LAMPIRAN

```
#define BLYNK_TEMPLATE_ID "TMPL6v_89-hNx"
#define BLYNK_TEMPLATE_NAME "ALat Pakan"
#define BLYNK_AUTH_TOKEN "Hd6kFjzT3Wa4rBvil9-Fi-yebcEB3HKP"

#define BLYNK_PRINT Serial

#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include <ESP32Servo.h>
#include <HX711.h>

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Wkwkwk";
char pass[] = "hahahahah";

const int servoPin = 2;
const int DOUT = 16;
const int CLK = 15;
const int trigPin1 = 5;
const int echoPin1 = 18;
const int trigPin2 = 4;
const int echoPin2 = 21;
const int trigPin3 = 14;
const int echoPin3 = 19;

Servo servo;
HX711 scale;
```

```
// Gunakan nilai kalibrasi yang benar berdasarkan kalibrasi sebelumnya
const float calibrationFactor = 3.000000;  // Sesuaikan dengan faktor kalibrasi Anda
const long offsetValue = -535;  // Sesuaikan dengan nilai offset Anda

void setup() {
  Serial.begin(115200);
  Blynk.begin(auth, ssid, pass);

  servo.attach(servoPin, 0, 3600);
  servo.write(0);

  scale.begin(DOUT, CLK);
  scale.set_offset(offsetValue); // Atur offset
  scale.set_scale(calibrationFactor); // Atur faktor kalibrasi
  scale.tare(); // Setel tare untuk menimbang dari nol

  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);
  pinMode(trigPin3, OUTPUT);
  pinMode(echoPin3, INPUT);
}

void loop() {
  Blynk.run();

  float ultrasonicDistance = getAverageDistance();
  Serial.print(ultraSonic1()); Serial.println("-1");
  Serial.print(ultraSonic2()); Serial.println("-2");
  Serial.print(ultraSonic3()); Serial.println("-3");
```

```
  Serial.print(ultrasonicDistance); Serial.println("-all/3");
  int feedStatus = getDistanceStatus(ultrasonicDistance);
  if(feedStatus == 1){
    Blynk.virtualWrite(V0, "Habis");
    Blynk.logEvent("pakanhabis");
  }else if(feedStatus == 2){
    Blynk.virtualWrite(V0, "Sedikit");
  }else if(feedStatus == 3){
    Blynk.virtualWrite(V0, "Cukup");
  }else if(feedStatus == 4){
    Blynk.virtualWrite(V0, "Penuh");
  }else{
    Blynk.virtualWrite(V0, "Error");
  }
  Blynk.virtualWrite(V1, "-");
  Blynk.virtualWrite(V3, 0);

  delay(1000);
}

BLYNK_WRITE(V2) {
  int value = param.asInt();
  Blynk.virtualWrite(V2, value);
  beriPakan(5000, 40);
  Blynk.virtualWrite(V2, 0);
}
BLYNK_WRITE(V4) {
  int value = param.asInt();
  Blynk.virtualWrite(V4, value);
  beriPakan(4000, 30);
  Blynk.virtualWrite(V4, 0);
```

```
}

void beriPakan(unsigned long delays, int pakan) {
  float ultrasonicDistance = getAverageDistance();
  int feedStatus = getDistanceStatus(ultrasonicDistance);
  float berat = scale.get_units(10); // Ambil rata-rata dari 10 pembacaan

  Blynk.virtualWrite(V3, berat);

  if (feedStatus == 1) {
    Blynk.virtualWrite(V1, "Pakan Kurang");
    Blynk.logEvent("gagalberipakan");
  } else {
    for (int pos = 0; pos <= 180; pos++) {
      servo.write(pos);
      delay(15);
    }
    Blynk.virtualWrite(V1, "Takaran Sedang Diisi");

    delay(delays);
    Blynk.virtualWrite(V3, pakan);

    Blynk.virtualWrite(V1, "Takaran Telah Diisi");

    for (int pos = 180; pos >= 0; pos--) {
      servo.write(pos);
      delay(15);
    }
    Blynk.virtualWrite(V1, "Pakan Telah Diberikan");

    if(pakan == 30){
```

```
      Blynk.logEvent("beripakan30");
    }else if(pakan == 40){
      Blynk.logEvent("beripakan40");
    }
  }
}

float getDistance(int trigPin, int echoPin) {
  long sum = 0;
  const int numMeasurements = 10; // Reduced number of measurements for faster
response
  for (int i = 0; i < numMeasurements; i++) {
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    long duration = pulseIn(echoPin, HIGH);
    float distance = duration * 0.034 / 2;
    sum += distance;
    delay(5);  // Delay sedikit untuk memberikan waktu sensor beristirahat
  }
  return sum / numMeasurements;
}

float ultraSonic1() {
  return getDistance(trigPin1, echoPin1);
}
float ultraSonic2() {
  return getDistance(trigPin2, echoPin2);
}
```

```
float ultraSonic3() {
  return getDistance(trigPin3, echoPin3);
}

float getAverageDistance() {
  float distance1 = ultraSonic1();
  float distance2 = ultraSonic2();
  float distance3 = ultraSonic3();
  return (distance1 + distance2 + distance3) / 3;
}

int getDistanceStatus(float distance) {
  if (distance <= 4) {
    return 4; // penuh
  } else if (distance <= 6) {
    return 3; // cukup
  } else if (distance <= 9) {
    return 2; // sedikit
  } else if (distance <= 20) {
    return 1; // habis
  } else {
    return 0; // error
  }
}
```