WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Design and Construction of Arduino-Based Experimental Training Kit

Isizoh A.N.[1], Ejimofor I.A.[2],Alagbu E.E.[3], Obianyo O.R.[4], Ebih U.J.[5]

1,3,5: Dept. of Electronic and Computer Engineering, Nnamdi Azikiwe University, Awka, Nigeria,
2,4: Dept. of Computer Engineering, Madonna University of Nigeria, Akpugo Campus.

ABSTRACT

This paper analyzes the design and construction of arduino-based experimental training kit. The research was motivated by lack of practical training on the use of arduino board which is now replacing the conventional microcontrollers (eg. AT89C51, AT89C52, etc). The system uses Arduino board which contains ATmega328 microcontroller, bread board, Liquid Crystal Display (LCD), Light Dependent Resistor (LDR), GSM module, keypad, DC to DC converter, relays, switches, and several electronic components. The above devices were interfaced to the arduino board, and a program written in arduino programming language was developed and burnt inside the ATmega328 microcontroller. This program controls the operation of the system whenever a particular experiment (demonstration) is required. Displays on LCD and LEDs were done for testing, and the system worked perfectly well.

KEYWORDS: Arduino, microcontroller, keypad, Liquid Crystal Display, relay, Crystal Oscillator.

I. INTRODUCTION

1.1 Background of Study

Arduino is an open source programmable circuit board that can be integrated into a wide variety of projects both simple and complex. This board contains a microcontroller which is able to be programmed to sense and control objects in the physical world. Most Arduino boards consist of an Atmel 8-bit AVR microcontroller (ATmega8, ATmega168, ATmega328, ATmega1280, or ATmega2560) with varying amounts of flash memory, pins, and features. The popular Arduino Uno uses the ATmega328 microcontroller. The 32-bit Arduino Due, based on the Atmel SAM3X8E was introduced in 2012 [1].

The boards use single or double-row pins or female headers that facilitate connections for programming and incorporation into other circuits. By responding to sensors and inputs, the Arduino is able to interact with a large array of outputs such as LEDs, motors and displays. Because of its flexibility and low cost, Arduino has become a very popular choice for makers and makerspaces looking to create interactive hardware projects. Arduino was introduced back in 2005 in Italy by Massimo Banzi as a way for non-Engineers to have access to a low cost, simple tool for creating hardware projects. Since the board is open-source, it is released under a Creative Commons license which allows anyone to produce their own board. If you search the web, you will find there are hundreds of Arduino compatible clones and variations available but the only official boards have Arduino in its name [2].

Arduino microcontrollers are pre-programmed with a boot loader that simplifies uploading of programs to the on-chip flash memory. The default bootloader of the Arduino Uno is the Optiboot bootloader. Boards are loaded with program code via a serial connection to another computer. Some serial Arduino boards contain a level shifter circuit to convert between RS-232 logic levels and transistor–transistor logic (TTL) level signals. Current Arduino boards are programmed via Universal Serial Bus (USB), implemented using USB-to-serial adapter chips such as the 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 Arduino Uno, use a detachable USB-to-serial adapter board or cable, Bluetooth or other methods. When used with traditional microcontroller tools, instead of the Arduino IDE, standard AVR in-system programming (ISP) programming is used [3].

II. LITERATURE SURVEY

2.1 Review of Past Related Works

Isizoh at al. [13] developed an Intelligent System Design of Microcontroller-based Real-Time Process Control Trainer, which was published in International Journal of Recent Trends in Engineering & Research (IJRTER), Volume 05, Issue 04 of April 2019. The work did not discuss arduino board, but only conventional microcontrollers.

Muhammad Ali [14] designed a low cost digital trainer for students experimental programmes. The module was developed to assist the student undertaken digital logic and basic electronics courses to perform and replicate the digital experimental exercise at home using a technology called PortBuffer. The system was inexpensive and portable, but the module is a software simulation based functioning; its operations depend on computer application and could not perform SLC and other tasks.

An effort on microcontroller-based real time emulator for logic gate and structured logic devices was presented by Floyd T.F [15]. The aim of the

author was to find a way out to make the microcontroller become a convenient substitute for any digital device that is readily unavailable for use in the logic design. A system emulator was suggested to take the advantages of digital logic design which can lead to offer of low cost laboratory equipment for efficient skills transfer in the educational systems. The design for SLC is more difficult task than CLC because of the feedback part and redundant states in sequential logic circuits.

A module-level Evolvable Hardware (EHW) approach to design synchronous sequential logic circuits was presented by Kopetz [16]. Genetic Algorithm (GA) was used to minimize the circuit complexity, number of logic gates, wires attached, and obtain near-optimal state assignment. Therefore, sequence detectors, modulo counters, and ISCAS'89 circuit are used as the proof for the simulation evolutionary design approach.

Also, genetic algorithm (GA) was used to design combinational logic circuits (CLCs) in the optimization of combinational logic circuits through decomposition of truth tables and evolution of sub-circuits presented in [17]. The goal of the author is to minimize the number of logic elements in the circuit, by proposing a new coding for circuits using a multiplexer (MUX) at the output of the circuit.

Predko [18] in his work, made enormous contribution in the area of Microcontroller and logic circuit as reviewed in his research work.

Thus, the proposed system and illustrations attempt to put into the practices all the techniques, proof from the reviewed paper about the circuit optimization, circuit complexity and minimization. This work makes apparent practicality by developed a low-cost digital logic training module for student laboratory experiment. The approach of "learn-while doing" on the digital electronics, logic circuit design and embedded system projects will help students connect their reasoning from theoretical background to a practical development.

2.2 Theoretical Frameworks of Some Arduino Types

Arduino boards are of so many different families. But going forward, the research will focus on Arduino Uno as our standard device. Indeed, this is by far the most used of the Arduino boards, but the boards are all programmed using the same language and largely have the same connections to the outside world, so you can easily use a different board. The following are the most popular Arduino boards:

2.2.1 Arduino Uno

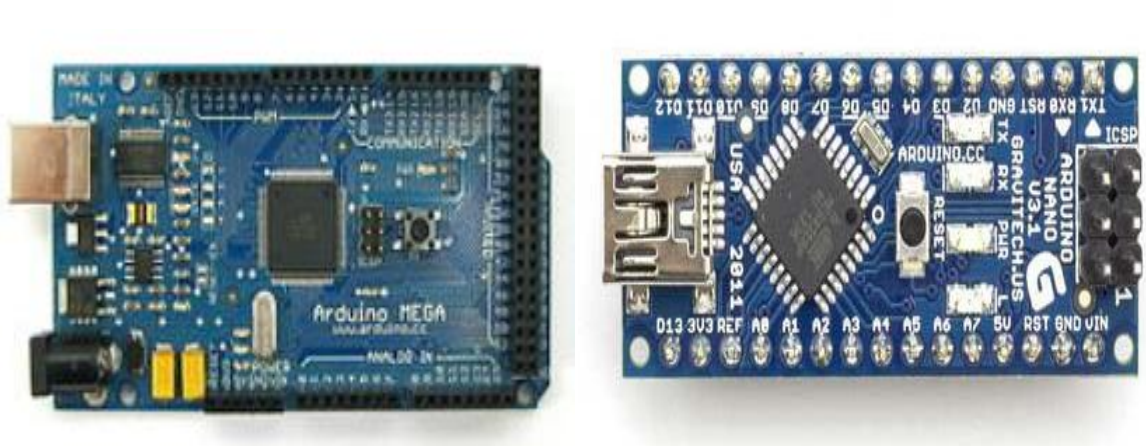
The Arduino Uno is the latest incarnation of the most popular series of Arduino boards. The series includes the Diecimila (Italian for 10,000) and the Duemilanove (Italian for 2011). These older boards look very similar to the Arduino Uno. They both have the same connectors and a USB socket and are generally compatible with each other [4]. The most significant difference between the Uno and the earlier boards is that the Uno uses a different USB chip. This does not affect how you use the board, but it does make installation of the Arduino software easier and allows higher speeds of communication with the computer. The Uno can also supply more current on its 3.3V supply and always comes equipped with the ATmega328. The earlier boards will have either an ATmega328 or ATmega168. The ATmega328 has more memory, but unless you are creating a large sketch, this will make no difference.

2.2.2 Arduino Mega

The Arduino Mega shown in Figure 1, is the muscle car of Arduino boards. It boasts a huge collection of input output ports, but cleverly adds these as extra connectors at one end of the board, allowing it to remain pin-compatible with the Arduino Uno and all the shields available for Arduino. It uses a processor with more input output pins, the ATmega1280, which is a surface mount chip that is fixed permanently to the board [5]. So, unlike with the Uno and similar boards, you cannot replace the processor if you accidentally damage it. The extra connectors are arranged at the end of the board. Extra features provided by the Mega include the following: 54 input/output pins, 128KB of flash memory for storing sketches (programs) and fixed data (compared to the Uno's 32KB) • 8KB of RAM and 4KB of EEPROM.

2.2.3 Arduino Nano

The Arduino Nano of Figure 1 is a very useful device for use with a solderless breadboard. It is a small, complete, and breadboard-friendly board based on the ATmega328P released in 2008. It offers the same connectivity and specs of the Arduino Uno board in a smaller form factor. If you fit pins to it, it can just plug into the breadboard as if it were a chip. The downside of the Nano is that because it is so much smaller than an Uno, it cannot accept Uno-sized shields [6].



Arduino Mega Arduino Nano

Figure 1: Arduino Mega and Arduino Nano

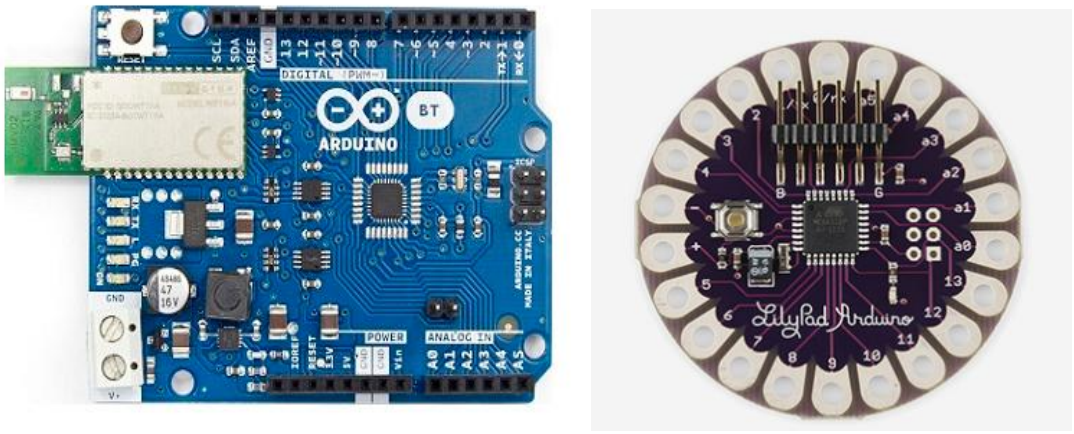
2.2.4 Arduino Bluetooth

The Arduino Bluetooth show in Figure 2 is an interesting microcontroller device as it includes Bluetooth hardware in place of the USB connector. This allows the device to even be programmed wirelessly. It is a microcontroller board that was originally based on the ATmega168, but now is supplied with the 328P and the Bluegiga WT11 Bluetooth module. It supports wireless serial communication over Bluetooth (but is not compatible with Bluetooth headsets or other audio devices). It has 14 digital input/output pins (of which 6 can be used as PWM outputs and one can be used to reset the WT11 module), 6 analog inputs, a 16 MHz crystal oscillator, screw terminals for power, an ICSP header, and a reset button. It contains everything needed to support the microcontroller and can be programmed wirelessly over the Bluetooth connection The Arduino Bluetooth is not a cheap board,

and it is often cheaper to attach a third party Bluetooth module to a regular Arduino Uno [7].

2.2.5 Arduino Lilypad

The Lilypad also shown in Figure 2 is a tiny, thin Arduino board that can be stitched into clothing for applications that have become known as wearable computing. The Main Board is based on the ATmega168V (the low-power version of the ATmega168) or the ATmega328V. The Lilypad does not have a USB connection, and you must use a separate adaptor to program it. It has an exceptionally beautiful design.



Arduino Bluetooth Arduino Lilypad

Figure 2: Arduino Bluetooth and Arduino Lilypad

The above listed families are the most popular families of Arduino boards. But this research will closely look at Arduino Uno since it will be used for this work.

2.3 Arduino Uno

The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. *Its pin diagram* is shown in Figure 3. The board has 14 digital I/O pins (out of which 6 is capable of PWM), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts, Uses 16 MHz quartz crystal oscillator, a power jack, an ICSP header and RST button [8].

The word "uno" means "one" in Italian and was chosen to mark the initial release of Arduino Software. The Uno board is the first in a series of USB-based Arduino boards; it and version 1.0 of the Arduino IDE were the reference versions of Arduino, which have now evolved to newer releases. The ATmega328 on the board comes preprogrammed with a bootloader that allows uploading new code to it without the use of an external hardware programmer [9].

2.3.1 Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board [10]. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A Software Serial library allows for serial communication on any of the Uno's digital pins. The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus [11].

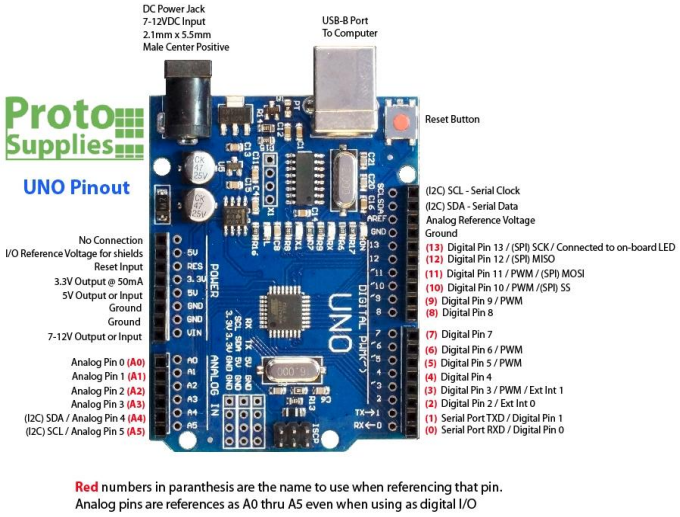
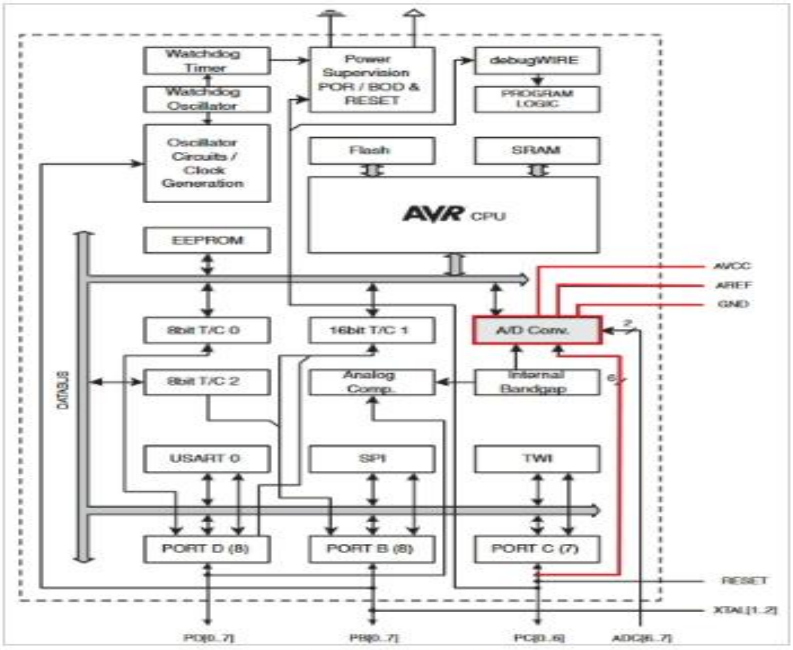


Figure 3: The Arduino Uno Board Layout

The ATmega328 block diagram is shown in the diagram of Figure 4.



ATmega328 block diagram.

Figure 4: Atmega328 Block Diagram.

2.3.2 Advantages of Using Arduino Uno

Arduino Uno is the most standard board available and probably the best choice for a beginner. It is a good all-purpose board that has enough features for a beginner to get started with. Some of its better features are:

1. Its biggest advantage is that we connect the board to the computer via a USB cable which does a dual purpose of supplying power and acting as a Serial port to interface the Arduino and the computer.
2. It can also be powered by a 9V-12V AC to DC adapter.
3. The ATmega328 chip can be newly bought, removed and replaced if damaged which is not possible with other versions.
4. The board operates at 5V throughout, i.e. digital pins output or read 5v and analog pins read in the range 0-5V.
5. Lots of example code and projects are done using Arduino Uno, hence will get good support.
6. The Uno features 14 Digital I/O pins and 6 Analog I/O pins.
7. Lot of extra Add-on hardware is built for Uno. Special hardware is available for Internet, Bluetooth, Motor control etc.
8. It is the cheapest board (Rs.1600 only) with all these features.

2.4 Arduino Uno Environment and Platform

2.4.1 The Arduino IDE

The Arduino IDE, shown in Figure 5, is incredibly minimalistic, yet it provides a near-complete environment for most Arduino-based projects. The top menu bar has the standard options, including “File” (new, load save, etc.), “Edit” (font, copy, paste, etc.), “Sketch” (for compiling and programming), “Tools” (useful options for testing projects), and “Help”. The middle section of the IDE is a simple text editor that where you can enter the program code. The bottom section of the IDE is dedicated to an output window that is used to see the status of the compilation, how much memory has been used, any errors that were found in the program, and various other useful messages.

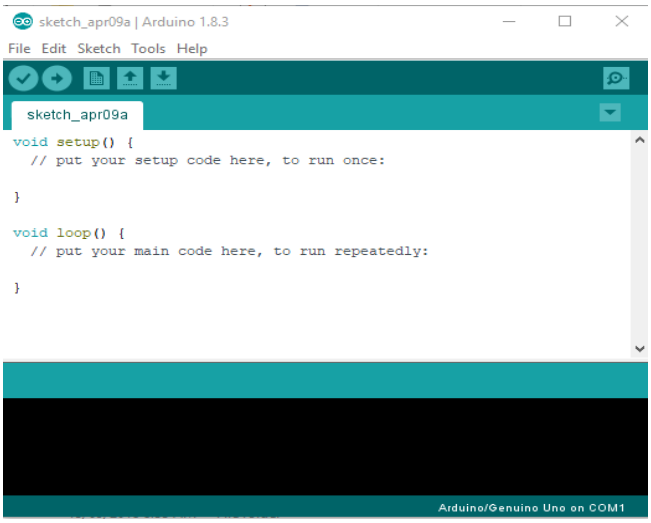


Figure 5: The Arduino IDE in its Default State

The Arduino development environment main program has two functions already defined. The setup and loop function. The setup function is where the code that should run once is written. Setup codes like defining the input and output pins, initializations etc. are written inside the setup function. The loop function should contain the codes that are meant to run continuously in the project. Examples of such codes could be continuous reading of sensor inputs to know when the state changes, displaying outputs in the selected display device etc.
Projects made using the Arduino are called sketches, and such sketches are usually written in a cut-down version of C++ (a number of C++ features are not included). Because programming a microcontroller is somewhat different from programming a computer, there are a number of device-specific libraries (e.g., changing pin modes, output data on pins, reading analog values, and timers). This sometimes confuses users who think Arduino is programmed in an "Arduino language." However, the Arduino is, in fact, programmed in C++. It just uses unique libraries for the device [12].

2.4.2 Steps For Programming Arduino Board

Step 1
First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer.
Step 2 − Download Arduino IDE Software.
You can get different versions of Arduino IDE from the Download page on the Arduino Official website. You must select your software, which is compatible with your operating system (Windows, IOS, or Linux). After your file download is complete, unzip the file.
Step 3 − Power up your board.
The Arduino Uno, Mega, Duemilanove and Arduino Nano automatically draw power from either, the USB connection to the computer or an external power supply. If you are using an Arduino Diecimila, you have to make sure that the board is configured to draw power from the USB connection. The power source is selected with a jumper, a small piece of plastic that fits onto two of the three pins between the USB and power jacks. Check that it is on the two pins closest to the USB port. Connect the Arduino board to your computer using the USB cable. The green power LED (labeled PWR) should glow.
Step 4 − Launch Arduino IDE.
After your Arduino IDE software is downloaded, you need to unzip the folder. Inside the folder, you can find the application icon with an infinity label (application.exe). Double-click the icon to start the IDE.
Step 5 − Open your first project.
Once the software starts, you have two options −
Create a new project.
Open an existing project example.
To create a new project, select File → New
Step 6 − Select your Arduino board.
To avoid any error while uploading your program to the board, you must select the correct Arduino board name, which matches with the board connected to your computer.
Go to Tools → Board and select your board.
Here, we have selected Arduino Uno board according to our tutorial, but you must select the name matching the board that you are using.
Step 7 − Select your serial port.
Select the serial device of the Arduino board. Go to Tools → Serial Port menu. This is likely to be COM3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu, the entry that disappears should be of the Arduino board. Reconnect the board and select that serial port.
Step 8 − Upload the program to your board.
Before explaining how we can upload our program to the board, we must demonstrate the function of each symbol appearing in the Arduino IDE toolbar.
Now, simply click the "Upload" button in the environment. Wait a few seconds; you will see the RX and TX LEDs on the board, flashing. If the upload is successful, the message "Done uploading" will appear in the status bar

2.4.3 Steps For Burning Programs Into the Arduino Board

1. Connect your Arduino using the USB cable. The square end of the USB cable connects to your Arduino and the flat end connects to a USB port on your computer.
2. Choose Tools→Board→Arduino Uno to find your board in the Arduino menu. You can also find all boards through this menu, such as the Arduino MEGA 2560 and Arduino Leonardo.
3. Choose the correct serial port for your board. You find a list of all the available serial ports by choosing Tools→Serial Port→ comX or /dev/tty.usbmodemXXXXX. X marks a sequentially or randomly assigned number. In Windows, if you have just connected your Arduino, the COM port will normally be the highest number, such as com 3 or com 15.

Many devices can be listed on the COM port list, and if you plug in multiple Arduinos, each one will be assigned a new number. On Mac OS X, the /dev/tty.usbmodem number will be randomly assigned and can vary in length, such as /dev/tty.usbmodem1421 or /dev/tty.usbmodem262471. Unless you have another Arduino connected, it should be the only one visible.

III. SYSTEM DESIGN AND ANALYSIS

3.1 Block Diagram of the Arduino Experimental Training Kit

The block diagram of an Arduino Experimental Training kit is shown in figure 6. It shows all the block units of the system, which comprises of : the arduino uno (control unit), power supply, button unit, keypad unit, LED unit, Op-amp unit, sound unit, switching unit, sensor unit, SMS unit, and display unit.

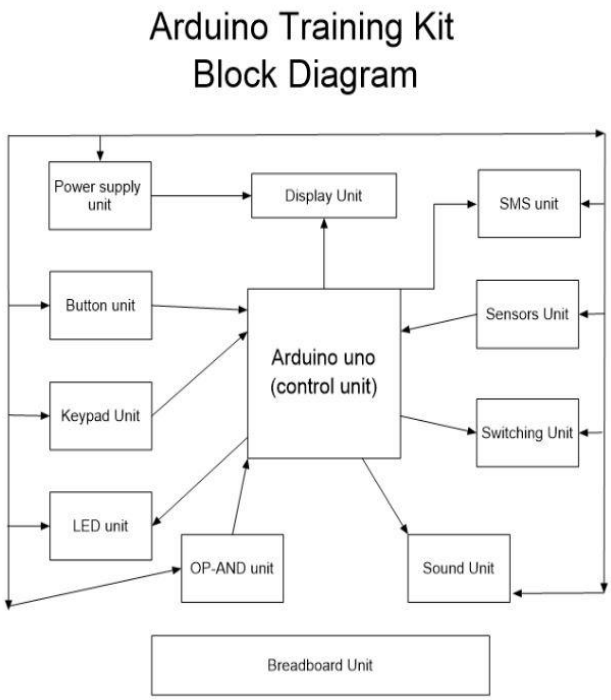


Figure 6: Block Diagram of Arduino Experimental Training Kit

3.2 Buttons Interfaced to Arduino for LED Control

In an initial design, let us consider a very simple design where a button is used to turn on and off some group of LEDs. When the button is pressed, the green LEDs turn on, while others are turned off. When the button is released, the LEDs reverse their state (those that are ON will turn OFF, while those that are OFF will turn ON). The block diagram of the Button/LED interface to Arduino board is as shown in Figure 7.

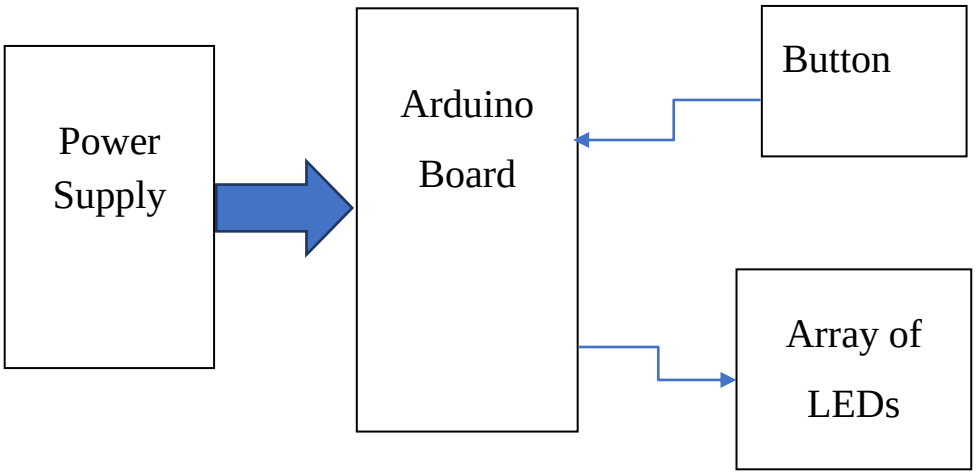


Figure 7: Block diagram of a Buttoned Controlled LED Arduino Training Kit

The block diagram in Figure 7 depicts a simple button controlled array of LEDs. The power supply to Arduino could be either from the DC power jack (7 - 12V), the USB connector (5V), or the V_{in} pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage the board. The button will serve as an input source to the ATmega328 controller, hence the pin where it is connected will be defined as an input pin; while the pins where the array of LEDs are connected will be defined as output port.

The circuit diagram for the button controlled LED interface is shown in Figure 8. The button is single push single throw button which is used to send signal to the Arduino board. Initially when the button is not pressed down, the pin 13 of the Arduino controller will be set to ground by the connection , but as soon as the button is pressed, the 5V source get connected to the Arduino and the controller would sense that the button is pressed. The corresponding code is now written to turn off or on the LEDs depending on the state of pin 13.

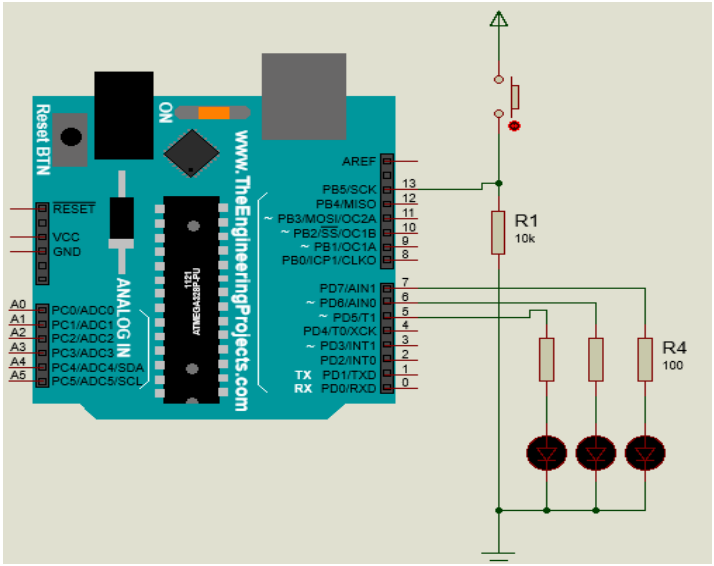


Figure 8: Circuit Diagram of the Button Controlled LED Display

The algorithm flowchart for the described system is shown in Figure 9.

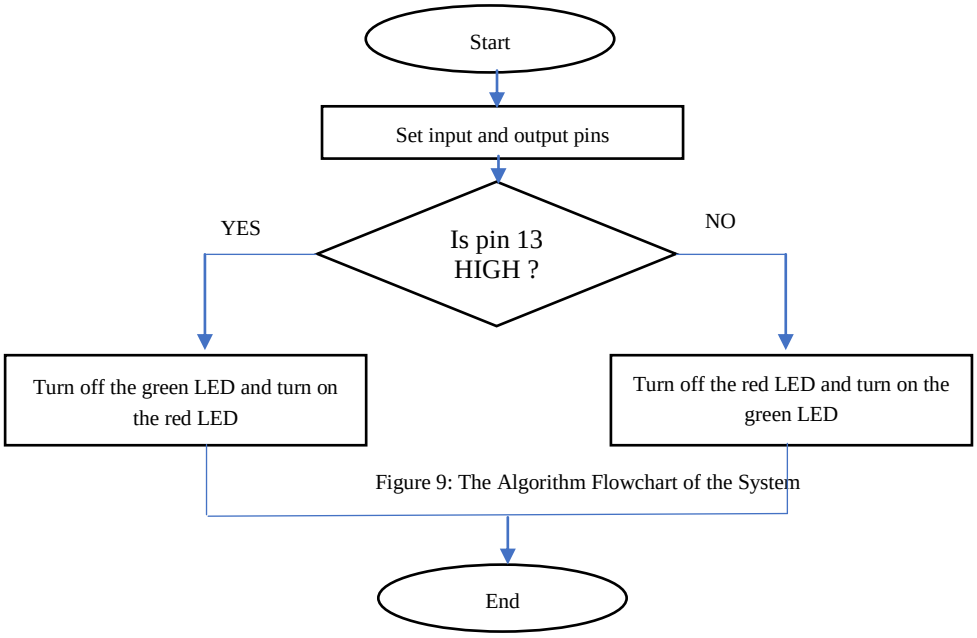
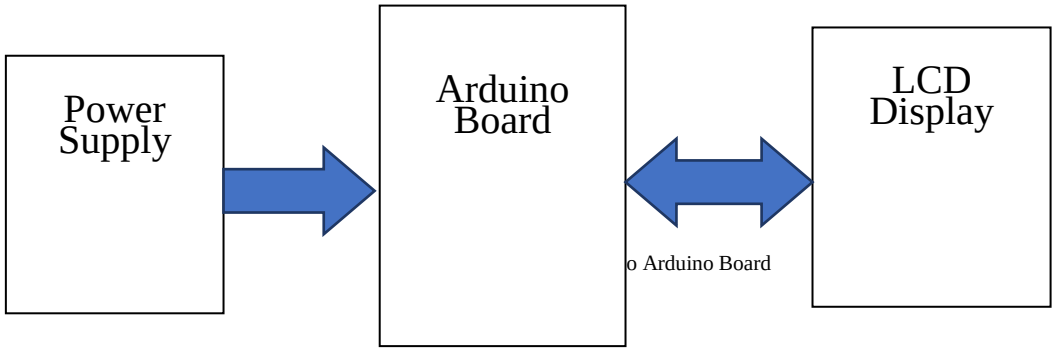


Figure 9: The Algorithm Flowchart of the System

3.3 LCD Interfaced to Arduino

Arduino board can be interfaced with several displays including seven segment, LCD and also to Dot matrix displays using shift registers [20]. In this project, we will be interfacing an LCD display to Arduino Uno. We will be using the following components Arduino Uno, 16x2 LCD and 5k potentiometer. The Potentiometer will be used for contrast adjustment of the LCD. The block diagram for this interface is shown in Figure 10.



The above figure depicts the interfacing of LCD to Arduino. Observe that there is bidirectional arrow between the LCD and the Arduino. This is because the Arduino board can write to the LCD display and as well read from it. The circuit diagram is as shown in Figure 11.

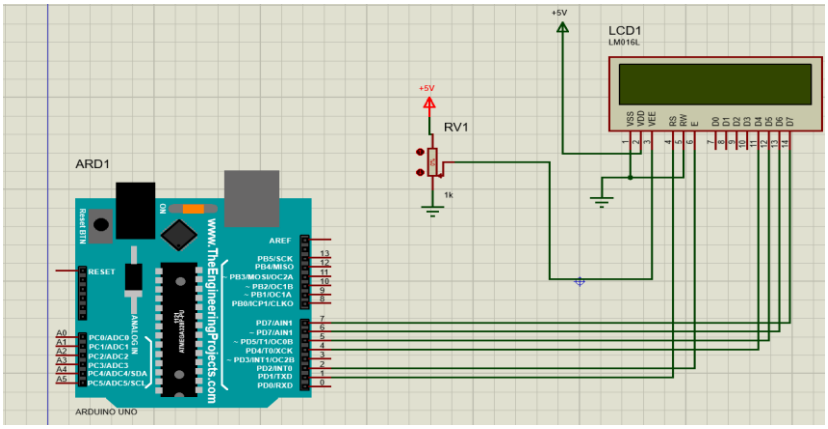


Figure 11: LCD Interfaced to Arduino Uno

3.4 Software Design of the System

The program for the system of operation was developed in Arduino Programming language. The program sample is too long to be included here. It was developed from the algorithm.

3.4.1 Pseudocode for the Experimental Kit

(Button Experiment or LED Manipulation)

1. Start the program
2. System Initialization
3. Select button (1 or 2 or 3 or 4) for experiment
4. If button 1 is selected, perform swap dance in 4's
5. If button 2 is selected, perform swap dance in 2's
6. If button 3 is selected, perform LED blinking one by one
7. If button 3 is selected, perform LED blinking in 4's
8. Keep checking for button press
9. End the program

3.4.2 The System Flowchart

The flowchart for the system of operation of the buttons controlling the LEDs is shown in Figure 12.

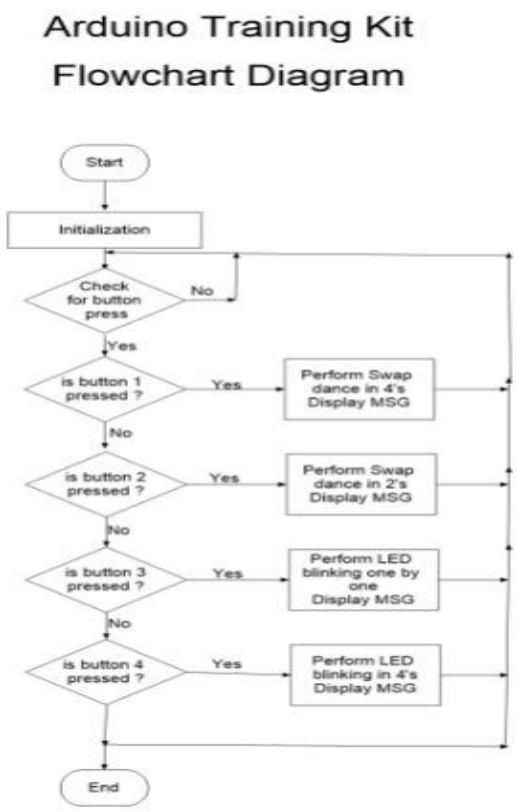


Figure 12: System Flowchart for the Operation of the Buttons controlled LEDs

The circuit diagram of the system was developed using Referencing technique in a Proteus environment as shown in Figure 13, and that is why there are no wire connections.

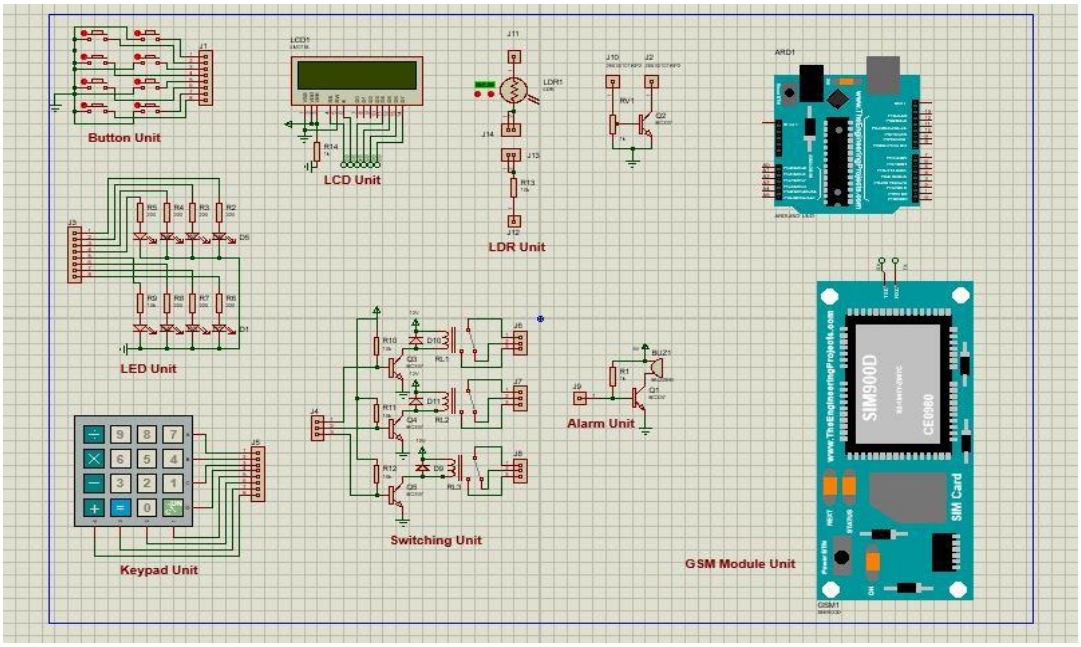


Figure 13: The Circuit Diagram of the System

IV. RESULTS AND DISCUSSSION

4.1 Principle of Operation of Arduino Experimental Kit

The Arduino experimental kit uses an arduino Uno which is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

The Experimental Kit is a beginner friendly learning kit which beginners can use to carry out electronic project before implementing in real-time environment. It has different units for different operations and experiments which a user can use. It has 12 units namely:

1. **Power Supply Unit**: This unit is where power is been supplied to the whole circuit, this unit is as important as every other one because without it power won't be supplied to the circuitry
2. **Display Unit**: At this unit an LCD(Liquid crystal display) is used to read out the data from the arduino. This helps to display the values, signals from the kit
3. **SMS Unit**: This is where the SIM module (SIM 900A) is used for experiment like Intruder alert system where a user is alerted for potential security threat. A SIM module is used for this experiment.
4. **Button Unit**: The button unit allows a user to carry out experiment such as LED MANIPULATION where a button is used to control the behavior of an LED.
5. **Sensor Unit**: At this unit a sensor (flame sensor) is used to carry out experiment for fire detection system.
6. **Keypad Unit**: Experiment that involves calculation like Addition, subtraction is done here.
7. **Switching Unit**: This is the unit where LDR, light dependent resistor is used to carry out experiment like dark activated circuit where low resistance triggers the circuit to switch on. It can also serve as a sensor.
8. **LED Unit**: This unit serves as indicator and it is used for carrying out experiment like LED MANIPULATION or BUTTON EXPERIMENT. The kit comes with several LEDs in various colors (red, yellow, green and white) as well as a single RGB LED. The RGB LED features 4 pins (one for each color and one common cathode - or ground), so you can experiment with color mixing, or create displays of random color. It's quite cool to play with once you dive into code
9. **Sound Unit** : The sound unit helps to output sound signal from the experimental kit with the help of the buzzer as a sound device
10. **Breadboard**: The breadboard serves a connecting point where other component or devices can be connected to the kit
11. **OP-AMP**: This is an Operational amplifier that amplifies voltage that is sent into the kit
12. **Arduino Uno (Control Unit)**: This unit serves as the brain box and the major contributing unit of the system. This is where programs that the system will use is written. The success of the system is dependent on this unit.

4.2 Result of the Operation of the System

When the unit modules were interconnected to form a system, it was tested as outlined above. There was no fault and the system worked as it was programmed. The constructed prototype of the system is shown in Figure 14 after integration and testing.

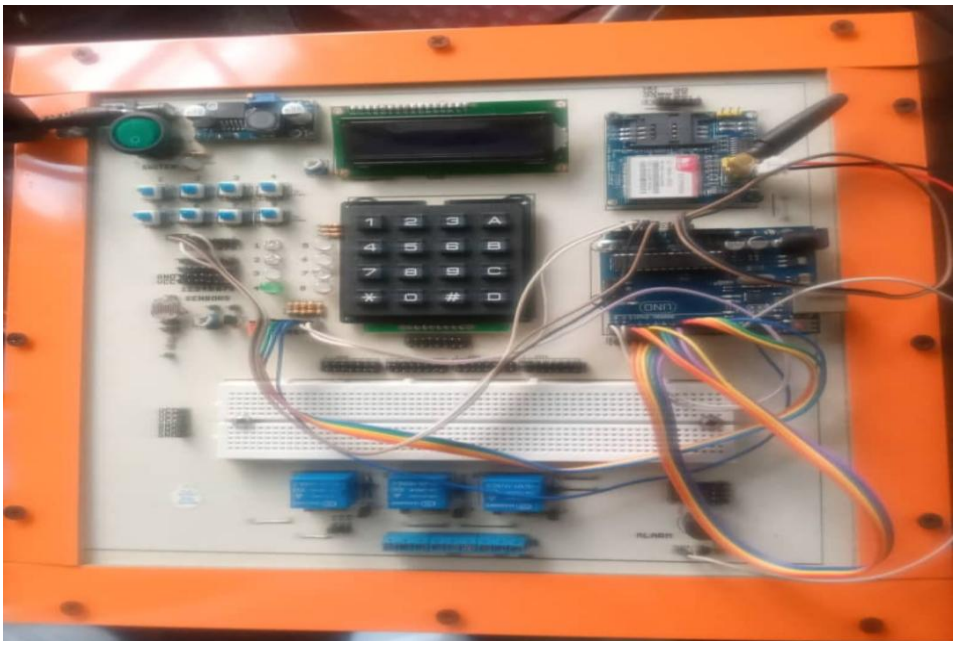


Figure 14: The Constructed Prototype

4.3 Testing

Testing is necessary in determining if the circuit meets the desired purpose for which it was constructed as well as for optimization. The circuit of this system was tested module by module as it is being integrated. These testing process includes:

- Each component was checked to see if they are ok before they were used in the circuitry.
- Potentiometer was adjusted to the proper range.
- The power supply was properly checked to see if it falls within the tolerance calculated theoretically so as to avoid damage to the components.
- The components which were not functioning properly were changed as soon as possible to avoid other components being affected by the damaged.
- Testing equipment was in proper range of the output measured at any point of the circuit, or component so as to avoid wrong readings.

4.4 Problems Encountered

Apart from the normal circuit faults and components being damaged by wrong connections, the major problem was developing an instruction or code that will implement an "IF-Else" condition in the program; but it was later resolved.

Another problem encountered was how to stabilize the output as it reaches a set condition. But this problem too was later overcome by adding a delay.

4.5 Precautions Taken

4.5.1 During Soldering of Components into Vero board

- The bit of soldering iron was kept clean with the help of a file from time to time.
- The solder wires are of smaller thickness.
- Extra solder was not used because it may cause a short circuit in the conductive path.
- The components were not overheated during soldering.
- The leads of the components were cleaned with a sand paper before soldering.
- The bit of the soldering iron was cleaned properly before soldering.
- The joint was heated up to the required temperature at which the solder melted before coming around the joint. The joint was not disturbed before the solder set.

4.5.2 During Using the Power Supply

- Switches were used in the circuit.
- Only insulated wires were used.
- Power supply is switched off, when it is not required.
- Any fault in the circuit was repaired before connecting power supply.

V. CONCLUSION AND RECOMMENDATION

5.1 Conclusion

In conclusion, Arduino-Based Experimental Training Kit is easy to use kit which beginners can lay hands on to carry out different electronic projects. It is very flexible to use. The system has GSM module which makes the system capable of communicating with GSM phones after operation. It has port which allows it to communicates with laptop or other computing devices. The LCD that was interfaced to the system is useful in making the system to be user friendly. It makes the user to know the condition of operation.

The system was however developed with a view of making researchers and students understand how arduino board works. The expected result is that a researcher who develops a control program for arduino control can see the physical or real-time effect. This will greatly improve the zeal and encouragement one needs to improve on the research.

5.2 Recommendations

After analyzing the results obtained from testing, the following recommendations were noted for future improvement of this work:

i. A good arduino board should used to improve performance in terms of response time and stability of the system.
ii. Employing the power of modern-day network capabilities to remotely control the system will also be considered as an improvement on the project.
iii. The system can also be extended in functionality by adding other related sensors to it so that it can measure as well as regulate other physical properties like pressure and humidity.
iv. Good switches should always be used to avoid constant damages and replacements

REFERENCES

[1] "Microcontroller Maniacs Rejoice: Arduino finally releases the 32-bit Due," Available Online, Accessed on 27th December, 2021.

[2] "Optiboot Bootloader for Arduino and Atmel AVR", Available Online, Accessed on 29th December, 2021.

[3] S. Monk, "Programming Aduino: Getting Started with Sketches", McGraw-Hill Publishing Companies, New York, 2012, Pp. 23-34.

[4] Sparkfun, "What is Arduino," 4 February 2018, [Online], Available: learn.sparkfun.edu, Accessed on 30th December, 2021.

[5] Princeton, "Introduction to Arduino", Available Online at www.princeton.edu, Accessed on 31st December, 2021.

[6] Dogan Ibrahim, "Microcontoller-Based Temperature Monitoring and Control", Elsevier Science & Technology Books, Maryland USA, 2002, Pp. 63, 87, 107-129.

[7] Douglas V.H., "Microcontrollers and Interfacing: "Programming Hardware" McGraw Hill Inc, New York, 2008, Pp. 270-344.

[8] George Guest, "The March of Civilization", Spectrum Books Limited, Ibadan, Nigeria, 2009, Pp. 150-156.

[9] F.L. Lewis, "Applied Optimal Control and Estimation", Prentice Hall, USA, 2002.

[10] Wakerly John F., "Digital Designs: Principles and Practices", 4th Edition, PHI Learning Private Limited, New Delhi, 2011, Pp 5-6.

[11] Edward Hughes, "Electrical and Electronics Technology", Pearson Education Ltd, India, 2005, PP 634.

[12] Millman – Halkias, "Integrated Electronics", McGraw – Hills Kogakusha, 2003, Pg 112-114.

[13] Anthony Nosike Isizoh, Kelvin Ndubuisi Nnamani, Emmanuel Ugochukwu Chiboka, "Intelligent System Design of Microcontroller-based Real-Time Process Control Trainer", International Journal of Recent Trends in Engineering & Research (IJRTER), Vol. 05, Issue 04, April 2019.

[14] Muhammad Ali Mazidi, "The 8051 Microcontroller and Embedded Systems", Prentitce Hall, New Jersey USA, 2014.

[15] Floyd T.F., "Digital Fundamental", 6th Edition, Prentice-Hall International Inc, New Delhi, 2008, Pp 4-7.

[16] Kopetz H., "Real-Time Systems: Design Principles for Distributed Embedded Applications", Kluwer Academic Publishers, Dordrecht, 2011, Pp 6-11.

[17] P. T. Chimford, "The Concise Book of real time systems", Timesys Corporation, Pittsburgh, 2012.

[18] Predko Myke, "Handbook of Microcontrollers", McGraw Hill, New York, USA, 2007, Pp 1-17.

[19] Holliday D. and Resmick Robert, "Fundamentals of Microcontrollers", Don Peters Press Ltd, Fulmar, 2010, P 88.

[20] Katz P., "Digital Controls Using Microcontrollers", Prentice Hall International Inc, New Delhi, 2009, Pp 15-43.