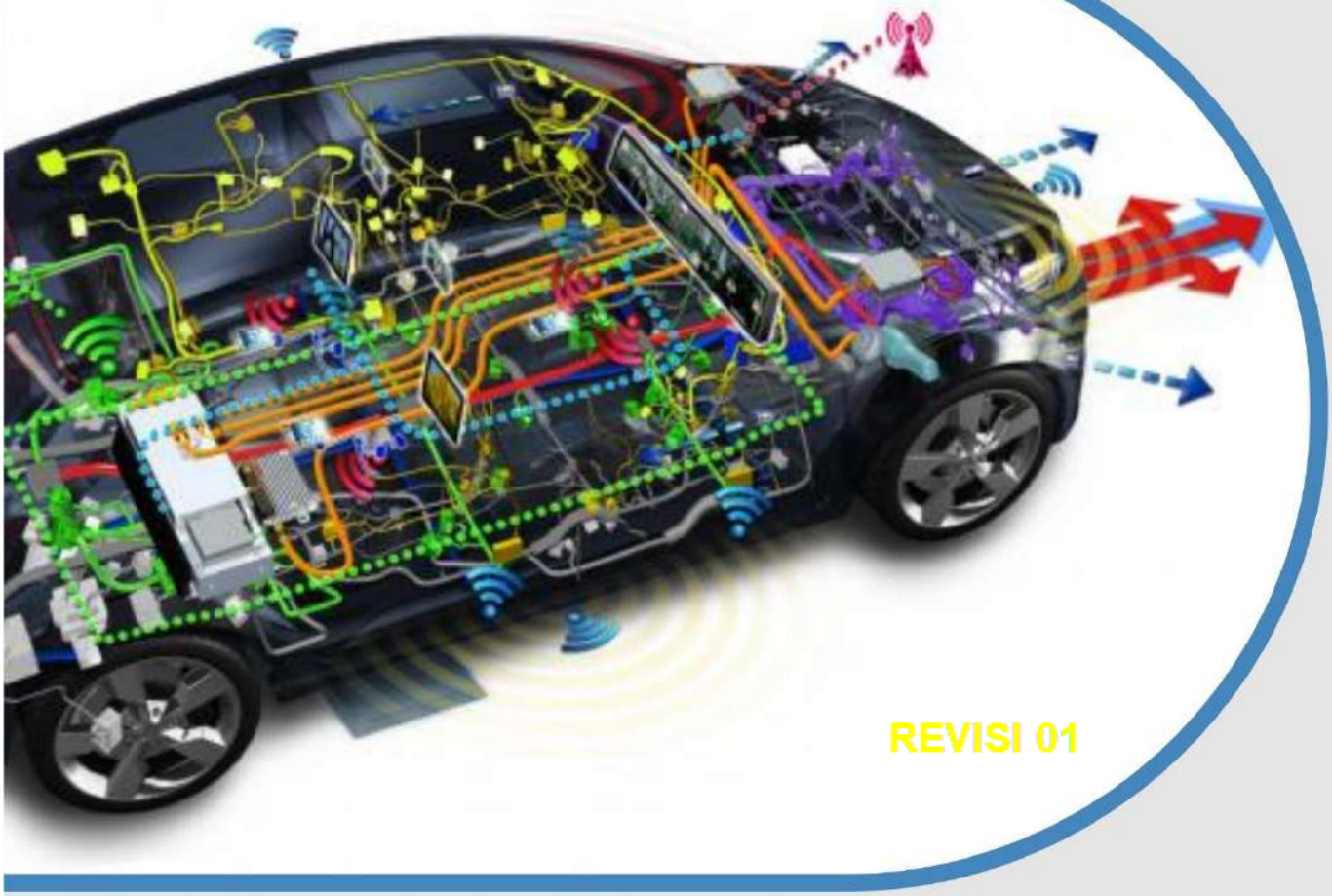
PROGRAM STUDI MESIN OTOMOTIF
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH MAGELANG

MODUL PRAKTEK

Mata Kuliah:

Automotive Mechatronics

PM-UMM-02-03/L1



REVISI 01

Alamat:
Gedung C Kampus 2 Universitas Muhammadiyah Magelang
Jl. Mayjend Bambang Soegeng Km. 5 Mertoyudan Magelang

Automotive Mechatronics

PENGESAHAN

MODUL PRAKTEK

Mata Kuliah:
Automotive Mechatronics - KPT0503423

Form/STD/04.03-01.01

Revisi	:	01
Tanggal	:	02 -08- 2022
Dikaji Ulang Oleh	:	Peer Review
Dikendalikan Oleh	:	Gugus Kendali Mutu Fakultas
Disetujui Oleh	:	Kepala Program Studi

No. Dokumen	: PM – UMM – 02-03/L1	Tanggal	: 30-08-2022
No.Revisi	00	No. Hal	:

Disiapkan oleh :	Diperiksa oleh :	Disahkan oleh :
Koordinator Mata Kuliah	Peer Review	Ketua Program Studi Mesin Otomotif
Suroto Munahar, ST., MT	Dr. Budi Waluyo, ST.,MT	Bagiyo [illegible]., ST., M.Eng
NIDN. 0620127805	NIDN. 0627057701	NIDN. 8868960018

INFORMASI PRAKTEK

Spesifikasi Mata Kuliah

Nama Mata Kuliah	:	AUTOMOTIVE MECHATRONICS
Kode Mata Kuliah	:	KPT0503423
Bobot	:	4 sks
Substansi kajian		Signal, Actuator, & Wiring, Control systems, Microcontroller Data acquisition, dan Embedded Systems.
Capaian Pembelajaran Mata Kuliah (CPMK)	:	1. Menguasai konsep dan mampu menerapkan *signal, actuator* & *wiring* pada kendaraan. 2. Menguasai konsep dan mampu menerapkan *control systems* pada kendaraan. 3. Menguasai konsep dan mampu menerapkan *microcontroller* pada kendaraan. 4. Menguasai konsep dan mampu menerapkan *data acquisition* pada kendaraan. 5. Menguasai konsep dan mampu menerapkan *embeded systems* pada kendaraan berdasarkan MBKM (*Case Based Learning).*
Capaian Pembelajaran Praktek		1. Mampu menerapkan *signal, actuator* & *wiring* pada kendaraan. 2. Mampu menerapkan *control systems* pada kendaraan. 3. Mampu menerapkan *microcontroller* pada kendaraan. 4. Mampu menerapkan *data acquisition* pada kendaraan. 5. Mampu menerapkan *embeded systems* pada kendaraan berdasarkan MBKM (*Case Based Learning).*
Kualifikasi pengampu	:	Dosen berkualifikasi akademik minimal S2 dan memiliki pengalaman penelitian dalam bidang *mechatronics* untuk diintegrasikan ke dalam pembelajaran.
Sarana dan Prasarana	:	1. Laboratorium komputer yang dilengkapi dengan software yang mendukung CPMK. 2. *Microcontroller*, komponen *electronics* dan komponen kendaraan.

1.1. Pengampu

Nama (Pengampu 1)	:	Suroto Munahar, ST., MT
NIDN	:	0620127805
Nama (Pengampu 2)	:	Mukhtar Hanafi, ST., MCs.
NIDN	:	0602047502
Jabatan	:	Lektor.
Fakultas/Program Studi	:	Teknik/Mesin Otomotif.
Universitas	:	Universitas Muhammadiyah Magelang

INTEGRASI PENELITIAN KE DALAM PRAKTEK

Pengalaman penelitian dosen yang diintegrasikan kedalam praktek ini antara lain:

No	Tahun	Judul Penelitian	Pendanaan
1.	2017	**Penelitian Dosen Pemula:** Pengembangan Engine Control Unit-(ECU) Pada EFI Engine Dengan Drive Train Controller*	PRVI
2.	2019	**PTUPT**: Sistem Kontrol Aliran LPG Pada Kendaraan Bi-Fuel (LPG-Bensin) Untuk Meningkatkan Efisiensi Penggunaan Bahan Bakar**	Kemenristek-dikti
3.	2019	**Penelitian Akselerasi:** Pengembangan ECU Of LPG Injection Untuk Mengendalikan Air To Fuel Ratio- AFR Berdasarkan Kondisi Jalan Pada Kendaraan Bi-Fuel***	PRVI
4.	2020	**PTUPT:** Sistem Kontrol Aliran LPG Pada Kendaraan Bi-Fuel (LPG-Bensin) Untuk Meningkatkan Efisiensi Penggunaan Bahan Bakar****	Kemenristek-dikti
5.	2021	**Penelitian Akselerasi:** Analisis Pengembangan Engine Control Unit (ECU) Modelling Untuk Pengendalian Bahan Bakar Berdasarkan Driver Behavior Pada Passenger Car*****	PRVI
6.	2021	**PDUPT** Studi pemanfaatan Artificial Intelligence (AI) sebagai Sistem Kontrol Mesin pada Kendaraan Berbahan Bakar CNG untuk Meningkatkan Efisiensi******	Kemenristek/BRIN
7.	2022	**Penelitian Dasar** Studi pemanfaatan Artificial Intelligence (AI) sebagai Sistem Kontrol Mesin pada pada Kendaraan Berbahan Bakar CNG untuk Meningkatkan Efisiensi*******	Kemenristek/BRIN

DAFTAR ISI

TATA TERTIB PRAKTIKUM

Dalam melakukan praktikum ada beberapa tata tertib yang dijalankan :

1. Ketika selesai praktikum mahasiswa diwajibkan membersihan tempat praktikum.
2. Membuat laporan harian.
3. Membuat laporan praktikum.
4. Mahasiswa dilarang mengenakan asesoris yang tidak pantas.
5. Mahasiswa berambut pendek dan rapi serta berkuku pendek.
6. Mahasiswa dilarang bercanda ketika melakukan praktikum .
7. Mahasiswa dilarang mencorat-coret dan merusak fasilitas bengkel.
8. Mahasiswa datang 15 menit sebelum praktikum dimulai.
9. Mahasiswa wajib menjaga keutuhan bahan dan alat praktikum.
10. Mahasiswa harus menggunakan modul yang telah disediakan.
11. Ketika praktik berhati-hati terhadap bahaya konsleting.

BAB 1. KEGIATAN BELAJAR SUB CPMK 1

1.1. TARGET CPL MATA KULIAH

SUB CPMK	TUJUAN
AM - 01	Menguasai konsep dan mampu menerapkan *signal, actuator* & *wiring* pada kendaraan

1.2. TARGET PEMBELAJARAN PRAKTEK

SUB CPMK	TUJUAN PEMBELAJARAN PRAKTEK
AM - 01	Mampu menerapkan *signal, actuator* & *wiring* pada kendaraan

1.3. URAIAN

Target pembelajaran praktek pada sub CPMK AM-01 fokus pada pencapian untuk mampu menerapkan *signal, actuator* & *wiring* pada kendaraan. Untuk mencapai target ini ada beberapa langkah yang harus dilakukan:

- Mampu memonitoring signal yang dibangkitkan oleh sensor pada kendaraan. Signal yang dibangkitkan sensor perlu dimonitoring untuk proses pemeriksaan, penyimpanan, evaluasi dan analisis. Hal ini perlu dilakukan dengan cara merancang data monitoring/data akuisisi dari signal yang dihasilkan oleh sensor kendaraan.
- Membuat rangkaian instalasi / *wiring* untuk membaca signal yang dihasilkan oleh sensor / *actuator* pada kendaraan.

Adapun untuk merancang data monitoring / data akuisisi dapat dilakukan dengan *LabVIEW for everyone*.

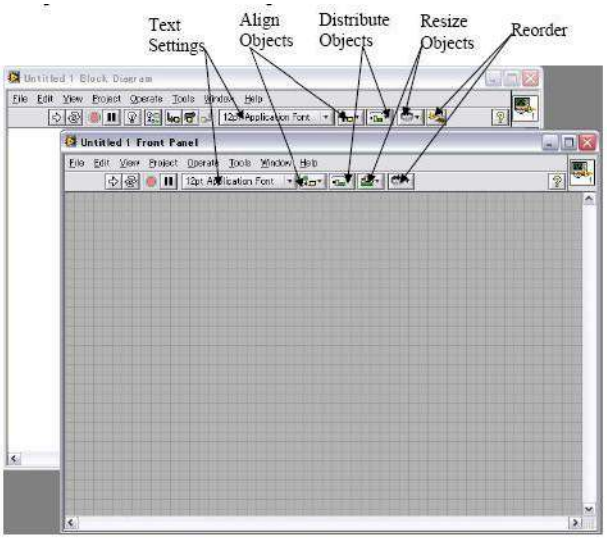
1.4. ALAT DAN BAHAN PRAKTEK

No.	Deskripsi	Item	Keterangan
1.	*Software LabVIEW*	1 Paket	National Instrument
2.	Komputer	24 unit	Ram 4/6/8 Giga
3.	Sensor kendaraan	1 set	Sensor TPS, MAP, ECT dan lain-lain
4.	Kabel	1 set	Jumper, NYY/NYA
5.	*Microcontroller*	1 set	NI 6008/Mega 2560

1.5. *SOFTWARE LABVIEW FOR EVERYONE*

LabVIEW For Everyone merupakan pemrograman grafis yang menjadi salah satu dasar dalam pengembangan teknologi sistem kontrol. Pemrograman ini dikembangkan berdasarkan pada penggunakan simbol maupun *icon*. Pemrograman grafis menjadi salah satu pemrograman data akuisisi tingkat tinggi. Hal ini disebabkan karena dengan pemrograman ini dibuat lebih mudah dipahami dan memudahkan dalam mencari error. Pemrograman grafis dikembangkan dari pemrograman bahasa C, bahasa C++, bahasa matematis, bahasa DOS dan lain – lain. Baru-baru ini pemrograman grafis banyak dikembangkan seperti *software Matlab Simulink* [1], *Ledder*[2], *LabView* [3] dan lain – lain. Modul praktik ini fokus mempelajari tentang pemrograman menggunakan *Software LabView* keluaran dari *National Intrument.*

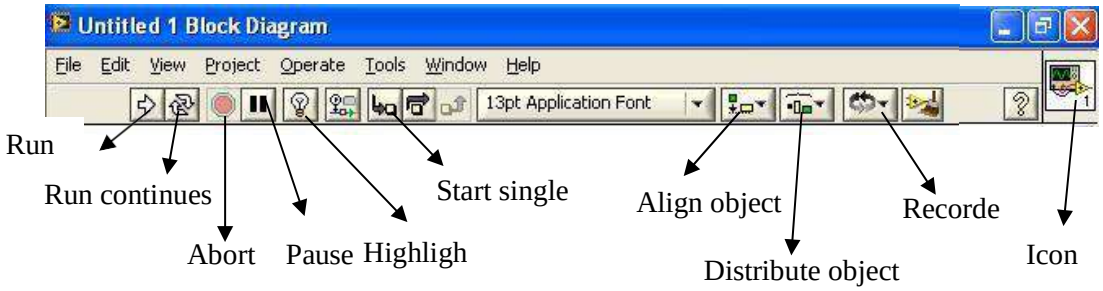
LabVIEW adalah sebuah *software* pemrograman buatan *National Instrument* dengan konsep yang berbeda seperti bahasa pemrograman lainnya yaitu: *C++*, *Matlab* atau *Visual Basic*, tetapi mempunyai fungi yang sama [4]. Bahasa pemrograman *LabView* berbasis pada grafis atau blok diagram sementara yang lain menggunakan basis text. *LabView* bekerja mempunyai dua bagian yaitu: *front panel* digunakan sebagai *user interface* yang akan mensimulasikan panel untuk instrument dan *block diagram* digunakan sebagai *source code* dibuat dan berfungsi sebagai instruksi *untuk front panel* sajikan pada Gambar.1.1.



Gambar 1.1 *Front panel* dan *block diagram* [4].

1.5.1 *Menu Bar* Pada *Front Panel LabVIEW*

Mengenal *menu bar* pada *front panel* dilakukan untuk melakukan seting awalan yang terlihat pada Gambar 1.2.



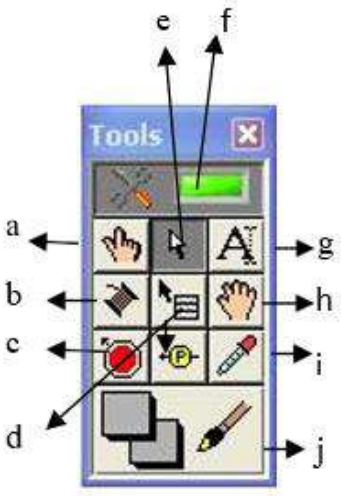
Gambar 1.2 Menu bar pada *front panel.*

Fungsi dari masing-masing bagian adalah:

a. *Run* : Mengeksekusi VI sampai process selesai.
b. *Run continuously* : Mengeksekusi VI secara kontinu, setelah satu proses selesai maka VI kembali dieksekusi sampai abort ditekan.
c. *Abort* : Menghentikan eksekusi.
d. *Pause* : Menghentikan eksekusi sementara
e. *Highlight* : Melihat alur dari jalan program secara berlahan pada *front* panel.
f. *Start single stepping* : Mengeksekusi VI per step.
g. *Align object* : Mengatur tampilan objek.
h. *Distribute objek* : Mengatur tampilan beberapa objek.
i. *Recorder* : Mengatur tampilan beberapa objek yang saling bertumpukan.
j. *Icon* : Gambar yang ditampilkan VI tersebut bila dijadikan sub VI.

1.5.2 *Tool Pallete*

Tool pallete dikeluarkan dengan cara meng-klik kanan pada *front panel/diagram block* yang terlihat dalam Gambar 1.3.



Gambar 1.3 *Tool pallete* .

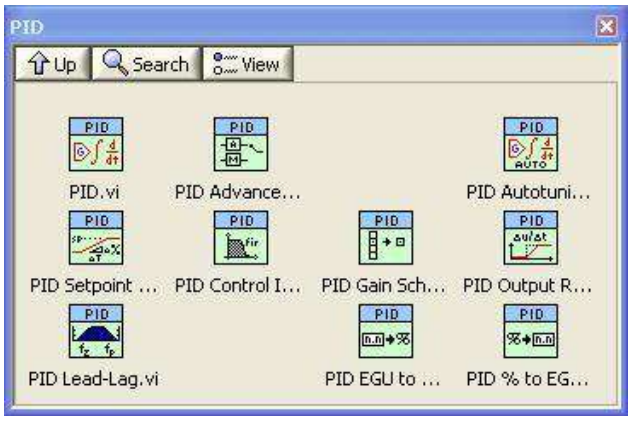
Fungsi icon pada *tool pallete* sebagai berikut:

a. *Operate value* : Mengubah nilai parameter dari suatu objek.
b. *Connect wire* : Menghubungkan beberapa objek dengan kabel.
c. *Set/clear breakpoint* : Membuat atau menghilangkan sebuah breakpoint.
d. *Probe data* : Membuat sebuah probe yang berfungsi untuk monitoring data.
e. *Object Pop Up* : Menu yang berhubungkan dengan objek tersebut tersebut atau memunculkan daftar objek.
f. *Position /size/select* : Memindahkan, mengubahkan ukuran atau memilih suatu objek.
g. *Edit text* : Mengedit atau membuat tulisan.
h. *Scroll window* : Memindahkan sudut pandangan pada layar
i. *Get color* : Mengambil sampel warna.
j. *Set color* : Mengubah warna dan suatu objek.

1.5.3 *Control Palette*

Dalam pemrograman berbasis grafis, hal yang perlu dilakukan untuk membuat suata program adalah menaruh beberapa fungsi dan kemudian menghubungkan dengan kabel pada bagian diagram. Fungsi-fungsi tersebut terletak pada *control palette*. Banyak fungsi yang terletak dalam *control palette* bervariasi tergantung pada seberapa lengkap *LabView* yang diinstall. Pada praktikum dapat menggunakan

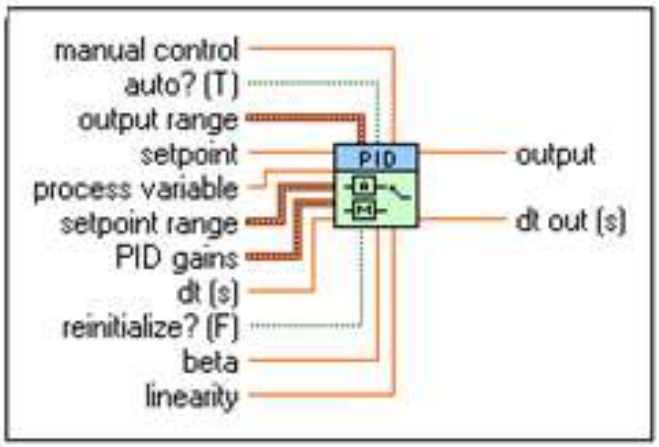
control & simulation. *PID palette* yang digunakan untuk membuat kontrol *PID* pada *front panel LabVIEW* disajikan pada Gambar 1.4 dan Gambar 1.5.



Gambar 1.4 *Control panel* PID *LabView*[4].

.

Dalam menggunakan *PID advanced*, sebagai kontroler *fungsi PID* lanjut (*PIDAdvanced*) menerapkan sebuah fungsi kontroller *PID* dalam bentuk penjumlahan, contohnya adalah *P*, *I* dan *D* dijumlahkan, dengan parameter pengendalian *Kc*, *Ti* dan *Td* (Katshuhiko, 2002). (Lebih tepatnya, bentuk penjumlahan akan membentuk sebuah fungsi pengendali *PID* ideal. Bentuk penjumlahan lain adalah betuk parallel dengan parameter pengendali *Kc*, *Ki = Kc/Ti* dan *Kd = (Kc*Td*). Fungsi pengendali *PID* ini menerapkan *anti-wind up* dan dapat dipasang pada modus manual. Pilihan yang tersedia adalah gain yang *non-linier*, dan bobot setpoint yang dikurangi dalam jumlah yang proporsional. Fungsi pengendali ini tidak memiliki filter *low pass* pada penggunaan derivatif. (Pada penerapan, fungsi pengendali lanjut selalu digunakan, dikarenakan oleh kurangnya pilihan mode manual).



Gambar 1.5 PID *Advanced* (DBL).

Keterangan tentang fungsi pada parameter pada *PID advenced* sebagai berikut.

a. ***Manual Control*** menentukan besarnya keluaran kontrol disaat **"*auto*?"** adalah *FALSE*.

b. ***auto*** menentukan apakah kontrol manual atau otomatis yang dipakai. Saat **"auto?"** adalah *FALSE*, maka VI ini menggunakan kontrol manual. VI ini menggunakan perpindahan dari kontrol mode manual menuju mode otomatis. Pada kondisi default harganya adalah "true".

c. ***Output range*** menentukan rentang yang memaksa keluaran dari kontrol. Rentang normalnya adalah -100 sampai 100.

   1) ***Output High*** menentukan harga maksimum dari keluaran kontroller, normalnya adalah 100.

   2) ***Output Low*** menentukan harga minimum dari keluaran kontroller, normalnya adalah -100.

d. ***Setpoint*** menentukan harga dari Setpoint atau harga yang diinginkan dari variabel proses yang sedang dikendalikan.

e. ***Process Variable*** menentukan harga yang diinginkan dari variabel proses yang dikendalikan. Harga ini setara dengan harga umpan balik dari lup kendali umpan balik.

f. ***Setpoint Range*** menentukan harga maksimum dan minimum dari rentang variabel proses dan setpoint. VI ini menggunkan rentang setpoint untuk menghitung aksi integral non-linier. Harga normal berkisar di 0-100.

   1) ***Setpoint Low*** menentukan harga minimum dari rentang proses variabel dan setpoint.

   2) ***Setpoint High*** menentukan harga maksimum dari rentang variable proses atau setpoint.

g. ***PID Gains*** menentukan parameter gain proporsional, waktu integral dan waktu derivatif dari kontroller.

   1) ***Proportional Gain (Kc)*** menentukan gain proporsional dari kontroller. Harga defaultnya 1. Pada persamaan yang menentukan

kontroller PID, Kc merepresentasikan dari gain proporsional.

2) ***Integral Time*** **(Ti, min)** menentukan waktu integral dalam menit. Harga default adalah 0.01

3) ***Derivative Time*** **(Td, min)** menentukan waktu derivatif dalam menit. Harga default adalah 0.

h. **dt (s)** menentukan interval dalam detik, dimana VI ini disebut. Jika dt (s) kurang dari atau setara dengan 0, VI ini menghitung waktu sejak ini terakhir disebut menggunakan timer internal dengan resolusi milisekon. Harga default adalah 1.

i. ***Reinitialize*****?**, menentukan apakah parameter internal harus dikenali ulang, seperti error integrasi dari kontroller. Harga default adalah *FALSE*.

j. **Beta** menentukan penekanan relatif dari penolakan gangguan (*disturbance rejection*) sampai ke penelusuran setpoint (*setpoint tracking*). Harga default adalah 1 untuk kebanyakan aplikasi. Harga diantara 0 sampai 1 bisa digunakan untuk menentukan penekanan pada penolakan gangguan, seperti perubahan beban dari proses.

k. **Linearity** menentukan linearitas dari respon error. Harga yang valid dari linearitas adalah dari 0-1. Harga dari satu memberikan respon linier yang normal, dimana harga sebesar 0.1 memberikan sebuah respon parabolik.

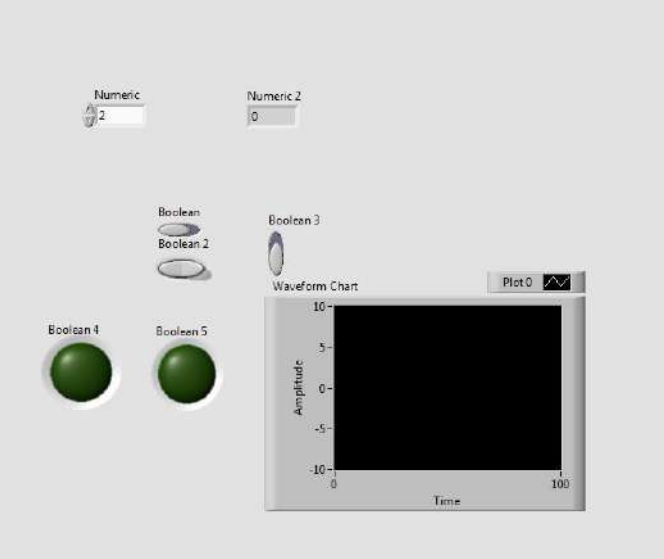
l. ***Output*** mengembalikan keluaran kontrol dari algoritma PID yang diterapkan pada proses kontrol.

m. ***dt out*** **(s)** mengembalikan rentang waktu aktual dalam detik. dt out (s) mengembalikan harga dari dt (s) atau rentang terhitung jika dt (s) dipasang pada harga -1.

1.6. *ICON -ICON PADA SOFTWARE LABVIEW*

1.6.1. Mengenal *Icon – Icon* pada *Software LabVIEW*

Icon-icon pada *Software LabVIEW* ada beberapa bagian, diantaranya pada kontrol panel, diagram board, menu bar dan lain-lain. Langkah awal keluar icon-icon pada kontrol panel sesuai arahan instruktur praktik seperti pada Gambar 1.6.



Gambar 1. 6 *Icon – icon LabVIEW* pada kontrol panel.

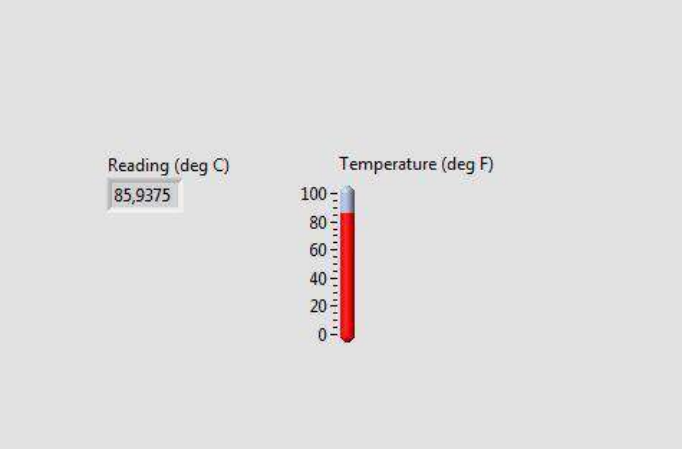
1.6.2. Membuat Sistem Pemrograman.

Tahap berikutnya membuat pemrograman grafis yang disajikam pada Gambar 1.7 sampai Gambar 1.63. adapun pemrogram grafis dibuat secara bertahap dari point a sampai bb.

a. Membuat *sistem thermometer.*

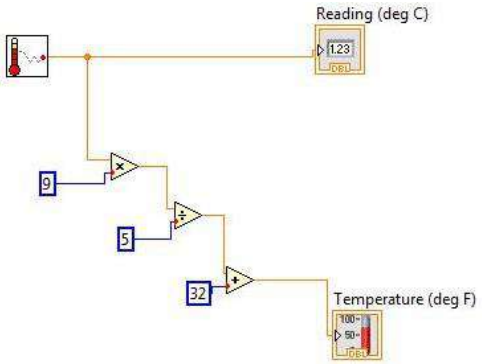
1) Rumus :

Celcius = 9/5 + 32°F ..... (1)



Gambar 1.7 Tampilan *thermometer.*

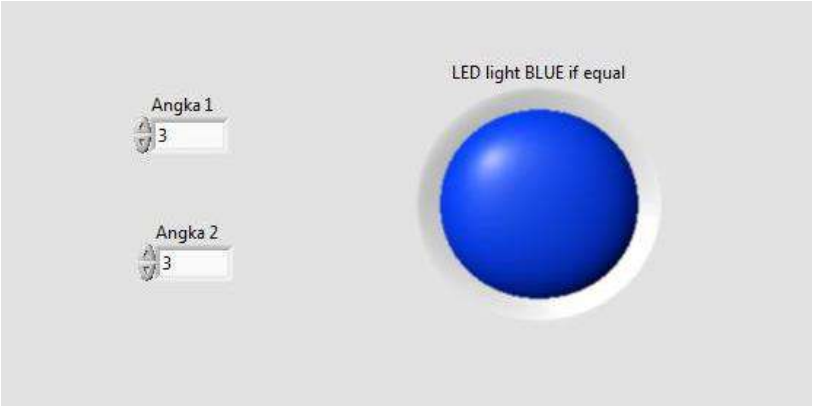
2) *Thermometer Programming*.



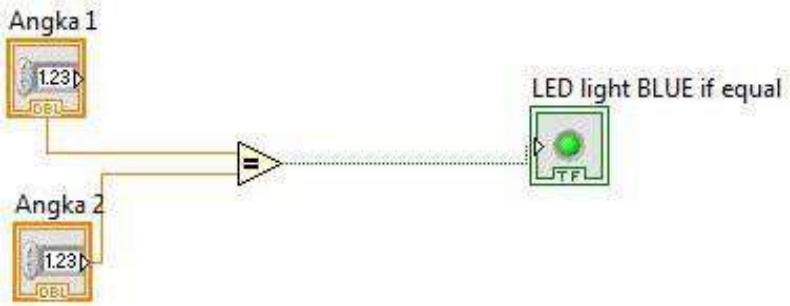
Gambar 1.8 *Thermometer Programming.*

b. *Comparison practise*

- *Indicator lamp*.

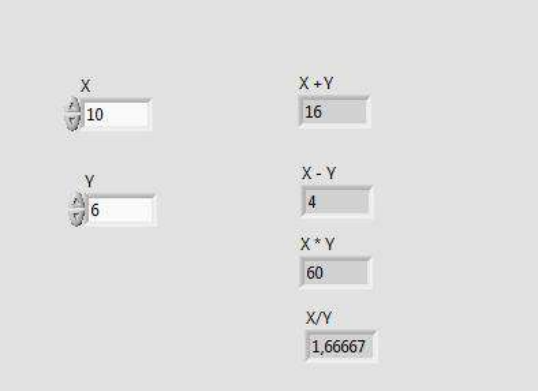


Gambar 1.9 *Indicator lamp*.

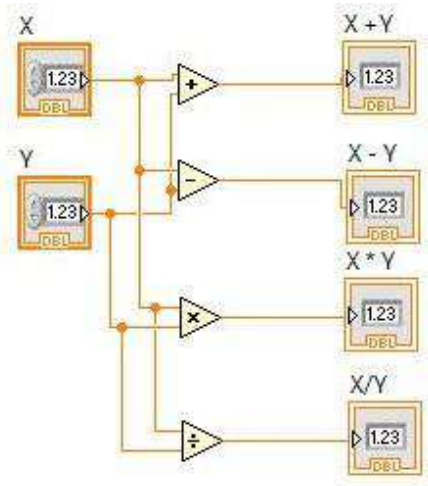


Gambar 1.10 *Indicator Lamp Programming.*

c. *Calculator*

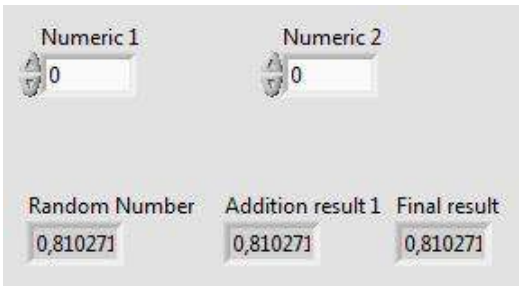


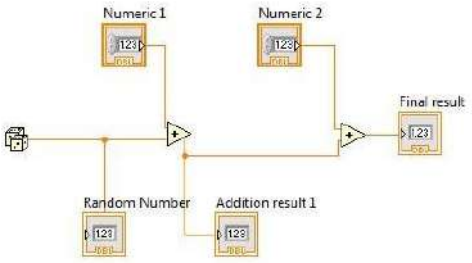
Gambar 1.11 Template visual.



Gambar 1.12 *Programming Control*.

d. *Debugging Callenge*



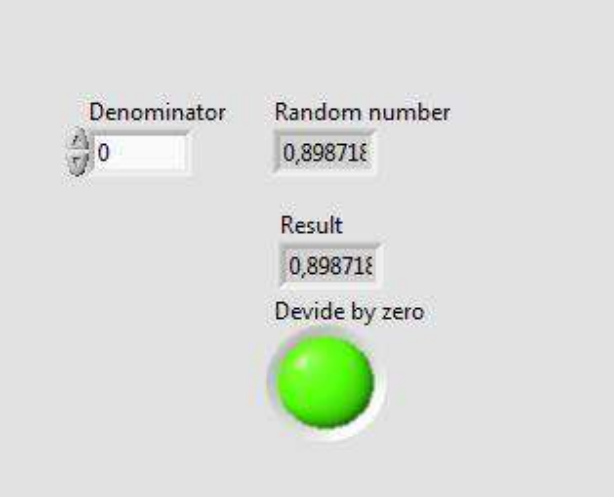


Gambar 1.13 *Numeric regulation* dan *programming control*.

e. *Find Average*

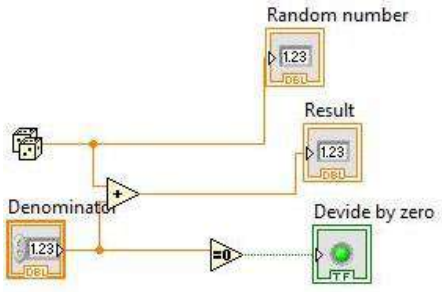
Gambar 1.14 *Average number.*

f. *Devide By Zero*



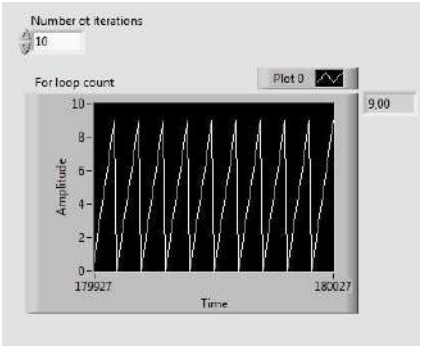
Gambar 1.15 *Devide by zero aplication.*

.

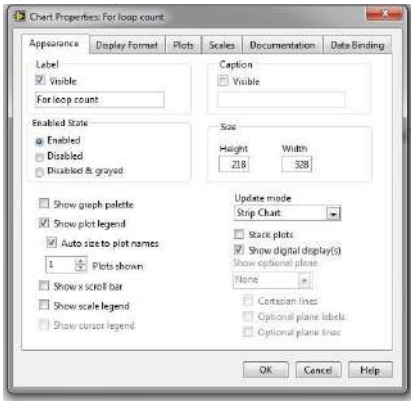


Gambar 1.16 *Devide by zero programming*.

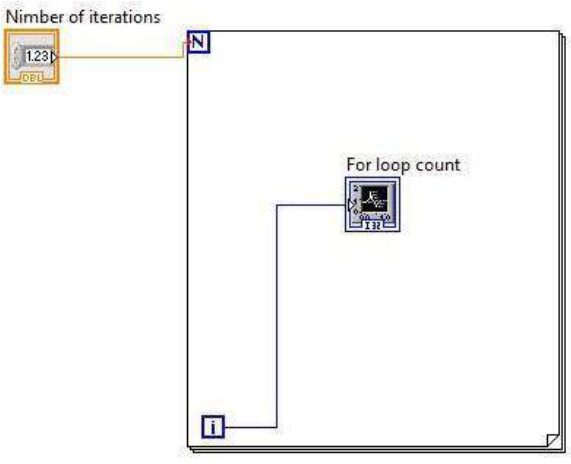
g. *Counting Loops 1*



Gambar 1.17 *Counting loops 1*.

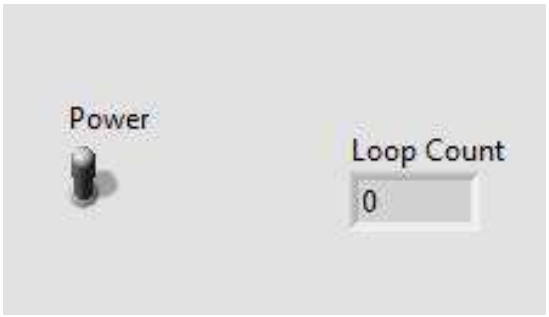


Gambar 1.18 Set-up.

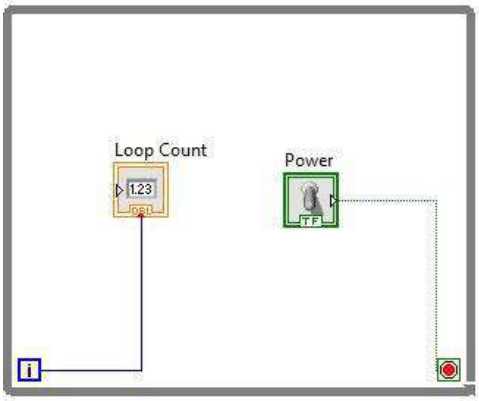


Gambar 1.19 *Loops programming.*

h. *Counting loops 2*

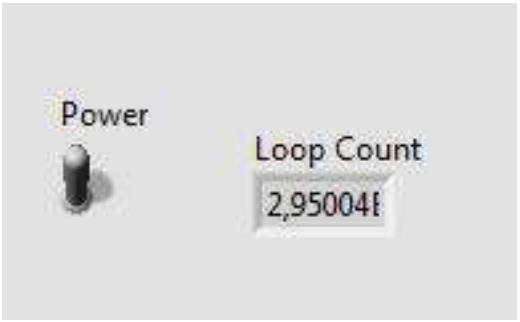


Gambar 1.20 *Counting Loops 2.*

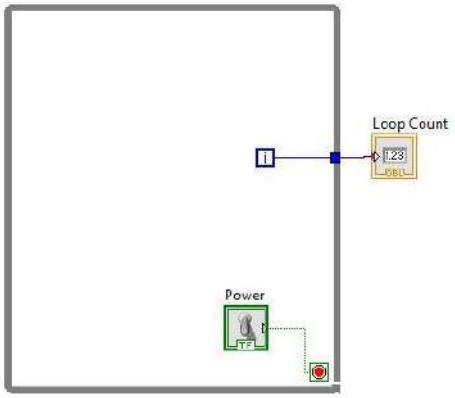


Gambar 1.21 *Counting Loops 2 Programming.*

i. *Counting loops 3*



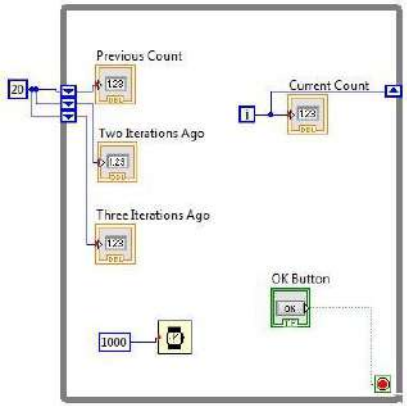
Gambar 1.22 *Counting Loops 3.*



Gambar 1.23 *Counting Loops 3 Programming.*

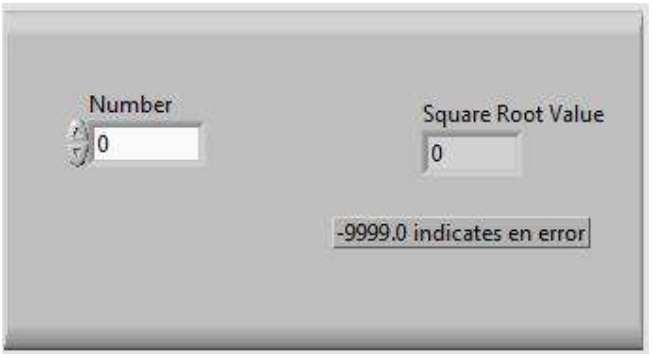
j. *Shift Register 1*

Gambar 1. 24 *Shift Register* 1.

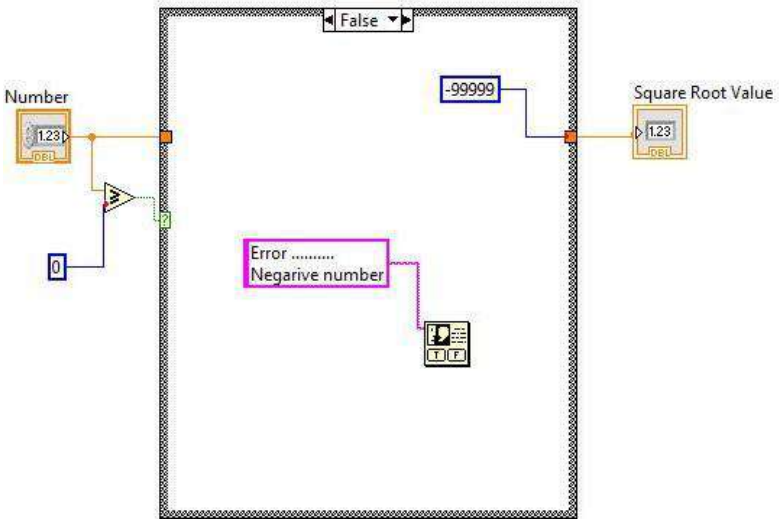


Gambar 1. 25 *Shift Register 1 Programming*.

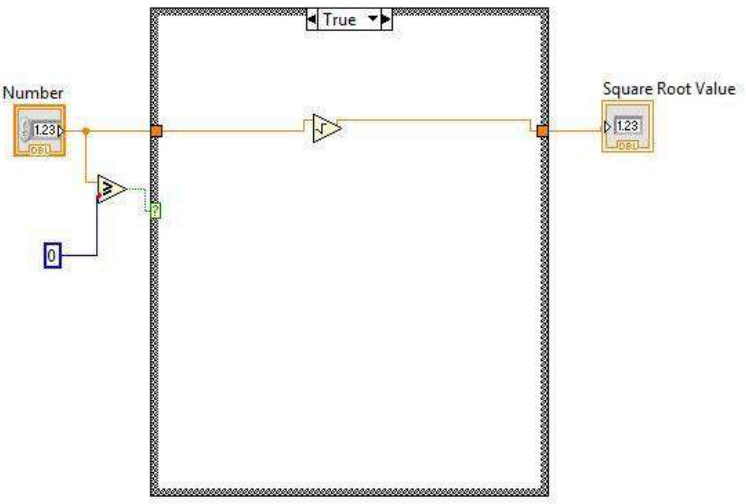
k. *Square Root*



Gambar 1.26 *Icon square root.*

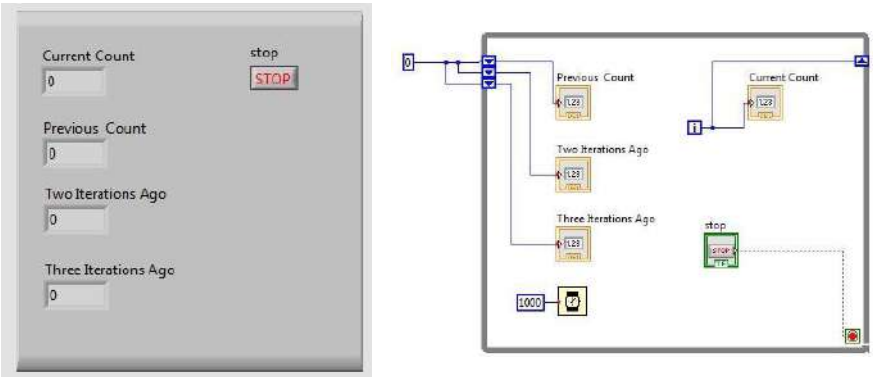


Gambar 1.27 *Roots system.*



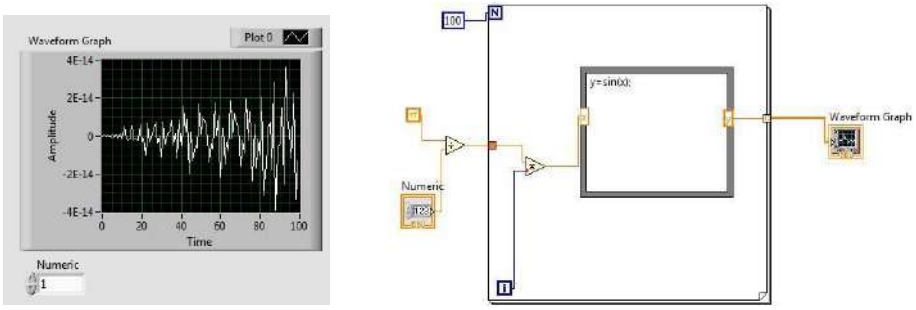
Gambar 1.28 *Roots system 2.*

l. *Shift Register 2*



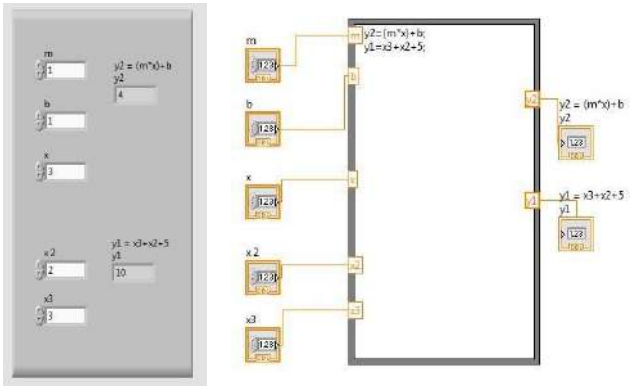
Gambar 1.29 *Logika program shift register 2* (Integrasi penelitian ke dalam praktik)*****.

m. *Formula Fun*



Gambar 1. 30 *Formula logic fun.*

n. *Equation*

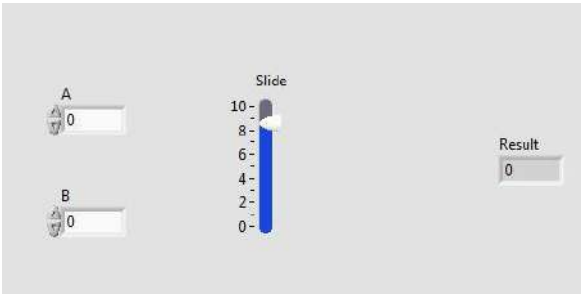


Gambar 1.31 Logika pemograman *equation* .

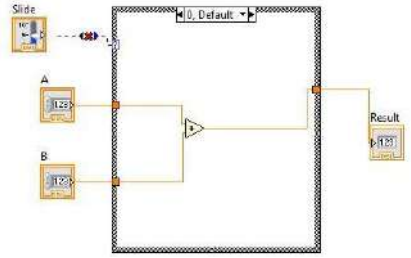
o. ***Calculator 2***

Langkah 1

1) Buat string A dan string B.
2) Buat slide.
3) Munculkan result dengan string.

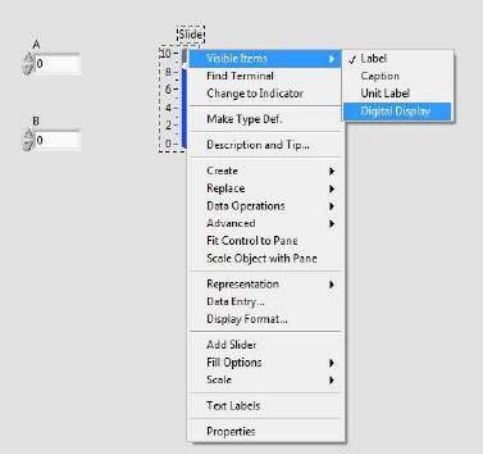


Gambar 1. 32 *Interface panel*.

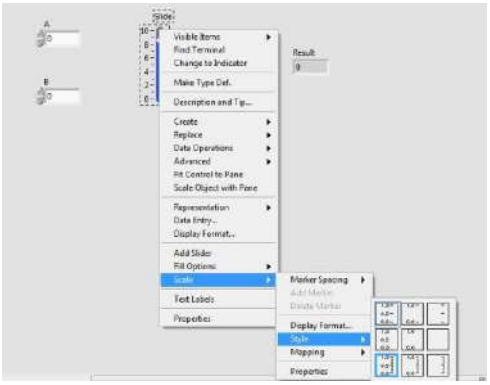


Gambar 1. 33 Logika program 1.

4) Hubungkan logika program *interface.*
5) Munculkan *digital display*.



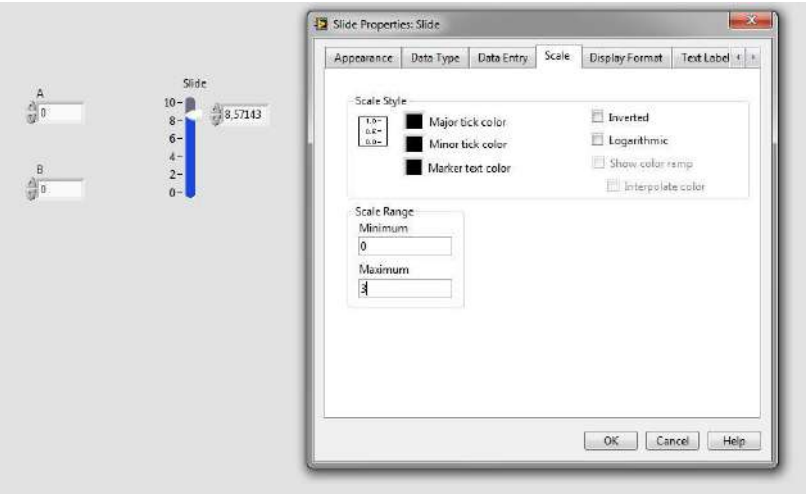
Gambar 1. 34 *Digital display*.



Gambar 1.35 *Setting style.*

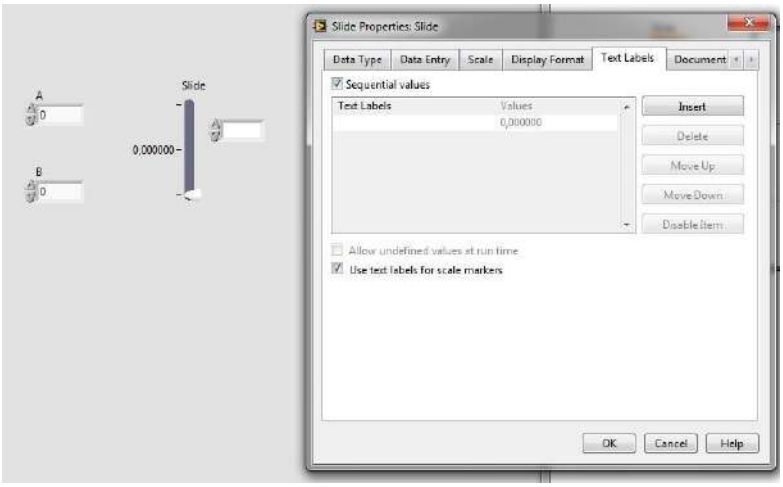
Langkah 2

1) Atur scale properties range mimimum 0 maximum 3



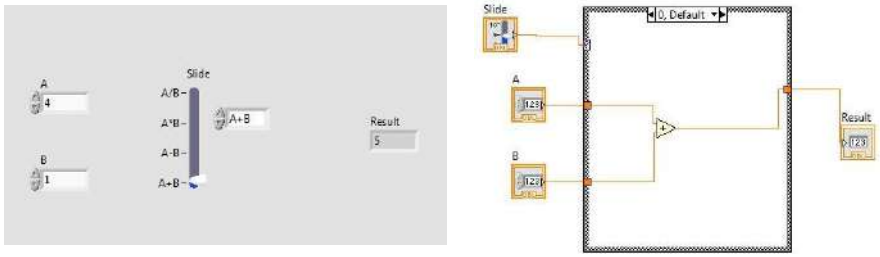
Gambar 1.36 *Properties manager.*

2) Gunakan text label dengan memberi tanda *use text label for make*



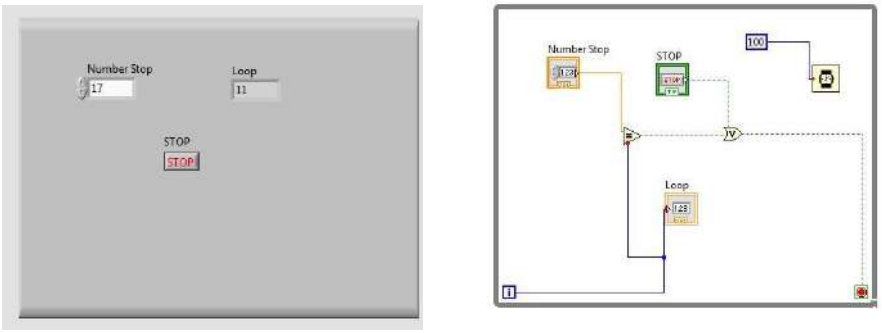
Gambar 1. 37 *Text manager.*

3) Hasil logika program



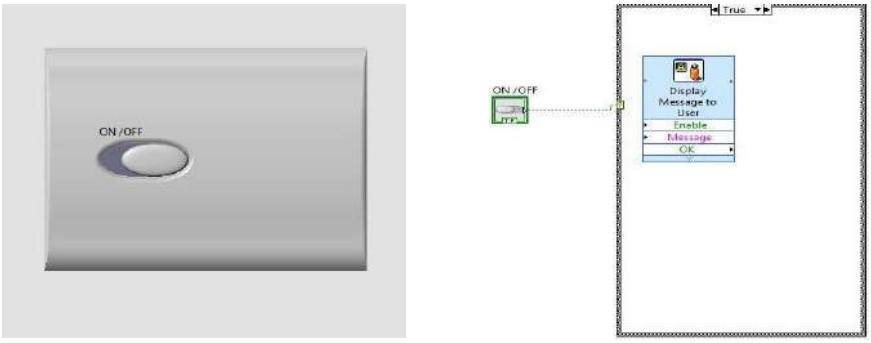
Gambar 1. 38 Logika program kelanjutan.

p. ***Loops System***



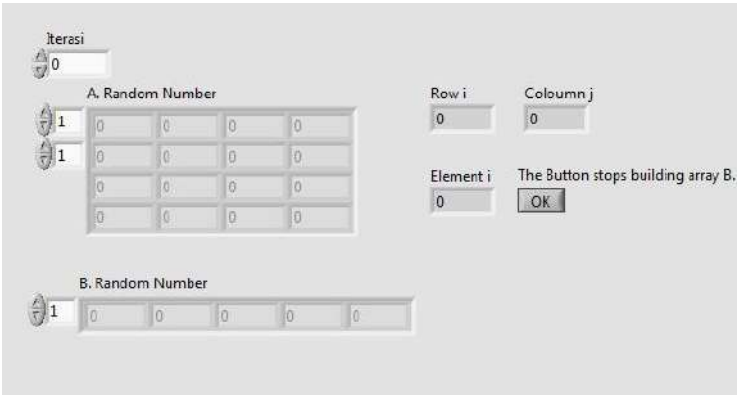
Gambar 1. 39 *Loops system* program.

q. ***Display Digital***

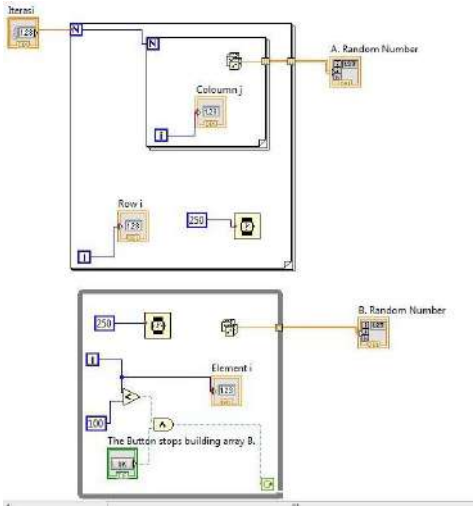


Gambar 1. 40 *Digital display program.*

r. *Building Auto Array Index*

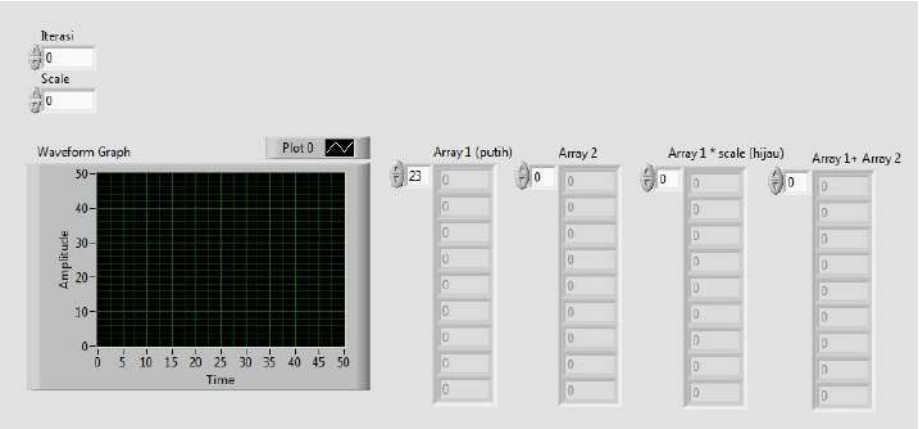


Gambar 1. 41 Panel *Array index*.

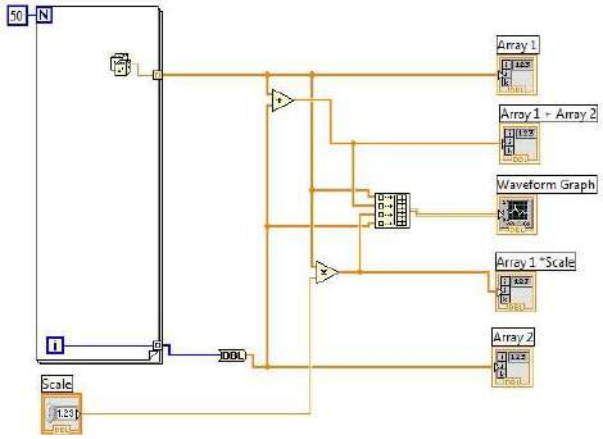


Gambar 1. 42 *Algoritma index*.

s. *Polimorphism Exersize*

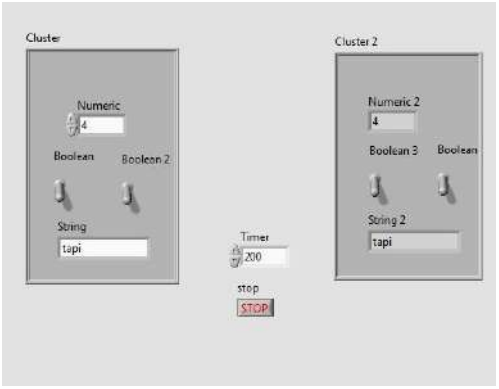


Gambar 1. 43 *Polimorphism display*.

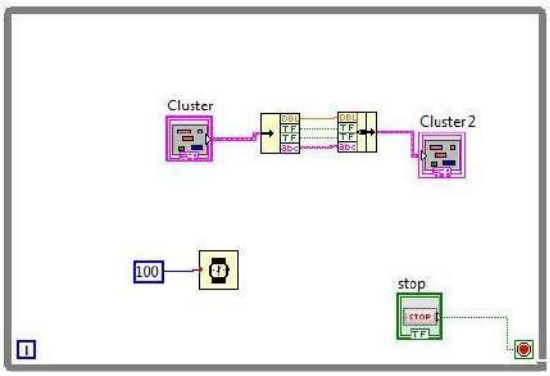


Gambar 1. 44 *Array logaritm squence*.

t. *Clustering Practise*

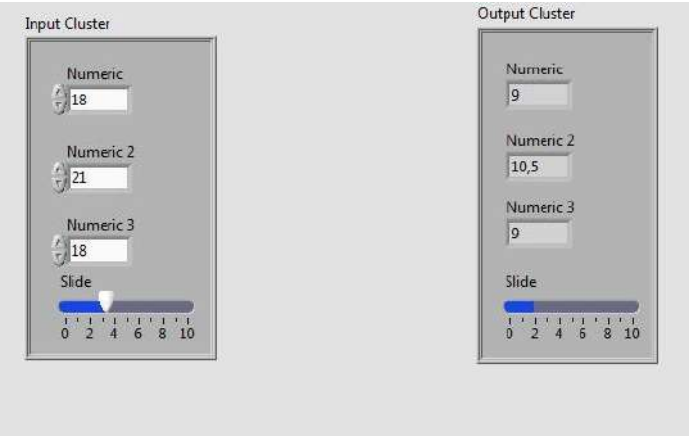


Gambar 1. 45 *Sistem cluster*.

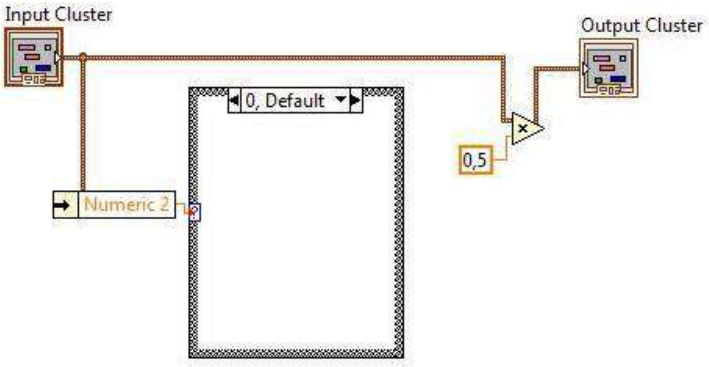


Gambar 1. 46 *Input data cluster unit.*

u. *More Fun With Cluster*

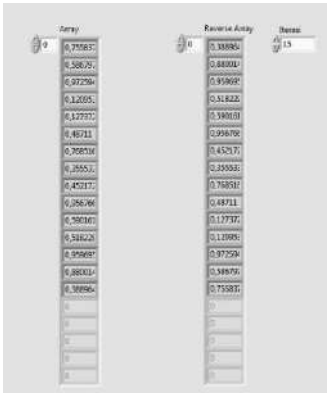


Gambar 1.47 *Input cluster anlys panel*.

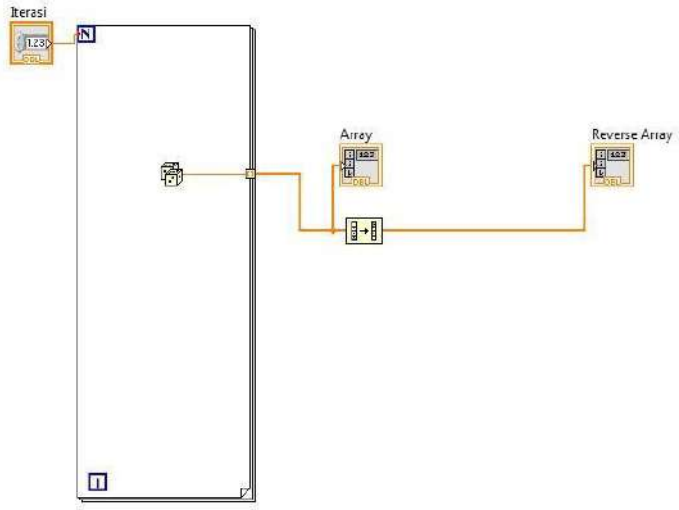


Gambar 1.48 *Including cluster numeric*.

v. *Reverse Order Chalange*

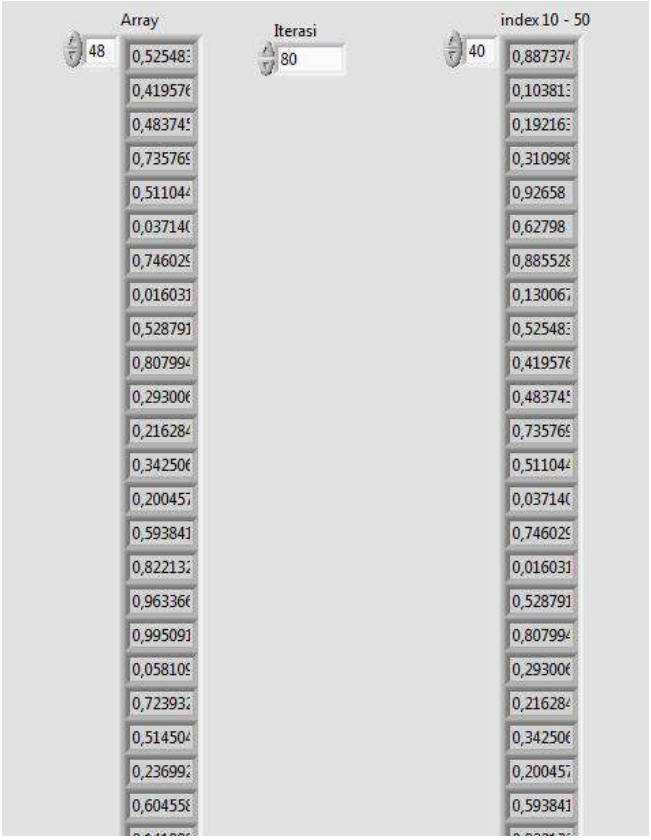


Gambar 1. 49 *Reverse data array*.

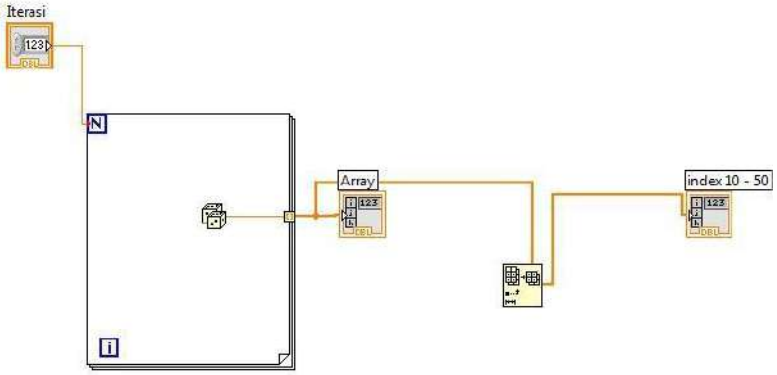


Gambar 1. 50 *Reverse data logic.*

w. *Taking Subset*

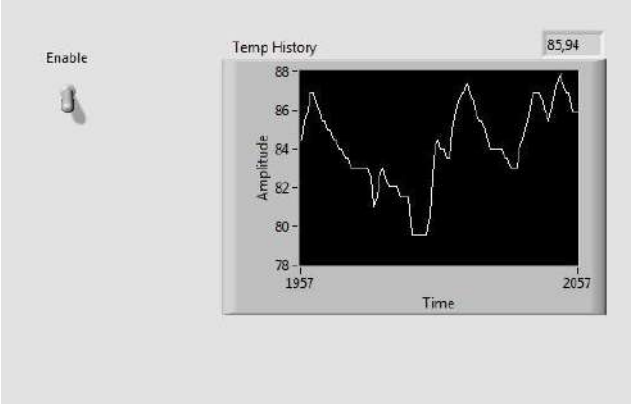


Gambar 1. 51 *Taking subset programming*.

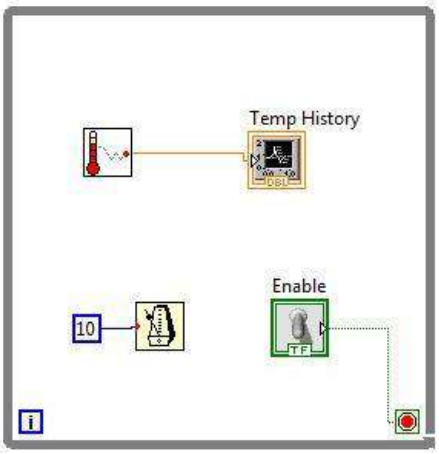


Gambar 1.52 Order *array aragement*.

x. *Temperature Monitor Simulasition*

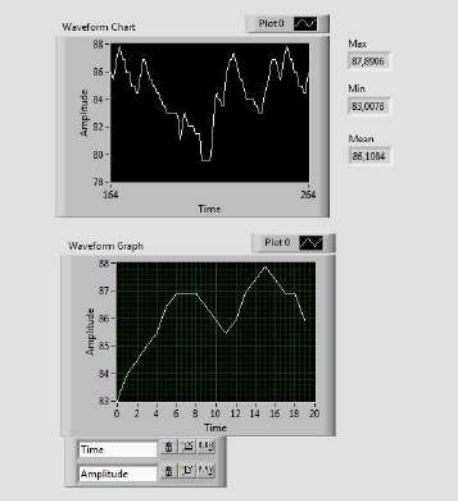


Gambar 1.53 *Display monitor temperature*.



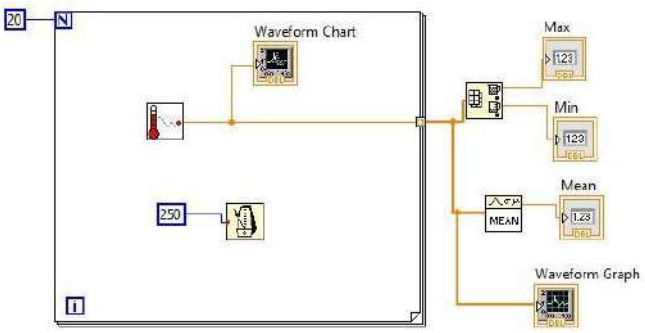
Gambar 1.54 *Temperature loops*.

y. ***Temperature Analysis***



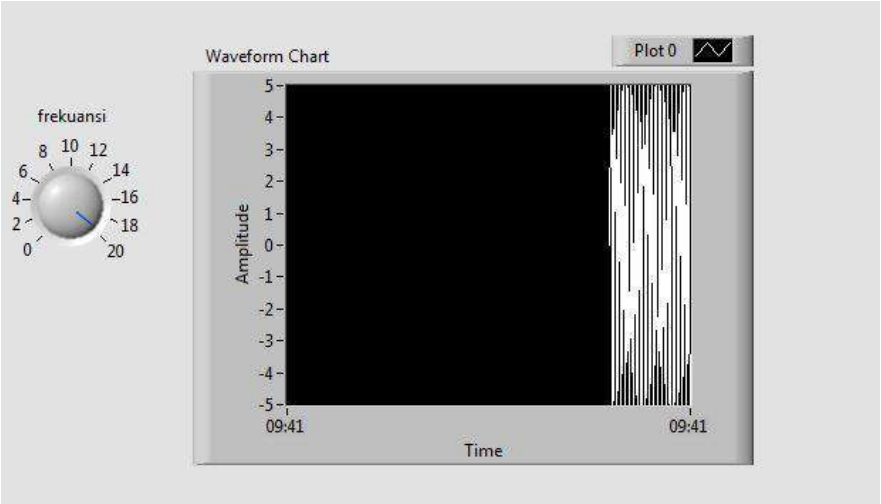
z.

Gambar 1. 55 *Display monitor temperature analysis.*

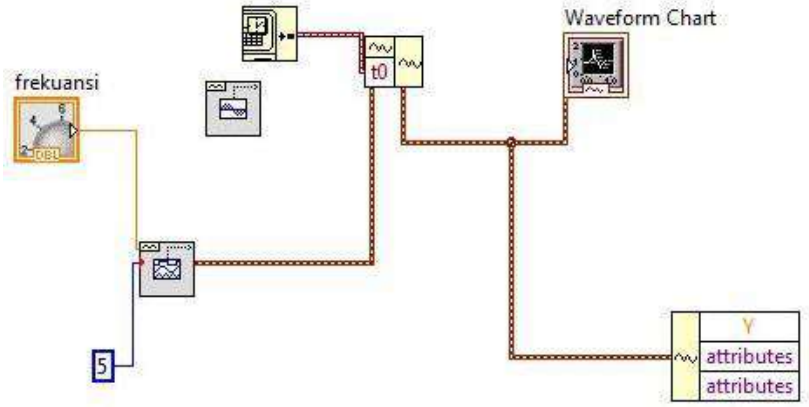


Gambar 1.56 Temperature *loops analysis.*

aa. ***Ganerate and Plot Waveform***

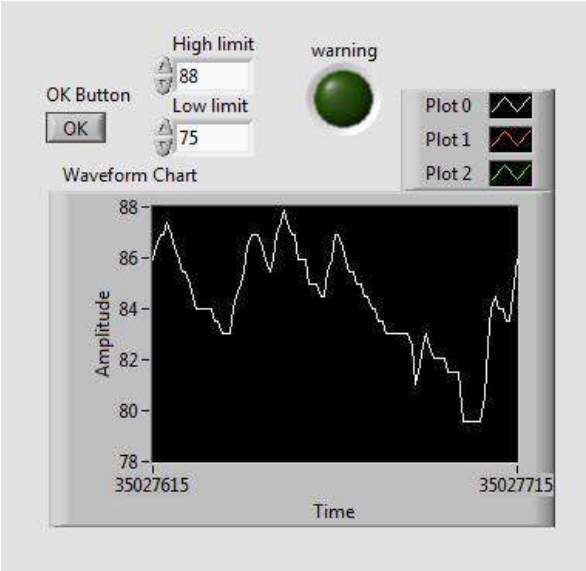


Gambar 1. 57 *Ganerate and plot waveform display*.

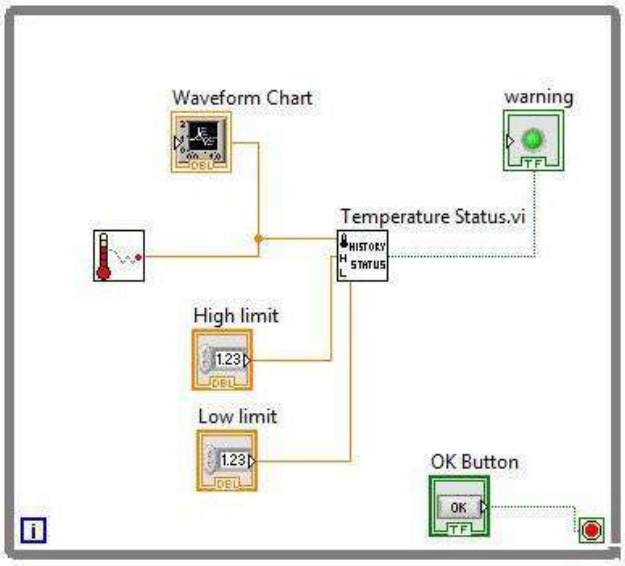


Gambar 1. 58 *Ganerate and plot waveform squence*.

bb. *Temperature Limit*

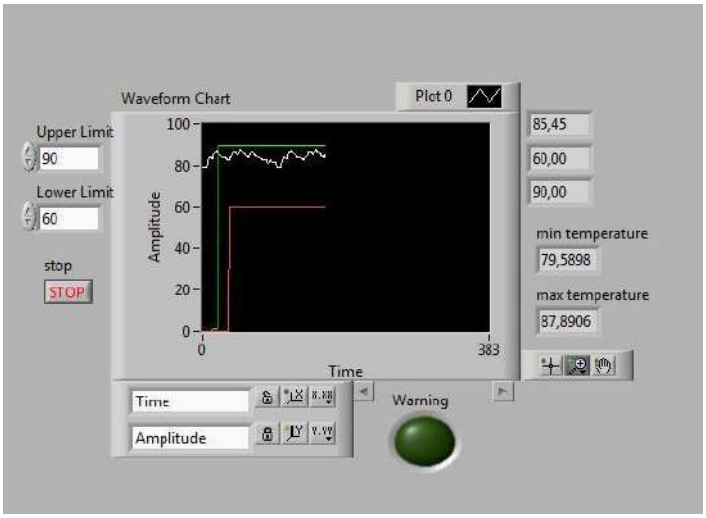


Gambar 1. 59 *Temperature limit programming integrated*.

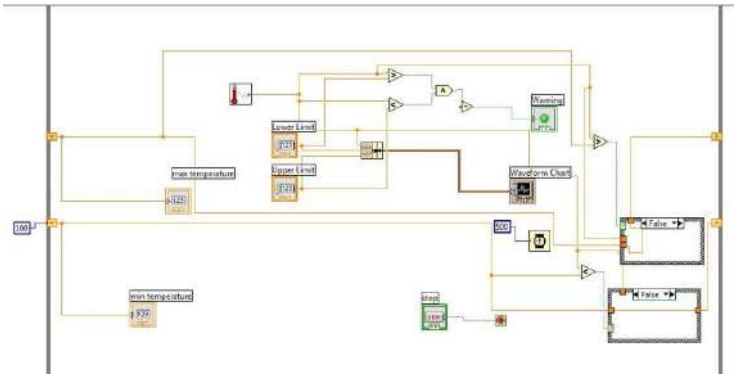


Gambar 1. 60 *Signal processing unit* (Integrasi penelitian ke dalam praktik).******.

cc. *Max Min Temperature Limit*



Gambar 1. 61 *Max min temperature limit*(Integrasi penelitian ke dalam praktik).*******.



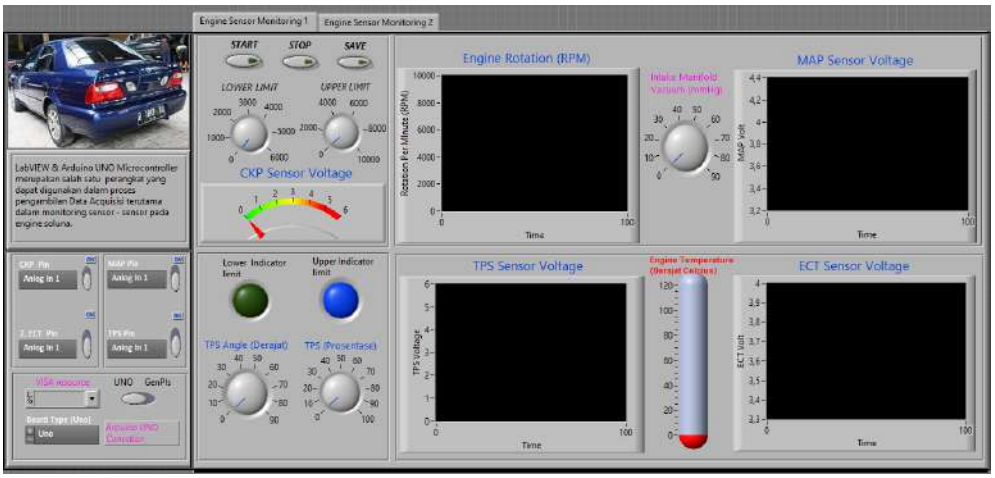
Gambar 1. 62 *Max min temperature limit algoritm*.

1.7. KONEKSI *SENSOR KE LABVIEW FOR EVERYONE*

Setelah merancang pemrogram grafis dengan *software LabVIEW* langkah selanjutnya melakukan koneksi antara sensor, *microcontroller* dan *software LabVIEW* untuk memonitoring *signal* yang dibangkitkan oleh sensor pada kendaraan. Langkah koneksi antara sensor, *microcontroller* dan *software LabVIEW* dibahas pada modul praktek tersendiri.

1.8. TUGAS

- Buatlah monitoring signal seperti pada Gambar 1.63 yang dilengkapi dengan pemrograman dengan *front panel* dan *diagram block*!



Gambar 1.63 Sistem monitoring signal sensor TPS , MAP, ECT dan *engine speed* pada kendaraan(Integrasi penelitian ke dalam praktik).*

BAB 2. KEGIATAN BELAJAR SUB CPMK 2

2.1 TARGET CPL MATA KULIAH

SUB CPMK	TUJUAN
AM - 02	Menguasai konsep dan mampu menerapkan *control systems* pada kendaraan.

2.2 TARGET PEMBELAJARAN PRAKTEK

SUB CPMK	TUJUAN PEMBELAJARAN PRAKTEK
AM - 02	Mampu menerapkan *control systems* pada kendaraan

2.3 URAIAN

Target pembelajaran praktek pada sub CPMK AM-02 fokus pada pencapian untuk mampu menerapkan *control system* pada kendaraan. Untuk mencapai target ini ada beberapa langkah yang harus dilakukan:

- Melakukan identifikasi *control system* pada kendaraan.
- Merancang *diagram block* pada *control system* yang ada dalam kendaraan.
- Membuat hubungan *input, plant, output* dan sistem *mapping* pada *control system* kendaraan.
- Merancang *loops* pada *control system* pada kendaraan.
- Mencoba membuat *loops* pada *control system* pada kendaraan.

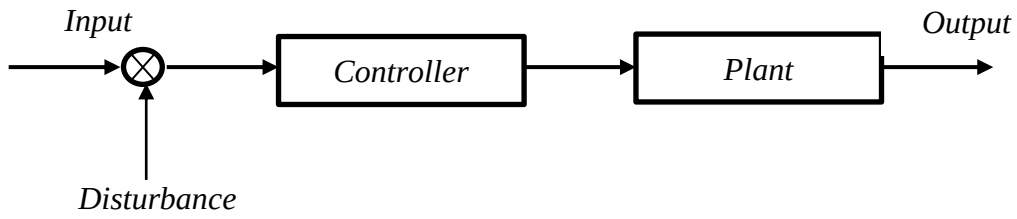
2.4 ALAT DAN BAHAN PRAKTEK

No.	Deskripsi	Item	Keterangan
1.	*Software power point/Microsoft Visio/word*	1 Paket	National Instrument
2.	Komputer	24 unit	Ram 4/6/8 Giga

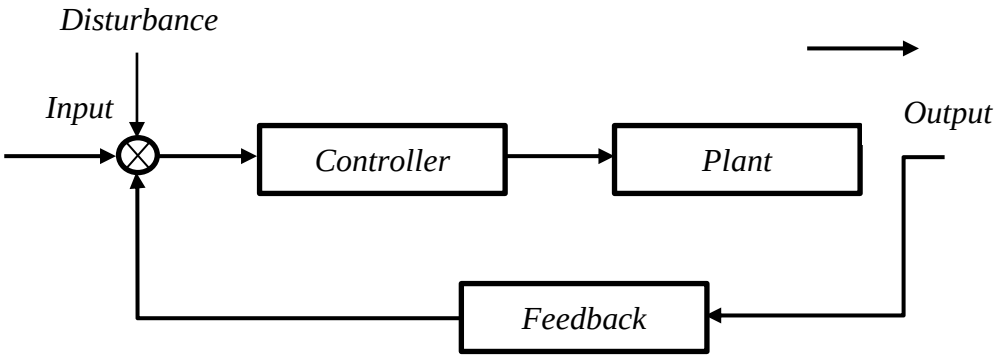
2.5 SKEMA *LOOPS* PADA *CONTROL SYSTEM* KENDARAAN

Skema *loops* ini digunakan untuk memodelkan *control system* yang digunakan sebagai acuan perancangan. Masing – masing loops memiliki karakter yang berbeda, untuk itu perlu melakukan identifikasi control system yang dirancang. Loops pada control system ada dua macam, yaitu *open loops control*

system dan *closed loops control system* yang disajikan pada Gambar 2.1 dan Gambar 2.2.



Gambar 2.1 *Loops* dari *open control system.*



Gambar 2.2 Loops dari *closed control system*.

2.6 TUGAS

Buatlah *loops* dari masing-masing *control system* pada kendaraan baik pada *control system* pada *engine, drivetrain*, sistem *supporting* maupun *body electrical*!

BAB 3. KEGIATAN BELAJAR SUB CPMK 3

3.1. TARGET CPL MATA KULIAH

SUB CPMK	TUJUAN
AM - 03	Menguasai konsep dan mampu menerapkan *microcontroller* pada kendaraan

3.2. TARGET PEMBELAJARAN PRAKTEK

SUB CPMK	TUJUAN PEMBELAJARAN PRAKTEK
AM - 03	Mampu menerapkan *microcontroller* pada kendaraan

3.3. URAIAN

Target pembelajaran praktek pada sub CPMK AM-03 fokus pada pencapian untuk mampu menerapkan *microcontroller* pada kendaraan. Untuk mencapai target ini ada beberapa langkah yang harus dilakukan:

- Merancang pemrograman yang dapat di-*embedded*-kan pada *Central Processing Unit- CPU* pada sebuah *microcontroller* untuk mengendalikan sistem yang akan dikontrol.
- Merancang sistem bahasa pemrogram meliputi Bahasa C, Bahasa C++, bahasa Grafis dan Bahasa Matematis secara simulasi dan realtime. Adapun untuk merancang pemrograman ini menggunakan *Software Integrate Development Enviroment* (IDE) yang dikeluarkan oleh perusahaan pengembang *microcontroller* tipe mega (Arduino).

3.4. ALAT DAN BAHAN PRAKTEK

No.	Deskripsi	Item	Keterangan
1.	Software IDE Arduino	1 Paket	National Instrument
2.	Komputer	24 unit	Ram 4/6/8 Giga
4.	Kabel	1 set	Jumper, NYY/NYA
5.	Microcontroller	1 set	Mega 2560, Mega 32 dan lain - lain

3.5. PEMROGRAMAN BAHASA C

3.5.1. Struktur Dasar Bahasa C

Dalam bahasa C memiliki struktur dasar yang menjadi fundamental algoritma sebagai berikut:

a. *Incude*

Struktur ini digunakan memasukkan beberapa fungsi yang ada dalam *header file.*

Header file sering digunakan untuk mendefinisikan fungsi yang sudah dirancang. *Header file* akan membantu Bahasa C mengenali pemrograman yang dibuat. File tersebut memiliki ektensi (.h), misalnya : stdio.h. file stdio.h sudah terdapat dalam computer pada saat melakukan intalasi gcc.

b. Fungsi *main*()

Fungsi main() merupakan bagian utama dari struktur pemrograman bahasa C. Pada saat bahasa C mulai beroperasi fungsi main() akan mengalami eksekusi program pertama kali.

Contoh :

```
Int main() {
//…….
…….
Return 0
}
```

Fungsi `int` yang berada pada sebelah kiri `main()` sebagai data akan dikembalikan saat program selesai. Dalam penulisan wajib menuliskan `return 0`, karena `main()` akan memulangkan nilai `0` jika selesai eksekusi. Selain int dapat juga menggunakan `void`. `Void` memiliki makna kosong, sehingga penggunaan `void`, maka tidak perlu lagi menambahkan `return` di akhir fungsi.

c. *Statement*

Statement diartikan sebagai perintah untuk melakukan sebuah eksekusi pemrograman.

Contoh : `printf (" Selamat Pagi");`

Artinya pengguna memerintahkan komputer untuk menampilkan tulisan `Selamat Pagi` ke *consule.*

d. *Blok Kode*

Blok kode merupakan rangkaian statement yang ditulisan dalam pemrograman bahasa C. Blok kode ditulis dengan kurung kurawal `{ ..... }`.

Contoh :

```
If ( A=x) {
// tulisan blok kode
// sampai akhir kurung kurawal
}
```

Penulisan blok kode diantaranya `if, for, while, do/while, fungsi` dan lain –lain. Untuk membuat komentar dalam satu baris digunakan (//) sedangkam dalam membuat komentar dalam beberapa garis menggunakan (`/**/`) untuk membuat komentar beberapa baris.

Contoh :

```
# include <stdio.h>
Int main () {
// satu baris ditulisakan komentar
Prinf(" Selamat Pagi");

/*
Ini komentar
lebih
nyaman.
*/return 0;
}
```

e. *Case Sensitive*

Case sensitive merupakan penulisan bahasa C yang memperhatikan tentang besar huruf dalam statement. Seperti pada penulisan fungsi `suhu` dengan `Suhu` dengan huruf capital/tidak di bawah ini memiliki variable yang berbeda.

```
// Penulisan variable
```

```
String suhu = " temperature"
String Suhu = " panas"
```

3.5.2. Mengenal *Output* Dan *Input* Bahasa C

Dalam sistem pengendali dikenal istilah input, sistem kontrol dan output. Input digunakan sebagai masukan yang dibutuhkan ke dalam suatu program *embedded*. Sistem kontrol berfungsi sebagai suatu device yang melakukan pengambilan keputusan. Output merupakan luaran yang dihasilkan oleh sistem kontrol. Input yang dapat sebagai masukan dari sistem kontrol dapat berupa microphone, kamera, keyboard, sensor dan lain – lain.

a. Fungsi Input Pada Bahasa C

1) Fungsi *scanf()*

   Fungsi `scanf()` sebagai fungsi yang digunakan untuk mengambil inputan data dari keyborad.

2) Fungsi gets()

   Fungsi `gets()` digunakan sebagai device mengambil input dalam satu baris. fungsi gets() memiliki kelemahan dapat mengakibatkan masalah pada *buffer overflow* pada program.

3) Fungsi fgets()

   Fungsi `fgets()` digunakan sebagai device mengambil input dalam satu baris namun fungsi ini dapat menentukan ukuran buffer dan sumber inputan, sehingga lebih aman jika dibandingkan dengan fungsi `gets()`.

b. Fungsi Output Pada Bahasa C

1) Fungsi printf()

   Fungsi `printf()` digunakan untuk menampilkan data ke layar monitor. Fungsi ini berasal library `stdio.h`. oleh sebab itu perlu menuliskan `#include <stdio.h>` dalam awal pemrograman.

   Contoh program fungsi `print()`:

```
#include <stdio.h>
Int main(){
printf(" my name, surahman");
```

```
printf(" I am fisherman %\n',” I am hungry”);
return 0;
}
```

Fungsi `printf()` perlu menggunakan simbol `%d`, `%s`, dan `\n` untuk format teks.

`\n` sebagai simbol untuk memunculkan atau membuat baris baru.

`%s` untuk menampilkan angka atau menampilkan bilangan desimal

`%d` untuk menampilkan data string.

2) **Fungsi puts()**

Fungsi `puts()` digunakan untuk menampilkan output dilayar monitor namun pada fungsi puts tidak memerlukan format dan selalu membuat baris baru. Fungsi ini tidak membutuhkan menggunakan simbol `\n` seperti pada fungsi `printf()`.

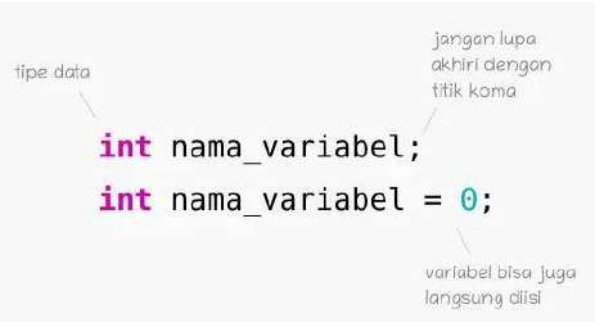
Contoh program fungsi `puts()`

```
#include <stdio.h>

Int main(){
puts(" my name surahman”);
puts (" I am fisherman I am hungry”);
return 0;
}
```

c. Variabel Pada Pemrograman Bahasa C

Format variabel



Contoh program :

```
#include <stdio.h>
int tinggi ;
```

atau

```
#include <stdio.h>
int besar=40 ;
```

Artinya :

Variabel berupa `tinggi` dengan tipe datanya adalah `int`(*integer*) atau variabel `besar` dengan nilai 40 dan tipe data *integer* kemudian diisi dengan nilai `40`. Dalam penulisan variable harus diakhiri dengan titik koma. Ketentuan – ketentuan penulisan variabel dalam pemrograman bahasa C, diantaranya :

1) Variabel tidak diperbolehkan diawali oleh angka maupun simbol.
2) Variabel tidak diperbolehkan menggunakan *keyword* yang ada dalam bahasa C, contoh: `void`, `int`, `if` dan lain lain.
3) Variabel ditulis dengan huruf besar dan kecil memiliki makna perintah yang berbeda, contoh: `tirai` dan `Tirai` adalah dua variabel yang berbeda.
4) Disarankan penulisan *underscore* pada variabel yang berasal dari dua suku kata, contoh: `tirai_bambu`

Contoh penulisan :

Outputnya :

```
Berat badan saya adalah 54 kg dan tinggi 178 cm
```

d. Tipe Data pada Bahasa C

Tipe data pada bahasa C ada delapan, diantaranya :

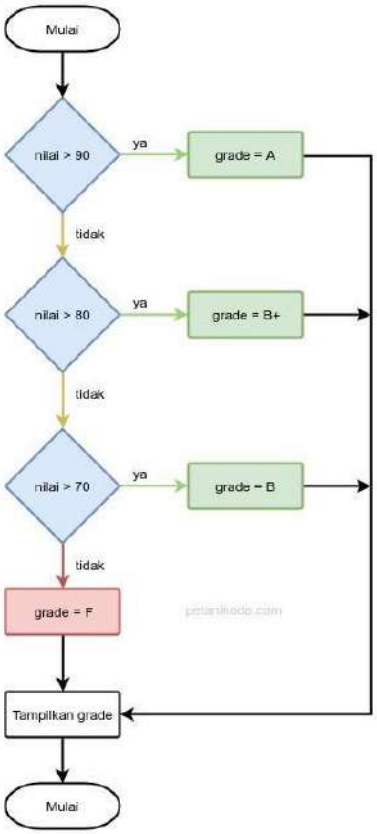
1) *Structure*
2) *Array*
3) *Void*
4) *Char*
5) *Enum*
6) *Integer*
7) *Float*
8) *Pointer*

e. Operator bahasa C

1) Operator Artimatika;
2) Operator Penugasan;
3) Operator Pembanding;
4) Operator Logika;
5) Operator Bitwise;
6) dan Operator Lain-lain.

f. Blok Percabangan

1) Percabangan if
2) Percabangan if/else
3) Percabangan if/else/if



Gambar 3. 1 Logika blok percabangan.

Tabel 2. 1 Deklarasi Data

No	Jenis data		Ukuran Range	Format
1	Char	Char untuk type data karakter	1 byte -128 s/d 127	%c
2	Int	Untuk type data bulat	2 byte -32768 s/d 32767	%i, %d
3	Float	Untuk bilangan pecahan	-3,4E-38 s/d 3.4E +38	%f
4	long int	Untuk bilangan integer yang lebih luas		%ld
5	double	Untuk pecahan lebih luas		%lf

3.5.3. Embeded *Programming*

Latihan pemrograman bahasa C di *microcontroller*

1. Pemrograman *Analog Read Serial*(Integrasi penelitian ke dalam praktik*)***

```
/*
  AnalogReadSerial
  Reads an analog input on pin 0, prints the result to the serial monitor.
  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and ground.

 This example code is in the public domain.
 */

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(1);        // delay in between reads for stability
}
```

2. Program *Blink*

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

3. Program *Digital Read Serial*

```
/*
  DigitalReadSerial
 Reads a digital input on pin 2, prints the result to the serial monitor

 This example code is in the public domain.
 */

// digital pin 2 has a pushbutton attached to it. Give it a name:
int pushButton = 2;

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  // make the pushbutton's pin an input:
  pinMode(pushButton, INPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input pin:
  int buttonState = digitalRead(pushButton);
  // print out the state of the button:
  Serial.println(buttonState);
  delay(1);        // delay in between reads for stability
}
```

```
/*
  ReadAnalogVoltage
  Reads an analog input on pin 0, converts it to voltage, and prints the result to the serial monitor.
  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and ground.

 This example code is in the public domain.
 */

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 5V):
  float voltage = sensorValue * (5.0 / 1023.0);
  // print out the value you read:
  Serial.println(voltage);
}
```

4. *Blink Without Delay*

```
/* Blink without Delay

Turns on and off a light emitting diode(LED) connected to a digital
pin, without using the delay() function.  This means that other code
can run at the same time without being interrupted by the LED code.

The circuit:
* LED attached from pin 13 to ground.
* Note: on most Arduinos, there is already an LED on the board
that's attached to pin 13, so no hardware is needed for this example.


created 2005
by David A. Mellis
modified 8 Feb 2010
by Paul Stoffregen

This example code is in the public domain.

 */

// constants won't change. Used here to
// set pin numbers:
const int ledPin =  13;      // the number of the LED pin

// Variables will change:
int ledState = LOW;             // ledState used to set the LED
long previousMillis = 0;        // will store last time LED was updated

// the follow variables is a long because the time, measured in miliseconds,
// will quickly become a bigger number than can be stored in an int.
long interval = 1000;           // interval at which to blink (milliseconds)

void setup() {
  // set the digital pin as output:
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  // here is where you'd put code that needs to be running all the time.

  // check to see if it's time to blink the LED; that is, if the
  // difference between the current time and last time you blinked
  // the LED is bigger than the interval at which you want to
  // blink the LED.
  unsigned long currentMillis = millis();
```

```
  // difference between the current time and last time you blinked
  // the LED is bigger than the interval at which you want to
  // blink the LED.
  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis > interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;

    // if the LED is off turn it on and vice-versa:
    if (ledState == LOW)
      ledState = HIGH;
    else
      ledState = LOW;

    // set the LED with the ledState of the variable:
    digitalWrite(ledPin, ledState);
  }
}
```

5. *Analog Output Serial*(Integrasi penelitian ke dalam praktik)********

```
/
  Analog input, analog output, serial output

 Reads an analog input pin, maps the result to a range from 0 to 255
 and uses the result to set the pulsewidth modulation (PWM) of an output pin.
 Also prints the results to the serial monitor.

 The circuit:
 * potentiometer connected to analog pin 0.
   Center pin of the potentiometer goes to the analog pin.
   side pins of the potentiometer go to +5V and ground
 * LED connected from digital pin 9 to ground

 created 29 Dec. 2008
 modified 9 Apr 2012
 by Tom Igoe

 This example code is in the public domain.

 */

// These constants won't change.  They're used to give names
// to the pins used:
const int analogInPin = A0;  // Analog input pin that the potentiometer is attached to
const int analogOutPin = 9; // Analog output pin that the LED is attached to

int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM (analog out)
```

```
void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  // change the analog out value:
  analogWrite(analogOutPin, outputValue);

  // print the results to the serial monitor:
  Serial.print("sensor = " );
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  // wait 2 milliseconds before the next loop
  // for the analog-to-digital converter to settle
  // after the last reading:
  delay(2);
}
```

6. *Analog Input*

```
/*
  Analog Input
 Demonstrates analog input by reading an analog sensor on analog pin 0 and
 turning on and off a light emitting diode(LED)  connected to digital pin 13.
 The amount of time the LED will be on and off depends on
 the value obtained by analogRead().

 The circuit:
 * Potentiometer attached to analog input 0
 * center pin of the potentiometer to the analog pin
 * one side pin (either one) to ground
 * the other side pin to +5V
 * LED anode (long leg) attached to digital output 13
 * LED cathode (short leg) attached to ground

 * Note: because most Arduinos have a built-in LED attached
 to pin 13 on the board, the LED is optional.


 Created by David Cuartielles
 modified 30 Aug 2011
 By Tom Igoe

 This example code is in the public domain.
```

```
*/

int sensorPin = A0;    // select the input pin for the potentiometer
int ledPin = 13;      // select the pin for the LED
int sensorValue = 0;  // variable to store the value coming from the sensor

void setup() {
  // declare the ledPin as an OUTPUT:
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // read the value from the sensor:
  sensorValue = analogRead(sensorPin);
  // turn the ledPin on
  digitalWrite(ledPin, HIGH);
  // stop the program for <sensorValue> milliseconds:
  delay(sensorValue);
  // turn the ledPin off:
  digitalWrite(ledPin, LOW);
  // stop the program for for <sensorValue> milliseconds:
  delay(sensorValue);
}
```

7. *Analog Write Mega*

```
/*
  Mega analogWrite() test

  This sketch fades LEDs up and down one at a time on digital pins 2 through 13.
  This sketch was written for the Arduino Mega, and will not work on previous boards.

  The circuit:
  * LEDs attached from pins 2 through 13 to ground.

  created 8 Feb 2009
  by Tom Igoe

  This example code is in the public domain.

 */
// These constants won't change.  They're used to give names
// to the pins used:
const int lowestPin = 2;
const int highestPin = 13;


void setup() {
  // set pins 2 through 13 as outputs:
  for (int thisPin =lowestPin; thisPin <= highestPin; thisPin++) {
    pinMode(thisPin, OUTPUT);
  }
}
```

8. ***Pemrograman Embedded System*** **(Integrasi Penelitian kedalam praktek (Suroto et al., 2019)***)**

soluna_3_dgn_Fastidle_ok_dingin_1

```
 */

int sensorPin = A1; `   // select the input pin for the potentiometer
int inj1 = 2;       // select the pin for injector 1
int inj2 = 3;       // select the pin for fast idle
int inj3 = 4;       // select the pin for injector 2
int inj4 = 5;       // select the pin fo injector 3
int inj5 = 6;       // select the pin for injector 4
int sensorValue = 0;  // variable to store the value coming from the sensor

void setup() {
  // declare the ledPin as an OUTPUT:
  //Serial.begin(9600);
  pinMode(inj1, OUTPUT);
  pinMode(inj2, OUTPUT);
  pinMode(inj3, OUTPUT);
  pinMode(inj4, OUTPUT);
  pinMode(inj5, OUTPUT);
}
void pwm(int lw,int hg,int inj)
{
  if(inj == 1)
  {
      digitalWrite(inj1,HIGH);
      digitalWrite(inj2,LOW);
      digitalWrite(inj3,LOW);
      digitalWrite(inj4,LOW);
      digitalWrite(inj5,LOW);
```

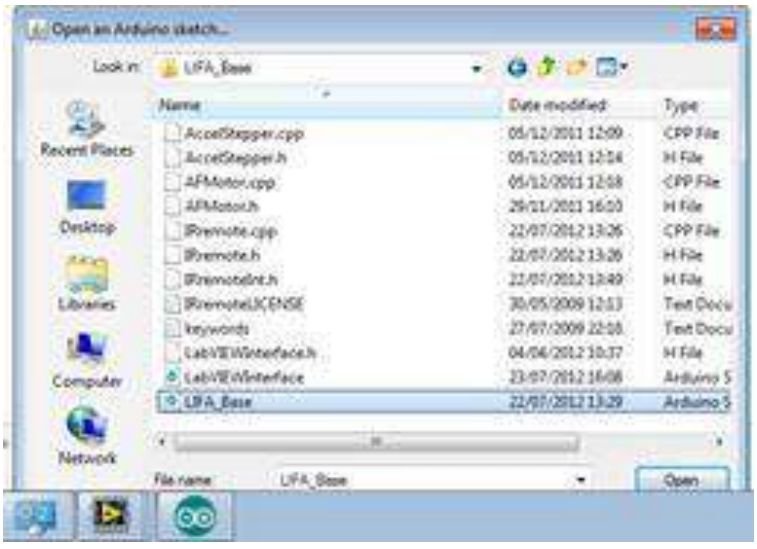
```
        digitalWrite(inj4,LOW);
        digitalWrite(inj5,LOW);
        delay(hg);
        digitalWrite(inj1,LOW);
        digitalWrite(inj2,LOW);
        digitalWrite(inj3,LOW);
        digitalWrite(inj4,LOW);
        digitalWrite(inj5,LOW);
        delay(lw);
    }
  else if(inj == 2)
    {
        digitalWrite(inj1,HIGH);
        digitalWrite(inj2,HIGH);
        digitalWrite(inj3,LOW);
        digitalWrite(inj4,LOW);
        digitalWrite(inj5,LOW);
        delay(hg);
        digitalWrite(inj1,LOW);
        digitalWrite(inj2,LOW);
        digitalWrite(inj3,LOW);
        digitalWrite(inj4,LOW);
        digitalWrite(inj5,LOW);
        delay(lw);
    }
    else if(inj == 3)
    {
        digitalWrite(inj1,HIGH);
        digitalWrite(inj2,HIGH);
        digitalWrite(inj3,HIGH);
        digitalWrite(inj4,LOW);
        digitalWrite(inj5,LOW);
        delay(hg);
        digitalWrite(inj1,LOW);
        digitalWrite(inj2,LOW);
        digitalWrite(inj3,LOW);
        digitalWrite(inj4,LOW);
        digitalWrite(inj5,LOW);
        delay(lw);
    }
     else if(inj == 4)
    {
        digitalWrite(inj1,HIGH);
        digitalWrite(inj2,HIGH);
        digitalWrite(inj3,HIGH);
        digitalWrite(inj4,HIGH);
        digitalWrite(inj5,LOW);
        delay(hg);
        digitalWrite(inj1,LOW);
        digitalWrite(inj2,LOW);
        digitalWrite(inj3,LOW);
        digitalWrite(inj4,LOW);
        digitalWrite(inj5,LOW);
        delay(lw);
```

```
      else if(inj == 5)
  {
       digitalWrite(inj1,HIGH);
       digitalWrite(inj2,HIGH);
       digitalWrite(inj3,HIGH);
       digitalWrite(inj4,HIGH);
       digitalWrite(inj5,HIGH);
       delay(hg);
       digitalWrite(inj1,LOW);
       digitalWrite(inj2,LOW);
       digitalWrite(inj3,LOW);
       digitalWrite(inj4,LOW);
       digitalWrite(inj5,LOW);
       delay(lw);
  }
}
void loop() {
  sensorValue = analogRead(sensorPin);
  if( sensorValue <= 140)
  {
    //pwm(low,high,inj)injector idle
      pwm(0,10,1);
  }
  else if(sensorValue >= 170 and sensorValue <= 190)
{
   pwm(0,20,2);// injector fast idle
}
 else if(sensorValue >= 191 and sensorValue <= 193)
{
   pwm(0,8,2);
}
else if(sensorValue >= 194 and sensorValue <= 193)
{
   pwm(0,100,2);
}
else if(sensorValue >= 194 and sensorValue <= 199)
{
   pwm(0,100,2);
}
else if(sensorValue >= 200 and sensorValue <= 201)
{
   pwm(0,6,2);
```

```
  else if(sensorValue >= 202 and sensorValue <= 203)
  {
     pwm(0,8,2);
  }
  else if(sensorValue >= 204 and sensorValue <= 210)
  {
     pwm(1,50,3);// injector 2
  }
   else if(sensorValue >= 211 and sensorValue <= 224)
  {
     pwm(0,50,3);//injector 2
  }
   else if(sensorValue >= 225 and sensorValue <= 599)
  {
     pwm(0,50,4);//injector 3
  }
     else if(sensorValue >= 600 and sensorValue <= 749)
  {
     pwm(0,50,5);//injector 4
  }
  else if(sensorValue >= 750)
  {
     pwm(0,10,5);
  }
  //Serial.print("sensor = ");
  //Serial.println(sensorValue);
}
```

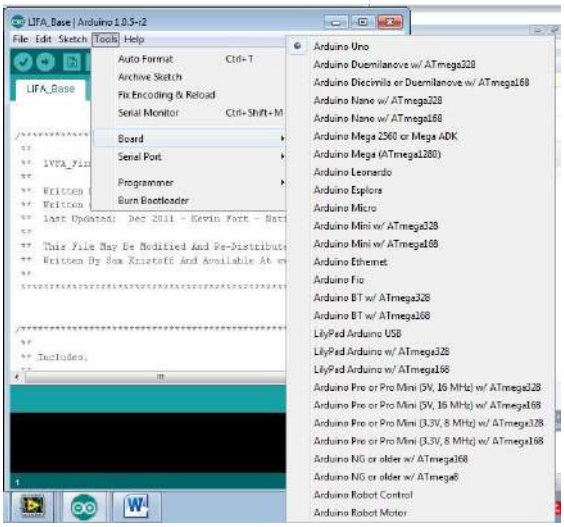
3.6. SETTING *SOFTWARE* KE *HARDWARE*

3.6.1 Import *LIFA_BASE*



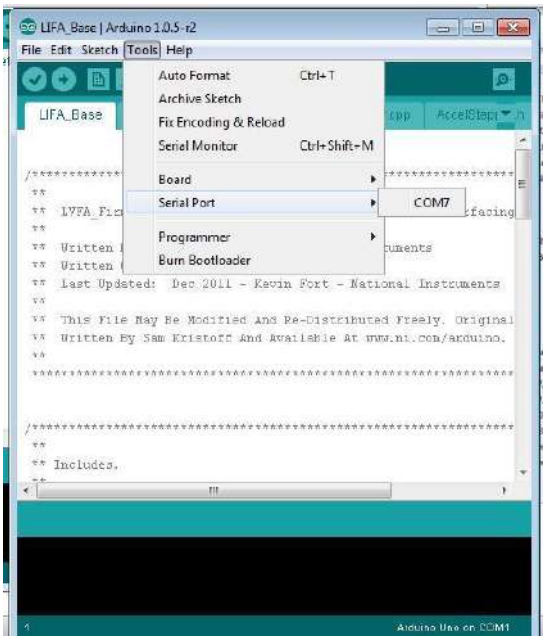
Gambar 3.2 Import *LIFA_BASE* (Integrasi penelitian ke dalam praktik).**

3.6.2 Setting Board Pada *Microcontroller*



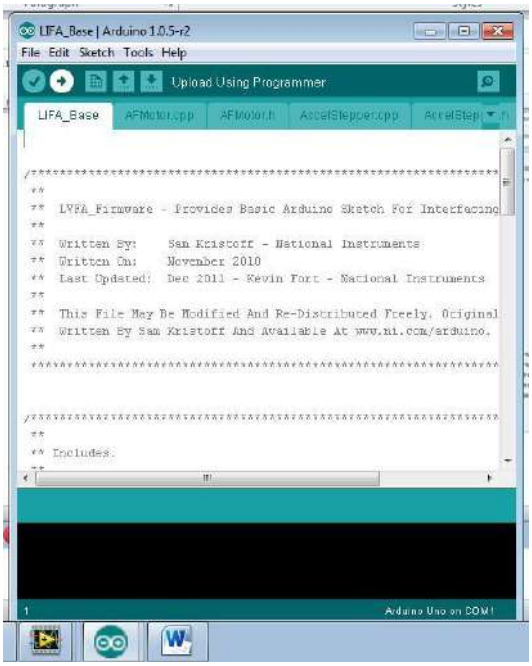
Gambar 3. 3 Setting board pada *microcontroller*.

3.6.3 Setting *Port* Pada *Microcontroller*



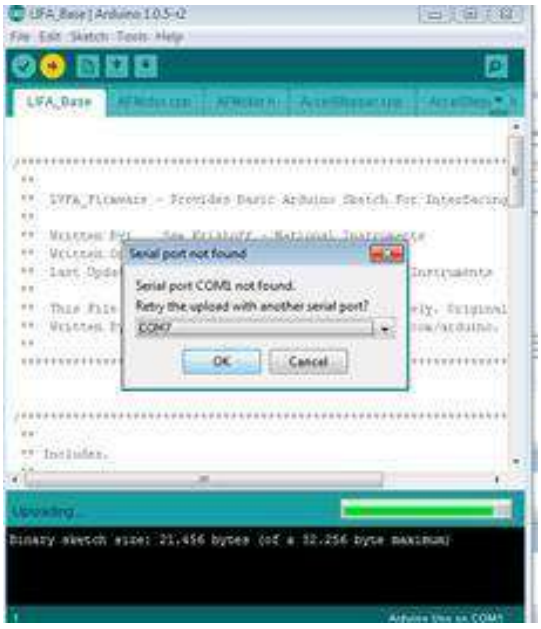
Gambar 3. 4 *Setting Port.****.

3.6.4 Upload Program (*Embeded*)



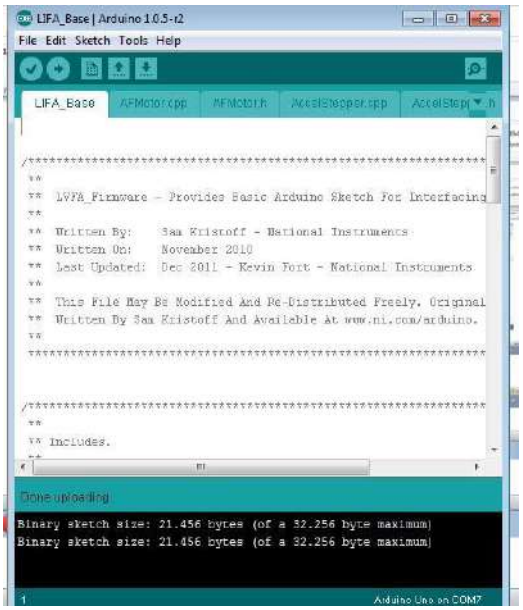
Gambar 3. 5 Upload Program (Embeded).

3.6.5 Searching *Port USB*



Gambar 3. 6 Searching Port USB.

3.6.6 Indikator Embeded berhasil



Gambar 3. 7 Indikator embeded berhasil .

BAB 4. KEGIATAN BELAJAR SUB CPMK 4

4.1. TARGET CPL MATA KULIAH

SUB CPMK	TUJUAN
AM - 04	Menguasai konsep dan mampu menerapkan *data acquisition* pada kendaraan

4.2. TARGET PEMBELAJARAN PRAKTEK

SUB CPMK	TUJUAN PEMBELAJARAN PRAKTEK
AM - 04	Mampu menerapkan *data acquisition* pada kendaraan

4.3. URAIAN

Target pembelajaran praktek pada sub CPMK AM-04 fokus pada pencapian untuk mampu menerapkan *data acquisition* pada kendaraan. Untuk mencapai target ini ada beberapa langkah yang harus dilakukan:

- Merancang pemrograman grafis dan LabVIEW yang sudah dibahas pada BAB 1.
- Merancang control system yang dibahas pada BAB 2.
- Merancang koneksi software dan hardware yang dibahas pada BAB 3.
- Mengukur signal sensor kendaraan yang terintegrasi.
- Buat instalasi antara sensor pada kendaraan, data akuisisi board dan komputer.
- Lakukan pengukuran signal pada sensor.

4.4. ALAT DAN BAHAN PRAKTEK

No.	Deskripsi	Item	Keterangan
1.	Software IDE	1 Paket	Arduino
2.	Software Lab*VIEW*	1 paket	National Instrument
3.	Komputer	24 unit	Ram 4/6/8 Giga
4.	Kabel	1 set	Jumper, NYY/NYA
5.	*Microcontroller*	1 set	Mega 2560, Mega 32, NI 6008/6009 dan lain - lain
6.	Sensor pada kendaraan	1 set	Sensor TPS, ECT, CKP, MAP, IAT dan lain

4.5. TUGAS

Buatlah sistem data akuisisi untuk mengukur signal sensor pada kendaraan yang disertai denga wiring diagramnya !

BAB 5. KEGIATAN BELAJAR SUB CPMK 5

5.1 TARGET CPL MATA KULIAH

SUB CPMK	TUJUAN
AM - 05	Menguasai konsep dan mampu menerapkan *embeded systems* pada sistem kendaraan berdasarkan ***Case Based Learning*** **- MBKM**

5.2 TARGET PEMBELAJARAN PRAKTEK

SUB CPMK	TUJUAN PEMBELAJARAN PRAKTEK
AM - 05	Mampu menerapkan *embeded systems* pada sistem kendaraan berdasarkan ***Case Based Learning*** **- MBKM**

5.3 URAIAN

Target pembelajaran praktek pada sub CPMK AM-03 fokus pada pencapian untuk mampu menerapkan *embeded systems* pada kendaraan berdasarkan ***Case Based Learning*** **- MBKM**. Untuk mencapai target ini ada beberapa langkah yang harus dilakukan:

- Membuat data akuisisi yang sudah dibahas pada BAB 4.
- Merancang pemrograman yang dapat di-*embedded*-kan pada *Central Processing Unit- CPU* pada sebuah *microcontroller* untuk mengendalikan sistem yang akan dikontrol yang sudah dibahas pada BAB 3.
- Merancang sistem bahasa pemrogram meliputi Bahasa C, Bahasa C++, bahasa Grafis dan Bahasa Matematis secara simulasi dan realtime. Adapun untuk merancang pemrograman ini menggunakan *Software Integrate Development Enviroment* (IDE) yang dikeluarkan oleh perusahaan pengembang *microcontroller* tipe mega (Arduino) yang sudah dibahas pada BAB 3.
- Merancang sistem kontrol terintegrasi yang terdiri dari sensor, actuator, microcontroller yang menggunakan embedded system pada sistem kendaraan nyata.
- Sistem kontrol yang dirancang berangkat dari permasalahan/kasus yang terjadi pada kendaraan nyata.

Sistem kontrol yang dirangkai dalam kendaraan dapat meningkatkan berbagai keuntungan yang diperoleh, seperti pada penelitian sebelumnya sistem kontrol

dengan *embedded system* dapat meningkatkan penghematan bahan bakar, mengenali perilaku manusia dan pengendalian AC pada kendaraan [4]-[7]. Untuk itu perlu memberi penugasan pada kegiatan praktkk ini berdasarkan kasus pada kendaraan nyata.

5.4 ALAT DAN BAHAN PRAKTEK

No.	Deskripsi	Item	Keterangan
1.	Software IDE	1 Paket	Tune Audio, Arduino
2.	Software LabVIEW	1 Paket	National Instrument
3.	Komputer	24 unit	Ram 4/6/8 Giga
4.	Kabel	1 set	Jumper, NYY/NYA
5.	Microcontroller	1 set	Mega 2560, Mega 32 dan lain - lain
6.	Sensor pada kendaraan	1 set	Sensor TPS, ECT, CKP, MAP, IAT dan lain.
7.	Actuator pada kendaraan	1 paket	Fuel pump, injector, motor starter, Motor DC, Motor AC dan lain-lain.

5.5 TUGAS

Buatlah sistem kontrol yang dirancang berangkat dari permasalahan/kasus yang terjadi pada kendaraan nyata yang disertai dengan wiring diagramnya !

BAB III. PENUTUP

Dalam peningkatan kemampuan memahami logika programan dengan *Software LabView* memerlukan latihan secara komperhensif. Hasil yang diperoleh dari praktikum ini memberikan bekal terhadap kemampuan ketrampilan mahasiswa dibidang *automotive mechatronics.* Demikian pula untuk pemahaman pemrograman dengan Bahasa C/C++, Matematis memerlukan latihan yang cukup tinggi, untuk itu perlu dikembangkan lebih lanjut.

DAFTAR PUSTAKA

[1] Measurement Computing, 2012, " *A Reference For DAQ And Analog & Digital Signal Conditioning*", Data acquisition Handbook Third Edition Published 2004-2012 in the United States of America.

[2] Longoria R.G, 2006," Digital Measurement Interfaces and Computer-Aided Data Acquisition," Department of Mechanical Engineering,The University of Texas at Austin.

[3] Katshuhiko O., 2002 "*Modern Control Engineering*,"Prentice-Hall, Inc. the United States of Anierica.

[4] Jeffrey T. and Jim K., 2006," *LabVIEW for Everyone: Graphical Programming Made Easy and Fun, Third Edition*,"Prentice Hall Inc. the United States of America.

[5] Suroto M., Muji S., Bagiyo C.P., Aris T. and Joga D.S, 2020," Design and application of air to fuel ratio controller for LPG fueled vehicles at typical down-way," *SN Applied Sciences, Vol. 2,No.*

[6] Triwiyatno, A., Sinuraya, E. W., Setiawan, J. D., & Munahar, S. (2015, October). Smart controller design of air to fuel ratio (AFR) and brake control system on gasoline engine. In *2015 2nd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)* (pp. 233-238). IEEE.

[7] Munahar, S., Triwiyatno, A., Munadi, M., & Setiawan, J. D. (2022). Fuel Saving Index Assessment On Driving Behavior Control System Of Prototype Model Using Neural Network. *Archieve of transport*, *63*(3), 123-141.

[8] Munahar, S., Purnomo, B. C., Izzudin, M., Setiyo, M., & Saudi, M. M. (2022). Vehicle Air Conditioner (VAC) Control System Based On Passenger Comfort: A Proof Of Concept. *IIUM Engineering Journal*