ALAT UJI MUATAN ROKET KOMURINDO BEBASIS FPGA (*FIELD PROGAMMABLE GATE ARRAY*) BAGIAN PENGUJIAN FUNGSIONAL *G-FORCE*

SKRIPSI

Diajukan Untuk Memenuhi Persyaratan

Memperoleh Gelar Sarjana Teknik

DISUSUN OLEH:

DENNY SATRIO N.

NIM. 0810630042-63

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS BRAWIJAYA

MALANG

2013

LEMBAR PERSETUJUAN

ALAT UJI MUATAN ROKET KOMURINDO BEBASIS FPGA (*FIELD PROGAMMABLE GATE ARRAY*) BAGIAN PENGUJIAN FUNGSIONAL *G-FORCE*

SKRIPSI

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
DENNY SATRIO N.
NIM. 0810630042-63

Telah diperiksa dan disetujui oleh :

Pembimbing I	Pembimbing II
M. Julius St., Ir., MS.	**Mochammad Rif'an, ST., MT.**
NIP. 19540720 198203 1 002	**NIP. 19710301 200012 1 001**

LEMBAR PENGESAHAN

ALAT UJI MUATAN ROKET KOMURINDO BEBASIS FPGA (*FIELD PROGAMMABLE GATE ARRAY*) BAGIAN PENGUJIAN FUNGSIONAL *G-FORCE*

SKRIPSI
JURUSAN TEKNIK ELEKTRO

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:

DENNY SATRIO N.

NIM. 0810630042-63

Skripsi ini telah diuji dan dinyatakan lulus pada
tanggal 11 April 2013

DOSEN PENGUJI

Ir. Nurussa'adah, MT.	**Ponco Siwindarto, Ir., M.Eng.Sc.**
NIP. 19680706 199203 2 001	**NIP. 19590304 198903 1 001**

Rahmadwati, ST., MT.

NIP 19771102 200604 2 003

Mengetahui,
Ketua Jurusan Teknik Elektro

Dr. Sholeh Hadi Pramono, Ir., MS.

NIP. 19580728 198701 1 001

PENGANTAR

Puji syukur kehadirat Allah SWT atas limpahan rahmat dan karuniaNya, penulis dapat menyelesaikan skripsi yang berjudul "Alat Uji Muatan Roket KOMURINDO Berbasis FPGA (*Field Programmable Gate Array*) Bagian Pengujian Fungsional *G-Force*". Skripsi ini disusun sebagai persyaratan untuk memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Universitas Brawijaya.

Penulis menyadari bahwa penyusunan skripsi ini tidak terlepas dari bantuan bebagai pihak. Oleh karena itu, dengan ketulusan dan kerendahan hati penulis menyampaikan terima kasih kepada :

- Mama, Papa, atas segala nasehat, kasih sayang, perhatian dan kesabarannya di dalam membesarkan dan mendidik penulis, serta telah banyak mendoakan kelancaran penulis hingga terselesaikannya skripsi ini,
- Kakak dan adikku yang banyak mendoakan kelancaran penulis hingga terselesaikannya skripsi ini,
- Bapak Dr. Sholeh Hadi Pramono, Ir., MS selaku Ketua Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak M. Aziz Muslim, ST., MT, Ph.D selaku Sekretaris Jurusan Teknik Elektro Universitas Brawijaya,
- Bapak Mochammad Rif'an, ST., MT sebagai Ketua Program Studi Jurusan Teknik Elektro Universitas Brawijaya serta sebagai Dosen Pembimbing II atas segala bimbingan, pengarahan, gagasan, ide, saran serta motivasi yang telah diberikan,
- Bapak M. Julius St, Ir., MS sebagai Ketua Kelompok Dosen Keahlian Elektronika Jurusan Teknik Elektro Universitas Brawijaya serta sebagai Dosen Pembimbing I atas segala bimbingan, pengarahan, gagasan, ide, saran serta motivasi yang telah diberikan,

- Staf Rekording, staf Pengajaran, staf Pengajar, dan staf Ruang Baca Jurusan Teknik Elektro yang telah membantu segala urusan penulis selama ini,
- Tim KOMURINDO dan alumni tim.
- Sahabat seperjuangan Mas Hendra, Aris, Fikri, Cholik, dan Nanang.
- Teman-teman Laboratorium SISDIG.
- Teman-teman Concordes angkatan 2008 yang telah berbagi ilmu dengan penulis dan selalu memberikan semangat,
- Seluruh teman-teman, senior serta semua pihak yang tidak mungkin untuk dicantumkan namanya satu-persatu, terima kasih banyak atas bantuan dan dukungannya.

Penulis menyadari bahwa skripsi ini masih belum sempurna. Oleh karena itu, penulis mengharapkan kritik dan saran untuk penyempurnaan tulisan di masa yang akan datang. Penulis berharap, semoga skripsi ini bermanfaat bagi kita semua.

Malang, April 2013

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Denny Satrio N., Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya, Februari 2013, *Alat Uji Muatan Roket KOMURINDO Berbasis FPGA (Field Programmable Gate Array) Bagian Pengujian Fungsional G-Force*, Dosen Pembimbing : M. Julius St, Ir., MS dan Mochammad Rif'an, ST., MT.

Kompetisi Muatan Roket Indonesia (KOMURINDO) merupakan suatu ajang kompetisi perancangan dan pembuatan muatan/*payload* roket. Tujuan dari kontes ini adalah untuk menumbuh kembangkan kreatifitas dan minat para mahasiswa dalam pengembangan teknologi roket. Dalam kontes tersebut terdapat beberapa tahap seleksi pengujian, diantaranya adalah pengujian *g-force*. Pengujian ini dilakukan dengan memberikan gaya g sebesar 6g terhadap muatan roket sesuai dengan *rule* KOMURINDO.

Sistem konrol utama alat uji *g-force* ini adalah menggunakan FPGA (*Field Progammable Gate Array*). Sinyal yang diproses FPGA adalah sinyal PWM yang diberikan ke driver motor untuk mengatur kecepatan putar alat agar alat memiliki gaya sentrifugal. Untuk Mengetahui besarnya kecepatan putaran digunakan sensor rotary encoder E40S6. Agar kecepatan putaran dapat setabil maka dibutuhkan *feedback controller*. Hasil RPM yang terbaca oleh sensor kemudian dikonversi menjadi gaya *g* dengan menggunakan RCF (*Relative Centrifuge Force).*

Hasil pengujian menunjukkan nilai RPM yang diproses oleh FPGA dengan alat ukur tachometer kesalahan rata-rata adalah 1,13% dengan nilai RPM yang berbeda-beda. Dan hasil output maksimal yang dapat dihasilkan alat adalah sebesar 231 RPM atau 9g.

Kata Kunci: KOMURINDO, FPGA, PWM, Sensor Rotary Encoder E40S6, *feedback controller,* RCF.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi roket yang semakin pesat dan didukung oleh kemajuan ilmu komputer dan sistem kontrol sangat membantu manusia dalam mendesain dan mengembangkan sebuah roket. Roket yang handal adalah roket yang mampu melakukan tugasnya dengan baik dan sesuai dengan yang dirancang oleh pembuat roket. Teknologi roket tersebut dapat diaplikasikan dalam berbagai bidang, diantaranya adalah pengambilan citra di udara, pengiriman satelit keluar angkasa, pertahanan militer dan misi-misi yang sangat penting dapat dilakukan oleh teknologi roket.

Sebagai bentuk turut mengembangkan teknologi roket, Lembaga Penerbangan dan Antariksa Nasional (LAPAN) mengadakan kontes dalam bidang roket yang diadakan setiap tahun sekali yang diikuti oleh mahasiswa-mahasiswa yang mewakili masing-masing universitas yang ada di Indonesia. Kontes tersebut adalah Kompetisi Muatan Roket Inonesia (KOMURINDO).

Dalam Kompetisi Muatan Roket Indonesia (KOMURINDO), terdapat 2 bagian utama dari penyusun roket, yaitu peluncur roket dan muatan roket. Peluncur roket berfungsi untuk memberikan daya dorong roket agar dapat meluncur ke udara, peluncur roket disiapkan oleh panitia KOMURINDO. Muatan roket berfungsi untuk mengirim data dari hasil pemantauan udara yang dibuat oleh para peserta KOMURINDO dari berbagai universitas di Indonesia.

Terdapat 2 pengujian dalam KOMURINDO, yaitu pengujian darat dan udara. Pengujian darat meliputi pengujian fungsional vibrasi, pengujian fungsional *g-force* dan pengujian fungsional *g-shock*.

Pengujian *g*-force adalah pengujian yang menggantikan *payload* roket pada saat *take off* sampai dengan separasi (pemisahan muatan roket dengan peluncur roket). Pembuatan alat uji *g-force* dapat mengatasi masalah dalam hal pengujian gaya gravitasi yang diterima roket, sehingga tidak perlu melakukan peluncuran roket berulang-ulang. Berdasarkan fungsi alat uji fungsional *g-force*

tersebut, maka disusunlah skripsi, yang akan membahas secara rinci mengenai segala komponen yang dibutuhkan dalam perancangan alat uji tersebut. Tetapi dalam penelitian ini, hanya dibahas mengenai perancangan alat uji fungsional *g-force* muatan roket. Tidak termasuk perancangan muatan roket.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan di atas, maka pembahasan skripsi ini ditekankan pada :

1) Bagaimana merancang dan memprogram rangkaian pada FPGA untuk menghasilkan output data *G-force* dengan nilai antara 1g sampai dengan 9g.
2) Bagaimana merancang suatu alat lengan putar yang dapat menghasilkan nilai RPM yang dibutuhkan untuk mendapatkan *g-force* maksimal sebesar 9g.

1.3 Ruang Lingkup

Dengan mengacu pada permasalahan yang telah dirumuskan, maka hal-hal yang berkaitan dengan alat diberi batasan sebagai berikut :

1) Maksimum gaya yang dapat diberikan dan diukur adalah 9*g*.
2) Alat yang dirancang hanya alat uji roket, tidak termasuk muatan/*payload* roket.

1.4 Tujuan

Tujuan skripsi ini adalah merancang dan membuat suatu alat uji *G-force* untuk muatan roket yang dapat digunakan untuk menentukan *G-force* yang diinginkan, pengujian kekuatan/ ketahanan mekanik dan elektrik muatan roket KOMURINDO.

1.5 Sistematika Penulisan

Sistematika penulisan dalam penelitian ini sebagai berikut :

BAB I Pendahuluan

Memuat latar belakang, rumusan masalah, ruang lingkup, tujuan, dan sistematika pembahasan.

BAB II Tinjauan Pustaka

Membahas teori-teori yang mendukung dalam perencanaan dan pembuatan alat.

BAB III Metodologi

Berisi tentang metode-metode yang dipakai dalam melakukan perancangan, pengujian, dan analisis data.

BAB IV Perancangan

Perancangan dan perealisasian alat yang meliputi spesifikasi, perencanaan diagram blok, prinsip kerja dan realisasi alat.

BAB V Pengujian dan Analisis

Memuat aspek pengujian meliputi penjelasan tentang cara pengujian dan hasil pengujian. Aspek analisis meliputi penilaian atau komentar terhadap hasil-hasil pengujian. Pengujian dan analisis ini terhadap alat yang telah direalisasikan berdasarkan masing-masing blok dan sistem secara keseluruhan.

BAB VI Kesimpulan dan Saran

Memuat intisari hasil pengujian dan menjawab rumusan masalah serta memberikan rekomendasi untuk perbaikan kualitas penelitian yang akan datang.

BAB II
TINJAUAN PUSTAKA

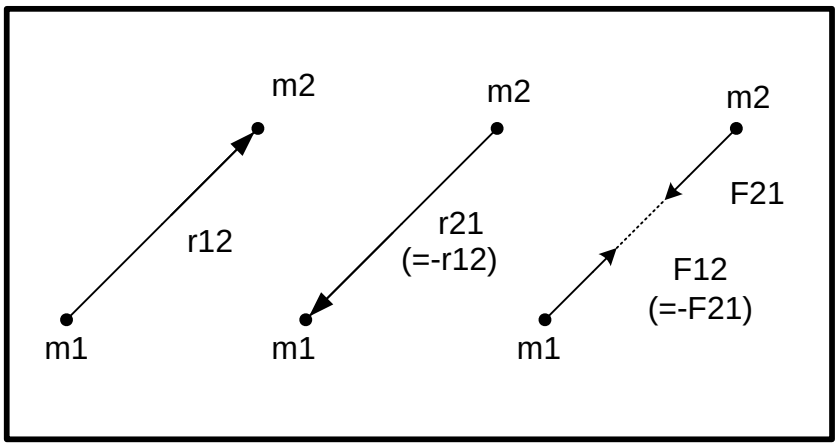
Alat uji fungsional *g-force* muatan roket adalah suatu alat yang berfungsi untuk memberikan gaya sentrifugal pada muatan roket yang akan diuji. Pengujian ini bertujuan untuk menggantikan gaya g yang diterima roket mulai dari roket *take off* sampai dengan roket mengalami separasi (pemisahan roket dengan muatan roket). Untuk mendapatkan nilai gaya g/*g-force* dari pengujian tersebut, dibutuhkan nilai RPM (*Revolution Per Minute*) dan jari-jari lengan putar alat untuk dikonversi menjadi nilai RCF (*Relative Centrifuge Force*)/*g-force.*

Dalam perancangan alat uji *g-force* muatan roket dibutuhkan pemahaman tentang berbagai hal yang mendukung perancangan. Pemahaman ini akan bermanfaat untuk merancang perangkat keras dan perangkat lunak model alat ini agar sistem dapat bekerja dengan baik.

Beberapa teori pendukung yang perlu dibahas dalam pembuatan sistem ini meliputi literatur mengenai, gaya g atau *g*-force, gaya sentrifugal, Kompetisi Muatan Roket Indonesia (KOMURINDO), motor DC, *rotary encoder*, dan FPGA (*Field Programmable Gate Array*) Nexys 2.

2.1 Gaya G atau *G-force*

G adalah singkatan dari gravitasi, sehingga *G-force* dapat diartikan sebagai gaya gravitasi. Suatu benda jika mendapat gaya sebesar 2g, artinya benda tersebut mendapat 2 kali gaya gravitasi. Benda diam memiliki gaya g sebesar 1g yang didefinisikan memiliki nilai 9,80665 m/s^2. Nilai g tersebut diperoleh dari Persamaan 2.1. Gambar 2.1 menunjukkan gaya-gaya suatu pasangan aksi reaksi 2 benda



Gambar 2.1 Gaya-gaya suatu pasangan aksi reaksi 2 benda

Sumber: *Physycs, 3rd edition*, David Halliday, 1978.

$$\overrightarrow{F12} = - G \frac{m2 . m1}{r21^3} r12 \qquad (2.1)$$

Keterangan:

- G = konstanta grafitasi terminal ($6{,}6720 . 10^{-11}$)
- $\overrightarrow{F12}$ = Gaya tarik suatu benda ($N.kg^2/m^2$)
- m1 = Massa bumi (kg)
- m2 = masa obyek (kg)
- r = jarak antara titik pusat massa bumi dengan pusat mass obyek (m)

2.2 Gaya sentrifugal

Suatu benda yang bergerak dengan kecepatan yang teratur baik dengan kecepatan yang berubah-ubah maupun dengan kecepatan yang konstan, maka benda tersebut akan mempunyai suatu bentuk lintasan tertentu, baik lintasan yang teratur bentuknya maupun lintasan yang acak atau random. Salah satu dari lintasan yang teratur bentuknya adalah lintasan dengan bentuk lengkung. Kita dapat menyebutkan dengan pasti kedudukan keseluruhan benda berputar dalam kerangka acuan kita jika kita mengetahui letak salah satu partikel dari benda tersebut dalam kerangka acuan. Jadi cukup meninjau gerak (dua dimensi) partikel dalam lingkaran. (David Halliday, 1978)

Sebuah benda yang mengalami gaya sentrifugal dengan jari-jari R dan berputar di dalam lingkaran horisontal terhadap sebuah titik tertentu. Besarnya gaya berat adalah:

$$W = m \cdot g \qquad (2.2)$$

atau

$$F_g = m \cdot g$$

Dimana:

- m = massa benda (kg)
- g = gaya gravitasi bumi (9,80665 m/s^2)

Besarnya $\mathbf{F_r}$ dapat juga ditentukan dengan persamaan sebagai berikut:

$$F_r = m \cdot a_r \qquad (2.3)$$

$$a_r = \frac{V^2}{R} \quad \Rightarrow \quad F_r = m \cdot \frac{V^2}{R} \qquad (2.4)$$

$$V = \omega \cdot R \quad \Rightarrow \quad F_r = \frac{m \cdot \omega^2 \cdot R^2}{R}$$

$$= m \cdot \omega^2 \cdot R \qquad (2.5)$$

Dimana F_r gaya tarik benda yang memiliki jari-jari (N), a_r percepatan yang dialami benda (m/s^2) dan V kecepatan (m/s). Menurut persamaan diatas, untuk mencari gaya sentrifugal yang dialami suatu benda dengan jari-jari (cm) dan massa benda (kg) dituliskan sebagai:

$$Fc = m \cdot (\omega)^2 \cdot R \qquad (2.6)$$

kecepatan sudut dalam satuan RPM (*Revolutions Per Minute*)/putaran per menit ditunjukkan dalam Persamaan 2.7, yang dinyatan sebagai berikut:

$$\omega = \frac{\Delta\theta}{\Delta t} \qquad (2.7)$$

Dimana $\Delta\Theta$ dapat didefinisikan selisih sudut putar (rad) dan Δt adalah waktu tempuh(s), sedangkan RPM didefinisikan sebagai jumlah putaran penuh yang ditempuh dalam 1 menit, sehingga dari definisi RPM tersebut dapat dihubungkan dengan Persamaan 2.7 sebagai kecepatan sudut, yang ditunjukkan dalam Persamaan 2.8 berikut:

$$\Delta\Theta = 2 \cdot \Pi \cdot N \qquad (2.8)$$

$$\Delta t = 60 \text{ sekon}$$

Dimana Π adalah 22/7 dan N adalah nilai dari RPM benda. Sehinga didapatkan Persamaan 2.9 :

$$\omega = \frac{2\,\Pi\,N}{60} \qquad (2.9)$$

Sehingga gaya sentrifugal dalam pola g adalah:

$$g = \omega^2 \cdot R \qquad (2.10)$$

$$g = \frac{(2\,\Pi\,N)^2 \cdot R}{3600} \qquad (2.11)$$

Selama gerak melingkar percepatan adalah perkalian jari-jari (cm) dengan kuadrat dari kecepatan sudut. Hubungan antara percepatan dengan "g" secara relative adalah RCF (*Relative Centrifugal Force*). Percepatan tersebut diukur dalam kelipatan "g" atau x "g" yang merupakan percepatan gravitasi bumi. Tabel 2.1 menunjukkan nilai konversi RCF ke RPM dengan jari-jari lengan putar alat sebesar 15 cm. Persamaan RCF dapat dinyatakan sebagai berikut:

$$RCF = \frac{F_c}{F_g} \qquad (2.12)$$

$$RCF = \frac{m \,.\, (\omega)^{2} \,.\, R}{m \,.\, g}$$

$$RCF = \frac{(\omega)^{2} \,.\, R}{g} \qquad (2.13)$$

$$RCF = \frac{(2\,\Pi\,N)^{2} \,.\, R}{3600.9{,}8.100}$$

$$RCF = \frac{4\,\Pi^{2}(N)^{2} \,.\, R}{3600.9{,}8.100}$$

$$RCF = 1{,}118 \,.\, 10^{-5} \,.\, R\,(N)^{2} \qquad (2.14)$$

Keterangan:

- g = gaya gravitasi bumi = 9,8 m/s^2
- m = massa benda (kg)
- ω = Kecepatan sudut (rad/s)
- R = jari-jari benda ke pusat rotor (cm)
- N = putaran per menit (RPM)
- RCF = gaya sentrifugal relatif (x “g”)

Tabel 2.1 Contoh Nilai konversi RCF ke RPM dengan jari-jari lengan putar alat 15 cm

RCF (x "g")	RPM
0	0
1	77,22
2	109,20
3	133,75
4	154,44
5	172,67
6	189,15
7	204,30
8	218,41

2.3 KOMURINDO

KOMURINDO (Kompetisi Muatan Roket Indonesia) merupakan suatu kontes yang dapat menumbuhkembangkan kemampuan mahasiswa dalam hal rancang bangun teknologi peroketan, baik dari sisi roket maupun muatannya mulai dari tahap desain, membuat, menguji hingga uji terbang. Kompetisi ini diadakan setiap tahun sejak tahun 2009 dengan tujuan sebagai sarana untuk mengajak, mendidik dan menarik minat mahasiswa dalam rangka menyiapkan bibit unggul peneliti dan ahli peroketan di Indonesia masa depan.

Melalui pemahaman perilaku roket peluncur yang diterapkan pada persyaratan operasional muatan roket, mahasiswa akan mampu memahami teknologi peroketan, yang pada perkembangannya, muatan hasil rancang bangun

mahasiswa ini dapat menjadi cikal bakal lahirnya satelit Indonesia hasil karya bangsa Indonesia secara mandiri. Sedangkan roket peluncurnya, dalam skala besar dan teknologi yang lebih canggih dapat dikembangkan menjadi Roket Peluncur Satelit. Disamping itu, Kompetisi roket ini juga dapat meningkatkan rasa persatuan dan nasionalisme mahasiswa khususnya serta masyarakat pada umumnya di bidang teknologi peroketan. Juga dapat memperpendek jarak perbedaan penguasaan iptek dirgantara dan memperluas penyebarannya diantara perguruan tinggi di seluruh Indonesia.

Untuk meningkatkan kualitas KOMURINDO yang sejak kompetisi pertama tahun 2009 hingga tahun lalu (2012) hanya memiliki satu kategori, yaitu muatan roket (payload), maka untuk tahun 2013 temanya dikembangkan bukan hanya pada payload namun juga pada roketnya. Jika pada kategori muatan roket temanya adalah High Rate Attitude Data Monitoring and Surveillance Payload maka untuk kategori roket tema yang diangkat adalah Autonomous Low Speed EDF Rocket, yaitu rancang bangun roket yang dapat dikendalikan menggunakan motor roket tipe EDF (Electric Ducted Fan). Untuk permulaan, roket yang didesain tidak boleh melebihi berat 1,5 kg dengan panjang tubuh dari kepala hingga ekor tidak boleh melebihi 1 m. Kompetisi rancang bangun roket kendali mini melalui ajang KOMURINDO 2013 ini diharapkan dapat menjadi salah satu media pemercepat dalam memasyarakatkan teknologi roket kendali di kalangan mahasiswa dan perguruan tinggi.

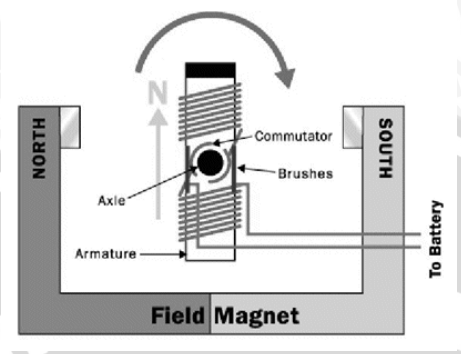
2.4 Motor DC

2.4.1 Prinsip Kerja Motor DC

Segulung kawat yang dialiri arus listrik dan ditempatkan di dalam suatu medan magnet akan mengalami gaya yang sebanding dengan arus dan kekuatan medan magnetnya. Gaya yang ditimbulkan disebut dengan Gaya Lorentz yang dapat dirumuskan sebagai berikut:

$$F = I.L.B \ \ldots (N) \qquad (2.15)$$

Dalam hal ini *I* adalah arus yang mengalir (*A*), L adalah panjang kawat (*m*) dan B adalah kerapatan *fluks* magnet (*Wb*/m^2). Gambar penampang melintang dari motor DC ditunjukkan dalam Gambar 2.2.



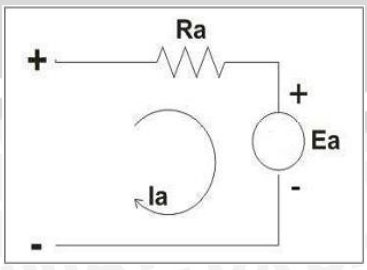
Gambar 2.2 Penampang melintang motor DC

Sumber: www.electronics-scheme.com

Persamaan 2.16 menunjukkan prinsip dasar sebuah motor, di mana terjadinya proses perubahan energy listrik (*I*) menjadi energi mekanik (*F*). Jika motor mempunyai jari-jari sebesar *r*, maka akan menimbulkan torsi sebesar

$$\text{Torque} = F.\ r = B\ .\ I\ .\ L\ .\ r\ \ldots\ (Nm) \qquad (2.16)$$

Kopel (Torque) didefinisikan sebagai aksi dari suatu gaya pada benda yang cenderung menyebabkan benda berputar. Jadi ukuran kecenderungan dari suatu jangkar motor untuk beroutar disebut kopel dari motor (Eugene Lister, 1984). Pada saat dibangkitkan, konduktor akan bergerak di dalam medan magnet dan akan menimbulkan gaya gerak listrik (ggl) yang merupakan reaksi lawan terhadap tegangan penyebabnya. Proses konversi energi listrik menjadi energi mekanik dapat berlangsung jika tegangan sumber lebih besar dari gaya gerak listrik lawan. Gambar 2.3 menunjukkan proses kerja motor DC.



Gambar 2.3 Proses Kerja Motor DC

Sumber: www.mikron123.com

Motor dapat berputar jika tegangan masukan motor lebih besar dari ggl yang timbul. Hubungan antara tegangan sumber dan ggl lawan seperti ditunjukkan dalam Gambar 2.3 dapat dirumuskan sebagai berikut:

$$Ea = Vin - Ia.Ra \ldots(V) \quad (2.17)$$

Dalam hal ini Ea adalah tegangan pada jangkar, V_{in} adalah tegangan masukan, I_a adalah arus jangkar dan R_a adalah tahanan jangkar, sedangkan induksi yang timbul adalah:

$$Ea = C\ n\ \Phi \ldots(V) \quad (2.18)$$

Dengan C adalah konstanta, n adalah kecepatan motor, dan Φ adalah fluks magnetik yang besarnya sebanding dengan arus penguatan torsi. Torsi pada motor juga sebanding dengan *fluks* magnetik dan arus. Hal ini ditunjukkan dalam Persamaan 2.19 :

$$\tau = C\ \Phi\ Ia \ldots(Nm) \quad (2.19)$$

Jika diketahui kecepatan sudut ω adalah :

$$\omega = \frac{2\pi.n}{60} \ldots(\text{rad/s}) \quad (2.20)$$

Maka hubungan torsi dan kecepatan motor adalah :

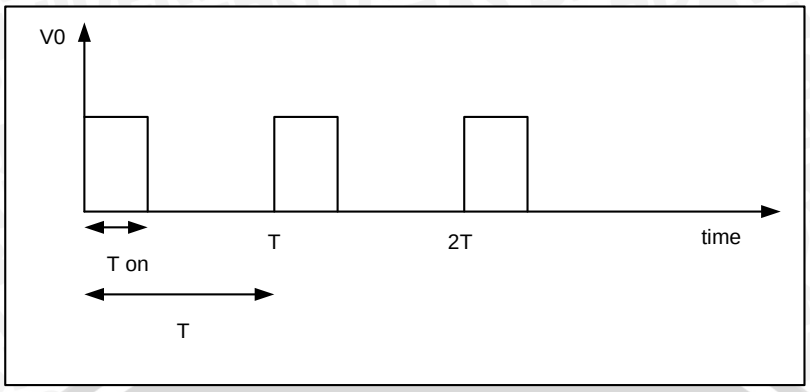
$$\tau = P/\omega \ \ldots\ (Nm) \quad (2.21)$$

$$\tau = \frac{P}{2\pi n/60} \ldots(Nm) \quad (2.22)$$

2.4.2 PWM (*Pulse Width Modulation*)

PWM (*Pulse Width Modulation*) digunakan untuk mengatur kecepatan dari motor DC. Di mana kecepatan motor DC tergantung pada besarnya *Duty cycle* yang diberikan pada motor DC tersebut.

Pada sinyal PWM, frekuensi sinyal konstan sedangkan *Duty cycle* bervariasi dari 0%-100%. Dengan mengatur *Duty cycle* akan diperoleh keluaran yang diinginkan. Sinyal PWM (*Pulse width Modulation*) secara umum ditunjukkan dalam Gambar 2.4.



Gambar 2.4. Sinyal PWM secara umum

Sumber: www.electronics-scheme.com

$$Dutycycle = \frac{Ton}{T}\ x\ 100\%\ \ldots\ (\%) \quad (2.23)$$

Dengan :

T on = Periode logika tinggi

T = Periode keseluruhan

$$Vdc = Dutycycle\ x\ Vcc\ \ldots(V) \quad (2.24)$$

Sedangkan frekuensi sinyal dapat ditentukan dengan rumus berikut :

$$fon = \frac{fclk\ I/O}{N.256}\ \ldots\ (Hz) \quad (2.25)$$

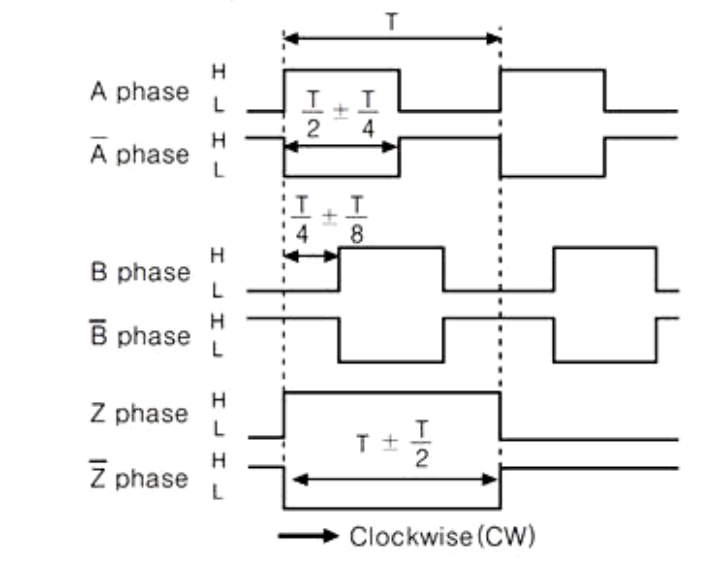
2.5 Rotary Encoder E40S6

Rotary encoder adalah divais elektromagnetik yang dapat memonitor gerakan dan posisi. Rotary encoder umumnya menggunakan sensor optik untuk menghasilkan serial pulsa yang dapat diartikan menjadi gerakan, posisi, dan arah. Sehingga posisi sudut suatu poros benda berputar dapat diolah menjadi informasi berupa kode digital oleh rotary encoder untuk diteruskan oleh rangkaian kendali.

Sebuah *incremental position encoder* terdiri dari sebuah piringan/*disk* yang bergerak secara linier atau dengan inersia yang rendah. Dikendalikan oleh bagian yang posisinya sudah ditetapkan. Bentuk fisik yang digunakan untuk daerah pembeda pola dapat berupa bahan magnetic, elektrik, atau optikal. Dimana dalam hal ini, output dasar yang dihasilkan sensor dalam bentuk sebuah pulsa dengan *duty cycle* sebesar 50%. (Ramon and John, 1991)

LED ditempatkan pada salah satu sisi piringan sehingga cahaya akan menuju ke piringan. Di sisi yang lain suatu photo-transistor diletakkan sehingga photo-transistor ini dapat mendeteksi cahaya dari LED yang berseberangan. Piringan tipis tadi dikopel dengan poros motor, atau divais berputar lainnya yang ingin kita ketahui posisinya, sehingga ketika motor berputar piringan juga akan ikut berputar. Apabila posisi piringan mengakibatkan cahaya dari LED dapat mencapai photo-transistor melalui lubang-lubang yang ada, maka photo-transistor akan mengalami saturasi dan akan menghasilkan suatu pulsa gelombang persegi. Semakin banyak deretan pulsa yang dihasilkan pada satu putaran menentukan akurasi rotary encoder tersebut, akibatnya semakin banyak jumlah lubang yang dapat dibuat pada piringan menentukan akurasi rotary encoder tersebut.

Rotary encoder E40S6 merupakan rotary encoder tipe incremental dengan 3 channel output yang disebut channel A, channel B, dan channel Z. Setiap channel menghasilkan satu pulsa per putaran yang berguna untuk menghitung jumlah putaran yang terjadi. Gambar 2.5 menunjukkan kondisi sinyal keluaran Channel A, B, dan Z.



Gambar 2.5. Timing diagram channel Output Rotary Encoder E40S6

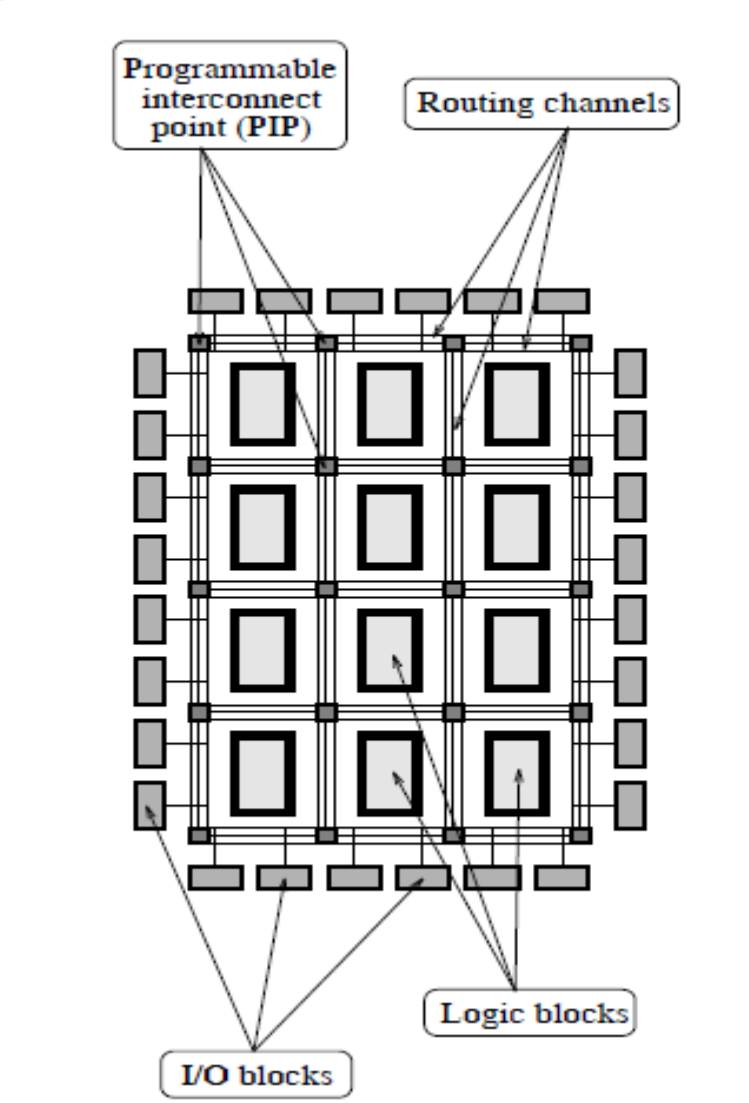
Sumber: Autonics-Rotary Encoder E40S6 datasheet

2.6 FPGA (*Field Programmable Gate Array*) Nexys 2

FPGA (*Field Programmable Gate Array*) adalah komponen elektronika dan semikonduktor yang mempunyai komponen gerbang terprogram (*Programmable Logic*) dan sambungan terprogram berdasarkan *array*. Komponen gerbang yang dimiliki meliputi jenis gerbang logika biasa (AND, OR, XOR, NOT) maupun jenis fungsi matematis dan kombinatorik yang lebih kompleks (decoder, adder, subtracktor, multiplier dan lain-lain). Blok-blok komponen didalam FPGA bisa juga mengandung berupa elemen memory (register) mulai flip-flop sampai pada RAM (*Random Acces Memory*).

Secara umum arsitektur bagian dalam dari IC FPGA terdiri atas tiga elemen utama yaitu *Input/Output Blok (IOB)*, *Configurable Logic Block (CLB)* dan *Interkoneksi*. Arsitektur FPGA tersebut memungkinkan setiap logic block dapat digunakan dan diimplementasikan ke dalam suatu fungsi yang sederhana. Sebagai contoh hal itu mungkin untuk mengkofigurasi blok logika menjadi sebuah seperti fungsi yang memiliki 3 input, seperti gerbang logika (AND, OR, NAND, dan lain-lain) atau sebuah elemen penyimpan (D Flip-flop, D Latch, dan lain-lain) (Clive Maxfield, 2009). Gambar 2.6 menunjukkan arsitektur FPGA.

- *Configurable Logic Blocks* :
  - *Look up table based complex structure*
  - *Implement the sequential circuit*
- *Programmable Interconnect* :
  - Berisi *wire segments* dan *programmable switches*
  - Menghubungkan antar *Configurable Logic Blocks* yang berbeda
- *Input/output block* :
  - Sebagai antar muka/*interface* antara external *package* pin dari device dan internal *user logic*



Gambar 2.1. Arsitektur FPGA

Sumber : *Digital Signal Processing with Field Programmable Gate Array*. U, Meyer and Baese, 2007.

BAB III
METODOLOGI

Untuk menyelesaikan rumusan masalah dan merealisasikan tujuan penelitian yang terdapat di bab pendahuluan maka diperlukan metode untuk menyelesaikan masalah tersebut.

3.1 Studi Literatur

Studi literatur dilakukan untuk mempelajari teori penunjang sistem yang dibutuhkan dalam perencanaan dan pembuatan alat. Teori yang diperlukan antara lain berkaitan dengan FPGA (*Field Programmable Gate Array*) Nexys 2, motor DC, *rotary encoder*, gaya G atau *G-force*, gaya sentrifugal dan sentripetal..

3.2 Penentuan Spesifikasi Alat

Spesifikasi alat secara global ditetapkan terlebih dahulu sebagai acuan dalam perancangan selanjutnya. Spesifikasi alat yang direncanakan adalah sebagai berikut :

1) FPGA nexys 2 digunakan sebagai *controller* utama sistem.
2) Driver motor menggunakan Transistor MOSFET IRFZ 44N.
3) Rotary encoder yang digunakan adalah jenis incremental rotary encoder dengan 360 pulsa/data dalam setiap putaran tipe E40S6.
4) G-force yang diberikan oleh alat adalah 1*g* sampai dengan 9*g*.
5) Program untuk memprogram FPGA menggunakan Xilinx ISE 14.1 dan Digilent Adept untuk writer FPGA.
6) Catudaya motor menggunakan 2 buah *accu* 12V 3,5 Ah yang dipasang secara seri dan voltage power supply 5 V.

3.3 Perancangan dan Pembuatan Alat

3.3.1 Perancangan dan Pembuatan Perangkat Keras (*Hardware*)

1). Pembuatan blok diagram lengkap sistem
2). Pembuatan mekanik alat

3). Pembuatan dan perhitungan komponen yang akan digunakan

4). Merakit perangkat keras masing-masing blok.

3.3.2 Perancangan dan Penyusunan Perangkat Lunak

Setelah kita mengetahui seperti apa perangkat keras yang dirancang, maka kita membutuhkan perangkat lunak untuk mengendalikan dan mengatur kerja dari alat ini. Desain dan parameter yang telah dirancang kemudian diterapkan kedalam FPGA Nexys2 dengan menggunakan bahasa VHDL dan *compiler* Digilent Adept.

3.4 Pengujian Alat

Untuk memastikan bahwa sistem ini berjalan sesuai yang direncanakan maka perlu dilakukan pengujian alat meliputi perangkat keras *(hardware)* yang dilakukan baik per blok maupun keseluruhan sistem.

3.4.1 Pengujian Tiap Blok

Pengujian per blok dilakukan dengan tujuan untuk menyesuaikan nilai masukan dan nilai keluaran tiap-tiap blok sesuai dengan perancangan yang dilakukan sebelumnya.

3.4.2 Pengujian Keseluruhan Sistem

Pengujian sistem secara keseluruhan dilakukan dengan tujuan untuk mengetahui unjuk kerja alat setelah perangkat keras dan perangkat lunak diintegrasikan bersama.

3.5 Pengambilan Kesimpulan

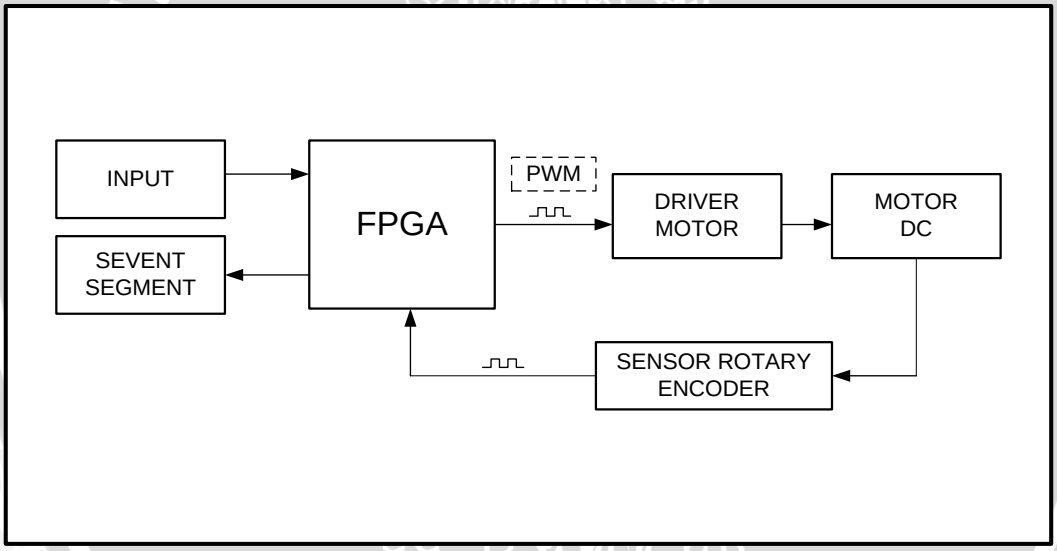
Pengambilan kesimpulan dilakukan setelah didapatkan hasil dari pengujian. Jika hasil yang diperoleh telah sesuai dengan spesifikasi yang direncanakan maka alat tersebut telah memenuhi harapan dan memerlukan pengembangan untuk penyempurnaannya.

BAB IV
PERANCANGAN DAN PEMBUATAN ALAT

Bab ini menjelaskan tentang perancangan dan pembuatan alat uji *g-force* muatan roket KOMURINDO mulai dari diagram blok sistem, desain mekanik alat, *perancangan perangkat keras, dan perancangan perangkat lunak. Perancangan dan* pembuatan dilakukan secara bertahap dan sistematis, sehingga nantinya akan memudahkan dalam analisis sistem.

4.1 Perancangan Sistem

Secara garis besar, diagram blok perancangan *hardware* sistem secara keseluruhan ditunjukkan dalam Gambar 4.1.



Gambar 4.1 Diagram blok sistem secara keseluruhan

Input sistem ini berasal dari 8 bit binary input dari switch FPGA. Nilai input range yaitu dari 0 sampai 256 akan diproses FPGA untuk membangkitkan sinyal PWM. Didalam FPGA nilai PWM akan disesuaikan dengan nilai INPUT dan nilai *feedback control*, sehingga terdapat pengaturan nilai PWM secara otomatis dari *feedback control* agar PWM sesuai dengan nilai input yang diinginkan. Sinyal PWM dari FPGA diteruskan ke driver motor yang berupa

Optocoupler dan transistor FET IRFZ 44 N sebagai pengendali kecepatan motor dengan menggunakan mode saturasi dan *cut off*. Nilai RPM atau putaran per menit dari motor didapat dari nilai PWM untuk menggerakkan lengan putar alat. Saat lengan putar berputar, nilai kecepatan putaran akan didapat melalui sensor *rotary encoder* yang berupa pulsa *high* dan *low*. Pulsa dari *rotary encoder* masuk kedalam input FPGA untuk diproses dan dikonversi menjadi nilai RPM. Data RPM dari *rotary encoder* dibandingkan dengan data RPM INPUT untuk digunakan sebagai *feedback control*. Hasil dari perhitungan nilai RPM didalam FPGA ditampilkan di 4 digit seven segment, yaitu berupa nilai RPM.

4.2 Perancangan Perangkat Keras

4.2.1 Perancangan Mekanik Alat

Perancangan mekanik alat pada penelitian ini terdapat beberapa komponen utama yang ditunjukkan dalam Gambar 4.2 dan 4.3. Simbol A dalam Gambar 4.2 menunjukkan pemberat alat yang berfungsi untuk menjaga keseimbangan lengan putar saat diberi beban muatan roket. Simbol B dalam Gambar 4.2 menunjukkan bagian yang berbentuk silinder yang berfungsi sebagai tempat muatan roket diletakkan. Simbol C dalam Gambar 4.2 menunjukkan sensor *rotary encoder* yang berfungsi menerima data pulsa kecepatan putaran alat. Simbol D dalam Gambar 4.2 menunjukkan motor DC 24V yang berfungsi untuk menggerakkan lengan putar alat. Simbol E dalam Gambar 4.3 menunjukkan gear penggerak dengan perbandingan 5:1, yaitu gear kecil dengan diameter 4 cm dan gear besar 12 cm.

Diameter lengan putar alat sebesar 30 cm, hal ini mengacu pada perhitungan RCF (*Relative Centrifuge Force*). Perancangan alat uji ini adalah mampu memberikan gaya g atau RCF sebesar maksimal 9g, RPM maksimal yang dapat dihasilkan alat sebesar 231 RPM. Sehingga untuk mencari nilai diameter alat ditunjukkan sebagai berikut:

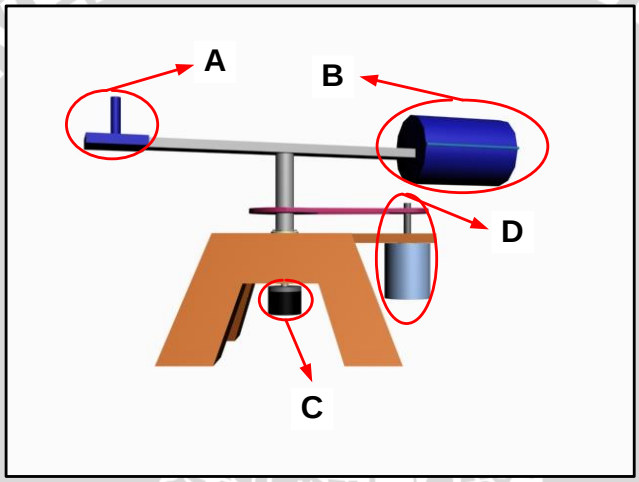
$$\mathrm{RCF} = 1{,}118 \,.\, 10^{-5} \,.\, \mathrm{R}\,(231)^2$$

$$\mathrm{R} = \frac{\mathrm{RCF}}{1{,}118 \,.\, 10^{-5} \,.\, (231)^2}$$

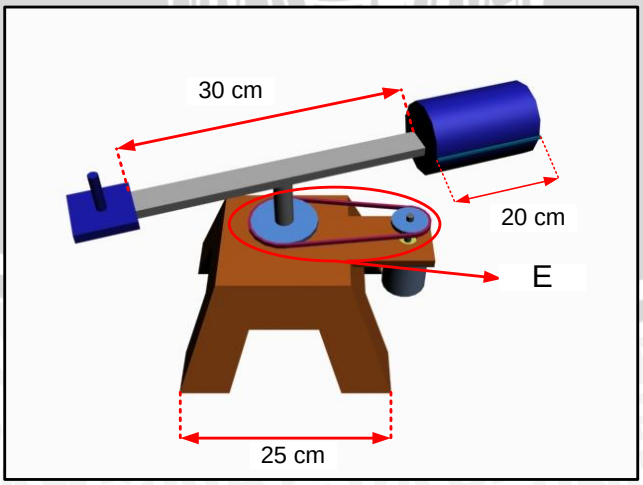
$$R = \frac{9}{1{,}118 \,.\, 10^{-5} \,.\, (231)^2}$$

$$R = 15{,}086 \text{ cm}$$

Dalam proses perhitungan, didapatkan jari-jari lengan putar alat sebesar 15,086 cm. Untuk memudahkan pembuatan mekanik, maka nilai jari-jari (R) tersebut dibulatkan menjadi 15 cm atau diameter sebesar 30 cm. Ukuran tabung untuk tempat *payload* roket menyesuaikan ukuran *payload*roket yang telah ditetapkan oleh panitia KOMURINDO, yaitu diameter maksimal sebesar 12 cm dan panjang tabung maksimal 20 cm.



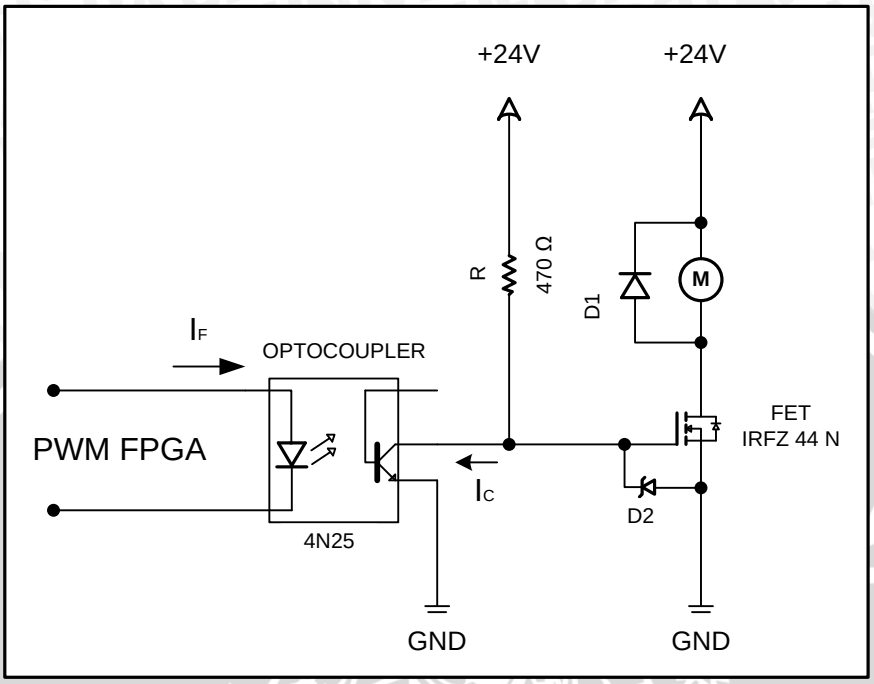
Gambar 4.2. Rancangan Alat Tampak Samping



Gambar 4.3 Rancangan Alat Tampak Prespektif

4.3 Perancangan Rangkaian Elektrik

4.3.1 Rangkaian Driver Motor



Gambar 4.4 Rangkaian Pengontrol Kecepatan dengan Menggunakan E-MOSFET

Pada perancangan penelitian ini digunakan komponen E-MOSFET kanal N dengan masukan berupa sinyal PWM FPGA. Gambar 4.4 menunjukkan rangkaian pengontrol kecepatan motor.

Jenis FET yang digunakan adalah IRFZ44N dengan spesifikasi

- $V_{DS\ max}$ = 55V
- $I_{D\ max}$ = 49A
- $R_{DS\ on\ max}$ = 22 Ω
- $V_{GS\ max}$ = 20V
- $V_{GS\ threshold}$ = 3V

IRFZ44N digunakan karena IRFZ44N memiliki $I_{D\ max}$ yang cukup besar dan $R_{DS\ on\ max}$ lebih kecil dibandingkan beberapa jenis E-MOSFET kanal N yang lain. Hal ini mengakibatkan IRFZ44N tidak cepat menjadi panas ketika dilewati arus yang besar.

Fungsi dioda daya D1 pada gambar rangkaian elektrik driver motor berfungsi sebagai pengaman komponen MOSFET IRFZ44N pada saat sistem berputar kemudian tiba-tiba berhenti, sehingga terjadi arus sisa pada lilitan motor DC.

Untuk aplikasi pengontrol kecepatan motor, FET selalu dikondisikan dalam keadaan saturasi atau *cut off*-nya. Hal ini dimaksudkan agar tidak terlalu banyak daya yang terbuang dalam FET itu sendiri.

Untuk E-MOSFET kanal N syarat agar komponen dalam kondisi *cut off* adalah ketika $V_{GS} < V_{threshold}$. Dengan $V_{threshold}$ IRFZ44N = 3V,maka V_{GS} *cut off* yang digunakan kurang dari 3V. V_{GS} yang digunakan adalah 0V.

Syarat agar E-MOSFET kanal N dalam kondisi aktif saturasi adalah ketika $V_{GS} > V_{threshold}$, dan $V_{DS} > (V_{GS} - V_{threshold})$. Menggunakan V_{DS} sebesar 24V, maka $V_{GS} < 27V$. Karena $V_{GS\ max} = 20V$, maka $3V < V_{GS} < 20V$. V_{GS} saturasi yang digunakan sebesar 10V. Untuk membatasi V_{GS} saturasi ini digunakan dioda zener (D2) 10V.

Perhitungan nilai resistor untuk tegangan masukan E-MOSFET sebagai berikut

$$R = \frac{24V - V_{CE\ sat}}{I_C}$$
$$= \frac{24V - 0{,}3V}{100\ X\ 10^{-3}}$$
$$= 237\ \Omega.$$

Nilai R yang digunakan sebesar 470Ω.

$$I_C = \frac{24V - V_{CE\ sat}}{R}$$
$$= \frac{24V - 0{,}3V}{470\Omega}$$
$$= 50{,}4\ \text{mA}$$

Optocoupler yang digunakan adalah tipe 4N25 dengan spesifikasi sebagai berikut:

- Turn off time 4,5 μs

- Turn on time 2,8 μs
- V_F = 1,5 V
- I_{Fmax} = 60 mA
- IC_{max} = 100 mA
- $V_{CE(sat)}$ = 0,5 V
- CTR_{typ} = 500%

Alasan digunakan komponen optocoupler 4N25 adalah karena 4N25 memiliki turn on time yang cukup kecil yaitu 2,8 μs dan turn off time 4,5 μs. Dengan *turn on time* dan *turn off time* yang kecil memungkinkan *optocoupler* mampu meneruskan sinyal dengan frekuensi tinggi.

Nilai I_C yang dihasilkan adalah sebesar 50,4 mA, nilai ini masih dibawah nilai arus maksimal (I_{cmax}) optocoupler sebesar 100 mA. Dengan nilai CTR (*Current Transfer Ratio*) sebesar 500%, arus masukan I_F *optocoupler* dapat dihitung dengan persamaan sebagai berikut:

$$CTR = \frac{I_C}{I_F} \text{ x } 100\% \qquad (4.1)$$

$$I_F = \frac{I_C}{CTR} \text{ x } 100\%$$

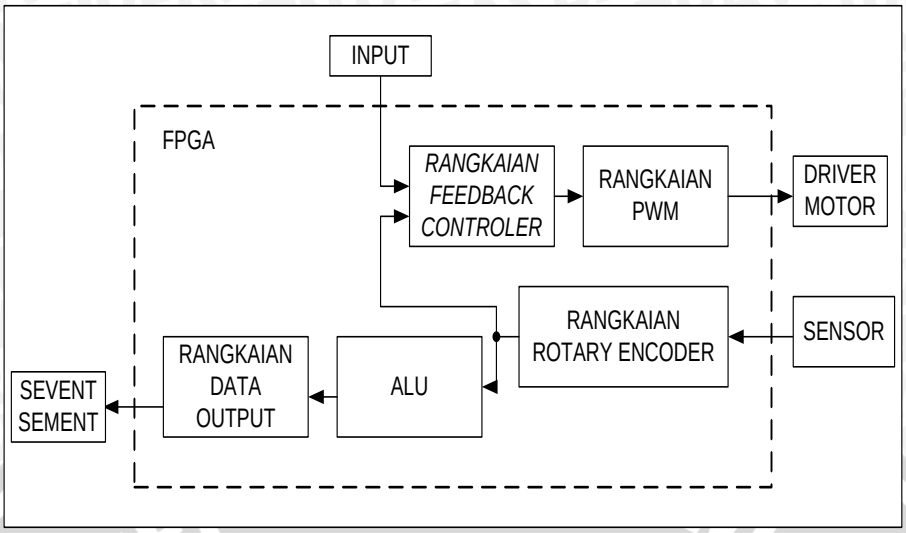
$$I_F = \frac{50{,}4\ mA}{500} \text{ x } 100\%$$

$$I_F = 10{,}08 \text{ mA}$$

Jadi arus yang mengalir dalam LED optocoupler sebesar 10,08 mA. Masih lebih kecil arus maksimal output FPGA sebesar 50 mA dan I_{Fmax} optocoupler sebesar 60 mA.

4.4 Perancangan dan Pembuatan Perangkat Lunak

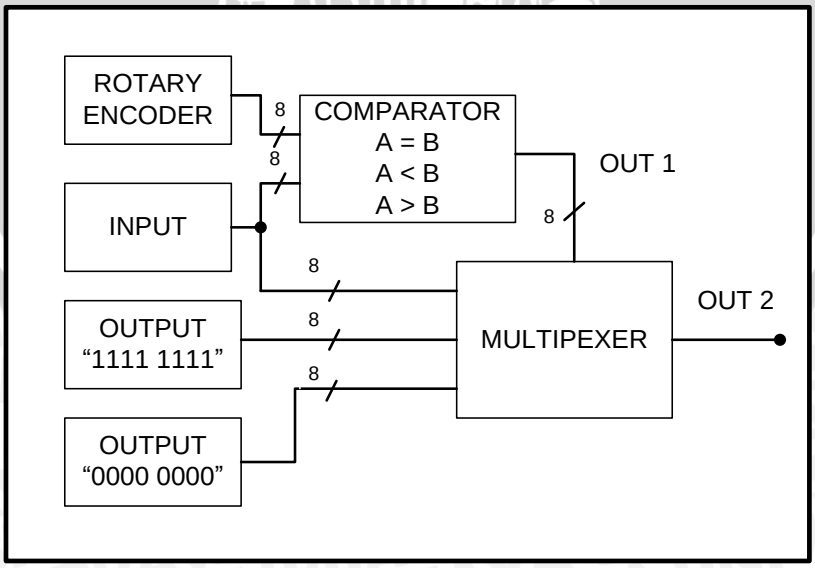
Dalam skripsi ini perancangan perangkat lunak terdiri 5 komponen yang disatukan menjadi 1 bagian sistem, yaitu perancangan perangkat lunak rangkaian pembangkit sinyal pwm, rangkaian *feedback controler*, rangkaian penerima pulsa rotary encoder, dan rangkaian data *seven segment.* Gambar 4.5 menunjukkan perancangan perangkat lunak sistem dalam FPGA.



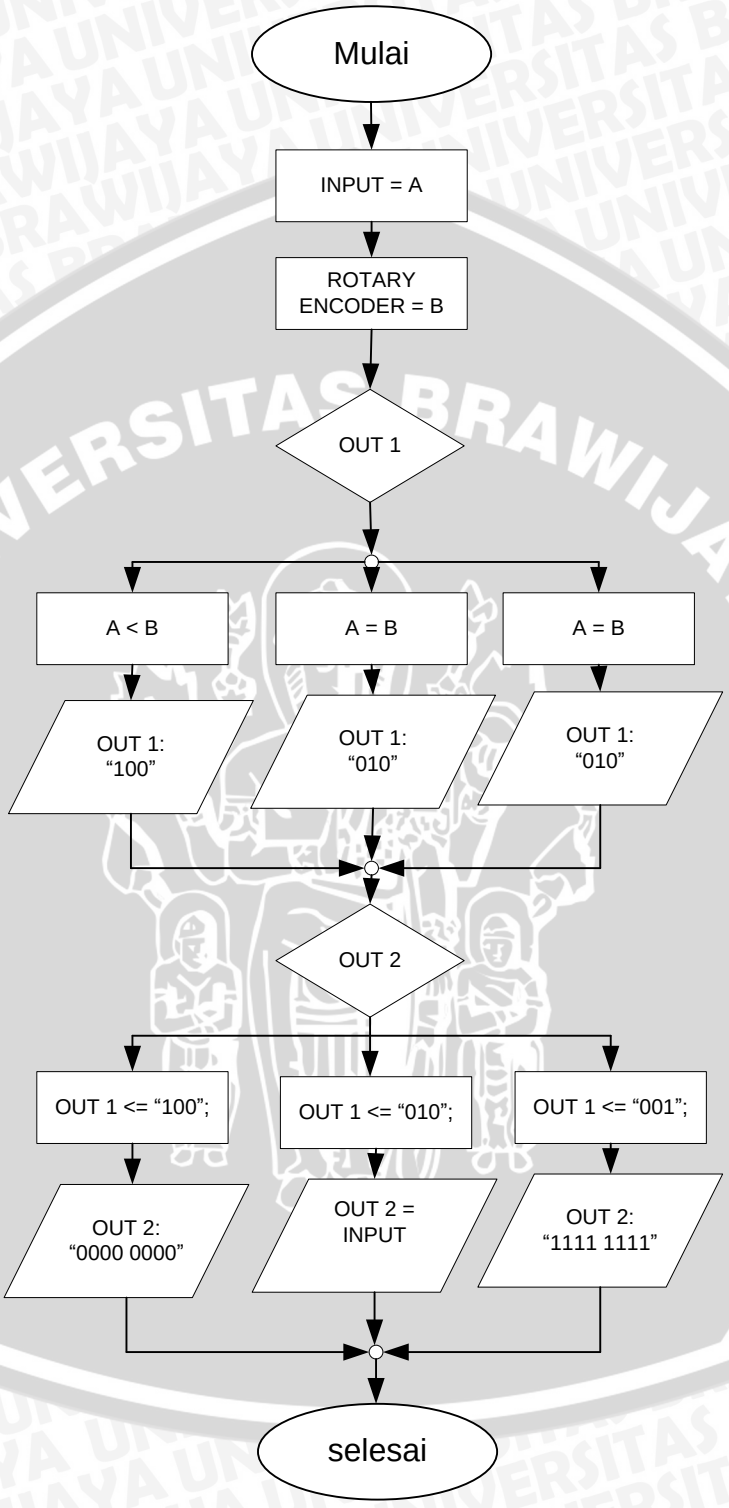
Gambar 4.5 Perancangan perangkat lunak sistem dalam FPGA

4.4.1 Perancangan rangkaian *feedback controler*

Perancangan rangkaian *feedback controler* adalah dengan membandingkan data dari input dengan data dari rangkaian rotary encoder. Hasil pembandingan tersebut (OUT 1) menjadi sinyal *selector* rangkaian multiplekser, hasil output multiplekser (OUT 2) menjadi input dari blok rangkaian PWM. Data 8 bit output yang dipilih oleh rangkaian multiplekser adalah data 8 bit input, data output 8 bit logika '1', dan data output 8 bit logika '0'. Gambar 4.6 dan 4.7 menunjukkan diagram blog dan flowchart rangkaian *feedback controller*.



Gambar 4.6 Diagram blok rangkaian *feedback controler*

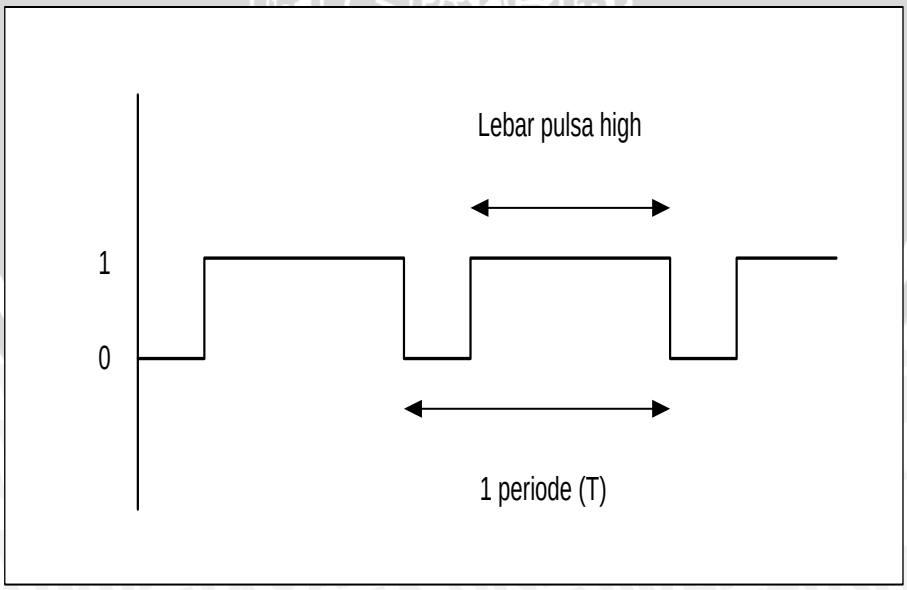


Gambar 4.7 Flowchart perancangan rangkaian *feedback controller*

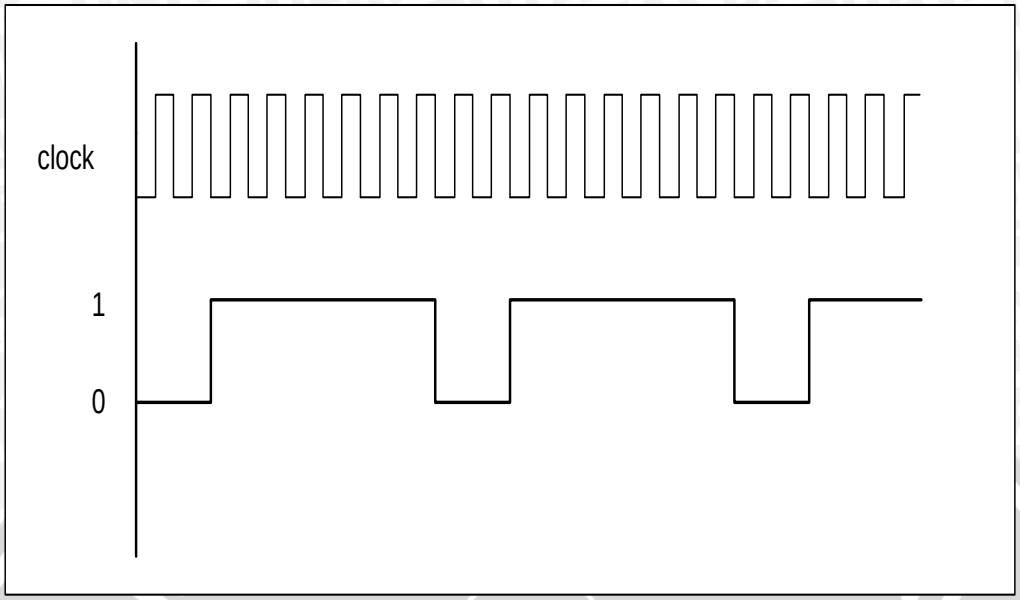
Dalam *flowchart* perancangan rangkaian *feedback controller* data input (A) dan data pulsa *rotary encoder* (B) dikomparasi dengan 3 logika output (OUT1) . Saat A<B maka data output (OUT1) adalah “100”. Saat A=B maka data output (OUT1) adalah “010”. Saat A>B maka data output (OUT1) adalah “001”. Data OUT1 selanjutnya digunakan sebagai data INPUT (*selector*) multiplekser. Saat OUT1=”100”, maka data yang dipilih dan dikeluarkan di OUT2 adalah data “00000000”. Saat OUT1=”010”, maka data yang dipilih dan dikeluarkan di OUT2 adalah data INPUT. Saat OUT1=”001”, maka data yang dipilih dan dikeluarkan di OUT2 adalah data “11111111”, dan proses selesai.

4.4.2 Perancangan rangkaian PWM

Perancangan rangkaian PWM (*Pulse Widht Modulation*) adalah dengan mengatur nilai *duty cycle* pada pulsa yang dihasilkan oleh sinyal PWM. *Duty cycle* merupakan perbandingan besarnya nilai pulsa high dengan nilai 1 periode pulsa dikali dengan 100%. Gambar 4.8 menunjukkan *duty cycle* dengan nilai 75%. Dari gambar tersebut, maka pulsa PWM dapat diimplementasikan ke dalam sebuah timing diagram yang ditunjukkan dalam Gambar 4.9.



Gambar 4.8 Sinyal PWM dengan nilai *duty cycle* 75%



Gambar 4.9 timing diagram sinyal PWM

Gambar 4.9 menunjukkan timing diagram sinyal PWM yang besarnya periode dan pulsa *high* dapat diketahui dari besarnya jumlah *clock.* Untuk mengetahui besarnya jumlah dari *clock* dibutuhkan rangkaian counter. Besarnya bit output counter ditentukan jumlah dari jumlah *clock* dalam 1 periode dan frekuensi sinyal PWM yang dapat dirumuskan sebagai berikut:

$$m = 2^n \quad (4.2)$$

$$f = \frac{z}{m} \quad (4.3)$$

keterangan:

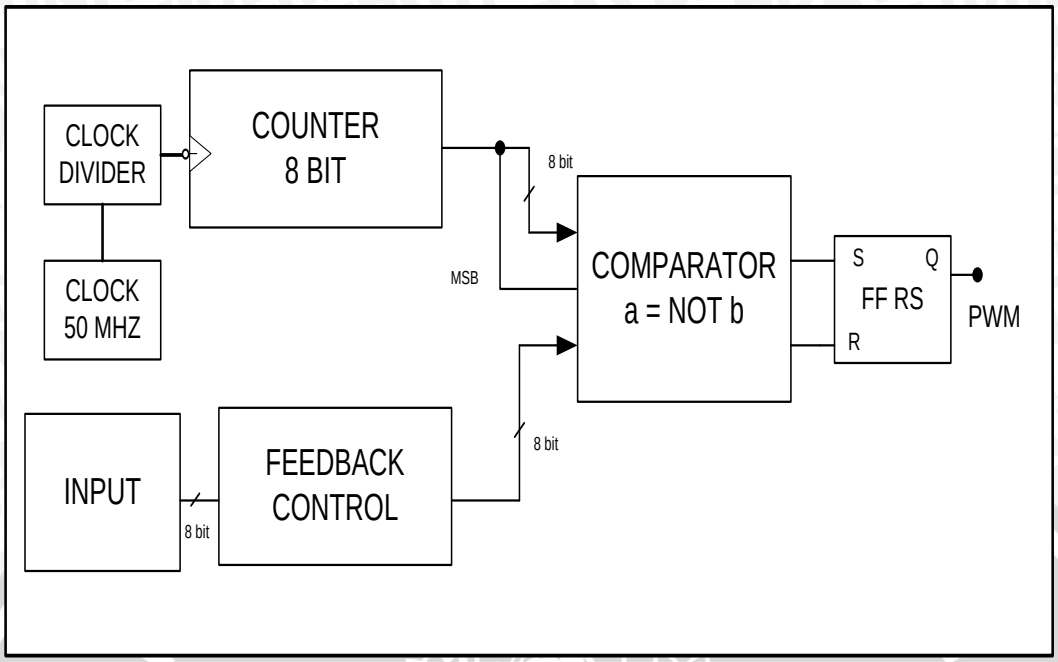
m : jumlah clock dalam 1 periode

n : jumlah bit output counter

z : frekuensi clock (Hz)

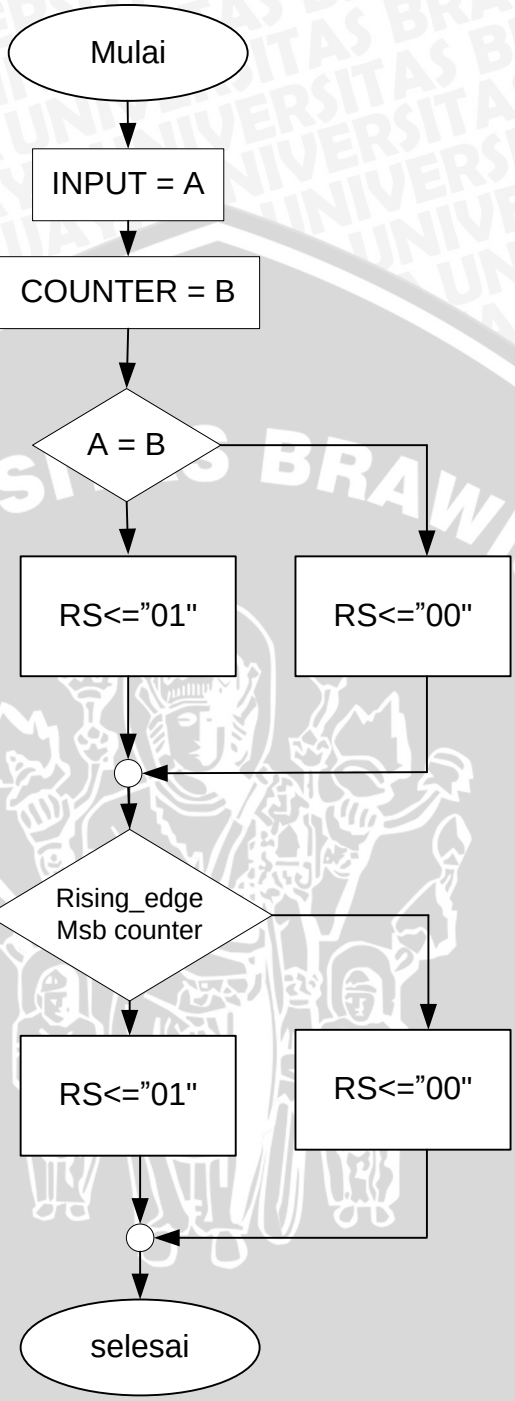
f : frekuensi PWM (Hz)

.



Gambar 4.10 Diagram blok rangkaian pembangkit sinyal PWM pada FPGA

Pengaturan nilai *duty cycle* didapatkan melalui pengaturan nilai data dari input. Untuk mendapatkan nilai duty cycle yang dinginkan, maka data pada input adalah sama dengan jumlah clock dalam 1 periode. Berubahnya kondisi dari low ke high adalah sama dengan nilai jumlah clock tertentu dalam 1 periode. Misalkan duty cycle 25%, perubahan kondisi low ke high pulsa terjadi pada jumlah clock = 64 dari total jumlah clock yaitu 256. Dalam kondisi tersebut, maka dapat dirancang suatu blok diagram dan flowchart rangkaian PWM yang ditunjukkan dalam Gambar 4.10 dan 4.11.



Gambar 4.11 flowchart rangkaian pembangkit sinyal PWM

Dalam flowchar rangkaian pembangkit sinyal PWM data INPUT (A) dan data counter (B) dikomparasi dengan logika jika A=B maka data input flip-flop RS=”10”, jika tidak maka data input flip-flop RS=”00”. Selanjutnya jika tepi naik

pada MSB output counter maka data input flip-flop RS=”01” ”, jika tidak maka data input flip-flop RS=”00”, proses selesai.

4.4.3 Perancangan rangkaian penerima pulsa rotary encoder

Dalam perancangan rangkaian penerima pulsa rotary encoder, data input rangkaian didapat dari pulsa *high* dan *low* sensor rotary encoder. Sensor rotary encoder yang digunakan adalah tipe E40S6 dengan jumlah pulsa 360 dalam 1 kali putaran. Untuk mengetahui jumlah pulsa yang dibaca rangkaian dalam per waktu yang ditentukan, maka digunakan rangkaian logika counter dengan pulsa rotary encoder sebagai pemicu counter. Jumlah pulsa rotary per time ms menggunakan reset dari counter. Hasil dari proses sampling didapatkan kecepatan sampling (V) yang ditunjukkan dalam Persamaan 4.4. Proses sampling pulsa rotary ditunjukkan dalam Gambar 4.12.

$$V = \frac{n}{time} \tag{4.4}$$

Misal untuk 60 RPM, berarti selama 1 menit atau 60 sekon terdapat 60 putaran. 1 sekon berarti 1 putaran. Dari Persamaan 4.4 dapat dicari persamaan untuk mencari RPM dari alat yang akan dirancang, yaitu ditunjukkan pada persamaan:

$$RPS = \frac{n}{m} \times \frac{1}{time} \tag{4.5}$$

$$RPM = \frac{n}{m} \times \frac{1}{time} x\ 60 \tag{4.6}$$

Keterangan:

V = kecepatan sampling (n/s)

RPM = Revolutions Per Minute

RPS = Revolutions Per Second

n = jumlah pulsa rotary encoder

m = Jumlah pulsa rotary encoder E40S6 tiap 1 putaran

time = waktu sampling (s)

Time/waktu sampling yang digunakan dalam perancangan alat ini adalah 50 ms, sehingga:

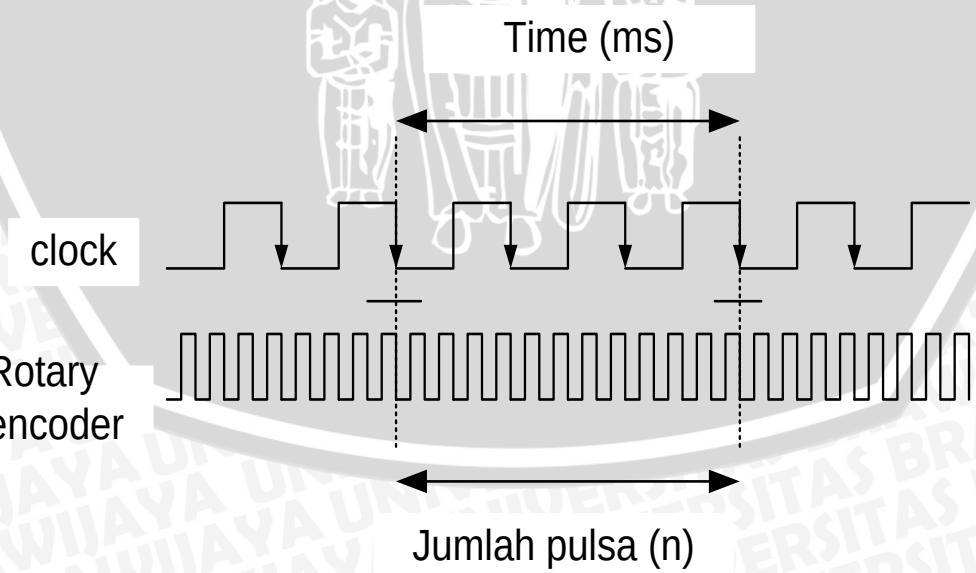
$$\text{RPM} = \frac{n}{360} \times \frac{1}{50\text{x}10^{-3}} \text{ X } 60$$

$$= \frac{n}{6} \times \frac{1}{50\text{x}10^{-3}}$$

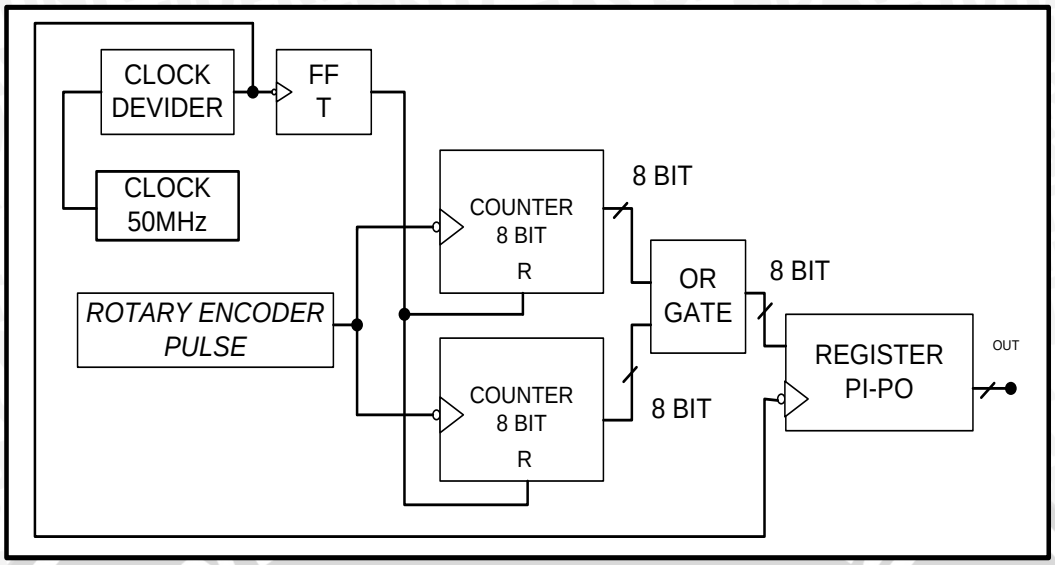
$$= \text{n} \times \frac{10}{3}$$

Dalam sistem FPGA type data yang digunakan ALU adalah UNSIGNED dan INTEGER, sehingga untuk nilai 10/3 = 3,33333 dibulatkan menjadi 3,3 = 33 x 10^{-1}. Nilai 33 adalah integer yang digunakan dengan 10^{-1} sebagai penanda koma digit pertama pada 4 digit output seven segment. Sehingga nilai RPM pada time sampling 50 ms, yaitu :

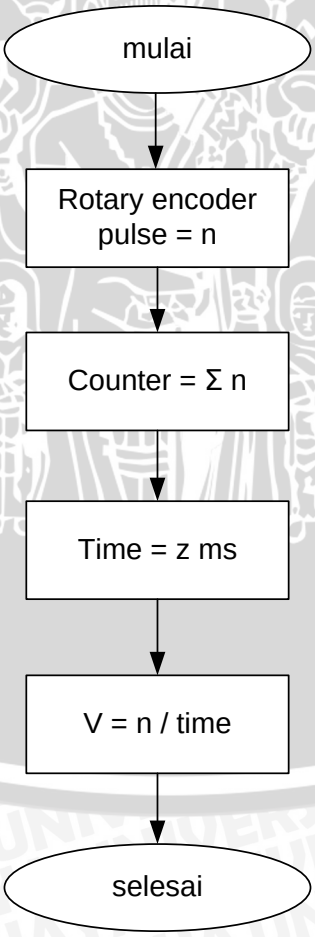
$$\text{RPM} = \text{n x } 33 \text{ x } 10^{-1} \qquad (4.7)$$



Gambar 4.12 timing diagram proses sampling pulsa rotary encoder



Gambar 4.13 Diagram blok rangkaian penerima pulsa rotary encoder pada FPGA



Gambar 4.14 flowchart nilai RPM dari pulsa rotary encoder

Nilai RPM dari pulsa rotary encoder, pulsa dari sensor rotary encoder adalah “n”. Pulsa dari sensor rotary encoder tersebut kemudian dihtung dengan menggunakan counter, counter = Σn. Waktu sampling didefinisikan sebagai z, time = z ms yang kerjanya adalah mereset counter tiap z ms. Selanjutnya data n dan time dimasukkan dalam Persamaan 4.3, proses selesai. Proses tersebut ditunujukkan dalam Gambar 4.14 flowchart nilai RPM dari pulsa rotary encoder.

4.4.4 Perancangan rangkaian data output

Output yang digunakan dalam sistem ini adalah 4 digit seven segment yang terdapat dalam modul FPGA. Data input seven segment adalah berupa kode biner untuk bilangan heksa desimal pada output seven segment. Sedangkan untuk output desimal data input seven segmen adalah berupa bilangan BCD (*Binary Code Decimal*). Untuk mengubah data RPM yang berupa bilangan biner menjadi data BCD, maka diperlukan rangkaian konversi bilangan biner ke bilangan BCD (*Binery to BCD*).

Konversi bilangan biner ke bilangan BCD adalah dengan cara menggunakan algoritma *double dabble*. Misalnya data biner 8 bit adalah “11111111”, maka algoritma untuk nilai bilangan biner tersebut menjadi bilangan BCD adalah ditunjukkan dalam Tabel 4.1.

Algoritma double dabble bertujuan agar data output yang ditampilkan ke seven segment tidak berupa bilangan heksa desimal melainkan bilangan desimal. Hal tersebut agar memudahkan proses pembacaan nilai RPM yang telah diproses oleh FPGA.

Tabel 4.1 menunjukkan bahwa nilai heksa desimal FF akan dikonversi menjadi data desimal, yaitu 256.

Tabel 4.1 Algoritma *double dabble* bilangan biner 8 bit ke bilangan BCD

Operation	Hundreds	Tens	Units	Binary
INPUT				7 4 3 0
HEX				F F
Start				**1 1 1 1 1 1 1 1**
Shift 1				1 1 1 1 1 1 1 1
Shift 2				1 1 1 1 1 1 1 1
Shift 3				1 1 1 1 1 1 1 1
Add 3				1 1 1 1 1 1 1 1
Shift 4				1 1 1 1 1 1 1 1
Add 3				1 1 1 1 1 1 1 1
Shift 5				1 1 1 1 1 1 1 1
Shift 6				1 1 1 1 1 1 1 1
Add 3				1 1 1 1 1 1 1 1
Shift 7				1 1 1 1 1 1 1 1
Add 3				1 1 1 1 1 1 1 1
Shift 8				1 1 1 1 1 1 1 1
BCD	**2**	**5**	**5**	
OUTPUT	9 8	7 4	3 0	
Z	17 16	15 12	11 8	7 4 3 0

Tabel 4.1 menunjukkan tahapan algoritma *double dabble* adalah sebagai berikut:

1). Menggeser bilangan biner ke kiri sebanyak 1 bit.

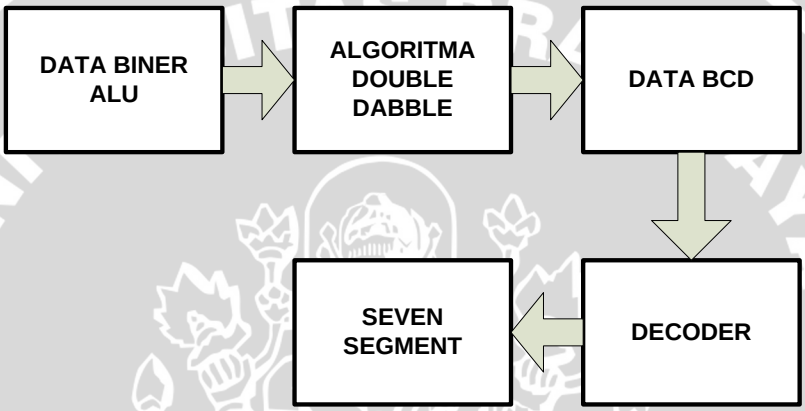
2). Jumlah n-shift adalah sama dengan n-bit input bilangan biner. Hasil dari tabel diatas untuk bilangan BCD adalah pada *shift* ke-8 untuk 8 bit input bilangan biner, yaitu *hundreds*, *tens*, dan *units*.

3). Jika nilai nilai dari kolom BCD adalah sama dengan 5 atau lebih besar, maka nilai pada kolom BCD ditambah 3.

4). Kembali ke proses awal.

Tabel 4.1 menunjukkan input bilangan biner adalah 8 bit yang menghasilkan bilangan BCD ratusan, puluhan, dan satuan. Nilai dari n-bit input bilangan biner tersebut bisa diubah menjadi lebih dari 8 bit dengan tetap menggunakan algoritma yang sama.

Data hasil konversi dari biner ke BCD tersebut menjadi data input untuk decoder seven segment. Gambar 4.14 menunjukkan blok diagram rangkaian data output untuk seven segment.



Gambar 4.15 Diagram blok data output ke seven segment

Data yang ditampilkan pada output seven segment adalah data RPM. Untuk mengetahui nilai g-force dari hasil konversi RPM ke g-force menggunakan Persamaan 4.8.

$$\text{RCF} = \frac{F_c}{F_g}$$

$$\text{RCF} = \frac{\text{m} \,.\, (\omega)^{2} \,.\, \text{R}}{\text{m} \,.\, \text{g}}$$

$$\text{RCF} = \frac{(\omega)^{2} \,.\, \text{R}}{\text{g}}$$

$$\text{RCF} = \frac{(2\,\Pi\,\text{N})^{2} \,.\, \text{R}}{3600.9{,}8.100}$$

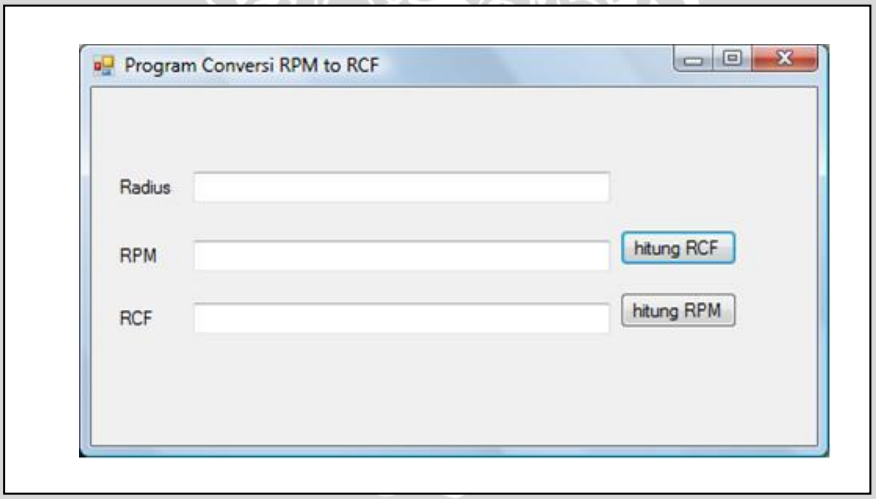
$$RCF = \frac{4\,\Pi^2 (N)^{\,2}\,.\,R}{3600.9{,}8.100}$$

$$RCF = 1{,}118\,.\,10^{-5}\,.\,R\,(N)^2 \qquad (4.8)$$

Keterangan:

- ω = Kecepatan sudut (rad/s)
- R = jari-jari benda ke pusat rotor
- N = putaran per menit (RPM)
- RCF = gaya sentrifugal relatif (x "g")

Nilai konversi tersebut menggunakan program yang dirancang dari Visual Basic seperti yang terlihat dalam Gambar 4.15. Dengan memasukkan nilai RPM dari data seven segment FPGA, maka didapat nilai g-force/RCF.



Gambar 4.16 program konversi data RPM ke RCF/g-force

BAB V
PENGUJIAN DAN ANALISIS

Pengujian dan analisis dilakukan untuk mengetahui apakah sistem telah bekerja sesuai perancangan. Pengujian dilakukan perblok kemudian secara keseluruhan. Adapun pengujian yang perlu dilakukan sebagai berikut:

1). Pengujian Rangkaian Catu Daya
2). Pengujian Sensor Rotary Encoder
3). Pengujian Input Output
4). Pengujian PWM
5). Pengujian *feedback controller*
6). Pengujian ALU
7). Pengujian secara keseluruhan

Instrumen penunjang yang digunakan dalam melakukan pengujian–pengujian di atas antara lain, osiloskop TEKTRONIX TDS-1012B, multimeter, Tachometer digital dan *logic analyzer* ELAB-080.

5.1 Pengujian Catu Daya

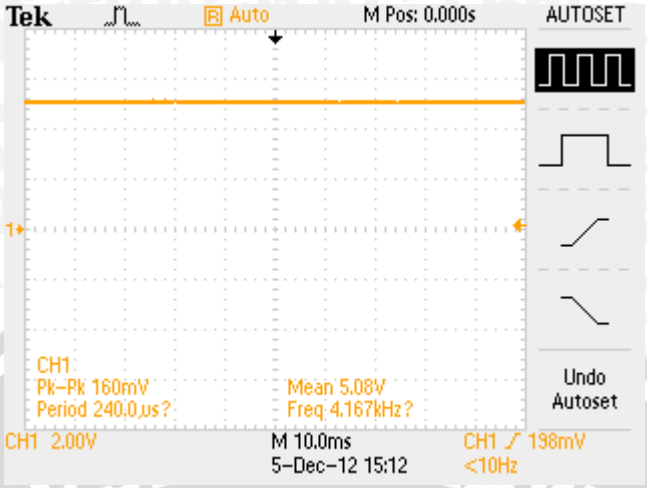
Sebelum melakukan pengujian pada rangkaian keseluruhan pada sistem alat, maka perlu dilakukan pengujian terhadap rangkaian catu daya. Pengujian ini bertujuan untuk mengetahui kesesuaian keluaran pada catu daya. Pemeriksaan dilakukan dengan menghubungkan keluaran rangkaian catu daya dengan osciloscop TEKTRONIX TDS-1012B untuk diketahui nilai tegangannya. Gambar 5.1 menunjukkan proses pengujian rangkaian catu daya.

Pada pengujian catu daya, sistem alat menggunakan 2 catu daya, yaitu catu daya dari *accu* dengan tegangan 12V 2 buah yang disusun secara seri dan output tegangan dari regulator modul FPGA dengan tegangan 5V.

Dari hasil pengujian diperoleh nilai tegangan keluaran rangkaian catu daya 24V sebesar 25,26V. Nilai tersebut merupakan nilai tegangan keluaran maksimal (tanpa beban) yang mampu dihasilkan oleh rangkaian catu daya *accu*. Nilai tegangan tersebut dapat digunakan sebagai catu daya bagi motor DC. Gambar 5.1 menunjukkan tegangan masukan dan keluaran pada rangkaian catu daya 25V dengan menggunakan multimeter.

Gambar 5.1 Pengujian tegangan masukan dan keluaran pada rangkaian catu daya 25V dengan menggunakan multimeter

Pada pengujian kedua catu masukan berupa output regulator dari modul FPGA 5V. Dari hasil pengujian selanjutnya diperoleh nilai tegangan keluaran rangkaian catu daya 5V sebesar 5V. Nilai tersebut merupakan nilai tegangan keluaran maksimal (tanpa beban) yang mampu dihasilkan oleh rangkaian catu daya. Nilai tegangan tersebut dapat digunakan sebagai catu daya bagi sensor rotary encoder E40S6. Gambar 5.2 menunjukkan tegangan pada rangkaian catu daya 5V menggunakan alat uji Osciloscop TDS-1012B.



Gambar 5.2 PengujianTegangan Pada Rangkaian Catu Daya 5V Menggunakan alat uji Osciloscop TDS-1012B

5.2 Pengujian Input/Output

Pengujian input/output terdapat dalam modul FPGA. Pengujian ini bertujuan untuk mengetahui proses dan kerja unit I/O FPGA berjalan dengan baik. Pengujian input berupa pengujian 8 buah switch dan 4 buat push button yang terdapat dalam modul FPGA. Pengujian output berupa 4 digit seven segment dan 8 bit output LED. Pengujian PIN yang dapat difungsikan sebagai input atau output.

Pengujian input switch dan push button pada FPGA dilakukan dengan memprogram dan menetapkan PIN input dalam file UCF sesuai dengan datasheet FPGA Nexys 2 spartan 3e, yaitu:

NET "switch<0>" LOC = "G18";
NET "switch<1>" LOC = "H18";
NET "switch<2>" LOC = "K18";
NET "switch<3>" LOC = "K17";
NET "switch<4>" LOC = "L14";
NET "switch<5>" LOC = "L13";
NET "switch<6>" LOC = "N17";
NET "switch<7>" LOC = "R17";

```
NET "push_button<0>" LOC = "B18";
NET "push_button<1>" LOC = "D18";
NET "push_button<2>" LOC = "E18";
NET "push_button<3>" LOC = "H13";
```

Pengujian output seven segment pada FPGA dilakukan dengan memprogram dan menetapkan PIN input/output dalam file UCF sesuai dengan datasheet FPGA Nexys 2 spartan 3e, yaitu:

```
NET "seg_a<0>" LOC = "L18";
NET "seg_b<1>" LOC = "F18";
NET "seg_c<2>" LOC = "D17";
NET "seg_d<3>" LOC = "D16";
NET "seg_e<4>" LOC = "G14";
NET "seg_f<5>" LOC = "J17";
NET "seg_g<6>" LOC = "H14";
NET "depth_point"     LOC = "C17";

NET "anode<0>" LOC = "F17";
NET "anode<1>" LOC = "H17";
NET "anode<2>" LOC = "C18";
NET "anode<3>" LOC = "F15";

NET "Led<0>"  LOC = "J14";
NET "Led<1>"  LOC = "J15";
NET "Led<2>"  LOC = "K15";
NET "Led<3>"  LOC = "K14";
NET "Led<4>"  LOC = "E17";
NET "Led<5>"  LOC = "P15";
NET "Led<6>"  LOC = "F4";
NET "Led<7>"  LOC = "R4";
```

Pada pengujian I/O FPGA, data input pada switch adalah 8 bit biner diproses dan dikeluarkan di output seven segment dengan data input data hasil output seven segment ditunjukkan dalam Tabel 5.1 dan Gambar 5.3.

Tabel 5.1 Pengujian Input Output Pada FPGA

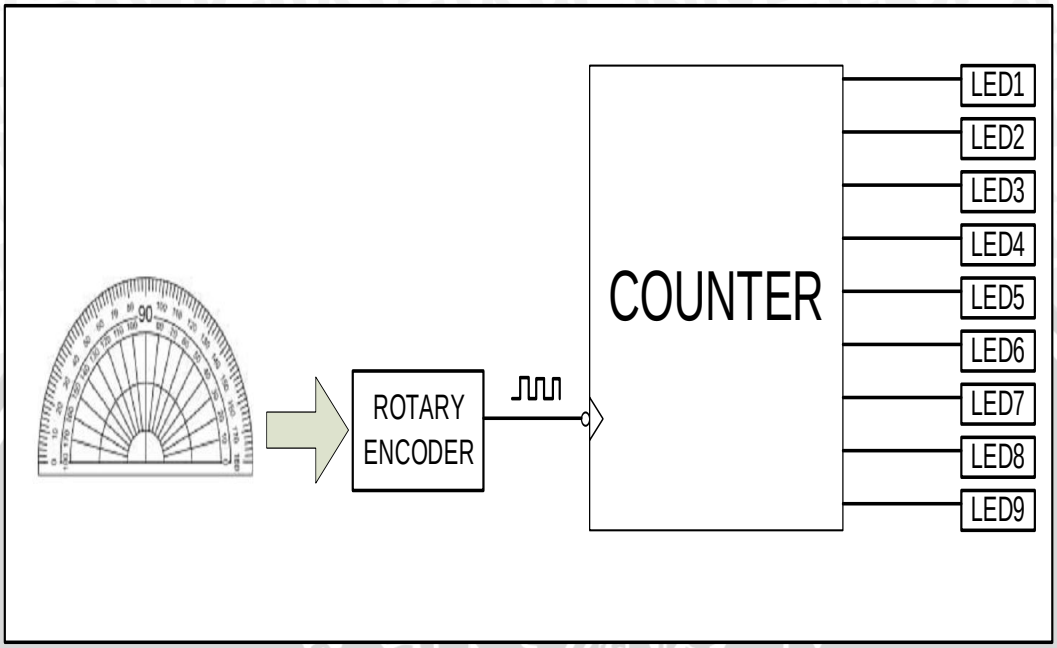
Data Input 8 bit Switch	Data Output 4 digit seven segment	Data Output LED 8 bit	Desimal
0000 0000	0000	0000 0000	0
0000 0001	0001	0000 0001	1
0000 0011	0003	0000 0011	3
0000 0111	0007	0000 0111	7
0000 1111	000F	0000 1111	15
0001 1111	001F	0001 1111	31
0011 1111	003F	0011 1111	63
0111 1111	007F	0111 1111	127
1111 1111	00FF	1111 1111	255

Gambar 5.3 Pengujian input/output pada modul FPGA

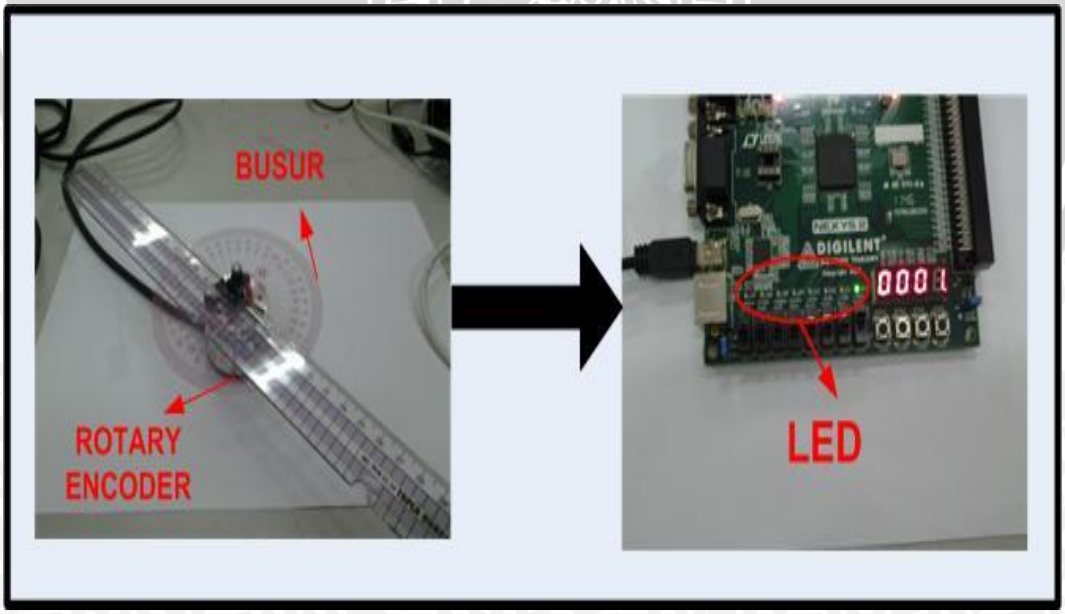
5.3 Pengujian Sensor Rotary Encoder

Pada pengujian sensor rotary encoder, untuk mengetahui jumlah pulsa yang dihasilkan oleh sensor rotary encoder menggunakan counter 9 bit output

dengan pulsa rotary encoder sebagai pemicu tepi turun dari counter. Gambar 5.4 dan 5.5 menunjukkan rangkaian pengujian rotary encoder menggunakan counter 9 bit dengan output LED.



Gambar 5.4 Rangkaian Pengujian Rotary Encoder Menggunakan Counter 9 bit Dengan Output LED



Gambar 5.5 Rangkaian Pengujian Rotary Encoder Menggunakan Counter 9 bit pada program FPGA Dengan Output LED

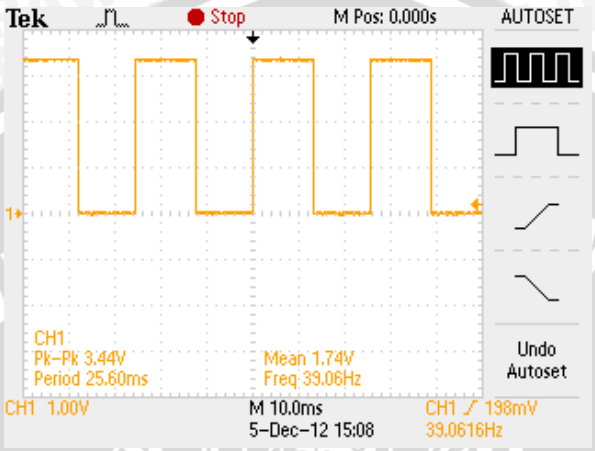
Tabel 5.2 Data Hasil Pengujian Sensor Rotary Encoder

Sudut (°)	Data Output LED 8 bit (biner)	Desimal	Kesalahan (dalam %)
20	000010101	21	0,05
40	000101000	40	0
60	000111101	61	0,016
80	001010010	82	0,025
100	001100100	100	0
120	001111010	122	0,016
140	010001110	142	0,014
160	010100001	161	0,006
180	010110110	182	0,011
200	011001011	203	0,015
220	011011101	221	0,004
240	011110011	243	0,012
260	100000101	261	0,003
280	100011000	280	0
300	100101011	299	0,003
320	101000000	320	0
340	101010101	341	0,002
360	101101010	362	0,005
Kesalahan Rata-Rata			0,010

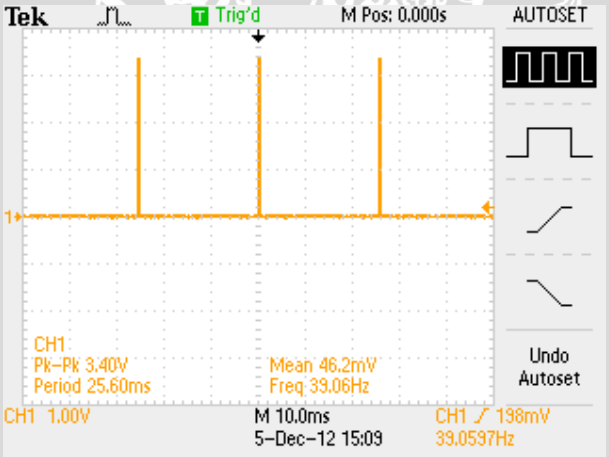
Tabel 5.2 menunjukkan data output biner dihasilkan oleh LED FPGA. Saat rotary encoder diputar dengan sudut tertentu, maka output biner dapat diamati melalui output LED, kemudian diubah ke bilangan desimal untuk mengetahui besarnya prosenstase kesalahan. Prosesntase kesalahan tersebut terjadi karena terdapat kesalahan pengamatan saat memutar rotary encoder pada sudut yang ditetapkan. Kesalahan rata-rata mencapai 0,010%.

5.4 Pengujian PWM

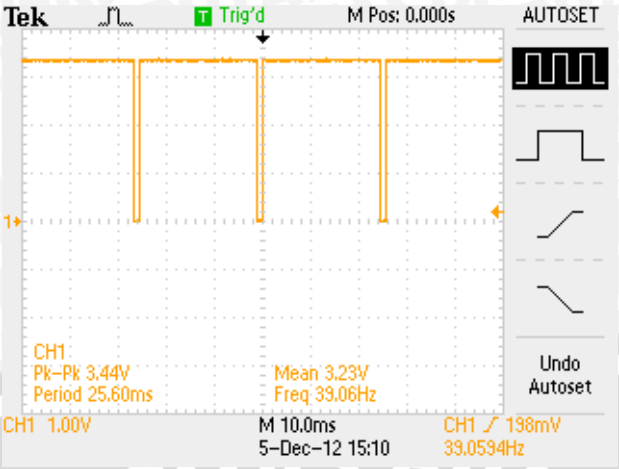
Pengujian PWM bertujuan untuk mengetahui bentuk pulsa periodik yang dihasilkan FPGA, khususnya untuk mengetahui besarnya lebar pulsa high dalam 1 periode. Sinyal PWM yang dibangkitkan oleh FPGA diamati melalui Logic Analyzer E-Lab 080 yang ditunjukkan dalam Gambar 5.9.



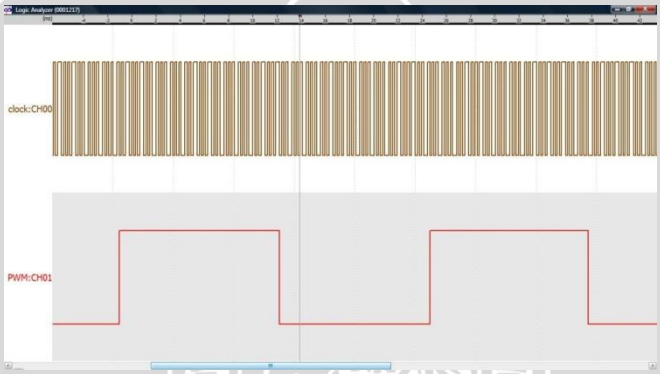
Gambar 5.6 Pengujian sinyal PWM FPGA dengan Input biner "10000000" pada osciloscop TDS-1012B



Gambar 5.7 Pengujian sinyal PWM FPGA dengan Input biner "00000001" pada osciloscop TDS-1012B



Gambar 5.8 Pengujian sinyal PWM FPGA dengan Input biner "11110000" pada osciloscop TDS-1012B



Gambar 5.9 Pengujian sinyal PWM FPGA dengan Input biner "10000000" pada Logic Analyzer Elab-080

Gambar 5.6. Gambar 5.7 dan Gambar 5.8 menunjukkan perubahan lebar logika *high* pada sinyal PWM yang dihasilkan oleh FPGA yang diamati melalui osciloscop TDS-1012B Dengan input 8 bit yang diubah-ubah lebar pulsa *high* pada PWM juga ikut berubah, namun lebar 1 periode tetap/tidak terjadi perubahan. Hal ini menunjukkan terdapat perhitungan *duty cycle* yang berhubungan dengan kombinasi n-bit input. Nilai *duty cycle* juga berpengaruh terhadap tegangan referensi 3,291 volt yang dihasilkan oleh FPGA yang dapat diamati dalam Tabel 5.3. Nilai *duty cycle* dapat diperoleh dalam Persamaan 5.1. Sedangkan nilai tegangan rata-rata yang dihasilkan oleh PWM FPGA diperoleh dalam Persamaan 5.2.

$$duty\ cycle = \frac{t_1}{t_2} \times 100\% \quad (5.1)$$

$$V = duty\ cycle \times Vref \quad (5.2)$$

Keterangan:

- t_1 = Lebar pulsa *high* dalam 1 periode
- t_2 = Lebar pulsa dalam 1 periode
- Vref = Tegangan referensi FPGA
- V = Tegangan rata-rata PWM

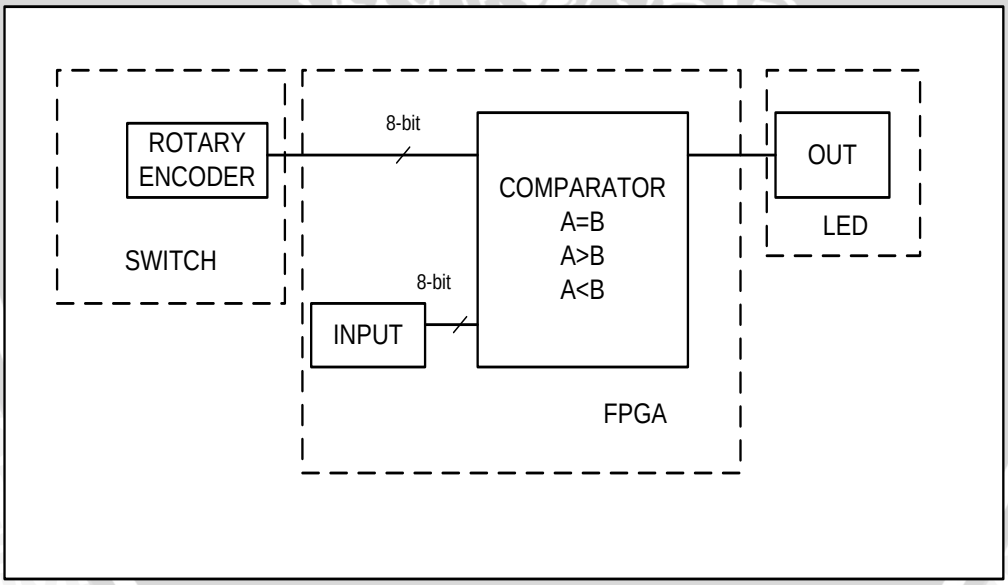
Tabel 5.3 Pengujian sinyal PWM FPGA dalam volt

Logika biner INPUT	INPUT (dalam decimal)	*Duty cycle* (dalam %)	Sinyal PWM pada Pengujian (dalam volt)	Sinyal PWM Teori (dalam volt)	Kesalahan (dalam%)
11111111	255	100,0	3,291	3,291	0
11101111	239	93,7	3,084	3,085	0,032
11011111	223	87,5	2,879	2,880	0,034
11001111	207	81,2	2,673	2,674	0,037
10111111	191	75,0	2,467	2,468	0,040
10101111	175	68,7	2,261	2,263	0.088
10011111	159	62,5	2,055	2,057	0,097
10001111	143	56,2	1,849	1,851	0,108
01111111	127	50,0	1,643	1,645	0,121
01101111	111	43,7	1,438	1,440	0,139
01011111	95	37,5	1,232	1,234	0,162
01001111	79	31,2	1,027	1,028	0,097
00111111	63	25,0	0,821	0,823	2,436
00101111	47	18,7	0,616	0,617	1,620
00011111	32	12,5	0,411	0,411	0
00001111	15	6,2	0,206	0,206	0
00000000	0	0	0	0	0
Kesalahan Rata-Rata					0,295

Berdasarkan Tabel 5.3 diperoleh perhitungan sinyal PWM dalam volt yang diinginkan dan data hasil percobaan. Dari Tabel 5.3 dapat disimpulkan bahwa tingkat kesalahan semakin besar apabila nilai input semakin kecil. Kesalahan rata-rata dapat diperoleh sebesar 0,295%.

5.5 Pengujian Feedback Controller

Rangkaian *feedback controller* merupakan perbandingan antara data hasil dari rangkaian penerima sinyal rotary encoder dengan data input. Pada Pengujian ini, diamati hasil perbandingan 2 data tersebut. Untuk memudahkan dalam pengujian, maka data dari rotary encoder digunakan sebagai switch input dalam modul FPGA, sedangkan data INPUT yang dibandingkan ditulis dalam program FPGA. Gambar 5.9 menunjukkan rangkaian pengujian *feedback controller* pada FPGA.



Gambar 5.10 Rangkaian Pengujian *Feedback Controller*

Prosedur pengujian rangkaian *feedback controller* adalah dengan menetapkan data input waktu FPGA deprogram dan data rotary encoder yang dapat diubah-ubah melalui switch. Hasil output diamati melalui LED pada modul FPGA.

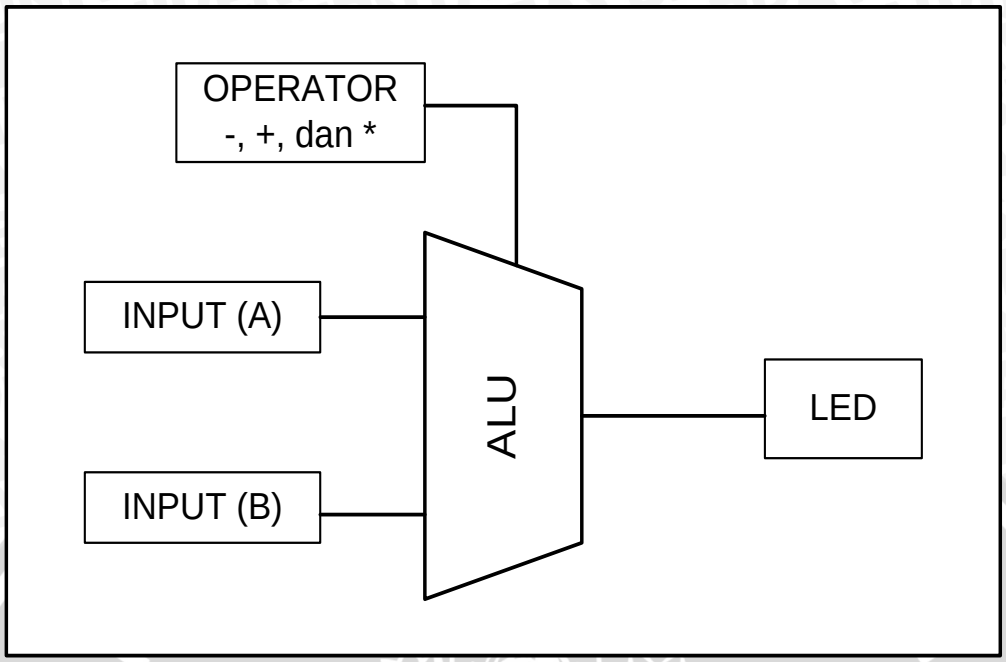
Tabel 5.4 Data Hasil Pengujian Rangkaian *Feedback Controller*

Rotary encoder (A)	INPUT (B)	Komparasi Kondisi Logika input	Output LED
00000111	00001111	A<B	Terang
00001111	00001111	A=B	Redup
00011111	00001111	A>B	Mati

Tabel 5.4 menunjukkan data INPUT (B) ditetapkan adalah “00001111” atau dalam desimal=15. Data rotary encoder (A) menggunakan switch yang diubah-ubah untuk dibandingkan dengan data INPUT (B). Saat data rotary encoder kurang dari data INPUT, maka kondisi LED adalah terang. Hal ini bertujuan agar kondisi output tersebut dapat menggerakkan motor lebih cepat agar mencapai data yang diinginkan, sebaliknya jika data Rotary encoder lebih besar dari data INPUT, maka kondisi LED akan mati. Hal ini bertujuan agar motor mengurangi kecepatan lebih lambat agar mencapai data yang diinginkan. Kondisi LED redup saat A=B menandakan sinyal PWM yang dihasilkan FPGA dan kondisi output sesuai dengan data yang diinginkan.

5.6 Pengujian ALU

Pengujian ALU (Arithmatika Logic Unit) pada FPGA bertujuan untuk mengetahui hasil penggunaan operator kurang (-), tambah (+) dan kali (*). Pada pengujian ini prosedur yang dilakukan adalah menyiapkan program pada FPGA dan menggunakan *switch* dan LED sebagai indikator hasil dalam modul FPGA. Gambar 5.10 menunjukkan rangkaian Pengujian ALU pada FPGA.



Gambar 5.11 Rangkaian Pengujian ALU pada FPGA

Tabel 5.5 Pengujian ALU pada FPGA dengan operator Tambah (+)

INPUT (A)	INPUT (B)	A+B	Dalam bentuk desimal
0000	0000	0000	0+0=0
0001	0000	0001	1+0=1
0001	0001	0010	1+1=2
0011	0011	0100	3+3=6
0011	0111	1010	3+7=10
0111	0111	1110	7+7=14
1111	1111	10000	15+15=30

Tabel 5.6 Pengujian ALU pada FPGA dengan operator Tambah (+)

INPUT (A)	INPUT (B)	A-B	Dalam bentuk desimal
0000	0000	0000	0-0=0
0001	0000	0001	1-0=1
0001	0001	0000	1-1=0
0100	0011	0001	4-3=1
1010	0111	0011	10-7=3
1100	0111	0101	12-7=5
1111	1110	0001	15-14=1

Tabel 5.7 Pengujian ALU pada FPGA dengan operator kali (*)

INPUT (A)	INPUT (B)	A*B	Dalam bentuk desimal
0000	0000	00000000	0*0=0
0001	0000	00000000	1*0=0
0001	0001	00000001	1*1=1
0011	0011	00001001	3*3=9
0011	0111	00010101	3*7=21
0111	0111	00110001	7*7=49
1111	1111	11100001	15*15=225

Tabel 5.5 Tabel 5.6 dan Tabel 5.7 menunjukkan proses aritmatika 2 input 4 bit biner dengan operator kurang (-), tambah (+) dan kali (*). Dari hasil pengujian dapat disimpulkan bahwa hasil dari ALU yang digunakan dalam FPGA bekerja dengan baik dan sesuai dengan data yang diinginkan.

5.7 Pengujian Secara Keseluruhan

Pengujian RPM bertujuan untuk membandingkan data RPM yang diproses oleh FPGA dengan data RPM yang diperoleh dari alat ukur lain, yaitu Tachometer. Proses ini menjadi pengujian secara keseluruhan karena seluruh sistem dari masing-masing komponen secara aktif bekerja untuk menghasilkan output akhir yang dinginkan, yaitu nilai RPM.

Tujuan dari pengujian ini adalah untuk mengkalibrasi dan membandingkan data yang dihasilkan oleh FPGA berupa RPM dengan alat ukur lain berupa Tachometer Digital.

Prosedur yang dilakukan pada pengujian ini adalah menyiapkan alat-alat yang digunakan untuk menjalankan program FPGA dan alat ukur tachometer. Diambil beberapa data dengan input data biner yang berbeda-beda. Dari pembacaan RPM di output FPGA dengan pembacaan RPM dari alat ukur Tachometer dapat dicari besarnya error atau kesalahan.

Hasil yang tercatat adalah berupa data RPM dari FPGA dengan alat ukur Tachometer, dan besarnya error. Tabel 5.8 menunjukkan data hasil pengujian RPM dalam proses FPGA dengan menggunakan alat ukur Tachometer.

Tabel 5.8 Data Hasil Pengujian RPM dalam proses FPGA dengan menggunakan alat ukur Tachometer

DATA INPUT	PEMBACAAN RPM FPGA (TEORI)	PEMBACAAN RPM TACHOMETER (PRAKTIK)	KESALAHAN (%)
00010011	66	65,1	1,36
00010111	75,9	78,1	2,90
00011111	102,3	104,2	1,86
00100011	115,5	117,2	1,73
00100111	128,7	130,3	1,24
00101111	155,1	156,5	0,90
00110111	181,2	182,3	0,61
00111011	194,7	195,6	0,46
00111110	204,6	205	0,19
10000111	231,7	231,5	0,08
RATA-RATA KESALAHAN			1,13

Tabel 5.8 menunjukkan hasil pengujian pengukuran RPM dengan menggunakan Tachometer digital. Saat nilai input diperbesar, maka nilai RPM yang terbaca juga mengalami kenaikan, hal ini dikarenakan nilai PWM = nilai data biner pada input. Rata-rata error yang didapat dalam pengujian ini adalah 1,13%.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Dari Hasil perancangan dan pengujian yang telah dilakukan, dapat disimpulkan beberapa hal sebagai berikut .

1). Hasil pengujian sinyal PWM yang dihasilkan FPGA secara praktek dan teori memiliki kesalahan rata-rata 0,295%. Hal ini tidak terlalu mempengaruhi kinerja dari sistem alat yang dirancang.

2). Pengujian secara kesuluruhan untuk membandingkan nilai RPM yang dihasilkan oleh sistem dengan alat ukur lain diperoleh kesalahan rata-rata sebesar 1,13%. Hal ini menunjukkan nilai RPM yang dihasilkan sistem sesuai dengan yang diinginkan.

3). Hasil output dari perancangan alat uji fungsional *g-force* sudah sesuai dengan yang diinginkan. Karena menurut aturan KOMURINDO dalam pengujian *g-force* muatan roket akan diberikan gaya g sebesar 6g atau sebesar 190 RPM dalam radius 15 cm. Sedangkan hasil dari sistem yang dirancang dapat memberikan gaya g sebesar 9g atau sebesar 231 RPM dalam radius 15 cm.

6.2 Saran

Beberapa hal yang direkomendasikan untuk pengembangan alat uji Muatan roket bagian fungsional G-force lebih lanjut adalah**:**

1). Disarankan untuk membuat desain mekanik alat yang lebih presisi agar kerja motor DC lebih optimal dan sesuai dengan yang diinginkan.

2). Mengembangkan sistem program pada FPGA atau kontroler utama yang lebih kompleks dan sensor yang lebih banyak, karena keunggulan FPGA yang dapat memproses suatu sistem secara paralel

DAFTAR PUSTAKA

Areny, P.Ramon. 1991. *Sensor and Signal Conditioning*. John Wiley & Sons Inc. Canada.

Halliday, david. 1978. *Physics, 3rdedtion*. John Wiley & Sons Inc. Canada.

Digilent. 2011. *Digilent nexys2 Board Reference Manual.* http://www.digilentinc.com/Products

Lister, Eugene.C. 1984. *Elektric Circuits and Machines Sixth Edition*. McGraw-Hill Inc. London.

Maxfield, C. 2009. *FPGAs World Class Designs.* Elsevier. Oxford.

Meyer, U and Baese. 2007. *Digital Signal Processing with Field Programmable Gate Array*. Springer Berlin Heidelberg. New York.

Smith, R.S. 2010. *FPGAs 101-Everything you need to know to get started*. Elseive Inc. USA.

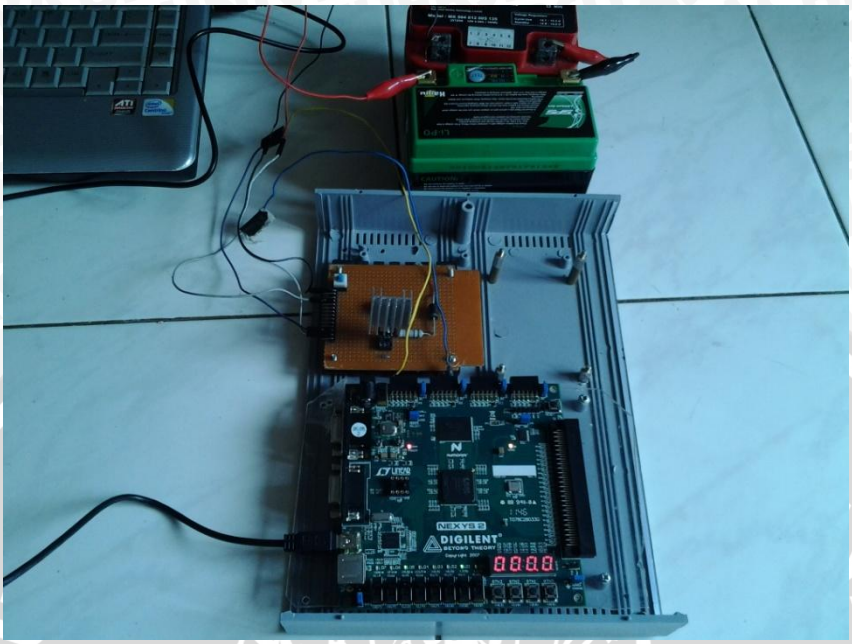
LAMPIRAN

LAMPIRAN 1

FOTO ALAT

Gambar 1. Mekanik Alat Tampak Samping

Gambar 2. Mekanik Alat Tampak Prespektif



Gambar 3. Sistem Elektrik Catu Daya 24V, Driver Motor, dan FPGA

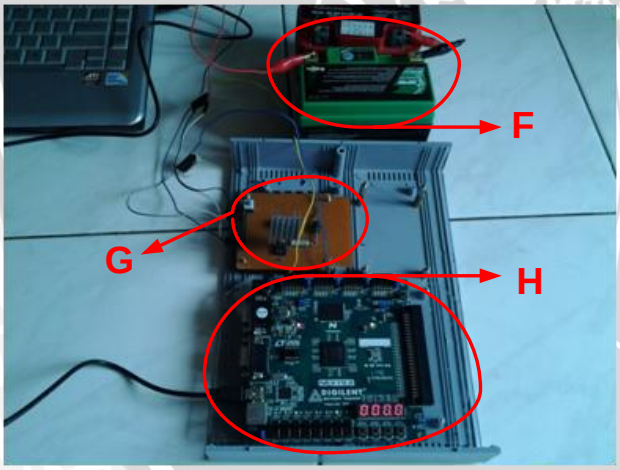
LAMPIRAN 2

PENJELASAN BAGIAN-BAGIAN ALAT

Gambar 4. Penjelasan Bagian-bagian Mekanik Alat

KETERANGAN:

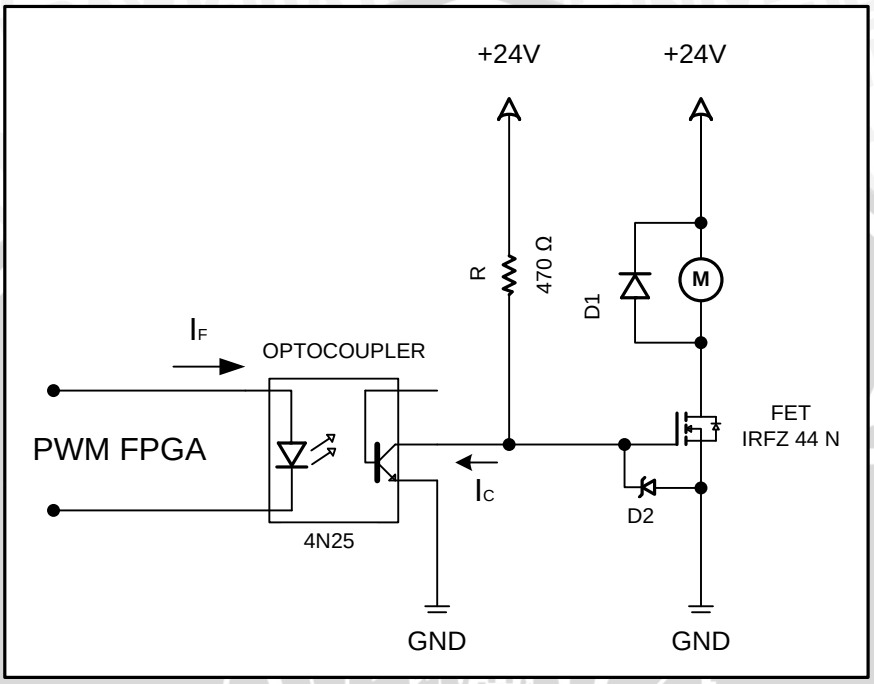
A. Tabung *Payload* Roket
B. Penyeimbang Lengan Putar
C. Belt dan Gear 5:1
D. Motor DC 24V
E. Rotary Encoder
F. 2 Buah *Accu* 12V
G. Elektrik Driver Motor
H. Modul FPGA Nexys2



Gambar 5. Penjelasan Bagian-bagian Elektrik Alat

LAMPIRAN 3

SKEMA RANGKAIAN ELEKTRIK



Gambar 6. Skema Elektrik Driver Motor

LAMPIRAN 4

LISTING PROGRAM

"PROGRAM VHDL XILINX ISE 14.1"

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_ARITH.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5   use ieee.numeric_std.ALL;
6
7   entity g_force is
8   Port ( Clk_50MHz :          in std_logic;
9          PD,re,T,en :         inout std_logic; --re=sinyal rotary encoder--
10         Bcd :                inout std_logic_vector (16 downto 0);
11         Dp :                 in  std_logic_vector (3 downto 0);
12         Segm :               out std_logic_vector (1 to 8);
13         An :                 out std_logic_vector (3 downto 0);
14         P :                  inout std_logic_vector (16 downto 0);
15         out_RE,d,tu ,ta:     inout std_logic_vector (7 downto 0);
16         kali1 :               inout integer ;
17         Clk_1kHz, Clk_2kHz  : out std_logic;
18         switch:          inout std_logic_vector ( 7 downto 0);--input 8 bit
19         mux_8bit ,n,o:       inout std_logic_vector ( 7 downto 0);
20         w,r,v,H:             inout std_logic;
```

```
21          selecktor:          inout std_logic_vector ( 2 downto 0);
22          au:                 inout std_logic_vector ( 7 downto 0);
23           b:                 inout std_logic_vector ( 1 downto 0) ;
24          output1 :           inout std_logic_vector ( 7 downto 0);
25          Clk_3kHz ,PWM:      inout std_logic);
26      end g_force;
27      architecture Behavioral of g_force is
28      Signal Clk2:            std_logic;
29      Signal Clk:             std_logic;
30      Signal Bcd_ciffer:      std_logic_vector (3 downto 0);
31      Signal Decimal_Punkt: std_logic;
32      Signal Segmenter:       std_logic_vector (1 to 7);
33      signal a12 :            std_logic_vector ( 16 downto 0);
34      signal output:          std_logic_vector ( 7 downto 0);
35      signal output2:         std_logic_vector ( 7 downto 0);
36      signal Out_OR :         std_logic_vector ( 7 downto 0 ) ;
37      signal Reg_PIPO :       std_logic_vector ( 7 downto 0 ) ;
38      signal kali :           integer ;
39      Signal Clk1:            std_logic;
40      Signal a14:             std_logic_vector(7 downto 0) ;
41      signal S :              std_logic_vector (1 downto 0);
42
43      -- Fungsi untuk membuat conversi Boolean ke STD_logic
44      function Conv_boolean (B:boolean) return std_logic is
45        Variable Return_data: std_logic :=  '0';
```

```
46      begin
47        if B then Return_data :=  '1'; end if;
48              return Return_data;
50      end Conv_boolean;
51      ------------------------------------------------------
52      begin
53      Clk_1kHz <= Clk2;
54      Clk_2kHz <= Clk;
55      p <= a12;
56      bcd (16 downto 0) <= a12;
57      tu<= output;
58      ta<= output;
59      out_RE <= out_OR;
60      d<=Reg_pipo;
61      kali1<=kali;
62      ------------------------------------------------
63      -- proses perhitungan nilai RPM--
64      kali <= conv_integer  (signed (Reg_PIPO))*33;
65      au<=Reg_PIPO;
66      ------------------------------------------------
67      -- Program Rangkaian Feedback Controller
68      ------------------------------------------------
69      process (switch,au)
70      begin
71       if (switch <au)  then
```

```
72              w <= '1';
 73             else  w <= '0';
74      end if;
75      end process;
76      process ( switch,au )
77      begin
78       if (switch >au)  then
79                r <= '1';
80       else  r <= '0';
81
82      end if;
83      end process;
84      process ( switch,au )
85      begin
86       if (switch =au)  then
89              v <= '1';
90      else  v <= '0';
91      end if;
92      end process;
93
94      selecktor <= w & r & v ; -- w , r , v = variable hasil komparator --
95
96      process (selecktor,mux_8bit,n,o,switch)
97      begin
98      case selecktor is
```

```
99      when "100" => mux_8bit<=n;
100     when "010" => mux_8bit<=o;
101     when others => mux_8bit<=switch; -- hasil output multiplekser =i masukan PWM --
102     end case;
103     end process;
104
105     n<=X"FF";
106     o<=X"01";
107
108
109
110
111     ------------------------------------------------
112     --Rangkaian Rangkaian Pembangkit Sinyal PWM
113     ------------------------------------------------
114     Clk_3kHz  <= Clk1;
115     output1  <=  a14;
116     H <= a14 (7);
117     b  <= S;
118     process ( Clk_50MHz) -- membuat delay agar kecepatan clock turun
119     variable Q1: integer range 0 to 50000; -- perulangan, 50M/50k = 5k Hz
120     begin
121       if rising_edge ( Clk_50MHz) then
122               if Q1<5000 then
```

```
123                     Q1 := Q1 + 1;
124                         Clk1 <= '0';
125                 else
126                     Q1 := 1;
127                         Clk1 <= '1';
128                 end if;
129     end if;
130     end process;
131
132     output_a:
133     process (clk1) -- program counter 8 bit
134     variable count : unsigned (7 downto 0);
135     begin
136     if rising_edge (clk1) then
137     count := count +1  ;
138     end if;
139     a14 <= std_logic_vector (count);
140     end process;
141
142     output_1:
143     process ( output1, mux_8bit,h)
144     begin
145     if (output1 = not mux_8bit) then
146      S<= "10";
147     elsif falling_edge (h) then
```

```
148       S<="01";
149     end if;
150     end process;
151
152     process (S)
153     begin
154     case S is
155               when "10"=> PWM <='1';
156               when "01"=> PWM <='0';
157     when others => PWM <='1';
158     end case;
159     end process;
160
161     ------------------------------------------------
162     --Program Rangkaian Penerima Sinyal Rotary Encoder--
163     ------------------------------------------------

164     process( Clk_50MHz)          --program clock divider--
165               variable Q: integer range 0 to 50000; --perulangan 50.000 kali--
166     begin
167       if rising_edge ( Clk_50MHz) then
168                 if Q<50000 then
169                           Q := Q + 1;
170                                 Clk2 <= '0';
171                       else
```

```
172                     Q := 1;
173                         Clk2 <= '1';
174                 end if;
175 end if;
176 end process; -- clock menjadi 50M/50k = 5k HZ--
177
178 process ( Clk_50MHz)          --program clock divider--
179         variable H: integer range 0 to 2500000;
180 begin
181   if rising_edge ( Clk_50MHz) then
182       if H< 2500000 then
183                 H := H + 1;
184                     Clk <= '0';
185                 else
186                   H := 1;
187                       Clk <= '1';
188                 end if;
189 end if;
190 end process;
191 process (clk)
192 begin
193 if falling_edge (clk) then
194    T<= not T ;
195 end if;
196 end process;
```

```
197
198     en <='1';
199     process(re) -- program untuk menghitung sinyal rotary encoder (counter1)
200     variable count : unsigned (7 downto 0);
201     begin
202     if Falling_edge (re) then
203             if T = '0' then
204                     -- Reset counter
205                         count := (others => '0');
206                 elsif en = '1' then
207                         -- Increment the counter if counting is enabled
208                         count := count + 1;
209                 end if;
210     end if;
211     output <= std_logic_vector (count);
212     end process;
213
214     process(re)  -- program untuk menghitung sinyal rotary encoder (counter2)
215     variable count2 : unsigned (7 downto 0);
216     begin
217     if Falling_edge (re) then
218             if T = '1' then
219                     -- Reset the counter to 0
220                     count2 := (others => '0');
221                 elsif en = '1' then
```

```
222                         -- Increment the counter if counting is enabled
223                         count2 := count2 + 1;
224                 end if;
225     end if;
226     output2 <= std_logic_vector (count2);
227     end process;
228
229     Out_OR <= output OR output2; -- Gerbang OR 8 bit--
230
231     process (clk)
232     begin
233     if falling_edge (clk) then
234     Reg_PIPO<=out_OR ;
235     end if;
236     end process;
237
238     ----------------------------------------------------
239     --program algoritma double dabble--
240     ----------------------------------------------------
241
242     bcd1:
243     process (kali)
244     variable z: STD_LOGIC_VECTOR (30 downto 0);
245     begin
246     for i in 0 to 30 loop
```

```
247     z (i) := '0';
248     end loop;
249     z (16 downto 3) := std_logic_vector (to_unsigned (kali,14));
250     for i in 0 to 10 loop
251             if z (17 downto 14) > 4 then
252                     z( 17 downto 14) := z (17 downto 14) + 3;
253             end if;
254             if z (21 downto 18) > 4 then
255                     z (21 downto 18) := z (21 downto 18) + 3;
256             end if;
257             if z (25 downto 22) > 4 then
258                     z (25 downto 22) := z (25 downto 22) + 3;
259             end if;
260             if z (29 downto 26) > 4 then
261                     z (29 downto 26) := z (29 downto 26) + 3;
262             end if;
263
264     z (30 downto 1) := z (29 downto 0);
265     end loop;
266     a12 <= z (30 downto 14);
267     end process bcd1;
268
269     ----------------------------------------------------------------------
270     -- program BCD --
280     ----------------------------------------------------------------------
```

```
281     Multiplekser:
282     process( Clk2)
283       variable S: std_logic_vector ( 1 downto 0);
284     begin
285       if rising_edge (Clk2) then
286         S := S+1;
287                     case S is
288                       when "00" =>
289                             Decimal_Punkt <= Dp (0);
290                                 Bcd_ciffer    <= Bcd ( 3 downto 0);
291                                 An            <= "1110";
292                                 pd <= '1';
293                       when "01" =>
294                             Decimal_Punkt <= Dp (1);
295                                 Bcd_ciffer    <= Bcd ( 7 downto 4);
296                                 An            <= "1101";
297                                  pd <= '0';
298                       when "10" =>
299                             Decimal_Punkt <= Dp (2);
300                                   Bcd_ciffer    <= Bcd (11 downto 8);
301                                 An            <= "1011";
302                                  pd <= '1';
303                       when "11" =>
304                             Decimal_Punkt <= Dp (3);
```

```
305                                 Bcd_ciffer    <= Bcd (15 downto 12);
306                                 An            <= "0111";
307                                 pd <= '1';
308                             when others =>
309                                 null;
310       end case;
311             end if;
312   end process;
313
314   ---------------------------------------------------------------------
315   --Program menampilkan data ke output seven segment --
316   ---------------------------------------------------------------------
317   BCD27segm:
318   process ( Bcd_ciffer)
319     type D16_bit is array ( 15 downto 0) of boolean;
320     variable D: D16_bit;
321           variable Sa,Sb,Sc,Sd,Se,Sf,Sg: std_logic;
322   begin
323           D := (others => false);                 -- Set all D(x) = false;
324   D ( Conv_integer (Bcd_ciffer)) := true;         -- Set D(bcd)   = true;
325   Sa := Conv_Boolean( D( 1) or D( 4) or D(11) or D(13));
326  Sb := Conv_Boolean( D( 5) or D( 6) or D(11) or D(12) or D(14) or D(15));
327   Sc := Conv_Boolean( D( 2) or D(12) or D(14) or D(15));
328   Sd := Conv_Boolean( D( 1) or D( 4) or D( 7) or D(10) or D(15));
329   Se := Conv_Boolean( D( 1) or D( 3) or D( 4) or D( 5) or D( 7) or D( 9));
```

```
330     Sf := Conv_Boolean( D( 1) or D( 2) or D( 3) or D( 7) or D(13));
331     Sg := Conv_Boolean( D( 0) or D( 1) or D( 7) or D(12));
332
333             Segmenter <= Sa & Sb & Sc & Sd & Se & Sf & Sg;
334     end process;
335     --7 Segments and the Decimalpoint joined together
336     Segm <=  Segmenter & Decimal_punkt;
337
338     end Behavioral;
```

“FILE UCF Xilinx ISE 14.1”

```
1       NET "segm<1>" LOC = "L18";
2       NET "segm<3>" LOC = "D17";
3       NET "segm<4>" LOC = "D16";
4       NET "segm<5>" LOC = "G14";
5       NET "segm<6>" LOC = "J17";
6       NET "segm<7>" LOC = "H14";
7
8       NET "an<0>" LOC = "F17";
9       NET "an<1>" LOC = "H17";
10      NET "an<2>" LOC = "C18";
11      NET "an<3>" LOC = "F15";
12
13      NET "Clk_50MHz"   LOC = "B8";
14      NET "pd"    LOC = "C17"; # Ban
```

```
15
16      NET "Re" LOC = "L15";
17      NET "re"  CLOCK_DEDICATED_ROUTE= false; # Bank = 1
18      NET "switch<0>" LOC = "G18";
19      NET "switch<1>" LOC = "H18";
20      NET "switch<2>" LOC = "K18";
21      NET "switch<3>" LOC = "K17";
22      NET "switch<4>" LOC = "L14";
23      NET "switch<5>" LOC = "L13";
24      NET "switch<6>" LOC = "N17";
25      NET "switch<7>" LOC = "R17";
26
27      NET "PWM" LOC = "K13";
```

"LISTING PROGRAM RCF TO RPM CONVERSION DENGAN VISUAL BASIC"

```
Public Class Form1

    Private Sub TextBox1_TextChanged(sender As Object, e As EventArgs) Handles TextBox1.TextChanged

    End Sub

    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click

        Dim radius As Double

        Dim rpm As Double

        Dim rcf As Double
```

```
        radius = Convert.ToDouble(TextBox1.Text)
        rpm = Convert.ToDouble(TextBox2.Text)
        rcf = 1.118 * radius * rpm * rpm * 0.00001
        TextBox3.Text = Convert.ToString(rcf)
    End Sub

    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles
Button2.Click
        Dim radius As Double
        Dim rpm As Double
        Dim rcf As Double

        radius = Convert.ToDouble(TextBox1.Text)
        rcf = Convert.ToDouble(TextBox3.Text)
        rpm = Math.Sqrt((rcf * 100000) / (1.118 * radius))
        TextBox2.Text = Convert.ToString(rpm)
    End Sub
End Class
```

LAMPIRAN 5

DATASHEETS