalikan bot melalui program, dalam hal ini program yang dijalankan pada modul ESP8266. Contoh format token: 123456789:ABCDefGhIJKlmNoPQRstuVWxy *Token API* akan digunakan pada bagian inisialisasi pustaka (*Library*) seperti *UniversalTelegramBot* untuk menghubungkan sistem dengan layanan Telegram Bot API.
6. Token API bersifat sensitif dan harus dijaga kerahasiaannya. Apabila token diketahui oleh pihak yang tidak berwenang, maka bot dapat diakses, dikendalikan, dan digunakan untuk mengirim atau menerima pesan tanpa izin.

Oleh karena itu:

a. Token tidak boleh dibagikan di media publik, forum, atau repositori open source.

b. Simpan token pada berkas konfigurasi yang bersifat lokal dan tidak diunggah ke layanan penyimpanan publik.

c. Jika token terindikasi bocor, segera lakukan proses pembaruan token melalui perintah /revoke di @BotFather untuk menghasilkan token baru.



Gambar 4. 8 Pendaftaran Bot Baru

4.3.2 Mendapatkan Chat ID Telegram

Agar sistem dapat mengirimkan notifikasi otomatis kepada pengguna melalui *Bot Telegram* ketika suhu pembakaran melewati ambang batas yang telah ditentukan, diperlukan Chat ID sebagai alamat tujuan pengiriman pesan. Chat ID bersifat unik untuk setiap akun atau grup Telegram, sehingga harus diperoleh terlebih dahulu sebelum diintegrasikan ke dalam program ESP8266.

Prosedur mendapatkan Chat ID pengguna adalah sebagai berikut:

1. Buka aplikasi Telegram pada perangkat pengguna.
2. Cari dan buka bot resmi @userinfobot pada kolom pencarian Telegram.
3. Mulai interaksi dengan mengetuk tombol *Start* atau mengetik perintah /start.

Gambar 4. 9 Mendapatkan Chat ID

4. Bot akan membalas dengan informasi detail akun Telegram pengguna, termasuk *Chat ID*. *Chat ID* biasanya berupa angka unik yang menjadi identitas penerima pesan.
5. Salin *Chat ID* tersebut, kemudian masukkan ke dalam kode program ESP8266 pada variabel penerima pesan (*CHAT_ID*).

6. Pastikan nilai *Chat ID* dimasukkan dengan benar dan sesuai format agar pesan dapat terkirim dengan tepat.

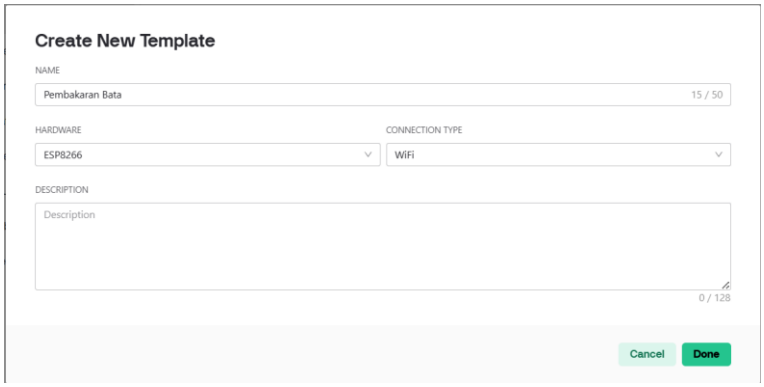
Dengan adanya *Chat ID* yang *valid*, sistem dapat mengirimkan notifikasi peringatan secara langsung ketika suhu pembakaran berada di luar rentang aman, sehingga operator dapat segera mengambil tindakan *korektif*.

4.3.3 Integrasi dengan Blynk

Selain memanfaatkan Bot Telegram, sistem pengendalian suhu pembakaran batu bata ini juga diintegrasikan dengan Blynk sebagai platform pemantauan dan pengendalian berbasis *Internet of Things* (*IoT*). *Blynk* memungkinkan pengguna memantau kondisi suhu secara *real-time* melalui aplikasi yang tersedia pada perangkat Android maupun iOS, serta mengontrol fungsi tertentu pada sistem tanpa harus menggunakan perintah teks seperti di Telegram. Keunggulan *Blynk* adalah kemudahan penggunaannya, di mana proses pembuatan *dashboard* hanya memerlukan waktu singkat dengan metode *drag* and *drop*, serta fleksibilitasnya yang tidak terbatas pada jenis papan mikrokontroler tertentu.

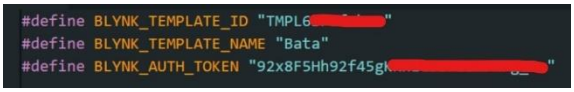
Langkah Integrasi Blynk pada Sistem:

1. Pembuatan Template di *Blynk*
    a. Buka aplikasi *Blynk IoT*, kemudian buat Template baru dengan menentukan nama template, tipe perangkat (*Device Type*) yaitu ESP8266, dan koneksi (*Wi-Fi*).



Gambar 4. 10 Buat Template Blynk

b. Sistem akan menghasilkan *Template ID* dan *Auth Token* yang digunakan sebagai kredensial untuk menghubungkan modul *ESP8266* dengan *server Blynk.*



Gambar 4. 11 Auth Token Blynk

2. Pembuatan Datastream

   a. Masuk ke menu *Datastreams* pada *template* yang sudah dibuat.

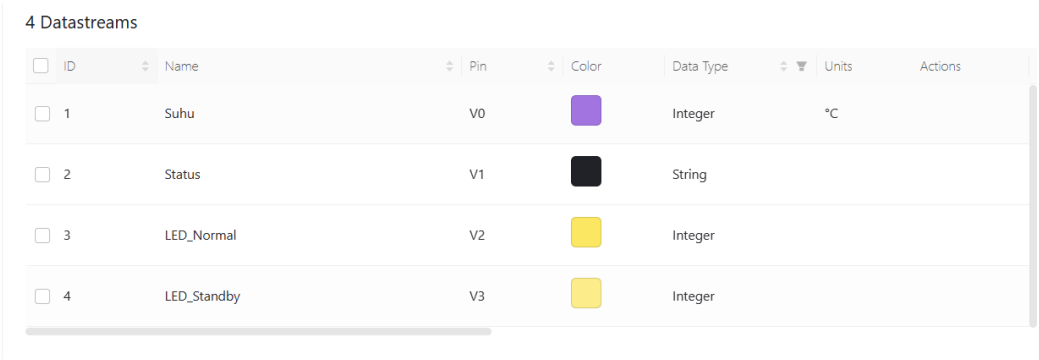
   b. Klik *New Datastream* → *Virtual Pin* untuk membuat jalur komunikasi antara *ESP8266* dan *Blynk*.

   c. Tentukan *parameter* berikut untuk setiap *datastream*:

      1. Suhu pada pin V0 dengan tipe data *Integer* dan satuan °C untuk menampilkan pembacaan suhu dari sensor.
      2. Status pada pin V1 dengan tipe data *String* untuk menampilkan status sistem, misalnya “Normal”, “Rendah” atau “Tinggi”.
      3. LED_Normal pada pin V2 dengan tipe data *Integer* untuk mengontrol atau menampilkan status *LED indikator* ketika suhu mencapai abnormal.

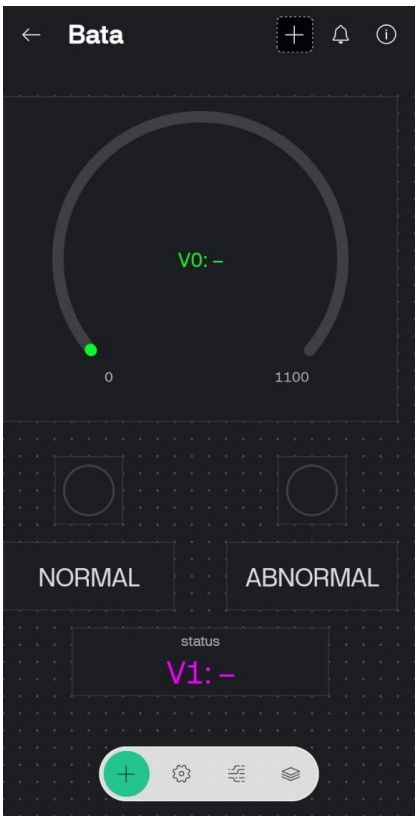
4. LED_Standby pada pin V3 dengan tipe data *Integer* untuk mengontrol atau menampilkan status *LED indikator* mode aktif dan normal.

d. Setelah semua *datastream* dibuat, klik *Save And Apply* untuk menyimpan pengaturan.



Gambar 4. 12 Datastream Blynk

3. Pembuatan Dashboard



Gambar 4. 13 Dashboard Blynk

a. Tambahkan *widget* yang dibutuhkan seperti *Gauge* untuk menampilkan suhu, LED untuk status sistem, dan Label untuk menampilkan status sistem, misalnya “Normal”, “Rendah” atau “Tinggi”.
b. Setiap widget dikonfigurasi dengan Virtual Pin yang sudah dibuat dan akan digunakan dalam pemrograman ESP8266 untuk mengirim atau menerima data.

4. Integrasi dengan Kode Program *ESP8266*

a. Gunakan pustaka *BlynkSimpleEsp8266.h* untuk menginisialisasi koneksi ke *Blynk Cloud*.
b. Masukkan Auth Token, SSID, dan password Wi-Fi ke dalam kode.
c. Pada bagian loop program, data suhu dari sensor dikirim ke Virtual Pin yang terhubung dengan widget pada dashboard Blynk. Status sistem juga diperbarui secara berkala sehingga pengguna dapat memantau kondisi terkini.

5. Pengiriman Data dan Kontrol

a. Data suhu yang diukur oleh Arduino Uno melalui sensor termokopel MAX6675 dikirimkan ke ESP8266 melalui komunikasi serial. ESP8266 kemudian mengirim data tersebut ke Blynk Cloud, yang akan memperbarui tampilan pada dashboard pengguna secara real-time.
b. Pengguna dapat mengaktifkan atau menonaktifkan fungsi tertentu seperti reset sistem melalui tombol pada dashboard, yang akan mengirimkan sinyal balik ke ESP8266 untuk diproses.

Dengan integrasi Blynk ini, sistem menjadi lebih interaktif dan mudah digunakan, karena pengguna tidak hanya menerima notifikasi atau perintah melalui

teks (seperti di Telegram), tetapi juga dapat memanfaatkan antarmuka visual yang informatif. Hal ini meningkatkan pengalaman pengguna sekaligus memperluas opsi kendali terhadap proses pembakaran batu bata dari jarak jauh.

4.3.4 Integrasi dengan ESP8266

Integrasi antara *Bot Telegram* dan perangkat *ESP8266* dilakukan melalui pemrograman yang memanfaatkan sejumlah pustaka (*Library*) pendukung. Pustaka tersebut berfungsi untuk mengelola koneksi jaringan, komunikasi aman dengan *server* Telegram, dan pengelolaan pesan masuk maupun keluar. Pustaka yang digunakan adalah sebagai berikut:

1. *ESP8266WiFi.h* – digunakan untuk menginisialisasi dan mengatur koneksi modul ESP8266 ke jaringan *Wireless Fidelity* (*Wi-Fi*) dengan memasukkan *Service Set Identifier* (*SSID*) serta kata sandi (*password*) yang sesuai.
2. *WiFiClientSecure.h* – menyediakan koneksi aman berbasis protokol *HTTPS* sehingga komunikasi data antara ESP8266 dan *server* Telegram berlangsung *terenkripsi* dan terlindungi dari potensi penyadapan.
3. *UniversalTelegramBot.h* – mengelola interaksi dengan Telegram *Bot API*, termasuk fungsi untuk membaca pesan masuk dari pengguna, memproses perintah, dan mengirimkan balasan secara terprogram.
4. *BlynkSimpleEsp8266.h* – Mengelola Interaksi dengan Blynk untuk membaca pembacaan sensor termokopel secara realtime.

Konfigurasi awal dalam pemrograman ESP8266 meliputi:

1. Penyetelan parameter jaringan *Wi-Fi*, yaitu *SSID* dan kata sandi, agar perangkat dapat terhubung ke *internet*.
2. Penyimpanan *Token API* yang diperoleh dari *@BotFather*, berfungsi sebagai

kredensial autentikasi agar *ESP8266* dapat mengakses layanan *Bot Telegram*.

3. Penetapan *Chat ID* pengguna yang berfungsi sebagai alamat tujuan pengiriman balasan bot. *Chat ID* dapat diperoleh dengan mengirimkan pesan awal ke *bot* dan membaca data respons yang diterima.
4. *Konfigurasi kredensial Blynk*, meliputi *Template ID*, *Device Name*, dan *Auth Token* yang diperoleh dari *Blynk Console*. *Kredensial* ini digunakan untuk menghubungkan *ESP8266* ke *server Blynk IoT* sehingga data dapat dipantau secara *real-time*.
5. Inisialisasi koneksi ke *Blynk Cloud* dengan *library Blynk*, yang memungkinkan pengiriman dan pembaruan data suhu, status sistem, serta *indikator LED* ke *Virtual Pin* yang telah dibuat di *Datastreams*.

Setelah inisialisasi selesai, program *ESP8266* dijalankan dengan mekanisme *loop* yang secara *periodik* membaca data terkini dari Arduino melalui komunikasi *serial* (*UART*) atau sumber *data internal*. Sistem akan memantau nilai suhu secara terus-menerus, lalu mengirimkan pembaruan data tersebut ke *Blynk* agar dapat dipantau secara *real-time* melalui *dashboard*. Selain itu, jika terdeteksi kondisi tertentu seperti suhu terlalu rendah atau terlalu tinggi, *ESP8266* akan secara otomatis mengirimkan notifikasi melalui *Bot Telegram* kepada pengguna. Pendekatan ini memastikan informasi penting tersampaikan tepat waktu tanpa memerlukan permintaan manual dari pengguna.

4.4 Implementasi Program

Program pada sistem ini dibagi menjadi dua bagian utama, yaitu *program* pada *mikrokontroler Arduino UNO* dan *program* pada *ESP8266*. Masing-masing bagian memiliki tugas spesifik dan saling terintegrasi melalui komunikasi *serial*.

a. Arduino UNO bertanggung jawab untuk membaca data suhu dari *sensor termokopel tipe K* (melalui *modul MAX6675*), menentukan status pembakaran (NORMAL, SUHU TINGGI, SUHU RENDAH), serta mengirimkan data tersebut ke *ESP8266*. Data suhu juga ditampilkan pada *LCD I2C* agar dapat dipantau langsung oleh *operator* di lokasi.
b. *ESP8266* bertugas menerima data dari Arduino, memprosesnya, dan mengirimkan pembaruan secara *real-time* ke *platform Blynk*. Selain itu, *ESP8266* juga mengirimkan notifikasi otomatis melalui *Bot Telegram* apabila suhu terdeteksi berada di luar batas aman.

4.4.1 Program Arduino UNO

Program Arduino ditulis menggunakan Arduino IDE dengan beberapa library tambahan, antara lain:

a. max6675.h untuk membaca suhu dari sensor termokopel tipe K melalui modul MAX6675.
b. Wire.h untuk komunikasi dengan LCD I2C.
c. LiquidCrystal_I2C.h untuk menampilkan data suhu dan status pada LCD.
d. SoftwareSerial.h untuk komunikasi data dengan ESP8266.

Fungsi Program Arduino:

a. Membaca suhu ruang pembakaran menggunakan sensor termokopel tipe K dan modul MAX6675.

b. Menentukan status pembakaran (NORMAL, SUHU TINGGI, atau SUHU RENDAH) berdasarkan batas suhu yang telah ditentukan (≥1000 °C untuk suhu tinggi, <700 °C untuk suhu rendah).
c. Menampilkan data suhu dan status pada LCD I2C secara real-time.
d. Mengirim data suhu dan status pembakaran ke ESP8266 melalui komunikasi serial.

4.4.2 Program ESP8266

Program ESP8266 juga ditulis di Arduino IDE, dengan library:

a. ESP8266WiFi.h untuk koneksi ke jaringan WiFi.
b. WiFiClientSecure.h untuk komunikasi HTTPS.
c. UniversalTelegramBot.h untuk integrasi dengan bot Telegram.

Fungsi Program ESP8266:

a. Terhubung ke jaringan WiFi dan menjaga koneksi tetap aktif.
b. Menerima data suhu dan status dari Arduino melalui komunikasi serial.
c. Mengirim data secara terus-menerus ke *Blynk* untuk *Monitoring real-time.*
d. Mengirim notifikasi otomatis ketika suhu mencapai atau melebihi 1000 °C (peringatan suhu tinggi) atau ketika suhu turun di bawah 700 °C (peringatan suhu rendah).

4.4.3 Pengujian Sistem

Setelah program berhasil diimplementasikan pada Arduino UNO dan ESP8266, dilakukan pengujian sistem secara menyeluruh untuk memastikan bahwa semua komponen bekerja sesuai fungsinya. Pengujian dilakukan terhadap beberapa bagian penting, yaitu sensor suhu MAX6675, LCD I2C, komunikasi Arduino–ESP8266, dan integrasi dengan bot Telegram.

Pengujian dilakukan dengan mensimulasikan kondisi suhu pembakaran pada beberapa titik pengujian, kemudian mengamati respon sistem terhadap data yang terbaca. Selain itu, pengujian juga mencakup respon bot Telegram terhadap perintah pengguna serta notifikasi otomatis.

Tabel 4. 2 Pengujian Sensor Termokopel dan Notifikasi

No	Suhu Terbaca (°C)	Indikator (LED&Buzzer)	Status yang Diharapkan	Status yang Ditampilkan	Notifikasi Terkirim	Status
1	750	LED Mati, Buzzer mati	NORMAL	NORMAL	Tidak	☑
2	1020	LED kedip, Buzzer nyala	TINGGI	TINGGI	Ya	☑
3	690	LED kedip, Buzzer nyala	RENDAH	RENDAH	Ya	☑
4	900	LED Mati, Buzzer mati	NORMAL	NORMAL	Tidak	☑

Berdasarkan hasil pengujian yang ditunjukkan pada Tabel 4.2, sistem pemantauan suhu pembakaran batu bata mampu bekerja secara konsisten sesuai dengan perancangan. Pembacaan suhu dari sensor termokopel menunjukkan nilai yang akurat, kemudian diolah oleh sistem untuk menentukan status kondisi pembakaran. Pada suhu 750°C dan 900°C, indikator LED menyala serta buzzer dalam keadaan mati, sesuai dengan status NORMAL yang diharapkan. Saat suhu mencapai 1020°C, LED berkedip dan buzzer menyala untuk menandakan kondisi TINGGI, dan notifikasi terkirim ke pengguna melalui bot Telegram. Begitu pula ketika suhu berada pada 690°C, sistem menampilkan status RENDAH dengan indikator LED berkedip dan buzzer menyala, serta mengirimkan notifikasi secara otomatis. Hasil ini menunjukkan bahwa indikator visual dan audio bekerja dengan

tepat, dan fitur notifikasi mampu menginformasikan kondisi kritis secara real-time. Dengan demikian, sistem ini tidak hanya mampu memantau suhu pembakaran secara akurat, tetapi juga memberikan peringatan dini yang dapat membantu operator mengambil langkah penanganan yang cepat dan tepat.

4.4.4 Hasil Implementasi

Setelah seluruh proses perancangan dan integrasi selesai, sistem diuji untuk memastikan bahwa setiap komponen dan fungsi berjalan sesuai dengan perencanaan. Pengujian dilakukan terhadap *sensor termokopel*, tampilan *LCD*, komunikasi Arduino – *ESP8266*, *platform Blynk*, serta fungsi notifikasi *Bot Telegram*.

1. Sensor Suhu *MAX6675*

   Pengujian ini bertujuan untuk memastikan bahwa *sensor termokopel tipe K* melalui *modul MAX6675* mampu membaca suhu ruang pembakaran secara akurat. Sensor diuji dengan memanaskan ruang pembakaran secara bertahap.

   Hasil:

   a. Sensor mampu membaca suhu hingga di atas 1000 °C.
   b. Respon pembacaan cepat, dengan pembaruan data setiap *interval* 1 detik.
   c. Data suhu stabil tanpa *fluktuasi* signifikan ketika kondisi pembakaran stabil.

2. *LCD I2C*

   LCD I2C digunakan untuk menampilkan suhu terkini dan status kondisi pembakaran.

   Hasil:

   a. Tampilan suhu pada *LCD* sesuai dengan data yang dibaca oleh sensor.

b. Status kondisi (NORMAL, TINGGI, RENDAH) tampil secara *real-time*.

c. Kontras dan keterbacaan layar tetap baik pada berbagai kondisi pencahayaan.

3. Komunikasi Arduino ke *ESP8266*

Pengujian dilakukan untuk memastikan data suhu dan status terkirim dengan benar dari Arduino Uno ke *ESP8266* melalui komunikasi *serial*.

Hasil:

a. Data terkirim dalam format "Suhu: xxx °C, Status: …" dengan benar dan konsisten.

b. Tidak ditemukan keterlambatan pengiriman data yang signifikan (< 1 detik).

c. ESP8266 dapat memproses data tersebut untuk keperluan pengiriman ke Bot Telegram.

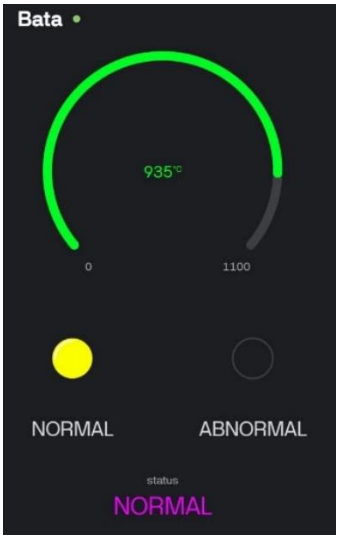
4. Blynk IoT

Platform Blynk digunakan untuk memantau suhu dan status pembakaran secara *real-time* melalui *smartphone*.

Hasil:

a. Kondisi NORMAL

Ditampilkan ketika suhu berada pada rentang aman, yaitu 700 °C hingga kurang dari 1000 °C. *Dashboard Blynk* menampilkan indikator *Normal* dengan warna kuning pada label normal, dan nilai suhu sesuai pembacaan sensor.



Gambar 4. 14 Tampilan Blynk Normal

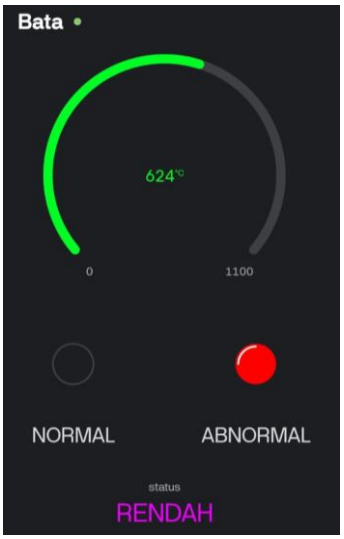
b. Kondisi TINGGI

Ditampilkan ketika suhu mencapai atau melebihi 1000 °C. *Dashboard Blynk* menampilkan indikator Tinggi dengan warna merah pada label status, disertai nilai suhu aktual yang tinggi. Kondisi ini memicu notifikasi otomatis ke Telegram.

Gambar 4. 15 Tampilan Blynk Tinggi

c. Kondisi RENDAH

Ditampilkan ketika suhu kurang dari 700 °C. *Dashboard Blynk* menampilkan indikator Rendah dengan warna biru pada label status, serta nilai suhu sesuai pembacaan sensor. Kondisi ini juga memicu notifikasi otomatis ke Telegram.



Gambar 4. 16 Tampilan Blynk Rendah

5. Notifikasi Bot Telegram

Bot Telegram diuji untuk memastikan dapat mengirimkan notifikasi otomatis sesuai batas suhu yang telah ditentukan.

Hasil:

a. Notifikasi suhu tinggi terkirim saat suhu ≥ 1000 °C dengan pesan peringatan yang jelas.
b. Notifikasi suhu rendah terkirim saat suhu < 700 °C dengan pesan peringatan yang jelas.
c. Pesan notifikasi terkirim ke Telegram Android, iOS, dan desktop tanpa kendala.

Gambar 4. 17 Pesan Saat Kondisi Abnormal

Secara keseluruhan, hasil pengujian menunjukkan bahwa semua komponen bekerja sesuai rancangan, dengan respon cepat dan data yang akurat. *Integrasi Blynk* memberikan kemudahan pemantauan suhu dan status secara *real-time*, sedangkan notifikasi otomatis Telegram membantu operator segera mengetahui kondisi kritis dan mengambil tindakan korektif dengan cepat.