BAB 5. PENGUJIAN SISTEM

5.1 Kalibrasi Sensor

Sebelum menggunakan sensor yang akan dipakai pada penelitian ini, maka diperlukan kalibrasi untuk mendapatkan hasil output yang tepat. Namun tidak semua sensor memerlukan kalibrasi sebelum sensor dipakai seperti sensor *ultrasonic*. Karena sensor *ultrasonic* memakai rumus untuk mencari jarak dengan variabel waktu dan kecepatan udara.

5.1.1 Kalibrasi Sensor Temperatur

Sebelum sensor berat digunakan pada penelitian ini, diperlukan kalibrasi sensor berat untuk mendapat hasil *output* dengan tingkat akurasi yang tinggi. Langkah pertama untuk kalibrasi sensor temperatur yaitu dengan menyiapkan semua alat (sensor temperatur, sensor temperatur yang sudah teruji, dan WeMos) dan bahan (panci berisi air dan gelas berisi air dan es) yang diperlukan. Sensor temperatur akan dicelupkan pada air yang mendidih untuk batas tinggi temperatur serta dicelupkan pada air yang berisi banyak es untuk batas rendah temperatur. Langkah kedua yaitu meng-*upload* program dengan menancapkan kabel USB pada laptop serta mencelupkan sensor temperatur pada larutan. Program yang di-*upload* dapat dilihat pada segmen program 5.1 berikut.

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

// Data wire is plugged into pin 2 on the Arduino
#define ONE_WIRE_BUS 4

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);
```

Segmen Program 5.1 Program Kalibrasi Sensor Temperatur

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

void setup(void)
{
  // Initialize the Serial Monitor at a baud rate of 9600
  Serial.begin(115200);

  // Initialize the sensor
  temp_sensor.begin();

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

void myTimerEvent()
{
  // call sensors.requestTemperatures() to issue a global
temperature
  // request to all devices on the bus
  temp_sensor.requestTemperatures(); // Send the command to get
temperature readings
  celcius = temp_sensor.getTempCByIndex(0);

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

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

Segmen Program 5.1 Program Kalibrasi Sensor Temperatur (Lanjutan)

Langkah keempat yaitu dengan mendidihkan air yang berada pada panci di atas kompor. Setelah mendidih maka dapat mencelupkan ujung sensor temperatur (yang terdiri dari besi) sekaligus mencelupkan sensor temperatur lain ke dalam panci sehingga kedua sensor akan membaca temperatur pada air mendidih. Selanjutnya menunggu hingga angka yang ditampilkan cenderung stabil, bila angka yang ditampilkan sudah stabil, catat kedua hasil yang didapat. Langkah kelima yaitu dengan mengulangi langkah keempat namun dengan air es sehingga mendapatkan 2 nilai *output* yang berbeda. Langkah keenam yaitu dengan mengulangi beberapa kali langkah keempat namun dengan temperatur yang berbeda-beda. Langkah ketujuh yaitu dengan memasukkan persamaan yang telah didapatkan pada Sub Bab 4.1.1 ke dalam program serta meng-*upload*

program dengan persamaan yang telah didapat. Untuk program selengkapnya dapat dilihat pada segmen program 5.2 berikut.

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

// Data wire is plugged into pin 2 on the Arduino
#define ONE_WIRE_BUS 4

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";

void setup(void)
{
  // Initialize the Serial Monitor at a baud rate of 9600
  Serial.begin(115200);

  // Initialize the sensor
  temp_sensor.begin();

  // 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
  temp_sensor.requestTemperatures(); // Send the command to get
temperature readings

  celcius = temp_sensor.getTempCByIndex(0);

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

  double y_temp = m_temp * x_temp + b_temp;

  String hsl_celcius = String(y_temp) + " °C";
```

Segmen Program 5.2 Program Sensor Temperatur

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

  if (y_temp <= input_temp - 2) {
    digitalWrite(relay_temp_pin, HIGH); // fan deactivated
    fan_state = "OFF";
  }
  else if (y_temp >= input_temp) {
    digitalWrite(relay_temp_pin, LOW); // fan activated
    fan_state = "ON";
  }
}

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

Segmen Program 5.2 Program Sensor Temperatur

5.1.2 Kalibrasi Sensor *Load Cell*

Sebelum sensor *load cell*/ sensor berat digunakan pada penelitian ini, diperlukan kalibrasi sensor berat untuk mendapat hasil *output* dengan tingkat akurasi yang tinggi. Langkah pertama untuk kalibrasi sensor berat yaitu menyiapkan semua alat (sensor berat dan WeMos) dan bahan (bahan yang akan ditimbang dan alat timbang lainnya) yang diperlukan. Langkah kedua yaitu dengan menimbang bahan yang akan ditimbang dengan alat timbang lainnya dan mencatat hasil timbangan. Hasil timbangan tersebut akan dibandingkan dengan hasil dari sensor berat. Semakin kecil selisih antara hasil timbangan dengan hasil dari sensor berat, maka semakin tinggi tingkat akurasinya. Langkah ketiga yaitu dengan meng-*upload* program pada Arduino IDE. Perlu diingat bahwa ketika meng-*upload* program, keadaan sensor berat harus dalam keadaan tidak ada berat apapun di atasnya. Untuk program yang di-*upload* dapat dilihat pada Segmen Program 5.3 berikut.

```
#define BLYNK_PRINT Serial
#include "HX711.h"

#define DOUT  D5
#define CLK  D6

BlynkTimer timer;
HX711 scale;

float calibration_factor = 455004;
float decimal_digits = 3;

void setup() {
  Serial.begin(115200);
  Serial.println("HX711 calibration sketch");
  Serial.println("Remove all weight from scale");
  Serial.println("After readings begin, place known weight on
scale");
  Serial.println("Press + or a to increase calibration
factor");
  Serial.println("Press - or z to decrease calibration
factor");

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

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

void myTimerEvent()
{
  scale.set_scale(calibration_factor); //Adjust to this
calibration factor

  // 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;
  }

  Serial.print("Reading: ");
  Serial.print(tmp5);
  Serial.print(" gr");
```

Segmen Program 5.3 Program Kalibrasi Sensor *Load Cell*

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

  if(Serial.available())
  {
    char temp = Serial.read();
    if(temp == '+' || temp == 'a')
      calibration_factor += 1;
    else if(temp == '-' || temp == 'z')
      calibration_factor -= 1 ;
  }
}

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

Segmen Program 5.3 Program Kalibrasi Sensor *Load Cell* (Lanjutan)

Setelah meng-*upload* program tersebut, langkah keempat yaitu mengecek apakah hasil yang ditampilkan pada *Serial Monitor* pada Arduino IDE mendekati angka 0 jika tidak ada beban yang diletakkan pada bagian atas sensor berat. Jika angka yang ditampilkan sudah sangat mendekati angka 0 maka dapat langsung lanjut ke langkah terakhir, sebaliknya jika angka yang ditampilkan sudah sangat tidak mendekati angka 0 atau masih jauh dari angka 0 maka dapat lanjut ke langkah selanjutnya (langkah kelima). Langkah kelima yaitu dengan menekan tombol '+' atau tombol '-' untuk menaikkan atau menurunkan nilai *calibration factor* sehingga hasil berat yang didapatkan mendekati dengan hasil timbangan yang telah dicatat. Jika hasil yang didapatkan pada sensor berat sudah sangat mendekati dengan hasil timbangan yang telah dicatat, maka melakukan langkah keenam yaitu mencatat nilai *calibration factor*. Nilai tersebut yang akan dijadikan sebagai patokan untuk menetralkan sensor berat sehingga ketika tidak ada beban yang terdapat pada sensor berat maka akan ditampilkan angkat berat mendekati 0 gram. Langkah terakhir yaitu memasukkan nilai *calibration factor* pada program (seperti yang dapat dilihat pada Segmen Program 5.4 berikut) untuk mengetes hasil kalibrasi dari sensor berat serta meng-*upload* program dan setelah selesai *upload*, tunggu hingga keluar angka 0 pada *Serial Monitor* pada Arduino IDE, kemudian meletakkan bahan yang akan ditimbang pada bagian atas pada sensor berat. Setelah itu membandingkan hasil yang ditampilkan pada *Serial Monitor*

pada Arduino IDE dengan hasil timbangan yang telah dicatat. Semakin kecil selisihnya, maka semakin akurat pula hasil kalibrasinya.

```
#define BLYNK_PRINT Serial
#include "HX711.h"

#define DOUT  D5
#define CLK  D6

BlynkTimer timer;
HX711 scale;

float calibration_factor = 455004;
float decimal_digits = 3;

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

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

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

void myTimerEvent()
{
  //Adjust to this calibration factor
  scale.set_scale(calibration_factor);

  // 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;
  }

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

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

Segmen Program 5.4 Program Sensor *Load Cell*

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

Segmen Program 5.4 Program Sensor *Load Cell* (Lanjutan)

5.1.3 Kalibrasi Sensor pH

Sebelum sensor pH digunakan pada penelitian ini, diperlukan kalibrasi sensor pH untuk mendapat nilai persamaan yang digunakan untuk mencari nilai pH. Metode yang digunakan yaitu persamaan garis regresi linear di mana metode statistika ini mengukur sejauh mana hubungan variabel penyebab (pembacaan sensor) terhadap variabel akibat (nilai pH). Persamaan garis tersebut digunakan untuk mencari fungsi linear yang menyerupai kumpulan data yang diketahui yang menghasilkan kurva yang bentuknya mendekati garis lurus. garis lurus pada grafik berdasarkan data 2 dimensi. Berikut merupakan persamaan yang digunakan untuk regresi linear:

$$y = mx + b$$

Di mana y merupakan nilai pH, m merupakan gradien, x merupakan nilai ADC sebagai output sensor pH, dan b merupakan konstanta. Sensor pH yang digunakan pada penelitian ini, menghasilkan *output* berupa ADC dengan *input* tegangan sebesar 5 *volt*. Nilai output yaitu nilai ADC yang dihasilkan sensor pH akan digunakan dalam proses kalibrasi untuk dikonversi menjadi satuan pH. Kalibrasi sensor pH dilakukan dengan menggunakan 3 jenis larutan *buffer* yang memiliki pH 4.01, 6.86, dan 9.18. Sensor pH akan diujikan pada setiap jenis larutan *buffer* yang terdiri dari komposisi 250 ml *distilled water* (atau dengan air aki dengan tutup botol biru) dan bubuk *buffer* pH yang masing-masing dilarutkan pada botol yang berbeda.

Langkah pertama untuk kalibrasi sensor pH yaitu menyiapkan semua alat (sensor pH dan WeMos) dan bahan (3 jenis bubuk *buffer*, 3 botol kosong, dan 2 botol *distilled water*/ air aki yang tiap botolnya berisi 600 ml) yang diperlukan. Langkah kedua yaitu menuang 250 ml *distilled water*/ air aki pada masing-masing botol kosong yang sudah disediakan. Langkah ketiga yaitu dengan menuang bubuk *buffer* untuk tiap jenis pada masing-masing botol serta mengaduk hingga

bubuk tersebut larut. Langkah keempat yaitu dengan melakukan *wiring* untuk sensor pH pada WeMos. Langkah kelima yaitu dengan membilas sensor pH dalam *distilled water*/ air aki dan mengeringkan tetesan air dengan *tissue* kemudian memasukkan sensor pH pada salah satu botol yang berisi larutan *buffer*. Langkah keenam yaitu dengan meng-*upload* program pada Arduino IDE (seperti yang dapat dilihat pada Segmen Program 5.5 berikut) dan mengaduk secara perlahan hingga nilai yang ditampilkan pada *Serial Monitor* pada Arduino IDE secara stabil serta mencatat nilai tegangan yang ditampilkan.

```
#define BLYNK_PRINT Serial
#define phSensorPin 0

BlynkTimer timer; // Announcing the timer

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

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

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);
}

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

Segmen Program 5.5 Program Kalibrasi Sensor pH

Langkah ketujuh yaitu dengan mengulangi langkah ke lima hingga enam dengan jenis larutan *buffer* pH yang berbeda. Langkah kedelapan yaitu memasukkan data yang telah dicatat pada link http://www.alcula.com/calculators/statistics/linear-regression/ sehingga akan

mendapatkan persamaan garis regresi linear. Langkah kesembilan yaitu dengan memasukkan persamaan garis regresi linear pada program (seperti yang dapat dilihat pada Segmen Program 5.6 berikut), meng-*upload* program, serta memasukkan sensor pH pada larutan yang akan diuji sehingga mendapat *output* berupa nilai pH dari larutan yang akan diuji.

```
#define BLYNK_PRINT Serial
#define phSensorPin 0

BlynkTimer timer; // Announcing the timer

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

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

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

  double voltage = adc_value * 5 / 1024.0; // convert the
analog into millivolt
  double x = voltage;
  double m = 2.185;
  double b = -0.641;

  // Using linear regression algebra to convert millivolt into
pH value
  double y = m * x + b;
  Serial.print(" pH: ");
  Serial.println(y);
}

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

Segmen Program 5.6 Program Sensor pH

5.2 Pengujian

Pengujian dilakukan dalam 2 lingkungan yang berbeda pada ikan yaitu lingkungan yang tidak menggunakan sistem *monitoring* dan lingkungan yang

menggunakan sistem *monitoring*. Sistem *monitoring* pada penelitian ini tidak hanya memantau 4 parameter pada akuarium (*feeding*, temperatur air, ketinggian air, serta pH air) namun juga mengontrol sebagian besar parameter pada akuarium (kecuali pH air). Lingkungan yang menggunakan sistem *monitoring* akan dikontrol berdasarkan parameter yang telah dijelaskan, namun pada lingkungan yang tidak menggunakan sistem *monitoring* hanya 1 parameter saja yang akan tetap diberikan yaitu *feeding* namun diberi secara tidak terjadwal. Semua pengujian dilakukan dengan rentang waktu yang sama yaitu 1 minggu. Tingkat kematian ikan pada lingkungan yang tidak menggunakan sistem *monitoring* sebesar 50% dengan suhu di atas 30°C. Sedangkan pada lingkungan yang menggunakan sistem *monitoring*, tingkat kematian ikan sebesar 20% dengan suhu sekitar 28°C.

Pengujian juga dilakukan dengan pemberian kondisi yang ekstrem pada ikan. Pada lingkungan pertama yaitu lingkungan dengan ketinggian air yang minim sehingga tempat ikan bergerak semakin sedikit. Pada lingkungan ini ikan mati dalam 5 hari. Pada lingkungan kedua yaitu lingkungan dengan temperatur tinggi (di atas 30°C). Pada lingkungan ini ikan mati dalam 3 jam. Pada lingkungan ketiga yaitu lingkungan dengan pemberian makanan yang berlebih. Pada lingkungan ini, kondisi air akuarium semakin lama semakin keruh akibat komposisi makanan yang terurai sehingga ikan mati dalam 3 hari. Pada lingkungan keempat yaitu lingkungan dengan tidak adanya pemberian makanan pada ikan. Lingkungan tersebut mengakibatkan ikan mati dalam 4 hari.

5.2.1 Pengujian *Database*

Semua data yang dikirim dari sensor-sensor akan dimasukkan ke dalam *database*. Semua data untuk setiap jenis sensor akan dimasukkan ke dalam tabel yang berbeda pada *database* untuk memudahkan pengecekan. Kemudian data dalam *database* akan ditampilkan pada *website* sehingga dapat memonitor kondisi *aquarium* secara *real time*. *Database* berisi 4 tabel, yaitu tabel *ultrasonicData*, tabel *temperatureData*, tabel *loadcellData*, dan tabel *phData*.

Pada tabel *ultrasonicData* terdapat 3 kolom, yaitu *ultrasonicDataId*, *heightWater*, dan *ultrasonicDate*. Kolom *ultrasonicDataId* dijadikan sebagai

primary key tabel *ultrasonicData* dengan tipe data *integer* dan dibuat *auto-increment*. Kolom *heightWater* digunakan untuk menampung data jarak antara tinggi *aquarium* dan tinggi air. Sedangkan kolom *ultrasonicDate* digunakan untuk menampung data waktu (berupa tanggal, jam, menit, serta detik) ketika microcontroller mendapat data dari sensor *ultrasonic* dengan *interval* 15 menit yang di-*setting* dari program. Tabel *ultrasonicData* dapat dilihat pada gambar 5.1 berikut.

ultrasonicDataId ▲ 1	heightWater	ultrasonicDate
1	5	2019-11-23 10:14:41
2	6	2019-11-23 10:29:41
3	6	2019-11-23 10:44:41
4	5	2019-11-23 10:59:41
5	5	2019-11-23 11:14:41
6	6	2019-11-23 11:29:41
7	6	2019-11-23 11:44:41
8	5	2019-11-23 11:59:41
9	5	2019-11-23 12:14:41
10	5	2019-11-23 12:29:41

Tabel 5.1 Tabel *ultrasonicData*

Pada tabel *temperatureData* terdapat 3 kolom, yaitu *temperatureDataId*, *temperatureWater*, dan *temperatureDate*. Kolom *temperatureDataId* dijadikan sebagai *primary key* tabel *temperatureData* dengan tipe data *integer* dan dibuat *auto-increment*. Kolom *temperatureWater* digunakan untuk menampung data temperatur air pada akuarium. Sedangkan kolom *temperatureDate* digunakan untuk menampung data waktu (berupa tanggal, jam, menit, serta detik) ketika microcontroller mendapat data dari sensor temperatur dengan *interval* 15 menit yang di-*setting* dari program. Tabel *temperatureData* dapat dilihat pada gambar 5.2 berikut.

temperatureDataId	temperatureWater	temperatureDate
1	29.82	2019-11-23 10:14:41
2	29.79	2019-11-23 10:29:41
3	29.88	2019-11-23 10:44:41
4	29.83	2019-11-23 10:59:41
5	29.81	2019-11-23 11:14:41
6	29.78	2019-11-23 11:29:41
7	29.75	2019-11-23 11:44:41
8	29.72	2019-11-23 11:59:41
9	29.84	2019-11-23 12:14:41
10	29.9	2019-11-23 12:29:41

Tabel 5.2 Tabel *temperatureData*

Pada tabel *phData* terdapat 3 kolom, yaitu *phDataId*, *phWater*, dan *phDate*. Kolom *phDataId* dijadikan sebagai *primary key* tabel *phData* dengan tipe data *integer* dan dibuat *auto-increment*. Kolom *phWater* digunakan untuk menampung data pH air pada akuarium. Sedangkan kolom *phDate* digunakan untuk menampung data waktu (berupa tanggal, jam, menit, serta detik) ketika microcontroller mendapat data dari sensor pH dengan *interval* 15 menit yang di-*setting* dari program. Tabel *phData* dapat dilihat pada gambar 5.3 berikut.

phDataId ▲ 1	phWater	phDate
1	7.85	2019-11-23 10:14:41
2	7.72	2019-11-23 10:29:41
3	7.75	2019-11-23 10:44:41
4	7.89	2019-11-23 10:59:41
5	7.84	2019-11-23 11:14:41
6	7.71	2019-11-23 11:29:41
7	7.73	2019-11-23 11:44:41
8	7.79	2019-11-23 11:59:41
9	7.72	2019-11-23 12:14:41
10	7.91	2019-11-23 12:29:41

Tabel 5.3 Tabel *phData*

Pada tabel *loadcellData* terdapat 3 kolom, yaitu *loadcellDataId*, *bodyWeight*, dan *loadcellDate*. Kolom *phDataId* dijadikan sebagai *primary key* tabel *loadcellData* dengan tipe data *integer* dan dibuat *auto-increment*. Kolom *bodyWeight* digunakan untuk menampung data berat ikan. Sedangkan kolom *loadcellDate* digunakan untuk menampung data waktu (berupa tanggal, jam, menit, serta detik) ketika microcontroller mendapat data dari sensor *load cell* dengan *interval* 15 menit yang di-*setting* dari program. Tabel *loadcellData* dapat dilihat pada gambar 5.4 berikut.

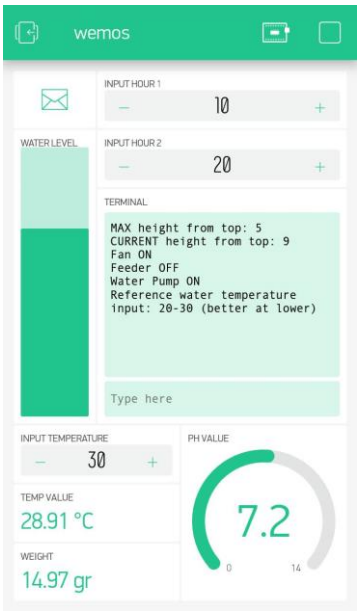
loadcellDataId	bodyWeight	loadcellDate
1	5.5	2019-11-23 10:14:41
2	5.56	2019-11-23 10:29:41
3	5.69	2019-11-23 10:44:41
4	5.61	2019-11-23 10:59:41
5	5.53	2019-11-23 11:14:41
6	5.56	2019-11-23 11:29:41
7	5.67	2019-11-23 11:44:41
8	5.7	2019-11-23 11:59:41
9	5.76	2019-11-23 12:14:41
10	5.72	2019-11-23 12:29:41

Tabel 5.4 Tabel *loadcellData*

5.2.2 Pengujian Aplikasi *Blynk*

Salah satu pengujian dari sistem adalah pada aplikasi *Blynk*. Pada aplikasi *Blynk*, terdapat beberapa macam *widget* yang dipakai yaitu *widget display*, *widget interface*, dan *widget notification*. Pada pengujian *widget display* yang dipakai yaitu *value display*, *terminal*, *gauge*, dan *level* V. Untuk *widget interface* yang dipakai yaitu *numeric input*. Sedangkan *widget notification* yang dipakai yaitu *widget email*.

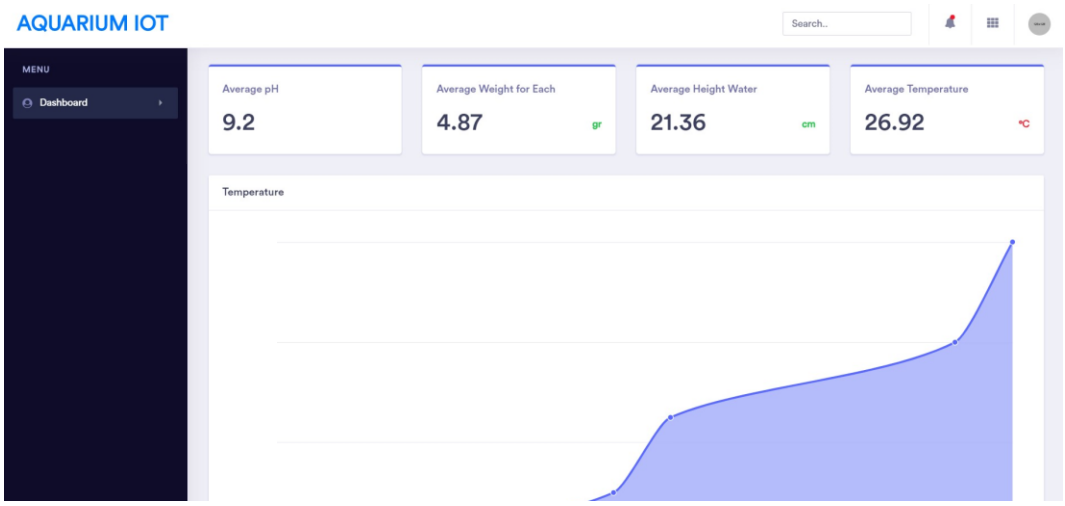
Widget numeric input digunakan untuk mengatur jam penjadwalan *feeder* yang digerakkan oleh *motor servo*. *Widget water level* digunakan untuk menampilkan tinggi air akuarium yang datanya didapat dari sensor *ultrasonic*. *Widget terminal* digunakan untuk menampilkan *text-text output* pada *aquator* (*feeder*, *cooler fan*, dan *water pump*) apakah dalam kondisi menyala atau mati. *Widget value display* digunakan untuk menampilkan data yang diterima dari sensor temperatur dan sensor *load cell*. Untuk *widget gauge* menampilkan data yang diterima dari sensor pH. Sedangkan *widget email* digunakan untuk mengirimkan notifikasi jika ada perubahan pada sistem. Untuk *user interface* pada aplikasi *Blynk* dapat dilihat pada gambar 5.1 berikut.



Gambar 5.1 *User Interface* pada Aplikasi *Blynk*

5.2.3 Pengujian *Website*

Data yang didapat dari sensor-sensor pada *aquarium* dimasukkan ke dalam *database*. Kemudian data-data yang berada pada *database* ditampilkan di *website*. *Website* dibuat menggunakan HTML dan PHP serta dilengkapi dengan *bootstrap*. *Website* yang dibuat berfungsi untuk menampilkan data dari *database* dan bukan untuk meng-*edit* data atau melakukan *input* data pada sistem. Data-data untuk setiap sensor yang terdapat pada *database* akan ditampilkan baik dalam bentuk angka ataupun grafik seperti yang dapat dilihat pada gambar 5.2 berikut.



Gambar 5.1 *User Interface* pada *Website*