BAB 4. IMPLEMENTASI

4.1 Implementasi Awal

Implementasi awal dari sistem dimulai dari penghitungan persamaan untuk kalibrasi sensor dan meng-*instal* aplikasi-aplikasi yang digunakan untuk membuat sistem. Aplikasi-aplikasi yang digunakan antara lain adalah Arduino *Integrated Development Environment* (Arduino IDE) dan *Blynk*. Dengan menggunakan Arduino IDE, sensor-sensor beserta perangkat elektronik yang terhubung pada WeMos akan dapat dikontrol. Untuk membuat *user interface* yang dapat mengontrol sensor-sensor melalui koneksi internet dibutuhkan aplikasi *Blynk*.

4.1.1 Penghitungan Persamaan untuk Kalibrasi Sensor

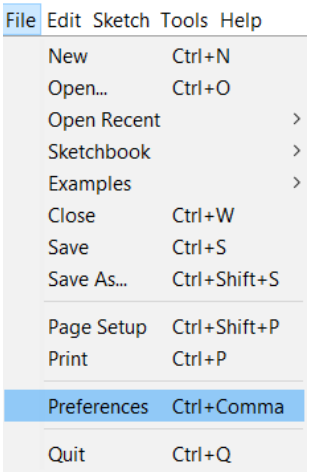
Kalibrasi sensor diperlukan untuk meningkatkan keakuratan hasil *output* sensor. Untuk melakukan kalibrasi sensor temperatur dan pH, diperlukan persamaan linear regresi yang akan dimasukkan dalam program kalibrasi sensor. Persamaan linear regresi dapat didapatkan dari data-data berupa (x,y) yang dimasukkan pada *link* http://www.alcula.com/calculators/statistics/linear-regression/ sehingga akan menghasilkan persamaan linear regresi. Data-data yang dimasukkan ke dalam *link* tersebut berisi data hasil sensor-sensor WeMos sebelum dikalibrasi dan data hasil sensor-sensor referensi (sensor lain yang sudah teruji keakuratannya). Untuk sensor temperatur mendapatkan grafik dengan persamaan y = 0.96102302970732x + 4.073482050974 sedangkan untuk sensor pH mendapatkan grafik dengan persamaan y = 2.1856210042767x-0.6410403120014.

4.1.2 Instalasi Arduino *Integrated Development Environment*

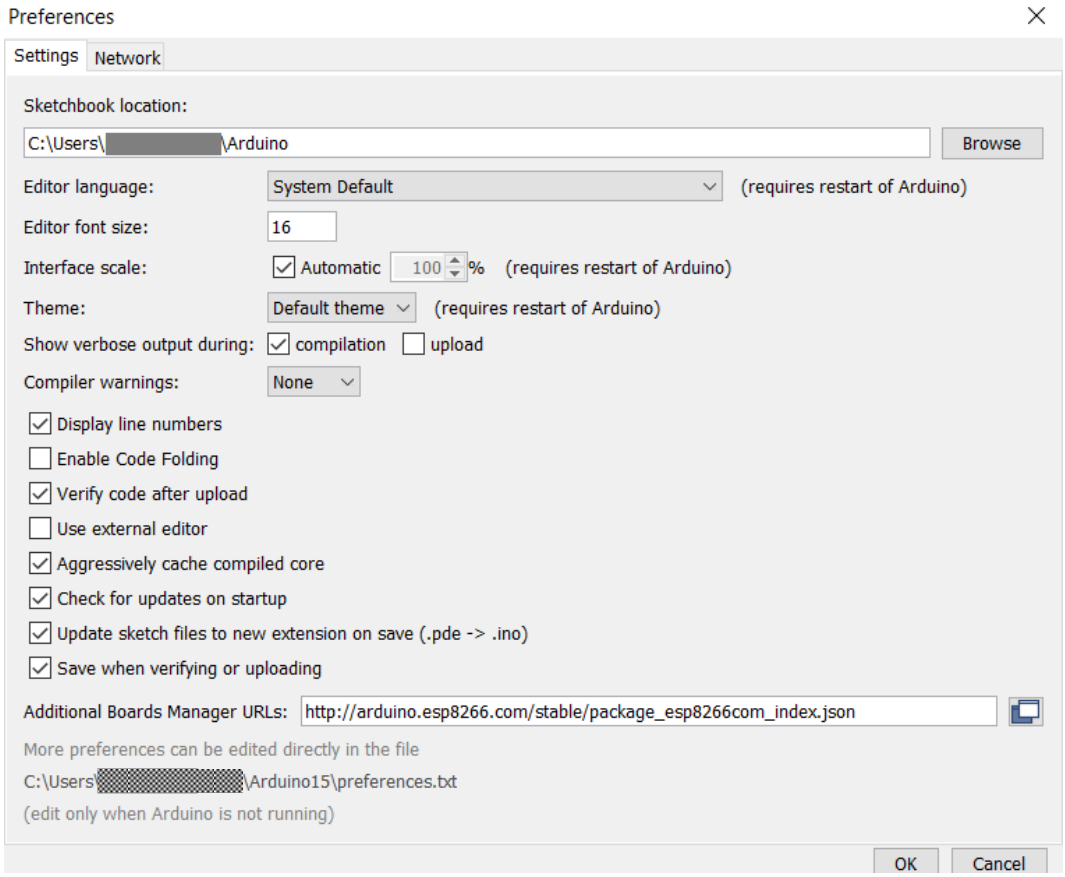
Arduino *Integrated Development Environment* (Arduino IDE) digunakan sebagai penghubung antara *microcontroller* WeMos dengan pemrograman yang dibuat. Dengan menggunakan Arduino, pemrograman yang dibuat dapat ditulis

pada memori *microcontroller*. Karena bersifat *open-source,* Arduino IDE dapat men-*support microcontroller* lain selain Arduino. *Microcontroller* yang di-*support* Arduino IDE beberapa di antaranya yaitu Arduino, WeMos, dan produk ESP 8266.

Untuk men-*download software* Arduino IDE, terdapat pada link https://www.arduino.cc/en/Main/Software. Arduino IDE dapat di-*instal* pada beberapa OS yaitu Windows, Mac OS, dan Linux. Setelah men-*download*, proses selanjutnya adalah meng-*instal* Arduino IDE. Setelah meng-*instal* Arduino IDE, konfigurasi *additional board manager* ESP8266 dibutuhkan untuk meng-*install* semua *library* yang dibutuhkan pada *board* ESP8266. Konfigurasi dilakukan dengan membuka Arduino IDE, kemudian menekan *File* → *Preference* → mengisi Arduino *Boards Manager* URLs dengan *url link* http://arduino.esp8266.com/stable/package_esp8266com_index.json. Untuk lebih jelasnya dapat dilihat seperti pada gambar 4.1 berikut.



Gambar 4.1 Cara Konfigurasi untuk Menambah *Board Manager* URL

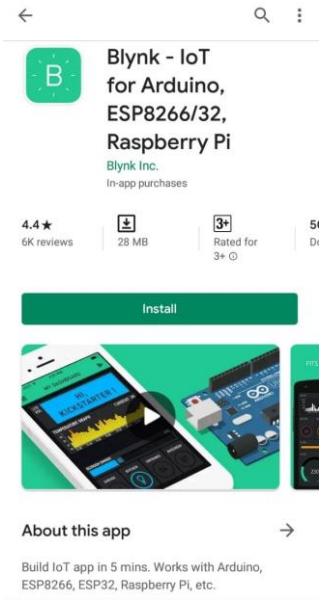


Gambar 4.1 Cara Konfigurasi untuk Menambah *Board Manager* URL (Lanjutan)

4.1.3 Instalasi dan Persiapan Aplikasi *Blynk*

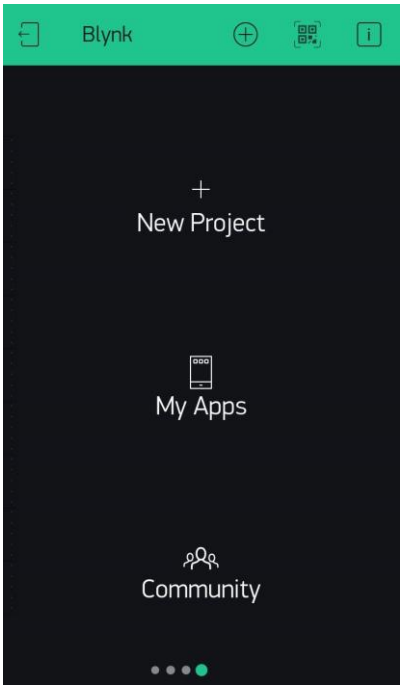
Aplikasi *Blynk* digunakan sebagai aplikasi *user interface* yang dapat memantau berdasarkan 4 parameter (*feeding*, temperatur air, ketinggian air, serta pH air) serta mengontrol kondisi akuarium kecuali kondisi pH air. Aplikasi *Blynk*

dapat terhubung oleh berbagai macam *microcontroller*, salah satunya yaitu *microcontroller* WeMos. Aplikasi *Blynk* dapat di-*install* pada *handphone* dengan *operating sistem* Android maupun iOS. Cara menginstal aplikasi Blynk dapat mengikuti petunjuk di bawah ini seperti gambar 4.2 berikut.



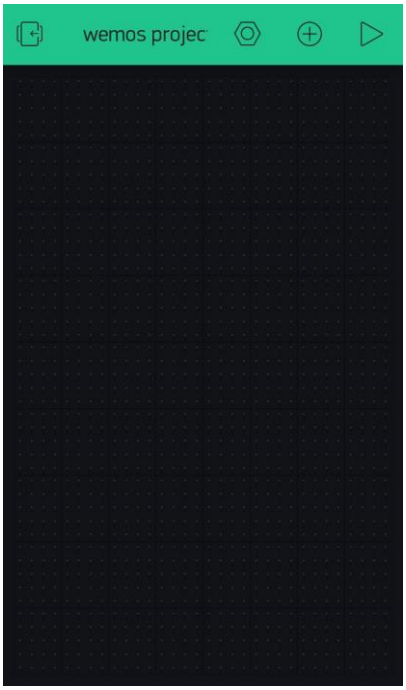
Gambar 4.2 Instalasi Aplikasi *Blynk*

Sebelum menggunakan atau mendesain *user interface* aplikasi *Blynk*, diperlukan untuk membuat *project* baru dengan menekan tombol '*new project*' pada halaman *home* aplikasi *Blynk* seperti pada gambar 4.3 berikut.



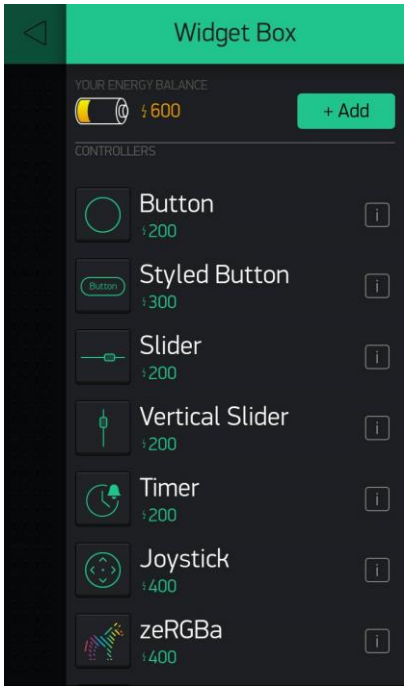
Gambar 4.3 Tampilan *Home* pada Aplikasi *Blynk*

Kemudian mengisi nama *project*, memilih jenis *microcontroller* yang akan digunakan, dan menekan tombol "*create*". Kemudian akan keluar *pop-up* berisi *authentication token* yang telah dikirim ke *email* yang terdaftar pada *account* yang telah dibuat. Kemudian akan muncul *project* baru yang masih kosong seperti yang dapat dilihat pada gambar 4.4 berikut.



Gambar 4.4 Tampilan *New Project* pada Aplikasi *Blynk*

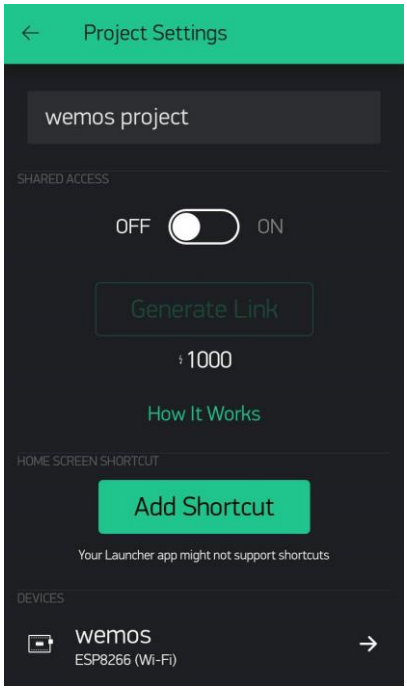
Setelah selesai mendesain user interface pada aplikasi Blynk, user dapat menjalankan aplikasi yang telah dibuat dengan menekan tombol "*play*" seperti yang dapat dilihat pada gambar 4.4 di atas. Pada *project* yang telah dibuat, *user* dapat menambah *widget* dengan mengklik tombol (+) pada pojok kanan atas sehingga akan muncul tampilan seperti gambar 4.5 berikut.



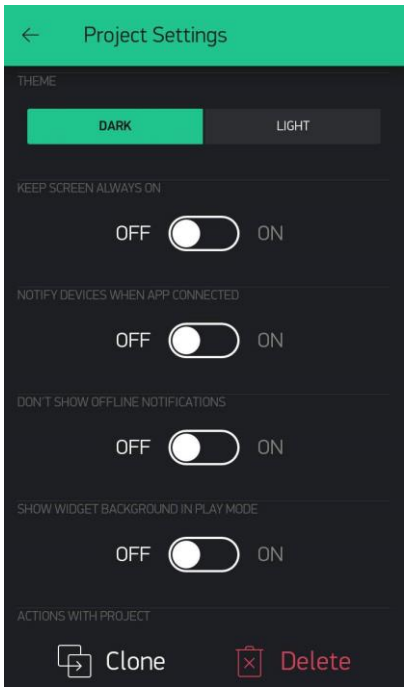
Gambar 4.5 Tampilan Setelah Menekan Tombol *Add Widget*

Seperti yang telah dijelaskan pada bab 3 sebelumnya bahwa pada aplikasi *Blynk* terdapat berbagai variasi jenis *widget*. Untuk setiap jenis *widget* terdapat beberapa *widget* yang dapat digunakan. Untuk *user* yang baru pertama kali menggunakan aplikasi ini, diharuskan untuk membuat *account* baru dan setelah itu mendapatkan 2000 *energy* secara gratis. *Energy* tersebut digunakan untuk membeli *widget* yang akan digunakan pada *project*. Setiap *widget* memiliki *energy* yang bervariasi dari 100 *energy* hingga 1900 *energy*. Jika jumlah *energy* dirasa kurang, maka *user* dapat membeli *energy* sesuai dengan kebutuhan.

Untuk mengatur pengaturan *project* dapat menekan tombol yang berbentuk seperti mur atau dapat dilihat pada gambar 4.5 di atas. Setelah menekan tombol tersebut maka akan ditampilkan tampilan seperti pada gambar 4.6 dan gambar 4.7 berikut.



Gambar 4.6 Tampilan *Project Settings*



Gambar 4.7 Tampilan *Project Settings* (Lanjutan)

Pada *project settings*, *user* dapat mengubah pengaturan *project* sesuai dengan keinginan. *User* dapat menyalakan/ mematikan tombol "*on-off*" pada *shared devices*. User juga dapat menambahkan shortcut pada home screen dengan mengklik tombol "*Add shortcut*". Untuk melihat *authentication token* pada *project* ini, dapat dengan menekan pada bagian *devices* pada gambar 4.7 di atas. Untuk

tampilan, user dapat mengganti tema warna *background project* (antara *dark*/ hitam dan *light*/ putih). *User* juga dapat mengganti pengaturan lainnya dengan menglik *switch button* pada gambar 4.7 di atas.

4.2 Implementasi Sistem

Implementasi sistem pada penelitian ini berupa program. Program yang dibuat menggunakan bahasa pemrograman PHP dan C++. Pemrograman dengan menggunakan bahasa PHP digunakan untuk data yang berhubungan dengan *database* dan *website*. Di antaranya yaitu konfigurasi pada *database* MySQL, koneksi ke *database* MySQL, melakukan HTTP *request*, dan melakukan SQL *command*. Sedangkan pemrograman dengan menggunakan C++ pada Arduino *Integrated Development Environment* (Arduino IDE) digunakan untuk mendapatkan data dari setiap sensor serta mengirim data menuju *database* menggunakan HTTP *request method*.

4.2.1 Program menggunakan bahasa pemrograman PHP

Agar *website* dapat terhubung dengan *database* MySQL, ada beberapa cara yang dilakukan. Salah satunya dengan menggunakan bahasa pemrograman PHP. Dengan memanfaatkan *public website hosting*, *file* PHP dan *database* dapat ditempatkan pada *url* yang sama sehingga dapat diakses oleh Arduino IDE. Dalam *public website hosting*, biasanya terdapat *file manager* sebagai tempat up*load file-file* untuk membuat *website*. Sebagian besar *public website hosting* juga dapat terintegrasi dengan phpMyAdmin sehingga *database* MySQL dapat di-*export* dari phpMyAdmin lokal dan di-*import*-kan ke phpMyAdmin pada *public website hosting*.

4.2.1.1 Konfigurasi *Default Timezone* pada *Database* MySQL

Pada kondisi awal konfigurasi MySQL setelah *instal*, *Europe*/Helsinki, US/*Eastern*, atau MET adalah *default timezone* yang sudah otomatis terpasang pada MySQL. Dengan perintah ’*date_default_timezone_set’*, konfigurasi dapat dilakukan sesuai benua dan kota masing-masing. Untuk mengubah format waktu, dapat dilakukan dengan perintah *’date()*’ dengan parameter ’Y’ untuk 4 *digit*

tahun, ’m’ untuk 2 *digit* numeric bulan, ’d’ untuk 2 *digit* numeric tanggal, ’H’ untuk format 24 jam, ’i’ untuk menit dengan awalan 0, dan ’s’ untuk detik dengan awalan 0. Konfigurasi *default timezone* pada *database* dapat dilihat pada Segmen Program 4.1 berikut.

```
// Set the default timezone
date_default_timezone_set('Asia/Jakarta');
$date = date('Y/m/d H:i:s');
```

Segmen Program 4.1 *Source Code* Konfigurasi *Default Timezone Database*

4.2.1.2 Koneksi ke *Database* MySQL

Koneksi *database* diperlukan untuk menghubungkan *website* dengan *database*. Dengan adanya koneksi *database*, mengontrol *database* dapat dilakukan melalui *website* seperti menambah data dan mengelola data yang tersimpan. Koneksi *database* dibuat di *file* PHP. *Source code* koneksi *database* program dapat dilihat pada Segmen Program 4.2 berikut.

```
// Set database connection 2
$servername = "localhost";
$dbname = "xxx";
$username = "xxx";
$password = "xxx";

…

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

…

if ($conn->multi_query($sql) === TRUE) {
     echo "New record created successfully";
 }
else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}

// Close connection
$conn->close();
```

Segmen Program 4.2 *Source Code* Koneksi *Database*

4.2.1.3 HTTP *Request Method*

HTTP *Request Method* digunakan untuk mengetahui apakah ada *request* yang dikirim. Terdapat 4 metode yang paling umum digunakan adalah POST, POST, PUT, dan DELETE. Metode POST digunakan untuk menambah data dari *server* menuju dalam *database* melalui parameter yang diberikan pada URI. Metode GET hanya digunakan untuk mendapat data dan tidak bisa mengubah pada data pada *server*. Metode POST digunakan untuk mengirim data beserta parameter yang tersimpan pada *body* dari HTTP *request* ke *server*. Metode POST biasa digunakan untuk menambah data/ baris baru pada tabel. Metode PUT digunakan untuk mengirim data ke *server* dan biasa digunakan untuk meng-*update* data/ mengganti/ meng-*edit* data dengan data terbaru. Metode DELETE digunakan untuk menghapus data pada *server*. HTTP *Request Method* yang dipakai adalah POST dan dapat dilihat pada Segmen Program 4.3 berikut.

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    …
}
else {
    echo "No data posted with HTTP POST request";
}
```

Segmen Program 4.3 *Source Code* HTTP GET *Request Method*

4.2.1.4 SQL *Command*

SQL *command* diperlukan untuk mengakses data pada *database* MySQL. SQL *command* yang sering digunakan adalah SELECT, INSERT, UPDATE, dan DELETE. Perintah SELECT digunakan untuk menampilkan data dari *database*. Perintah INSERT digunakan untuk menambah data pada *database*. Perintah UPDATE digunakan untuk mengganti data pada *database* dengan data yang baru. Perintah DELETE digunakan untuk menghapus data pada *database*. SQL *command* yang digunakan adalah INSERT untuk menambah data (yang didapat dari sensor-sensor) pada *database*. *Source code* SQL INSERT *command* dapat dilihat pada Segmen Program 4.4 berikut.

```
if ($_POST["time"]) { // To get time data that passed from RTC
in WeMos microcontroller
    $date = $_POST["time"];
}
if ($_POST ["temperatureWater"]) { // SQL command to insert
temperature sensor data to MySQL database
    $sql .= "INSERT INTO temperaturedata (temperatureDataId,
temperatureWater, temperatureDate) VALUES (null,
".$_POST["temperatureWater"].", ".$date.");";
}
if ($_POST["heightWater"]) { // SQL command to insert
ultrasonic data to MySQL database
    $sql = "INSERT INTO ultrasonicdata (ultrasonicDataId,
heightWater, ultrasonicDate) VALUES (null,
".$_POST["heightWater"].", ".$date.");";
}
if ($_POST ["bodyWeight"]) { // SQL command to insert load cell
sensor data to MySQL database
    $sql .= "INSERT INTO loadcelldata (loadcellDataId,
bodyWeight, loadcellDate) VALUES (null,
".$_POST["bodyWeight"].", ".$date.");";
}
if ($_POST ["phWater"]) { // SQL command to insert ph sensor
data to MySQL database
    $sql .= "INSERT INTO phdata (phDataId, phWater, phDate)
VALUES (null, ".$_POST["phWater"].", ".$date.");";
}
```

Segmen Program 4.5 *Source Code* SQL *Command*

4.2.2 Program pada Arduino IDE

Arduino IDE digunakan untuk memprogram *microcontroller* beserta sensor-sensor yang digunakan dengan menggunakan bahasa C++. *Microcontroller* yang di-*support* oleh Arduino IDE beraneka ragam, beberapa di antaranya adalah *Adafruit*, Arduino, DFRobot, ESP8266, dan *SparkFun*. Sensor-sensor yang digunakan antara lain sensor *ultrasonic*, sensor temperatur, sensor pH, dan sensor *load cell*. Data-data yang didapat dari sensor akan dimasukkan ke dalam database. Agar *database* dapat diakses oleh Arduino IDE maka juga dibutuhkan pemrograman dari Arduino IDE.

4.2.2.1 Sensor DS18B20 *Water Temperature*

Setelah memasukkan *library* yang dibutuhkan dan melakukan inisialisasi, sensor temperatur akan dijalankan dan data sensor temperatur akan dibaca. Data sensor temperatur yang dibaca akan dimasukkan ke dalam variabel x pada

persamaan y = 0.96102302970732x + 4.073482050974 dengan rumus y = mx + b di mana m = 0.96102302970732 dan b = 4.073482050974 berdasarkan hasil kalibrasi yang telah dijelaskan pada sub bab sebelumnya (Sub Bab 4.1.1). setelah mendapat hasil dari persamaan tersebut yang tersimpan pada variabel y maka akan dilakukan pengecekan. Jika sensor temperatur melebihi batas suhu yang di-*input* dari aplikasi *Blynk*, maka *fan cooler* akan dinyalakan. Sebaliknya jika sensor temperatur kurang dari atau sama dengan batas suhu yang di-*input* dari aplikasi *Blynk*, maka *fan cooler* akan dimatikan. Data yang dibaca dari sensor temperatur akan dimasukkan ke dalam *database*. *Source code* sensor temperatur dapat dilihat pada Segmen Program 4.6 berikut.

```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

#include <OneWire.h>
#include <DallasTemperature.h>

// Set Blynk Authentication
char auth[] = "xxx";

// Set WiFi credentials.
char ssid[] = "xxx";
char pass[] = "xxx";

// Data wire is plugged into pin 2 on the Arduino
#define ONE_WIRE_BUS 4 //D14
int relay_pump_pin = D8;
WidgetTerminal terminal(V4);
BlynkTimer timer; // Announcing the timer

// Setup a oneWire instance to communicate with any OneWire
devices
OneWire oneWire(ONE_WIRE_BUS);

// Pass our oneWire reference to Dallas Temperature.
DallasTemperature temp_sensor(&oneWire);

float celcius;
float input_temp;
String fan_state = "OFF";

double x_temp, m_temp, b_temp, y_temp;

  double y_temp = m_temp * x_temp + b;_temp
// Gets the value stored in V5 (numeric input) as an integer
BLYNK_WRITE(V5) {
  input_temp = param.asFloat();
}
```

Segmen Program 4.6 *Source Code* Sensor Temperatur

```
void setup(void)
{
  // Initialize the Serial Monitor at a baud rate of 9600
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);

  // Initialize the sensor
  temp_sensor.begin();

  while (!Serial);
  pinMode(relay_pump_pin, OUTPUT); // variant low/high
  digitalWrite(relay_pump_pin, HIGH); // pump deactivated

  // Setup a function to be called every second
  timer.setInterval(1000L, myTimerEvent);
}

void myTimerEvent()
{
  terminal.clear();
  // call sensors.requestTemperatures() to issue a global
temperature
  // request to all devices on the bus
  // Send the command to get temperature readings
  temp_sensor.requestTemperatures();

  x_temp = celcius;
  m_temp = 0.96102302970732;
  b_temp = 4.073482050974;

  y_temp = m_temp * x_temp + b_temp;

  // Get and print the temperature in Celcius
  Serial.print("Temperature is: ");
  Serial.print(y_temp);

  String temp = String(y_temp) + " °C";
  Blynk.virtualWrite(V6, temp); // Show output in Widget Value
Display(Temperature Value Display)

  if (y_temp < input_temp)
  {
    digitalWrite(relay_pump_pin, HIGH); // fan deactivated
    fan_state = "OFF";
    Serial.println(" fan OFF");
  }
  else if (y_temp >= input_temp + 3)
  {
    digitalWrite(relay_pump_pin, LOW); // fan activated
    fan_state = "ON";
    Serial.println(" fan ON");
  }

  terminal.flush();
```

Segmen Program 4.6 *Source Code* Sensor Temperatur (Lanjutan)

```
  // Check the current connection status
  if ((WiFi.status() == WL_CONNECTED)) {
    HTTPClient http;
    http.begin(str);
    http.addHeader("Content-Type", "application/x-www-form-
urlencoded");

    // Make the POST request
    String query = "time='" + string_time +
"&temperatureWater=" + y_temp;
    Serial.println(query);
    int httpResponse = http.POST(query);

    if (httpResponse > 0) { //Check for the returning code
        String payload = http.getString();
        Serial.println("HTTP Response code: ");
        Serial.println(httpResponse);
        Serial.println(payload);
        Serial.println("-----------------------------------");
      }
    else {
      Serial.println("Error on HTTP request");
      Serial.print("Error code: ");
      Serial.println(httpResponse);
    }

    // End the connection
    http.end();
}

void loop(void)
{
  Blynk.run();
  timer.run();  // Initiates BlynkTimer
}
```

Segmen Program 4.6 *Source Code* Sensor Temperatur (Lanjutan)

4.2.2.2 Sensor *Ultrasonic*

Setelah memasukkan *library* yang dibutuhkan dan melakukan inisialisasi, sensor *ultrasonic* akan dijalankan dan data sensor *ultrasonic* akan dibaca dan dilakukan pengecekan. Jika data sensor *ultrasonic* melebihi atau sama dengan *input*an tinggi maksimum dari aplikasi *Blynk*, maka *water pump* akan dinyalakan. Sebaliknya jika sensor *ultrasonic* kurang dari *input*an tinggi maksimum dari aplikasi *Blynk*, maka *water pump* akan dimatikan. Data yang dibaca dari sensor *ultrasonic* akan dimasukkan ke dalam *database*. *Source code* sensor *ultrasonic* dapat dilihat pada Segmen Program 4.7 berikut.

```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>

String serverName =
"http://akuarium.ejournal.web.id/insert_data.php";

// Set Blynk Authentication
char auth[] = "xxx";

// Set WiFi credentials.
char ssid[] = "xxx";
char pass[] = "xxx";

#define TRIGGERPIN D11
#define ECHOPIN    D10
int relay_pump_pin = D7;

int input_max_height = 5;
String pump_state = "OFF";

long duration, distance, real_distance; // ultrasonic

WidgetTerminal terminal(V4); // Attach virtual serial terminal
to Virtual Pin V4
BlynkTimer timer; // Announcing the timer

BLYNK_WRITE(V4) { //Get data from Terminal Widget from Blynk
App to hardware
  input_max_height = param.asInt();
}

void setup()
{
  // Debug console
  Serial.begin(115200);

  // Setting Wifi connection to database
  WiFi.begin(ssid, pass);
  Serial.println("Connecting");
  while(WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  // Set input dan output in ultrasonic sensor
  pinMode(TRIGGERPIN, OUTPUT);
  pinMode(ECHOPIN, INPUT);

  Blynk.begin(auth, ssid, pass);

  // Setup a function to be called every period of time
  timer.setInterval(1000L, myTimerEvent); // 1 second
```

Segmen Program 4.7 *Source Code* Sensor *Ultrasonic*

```
  while (!Serial);
  pinMode(relay_pump_pin, OUTPUT); // variant low/high
  digitalWrite(relay_pump_pin, HIGH); // pump deactivated
}

void myTimerEvent()
{
  // ----------------------ULTRASONIC-----------------------
  digitalWrite(TRIGGERPIN, LOW);
  delayMicroseconds(3);
  digitalWrite(TRIGGERPIN, HIGH);

  delayMicroseconds(12);
  digitalWrite(TRIGGERPIN, LOW);

  duration = pulseIn(ECHOPIN, HIGH);
  // distance = (traveltime/2) x speed of sound
  // The speed of sound is: 343m/s = 0.0343 cm/uS = 1/29.1
cm/uS
  distance = (duration/2) / 29.1; // Convert the time into a
distance

  // ----------------------DISPLAY-----------------------
  terminal.clear(); // Clear the terminal content

  // Show max height in Terminal Widget
  terminal.print("MAX height from top: ");
  terminal.println(input_max_height);

  // Show distance in Widget Terminal
  terminal.print("CURRENT height from top: ");
  terminal.println(distance);

  // Show distance in Serial Monitor
  Serial.print(distance);
  Serial.println(" cm");

  // ----------------------RELAY-----------------------
  if (distance <= input_max_height) {
    digitalWrite(relay_pump_pin, HIGH); // pump deactivated
    pump_state = "OFF";
  }
  else if (distance >= input_max_height + 4) {
    digitalWrite(relay_pump_pin, HIGH); // pump activated
    pump_state = "ON";
    Blynk.email("keziak.019@gmail.com", "Wemos Alert", "Low
water height but water pump on!"); // Set the trigger and send
email
  }

  terminal.print("Water Pump ");
  terminal.println(pump_state);
```

Segmen Program 4.7 *Source Code* Sensor *Ultrasonic* (Lanjutan)

```
  // --------------------- Blynk ----------------------------
  Blynk.virtualWrite(V3, distance); // Show output in Widget
Level V (Water Level)
  terminal.flush(); // Ensure everything is sent

  Check the current connection status
  if ((WiFi.status() == WL_CONNECTED)) {
    HTTPClient http;
    http.begin(str);
    http.addHeader("Content-Type", "application/x-www-form-
urlencoded");

    // Make the POST request
    String query = "time='" + string_time + "'&heightWater=" +
distance;

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

    // Make the POST request
    String query = "time='" + string_time + "'&heightWater=" +
distance;
    Serial.println(query);
    int httpResponse = http.POST(query);

    if (httpResponse > 0) { //Check for the returning code
        String payload = http.getString();
        Serial.println("HTTP Response code: ");
        Serial.println(httpResponse);
        Serial.println(payload);
        Serial.println("-----------------------------------------
-");
      }
    else {
      Serial.println("Error on HTTP request");
      Serial.print("Error code: ");
      Serial.println(httpResponse);
    }

    // End the connection
    http.end();
  }
}

void loop()
{
  Blynk.run();
  timer.run(); // Initiates BlynkTimer
}
```

Segmen Program 4.7 *Source Code* Sensor *Ultrasonic* (Lanjutan)

4.2.2.3 Sensor *Load Cell*

Setelah memasukkan *library* yang dibutuhkan dan melakukan inisialisasi, sensor *load cell* akan dijalankan. Data sensor *load cell* akan dibaca dan ditampilkan pada aplikasi *Blynk*. Sekitar 3-4% dari berat total ikan akan menjadi berat total makanan ikan yang akan diberikan. Jika data sensor *load cell* termasuk pada *range* berat tertentu maka akan mempengaruhi kecepatan *servo motor*. *Source code* sensor *load cell* dapat dilihat pada Segmen Program 4.8 berikut.

```
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

#include "HX711.h"

#include <Wire.h>       // I2C library
#include <RtcDS3231.h>  // RTC library
#include <Servo.h>      // Servo library

// Set Blynk Authentication
char auth[] = "xxx";

// Set WiFi credentials.
char ssid[] = "xxx";
char pass[] = "xxx";

BlynkTimer timer;

#define DOUT  D5
#define CLK  D6

HX711 scale;

float calibration_factor = 455004;
float decimal_digits = 3;

//--------------- RTC Servo -------------

RtcDS3231<TwoWire> rtcObject(Wire); Servo myservo;  // Create
servo object to control a servo
WidgetTerminal terminal(V4); // Attach virtual serial terminal
to Virtual Pin V4

int input_hour1, input_hour2;

// Gets the value stored in V1 (numeric input) as an integer
BLYNK_WRITE(V1) {
  input_hour1 = param.asInt();
}
// Gets the value stored in V2 (numeric input) as an integer
BLYNK_WRITE(V2) {
  input_hour2 = param.asInt();
}
```

Segmen Program 4.8 *Source Code* Sensor *Load Cell*

```
void setup() {
  Serial.begin(115200);
  rtcObject.Begin();    // Starts I2C
  Blynk.begin(auth, ssid, pass);

  scale.begin(DOUT, CLK);
  scale.set_scale();
  scale.tare(); // Reset the scale to 0

  long zero_factor = scale.read_average(); // Get a baseline
reading

  Serial.print("Zero factor: "); // This can be used to remove
the need to tare the scale. Useful in permanent scale projects.
  Serial.println(zero_factor);

  // attaches the servo on pin 2 to the servo object
  myservo.attach(D2);
  myservo.write(0);  // Tell servo to go to 90 degree position

  timer.setInterval(1000L, myTimerEvent); // 1 second
}

void myTimerEvent()
{
  scale.set_scale(calibration_factor); //Adjust to this
calibration factor
  Serial.print("Reading: ");

  // Get data from scale with 5 digit after decimal point
  float tmp = scale.get_units();
  char tmp2[8];
  dtostrf(tmp, 8, 5, tmp2);

  // Converting from pound(lb) to gram
  String tmp3 = tmp2;
  float tmp4 = tmp3.toFloat();
  float tmp5 = tmp4 * 453.592;

  // Remove negative output value
  if(tmp5 < 0) {
    tmp5 = tmp5 * -1;
  }

  String tmp6 = String(tmp5) + " gr";
  Blynk.virtualWrite(V7, tmp6);

  Serial.print(tmp5);
  Serial.print(" gr");

  Serial.print(" calibration_factor: ");
  Serial.print(calibration_factor);
  Serial.println();

  // get the time from the RTC
  RtcDateTime currentTime = rtcObject.GetDateTime();
```

Segmen Program 4.8 *Source Code* Sensor L*oad Cell* (Lanjutan)

```
  // declare a string as an array of chars
  char str[15];

  int year, month, day, hour, minute, second;
  year = currentTime.Year();   // get year method
  month = currentTime.Month();  // get month method
  day = currentTime.Day();    // get day method
  hour = currentTime.Hour();   // get hour method
  minute = currentTime.Minute(); // get minute method
  second = currentTime.Second();  // get second method

  sprintf(str, "%d/%d/%d %d:%d:%d",   // %d allows to print an
int to the string
                year,
                month,
                day,
                hour,
                minute,
                second
  );
  Serial.print(str);

  if ( ((hour == input_hour1 || hour == input_hour1 + 1) ||
(hour == input_hour2 || hour == input_hour2 + 1)) && minute ==
0 && second == 0 ) {

    int n_fish = 8;
    float tot_food = tmp5 * n_fish * 0.04;

    if (tot_food <= 2.5) {
      myservo.write(90);  // tell servo to go to 90 degree
position
      delay(100);       // waits 100 ms for the servo to reach
the position
      myservo.write(0);   // tell servo to go to 0 degree
position
      delay(100);       // waits 100 ms for the servo to reach
the position
    }
    else if (tot_food <= 5) {
      myservo.write(90);  // tell servo to go to 90 degree
position
      delay(150);       // waits 150 ms for the servo to reach
the position
      myservo.write(0);   // tell servo to go to 0 degree
position
      delay(150);       // waits 150 ms for the servo to reach
the position
    }
    else if (tot_food <= 7) {
      myservo.write(90);  // tell servo to go to 90 degree
position
      delay(170);       // waits 170 ms for the servo to reach
the position
      myservo.write(0);   // tell servo to go to 0 degree
position
```

Segmen Program 4.8 *Source Code* Sensor L*oad Cell* (Lanjutan)

```
      delay(170);       // waits 170 ms for the servo to reach
the position
    }
    else if (tot_food <= 9) {
      myservo.write(90);  // tell servo to go to 90 degree
position
      delay(190); // waits 190 ms for the servo to reach the
position
      myservo.write(0); // tell servo to go to 0 degree
position
      delay(190); // waits 190 ms for the servo to reach the
position
    }

  } else {
    // print the string to the serial port
    Serial.print(" ");
    Serial.print(input_hour1);
    Serial.print(":");
    Serial.println(input_hour2);
  }

  // Check the current connection status
  if ((WiFi.status() == WL_CONNECTED)) {
    HTTPClient http;
    http.begin(str);
    http.addHeader("Content-Type", "application/x-www-form-
urlencoded");

    // Make the POST request
    String query = "time='" + string_time + "&bodyWeight=" +
loadcell;
    Serial.println(query);
    int httpResponse = http.POST(query);

    if (httpResponse > 0) { //Check for the returning code
        String payload = http.getString();
        Serial.println("HTTP Response code: ");
        Serial.println(httpResponse);
        Serial.println(payload);
        Serial.println("-----------------------------------");
      }
    else {
      Serial.println("Error on HTTP request");
      Serial.print("Error code: ");
      Serial.println(httpResponse);
    }

    // End the connection
    http.end();
  }
  else {
    Serial.println("WiFi Disconnected!");
  }
}
```

Segmen Program 4.8 *Source Code* Sensor L*oad Cell* (Lanjutan)

```
void loop()
{
  Blynk.run();
  timer.run(); // Initiates BlynkTimer
}
```

Segmen Program 4.8 *Source Code* Sensor L*oad Cell* (Lanjutan)

4.2.2.4 Sensor pH

Setelah memasukkan *library* yang dibutuhkan dan melakukan inisialisasi, sensor pH akan dijalankan. Berdasarkan hasil kalibrasi yang dilakukan pada Sub-Bab 4.1.1, didapatkan persamaan y = 2.1856210042767x - 0.6410403120014 yang berasal dari rumus y = mx + b di mana m = 2.1856210042767, b = 0.6410403120014. Hasil persamaan tersebut dimasukkan pada program yang akan dijalankan dengan data sensor pH yang dimasukkan ke dalam variabel x. Kemudian akan didapatkan hasil persamaan yang disimpan dalam variabel y dan ditampilkan pada aplikasi *Blynk* serta dimasukkan ke dalam *database*. *Source code* sensor pH dapat dilihat pada Segmen Program 4.9 berikut.

```
#define BLYNK_PRINT Serial

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// Set Blynk Authentication
char auth[] = "xxx";

// Set WiFi credentials.
char ssid[] = "xxx";
char pass[] = "xxx";

BlynkTimer timer; // Announcing the timer

//const int phSensorPin  = A0;
#define phSensorPin 0

void setup()
{
  Serial.begin(115200);
  pinMode (phSensorPin, INPUT);

  Blynk.begin(auth, ssid, pass);
  timer.setInterval(1000L, myTimerEvent); // 1 second
}
```

Segmen Program 4.9 *Source Code* Sensor pH

```
void myTimerEvent()
{
  int adc_value = analogRead(phSensorPin);
  Serial.print("ADC value: ");
  Serial.print(adc_value);

  // Calibrating
  double voltage = adc_value * 5 / 1024.0; // convert the
analog into millivolt
//   Serial.print(" Voltage value: ");
//   Serial.println(voltage, 3);

   double x = voltage;
   double m = 2.185;
   double b = -0.641;
   // m = 2.185; b = -0.641; accuracy ph in buffer 7 = +- 0.2pH
   // m = 1.832; b = 1.107; accuracy ph in buffer 7 = +- 1pH

   // Using linear regression algebra to convert millivolt into
pH value
   double y = m * x + b;

   // random value
//  phValue = 3.5 * phValue;                                  //
convert the millivolt into pH value
//  phValue = rand() % 1200 + 7100;
//  d = phValue / 1000;

   Serial.print(" pH: ");
   Serial.println(y);
   Blynk.virtualWrite(V8, y);

  // Check the current connection status
  if ((WiFi.status() == WL_CONNECTED)) {
    HTTPClient http;
    http.begin(str);
    http.addHeader("Content-Type", "application/x-www-form-
urlencoded");

    // Make the POST request
    String query = "time='" + string_time + "&phWater=" + y_ph;
    Serial.println(query);
    int httpResponse = http.POST(query);

    if (httpResponse > 0) { //Check for the returning code
        String payload = http.getString();
        Serial.println("HTTP Response code: ");
        Serial.println(httpResponse);
        Serial.println(payload);
        Serial.println("-----------------------------------");
      }
    else {
      Serial.println("Error on HTTP request");
      Serial.print("Error code: ");
      Serial.println(httpResponse);
    }
```

Segmen Program 4.9 *Source Code* Sensor pH

```
    // End the connection
    http.end();

  }
  else {
    Serial.println("WiFi Disconnected!");
  }
}

void loop()
{
  Blynk.run();

  // Initiates BlynkTimer
  timer.run();
}
```

Segmen Program 4.9 *Source Code* Sensor pH