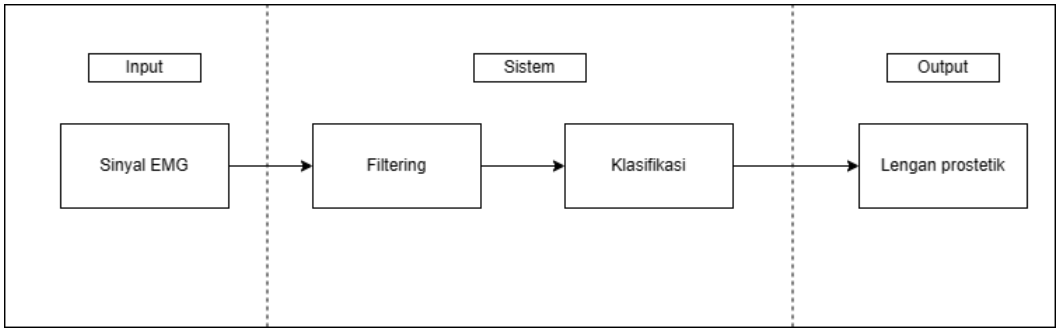
BAB 5 PERANCANGAN DAN IMPLEMENTASI

5.1 Perancangan Sistem

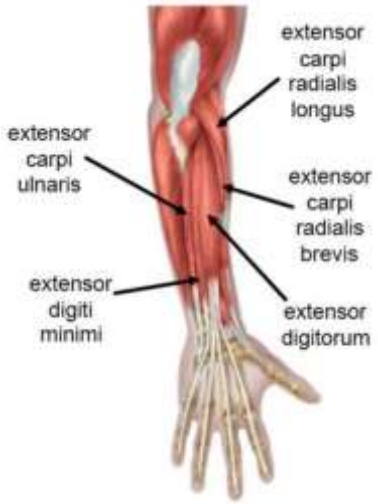


Gambar 5.1 Diagram Blok Sistem

Sistem akan dimulai dengan pembacaan sinyal melalui sensor EMG yang akan dijadikan sebagai input dari sistem. Sinyal tersebut akan dikonversi ke nilai tegangan yang kemudian akan difilter menggunakan 3 filter yang berbeda. Setelah melalui proses filtrasi, sinyal akan diklasifikasikan berdasarkan jenis gerakan dengan menggunakan algoritma K-NN. Hasil dari pengklasifikasian tersebut kemudian akan menjadi acuan dari keluaran sistem yang berupa gerakan pada lengan prostetik.

5.1.1 Perancangan Peletakan Pad Elektroda

Peletakan pad elektroda menjadi hal penting pada penelitian ini agar sinyal dari aktifitas otot dapat dideteksi dengan akurasi yang baik. Otot fleksor digitorum superficialis adalah otot yang dinilai menjadi tempat yang tepat untuk mendeteksi sinyal dari aktifitas otot yang dilakukan jari. Otot ekstensor adalah otot yang mengendalikan pergerakan jari, terletak di bagian atas lengan dan membentang sepanjang jari.

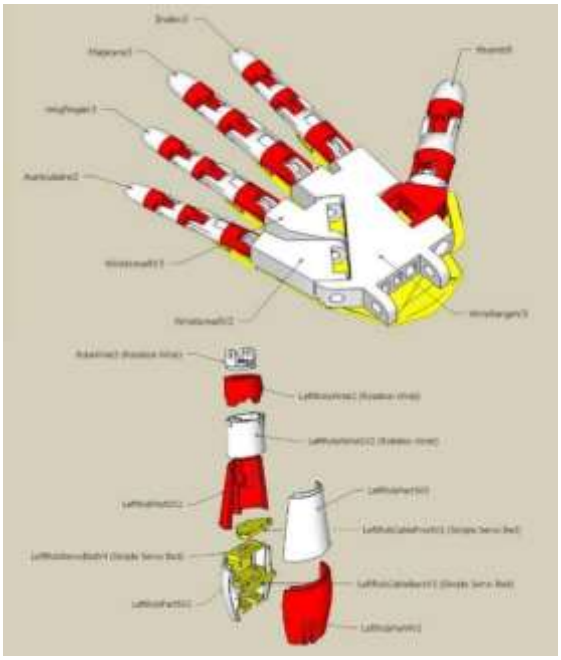


Gambar 5.2 Otot Ekstensor Digitorum

5.1.2 Perancangan Prototipe Lengan Bionik

Dalam melakukan penelitian yang membutuhkan pembuatan alat, maka diperlukan untuk merancang prototipenya agar alat dapat bekerja dengan baik. Pada penelitian ini, ada 2 bagian penting dari prototipe yang akan dibuat. Pertama, desain pergelangan tangan dan jari-jari. Desain pergelangan tangan dan jari-jari perlu dibuat berdasarkan pertimbangan kebutuhan pergerakan yang kompleks seperti ekstensi, adduksi, fleksi dan abduksi. Dibutuhkan kekuatan dan kestabilan agar dapat menopang beban saat menggunakannya.

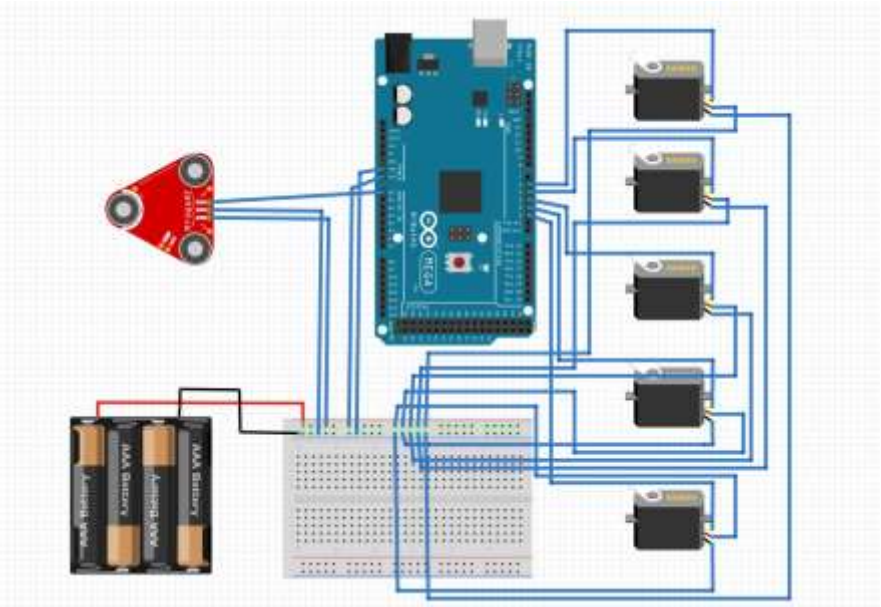
Kedua, desain lengan bawah dan tempat servo. Desain lengan bawah dan tempat servo perlu dibuat berdasarkan pertimbangan kemudahan dan kenyamanan pengguna. Lengan bagian bawah juga harus memiliki kekuatan untuk menahan gerakan dan beban yang diberikan pada saat bagian pergelangan dan jari-jari melakukan gerakan, sekaligus menjadi tempat untuk servo yang harus dirancang agar servo dapat terpasang dengan aman dan mudah diakses untuk kepentingan perawatan. Desain dari prototipe dapat dilihat pada Gambar 5.2.



Gambar 5.3 Desain Prototipe Lengan Bionik.

5.1.3 Perancangan Perangkat Keras

Ada lima komponen perangkat keras utama yang mendukung sistem dalam penelitian ini, yaitu mikrokontroler Arduino Mega 2560, sensor MyoWare 2.0, servo, breadboard atau PCB, dan baterai. Setiap servo dihubungkan dengan baterai dan pin 2, 3, 4, 5, dan 6 pada mikrokontroler. Kemudian, sensor MyoWare dihubungkan ke mikrokontroler. Ilustrasi koneksi dapat dilihat pada Gambar 5.3 dan informasi rinci tentang koneksi tersedia pada Tabel 5.1.



Gambar 5.4 Wiring diagram

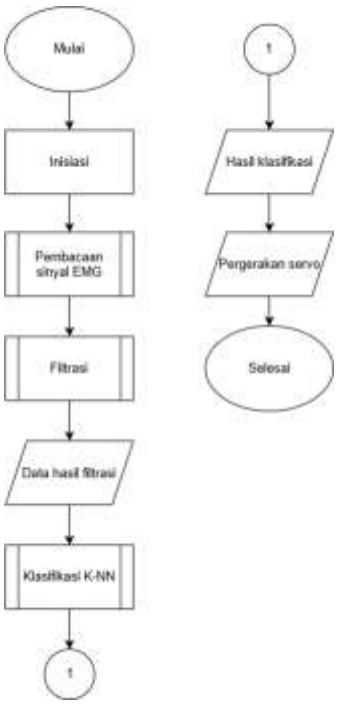
Tabel 5.1 Pin Skematik Sistem

Baterai	Arduino	Sensor EMG	Servo 1	Servo 2	Servo 3	Servo 4	Servo 5
+	Vin	+	+	+	+	+	+
-	Ground	-	-	-	-	-	-
	A0	SIG					
	2		SIG				
	3			SIG			
	4				SIG		
	5					SIG	
	6						SIG

5.1.4 Perancangan Perangkat Lunak

5.1.4.1 Perancangan Sistem Utama

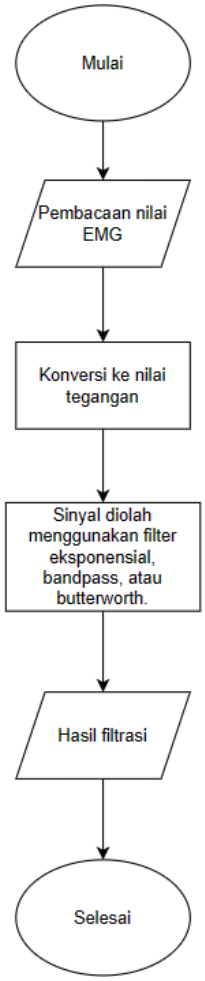
Pertama, sistem akan memulai dengan inisiasi variabel. Kemudian, melakukan pembacaan sinyal EMG menggunakan sensor myoware. Data yang terbaca akan melewati proses filtrasi untuk mengurangi noise dan interferensi pada sinyal. Setelah proses filtrasi selesai, data akan diklasifikasikan menggunakan algoritma K-NN. Kemudian sistem akan menghasilkan keluaran berupa gerakan pada servo berdasarkan hasil klasifikasi sebelumya. Agar lebih jelas, proses dapat dilihat pada gambar 5.4.



Gambar 5.5 Flowchart Sistem Utama

5.1.4.2 Perancangan Filter Eksponensial, Filter *Bandpass*, dan Filter *Butterworth*

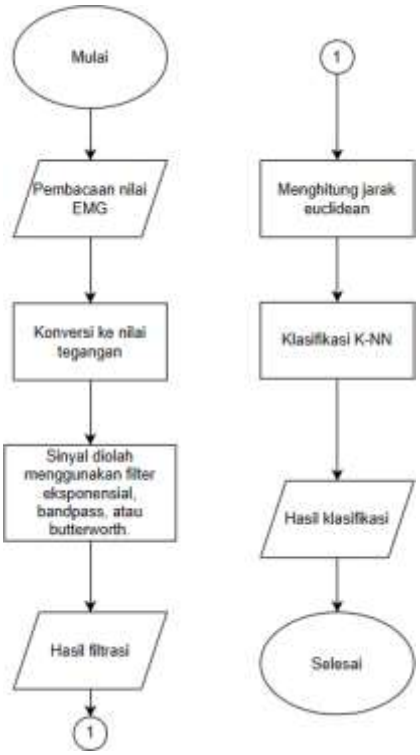
Masing-masing kode filter dimulai dengan membaca sinyal EMG dari sensor. Kemudian data hasil pembacaan sensor dikonversi ke nilai tegangan. Setelah dikonversi, data barulah difilter. Untuk lebih jelasnya, proses dapat dilihat pada Gambar 5.5.



Gambar 5.6 Flowchart Filter

5.1.4.3 Perancangan K-NN

Program pertama-tama akan mengambil data dari hasil pembacaan sinyal. Kemudian, data yang diperoleh akan dikonversi menjadi nilai tegangan. Setelah itu, data akan difiltrasi. Setelah proses filtrasi selesai, program akan menghitung jarak Euclidean data tersebut dan membandingkannya dengan dataset yang telah diinisialisasi pada awal program. Jika program menemukan jarak terdekatnya, maka nilai K yang terdekat dari data latih akan dipilih. Terakhir, program akan mengeluarkan output berupa hasil klasifikasi gerakan. Lebih jelasnya, proses dapat dilihat dalam bentuk flowchart pada Gambar 5.6



Gambar 5.7 Flowchart K-NN

5.2 Implementasi Sistem

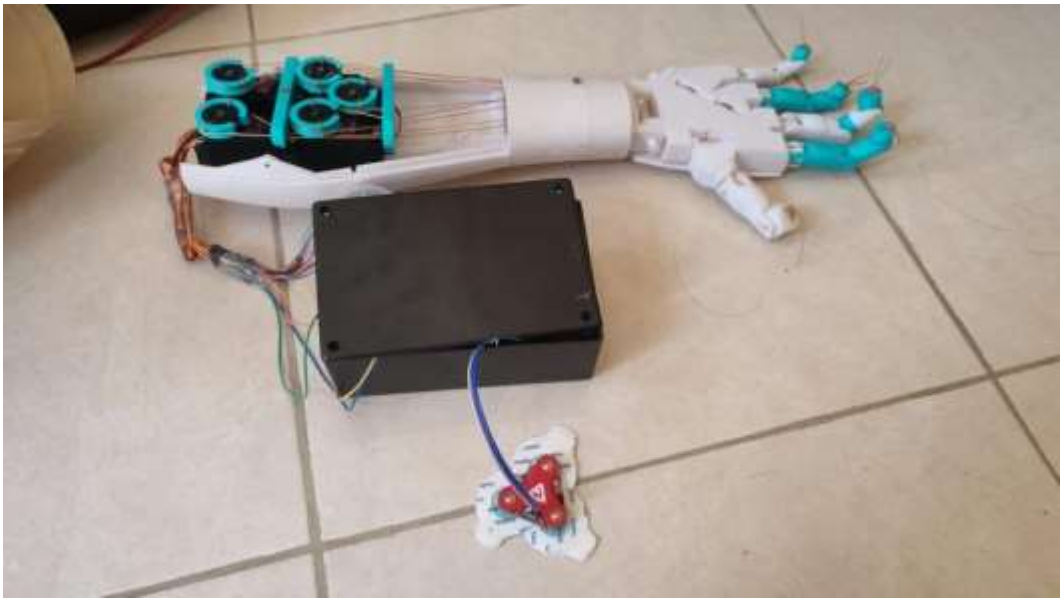
5.2.1 Implementasi Peletakan Pad Elektroda

Langkah implementasi pad elektroda terdiri dari beberapa tahap. Pertama, hubungkan pad elektroda pada sensor MyoWare 2.0. Selanjutnya, bersihkan area yang akan ditempelkan pad elektroda. Hal ini bertujuan agar pad elektroda dapat mendeteksi sinyal listrik dengan baik. Setelah itu, tempelkan pad elektroda yang telah terhubung ke sensor MyoWare 2.0 pada area yang telah ditentukan, yaitu pada otot ekstensor digitorum yang terletak di bagian atas lengan. Untuk petunjuk lebih detail, peletakan pad elektroda dapat dilihat pada Gambar 5.7.

Gambar 5.8 Implementasi Peletakan Pad Elektroda

5.2.2 Implementasi Prototipe Lengan Bionik

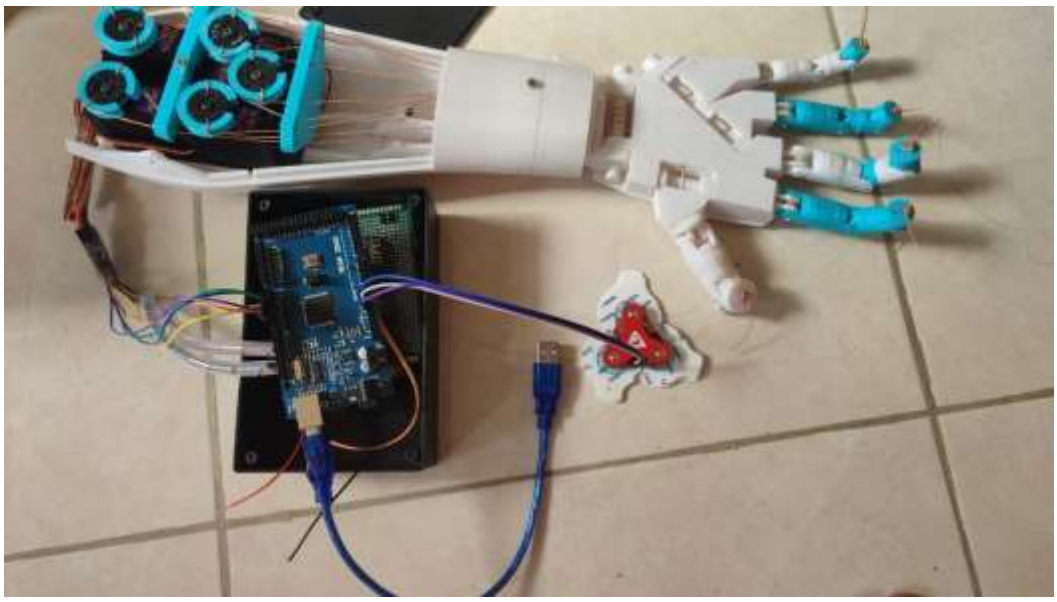
Implementasi prototipe lengan bionik melibatkan beberapa tahapan. Pertama, hubungkan komponen-komponennya. Selanjutnya, pasang pad elektroda pada tangan pengguna lengan bionik. Setelah pad elektroda terpasang pada area yang ditentukan, pengguna dapat melakukan kontraksi dan relaksasi pada jari yang kemudian akan diproses oleh sistem untuk menghasilkan gerakan serupa pada lengan bionik. Ilustrasi proses implementasi prototipe lengan bionik dapat dilihatpada Gambar 5.8.



Gambar 5.9 Implementasi Prototipe Lengan Bionik

5.2.3 Implementasi Perangkat Keras

Sistem ini terdiri dari beberapa komponen penting. Menggunakan mikrokontroler Arduino Mega 2560 sebagai kontroler utama, sistem ini dirancang agar dapat beroperasi secara portable dengan menggunakan baterai sebagai sumber daya. Sensor MyoWare 2.0 digunakan sebagai input untuk mengambil data dari pengguna lengan bionik. Data tersebut kemudian diolah dan diproses oleh sistem. Setelah proses pengolahan data pada Arduino Mega 2560, sistem akan menghasilkan keluaran berupa gerakan jari pada lengan bionik yang dikendalikan oleh servo MG996R yang terhubung melalui benang nilon. Ilustrasi implementasi perangkat keras lengan bionik tersedia pada Gambar 5.9.



Gambar 5.10 Implementasi Perangkat Keras

5.2.4 Implementasi Perangkat Lunak

5.2.4.1 Implementasi Sistem Utama

Kode Sistem Utama

```
#include <Servo.h>
#include <math.h>

// Definisi jumlah data dan jumlah fitur
const int NUM_GESTURES = 8;
const int NUM_FEATURES = 21;
const int filterOrder = 4; // Orde filter


Servo servo1;
Servo servo2;
Servo servo3;
Servo servo4;
Servo servo5;

float sensorValues[filterOrder + 1] = {0};
int dataIndex = 0;

// Definisi dataset
double gestures[NUM_GESTURES][NUM_FEATURES] = {

{0.05,0.1,0.14,0.18,0.22,0.22,0.21,0.21,0.21,0.21,0.39,0.64,0.74
,0.77,0.79,0.61,0.36,0.27,0.25,0.24,0.25}, // Gesture 1: cool

{0.04,0.08,0.11,0.15,0.19,0.19,0.19,0.19,0.19,0.35,0.55,0.72,0.8
9,1,0.92,0.82,0.7,0.57,0.51,0.5,0.44}, // Gesture 2: spiderman

{0.1,0.19,0.26,0.31,0.37,0.32,0.27,0.26,0.26,0.37,0.58,0.82,1.05
,1.27,1.25,1.11,0.94,0.71,0.55,0.47,0.4}, // Gesture 3: 3 pew

{0.05,0.09,0.14,0.2,0.25,0.25,0.25,0.25,0.24,0.24,0.37,0.44,0.45
,0.45,0.43,0.3,0.22,0.2,0.19,0.19,0.19},  // Gesture 4: 4 tengah

{0.18,0.18,0.18,0.19,0.19,0.18,0.19,0.19,0.19,0.18,0.18,0.18,0.1
8,0.19,0.19,0.18,0.19,0.19,0.19,0.18,0.18}, // Gesture 5: 5
terbuka

{0.19,0.19,0.25,0.84,1.17,2.1,2.31,1.95,1.53,1.34,1.12,1.08,1.01
,1.07,1.23,1.09,1.15,0.74,0.5,0.35,0.28}, // Gesture 6 : 6
Genggam

{0.18,0.19,0.47,0.45,0.42,0.37,0.35,0.31,0.31,0.32,0.33,0.34,0.3
,0.3,0.3,0.28,0.27,0.23,0.21,0.19,0.19}, // Gesture 7 : 7 oke
tengah

{0.19,0.19,0.58,1.24,1.29,1.31,1.09,0.99,0.92,0.88,0.88,0.76,0.7
2,0.66,0.63,0.6,0.52,0.48,0.46,0.35,0.29} // Gesture 8 : 8 Sip
};

// Fungsi untuk menghitung jarak Euclidean antara dua vektor
double euclideanDistance(const double a[], const double b[], int
length) {
    double sum = 0.0;
    for (int i = 0; i < length; ++i) {
        sum += pow((a[i] - b[i]), 2);
    }
    return sqrt(sum);
}

```

```
60    // Fungsi untuk melakukan klasifikasi menggunakan k-NN
61    int classifyGesture(const double input[], int k, double
62    distances[]) {
63        // Hitung jarak Euclidean antara input dan setiap data latih
64        for (int i = 0; i < NUM_GESTURES; ++i) {
65            distances[i] = euclideanDistance(input, gestures[i],
66    NUM_FEATURES);
67        }
68
69        // Temukan kategori terdekat
70        int minIndex = 0;
71        for (int i = 1; i < NUM_GESTURES; ++i) {
72            if (distances[i] < distances[minIndex]) {
73                minIndex = i;
74            }
75        }
76
77        return minIndex + 1; // Mengembalikan indeks yang sesuai
78    dengan kategori gerakan
79    }
80
81    void setup() {
82      Serial.begin(9600);
83
84      servo1.attach(2); // Assign servo pin numbers
85      servo2.attach(3);
86      servo3.attach(4);
87      servo4.attach(5);
88      servo5.attach(6);
89    }
90
91    void loop() {
92      // Contoh penggunaan: klasifikasi data baru
93      double newData[NUM_FEATURES];
94        for (int i = 0; i < NUM_FEATURES; ++i) {
95            int sensorValue = analogRead(A0); // Ubah A0 menjadi pin
96    tempat sensor EMG terhubung
97
98            // Konversi nilai ADC ke tegangan
99            float voltage = (sensorValue * 5.0) / 1023; // 5V
100   sebagai tegangan referensi
101
102           // Memperbarui array nilai sensor dengan nilai tegangan
103           for (int j = filterOrder; j > 0; j--) {
104               sensorValues[j] = sensorValues[j - 1];
105           }
106
107           sensorValues[0] = voltage;
108
109           // Hitung hasil filter Butterworth
110           float filteredValue = 0;
111           for (int j = 0; j <= filterOrder; j++) {
112               filteredValue += sensorValues[j];
113           }
114
115           filteredValue /= (filterOrder + 1);
116
117           // Simpan nilai yang telah difilter ke dalam newData
118           newData[i] = filteredValue;
119
120           // Tunggu sebentar antara pembacaan untuk stabilitas
121           delay(10);
122       }
123     int k = 3;
124
```

```
    double distances[NUM_GESTURES];
    int classifiedGesture = classifyGesture(newData, k,
  distances);

    // Tampilkan hasil klasifikasi dan perhitungan jarak Euclidean
    Serial.print("Data baru diklasifikasikan sebagai Gesture ");
    Serial.println(classifiedGesture);

    // Gunakan hasil klasifikasi untuk menggerakkan servo
    moveServos(classifiedGesture);

    // Tambahkan penundaan atau logika lain sesuai kebutuhan Anda
    delay(2000);
  }

  void moveServos(int gesture) {
    // Sesuaikan gerakan servo berdasarkan hasil klasifikasi
    switch (gesture) {
      case 1: // Gesture 1: cool
        // Atur posisi servo untuk Gesture 1
        servo1.write(0);
        servo2.write(120);
        servo3.write(160);
        servo4.write(120);
        servo5.write(0);
        break;
      case 2: // Gesture 2: spiderman
        // Atur posisi servo untuk Gesture 2
        servo1.write(0);
        servo2.write(0);
        servo3.write(160);
        servo4.write(120);
        servo5.write(0);
        break;
      case 3: // Gesture 3: 3 pew
        // Atur posisi servo untuk Gesture 3
        servo1.write(0);
        servo2.write(120);
        servo3.write(160);
        servo4.write(120);
        servo5.write(120);
        break;
      case 4: // Gesture 4: 4 tengah
        // Atur posisi servo untuk Gesture 4
        servo1.write(0);
        servo2.write(0);
        servo3.write(160);
        servo4.write(0);
        servo5.write(0);
        // Tidak melakukan apa pun jika hasil klasifikasi tidak
  valid
        break;
      case 5: // Gesture 5: 5 Terbuka
        // Atur posisi servo untuk Gesture 5
        servo1.write(0);
        servo2.write(0);
        servo3.write(0);
        servo4.write(0);
        servo5.write(0);
        // Tidak melakukan apa pun jika hasil klasifikasi tidak
  valid
        break;
      case 6: // Gesture 6: 6 genggam
        // Atur posisi servo untuk Gesture 6
        servo1.write(120);
```

```
190           servo2.write(120);
191           servo3.write(160);
192           servo4.write(120);
193           servo5.write(120);
194           // Tidak melakukan apa pun jika hasil klasifikasi tidak
195     valid
196           break;
197         case 7: // Gesture 7: 7 oke tengah
198           // Atur posisi servo untuk Gesture 7
199           servo1.write(120);
200           servo2.write(120);
201           servo3.write(160);
202           servo4.write(0);
203           servo5.write(0);
204           // Tidak melakukan apa pun jika hasil klasifikasi tidak
205     valid
206           break;
207         case 8: // Gesture 8: 8 sip
208           // Atur posisi servo untuk Gesture 8
209           servo1.write(0);
210           servo2.write(120);
211           servo3.write(160);
212           servo4.write(120);
213           servo5.write(120);
214           // Tidak melakukan apa pun jika hasil klasifikasi tidak
215     valid
216           break;
217       }
218     }
```

Kode di atas merujuk pada penggunaan sensor EMG untuk membaca sinyal dari otot-otot dan mengklasifikasikan gestur berdasarkan sinyal yang terbaca. Proses ini melibatkan beberapa langkah, mulai dari pembacaan data sensor hingga klasifikasi gestur dengan menggunakan metode K-NN. Setelah membaca dan memproses sinyal dari sensor EMG, kode tersebut menggunakan algoritma k-NN untuk membandingkan sinyal dengan dataset gestur yang telah ditentukan. Hasil dari klasifikasi digunakan untuk menggerakkan servomotor yang terhubung, sehingga membantu dalam membuat respons fisik berdasarkan gestur yang dikenali. Berikut adalah penjelasan bagian-bagian dalam kode:

Baris 1 dan 2 : Pemanggilan library servo dan math.
Baris 4-7 : Mendefinisikan jumlah gestur, jumlah fitur tiap gestur, dan orde Filter.
Baris 10-14 : Deklarasi 5 servo yang akan digunakan.
Baris 20-45 : Deklarasi dataset yang akan digunakan sebagai acuan klasifikasi.
Baris 47-54 : Fungsi untuk menghitung jarak euclidean.
Baris 56-69 : Fungsi untuk pengklasifikasian K-NN.
Baris 76 : Mengatur baudrate.
Baris 78-82 : Deklarasi pin yang digunaka servo.
Baris 87-116 : Pembacaan sensor. Data dari sensor dikonversi ke nilai teganga n dan difilter.
Baris 117 : Nilai K yang akan dipakai.
Baris 119-120 : Klasifikasi data.
Baris 123-124 : Keluaran hasil klasifikasi data.
Baris 127 : Hasil klasifikasi data akan digunakan untuk menggerakkan servo.
Baris 133-206 : Fungsi respon servo terhadap hasil klasifikasi gerakan.

5.2.4.2 Implementasi Filter Eksponensial, Filter *Bandpass*, dan Filter *Butterworth*

Kode di bawah ini berfungsi utnuk membaca data dari sensor EMG, mengonversi nilai sensor menjadi nilai volt, dan kemudian menerapkan 3 filter berbeda yang sudah ditentukan untuk melakukan penghalusan sinyal. Hal ini berguna untuk mengurangi noise interferensi yang tidak diinginkan pada sinyal sensor, sehingga menghasilkan nilai yang lebih stabil dan lebih mudah untuk diinterpretasikan atau diproses lebih lanjut. Penjelasan bagian-bagian pada kode ada di bawah kode. Berikut adalah kode untuk masing-masing filter yang digunakan.

Kode Filter Eksponensial

```
1   const float alpha = 0.5; // Faktor penghalusan (0 < alpha < 1)
2   const float maxADCValue = 1023.0; // Rentang nilai ADC
3   const float maxVoltage = 5.0; // Rentang volt yang diinginkan
4
5   const int filterOrder = 4; // Orde filter
6   float sensorValues[filterOrder + 1] = {0};
7   float smoothedValue = 0.0;
8
9   int dataIndex = 0;
10  bool allDataFiltered = false; // Flag untuk menandai jika semua
11  data telah difilter
12
13  void setup() {
14    Serial.begin(9600);
15  }
16
17  void loop() {
18    if (!allDataFiltered) {
19      if (dataIndex < sizeof(data) / sizeof(data[0])) {
20         int sensorValue = analogRead(A0); // Membaca nilai sensor
21  EMG
22
23        // Konversi nilai sensor ke nilai volt (dari rentang 0-
24  1023 ke rentang 0-5 volt)
25         float voltage = sensorValue * (maxVoltage / maxADCValue);
26
27        // Memperbarui array nilai sensor dengan nilai volt
28        for (int i = filterOrder; i > 0; i--) {
29          sensorValues[i] = sensorValues[i - 1];
30        }
31
32        sensorValues[0] = voltage;
33
34        // Hitung nilai yang telah difilter menggunakan filter
35  Butterworth
36        smoothedValue = 0;
37        for (int i = 0; i <= filterOrder; i++) {
38          smoothedValue += sensorValues[i];
39        }
40        smoothedValue /= (filterOrder + 1);
41
42        Serial.print("Sensor (ADC): ");
43        Serial.print(sensorValue);
44        Serial.print("\tVoltage: ");
45         Serial.print(voltage, 2); // Menampilkan nilai volt dengan
46  dua desimal
```

```
47          Serial.print(" V\tSmoothed: ");
48          Serial.println(smoothedValue, 2);
49
50          dataIndex++;
51        } else {
52          Serial.println("Semua data telah difilter. Penghalusan
53    selesai.");
54          allDataFiltered = true; // Menandai bahwa semua data telah
55    difilter
56          return; // Keluar dari loop utama
57        }
58      }
59      delay(100); // Delay sesuai dengan kebutuhan Anda
60    }
```

Baris 1-11 : Deklarasi variabel-variabel yang dibutuhkan.

Baris 14 : Mengatur kecepatan baudrate.

Baris 18-38 : Membaca nilai dari sensor. Kemudian nilai tersebut dikonversi ke nilai tegangan dan difilter.

Baris 40-51 : Keluaran program.

Kode Filter *Bandpass*

```
1     const int order = 4;
2     const double h[] = {0.0054, 0.2393, 0.6355, 0.2393, 0.0054};  //
3     Koefisien filter FIR
4
5     const int bufferSize = order + 1;
6     double inputBuffer[bufferSize] = {0};  // Buffer untuk menyimpan
7     data masukan
8     double output = 0;  // Hasil keluaran filter
9
10    const int maxADC = 1023;  // Nilai maksimum ADC
11    const double maxVolt = 5.0;  // Nilai maksimum tegangan (dalam
12    volt)
13
14    void setup() {
15      Serial.begin(9600);
16    }
17
18    void loop() {
19      // Membaca nilai sensor EMG
20      int sensorValue = analogRead(A0);
21
22      // Konversi nilai ADC ke tegangan
23      double voltage = (sensorValue * maxVolt) / maxADC;
24
25      // Update buffer dengan data masukan baru
26      for (int i = bufferSize - 1; i > 0; i--) {
27        inputBuffer[i] = inputBuffer[i - 1];
28      }
29      inputBuffer[0] = voltage;  // Simpan nilai tegangan ke dalam
30    buffer
31
32      // Hitung keluaran filter FIR
33      for (int i = 0; i <= order; i++) {
34        output += h[i] * inputBuffer[i];
35      }
36
37      // Kirim hasil keluaran ke Serial Monitor
38      Serial.println(output);
39
40      // Reset nilai output untuk iterasi berikutnya
41      output = 0;
42
```

```
43      // Tambahkan delay sesuai dengan frekuensi sampel (misalnya, 1
44    kHz)
45      delay(1);  // 1 ms delay
46    }
```

Baris 1-10 : Deklarasi variabel-variabel yang dibutuhkan.
Baris 13 : Mengatur kecepatan baudrate.
Baris 17-31 : Membaca nilai dari sensor. Kemudian nilai tersebut dikonversi ke nilai tegangan dan difilter.
Baris 35 : Keluaran program.

Kode Filter Eksponensial

```
1     const int filterOrder = 4; // Orde filter
2
3     float sensorValues[filterOrder + 1] = {0};
4     float filteredValue = 0;
5     int dataIndex = 0;
6
7     const int maxADC = 1023;  // Nilai maksimum ADC
8     const float maxVolt = 5.0;  // Nilai maksimum tegangan (dalam
9     volt)
10
11    void setup() {
12      // Mulai komunikasi serial
13      Serial.begin(9600);
14    }
15
16    void loop() {
17      // Baca nilai dari sensor EMG (misalnya, dari pin A0)
18      int sensorValue = analogRead(A0);
19
20      // Konversi nilai ADC ke tegangan
21      float voltage = (sensorValue * maxVolt) / maxADC;
22
23      // Memperbarui array nilai sensor dengan nilai tegangan
24      for (int i = filterOrder; i > 0; i--) {
25        sensorValues[i] = sensorValues[i - 1];
26      }
27
28      sensorValues[0] = voltage;
29
30      // Hitung hasil filter Butterworth
31      filteredValue = 0;
32      for (int i = 0; i <= filterOrder; i++) {
33        filteredValue += sensorValues[i];
34      }
35
36      filteredValue /= (filterOrder + 1);
37
38      // Tampilkan nilai sensor dan hasil filter ke Serial Monitor
39      Serial.println(filteredValue);
40
41      // Tambahkan delay sesuai dengan frekuensi sampel (misalnya, 1
42    kHz)
43      delay(1);  // 1 ms delay
44    }
```

Baris 1-8 : Deklarasi variabel-variabel yang dibutuhkan.
Baris 12 : Mengatur kecepatan baudrate.
Baris 17-35 : Membaca nilai dari sensor. Kemudian nilai tersebut dikonversi ke nilai tegangan dan difilter.
Baris 38 : Keluaran program.

5.2.4.3 Implementasi K-NN

Kode K-NN

```
#include <math.h>

// Definisi jumlah data dan jumlah fitur
const int NUM_GESTURES = 8;
const int NUM_FEATURES = 21; // Ubah jumlah fitur menjadi 1

// Definisi dataset
double gestures[NUM_GESTURES][NUM_FEATURES] = {

  {0.05,0.1,0.14,0.18,0.22,0.22,0.21,0.21,0.21,0.21,0.39,0.64,0.74
,0.77,0.79,0.61,0.36,0.27,0.25,0.24,0.25}, // Gesture 1: cool

  {0.04,0.08,0.11,0.15,0.19,0.19,0.19,0.19,0.19,0.35,0.55,0.72,0.8
9,1,0.92,0.82,0.7,0.57,0.51,0.5,0.44}, // Gesture 2: spiderman

  {0.1,0.19,0.26,0.31,0.37,0.32,0.27,0.26,0.26,0.37,0.58,0.82,1.05
,1.27,1.25,1.11,0.94,0.71,0.55,0.47,0.4}, // Gesture 3: 3 pew

  {0.05,0.09,0.14,0.2,0.25,0.25,0.25,0.25,0.24,0.24,0.37,0.44,0.45
 ,0.45,0.43,0.3,0.22,0.2,0.19,0.19,0.19},  // Gesture 4: 4 tengah

  {0.18,0.18,0.18,0.19,0.19,0.18,0.19,0.19,0.19,0.18,0.18,0.18,0.1
8,0.19,0.19,0.18,0.19,0.19,0.19,0.18,0.18}, // Gesture 5: 5
terbuka

  {0.19,0.19,0.25,0.84,1.17,2.1,2.31,1.95,1.53,1.34,1.12,1.08,1.01
,1.07,1.23,1.09,1.15,0.74,0.5,0.35,0.28}, // Gesture 6 : 6
Genggam

  {0.18,0.19,0.47,0.45,0.42,0.37,0.35,0.31,0.31,0.32,0.33,0.34,0.3
,0.3,0.3,0.28,0.27,0.23,0.21,0.19,0.19}, // Gesture 7 : 7 oke
tengah

  {0.19,0.19,0.58,1.24,1.29,1.31,1.09,0.99,0.92,0.88,0.88,0.76,0.7
2,0.66,0.63,0.6,0.52,0.48,0.46,0.35,0.29} // Gesture 8 : 8 Sip
};


// Fungsi untuk menghitung jarak Euclidean antara dua vektor
 double euclideanDistance(const double a[], const double b[], int
length) {
    double sum = 0.0;
    for (int i = 0; i < length; ++i) {
        sum += pow((a[i] - b[i]), 2);
    }
    return sqrt(sum);
}

// Fungsi untuk melakukan klasifikasi menggunakan k-NN
int classifyGesture(const double input[], int k, double
distances[]) {
     // Hitung jarak Euclidean antara input dan setiap data latih
    for (int i = 0; i < NUM_GESTURES; ++i) {
        distances[i] = euclideanDistance(input, gestures[i],
NUM_FEATURES);
    }

    // Temukan kategori terdekat
    int minIndex = 0;
    for (int i = 1; i < NUM_GESTURES; ++i) {
        if (distances[i] < distances[minIndex]) {
            minIndex = i;
```

```
            }
        }

        return minIndex + 1; // Mengembalikan indeks yang sesuai
    dengan kategori gerakan
    }

    const float alpha = 0.5;
    const float maxADCValue = 1023.0;
    const float maxVoltage = 5.0;
    const int sensorPin = A0; // Ganti dengan pin sensor EMG Anda
    float smoothedValue = 0.0;


    void setup() {
        Serial.begin(9600);
    }

    void loop() {
        double newData[NUM_FEATURES]; // Sesuaikan dengan jumlah
    fitur

        // Membaca nilai dari sensor EMG dan memperbarui newData
        for (int i = 0; i < NUM_FEATURES; ++i) {
            int sensorValue = analogRead(sensorPin); // Membaca
    nilai dari sensor EMG

            // Konversi nilai dari sensor ke tegangan (dari rentang
    0-1023 ke rentang 0-5 volt)
            float voltage = sensorValue * (maxVoltage /
    maxADCValue);

            // Proses filtrasi
            smoothedValue = alpha * voltage + (1 - alpha) *
    smoothedValue;

            newData[i] = smoothedValue; // Menggunakan nilai yang
    telah difiltrasi

            delay(10); // Delay untuk stabilitas
        }

        int k = 3;
        double distances[NUM_GESTURES];
        int classifiedGesture = classifyGesture(newData, k,
    distances);

        // Menampilkan hasil klasifikasi dan perhitungan jarak
    Euclidean
        Serial.print("Data baru diklasifikasikan sebagai Gesture ");
        Serial.println(classifiedGesture);

        Serial.println("Jarak Euclidean ke setiap kategori:");
        for (int i = 0; i < NUM_GESTURES; ++i) {
            Serial.print("Gesture ");
            Serial.print(i + 1);
            Serial.print(": ");
            Serial.println(distances[i]);
        }

        delay(5000);
    }
```

Kode di atas mengimplementasikan teknik klasifikasi gestur berbasis sensor EMG dengan mengukur kemiripan antara data sensor baru dan dataset

gestur yang telah ditentukan. Dengan menggunakan teknik k-NN dan jarak Euclidean, kode ini dapat mengenali gestur yang paling mirip dengan data sensor yang baru untuk kemudian digunakan dalam aplikasi pengendalian atau identifikasi berbasis gestur. Berikut adalah penjelasan bagian-bagian pada kodenya:

Baris 1 : Pemanggilan library math.
Baris 4-5 : Mendefinisikan jumlah gestur, jumlah fitur tiap gestur, dan orde filter.
Baris 8-33 : Deklarasi dataset yang akan digunakan sebagai acuan klasifikasi.
Baris 36-43 : Fungsi untuk menghitung jarak euclidean.
Baris 45-58 : Fungsi untuk pengklasifikasian K-NN.
Baris 64-68 : Deklarasi variabel-variabel yang dibutuhkan.
Baris 70 : Mengatur kecepatan baudrate.
Baris 76-93 : Pembacaan sensor. Data dari sensor dikonversi ke nilai tegangan dan difilter.
Baris 95 : Nilai K yang akan dipakai.
Baris 96-97 : Klasifikasi data.
Baris 100-108 : Keluaran hasil klasifikasi data.