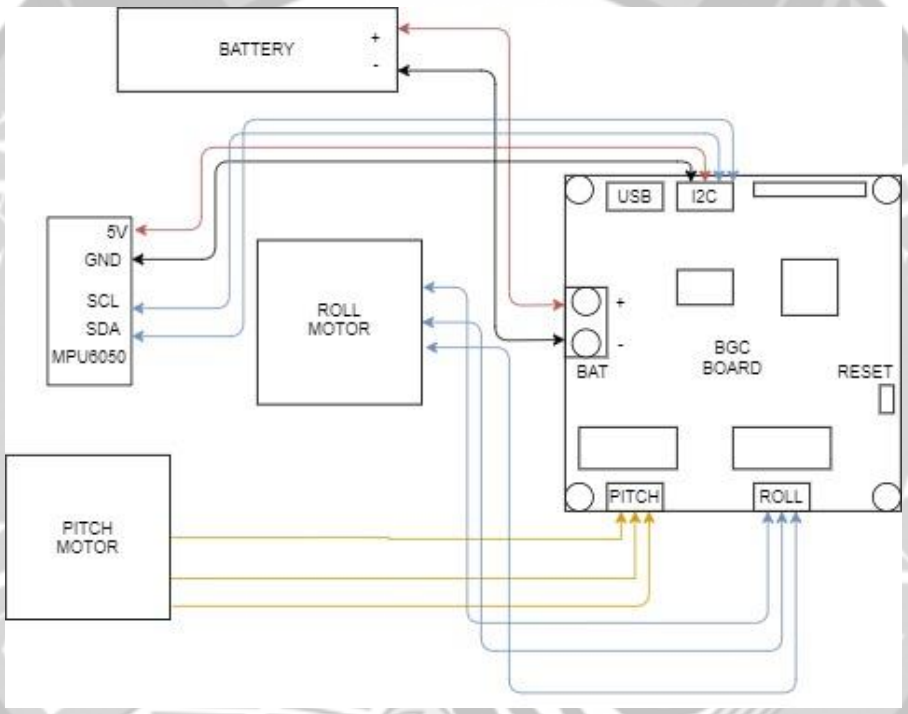
BAB IV

PERANCANGAN SISTEM GIMBAL 2-AXIS

Pada bab ini dijelaskan perancangan sistem gimbal dengan menggunakan sensor MPU6050, *brushless* motor dan BGC *board* terhadap pergerakan pengguna. Skema perancangan dapat dilihat pada Gambar 4.1

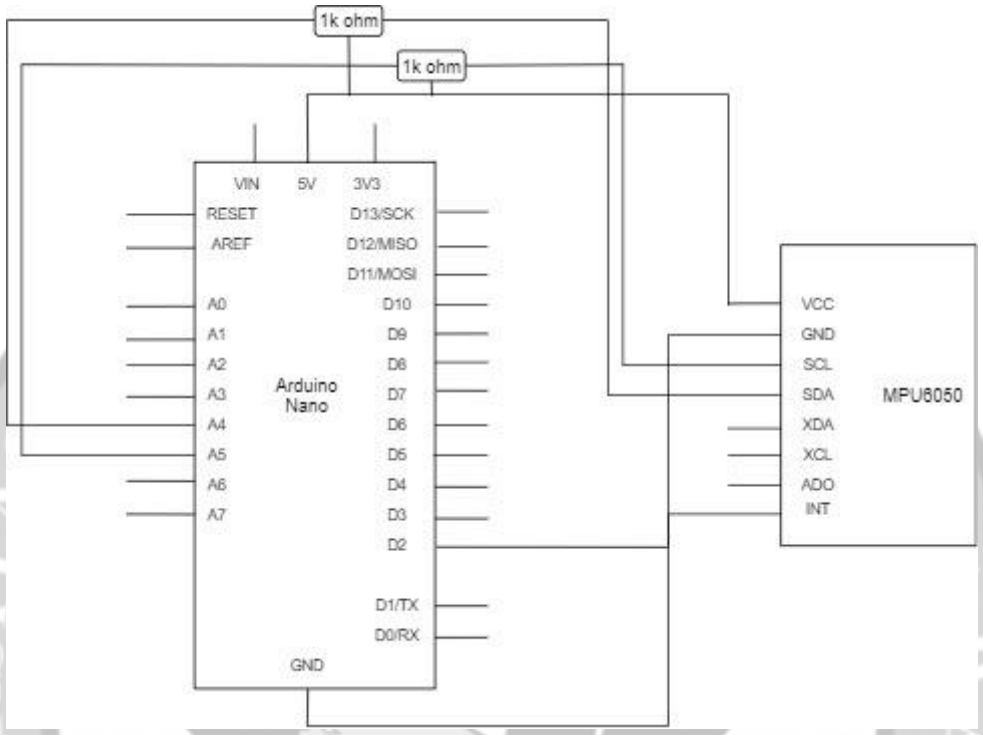


Gambar 4.1 Skema sistem gimbal

4.1 Perancangan sistem sensor MPU6050

Pada perancangan ini sensor MPU6050 tidak bisa kerja dengan sendirinya. Karena itu diperlukan Arduino Nano untuk berkerja sebagai *microcontroller*. Desain perancangan ini bisa dilihat pada Gambar 4.2. Dengan menggunakan sumber daya

sebesar 5V dan input dari sensor ke *microcontroller* untuk menjalankan *gyroscope* dan *accelerometer*nya

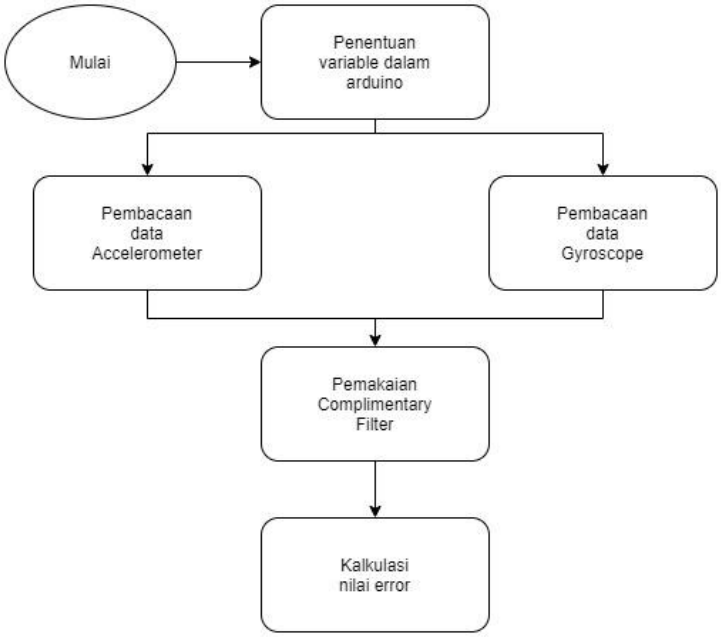


Gambar 4.2 Desain koneksi antara sensor dengan *microcontroller*

Untuk sensor MPU6050 sudah cukup dan bisa berjalan sendiri dengan *coding* dan *setting* rangkaian gimbal pada Gambar 4.1. Tetapi khusus Gambar 4.2 diperlihatkan dan diperlukan untuk akuisisi data untuk pengukuran pada sistem rangkaian Gambar 4.1. Kunci pada pengukuran alat gimbal tersebut adalah sensor MPU6050 dengan pengambilan data output *accelerometer* dan *gyroscope*.

4.2 **Coding pada microcontroller**

Struktur kerja *coding* untuk *microcontroller* bisa dilihat pada Gambar 4.3. Pertama-tama ditentukan variabel untuk pembacaan data *accelerometer* dan *gyroscope*. Setelah itu dilanjutkan dengan menggunakan *complementary filter* dan penentuan *output error* dari *coding*nya.



Gambar 4.3 *Flowchart coding* dalam *microcontroller*

4.2.1 Penentuan Variabel dalam Arduino

Pertama – tama ditentukan dulu variabel dalam *coding* yaitu variabel untuk data *accelerometer* dan data *gyroscope*. Variabel yang ditentukan bisa dilihat pada Gambar 4.4 tersebut.

```
#include <Wire.h>

const int MPU = 0x68; // MPU6050 I2C address
float AccX, AccY, AccZ;
float GyroX, GyroY, GyroZ;
float accAngleX, accAngleY, gyroAngleX, gyroAngleY, gyroAngleZ;
float roll, pitch, yaw;
float AccErrorX, AccErrorY, GyroErrorX, GyroErrorY, GyroErrorZ;
float elapsedTime, currentTime, previousTime;
int c = 0;
```

Gambar 4.4 *Coding* Arduino untuk deklarasi variabel

4.2.2 Pembacaan data *accelerometer*

Seperti pada datasheet MPU6050[4], untuk mendapatkan data yang efektif yang dipakai dalam sensor adalah dengan menggunakan pembacaan setiap axis. Langkah selanjutnya adalah kalkulasi *roll* dan *pitch* dari data *accelerometer*. *Coding* dan rumus untuk menentukan *roll* dan *pitch accelerometer* dapat dilihat pada Gambar 4.5.

```
//read accelerometer data//
Wire.beginTransmission(MPU);
Wire.write(0x3B);
Wire.endTransmission(false);
Wire.requestFrom(MPU, 6, true);

AccX = (Wire.read() << 8 | Wire.read()) / 16384.0; // X axis value
AccY = (Wire.read() << 8 | Wire.read()) / 16384.0; // Y axis value
AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0; // Z axis value
//Calculate Roll and Pitch from the accelerometer data
accAngleX = (atan(AccY / sqrt(pow(AccX, 2) + pow(AccZ, 2))) * 180 / PI) - 0.58;
accAngleY = (atan(-1 * AccX / sqrt(pow(AccY, 2) + pow(AccZ,2))) * 180 / PI) + 1.58;
```

Gambar 4.5 *Coding* data *accelerometer*

4.2.3 Pembacaan data *gyroscope*

Setelah pembacaan data *accelerometer*, selanjutnya menentukan data *gyroscope*. Codingan data gyroscope bisa dilihat pada Gambar 4.6

```
//read gyroscope data//
previousTime = currentTime;
currentTime = millis();
elapsedTime = (currentTime - previousTime) / 1000;
Wire.beginTransmission(MPU);
Wire.write(0x43); //gyro data 1st register address 0x43
Wire.endTransmission(false);
Wire.requestFrom(MPU, 6, true);
GyroX = (Wire.read() << 8 | Wire.read()) / 131.0; // for a 250 degree/s range
GyroY = (Wire.read() << 8 | Wire.read()) / 131.0;
GyroZ = (Wire.read() << 8 | Wire.read()) / 131.0;
```

Gambar 4.6 Coding data *gyroscope*

Output yang telah didapat dari data *accelerometer* dan *gyroscope* adalah *degree/second*. Maka dengan itu *output* tersebut dikalikan dengan waktu untuk mendapatkan satuan derajat yang bisa dilihat dalam Gambar 4.7.

```
// currently the raw values are in deg/s so we need to make it just a deg value
gyroAngleX = gyroAngleX + GyroX * elapsedTime; // deg/s * s = deg
gyroAngleY = gyroAngleY + GyroY * elapsedTime;
yaw = yaw + GyroZ * elapsedTime;
```

Gambar 4.7 *Coding output* data *Gyroscope*

4.2.4 Pemakaian *complementary filter*

Dari *datasheet* MPU6050[4] telah ditentukan bahwa untuk pemakaian sensor *gyroscope* itu sebesar 96%. Tetapi *gyroscope* memiliki error, maka dengan itu data *accelerometer* bisa membantu sebesar 4% untuk mengeliminasi error *gyroscope*. Rumus *complementary filter* bisa dilihat pada Gambar 4.8

```
//complementary Filter = combine accelerometer and gyro angle values
roll = 0.96 * gyroAngleX + 0.04 * accAngleX;
pitch = 0.96 * gyroAngleY + 0.04 * accAngleY;
```

Gambar 4.8 *Coding complementary filter*

4.2.5 Penentuan *error* dari data

Untuk mendapatkan data output sensor *gyroscope* dan *accelerometer*, maka pembacaan data dipakai sebesar 200 *readings* dan dibagikan nilai sebesar 200 untuk mendapatkan nilai yang bisa dilihat proses pengambilan data error pada Gambar 4.10. Sensor tersebut harus diletakkan secara datar supaya pembacaan bisa lebih akurat di mana diharapkan output pada coding tersebut adalah nol. Jika nilai output sebesar 1 atau lebih maka perlu

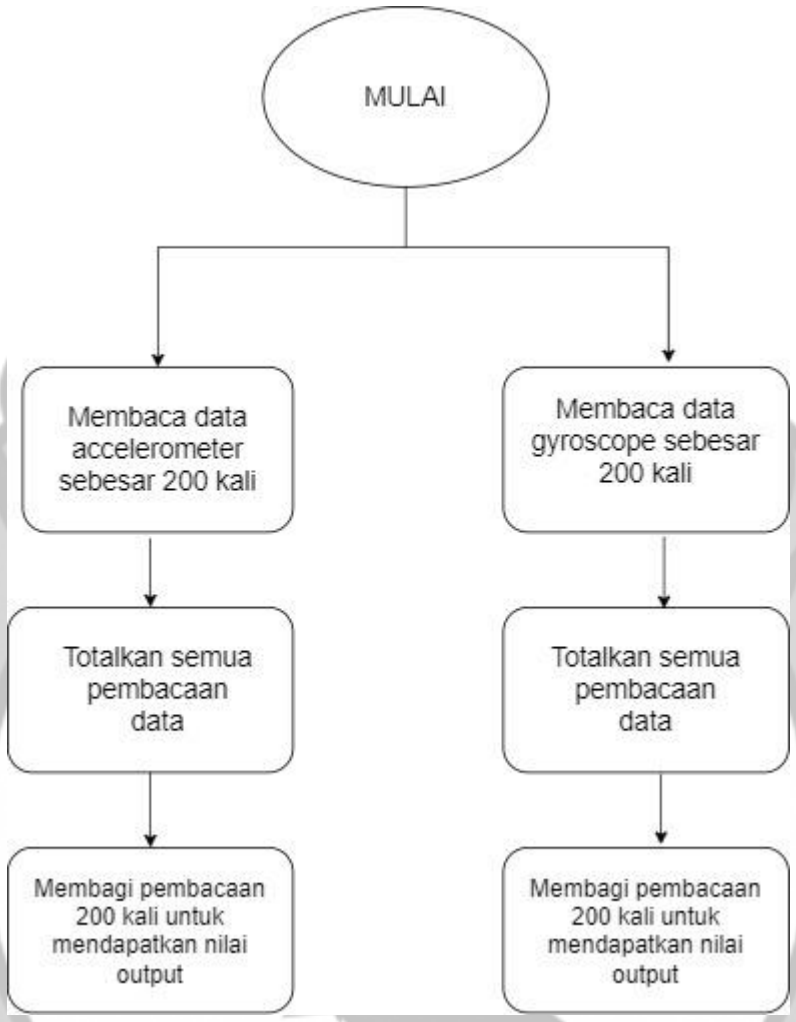
dikalibrasikan lagi dengan menggantikan nilai tersebut ke dalam data *accelerometer* dan *gyroscope* sebelumnya. *Coding* untuk menentukan *output* bisa dilihat *coding* pada Gambar 4.9.

```
void calculate_IMU_error(){
  // a function to calculate the accelerometer and gyro
  // to read the values the IMU needs to be in a flat surface

  // Read accelerometer 200 times
  while(c < 200) {
    Wire.beginTransmission(MPU);
    Wire.write(0x3B);
    Wire.endTransmission(false);
    Wire.requestFrom(MPU, 6, true);
    AccX = (Wire.read() << 8 | Wire.read()) / 16384.0;
    AccY = (Wire.read() << 8 | Wire.read()) / 16384.0;
    AccZ = (Wire.read() << 8 | Wire.read()) / 16384.0;
    // sum all readings
    AccErrorX = AccErrorX + ((atan((AccY) / sqrt(pow((AccX), 2) + pow((AccZ), 2))) * 180));
    AccErrorY = AccErrorY + ((atan(-1 * (AccX) / sqrt(pow((AccY), 2) + pow((AccZ), 2))) * 180));
    c++;
    }
  // Divide the sum by 200 to get the error value
  AccErrorX = AccErrorX / 200;
  AccErrorY = AccErrorY / 200;
  c = 0;

  //read gyro values by 200 times
  while(c < 200){
    Wire.beginTransmission(MPU);
    Wire.write(0x43);
    Wire.endTransmission(false);
    Wire.requestFrom(MPU, 6, true);
    GyroX = Wire.read() << 8 | Wire.read();
    GyroY = Wire.read() << 8 | Wire.read();
    GyroZ = Wire.read() << 8 | Wire.read();
    //sum all readings
    GyroErrorX = GyroErrorX + (GyroX / 131.0);
    GyroErrorY = GyroErrorY + (GyroY / 131.0);
    GyroErrorZ = GyroErrorZ + (GyroZ / 131.0);
    c++;
    }
   // Divide the sum by 200 to get the error value
   GyroErrorX = GyroErrorX / 200;
   GyroErrorY = GyroErrorY / 200;
   GyroErrorZ = GyroErrorZ / 200;
```

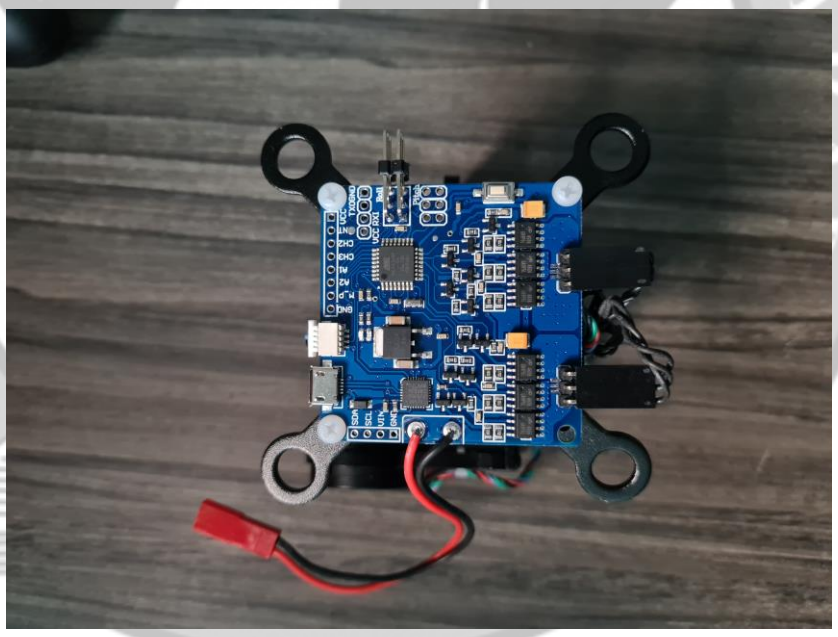
Gambar 4.9 *Coding* untuk menentukan *output* data



Gambar 4.10 Cara kerja coding penentuan nilai output

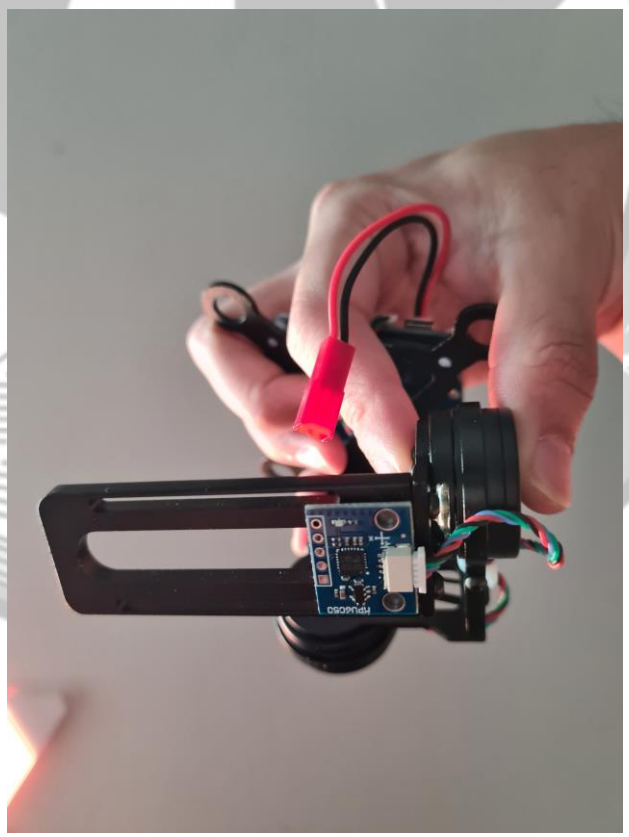
4.3 Pemasangan sensor MPU6050 dan brushless motor pada BGC board

Dengan sistem sensor MPU6050 pada Gambar 4.2, maka bisa diimplementasikan cara kerja dan pergerakan *accelerometer*, *gyroscope* dengan *complementary filter* kepada alat gimbal yang ada pada skema Gambar 4.1. *Board* BGC tersebut akan ditempatkan pada badan besi yang sudah dibuatkan, seperti pada Gambar 4.11. Untuk *brushless* motor dipasangkan kepada tangkai besi yang sudah dihubungkan pada badan besi seperti pada Gambar 4.12 dan untuk sensor MPU6050 dipasangkan pada bawah besi dudukan untuk kamera yang bisa dilihat pada Gambar 4.13



Gambar 4.11 Pemasangan BGC board pada badan

Gambar 4.12 Pemasangan *brushless* motor pada badan



Gambar 4.13 Pemasangan sensor MPU6050 pada badan