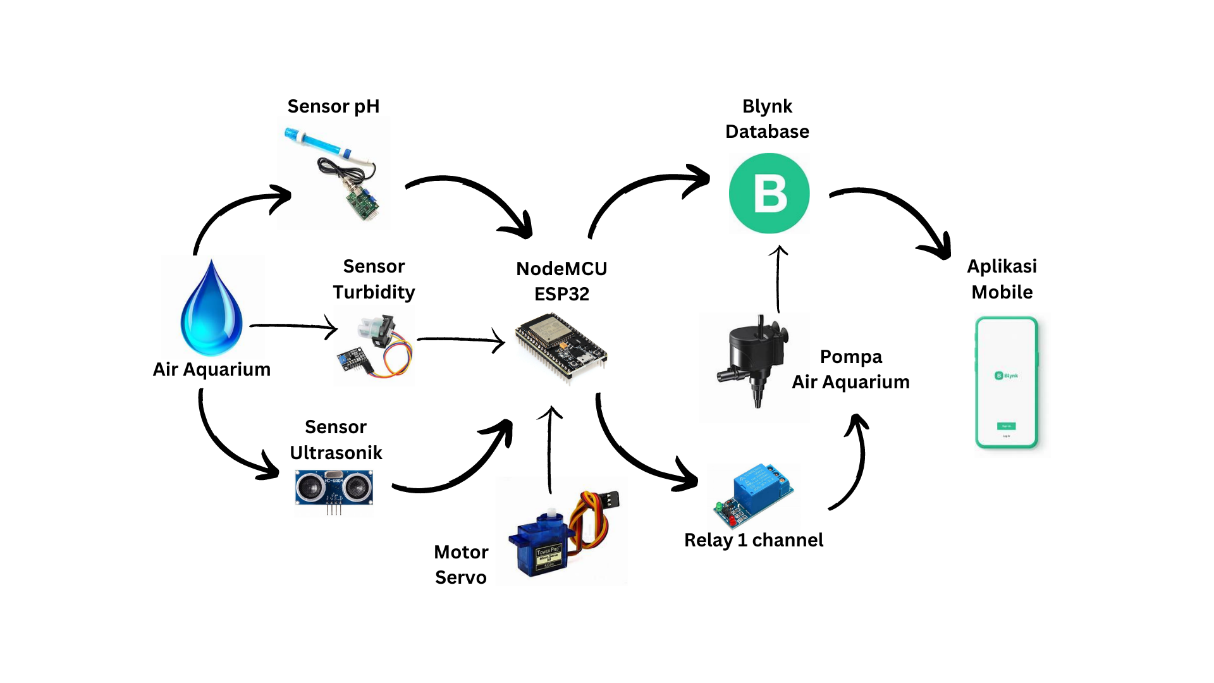
**3. PERANCANGAN DAN IMPLEMENTASI**

**3.1. Gambaran Umum Kerja Sistem**

Sistem aquarium ini merupakan alat yang dirancang untuk memudahkan pemeliharaan ikan hias khususnya ikan mas koki agar dapat dipantau secara otomatis. Dalam pengimplementasian alat ini, digunakan 2 sensor untuk memantau kualitas air yang ideal untuk ikan mas koki yaitu, sensor pH dan sensor Turbidity, kedua sensor ini digunakan sebagai trigger untuk mengatifkan pompa dan memfilter air agar kembali jernih. Selain memantau kualitas air terdapat pula sensor ultrasonik untuk memantau ketinggian air pada aquarium serta motor servo untuk memberikan makanan ikan secara otomatis, yang dapat dikendalikan dengan penjadwalan melalui aplikasi Blynk



***Gambar* 3.1 Gambaran Umum Sistem Aquarium Pintar Bberbasis Blynk**

**3.2. Perancangan Alat**

Dalam pembuatan alat ini berdasarkan analisa dan tahapan perencanaan yang telah dilakukan, diperlukan pembuatan perancangan dari segi perangkat keras maupun perangkat lunak

1. **Perangkat keras**

Perancangan perangkat keras yang dibutuhkan pada Aquarium Pintar berbasis IoT menggunakan Blynk

* NodeMCU ESP32 : digunakan sebagai pemrosesan data pada sistem yang digunakan
* Sensor pH 4502C : digunakan untuk mendeteksi nilai pH pada air
* Sensor Turbidity SEN0189 : digunakan sebagai pendeteksi tingkat kekeruhan air pada aquarium
* Sensor Ultrasonik HC-SR04 : digunakan untuk mendeteksi ketinggian air pada aquarium
* Motor Servo : berfungsi untuk memberikan makanan ikan secara otomatis maupun manual
* Modul Step-up MT3608 : berfungsi untuk menaikan tegangan output dan memberikan tegangan yang sesuai kebutuhan komponen
* Modul Step-down LM259 : berfungsi untuk menurunkan tegangan output dan memberikan tegangan sesuai dengan kebutuhan komponen
* Fan : digunakan untuk mendinginkan komponen perangkat keras yang digunakan
* Relay : berfungsi sebagai saklar dan menjalankan logika yang diberikan untuk menjalankan pompa dan mengganti cadangan daya dari adapter ke powerbank
* Switch : sebagai inputan ke relay untuk mengganti cadangan daya
* Baterai 3.25 V : berfungsi sebagai daya untuk menjalankan relay yang mengatur cadangan daya
* Pompa aquarium : digunakan untuk menyaring air ketika sudah keruh atau pH naik

1. **Perangkat Lunak**

Perancangan perangkat lunak yang dibutuhkan pada Aquarium Pintar berbasis IoT menggunakan Blynk

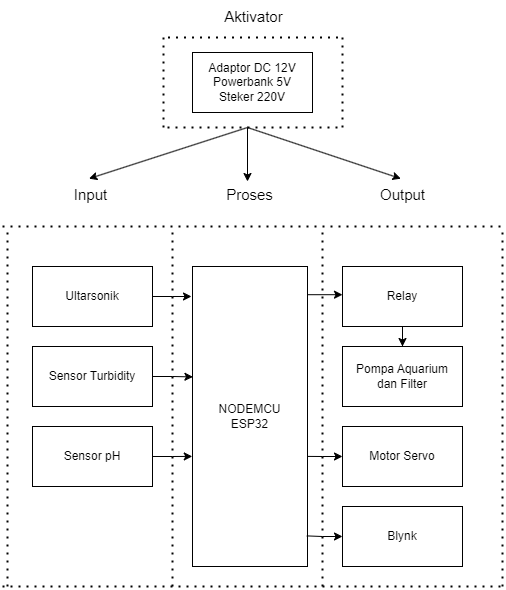
* Arduino IDE

Arduino IDE digunakan untuk membuat, mengubah dan menghapus code program yang ingin digunakan untuk mengkontrol suatu projek berbasis mikrokontroler. Arduino IDE memiliki banyak library yang dapat digunakan untuk komponen yang ingin digunakan pada suatu proyek, serta memiliki fitur untuk mengunggah kode melalui koneksi USB. Selain itu, Arduino IDE memiliki Serial Monitor untuk debugging.

* Blynk

Blynk adalah platform IoT (Internet of Things) yang memungkinkan pengguna untuk mengontrol dan memantau perangkat keras melalui aplikasi mobile. Dengan Blynk, pengguna dapat membuat antarmuka pengguna grafis (GUI) yang interaktif untuk perangkat IoT tanpa harus menulis banyak kode.

**3.2.1. Perancangan Perangkat Keras**

****

***Gambar* 3.2 Blok Diagram**

Rancangan sistem Aquarium Pintar dapat dilihat melalui blok diagram pada ***Gambar 3.2.*** Blok diagram diperlukan dalam perancangan sistem karena memberikan gambaran visual yang jelas dan sederhana mengenai komponen-komponen utama dan hubungan antar komponen dalam sistem. Berikut adalah penjelasan masing-masing blok yang digunakan :

**3.2.1.1. Blok Aktivator**

Pada blok aktivator terdapat Adaptor DC 12V sebagai sumber daya utama untuk komponen yang digunakan selain pompa aquarium. Modul Step-Down LM259 digunakan untuk menurunkan tegangan masukan adaptor dari 12V menjadi 5V sesuai dengan kebutuhann tegangan masukan ESP32. Powerbank dengan keluaran 5V digunakan sebagai sumber daya cadangan apabila terjadi pemadaman listrik. Dan steker 220V merupakan daya untuk menjalankan pompa aquarium. Modul Step-Up MT3608 pertama digunakan untuk menaikan voltase dari relay pertama, karena keluaran dari relay pertama kurang dari 5V, sedangkan ESP32 membutuhkan daya masukan sebesar 5V agar semua komponen bisa berfungsi sebagaimana mestinya.

**3.2.1.2. Blok Input**

Pada blok input terdapat sensor ultrasonik sebagai masukan ketinggian air pada aquarium. Perhitungan tinggi air pada aquarium dapat dihitung dengan cara *Tinggi air = tinggi wadah – jarak pengukuran sensor.* Pin *trigger* memancarkan gelombang ultrasonik, kemudian gelombang tersebut terpantul kembali ke sensor dan di tangkap oleh pin *echo.* Sensor turbidity digunakan untuk mendeteksi tingkat kekeruhan yang ada pada aquarium, dan sensor pH digunakan untuk mendeteksi tingkat keasaman air / pH yang terkandung dalam air tersebut. Ketiga sensor tersebut akan membaca keadaan air aquarium dan mengirimkan data ke mikrokontroller ESP32 untuk diproses dan menghasilkan output pada aplikasi blynk yang sudah terintegrasi secara realtime.

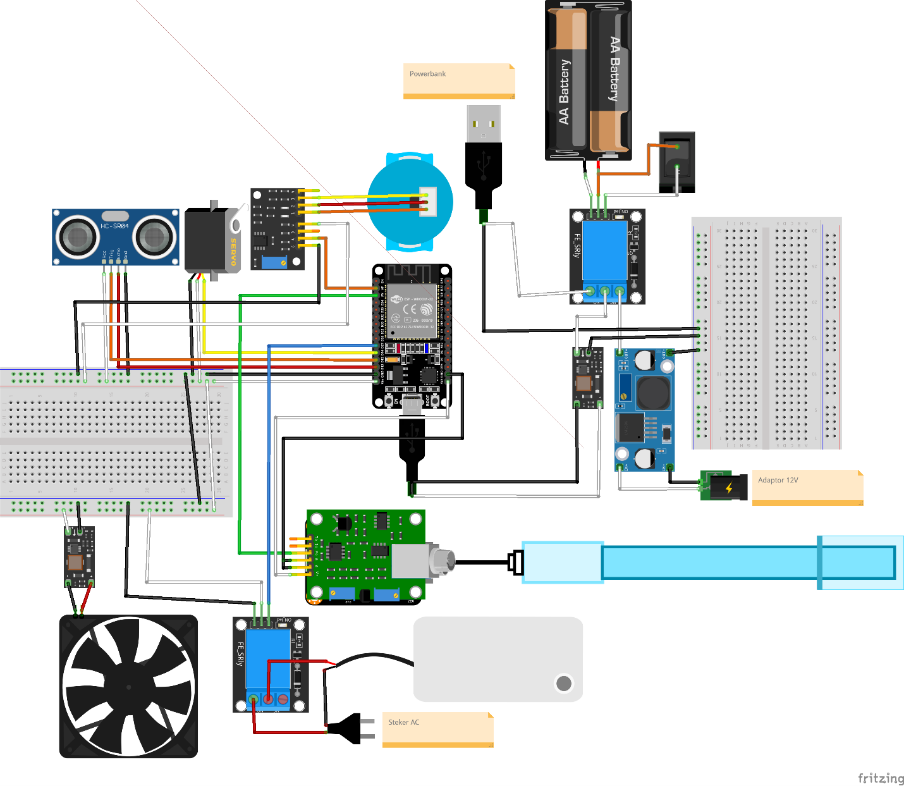
**3.2.1.3. Blok Proses**

Pemrosesan yang dilakukan pada alat ini adalah mikrokontroller NodeMCU ESP32. Pada blok ini semua data yang didapatkan dari setiap sensor akan di eksekusi sesuai code perintah yang sudah di upload pada mikrokontroller ini. NodeMCU ESP32 sudah terintegrasi secara realtime dengan aplikasi blynk, sehingga setelah data diproses maka akan diteruskan ke blok output.

**3.2.1.4. Blok Output**

Sedangkan pada blok output terdapat relay untuk mengkontrol pompa aktif atau mati, pompa akan aktif jika memenuhi kondisi yang sudah ditentukan pada mikrokontroller tersebut. Selain pompa, motor servo memberikan output berupa makanan ikan sesuai jadwal yang diatur pada aplikasi blynk dan fan sebagai keluaran untuk mendinginkan komponen yang digunakan pada sistem ini. Semua data

Berdasarkan blok diagram pada gambar 3, rangkaian skematik sistem aquarim pintar ini dapat diuraikan sebagai berikut :



**Gambar 3.3 Skematik Rangkaian**

Pada **Gambar 3.3** terdapat rangkaian skematik dari sistem aquarium pintar ini, untuk lebih memahami lebih detail terkait komponen apa saja yang digunakan dan bagaimana setaip komponen saling terhubung, dapat dilihat pada tabel berikut :

**Tabel 3.1 Komponen dan Pin**

|  |  |  |  |
| --- | --- | --- | --- |
| KOMPONEN | PIN KOMPONEN | TERHUBUNG KE | |
| PIN | KOMPONEN |
| NodeMCU ESP32 | + 5v | Positif | Breadboard |
| Gnd | Negatif | Breadboard |
| D12 | Trig | Ultrasonik |
| D13 | Echo | Ultrasonik |
| D14 | Sinyal | Motor Servo |
| D27 | Common | Relay 2 |
| Vn | Po | Modul pH |
| Vp | Analog | Modul Turbidity |
| Motor Servo | Sinyal | D14 | NodeMCU ESP32 |
| Vcc | Posiitif | Breadboard |
| Gnd | Negatif | Breadboard |
| Ultrasonik HC-SR04 | Vcc | Positif | Breadboard |
| Echo | D13 | NodeMCU ESP32 |
| Trig | D12 | NodeMCU ESP32 |
| Gnd | Negatif | Breadboard |
| Modul Sensor pH 4502C | Vcc | 3.3v | NodeMCU ESP32 |
| Gnd | Gnd | NodeMCU ESP32 |
| Po | Vn | NodeMCU ESP32 |
| To | - | - |
| Do | - | - |
| Modul Sensor Turbidity SEN0189 | Gnd | Negatif | Breadboard |
| Analog | Vp | NodeMCU ESP32 |
| Digital | None | None |
| Vcc | Positif | Breadboard |
| 1 | Vcc | Sensor Turbidity |
| 2 | Sinyal | Sensor Turbidity |
| 3 | Gnd | Sensor Turbidity |
| 4 | - | - |
| Sensor Turbidity | Vcc | 1 | Modul sensor Turbidity SEN0189 |
| Sinyal | 2 | Modul sensor Turbidity SEN0189 |
| Gnd | 3 | Modul sensor Turbidity SEN0189 |
| Relay 1 | In | In | Switch |
| Gnd | Negatif | Breadboard |
| Vcc | Vcc | Switch |
| No | Powerbank | Modul Step-up 1 |
| C | Vin + | Modul Step-up 1 |
| Nc | Vout + | Modul Step-down |
| Switch | Vcc | Vcc | Relay 1 |
| In | In | Relay 1 |
| Baterai | Positif (+) | * Vcc * Vcc | * Switch * Relay 1 |
| Negatif (-) | Gnd | Relay 1 |
| Modul Step-up 1 | Vin + | C | Relay 1 |
| Vin - | Negatif | Breadboard |
| Vout + | Kabel USB | NodeMCU ESP32 |
| Vout - | Kabel USB | NodeMCU ESP32 |
| Modul Step-down | Vin + | Positif adapter | Relay 1 |
| Vin - | Negatif adapter | Adapter |
| Vout + | Nc | Relay 1 |
| Vout - | Negatif | Breadboard |
| Relay 2 | In | D27 | NodeMCU ESP32 |
| Vcc | Positif | Breadboard |
| Gnd | Negatif | Breadboard |
| Nc | - | - |
| C | Positif Pompa | Pompa |
| No | Positif Steker | Steker |
| Modul Step-up 2 | Vin + | Positif | Breadboard |
|  | Vin - | Negatif | Breadboard |
|  | Vout + | Negatif Fan | Fan |
|  | Vout - | Positif Fan | Fan |

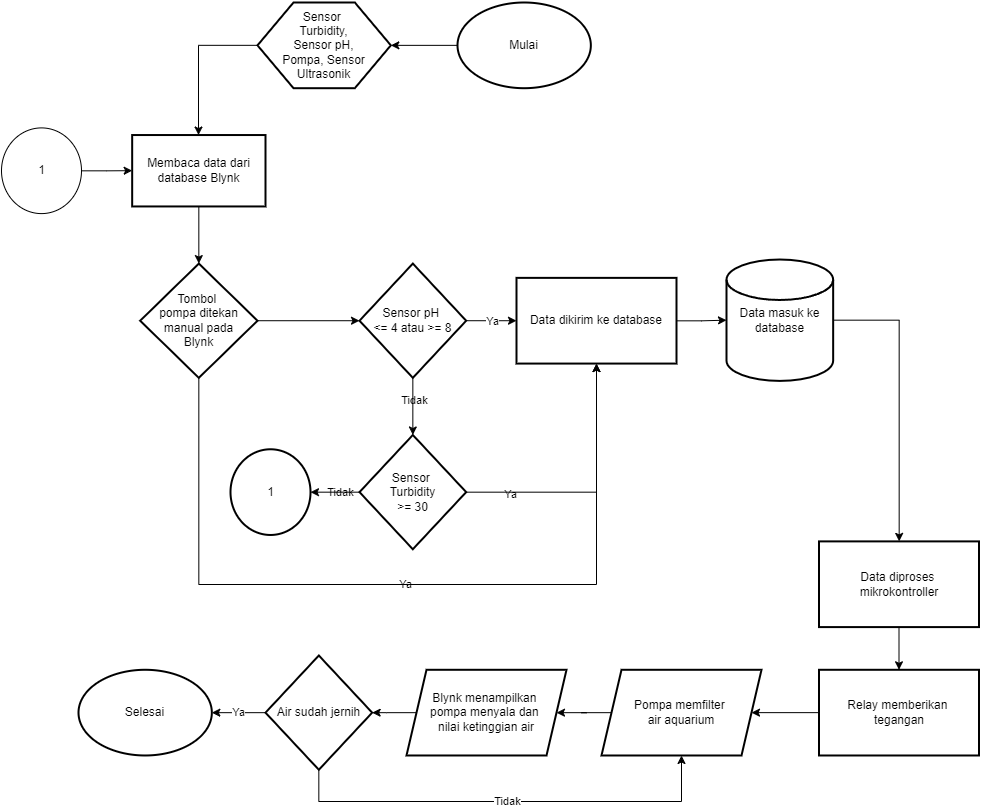
**3.2.2. Perancangan Perangkat Lunak**

Dalam pembuatan sistem sesuai skematik diatas, code program diperlukan untuk mengendalikan setiap komponennya. Bahasa pemrograman yang digunakan yaitu bahasa C dan akan dijelaskan alur program untuk menghasilkan ouput dalam flowchart. Berikut adalah flowchart dari komponen yang digunakan selain motor servo, karena motor servo tidak berkaitan dengan kondisi air pada aquarium, sehingga memiliki flowchart sendiri.



**Gambar 3.4 Tampilan aplikasi Blynk**

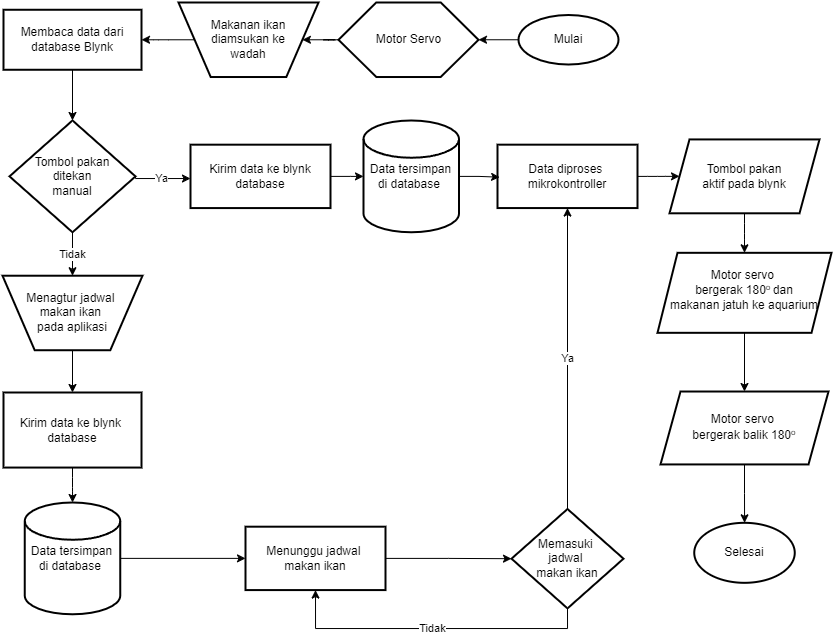
**3.2.2.1 Flowchart**

****

**Gambar 3.5 Flowchart Sensor pH, Sensor Turbidity dan**

**Sensor Ultrasonik**

Berdasarkan *gambar 3.5* dijelaskan bagaimana alur kerja sistem aquarium pintar ini. Dimulai dengan inisialisasi komponen yang digunakan (Sensor pH, sensor turbidity, pompa air dan sensor ultrasonik), pengaktifan mikrokontroller yang digunakan, pendefinisian pin, pendeklarasian variabel yang digunakan dan juga menghubungakan perangkat dengan blynk serta internet. Setelah proses inisialisasi selesai, mikrokontroller akan membaca data awal dari blynk, yang mana data awal itu relay tidak memiliki tegangan sehingga pompa air belum aktif. Kemudian pengguna bisa menekan tombol pompa manual atau tidak pada tampilan aplikasi blynk. Jika pengguna menekan tombol manual *pompa* yang dapat dilihat pada *gambar 3.4*, maka data akan dikirimkan ke blynk database mikrokontroller ESP32 akan memproses data keluaran dari database, sehingga relay diberikan tegangan yang memicu pompa agar menyala. Kemudian jika air tidak keruh maka pompa akan otomatis untuk mati dan program selesai. Namun jika pengguna tidak menekan tombol manual pada aplikasi blynk, sensor sebagai pemicu relay untuk menyalakan pompa. *Kondisi pertama* jika sensor pH bernilai ≤ 4 atau ≥ 8 maka relay diberikan tegangan dan pompa air akan menyala. Jika sensor pH mendeteksi nilai tersebut maka data akan dikirmkan ke database blynk, kemudian akan diproses oleh mikrokontroller ESP32 untuk memberikan tegangan pada relay dan mengaktifkan pompa, pompa akan berhenti jika nilai pH sudah tidak memenuhi kondisi tersebut. *Kondisi kedua* jika sensor turbidity (kekeruhan) bernilai ≥ 30, maka prosesnya akan sama seperti kondisi sensor pH tadi. Jika kedua kondisi tidak terpenuhi maka akan berulang pada proses sebelum pengguna menekan tombol manual pompa pada blynk.



**Gambar 3.6 Flowchart Motor Servo**

Berdasarkan *gambar 3.6* dijelaskan bagaimana alur kerja makanan ikan dari awal sampai akhir. Dimulai dengan inisialisasi komponen yang digunakan (motor servo), pengaktifan mikrokontroller yang digunakan, pendefinisian pin, pendeklarasian variabel yang digunakan dan juga menghubungakan perangkat dengan blynk serta internet. Setelah inisialisasi pengguna memasukan terlebih dahulu makanan ikan pada wadah yang digunakan untuk dikontrol melalui blynk. Selanjutnya mikrokontroller membaca data awal dari database blynk, yang dimana data awal motor servo berlogika 0 (LOW) sehingga motor servo belum bergerak. Kemudian pengguna bisa menekan tombol *pakan* (*gambar 3.4*) secara manual atau melakukan penjadwalan pada aplikasi blynk. Jika pengguna menekan tombol pakan, data akan dikirmkan ke blynk database kemudian diproses oleh mikrokontroller ESP32 dan memberikan logika 1 (HIGH) pada motor servo, sehingga motor servo akan bergerak 180° dan makanan ikan akan jatuh ke aquarium, selanjutnya motor servo akan kembali ke posisi 0° (posisi awal). Namun jika pengguna tidak menekan tombol pakan pada aplikasi blynk, maka pengguna bisa mengatur jadwal kapan makanan ikan tersebut akan diberikan ke ikan. Setelah pengguna mengatur jadwal yang diinginkan untuk memberikan makanan ikan, maka data akan dikirimkan ke blynk database dan proses seterusnya sama dengan proses ketika setelah menekan tombol *pakan* secara manual dari aplikasi blynk.

**3.2.2.2 Analisa Program**

Berdasarkan flowchart pada *gambar 3.5* dan *gambar 3.6* berikut adalah penjelasan setiap baris code untuk menjalankan sistem *AquaSmart Berbasis Blynk IoT* agar sesuai dengan kedua gambar tersebut.



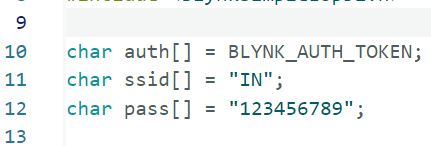
**Gambar 3.7 Menghubungkan Blynk dan Library**

**Yang Dibutuhkan**

Berikut adalah penjelasan dari code / instruksi yang diberikan pada *gambar 3.7 :*

**Tabel 3.2 Menghubungkan Blynk dan Library Yang Dibutuhkan**

|  |  |
| --- | --- |
| **Program** | **Keterangan** |
| #define BLYNK\_TEMPLATE\_ID "TMPL6n5YuDce4"  #define BLYNK\_TEMPLATE\_NAME "AquaSmart"  #define BLYNK\_AUTH\_TOKEN "s75chNlwx1Cx7yjJDr2RLzf77005UFCT" | Baris code ini digunakan untuk mengonfigurasi dan menghubungkan ESP32 ke platform Blynk. BLYNK\_TEMPLATE\_ID dan BLYNK\_TEMPLATE\_NAME mengidentifikasi template Blynk, sedangkan BLYNK\_AUTH\_TOKEN adalah token autentikasi yang unik. |
| #define BLYNK\_PRINT Serial | Code mengarahkan keluaran debug Blynk ke serial monitor untuk pemantauan. |
| #include <ESP32Servo.h>  #include <WiFi.h>  #include <BlynkSimpleEsp32.h> | #include <ESP32Servo.h>: Library untuk mengontrol servo dengan ESP32.  #include <WiFi.h>: Library untuk menghubungkan ESP32 ke jaringan WiFi.  #include <BlynkSimpleEsp32.h>: Library untuk menghubungkan ESP32 ke platform Blynk. |



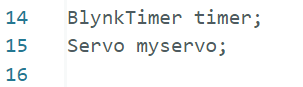
**Gambar 3.8 Variabel Yang Digunakan**

**Untuk Terhubung di Internet**

Berikut adalah penjelasan dari code / instruksi yang diberikan pada *gambar 3.8 :*

**Tabel 3.3 Variabel Yang Digunakan Untuk Terhubung di Internet**

|  |  |
| --- | --- |
| **Komponen** | **Keterangan** |
| char auth[] = BLYNK\_AUTH\_TOKEN;  char ssid[] = "IN";  char pass[] = "123456789"; | Baris code ini menjelaskan tentang bagaiamana perangakat terhubung dengan blynk dengan memasukan code token yang disediakan dari blynk dan menggunakan ssid serta password jaringan internet untuk menghubungkan blynk dengan perangkat |



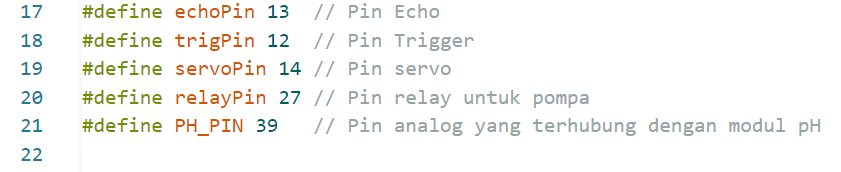
**Gambar 3.9 Pendeklarasian Variabel**

**Untuk Timer dan Motor Servo**

Berikut adalah penjelasan dari code / instruksi yang diberikan pada *gambar 3.9 :*

**Tabel 3.4 Pendeklarasian Variabel Untuk Timer dan Motor Servo**

|  |  |
| --- | --- |
| **Komponen** | **Keterangan** |
| BlynkTimer timer;  Servo myservo; | BlynkTimer dan Servo merupakan objeck yang digunakan untuk menkontrol : BlynkTimer timer, Objek timer untuk mengatur interval tugas (misalnya mengirim data sensor secara periodik).  Servo myservo, untuk menkontrol servo |



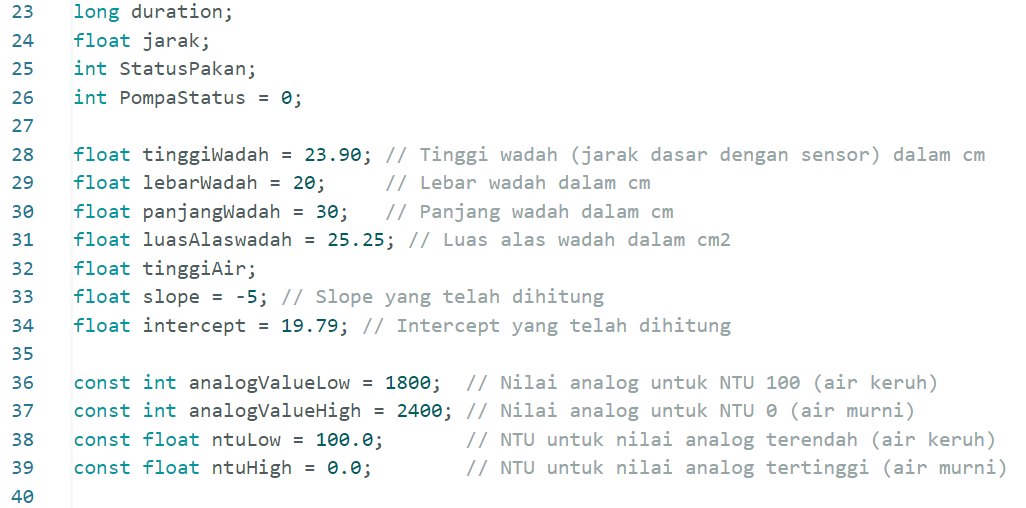
**Gambar 3.10 Pendefinisian Setiap Pin**

**Yang digunakan pada ESP32**

Berikut adalah penjelasan dari code / instruksi yang diberikan pada *gambar 3.10 :*

**Tabel 3.5 Pendefinisian Setiap Pin Yang digunakan pada ESP32**

|  |  |
| --- | --- |
| **Komponen** | **Keteranagn** |
| #define echoPin 13  // Pin Echo  #define trigPin 12  // Pin Trigger  #define servoPin 14 // Pin servo  #define relayPin 27 // Pin relay untuk pompa  #define PH\_PIN 39   // Pin analog yang terhubung dengan modul pH | Mendefinisikan pin pada komponen yang terhhubung dengan ESP32  Pin Echo Ultrasonik terhubung dengan pin D13 ESP32  Pin Trig Ultrasonik terhubung dengan pin D12 ESP32  Pin Sinyal pada servo terhubung dengan pin D14 ESP32  Pin IN pada relay terhubung dengan pin D27 ESP32  Pin Po pada modul pH terhubung dengan pin Vp (GPIO 39) pada ESP32 |



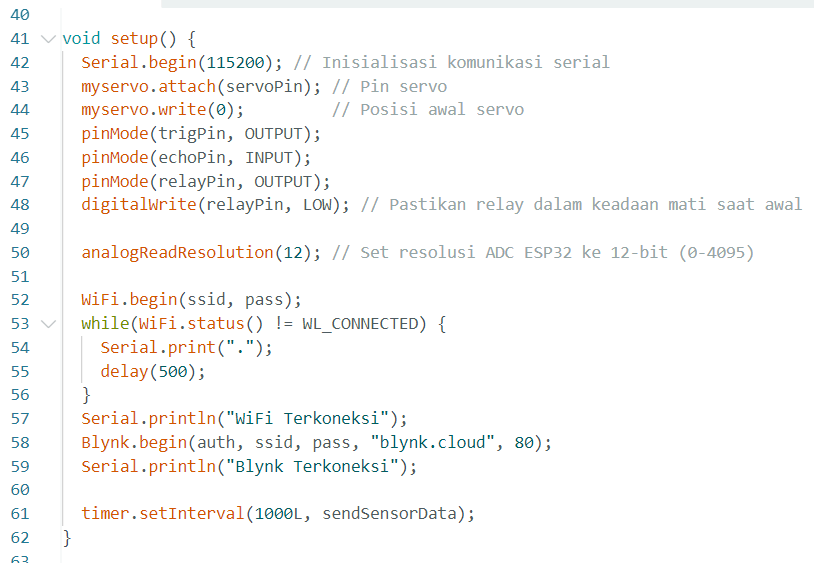
**Gambar 3.11 Mendeklarasikan Variabel pada**

**Setiap Objek yang Menyimpan Nilai Pengukuran**

Berikut adalah penjelasan dari code / instruksi yang diberikan pada *gambar 3.11 :*

**Tabel 3.6 Mendeklarasikan Variabel pada Setiap Objek yang Menyimpan Nilai Pengukuran**

|  |  |
| --- | --- |
| **Komponen** | **Keterangan** |
| long duration;  float jarak;  int StatusPakan;  int PompaStatus = 0; | long duration, float jarak merupkan variabel untuk menyimpan durasi sinyal ultrasonik dan jarak yang diukur.  Int StatusPakan menunjukan nilai variabel dari StatusPakan  int PompaStatus = 0 menunjukan nilai awal dari pompa itu 0 (aktif high) |
| float tinggiWadah = 23.90; // Tinggi wadah (jarak dasar dengan sensor) dalam cm  float lebarWadah = 20;     // Lebar wadah dalam cm  float panjangWadah = 30;   // Panjang wadah dalam cm  float luasAlaswadah = 25.25; // Luas alas wadah dalam cm2  float tinggiAir; | float tinggiWadah, float lebarWadah, float panjangWadah, float luasAlaswadah, float tinggiAir merupakan variabel untuk menyimpan parameter fisik wadah air. |
| float slope = -5; // Slope yang telah dihitung  float intercept = 19.79; // Intercept yang telah dihitung | float slope, float intercept merupakan parameter kalibrasi untuk sensor pH. |
| const int analogValueLow = 1800;  // Nilai analog untuk NTU 100 (air keruh)  const int analogValueHigh = 2400; // Nilai analog untuk NTU 0 (air murni)  const float ntuLow = 100.0;       // NTU untuk nilai analog terendah (air keruh)  const float ntuHigh = 0.0;        // NTU untuk nilai analog tertinggi (air murni) | const int analogValueLow, const int analogValueHigh, const float ntuLow, const float ntuHigh merupakan parameter kalibrasi untuk sensor kekeruhan. |

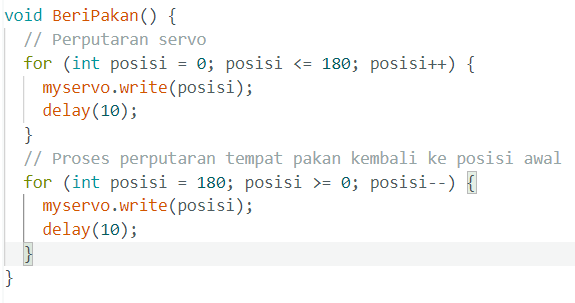


**Gambar 3.12 Perintah pada Void setup**

Berikut adalah penjelasan dari code / instruksi yang diberikan pada *gambar 3.12 :*

**Tabel 3.7 Perintah pada Void setup**

|  |  |
| --- | --- |
| **Code** | **Keterangan** |
| void setup() | void setup(), merupakan fungsi yang dijalankan sekali saat mikrokontroler dinyalakan atau di-reset. |
| Serial.begin(115200); | Serial.begin(115200), menginisialisasi komunikasi serial dengan baud rate 115200. |
| myservo.attach(servoPin);  myservo.write(0); | myservo.attach(servoPin), myservo.write(0), baris code ini menginisialisasi servo dan mengatur posisi awalnya ke 0 derajat. |
| pinMode(trigPin, OUTPUT);  pinMode(echoPin, INPUT);  pinMode(relayPin, OUTPUT); | pinMode, merupakan code untuk mengatur mode pin sebagai input atau output. |
| digitalWrite(relayPin, LOW); | digitalWrite(relayPin, LOW), code ini memastikan relay dalam keadaan mati saat awal (Low). |
| analogReadResolution(12); // Set resolusi ADC ESP32 ke 12-bit (0-4095) | analogReadResolution(12), mengatur resolusi ADC ESP32 ke 12-bit (0-4095). |
| WiFi.begin(ssid, pass); | WiFi.begin(ssid, pass), code untuk memulai koneksi WiFi dengan SSID dan password yang diberikan. |
| while(WiFi.status() != WL\_CONNECTED) {      Serial.print(".");      delay(500);    } | while(WiFi.status() != WL\_CONNECTED), code yang menunjukan proses ESP32 menunggu terhubung ke WiFi. |
| Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);    Serial.println("Blynk Terkoneksi"); | Blynk.begin(auth, ssid, pass, "blynk.cloud", 80): code untuk menghubungkan ESP32 ke Blynk. |
| timer.setInterval(1000L, sendSensorData); | timer.setInterval(1000L, sendSensorData), mengatur interval tugas untuk mengirim data sensor setiap 1 detik. |



**Gambar 3.13 Perintah pada Void BeriPakan untuk Mengkontrol**

**Pergerakan Motor Servo**

Berikut adalah penjelasan dari code / instruksi yang diberikan pada *gambar 3.13 :*

**Tabel 3.8 Perintah pada Void BeriPakan untuk Mengkontrol Pergerakan Motor Servo**

|  |  |
| --- | --- |
| **Code** | **Keterangan** |
| void BeriPakan() {    // Perputaran servo    for (int posisi = 0; posisi <= 180; posisi++) {      myservo.write(posisi);      delay(10);    }    // Proses perputaran tempat pakan kembali ke posisi awal    for (int posisi = 180; posisi >= 0; posisi--) {      myservo.write(posisi);      delay(10);    }  } | void BeriPakan(), code ini berfungsi untuk mengontrol pergerakan servo untuk memberikan pakan. Perulangan *for* digunakan untuk mencontrol perputaran servo dari posisi 0 derajat sampai posisi 180 derajat, kemudian kembali lagi dari 180 derajat ke 0 derajat |

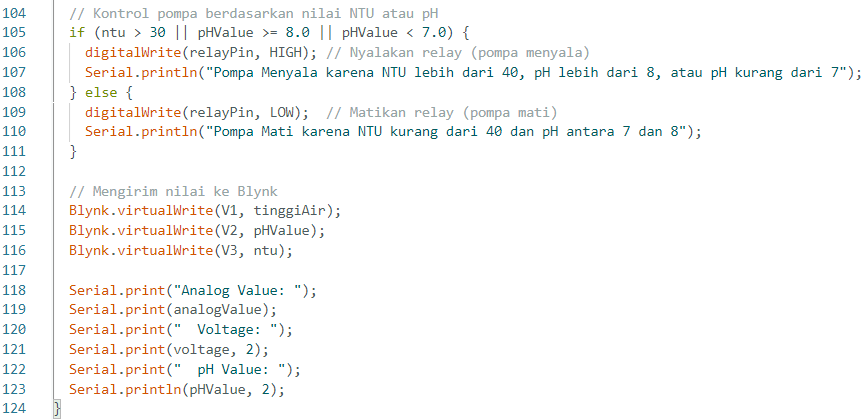


**Gambar 3.14 Perintah pada Void sendSensorData untuk Mengirim Data Semua Sensor Yang Digunakan**

Berikut adalah penjelasan dari code / instruksi yang diberikan pada *gambar 3.14 :*

**Tabel 3.9 Perintah pada Void sendSensorData untuk Mengirim Data Semua Sensor Yang Digunakan**

|  |  |
| --- | --- |
| **Code** | **Keterangan** |
| void sendSensorData() | Merupakan fungsi yang dijalankan secara periodik untuk mengukur dan mengirim data sensor ke Blynk. |
| // Mengukur jarak dengan sensor ultrasonik    digitalWrite(trigPin, LOW);    delayMicroseconds(2);    digitalWrite(trigPin, HIGH);    delayMicroseconds(10);    digitalWrite(trigPin, LOW);    duration = pulseIn(echoPin, HIGH);    jarak = duration \* 0.034 / 2; // Konversi ke jarak sebenarnya (cm) | Baris code ini menunjukan pengukuran nilai ketinggian air dengan menggunakan sensor ultrasonik, kemudian nilai yang dihasilkan dari pengukuran ultrasonik akan dikonversi dengan rumus yang sudah di definisikan pada variabel *jarak.* |
| // Menghitung tinggi air    tinggiAir = tinggiWadah - jarak; | Code ini digunakan setelah nilai konversi sensor ultrasonik diketahui, kemudian akan diukur ketinggian air pada aquariumnya |
| Blynk.run(); | Code untuk menjalankan Blynk |
| int sensorValue = analogRead(34);  // Membaca nilai analog dari pin 34 pada ESP32    // Kalibrasi nilai sensor ke NTU    float ntu = map(sensorValue, analogValueLow, analogValueHigh, ntuLow \* 100, ntuHigh \* 100) / 100.0;    Serial.print("TURBIDITY: ");    Serial.print(ntu);    Serial.println(" NTU"); | int sensorValue = analogRead(34), merupakan variabel untuk menghubungkan sensor Turbidity dengan ESP32 pada GPIO 34.  Fungsi *map* digunakan untuk menghitung nilai kekeruhan pada air aquarium |
| // Membaca nilai pH    int analogValue = analogRead(PH\_PIN); // Membaca nilai analog dari pin pH    float voltage = analogValue \* (3.3 / 4095.0); // Konversi nilai ADC ke tegangan    float pHValue = slope \* voltage + intercept; // Menghitung nilai pH berdasarkan tegangan | Code untuk membaca nilai pH air aquarium |



**Gambar 3.15 Kondisi Untuk Menyalakan Pompa Aquarium dan Mengirim**

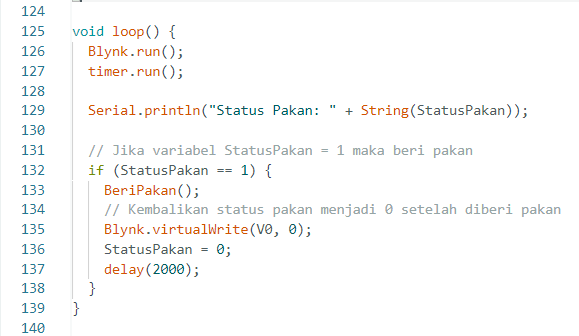
**Nilai ke Blynk**

Berikut adalah penjelasan dari code / instruksi yang diberikan pada *gambar 3.15 :*

**Tabel 3.10 Kondisi Untuk Menyalakan Pompa Aquarium dan Mengirim**

**Nilai ke Blynk**

|  |  |
| --- | --- |
| **Code** | **Keterangan** |
| // Kontrol pompa berdasarkan nilai NTU atau pH    if (ntu > 30 || pHValue >= 8.0 || pHValue < 7.0) {      digitalWrite(relayPin, HIGH); // Nyalakan relay (pompa menyala)      Serial.println("Pompa Menyala karena NTU lebih dari 40, pH lebih dari 8, atau pH kurang dari 7");    } else {      digitalWrite(relayPin, LOW);  // Matikan relay (pompa mati)      Serial.println("Pompa Mati karena NTU kurang dari 40 dan pH antara 7 dan 8");    } | Mengkontrol pompa sesuai dengan kondisi yang diinginkan |
| // Mengirim nilai ke Blynk    Blynk.virtualWrite(V1, tinggiAir);    Blynk.virtualWrite(V2, pHValue);    Blynk.virtualWrite(V3, ntu); | Mengirim data ke blynk melalui pin virtual yang disediakan oleh blynk (V1, V2 dan V3). |
| Serial.print("Analog Value: ");    Serial.print(analogValue);    Serial.print("  Voltage: ");    Serial.print(voltage, 2);    Serial.print("  pH Value: ");    Serial.println(pHValue, 2); | Menampilkan nilai output komponen pada serial monitor |



**Gambar 3.16 Perintah Pada void loop untuk menjalankan Blynk dan untuk**

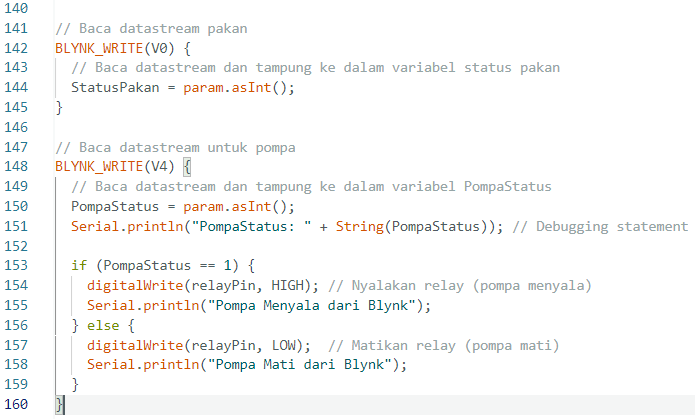
**Mengkontrol Motor Servo**

Berikut adalah penjelasan dari code / instruksi yang diberikan pada *gambar 3.16 :*

**Tabel 3.11 Perintah Pada void loop untuk menjalankan Blynk dan untuk**

**Mengkontrol Motor Servo**

|  |  |
| --- | --- |
| **Code** | **Keterangan** |
| void loop() {    Blynk.run();    timer.run();    Serial.println("Status Pakan: " + String(StatusPakan));      // Jika variabel StatusPakan = 1 maka beri pakan    if (StatusPakan == 1) {      BeriPakan();      // Kembalikan status pakan menjadi 0 setelah diberi pakan      Blynk.virtualWrite(V0, 0);      StatusPakan = 0;      delay(2000);    }  } | Memastikan Blynk dan timer terus berjalan. Jika StatusPakan adalah 1, maka fungsi BeriPakan dipanggil dan StatusPakan di-reset ke 0. |



**Gambar 3.17 Membaca datastream Pada Pin V0 (Motor Servo)**

**Dan Pin V4 (Relay)**

Berikut adalah penjelasan dari code / instruksi yang diberikan pada *gambar 3.17 :*

**Tabel 3.12 Membaca datastream Pada Pin V0 (Motor Servo)**

**Dan Pin V4 (Relay)**

|  |  |
| --- | --- |
| **Code** | **Keterangan** |
| // Baca datastream pakan  BLYNK\_WRITE(V0) {    // Baca datastream dan tampung ke dalam variabel status pakan    StatusPakan = param.asInt();  } | Membaca void StatusPakan pada pin virtual V0 |
| // Baca datastream untuk pompa  BLYNK\_WRITE(V4) {    // Baca datastream dan tampung ke dalam variabel PompaStatus    PompaStatus = param.asInt();    Serial.println("PompaStatus: " + String(PompaStatus)); // Debugging statement | Membaca PompaStatus ke blynk melalui pin virtual V4 |
| if (PompaStatus == 1) {      digitalWrite(relayPin, HIGH); // Nyalakan relay (pompa menyala)      Serial.println("Pompa Menyala dari Blynk");    } else {      digitalWrite(relayPin, LOW);  // Matikan relay (pompa mati)      Serial.println("Pompa Mati dari Blynk");    }  } | Kondisi untuk menjalankan pompa, jika relay berlogika 1 maka pompa akan menyala |

**3.3 Pengujian Alat**