PERANCANGAN DAN PEMBUATAN KONTROL LOGIKA *FUZZY* UNTUK PENGENDALIAN PUTARAN *ENGINE* BERDASARKAN POSISI *THROTTLE*

SKRIPSI

TEKNIK ELEKTRO KONSENTRASI TEKNIK KONTROL

Ditujukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

GANDA LESMANA

NIM. 115060307111018

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2016

RINGKASAN

Ganda Lesmana, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Januari 2016, *Perancangn dan Pembuatan Kontrol Logika Fuzzy untuk Pengendalian Putaran Engine Berdasarkan Posisi Throttle*, Dosen Pembimbing : Muhammad Aziz Muslim dan Goegoes Dwi Nusantoro.

Perkembangan teknologi suplai bahan bakar pada mesin sepeda motor telah mengalami kemajuan sangat pesat. Kini, sistem karburator telah digantikan dengan sistem injeksi agar perhitungan jumlah bahan bakar yang disemprotkan ke dalam mesin menjadi presisi. Salah satu permasalahan dalam teknologi otomotif khususnya dalam *Engine Control Unit* adalah hal tersebut masih menjadi rahasia dari produsen. Sehingga untuk mengatur putaran *engine* masih cukup sulit karena durasi injektor telah di *setting* oleh produsen.

Penelitian ini dilakukan dengan menggunakan sepeda motor vario 110 FI sebagai plant penelitian dengan Kontrol Logika *Fuzzy* (KLF) sebagai sistem kontrol. Dalam penelitian ini KLF menggunakan 3 *membership function* untuk masukan *error* dan referensi kecepatan serta keluaran lebar pulsa injeksi, metode inferensi *Max-Min*, dan metode defuzzifikasi *Center of Area* (COA).

Hasil penelitian pada posisi *throttle* 0% menunjukkan *setpoint* kecepatan putaran *engine* 1500 rpm dengan *error steady state* sebesar 1,08%, dan pada posisi *throttle* 5% didapatkan *setpoint* kecepatan putaran *engine* 2000 rpm dengan *error steady state* 0,29%. Dari hasil variasi dua posisi *throttle* tersebut, maka bisa disimpulkan bahwa Kontrol Logika *Fuzzy* dapat menjaga kecepatan putaran *engine* sama dengan *setpoint*.

Kata kunci : *Engine Control Unit*, sistem injeksi, Kontrol Logika *Fuzzy*, Vario 110FI

SUMMARY

Ganda Lesmana, Department of Electrical Engineering, Faculty of Engineering, Brawijaya University, January 2016, "Designing and Manufacturing Fuzzy Logic Control for Rotation Control Engine Based on Throttle Position", Lecturer: Muhammad Aziz Muslim and Goegoes Dwi Nusantoro.

Technological development of fuel supply to the motorcycles engine has progressed rapidly. Now, the carburetor system has been replaced with the injection system causing the calculation of the amount of fuel that sprayed into the engine become more precise. One of the problems in the automotive technology specifically in Engine Control Unit is that thing still become a secret from manufacturer. Therefore, to adjust the engine speed is still quite difficult due to the duration of the injector has been set by the manufacturer.

This research was conducted by using a motorcycle vario 110 FI as plant research with Fuzzy Logic Control (KLF) as the control system. In this research KLF use three membership functions for errors input and speed references also injection pulse output width, Max-Min method of inference and defuzzification method Center of Area (COA).

The result showed at throttle position of 0% indicates the setpoint of engine speed rotation 1500 rpm with steady state error of 1.08%, on the 5% throttle position obtained setpoint of engine rotation speed 2000rpm with steady state error of 0.29%. From the results of the variations of the throttle position, it can be concluded that Fuzzy Logic Control are able to keep the engine rotation speed equal to the setpoint.

Keywords: Engine Control Unit, injection system, Fuzzy Logic Control, Vario 110FI.

PENGANTAR

Bismillahirrohmanirrohim. Alhamdulillah, puji syukur penulis panjatkan kehadirat Allah SWT yang telah memberikan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi yang berjudul "Perancangan dan Pembuatan Kontrol Logika *Fuzzy* untuk Pengendalian Putaran *Engine* Berdasarkan Posisi *Throttle*" dengan baik. Tak lepas shalawat serta salam tercurahkan kepada junjungan kita Nabi Muhammad SAW yang telah menjadi suri tauladan bagi yang mengharapkan rahmat dan hidayah-Nya.

Penulis menyadari bahwa penyusunan skripsi ini tidak lepas dari bantuan, bimbingan serta dorongan dari berbagai pihak. Pada kesempatan ini penulis menyampaikan rasa terima kasih yang sebesar – besarnya kepada:

- Allah SWT yang telah memberikan kelancaran, kemudahan dan hidayah-Nya.
- Kedua orang tua Agus Rianto dan Idhama Kholidah, dan kakak Putri Ika Nirwanasari, serta kedua adik Arini Anggun Dahlia dan Hafizh Fazan Zanuba yang telah banyak memberikan doa, kasih sayang, dukungan, serta semangat.
- Bapak M. Aziz Muslim, S.T., M.T., Ph.D. selaku Ketua Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya Malang dan dosen pembimbing skripsi yang telah memberikan banyak waktu dan tenaganya untuk membimbing dari awal, memberikan saran, nasehat-nasehat, dan pelajaran..
- Bapak Goegoes Dwi Nusantoro, S.T., M.T. selaku dosen pembimbing skripsi yang telah memberikan banyak waktu dan tenaganya untuk membimbing dari awal, memberikan saran, nasehat-nasehat, dan pelajaran.
- Bapak Hadi Suyono S.T., M.T., Ph.D selaku Sekretaris Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya.
- Bapak M. Ali Mustofa, S.T., M.T. selaku Ketua Program Studi Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya yang banyak memberikan pengarahan dalam hal akademik dan penulisan skripsi.
- Bapak Ir. Purwanto, M.T. selaku Ketua Kelompok Dosen Keahlian Teknik Kontrol Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
- **Bapak Ir. Teguh Utomo, M.T.** Selaku dosen penasehat akademik. Yang telah banyak memberikan nasihat-nasihat akademiknya.

- Bapak Dwi Fadilla Kurniawan, S.T., M.T. yang telah memberikan banyak waktu dan tenaganya untuk membimbing dari awal, memberikan saran, nasehat-nasehat, dan pelajaran.
- Bapak, Ibu dosen serta segenap staf dan karyawan Jurusan Teknik Elektro baik secara langsung maupun tidak langsung yang telah banyak membantu dalam menyelesaikan skripsi ini.
- Tim *Engine Control Unit* (ECU) Yudhanto Iman, Intanto Oktavian, dan mantan Tim ECU Johanes Reinhart yang tak henti hentinya berdiskusi dan bertukar pendapat hingga larut malam tentang masalah skripsi ini.
- Cynthia Putri yang tak henti hentinya memberikan semangat agar cepat lulus dan segera wisuda.
- Teman-teman Deworengku. Terima kasih semuanya.
- Tak lupa keluarga besar INVERTER'11, yang memberikan semangat, do'a, dan dukungan.
- Semua pihak yang telah memberikan bantuan serta dukungan baik secara langsung maupun tidak langsung atas penyusunan skripsi ini.

Dalam penyusunan skripsi ini, penulis menyadari bahwa skripsi ini belumlah sempurna, karena keterbatasan ilmu dan kendala – kendala lain yang terjadi selama pengerjaan skripsi ini. Oleh karena itu, penulis berharap kritik dan saran untuk penyempurnaan tulisan di masa yang akan datang, semoga tulisan ini dapat bermanfaat dan dapat digunakan untuk pengembangan lebih lanjut.

Malang, Januari 2016

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1. Latar Belakang

Perkembangan teknologi suplai bahan bakar pada mesin sepeda motor telah mengalami kemajuan sangat pesat. Kini, sistem karburator telah digantikan dengan sistem injeksi agar perhitungan jumlah bahan bakar yang disemprotkan ke dalam mesin menjadi presisi, yakni sesuai dengan perbandingan udara dan bahan bakar yang dibutuhkan. Untuk menghasilkan mesin dengan kadar emisi yang rendah dan performa yang baik dapat diperoleh dengan cara mengatur waktu injeksi bahan bakar.

Mesin-mesin yang telah memiliki *Engine Control Unit* (ECU), yaitu sebuah komponen elektronika yang berfungsi mengontrol kinerja mesin dengan mempertimbangkan masukan dari sensor-sensor, akan dapat mengatur volume bahan bakar yang diinjeksikan ke ruang bakar. Secara detil cara kerja dari *Engine Control Unit* masih menjadi rahasia bagi produsen sepeda motor. Data yang terdapat pada ECU adalah data yang sudah di*setting* oleh pabrikan sesuai dengan kebutuhan masing-masing jenis sepeda motor sehingga data ECU tidak dapat disetting secara umum. Dengan keterbatasan itu maka sistem injeksi tidak dapat disetting sesuai kebutuhan.

Posisi *throttle* berfungsi sebagai gerbang masuknya udara sebelum bercampur dengan bahan bakar yang disemprotkan oleh injektor. Semakin besar posisi pada *throttle* maka udara yang masuk semakin banyak sehingga dibutuhkan semprotan bahan bakar yang banyak juga. Dan semakin kecil bukaan *throttle* maka semakin sedikit pula bahan bakar yang diinjeksikan ke dalam ruang pembakaran.

Fuzzy sebagai metode yang memiliki kemampuan yang baik untuk melakukan pengaturan pada suatu sistem baik linier atau non-linier. Kontroler logika *fuzzy* merupakan salah satu kontroler yang membutuhkan perhitungan yang cukup panjang. Pada implementasinya algoritma kontroler akan ditanamkan pada rangkaian sistem injeksi yang memiliki keterbatasan memori dan kecepatan eksekusi data.

Kontroler diimplementasikan pada sistem injeksi berbasis mikrokontroler. Hasil yang diperoleh dari penelitian ini adalah mengimplementsikan sebuah kontroler *Fuzzy* untuk mengendalikan putaran *engine* sesuai dengan *setpoint*.

1.2. Rumusan Masalah

Berdasarkan latar belakang tersebut, maka dapat diambil suatu rumusan masalah yang akan diteliti yaitu bagaimana perancangan dan implementasi dari kontroler logika *fuzzy* untuk pengendalian putaran *engine* pada sepeda motor Honda Vario 110 FI.

1.3. Batasan Masalah

Untuk membatasi agar penelitian ini lebih spesifik dan lebih terarah maka diberikan batasan-batasan masalah sebagai berikut :

1. Plant yang digunakan adalah mesin motor Honda Vario 110 FI.
2. Sensor yang digunakan sebagai masukan adalah sensor CKP dan TPS (*Throttle Position Sensor*).
3. Menggunakan mikrokontroler ATmega32.
4. Tidak membahas lebih detail tentang prinsip kerja mesin dan termodinamika.
5. Bahan bakar yang digunakan adalah jenis premium ber-oktan 88.
6. Tidak membahas sistem pengapian.

1.4. Tujuan Penelitian

Tujuan yang diperoleh dari penelitian ini adalah membuat rangkaian injektor sepeda motor yang dapat menyesuaikan putaran *engine* terhadap perubahan posisi *throttle* menggunakan metode logika *fuzzy*.

1.5. Sistematika Pembahasan

Sistematika pembahasan dalam penulisan ini sebagai berikut :

BAB I **PENDAHULUAN**, menjelaskan tentang latar belakang masalah, rumusan masalah, batasan masalah, tujuan, manfaat, dan sistematika penulisan.

BAB II **DASAR TEORI**, membahas dasar teori, mesin 4 langkah, sistem injeksi, *Throttle Position Sensor* (TPS), kontroler *fuzzy*, dan ATmega32.

BAB III **METODOLOGI PENELITIAN**, membahas tentang metode penelitian perancangan dan pembuatan sistem.

BAB IV **PERANCANGAN DAN PEMBUATAN ALAT**, menjelaskan tentang perancangan dan pembuatan .

BAB V **PENGUJIAN ALAT,** membahas hasil dari pengujian alat.

BAB VI **PENUTUP**, berisi kesimpulan perancangan, pengujian dan saran-saran yang diperlukan untuk melakukan pengembangan aplikasi selanjutnya.

BAB II
TINJAUAN PUSTAKA

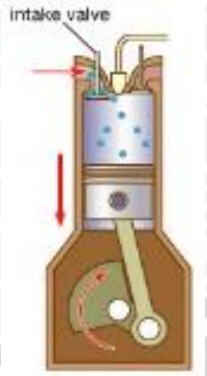
2.1. Proses Motor Bakar 4 Langkah

Mesin pembakaran dalam adalah sebuah mesin yang sumber tenaganya berasal dari pengembangan gas-gas panas bertekanan tinggi hasil pembakaran campuran bahan bakar dan udara yang berlangsung di dalam ruang tertutup dalam mesin.

Motor bakar empat langkah adalah mesin pembakaran dalam, yang dalam satu kali siklus pembakaran akan mengalami empat langkah piston. Sekarang ini, mesin pembakaran dalam pada mobil, sepeda motor, truk, pesawat terbang, kapal, alat berat dan sebagainya, umumnya menggunakan siklus empat langkah. Empat langkah tersebut meliputi langkah hisap (pemasukan), kompresi, tenaga dan langkah buang. Yang secara keseluruhan memerlukan dua putaran poros engkol (*crankshaft*) per satu siklus pada mesin bensin atau mesin diesel.

TMA (titik mati atas) atau TDC (*top dead centre*) adalah posisi piston berada pada titik paling atas dalam silinder mesin atau piston berada pada titik paling jauh dari poros engkol (*crankshaft*). TMB (titik mati bawah) atau BDC (*bottom dead centre*) adalah posisi piston berada pada titik paling bawah dalam silinder mesin atau piston berada pada titik paling dekat dengan poros engkol (*crankshaft*).

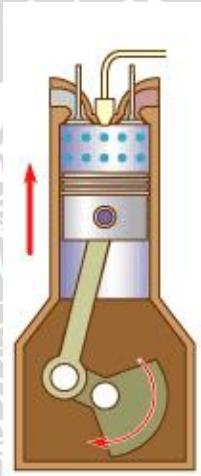
1. Langkah *Air Intake*



Gambar 2.1. Proses *Air Intake*
(sumber : belajar.kemdikbud.go.id)

Piston bergerak dari TMA ke TMB, posisi katup masuk terbuka dan katup keluar tertutup, mengakibatkan udara (mesin diesel) atau gas (sebagian besar mesin bensin) terhisap masuk ke dalam ruang bakar.

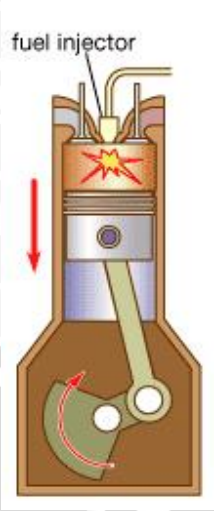
2. Langkah *Compression*



Gambar 2.2. *Compression*
(sumber : belajar.kemdikbud.go.id)

Piston bergerak dari TMB ke TMA, posisi katup masuk dan keluar tertutup, mengakibatkan udara atau gas dalam ruang bakar terkompresi. Beberapa saat sebelum piston sampai pada posisi TMA, waktu penyalaan (*timing ignition*) terjadi pada mesin bensin berupa nyala busi sedangkan pada mesin diesel berupa semprotan bahan bakar.

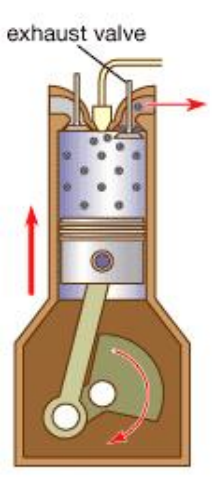
3. Langkah *Combustion*



Gambar 2.3. *Combustion*
(sumber : belajar.kemdikbud.go.id)

Gas yang terbakar dalam ruang bakar akan meningkatkan tekanan dalam ruang bakar, mengakibatkan piston terdorong dari TMA ke TMB. Langkah ini adalah proses yang akan menghasilkan tenaga.

4. Langkah *Exhaust Emission*



Gambar 2.4. *Exhaust Emission*
(sumber : belajar.kemdikbud.go.id)

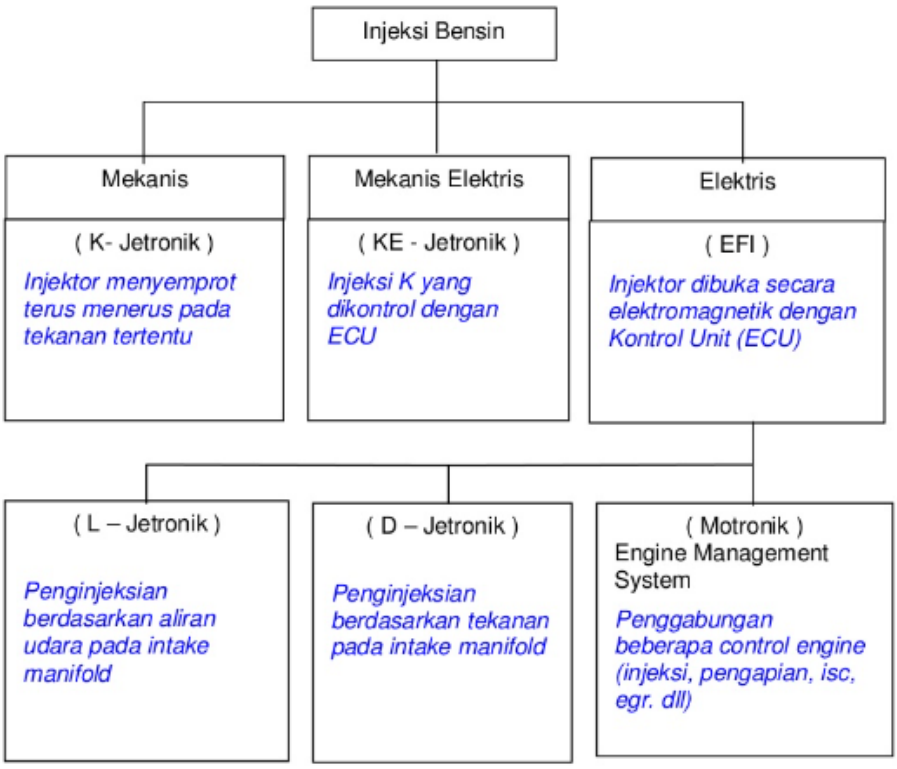
Piston bergerak dari TMB ke TMA, posisi katup masuk tertutup dan katup keluar terbuka, mendorong sisa gas pembakaran menuju ke katup keluar yang sedang terbuka untuk diteruskan ke lubang pembuangan.

2.2. Sistem Injeksi

Sistem injeksi mulai ditemukan pada tahun 1922-1927 ketika Robert bosch menemukan pompa injeksi diesel. Sejak tahun 1960 prinsip injeksi bensin mulai diterapkan pada kendaraan bermotor, pada tahun 1967 pabrik mobil VW sudah menerapkan sistem D-Jetronik, baru tahun 1973 sistem injeksi bensin mulai dipakai secara meluas pada kendaraan bermotor.

Sistem injeksi digunakan untuk menyemprotkan bahan bakar ke dalam *engine* yang akan dicampur dengan udara untuk keperluan pembakaran. Ditinjau dari sistem pengontrol penyemprotan sistem injeksi dapat dibedakan menjadi 3, yaitu sistem injeksi mekanis, sistem injeksi mekanis elektronik, sistem injeksi elektronik.

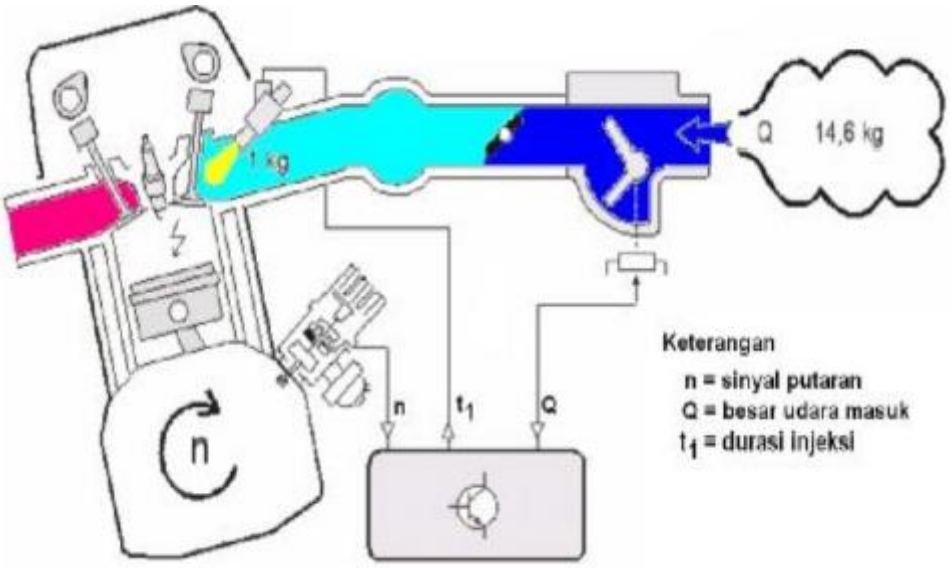
Prinsip kerja dari semua sistem injeksi sama, ada yang sistem injeksi dan sistem pengapian dibuat dengan ECU terpisah dan ada juga yang satu sistem ECU untuk sistem injeksi dan sistem pengapian.



Gambar 2.5. Penggolongan Injeksi Berdasarkan Sistem Pengontrol
(Sumber : Slamet Setiyono, 2008)

Sistem injeksi elektronik dilengkapi dengan sensor-sensor yang dapat dibedakan menjadi sensor utama dan sensor-sensor pengoreksi. Sensor utama digunakan untuk menentukan jumlah penyemprotan injeksi dasar dan sensor-sensor pengoreksi untuk

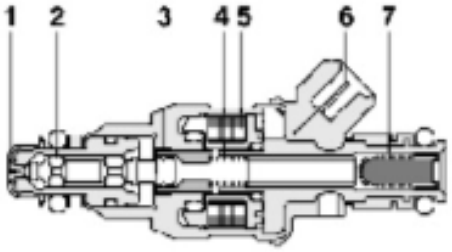
merubah jumlah penyemprotan berdasarkan keadaan-keadaan kerja *engine*. Sesnsor utama untuk mengetahui jumlah udara yang masuk ke *engine* merupakan kombinasi dari sensor massa udara dan sensor putaran mesin. Kedua sensor tersebut menginformasikan kepada ECU berapa jumlah udara yang masuk ke *engine* pada setiap putaran, lalu ECU memberi sinyal kepada injektor dengan durasi penyemprotan tertentu. Harapan dari pengaturan menginginkan bahan bakar diinjeksikan dengan jumlah yang perbandingannya sesuai dengan hokum *stoichiometric*, 14,7 kg massa udara untuk setiap 1 kg bensin. Karena keadaan kerja *engine* sangat beragam dan kebutuhan perbandingan campuran juga beragam maka dipasangkan sensor-sensor lain.



Gambar 2.6. Sensor Utama dan Pengaturan Injeksi
(Sumber : Slamet Setiyono, 2008)

2.2.1. Injektor

Injektor berfungsi menyemprotkan bensin menuju *engine* untuk dicampur dengan udara. Agar bensin mudah bercampur dengan udara maka bensin dikabutkan dengan halus sehingga mudah berubah menjadi uap.



Gambar 2.7. Injektor
(Sumber : Slamet Setiyono, 2008)

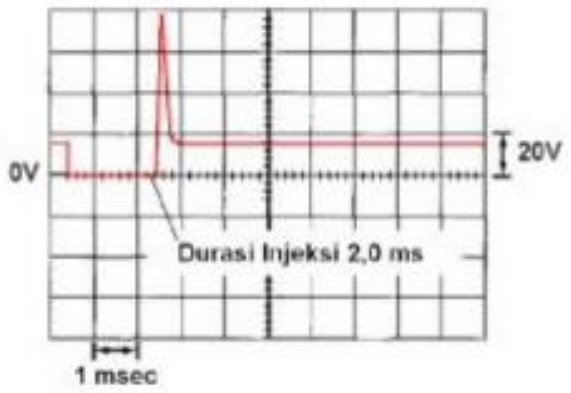
Keterangan :

1. *Pintle*
2. Katup jarum
3. Jangkar
4. Pegas
5. Kumparan solenoid
6. Terminal/konektor
7. *Strainer*/saringan

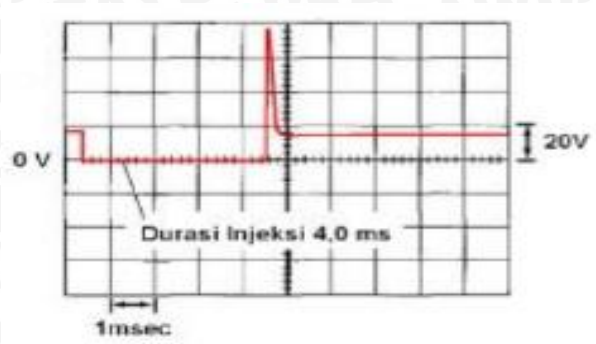
Injektor pada sistem injeksi bekerja secara elektromagnetik. Kerja injector dikontrol oleh ECU dengan sinyal negatif. Lebar pulsa sinyal dari ECU akan menentukan jumlah bahan bakar yang terkabutkan, semakin panjang pulsa semakin banyak bensin terkabutkan.

2.2.2. Pengaturan Injeksi

Durasi injeksi pada sistem injeksi elektronik dipengaruhi oleh kondisi-kondisi kerja *engine* yang dapat diketahui dengan memasangkan sensor-sensor.



Gambar 2.8. Durasi Injeksi Saat Idle
(Sumber : Slamet Setiyono, 2008)

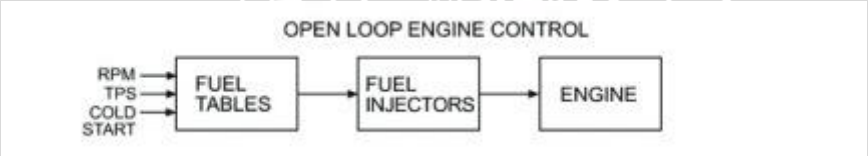


Gambar 2.9. Durasi Saat Berbeban
(Sumber : Slamet Setiyono, 2008)

2.3. *Engine Control Unit* (ECU)

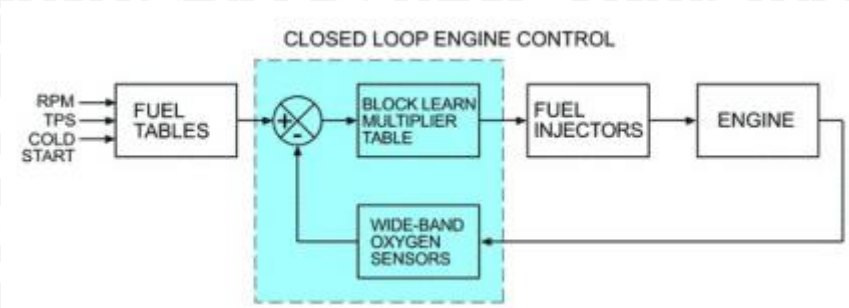
ECU merupakan *embedded system* yang terdiri dari mikrokontroler dan rangkaian elektronis lainnya, yang didalamnya terdapat program dan memori mengenai perlakuan terhadap mesin pada kondisi-kondisi yang berbeda-beda. Misalnya kondisi dihidupkan dalam keadaan mesin dingin, kondisi massa udara yang tinggi, kondisi kecepatan rpm tinggi dan lainnya.

Dalam tugasnya mengontrol injektor, jenis ECU ada dua macam, *open loop* dan *close loop*.



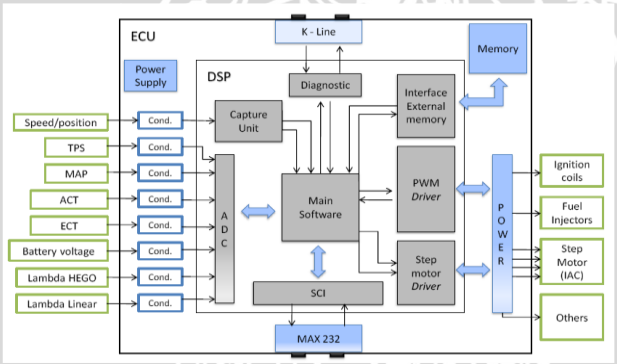
Gambar 2.10. Sistem *Open Loop* ECU
(Sumber : daytona-twintec.com)

Sistem *open loop*, ECU membaca sinyal dari sensor rpm, *Throttle Position Sensor* (TPS), *air flow*, *manifold pressure*, *intake air pressure*, dll. Kemudian memilihkan suatu nilai dalam tabel di memori, berdasarkan nilai tersebut dibangkitkan sinyal untuk dikirim ke injektor agar mensuplai bensin untuk mencapai *air fuel rasio* yang diinginkan.



Gambar 2.11. Sistem *Close Loop* ECU
(Sumber : daytona-twintec.com)

Sedangkan yang *close loop*, merupakan pengembangan dari sistem *open loop*, yang ditambahkan sensor O_2 pada knalpot untuk mengukur kandungan O_2 pada knalpot, jika kandungan O_2 tinggi maka pembakaran miskin dan sebaliknya. Kemudian sensor O_2 memberikan sinyal ke ECU untuk mengoreksi suplai injektor bbm. ECU jenis *close loop* jauh lebih maju atau rumit daripada *open loop* karena cara kerjanya memerlukan algoritma program mikrokontroler yang panjang.



Gambar 2.12. Diagram Blok ECU
(Sumber : motogokil.com)

2.4. Perbandingan Bahan Bakar dan Udara (AFR)

Pada kondisi beroperasi, *spark-ignition engine* membutuhkan campuran antara bahan bakar dan udara yang direpresentasikan dengan rasio perbandingan jumlah bahan bakar dengan udara yang tercampur (*Air to Fuel Ratio / AFR*). Perbandingan ideal dari udara dan bahan bakar untuk satu kali proses pembakaran adalah 14.7 : 1yang berarti bahan bakar sebanyak 1 g berbanding dengan udara sebanyak 14,7 g dan disebut sebagai AFR stokiometrik.

Nilai AFR yang ideal menggambarkan titik keseimbangan terbaik antara daya mesin dengan bahan bakar yang dikeluarkan. Nilai AFR yang lebih besar dari 14,7 : 1 disebut campuran miskin, yang berarti udara lebih banyak daripada kondisi ideal, sehingga daya mesin berkurang dan bahan bakar lebih irit. Nilai AFR yang lebih kecil dari 14,7 : 1 disebut campuran kaya, yang berarti bahan bakar lebih banyak dikeluarkan, namun daya mesin meningkat . AFR kondisi aktual yang terjadi di ruang pembakaran dibagi dengan AFR stokiometrik diperoleh harga AFR relatif yang disebut *Lambda* (λ).

2.5. Sensor *Crankshaft Position* (CKP)

Sensor *Crankshaft Position* adalah perangkat elektronik yang digunakan dalam mesin pembakaran internal untuk memantau posisi atau kecepatan rotasi crankshaft. Informasi ini digunakan oleh sistem manajemen mesin untuk mengontrol waktu sistem pengapian dan parameter mesin lainnya. Sebelum sensor engkol elektronik yang tersedia, distributor harus diatur secara manual untuk tanda waktu pada mesin.

Gambar 2.13. *Crankshaft Position Sensor* (CKP)
(Sumber : Wikipedia.org)

Sensor engkol dapat digunakan dalam kombinasi dengan sensor posisi camshaft mirip dengan memantau hubungan antara piston dan katup di mesin, yang sangat penting dalam mesin dengan variable valve timing. Metode ini juga digunakan untuk "sinkronisasi" mesin empat langkah pada awal, memungkinkan sistem manajemen untuk mengetahui kapan

harus menyuntikkan bahan bakar. Hal ini juga sering digunakan sebagai sumber utama untuk pengukuran kecepatan mesin di putaran per menit.

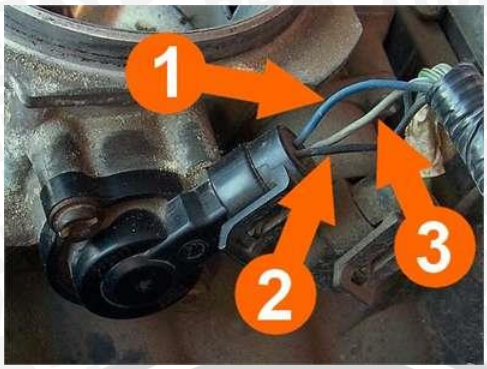
Lokasi pemasangan umum termasuk katrol engkol utama, roda gila, camshaft atau poros engkol itu sendiri. Sensor ini adalah sensor yang paling penting dalam mesin modern. Ketika gagal, ada kemungkinan mesin tidak akan mulai, atau memotong sambil berjalan.

2.6. *Throttle Position Sensor* (TPS)

Throttle adalah bagian dari mesin injeksi yang mengatur masuknya udara ke mesin pembakaran. Fungsi *Throttle Position Sensor* (TPS) adalah sensor yang digunakan untuk memantau posisi *throttle* apakah terbuka sebagian, terbuka penuh atau tertutup. Sensor ini biasanya terletak pada poros kupu-kupu sehingga dapat langsung memantau posisi *throttle*. ECU (*Engine Control Unit*) dapat mengontrol posisi *throttle*.

Gambar 2.14. *Throttle Position Sensor* (TPS)
(Sumber : amazon.com)

Cara kerja *Throttle Position Sensor* adalah ketika pedal gas diinjak atau diputar, maka plat *throttle* akan membuka dan TPS mengukur berapa banyak udara yang masuk dan mengukur sudut bukaan *throttle* dan mengirimkannya ke ECU. Kemudian ECU akan menyemprotkan bahan bakar lebih banyak. Ketika pedal gas dilepas, Plat *throttle* akan menutup dan TPS akan mengukur berapa banyak udara yang masih masuk dan mengukur sudut saat plat *throttle* menutup dan mengirimkannya ke ECU. Sehingga, ECU akan menyemprotkan bahan bakar lebih sedikit.



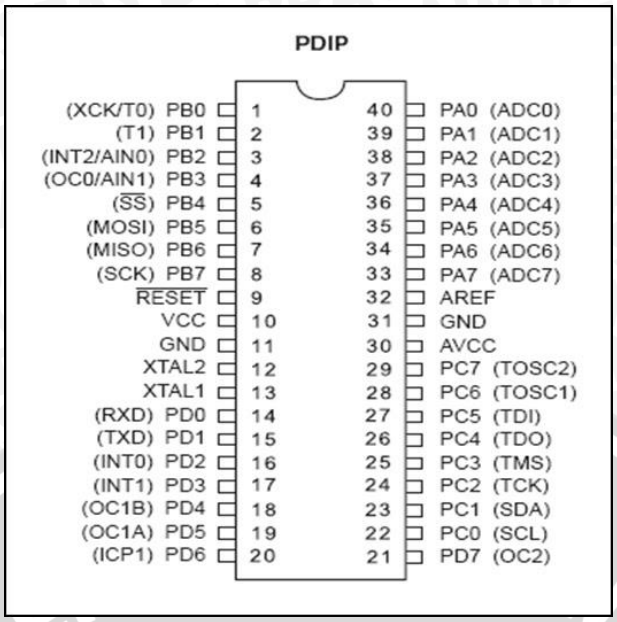
Gambar 2.15. *Wiring* pada TPS
(Sumber : motogurumag.com)

Terdapat 3 kabel yang terhubung dengan *Throttle Position Sensor*. Tiap kabel memiliki fungsi masing-masing, yaitu :

1. Kabel yang bertuliskan angka 1
   - Kabel yang menyediakan *ground* ke sensor.
   - *Ground* disediakan oleh ECU.
2. Kabel yang bertuliskan angka 2
   - Kabel yang menyediakan tegangan sinyal sudut *throttle* ke ECU.
   - Sinyal sudut *throttle* bervariasi tergantung dari jumlah pembukaan plat *throttle*.
3. Kabel yang bertuliskan angka 3
   - Kabel yang menyediakan listrik atau arus B+ ke *Throttle Position Sensor*.
   - Listrik langsung berasal dari ECU.

Sensor TP adalah potensiometer. Perubahan resistensi dalam menanggapi perubahan sudut plat *throttle*. Dengan *throttle* ditutup, tegangan yang dibuat kecil dan dikirimkan ke ECU. Pada saat *throttle* tertutup, sensor TP output sekitar 0.5 Volt DC. Dengan *throttle* dibuka posisi plat *throttle* terbuka lebar, tegangan yang dibuat lebih besar dan dikirim ke ECU. Pada saat *throttle* terbuka lebar, output sensor TP sekitar 4,5 Volt DC.

2.7. Mikrokontroler ATmega32



Gambar 2.16. Mikrokontroler ATmega32
(Sumber : atmel.com)

Secara historis mikrokontroler seri AVR pertama kali diperkenalkan ke pasaran sekitar tahun 1997 oleh perusahaan ATMEL, yaitu sebuah perusahaan yang sangat terkenal dengan produk mikrokontroler seri AT89S51/52-nya yang sampai sekarang masih banyak digunakan di lapangan. Tidak seperti mikrokontroler seri AT89S51/52 yang masih mempertahankan arsitektur dan set instruksi dasar mikrokontroler 8031 dari perusahaan INTEL. Mikrokontroler AVR ini diklaim memiliki arsitektur dan set instruksi yang benar-benar baru dan berbeda dengan arsitektur mikrokontroler sebelumnya yang diproduksioleh perusahaan tersebut. Tetapi walaupun demikian, bagi para programmer yang sebelumnya telah terbiasa dengan mikrokontroler seri AT89S51/52, dan berencana untuk beralih ke mikrokontroler AVR, maka secara teknis tidak akan banyak kesulitan yang berarti, hal ini dikarenakan selain konsep dan istilah-istilah dasarnya hampir sama, pemrograman level *assembler*-nya pun relative tidak jauh berbeda.

Berdasarkan arsitekturnya, AVR merupakan mikrokontroler RISC (*Reduce Instruction Set Computer*) dengan lebar bus data 8 bit. Berbeda dengan sistem AT89S51/52 yang memiliki frekuensi kerja seperduabelas kali frekuensi oscilator, frekuensi kerja mikrokontroler ini pada dasarnya sama dengan frekuensi *oscilator*, sehingga hal tersebut menyebabkan kecepatan kerja AVR untuk frekuensi *oscilator* yang sama, akan dua belas kali lebih cepat dibandingkan dengan mikrokontroler keluarga AT89S51/52.

Dengan instruksi yang sangat variatif (mirip dengan sistem CISC-*Complex Instruction Set Computer*) serta jumlah register serbaguna (*General Purpose Register*) sebanyak 32 buah yang semuanya terhubung secara langsung ke ALU (*Aritmetic Logic Unit*), kecepatan operasi mikrokontroler AVR ini dapat mencapai 16 MIPS (enam belas juta instruksi per detik) sebuah kecepatan yang sangat tinggi untuk ukuran mikrokontroler 8 bit yang ada di pasaran sampai saat ini.

Untuk memenuhi kebutuhan dan aplikasi industri yng sangat beragam, mikrokontroler keluarga AVR ini muncul di pasaran dengan tiga seri utama : tinyAVR, ClassicAVR (AVR), megaAVR. Berikut ini beberapa seri yang dapat anda jumpai di pasaran :

ATtiny13	AT90S2313	ATmega103
ATtiny22	AT90S2323	ATmega128
ATtiny22L	AT90S2333	ATmega16
ATtiny2313	AT90S4414	ATmega162
ATtiny2313V	AT90S4433	ATmega168

Keseluruhan seri AVR ini pada dasarnya memiliki organisasi memori dan set instruksi yang sama (sehingga dengan demikian jika telah mahir menggunakan salah satu seri AVR untuk beralih ke seri yang lain akan relativ mudah). Perbedaan antara tinyAVR, AVR dan megaAVR pada kenyataannya hanya merefleksikan tambahan-tambahan fitur yang ditawarkannya saja (misal adanya tambahan ADC internal pada seri AVR tertentu, jumlah port I/O serta memori yang berbeda, dan sebagainya). Diantara ketiganya, megaAVR umumnya memiliki fitur yang paling lengkap, disusul oleh AVR, dan terakhir tinyAVR.

Untuk memberi gambaran yang lebih jelas, pada tabel di bawah memperlihatkan perbedaan ketiga seri AVR ditinjau dari jumlah memori yang dimilikinya.

Tabel 2.1. Seri AVR

Mikrokontroler AVR		**Memori (byte)**		
Jenis	**Paket IC**	**Flash**	**EEPROM**	**SRAM**
TinyAVR	8 – 32 pin	1 – 2k	64 – 128	0 – 128

AVR (classic AVR)	20 – 44 pin	1 – 8k	128 – 512	0 – 1k
MegaAVR	32 – 64 pin	8 – 128k	512 – 4k	512 – 4k

Seperti terlihat pada tabel tersebut, semua jenis AVR ini telah dilengkapi dengan memori flash sebagai memori program. Tergantung serinya, kapasitas memori flash yang dimiliki bervariasi dari 1k sampai 128kb. Secara teknis, memori jenis ini dapat diprogram melalui saluran antarmuka yang dikenal dengan nama *Serial Peripheral Interface* (SPI) yang terdapat pada setiap seri AVR tersebut. Dengan menggunakan perangkat lunak *programmer (downloader)* yang tepat, pengisian memori *Flash* dengan menggunakan saluran SPI ini dapat dilakukan bahkan ketika chip AVR terpasang pada sistem akhir (*end system*), sehingga dengan demikian pemrogramannya sangat fleksibel dan tidak merepotkan pengguna (secara praktis metode ini dikenal dengan istilah ISP-*In System Programming*-sedangkan perangkat lunaknya dinamakan *In System Programmer*).

Untuk pentimpanan data, mikrokontroler AVR menyediakan dua jenis memori yang berbeda : EEPROM (*Electrically Erasable Programmablr Read Only*) dan SRAM (*Static Random Access Memory*). EEPROM umumnya digunakan untuk menyimpan data-data program yang bersifat permanan, sedangkan SRAM digunakan untuk menyimpan data variabel yang dimungkinkan berubah setiap saatnya. Kapasitas simpan data kedua memori ini bervariasi tergantung pada jenis AVR-nya (lihat Tabel 2.2). Untuk seri AVR yang tidak memiliki SRAM, penyimpanan data variabel dapat dilakukan pada register serbaguna yang terdapat pada CPU mikrokontroler tersebut.

Selain seri-seri diatas yang sifatnya lebih umum, perusahaan ATMEL juga memproduksi beberapa jenis mikrokontroler AVR untuk tujuan yang lebih khusus dan terbatas, seperti seri AT86RF401 yang khusus digunakan untuk aplikasi *wireless remote control* dengan menggunakan gelombang radio (RF), seri AT90SC yang khusus digunakan untuk peralatan sistem-sistem keamanan kartu SIM GSM, pembayaran via internet, dan lain sebagainya.

2.8. Arsitektur Atmega32

Mikrokontroler AVR mempunyai arsitektur *harvard*, dimana memori untuk data dan untuk program terpisah, bus untuk data dan bus untuk program pun terpisah. Dalam

arsitektur AVR, seluruh GPR (*General Purpose Register*) terhubung langsung ke ALU (*Aritmetic and Logic Unit*) *processor*. Sehingga eksekusi instruksi lebih cepat. Dalam satu siklusclock terdapat dua register yang dapat diakses oleh satu instruksi. Teknik yang digunakan adalah memegang sambil mengerjakan. Hal ini berarti, dua operan dibaca dari satu register, dilakukan eksekusi operasi, dan hasilnya disimpan kembali dalam satu register, semuanya dilakukan hanya dalam satu siklus clock.

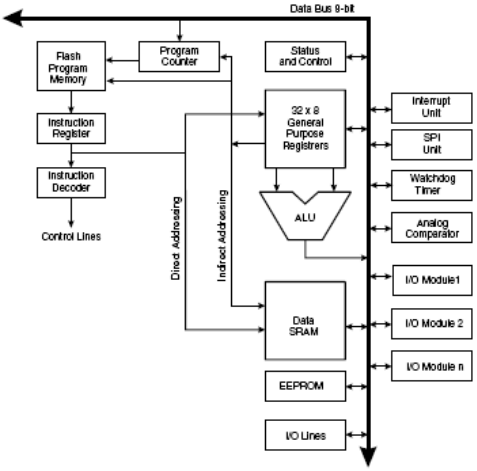
Dari 32 register terdapat enam buah register yang dapat digunakan untuk pengalamatan tidak langsung 16-bit sebagai pointer (penunjuk). Register tersebut memiliki nama khusus, yaitu X, Y, Z. Masing-masing terdiri dari sepasang register, seperti : X (R27:26), Y (R29:28), Z (R31:30).

ALU mendukung operasi bit, fungsi aritmatika dan logika antara register dengan register atau antara register dengan nilai konstan, atau hanya operasi satu register.

Untuk kontrol aliran program disediakan instruksi lompatan bersyarat dan tak bersyarat, instruksi CALL (panggil), dapat ditempatkan di seluruh ruangan program. Kebanyakan instruki AVR mempunyai format 16-bit *word*. Setiap alamat memori program mengandung instruksi 16 atau 32-bit.

Selama interupsi dan pemanggilan subrutin, alaat *progam counter* (PC) disimpan kedalam *stack. Stack* akan efektif jika diletakkan di SRAM, dan konsekuensinya ukuran stack dibatasi oleh ukuran SRAM. Setiap pengguna harus menginisialisasi SP dalam rutin RESERT sebelum subrutin atau interupsi dieksekusi. *Stack pointer* SP dapat ditulis dan dibaca dalam ruangan I/O. SRAM dapat diakses melalui lima mode pengalamatan dala arsitektur AVR.

AVR ATMega32 mempunyai 8 *channel* ADC dan 4 port I/O, yaitu port A (8 bit), port B (8 bit), port C (8 bit), dan port D (8 bit).



Gambar 2.17. Arsitektur ATmega32
(Sumber : atmel.com)

2.9. Kontrol Logika Fuzzy (KLF)

Fuzzy secara harfiah berarti samar, sedangkan kebalikannya dalam hal ini adalah Crisp yang secara harfiah berarti tegas. Dalam kehidupan sehari-hari nilai samar lebih akrab daripada nilai tegas. Temperatur/suhu tertentu biasa dinyatakan sebagai panas, agak panas, atau sangat dingin daripada dinyatakan dalam nilai terukur tertentu.

Tahun 1965 L.A. Zadeh memodifikasi teori himpunan yang disebut himpunan kabur (Fuzzy Set). Himpunan fuzzy didasarkan pada gagasan untuk memperluas jangkauan fungsi karakteristik sehingga fungsi tersebut akan mencakup bilangan real pada interval [0, 1]. Nilai keanggotaannya menunjukkan bahwa suatu nilai dalam semesta pembicaraan tidak hanya berada pada 0 atau 1, namun juga nilai yang terletak diantaranya. Dengan kata lain nilai kebenaran suatu hal tidak hanya bernilai benar atau salah. Nilai 0 menunjukkan salah, nilai 1 menunjukkan benar dan masih ada nilai-nilai yang terletak diantaranya.

Sejak tahun 1985 pengaturan berbasis logika fuzzy mengalami perkembangan pesat, terutama dalam hubungannya dengan penyelesaian masalah kendali yang bersifat tak linier, sulit dimodelkan, berubah karakteristiknya terhadap waktu (time varying) dan kompleks.

2.9.1. Struktur Dasar Logika Fuzzy

Dalam sistem pengaturan dengan logika fuzzy dilibatkan suatu blok pengendali yang menerima satu atau lebih masukan dan mengumpankan satu atau lebih keluaran ke plant atau

blok lain. Komponen utama penyusun Fuzzy Logic Controller adalah unit fuzzifikasi, fuzzy inference, basis pengetahuan dan unit defuzzifikasi.

Basis pengetahuan terdiri dari dua jenis (Yan,1994):

1. Basis data

   Mendefinisikan parameter fuzzy sebagai bagian dari himpunan fuzzy dengan menentukan batas-batas fungsi keanggotaan pada semesta pembicaraan untuk tiap-tiap variabel.

2. Basis aturan

   Memetakan nilai masukan fuzzy menjadi nilai keluaran fuzzy.

2.9.2. Fungsi Keanggotaan

Fungsi keanggotaan menotasikan nilai kebenaran anggota-anggota himpunan fuzzy. Interval nilai yang digunakan untuk menentukan fungsi keanggotaan, yaitu nol dan satu. Tiap fungsi keanggotaan memetakan elemen himpunan crisp ke semesta himpunan fuzzy.

Suatu himpunan fuzzy A dalam semesta pembicaraan U dinyatakan dengan fungsi keanggotaan, μ_A yang harganya berada dalam interval [0,1] (Kuswadi, 2000:27). Secara matematika hal ini dinyatakan dalam Persamaan (2-1) berikut:

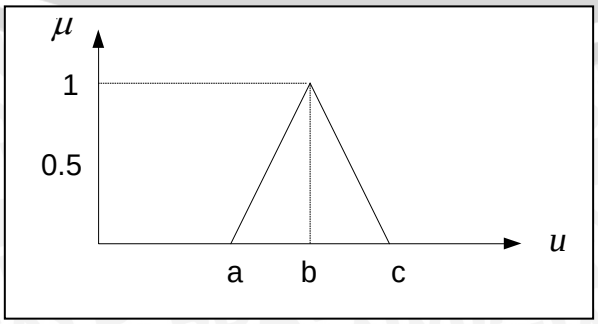
$$\mu_A : U \rightarrow [0,1] \qquad (2\text{-}1)$$

- Fungsi keanggotaan bentuk Triangular

  Definisi fungsi triangular ditunjukkan dalam Persamaan (2-2) berikut:

$$T(u;a,b,c) = \begin{cases} 0 & u < a \\ \dfrac{u-a}{b-a} & a \le u \le b \\ \dfrac{c-u}{c-b} & b \le u \le c \\ 0 & u > c \end{cases} \qquad (2\text{-}2)$$

  Fungsi keanggotaan bentuk triangular ditunjukkan dalam Gambar 2.18.



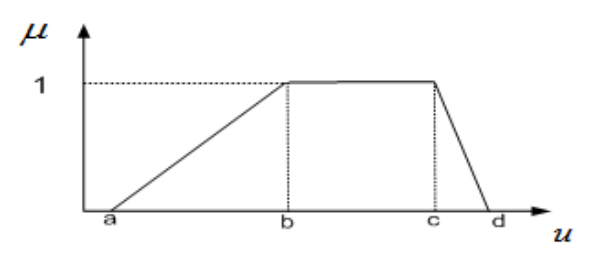
Gambar 2.18. Fungsi Keanggotaan Bentuk Triangular
(Sumber: Yan, 1994)

Fungsi keanggotaan bentuk triangular ini digunakan bila diinginkan himpunan fuzzy mempunyai nilai proporsional terhadap nol maupun satu.

- Fungsi keanggotaan bentuk Trapesium
  Definisi fungsi trapesium ditunjukkan dalam Persamaan (2-3) berikut:

$$T(u;a,b,c,d) = \begin{cases} 0 & u < a \\ \dfrac{u-a}{b-a} & a \le u \le b \\ 1 & b \le u \le c \\ \dfrac{d-u}{d-c} & c \le u \le d \\ 0 & d \le u \end{cases} \quad (2\text{-}3)$$

Fungsi keanggotaan bentuk trapesium ditunjukkan dalam Gambar 2.19.



Gambar 2.19. Fungsi Keanggotaan Bentuk Trapesium
(Sumber: Yan, 1994)

2.9.3. Kontroler Logika Fuzzy

Kontrol Logika Fuzzy (KLF) adalah sistem berbasis aturan (*rule based system*) yang didalamnya terdapat himpunan aturan fuzzy yang mempresentasikan mekanisme pengambilan keputusan. Aturan yang dibuat digunakan untuk memetakan variabel input ke variabel output dengan pernyataan If – Then.

Kontroler ini akan menggunakan data tertentu (crisp) dari sejumlah sensor kemudian mengubahnya menjadi bentuk linguistik atau fungsi keanggotaan melalui proses fuzzifikasi. Lalu dengan aturan fuzzy, inference engine yang akan menentukan hasil keluaran fuzzy.

Setelah itu hasil ini akan diubah kembali menjadi bentuk numerik melalui proses defuzzifikasi.

2.9.3.1. Fuzzifikasi

Proses fuzzifikasi merupakan proses untuk mengubah variabel non fuzzy (variabel numerik) menjadi variabel fuzzy (variabel linguistik). Nilai masukan-masukan yang masih dalam bentuk variabel numerik yang telah dikuantisasi sebelum diolah oleh pengendali logika fuzzy harus diubah terlebih dahulu ke dalam variabel fuzzy. Melalui fungsi keanggotaan yang telah disusun, maka dari nilai-nilai masukan tersebut menjadi informasi fuzzy yang berguna nantinya untuk proses pengolahan secara fuzzy pula. Proses ini disebut fuzzifikasi (Yan,1994). Proses fuzzifikasi diekspresikan sebagai berikut:

X = fuzzifier (x_0)

dengan:

x_0 = nilai crisp variabel masukan

x = himpunan fuzzy variabel yang terdefinisi

fuzzifier = operator fuzzifikasi yang memetakan himpunan crisp ke himpunan fuzzy

Pedoman memilih fungsi keanggotaan untuk proses fuzzifikasi, menurut Jun Yan, menggunakan:

1. Himpunan fuzzy dengan distribusi simetris.
2. Gunakan himpunan fuzzy dengan jumlah ganjil, berkaitan erat dengan jumlah kaidah (rules).
3. Mengatur himpunan fuzzy agar saling menumpuk.
4. Menggunakan fungsi keanggotaan bentuk segitiga atau trapesium.

2.9.3.2. Kaidah Aturan Fuzzy (Fuzzy Rule)

Fuzzy rule adalah bagian yang menggambarkan dinamika suatu sistem terhadap masukan yang dikarakteristikan oleh sekumpulan variabel-variabel linguistik dan berbasis pengetahuan seorang operator ahli. Pernyataan tersebut umumnya dinyatakan oleh suatu pernyataan bersyarat.

Dalam pengendali berbasis fuzzy, aturan pengendalian fuzzy berbentuk aturan “IF – THEN”. Untuk sebuah sistem Multi Input Single Output (MISO) basis aturan pengendalian fuzzy berbentuk seperti berikut ini:

Rule 1 IF X is A1 AND Y is B1 THEN Z is C1
Rule 2 IF X is A2 AND Y is B2 THEN Z is C2
. .
. .
. .
Rule n IF X is An AND Y is Bn THEN Z is Cn

Dengan X, Y, Z merupakan variabel linguistik, dimana X dan Y merupakan variabel masukan, dan Z merupakan variabel keluaran sistem. An, Bn, dan Cn merupakan nilai linguistik dari X, Y, dan Z (Lee, 1990).

2.9.3.3. Metode Inferensi MAX-MIN

Metode inferensi merupakan proses untuk mendapatkan keluaran dari suatu kondisi masukan dengan mengikuti aturan-aturan yang telah ditetapkan. Keputusan yang didapatkan pada proses ini masih dalam bentuk fuzzy yaitu derajat keanggotaan keluaran.

Pada metode Max–Min aturan operasi minimum Mamdani digunakan untuk implikasi fuzzy yang ditunjukkan dalam Persamaan (2-4) dan Persamaan (2-5). Persamaan aturan minimum adalah:

$$\mu_{C'} = \bigcup_{1}^{n} \alpha_i \wedge \mu ci \qquad (2\text{-}4)$$

dengan $\alpha_i = \mu_{Ai}(x_0) \wedge \mu_{Bi}(y_0)$ (2-5)

Sebagai contoh, terdapat dua baris kaidah atur fuzzy, yaitu:

R1 : Jika x adalah A1 dan y adalah B1 maka z adalah C1

R2 : Jika x adalah A2 dan y adalah B2 maka z adalah C2

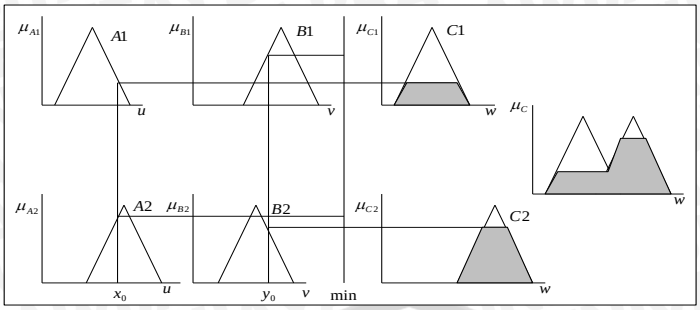
Pada metode penalaran MAX-MIN fungsi keanggotaan konsekuen dinyatakan dalam Persamaan (2-6), Persamaan (2-7), dan Persamaan (2-8) berikut:

$$\mu c_{1'}(W) = \mu_{c'1} \vee \mu_{c'2} = [\alpha_1 \wedge \mu_{c1}(w)] \vee [\alpha_2 \wedge \mu_{c2}(w)] \qquad (2\text{-}6)$$

dimana $\alpha_1 = \mu_{A1}(x_0) \wedge \mu_{B1}(y_0)$ (2-7)

$$\alpha_2 = \mu_{A2}(x_0) \wedge \mu_{B2}(y_0) \qquad (2\text{-}8)$$

Lebih jelas metode ini dideskripsikan dalam Gambar 2.20.



Gambar 2.20. Inferensi Fuzzy dengan Metode MAX-MIN

Sumber: Yan, 1994

2.9.3.4. Metode Defuzzifikasi Weighted Average (WA)

Defuzzifikasi adalah proses untuk mendapatkan nilai numerik dari data fuzzy yang dihasilkan dari proses inferensi (Yan, 1994). Proses defuzzifikasi dinyatakan dalam Persamaan (2-9) sebagai berikut:

$$y_0 = defuzzifier\ (y) \qquad (2\text{-}9)$$

dengan:

y : aksi kontrol *fuzzy*

y_0 : aksi kontrol *crisp*

defuzzifier : operator defuzzifikasi

Metode Weighted Average (WA) ini didefinisikan dalam Persamaan (2-10) sebagai berikut:

$$U = \frac{\sum_{i=1}^{n} w_i u_i}{\sum_{i=1}^{n} w_i} \qquad (2\text{-}10)$$

dengan:

U = Keluaran

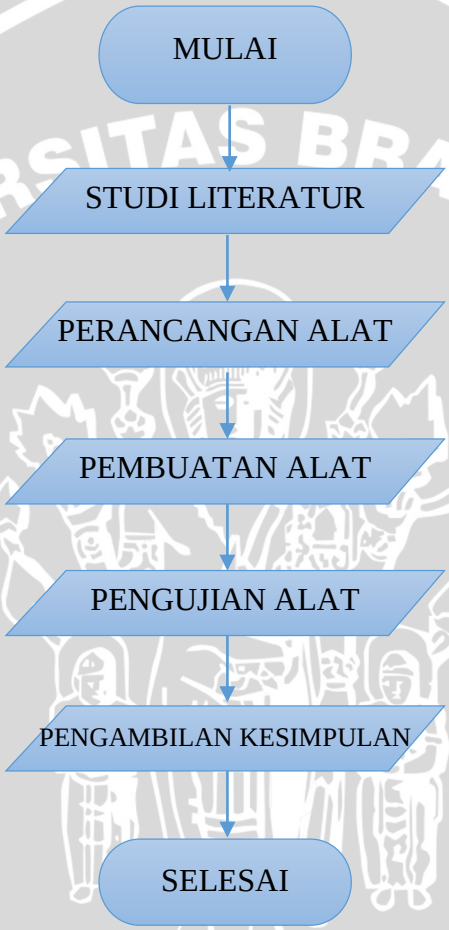
w_i = Bobot nilai benar w_i

u_i = Nilai linguistik pada fungsi keanggotaan keluaran

n = Banyak derajat keanggotaan

BAB III
METODOLOGI

Kajian yang dilakukan dalam penelitian ini adalah perancangan dan pembuatan kontroler *fuzzy* untuk pengendalian putaran *engine* terhadap posisi *throttle*. Metode yang digunakan untuk merealisasikan alat tersebut seperti pada diagram alir di bawah.



Gambar 3.1. Diagram Alir Metode Penelitian

3.1. Studi Literatur

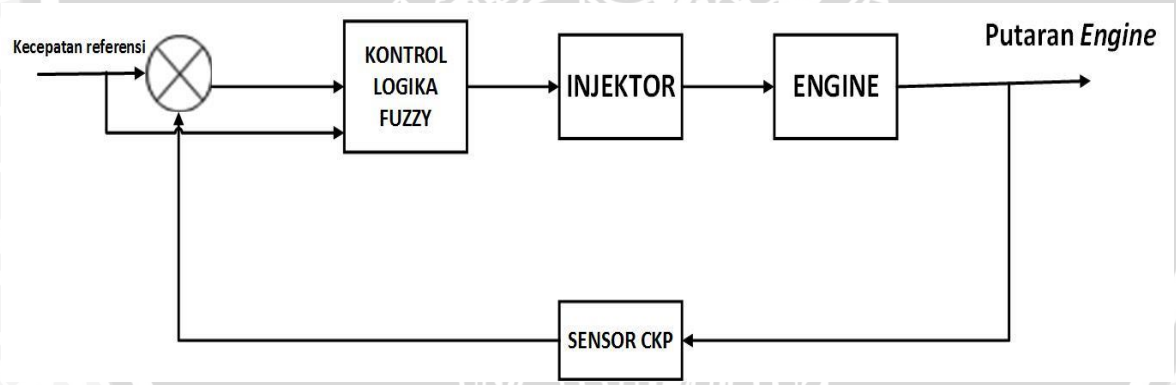
Studi literatur dilakukan untuk mendapatkan pengetahuan dasar tentang segala sesuatu yang mendukung perancangan serta pembuatan alat ini. Pembuatan alat ini menggunakan referensi dari buku-buku maupun artikel dari internet agar mengetahui prinsip kerja, karakteristik komponen, serta teori yang menunjang.

3.2. Perancangan Alat

Perancangan *plant* dilakukan dengan perancangan komponen sistem injeksi dan sensor-sensor yang dibutuhkan dalam pengontrolan mesin. *Plant* yang digunakan dalam penelitian ini adalah mesin Honda Vario 110 FI menggunakan sistem injeksi bahan bakar. Sistem injeksi merupakan sistem distribusi bahan bakar yang menggunakan injektor sebagai penyemprot bahan bakar yang dikendalikan oleh suatu mikrokontroler sebagai pengontrol dan *driver* injektor sebagai aktuator.

3.2.1. Perancangan Blok Diagram Sistem

Diagram balok merupakan penggambaran suatu sistem dalam bentuk balok, setiap balok mewakili elemen sistem yang memiliki fungsi tertentu. Diagram balok dapat memudahkan dalam menganalisa dan merancang sistem yang akan dibuat. Berikut diagram balok dari perancangan dan pembuatan kontroler *fuzzy* untuk pengendalian putaran *engine* terhadap bukaan throttle:



Gambar 3.2. Diagram Blok Sistem

3.2.2. Konfigurasi Sensor

Karena penggunaan sistem EFI, mesin membutuhkan beberapa sensor untuk mendeteksi perubahan parameter yang terjadi pada mesin. Yaitu, Sensor TPS (*Throttle Position Sensor*) dan sensor *Crankshaft Position* (CKP). Bacaan sensor tersebut akan menjadi masukan untuk ECU (*Electronic Control Unit*) sehingga dapat menentukan besarnya sinyal kontrol yang diberikan. Dalam hal ini sinyal kontrol yang diberikan adalah durasi injeksi yang sesuai.

3.3. Pembuatan Alat

Dalam pembuatan alat dalam penelitian tentang perancangan dan pembuatan kontroler logika *fuzzy* untuk pengendalian putaran *engine* terhadap posisi *throttle* dibagi menjadi beberapa bagian.

3.3.1. Perancangan *Hardware* Elektronika

Engine Control Unit (ECU) merupakan elemen kontrol elektronik utama pada sistem injeksi. Penentuan banyak bahan bakar yang digunakan, waktu pengapian dan parameter-parameter lain yang mempengaruhi kinerja mesin dilakukan pada elemen ini, sehingga tanpa adanya aksi dari ECU mesin tidak akan mampu beroperasi seperti sebagaimana mestinya. Masukan dari sensor-sensor yang terdapat pada mesin diolah dan dikalkulasi di dalam ECU, sehingga mampu memberikan aksi kontrol langsung pada aktuator. Pengolahan sinyal masukan hingga memberikan nilai keluaran dilakukan dengan algoritma tertentu yang ditanamkan pada mikrokontroler. Di dalam ECU terdapat 2 unit mikrokontroler, rangkaian *debouncing*, dan pengkondisi sinyal keluaran. Mikrokontroler berfungsi sebagai elemen pengolah, pengkalkulasi dan penentu keputusan aksi kontrol,rangkaian, *debouncing* berfungsi sebagai penghalus sinyal sensor TPS dan CKP akibat munculnya efek bounching (semacam *ripple* kecil pada bagian keadaan *high*), sedangkan pengkondisi sinyal keluaran digunakan untuk menyesuaikan sinyal agar dapat diterima oleh aktuator.

3.3.2. Perancangan *Software*

Pada perancangan tugas akhir ini, kondisi mesin adalah tidak berbeban, sehingga dengan penambahan bukaan katup *throttle* akan menambah putaran mesin. Perancangan kontroler logika *fuzzy* didasari dari data hubungan lebar pulsa injeksi dengan kecepatan putar mesin dan variasi bukaan katup throttle. Nilai *operating point* atau titik tengah dari fungsi keanggotaan *fuzzy* merupakan nilai optimal dari konsumsi bahan bakar. Nilai optimal adalah nilai konsumsi bahan bakar yang paling sedikit untuk mencapai kecepatan tertinggi pada bukaan katup *throttle* tertentu, nilai optimal dicari dengan cara melakukan eksperimen.

3.4. Pengujian Alat

Untuk mengetahui tingkat keberhasilan sistem dan kesesuaian dengan perancangan maka dilakukan pengujian rangkaian. Pengujian dilakukan pada masing-masing blok maupun keseluruhan sistem.

3.5. Pengambilan Kesimpulan dan Saran

Kesimpulan dapat berdasarkan dari hasil perealisasian dan pengujian alat sesuai dengan tujuan dan rumusan masalah. Saran diberikan setelah melihat adanya kekurangan dalam sistem yang telah dibuat, dengan maksud agar alat ini dapat dikembangkan dengan baik.

BAB IV
PERANCANGAN DAN PEMBUATAN ALAT

Dalam bab ini membahas mengenai perancangan dan pembuatan kontroler logika fuzzy untuk pengendalian putaran *engine* terhadap bukaan *throttle*. Perancangan perangkat tersebut meliputi perancangan perangkat keras dan perancangan perangkat lunak. Sedangkan pembuatan bertujuan untuk menghasilkan semua perangkat pendukung maupun alat secara keseluruhan.

4.1. Perancangan Sistem

Perancangan alat ini dilakukan bertahap dalam bentuk diagram blok sehingga memudahkan dalam analisis pada setiap bloknya maupun secara keseluruhan. Perancangan ini terdiri atas :

1. Perancangan perangkat keras (*Engine*-4 tak, sensor *Throttle Position*, *Fuel Pump*, Rangkaian Injektor, Rangkaian Sensor Injektor, Rangkaian Pengkondisi Sinyal Pulser, Rangkaian Mikrokontroler, dan Rangkaian Keseluruhan).
2. Perancangan perangkat lunak (perancangan algoritma kontroler logika fuzzy pada software BASCOM AVR).

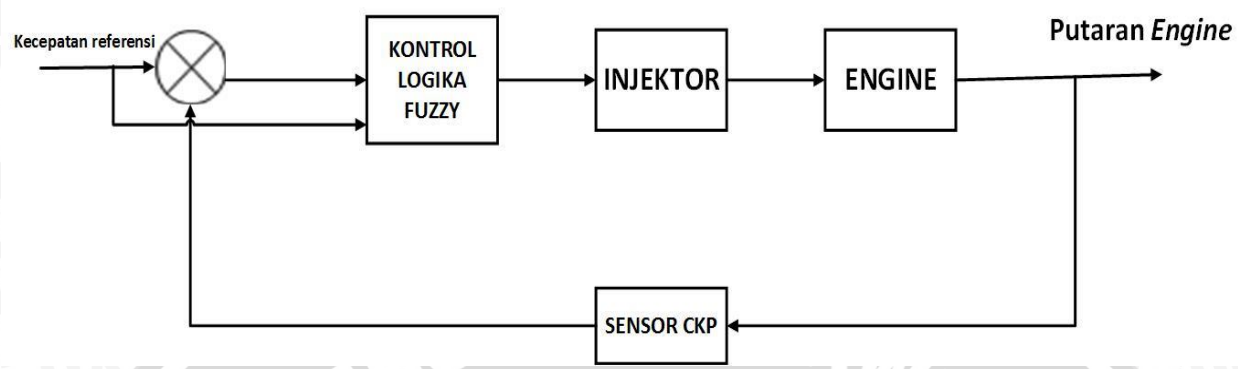
4.2. Spesifikasi Sistem

Spesifikasi alat yang dirancang adalah sebagai berikut :

1. Mesin OHC 4 langkah.
2. Injektor menggunakan keihin vario 110 fi dengan jumlah hole 6.
3. CDI (*Capacitor Discharge Ignition*) sebagai alat bantu sistem pengapian pada mesin pembakaran dalam.
4. Aki sebagai pengkondisi sinyal tegangan masukan CDI sebesar 12 V.
5. Putaran *Engine* didapatkan dari proses pembakaran dalam melalui campuran udara dan bahan bakar (oktan 88).
6. Kecepatan putaran yang dihasilkan *Engine* yaitu 1500 - 8500.
7. ATmega32 dan software BASCOM AVR sebagai tempat pemrograman kontrol logika fuzzy.
8. Menggunakan sensor CKP dan sensor *Throttle Position*.

4.3. Diagram Blok Sistem

Dalam skripsi ini dibuat diagram blok agar dalam pengerjaan dapat dilakukan sesuai dengan rancangan sistem. Adapun diagram blok tersebut sebagai berikut.



Gambar 4.1. Diagram blok sistem

Keterangan dari diagram blok dalam Gambar 4.1. :

- Masukan / *setpoint* berupa kecepatan putaran (rpm) berdasarkan posisi *throttle* yang diberikan melalui program pada ATmega32.
- Kemudian input diolah dan menghasilkan sinyal PWM yang akan menjadi masukan untuk menggerakkan injektor.
- Sinyal dari injektor kemudian akan menggerakkan CDI yang berfungsi untuk pengapian bahan bakar sehingga menghasilkan putaran mesin sesuai *setpoint*.
- Keluaran putaran kemudian dibaca oleh sensor CKP untuk mengetahui putaran yang dihasilkan oleh *engine*.
- Hasil akhir pembacaan sensor kemudian dikurangkan dengan *input/setpoint* sehingga mikrokontroler mengkompensasi *error* yang terjadi.

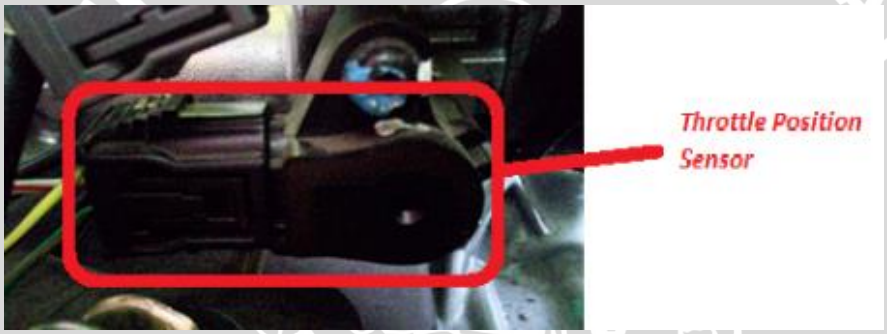
4.4. Perancangan Perangkat Keras

4.4.1. Engine 4 tak

Engine 4 tak adalah mesin pembakaran dalam yang akan mengalami empat langkah piston dalam satu kali proses untuk mengahsilkan putaran. Spesifikasi yang dipilih berdasarkan kebutuhan standart sepeda motor pabrikan, mesin dengan tipe 4 tak, kapasitas 108cc ini memiliki kekuatan maksimal 8,52 PS/8.500 rpm.

Gambar 4.2. Mesin Vario 110 FI

4.4.2. Sensor *Throttle Position*

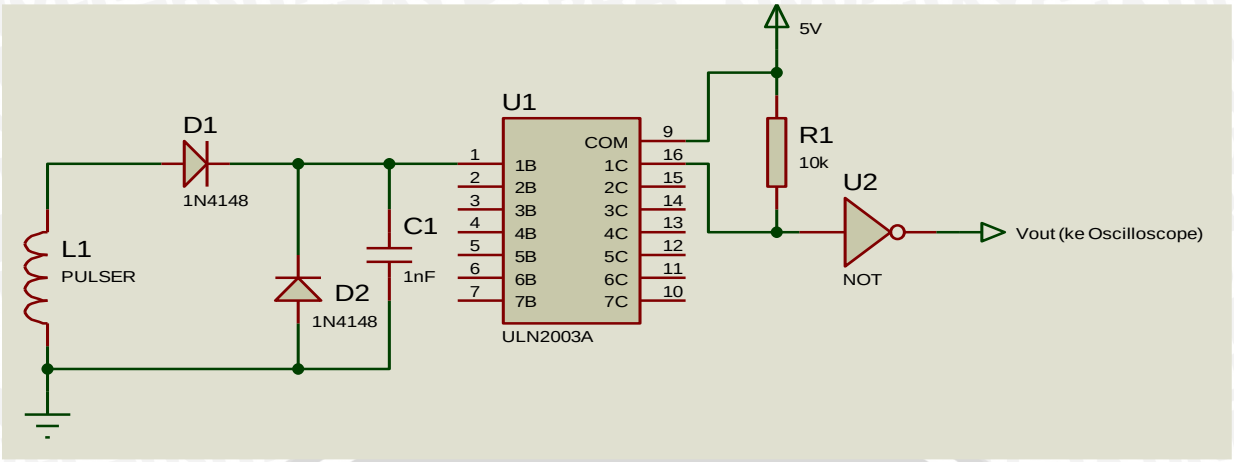


Gambar 4.3. *Throttle Position Sensor*

Sensor ini bekerja membaca besar kecilnya bukaan throttle yang berfungsi untuk mengatur udara yang masuk ke dalam ruang pembakaran. Sensor throttle position akan diolah oleh ECU untuk mengubah sudut katup throttle ke dalam sinyal tegangan listrik dari 0,39 V – 4,4 V. Jika *throttle* membuka, maka tegangan semakin meningkat dan sebaliknya.

4.4.3. Perancangan Pengkondisi Sinyal Sensor CKP

Sensor CKP ini berfungsi untuk mengetahui putaran mesin dan mengubah putaran tersebut ke dalam bentuk pulsa untuk diolah ECU. Kemudian pulsa yang diterima oleh ECU akan diproses untuk menentukan durasi bahan bakar yang akan diinjekkan ke dalam ruang bakar.

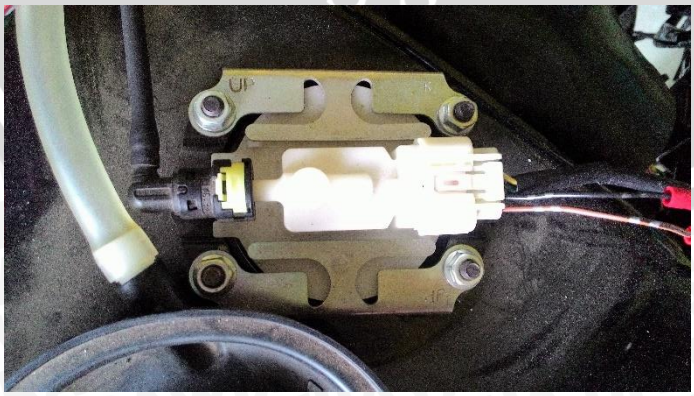


Gambar 4.4. Rangkaian Pengkondisi Sinyal Sensor CKP

Gambar 4.4 merupakan rangkaian pengkondisi sinyal dari pulser yang digunakan pada sistem. Pulser ini sangat penting karena merupakan parameter utama untuk sistem keseluruhan.

Pada rangkaian Gambar 4.4. D1 dan D2 berfungsi untuk mengubah sinyal AC menjadi DC, dimana sinyal pulser yang awalnya sinyal hanya bagian positif yang dilewatkan melalui D1, sementara D2 bertugas meniadakan sinyal minus dengan menshortkan ke ground, selanjutnya keluaran D1 dihubungkan ke driver ULN2003 yang terdiri dari transistor dengan rangkain darlington didalamnya. Pada saat input ULN2003 high, maka output ULN2003 menjadi low dan begitu sebaliknya. Output dari ULN2003 selanjutnya dihubungkan ke inverter NOT agar sinyal kembali ke posisi semula sesuai polaritas input. Dengan demikian *output* yang dihasilkan akan membentuk gelombang kotak persegi.

4.4.4. Fuel Pump



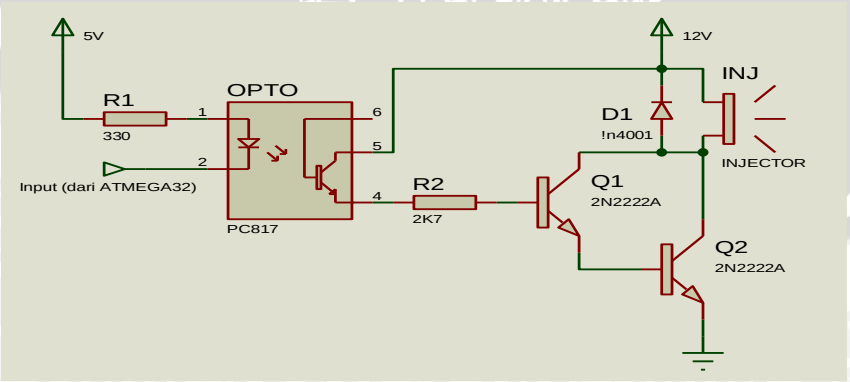
Gambar 4.5. *Fuel Pump*

Bahan bakar dari tangki akan dialirkan menggunakan pompa kemudian akan disemprotkan dengan injektor. Pompa ini bekerja dengan tegangan 12 V. kecepatan minimum aliran pompa bahan bakar adalah 82cm³/10detik.

4.4.5. Perancangan Injektor

Gambar 4.6. Injektor

Injektor berfungsi sebagai aktuator dengan menerima informasi dari ECU untuk menyemprotkan jumlah bahan bakar bensin ke dalam ruang bakar. Injektor ini bekerja pada 9 - 12 Volt. Jumlah bahan bakar yang disemprotkan oleh injektor berdasarkan putaran mesin dan bukaan throttle. Semakin besar bukaan throttle maka durasi injektor semakin panjang sehingga putaran mesin semakin bertambah.

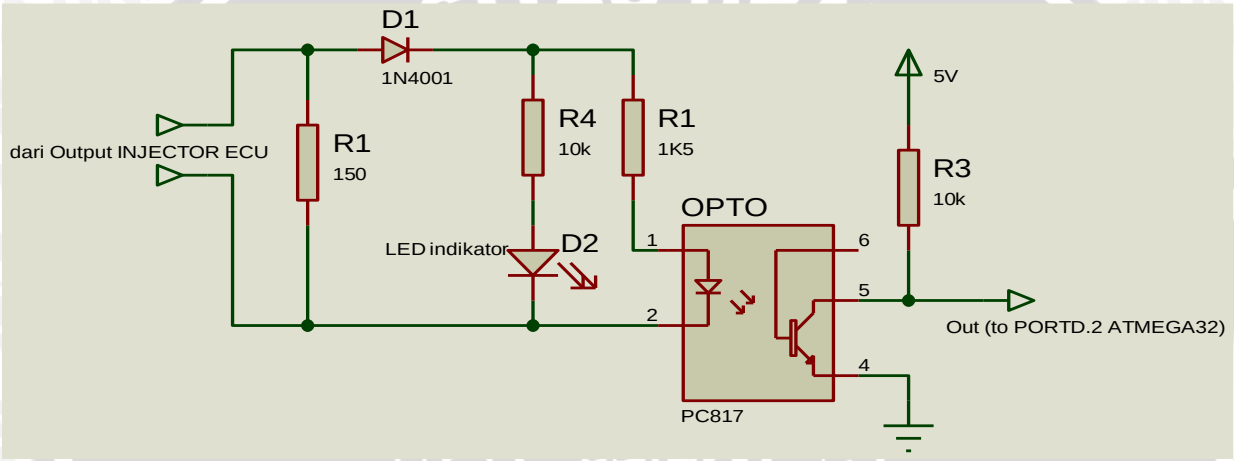


Gambar 4.7. Rangkaian *driver* injektor

Pada Gambar 4.6 merupakan rangkaian *driver* injektor, dimana dalam rangkaian tersebut menggunakan optocoupler, dioda, resistor, dan transistor. Optocoupler disini berfungsi sebagai pemisah antara rangkaian power dan rangkaian kontrol. Untuk memperkuat sinyal maka dalam rangkaian menggunakan transistor 2N222A.

4.4.6. Perancangan Rangkaian Detektor Injektor

Rangkaian detektor injektor ini berfungsi untuk mendeteksi tegangan yang keluar dari ECU sepeda motor agar ECU dapat tetap bekerja dan sebagai penentu awal aktifnya injektor dari ECU.

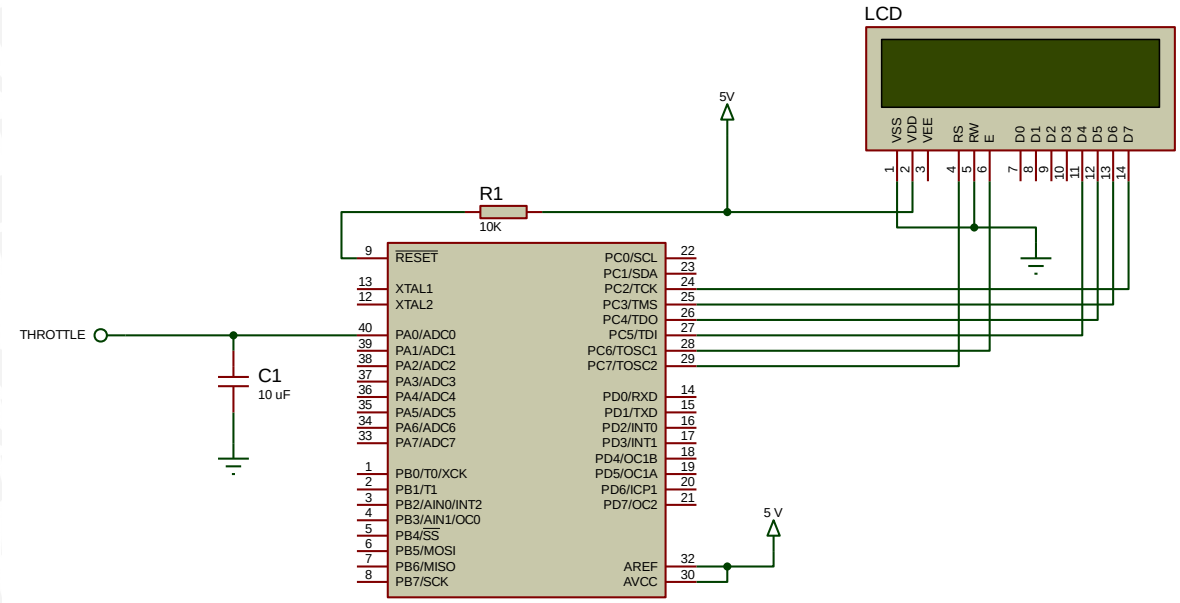


Gambar 4.8. Rangkaian sensor injektor

Dari Gambar 4.7. dapat diketahui bahwa saat ECU mengeluarkan tegangan untuk injector, maka tegangan akan tebaca pada R1 (150Ω) sebagai pengganti beban (injector), selanjutnya arus masuk melalui diode kemudian menuju R2 dan mengaktifkan led didalam optocoupler, karena led optocoupler terpicu, maka output optocoupler akan berlogika low (output = VCEsat) sedangkan saat saat input dari detector tidak bertegangan (OFF), maka output dari optocoupler menjadi high

4.4.7. Perancangan Mikrokontroler

Pada perancangan mikrokontroler menggunakan ATmega32 sebagai pengolah utama untuk pemrosesan algoritma pada sistem keseluruhan. Konfigurasi kaki I/O dari modul ATmega32 dapat dilihat pada gambar di bawah.

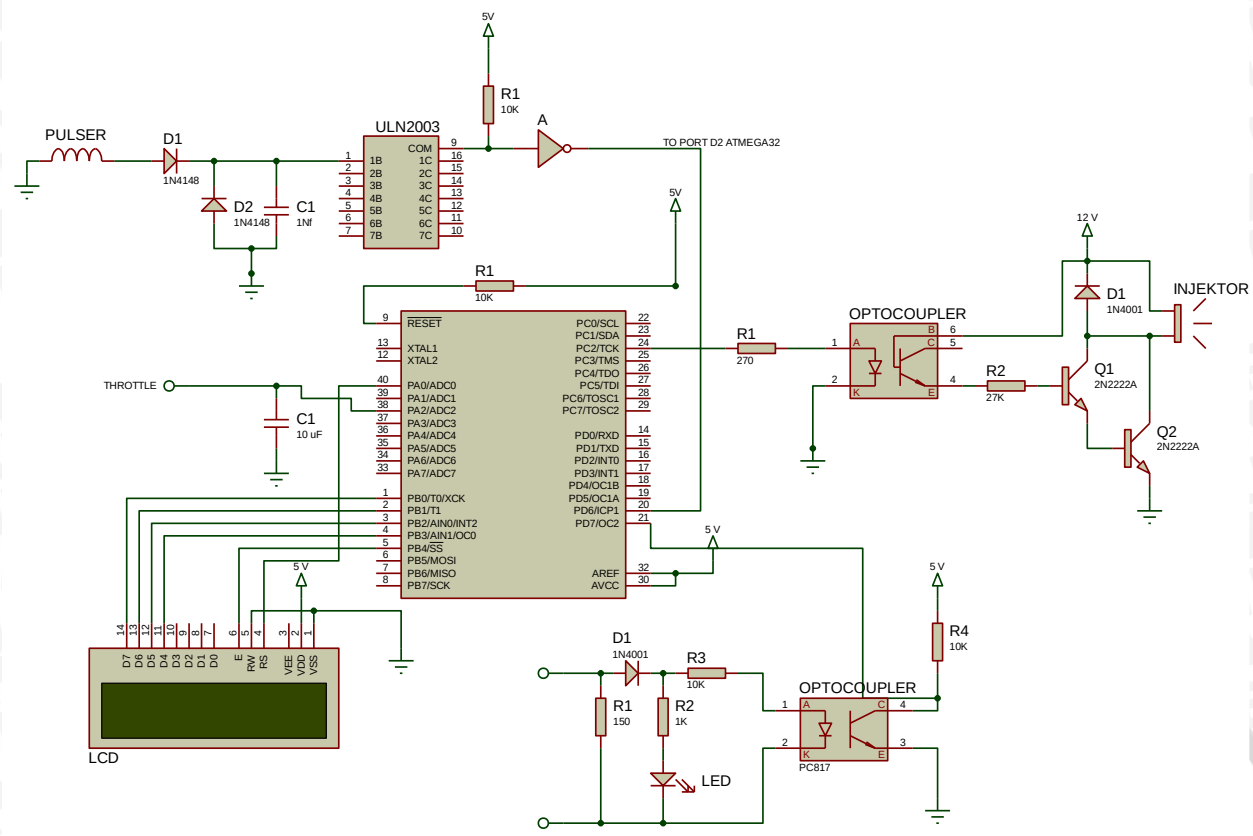


Gambar 4.9. Rangkaian Mikrokontroler ATmega32

Pada gambar 4.8 menampilkan rangkaian untuk menampilkan hasil menggunakan lcd. Setiap pin memiliki fungsi masing-masing seperti yang tertera pada datasheet dari ATmega32.

4.4.8. Perancangan Keseluruhan

Perancangan sistem keseluruhan merupakan gabungan dari perancangan per blok diagram sistem menjadi suatu kesatuan yang dapat berfungsi sesuai dengan tujuan dari penelitian ini. Perancangan rangkaian sistem secara keseluruhan ini digambar menggunakan software Proteus 8. Perancangan sistem secara keseluruhan terdapat dalam Gambar 4.9.



Gambar 4.10. Rangkaian keseluruhan sistem

Gambar 4.9 merupakan gambar rangkaian dari per blok diagram sistem yang digabungkan menjadi satu sistem. Pada rangkaian sistem terdapat rangkaian driver injeksi, sensor injeksi, rangkaian throttle, dan pengkondisi sinyal pulser.

4.5. Perancangan Perangkat Lunak

Perancangan perangkat lunak terbagi menjadi 2 bagian, yaitu :

1. Perangkat Kontrol Logika Fuzzy (KLF)
2. Perancangan Algoritma

4.5.1. Perancangan Kontrol Logika Fuzzy (KLF)

4.5.1.1. Variabel Masukan dan Keluaran

Pada perancangan skripsi ini, kondisi mesin adalah tidak berbeban, sehingga semakin besar bukaan katup throttle akan menambah rpm *engine*. Perancangan kontroler logika fuzzy didasari dari data hubungan variasi bukaan throttle dan kecepatan putaran mesin dengan lebar pulsa injeksi. Sehingga masukan untuk kontroler logika fuzzy adalah error dan

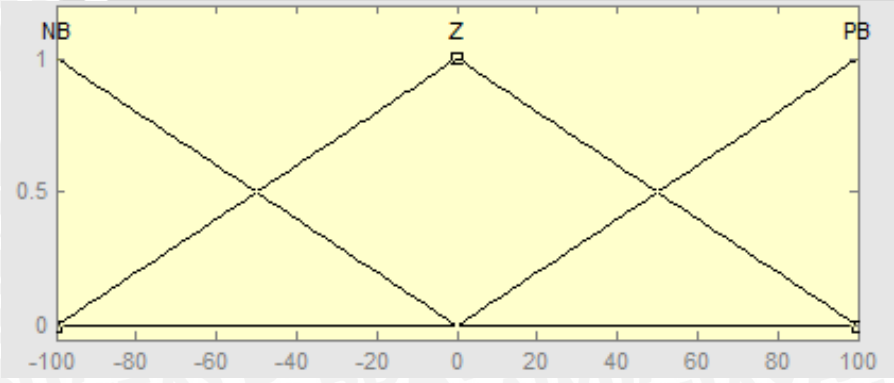
kecepatan referensi, sedangkan keluarannya adalah lebar pulsa injeksi. Data tersebut dicari dengan melakukan eksperimen.

Tabel 4.1. Variasi Posisi Throttle dan Kecepatan Putaran Mesin dengan Lebar Pulsa Injeksi

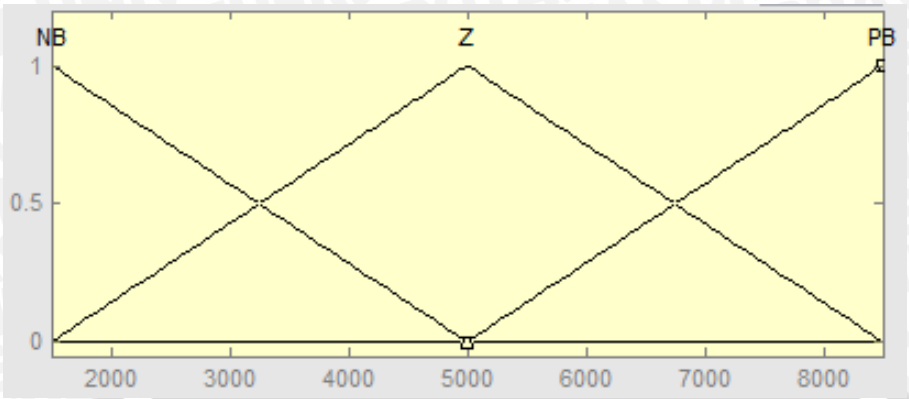
Posisi *Throttle*	Kondisi Injeksi Minimal		Kondisi Injeksi Optimal		Kondisi Injeksi Maksimal	
	LPI (ms)	Kecepatan (rpm)	LPI (ms)	Kecepatan (rpm)	LPI (ms)	Kecepatan (rpm)
0%	1,6	1399	1,8	1500	2,3	1650
10%	2,2	2599	2,5	3000	2,6	3200
20%	2,4	3005	2,5	3400	2,7	3566
30%	2,4	3499	2,7	3700	3,0	3899
40%	2,6	4066	2,8	4500	2,9	4650
50%	2,6	4599	2,9	4700	3,2	4830
60%	3,3	5002	3,5	5100	3,6	5340
70%	3,7	5966	3,8	6200	3,9	6299
80%	3,7	7152	3,9	7500	4,2	7705
90%	4,0	7900	4,3	8200	4,4	8366
100%	4,3	8155	4,5	8500	4,7	8900

4.5.1.2. Fungsi Keanggotaan Masukan dan Keluaran

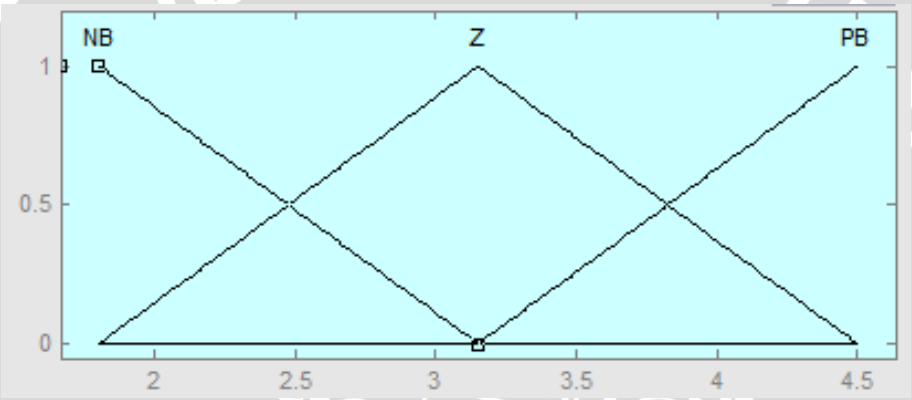
Fungsi keanggotaan dari masukan dan keluaran terdiri dari tiga label, yaitu *Negative Big* (NB), *Zero* (Z), dan *Positive Big* (PB). Jenis fungsi keanggotaan yaitu segitiga sama kaki dan digunakan dua variabel masukan yaitu nilai error dan nilai kecepatan referensi. Keluaran dari kontroler fuzzy adalah lebar pulsa injeksi menggunakan jenis keanggotaan segitiga sama kaki dan trapesium.



Gambar 4.11. *Membership Function Input Error*



Gambar 4.12. *Membership Function Input* Kecepatan Referensi



Gambar 4.13. *Membership Function Output* Lebar Pulsa Injeksi

4.5.1.3. Perancangan Aturan *Fuzzy*

Aturan fuzzy (rule) digunakan sebagai penentu keluaran dari fuzzifikasi yang akan diolah dalam proses defuzzifikasi dengan jumlah 9 macam *rule*. *Rule* tersebut didapat berdasarkan hasil percobaan pada Tabel 4.1 dengan 3 buah label dalam fungsi keanggotaan input error dan kecepatan referensi. *Fuzzy rule* yang direncanakan terdapat dalam Tabel 4.2.

Tabel 4.2. Rule Base Fuzzy

E / Ref	NB	Z	PB
NB	NB	NB	Z
Z	NB	Z	PB

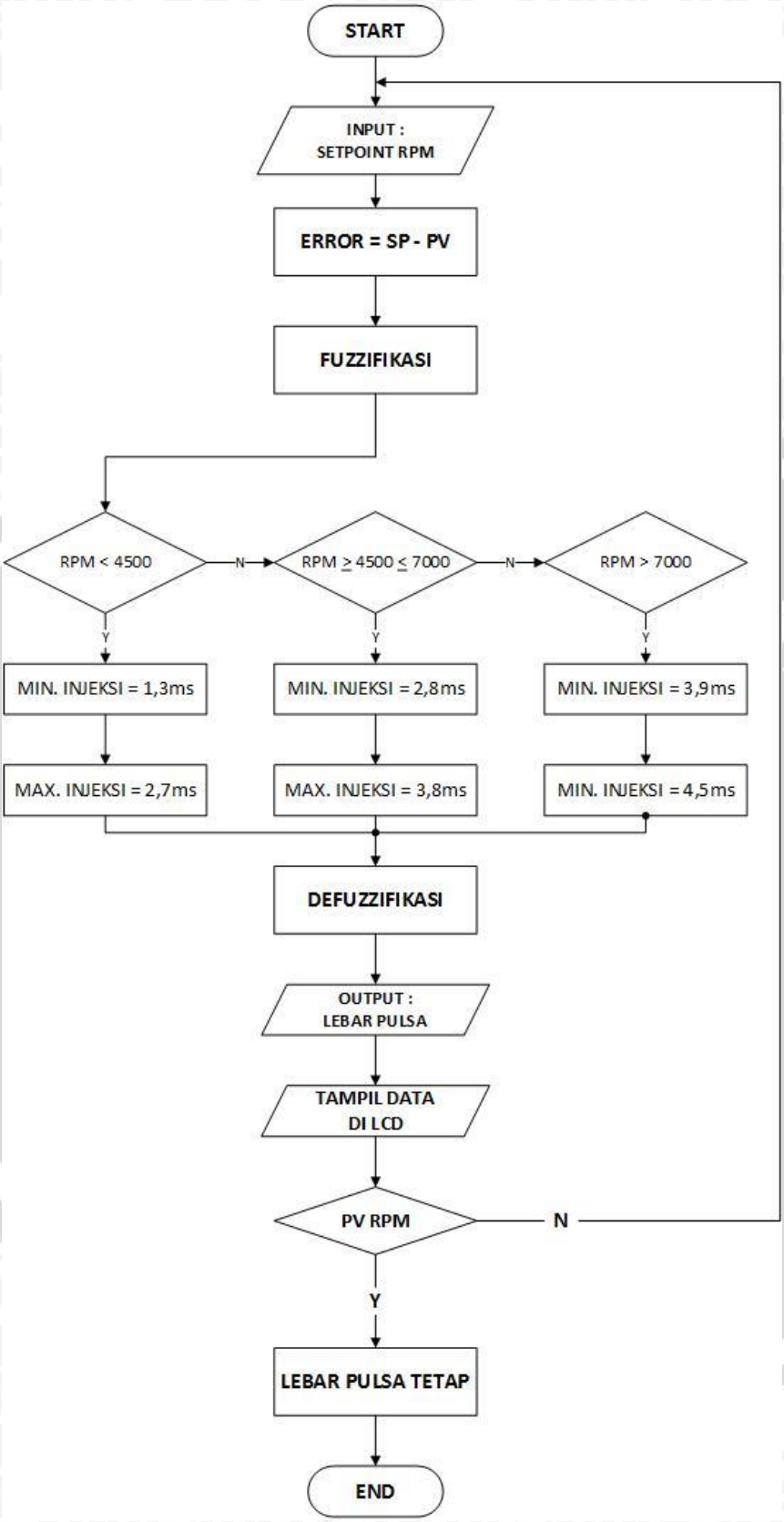
PB	Z	PB	PB

4.5.1.4. Metode Inferensi dan Defuzzifikasi

Metode inferensi fuzzy yang direncanakan menggunakan inferensi *Max-Min*. Setelah dari *input crisp* diubah menjadi *fuzzy* melalui proses fuzzifikasi, maka untuk melihat keluaran dari KLF ini nilai-nilai *fuzzy* tersebut harus diubah menjadi bentuk *crisp* lagi menggunakan proses yang disebut defuzzifikasi. Hasil dari defuzzifiksi ini akan digunakan sebagai pengatur lebar pulsa injeksi. Metode defuzzifikasi yang direncanakan adalah *Center of Area (*COA).

4.5.2. Perancangan Algoritma

Gambar 4.13 merupakan *flowchart* sistem secara keseluruhan yang juga berkaitan dengan *flowchart* program keseluruhan.



Gambar 4.14. *Flowchart* Sistem Keseluruhan

BAB V
PENGUJIAN DAN ANALISA

Pengujian dan analisis dilakukan untuk mengetahui apakah sistem yang telah dirancang di bab sebelumnya mampu bekerja sesuai perancangan atau tidak. Pengujian dilakukan pada setiap blok sistem dan pengujian keseluruhan. Pengujian yang dilakukan antara lain :

1. Pengujian *Output Throttle*
2. Pengujian Pengkondisi Sinyal Sensor CKP
3. Pengujian *Driver* Injektor
4. Pengujian Rangkaian Detektor Injektor
5. Pengujian Sistem Keseluruhan

5.1. Pengujian *Output Throttle*

5.1.1. Tujuan

Pengujian *output throttle* bertujuan untuk mengetahui hasil tegangan dari perubahan posisi sensor *throttle* yang dibaca oleh mikrokontroler melalui ADC agar dapat diolah perangkat lunak.

5.1.2. Peralatan yang Digunakan

1. *Power Supply* 12 V
2. Multimeter
3. Rangkaian AVR ATmega32, LCD dan software
4. Sensor TPS

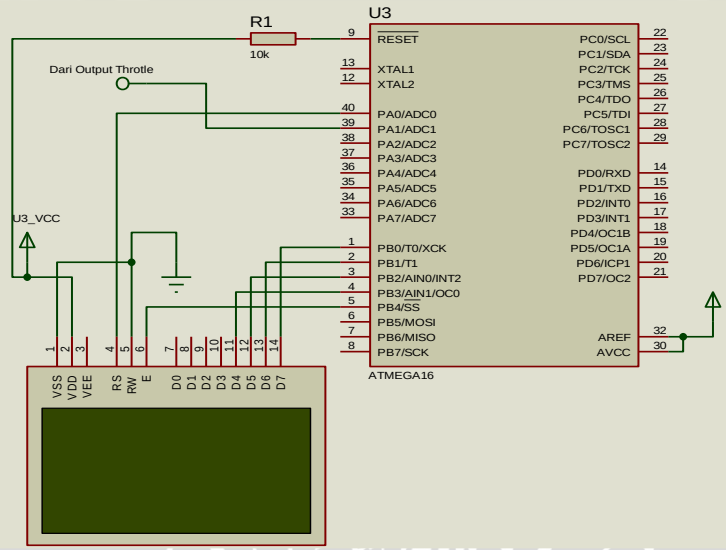
5.1.3. Langkah Pengujian

Adapun langkah-langkah pengujiannya adalah sebagai berikut :

1. Hubungkan *output throttle* pada *input* ADC ATMEGA32
2. *Download* perangkat lunak pada ATMEGA32
3. Nyalakan *power supply* dan posisikan kontak sepeda motor pada posisi ON
4. Atur putaran *throttle* dan ukur *output throttle* pada multimeter dan hasil pada LCD

5.1.4. Diagram Pengujian Rangkaian

Dari hasil percobaan dan pengujian secara manual, *throttle* bekerja sebagaimana layaknya potensiometer yang membentuk rangkaian pembagi tegangan, dimana pada ujungnya terdapat Vcc 5V dan *Ground* pada ujung lainnya, sementara *output* berkisar mulai 0,4 hingga 4,4V. Dengan demikian maka *output* dapat langsung dihubungkan ke *input* ADC ATMEGA32 karena masih berada pada *range* ADC yaitu dibawah 5V. Adapun rangkaian pengujian *throttle* yang digunakan ditunjukkan pada Gambar 5.1.



Gambar 5.1. Rangkaian Pengujian *Throttle* via ADC

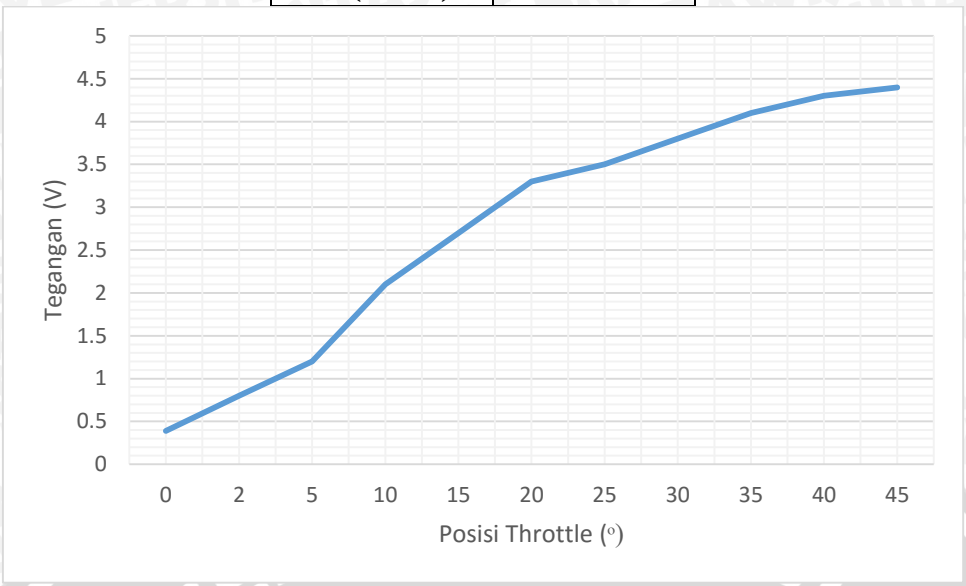
5.1.5. Hasil Pengujian

Hasil pembacaan sensor *throttle* menggunakan ADC dengan merubah step derajat putaran *throttle* pada beberapa posisi ditunjukkan pada Tabel 5.1.

Tabel 5.1. Pengujian *Throttle*

Posisi *Throttle*	**V_{Out} *Throttle***
0^o (idle)	0,39 V
2^o	0,8 V
5^o	1,2 V
10^o	2,1 V
15^o	2,7 V
20^o	3,3 V
25^o	3,5 V
30^o	3,8 V
35^o	4,1V
40^o	4,3V

45 (FULL)	4,4V



Gambar 5.2. Pengujian *Output Throttle*

5.1.6. Analisa

Berdasarkan data hasil pengujian didapat perubahan output yang terbaca melalui multimeter maupun hasil pengolahan ADC yang ditampilkan pada LCD. Dari hasil ini diketahui bahwa perubahan posisi *throttle* dapat diketahui berdasarkan perubahan tegangan pada sensor *throttle* dan derajat perubahan *throttle* yang didapat tidak *linear* terhadap tegangan *output*. Namun demikian rangkaian *throttle* dinyatakan bekerja dengan perubahan tegangan yang berkisar dari 0,39V hingga 4,4V.

5.2. Pengujian Rangkaian Pengkondisi Sinyal Sensor CKP

5.2.1. Tujuan

Untuk mengetahui respon dan sinyal keluaran sensor terhadap pulsa *input* yang berasal dari pulser serta hasil perhitungan perangkat lunak dalam menghitung RPM.

5.2.2. Peralatan yang Digunakan

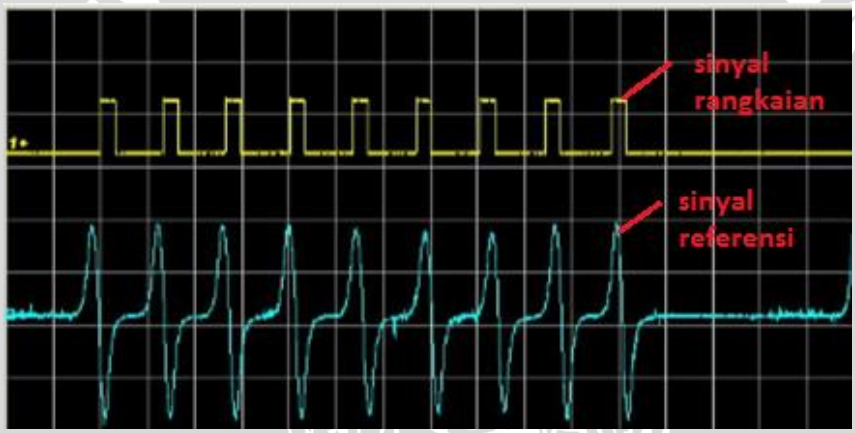
1. Power Supply
2. Rangkaian pengkondisi sinyal sensor pulser.
3. Pulser pada sepeda motor
4. Oscilloscope.

5.2.3. Langkah Pengujian

Adapun langkah-langkah pengujiannya adalah :

1. Hubungkan osciloscope dengan output rangkaian.
2. Hubungkan pulser pada input rangkaian
3. Nyalakan power supply
4. Nyalakan mesin
5. Amati hasil keluaran sinyal pada *osciloscope*.

5.2.4. Hasil Pengujian



Gambar 5.3. Hasil Pengujian Sinyal Referensi dan Rangkaian CKP

5.2.5. Analisa

Hasil pengujian yang dilakukan didapatkan data dalam Gambar 5.3. sinyal referensi dan rangkaian dengan 9 pulsa sesuai dengan jumlah knock pada pulser sehingga dapat disimpulkan bahwa rangkaian pengkondisi sinyal sensor CKP telah sesuai dan dapat bekerja dengan baik.

5.3. Pengujian *Driver* Injektor

5.3.1. Tujuan

Pengujian ini bertujuan untuk mengetahui apakah rangkaian *driver* dapat berfungsi dengan baik untuk mengaktifkan tegangan injektor.

5.3.2. Peralatan yang Digunakan

1. Power Supply 12V
2. Rangkaian driver
3. Injektor

5.3.3. Langkah Pengujian

1. Merangkai rangkaian
2. Mengamati perubahan injektor saat diberikan logika *input*.
3. Mengukur tegangan *output driver*.

5.3.4. Hasil Pengujian

Setelah melakukan prosedur pengujian, didapatkan hasil seperti yang ditunjukkan dalam Tabel 5.2

Tabel 5.2. Pengujian *Driver* Injektor

No	*Input* (PortC.6 ATMEGA32)	Kondisi Injektor	Tegangan
1	0 (GND)	ON (Spray)	11,4 Volt
2	1 (5V/Vcc)	OFF (Diam)	0,3 Volt

5.3.5. Analisa

Berdasarkan data hasil pengujian yang ditunjukkan pada Tabel 5.2. saat *input* diberikan logika *low* (0), maka injektor aktif dan menyemprotkan bahan bakar. Sebaliknya, saat *input high*, injektor tetap off. Dengan demikian, maka pengujian *driver* injektor dinyatakan sesuai dengan perancangan.

5.4 Pengujian Rangkaian Detektor Injektor

5.4.1 Tujuan Pengujian

Untuk mengetahui respon dan sinyal rangkaian detektor terhadap pulsa referensi yang berasal dari *output* injektor ECU.

5.4.2 Peralatan Yang Digunakan

1. Power Supply
2. Rangkaian *detector Injector*.
3. ECU pada sepeda motor

4. *Oscilloscope*.

5.4.3 Langkah-Langkah Pengujian

1. Putus jalur ECU yang menuju injektor, lalu hubungkan dengan input rangkaian
2. Hubungkan *osciloscope* dengan output rangkaian.
3. Nyalakan power supply
4. Nyalakan mesin
5. Amati hasil keluaran sinyal pada *osciloscope*.

5.4.4 Hasil Pengujian

Dari hasil pengujian didapat hasil seperti yang ditunjukkan pada gambar 5.4.

Gambar 5.4. Hasil pengujian detektor injektor

Berdasarkan data hasil pengujian, terjadi penyimpangan antara lebar pulsa rangkaian dan referensi. Berikut merupakan contoh perhitungan *error* pada data ke-1 :

Error (*ms*) = $|Lebar\ pulsa\ rangkaian - Lebar\ pulsa\ referensi|$

= $|2{,}05 - 2{,}03|$

= $0{,}02\ ms$

Error (%) = $\frac{Error}{Lebar\ pulsa\ referensi}\ x\ 100\%$

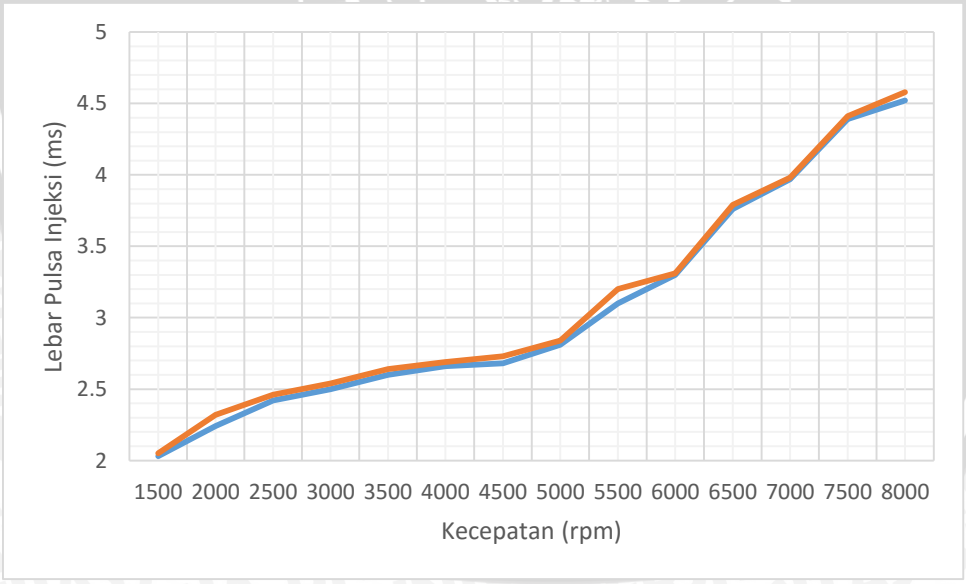
$= \frac{0,02}{2,03} x\ 100\%$

$= 0,98\%$

Hasil *error* pengujian detektor injektor dengan variasi rpm dapat dilihat pada Tabel 5.3.

Tabel 5.3. Hasil Pengujian Detektor Injektor dengan Variasi RPM

No	RPM	Lebar pulsa (ms)		Error (%)
		$T_{referensi}$	$T_{rangkaian}$	
1	1500	2,03	2,05	0,98
2	2000	2,24	2,32	3,57
3	2500	2,42	2,46	1,65
4	3000	2,5	2,54	1,6
5	3500	2,6	2,64	1,53
6	4000	2,66	2,69	1,12
7	4500	2,68	2,73	1,86
8	5000	2,81	2,84	1,06
9	5500	3,1	3,2	3,22
10	6000	3,3	3,31	0,31
11	6500	3,76	3,79	0,79
12	7000	3,97	3,98	0,26
13	7500	4,39	4,41	0,46
14	8000	4,52	4,58	1,32
Rata-rata Error				**1,41**



Gambar 5.5. Hasil Pengujian Detektor Injektor dengan Variasi RPM

5.4.5 Analisa

Berdasarkan hasil pengujian pada Tabel 5.3. diketahui bahwa nilai rata-rata error sebesar 1,41%. Kesalahan tersebut relatif kecil sehingga dapat disimpulkan bahwa rangkaian detektor injektor dapat berjalan dengan baik.

5.5. Pengujian Keseluruhan

5.5.1. Tujuan

Pengujian ini dilakukan untuk mengetahui kinerja *hardware* dan *software* serta untuk mengetahui respon sistem secara keseluruhan dengan menggunakan Kontrol Logika Fuzy (KLF) apakah bias memberikan hasil respon *output* seperti yang diinginkan.

5.5.2. Peralatan yang Digunakan

1. Power Supply
2. *Throttle Position Sensor*
3. Sensor CKP
4. Rangkaian Pengkondisi Sinyal Sensor CKP
5. Injektor
6. *Driver* Injektor
7. Rangkaian Detektor Injektor
8. Rangkaian LCD
9. Mikrokontroler ATmega32
10. Laptop

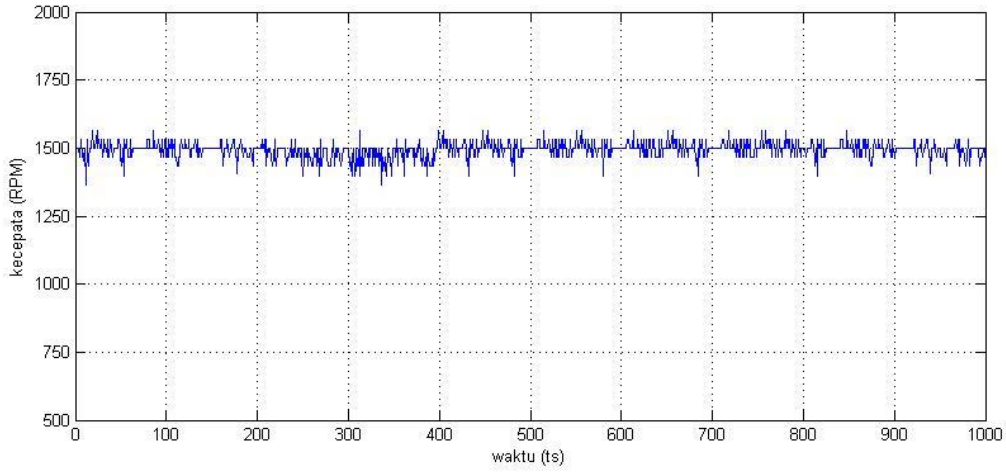
5.5.3. Prosedur Pengujian

1. Mengunduh program melalui software BASCOM AVR.
2. Menampilkan dan merekam nilai posisi *throttle* dan rpm *engine* pada *HyperTerminal software* untuk data serial dan LCD.
3. Membuat grafik dari data yang didapat.

5.5.4. Hasil Pengujian

Dari hasil pengujian keseluruhan dengan menggunakan 3 fungsi keanggotaan masukan dan keluaran, metode inferensi *Max-Min* serta metode defuzifikasi *Center of Area* (COA) didapatkan hasil seperti berikut.

1. Hasil Pengujian pada Posisi *Throttle* 0%



Gambar 5.6. *Output* pada Posisi *Throttle* 0%

Pada Gambar 5.6. ditunjukkan hasil pengujian pada kondisi posisi *throttle* 0%. Saat keadaan *steady state* nilai kecepatan tengah *steady state* ($V_{\text{tengah steady state}}$) 1494,13 rpm dengan kecepatan mula-mula ($V_{\text{mula-mula}}$) adalah 961 rpm, serta kecepatan *setpoint* ($V_{setpoint}$) adalah 1500 rpm, sehingga *error steady state* nya adalah:

$$e_{ss}\ (\%) = \left|\frac{(V\ tengah\ steady\ state) - (V\ setpoint)}{(V\ setpoint) - (V\ mula{-}mula)}\right| x\ 100\%$$

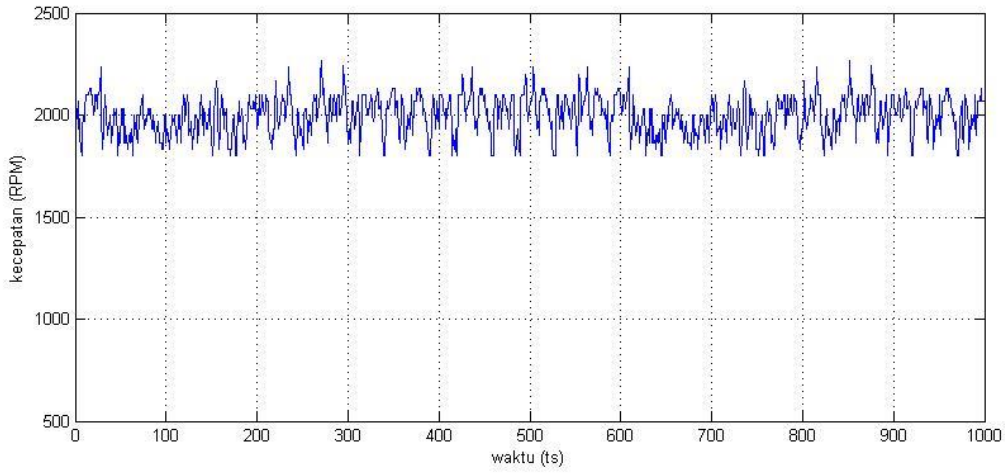
$$= \left|\frac{1494{,}13-1500}{1500-961}\right| x\ 100\%$$

$$= 1{,}08\%$$

Keterangan :

- e_{ss} : *Error Steady State*
- $V_{\text{tengah } steady\ state}$: kecepatan rata-rata dari waktu ke-2 sampai dengan waktu ke-n
- $V_{\text{mula-mula}}$: kecepatan saat waktu pertama atau waktu ke-0
- $V_{setpoint}$: kecepatan yang diinginkan yaitu 1500 rpm

2. Hasil Pengujian pada Posisi *Throttle* 5%



Gambar 5.7. *Output* pada Posisi *Throttle* 5%

Pada Gambar 5.7. ditunjukkan hasil pengujian pada kondisi posisi *throttle* 5%. Saat keadaan *steady state* nilai kecepatan tengah *steady state* ($V_{\text{tengah } steady\ state}$) 1998,54 rpm dengan kecepatan mula-mula ($V_{\text{mula-mula}}$) adalah 1502 rpm, serta kecepatan *setpoint* ($V_{setpoint}$) adalah 2000 rpm, sehingga *error steady state* nya adalah:

$$e_{ss}\ (\%) = \left|\frac{(V\ tengah\ steady\ state) - (V\ setpoint)}{(V\ setpoint) - (V\ mula{-}mula)}\right| x\ 100\%$$

$$= \left|\frac{1998{,}54-2000}{2000-1502}\right| x\ 100\%$$

= 0,29%

Keterangan :

- e_{ss} : *Error Steady State*
- $V_{\text{tengah } steady\ state}$: kecepatan rata-rata dari waktu ke-2 sampai dengan waktu ke-n
- $V_{\text{mula-mula}}$: kecepatan saat waktu pertama atau waktu ke-0
- $V_{setpoint}$: kecepatan yang diinginkan yaitu 2000 rpm

Pengujian dilakukan pada kondisi posisi *throttle* 0% dan 5% disebabkan ketika kondisi tersebut, sepeda motor tidak mengalami getaran sehingga tidak merubah posisi *throttle* untuk menentukan *setpoint* kecepatan putaran *engine*.

BAB VI

PENUTUP

6.1 Kesimpulan

Penggunaan KLF sebagai pengontrol kecepatan putaran *engine* (rpm) menggunakan 3 label fungsi keanggotaan untuk masukan *error* dan kecepatan referensi, serta 3 label fungsi keanggotaan keluaran lebar pulsa injeksi, dan juga menggunakan metode inferensi *Max-Min*, serta metode defuzzifikasi *Center of Area* dapat mencapai nilai *setpoint* pada posisi *throttle* 0% dan 5% dengn *error steady state* sebesar 1,08% dan 0,29%. Dari hasil tersebut, maka bisa disimpulkan Kontroler Logika *Fuzzy* dapat menjaga suhu sama dengan *setpoint.*

6.2 Saran

Pada penelitian selanjutnya disarankan agar mengembangkan dengan sistem pengapian, hal ini dikarenakan keterkaitan antara sistem injeksi dan pengapian sangat berpengaruh, pengendalian injeksi dan pengapian yang tepat dapat memengaruhi performansi mesin. Selain itu, perlu dikembangkan *membership function* dan *rule base* agar semakin banyak kemungkinan kondisi yang tercipta dan semakin banyak pula aksi kontrol yang akan dipilih.

DAFTAR PUSTAKA

Bolton, W. 2004. *Sistem Instrumentasi dan Sistem Kontrol (Judul asli: Instrumentation and Control System)*. Jakarta: Erlangga.

Fahmi, Faizal. 2013. Perancangan dan Unjuk Kerja *Engine Control Unit* (ECU) iquteche pada Motor Yamaha Vixion. Surabaya: Institut Teknologi Sepuluh November.

Firmansyah, Reza Adin. 2015. Pengendalian Kecepatan Putaran Gas *Engine* pada UAV RC Airplane Menggunakan Kontrol Logika Fuzzy. Skripsi. Tidak dipublikasikan. Malang: Universitas Brawijaya.

Firmansyah, Rifqi. 2011. Perancangan dan Implementasi *Fuzzy Lookup Table* untuk Pengaturan Injeksi Bahan Bakar saat Kecepatan Stasioner pada Mesin *Spark Ignition*. Surabaya: Institut Teknologi Sepuluh November.

Gopal, Dr.M. 1998. *Digital Control Engineering*. New Delhi: New Age International Publisher, Ltd.

Hanief, Nief. http://www.scribd.com/doc/295397859/Bab-11-Sistem-Injeksi#scribd. (diakses pada 20 Oktober 2015).

Kemendikbud. https://belajar.kemdikbud.go.id/SumberBelajar/tampilajar.php?ver=99&idmateri=220&mnu=Materi3. (diakses pada 20 Oktober 2015)

Lee, S.H., Howlett, R.J., and Walters, S.D. 2004. *Engine Fuel Injection Control Using Fuzzy Logic*. Brighton: University of Brighton.

Motogokil. Belajar ECU. http://motogokil.com/2013/06/03/belajar-ecu-bag-1-ecu-ternyata-benar-sesuatu-yang-menakutkan-bagi-mekanik-sepeda-motor/. (diakses pada 20 Oktober 2015).

Muslim, Muhammad Aziz, Nusantoro, Goegoes Dwi, Kurniawan, Dwi Fadilla. 2014. Peningkatan Efisiensi Engine Melalui Penerapan Kecerdasan Buatan Pada *Engine Control Unit* (ECU). Malang: Universitas Brawijaya.

Noorizky, Yudhanto Iman. 2016. Desain dan Implementasi Kontrol Logika *Fuzzy* Terhadap Pengaturan Injeksi Berdasarkan Gas Hasil Sisa Pembakaran pada Motor Bensin 4-Langkah. Skripsi. Tidak dipublikasikan. Malang: Universitas Brawijaya.

Ogata, Katsuhiko. 1985. *Teknik Kontrol Automatik (Sistem Pengaturan) Jilid 1*. Jakarta: Erlangga.

Philip, C. L. & Harbor, R. D. 1996. *Feedback Control System*. Diterjemahkan oleh Widodo.R.J. New Jersey: Prentice Hall.

Puthut, Tri Wahyudi. 2012. Desain dan Implementasi Kontroler Prediktif Logika *Fuzzy* untuk Pengaturan Injeksi Bahan Bakar *Ignition Engine*. Surabaya : Institut Teknologi Sepuluh November.

Setiyono, Slamet. https://www.slideshare.net/SlametSetiyono/sisteminjeksi. (diakses pada 20 Oktober 2015)

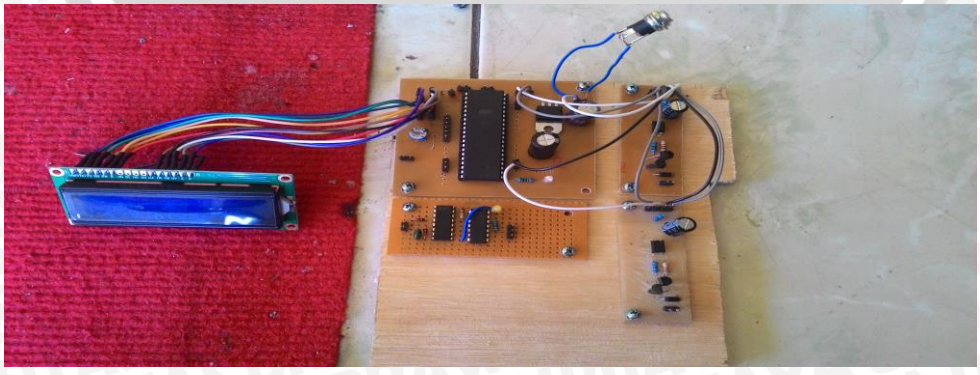
Syahrul. 2014. Pemrograman Mikrokontroler AVR Bahasa Assembly dan C. Bandung.

Yan, J., Ryan, M., dan Power, J. 1993. *Using Fuzzy Logic*. NewYork: Prentice Hall.

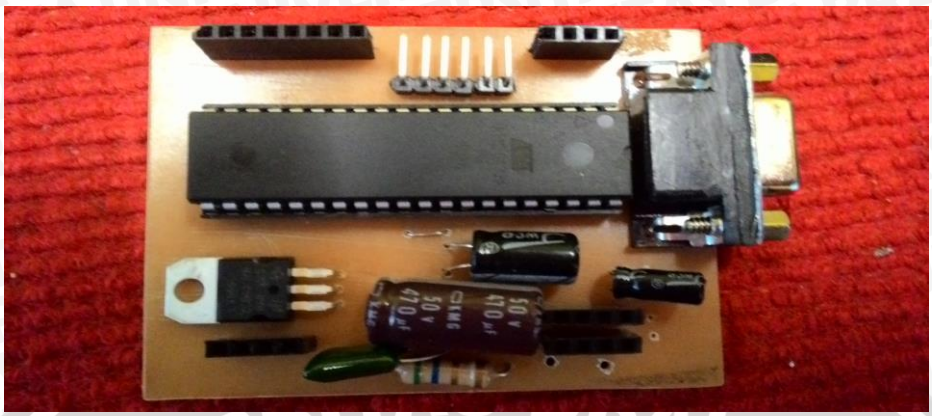
LAMPIRAN

Injektor

Pemasangan Probe pada Injektor



RPS CKP dan *Driver* Injektor



Rangkaian ATmega32

Busi

Osiloskop Siglent

Pemasangan Probe Keseluruhan

Pengambilan Data

Listing Program

```
'$sim

$regfile = "M32def.dat"

$crystal = 12000000

'$crystal = 8000000

$hwstack = 62                                                 '
default use 32 for the hardware stack

$swstack = 20                                                 '
default use 10 for the SW stack

$framesize = 50                                               '
default use 40 for the frame space



$baud = 9600


Declare Function Konversi_ke_tcnt(byval Nilai As Single) As Integer

Declare Function Set_rpm_by_throtle(byval Posisi As Single) As
Integer


Const Vcc = 5

Const Max_adc = 1024
```

```
Const Rl = 10000

Const Min_rpm = 1400

Const Max_rpm = 9000


Const Resolusi_timer2 = 0.00008533


'Const Nb = -100

'Const Ns = -0.5

'Const Zero = 0

'Const Ps = 0.5

'Const Pb = 1


Const Nb = -100

Const Ns = -50

Const Zero = 0

Const Ps = 50

Const Pb = 100


Burn Alias 1

Inject Alias 0


Injector Alias Portc.2

Fuel_pump Alias Portc.3

Ignition Alias Portd.4


True Alias 1

False Alias 0
```

```
Set Injector

Set Fuel_pump

Set Ignition


Indikator Alias Portc.0

Config Indikator = Output

Config Injector = Output

Config Fuel_pump = Output

Config Ignition = Output


Dim Hitungan As Integer


Config Lcdpin = Pin , Db4 = Portb.3 , Db5 = Portb.2 , Db6 = Portb.1
, Db7 = Portb.0 , E = Portb.4 , Rs = Porta.0


Config Lcd = 16 * 2
'configure lcd screen

Config Portd.2 = Input


Config Adc = Single , Prescaler = Auto

Start Adc


Config Timer0 = Timer , Prescale = 1024

Stop Timer0

Config Timer1 = Timer , Prescale = 256


Config Timer2 = Timer , Prescale = 1024
```

```
Stop Timer2


Tcnt1h = &H0B

Tcnt1l = &HDC

On Ovf1 Tim1_isr

On Ovf2 Tim2_isr


Config Int0 = Falling

Enable Int0                                          'enable
the interrupt

On Int0 Interupt                                     'jump to
label2 on INT0


Config Int1 = Falling

Enable Int1                                          'enable
the interrupt

On Int1 Pulse_inject                                 'jump to
label2 on INT0

Config Portd.3 = Input


Dim Rpm As Single , Tempr As Single , Pulser As Single , K As
Integer

Dim Drpm As Integer , Pulsa As Long , Pulsa_low As Long , Pulsa_high
As Long

Dim Periode As Bit , First_start As Bit , Starting As Bit

Dim T_high As Single , Counter_pulse As Integer , Count As Integer

Dim Temp As Single , Vout_sensor As Single , I As Integer , Data_adc
As Word

Dim Posisi_throtle As Single , Minscope As Integer , Maxscope As
Integer
```

```
Dim Set_rpm As Integer


Dim De As Single , A As Byte

Dim Nbe As Single , Nme As Single , Zee As Single , Pme As Single ,
Pbe As Single

Dim Nbd As Single , Nmd As Single , Zed As Single , Pmd As Single ,
Pbd As Single

Dim Nbo As Word , Nmo As Word , Zeo As Word , Pmo As Word , Pbo As
Word

Dim Nbc(10) As Word , Nmc(10) As Word , Zec(10) As Word , Pmc(10) As
Word , Pbc(10) As Word

Dim Z(5) As Word , J As Word

Dim Kp As Single

Dim E As Single , Elst As Single , Flag As String * 2

Dim Setpoint As Single , Pv As Single , Selisih As Integer

Dim Kalkulasi As Single , Timer_injector As Integer


Cls

Cursor Off Noblink

Reset Indikator

Wait 1

Enable Timer1                                          ' enable
the timer interrupt

Enable Timer2                                          ' enable
the timer interrupt


Dim Data_throtle(16) As Single

Dim Data_rpm(16)as Integer
```

```
Data_throtle(1) = 2.85

Data_throtle(2) = 3.3

Data_throtle(3) = 4

Data_throtle(4) = 4.5

Data_throtle(5) = 5

Data_throtle(6) = 5.5

Data_throtle(7) = 6

Data_throtle(8) = 9

Data_throtle(9) = 10

Data_throtle(10) = 10.5

Data_throtle(11) = 11

Data_throtle(12) = 12

Data_throtle(13) = 15

Data_throtle(14) = 21

Data_throtle(15) = 25

Data_throtle(16) = 94


Data_rpm(1) = Min_rpm

Data_rpm(2) = 2000

Data_rpm(3) = 2500

Data_rpm(4) = 3000

Data_rpm(5) = 3500

Data_rpm(6) = 4000

Data_rpm(7) = 4500

Data_rpm(8) = 5000

Data_rpm(9) = 5500

Data_rpm(10) = 6000
```

```
Data_rpm(11) = 6500
Data_rpm(12) = 7000
Data_rpm(13) = 7500
Data_rpm(14) = 8000
Data_rpm(15) = 8500
Data_rpm(16) = Max_rpm


Pulsa_low = 1000
Pulsa_high = 0
Pulsa = 0
Enable Interrupts
Start Timer1
Tcnt0 = &H0
First_start = True
Starting = True
Tcnt2 = &HEA
Cls
Lcd "Ready"
Periode = Inject
Counter_pulse = 0
Reset Ignition
Periode = Burn
Count = 1


Do
```

```
    Gosub Baca_trotle
    If Count = 100 Then
       Locate 1 , 1
       Lcd "Set RPM:" ; Set_rpm ; "  "
       Locate 2 , 1
       Lcd "k:" ; Kp ; "  "
       Count = 0 .
    End If
    Incr Count
Loop

Actuator_inject:
   If Kp < 1 Then                                    'RPM <
set
      Selisih = Minscope - Maxscope
'      Print "selisih:" ; Selisih
      Tempr = Kp
      Kalkulasi = Tempr / 1
'     Print "kp /1:" ; Kalkulasi
      Kalkulasi = Kalkulasi * Selisih
'     Print "x selisih:" ; Kalkulasi
      Tempr = Minscope - Kalkulasi
   '   Print "MIN KP:" ; Tempr
   Elseif Kp > 1 Then
   If Kp > 2 Then Kp = 2
      Selisih = Minscope - Maxscope
'      Print "selisih:" ; Selisih
      Tempr = Kp - 1
```

```
        Kalkulasi = Tempr / 1
'       Print "kp /1:" ; Kalkulasi
        Kalkulasi = Kalkulasi * Selisih
'       Print "x selisih:" ; Kalkulasi
        Tempr = Kalkulasi + Maxscope
'       Print "MIN KP:" ; Tempr
    End If
    Timer_injector = Int(tempr)
Return

Baca_trotle:
    Temp = 0
    For I = 1 To 10
        Data_adc = Getadc(1)
        Temp = Temp + Data_adc
'       Waitms 1
    Next
    Vout_sensor = Temp / 10
    'koversi ke tegangan Vin ADC
    Vout_sensor = Vout_sensor * Vcc
    Vout_sensor = Vout_sensor / Max_adc
    Temp = Vout_sensor - 0.3
    Temp = Temp / 4.2
    Posisi_throtle = Temp * 100
Return
```

```
'the following code is executed when the timer rolls over
Tim1_isr:
   Stop Timer1
   Rpm = Pulser / 9                                        'jumlah
triger 9 step
   Rpm = Rpm * 5                                           '1detik
putaran (200ms timer x 5 = 1detik)
   Rpm = Rpm * 60                                          '1menit
putaran
   Tcnt1 = &HDB61
   Start Timer1
   Pulser = 0
Return

Tim2_isr:
   Stop Timer2
   Set Injector
   Set Indikator
Return

Pulse_inject:
   Pulsa = 3
   Reset Injector
   Disable Int1
   Reset Indikator
   Tcnt2 = Timer_injector
      Start Timer2
   Enable Int1
```

```
Return
'generates a RET because it is the second RETURN


Interupt:
   Disable Int0
   If Pulsa = 8 Then
   Set_rpm = Set_rpm_by_throtle(posisi_throtle)
   If Set_rpm >= 1500 And Set_rpm < 4500 Then
      Minscope = Konversi_ke_tcnt(1.8)
      Maxscope = Konversi_ke_tcnt(2.7)
   Elseif Set_rpm >= 4500 And Set_rpm < 7000 Then
      Minscope = Konversi_ke_tcnt(2.8)
      Maxscope = Konversi_ke_tcnt(3.8)
   Elseif Set_rpm >= 7000 And Set_rpm <= 10000 Then
      Minscope = Konversi_ke_tcnt(3.9)
      Maxscope = Konversi_ke_tcnt(4.5)
   End If
   Setpoint = Set_rpm
      Gosub Baca_pv
      Gosub Member_error
      Gosub Member_del_error
      Gosub Rule_base
      Gosub Implikasi
      Gosub Defusifikasi
      Gosub Actuator_inject
      Elst = E                                          'error
disimpan sebagai error sebelumnya
   End If
```

```
   Incr Pulsa
   Enable Int0

Return
'generates a RET because it is the second RETURN

End

Function Konversi_ke_tcnt(byval Nilai As Single) As Integer
Local Tempd As Single , Nilai_t As Single
 ' Print Nilai
   Tempd = Nilai / 1000                                    'koversi
ke ms
 '  Print "tmp: " ; Tempd
   Nilai_t = Tempd / Resolusi_timer2                       'data ms
/ resolusi timer 2
'   Print "Nilai:" ; Nilai_t
   Konversi_ke_tcnt = 256 - Int(nilai_t)
'konversi ke nilai TCNT
End Function

Function Set_rpm_by_throtle(byval Posisi As Single) As Integer
Local Sterm1 As Single , Sterm2 As Single , Sterm3 As Single , Hasil
As Single
Local Tempd As Single , Tempd2 As Single

  ' Check throtle terhadap nilai tertinggi dan terrendah RPM
  If Posisi >= Data_throtle(16) Then
```

```
 Set_rpm_by_throtle = Max_rpm
 Exit Function
End If

If Posisi <= Data_throtle(1) Then
 Set_rpm_by_throtle = Min_rpm
 Exit Function
End If

For K = 1 To 16                                          'cek
nilai trhotle terhadap array RPM
  If Posisi <= Data_throtle(k) Then Exit For
Next

If Posisi = Data_throtle(k) Then
  Set_rpm_by_throtle = Data_rpm(k)                       'jika
sama, data array RPM = array Trotle
 Exit Function
End If

Sterm1 = Posisi - Data_throtle(k)

Sterm2 = Data_rpm(k) - Data_rpm(k -1)
Sterm3 = Data_throtle(k) - Data_throtle(k - 1)
Tempd = Sterm1 * Sterm2
Tempd2 = Tempd / Sterm3
Hasil = Data_rpm(k ) + Tempd2
If Hasil > Max_rpm Then Hasil = Max_rpm
```

```
  If Hasil < Min_rpm Then Hasil = Min_rpm
  Set_rpm_by_throtle = Hasil

End Function


'====================baca SP & AV via adc=============
Baca_pv:
   Pv = Rpm
'    Setpoint = Set_point_keypad
   E = Setpoint - Pv                                    '- Sp
   De = E - Elst
Return


'====================program menber input1/error============
Member_error:
If E <= Nb Then
   Nbe = 1
   Nme = 0
   Zee = 0
   Pme = 0
   Pbe = 0
Elseif E <= Ns Then
   'nbe=-(1/0.5)*(e+0.5)
   Nbe = -e
   Nbe = Nbe - 0.5
   Nbe = Nbe * 2
```

```
    'nme=(1/0.5)*(e+1)
    Nme = E + 1
    Nme = Nme * 2
    Zee = 0
    Pme = 0
    Pbe = 0
Elseif E <= Zero Then
    Nbe = 0
    'nme=-(1/0.5)*(e)
    Nme = -e * 2
    'zee=(1/0.5)*(e+0.5)
    Zee = E + 0.5
    Zee = Zee * 2
    Pme = 0
    Pbe = 0
Elseif E <= Ps Then
    Nbe = 0
    Nme = 0
    'zee=-(1/0.5)*(e-0.5)
    Zee = -e
    Zee = Zee + 0.5
    Zee = Zee * 2
    'pme=(1/0.5)*(e)
    Pme = E * 2
    Pbe = 0
Elseif E <= Pb Then
    Nbe = 0
```

```
    Nme = 0
    Zee = 0
    'pme=-(1/0.5)*(e-1)
    Pme = -e
    Pme = Pme + 1
    Pme = Pme * 2
    'pbe=(1/0.5)*(e-0.5)
    Pbe = E - 0.5
    Pbe = Pbe * 2
Else
  Nbe = 0
    Nme = 0
    Zee = 0
    Pme = 0
    Pme = 0
    Pbe = 1
End If
Return

'====================program menber input2/delta error==========
Member_del_error:
If De <= Nb Then
    Nbd = 1
    Nmd = 0
    Zed = 0
    Pmd = 0
    Pbd = 0
```

```
Elseif De <= Ns Then
    'nbe=-(1/0.5)*(de+0.5)
    Nbd = -de
    Nbd = Nbd - 0.5
    Nbd = Nbd * 2
    'nme=(1/0.5)*(de+1)
    Nmd = De + 1
    Nmd = Nmd * 2
    Zed = 0
    Pmd = 0
    Pbd = 0
Elseif De <= Zero Then
    Nbd = 0
    'nme=-(1/0.5)*(de)
    Nmd = -de
    Nmd = Nmd * 2
    'zee=(1/0.5)*(de+0.5)
    Zed = De + 0.5
    Zed = Zed * 2
    Pmd = 0
    Pbd = 0
Elseif De <= Ps Then
    Nbd = 0
    Nmd = 0
    'zee=-(1/0.5)*(de-0.5)
    Zed = -de
    Zed = Zed + 0.5
```

```
    Zed = Zed * 2
    'pme=(1/0.5)*(de)
    Pmd = De * 2
    Pbd = 0
Elseif De <= Pb Then
    Nbd = 0
    Nmd = 0
    Zed = 0
    'pme=-(1/0.5)*(de-1)
    Pmd = -de
    Pmd = Pmd + 1
    Pmd = Pmd * 2
    'pbe=(1/0.5)*(de-0.5)
    Pbd = De - 0.5
    Pbd = Pbd * 2
Else
   Nbd = 0
    Nmd = 0
    Zed = 0
    Pmd = 0
    Pmd = 0
    Pbd = 1
End If
Return

'=======================program rule=======================
Rule_base:
```

```
'Nbe

'1

If Nbe < Nbd Then

      Nbc(1) = Nbe * 100

Else

      Nbc(1) = Nbd * 100

End If

'2

If Nbe < Nmd Then

      Nbc(2) = Nbe * 100

Else

      Nbc(2) = Nmd * 100

End If

'3

If Nbe < Zed Then

      Nmc(1) = Nbe * 100

Else

      Nmc(1) = Zed * 100

End If

'4

If Nbe < Pmd Then

      Nmc(2) = Nbe * 100

Else

      Nmc(2) = Pmd * 100

End If

'5

If Nbe < Pbd Then
```

```
        Zec(1) = Nbe * 100
Else
        Zec(1) = Pbd * 100
End If

'Nme
'1
If Nme < Nbd Then
        Nbc(3) = Nme * 100
Else

        Nbc(3) = Nbd * 100
End If
'2
If Nme < Nmd Then
        Nmc(3) = Nme * 100
Else
        Nmc(3) = Nmd * 100
End If
'3
If Nme < Zed Then
        Nmc(4) = Nme * 100
Else
        Nmc(4) = Zed * 100
End If
'4
If Nme < Pmd Then
```

```
        Zec(2) = Nme * 100
Else
        Zec(2) = Pmd * 100
End If
'5
If Nme < Pbd Then
        Pmc(1) = Nme * 100
Else
        Pmc(1) = Pbd * 100
End If

'Zed
'1
If Zee < Nbd Then
        Nmc(5) = Zee * 100
Else
        Nmc(5) = Nbd * 100
End If
'2
If Zee < Nmd Then
        Nmc(6) = Zee * 100
Else
        Nmc(6) = Nmd * 100
End If
'3
If Zee < Zed Then
        Zec(3) = Zee * 100
```

```
Else
      Zec(3) = Zed * 100
End If
'4
If Zee < Pmd Then
      Pmc(2) = Zee * 100
Else
      Pmc(2) = Pmd * 100
End If
'5
If Zee < Pbd Then
      Pmc(3) = Zee * 100
Else
      Pmc(3) = Pbd * 100
End If

'Pme
'1
If Pme < Nbd Then
      Nmc(7) = Pme * 100
Else
      Nmc(7) = Nbd * 100
End If
'2
If Pme < Nmd Then
      Zec(4) = Pme * 100
Else
```

```
        Zec(4) = Nmd * 100
End If
'3
If Pme < Zed Then
        Pmc(4) = Pme * 100
Else
        Pmc(4) = Zed * 100
End If
'4
If Pme < Pmd Then
        Pmc(5) = Pme * 100
Else
        Pmc(5) = Pmd * 100
End If
'5
If Pme < Pbd Then
        Pbc(1) = Pme * 100
Else
        Pbc(1) = Pbd * 100
End If

'Pbe
'1
If Pbe < Nbd Then
        Zec(5) = Pbe * 100
Else
        Zec(5) = Nbd * 100
```

```
End If
'2
If Pbe < Nmd Then
      Pmc(6) = Pbe * 100
Else
      Pmc(6) = Nmd * 100
End If
'3
If Pbe < Zed Then
      Pmc(7) = Pbe * 100
Else
      Pmc(7) = Zed * 100
End If
'4
If Pbe < Pmd Then
      Pbc(2) = Pbe * 100
Else
      Pbc(2) = Pmd * 100
End If
'5
If Pbe < Pbd Then
      Pbc(3) = Pbe * 100
Else
      Pbc(3) = Pbd * 100
End If
Return
```

```
'====================implikasi================================

Implikasi:

Nbo = Max(nbc(1))

Nmo = Max(nmc(1))

Zeo = Max(zec(1))

Pmo = Max(pmc(1))

Pbo = Max(pbc(1))


J = Nbo + Nmo

J = J + Zeo

J = J + Pmo

J = J + Pbo

Return


'====================defuzifikasi============================

Defusifikasi:

Z(1) = Nbo * 0

Z(2) = Nmo * 0.25

Z(3) = Zeo * 0.5

Z(4) = Pmo * 0.75

Z(5) = Pbo * 1


Z(1) = Z(1) + Z(2)

Z(1) = Z(1) + Z(3)

Z(1) = Z(1) + Z(4)

Z(1) = Z(1) + Z(5)

Z(1) = Z(1)
```

```
Kp = Z(1) / J


Return
```