BAB IV
PEMBUATAN ALAT

4.1 Pembuatan Prototipe

Pembuatan prototipe merupakan salah satu tahapan penting dalam proses pengembangan alat, khususnya dalam penyusunan tugas akhir ini. Prototipe berfungsi sebagai model awal dari sistem yang dirancang untuk menunjukkan cara kerja, menguji fungsi, serta mengevaluasi efektivitas rancangan secara praktis sebelum diaplikasikan lebih lanjut. Dalam konteks tugas akhir ini, prototipe ini fokus pada perancangan dan pembangunan perangkat keras (*Hardware*) yang mampu mendeteksi tingkat kelembaban tanah secara *Real-time* menggunakan sensor kelembaban tanah. Serta merealisasikan sistem *monitoring* dan penyiraman tanaman secara otomatis berbasis *Internet of Things (IoT)*. Data yang diperoleh dari sensor tersebut digunakan sebagai dasar pengambilan keputusan untuk mengontrol sistem penyiraman secara otomatis. Sistem ini dirancang agar pompa air hanya akan aktif ketika tingkat kelembaban tanah berada di bawah ambang batas yang telah ditentukan sebelumnya. Dengan demikian, penggunaan air dan energi dapat dihemat, serta kestabilan kondisi tanah tetap terjaga. Secara umum, proses pembuatan alat dalam tugas akhir ini terbagi menjadi 2 tahapan utama, yaitu:

a. Pembuatan Perangkat Keras

Pembuatan perangkat keras adalah langkah merancang dan menyusun elemen fisik dari suatu sistem project yang akan kita rancang. Perangkat keras mencakup berbagai aspek seperti mikrokontroler, sensor, aktuator, kabel, konektor, dan komponen pendukung lainnya yang berkolaborasi untuk melaksanakan fungsi tertentu. Proses ini dimulai dengan menentukan kebutuhan sistem, diikuti dengan pemilihan komponen yang sesuai, perakitan, dan kemudian pengujian untuk memastikan sistem berfungsi dengan baik. Pembuatan perangkat keras merupakan fase krusial dalam pengembangan teknologi karena menjadi dasar utama yang memungkinkan perangkat lunak berjalan dalam aplikasi nyata.

Selain faktor teknis, pembuatan perangkat keras juga menganggap penting daya tahan, efisiensi energi, dan kemudahan untuk terintegrasi dengan sistem lain. Dalam banyak situasi, perangkat keras dirancang agar dapat beroperasi secara otomatis, responsif terhadap lingkungan melalui sensor, dan mampu mengatur output seperti lampu, atau pompa. Keberadaan mikrokontroler sangat vital dalam mengelola semua proses tersebut, berfungsi sebagai pusat yang menangani input dan output dari sistem. Oleh karena itu, pembuatan perangkat keras tidak hanya memerlukan keterampilan dalam perakitan fisik, tetapi juga pengetahuan tentang sistem tertanam dan komunikasi data antar perangkat.

b. Pembuatan Perangkat Lunak

Pembuatan perangkat lunak adalah suatu proses yang mencakup desain, pengkodean, pengujian, dan pemeliharaan seperangkat instruksi atau kode program yang dioperasikan oleh perangkat keras untuk menyelesaikan tugas tertentu. Jenis perangkat lunak ini dapat berupa aplikasi komputer, sistem tertanam, serta aplikasi yang berjalan di web dan perangkat mobile yang ditujukan untuk menyederhanakan pekerjaan, mengelola informasi, atau menyediakan antarmuka bagi pengguna. Proses pembuatan perangkat lunak terdiri dari analisis kebutuhan, desain sistem, penulisan kode dengan bahasa pemrograman, serta pengujian untuk memastikan perangkat lunak berfungsi seperti yang diharapkan. Selain itu, proses ini juga mencakup dokumentasi dan pembaruan rutin untuk disesuaikan dengan kebutuhan pengguna dan perkembangan teknologi yang ada.

Salah satu elemen krusial dalam pengembangan perangkat lunak adalah sistem pemantauan, yaitu fitur yang memungkinkan pengguna untuk melihat, menganalisis, dan mengendalikan proses atau data dari perangkat keras secara langsung. Pemantauan ini bisa berupa dasbor, grafik, pemberitahuan, atau tampilan visual lainnya yang menampilkan informasi dari sensor atau sistem otomatis. Misalnya, dalam sistem pertanian pintar, perangkat lunak pemantauan memberikan kemampuan kepada pengguna untuk mengawasi tingkat kelembaban tanah, suhu, dan kondisi pompa air melalui aplikasi di web atau perangkat mobile. Dengan hadirnya pemantauan dalam perangkat lunak, pengguna dapat membuat keputusan dengan lebih cepat dan tepat, serta meningkatkan efisiensi dan keamanan

keseluruhan dari sistem. Selain *monitoring*, alat ini juga terdapat *controlling* untuk menyalakan 3 buah pompa dan 1 buah lampu 12 vdc yang terdapat pada *Greenhouse*.

4.2 Alat dan Bahan

Dalam pembuatan proyek tugas akhir ini, ada berbagai kebutuhan utama dan tambahan yang harus dipenuhi. Menentukan kebutuhan ini adalah langkah penting dalam proses pengembangan, yang bertujuan untuk membuat daftar lengkap tentang komponen dan sumber daya yang diperlukan. Daftar ini akan berfungsi sebagai dasar dalam merancang dan membangun perangkat, sehingga memastikan bahwa sistem yang dikembangkan sesuai dengan spesifikasi dan tujuan penelitian. Kebutuhan utama umumnya meliputi komponen utama seperti mikrokontroler, sensor, aktuator, dan sistem daya yang mendukung kinerja perangkat secara keseluruhan. Di sisi lain, kebutuhan tambahan mencakup elemen-elemen seperti kabel penghubung, modul komunikasi, serta alat untuk instalasi dan pengujian. Dengan pemetaan kebutuhan yang akurat, tim pengembang dapat menghindari kekurangan material, menghemat waktu dan biaya, serta mengurangi risiko kegagalan selama tahap implementasi. Proses ini juga memungkinkan pengembangan sistem dilakukan secara terstruktur dan terukur, mulai dari fase perencanaan hingga pengujian akhir. Proses awal dalam pembuatan project tugas akhir adalah mempersiapkan alat dan bahan, adapun bahan yang digunakan bisa dilihat pada tabel 4-1.

Tabel 4- 1 Bahan yang digunakan

No	Nama Bahan	Jumlah
1	ESP32 30 Pin	1 buah
2	Sensor Kelembaban Tanah resistif	1 buah
3	Sensor HCSR04T	3 buah
4	Sensor Cahaya BH 1750	1 buah
5	Sensor Hujan	1 buah

No	Nama Bahan	Jumlah
6	Modul RTC DS3231	1 buah
7	Buck Converter LM 2596	1 buah
8	Panel Surya 30Wp	1 buah
9	Solar Charge Controller 10 A	1 buah
10	Pompa 12 VDC	3 buah
11	Lampu 12 VDC (Strip Samsung)	3 strip
12	Relay 5V 6 Channel	1 buah
13	Aki 12V 9Ah	1 buah
14	Lcd 20 x 4	1 buah
15	Keypad 1 x 4	1 buah
16	Kabel Bintik 0,85mm	15m x 4 warna
17	Selector switch	1 buah
18	Toggle switch	1 buah
19	Pilot Lamp	3 buah
20	Besi Hollow 4x4	2 meter
21	Akrilik	1 lembar
22	Timah solder	1 roll
23	PCB Bolong	1 buah
24	Baut 10	6 buah
25	Plastik UV *Greenhouse*	2 meter
26	Roda 1 inch	4 buah
27	Panel Box 40 x 30 x 20	1 buah
28	Kabel tis	1 pack
29	Isolasi 3M	1 roll

Pada Tabel 4-2 dituliskan alat yang digunakan untuk membuat project tugas akhir sistem kelembaban tanah.

Tabel 4- 2 Alat yang digunakan

No	Nama Alat	Jumlah
1	Multimeter Digital	1 buah
2	Gunting	1 buah
3	Solder	1 buah
4	Obeng	1 buah
5	Obeng Type Plus	1 buah
6	Obeng Type Minus	1 buah
7	Gunting	1 buah
8	Isolasi Hitam	1 buah
9	Double Tape Foam Busa	1 buah
10	Tang Potong	1 buah
11	Tang Kupas	1 buah
12	Alat Tulis	1 set
13	Lem Tembak	1 buah
14	Penggaris	1 buah
15	Cutter	1 buah

4.3 Pembuatan perangkat keras

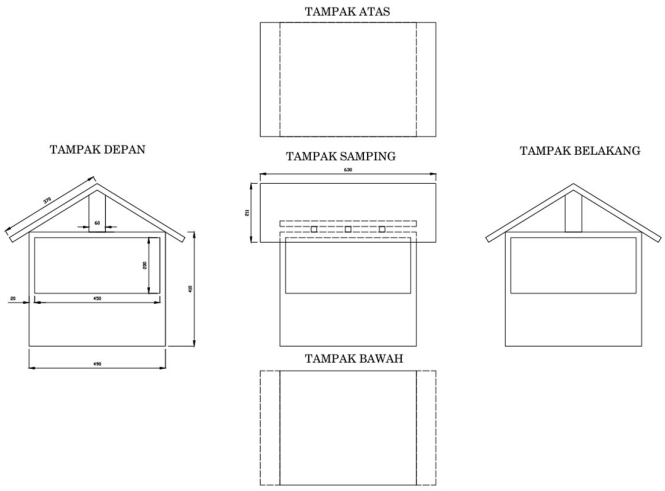
Proses pembuatan perangkat keras dimulai dengan melakukan kajian literatur dan mencari informasi dari berbagai sumber tepercaya, seperti buku teknik, makalah ilmiah, dan studi sebelumnya yang selaras dengan sistem yang ada. Tahap awal ini bertujuan untuk mendapatkan pemahaman menyeluruh tentang cara kerja sistem yang akan dibuat, serta sebagai fondasi dalam menentukan komponen yang diperlukan. Selanjutnya, dilakukan analisis teknis untuk merumuskan spesifikasi sistem, merancang strategi desain, dan menyesuaikan rancangan dengan kebutuhan operasional perangkat. Dengan pendekatan ini, pengembangan perangkat keras dapat dilaksanakan dengan lebih terarah, efektif, dan sesuai dengan tujuan sistem yang direncanakan.

4.3.1 Perancangan Mekanik

Dalam bagian perancangan mekanik proses ini dimulai dengan menentukan kebutuhan mekanik yang didasarkan pada fungsi alat serta lingkungan di mana alat tersebut digunakan, termasuk ukuran, bentuk, material, dan sistem penempatan bagian-bagian. Tujuannya adalah untuk memastikan bahwa alat dapat beroperasi dengan stabil, aman, dan efisien, serta dapat melindungi komponen elektronik dari kerusakan fisik dan gangguan eksternal seperti panas, air, atau debu. Dalam fase ini, referensi desain dari alat serupa serta pertimbangan ergonomi dan kemudahan perawatan digunakan sebagai acuan.

Desain protipe greenhouse berorientasi pada penciptaan struktur fisik yang dapat mendukung pertumbuhan tanaman secara maksimal dengan memperhatikan aspek fungsionalitas, ketahanan, dan penggunaan ruang yang efisien. Proses ini dimulai dengan menetapkan ukuran dan bentuk greenhouse yang sesuai dengan luas lahan serta kebutuhan budidaya tanaman.

Setelah desain selesai dan telah melewati penilaian awal, langkah berikutnya adalah pelaksanaan mekanik dengan membuat prototipe menggunakan bahan seperti pipa PVC, baja ringan, aluminium, plastik tahan UV, atau kaca. Pekerjaan pemotongan dan perakitan dilakukan dengan cermat agar sesuai dengan desain yang telah dibangun. Untuk tampilan design *Greenhouse* bisa dilihat pada Gambar 4.1.



Gambar 4.1 Design *Greenhouse*

4.3.2 Perancangan *Greenhouse*

Pembuatan *Greenhouse* berdasarkan desain teknik yang telah dijelaskan dimulai dengan menyiapkan komponen utama yang berupa besi hollow berukuran 4x4 cm, yang akan berfungsi sebagai kerangka utama. Proses awal mengharuskan pemotongan besi hollow sesuai dengan ukuran yang terdapat dalam gambar. Untuk bagian bawah, besi hollow dipotong sepanjang 490 mm (lebar) dan 620 mm (panjang), kemudian dirakit menjadi sebuah kerangka persegi panjang sebagai dasar pondasi. Penyambungan dilakukan melalui proses pengelasan atau dengan mempergunakan baut dan braket logam agar kerangka dasar kuat dan tepat.

Gambar 4.2 Pembuatan *Greenhouse*

Gambar 4.2 adalah kerangka dasar dalam proses pembuatan *Greenhouse*, langkah selanjutnya adalah mengatur kerangka vertikal atau tiang penyangga. Tiang dipasang di setiap sudut serta titik tengah dengan tinggi total struktur mencapai 620 mm, dan bagian atas dirancang menyerupai atap segitiga. Proses pembuatan kerangka atap segitiga mengacu pada sudut kemiringan 370 mm dengan puncak setinggi 120 mm dari titik dasar. Penyambungan antar besi dikerjakan dengan ketelitian untuk memastikan posisi atap simetris serta menjaga stabilitas struktur

secara keseluruhan. Setelah semua kerangka vertikal dan atap terpasang, pemeriksaan sudut dan kekuatan sambungan dilakukan sebelum melanjutkan ke tahap berikutnya.

Tahapan berikutnya yaitu pemasangan penutup dinding dan atap yang bisa dilihat pada Gambar 4.3. Lembaran penutup dipotong sesuai dengan ukuran panel dinding dan atap, dan kemudian dipasang dengan bantuan lem tembak atau lem khusus agar terikat dengan baik pada rangka besi hollow. Di bagian depan, terdapat bukaan berukuran 430 mm x 250 mm sesuai dengan desain tampak depan, yang dapat berfungsi sebagai jendela ventilasi atau akses untuk memantau tanaman.

Gambar 4. 3 Pemasangan alas *Greenhouse*

Proses terakhir adalah merapikan setiap sambungan, menerapkan cat anti karat pada seluruh area besi hollow untuk memperpanjang masa pakai struktur, serta menguji kestabilan rumah kaca dengan menggesernya atau menekan sisi-sisi struktur dengan lembut. Rumah kaca dapat diletakkan pada fondasi yang datar, seperti paving atau lantai semen, agar posisinya tidak berpindah. Setelah semua selesai, struktur siap digunakan sebagai tempat budidaya tanaman mini dengan perlindungan dari hujan dan sinar matahari langsung.

4.3.3 Pembuatan Penyangga Panel Surya

Pembuatan penyangga untuk panel surya seperti yang ditunjukkan pada gambar 4.4 dimulai dengan memotong besi hollow sebagai bahan utama untuk kerangka. Langkah pertama adalah membentuk kerangka dasar berbentuk kotak yang akan dihubungkan dengan sebagai pondasi penyangga. Keempat sisi kerangka dasar disambungkan menggunakan teknik pengelasan untuk menciptakan kerangka yang kuat dan stabil.

Setelah itu, tiang vertikal utama dipasang tegak dari tengah rangka bawah dan dilas dengan kuat. Tiang ini berfungsi sebagai penopang dan poros perputaran penyangga. Pada bagian tengah hingga atas tiang, dipasang penyangga horizontal sebanyak dua hingga tiga bar, yang berfungsi menahan kerangka yang mendukung panel surya. Setiap penyangga horizontal dipasang secara melintang dan dilengkapi dengan lubang untuk baut atau engsel agar memungkinkan rotasi atau pengaturan sudut panel. Dalam rancangan ini, sistem rotasi dirancang agar bisa mencapai kemiringan hingga 180 derajat, sehingga panel dapat disesuaikan dengan posisi matahari demi efisiensi maksimal.

Selanjutnya, kerangka penopang panel surya dipasang di bagian atas. Penopang ini terdiri dari dua sisi rangka horizontal yang membentuk persegi panjang sesuai ukuran panel surya. Kerangka ini dihubungkan ke penyangga horizontal menggunakan heavy-duty hinges atau pivot bolts, yang memungkinkan panel dimiringkan ke depan dan ke belakang.

Tahap terakhir adalah mengecat seluruh permukaan besi dengan cat anti-karat agar tahan terhadap cuaca luar. Setelah cat mengering, braket siap digunakan dan dipasang di lokasi dengan paparan sinar matahari terbaik. Desain ini sangat cocok untuk panel surya portabel maupun sistem panel kecil menengah yang digunakan di lapangan, karena menawarkan mobilitas yang tinggi dan pemasangan yang tidak bersifat permanen. Kelebihannya adalah kemudahan dalam mengatur sudut panel secara manual tanpa memerlukan alat bantu tambahan.

Gambar 4. 4 Penyangga Panel Box & Panel Surya

4.3.4 Pembuatan Panel Box

Gambar 4. 5 Membolongi Sisi Samping Panel Box Untuk Konektor GX - 16

Panel box yang terlihat pada Gambar 4.5 berfungsi sebagai pelindung dan pengelola komponen elektronik seperti mikrokontroler, modul sensor, sumber daya,

dan indikator sistem. Fungsinya tidak hanya sebagai tempat penyimpanan, tetapi juga memberikan perlindungan dari debu, air, dan benturan fisik yang bisa mengganggu kinerja sistem. Dalam sistem otomatisasi atau *monitoring* yang memakai ESP32 atau mikrokontroler lain, panel box ini memegang peranan penting dalam memastikan ketahanan dan keselamatan rangkaian. Desain panel ini akan dilengkapi dengan LCD 20x4 untuk tampilan data, keypad 1x4 untuk input dari pengguna, serta 3 lampu indikator untuk menunjukkan status sistem secara visual.

Untuk membuat lubang pada panel, dilakukan dengan hati-hati menggunakan bor listrik dan lubang gergaji untuk lubang bulat (seperti pada pilot lamp) serta gerinda potong atau nibbler manual untuk lubang persegi panjang seperti pada LCD dan keypad. Pada Gambar 4.6 terlihat bahwa lubang-lubang telah dibuat dengan pola yang rapi untuk tombol atau indikator tambahan di depan panel box. Setelah lubang selesai, tepi-tepi lubang akan diperhalus menggunakan amplas besi atau kikir agar tidak melukai saat pemasangan dan juga untuk melindungi kabel dari kerusakan akibat gesekan dengan tepi yang tajam.

Gambar 4. 6 Pembolongan Panel Box untuk Komponen Alat Tugas Akhir

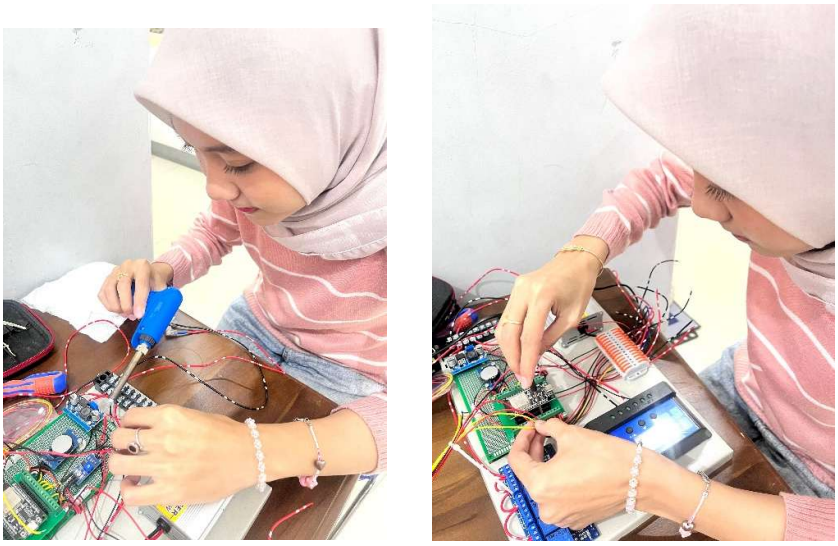
4.3.5 Perancangan pada PCB

Perancangan rangkaian sistem pada PCB bolong (veroboard) dimulai dengan merencanakan penempatan komponen secara manual. Tahapan ini sangat

penting untuk memastikan semua komponen utama seperti ESP32, header sensor, LCD I2C, dan modul relay terorganisir dengan baik dan memiliki jalur koneksi yang optimal. Penempatan dimulai dengan memilih lokasi untuk ESP32 sebagai pengendali utama sistem. ESP32 biasanya ditempatkan di tengah atau di salah satu sisi veroboard untuk memudahkan koneksi dengan komponen lainnya. Selanjutnya, lokasi untuk header sensor, seperti sensor kelembaban tanah, hujan, cahaya, dan level air ditentukan.

Setelah tata letak ditentukan, langkah selanjutnya adalah menyolder header pin ESP32 ke PCB. Penyolderan dilakukan dengan hati-hati agar tidak terjadi hubungan pendek antara pin-pin tersebut. Kemudian, LCD I2C dipasang pada bagian atas atau samping PCB, tergantung pada posisi tampilan yang diinginkan di dalam panel box. Karena LCD I2C hanya memanfaatkan jalur SDA dan SCL, maka jalur data dihubungkan langsung ke pin I2C pada ESP32 dengan menggunakan kabel jumper dan penyolderan kecil.

Pada tahap selanjutnya, jalur koneksi antar komponen disiapkan dan disolder menggunakan kabel jumper atau kawat email. Jalur data, tegangan (VCC), dan ground (GND) dari ESP32 dihubungkan secara sistematis ke setiap modul sensor dan LCD. Untuk menghindari interferensi, jalur power 5V dan GND dibuat dengan jarak yang cukup dan menggunakan kabel yang sedikit lebih besar. Di samping itu, beberapa konektor output seperti terminal screw atau header female disiapkan untuk membantu koneksi ke panel luar, sehingga memudahkan dalam perawatan atau penggantian sensor. Semua sambungan diperiksa menggunakan multimeter untuk memastikan tidak ada jalur yang terputus atau short. Perancangan komponen pada PCB bisa dilihar pada gambar 4.7.



Gambar 4. 7 Perancangan Komponen pada Bagian Dalam Panel Box

4.3.6 Pengaturan Catu Daya Sistem

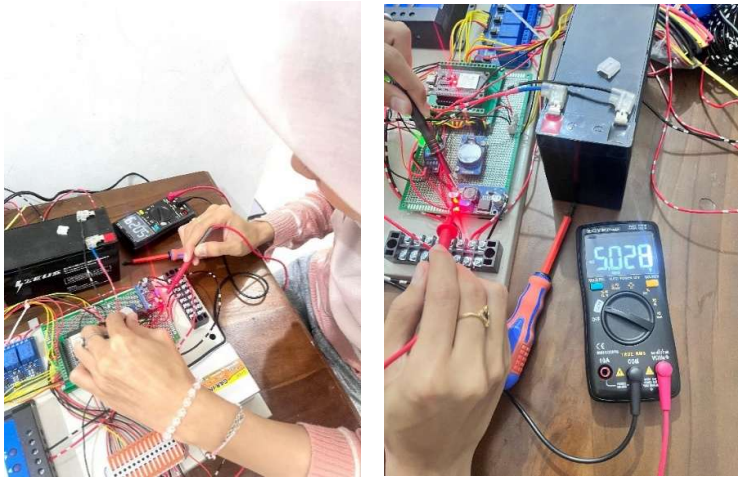
Pengaturan catu daya adalah salah satu aspek penting dalam sistem *monitoring* kelembaban tanah berbasis ESP32 ini, karena menjamin kestabilan suplai tegangan dan arus ke seluruh rangkaian elektronik. Sistem ini memanfaatkan sumber utama energi berupa panel surya yang mengisi baterai aki 12V 9Ah sebagai tempat penyimpanan energi. Untuk menyesuaikan kebutuhan tegangan pada berbagai komponen, khususnya ESP32 dan sensor-sensor yang beroperasi pada tegangan 5V atau 3.3V, digunakan modul step-down buck converter LM2596 sebagai pengatur tegangan.

Uji coba pengaturan daya dilakukan dengan mengukur nilai masukan dan keluaran dari aki 9Ah yang telah terhubung dengan modul LM2596. Tegangan masukan diukur langsung dari terminal aki, sedangkan tegangan keluaran diambil dari sisi keluaran LM2596 yang telah diatur menjadi 5V untuk memenuhi kebutuhan ESP32 dan sensor. Pengukuran ini bertujuan untuk memastikan bahwa tegangan keluaran tetap konsisten pada nilai yang diperlukan, meskipun terjadi fluktuasi pada tegangan aki akibat proses pengisian dari panel surya maupun beban dari sistem *monitoring*. Adapun kegiatan pengaturan daya dilakukan seperti Gambar 4.8.

Gambar 4.8 Pengukuran Catu Daya

Dalam pengukuran ini, digunakan multimeter digital untuk mencatat nilai tegangan masukan dan keluaran secara *Real-time*. Di samping itu, pengamatan juga dilakukan pada beban sistem saat aktif untuk mengecek apakah terjadi penurunan tegangan yang signifikan. Hasil pengamatan menunjukkan bahwa tegangan masukan dari aki berada pada rentang 11.8V hingga 12.8V tergantung pada tingkat pengisian, sementara keluaran dari LM2596 dapat dipertahankan stabil dalam kisaran 5. 02V hingga 5.2V. Ini menunjukkan bahwa LM2596 dapat menjalankan fungsi sebagai penurun tegangan dengan baik dan efisien. Adapun hasil dari pengukuran catu daya bisa dilihat pada Gambar 4.9 dan 4.10

Gambar 4.9 Hasil Pengukuran Input LM 2596



Gambar 4. 10 Hasil Pengukuran Output LM 2596

Kestabilan catu daya sangat penting guna menjaga performa dari sensor kelembaban, sensor cahaya, modul *Wi-Fi* ESP32, serta komponen aktuator seperti relay dan pompa. Gangguan pada pasokan daya dapat mengakibatkan sistem tidak berfungsi dengan baik, bahkan dapat merusak beberapa komponen tertentu. Oleh karena itu, pengaturan catu daya yang tepat melalui kombinasi antara aki dan LM2596 membangun dasar yang kuat untuk keandalan sistem pemantauan kelembaban tanah secara keseluruhan. Selain efisiensi, pengaturan ini juga memberikan fleksibilitas dan ketahanan terhadap gangguan daya dari lingkungan.

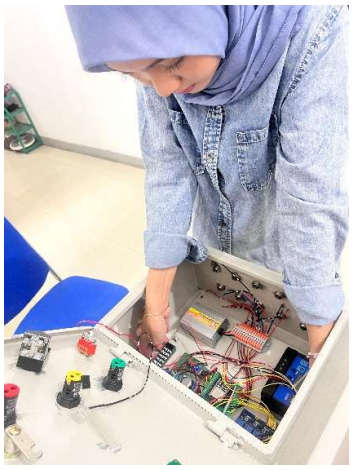
4.3.7 Perakitan Komponen ke Dalam Panel Box

Perakitan komponen ke dalam kotak panel adalah langkah terakhir dalam desain fisik sistem, di mana semua modul, sensor, aktuator, dan sumber daya disusun secara teratur dalam satu tempat yang terlindungi. Panel Box berfungsi sebagai pelindung untuk seluruh rangkaian elektronik dari pengaruh luar seperti debu, air, panas, dan gangguan fisik lainnya.

Semua komponen utama seperti mikrokontroler ESP32, sensor kelembaban tanah, modul relay, konverter LM2596, RTC DS3231, dipasang di atas tatakan dipasang dengan tepat di dalam kotak panel. Penataan ini dilakukan dengan

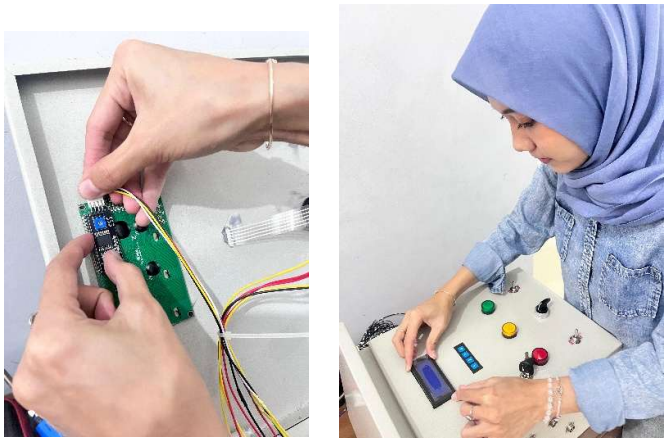
mempertimbangkan jalur koneksi antara komponen agar tidak terjadi kekacauan kabel dan memudahkan saat perawatan atau pemecahan masalah.

Setiap koneksi antara elemen diatur rapi menggunakan kabel jumper, konektor, atau terminal blok. Selain itu, terminal input dari panel surya serta keluarannya ke pompa diletakkan di bagian luar kotak panel melalui lubang konektor kedap air, sehingga memudahkan instalasi di lapangan tanpa perlu membuka seluruh tutup kotak. Pada Gambar 4.11 terlihat peletakan komponen ke dalam Panel Box.



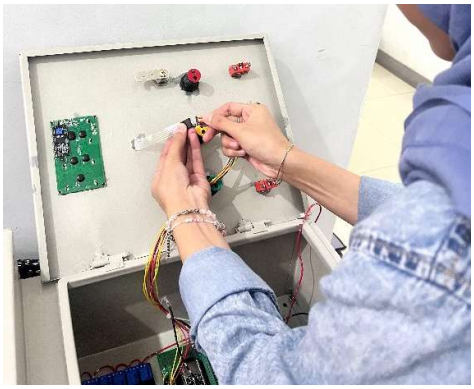
Gambar 4.11 Peletakan Komponen ke dalam Panel Box

Perakitan elemen dalam kotak panel ini menghasilkan sebuah unit yang kuat, aman, dan efisien. Dalam proyek ini, seluruh komponen utama, termasuk keypad dan LCD, dirakit dan dipasang secara terstruktur di dalam dan pada bagian luar panel box. Panel box digunakan sebagai tempat perlindungan seluruh rangkaian dari gangguan luar seperti debu, kelembaban, dan benturan fisik, sekaligus sebagai wadah yang memudahkan pengoperasian alat oleh pengguna. Gambar 4.12 adalah pemasangan Kabel ke LCD.



Gambar 4.12 Pemasangan Kabel ke Komponen sisi depan Panel Box

Keypad 1x4 dipasang di bagian depan panel box dengan membuat lubang sesuai ukuran tombol menggunakan alat pemotong presisi atau cutter. Penempatan keypad pada permukaan luar panel dirancang agar mudah diakses oleh pengguna tanpa harus membuka tutup panel box. Posisi ini memungkinkan pengguna untuk memberikan input secara langsung, seperti pengaturan jadwal atau pilihan menu. Gambar 4.13 adalah foto dari pemasangan Keypad.



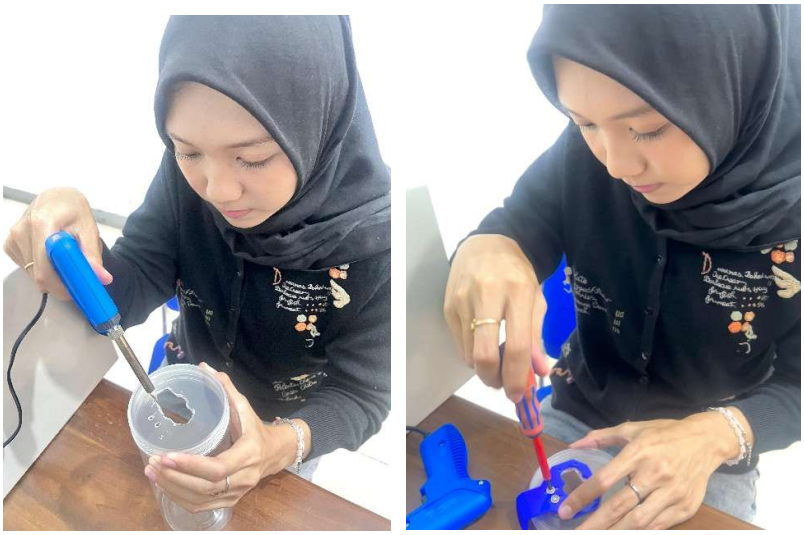
Gambar 4.13 Pemasangan Keypad pada Sisi Depan Panel Box

4.3.8 Perakitan Tangki Air ke *Greenhouse*

Tangki air berperan sebagai sumber utama untuk menyirami tanaman, sehingga perlu dipasang dengan cara yang stabil, aman, dan terhubung dengan

sistem sensor. Pada tahap ini, tangki air diletakkan dekat dengan bangunan greenhouse untuk memudahkan distribusi air ke area tanaman. Selain itu, lokasi tangki akan disesuaikan agar mudah diakses untuk pengisian air dan perawatan rutin.

Proses awal yang dilakukan adalah menyolder atau memasang tutup tangki dengan metode pemanasan dan lem plastik khusus seperti pada Gambar 4.14. Tutup tangki ini juga berfungsi sebagai tempat utama untuk memasang sensor ultrasonik HC-SR04T yang bertugas mengukur tingkat ketinggian air di dalam tangki. Penting sekali untuk menjaga keamanan dan kedapnya tutup agar debu, serangga, atau cipratan air dari luar tidak mengganggu fungsi sensor dan kebersihan air di dalam tangki. Setelah tutup tangki terpasang dengan baik, langkah selanjutnya adalah memasang bracket khusus untuk mendukung sensor HC-SR04T.

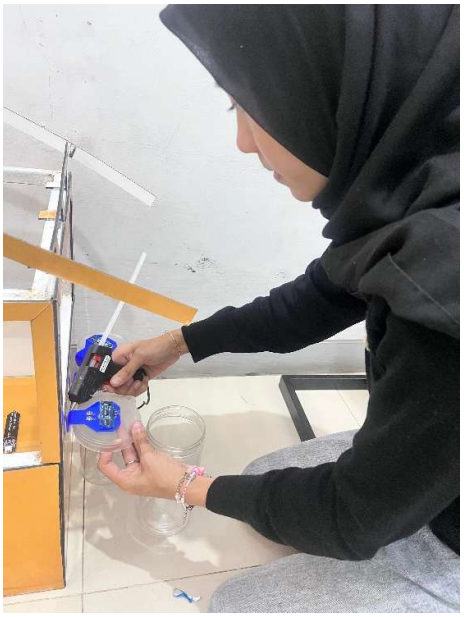


Gambar 4. 14 Melubangi dan Pemasangan Bracket Sensor HC-SR04T

Bracket ini terbuat dari bahan akrilik atau PVC yang tahan air dan dipasang dengan kuat di tengah tutup tangki menggunakan baut atau lem industri. Tujuan dari bracket ini adalah agar sensor bisa diposisikan secara tegak lurus terhadap permukaan air, sehingga pengukuran jarak dapat dilakukan dengan tepat. Sensor HC-SR04T kemudian dipasang dan dikencangkan pada bracket dengan posisi

sensor menghadap langsung ke bagian dasar tangki. Untuk pemasangan bracket pada sensor HC-SR04T dapat dilihat pada Gambar 4.15 dan 4.16.

Gambar 4.15 Pemasangan Sensor ke Bracket



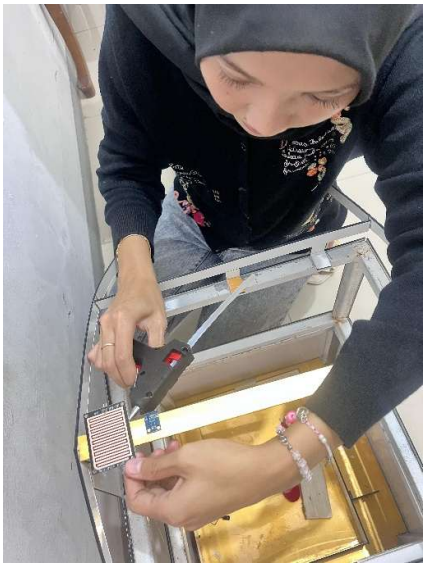
Gambar 4. 16 Pemasangan Tangki

4.3.9 Pemasangan Sensor Hujan dan Sensor BH 1750

Pemasangan sensor hujan dan sensor BH1750 merupakan tahap penting dalam meningkatkan sistem *monitoring* di dalam *Greenhouse*. Kedua alat ini saling melengkapi dengan memberikan informasi cuaca luar, seperti intensitas cahaya dan

status hujan, yang berguna untuk menyesuaikan pengoperasian sistem irigasi dan pencahayaan otomatis. Agar hasil pengukuran yang diperoleh tepat dan representatif, sensor hujan dan BH1750 dipasang di atap greenhouse supaya terpapar langsung dengan kondisi luar dan tidak terhalang oleh sesuatu di sekitarnya.

Sensor hujan dipasang secara horizontal di atas atap greenhouse menggunakan lem yang kuat dan tahan air seperti lem tembak dan terpasang pada kerangka atap. Posisi sensor dijaga tetap lurus agar air hujan dapat mengalir dengan baik ke permukaan deteksi. Sensor ini juga diletakkan di bagian atap yang tidak dilindungi oleh plastik, sehingga tetap terpapar air hujan tetapi terhindar dari gangguan fisik seperti daun atau kotoran lain yang bisa mengganggu akurasi deteksi.



Gambar 4. 17 Pemasangan Sensor Hujan

Sementara itu, sensor BH1750 yang berguna untuk mengukur tingkat cahaya dipasang dekat dengan sensor hujan tetapi dalam sudut miring ke atas supaya dapat menangkap cahaya matahari secara maksimal. Sensor ini dipasang menggunakan bracket kecil atau diletakkan pada dudukan akrilik agar sudutnya tetap stabil saat terpapar angin atau getaran. Kabel dari kedua sensor diarahkan ke dalam panel box lewat pipa pelindung atau selongsong kabel untuk melindungi dari

kerusakan akibat suhu tinggi dan air. Dengan pemasangan yang tepat di atap greenhouse, sensor hujan dan BH1750 dapat menyediakan data yang akurat dan berkelanjutan untuk mendukung otomatisasi sistem dengan lebih cerdas. Pemasangan Sensor Hujan pada Gambar 4.17 dan pemasangan sensor BH 1750 terdapat pada Gambar 4.18

Gambar 4. 18 Pemasangan Sensor BH 1750

4.3.10 Pemasangan Roda untuk Mobilisasi *Greenhouse*

Pada tahap ini, roda dipasang di bagian bawah kerangka greenhouse mini untuk mempermudah mobilitas. Roda diperlukan agar struktur greenhouse bisa dipindahkan dengan gampang sesuai keperluan, baik untuk menyesuaikan dengan arah cahaya matahari, merawat area sekitar, maupun untuk keperluan pengujian alat di lokasi lainnya.. Pemasangan roda dilakukan di keempat sudut kerangka utama dengan menggunakan baut dan mur yang dikencangkan demi memastikan kekuatan dan keamanan saat alat dipindahkan. Penambahan roda ini menawarkan fleksibilitas tinggi tanpa banyak memerlukan tenaga saat memindahkan alat, sekaligus memastikan efisiensi waktu dan tenaga selama proses perawatan dan pengamatan. Dengan adanya sistem roda ini, prototipe greenhouse tidak hanya

bersifat tetap, tetapi juga dapat beradaptasi dengan kondisi sekitar dan operasional di lapangan. Gambar 4.19 menunjukkan proses pemasangan roda pada *Greenhouse*.

Gambar 4. 19 Pemasangan Roda *Greenhouse*

4.4 Pembuatan Perangkat Lunak

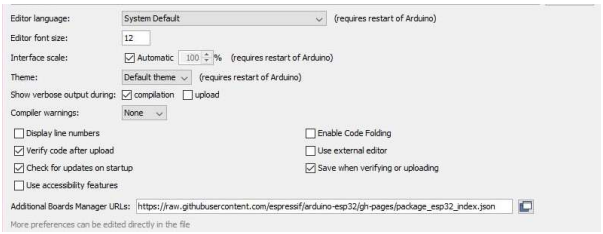
Tahapan dalam proses pembuatan perangkat lunak adalah langkah krusial untuk mewujudkan fungsi sistem berbasis ESP32. Pada fase ini, dilakukan pengembangan kode yang ditulis dengan menggunakan Arduino IDE, sebuah alat pemrograman sumber terbuka yang dapat digunakan dengan berbagai tipe mikrokontroler, termasuk ESP32. Perangkat lunak ini dibuat untuk mengelola logika operasional sistem, mulai dari membaca data sensor kelembaban tanah, memproses data, hingga mengendalikan aktuator seperti pompa dan relay secara otomatis. Di samping itu, perangkat lunak juga meliputi integrasi dengan layanan IoT untuk pemantauan jarak jauh melalui aplikasi. Seluruh proses pemrograman dilakukan dengan memperhatikan efisiensi sistem, akurasi respons, dan kemampuan untuk berinteraksi secara handal dengan semua komponen perangkat keras yang terlibat. Tahapan berikut ini menjelaskan langkah-langkah dalam pembuatan program menggunakan Arduino IDE, yang diawali dengan merencanakan alur logika sistem, menyusun skrip, melakukan pengujian baik

langsung pada perangkat keras maupun melalui simulasi, hingga proses mencari dan memperbaiki kesalahan (debugging) untuk memastikan sistem beroperasi sesuai dengan rencana.

4.4.1 Persiapan Awal Pembuatam Program pada Arduino Ide

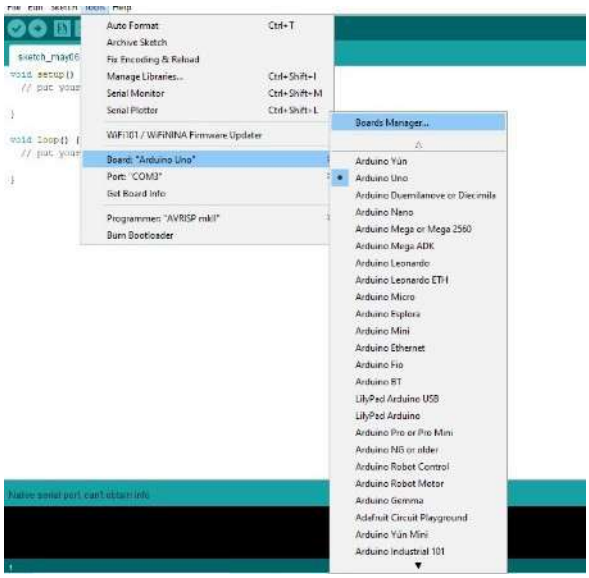
Langkah pertama dalam membangun sistem pemantauan kelembaban tanah berbasis ESP32 dimulai dengan menyiapkan perangkat keras yang diperlukan, seperti laptop atau komputer yang akan digunakan untuk menulis dan meng-upload program, serta kabel USB yang berfungsi menghubungkan perangkat dengan mikrokontroler ESP32.

Selanjutnya, pengguna harus menginstal versi terbaru dari Arduino IDE yang dapat diunduh dari situs resmi https://www.arduino.cc/. 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 4.20.

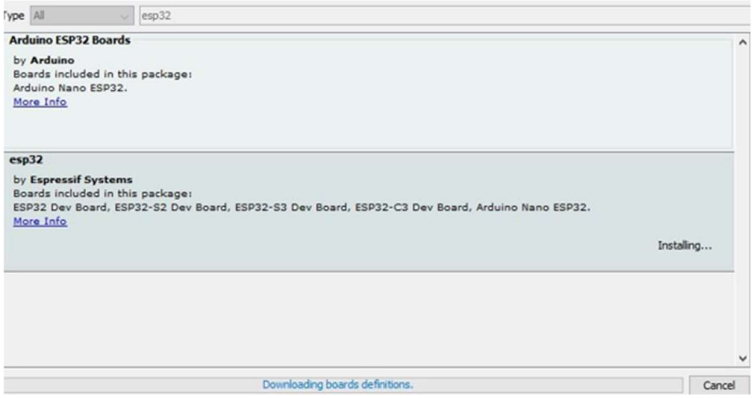


Gambar 4. 20 Menambahkan Additional Boards Manager pada Arduino IDE

Setelah menambahkan *board* ESP32 pada bagian *preference* selanjutnya dapat dilanjutkan pada bagian *boards manager* mengunduh *board* ESP32 pada menu *tools* yang ditampilkan pada gambar 4.8.



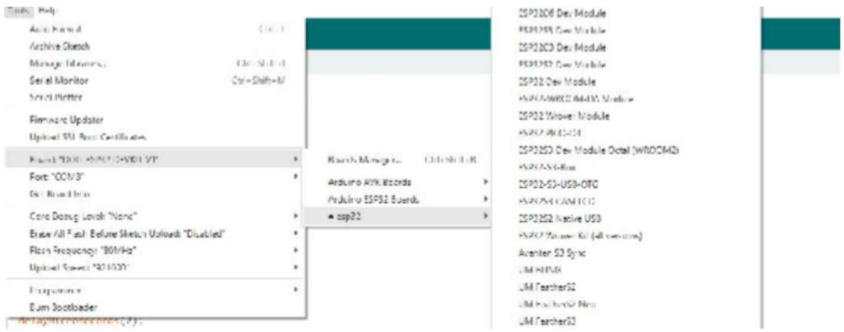
Gambar 4. 21 Melakukan Install ESP32 pada Boards Manager Arduino IDE



Gambar 4. 22 Tampilan Install ESP32 pada Arduino IDE

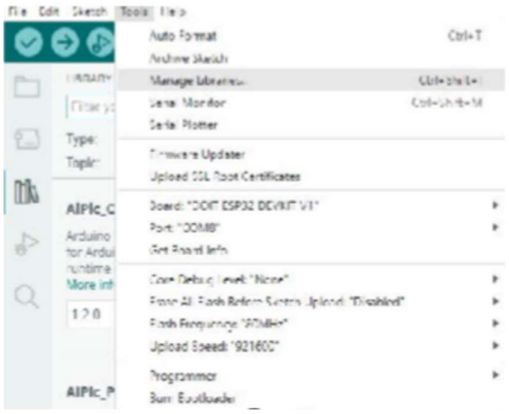
Setelah mengunduh *board* ESP32 pada Arduino IDE yang terlampir pada gambar 4.21 selanjutnya dapat menghubungkan laptop dengan mikrokontroler. Dalam penggunaan ESP32 yang penyusun gunakan, mikrokontroler tersebut dilengkapi dengan tipe *type*. Sehingga, kabel konektor yang penyusun gunakan berupa kabel 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 4.22.

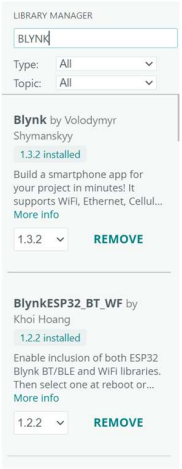


Gambar 4. 23 ESP32 sebagai Board pada Arduino IDE

Setelah mengatur *board* pada Arduino IDE, dapat mengunduh *library* yang akan digunakan terlampir pada gambar 4.23 dan gambar 4.24, dalam alat ini penyusun menggunakan *library* LCD, pengukuran pembacaan sensor, penggunaan *Wi-Fi*, Blynk, Telegram dan Spreadsheet.



Gambar 4. 24 Menentukan Pengunduhan Library Arduino IDE



Gambar 4. 25 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 4.25.

```
#include <ezButton.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <BH1750.h>
#include "RTClib.h"
#include <WiFiManager.h>
#include <HTTPClient.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <BlynkSimpleEsp32.h>
```

Gambar 4. 26 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 relai yang terhubung.

Pada gambar 4.26 menunjukkan penentuan *pin* modul pada Arduino IDE serta pada gambar 4.27 menampilkan penggunaan *pin* yang akan dihubungkan dengan *datastreams Dashboard* Blynk.

```
#define BOTLIBRARY_HTTPS_CLIENT true  // Gunakan HTTPS client yang lebih ringan
#define BOTLIBRARY_WIFI_CLIENT true   // Gunakan WiFi client dasar
#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 PIN_SENSOR_HUJAN 35
#define PIN_SENSOR_TANAH 32
#define SOUND_SPEED 0.034
#define RELAY_AIR 23  //pompa 3
#define RELAY_PUPUK 4    // Pompa 2
#define RELAY_PESTISIDA 15 // Pompa 1
#define RELAY_LAMPU_KUNING 19
#define RELAY_LAMPU_GH 5
#define RELAY_LAMPU_MERAH 18
#define AMBANG_GELAP 50  // Lux dari sensor BH1750
#define TINGGI_WADAH_CM 22  // tinggi wadah dari sensor ke dasar

const int trigPin1 = 13; // tinggi air
const int echoPin1 = 34;
const int trigPin2 = 12;// tinggi pupuk
const int echoPin2 = 39;
const int trigPin4 = 14; // tinggi pestisida
const int echoPin4 = 36;
```

Gambar 4. 27 Pin Modul pada Arduino IDE

Selanjutnya melakukan pemrograman void setup() pada Arduino IDE yang tertampil pada gambar 4.28. Fungsi void setup() di Arduino IDE berfungsi sebagai blok inisialisasi yang hanya dijalankan sekali saat program pertama kali dijalankan (saat papan Arduino atau ESP32 di-*reset* atau dinyalakan). Fungsi ini digunakan untuk menyiapkan kondisi awal dari sistem, seperti konfigurasi pin, memulai komunikasi serial, dan inisialisasi modul-modul sensor atau perangkat lainnya.

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

  lcd.init();
  lcd.backlight();
  lcd.clear();

    // Konfigurasi WiFiManager dengan timeout
  wifiManager.setTimeout(180); // Timeout 3 menit
  wifiManager.setConnectTimeout(60); // Timeout koneksi 1 menit

  lcd.setCursor(0, 0); lcd.print("Mencoba Koneksi WiFi");
  lcd.setCursor(0, 1); lcd.print("SSID: SmartGreenHouse");
  lcd.setCursor(0, 2); lcd.print("Silakan Hubungkan");
  lcd.setCursor(0, 3); lcd.print("Tunggu...");

  wifiManager.autoConnect("SmartGreenHouse - TA GALUH", "12345678");

  lcd.clear();

  if (WiFi.isConnected()) {
  Blynk.config(BLYNK_AUTH_TOKEN);
  Blynk.connect();
  secured_client.setInsecure();  // Pindah ke sini
  Serial.println("WiFi Tersambung!");
  lcd.setCursor(0, 0); lcd.print("WiFi Terhubung!");
  lcd.setCursor(0, 1); lcd.print(WiFi.localIP());
  delay(2000);
  kirimTelegram("✅ *Alat Monitoring & Kontroling Smart Green House TA Galuh Aktif*");
    kirimTelegram(
  "📌 *Panduan Perintah Bot:*\n\n"
  "🔧 *Reset WiFi*\n"
  "`/resetwifi`\n\n"
  "🕒 *Set Waktu RTC:*\n"
  "`/setrtc <Hari> YYYY-MM-DD HH:MM:SS`\n\n"
  "💧 *Set Interval Penyemprotan:*\n"
  "`/setintervalpompa1 <menit>` untuk *Pestisida*\n"
  "`/setintervalpompa2 <menit>` untuk *Pupuk*"
  );
  }

  if (!rtc.begin()) {
    Serial.println("RTC tidak ditemukan!");
    lcd.clear();
    lcd.setCursor(0, 0); lcd.print("RTC ERROR");
    delay(3000);
    // Tetap lanjut meskipun tanpa RTC
  }

  // Uncomment untuk atur waktu RTC pertama kali
  // rtc.adjust(DateTime(2025, 7, 13, 21, 52, 0));

  analogSetAttenuation(ADC_11db);
  analogReadResolution(12);

  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);
  pinMode(trigPin4, OUTPUT);
  pinMode(echoPin4, INPUT);
```

```
188   pinMode(RELAY_AIR, OUTPUT);
189   pinMode(RELAY_PUPUK, OUTPUT);
190   pinMode(RELAY_PESTISIDA, OUTPUT);
191   pinMode(RELAY_LAMPU_KUNING, OUTPUT);
192   pinMode(RELAY_LAMPU_GH, OUTPUT);
193   pinMode(RELAY_LAMPU_MERAH, OUTPUT);
194
195   // Matikan semua relay (HIGH = OFF untuk modul relay high trigger)
196   digitalWrite(RELAY_AIR, HIGH);
197   digitalWrite(RELAY_PUPUK, HIGH);
198   digitalWrite(RELAY_PESTISIDA, HIGH);
199   digitalWrite(RELAY_LAMPU_KUNING, HIGH);
200   digitalWrite(RELAY_LAMPU_GH, HIGH);
201   digitalWrite(RELAY_LAMPU_MERAH, HIGH);
202
203   lightMeter.begin();
204
205   for (byte i = 0; i < KEY_NUM; i++) {
206     keypad_1x4[i].setDebounceTime(100);
207   }
208
209   lcd.clear();
210   lcd.setCursor(0, 0); lcd.print("Galuh Indira Sukma J");
211   lcd.setCursor(3, 1); lcd.print("40040621650021");
212   lcd.setCursor(1, 2); lcd.print("Tugas Akhir Sistem");
213   lcd.setCursor(1, 3); lcd.print("Smart Green House");
214   delay(5000);
215
216   tampilMenuUtama(); // tampilkan menu awal di LCD
217 }
```

Gambar 4. 28 Tampilan *Void Setup* Arduino IDE

Setelah setup() selesai dijalankan, program akan masuk ke fungsi loop() yang berjalan terus-menerus. *Voidloop()* sebagai proses pemrograman yang akan bekerja secara terus menerus yang telah disesuai dengan perintah, pemrograman ini tertampil pada gambar 4.29.

```
void loop() {
  static unsigned long lastSensorUpdate = 0;
  const unsigned long sensorInterval = 1000;

  // Non-blocking sensor updates
  if (millis() - lastSensorUpdate >= sensorInterval) {
    lastSensorUpdate = millis();
    updateSensor();
    tampilkanData();
  }
  Blynk.run();
  kontrolPompaPestisida();
  kontrolPompaPupuk();
  kontrolPompaAir();
  responTombol();
  periksaPesanTelegram();

  static unsigned long lastKirim = 0;
  if (millis() - lastKirim > 2 * 60 * 1000) {
  lastKirim = millis();
  kirimKeSpreadsheet();
  }
}
```

Gambar 4.29 Tampilan Void Loop pada Arduino IDE

4.4.2 Coding Integrasi ESP32 dengan IoT

```
#define BLYNK_TEMPLATE_ID "TMPL6IeZZHqgG"
#define BLYNK_TEMPLATE_NAME "Prototype Green House Tanaman Cabai TA GALUH"
#define BLYNK_AUTH_TOKEN "pqkhsto6E-KKegyN1sMDVIdoPLdvZ0YN"
#define BOT_TOKEN "7847775326:AAEPKcXDl2zh71rlI7woyGL1qxGk7t2V6VU"
#define CHAT_ID "6886361735"

const char* sheetURL = "https://script.google.com/macros/s/AKfycbx93eVQULMIh0phB4ex57RhBItvVpLCMyFO4tg0_K5wykUsdzkX9khf4WpNy5PDsNj
```

Gambar 4. 30 Coding Deklarasi ESP32 dengan IoT

Gambar 4.30 menunjukkan deklarasi awal parameter yang diperlukan untuk konektivitas ESP32 dengan *Platform* IoT. Nilai-nilai seperti BLYNK_TEMPLATE_ID, BLYNK_TEMPLATE_NAME, dan BLYNK_AUTH_TOKEN digunakan untuk menghubungkan perangkat dengan server Blynk agar dapat memantau dan mengendalikan sistem secara *Real-time*. Selain itu, BOT_TOKEN dan CHAT_ID digunakan untuk mengintegrasikan sistem

dengan Bot Telegram, memungkinkan pengiriman notifikasi atau perintah melalui aplikasi Telegram. *URL sheet* merujuk ke Google Apps Script yang digunakan untuk menyimpan data kedalam Spreadsheet secara *online*.

4.4.3 Program untuk Tampilan LCD

```
276  void tampilMenuUtama() {
277    lcd.clear();
278    lcd.setCursor(0, 0); lcd.print("Menu Utama Monitor");
279    lcd.setCursor(0, 1); lcd.print("1 = Green House");
280    lcd.setCursor(0, 2); lcd.print("2 = Monitor Waktu");
281    lcd.setCursor(0, 3); lcd.print("3 = Monitor Tinggi");
282  }
```

```
285  void tampilKesatu() {
286    if (lastKelembaban != persenKelembaban) {
287      // Cetak label di baris 0
288      lcd.setCursor(0, 0);
289      lcd.print("Kelembaban Tanah:");
290
291      // Baris 1, clear lebih pasti
292      lcd.setCursor(0, 1);
293      lcd.print("                    ");  // 20 spasi kalau 20x4
294
295      // Cetak nilai dengan padding tetap
296      lcd.setCursor(0, 1);
297      lcd.print(">> ");
298      if (persenKelembaban < 10) lcd.print(" ");  // Padding untuk 1 digit
299      lcd.print(persenKelembaban);
300      lcd.print(" %");
301
302      lastKelembaban = persenKelembaban;
303    }
304    if (abs(lastLux - lux) >= 1.0) {
305      lcd.setCursor(0, 2);
306      lcd.print("Intensitas Cahaya:");
307
308      lcd.setCursor(0, 3);
309      lcd.print("                    ");  // 20 spasi
310
311      lcd.setCursor(0, 3);
312      lcd.print(">> ");
313      lcd.print((int)lux);
314      lcd.print(" Lux");
315      lastLux = lux;
316    }
```

```
321  void tampilKedua() {
322    DateTime now = rtc.now();
323    // Nama hari
324    String hari = "";
325    switch (now.dayOfTheWeek()) {
326      case 0: hari = "Minggu"; break;
327      case 1: hari = "Senin"; break;
328      case 2: hari = "Selasa"; break;
329      case 3: hari = "Rabu"; break;
330      case 4: hari = "Kamis"; break;
331      case 5: hari = "Jumat"; break;
332      case 6: hari = "Sabtu"; break;
333    }
334    // Baris 0: Hari, tanggal
335    lcd.setCursor(0, 0);
336    lcd.print("                    ");  // clear line
337    lcd.setCursor(0, 0);
338    lcd.print(hari);
339    lcd.print(", ");
340    lcd.print(now.day());
341    lcd.print("/");
342    lcd.print(now.month());
343    lcd.print("/");
344    lcd.print(now.year());
345    // Baris 1: Judul / separator
346    lcd.setCursor(0, 1);
347    lcd.print("Pukul:              "); // bersihkan line
348    // Baris 2: Waktu lengkap dengan detik
349    lcd.setCursor(0, 2);
350    char waktu[21];
351    snprintf(waktu, sizeof(waktu), "    %02d:%02d:%02d WIB    ",
352             now.hour(), now.minute(), now.second());
353    lcd.print(waktu);
```

```
361  void tampilKetiga() {
362    if (abs(tinggiAir - lastAir) >= 1.0) {
363      lcd.setCursor(0, 0);
364      lcd.print("Air      = ");
365      lcd.setCursor(12, 0);
366      lcd.print("     "); // bersihkan nilai lama
367      lcd.setCursor(12, 0);
368      lcd.print(tinggiAir, 0);
369      lcd.print(" cm");
370      lastAir = tinggiAir;
371    }
372
373    if (abs(tinggiPupuk - lastPupuk) >= 1.0) {
374      lcd.setCursor(0, 1);
375      lcd.print("Pupuk    = ");
376      lcd.setCursor(12, 1);
377      lcd.print("     ");
378      lcd.setCursor(12, 1);
379      lcd.print(tinggiPupuk, 0);
380      lcd.print(" cm");
381      lastPupuk = tinggiPupuk;
382    }
383
384    if (abs(tinggiPestisida - lastPestisida) >= 1.0) {
385      lcd.setCursor(0, 2);
386      lcd.print("Pestisida= ");
387      lcd.setCursor(12, 2);
388      lcd.print("     ");
389      lcd.setCursor(12, 2);
390      lcd.print(tinggiPestisida, 0);
391      lcd.print(" cm");
392      lastPestisida = tinggiPestisida;

395    lcd.setCursor(0, 3);
396    lcd.print("4 = Menu Utama     ");
397  }
```

Gambar 4. 31 Program Empat Kondisi Tampilan LCD

Pada Gambar 4.31, ditampilkan empat jenis tampilan utama pada LCD sistem *Smart Greenhouse*. Tampilan pertama (void tampilMenuUtama) menampilkan menu navigasi utama yang memungkinkan pengguna memilih fitur *monitoring* yang tersedia. Tampilan kedua (void tampilKesatu) menyajikan data kelembaban tanah dan intensitas cahaya secara *Real-time*. Tampilan ketiga (void tampilKedua) berfungsi sebagai monitor waktu yang menampilkan hari, tanggal, dan jam saat ini berdasarkan modul RTC. Sedangkan tampilan keempat (void tampilKetiga) digunakan untuk menampilkan tinggi permukaan cairan, yaitu air, pupuk, dan pestisida, masing-masing dalam satuan centimeter. Setiap tampilan dirancang

dengan pemutakhiran nilai secara dinamis agar informasi yang ditampilkan selalu akurat dan terkini.

4.4.4 Program Sensor pada Arduino IDE

Arduino IDE berperan sebagai *Platform* pemrograman yang digunakan untuk mengembangkan dan mengunggah kode ke mikrokontroler. Dengan memanfaatkan pustaka (*library*) yang telah dipanggil sebelumnya, sistem ini memungkinkan pemrograman berbagai fungsi pembacaan sensor secara terintegrasi. Dalam hal ini, kode yang ditulis berfungsi untuk membaca data dari beberapa sensor penting, seperti sensor cahaya BH1750, sensor kelembaban tanah, sensor hujan, dan sensor ultrasonik untuk mengukur tinggi permukaan cairan (air, pupuk, dan pestisida). Setiap sensor diprogram agar data yang diperoleh dapat dikonversi menjadi satuan yang bermakna, seperti persentase atau centimeter, lalu ditampilkan melalui serial monitor sebagai dasar pemantauan dan pengambilan keputusan dalam sistem *Smart Greenhouse*.

```
void sensorbh1750(){
  lux = lightMeter.readLightLevel();
  Serial.print("Light: ");
  Serial.print(lux);
  Serial.println(" lx");
}

void sensorTanah() {
  delay(10);  // Kasih waktu ADC settle
  nilaiADCtanah = analogRead(PIN_SENSOR_TANAH);
  delay(10);

  persenKelembaban = map(nilaiADCtanah, 4095, 1500, 0, 100);
  persenKelembaban = constrain(persenKelembaban, 0, 100);

  Serial.print("ADC Tanah: ");
  Serial.print(nilaiADCtanah);
  Serial.print(" => ");
  Serial.print(persenKelembaban);
  Serial.println("%");
}
```

```
void sensorhujan() {
  int adcValue = analogRead(PIN_SENSOR_HUJAN);
  persentaseAir = map(adcValue, hujan_kering, hujan_basah, 0, 100);  // Kering = 0%, Basah = 100%
  persentaseAir = constrain(persentaseAir, 0, 100);

  Serial.print("ADC: ");
  Serial.print(adcValue);
  Serial.print(" | Intensitas Hujan: ");
  Serial.print(persentaseAir);
  Serial.println(" %");
}

float readUltrasonic(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

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

  long duration = pulseIn(echoPin, HIGH, 30000); // Timeout 30ms
  float distance = duration * SOUND_SPEED / 2;
  return distance;
}

void sensortinggi() {
  // Baca jarak dari sensor ke permukaan cairan
  float jarakAir = readUltrasonic(trigPin1, echoPin1);
  float jarakPupuk = readUltrasonic(trigPin2, echoPin2);
  float jarakPestisida = readUltrasonic(trigPin4, echoPin4);

  // Hitung tinggi cairan aktual dalam cm
  tinggiAir = TINGGI_WADAH_CM - jarakAir;
  tinggiPupuk = TINGGI_WADAH_CM - jarakPupuk;
  tinggiPestisida = TINGGI_WADAH_CM - jarakPestisida;

  // Batasi ke minimum 0 cm jika hasil negatif
  tinggiAir = tinggiAir < 0 ? 0 : tinggiAir;
  tinggiPupuk = tinggiPupuk < 0 ? 0 : tinggiPupuk;
  tinggiPestisida = tinggiPestisida < 0 ? 0 : tinggiPestisida;

  Serial.print("Tinggi Air: "); Serial.print(tinggiAir); Serial.println(" cm");
  Serial.print("Tinggi Pupuk: "); Serial.print(tinggiPupuk); Serial.println(" cm");
  Serial.print("Tinggi Pestisida: "); Serial.print(tinggiPestisida); Serial.println(" cm");
}
```

Gambar 4. 32 Program Sensor pada Arduino IDE

Pada Gambar 4.32, ditampilkan proses pembacaan data dari berbagai sensor yang digunakan dalam sistem. Fungsi sensorbh1750() membaca intensitas cahaya menggunakan sensor BH1750 dan menampilkannya dalam satuan lux. Fungsi sensorTanah() melakukan pembacaan nilai analog dari sensor kelembaban tanah, kemudian mengonversinya menjadi persentase kelembaban menggunakan fungsi map() dan constrain() agar hasil tetap dalam rentang 0–100%. Fungsi sensorhujan() membaca intensitas hujan melalui sensor analog dan mengubah nilai ADC menjadi

persentase kelembapan permukaan, yang menggambarkan kondisi cuaca basah atau kering. Sementara itu, fungsi sensortinggi() digunakan untuk mengukur tinggi permukaan cairan (air, pupuk, dan pestisida) dalam wadah dengan memanfaatkan sensor ultrasonik. Selisih antara tinggi wadah dan jarak yang terbaca dari sensor digunakan untuk menentukan tinggi aktual cairan, serta dikoreksi agar tidak bernilai negatif. Semua hasil pembacaan ditampilkan ke serial monitor untuk pemantauan secara langsung.

4.4.5 Progam Notifikasi Kondisi Alat *Smart Greenhouse*

```
void notifikasitinggi(){
  // Notifikasi Air
  if (tinggiAir < 5.0) {
  digitalWrite(RELAY_LAMPU_MERAH, LOW);
  if (!notifAirSent) {
    kirimTelegram("🚨 *PERINGATAN*: Tinggi air rendah! Harap isi ulang tangki air.");
    notifAirSent = true;
    }
  } else {
  notifAirSent = false;
  }

  // Notifikasi Pupuk
  if (tinggiPupuk < 5.0) {
  digitalWrite(RELAY_LAMPU_MERAH, LOW);
  if (!notifPupukSent) {
    kirimTelegram("🚨 *PERINGATAN*: Tinggi pupuk rendah! Harap isi ulang tangki pupuk.");
    notifPupukSent = true;
    }
  } else {
  notifPupukSent = false;
  }

  // Notifikasi Pestisida
  if (tinggiPestisida < 5.0) {
  digitalWrite(RELAY_LAMPU_MERAH, LOW);
  if (!notifPestisidaSent) {
    kirimTelegram("🚨 *PERINGATAN*: Tinggi pestisida rendah! Harap isi ulang tangki pestisida.");
    notifPestisidaSent = true;
  }
  } else {
  notifPestisidaSent = false;
  }

  // Kalau semua normal, matikan lampu merah
  if (tinggiAir >= 5.0 && tinggiPupuk >= 5.0 && tinggiPestisida >= 5.0) {
  digitalWrite(RELAY_LAMPU_MERAH, HIGH);
  }
}
```

```
void kontrolLampuGreenhouse() {
  bool kondisiGelap = lux < AMBANG_GELAP;
  bool kondisiHujan = persentaseAir > 50;

  if ((kondisiHujan || kondisiGelap) && !lampuGHNyala) {
    digitalWrite(RELAY_LAMPU_GH, LOW); // nyalakan lampu Green House
    digitalWrite(RELAY_LAMPU_KUNING, LOW); // nyalakan lampu Kuning
    lampuGHNyala = true;

    // Update status ke Blynk
    Blynk.virtualWrite(V5, 1);

    String alasan = kondisiHujan && kondisiGelap ? "karena *hujan dan gelap*" :
                    kondisiHujan ? "karena *hujan*" : "karena *kondisi gelap*";

    kirimTelegram("💡 *Lampu Greenhouse dinyalakan* " + alasan + ".");
    Serial.println("Lampu dinyalakan " + alasan);
  }
  else if (!kondisiHujan && !kondisiGelap && lampuGHNyala) {
    digitalWrite(RELAY_LAMPU_GH, HIGH); // matikan lampu AC
    digitalWrite(RELAY_LAMPU_KUNING, HIGH); // nyalakan lampu Kuning
    lampuGHNyala = false;

    // Update status ke Blynk
    Blynk.virtualWrite(V5, 0);

    kirimTelegram("🔪 *Lampu Greenhouse dimatikan* karena cuaca terang dan tidak hujan.");
    Serial.println("Lampu dimatikan.");
  }
}
```

Gambar 4. 33 Program Notifikasi Kondisi Alat *Smart Greenhouse*

Pada Gambar 4.33, ditampilkan program notifikasi otomatis yang berfungsi sebagai sistem peringatan dan kontrol pencahayaan dalam *Smart Greenhouse*. Program ini memantau kondisi tinggi cairan pada tiga tangki yaitu air, pupuk, dan pestisida. Jika tinggi salah satu cairan berada di bawah ambang batas (kurang dari 5 cm), maka sistem akan mengaktifkan lampu indikator merah dan mengirimkan pesan peringatan secara otomatis melalui bot Telegram, sehingga pengguna dapat segera melakukan pengisian ulang. Selain itu, fungsi kontrol lampu kontrolLampuGreenhouse() secara cerdas menyalakan lampu di dalam greenhouse jika terdeteksi kondisi gelap atau sedang hujan dengan kondisi sangat mendung, berdasarkan data dari sensor cahaya dan sensor hujan. Sebaliknya, lampu akan dimatikan secara otomatis jika cuaca cerah dan terang. Notifikasi juga dikirim ke Telegram untuk setiap perubahan kondisi lampu, memberikan pengalaman *monitoring* yang responsif dan *Real-time* bagi pengguna.

4.4.6 Melakukan *String Massanger* Pengaturan Setting Waktu RTC Melalui Aplikasi Telegram

```
void prosesSetRTC(String msg) {
  // Format diharuskan: /setrtc <Hari> YYYY-MM-DD HH:MM:SS
  msg.replace("/setrtc ", "");
  msg.trim();

  int spasiIndex = msg.indexOf(' ');
  if (spasiIndex <= 0) {
    kirimTelegram("❌ Format salah! Gunakan:\n/setrtc <Hari> YYYY-MM-DD HH:MM:SS");
    return;
  }

  String hariNama = msg.substring(0, spasiIndex);
  String sisa = msg.substring(spasiIndex + 1);
  sisa.trim();

  // Konversi nama hari ke indeks
  int targetHari = -1;
  if (hariNama.equalsIgnoreCase("Minggu")) targetHari = 0;
  else if (hariNama.equalsIgnoreCase("Senin")) targetHari = 1;
  else if (hariNama.equalsIgnoreCase("Selasa")) targetHari = 2;
  else if (hariNama.equalsIgnoreCase("Rabu")) targetHari = 3;
  else if (hariNama.equalsIgnoreCase("Kamis")) targetHari = 4;
  else if (hariNama.equalsIgnoreCase("Jumat")) targetHari = 5;
  else if (hariNama.equalsIgnoreCase("Sabtu")) targetHari = 6;

  if (targetHari < 0) {
    kirimTelegram("❌ Nama hari tidak dikenali. Gunakan: Senin, Selasa, dst.");
    return;
  }

    rtc.adjust(newTime);

    String konfirmasi = "✅ RTC di-set ke:\n" + hariNama + " " +
                        String(y) + "-" + String(mo) + "-" + String(d) + " " +
                        String(h) + ":" + String(mi) + ":" + String(s);
    kirimTelegram(konfirmasi);
    Serial.println(konfirmasi);
  } else {
    kirimTelegram("❌ Format salah! Gunakan:\n/setrtc <Hari> YYYY-MM-DD HH:MM:SS");
  }
}
```

Gambar 4. 34 Program *String Massanger* Pengaturan Set RTC melalui Aplikasi Telegram

Pada Gambar 4.34, ditampilkan fungsi prosesSetRTC() yang digunakan untuk mengatur waktu dan tanggal secara manual melalui perintah yang dikirim via bot Telegram. Format yang digunakan adalah /setrtc <Hari> YYYY-MM-DD HH:MM:SS, di mana sistem akan memisahkan nama hari dan format waktu. Nama hari dikonversi menjadi indeks (0–6) sesuai standar RTC, lalu dibandingkan dengan tanggal yang dimasukkan untuk memastikan kecocokan. Jika hari dan tanggal tidak

sesuai, sistem akan memberikan peringatan agar pengguna mengecek ulang. Apabila format valid dan data sesuai, waktu RTC akan diperbarui dan notifikasi konfirmasi akan dikirimkan. Fungsi ini memungkinkan pengguna melakukan sinkronisasi waktu secara fleksibel tanpa perlu memprogram ulang perangkat, menjadikan sistem lebih interaktif dan mudah dikendalikan dari jarak jauh.

4.4.7 Pembuatan Program untuk Mengontrol Pompa Pestisida, Pupuk dan Air

```
unsigned long intervalPompa1 = 7 * 60 * 1000UL; // default 7 menit (Pestisida)
unsigned long intervalPompa2 = 6 * 60 * 1000UL; // default 6 menit (Pupuk)
const unsigned long durasiPompa = 20000; // 20 detik

void kontrolPompaPestisida() {
  const unsigned long durasiPompa = 20000;

  unsigned long currentMillis = millis();
  if (currentMillis - lastPompa1 >= intervalPompa1 && !pompa1Aktif) {
    digitalWrite(RELAY_PESTISIDA, LOW);
    pompa1Aktif = true;
    pompa1Mulai = currentMillis;
    lastPompa1 = currentMillis;
    Serial.println("Pompa 1 Aktif (Pestisida)");
  }

  if (pompa1Aktif && currentMillis - pompa1Mulai >= durasiPompa) {
    digitalWrite(RELAY_PESTISIDA, HIGH);
    pompa1Aktif = false;
    Serial.println("Pompa 1 Nonaktif (Pestisida)");
    kirimTelegram("🧴 Penyemprotan pestisida telah dilakukan selama 20 detik.");
  }
}
```

```
void kontrolPompaPupuk() {
  const unsigned long durasiPompa = 20000;

  unsigned long currentMillis = millis();
  if (currentMillis - lastPompa2 >= intervalPompa2 && !pompa2Aktif) {
    digitalWrite(RELAY_PUPUK, LOW);
    pompa2Aktif = true;
    pompa2Mulai = currentMillis;
    lastPompa2 = currentMillis;
    Serial.println("Pompa 2 Aktif (Pupuk)");
  }

  if (pompa2Aktif && currentMillis - pompa2Mulai >= durasiPompa) {
    digitalWrite(RELAY_PUPUK, HIGH);
    pompa2Aktif = false;
    Serial.println("Pompa 2 Nonaktif (Pupuk)");
    kirimTelegram("🌱 Pemupukan telah dilakukan selama 20 detik.");
  }
}

void kontrolPompaAir() {
  if (persenKelembaban < 60 && !pompaAirAktif) {
    digitalWrite(RELAY_AIR, LOW);
    pompaAirAktif = true;
    Serial.println("Pompa Air Aktif (Tanah < 60%)");
    kirimTelegram("💧 *Pompa Air dinyalakan* karena kelembaban tanah < 60%.");
  } else if (persenKelembaban >= 80 && pompaAirAktif) {
    digitalWrite(RELAY_AIR, HIGH);
    pompaAirAktif = false;
    Serial.println("Pompa Air Nonaktif (Tanah >= 80%)");
    kirimTelegram("🔴 *Pompa Air dimatikan* karena kelembaban tanah >= 80%.");
  }
}
```

Gambar 4. 35 Program Kontrolling Pompa Pestisida, Pupuk dan Air

Pada Gambar 4.35, diperlihatkan program kendali otomatis untuk tiga jenis pompa dalam sistem *Smart Greenhouse*, yaitu pompa air, pupuk, dan pestisida. Fungsi kontrolPompaPestisida() dan kontrolPompaPupuk() bekerja berdasarkan interval waktu tertentu, yaitu setiap 6 menit dan 7 menit secara default. Saat waktu terpenuhi, pompa akan diaktifkan selama 20 detik untuk menjalankan proses penyemprotan atau pemupukan, kemudian dinonaktifkan kembali secara otomatis. Notifikasi keberhasilan proses juga dikirimkan ke Telegram agar pengguna mendapatkan laporan waktu nyata. Sementara itu, fungsi kontrolPompaAir() dikendalikan oleh data kelembaban tanah. Pompa air akan menyala secara otomatis jika kelembaban turun di bawah 60%, dan akan mati ketika kelembaban tanah

mencapai 80% atau lebih. Logika ini dirancang agar penggunaan air, pupuk, dan pestisida menjadi efisien dan responsif terhadap kondisi aktual tanaman, tanpa memerlukan intervensi manual secara terus-menerus.

4.4.8 Program Fitur Bot Telegram terhadap *Setting* Kondisi Alat *Smart Greenhouse*

```
void periksaPesanTelegram() {
  static unsigned long lastCheck = 0;
  const unsigned long checkInterval = 3000;

  if (millis() - lastCheck > checkInterval) {
    // ambil update baru sejak terakhir
    int newMessages = bot.getUpdates(bot.last_message_received + 1);
    if (newMessages) {
      for (int i = 0; i < newMessages; i++) {
        String msg = bot.messages[i].text;

        Serial.print("Pesan diterima: ");
        Serial.println(msg);

        if (msg.startsWith("/setrtc")) {
          prosesSetRTC(msg);
        }
        else if (msg == "/resetwifi") {
          for (int j = 0; j < chatCount; j++) {
            bot.sendMessage(chatIDs[j], "🔄 Pengaturan WiFi akan direset. ESP akan restart.", "");
            delay(500);
          }
          wifiManager.resetSettings();
          delay(1000);
          ESP.restart();
        }
        else if (msg.startsWith("/setintervalpompa1 ")) {
          msg.replace("/setintervalpompa1 ", "");
          int menit = msg.toInt();
          if (menit >= 1 && menit <= 1440) {
            intervalPompa1 = menit * 60 * 1000UL;
            kirimTelegram("✅ Interval Pompa 1 (Pestisida) diatur ke " + String(menit) + " menit.");
```

```
      } else {
        kirimTelegram("❌ Interval tidak valid. Gunakan antara 1 - 1440 menit.");
      }
    }
    else if (msg.startsWith("/setintervalpompa2 ")) {
      msg.replace("/setintervalpompa2 ", "");
      int menit = msg.toInt();
      if (menit >= 1 && menit <= 1440) {
        intervalPompa2 = menit * 60 * 1000UL;
        kirimTelegram("✅ Interval Pompa 2 (Pupuk) diatur ke " + String(menit) + " menit.");
      } else {
        kirimTelegram("❌ Interval tidak valid. Gunakan antara 1 - 1440 menit.");
      }
    }

    // ⚠ penting: advance offset supaya tidak diproses ulang
    bot.last_message_received = bot.messages[i].update_id;
  }
 }

 lastCheck = millis();
 }
}
```

Gambar 4. 36 Program Fitur Bot Telegram terhadap Setting Kondisi Alat *Smart Greenhouse*

Pada Gambar 4.36, ditampilkan fungsi periksaPesanTelegram() yang berfungsi untuk memantau dan memproses pesan masuk dari bot Telegram secara berkala setiap 3 detik. Program ini memungkinkan pengguna berinteraksi langsung dengan sistem *Smart Greenhouse* melalui berbagai perintah. Dalam hal ini, pengguna dapat mengatur waktu RTC dengan perintah /setrtc, mereset koneksi WiFi dengan /resetwifi, serta mengubah interval kerja pompa pestisida dan pupuk menggunakan /setintervalpompa1 dan /setintervalpompa2. Sistem akan memvalidasi setiap masukan dan memberikan respons langsung melalui Telegram, baik berupa konfirmasi keberhasilan maupun peringatan jika format atau nilai tidak sesuai. Fungsi ini menjadikan sistem lebih fleksibel dan user-friendly karena kontrol dan konfigurasi dapat dilakukan dari jarak jauh tanpa memerlukan perangkat keras tambahan atau pemrograman ulang.

4.4.9 Program Integrasi Arduino IDE dengan *Datastreams* Blynk

```
void kirimDataBlynk() {
  Blynk.virtualWrite(V0, (int)tinggiAir);
  Blynk.virtualWrite(V1, (int)tinggiPestisida);
  Blynk.virtualWrite(V2, (int)tinggiPupuk);
  Blynk.virtualWrite(V3, persenKelembaban);
  Blynk.virtualWrite(V4, lux);


  // Relay status
  Blynk.virtualWrite(V5, lampuGHNyala ? 1 : 0);
  Blynk.virtualWrite(V7, pompaAirAktif ? 1 : 0);
  Blynk.virtualWrite(V8, pompa2Aktif ? 1 : 0);
  Blynk.virtualWrite(V9, pompa1Aktif ? 1 : 0);

  // RTC time
  DateTime now = rtc.now();
  char waktu[20];
  snprintf(waktu, sizeof(waktu), "%02d:%02d:%02d", now.hour(), now.minute(), now.second());
  char tanggal[20];
  snprintf(tanggal, sizeof(tanggal), "%02d/%02d/%04d", now.day(), now.month(), now.year());

  Blynk.virtualWrite(V10, waktu);
  Blynk.virtualWrite(V11, tanggal);
```

```
  String hari = "";
  switch (now.dayOfTheWeek()) {
    case 0: hari = "Minggu"; break;
    case 1: hari = "Senin"; break;
    case 2: hari = "Selasa"; break;
    case 3: hari = "Rabu"; break;
    case 4: hari = "Kamis"; break;
    case 5: hari = "Jumat"; break;
    case 6: hari = "Sabtu"; break;
  }
  Blynk.virtualWrite(V12, hari);

  String statusHujan;
  if (persentaseAir > 50) {
  statusHujan = "Hujan (" + String(persentaseAir) + "%)";
  } else {
  statusHujan = "Tidak Hujan (" + String(persentaseAir) + "%)";
  }
  Blynk.virtualWrite(V6, statusHujan);

}
```

```
BLYNK_WRITE(V5) {
  int pinValue = param.asInt();
  if (pinValue) {
    digitalWrite(RELAY_LAMPU_GH, LOW);
    digitalWrite(RELAY_LAMPU_KUNING, LOW); // nyalakan lampu Kuning
    lampuGHNyala = true;
    kirimTelegram("💡 *Lampu Green House dinyalakan* dari Blynk.");
  } else {
    digitalWrite(RELAY_LAMPU_GH, HIGH);
    digitalWrite(RELAY_LAMPU_KUNING, HIGH); // nyalakan lampu Kuning
    lampuGHNyala = false;
    kirimTelegram("🔦 *Lampu Green House dimatikan* dari Blynk.");
  }
}

BLYNK_WRITE(V7) {  // Pompa Air manual via Blynk
  int val = param.asInt();
  if (val) {
    digitalWrite(RELAY_AIR, LOW);
    pompaAirAktif = true;
    kirimTelegram("💧 *Pompa Air dinyalakan* via Blynk.");
  } else {
    digitalWrite(RELAY_AIR, HIGH);
    pompaAirAktif = false;
    kirimTelegram("🔴 *Pompa Air dimatikan* via Blynk.");
  }
}
```

```
BLYNK_WRITE(V8) {  // Pompa Pupuk manual
  int val = param.asInt();
  if (val) {
    digitalWrite(RELAY_PUPUK, LOW);
    pompa2Aktif = true;
    kirimTelegram("🌱 *Pompa Pupuk dinyalakan* via Blynk.");
  } else {
    digitalWrite(RELAY_PUPUK, HIGH);
    pompa2Aktif = false;
    kirimTelegram("🔴 *Pompa Pupuk dimatikan* via Blynk.");
  }
}

BLYNK_WRITE(V9) {  // Pompa Pestisida manual
  int val = param.asInt();
  if (val) {
    digitalWrite(RELAY_PESTISIDA, LOW);
    pompa1Aktif = true;
    kirimTelegram("🧴 *Pompa Pestisida dinyalakan* via Blynk.");
  } else {
    digitalWrite(RELAY_PESTISIDA, HIGH);
    pompa1Aktif = false;
    kirimTelegram("🔴 *Pompa Pestisida dimatikan* via Blynk.");
  }
}
```

Gambar 4. 37 Program Integrasi Arduino IDE dengan *Datasreams* Blynk

Pada Gambar 4.37, ditunjukkan program integrasi antara Arduino IDE dan *Platform* Blynk yang memungkinkan sistem *Smart Greenhouse* mengirim dan menerima data secara *Real-time* melalui *Virtual Pin* (datastreams). Fungsi kirimDataBlynk() digunakan untuk mengirimkan berbagai data sensor ke aplikasi Blynk, seperti tinggi cairan (air, pupuk, pestisida), kelembaban tanah, intensitas cahaya, serta informasi waktu dan status cuaca. Selain itu, status pompa dan lampu juga dikirim untuk memudahkan pemantauan dari jarak jauh. Sementara itu, bagian BLYNK_WRITE() memungkinkan pengguna untuk mengendalikan perangkat secara manual melalui aplikasi, seperti menyalakan atau mematikan pompa dan lampu greenhouse. Setiap aksi manual akan disertai dengan pengiriman notifikasi melalui Telegram untuk memberikan transparansi dan kontrol penuh kepada

pengguna. Integrasi ini menjadikan sistem lebih interaktif, responsif, dan mudah diakses melalui perangkat mobile berbasis *Internet of Things (IoT)*.

4.4.10 Program Integrasi Arduino IDE dengan Database Spreadsheet

```
void kirimKeSpreadsheet() {
  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(sheetURL);
    http.addHeader("Content-Type", "application/json");

    DateTime now = rtc.now();
    char waktu[10];
    snprintf(waktu, sizeof(waktu), "%02d:%02d:%02d", now.hour(), now.minute(), now.second());
    char tanggal[15];
    snprintf(tanggal, sizeof(tanggal), "%02d/%02d/%04d", now.day(), now.month(), now.year());

    String hari;
    switch (now.dayOfTheWeek()) {
      case 0: hari = "Minggu"; break;
      case 1: hari = "Senin"; break;
      case 2: hari = "Selasa"; break;
      case 3: hari = "Rabu"; break;
      case 4: hari = "Kamis"; break;
      case 5: hari = "Jumat"; break;
      case 6: hari = "Sabtu"; break;
    }

    String kondisiLampu = lampuGHNyala ? "Nyala" : "Mati";
    String kondisiPompaAir = pompaAirAktif ? "Nyala" : "Mati";
    String kondisiPompaPupuk = pompa2Aktif ? "Nyala" : "Mati";
    String kondisiPompaPestisida = pompa1Aktif ? "Nyala" : "Mati";
    String kondisiHujan = persentaseAir > 50 ? "Hujan" : "Tidak Hujan";
```

```
  String json = "{";
  json += "\"hari\":\"" + hari + "\",";
  json += "\"tanggal\":\"" + String(tanggal) + "\",";
  json += "\"pukul\":\"" + String(waktu) + "\",";
  json += "\"lux\":\"" + String((int)lux) + "\",";
  json += "\"kelembaban\":\"" + String(persenKelembaban) + "\",";
  json += "\"tinggi_air\":\"" + String((int)tinggiAir) + "\",";
  json += "\"tinggi_pupuk\":\"" + String((int)tinggiPupuk) + "\",";
  json += "\"tinggi_pestisida\":\"" + String((int)tinggiPestisida) + "\",";
  json += "\"lampu\":\"" + kondisiLampu + "\",";
  json += "\"pompa_air\":\"" + kondisiPompaAir + "\",";
  json += "\"pompa_pupuk\":\"" + kondisiPompaPupuk + "\",";
  json += "\"pompa_pestisida\":\"" + kondisiPompaPestisida + "\",";
  json += "\"kondisi_hujan\":\"" + kondisiHujan + "\"";
  json += "}";

  int httpCode = http.POST(json);
  if (httpCode > 0) {
    Serial.println("Spreadsheet OK: " + String(http.getString()));
  } else {
    Serial.println("Gagal kirim data ke Spreadsheet");
  }

  http.end();
}
}
```

Gambar 4. 38 Program Integrasi Arduino IDE dengan Database Spreadsheet

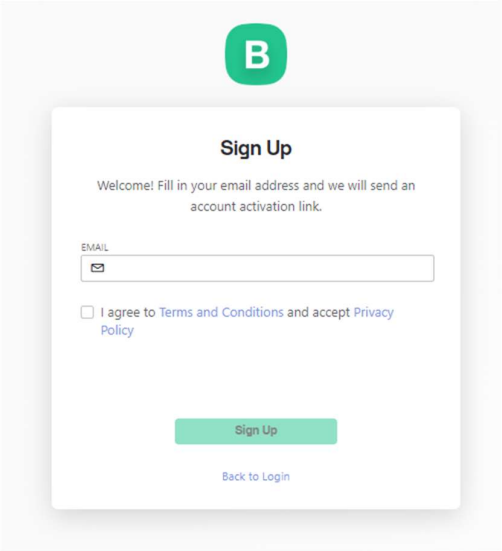
Pada Gambar 4.38, ditampilkan fungsi kirimKeSpreadsheet() yang berperan mengirimkan data hasil *Monitoring Smart Greenhouse* secara berkala ke database spreadsheet berbasis cloud melalui protokol HTTP. Dengan koneksi WiFi yang aktif, fungsi ini menyusun data sensor seperti intensitas cahaya, kelembaban tanah, tinggi cairan (air, pupuk, pestisida), serta status perangkat seperti lampu dan pompa dalam format JSON yang kemudian dikirim ke URL spreadsheet yang telah disiapkan. Informasi waktu lengkap, termasuk hari, tanggal, dan jam, juga dikirimkan untuk memastikan data terekam secara akurat sesuai waktu nyata. Respon server dicatat melalui serial monitor sebagai indikator keberhasilan pengiriman data. Dengan integrasi ini, pengguna dapat memantau dan menyimpan riwayat kondisi greenhouse secara efisien dan terstruktur, mendukung analisis data jangka panjang dan pengambilan keputusan berbasis data.

4.4.11 Pembuatan *Template* Alat pada Aplikasi Blynk

Langkah pembuatan *Dashboard* Blynk adalah sebagai berikut :

1. Pembuatan Akun Blynk

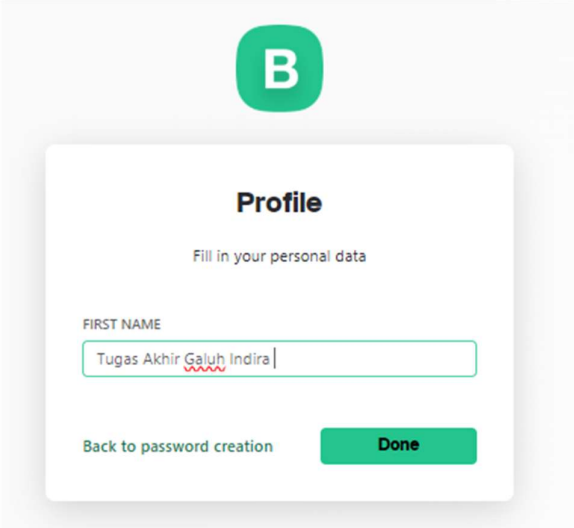
   Langkah pertama dalam pengembangan perangkat lunak *monitoring* dan *controlling* berbasis IoT ini adalah membuat akun Blynk. Akun ini diperlukan agar pengguna dapat mengakses *Dashboard*, membuat template, serta menghubungkan perangkat ESP32 ke *Platform* Blynk. Proses pendaftaran dapat dilakukan melalui aplikasi Blynk di perangkat Android/iOS atau melalui situs web resminya dengan menggunakan alamat email aktif. Tampilan awal bisa dilihat pada Gambar 4.39.



Gambar 4. 39 Tampilan Sign-Up Blynk

2. Pembuatan Profile Blynk

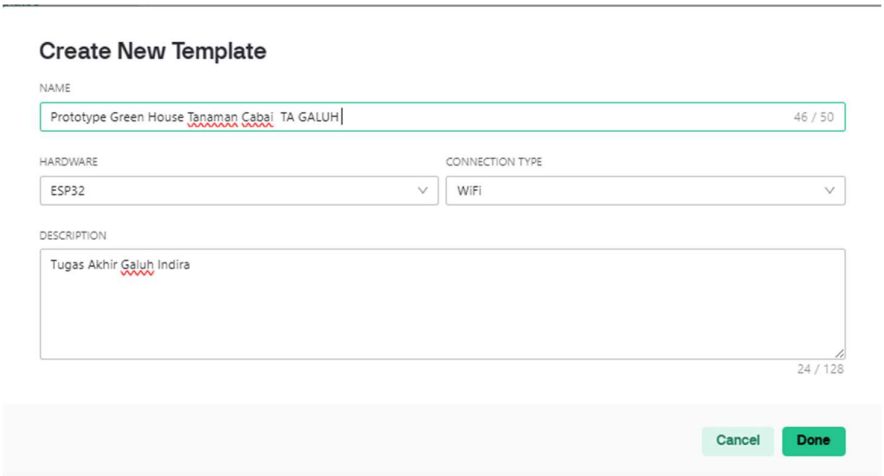
   Setelah berhasil membuat akun, tahap selanjutnya adalah membuat profil pengguna di dalam *Platform* Blynk. Profil ini akan menjadi identitas utama pengguna dalam pengelolaan proyek-proyek IoT, termasuk pengaturan template, device, dan datastreams. Profil ini juga menyimpan informasi pengguna serta keterkaitannya dengan proyek yang sedang dikerjakan. Tampilan selanjutnya yaitu seperti Gambar 4.40.



Gambar 4. 40 Pembuatan Profile pada Aplikasi Blynk

3. Pembuatan Template pada Blynk

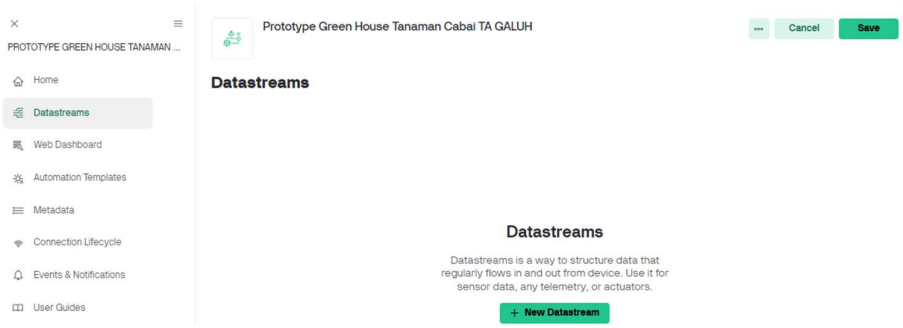
   Template adalah kerangka kerja utama di Blynk untuk mendefinisikan fungsi, tampilan, dan data dari perangkat yang akan dikendalikan. Dalam tahap ini, pengguna membuat template dengan memberikan nama, memilih jenis perangkat (misalnya ESP32), dan mendefinisikan mode koneksi seperti WiFi. Template yang tedapat pada Gambar 4.41 inilah yang nantinya digunakan sebagai dasar dalam membuat project *monitoring* dan *controlling* secara *Real-time*.



Gambar 4. 41 Pembuatan Template pada Aplikasi Blynk

4. Pembuatan Datastreams

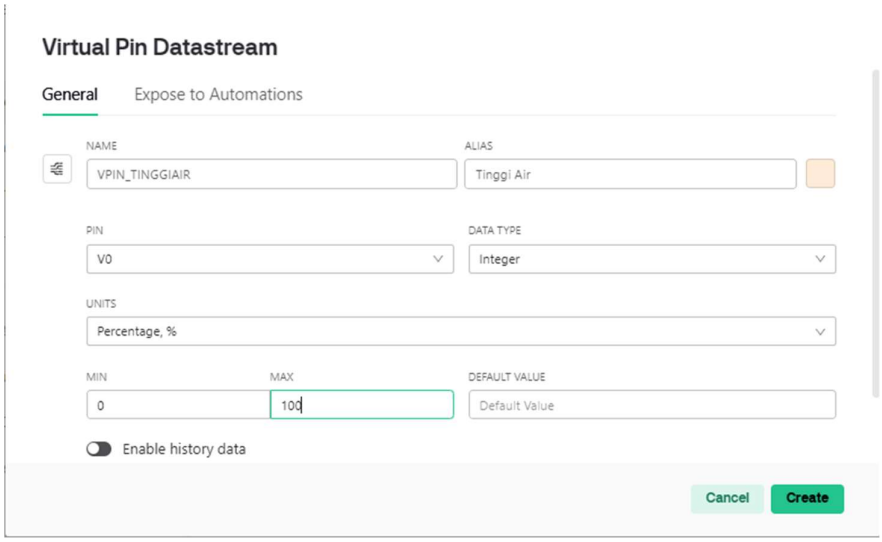
   Datastream merupakan jalur data virtual yang menghubungkan antara sensor atau kontrol fisik dengan tampilan digital di Blynk. Pada tahap ini, dilakukan pembuatan datastreams untuk masing-masing parameter seperti kelembaban tanah, status pompa, cuaca (hujan), serta kontrol untuk menghidupkan atau mematikan pompa atau lampu. Setiap datastream dikonfigurasi dengan tipe data yang sesuai seperti Integer, Boolean, atau String dan dikaitkan dengan *Virtual Pin* (VPin). Tampilan Pembuatan *Datastreams* pada Aplikasi Blynk bisa dilihat pada Gambar 4.2.



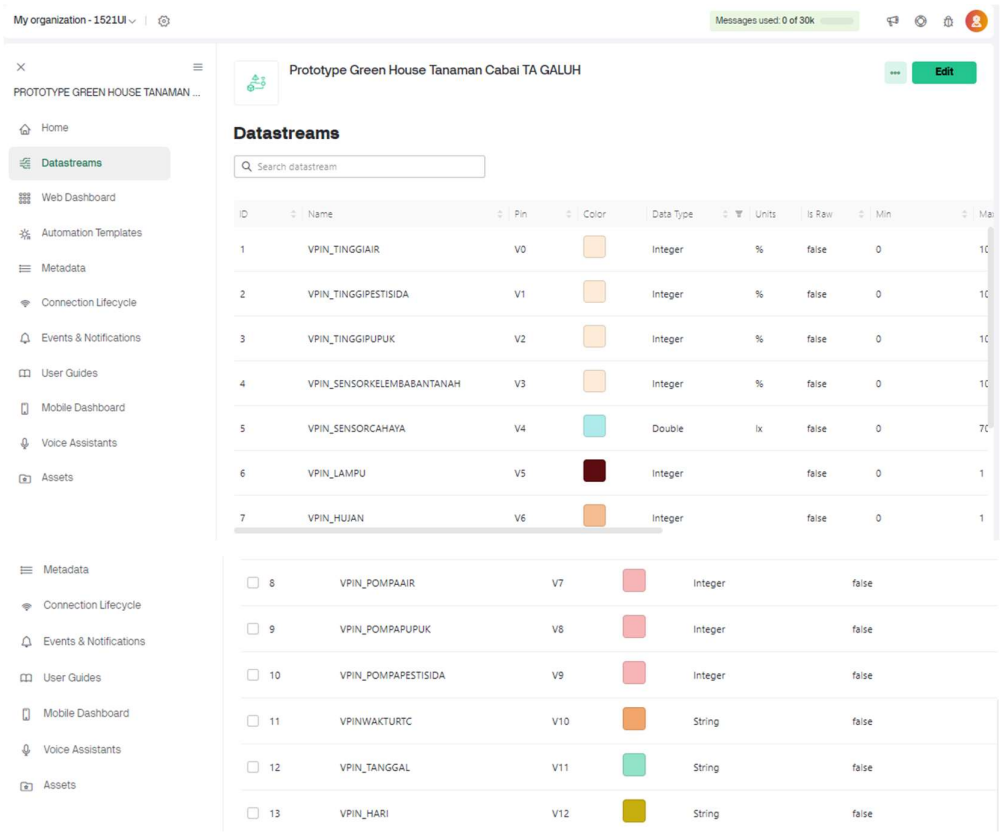
Gambar 4. 42 Tampilan Pilihan Pembuatan *Datastreams* pada Aplikasi Blynk

5. Membuat Datastreams dari Masing – Masing Kondisi *Monitoring & Controlling* Blynk

   Datastream tidak hanya dibuat secara umum, tetapi secara khusus dipecah berdasarkan setiap kondisi yang dibutuhkan dalam sistem *monitoring* dan *controlling*. Contohnya, dibuat datastream khusus untuk mendeteksi kelembaban di bawah ambang batas, datastream untuk status hujan dari sensor, hingga datastream yang mengaktifkan pompa berdasarkan perintah dari aplikasi. Hal ini dilakukan agar sistem dapat merespon kondisi lingkungan secara otomatis maupun manual melalui *Dashboard* Blynk. Untuk melihat tampilan keseluruhan *Datastreams* terdapat pada Gambar 4.43 dan 4.44.



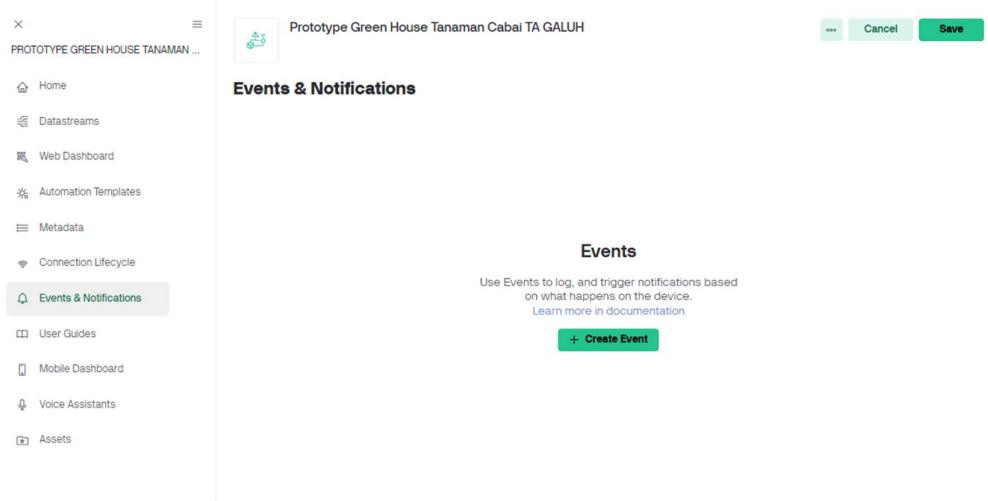
Gambar 4. 43 Pembuatan *Datastreams* pada Aplikasi Blynk dengan *Virtual Pin*



Gambar 4. 44 Tampilan Keseluruhan *Datastreams* yang Telah Dibuat

6. Pembuatan *Setting* Notifikasi

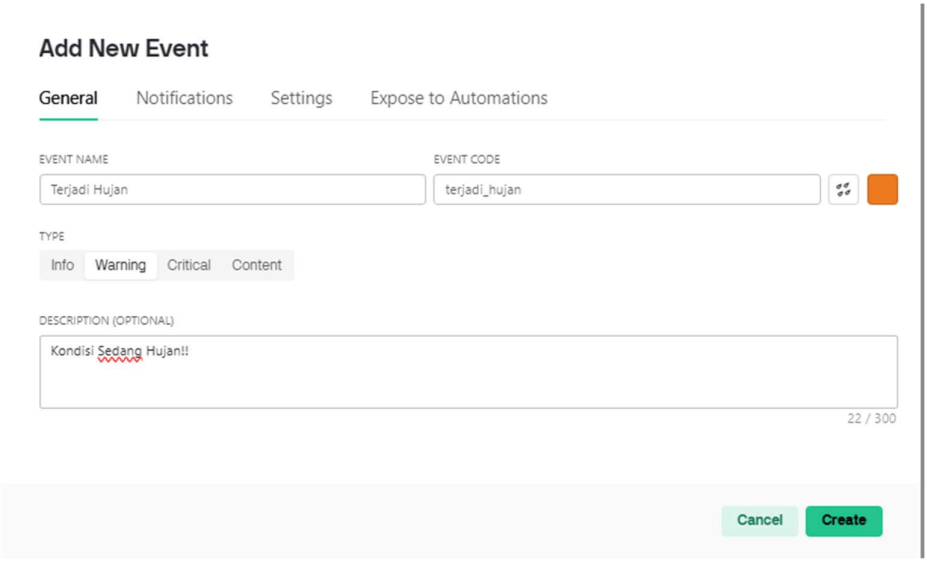
   Untuk meningkatkan interaksi pengguna dengan sistem, ditambahkan pengaturan notifikasi ketika terdeteksi kondisi hujan. Pengguna dapat membuat event notifikasi pada Blynk yang akan aktif ketika sensor hujan mendeteksi adanya air, lalu dikaitkan dengan datastream tertentu. Dengan demikian, pengguna akan mendapat pemberitahuan langsung di aplikasi tanpa harus selalu memantau *Dashboard*. Gambar 4.45 yang terdapat di bawah ini menunjukan tampilan setting *Event & Notifications* pada Aplikasi Blynk.



Gambar 4. 45 Tampilan Setting *Event & Notifications* pada Aplikasi Blynk

7. Penambahan Event Hujan

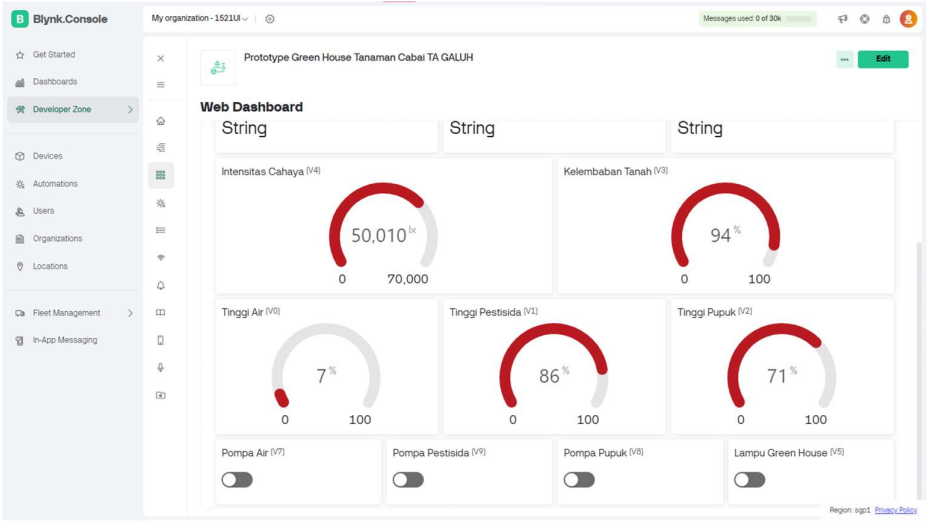
   Setelah notifikasi diaktifkan, tahap lanjutan adalah menambahkan event khusus bernama "Hujan" di dalam template. Event ini akan berfungsi sebagai pemicu otomatis yang mengirimkan notifikasi ke pengguna ketika kondisi cuaca tidak memungkinkan untuk melakukan penyiraman. Pengaturan ini menambah kecerdasan sistem untuk menghindari penyiraman yang tidak efisien. Setelah menambahkan event hujan maka tampilan yang terdapat pada Blynk terlihat seperti Gambar 4.46.



Gambar 4. 46 Penambahan *Event* Notifikasi Hujan pada Aplikasi Blynk

8. Pembuatan *Dashboard* Blynk Web

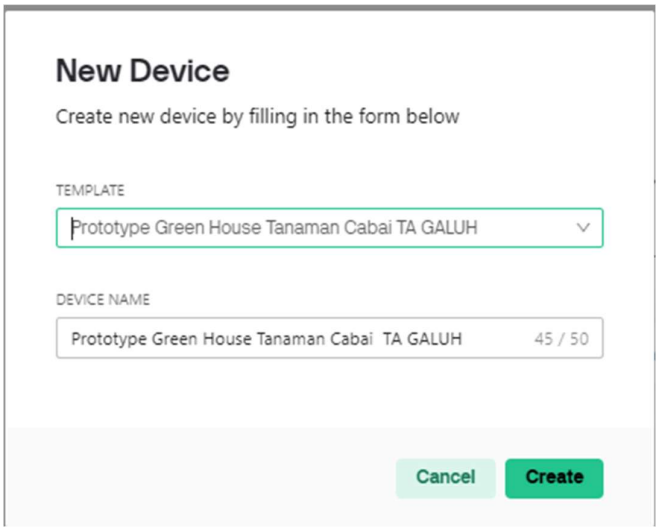
   Dashboard di Blynk Web berfungsi untuk memvisualisasikan seluruh data dan kontrol perangkat yang telah dibuat. Dalam tahap ini, dilakukan penambahan widget seperti gauge, value display, button, dan *Notification*, yang masing-masing dikaitkan dengan datastream terkait. Desain *Dashboard* dibuat sesederhana mungkin namun informatif agar pengguna dapat dengan mudah membaca data sensor dan mengontrol perangkat. Adapun tampilan *Dashboard* yang sudah dibuat bisa dilihat pada Gambar 4.47.



Gambar 4. 47 Tampilan *Dashboard* pada Aplikasi Blynk

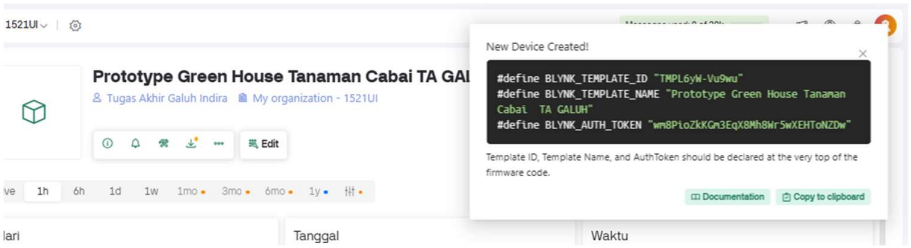
9. Penambahan *Device* ESP32 pada Blynk

   Setelah template dan *Dashboard* siap, dilakukan penambahan perangkat (device) ke dalam project. Perangkat ini adalah representasi fisik dari ESP32 yang akan berkomunikasi dengan *Platform* Blynk. Saat device ditambahkan, sistem akan memberikan Auth Token, Device ID, dan Template ID yang akan digunakan dalam pemrograman mikrokontroler. Tampilan dari penambahan Device ESP 32 terdapat pada Gambar 4.48.



Gambar 4. 48 Penambahan *Device* ESP32 pada Aplikasi Blynk

10. Hubungkan Blynk dengan ESP32 berdasarkan Kode yang diberikan Blynk ke coding ESP32
11. Agar ESP32 dapat berkomunikasi dengan Blynk, kode program harus disesuaikan dengan memasukkan Template ID, Device Name, dan Auth Token yang diberikan oleh Blynk. Selain itu, ESP32 juga harus dikonfigurasikan dengan koneksi WiFi yang stabil agar dapat mengirim dan menerima data dari server Blynk secara *Real-time*. Untuk melihat Kode yang diberikan Blynk ke coding ESP32 terdapat pada Gambar 4.49



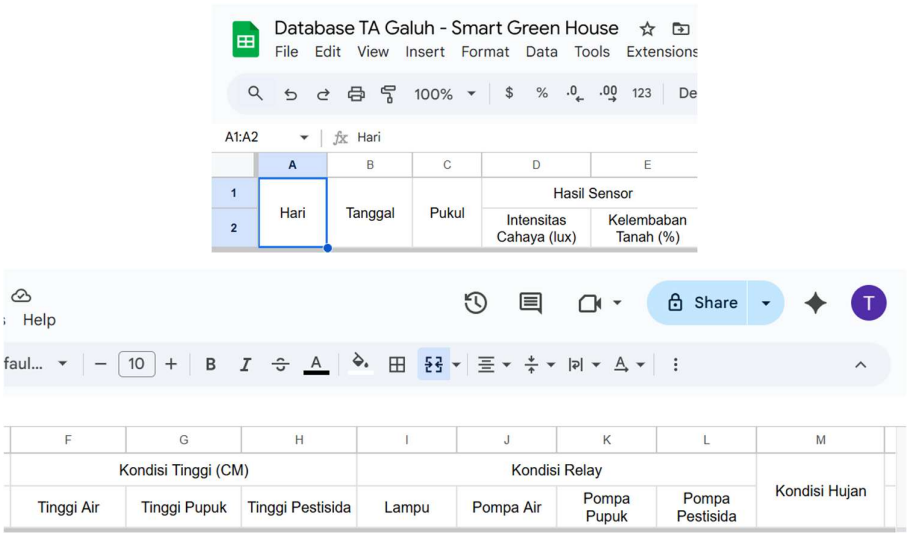
Gambar 4. 49 Integrasi *Device* ESP32 Blynk

4.4.12 Pengiriman Database ke Spreadsheet

Langkah pengiriman database ke Spreadsheet adalah sebagai berikut:

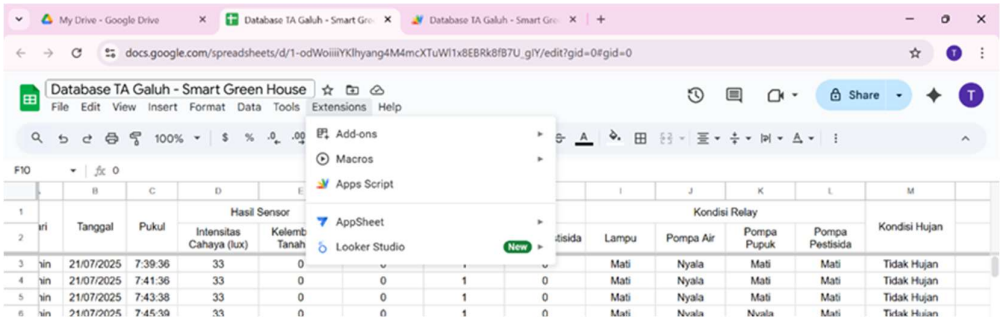
1. Pertama, tentukan parameter apa yang akan ditampilkan dalam database. Tuliskan nama parameter di baris pertama sebagai judul kolom. Kolom-kolom awal pada spreadsheet, yakni Hari, Tanggal, dan Pukul, digunakan untuk mencatat informasi waktu secara *Real-time* setiap kali data dari *Greenhouse* direkam. Bagian Hasil Sensor mencakup Intensitas Cahaya (lux) dan Kelembaban Tanah (%), yang merupakan hasil pembacaan dari sensor BH1750 dan sensor kelembapan tanah tipe resistif. Sementara itu, bagian Kondisi Tinggi (CM) mencatat Tinggi Air, Tinggi Pupuk, dan Tinggi Pestisida berdasarkan data dari sensor level yang dipasang di masing-masing tangki, untuk mengetahui jumlah cairan yang tersedia. Pada bagian Kondisi Relay, terdapat kolom untuk Lampu, Pompa Air, Pompa Pupuk, dan Pompa Pestisida, yang menunjukkan status aktif atau tidaknya perangkat yang dikontrol oleh ESP32. Terakhir, kolom Kondisi Hujan digunakan untuk mencatat keberadaan

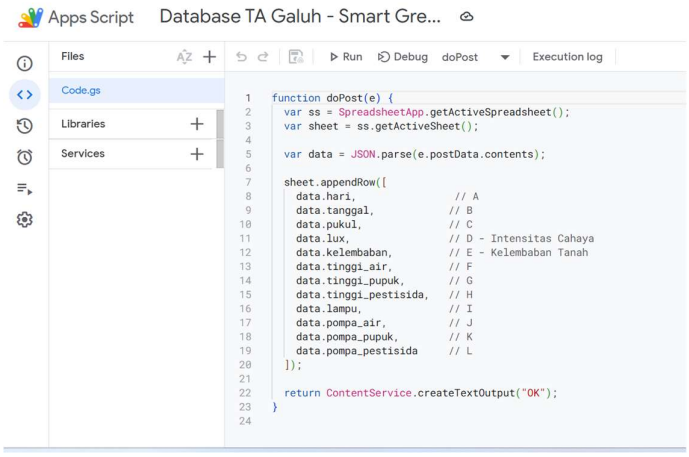
hujan sebagai parameter tambahan dalam sistem penyiraman otomatis. Penyusunan header ini dirancang secara terstruktur agar mempermudah proses *monitoring*, interpretasi data, serta pembuatan laporan dari sistem *Greenhouse* yang dipantau. Gambar 4.50 menunjukkan tampilan parameter yang terdapat di Spreadsheet



Gambar 4. 50 Pembuatan Parameter *Monitoring* untuk Database pada Spreadsheet

2. Selanjutnya untuk *monitoring* ESP 32 dapat klik Extensions → klik apps script

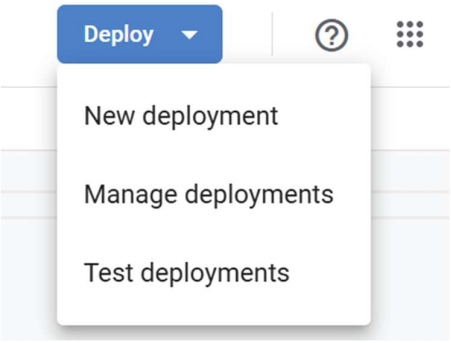




Gambar 4. 51 Membuat Program untuk Spreadsheet melalui Google Apps Script

Script Google Apps yang terdapat pada Gambar 4.51 berfungsi untuk menghubungkan perangkat IoT seperti ESP32 dengan Google Spreadsheet, memungkinkan pencatatan data secara otomatis melalui metode HTTP POST. Fungsi doPost(e) akan dijalankan ketika spreadsheet menerima data dalam bentuk JSON yang dikirim dari perangkat eksternal. Pada bagian awal, variabel ss dipakai untuk mengakses spreadsheet aktif, dan sheet digunakan untuk mengambil sheet aktif tempat data akan disimpan. Data yang diterima kemudian diubah dari format JSON menjadi objek JavaScript menggunakan perintah JSON.parse(e.postData.contents). Setelah itu, fungsi sheet.appendRow([...]) menambahkan satu baris baru ke spreadsheet yang berisi data dari parameter seperti hari, tanggal, pukul, intensitas cahaya (lux), kelembapan tanah, tinggi air, serta status perangkat seperti pompa dan lampu. Setiap nilai mewakili pembacaan sensor atau kondisi relay dari sistem *monitoring Greenhouse*. Terakhir, ContentService.createTextOutput("OK") akan mengirim balasan sebagai konfirmasi bahwa data berhasil diproses. Dengan script ini, proses pencatatan data menjadi otomatis dan efisien, sangat cocok untuk sistem *monitoring* berbasis IoT yang membutuhkan pembaruan data secara terus-menerus.

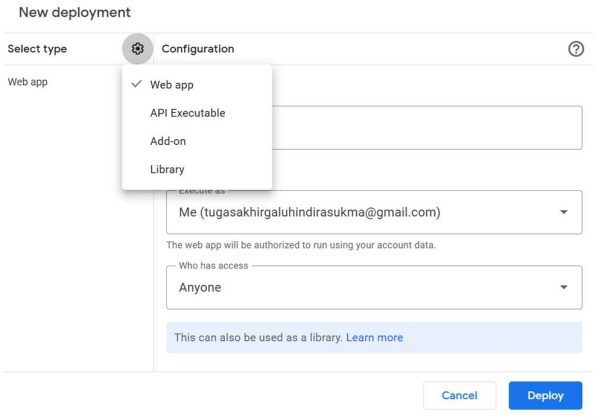
3. Setelah menyelesaikan penulisan kode pada Google Apps Script dan memastikan fungsinya berjalan dengan baik, langkah berikutnya adalah memilih opsi "New *Deployment*" pada menu Deploy.



Gambar 4. 52 Melakukan Pembuatan *Deployment* pada Google Apps Script

Opsi Melakukan Pembuatan *Deployment* pada Google Apps Script seperti Gambar 4.52 digunakan untuk membuat versi pertama dari script yang siap digunakan dan diakses oleh perangkat luar, seperti ESP32. Saat memilih "New *Deployment*", pengguna akan diminta untuk menentukan jenis *Deployment*, seperti Web App, yang umum dipakai untuk menerima data dari perangkat IoT.

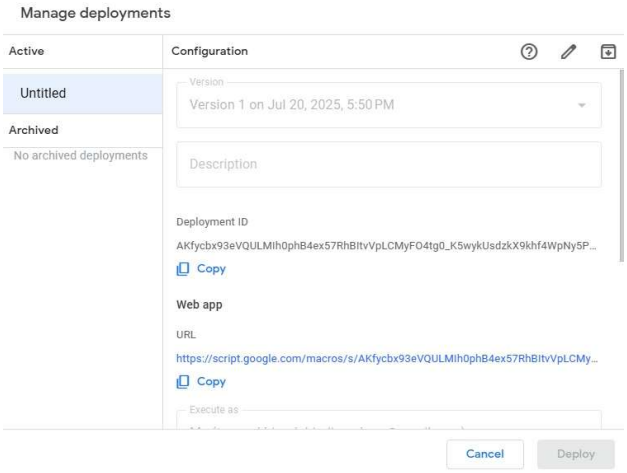
4. Setelah memilih opsi New *Deployment*, langkah selanjutnya adalah menentukan jenis *Deployment* yang akan digunakan, di mana pengguna perlu memilih Web app sebagai tipe utama.



Gambar 4. 53 Pembuatan *Deployment* dengan Tipe Web App

Pemilihan Web app pada Gambar 4.53 sangat penting karena memungkinkan script Google Apps untuk diakses melalui URL, sehingga perangkat IoT seperti ESP32 dapat mengirim data secara langsung menggunakan permintaan HTTP POST. Pada bagian konfigurasi ini, pengguna harus mengisi nama *Deployment* dan memastikan bahwa script dijalankan oleh akun pribadi (misalnya "Me") agar memiliki hak akses penuh terhadap spreadsheet. Kemudian pada bagian Who has access, perlu dipilih opsi "Anyone" agar URL endpoint bisa diakses oleh perangkat dari luar sistem Google tanpa perlu login akun.

5. Setelah proses Web App berhasil dideploy, pengguna akan diarahkan ke halaman Manage *Deployment*s, seperti yang terlihat pada gambar. Di dalam halaman ini terdapat informasi penting yaitu *Deployment* ID, yang merupakan kode unik yang mengidentifikasi versi spesifik dari *Deployment* script tersebut. *Deployment* ID ini bersifat otomatis dan tidak berubah, sehingga sangat berguna untuk pelacakan atau integrasi lanjutan, terutama jika script akan digunakan dalam skala besar atau dipanggil oleh aplikasi pihak ketiga. Selain *Deployment* ID, pengguna juga mendapatkan URL Web App yang berfungsi sebagai endpoint untuk menerima data dari perangkat IoT seperti ESP32.



Gambar 4. 54 Tampilan URL Hasil *Deployment pada* Google Apps Script

URL yang terdapat pada Gambar 4.54 dapat langsung digunakan dalam permintaan HTTP POST agar data dari sensor dapat masuk ke spreadsheet. Dengan adanya *Deployment* ID, pengguna juga bisa memantau versi script yang aktif dan mengelolanya secara efisien melalui fitur pembaruan atau rollback jika diperlukan. Hal ini memastikan bahwa sistem *monitoring Greenhouse* tetap berjalan stabil dan terdokumentasi dengan baik.

6. Selanjutnya link yang terdapat pada tampilan diatas, dimasukan dalam codingan pada Arduino Ide. Link dimasukkan pada program Arduino IDE sebagai mana yang ditampilkan pada gambar 4.30. Potongan kode tersebut merupakan bagian dari fungsi kirimKeSpreadsheet() pada gambar 4.38 yang digunakan untuk mengirim data dari mikrokontroler (seperti ESP32) ke Google Spreadsheet melalui koneksi internet. Pertama, kode memeriksa apakah perangkat telah terhubung ke jaringan WiFi dengan WiFi.status() == WL_CONNECTED. Jika terhubung, maka objek HTTPClient digunakan untuk memulai komunikasi HTTP menuju URL spreadsheet yang telah didefinisikan dalam variabel sheetURL. Header ditambahkan dengan http.addHeader("Content-Type", "application/json") untuk memberi tahu bahwa data yang dikirimkan berupa format JSON. Selanjutnya, waktu saat ini diambil dari modul RTC menggunakan rtc.now() dan disimpan dalam variabel now. Nilai jam, menit, dan detik diformat menjadi string waktu menggunakan snprintf, begitu pula dengan tanggal dalam format hari/bulan/tahun. Setelah itu, fungsi switch digunakan untuk menentukan nama hari berdasarkan nilai now.dayOfTheWeek() yang mengembalikan angka 0–6. Masing-masing nilai dikonversi menjadi nama hari dalam bahasa Indonesia, seperti "Minggu", "Senin", dan seterusnya. Semua data ini nantinya akan digunakan untuk membuat payload JSON yang dikirim ke Google Apps Script agar dapat disimpan otomatis di dalam Google Spreadsheet untuk keperluan *monitoring* sistem *Greenhouse*.