TUGAS AKHIR – TE 141599

IMPLEMENTASI KONTROL PID PADA LENGAN ROBOT UNTUK MENCARI SUMBER GAS MENGGUNAKAN STM32F4

NAFI' ABDUL HAKIM
NRP 2211 100076

Dosen Pembimbing
Dr. Muhammad Rivai, ST., MT.
Rudy Dikairono, ST., MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2015

FINAL PROJECT – TE 141599

PID CONTROL IMPLEMENTATION ON ARM ROBOT FOR GAS TRACKING USING STM32F4

NAFI' ABDUL HAKIM
NRP 2211 100076

Supervisor
Dr. Muhammad Rivai, ST., MT.
Rudy Dikairono, ST., MT.

DEPARTMENT OF ELECTRICAL ENGINEERING
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya 2015

IMPLEMENTASI KONTROL PID PADA LENGAN ROBOT UNTUK MENCARI SUMBER GAS MENGGUNAKAN STM32F4

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan Untuk
Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Elektronika
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopemeber

Menyetujui :

Dosen Pembimbing I,	Dosen Pembimbing II,
Dr. Muhammad Rivai, ST., MT.	**Rudy Dikairono, ST., MT.**
NIP: 196904261994031003	**NIP: 198103252005011002**

SURABAYA
JULI, 2015

IMPLEMENTASI KONTROL PID PADA LENGAN ROBOT UNTUK MENCARI SUMBER GAS MENGGUNAKAN STM32F4

Nama : Nafi' Abdul Hakim
Pembimbing I : Dr. Muhammad Rivai, ST., MT.
Pembimbing II : Rudy Dikairono, ST., MT.

ABSTRAK

Kebakaran adalah salah satu penyebab kematian pekerja pada perusahaan gas berbahaya. Salah satu penyebab kebakaran adalah kebocoran gas. Kebocoran gas berbahaya apabila diabaikan akan banyak menimbulkan bahaya. Lengan robot dengan sensor gas dapat digunakan untuk mendeteksi sumber kebocoran gas. Lengan robot dapat memposisikan sensor gas pada posisi terdekat dengan sumber gas yang dapat dijangkau. Dengan menggunakan sensor gas TGS2600 dengan target gas *ethanol*, maka lengan robot dapat menemukan sumber kebocoran gas. Implementasi kendali *Proposional Derivative Integral* (PID) pada lengan robot akan mengolah masukan dari sensor gas pada prosessor STM32F4 yang akan menghasilkan keluaran data untuk menggerakkan aktuator berupa servo secara otomatis. Pada tugas akhir ini, kendali PID digunakan untuk mengontrol gerakan lengan robot 3 DOF dengan sudut 0^o-180^o pada sumbu horizontal. Jarak jangkauan maksimal lengan robot dapat mendeteksi sumber gas dengan sensor TGS2600 adalah 15cm. Dibutuhkan sebuah kipas penghisap yang diletakkan pada ujung lengan robot di bagian sensor gas untuk membantu sensor mendeteksi dengan lebih fokus. Kendali pada lengan robot yang memiliki nilai error rata-rata rendah ketika mencari sumber gas pada sudut uji 90^o adalah dengan menggunakan konstanta Kp=1, Kd=0,09, Ki=0.0001 dengan masing-masing nilai rata-rata error sebesar 4,401%,1,109%, dan 1,072%.

Kata Kunci : Kebakaran, *PID,* Sensor TGS 2600.

PID CONTROL IMPLEMENTATION ON ARM ROBOT FOR GAS TRACKING USING STM32F4

Name : **Nafi' Abdul Hakim**
1st Advisor : **Dr. Muhammad Rivai, ST., MT.**
2nd Advisor : **Rudy Dikairono, ST., MT.**

ABSTRACT

Fire accident is one of the mortal cause of workers in hazardous gas company. One cause of the fire accident was a gas leak. Gas leak is dangerous if neglected and will cause a lot of danger. Arm robot with gas sensors can be used to detect the source of the gas leak. The robotic arm can move the gas sensor to the closest position of the gas source. By using TGS2600 as the gas sensor and ethanol as the target, then the robot arm can locate the source of the gas leak. Implementation of control Proportional Integral Derivative (PID) on the robotic arm will process the input from the sensors using processor STM32F4 and generate data output to drive the servo actuators automatically. In this research, PID control is used to control the movement of the 3 DOF robot arm with 0o-180o angle on the horizontal axis. Maximum range of the robot arm's detection ability is 15 cm. a suction fan is placed at the end of the robot arm at the gas sensor to help detect sensor with more focus. robot arm's control has a low average error value when looking for the source of the gas at a test angle of 90o with constant Kp = 1, Kd = 0.09, Ki = 0.0001 and the respective average value of error of 4.401%, 1.109%, and 1.072%.

Keyword: *Fire, PID, Sensor Semiconductor TGS 2600.*

KATA PENGANTAR

Puji syukur atas rahmat yang diberikan oleh Tuhan Yang Maha Esa. Karena berkat rahmat dan hidayah-Nya penulis dapat menyelesaikan penelitian ini. Selama pelaksanaan penelitian Tugas Akhir ini, penulis mendapatkan bantuan dari berbagai pihak baik dukungan secara moril maupun materiil. Terima kasih yang sebesar-besarnya penulis sampaikan kepada berbagai pihak yang mendukung dan membantu dalam tugas akhir ini , diantaranya :

1. Kedua orang tua tercinta, Bapak Drs. Amar Ma'ruf dan Ibu Soelijati, S.Pd yang tidak pernah putus untuk seluruh do'a, nasihat, motivasi, dan dukungannya.
2. Dr. Muhammad Rivai, ST., MT. selaku dosen pembimbing pertama, atas bimbingan, inspirasi, pengarahan, dan motivasi yang diberikan selama pengerjaan penelitian tugas akhir ini.
3. Rudy Dikairono, ST., MT. selaku dosen pembimbing kedua, atas bimbingan, inspirasi, pengarahan, dan motivasi yang diberikan selama pengerjaan penelitian tugas akhir ini.
4. Teman-teman asisten Lab B202 yang senantiasa membantu dan memberikan semangat dalam mengerjakan tugas akhir.
5. Teman-teman asisten Lab B402 yang senantiasa membantu dan memberikan semangat dalam mengerjakan tugas akhir.
6. Mas Hani, Mas Sulfan, Mas Anhar yang senantiasa membantu dan memberikan semangat dalam mengerjakan tugas akhir ini.

Penulis sadar bahwa Tugas Akhir ini masih belum sempurna dan masih banyak hal yang perlu diperbaiki. Saran, kritik dan masukan baik dari semua pihak sangat membantu penulis terutama untuk berbagai kemungkinan pengembangan lebih lanjut.

Surabaya,

Nafi' Abdul Hakim
2211100076

DAFTAR ISI

Halaman

HALAMAN JUDUL
PERNYATAAN KEASLIAN TUGAS AKHIR
LEMBAR PENGESAHAN
ABSTRAK i
ABSTRACT iii
KATA PENGANTAR v
DAFTAR ISI vii
DAFTAR GAMBAR xi
DAFTAR TABEL xiii
I. PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Perumusan Masalah 2
1.3 Tujuan Penelitian 2
1.4 Batasan Masalah 2
1.5 Metodologi Penelitian 2
1.6 Sistematika Penulisan 4
1.7 Relevansi 5
II. DASAR TEORI 7
2.1 Sensor Semikonduktor 7
2.1.1 Prinsip Kerja Sensor Gas Tipe Semikonduktor 7
2.1.2 Sensitivitas Terhadap Gas 9
2.1.3 Respons Sensor 10
2.1.4 Rangkaian Pengukuran Dasar 11
2.1.5 Sensor Figaro TGS2600 11
2.2 Komunikasi Serial 12
2.3 Motor Servo 12
2.4 Robot Lengan (Robot Arm) 14
2.4.1 Inverse Kinematic 15
2.5 Voltage Regulator 18
2.6 STM32F4 18
2.7 Servocontroller 20

2.8 PID 23
2.8.1 Kendali Proporsional 24
2.8.2 Kendali Integral 24
2.8.3 Kendali Detivatif 24
2.8.4 Pendekatan Ziegler-Nichlos 25
III. PERANCANGAN SISTEM 27
3.1 Diagram Blok Sistem 27
3.2 Perancangan Perangkat Keras 28
3.2.1 Perangkat Elektrik 28
3.2.1.1 Power supply 29
3.2.1.3 Rangkaian Sensor 30
3.2.1.4 STM32F4 31
3.2.1.5 Servo controller 32
3.2.3 Perangkat lunak 34
3.2.3.1 Proses Kontrol PID 35
IV. PENGUKURAN DAN ANALISIS SISTEM 41
4.1 Pengujian Perangkat Keras 41
4.1.1 Pengujian Supply Kontrol lengan robot tiga DOF 41
4.1.2 Pengujian Rangkaian Sensor 42
4.1.3 Pengujian Servo controller 42
4.1.3.1 Pengujian Vinput Pada servo controller 43
4.1.3.2 Pengujian Output PWM pada servo controller 44
4.1.4 Pengujian sensor gas 46
4.2 Pengujian Seluruh Sistem 47
4.2.1 Pengujian Kp=10 48
4.2.2 Pengujian Kp=10 Kd=1 50
4.2.3 Pengujian Kp=25 Kd=10 52
4.2.4 Pengujian Kp=25 Kd=10 Ki=0.1 55
4.2.5 Pengujian Kp=25 Kd=10 Ki=0.0001 57
4.2.6 Pengujian Gerak Maju 59
4.3 Evaluasi Sistem 61
V. PENUTUP 63
5.1 Kesimpulan 63
5.2 Saran 63

DAFTAR PUSTAKA 65
LAMPIRAN 67
BIODATA PENULIS 85

DAFTAR TABEL

Halaman

Table 2.1 Konstanta Pengali PID Metode Ziegler Nichlos 26
Tabel 4.1 Pengujian Rangkaian Suppy 44
Tabel 4.2 Pengujian Vinput pada servo-controller..... 45
Tabel 4.3 Output PWM servo-controller..... 46
Tabel 4.4 Pengujian Pembacaan ADC Sensor TGS2600..... 48
Tabel 4.4 Rata-Rata Error Pada Setiap Servo 60

DAFTAR GAMBAR

Halaman

Gambar 1.1 Diagram Blok Perancangan Lengan Robot tiga DOF 3
Gambar 2.1 Ilustrasi penyerapan O2 oleh sensor 7
Gambar 2.2 Bentuk penghalang potensial Kristal mikro SnO2. 8
Gambar 2.3 Model penghalang potensial 8
Gambar 2.4 Karakteristik sensitivitas sensor 9
Gambar 2.5 Contoh respons sensor 10
Gambar 2.6 Aksi Awal. 10
Gambar 2.7 Rangkaian Pengukuran dasar sensor 11
Gambar 2.8 Komunikasi serial asinkron 12
Gambar 2.9 Motor servo 13
Gambar 2.10 Pensinyalan Control Servo 13
Gambar 2.11 Diagram blok motorservo 14
Gambar 2.12 Lengan robot 15
Gambar 2.13 Komponen Lengan Robot. 16
Gambar 2.14 Tampak Lengan Robot Samping dan Kanan 17
Gambar 2.15 78xx untuk regulator positif 18
Gambar 2.16 STM32F4 19
Gambar 2.17 Servo Controller XISC 32 20
Gambar 2.18 Konfigurasi Pin XISC 20
Gambar 2.19 Pin Konektor XISC. 22
Gambar 2.20 PID 23
Gambar 2.21 Respons Kurva S 25
Gambar 3.1 Diagram Blok Sistem 28
Gambar 3.2 Keseluruhan Perangkat Keras 29
Gambar 3.3 Rangkaian Power Supply 30
Gambar 3.4 Rangkaian Voltage Divider 31
Gambar 3.5 Rangkaian Supply Board 32
Gambar 3.6 Servo Controller 33
Gambar 3.7 Perancangan Posisi Sensor 34
Gambar 3.8 Sistem Kontrol Lengan Robot 35
Gambar 3.9 Diagram Pembacaan ADC Sensor Gas 36

Gambar 3.10 Inisialisasi Gerakan Scanning..... 37
Gambar 3.11 Inisialisasi ADC Sensor Gas dan Delta Error..... 38
Gambar 3.12 Perhitungan Error[0] dan PID 39
Gambar 3.13 Cek Nilai Delta Error 41
Gambar 4.1 Lengan Robot 3DOF 41
Gambar 4.2 (a) Pengujian Vout 9v, dan (b) Pengujian Vout 9 v 42
Gambar 4.3 Vin Driver Sensor..... 43
Gambar 4.4 (a) Vsupply Logic servocontroller, (b) Vin servo 44
Gambar 4.5 Uji PWM Servocontroller 44
Gambar 4.6 Grafik Pengujian PWM Servo Controller 45
Gambar 4.7 Pengujian Sensor 46
Gambar 4.8 Ilustrasi Pengujian Sensor 47
Gambar 4.9 Tampak Depan Lengan Robot 3DOF 48
Gambar 4.10 Pengujian Servo 0 Dengan Sumber Uji Gas Pada 90 ... 49
Gambar 4.11 Pengujian Servo 1 Dengan Sumber Uji Gas Pada 90 ... 49
Gambar 4.12 Pengujian Servo 2 Dengan Sumber Uji Gas Pada 90 ... 50
Gambar 4.13 Pengujian Servo 0 Dengan Sumber Uji Gas Pada 90 ... 51
Gambar 4.14 Pengujian Servo 1 Dengan Sumber Uji Gas Pada 90 ... 51
Gambar 4.15 Pengujian Servo 2 Dengan Sumber Uji Gas Pada 90 ... 52
Gambar 4.16 Pengujian Servo 0 Dengan Sumber Uji Gas Pada 90 ... 53
Gambar 4.17 Pengujian Servo 1 Dengan Sumber Uji Gas Pada 90 ... 53
Gambar 4.18 Pengujian Servo 2 Dengan Sumber Uji Gas Pada 90 ... 54
Gambar 4.19 Pengujian Servo 0 Dengan Sumber Uji Gas Pada 90 ... 55
Gambar 4.20 Pengujian Servo 1 Dengan Sumber Uji Gas Pada 90 ... 56
Gambar 4.21 Pengujian Servo 2 Dengan Sumber Uji Gas Pada 90 ... 56
Gambar 4.22 Pengujian Servo 0 Dengan Sumber Uji Gas Pada 90 ... 57
Gambar 4.23 Pengujian Servo 1 Dengan Sumber Uji Gas Pada 90 ... 58
Gambar 4.24 Pengujian Servo 2 Dengan Sumber Uji Gas Pada 90 ... 58
Gambar 4.25 Pengujian Servo 0 Dengan Sumber Uji Gas Pada 90 ... 59
Gambar 4.26 Pengujian Servo 1 Dengan Sumber Uji Gas Pada 90 ... 60
Gambar 4.27 Pengujian Servo 2 Dengan Sumber Uji Gas Pada 90 ... 60

BAB I
PENDAHULUAN

1.1 Latar Belakang

Beberapa waktu belakangan ini kebakaran menjadi suatu hal yang serius. Kebakaran berasal dari berbagai macam penyebab, salah satu penyebabnya adalah kebocoran gas yang mudah terbakar dan karena sifatnya yang sulit dideteksi oleh penglihatan dan penciuman manusia. Kecelakaan yang disebabkan oleh adanya kebocoran gas terjadi di hampir seluruh bagian di dunia, salah satunya di Manisa, Turki, dua pekerja tambang tewas, dan tiga korban lainnya dilarikan ke rumah sakit setempat setelah terpapar gas di sebuah tambang di Turki [1]. Selain itu kebocoran gas juga terjadi di sebuah pabrik kimia di Houston, Amerika Serikat. Bocornya gas *methyl* yang dipergunakan untuk membuat pestisida empat pekerjanya. Zat kimia yang digunakan untuk membuat pestisida ini berwujud cair ketika masih berada di penampungan, namun akan seketika berubah menjadi gas yang berbahaya ketika terlepas [2].

Teknologi robot berkembang dengan begitu pesatnya di dunia industri dimana robot sudah menjadi bagian utama dalam proses industri. *Mobile* robot dengan sensor gas hanya dapat memposisikan sensor pada posisi terdekat sumber gas yang dapat dijangkau. Letak sumber gas yang tidak pasti sejajar dengan sensor gas pada *mobile* robot, tidak memungkinkan untuk menemukan letak sumber gas. Untuk menangani permasalahan tersebut, maka diimplementasikan sensor pada lengan robot untuk mencari sumber gas yang dikontrol secara otomatis. Sistem gerak robot ini menggunakan metode *inverse kinematic* dalam melakukan pergerakan lengan robot. Dan metode yang digunakan untuk sistem kendali lengan robotnya untuk mencari sumber gas, digunakan kendali *Proposional Integral Derivatif* (PID).

Ada beberapa alasan menggunakan metode PID pada lengan robot untuk mencari sumber gas, antara lain:

a. Metode ini mudah untuk diaplikasikan untuk pergerakan lengan robot
b. Nilai konstanta pengalinya dapat di *tunning* secara *manual*

Dengan menggunakan kendali PID pada lengan robot dengan STM32F4 proses mencari sumber gas menjadi presisi. Jumlah sensor

gas yang diimplementasikan pada lengan robot adalah dua buah sensor yaitu sensor gas semikonduktor tipe TGS2600 untuk mendeteksi udara yang terkontaminasi.

1.2 Perumusan Masalah

Permasalahan yang dibahas dalam tugas akhir ini adalah :

1. Bagaimana merealisasikan sistem pergerakan lengan robot mendeteksi sumber gas
2. Bagaimana merealisasikan sensor mendeteksi sumber gas
3. Bagaimana sistem lengan robot dengan kendali PID mengikuti sumber gas

1.3 Tujuan Penelitian

Penelitian pada tugas akhir ini bertujuan sebagai berikut :

1. Lengan robot mampu bergerak menggunakan metode *inverse kinematic*
2. Sensor gas dapat mendeteksi sumber gas
3. Robot mampu mendeteksi sumber gas dengan kendali PID

1.4 Batasan Masalah

Batasan masalah dari tugas akhir ini adalah sebagai berikut:

1. Sensor gas yang digunakan berupa sensor gas semikonduktor TGS2600.
2. Pergerakan lengan menggunakan metode *inverse kinematic*.
3. Sudut jangkauan pendeteksian hanya 180°.
4. Proses pengolahan data sensor dan PID dilakukan oleh STM32F4

1.5 Metodologi Penelitian

Dalam penyelesaian tugas akhir ini digunakan metodologi sebagai berikut:

1. Studi literatur

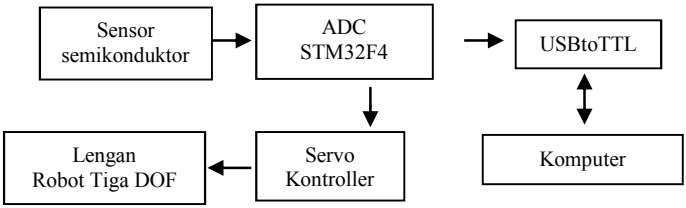
   Pada tahap ini dilakukan pengumpulan dasar teori yang menunjang dalam penulisan tugas akhir. Dasar teori ini dapat diambil dari buku-buku, jurnal, *proceeding*, dan artikel-artikel di internet.

2. Perancangan sistem

Setelah mempelajari literatur yang ada, selanjutnya akan dilakukan perancangan sistem. Perancangan sistem terbagi sebagai berikut:

a. Perancangan Perangkat Keras

Cara kerja dari sistem ini menggunakan sensor gas *semiconductor* (TGS 2600) diimplementasikan pada lengan robot tiga DOF, *error* sensor tersebut dijadikan masukan pada kendali PID. Pendeteksian sensor akan dibaca oleh STM32F4 dengan *Analog to digital converter* (ADC). Hasil pembacaan tersebut diolah oleh kontrol PID dalam STM32F4 yang outputnya akan ditambahkan atau dijumlahkan dengan PWM untuk mengatur servo melalui *servocontroller*. Komunikasi USART yang digunakan adalah 2 USART, yakni komunikasi dengan servocontroller dan mengirimkan output PID ke komputer. Berikut diagram blok dari perancangan kontrol lengan robot tiga DOF untuk mendeteksi sumber gas:



Gambar 1.1 Diagram Blok Perancangan Lengan Robot tiga DOF

b. Perancangan perangkat lunak

Perancangan perangkat lunak meliputi dua proses yaitu pemprograman mengkonversikan data sensor menjadi data digital, dan proses program kontrol lengan robot tiga DOF dengan metode kendali PID.

c. Perancangan posisi sensor

Sensor semikonduktor TGS 2600 diimplementasikan pada lengan robot tiga DOF. Tujuan tugas akhir ini mendeteksi sumber gas dengan pergerakan lengan robot tiga DOF untuk itu diperlukan jumlah sensor semikonduktor sebanyak dua buah yang akan dirancang perangkat kerasnya. proses perancangan di posisikan memudahkan mendeteksi sumber gas dengan gerakan horizontal dan vertikal.

d. Pengujian sistem

Pengujian sistem dilakukan dengan beberapa tahap, yaitu :

- Pengujian perbagian dalam blok diagram dari keseluruhan rangkaian guna mengetahui fungsi kerja elektrik secara tepat.
- Pengujian secara langsung dengan mengambil beberapa sampel guna mengetahui keoptimalan kerja lengan robot yang telah dibuat.

e. Pengolahan data

Melakukan analisa data dari hasil eksperimen, sehingga dapat dipilih solusi yang terbaik untuk mengatasi permasalahan yang dihadapi.

f. Penulisan laporan Tugas Akhir

Tahap penulisan laporan Tugas Akhir dilakukan pada saat tahap pengujian sistem dimulai serta setelahnya.

1.6 Sistematika Penulisan

Laporan tugas akhir ini terdiri dari Lima Bab dengan sistematika penulisan sebagai berikut:

- Bab 1 : Pendahuluan
  Bab ini meliputi latar belakang, perumusan masalah, tujuan penelitian, batasan masalah, metodologi, sistematika penulisan, dan relevansi.
- Bab 2 : Dasar Teori
  Bab ini menjelaskan tentang dasar-dasar teori yang dibutuhkan dalam pengerjaan tugas akhir ini, yang meliputi teori dasar sensor semikonduktor, kendali *PID,* STM32F4, dan *servo controller*.
- Bab 3: Perancangan Sistem
  Bab ini menjelaskan tentang perencanaan sistem perangkat keras elektrik dan mekanik, serta perangkat lunak. Bab ini juga berisi menjelaskan tentang prosedur pengujian yang dilakukan dalam penelitian.
- Bab 4 : Pengukuran dan Analisis Sistem
  Bab ini menjelaskan tentang hasil yang didapat dari pengujian tiap Blok sistem secara keseluruhan

- Bab 5 : Penutup
  Bab ini menjelaskan tentang kesimpulan meliputi kekurangan-kekurangan pada kerja alat dari hasil analisa serta saran untuk pengembangan ke depan.

1.7 Relevansi

Hasil dari tugas akhir ini diharapkan dapat memberikan manfaat sebagai berikut:

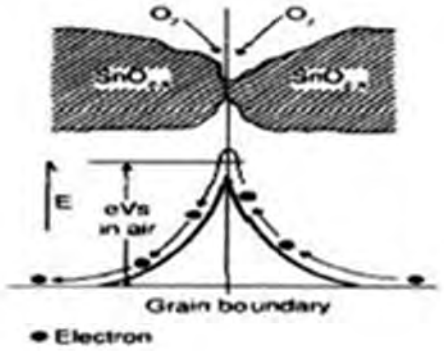
1. Dapat digunakan sebagai alat pengidentifikasi sumber kebocoran gas.
2. Sebagai dasar penelitian lebih lanjut, agar dapat lebih dikembangkan.

.....Halaman ini sengaja dikosongkan.....

BAB II
DASAR TEORI

2.1 Sensor Semikonduktor

Sensor gas adalah sensor yang berfungsi untuk mengukur senyawa gas polutan yang ada di udara seperti *karbon monoksida, hidrokarbon, nitrooksida,* dan lain-lain. Sensor gas semikonduktor memiliki banyak jenis, tentunya dibedakan oleh sensitivitas sensor tersebut. Berdasarkan gambar 2.1 prinsip kerja dari sensor ini semakin tinggi kosentrasi gas maka resistansinya semakin rendah. Sensor gas semikonduktor terbentuk pada permukaan luar kristal, tegangan permukaan yang terbentuk akan menghambat laju aliran elektron. Seperti ilustrasi penyerapan O_2 oleh sensor pada gambar di bawah ini:

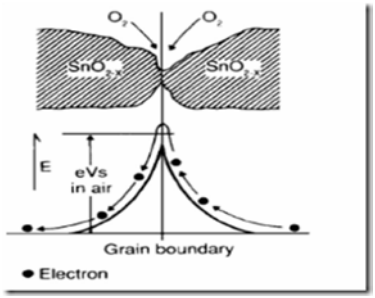


Gambar 2.1 Ilustrasi penyerapan O_2 oleh sensor [3]

2.1.1 Prinsip Kerja Sensor Gas Tipe Semikonduktor

Sensor gas tipe semikonduktor terdiri dari elemen sensor, dasar sensor dan tudung sensor. Elemen sensor menggunakan bahan-bahan seperti timah(IV)oksida SnO_2, wolfram (VI) oksida WO_3 , dan lain-lain. Bila suatu kristal oksida logam seperti SnO_2 dipanaskan pada suhu tinggi tertentu di udara, oksigen teradopsi pada permukaan kristal dengan muatan negatif. Pada gambar 2.2 elektron – elektron donor pada permukaan kristal ditransfer ke

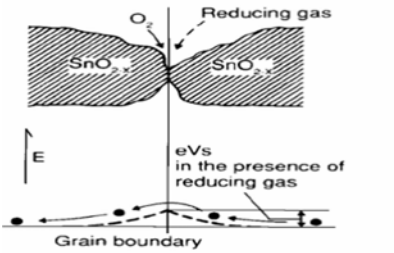
oksigen teradopsi, sehingga menghasilkan suatu lapisan ruang bermuatan positif. Akibat dari peristiwa tersebut potensial permukaan terbentuk yang akan menghambat aliran elektron.Proses didalam sensor arus listrik mengalir melalui bagian – bagian penghubung (batas butir) kristal-kristal mkro SnO2. Di batas-batas antar butir, oksigen yang teradopsi membentuk penghalang potensial yang menghambat muatan bebas bergerak. Tahan listrik sensor disebabkan oleh penghalang potensial.



Ket:
eVs: Nilai energy penghalang permukaan.

Gambar 2.2 Bentuk penghalang potensial antar butir Kristal mikro SnO2 saat adanya tanpa adanya gas[3].

Pada gambar 2.3 ketika Model penghalang potensial antar butir dalam lingkungan gas atau terdeteksi gas, kerapatan oksigen teradopsi bermuatan negatif pada permukaan semikonduktor sensor menjadi berkurang, sehingga ketinggian penghalang pada batas antar butir berkurang. Sehingga menyebabkan berkurangnya tahanan butir dalam lingkungan gas.



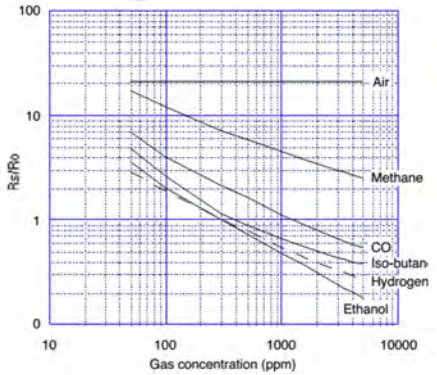
Gambar 2.3 Model penghalang potensial antar butir dalam lingkungan gas[3]

Hubungan antar tahanan sensor dan kosentrasi gas pereduksi pada suatu rentang konsetrasi gas dapat dinyatakan dengan persamaan berikut [3] :

$$Rs = A[C]^{-a} \quad (2.1)$$

Keterangan : Rs = Resistansi Sensor
A = Konstanta
[C] = Konsentrasi gas
a = gradien kurva Rs

2.1.2 Sensitivitas Terhadap Gas

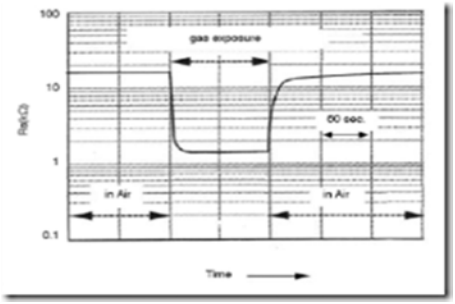


Gambar 2.4 Karakteristik sensitivitas sensor tertentu terhadap berbagai gas [4].

Berdasarkan persamaan (2.1), hubungan resistansi sensor terhadap konsentrasi gas adalah linier dalam bentuk logaritma, dalam rentang tertentu konsentrasi gas (dari beberapa ppm ke beberapa ribu ppm). Pada gambar 2.4 adalah suatu contoh hubungan antar resistansi sensor dan konsentrasi gas. Sensor memperlihatkan kepekaan yang berbeda-beda terhadap berbagai gas. Tingkat kepekaan relatif suatu sensor terhadap gas juga tergantung pada jenis bahan sensor dan temperatur. Sehingga karakteristik sensor dinyatakan sebagai rasio resistansi sensor dalam berbagai konsentrasi gas (Rs) dengan resistansinya dalam konsentrasi tertentu suatu gas target (Ro).

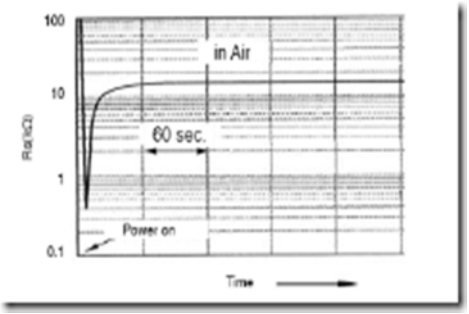
2.1.3 Respons Sensor

Respon sensor ketika suatu sensor dikenakan suatu gas pendeoksida dan ketika sensor dijauhkan dari gas pendeoksida. Seperti pada gambar 2.5 resistansi sensor akan menurun tajam dengan cepat ketika dikenakan pada gas, dan ketika dijauhkan dari gas, resistansinya akan kembali ke harga semula setelah waktu yang singkat. Perlu diperhatikan kecepatan respond kembalinya ke keadaan semula bervariasi dengan jenis sensor dan jenis gas yang terdeteksi.



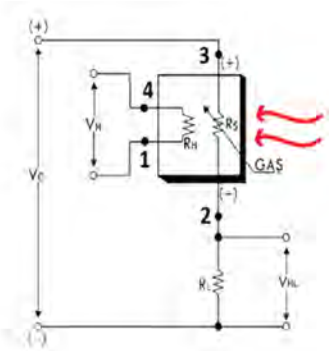
Gambar 2.5 Contoh respon sensor [3].

Semua sensor memiliki sifat sementara disebut aksi awal. Pada gambar 2.6 besarnya Rs turun seketika dengan tajam selama beberapa detik, dalam keadaan ada atau tidak ada gas yang terdeteksi, selanjutnya akan mencapai tingkat yang stabil sesuai dengan keadaan atmosfir sekitarnya. Lamanya aksi awal tergantung pada kondisi atmosfir selama penyimpanan, dan jenis sensor.



Gambar 2.6 Aksi Awal [3]

2.1.4 Rangkaian Pengukuran Dasar



Ket:
V_H: Voltage pemanas
Vc: Voltage sirkit Sensor
RL: Tahanan Beban
Vrl: Voltage antar kedua terminal tahanan beban
Rs: Tahanan Semikonduktor

Gambar 2.7 Rangkaian Pengukuran dasar sensor[4]

Sesuai dengan gambar 2.7 sensor memerlukan dua sumber tegangan, yakni tegangan pemanas (V_H) dan tegangan sirkit/rangkaian sensor (Vc). Tegangan pemanas dipakai pada pemanas terintegrasi untuk mempertahankan elemen sensor pada suhu tertentu yang optimal. Tegangan sirkit digunakan untuk memungkinkan pengukuran tegangan (V_{RL}) antar kedua terminal tahanan beban (R_L) yang dihubungkan seri dengan sensor. Suatu sirkit catu daya umum dapat digunakan baik untuk Vc maupun V_H untuk memenuhi kebutuhan listrik sensor. Konsumsi daya (Ps) pada semikonduktor dibawah 15 mW.

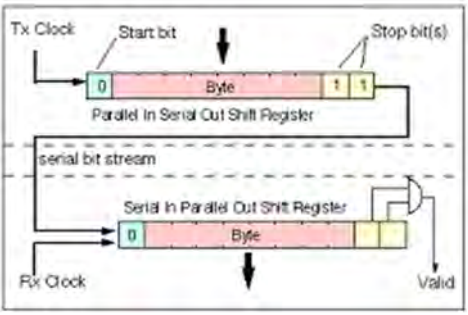
2.1.5 *Sensor Figaro 2600*

TGS 2600 General Air Quality Sensor gas untuk mendeteksi besarnya kontaminasi dalam udara dengan spesifikasi:

- Target gas : *Hydrogen* dan *carbon monoxide*
- *Resistance Output*
- *Typical detection range* : 1-10 ppm (*hydrogen*)
- *Circuit Voltage*: 5VDC.
- *Heater Voltage* : 5VDC/AC
- *Sensor Resistance: 10K-90K* ohm di udara. [4]

2.2 Komunikasi Serial

Komunikasi serial adalah proses transfer data yang secara berurutan mengirimkan/menerima hanya 1 bit data dalam satu waktu. Komunikasi serial dibagi menjadi dua Jenis yaitu, sinkron dan asinkron. Sinkron adalah data 8-bit dikirimkan dalam periode clock. Misalnya SPI, dan I2C. Asinkron adalah data 8bit dikirimkan setidaknya dalam 8 periode bit. Misalnya, USBtoTTL.



Gambar 2.8 Komunikasi serial asinkron[5]

Pada gambar 2.8 paket data berjumlah *11 bit (data 8bit + 3 bit header/tailer).Start bit* berjumlah 1bit, *Stop bit* berjumlah 1 bit, dan *parity* berjumlah *1 bit.*

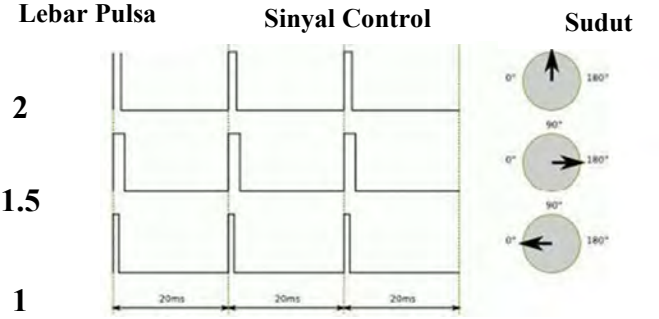
2.3 Motor Servo

Motor servo adalah sebuah perangkat atau aktuator putar (motor) yang dirancang dengan sistem kontrol umpan balik loop tertutup sehingga dapat diatur untuk menentukan dan memastikan posisi sudut dari poros output motor tersebut.

Gambar 2.9 Motor servo[6]

Terdapat 3 kabel yaitu *power, ground, control*. Jenis – jenis motor servo adalah motor servo standar 180° dan *motor servo continuous.* Motor servo standar 180° mampu bergerak dua arah (CW dan CCW) dengan defleksi masing-masing sudut mencapai 90° sehingga total sudut dari kanan-tengah-kiri adalah 180°. Motor servo continuous mampu bergerak dua arah (CW dan CCW) tanpa batasan defleksi sudut putar (dapat berputar secara kontinyu).

Pengendalian gerakan batang motor servo dapat dilakukan dengan metode PWM seperti pada gambar 2.10. Teknik ini menggunakan sistem lebar pulsa untuk mengemudikan putaran motor. Sudut dari sumbu motor servo diatur berdasarkan lebar pulsa yang dikirim melalui kaki sinyal dari kabel motor. Ketika pulsa 1.5 ms pada periode selebar 2 ms, maka sudut dari sumbu motor akan berada pada posisi tengah. Berikut sistem sinyal control servo dijelaskan pada gambar dibawah ini:



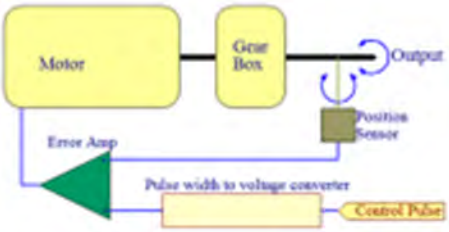
Gambar 2.10 Pensinyalan Control Servo [6]

Pergerakan *motor servo* ke kanan atau ke kiri, tergantung dari *delay* yang diberikan. Ketika pulsa diberikan 1.5 ms servo pada posisi center, diberikan pulsa ≤ 1.3 ms servo memutar ke kanan, dan ketika diberikan pulsa ≥ 1.7 ms untuk berputar kekiri dengan delay 20ms, sesuai yang dijelaskan pada gambar di atas. Salah satu perbedaan utama antara *motor servo* dan *steppermotor* adalah bahwa *motor servo*, dari definisi, dijalankan dengan menggunakan control loop dan memerlukan sejumlah umpanbalik. Control loop menggunakan umpanbalik dari motor untuk membantu motor memperoleh keadaan *(state)* yang diinginkan (posisi, kecepatan, dan sebagainya). Karena motor servo

mempunyai control loop umumnya lebih andal daripada steppermotor, bila steppermotor gagal dalam suatu step karena suatu hal, tidak ada *control loop* untuk mengkompensasi gerakan. *Control loop* pada *motor servo* secara tetap memeriksa apakah motor dalam lintasan yang benar dan jika tidak maka dilakukan adjustment yang diperlukan. Beberapa keuntungan motor servo dibandingkan dengan steppermotor adalah:

1. *High intermittent torque*
2. *Torsi* tinggi untuk *inertia ratio*
3. Kecepatannya tinggi
4. Bekerja baik untuk kontrol kecepatan
5. Tersedia dalam banyak ukuran
6. Tidak menimbulkan suara keras.

Motor servo terdiri dari beberapa bagian utama, motor dan *gearbox*, sensor posisi, *error amplifier* dan *motor driver* serta sircuit yang mengkode posisi yang diminta. Gambar 2.11 menunjukkan Blok diagram*motorservo*(*typical*). *Radio control receiver system* membangkitkan suatu pulsa yang lebarnya berubah sekitar setiap 20ms. Lebar pulsa digunakan oleh *servo* untuk menentukan posisi rotasi yang dikehendaki.

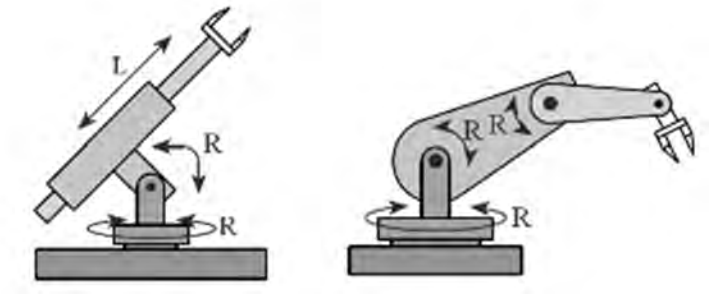


Gambar 2.11 Diagram blok motorservo[6]

2.4 Robot Lengan

Robot lengan adalah jenis robot yang memiliki tipe lengan mekanik terprogram dengan fungsi yang mirip dengan fungsi lengan manusia. Robot lengan mempunyai link manipulator yang terdiri dari *base, shoulder, elbow, arm,* dan *finger*. Link manipulator tersebut dihubungkan oleh sendi sehingga memiliki gerak rotasi atau translasi (liner) sehingga link manipulator dapat dianggap membentuk rantai kinematik. Ujung rantai kinematik manipulator disebut dengan *end effector* yang memiliki fungsi seperti jari-jari pada tangan manusia. *End*

effector dapat dirancang untuk melakukan tugas yang diinginkan seperti las, mencekram, berputar, dan lain sebagainya tergantung pada aplikasi.



Gambar 2.12 Komponen lengan robot

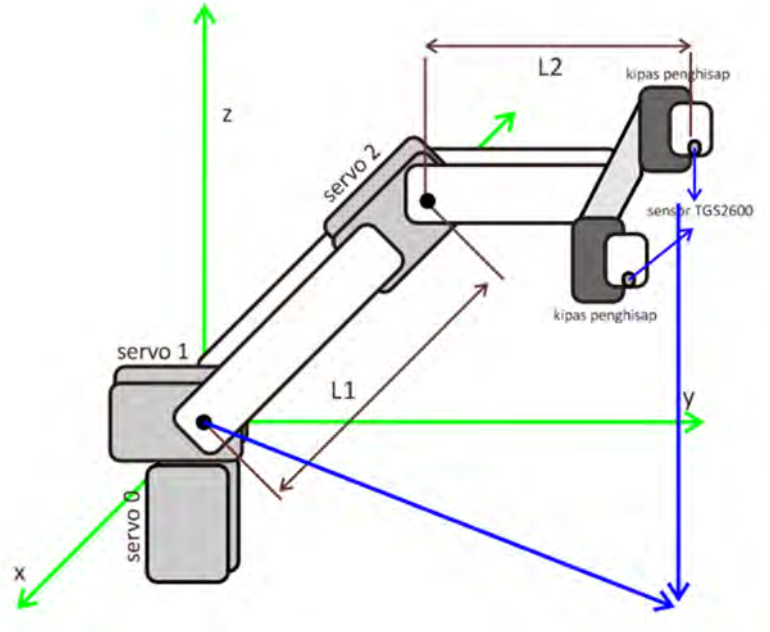
Berikut komponen lengan robot seperti pada gambar 2.12:

1. *Link*: Bagian robot yang bentuknya tetap dan dapat bergerak. *Link* biasanya dihubungkan dengan *joint*.
2. *Joint*: Penghubung *link* dengan *link* atau base yang dapat bergerak aktif (biasanya terdapat akuator).
3. *End of Effector* : ujung robot yang dapat berinteraksi dengan objek
4. *Tools*: Bagian yang diletakkan pada ujung robot *(End of Effector)*. Terdapat banyak jenis tools yang dapat dipasang. Namun pada robot lengan tools umumnya berupa *gripper* atau penjepit.
5. DOF *(Degree of Freedom)*: adalah jumlah derajat kebebasan atau jumlah gerakan independen yang dapat dilakukan oleh suatu robot. [7]

2.4.1 Inverse Kinematic

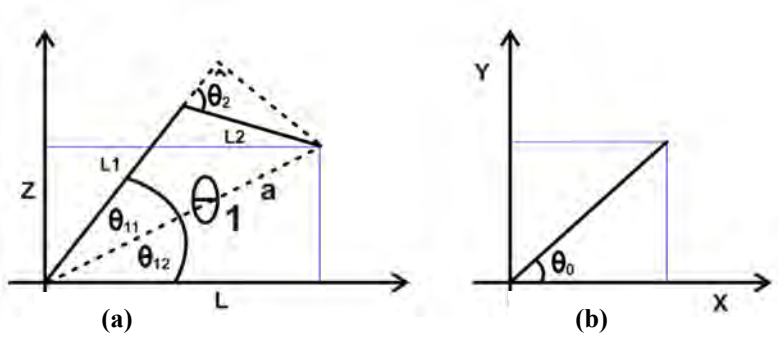
Pengertian Kinematika robot dapat didefinisikan sebagai pergerakan lengan robot *(motion)* tanpa memperhatikan gaya ataupun faktor lain yang mempengaruhi gerakan lengan robot tersebut. Kinematika secara umum terbagi menjadi dua yakni *Forward kinematics* dan *Inverse Kinematics*. *Forward kinematics* adalah analisis *kinematic* untuk mendapatkan kordinat posisi (x,y,z) jika diketahui sudut dari tiap sendi.

Dengan penerapan *Inverse Kinematics*, kita mendapatkan sudut tiap servo dengan menginputkan koordinat posisi pada robot. Pada lengan robot 3 DOF memiliki tiga buah sumbu utama yaitu sumbu x, sumbu y dan sumbu z.



Gambar 2.13 Komponen lengan robot

Pada gambar 2.12 menunjukkan bahwa lengan robot diwakili oleh tiga servo, yaitu servo 0, servo 1, dan sevo 2. Servo tersebut dikendalikan dengan pengaturan T_{on} mulai dari 600 us – 2400 us. Pembagian masing-masing sudut kerja servo yakni, Servo0 (θ_0), Servo1 (θ_1), dan Servo2 (θ_2). Lengan yang menghubungkan antara servo 1 dan servo 2 di misalkan L2. Dan servo 2 dengan ujung lengan di misalkan L1. Dari kombinasi antara servo dan lengan membentuk sudut, yaitu T1 dan T2.



Gambar 2.14 Tampak lengan robot dari samping (a), bampak lengan robot dari atas (b)

$$a^2 = z^2 + L^2 \tag{2.2}$$
$$a = \sqrt{z^2 + L^2}$$

Dengan mensubstitusikan persamaan 2.2 pada persamaan 2.3, sehingga didapatkan θ_2 sebagai berikut :

$$L = \sqrt{x^2 + y^2}$$
$$a^2 = {l_1}^2 + {l_2}^2 - 2l_1 l_2 \cos(180 - \theta_2) \tag{2.3}$$
$$a^2 = {l_1}^2 + {l_2}^2 + 2l_1 l_2 \cos(\theta_2)$$
$$-2l_1 l_2 \cos(\theta_2) = -a^2 + {l_1}^2 + {l_2}^2$$
$$\cos(\theta_2) = \frac{-a^2 + {l_1}^2 + {l_2}^2}{-2l_1 l_2}$$
$$\theta_2 = \frac{-a^2 + {l_1}^2 + {l_2}^2}{-2l_1 l_2}$$

Dimana θ_1 adalah

$$\theta_1 = \theta_{11} + \theta_{12} \tag{2.4}$$
$$\theta_{11} = tan^{-1}\frac{z}{L} = tan^{-1}\frac{z}{\sqrt{x^2+y^2}} \tag{2.5}$$
$$\theta_{12} = tan^{-1}\frac{l_2 sin\theta_2}{l_1 + l_2 cos\theta_2} \tag{2.6}$$

Dengan mensubstitusikan persamaan 2.5 dan persamaan 2.6 ke persamaan 2.4, didapatkan θ_1 sebagai berikut :

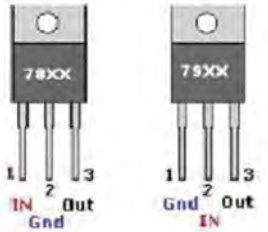
$$\theta_1 = \theta_{11} + \theta_{12} = tan^{-1}\frac{z}{\sqrt{x^2+y^2}} + tan^{-1}\frac{l_2 sin\theta_2}{l_1+l_2 cos\theta_2} \quad (2.7)$$

Dimana θ_0 adalah

$$\theta_0 = tan^{-1}\frac{y}{x} \quad (2.8)$$

2.5. Voltage Regulator

Regulator Voltage berfungsi sebagai *filter* tegangan agar sesuai dengan keinginan. Oleh karena itu biasanya dalam rangkaian power supply maka IC regulator tegangan ini selalu dipakai untuk stabilnya outputan tegangan.



Gambar 2.15 78xx untuk regulator positif, 79xx untuk regulator negative [8]

Pada gambar 2.15 adalah bentuk dari regulator 7805. Regulator 7805 adalah regulator untuk mendapat tegangan +5 volt, 7812 regulator +12 dan sebagainya. Sedangkan seri 79xx misalnya adalah 7905 dan 7912 yang berturut-turut adalah regulator tegangan -5 dan – 12 volt.

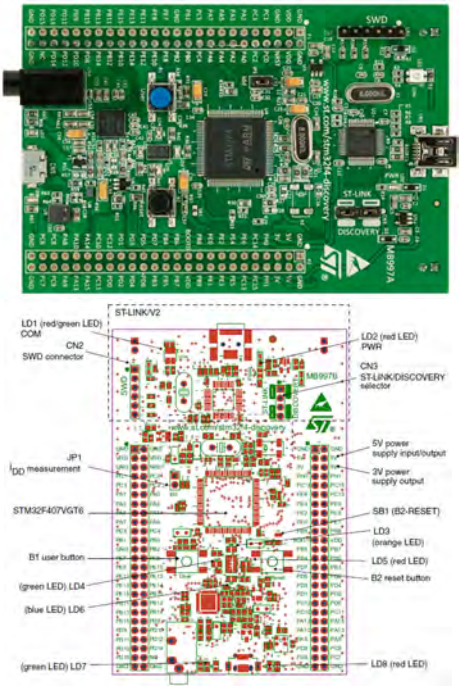
2.6 STM32F4

STM32F4 adalah produk dari ST Electronics yang memudahkan penggunanya untuk melakukan bebagai macam eksperimen elektronik. Bentuk board dari STM32F4 dapat dilihat pada gambar 2.16.

Berikut adalah beberapa fitur dari STM32F4 :

1. STM32F407VGT6 *microcontroller* menggunakan 32-bit ARM Cortex-M4F core, 1 MB Flash dan 192 KB RAM
2. Supply Board melalui USB atau eksternal 5 V

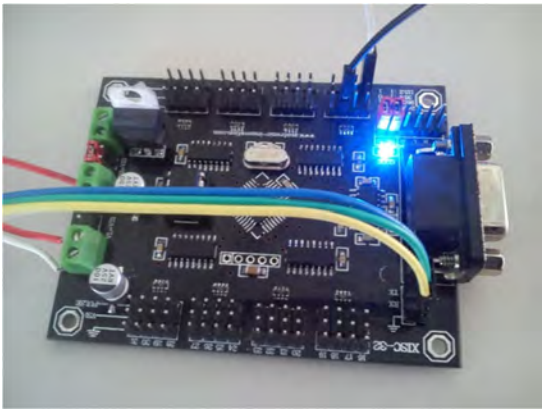
3. Eksternal power supply: 3 V and 5 V yang disediakan oleh board STM32F4
4. LD1 (merah/hijau) untuk komunikasi dengan USB
5. LD2 (merah) funtuk 3.3 V indikator supply



Gambar 2.16 STM32F4 - Discovery [9]

2.7 XISC 32

Servo kontroller XISC 32 digunakan untuk mengatur 32 motor servo dengan menggunakan komunikasi serial. Komunikasi ini antara master ke servo kontroller. Yang bertindak sebagai master adalah Atmega 32. Gambar 2.17 menunjukkan gambar dari XISC 32

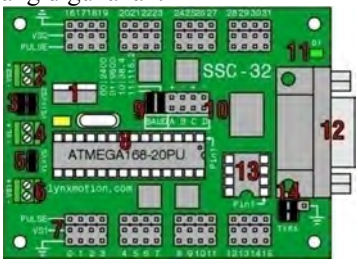


Gambar 2.17 Board XISC 32

XISC 32 ini membutuhkan catu daya 6-9 volt. Karena di XISC 32 terdapat regulator 7805 yang digunakan untuk memberikan tegangan pada Atmega 168sebesar 5 volt. Servo kontroler ini terdapat pin TX RX yang digunakan untuk komunikasi serial dengan PC. Beberapa *baudrate* yang tersedia pada XISC 32, diantaranya 2400, 9600, 38400, 115200 bps. Dan biasa yang digunakan untuk PC menggunakan *baudrate* standar PC adalah 115200.

2.7.1 Konfigurasi Servo Kontroller

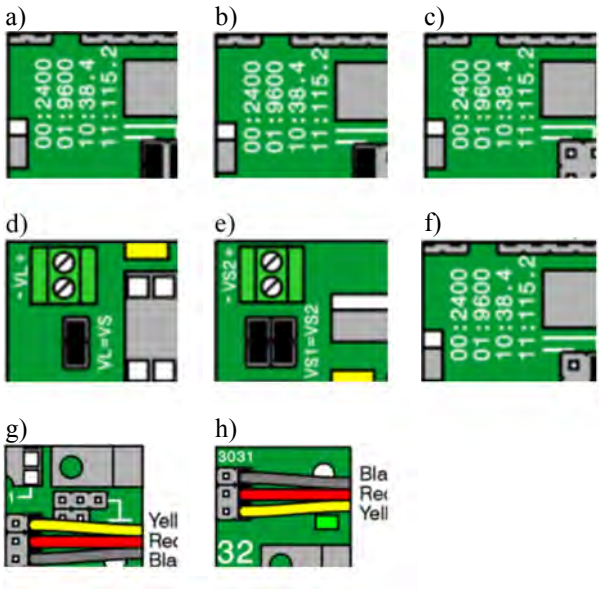
Pada XISC32 seperti pada gambar 2.18, memiliki beberapa pin konfigurasi yang digunakan.



Gambar 2.18 Konfigurasi pin XISC 32

Berikut adalah konfigurasi pin dari XISC 32:

1. Regulator 7805 yang digunakan untuk memberikan tegangan 5volt ke mikrokontroller atmega 168.
2. Konektor yang digunakan untukmemberi tegangan 16 atau 32 servo. Tegangan yang dapat beroperasi pada 16 atau 32 servo tersebut adalah 5volt – 7,4volt.
3. Konektor yang digunakan untuk menggabungkan catu daya servo menjadi 1 baterai.
4. Konektor ini digunakan untuk memberikan catu daya pada regulator 7805.
5. Konektor ini digunakan untuk menjadikan catu daya satu sumber, antara atmega 168dan catu daya servo.
6. Konektor untuk memberikan tegangan 16 atau 32 servo. Tegangan kerja servo 5volt – 7,4 volt.
7. Konektor ini digunakan untuk koneksi ke motor servo
8. Mikrokontroller ini digunakan untuk mengatur sistem dari XISC 32
9. Digunakan untuk mengatur kecepatan pengiriman data dalam komunikasi serial
10. Digunakan untuk sistem *latching* dan statik.
11. Led digunakan sebagai indikator pengiriman data serialnya.
12. DB9 digunakan untuk komunikasi dengan PC.
13. IC eeprom digunakan untuk *firmware*
14. Ini digunakan untuk komunikasi serial dari master ke XISC 32.



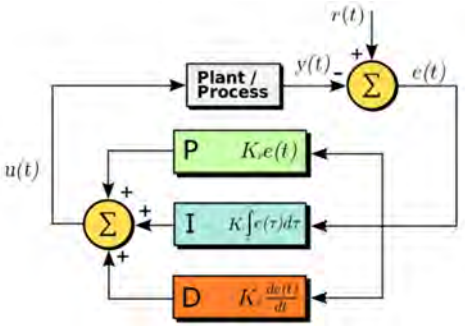
Gambar 2.19 Pin konektor XISC 32

Keterangan gambar 2.19 terdapat beberapa pin konektor XISC 32, diantaranya:

a) Setting *jumper* untuk menentukan *baudrate*pada XISC 32 senilai 115200bps
b) Setting *jumper* untuk menentukan *baudrate*pada XISC 32 senilai 38400 bps
c) Setting *jumper* untuk menentukan *baudrate*pada XISC 32 senilai 2400bps
d) Setting *jumper* ini digunakan untuk setting catu daya servo dan catu daya atmega 168menjadi satu
e) Setting *jumper* ini digunakan untuk kedua catu daya servo menjadi satu
f) Setting *jumper* untuk menentukan *baudrate*pada XISC 32 senilai 9600bps
g) Digunakan untuk standart pemasangan servo pada pin 0-15
h) Digunakan untuk standart pemasangan servo pada pin 16-31 [10]

2.8 Pengendali Propotional Integral Derivatif (PID)

Untuk menutupi semua kekurangan pengendali PI maupun pengendali PD, maka ketiga mode yang ada digabung menjadi pengendali PID. Unsur P, I, maupun D berfungsi untuk mempercepat reaksi sistem menghilangkan offset. Karena masing-masing mempunyai kelebihan, maka men-tuning Kp, Ki, dan Kd unsur tersebut dibuat lebih menonjol dari pada yang lain. Misalnya untuk P dibuat lebih menonjol dari I maupun D, atau unsur I dibuat lebih menobjol daripada P maupun D, unsur yang menonjol itulah yang kemudian akan membawa pengaruh pada respon sistem keseluruhan.



Gambar 2.20 Implementasi PID pada *plant* dengan *feedback* [11]

Dari gambar 2.20 fungsi transfer untuk pengendalian PID adalah,

$$O = Kp * e + Ki \int e\,dt + Kd\,de/dt \quad (2.9)$$

dimana :

Kp = 100%/PB (konstanta penguatan proporsional)
Ki = K/Ti (konstanta penguatan integral)
Kd = KpxTd (konstanta penguatan diferensial)

Pada tanggapan transien ini ada bebarapa parameter yang perlu diketahui, yaitu:

a. Waktu Tunda (*Delay Time*), adalah waktu yang diperlukan sistem untuk mencapai seperuh dari harga akhirnya untuk pertama kali.
b. Waktu Naik (*Rise Time*), adalah waktu yang diperlukan sistem untuk naik dari 10% sampai 90% nilai akhir.

c. Waktu Puncak (*Peak Time*), waktu yang diperlukan sistem untuk mencapai puncak pertama kali.

2.8.1 Kendali Proposional

Salah satu dari mode pengendali yang paling populer adalah unit pengendali proportional. Seperti yang tercermin dari namanya, unit pengendalian ini memberikan output-an yang sebanding (proporsional) dengan besarnya error. Perubahan nilai *Proportional Gain/ proportional Band* akan mempengaruhi respon sistem terhadap perubahan error dan load. Persamaan dari unit kendali ini adalah sebagai berikut :

$$Output = (Error\ x\ Gc) + Bias \quad (2.10)$$

Gain unit control / Kp proportional dapat berupa bilangan bulat, atau bilangan pecahan. Semakin besar nilai gain akan menyebabkan pengendali semakin reaktif terhadap error, hal ini ditandai dengan adanya overshoot pada kondisi transient dan sebaliknya. Unit pengendali proporsional tidak bergantung pada fungsi waktu.

2.8.2 Kendali Integral

Unit pengendali ini disebut juga sebagai unit pengendali reset karena kemampuannya mengeliminasi offset yang ditinggalkan oleh pengendali proportional. Dengan persamaan :

$$Output = Ki \int Err\ dt + Bias \quad (2.11)$$

dengan $Ki = 1/Ti$

Dimana: Ki = *Integral Gain* Ti = *Integral time*

Kontrol I dapat memperbaiki sekaligus menghilangkan respon *steady-state*, namun pemilihan Ki yang tidak tepat dapat menyebabkan respon transien yang tinggi sehingga dapat menyebabkan ketidakstabilan sistem.

2.8.3 Kendali Derivatif

Unit pengendali ini disebut juga preact karena karakteristiknya yang memberikan energi ekstra pada saat-saat awal dan sensitif terhadap noise.

$$Output = Kd * de/dt + Bias \quad (2.12)$$

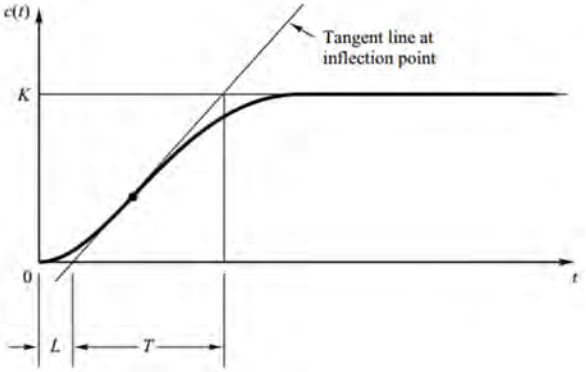
Pengendali differensial ini tidak dapat berdiri sendiri, unit pengendali D ini selalu dipakai dalam kombinasi dengan P dan I, menjadi pengendali PD atau pengendali PID. Selain itu, pengendali D tidak dapat dipakai untuk proses variabel yang beriak (mengandung

noise) karena karakteristiknya yang sangat responsive. Dari fungsi transfer diatas dapat dilihat bahwa besarnya output tergantung pada Gain (Kd) dan besarnya perubahan error. [12]

2.8.4 Pendekatan Ziegler-Nichlos

Metode ini merupakan metode yang digunakan dalam kendali PID untuk mendapatkan penguatan dari Kp, Ti, dan Td yang dapat diperoleh melalui respons transien dari keluaran yang diberikan oleh *plant*.

Metode ini dapat dulakukan dengan percobaan unit step input yang akan menghasilkan kurva yang berbentuk menyerupai huruf S seperti pada gambar 2.21 di bawah:



Gambar 2.21 Respons dari kurva S

Kurva tersebut memiliki dua buah konstanta, yakni konstanta L dan T. Konstanta ini dapat diketahui dengan menggambar sebuah garis yang sejajar dengan infections points dan yang juga memotong sumbu x. Setelah mendapatkan kedua nilai ini, maka selanjutnya nilai dari konstanta pengali PID dapat diketahui melalui tabel 2.1. [13]

Tabel 2.1 Konstanta Pengali PID metode Ziegler-Nichlos

Type of Controller	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

BAB III
PERANCANGAN SISTEM

Perancangan sistem meliputi perangkat keras dan perangkat lunak. Perangkat keras terbagi meliputi perangkat mekanik dan elektrik, Perangkat keras berguna mendapatkan data sensor untuk diolah dengan kendali Proposional Integral Derivatif (PID) dan lengan robot 3 *Degree of Freedom* (DOF) untuk mencari sumber gas. Sensor yang digunakan merupakan sensor semikonduktor TGS 2600. Sensor ini sensitive dengan udara yang terkontaminasi seperti, *gas methane, Carbon monoxide, Iso-butane, ethanol, dan hydrogen.*

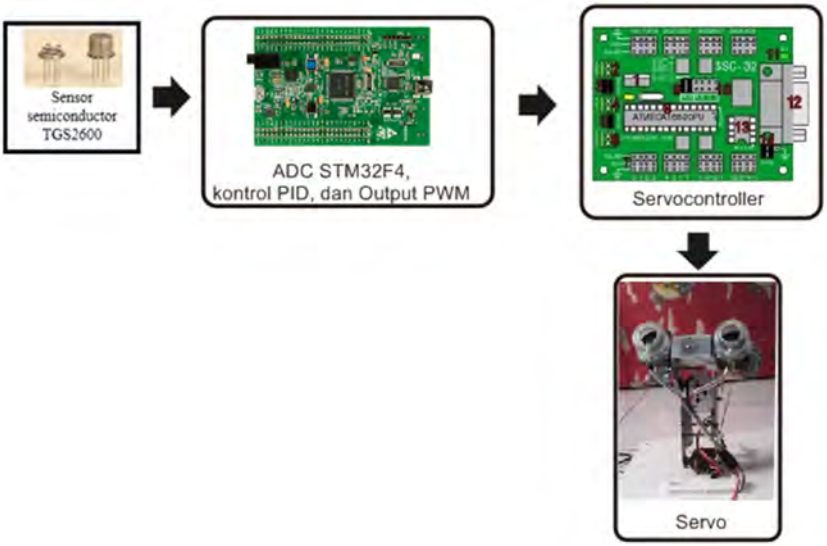
Perangkat lunak meliputi program STM32F4-Discovery dan kendali Proposional Integral Derivatif (PID) menggunakan *software keil uvision,* dan *Borland Delphi*. Kendali Proposional Integral Derivatif (PID) adalah sistem kontrol yang digunakan untuk mengkontrol lengan robot (*Robot Arm*). Bab ini menjelaskan secara keseluruhan dan desain tiap – tiap blok yang ada. Perangkat sistem ini didesain dengan sumber 9 volt untuk *logic servo controller*, 6 volt untuk motor servo, 12 volt untuk kipas penghisap udara, dan 5 volt untuk STM32F4-Discovery dan *driver sensor*.

3.1 Diagram Blok Sistem

Secara umum sistem ini terdiri dari perangkat keras dan perangkat lunak. Perangkat keras meliputi perancangan perangkat elektrik, dan perangkat mekanik. Perangkat lunak meliputi program STM32F4 dan kendali PID menggunakan *software keil uvision,* dan *Borland Delphi*.

Sistem kerja dari lengan robot ini yaitu mendeteksi dan menjangkau sumber gas dengan menggunakan sensor semikonduktor TGS 2600 dengan metode kendali PID. Proses interface yang digunakan untuk menghubungkan ke komputer adalah komunikasi serial USBtoTTL. Output dari sensor berupa tegangan, tegangan ini adalah data analog yang akan di inputkan ke sistem kendali PID. Sebelum diinputkan ke sistem kendali data analog ini harus diubah dalam bentuk digital dengan memanfaatkan ADC internal dari STM32F4 dengan resolusi ADC 12 bit. Output dari STM32F4 yang berupa nilai PID dikirim ke komputer dengan komunikasi serial USBtoTTL, sehingga komputer menerima data tersebut.

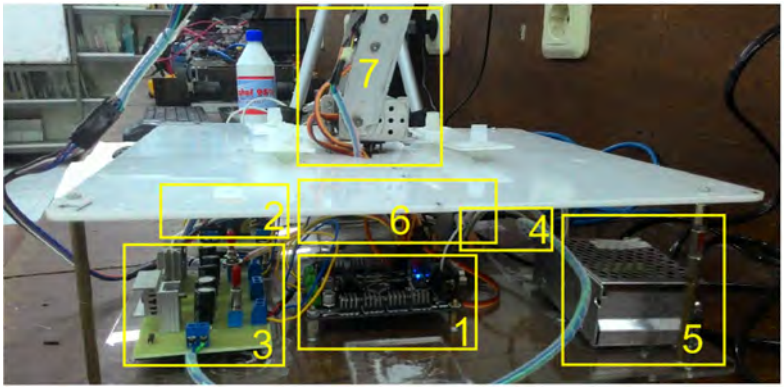
Output dari sistem kontrol ini berupa penambahan atau pengurangan PWM. Hal ini dikarenakan *servocontroller* hanya mengirimkan data PWM ke motor servo untuk menggerakkan lengan robot *(robot arm)*. Pada gambar 3.1 merupakan blok diagram perancangan kontrol lengan robot 3 DOF untuk mendeteksi sumber gas dengan kendali PID.



Gambar 3.1 Diagram Blok Sistem Lengan Robot 3 DOF

3.2 Perancangan Perangkat *Keras*

Perangkat Keras terdiri dari perangkat elektrik dan perangkat mekanik. Perangkat elektrik meliputi *power supply* , rangkaian *sensor*, STM32F4, komunikasi serial dan *servo controller*. Untuk perangkat mekanik adalah kinematic lengan robot tiga DOF. Gambar 3.2 merupakan keseluruhan perangkat keras dari lengan robot 3DOF.



Gambar 3.2 Keseluruhan Perangkat Keras

Keterangan :

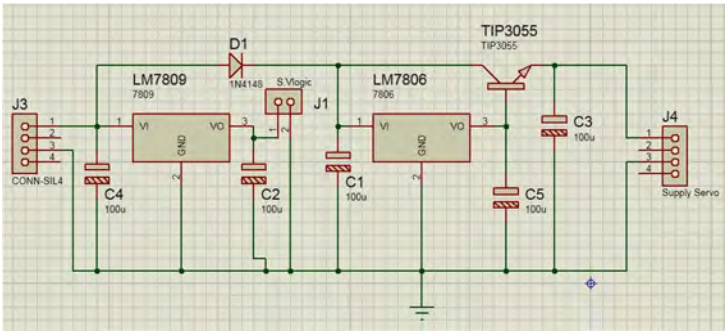
1. Servocontroller
2. Rangkaian pembagi tegangan sensor danPower supply kipas 12v
3. Power supply logic servocontroller 9.2v dan supply servo 5.76v
4. Power supply sensor 5v
5. Adaptor Switching 12v 2A
6. Board STM32F4
7. Rangka lengan robot

3.2.1 Perangkat Elektrik

Perangkat elektrik meliputi *power supply,* Rangkaian *sensor*, STM32F4, komunikasi serial, dan *servo controller.*

3.2.1.1 Power Supply

Power Supply adalah perangkat elektronika yang mensuplai sumber listrik ke perangkat elektronika lainnya. *Power supply* berfungsi mengkonversikan tegangan AC menjadi tegangan DC. Tegangan yang dihasilkan *Power supply* adalah 9 volt untuk *logic servo controller*, 6 volt untuk motor servo, 12 volt untuk kipas penghisap udara, dan 5 volt untuk STM32F4 dan *driver sensor*.



Gambar 3.3 Rangkaian *power supply*

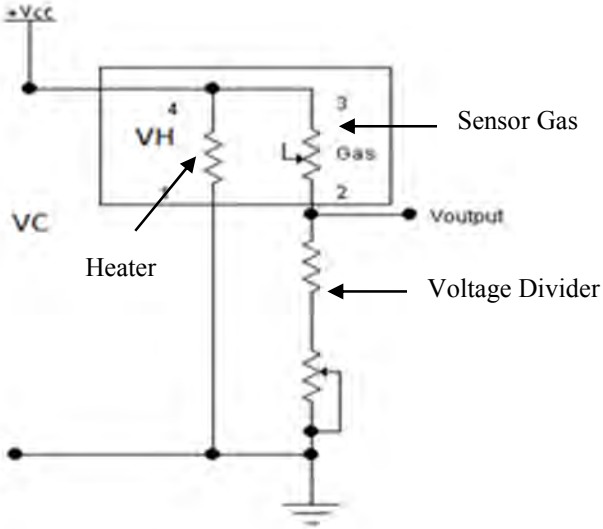
Pada gambar 3.1 kapasitor berfungsi sebagai *filter* menghilangkan *ripple* dari output rangkaian penyearah sistem jembatan. Untuk mendapatkan keluaran tegangan yang dinginkan menggunakan *IC regulator*, rangkaian diatas menggunakan dua *IC regulator* yaitu 7806, dan 7809. *IC regulator* harus mendapatkan tegangan input lebih besar 2 volt dari tegangan output. Untuk mendapatkan tegangan 6 volt maka menggunakan *IC regulator* 7805 dengan tegangan input dapat diambil dari parallel intpu *Supply* 12 volt yang berasal dari sumber adaptor yakni 12 V 2 A.

Menggerakkan motor servo membutuhkan arus yang besar dengan tegangan 6 Volt sebagai V sumber. Diperlukan rangkaian penguat arus, untuk penguatan arus bisa memanfaatkan penguatan transistor TIP 3055.

3.2.1.2 Rangkaian Sensor

Rangkaian sensor adalah rangkaian pengondisian tegangan output dari sensor TGS 2600. Perangkat elektrik ini diperlukan untuk mendapatkan tegangan yang stabil. Rangkaian yang digunakan berupa *Voltage divider*, Rangkaian ini berfungsi mengatur tegangan Vout saat tegangan sensor tidak mendeteksi gas seperti pada gambar 3.4. Ketika sensor diaktifkan nilai tegangan tinggi, ketika tidak mendeteksi gas kembali tegangan Vout turun secara perlahan. Kondisi bernilai satu berada pada tegangan 2,5 volt - 3,3 volt, saat sensor mendeteksi gas. Kondisi

bernilai nol pada tegangan 0 – 2.5 volt, saat sensor tidak mendeteksi gas.



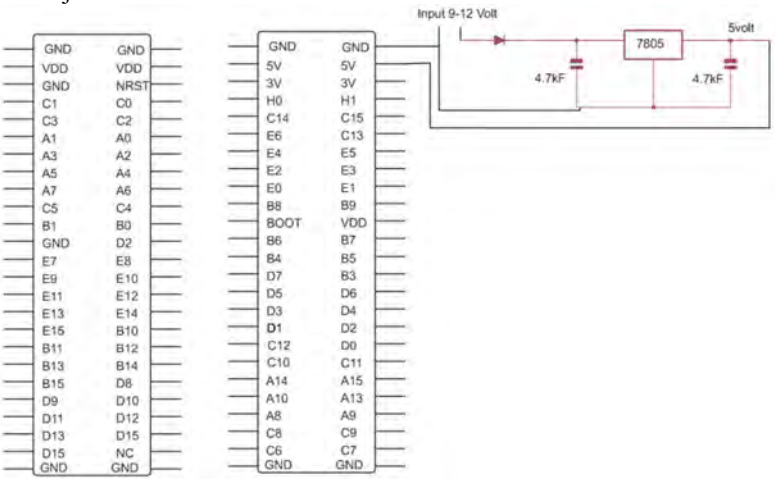
Gambar 3.4 Rangkaian *Voltage divider sensor*

Sesuai dengan dasar teori pada bab dua bahwa sensor TGS 2600 memerlukan dua sumber tegangan, yakni tegangan pemanas (V_H) dan tegangan rangkaian sensor (V_C). Tegangan pemanas dipakai pada pemanas terintegrasi untuk mempertahankan elemen sensor pada suhu tertentu yang optimal. Tegangan sirkit digunakan untuk memungkinkan pengukuran tegangan (V_{RL}) antar kedua terminal tahanan beban (R_L) yang dihubungkan seri dengan sensor. Konsumsi daya (Ps) pada sensor ini adalah 15mW.

3.2.1.3 STM32F4

Dalam tugas akhir ini menggunakan STM32F. Tegangan *Output* pada sensor berupa data *analog* agar dapat STM32F4 data tersebut harus diubah dalam bentuk analog. Fungsi dari STM32F4 ini untuk mengkonversi data *analog* menjadi *digital* dengan memanfaatkan ADC *internal* dengan resolusi 12 bit dan mengolahnya menjadi *input* dari kendali PID untuk mengatur pergerakan lengan robot 3 DOF.

STM32F4 membutuhkan suplai daya dengan tegangan tetap 5V seperti pada rangkaian supply Board STM32F4 pada gambar 3.15. Dalam tugas akhir ini, Rangkaian *power supply* menggunakan LM7805 yang dapat meregulasi tegangan dari 9V menjadi 5V.



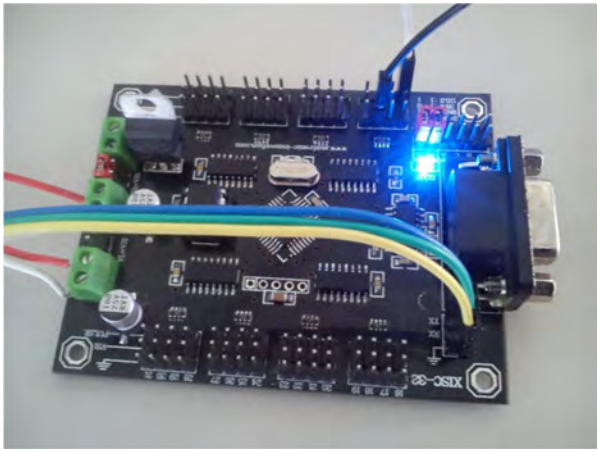
Gambar 3.5 Rangkaian Supply *Board STM32F4*

Proses pemprograman STM32F4 menggunakan *software keil uvision5*. Proses *download* program dari komputer ke STM32F4 cukup menggunakan USB to miniUSB. Sensor yang digunakan berjumlah dua sensor TGS2600, dan menggunakan port A5 dan A7. Tombol *reset* pada STM32F4 dapat digunakan untuk mereset program di dalam STM32F4. STM32F4 telah dilengkapi *USART* untuk proses pengiriman dan penerima data. Keluaran STM32F4 ini dikirim ke komputer dan *servo controller* dengan menggunakan *port serial TX (transmitter*) pada port A2 dan B6.

3.2.1.4 *Servocontroller*

ServoController yang digunakan adalah *servocontroller XISC 32*, memiliki spesifikasi berbasiskan ATmega 168, mampu mengendalikan 32 motor servo, sistem antarmuka *UART RS232/TTL*, tipe servo adalah standar servo (180°), resolusi sudut pergerakan servo 1µs / 0.9°, dan resolusi kecepatan pergerakan *servo* 1µs. Vsumber yang dibutuhkan untuk mengaktifkan *servo*

kontroler sebesar 9V, dan motor servo 6V. Servo kontroller yang digunakan pada tugas akhir ini seperti pada gambar 3.6. Rangkaian suplai *servo controller* menggunakan IC regulator 7809, dan 7806.



Gambar 3.6 *ServoController XISC 32*

Output dari STM32F4 diterima *servocontroller*, sehingga yang digunakan *port RX (receiver)*. Data yang diterima berupa data PWM dengan contoh data yang diterima sebagai berikut:

Contoh:

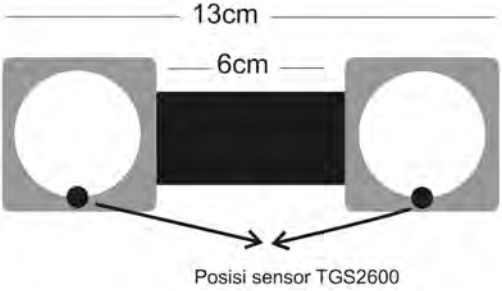
```
0P1250S900
```

0P	: motor *servo* yang digunakan berada pada port 00
1250	: Data PWM
S900	: Besar kecepatan torsi motor.

3.2.1.5 Perancangan Posisi Sensor

Mekanik pada tugas akhir ini adalah memposisikan sensor yang tepat posisinya pada lengan robot (*robot arm*). Posisi sensor yang diletakkan pada lengan robot harus diposisikan secara tepat agar sensor dapat mendeteksi gas diudara dengan tingkat kepekaan yang

baik. Sensor yang digunakan berjumlah dua sensor semikonduktor TGS2600 seperti pada gambar 3.7.



Gambar 3.7 Perancangan Posisi Sensor

Dalam proses *Running* diperlukan Kipas penyedot untuk menarik udara yang terkontaminasi partikel gas melewati dalam tabung sehingga sensor menerima gas lebih banyak. Proses ini membantu gerakan lengan robot tiga DOF.

3.2.1 Perangkat lunak

Perangkat lunak yang dirancang terdiri atas dua bagian yaitu perangkat lunak pada STM32F4 dan perangkat lunak pada Laptop / PC. Perangkat lunak pada STM32F4 berupa perangakat lunak untuk melakukan proses ADC sebagai konversi data sensor analog menjadi data digital dan pemrosesan input berupa data digital sensor yang telah dikonversi untuk diolah dengan kendali PID dan mengendalikan pergerakan lengan robot 3 DOF. Sedangkan laptop /PC, perangkat lunak yang dirancang adalah berupa program untuk menampilkan grafik output dari PID.

3.2.2.1 Perangkat Lunak STM32F4

Perangkat lunak STM32 secara umum digunakan untuk mengkonversikan data *analog* menjadi data *digital* dengan memanfaatkan ADC internal sebesar 12 bit. Proses ini menggunakan *software keil uvision* dalam proses program data sensor dan kendali PID. Perangkat lunak STM32F4 menggunakan clock 12 MHz, *USART transmitter* dengan *Baud Rate* 115200, dan ADC yang digunakan 12 bit.

Dalam tugas akhir ini menggunakan ADC dengan fidelitas 12 bit, sehingga perhitungan ADC *(Analog Digital Converter)* dapat dirumuskan sebagai berikut:

$$\text{ADC} = \frac{Vin\ x\ 4095}{Vref} \qquad (3.1)$$

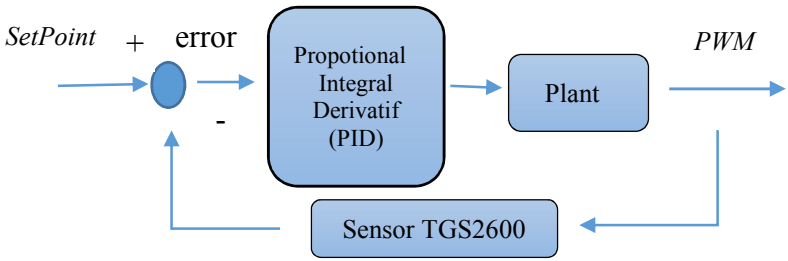
Dan dalam aplikasi program pembacaan data sensor adalah sebagai berikut:

```
baca_sensor[0]=TM_ADC_Read(ADC1, ADC_Channel_5);
baca_sensor[1]=TM_ADC_Read(ADC1, ADC_Channel_7);
```

Dari program diatas data tersebut port yang digunakan adalah PA5 dan PA7 dengan input maksimal sebesar 3,3 volt.

3.2.2.2 Proses Kontrol *PID*

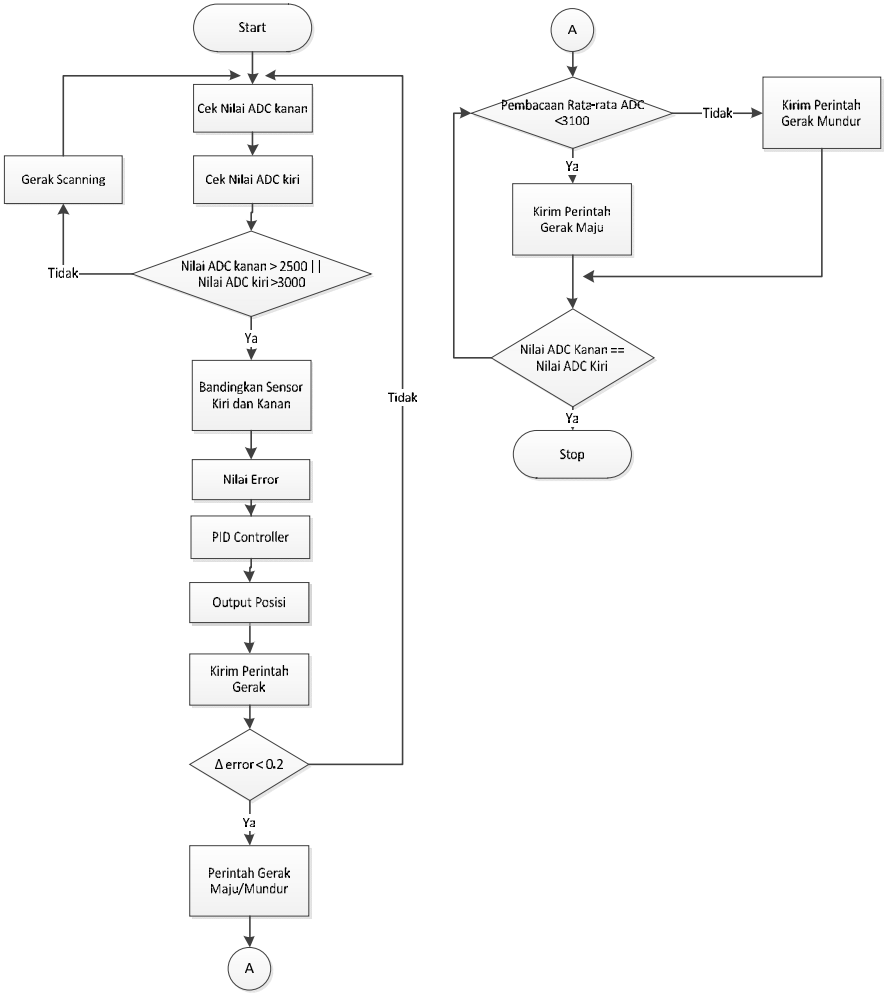
Proses Kontrol lengan robot tiga DOF mengambil nilai *error* sensor TGS2600. *Error* tersebut diperoleh dari hasil pengurangan antara *setpoint* dengan data *output sensor* TGS2600. Proses kontrol lengan robot tiga DOF dijelaskan pada gambar 3.8 sistem kontrol lengan robot tiga DOF. *Error* sensor adalah input dari proses kontrol *PID*.



Gambar 3.8 Sistem Kontrol Lengan Robot tiga DOF

Pergerakan lengan robot tiga DOF ini diatur melalui output PWM yang dihasilkan oleh *inverse kinematic*. Proses *inverse kinematic* mengolah masukan berupa posisi yang dihasilkan oleh output PID.

Output PID merupakan pengolahan data error yang dihasilkan oleh perbandingan pembacaan sensor[0] dan sensor[1] setelah pembacaan ADC tersebut dikurangkan dengan nilai set point dari masing-masing sensor.



Gambar 3.9 Diagram Pembacaan ADC Sensor Gas

Penjelasan :

Pada gambar 3.9, lengan robot akan melakukan gerakan scanning ketika sensor gas belum mendapatkan input berupa gas. Ketika sensor gas mendapatkan input, maka error akan didapatkan dengan mengurangkan set point dengan nilai pembacaan sensor. Kemudian Error akan dijadikan sebagai input pada kontrol PID yang akan menggerakkan lengan robot untuk memposisikan kedua sensor gas agar pembacaan antara kedua sensor sama. Kemudian ketika nilai dari Delta Error dari sensor bernilai kurang dari 0.2, maka lengan robot akan bergerak maju ketika nilai rata2_adc[0] bernilai kurang dari 3100 dan akan bergerak maju ketika nilai rata2_adc[0] bernilai lebih dar 0.2. Setelah bergerak maju dan mundur, akan dibandingkan kembali nilai rata-rata ADC = 0.2, apabila syarat tersebut terpenuhi maka robot akan berhenti dan apabila tidak, lengan robot akan bergerak maju dan mundur kembali.

```
switch(flag){

  case 0:
  reset_posisi();
  PID[0]=0;
  counter_set_awal ++ ;
  if(counter_set_awal > 50)
  {
    flag = 1 ;
  }
  case 1:
    counter_set_awal = 0;
        if((baca_sensor[0]<2500) && (baca_sensor[1]<2000)){
                        for(i=0;i<=7;i++){
                          geser(i,0,0); Delayms(50);
                        }

                        for(i=7;i>=-7;i--){
                          geser(i,0,0); Delayms(50);
                        }
        }
        else if((baca_sensor[0]>=2500) || (baca_sensor[1]>=2000)){
              flag=2;}
    break;
```

Gambar 3.10 Inisialisasi Gerakan Scanning

Penjelasan :

Pada mulanya, ketika semua supply diaktifkan, maka ADC sensor akan mengecek apakah nilai dari pembacaan ADC0 dan ADC1 pada PA5 dan PA7 apakah sudah melebihi 2500 dan 3000 . Apabila kondisi tersebut belum terpenuhi, maka lengan robot akan melakukan gerakan scanning yakni 7 gerakan ke kiri dan 7 gerakan ke kanan yang bergerak pada sumbu x horizontal seperti pada gambar 3.10.

Pada gambar 3.11, ketika kondisi pembacaan ADC tersebut terpenuhi, maka STM32F4 akan membandingkan manakah pembacaan nilai ADC yang lebih tinggi. Proses pembandingan nilai ADC ini sebelumnya masing-masing ADC0 dan ADC1 dibagi dengan nilai pembacaan ADC sensor TGS maksimum, yakni dibagi dengan 4095, sesuai dengan ADC pada STM32F4 yang memiliki resolusi sebesar 12bit. Setelah dibagi dengan nilai maksimal, barulah kedua nilai tersebut dibandingkan.

```
        TM_ADC_Init(ADC1, ADC_Channel_4); //ADC1 ch 4, PA4
        TM_ADC_Init(ADC1, ADC_Channel_5); //ADC1 ch 5, PA5
        TM_ADC_Init(ADC1, ADC_Channel_6); //ADC1 ch 6, PA6
        TM_ADC_Init(ADC1, ADC_Channel_7); //ADC1 ch 7, PA7

a++;  if(a>=20)
{ a=0;

      baca_sensor[0]=TM_ADC_Read(ADC1, ADC_Channel_5);
      baca_sensor[1]=TM_ADC_Read(ADC1, ADC_Channel_7);


      sensor[0]= (TM_ADC_Read(ADC1, ADC_Channel_5))  / set_adcA;    //kiri
      sensor[1]= (TM_ADC_Read(ADC1, ADC_Channel_7))  / set_adcB;    //kanan


      delta_error[0]=set_adc-sensor[0];
            if(delta_error[0]<0){
                delta_error[0]=delta_error[0]*(-1);
            }
            //else{delta_error[0]=delta_error[0];}

      delta_error[1]=set_adc-sensor[1];
            if(delta_error[1]<0){
                delta_error[1]=delta_error[1]*(-1);
            }
```

Gambar 3.11 Inisialisasi ADC Sensor Gas dan Perhitungan Delta Error

Setelah didapatkan manakah nilai yang lebih besar, maka selanjutnya adalah menghitung nilai error[0], yakni dengan cara mengurangkan nilai set_adc = 0.5 dengan nilai perbandingan pembacaan

ADC tadi yang paling besar. Setelah nilai error[0] didapatkan, selanjutnya nilai tersebut akan diolah oleh kontrol PID, baik kontrol proporsional, integral, atau derivatif. Dari nilai output PID[0], maka akan dikeluarkan sebagai output posisi pada perintah gerak(PID[0],0,0) seperti pada gambar 3.12.

```
case 2:
      if(sensor[0]<sensor[1]){
          temp_error=set_adc-sensor[1];
              if (temp_error<0){
                  temp_error=(-1)*temp_error;
              }

          error[0]=(-1)*temp_error;
      }
      else if(sensor[0]>sensor[1]){
          temp_error2=set_adc-sensor[0];
          if (temp_error2<0){
                  temp_error2=(-1)*temp_error2;
              }

          error[0]=temp_error2;
      }

      //kontrol proporsional
      proporsional[0]=error[0]*konsp;
      //kontrol Integral
      errorI=errorI+error[0];
      integral[0]=(errorI)*konsi;
      errorI=error[0];
      //kontrol derivatif
      derivatif[0]=(error[0]-errorD)*konsd;
      errorD=error[0];
      PID[0]=proporsional[0]+derivatif[0]+integral[0];
      if(PID[0]>7){PID[0]=7;}
      if(PID[0]<-7){PID[0]=-7;}
```

Gambar 3.12 Perhitungan error[0] dan kendali PID

Pada akhirnya, setelah didapatkan output posisi yang akan dijadikan masukan pada inverse kinematic, sensor TGS1 dan TGS2 akan mengecek kembali pembacaan ADC0 dan ADC1 apakah kurang dari batas minimal pembacaan ADC. Jika kondisi ini terpenuhi, maka proses akan kembali pada pengecekan sensor kanan dan kiri pada awal. Jika syarat ini tidak terpenuhi, maka sistem akan menyimpan nilai PID lama sekaligus melakukan pengecekan apakah nilai dari delta error sudah

kurang dari 0,01. Jika nilai ini terpenuhi, maka proses akan berhenti seperti pad gambar 3.13.

```
if(delta_error[0]<0.01 && delta_error[1]<0.01){
      PID[0]=PID[0];
      geser(PID[0],0,0);
}

if((baca_sensor[0]<2500) && (baca_sensor[1]<2000)){
      flag=0;
}
```

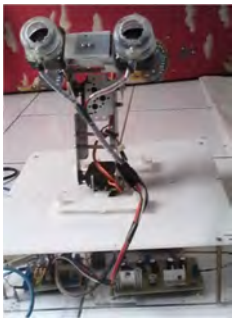
Gambar 3.13 Cek nilai delta error dan cek pembacaan minimal sensor

Variabel delta_error[0] dan delta_error[1] merupakan pengurangan dari pembacaan ADC0 dan ADC1 yang telah bagi dengan masing-masing nilai pembacaan maksimal ADC-nya yang kemudian dikurangkan dengan nilai dari set_adc yang bernilai 0.5.

Ketika nilai delta_error ini tidak terpenuhi, maka proses perhitungan PID ataupun pembacaan nilai ADC masih terpenuhi sesuai dengan treshold minimal akan terus berlangsung.

BAB IV
PENGUKURAN DAN ANALISIS SISTEM

Pengujian sistem kontrol lengan robot tiga DOF dibagi menjadi beberapa tahap bagian, dimulai dari pengujian perangkat keras, dan perangkat lunak. Tujuan dari pengujian dari berbagai aspek dari perancangan ini untuk mendapatkan parameter atau evaluasi performa dari perancangan kontrol lengan robot tiga DOF.



Gambar 4.1 Lengan Robot tiga DOF

4.1 Pengujian Perangkat Keras

Pengujian perangkat keras dilakukan dengan menguji hasil output dari tiap sub rangkaian elektrik secara keseluruhan. Pengujian yang dilakukan meliputi pengujian dari *supply* dari *IC regulator* yang digunakan, pengujian Rangkaian *sensor*, dan pengujian *servocontroller*. Pengujian dilakukan agar memenuhi batasan – batasan masalah pada tugas akhir ini.

4.1.1 Pengujian *Supply* Kontrol lengan robot tiga DOF

Rangkaian *power supply* merupakan rangkaian terpenting yang digunakan sembagai sumber daya dari seluruh sub rangkaian elektrik. Perangkat *supply* berfungsi mengkonversikan tegangan AC menjadi tegangan DC. Output dari rangkaian ini adalah tegangan DC 9V, 6V, dan 5V. Hasil dari pengujian ditunjukkan pada table 4.1. Tegangan input DC yang sudah melewati rangkaian *filter* digunakan

sebagai tegangan input. Pada rangkaian *power supply* menggunakan *IC regulator* yaitu IC 7805, 7806, dan 7809.

Tabel 4.1 pengujian output Rangkaian *supply.*

Vin (Volt)	Hasil Pengukuran Vout(Volt)	Target Vout (volt)
12.00	9.27	9.00
12.00	5.76	6.00
12.00	4,99	5.00

(a) (b)

Gambar 4.2 (a) Pengujian Vout 9 volt, dan (b) Pengujian Vout 9 volt

Untuk mendapatkan tegangan 9V, 6V, dan 5V pada rangkaian *power supply* menggunakan rangkaian *converter DC to DC* IC regulator 7809, 7806, dan 7805. Dari hasil pengujian seperti pada gambar 4.2, terdapat error pada Vout 9V dan Vout 5V hal ini disebabkan pengaruh IC regulator. IC regulator memiliki kelemahan relative rendahnya kemampuan mengeluarkan arus listrik berkisar 700mA.

4.1.2 Pengujian Rangkaian *Sensor*

Rangkaian *sensor* merupakan rangkaian pembagi tegangan sensor TGS 2600. Pada pengujian ini tegangan masukan pada Rangkaian sensor 5 Volt. Hal yang di uji dari Rangkaian sensor adalah Vin, dan tegangan stabil pada sensor. Pengujian ini diperlihatkan pada gambar 4.3.

Gambar 4.3 Vin driver sensor

Vin rangkaian *sensor* adalah 5V, terdapat *error* 0,03V. Tegangan 5,03 V masih batas toleransi Vin sensor karena, *Vsupply* sensor TGS 2600 antara 4,8V-5,2V. Kecilnya *error* tidak berpengaruh besar pada rangkaian elektrik *sensor*.

4.1.3 Pengujian *Servocontroller*

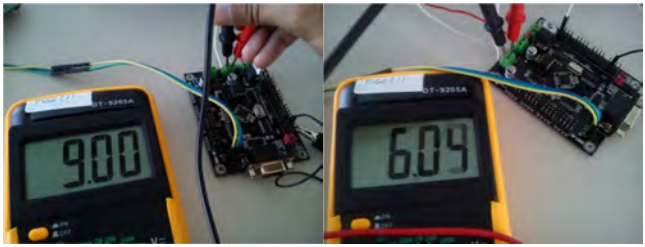
Servocontroller berfungsi mengirimkan data PWM ke motor servo. Pengujian *servocontroller* meliputi dua pengujian yaitu, Pengujian Vin *servocontroller*, dan pengujian output PWM pada *servocontroller*.

4.1.3.1 Pengujian Vinput Pada *servocontroller*

Mengaktifkan *servocontroller* memerlukan *Vsupply* 9V, dan motor servo *Vsupply* 6V. Pengujian ini diperlukan untuk memastikan *Vsupply* yang diterima *servo controller* sesuai seperti pada tabel 4.2.

Tabel 4.2 Pengujian Vinput pada servocontroller

Keterangan	Vinput (V)	Target Vin (V)	Error (%)
Servocontroller	9.00	9.00	0
Motor Servo	6.04	6.00	0.67



(a) (b)

Gambar 4.4 (a) *Vsupply Logic servocontroller*, (b) *Vin servo*

Dari pengujian tegangan masukan servo controller seperti pada gambar 4.4, terdapat error 0.67 % pada pengujian ini dipengaruhi nilai arus yang masih besar karena belum ada beban dari. *Error* yang kecil tidak berpengaruh besar pada rangkain elektrik karena masih dalam batas toleransinya.

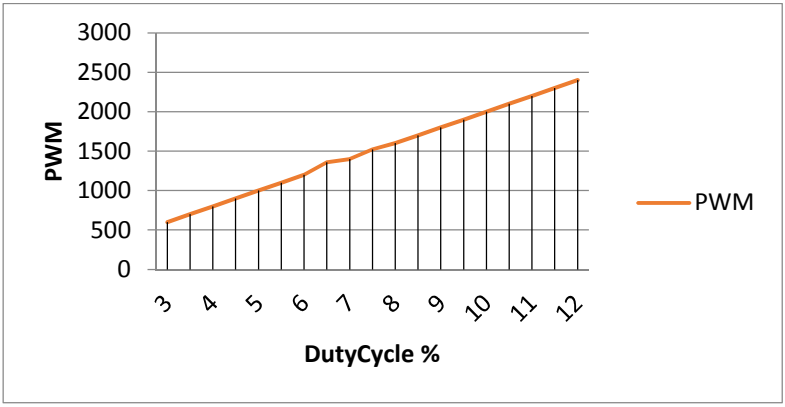
4.1.3.2 Pengujian Output PWM pada *servocontroller*

Pengujian output PWM servokontroller dilakukan dengan mengirimkan data PWM dari komputer melalui komunikasi serial dan Output PWM ditampilkan pada *oscilloscope* seperti pada gambar 4.5. Tujuan dari pengujian ini adalah melihat *duty cycle* pada Output PWM. Data pengujian dapat dilihat pada abel 4.3.

Gambar 4.5 Proses pengujian *output PWM servocotroller*

Tabel 4.3 Pengujian *Output PWM servocontroller*

Pengujian	Data Serial PWM	Duty Cycle teruji (%)
1	700	3.5
2	900	4.56
3	1100	5.4
4	1300	6.5
5	1500	7.5
6	1700	8.5
7	1900	9.5
8	2100	10.5
9	2300	11.5
10	2400	12

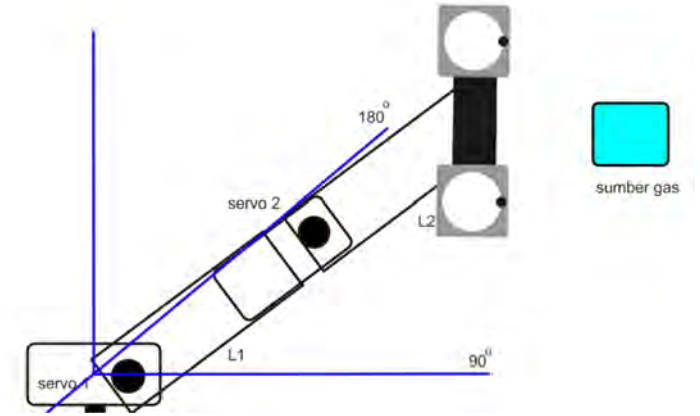


Gambar 4.6 Grafik pengujian *Output PWM Servocontroller*

Dari gambar 4.6 *Duty cycle* tersebut linier dengan nilai PWM, umumnya sinyal PWM memiliki *amplitude* dan frekuensi dasar yang tetap, namun memiliki lebar pulsa yang berbeda. *Duty cycle* adalah lamanya *pulsa high* dalam satu periode.

4.1.4 Pengujian Sensor Gas

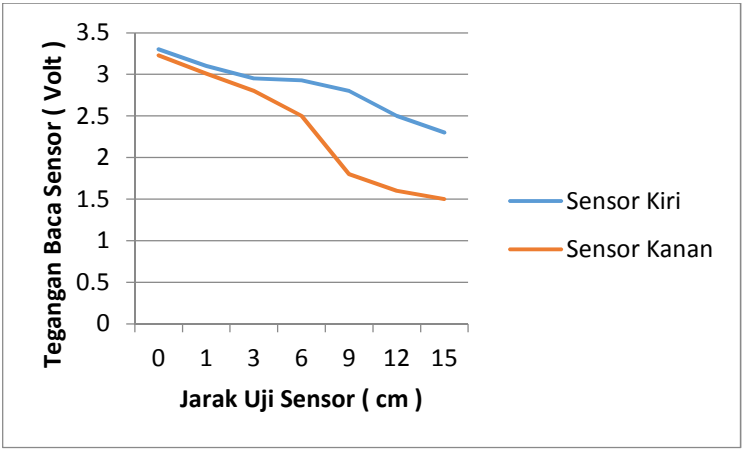
Pengujian sensor gas meliputi tiga pengujian, yaitu ketika sumber gas berada diposisi 90° . Pengujian ini dilakukan saat suhu udara dalam keadaan normal, dan dengan bantuan pompa menarik udara melewati tabung sensor. Pengujian ini dilakukan dengan gas Alkohol yang termasuk dalam golongan *ethanol*. Posisi sumber gas dan posisi sensor diilustrasikan pada gambar 4.7.



Gambar 4.7 Ilustrasi pengujian sensor

Tabel 4.4 Pengujian Pembacaan ADC Sensor

Jarak (cm)	Sensor Kiri (volt)	Sensor Kanan (volt)
0	3,3	3,23
1	3,1	3,01
3	2,95	2,8
6	2,93	2,5
9	2,8	1,8
12	2,5	1,6
15	2,3	1,5



Gambar 4.8 Ilustrasi pengujian sensor

Analisa :

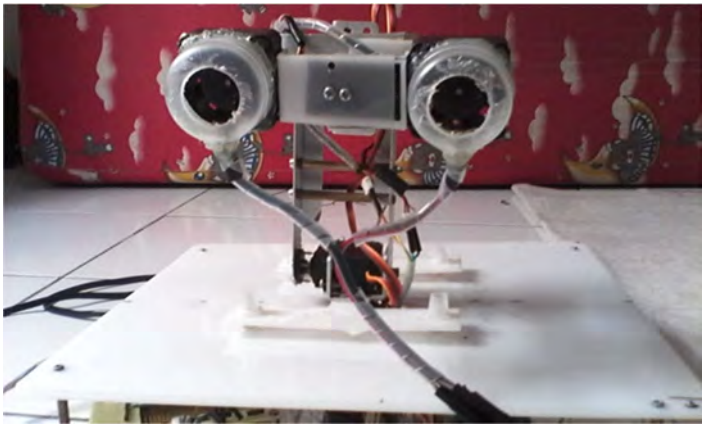
Hasil pengujian pada tabel 4.4 ketika sumber gas Alkohol berada pada posisi 90°, dan sensor gas TGS 2600 berada diposisi *center* yaitu 90° Horizontal. Pembacaan ADC kedua sensor bernilai linier, dimana nilai maksimal dari ADC sebesar 12 bit (4095) akan didapatkan ketika sumber gas berupa alkohol didekatkan pada jarak 3 cm dari sensor TGS2600 seperti pada gambar 4.8.

Dari pengujian ini dapat disimpulkan ketika sumber gas alkohol berada diposisi 90° sensor kiri mendapatkan ADC lebih kecil dibandingkan sensor kanan. Hal ini disebabkan perbedaan kepekaan masing-masing sensor. Sehingga, untuk mengatasi perbedaan nilai pembacaan kedua sensor TGS2600 dilakukan metode linierisasi. Metode ini dilakukan dengan membagi nilai pembacaan ADC sensor dengan nilai maksimal ADC, yakni sebesar 4095. Hal ini bertujuan agar nilai pembacaan ADC kedua sensor bernilai setara dan dapat dibandingkan meskipun nilai tegangan awal pendeteksian gas alkohol antara sensor 0 dan sensor 1 berbeda.

4.2 Pengujian Seluruh Sistem

Pengujian seluruh sistem dilakukan dengan lengkap dimana keberasilan pendeteksian gas menjadi parameter dari pengujian ini. Pengujian dilakukan dengan bantuan pompa penyedot pada *tube*

sensor gas untuk mengaliri udara melewati sensor gas. Posisi awal dari sumber gas berada pada sudut 90° pada sumbu horizontal seperti pada gambar 4.9.

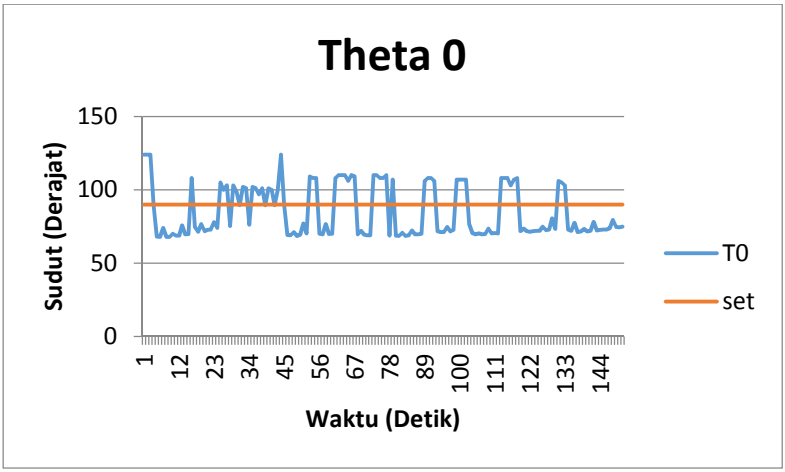


Gambar 4.9 Tampak depan lengan robot tiga DOF

Pengujian ini bertujuan mendapatkan sistem berjalan sesuai perancangan. Berikut adalah uji kendali P, PD, dan PID pada lengan robot. Nilai persen error didapatkan dengan membagi nilai dari delta error dari setiap theta dengan set point sudut yang kemudian dikalikan dengan 100%.

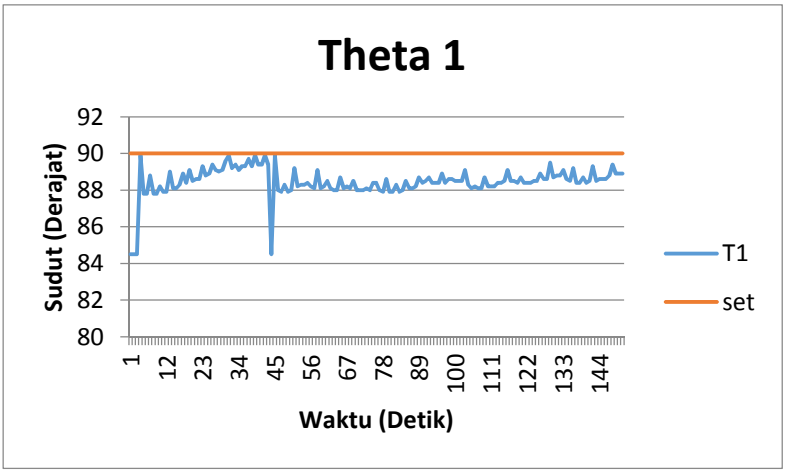
4.2.1 Pengujian Kp= 10

Pada pengujian berikut, nilai konstanta pengali pada kendali proporsional adalah 10. Pengujian dilakukan pada saat lengan robot melakukan scanning dan posisi sumber gas berada pada sudut 90^{o} pada sumbu horizontal.



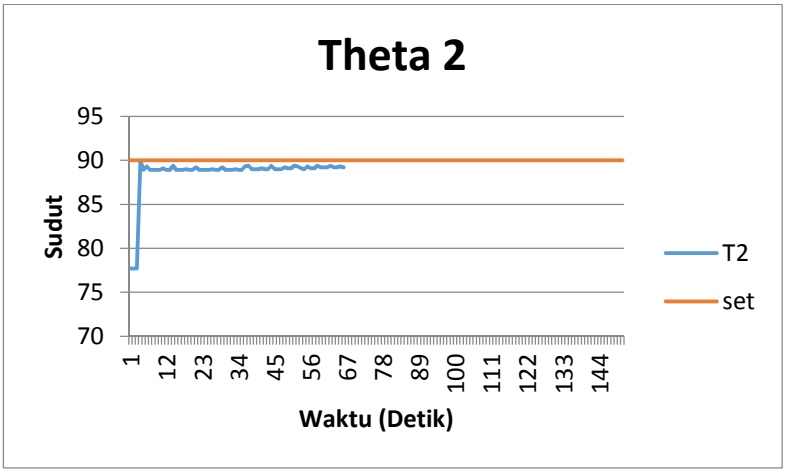
Gambar 4.10 Pengujian Servo 0 Dengan Sumber Uji Gas Pada 90^{o}

Pada pengujian ini, lengan robot memiliki nilai rata-rata error 19,12%.



Gambar 4.11 Pengujian Servo 1 Dengan Sumber Uji Gas Pada 90^{o}

Pada pengujian ini, lengan robot memiliki nilai rata-rata error sebesar 1,706%.



Gambar 4.12 Pengujian Servo 2 Dengan Sumber Uji Gas Pada 90^o

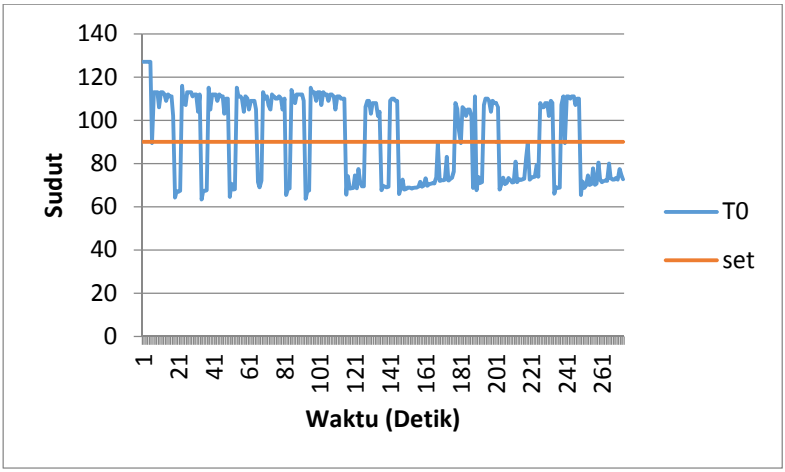
Pada pengujian ini, lengan robot memiliki nilai rata-rata error sebesar 3,9704%.

Analisa :

Pada pengujian lengan robot dengan Kp=10, theta 0 pada gambar 4.10 akan selalu berosilasi pada nilai maksimal dan minimalnya, karena nilai error akan selalu dilakukan dengan 10, sedangkan pada theta1 dan theta 2 akan mempertahankan posisi mendekati 90^o dengan $\Delta error < 0.2$ seperti pada gambar 4.11 dan gambar 4.12.

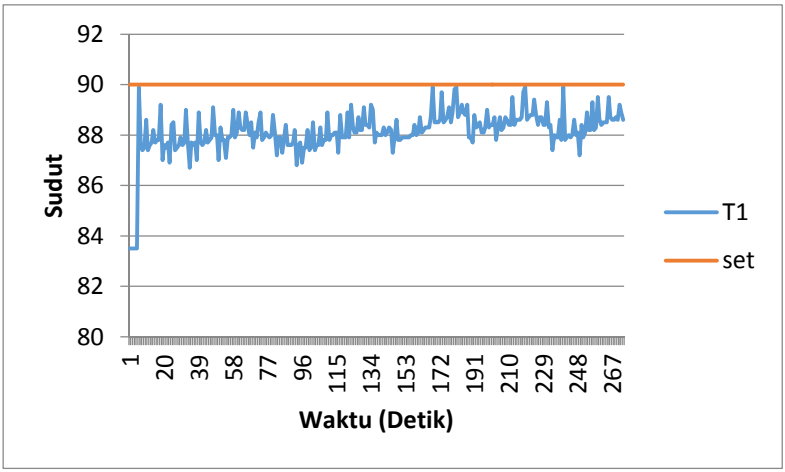
4.2.2 Pengujian Kp= 10 Kd=1

Pada pengujian berikut, nilai konstanta pengali pada kendali proporsional adalah 10 dan derivatif adalah 1. Pengujian dilakukan pada saat lengan robot melakukan scanning dan posisi sumber gas berada pada sudut 90^o pada sumbu horizontal.



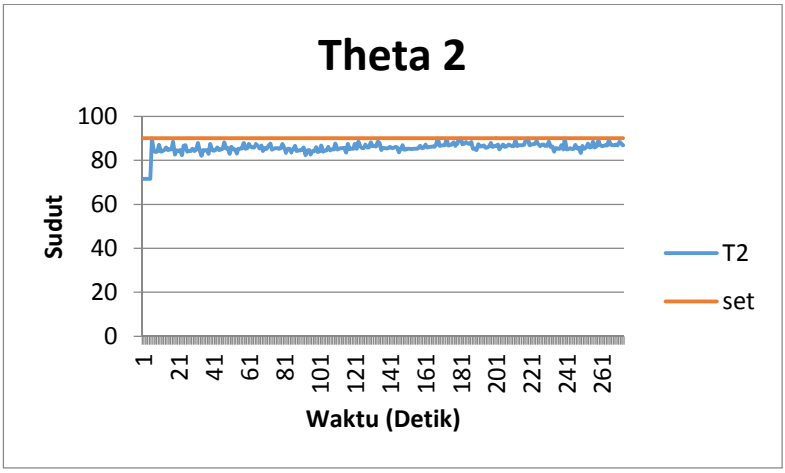
Gambar 4.13 Pengujian Servo 0 Dengan Sumber Uji Gas Pada 90^{o}

Pada pengujian ini, lengan robot memiliki nilai rata-rata error 21,286%.



Gambar 4.14 Pengujian Servo 1 Dengan Sumber Uji Gas Pada 90^{o}

Pada pengujian ini, lengan robot memiliki nilai rata-rata error sebesar 2,065%.



Gambar 4.15 Pengujian Servo 2 Dengan Sumber Uji Gas Pada 90^o

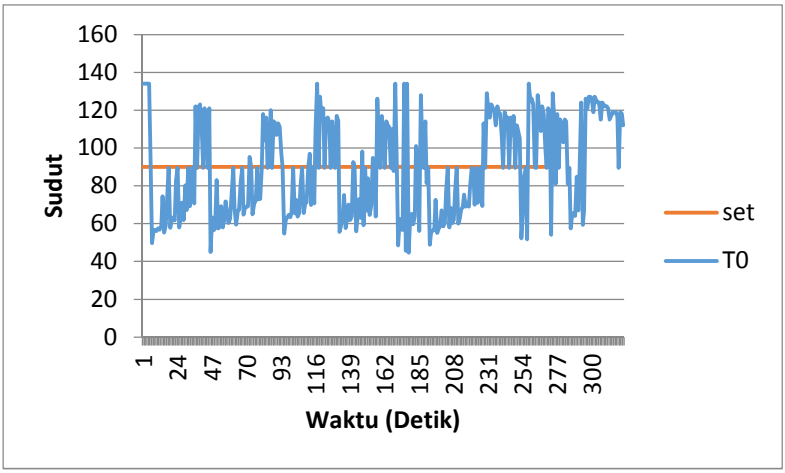
Pada pengujian ini, lengan robot mencapai kondisi steadystate pada detik ke 6, dan kendali ini memiliki nilai rata-rata error sebesar 4,768%.

Analisa :

Pada pengujian lengan robot dengan Kp=10 dan Kd=1, theta 0 pada gambar 4.13 akan selalu berosilasi pada nilai maksimal dan minimalnya, karena nilai error akan selalu dilakukan dengan 10 dan dengan adanya penambahan nilai Kd=1, error overshoot pada lengan robot akan lebih kecil dibandingkan pada Kp=10 tanpa ada penambahan nilai pengali Kd=1, sedangkan pada theta1 dan theta 2 akan mempertahankan posisi mendekati 90^o dengan $\Delta error < 0.2$ seperti pada gambar 4.14 dan gambar 4.15.

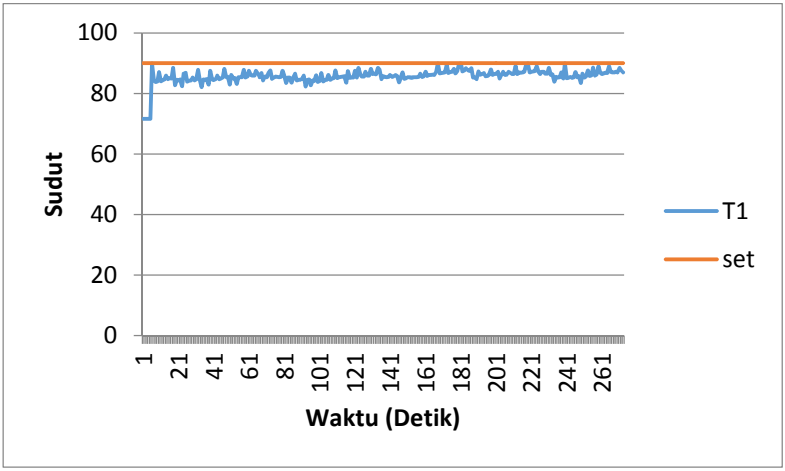
4.2.3 Pengujian Kp= 25 Kd=10

Pada pengujian berikut, nilai konstanta pengali pada kendali proporsional adalah 25 dan derivatif adalah 10. Pengujian dilakukan pada saat lengan robot melakukan scanning dan posisi sumber gas berada pada sudut 90^o pada sumbu horizontal.



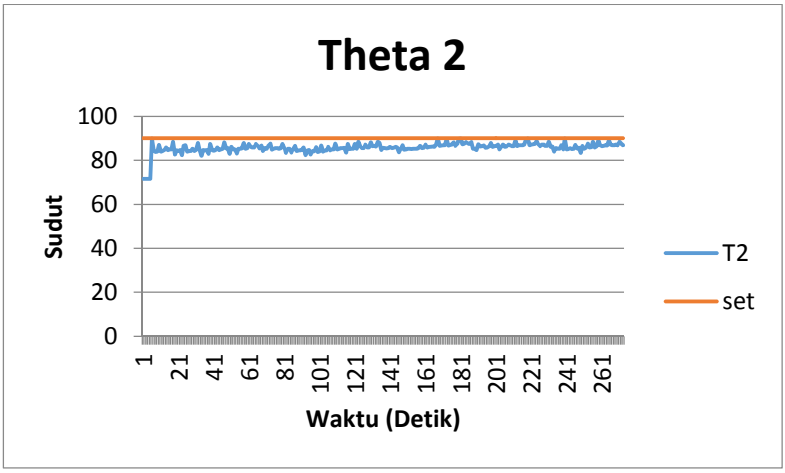
Gambar 4.16 Pengujian Servo 0 Dengan Sumber Uji Gas Pada 90^{o}

Pada pengujian ini, lengan robot memiliki nilai rata-rata error 24,164%.



Gambar 4.17 Pengujian Servo 1 Dengan Sumber Uji Gas Pada 90^{o}

Pada pengujian ini, lengan robot memiliki nilai rata-rata error sebesar 3,154%.



Gambar 4.18 Pengujian Servo 2 Dengan Sumber Uji Gas Pada 90^o

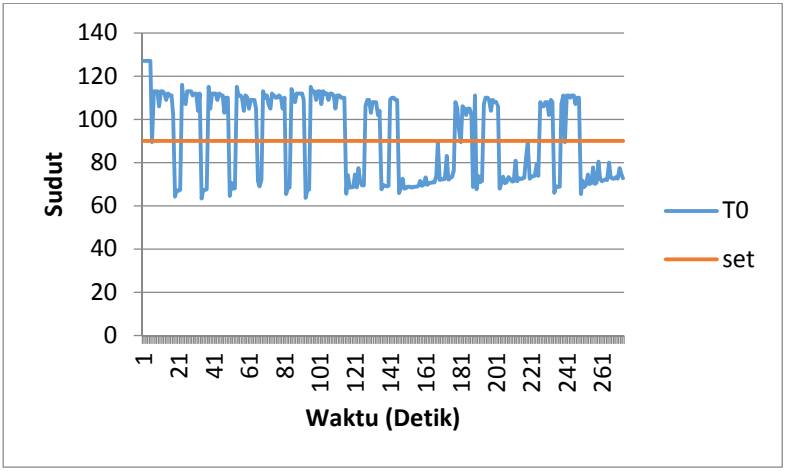
Pada pengujian ini, lengan robot memiliki nilai rata-rata error sebesar 8,762%.

Analisa :

Pada pengujian lengan robot dengan Kp=25 dan Kd=10, theta 0 pada gambar 4.16 akan selalu berosilasi pada nilai maksimal dan minimalnya, karena nilai error akan selalu dilakukan dengan 25 dan dengan adanya penambahan nilai Kd=10, *error overshoot* pada lengan robot akan lebih kecil, karena pada pengendali derivatif adalah pengendali untuk mengurangi error dari overshoot. Hal ini disebabkan karena pengendali derivatif adalah pengurangan nilai error yang sekarang dengan nilai error yang sebelumnya, sehingga nilai PID akan semakin besar. Sedangkan pada theta1 dan theta 2 akan mempertahankan posisi mendekati 90^o dengan $\Delta error < 0.2$ seperti pada gambar 4.17 dan gambar 4.18.

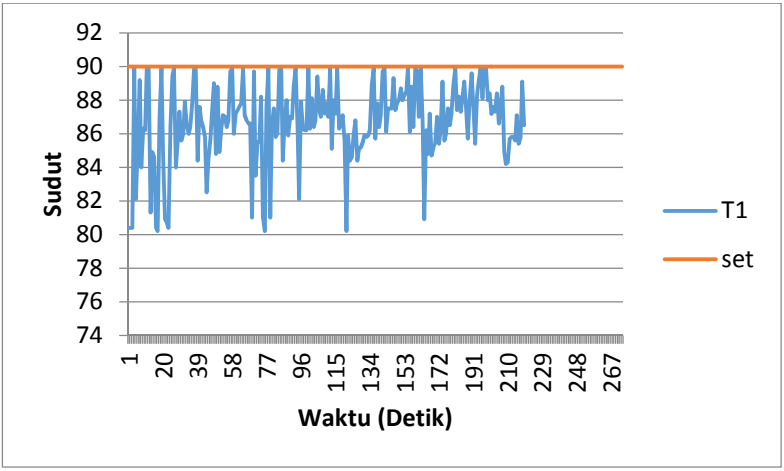
4.2.4 Pengujian Kp= 25 Kd=10 Ki=0.1

Pada pengujian berikut, nilai konstanta pengali pada kendali proporsional adalah 25, derivatif adalah 10, dan Ki=0.1. Pengujian dilakukan pada saat lengan robot melakukan scanning dan posisi sumber gas berada pada sudut 90^{o} pada sumbu horizontal.



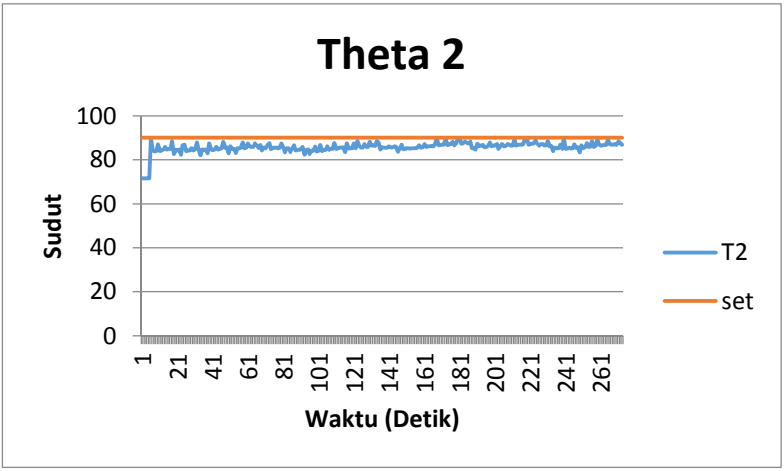
Gambar 4.19 Pengujian Servo 0 Dengan Sumber Uji Gas Pada 90^{o}

Pada pengujian ini, lengan robot memiliki nilai rata-rata error 24,044%.



Gambar 4.20 Pengujian Servo 1 Dengan Sumber Uji Gas Pada 90^{o}

Pada pengujian ini, lengan robot memiliki nilai rata-rata error sebesar 3,3301%.



Gambar 4.21 Pengujian Servo 2 Dengan Sumber Uji Gas Pada 90^{o}

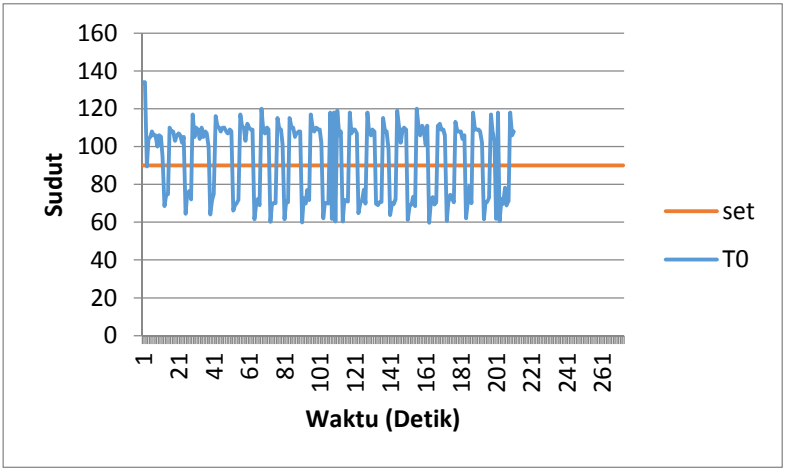
Pada pengujian ini, lengan robot memiliki nilai rata-rata error sebesar 8,859%.

Analisa :

Pada pengujian lengan robot dengan Kp=25, Kd=10 dan Ki=0.1, theta 0 pada gambar 4.19 akan selalu berosilasi pada nilai maksimal dan minimalnya, karena nilai error akan selalu dilakukan dengan 25 dan dengan adanya penambahan nilai Kd=10 dan Ki=0,1, *error* steadystate pada lengan robot akan lebih kecil karena pengendali integral dapat mengurangi *error steadystate*. Sedangkan pada theta1 dan theta 2 akan mempertahankan posisi mendekati 90^o dengan $\Delta error < 0.2$ seperti pada gambar 4.20 dan gambar 4.21.

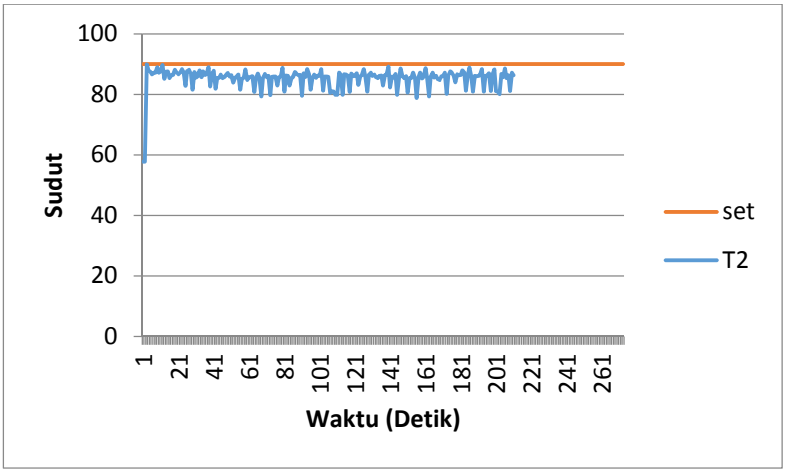
4.2.5 Pengujian Kp= 10 Kd=3 Ki=0.0001

Pada pengujian berikut, nilai konstanta pengali pada kendali proporsional adalah 10, derivatif adalah 3, dan Ki=0.0001. Pengujian dilakukan pada saat lengan robot melakukan scanning dan posisi sumber gas berada pada sudut 90^o pada sumbu horizontal.



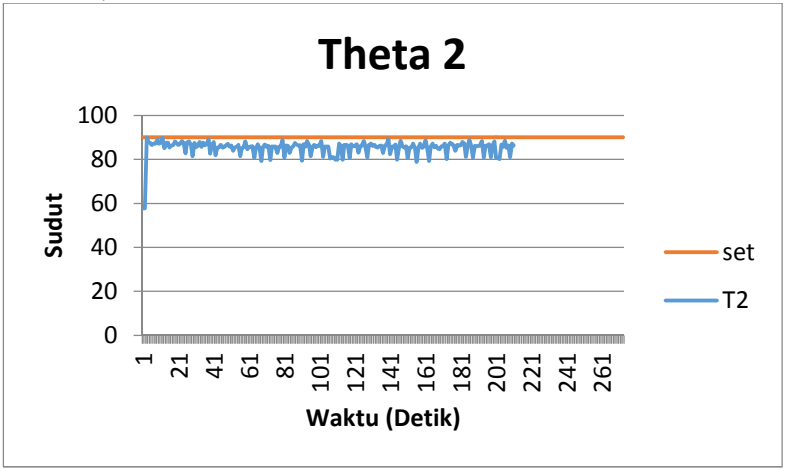
Gambar 4.22 Pengujian Servo 0 Dengan Sumber Uji Gas Pada 90^o

Pada pengujian ini, lengan robot memiliki nilai rata-rata error 22,003%.



Gambar 4.23 Pengujian Servo 1 Dengan Sumber Uji Gas Pada 90^{o}

Pada pengujian ini, lengan robot memiliki nilai rata-rata error sebesar 2,239%.



Gambar 4.24 Pengujian Servo 2 Dengan Sumber Uji Gas Pada 90^{o}

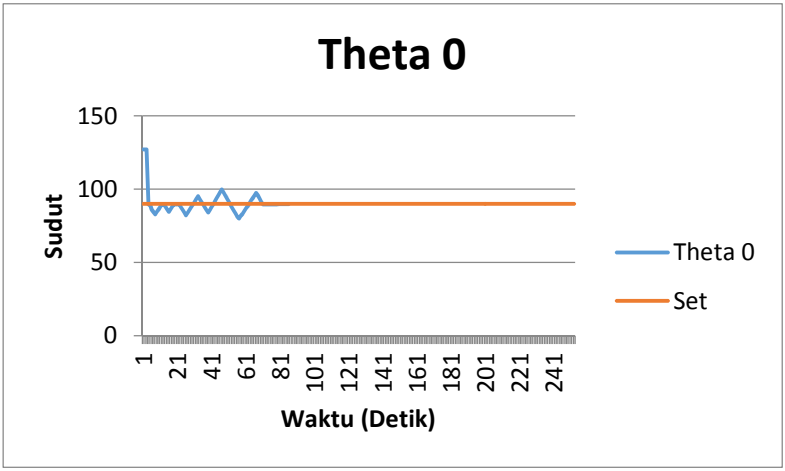
Pada pengujian ini, lengan robot memiliki nilai rata-rata error sebesar 5,299%.

Analisa :

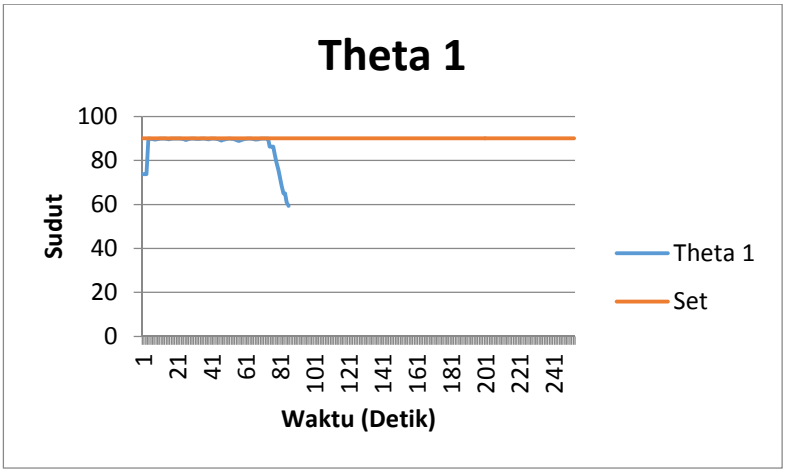
Pada pengujian lengan robot dengan Kp=10 dan Kd=3 dan Ki=0.0001, theta 0 pada gambar 4.22 akan selalu berosilasi pada nilai maksimal dan minimalnya, karena nilai error akan selalu dilakukan dengan 10 dan dengan adanya penambahan nilai Kd=3 dan Ki=0.0001, *error* steadystate dan *overshoot* pada lengan robot akan lebih kecil karena pada pengendali integral dapat memperkecil *error* dari *steadystate* dan pengendali derivatif dapat mengurangi *overshoot*. Sedangkan pada theta1 dan theta 2 akan mempertahankan posisi mendekati 90^o dengan $\Delta error < 0.2$ seperti pada gambar 4.23 dan gambar 4.24.

4.2.6 Pengujian Gerak Maju dengan Kp= 1 Kd=0,09 Ki=0.0001

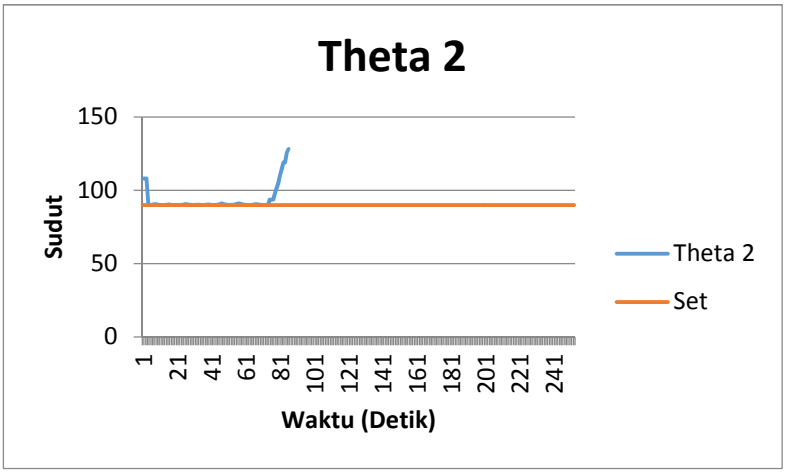
Pada pengujian ini, nilai konstanta pengali Kp= 1, Kd= 0.09, dan Ki=0.0001 diberikan untuk mendapatkan gerakan maju dan mundur setelah lengan robot telah memposisikan pada sudut 90^o dengan $\Delta error < 0.2$.



Gambar 4.25 Pengujian Servo 0 Dengan Sumber Uji Gas Pada 90^o



Gambar 4.26 Pengujian Servo 1 Dengan Sumber Uji Gas Pada 90^{o}



Gambar 4.27 Pengujian Servo 2 Dengan Sumber Uji Gas Pada 90^{o}

Analisa:

Pada pengujian gerak maju mundur, lengan robot akan berusaha untuk memposisikan theta 0 pada sudut 90^{o} seperti pada gambar 4.25 pada detik 70. Setelah theta 0 berada pada posisi sudut 90o dengan *delta*

error < 0.2, maka theta 1 dan theta 2 akan menggerakkan lengan robot untuk maju seperti pada gambar 4.26 dan gambar 4.27.

4.3 Evaluasi Sistem

Pengujian sistem secara keseluruhan lengan robot bergerak mencari sumber gas dengan metode kontrol *PID* seperti pada tabel 4.4, dari pengujian dengan nilai Kp=10 pada sudut pengujian 90 didapatkan nilai rata-rata error pada theta0, theta1, theta2 adalah 19,12%, 1,706%, dan 3,9704%. Dan dengan penambahan kendali derivatif dengan kd=1, didapatkan nilai rata-rata error pada theta0, theta1, theta2 adalah 21,286%, 2,065%, dan 4,768%.

Tabel 4.4 Rata-rata Error Pada Setiap Servo Pada Pengujian PID

	Kp=10	Kp=10 Kd=1	Kp=25 Kd=10	Kp=25 Kd=10 Ki=0.1	Kp=10 Kd=3 Ki=0.0001	Kp=1 Kd=0.09 Ki=0.0001
Theta 0	19,12%	21,28%	24,16%	24,04%	22,00%	4.40%
Theta 1	1,70%	2,065	3,15%	3,33%	2,23%	1.10%
Theta 2	3,97%	4,768%	8,762%	8,85%	5,29%	1.07%

Pada pengujian ketiga, dengan menambahkan Ki=0.1 pada sudut pengujian 90 didapatkan nilai rata-rata error pada theta0, theta1, theta2 adalah 24,044%, 3,3301%, dan 8,859%. Pengujian selanjutnya adalah dengan mengubah ketiga konstanta pengali, dimana Kp=10 Kd=3 dan Ki=0.0001, hasil yang didapatkan adalah error masing-masing sudut yakni pada theta0, theta1, theta2 adalah 22,003%, 2,239%, dan 5,299%. Dan pada pengujian yang terakhir, yakni dengan nilai konstanta pengali Kp=1, Kd=0.09 dan Ki=0.0001, didapatkan nilai error dari masing-masing sudut pada theta0, theta1, theta2 adalah 4,401%, 1,109%, dan 1,072%. Dari kelima percobaan kendali PID dengan metode trial and error, kendali yang paling baik adalah kendali dengan Kp=1, Kd=0.09 dan Ki=0.0001, kendali ini dipilih karena memiliki nilai error yang paling kecil dibandingkan dengan konstanta pengali pada percobaan yang dilakukan sebelumnya.

Pada pengujian gerakan maju mundur dengan nilai kendali Kp=1 Kd=0.09 dan Ki=0.0001, lengan robot akan berusaha untuk memposisikan *end-of-effector* pada sudut 90^{o} horizontal dengan nilai $\Delta error < 0.2$, kemudian lengan robot akan membandingkan nilai pembacaan ADC[0]

dan ADC[1] rata-rata yang sekarang dengan nilai pembacaan ADC rata-rata sebelumnya. Apabila nilai pembacaan nilai rata-rata sekarang lebih kecil dari yang sebelumnya, maka lengan robot akan bergerak maju.

Pengujian mekanik dimulai dengan pengujian tegangan suplai di setiap rangkaian elektrik, Pengujian selanjutnya adalah pengujian kerja sensor melihat seberapa jauh daerah jangkauan pendeteksian sumber gas oleh sensor TGS2600, pengujian ini dilakukan 5 kali pengujian dengan jarak yang berbeda dan posisi sumber gas berada pada 90^{o} dari lengan robot. Didapatkan jarak estimasi kemampuan sensor gas mendeteksi sumber gas adalah dengan jarak 15 cm.

BAB V
PENUTUP

5.1 Kesimpulan

Dari perancangan, realisasi, dan pengujian alat pada tugas akhir ini dapat disimpulkan sensor TGS2600 mendeteksi sumber gas berupa alkohol dengan jarak maksimal 15 cm dan posisi jangkauan pendeteksian sumber gas dari 0° – 180°. Nilai pembacaan sensor 0 dan sensor 1 pada sumber gas alkohol dengan jarak 1cm adalah 3,3 volt. Pembacaan sensor 0 dan sensor 1 pada jarak 15cm adalah 2,3 volt dan 1,5 volt. Karena adanya perbedaan pembacaan nilai ADC pada dua sensor TGS2600, diterapkan metode linierisasi. Lengan robot bergerak 3DOF sesuai dengan metode PID. Lengan robot akan akan bergerak maju ketika nilai rata-rata pembacaan ADC sekarang lebih kecil dari nilai pembacaan rata-rata sebelumnya. Pada implementasi kendali PID dengan konstanta Kp=1 Kd=0,09 Ki=0.0001 adalah nilai konstanta pengali PID pada lengan robot 3DOF yang mampu untuk mencari sumber gas berupa alkohol menggunakan STM32F4 dengan nilai error pada masing-masing servo 4,401%, 1,109%, dan 1,072%.

.

5.2 Saran

Dalam mendesain sebuah rangka mekanik lengan robot diperlukan bahan yang kokoh dan ringan, karena kemampuan rangka lengan robot yang baik dapat memberikan gerakan lengan robot presisi sesuai dengan kinematik yang digunakan. Penggunaan rangka berupa aluminium ringan dapat mengurangi beban pada yang diterima oleh masing-masing servo, sehingga servo akan lebih tahan lama. Berdasarkan dari kelemahan sistem diperlukan rencana kedepan dalam penelitian ini, yaitu diperlukan tambahan sensor pada atas atau bawah dari kedua sensor yang telah digunakan, sehingga lengan robot 3DOF dapat mendeteksi sumber gas dari arah vertikal lengan robot. Perancangan kontrol PID akan lebih bagus jika tunning nilai kp, ki, kd menggunakan metode Zieger Nichlos. Penggunaan metode ini dapat mempercepat pengguna untuk mendapatkan kisaran nilai konstanta pengali PID dengan lebih cepat dibandingkan dengan melakukan metode *tunning*.

LAMPIRAN

Program Keil

```
void homing(int _x,int _y,int _z);
void inverse_kinematic();
void reset_posisi();
void geser(float _x,float _y,float _z);
void test_3();

#include "stm32f4xx.h"
#include "defines.h"
#include "tm_stm32f4_delay.h"
#include "tm_stm32f4_servo.h"
#include "tm_stm32f4_adc.h"
#include "tm_stm32f4_usart.h"
#include "gpio.h"
#include <stdio.h>
#include <math.h>
#include <string.h>


float rad_pulse=57.27272727;        //konversi dari radian ke degree 180 per Phi
float l1=12;
float l2=13;
float sensor[4];
float koordinat[3];
float x[3];float y[3];float z[3];
float temp[3];
float tangan_0, tangan_1, tangan_2;
int i,a;
char kirimservo[255],kirimPID[255];
char tes_out[255];
int PID_2, PID_3;
int flag=0;

//float konsi;
//float konsd;
```

```
//setting input PID & SET POINT
float set_adcA=3950;
                                //kiri
float set_adcB=4095;
                                //kanan
float set_adc=0.5;

//PENGALI PID----------------
float tei=10.4, ted=41.67;
//float konsp=5;



float konsp=10, konsi=0.00001, konsd=3;


//define inverse
double r;
double b, b2;
double a_tan1,a_tan2,a_cos1,a_cos2;
float theta0,theta1,theta2;
double convert_theta0,convert_theta1,convert_theta2;
//define PID
float temp_error, temp_error2;
float delta_error[2];
float baca_sensor[3];
float error[2], next_error[2];
float errorI=0, errorD=0;
float proporsional[2], integral[2], derivatif[2];
float sum_error[2], derivat_error[2];
float PID[3];
int buff_1,buff_2,buff_3;
char counter_set_awal=0;
char counterawal=0;

void USART_Con(){
RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,ENABLE);
GPIO_PinAFConfig(GPIOA,GPIO_PinSource9,GPIO_AF_USART1);
```

```
}


void TIM_Config(void)
{
TIM_TimeBaseInitTypeDef   TIM_TimeBaseStructure;
//          TIM_OCInitTypeDef
            TIM_OCInitStructure;
NVIC_InitTypeDef
            NVIC_InitStructure;

//untuk timer2
/* TIM2 clock enable */
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
/* Time base configuration */
TIM_TimeBaseStructure.TIM_Period = 1000 - 1; // 1 MHz down to 1
KHz (1 ms)
TIM_TimeBaseStructure.TIM_Prescaler = 60 - 1; // 24 MHz Clock
down to 1 MHz (adjust per your clock)
TIM_TimeBaseStructure.TIM_ClockDivision = 0;
TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
/* TIM IT enable */
TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
/* TIM2 enable counter */
TIM_Cmd(TIM2, ENABLE);

NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);

//untuk timer3
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
/* Time base configuration */
```

```
TIM_TimeBaseStructure.TIM_Period = 1000 - 1; // 1 MHz down to 1
KHz (1 ms)
TIM_TimeBaseStructure.TIM_Prescaler = 60 - 1; // 24 MHz Clock
down to 1 MHz (adjust per your clock)
TIM_TimeBaseStructure.TIM_ClockDivision = 0;
TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
TIM_TimeBaseInit(TIM3, &TIM_TimeBaseStructure);
/* TIM IT enable */
TIM_ITConfig(TIM3, TIM_IT_Update, ENABLE);
/* TIM2 enable counter */
TIM_Cmd(TIM3, ENABLE);

NVIC_InitStructure.NVIC_IRQChannel = TIM3_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);

}
void TIM2_IRQHandler(void){

if (TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET){
TIM_ClearITPendingBit(TIM2, TIM_IT_Update);

TM_ADC_Init(ADC1, ADC_Channel_4); //ADC1 ch 4, PA4
TM_ADC_Init(ADC1, ADC_Channel_5);    //ADC1 ch 5, PA5
TM_ADC_Init(ADC1, ADC_Channel_6); //ADC1 ch 6, PA6
TM_ADC_Init(ADC1, ADC_Channel_7); //ADC1 ch 7, PA7

a++;      if(a>=20)
{         a=0;

baca_sensor[0]=TM_ADC_Read(ADC1, ADC_Channel_5);
baca_sensor[1]=TM_ADC_Read(ADC1, ADC_Channel_7);


sensor[0]= (TM_ADC_Read(ADC1, ADC_Channel_5))  / set_adcA;
          //kiri
```

```
sensor[1]= (TM_ADC_Read(ADC1, ADC_Channel_7))  / set_adcB;
          //kanan


delta_error[0]=set_adc-sensor[0];
if(delta_error[0]<0){
delta_error[0]=delta_error[0]*(-1);
}
//else{delta_error[0]=delta_error[0];}

delta_error[1]=set_adc-sensor[1];
if(delta_error[1]<0){
delta_error[1]=delta_error[1]*(-1);
}
//else{delta_error[1]=delta_error[1];}

}
}
}

void TIM3_IRQHandler(void){
if (TIM_GetITStatus(TIM3, TIM_IT_Update) != RESET){
TIM_ClearITPendingBit(TIM3, TIM_IT_Update);

}
}
void init_usart(void){

GPIO_InitTypeDef GPIO_InitStructure;
USART_InitTypeDef USART_InitStructure;

/* enable peripheral clock for USART2 */
RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);

/* GPIOA clock enable */
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);

/* GPIOA Configuration:  USART2 TX on PA2 */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
```

```
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP ;
GPIO_Init(GPIOA, &GPIO_InitStructure);

/* Connect USART2 pins to AF2 */
// TX = PA2
GPIO_PinAFConfig(GPIOA, GPIO_PinSource2, GPIO_AF_USART2);
//GPIO diubah menjadi usart
USART_InitStructure.USART_BaudRate = 115200;
USART_InitStructure.USART_WordLength                    =
USART_WordLength_8b; //panjang kata 8 bit
USART_InitStructure.USART_StopBits = USART_StopBits_1;
USART_InitStructure.USART_Parity = USART_Parity_No;
USART_InitStructure.USART_HardwareFlowControl           =
USART_HardwareFlowControl_None;
USART_InitStructure.USART_Mode = USART_Mode_Tx;
USART_Init(USART2, &USART_InitStructure);

USART_Cmd(USART2, ENABLE); // enable USART2

// untuk USART 1 ==============================
RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);

RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6; // Pins 6 (TX) and 7
(RX) are used
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
        // the pins are configured as alternate function so the USART
peripheral has access to them
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        // this defines the IO speed and has nothing to do with the
baudrate!
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
        // this defines the output type as push pull mode (as opposed to
open drain)
```

```
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
        // this activates the pullup resistors on the IO pins
GPIO_Init(GPIOB, &GPIO_InitStructure);
        // now all the values are passed to the GPIO_Init() function
which sets the GPIO registers
GPIO_PinAFConfig(GPIOB, GPIO_PinSource6, GPIO_AF_USART1);
//

USART_InitStructure.USART_BaudRate = 9600;
        // the baudrate is set to the value we passed into this init
function
USART_InitStructure.USART_WordLength                    =
USART_WordLength_8b;// we want the data frame size to be 8 bits
(standard)
USART_InitStructure.USART_StopBits = USART_StopBits_1;
        // we want 1 stop bit (standard)
USART_InitStructure.USART_Parity = USART_Parity_No;
        // we don't want a parity bit (standard)
USART_InitStructure.USART_HardwareFlowControl           =
USART_HardwareFlowControl_None; // we don't want flow control
(standard)
USART_InitStructure.USART_Mode = USART_Mode_Tx; // we want
to enable the transmitter and the receiver
USART_Init(USART1, &USART_InitStructure);
                // again all the properties are passed to the
USART_Init function which takes care of all the bit setting
USART_Cmd(USART1, ENABLE); // enable USART2

// untuk USART 3 ==============================

USART_InitStructure.USART_BaudRate = 115200;
USART_InitStructure.USART_WordLength                    =
USART_WordLength_8b;
USART_InitStructure.USART_StopBits = USART_StopBits_1;
USART_InitStructure.USART_Parity = USART_Parity_No;
USART_InitStructure.USART_HardwareFlowControl           =
USART_HardwareFlowControl_None;
USART_InitStructure.USART_Mode      =      USART_Mode_Rx      |
USART_Mode_Tx;
```

```
RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART3, ENABLE);

RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOD, ENABLE);

GPIO_PinAFConfig(GPIOD,GPIO_PinSource8,GPIO_AF_USART3);
GPIO_PinAFConfig(GPIOD,GPIO_PinSource9,GPIO_AF_USART3);

GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOD, &GPIO_InitStructure);

GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
GPIO_Init(GPIOD, &GPIO_InitStructure);

USART_Init(USART3, &USART_InitStructure);

// Enable USART
USART_Cmd(USART3, ENABLE);
}
void USART_puts(USART_TypeDef* USARTx, volatile char *s){

while(*s){
// wait until data register is empty
while( !(USARTx->SR & 0x00000040) );
USART_SendData(USARTx,*s);
*s++;
}
}

int main(void) {
init_usart();
//tm_init(A, BV(2), OD, OUT, NOPULL, 100);
SystemInit();                           // Init system
```

```
TM_DELAY_Init();                    // Init delay
TIM_Config();

reset_posisi(); Delayms(100);


while (1)
{

sprintf(kirimPID,"$A%fB%fC%fD%fE%fF%fG%f#",
temp[0],temp[1],temp[2],x[1],y[1],z[1],error[0]);
USART_puts(USART1,kirimPID);
Delayms(100);

switch(flag){

case 0:
reset_posisi();
PID[0]=0;
counter_set_awal ++ ;
if(counter_set_awal > 50)
{
flag = 1 ;
}
case 1:
counter_set_awal = 0;
if((baca_sensor[0]<2500) && (baca_sensor[1]<3000)){


for(i=0;i<=13;i++){
geser(i,0,0); Delayms(50);

}

for(i=13;i>=-13;i--){
geser(i,0,0);       Delayms(50);

}
```

```
}

else if((baca_sensor[0]>=2500) || (baca_sensor[1]>=3000)){
flag=2;}
break;

case 2:
if(sensor[0]<sensor[1]){
temp_error=set_adc-sensor[1];
if (temp_error<0){
temp_error=(-1)*temp_error;
}

error[0]=(-1)*temp_error;
}
else if(sensor[0]>sensor[1]){
temp_error2=set_adc-sensor[0];
if (temp_error2<0){
temp_error2=(-1)*temp_error2;
}

error[0]=temp_error2;
}


proporsional[0]=error[0]*konsp;


errorI=errorI+error[0];
integral[0]=(errorI)*konsi;
errorI=error[0];


derivatif[0]=(error[0]-errorD)*konsd;
errorD=error[0];


PID[0]=proporsional[0]+derivatif[0]+integral[0];
if(PID[0]>13){PID[0]=13;}
```

```
if(PID[0]<-13){PID[0]=-13;}

//kirim perintah gerak
geser(PID[0],0,0); Delayms(100);

PID_2=PID[0];


if(delta_error[0]<0.01 || delta_error[1]<0.01){
PID[0]=PID[0];
geser(PID[0],0,0);

}

if((baca_sensor[0]<2500) && (baca_sensor[1]<3000)){
flag=0;
}

break;
}

} //akhir while 1

}

void inverse_kinematic(int num){
float temp_theta0[1];

theta2 =
acos((z[num]*z[num]+(sqrt(x[num]*x[num]+y[num]*y[num]))*(sqrt(x[
num]*x[num]+y[num]*y[num]))-l1*l1-l2*l2)/(2*l1*l2));
theta1 =
atan(z[num]/(sqrt(x[num]*x[num]+y[num]*y[num])))+atan(l2*cos((90)/
180*3.14285714)/(l1+l2*sin((90)/180*3.14285714)));
theta0 = atan(y[num]/x[num]);

theta2 = theta2*rad_pulse;
theta1 = (theta1*rad_pulse);
theta0 = (theta0*rad_pulse);
```

```
if (theta0<0){
temp_theta0[0]=180-(-1*theta0);
theta0=temp_theta0[0];
}
else {
theta0=theta0;
}

temp[0]=theta0;
temp[1]=theta1;
temp[2]=theta2;

buff_1=(int)theta0*10;
buff_2=(int)theta1*10;
buff_3=(int)theta2*10;


tangan_0=temp[0]*(19.4);
tangan_1=temp[1]*(19);
tangan_2=temp[2]*(17.5);

test_3();
}

//gerak kanan kiri atas bawah
void geser(float _x,float _y,float _z){
x[1] = koordinat[0] + _x;
y[1] = koordinat[1] + _y;
z[1] = koordinat[2] + _z;

inverse_kinematic(1);
}
osisi awal
void homing(int _x,int _y,int _z){
x[0] = koordinat[0]+ _x ;
y[0] = koordinat[1]+ _y ;
z[0] = koordinat[2]+ _z ;
inverse_kinematic(0);
```

```
}

//koordinat acuan awal
void reset_posisi(){
//10.1.0
koordinat[0] = 0.1;                    //x
koordinat[1] = 13;                     //y
koordinat[2] = 12;          //z
homing(0,0,0);
}

//kirim serial ke servo controller
void test_3(){
sprintf(kirimservo,"#0P%f      #1P%f      #2P%f      T1000      %c
\n\r",tangan_0,tangan_1,tangan_2,13 );
USART_puts(USART2,kirimservo);
//while(USART_GetFlagStatus(USART2,USART_FLAG_TXE)==RES
ET);
Delayms(100);
}
```

Program Delphi

```
unit Fuzzy;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, Buttons, TeEngine, Series, ExtCtrls, TeeProcs,
Chart,
  ComCtrls, CPort;

type
  TForm1 = class(TForm)
    BitBtn2: TBitBtn;
    Button2: TButton;
    GroupBox1: TGroupBox;
    Edit3: TEdit;
    Edit4: TEdit;
    SpeedButton1: TSpeedButton;
    Button1: TButton;
    Timer1: TTimer;
    Chart1: TChart;
    SaveDialog1: TSaveDialog;
    Series1: TLineSeries;
    Series2: TLineSeries;
    ComPort1: TComPort;
    Memo1: TMemo;
    Edit1: TEdit;
    Chart2: TChart;
    Series3: TLineSeries;
    Series4: TLineSeries;
    Series5: TLineSeries;
    Series6: TLineSeries;
    StaticText1: TStaticText;
    StaticText2: TStaticText;
    Button3: TButton;
    StaticText3: TStaticText;
    StaticText4: TStaticText;
```

```
    StaticText5: TStaticText;
    Memo2: TMemo;
    Memo3: TMemo;
    Memo4: TMemo;
    Memo5: TMemo;
    Memo6: TMemo;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    procedure Button2Click(Sender: TObject);
    procedure ComPort1RxChar(Sender: TObject; Count: Integer);
    procedure SpeedButton1Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    //procedure Timer1Timer(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  dataacak,data,str,str1,ekiri,ekanan  : string;
  lokasiheader,lokasi,n: longint;
  status : boolean;
  databaru,
sensorA,sensorB,sensorC,sensorD,sensorE,sensorF,sensorG,cekTail      :
string;
  intsensorA,intsensorB,intsensorC : double;
  namafile,teta0,teta1,teta2 :Textfile;
  k:integer;
implementation

uses Math;
```

```
{$R *.dfm}

procedure TForm1.Button2Click(Sender: TObject);
begin
ComPort1.Showsetupdialog;
end;

procedure TForm1.ComPort1RxChar(Sender: TObject; Count: Integer);
var
pjg:integer;
begin
  comport1.ReadStr(dataacak,1);

  //listbox1.Items.Add(dataacak);

  if (dataAcak='$')then
  begin
  status:=true;
  end;

 if  (status=true) then
 begin

 data:=data+dataAcak;
 if dataacak='#' then
 begin

  lokasiheader :=pos('$',data);
  lokasi := pos('#',data);
  if  (copy(data,lokasiheader,1)= '$') AND (copy(data,lokasi,1)='#') then
  begin

  databaru := copy(data,lokasiheader+1,lokasi-1);
  //teta
  sensorA :=copy(databaru,pos('A',databaru)+1,4);
  sensorB :=copy(databaru,pos('B',databaru)+1,4);
  sensorC :=copy(databaru,pos('C',databaru)+1,4);

  //posisi
```

```
  sensorD :=copy(databaru,pos('D',databaru)+1,4);
  sensorE :=copy(databaru,pos('E',databaru)+1,4);
  sensorF :=copy(databaru,pos('F',databaru)+1,4);

  sensorG :=copy(databaru,pos('G',databaru)+1,4);


  Memo1.Lines.Add(databaru);
  Memo2.Lines.Add(sensorA);
  Memo3.Lines.Add(sensorB);
  Memo4.Lines.Add(sensorC);
  Memo5.Lines.Add(sensorD+','+sensorE+','+sensorF);
  Memo6.Lines.Add(sensorG);

  Edit3.Text:=sensorA; //sensorA;
  Edit4.Text:=sensorB;
  Edit1.Text:=sensorC;

  intsensorA:=StrToFloat(sensorA);
  intsensorB:=StrToFloat(sensorB);
  intsensorC:=StrToFloat(sensorC);

  Series1.AddXY(k,intsensorA);
  Series2.AddXY(k,intsensorB);
  Series3.AddXY(k,intsensorC);

append(teta0);
writeln(teta0,sensorA);
append(teta1);
writeln(teta1,sensorB);
append(teta2);
writeln(teta2,sensorC);
k:=k+1;
end;

   data:='0';
  status:=false;

  end;
```

```
  end;
end;


procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
k:=0;
Comport1.Open;
//timer1.interval := 2;
timer1.enabled:= true;
status :=false;

assignfile(namafile,'data.txt');
assignfile(teta0,'teta0.txt');
assignfile(teta1,'teta1.txt');
assignfile(teta2,'teta2.txt');

end;

procedure TForm1.Button1Click(Sender: TObject);
begin
comport1.Close;
timer1.Enabled:=false;

end;


procedure TForm1.Button3Click(Sender: TObject);

begin
Series1.Clear;
Series2.Clear;
Series3.Clear;

 end;



end.
```

DAFTAR PUSTAKA

[1] Kose, Ozan. 2015. *Kebocoran Gas Tewaskan Dua Pekerja Tambang*. MetrotvnewsBlog: Metrotv.

[2] Adiputri, Novi Christiastuti. 2014. *4 Pekerja Tewas Akibat Kebocoran Gas Kimia di Pabrik AS*. DetikBlog: Detik.

[3] Panjaitan Berkat, Siringo-Ringo Berliana, dkk. 2013.*Sensor Gas*. Medan: Technology Science, Sains, fisika.

[4] Figaro.2014.*Datasheet2600,*http://www.figarosensor.com/products/2600.pdf (diakses 23 Februari 2015)

[5] Widianto, Didik. 2011. *Antarmuka Serial Standar*. Semarang: Universitas Diponogoro.

[6] Syahrul.2006. *Karakteristik dan Pengontrolan servomotor*. Bandung: Jurusan Teknik Komputer, Universitas Komputer Indonesia.

[7] Robotika. "Lengan Robot (Robotic Arm) http://robotika.web.id/robot-lengan-arm-robot/ (diakses 4 Januari 2015)

[8] Chriss Bannet. 2007. *Gear – Using Garmin Forerunner 305 for an Ironman. https://triduffer.wordpress.com/2007/08/19/using-garmin-forerunner-305-for-an-ironman/* (Diakses 29 Mei 2015)

[9] ST-Electronics.2015."STM32F4Discovery", http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/PF252252 (diakses pada 1 Juni 2015)

[10] Lynmotion. 2005. SSC Manual. SSC32 : Pekin

[11] Wikipedia.2015.PID Controller, https://en.wikipedia.org/wiki/PID_controller (diakses 29 Mei 2015)

[12] Permata Sari, Ranti.2005. Penalaan Paramater Kontrol PID dengan Metode Heuristic, Aplikasi : Sistem Pengendalian Kecepatan Motor DC. Surabaya : Jurusan Teknik Fisika, Institut Teknologi Sepuluh Nopember

[13] Ogata, Katsuhiko.2010. Modern Control Engineering. Boston : Prentice Hall

BIODATA PENULIS

Nafi' Abdul Hakim di lahirkan di Surabaya, pada tanggal 22 Januari 1993 dari pasangan Bapak Amar Ma'ruf dan Ibu Soelijati. Penulis adalah anak kedua dari 3 bersaudara. Saudara yang pertama adalah Rahman Hakim yang sedang menjadi dosen di Politeknik Negeri Batam dan Adik Hakma Kurnia Lestari yang masih menempuh SMA di SMAN 11 Surabaya. Mottonya *work hard, play hard*. Motto tersebut memotivasi penulis dalam menjalani hidup.

Perjalanan hidup penulis dalam mengenyam pendidikan dimulai sejak umur 5 tahun masuk TK Kuncup Harapan yang kemudian dilanjutkan di SDN Benowo 1 Surabaya. Setelah lulus SD melanjutkan jenjang SMP Negeri 26 Surabaya. Penulis melanjutkan Ke SMA Negeri 11 Surabaya, dan kemudian melanjutkan kuliah di Jurusan Teknik Elektro, Institut Teknologi Sepuluh Nopember. Judul tugas akhir penulis adalah "*Implementasi Kontrol PID pada Lengan Robot untuk Mencari Sumber Gas Menggunakan STM32F4"*.

Email: nafi.abdulhakim@gmail.com