BAB 3

ANALISIS DAN PERANCANGAN SISTEM

Pada bab ini analisis dan perancangan sistem yang akan dibahas adalah mengenai analisis masalah, analisis sistem, spesifikasi kebutuhan perangkat, analisis kebutuhan non fungsional, analisis kebutuhan fungsional, perancangan simulasi lengan robot.

3.1 Analisis Masalah

Hasil analisis dari penelitian sebelumnya mengenai robot lengan penulis kata oleh Petrus Claver Hendar Kristyanto dari Universitas Sanata Dharma Yogyakarta didapat bahwa robot lengan hanya dapat menulis karakter berupa huruf alphabet saja dengan titik acuan yang sudah di tentukan sebelumnya, untuk penulisan karakter angka tidak bisa dan persentase keberhasilan penulisan pada semua karakter huruf mencapai 62%. Hasil dari percobaan menulis karakter dari huruf A sampai Z, diperoleh huruf yang tidak dapat di tulis ada 10 huruf yaitu huruf K, M, N, Q, R, V, W, X, Y, Z. Berikut tabel data keberhasilan penulisan dari penelitian sebelumnya :

Tabel 3.1 Data keberhasilan penulisan karakter dari penelitian sebelumnya

HURUF	POSISI KARAKTER				
	1	2	3	4	5
A	1	1	1	1	1
B	1	1	1	1	1
C	1	1	1	1	1
D	1	1	1	1	1
E	1	1	1	1	1
F	1	1	1	1	1
G	1	1	1	1	1
H	1	1	1	1	1

I	1	1	1	1	1
J	1	1	1	1	1
K	0	0	0	0	0
L	1	1	1	1	1
M	0	0	0	0	0
N	0	0	0	0	0
O	1	1	1	1	1
P	1	1	1	1	1
Q	0	0	0	0	0
R	0	0	0	0	0
S	1	1	1	1	1
T	1	1	1	1	1
U	1	1	1	1	1
V	0	0	0	0	0
W	0	0	0	0	0
X	0	0	0	0	0
Y	0	0	0	0	0
Z	0	0	0	0	0
TOTAL HURUF					130
HURUF YANG TERBACA					80
HURUF YANG TIDAK TERBACA					50
PRESENTASE KEBERHASILAN YANG TERBACA					62%
0 = TIDAK TERBACA, 1 = TERBACA					

3.2 Analisis Sistem

Dalam analisis sistem akan dijabarkan mengenai analisis yang berkaitan dengan pembuatan aplikasi seperti analisis solusi, proses yang dilakukan, basis data, kebutuhan fungsional dan kebutuhan non fungsional.

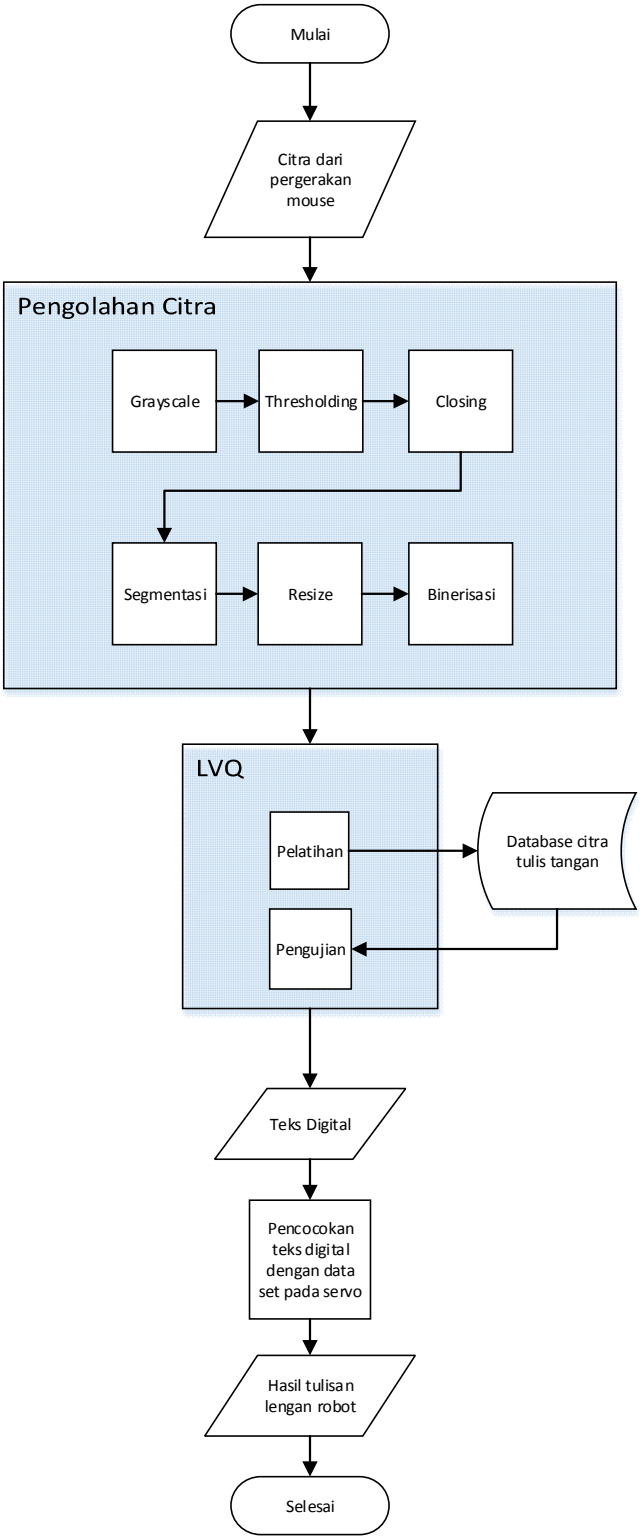
3.2.1 Analisis Solusi

Penyelesaian masalah dilakukan dengan membangun aplikasi pengenalan tulisan tangan menggunakan data set hasil penelitian dari **I Kadek Meganjaya**[17] dengan metode *Learning Vector Quantization* (LVQ) dan input berupa citra yang ditulis tangan. Data masukan diproses dengan pengolahan citra dengan implementasi metode *Learning Vector Quantization* yang menggunakan data latih dari 100 sampel tulisan tangan, sehingga data keluaran atau akurasi yang diharapkan nilainya tinggi. Setelah citra tulisan tangan sudah dikenali, maka selanjutnya program akan mengirimkan data dari proses pengolahan citra ke arduino dan mencocokan data teks dari pengolahan citra dengan data set yang tersedia untuk menggerakan servo menulis sesuai dengan data yang dikirim.

3.2.2 Analisis Proses

Proses yang dilakukan pada penelitian ini dibagi menjadi beberapa bagian dimana setiap proses memiliki peranan masing-masing dalam melakukan pengenalan tulisan tangan dan mengirimkan data hasil pengenalan citra ke arduino. Proses awal yang dilakukan yaitu memasukkan citra tulisan tangan ke dalam aplikasi. Langkah selanjutnya melalui tahap pengolahan citra meliputi *grayscale*, *thresholding*, *closing*, segmentasi, *resize* dan binerisasi. Dari tahap pengolahan citra akan didapatkan *array* angka biner dari setiap citra karakter yang sudah tersegmentasi. *Array* biner kemudian diolah kembali pada tahap klasifikasi dengan metode LVQ baik pelatihan maupun pengujian. Hasil dari pengenalan tulisan tangan berupa teks digital. [17]

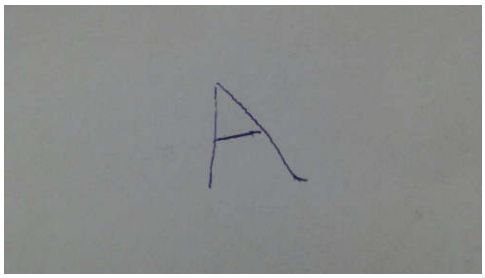
Setelah proses untuk pengenalan citra tulisan yang menghasilkan teks digital, maka selanjutnya teks digital tersebut dikirimkan melalui *port serial* yang nantinya akan dibaca dan menginstruksikan arduino untuk memproses menulis karakter tersebut sesuai dengan input yang diberikan. Berikut adalah arsitektur umum dari sistem yang akan dibangun.



Gambar 3.1 Arsitektur Umum Sistem

3.2.2.1 Data Masukan

Data masukan dapat diinputkan dengan 2 metode yaitu dengan menggambar langsung pada kanvas yang telah disediakan pada program, atau dengan *browse file* citra tulisan tangan dengan format gambar .jpg. Untuk metode *browse file* citra dapat diperoleh melalui scan atau foto resolusi minimal 8 *megapixel*. Citra tulisan tangan ditulis menggunakan pulpen pada kertas HVS dengan *background* putih. Dalam penelitian ini data masukan secara otomatis di-*resize* menjadi ukuran 750x1300 *pixel*.



Gambar 3.2 Contoh citra *input*

Berikut adalah tabel matrik warna *Red* (R), *Green* (G) dan *Blue* (B) dari contoh citra masukan di atas.

Tabel 3.2 Matriks Nilai RGB Citra Masukan

x/y	**0**	**1**	**2**	**3**	…	**749**
0	R = 155 G = 135 B = 162	R = 152 G = 132 B = 159	R = 155 G = 135 B = 162	R = 155 G = 135 B = 162	…	R = 148 G = 136 B = 156
1	R = 157 G = 138 B = 165	R = 154 G = 135 B = 162	R = 156 G = 136 B = 163	R = 159 G = 139 B = 166	…	R = 149 G = 137 B = 158
2	R = 154 G = 136 B = 162	R = 150 G = 132 B = 158	R = 158 G = 140 B = 166	R = 161 G = 143 B = 169	…	R = 150 G = 136 B = 159
3	R = 157 G = 139 B = 165	R = 151 G = 133 B = 159	R = 155 G = 137 B = 163	R = 154 G = 136 B = 162	…	R = 144 G = 130 B = 153
…	. …	. …	. …	. …	…	. …
1299	R = 89 G = 76 B = 94	R = 94 G = 81 B = 99	R = 92 G = 79 B = 97	R = 93 G = 80 B = 98	…	R = 93 G = 80 B = 97

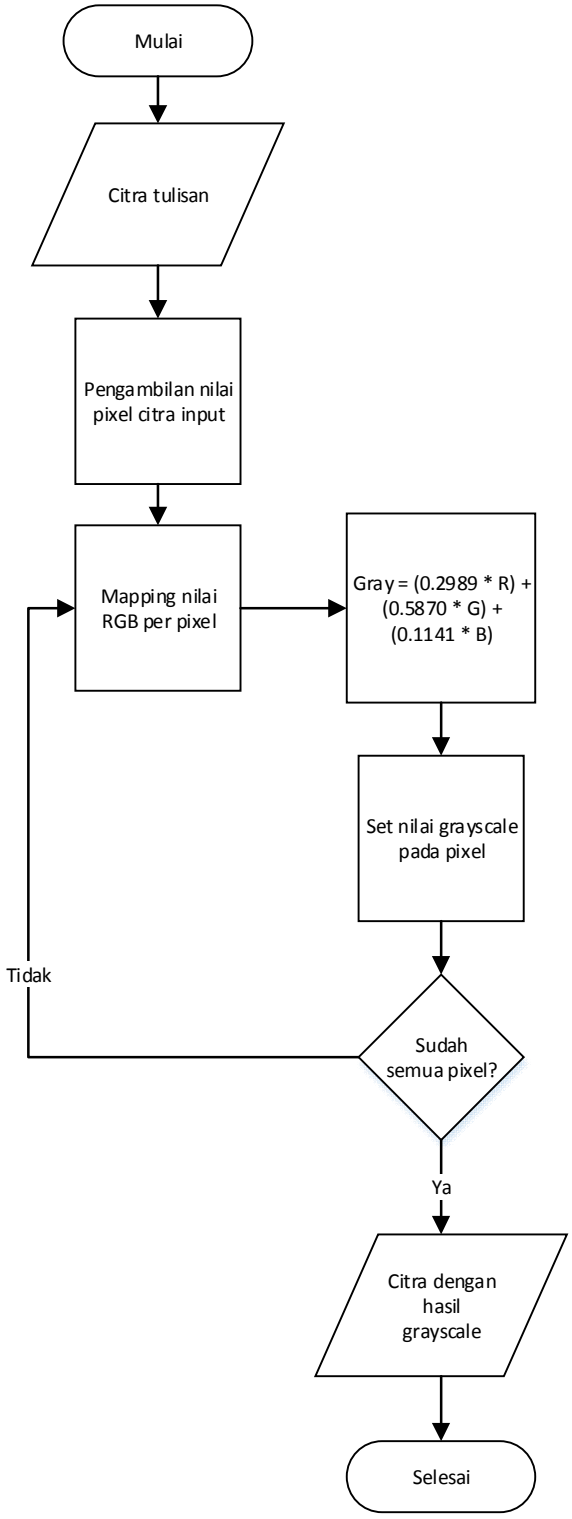
3.2.2.2 *Grayscale*

Mengubah format warna menjadi *grayscale* berfungsi untuk mengecilkan *range* warna menjadi dari 0 sampai dengan 255. Proses ini akan memudahkan ketika ingin melakukan *threshold* citra menjadi citra hitam putih. Berikut adalah *flowchart* dari proses *grayscale*. Adapun langkah-langkah untuk proses *grayscale* adalah sebagai berikut.

1. Warna citra dikelompokkan berdasarkan nilai *red, green dan blue*.
2. Kemudian menggunakan persamaan 2.1 , maka akan didapatkan nilai warna *grayscale* citra.
3. Nilai *grayscale* yang didapat menggantikan nilai RGB pada setiap *pixel*. Misalkan citra pda *pixel* (0,0) mempunyai nilai *Red* = 155, *Green* = 135, *Blue* = 162, maka berdasarkan persamaan 2.1 menjadi:

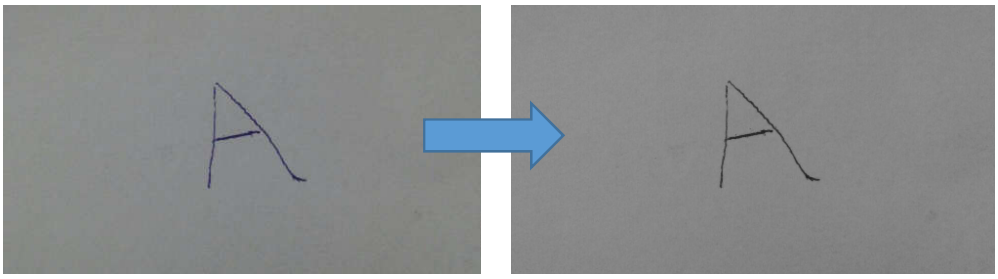
$$I = (0{,}2989 * R) + (0{,}5870 * G) + (0{,}1141 * B)$$
$$= (0{,}2989 * 155) + (0{,}5870 * 135) + (0{,}1141 * 162)$$
$$= 46{,}3295 + 79{,}245 + 18{,}4842 = 144$$

Dari perhitungan di atas, maka pixel yang tadinya bernilai Red = 155, Green = 135, Blue = 162 diperbaharui menjadi nilai grayscale = 144. Berikut merupakan *flowchart* dari proses *grayscale.*



Gambar 3.3 Flowchart Proses *Grayscale*

Berikut ini merupakan hasil dari proses *grayscale*



Gambar 3.4 Hasil Proses *Grayscale*

Berikut adalah tabel matrik warna *Red* (R), *Green* (G) dan *Blue* (B) dari hasil proses *grayscale*.

Tabel 3.3 Matriks Nilai RGB Citra *Grayscale*

x/y	**0**	**1**	**2**	**3**	…	**749**
0	R = 144 G = 144 B = 144	R = 141 G = 141 B = 141	R = 144 G = 144 B = 144	R = 144 G = 144 B = 144	…	R = 141 G = 141 B = 141
1	R = 146 G = 146 B = 146	R = 143 G = 143 B = 143	R = 145 G = 145 B = 145	R = 148 G = 148 B = 148	…	R = 142 G = 142 B = 142
2	R = 144 G = 144 B = 144	R = 140 G = 140 B = 140	R = 148 G = 148 B = 148	R = 151 G = 151 B = 151	…	R = 142 G = 142 B = 142
3	R = 147 G = 147 B = 147	R = 141 G = 141 B = 141	R = 145 G = 145 B = 145	R = 144 G = 144 B = 144	…	R = 136 G = 136 B = 136
…	…	…	…	…	…	…
1299	R = 81 G = 81 B = 81	R = 86 G = 86 B = 86	R = 84 G = 84 B = 84	R = 85 G = 85 B = 85	…	R = 85 G = 85 B = 85

3.2.2.3 *Threshold*

Metode *threshold* yang digunakan dalam penelitian ini yaitu menggunakan metode *Sauvola Threshold*. Sauvola merupakan kategori dari metode *local threshold*, dimana nilai ambang pada setiap *pixel* tergantung dari jumlah tetangga yang digunakan. Dalam penelitian ini, parameter yang digunakan sebagai berikut:

1. jumlah tetangga = 21 *pixel*. Nilai 21 cocok digunakan karena jika nilai tetangga teralu besar, maka waktu yang dibutuhkan menjadi lebih lama, atau jika terlalu kecil maka hasil yang didapatkan tidak maksimal.
2. Konstanta R = 128
3. k = 0,3
4. nilai m(x,y) didapatkan dari persamaan 2.3
5. nilai s(x,y) didapatkan dari persamaan standar deviasi 2.4

Dari citra tulisan tangan yang digunakan, akan dicari nilai ambang pada *pixel* (0,0) dengan nilai *grayscale* 144. Selanjutnya cari *pixel* tetangga mana saja yang sesuai dengan jumlah tetangga yang sudah ditentukan sebelumnya yaitu 21. Kemudian dari hasil pencarian tersebut didapatkan matriks citra seperti pada tabel, dimana tetangga *pixel* (0,0) adalah dari *pixel* (0,0) sampai dengan *pixel* (10,10).

Tabel 3.4 Contoh Matriks Citra yang akan di *Threshold*

x/y	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0	144	141	144	144	142	142	143	145	149	146	142
1	146	143	145	148	142	145	143	146	143	145	147
2	144	140	148	151	146	140	143	143	144	144	140
3	147	141	145	144	147	144	144	145	145	145	143
4	144	138	145	145	146	144	149	142	141	141	144
5	141	151	145	141	141	142	142	150	140	140	144
6	146	145	148	141	142	142	144	144	145	145	147
7	144	143	144	139	138	144	142	144	140	140	142
8	147	141	139	146	140	150	141	148	141	141	142
9	144	143	139	142	144	145	148	142	146	146	142
10	144	149	144	143	147	153	130	144	142	142	140

Langkah pertama cari nilai rata-rata m(0,0) dengan persamaan 2.3, sehingga hasilnya seperti berikut.

$$m(x,y) = \frac{\sum_{i=min}^{i=max} \sum_{i=min}^{i=max} img(i,j)}{i \quad j}$$

$$m(0,0) = \frac{144+142+\ \ +142+142}{11\ \ 11} = 143{,}79$$

Kemudian untuk mencari nilai standar deviasi s(0,0) digunakan persamaan 2.4 sehingga hasilnya sebagai berikut.

$$s(x,y) = \sqrt{\frac{\sum_{i=min}^{i=max}\sum_{i=min}^{i=max}(img(i,j)-m(x-y)^2)}{(i \quad j)-1}}$$

$$s(0,0) = \sqrt{\frac{(144-143{,}79)^2+(142-143{,}79)^2+ \quad +(142-143{,}79)^2+(143-143{,}79)^2}{121-1}} = 3{,}06$$

Langkah selanjutnya masukkan nilai m(0,0) dan s(0,0) ke dalam persamaan 2.2 untuk mendapatkan nilai ambang T(0,0). Sehingga hasilnya menjadi*:*

$$T(x,y) = m(x,y) * (1 + k * (\frac{s(x,y)}{R} \quad 1))$$

$$T(0,0) = m(0,0) * (1 + 0{,}3 * (\frac{s(0,0)}{128} - 1))$$

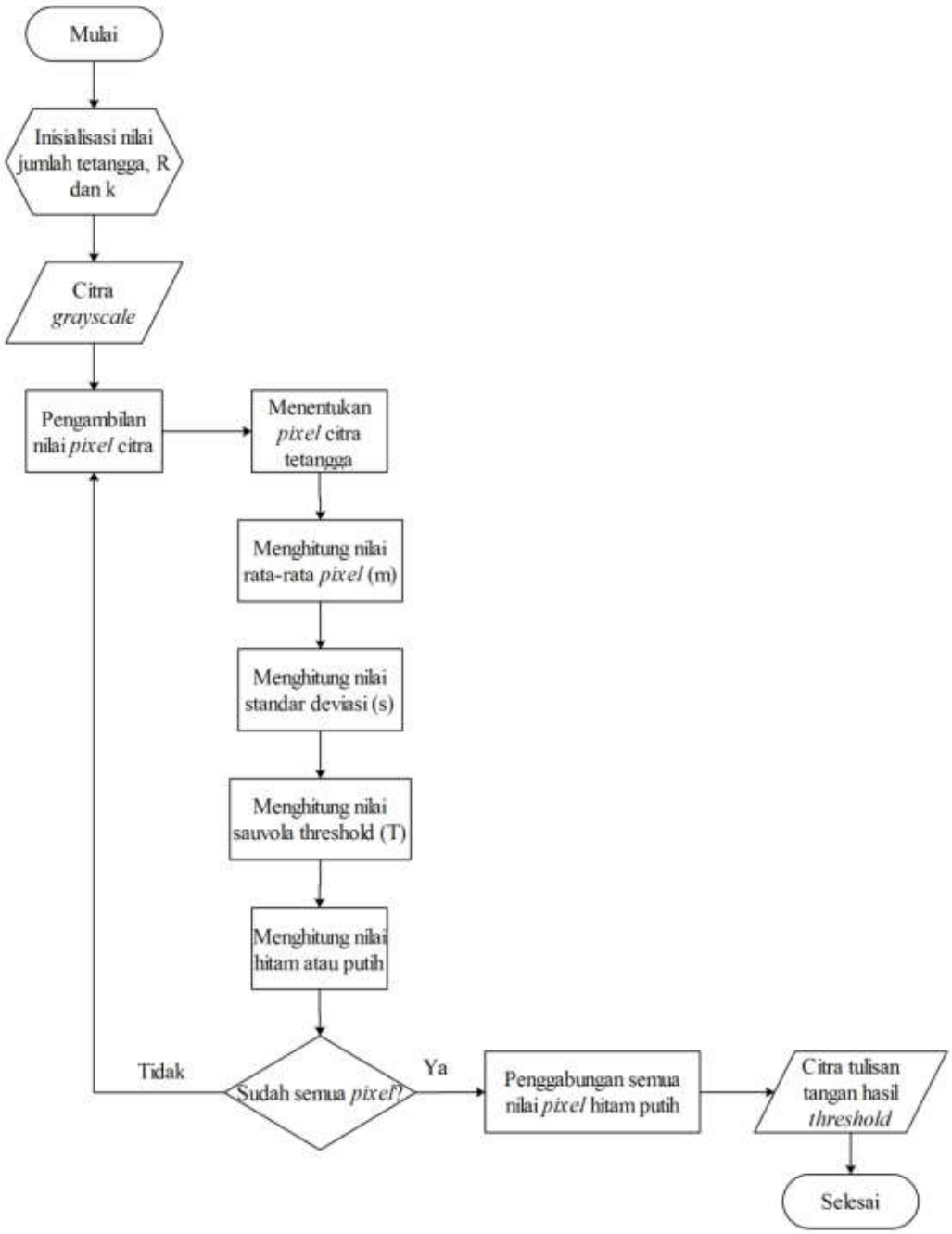
$$= 143{,}79 * (1 + 0{,}3 * (\frac{3{,}06}{128} - 1)) = 101$$

Dari perhitungan di atas didapatkan nilai ambang 101. Langkah selanjutnya langsung dimasukkan dalam persamaan 2.5 sehingga akan didapatkan nilai *pixel* baru. Dari contoh di atas maka *pixel* yang awalnya dengan nilai 144 akan berubah menjadi 255 karena 144 lebih besar dari 101. Berikut adalah tabel hasil perhitungan menggunakan *sauvola threshold.*

Tabel 3.5 Hasil Perhitungan *Sauvola Threshold*

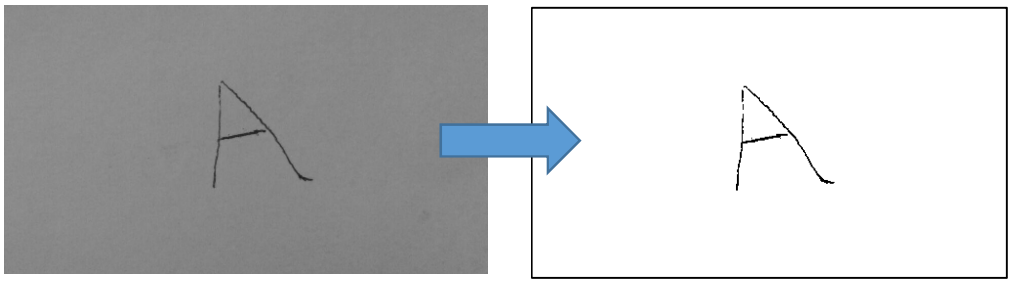
img(x,y)	**Nilai Gray**	**m(x,y)**	**s(x,y)**	**T(x,y)**	**Nilai Baru**
(0,0)	144	143,79	3,06	101	255
(1,0)	141	143,81	2,99	101	255
(2,0)	144	143,89	2,92	101	255
(3,0)	144	144,08	2,94	101	255
(4,0)	142	144,14	2,93	101	255
…	…	…	…	…	…
(98,120)	86	134,15	26,04	101	0
…	…	…	…	…	…
(749,1299)	85	87,28	3,25	61	255

Berikut adalah *flowchart* dari proses *sauvola threshold*.



Gambar 3.5 *Flowchart* proses *Sauvola Threshold*

Berikut adalah gambar citra yang sudah diproses dengan menggunakan metode *sauvola threshold*.



Gambar 3.6 Hasil Sauvola Threshold

Berikut adalah tabel matrik warna *Red* (R), *Green* (G) dan *Blue* (B) dari hasil proses *Sauvola Threshold*.

Tabel 3.6 Matriks Nilai RGB Citra *Threshold*

x/y	**0**	**1**	**2**	**3**	…	**749**
0	R = 255 G = 255 B = 255	R = 255 G = 255 B = 255	R = 255 G = 255 B = 255	R = 255 G = 255 B = 255	…	R = 255 G = 255 B = 255
1	R = 255 G = 255 B = 255	R = 255 G = 255 B = 255	R = 255 G = 255 B = 255	R = 255 G = 255 B = 255	…	R = 255 G = 255 B = 255
2	R = 255 G = 255 B = 255	R = 255 G = 255 B = 255	R = 255 G = 255 B = 255	R = 255 G = 255 B = 255	…	R = 255 G = 255 B = 255
3	R = 255 G = 255 B = 255	R = 255 G = 255 B = 255	R = 255 G = 255 B = 255	R = 255 G = 255 B = 255	…	R = 255 G = 255 B = 255
. . .	. . .	. . .	. . .	. . .	. . .	. . .
1299	R = 255 G = 255 B = 255	R = 255 G = 255 B = 255	R = 255 G = 255 B = 255	R = 255 G = 255 B = 255	…	R = 255 G = 255 B = 255

3.2.2.4 *Closing*

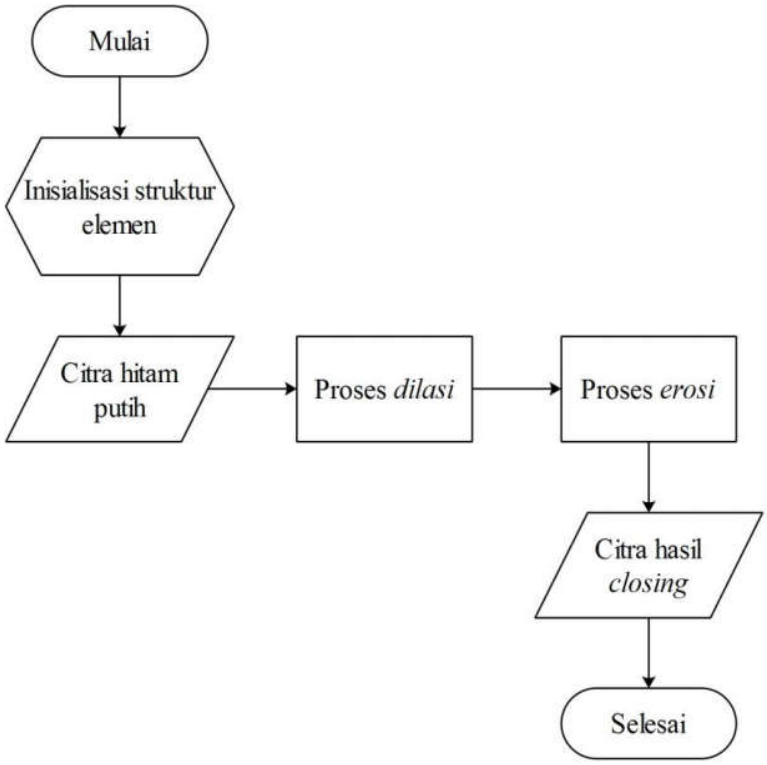
Proses *closing* berfungsi untuk menghitamkan bercak putih pada *foreground* citra. Proses ini dilakukan dengan dua proses yaitu proses *dilasi*

kemudian baru *erosi*. Dalam penelitian ini menggunakan Struktur Elemen (SE) 3x3. [17]



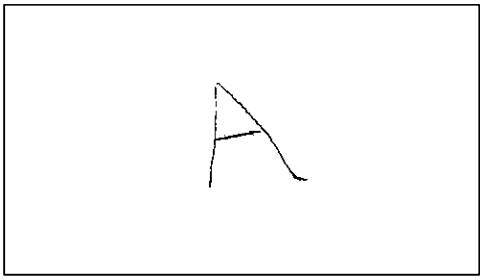
Gambar 3.7 Struktur Elemen 3x3

Berikut merupakan *flowchart* proses *closing.* [17]



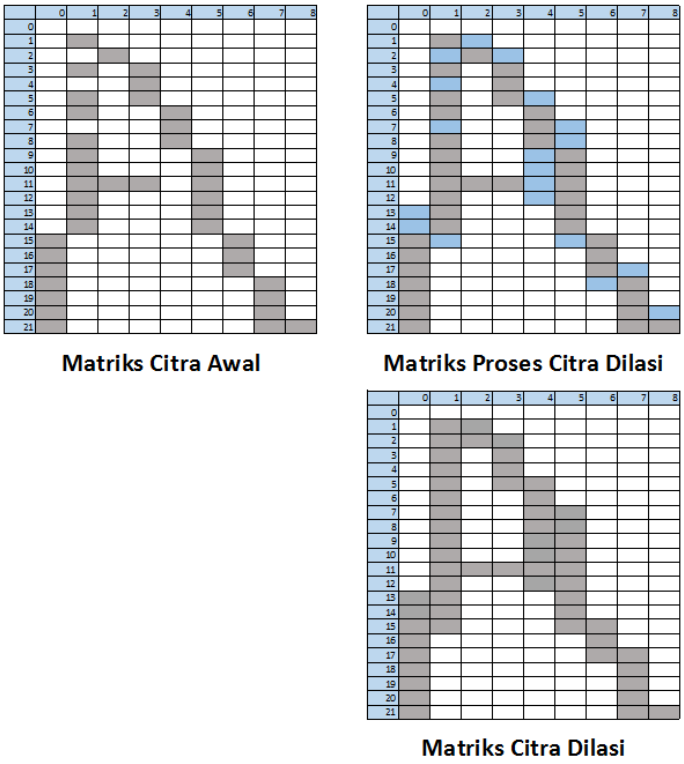
Gambar 3.8 *Flowchart* Proses *Closing*

Struktur elemen di atas akan digunakan dalam proses *closing* terhadap citra masukan. Citra yang dikenai operasi *closing* adalah citra yang sudah melalui proses *threshold*. Misalkan citra masukan yang digunakan seperti pada gambar di bawah ini.



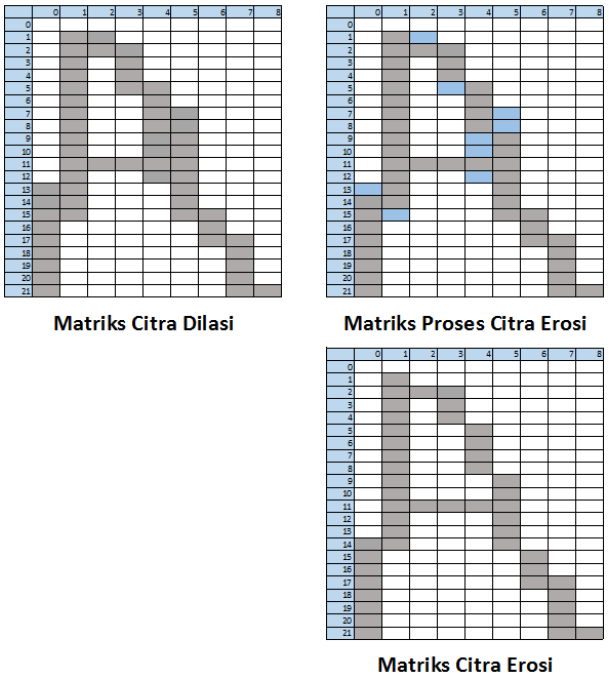
Gambar 3.9 Contoh Citra yang akan di *closing*

Sesuai urutan dalam proses *closing*, pertama dilakukan proses *dilasi* terlebih dahulu baru dilakukan proses *erosi*. Proses *dilasi* dilakukan dengan menandai *pixel* objek dengan struktur elemen yang digunakan. Gambar 3.10 merupakan potongan dari citra huruf A. Misalkan objek pada *pixel* (3,2) dikenakan operasi dilasi. Sesuai dengan struktur elemen yang digunakan, maka *pixel* (3,1), (2,2), (3,3) dan (4,2 berubah menjadi objek. Namun karena *pixel* (3,3) dan (4,2) merupakan objek, maka yang hanya berubah menjadi objek yaitu *pixel* (3,1) dan (2,2). Begitupun proses selanjutnya sampai semua *pixel* citra ditelusuri.



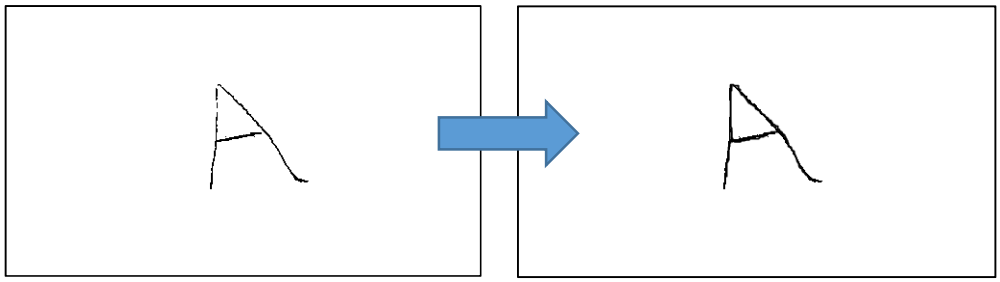
Gambar 3.10 Proses Dilasi

Langkah selanjutnya dilakukan proses *erosi* terhadap citra hasil *dilasi*. Sama dengan proses *dilasi* yaitu menandai *pixel* objek dengan struktur elemen yang digunakan. Namun dalam *erosi* jika *pixel* objek tidak memenuhi struktur elemen, maka *pixel* objek tersebut berubah menjadi *background*. Pada Gambar 3.11, *pixel* (3,1) berubah menjadi *background* karena tidak memenuhi struktur elemen, Sedangkan contoh pada *pixel* (3,2) tetap menjadi objek karena memenuhi struktur elemen yang digunakan. Begitupun proses selanjutnya sampai semua *pixel* citra ditelusuri.



Gambar 3.11 Proses Erosi

Hasil akhir proses *erosi* di atas merupakan hasil *closing* dari citra masukan sebelumnya. Gambar 3.12 merupakan citra setelah dilakukan proses *closing*.

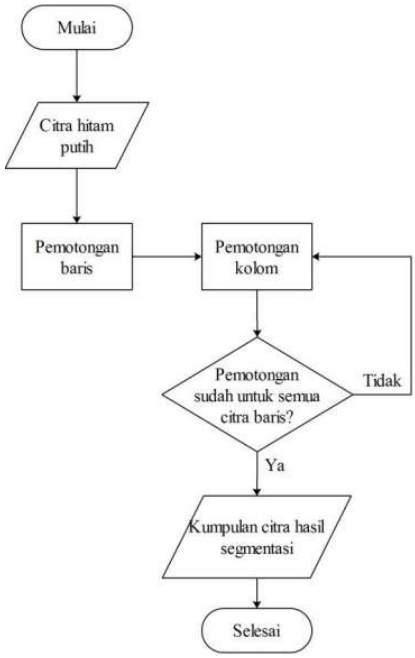


Gambar 3.12 Hasil Proses *Closing*

Dari proses *closing* yang sudah dilakukan terhadap citra input, Jika diperhatikan terlihat bahwa awalnya citra memiliki bercak putih di bagian tengah. Setelah dilakukan proses *closing* terlihat bintik putih tertutup oleh warna *foreground* atau objek.

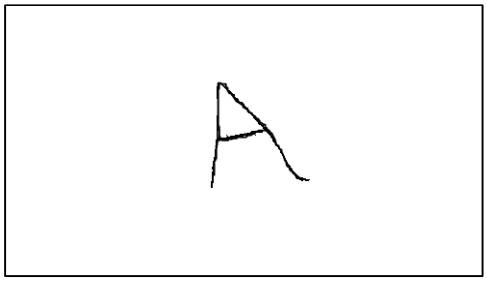
3.2.2.5 Segmentasi Citra

Pada proses ini, input yang digunakan yaitu citra hitam putih hasil *sauvola threshold* dan *closing.* Selanjutnya akan dilakukan pemotongan untuk mendapatkan citra huruf tulisan tangan. Berikut adalah *flowchart* proses segmentasi dalam gambar 3.13 [17]



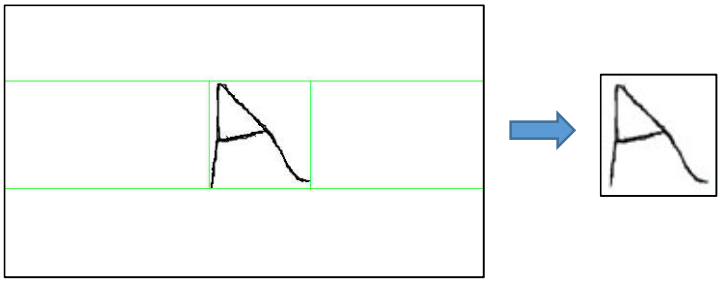
Gambar 3.13 *Flowchart* Proses Segmentasi

Pemotongan dilakukan untuk setiap baris (*vertical*) pada citra input terlebih dahulu, kemudian melakukan pemotongan setiap kolom (*horizontal*) pada setiap citra hasil pemotongan secara baris. Gambar berikut merupakan citra tulisan tangan yang sudah melalui proses *threshold* dan *closing* kemudian akan dilakukan proses segmentasi.



Gambar 3.14 Citra yang akan disegmentasi

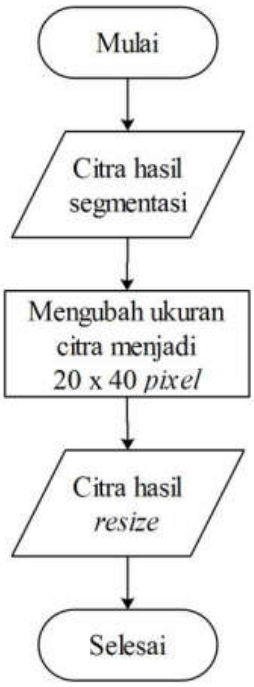
Pemotongan untuk setiap baris (*vertical*) dilakukan dengan menelusuri *pixel* citra dimulai dari *pixel* baris ke-0. Penelusuran terus dilakukan sampai menemukan *pixel* objek, kemudian tandai sebagai label awal pemotongan. Selanjutnya lakukan penelusuran kembali sampai dalam satu baris *pixel* citra tidak ditemukan *pixel* objek, kemudian tandai sebagai label akhir pemotongan. Label awal dan label akhir ini yang digunakan sebagai acuan untuk memotong citra setiap baris (*vertical*). Lakukan hal yang sama untuk pemotongan baris selanjutnya. Untuk pemotogan setiap kolom (*horizontal*) sama seperti pemotongan baris, hanya saja penelusuran citra dimulai dari *pixel* kolom ke-0. Berikut hasil segmentasi citra tulisan tangan.



Gambar 3.15 Hasil Segmentasi Citra

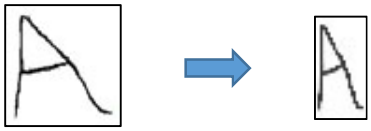
3.2.2.6 *Resize*

Resize artinya mengubah ukuran citra menjadi lebih besar jika ukuran citra awal lebih kecil dari ukuran yang ditetapkan atau menjadi lebih kecil jika ukuran citra awal lebih besar dari ukuran yang ditetapkan. Berikut adalah *flowchart* proses *resize.*



Gambar 3.16 *Flowchart* Proses *Resize*

Proses ini dilakukan agar input yang di proses pada saat metode klasifikasi seragam atau konsisten. Dalam penelitian ini karakter-karakter yang sudah tersegmen pada tahap segmentasi di-*resize* menjadi ukuran 20 x 40 *pixel,* sehingga aplikasi konsisten membaca sebanyak 800 *pixel*. Berikut merupakan contoh hasil dari proses *resize* dalam *pixel*. [17]

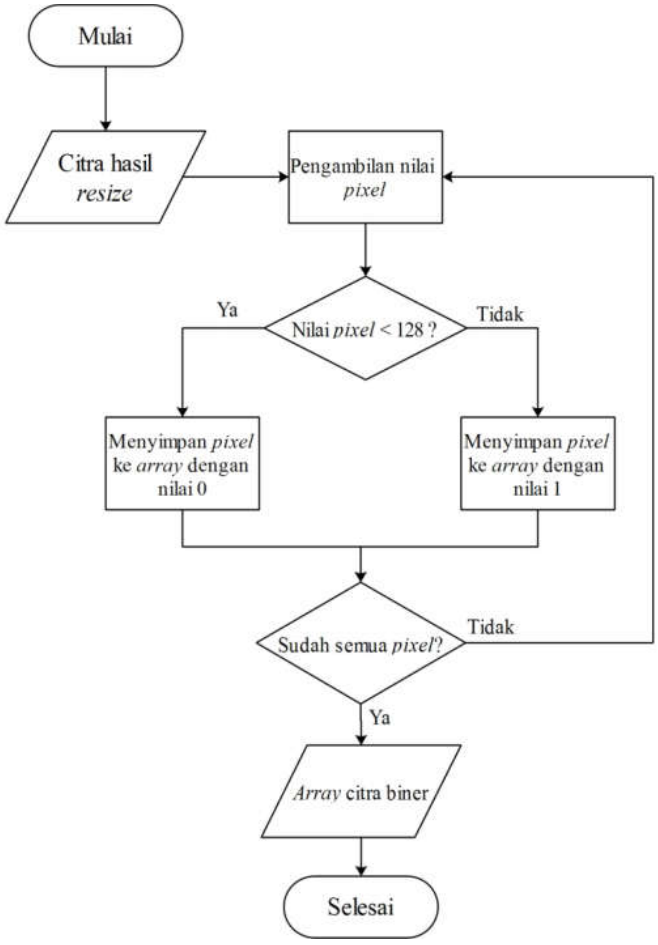


61x64 pixel 20x40 pixel

Gambar 3.17 Hasil Proses *Resize*

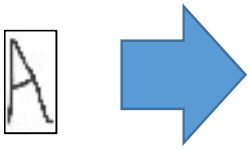
3.2.2.7 Binerisasi Citra

Proses binerisasi citra dilakukan untuk mendapatkan data biner dari citra huruf tulisan tangan yang sudah disegmentasi. Berikut merupakan *flowchart* dari proses binerisasi citra.



Gambar 3.18 Flowchart Proses Binerisasi

Data biner ini akan digunakan sebagai nilai masukan saat melakukan proses pelatihan maupun pengujian pada tahap klasifikasi. Dengan menggunakan persamaan 2.9 maka didapatkan hasil data biner sebagai berikut.

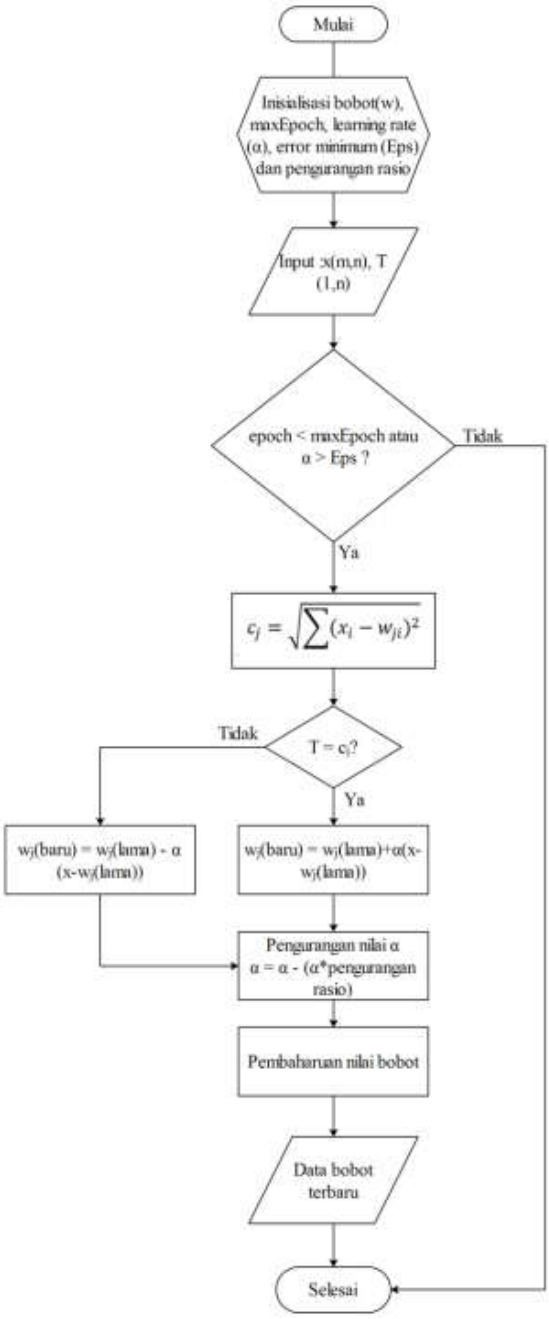


```
00000000000000000000
00000000000000000000
00000000000000000000
00110000000000000000
00111000000000000000
00101000000000000000
00101100000000000000
00100100000000000000
00100110000000000000
00100010000000000000
00100011000000000000
00100001000000000000
00100001000000000000
00100000100000000000
00100000110000000000
00100000011000000000
00100000001000000000
00100000001100000000
00100000001100000000
00100000111100000000
00100011100110000000
00111110000010000000
00111000000011000000
00100000000011000000
00100000000001000000
00100000000001000000
00100000000001000000
00100000000001100000
00100000000000100000
00100000000000110000
01100000000000110000
01100000000000010000
01100000000000010000
01000000000000011000
01000000000000001000
01000000000000001110
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
```

Gambar 3.19 Hasil Binerisasi Citra

3.2.2.8 Proses Pelatihan Metode *Learning Vector Quantization*

Metode klasifikasi yang digunakan dalam penelitian ini adalah *Learning Vector Quantization*. Gambar berikut merupakan *flowchart* dari algoritma proses pelatihan metode *Learning Vector Quantization* [28].



Gambar 3.20 Flowchart Proses Pelatihan Learning Vector Quantization

Gambar di atas merupakan *flowchart* proses pelatihan *Learning Vector Quantization*. Dalam pelatihan, citra yang digunakan sebagai data kelas dan data latih menggunakan citra hasil proses *preprocessing* pada tahap sebelumnya. Berikut adalah langkah-langkah secara lebih spesifik dalam proses pelatihan.

1. Tentukan bobot (w) awal yang merupakan kelas dari pelatihan. Bobot (w) merupakan deretan angka biner yang akan terus di-*update* selama proses pelatihan. Dalam penelitian ini menggunakan inisialisai bobot sebanyak 62 data citra huruf tulisan tangan yang terdiriri dari huruf A-Z, a-z, dan 0-9. Dalam contoh disini hanya ditampilkan 3 inisialisasi bobot.

Tabel 3.7 Inisialisasi Bobot Proses Pelatihan LVQ

Kelas	Citra	Bobot
A		w1
B		w2
C		w3

Tabel 3.7 merupakan data inisialisasi bobot yang nilai bobotnya nanti akan diupdate. w1,w2,w3 merupakan Bobot yang berisi deretan angka biner dari citra data kelas, kemudian A, B dan C merupakan kelas pelatihan.

2. Tentukan nilai inisialisasi untuk melakukan pelatihan data latih, yaitu nilai maksimum perulangan (*MaxEpoch)*, *error* minimum (Eps), *learning rate* (α) dan Pengurangan *learning rate*. Berikut adalah nilai inisialisasi awal yang digunakan.

Learning rate (α) = 0,00008
Pengurangan *learning rate* = 0,0002
Error minimum (*Eps*) = 0,00001
Maksimum Perulangan (*MaxEpoch*) = 500

Masukkan data training yang akan dilatih beserta target kelas pelatihannya. Dalam penelitian ini, data latih (m) menggunakan 100 sampel tulisan tangan, yang artinya ada 100 x 62 = 6.200 data latih.

3. Misalkan data latihnya adalah citra x(1,800) dengan target kelas pelatihan adalah kelas A.



Gambar 3.21 Data Latih x(1,800)

4. Lakukan perulangan untuk memperbaharui nilai bobot data kelas. Perulangan terus dilakukan sampai memenuhi salah satu parameter antara perulangan sudah mencapai MaxEpoch atau *learning rate* sudah mencapai *error minimum.*
5. Lakukan perhitungan untuk setiap data latih (x) terhadap setiap data bobot (w) menggunakan persamaan 2.10. Perhitungan ini dilakukan untuk mendapatkan jarak *euclidean*(C_j) minimum dengan data :

   m = 6.200 (banyaknya data latih)

   n = 800 (didapat dari perhitungan 20 x 40 = 800 yang merupakan ukuran dari citra input data training)

   $$cj = \sqrt{\sum_{i=1}^{i=800}(x_i \quad w_{ji})^2}$$

   $$(w1) : c1 = \sqrt{(0 \quad 0)^2 + (0 \quad 0)^2 + .. + (0 \quad 0)^2 + (0 \quad 0)^2} = 206{,}0$$

   $$(w2) : c2 = \sqrt{(0 \quad 0)^2 + (0 \quad 0)^2 + .. + (0 \quad 0)^2 + (0 \quad 0)^2} = 307{,}0$$

   $$(w3) : c3 = \sqrt{(0 \quad 0)^2 + (0 \quad 0)^2 + .. + (0 \quad 0)^2 + (0 \quad 0)^2} = 300{,}0$$

6. Setelah data latih x sudah dihitung terhadap data bobot w, selanjutnya tentukan jarak minimum dengan melihat nilai c terkecil. Dari contoh diatas jarak *euclidean* (c_j) minimum adalah c_1 dimana kelasnya adalah A. Langkah berikutnya lihat apakah data target kelas masukan x1 adalah A juga. Jika sama, maka gunakan persamaan 2.11 untuk melakukan perhitungan update bobot, sedangkan jika tidak sama gunakan persamaan 2.12. Karena target

kelas latih x1 adalah A sama dengan bobot (w1) adalah A juga, maka persamaan yang digunakan adalah 2.11. Sehingga perhitungannya menjadi

$w_j(baru) = w_j(lama) + \alpha[x - w_j(lama)]$

w1.1 = 0 + 0,00008(0-0) = 0

w1.2 = 0 + 0,00008 (0-0) =0

……

w1.799 = 0 + 0,00008 (0-0) = 0

w1.800 = 0 + 0,00008 (0-0) = 0

7. Setelah semua data latih (x) sudah dihitung terhadap setiap data bobot(w), selanjutnya kurangi nilai *learning rate* (α).

   α = 0,00008 – (0,0002 * 0,00008) = 0,000799984

8. Lakukan perulangan sampai memenuhi kondisi pada bagian 4 sehingga mendapatkan data nilai bobot akhir.

3.2.2.9 Proses Pengujian Metode *Learning Vector Quantization*

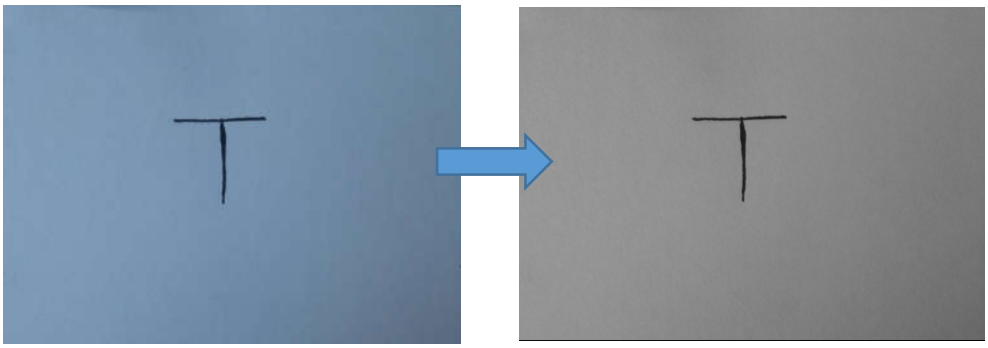
Sebelum masuk dalam tahap utama pengujian LVQ, pada Citra tulisan dilakukan proses *preprocessing* sama seperti tahap pelatihan. Berikut adalah contoh citra tulisan yang diproses sampai mendapatkan deretan angka biner setiap karakter.

1. Contoh citra tulisan tangan



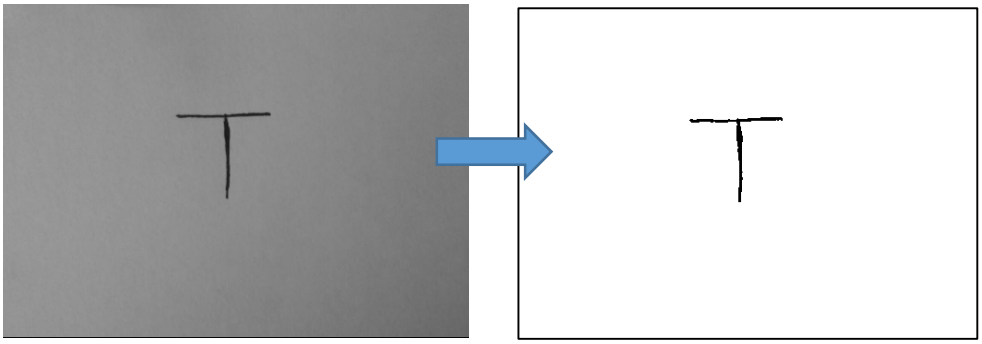
Gambar 3.22 Contoh Citra

2. Proses *grayscale* terhadap citra



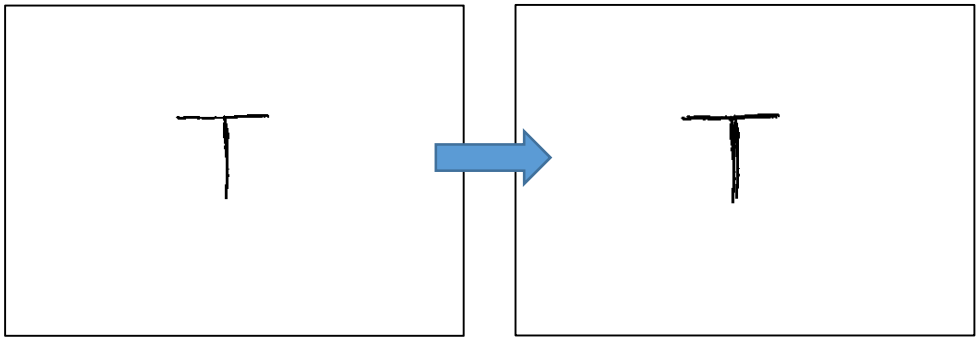
Gambar 3.23 Contoh Citra Hasil Grayscale

3. Citra hasil *grayscale* kemudian dilakukan *threshold* untuk menjadikan citra menjadi hitam putih



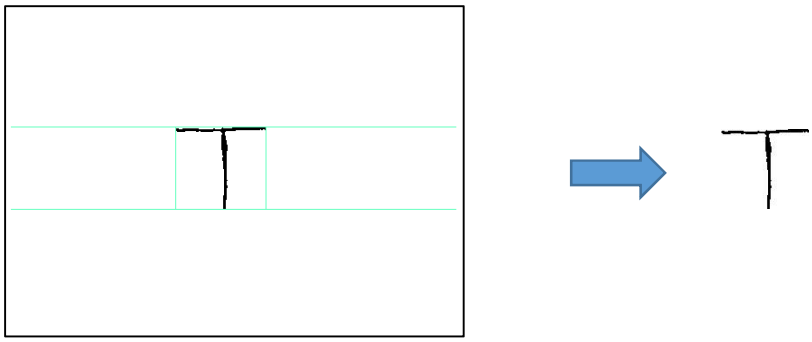
Gambar 3.24 Hasil Proses *Sauvola Threshold* pada Citra

4. Selanjutnya dilakukan proses *closing*



Gambar 3.25 Hasil Proses *Closing* pada Citra

5. Masuk ke tahap segmentasi dimana citra dipotong menjadi per karakter.



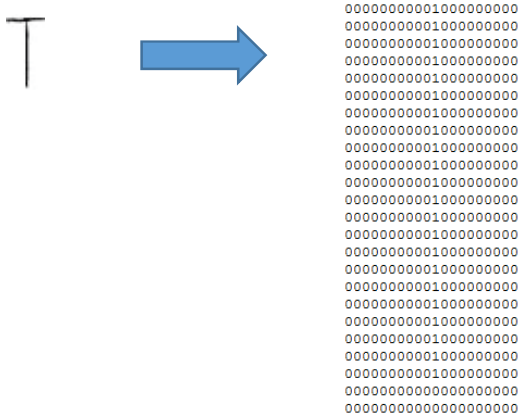
Gambar 3.26 Hasil Segmentasi Citra

6. Selanjutnya setiap citra hasil segmentasi, diresize menjadi 20x40 *pixel.*



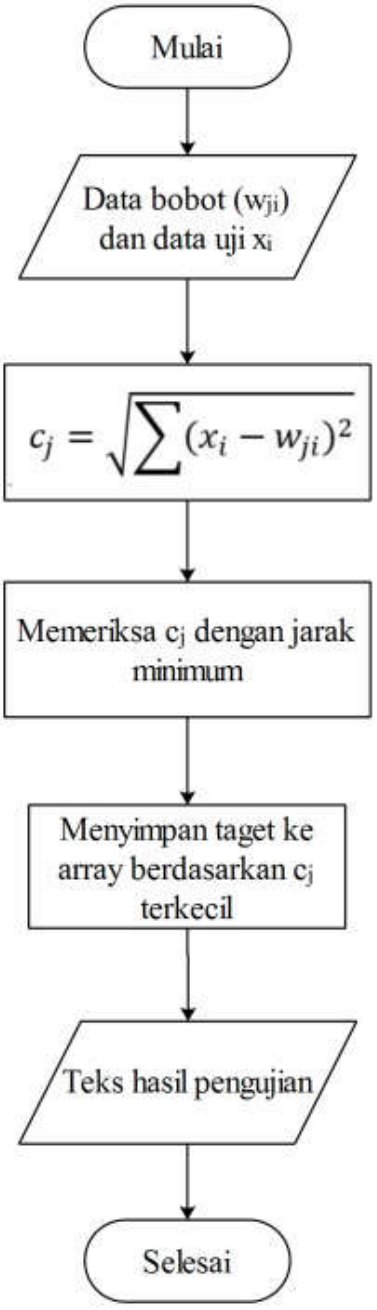
Gambar 3.27 Hasil Proses *Resize* Karakter

7. Terakhir melakukan proses binerisasi untuk setiap karakter hasil segmentasi dan *resize.*



Gambar 3.28 Hasil Proses Binerisasi Karakter

Proses pengujian pada metode *Learning Vector Quantization* menggunakan persamaan yang sama pada saat melakukan pelatihan. Hanya saja dalam pengujian tidak melakukan *update* data bobot. Berikut merupakan *flowchart* dari langkah-langkah pengujian LVQ.



Gambar 3.29 *Flowchart* Pengujian LVQ

Citra karakter yang akan diuji adalah citra hasil segmentasi. Berikut adalah langkah-langkah yang dilakukan dalam pengujian metode *Learning Vector Quantization* :



Gambar 3.30 Citra Karakter yang Diuji

1. Lakukan perhitungan untuk mencari jarak *euclidean* (c_j) minimum dari data uji (x) terhadap semua data bobot (w) menggunakan persamaan 2.10 dengan data n = 800 (didapat dari 20 x 40 yang merupakan ukuran pixel citra). Hitung untuk setiap data bobot w (dalam penelitian ini ada 62 data kelas yang artinya ada 62 data bobot) :

   Bobot ke -1 (w1)

   $$cj = \sqrt{\sum_{i=1}^{i=800}(x_i \quad w_{ji})^2}$$

   $$c1 = \sqrt{(0 \quad 3.164)^2 + (0 \quad 2.59)^2 + .. + (0 \quad 0)^2 + (0 \quad 0)^2}$$

   $$= 37{,}23$$

   Lakukan perhitungan terhadap seluruh data bobot (sampai w62).
2. Setelah semua nilai c dihitung, kemudian lihat nilai c_j terkecil atau dengan jarak terpendek.
3. Dari hasil perhitungan nilai bobot, didapat nilai c_j terkecil adalah c_{30} dengan kelasnya adalah T, maka citra uji dikenali sebagai huruf T.
4. Dari proses pengujian yang sudah dilakukan, didapatkan teks hasil pengenalan yaitu “T”.

3.2.2.10 Analisis Hasil Pengujian

Dari teks hasil pengujian yang didapatkan, terdapat huruf yang tidak dikenali dengan baik. Berikut adalah tabel hasil pengujian yang dilakukan.

Tabel 3.8 Analisis Hasil Pengujian

Citra Uji	Kelas	Hasil Pengenalan	Keterangan
	T	T	Sama
	a	a	Sama
	k	K	Tidak Sama
	d	d	Sama
	h	h	Sama
	t	t	Sama
	i	i	Sama
	o	o	Sama
	n	n	Sama
	l	l	Tidak Sama

Dari tabel analisis pengujian di atas, didapatkan hasil sebagai berikut.

a) Jumlah karakter sama = 8
b) Jumlah karakter Tidak sama = 2
c) Jumlah keseluruhan karakter = 10

Kemudian untuk menghitung nilai akurasi pengenalannya digunakan persamaan 2.15, sehingga hasilnya sebagai berikut.

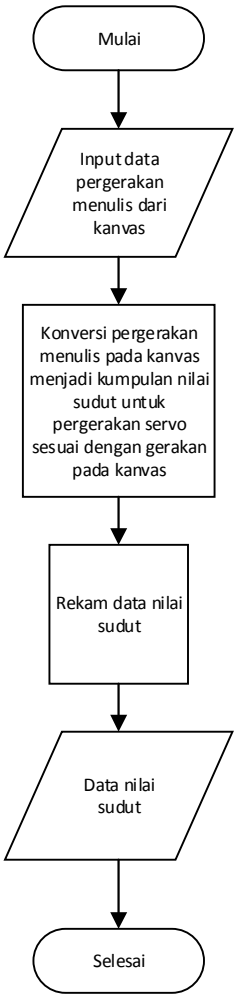
$$\text{Akurasi} = \frac{Jumla \quad Karakter\ sama}{Jumla \quad karakter\ keseluru \quad an} \ x\ 100\%$$

$$= \frac{8}{10}\ x\ 100\% = 80\ \%$$

Dari hasil perhitungan akurasi di atas, maka akurasi pengenalan yang didapatkan adalah 80%. Nilai akurasi ini merupakan hasil yang didapatkan dari pelatihan dan pengujian yang dilakukan pada tahap sebelumnya. Nilai akurasi ini bisa berubah jika pelatihan yang dilakukan menggunakan inisialisasi parameter yang berbeda.

3.2.2.11 Proses Rekam Nilai Sudut Untuk Pergerakan Servo

Proses rekam diperlukan untuk mendapatkan nilai sudut yang diperlukan dalam menulis huruf / karakter oleh lengan robot. Berikut adalah flowchart untuk merekam data nilai sudut :



Gambar 3.31 alur proses perekaman data

Dari hasil perhitungan invers kinematika yang dilakukan diperoleh nilai-nilai Θ_1 dan Θ_2. Berikut contoh rekam data nilai sudut untuk tiap derajat pergerakan yang akan diinstruksikan pada servo untuk membentuk huruf A :

```
'M',31,114,'U',0,0,'M',31,114,'M',30,113,'M',29,110,'M',26,106,'M'
,25,104,'M',24,102,'M',23,100,'M',22,99,'M',21,97,'M',19,95,'M',17
,91,'M',15,88,'M',14,87,'M',13,85,'M',12,84,'D',0,0,'M',30,111,'U'
,0,0,'M',29,110,'M',28,108,'M',27,104,'M',26,102,'M',25,100,'M',24
,93,'M',23,92,'M',22,89,'D',0,0,'M',20,98,'U',0,0,'M',21,98,'M',27
,99,'D',0,0
```

Pada data tersebut diberikan inisialisasi data yang berguna untuk menentukan instruksi yang akan di proses oleh servo. Penjelasan mengenai insialisasi data adalah sebagai berikut :

- "I" : untuk inisialisasi dari data awal.
- "M" : menginstruksikan servo bergerak ke derajat yang ditentukan.
- "D" : menginstruksikan servo yang berfungsi sebagai *pen holder* untuk menurunkan spidol / pen.
- "U" : menginstruksikan servo yang berfungsi sebagai *pen holder* untuk menaikan spidol / pen.
- "N" : Debug data.

3.2.3 Analisis Keluaran

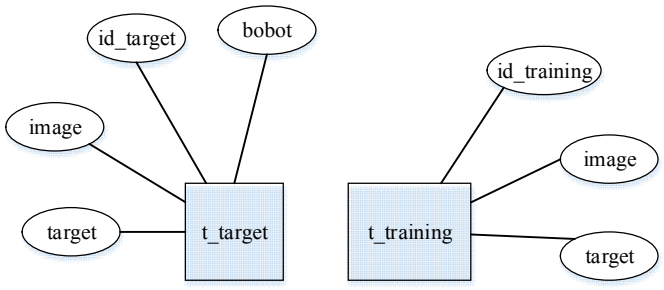
Analisis ini dilakukan untuk mengetahui data keluaran yang dihasilkan oleh sistem untuk pengguna. Data keluaran yang dihasilkan berupa huruf yang berhasil ditulis pada papan *white board.*

3.2.4 Analisis Basis Data

Dalam analisis basis data akan dibahas mengenai *Entity Relationship Diagram* (ERD) dan kamus data dari ERD tersebut.

3.2.4.1 *Entity Relationship Diagram* (ERD)

ERD menjelaskan hubungan antar data dalam basis data. berikut adalah ERD yang digunakan.



Gambar 3.32 Entity Relationship Diagram (ERD)

3.2.4.2 Kamus Data ERD

Kamus Data ERD menjelaskan atribut yang dimiliki dari setiap entitas yang ada. Berikut adalah tabel Kamus Data dari ERD sebelumnya.

Tabel 3.9 Kamus Data ERD

No.	Entitas	Atribut	Keterangan
1	t_target	id_target	Id dari kelas
		image	Nama gambar kelas
		target	Kelas pelatihan
		bobot	Bobot dari kelas
2	t_training	id_training	Id dari data latih
		image	Nama gambar data latih
		target	Nama data latih

3.2.5 Analisis Kebutuhan Non Fungsional

Merupakan penggambaran kebutuhan di luar sistem yang diperlukan untuk menjalankan sistem yang akan dibangun. Analisis yang dilakukan diantaranya adalah Analisis Kebutuhan Perangkat Keras (*Hardware*), Analisis Kebutuhan

Perangkat Lunak (*Software*), Analisis Pengguna dan Analisis Perancangan Kontrol utama.

3.2.5.1 Analisis Kebutuhan Perangkat Keras

Analisis kebutuhan yang diperlukan terbagi menjadi dua yaitu kebutuhan perangkat keras untuk *software* dan kebutuhan perangkat keras untuk pembangunan lengan robot. Berikut adalah spesifikasi perangkat keras minimum untuk software yang direkomendasikan :

1. *Processor* Dual Core 2.1 GHz
2. *Memory* RAM 512 MB

Spesifikasi perangkat keras yang diperlukan untuk pembangunan lengan robot pada penelitian ini adalah sebagai berikut :

a. Mikrokontroler Arduino UNO R3.
b. Motor servo.
c. Papan *white board* ukuran 60 x 40 cm.
d. Kabel *jumper male* dan *female.*

3.2.5.2 Analisis Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak yang diharapkan dapat memenuhi kebutuhan dalam menjalankan lengan robot ini adalah sebagai berikut :

Tabel 3.10 Spesifikasi Perangkat Lunak yang dibutuhkan

No	Jenis
1	Sistem Operasi Windows 7 Ultimate
2	Processing versi 3.1.1
3	Arduino IDE versi 1.6.9
4	Netbean IDE versi 8.1
5	JavaDevelopment Kit 8u66
6	Java Runtime Environment 8u66

3.2.5.3 Analisis Pengguna

Agar aplikasi dan lengan robot dapat dijalankan, maka dibutuhkan pengguna dengan kemampuan yang dapat menjalankan aplikasi yang dibangun dan dapat membangun lengan robot sesuai dengan kebutuhan. Kemampuan pengguna yang dibutuhkan yaitu :

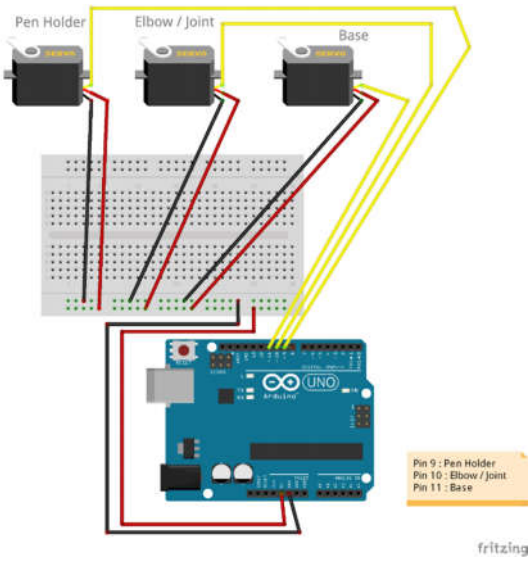
1. Dapat mengoperasikan komputer.
2. Mengerti tentang penggunaan *Xampp*.
3. Sedikitnya mengerti mengenai instalasi kelistrikan untuk memasang perangkat dan pengetahuan dasar untuk arduino.

3.2.5.4 Analisis Perancangan Kontrol utama

Rancangan untuk kontrol utama pada robot lengan terdiri dari:

- Arduino Uno R3
- Bread Board
- RC Servo

Servo terhubung dengan dengan mikrokontroler arduino Uno pada pin 9, 10, 11. Pin 9 terhubung dengan servo untuk *pen holder*, pin 10 terhubung dengan servo untuk *elbow* / *joint* dan pin 11 terhubung dengan servo untuk *base*.



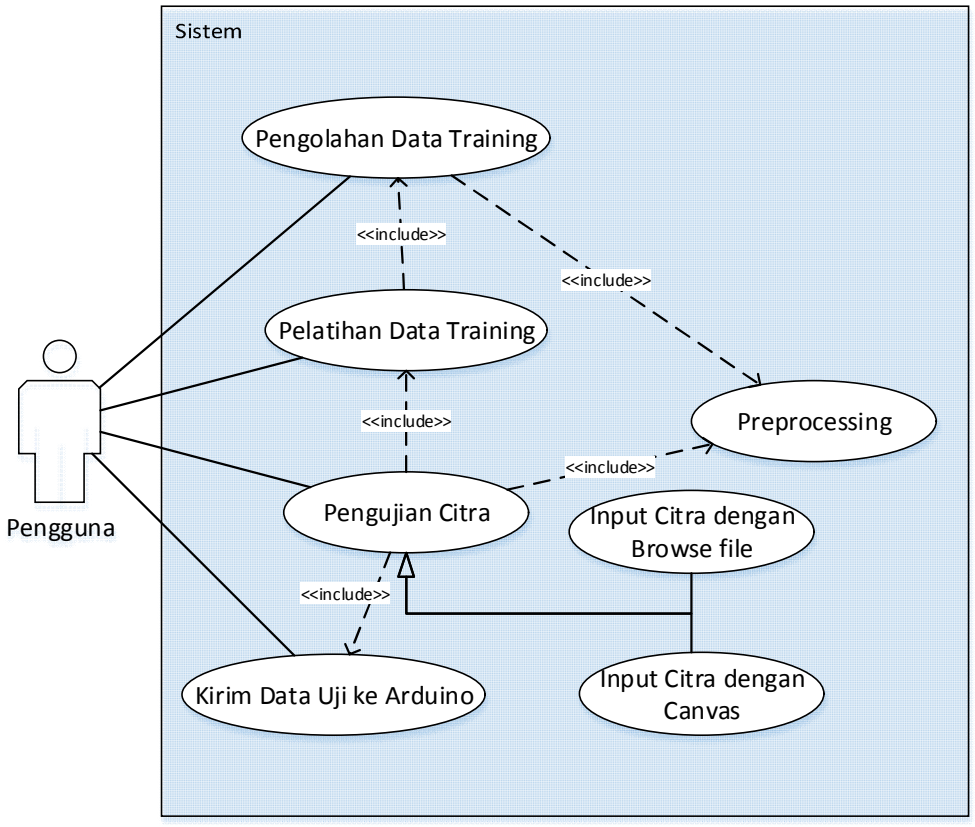
Gambar 3.33 Skema kontrol utama

3.2.6 Analsis Kebutuhan Fungsional

Analisis terhadap kebutuhan fungsional pada penelitian ini meliputi *use case diagram*, *use case scenario*, *activity diagram*, *sequence diagram* dan *class diagram* untuk analisis kebutuhan pada bagian *image processing*. Untuk bagian *hardware* pada arduino akan di jelaskan pada Diagram Konteks, *Data Flow Diagram*, dan spesifikasi Proses.

3.2.6.1 *Use Case Diagram*

Use case diagram merupakan pemodelan yang menunjukkan interaksi antara sistem dengan pengguna (aktor). Dalam *use case* yang dibuat menggunakan enam *use case*, dimana ada *use case* yang berelasi dengan *use case* lainnya. Relasi yang digunakan yaitu *include*. Berikut adalah *use case diagram* untuk aplikasi yang dibangun.



Gambar 3.34 *Use Case Diagram*

1. Definisi Aktor

   Aktor merupakan pihak yang mengakses *use case*. Berikut adalah deskripsi dari aktor.

Tabel 3.11 Deskripsi Aktor

No	Aktor	Deskripsi
1	Pengguna	Pengguna Aplikasi

2. Definisi *Use case*

   Use case menggambarkan apa saja yang dapat dilakukan oleh sistem. Berikut adalah deskripsi dari setiap *use case*.

Tabel 3.12 Deskripsi *Use Case*

No	Aktor	Deskripsi
1	Pengolahan Data *Training*	Proses pengolahan data kelas dan latih
2	Pelatihan Data *Training*	Proses hitung dan *update* data bobot kelas
3	Pengujian Citra	Proses input citra yang akan diolah
4	*Input* Citra dengan *Browse File*	Proses *input file* dengan metode *browse file*
5	*Input* Citra dengan *Canvas*	Proses *input file* dengan metode gambar pada *canvas*
6	Kirim Data Uji Ke Arduino	Proses kirim hasil pengujian pada arduino
7	*Preprocessing*	Proses pengolahan citra

3. *Use case Scenario*

   Use case scenario menjelaskan bagaimana setiap *use case* bekerja. Berikut adalah skenario dari setiap *use case* :

Tabel 3.13 *Use Case Scenario* Pengolahan Data *Training*

Nama *Use Case*	Pengolahan data *training*	
***Use Case* Terkait**	Pelatihan Data *Training*, *Preprocessing*	
Tujuan	Melakukan penambahan dan penghapusan data *training*	
Kondisi Awal	Data *training* sudah tersedia	
Kondisi Akhir	*Update* data *training* setelah melakukan penambahan atau penghapusan	
Aktor	Pengguna	
Skenario Utama (Penambahan data kelas)	**Langkah**	**Aksi**
	1	Pengguna memilih *input* citra
	2	Sistem menampilkan pilihan citra
	3	Pengguna memilih citra karakter tulisan tangan
	4	Sistem menampilkan citra yang dipilih pengguna
	5	Pengguna menginputkan karakter kelas kemudian memilih simpan
	6	Sistem menampilkan data kelas terbaru
Skenario Alternatif 1 (menghapus data kelas)	**Langkah**	**Aksi**
	1	Pengguna memilih data kelas yang akan dihapus kemudian memilih hapus
	2	Sistem melakukan proses hapus data kelas kemudian menampilkan data kelas terbaru
Skenario Alternatif 2 (menambahkan data latih)	**Langkah**	**Aksi**
	1	Pengguna memilih *input* citra
	2	Sistem menampilkan pilihan citra

	3	Pengguna memilih citra karakter tulisan tangan
	4	Sistem menampilkan citra yang dipilih pengguna
	5	Pengguna memilih target kelas data latih kemudian memilih simpan
	6	Sistem menampilkan data latih terbaru
Skenario Alternatif 3 (menghapus data latih)	**Langkah**	**Aksi**
	1	Pengguna memilih data latih yang akan dihapus kemudian memilih hapus
	2	Sistem melakukan proses hapus data latih kemudian menampilkan data latih terbaru

Tabel 3.14 *Use Case Scenario* Pelatihan Data *Training*

Nama *Use Case*	Pelatihan Data *Training*	
***Use Case* Terkait**	Pengolahan Data *Training*, Pengujian Citra	
Tujuan	Menghitung dan meng-*update* data bobot kelas	
Kondisi Awal	Data latih dan data kelas sudah tersedia	
Kondisi Akhir	Data bobot kelas berhasil dihitung dan diupdate	
Aktor	Pengguna	
Skenario Utama	**Langkah**	**Aksi**
	1	Pengguna Memasukkan inisialisasi (*learning rate*, *MaxEpoch*, *error minimum*, pengurangan)
	2	Pengguna memilih pelatihan
	3	Sistem melakukan perhitungan untuk mengupdate data bobot kelas, kemudian menampilkan pesan jika pelatihan sudah selesai
Skenario Alternatif 1 (Jika inisialisasi yang dimasukkan tidak lengkap)	**Langkah**	**Aksi**

	1	Pengguna Memasukkan inisialisasi (*learning rate*, *MaxEpoch*, *error minimum*, pengurangan)
	2	Pengguna memilih pelatihan
	3	Sistem menampilkan pesan kesalahan data inisialisasi tidak lengkap
	4	Pengguna mengisi data inisialisasi dengan benar kembali
	5	Pengguna memilih pelatihan
	6	Sistem melakukan perhitungan untuk mengupdate data bobot target, kemudian menampilkan pesan jika pelatihan sudah selesai
Skenario Alternatif 2 (Jika ingin melakukan pengolahan data training terlebih dahulu)	**Langkah**	**Aksi**
	1	<<include>>Pengolahan data training
	2	Pengguna Memasukkan inisialisasi (*learning rate*, *MaxEpoch*, *error minimum*, pengurangan)
	3	Pengguna memilih pelatihan
	4	Sistem melakukan perhitungan untuk meng-*update* data bobot kelas, kemudian menampilkan pesan jika pelatihan sudah selesai

Tabel 3.15 *Use Case Scenario* Pengujian Citra

Nama *Use Case*	Pengujian Citra
***Use Case* Terkait**	Pelatihan Data *Training*, *Preprocessing*
Tujuan	Menghitung dan menampilkan hasil konversi dari citra input menjadi teks digital
Kondisi Awal	Citra *input* belum di masukan dan teks hasil konversi belum ada.

Kondisi Akhir	Citra uji ditampilkan beserta hasil konversi teks digital	
Aktor	Pengguna	
Skenario Utama	**Langkah**	**Aksi**
	1	<<include>> *Preprocessing*
	2	Pengguna memproses tahapan dari *preprocessing*
	3	Sistem menampilkan hasil pengujian berupa teks digital

Tabel 3.16 *Use Case Scenario Input* Citra dengan *Browse File*

Nama *Use Case*	*Input* Citra dengan *Browse File*	
***Use Case* Terkait**	Pengujian Citra	
Tujuan	Proses input data citra dengan metode *browse file*	
Kondisi Awal	Citra belum diinputkan	
Kondisi Akhir	Citra telah diinputkan denga metode *browse file*	
Aktor	Pengguna	
Skenario Utama	**Langkah**	**Aksi**
	1	Pengguna memilih *file* citra yang akan diproses
	2	Sistem menerima *file* citra yang telah pengguna pilih

Tabel 3.17 *Use Case Scenario Input* Citra dengan *Canvas*

Nama *Use Case*	*Input* Citra dengan *Canvas*	
***Use Case* Terkait**	Pengujian Citra	
Tujuan	Proses *input* data citra dengan menggambar pada *canvas* yang disedikan	
Kondisi Awal	Citra belum diinputkan	
Kondisi Akhir	Citra telah digambar oleh pengguna pada *canvas*	
Aktor	Pengguna	
Skenario Utama	**Langkah**	**Aksi**
	1	Pengguna menggambar citra yang akan diproses pada *canvas*
	2	Sistem menerima *file* citra pada *canvas*

Tabel 3.18 *Use Case Scenario* Kirim Data Uji Ke Arduino

Nama *Use Case*	Kirim Data Uji Ke Arduino	
***Use Case* Terkait**	Pengujian Citra	
Tujuan	Mengirim data dari hasil pengujian citra yang berupa teks digital ke arduino	
Kondisi Awal	Teks digital belum dihasilkan	
Kondisi Akhir	Teks digital didapat dan dikirimkan ke arduino	
Aktor	Pengguna	
Skenario Utama	**Langkah**	**Aksi**
	1	Sistem memproses pengujian citra hingga proses selesai dengan menghasilkan teks digital
	2	Teks dikirimkan oleh sistem ke arduino

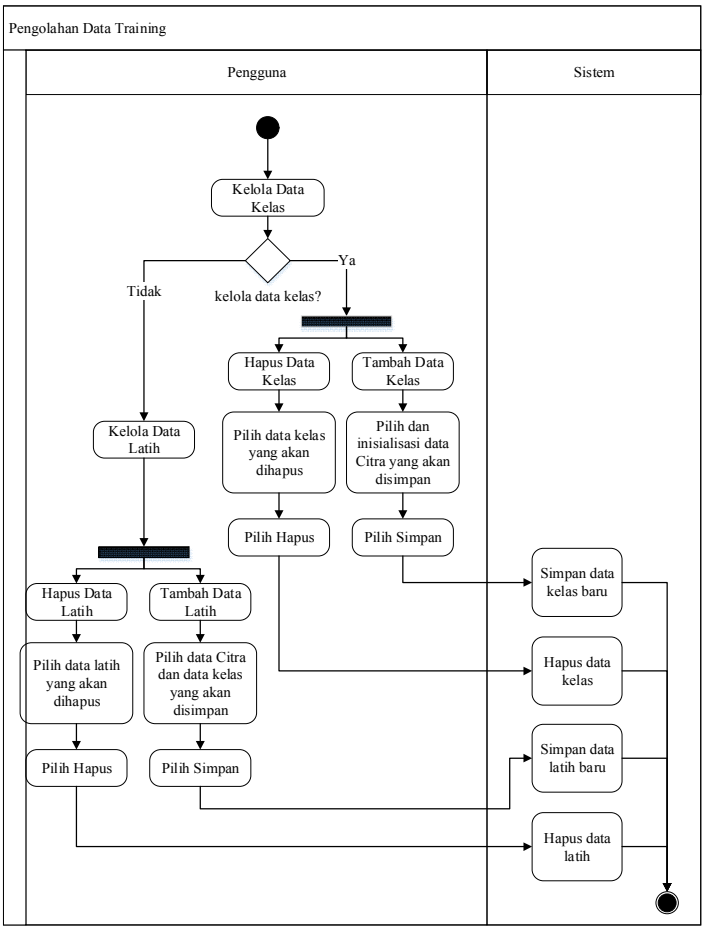
Tabel 3.19 *Use Case Scenario Preprocessing*

Nama *Use Case*	*Preprocessing*	
***Use Case* Terkait**	Pengolahan Data *Training*, Pengujian Citra	
Tujuan	Pengolahan data citra untuk kemudian data tersebut digunakan untuk pengujian	
Kondisi Awal	Citra yang akan diolah sudah diinputkan dalam aplikasi	
Kondisi Akhir	Citra sudah tersegmentasi	
Aktor	Pengguna	
Skenario Utama	**Langkah**	**Aksi**
	1	Pengguna memilih input citra
	2	Sistem menampilkan pilihan citra
	3	Pengguna memilih citra tulisan tangan
	4	Sistem menampilkan citra yang dipilih pengguna
	5	Pengguna memilih Grayscale
	6	Sistem menampilkan hasil *grayscale*
	7	Pengguna memilih Sauvola T.
	8	Sistem menampilkan hasil *sauvola thresholding*
	9	Pengguna memilih Closing

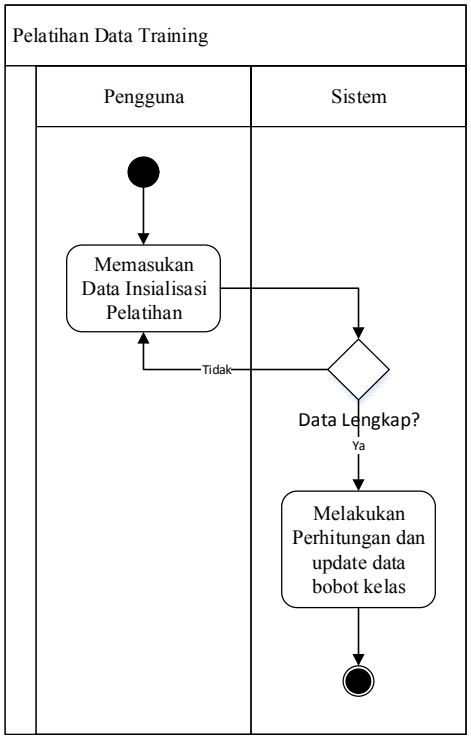
	10	Sistem menampilkan hasil *closing*
	11	Pengguna memilih Segmentasi
	12	Sistem menampilkan hasil segmentasi

3.2.6.2 *Activity Diagram*

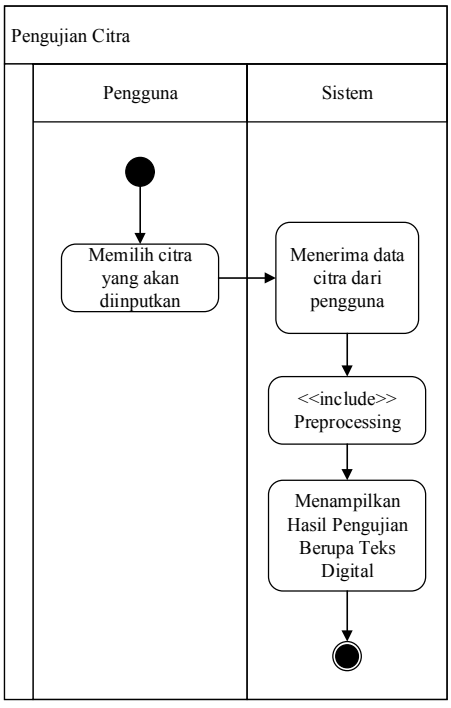
Activity diagram memodelkan bagaimana aliran *use case* bekerja dan dibuat berdasarkan *Use Case* pada *Use Case* diagram sebelumnya. Berikut adalah *activity diagram* dari *use case* sebelumnya.



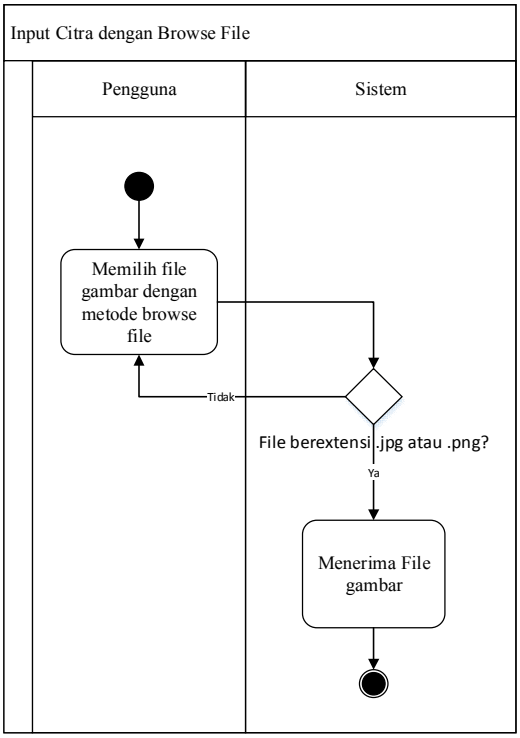
Gambar 3.35 *Activity Diagram* Pengolahan Data *Training*



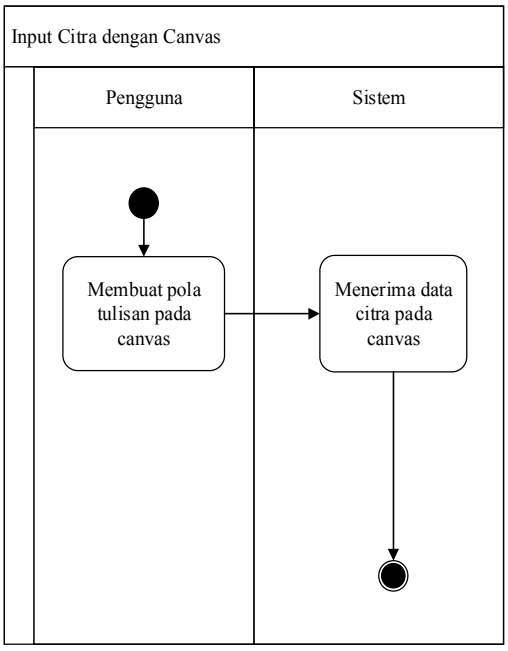
Gambar 3.36 *Activity Diagram* Pelatihan Data *Training*



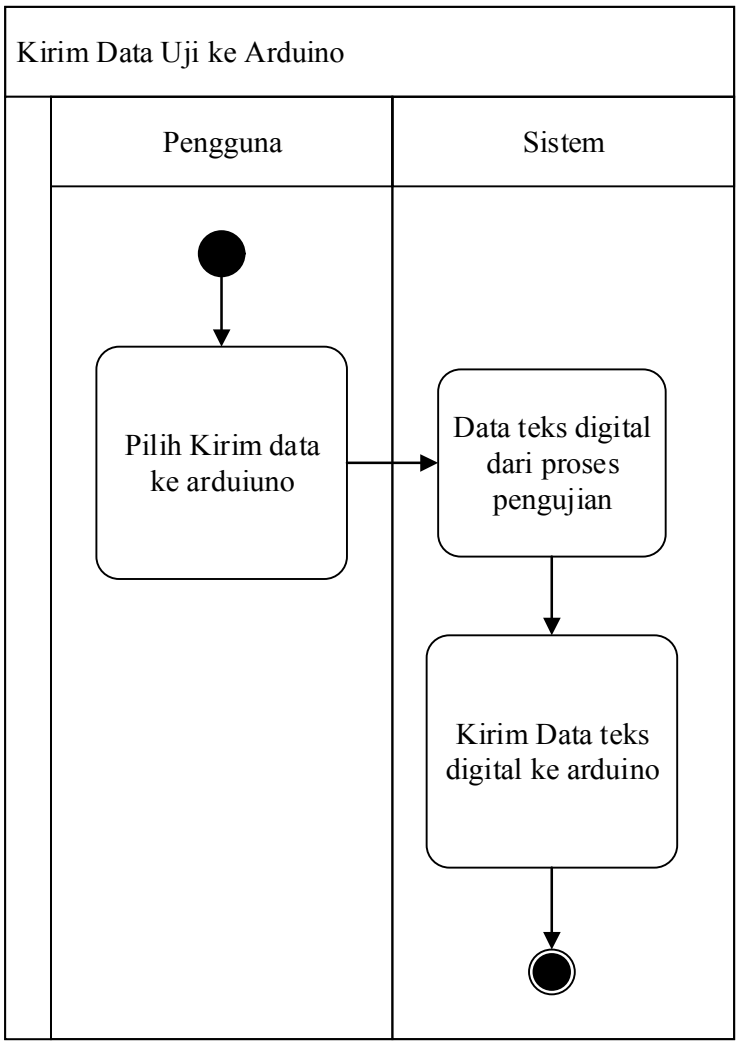
Gambar 3.37 *Activity Diagram* Pengujian Citra



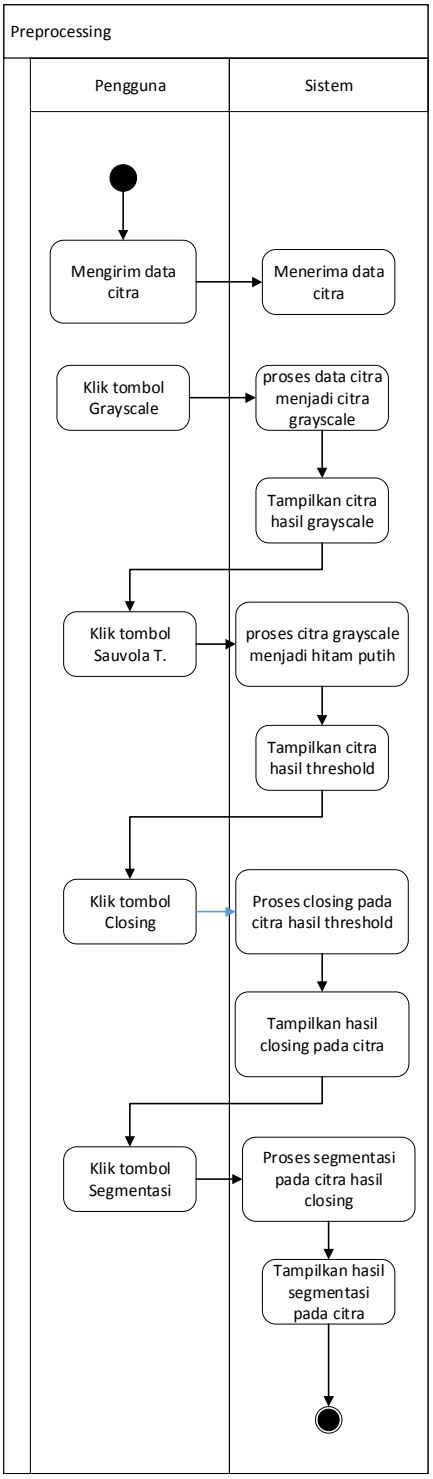
Gambar 3.38 *Activity Diagram* Input Citra dengan *Browse File*



Gambar 3.39 *Activity Diagram* Input Citra dengan *Canvas*



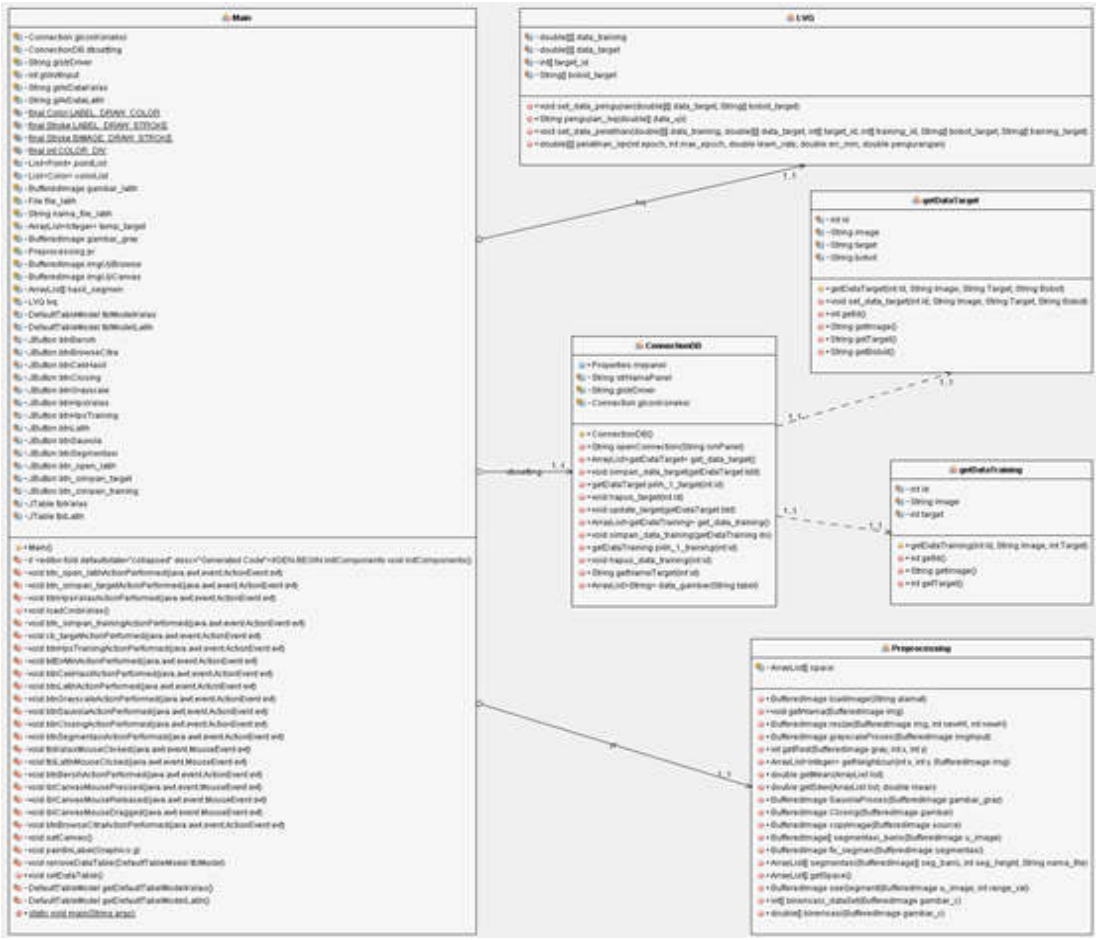
Gambar 3.40 *Activity Diagram* Kirim Data Uji ke Arduino



Gambar 3.41 Activity Diagram Prepocessing

3.2.6.3 *Class Diagram*

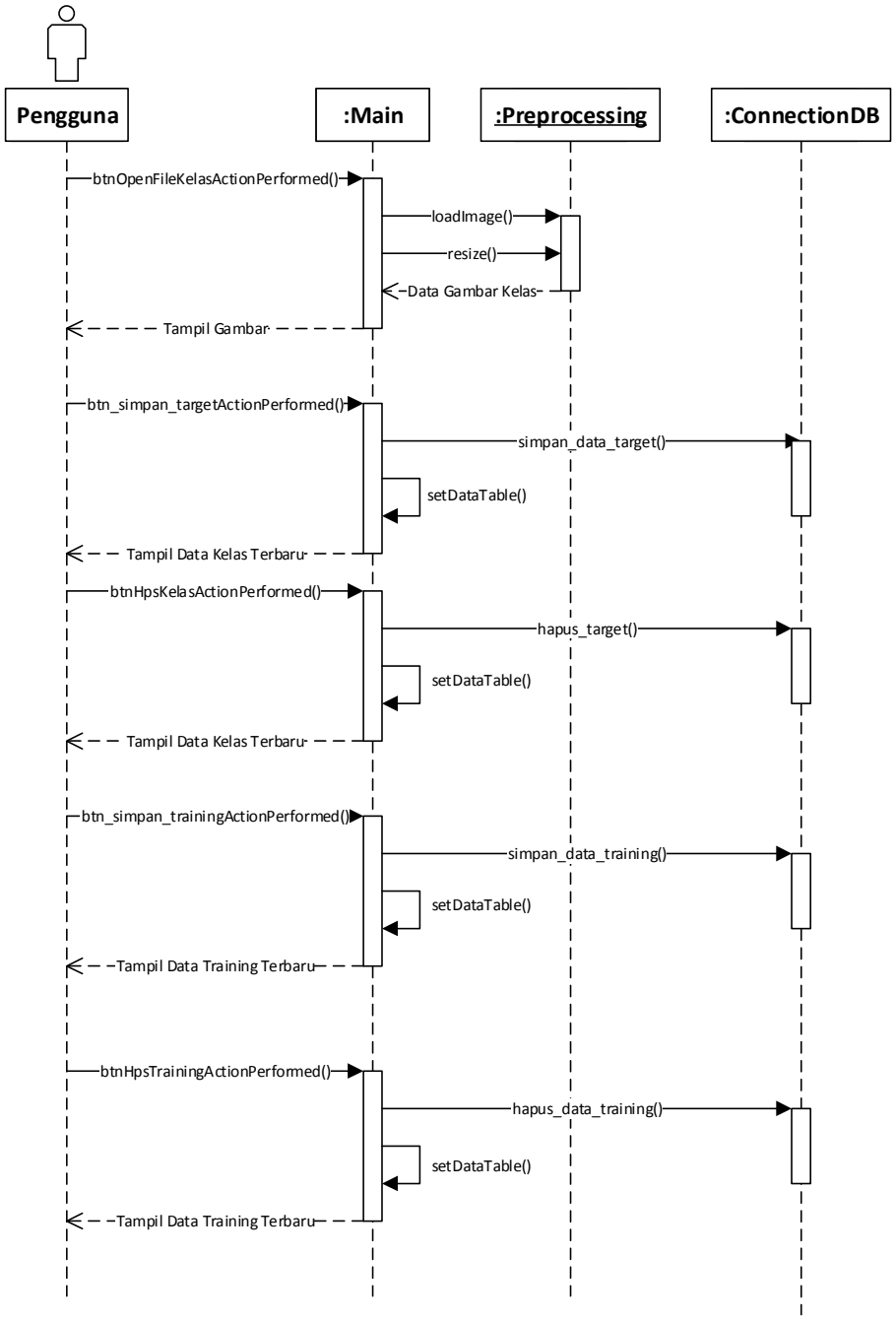
Class diagram menggambarkan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling sering ditemui dalam pemodelan sistem berorientasi objek. Berikut adalah *class diagram* dari sistem :



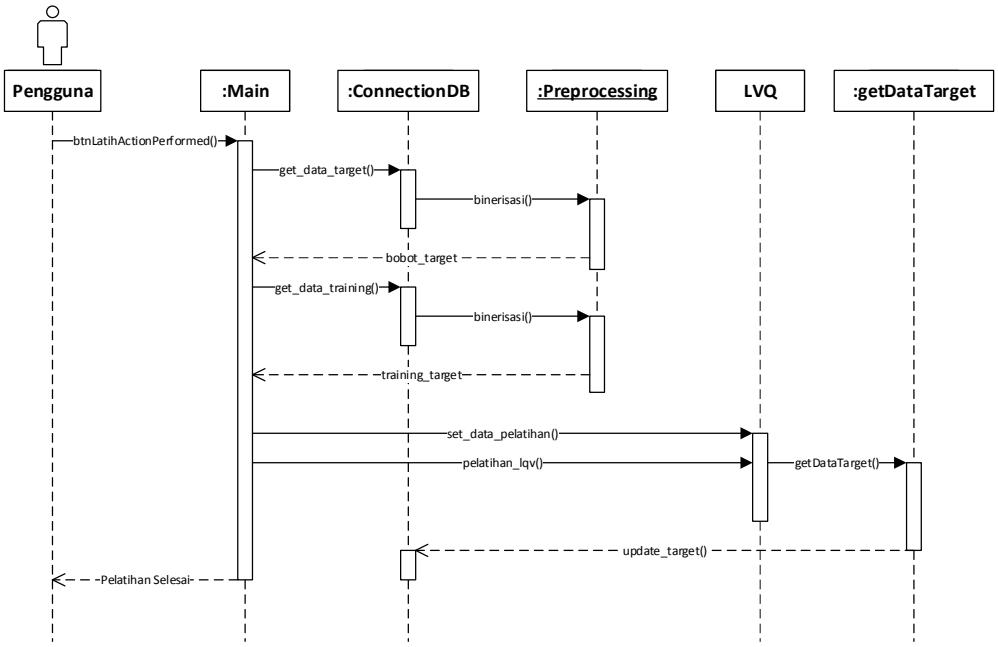
Gambar 3.42 Class Diagram

3.2.6.4 *Sequence Diagram*

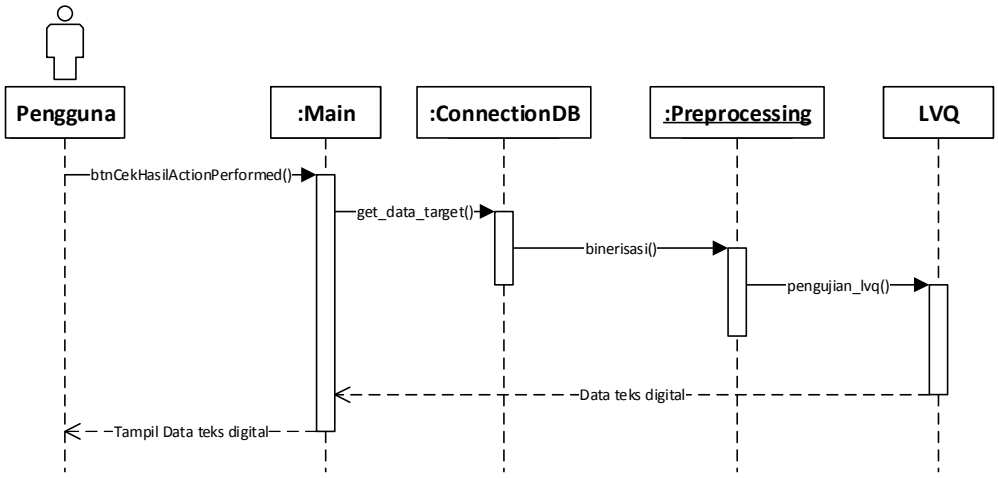
Sequence diagram memungkinkan melihat aliran pesan dari objek ke objek dalam urutan waktu. Berikut adalah *Sequence diagram* pada sistem ini.



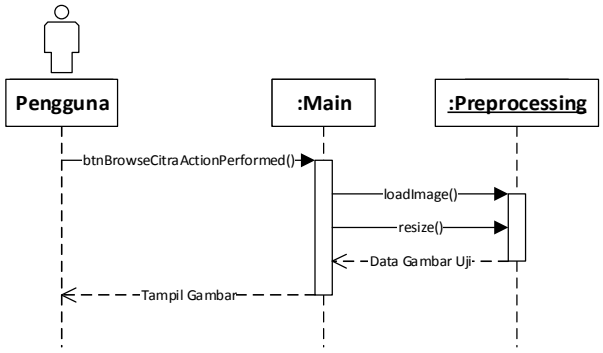
Gambar 3.43 *Sequence Diagram* Pengolahan Data *Training*



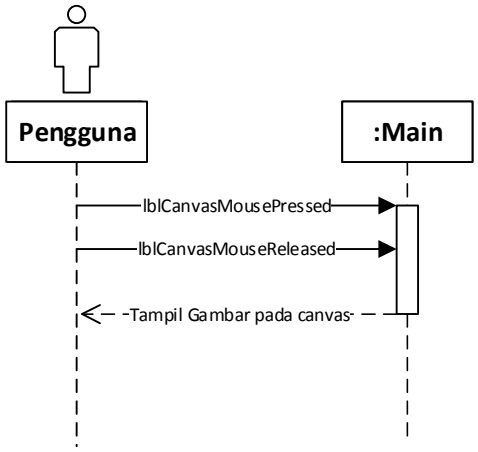
Gambar 3.44 Sequence Diagram Pelatihan Data Training



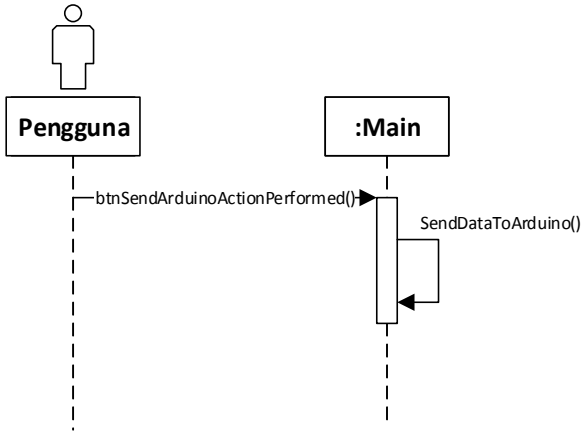
Gambar 3.45 *Sequence Diagram* Pengujian Citra



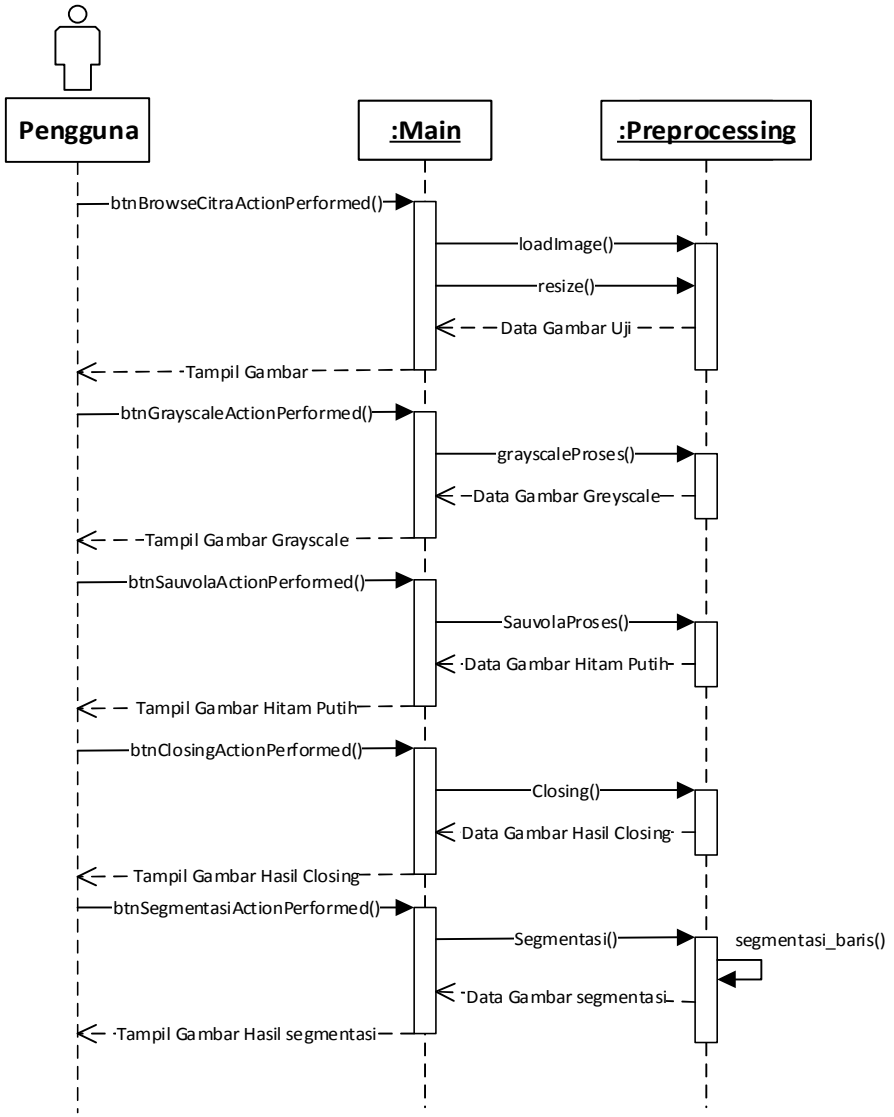
Gambar 3.46 *Sequence Diagram* Input Citra dengan *Browse File*



Gambar 3.47 *Sequence Diagram* Input Citra dengan *Canvas*



Gambar 3.48 *Sequence Diagram* Kirim Data Uji ke Arduino

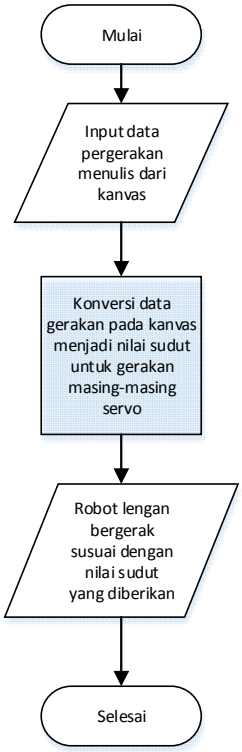


Gambar 3.49 *Sequence Diagram Preprocessing*

3.2.6.5 Perancangan Prosedural

Dalam perancangan prosedural akan dijelaskan mengenai proses untuk mendapat nilai sudut yang dibutuhkan untuk pergerakan servo dalam menulis setiap karakter huruf yang dilakukan lengan robot. Dijelaskan bahwa data gerakan pada kanvas akan dikonversi menjadi nilai sudut untuk masing-masing servo. Data tersebut akan disimpan yang nantinya akan dipanggil sesuai dengan data yang

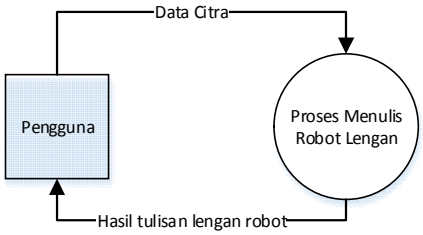
diperlukan untuk menulis karakter huruf. Berikut flowchart untuk rekam data nilai sudut :



Gambar 3.50 Proses rekam nilai sudut

3.2.6.6 Diagram Konteks

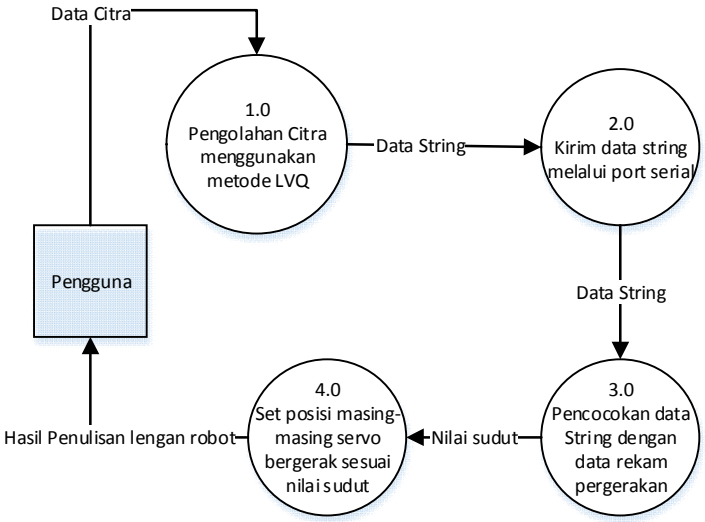
Diagram konteks adalah diagram yang mencakup masukan-masukan dasar, sistem umum dan keluaran, diagram ini merupakan tingkatan tertinggi dalam diagram aliran data dan hanya memuat satu proses, menunjukan sistem secara keseluruhan. Diagram konteks untuk proses dari robot lengan ini adalah sebagai berikut :



Gambar 3.51 Diagram Konteks

3.2.6.7 *Data Flow Diagram* Level 1

Berikut adalah Data Flow Diagram level 1 untuk proses perancangan robot lengan



Gambar 3.52 *Data Flow Diagram* Level 1

3.2.6.8 Spesifikasi Proses

Spesifikasi proses digunakan untuk menggambarkan proses diagram alir yang terdapat pada *data flow diagram*. Spesifikasi proses dari *data flow diagram* level 1 dijelaskan sebagai berikut:

Tabel 3.20 Spesifikasi Proses

No	Proses	Keterangan
1	No. Proses	1.0
	Nama	Pengolahan Citra menggunakan metode LVQ
	Source (sumber)	Pengguna
	Input	Data Citra
	Output	Data *String*

	Logika Proses	1. Data citra berupa gambar berextensi .jpg 2. Proses pengolahan citra dengan metode LVQ dengan hasil akhir yaitu berupa data *string* dari citra yang dimaksud oleh pengguna
No	**Proses**	**Keterangan**
2	No. Proses	2.0
	Nama	Kirim data *string* melalui *port serial*
	Source (sumber)	Proses : Pengolahan Citra menggunakan metode LVQ
	Input	Data *String*
	Output	Data *String*
	Logika Proses	1. Proses:Pengolahan Citra menggunakan metode LVQ 2. Data hasil pengolahan citra dikirim melalui *port serial.*
No	**Proses**	**Keterangan**
3	No. Proses	3.0
	Nama	Pencocokan data *String* dengan data rekam pergerakan
	Source (sumber)	Proses : Kirim data nilai sudut melalui *port serial*
	Input	Data *String*
	Output	Nilai sudut
	Logika Proses	1. Proses: Kirim data nilai sudut melalui port serial. 2. Menerima data String lalu cocokan data dengan data rekaman yang tersimpan pada arduino.
No	**Proses**	**Keterangan**
4	No. Proses	4.0

	Nama	Set posisi masing-masing servo bergerak sesuai nilai sudut
	Source (sumber)	Proses : Pencocokan data *String* dengan data rekam pergerakan
	Input	Nilai sudut
	Output	Gerakan menulis sesuai dengan data gerakan
	Logika Proses	1. Proses: Pencocokan data *String* dengan data rekam pergerakan 2. Set posisi masing-masing servo sesuai dengan data nilai sudut yang diberikan dan bergerak sesuai gerakan

3.2.6.9 Kamus Data

Kamus data merupakan deskripsi formal mengenai seluruh elemen yang tercakup dalam DFD. Kamus data untuk diagram alir data pada sistem tanya jawab akan dijelaskan pada tabel berikut.

Tabel 3.21 Kamus Data

Nama Aliran Data	**Data Citra**
Digunakan pada	Pengolahan Citra menggunakan metode LVQ Proses 1.0
Deskripsi	Berisi data citra dari pengguna berupa citra tulis berextensi .jpg
Struktur data	-
Nama Aliran Data	**Data *String***
Digunakan pada	- Kirim data *string* melalui *port serial* Proses 2.0 - Pencocokan data *String* dengan data rekam pergerakan Proses 3.0

Deskripsi	Berisi data String hasil dari proses pengolahan citra
Struktur data	-
Nama Aliran Data	**Nilai sudut**
Digunakan pada	Set posisi masing-masing servo bergerak sesuai nilai sudut Proses 4.0
Deskripsi	Berisi nilait sudut yang diperlukan arduino untuk menggerakan servo agar lengan robot menulis sesuai dengan data inputan dari pengguna
Struktur data	-

3.3 Perancangan Sistem

Dalam Perancangan sistem ini, dibuat beberapa perancangan seperti perancangan *database*, antar muka, pesan (*Alert*) dan jaringan semantik.

3.3.1 Perancangan *Database*

Perancangan *database* dalam penelitian ini dituliskan dalam bentuk skema relasi dan struktur tabel. Berikut merupakan penjelasan dari masing-masing perancangan.

3.3.1.1 Skema Relasi

Skema relasi menggambarkan bagaimana suatu data antar entitas saling berelasi. Pada rancangan *database* disini tidak ada relasi yang berkaitan antar tabel

3.3.1.2 Struktur Tabel

Struktur Tabel mendeskripsikan atribut-atribut lengkap yang dimiliki oleh setiap entias dari skema relasi. Berikut adalah struktur tabel untuk setiap entitas.

Tabel 3.22 Struktur Tabel Data t_target

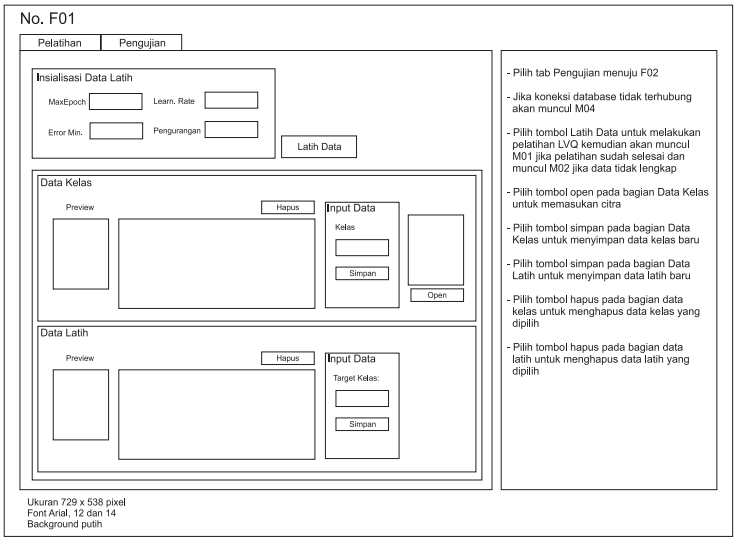
Tabel : t_target					
NO	Primary Key	Nama Atribut	Tipe Data	Panjang	Null
1	PK	id_target	int	11	Not Null
2	-	image	varchar	100	Not Null
3	-	target	char	1	Not Null
4	-	bobot	text	-	Not Null

Tabel 3.24 Struktur Tabel Data Latih

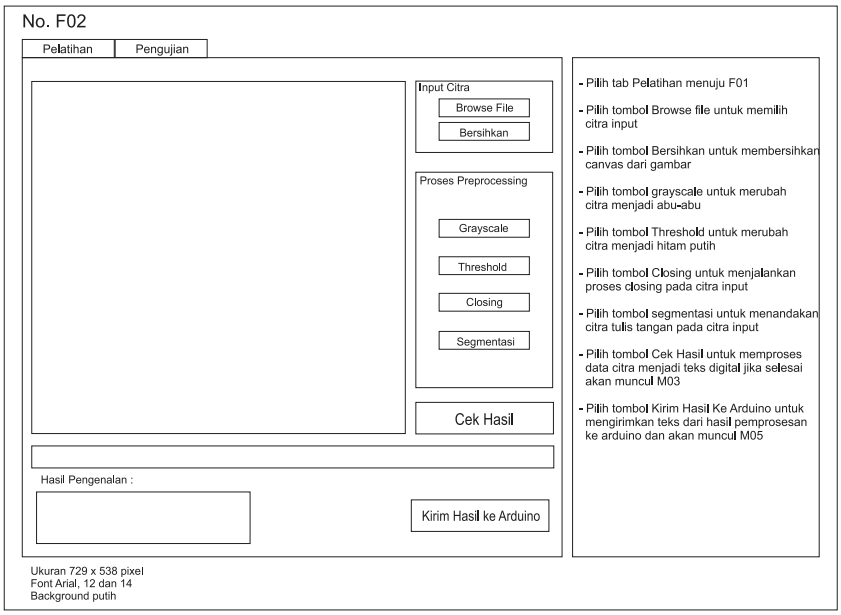
Tabel : t_training					
NO	Primary Key	Nama Atribut	Tipe Data	Panjang	Null
1	PK	id_training	int	11	Not Null
2	-	image	varchar	100	Not Null
3	-	target	int	11	Not Null

3.3.2 Perancangan Antar Muka

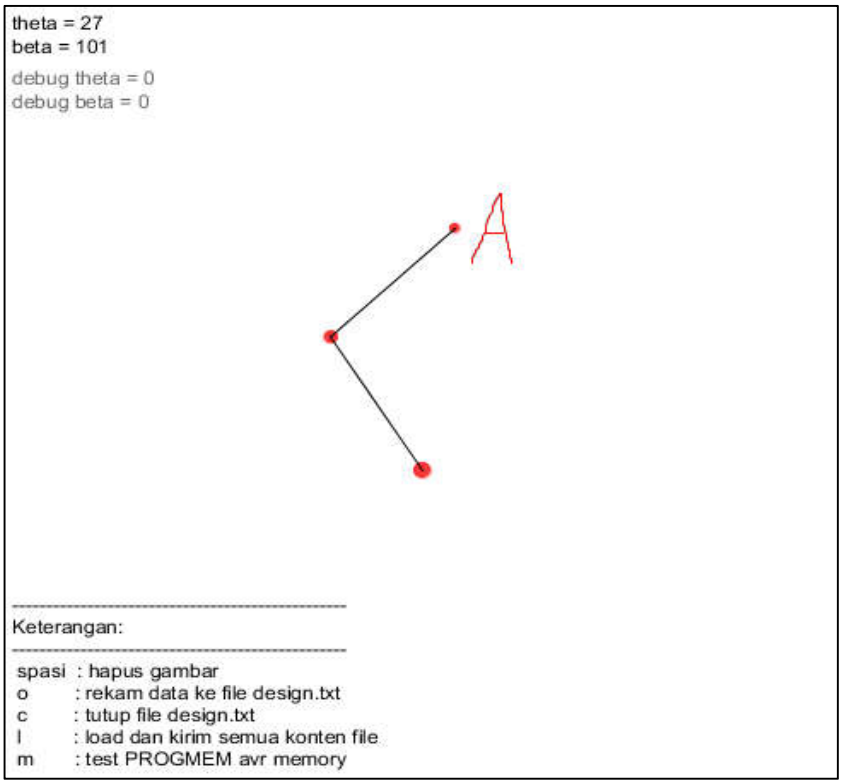
Perancangan antar muka mendeskripsikan rencana tampilan dari setiap tampilan yang akan digunakan pada penelitian ini. Perancangan antar muka ini terdiri dari perancangan form dan pesan.



Gambar 3.53 Perancangan Antarmuka Menu Pelatihan



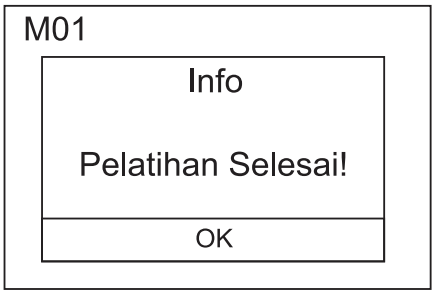
Gambar 3.54 Perancangan Antarmuka Menu Pengujian



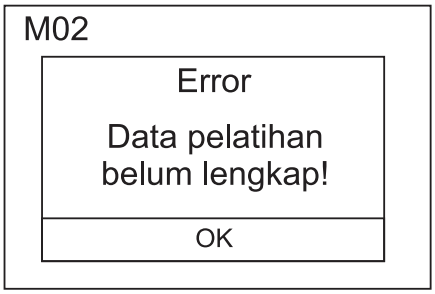
Gambar 3.55 Perancangan Antarmuka Menu Rekam Data Nilai Sudut

3.3.3 Perancangan Pesan

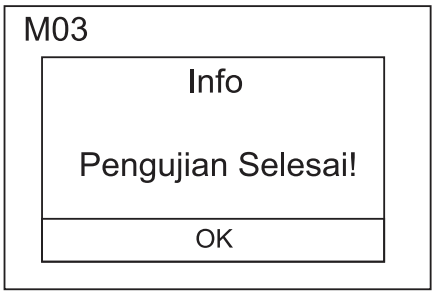
Pesan berguna untuk membuat sistem lebih interaktif, karena dapat menyampaikan informasi apakah suatu proses berjalan sukses atau sebaliknya. Berikut adalah beberapa Perancangan *Message Box* dari sistem yang dibuat.



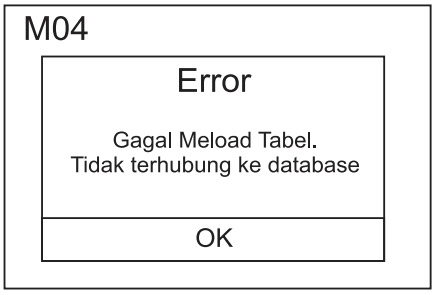
Gambar 3.56 Perancangan Pesan Pelatihan Selesai



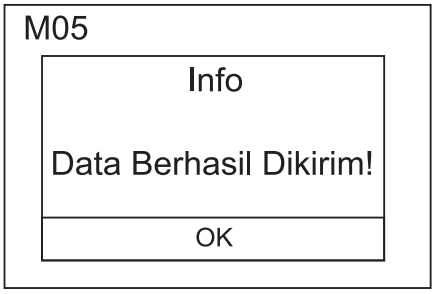
Gambar 3.57 Perancangan Pesan data Pelatihan Belum Lengkap



Gambar 3.58 Perancangan Pesan Pengujian Selesai



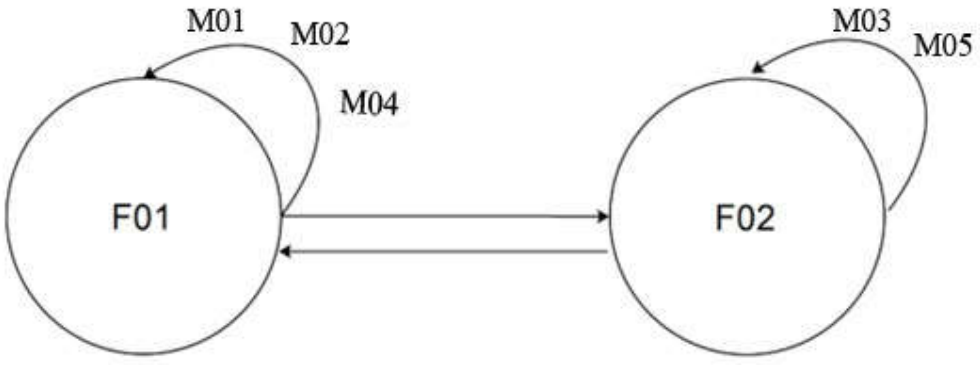
Gambar 3.59 Perancangan Pesan Gagal Load Tabel



Gambar 3.60 Perancangan Pesan Data Berhasil Dikirim

3.3.4 Jaringan Semantik

Jaringan Semantik dapat memudahkan untuk melihat bagaimana alur atau keterhubungan antara satu antarmuka dengan antarmuka yang lainnya atau antar muka dengan pesan. Berikut adalah Jaringan Semantik dari sistem yang dibuat.



Gambar 3.61 Jaringan Semantik