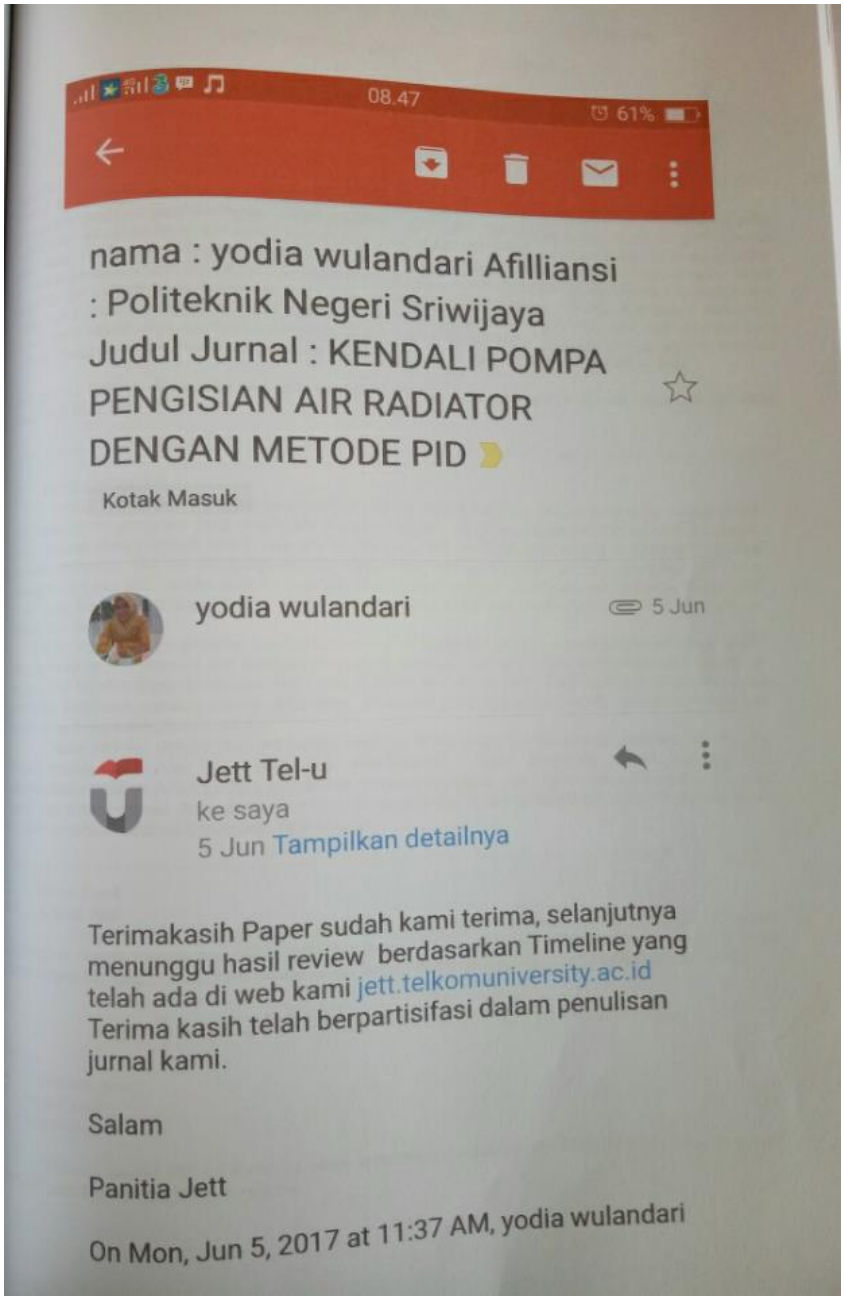

08.47
61%
nama : yodia wulandari Afilliansi
: Politeknik Negeri Sriwijaya
Judul Jurnal : KENDALI POMPA
PENGISIAN AIR RADIATOR
DENGAN METODE PID
Kotak Masuk
yodia wulandari
5 Jun
Jett Tel-u
ke saya
5 Jun Tampilkan detailnya
Terimakasih Paper sudah kami terima, selanjutnya
menunggu hasil review berdasarkan Timeline yang
telah ada di web kami jett.telkomuniversity.ac.id
Terima kasih telah berpartisifasi dalam penulisan
jurnal kami.
Salam
Panitia Jett
On Mon, Jun 5, 2017 at 11:37 AM, yodia wulandari

KENDALI POMPA PENGISIAN AIR RADIATOR DENGAN METODE PID
CONTROL OF RADIATOR WATER FILLING PUMP WITH PID METHOD

Yodia Wulandari[1],Ahmad Taqwa[2],Ibnu Ziad[3]

[123]ProgramStudiTeknikTelekomunikasiPoliteknik Negeri Sriwijaya,

Jl Srijaya Negara, Bukit Besar, Ilir Barat 1, Kota Palembang,
Sumatera Selatan

[1]yodiawulan@gmail.com[2]a_taqwa@yahahoo.com[3]ibnuziad759@gmail.com

Abstrak

Jumlah kendaraan meningkat pesat sampai saat ini terus bertambah. Seiring dengan itu jumlah mobil yang menggalami gangguan juga meningkat. Setiap kendaraan pasti perlu perawatan seperti ganti oli, ganti ban, isi minyak rem, isi air radiator, dan masih banyak lagi. Terkadang kita sebagai pengguna kurang memperhatikan perawataan kendaraan. Mungkin karena lupa, kurang memperhatikan atau tidak mempunyai waktu untuk melakukan perawatan pada kendaraan. Oleh karena itu, Wireless Sensor Network digunakan untuk kendali pompa pengisian air radiator pada kendaraan dengan menggunakan mikrokontroler ATMega 328, sensor ultrasonik. Modul wifi ESP8266. Pompa sentrifugal. Dan buzzer dengan menggunakan logika PID. Saat level air pada level rendah maka sensor akan mendeteksi kemudian diolah oleh mikrokontroler kemudian pompa akan mengisi air radiator. Pengguna bisa memantau melalui handphone dan laptop dengan menggunakan aplikasi *blynk.*

Katakunci: ***Wireless Sensor Network. Microcontroller ATMega328. PID metode. Sensor Ultrasonik HC-SR04. Modul Wifi ESP6288.***

Abstract

The number of vehicles increased rapidly to date continues to grow. Along with that the number of cars that menggalami interference also increased. Each vehicle will need maintenance such as oil change, tire change, brake fluid content, radiator water content, and much more. Sometimes we as users pay less attention to vehicle perawataan. Perhaps because of forgetting, lack of attention or do not have time to do

maintenance on the vehicle. Therefore, Wireless Sensor Network is used to control the radiator water pump in the vehicle by using ATMega 328 microcontroller, ultrasonic sensor. ESP8266 wifi module. Centrifugal pump. And buzzer by using PID logic. When the water level at the low level then the sensor will detect and then processed by the microcontroller then the pump will fill the water radiator. Users can monitor via mobile phones and laptops by using blynk applications.

Keywords: : ***Wireless Sensor Network. Microcontroller ATMega328. PID metode. Sensor Ultrasonik HC-SR04. Modul Wifi ESP6288*****.**

1. PENDAHULUAN

Jumlah kendaraan meningkat pesat sampai saat ini terus bertambah. Seiring dengan itu jumlah mobil yang menggalami gangguan juga meningkat. Semakin banyak pengguna kendaraan bermotor di Indonesia. Maka meningkatkan populasi mobil yang ada dijalan raya, hal ini dibuktikan dengan terjadinya kemacetan didaerah perkotaan. Hal ini dibuktikan dengan adanya data bahwa “Populasi mobil pada tahun 2000 baru mencapai 5,04 juta unit. Angka tersebut melonjak 117,7 persen menjadi 10,97 juta unit terhitung hingga Mei 2012," kata Ketua I Gabungan Industri Kendaraan Bermotor Indonesia (Gaikindo) Jongkie D Sugiarto, di Jakarta, Kamis.

Seiring bertambahnya populasi kendaraan bermotor di Indonesia meningkat pula kecelakaan yang terjadi yang disebabkan oleh tabrakan dua kendaraan bahkan lebih banyak kendaraan yang terlibat ataupun kecelakaan tunggal. Hal tersebut dapat disebabkan oleh berbagai faktor dari human error, Oleh karena itu jasa pelayanaan perawatan kendaraan semakin banyak dibutuhkan. Perawatan pada kendaraan banyak macamnya Perawatan tersebut bisa berupa mengisi air radiator.

Maka dari itu, penulis akan merancang prototipe WSN untuk kendali pompa pengisian air radiator dengan metode PID. Alat ini menggunakan mikrokontroler, sensor ultrasonik, modul wifi esp6288, mikrokontroler arduino ATMega 328, wifi Router, pompa sentrifugal, dan buzzer. Data dari sensor ini akan diolah oleh mikrokontroler dan mendeteksi adanya kebocoran, saat sensor mendeteksi level air berkurang maka alat akan memberi notifikasi berupa bunyi pada buzzer.

2. TINJAUAN PUSTAKA

2.1 Logika *PID*

. Controller *PID* merupakan jenis pengendali yang banyak digunakan. Selain itu sistem ini mudah digabungkan dengan metoda pengaturan yang lain seperti Fuzzy dan Robust. Sehingga akan menjadi suatu sistem pengendali yang semakin baik. Secara umum fungsi transfer dari PID controller adalah sebagai berikut :

$$kp + \frac{K_i}{S} + KD = \frac{K_D S^2 + K_p S + K_i}{S} \quad (1)$$

Dengan,

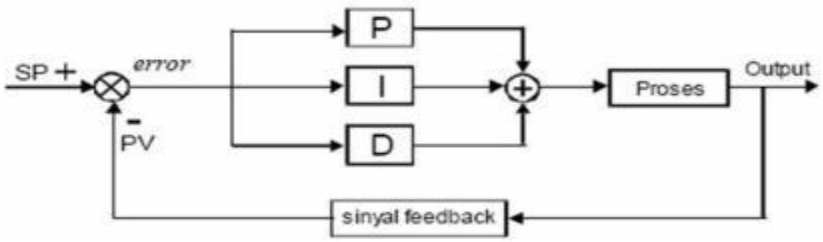
$_{Kp}$ = *Propotional gain*

$_{Ki}$ = *Integral gain*

$_{KD}$ = *Derivative gain*

Tunning kontrol PID ini bertujuan untuk menentukan parameter aksi kontrol Proportional, Integratif, Derivatif pada kendali pengisian air radiator. Proses ini dapat dilakukan dengan cara ***trial and error***. Keunggulan cara ini kita tidak diharuskan mengidentifikasi plant, membuat model matematis plant, menentukan parameter plant dengan grafis ataupun analitis namun cukup dengan cara coba-coba memberikan konstanta P-I-D pada formula PID hingga di peroleh hasil yang di inginkan, dengan mengacu pada karakteristik masing-masing kontrol P-I-D. Setelah didapatkan suatu sinyal kesalahan atau error, nilai error tersebut diolah dengan formula PID untuk dijadikan suatu sinyal kendali atau sinyal kontrol yang akan diteruskan ke actuator yakni berupa kendali gerak pada motor pompa dc.

Berikut ini adalah gambaran blok diagram umpan balik loop tertutup pada perancangan kedali PID pada kendali pengisian air radiator berikut ini:



Gambar 1 Blok Diagram suatu sistem loop tertutup

Dari blok diagram diatas dapat di jelasin sebagai berikut :

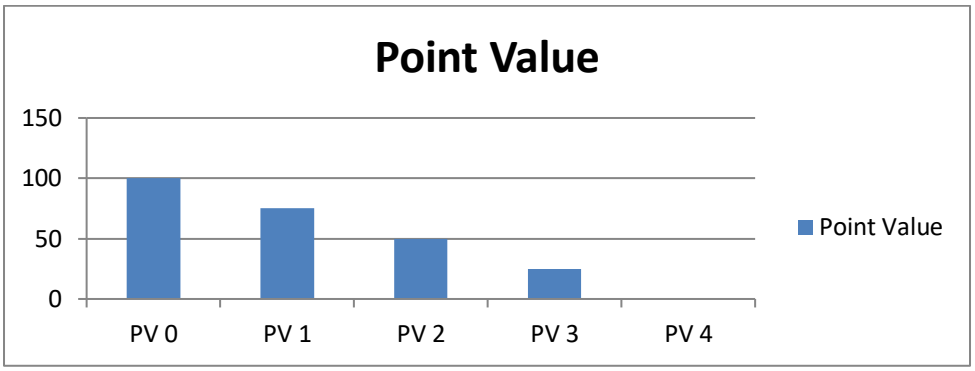
1. SP = Set point, secara simple maksudnya ialah suatu prameter nilai acuan atau nilai yang kita inginkan yaitu nilai ideal dari keadaan pengisian air radiator, dalam hal ini adalah kondisi 100%.
2. PV = Present Value, kalo yang ini maksudnya ialah nilai bobot pembacaan sensor saat itu atau variabel terukur yang di umpan balikan oleh sensor (sinyal feedback dari sensor). Nilai ini nantinya didapatkan

dari pembacaan sensor ultrasonic yang menerjemahkan keadaan pembacaan jumlah debit air pada penampungan

3. Error = nilai kesalahan, pengertiannya ialah Deviasi atau simpangan antar variabel terukur atau bobot sensor (PV) dengan nilai acuan (SP)

$$error = SP - PV \quad (2)$$

Berikut ilustrasi pemberian bobot sensor (nilai kesalahan pembacaan sensor) pada kendali pengisian air radiator:



Gambar 2 Point Value

Proses pemberian bobot sensor dapat dilakukan dengan proses pemetaan (mapping) pembacaan sensor terhadap kapasitas air yang berkurang, berikut sample proses mapping sensor:

Keadaaan kekurangan air 100% (PV=0)
Keadaaan kekurangan air 75% (PV=1)
Keadaaan kekurangan air 50% (PV=2)
Keadaaan kekurangan air 25% (PV=3)
Keadaaan kekurangan air 0% (PV=4)

Kondisi ideal terjadi saat kondisi air pada PV= 0 (misal kondisi nilai sensor dengan ketinggian 16.5 cm , nilai 0 merepresentasikan s ensor membaca keadaan air full, atau SP = 0 ialah kondisi ideal

dari kendali pengisian air radiator. Jika PV tidak sama dengan 0 berarti air tidak berada pada kondisi ideal dan artinya ada sinyal kesalahan (error). Pada kondisi error inilah formula PID akan menentukan hasil sinyal kendalinya. Berikut ini penjelasan tentang penerapan PID pada kendali pengisian air radiator:

Proporsional control:

Kondisi ideal pada system adalah ketika keasaan air 100% atau full, dengan kata lain PV = 0. Dari sini dapat diasumsikan bahwa Set Point (SP) / kondisi ideal adalah saat SP = 0. Nilai sensor yang dibaca oleh sensor disebut Process Variable (PV) / nilai aktual pembacaan. Menyimpangnya kondisi keadaan air dari set point disebut sebagai error (e), yang didapat dari e = SP – PV. Dengan mengetahui besar error, mikrokontroler dapat memberikan nilai PWM motor pompa dc yang sesuai agar dapat menuju ke posisi ideal (SP = 0). Besarnaya nilai PWM ini dapat diperoleh dengan menggunakan kontrol Proporsional (P), dimana P = e Kp (Kp adalah konstanta proporsional yang nilainya di set sendiri dari hasil tuning).

Perhitungan Kontroler Proporsional

```
Sp_sensor = 0                 'setpoint sensor
Error = Sp_sensor – Pv        'nilai error
P = Kp * Error                'proporsional control
```

Aplikasi kontrol proporsional pada PWM ialah sebagai berikut:

```
Pwm = Sp_pwm + P              'motor kiri
```

Derivatif control

Kontrol D digunakan untuk mengukur seberapa cepat respon pompa menanggapi kondisi pembacaan sensor. Semakin cepat, maka semakin besar nilai D. Konstanta D (Kd) digunakan untuk menambah atau mengurangi imbas dari derivatif. Nilai D didapat dari D = Kd/Ts * rate, dimana Ts ialah time sampling atau waktu cuplik dan rate = e(n) – e(n-1). Dalam program nilai error (SP – PV) saat itu menjadi nilai last_error, sehingga rate didapat dari error – last_error. Untuk menambahkan kontrol D, program dimodifikasi menjadi:

Perhitungan Kontroler Proporsional + Derivatif

```
Sp_sensor = 0                 ‘setpoint sensor

Error = Sp_sensor – Pv        ‘nilai error

P = Kp * Error                ‘proporsional control

D1 = Kd * 10                  ‘derivatif control

D2 = D1 / Ts

D3 = Error – Last_error       ‘rate

D = D2 * D3

Last_error = Error            ‘error lampau

Pd = P + D                    ‘proporsional-derivatif control
```

Aplikasi kontrol proporsional dan drivatif pada PWM ialah sebagai berikut: Pwm = Sp_pwm + Pd ‘pwm motor pompa dc

Integratif control

Jika dengan P + D sudah membuat pergerakan motor pompa cukup *smooth*, maka penambahan Integratif menjadi opsional. Integratif digunakan untuk mengakumulasi error dan mengetahui durasi error. Dengan menjumlahkan error disetiap pembacaan PV akan memberikan akumulasi offset yang harus diperbaiki sebelumnya. Semakin lama tidak mendapatkan SP, maka semakin besar nilai I. Degan mendapatkan nilai Ki yang tepat, imbas dari Integratif bisa dikurangi.

Nilai akumulasi error didapat dari: error + last_error. Untuk menambahkan kontrol I, maka program dimodifikasi menjadi:

Perhitungan Kontroler Proporsional + Integratif + Derivatif

```
Sp_sensor = 100               ‘setpoint sensor

Error = Sp_sensor – Pv        ‘nilai error

P = Kp * Error                ‘proporsional control

D1 = Kd * 10                  ‘derivatif control

D2 = D1 / Ts

D3 = Error – Last_error       ‘rate
```

D = D2 * D3

I1 = Ki / 100 ‘integratif control

I2 = Error + Last_error ‘akumulasi error

I3 = I1 * I2

I = I3 * Ts

Last_error = Error ‘error lampau

Pd = P + D ‘proporsional-derivatif control

Pid = Pd+I ‘proporsional-integratif-derivatif

Aplikasi kontrol proporsional, integratif dan drivatif pada PWM ialah sebagai berikut:
Pwm = Sp_pwm + Pid ‘motor pompa dc

2.2 Mikrokontroler ATMega 328

Mikrokontroler (pengendali mikro) pada suatu rangkaian elektronik berfungsi sebagai pengendali yang mengatur jalannya proses kerja dari rangkaian elektronik. Didalam sebuah IC mikrokontroler terdapat CPU, Memori, Timer, Saluran komunikasi serial dan pararel, port input/output, ADC, dan lain lain.

2.3 Modul Wifi ESP8266

ESP8266 adalah sebuah komponen chip terintegrasi yang didesain untuk keperluan dunia masa kini yang serba tersambung. Chip ini menawarkan solusi networking Wi-Fi yang lengkap dan menyatu, yang dapat digunakan sebagai fungsi networking Wi-Fi ke pemproses aplikasi lainnya.

2.4 Sensor Ultrasonik

Sensor ultrasonik bekerja dengan cara memancarkan satu gelombang dan kemudian menghitung waktu pantulan gelombang tersebut. Gelombang ultrasonik bekerja pada frekuensi mulai dari 20 Khz sampai dengan 20 Mhz.

2.5 Buzzer

Buzzer Listrik adalah sebuah komponen elektronika yang dapat mengubah sinyal listrik menjadi getaran suara.

2.6 WSN (Wireless sensor network)

Wireless Sensor Network (WSN) merupakan suatu kesatuan dari proses pengukuran, komputasi, dan komunikasi yang memberikan kemampuan administratif kepada sebuah perangkat, observasi, dan melakukan penanganan terhadap setiap kejadian dan fenomena yang terjadi di lingkungan yang mengunakan teknologi *wireless.*

2.7 Pompa Sentrifugal

Pompa sentrifugal merupakan pompa kerja dinamis yang menghasilkan head melalui putaran impeller, sehingga ada hubungan antara kecepatan keliling impeller dan head yang dibangkitkan.

2.8 L298N

L298N adalah jenis IC driver motor yang dapat mengendalikan arah putaran dan kecepatan motor DC ataupun Motor stepper. Mampu mengeluarkan output tegangan untuk Motor dc dan motor stepper sebesar 50 volt. IC l298 terdiri dari transistor-transistor logik (TTL) dengan gerbang nand yang

2.9 Relay

Pengertian Relay dan Fungsinya – Relay adalah Saklar (Switch) yang dioperasikan secara listrik dan merupakan komponen Electromechanical (Elektromekanikal) yang terdiri dari 2 bagian utama yakni Elektromagnet (Coil) dan Mekanikal (seperangkat Kontak Saklar/Switch

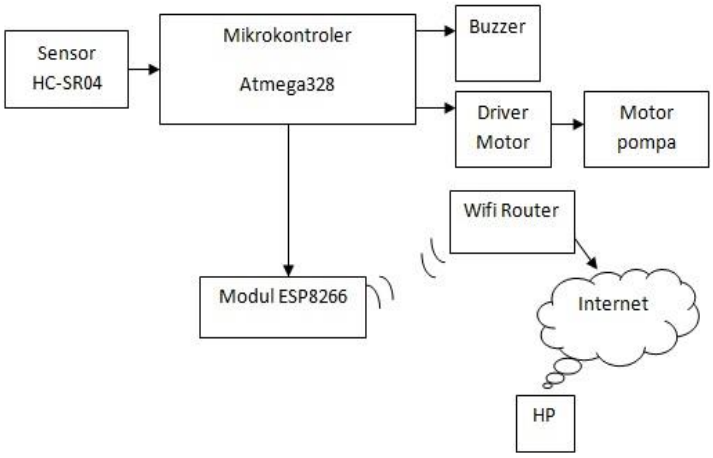
2.10 Power Supplay

Power Supply adalah perangkat keras yang berfungsi untuk menyuplai tegangan langsung kekomponen dalam casing yang membutuhkan tegangan, misalnya motherboard, hardisk, kipas, dll. Input power supply berupa arus bolak-balik (AC) sehingga power supply harus mengubah tegangan AC menjadi DC (arus searah), karena hardware komputer hanya dapat beroperasi dengan arus DC. Power supply berupa kotak yang umumnya diletakan dibagian belakang atas casing.

3. PEMBAHASAN

3.1 Perancangan Perangkat Keras (*Hardware*)

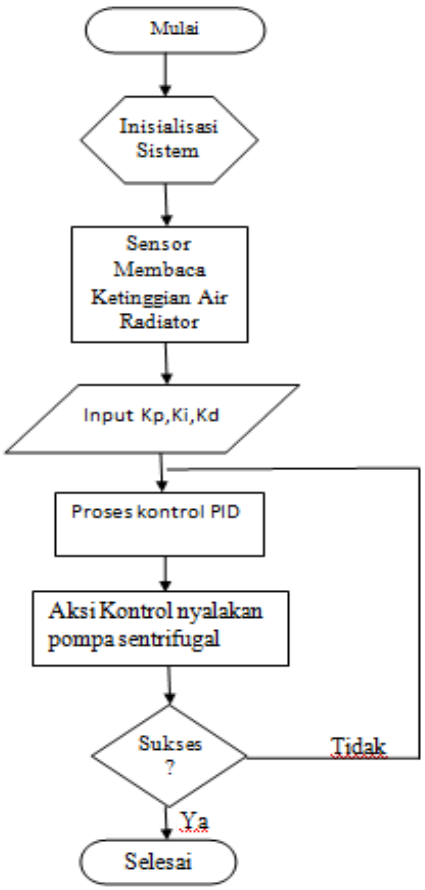
Dalam merancang perangkat keras dibutuhkan beberapa komponen. Perangkat ini terdiri dari sensor ultrasonik HC-SR04, modul Arduino Atmega328, Modul Wifi ESP8266, dan handphone. Driver motor, pompa, tangki *reservoil*, dan *buzzer*. Sensor ultrasonik HC-SR04 berfungsi memberikan informasi real-time pada arduino mengenai kondisi level air radiator dalam tangki melalui port serial COM. Arduino mengolah data yang dapat ditampilkan di laptop dan dengan jaringan internet melalui modul wifi ESP8266 kita bias memantau melalui *handphone.* Adapun Diagram blok system secara keseluruhan sebagai berikut :



Gambar 3 Blok Diagram Perangkat Keras (*Hardware*)

3.2 Perancangan Perangkat Lunak (*Software*)

Sistem ini dirancang untuk mengendalikan pompa pengisian air radiator pada tangki kendaraan dengan menggunakan sensor ultrasonik sebagai pendeteksi jarak ketinggian level air radiator. Sistem ini akan mulai beroperasi ketika diberikan sumber tegangan melalui *port* USB laptop ke Arduino. *Port* USB ini berfungsi sebagai jembatan pengiriman data dari Arduino ke laptop.



Gambar 4 Blok Diagram Perangkat Lunak (*Software)*

3.3 Pengujian Sensor Ultrasonik

Pengujian sensor ultrasonik dilakukan untuk mengetahui apakah sensor bekerja dengan baik atau tidak. Adapun hasil pembacaan dari sensor ultrasonic berupa nilai ketinggian air. Pengujian dilakukan dengan cara menghubungkan sensor ultrasonik dengan sistem minimum yang telah diberikan program pembacaan sensor ultrasonik dengan mengeluarkan data jarak dilaptop untuk melakukkan pengecekan data jarak yang dibaca oleh sensor. Sensor diletakan di dalam tangki, kemudian diproses oleh arduino dan nilai ketinggian air ditampilkan pada laptop.

Tabel 2 Data Hasil Pengukuran Sensor HC-SR04

No	Data Ultrasonik	Nilai Ketinggian Air (liter)	Buzzer
1	15,26 cm	1, 86	OFF

2	12,03 cm	1,46	OFF
3	8,24 cm	1,00	OFF
4	4,62 cm	0,56	ON

3.4 Pengukuran PWM Berdasarkan Kondisi Sensor Sesuai Dengan Logika PID

Pengujian sensor ultrasonik dilakukan untuk mengetahui apakah sensor bekerja dengan baik atau tidak. Adapun hasil pembacaan dari sensor ultrasonic berupa nilai ketinggian air. Pengujian dilakukan dengan cara menghubungkan sensor ultrasonik dengan sistem minimum yang telah diberikan program pembacaan sensor ultrasonik dengan mengeluarkan data jarak dilaptop untuk melakukkan pengecekan data jarak yang dibaca oleh sensor. Sensor diletakan di dalam tangki, kemudian diproses oleh arduino dan nilai ketinggian air ditampilkan pada laptop.

Perhitungan Kontroler Proporsional + Integratif + Derivatif

```
Sp_sensor = 100                          'setpoint sensor
Error = Sp_sensor – Pv                   'nilai error
P = Kp * Error                           'proporsional control
D1 = Kd * 10                             'derivatif control
D2 = D1 / Ts
D3 = Error – Last_error                  'rate
D = D2 * D3
I1 = Ki / 100                            'integratif control
I2 = Error + Last_error                  'akumulasi error
I3 = I1 * I2
I = I3 * Ts
Last_error = Error                       'error lampau
Pd = P + D                               'proporsional-derivatif control
Pid = Pd+I                               'proporsional-integratif-derivatif
```

Aplikasi kontrol proporsional, integratif dan drivatif pada PWM ialah sebagai berikut:

```
Pwm = Sp_pwm + Pid    'motor pompa dc
```

Tabel 3 Data Hasil Pengukuran PWM Berdasarkan Kondisi Sensor Sesuai Dengan Logika PID

No	Nilai Level	Nilai PWM	Error	Vout Motor	Nilai konstanta PID	Time (s)
1	High	255	100%	8,63 V	227	02:10 m

2	Medium	200	75%	6,63 V	200	01: 15 m
3	Low	150	50%	2,78 V	140	34:77 s
4	Lower	100	25%	0,03 V	80	17:28 s

4. KESIMPULAN

Hasil pengujian dan analisis yang telah dilakukan pada perancangan kendali pompa pengisian air radiator dengan menggunakan metode PID adalah sebagai berikut.

1. Level ketinggian Sensor Ultrasonik pada tangki air radiator ada 4 level yaitu level 1: tinggi high, level 2: medium, level 3:low, level 4:lower.
2. Dari percobaan ini, jika PWM 255 atau berada pada level high maka pompa akan mengisi dengan cepat, dan sebaliknya jika PWM 100 maka pompa akan mengisi dengan sangat lambat.
3. *Buzzer* akan ON jika level air radiator berada di posisi sangat rendah .

DAFTAR PUSTAKA

[1] Anonim, jurnal sistem perawatan air brake (rem) pada kendaraan, 2014

[2] Syahrir Abdussamad, Simulasi kendalian Flow control unit G.U.N.T tipe 020 dengan pengendali PID, 2009

[3] Yopi Sukita Defriyadi, Pengendali intensitas cahaya, suhu, dan kelembapan pada rumah kaca dengan metode PID, 2014

[4] Heri Andrianto, Aan Darmawan, Arduino belajar cepat dan pemrograman, 2016

[5] Wahyu Djalmono Putro, Pengujian Kinerja Pompa Sentrifugal Menggunakan Kontrol Inventer, 2010

[6] Eko Radiansyah, Muhamad Suyatno, Sunnarsih, Analisis dan pemanfaatan jaringan wireless menggunakan linksys smart wifi cisco router e2500

[7] M. Y. Hariyawan1, A. Gunawan2 & E.H. Putra, Implementasi Wireless Sensor Network untuk Pendeteksi Dini Kebakaran Hutan, 2009

[8] Fahmizal, Metode Tuning Ziegler Nichols dalam desain kontroler PID, 2010

```
#define BLYNK_PRINT Serial

const int V_a=0;                    // Virtual pin on Blynk app to indicate water level is LOW

const int V_b=1;

const int V_c=2;                    // Virtual pin on Blynk app to indicate water level is LOW

const int V_d=3;


#define echoPin D8 // Echo Pin

#define trigPin D9 // Trigger Pin


long duration, distance; // Duration used to calculate distance

float percent;                      // Percentage of Water Level in the tank


#include <ESP8266WiFi.h>

#include <BlynkSimpleEsp8266.h>


// You should get Auth Token in the Blynk App.

// Go to the Project Settings (nut icon).

char auth[] = "a48c7fd5e5e74b0daabf02c1e1948330";

// Your WiFi credentials.

// Set password to "" for open networks.
```

```
char ssid[] = "yodiawulandari";

char pass[] = "wulandarii";


// This code will update the virtual port 5

BLYNK_WRITE(V12) {

  int pinData = param.asInt();

}


void setup(){

Serial.begin(74880);                          //baud rate should be set as 74880


//  Serial.begin(9600);

//   pinMode(led_pin, OUTPUT);

 pinMode(trigPin, OUTPUT);

 pinMode(echoPin, INPUT);

  Blynk.begin(auth, ssid, pass);

}


void loop(){


  Blynk.run();
```

```
if (Serial.available()) {       // If anything comes in Serial (USB),


 Blynk.notify(Serial.read()); // You get a notification when tank is Full. Make Sure you included Notification Widget in the App


  //  Serial1.write(Serial.read());   // read it and send it out Serial1 (pins 0 & 1)

  }


 Serial.println("Tank is almost full. please turn off the motor");


 //Serial.println(distance);

  // Initiates Blynk

// BACA US


                //..... Start of code for Ultra Sonic sensor.....//

digitalWrite(trigPin, LOW);

delayMicroseconds(2);


digitalWrite(trigPin, HIGH);

delayMicroseconds(10);
```

```
digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

                //..... End of code for Ultra Sonic sensor.....//

Serial.println("Sleep");

//ESP.deepSleep(SLEEP_LENGTH * 1000000,WAKE_RF_DEFAULT);    //try the default deep sleep mode

delay(100);

Serial.println("wake");


                //.....Calculate the distance (in cm) based on the speed of sound..//

distance = duration/58.2;

//percent = 100 - (distance/3);


percent=map(distance,2,18,100,0);

Serial.println(percent);

if(distance<=2){distance=2;}

if(distance>=17){distance=17;}

distance=map(distance,2,18,16,0);

Serial.println(distance);
```

```
Blynk.run();        // Main function to run Blynk app

if(percent <=25  )  // depending on the tank parameters change these values,"25"

{

  Blynk.virtualWrite(V_a,1023); // Low LED is ON

  Blynk.virtualWrite(V_b,0);    // High LED is OFF

  Blynk.virtualWrite(V_c,0); // Low LED is ON

  Blynk.virtualWrite(V_d,0);    // High LED is OFF


  Blynk.notify("Tank is SANGAT RENDAH "); // You get a notification when tank is Empty. Make Sure you included Notification Widget in the App

  Serial.println("Tank is SANGAT RENDAH ");

  }


if(percent >25 && percent <=50 )  // depending on the tank parameters change these values,"25"

{

  Blynk.virtualWrite(V_a,0); // Low LED is ON

  Blynk.virtualWrite(V_b,1023);    // High LED is OFF

  Blynk.virtualWrite(V_c,0); // Low LED is ON

  Blynk.virtualWrite(V_d,0);    // High LED is OFF


  Blynk.notify("Tank is RENDAH "); // You get a notification when tank is Empty. Make Sure you included Notification Widget in the App
```

```
  Serial.println("Tank is RENDAH ");
  }

if(percent >50 && percent <=75  )  // depending on the tank parameters change these values,"25"
{

  Blynk.virtualWrite(V_a,0); // Low LED is ON
  Blynk.virtualWrite(V_b,0);    // High LED is OFF
  Blynk.virtualWrite(V_c,1023); // Low LED is ON
  Blynk.virtualWrite(V_d,0);    // High LED is OFF

  Blynk.notify("Tank is SEDANG. "); // You get a notification when tank is Empty. Make Sure you included Notification Widget in the App
  Serial.println("Tank is SEDANG. ");
  }
if(percent >75  )  // depending on the tank parameters change these values,"25"
{

  Blynk.virtualWrite(V_a,0); // Low LED is ON
  Blynk.virtualWrite(V_b,0);    // High LED is OFF
  Blynk.virtualWrite(V_c,0); // Low LED is ON
```

```
  Blynk.virtualWrite(V_d,1023);    // High LED is OFF


  Blynk.notify("Tank is TINGGI. "); // You get a notification when tank is Empty. Make Sure you included Notification Widget in the App
  Serial.println("Tank is TINGGI. ");
  }



 Serial.println(distance);
  // Initiates Blynk
  Blynk.virtualWrite(8,distance); // Write depth of water to App
  Blynk.virtualWrite(9,percent);  // Write the percentage of water in tank
 Serial.println("distance written");
}
```

```
const int pingPin_e = A1;

const int pingPin_t = A0;


int error;

int kp=9;

int kd=8;

int ki=8;


int ts=1;

int kpid;

int lasterror;

int i1;

int i2;

int i3;

int d1;

int d2;


int i;

int d;

int p;
```

```
int kec;

int setkec;

const int buzzer = 6;

const int motor = 9;


const int pos_sensor = 19;

float ketinggian=0;

const float lebar=9.0;

const float panjang=13.5;


float liter;

float kubik;

float x;


int gain=58;


void setup() {

  // initialize serial communication:

  Serial.begin(9600);

  pinMode(buzzer,OUTPUT);

  pinMode(motor,OUTPUT);
```

```
}
void loop()
{
  // establish variables for duration of the ping,
  // and the distance result in inches and centimeters:
  long duration, inches;
  float cm;

  // The PING))) is triggered by a HIGH pulse of 2 or more microseconds.
  // Give a short LOW pulse beforehand to ensure a clean HIGH pulse:
  pinMode(pingPin_t, OUTPUT);
  digitalWrite(pingPin_t, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin_t, HIGH);
  delayMicroseconds(5);
  digitalWrite(pingPin_t, LOW);

  // The same pin is used to read the signal from the PING))): a HIGH
  // pulse whose duration is the time (in microseconds) from the sending
  // of the ping to the reception of its echo off of an object.
  pinMode(pingPin_e, INPUT);
```

```
  duration = pulseIn(pingPin_e, HIGH);

  // convert the time into a distance
  inches = microsecondsToInches(duration);
  cm = microsecondsToCentimeters(duration);
 cm=float(cm/29.0);
cm=float(cm/2.0);

  ketinggian=float(pos_sensor - cm);
if(ketinggian<0){ketinggian=0;}

if(ketinggian<=4){digitalWrite(buzzer,HIGH);}
if(ketinggian>4){digitalWrite(buzzer,LOW);}

kubik=float(panjang*lebar*ketinggian);
liter=float(kubik*0.001);

  Serial.println();
//
Serial.print(liter);
  Serial.print("liter");
```

```
  Serial.println();
  Serial.print("tinggi");
Serial.println(ketinggian);
x=ketinggian;

hitungerror();
pid();
printpwm();
  delay(2000);
}

void hitungerror(){
lasterror=error;
if (x>=0 && x<2)
    {error=10;}
if (x>=2 && x<4)
    {error=9;}
if (x>=4 && x<6)
    {error=8;}
if (x>=6 && x<8)
    {error=7;}
```

```
if (x>=8 && x<9)
    {error=6;}
if (x>=9 && x<11)
    {error=5;}
if (x>=11 && x<12)
    {error=4;}
if (x>=12 && x<13)
    {error=3;}
if (x>=14 && x<15)
    {error=2;}
if (x>=15 && x<16)
    {error=1;}
if (x>=16)
    {error=0;}

Serial.print("error");
Serial.println(error);
Serial.print("last error");
Serial.println(lasterror);

}
```

```
//=================


void pid(){
//==========
  p=kp*error;
Serial.print("  P  =");
Serial.println(p);
//==========
i1=ki;
i2=error+lasterror;
i3=i1*i2;
i=i3*ts;
Serial.print("  I  =");
Serial.println(i);
//=========
d1=kd*ts;
d2=error-lasterror;
d=d1*d2;
Serial.print("  D  =");
Serial.println(d);
}
```

```
void printpwm(){
kec=setkec+p+i+d;
if(kec<=55 && kec>0){kec=kec+gain;}

Serial.print("pwm");
Serial.println(kec);
analogWrite (motor,kec);
}

long microsecondsToInches(long microseconds)
{
  // According to Parallax's datasheet for the PING))), there are
  // 73.746 microseconds per inch (i.e. sound travels at 1130 feet per
  // second).  This gives the distance travelled by the ping, outbound
  // and return, so we divide by 2 to get the distance of the obstacle.
  // See: http://www.parallax.com/dl/docs/prod/acc/28015-PING-v1.3.pdf
  return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds)
```

```
{
  // The speed of sound is 340 m/s or 29 microseconds per centimeter.
  // The ping travels out and back, so to find the distance of the
  // object we take half of the distance travelled.
  return microseconds;
}
```