Design and control of robotic vehicle using GUIDE in MATLAB

[1]*V.Sumanth*, [2]*SK.Nowshad*, [3]*J.Madhu*

[1]*Assistant Professor*, [2] *Assistant Professor*, [3] *Assistant Professor*

[1]*Electronics and Instrumentation Engineering,*

[1]*Narayana Engineering College, Nellore,India*

Abstract : **This robotic vehicle is a unique blend of software and hardware. It is a mobile platform that user can control through his computer/Laptop with proper feedback through wire.**

The Graphical User Interface (GUI) is made through MATLAB software which gives an opportunity to user to understand basic MATLAB and its toolboxes. It also helps to understand serial interfacing of devices through MATLAB.

In this paper the real time control of robotic vehicle is implemented using MATLAB. This paper exploits the serial communication capability of microcontrollers and the MATLAB software along with graphical design tools of MATLAB to create a MATLAB based Graphical User Interface (GUI) environment for ATMEGA48P microcontroller. MATLAB based GUI environment allows data acquisition, data processing, data visualization and control.

Index Terms – **Autonomous Ground Vehicle (AGV),GUIDE, Mobile Robotics.**

I. Introduction

Robotic Vehicles are physical programmable machines that have sensors and actuators, and are given goals for what they should achieve in the world. Algorithms process the sensors inputs, a control program decides how a robot should behave given its goals and current circumstances, and commands are sent to the motors to make the robot act in the world. Some robots are mobile but others are rooted to a fixed location. There are a lot more that robots are capable of, and many more research challenges beyond navigation that will enable new capabilities.

BLOCK DIAGRAM

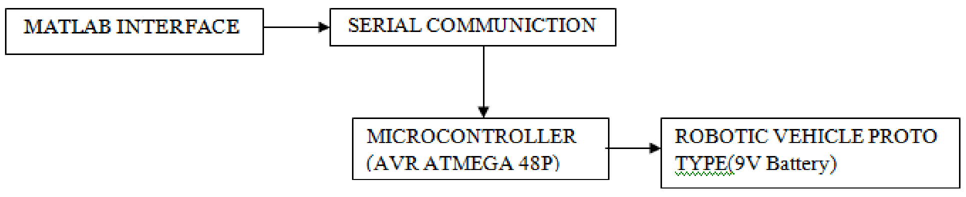

MATLAB INTERFACE
SERIAL COMMUNICTION
MICROCONTROLLER (AVR ATMEGA 48P)
ROBOTIC VEHICLE PROTO TYPE(9V Battery)

Fig 1: Block Diagram of Design and control of robotic vehicle using GUIDE in MATLAB

Robotic Vehicle can move in all directions from origin like left, right, forward, backward. It is controlled by giving directions from Personal Computer/Laptop. The AVR ATMEGA48P MICROCONTROLLER is used to dump the code from PC to control the robotic vehicle. Coding is performed based on MATLAB and ARDUINO1.0.5 softwares.

Hardware Requirements:

The following components are used to develop Robotic Vehicle .

1) USB (Universal Serial Bus)
2) Personal Computer
3) AVR ATMEGA48P microcontroller
4) ARDUINO Board
5) IC L293D
6) Power Supply (9V Battery)

Software Requirements:

The MATLAB 7.1 and ARDUINO 1.0.5 are for controlling the robot through personal computer.

II. INTRODUCTION TO MATLAB

The name MATLAB stands for Matrix Laboratory. MATLAB was written originally to provide easy access to matrix software developed by the LINPACK (linear system package)and EISPACK (Eigen system package) projects.

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming environment. Furthermore, MATLAB is a modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming. These factors make MATLAB an excellent tool for teaching and research.

MATLAB has many advantages compared to conventional computer languages (e.g., FORTRAN) for solving technical problems. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. The software package has been commercially available since 1984 and is now considered as a standard tool at most universities and industries worldwide.

It has powerful built-in routines that enable a very wide variety of computations. It also has easy to use graphics commands that make the visualization of results immediately available. Specific applications are collected in packages referred to as toolbox. There are toolboxes for signal processing, symbolic computation, control theory, simulation, optimization, and several other fields of applied science and engineering.

III.GUI (Graphical User Interface)

A graphical user interface (GUI) is a graphical display in one or more windows containing controls, called components that enable a user to perform interactive tasks. The user of the GUI 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 of a GUI need not understand the details of how the tasks are performed. GUI components can include menus, toolbars, push buttons, radio buttons, list boxes, and sliders—just to name a few. GUIs created using MATLAB tools can also perform any type of computation, read and write data files, communicate with other GUIs, and display data as tables or as plots. The following figure illustrates a simple GUI that can be easily build.

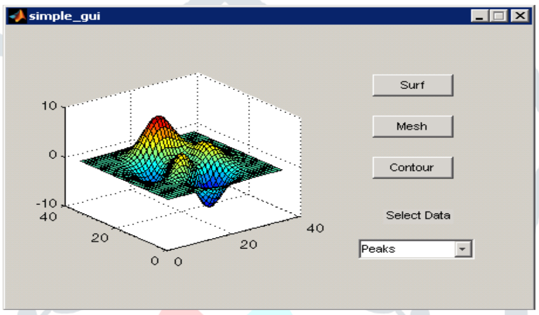


Fig 2: Illustration for GUI (Graphical User Interface)

The GUI contains:

1) An axes component
2) 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.
3) Three buttons that provide different kinds of plots: surface, mesh, and contour.

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

3.1 Working of GUI

Typically, GUIs wait for an end user to manipulate a control, and then respond to each user action in turn. Each control, and the GUI itself, has one or more call backs, named for the fact that they "call back" to MATLAB to ask it to do things. A particular user action, such as pressing a screen button, or passing the cursor over a component, triggers the execution of each call-back. The GUI then responds to these events. You, as the GUI creator, write call-backs that define what the components do to handle events.

This kind of programming is often referred to as event-driven programming. In event-driven programming, call back execution is asynchronous, that is, events external to the software trigger call back execution. In the case of MATLAB GUIs, most events are user interactions with the GUI, but the GUI can respond to other kinds of events as well, for example, the creation of a file or connecting a device to the computer.

We can code call backs in two distinct ways:

1) As MATLAB language functions stored in files.
2) As strings containing MATLAB expressions or commands (such as 'c = sqrt (a*a + b*b);'or 'print').

Using functions stored in code files as call backs is preferable to using strings, because functions have access to arguments and are more powerful and flexible. We cannot use MATLAB scripts (sequences of statements stored in code files that do not define functions) as call backs.

Although you can provide a call back with certain data and make it do anything you want, you cannot control when call backs execute. That is, when your GUI is being used, we have no control over the sequence of events that trigger particular call backs or what other call backs might still be running at those times. This distinguishes event-driven programming from other types of control flow, for example, processing sequential data files.

3.2 Create a Simple GUIDE GUI:

This example shows how to create a simple GUIDE graphical user interface (GUI), such as shown in the following figure.

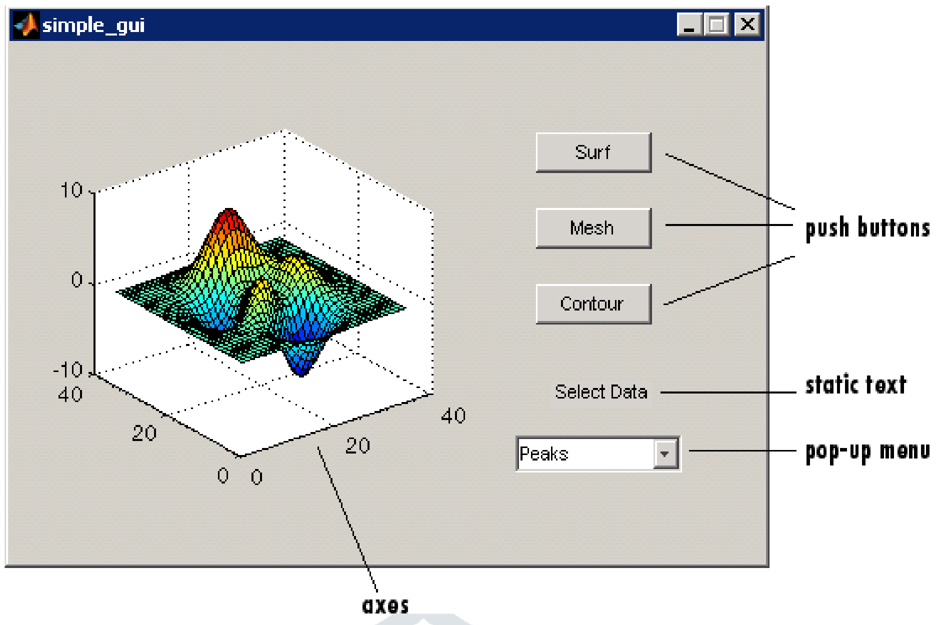


Fig 3: A GUI Interface to create a simple GUIDE

Subsequent topics guide the process of creating this GUI.

If we prefer to view and run the code that created this GUI without creating it, set the current folder to one to which we have write access. Copy the example code and open it in the Editor by issuing the following MATLAB commands:

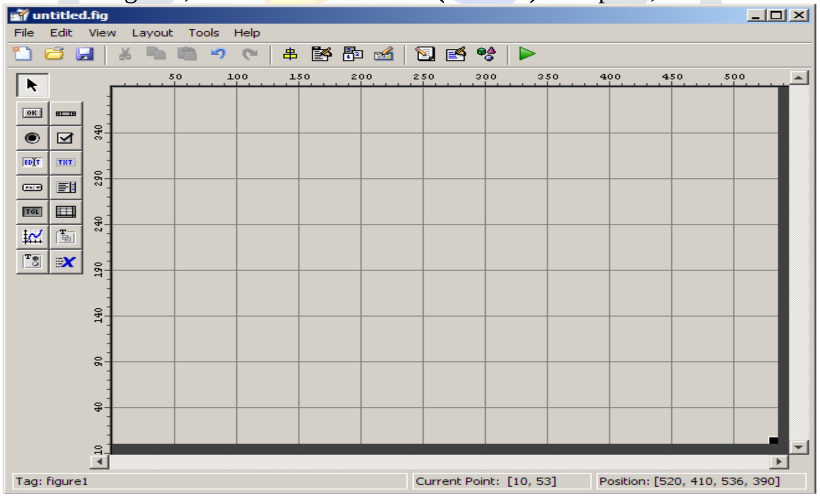
```
Copy file (fullfile (docroot, 'techdoc','creating_guis'...
'examples','simple_gui*.*')),fileattrib('simple_gui*.*', '+w');
guide simple_gui.fig;
edit simple_gui.m
```

- To run the GUI, on the **Editor** tab, in the **Run** section, click **Run.**

3.3 Open a New GUI in the GUIDE Layout Editor

1) Start GUIDE by typing guide at the MATLAB prompt.

2 In the GUIDE Quick Start dialog box, select the **Blank GUI (Default)** template, and then click **OK**.



Tag: figure1 | Current Point: [10, 53] | Position: [520, 410, 536, 390]

Fig 4 : Blank GUI Template

1) Display the names of the GUI components in the component palette:

- Select **File** > **Preferences** > **GUIDE**.
- Select **Show names in component palette**.
- Click **OK**.

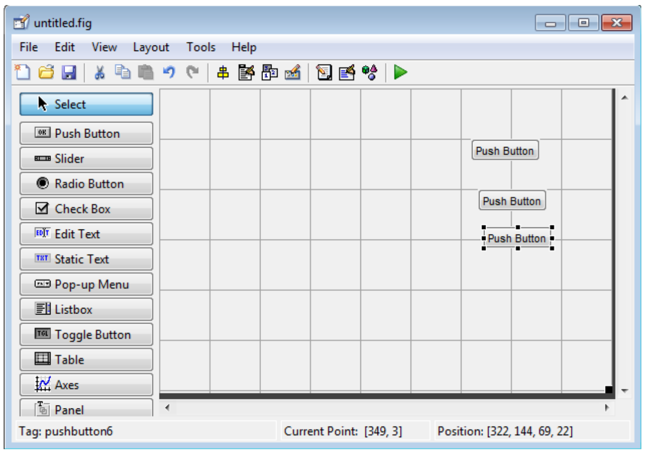
3.4 Set the GUI Figure Size in GUIDE:

Set the size of the GUI by resizing the grid area in the Layout Editor. Click the lower-right corner and drag it until the GUI is approximately 3 in. High and 4 in. wide. If necessary, make the window larger.

3.5 Layout the Simple GUIDE GUI:

Add, align, and label the components in the GUI.

1) Add the three push buttons to the GUI. 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.



Tag: pushbutton6 Current Point: [349, 3] Position: [322, 144, 69, 22]

Fig 5: Graphical Interface To Create Push Buttons

2) Add the remaining components to the GUI.
- A static text area
- A pop-up menu
- An axes

Resize the axes component to approximately 2-by-2 inches.

IV. AURDUINO

The Arduino is an easy to use yet powerful single board computer that has gained considerable traction in the hobby and professional market. The Arduino is open-source, which means hardware is reasonably priced and development software is free. The Arduino programming language is a simplified version of C/C++.

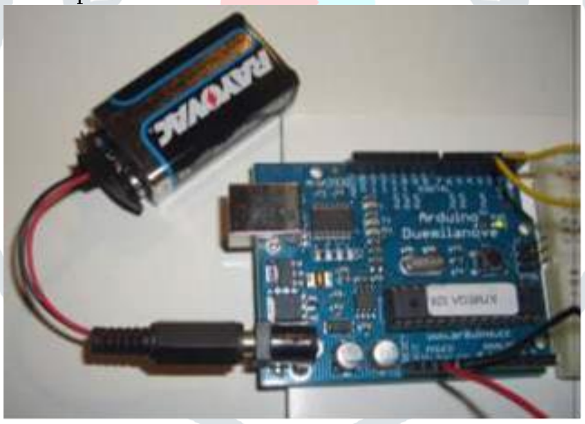


Fig 6: ARDUINO 1.0.5 Board

The board has 14 digital I/O pins and 6 analog input pins. There is a USB connector for talking to the host computer and a DC power jack for connecting an external 6-20 V power source, for example a 9 V battery, when running a program while not connected to the host.

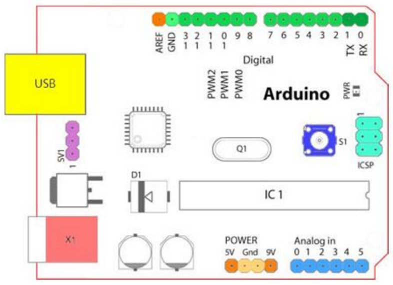


Fig 7: Pins for Arduino 1.0.5

4.1 Pin Details:

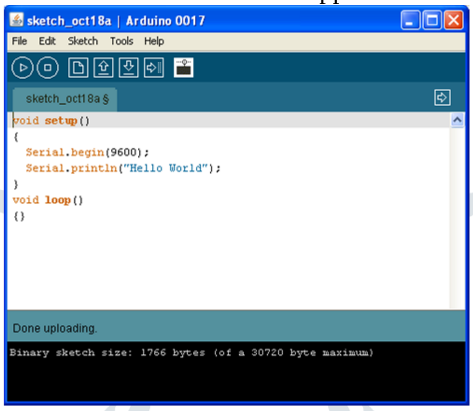
1) Analog Reference pin (orange)
2) Digital Ground (light green)
3) Digital Pins 2-13 (green)
4) Digital Pins 0-1/Serial In/Out - TX/RX (dark green) - These pins cannot be used for digital I/O (digital Read and digital Write) if you are also using serial communication (e.g. Serial. Begin).
5) Reset Button - S1 (dark blue)

6) In-circuit Serial Programmer (blue-green)
7) Analog In Pins 0-5 (light blue)
8) Power and Ground Pins (power: orange, grounds: light orange)
9) External Power Supply In (9-12VDC) - X1 (pink)
10) Toggles External Power and USB Power (place jumper on two pins closest to desired supply) - SV1 (purple)
11) USB (used for uploading sketches to the board and for serial communication between the board and the computer; can be used to power the board) (yellow)

4.2 Digital Pins:

In addition to the specific functions listed below, the digital pins on an Arduino board can be used for general purpose input and output via the pin Mode (), digital Read (), and digital Write () commands. Each pin has an internal pull-up resistor which can be turned on and off using digital Write () (w/ a value of HIGH or LOW, respectively) when the pin is configured as an input.

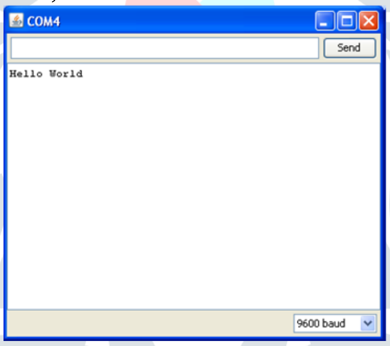
In Arduino, programs are called “sketches”, but here we will just call them programs. In the editing window that comes up, enter the following program, paying attention to where semi-colons appear at the end of command lines.



```
void setup()
{
  Serial.begin(9600);
  Serial.println("Hello World");
}
void loop()
{}
```

```
Binary sketch size: 1766 bytes (of a 30720 byte maximum)
```

Click the Upload button or Ctrl-U to compile the program and load on the Arduino board.
Click the Serial Monitor button. If all has gone well, the monitor window will show your message and look something like this



```
Hello World
```

4.3 Program Structure:

All Arduino programs have two functions, setup () and loop (). The instructions you place in the startup () function are executed once when the program begins and are used to initialize. Use it to set directions of pins or to initialize variables. The instructions placed in loop are executed repeatedly and form the main tasks of the program. Therefore every program has this structure.

Void setup ()
{
// commands to initialize go here
}
Void loop ()
{
// commands to run your machine go here
}

4.4 PinMode:

This command, which goes in the setup() function, is used to set the direction of a digital I/O pin. Set the pin to OUTPUT if the pin is driving and LED, motor or other device. Set the pin to INPUT if the pin is reading a switch or other sensor. On power up or reset, all pins default to inputs. This example sets pin 2 to an output and pin 3 to an input.

Void setup ()
{
PinMode (2, OUTPUT);
PinMode (3, INPUT);
}
Void loop () {}

4.5 Serial. Print:

The Serial. Print command lets you see what's going on inside the Arduino from your computer. For example, you can see the result of a math operation to determine if you are getting the right number. Or, you can see the state of a digital input pin to see if the Arduino is a sensor or switch properly. When your interface circuits or program does not seem to be working, use the Serial. Print command to shed a little light on the situation. For this command to show anything, you need to have the Arduino connected to the host computer with the USB cable.

For the command to work, the command Serial.begin (9600) must be placed in the setup () function. After the program is uploaded, you must open the Serial Monitor window to see the response.

There are two forms of the print command. Serial.Print () prints on the same line while Serial.println() starts the print on a new line.

Here is a brief program to check if your board is alive and connected to the PC.

```
Void setup ()
{
Serial.begin(9600);
Serial.println("Hello World");
}
Void loop () {}
```

4.6 digital Write:

This command sets an I/O pin high (+5V) or low (0V) and is the workhorse for commanding the outside world of lights, motors, and anything else interfaced to your board. Use the pinMode () command in the setup () function to set the pin to an output.

```
digitalWrite(2, HIGH); // sets pin 2 to +5 volts
digitalWrite (2, LOW); // sets pin 2 to zero volts
```

VI. ROBOTIC VEHICLE MOVEMENT

In practical applications (for example robotic vehicles) the polarity cannot be interchanged manually and hence there is a need to form a circuit using which the direction of current through a motor can be changed. One such commonly used circuit configuration is a H Bridge.

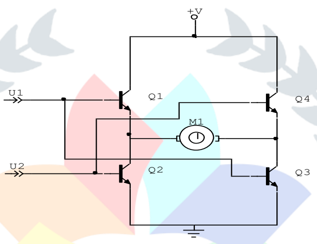
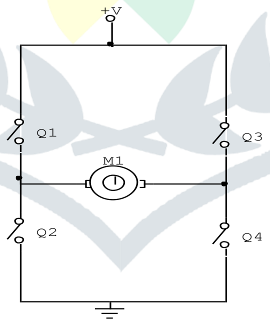
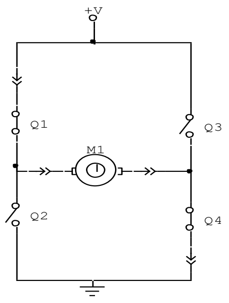


Fig 8: H-Bridge

An H Bridge is a circuit that is constructed using four switching devices (transistors / FET/ IGBT). A transistor based H bridge is shown above. Using the above circuit the direction of current through the motor (M1) can be changed, by controlling the four transistors.



To understand the working of an H Bridge the transistors can be thought of as four switches connected as shown above. When the switches Q1 and Q4 are closed, the current through the motor flows in the direction, as shown below.



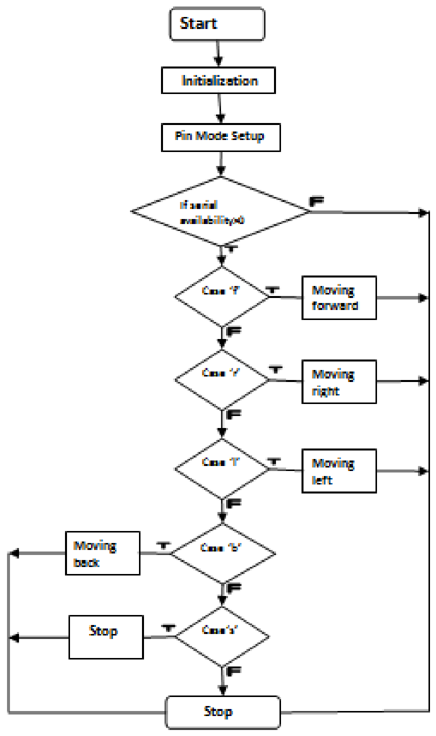
Similarly when the switches Q3 and Q2 are closed as shown below, the current flows in the opposite direction. Thus using a H Bridge the direction of current through a DC motor can be changed by controlling the four switching devices used in the bridge.

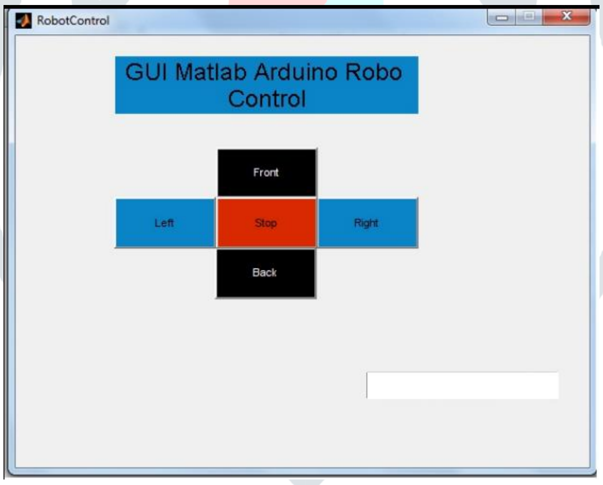
In the transistor circuit shown earlier the base of the Q1 and Q4 are connected together and similarly the base of the transistors Q2 and Q3 are also connected together. When a high signal is given to the input U1, transistors Q1 and Q4 turn on and thus allowing the current to flow through the motor in one direction. Alternately when the high signal is applied to U2, transistors Q2 and Q3 turn on, allowing the current to flow in the opposite direction.

Thus four switching devices connected in a H bridge configuration can be used to make a motor rotate in opposite directions, as required in most robotic applications.

IV. FLOWCHART AND RESULTS

Movement of robot and controlling of robot in different directions like left, right, forward, backward and stop using the software MATLAB and ARDUINO1.0.5.





REFERENCES

[1].“ROBOTICS” by K.S.FU, R.C.Gonznelz, C.S.G.Lee, Mc Graw Hill Publications
[2].R.U.Ayres and S.M.Miller Robotics, Applications and social Implications Ballinger, Cambridge
[3].P.K.Sinha:Microcontrollers:interfacing for real time applications: McGraw Hill co,1965