Tugas Hierarchical Composition FSM untuk Kendali Kecepatan Motor

Agape D'sky / 13219010

Contents

1 Spesifikasi

- Sistem yang dikendalikan adalah kecepatan dengan fungsi transfer seperti pada tugas sebelumnya.
- Pengendali menggunakan PID
- Display menggunakan LED 7 segment atau LCD
- Parameter Kp,Ki dan Kd dapat ditampilkan di display
- Parameter Kp,Ki dan Kd dapat diubah menggunakan tombol push button
- Algoritma perubahan parameter meniru seperti pada pengendali REX C100
- Nilai parameter misal dari 0.01 sampai 10.00 (asumsi LED 4 digit)
- Input kecepatan menggunakan potensiometer
- Parameter-parameter sistem dicatat di komputer PC/Laptop (kecepatan, control value, waktu)
- Parameter Kp Ki Kd disimpan di EEPROM / Flash setelah diubah, jadi tidak hilang ketika listrik dimatikan).

- Terdapat 3 buah tombol: set (untuk masuk dan keluar mode setting), + (untuk menambah nilai) dan – (untuk mengurangi nilai)

2 Perhitungan Transformasi Bilinear

Fungsi alih dari sistem ditunjukkan dengan persamaan umum:

$$H(s) = \frac{1}{\tau s + 1}$$

Sesuai spesifikasi, nilai τ = 1.2 sekon, maka persamaannya menjadi:

$$H(s) = \frac{1}{1.2s + 1}$$

Dengan transformasi bilinear, diperoleh fungsi alih pada domain frekuensi diskrit z:

$$H(z) = H(s)|_{s=\frac{2z-1}{Tz+1}} = \frac{T(z+1)}{2\tau(z-1) + T(z+1)} = \frac{T + Tz^{-1}}{(2\tau + T) + (T - 2\tau)z^{-1}}$$

Jika diubah bentuknya ke dalam persamaan diferens, maka diperoleh:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{T + Tz^{-1}}{(2\tau + T) + (T - 2\tau)z^{-1}} \rightarrow Y(z)[(2\tau + T) + (T - 2\tau)z^{-1}] = X(z)[T + Tz^{-1}]$$

$$\leftrightarrow (2\tau + T)y(n) + (T - 2\tau)y(n-1) = Tx(n) + Tx(n-1)$$

$$\leftrightarrow \boldsymbol{y(n) = \frac{Tx(n) + Tx(n-1) - (T - 2\tau)y(n-1)}{(2\tau + T)}} \ldots (\mathbf{1})$$

Di sisi lain, untuk pengendali PID, fungsi alihnya ditunjukkan dengan persamaan berikut:

$$G(s) = Kp + \frac{Ki}{s} + Kd.s = \frac{Kd.s^2 + Kp.s + Ki}{s}$$

Melalui transformasi bilinear, diperoleh fungsi alih pada domain frekuensi diskrit z:

$$G(z) = G(s)|_{s=\frac{2z-1}{Tz+1}} = \frac{Kd.\frac{2}{T}\frac{z-1}{z+1}^2 + Kp.\frac{2}{T}\frac{z-1}{z+1} + Ki}{\frac{2}{T}\frac{z-1}{z+1}}$$

$$= \frac{\frac{(4Kd + 2T.Kp + Ki.T^2)}{2T} + \frac{z^{-1}(-8Kd + 2Ki.T^2)}{2T} + \frac{z^{-2}(4Kd - 2T.Kp + Ki.T^2)}{2T}}{1 - z^{-2}}$$

Persamaan dalam domain z dapat diterjemahkan menjadi fungsi rekursif di domain waktu, yakni:

$$G(z) = \frac{Y(z)}{X(z)} \leftrightarrow Y(z)[1 - z^{-2}]$$

$$= X(z)[\frac{(4Kd + 2T.Kp + Ki.T^2)}{2T} + \frac{z^{-1}(-8Kd + 2Ki.T^2)}{2T}$$

$$+ \frac{z^{-2}(4Kd - 2T.Kp + Ki.T^2)}{2T}]$$

$$\leftrightarrow y(n) = \frac{(4Kd + 2T.Kp + Ki.T^2)}{2T}x(n) + \frac{(-8Kd + 2Ki.T^2)}{2T}x(n-1) + \frac{(4Kd - 2T.Kp + Ki.T^2)}{2T}x(n-2) + y(n-2) \quad \ldots(2)$$

Persamaan (1) dan (2) nantinya akan digunakan untuk implementasi pada kode pengendali dan *plant*.

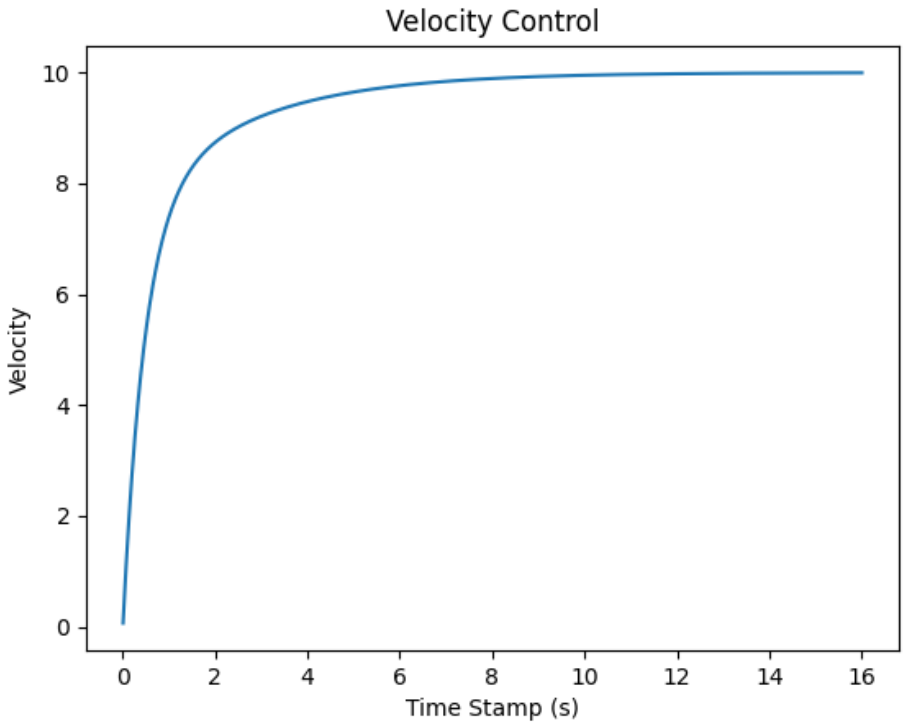
3 Simulasi Kendali

Simulasi terhadap time response dari persamaan dibuat dalam bahasa Python dan diplot dengan menggunakan matplotlib.pyplot. Kode dan hasilnya ditunjukkan pada Gambar 1 dan 2.

```
import matplotlib.pyplot as plt

#controller
kp = 2
ki = 1
kd = 0
sampling_time = 0.008

#plant
tau = 1.2; T = 0.008;

###############

target = 10.;
err = 0.; err_1 = 0.; err_2 = 0.;
effort = 0.; effort_1 = 0.; effort_2 = 0.;
output = 0.; output_1 = 0.;

a =
(4*kd+2*sampling_time*kp+sampling_time*sampling_time*ki)/(2*sampling_time);
b = (2*sampling_time*sampling_time*ki-8*kd)/(2*sampling_time);
c = (4*kd-
2*sampling_time*kp+sampling_time*sampling_time*ki)/(2*sampling_time);
d = 1.;

print(a,b,c,d)

def compute_control():
    return a*err+b*err_1+c*err_2+d*effort_2;

def compute_plant():
    return effort_1/(2*tau+T)*T + effort/(2*tau+T)*T - output*(T-
2*tau)/(2*tau+T);

###############

fig = plt.figure()
i = 0
x = list()
y = list()

while i < 2000 :
    err = target-output;
```

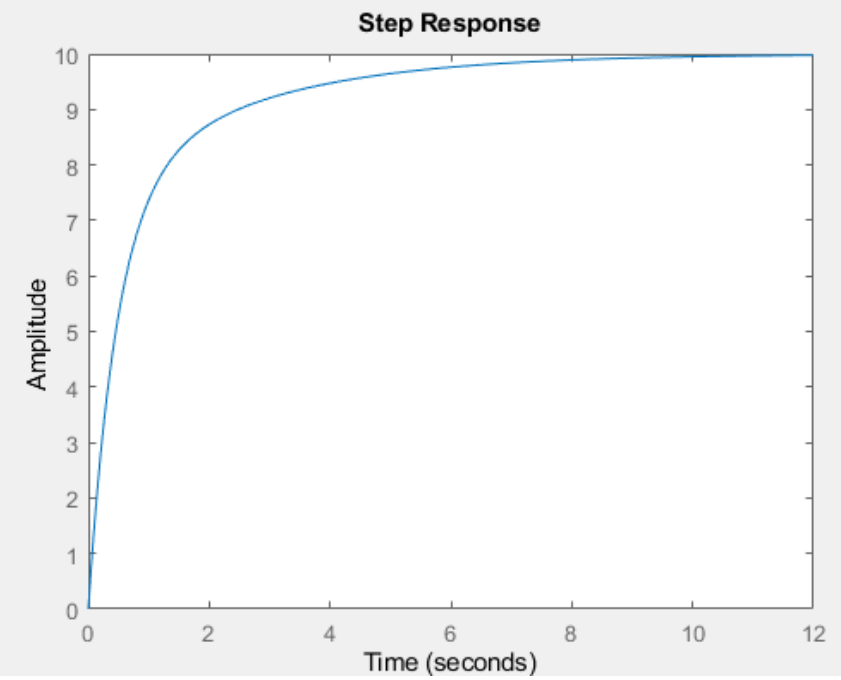
```
    effort = compute_control();
    output = compute_plant();
    i = i + 1
    y.append(output)
    x.append(i*T)

    err_2 = err_1;
    err_1 = err;
    effort_2 = effort_1;
    effort_1 = effort;
    output_1 = output;

plt.plot(x,y)
plt.title('Velocity Control')
plt.xlabel('Time Stamp (s)')
plt.ylabel('Velocity')
plt.show()
```

Gambar 1 Kode Simulasi Step Respons Kendali Feedback dengan Kontroller PID (Kp=2, Ki=1, Kd=0)



Gambar 2 Hasil Simulasi Step Respons Kendali Feedback dengan Kontroller PID (Kp=2, Ki=1, Kd=0)

Jika disetarakan dengan hasil simulasi di MATLAB, diperoleh hasil yang kurang lebih sama secara visual, ditunjukkan pada Gambar 3.

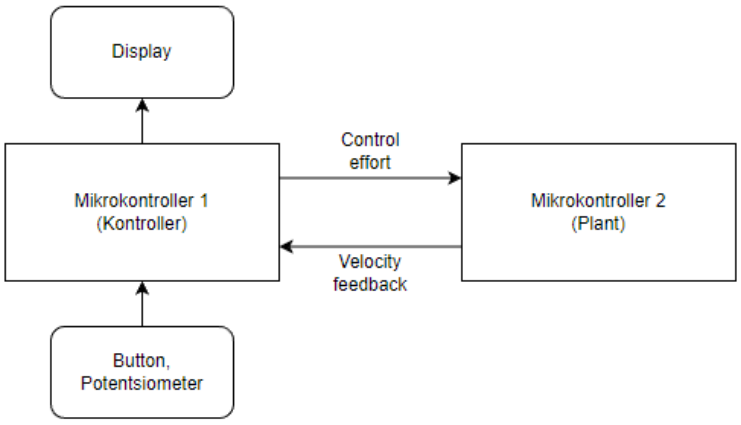


Gambar 3 Hasil Simulasi Step Respons Kendali Feedback dengan Kontroller PID pada MATLAB (Kp=2, Ki=1, Kd=0)

Dengan demikian, bisa dikatakan bahwa algoritma rekursif yang dibuat sesuai dengan yang diharapkan.

4 Implementasi Sistem pada Mikrokontroler

Secara umum, sistem dideskripsikan dengan gambar berikut:

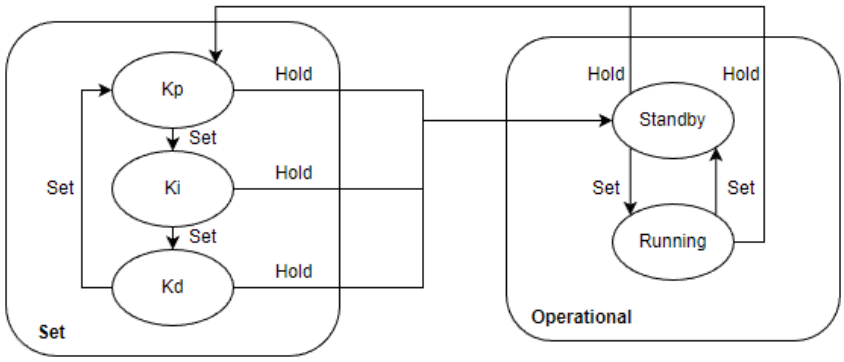


Gambar 4 Abstraksi Level Atas Sistem

Sistem dibangun dengan menggunakan 2 buah mikrokontroller. Mikrokontroller pertama berperan sebagai pengendali, kanal input, dan display. Mikrokontroller pertama mengirimkan informasi berupa *control effort* yang dipakai untuk mengendalikan plant (mikrokontroller 2). Dalam setiap prosesnya, mikrokontroller 2 juga mengirimkan *feedback* berupa kecepatan *plant* di waktu tersebut.

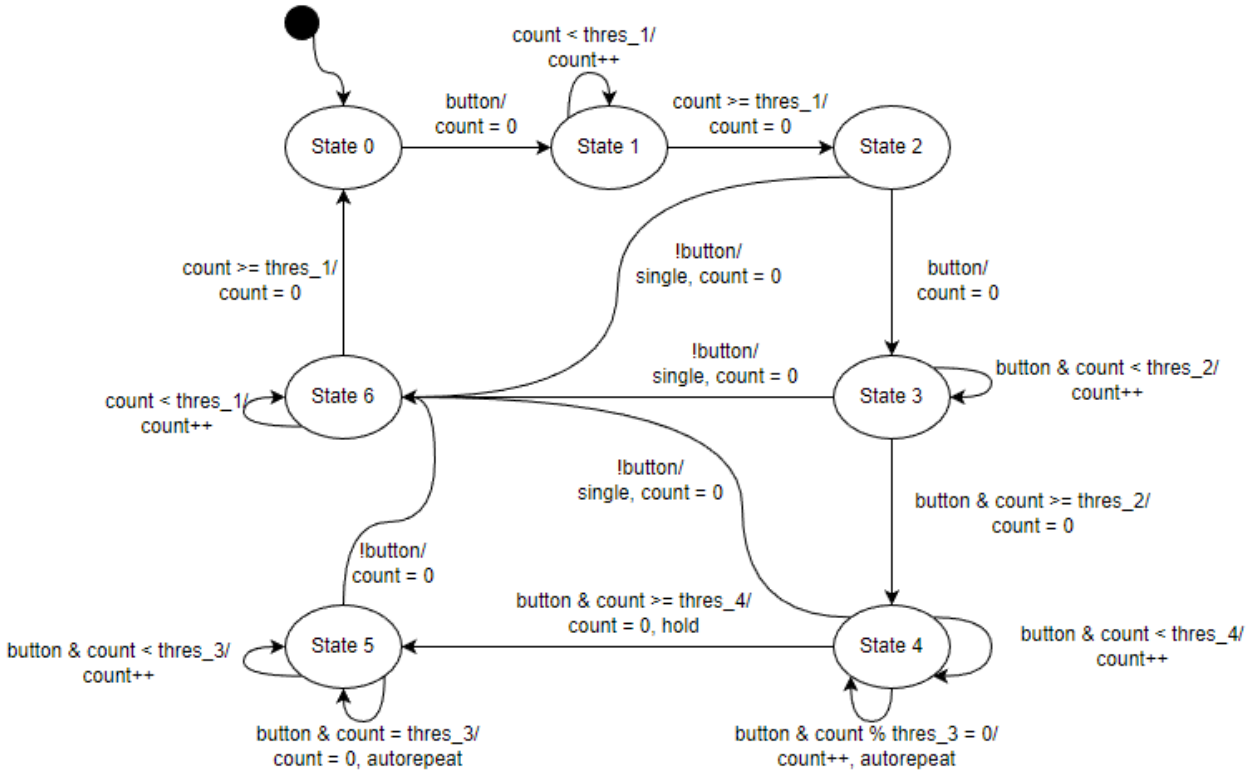
4.1 Desain Pengendali

Pengendali pada mikrokontroller pertama dibuat dengan beberapa unsur. Aspek utama dari pengendali adalah *behavior* FSM yang ditunjukkan pada Gambar 5.



Gambar 5 FSM Kontroller PID

FSM disusun secara *hierarchical* dengan menggunakan 2 buah *super state* (**Set** dan **Operational**). *Super state* **Set** digunakan untuk pengaturan parameter kendali, yakni **Kp**, **Ki**, dan **Kd**, sedangkan **Operational** digunakan untuk memulai proses pengendalian. Pergantian antara *super state* dilakukan dengan perintah *hold*, sedangkan perubahan *state* internal dilakukan dengan perintah *set*. Adapun, perintah-perintah ini diperoleh melalui tombol-tombol yang diatur dengan menggunakan FSM lain.

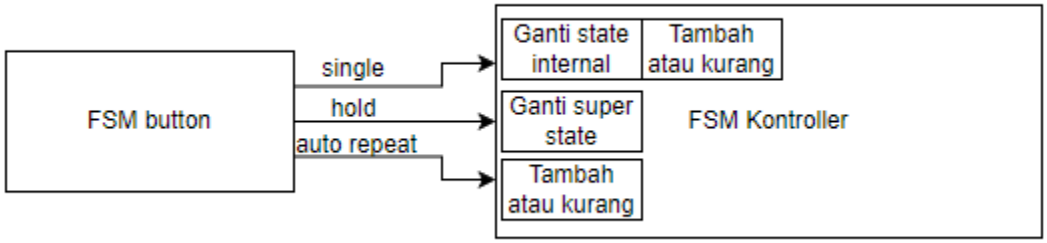


Gambar 6 FSM Tombol

FSM tombol dibuat untuk dapat memberikan 3 jenis informasi dari sebuah tombol, antara lain *single command* (penekanan singkat), *hold command* (penekanan dengan interval waktu tertentu), dan *auto repeat command* (penekanan yang ditahan, dibaca sebagai penekanan singkat berulang), dan di saat yang sama juga digunakan untuk melakukan *filtering* terhadap *debouncing effect* yang muncul dari tombol. FSM ini terdiri atas 7 buah *state* dan dikendalikan dengan sebuah input *button*.

Pada kondisi awal, sistem memasuki *state* 0. Ketika input on diberikan, *state* berpindah ke *state* 1 dan menunggu sampai *counting* selesai (untuk menghilangkan *debouncing*). Setelah penghitungan selesai, sistem beralih ke *state* 2. Jika *button* masih ditekan, maka *state* segera berpindah ke *state* 3. Di sini, FSM menghitung sekuens sampai interval tertentu, sampai *auto repeat* bisa dideteksi. Saat kondisi tersebut tercapai, *state* berpindah ke *state* 4. Di *state* 4, dilakukan penungguan sampai sinyal *hold* dapat dikeluarkan, sekaligus juga berjalan dengan keluaran *auto repeat*. Ketika *threshold* tercapai, *state* beralih ke *state* 5, di mana *hold* akan dikeluarkan. Dari *state* 2 sampai *state* 5, apabila suatu ketika *button* dilepas, *state* berpindah ke *state* 6 untuk dilakukan penghitungan sehingga terhindar dari *debouncing* pada *negative edge* sinyal tombol.

Penggabungan dari kedua FSM di atas dapat diilustrasikan sebagai berikut:



Gambar 7 Hubungan antara Kedua FSM

4.2 Implementasi FSM Tombol dan Kontroller

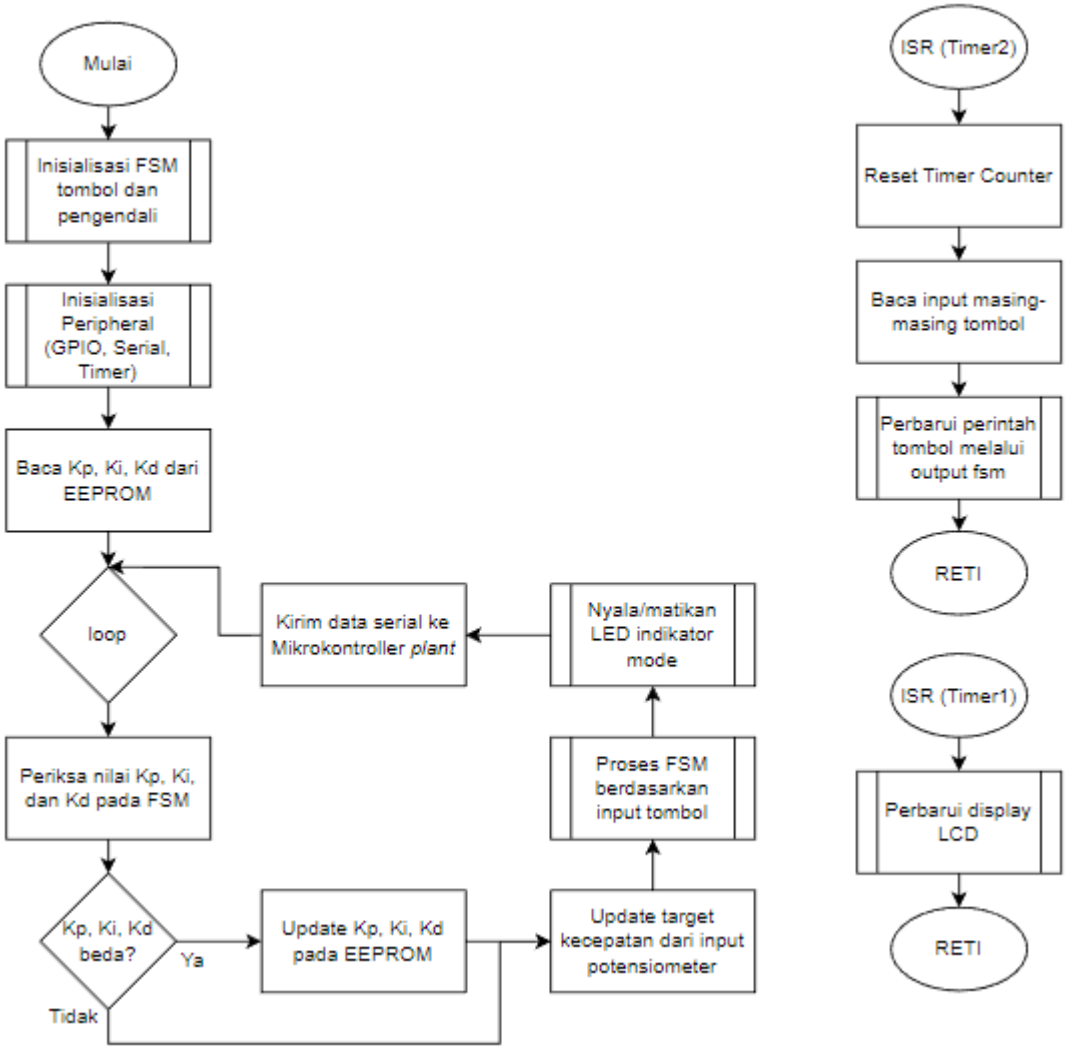
Implementasi dari desain pada sub 4.1 dilakukan dengan bahasa C++, yakni dituliskan pada *file* button_fsm.cpp, button_fsm.h, device_fsm.cpp, dan device_fsm.h. Seluruh *file* ini dilampirkan pada *repository* di *folder ~/Tugas Hierarchical Composition/src/*.

4.3 Implementasi Program Pengendali pada Mikrokontroller Pertama

Pengendali yang dibuat dalam bentuk gabungan FSM tombol dan FSM kontroller diimplementasikan pada mikrokontroller Arduino Uno (sebagai mikrokontroller pertama di Gambar 4). *Flowchart* implementasi ditunjukkan pada Gambar 8. Terdapat 3 bagian utama dari program, yakni *main loop*, ISR Timer 1, dan ISR Timer 2. *Main loop* digunakan untuk pemrosesan perintah tombol ke dalam algoritma pengendalian. Pada *main loop*, dilakukan eksekusi perintah dari tombol, pembacaan target melalui potensiometer, dan pengiriman informasi *control effort* (hasil kalkulasi pengedali) ke *plant*. ISR Timer 2 digunakan untuk mengubah informasi input tombol menjadi perintah *single*, *hold*, dan *auto repeat* yang selanjutnya dikirim ke FSM kontroller. Terakhir, ISR Timer 1 dipakai untuk melakukan pembaruan terhadap *display*.

Pembagian seperti ini dibuat agar masing-masing sistem bisa dieksekusi dengan periode pemrosesan yang terpisah. Kendali dibuat dalam 8 ms, pembacaan input tombol dibuat dalam 100 us, sedangkan *display*

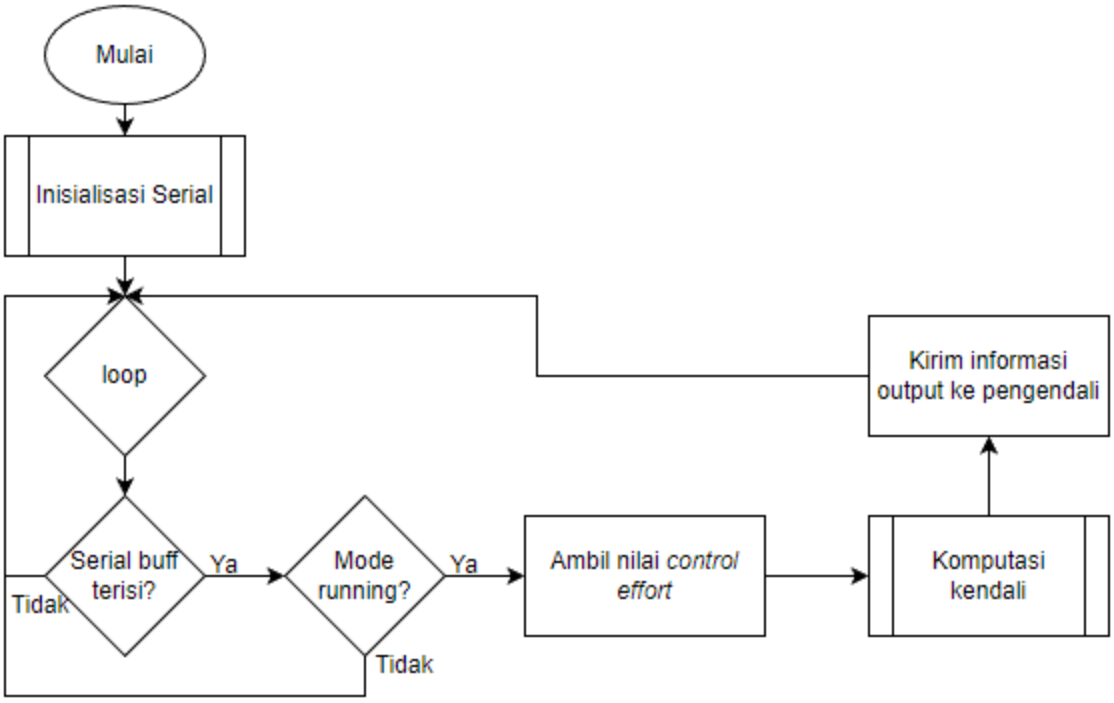
dibuat dalam 160 ms. Angka-angka tersebut dipilih untuk memberikan akurasi dan kenyamanan dari sudut pandang pengguna.



Gambar 8 Flowchart Program Pengendali pada Mikrokontroller Pertama

4.4 Implementasi Program *Plant* pada Mikrokontroller Kedua

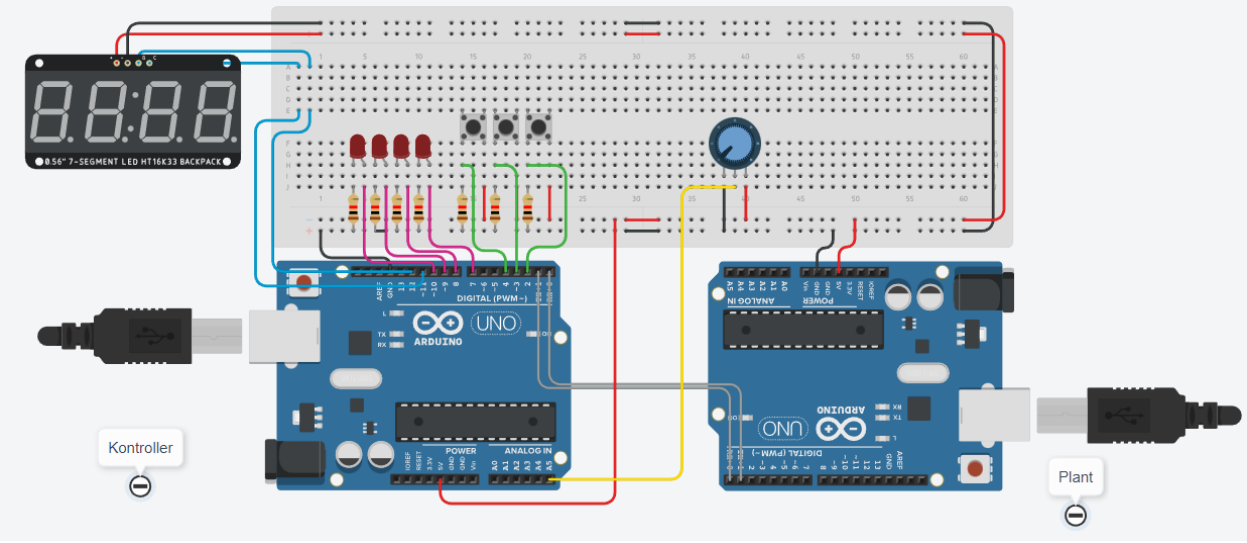
Sebagai *plant*, kembali digunakan mikrokontroller Arduino Uno.Tugas dari *plant* adalah menerima input dari pengendali (berupa *control effort*) dan melakukan penghitungan untuk memperoleh output kecepatan berdasarkan *control effort* dan mode kerja yang sedang berjalan. Komputasi hanya dilakukan pada mode kerja *Operational-Running*. Output kecepatan lalu dikirimkan balik ke mikrokontroller pengendali untuk dapat melakukan kalkulasi kendali *timestep* selanjutnya. *Flowchart* dari program ditunjukkan pada Gambar 9.



Gambar 9 Flowchart Program Plant pada Mikrokontroller Kedua

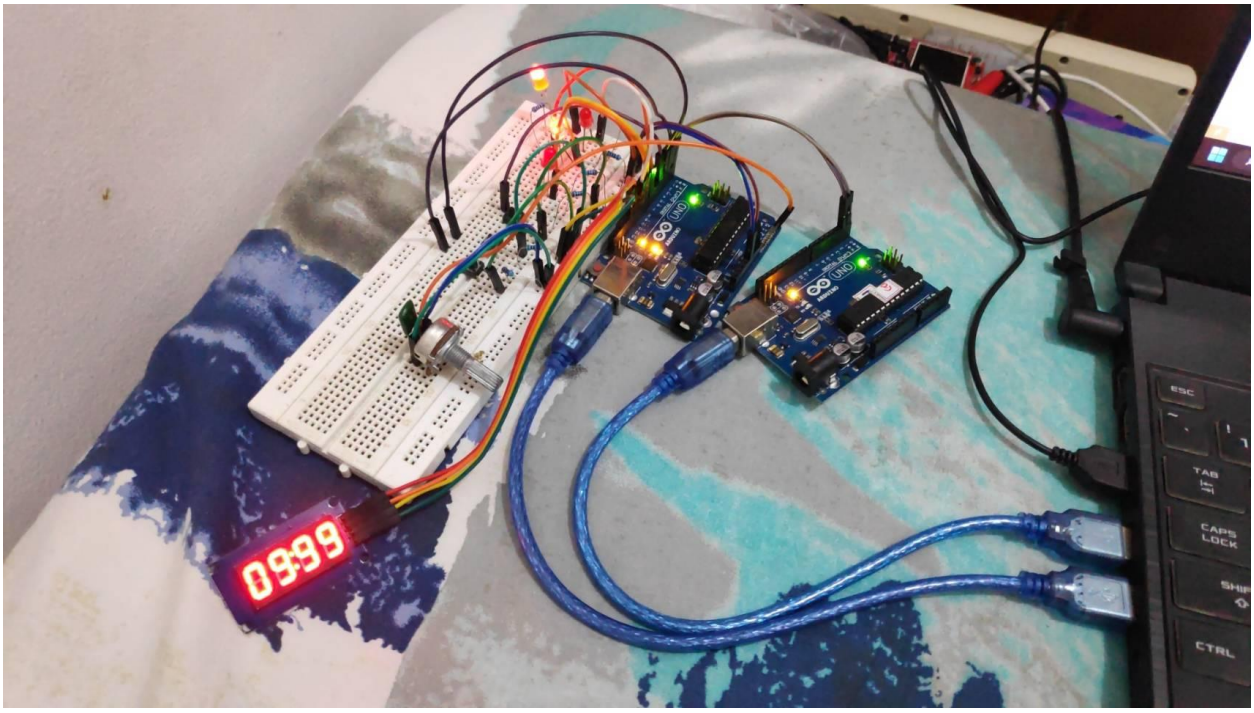
4.5 Konfigurasi Hardware

Konfigurasi *hardware* dari pengendali dan *plant* ditunjukkan pada Gambar 10. *Wire* berwarna merah, hitam, pink, biru, kuning, hijau berturut-turut menunjukkan jalur 5V, GND, output digital, data *display*, sinyal analog, dan input digital. Terdapat 3 buah tombol untuk fungsi +, -, dan set. Lalu, terdapat 4 buah indikator LED (1 untuk menunjukkan *super state*, yang lainnya menunjukkan *substate*).



Gambar 10 Rangkaian yang Digunakan untuk Uji Coba

Konfigurasi riil dari rangkaian Gambar 10 ditunjukkan pada Gambar 11.



Gambar 11 Implementasi Riil Rangkaian Gambar 10

4.6 Hasil Simulasi

4.6.1 Simulasi FSM Tombol

FSM tombol disimulasikan dengan menggunakan program *button_sim.cpp* pada *folder simulation*. Program menerima input tombol artifisial (dengan menggunakan input terminal) dan menampilkan variabel-variabel pada pengendali untuk setiap input. Simulasi ditujukan untuk melihat apakah ketiga buah keluaran yang dihasilkan muncul pada saat yang tepat. Untuk kemudahan dalam simulasi, seluruh *threshold* diberi nilai 2.

Gambar 12 menunjukkan simulasi untuk pemunculan output *single*.

```
single: 0  hold: 0  auto: 0  state: 0  count: 0
1
single: 0  hold: 0  auto: 0  state: 1  count: 0
1
single: 0  hold: 0  auto: 0  state: 1  count: 1
1
single: 0  hold: 0  auto: 0  state: 1  count: 2
1
single: 0  hold: 0  auto: 0  state: 2  count: 0
0
single: 1  hold: 0  auto: 0  state: 6  count: 0
0
single: 0  hold: 0  auto: 0  state: 6  count: 1
0
single: 0  hold: 0  auto: 0  state: 6  count: 2
0
single: 0  hold: 0  auto: 0  state: 0  count: 0
```

Gambar 12 Simulasi FSM Tombol : Output Single

Dari gambar, dapat dilihat bahwa sistem baru berpindah *state* ke *state* 1 ketika input dinyalakan. Lalu, pada *state* 1, dilakukan penghitungan sampai *threshold* terlampaui, sebelum akhirnya menuju *state* 2. Pada *state* 2, diberikan input off (menandakan penekanan tombol yang singkat), sehingga output *single* langsung dimunculkan.

Sebagai lanjutan simulasi, dilakukan juga pemeriksaan terhadap keluaran *hold* dan *auto correct*, sebagaimana ditunjukkan Gambar 13. Input on diberikan secara terus menerus, sehingga sistem dapat bergeser sampai *state* 5. Pada peralihan *state* 4 ke 5, keluaran *hold* diberikan, dan *auto repeat* terjadi secara berulang pada *state* 5. Untuk mencapai *state* 5, sistem perlu menunggu sampai *threshold* 1 terlewati di *state* 1 (menghilangkan *debouncing*), *threshold* 2 terlewati di *state* 3 (waktu tunggu sebelum *auto repeat*), dan *threshold 4* di *state* 4 (waktu tunggu sebelum *hold*). Dengan demikian, perilaku dari FSM sudah sesuai dengan harapan.

```
single: 0  hold: 0  auto: 0  state: 0  count: 0
1
single: 0  hold: 0  auto: 0  state: 1  count: 0
1
single: 0  hold: 0  auto: 0  state: 1  count: 1
1
single: 0  hold: 0  auto: 0  state: 1  count: 2
1
single: 0  hold: 0  auto: 0  state: 2  count: 0
1
single: 0  hold: 0  auto: 0  state: 3  count: 0
1
single: 0  hold: 0  auto: 0  state: 3  count: 1
1
single: 0  hold: 0  auto: 0  state: 3  count: 2
1
single: 0  hold: 0  auto: 0  state: 4  count: 0
1
single: 0  hold: 0  auto: 0  state: 4  count: 1
1
single: 0  hold: 0  auto: 0  state: 4  count: 2
1
single: 0  hold: 1  auto: 0  state: 5  count: 1
1
single: 0  hold: 0  auto: 0  state: 5  count: 2
1
single: 0  hold: 0  auto: 1  state: 5  count: 0
1
single: 0  hold: 0  auto: 0  state: 5  count: 1
1
single: 0  hold: 0  auto: 0  state: 5  count: 2
1
single: 0  hold: 0  auto: 1  state: 5  count: 0
0
single: 0  hold: 0  auto: 0  state: 6  count: 0
```

Gambar 13 Simulasi FSM Tombol : Output Hold dan Auto Repeat

4.6.2 Simulasi FSM Pengendali

FSM pengendali disimulasikan melalui program *device_sim.cpp* di *folder simulation*. Program dibuat untuk menerima 4 buah input secara berturut-turut sebagai logika *plus*, *minus, set,* dan *hold*. Setelah itu, diperlihatkan *return value* dari FSM, yakni nilai *super_state, operational_state, set_state*, dan *value*. Adapun, nilai *value* bergantung erat dengan mode yang sedang bekerja. Hasil simulasi ditunjukkan pada Gambar 14.

```
0 0 0 0
SuperState:0   OperationalState:0   SetState:0   Value:0.000000
0 0 1 0
SuperState:0   OperationalState:1   SetState:0   Value:0.000000
0 0 1 0
SuperState:0   OperationalState:0   SetState:0   Value:0.000000
0 0 0 1
SuperState:1   OperationalState:0   SetState:0   Value:2.000000
1 0 0 0
SuperState:1   OperationalState:0   SetState:0   Value:2.500000
0 1 0 0
SuperState:1   OperationalState:0   SetState:0   Value:2.000000
0 0 1 0
SuperState:1   OperationalState:0   SetState:1   Value:1.000000
1 0 0 0
SuperState:1   OperationalState:0   SetState:1   Value:1.500000
0 0 1 0
SuperState:1   OperationalState:0   SetState:2   Value:0.000000
0 1 0 0
SuperState:1   OperationalState:0   SetState:2   Value:0.000000
0 0 0 1
SuperState:0   OperationalState:0   SetState:2   Value:0.000000
```

```
enum SuperState {
    OPERATIONAL,
    SET
};
enum OperationalState {
    STANDBY,
    RUNNING
};
enum SetState {
    KP,
    KI,
    KD
};
```

Gambar 14 (kiri) Simulasi FSM Pengendali (kanan) Enumerasi State yang Tersedia

Nilai-nilai *state* ditunjukkan dengan angka enumerasi, ditunjukkan pada Gambar 14 kanan. Pada kondisi awal, FSM berada pada *state* **Operational-Standby** (*value* menampilkan target kecepatan). Ketika input *set* diberikan, *substate* **Standby** bergeser ke **Running** (*value* menampilkan kecepatan saat itu). Besar kecepatan 0 karena *feedback* pada kode selalu diberikan sebagai 0. Lalu, ketika tombol *hold* ditekan, *super state* berpindah ke mode **Set-Kp** (Kp ditunjukkan bernilai 2.0). Ketika tombol *plus* ditekan, Kp bertambah sebanyak 0.5, begitu juga ketika tombol *minus* ditekan, Kp berkurang 0.5. Ketika tombol *set* ditekan lagi, *set state* berpindah secara sekuensial (0 → 1 → 2 → 0, dan seterusnya) dan tampilan *value* juga berubah-ubah (menunjukkan Kp, Ki, dan Kd). Sistem dikembalikan ke *super state* **Operational** dengan cara memberikan input *hold* lagi.

Simulasi kendali dari sistem tidak lagi dilakukan karena algoritmanya sudah diujikan pada program Python (Lihat Gambar 1).

4.6.3 Simulasi Gabungan

Simulasi dilakukan dengan menjalankan program pada kontroller dan *plant* secara konkuren dengan menggunakan *hardware*. Hasil dari simulasi bisa diamati secara visual (melalui LED dan *display* TM1637). Adapun, hasil simulasi dibuat dalam bentuk video yang bisa diakses pada *repository*.

5 Kesimpulan

- FSM dari pengendali dan *button* dapat dibuat dengan baik dan menyerupai devais acuan REXC100, yakni ditinjau dari fungsi *auto repeat*, pengubahan parameter PID, penggunaan variabel non-volatile, dan kemultifungsian tombol
- Simulasi dengan *software* menunjukkan algoritma pada FSM tombol dan kendali berjalan sesuai dengan harapan perancangan
- Simulasi langsung dengan mikrokontroller menunjukkan sistem berjalan dengan baik

6 Lampiran

Link repository: https://github.com/AgapeDsky/Tugas-Hierarchical-Composition

Video terlampir pada repository