University of Mosul / College of Engineering / Department of Computer Engineering

Level 4 Laboratories

Description of Real-Time Systems Lab

1. General Information:	
Laboratory Name and Laboratory Number:	Real-Time Systems Lab (211)
Linked Course Name:	Real-Time Systems
Department:	Computer Engineering
Number of weekly laboratory hours:	3 hour
Number of Semester Weeks:	15 weeks
Study Level:	Fourth Year
Laboratory Supervisor:	Dr. Basman Mahmoud Al-Hafez

2. General description of the laboratory:

This course introduces students to the **Real-Time Systems** field, which serves as a foundation to enter embedded systems programming. It starts from the basics of real-time computing and gives a comprehensive overview of the **Real-Time Systems** field, detailing all aspects of the **RTS** language: from sensors, actuators, signal conditioning, embedded programming, microcontrollers, as well as hardware platforms, communication protocols, and data processing. The course provides practical training to help the student develop embedded systems, test programming skills, and prepare him for realistic applications.

3. Laboratory Objectives:

No.	Objective
1.	Understand the basics of Real-Time Systems programming
2.	Develop practical knowledge of the software infrastructure: sensors, actuators, signal conditioning, data buses.
3.	Learn about embedded programming concepts
4.	Use microcontrollers and development boards (Arduino, Raspberry Pi)
5.	Enhance Python programming skills
6.	Define and use sensors to organize data acquisition and improve system reliability
7.	Working with embedded platforms
8.	Promote analytical thinking and problem-solving
9.	Debugging and software testing practice
10.	Develop team skills and cooperative programming skills
11.	Understand data processing and organization to group relevant data efficiently

4. Learning Outcomes:

At the end of the semester the following objectives are achieved:

• Students understand "Real-Time Systems programming" with an emphasis on the basic concepts and skills needed to develop embedded software using hardware platforms and programming languages.

• Students gain a solid understanding of embedded systems programming principles and will be able to apply them effectively in practical programming scenarios. • Documenting experiences and preparing professional reports. • Work within a team to solve programming problems.

5. Weekly trial schedule:			
week	**Experiment title**	**Tools / Software Used**	**Main objective**
1	Introduction	Datashow	Learn about general concepts of real-time systems
2	Sensor Characteristics Part 1	Sensor kit & KL31001 board	Study characteristics and operation of various sensors
3	Sensor Characteristics Part 2	Sensor kit & KL31001 board	Study characteristics and operation of various sensors
4	Arduino Sensors - Modules 1, 2, 3	Arduino board and sensors	Introduction to Arduino and its sensors
5	Example of Real-World Applications	Arduino board & ESP32 and sensors	Learn how to design real-time systems
6	Review and Quiz 1	---	General review and assessment
7	Introduction to Python Language	Datashow	Introduction to Python language fundamentals
8	Introduction to Raspberry Pi Hardware	Raspberry Pi	Introduction to Raspberry Pi and programming methods

9	Raspberry Pi - Sense HAT Part 1	Raspberry Pi & Sense HAT	How to use Raspberry Pi with sensors
10	Raspberry Pi - Sense HAT Part 2	Raspberry Pi & Sense HAT	How to use Raspberry Pi with sensors
11	Review and Quiz 2	All components needed	General review and assessment
12	Project Implementation 1	All components needed	Practical assessment for implementing group projects
13	Project Implementation 2	All components needed	Practical assessment for implementing group projects
14	Project Discussions - Part 1	Datashow	Discussion and evaluation of practical projects
15	Project Discussions - Part 2	Datashow	Discussion and evaluation of practical projects

6. Tools and equipment used:
• Raspberry Pi 4 • K&H Sensors Kit • KL-31001 • Raspberry Pi 3 • Sense HAT • Arduino Mega • Avometer • Oscilloscope • Desktop Computer

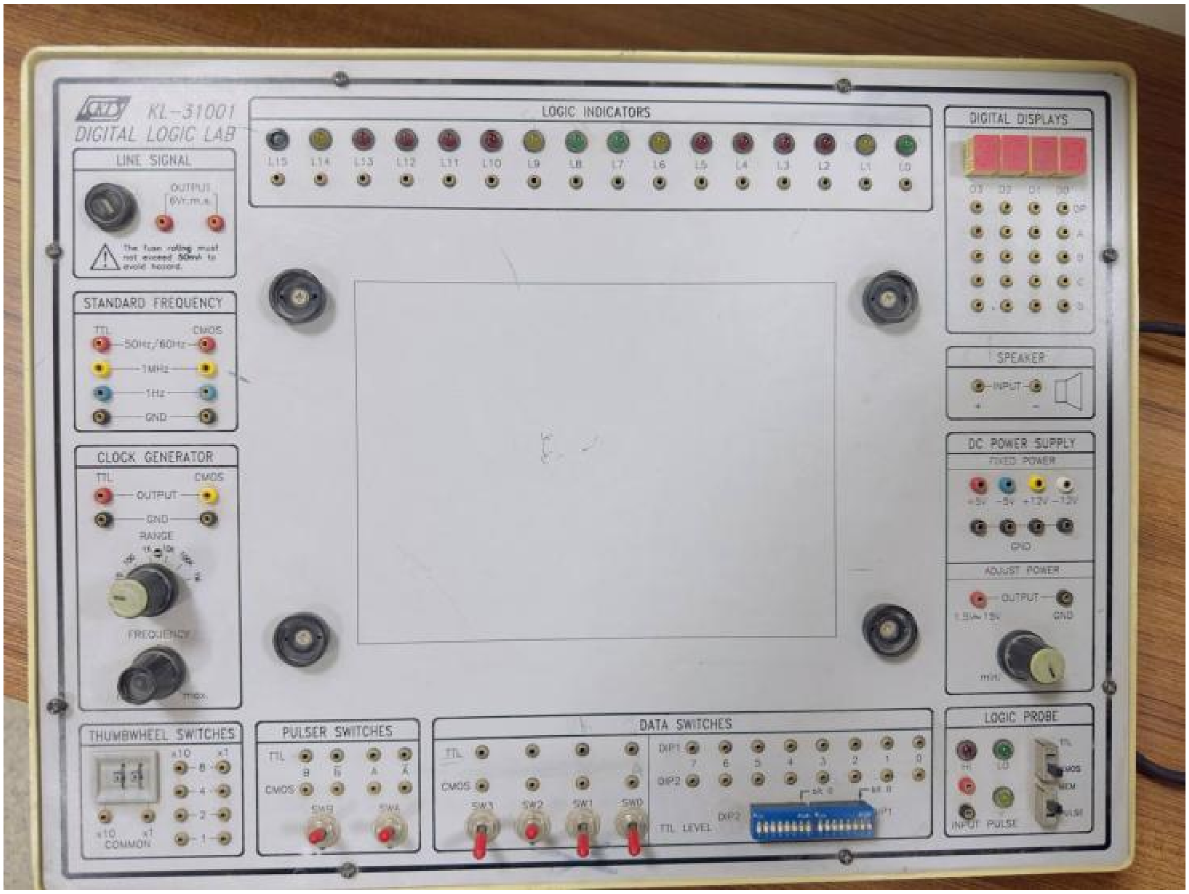
7. Safety Guide:
• Ensure electrical power is disconnected when connecting devices. • Do not touch electrical ports or network components without supervisor permission. • Maintain quiet environment and organize cables to avoid accidents. • Use simulation software before testing on actual devices.

8. Assessment Mechanism:

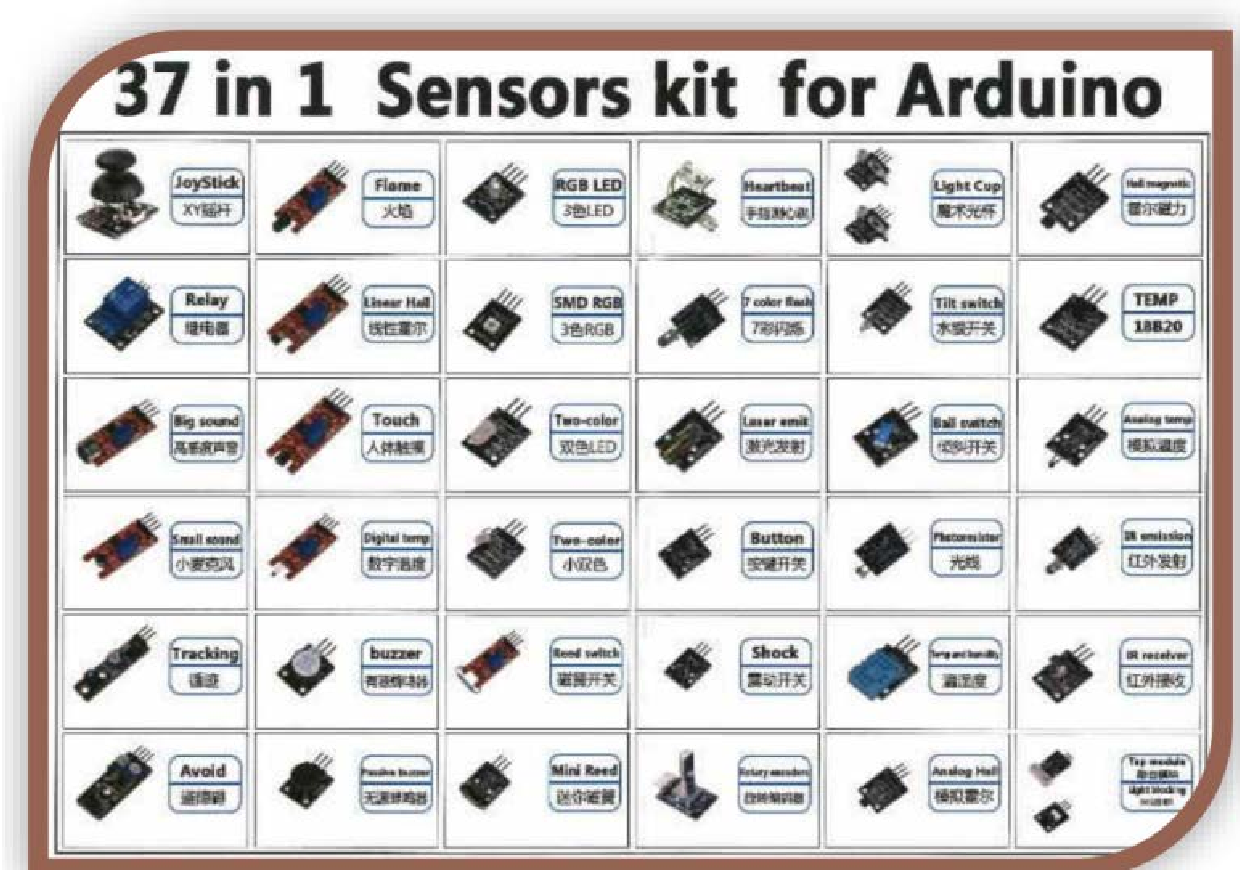
Assessment Component	Percentage
Attendance and Participation	12%
Quizzes	20%
Final Practical Exam	40%
Practical Project	28%

9. References and Sources:
• K&H Documents • Arduino IDE and Tinkercad software usage • electronicshub.org - Getting Started with ESP32 Introduction to ESP32 • Educational websites: anaconda.com, Arduino.com, https://www.raspberrypi.com/

10. Attachments:	
1.	Action Plan for Classroom Trials and Evaluation Method
2.	Hazard_level_of_real-time_computer_engineering_department

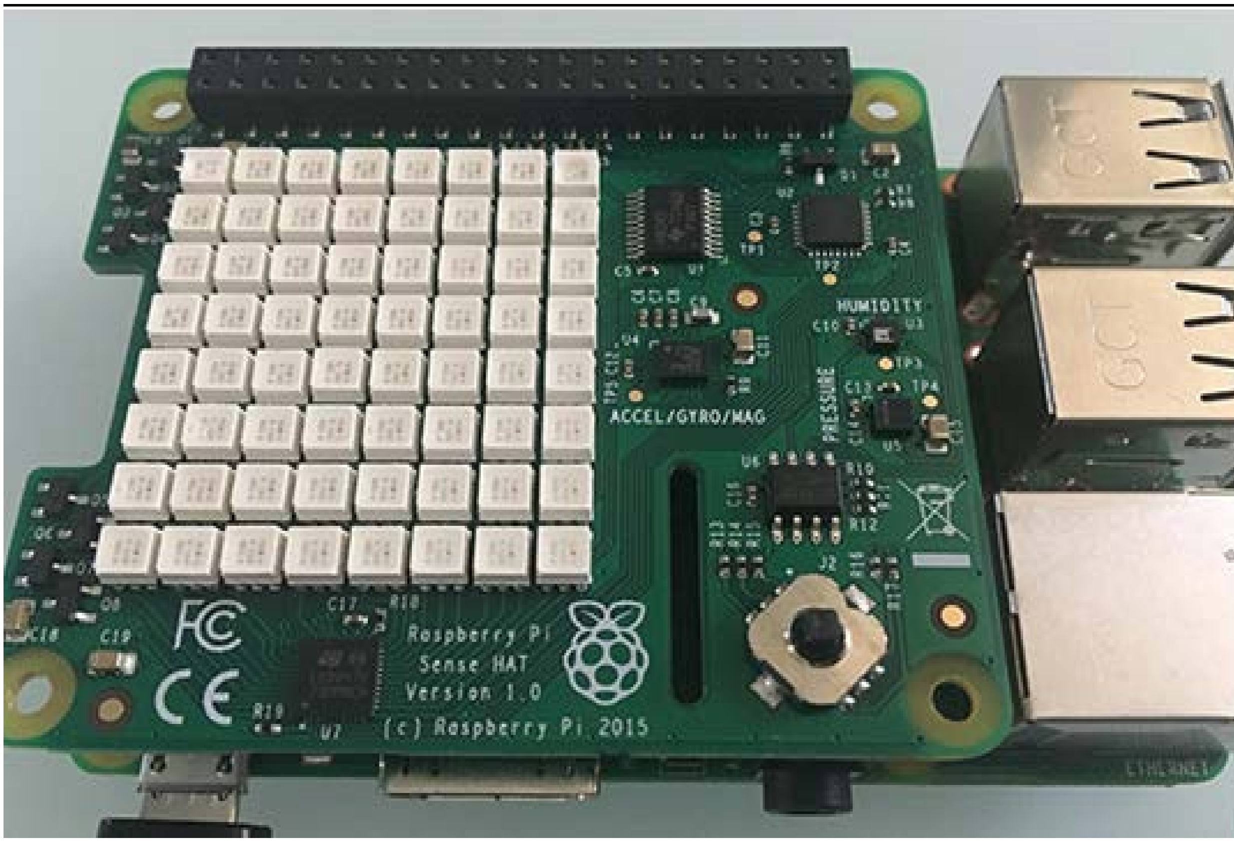

KL-31001
DIGITAL LOGIC LAB
LOGIC INDICATORS
DIGITAL DISPLAYS
LINE SIGNAL
STANDARD FREQUENCY
SPEAKER
DC POWER SUPPLY
CLOCK GENERATOR
THUMBWHEEL SWITCHES
PULSER SWITCHES
DATA SWITCHES
LOGIC PROBE

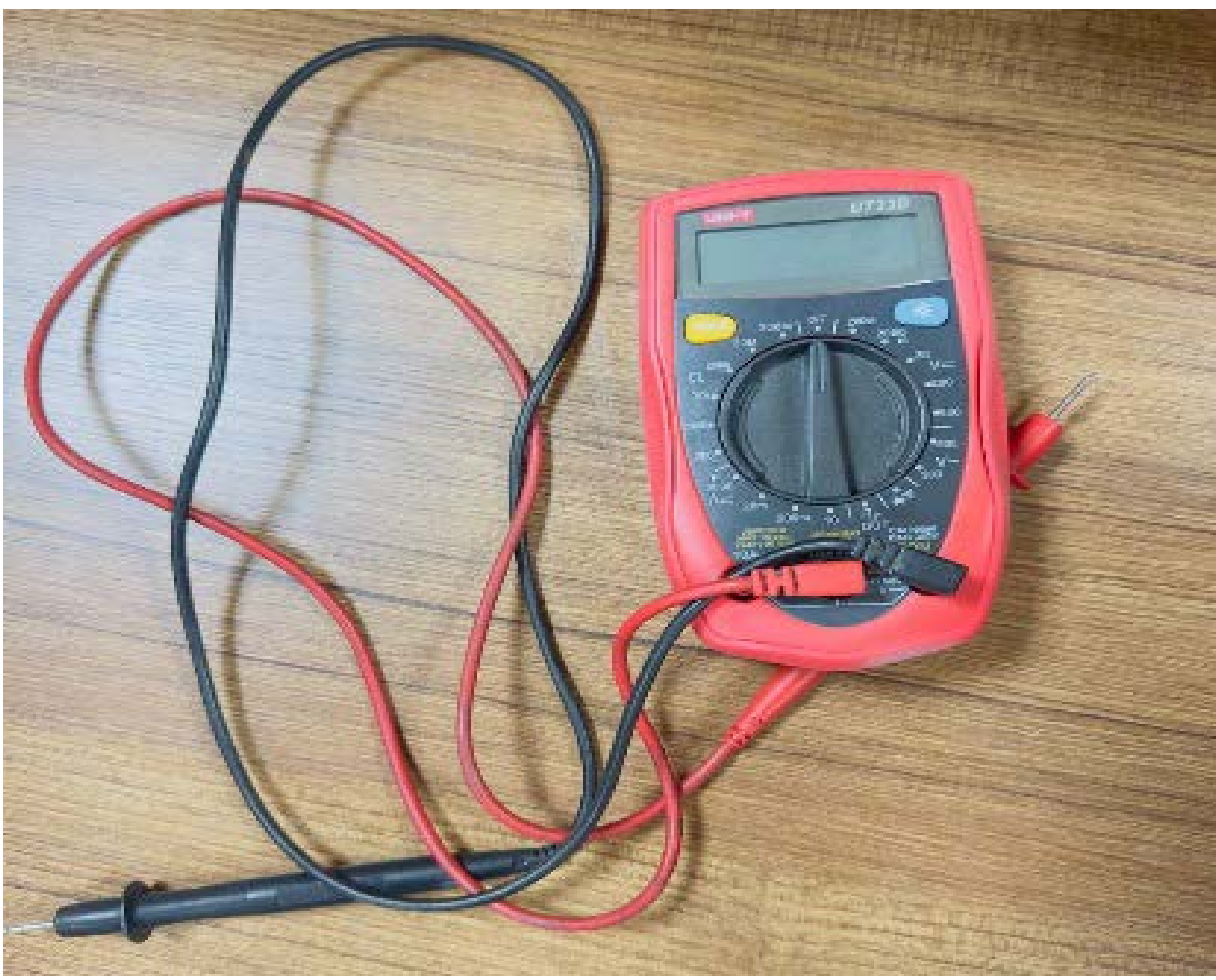

KL-64002
SENSOR MODULE 2


37 in 1 Sensors kit for Arduino
JoyStick
XY摇杆
Flame
火焰
RGB LED
3色LED
Heartbeat
Light Cup
Relay
继电器
Linear Hall
SMD RGB
3色RGB
7 color flash
Tilt switch
TEMP
18B20
Big sound
Touch
Two-color
双色LED
Laser emit
激光发射
Ball switch
Analog temp
Small sound
Digital temp
数字温度
Two-color
Button
按键开关
Photoresistor
IR emission
红外发射
Tracking
buzzer
Reed switch
磁簧开关
Shock
IR receiver
红外接收
Avoid
Passive buzzer
Mini Reed
Analog Hall
Tap module


KL-64003
SENSOR MODULE 3


Raspberry Pi 3


HUMIDITY
ACCEL/GYRO/MAG
PRESSURE
Raspberry Pi
Sense HAT
Version 1.0
(c) Raspberry Pi 2015



Description of the Control systems Laboratory (Lab. 210)

1. General Information:	
Laboratory Name:	Control Lab
Associated Course Name:	Control systems
Department:	Computer engineering
Weekly Lab Hours:	4 Hours
Number of Weeks in the Semester:	15
Academic Level:	Fourth level
Lab Supervisor:	Dr.Sura Nawfal

2. General Description of the Laboratory:

The laboratory provides students with hands-on experience in control system analysis, design, and implementation. It covers a wide range of topics from basic MATLAB programming to advanced control techniques, including analogue and digital control systems. The lab integrates theoretical knowledge with practical applications, using tools like MATLAB, Simulink, PLCs, and analogue computers

3. Laboratory Objectives:

- To familiarize students with essential tools like MATLAB and Simulink for control system analysis.
- To provide practical experience in designing and implementing Control Systems using both analogue and digital methods.
- To introduce students to real-time control systems and programmable logic controllers (PLCs).
- To develop skills in analyzing system responses, tuning controllers, and handling control challenges.
- To encourage teamwork and problem-solving through a control system design group

4. Learning Outcomes:
By the end of the course, students will demonstrate the ability to **design and implement control systems**, including PID controllers and PLC programming. They will apply theoretical concepts to **tune system responses**, optimize performance, and troubleshoot real-world control challenges. students will develop skills in **integrating hardware and software** to program real-time control systems with smart sensors. These competencies prepare them for advanced engineering tasks and industrial automation applications. (**outcome 2**).

5. Weekly Experiment Schedule:

Week	Experiment Title	Tools / Software Used	Main Objective
1	Matlab Basics for control systems	Matlab	Familiarization with MATLAB, an essential tool for control system analysis and design.
2	Graphical User Interface (GUI) using MATLAB	Matlab	Building on MATLAB basics, students learn to create user-friendly interfaces for control applications.
3	Control Basics and Block Reduction Using MATLAB.	Matlab	Understanding fundamental control concepts and applying MATLAB for block reduction techniques.
4	Transfer Function and Converting Between System Representations	Matlab	Exploring transfer functions and converting between different system representations.
5	Step response and steady state error.	Matlab	Applying Laplace transforms in MATLAB to analyze step response and study steady-state error.

6	Analogue Computers	Multisim+Matlab	Hands-on exploration of analogue computers and their relevance in simulating control system dynamics.
7	PID controller using Analogue Computers	Multisim+Matlab	Design a PID components using RLC and op-amplifiers
8	PLC Programming	LOGO Siemens Simulator	Introduction to programmable logic controllers and hands-on experience in programming.
9	PLC Programming (special ladder diagrams)	LOGO Siemens Simulator	Introduction to program and connect special ladder diagrams
10	PLC applications	LOGO Siemens Simulator	Real-world applications of PLCs in control scenarios, emphasizing industrial use cases.
11	Bode plot	Matlab	Understanding frequency response through Bode plot analysis.
12	LabVIEW LINX connection (Raspberry PI)	LabVIEW LINX+ **Raspberry PI**	Interfacing LabVIEW with external devices Raspberry Pi using LabVIEW LINX.
13	Pole Placement Design	MATLAB, LabVIEW	Apply pole placement technique to achieve desired system performance.
14	Comprehensive Review	MATLAB, LabVIEW, all Hardware Interface	Review all experiments, evaluate practical understanding, and overall performance.

15	**Semester exam**	**Semester exam**	**Semester exam**

6. Tools and Equipment Used:

Software Tools:

1. MATLAB
2. Multisim (or equivalent analog circuit simulator)
3. LabVIEW
4. LOGO Siemens Simulator (PLC Software)

Hardware Equipment:

1. Raspberry Pi
2. Analog Computers / Trainer Kits
3. PLC Hardware (if available) **Siemens LOGO PLC hardware**
4. Power Supplies
5. Measurement Instruments **Oscilloscope and Digital Multimeters**
6. General Electronics Lab Equipment: **Breadboards,Wires, connectors, and jumpers**

7. Safety Guidelines:

- Always follow the instructor's directions before operating any hardware or equipment.
- Ensure proper connection of all hardware interfaces before powering up.
- Disconnect power sources before modifying any circuit.
- Handle all computers, data acquisition devices, and controllers carefully to prevent physical or electrical damage.
- Report any abnormal behaviour of the system (unexpected output, overheating, error messages) to the instructor immediately.
- Maintain a clean and organized workspace to prevent accidental cable disconnection or equipment falls.
- Follow all general laboratory safety protocols.

8. Evaluation Method:

Percentage	Evaluation Item
13%	Attendance and Participation
13%	Practical Quiz
27%	Laboratory Performance
13%	Laboratory Visiting Reports
34%	Semester Practical Exam

9. References and Sources:

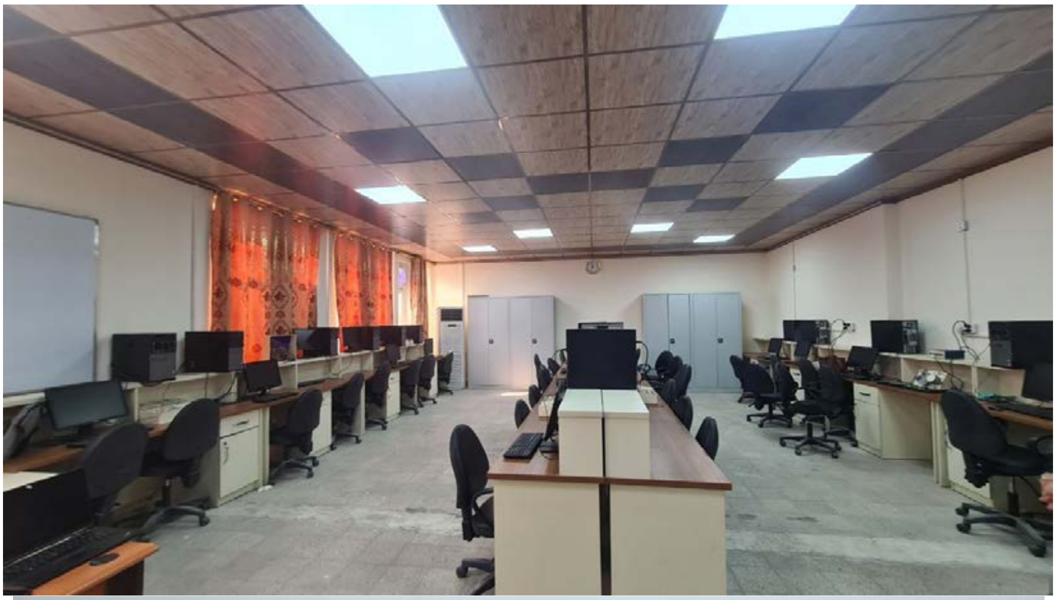
- Nise, N. S. *Control Systems Engineering*, 7th Edition, Wiley.
- Ogata, K. *Modern Control Engineering*, 5th Edition, Prentice Hall.
- MATLAB Documentation: https://www.mathworks.com/help/matlab/
- LabVIEW User Manual, National Instruments.

10.Attachments:

- Laboratory Schedule.
- Experiment Procedure Sheet.
- Sample LabVIEW VI Files.
- Sample of Ladder diagram.
- Sample MATLAB Scripts.

<table>
<tr><th colspan="2">Laboratory tools instruments</th></tr>
<tr><td></td><td>Analogue Servo unit 110-33 , Mechanical unit control and instrumentation 100-33.
</td></tr>
<tr><td></td><td>Data Acquisition Cards.

</td></tr>
<tr><td></td><td>Raspberry Pi 4 and 3 versions. Arduino and sensors set.
</td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td>LOGO 12\24 REC PLC.
</td></tr>
</table>




210
113LGE040011
مختبر السيطرة
113LGE04004
مختبر المعالجات
113LGE03003
مختبر المنطق

Description of the Digital Control Laboratory(Lab. 210)

1. General Information:	
Laboratory Name:	Digital Control Lab
Associated Course Name:	Digital Control
Department:	Computer engineering
Weekly Lab Hours:	4 Hours
Number of Weeks in the Semester:	15
Academic Level:	Fourth level
Lab Supervisor:	Dr.Sura Nawfal

2. General Description of the Laboratory:

This laboratory provides hands-on experience in the analysis, design, simulation, and implementation of control systems. The experiments integrate software tools such as MATLAB and LabVIEW, covering fundamental and advanced control concepts, including classical control, state-space modeling, digital control, and real-time implementation. The lab aims to bridge theoretical knowledge with practical applications using simulation and hardware interfacing.

3. Laboratory Objectives:

- This lab aims to bridge the gap between theoretical knowledge and practical applications through a series of hands-on experiments. The main objectives are:
- To provide practical experience in modeling, simulating, and designing control systems.
- To familiarize students with industry-standard tools: MATLAB and LabVIEW.
- To develop skills in classical control methods (root locus, Bode, Nyquist).
- To introduce advanced topics such as state space, observers, and digital control.
- To implement real-time control systems using hardware interfaces and LabVIEW Real-Time modules.

4. Learning Outcomes:

Solve digital control system problems using Z-transform, simulation diagram of digital control systems, then, design and implement various digital controllers using MATLAB to control a DC and stepper motors. Integrate and program real-time control systems with smart sensors (**outcome 2**).

5. Weekly Experiment Schedule:

Week	Experiment Title	Tools / Software Used	Main Objective
1	**Laplace Transform Using MATLAB**	MATLAB	Laplace Transform Using MATLAB
2	**LabVIEW Basics**	LabVIEW	Introduce LabVIEW environment and basic functionalities.
3	**LabVIEW Control Design Model**	LabVIEW Control Design Toolkit	Apply control design principles using LabVIEW with simulation and modeling.
4	**Design of Root Locus**	Matlab	Understand and apply root locus method for control system design.
5	**Root Locus compensators**	Matlab/SISO tool	Design compensators using root locus to achieve desired control performance.
6	**Time response analysis with LabVIEW**	LabVIEW	Analyzing and visualizing time response characteristics using LabVIEW.
7	**PID Controllers Design**	LabVIEW, MATLAB	Designing Proportional-Integral-Derivative (PID) controllers using the Ziegler–Nichols tuning method.

8	**Design with LabVIEW environment**	LabVIEW	Advanced LabVIEW applications in control system design.
9	**LabVIEW LINX connection**	LabVIEW LINX+Arduino	Interfacing LabVIEW with external devices like Arduino and Raspberry Pi using LabVIEW LINX.
10	**Motor Control using Arduino and LabVIEW**	LabVIEW LINX+Arduino	Hands-on experience in controlling motors using Arduino and LabVIEW.
11	**Digital Control Design**	MATLAB	Introduction to digital control systems and design principles.
12	**Stepper motor controller using LabVIEW via DAQ**	LabVIEW+DAQ+stepper	Utilizing LabVIEW for Data Acquisition (DAQ) with stepper motors.
13	**DC motor control p-controller**	LabVIEW+DAQ+servo motor	Design a proportional controller.
14	**DC motor control PID-controller**	LabVIEW+DAQ+servo motor	Implementing PID control for DC motor systems.
15	**Practical semester exam**	Practical semester exam	Practical semester exam

6. Tools and Equipment Used:

Software:

- MATLAB (Control System Toolbox, Simulink)
- LabVIEW (Control Design and Simulation Module, Real-Time Module)
- LabVIEW DAQmx Drivers **Data Acquisition Devices (DAQ) /** NI USB-6009
- National Instruments Measurement & Automation Explorer (NI MAX)

Hardware:

- National Instruments Data Acquisition Devices (NI USB-6009)
- Power Supply Units
- DC Motor and Servomechanism Models
- Sensors (Encoders, Tachometers, Potentiometers,…)
- Signal Generators (Function Generators)
- Oscilloscope

General Laboratory Equipment:

- Breadboards and Prototyping Boards
- Cables and Connectors
- Measurement Probes

7. Safety Guidelines:

- Always follow the instructor's directions before operating any hardware or equipment.
- Ensure proper connection of all hardware interfaces before powering up.
- Verify the proper configuration of LabVIEW Real-Time and DAQ hardware to avoid system damage.
- Disconnect power sources before modifying any circuit.
- Handle all computers, data acquisition devices, and controllers carefully to prevent

physical or electrical damage.

- Report any abnormal behavior of the system (unexpected output, overheating, error messages) to the instructor immediately.
- Maintain a clean and organized workspace to prevent accidental cable disconnection or equipment falls.
- Follow all general laboratory safety protocols.

8. Evaluation Method:

Percentage	Evaluation Item
13%	Attendance and Participation
13%	Practical Quiz
27%	Laboratory Performance
13%	Scientific Visiting Reports
34%	Semester Practical Exam
	bonus
100%	Final

9. References and Sources:

- Nise, N. S. *Control Systems Engineering*, 7th Edition, Wiley.
- Ogata, K. *Modern Control Engineering*, 5th Edition, Prentice Hall.
- MATLAB Documentation: https://www.mathworks.com/help/matlab/
- LabVIEW Control Design Toolkit Documentation: https://www.ni.com
- LabVIEW User Manual, National Instruments.

10. Attachments:

- Laboratory Schedule.
- Experiment Procedure Sheet.
- Sample LabVIEW VI Files.
- Sample MATLAB Scripts.

Laboratory tools instruments

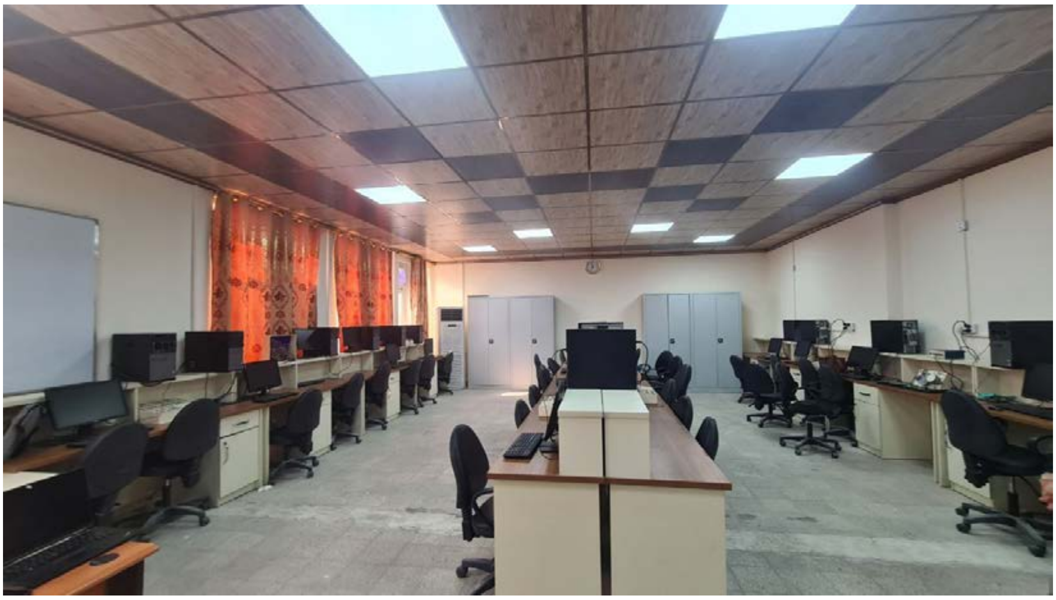
Analogue Servo unit 110-33 , Mechanical unit control and instrumentation 100-33.

Data Acquisition Cards.

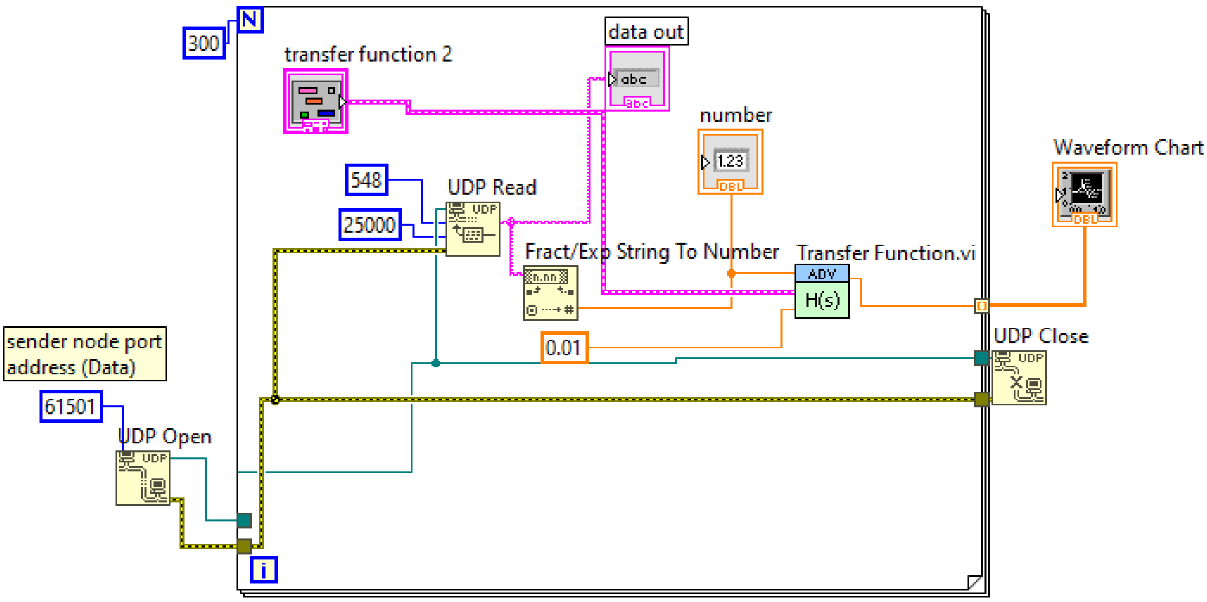
Raspberry Pi 4 and 3 versions. Arduino and sensors set.

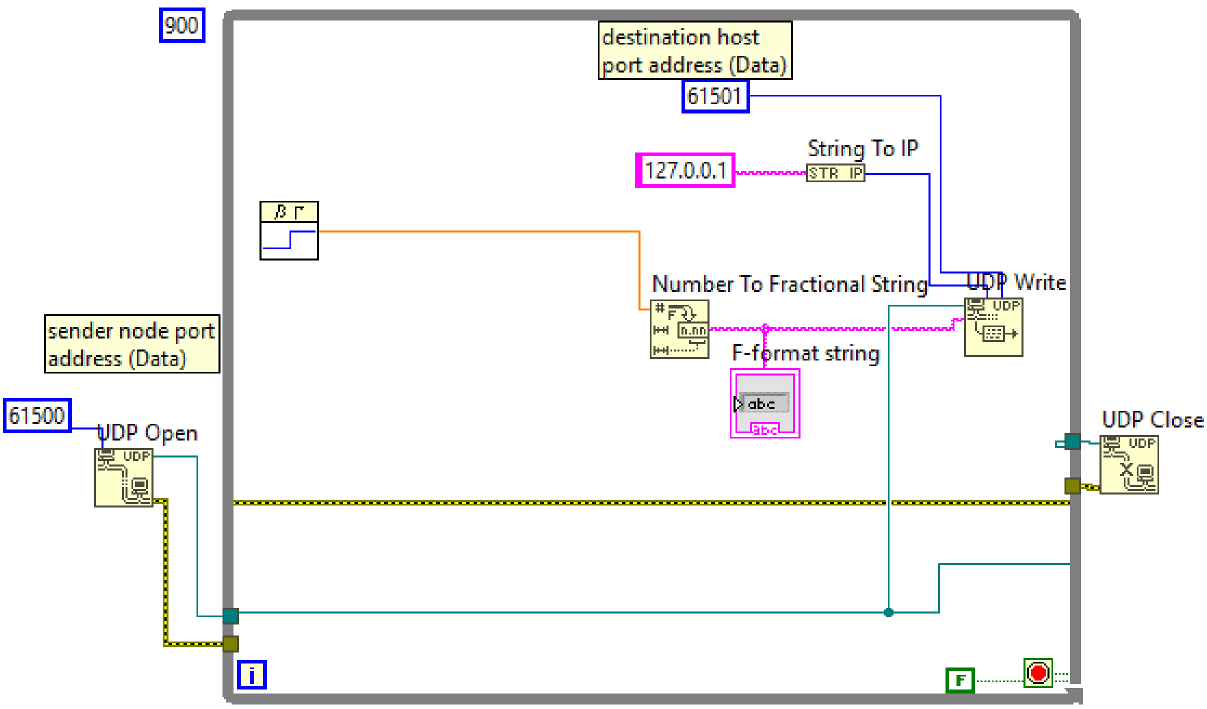
LOGO 12\24 REC PLC.

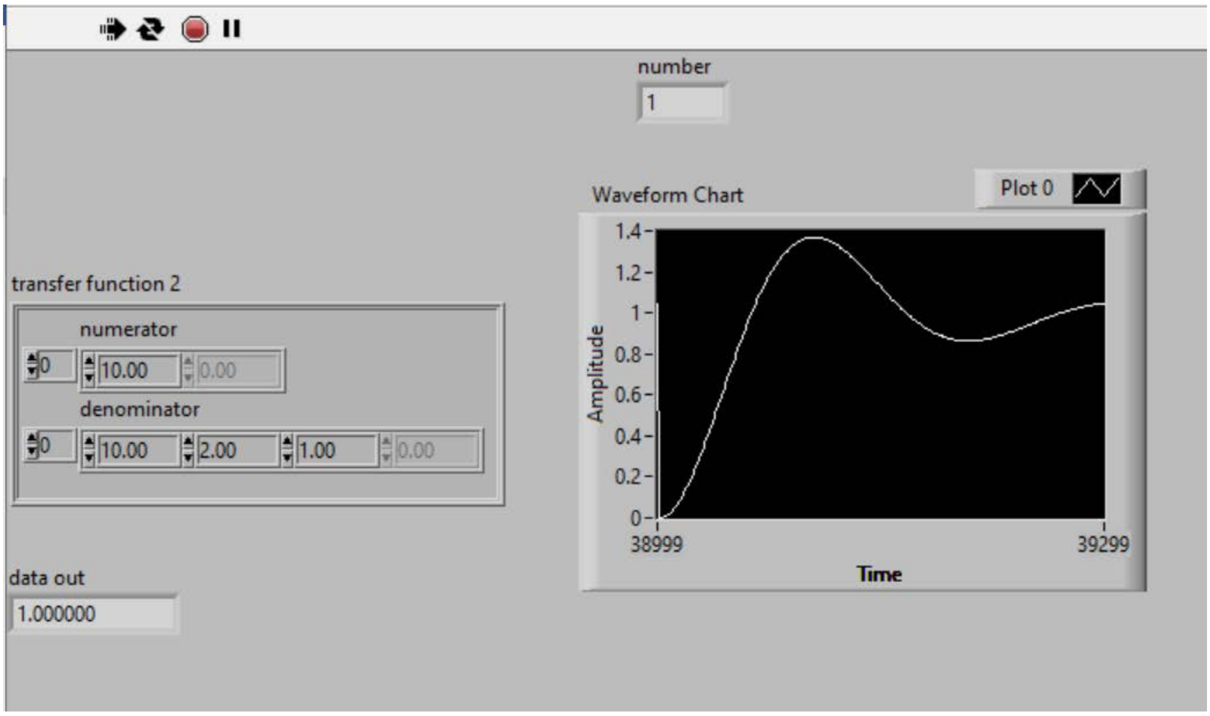

210
113LGE040011
مختبر السيطرة
113LGE04004
مختبر المعالجات
113LGE03003
مختبر المنطق



LabVIEW (.vi sample programs)






number
1
Waveform Chart
Plot 0
transfer function 2
numerator
0
10.00
0.00
denominator
0
10.00
2.00
1.00
0.00
Amplitude
1.4
1.2
1
0.8
0.6
0.4
0.2
0
38999
39299
Time
data out
1.000000

M file sample (mathscript)

```
K1=2;K2=3;
s=tf('s');
G1=1/(s+1)/(s+2);
H1=s;
G2=1/(s+3)
H2=s^2+1
G3=(s^3+s)/((s+0.1)*(s+0.2))
H3=s+1;
F1=feedback (G1,H1)
F2=parallel (H2,G2)
F3=feedback (H3,G3)
F4=feedback(F2*F3,K1,+1)
F5=feedback(F4,K2/H3*F1)
F6=series (F5,1+F1)
```

University of Mosul
College of Engineering
Computer Engineering Dept.

Fourth Class
Control Lab.
Experiment:2

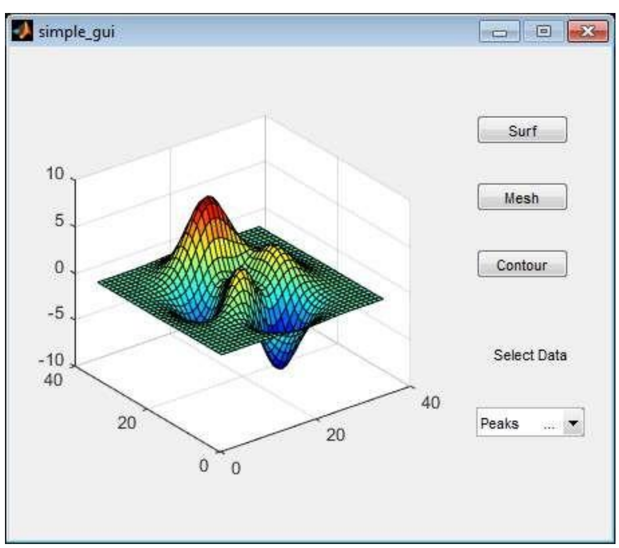
Graphical User Interface (GUI) in Matlab

What Is a UI?

A user interface (UI) is a graphical display in one or more windows containing controls, called components, that enable a user to perform interactive tasks. The user does not have to create a script or type commands at the command line to accomplish the tasks.

Unlike coding programs to accomplish tasks, the user does not need to understand the details of how the tasks are performed. UI components can include menus, toolbars, push buttons, radio buttons, list boxes, and sliders—just to name a few. UIs created using MATLAB® tools can also perform any type of computation, read and write data files, communicate with other UIs, and display data as tables or as plots.

The following figure illustrates a simple UI that you can easily build yourself.



The UI contains these components:

• An axes component

• A pop-up menu listing three data sets that correspond to MATLAB functions: peaks, membrane, and sinc

• A static text component to label the pop-up menu

• Three buttons that provide different kinds of plots: surface, mesh, and contour.

When you click a push button, the axes component displays the selected data set using the specified type of 3-D plot.

Ways to Build MATLAB UIs:

A MATLAB UI is a figure window to which you add user-operated components. You can select, size, and position these components as you like. Using callbacks you can make the components do what you want when the user clicks or manipulates the components with keystrokes.

You can build MATLAB UIs in two ways:

• **Create the UI using GUIDE**

This approach starts with a figure that you populate with components from within a graphic layout editor. GUIDE creates an associated code file containing callbacks for the UI and its components. GUIDE saves both the figure (as a FIG-file) and the code file. You can launch your application from either file.

• **Create the UI programmatically**

Using this approach, you create a code file that defines all component properties and behaviors. When a user executes the file, it creates a figure, populates it with components, and handles user interactions. Typically, the figure is not saved between sessions because the code in the file creates a new one each time it runs.

The code files of the two approaches look different. Programmatic UI files are generally longer, because they explicitly define every property of the figure and its controls, as well as the callbacks. GUIDE UIs define most of the properties within the figure itself. They store the definitions in its FIG-file rather than in its code file. The code file contains callbacks and other functions that initialize the UI when it opens.

You can create a UI with GUIDE and then modify it programmatically. However, you cannot create a UI programmatically and then modify it with GUIDE.

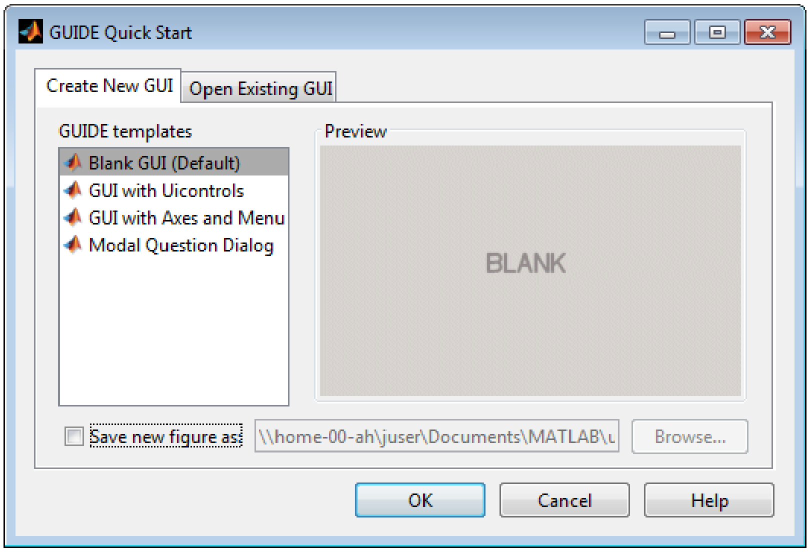
How to Create a UI with GUIDE?

We are going to develop a simple **Matlab GUI**. We'll use the **Matlab GUIDE** (**Graphical User Interface Development Environment**) which is pretty handy.

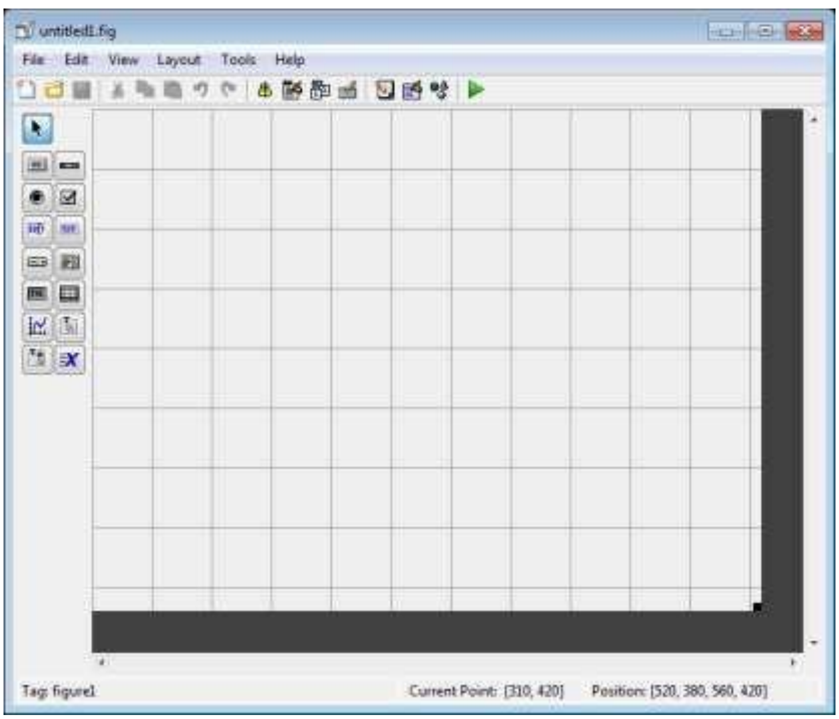
Let's start the ride! On the command window type:

```
>>guide
```

This will open the '**Quick Start**' window, where you can study or review their examples, too! Follow the steps and select the first option: '**Blank GUI (Default)**'.



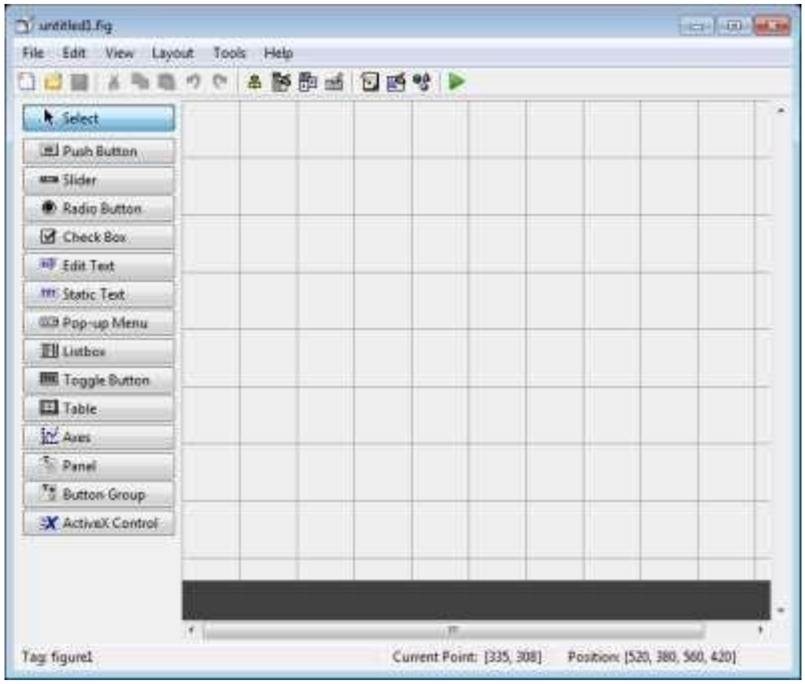
Then, an untitled figure will pop-up. You have some components on the left menu, which you can drag onto your interface.



Display the names of the UI components in the component palette:

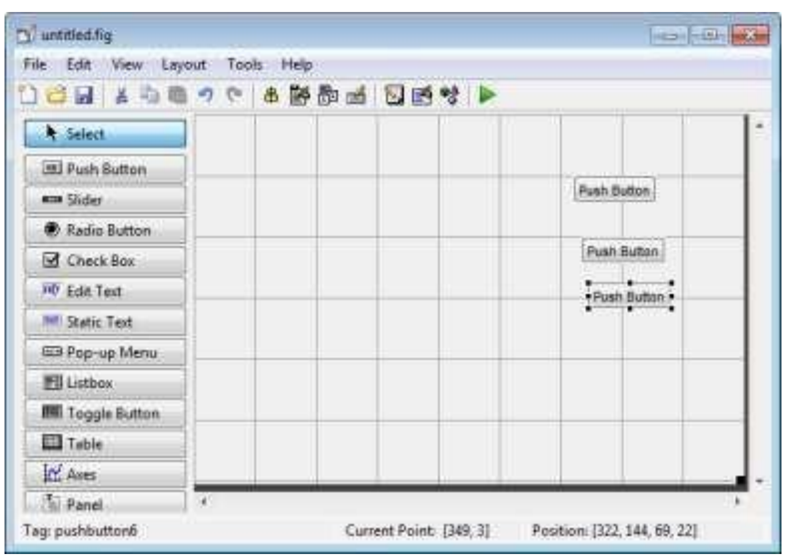
1. Select File > Preferences > GUIDE.
2. Select Show names in component palette.

3. Click OK.



Add, align, and label the components in the UI:

Add push buttons to the UI. Select the push button tool from the component palette at the left side of the Layout Editor and drag it into the layout area. Create three buttons, positioning them approximately as shown in the following figure.



Align the Components:

If several components have the same parent, you can use the Alignment Tool to align them to one another. To align the three push buttons:

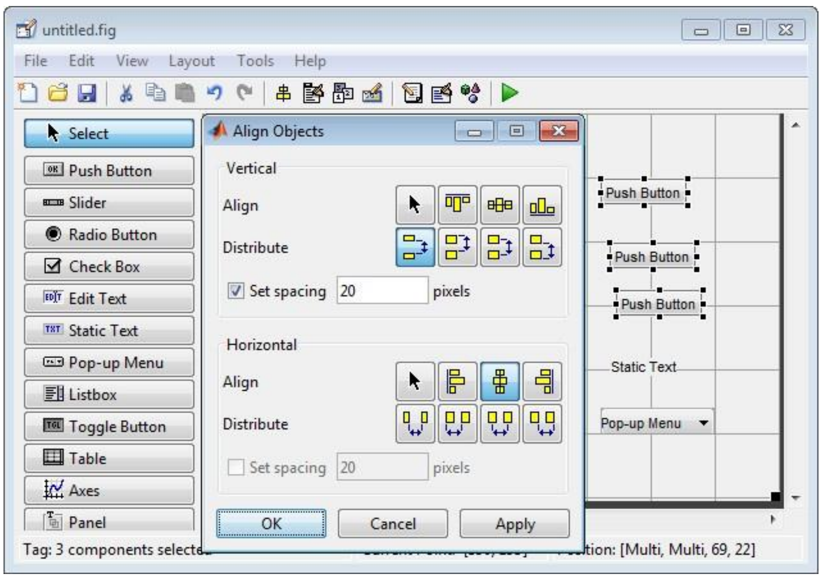
1 Select all three push buttons by pressing Ctrl and clicking them.

2 Select Tools > Align Objects.

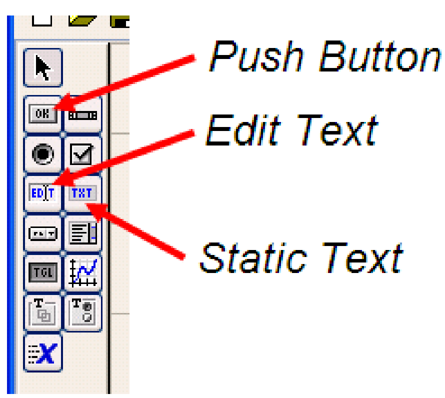
3 Make these settings in the Alignment Tool:

• Left-aligned in the horizontal direction.

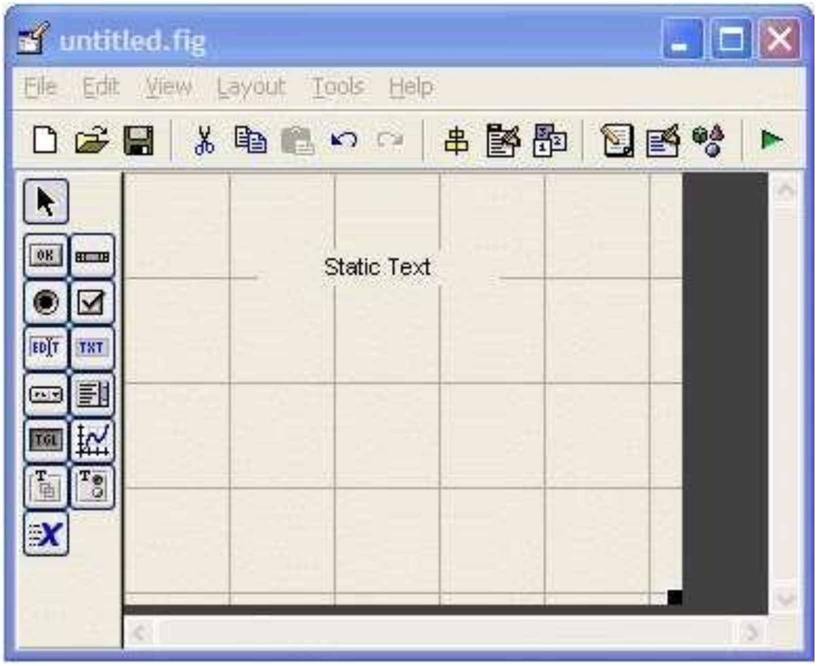
• 20 pixels spacing between push buttons in the vertical direction.



In this example we are going to use only two ‘**push buttons**’ and one ‘**static text**’.



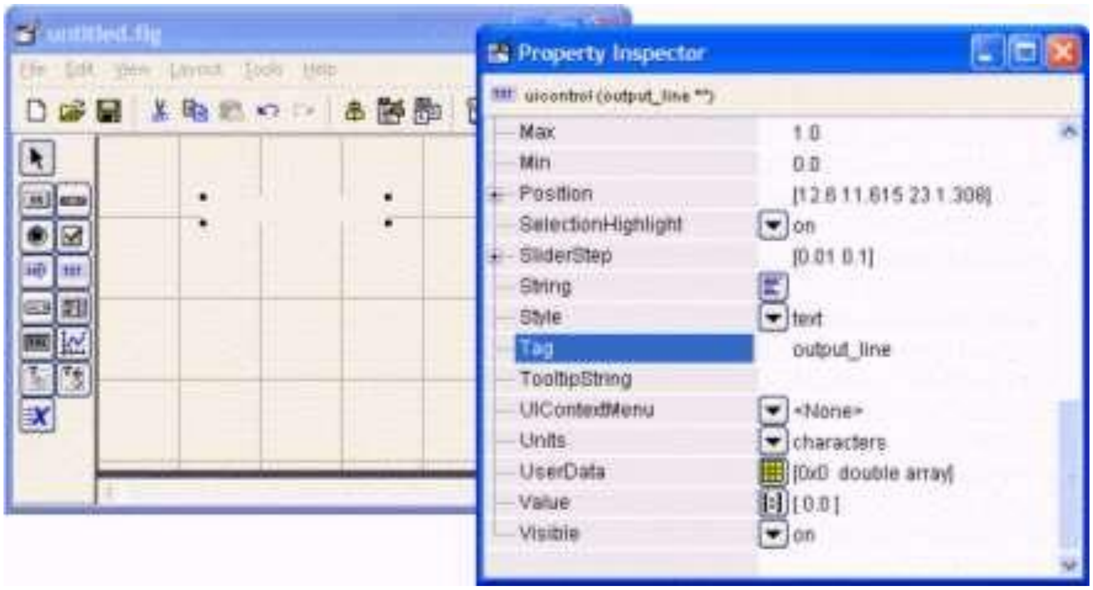
Drag and drop a ‘**static text**’ onto your Matlab GUI. You can reduce or increase the size of your interface window by dragging its bottom-right corner, as it’s done in other drawing programs.



Double click on this ‘**static text**’ and a ‘**Property Inspector**’ window will appear. Scroll down and look for the ‘**String**’ property and delete what's in there. For the moment we want it to be blank.

Then, make the ‘**Tag**’ property to be ‘**output_line**’. You can use whatever name you want.

Your windows must look similar to the figure below:

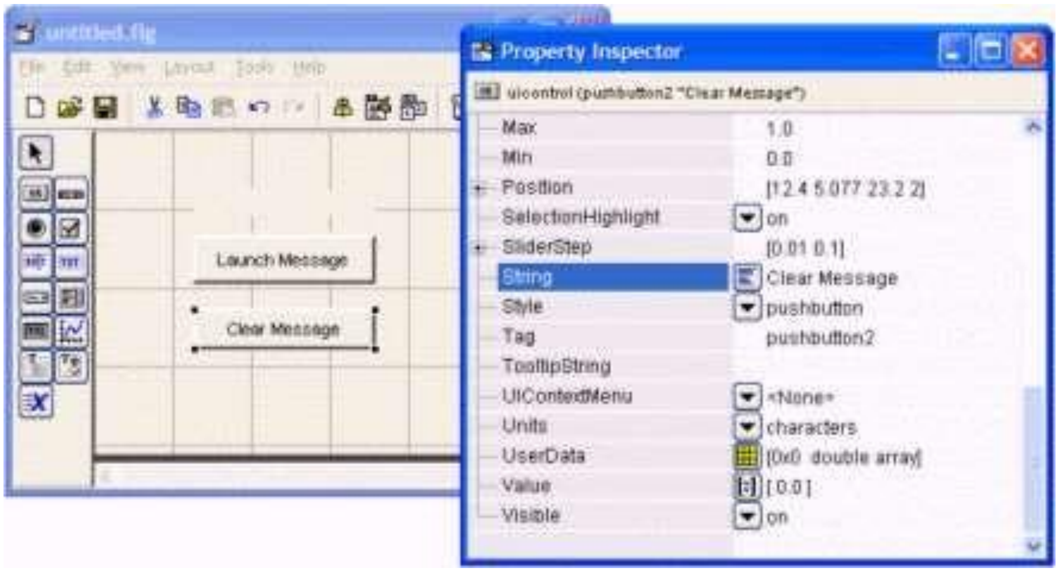


Then, drag-and-drop a '**push button**' onto your interface. Modify its '**String**' property to read '**Launch Message**'. Let its '**Tag**' property intact. You could change this tag... it's the name or identifier of the object as it's going to be recognized in the rest of the code.

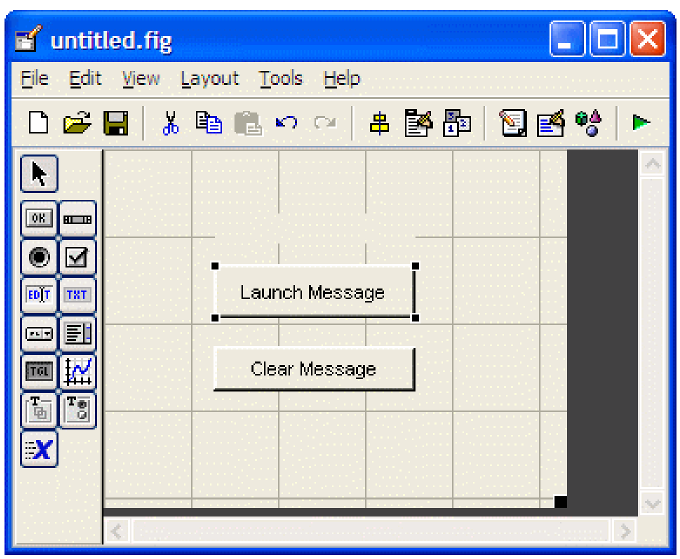
Your windows must look similar to the figure below:

Drag-and-drop another '**push button**'. Modify its '**String**' property to read

'**Clear Message**' and leave its '**Tag**' as it is. You'll produce these results.



Now, right-click on the '**Launch Message**' button and choose '**View Callbacks**' -> '**Callback**'



You'll be asked to save your figure. A good name is **hello_world.fig**... use the name that you like.

You'll be taken to the Matlab code (in the editor window) that will drive your interface. Matlab has **automatically** created functions related to your components. You have to make the final touches... For the moment, don't worry about the many lines automatically created. Just focus on what we need to do.

The '**Callback**' functions are the instructions that will be executed when the user pushes the buttons or does something with the components that you have included in your Matlab GUI. In this case, you'll see something like this code.

```
% --- 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 MAT
% handles    structure with handles and user data (see GUIDATA)
```

A '**set**' instruction sets the properties of the elements that you indicate. Do you remember that you have a '**static text**' with the tag (identifier) '**output_line**'? We are going to modify it when the user pushes the button with the string (or label) '**Launch Message**'. This is accomplished with the instruction:

```
set(handles.output_line,'String','Hello  World!!')
```

The **first parameter is the object** (component) that you're going to modify. It starts with '**handles.**'. The **second argument is the object's property** that you're going to modify, and in this case is the '**String**' property. The **third argument is the value** that you want to assign to the property.

So, the result is that when the user presses the '**Launch Message**' button, a message reading '**Hello World!!**' will appear in the '**output line**' (officialy named '**handles.output_line**'). Add this single line to the code, so that it looks like this:

```
% --- 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 MAT
% handles    structure with handles and user data (see GUIDATA)
set(handles.output_line,'String','Hello World!!')
```

We'll do something similar to the '**callback**' corresponding to the '**Clear Message**' button. So change this original code...

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MAT
% handles    structure with handles and user data (see GUIDATA)
```

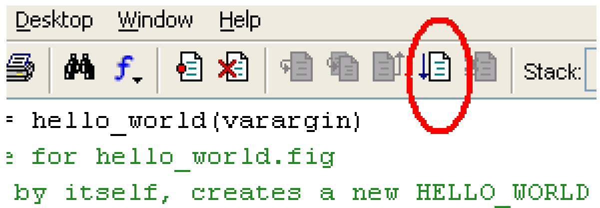
into this...

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MAT
% handles    structure with handles and user data (see GUIDATA)
set(handles.output_line,'String','')
```

The result is that when the user presses the '**Clear Message**' button, a blank message will appear in the '**output line**' (officialy named '**handles.output_line**').

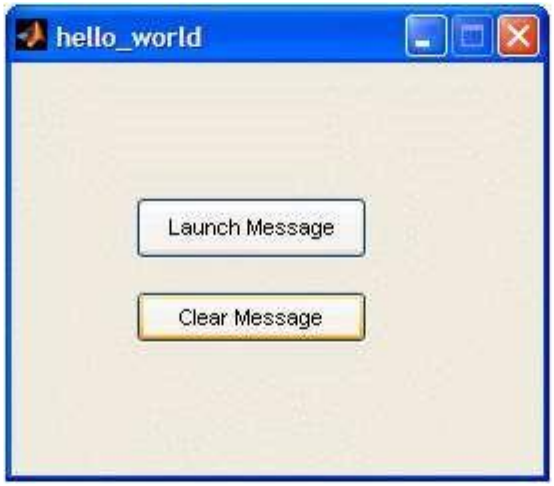
Magic is about to happen!

Now, run your interface by clicking the '**run**' icon at the top of the editor window...



Try, it! Press the '**Launch Message**' button... and an interesting message appears...

then, press the ‘**Clear Message**’ button...



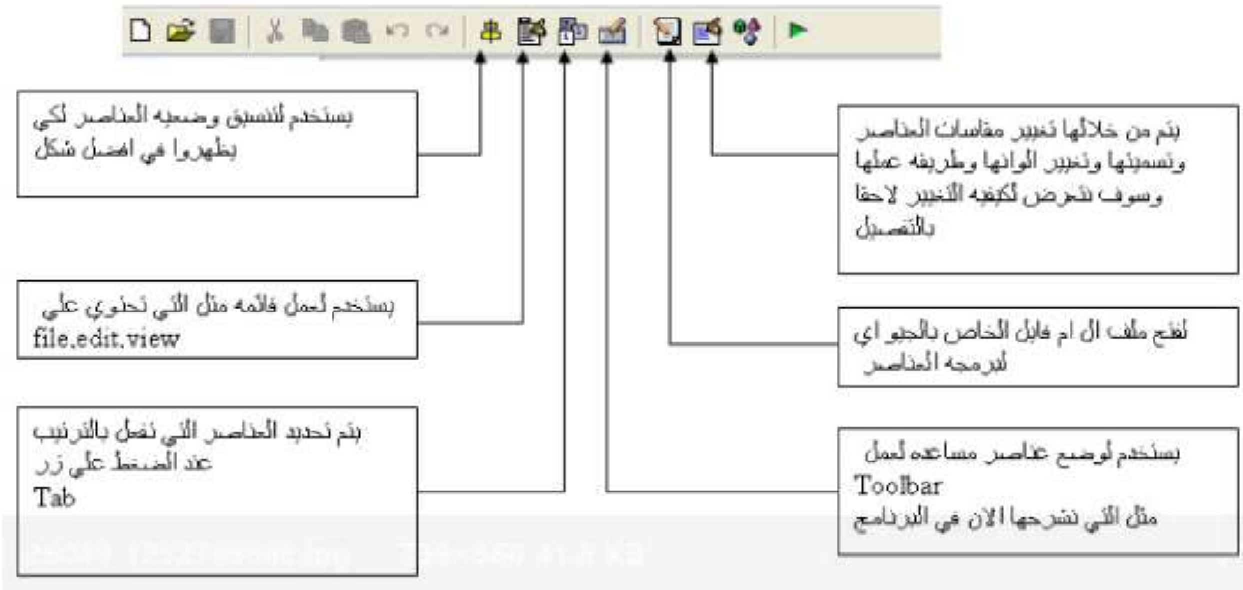
let’s summarize:

- You can drag-and-drop your components onto your graphic interface to start your Matlab GUI.

- Matlab will automatically create callback-functions, related to the buttons or other components that you include.

- The ‘set’ instruction assigns values to properties of the elements that you want to modify. The general format is:

set(handles.tag_of_your_component, 'Property', value)



التطبيق الاول:

ادخال رقمين ثم اخيار العمليه المراد تطبيقها ثم اظهار الناتج

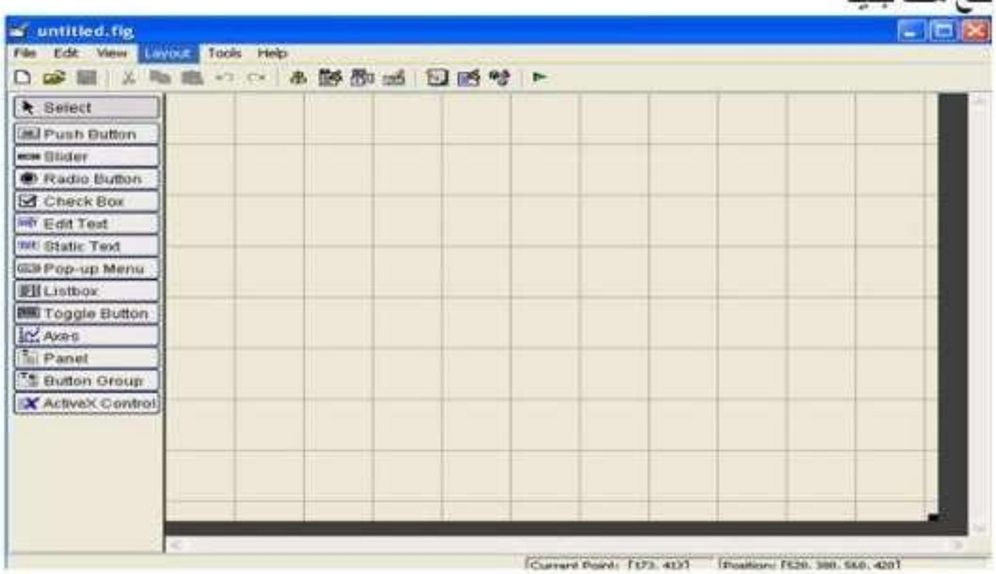
الغرض: التعرف علي برمجه ال

1-push button

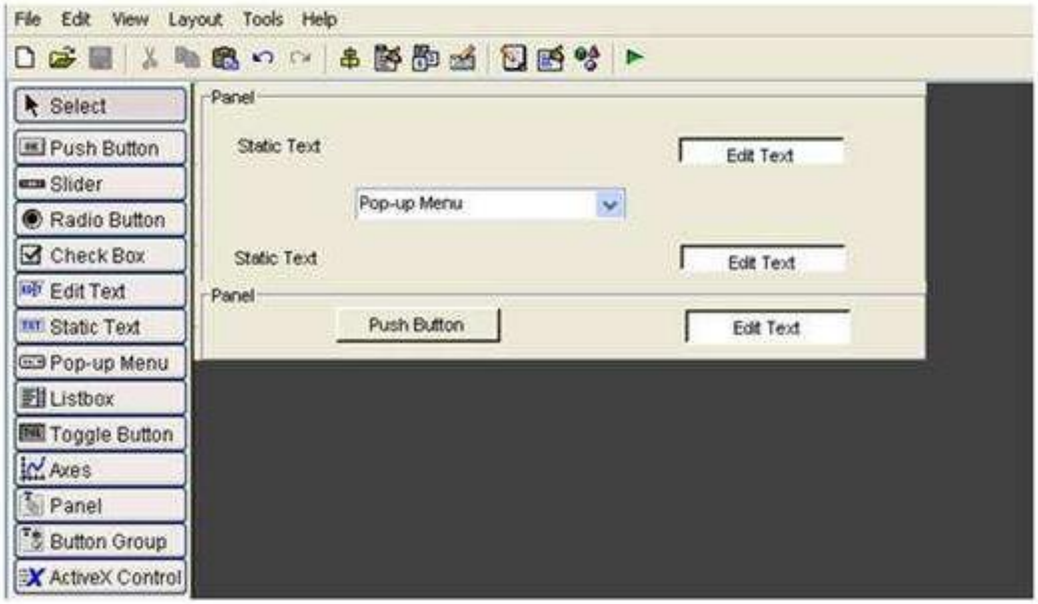
2-pop-up menu

الخطوات:

فتح ملف جديد



سحب وادراج العناصر علي الخلفيه كما يظهر في الصوره ,ثم تصغير الخلفيه لتلائم التصميم



الوقوف علي كل عنصر منفصلا والضغط علي زر الماوس الايمن واختيار

Property inspector

والنزول من الجدول حتي الوصول الي هذه الخانه

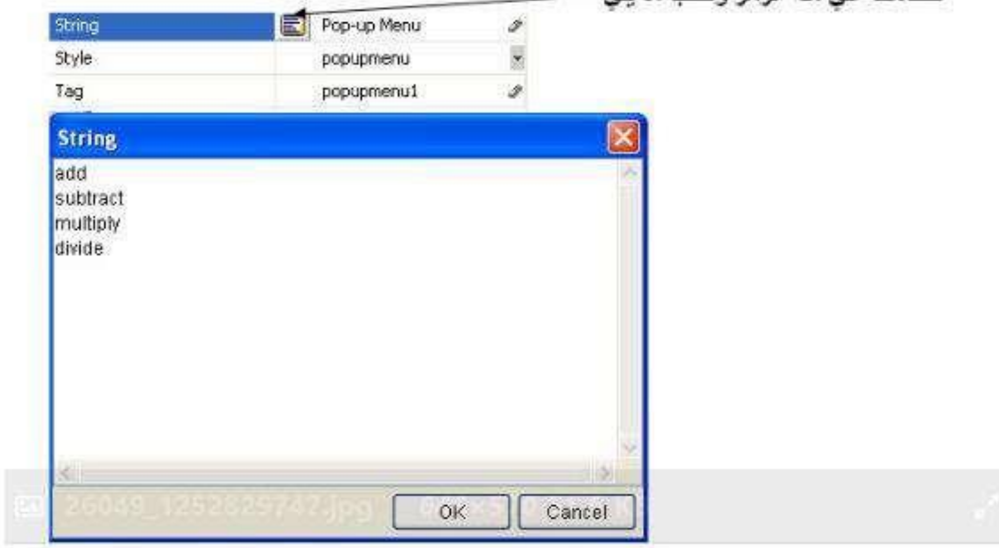
ثم تفريغ الخانه من الكلام في الثلاثه عناصر لل

Edit text

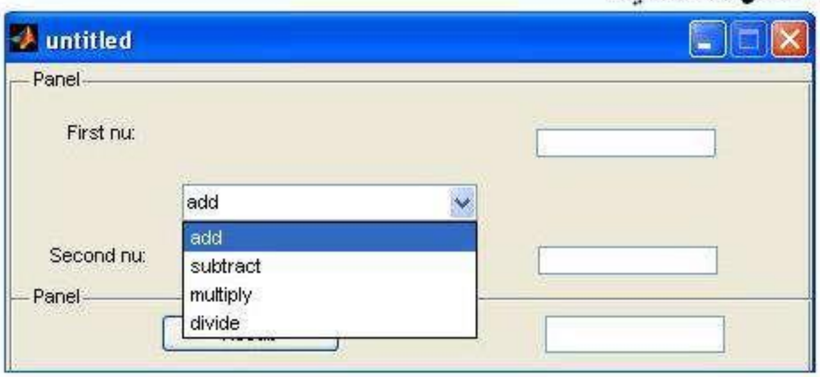
اما باقي العناصر فيكتب الاسم المراد ظهوره للمستخدم
في حاله ال

Pop-up menu

فنضغط على هذا الرمز ونكتب ما يلي



فيصبح البرنامج بهذا الشكل عند تشغيله

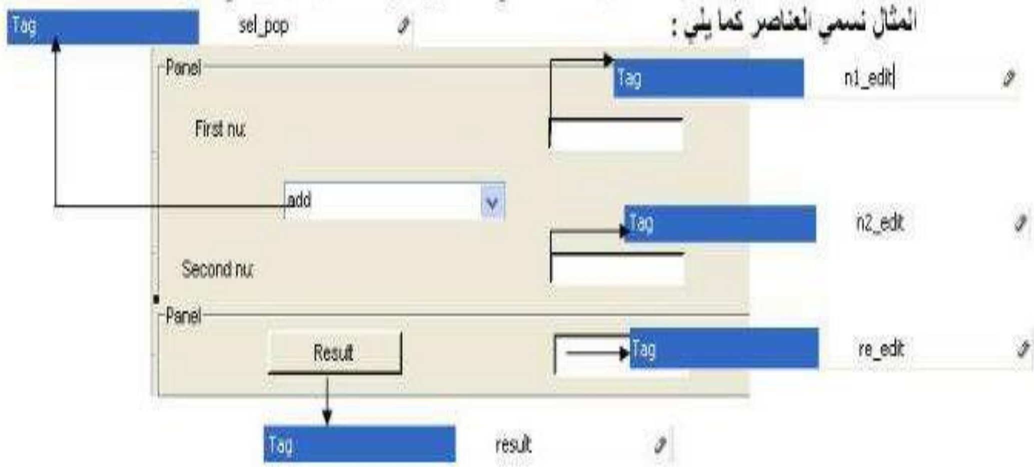


بالرجوع الي خلفيه التصميم والضغط علي الزر الايمن للماوس واختيار

Property inspector

والنزول في الجدول الي ان نصل الي هذه الخانه

من خلال هذه الخانه يتم تسميه العناصر بالاسماء التي نستدعيها بها اثناء البرمجه وعلي سبيل المثال نسمي العناصر كما يلي :



جميع الخطوات السابقه ادت الي وصولنا الي التصميم النهائي والمتبقي فقط هو برمجه هذا التصميم

لذلك علينا فهم اسلوب عمل البرنامج قبل بدايه البرمجه

عند الضغط علي

Result

يتم جلب الكتابه المكتوبه في المكان الاول وتحويلها الي ارقام لنستطيع التعامل معها

وكذلك الكتابه المكتوبه في المكان الثاني وتحويلها الي ارقام ايضا

وبعد ذلك يرى رقم الخيار الذي اختاره المستخدم للعمليه الحسابيه

فالرقم واحد يعني جمع

والرقم اثنان يعني الطرح

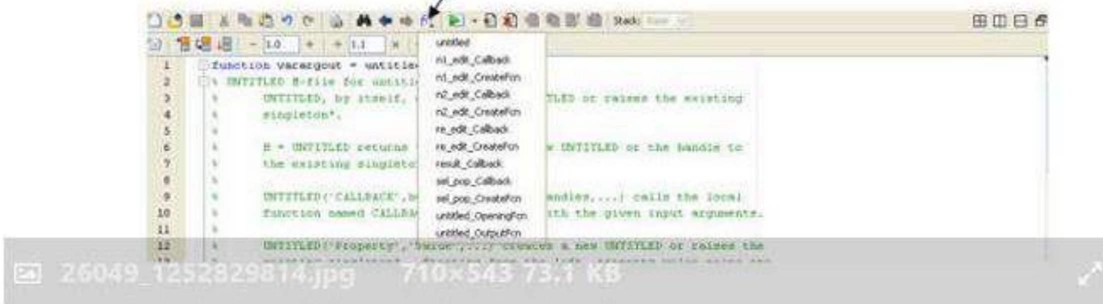
وثلاثه الضرب

واربعه تعني القسمه

الان علينا تنفيذ ما فهمناه

نقوم بالضغط علي هذا الرمز لفتح ملف ال ام فايل المرتبط بالتصميم

ثم الضغط على هذا الرمز داخل ال ام فايل لنري جميع العناصر



نختار الاسم هذا من القائمه للزهاب الي زر

عند اختيار هذا الاسم نجد ان الماوس وقف عند هذا الجزء من الكود

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

تحت الكتابه المكتوبه باللون الاخضر يتم كتابه كود برمجه الزر وهي كما يلي

```
n1=str2num(get(handles.n1_edit,'string'));
n2=str2num(get(handles.n2_edit,'string'));
sel=get(handles.sel_pop,'value');
switch sel
    case 1
        c=n1+n2;
    case 2
        c=n1-n2;
    case 3
        c=n1*n2;
    case 4
        c=n1/n2;
end
```

فتظهر النتيجة بالشكل النهائي: