**SUMMER TRAINING REPORT**

**On**

**MATLAB**

[**Guru Gobind Singh Indraprastha University, Delhi (India**](https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CC0QFjAA&url=http%3A%2F%2Fwww.ggsipu.nic.in%2F&ei=vfIaUvV0wv-IB9nFgIAI&usg=AFQjCNFGwnhD8_pGr-fs5uw5WWtLN3dUYg&sig2=m6ysfK3TOY9J2VMB7gvtww)**)** **in partial fulfillment of the requirement for the award of the degree of**

**B.TECH**

**in**

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**Submitted to**

**Mr. SUSHIL KUMAR**

**DEPTT. OF ELECTRONICS AND COMMUNICATION ENGINEERING**



**Submitted by**

**Ayush Girdhar**

**ECE-1(5TH Semester)**

**MAHARAJA SURAJMAL INSTITUTE OF TECHNOLOGY**

**NEW DELHI 110058**

**AUGUST 2016**

**PREFACE**

A student gets theoretical knowledge from classroom and gets practical knowledge from summer training. When both these aspects of theoretical knowledge and practical experience together then a student is fully equipped to secure his best.

MATLAB helps in simplifying calculations of transfer functions of circuits, finding/plotting poles and zeroes, evaluating time-response of systems, and solving differential equations pertaining to them. Much of its functionality is dedicated to communications and signal processing, which is again of importance in electronics. One could build theoretical models for analogue, digital as well as mixed signal systems using MATLAB.

As a part of Guru Gobind Singh Indraprastha University syllabus all B.tech students are required to do 6 Weeks Summer Training after completion of their Second Year. So I took this opportunity to Learn **MATLAB**  from **Aedifico Tech Pvt. Ltd**

**ACKNOWLEDGEMENT**

A research work owes to success from commencement to completion. I would like to express my gratitude to those who help me in various stages of the study. I would like to thank Dr. Puneet Azad sir for allowing me to undergo six week MATLAB training. I gratefully acknowledge the effective guidance & constant encouragement received from our teacher Mr. Vaseem Durrani and Mr. Rohit Kumar who has been a source of great inspiration. Their constant motivation and guidance has helped me to complete this project

Also I would like to thank my friends and family for his utmost co-operation and support. The immense support and thorough supervision of my mentors have constantly been my guiding source which has made me cross all the hurdles while working on this project.

I thank them and am very obliged to work under their assistance

AYUSH GIRDHAR

ECE-1(5th Semester)

2014-2018

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.no** | **TOPIC** | **Page No** |
| 1 | Introduction | 6 |
| 2 | MATLAB | 6 |
| 3 | Morse Code | 11 |
| 4 | Morse Code Key | 13 |
| 5 | Embedded Systems | 14 |
| 6 | Arduino Uno | 18 |
| 7 | Arduino Board | 21 |
| 7 | Bluetooth | 22 |
| 8 | LCD | 23 |
| 9 | Dotted PCB | 24 |
| 10 | Buzzer | 26 |
| 11 | Objectives | 27 |
| 12 | Tools and Environment | 28 |
| 13 | Data Flow Diagram | 29 |
| 14 | Module Details | 30 |
| 15 | User interface | 34 |
| 16 | Circuit Diagram | 37 |
| 17 | Hardware Design | 38 |
| 18 | Testing | 39 |
| 19 | Future Scope | 40 |
| 20 | Bibliography | 41 |

**INTRODUCTION**

**MATLAB**

**MATLAB** (**mat**rix **lab**oratory) is a [multi-paradigm](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language) numerical computing environment and [fourth-generation programming language](https://en.wikipedia.org/wiki/Fourth-generation_programming_language). A [proprietary programming language](https://en.wikipedia.org/wiki/Proprietary_programming_language) developed by **[MathWorks](https://en.wikipedia.org/wiki/MathWorks" \o "MathWorks)**, MATLAB allows matrix manipulations, plotting of [functions](https://en.wikipedia.org/wiki/Function_(mathematics)) and data, implementation of [algorithms](https://en.wikipedia.org/wiki/Algorithm), creation of [user interfaces](https://en.wikipedia.org/wiki/User_interface), and interfacing with programs written in other languages, including [C](https://en.wikipedia.org/wiki/C_(programming_language)), [C++](https://en.wikipedia.org/wiki/C%2B%2B), [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), [Fortran](https://en.wikipedia.org/wiki/Fortran) and [Python](https://en.wikipedia.org/wiki/Python_(programming_language)).

Although MATLAB is intended primarily for numerical computing, an optional toolbox uses the [MuPAD](https://en.wikipedia.org/wiki/MuPAD" \o "MuPAD) [symbolic engine](https://en.wikipedia.org/wiki/Computer_algebra_system), allowing access to [symbolic computing](https://en.wikipedia.org/wiki/Symbolic_computing) abilities. An additional package, [Simulink](https://en.wikipedia.org/wiki/Simulink), adds graphical multi-domain simulation and [model-based design](https://en.wikipedia.org/wiki/Model-based_design) for [dynamic](https://en.wikipedia.org/wiki/Dynamical_system) and [embedded systems](https://en.wikipedia.org/wiki/Embedded_system).

Millions of engineers and scientists worldwide use MATLAB to analyze and design the systems and products transforming our world. MATLAB is in automobile active safety systems, interplanetary spacecraft, health monitoring devices, smart power grids, and LTE cellular networks. It is used for machine learning, signal processing, image processing, computer vision, communications, computational finance, control design, robotics, and much more.

**Math Graphics Programming**

The MATLAB platform is optimized for solving engineering and scientific problems. The matrix-based MATLAB language is the world’s most natural way to express computational mathematics. Built-in graphics make it easy to visualize and gain insights from data. A vast library of prebuilt toolboxes lets you get started right away with algorithms essential to your domain. The desktop environment invites experimentation, exploration, and discovery. These MATLAB tools and capabilities are all rigorously tested and designed to work together.

**Scale Integrate Deploy**

MATLAB helps you take your ideas beyond the desktop. You can run your analyses on larger data sets and scale up to clusters and clouds. MATLAB code can be integrated with other languages, enabling you to deploy algorithms and applications within web, enterprise, and production systems.

**Key Features**

* High-level language for scientific and engineering computing
* Desktop environment tuned for iterative exploration, design, and problem-solving
* Graphics for visualizing data and tools for creating custom plots
* Apps for curve fitting, data classification, signal analysis, and many other domain-specific tasks
* Add-on toolboxes for a wide range of engineering and scientific applications
* Tools for building applications with custom user interfaces
* Interfaces to C/C++, Java, .NET, Python, SQL, Hadoop, and Microsoft Excel
* Royalty-free deployment options for sharing MATLAB programs with end users

**MATLAB Speaks Math**

The matrix-based MATLAB language is the world’s most natural way to express computational mathematics. Linear algebra in MATLAB looks like linear algebra in a textbook. This makes it straightforward to capture the mathematics behind your ideas, which means your code is easier to write, easier to read and understand, and easier to maintain.

You can trust the results of your computations. MATLAB, which has [strong roots in the numerical analysis research community](http://in.mathworks.com/company/newsletters/articles/the-origins-of-matlab.html), is known for its impeccable numerics. A MathWorks team **of 350 engineers** continuously verifies quality by running millions of tests on the MATLAB code base every day.

MATLAB does the hard work to ensure your code runs quickly. Math operations are distributed across [**multiple cores**](http://in.mathworks.com/discovery/matlab-multicore.html) on your computer, library calls are heavily optimized, and [all code is just-in-time compiled](http://in.mathworks.com/products/matlab/matlab-execution-engine/). You can run your algorithms in parallel by changing for-loops into parallel for-loops or by changing standard arrays into GPU or distributed arrays. Run [**parallel algorithms**](http://in.mathworks.com/solutions/parallel-computing/) in infinitely scalable public or private clouds with no code changes.

The MATLAB language also provides features of traditional programming languages, including flow control, error handling, object-oriented programming, unit testing, and source control integration.

## MATLAB Is Designed for Engineers and Scientists

MATLAB provides a desktop environment tuned for iterative engineering and scientific workflows. Integrated tools support simultaneous exploration of data and programs, letting you evaluate more ideas in less time.

* You can interactively preview, select, and preprocess the data you want to [import](http://in.mathworks.com/help/matlab/data-import-and-export.html).
* An extensive set of built-in [math functions](http://in.mathworks.com/help/matlab/mathematics.html) supports your engineering and scientific analysis.
* 2D and 3D [plotting functions](http://in.mathworks.com/help/matlab/graphics.html) enable you to visualize and understand your data and communicate results.
* [MATLAB apps](http://in.mathworks.com/discovery/matlab-apps.html) allow you to perform common engineering tasks without having to program. Visualize how different algorithms work with your data, and iterate until you’ve got the results you want.
* The integrated [editing and debugging tools](http://in.mathworks.com/help/matlab/programming-and-data-types.html) let you quickly explore multiple options, refine your analysis, and iterate to an optimal solution.
* You can capture your work as [sharable, interactive narratives](http://in.mathworks.com/products/matlab/live-editor/).

Comprehensive, professional documentation written by engineers and scientists is always at your fingertips to keep you productive. Reliable, real-time technical support staff answers your questions quickly. And you can tap into the knowledge and experience of over 100,000 community members and MathWorks engineers on [MATLAB Central](http://in.mathworks.com/matlabcentral/), an open exchange for MATLAB and Simulink users.

MATLAB and [add-on toolboxes](http://in.mathworks.com/products/matlab/index.html#addon_products) are integrated with each other and designed to work together. They offer professionally developed, rigorously tested, field-hardened, and fully documented functionality specifically for scientific and engineering applications

## MATLAB Integrates Workflows

Major engineering and scientific challenges require broad coordination to take ideas to implementation. Every handoff along the way adds errors and delays.

MATLAB automates the entire path from research through production. You can:

* Build and [package custom MATLAB apps and toolboxes](http://in.mathworks.com/help/matlab/creating-help.html) to share with other MATLAB users.
* [Create standalone executables](http://in.mathworks.com/products/compiler/) to share with others who do not have MATLAB.
* Integrate with [C/C++, Java, .NET, and Python](http://in.mathworks.com/solutions/matlab-and-other-programming-languages/). Call those languages directly from MATLAB, or package MATLAB algorithms and applications for deployment within web, enterprise, and production systems.
* [Convert MATLAB algorithms to C](http://in.mathworks.com/products/matlab-coder/), [HDL](http://in.mathworks.com/products/hdl-coder/), and PLC code to run on embedded devices.
* Deploy MATLAB code to [run on production Hadoop systems](http://in.mathworks.com/discovery/matlab-mapreduce-hadoop.html).

MATLAB is also a key part of Model-Based Design, which is used for multi domain simulation, physical and discrete-event simulation, and verification and code generation. [**Simulink**](http://in.mathworks.com/products/simulink), **[Simscape](http://in.mathworks.com/products/simscape)**[,](http://in.mathworks.com/products/simscape) and **[Stateflow](http://in.mathworks.com/products/stateflow)** help us in [Model-Based Design](http://in.mathworks.com/solutions/model-based-design).

## DESKTOP ENVIRONMENT

The MATLABdesktop environment helps you run commands, manage files, and view results. The desktop layout and set preferences, such as fonts, keyboard shortcuts, and initial working folder can also be changed.

* [**Start-up and Shutdown**](http://in.mathworks.com/help/matlab/startup-and-shutdown.html)

Start-up command line flags, start up and shutdown files

* [**Basic Settings**](http://in.mathworks.com/help/matlab/basic-settings.html)

Desktop appearance, fonts, colors, keyboard shortcuts

* [**Add-Ons**](http://in.mathworks.com/help/matlab/add-ons.html)

Find, run, and install add-ons, including apps, toolboxes, support packages, and more

* [**Platform and License**](http://in.mathworks.com/help/matlab/matlab-version-and-license.html)

Information about current computer, license, product version

* [**Internationalization**](http://in.mathworks.com/help/matlab/internationalization.html)

Locale settings and messages

* [**Help and Support**](http://in.mathworks.com/help/matlab/help-and-support.html)

Product help, technical support

## DESKTOP BASICS

When you start MATLAB, the desktop appears in its default layout.

The desktop includes these panels:

* **Current Folder** — Access your files.
* **Command Window** — Enter commands at the command line, indicated by the prompt (>>).
* **Workspace** — Explore data that you create or import from files.
* **Command History**-View or enter commands that you entered at the command line.

**Morse code**

**Morse code** is a method of transmitting [text](https://en.wikipedia.org/wiki/Written_language) information as a series of on-off tones, lights, or clicks that can be directly understood by a skilled listener or observer without special equipment. The International Morse Code encodes the [ISO basic Latin alphabet](https://en.wikipedia.org/wiki/ISO_basic_Latin_alphabet), some extra Latin letters, the [Arabic numerals](https://en.wikipedia.org/wiki/Arabic_numerals) and a small set of punctuation and procedural signals ([pro signs](https://en.wikipedia.org/wiki/Prosigns_for_Morse_code)) as standardized sequences of short and long signals called "**dots**" and "**dashes**", or "**dits**" and "**dahs**", as in [amateur radio](https://en.wikipedia.org/wiki/Amateur_radio) practice. Because many non-English natural languages use more than the 26 Roman letters, extensions to the Morse alphabet exist for those languages.

Each Morse code symbol represents either a text character (letter or numeral) or a pro sign and is represented by a unique sequence of dots and dashes. The duration of a dash is three times the duration of a dot. Each dot or dash is followed by a short silence, equal to the dot duration. The letters of a word are separated by a space equal to three dots (one dash), and the words are separated by a space equal to seven dots. The dot duration is the basic unit of time measurement in code transmission. To increase the speed of the communication, the code was designed so that [the length of each character in Morse varies approximately inversely to its frequency of occurrence](https://en.wikipedia.org/wiki/Entropy_encoding) in English. Thus the most common letter in English, the letter "E", has the shortest code, a single dot.

Morse code is used by some [amateur radio operators](https://en.wikipedia.org/wiki/Amateur_radio_operator), although knowledge of and proficiency with it is no longer required for [licensing](https://en.wikipedia.org/wiki/Amateur_radio_license) in most countries. [Pilots](https://en.wikipedia.org/wiki/Aviator) and [air traffic controllers](https://en.wikipedia.org/wiki/Air_traffic_controllers) usually need only a cursory understanding. Aeronautical [navigational aids](https://en.wikipedia.org/wiki/Navigational_aid), such as [VORs](https://en.wikipedia.org/wiki/VHF_omnidirectional_range) and [NDBs](https://en.wikipedia.org/wiki/Non-directional_beacon), constantly identify in Morse code. Compared to voice, Morse code is less sensitive to poor signal conditions, yet still comprehensible to humans without a decoding device. Morse is therefore a useful alternative to synthesized speech for sending automated data to skilled listeners on voice channels. Many [amateur radio repeaters](https://en.wikipedia.org/wiki/Amateur_radio_repeater), for example, identify with Morse, even though they are used for voice communications.

In an emergency, Morse code can be sent by improvised methods that can be easily "keyed" on and off, making it one of the simplest and most versatile methods of [telecommunication](https://en.wikipedia.org/wiki/Telecommunication). The most common distress signal is [SOS](https://en.wikipedia.org/wiki/SOS) or three dots, three dashes and three dots, internationally recognized by treaty.

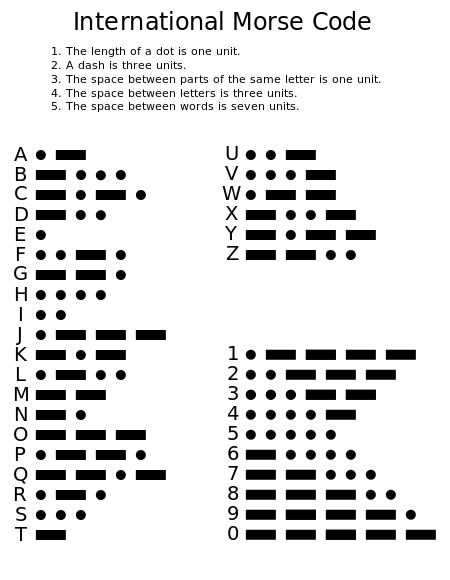
**Representation, timing and speed**

* short mark, dot or "dit" (·) : "dot duration" is one time unit long
* longer mark, dash or "dah" (–) : three time units long
* inter-element gap between the dots and dashes within a character : one dot duration or one unit long
* short gap (between letters) : three time units long
* medium gap (between words) : seven time units long

**Sectors of use**

* Aviation
* Amateur Radio
* Disaster Management

**Morse code key**



**Embedded Systems**

An embedded system is a [computer](https://en.wikipedia.org/wiki/Computer) [system](https://en.wikipedia.org/wiki/System) with a dedicated function within a larger mechanical or electrical system, often with [real-time computing](https://en.wikipedia.org/wiki/Real-time_computing) constraints. It is *embedded* as part of a complete device often including hardware and mechanical parts. Embedded systems control many devices in common use today. Ninety-eight percent of all [microprocessors](https://en.wikipedia.org/wiki/Microprocessors) are manufactured as components of embedded systems.

Examples of properties of typically embedded computers when compared with general-purpose counterparts are low power consumption, small size, rugged operating ranges, and low per-unit cost. This comes at the price of limited processing resources, which make them significantly more difficult to program and to interact with. However, by building intelligence mechanisms on top of the hardware, taking advantage of possible existing sensors and the existence of a network of embedded units, one can both optimally manage available resources at the unit and network levels as well as provide augmented functions, well beyond those available. For example, intelligent techniques can be designed to manage power consumption of embedded systems.

Modern embedded systems are often based on [microcontrollers](https://en.wikipedia.org/wiki/Microcontroller) (i.e. CPUs with integrated memory or peripheral interfaces), but ordinary microprocessors (using external chips for memory and peripheral interface circuits) are also common, especially in more-complex systems. In either case, the processor(s) used may be types ranging from general purpose to those specialised in certain class of computations, or even custom designed for the application at hand. A common standard class of dedicated processors is the [digital signal processor](https://en.wikipedia.org/wiki/Digital_signal_processor) (DSP).

Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from [economies of scale](https://en.wikipedia.org/wiki/Economies_of_scale).

Embedded systems range from portable devices such as [digital watches](https://en.wikipedia.org/wiki/Digital_watch) and [MP3 players](https://en.wikipedia.org/wiki/Digital_audio_player), to large stationary installations like [traffic lights](https://en.wikipedia.org/wiki/Traffic_light), [factory controllers](https://en.wikipedia.org/wiki/Programmable_logic_controller), and largely complex systems like [hybrid vehicles](https://en.wikipedia.org/wiki/Hybrid_vehicles), [MRI](https://en.wikipedia.org/wiki/MRI), and [avionics](https://en.wikipedia.org/wiki/Avionics). Complexity varies from low, with a single [microcontroller](https://en.wikipedia.org/wiki/Microcontroller) chip, to very high with multiple units, [peripherals](https://en.wikipedia.org/wiki/Peripheral) and networks mounted inside a large [chassis](https://en.wikipedia.org/wiki/Chassis) or enclosure.

An embedded system has three components:

* It has hardware.
* It has application software.
* It has Real Time Operating system (RTOS) that supervises the application software and provide mechanism to let the processor run a process as per scheduling by following a plan to control the latencies. RTOS defines the way the system works. It sets the rules during the execution of application program. A small scale embedded system may not have RTOS.

**Characteristics of an Embedded System**

* Must be **dependable**:

* **Reliability:** R(t) = probability of system working correctly provided that is was working at t=0
* **Maintainability:** M(d) = probability of system working correctly d time units after error occurred.
* **Availability:** probability of system working at time t
* **Safety:** no harm to be caused
* **Security:** confidential and authentic communication
* Must be **efficient**:
* **Energy** efficient
* **Code-size** efficient (especially for systems on a chip)
* **Run-time** efficient
* **Weight** efficient
* **Cost** efficient
* **Dedicated** towards a certain **application**: Knowledge about behaviour at design time can be used to minimize resources and to maximize robustness.
* **Dedicated** user **interface** (no mouse, keyboard and screen).
* Many ES must meet **real-time constraints**:
* A real-time system must **react to stimuli** from the controlled object (or the operator) within the time interval dictated by the environment.
* For real-time systems, right answers arriving too late (or even

Too early) are wrong.

* All other time-constraints are called soft.
* A **guaranteed system response** has to be explained without

Statistical arguments.

**Basic Structure of an Embedded System**

**Sensor** – It measures the physical quantity and converts it to an electrical signal which can be read by an observer or by any electronic instrument like an A2D converter. A sensor stores the measured quantity to the memory.

* **A-D Converter** – An analog-to-digital converter converts the analog signal sent by the sensor into a digital signal.
* **Processor & ASICs** – Processors process the data to measure the output and store it to the memory.
* **D-A Converter** – A digital-to-analog converter converts the digital data fed by the processor to analog data.
* **Actuator** – An actuator compares the output given by the D-A Converter to the actual (expected) output stored in it and stores the approved output.

**Microcontroller**

A **microcontroller** (or **MCU**, short for *microcontroller unit*) is a small computer ([SoC](https://en.wikipedia.org/wiki/System_on_a_chip" \o "System on a chip)) on a single integrated circuit containing a processor core, memory, and programmable [input/output](https://en.wikipedia.org/wiki/Input/output) peripherals. Program memory in the form of [Ferroelectric RAM](https://en.wikipedia.org/wiki/Ferroelectric_RAM), [NOR flash](https://en.wikipedia.org/wiki/NOR_flash) or [OTP ROM](https://en.wikipedia.org/wiki/Programmable_read-only_memory) is also often included on chip, as well as a typically small amount of [RAM](https://en.wikipedia.org/wiki/Random-access_memory). Microcontrollers are designed for embedded applications, in contrast to the [microprocessors](https://en.wikipedia.org/wiki/Microprocessor) used in [personal computers](https://en.wikipedia.org/wiki/Personal_computer) or other general purpose applications consisting of various discrete chips.

Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, implantable medical devices, remote controls, office machines, appliances, power tools, toys and other [embedded systems](https://en.wikipedia.org/wiki/Embedded_system). By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output devices, microcontrollers make it economical to digitally control even more devices and processes. [Mixed signal](https://en.wikipedia.org/wiki/Mixed-signal_integrated_circuit) microcontrollers are common, integrating analog components needed to control non-digital electronic systems.

.

**Arduino Uno**

**Arduino**, sold as **Genuino** [outside the U.S. and U.K.](https://en.wikipedia.org/wiki/Arduino#Trademark_dispute), is a hardware and software company, project, and user community that designs and manufactures computer [open-source hardware](https://en.wikipedia.org/wiki/Open-source_hardware), [open-source software](https://en.wikipedia.org/wiki/Open-source_software), and [microcontroller](https://en.wikipedia.org/wiki/Microcontroller)-based kits for building digital devices and interactive objects that can sense and control physical devices.

The project is based on microcontroller board designs, produced by several vendors, using various microcontrollers. These systems provide sets of digital and analog [input/output](https://en.wikipedia.org/wiki/Input/output) (I/O) pins that can interface to various expansion boards (termed *shields*) and other circuits. The boards feature serial communication interfaces, including Universal Serial Bus ([USB](https://en.wikipedia.org/wiki/USB)) on some models, for loading programs from personal computers. For programming the microcontrollers, the Arduino project provides an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) based on a programming language named [*Processing*](https://en.wikipedia.org/wiki/Processing_(programming_language)), which also supports the languages [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B).

The first Arduino was introduced in 2005, aiming to provide a low cost, easy way for novices and professionals to create devices that interact with their environment using [sensors](https://en.wikipedia.org/wiki/Sensor) and [actuators](https://en.wikipedia.org/wiki/Actuator). Common examples of such devices intended for beginner hobbyists include simple [robots](https://en.wikipedia.org/wiki/Robot), [thermostats](https://en.wikipedia.org/wiki/Thermostat), and motion detectors.

Arduino boards are available commercially in preassembled form, or as [do-it-yourself](https://en.wikipedia.org/wiki/Do-it-yourself) kits. The hardware design specifications are openly available, allowing the Arduino boards to be produced by anyone

An Arduino board historically consists of an [Atmel](https://en.wikipedia.org/wiki/Atmel) 8-, 16- or 32-bit AVR [microcontroller](https://en.wikipedia.org/wiki/Microcontroller) (although since 2015 other makers' microcontrollers have been used) with complementary components that facilitate programming and incorporation into other circuits. An important aspect of the Arduino is its standard connectors, which let users connect the CPU board to a variety of interchangeable add-on modules termed shields. Some shields communicate with the Arduino board directly over various pins, but many shields are individually addressable via an [I²C](https://en.wikipedia.org/wiki/I%C2%B2C) [serial bus](https://en.wikipedia.org/wiki/Serial_bus)—so many shields can be stacked and used in parallel. Before 2015, Official Arduinos had used the Atmel [megaAVR](https://en.wikipedia.org/wiki/MegaAVR" \o "MegaAVR) series of chips, specifically the ATmega8, ATmega168 [ATmega328](https://en.wikipedia.org/wiki/ATmega328), [ATmega1280](https://en.wikipedia.org/w/index.php?title=ATmega1280&action=edit&redlink=1), and [ATmega2560](https://en.wikipedia.org/w/index.php?title=ATmega2560&action=edit&redlink=1). In 2015, units by other producers were added. A handful of other processors have also been used by Arduino compatible devices. Most boards include a 5 V [linear regulator](https://en.wikipedia.org/wiki/Linear_regulator) and a 16 MHz [crystal oscillator](https://en.wikipedia.org/wiki/Crystal_oscillator) (or [ceramic resonator](https://en.wikipedia.org/wiki/Ceramic_resonator) in some variants), although some designs such as the LilyPad run at 8 MHz and dispense with the onboard voltage regulator due to specific form-factor restrictions. An Arduino's microcontroller is also pre-programmed with a [boot loader](https://en.wikipedia.org/wiki/Boot_loader) that simplifies uploading of programs to the on-chip [flash memory](https://en.wikipedia.org/wiki/Flash_memory), compared with other devices that typically need an external [programmer](https://en.wikipedia.org/wiki/Programmer_(hardware)). This makes using an Arduino more straightforward by allowing the use of an ordinary computer as the programmer. Currently, optiboot bootloader is the default bootloader installed on Arduino UNO.

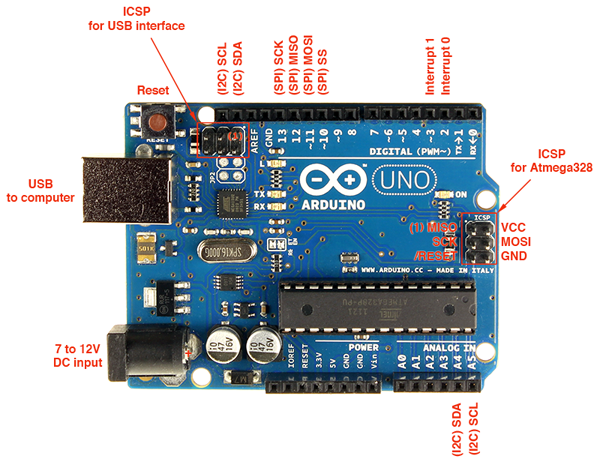
At a conceptual level, when using the Arduino integrated development environment, all boards are programmed over a serial connection. Its implementation varies with the hardware version. Some serial Arduino boards contain a level shifter circuit to convert between [RS-232](https://en.wikipedia.org/wiki/RS-232)logic levels and [transistor–transistor logic](https://en.wikipedia.org/wiki/Transistor%E2%80%93transistor_logic) (TTL) level signals. Current Arduino boards are programmed via [Universal Serial Bus](https://en.wikipedia.org/wiki/Universal_Serial_Bus) (USB), implemented using USB-to-serial adapter chips such as the [FTDI](https://en.wikipedia.org/wiki/FTDI) FT232. Some boards, such as later-model Uno boards, substitute the FTDI chip with a separate AVR chip containing USB-to-serial firmware, which is reprogrammable via its own ICSP header. Other variants, such as the Arduino Mini and the unofficial Boarduino, use a detachable USB-to-serial adapter board or cable,[Bluetooth](https://en.wikipedia.org/wiki/Bluetooth" \o "Bluetooth) or other methods, when used with traditional microcontroller tools instead of the Arduino IDE, standard AVR [in-system programming](https://en.wikipedia.org/wiki/In-system_programming) (ISP) programming is used.

The Arduino board exposes most **14 digital I/O pins**, six of which can produce [**pulse-width modulated**](https://en.wikipedia.org/wiki/Pulse-width_modulation) signals, and six analog inputs, which can also be used as six digital I/O pins. These pins are on the top of the board, via female 0.1-inch (2.54 mm) headers. Several plug-in application shields are also commercially available. The Arduino Nano, and Arduino-compatible Bare Bones Board and Boarduino boards may provide male header pins on the underside of the board that can plug into solderless [breadboards](https://en.wikipedia.org/wiki/Breadboard).

**Technical specs**

|  |  |
| --- | --- |
| **Microcontroller** | [**ATmega328P**](http://www.atmel.com/Images/doc8161.pdf) |
| **Operating Voltage** | 5V |
| **Input Voltage (recommended)** | 7-12V |
| **Input Voltage (limit)** | 6-20V |
| **Digital I/O Pins** | 14 (of which 6 provide PWM output) |
| **PWM Digital I/O Pins** | 6 |
| **Analog Input Pins** | 6 |
| **DC Current per I/O Pin** | 20 mA |
| **DC Current for 3.3V Pin** | 50 mA |
| **Flash Memory** | 32 KB (ATmega328P) of which 0.5 KB used by bootloader |
| **SRAM** | 2 KB (ATmega328P) |
| **EEPROM** | 1 KB (ATmega328P) |
| **Clock Speed** | 16 MHz |
| **Length** | 68.6 mm |
| **Width** | 53.4 mm |
| **Weight** | 25 g |

**Arduino Board**

**Bluetooth**

**Bluetooth**is a [wireless](https://en.wikipedia.org/wiki/Wireless) technology standard for exchanging data over short distances (using short-wavelength [UHF](https://en.wikipedia.org/wiki/UHF) [radio waves](https://en.wikipedia.org/wiki/Radio_waves) in the [ISM band](https://en.wikipedia.org/wiki/ISM_band) from 2.4 to 2.485 GHz) from fixed and mobile devices, and building [personal area networks](https://en.wikipedia.org/wiki/Personal_area_network) (PANs). Invented by telecom vendor [Ericsson](https://en.wikipedia.org/wiki/Ericsson) in 1994, it was originally conceived as a wireless alternative to [RS-232](https://en.wikipedia.org/wiki/RS-232) data cables. It can connect several devices, overcoming problems of synchronization.

Bluetooth is managed by the [**Bluetooth Special Interest Group**](https://en.wikipedia.org/wiki/Bluetooth_Special_Interest_Group) (SIG), which has more than 25,000 member companies in the areas of telecommunication, computing, networking, and consumer electronics. The [IEEE](https://en.wikipedia.org/wiki/Institute_of_Electrical_and_Electronics_Engineers) standardized Bluetooth as **IEEE 802.15.1**, but no longer maintains the standard. The Bluetooth SIG oversees development of the specification, manages the qualification program, and protects the trademarks. A manufacturer must make a device meet [Bluetooth SIG standards](https://en.wikipedia.org/wiki/Bluetooth_Special_Interest_Group#Qualification) to market it as a Bluetooth device. A network of [patents](https://en.wikipedia.org/wiki/Patent) apply to the technology, which are licensed to individual qualifying devices

The development of the "**short-link**" radio technology, later named **Bluetooth**, was initiated in 1989 by Dr. Nils Rydbeck CTO at Ericsson Mobile in [Lund](https://en.wikipedia.org/wiki/Lund) and Dr. [Johan Ullman](https://en.wikipedia.org/wiki/Johan_Ullman). The purpose was to develop wireless headsets, according to two inventions by [Johan Ullman](https://en.wikipedia.org/wiki/Johan_Ullman), [SE 8902098-6](http://worldwide.espacenet.com/textdoc?DB=EPODOC&IDX=SE8902098-6), issued 1989-06-12 and [SE 9202239](http://worldwide.espacenet.com/textdoc?DB=EPODOC&IDX=SE9202239), issued 1992-07-24. Nils Rydbeck tasked Tord Wingren with specifying and [Jaap Haartsen](https://en.wikipedia.org/wiki/Jaap_Haartsen" \o "Jaap Haartsen) and Sven Mattisson with developing. Both were working for [Ericsson](https://en.wikipedia.org/wiki/Ericsson) in [Lund](https://en.wikipedia.org/wiki/Lund), Sweden. The specification is based on [frequency-hopping spread spectrum](https://en.wikipedia.org/wiki/Frequency-hopping_spread_spectrum) technology.

**LCD**

A **liquid crystal display** (**LCD**) is a thin, flat display device made up of any number of color or monochrome pixels arrayed in front of a light source or reflector. It is often utilized in battery powered electronic devices because it uses very small amounts of electric power.

The LCD is a dot matrix liquid crystal display that displays alphanumeric, Kana (Japanese) character and symbols. The built - in controller & driver LSIs provide convenient connectively between a dot matrix LCD and most 4 or 8 bit microprocessors or microcontrollers. All the functions required for dot matrix liquid crystal display drive are internally provided. Internal refresh is provided by the LCD. The CMOS technology makes the device ideal for application in hand held, portable and other battery powered instruments with low power consumption.

**FEATURES:**

* Easy interface with a 4-bit or 8-bit MPU.
* Built-in dot matrix LCD controller with font 5 X 7 or 5 X 10 dots.
* Display data RAM for 80 characters (80 X 8bits).
* Character generator ROM, which provides 160 characters with font 5 X 7 dots and 32 characters with font 5 X 10 dots.
* Both display data and character generator RAMs can be read from the

MPU. Wide range of instruction functions: Clear display, cursor home, Display ON/OFF, Cursor Shift, Display Shift.

**Dotted PCB**

Perfboard is a material for [prototyping](https://en.wikipedia.org/wiki/Prototyping) [electronic circuits](https://en.wikipedia.org/wiki/Electronic_circuits) (also called DOT PCB). It is a thin, rigid sheet with holes pre-drilled at standard intervals across a grid, usually a square grid of 2.54 mm (0.1 in) spacing. These holes are ringed by round or square copper pads. Inexpensive perfboard may have pads on only one side of the board, while better quality perfboard can have pads on both sides ([plate-through holes](https://en.wikipedia.org/wiki/Via_(electronics))). Since each pad is electrically isolated, the builder makes all connections with either [wire wrap](https://en.wikipedia.org/wiki/Wire_wrap) or miniature [point to point wiring](https://en.wikipedia.org/wiki/Point_to_point_wiring) techniques. Discrete components are soldered to the prototype board such as [resistors](https://en.wikipedia.org/wiki/Resistor), [capacitors](https://en.wikipedia.org/wiki/Capacitor), and [integrated circuits](https://en.wikipedia.org/wiki/Integrated_circuit). The substrate is typically made of paper laminated with [phenolic resin](https://en.wikipedia.org/wiki/Phenolic_resin) (such as [FR-2](https://en.wikipedia.org/wiki/Synthetic_resin_bonded_paper)) or a fiberglass-reinforced epoxy laminate ([FR-4](https://en.wikipedia.org/wiki/FR-4)).

The 0.1 in grid system accommodates integrated circuits in [DIP](https://en.wikipedia.org/wiki/Dual_inline_package) packages and many other types of [through-hole](https://en.wikipedia.org/wiki/Through-hole_technology) components. Perfboard is not designed for prototyping [surface mount devices](https://en.wikipedia.org/wiki/Surface_mount_devices).

Before building a circuit on perfboard, the locations of the components and connections are typically planned in detail on paper or with [software tools](https://en.wikipedia.org/wiki/Electronic_design_automation). Small scale prototypes, however, are often built ad hoc, using an oversized perfboard.

Software for [PCB](https://en.wikipedia.org/wiki/Printed_circuit_board) layout can often be used to generate perfboard layouts as well. In this case, the designer positions the components so all leads fall on intersections of a 0.1 in grid. When routing the connections more than 2 copper layers can be used, as multiple overlaps are not a problem for insulated wires.

Once the layout is finalized, the components are soldered in their designated locations, paying attention to orientation of polarized parts such as electrolytic capacitors, diodes, and integrated circuits. Next, electrical connections are made as called for in the layout.

One school of thought is to make as many connections as possible without adding extra wire. This is done by bending the existing leads on resistors, capacitors, etc. into position, trimming off extra length, and soldering the lead to make the required electrical connection. Another school of thought refuses to bend the excessive leads of components and use them for wiring, on the ground that this makes removing a component later hard or impossible, e.g. when a repair is needed.

If extra wires need to be used, or are used for principle reasons, they are typically routed entirely on the copper side of perfboards. Because, as opposite to strip boards, nearby holes aren't connected, and the only hole in a pad is already occupied by a component's lead. Wires used range from isolated wires, including verowire (enameled copper wire with a polyurethane insulation supposed to melt when soldered), to bare copper wire, depending on individual preference, and often also on what is currently at hand in the workshop.

For insulated wires thin solid core wire with temperature-resistant insulation such as [Kynar](https://en.wikipedia.org/wiki/Kynar" \o "Kynar) or Tefzel is preferred. The wire gauge is typically 24 - 30 [AWG](https://en.wikipedia.org/wiki/American_wire_gauge). A special stripping tool can be used, incorporating a thin steel blade with a slit that the wire is simply inserted into and then pulled loose, leaving a clean stripped end. This wire was developed initially for circuit assembly by the [wire wrap](https://en.wikipedia.org/wiki/Wire_wrap) technique but also serves well for miniature point-to-point wiring on perfboard. Bare copper wire is useful when merging a number of connections to form an [electrical bus](https://en.wikipedia.org/wiki/Electrical_bus) such as the circuit's [ground](https://en.wikipedia.org/wiki/Electrical_ground), and when there is enough space to properly route connections, instead of wiring them rats-nest style.

**BUZZER**

A **buzzer** or **beeper** is an [audio](https://en.wikipedia.org/wiki/Sound) signaling device, which may be **mechanical,** [**electromechanical**](https://en.wikipedia.org/wiki/Electromechanics), or [**piezoelectric**](https://en.wikipedia.org/wiki/Piezoelectricity). Typical uses of buzzers and beepers include [alarm devices](https://en.wikipedia.org/wiki/Alarm_devices), [timers](https://en.wikipedia.org/wiki/Timer) and confirmation of user input such as a mouse click or keystroke.

**Types of buzzers**

### Electromechanical

Early devices were based on an electromechanical system identical to an [electric bell](https://en.wikipedia.org/wiki/Electric_bell) without the metal gong. Often these units were anchored to a wall or ceiling to use it as a sounding board. The word "buzzer" comes from the rasping noise that electromechanical buzzers made.

### Mechanical

A [joy buzzer](https://en.wikipedia.org/wiki/Joy_buzzer) is an example of a purely mechanical buzzer. They require drivers.

### Piezoelectric

A piezoelectric element may be driven by an [oscillating](https://en.wikipedia.org/wiki/Oscillation) electronic circuit or other [audio signal](https://en.wikipedia.org/wiki/Audio_signal) source, driven with a [piezoelectric audio amplifier](https://en.wikipedia.org/wiki/Piezoelectric_audio_amplifier). Sounds commonly used to indicate that a button has been pressed are a click, a ring.

**Objective**

The prime objective of the project is to develop a project with MATLAB and Embedded Systems which can perform communication between microcontroller and the MATLAB.

The **Morse Code Transmitter and Receiver** perform the communication between PIC microcontroller and MATLAB using serial communication to send and receive data using **Bluetooth**.

It acts as a two way system which can perform transmission as well as receiving so that the trainer can analyse his performance himself, without any other person’s involvement.

It can also serve as a **Handbook** for the trainees so that they can prepare in a colourful and attractive environment.

Being a Bluetooth connection no need of stubby wires and constant USB connection from the system trainee can perform operations from distance.

**Tools and Environment**

**Hardware**

* Arduino Uno
* Bluetooth(HC-05)
* 16X2 Character LCD
* Laptop Bluetooth
* Dotted PCB
* Buttons
* Oscillator
* Resistors
* Capacitors
* Buzzer
* LEDs
* Connecting Wires

**Software**

* MATLAB 2013a
* Instrument Control Toolbox
* Arduino IDE 1.6.9.0
* Proteus 8.1

**Data Flow Diagram**

MATLAB HomeScreen

Connection between ARDUINO&MATLAB

Read the Morse Code text -> Process the text ->Display the image of the character ->Transmit the character to ARDUINO -> Display the character and buzz the codes

Transmit the Morse Code through ARDUINO-> Read the Morse Code -> Process the Morse Code ->Display the image of the character

Read the text -> Process the text -> Display the image of the character ->Transmit the character to ARDUINO -> Display the character and buzz the codes

Morse Code Transmitter

Alpha Numeric Transmitter

Morse Code Transmitter

MATLAB Receiver

ARDUINO Transmitter

MATLAB Transmitter

ARDUINO Receiver

**Module Details**

**MATLAB**

MATLAB acts as a backbone of this project. All the processing in the project is performed by the MATLAB right from establishing connection, checking the character transmitted to the processing of the images.

It establishes connection with the ARDUINO microcontroller with the help of the Bluetooth of the Laptop and connects with the HC-05 Bluetooth module connected to the microcontroller.

It only checks the character or the Morse code entered by the user when it acts as a transmitter and performs the transmission according to it and display the corresponding picture.

When it works as a receiver it keeps a record of the dots and dashes received and when the user presses the submit button it checks with its record for the same and gives us the corresponding code and the pop up image.

**Instrument Control Toolbox**

Instrument Control Toolbox lets you connect MATLAB directly to instruments such as oscilloscopes, function generators, signal analyzers, power supplies, and analytical instruments. The toolbox connects to your instruments via instrument drivers such as IVI and VXI plug & play, or via text-based SCPI commands over commonly used communication protocols such as GPIB, VISA, TCP/IP, and UDP. You can also control and acquire data from your test equipment without writing code.

With Instrument Control Toolbox, you can generate data in MATLAB to send out to an instrument, or read data into MATLAB for analysis and visualization. You can automate tests, verify hardware designs, and build test systems based on LXI, PXI, and AXIe standards.

For remote communication with other computers and devices from MATLAB, the toolbox provides built-in support for TCP/IP, UDP, I2C, SPI, and Bluetooth serial protocols.

In this project **Instrument Control Toolbox** is used to establish a **Bluetooth** connection between the MATLAB and the microcontroller to communicate wirelessly between ARDUINO UNO and MATLAB.

**MATLAB GUI**

**Graphical User Interface**(GUI) of MATLAB is used to give graphical environment which is attractive to the user and also helps in establishing a user friendly medium.

In this project everything is happening in the GUI three GUIs have been clubbed so navigation is easy for the user to visit one GUI from the other.

The three GUIs are

* **homescreen** which is the first screen to come when we execute the program and the user has the choice weather he wants to transmit the information or receive the information
* **transmitter** the second GUI which comes up when the user decides to transmit the information.

It contains two types of transmission

* **Morse Code** where the user supplies the Morse Code and gets the corresponding alphanumeric character for the same
* **AlphaNumeric** where the user supplies the alphanumeric character and gets the Morse Code for the same
* **receiver** the third GUI which comes up when the user decides to recieve the information

**Arduino Uno**

It is the microcontroller used in the program to interface the Bluetooth module and other electronic components with the MATLAB.

It has the connections with the 16X2 Character LCD, Buzzer, LED, Transmitter buttons and serves as there controller.

It only checks the data received or transmitted from or to the MATLAB and instructs the different modules for their required tasks

**Bluetooth HC-05**

The Bluetooth HC-05 module is used to setup the Bluetooth connection between the MATLAB and the ARDUINO UNO. The six pin set has only 4 connections to be made

VCC connected to +5V

GND connected to ground

RX connected to the TX of ARDUINO UNO

TX connected to the RX of ARDUINO UNO

When paired by the password it can send and receive the information

**LCD**

The 16X2 Character LCD connected to the PORTB of the ARDUINO UNO is used as a display for the user at the Embedded end.

* It shows the transmitted dots and dashes when ARDUINO UNO works as a **transmitter**
* It shows the receiver character when ARDUINO UNO works as a receiver

**Power Supply**

12V 1amp DC adapter is used to supply power to Arduino Board whose voltage is regulated by using Voltage Controller IC7805.

**Buzzer**

Buzzer beeps on the bit transmitted its delay vary depending on weather the bit is a dot or a dash and the listener can check for the character transmitted by keeping a record of the beeps.

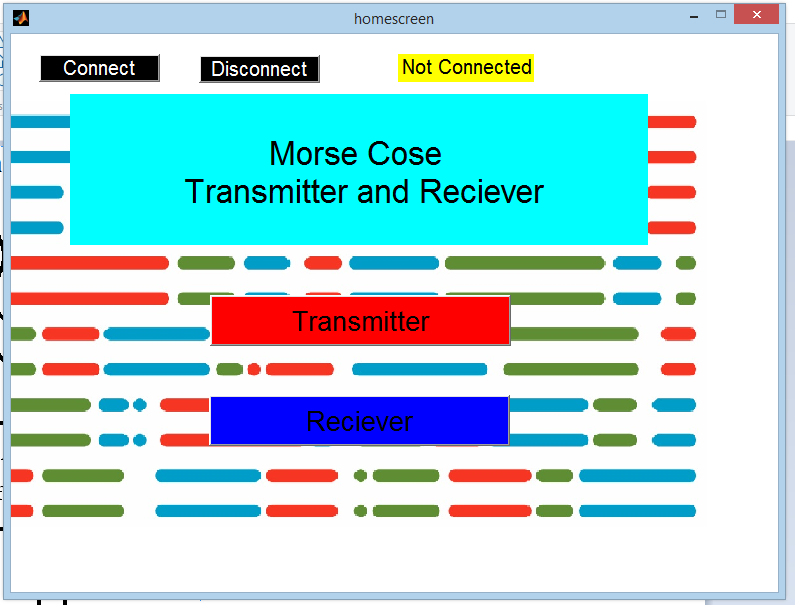
**LED**

LED blinks on the bit transmitted its delay vary depending on weather the bit is a dot or a dashand the listener can check for the character transmitted by keeping a record of the blinks.

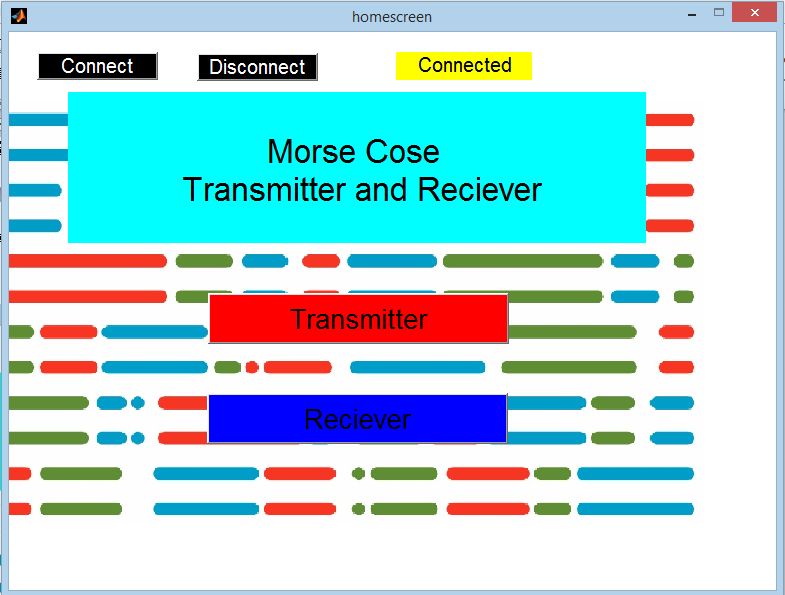
**Dotted PCB**

Dotted PCB or through hole has been used in my project to make connections and the PCB for the circuit to mount and connect the components

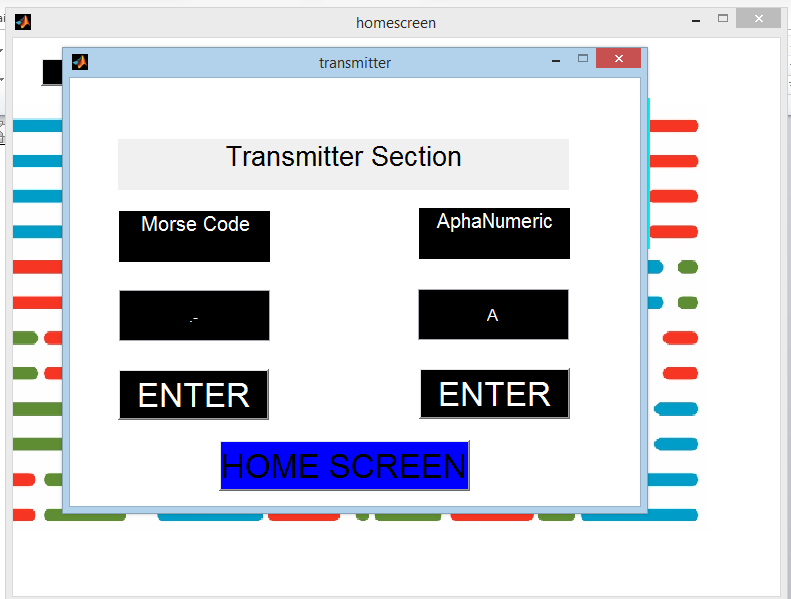
**User Interface Design**

****

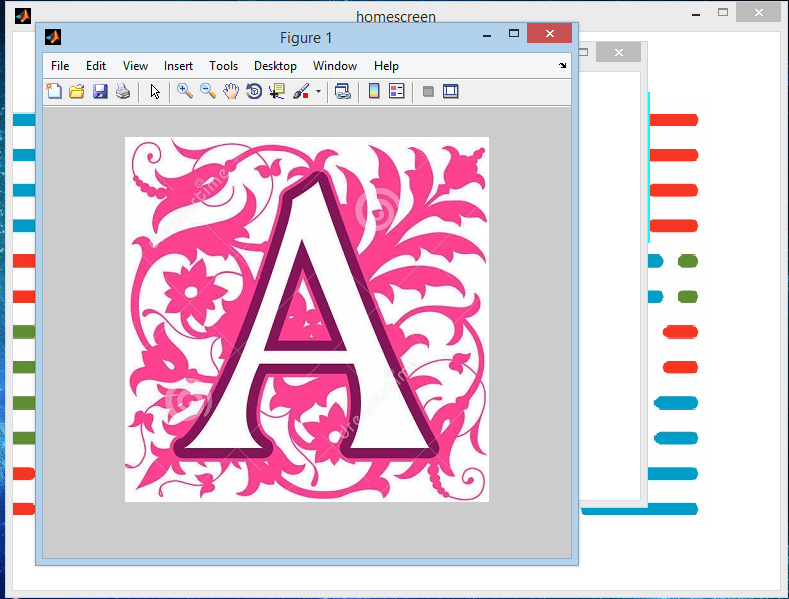
Home screen of MATLAB project

****

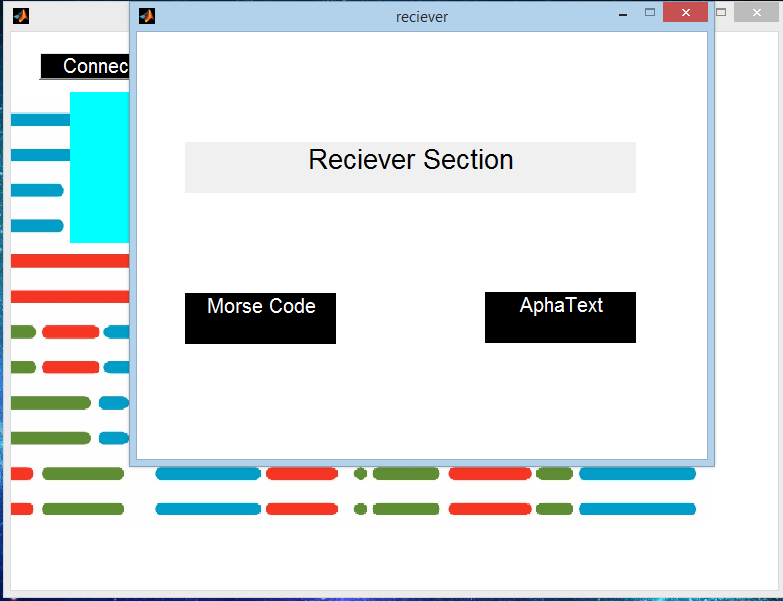
When connection has been established

****

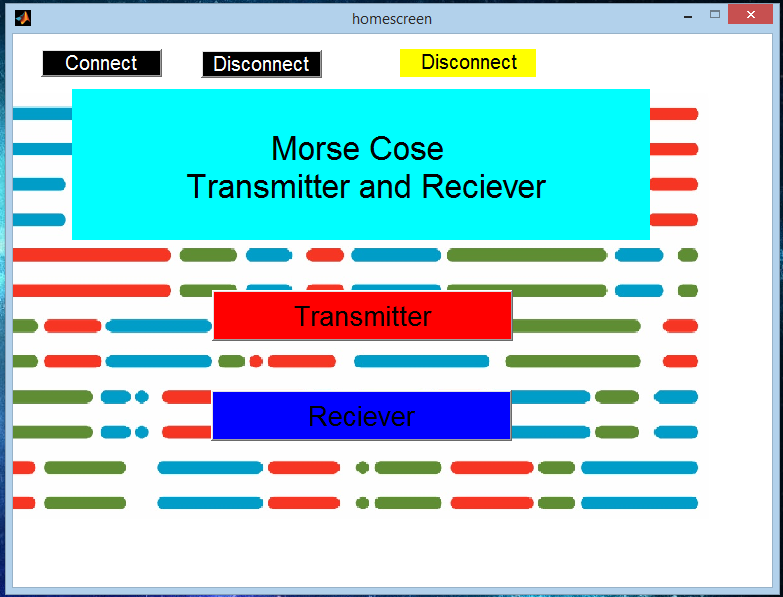
Transmitter section when A has been transmitted

****

‘A’ letter popped up on screen

****

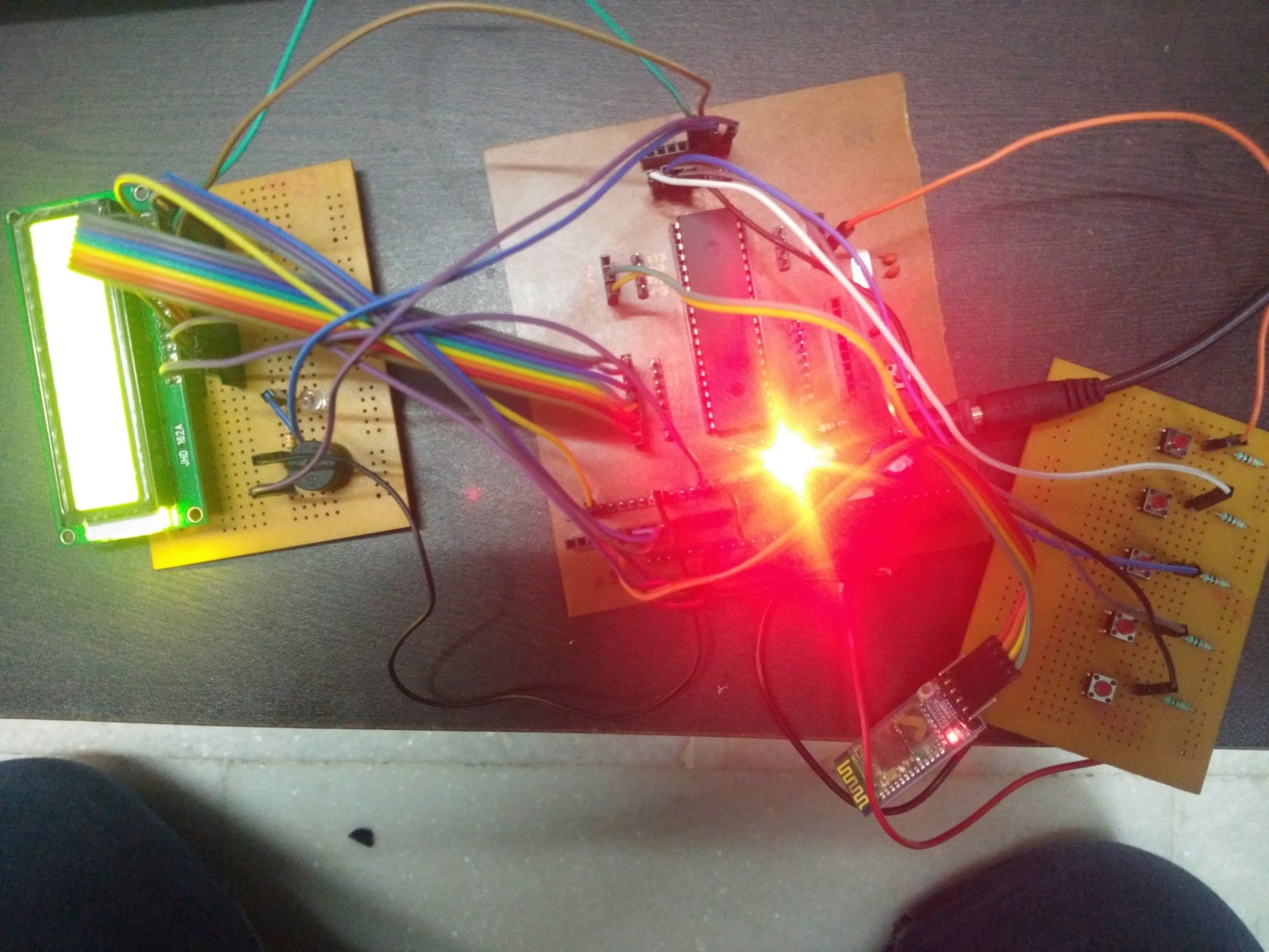
Reciever Section of the MATLAB



When the connection has been disconnected**Circuit Diagram**

****

**Hardware Design**

****

**Testing**

* Individual **GUI** of **MATLAB** were tested for the first time to check for user interface
* Control transfer in various GUI
* PCB board and tracks were checked for short circuiting
* **Power off Testing** was performed after soldering of components to check the solder
* **Power on Testing** was performed to check supply and voltages
* Small programs were burned to check for individual components testing
* Full Program was burned and tested

**Problems Faced**

* Control transfer in various GUI
* Scope of MATLB variables
* Setting the registers value for transmission
* Bluetooth transmission

**Future Scope**

* I plan to develop a **mobile app** for this so that person can use their mobile phones to send the Morse code as well as receive the transmitted characters.
* To increase the range I am planning to use **Wi-Fi** module instead of **Bluetooth**.
* Graphic LCD can be used instead of Character LCD to make it look attractive.

**Bibliography**

* [www.en.wikipedia.org](http://www.en.wikipedia.org)
* [www.google.co.in](http://www.google.co.in)
* www.mathworks.com
* LCD datasheet
* Arduino examples
* MATLAB Help