PENDETEKSI GERAKAN UNTUK PEMBIDIK TANK OTOMATIS BERBASIS FPGA

FPGA-BASED MOTION DETECTOR FOR AUTOMATIC TANK TARGETING

Teuku Rafifsyah Thomi, Iswahyudi Hidayat, Rizki Ardianto
Prodi S1 Teknik Elektro, Fakultas Teknik Elektro, Universitas Telkom
rafifboss@students.telkomuniversity.ac.id iswahyudihidayat@telkomuniversity.ac.id
rizkia@telkomuniversity.ac.id

Abstrak

Perkembangan teknologi yang terus terjadi merupakan salah satu faktor munculnya berbagai kebutuhan baru. mulai dari hal sederhana hingga sangat penting seperti pertahanan negara. Jika dilihat dari sudut pandang negara berkembang, teknologi sistem militer negara maju dapat menjadi ancaman keamanan. Namun, hal tersebut dapat dikurangi dengan memanfaat kemajuan teknologi. Salah satunya menggunakan metode pendeteksi gerak untuk pembidik otomatis pada sistem persenjataan.

Dalam perancangan sistem pembidik otomatis ini, objek yang menjadi target ialah Tank. Sistem ini berupa kamera yang diintegrasikan pada lengan robot yang menggunakan motor sebagai penggeraknya dan dikendalikan menggunakan FPGA. Dengan demikian, sistem ini dapat membidik pada trarget yang diinginkan.

Sistem pembidik otomatis yang diimplementasikan pada FPGA ini, memiliki tingkat akurasi, presisi, dan kecepatan sudut yang berbeda untuk tiap sumbunya. Motor sumbu X memiliki akurasi 90,9% presisi 84,52%, dan kecepatan sudut $71,316^0 \pm 8,158^0/s$. Sedangkan motor sumbu Y memiliki akurasi 92,34% presisi 84,15%, dan kecepatan sudut $70,206^0 \pm 8,352^0/s$.

Kata kunci : **pembidik otomatis, HDL, FPGA, Tank**

Abstract

The continously advancing technology is one of many factor that increase our needs, from trvial to significant thing such as national defence. From developing nation perspective, advanced nation military technology can become a threat. But, it can be reduced. In domain of weapon, it can be reduced by using motion detection method to aim target automatically.

In this automatic aiming system design, the object of aim is tank. The physical form of this system is a camera that attached on robot arm that use motor to move and controlled by FPGA. Thus, it can aim the desired target.

When implemented in FPGA, this automatic targeting system has different value of accuracy, precision and angle speed for each axis. X axis motor has 90,9% accuration, 84,52% precision, and $71,316^0 \pm 8,158^0/s$ angle speed. Whereas Y axis motor has has 92,34% accuration, 84,15% precision, and $70,206^0 \pm 8,352^0/s$ angle speed.

Keywords : **automatic Aim, VHDL, FPGA**

1. Pendahuluan

Dewasa ini, perkembangan teknologi sangat pesat dan mempengaruhi segala aspek dalam kehidupan, khususnya bagian pertahanan negara. Dengan adanya sistem senjata yang memiliki daya hancur besar seperti tank yang dilengkapi dengan sistem navigasi, komunikasi dan mobilitas yang tinggi, maka akan menjadi salah satu faktor ancaman yang perlu diwaspadai. Salah satu cara untuk mengatasi masalah ini adalah dengan membuat sebuah sistem deteksi yang bisa mendeteksi pergerakan tank. Salah satu sistem yang bisa digunakan untuk memperkuat persenjatan adalah dengan memanfaatkan *motion detection*. *Motion detection* adalah sebuah metode untuk mendeteksi sebuah pergerakan dengan memanfaatkan informasi gambar (*image processing*).

2. Pembidik Tank dengan Deteksi Gerakan

2.1. Video Digital

Video digital pada dasarnya tersusun atas serangkaian citra. Rangkaian citra tersebut ditampilkan pada layar dengan kecepatan tertentu, bergantung pada laju citra yang diberikan (dalam citra/detik). Jika laju citra cukup tinggi, mata manusia tidak dapat melihatnya sebagai gambar yang diam, melainkan sebagai gambar yang bergerak. Suatu citra direpresentasikan dengan sebuah matriks yang masing – masing elemennya

merepresentasikan nilai intensitas. Jika I adalah matiks dua dimensi I(x,y) adalah nilai intensitas yang sesuai pada baris x dan kolom y pada matriks tersebut, maka x dan y adalah posisi dari sebuah piksel.

Parameter umum suatu video digital ditentukan oleh resolusi, *pixel depth* (kedalaman piksel), dan *frame rate* (laju citra). Parameter tersebut digunakan untuk memprediksi kualitas video dan jumlah bit yang dibutuhkan untuk menyimpan atau mentransmisikannya.

2.2. Motor Servo

Motor servo adalah sebuah perangkat atau aktuator putar (motor) yang dirancang dengan sistem kontrol umpan balik loop tertutup (servo), sehingga dapat diatur untuk menentukan dan memastikan posisi sudut dari poros output motor. Motor servo merupakan perangkat yang terdiri dari motor DC, serangkaian *gear*, rangkaian kontrol dan potensiometer. Serangkaian *gear* yang melekat pada poros motor DC akan memperlambat putaran poros dan meningkatkan torsi motor servo, sedangkan potensiometer dengan perubahan resistansinya saat motor berputar berfungsi sebagai penentu batas posisi putaran poros motor servo.

Motor servo biasa digunakan dalam aplikasi-aplikasi di industri, selain itu juga digunakan dalam berbagai aplikasi lain seperti pada mobil mainan radio kontrol, robot, pesawat, dan lain sebagainya.

2.3. UART

Pada tugas akhir ini, blok komunikasi UART dirancang sebagai komunikasi utama pada sistem pembidik. Protokol komunikasi dengan Raspberry Pi yang digunakan ialah serial UART dengan 1 *start* bit (*low pulse*), 8 data bit dengan 0 *parity* dan 1 *stop* bit (*high pulse*). Berikut blok UART yang dirancang :

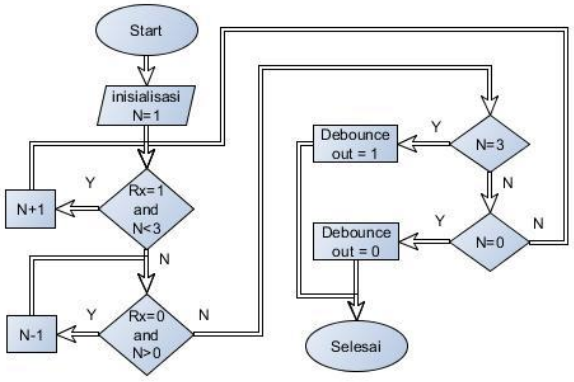
Gambar 1. Blok UART

Tabel 1. Tabel Keterangan Port UART

Pin *Receiver*	Komunikasi	Tipe	Keterangan
Rx	Raspberry Pi	in	Menerima data dari Raspberry Pi
data_stream_out	Dekoder	out	Tempat pengiriman data yang dikumpulkan dari port "rx" ke dekoder
data_stream_out_stb	Dekoder	out	Memberikan *flag* pengiriman port "data_stream_out" untuk dekoder
data_stream_out_ack	Dekoder	in	Port *feedback* yang menginformasikan bahwa data sudah diterima oleh dekoder

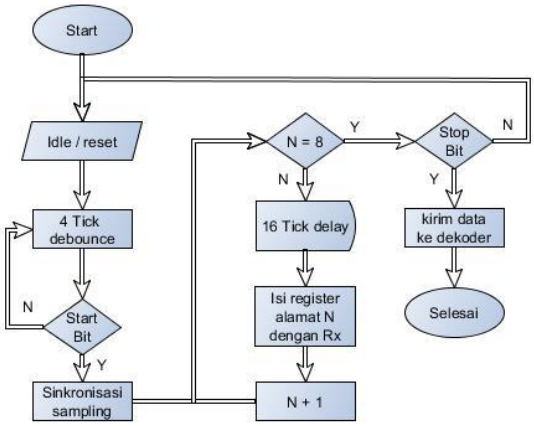
Pin *receiver* ialah pin-pin yang digunakan *receiver* UART untuk menerima data dari Raspberry Pi. Sedangkan pin transmitter digunakan untuk mempermudah proses analisis (membandingkan data yang dikirim Raspberry Pi dengan yang diterima UART). Pin transmitter berfungsi untuk melanjutkan data pada UART ke aplikasi Hyper Terminal pada PC.

Pada bagian *receiver* pada UART, digunakan metode 16 *times oversampling* (16 kali sampel tiap 1 bit) untuk meningkatkan kestabilan data. Metode ini mendukung implementasi 4 *Tick Debounce* sebagai filter dari masukan *spike*. Masukan yang dianggap sebagai *spike* adalah sinyal yang memiliki durasi pulsa 1 atau 0 sepanjang 1/16 dari durasi normal bit (1 sampel).



Gambar 2 Diagram alir filter *spike*

Untuk setiap sampel, sinyal pada port “Rx” akan dibaca untuk menentukan proses aritmatik yang terjadi pada *counter* (N). Nilai *counter* akan ditambah 1 jika masukan bernilai 1 dan N < 3. Sedangkan pengurangan nilai *counter* terjadi jika masukan bernilai 0 dan N > 0. Keluaran *debounce* ditentukan oleh besarnya nilai *counter* selama proses aritmatik, keluaran bernilai 1 jika N = 3, sedangkan bernilai 0 jika N = 0.



Gambar 3 Diagram alir *Receiver* UART

Pada bagian *receiver*, kondisi *idle* dipertahankan selama instruksi *reset* diberikan. Dengan menggunakan 4 *Tick debounce* yang sudah dijelaskan pada Gambar 2, maka *start* bit (Rx=0) akan dideteksi pada sampel yang ke-2. Pendeteksian *start* bit akan selalu memicu dimulainya proses sinkronisasi sampling. Proses ini memungkinkan sistem untuk mengambil data pada sampel yang ke-8. Meskipun data sudah aman dari *spike*, proses ini perlu dilakukan untuk menjamin pulsa data memiliki periode yang *valid*, yaitu setengah dari lebar pulsa normal (16 sampel). Proses berikutnya adalah pengambilan data, proses ini dilakukan sebanyak 8 kali, hal ini sesuai dengan protokol komunikasi yang direncanakan. Setiap data *valid* akan disimpan ke register dengan urutan dari LSB ke MSB. Setelah pengambilan data selesai, proses selanjutnya adalah menunggu *stop* bit (RX=1). Saat *stop* bit terdeteksi, sistem akan memberikan sinyal bahwa data siap dikirim ke dekoder (data_stream_out_stb=1). Jika pada pengambilan data terdeteksi pulsa yang tidak stabil (selalu terjadi *spike*, durasi kurang dari 8 sampel, tidak memiliki *stop* bit), maka komunikasi dianggap gagal dan sistem akan kembali pada kondisi *idle*.

.

2.4. Dekoder

Data yang dikeluarkan oleh blok UART ialah karakter dengan format ASCII. Sehingga perlu dirancang blok dekoder untuk mendapatkan nilai koordinat yang diinginkan dari karakter-karakter ini. Berikut blok dekoder yang dirancang :

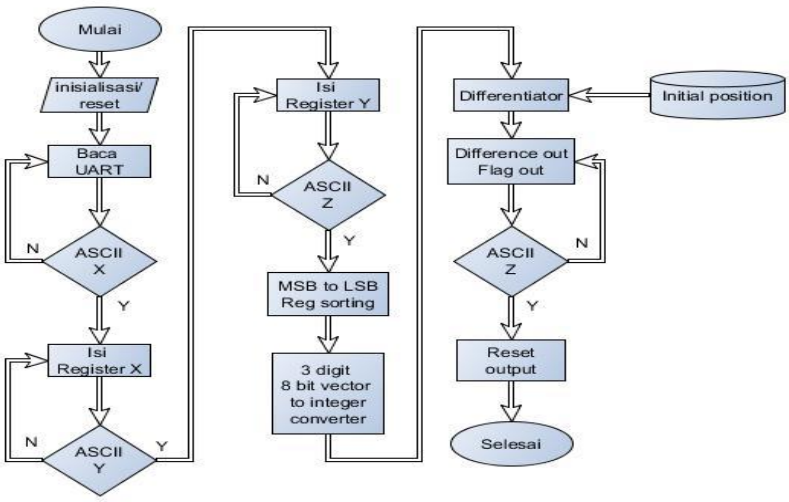
Gambar 4. Blok Dekoder

Tabel 2. Tabel Keterangan Pin Dekoder

Pin	Komunikasi	Tipe	Keterangan
from UART	UART	in	Menerima data dari blok UART
stb	UART	in	Menerima *flag* bahwa data akan dikirim dari blok UART
ack	UART	out	Memberikan *flag* ke blok UART bahwa data sudah diterima oleh dekoder
arah	Motor Controller	out	Memberikan *flag* arah putar ke motor controller
durasi	Motor Controller	out	Memberikan data durasi *high pulse* ke motor controller

Dekoder yang dirancang memiliki spesifikasi berikut :

1. Data masukan memiliki format ASCII
2. Maksimal data untuk 1 kali proses *decoding* ialah 10 karakter ASCII dengan urutan X###Y###ZZ (# ialah karakter angka dalam ASCII) yang disimpan di dalam 2 buah Register (register x dan register y)
3. Data keluaran ialah 8 bit selisih koordinat dan 1 bit *flag* selisih.



Gambar 5. Diagram Alir Dekoder

Pada awal proses, dekoder menunggu karakter "X" sebagai *trigger* untuk pengisian register x. Dekoder juga akan mengisi register y jika terdeteksi karakter "Y", sehingga register x dan y tidak harus selalu diisi sampai penuh (karakter "X"/"Y" + 3 digit angka koordinat). Pengisian kedua register ini menggunakan metode SISO dari kiri ke kanan, sehingga isi data pada register akan memiliki urutan LSB di kiri dan MSB di kanan("###X" dan "###Y"). Langkah berikutnya adalah menunggu karakter "Z" sebagai tanda bahwa data koordinat sudah selesai dikirim, dan boleh dilakukan proses *decoding.* Untuk mempermudah proses *decoding*, maka proses selanjutnya adalah membalik isi register "X###" dan "Y###". Untuk register yang belum penuh (contoh X5), akan diselipkan angka nol (0) di sebelah kanan karakter *trigger* menjadi (X005). Langkah selanjutnya adalah mengkonversi karakter angka pada register menjadi angka dalam format integer. Proses konversi ini dapat ditulis dalam formula berikut.

$$Int_{bit} = conv_{integer}(reg(bit))\ x\ 10^{bit}$$

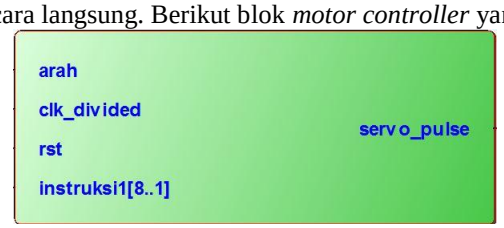
Setelah konversi dilakukan, setiap integer untuk semua bit dijumlahkan menggunakan formula berikut.

$$Int = \sum_{bit=0}^{MSB} Int_{bit}$$

Langkah selanjutnya adalah mencari koordinat yang diinginkan dengan mencari selisih antara nilai yang diberikan UART dengan nilai awal (*initial position*). Proses ini akan menghasilkan koordinat yang diinginkan beserta *flag* arah untuk blok *motor controller*. Langkah terakhir adalah menunggu karakter "Z" untuk mengirim koordinat dan *flag* ke blok *motor controller*.

2.5. Motor Controller

Data koordinat dan flag yang dikirim dari blok dekoder, akan diproses oleh blok *motor controller* yang akan mengendalikan motor servo secara langsung. Berikut blok *motor controller* yang dirancang:

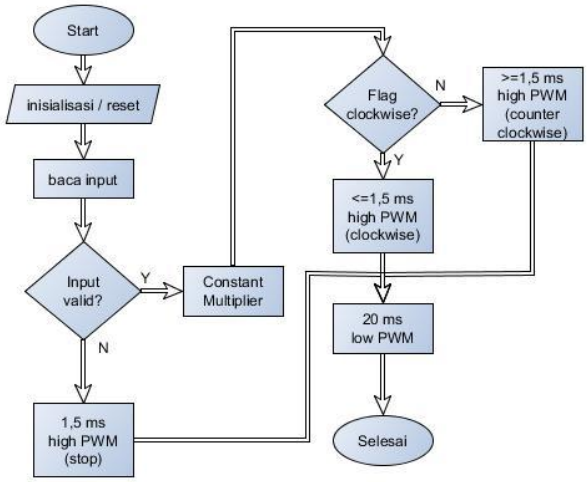


Gambar 6. Blok *Motor Controller*

Tabel 3. Tabel Keterangan Pin *Motor Controller*

Pin	Komunikasi	Tipe	Keterangan
arah	Dekoder	in	Menerima data arah putar dari Dekoder
instruksi	Dekoder	in	Menerima data durasi *high pulse* dari Dekoder
servo_pulse	Motor servo	out	Sinyal PWM ke motor servo

Blok *motor controller* mengendalikan servo dengan sinyal PWM melalui port GPIO FPGA. Lebar dari *high pulse* (3,3/5 V) digunakan untuk menentukan kecepatan dan arah putar motor servo, sedangkan lebar *low pulse* (0 V) digunakan sebagai delay antar *high pulse* untuk mendapatkan putaran yang stabil. *Low pulse* selalu memiliki durasi 20ms, hal ini disebabkan oleh spesifikasi kestabilan putar dari motor servo yang digunakan.



Gambar 7. Diagram alir *motor controller*

Masukan yang diterima oleh *motor controller* ialah 8 bit selisih koordinat untuk kecepatan dan 1 bit *flag* untuk arah gerak motor servo. Seluruh bentuk masukan akan dianggap *invalid* jika memiliki nilai yang berubah selama motor masih mengerjakan instruksi sebelumnya. Masukan kecepatan yang *valid* akan dikali oleh sebuah konstanta untuk memperbesar torsi yang diberikan, meskipun kecepatan juga bertambah pada proses ini. Hal ini disebabkan oleh sifat motor servo yang memiliki perbandingan linier antara kecepatan dan torsi, sehingga jika torsi terlalu kecil, motor servo tidak berputar. Langkah selanjutnya adalah membaca nilai *flag* untuk menentukan arah putar motor servo. Penentuan arah putar ini dapat ditulis dengan persamaan berikut:

$$t_H = 1500 \pm d$$

$$1300 \text{ us} \leq t_H \leq 1700 \text{ us}$$

Durasi pulsa *high* (t_H) ditentukan oleh durasi pulsa "diam" (1500)us , *flag* ($\pm$), dan masukan kecepatan (d)us. Motor servo berputar searah jarum jam (*clockwise*) jika *flag* bernilai 1(+), dan sebaliknya jika bernilai 0(-). Jika nilai (t_H) keluar dari batas (1300us atau 1700us), maka akan dibulatkan ke batas terdekat.

3. Implementasi FPGA

3.1. Pengujian Komunikasi UART

Pengujian dilakukan dengan memberikan beberapa karakter ASCII dari pin TX UART Raspberry Pi ke pin Rx UART FPGA lalu mengeluarkan karakter yang sama ke PC melalui pin Tx FPGA. Pada PC, digunakan aplikasi Hyper Terminal untuk membaca karakter ASCII yang diterima. Nilai *baud-rate* ditingkatkan sampai mendapatkan *error* yang terkecil.

Tabel berikut berisi rangkuman pengujian blok UART.

Tabel 4. Pengujian UART pada FPGA

Karakter yang dikirim ke UART	**Karakter yang diterima oleh UART**				
	***Baud-rate* (bps)**				
	9600	**19200**	**38400**	**57600**	**115200**
X320Y240ZZ	X320Y2™0ZZ	X320Y240ZZ	X320Y240ZZ	X320Y240ZZ	X320Y240ZZ
ABCDEFGHIJ	ABCDEFGHI™	ABCDEFGHœJ	ABCDEFGHIJ	ABCDEFGHIJ	ABCDEFGHIJ
1234567890	1234567˜89	123456790˜	1234567890	1234567890	1234567890
1A2B3C4D5E	1A2BœC4D5E	1A2B3C4D5˜	1A2B3C4D5E	1A2B3C4D5E	1A2B3C4D5E
***Error* (%)**	**10**	**7,5**	**0**	**0**	**0**

Karakter yang dikirim selalu berjumlah 10 buah, karena format penerimaan koordinat yang digunakan ialah "X###Y###ZZ". Karena *error* berbanding terbalik dengan *baud-rate*, maka akan diambil nilai *baud-rate* terkecil. Meskipun demikian, munculnya *error* pada *baud-rate* dibawah 38400 bps disebabkan oleh terlambatnya pengambilan data pada FIFO UART. Sehingga dapat disimpulkan bahwa *baud-rate* optimal untuk blok UART ialah 38400 bps, karena memiliki nilai paling rendah yang tidak menyebabkan *error* pada FIFO.

3.2. Pengujian Durasi Putar Motor

Pengujian dilakukan dengan cara memberikan koordinat target melalui Raspberry Pi, lalu membandingkan hasil pergerakan motor terhadap koordinat yang diberikan. Kecepatan putar motor berbanding lurus terhadap selisih koordinat awal dengan koordinat target. Sedangkan durasi putar motor merupakan variabel konstan yang dapat diubah sebelum pengujian dimulai.

Tabel 5. Pengujian Motor untuk sumbu Y

Durasi putar motor servo (ms)	Jarak target (cm)	Koordinat asal (Y)	Koordinat target (Y)	*Error margin* (+/-piksel)
70	10	240	160	4
70	10	240	320	4
60	10	240	160	2
60	10	240	320	2
50	10	240	160	5
50	10	240	320	5

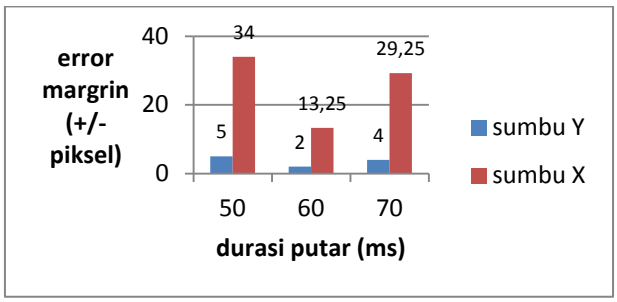
Tabel 6. Pengujian Motor untuk sumbu X

Durasi putar motor servo (ms)	Jarak target (cm)	Koordinat asal (X)	Koordinat target (X)	*Error margin* (+/-piksel)
70	10	320	240	45
70	10	240	160	30
70	10	320	400	22
70	10	400	480	20
60	10	320	240	0
60	10	240	160	30
60	10	320	400	10
60	10	400	480	13
50	10	320	240	47
50	10	240	160	50
50	10	320	400	17
50	10	400	480	22

Dari tabel 5 dan 6, nilai *error margin* yang didapat selalu berubah, meskipun selisih antar koordinat asal dan koordinat target selalu sama. Sehingga diperlukan nilai rata-rata untuk mempermudah pemilihan durasi putar motor servo. Rata - rata *error margin* dapat dicari dengan rumus:

$$\overline{X_{\text{durasi putar}}} = \frac{\sum_{i=1}^{n} \text{error margin}}{n}$$

Sehingga dapat digambar grafik berikut :

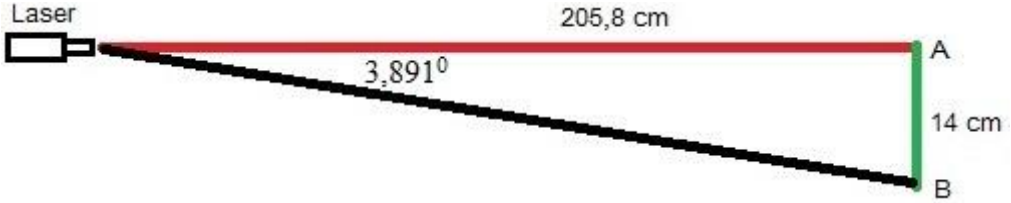


Gambar 8. Grafik rata – rata *error margin* motor servo

Dari gambar 8, dapat disimpulkan bahwa durasi putar motor servo untuk setiap instruksi yang memiliki *error margin* terkecil ialah 60 ms.

3.3. Pengujian Akurasi, Presisi dan Kecepatan Motor

Pengujian dilakukan dengan cara memberikan koordinat target (14 cm dari posisi awal) secara manual target dari Raspberry Pi. Selisih dihitung dengan menggunakan mengukur posisi laser terhadap koordinat yang diberikan. Sedangkan jarak target dari laser (205,8 cm) merupakan variabel konstan yang ditetapkan sebelum pengujian dimulai.



Gambar 9. Ilustrasi pengujian; A=Posisi awal, B=Posisi target

Dengan mendapatkan selisih posisi target dengan posisi laser, akurasi, presisi dan kecepatan putar motor dapat dihitung dengan formula berikut.

$$\text{Akurasi} = \frac{\text{Skala Akurasi}}{\text{Skala Akurasi} + \overline{X_{\text{selısıh}}}} \text{ X } 100\%$$

$$\text{Presisi} = 1 - \frac{\sum|\text{Selisih} - \overline{X_{\text{selısıh}}}|}{n} \text{ X } 100\%$$

$$\text{Kecepatan sudut} = \frac{\overline{X_{\text{Laser}}} \pm \frac{\sum|\text{Selisih} - \overline{X_{\text{selısıh}}}|}{\sqrt{n}}}{\text{Durasi Putar}}$$

Pada percobaan ini, skala akurasi yang digunakan adalah $3{,}891^0$ (akurasi akan bernilai 50% jika total selisih sebesar $3{,}891^0$).

Untuk mempermudah pengujian, jarak target (205,8cm), posisi target ($3{,}891^0$), dan durasi putar motor servo (60ms) selalu sama. Sehingga didapat data pengujian sebagai berikut :

Tabel 7. Selisih jarak laser terhadap sumbu X

Jarak target (cm)	**Posisi target (derajat)**	**Posisi laser (derajat)**	**Selisih (derajat)**	$\|\textbf{Selisih} - \overline{\textbf{X}_{\textbf{selısıh}}}\|$
205,8	3,891	4,141	0,25	0,138
205,8	3,891	4,224	0,333	0,055
205,8	3,891	4,196	0,305	0,083
205,8	3,891	5,053	1,162	0,774
205,8	3,891	4,113	0,222	0,166
205,8	3,891	4,168	0,277	0,111
205,8	3,891	4,196	0,305	0,083
205,8	3,891	4,251	0,36	0,028
205,8	3,891	4,224	0,333	0,055
205,8	3,891	4,224	0,333	0,055
		$\overline{\textbf{X}_{\textbf{Laser}}}$	$\overline{\textbf{X}_{\textbf{selısıh}}}$	$\sum\|\textbf{Selisih} - \overline{\textbf{X}_{\textbf{selısıh}}}\|$
		4,279	**0,388**	**1,548**

Tabel 8. Selisih jarak laser terhadap sumbu Y

Jarak target (cm)	**Posisi target (derajat)**	**Posisi laser (derajat)**	**Selisih (derajat)**	$\|\textbf{Selisih} - \overline{\textbf{X}_{\textbf{selısıh}}}\|$
205,8	3,891	4,057	0,166	0,1554
205,8	3,891	4,611	0,72	0,3986
205,8	3,891	4,085	0,194	0,1274
205,8	3,891	4,224	0,333	0,0116
205,8	3,891	4,39	0,499	0,1776
205,8	3,891	4,085	0,194	0,1274
205,8	3,891	4,002	0,111	0,2104
205,8	3,891	4,085	0,194	0,1274
205,8	3,891	4,168	0,277	0,0444
205,8	3,891	4,417	0,526	0,2046
		$\overline{\textbf{X}_{\textbf{Laser}}}$	$\overline{\textbf{X}_{\textbf{selısıh}}}$	$\sum\|\textbf{Selisih} - \overline{\textbf{X}_{\textbf{selısıh}}}\|$
		4,2124	**0,3214**	**1,5848**

Dari tabel 7 dan 8, nilai akurasi, presisi dan kecepatan sudut untuk motor servo tiap sumbu dapat dihitung.

$$\text{Akurasi X} = \frac{3{,}891}{3{,}891 + 0{,}388} \text{ X } 100\% = 90{,}9\%$$

$$\text{Presisi X} = 1 - \frac{1{,}548}{10} \text{ X } 100\% = 84{,}52\%$$

$$\text{Kecepatan sudut X} = \frac{4{,}279 \pm \frac{1{,}548}{\sqrt{10}}}{60\text{ms}} = 71{,}316^0 \pm 8{,}158^0/s$$

$$\text{Akurasi sumbu Y} = \frac{3{,}891}{3{,}891 + 0{,}3214}\ \text{X}\ 100\% = 92{,}34\%$$

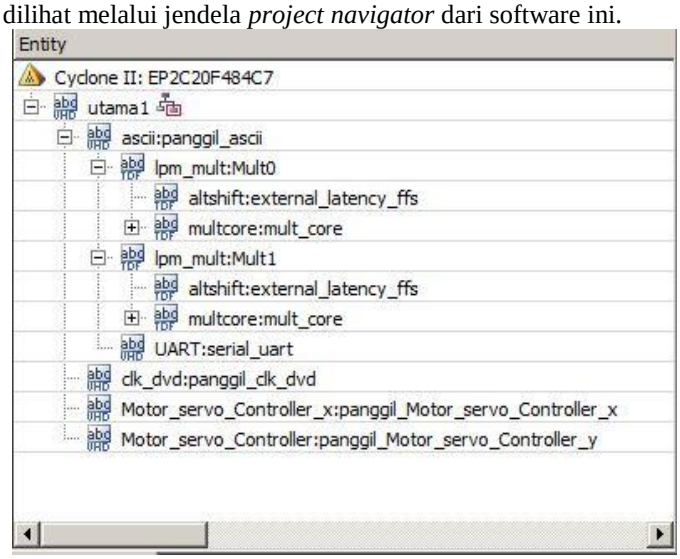
$$\text{Presisi sumbu Y} = 1 - \frac{1{,}5848}{10}\ \text{X}\ 100\% = 84{,}15\%$$

$$\text{Kecepatan sudut Y} = \frac{4{,}2124 \pm \frac{1{,}5848}{\sqrt{10}}}{60\text{ms}} = 70{,}206^0 \pm 8{,}352^0/s$$

Sehingga dapat diketahui bahwa motor sumbu X memiliki akurasi 90,9% presisi 84,52%, kecepatan sudut $71{,}316^0 \pm 8{,}158^0/s$. Sedangkan motor sumbu Y memiliki akurasi 92,34% presisi 84,15%, kecepatan sudut $70{,}206^0 \pm 8{,}352^0/s$.

3.4. Implementasi sistem pembidik pada FPGA

Proses implementasi sistem pembidik pada FPGA menggunakan *software* Quartus II 12.1. Sehingga hirarki dari sistem dapat dilihat melalui jendela *project navigator* dari software ini.



Gambar 10. Hirarki Sistem Pembidik Pada Quartus II 12.1

Dapat dilihat pada gambar 10, bahwa desain sistem terdiri dari sebuah *top* modul dan beberapa *sub* modul.

Blok ASCII merupakan antarmuka utama sistem untuk menerima dan melakukan proses *decoding* koordinat yang diberikan. Blok ini juga mencakup blok UART.

Blok selanjutnya adalah *motor controller x* dan *motor controller y*. Pada dasarnya, kedua blok ini memliki fungsi yang sama, tapi memiliki variabel penentu kecepatan putar yang berbeda, hal ini dibutuhkan karena adanya perbedaan beban pada setiap motor. Kedua blok ini menggunakan *clock* yang diberikan oleh blok *clock divider* untuk mengatur PWM. Optimalisasi *resource* dilakukan pada blok ini dengan cara mengubah algoritma pemrosesan PWM. Hasil optimalisasi *resource* blok *motor controller* dapat dilihat pada tabel berikut.

Tabel 9. *Resource motor controller*

Optimization	Before	After
Total logic elements	345 / 18,752 (2 %)	71 / 18,752 (< 1 %)
Combinational with no register	322	33
register only	0	0
Combinational with a register	23	38
Dedicated logic registers	23 / 18,752 (< 1 %)	38 / 18,752 (< 1 %)

Sistem pembidik yang diimplementasikan pada FPGA memiliki data *summary* yang dapat dilihat pada tabel 10.

Tabel 10. Data *summary* Implementasi FPGA dari sistem pembidik

Total logic elements	717 / 18,752 (4 %)
Combinational with no register	348
register only	57
Combinational with a register	312
Logic element usage by number of LUT inputs	
4 input functions	152
3 input functions	254
2 input functions	254
register only	57
Logic elements by mode	
normal mode	393
arithmetic mode	267
Total registers	369 / 19,649 (2 %)
Dedicated logic registers	369 / 18,752 (2 %)
I/O registers	0 / 897 (0 %)
I/O pins	8 / 315 (3 %)
Clock pins	2 / 8 (25 %)
Embedded Multiplier 9-bit elements	0
Maximum fan-out	234
Total fan-out	3127
Average fan-out	3.02

Berdasarkan tabel 10, dapat disimpulkan bahwa sistem pembidik yang diimplementasikan pada FPGA menggunakan rangkaian kombinasional dan register sebagai *resource* utama.

4. Kesimpulan

Berdasarkan hasil perancangan, implementasi dan pengujian keseluruhan pada tugas akhir ini, sistem memiliki *error margin* terbesar 30 piksel dan *error margin* terkecil 0 piksel untuk durasi putar 60 ms. Komunikasi UART yang diimplementasikan memiliki nilai *baud-rate* 38400 bps. Sistem juga memiliki tingkat akurasi, presisi, dan kecepatan sudut yang berbeda untuk tiap sumbunya. Motor sumbu X memiliki akurasi 90,9% presisi 84,52%, kecepatan sudut $71{,}316^0 \pm 8{,}158^0/s$. Sedangkan motor sumbu Y memiliki akurasi 92,34% presisi 84,15%, kecepatan sudut $70{,}206^0 \pm 8{,}352^0/s$.

Daftar Pustaka:

[1] Yuan Yuan, Sabu Emmanuel, Yuming Fang, and Weisi Lin.Visual Object Tracking Based on Backward Model Validation. *IEEE Trans*. circuits and systems for video technology. vol. 24, no. 11, pp. 1898-1910, Nov 2014.

[2] Yuan Yao, and Yun Fu. Contour Model-Based Hand-Gesture RecognitionUsing the Kinect Sensor. *IEEE Trans*. circuits and systems for video technology. vol. 24, no. 11, pp. 1935-1944, Nov 2014.

[3] Annan Li, and Shuicheng Yan. Object Tracking With Only Background Cues. *IEEE Trans*. circuits and systems for video technology, vol. 24, no. 11, pp. 1911-1919, Nov 2014.

[4] Shih-Lun Chen, and En-Di Ma. VLSI Implementation of an AdaptiveEdge-Enhanced Color Interpolation Processor for Real-TimeVideo Applications. *IEEE Trans*. circuits and systems for video technology, vol. 24, no. 11, pp. 1982-1991, Nov 2014.

[5] Frau, J. and V. Llario. 1991. Predictive Tracking of Targets Using Image Sequences. Proceeding of IEEE/RSJ International Workshop on Intelligent Robots and Systems (IROS '91). 848-854.

[6] andyq3lectra.wordpress.com/2009/11/30/image-processing-menggunakan-delphi-1a [diakses 21 Agustus 2015]

[7] www.electronickits.com/robot/BioloidAX-12(english).pdf [diakses 21 Agustus 2015]

[8] https://www.parallax.com/product/900-00008 [diakses 21 Agustus 2015]

[9] https://github.com/sd2k9/ [diakses 21 Agustus 2015]