BAB IV
PEMBUATAN ALAT RANCANG BANGUN PANEL SURYA DENGAN PENDINGIN AIR GUNA OPTIMALISASI DAYA *OUTPUT* DENGAN IOT BERBASIS ESP32

4.1 Pembuatan Perangkat Keras (*Hardware*)

Pembuatan perangkat keras merupakan tahapan utama dalam pengembangan sistem pendingin air berbasis *Internet of Things* (IoT) pada panel surya guna meningkatkan efisiensi daya *output*. Perangkat keras yang dirancang ini bertujuan untuk mendeteksi suhu permukaan panel surya secara *real-time*, mengaktifkan sistem pendinginan otomatis ketika suhu melebihi ambang batas tertentu, serta menampilkan dan mengirimkan data hasil pemantauan melalui aplikasi Blynk dan Telegram. Mikrokontroler ESP32 digunakan sebagai pusat kendali sistem yang mengoordinasikan kerja sensor, aktuator, dan komunikasi data.

Sistem ini dibangun menggunakan beberapa komponen utama, yaitu panel surya berkapasitas 30 Wp sebagai sumber energi, yang dikoneksikan dengan *Solar Charge Controller* tipe PWM untuk mengatur pengisian daya ke baterai VRLA 12V 9Ah. Panel surya dilengkapi dengan rangkaian pendingin air berbasis pipa tembaga yang dipasang secara zigzag di bagian belakang panel. Pipa ini dialiri air dari tandon dengan bantuan pompa air DC, yang digerakkan secara otomatis berdasarkan pembacaan suhu dari sensor DS18B20. Sensor suhu tersebut dipasang langsung di permukaan belakang panel dan berfungsi sebagai pengukur suhu kerja panel surya secara *real-time*.

Selain itu, sistem dilengkapi dengan sensor arus ACS712 dan sensor tegangan DC untuk memantau besaran *output* listrik dari panel surya. Untuk menjamin ketersediaan air pendingin, digunakan sensor ketinggian air JSN-SR04T yang dipasang pada bagian atas tandon. Sensor ini memberikan informasi tinggi permukaan air, sehingga ketika volume air tidak mencukupi, sistem dapat memberikan peringatan dan menghentikan aliran air agar tidak merusak pompa. Pompa air, kipas DC, dan solenoid valve dikendalikan melalui modul relay 5V yang menerima sinyal logika dari ESP32. Kipas pendingin berfungsi untuk membantu

sirkulasi udara dan mempercepat proses penurunan suhu, sementara solenoid valve difungsikan sebagai katup otomatis untuk membuang air apabila volume dalam tandon melebihi ambang batas.

Seluruh data dari sensor diproses oleh ESP32 dan ditampilkan secara lokal menggunakan LCD 20x4 dengan antarmuka I2C. Selain itu, data juga dikirim secara *real-time* ke aplikasi Blynk melalui koneksi Wi-Fi yang tersedia, sehingga pengguna dapat memantau suhu, tegangan, arus, dan status sistem pendingin dari jarak jauh. Sistem ini juga terhubung dengan aplikasi Telegram yang berfungsi untuk memberikan notifikasi jika suhu panel terlalu tinggi atau jika air dalam tandon habis. Untuk memenuhi kebutuhan tegangan yang berbeda-beda pada tiap komponen, digunakan modul step-down LM2596 yang menurunkan tegangan dari 12V ke 5V dan 3.3V sesuai dengan kebutuhan sensor dan mikrokontroler.

Dengan integrasi seluruh komponen tersebut, perangkat keras ini mampu berfungsi secara otomatis dan mandiri dalam mendeteksi serta menangani kondisi suhu tinggi pada panel surya. Sistem pendingin bekerja secara responsif, efisien, dan dapat dipantau dari mana saja melalui perangkat mobile. Hal ini diharapkan dapat meningkatkan efisiensi daya *output* panel surya dan memperpanjang umur operasional perangkat secara keseluruhan.

4.2 Alat dan bahan

Dalam melalukan pembuatan alat Tugas Akhir ini menggunakan alat – alat yang dapat mempermudah penyusun dalam melakukan pembuatan alat dan menggunakan bahan – bahan dalam membantu pembuatan alat. Penyusun melampirkan bahan-bahan dan alat-alat yang digunakan pada tabel 4.1 dan tabel 4.2.

Tabel 4. 1 Peralatan Pembuatan Alat

No.	Nama Alat	Spesifikasi	Jumlah
1	Solder	Taffware CS-31, 65 Watt	1 Buah
2	Obeng	Freed 17150A 75 mm	1 Set
3	Multimeter Digital	Aneng SZ-304	1 Buah

4	Tang Potong Sedang	6 inch	1 Buah
5	Bor Minidrill	130 Watt	1 Buah
6	Lem Tembak	20 Watt	1 Buah
7	Kabel Ties	15 cm, isi ±100 pcs	1 Pack
8	Isolasi Listrik	PVC, 10 meter	1 Rol
9	Gunting Kabel	6 inch	1 Buah
10	Cutter / Pisau Kecil	Kenko L-500	1 Buah
11	Kabel USB to Type-C	1 meter	1 Buah
12	Thermometer HTC-2	10°C s/d 50°C, Resolusi 0,1°C, Baterai AAA 1,5V	1 Set

Tabel 4. 2 Bahan Pembuatan Alat

No.	Nama Bahan	Spesifikasi	Jumlah
1	Panel Surya	30 Watt Peak	1 Buah
2	*Solar Charge Controller*	10 Ampere	1 Buah
3	Baterai VRLA	12V 9Ah, Deep Cycle, merk ZEUS	1 Buah
4	ESP32 Type C	32-bit, WiFi + Bluetooth, USB-C	1 Buah
5	Sensor Suhu DS18B20	Digital, 3-pin, tahan air	1 Buah
6	Sensor Arus ACS712	0–5A, *output* analog 185 mV/A	1 Buah
7	Sensor Tegangan DC	0–25V, *output* analog	1 Buah
8	Sensor Ketinggian JSN-SR04T	Ultrasonik, waterproof, 20–600 cm	1 Buah
9	Pompa Air DC	12V, aliran 3–5 L/menit	1 Buah
10	Kipas DC	12V, ukuran 8x8 cm	2 Buah
11	Solenoid Valve	12V DC, 1/4 inch, normal tertutup	1 Buah

12	Modul Relay	5 Channel, 5V, aktif LOW Trigger	5 Channel
13	LCD 20x4 + I2C	5V, interface I2C, 20 karakter x 4 baris	1 Buah
14	Modul Step-down LM2596	Input 4–35V, *output* 1.23–30V, max 3A	1 Buah
15	Pipa Tembaga	Diameter 1/4 inch, panjang ±3,5 meter	±3,5 Meter
16	Selang Bening	Diameter 1/4 inch, panjang ±2,5 meter	±2,5 Meter
17	Tandon Air (Ember Plastik)	Volume ±20 liter, bahan plastik PE	1 Buah
18	Box Panel Besi	Dimensi ±30x25x15 cm, warna abu, knockdown	1 Buah
19	Pilot Lamp Merah dan Hijau	220V AC, LED indikator panel	2 Buah
20	Saklar Toggle	2 posisi ON-OFF, 250V 6A	1 Buah
21	Sakelar	Saklar biasa 220V AC	2 Buah
22	Tiang + Bracket	Tinggi ±1,4 meter, lebar ±50–70 cm, panjang ±70–80 cm, material besi, sudut disesuaikan posisi panel surya	1 Set
23	Stop Kontak	220V AC, 3 lubang	1 Buah
24	Inverter (Modified Sine Wave)	Inverter DC to AC 220W, 12VDC (Input), 220VAC (*Output*), 50 Hz	1 Buah
25	Terminal Block	2 pin, screw type, untuk sambungan kabel	±5 Buah
26	Socket Pin Header (Female)	40 pin/strip, jarak 2.54 mm	5 Strip

27	Kabel Jumper	Male-Male dan Female-Female, panjang ±20 cm	±10 Buah
28	Kabel Power DC	Diameter ±2 mm, warna merah-hitam, ±3 meter	±3 Meter
29	PCB Bolong	9x15 cm, bahan fiber, lubang 2.54 mm	1 Lembar
30	Keypad 1x4	4 tombol, bentuk baris tunggal	1 Buah
31	Lampu 5W	LED, 220V AC	1 Buah
32	Kabel Bintik	Diameter ±1 mm, isi 2, ±12 meter	±12 Meter
33	Timah	Diameter ±0.8 mm, gulungan ±10 meter	1 Gulung
34	Alumunium Foil	Lembaran ±30x30 cm, reflektif panas	1 Lembar
35	Seal Tape Pipa	Teflon tape, lebar ±12 mm, panjang ±10 meter, warna putih	1 Rol
36	Selang Dispenser	Diameter 5/16 inch, panjang ±80 cm	2 Buah
37	Modul Peltier	Tipe TEC1-12706, tegangan kerja 12V DC, ukuran 40x40 mm	1 Buah

4.3 Langkah-Langkah Pembuatan Perangkat Keras (*Hardware*)

4.3.1 Pembuatan Penyangga Panel Surya

Tahap awal dalam proses pembuatan perangkat keras adalah menyiapkan penyangga untuk panel surya. Penyangga dibuat dari besi hollow yang dipotong dan disambung dengan teknik pengelasan, membentuk struktur rangka vertikal yang kokoh dan stabil. Rangka ini dirancang agar panel surya dapat dipasang pada dudukan yang fleksibel untuk diarahkan sesuai kebutuhan.

Penyangga panel dilengkapi dengan mekanisme penguncian manual menggunakan baut pada dua titik sumbu, yaitu arah horizontal (timur–barat) dan vertikal (kemiringan elevasi). Untuk mengatur arah panel, baut pengunci cukup dilonggarkan, kemudian posisi panel diubah, lalu dikencangkan kembali setelah mencapai sudut terbaik terhadap arah sinar matahari. Meskipun belum otomatis, sistem ini sudah cukup efektif untuk mengoptimalkan penyerapan energi matahari pada berbagai waktu dalam sehari.

Gambar 4. 1 Tiang Penyangga Panel Surya

4.3.2 Pembuatan Panel Box

Gambar 4. 2 Modifikasi Panel Box

Panel box berfungsi sebagai pusat distribusi dan pengendali seluruh sistem panel surya. Panel ini tidak hanya menampung sistem monitoring, tetapi juga mencakup seluruh komponen utama seperti baterai VRLA 12V 9Ah, *Solar Charge Controller* (SCC), mikrokontroler ESP32, sensor-sensor, relay, dan regulator tegangan LM2596.

Panel yang digunakan berbahan dasar logam (besi) yang kuat dan tahan terhadap kondisi luar ruangan. Modifikasi dilakukan dengan membuat lubang pada bagian muka panel untuk pemasangan LCD 20x4, key switch, pilot lamp, dan tombol pengendali. Di bagian dalam, komponen-komponen disusun secara sistematis untuk memudahkan pengkabelan dan sirkulasi udara.

Selain sebagai wadah, panel box ini juga berfungsi sebagai sistem proteksi terhadap komponen dari air, debu, dan gangguan luar. Dengan adanya integrasi seluruh sistem ke dalam satu box, pengoperasian dan pemeliharaan menjadi lebih efisien dan aman.

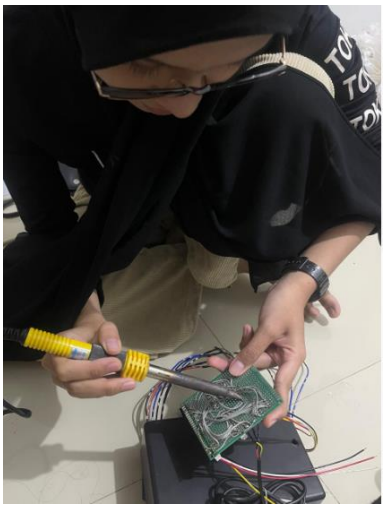
Gambar 4. 3 Panel Box Tampak Depan Setelah Dimodifikasi

4.3.3 Perancangan Rangkaian Sistem pada PCB

Perancangan rangkaian dilakukan pada PCB bolong (veroboard) sebagai tempat penyusunan jalur dan penyolderan komponen. Komponen utama yang dirakit pada PCB ini meliputi ESP32, header sensor, LCD I2C, modul relay, dan beberapa konektor *output*.

Penyolderan dilakukan dengan hati-hati menggunakan solder uap dan timah berkualitas untuk menghasilkan sambungan yang kuat dan rapi. Jalur VCC dan GND disusun menyebar untuk menghindari lonjakan arus dan gangguan sinyal. Posisi tiap komponen disesuaikan dengan wiring diagram dan kebutuhan koneksi ke komponen lain yang diletakkan dalam box.

Perancangan ini menjadi inti dari sistem kendali dan monitoring karena dari PCB inilah ESP32 akan menerima input dari sensor, memproses data, dan mengirimkan *output* ke aktuator serta tampilan LCD. Penempatan pada papan veroboard dipilih agar fleksibel dan ekonomis dalam pengembangan sistem skala prototipe.



Gambar 4. 4 Penyolderan Rangkaian PCB

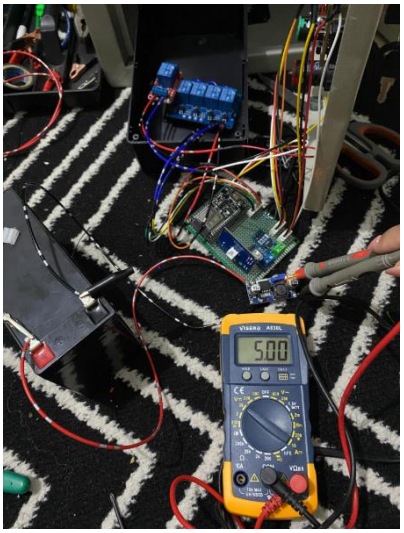
4.3.4 Pengaturan Catu Daya Sistem

Untuk memastikan seluruh komponen sistem mendapatkan suplai tegangan yang sesuai, dilakukan pengaturan catu daya menggunakan modul step-down LM2596. Panel surya menghasilkan tegangan yang disalurkan ke baterai VRLA

12V melalui *Solar Charge Controller* (SCC). Tegangan dari baterai inilah yang menjadi sumber utama untuk sistem kontrol dan monitoring.

Karena mikrokontroler ESP32 dan beberapa sensor hanya membutuhkan tegangan sebesar 5V, maka diperlukan penurunan tegangan dari 12V menjadi 5V. LM2596 digunakan untuk mengkonversi tegangan ini secara stabil dan efisien. Proses pengaturan dilakukan dengan memutar trimpot pada modul LM2596 hingga *output* yang diukur menggunakan multimeter digital menunjukkan nilai 5V.

Output dari LM2596 kemudian disambungkan ke jalur input ESP32, sensor suhu DS18B20, LCD 20x4, serta modul relay. Penggunaan regulator ini penting untuk mencegah overvoltage yang dapat merusak komponen mikrokontroler dan sensor digital lainnya. LM2596 juga dipasang di dalam panel box agar terlindungi dari air dan gangguan luar.



Gambar 4. 5 Pengaturan LM2596 dari *Output* Aki 12V to 5VDC (Input Mikrokontroler)

4.3.5 Perakitan Komponen dalam Box

Setelah semua rangkaian dan sistem catu daya selesai dirakit dan diuji, langkah selanjutnya adalah merakit seluruh komponen ke dalam panel box. Panel box berfungsi sebagai pusat pengendali dan pelindung seluruh sistem elektronik.

Komponen yang dipasang di dalam panel box antara lain mikrokontroler ESP32, modul relay, sensor-sensor, LM2596, serta konektor input-*output*.

Rangkaian hasil penyolderan pada PCB ditempatkan secara tetap dan diberi dudukan atau perekat agar tidak bergeser. Penataan dilakukan secara sistematis untuk memudahkan proses perawatan dan menghindari hubungan pendek.

LCD 20x4 dipasang di bagian atas panel box agar informasi suhu dan data sistem bisa terbaca dari luar. Keypad 1x4 juga dipasang pada permukaan box sebagai antarmuka pengguna untuk mengatur ambang batas suhu. Pilot lamp digunakan sebagai indikator status kerja sistem, sedangkan key switch berfungsi sebagai saklar pengaman untuk mengaktifkan atau menonaktifkan sistem secara keseluruhan.

Pengelompokan kabel dilakukan dengan kabel ties agar jalur kabel rapi dan tidak saling bersilangan. Pemasangan semua komponen dalam box disusun untuk memastikan sistem dapat bekerja dengan stabil dan aman dalam kondisi lingkungan luar.

Gambar 4. 6 Perakitan Komponen Pada Box

4.3.6 Perakitan Pipa Tembaga

Perakitan pipa tembaga merupakan bagian penting dalam sistem pendingin air untuk panel surya. Pipa tembaga berfungsi sebagai media penghantar panas dari permukaan belakang panel ke air pendingin yang mengalir di dalamnya. Pipa dipilih karena memiliki konduktivitas termal yang tinggi, sehingga sangat efisien dalam menyerap panas.

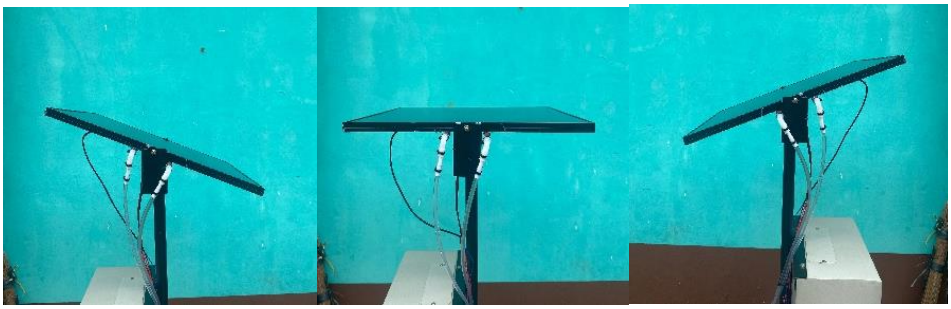
Pipa dibentuk mengikuti pola zigzag menggunakan alat pelengkung pipa agar menjangkau area yang luas di belakang panel surya. Pola ini dirancang untuk memperbesar luas permukaan kontak antara pipa dan panel agar transfer panas lebih optimal. Setelah dibentuk, pipa ditempelkan langsung ke bagian belakang panel menggunakan lem tembak dan diperkuat dengan isolasi pipa serta aluminium foil agar kontak termal lebih baik dan tahan terhadap kondisi luar ruangan.

Dua ujung pipa dihubungkan ke selang fleksibel yang akan digunakan sebagai saluran masuk dan keluar air pendingin. Jalur ini kemudian terhubung dengan pompa air dan tandon, sehingga membentuk sistem sirkulasi tertutup. Dengan perakitan ini, panas dari panel dapat diserap oleh air dan dibuang secara efisien untuk menjaga suhu panel tetap optimal.

Gambar 4. 7 Pemasangan Pipa Tembaga

4.3.7 Pemasangan Panel Surya dan Panel Box pada Kerangka

Setelah seluruh komponen sistem selesai dirakit, tahap berikutnya adalah pemasangan panel surya dan panel box pada kerangka penyangga. Panel surya dipasang di bagian atas kerangka dan diposisikan agar dapat menerima sinar matahari secara optimal. Pemasangan dilakukan dengan menggunakan baut dan mur pada kedua sisi panel untuk memastikan panel terpasang dengan stabil, namun tetap dapat diarahkan secara manual sesuai kebutuhan penyesuaian arah sinar matahari.



Gambar 4. 8 Modifikasi Pemasangan Panel Surya

Panel box dipasang di bagian bawah atau samping kerangka pada posisi yang mudah dijangkau oleh pengguna. Peletakannya mempertimbangkan kemudahan dalam jalur pengkabelan dari panel surya ke *Solar Charge Controller*, baterai, dan seluruh sistem yang dirangkai di dalam box.

Seluruh kabel disusun dengan rapi menggunakan isolasi spiral dan pengikat kabel untuk mencegah kerusakan dan gangguan sinyal. Pengelolaan kabel ini juga penting untuk menjaga keamanan sistem dari kerusakan fisik maupun gangguan lingkungan. Pemasangan keseluruhan dilakukan dengan mempertimbangkan stabilitas, kemudahan perawatan, dan keandalan sistem.

Gambar 4. 9 Pemasangan Panel Surya dan Panel Box

4.3.8 Perakitan Saluran Air Pada Sistem Pendingin

Sistem pendingin air pada alat ini dirancang untuk menjaga suhu panel surya agar tetap dalam kondisi optimal. Proses perakitan dimulai dengan pemasangan

pompa air 12V yang berfungsi sebagai pemindah air dari tandon menuju pipa tembaga yang menempel di belakang panel.

Saluran air dibuat dalam sistem sirkulasi tertutup, menggunakan selang fleksibel yang menghubungkan pompa ke ujung masuk pipa tembaga, dan selang lain dari ujung keluar pipa kembali ke tandon. Pompa akan mengalirkan air secara berkelanjutan saat suhu panel meningkat, sehingga panas dari permukaan panel dapat diserap oleh air di dalam pipa.

Untuk mengatur volume air dalam tandon, digunakan sensor ultrasonik JSN-SR04T yang diletakkan di bagian atas tandon. Sensor ini mendeteksi tinggi permukaan air dan mengaktifkan solenoid valve apabila ketinggian air melebihi batas yang telah ditentukan, sehingga air berlebih akan dibuang secara otomatis.

Aktivasi pompa dan kipas dikendalikan oleh mikrokontroler ESP32 berdasarkan data dari sensor suhu DS18B20. Jika suhu panel mencapai nilai ambang, sistem akan mengaktifkan pompa, kipas, dan peltier untuk menurunkan suhu panel surya. Dengan sistem ini, proses pendinginan berlangsung secara otomatis tanpa intervensi manual.

Gambar 4. 10 Pemasangan Pendingin Air

4.4 Langkah-Langkah Pembuatan Perangkat Lunak (*Software*)

Dalam melakukan pembuatan alat melalui beberapa langkah yaitu:

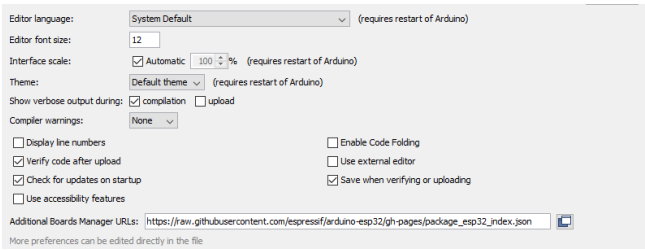
4.4.1 Pemrograman Arduino IDE

Pada tahapan pembuatan pemrograman mikrokontroler menggunakan Arduino IDE sebagai aplikasi *open source* yang dapat menjadi acuan dalam proses kerja mikrokontroler ESP32 untuk dapat menjalankan tugasnya sesuai dengan cara kerja yang diinginkan. Dalam pembuatan pemrograman dapat melalui tahap – tahap yang akan dijelaskan.

Mengunduh aplikasi Arduino IDE, kemudian membuka aplikasi Arduino IDE. Tampilan awal Arduino IDE terlampir pada gambar 4.11.

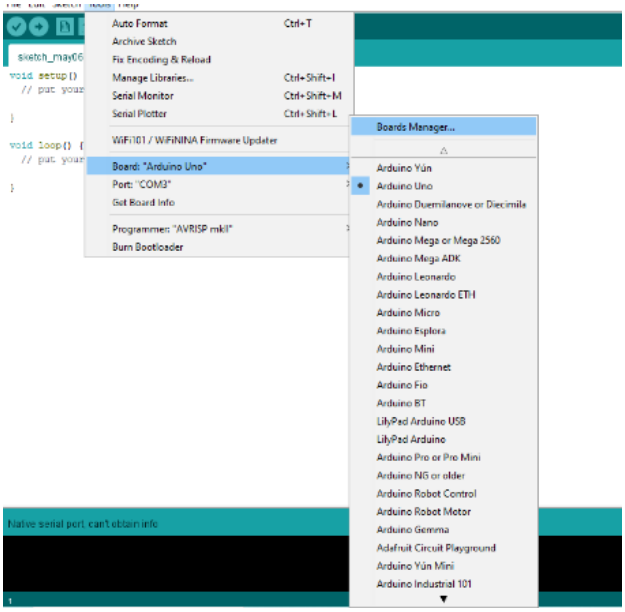
Gambar 4. 11 Tampilan Start Arduino IDE

Setelah membuka aplikasi, pada Arduino IDE terlebih dahulu dapat melakukan *install board* ESP32 sebelum melakukan penghubungan program dengan mikrokontroler. Proses dapat dilakukan dengan terlebih dahulu memasukan https://raw.githubusercontent.com/espressif/arduinoesp32/ghpages/package_esp32_index.json Pada *additional boards manager* di bagian *preference* dalam menu file terlampir pada gambar 4.12.

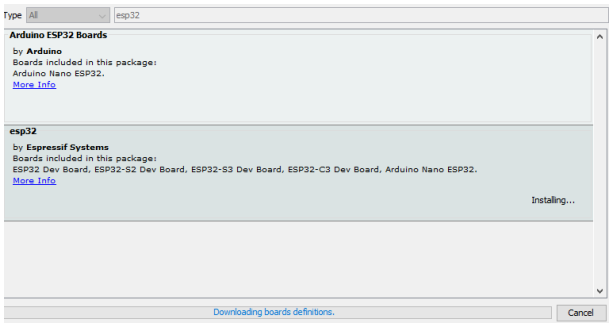


Gambar 4. 12 Menambahkan Additional Boards Manager pada Arduino IDE

Setelah menambahkan *board* ESP32 pada bagian *preference* selanjutnya dapat dilanjutkan pada bagian *boards manager* mengunduh *board* ESP32 pada menu *tools* yang ditampilkan pada gambar 4.13.



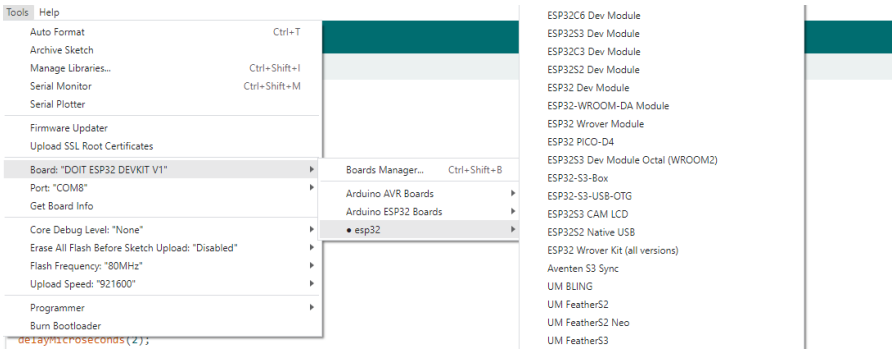
Gambar 4. 13 Melakukan Install ESP32 pada Boards Manager Arduino IDE



Gambar 4. 14 Tampilan Install ESP32 pada Arduino IDE

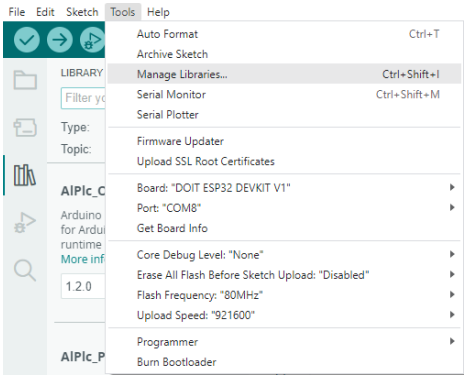
Setelah mengunduh *board* ESP32 pada Arduino IDE yang terlampir pada gambar 4.14 selanjutnya dapat menghubungkan laptop dengan mikrokontroler. Dalam penggunaan ESP32 yang penyusun gunakan, mikrokontroler tersebut dilengkapi dengan tipe *type*. Sehingga, kabel konektor yang penyusun gunakan berupa kabel usb yang akan dihubungkan ke laptop dan *type C* yang akan dihubungkan ke mikrokontroler ESP32.

Setelah terhubung, dapat mengatur *board* menggunakan ESP32 pada Arduino IDE yang ditunjukkan pada gambar 4.15.

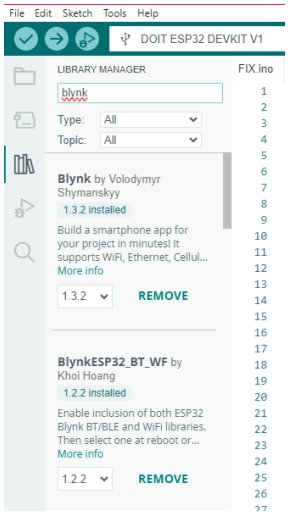


Gambar 4. 15 ESP32 sebagai Board pada Arduino IDE

Setelah mengatur *board* pada Arduino IDE, dapat mengunduh *library* yang akan digunakan terlampir pada gambar 4.16 dan gambar 4.17, dalam alat ini penyusun menggunakan *library* LCD, pengukuran pembacaan sensor, penggunaan Wi-Fi dan *Blynk*.



Gambar 4. 16 Menentukan Pengunduhan *Library* Arduino IDE



Gambar 4. 17 Pengunduhan *Library* Arduino IDE

Ketika sudah melakukan pengunduhan *library* sesuai dengan *library* yang akan digunakan, *library* akan dapat digunakan sesuai fungsinya dan tertampil pada tampilan Arduino IDE seperti pada gambar 4.18.

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <ezButton.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Preferences.h>
#include <WiFiManager.h>
#include "ACS712.h"
#include <BlynkSimpleEsp32.h>
#include <HTTPClient.h>
```

Gambar 4. 18 Tampilan Penggunaan Library pada Arduino IDE

Setelah penggunaan *library* pada Arduino IDE telah dilakukan, dapat dilanjutkan dengan penentuan pin dalam melakukan deklarasi nilai variable yang akan digunakan pada mikrokontroler untuk masing – masing sensor atau relay yang terhubung.

Pada gambar 4.19 menunjukkan penentuan *pin* modul pada Arduino IDE serta pada gambar 4.20 menampilkan penggunaan *pin* yang akan dihubungkan dengan *datastreams* aplikasi *blynk*.

```
#define ONE_WIRE_BUS 4 // sensor DS18B20
#define RELAY_POMPA 13
#define RELAY_KIPAS 15
#define RELAY_LAMPUON 14
#define RELAY_SOLENOID 17
#define PIN_TEGANGAN 34  // Sensor Tegangan
#define PIN_ARUS 36 // Sensor Arus
#define RELAY_GREENLAMP 32
#define KEY_NUM 4  // Jumlah tombol
#define PIN_KEY_1 33
#define PIN_KEY_2 27
#define PIN_KEY_3 26
#define PIN_KEY_4 25
#define echoPin 18 // Pin Echo sensor JSN-SR04
#define trigPin 5  // Pin Trigger sensor JSN-SR04
```

Gambar 4. 19 Pin Modul pada Arduino IDE

Pada bagian ini dilakukan inisialisasi variabel bool dan float sebagai pengaturan awal sistem. Nilai-nilai tersebut digunakan untuk mengatur kondisi logika, status menu, kontrol aktuator, serta parameter awal pembacaan sensor agar sistem dapat berjalan sesuai fungsinya sejak pertama dijalankan.

```
LiquidCrystal_I2C lcd(0x27, 20, 4);
bool inMenu = false;
bool menuActive = false;
float batasSuhu = 38.0;
Preferences preferences;
bool pompaManual = false;
bool kipasManual = false;
bool solenoidManual = false;
bool sudahKirimNotif = false;
bool sudahKirimNotifAir = false;
bool sudahKirimNotifSolenoid = false;

WiFiManager wifiManager; // Objek WiFiManager
ACS712 ACS(PIN_ARUS, 3.3, 4095, 185);  // Pin 36, Vref 3.3V, resolusi 4095, sensitivitas 100 mV/A
float tegangan_ref = 3.3;   // Tegangan referensi ESP32
int resolusi_adc = 4095;     // Resolusi ADC ESP32
float faktor_skala = 12.0 / 2861.0;  // Faktor skala berdasarkan ADC 2861 = 12V DC
float offset_arus = 0.00;
float daya = 0.0;
float arus = 0.0;
float tinggiAir = 0.0;
float suhu = 0.0;
unsigned long lastUpdate = 0;
const int updateInterval = 1000;  // 1 detik
```

Gambar 4. 20 Inisialisasi nilai awal variabel bool dan float pada sistem berbasis ESP32.

Bagian ini berisi deklarasi fungsi-fungsi utama seperti tampilan layar, pembacaan sensor (suhu, tegangan, arus, tinggi air), serta pemrosesan input tombol. Fungsi dibuat terpisah agar program lebih rapi dan setiap tugas dapat dijalankan sesuai perintah.

```
void displayStartupScreen();
void displayDefaultScreen();
void displayMenuScreen();
void displaySaveConfirmation();
void updateTemperature();
void updateVoltage();
void updateCurrent();
void updateDaya(float tegangan, float arus);
void updateTinggiAir();
int getKeyPressed();
```

Gambar 4. 21 Daftar fungsi utama dalam program.

Selanjutnya melakukan pemrograman *void setup()* pada Arduino IDE yang tertampil pada gambar 4.22.

```
void setup() {
    lcd.init();
    lcd.backlight();
    sensors.begin();

    pinMode(RELAY_POMPA, OUTPUT);
    pinMode(RELAY_KIPAS, OUTPUT);
    pinMode(RELAY_LAMPUPOMPA, OUTPUT);
    pinMode(RELAY_LAMPUON, OUTPUT);
    pinMode(RELAY_GREENLAMP, OUTPUT);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);

    digitalWrite(RELAY_POMPA, HIGH);
    digitalWrite(RELAY_KIPAS, HIGH);
    digitalWrite(RELAY_LAMPUON, HIGH);  //(lampu merah)
    digitalWrite(RELAY_GREENLAMP, HIGH);  //(lampu hijau)
    digitalWrite(RELAY_LAMPUPOMPA,HIGH);

    ACS.autoMidPoint();  // Kalibrasi titik tengah otomatis

    for (byte i = 0; i < KEY_NUM; i++) {
        keypad_1x4[i].setDebounceTime(100);
    }

     // Inisialisasi Preferences
    preferences.begin("config", false);
    batasSuhu = preferences.getFloat("batasSuhu", 38.0); // Ambil nilai tersimpan
    preferences.end();

    displayStartupScreen();
```

```
  // Tampilan menunggu WiFi
  lcd.setCursor(0, 0);
  lcd.print("Waiting for WiFi");

  // Koneksi WiFi menggunakan WiFiManager
  wifiManager.autoConnect("Tugas Akhir Naiya Kartika A.", "12345678");

  delay(2000);

  Blynk.config(BLYNK_AUTH_TOKEN);
  Blynk.connect();
  sendTelegramMessage("📢 Alat Monitoring dan Optimasi Output Panel Surya ⚡\n🎓 (Tugas Akhir Naiya Kartika A.)\n✅ Aktif!");

    // Pastikan relay tidak berubah setelah WiFi terhubung
  delay(500);
  digitalWrite(RELAY_POMPA, HIGH);
  digitalWrite(RELAY_KIPAS, HIGH);
  digitalWrite(RELAY_LAMPUPOMPA, HIGH);
  digitalWrite(RELAY_LAMPUON, HIGH);
  digitalWrite(RELAY_GREENLAMP, HIGH);

  // Jika berhasil terkoneksi
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("WiFi Connected");
  delay(2000);
  lcd.clear();

displayDefaultScreen();
}
```

Gambar 4. 22 Tampilan *Void Setup* Arduino IDE

Selanjutnya mengatur pada voidloop() sebagai proses pemrograman yang akan bekerja secara terus menerus yang telah disesuai dengan perintah, pemrograman ini tertampil pada gambar 4.23

```
void loop() {
    int key = getKeyPressed(); // Baca tombol keypad yang ditekan
    Blynk.run(); // Jalankan Blynk

    int nilai_adc = analogRead(PIN_TEGANGAN);
    float tegangan = nilai_adc * faktor_skala;

    // Cek apakah tombol ditekan
    if (key == 1) {
        if (inMenu) {
            inMenu = false;
            menuActive = false;
            lcd.clear();
            displayDefaultScreen(); // Kembali ke layar utama
        } else {
            inMenu = true;
            menuActive = true;
            lcd.clear();
            displayMenuScreen(); // Masuk ke menu
        }
    }

    // Jika sedang berada di dalam menu, tangani input keypad
    if (inMenu) {
        if (key == 2) {
            batasSuhu += 0.5;
            displayMenuScreen(); // Perbarui tampilan LCD
        }
        if (key == 3) {
            batasSuhu -= 0.5;
            displayMenuScreen(); // Perbarui tampilan LCD
        }
```

```
        }
        if (key == 4) {
            // Simpan batas suhu ke memori
            preferences.begin("config", false);
            preferences.putFloat("batasSuhu", batasSuhu);
            preferences.end();

            lcd.clear();
            displaySaveConfirmation(); // Tampilkan konfirmasi penyimpanan

            // Kirim notifikasi Telegram
            String pesan = "🔔 *Batas Suhu Baru Disimpan!* 🔔\n";
            pesan += "📌 Batas Suhu: " + String(batasSuhu, 1) + " °C\n";
            sendTelegramMessage(pesan);

            delay(2000);
            inMenu = false;
            menuActive = false;
            lcd.clear();
            displayDefaultScreen(); // Kembali ke layar utama
        }
    }

    // Jika tidak dalam mode menu, perbarui sensor dan kirim data ke Blynk
    if (!inMenu) {
        updateTemperature();
        updateVoltage();
        updateCurrent();
        updateTinggiAir();
```

```
    // Jika tidak dalam mode menu, perbarui sensor dan kirim data ke Blynk
    if (!inMenu) {
        updateTemperature();
        updateVoltage();
        updateCurrent();
        updateTinggiAir();

        // Pastikan variabel arus dan daya sudah terdefinisi sebelumnya
        updateDaya(tegangan, arus);

        // Kirim data ke Blynk Virtual Pin
        Blynk.virtualWrite(V0, tegangan);  // Tegangan Panel (V)
        Blynk.virtualWrite(V1, arus);      // Arus Panel (A)
        Blynk.virtualWrite(V2, daya);      // Daya Panel (W)
        Blynk.virtualWrite(V3, tinggiPersen); // Tinggi Air (%)
        Blynk.virtualWrite(V7, suhu);      // Suhu (°C)
    }
}
```

Gambar 4. 23 Tampilan *Void Loop* pada Arduino IDE

Pada gambar 4.24 menunjukkan pemrograman yang akan digunakan dalam melakukan hubungan koneksi antara Arduino IDE dengan aplikasi *blynk.*

```
#define BLYNK_TEMPLATE_ID "TMPL6v7rbvLq0"
#define BLYNK_TEMPLATE_NAME "Tugas Akhir Naiya Kartika A"
#define BLYNK_AUTH_TOKEN "qguoldCbjvkeFZDtHVDeWx0CcxfaXr2F"
```

Gambar 4. 24 Pemrograman Koneksi ESP32 dengan Aplikasi Blynk

```
BLYNK_WRITE(V4) {
    int pompaState = param.asInt();
    digitalWrite(RELAY_POMPA, pompaState ? LOW : HIGH);
    pompaManual = (pompaState == 1);  // Hanya manual jika ON
}

BLYNK_WRITE(V5) {
    int kipasState = param.asInt();
    digitalWrite(RELAY_KIPAS, kipasState ? LOW : HIGH);
    kipasManual = (kipasState == 1);
}

BLYNK_WRITE(V6) {
    int solenoidState = param.asInt();
    digitalWrite(RELAY_LAMPUPOMPA, solenoidState ? LOW : HIGH);
    solenoidManual = (solenoidState == 1);
}
```

Gambar 4. 25 Coding Controlling Blynk

Pada tahap ini, fungsi updateTemperature() digunakan untuk memperbarui nilai suhu yang dibaca dari sensor. Nilai suhu tersebut kemudian dikirimkan ke *platform* Blynk dan Telegram sebagai notifikasi. Selain itu, jika suhu melebihi nilai setpoint yang telah ditentukan, maka sistem akan secara otomatis mengaktifkan pendingin, yang terdiri dari pompa, kipas, dan solenoid valve**.**

```
void updateTemperature() {
  sensors.requestTemperatures();
  suhu = sensors.getTempCByIndex(0);

    // Tangani error -127 (sensor tidak terbaca)
  if (suhu == -127.0) {
    suhu = 0.0;  // Atur menjadi 0 jika error
  }

  lcd.setCursor(15, 0);
  lcd.print("    ");
  lcd.setCursor(15, 0);
  lcd.print(suhu, 1);

  if (suhu > batasSuhu) {
    if (!sudahKirimNotif) {
      sendTelegramMessage("⚠️ *Peringatan!* Suhu panel melebihi batas!\n🌡️ Suhu Saat Ini: " + String(suhu, 1) + " °C\n📌 Batas Suhu: " + String(batasSuhu,
      sudahKirimNotif = true;
    }
    digitalWrite(RELAY_LAMPUON, LOW);
    if (!pompaManual) digitalWrite(RELAY_POMPA, LOW);  // Nyalakan pompa
    if (!kipasManual) digitalWrite(RELAY_KIPAS, HIGH);  // Nyalakan kipas
    if (!solenoidManual) digitalWrite(RELAY_SOLENOID, LOW);  // Nyalakan solenoid
  } else {
    if (sudahKirimNotif) {
      sendTelegramMessage("✅ *Status Normal!* Suhu panel kembali normal.\n❄️ Mematikan sistem pendingin.\n🌡️ Suhu Saat Ini: " + String(suhu, 1) + " °C\n📌
      sudahKirimNotif = false;
    }
    digitalWrite(RELAY_LAMPUON, HIGH);
    if (!pompaManual) digitalWrite(RELAY_POMPA, HIGH);  // Matikan pompa
    if (!kipasManual) digitalWrite(RELAY_KIPAS, HIGH);  // Matikan kipas
    if (!solenoidManual) digitalWrite(RELAY_SOLENOID, HIGH);  // Matikan solenoid
  }
```

Gambar 4. 26 Potongan kode fungsi updateTemperature() pada ESP32.

```
void updateVoltage() {
    int nilai_adc = analogRead(PIN_TEGANGAN);
    float tegangan = nilai_adc * faktor_skala;

    lcd.setCursor(3, 0);
    lcd.print("    "); // Hapus angka sebelumnya
    lcd.setCursor(3, 0);
    lcd.print(tegangan, 1); // Tampilkan tegangan dengan 2 angka di belakang koma
}
```

Gambar 4. 27 Void Update Nilai Tegangan dari Sensor Tegangan 25VDC dan Pengaturan Tampilan di LCD

```
void updateCurrent() {
  float offsetTegangan = 2.390; // offset hasil kalibrasi
  float sensitivitas = 0.185;   // untuk ACS712-5A, dalam V/A
  const int samples = 100;
  float teganganADC = 0;

  for (int i = 0; i < samples; i++) {
    int adc = analogRead(PIN_ARUS);
    teganganADC += (adc * 3.3) / 4095.0;
    delay(2);
  }
  teganganADC /= samples;

  arus = abs(teganganADC - offsetTegangan) / sensitivitas;

  // Hilangkan noise di bawah 0.1A
  if (arus < 0.1) arus = 0.0;

  Serial.print("ADC rata-rata: "); Serial.print(teganganADC, 3);
  Serial.print(" V | Arus: "); Serial.print(arus, 3); Serial.println(" A");

  // Tampilkan ke LCD (baris 1 adalah index ke-0)
  lcd.setCursor(13, 1);
  lcd.print("     "); // hapus nilai lama
  lcd.setCursor(13, 1);
  lcd.print(arus, 2);
}
```

Gambar 4. 28 Void Update Nilai Arus dari Sensor ACS712 dan Pengaturan Tampilan di LCD

```
void updateDaya(float tegangan, float arus) {
    daya = tegangan * arus;  // Hitung daya
    lcd.setCursor(13, 2);
    lcd.print("    ");  // Hapus nilai lama
    lcd.setCursor(13, 2);
    lcd.print(daya, 1);
}
```

Gambar 4. 29 Void update nilai daya dan settingan tampilan di LCD

```
void displayStartupScreen() {
    lcd.setCursor(0, 0);
    lcd.print("  NAIYA KARTIKA A.  ");
    lcd.setCursor(0, 1);
    lcd.print("   40040621650051   ");
    lcd.setCursor(0, 2);
    lcd.print(" MONITOR & OPTIMASI ");
    lcd.setCursor(0, 3);
    lcd.print(" OUTPUT PANEL SURYA ");
}

void displayDefaultScreen() {
    lcd.setCursor(0, 0);
    lcd.print("V=     V Suhu=     C");
    lcd.setCursor(0, 1);
    lcd.print("Iout Panel =       A");
    lcd.setCursor(0, 2);
    lcd.print("Pout Panel =       W");
    lcd.setCursor(0, 3);
    lcd.print("Tinggi Air =       %");
}
```

Gambar 4. 30 Void tampilan Awal dan Utama LCD

```
void displayMenuScreen() {
    lcd.setCursor(0, 0);
    lcd.print("  Batas Suhu Panel=  ");
    lcd.setCursor(6, 1);
    lcd.print("      ");
    lcd.setCursor(6, 1);
    lcd.print(batasSuhu);
    lcd.setCursor(12, 1);
    lcd.print(" C   ");
    lcd.setCursor(0, 2);
    lcd.print("1 = Back    4 = Save");
    lcd.setCursor(0, 3);
    lcd.print("2 = Up      3 = Down");
}

void displaySaveConfirmation() {
    lcd.setCursor(0, 0);
    lcd.print("     Nilai Baru     ");
    lcd.setCursor(0, 1);
    lcd.print("  Batas Suhu Panel  ");
    lcd.setCursor(0, 2);
    lcd.print("     Tersimpan!     ");
    lcd.setCursor(6, 3);
    lcd.print(batasSuhu);
    lcd.setCursor(12, 3);
    lcd.print(" C   ");
}
```

Gambar 4. 31 Void Tampilan Setting Nilai Setpoint Suhu dan Penyimpanan Setpoint Suhu Berhasil

```
void updateTinggiAir() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.0344 / 2;

  // Perhitungan persentase berdasarkan jarak 20cm = 100%, 34cm = 0%
  tinggiPersen = ((34.0 - distance) / (34.0 - 20.0)) * 100.0;
  tinggiPersen = constrain(tinggiPersen, 0, 100); // Batasi agar tetap 0–100%

  // Tampilkan di LCD
  lcd.setCursor(13, 3);
  lcd.print("     ");  // Hapus angka lama
  lcd.setCursor(13, 3);
  lcd.print(tinggiPersen, 1);
  // Kirim notifikasi jika di bawah 35%
  if (tinggiPersen < 35.0) {
    if (!sudahKirimNotifAir) {
      String pesan = "⚠️ *Peringatan!* Tinggi air sangat rendah.\n💧 Tinggi saat ini: " + String(tinggiPersen, 1) + " %\n📉 Minimum aman: 35 %";
      sendTelegramMessage(pesan);
      sudahKirimNotifAir = true;
    }
  } else {
    // Jika sudah kembali normal dan notifikasi sebelumnya pernah dikirim
    if (sudahKirimNotifAir) {
      String pesan = "✅ *Status Normal!* Tinggi air kembali di atas ambang.\n💧 Tinggi saat ini: " + String(tinggiPersen, 1) + " %";
      sendTelegramMessage(pesan);
      sudahKirimNotifAir = false;
    }
  }
  // ✅ Tambahan logika kontrol solenoid dan notifikasi Telegram
  if (!solenoidManual) {
    if (tinggiPersen > 90.0) {
      digitalWrite(RELAY_SOLENOID, LOW);  // Buka solenoid

      if (!sudahKirimNotifSolenoid) {
        String pesan = "⚠️ *Peringatan!* Tinggi air melebihi 90%.\n💧 Tinggi saat ini: " + String(tinggiPersen, 1) + " %\n🔒 Solenoid otomatis terbuka untuk menga
        sendTelegramMessage(pesan);
        sudahKirimNotifSolenoid = true;
      }
    } else {
      digitalWrite(RELAY_SOLENOID, HIGH); // Tutup solenoid

      if (sudahKirimNotifSolenoid) {
        String pesan = "✅ *Status Normal!* Tinggi air kembali di bawah ambang.\n💧 Tinggi saat ini: " + String(tinggiPersen, 1) + " %\n🔒 Solenoid otomatis tertu
        sendTelegramMessage(pesan);
        sudahKirimNotifSolenoid = false;
      }
    }
  }
}
```

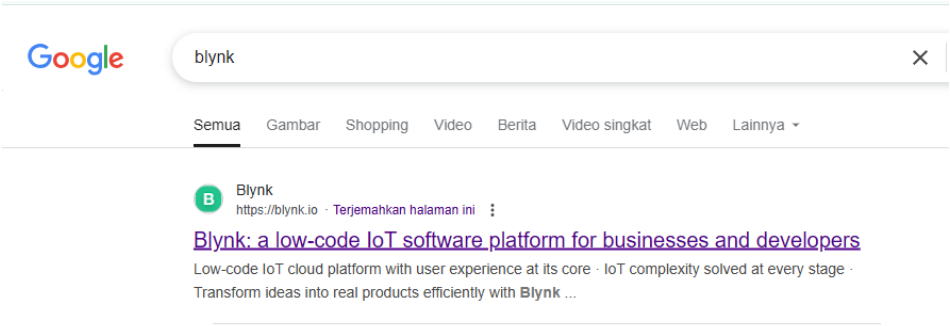
Gambar 4. 32 Void Pembacaan Sensor JSN-SR04T dengan Notifikasi Kondisi pada Telegram

```
void sendTelegramMessage(String message) {
  for (int i = 0; i < jumlahChat; i++) {
    HTTPClient http;
    String url = "https://api.telegram.org/bot" + String(telegramToken) + "/sendMessage";
    String payload = "chat_id=" + String(chatIDs[i]) + "&text=" + message + "&parse_mode=Markdown";

    http.begin(url);
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");

    int httpResponseCode = http.POST(payload);
    if (httpResponseCode > 0) {
      Serial.println("Pesan terkirim ke Telegram ID " + String(chatIDs[i]));
    } else {
      Serial.print("Error kirim ke ID "); Serial.print(chatIDs[i]);
      Serial.print(": "); Serial.println(httpResponseCode);
    }

    http.end();
    delay(500);  // Delay kecil untuk mencegah overload server Telegram
  }
}
```

Gambar 4. 33 Void pengiriman notifikasi ke Telegram dari ESP32 melalui jaringan Wi-Fi

4.4.2 Pembuatan Blynk

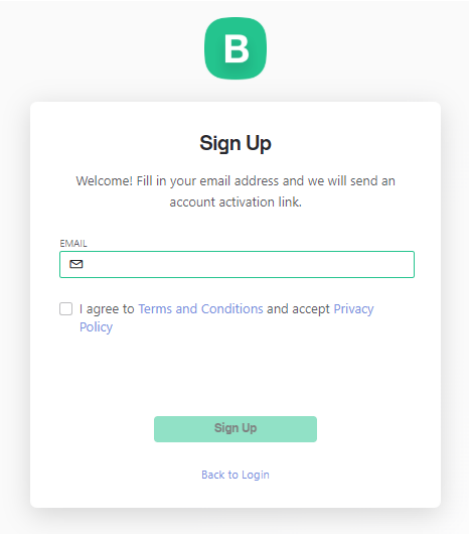
Pada tahap ini, pengguna mencari informasi mengenai Blynk untuk memulai proses integrasi sistem IoT. Hasil pencarian menampilkan situs resmi Blynk (https://blynk.io), yang merupakan *platform low-code* berbasis cloud yang dirancang untuk mempermudah pengembangan proyek berbasis *Internet of Things* (IoT). Melalui situs ini, pengguna dapat membuat akun, mengakses dokumentasi, serta memulai pembuatan proyek IoT seperti monitoring sistem panel surya yang terhubung dengan ESP32.



Gambar 4. 34 Tahap Awal Situs Resmi Blynk Melalui Pencarian Google

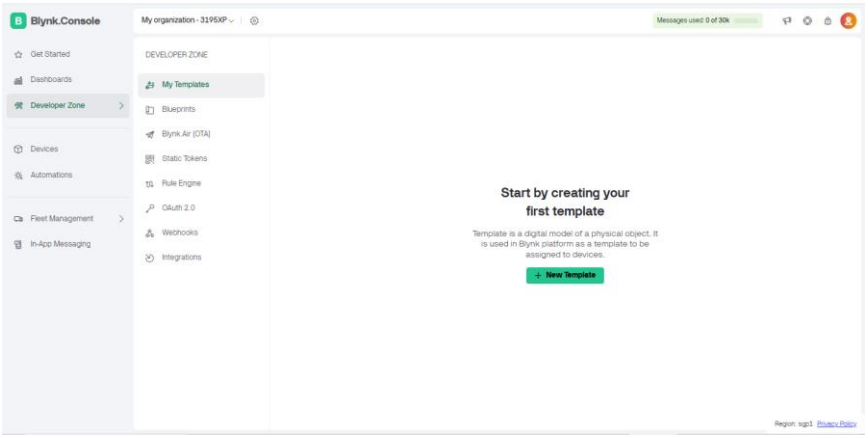
Penggunaan aplikasi Blynk dimulai dengan membuat akun menggunakan alamat email yang dimiliki. Ilustrasi proses pembuatan akun ditunjukkan pada

Gambar 4.35 Setelah itu, langkah selanjutnya adalah membuat datastreams, sebagaimana ditampilkan pada Gambar 4.36.



Gambar 4. 35 Tampilan Aplikasi *Sign Up Blynk*

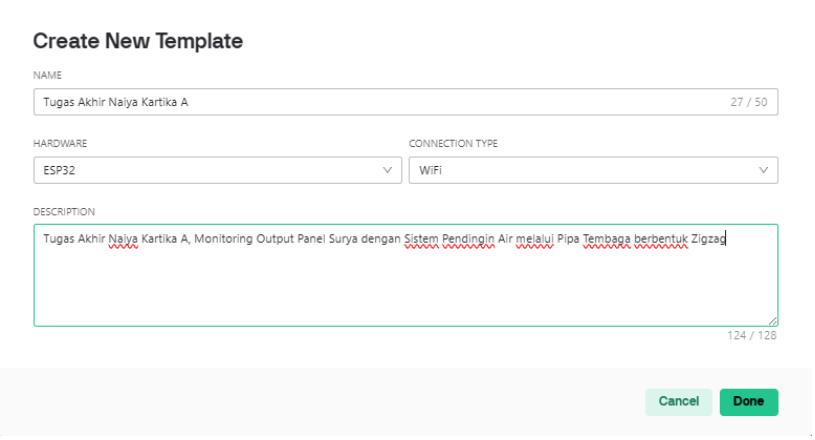
Selanjutnya, pembuatan *Datastreams* pada *template* baru di aplikasi Blynk. Tahapan ini bertujuan untuk menghubungkan data dari mikrokontroler ESP32 ke widget yang ada di dashboard, seperti suhu, tegangan, arus, dan status pompa. Setiap *Datastream* akan dikaitkan dengan *Virtual Pin* tertentu agar data dapat ditampilkan dan dikontrol secara *real-time* melalui aplikasi.



Gambar 4. 36 Pembuatan *Datastreams* Aplikasi *Blynk*

Pada tahap ini dilakukan pembuatan *Datastreams* setelah berhasil membuat *template* dengan nama “Tugas Akhir Naiya Kartika A”. Pembuatan *Datastreams* bertujuan untuk menghubungkan data sensor dari ESP32 ke aplikasi Blynk. Setiap parameter seperti suhu, tegangan, arus, dan status pompa akan dikonfigurasikan ke

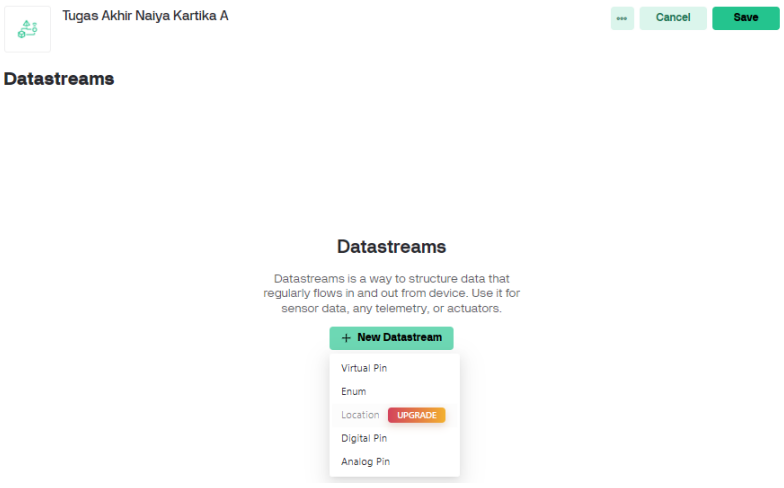
dalam *Virtual Pin* yang sesuai, agar data dapat ditampilkan dan dikontrol secara *real-time* melalui dashboard Blynk.



Gambar 4. 37 Tampilan Pembuatan Datastreams pada Template Baru

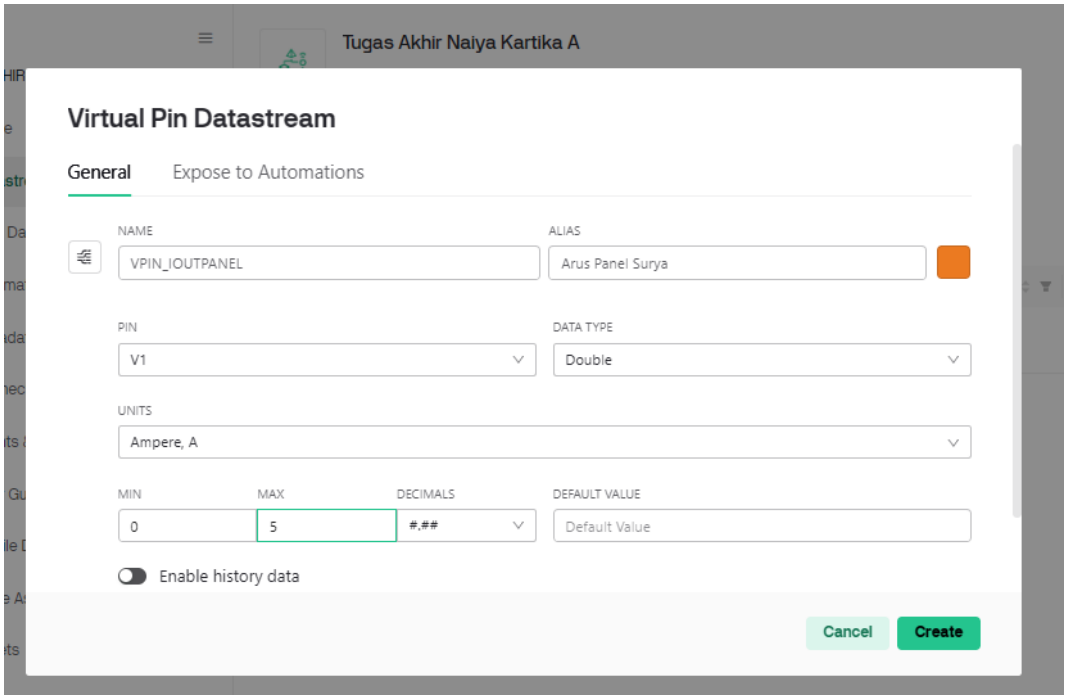
Setelah template berhasil dibuat, tahap berikutnya adalah menambahkan datastream untuk mengatur aliran data antara perangkat ESP32 dan aplikasi Blynk. Pada tahap ini, pengguna memilih jenis datastream yang sesuai dengan jenis data yang akan dikirim, seperti suhu, arus, tegangan, atau kontrol pompa. Berikut langkah-langkahnya:

1. Klik tombol "+ New Datastream".
2. Pilih opsi Virtual Pin, karena ESP32 akan berkomunikasi dengan Blynk melalui pin virtual untuk setiap parameter.
3. Setelah itu, kamu dapat mengisi konfigurasi seperti nama, tipe data (misalnya integer, float), dan nomor Virtual Pin (misalnya V0, V1, dst).



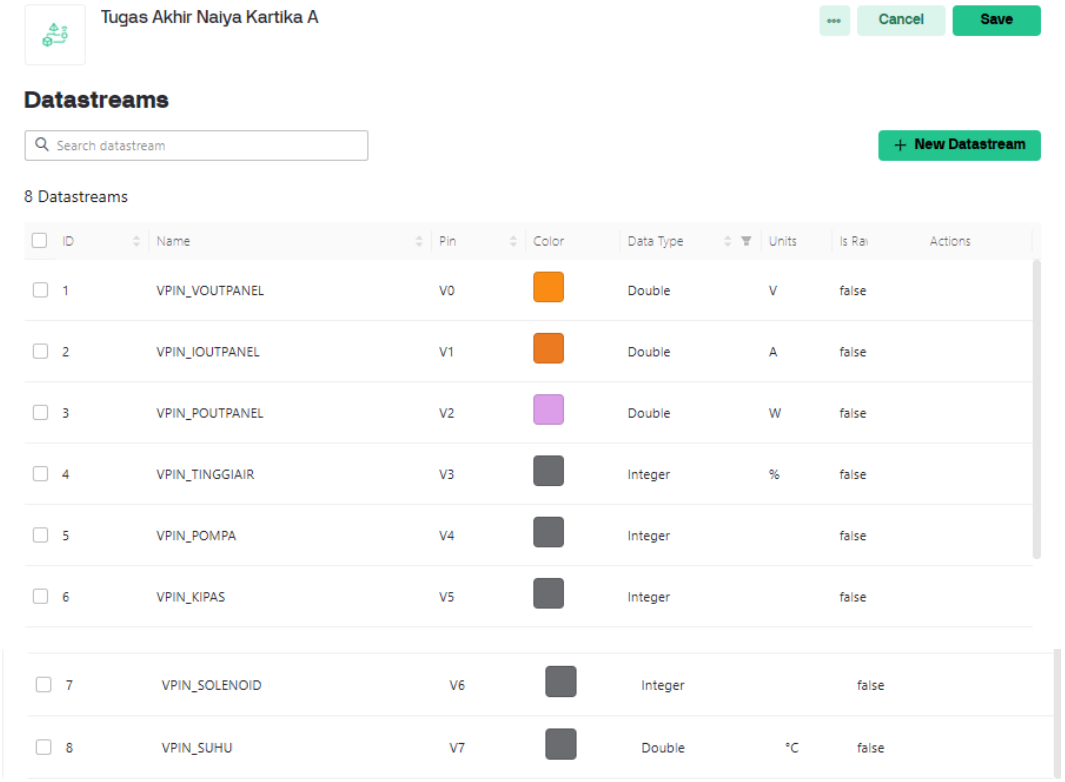
Gambar 4. 38 Pembuatan Datastream di Aplikasi Blynk

Pada tahap ini, dilakukan konfigurasi *Datastream* untuk menghubungkan data arus keluaran panel surya ke aplikasi Blynk. Jenis *Datastream* yang digunakan adalah Virtual Pin, karena komunikasi data antara ESP32 dan Blynk menggunakan jalur pin virtual.



Gambar 4. 39 Konfigurasi Virtual Pin Datastream

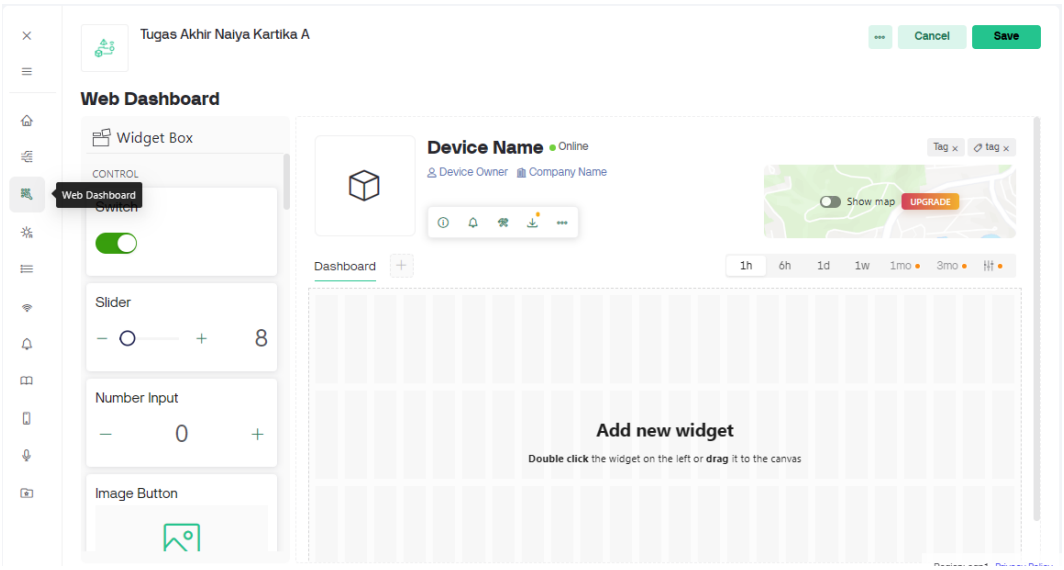
Gambar ini menunjukkan semua *datastream* yang telah dikonfigurasi untuk mendukung pemantauan dan kontrol sistem. Setiap virtual pin dikaitkan dengan satu parameter penting seperti suhu, arus, tegangan, tinggi air, daya keluaran, dan kontrol perangkat seperti pompa, kipas, serta solenoid valve. Konfigurasi ini menjadi dasar dalam membangun dashboard monitoring IoT berbasis ESP32.



Gambar 4. 40 Pengaturan Datastreams di Aplikasi Blynk

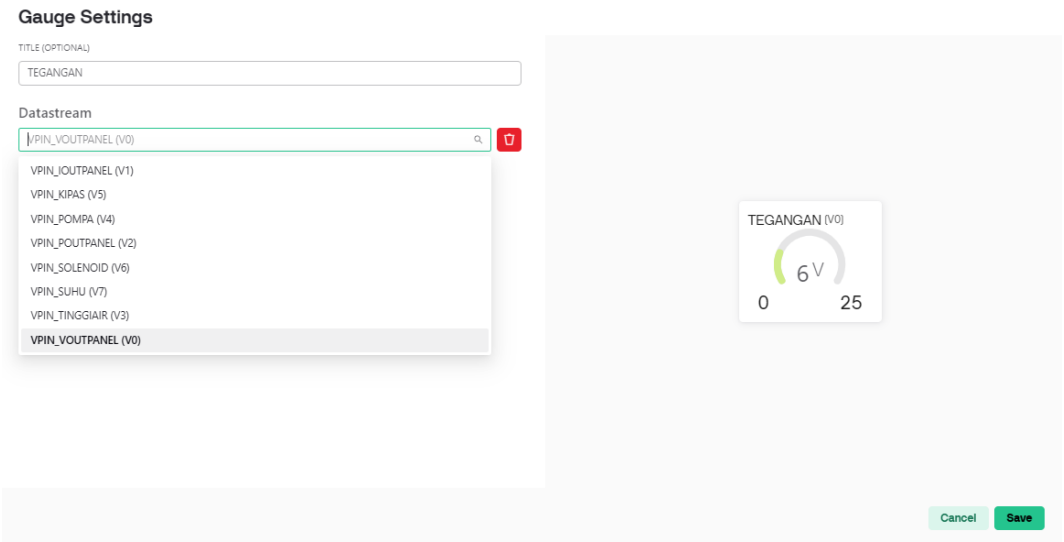
Tahap selanjutnya adalah melakukan pengaturan widget pada Web Dashboard Blynk. Setelah semua datastreams selesai dikonfigurasi, pengguna dapat mulai menambahkan dan menata widget seperti slider, switch, number input, dan lainnya ke dalam kanvas dashboard. Setiap widget nantinya akan dihubungkan ke salah satu virtual pin (V0–V7) sesuai fungsinya.

Melalui tahap ini, tampilan antarmuka monitoring dan kontrol mulai dibentuk, sehingga pengguna bisa melihat data seperti suhu, tegangan, arus, serta mengaktifkan atau menonaktifkan pompa, kipas, atau solenoid secara *real-time* melalui dashboard.



Gambar 4. 41 Pengaturan Datastreams untuk Widget pada Dashboard

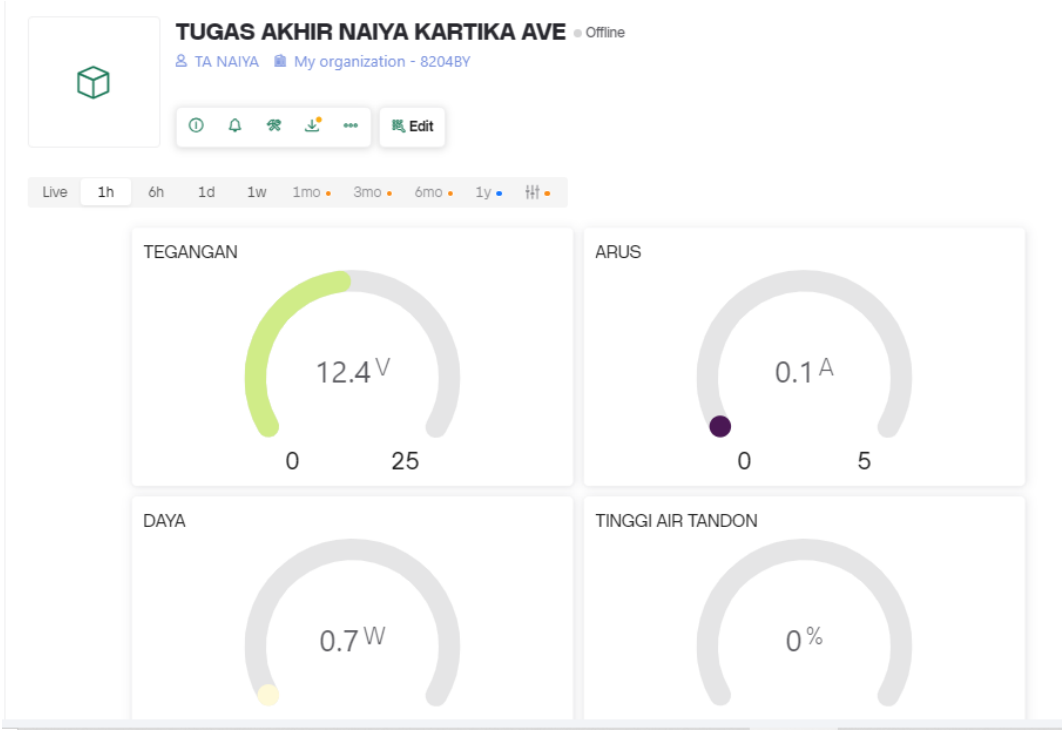
Selanjutnya, tahap pengaturan widget Gauge pada aplikasi Blynk, yaitu proses menghubungkan tampilan visual (gauge) dengan datastream VPIN_VOUTPANEL (V0) untuk menampilkan nilai tegangan dari sensor secara *real-time*. Pengguna menetapkan rentang nilai 0–25 Volt dan menamai tampilan dengan "TEGANGAN". Langkah ini merupakan bagian dari pembuatan antarmuka monitoring pada sistem IoT.



Gambar 4. 42 Pengaturan Widget Gauge pada Aplikasi Blynk

Tahap ini merupakan bagian akhir dari proses pembuatan sistem monitoring berbasis IoT, yaitu pembuatan tampilan dashboard pada aplikasi Blynk. Setelah

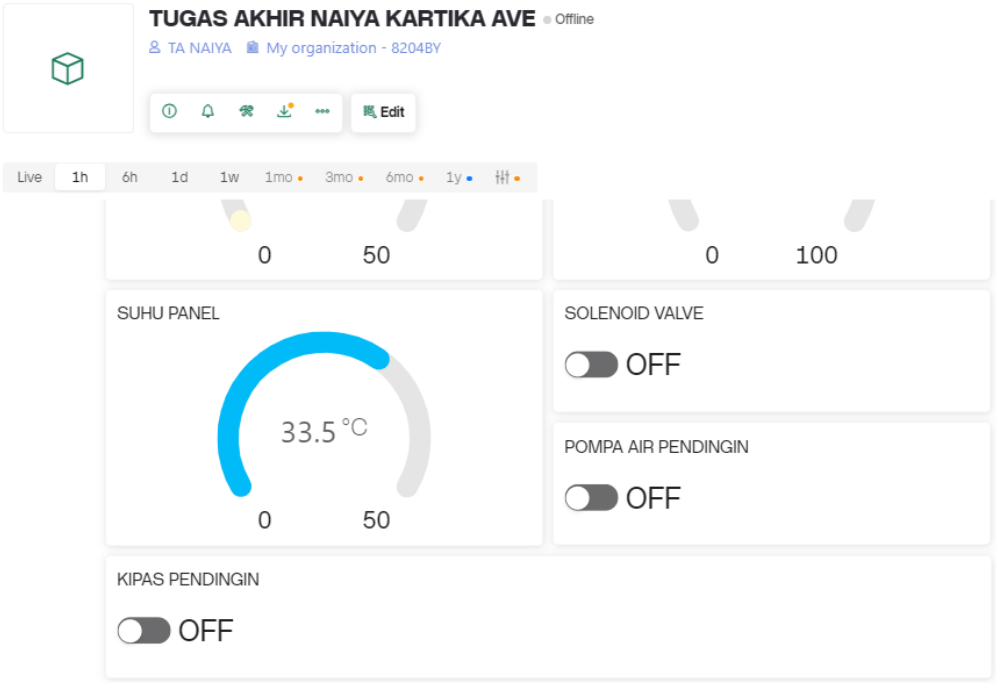
seluruh datastream dikonfigurasikan dan terhubung dengan perangkat ESP32, langkah selanjutnya adalah membangun antarmuka pemantauan yang akan digunakan untuk melihat dan mengendalikan parameter-parameter penting pada sistem. Dashboard dirancang agar dapat menampilkan data secara *real-time* melalui berbagai widget yang tersedia.



Gambar 4. 43 Tampilan Dashboard Blynk

Pada tampilan dashboard ditampilkan informasi seperti tegangan, arus, dan daya keluaran dari panel surya menggunakan widget gauge, serta informasi tinggi air pada tandon dalam bentuk persen. Selain itu, suhu permukaan panel surya juga ditampilkan dalam bentuk indikator melingkar. Untuk bagian pengendalian, terdapat tombol saklar (switch) yang digunakan untuk mengaktifkan atau menonaktifkan komponen seperti solenoid valve, pompa air pendingin, dan kipas pendingin. Semua komponen tersebut terhubung menggunakan virtual pin yang sesuai sehingga nilai sensor dapat dibaca dan aktuator dapat dikontrol secara sinkron melalui jaringan internet.

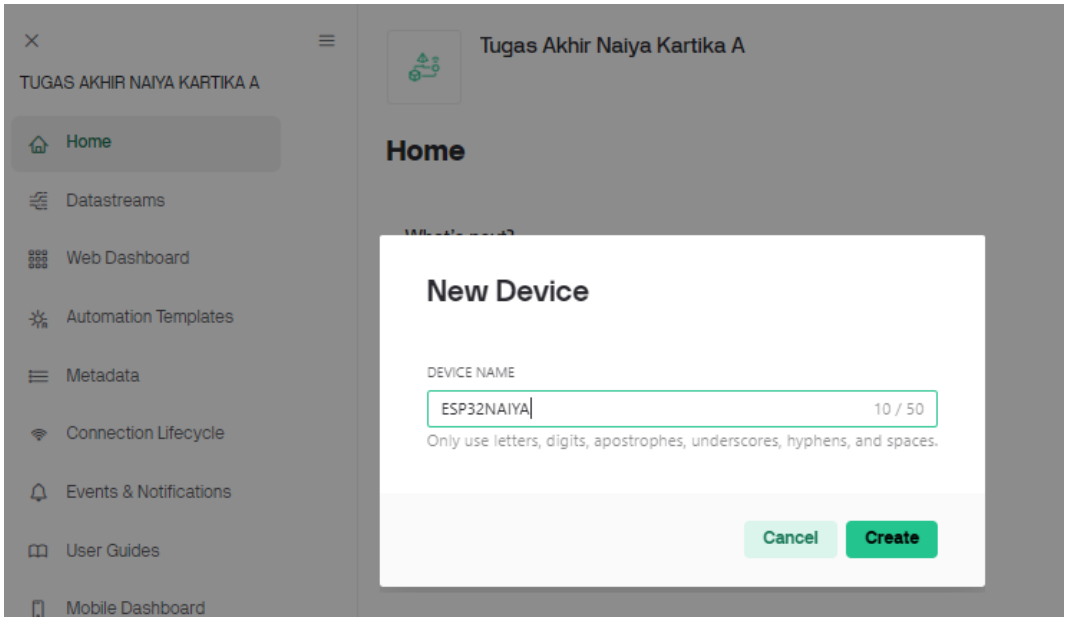
Dengan adanya dashboard ini, pengguna dapat memantau kinerja panel surya dan sistem pendinginnya dari jarak jauh secara praktis dan efisien melalui aplikasi Blynk.



Gambar 4. 44 Tampilan Lanjutan Dashboard Blynk

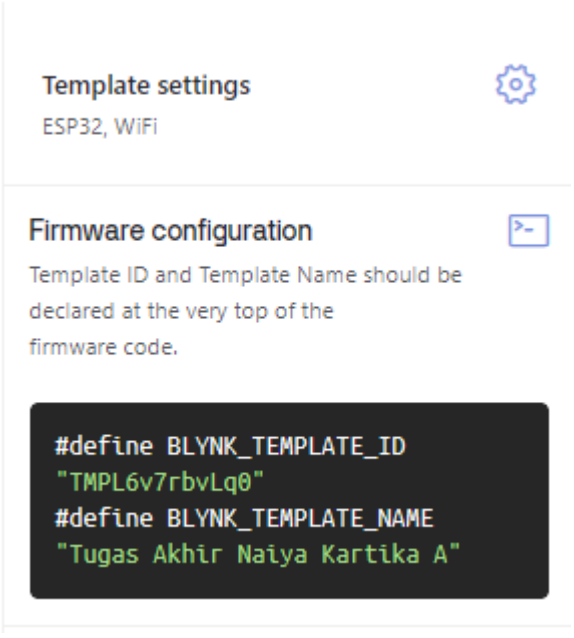
Gambar ini menunjukkan tahap penambahan perangkat baru (device) di aplikasi Blynk. Pada proses ini, pengguna memberi nama perangkat, dalam hal ini "ESP32NAIYA", sebagai identitas dari mikrokontroler yang akan digunakan dalam sistem IoT.

Tahap ini dilakukan setelah template selesai dibuat, dan bertujuan untuk menghubungkan antara proyek Blynk dengan perangkat fisik (ESP32). Setelah perangkat dibuat, aplikasi Blynk akan menyediakan Auth Token yang nantinya dimasukkan ke dalam program Arduino IDE agar ESP32 dapat terkoneksi dengan dashboard Blynk dan mengirimkan data sensor secara *real-time*.



Gambar 4. 45 Penambahan perangkat baru (New Device)

Gambar tersebut menunjukkan tahap pengambilan Template ID dan Template Name dari *platform* Blynk yang harus dimasukkan ke dalam bagian awal program ESP32 di Arduino IDE. Langkah ini penting untuk menghubungkan perangkat ESP32 dengan template proyek yang telah dibuat di Blynk, sehingga perangkat dapat berkomunikasi secara langsung dengan dashboard, mengirim data sensor, dan menerima perintah secara *real-time* melalui jaringan internet.

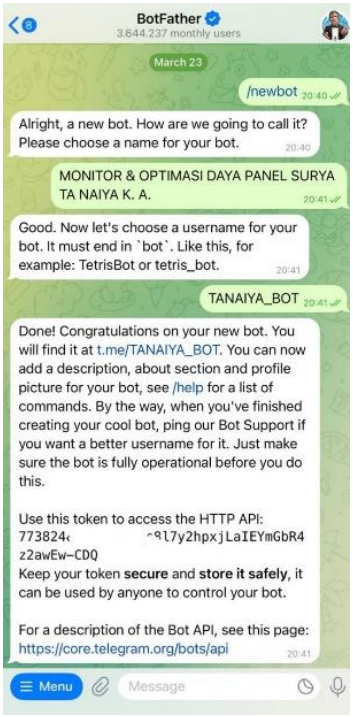


Gambar 4. 46 Konfigurasi Template ID dan Template Name

4.4.3 Pembuatan Akun Telegram

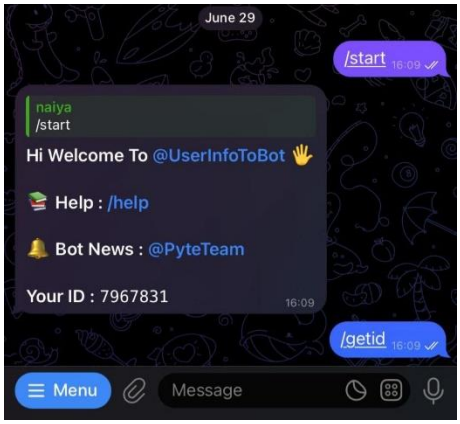
Sebagai bagian dari sistem monitoring berbasis IoT, digunakan fitur notifikasi otomatis melalui aplikasi Telegram. Fitur ini memungkinkan sistem untuk mengirimkan informasi penting seperti suhu panel yang melebihi ambang batas atau kondisi sistem lainnya secara *real-time* kepada pengguna. Untuk mewujudkan hal ini, dilakukan pembuatan dan konfigurasi bot Telegram yang dapat diakses oleh mikrokontroler ESP32 melalui API Telegram.

Untuk dapat menggunakan bot Telegram sebagai sistem notifikasi, dilakukan serangkaian tahapan mulai dari proses pembuatan bot, pengambilan Chat ID pengguna, hingga pengintegrasian kode program ke dalam sistem berbasis ESP32. Tahapan pertama dimulai dengan membuat bot melalui layanan resmi Telegram bernama @BotFather. Pengguna dapat mengetik perintah /newbot untuk memulai, kemudian diminta untuk memasukkan nama bot serta username yang bersifat unik dan diakhiri dengan kata “bot”. Setelah langkah tersebut berhasil dilakukan, BotFather akan memberikan Token API, yaitu sebuah kode akses rahasia yang berfungsi sebagai penghubung antara program dan bot yang telah dibuat.



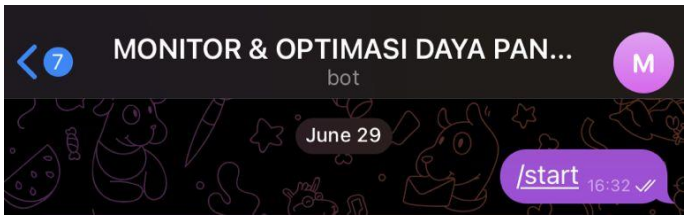
Gambar 4. 47 Pembuatan Bot Telegram menggunakan layanan @BotFather.

Agar sistem dapat mengirimkan notifikasi ke akun Telegram pengguna, maka diperlukan Chat ID. Chat ID ini dapat diperoleh dengan menggunakan bot pihak ketiga bernama @UserInfoToBot. Pengguna cukup mengirimkan perintah /start dan /getid, kemudian bot akan membalas dengan ID pengguna yang dapat dimasukkan ke dalam program.



Gambar 4. 48 Pengambilan Chat ID pengguna melalui bot @UserInfoToBot.

Sebelum bot dapat mengirimkan notifikasi kepada pengguna, pengguna wajib terlebih dahulu memulai interaksi dengan bot melalui tombol **Start** di Telegram. Hal ini penting untuk membuka akses antara bot dan pengguna, sehingga pengiriman pesan bisa dilakukan tanpa hambatan.



Gambar 4. 49 Tampilan awal profil bot Telegram sebelum digunakan.

Setelah mendapatkan Token API dan Chat ID, dilakukan penulisan kode program di Arduino IDE. Fungsi utama yang digunakan untuk mengirim pesan adalah sendTelegramMessage(), yang memanfaatkan HTTP POST ke API Telegram. Chat ID dimasukkan ke dalam array, sehingga memungkinkan sistem mengirim ke satu atau beberapa pengguna sekaligus.

```
const char* telegramToken = "773824     Hbqs8l7y2hpxjLaIEYmGbR4z2awEw-CDQ";
const char* chatIDs[] = {
  "796783     ""    //
};
const int jumlahChat = sizeof(chatIDs) / sizeof(chatIDs[0]);

void sendTelegramMessage(String message) {
  for (int i = 0; i < jumlahChat; i++) {
    HTTPClient http;
    String url = "https://api.telegram.org/bot" + String(telegramToken) + "/sendMessage";
    String payload = "chat_id=" + String(chatIDs[i]) + "&text=" + message + "&parse_mode=Markdown";

    http.begin(url);
    http.addHeader("Content-Type", "application/x-www-form-urlencoded");

    int httpResponseCode = http.POST(payload);
    if (httpResponseCode > 0) {
      Serial.println("Pesan terkirim ke Telegram ID " + String(chatIDs[i]));
    } else {
      Serial.print("Error kirim ke ID "); Serial.print(chatIDs[i]);
      Serial.print(": "); Serial.println(httpResponseCode);
    }

    http.end();
    delay(500);  // Delay kecil untuk mencegah overload server Telegram
  }
}
```

Gambar 4. 50 Potongan Kode Program Pengiriman Pesan Telegram Menggunakan ESP32.

Ketika suhu panel melebihi ambang batas, sistem akan secara otomatis mengirimkan pesan notifikasi kepada pengguna melalui aplikasi Telegram. Notifikasi yang dikirimkan bersifat informatif dan menggunakan format Markdown agar lebih rapi dan mudah dibaca.

Gambar 4. 51 Contoh tampilan notifikasi pesan dari Telegram Bot kepada pengguna.

Untuk mendukung sistem notifikasi jarak jauh pada proyek monitoring panel surya, dibuat sebuah bot Telegram menggunakan layanan BotFather. Bot ini berfungsi untuk mengirimkan pemberitahuan otomatis kepada pengguna saat terjadi kondisi tertentu, seperti suhu melebihi ambang batas atau tinggi air tidak normal.

Langkah awal dilakukan dengan mengetik perintah /newbot pada aplikasi Telegram untuk memulai pembuatan bot. Selanjutnya, bot diberi nama "MONITOR & OPTIMASI DAYA PANEL SURYA TA NAIYA K. A." dan username TANAIYA_BOT yang unik dan diakhiri dengan kata bot sesuai aturan Telegram. Setelah bot berhasil dibuat, sistem memberikan link akses ke bot serta token API yang nantinya akan digunakan untuk menghubungkan ESP32 dengan bot melalui pemrograman.

Token API ini bersifat rahasia dan digunakan dalam pemrograman ESP32 untuk mengirim pesan dari perangkat ke bot Telegram secara otomatis.