Tugas I Sistem Kendali Modern

KONTROL TEMPERATUR DENGAN PID

Diajukan untuk memenuhi Tugas Mata Kuliah

Sistem Kendali Modern (*Kontrol PID*)

Oleh :

Edhy Susanto	**03.2020.1.90715**
Hanifah Azizah	**03.2020.1.90724**
Ainur Rohman	**03.2020.1.90725**

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK ELEKTRO DAN TEKNOLOGI INFORMASI

INSTITUT TEKNOLOGI ADHI TAMA SURABAYA

2021

BAB I
LANDASAN TEORI

1.1.Time Response

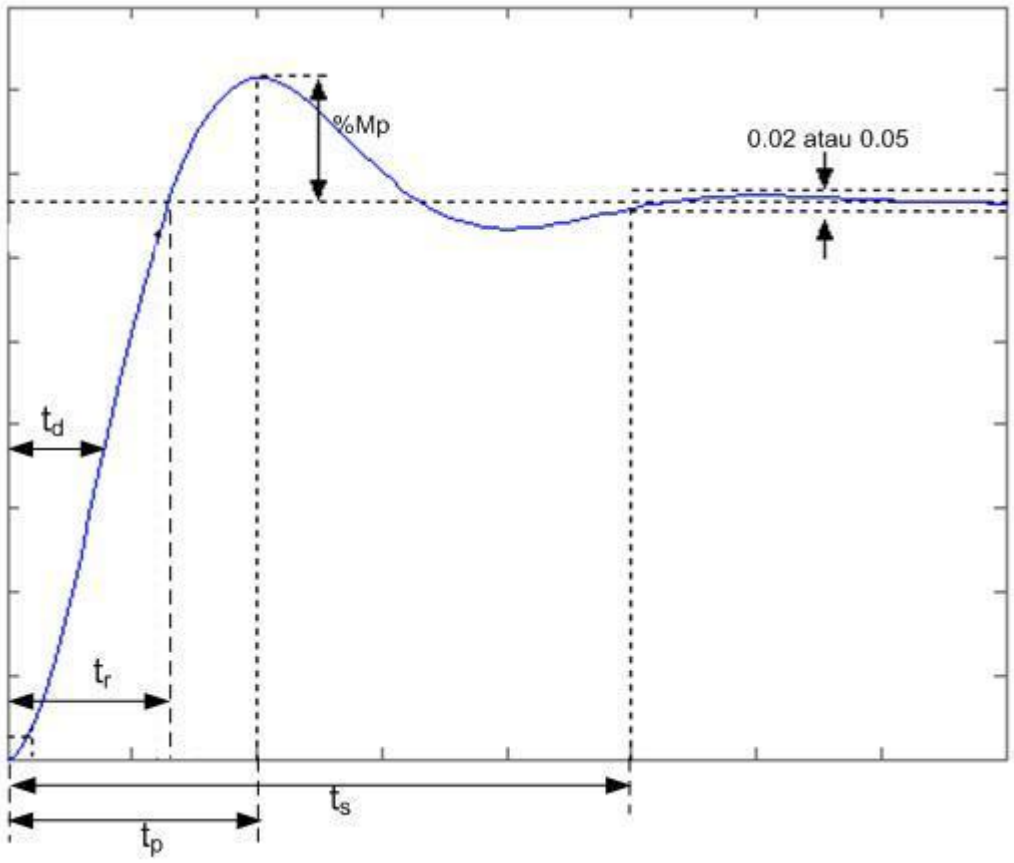
1.1.1. Keadaan Tunak (Steady State)

Keadaan tunak (steady state) adalah kondisi sewaktu sifat-sifat suatu sistem tak berubah dengan berjalannya waktu (konstan). Pada keadaan ini, turunan parsial terhadap waktu untuk setiap properti p dari sistem adalah nol. Pada kebanyakan sistem, keadaan tunak baru akan dicapai beberapa waktu setelah sistem dimulai atau diinisiasi. Kondisi awal ini sering disebut sebagai keadaan transien. Nilai output dari suatu sistem pada kondisi tunak disebut final value. Keadaan tunak (steady state) dari properti p dituliskan secara matematika dengan :

$$\frac{\partial \mathrm{p}}{\partial \mathrm{t}} = 0$$

1.1.2. Respon Transien (Transient Response)

Respon transient adalah respon sistem yang berlangsung dari keadaan awal sampai keadaan tunak, sedang respon steady state adalah kondisi keluaran sesudah respon transien berakhir hingga waktu relatif tak terhingga.



Dari grafik di atas diketahui karakteristik keluaran sistem orde dua terhadap masukan *unit step*, yaitu:

1. Waktu tunda (*delay time*), t_d

   Ukuran waktu yang menyatakan faktor keterlambatan respon output terhadap input, diukur mulai t = 0 s/d respon mencapai 50% dari respon *steady state*.

2. Waktu naik (*rise time*), t_r

   Waktu naik adalah ukuran waktu yang di ukur mulai dari respon $t=0$ sampai dengan respon memotong sumbu *steady state* yang pertama

3. Waktu puncak (*peak time*), t_p

   Waktu puncak adalah waktu yang diperlukan respon mulai dari $t=0$ hingga mencapai puncak pertama *overshoot*.

4. *Overshoot* maksimum, *Mp*

   Nilai reltif yang menyatakan perbandingan antara nilai maksimum respon (*overshoot*) yang melampaui nilai *steady state* dibanding dengan nilai *steady state*

5. Waktu tunak (*settling time*), t_s

   Waktu tunak adalah ukuran waktu yang menyatakan respon telah masuk ±5%, atau ±2%, atau ±0.5% dari keadaan *steady state*

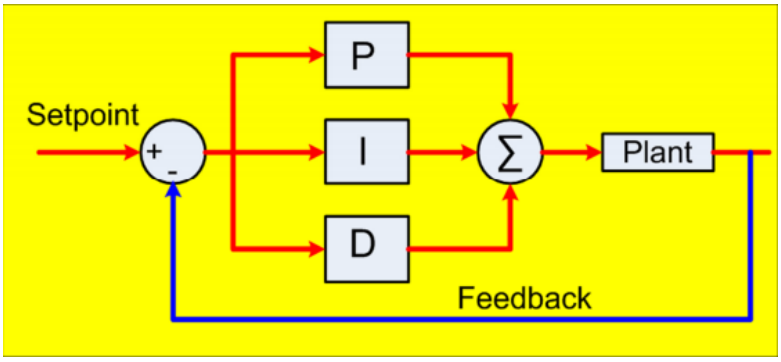
6. Steady-state error : Steady-state error adalah perbedaan antara hasil akhir yang diinginkan (set point) dan hasil akhir sebenarnya ketika sistem mencapai steady state (final value).

1.2 Sistem Kontrol PID

1.2.1 Definisi Kontroller PID

Sistem kontrol PID (*Proportional Integral Derivative*) merupakan sistem kontrol loop tertutup. Sistem kontrol ini memiliki respon terhadap sinyal input cepat dengan kesalahan kecil dan dapat meredam osilasi. Overshoot dari sistem pengontrolan posisi lebih kecil daripada overshoot dengan menggunakan sistem tanpa pengontrol PID [1].

Respon optimal PID ini bisa dilakukan dengan mangatur parameter Kp, Ki dan Kd. Pengaturan parameter nilai tersebut akan mempengaruhi rise time, settling time, overshoot dan error steady state.



Gambar 1.1 Control system dengan PID

(Sumber: Dokumentasi Kuliah Sistem Kendali Modern, Wahyu S. Pambudi)

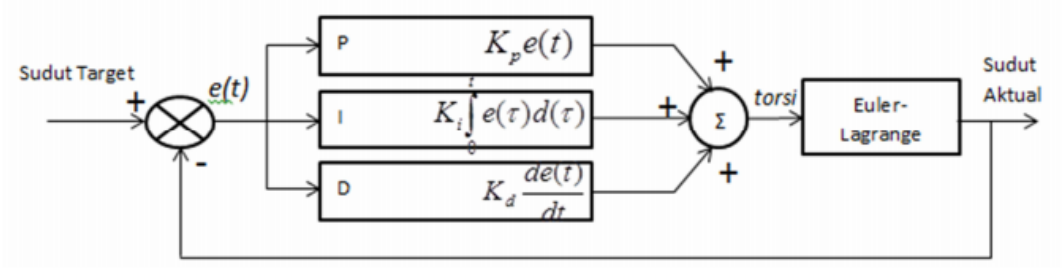
Kontroler mencoba untuk meminimalkan nilai kesalahan setiap waktu dengan penyetelan variabel kontrol, seperti posisi keran kontrol, damper, atau daya pada elemen pemanas ke nilai baru yang ditentukan oleh rumus kontroler

PID. Pada model ini, masing-masing koefisien memiliki peran masing-masing antara lain :

1. P bertanggung jawab untuk nilai kesalahan saat ini. Jika nilai kesalahan besar dan positif, maka keluaran kontrol juga besar dan positif.
2. I bertanggung jawab untuk nilai kesalahan sebelumnya. Contoh, jika keluaran saat ini terlalu kecil, maka kesalahan terakumulasi terus menerus dan kontroler merespon dengan keluaran lebih tinggi.
3. D bertanggung jawab untuk kemungkinan nilai kesalahan mendatang berdasarkan pada laju perubahan tiap waktu.

Karena kontroler PID hanya mengandalkan variabel proses terukur dan bukan mengandalkan pengetahuan mengenai prosesnya, maka kontroler PID dapat secara luas digunakan. Dengan penyesuaian (tuning) ketiga parameter model, kontroler PID dapat memenuhi kebutuhan proses. Respon kontroler dapat dijelaskan dengan bagaimana responnya terhadap kesalahan, besarnya overshoot dari setpoint, dan derajat osilasi sistem. penggunaan algoritma PID tidak menjamin kontrol optimum sistem atau bahkan kestabilannya. Beberapa aplikasi mungkin hanya menggunakan satu atau dua term untuk memberikan kontrol sistem yang sesuai. Hal ini dapat dicapai dengan mengontrol parameter yang lain menjadi nol. Kontroler PID dapat menjadi kontroler PI, PD, P atau I tergantung aksi apa yang digunakan.

1.2.2 Rumus Umum Kontroller PID



Gambar 1.2 Blok Diagram kontrol PID

(Sumber: Dokumentasi Kuliah Sistem Kendali Modern, Wahyu S. Pambudi) [2]

Controller P merupakan hasil perkalian nilai error dengan Kp yang ditentukan. Controller I merupakan hasil perkalian Ki dengan integral error. D controller merupakan hasil perkalian Kd dengan turunan error. Kp, Ki dan Kd adalah nilai gain yang ditetapkan agar nilai actual sama dengan setpoint. Jika diintrepretasikan dalam waktu, maka P menggunakan error waktu sekarang (present), I menggunakan akumulasi error lampau (past) dan D controller menggunakan prediksi error yang akan datang (future) berdasarkan rasio perubahan untuk mendapatkan nilai aktual yang sama dengan setpoint [2].

Persamaan sinyal control dapat dirumuskan seperti persamaan berikut ini.

$$\mathrm{u(t)} = \mathrm{Kp\ e(t)} + \mathrm{Ki} \int_0^{\mathrm{t}} \mathrm{e(t)\ dt} + \mathrm{Kd} \frac{\mathrm{de\,(t)}}{\mathrm{dt}}. \quad \text{..... (1.1)}$$

Keterangan:

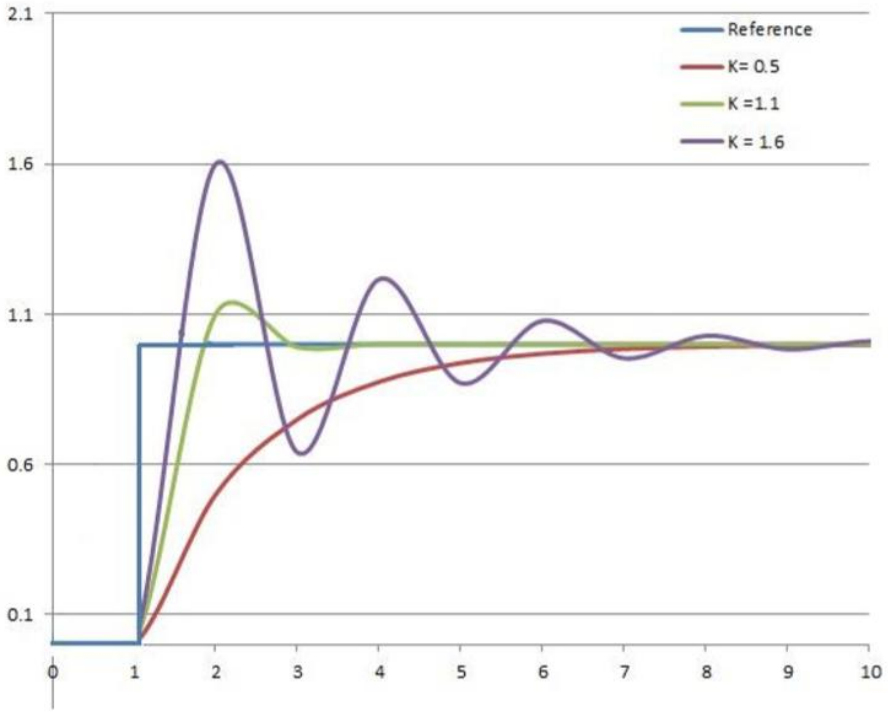
$u(t)$	: sinyal control
$e(t)$	: sinyal error
Kp	: Nilai parameter kontrol proportional
Ki	: Nilai parameter kontrol integral
Kd	: Nilai parameter kontrol derivative/diferensial

2.2.3. Term Proporsional

Term proporsional akan menghasilkan nilai keluaran yang berbanding lurus dengan nilai kesalahan. Responnya dapat diatur dengan mengalikan kesalahan (error) dengan konstanta Kp, disebut konstanta gain proporsional atau gain kontroler. Term proporsional dirumuskan:

$$Pout = Kp\ e(t)$$

$$Pout = Kp\ (Ysp - Ym)$$



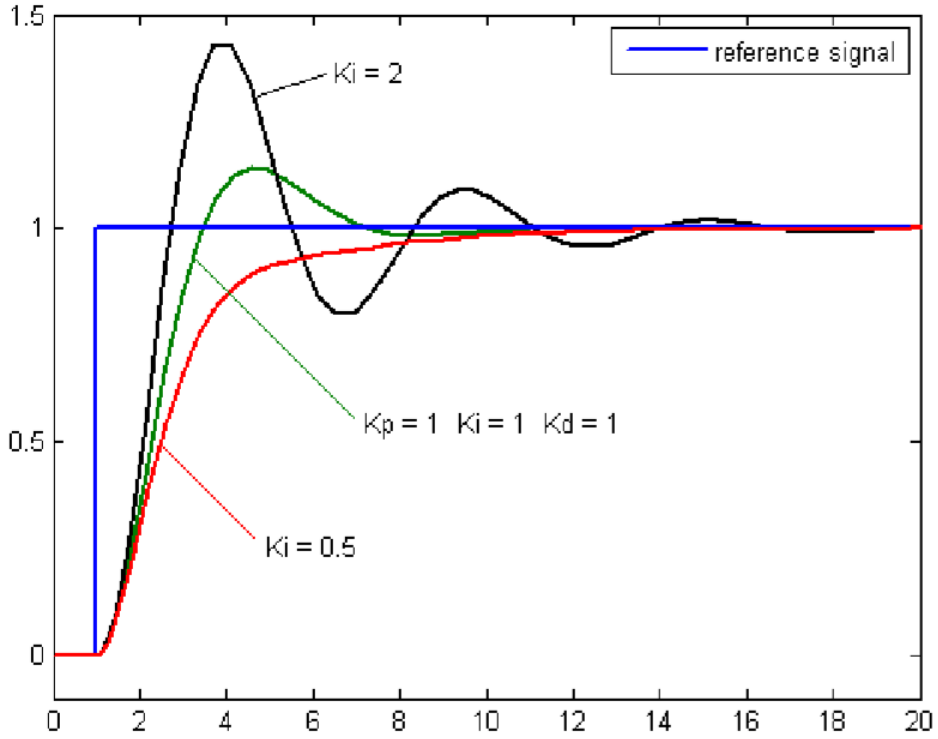
Gambar 1.2 Plot Y vs waktu, untuk 3 nilai Kp (Ki dan Kd dijaga konstan)

Grafik 1.2 menunjukkan bagaimana variasi nilai Kp berpengaruh terhadap time response dari suatu sistem. Gain yang besar menghasilkan perubahan yang besar pada keluaran untuk suatu nilai kesalahan tertentu. Namun, jika gain terlalu besar, sistem membutuhkan waktu yang cukup lama untuk mencapai kondisi steady-state. Sebaliknya, gain yang bernilai kecil maka respon keluaran juga kecil, sehingga kontroler menjadi kurang responsif/sensitif, hal ini mengakibatkan respon kontroler lebih lambat jika mendapatkan gangguan.

2.2.4. Term Integral

Peranan dari term integral berbanding lurus dengan besar dan lamanya error. Integral dalam kontroler PID adalah jumlahan error setiap waktu dan mengakumulasi offset yang sebelumnya telah dikoreksi. Error terakumulasi dikalikan dengan gain integral (Ki) dan menjadi keluaran kontroler. Term integral dirumuskan dengan:

$$Iout = Ki \int e(t)\, dt\ t\ 0$$



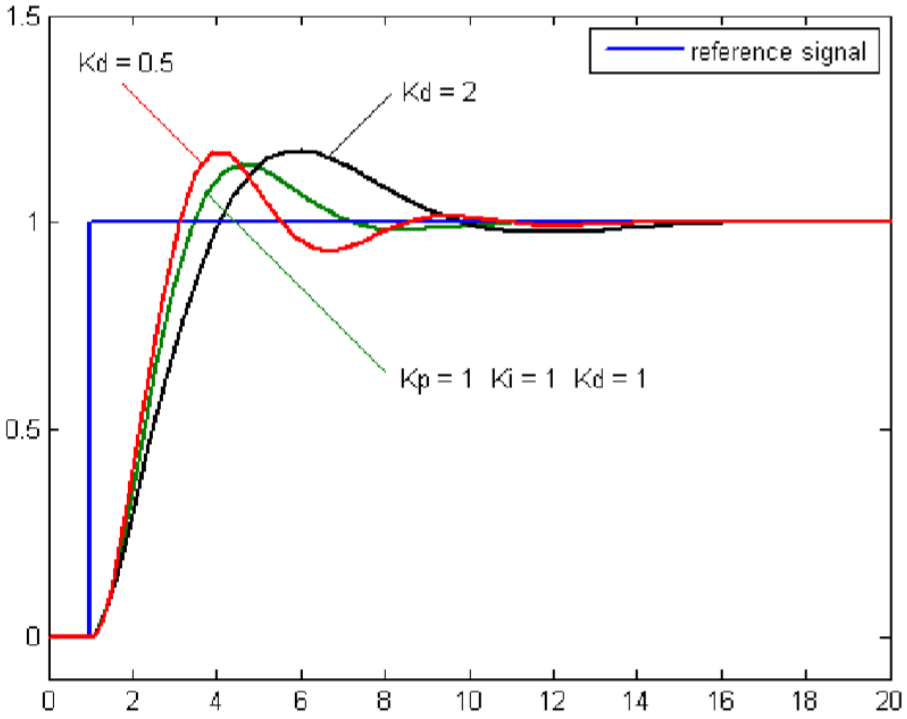
Gambar 1.3 Plot Y vs waktu, untuk 3 nilai Ki (Kp dan Kd dijaga konstan)

Grafik 1.3 menunjukkan bagaimana variasi nilai Ki berpengaruh terhadap time response dari suatu sistem. Term integral mempercepat perpindahan proses menuju setpoint dan menghilangkan steady-state error yang muncul pada kontroler proporsional. Namun, karena integral merespon terhadap error terakumulasi dari sebelumnya, maka dapat menyebabkan overshoot.

2.2.5. Term Derivatif

Turunan error pada proses dihitung dengan menentukan kemiringan error setiap waktu dan mengalikan perubahan tiap waktu dengan gain derivatif Kd. Term derivatif dirumuskan dengan:

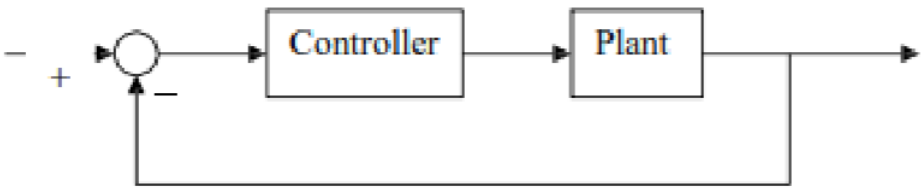
$$\mathrm{Dout} = \mathrm{Kd}\,\frac{\partial \mathrm{e(t)}}{\partial \mathrm{p}}$$



Gambar 1.4 Plot Y vs waktu, untuk 3 nilai Kd (Kp dan Ki dijaga konstan)

Grafik 1.4 menunjukkan bagaimana variasi nilai Kd berpengaruh terhadap time response dari suatu sistem. Aksi derivatif memprediksi perilaku sistem dan kemudian memperbaiki waktu tinggal dan stabilitas sistem. Aksi derivatif jarang digunakan pada industri. Dalam industri, diperkirakan hanya 25% kontroler menggunakan derivative karena akibatnya pada stabilitas sistem pada aplikasi dunia nyata.

2.2.6. Hubungan antara Kp, Ki, dan Kd terhadap Respon Waktu



Gambar 1.5 Blok diagram untuk unity feedback system

Ketiga subbab di atas menjelaskan bagaimana masing-masing term (Propotional, integral, derivative) berpengaruh terhadap suatu plant sistem. Sistem kontrol PID digunakan untuk menentukan presisi suatu sistem instrumentasi dengan karakteristik adanya umpan balik (feedback) pada sistem tersebut. Pengkombinasikan dilakukan dengan mengatur masing-masing kontrol (selanjutnya disebut sebagai konstanta) untuk melengkapi kekurangan yang dimiliki tiap-tiap kontrol sehingga didapat output yang dikehendaki.

Tabel 1.1 Respon PID Controller Terhadap Perubahan Konstanta

Closed Loop Response	*Rise Time*	*Overshoot*	*Settling Time*	*Steady State Error (%)*
Kp	*Decrease*	*Increase*	*Small change*	*Decrease*
Ki	*Decrease*	*Increase*	*Increase*	*Eliminate*
Kd	*Small change*	*decrease*	*decrease*	*Small change*

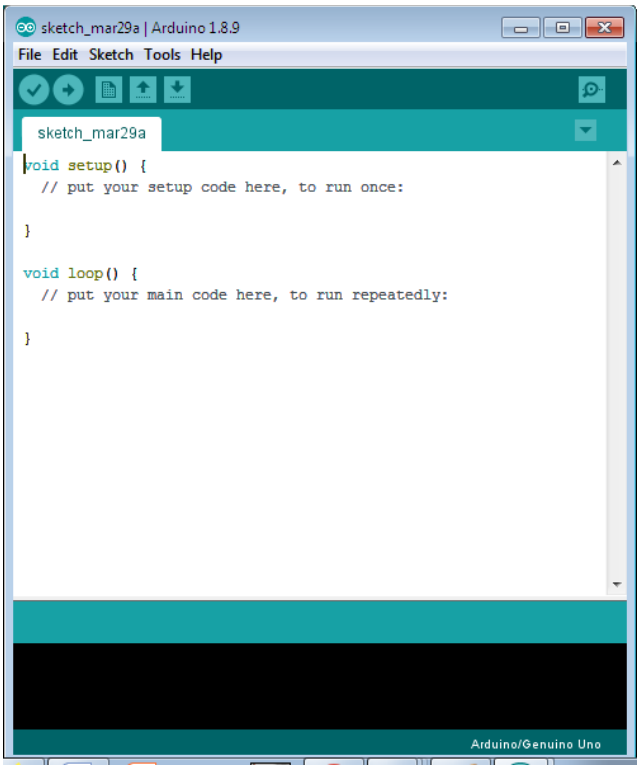
1.3 Arduino

Arduino adalah pengendali mikro single-boardyang bersifat sumber terbuka, diturunkan dari wiring platform, dirancang untuk memudahkan penggunaan elektronik dalam berbagai bidang.

Perangkat kerasnya memiliki prosesor Atmel AVR dan softwarenya memiliki bahasa pemrograman sendiri.Arduino menggunakan keluarga mikrokontroler ATMega yang dirilis oleh Atmel sebagai basis, namun terdapat jugaindividu/perusahaan yang membuat clonearduino dengan menggunakan mikrokontroler lain dan tetap kompatibel dengan arduino pada level hardware.

Untuk fleksibilitas, program dimasukkan melalui bootloadermeskipun terdapatopsi untuk mem-bypassbootloader dan menggunakan pengunduh untuk memprogram mikrokontroler secara langsung melalui port ISP.

Arduino juga merupakan senarai perangkat keras terbuka yang ditujukan kepada siapa saja yang ingin membuat purwarupa peralatan elektronik interaktif berdasarkan hardwaredan softwareyang fleksibel dan mudah digunakan. Mikrokontrolerdiprogram menggunakan bahasa pemrograman arduino yang memiliki kemiripan syntaxdengan bahasa pemrograman C. Karena sifatnya yang terbuka maka siapa saja dapat mengunduh skema hardwarearduino dan membangunnya.



Gambar 1.6 Tampilan Program Antarmuka Arduino

1.3.1. Arduino Uno

Terdapat beberapa jenis Arduino dengan kemampuan dan fiturnya masing-masing. Salah satu jenis perangkat Arduinoyang sering digunakan adalah Arduino UNO. Arduino Uno sangat cocok digunakan untuk para pemula yang ingin mempelajai microcontroller.

Gambar 1.7 Arduino Uno

Gambar 1.7 merupakan gambar dari salah satu jenis perangkat Arduino yaitu Arduino Uno REV3.Versi Arduino UNO yang terakhir adalah Arduino Uno R3 (Revisi 3), menggunakan ATMEGA328 sebagai microcontroller-nya, memiliki 14 pin I/O digital dan 6 pin input analog. Pemrograman dilakukan dengankoneksi USB tipe A ketipe B,sama seperti yang digunakan pada USB printer. Berikut ini merupakan spesifikasi arduino uno

Spesifikasi	Keterangan
Tegangan operasi	5V
Tegangan input	7V – 12V
Tegangan input (*limit, via jack DC*)	6V – 20V
Digital I/O	14 buah, 6 diantaranya menyediakan PWM
Analog input pin	6 buah
Arus DC per pin I/O	20 mA
Arus DC pin 3,3 V	50 mA
Memori *flash*	32 kB dan 0,5 kB telah digunakan untuk *bootloader*
SRAM	2 kB
EEPROM	1 kB
Clock speed	16 MHz
Dimensi	68,6 mm x 53,4 mm
Berat	25 gr

Gambar 1.8 Spesifikasi Arduino Uno

1.4 Pemanas

Pemanas yang digunakan pada project ini yaitu kawat nikelin dengan spesifikasi sebagai berikut :

Tabel 2.2 Spesifiaksi Pemanas

Jenis	Nikelin Bulat
Ukuran	0.45 mm
Panjang	2 meter
Resistansi	10.6 Ohm

Sehingga dari spesifikasi tersebut diperoleh perhitungan arus adalah sebagai berikut:

V = I x R

24 = I x 10.6

I = 2.2 A

P = V x I

= 24 x 2.2

= 52.8 Watt

BAB II
METODE PENYELESAIAN

2.1 Tahapan Perancangan Project

Tahapan perancangan sistem pada project kali ini adalah sebagai berikut:

1. Perancangan Perangkat Keras

   Perancangan perangkat keras dilakukan dengan melakukan perancangan sistem kerja purwarupa, permodelan sistem menggunakan software Matlab, perencanaan alat dan bahan, serta perancangan diagram kawat purwarupa.

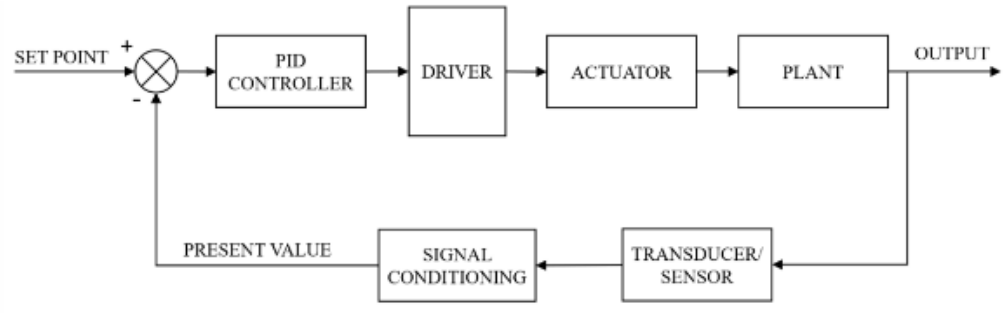
2. Perancangan Perangkat Lunak

   Perancangan perangkat lunak dilakukan dengan melakukan pemrograman perangkat controller.

3. Implementasi dan Pengujian

   Tahap Implementasi dan pengujian dilakukan dengan menyiapkan semua alat dan bahan sesuai dengan perencanaan kemudian dirangkai berdasarkan diagram kawat yang telah dirancang disertai program komputer dari controller yang telah dibuat. Kemudian dilakukan pengujian sistem secara keseluruhan.

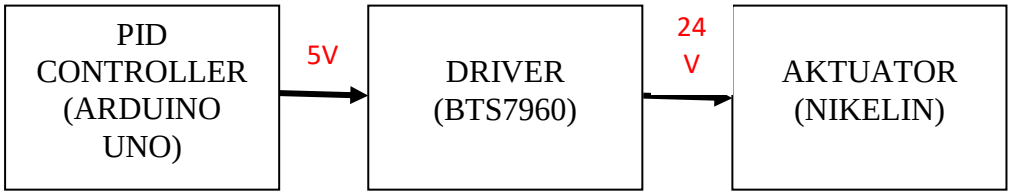
2.2 Perancangan Sistem Kerja PID



Gambar 2.1 Blog diagram sistem perancangan

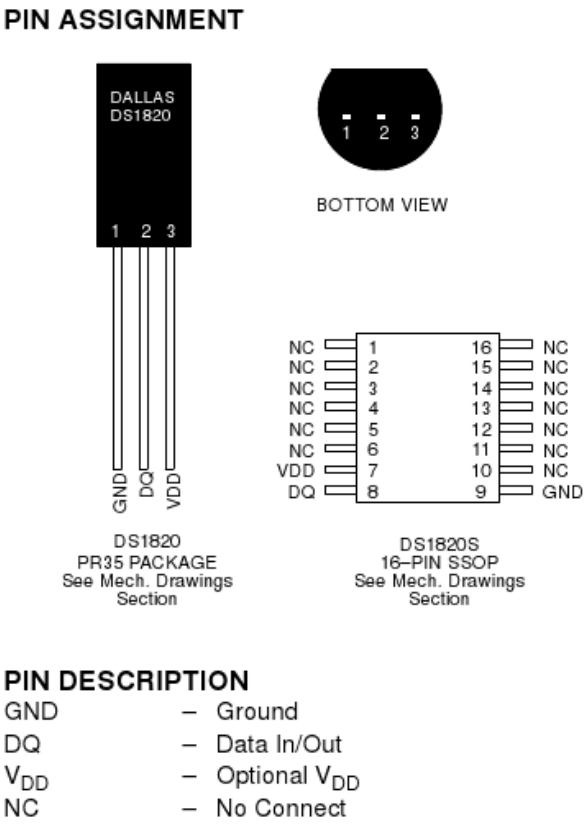
Blok diagram pada Gambar 2.1 menjelaskan bagaimana sistem kontrol berbasis PID bekerja pada purwarupa yang dibuat. Berikut penjelasannya :

1. Plant merupakan sistem target yang ingin kita kontrol.
2. Actuator merupakan perangkat yang merubah fenomena elektris menjadi fenomena fisik. Aktuator yang digunakan adalah kawat nikelin bertegangan 12 Volt DC dengan daya sebesar 52.8 Watt.
3. Sensor adalah perangkat yang merubah fenomena fisik menjadi fenomena elektris. Sensor yang digunakan adalah sensor DS1820.
4. PID controller menerima, mengelola, dan memberikan sinyal elektris sesuai dengan keinginan pengguna dengan mekanisme PID (Proportional-IntegralDerivative). Controller yang digunakan adalah Arduino Uno ATmega328P.
5. Driver menjadi penguat tegangan antara PID controller dan aktuator. Diperlukan driver dengan tegangan operasi, response time, dan arus listrik yang sesuai dengan spesifikasi controller dan actuator yang digunakan. Arduino Uno ATmega328P memiliki rentang tegangan luaran (voltage output) sebesar 0 Volt hingga 5 volt dengan arus DC (Direct Current). Pemanas air yang digunakan memiliki rentang tegangan masukan (voltage input) sebesar 0 Volt hingga 24 volt DC. Maka tegangan operasi dari driver yang diperlukan memiliki control input level beroperasi pada 5V dan operating voltage pada 24 Volt. Secara skema digambarkan di gambar 2.2.



Gambar 2.2 Operating Voltage untuk Driver

6. Signal conditioning menerima sinyal dari sensor dan mengonversinya ke dalam bentuk sinyal yang dapat diterima dan dikelola controller. Sinyal yang dihasilkan sensor DS1820 dapat dengan langsung diterima dan dikelola oleh controller Arduino Uno ATmega328P sesuai dengan gambar 2.3.



Gambar 2.3 Pinout dari DS1820

Maka dari itu, dalam pembuatan prototype ini, perangkat tambahan signal conditioning tidak diperlukan.

7. Set point/desired output/input merupakan nilai output yang diinginkan. Set point dari sistem kontrol pada percobaan ini adalah bernilai 40 ^{o}C
8. Output adalah luaran aktual dari jalannya keseluruhan sistem kontrol. Output dari sistem kontrol ini adalah nilai aktual dari temperatur akhir kawat nikelin.
9. Present value merupakan nilai output yang terukur sensor sebagai informasi timbal balik (feedback) untuk diproses controller dan dalam percobaan ini merupakan nilai temperatur akhir kawat nikelin yang dihasilkan dari pengukuran sensor.

2.3 Alat dan Bahan

2.3.1 Komponen dan Hardware

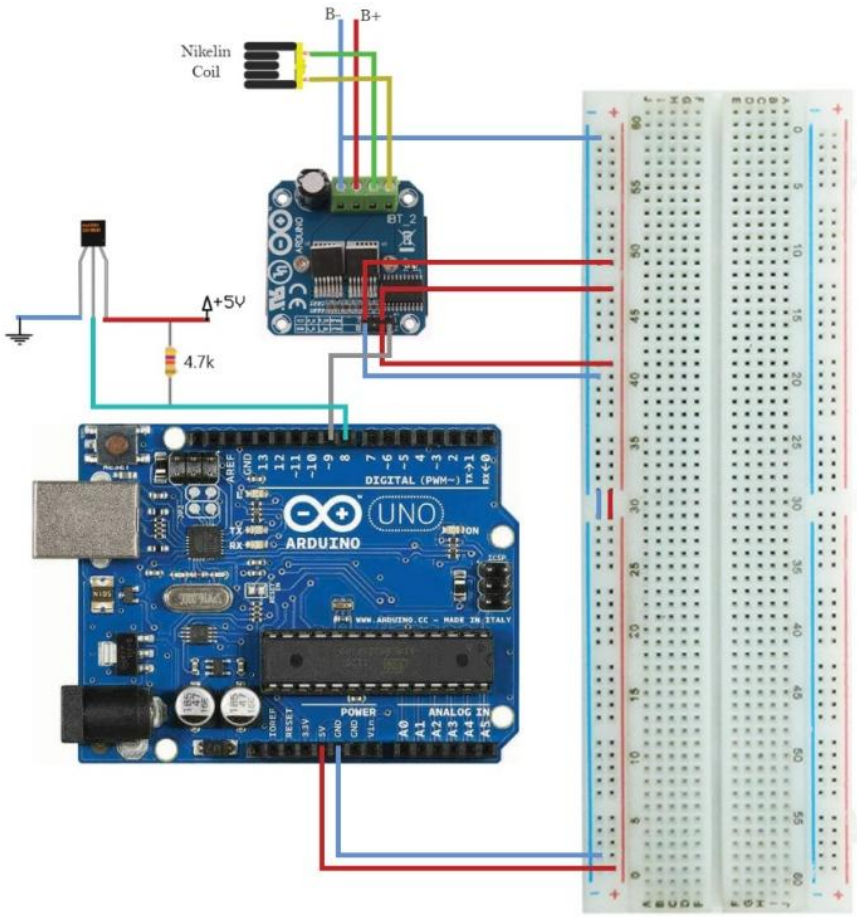
Komponen-komponen yang digunakan dalam pembuatan project ini diuraikan dalam tabel 2.2 di bawah ini.

Tabel 2.1 Daftar Komponen dari Pembuatan Project PID pada Pemanas Nikelin

No	Komponen atau Hardware	Jumlah
1	Power Supply Unit 24V 10A DC	1
2	Arduino Uno	1
3	Driver BTS7960	1
4	Sensor Suhu DS1820	1
5	Resistor 4.7KOhm	1
6	Kawat Nikelin 2m	1
7	Project Board	1
8	Kabel Jumper female to female	5
9	Kabel Jumper Male to Male	20
10	Kabel Steker 2 kaki	1
11	Kabel USB Arduino	1

2.3.1 Integrasi Hardware

Wiring keseluruhan hardware dapat dilihat pada gambar 2.4 dibawah ini.



Gambar 2.4 Integrasi hardware

BAB III
PENGUJIAN DA ANALISIS DATA

Dalam bab ini dibahas tentang pengujian berdasarkan perencanaan dari sistem yang dibuat. Pengujian dilakukan untuk mengetahui bagaimana kerja dari sistem dan untuk mengetahui apakah perangkat sudah sesuai dengan perencanaan atau belum. Pengambilan data pengujian dilakukan secara terpisah pada masing-masing unit rangkaian meskipun seluruh sistem dijalankan secara bersama.

3.1. Pengujian Sensor Suhu

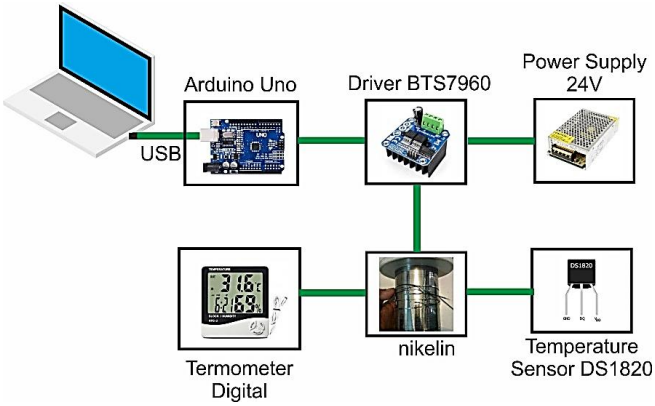
1. Tujuan

   Pengujian pembacaan sensor suhu DS1820 untuk mengetahui tingkat presisi sensor DS1820 dengan membandingkan sensor DS1820 dengan termometer digital yang akurat.

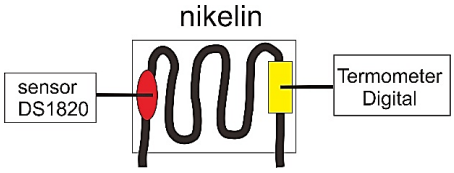
2. Peralatan dan perlengkapan

a. Arduino Uno 1 unit
b. PC 1 unit
c. Sensor suhu DS1820 1 unit
d. Termometer digital 1 unit
e. Kabel jumper secukupnya
f. Kabel USB 1 unit
g. *Project board* 1 unit
h. Perangkat lunak Arduino IDE

3. *Setting* pengujian



Gambar 3. 1 *Setting* Pengujian Sensor Suhu DS1820



Gambar 3. 2 Pengkabelan Pengujian Sensor Suhu DS1820

4. **Prosedur pengujian**

a. Menyiapkan peralatan yang digunakan.
b. Menyalakan PC.
c. Menghubungan Arduino Uno dengan PC menggunakan kabel USB
d. Memasang sensor DS1820 dan termometer digital pada pemanas (nikelin)
e. Membuka *software* Arduino IDE
f. Membuat program (terlampir) pada *software* Arduino IDE
g. Meng-*compile* program yang telah dibuat dan *upload* program ke Arduino Uno
h. Buka *serial monitor* untuk menampilkan data yang terbaca dengan cara pilih *tools* lalu klik *serial monitor*.
i. Membandingkan hasil pembacaan sensor dengan termometer digital

5. Hasil dan Analisa

Tabel 3.1 Tabel Pengujian Sensor Suhu DS1820

No.	Suhu sensor	Suhu termometer	Error (%)	Error Rata-Rata (%)
1	29.00	29.00	0	0.10
2	29.00	29.00	0	
3	29.00	29.00	0	
4	29.00	29.00	0	
5	29.00	29.00	0	
6	29.00	29.00	0	
7	29.00	29.00	0	
8	29.06	29.00	0,2	
9	29.06	29.00	0,2	
10	29.06	29.00	0,2	
11	29.06	29.00	0,2	
12	29.06	29.00	0,2	
13	29.06	29.00	0,2	
14	29.06	29.00	0,2	
15	29.06	29.00	0,2	

Dari hasil pengujian diketahui bahwa selisih (*Error*) antara pembacaan sensor suhu DS1820 dengan termometer digital didapatkan dari rumus berikut:

$$Error\ (\%) = \frac{Sensor\ \ DS1820 - Termometer\ \ \ Digital}{Termometer\ \ \ Digital}\ x\ 100\ \% \ldots\ldots\ldots \quad 3.1$$

3.2 Pengujian Karateristik Driver Motor BTS7960 43A

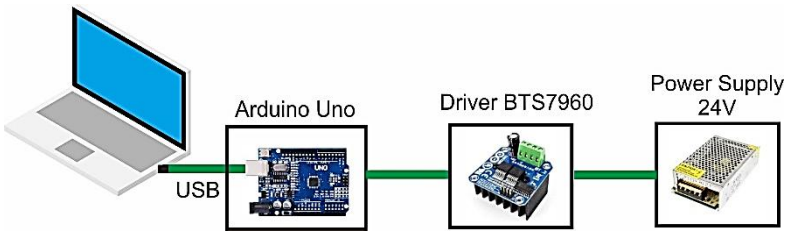
1. Tujuan

Mengetahui kinerja dan respon rangkaian driver motor BTS7960 43A dengan membandingkan output tegangan efektif driver dengan masukan *duty cycle* sinyal PWM yang diberikan oleh Arduino Uno.

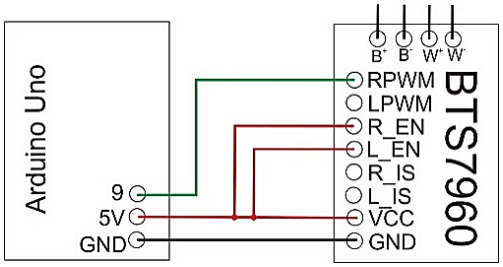
2. Peralatan dan perlengkapan

a. Arduino Uno 1 unit
b. PC 1 unit
c. *Multimeter digital* 1 unit
d. *Driver* BTS7960 1 unit
e. Power supply 1 unit
f. *Project board* 1 unit
g. Kabel *power* 1 unit
h. Kabel USB 1 unit
i. Kabel jumper secukupnya
j. *Software* Arduino IDE

3. *Setting* Pengujian



Gambar 3.3 *Setting* Pengujian Karakteristik *Driver* BTS7960



Gambar 3.4 Pengkabelan Arduino Uno dengan *Driver* BTS7960

4. Prosedur Pengujian

a. Menyiapkan semua alat yang dibutuhkan.

b. Melakukan pengkabelan seperti pada Gambar 4.11 Pengkabelan Arduino dengan *Driver* BTS7960
c. Menyambungkan B^+ *driver* pada 24V *power supply* dan B^- *driver* pada GND *power supply*
d. Memasang stop kontak *power supply* pada 220V
e. Membuka *software* Arduino IDE
f. Menuliskan program pada Arduino IDE (dilampirkan)
g. Meng-*compile* program yang telah dibuat dan *upload* program ke Arduino Uno
h. Mengubah-ubah nilai PWM pada program
i. Mengukur output driver pada pin W^- dan W+
j. Mencatat hasil pengukuran

5. Hasil dan Analisa

Data hasil pengujian driver BTS7960 ditunjukkan pada Tabel 4.2 Tabel Hasil Pengujian Driver BTS7960

Tabel 3.2 Tabel Pengujian Driver BTS7960

No	*Duty Cycle* (%)	PWM	Praktik	Teori	*Error* (%)	*Error* Rata-Rata (%)
			V Out (V)	V Out (V)		
1	0	0	0	0	0	0.39
2	20	51	4,77	4,8	0,62	
3	40	102	9,54	9,6	0,62	
4	60	153	14,33	14,4	0,49	
5	80	204	19,08	19,2	0,62	
6	100	255	24	24	0	

Semakin bertambah nilai masukan *duty cycle* sinyal PWM maka nilai V Out praktik semakin bertambah pula. Jadi dapat disimpulkan bahwa nilai *duty*

cycle berbanding lurus dengan Vout yang terukur. Selain itu nilai Vout praktikum dan teori tidak jauh berbeda dapat dilihat pada Tabel 4.2 Tabel Hasil Pengujian Driver BTS7960 didapatkan hasil *error* dengan rumus berikut:

$$Error\ (\%) = \frac{\text{Vout praktik} - \text{Vout teori}}{\text{Vout praktik}} \text{ x } 100\% \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 3.2$$

Sehingga nilai rata-rata dari ke 6 data *error* tersebut adalah 0.39 % jadi proses drive tegangan ke pemanas dapat bekerja dengan baik.

3.3 Pengujian Karateristik Driver Motor BTS7960 43A dengan Beban Kawat Nikelin

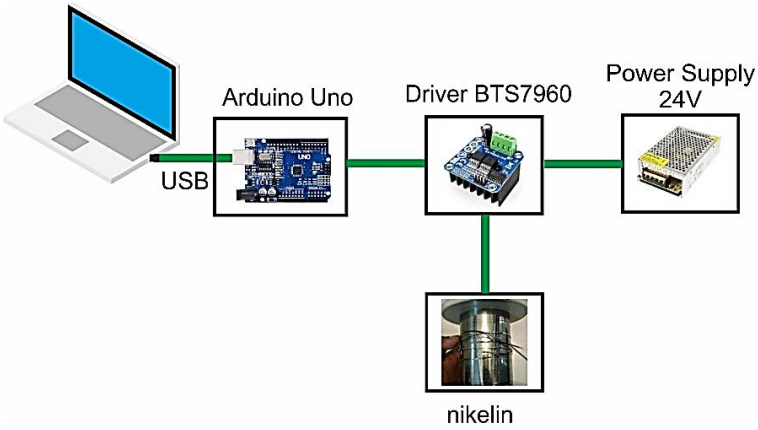
1. Tujuan

Mengetahui range atau karakteristik dari kawat nikelin dengan input PWM yang berbeda-beda

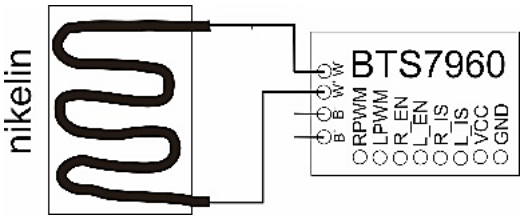
2. Peralatan dan perlengkapan

a. Arduino Uno	1 unit
b. PC	1 unit
c. *Multimeter digital*	1 unit
d. *Driver* BTS7960	1 unit
e. Kawat nikelin 3 meter	3 unit
f. Power supply 24V	1 unit
g. *Project board*	1 unit
h. Kabel *power*	1 unit
i. Kabel USB	1 unit
j. Kabel jumper	secukupnya
k. Software Arduino IDE	

3. *Setting* pengujian



Gambar 3.5 *Setting* Karateristik Driver Motor BTS7960 dengan Beban Kawat Nikelin



Gambar 3.6 Pengkabelan *driver* BTS7960 dengan kawat nikelin

4. Prosedur pengujian

a. Menyiapkan semua alat yang dibutuhkan.
b. Melakukan pengkabelan seperti pada Gambar 4.14 Pengkabelan Kawat Nikelin dengan *Driver* BTS7960
c. Menyambungkan B^+ *driver* pada 24V *power supply* dan B^- *driver* pada GND *power supply*
d. Menyambungkan beban kawat nikelin pada pin *driver* W^- dan W+
e. Memasang stop kontak *power supply* pada 220V
f. Membuka *software* Arduino IDE
g. Menuliskan program pada Arduino IDE (dilampirkan)
h. Meng-*compile* program yang telah dibuat dan *upload* program ke Arduino Uno
i. Mengubah-ubah nilai PWM pada program
j. Mengukur output driver pada pin W^- dan W+
k. Mencatat hasil pengukuran

5. Hasil dan Analisa

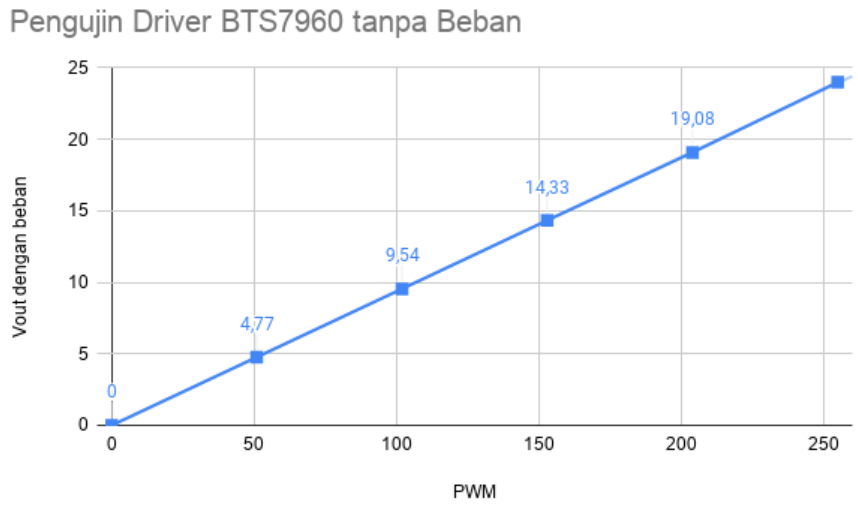
Berikut merupakan tabel hasil dari pengujian driver BTS7960 dengan beban kawat nikelin.

Tabel 3.3 Tabel pengujian driver BTS7960 dengan beban kawat nikelin

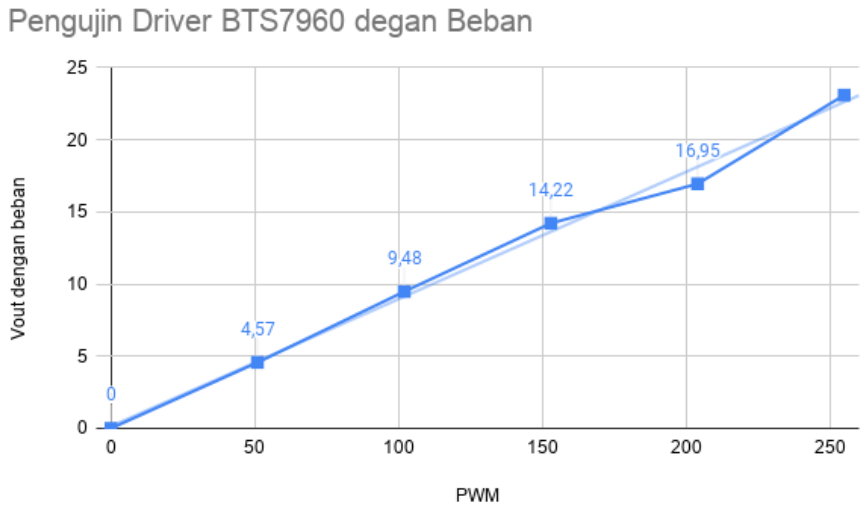
No	Duty Cycle (%)	PWM	Dengan Beban	Tanpa Beban
			V Out (V)	V Out (V)
1	0	0	0	0
2	20	51	4,57	4,77
3	40	102	9,48	9,54
4	60	153	14,22	14,33
5	80	204	16,95	19,08
6	100	255	23,10	24

Dari hasil pengujian, nilai Vout dengan beban makin bertambah seiring dengan kenaikan PWM. Selain itu nilai Vout tanpa beban dengan beban selisihnya tidak terlalu jauh dapat dilihat pada Tabel 3.3 Tabel pengujian driver BTS7960 dengan beban kawat nikelin, sehingga dapat dipastikan performa driver ini dapat bekerja dengan baik.

Berdasarkan tabel diatas dapat divisualisasikan dengan menggunakan grafik sebagai berikut.



Gambar 3.7 Grafik pengujian driver bts7960 tanpa beban (hubungan antara PWM dan Vout)



Gambar 3.8 Grafik Pengujian Driver BTS7960 Dengan Beban (Hubungan Antara PWM Dan Vout)

3.4 Pengujian Karakteristik Kawat Nikelin

1. Tujuan

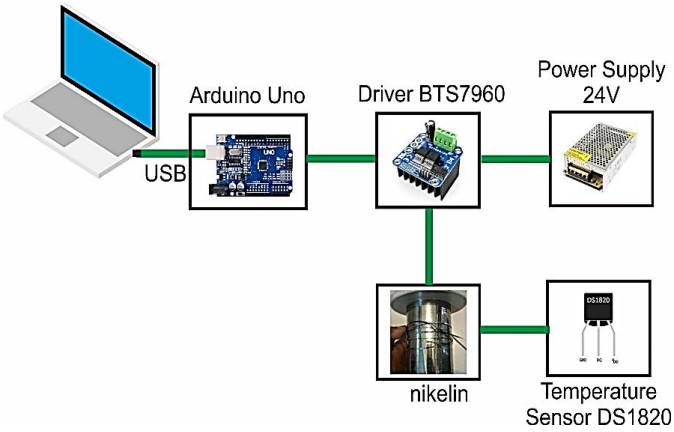
Mengetahui karakteristik kawat nikelin, dengan memberikan nilai input

berupa dutycycle sebesar 0%-100% menjadi output berupa suhu panas kawat nikelin.

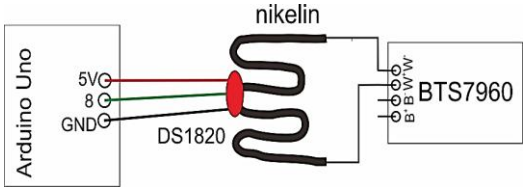
2. Peralatan dan perlengkapan

a.	Arduino Uno	1 unit
b.	PC	1 unit
c.	Sensor suhu DS1820	1 unit
d.	*Driver* BTS7960	1 unit
e.	Kawat nikelin 3 meter	3 unit
f.	Power supply 24V	1 unit
g.	*Project board*	1 unit
h.	Kabel *power*	1 unit
i.	Kabel USB	1 unit
j.	Kabel jumper	secukupnya
k.	*Software* Arduino IDE	

3. *Setting* pengujian



Gambar 3.9 *Setting* Pengujian Karakteristik Kawat Nikelin



Gambar 3.10 Pengkabelan Kawat Nikelin, DS1820 dengan Driver BTS7960

4. Prosedur pengujian

a. Menyiapkan semua alat yang dibutuhkan.
b. Memasang sensor suhu pada nikelin yang sudah dilapisi dengan isolator asbes yang digunakan dengan tape listrik
c. Melakukan pengkabelan seperti pada Gambar 4.17 Pengkabelan Kawat Nikelin, DS1820 dengan *Driver* BTS7960
d. Menyambungkan B^+ *driver* pada 24V *power supply* dan B^- *driver* pada GND *power supply*
e. Menyambungkan beban kawat nikelin pada pin *driver* W^- dan W+
f. Memasang stop kontak *power supply* pada 220V
g. Membuka *software* Arduino IDE
h. Menuliskan program pada Arduino IDE (dilampirkan)
i. Meng-*compile* program yang telah dibuat dan *upload* program ke Arduino Uno
j. Mengubah-ubah nilai PWM pada program
k. Mencatat hasil pembacaan suhu pada nikelin oleh sensor suhu DS1820

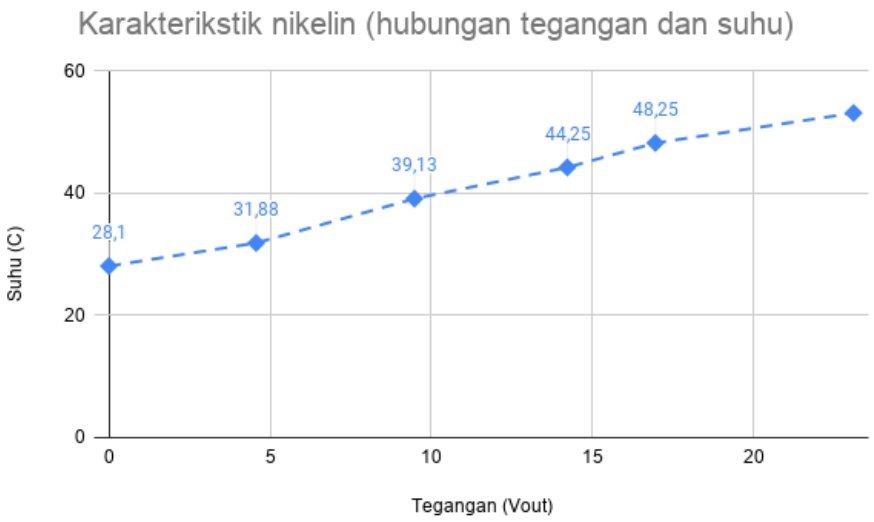
5. Hasil dan Analisa

Berikut merupakan tabel hasil pengujian karakteristik nikelin.

Tabel 3.4 Tabel pengujian karakteristik kawat nikelin

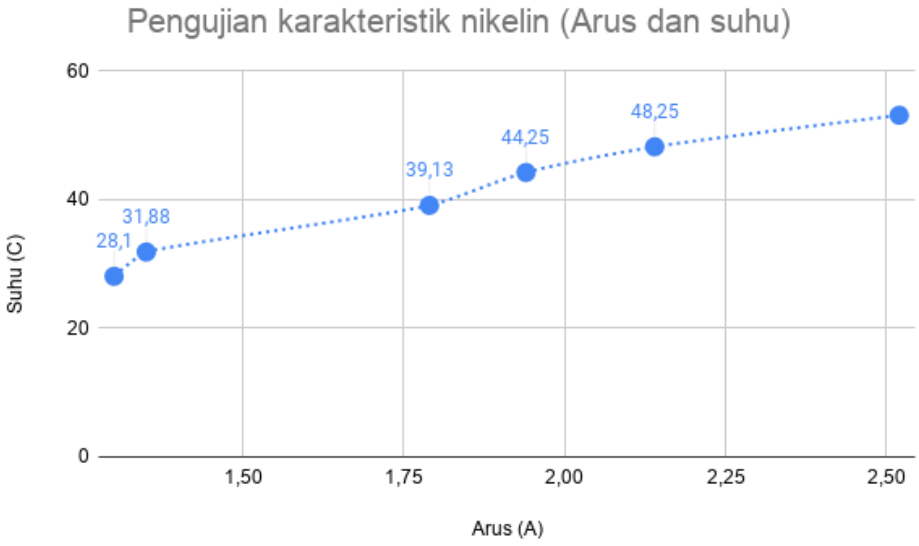
No	Duty Cycle (%)	PWM	Teori	Praktik	Teori	Praktik	Suhu (^oC)
			V Out (V)	V Out (V)	Arus (A)	Arus (A)	
1	0	0	0	0	0	1,3	28,1
2	20	51	4,8	4,57	0,45	1,35	31,88
3	40	102	9,6	9,48	0,90	1,79	39,13
4	60	153	14,4	14,22	1,35	1,94	44,25
5	80	204	19,2	16,95	1,81	2,14	48,25
6	100	255	24	23,10	2,26	2,52	53,15

Berdasarkan data diatas, maka dapat divisualisasikan dengan menggunakan grafik sebagai berikut:



Gambar 3.11 Hasil pengujian tegangan terhadap suhu

Data diatas menceritakan bahwa hubungan antara arus dengan suhu adalah linear. Apabila tegangan dinaikkan, maka suhu akan meningkat. Hal demikian juga berlaku untuk hubungan antara suhu dan arus, yang digambarkan pada gambar dibawah ini.



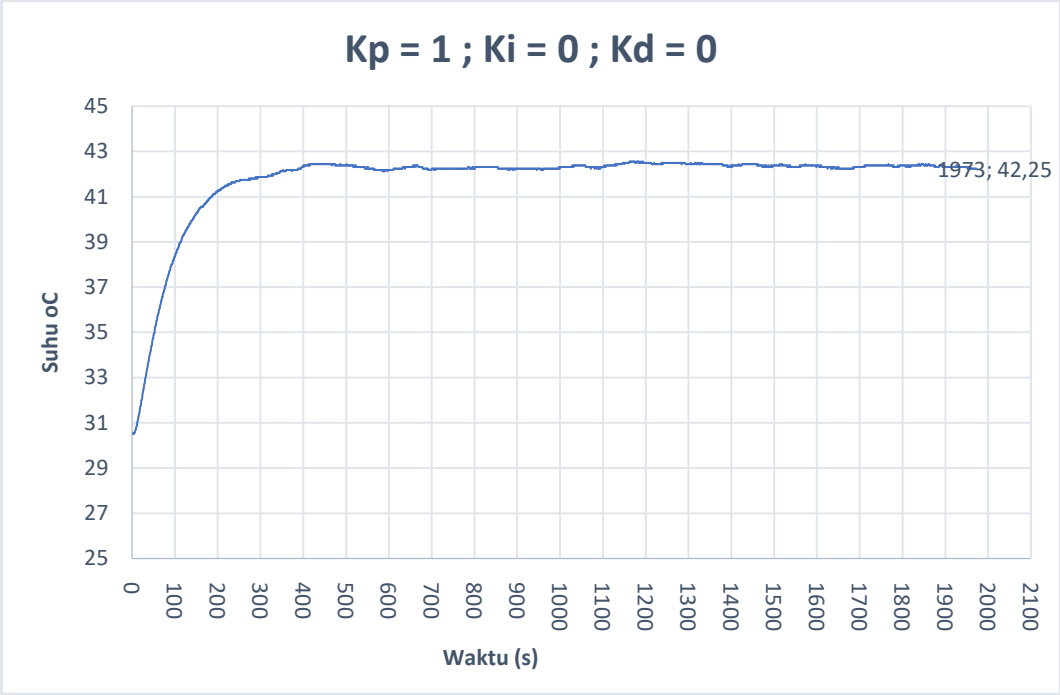
Gambar 3.11 Hasil pengujian arus tterhadap suhu

3.5 Pengujian Sistem Kontrol Proporsional (Kp)

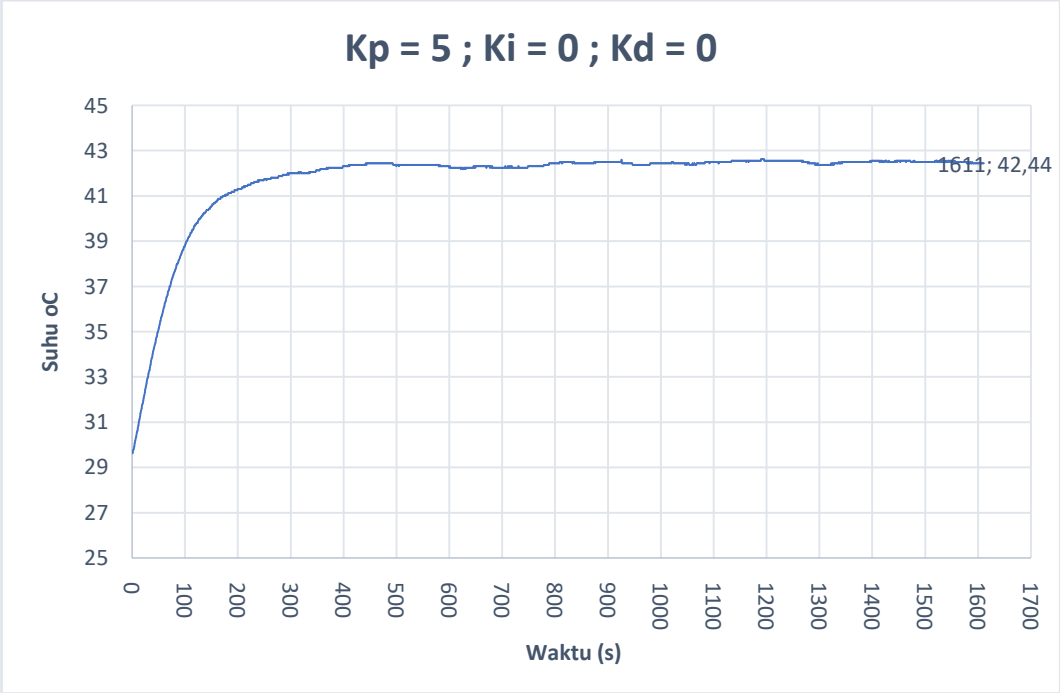
Di dalam percobaan dengan sistem kontrol proporsional, nilai Ki dan Kd adalah nol dan dilakukan variasi nilai Kp untuk melihat bagaimana pengaruh sistem kontrol proporsional terhadap plant. Berikut hasil percobaan dari sistem kontrol P pada tabel 3.5 :

Tabel 3.5. Data Hasil Percobaan Sistem Kontrol Proporsional

Set Point = 40 ^{o}C								
No	Kp	Delay Time (s)	Rise Time (s)	Overshoot (^{o}C)	Settling Time (s)	Peak Time (s)	Steady State (^{o}C)	Steady State Error (%)
1	1	67	400	42,44	700	418	42,38	5,95
2	5	63	370	42,63	800	1200	42,44	6,10
3	10	67	260	44,44	320	1008	44,00	10,00
4	15	63	280	44,81	300	1463	44,31	11,00
5	30	82	480	45,31	800	758	44,75	12,00
6	50	85	420	45,38	1500	1575	44,38	11,00

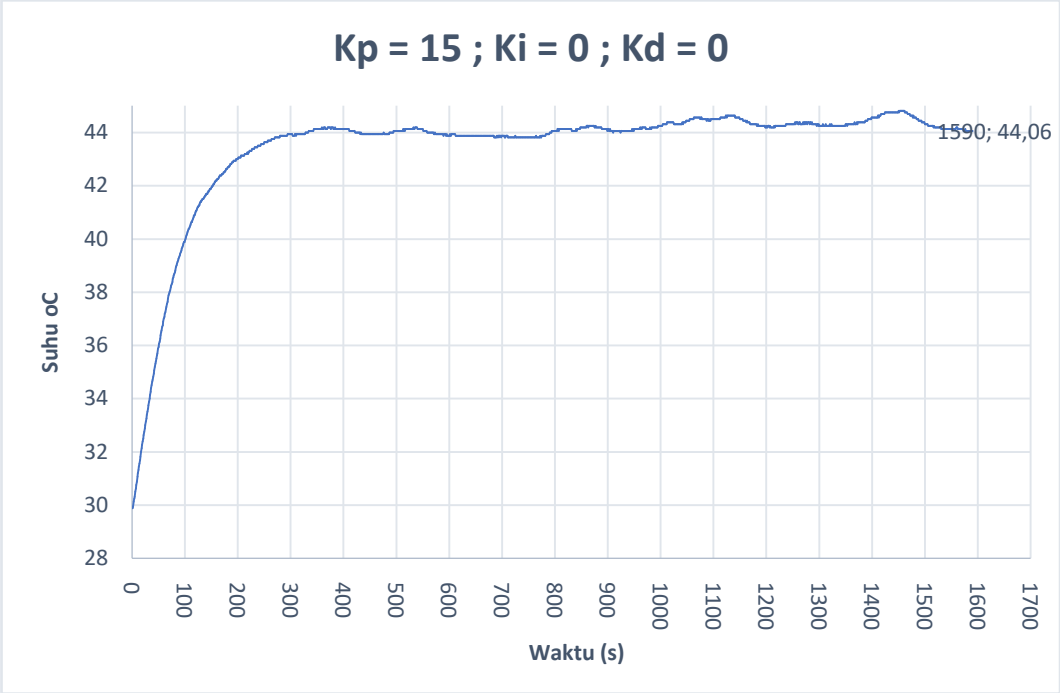


Gambar 3.12 Respon Waktu Kp=1; Ki=0; Kd=0

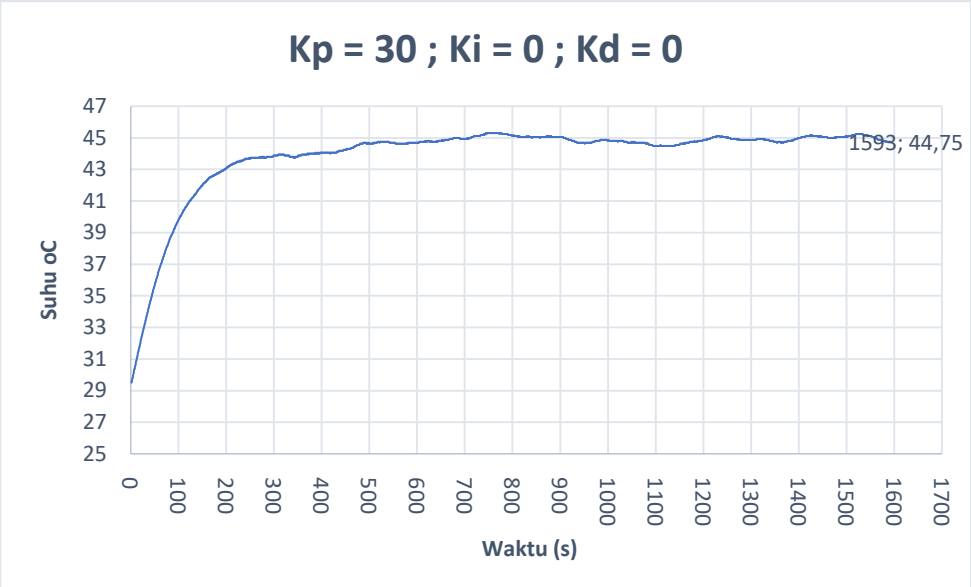


Gambar 3.13 Respon Waktu Kp=5; Ki=0; Kd=0

Gambar 3.14 Respon Waktu Kp=10; Ki=0; Kd=0

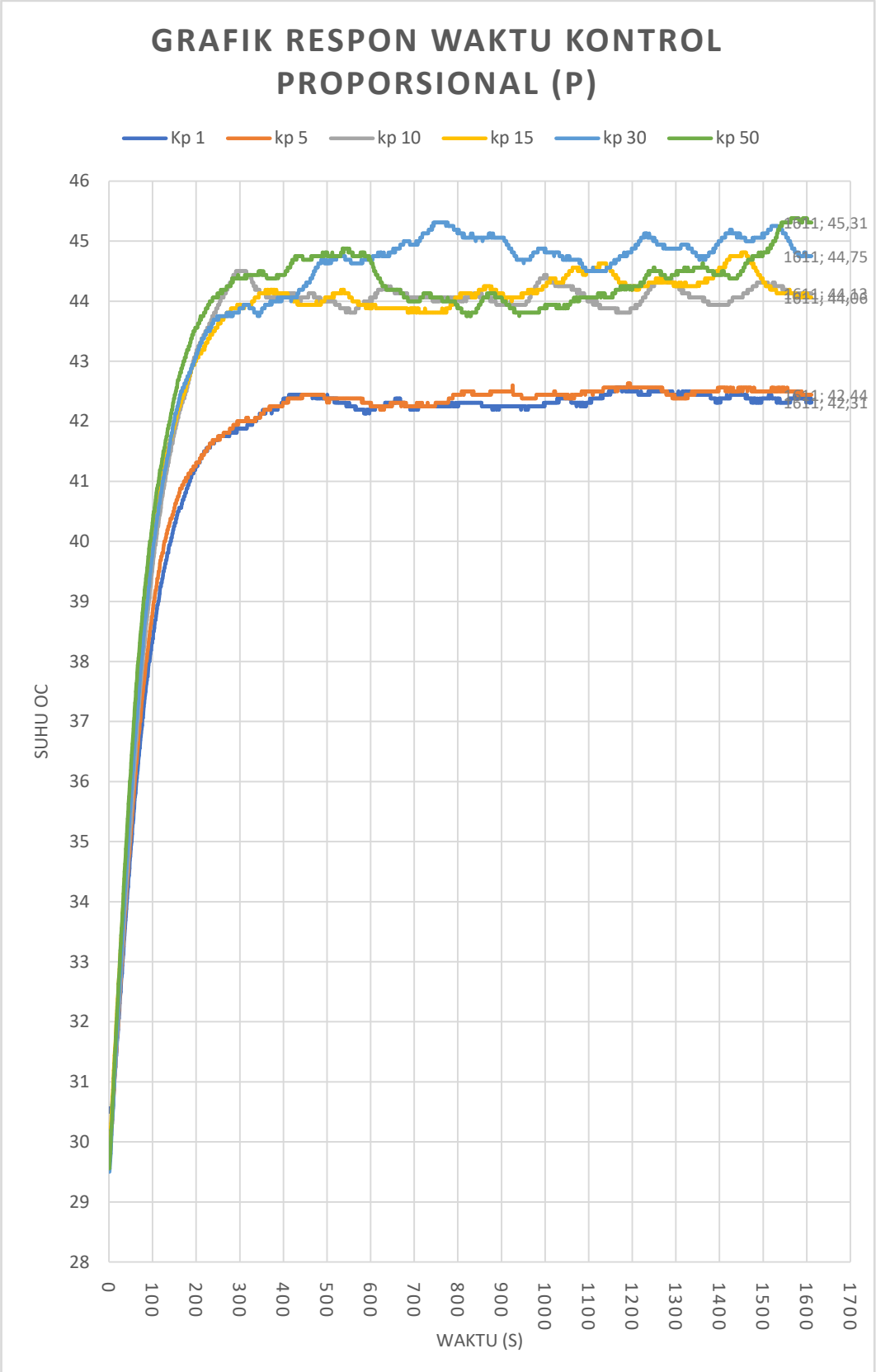


Gambar 3.15 Respon Waktu Kp=15; Ki=0; Kd=0



Gambar 3.16 Respon Waktu Kp=30; Ki=0; Kd=0

Gambar 3.17 Respon Waktu Kp=50; Ki=0; Kd=0



Gambar 3.18 Grafik Respon Waktu Kontrol Proporsional (P)

ANALISA

Dari tabel 3.5 dapat disimpulkan bahwa peningkatan nilai gain Kp berakibat pada meningkatnya *overshoot* dapat dilihat pada gambar 3.18. Kestabilan sistem semakin sulit dicapai seiring bertambahnya gain Kp. Sulit bagi sistem dengan kontrol proporsional untuk mampu mencapai *steady-state error* yang kecil. Bertambahnya Gain Kp hanya mampu menambah nilai steady-state error dengan perubahan yang cukup besar.

3.6 Pengujian Sistem Kontrol Integral (Ki)

Dengan hanya sistem kontrol proporsional, sulit bagi sistem untuk dapat mencapai *steady-state error* yang kecil mendekati nol. Maka dari itu ditambahkan sistem kontrol integral. Secara teori, sistem kontrol integral memberikan pengaruh besar dalam membawa steady-state error ke nilai nol.

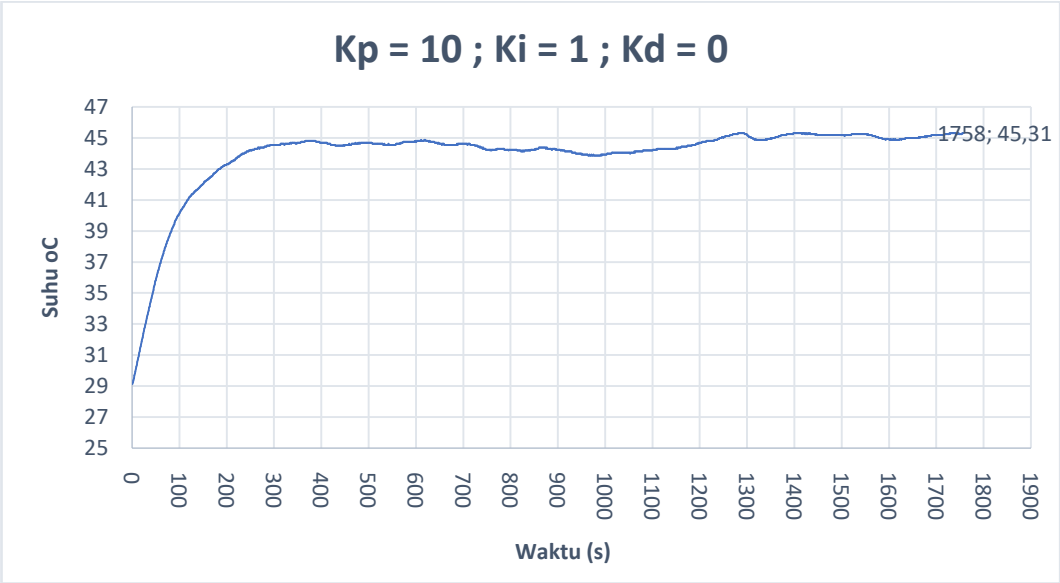
Sistem kontrol integral berpengaruh pada dipercepatnya rise time. Oleh karena itu dengan nilai Kp sebesar 10 sistem kontrol PI dilakukan kemudian dicari nilai Ki yang mampu membawa steady-state error ke nilai yang paling dekat dengan nilai nol dan dianalisa juga pengaruh lainnya dari variasi nilai Ki terhadap respon waktu dari sistem.

Di dalam percobaan dengan sistem kontrol proporsional-integral, nilai Kp adalah konstan sebesar 10, Kd adalah nol dan dilakukan variasi nilai Ki untuk melihat bagaimana pengaruh penambahan sistem kontrol integral terhadap plant. Berikut hasil percobaan dari sistem kontrol PI pada tabel 3.6.

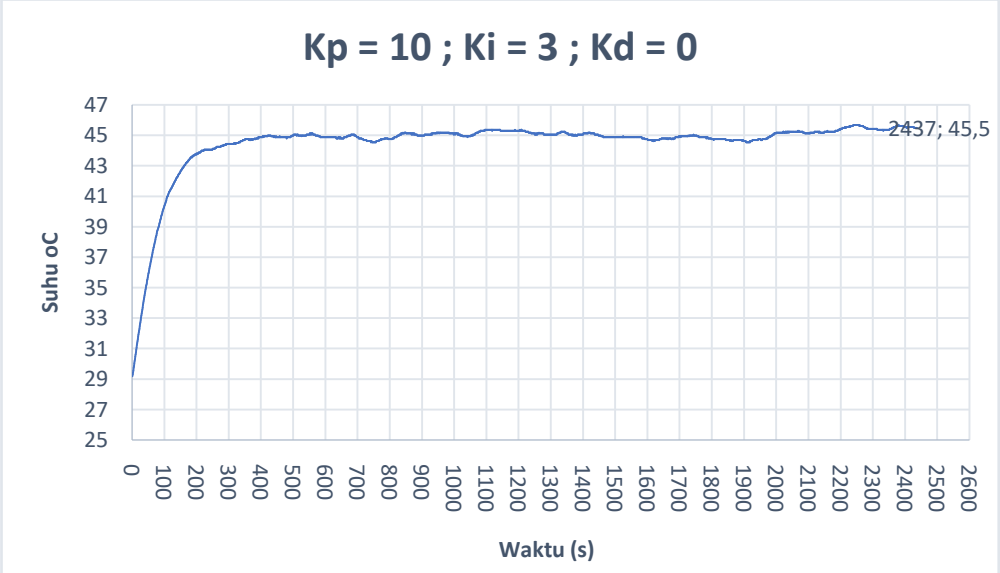
Tabel 3.6 Data Hasil Percobaan Sistem Kontrol Proporsional-Integral

Set Point = 40 ^{o}C ; Kp= 10								
No	Ki	Delay Time (s)	Rise Time (s)	Overshoot (^{o}C)	Settling Time (s)	Peak Time (s)	Steady State (^{o}C)	Steady State Error (%)
1	1	82	280	45,31	400	1729	45,19	12,90
2	3	86	225	45,69	700	2249	44,88	12,20

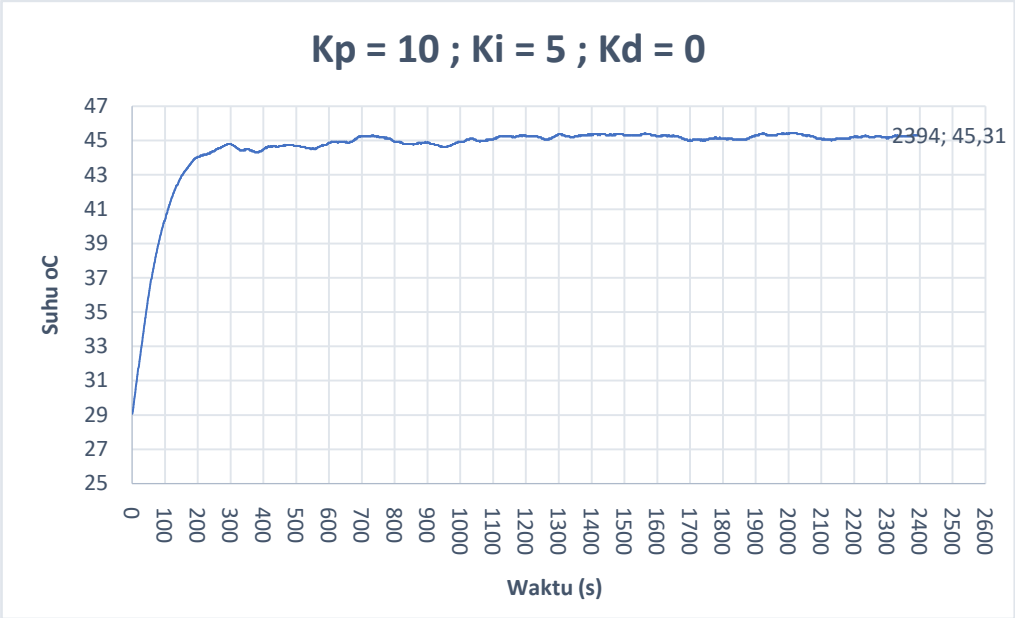
3	5	86	220	45,44	300	1930	45,25	13,10
4	30	88	210	45,69	600	1158	45,13	12,80



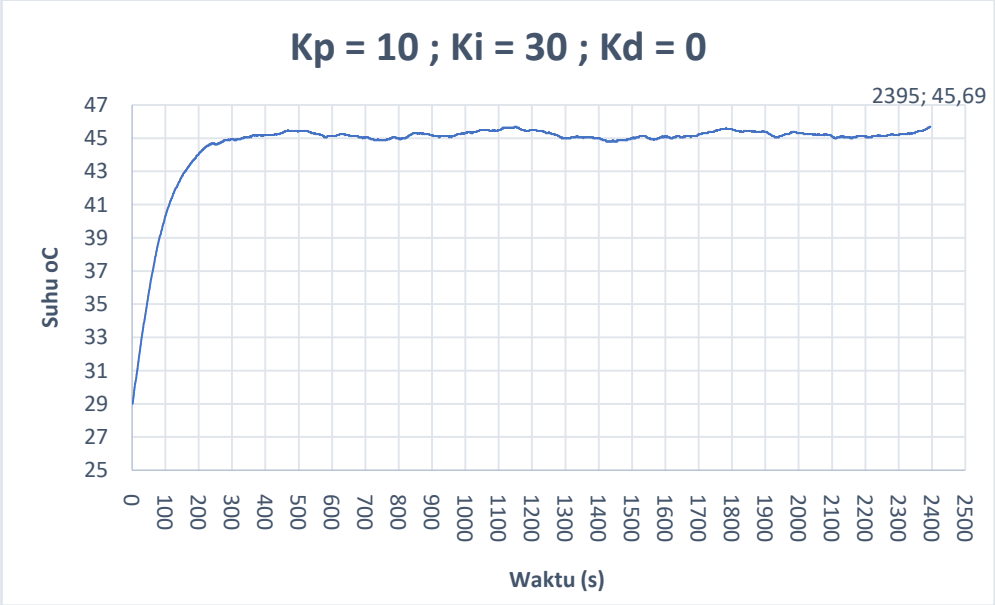
Gambar 3.19 Respon Waktu Kp=10; Ki=1; Kd=0



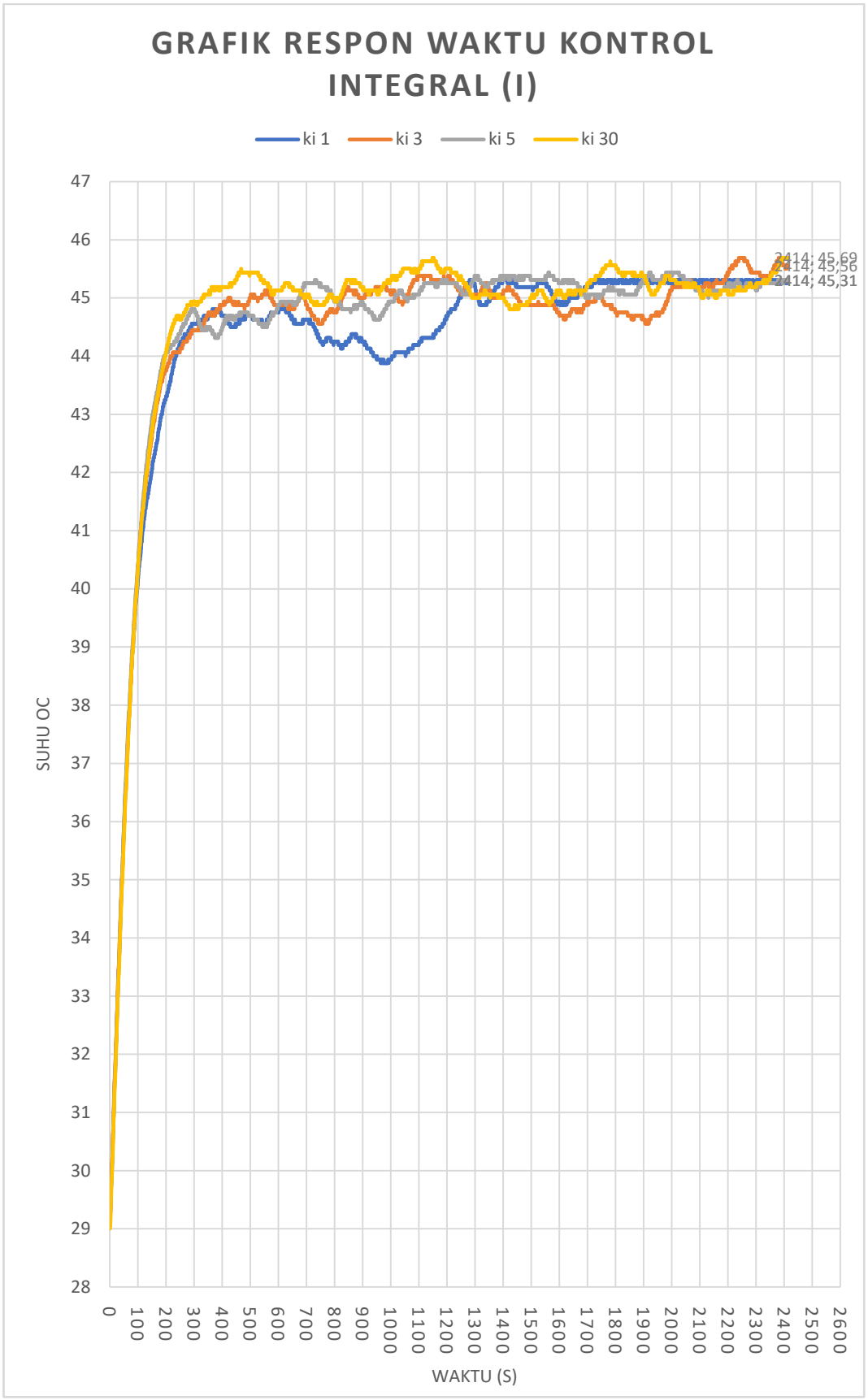
Gambar 3.20 Respon Waktu Kp=10; Ki=3; Kd=0



Gambar 3.21 Respon Waktu Kp=10; Ki=5; Kd=0



Gambar 3.22 Respon Waktu Kp=20; Ki=30; Kd=0



Gambar 3.23 Grafik Respon Waktu Kontrol Integral (I)

ANALISA

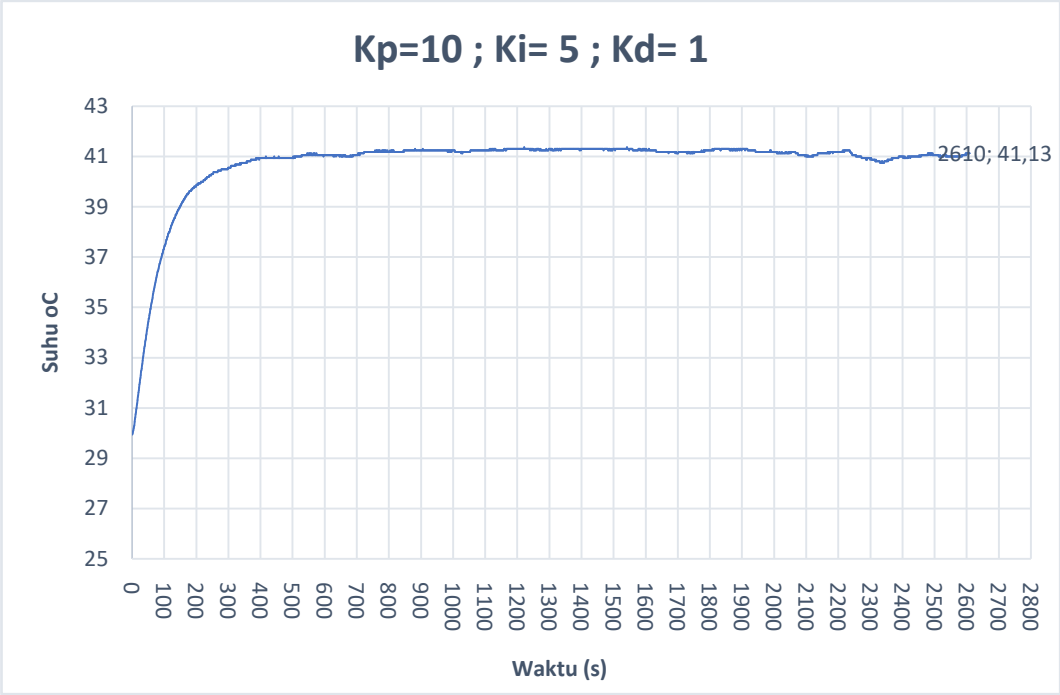
Pada tabel 3.5 dapat disimpulkan bahwa peningkatan nilai gain Ki berakibat pada dipercepatnya rise time, meningkatnya overshoot, dan meningkatkan steady-state error. Penambahan nilai Ki berakibat pada ketidakstabilan sistem. Dapat dilihat dari grafik 3.23 adanya osilasi dengan simpangan yang juga cukup besar.

3.7 Pengujian Sistem Kontrol PID

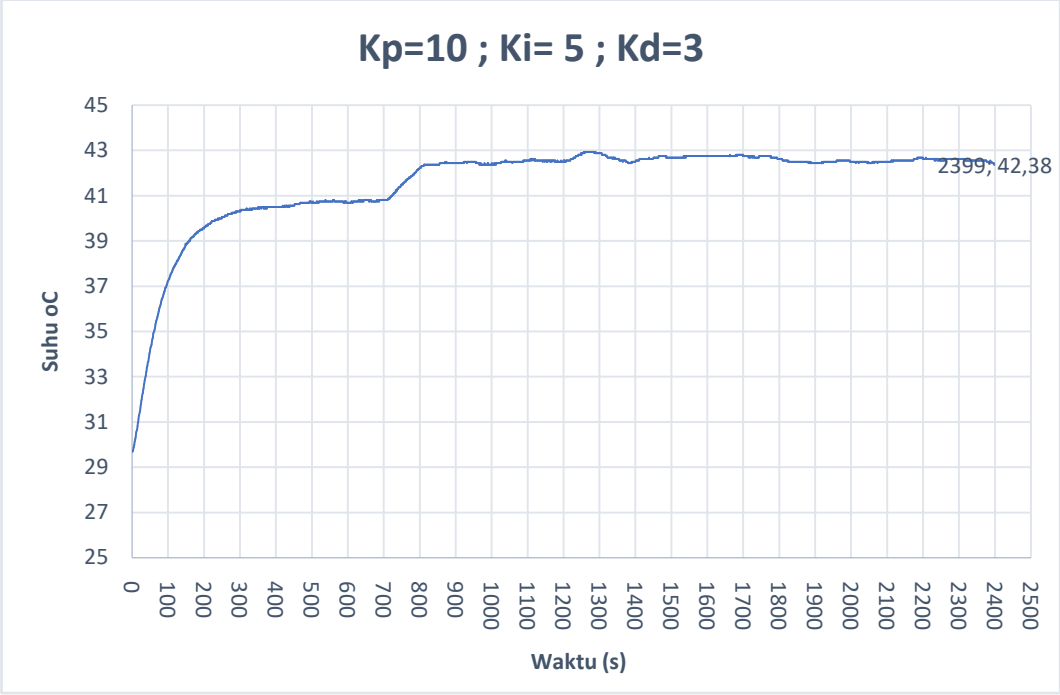
Sistem kontrol derivatif memperkecil settling time. Selain itu, secara teori, penambahan gain Kd juga berpengaruh terhadap diperbaikinya kestabilan respon sistem. Osilasi kurva yang timbul seiring besarnya nilai Kp dan ki dapat dikurangi dengan penambahan gain Kd. Di dalam percobaan dengan sistem kontrol proporsional-integral-derivative, nilai Kp adalah konstan sebesar 10, nilai Ki adalah konstan sebesar 5, dan dilakukan variasi nilai Kd untuk melihat bagaimana pengaruh penambahan sistem kontrol derivative terhadap plant. Berikut hasil percobaan dari sistem kontrol PID pada tabel 3.7.

Tabel 3.7 Data Hasil Percobaan Sistem Kontrol PID

Set Point = 40 ^{o}C ; Kp=10 ; Ki=5								
No	Kd	Delay Time (s)	Rise Time (s)	Overshoot (^{o}C)	Settling Time (s)	Peak Time (s)	Steady State (^{o}C)	Steady State Error (%)
1	1	66	400	41,38	757	1552	41,25	3,10
2	3	102	800	42,94	1037	1270	42,50	6,25



Gambar 3.24 Respon Waktu Kp=10; Ki=5; Kd=1



Gambar 3.25 Respon Waktu Kp=10; Ki=5; Kd=3

ANALISA

Dari tabel 3.7 disimpulkan bahwa peningkatan nilai gain Kd berakibat pada diperlambatnya settling time dan berkurangnya kestabilan respon sistem yang berarti juga dapat menambah overshoot. Dapat dilihat dari grafik 3.24 dan grafik 3.25, dengan penambahan Kd, settling time bertambah secara drastis. Selain itu, penambahan nilai Kd berakibat kecil pada diperlambatnya rise time. Dapat dilihat dari grafik 3.24 dan grafik 3.25, dengan penambahan Kd, rise time naik dengan nilai yang kecil. Penambahan gain Kd berakibat pada berubahnya nilai steady-state error. Dapat dilihat dari grafik 3.24 dan grafik 3.25.

BAB IV
KESIMPULAN

Kesimpulan dari percobaan ini adalah :

1. PID controller diterapkan pada pemanas kawat nikelin dengan cara membangun rancang sistem kerja yang benar. Ditentukan aktuator, sensor, perangkat controller, driver, dan perangkat signal conditioning yang tepat. Dalam percobaan ini, aktuator yang digunakan adalah kawat nikelin dengan panjang 2 meter 2.2 A dengan daya sebesar 52.8 Watt, sensor yang digunakan adalah sensor DS1820, perangkat controller yang digunakan adalah Arduino Atmega328P, driver yang digunakan adalah motor driver BTS7960, dan tanpa memerlukan perangkat signal conditioning.
2. Pada percobaan ini, jika diinginkan respon dengan rise time dan settling time tercepat serta kestabilan yang tinggi digunakan kontrol PID dengan Kp sebesar 10, Ki sebesar 5, dan Kd sebesar 3.
3. Bertambahnya nilai kontrol Kp dan Ki menyebabkan overshoot yang meningkat juga.
4. Bertambahnya nilai kontrol Kp menyebabkan kestabilan sistem sulit dicapai.

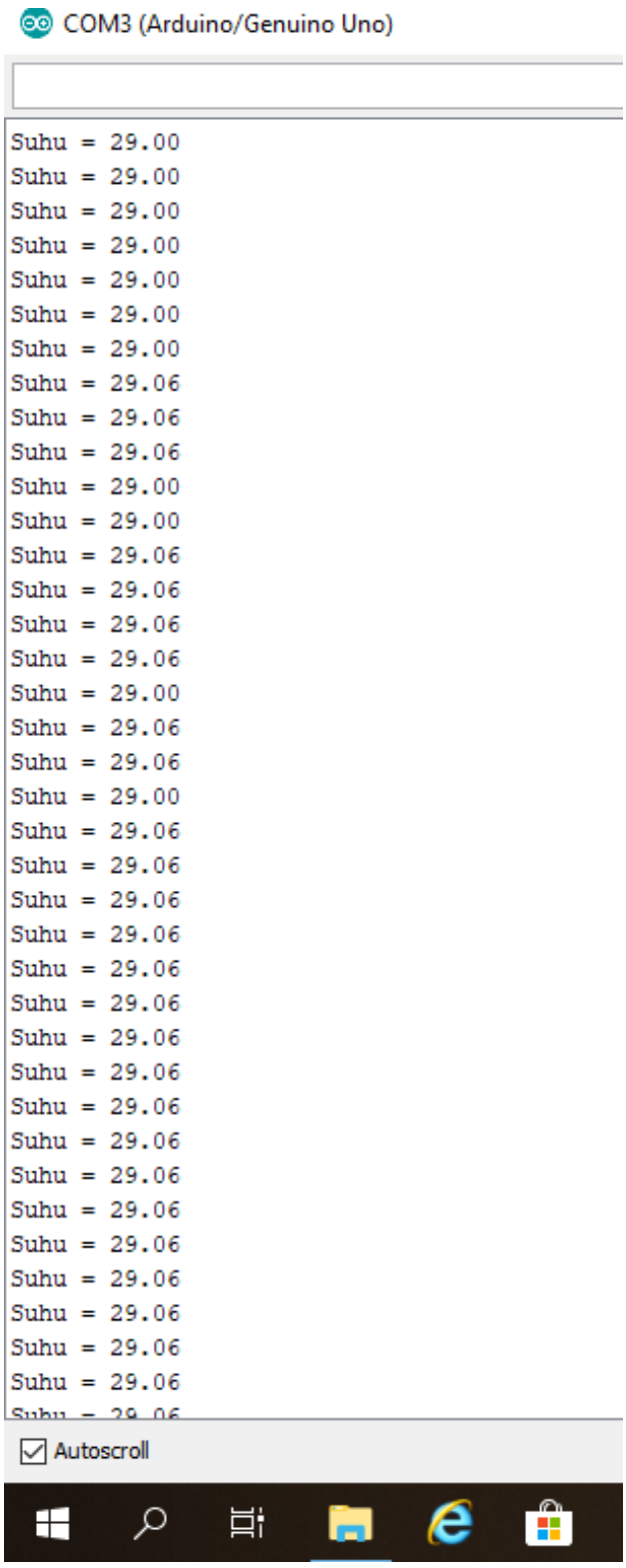
DAFTAR PUSTAKA

[1] Pambudi W. S et al., "*Aplikasi Sistem Kontrol Sudut Dengan Metode PID Pada Arm Flip Folding Machine Menggunakana Lego Mindstorm EV3*", Seminar Nasional Sains dan Teknologi Terapan V 2017 nstitut Teknologi Adhi Tama Surabaya, ISBN 978-602-98569-1-0, 2017.

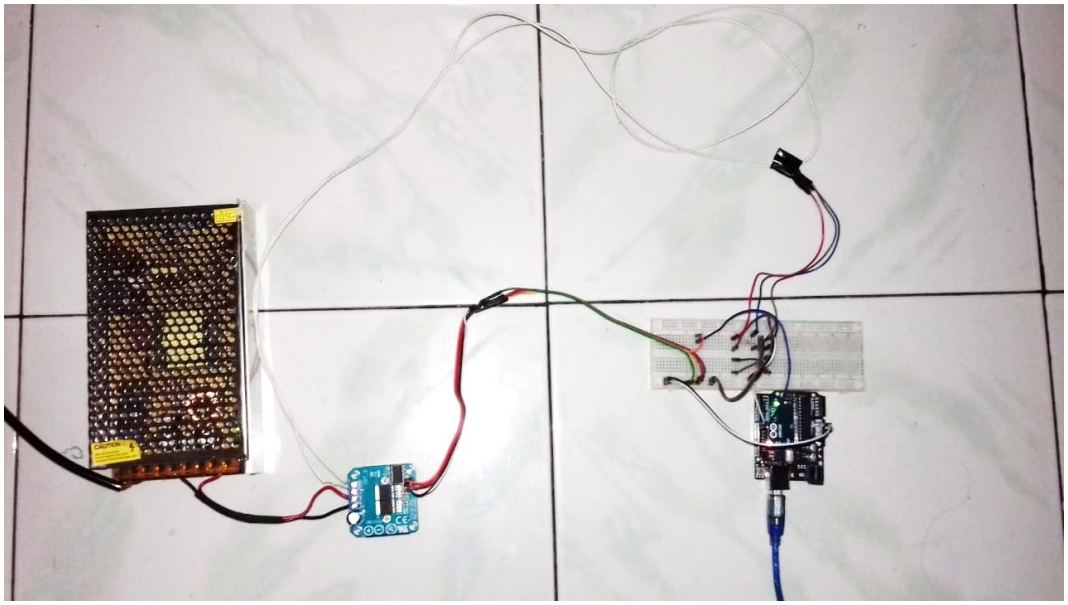
[2] Modul kuliah Sistem Kendali Modern, Wahyu S. Pambudi

[3] Sholahuddin, Abdurrafi, et all. "Pemanas Air Otomatis Berbasis PID Controller"., Laporan Proyek Mekatronika, Institut Teknologi Sepuluh Nopember, 2020.

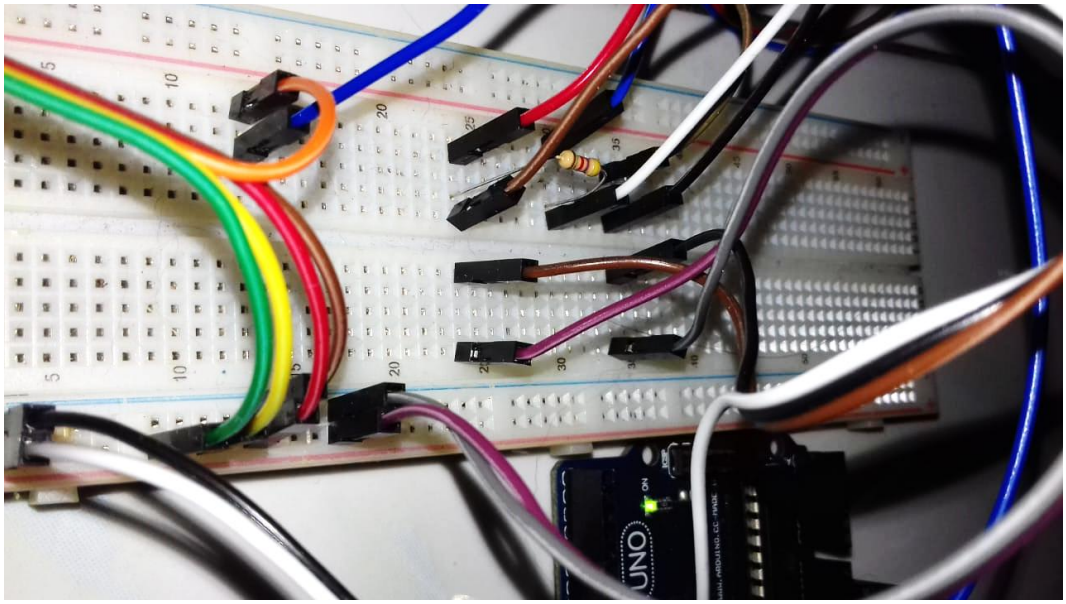
LAMPIRAN



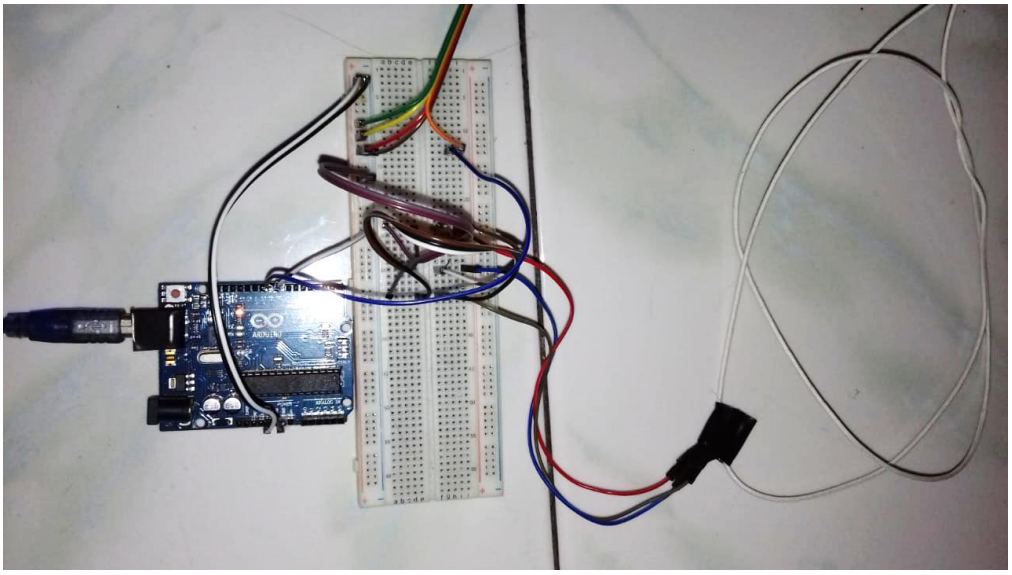
Gambar 1. Data suhu DS1820



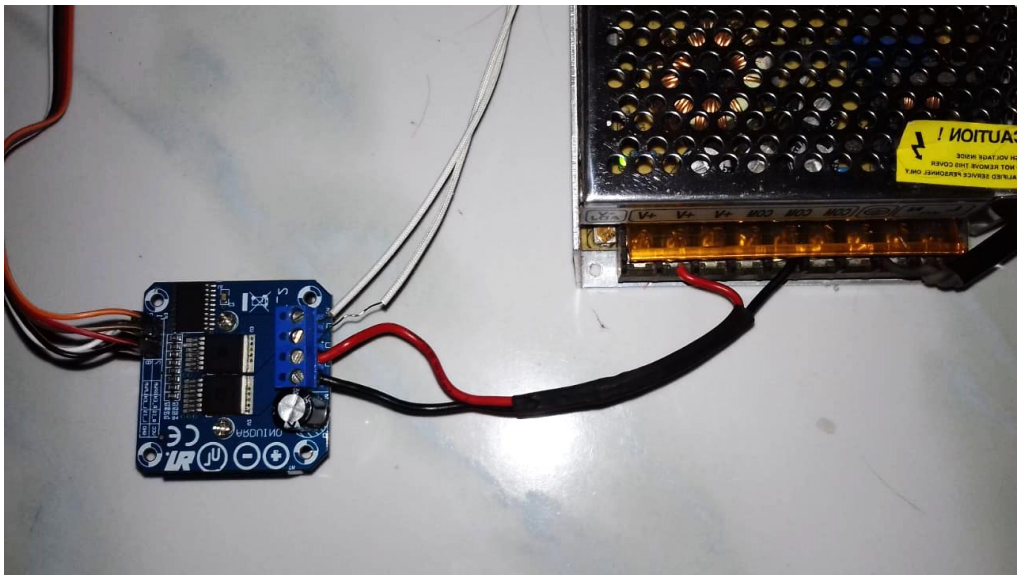
Gambar 2. Wiring Keseluruhan Sistem



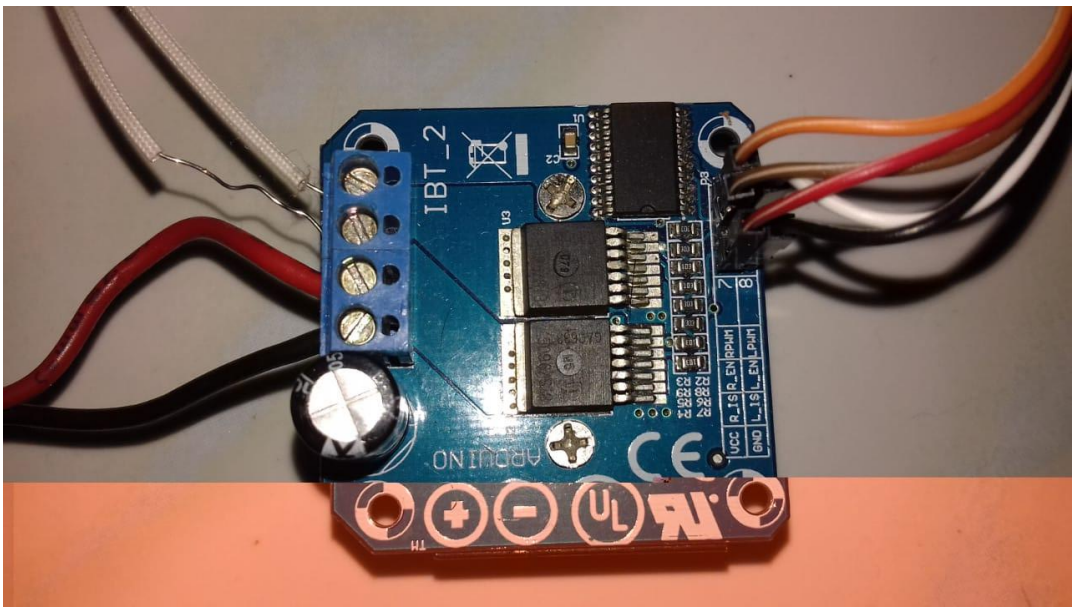
Gambar 3. Wiring pada Project Board



Gambar 4. Wiring Sensor Suhu



Gambar 5. Wiring Driver BTS7950 & Power Supply 24V 10A DC



Gambar 6. Driver BTS7960

PROGRAM ARDUINO PID

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <EEPROM.h>

#define pin_yes 12
#define pin_next 11
#define pin_up 10
#define pin_down 9

#define yes !digitalRead(pin_yes)
#define next !digitalRead(pin_next)
#define up !digitalRead(pin_up)
#define down !digitalRead(pin_down)



#define pin_pwm 9
#define ONE_WIRE_BUS 8

int kp = 10;
int ki = 5;
int kd = 1;
int ts = 2;
int sv_suhu=40;

LiquidCrystal_I2C lcd(0x27, 16, 2);
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
```

```
// note
// SV : Set Value (Set point) atau nilai acuan awal dari kondisi normal
// PV : Present value atau nilai sensor/ output saat ini

double pv_suhu=0;
double sv_pwm,pv_pwm;
double pid,error,last_error;




//A10B2C1D
void setup() {
  pinMode(pin_yes,INPUT_PULLUP);
  pinMode(pin_next,INPUT_PULLUP);
  pinMode(pin_up,INPUT_PULLUP);
  pinMode(pin_down,INPUT_PULLUP);

  lcd.begin();
  lcd.backlight();
  sensors.begin();
  pinMode(pin_pwm,OUTPUT);
  sv_pwm = 125; // posisi normal pwm adalah 90 derajat

  Serial.begin(9600);



  kp = EEPROM.read(0);
  ki = EEPROM.read(1);
  kd = EEPROM.read(2);
  sv_suhu=EEPROM.read(4);
```

```
  if (kp>=255){kp=0;}
  if (ki>=255){ki=0;}
  if (kd>=255){kd=0;}
  if (sv_suhu>=255){sv_suhu=0;}

  Serial.println("Nilai KP: " + String(kp));
  Serial.println("Nilai KI: " + String(ki));
  Serial.println("Nilai KD: " + String(kd));
  Serial.println("Set Point Suhu:"+String(sv_suhu));
  delay(2000);
}
void loop() {

  if (yes){
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Setting Prameter");
    lcd.setCursor(0,1);
    lcd.print("     PID        ");
    delay(2000);
    while(yes==true){}
    lcd.clear();
    //Setting parameter Sp Suhu
    bool sudah=false;
    sv_suhu=EEPROM.read(4);
    while(sudah==false){
    lcd.setCursor(0,0);
    lcd.print("Set Point Suhu:");
    lcd.setCursor(0,1);
    lcd.print("Sp:");
    lcd.print(sv_suhu);
    lcd.print("    ");
```

```
if (up){sv_suhu++;delay(200);}
else if (down){sv_suhu--;delay(200);}

if (sv_suhu<0){sv_suhu=0;}
else if (sv_suhu>255){sv_suhu=255;}

if (next){
  lcd.clear();
  lcd.print("DATA DISIMPAN");
  while(next==true){}
  delay(2000);
  EEPROM.write(4,sv_suhu);
  sudah=true;
}
}

lcd.clear();
//Setting parameter Sp Kp
sudah=false;
kp=EEPROM.read(0);
while(sudah==false){
lcd.setCursor(0,0);
lcd.print("Set Point Kp:");
lcd.setCursor(0,1);
lcd.print("Kp:");
lcd.print(kp);
lcd.print("    ");
if (up){kp++;delay(200);}
else if (down){kp--;delay(200);}

if (kp<0){kp=0;}
else if (kp>255){kp=255;}
```

```
if (next){
  lcd.clear();
  lcd.print("DATA DISIMPAN");
  while(next==true){}
  delay(2000);
  EEPROM.write(0,kp);
  sudah=true;
}
}

lcd.clear();
//Setting parameter Sp ki
sudah=false;
ki=EEPROM.read(1);
while(sudah==false){
  lcd.setCursor(0,0);
  lcd.print("Set Point Ki:");
  lcd.setCursor(0,1);
  lcd.print("Ki:");
  lcd.print(ki);
  lcd.print("    ");

  if (up){ki++;delay(200);}
  else if (down){ki--;delay(200);}

  if (ki<0){ki=0;}
  else if (ki>255){ki=255;}

  if (next){
    lcd.clear();
    lcd.print("DATA DISIMPAN");
```

```
    while(next==true){}
    delay(2000);
    EEPROM.write(1,ki);
    sudah=true;
  }
}
 lcd.clear();
//Setting parameter Sp kd
sudah=false;
kd=EEPROM.read(2);
while(sudah==false){
lcd.setCursor(0,0);
lcd.print("Set Point Kd:");
lcd.setCursor(0,1);
lcd.print("Kd:");
lcd.print(kd);
lcd.print("    ");
if (up){kd++;delay(200);}
else if (down){kd--;delay(200);}

if (kd<0){kd=0;}
else if (kd>255){kd=255;}

if (next){
  lcd.clear();
  lcd.print("DATA DISIMPAN");
  while(next==true){}
  delay(2000);
  EEPROM.write(2,kd);
  sudah=true;
}
}
```

```
  kp = EEPROM.read(0);
  ki = EEPROM.read(1);
  kd = EEPROM.read(2);
  sv_suhu=EEPROM.read(4);

  if (kp>=255){kp=0;}
  if (ki>=255){ki=0;}
  if (kd>=255){kd=0;}
  if (sv_suhu>=255){sv_suhu=0;}

}


if (Serial.available()>1){
  String data =Serial.readStringUntil('\n');
  int a = data.indexOf("A");
  int b = data.indexOf("B");
  int c = data.indexOf("C");
  int d = data.indexOf("D");
  int e = data.indexOf("E");

  if (a==0&& b>0 && c>0 && d>0){
    String _kp = data.substring(a+1,b);
    String _ki = data.substring(b+1,c);
    String _kd = data.substring(c+1,d);
    String _sv = data.substring(d+1,e);

    kp = _kp.toInt();
    ki = _ki.toInt();
    kd = _kd.toInt();
    sv_suhu = _sv.toInt();
```

```
    Serial.println("Set Nilai KP: " + String(kp));
    Serial.println("Set Nilai KI: " + String(ki));
    Serial.println("Set Nilai KD: " + String(kd));
    Serial.println("Set SV: " + String(sv_suhu));

    EEPROM.write(0,kp);
    EEPROM.write(1,ki);
    EEPROM.write(2,kd);
    EEPROM.write(4,sv_suhu);

    delay(2000);

  }else{
    Serial.println(" Format Error ");
  }

}


float limit = (100);
sensors.requestTemperatures();
pv_suhu = sensors.getTempCByIndex(0);
error =  sv_suhu -pv_suhu;
pid  =  (kp*error)+ki*(error+last_error)*ts + (kd/ts)*(error - last_error);
last_error = error;
pv_pwm = sv_pwm - pid;


 if (pv_pwm>=255){pv_pwm=255;}
 else if (pv_pwm<=15){pv_pwm=15;}
 analogWrite(pin_pwm,pv_pwm);
 lcd.setCursor(0,0);
```

```
  lcd.print("T:");
  lcd.print(pv_suhu);
  lcd.print((char)223);
  lcd.print("C E:");

  lcd.print(error);
  lcd.print("   ");

  lcd.setCursor(0,1);
  lcd.print("PID:");
  lcd.print(pid);

  lcd.print(" S:");
  lcd.print(pv_pwm);
  lcd.print("   ");


  Serial.println("  ");
  Serial.print(pv_suhu);
 // Serial.print(" ");
 // Serial.print(sv_suhu);
 // Serial.print(" ");
 // Serial.println(limit);


  delay(100);

}
```