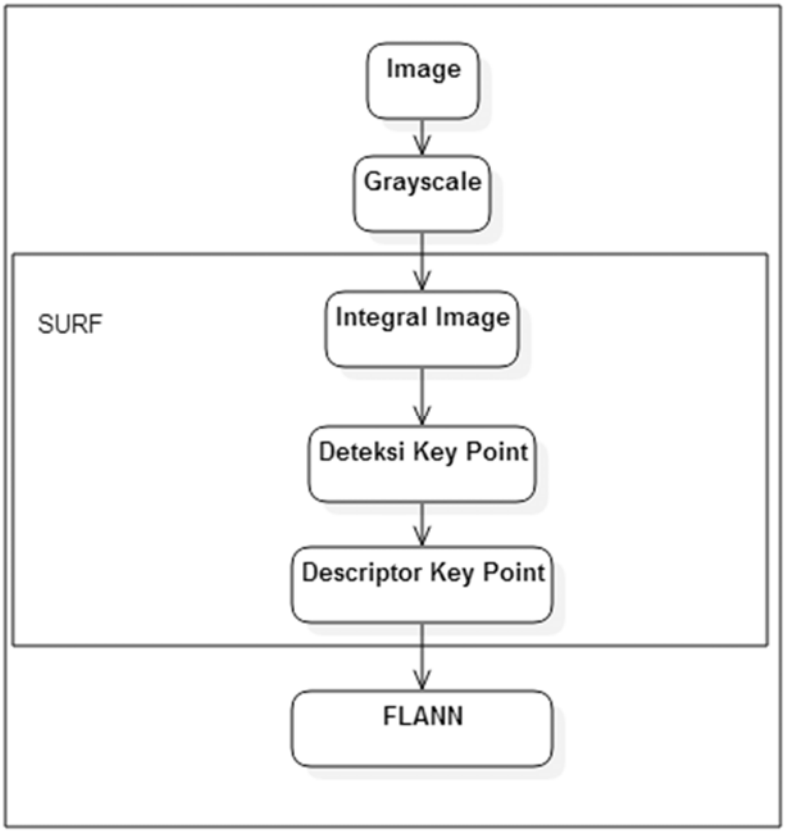
BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1. Analisis Masalah

Berdasarkan penelitian yang dilakukan dalam menentukan suatu properti terjual atau tidak pada permainan monopoli, dalam sebuah permainan di komputer untuk menentukannya menggunakan *variable* yang tertanam pada tiap objek, jika sebuah *variable* pada properti tersebut terisi maka properti tersebut sudah ada yang punya, jika *variable* tersebut tidak terisi maka properti tersebut belum ada yang punya, namun pada permainan asli tidak ada tempat untuk menanamkan *variable* seperti yang ada pada sebuah permainan komputer, melainkan membandingkan objek tersebut dengan objek lainya berdasarkan gambar maupun warna dan tulisan, agar sebuah kecerdasan buatan dapat melakukan tugasnya mendekati apa yang manusia lakukan.

3.2. Analisis Metode

Berikut adalah analisa metode pada Gambar 3.1 yang digunakan dalam proses pengenalan objek yang digunakan pada penelitian ini serta alur dari metode yang digunakan untuk mengenalli objek.



Gambar 3.1 Alur metode pencocokan gambar

3.2.1. Image

Kumpulan Gambar dari *webcam* secara terus menerus(*Video*) diambil dan dikirimkan ke komputer untuk kemudian diproses. Resolusi gambar yang digunakan adalah 640 x 480 *pixel*. Resolusi ini dipilih agar dapat mempercepat proses pendeteksian, karena penggunaan resolusi yang tinggi bisa meningkatkan waktu dari proses pendeteksian. Resolusi ini juga dipilih tidak terlalu kecil, karena selain terdapat batas minimal resolusi gambar pada *SURF* yaitu 100 x 100 *pixel*, juga agar calon titik kunci yang akan dideteksi berjumlah banyak. Sehingga bisa meningkatkan keakurasian dari metode *SURF*.

3.2.2. Grayscale

Grayscale adalah proses pengubahan sebuah gambar menjadi gambar keabu abuan. Dalam tiap *pixel* pada gambar mengandung satu *Sample* atau sinyal, yang membawa hanya informasi intensitas. *Grayscale* ini juga sering disebut sebagai hitam putih (*Black and White*) [13]. Dalam sebuah forum diskusi dengan judul "*How does EMGU CV convert an RGB image to Grayscale*" [29]. Pada *library Emgu CV* proses mengubah gambar *RGB* kedalam *Gray* sama seperti pada *Open CV*, yaitu menggunakan *Luminosity*. *Luminosity* adalah sebuah algoritma *Grayscale* dengan mengalikan setiap nilai *RGB* dengan konstanta tertentu yang sudah ditetapkan nilainya. Kemudian hasil perkalian seluruh *RGB* dijumlahkan dan didapatlah nilai *Grayscale* [30]. Nilai parameter yang sering digunakan pada *Luminosity* biasanya menggunakan *BT709 Grayscale*. Dimana dengan nilai parameternya adalah *Red*: 0.2125 *Green*: 0.7154 *Blue*: 0.0721. Dalam *Emgu CV* nilai paramater *Luminosity* yang digunakan adalah *Y-Grayscale* dengan nilai parameternya *Red*: 0.299 *Green*: 0.587 *Blue*: 0.114 [31]. Pada *BT709 Grayscale* ini biasa digunakan pada televisi berkualitas tinggi (*HDTV*), sedangkan *Y-Grayscale* digunakan pada *PAL* dan *NTSC.*

Dalam penelitian ini kedalaman warna yang digunakan untuk proses *Grayscale* adalah 8 *bit Grayscale* biasa disebut dengan *byte* yang memiliki 256 derajat keabuan dengan nilai dari 0 sampai 255. Dengan rumus *luminosity* dan nilai parameter *y-Grayscale* pada rumus (3.1) berikut.

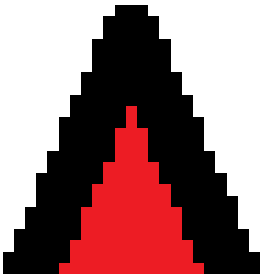
$$Y'=0.299R'+0.587G'+0.114B' \quad (3.1)$$

Misalkan saja terdapat sebuah nilai *RGB* pada suatu *pixel* adalah (100,0,150). Kemudian dimasukan kedalam rumus *luminosity* tersebut seperti pada persamaan (3.2), yang menghasilkan nilai Y' adalah 47, sehingga nilai *Grayscale*-nya (47,47,47) [32].

$$Y'= 0.299(100)+0.587(0)+0.114(150) \quad (3.2)$$
$$Y' = 47$$

Untuk gambar awal digunakan gambar 30 x 30 seperti Gambar 3.2 berikut. Gambar tersebut dalam bentuk tiga warna atau *RGB* dengan intensitas warnanya masing masing warna yaitu untuk warna merah pada Lampiran C1, warna hijau

pada Lampiran C2 dan warna biru pada Lampiran C3. Kemudian akan diubah kedalam bentuk *Grayscale* menggunakan *luminance* seperti pada rumus (3.1). Hasil dari proses ini dapat dilihat pada Lampiran C4..



Gambar 3.2 Gambar Awal

3.2.3. SURF

SURF atau *Speeded Up Robust Feature* adalah metode yang dikembangkan dari metode *SIFT*. Dengan fungsi yang sama sebagai algortima pendeteksi fitur yang dapat digunakan dalam penglihatan komputer (*Computer Vision*) seperti pengenalan objek (*Object Recognition*) [4]. Dalam tahap ini, terbagi beberapa *subprocess* diantaranya adalah *Integral Image*, *Key Point Detection, Descriptor*.

3.2.3.1. *Integral Image*

Dalam algoritma *SURF*, digunakan *Integral image* atau Gambar *integral* atau penjumlahan area tabel (*Summed Area Table*) yang dikomputasi secara cepat dari gambar masukan dan digunakan untuk mempercepat kalkulasi dari area persegi panjang yang tegak lurus. Dengan menggunakan gambar *integral* dalam menentukan jumlah intensitas *pixel* setelah dari proses *Grayscale,* dapat dikalkulasikan dengan rumus (2.1). Dimana titik x,y adalah jumlah dari semua pixel sampai atas kiri dan gambar yang digunakan adalah gambar *Grayscale* dari Lampiran C4, dan menghasilkan gambar *integral* pada Lampiran C5 [4].

3.2.3.2. *Key Point Detection*

SURF menggunakan pendekatan *Laplacian of Gaussian* dengan menggunakan *Box Filter*. *Box Filter* tersebut mempunyai *kernel* yang sudah diberi nilainya (*weight*). Ukuran *filter* terkecil yang dapat ditangani oleh *SURF* adalah 9x9 sesuai pada rumus 3.4 berikut. Secara *default* terdapat 3 buah *Octave* dan 4 buah *Interval* dalam tiap *Octave*-nya. Dengan menggunakan rumus 3.4, dapat menghitung ukuran *filter* pada *Octave* dan *Interval.* Gambar 3.3 dan Gambar 3.4 menunjukan ukuran *filter* pada *Octave* 1 dan *Interval* 1 arah X. Gambar 3.5 dan Gambar 3.6 menunjukan ukuran *filter* pada *Octave* 1 dan *Interval* 1 arah Y. Gambar 3.7 dan Gambar 3.8 menunjukan ukuran *filter* pada *Octave* 1 dan *Interval* 1 arah XY.

$$Filter\ Size = 3(2^{octave}\ x\ interval + 1) \quad \text{..... (3.4)}$$

Octave = 1, *Interval* = 1

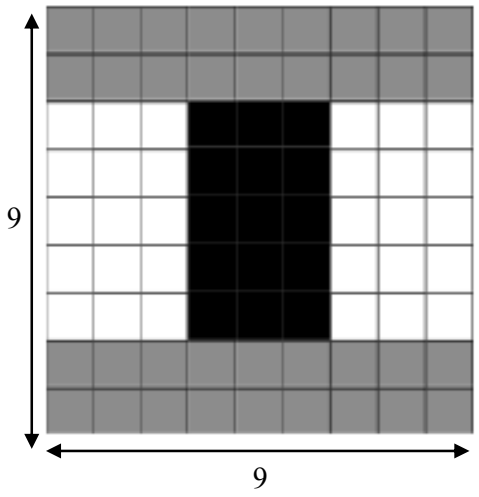
$Filter\ Size = 3(2^1\ x\ 1 + 1)$

$Filter\ Size = 9$

Octave = 1, *Interval* = 2

$Filter\ Size = 3(2^1\ x\ 2 + 1)$

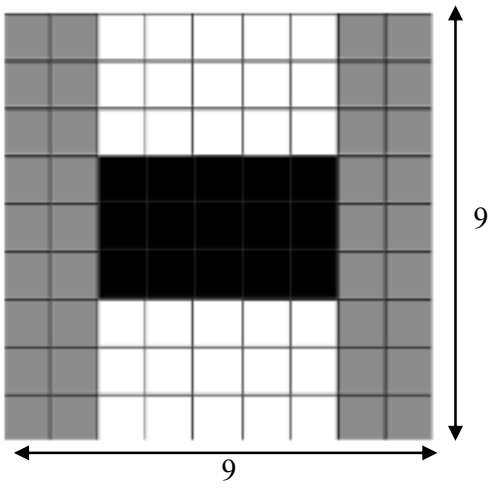
$Filter\ Size = 15$



Gambar 3.3 *Box Filter* Arah X, *Octave* =1 *Interval* =1 [4]

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
1	1	1	-2	-2	-2	1	1	1
1	1	1	-2	-2	-2	1	1	1
1	1	1	-2	-2	-2	1	1	1
1	1	1	-2	-2	-2	1	1	1
1	1	1	-2	-2	-2	1	1	1
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

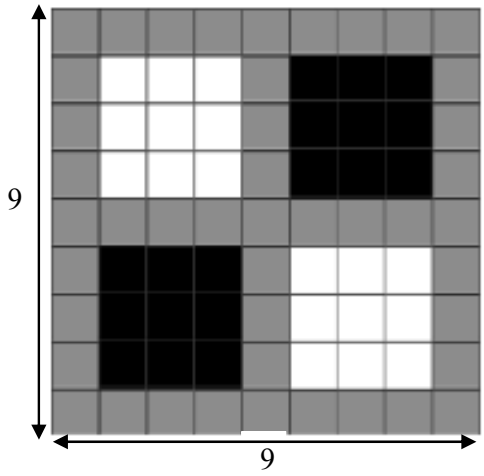
Gambar 3.4 Nilai *Box Filter* Arah X, *Octave* =1 *Interval* =1



Gambar 3.5 *Box Filter* Arah Y, *Octave* =1 *Interval* =1 [4]

0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	-2	-2	-2	-2	-2	0	0
0	0	-2	-2	-2	-2	-2	0	0
0	0	-2	-2	-2	-2	-2	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0

Gambar 3.6 Nilai *Box Filter* Arah Y, *Octave* =1 *Interval* =1



Gambar 3.7 *Box Filter* Arah XY, *Octave* =1 *Interval* =1 [4]

0	0	0	0	0	0	0	0	0
0	1	1	1	0	-2	-2	-2	0
0	1	1	1	0	-2	-2	-2	0
0	1	1	1	0	-2	-2	-2	0
0	0	0	0	0	0	0	0	0
0	-2	-2	-2	0	1	1	1	0
0	-2	-2	-2	0	1	1	1	0
0	-2	-2	-2	0	1	1	1	0
0	0	0	0	0	0	0	0	0

Gambar 3.8 Nilai *Box Filter* Arah XY, *Octave* =1 *Interval* =1

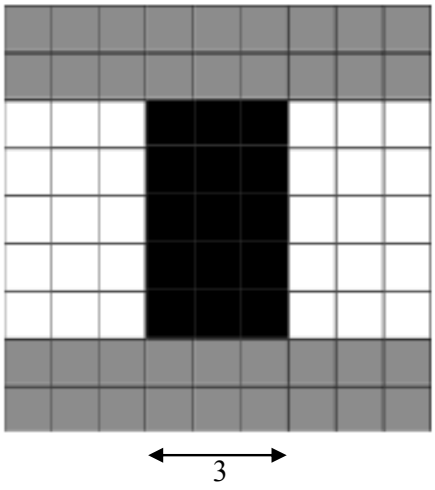
Untuk menentukan panjang *lobe* hitam pada tiap *Interval* menggunakan rumus *filter size* dibagi dengan 3 seperti rumus 3.5 berikut. Sehingga terlihat seperti pada Gambar 3.9, Gambar 3.10 dan Gambar 3.11 berikut.

$$Lobe\ Size = (2^{octave}\ x\ interval + 1) \quad \text{..... (3.5)}$$

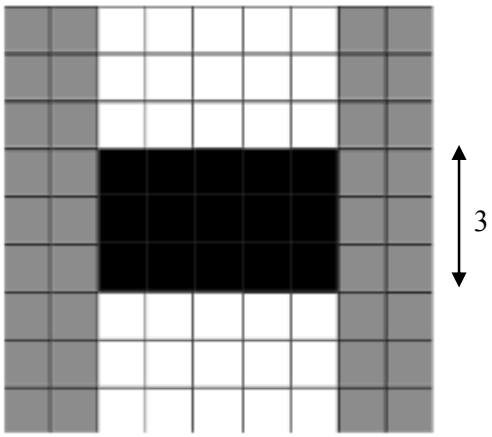
Octave = 1, *Interval* = 1

$Lobe\ Size = (2^1\ x\ 1 + 1)$

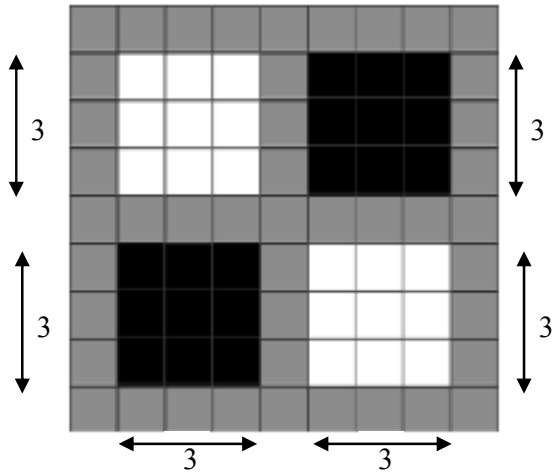
$Lobe\ Size = 3$



Gambar 3.9 Panjang *Lobe Box Filter* arah X [4]

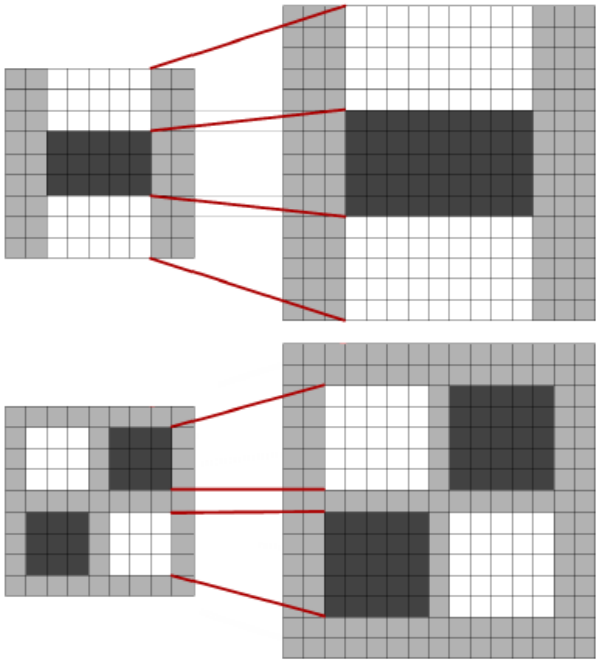


Gambar 3.10 Panjang *Lobe Box Filter* arah Y [4]



Gambar 3.11 Panjang *Lobe Box Filter* arah XY [4]

Untuk Dxx dan Dyy lebar *lobe*-nya ditambahkan 2 *pixel* untuk sisi kiri dan sisi kanan setelah *filter* 9x9, seperti pada Gambar 3.12.



Gambar 3.12 Pengukuran Lebar *lobe* 15x15 dari *lobe* awal [4]

Untuk menormalisasikan hasil dari konvolusi menggunakan rumus 3.6 berikut, dengan cara membagi nilai setelah konvolusi dengan nilai normalisasi berikut.

$$Normalize = (3\,.\,lobe)^2 \quad \text{..... (3.6)}$$

$Normalize = (3 . 3)^2$

$Normalize = 81$

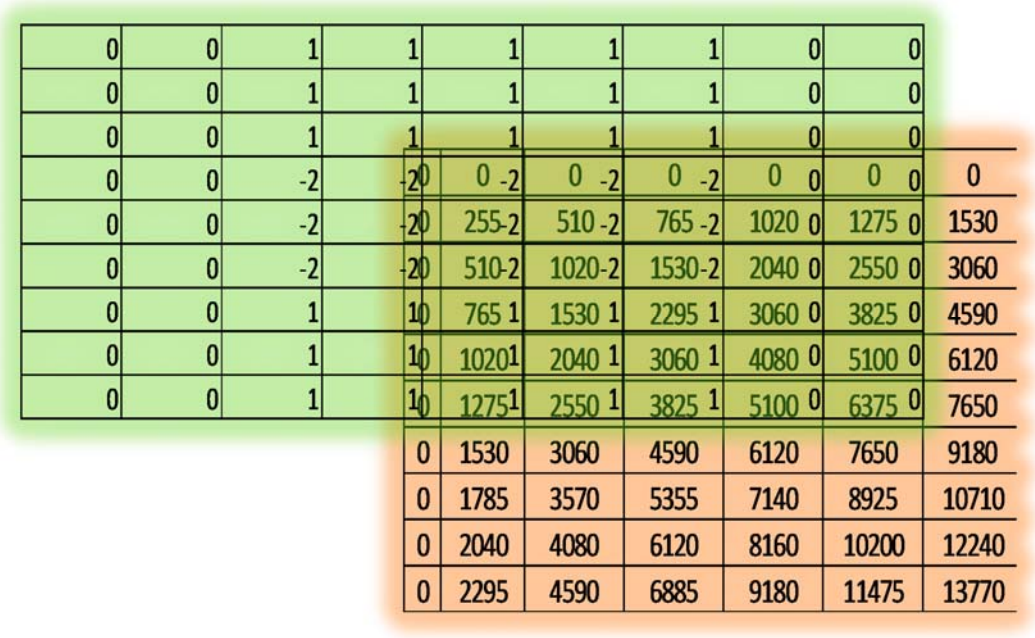
Untuk konvolusi antara gambar *integral* pada Gambar 3.13 dengan *Box Filter* 9x9 pada Gambar 3.14 dengan arah y adalah pada Gambar 3.15 dan Gambar 3.16 berikut.

0	0	0	0	0	0	0	0	0	0
0	255	510	765	1020	1275	1530	1785	2040	2295
0	510	1020	1530	2040	2550	3060	3570	4080	4590
0	765	1530	2295	3060	3825	4590	5355	6120	6885
0	1020	2040	3060	4080	5100	6120	7140	8160	9180
0	1275	2550	3825	5100	6375	7650	8925	10200	11475
0	1530	3060	4590	6120	7650	9180	10710	12240	13770
0	1785	3570	5355	7140	8925	10710	12495	14280	16065
0	2040	4080	6120	8160	10200	12240	14280	16320	18360
0	2295	4590	6885	9180	11475	13770	16065	18360	20655

Gambar 3.13 Contoh konvolusi *Box Filter,* gambar awal

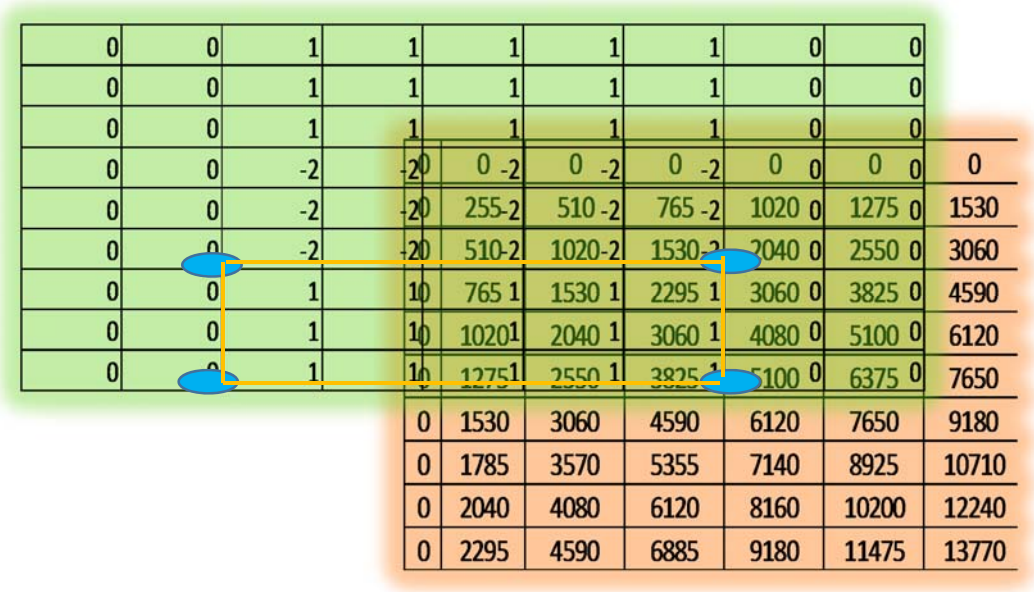
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	-2	-2	-2	-2	-2	0	0
0	0	-2	-2	-2	-2	-2	0	0
0	0	-2	-2	-2	-2	-2	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0	0

Gambar 3.14 *Kernel Box Filter* dengan arah y



Gambar 3.15 Proses konvolusi gambar *integral* dengan *kernel Box Filter*

Dengan memanfaatkan *integral image*, dapat dikalkulasikan dengan cepat perhitungan pada *Box Filter*. Untuk menghitung area yang diberi beban 1 tersebut, cukup dengan menggunakan rumus perhitungan *integral image* (2.3).



Gambar 3.16 Penggunaan *Integral Image* pada konvolusi *Box Filter*

0 + 3825 – 1530 – 0 = 2295

2295 * weight, wight adalah beban pada *kernel*, sehingga 2295 * 1 = 2295. Hasil ini kemudian dijumlahkan dengan perhitungan *integral image* pada beban lain pada *kernel*, seperti -2 dan 1 pada bagian atas. Untuk perhitungan arah X, Y dan XY ada pada Lampiran C6, Lampiran C7, dan Lampiran C8.

Setelah mengkonvolusikan *integral image* dengan *Box Filter*, hasil tersebut akan di normalisasikan dengan menggunakan rumus berikut.

$$Norm = (3 * lobe)^2 \quad \text{.....(3.7)}$$

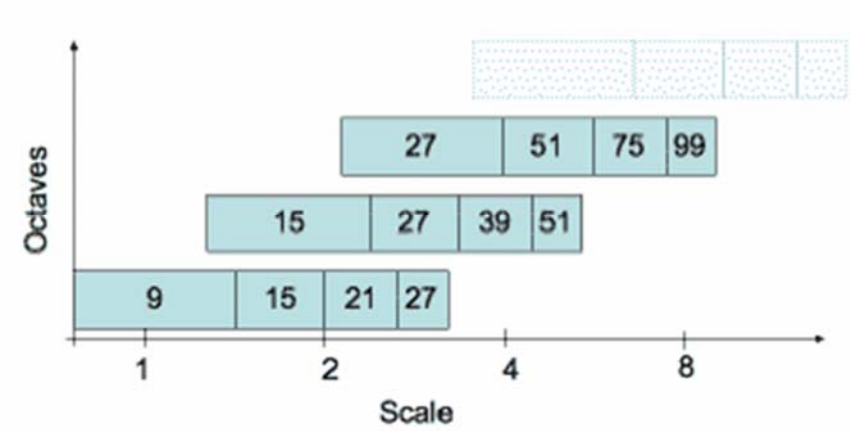
$Norm = (3 * 3)^2$

$Norm = 81$

Hasil dari pembagian atau normalisasi untuk arah x,y dan xy terlihat pada Lampiran C9, Lampiran C10 danLampiran C11. Setelah normalisasi, dicari determinan dari *hessian* menggunakan pendekatan *determinant of hessian* dengan rumus berikut 3.8 dan hasilnya seperti pada Lampiran C12 berikut.

$$\det(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \quad \text{.....(3.8)}$$

Hasil ini adalah lapisan awal atau *initial layer* pada *octav* dan *Interval* pertama, secara *default* terdapat 3 *Octave* dan 4 *Interval* dengan ukurannya pada Gambar 3.9. Kemudian dicari lagi dengan tingkatan *filter* selanjutnya, untuk hasilnya pada Lampiran C13 dengan *filter* ukuran 15, Lampiran C14 dengan *filter* ukuran 21 dan Lampiran C15 dengan *filter* ukuran 27.

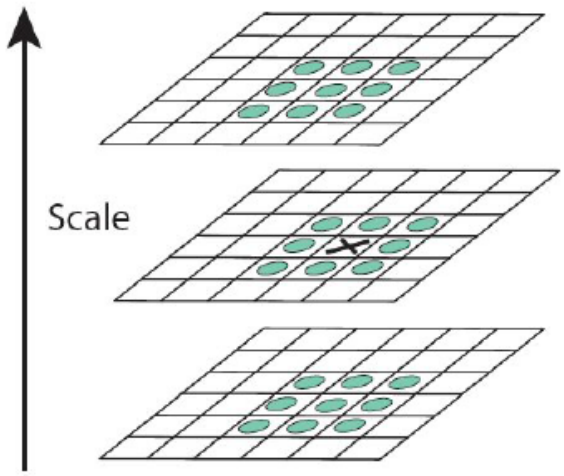


Gambar 3.17 Ruang Skala Pada Oktav Dan *Interval*

Setelah dihitung ke 3 *Octave* tersebut, kemudian akan diterapkan *tresholding* ambang batas tunggal menggunakan rumus pada tiap gambar tersebut. Dengan nilai keabuan yang dapat diatur sehingga nilai yang lebih atau sama dengan nilai *tresholding* saja yang akan di ekstrak. Untuk hasil dari ukuran 9 pada Lampiran C16, ukuran 15 pada Lampiran C17, ukuran 21 pada Lampiran C18, dan ukuran 27 pada Lampiran C19.

$$f(x,y) = \begin{cases} 0, f(x,y) < 500 \\ f(x,y), f(x,y) \geq 500 \end{cases} \text{.....(3.9)}$$

Setelah *tresholding* akan diterapkan *non-maximal suppression* pada tiap skala, ukuran dari *filter* memiliki skala yang berbeda beda. Penerapan *non-maximal suppression* seperti Gambar 3.18 pada titik X dibandingkan dengan 8 titik tetangga pada skala titik X dan 9 titik tetangga pada skala dibawah dan diatas skala titik X. Apabila titik X lebih besar dari titik tetangga pada skala sesudah dan sebelumnya, maka titik X adalah maxima.



Sumber : Jurnal " *Notes on the Open SURF Libary"*

Gambar 3.18 Non-Maximal Suppression

Contoh *Non-Maximal Suppression* pada ukuran 9,15,dan 21.

Ukuran 9

0	0	1829
0	1145	2680
923	3193	3038

Ukuran 15

0	640	3424
0	687	2909
0	564	2344

Ukuran 27

0	0	679
0	0	0
0	0	0

Dengan titik X berada pada *Interval* ke 2 atau ukuran 15 titik tengah, yaitu 687. Maka akan dibandingkan apakah titik X tersebut lebih besar dari tetangga di *Interval* tersebut dan apakah titik X tersebut lebih besar dari tetangga di *Interval* bawah dan diatas titik X tersebut. Jika tidak maka titik tersebut bukanlah *key point* atau point kunci. Apabila titik X lebih besar dari tetangga tetangganya, maka titik X menjadi calon dari *Key Point* yang akan disimpan dalam bentuk koordinat.

Seperti

keypoint 1 = (3,2)

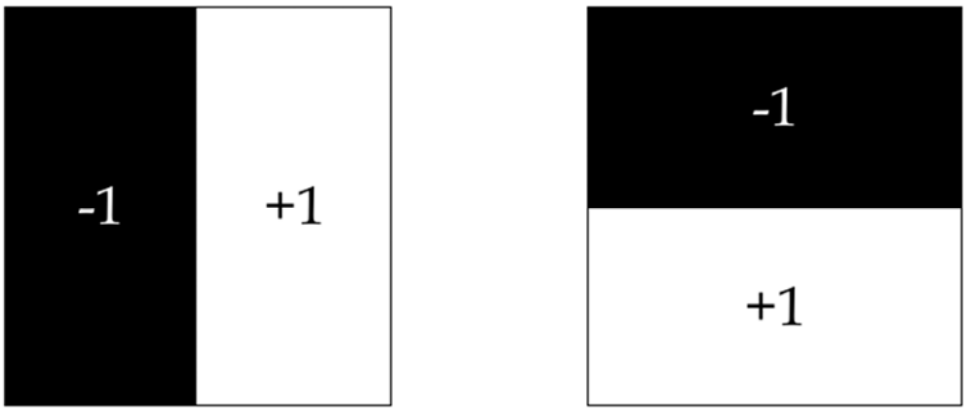
keypoint 2 = (2,5)

keypoint 3 = (5,5)

3.2.3.3. *Descriptor Key Point*

Untuk membuat deskriptor, *SURF* menggunakan *Haar Wavelet* seperti gambar Gambar 3.19 dalam prosesnya, untuk mencari nilai v dari rumus (3.10) berikut .

$$v = \{\sum dx\,, \sum |dx|, \sum dy\,, \sum |dy|\} \quad \text{..... (3.10)}$$



Gambar 3.19 ***Wavelet Response***

Tahapnya adalah sebagai berikut :

1. Sebuah kotak di buat dengan jarak 20 ditengah tiap tiap *keypoint* seperti Gambar 3.20 berikut .
2. Kemudian dibagi menjadi 4 x4 subarea atau dibagi 16 kotak seperti Gambar 3.21 berikut.
3. Hitung dengan menggunakan *Haar Wavelet* pada arah x dan y, x di *refer* sebagai dx dan y di *refer* sebagai dy.
4. Dengan menggunakan *filter wavelet response* pada Gambar 3.19 dapat dihitung dx, dy, abs(dx) atau *absolute(dx)* dan abs(dy) atau *absolute(dy).*

Untuk kotak pertama (kiri atas)

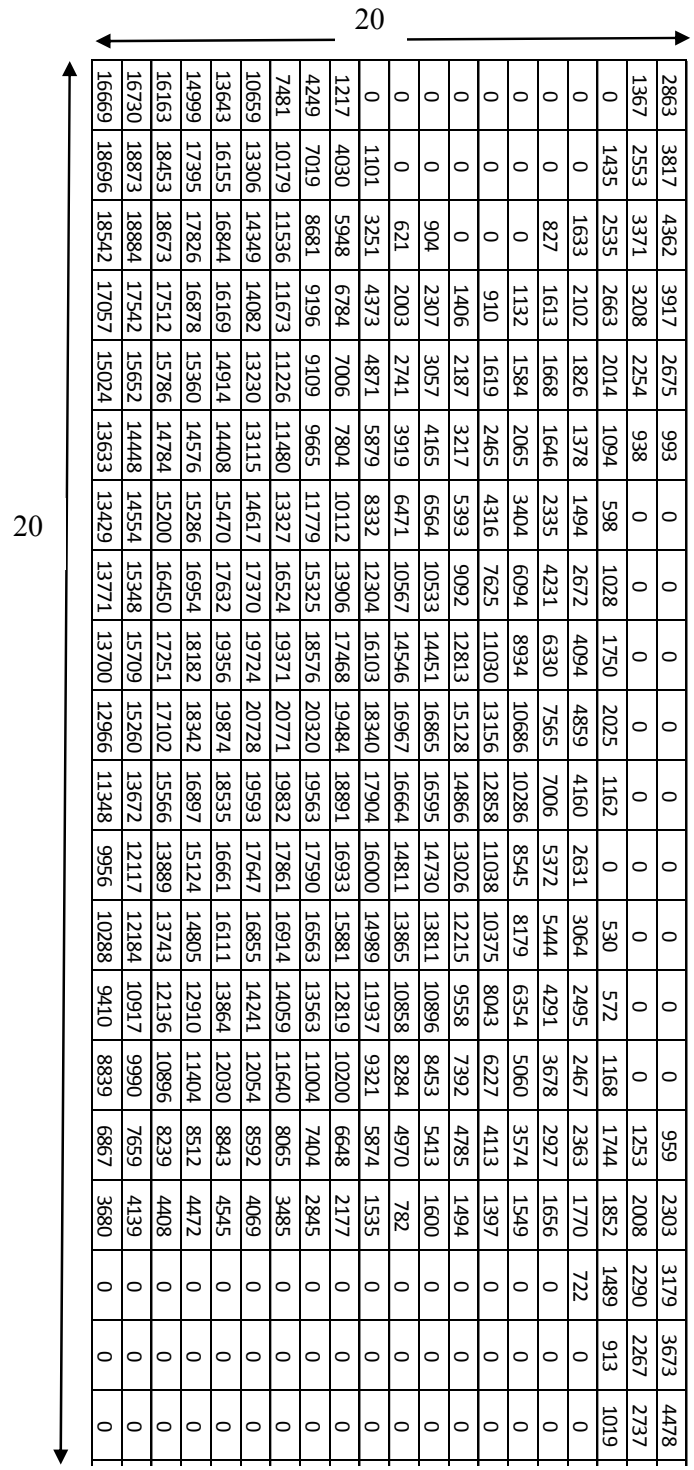
Dx = (3817-2863)+(4362-3817)+(3917-4362)+(2675-3917)+(2553-1367)+(3371-2553)+(3208-3371)+(2254-3208)+(1435-0)+(2535-1435)+(2663-2535)+(2014-2663)+(0-0)+(1633-0)+(2102-1633)+(1826-2102)+(0-0)+(827-0)+(1613-827)+(1668-1613)= 5937

Dy = (1367-2863)+(0-1367)+(0-0)+(0-0)+(2553-3817)+(1435-2553)+(0-1435) +(0-0)+(3371-4362)+(2535-3371)+(1633-2535)+(827-1613)+(2254-2675) +(2014-2254) +(1826-2014)+(1668-1826)= - 15420

$\sum|dx|$ =954 +545 +445 +1242+1186+818+163+954+1435+1100+128+649+1633 +276+827+786+55= 13196

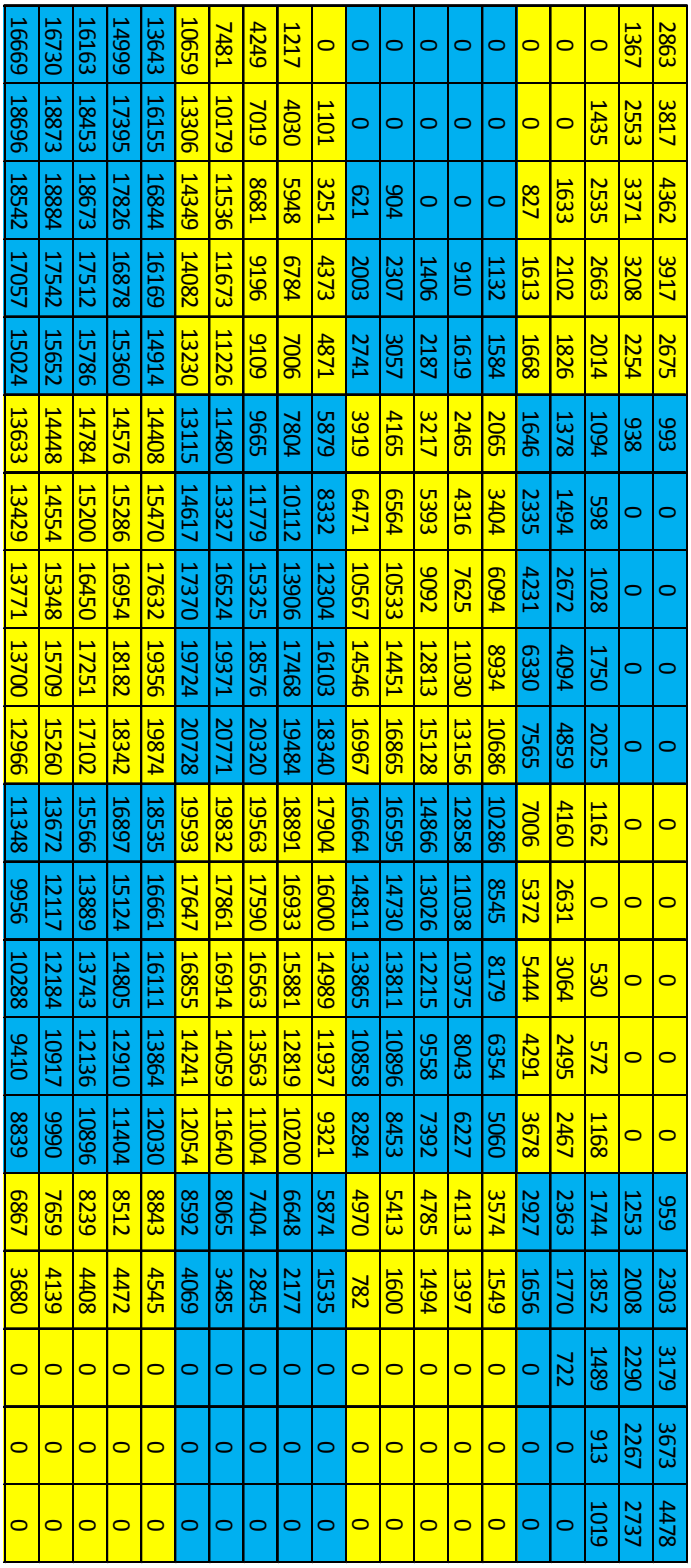
$\sum|dy|$ =1496+1367+1264+1118+1435+991+836+902+421+240+188+158 = 10416

$v = \{5937, -15420, 13196, 10416\}$



2863	3817	4362	3917	2675	993	0	0	0	0	0	0	0	0	0	0	959	2303	3179	3673	4478
1367	2553	3371	3208	2254	938	0	0	0	0	0	0	0	0	0	0	1253	2008	2290	2267	2737
0	1435	2535	2663	2014	1094	598	1028	1750	2025	1162	0	530	572	1168	1744	1852	1489	913	1019	
0	0	1633	2102	1826	1378	1494	2672	4094	4859	4160	2631	3064	2495	2467	2363	1770	722	0	0	
0	0	827	1613	1668	1646	2335	4231	6330	7565	7006	5372	5444	4291	3678	2927	1656	0	0	0	
0	0	0	1132	1584	2065	3404	6094	8934	10686	10286	8545	8179	6354	5060	3574	1549	0	0	0	
0	0	0	910	1619	2465	4316	7625	11030	13156	12858	11038	10375	8043	6227	4113	1397	0	0	0	
0	0	0	1406	2187	3217	5393	9092	12813	15128	14866	13026	12215	9558	7392	4785	1494	0	0	0	
0	0	904	2307	3057	4165	6564	10533	14451	16865	16595	14730	13811	10896	8453	5413	1600	0	0	0	
0	0	621	2003	2741	3919	6471	10567	14546	16967	16664	14811	13865	10858	8284	4970	782	0	0	0	
0	1101	3251	4373	4871	5879	8332	12304	16103	18340	17904	16000	14989	11937	9321	5874	1535	0	0	0	
1217	4030	5948	6784	7006	7804	10112	13906	17468	19484	18891	16933	15881	12819	10200	6648	2177	0	0	0	
4249	7019	8681	9196	9109	9665	11779	15325	18576	20320	19563	17590	16563	13563	11004	7404	2845	0	0	0	
7481	10179	11536	11673	11226	11480	13327	16524	19371	20771	19832	17861	16914	14059	11640	8065	3485	0	0	0	
10659	13306	14349	14082	13230	13115	14617	17370	19724	20728	19593	17647	16855	14241	12054	8592	4069	0	0	0	
13643	16155	16844	16169	14914	14408	15470	17632	19356	19874	18535	16661	16111	13864	12030	8843	4545	0	0	0	
14999	17395	17826	16878	15360	14576	15286	16954	18182	18342	16897	15124	14805	12910	11404	8512	4472	0	0	0	
16163	18453	18673	17512	15786	14784	15200	16450	17251	17102	15566	13889	13743	12136	10896	8239	4408	0	0	0	
16730	18873	18884	17542	15652	14448	14554	15348	15709	15260	13672	12117	12184	10917	9990	7659	4139	0	0	0	
16669	18696	18542	17057	15024	13633	13429	13771	13700	12966	11348	9956	10288	9410	8839	6867	3680	0	0	0	

Gambar 3.20 ***Haar Wavelet*** **20 X 20**



2863	3817	4362	3917	2675	993	0	0	0	0	0	0	0	959	2303	3179	3673	4478
1367	2553	3371	3208	2254	938	0	0	0	0	0	0	0	1253	2008	2290	2267	2737
0	1435	2535	2663	2014	1094	598	1028	1750	2025	1162	530	572	1168	1744	1852	1489	913	1019
0	0	1633	2102	1826	1378	1494	2672	4094	4859	4160	2631	3064	2495	2467	2363	1770	722	0	0
0	0	827	1613	1668	1646	2335	4231	6330	7565	7006	5372	5444	4291	3678	2927	1656	0	0	0
0	0	0	1132	1584	2065	3404	6094	8934	10686	10286	8545	8179	6354	5060	3574	1549	0	0	0
0	0	0	910	1619	2465	4316	7625	11030	13156	12858	11038	10375	8043	6227	4113	1397	0	0	0
0	0	0	1406	2187	3217	5393	9092	12813	15128	14866	13026	12215	9558	7392	4785	1494	0	0	0
0	0	904	2307	3057	4165	6564	10533	14451	16865	16595	14730	13811	10896	8453	5413	1600	0	0	0
0	0	621	2003	2741	3919	6471	10567	14546	16967	16664	14811	13865	10858	8284	4970	782	0	0	0
0	1101	3251	4373	4871	5879	8332	12304	16103	18340	17904	16000	14989	11937	9321	5874	1535	0	0	0
1217	4030	5948	6784	7006	7804	10112	13906	17468	19484	18891	16933	15881	12819	10200	6648	2177	0	0	0
4249	7019	8681	9196	9109	9665	11779	15325	18576	20320	19563	17590	16563	13563	11004	7404	2845	0	0	0
7481	10179	11536	11673	11226	11480	13327	16524	19371	20771	19832	17861	16914	14059	11640	8065	3485	0	0	0
10659	13306	14349	14082	13230	13115	14617	17370	19724	20728	19593	17647	16855	14241	12054	8592	4069	0	0	0
13643	16155	16844	16169	14914	14408	15470	17632	19356	19874	18535	16661	16111	13864	12030	8843	4545	0	0	0
14999	17395	17826	16878	15360	14576	15286	16954	18182	18342	16897	15124	14805	12910	11404	8512	4472	0	0	0
16163	18453	18673	17512	15786	14784	15200	16450	17251	17102	15566	13889	13743	12136	10896	8239	4408	0	0	0
16730	18873	18884	17542	15652	14448	14554	15348	15709	15260	13672	12117	12184	10917	9990	7659	4139	0	0	0
16669	18696	18542	17057	15024	13633	13429	13771	13700	12966	11348	9956	10288	9410	8839	6867	3680	0	0	0

Gambar 3.21 ***Haar Wavelet*** **dibagi menjadi 4x4** ***sub area*** **atau 16 petak**

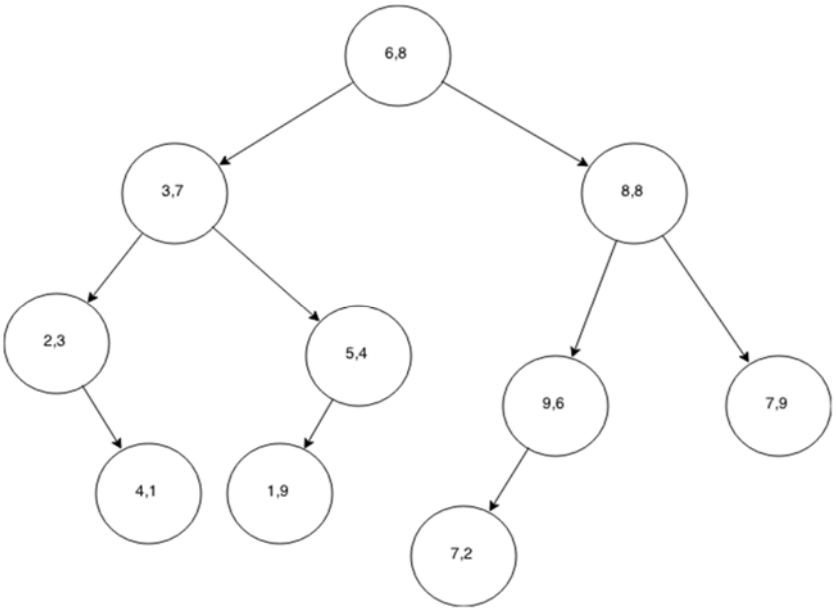
3.2.4. FLANN

FLANN adalah sebuah library untuk mencocokan tetanga yang terdekat. *FLANN* menggunakan *kd-tree* dalam prosesnya, dengan masukan beberapa buah *keypoint* dapat dihitung dengan cara berikut.

Keypoint= {1,9},{2,3},{4,1},{3,7},{5,4},{6,8},{7,2},{8,8},{7,9},{9,6}

1. Untuk urutan pertama cari median pada tiap X, median = 1 + 2 + 4 + 3 + 5 + 6 + 7 + 8 + 9 + 7= 52 / 10 = 5.2 dibulatkan ke atas yaitu 6
2. Titik 6, 8 menjadi median, selanjutnya pada iterasi pertama apabila x lebih kecil daripada x median, maka ditaruh disebelah kiri, jika tidak maka ditaruh disebelah kanan. Untuk iterasi ke dua dibedakan menggunakan y dan selanjutnya.
3. Kemudian untuk iterasi kedua ulangi tahap pertama pada tiap data yang x nya kurang dan lebih, cari median menggunakan y lalu buat *tree* nya.
4. Lalu titik dari *keypoint* lain diambil salah satu, kemudian akan dicocokan dengan *kd-tree* yang mana yang paling dekat, misal *keypointnya* adalah 7, 12 periksa pada iterasi pertama apakah 7 lebih besar dari 6, jika benar lanjut ke kanan bawah, apakah 12 lebih besar dari 8, jika benar maka periksa lagi *leaf* nya, jika sudah tidak ada, maka titik tersebut dianggap sebagai titik terdekat.

Untuk hasil nya adalah sebagai Gambar 3.22 berikut :



Gambar 3.22 Hasil *kd-tree*

3.3. Analisis Kebutuhan Non-Fungsional

Analisis Kebutuhan non-fungsional terdiri dari analisis perangkat keras, perangkat lunak dan pengguna yang menggunakan aplikasi yang akan dibangun.

3.3.1. Analisis Kebutuhan Perangkat Keras

Kebutuhan perangkat keras untuk menjalankan aplikasi ini terlihat pada Tabel 3.1 berikut .

Tabel 3.1 Analisis Kebutuhan Perangkat Keras

No	Tipe Perangkat	Spesifikasi Perangkat
1	Prosesor	Intel Core I7-3610QM 2.3Ghz
2	*Ram*	6 GB *DDR3 Dual Channel*
3	*VGA*	*Nvidia GT 630M 2GB*
4	Layar	LCD 14 “
5	Penyimpanan	256 GB *SSD (Solid State Disk)*
6	*Keyboard*	*Laptop Keyboard*
7	*Mouse*	*Logitech Wireless Mouse M185*
8	Kamera	*Logitech C170 5 MP*
9	Papan Prototipe	*Arduino Uno Rev 3, Bread board (*840 lubang*)*
10	*Servo*	1. 1 x *Hitech HS-805bb+ (24.7Kg/Cm)* 2. 2 x *GWS(Grand Wing System)S03N/STD/J (3.2Kg/Cm)* 3. 2 x *GWS(Grand Wing System)S03T/STD/F (8Kg/Cm)*
	Power Supply	1. *Montana Regulated Power Supply 5-13.8V 7A*

	Perangkat lain	1. Rangka alat bantu (*Alumunium*) 2. *Rainbow Flat Cable 2 M* 3. Beban Pemberat (*2 x Stainless Steel)* 4. *Usb Extension 1.5 M*

3.3.2. Analisis Kebutuhan Perangkat Lunak

Perangkat lunak yang dipakai dalam pembuatan aplikasi terlihat pada Tabel 3.2 sebagai berikut .

Tabel 3.2 Analisa Kebutuhan Perangkat Lunak

No	Tipe Perangkat	Spesifikasi Perangkat
1	Sistem Operasi	*Windows 7 Ultimate SP1 64 Bit*
2	Bahasa Pemprograman	*C#*
3	*Compiler*	*Microsoft Visual Studio Ultimate 2012, Arduino 1.63*
4	*FrameWork*	*Emgu CV 2.4.10.1940*

3.3.3. Analisis Pengguna

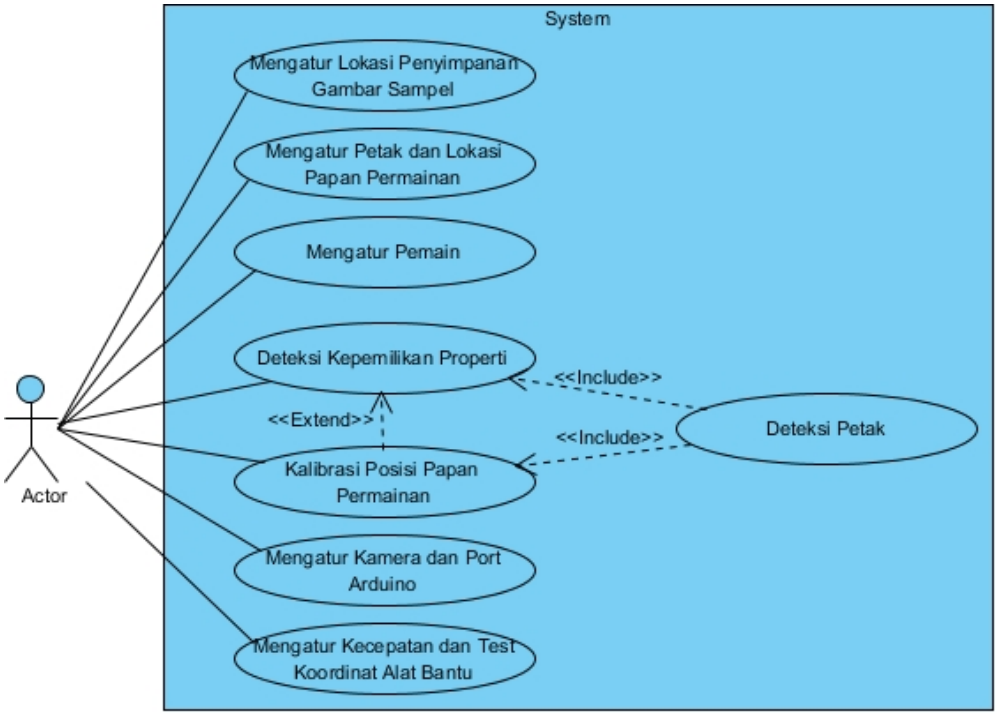
Pengguna yang menggunakan sistem ini dapat dimainkan dari anak anak yang dibimbing oleh orang tuanya sampai dapat dimainkan oleh anak remaja hingga dewasa, berkisar dari umur 10 tahun keatas.

3.4. Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional didalamnya terdapat diagram *UML* seperti *Use Case Diagram, Activity Diagram, Class Diagram* dan *Sequence Diagram*. Diagram ini digunakan sebagai desain rancangan untuk mempermudah proses pengembangan sebuah aplikasi, dan dikhusukan untuk aplikasi yang menggunakan pemrograman berorientasi objek.

3.4.1. Diagram Use Case

Diagram *Use Case* memperlihatkan pada kita hubungan-hubungan yang terjadi antara aktor-aktor dengan usecase-usecase di dalam sistem. Adapun *Use Case Diagram* yang dari aplikasi yang dibangun adalah sebagai Gambar 3.23 berikut



Gambar 3.23 Diagram *Use Case*

3.4.2. Skenario Use Case Mengatur Lokasi Penyimpanan Gambar Sampel

Skenario *Use Case* mengatur lokasi penyimpanan gambar sampel dalam sistem yang dibangun adalah seperti Tabel 3.3 berikut :

Tabel 3.3 Skenario *Use Case* Mengatur Lokasi Penyimpanan Gambar Sampel

Use Case name	Mengatur Lokasi Penyimpanan Gambar Sampel
Related Requirements	-
Goal in Context	Menyimpan lokasi penyimpanan gambar sampel
Precondition	*User* memiliki lokasi penyimpanan gambar
Successful End Condition	Lokasi penyimpanan gambar tersimpan

Failed End Condition	Lokasi penyimpanan gambar tidak tersimpan	
Primay Actors	*User*	
	Secondary Actor	Sistem
	Trigger	Setelah *user* memilih menu pengaturan petak dan lokasi papan permainan
Main Flow	*Step*	*Action*
	1	*User* memilih menu pengaturan petak dan lokasi papan permainan
	2	Sistem menampilkan form pengaturan petak dan lokasi papan permainan
	3	*User* memilih lokasi penyimpanan gambar sampel
	4	Sistem memverifikasi apakah ada datagambarsample.txt pada lokasi penyimpanan gambar sampel
	5	Sistem menload datagambarsampel.txt
	6	SIstem menyimpan lokasi penyimpanan gambar sampel
	7	Sistem memverifikasi lokasi penyimpanan gambar sampel apakah sama dengan "Belum ditentukan"
	8	Sistem mengaktifkan fungsi pengaturan petak, mengatur lokasi pencarian, deteksi

		kepemilikan dan kalibrasi papan permainan
Extension	*Step*	*Branch Action*
	3.1	Sistem memverifikasi lokasi penyimpanan gambar sampel apakah sama dengan "Belum ditentukan"
	4.1	SIstem menyimpan lokasi penyimpanan gambar sampel
	7.1	Sistem menonaktifkan fungsi pengaturan petak, mengatur lokasi pencarian, deteksi kepemilikan dan kalibrasi papan permainan

3.4.3. Skenario Use Case Mengatur Petak dan Lokasi Papan Permainan

Skenario *Use Case* mengatur petak dan lokasi papan permainan dalam sistem yang dibangun adalah seperti Tabel 3.4 berikut :

Tabel 3.4 Skenario *Use Case* Mengatur Petak dan Lokasi Papan Permainan

Use Case name	Mengatur Petak dan Lokasi Papan Permainan
Related Requirements	-
Goal in Context	Menyimpan lokasi, gambar, nama petak serta koordinat pergerakan kamera
Precondition	*User* memiliki lokasi, gambar, nama petak, serta koordinat pergerakan kamera
Successful End Condition	Lokasi, gambar, nama petak dan koordinat pergerakan kamera tersimpan

Failed End Condition	Lokasi, gambar, nama dan koordinat pergerakan kamera petak tidak tersimpan	
Primay Actors	*User*	
	Secondary Actor	Sistem
	Trigger	Setelah *user* memilih menu pengaturan petak dan papan permainan
Main Flow	*Step*	*Action*
	1	*User* memilih menu pengaturan petak dan papan permainan
	2	Sistem menampilkan form pengaturan petak dan papan permainan
	3	*User* memasukan nama, lokasi, koordinat dan gambar petak
	4	Sistem memverifikasi nama petak
	5	Sistem memverifikasi lokasi dan koordinat petak apakah ada yang sama
	6	Sistem memverifikasi gambar petak
	7	Sistem menyimpan pengaturan petak ke datagambarsampel.txt
	8	Sistem menampilkan pemberitahuan data berhasil ditambah
Extension	*Step*	*Branch Action*
	2.a1	*User* memilih data petak untuk diubah

	2.a2	*User* memilih ubah
	2.a3	Sistem menampilkan data yang dipilih
	2.a4	*User* merubah data petak
	2.a5	Sistem memverifikasi kelengkapan data
	2.a5.1	Sistem menverifikasi lokasi dan koordinat petak apakah ada yang sama
	2.a5.1.a1	Sistem menyimpan pengaturan petak ke datagambarsampel.txt
	2.a5.1.a2	Sistem menampilkan pemberitahuan data berhasil diubah
	2.a5.1.2	Sistem menampilkan pemberitahuan terdapat petak dengan lokasi atau koordinat yang sama
	2.a5.2	Sistem menampilkan pemberitahuan data petak belum terisi lengkap
	2.b1	*User* memilih data petak untuk dihapus
	2.b2	Sistem menghapus data yang dipilih
	2.b3	Sistem menampilkan pemberitahuan data berhasil dihapus

	4.1	Sistem menampilkan pemberitahuan nama petak harus diisi
	5.1	Sistem menampilkan pemberitahuan terdapat lokasi dan koordinat yang sama
	6.1	Sistem menampilkan pemberitahuan gambar petak belum dipilih

3.4.4. Skenario Use Case Mengatur Pemain

Skenario *Use Case* mengatur pemain dalam sistem yang dibangun adalah seperti Tabel 3.5 berikut :

Tabel 3.5 Skenario *Use Case* Mengatur Pemain

Use Case name	Mengatur Pemain	
Related Requirements	-	
Goal in Context	Pemain terdaftar dalam sistem	
Precondition	*User* memiliki posisi, nama dan jumlah pemain.	
Successful End Condition	Pemain terdaftar dalam sistem	
Failed End Condition	Pemain tidak terdaftar dalam sistem	
Primay Actors	*User*	
	Secondary Actor	Sistem
	Trigger	Setelah *user* memilih menu pengaturan pemain
Main Flow	*Step*	*Action*
	1	*User* memilih menu pengaturan pemain
	2	Sistem menampilkan form pengaturan pemain

	3	*User* memasukan nama, lokasi dan atribut pemain
	4	Sistem memverifikasi kelengkapan data pemain
	5	Sistem memverifikasi lokasi pemain yang sama
	6	Sistem memverifikasi data pemain apakah jumlah pemain = 4 dan tidak ada pemain dengan atribut komputer
	7	Sistem menyimpan data pemain
	8	Sistem menampilkan pemberitahuan data berhasil ditambah
Extension	*Step*	*Branch Action*
	2.a1	*User* memilih data untuk dirubah
	2.a2	Sistem menampilkan data yang dipilih
	2.a3	*User* mengubah data pemain
	2.a4	Sistem memerifikasi kelengkapan data
	2.a4.1	Sistem memverifikasi lokasi pemain apakah ada yang sama
	2.a4.1.1	Sistem menampilkan pemberitahuan terdapat pemain dengan lokasi yang sama
	2.a4.1.b1	Sistem menyimpan data pemain

	2.a4.1.b2	Sistem menampilkan pemberitahuan data berhasil di ubah
	2.a4.2	Sistem menampilkan pemberitahuan terdapat pemain dengan lokasi yang sama
	2.b1	*User* memilih data untuk dihapus
	2.b2	Sistem menghapus data pemain yang dipilih
	2.b3	Sistem menampilkan pemberitahuan data berhasil dihapus
	4.1	Sistem menampilkan pemberitahuan data pemain belum terisi lengkap
	5.1	Sistem menampilkan pemberitahuan terdapat pemain dengan lokasi yang sama
	6.1	Sistem menampilkan pemberitahuan harus ada pemain dengan atribut komputer

3.4.5. Skenario Use Case Deteksi Kepemilikan Properti

Skenario *Use Case* deteksi kepemilikan properti dalam sistem yang dibangun adalah seperti Tabel 3.6 berikut :

Tabel 3.6 Skenario *Use Case* Deteksi Kepemilikan Properti

Use Case name	Deteksi Kepemilikan Properti
Related Requirements	-
Goal in Context	Mendapatkan informasi kepemilikan properti

Precondition	Petak papan permainan, lokasi pencarian, pemain, kamera dan port *Arduino* sudah diatur	
Successful End Condition	Properti diketahui kepemilikannya	
Failed End Condition	Properti tidak diketahui kepemilikannya	
Primay Actors	*User*	
	Secondary Actor	Sistem
	Trigger	Setelah *user* memilih menu deteksi properti
Main Flow	*Step*	*Action*
	1	*User* memilih petak yang akan dicari kepemilikan propertinya
	2	Sistem memverifikasi kamera, port *Arduino* dan data petak
	3	Sistem memverifikasi jenis kartu
	4	Sistem memverifikasi apakah pernah melakukan kalibrasi
	5	Sistem menggerakan kamera ke lokasi pencarian yang dipilih
	6	Sistem menampilkan pesan pemberitahuan menggerakan alat bantu
	7	Deteksi Petak
	8	Sistem menggerakan kamera ke lokasi kartu properti pemain
	9	Sistem menampilkan pesan pemberitahuan menggerakan alat bantu
	10	Deteksi Petak

	11	Sistem menampilkan nama pemain dari lokasi pencarian dan lokasi pemain
	12	Sistem memverifikasi koordinat idle
	13	Sistem menggerakan kamera ke koordinat idle
	14	Sistem menampilkan pesan pemberitahuan menggerakan alat bantu
Extension	*Step*	*Branch Action*
	2.1	Sistem menampilkan pesan pemberitahuan Pastikan kamera, port *Arduino* sudah dipilih dan data petak sudah lengkap
	3.1	Sistem menampilkan pemberitahuan petak yang dipilih bukan kartu properti
	4.1	Kalibrasi Posisi Papan Permainan
	7.1	Sistem menampilkan pemberitahuan periksa data petak atau lakukan kalibrasi
	10.1	Sistem memverifikasi batas jumlah kartu pemain apakah masih ada
	10.1.1	Sistem menggerakan kamera ke lokasi kartu properti pemain

	10.1.2	Sistem memverifikasi jumlah pemain apakah masih ada pemain
	10.1.2.1	Sistem menggerakan kamera ke lokasi kartu properti pemain
	10.1.2.2	Sistem menampilkan pemberitahuan properti tersebut belum ada yang memiliki
	12.1	Sistem menggerakan kamera ke koordinat deafult idle

3.4.6. Skenario Use Case Kalibrasi Posisi Papan Permainan

Skenario *Use Case* kalibrasi posisi papan permainan dalam sistem yang dibangun adalah seperti Tabel 3.7 berikut :

Tabel 3.7 Skenario *Use Case* Kalibrasi Posisi Papan Permainan

Use Case name	Kalibrasi Posisi Papan Permainan	
Related Requirements	-	
Goal in Context	Posisi papan terkalibrasi pada sistem	
Precondition	Petak papan permainan, kamera dan port *Arduino* sudah diatur	
Successful End Condition	Papan terkalibrasi pada sistem	
Failed End Condition	Papan tidak terkalibrasi pada sistem	
Primay Actors	*User*	
	Secondary Actor	Sistem
	Trigger	Setelah *user* memilih menu kalibrasi papan permainan
Main Flow	*Step*	*Action*
	1	*User* memilih menu kalibrasi papan permainan

	2	Sistem memverifikasi kamera, port *Arduino* dan data petak
	3	Sistem menggerakan kamera ke lokasi pencarian
	4	Sistem menampilkan pesan pemberitahuan menggerakan alat bantu
	5	Deteksi Petak
	6	Sistem menampilkan pemberitahuan proses kalibrasi berhasil dengan lokasi petak
	7	Sistem menggerakan kamera ke koordinat idle
	8	Sistem menampilkan pesan pemberitahuan menggerakan alat bantu
Extension	*Step*	*Branch Action*
	2.1	Sistem menampilkan pesan pemberitahuan Pastikan kamera, port *Arduino* sudah dipilih dan data petak sudah lengkap
	5.1	Sistem memverifikasi apakah sudah dicari semua kemungkinan
	5.1.1	Sistem mengambil gambar dari kamera
	5.1.2	Sistem menampilkan pemberitahuan periksa kembali data petak

3.4.7. Skenario Use Case Mengatur Kamera dan Port Arduino

Skenario *Use Case* mengatur kamera dan *Port Arduino* dalam sistem yang dibangun adalah seperti Tabel 3.8 berikut :

Tabel 3.8 Skenario *Use Case* Mengatur Kamera dan *Port Arduino*

Use Case name	Mengatur Kamera dan *Port Aduino*	
Related Requirements	-	
Goal in Context	Kamera dan *port Arduino* terpilih	
Precondition	*User* memiliki kamera dan *port Arduino*	
Successful End Condition	Kamera dan *port Arduino* terpilih	
Failed End Condition	Kamera dan *port Arduino* tidak terpilih	
Primay Actors	*User*	
	Secondary Actor	Sistem
	Trigger	Setelah *user* akan memilih kamera dan *port Arduino*
Main Flow	*Step*	*Action*
	1	*User* memilih form deteksi properti
	2	Sistem menampilkan form deteksi properti
	3	*User* memilih kamera
	4	Sistem memvalidasi daftar kamera
	5	Sistem menggunakan kamera yang dipilih
Extension	*Step*	*Branch Action*
	2.a1	*User* memilih port *Arduino*
	2.a2	Sistem memvalidasi daftar port *Arduino*
	2.a2.1	Sistem menggunakan port *Arduino* yang terpilih

	2.a2.2	Sistem menampilkan pemberitahuan tidak ada port *Arduino* yang terkoneksi
	4.1	Sistem menampilkan pemberitahuan tidak ada kamera yang terkoneksi

3.4.8. Skenario Use Case Mengatur Kecepatan dan Test Koordinat Alat Bantu

Skenario *Use Case* mengatur kecepatan dan test koordinat alat bantu dalam sistem yang dibangun adalah seperti Tabel 3.9 berikut :

Tabel 3.9 Skenario *Use Case* Mengatur Kecepatan dan Test Koordinat Alat Bantu

Use Case name	Mengatur Kecepatan dan Test Koordinat Alat Bantu	
Related Requirements	-	
Goal in Context	Mengatur kecepatan dan mencoba koordinat untuk alat bantu	
Precondition	Kamera dan Port *Arduino* sudah terdeteksi	
Successful End Condition	Kecepatan dan test koordinat sudah diatur	
Failed End Condition	Kecepatan dan test koordinat tidak diatur	
Primay Actors	*User*	
	Secondary Actor	Sistem
	Trigger	Setelah *user* memilih menu pengaturan petak dan papan permainan
Main Flow	*Step*	*Action*
	1	*User* memilih form pengaturan petak dan papan permainan

	2	Sistem menampilkan form pengaturan petak dan papan permainan
	3	*User* memilih aktifkan test koordinat
	4	Sistem mengaktifkan test koordinat
	5	*User* memilih kecepatan alat bantu
	6	Sistem menyimpan kecepatan alat bantu
Extension	*Step*	*Branch Action*
	2.a1	*User* mengatur test koordinat
	2.a2	*User* memilih coba koordinat
	2.a3	Sistem mengeksekusi koordinat alat bantu

3.4.9. Skenario Use Case Deteksi Petak

Skenario *Use Case* deteksi petak dalam sistem yang dibangun adalah seperti Tabel 3.9 berikut :

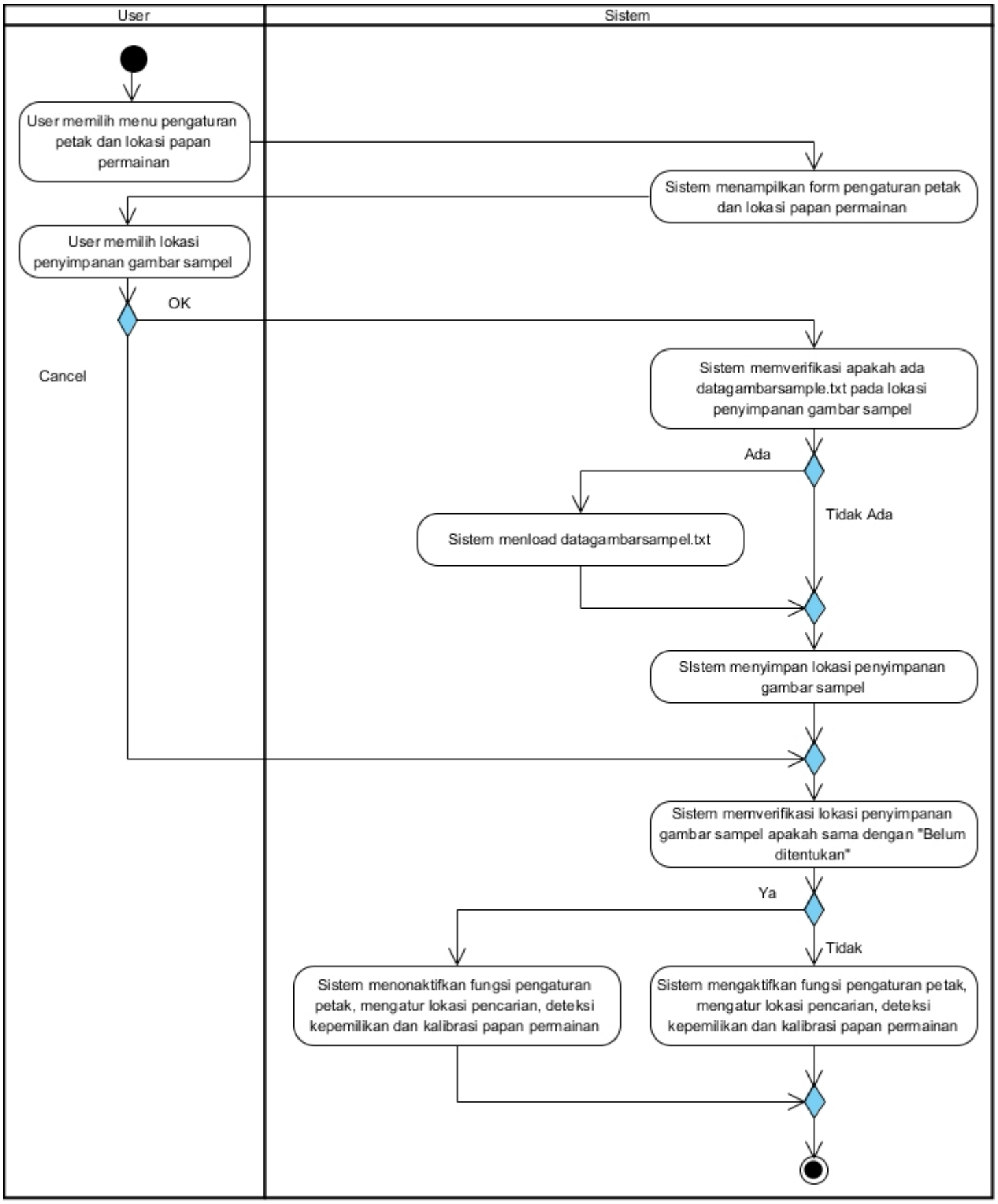
Tabel 3.10 Skenario *Use Case* Deteksi Petak

Use Case name	Deteksi Petak	
Related Requirements	-	
Goal in Context	Mendeteksi Petak	
Precondition	Kamera dan Port *Arduino* sudah terdeteksi	
Successful End Condition	Hasil deteksi petak diketahui	
Failed End Condition	-	
Primay Actors	Sistem	
	Secondary Actor	-

	Trigger	Setelah sistem menggerakan kamera
Main Flow	*Step*	*Action*
	1	Sistem mengambil gambar dari kamera
	2	Sistem menyamakan gambar dengan gambar sampel pada lokasi pencarian yang dipilih
	3	Hasil persamaan = 1
	4	Sistem menyimpan hasil persamaan
	5	Sistem memverifikasi perulangan sebanyak 10 kali
	6	Sistem menjumlahkan rata rata hasil persamaan
Extension	*Step*	*Branch Action*
	2.1	Hasil persamaan = 0
	5.1	Sistem mengambil gambar dari kamera

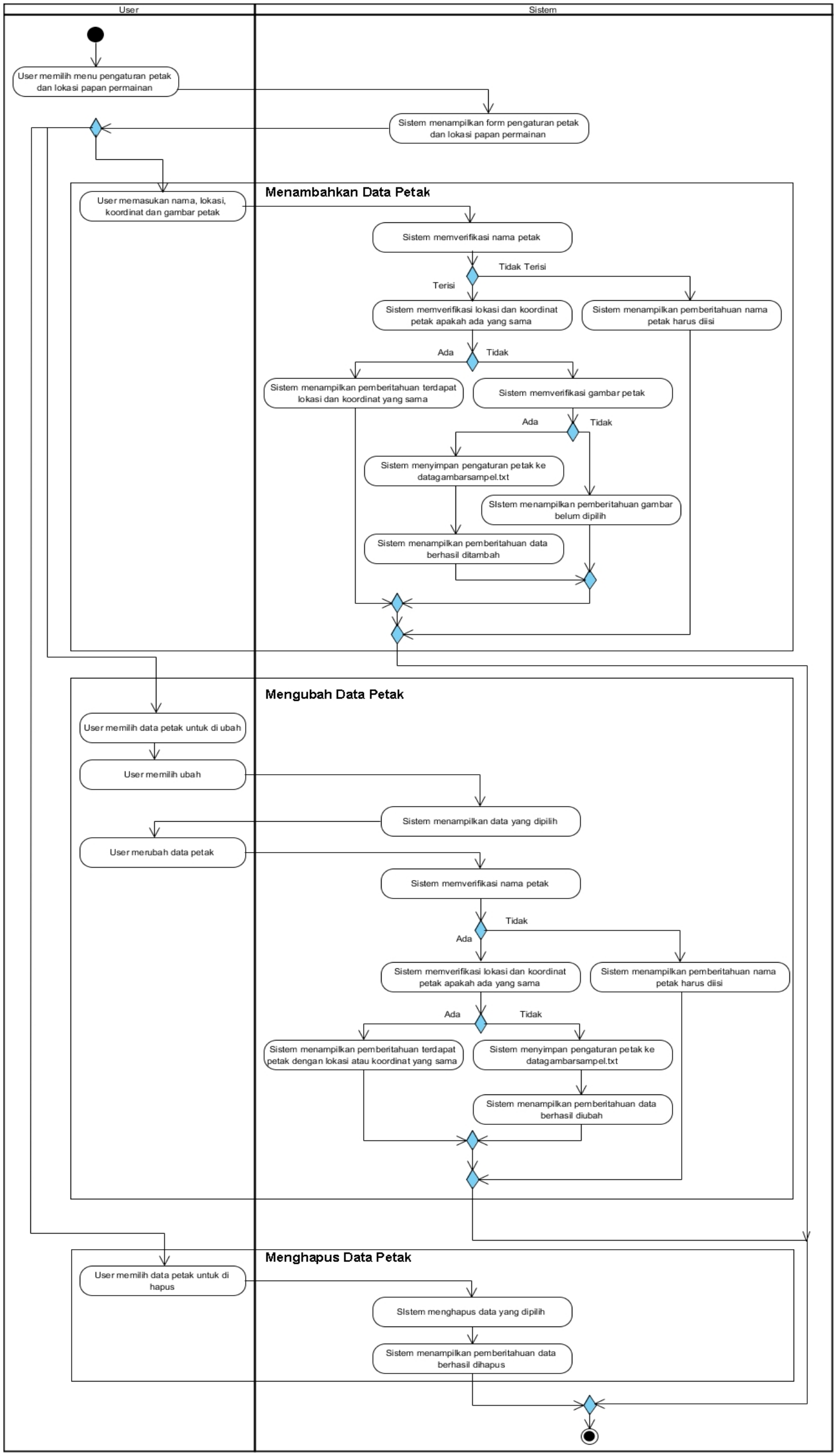
3.4.10. Activity Diagram

3.4.10.1. *Activity Diagram* Mengatur Lokasi Penyimpanan Gambar *Sample*



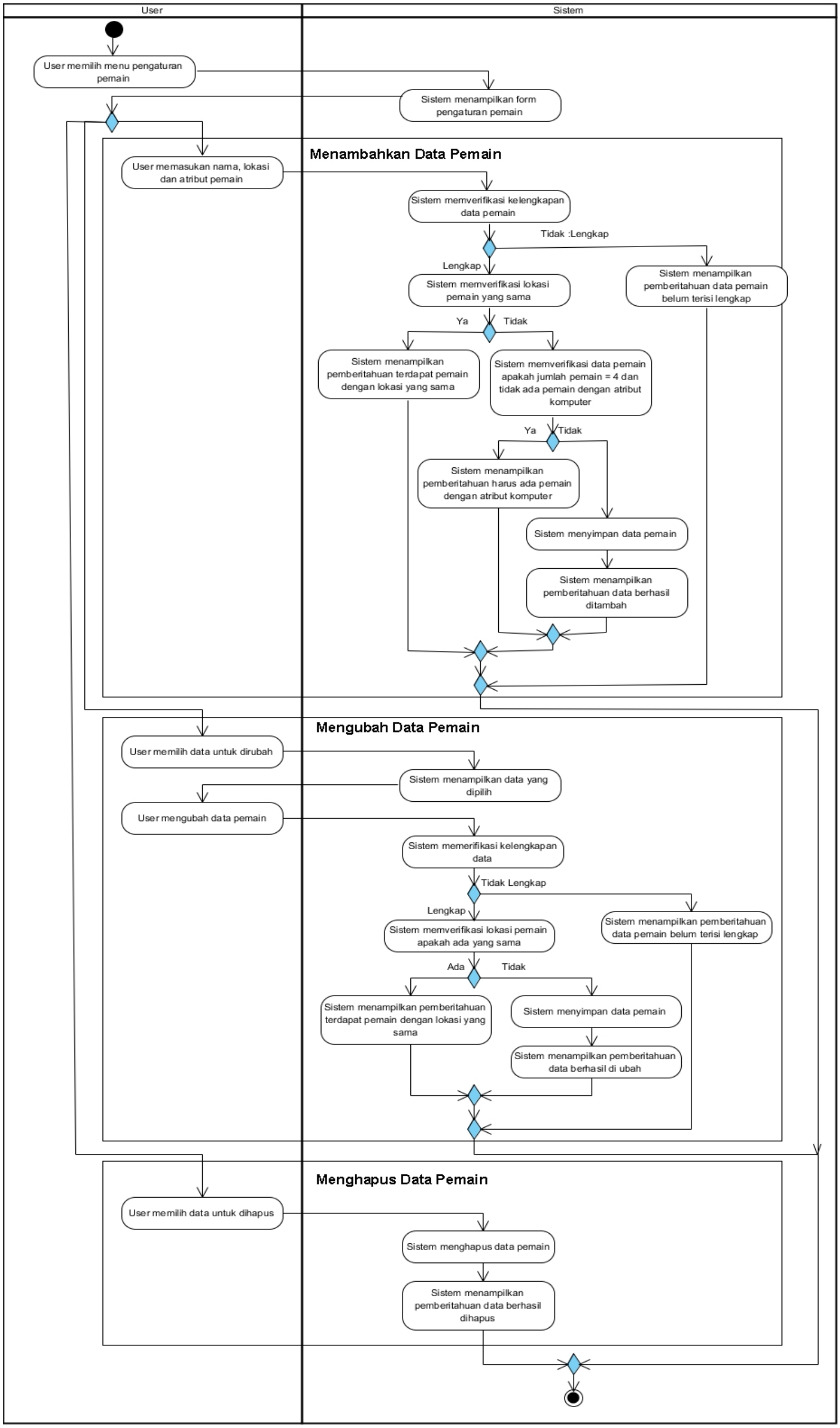
Gambar 3.24 *Activity Diagram* Mengatur Lokasi Penyimpanan Gambar *Sample*

3.4.10.2. *Activity Diagram* Mengatur Petak dan Lokasi Papan Permainan



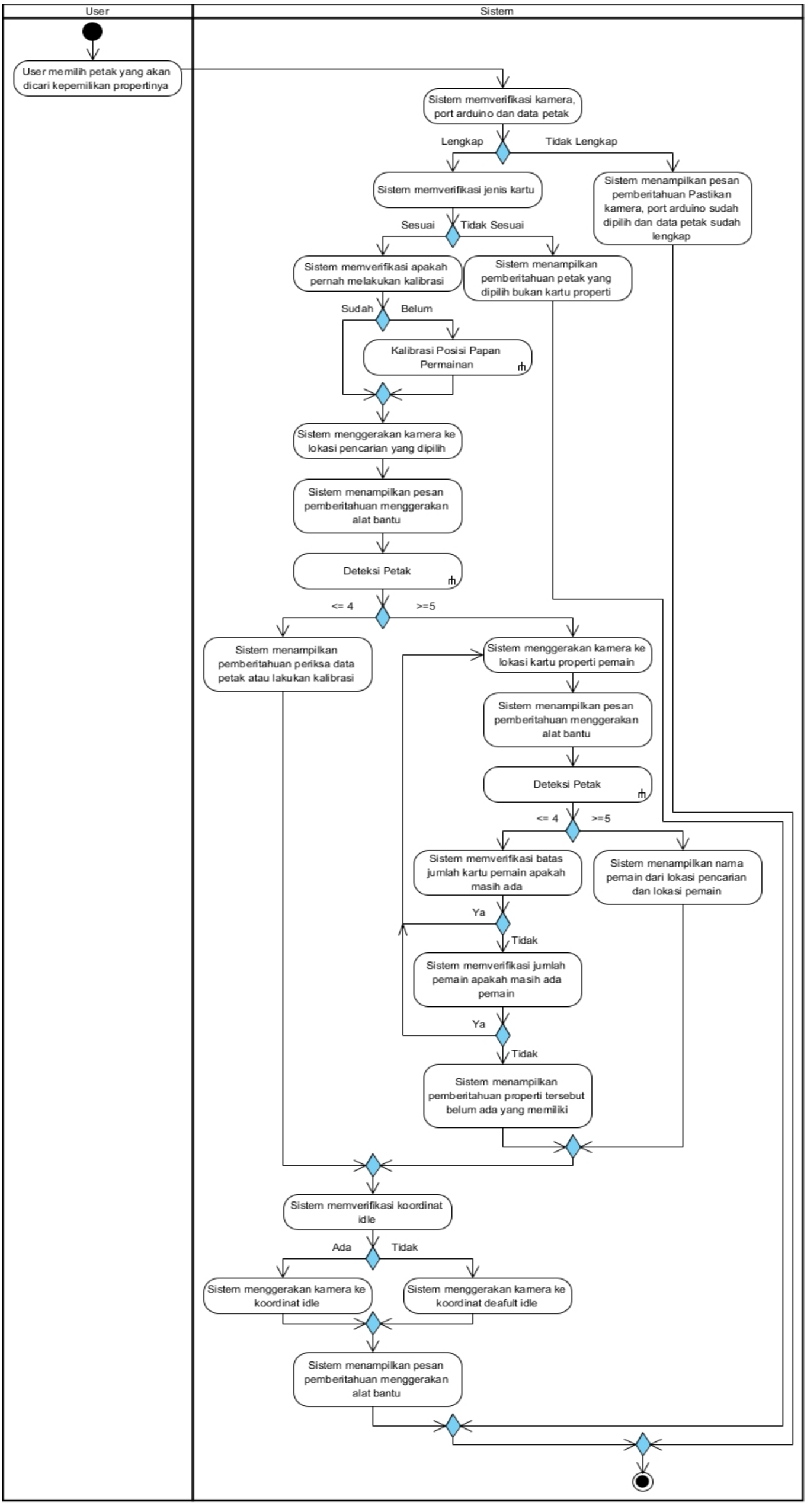
Gambar 3.25 *Activity Diagram* Mengatur Petak dan Lokasi Papan Permainan

3.4.10.3. *Activity Diagram* Mengatur Pemain



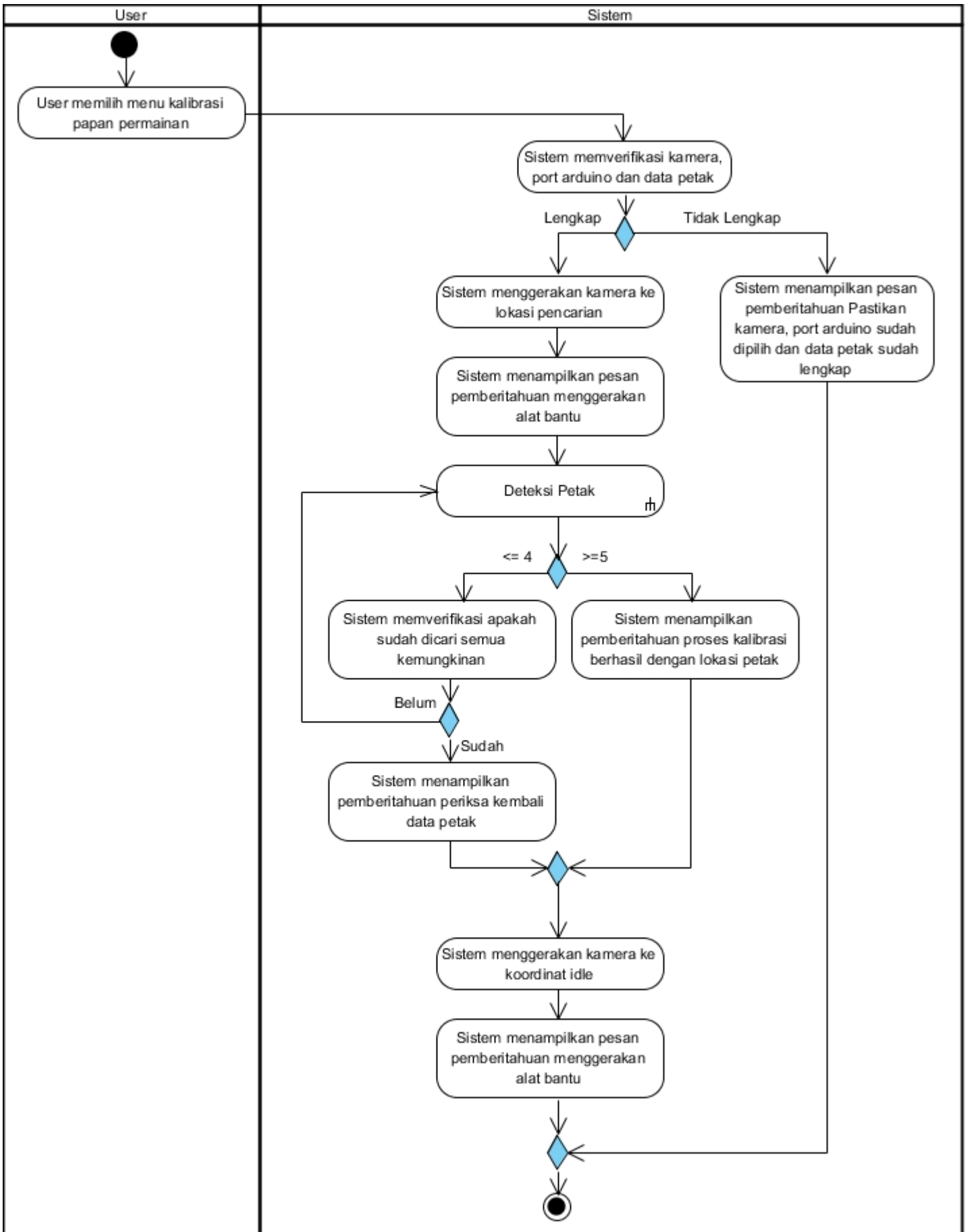
Gambar 3.26 *Activity Diagram* Mengatur Pemain

3.4.10.4. *Activity Diagram* Deteksi Kepemilikan Properti



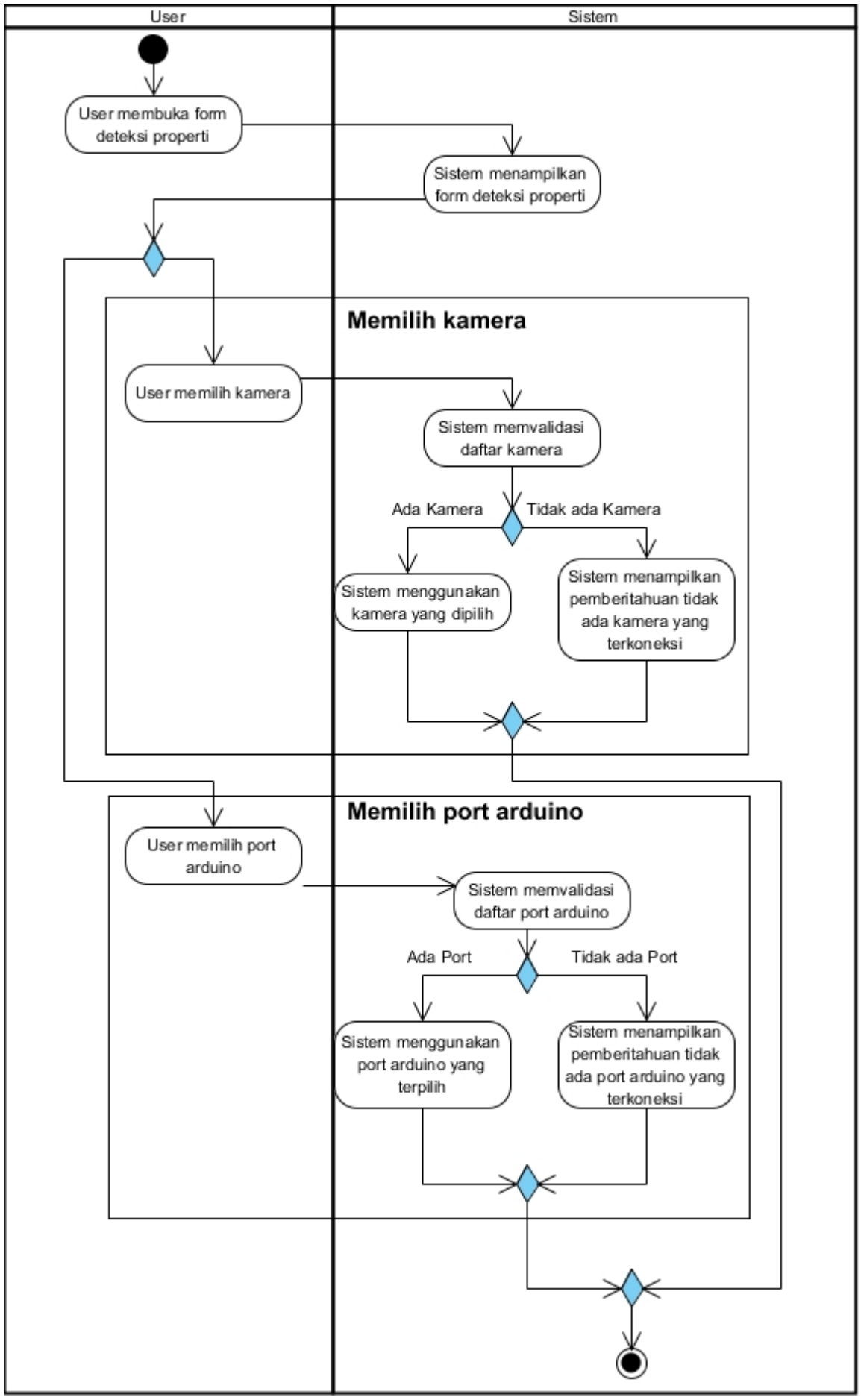
Gambar 3.27 *Activity Diagram* Deteksi Kepemilikan Properti

3.4.10.5. *Activity Diagram* Kalibrasi Posisi Papan Permainan



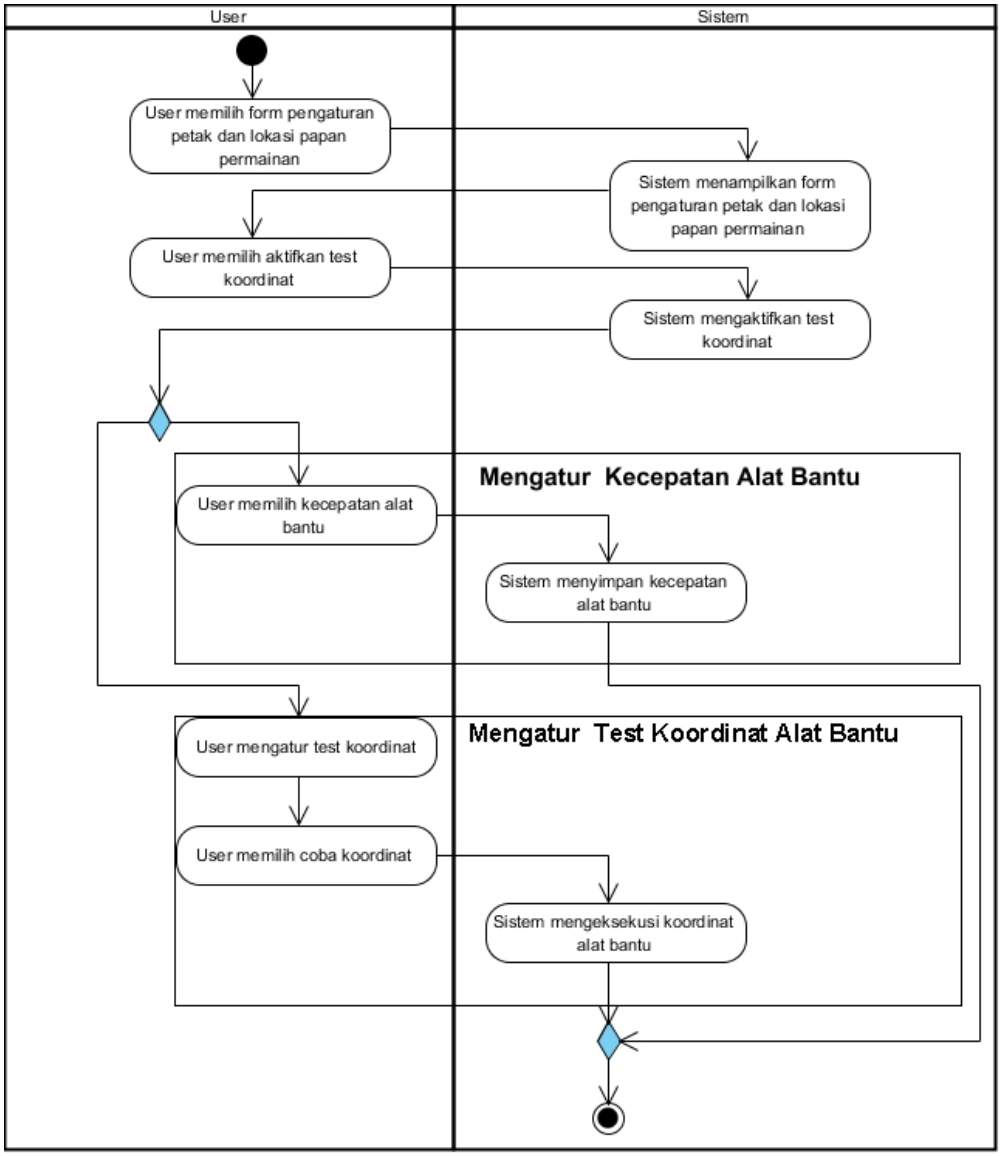
Gambar 3.28 ***Activity Diagram*** **Kalibrasi Posisi Papan Permainan**

3.4.10.6. *Activity Diagram* Mengatur Kamera dan Port *Arduino*



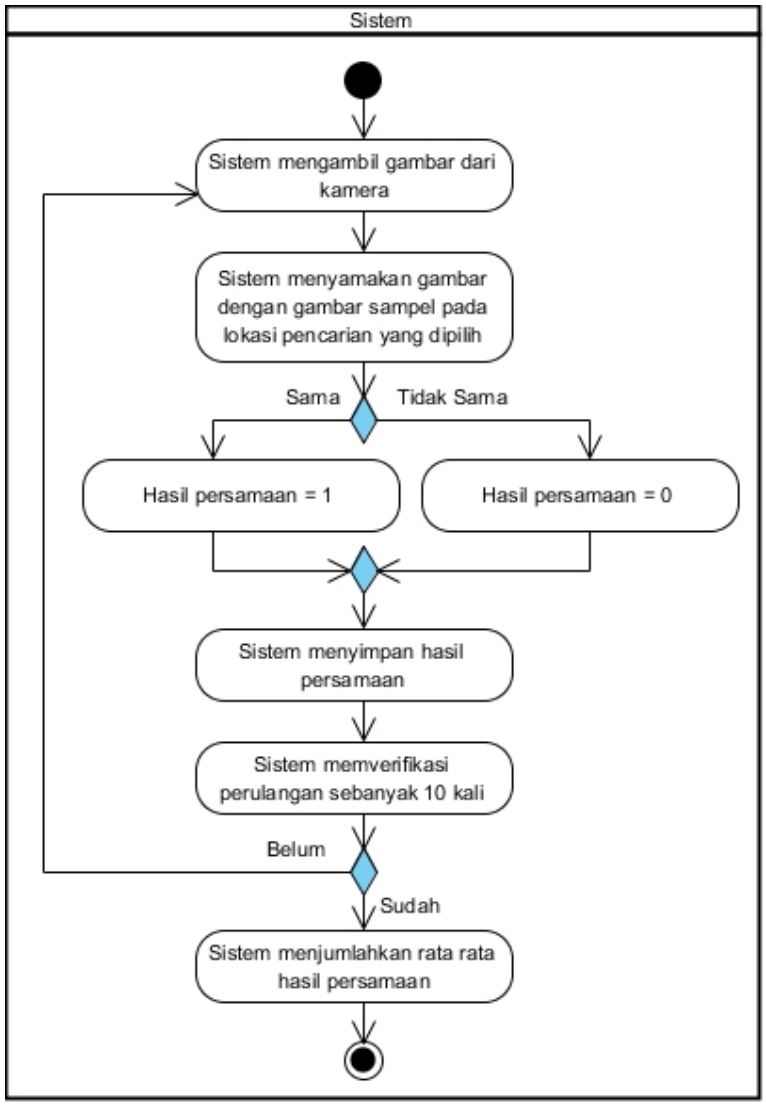
Gambar 3.29 *Activity Diagram* Mengatur Kamera dan Port *Arduino*

3.4.10.7. *Activity Diagram* Mengatur Kecepatan dan Test Koordinat Alat Bantu



Gambar 3.30 *Activity Diagram* Mengatur Kecepatan dan Test Koordinat Alat Bantu

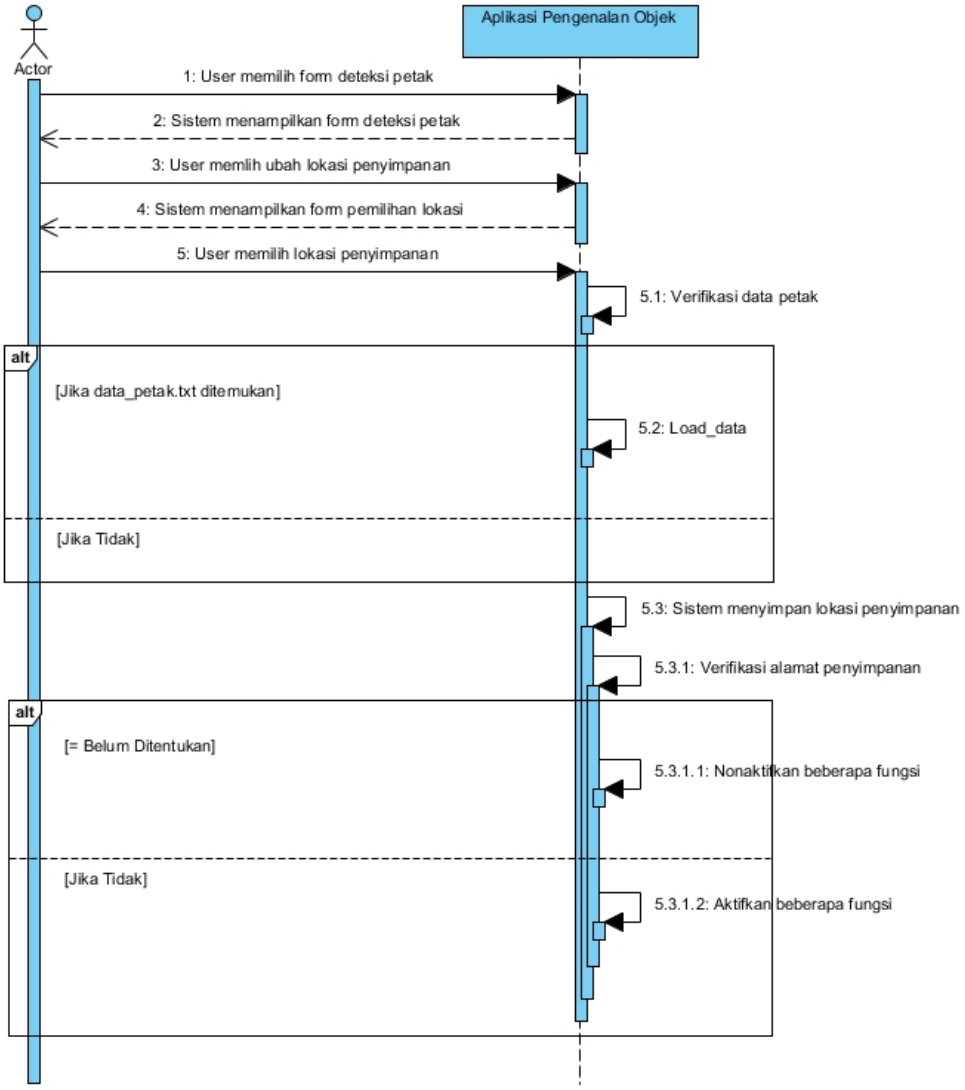
3.4.10.8. *Activity Diagram* Deteksi Petak



Gambar 3.31 ***Activity Diagram*** **Deteksi Petak**

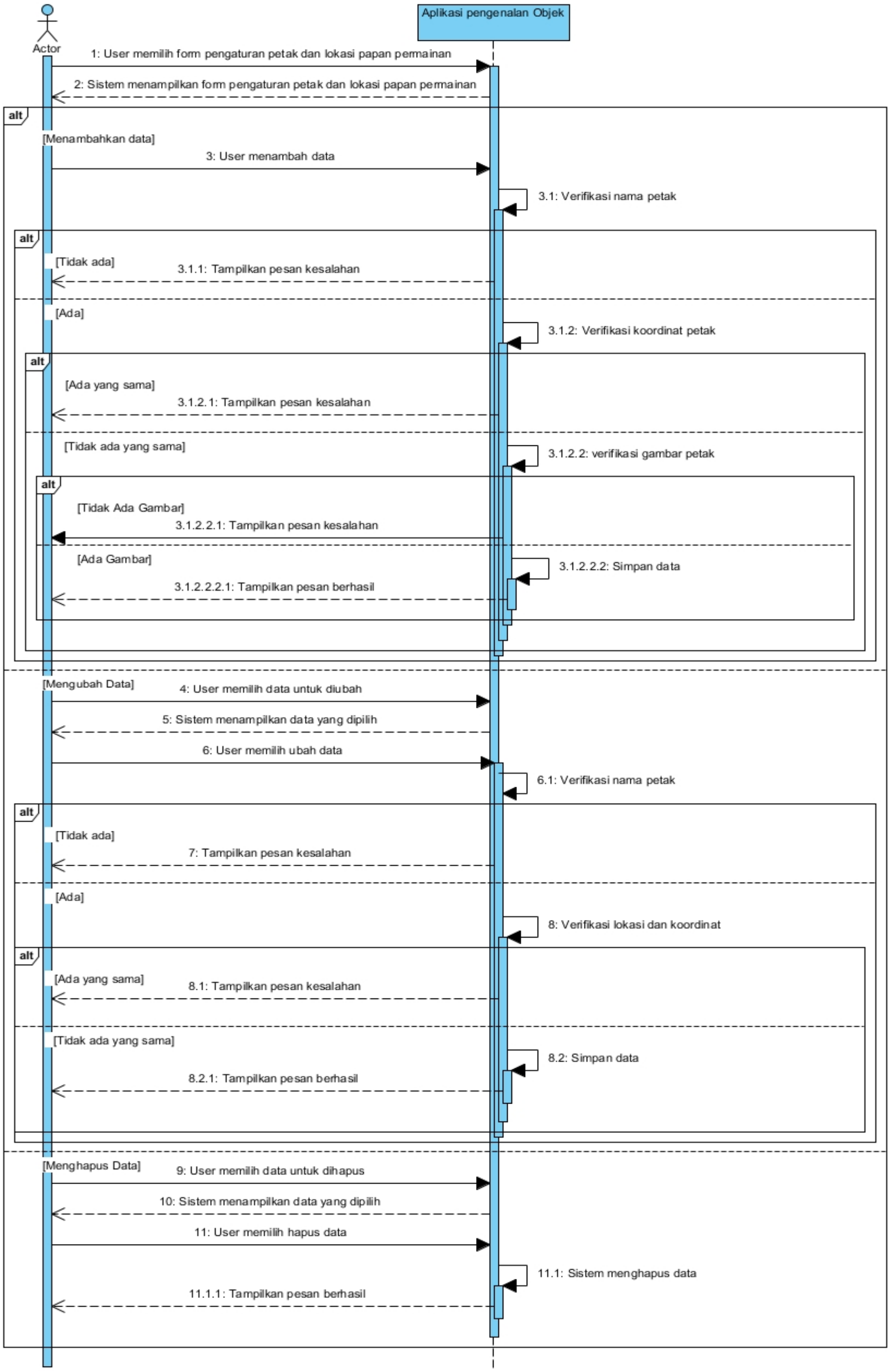
3.4.11. Sequence *Diagram*

3.4.11.1. *Sequence Diagram* Mengatur Lokasi Penyimpanan Gambar Sampel



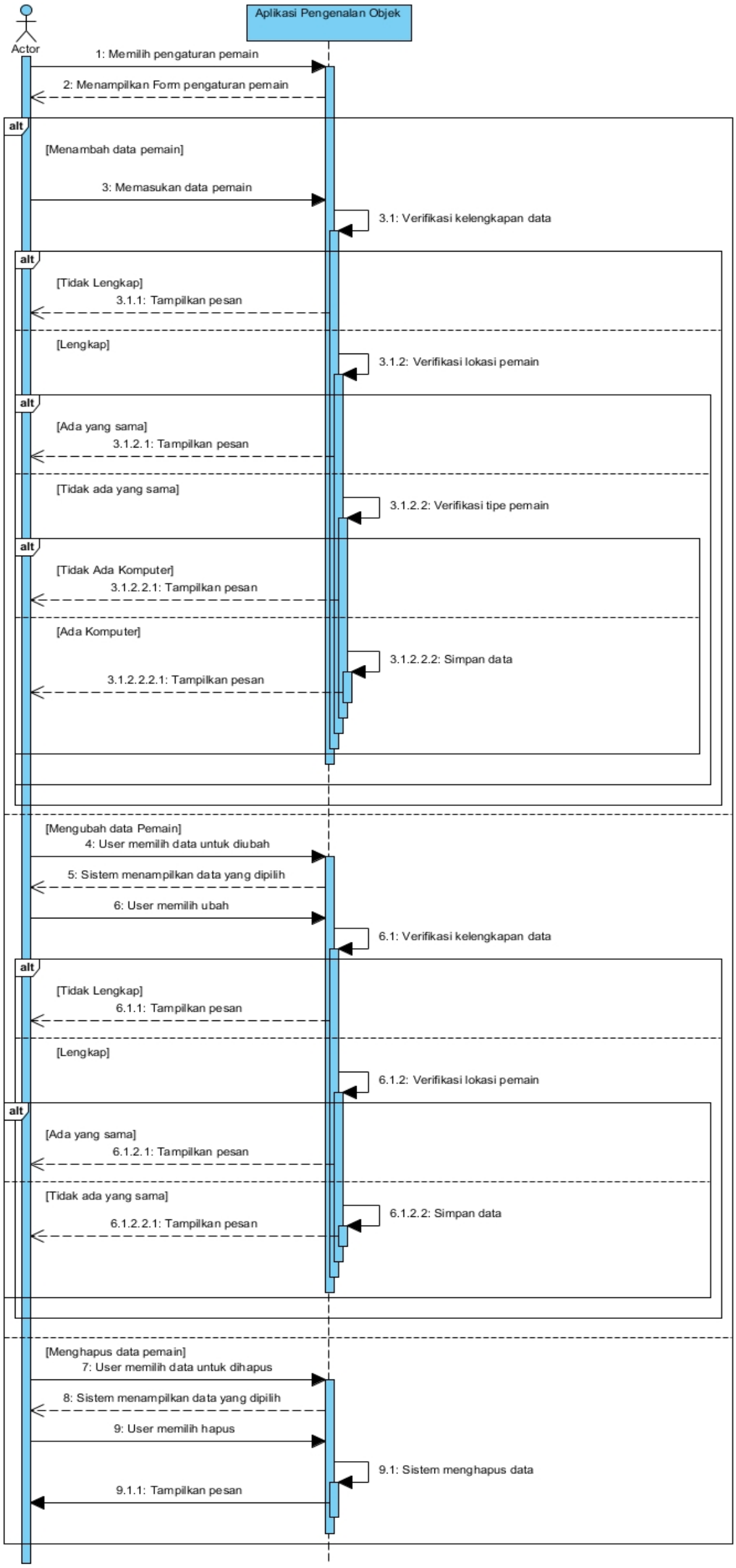
Gambar 3.32 *Sequence* Mengatur Lokasi Penyimpanan Gambar Sampel

3.4.11.2. *Sequence Diagram* Mengatur Petak dan Lokasi Papan Permainan



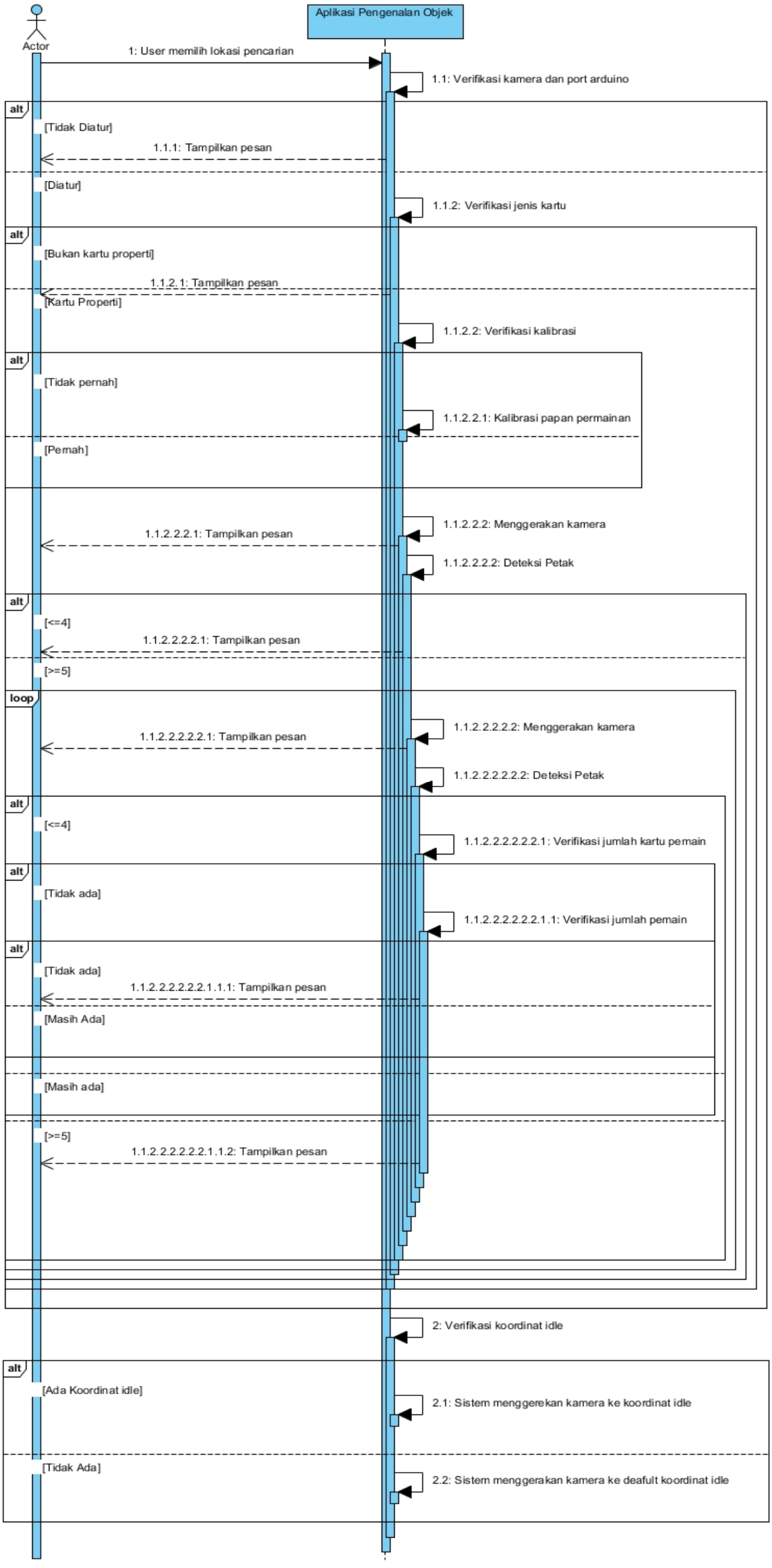
Gambar 3.33 *Sequence* Mengatur Petak dan Lokasi Papan Permainan

3.4.11.3. *Sequence Diagram* Mengatur Pemain



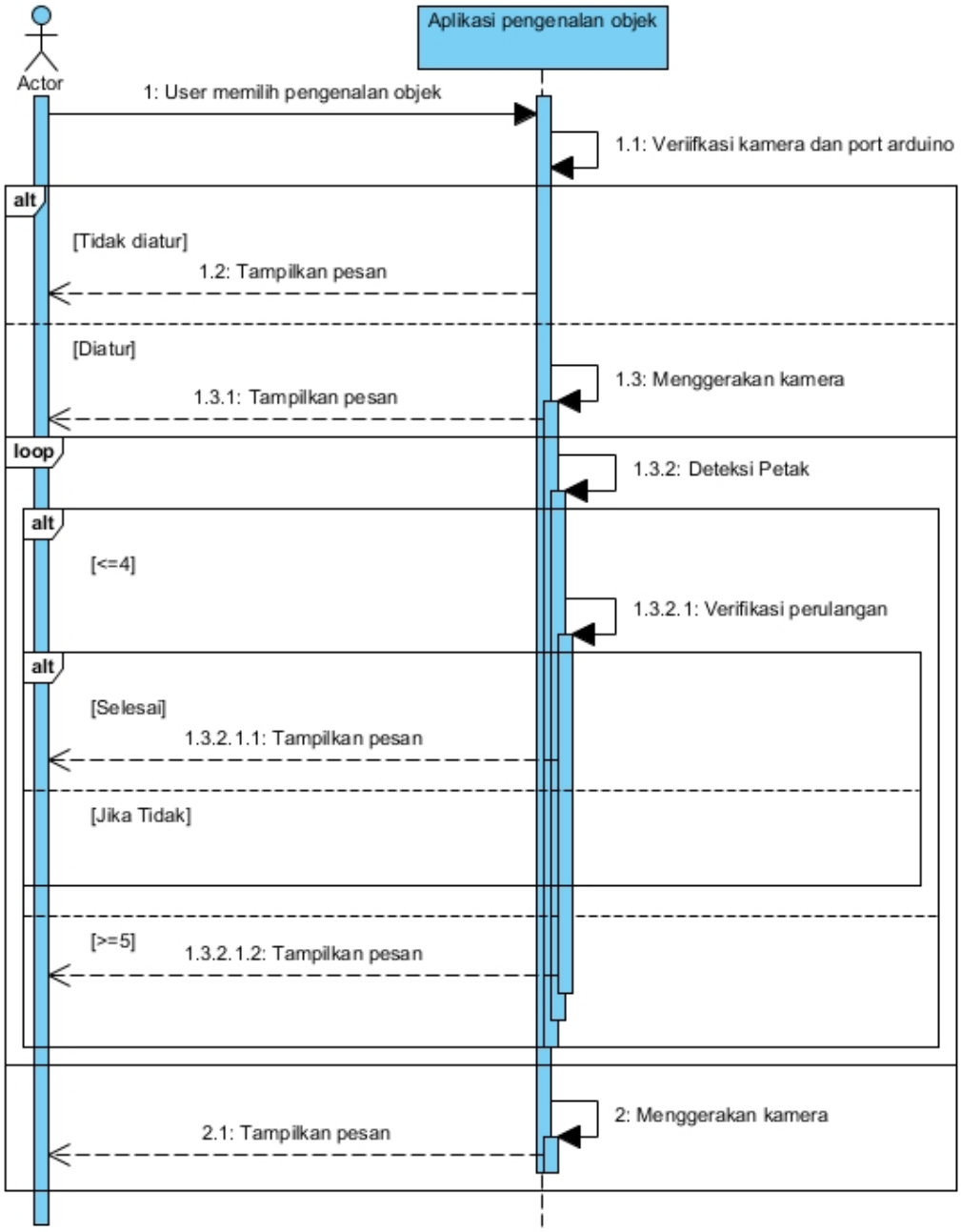
Gambar 3.34 *Sequence* Mengatur Pemain

3.4.11.4. *Sequence Diagram* Deteksi Kepemilikan Properti



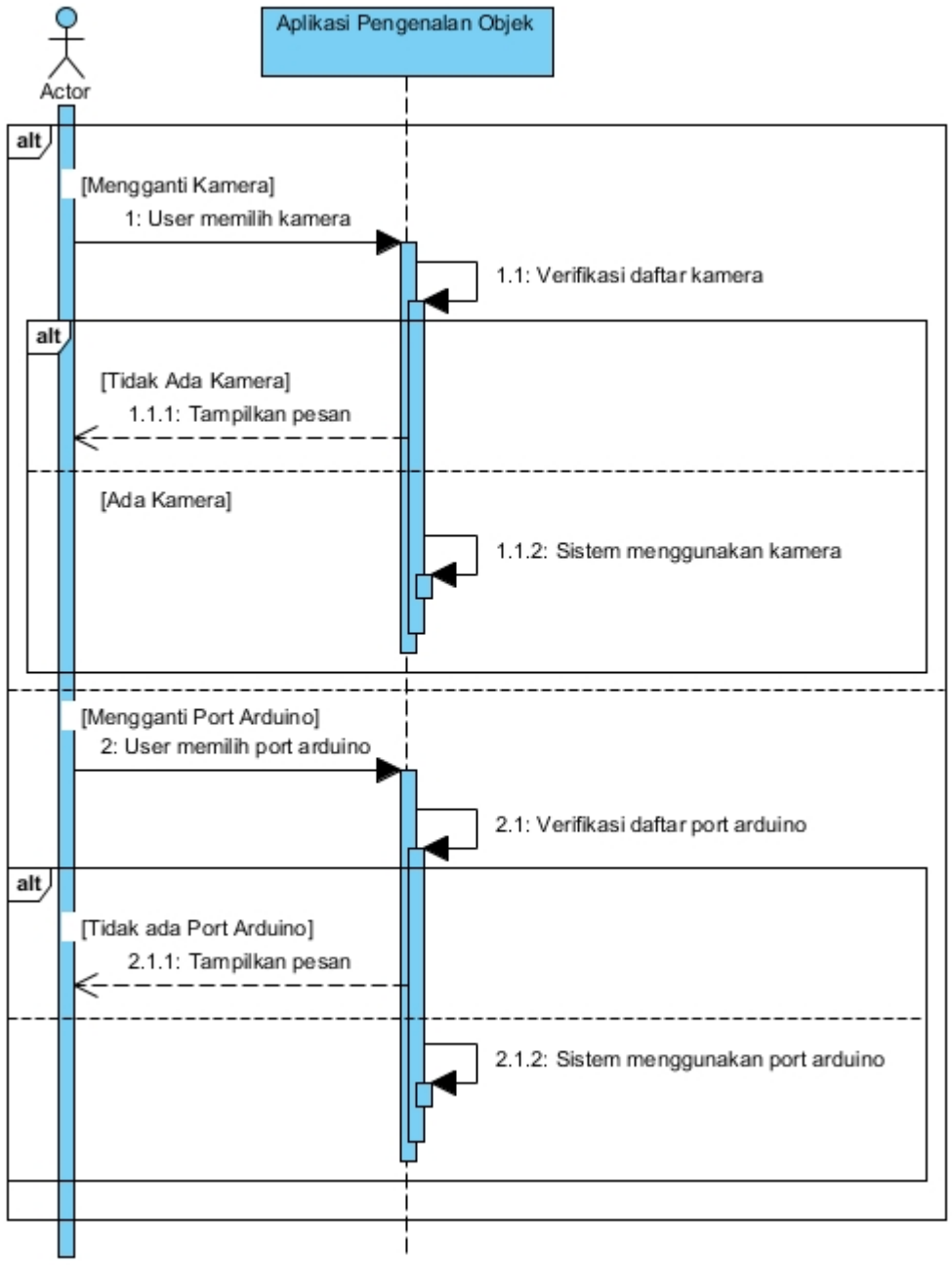
Gambar 3.35 *Sequence* Deteksi Kepemilikan Properti

3.4.11.5. *Sequence Diagram* Mengatur Lokasi Penyimpanan Gambar Sampel



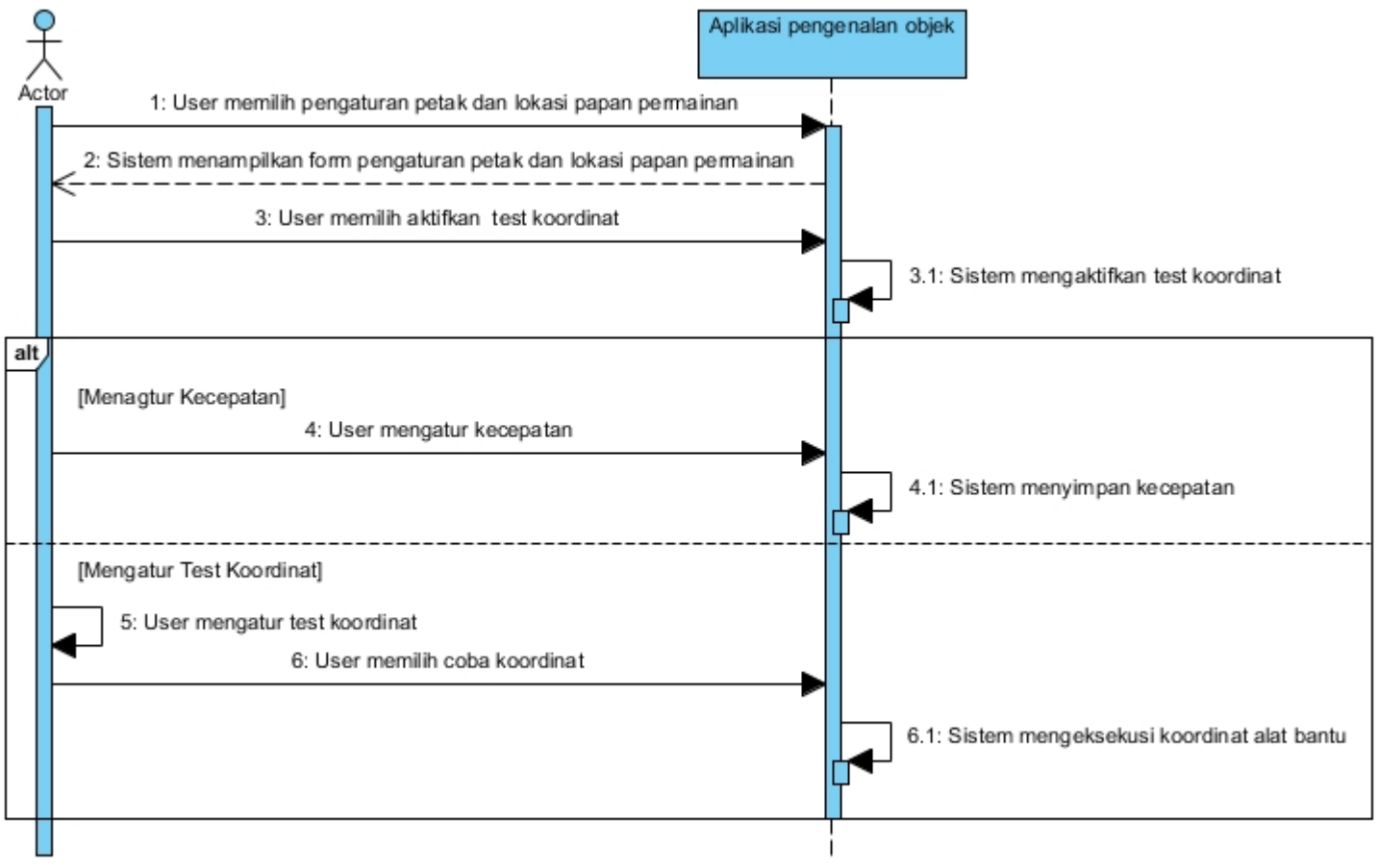
Gambar 3.36 *Sequence* Mengatur Lokasi Penyimpanan Gambar Sampel

3.4.11.6. *Sequence Diagram* Mengatur Kamera dan Port *Arduino*



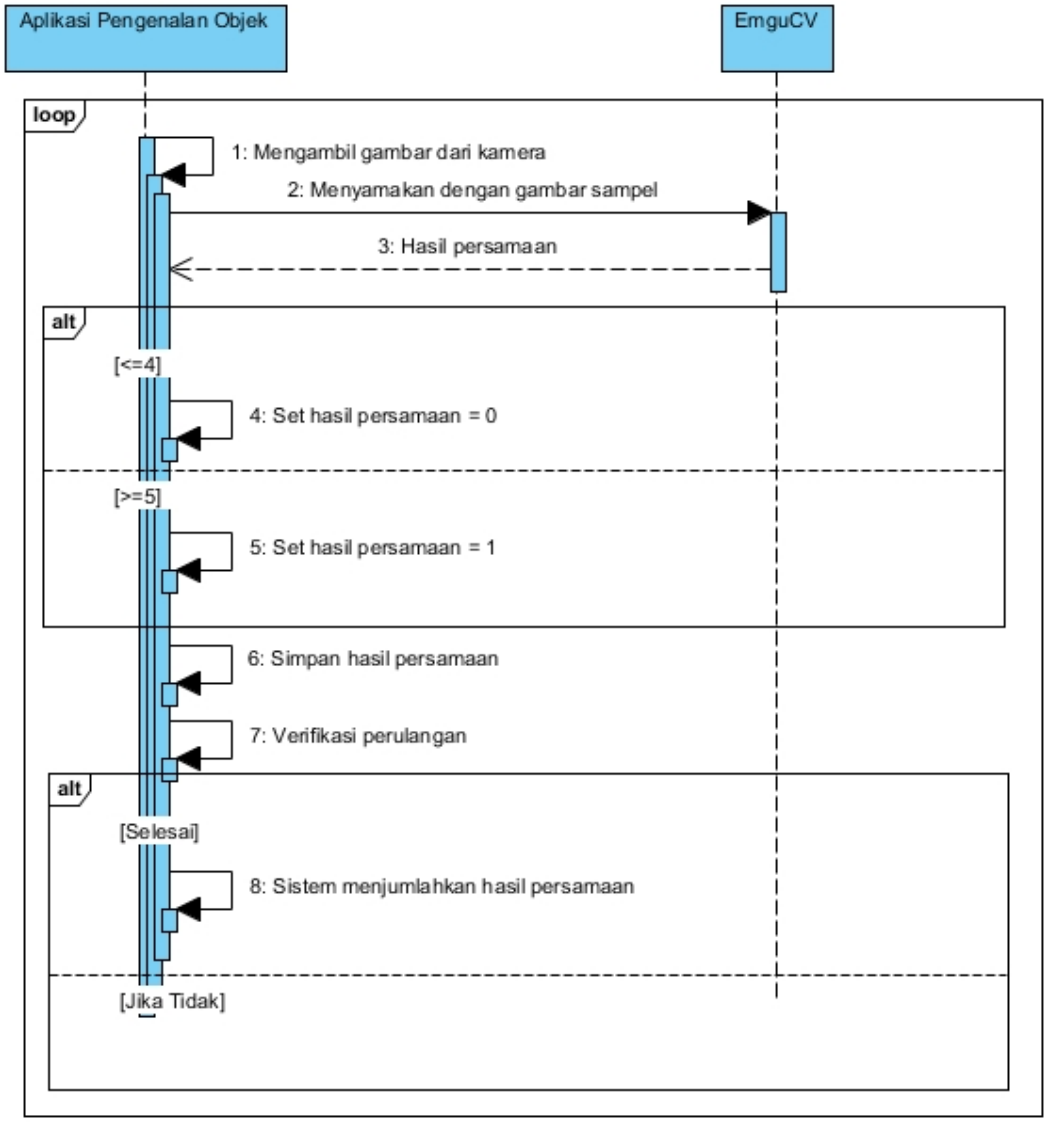
Gambar 3.37 *Sequence* Mengatur Kamera dan Port *Arduino*

3.4.11.7. *Sequence Diagram* Mengatur Kecepatan dan Test Koordinat



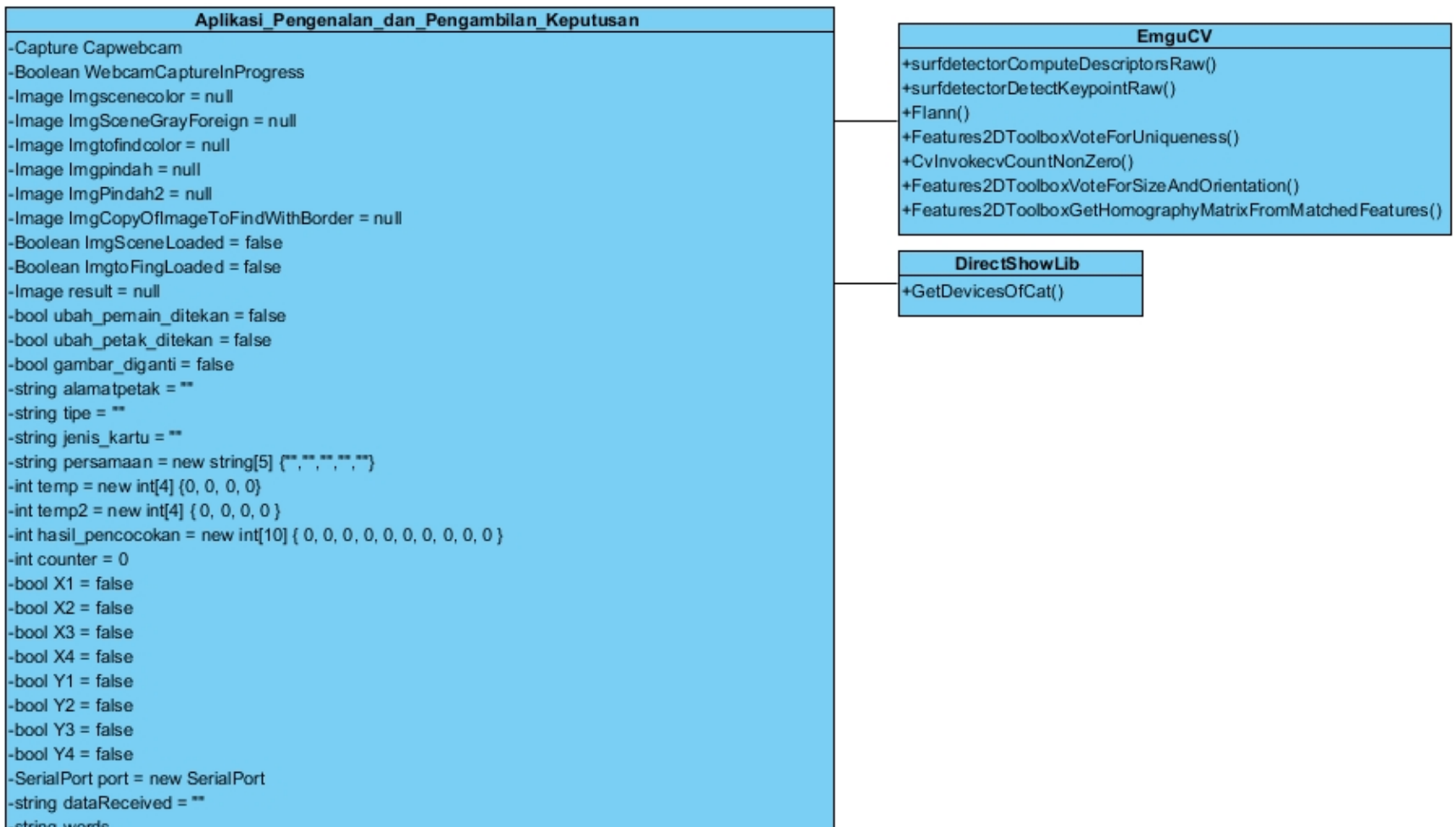
Gambar 3.38 *Sequence* Mengatur Kecepatan dan Test Koordinat

3.4.11.8. *Sequence Diagram* Deteksi Petak



Gambar 3.39 *Sequence* Deteksi Petak

3.4.12. Class Diagram

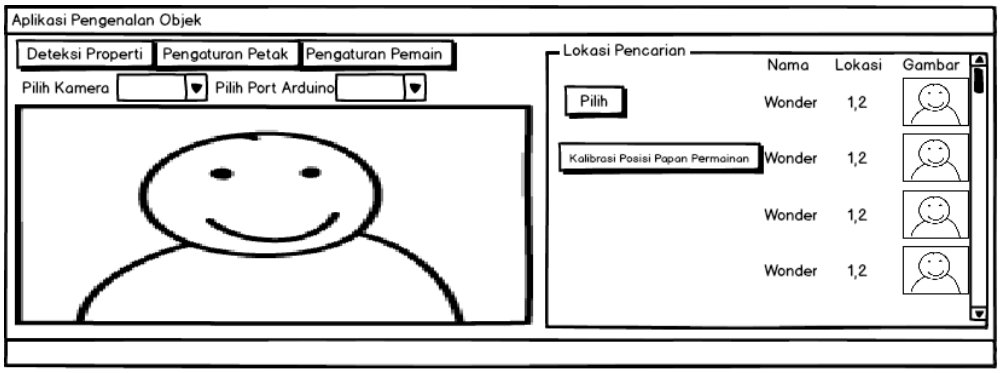


Gambar 3.40 ***Class Diagram***

3.5. Perancangan Sistem

3.5.1. Rancangan Antar Muka

3.5.1.1. Rancangan Menu Awal Pengenalan Objek



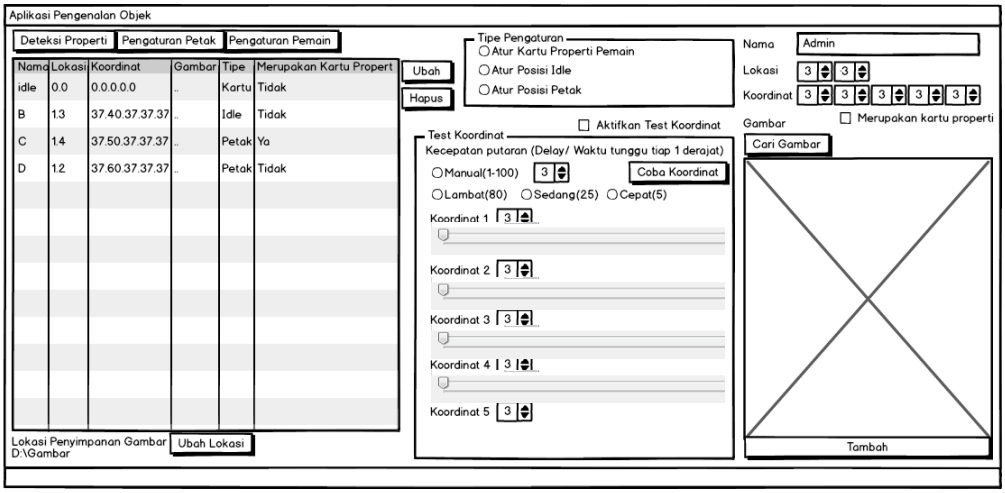
Gambar 3.41 Rancangan Menu Awal Pengenalan Objek

Tabel 3.11 Penjelasan Rancangan Menu Awal Pengenalan Objek

Nama tombol	Jenis	Penjelasan
Deteksi Properti	*Button*	Tombol untuk menampilkan form deteksi properti
Pengaturan Petak	*Button*	Tombol untuk menampilkan form Pengaturan Petak
Pengaturan Pemain	*Button*	Tombol untuk menampilkan form Pengaturan Pemain
Pilih	*Button*	Tombol untuk mendeteksi kepemilkan dari properti
Kalibrasi Posisi Papan Permainan	*Button*	Tombol untuk melakukan kalibrasi posisi papan permainan
List Kamera	*ListBox*	*List* kamera untuk dipilih

Nama tombol	Jenis	Penjelasan
List Port Arduino	*ListBox*	*List Port Arduino* untuk dipilih
Lokasi Pencarian	*DataGridView*	Gambar yang dapat dipilih oleh *user* untuk menentukan lokasi yang akan dicari
Gambar Kamera	*ImageBox*	Tampilan gambar dari kamera

3.5.1.2. Rancangan Menu Pengaturan Petak



Gambar 3.42 Rancangan Menu Pengaturan Petak

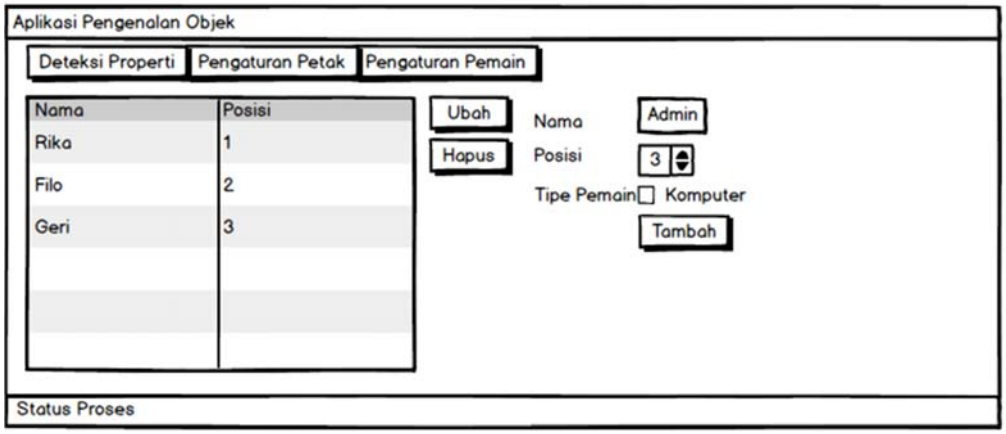
Tabel 3.12 Penjelasan Rancangan Menu Pengaturan Petak

Nama	Jenis	Penjelasan
Deteksi Properti	*Button*	Tombol untuk menampilkan form deteksi properti
Pengaturan Petak	*Button*	Tombol untuk menampilkan form Pengaturan Petak

Nama	Jenis	Penjelasan
Pengaturan Pemain	*Button*	Tombol untuk menampilkan form Pengaturan Pemain
Tabel data petak	*DataGridView*	Tabel yang berisi informasi mengenai petak seperti lokasi, nama, gambar, koordinat, tipe dan merupakan kartu properti atau tidak
Ubah	*Button*	Tombol Ubah
Hapus	*Button*	Tombol Hapus
Nama	*TextBox*	Nama dari petak yang dipilih, atau nama dari petak yang akan didaftarkan
Lokasi	*NumericUpDown*	Lokasi dari petak yang dipilih, atau lokasi dari petak yang akan didaftarkan
Koordinat	*NumericUpDown*	Koordinat dari petak yang dipilih, atau koordinat dari petak yang akan didaftarkan
Merupakan Kartu Properti	*CheckBox*	Bertujuan untung membedakan antara petak biasa dengan petak properti
Cari Gambar	*Button*	Tombol Pencarian Gambar yang akan didaftarkan kedalam sistem

Nama	Jenis	Penjelasan
Tambah	*Button*	Tombol Penambahan gambar yang akan didaftarkan kedalam sistem
Tipe pengaturan	*Radio Button*	Tombol yang membedakan pengaturan yang dilakukan oleh *user*
Aktifkan Test Koordinat	*CheckBox*	Untuk mengaktifkan fungsi test koordinat
Delay	*Radio Button*	Delay yang dapat dipilih untuk mengatur kecepatan alat bantu
Manual Delay	*Numeric Up Down*	Untuk mengisi nilai delay secara manual
KoordinatNUP	*Numeric Up Down*	Untuk mengganti nilai koordinat menggunakan *Numeric Up Down*
KoordinatTB	*TrackBar*	Untuk mengganti nilai koordinat menggunakan *TrackBar*
Test Koordinat	*Button*	Untuk melakukan test koordinat sesuai masukan dari *user*

3.5.1.3. Rancangan Menu Pengaturan Pemain



Gambar 3.43 Rancangan Menu Pengaturan Pemain

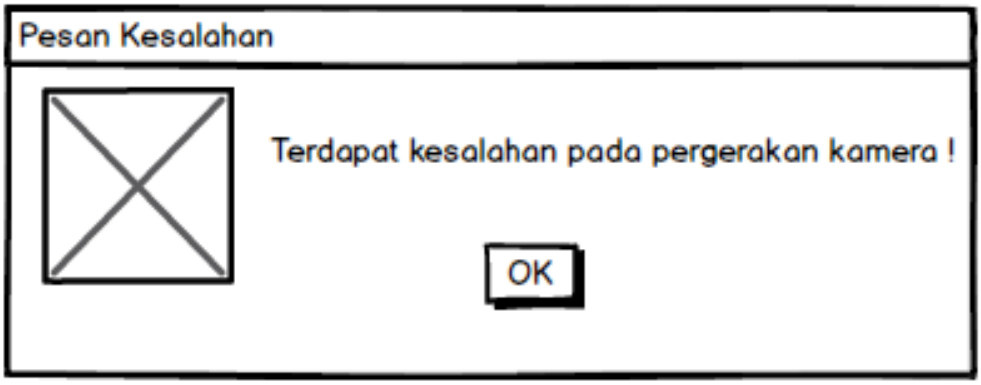
Tabel 3.13 Penjelasan Rancangan Menu Pengaturan Pemain

Nama	Jenis	Penjelasan
Deteksi Properti	*Button*	Tombol untuk menampilkan form deteksi properti
Pengaturan Petak	*Button*	Tombol untuk menampilkan form Pengaturan Petak
Pengaturan Pemain	*Button*	Tombol untuk menampilkan form Pengaturan Pemain
Tabel data pemain	*DataGridView*	Tabel yang berisi informasi mengenai pemain seperti nama dan posisi pemain
Ubah	*Button*	Tombol Ubah
Hapus	*Button*	Tombol Hapus
Nama	*TextBox*	Nama dari petak yang dipilih, atau nama dari

Nama	Jenis	Penjelasan
		petak yang akan didaftarkan
Posisi	*NumericUpDown*	Posisi dari pemain dari 1 – 4, 1 adalah posisi *horizontal* pada papan permainan yang terdapat petak go di posisi paling kanan,2 adalah posisi selanjutnya dari posisi 1 searah jarum jam
Tambah	*Button*	Tombol Penambahan pemain yang akan didaftarkan kedalam sistem
Tipe Pemain	*CheckBox*	Untuk membedakan yang mana komputer dan bukan komputer

3.5.2. Perancangan Pesan

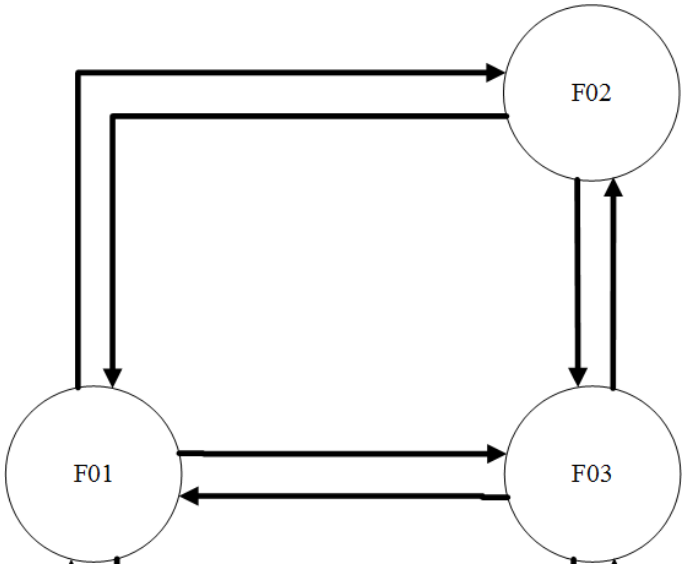
Perancangan pesan yang digunakan dalam sistem ini adalah pesan seperti Gambar 3.44.



Gambar 3.44 Perancangan Pesan

3.5.3. Jaringan Semantik

Jaringan semantik ini menunjukan form form yang bisa diakses oleh pengguna seperti ditunjukan pada Gambar 3.45 dibawah ini



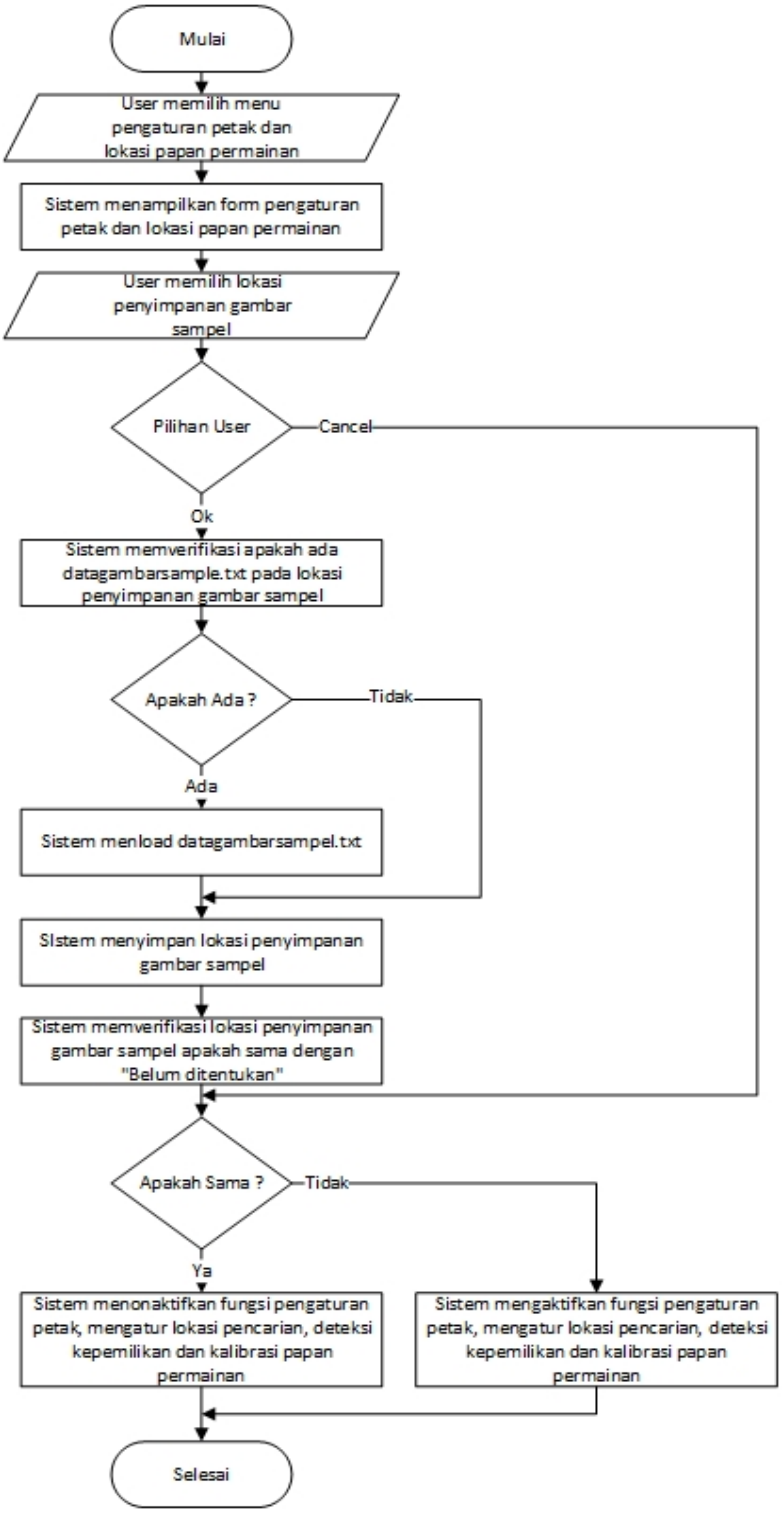
Gambar 3.45 Jaringan Semantik

Nama	Keterangan
F01	Tampilan Menu Deteksi Properti
F02	Tampilan Menu Pengaturan Petak dan Lokasi Papan Permainan
F03	Tampilan Menu Pengaturan Pemain

3.5.4. Perancangan Prosedural

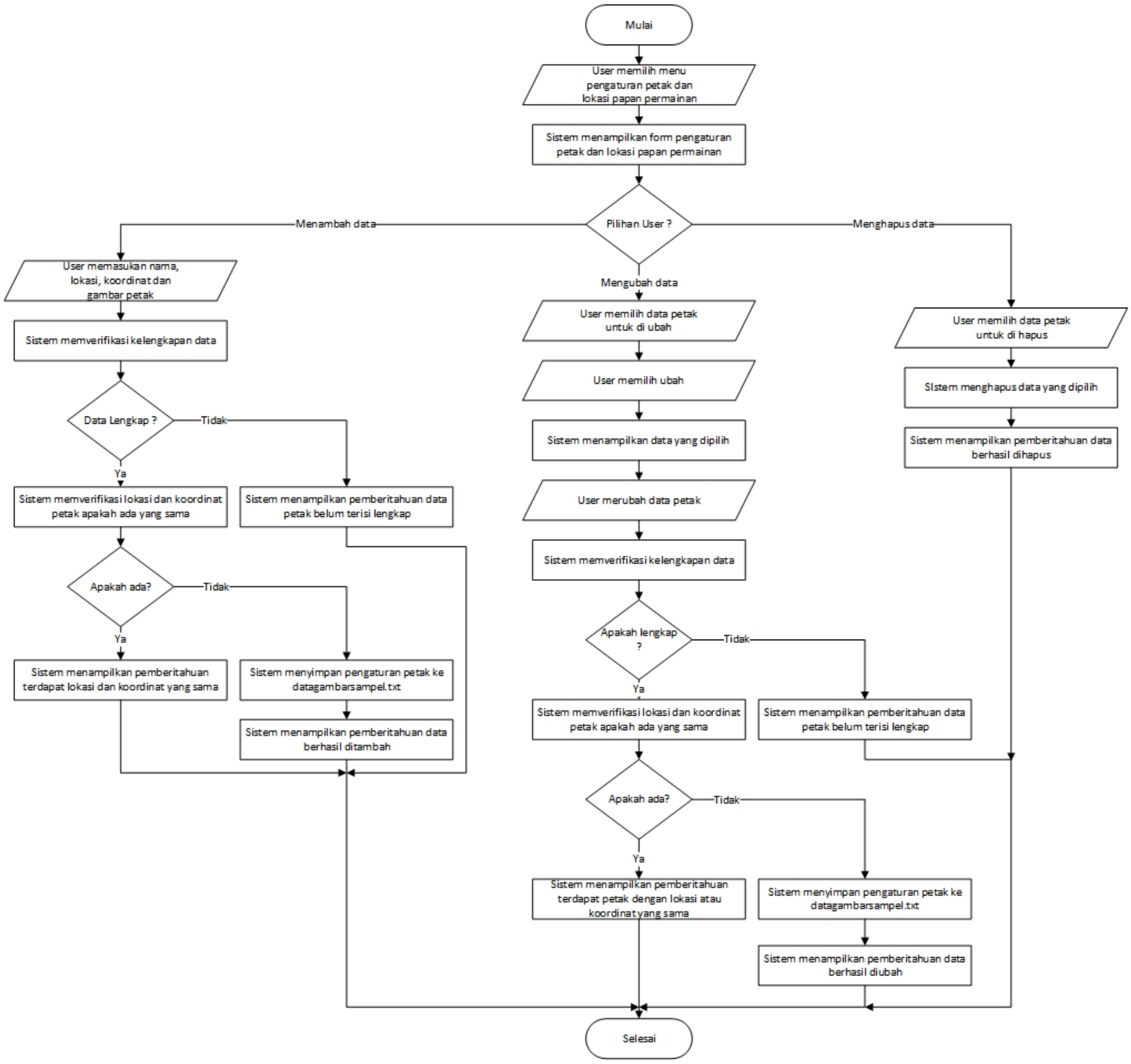
Flowchart adalah penggambaran secara grafik dari langkah-langkah dan urutan-urutan prosedur dari suatu aplikasi atau program sehingga pembuatan aplikasi dapat dilakukan secara terurut dari awal sampai akhir. Berikut adalah *flowchart* dari Sistem Pengenalan Objek Pada Permainan Monopoli Menggunakan Algoritma *Speeded Up Robust Feature*s.

3.5.4.1. *Flowchart* Mengatur Lokasi Penyimpanan Gambar *Sample*



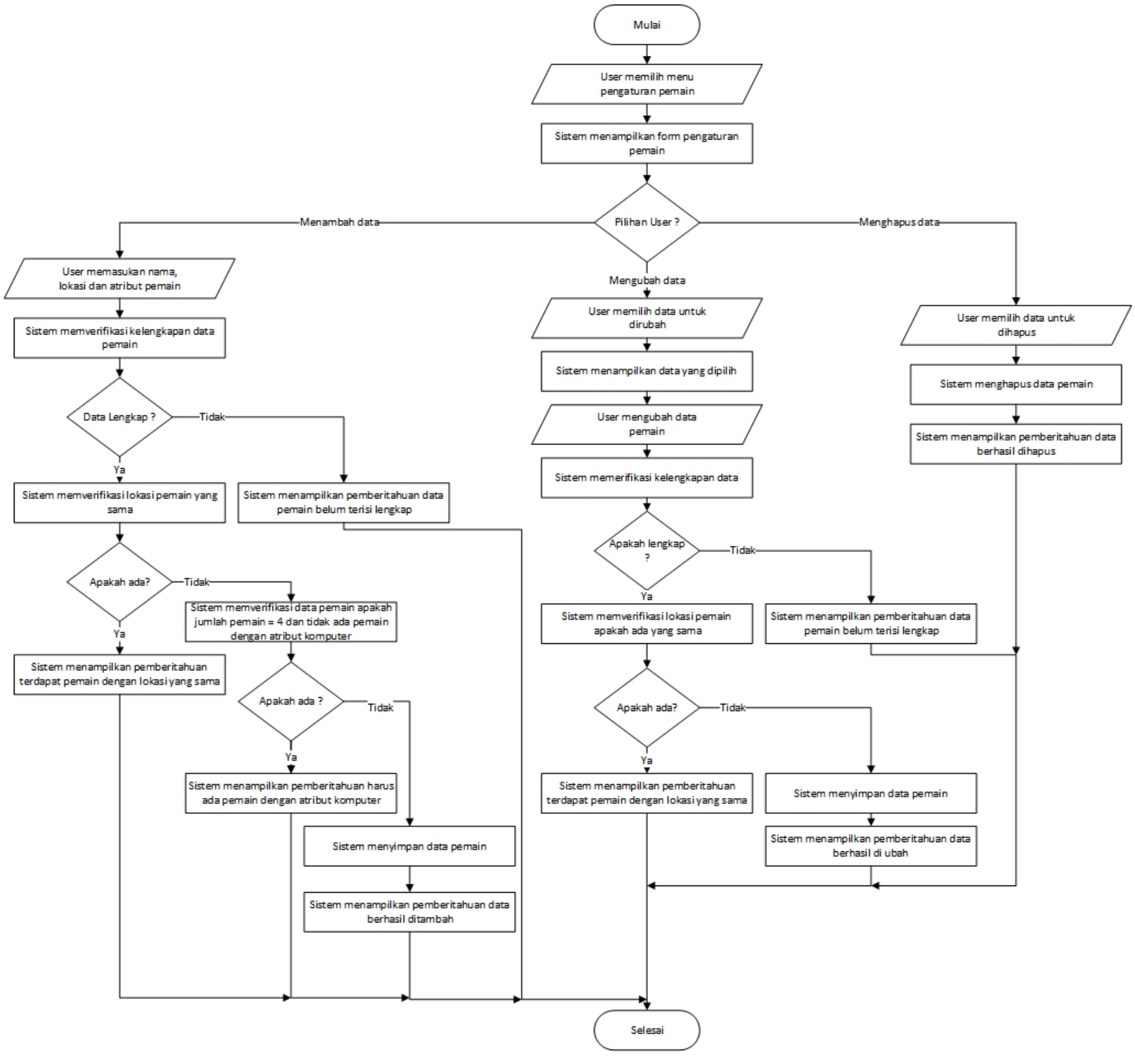
Gambar 3.46 *Flowchart* Mengatur Lokasi Penyimpanan Gambar *Sample*

3.5.4.2. *Flowchart* Mengatur Petak dan Lokasi Papan Permainan



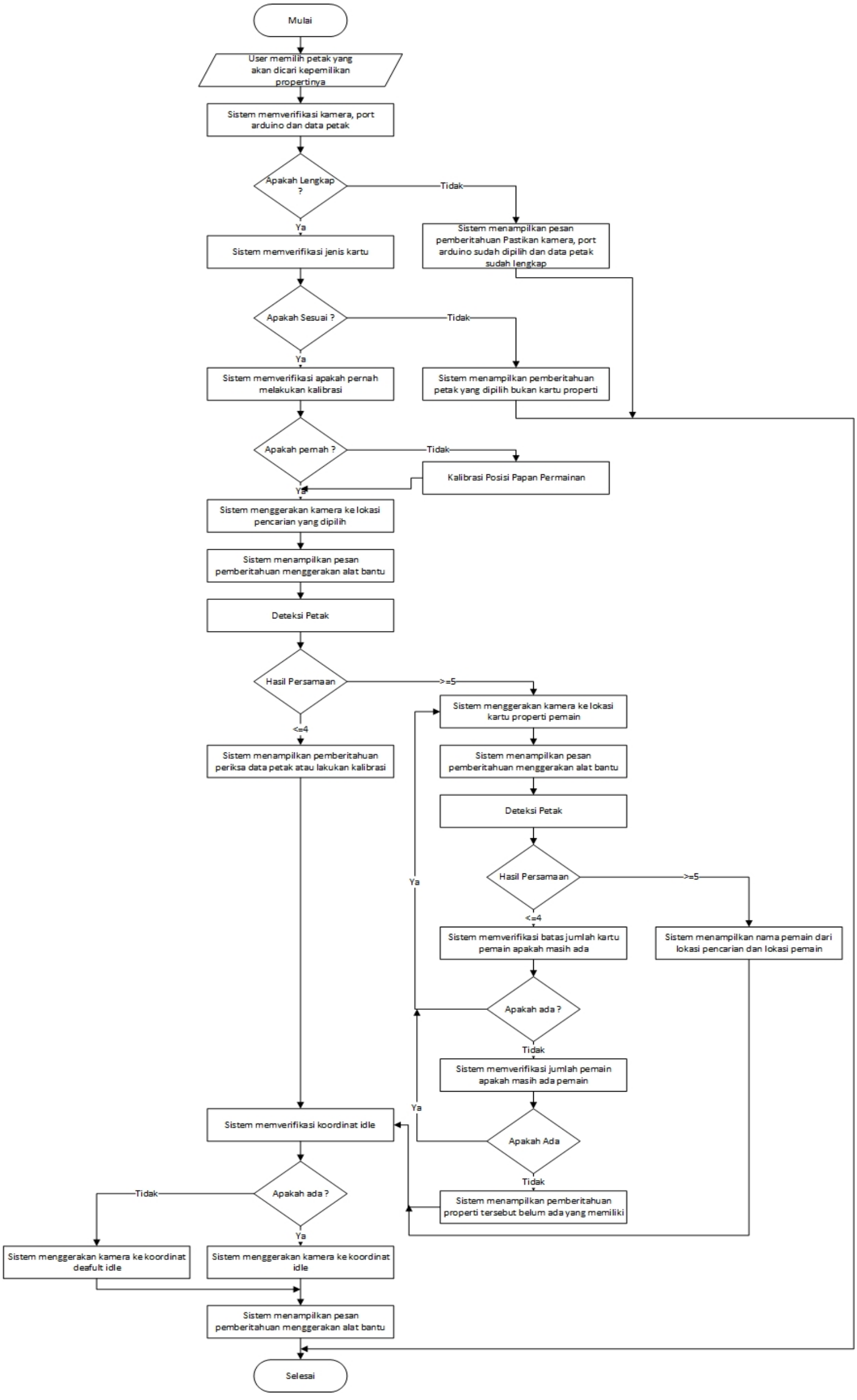
Gambar 3.47 *Flowchart* Mengatur Petak Papan Permainan

3.5.4.3. *Flowchart* Mengatur Pemain



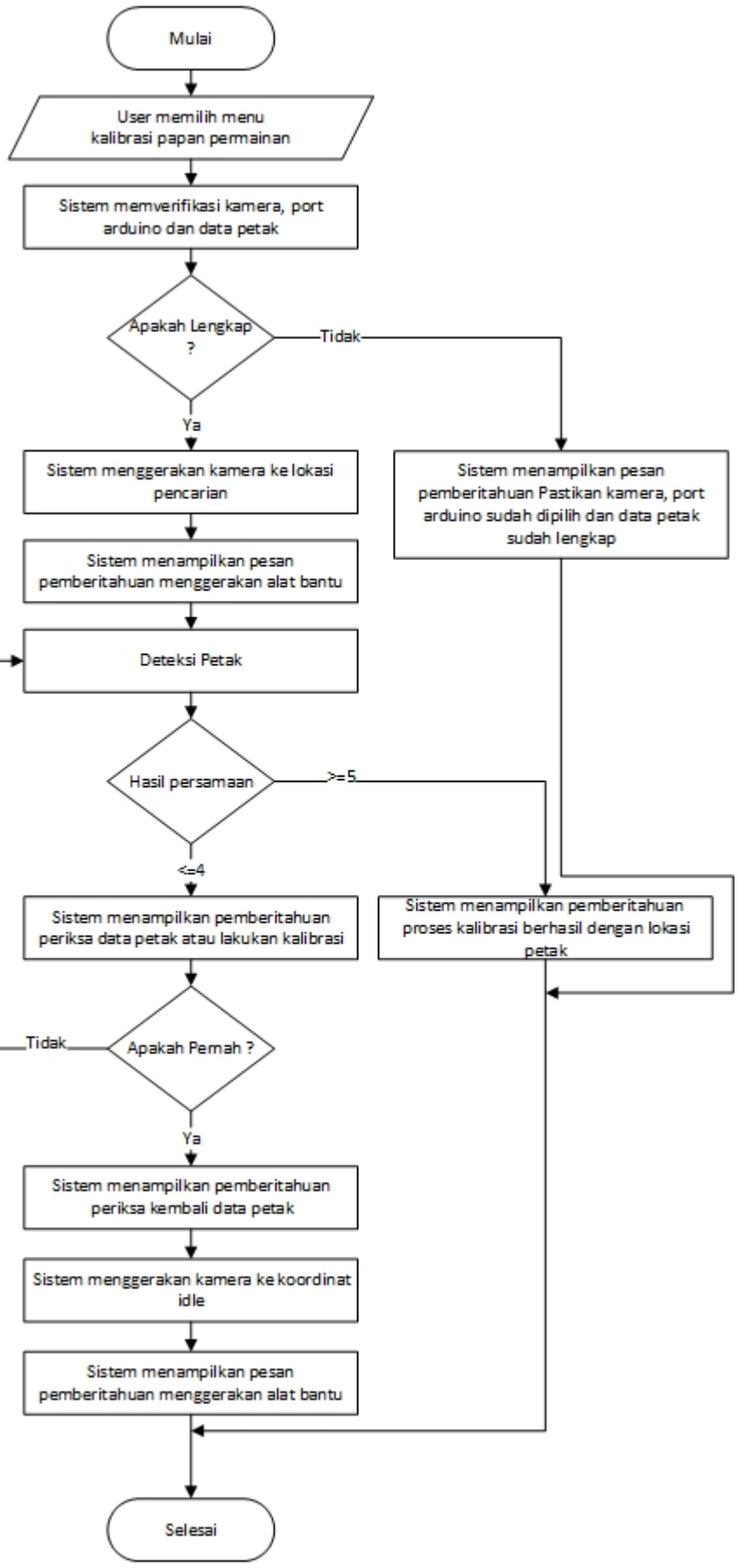
Gambar 3.48 *Flowchart* Mengatur Pemain

3.5.4.4. *Flowchart* Deteksi Kepemilikan Properti



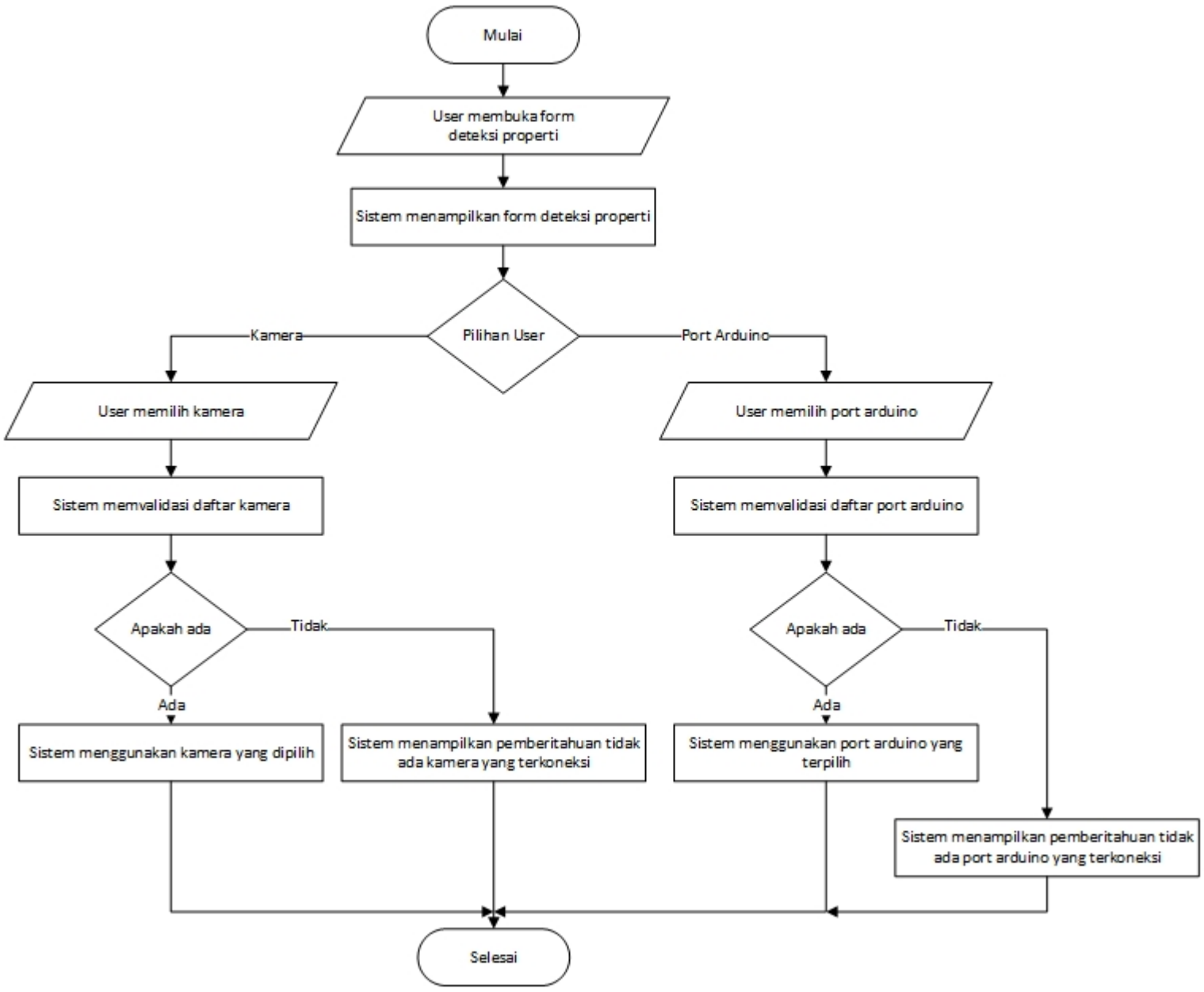
Gambar 3.49 *Flowchart* Deteksi Kepemilikan Properti

3.5.4.5. *Flowchart* Kalibrasi Posisi Papan Permainan



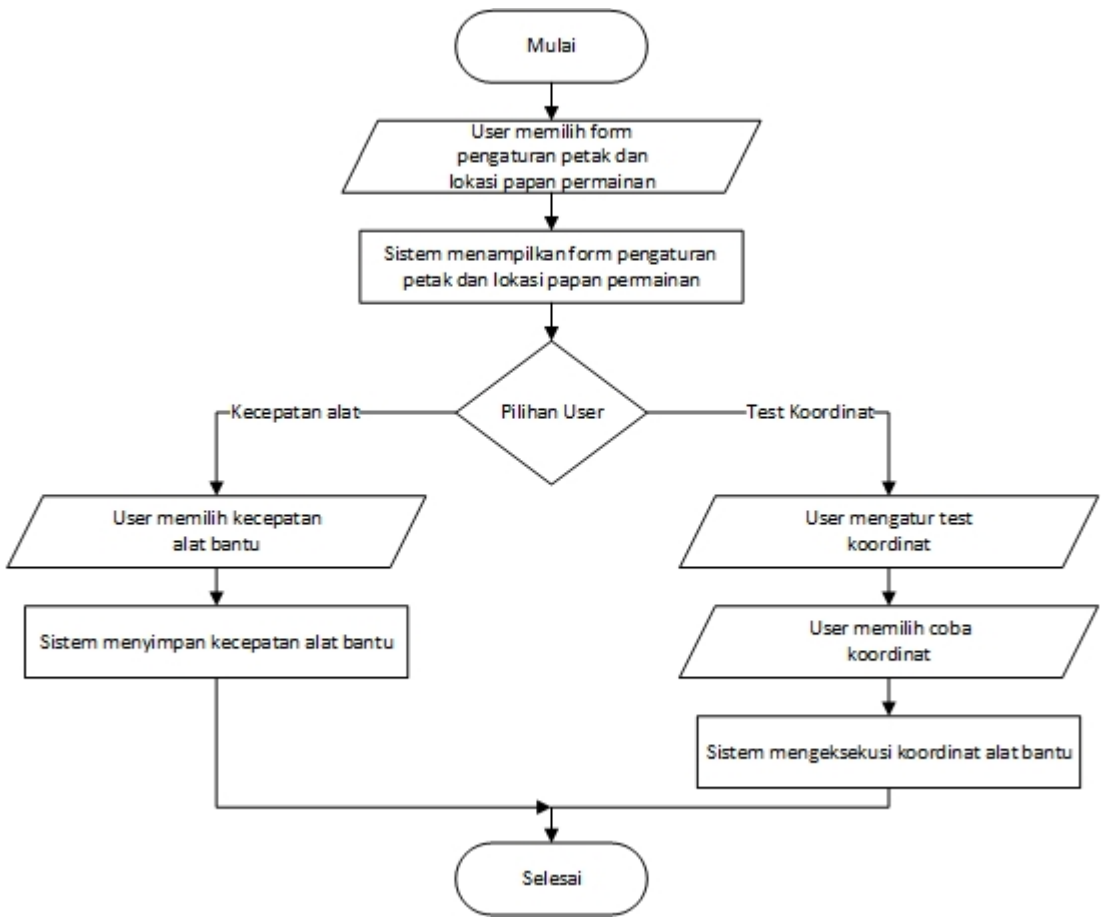
Gambar 3.50 *Flowchart* Kalibrasi Posisi Papan Permainan

3.5.4.6. *Flowchart* Mengatur Kamera dan Port *Arduino*



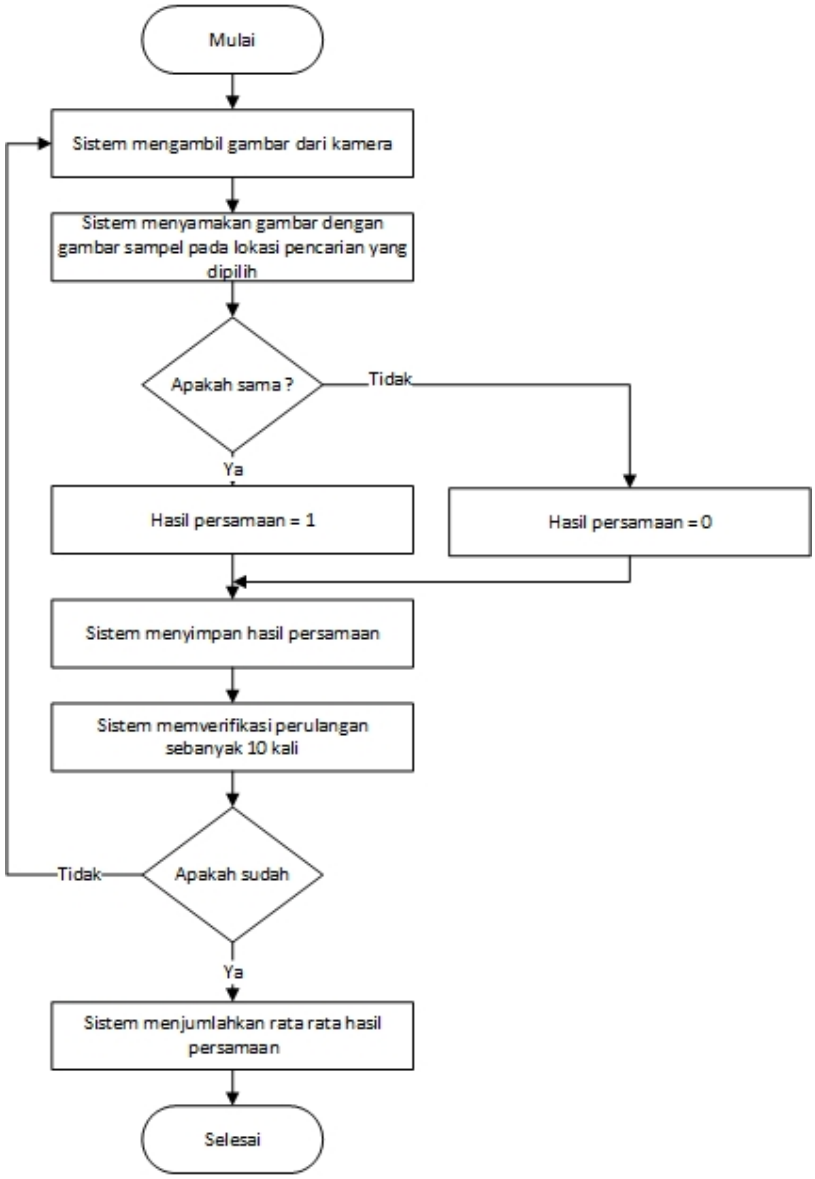
Gambar 3.51 *Flowchart* Mengatur Kamera dan Port *Arduino*

3.5.4.7. *Flowchart* Mengatur Kecepatan dan Test Koordinat



Gambar 3.52 *Flowchart* Mengatur Kecepatan dan Test Koordinat

3.5.4.8. *Flowchart* Deteksi Petak



Gambar 3.53 *Flowchart* Deteksi Petak