**LAPORAN TUGAS PROYEK**

**SISTEM PENGATURAN TERTANAM GASAL 2022/2023**

**“*Shooting Point Camera Stabillizer* pada Gimbal *2-Axis*”**



**Oleh :**

**Hakhi Gya Yektianto 07111940000022**

**Faiz Adhima Fahruddin 07111940000077**

**Departemen Teknik Elektro**

**Fakultas Teknologi Elektro dan Informatika Cerdas**

**Institut Teknologi Sepuluh Nopember Surabaya**

**2022**

## **BAB 1 : PENDAHULUAN**

### **Latar Belakang**

Dewasa ini, kamera merupakan suatu komponen/peralatan elektrik yang sangat menjamur di kalangan masyarakat, terutama masyarakat Indonesia. Berbagai teknologi terapan yang berbasis pada kamera juga menunjang berbagai aktivitas manusia, seperti halnya fotografi, videografi, bahkan hingga industri per-film-an. Perkembangan paham dan ilmu fotografi maupun videografi juga menjadi salah satu tuntutan tersendiri bagi proses pengembangan teknologi pengambilan gambar maupun video. Perkembangan yang cepat inilah menuntut agar kualitas kamera yang digunakan selalu prima. Kualitas kamera sendiri secara umum bergantung kepada komponen kamera itu sendiri dan juga komponen pendukung. Perangkat pendukung yang sering digunakan untuk mengoptimalkan fungsi kamera adalah gimbal. Gimbal merupakan alat pengendali gerakan kamera yang dapat menggerakkan kamera pada tiga sumbu yaitu pada sumbu x, y, dan z. Gimbal sendiri menjadi salah satu komponen penting yang dapat memengaruhi kinerja maupun hasil akhir yang diberikan oleh sebuah kamera. Penggunaan gimbal diharapkan dapat meningkatkan kinerja dari suatu *device* pengambil gambar, sehingga data yang dihasilkan baik dan memiliki tingkat *noise* yang minimum.

### **Rumusan Masalah**

Beberapa rumusan masalah yang dapat diformulakan adalah sebagai berikut :

1. Bagaimana merancang suatu sistem penyeimbang kamera dengan gimbal 2-axis untuk mendapatkan hasil tangkapan yang baik ?
2. Bagaimana merancang suatu integrasi sistem yang dapat memudahkan pengguna untuk mengakses maupun melakukan perubahan terhadap gimbal?

### **Tujuan**

Adapun beberapa tujuan dari tugas proyek ini adalah sebagai berikut :

1. Merancang suatu sistem penyeimbang kamera dengan gimbal 2-axis untuk mengoptimalkan fungsi pengambilan gambar pada kamera.
2. Merancang suatu bentuk pengintegrasian agar memudahkan pengguna untuk mengakses maupun melakukan perubahan terhadap gimbal?

## **BAB 2 : DASAR TEORI**

### ***Camera Gimbal* 2-Axis**

Camera gimbal adalah alat penstabil kamera, yang berfungsi untuk menghasilkan gambar atau video tanpa terpengaruh oleh kemiringan atau guncangan yang terjadi ketika pengambilan gambar atau video berlangsung. Memanfaatkan kerja gimbal yang dapat bergerak bebas pada porosnya mengakibatkan camera gimbal dapat menyesuaikan orientasi pergerakan ketika terjadi perubahan, sehingga kamera yang ditempatkan pada camera gimbal akan tetap stabil pada tempatnya.

*Gimbal* 2-sumbu terdiri dari dua buah *joints* yang dapat berputar pada sudut θ1 dan θ2 sepanjang sumbu *pitch* dan sumbu *roll*. *Gimbal* 2-sumbu yang digunakan terdiri dari tiga *body frame* (*body* (0), *body* (1), dan *body* (2)). Pada Gambar 2 ditunjukkan hubungan antar *body frame. Body* (0), *body* (1), dan *body* (2) dalam bentuk X0Y0Z0, X1Y1Z1, dan X2Y2Z2.

Chart, waterfall chart

Description automatically generated

**Gambar 2. 1** Camera Gimbal 2-Axis

### **ESP-8266**

ESP-8266 adalah mikrokontroler yang terintegrasi dengan modul Wi-Fi dan Bluetooth onboard. Projek berbasis IoT umumnya menggunakan Mikrokontroller ESP32 karena memiliki modul wi-Fi yang sudah terintegrasi onboard sehingga tidak . Selain itu ESP32 juga sudah terintegrasi dengan *built-in antenna switches*, *RF balun*, *power amplifier*, *low-noise receive amplifier*, *filters*, dan *power management modules*. sehingga sangat mendukung untuk pengaplikasian *Internet of Things*. ESP32 mempunyai memori RAM sebesar 320 KB dan ROM sebesar 448 KB. ESP32 memiliki *peripheral Interface* antara lain 34 pin GPIO (*General Purpose Input/Output*), 18 pin ADC (*Analog Digital Converter*), 2 pin DAC (*Digital Analog Converter*), 16 pin PWM (*Pulse Width Modulation*), 10 pin *Capasitive Sensing*, 2 jalur antarmuka UART, pin *interface* I2C, I2S, SPI, dll. Setiap pinout ESP 32 dapat menerima atau memberi tegangan hingga sebesar 3,3V (Espressif Systems, 2021).

### **Sensor IMU MPU-6050**

Sensor MPU-6050 merupakan sensor IMU 6 derajat kebebasan yang terdiri dari accelerometer 3 derajat kebebasan, gyroscope 3 derajat kebebasan dan sebuah prosesor gerakan digital (DMP). Modul ini menggunakan protokol I2C untuk berkomunikasi dengan mikrokontroler. Gyroscope MPU-6000 dapat diprogram untuk bekerja pada skala 250, 500, 1000 dan 2000 º/detik, untuk accelerometer dapat diprogram pada skala 2G, 4G, 8G, dan 16G. Data dari sensor konversi DMP dengan 16 bit ADC(Vamiko et al., 2013).

### **Motor Servo**

Motor servo adalah sebuah motor dengan sistem closed loop yang digunakan untuk mengendalikan kecepatan, akselerasi dan posisi akhir dari sebuah motor listrik dengan keakuratan yang tinggi. Umumnya, motor servo banyak digunakan sebagai aktuator yang membutuhkan posisi putaran motor yang presisi. Motor servo terdiri dari tiga bagian utama, yaitu: motor, sistem kontrol dan potensiometer yang terhubung dengan satu set internal gearbox. Motor servo memiliki putaran yang lambat, namun memiliki torsi yang kuat karena adanya internal gear.

Diagram

Description automatically generated

**Gambar 2. 2** Motor Servo

Motor servo dikendalikan dengan sinyal PWM dari potentiometer. Potentiometer berfungsi sebagai sensor yang memberikan sinyal umpan balik (feedback) untuk megkoreksi posisi target. Sedangkan sudut dari sumbu motor servo diatur berdasarkan lebar pulsa yang dikirim melalui kaki sinyal dari kabel motor.Potentiometer ini terdiri dari tiga kabel dengan 2 kabel untuk power dan 1 kabel untuk kabel sinyal. (Hilal & Manan, 2015)

## **BAB 3 : PENGUJIAN SISTEM**

### **Kalibrasi awal**

Pada sensor diperlukan kalibrasi nilai awal offset untuk agar pembacaan sensor akurat sebelum dimasukkan ke dalam algoritma perhitungan. Kalibrasi awal menggunakan library dari “MPU6050\_6Axis\_MotionApps20”. MPU 6050 Berikut merupakan kode untuk kalibrasi.

if (devStatus == 0) {

      // Calibration Time: generate offsets and calibrate our MPU6050

      mpu.CalibrateAccel(7);  //Kalibrasi Pertama kali

      mpu.CalibrateGyro(7);

      mpu.PrintActiveOffsets();

      mpu.setDMPEnabled(true);

      attachInterrupt(digitalPinToInterrupt(INTERRUPT\_PIN), dmpDataReady, RISING);

      mpuIntStatus = mpu.getIntStatus();

      dmpReady = true;

      packetSize = mpu.dmpGetFIFOPacketSize();

    }

    else{

      Serial.print(F("DMP Initialization failed (code "));

      Serial.print(devStatus);

      Serial.println(F(")"));

    }

}

nilai hasil kalibrasi

Text

Description automatically generated

Setelah memperoleh nilai hasil kalibrasi nilai tersebut dimasukkan ke dalam penentuan offset

    pinMode(INTERRUPT\_PIN, INPUT);

   mpu.initialize();

  devStatus = mpu.dmpInitialize();

    pinMode(INTERRUPT\_PIN, INPUT);

    // nilai offset hasil kalibrasi

    mpu.setXGyroOffset(45);

    mpu.setYGyroOffset(-27);

    mpu.setZGyroOffset(18);

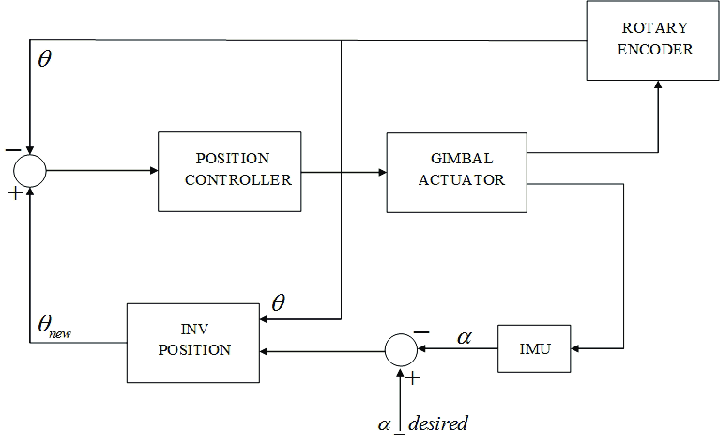
    mpu.setXAccelOffset(-1690);

    mpu.setYAccelOffset(1157);

    mpu.setZAccelOffset(2742);

### **Algoritma kontrol feedback**

Algoritma yang digunakan untuk kontrol servo pada projek ini adalah close loop sederhana seperti pada gambar berikut.



Sensor gyro (IMU) hanya dapat membaca nilai percepatan perubahan sudut dan percepatan dari gerak translasi dengan rentang -2G sampai +2G .yang man Adapun untuk algoritma perhitungan sudut yaw, pitch, roll menggunakan library “MPU6050\_6Axis\_MotionApps20”. Berikut merupakan pengaturan output yaw, pitch, roll

#include "I2Cdev.h"

#include "Servo.h"

#include "MPU6050\_6Axis\_MotionApps20.h"

#if I2CDEV\_IMPLEMENTATION == I2CDEV\_ARDUINO\_WIRE

    #include "Wire.h"

#endif

// MPU control/status vars

bool dmpReady = false;  // set true if DMP init was successful

uint8\_t mpuIntStatus;   // holds actual interrupt status byte from MPU

uint8\_t devStatus;      // return status after each device operation (0 = success, !0 = error)

uint16\_t packetSize;    // expected DMP packet size (default is 42 bytes)

uint16\_t fifoCount;     // count of all bytes currently in FIFO

uint8\_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars

Quaternion q;           // [w, x, y, z]         quaternion container

VectorFloat gravity;    // [x, y, z]            gravity vector

float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector

// ================================================================

// ===               INTERRUPT DETECTION ROUTINE                ===

// ================================================================

volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt pin has gone high

void ICACHE\_RAM\_ATTR dmpDataReady() {

    mpuInterrupt = true;

}

Setelah dapat memperoleh nilai hasil pembacaan nilai sudut roll, pitch, yaw. Niali tersebut perlu difilter dengan menggunakan low pass filter. Setelah memperoleh nilai sudut sudut tersebut nantinya akan digunakan sebagai koreksi nilai untuk menggerakkkan serve

void loop() {

  if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) { // Get the Latest packet

    // display Euler angles in degrees

    mpu.dmpGetQuaternion(&q, fifoBuffer);

    mpu.dmpGetGravity(&gravity, &q);

    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

    if(statusKalibrasi == 0){  //proses kalibrasi

      dataYaw = ypr[0] \* 180/M\_PI;

      dataPitch= ypr[1] \* 180/M\_PI;

      dataRoll= ypr[2] \* 180/M\_PI;

      tSekarang = millis();

      dT = tSekarang - tSebelum;

      if(dT < WAKTU\_CEK){

        yawSekarang=ypr[0];

        pitchSekarang=ypr[1];

        rollSekarang=ypr[2];

        }

      else{

        dYaw = abs(yawSekarang - yawSebelum);

        yawSebelum = yawSekarang;

        dRoll = abs(rollSekarang - rollSebelum);

        rollSebelum = rollSekarang;

        dPitch = abs(pitchSekarang - pitchSebelum);

        pitchSebelum = pitchSekarang;

        if(dYaw<ERROR\_MINIMAL || dRoll<ERROR\_MINIMAL || dPitch<ERROR\_MINIMAL ){

          counterKalibrasi++ ;

          }

        else counterKalibrasi = 0;

        if (counterKalibrasi>6){

            statusKalibrasi = 1;

            offsetSudutYaw = yawSekarang;

            offsetSudutPitch = pitchSekarang;

            offsetSudutRoll = rollSekarang;

          }

        tSebelum = tSekarang;

      }

    }

    else{

      dataYaw = (ypr[0]-offsetSudutYaw) \* 180/M\_PI;

      dataPitch = (ypr[1]-offsetSudutPitch) \* 180/M\_PI;

      dataRoll = (ypr[2]-offsetSudutRoll) \* 180/M\_PI;

      servo1.write(dataRoll);

      servo2.write(dataPitch);

    }

    // kirim data setiap 5 detik

    if ((millis()-last\_millis)>5000){

      last\_millis=millis();

      String protokol = "/update.php?sudutroll="+String(dataRoll)+

      "&sudutpitch="+String(dataPitch);

      Serial.println( "Send Data To Server: Data Roll = "+String(dataRoll)+

      " Data Pitch = "+String(dataPitch));

  }

    }

}

### **Pembuatan IoT**

#### **Pembuatan Protokol komunikasi**

Inisisasi program koneksi internet dan koneksi host

// bagian IoT

#include <ESP8266WiFi.h>

const char\* host = "gimbal.galium.ga";

const int httpPort = 80;

const char\* ssid     = "apoci";

const char\* password = "apoci123";

long last\_millis\_wifi;

String wifi="OFF";

long last\_millis;

WiFiClient client;

Pembuatan protokol komunikasi server dengan alat

 // kirim data setiap 5 detik

    if ((millis()-last\_millis)>5000){

      last\_millis=millis();

      String protokol = "/update.php?sudutroll="+String(dataRoll)+

      "&sudutpitch="+String(dataPitch);

      Serial.println( "Send Data To Server: Data Roll = "+String(dataRoll)+

      " Data Pitch = "+String(dataPitch));

  }

Protokol komunikasi kirim nilai pembacaan dan mengambil nilai set point dari website

String send(String url) {

  if (!client.connect(host, httpPort)) {

    return "Error Request Confirm";

  }

  client.print(String("GET ") + url + " HTTP/1.1\r\n" +

               "Host: " + host + "\r\n" +

               "User-Agent: BuildFailureDetectorESP8266\r\n" +

               "Connection: close\r\n\r\n");

  String body = "";

   long last\_millis\_x=millis();

  long next\_millis\_x=millis();

  while (client.connected()) {

      if ((millis()-last\_millis\_x)>10000){

        goto next3;

      }

      next\_millis\_x=millis();

    while (client.available()) {

       if ((millis()-next\_millis\_x)>10000){

        goto next3;

      }

      char c = client.read();

      body += String(c);

    }

  }

  next3:

  client.stop();

  int a = body.indexOf("~");

  int b = body.indexOf("!");

  int c = body.indexOf("@");

  String data\_dataRoll =  body.substring(a+1,b);

  String data\_dataPitch =  body.substring(b+1,c);

  if (data\_dataRoll!=""&&data\_dataPitch!=""){

      //Serial.println(body);

      Serial.println("SP="+data\_dataRoll+" "+data\_dataPitch);

      dataRoll = data\_dataRoll.toInt();

      dataPitch = data\_dataPitch.toInt();

      body="OK";

  }else{

    body ="ERROR";

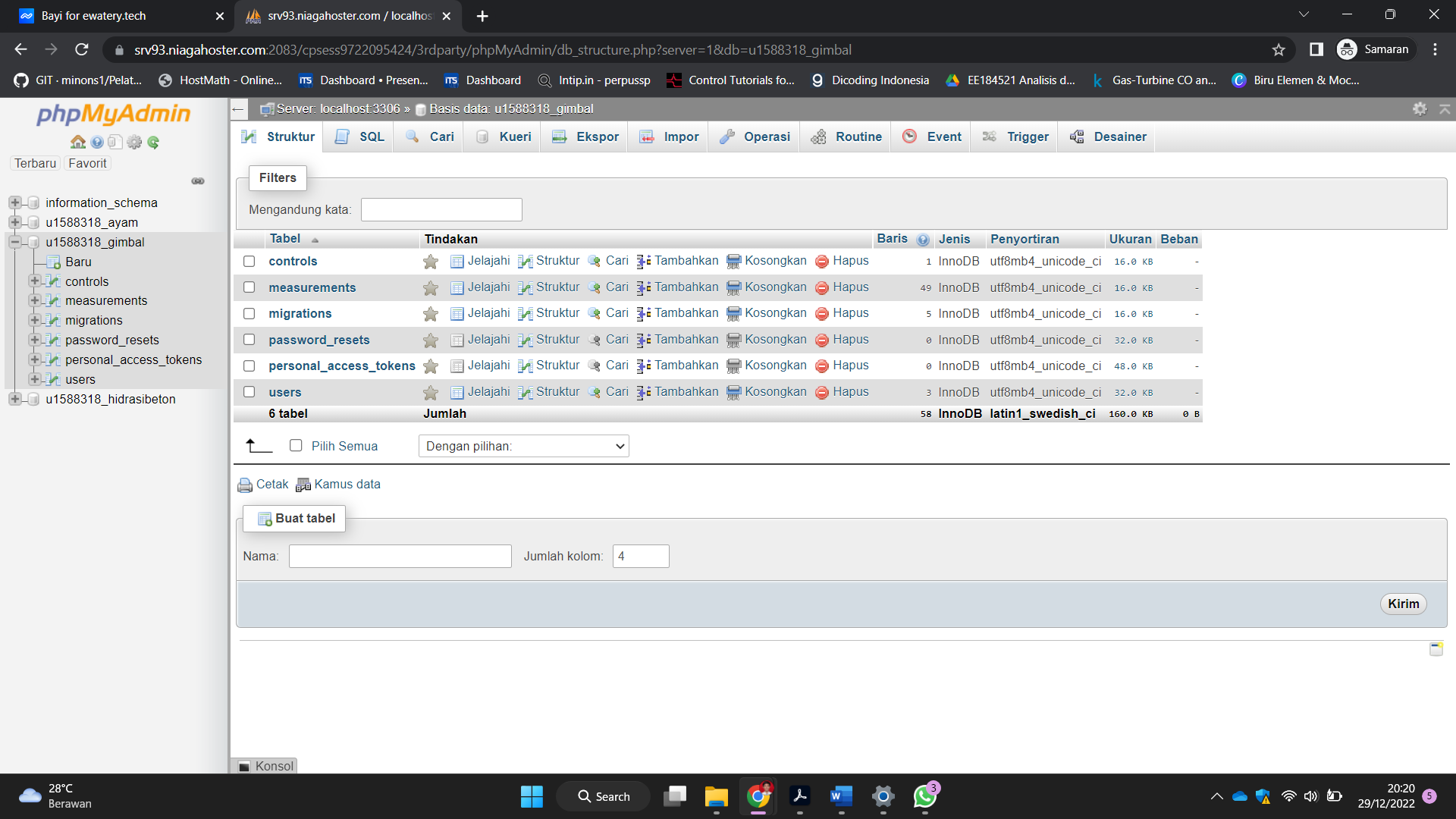
  }

  return body;

}

#### **Pembuatan Database MySQL**

MySQL adalah open souce Relational database management system yang sering digunakan untuk menyimpan data secara online (Open Source). MySQL digunakan dalam beberapa kasus, seperti pada automated system yang digunakan untuk mendapatkan data dan di input secara live. MySQL dapat disesuaikan dengan besar skala produk dan dapat melakukan penyesuaian sesuai dengan besar dari produk.



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

#### **Hasil pengujian IoT**

Hasil pengiriman data pada serial monitor

Text

Description automatically generated

Hasil tampilan pengiriman data pada website

A screenshot of a computer

Description automatically generated

# Lampiran

* Akses webiste

Link : [https://gimbal.galium.ga/](https://gimbal.galium.ga/%20)

User: spt2022

Pass: spt2022

* Code mikrokontroller

#include "I2Cdev.h"

#include "Servo.h"

#include "MPU6050\_6Axis\_MotionApps20.h"

#if I2CDEV\_IMPLEMENTATION == I2CDEV\_ARDUINO\_WIRE

    #include "Wire.h"

#endif

// bagian IoT

#include <ESP8266WiFi.h>

const char\* host = "gimbal.galium.ga";

const int httpPort = 80;

const char\* ssid     = "apoci";

const char\* password = "apoci123";

long last\_millis\_wifi;

String wifi="OFF";

long last\_millis;

WiFiClient client;

//bagian kontrol

MPU6050 mpu;

Servo servo1; //servo bagian bawah (roll)

Servo servo2; //servo bagian atas (pitch)

//keperluan kalibrasi

#define ERROR\_MINIMAL 0.001

#define WAKTU\_CEK 1000 //dalam milisekon

long int tSekarang;

long int tSebelum;

long int dT;

double yawSekarang;

double yawSebelum;

double dYaw;

double pitchSekarang;

double pitchSebelum;

double dPitch;

double rollSekarang;

double rollSebelum;

double dRoll;

double bufferDataKalibrasi[10];

char counterKalibrasi;

double dataYaw;

double dataPitch;

double dataRoll;

char dataYawKirim[4];

char statusKalibrasi = 0;

double offsetSudutYaw;

double offsetSudutPitch;

double offsetSudutRoll;

#define INTERRUPT\_PIN D8

// MPU control/status vars

bool dmpReady = false;  // set true if DMP init was successful

uint8\_t mpuIntStatus;   // holds actual interrupt status byte from MPU

uint8\_t devStatus;      // return status after each device operation (0 = success, !0 = error)

uint16\_t packetSize;    // expected DMP packet size (default is 42 bytes)

uint16\_t fifoCount;     // count of all bytes currently in FIFO

uint8\_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars

Quaternion q;           // [w, x, y, z]         quaternion container

VectorFloat gravity;    // [x, y, z]            gravity vector

float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll and gravity vector

// ================================================================

// ===               INTERRUPT DETECTION ROUTINE                ===

// ================================================================

volatile bool mpuInterrupt = false;

void ICACHE\_RAM\_ATTR dmpDataReady() {

    mpuInterrupt = true;

}

// ================================================================

// ===                      INITIAL SETUP                       ===

// ================================================================

void setup() {

    // join I2C bus (I2Cdev library doesn't do this automatically)

    #if I2CDEV\_IMPLEMENTATION == I2CDEV\_ARDUINO\_WIRE

        Wire.begin();

        Wire.setClock(400000);

    #elif I2CDEV\_IMPLEMENTATION == I2CDEV\_BUILTIN\_FASTWIRE

        Fastwire::setup(400, true);

    #endif

    Serial.begin(115200);

    Serial.print("void setup initialized");

    servo1.attach(D6);

    servo2.attach(D5);

    servo1.write(0);          //ATUR POSISI AWAL SERVO

    servo2.write(0);

    delay(1000);

    // konek wifi

  WiFi.mode(WIFI\_STA);

  WiFi.begin(ssid, password);

  Serial.print("Connecting.....");

  Serial.print(ssid);

  Serial.print("SUCCESS!");

  Serial.println("");

  Serial.println("WiFi connected");

  Serial.println("IP address: ");

  Serial.println(WiFi.localIP());

    mpu.initialize();

    pinMode(INTERRUPT\_PIN, INPUT);

    devStatus = mpu.dmpInitialize();

    pinMode(INTERRUPT\_PIN, INPUT);

    // nilai offset hasil kalibrasi

    mpu.setXGyroOffset(45);

    mpu.setYGyroOffset(-27);

    mpu.setZGyroOffset(18);

    mpu.setXAccelOffset(-1690);

    mpu.setYAccelOffset(1157);

    mpu.setZAccelOffset(2742);

    if (devStatus == 0) {

      // Calibration Time: generate offsets and calibrate our MPU6050

      mpu.CalibrateAccel(7);  //Kalibrasi Pertama kali

      mpu.CalibrateGyro(7);

      mpu.PrintActiveOffsets();

      mpu.setDMPEnabled(true);

      attachInterrupt(digitalPinToInterrupt(INTERRUPT\_PIN), dmpDataReady, RISING);

      mpuIntStatus = mpu.getIntStatus();

      dmpReady = true;

      packetSize = mpu.dmpGetFIFOPacketSize();

    }

    else{

      Serial.print(F("DMP Initialization failed (code "));

      Serial.print(devStatus);

      Serial.println(F(")"));

    }

}

// untuk kalibrasi awal

// void loop(){

// }

// ================================================================

// ===                    MAIN PROGRAM LOOP                     ===

// ================================================================

void loop() {

  if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) { // Get the Latest packet

    // display Euler angles in degrees

    mpu.dmpGetQuaternion(&q, fifoBuffer);

    mpu.dmpGetGravity(&gravity, &q);

    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

    if(statusKalibrasi == 0){  //proses kalibrasi

      dataYaw = ypr[0] \* 180/M\_PI;

      dataPitch= ypr[1] \* 180/M\_PI;

      dataRoll= ypr[2] \* 180/M\_PI;

      tSekarang = millis();

      dT = tSekarang - tSebelum;

      if(dT < WAKTU\_CEK){

        yawSekarang=ypr[0];

        pitchSekarang=ypr[1];

        rollSekarang=ypr[2];

        }

      else{

        dYaw = abs(yawSekarang - yawSebelum);

        yawSebelum = yawSekarang;

        dRoll = abs(rollSekarang - rollSebelum);

        rollSebelum = rollSekarang;

        dPitch = abs(pitchSekarang - pitchSebelum);

        pitchSebelum = pitchSekarang;

        if(dYaw<ERROR\_MINIMAL || dRoll<ERROR\_MINIMAL || dPitch<ERROR\_MINIMAL ){

          counterKalibrasi++ ;

          }

        else counterKalibrasi = 0;

        if (counterKalibrasi>6){

            statusKalibrasi = 1;

            offsetSudutYaw = yawSekarang;

            offsetSudutPitch = pitchSekarang;

            offsetSudutRoll = rollSekarang;

          }

        tSebelum = tSekarang;

      }

    }

    else{

      dataYaw = (ypr[0]-offsetSudutYaw) \* 180/M\_PI;

      dataPitch = (ypr[1]-offsetSudutPitch) \* 180/M\_PI;

      dataRoll = (ypr[2]-offsetSudutRoll) \* 180/M\_PI;

      // int servo1Value = map(dataRoll, -90, 90, 0, 180);

      // int servo2Value = map(dataPitch, -90, 90, 180, 0);

      // Serial.print("\t");

      // Serial.print(servo1Value);

      // Serial.print("\t");

      // Serial.print(servo2Value);

      // Serial.print("\t");

      // Serial.print(dataYaw);

      // Serial.print("\t");

      // Serial.print(dataPitch);

      // Serial.print("\t");

      // Serial.println(dataRoll);

      servo1.write(dataRoll);

      servo2.write(dataPitch);

    }

    // kirim data setiap 5 detik

    if ((millis()-last\_millis)>5000){

      last\_millis=millis();

      String protokol = "/update.php?sudutroll="+String(dataRoll)+

      "&sudutpitch="+String(dataPitch);

      Serial.println( "Send Data To Server: Data Roll = "+String(dataRoll)+

      " Data Pitch = "+String(dataPitch));

  }

    }

}

String send(String url) {

  if (!client.connect(host, httpPort)) {

    return "Error Request Confirm";

  }

  client.print(String("GET ") + url + " HTTP/1.1\r\n" +

               "Host: " + host + "\r\n" +

               "User-Agent: BuildFailureDetectorESP8266\r\n" +

               "Connection: close\r\n\r\n");

  String body = "";

   long last\_millis\_x=millis();

  long next\_millis\_x=millis();

  while (client.connected()) {

      if ((millis()-last\_millis\_x)>10000){

        goto next3;

      }

      next\_millis\_x=millis();

    while (client.available()) {

       if ((millis()-next\_millis\_x)>10000){

        goto next3;

      }

      char c = client.read();

      body += String(c);

    }

  }

  next3:

  client.stop();

  int a = body.indexOf("~");

  int b = body.indexOf("!");

  int c = body.indexOf("@");

  String data\_dataRoll =  body.substring(a+1,b);

  String data\_dataPitch =  body.substring(b+1,c);

  if (data\_dataRoll!=""&&data\_dataPitch!=""){

      //Serial.println(body);

      Serial.println("SP="+data\_dataRoll+" "+data\_dataPitch);

      dataRoll = data\_dataRoll.toInt();

      dataPitch = data\_dataPitch.toInt();

      body="OK";

  }else{

    body ="ERROR";

  }

  return body;

}