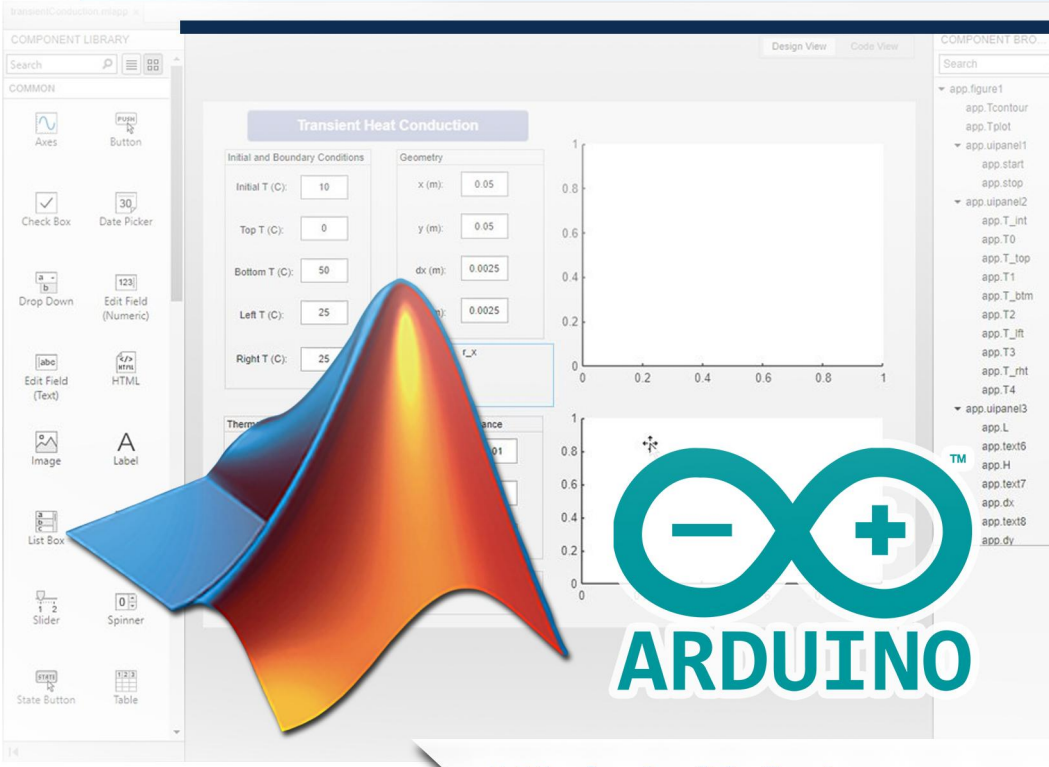
POLMANBABEL PRESS

GRAPHICAL USER INTERFACE (GUI) **MATLAB** DAN *INTERFACE* DENGAN **ARDUINO**



I Made Andik Setiawan

GRAPHICAL USER INTERFACE (GUI)
MATLAB DAN *INTERFACE*
DENGAN ARDUINO

UU No 28 tahun 2014 tentang Hak Cipta

Fungsi dan sifat hak cipta Pasal 4

Hak Cipta sebagaimana dimaksud dalam Pasal 3 huruf a merupakan hak ekslusif yang terdiri atas hak moral dan hak ekonomi.

Pembatasan Perlindungan Pasal 26

Ketentuan sebagaimana dimaksud dalam Pasal 23, Pasal 24, dan Pasal 25 tidak berlaku terhadap:

i. Penggunaan kutipan singkat Ciptaan dan/atau produk Hak Terkait untuk pelaporan peristiwa aktual yang ditujukan hanya untuk keperluan penyediaan informasi aktual;

ii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk kepentingan penelitian ilmu pengetahuan;

iii. Penggandaan Ciptaan dan/atau produk Hak Terkait untuk keperluan pengajaran, kecuali pertunjukkan dan Fonogram yang telah dilakukan Pengumuman seebagai bahan ajar; dan

iv. Penggunaan untuk kepentingan pendidikan dan pengembangan ilmu pengetahuan yang memungkinkan suatu Ciptaan dan/atau produk Hak Terkait dapat digunakan tanpa izin Pelaku Pertunjukan, Produser Fonogram, atau Lembaga Penyiaran.

Sanksi Pelanggaran Pasal 113

1. Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp 100.000.000 (serratus juta rupiah).
2. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp 500.000.000,00 (lima ratus juta rupiah).

GRAPHICAL USER INTERFACE (GUI) MATLAB DAN *INTERFACE* DENGAN ARDUINO

I MADE ANDIK SETIAWAN, PH.D

POLITEKNIK MANUFAKTUR NEGERI BANGKA BELITUNG

Graphical user Interface (GUI) Matlab dan _Interface_ dengan Arduino

I Made Andik Setiawan

Pengarah *: I Made Andik Setiawan*
Editor *: Mardliyah Ayu*
Korektor *: Subkhan*
Setting *: Sidhiq Andriyanto*
Desain Grafis *: Muhammad Zenda Rud*

Jumlah halaman *: x, 93 halaman*
Ukuran kertas *: A5, 15x21 cm*
Cetakan Pertama *: Juni 2022*
ISBN *: 9786239787066*

Penerbit
Politeknik Manufaktur Negeri Bangka Belitung
Kawasan Industri Air Kantung, Sungailiat, Bangka
Telp/Faks: (0717) 93586

KATA PENGANTAR

Buku ini ditulis untuk membantu para mahasiswa khususnya bidang elektronika dalam belajar dan mempraktekkan komunikasi dan interface antara Matlab dan Arduino. Graphical User Interface Matlab sebagai media menu dan tampilan untuk user sedangkan Arduino sebagai perangkat elektronik penggerak dan penghubung dengan komponen.

Tentunya masih banyak kekurangan dalam buku ini, dan akan terus menjadi bahan perbaikan dimasa yanag akan datang.

Kepada Allah, kami berharap amal jariyah dan pahala, dan semoga buku ini membawa manfaat dan keberkahan.

Masa-masa pandemi Covid-19

Penulis,

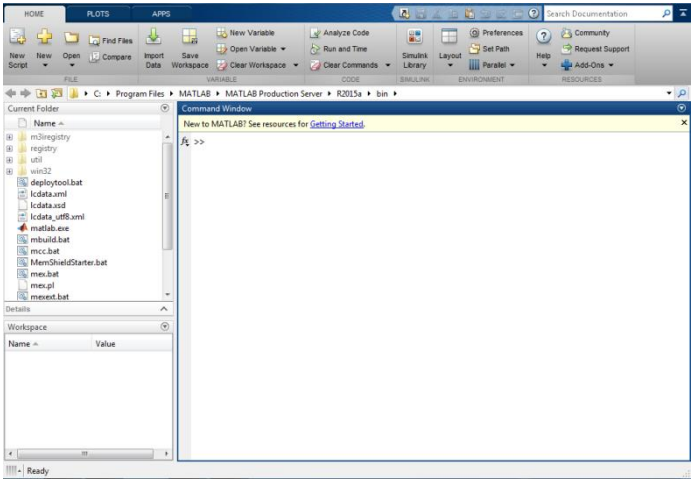
DAFTAR ISI

DAFTAR GAMBAR

BAB 1. PENDAHULUAN

Singkatan Matlab berasal dari kata Matrix Laboratory dan dibuat oleh The MathWorks (Houcque, 2005). Matlab adalah *high-level programming* dan memiliki fasilitas yang interaktif untuk visualisasi, komputasi dan programming (Point, 2014). Matlab juga berbasis *object-oriented-programming* sehingga cocok untuk keperluan pengajaran dan penelitian (Beyenir, 2013).

Tampilan awal ketika program Matlab di-*run* tampak pada Gambar 1.1 di bawah ini.

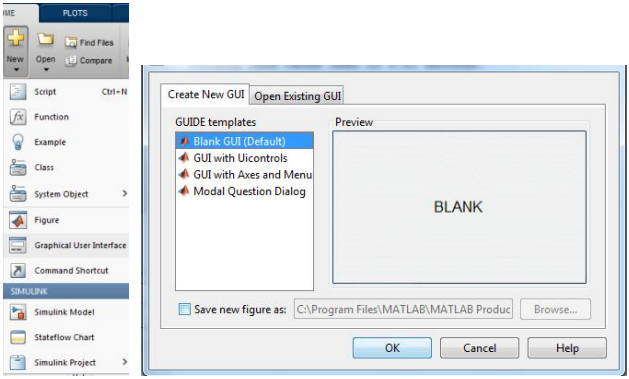


Gambar 1.1Tampilan awal Matlab

Graphical user Interface (GUI) Matlab adalah fasilitas interaktif yang disediakan oleh Matlab untuk programming berbasis *object*. Untuk membuka *Graphical user Interface*:

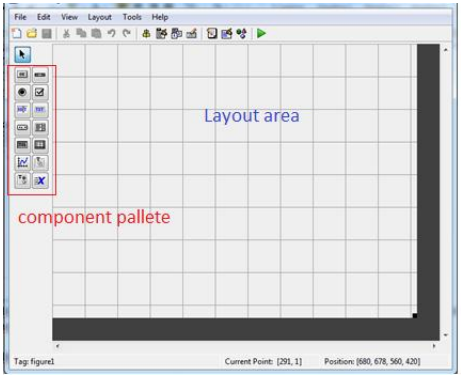
- Klik "*New*", pilih "*Graphical user Interface*",
- muncul *Dialogbox*,
- Pilih "*Create New GUI*" untuk membuat file aplikasi GUI yang baru dan pilih "*Open Existing GUI*" untuk membuka

- file GUI yang sudah pernah dibuat,



Gambar 1.2 *Dialogbox Graphical user Interface*

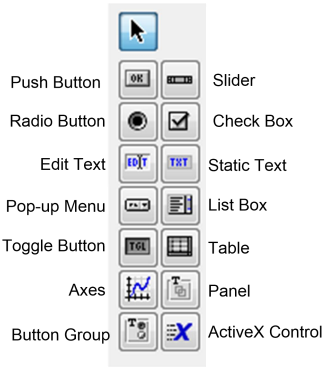
- Pilih "*Blank GUI*", maka akan muncul *layout GUI*



Gambar 1.3 Tampilan *Graphical user Interface*

Component pallete adalah *component - component GUI* yang disediakan oleh Matlab. Sedangkan *Layout area* adalah tampilan layar ketika program akan di-*running*.

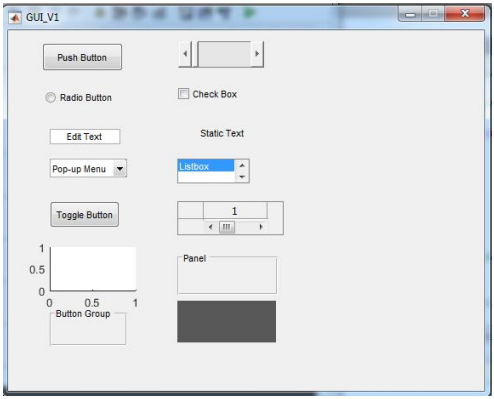
Component-components GUI yang tersedia antara lain *Push Button, Slider, Radio Button, Check Box, Edit Text, Static Text, Pop-up Menu, Listbox, Toggle Button, Table, Axes, Panel, Button Group*, dan *Active X Control*, seperti ditunjukkan pada Gambar 1.4.



Gambar 1.4 Komponen GUI yang tersedia di *Components Pallette*

Untuk dapat menggunakan *components* GUI, maka langkah yang dilakukan yaitu dengan Klik (jangan dilepas) pada *component* yang diinginkan dan Tarik (*drag*) ke layout area.

Contoh hasil dari klik dan *drag components* GUI Matlab adalah sebagai berikut:

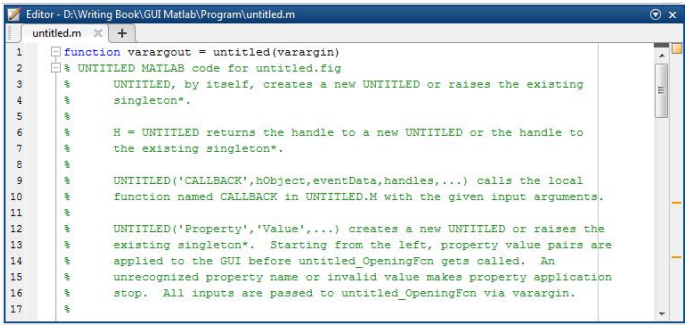


Gambar 1. 1 Hasil *drag* dari *components pallete*

Untuk menyimpan file dapat dilakukan dengan:

- Klik menu “File”, pilih “Save” atau “Save as”

- File jenis *Figures* (*.fig) adalah untuk menyimpan tampilan pada Layout area, sedangkan program yang akan dieksekusi akan disimpan pada M-file (*.m), seperti gambar berikut ini.



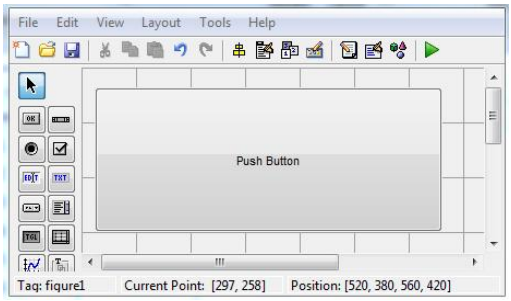
Gambar 1.6 Editor M-File Matlab

- Klik “Save”, pilih “Save As” untuk menyimpan kode program.

BAB 2. BEKERJA DENGAN KOMPONEN

2.1. Komponen *Push Button*

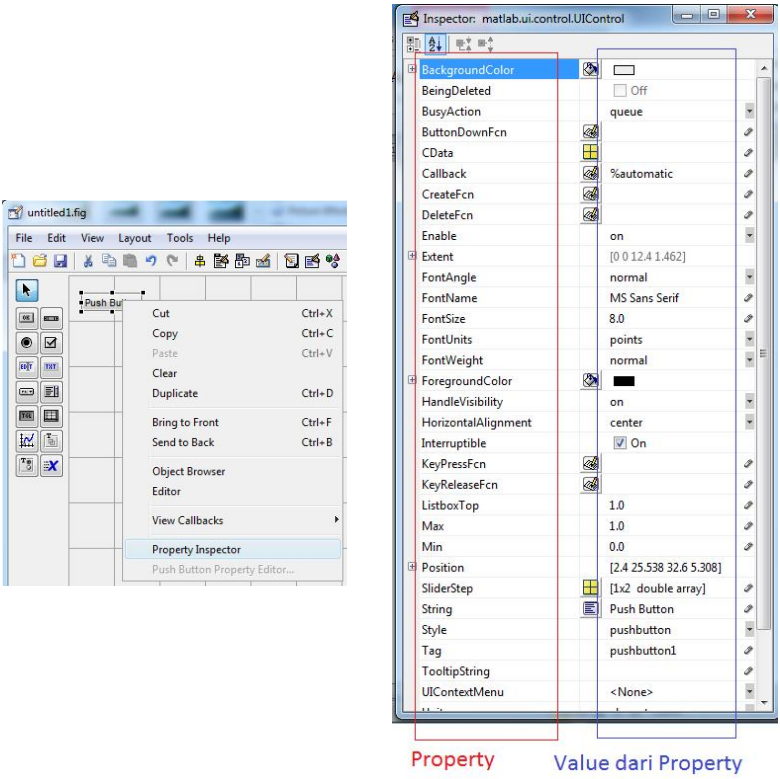
Push Button adalah *component* GUI Matlab jenis tombol yang dapat ditekan atau di-klik. Untuk dapat menggunakan komponen *Push Button*, maka klik dan *drag Push Button* dari *palette* ke *layout area*, seperti Gambar 2.1. Ukuran dari *Push Button* dapat diperbesar atau diperkecil dengan cara *drag* pada titik-titik sekeliling *Push Button*.



Gambar 2.1Komponen *Push Button*

Tampilan di atas adalah tampilan *default* dari komponen *Push Button*. Tampilan ini dapat kita ubah dengan mengubah *Property* (parameter) dari komponen tersebut, dengan cara:

- *Double click* pada komponen yang ingin diubah *property*-nya, atau Klik kanan pada komponen ingin diubah *property*-nya, pilih "*Property Inspector*",
- Maka akan muncul menu "*Property Inspector*" dari komponen *Push Button*,



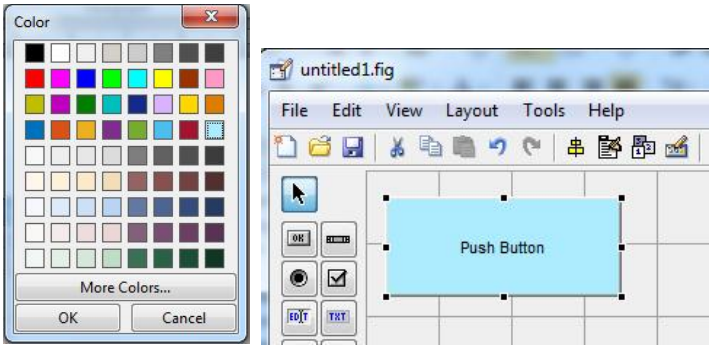
Gambar 2.2 *Property Inspector* komponen *Push Button*

2.2. Property Inspector

Property inspector terdiri dari dua parameter yaitu *Property* (kolom sebelah kiri) dan *Value* (kolom sebelah kanan). *Property* menunjukkan jenis parameter, sedangkan *Value* menunjukkan nilai atau isi dari *property* atau parameter tersebut.

Untuk mengubah parameter dari *Property Inspector Push Button* dapat dilakukan dengan klik pada *property* tersebut, sebagai contoh:

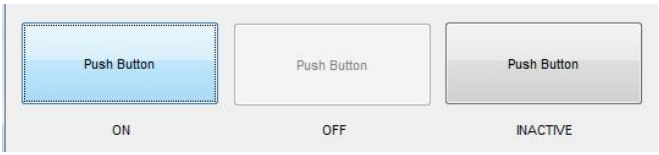
- Untuk mengubah nilai *property* "*BackgroundColor*", maka klik *value property backgroun*dcolor atau klik pada kotak sebelah kanan dari *property* "*BackgroundColor*".
- Akan muncul *Color dialog box*, pilih warna yang diinginkan, klik OK.



Gambar 2.3 *Color Dialogbox* pada komponen Push Button

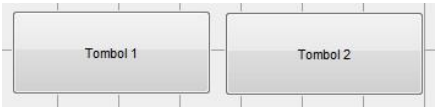
Properties yang lainnya dapat kita ubah nilainya dengan cara yang sama. Beberapa *Property Inspector* yang penting untuk diketahui dari komponen *Push Button*, antara lain:

a. *BackgroundColor*: warna latar belakang.
b. *Enable*: dapat digunakan atau tidak, nilai "on" (dapat ditekan), "off" (tidak dapat digunakan dan tidak dapat ditekan), "*inactive*" (tidak dapat ditekan).



c. *FontAngle, FontName, FontSize, FontUnit, FontWeight*: adalah parameter yang digunakan untuk mengatur kemiringan *font*, jenis *font*, ukuran *font*, satuan ukuran *font*, dan ketebalan *font*. *Font* ini akan muncul pada badan *Push Button*.
d. *ForegoundColor*: warna tampilan *font*.

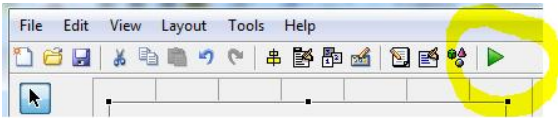
e. *String*: tulisan yang akan ditampilkan. *Default*-nya adalah "*Push Button*", dapat diganti dengan tulisan "Tombol 1", "Tombol 2", dan lain-lainnya.



f. *Tag*: nama atau Identitas dari komponen tersebut yang membedakan dari komponen lainnya.
g. *Visible*: nilai "on" maka komponen dapat dilihat setelah *running*, dan "off" berarti komponen tidak dapat dilihat (hilang) dari *layout* pada saat *running* program.

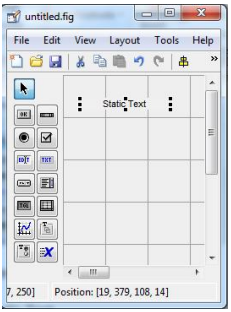
2.3. Running Program

Setelah selesai membuat *Layout* tampilan, maka "Save" atau "Save As" baik file *layout* (*.fig) maupun program (*.m). Untuk melihat hasil tampilan sebenarnya dapat dilakukan dengan *running* program, yaitu klik *icon running*  atau klik menu "*Tools*" dan pilih "*Run*". Untuk beberapa jenis *Property*, tampilan layar ketika proses mendesain dan ketika *running* dapat berbeda.



2.4. Komponen Static Text

Static Text adalah komponen yang dipergunakan untuk menampilkan *text* ke *Layout* area. *Text* yang ditampilkan bersifat *static* yaitu tidak dapat diubah oleh *user*.

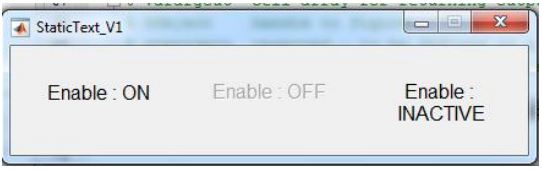


Gambar 2.4 Komponen *Static Text* dalam *layout area*

Beberapa *Property Inspector* yang penting untuk diketahui dari komponen *Static Text*, antara lain:

a. *BackgroundColor*: warna latar belakang.

b. *Enable*: dapat digunakan atau tidak, nilai "on" (dapat ditekan), "off" (tidak dapat digunakan dan tidak dapat ditekan), "inactive" (tidak dapat ditekan).



c. *FontAngle, FontName, FontSize, FontUnit, FontWeight:* adalah parameter yang digunakan untuk mengatur kemiringan *font*, jenis *font*, ukuran *font*, satuan ukuran *font*, dan ketebalan *font*. *Font* ini akan muncul pada badan *Push Button*.

d. *ForegroundColor*: warna *font*.

e. *String*: tulisan yang akan ditampilkan. Default-nya adalah "*Static Text*", dapat diganti dengan tulisan "*Red*", "*Green*", dan lain-lainnya.
f. *Tag*: nama atau Identitas dari komponen tersebut yang membedakan dari komponen lainnya.
g. *Visible*: nilai "*on*" maka komponen dapat dilihat setelah running, dan "*off*" berarti komponen tidak dapat dilihat (hilang) dari *layout* pada saat *running* program.

2.5. Function di M-File

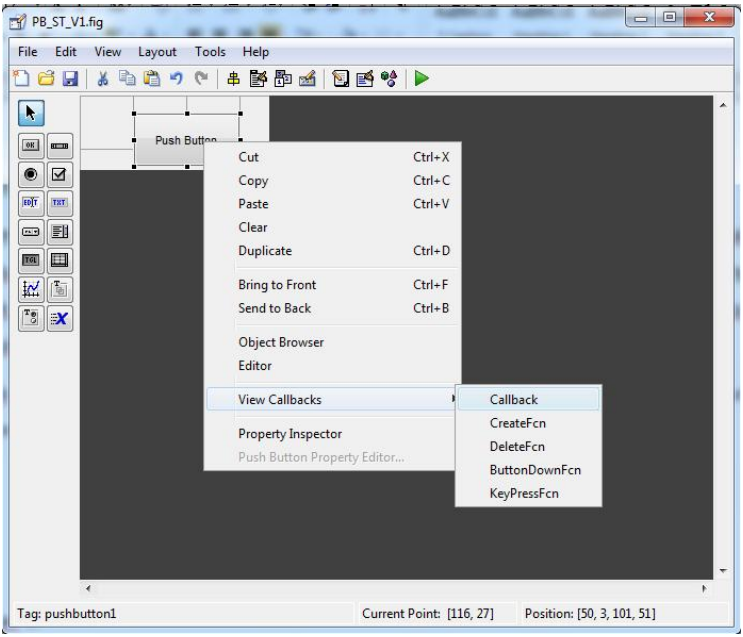
Property-property yang ada di dalam *Property Inspector* dapat diubah pada saat desain sebelum running (dilakukan seperti contoh di atas) maupun pada saat program telah di-*running*. Untuk mengubah *value property* pada saat program telah running, maka perubahan dapat dilakukan melalui program di M-File.

Untuk komponen *Push Button*, *function* program yang akan dieksekusi oleh Matlab ketika *Push Button* di-klik (setelah *running* program) dinamakan *Callback function*, dengan cara:

- Klik kanan pada komponen *Push Button*.
- Pilih "*View Callbacks*", kemudian pilih "*Callback*",

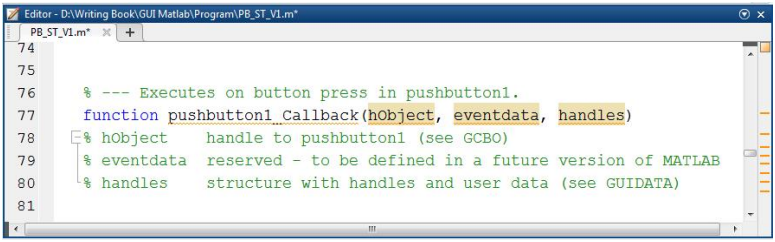
atau

- Klik menu "*View*",
- Pilih "*View Callbacks*", kemudian pilih "*Callback*"



Gambar 2.5 Memanggil *Callback function*

- Setelah itu akan muncul *Editor M-File* dari *Callback function* dari komponen *Push Button,*



Gambar 2.6 *Editor M-File* dari *Callback function* dari *Push Button*

Setelah itu kita dapat menuliskan program berbasis M-File sesuai dengan kebutuhan kita. Fungsi ini berfungsi akan di-*running* oleh Matlab ketika *Push Button* di-klik dengan *mouse*.

Program diketik pada blok "*% --- Executes on button press in Button1*".

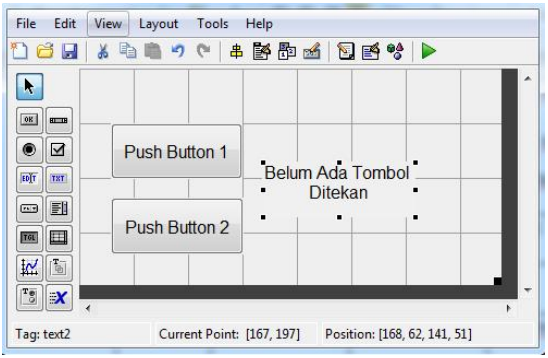
Latihan 1:

Ada dua buah *push button* dan satu buah *static text* dalam layar. Jika *Push Button* 1 diklik maka *static text* akan muncul tulisan "Tombol 1 ditekan" berwarna latar belakang merah. Dan ketika *Push Button* 2 diklik maka muncul tulisan "Tombol 2 ditekan" dan warna latar belakang kuning, maka langkah yang harus dilakukan adalah:

- *Drag* dan drop 2 buah komponen *Push Button* dan 1 satu komponen *Static Text* ke *layout area*,
- Ubah *Property* di *Property Inspector* untuk masing-masing komponen sebagai berikut:

Property	**Push Button 1**	**Push Button 2**	**Static Text**
Font Size	12	12	12
String	"Push Button 1"	"Push Button 2"	"Belum Ada Tombol Ditekan"
Tag	"Button1"	"Button2"	"Tulisan1"

- Sehingga tampilan pada *Layout area* akan tampak seperti berikut:



Gambar 2.7 Tampilan *Layout* setelah perubahan *Property*

- Klik kanan pada *Push Button* 1, pilih "*View Callbacks*", klik "*Callback*", kemudian ketik program ini pada fungsi tersebut:

```
% --- Executes on button press in Button1.
function Button1_Callback(hObject, eventdata, handles)
% hObject    handle to Button1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
set(handles.Tulisan1,'String','Tombol 1 Ditekan');
set(handles.Tulisan1,'BackgroundColor','red');
```

- dimana perintah "*set*" adalah untuk mengubah *property*, "*handles*" untuk menunjukkan *class member* dari GUI, "Tulisan1" adalah *Tag* komponen *Static Text*, "*String*" adalah *property* yang akan diubah, dan "Tombol 1 Ditekan" adalah *value* dari *property String* komponen *Static Text*. Perintah ini dapat dijelaskan berikut.

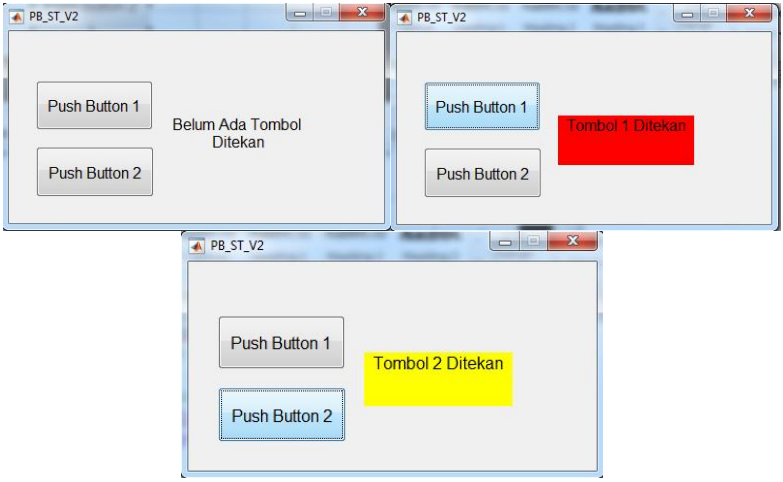
```
set(handles.Tulisan1,'String','Tombol 1 Ditekan');
```

Tag Property Value/nilai Property

- Klik kanan pada *Push Button* 2, pilih "*View Callbacks*", klik "*Callback*", kemudian ketik program ini pada fungsi tersebut:

```
% --- Executes on button press in Button2.
function Button2_Callback(hObject, eventdata, handles)
% hObject    handle to Button2 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
set(handles.Tulisan1,'String','Tombol 2 Ditekan');
set(handles.Tulisan1,'BackgroundColor','yellow');
```

- *Run* program dan klik *Push Button*:



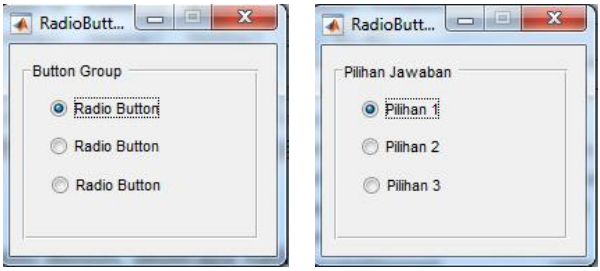
Gambar 2.8 Tampilan running program Latihan 1

BAB 3. LEBIH LANJUT DENGAN KOMPONEN

3.1. Komponen Radio Button dan Button Group

Radio Button adalah komponen GUI Matlab untuk memberikan alternatif dari beberapa kemungkinan yang dapat dipilih oleh *user*. Namun demikian pilihan yang dapat dipilih hanya ada satu pilihan dari beberapa pilihan yang tersedia.

Sedangkan *Button Group* adalah *panel* (pengumpul) dari beberapa komponen *Radio Button*. Beberapa *Radio Button* merupakan satu kumpulan alternatif pilihan bagi *user*, sehingga dikumpulkan dalam satu grup yang disebut dengan *Button Group*. Ilustrasi dari komponen *Radio Button* dan *Button Group* dapat digambarkan sebagai berikut:

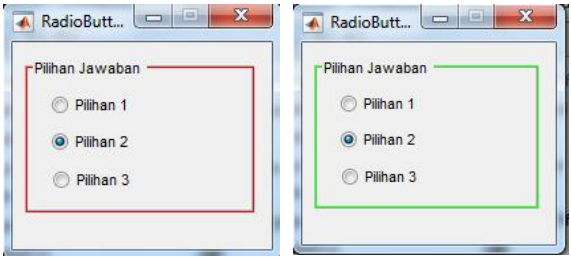


Gambar 3.1 Komponen Button Group dan Radio Button

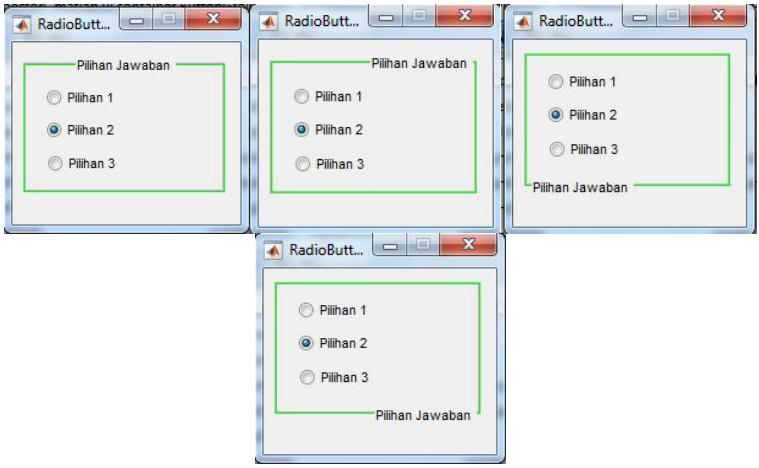
Pada Gambar di atas terdapat tiga komponen *Radio Button* (tiga kemungkinan jawaban) dalam satu grup pertanyaan (satu komponen *Button Group*).

Beberapa *property inspector* yang penting dari *Button Group* adalah:

- *HighlightColor*: warna garis tepi



- *Title*: Judul dari *Button Group*, pada Gambar di atas, *Title*-nya adalah "Pilihan Jawaban".
- *TitlePosition*: letak dari judul, tersedia "*lefttop*" (kiri atas), "*centertop*" (tengah atas), "*righttop*" (kanan atas), "*leftbottom*" (kiri bawah), "*centerbottom*" (tengah bawah), dan "*rightbottom*" (kanan bawah),

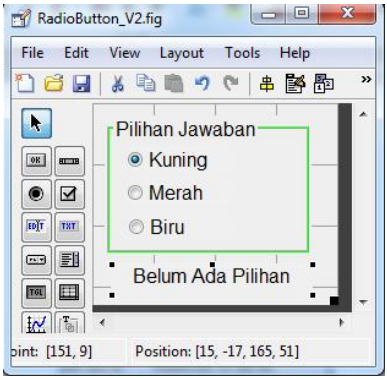


Sedangkan *property Radio Button* yang perlu diketahui:

- *Value*: status terpilih atau tidak, *value*=0 jika tidak terpilih, *value*=1 jika pilihan terpilih (ter-klik *user*).

Latihan 2:

Buat desain *layout* seperti Gambar di bawah ini



Gambar 3.2 Tampilan *Layout* Latihan 2

- Dengan *Property Inspector* untuk masing-masing komponen sebagai berikut:

Property	***Button Group***	***Radio Button 1***	***Radio Button 2***	***Radio Button 3***	***Static Text***
Font Size	12	12	12	12	12
String		"Kuning"	"Merah"	"Biru"	"Belum Ada Pilihan"
Tag	BG1	RB1	RB2	RB3	Tulisan 1
HighlightColor	*Green*				
Title	"Pilihan Jawaban"				
TitlePosition	*lefttop*				
Value		0.0	0.0	0.0	

- Jika *Radio Button* 1 di-klik, maka *Static Text* akan muncul tulisan "Kuning Terpilih" berwarna kuning. Jika *Radio*

Button 2 di-klik, maka *Static Text* akan muncul tulisan "Merah Terpilih" berwarna merah. Jika *Radio Button* 3 di-klik, maka *Static Text* akan muncul tulisan "Biru Terpilih" berwarna biru.

- Buka *Callback function* pada *Radio Button* 1 dan ketik program sebagai berikut:

```
% --- Executes on button press in RB1.
function RB1_Callback(hObject, eventdata, handles)
% hObject    handle to RB1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of RB1
Pilihan=get(handles.RB1,'Value');
if(Pilihan==1)
   set(handles.Tulisan1,'String','Kuning Terpilih');
   set(handles.Tulisan1,'ForegroundColor','yellow');
end
```

- Buka *Callback function* pada *Radio Button* 2 dan ketik program sebagai berikut:

```
% --- Executes on button press in RB2.
function RB2_Callback(hObject, eventdata, handles)
% hObject    handle to RB2 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of RB2
Pilihan=get(handles.RB2,'Value');
if(Pilihan==1)
   set(handles.Tulisan1,'String','Merah Terpilih');
```

```
   set(handles.Tulisan1,'ForegroundColor','red');
end
```

- Buka *Callback function* pada *Radio Button* 3 dan ketik program sebagai berikut:

```
% --- Executes on button press in RB3.
function RB3_Callback(hObject, eventdata, handles)
% hObject    handle to RB3 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of RB3
Pilihan=get(handles.RB3,'Value');
if(Pilihan==1)
   set(handles.Tulisan1,'String','Biru Terpilih');
   set(handles.Tulisan1,'ForegroundColor','blue');
end
```

- Arti kode program *Pilihan=get(handles.RB3,'Value');* ***get*** digunakan untuk mengambil/menerima data, dalam hal ini data dari komponen dengan "Tag" RB3 *property* "*Value*". Data kemudian disimpan dalam variabel bernama "Pilihan".
- if(Pilihan==1) : menunjukkan apakah RB3 terpilih atau tidak (Pilihan=1 jika terpilih, dan Pilihan=0 jika tidak terpilih).
- Tampilan ketika program di-*running.*

Gambar 3.3 Tampilan *running* program Latihan 2

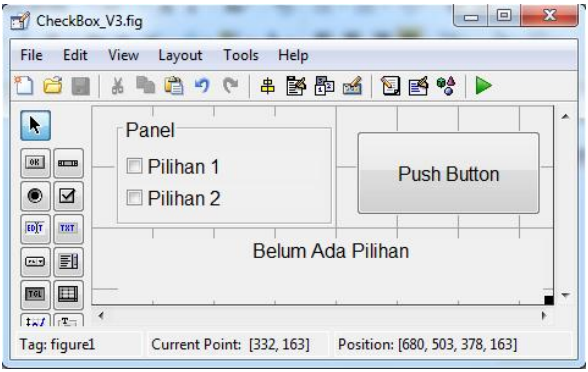
3.2. Komponen Panel dan Check Box

Komponen *Check Box* sama seperti komponen *Radio Button* yaitu komponen yang menyediakan alternatif beberapa pilihan untuk *user*. Tetapi perbedaannya terletak pada jumlah pilihan yang dapat diberikan oleh *user*. Pada komponen *Radio Button*, hanya ada satu jawaban yang dapat dipilih. Sedangkan pada komponen *Check Box*, jumlah pilihan dapat lebih dari satu pilihan.

Komponen *Panel* adalah untuk pengelompokkan (grup) dari komponen *Check Box*. Mempunyai fungsi yang sama dengan *Button Group* untuk komponen *Radio Button*.

Latihan 3

- Buat desain *layout* seperti Gambar di bawah ini



Gambar 3.4 Tampilan Layout Latihan 3

- Dengan *Property* untuk masing-masing komponen sebagai berikut:

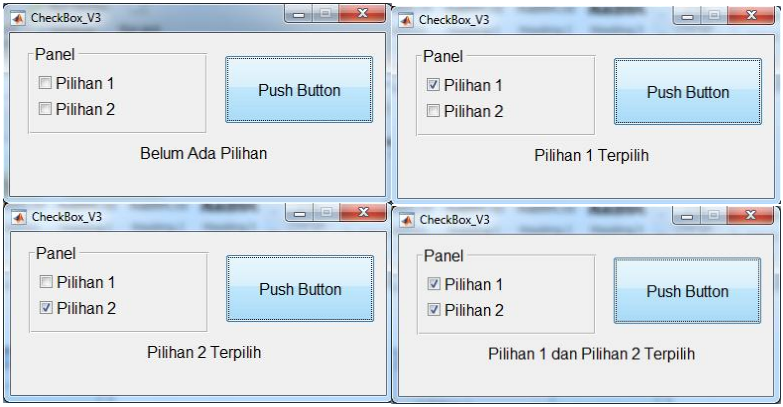
Property	***Panel***	***Check Box 1***	***Check Box 2***	***Push Button***	***Static Text***
Font Size	12	12	12	12	12
String		"Pilihan 1"	"Pilihan 2"	"Push Button"	"Belum Ada Pilihan"
Tag	uipanel1	CB1	CB2	pushbutton1	Tulisan1
Title	Panel				
TitlePosition	lefttop				
Value		0.0	0.0		

- Buka *Callback function* pada komponen *Push Button* dan ketik program sebagai berikut:

```
% --- Executes on button press in pushbutton1.
```

```
function pushbutton1_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
Pilihan1=get(handles.CB1,'Value');
Pilihan2=get(handles.CB2,'Value');
if(Pilihan1==1)&(Pilihan2==1)
   set(handles.Tulisan1,'String','Pilihan 1 dan Pilihan 2
Terpilih');
elseif(Pilihan1==1)&(Pilihan2==0)
   set(handles.Tulisan1,'String','Pilihan 1 Terpilih');
elseif(Pilihan1==0)&(Pilihan2==1)
   set(handles.Tulisan1,'String','Pilihan 2 Terpilih');
else
   set(handles.Tulisan1,'String','Belum Ada Pilihan');
end
```

- Tampilan ketika program di-*running*:



Gambar 3.5 Tampilan *running* program Latihan 3

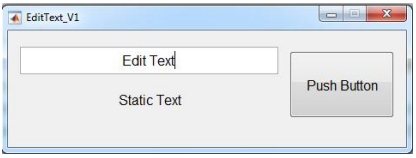
3.3. Komponen Edit Text

Komponen *Edit Text* digunakan untuk menampilkan tulisan kepada *user* sebagaimana komponen *Static Text*. Namun perbedaannya adalah pada komponen *Edit Text*, *user* dapat mengubah atau memasukkan tulisan yang diinginkan. Sedangkan pada komponen *Static Text*, user tidak dapat mengubah tampilan tulisannya.

Property yang dimiliki oleh *Edit Text* sama dengan *property Static Text* sehingga tidak dibahas kembali dalam sub-bab ini. Untuk lebih memahami komponen *Edit Text*, maka akan dijelaskan lebih lanjut melalui Latihan 4.

Latihan 4

- Buat desain *layout* seperti Gambar di bawah ini



Gambar 3.6 Tampilan *Layout* komponen *Edit Text*

- Dengan *Property* untuk masing-masing komponen sebagai berikut:

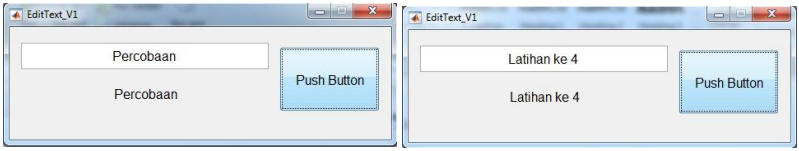
Property	**Edit Text**	**Static Text**	**Push Button**
Font Size	12	12	12
String	"Edit Text"	"Static Text"	Push Button
Tag	EditText1	StaticText1	pushbutton1

- Pada latihan ini semua yang ditulis oleh user pada komponen *Edit Tex*t akan ditampilkan pada *Static Text* setelah *Push Button* diklik.

- Ketik program pada *Callback function* dari komponen *Push Button*, sebagai berikut:

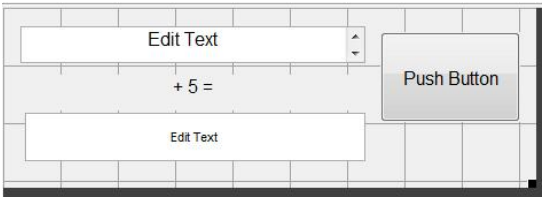
```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
Tulisan=get(handles.EditText1,'String');
set(handles.StaticText1,'String',Tulisan);
```

- Tampilan ketika program di-*running*:



Gambar 3.7 Tampilan running program Latihan 4

- Jika input pada *Edit Text* berupa bilangan, maka perlu konversi dari *String* ke bilangan (*integer, float*, dll).
- Contoh, tambah komponen GUI sebagai berikut:

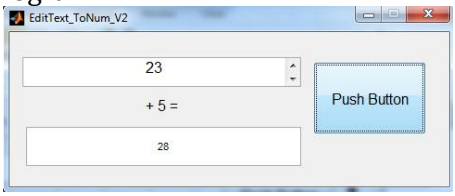


Gambar 3.8 Layout Latihan 4b

- Ubah String dari *Static Text* menjadi: "+ 5 =".
- Modifikasi program pada Callback pushbutton:

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of
MATLAB
% handles    structure with handles and user data (see
GUIDATA)
Tulisan=get(handles.EditText1,'String');
A=str2double(Tulisan);
B=A+5;
set(handles.edit2,'String',B);
```

- Fungsi "A=*str2double*(Tulisan);" adalah untuk meng-konversi nilai *string* pada variable Tulisan menjadi bilangan berbentuk *double* (pecahan).
- *running* program:



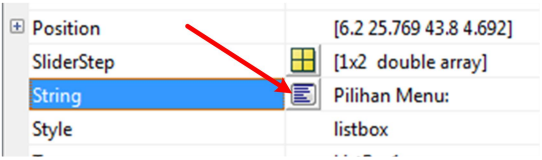
Gambar 3.9 Tampilan running Latihan 4b

3.4. Komponen List Menu

Komponen *List Menu* digunakan untuk menampilkan daftar menu atau pilihan. *user* dapat memilih salah satu menu yang disediakan dengan klik pada pilihan menu yang disediakan atau menekan tombol keatas atau kebawah.

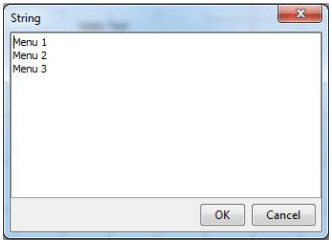
Untuk memasukkan pilihan input dalam *List Menu* dapat dilakukan melalui *property* "*String*", sebagai berikut:

- Klik kotak pada *property* "*String*"



Gambar 3.10 *property* "*String*" pada komponen *List Menu*

- Maka akan muncul *String dialog box*, dan ketik menu atau pilihan yang ingin ditampilkan.



Gambar 3.11 *String Dialog Box*

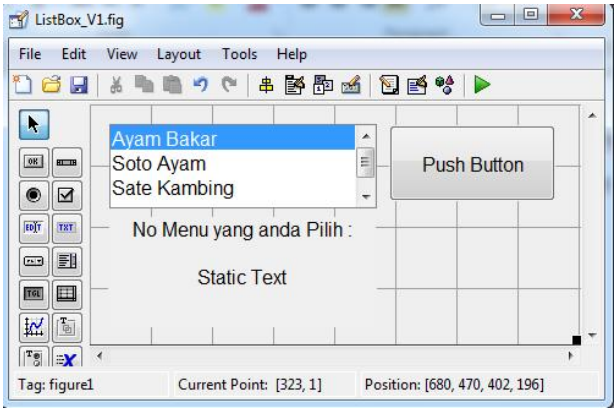
- Klik *OK* dan *run* program, maka daftar menu akan muncul

Gambar 3.12 Tampilan *List Menu*

- Pilihan menu yang dipilih oleh *user* dapat kita ketahui melalui *property* "*Value*", yaitu *value*=1 jika menu yang dipilih, *value*=2 jika menu kedua yang dipilih, dan seterusnya.

Latihan 5

- Buat desain *layout* seperti Gambar di bawah ini



Gambar 3.13 Tampilan Layout Latihan 5

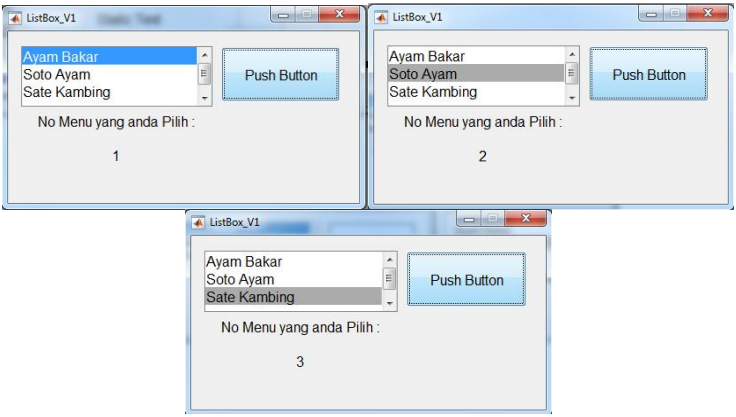
- Dengan *Property* untuk masing-masing komponen sebagai berikut:

Property	***ListBox***	***Static Text 1***	***Static Text 2***	***Push Button***
Font Size	12	12	12	12
String	"Ayam Bakar" "Soto Ayam" "Sate Kambing"	No Menu yang anda Pilih :	Static Text	Push Button
Tag	ListBox1	StaticText1	StaticText 2	pushbutton 1

- Pada latihan ini menu yang dipilih oleh *user* akan ditampilkan no urutnya (dalam angka) setelah *Push Button* ditekan.
- Ketik program pada *Callback function* dari komponen *Push Button*, sebagai berikut:

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
Pilihan_Ke=get(handles.ListBox1,'Value');
set(handles.StaticText2,'String',Pilihan_Ke);
```

- Tampilan ketika program di-*running*:

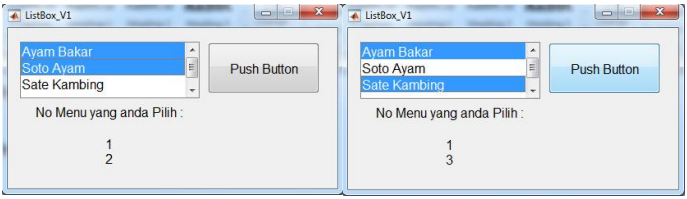


Gambar 3.14 Tampilan *running* program Latihan 5

- Jika menu yang disediakan boleh dipilih lebih dari satu menu, maka *property* "*Max*" dari *List Box* harus diubah.

Default dari property "*Max*" bernilai 1, yang artinya hanya ada satu menu yang dapat dipilih.

- Ubah *property* "*Max*" = 2.0, dan *run* program. Tampilan ketika program di-*running*:

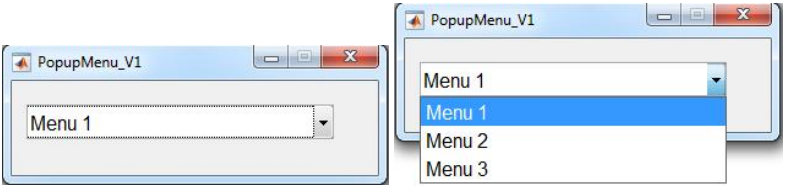


Gambar 3.15 Tampilan *running* program komponen *List Box* dengan pilihan lebih dari satu

- Untuk memilih lebih dari satu pilihan, maka tekan tombol *Ctrl* + klik menu

3.5. Komponen Pop-up Menu

Komponen *Pop-up Menu* mempunyai fungsi yang sama dengan komponen *List Box*, yaitu menampilkan daftar pilihan menu atau pilihan. Tetapi pada komponen *Pop-up Menu*, pilihan menu ditampilkan dalam bentuk *pop-up* dimana tidak semua menu ditampilkan. Dalam *List Box* menu ditampilkan dalam bentuk daftar (*list*). Ilustrasi singkat dari *Pop-up Menu* dapat digambarkan pada Gambar berikut ini.



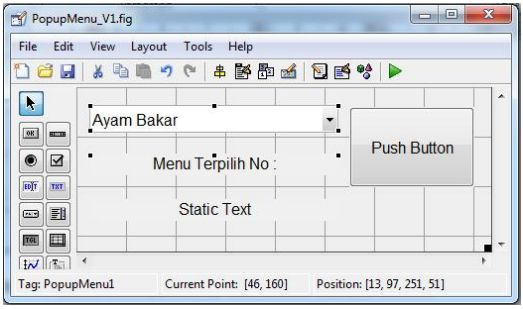
Gambar 3.16 Pop-up Menu

Untuk menambah daftar menu dapat dilakukan melalui *property* "*String*" sebagaimana pada komponen *List Box*.

Sedangkan untuk mengetahui pilihan atau menu yang dipilih oleh user dapat melalui *property* "*Value*".

Latihan 6

- Buat desain *layout* seperti Gambar di bawah ini.



Gambar 3.17 Tampilan Layout Latihan 6

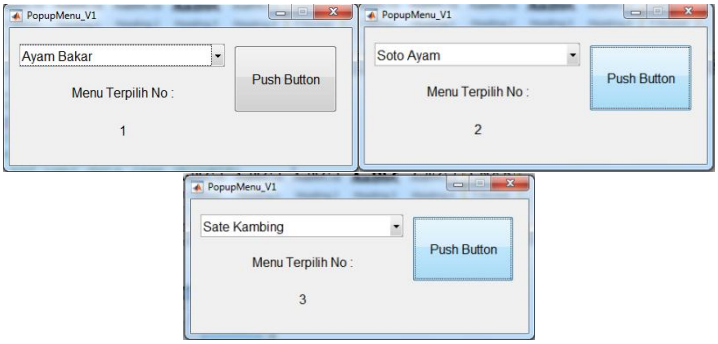
- Dengan *Property* untuk masing-masing komponen sebagai berikut:

Property	***Pop-up Menu***	***Static Text 1***	***Static Text 2***	***Push Button***
Font Size	12	12	12	12
String	"Ayam Bakar" "Soto Ayam" "Sate Kambing"	Menu Terpilih No :	Static Text	Push Button
Tag	PopupMenu 1	StaticText 1	StaticText 2	pushbutton 1

- Pada latihan ini menu yang dipilih oleh *user* akan ditampilkan no urutnya (dalam angka) setelah *Push Button* ditekan.
- Ketik program pada *Callback function* dari komponen *Push Button*, sebagai berikut:

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
Pilihan_Ke=get(handles.PopupMenu1,'Value');
set(handles.StaticText2,'String',Pilihan_Ke);
```

- Tampilan ketika program di-*running*:



Gambar 3.18 Tampilan *running* program Latihan 6

3.6. Komponen Slider

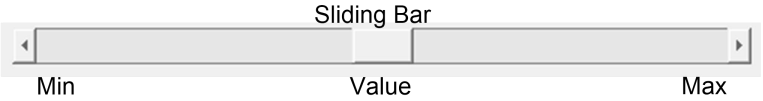
Komponen *Slider* adalah komponen GUI yang berfungsi menerima *input* dari *user* berupa nilai angka dengan cara menggerakkan *sliding bar* atau tanda panah. Nilai input

tersebut ditentukan dari letak dari *sliding bar* tersebut terhadap posisi minimal dan maksimum.

Property yang penting untuk diketahui dari komponen *Slider* antara lain:

- *Max* : nilai maksimum dari *sliding bar*.
- *Min* : nilai minimum dari *sliding bar*.
- *Value* : nilai saat ini dari *sliding bar*

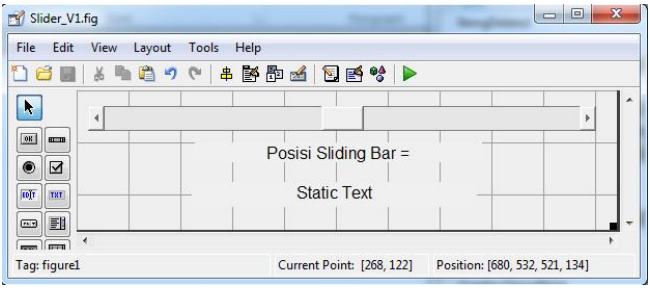
Ilustrasi dari *Property* di atas dapat digambarakan pada Gambar.



Gambar 3.19 Komponen *Slider*

Latihan 7

- Buat desain *layout* seperti Gambar di bawah ini



Gambar 3.20 Tampilan *Layout* Latihan 7

- Dengan *Property* untuk masing-masing komponen sebagai berikut:

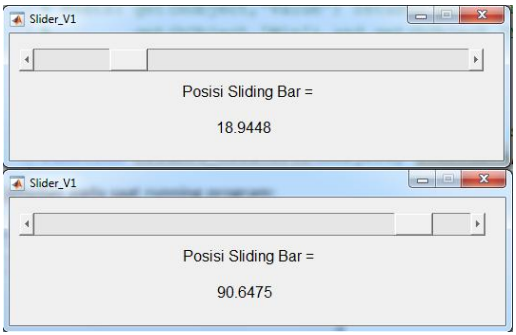
Property	***Slider***	***Static Text 1***	***Static Text 2***
Font Size	12	12	12
String	Slider	Posisi Sliding Bar =	Static Text
Tag	Slider1	StaticText1	StaticText2
Max	100		
Min	0		
Value	50		

- Pada latihan ini *user* akan menggerakkan *Sliding Bar*, dan posisi *Sliding Bar* akan ditampilkan pada *Static Text*.
- Ketik program pada *Callback function* dari komponen *Slider*, sebagai berikut:

```
function Slider1_Callback(hObject, eventdata, handles)
% hObject    handle to Slider1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
```

```
%       get(hObject,'Min') and get(hObject,'Max') to
determine range of slider
Nilai=get(handles.Slider1,'Value');
set(handles.StaticText2,'String',Nilai);
```

- Tampilan pada saat *running* program:

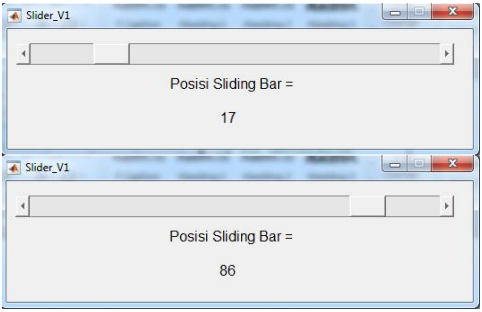


Gambar 3.21 Tampilan *running* program Latihan 7

- Default *property* "*Value*" dari *Sliding Bar* bertipe *Double* (pecahan) seperti tampak pada Gambar di atas. Untuk mengubah nilai *Sliding Bar* menjadi bilangan bulat (*integer*), maka dapat dilakukan konversi dari *double* ke *integer* (8 bit), dengan perintah *int8*, seperti :

```
function Slider1_Callback(hObject, eventdata, handles)
% hObject    handle to Slider1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%       get(hObject,'Min') and get(hObject,'Max') to
determine range of slider
Nilai=int8(get(handles.Slider1,'Value'));
set(handles.StaticText2,'String',Nilai);
```

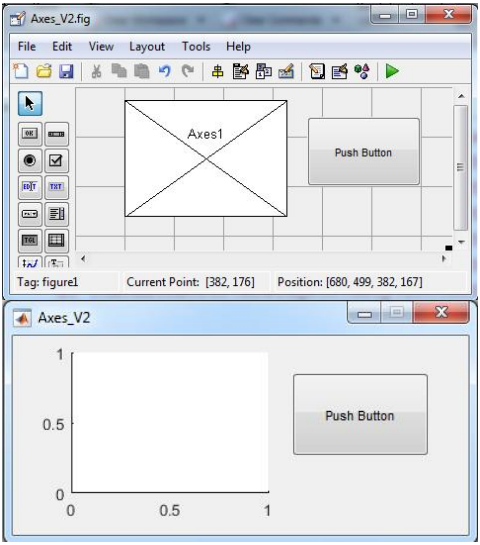
- Sehingga tampilan ketika running program:



Gambar 3.22 Tampilan *running* program komponen *Slider* dengan *value integer*

3.7. Komponen Axes

Komponen *Axes* adalah komponen untuk menampilkan grafik seperti *plot*. *Syntax* struktur perintah *plot* sama dengan *syntax* pada M-File.



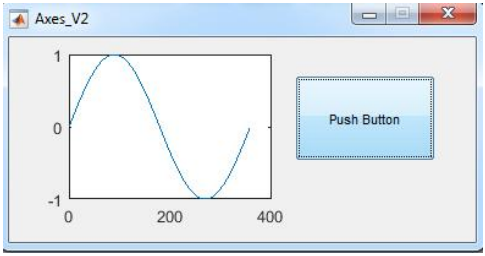
Gambar 3.23 Komponen *Axes* pada saat di *Layout* area dan *running* program

Contoh penggunaan dari komponen *Axes* yaitu:

- Ubah "*Tag*" pada *property inspector* menjadi: "*Axes1*".
- Buka *Callback* dari komponen *Push Button*, dan ketik:

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
axes(handles.Axes1);
t=0:1:360;
x=sind(t);
plot(t,x)
```

- Contoh program di atas adalah untuk menggambar grafik sinyal Sinus dari 0^0 sampai dengan 360^0, seperti terlihat pada Gambar berikut.



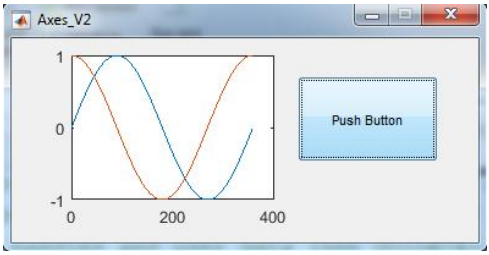
Gambar 3.24 Gambar sinyal sinus yang ditampilkan pada komponen *Axes*

- Untuk dapat menampilkan gambar sinyal lebih dari satu sinyal pada *Axes* yang sama, dapat dilakukan dengan cara menambah kode: "*hold on*", selengkapnya adalah sebagai berikut:

```
% --- Executes on button press in pushbutton1.
```

```
function pushbutton1_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
axes(handles.Axes1);
t=0:1:360;
x=sind(t);
plot(t,x)
hold on
x2=cosd(t);
plot(t,x2);
```

- Program tersebut di atas akan menampilkan dua buah sinyal yaitu sinyal sinus dan cosinus



Gambar 3.25 Gambar dua buah sinyal yang ditampilkan pada komponen *Axes*

- Apabila terdapat lebih dari satu komponen *Axes* dalam satu *Layout area*, maka untuk menentukan letak penggambaran (*plot*) pada *Axes* tertentu dapat menggunakan perintah:

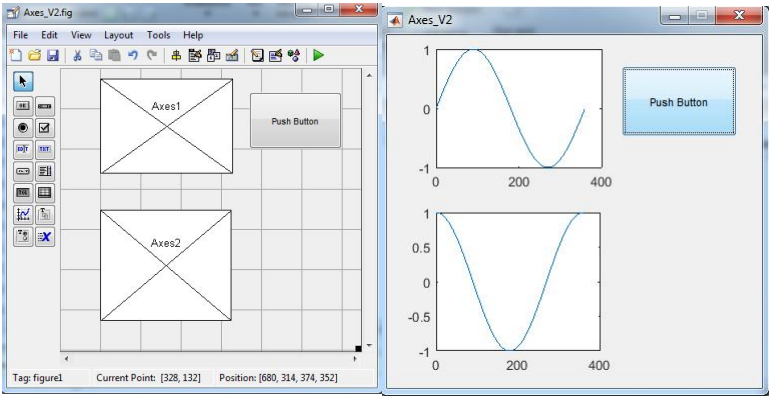
  axes(handles.Axes1);

- Perintah di atas menunjukkan bahwa komponen Axes yang ditunjuk untuk *plot* adalah "*Axes1*". "*Axes1*" merupakan *value* dari *property* "*Tag*". Contoh lengkap program untuk

penggambaran grafik dengan lebih dari satu komponen *Axes* adalah:

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata,
handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
axes(handles.Axes1);
t=0:1:360;
x=sind(t);
plot(t,x)

axes(handles.Axes2);
x2=cosd(t);
plot(t,x2);
```

- Program di atas digunakan untuk menggambar (*plot*) sinyal sinus pada "*Axes1*" dan sinyal cosinus pada "*Axes2*", seperti tampak pada Gambar.



Gambar 3.26 Menggambar *plot* grafik pada dua komponen *Axes*

BAB 4. DIALOG BOXES

Dialog Box adalah suatu tampilan yang dipergunakan untuk melaporkan status, meminta konfirmasi, memperingatkan atau menyediakan fasilitas pilihan untuk *user*. Sebagian besar *Dialog Box* sudah mempunyai desain *layout* tersendiri dan khas. Tampilan *Dialog Box* berada diluar dari tampilan *layout* GUI Matlab.

4.1. Error Dialo*g Box*

Error Dialog Box adalah *Dialog Box* yang berfungsi untuk menampilkan jika terdapat kesalahan (*error*) terjadi selama *running* program.

Syntax dari *Error Dialog Box* yaitu:

```
f = errordlg
f = errordlg(pesan)
f = errordlg(pesan,judul)
```

dengan contoh:

- f = errordlg()
- f = errordlg('Ada Kesalahan')
- f = errordlg('Ada Kesalahan','Pesan Kesalahan')

dimana contoh hasil dari perintah di atas yaitu:

Gambar 4.1Hasil *running Error Dialog Box*

4.2. Warning Dialog Box

Warning Dialog Box adalah *Dialog Box* yang berfungsi untuk memperingatkan *user*. *Syntax* dari *Warning Dialog Box* adalah:

```
f = warndlg
f = warndlg (pesan)
f = warndlg (pesan,judul)
```

dengan contoh:

- f = warndlg()
- f = warndlg('Ada Peringatan')
- f = warndlg('Ada Peringatan','Pesan Peringatan')

Contoh hasil tampilan *Warning Dialog Box* di atas adalah sebagai berikut:

Gambar 4.2 Hasil *running Warning Dialog Box*

4.3. Help Dialog

Help Dialog digunakan untuk membuat *dialog* bantuan, dengan *syntax*:

```
f=helpdlg
f=helpdlg(pesan)
f=helpdlg(pesan,judul)
```

dengan contoh:

- f = helpdlg()
- f = helpdlg('Mohon Bantuan')
- f = helpdlg('Mohon Bantuan','Bantuan')

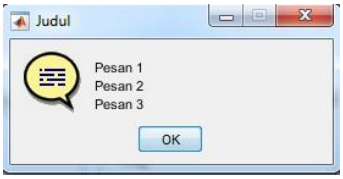
Dengan tampilan ketika *running* program:

Gambar 4.3 Tampilan *Help Dialog Box*

Jika diinginkan terdapat tampilan pesan lebih dari satu baris, maka dapat digunakan program:

```
f=helpdlg({'Pesan 1';'Pesan 2';'Pesan 3'},'Judul')
```

Dengan tampilan *output*:



Gambar 4.4 Tampilan *Help Dialog Box* dengan *multi-lines* pesan

4.4. Message Dialog Box

Message Dialog Box berfungsi untuk menyampaikan pesan (*message*) kepada *user*. *Syntax* dari *Message Dialog Box* adalah sebagai berikut:

```
f = msgbox(pesan)
f = msgbox(pesan,judul)
f = msgbox(pesan,judul,icon)
f = msgbox(pesan,judul,'custom',icondata,iconcmap)
```

Untuk menampilkan pesan lebih dari satu baris, maka dapat digunakan perintah:

```
f = msgbox({'Proses Telah'; 'Selesai'},'Judul')
```

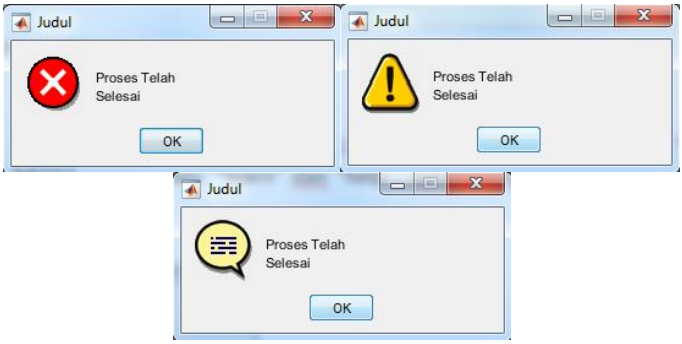
Contoh hasil tampilan *Message Dialog Box* di atas adalah sebagai berikut:

Gambar 4.5 Hasil tampilan *Message Box*

Message Box di atas tidak mempunyai *icon* pada tampilan *default*-nya, untuk menambah *icon* (*icon error*) dapat digunakan perintah:

```
f = msgbox({'Proses Telah'; 'Selesai'},'Judul','error')
```

Icon jenis lainnya yang disediakan antara lain: "*warn*" dan "*help*". Sehingga tampilan *Message Box* dengan *icon-icon* tersebut adalah:



Gambar 4.6 Tampilan *Message Box* dengan *icon* "*error*", "*warn*" dan "*help*"

Namun apabila diinginkan jenis *icon*/gambar lainnya selain *icon-icon* di atas dapat digunakan perintah:

```
f=msgbox({'Proses Telah';
'Selesai'},'Judul','custom',imread('gambar1.jpg'));
```

Perintah *imread* adalah perintah untuk membaca file gambar (*image read*), dalam hal ini file: "gambar1.jpg'. Tampilan *output* dari program di atas adalah sebagai berikut:

Gambar 4.7 Tampilan *Message Box* dengan *icon* berupa gambar

4.5. Wait Bar Dialog Box

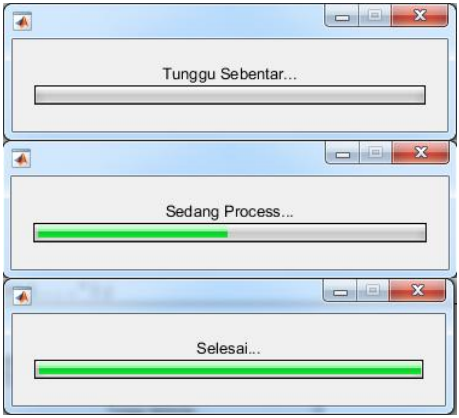
Wait Bar Dialog Box digunakan untuk menampilkan progress dari suatu proses, dengan *syntax*:

```
f = waitbar(x,pesan)
f = waitbar(x,pesan,Name,Value)
```

Dengan $0 \leq x \leq 1$, adalah nilai yang menunjukkan progress dari 0% hingga 100% (diwakili nilai 1). Contoh program *Wait Bar Dialog Box*:

```
▪ f = waitbar(0,'Tunggu Sebentar...');
▪ f = waitbar(0.5,'Sedang Process...');
▪ f = waitbar(1,'Selesai...');
```

Dengan tampilan program sebagai berikut:



Gambar 4.8 Tampilan *Wait Bar Dialog Box*

4.6. Question Dialog Box

Question Dialog Box digunakan sebagai konfirmasi pertanyaan dan jawaban yang diinginkan oleh *user*. *Default* jawaban dari

Question Dialog Box yaitu pilihan "*Yes*", "*No*" dan "*Cancel*" yang dapat dipilih oleh *user*.

Syntax dari *Question Dialog Box* yaitu:

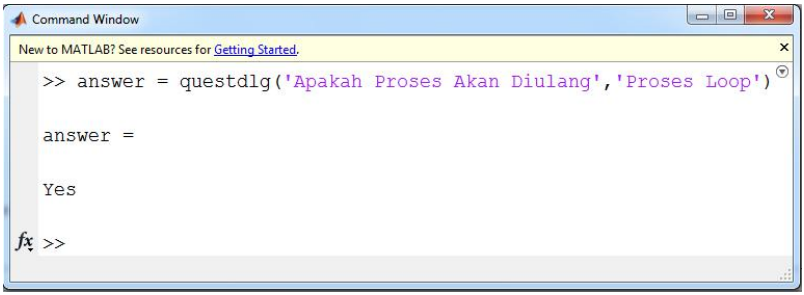
```
answer = questdlg(pertanyaan)
answer = questdlg(pertanyaan,judul)
answer = questdlg(pertanyaan,judul,default)
answer =
questdlg(pertanyaan,judul,tombol1,tombol2,default)
```

Jawaban dari *user* disimpan dalam *variable* "*answer*", contoh program adalah sebagai berikut:

- answer = questdlg('Apakah Proses Akan Diulang')
- answer= questdlg('Apakah Proses Akan Diulang','Proses Loop')

Gambar 4.9 *Question Dialog Box*

Variable "*answer*" akan bernilai "*Yes*" jika tombol *Yes* ditekan, bernilai "*No*" jika tombol *No* ditekan, dan bernilai "*Cancel*" jika tombol *Cancel* yang ditekan.



Gambar 4.10 Jawaban hasil klik tombol pada *Question Dialog Box*

Pada Gambar di atas, *default button* yang terpilih adalah tombol "*Yes*" dengan tanda warna biru dan *bold*. Jika diinginkan *default answer*-nya adalah tombol "*No*" atau "*Cancel*", maka kode program dapat ditambah:

- answer = questdlg('Apakah Proses Akan Diulang','Proses Loop','No')
- answer = questdlg('Apakah Proses Akan Diulang','Proses Loop','Cancel')

Sehingga tampilan program saat *running*:

Gambar 4.11 Tampilan *Question Dialog Box* Dengan *Default* Tombol Aktif "*No*" Dan "*Cancel*"

Jika diinginkan tombol pilihan jawaban bukan tombol "*Yes*", "*No*" atau "*Cancel*", tetapi tombol dengan tulisan yang kita *create* tersendiri, maka kode program dapat kita sebagai berikut:

- answer = questdlg('Apakah Proses Akan Diulang','Proses Loop','Tombol 1','Tombol 2','Tombol 3','Tombol 2')

Gambar 4.12 *Question Dialog Box* Dengan Modifikasi Tombol

Kode program di atas adalah untuk membuat Label pada tombol dengan tulisan "Tombol 1", "Tombol 2" dan "Tombol 3". Sedangkan tombol yang terpilih adalah "Tombol 2". *Variable answer* akan bernilai "Tombol 1", "Tombol 2" atau "Tombol 3" tergantung tombol yang dipilih oleh *user*.

4.7. Input Dialog Box

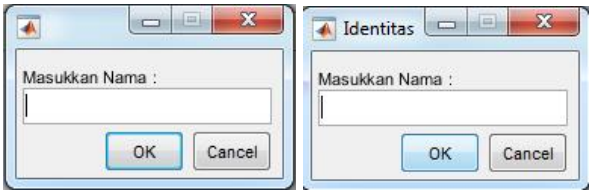
Input Dialog Box berfungsi untuk mengumpulkan informasi yang di-inputkan oleh *user* melalui *keyboard*. *Syntax* dari *Input Dialog Box* yaitu:

```
answer = inputdlg(pertanyaan)
answer = inputdlg(pertanyaan,judul)
answer = inputdlg(pertanyaan,judul,ukuran)
answer = inputdlg(pertanyaan,judul,ukuran,default)
```

Contoh kode program dari *Input Dialog Box* antara lain:

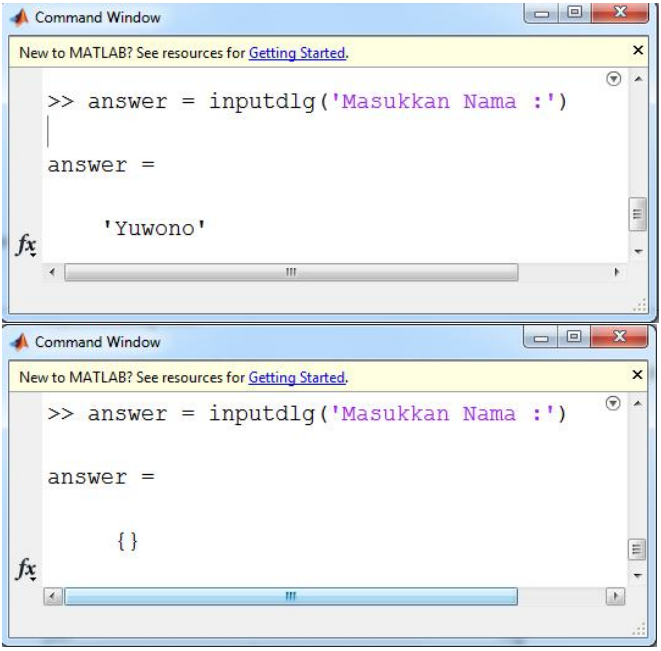
- answer = inputdlg('Masukkan Nama :')
- answer = inputdlg('Masukkan Nama :','Identitas')

Dan tampilan program ketika *running* adalah sebagai berikut:



Gambar 4.13 *Input Dialog Box*

Variable "*answer*" akan bernilai sesuai dengan input dari *user*. Jika *user* menekan tombol "*Cancel*", maka *variable* jawaban akan bernilai: "{ }" seperti tampak pada Gambar berikut ini.

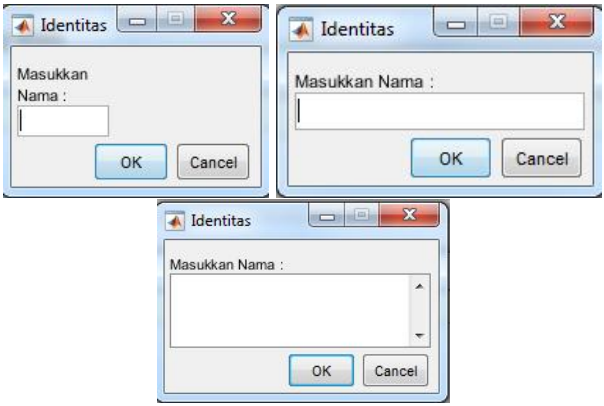


Gambar 4.14 Nilai Variable Dari Jawaban Pada *Input Dialog Box*

Ukuran dari *edit text* yang disediakan dapat kita ubah melalui parameter "ukuran", sebagai contoh:

- answer = inputdlg('Masukkan Nama :','Identitas',[1 10])
- answer = inputdlg('Masukkan Nama :','Identitas',[1 30])
- answer = inputdlg('Masukkan Nama :','Identitas',[3 30])

kode [1 10] menunjukkan bahwa *edit text* mempunyai ukuran height=1 dan width=10.

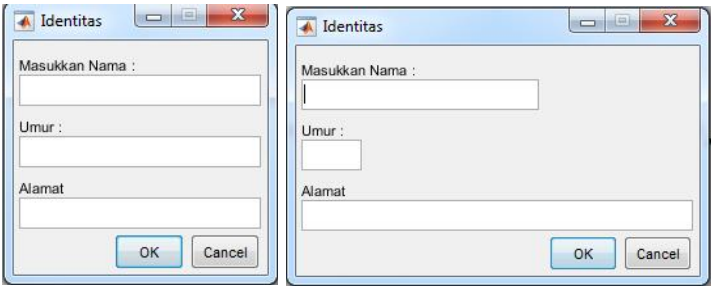


Gambar 4.15 *Input Dialog Box* Dengan Berbagai Ukuran *Edit Text*

Jika terdapat lebih dari satu pertanyaan yang harus dijawab oleh *user*, maka kode program dapat ditulis menjadi:

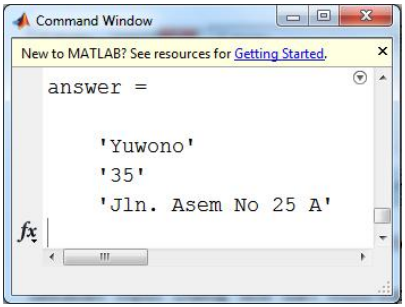
- answer = inputdlg({'Masukkan
  Nama :','Umur :','Alamat'},'Identitas',[1 30])
- answer = inputdlg({'Masukkan
  Nama :','Umur :','Alamat'},'Identitas',[1 30;1 7;1 50])

Dan tampilan *output* setelah *running* program adalah:



Gambar 4.16 *Input Dialog Box* dengan beberapa *Input Edit Text* dengan ukuran yang berbeda

Jawaban *Input Dialog Box* dari *multi-input* seperti Gambar di atas berbentuk *array*, sebagai berikut:

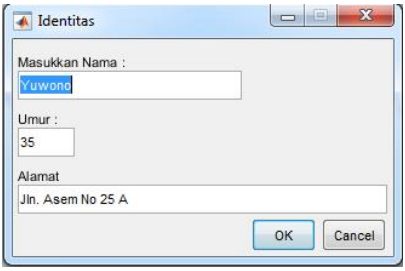


Gambar 4.17 *Variable output* dari *Input Dialog Box* dengan *multi-input*

Gambar-gambar *Input Dialog Box* di atas menunjukkan kolom input masih kosong dan belum terisi. Untuk mengisi nilai input secara *default* dapat digunakan program:

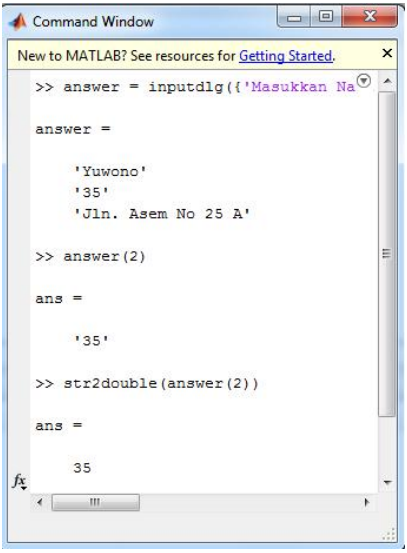
- answer = inputdlg({'Masukkan Nama :','Umur :','Alamat'},'Identitas',[1 30;1 7;1 50],{'Yuwono','35','Jln. Asem No 25 A'})

Dan tampilan ketika program running adalah sebagai berikut:



Gambar 4.18 *Input Dialog Box* dengan input sudah terisi secara default

Data yang diinputkan melalui *Input Dialog Box* bertipe data *string*. Apabila diperlukan konversi data, sebagai contoh dari tipe data *string* menjadi bilangan, maka dapat dapat digunakan perintah "*str2double*". Sebagai contoh nilai Jawaban "Umur" dari I*nput Dialog Box* di atas bernilai string '35', dan proses mengkonversi menjadi bilangan dapat digambarkan sebagai berikut:



Gambar 4.19 Konversi data dari *Input Dialog Box* (bertipe *string*) menjadi tipe bilangan (*double*)

4.8. List Dialog Box

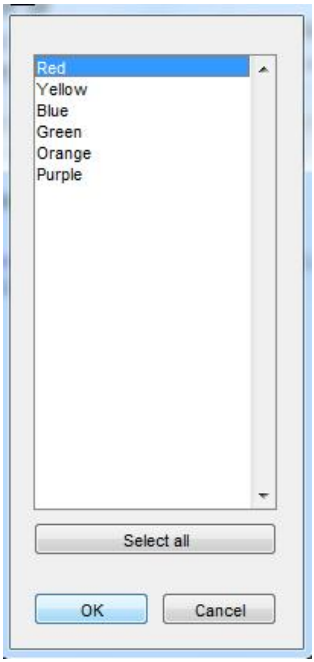
List Dialog Box berfungsi untuk menampilkan beberapa pilihan dalam bentuk *list* yang dapat dipilih oleh *user*. *user* dapat memilih lebih dari satu pilihan yang disediakan pada *List Dialog Box*. *Syntax* dari *List Dialog Box* yaitu:

```
[indx,tf] = listdlg('ListString',list)
[indx,tf] = listdlg('ListString',list,Name,Value)
```

Contoh kode program dari *List Dialog Box* antara lain:

- list = {'Red','Yellow','Blue','Green','Orange','Purple'};
- [indx,tf] = listdlg('ListString',list);

Setelah *running*, maka akan muncul



Gambar 4.20 *List Dialog Box*

Jika diinginkan *user* hanya boleh memilih satu pilihan saja, maka program dapat diubah menjadi:

- list = {'Red','Yellow','Blue','Green','Orange','Purple'};
- [indx,tf] = listdlg('SelectionMode','single','ListString',list)

Jika ada pilihan yang telah terpilih secara *default*, sebagai contoh pilihan nomer 2 telah terpilih pada saat awal program, maka program dapat dimodifikasi menjadi:

```
list = {'Red','Yellow','Blue','Green','Orange','Purple'};
[indx,tf] = listdlg('SelectionMode','single', ListString',list,
'InitialValue',2)
```

Output *arguments* dari contoh program di atas adalah:

- indx : berupa urutan dari jawaban yang dipilih oleh user. Pilihan jawaban yang paling atas bernilai 1 dan berurut sampai pada pilihan terakhir. Jika pilihan terpilih lebih dari satu, maka indx akan memberikan nilai juga lebih dari satu berupa deret atau array.
- tf : selection logical. tf akan bernilai satu jika button "OK" di klik dan bernilai 0 jika button "Cancel" yang ditekan.

4.9. Color Dialog Box

Digunakan untuk membuka dan memilih warna. *Syntax* dari *Color Dialog Box* yaitu:

```
c = uisetcolor
c = uisetcolor(title)
```

contoh tampilan ketika program

c = uisetcolor dan c = uisetcolor('Silahkan Dipilih Warnanya')

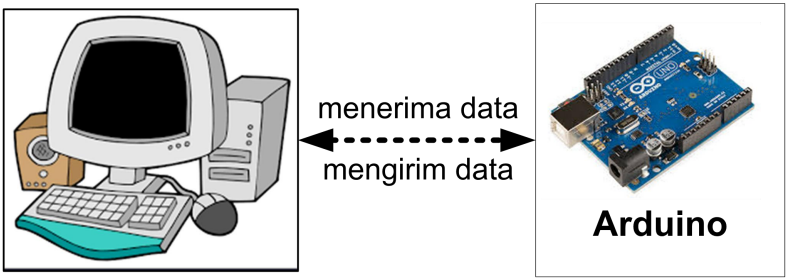
Gambar 4.21 *Color Dialog Box*

Output *argument* dari *Color Dialog Box* pada program di atas yaitu "c" berupa *array* nilai dari 0-1 kode RGB (*Red Green Blue*). Sebagai contoh:

- Merah, c = 1 0 0.
- Biru, c = 0 0 1.
- Kuning, c = 1 1 0. Karena warna kuning merupakan gabungan dari warna *Red* dan *Green*.
- Orange, c = 0.8706 0.4902 0

BAB 5. SERIAL COMMUNICATION DENGAN ARDUINO

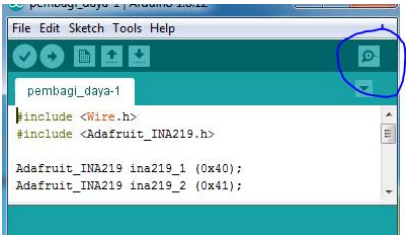
Pada Bab ini akan dibahas tentang tata cara koneksi hubungan atau komunikasi antara Matlab (di komputer) dengan Arduino. Dengan adanya komunikasi ini, maka proses yang dilakukan oleh Arduino dapat ditampilkan oleh GUI Matlab. Atau perintah yang di-inputkan oleh *user* melalui GUI Matlab dapat dieksekusi oleh Arduino. Komunikasi antara GUI Matlab dengan Arduino bersifat dua arah, artinya baik GUI Matlab maupun Arduino dapat saling menerima maupun mengirimkan data. Komunikasi ini dapat digambarkan pada Gambar sebagai berikut.



Gambar 5.1 Komunikasi dua arah antara *GUI Matlab* dan *Arduino*

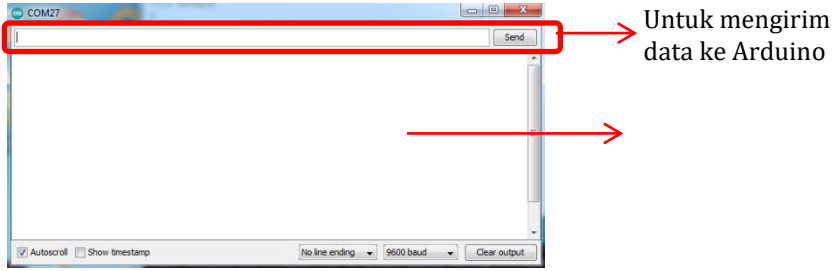
5.1. Serial Monitor

Software Integrated Development Environment (IDE) Arduino sudah menyediakan fasilitas untuk berkomunikasi secara dua arah antar komputer dan Arduino. IDE Arduino dapat menampilkan data yang dikirim oleh *hardware* Arduino maupun mengirimkan data yang akan diproses oleh Arduino. Fasilitas yang dapat digunakan pada IDE Arduino yaitu *Serial Monitor*, yang terletak pada bagian kanan atas IDE Arduino.



Gambar 5.2 *Serial Monitor* pada *software* IDE Arduino

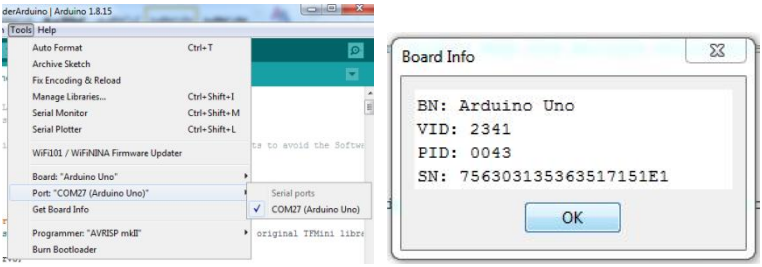
Setelah klik pada icon Serial Monitor, maka akan muncul tampilan Serial Monitor seperti pada Gambar 5.3.



Gambar 5.3 Tampilan *Serial Monitor Arduino IDE*

Untuk dapat menggunakan fasilitas Serial Monitor Arduino IDE, maka kita harus memastikan bahwa perangkat Arduino yang kita miliki telah terhubung / *connected* dengan komputer. Cara yang dapat dilakukan adalah:

- Sambungkan usb Arduino ke komputer.
- *running software* Arduino IDE.
- Klik menu “Tools”, “Board”, dan pilih board yang sesuai dengan perangkat Arduino kita, sebagai contoh “Arduino Uno”.
- Klik menu “Tools”, “Port:”, pilih port yang tersambung dengan perangkat Arduino, sebagai contoh : “COM27”.
- Untuk mengecek apakah koneksi Arduino kita dengan komputer sudah tersambung, klik “Tools”, “Get Board Info”. Apabila muncul *message box* sebagaimana Gambar 5.4, maka koneksi sudah tersambung dengan baik.



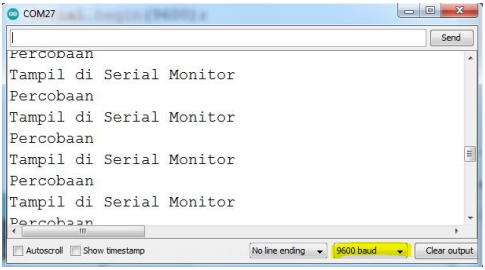
Gambar 5.4 Koneksi Arduino dengan komputer

5.2. Menampilkan data dari Arduino ke Serial Monitor

Untuk menampilkan data ke Serial Monitor, maka kita harus membuat kode programnya di Arduino, sebagai contoh:

```
void setup()
{
 Serial.begin(9600);
}
void loop()
{
 Serial.println("Tampil di Serial Monitor");
 Serial.println("Percobaan");
}
```

Upload program dan klik *Serial Monitor*, maka akan muncul

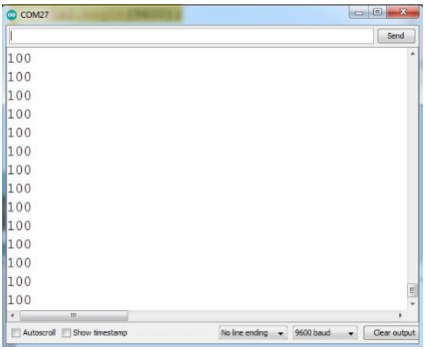


Gambar 5.5 Menampilkan data Arduino di *Serial Monitor*

Serial.begin(9600);
Untuk men-*set* kecepatan komunikasi (*baudrate*) antara Arduino dengan komputer, yaitu 9600bps. Kecepatan ini harus di-*set* sama antara yang diprogram maupun dengan yang ada di *Serial Monitor* (warna kuning di atas).

Serial.println("Tampil di Serial Monitor");
Adalah program untuk mengirimkan data dari Arduino (diantara dua tanda petik) ke Serial Monitor atau komunikasi serial lainnya yang terhubung.
Jika yang diinginkan ditampilkan data berupa bilangan dan bukan teks, maka program di Arduino adalah sebagai berikut:

```
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  int Nilai;
  Nilai=100;
  Serial.println(Nilai);
}
```



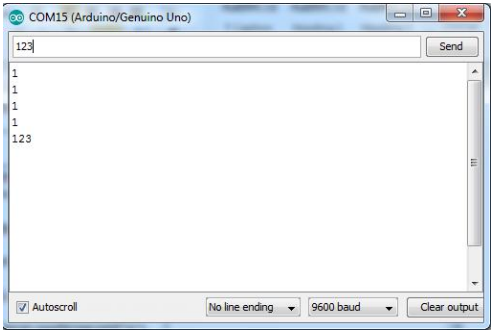
5.3. Arduino Menerima data dari Serial Monitor

Arduino dapat menerima input dari *Serial Monitor* atau perangkat elektronik lainnya yang terhubung secara serial dengannya. Program yang dapat digunakan sebagai contoh:

```
String Teks;
void setup()
{
  Serial.begin(9600);
}
```

```
void loop()
{
 if(Serial.available()>0)
 {
  Teks=Serial.readStringUntil('\n');
  Serial.println(Teks);
 }
 delay(100);
 Serial.flush();
}
```

Dengan hasil tampilan di *Serial Monitor* setelah *upload* adalah



Gambar 5.6 Arduino menerima input dari *Serial Monitor*

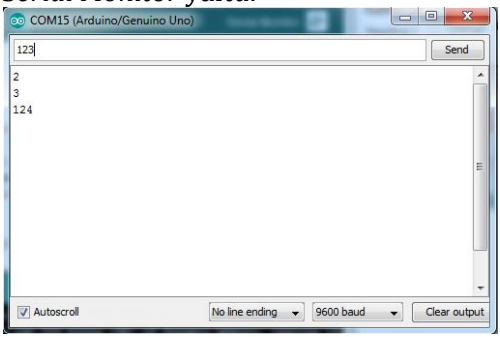
Teks=Serial.readStringUntil('\n');
Perintah di atas adalah untuk mengambil data yang di-*input* dari *Serial Monitor* (atau perangkat elektronik lainnya) via *serial communication*.

Jika data yang di-inputkan berupa data bilangan dan bukan bertipe teks, maka kita perlu melakukan konversi dari tipe data teks menjadi bilangan (sebagai contoh tipe integer). Program yang kita perlu buat yaitu:

```
String Teks;
void setup()
```

```
{
 Serial.begin(9600);
}
void loop()
{
 if(Serial.available()>0)
 {
  Teks=Serial.readStringUntil('\n');
  int Data=Teks.toInt()+1;
  Serial.println(Data);
 }
 delay(100);
 Serial.flush();}
```

int Data=Teks.toInt()+1;
adalah program untuk konversi *variable* "Teks" yang bertipe *string* (teks) menjadi bilangan (*integer*) dan disimpan dalam *variable* "Data". Setelah program di-*upload* dan *running*, tampilan di *Serial Monitor* yaitu:

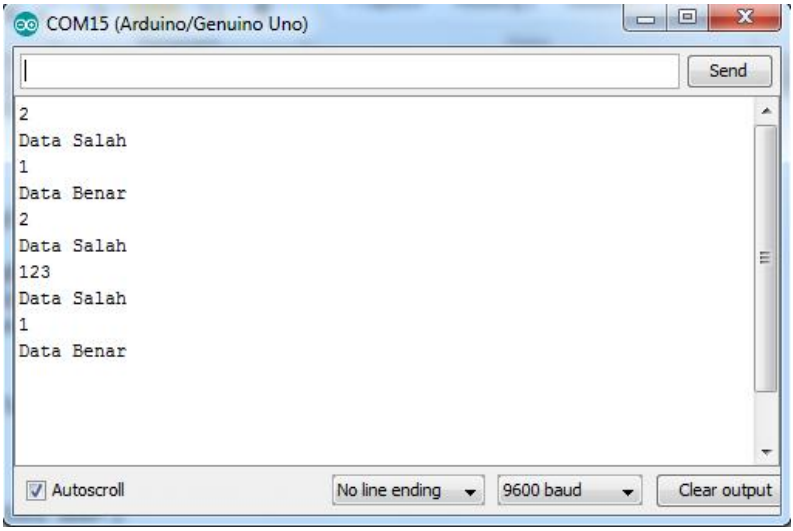


Gambar 5.7 Hasil konversi teks ke bilangan

- If Condition, jika input (dalam format bilangan) sesuai dengan kriteria yang diinginkan.

```
String Teks;
void setup() {
 Serial.begin(9600);
```

```
}
void loop()
{
 if(Serial.available()>0)
 {
  Teks=Serial.readStringUntil('\n');
  int Data=Teks.toInt();
  Serial.println(Data);
  if(Data==1)
  {
   Serial.println("Data Benar");
  }
  else
  {
   Serial.println("Data Salah");
  }
 }
 delay(100);
 Serial.flush();}
```



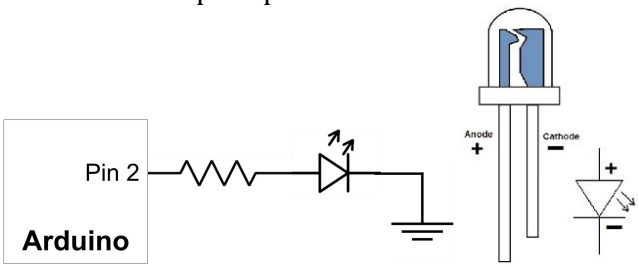
Gambar 5.8 Hasil konversi teks ke bilangan

- Atau, jika dalam bentuk string

```
String Teks;
void setup() {
 Serial.begin(9600);
}
void loop() {
 if(Serial.available()>0){
  Teks=Serial.readStringUntil('\n');
  Serial.println(Teks);
  if(Teks=="1"){
   Serial.println("Data Benar");
  }
  else
  {
   Serial.println("Data Salah");
  }
 }
 delay(100);
 Serial.flush();}
```

5.4. On-Off LED

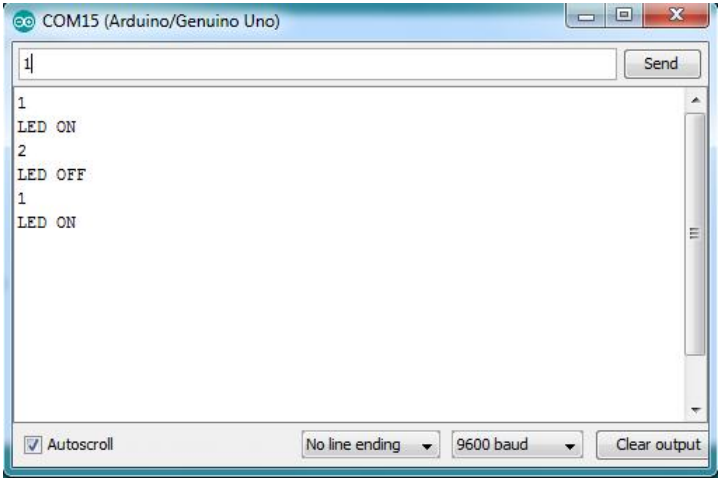
Untuk mengaktifkan LED (on atau off), maka kita perlu sambungkan (koneksi) dari Arduino ke LED. Sebagai contoh kita gunakan pin 2 pada Arduino dan disambungkan ke resistor dan kemudian LED seperti pada Gambar 5.9.



Gambar 5.9 koneksi sambungan antara LED dan Arduino

Sedangkan programnya adalah berikut:

```
String Teks;
void setup() {
  Serial.begin(9600);
  pinMode(2,OUTPUT);
}
void loop() {
  if(Serial.available()>0){
    Teks=Serial.readStringUntil('\n');
    int Data=Teks.toInt();
    Serial.println(Data);
    if(Data==1){
      Serial.println("LED ON");
      digitalWrite(2,HIGH); }
    else{
      Serial.println("LED OFF");
      digitalWrite(2,LOW); }
  }
  delay(100);
  Serial.flush();}
```



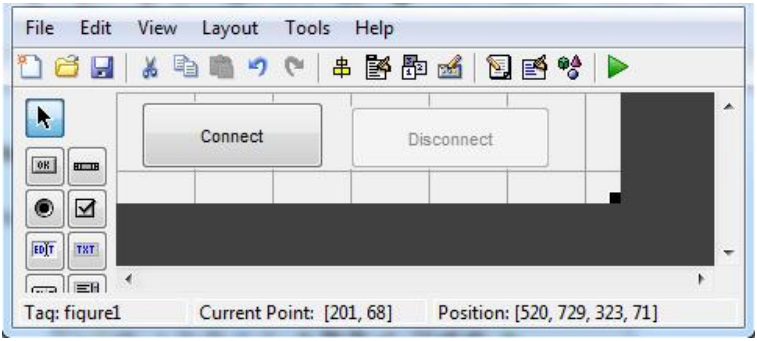
Gambar 5.10 Hasil program on-off LED

- pinMode(X,mode) : untuk mem-setting pin X sebagai Input atau Output (mode).
- digitalWrite(X,status): untuk mengeluarkan (output) pin X dengan status High atau Low.

5.5. Serial Communication Arduino - Matlab

Setelah membuat program komunikasi dari *Serial Monitor* Arduino dengan LED, maka saat ini kita akan membuat program komunikasi antara Matlab GUI dengan Arduino (dapat kita gunakan juga untuk komunikasi dengan *processor* lainnya seperti Raspberry).
Program yang sudah kita buat di Arduino pada prinsipnya dapat langsung kita pakai untuk komunikasi dengan Matlab.

- Buat Program dengan *component* di *Graphical user Interface* Matlab seperti Gambar 5.11.



Gambar 5.11 Tampilan GUI Matlab untuk komunikasi dengan Arduino

- Pada *Callback* "Connect", buat program:

```
% --- Executes on button press in Connect.
function Connect_Callback(hObject, eventdata, handles)
% hObject    handle to Connect (see GCBO)
```

```
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
global s;
s = serial('COM15','Baudrate',9600);
fopen(s);

set(handles.Disconnect,'Enable','on');
set(handles.Connect,'Enable','off');
```

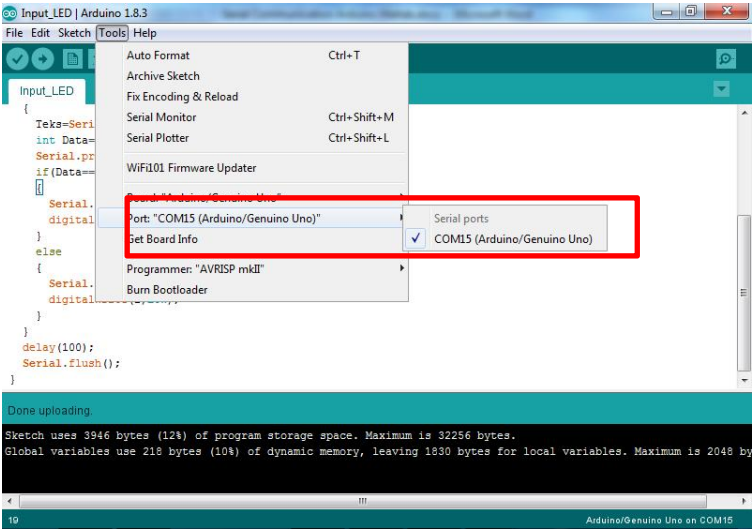
- Pada *Callback* "*Disconnect*":

```
% --- Executes on button press in CloseConnection.
function CloseConnection_Callback(hObject, eventdata,
handles)
% hObject    handle to CloseConnection (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
global s;
fclose(s);
instrreset;

set(handles.Disconnect,'Enable','off');
set(handles.Connect,'Enable','on');
```

- *global* s;
  - adalah untuk mendeklarasikan variable s secara global dan sebagai variable parameter komunikasi.
- s = serial('COM15','Baudrate',9600);
  - COM15 diganti dengan *Serial Communication Port* yang digunakan saat ini oleh Arduino.

- 9600 adalah baudrate yang kita gunakan dan harus sama dengan setting kita di Serial Monitor, sebagaimana ditunjukkan pada Gambar 5.12.

- fopen(s); fclose(s);
  - untuk membuka dan menutup koneksi antara Matlab dan Arduino.



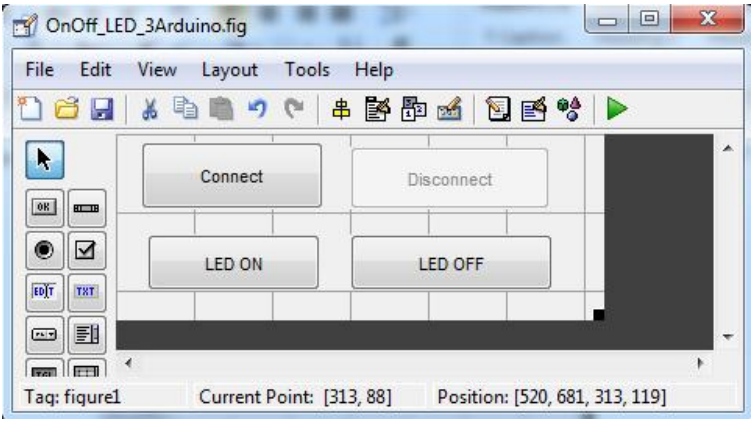
Gambar 5.12 *Com Port Serial Monitor*

- *running* program GUI Matlab
- Amati di *Command Window* Matlab, apakah ada *Error*?.
- Klik "*Connect*" *button*, amati *Command Window* Matlab jika ada *Error*.
- Setelah itu klik "Disconnect" *button*. Urutan harus berurut dan berpasangan dari *Connect* ke *Disconnect*, jika tidak akan muncul *error*.

5.6. Serial Communication Arduino - Matlab 2

Kita lanjutkan program di atas untuk meng-*on* dan *off* kan LED dengan *button* dari GUI Matlab.

- Buat *button* lagi untuk : LED ON dan LED OFF, dengan tag: "LED_ON" dan "LED_OFF" seperti pada Gambar 5.13



Gambar 5.13 Tampilan GUI Matlab LED ON dan LED OFF

- Buat Program pada Callback "LED ON" dan "LED OFF":

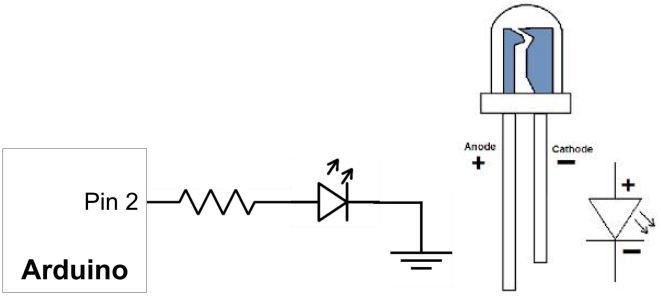
```
% --- Executes on button press in LED_ON.
function LED_ON_Callback(hObject, eventdata, handles)
% hObject    handle to LED_ON (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
global s;
fwrite(s,'1');
pause(0.5);

% --- Executes on button press in LED_OFF.
function LED_OFF_Callback(hObject, eventdata, handles)
% hObject    handle to LED_OFF (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
```

```
% handles    structure with handles and user data (see
GUIDATA)
global s;
fwrite(s,'2');
pause(0.5);
```

- Fungsi "*fwrite*" adalah untuk mengirimkan data melalui *Serial Communication*.
- Pada Arduino *upload* program sebagai berikut:

```
String Teks;
void setup()
{
 Serial.begin(9600);
 pinMode(2,OUTPUT);
}
void loop()
{
 if(Serial.available()>0)
 {
  Teks=Serial.readStringUntil('\n');
  int Data=Teks.toInt();
  Serial.println(Teks);
  if(Data==1)
  {
   digitalWrite(2,HIGH);
  }
  else
  {
   digitalWrite(2,LOW);
  }
 }
 delay(100);
 Serial.flush();
}
```

- Pasang koneksi LED dengan Arduino pada pin 2 sebagaimana telah dijelaskan pada Gambar 5.9 dan digambarkan Kembali di Gambar 5.14.
- *Serial Monitor* Arduino harus dalam posisi di-*close*, karena *serial communication port* akan digunakan oleh Matlab.
- Program antara GUI Matlab dan Arduino harus disinkronisasi terlebih dahulu. Sebagai contoh pada program di atas, sinkronisasi data adalah sebagai berikut:
    - Perintah LED On (dengan button LED ON) yaitu data terkirim/diterima = ‘‘1”;
      fwrite(s,'1'); (pada Matlab) dan
      if(Data==1) (pada Arduino)
    - Perintah LED Off, yaitu data terkirim/diterima = “2”
- Jika data tidak disinkronisasi, maka masing-masing Matlab dan Arduino dapat saling kirim/terima data, tetapi tidak saling mengerti/memahami apa yang diinginkan.
- Ketika button “LED ON” ditekan, maka Matlab akan mengirimkan data beruapa angka “1”. Jika button “LED OFF” ditekan, maka Matlab mengirimkan data angka “2”.
- Program Arduino akan mendeteksi input, jika input berupa angka “1”, maka mengirimkan sinyal “*high*” ke pin 2 (artinya LED akan o*n*). Jika input berupa angka “2”, maka pin 2 akan di-*set* “*low*”.
    -



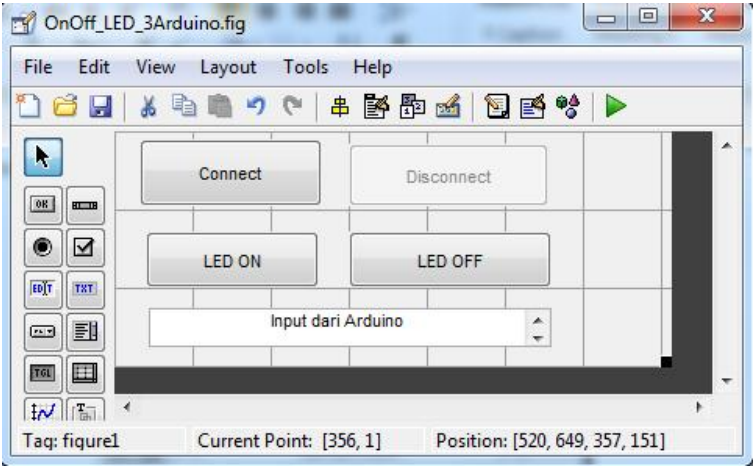
Gambar 5.14 Koneksi LED dan Arduino

- Running program GUI Matlab
- Klik "*Connect*", klik "LED ON", amati pada LED apakah *On*?.
- Klik "*LED OFF*", amati pada LED apakah *Off*?..
- Ulangi klik "*LED ON*" atau "*LED OFF*", dan amati.
- Klik "*Disconnect*" jika sudah selesai.

5.7. Mengambil/Mendeteksi data dari Arduino

Contoh Program di atas adalah untuk mengirimkan data dari Matlab ke Arduino, namun adakalanya kita membutuhkan data yang dikirimkan oleh Arduino dan diterima oleh Matlab, sebagai contoh pembacaan sensor seperti sensor suhu, *temperature*, kelembaban dan lainnya.

- Tambahkan komponen Edit pada GUI Matlab, dengan Tag: "edit_arduino" dan String: "Input dari Arduino".



Gambar 5.15 Tampilan GUI Matlab untuk ambil data dari Arduino

- Tambahkan program pada Callback "LED_ON" dan "LED_OFF":

```
% --- Executes on button press in LED_ON.
function LED_ON_Callback(hObject, eventdata, handles)
% hObject    handle to LED_ON (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
global s;
fwrite(s,'1');
pause(0.1);
Data1=fscanf(s);
pause(0.1);
set(handles.edit_arduino,'String',Data1);

% --- Executes on button press in LED_OFF.
function LED_OFF_Callback(hObject, eventdata, handles)
% hObject    handle to LED_OFF (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
global s;
fwrite(s,'2');
pause(0.1);
Data1=fscanf(s);
pause(0.1);
set(handles.edit_arduino,'String',Data1);
```

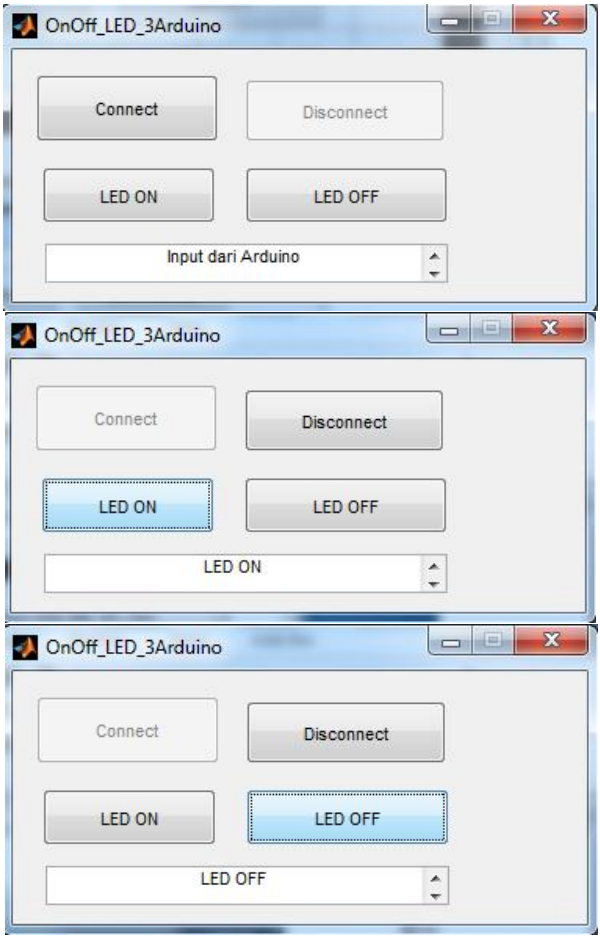
- *Upload* program pada Arduino:

```
String Teks;
void setup()
{
  Serial.begin(9600);
  pinMode(2,OUTPUT);
}
```

```
void loop()
{
 if(Serial.available()>0)
 {
  Teks=Serial.readStringUntil('\n');
  int Data=Teks.toInt();
  //Serial.println(Teks);          //di non-
aktifkan
  if(Data==1)
  {
   digitalWrite(2,HIGH);
   Serial.println("LED ON");
  }
  else
  {
   digitalWrite(2,LOW);
   Serial.println("LED OFF");
  }
 }
 delay(100);
 Serial.flush();
}
```

- *Running* program GUI Matlab.
- Klik “Connect”
- Klik “LED ON” atau “LED OFF”, amati status LED (on atau off) dan tampilan *text* di *Edit* GUI Matlab.
- Setelah selesai Klik “Disconnect”.
- Sinkronisasi dilakukan dengan cara:
    - LED On : data = “1”, dan LED Off : data = “2”
    - pause(0.1); pada Matlab adalah *delay* selama 0.1 *second*
    - Setelah menerima data dari Matlab, maka Arduino akan balas mengirimkan data berupa teks “LED ON” atau “LED OFF”, yaitu : Serial.println("LED ON");

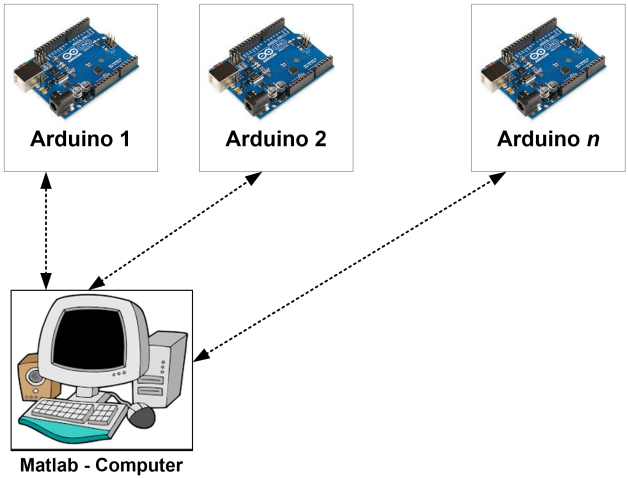
- Hasil dari program di atas dapat ditunjukkan pada Gambar 5.16



Gambar 5.16 Hasil program mengambil data dari Arduino

5.8. *Serial Communication* dengan 2 atau lebih Arduino dan Matlab

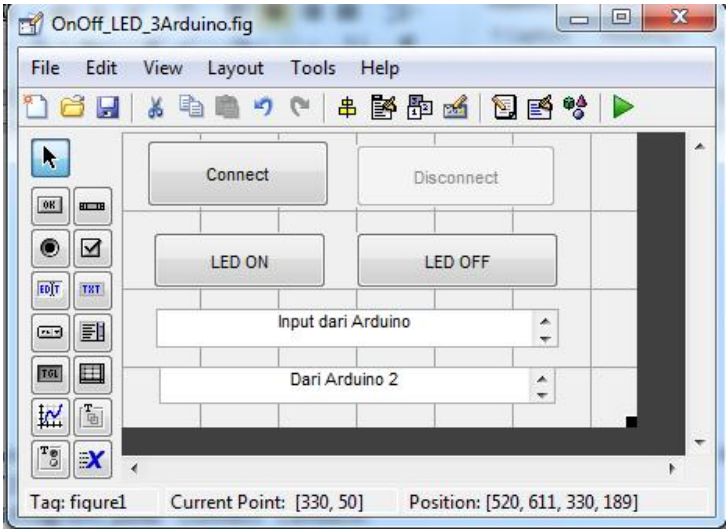
Contoh di atas adalah program *serial communication* antara Matlab dan Arduino. Jika diinginkan ada beberapa Arduino berkomunikasi dengan satu Matlab, maka dapat dilakukan sebagaimana pada Sub-Bab ini.



Gambar 5.17 Ilustrasi *Serial Communication* antara Matlab dan beberapa Arduino

Langkah yang dilakukan antara lain:

- Koneksikan dua Arduino dengan *computer*, dan dapatkan COM-nya dan *setting baudrate*-nya ke 9600.
- Modifikasi GUI Matlab pada Latihan sebelumnya menjadi seperti Gambar 5.18.
- Tambahkan 1 komponen *Edit*: dengan Tag:"edit_arduino2", dan *String*: "Dari Arduino 2".



Gambar 5.18 *Layout* GUI Matlab untuk Komunikasi Dua Arduino

- Modifikasi Program, pada "Connect" *Callback*:

```
% --- Executes on button press in OpenConnection.
Function OpenConnection_Callback(hObject, eventdata,
handles)
% hObject    handle to OpenConnection (see GCBO)
% eventdata  reserved – to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
global s;
s = serial('COM15','Baudrate',9600);
fopen(s);

global s2;
s2 = serial('COM17','Baudrate',9600);
fopen(s2);
```

- Modifikasi Program, pada "Disconnect" *Callback*:

```
% --- Executes on button press in CloseConnection.
function CloseConnection_Callback(hObject, eventdata,
handles)
% hObject    handle to CloseConnection (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
global s;
fclose(s);
global s2;
fclose(s2);
instrreset;
```

s = serial('COM15','Baudrate',9600); adalah untuk Arduino 1
s2 = serial('COM17','Baudrate',9600); adalah untuk Arduino 2

- Buat program pada "LED ON" *Callback pushbutton*:

```
% --- Executes on button press in LED_ON.
function LED_ON_Callback(hObject, eventdata, handles)
% hObject    handle to LED_ON (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
global s;
fwrite(s,'1');
pause(0.1);
Data1=fscanf(s);
pause(0.1);
set(handles.edit_arduino,'String',Data1);
```

```
global s2;
fwrite(s2,'1');
pause(0.1);
Data2=fscanf(s2);
pause(0.1);
set(handles.edit_arduino2,'String',Data2);
```

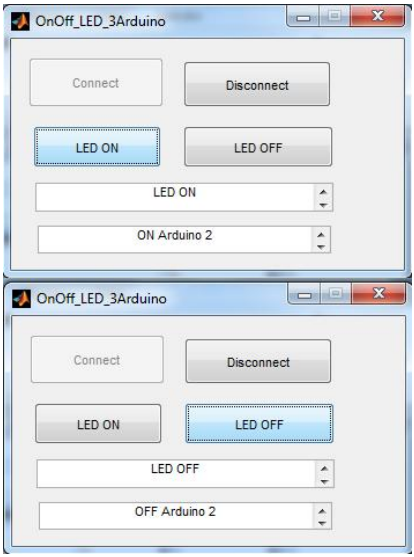
- Buat Program pada "LED OFF" *Callback pushbutton*:

```
% --- Executes on button press in LED_OFF.
function LED_OFF_Callback(hObject, eventdata, handles)
% hObject    handle to LED_OFF (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
global s;
fwrite(s,'2');
pause(0.1);
Data1=fscanf(s);
pause(0.1);
set(handles.edit_arduino,'String',Data1);

global s2;
fwrite(s2,'2');
pause(0.1);
Data2=fscanf(s2);
pause(0.1);
set(handles.edit_arduino2,'String',Data2);
```

- Jangan lupa bahwa program pada Arduino ke 1 sama dengan program on-off LED sebagaimana program *serial communication* sebelumnya.
- Untuk program pada Arduino 2, buat program sebagaimana berikut ini:

```
String Teks;
void setup()
{
 Serial.begin(9600);
 pinMode(2,OUTPUT);
}
void loop()
{
 if(Serial.available()>0)
 {
  Teks=Serial.readStringUntil('\n');
  int Data=Teks.toInt();
  if(Data==1)
  {
   digitalWrite(2,HIGH);
   Serial.println("ON Arduino 2");
  }
  else
  {
   digitalWrite(2,LOW);
   Serial.println("OFF Arduino 2");
  }
 }
 delay(100);
 Serial.flush();
}
```

- *Running* GUI Matlab
- Klik "Connect" *button*.
- Klik "LED ON" atau "LED OFF" *button*.
- Jika sudah selesai klik "Disconnect"

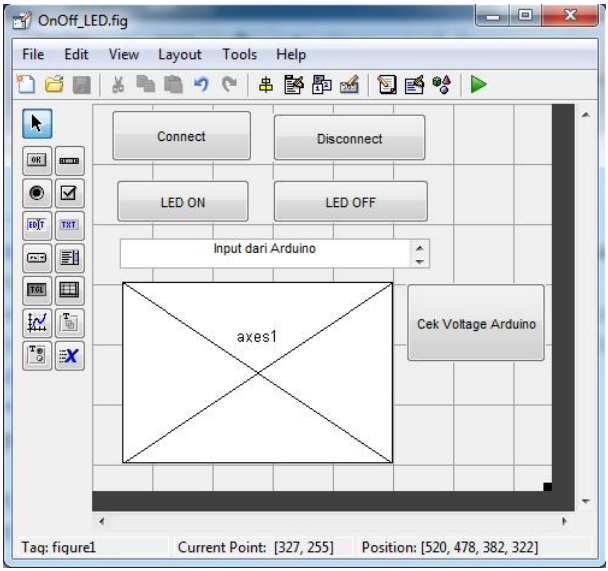


Gambar 5.19 Tampilan GUI Matlab ambil data dari 2 Arduino

5.9. Plotting Data dari Arduino

Jika data yang diambil dari Arduino berupa bilangan dan bersifat deret atau data yang dikumpulkan berdasarkan waktu, seperti suhu, kelembaban, tegangan, arus dan lainnya, maka data tersebut dapat ditampilkan dalam bentuk grafik atau plotting oleh Matlab.
Langkah-langkah yang diperlukan yaitu:

- Modifikasi program GUI Matlab menjadi:



Gambar 5.20 *Layout* GUI Matlab untuk *plotting* data dari Arduino

- Tambahkan komponen *button*, dengan *Tag*:"CekVoltage", dan *String*: "Cek Voltage Arduino".
- Tambahkan komponen *Axes*, dengan *Tag*:"axes1".
- Modifikasi program pada *Callback* "CekVoltage" *button*:

```
% --- Executes on button press in CekVoltage.
function CekVoltage_Callback(hObject, eventdata,
handles)
% hObject    handle to CekVoltage (see GCBO)
% eventdata  reserved - to be defined in a future
version of MATLAB
% handles    structure with handles and user data (see
GUIDATA)
global s;
set(handles.edit_arduino,'String','Cek data tegangan
dari Arduino....');
for i=1:1:20
```

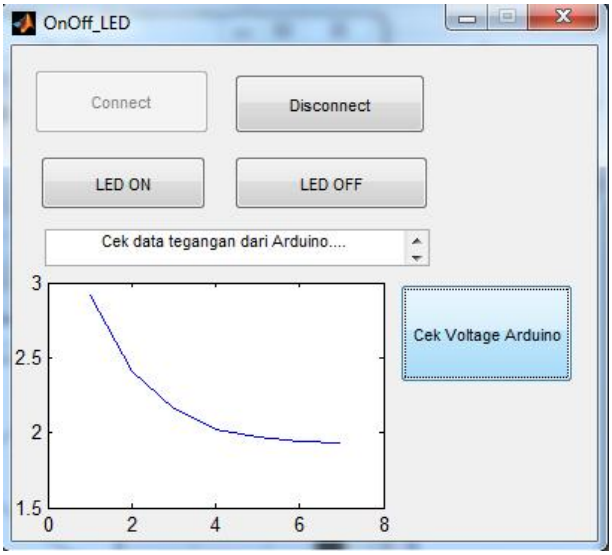
```
    fwrite(s,'1');
    Data1=fscanf(s);
    Data2(i)=str2num(Data1);
    X(i)=i;
    axes(handles.axes1);
    plot(X,Data2);
    hold on;
    pause(0.01);
end
```

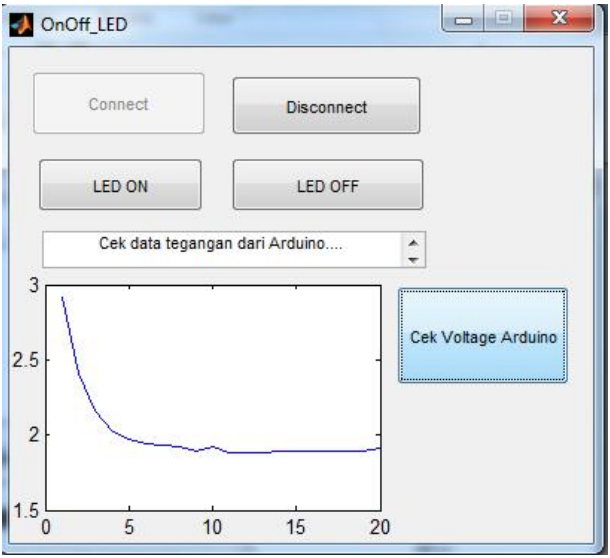
- *Code* "axes(handles.axes1);" adalah untuk mengaktifkan bahwa plot yang aktif akan ditampilkan pada "axes1".
- Data2(i) dan X(i) adalah *array* (sekumpulan data) agar dapat ditampilkan dalam plot secara bersambung.
- str2num untuk mengkonversi dari tipe *string* menjadi *number* (*double*).
- Program Matlab di atas digunakan untuk mengambil data dari Arduino sebanyak 20 kali dengan interval 0.01 *second* per-data.
- Untuk program di Arduino adalah sebagai berikut:

```
String Teks;
void setup()
{
  Serial.begin(9600);
  pinMode(2,OUTPUT);
}
void loop()
{
  if(Serial.available()>0)
  {
    Teks=Serial.readStringUntil('\n');
    int Data=Teks.toInt();
    if(Data==1)
    {
      int sensorValue = analogRead(A0);
```

```
    float voltage = sensorValue * (5.0 / 1023.0);
    Serial.println(voltage);
   }
  }
  delay(10);
  Serial.flush();
}
```

- Program di atas mengambil besaran konversi ADC dari *port* Analog 0 (pin A0) dan diubah dalam *range* 0-5 Volt.
- *Running* program, klik "Connect", klik "Cek Voltage Arduino", amati..
- Klik "Disconnect" jika sudah selesai.





Gambar 5.21Tampilan hasil *plotting* data dari Arduino

DAFTAR PUSTAKA

Colgren, R. (2014). *MATLAB Numerical Computing*. Tutoria Point. https://doi.org/10.2514/5.9781600861628.0001.0042

Houcque, D. (2005). Introduction To Matlab for Engineering Students. In *School of Engineering & Applied Science* (Issue August). Northwestern University.

Serhat Beyenir. (2013). A Brief Introduction to Engineering Computation with MATLAB. Connexions, Rice University.

INDEX

Q

R

S

T

U

V

W

GLOSSARY

Axes adalah komponen untuk menampilkan grafik seperti *plot*.

Button Group adalah *panel* (pengumpul) dari beberapa komponen *Radio Button*. Beberapa *Radio Button* merupakan satu kumpulan alternatif pilihan bagi *user*, sehingga dikumpulkan dalam satu grup yang disebut dengan *Button Group.*

Check Box sama seperti komponen *Radio Button* yaitu komponen yang menyediakan alternatif beberapa pilihan untuk *user*. Tetapi perbedaannya terletak pada jumlah pilihan yang dapat diberikan oleh *user*. Pada komponen *Radio Button*, hanya ada satu jawaban yang dapat dipilih. Sedangkan pada komponen *Check Box*, jumlah pilihan dapat lebih dari satu pilihan.

Color Dialog Box digunakan untuk membuka dan memilih warna.

Component pallete adalah *component - component GUI* yang disediakan oleh Matlab.

Dialog Box adalah suatu tampilan yang dipergunakan untuk melaporkan status, meminta konfirmasi, memperingatkan atau menyediakan fasilitas pilihan untuk *user*.

Edit Text digunakan untuk menampilkan tulisan kepada *user* sebagaimana komponen *Static Text*. Namun perbedaannya adalah pada komponen *Edit Text*, *user* dapat mengubah atau memasukkan tulisan yang diinginkan.

Error Dialog Box adalah *Dialog Box* yang berfungsi untuk menampilkan jika terdapat kesalahan (*error*) terjadi selama *running* program.

Graphical user Interface (GUI) Matlab adalah fasilitas interaktif yang disediakan oleh Matlab untuk programming berbasis *object.*

Help Dialog digunakan untuk membuat *dialog* bantuan.

Input Dialog Box berfungsi untuk mengumpulkan informasi yang di-inputkan oleh *user* melalui *keyboard*.

List Dialog Box berfungsi untuk menampilkan beberapa pilihan dalam bentuk *list* yang dapat dipilih oleh *user*.

List Menu digunakan untuk menampilkan daftar menu atau pilihan. *user* dapat memilih salah satu menu yang disediakan dengan klik pada pilihan menu yang disediakan atau menekan tombol keatas atau kebawah.

Matlab berasal dari kata Matrix Laboratory dan dibuat oleh The MathWorks (Houcque, 2005). Matlab adalah *high-level programming* dan memiliki fasilitas yang interaktif untuk visualisasi, komputasi dan programming.

Message Dialog Box berfungsi untuk menyampaikan pesan (*message*) kepada *user*.

Panel adalah untuk pengelompokkan (grup) dari komponen *Check Box*. Mempunyai fungsi yang sama dengan *Button Group* untuk komponen *Radio Button.*

Pop-up Menu mempunyai fungsi yang sama dengan komponen *List Box*, yaitu menampilkan daftar pilihan menu atau pilihan. Tetapi pada komponen *Pop-up Menu*, pilihan menu ditampilkan dalam bentuk *pop-up* dimana tidak semua menu ditampilkan. Dalam *List Box* menu ditampilkan dalam bentuk daftar (*list*).

Property inspector terdiri dari dua parameter yaitu *Property* (kolom sebelah kiri) dan *Value* (kolom sebelah kanan). *Property* menunjukkan jenis parameter, sedangkan *Value* menunjukkan nilai atau isi dari *property* atau parameter tersebut.

Push Button adalah *component* GUI Matlab jenis tombol yang dapat ditekan atau di-klik.

Question Dialog Box digunakan sebagai konfirmasi pertanyaan dan jawaban yang diinginkan oleh *user*.

Radio Button adalah komponen GUI Matlab untuk memberikan alternatif dari beberapa kemungkinan yang dapat dipilih oleh *user*.

Slider adalah komponen GUI yang berfungsi menerima *input* dari *user* berupa nilai angka dengan cara menggerakkan *sliding bar* atau tanda panah.

Static Text adalah komponen yang dipergunakan untuk menampilkan *text* ke *Layout* area.

Wait Bar Dialog Box digunakan untuk menampilkan progress dari suatu proses.

Warning Dialog Box adalah *Dialog Box* yang berfungsi untuk memperingatkan *user*.

TENTANG PENULIS

Penulis adalah seorang dosen tetap di Jurusan Teknik Elektronika dan Informatika, Politeknik Manufaktur Negeri Bangka Belitung.

Penulis lulusan Sarjana Terapan Politeknik Elektronika Negeri Surabaya (ITS), *University of* Wollongong - Australia (S2) dan Curtin *University* - Australia (S3). Seluruh biaya pendidikan penulis dibiayai melalui Beasiswa baik dalam maupun luar negeri.

Keahlian penulis adalah pada bidang algoritma, *control system*, *automation* dan *smart grid*.

Buku ini membahas tentang Graphical User Interface (GUI) Matlab dimulai dari dasar-dasar Matlab, komponen-komponen GUI Matlab seperti *Push Button, Slider, Radio Button, Check Box, Edit Text, Static Text, Pop-up Menu, Listbox, Toggle Button, Table, Axes, Panel,* dan *Button Group* serta *dialog box* seperti *error, warning, help, message, waitbar, question, input, listbox,* dan *color dialog box*.

Semua dibahas satu-per-satu dan diberikan contoh program, penjelasan dan tampilan saat running. Sehingga diharapkan, para pembaca dapat memahami secara berlahan dan utuh dan dapat dilakukan secara mandiri.

Selanjutnya, buku ini membahas dengan interface komunikasi menerima dan mengirim data dengan Arduino. Contoh berikut penjelasan, program dan tampilan dibahas secara satu-per-satu step-by-step sehingga dapat diikuti secara mudah dan mandiri.

POLMANBABEL PRESS

Penerbit Politeknik Manufaktur Negeri Bangka Belitung (POLMANBABEL PRESS)
Kawasan Industri Air Kantung, Sungailiat, Bangka 33211
Telp: 0717 93586
E-Mail: polman@polman-babel.ac.id

ISBN 978-623-97870-6-6