JITEL (Jurnal Ilmiah Telekomunikasi, Elektronika, dan Listrik Tenaga) p-ISSN: 2774-7972
Vol. 1, No. 1, Maret 2021, pp. 69-78 e-ISSN: 2775-6696
Website: https://jitel.polban.ac.id/jitel

Sistem kendali posisi motor DC menggunakan *state feedback controller* dan *real-time operating system*

Martin*
Jurusan Teknik Elektro, Politeknik Negeri Bandung
Jl. Gegerkalong Hilir, Ciwaruga, Kec. Parongpong, Kabupaten Bandung Barat, Indonesia
*martin@polban.ac.id

ABSTRAK

Motor DC merupakan sistem penggerak yang paling banyak digunakan di bidang industri, otomasi, robotika, ataupun lainnya. Penggunaan sistem kendali banyak diterapkan untuk pengaturan pergerakan kecepatan ataupun posisi dari motor DC. Pada penelitian ini, *state feedback controller* dan penambahan kendali integral dengan estimator digunakan untuk mengendalikan posisi motor DC. Sistem dibuat berbasis *real-time operarting system* (RTOS) untuk pembacaan sensor, perhitungan matematis kendali, dan pengiriman sinyal *pulse width modulation* (PWM). Pengendalian dilakukan pada motor DC Quanser yang terhubung dengan Arduino Mega 2560 untuk membaca sensor *encoder* dan menampilkan data pengujian. Hasil pengujian menunjukkan bahwa *state feedback controller* dapat mengendalikan posisi motor DC dengan nilai penguat K sebesar 2,66 dan 115,37, nilai penguat N_bar sebesar 0,49 dan nilai estimator sebesar 7,75 dan 0,26. Penggunaan RTOS sebagai inti pemrograman dapat menyelesaikan permasalahan dalam pengerjaan *task-task* seperti pembacaan sensor, perhitungan parameter kendali, dan pengiriman sinyal kendali tanpa terjadi *error* selama pengujian sistem. Hasil analisa menunjukan keluaran sistem kendali posisi memiliki nilai *overshoot* sebesar 2,63% pada pengujian pertama dan 2,66% pada pengujian kedua.

Kata kunci: *state feedback controller*, *real-time operating system*, motor DC Quanser

ABSTRACT

DC motors are the most widely used drive systems in industry, automation, robotics, or others. The use of a control system is widely applied to adjust the speed or position movement of the DC motor. In this study, the state feedback controller and the addition of an integral control with an estimator are used to control the position of the DC motor. The system is made based on a real-time operating system (RTOS) for sensor readings, control mathematical calculations, and pulse width modulation (PWM) signal transmission. Control is carried out on a Quanser DC motor connected to the Arduino Mega 2560 to read the encoder sensor and display the test data. The test results show that the state feedback controller can control the position of the DC motor with amplifier K values of 2.66 and 115.37, N_bar amplifier values of 0.49 and estimator values of 7.75 and 0.26. The use of RTOS as the core of programming can solve problems in carrying out tasks such as reading sensors, calculating control parameters, and sending control signals without errors during system testing. The results of the analysis show that the position control system output has an overshoot value of 2.63% in the first test and 2.66% in the second test.

Keywords: *state feedback controller*, *real-time operating system*, *Quanser DC motor*

1. PENDAHULUAN

Motor DC merupakan salah satu komponen penting yang banyak digunakan dalam industri, otomotif, ataupun kehidupan sehari-hari. Penggunaan komponen tersebut seperti pada mobil *remote control*, penggerak *belt conveyor*, ataupun dapat digunakan sebagai generator. Motor DC merupakan salah satu penggerak elektrik yang sangat sering digunakan dan juga dikembangkan karena motor DC mudah untuk dikendalikan menggunakan kendali linier [1]. Salah satu pengembangan yang paling banyak dilakukan yaitu dengan mengatur atau mengendalikan dari posisi pergerakan motor DC. Posisi dari pergerakan motor DC dapat diketahui dan dikendalikan dengan meletakan sensor untuk mendeteksi perubahan yang telah terjadi. Sensor *encoder* merupakan salah satu komponen untuk membaca pergerakan kecepatan ataupun posisi putaran motor DC. Sensor tersebut menghasilkan

Naskah diterima tanggal 27 Januari 2021, disetujui tanggal 5 Maret 2021
*E-mail korespondensi

keluaran berupa tegangan DC yang akan dijadikan umpan balik dari alat yang akan dikendalikan. Penerapan kendali posisi pada motor DC merupakan sebuah tantangan tersendiri dengan banyaknya jenis kendali yang telah ditemukan, seperti kendali proporsional integral derivatif (PID), logika *fuzzy*, *pole placement*, *linear quadratic regulator* (LQR), dan *linear quadratic gaussian* (LQG) [2], [3].

Penelitian tentang pengendalian motor DC berserta penerapannya telah banyak dilakukan. Peneliti [4] melakukan penelitian tentang pemodelan dan kendali posisi dari sebuah motor DC. Peneliti tersebut melakukan perbandingan antara kendali PID dan LQR dalam mengendalikan posisi motor DC yang telah dimodelkan. Pada penelitian [5] menampilkan pengendalian kecepatan motor *brushless* DC (BLDC) menggunakan *hall-effect position sensor* sebagai umpan balik untuk mengetahui kecepatan motor BLDC. Hasil dari penelitian tersebut menampilkan bahwa sensor posisi dapat digunakan sebagai pendeteksi kecepatan motor BLDC. Logika *fuzzy* juga digunakan pada penelitian [6]-[8] untuk mengendalikan posisi sebuah motor DC, walaupun terdapat perbedaan logika *fuzzy* yang digunakan pada ketiga judul penelitian tersebut. Penelitian [6] menggunakan *fast fuzzy controller* untuk mengendalikan posisi motor DC yang diimplementasikan pada *field-programmable gate array* (FPGA). Kemudian penelitian [7] menggunakan *adaptive fuzzy controller* untuk mengendalikan motor DC dengan melakukan pengoptimalan dalam pengendalian berbagai jenis beban pada sistem. Selanjutnya peneliti [8] melakukan penelitian untuk mengendalikan posisi motor DC menggunakan logika *fuzzy* digabungkan dengan kendali proporsional integral (PI). Logika *fuzzy* digunakan untuk menentukan parameter PI terbaik yang akan diimplementasikan pada pengedalian posisi motor DC. Adapun penerapan pada kendali posisi motor DC salah satunya untuk *solar tracking system*. Pengendali *sliding mode* digunakan untuk mengendalikan posisi motor DC sehingga dapat mengikuti arah cahaya matahari agar dapat digunakan maksimal pada panel surya [9]. Pengendali LQG yang digabungkan dengan filter Kalman juga dapat digunakan sebagai kendali posisi motor DC. Pengoptimalan nilai penguatan umpan balik dilakukan oleh filter Kalman seperti pada penelitian [10]. Pengendali lainnya yang dapat digunakan yaitu *pole placement* atau yang dapat disebut *state feedback controller*. Penguatan diberikan pada umpan balik sistem dan masukan sistem dimana penguatan tersebut didapatkan berdasarkan perhitungan matematis yang didapatkan setelah mengetahui model dari sistem yang akan dikendalikan. Terakhir, penelitian [11] dan [12] merancang kendali *state feedback* dengan menerapkan pengendalian pada jenis motor DC yang berbeda.

Dalam penelitian ini akan dilakukan pengendalian posisi motor DC menggunakan *state feedcback controller* dan diimplementasikan pada mikrokontroler yang telah diberikan *real time operating system* (RTOS). Motor DC dikopling dengan *high resolution optical encoder* dimana telah diketahui modelnya. Penelitian-penelitian yang dijadikan sebagai referensi hanya menggunakan metode *loop* standar ataupun *super-loop* dalam menjalankan *task* program. Pada penelitian ini, penggunaan RTOS bertujuan untuk menjalankan perintah-perintah dalam melaksanakan *task* baik dalam pembacaan sensor, pengendalian motor, ataupun menampilkan grafik sehingga dapat memaksimalkan pelaksanaan *task*. Pelaksanaan *task* dilaksanakan lebih cepat dengan *delay* yang sangat rendah. Penggunakan RTOS pada mikrokontroler akan sangat berbeda apabila dibandingkan hanya dengan menggunakan *loop* ataupun *super-loop* program.

2. METODE PENELITIAN

2.1 *State Feedback Controller*

Pole placement merupakan dasar dalam metode *state feedback controller* yaitu dengan mengasumsikan bahwa semua variabel dapat terukur dan bisa dijadikan umpan balik sistem yang digunakan. Namun, pada pengendalian ini semua sistem harus dapat teramati dan terkendali. Metode ini digunakan dengan menentukan nilai kutub-kutub berdasarkan respon sistem bentuk transien ataupun frekuensi, seperti kecepatan, damping rasio, atau *bandwidth* [2].

Untuk mendapatkan nilai kutub yang digunakan, diasumsikan terdapat satu masukan dan satu keluaran pada sistem. Kemudian sistem terkendali dimana semua variabel terhubung ke bagian masukan. Setelah sistem dalam keadaan *close loop*, maka dapat dicari nilai kutub yang akan digunakan.

Asumsikan sistem dinamik linier sebagai berikut

$$\dot{x} = Ax + Bu \tag{1}$$

$$y = Cx + Du \tag{2}$$

dengan

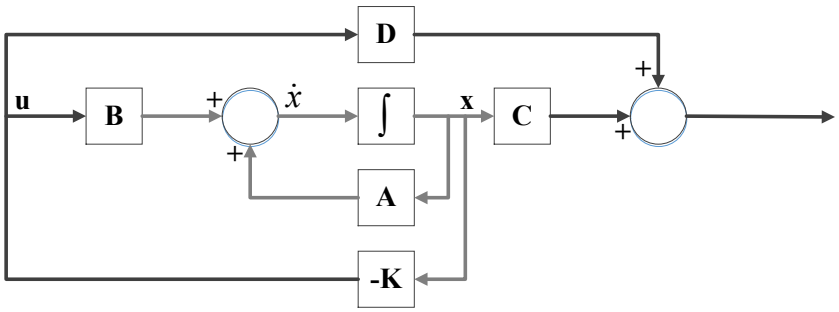
$$u = -Kx \tag{3}$$

dimana x adalah vektor *state* (*n-vector*), y adalah sinyal keluaran (skalar), u adalah sinyal kontrol (skalar), A, B, C, D berturut-turut adalah matriks konstan berdimensi n x n, dan K adalah matriks konstan berdimensi 1 x n.

Setalah diketahui persamaan (1) sampai (3) ditentukan bahwa masukan sistem adalah *state* yang kontinyu yang didapatkan dari sinyal umpan balik. Nilai K merupakan matriks penguatan pada sistem yang berdimensi 1 x n. Bila persamaan (3) disubstitusikan ke persamaan ke (1), maka akan didapatkan persamaan baru seperti dibawah ini.

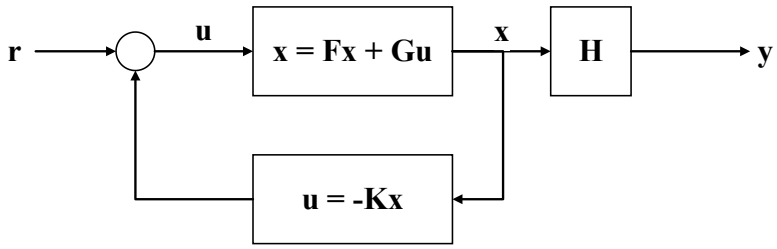
$$\dot{x} = (A - BK)x \tag{4}$$

Sesuai dengan persamaan (4), maka kestabilan dan transien respon ditentukan oleh nilai *eigen* dari matriks $A - BK$. Nilai K didapatkan dari karakteristik sistem yang diinginkan sehingga didapatkan matriks yang stabil dan nilai *eigen* dari $A - BK$ merupakan kutub regulator dari sistem. Penempatan kutub disesuaikan dengan variabel yang akan dijadikan umpan balik. Diagram blok dari persamaan (4) akan didapatkan seperti Gambar 1.



Gambar 1. Diagram blok *pole placement*

Persamaan (4) yang telah dijelaskan merupakan *state feedback controller* dengan status tidak penuh. *State feedback controller* berstatus penuh dapat dibuat dengan menambahkan kendali integral pada *state feedback controller* dengan status tidak penuh. Diagram blok dari *state feedback controller* pada sebuah sistem *close loop* dengan masukan acuan (r) dapat dilihat pada Gambar 2.



Gambar 2. Diagram blok dengan masukan acuan

Sistem akan mempunyai kesalahan dalam keadaan setimbang tak nol terhadap masukan langkah. Untuk mengoreksinya, pada keadaan setimbang dihitung nilai status dan masukan kendali yang menghasilkan kesalahan keluaran nol, kemudian menekannya untuk bertahan pada nilai ini. Bila nilai akhir dari status ini dan masukan kendali adalah x_{ss} dan u_{ss}, maka formula kendali yang baru sebagai berikut [3].

$$u = u_{ss} - K(x - x_{ss}) \quad (5)$$

Bila nilai x sama dengan nilai x_{ss}, maka nilai u sama dengan nilai u_{ss}. Nilai kesalahan setimbang harus bernilai nol terhadap sembarang masukan, agar nilai akhir atau keluaran akan sama dengan nilai masukan acuan dari sistem. Persamaan (1) dan (2) jika matrik A, B, C dan D diubah menjadi F, G, H dan J dalam keadaan setimbang akan berubah menjadi persamaan dibawah ini.

$$0 = Fx_{ss} + Gu_{ss} \quad (6)$$

$$y = Hx_{ss} + Ju_{ss} \quad (7)$$

Dengan demikian, nilai keluaran sama dengan nilai masukan acuan ($y_{ss} = r_{ss}$) dengan nilai r_{ss} sembarang dan dinyatakan dalam

$$x_{ss} = N_x r_{ss} \, , u_{ss} = N_u r_{ss} \quad (8)$$

Persamaan kendali dengan menggabungkan persamaan (7) sampai dengan (8) yang telah didapatkan *state space* yaitu seperti berikut.

$$\begin{bmatrix} F & G \\ H & J \end{bmatrix} \begin{bmatrix} N_x \\ N_u \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (9)$$

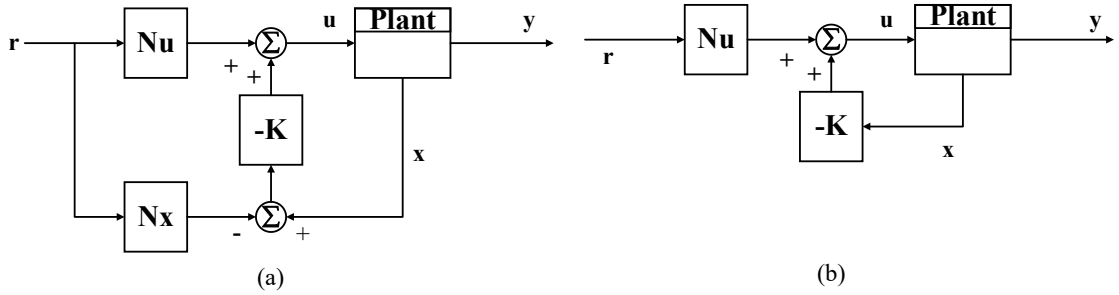
Untuk memperoleh kesalahan setimbang sistem nol, maka masukan acuan harus memenuhi nilai seperti persamaan (10), sedangkan nilai N_u dan N_x dapat dijadikan komposit tunggal sehingga didapatkan persamaan (11) dibawah ini.

$$u = -Kx + (N_u + KN_x)r \quad (10)$$

$$u = -Kx + \overline{N}r \quad (11)$$

Dengan demikian, akan dihasilkan diagram blok kendali lingkar tertutup seperti pada Gambar 3 berikut

.



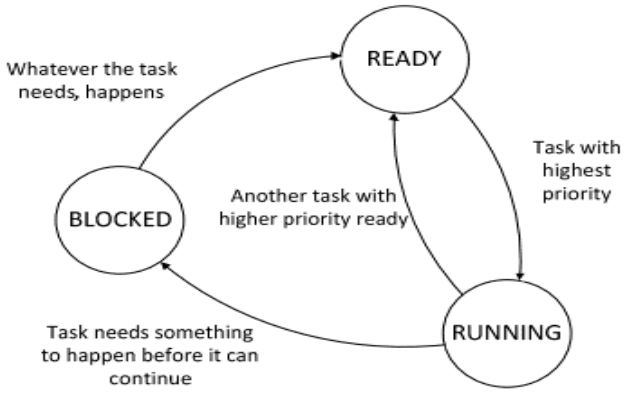
Gambar 3. Diagram blok *full state feedback controller*: (a) persamaan (10), (b) persamaan (11)

2.2 *Real Time Operating System*

Sebagian besar pengendalian sistem dinamik akan memiliki masalah pada waktu pengerjaan program. Dalam pengendalian, berbagai perintah harus diterima pada waktu yang tepat untuk memastikan pengoperasian berjalan dengan lancar dan memenuhi *real-time* [13]. *Real-time operating system* (RTOS) adalah suatu sistem operasi *multitasking* yang diperuntukan untuk aplikasi *real-time*. RTOS dapat menjadikan sistem dapat bekerja secara *real-time*, akan tetapi hasil akhir yang diperoleh tidak dapat dipastikan akan bekerja secara *real-time*. Hal ini dipengaruhi oleh beberapa faktor dan pengembangan yang dilakukan oleh pengembang dengan baik. Karakter dasar dari RTOS yaitu sistem yang memiliki beberapa konsekuensi yang dapat mempengaruhi kinerja sistem apabila batas waktu dari pelaksanaan *task* tidak terpenuhi. Berdasarkan karakteristik itu sendiri, RTOS terbagi menjadi dua bagian yaitu *soft* dan *hard* RTOS. *Soft* RTOS merupakan sistem yang dalam menjalani sistem memiliki tingkat persentase penyelesaian waktu secara *real-time* sebesar kurang dari 100%, sedangkan *hard* RTOS merupakan sistem yang dapat menyelesaikan *task* dalam waktu yang telah ditentukan

dengan tingkat keberhasilan 100%. Penjelasan RTOS akan dibagi menjadi tiga poin utama, antara lain *task*, kernel, dan *mutual exclusion* atau pengerjaan diwaktu yang sama.

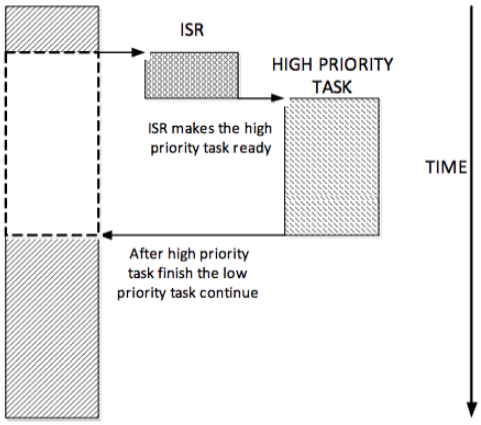
Task merupakan sebuah objek/program yang dieksekusi dan dijalankan layaknya mempunyai CPU tersendiri. Pembagian pekerjaan dalam aplikasi menjadi beberapa *task* merupakan salah satu proses perancangan dari sebuah RTOS. Setiap *task* merupakan lingkaran yang akan terus berulang selama proses menjalankan aplikasi. Selama proses tersebut, *task* akan mengalami tiga buah keadaan seperti yang ditunjukan oleh Gambar 4.



Gambar 4. Siklus *state* pada sebuah RTOS [13]

Kernel merupakan salah satu bagian dari sistem *multitasking* yang mempunyai fungsi sebagai manajemen dari seluruh *task*, mengatur komunikasi tiap *task*, dan yang terpenting adalah mengatur pewaktuan untuk *central processing unit* (CPU) sehingga tidak terjadi *crash* pada CPU. Untuk kernel sendiri terdiri dari dua jenis yaitu, *non-preemptive* dan *preemptive*. Dalam sistem yang digunakan akan menggunakan RTOS yang bersifat *preemptive*.

Preemptive kernel banyak digunakan untuk membuat aplikasi dengan RTOS. Hal ini karena *preemptive* kernel mempunyai respon yang lebih bagus daripada *non-preemptive* kernel. Jadi dapat disimpulkan bahwa *preemptive* kernel selalu mendahulukan *task* dengan prioritas tertinggi yang siap untuk dieksekusi. Dengan *preemptive* kernel, respon sistem bisa mencapai optimal dan waktu untuk menjalankan *task* dengan prioritas tertinggi bisa ditentukan, berbeda dengan *non-preemptive* kernel yang tidak bisa ditentukan. Gambar 5 menampilkan prinsip kerja *preemptive* kernel.

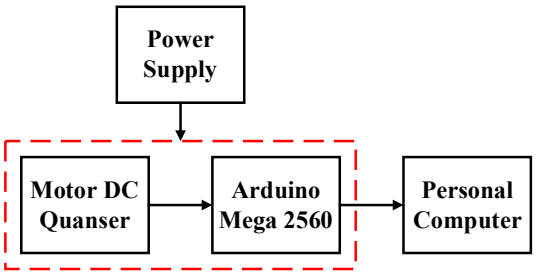


Gambar 5. Prinsip kerja *preemptive* kernel [13]

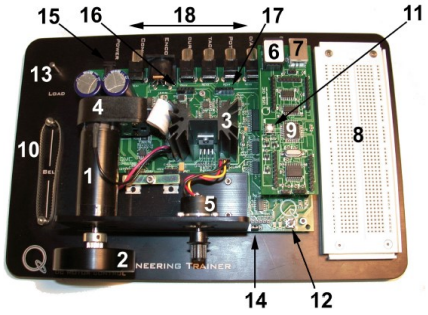
Mutual exclusion merupakan kondisi setiap sumber daya diberikan tepat pada satu waktu. Ada beberapa metode untuk mendukung *mutual exclusion*, antara lain adalah *semaphore* dan *message queue*. *Message queue* digunakan untuk mengirim satu atau lebih pesan kepada *message queue* digunakan untuk mencegah terjadinya *error* karena *shared data*.

2.3 Perancangan Perangkat Keras

Perancangan perangkat keras menggunakan dilakukan menggunakan modul motor DC Quanser yang dihubungkan ke Arduino Mega 2560 untuk membaca nilai sensor *encoder* sebagai umpan balik sistem. Diagram blok perangkat keras dan perangkat keras yang digunakan dapat dilihat pada Gambar 6 dan Gambar 7.



Gambar 6. Diagram blok perangkat keras



Gambar 7. Motor DC Quanser

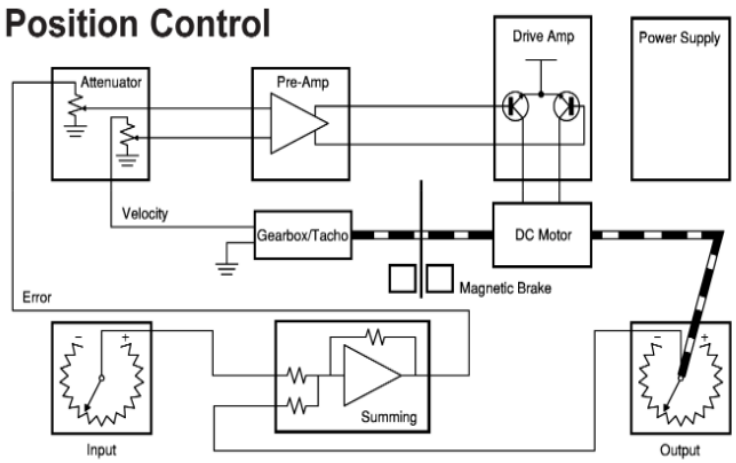
Modul motor DC Quanser seperti pada Gambar 7 memiliki beberapa komponen yang akan digunakan dan ada juga yang tidak akan digunakan dengan detail pada Tabel 1 berikut.

Tabel 1. Deskripsi komponen motor DC Quanser pada Gambar 7

No.	Deskripsi
1	*Maxon DC motor*
2	*Removable inertial load*
3	*Linear power amplifier*
4	*High resolution optical encoder*
5	*Ball bearing servo potentiometer*
6	*RJ11 port on QIC: for downloading firmware using a compatible programming device*
7	*USB port on QIC: for online tuning and plotting using LABview*
8	*Breadboard option: to implement controllers with your own circuits*
9	*Embedded/portable option*
10	*Removable belt: to drive potentiometer*
11	*PIC reset switch*
12	*User switch*
13	*Inertial load storage pin*
14	*Jumper J6: switch between DAQ and QIC*
15	*6-mm power jack*
16	*Power supply header: J4*
17	*Analog signal header: J11*
18	*i. PC interface option* *ii. Analog controller option*

Block circuit motor DC Quanser yang akan digunakan dapat dilihat pada Gambar 8. Adapun model fungsi transfer pada motor DC Quanser telah diketahui dari *datasheet* dengan persamaan sebagai berikut.

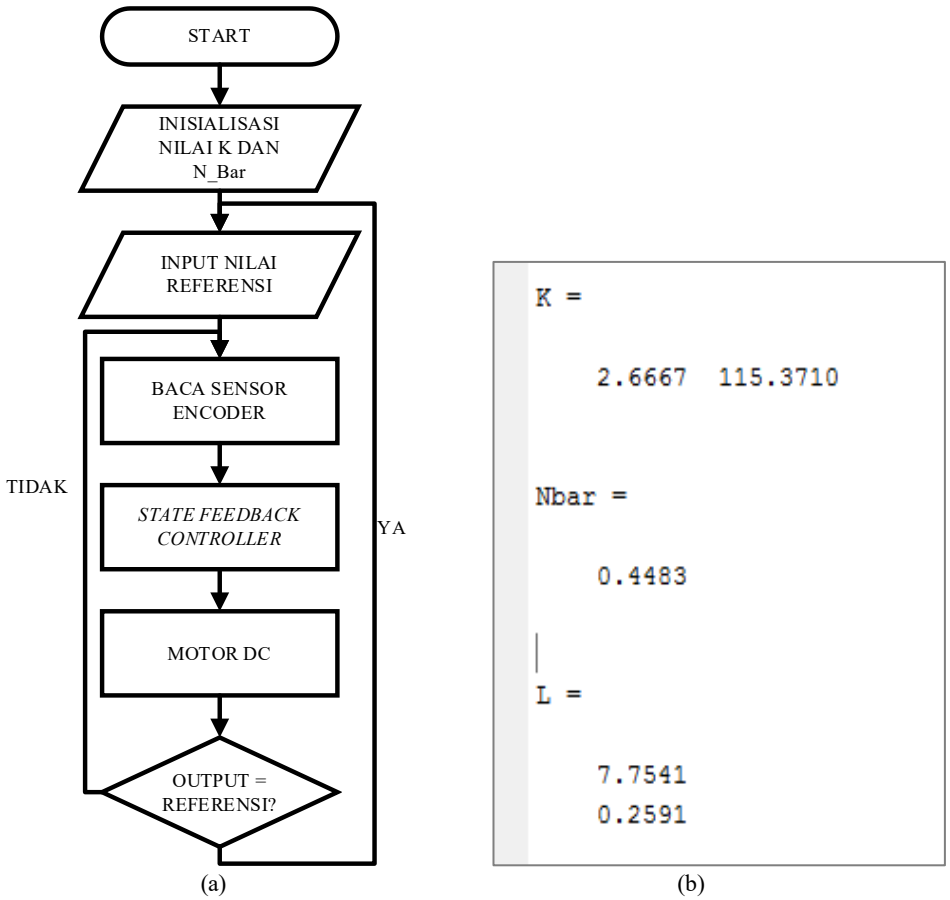
$$Tf_m = \frac{19{,}3}{0{,}07s^2+s} \tag{12}$$



Gambar 8. *Block circuit* motor DC

2.4 Perancangan Perangkat Lunak

Perancangan perangkat lunak pada implementasi ini sesuai dengan tujuan yaitu bagaimana menghubungkan RTOS dengan kendali *state feedback controller* dalam sistem *embedded* untuk mengontrol posisi motor. Perangkat lunak MATLAB digunakan untuk mengetahui nilai penguatan K, estimator L, dan $\overline{N}$. *Flowchart* perangkat lunak kendali *state feedback controller*, serta nilai K, L, dan $\overline{N}$ dapat dilihat pada Gambar 9.



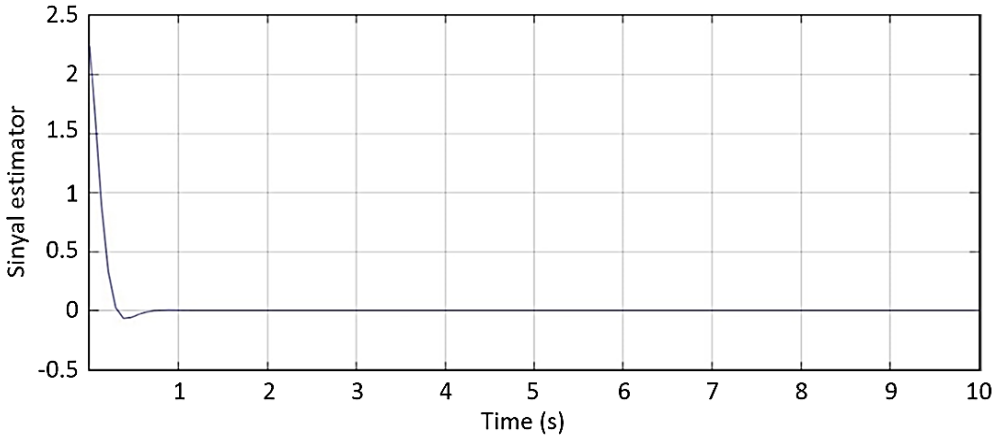
Gambar 9. Perangkat lunak: (a) *flowchart* kendali posisi motor DC, (b) nilai K, L, dan $\overline{N}$

3. HASIL DAN PEMBAHASAN

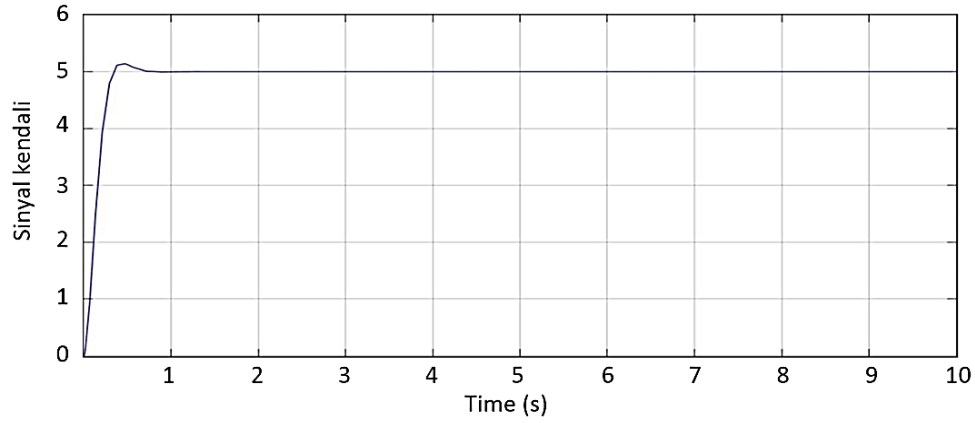
Pengujian dilakukan secara eksperimental pada sistem kendali posisi motor DC menggunakan *state feedback controller* berbasis RTOS. Data yang akan ditampilkan yaitu sinyal kendali, sinyal estimator, dan grafik keluaran pengujian sistem secara keseluruhan. Spesifikasi sistem yang diharapkan pada kendali posisi motor DC dapat dilihat pada Tabel 2. Gambar 10, Gambar 11, dan Gambar 12 merupakan hasil pengujian yang telah dilakukan.

Tabel 2. Spesifikasi pengendalian yang diharapkan

Spesifikasi	Nilai
Maksimum *% overshoot*	≤ 3%
Time sampling (T_s)	0,5 detik
Error steady state (E_{ss})	≤ 10
Desired pole (K_1/K_2)	$0 \leq K_1/K_2 \leq 150$



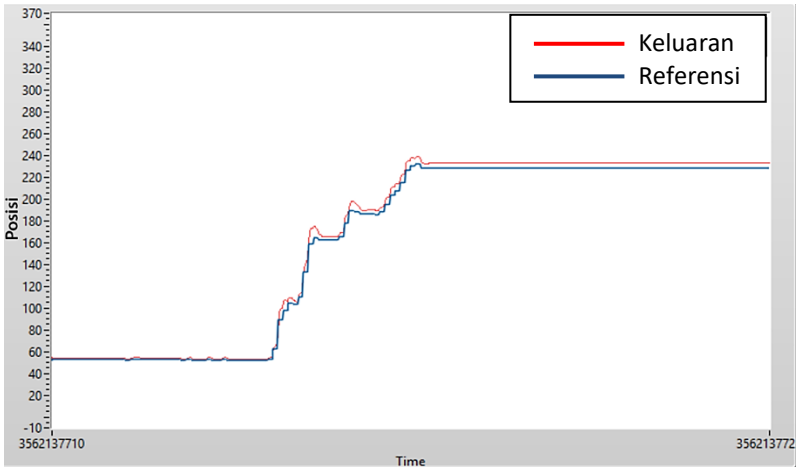
(a)



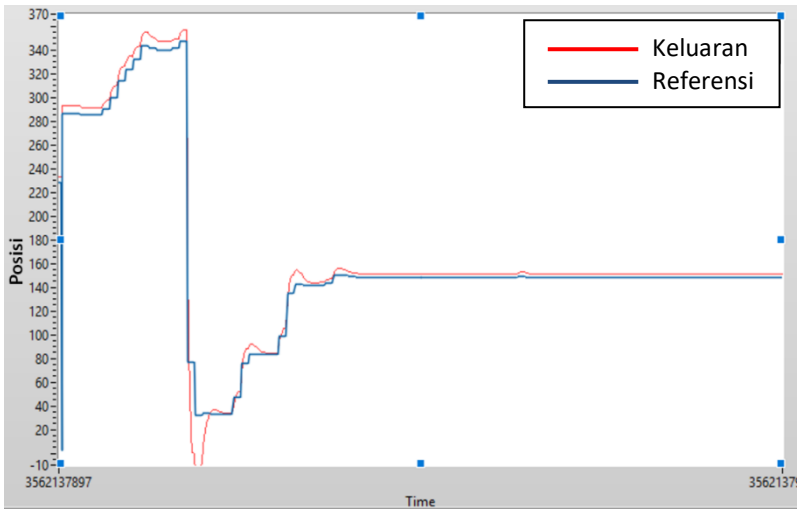
(b)

Gambar 10. Hasil pengujian sistem: (a) respon sinyal estimator, (b) respon sinyal kendali

Gambar 10 merupakan respon keluaran hasil simulasi yang dilakukan pada perangkat lunak MATLAB. Simulasi dengan parameter nilai penguat K, L, dan $\overline{N}$ dilakukan untuk menguji perkiraan respon keluaran sistem kendali posisi motor DC. Parameter *state feedback controller* diperoleh melalui perhitungan dari persamaan (1) hingga (11), serta menggunakan persamaan (12) sebagai fungsi transfer motor DC. Hasil perhitungan tersebut menghasilkan nilai penguat K sebesar 2,66 dan 115,37, nilai penguat $\overline{N}$ sebesar 0,45 dan nilai penguat L sebesar 7,75 dan 0,26.



Gambar 11. Respon keluaran pengujian kesatu



Gambar 12. Respon keluaran pengujian kedua

Hasil sinyal keluaran respon kendali posisi motor ditunjukkan pada Gambar 11 dan Gambar 12. Hasil pengujian menunjukan penggunaan RTOS sebagai inti pemrograman dapat berjalan dengan baik tanpa terdapat *error* atau keterlambatan baik dalam pengukuran sensor ataupun pengiriman sinyal kendali. Hal ini berdampak pada keluaran respon yang tidak sesuai rancangan spesifikasi sistem, dimana sistem memiliki nilai *time sampling* (T_s) sebesar 0,2 detik. Gambar 11 menunjukan respon keluaran sistem memiliki nilai *overshoot* sebesar 234 dan nilai *steady state* sebesar 230 dengan nilai *setpoint* sebesar 228, maka nilai *% overshoot* respon keluaran sebesar 2,63% dan nilai E_{ss} sebesar 2. Gambar 13 memiliki nilai *overshoot* sebesar 154 dan nilai *steady state* sebesar 152 dengan nilai *setpoint* 150, maka nilai *% overshoot* sebesar 2,66% dan nilai E_{ss} sebesar 2. Keluaran sistem kendali posisi motor DC ini sesuai dengan rancangan spesifikasi sistem yang ditampilkan pada Tabel 2.

4. KESIMPULAN

Berdasarkan hasil uji coba pada sistem didapatkan bahwa pengendalian posisi motor DC menggunakan *state feedback controller* dan RTOS dapat berjalan sesuai tujuan dan spesifikasi sistem yang diinginkan. RTOS mampu melaksanakan *task* yang diberikan dengan baik, sehingga selama pengendalian posisi motor DC tidak terjadi *error* dalam pengukuran nilai sensor dan pengiriman sinyal kendali kepada motor DC. Hasil analisa dari data pengujian didapatkan nilai spesifikasi pengendalian yang telah ditentukan dapat dicapai. Penelitian selanjutnya dapat dilakukan pengembangan pada metode pengendalian yang lebih baik, khususnya untuk mengatasi adanya gangguan dan derau pengukuran.

UCAPAN TERIMA KASIH

Terima kasih disampaikan kepada Institut Teknologi Bandung, Sekolah Teknik Elektro dan Informatika, Opsi Kendali dan Sistem Cerdas, atas fasilitas yang penyediaan fasilitas penelitian berupa motor DC Quanser.

REFERENSI

[1] S. K. Das, N. Mondol, and N. A. Sultana, "DESIGN AND IMPLEMENT OF A STATE FEEDBACK POSITION OUTPUT CONTROLLER FOR A MAXON S-DC MOTOR WITH dSPACE," in *International Conference on Mechanical Engineering 2011*, 2011, vol. 2011, pp. 18–20.

[2] K. Ogata, *Modern Control Engineering*, Fifth. New Jersey: Prentice Hall, 2010.

[3] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, Eight. New York: Pearson, 2018.

[4] J. Chotai and K. Narwekar, "Modelling and Position Control of a DC Motor," in *IEEE Proceedings: Electric Power Applications*, 2017, pp. 1–8.

[5] P. Mukherjee and M. Sengupta, "Closed loop speed control of a laboratory fabricated brush-less DC motor drive prototype using position sensor," in *2017 National Power Electronics Conference, NPEC 2017*, 2018, vol. 2018, no. 1, pp. 166–171.

[6] H. P. Wang, "Design of fast fuzzy controller and its application on position control of DC motor," in *2011 International Conference on Consumer Electronics, Communications and Networks, CECNet 2011 - Proceedings*, 2011, pp. 4902–4905.

[7] K. Sharma and D. K. Palwalia, "A modified PID control with adaptive fuzzy controller applied to DC motor, " in *2017 International Conference on Information, Communication, Instrumentation and Control (ICICIC)*, 2017, pp. 1-6.

[8] R. Manikandan and R. Arulmozhiyal, "Position control of DC servo drive using fuzzy logic controller," 2014.

[9] K. Boudaraia, H. Mahmoudi, M. Abbou, and M. Hilal, "DC motor position control of a solar tracking system using second order sliding mode," in *International Conference on Multimedia Computing and Systems -Proceedings*, 2017, vol. 0, no. 3, pp. 594–598.

[10] M. A. Aravind, N. Saikumar, and N. S. Dinesh, "Optimal position control of a DC motor using LQG with EKF," *2017 Int. Conf. Mech. Syst. Control Eng. ICMSC 2017*, no. 2, pp. 149–154, 2017.

[11] M. Mohamed and A. Mahmoud, "Design of State Feedback Gain Matrix for DC Motor Control Based on Damping Ratio and Natural Frequency," *Int. J. Eng. Comput. Sci.*, vol. 2, no. 7, pp. 2186–2188, 2013.

[12] S. A. Kamilu, M. D. A. Hakeem, and L. Olatomiwa, "Design and Comparative Assessment of State Feedback Controllers for Position Control of 8692 DC Servomotor," *Int. J. Intell. Syst. Appl.*, vol. 7, no. 9, pp. 28–33, 2015.

[13] F. Rammig, "Basic concepts of real time operating systems," in *Hardware-dependent Software: Principles and Practice*, Munich, Germany: Springer, 2009, pp. 15–45.