Design & Development of a Microcontroller Based Electronic Queue Control

Developed by:

Morium Akter

ID: 2012-1-56-019

&

Naheed Md. Tousif Khan

ID: 2012-3-50-007

East West University

Department of Electronics & Communication Engineering

Project Supervisor:

Dr. Md. Habibur Rahman

Professor

Dept. of Electrical and Electronics Engineering

University of Dhaka

DECLARATION

We hereby declare that we carried out the work reported in this project in the **Department of Electronics and Communication Engineering, East West University,** under the supervision of Dr. Md. Habibur Rahman. We solemnly declare that to the best of our knowledge, no part of this report has been submitted to elsewhere for award of any degree. All sources of knowledge, given in this report, have been duly acknowledged.

Signature:

.....

Morium Akter

ID : 2012-1-56-019

.....

Naheed Md. Tousif Khan

ID: 2012-3-50-007

Supervisor

Dr. Md. Habibur Rahman

Professor

Dept. of Electrical and Electronics Engineering

University of Dhaka

APPROVAL

This is to certify that the Project titled **"Design & Development of a Microcontroller Based Electronic Queue Control"** submitted to the honorable members of the Board of Examiners of the Faculty of Engineering for partial fulfillment of the requirements for the degree of Bachelor of Science in Electronics & Telecommunications Engineering by the following students has been accepted as satisfactory.

Submitted By:

Morium Akter
ID : 2012-1-56-019
&
Naheed Md. Tousif Khan
ID : 2012-3-50-007

Dr. Md. Habibur Rahman
Professor, Electrical and Electronics Engineering
University of Dhaka

Dr. M. Mofazzal Hossain
Chairperson
Dept. of Electronics and Communication Engineering
East West University

ACKNOWLEDGEMENTS

Many people deserve our thanks for their help in completing this project. We would like to thank our department for giving us the chance to do this project and would remain grateful to all the faculties of ECE Department. We want to express our thanks and deep appreciation to our advisor Dr. Md. Habibur Rahman as he has exhausted all his knowledge and time by following up our daily progress and encouraging advices and even by sharing on the troubles. We would like to extend our thanks to the laboratory staff of ECE Dept. for their fast response and cooperation with us to get some materials we need for our case. I have special acknowledgment for my group member for his understanding and hard working from the beginning up to the end.

Finally, we would like to thank our family and all persons who were involved with this project for their valuable help and professionalism during this project.

ABSTRACT

In modern days, we must use various high-tech electronic devices and equipment to get our jobs done and make the life easier. The purpose of this project is to design an Electronic Queue Control System. Here, we have used 4-bit parallel registers (74LS175), BCD to 7-segment display decoder driver (74LS47) and 7-segment display unit and an Arduino Uno board. Four I/O pins of Arduino are arranging in the multiplexed form to send data to the display units. Different Clock pulses for 74LS175 registers have been used to save the input data in a particular register. 4-bit parallel register save the input data from Arduino and send to the input of BCD to 7-segment display decoder driver. Getting the output, BCD to 7-segment decoder driver display the binary number as (0-9) in the 7-segment displays. This project will show how three 7-segment displays can be used for counter numbers (0-9) (where services to the customers will be provided) and serial numbers (0-99). The developed system can be used like a Queue Control system of any Supermarket checkouts, Banks, Customer Care, Airport security etc. The system has been designed and implemented practically. It is found that the system works perfectly.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1Queue

In general, a queue is a line of people or things waiting to be handled, usually in sequential order starting at the beginning or top of the line or sequence. In computer technology, a queue is a sequence of work objects that are waiting to be processed. The possible factors, arrangements, and processes related to queues. Common implementations are circular buffers and linked lists. Examples:: include checking out groceries or other goods that have been collected in a self service shop, at an ATM, at a ticket desk, a city bus, or in a taxi stand.

1.2Applications of Queue

People form a queue in a fixed, predictable position, such as at supermarket checkouts, some other retail locations such as banks, airport security In the field of facilities management, structured queues are commonly known with different names like "Queue Managers" or "Crowd Controllers" or "Public Guidance Systems Very often, queue management systems are set up to manage ticket ranking for a service (with or without a numbered ticket) with identification and thus enable a serene and stress-free waiting. Extending the different possibilities, planned reception by appointment and remotely rank allocation on or through Short Message Service can also be included.

1.3Purpose of This Project

This project will show how can we set up Arduino with BCD to 7-segment display decoder and 4-bit parallel register and 7-segment display as Queue Control System. In recent years many places use this system to remove crowd and for doing a peaceful work. This system is work as like First-Come-First-Service. Clock pulse of 4-bit parallel register is essential for creating this queue system. Because it is used to save the Input data and show as a counter display. 4-bit parallel register save the input data from Arduino and send the output to BCD to 7-segment display decoder driver. Getting the output, BCD to 7-segment display decoder driver display the binary number as (0-9) in the 7-segment display. This project will show how 3piece 7-segment display use as counter number(0-9) and serial number (0-99) like a Queue Control system.

In short the purpose of the project is---

- To design a smart control system
- To construct a smart electronic queue controller circuit in the breadboard
- Test for its functionality
- Product commercialization
- To design the control system with low cost components

1.4 Outline of the report

Here we describe six chapters on the purpose of this project. First we describe the introduction part that contains Queue, Applications, Purpose of this project. In the 2nd chapter we describe An over view of Microcontroller that contains Introduction, Microcontroller, Advantages of Microtroller. In the 3rd chapter we describes about Arduino, Schematic diagram of Arduino, Introduction of 7-Segment Display, BCD to 7-Segment display decoder driver(74LS47), 4-bit parallel Register(74LS175), Truth table, Logic diagram of 74LS47 & 74LS175 etc. In the 4th chapter we describes "The Theory Behind The Project" that brifly describes about the equipment related to the project as 74LS47 & 74LS175, Switch, 7-segment display ,Block diagram, Full diagram of Ckt, All Schematic Diagrams, Programming Code, Flow Chart etc. In the 5th chapter we describes about Implementation & Result of the project. Last of all we describe the conclusion part that contains conclusion and future work scope etc.

Chapter 2

AN OVERVIEW OF MICROCONTROLLERS

2.1 Introduction

Arduino is an open source electronics platform based on easy-to-use hardware and software. In this project we use Arduino-UNO board. The Arduino hardware platform already has the power and reset circuitry setup as well as circuitry to program and communicate with the microcontroller over USB. In addition, the I/O pins of the microcontroller are typically already fed out to sockets/headers for easy access. On the software side, Arduino provides a number of libraries to make programming the microcontroller easier. The simplest of these are functions to control and read the I/O pins rather than having to fiddle with the bus/bit masks normally used to interface with the At mega I/O.

2.2 Microcontroller

A microcontroller is a small computer on a single integrated circuit containing a processor core, memory and programmable input/output peripherals. It is highly integrated chip that contains all the components comprising a controller. Microcontrollers are used in automatically controlled products and devices, such as automobile engine control systems, remote controls, office machines, implantable medical devices, appliance and other embedded systems. We used here for appliance cases.

2.3Advantages of Microcontroller

- The simplest computer processor is used as the "brain" of the future system.
- Single purposes and low power consumption.
- Depending on the taste of the manufacturer, a bit of memory, a few A/D converters, timers, input/output lines etc. are added.
- Low cost and small packaging.
- Simple software able to control it all and which everyone can easily learn about has been developed.

2.4 Development

The computer, commonly defined as a tool for processing, storing, and displaying information, arose from a long line of analog devices used for effective counting and calculation, ranging from the simple abacus (first invented in Sumeria around 2300 BC), to Napier's Bones (conceived in 1617, and the precursor to the slide rule), to Blaise Pascal's gear-based mechanical calculator (1645).The development of the computer accelerated during the 1940's, spurred on largely by the highly industrialized nature of military production in World War II.The 1960's marked a significant evolutionary leap for computing, due to the development of solid state computers (such as the IBM 1401), which used transistors for processing operations, and magnetic core memory for storage. The invention of integrated circuits in 1959 by Jack Kilby, which enabled transistors and circuits to be fused onto small chips of semiconducting materials (such as silicon), allowed further miniaturization of computer components. Another important development during this decade was the advent of high-level computer programming languages that were written in symbolic language, making computer code somewhat easier to read and learn than previous machine languages. COBOL and FORTRAN were the main languages introduced during this period. The microprocessor was introduced in 1970. The microprocessor essentially miniaturized all hardware components of a computers central processing unit to fit onto a single, tiny integrated circuit, now more popularly known as a microchip. The microchip also became the main driving component of microcontrollers (such as the Arduino), which generally consist of a microchip, memory storage hardware, and input/ output hardware for sensors. The 1970's and 1980's also saw the development of a new generation of more powerful programming languages (such as C, C++, and later Java) for applications in business and science.

2.5 Evoluation

The PIC microcontroller board, introduced in 1985 by General Instruments, became one of the most popular tools for electronics enthusiasts (before the Arduino) for several reasons. Other popular boards for hobbyists include the BASIC Stamp (Parallax Inc., 1990), and Wiring both of which share the benefits of simplicity of programming, and a resulting ease of rapid-prototyping. The Arduino project grew largely out of the "DIY" climate created by the burgeoning popularity of rapid- prototyping boards like PIC. In fact, the immediate precursor to the Arduino was a custom made Wiring microcontroller created by the artist/ designer Hernando Barragan in 2004 for his. In 2005, the Arduino team was formed in Ivrea, Italy, consisting of Barragan, Massimo Banzi, David Cuartielles, Dave Mellis, Gianluca Marino, and Nicholas Zambetti. As a result, the Arduino incorporated the following characteristic, a programming environment based on
processing language (a programming language conceived by Ben Fry and Casey Reas, also conceived for artists/ designers), the ability to program the board via a standard USB connection, and a low price point. The Arduino achieved rapid success even within its first two years of existence, selling in a quantity of more than 50,000 boards. By 2009, it had spawned over 13 different incarnations, each specialized for different applications. Today, the Arduino microcontroller has become one of the most popular prototyping platforms in the world, and is a prime example of how hardware and software technologies originally created for military, business, and scientific applications can be repurposed to serve the needs of individuals creating projects in the realms of new media art and design.

Chapter 3

THEORY BEHIND THE PROJECT

3.1 Arduino-UNO

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. Arduino consists of both a physical programmable circuit board and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board. The Arduino platform has become quite popular with people just starting out with electronics, and for good reason. Unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware in order to load new code onto the board – simply use a USB cable. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package. The Arduino software is free, the hardware boards are pretty cheap, and both the software and hardware are easy to learn has led to a large community of users who have contributed code and released instructions for a huge variety of Arduino-based projects.

3.1.1 History of Arduino-UNO

The Arduino is developed in 2005. The Arduino microcontroller was initially created as an educational platform for a class project at the Interaction Design Institute Ivrea in Milan (Italy) in 2005. It derived from a previous work of the Wiring microcontroller designed by Hernando Barragan in 2004. From the beginning, the Arduino board was developed to attract artists and designers. The Wiring microcontroller was created by Hernando Barragan to be used for parsing data to electronic devices. His aim was that it could be used by non-technical people who only had basic experience with using computers. He first of all wanted it to be used as a prototyping tool. Since he needed help to create an easy software tool to programmed the board he engaged Casey Reas and Massimo Banzi as his assistants. Reas created the visual programming language for the prototyping tool[2].

3.1.2 Figure of Arduino-UNO

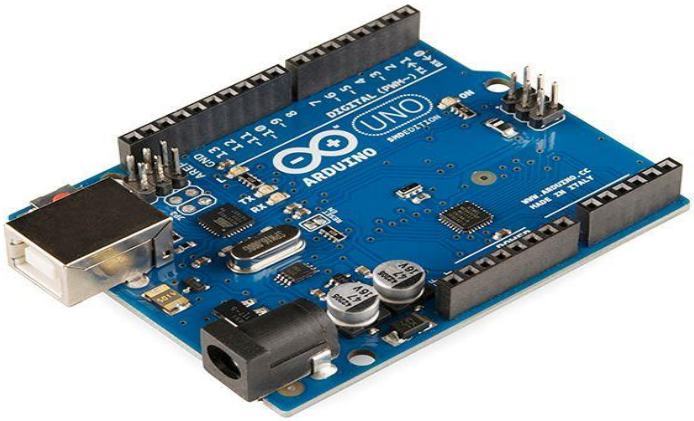


Fig 3.1.2: Arduino-Uno

3.1.3 Architecture of Arduino-UNO

There are many varieties of Arduino boards that can be used for different purposes. The Arduino UNO components are:

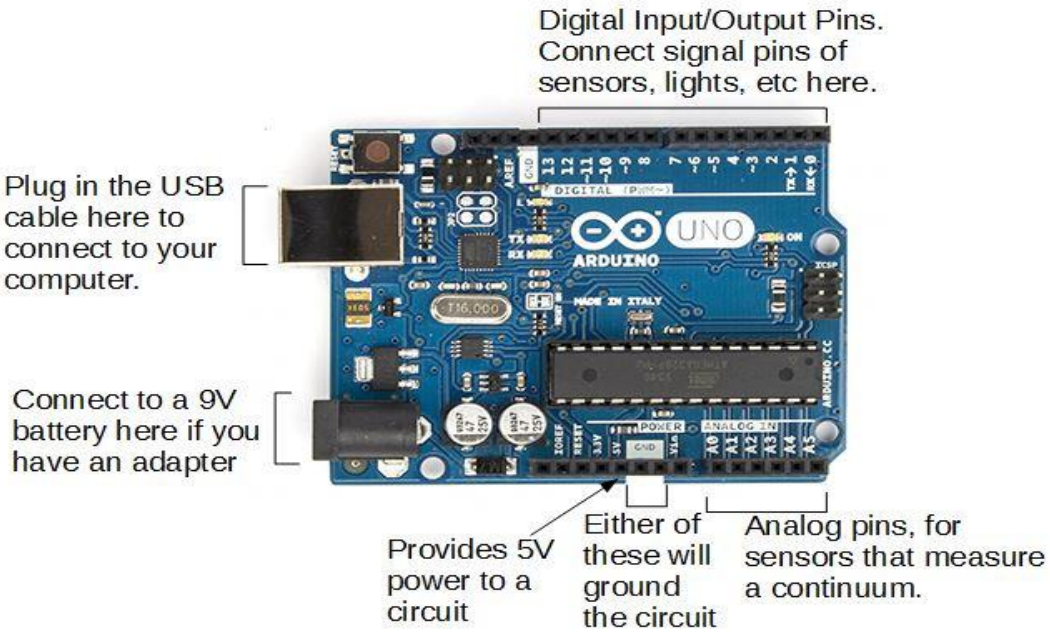


Fig 3.1.3: Arduino-UNO R3 Board

3.1.4 Power

The Arduino/Genuino Uno board can be powered via the USB connection or with an external power supply. The power source is selected automatically.
The power pins are as follows:

- **V_{in}:** The input voltage to the Arduino/Genuino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V:** This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- **3V3. A 3.3**: volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND:** Ground pins.
- **IOREF:** This pin on the Arduino/Genuino board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage

and select the appropriate power source or enable voltage translators on the outputs to work with the 5V or 3.3V.

3.1.5 Power LED Indicator

Just beneath and to the right of the word “UNO” on circuit board, there’s a tiny LED next to the word ‘ON’. This LED should light up whenever plug Arduino into a power source. If this light doesn’t turn on, there’s a good chance something is wrong.

3.1.6 Reset Button

The Arduino has a reset button. Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if code doesn’t repeat, but we want to test it multiple times. Unlike the original Nintendo however, blowing on the Arduino doesn’t usually fix any problems.

3.1.7 Tx Rx LEDs

Tx is short for transmit, Rx is short for receive. In our case, there are two places on the Arduino UNO where Tx and Rx appear once by digital pins 0 and 1, and a second time next to the Tx and Rx indicator LEDs. These LEDs will give us some nice visual indications whenever Arduino is receiving or transmitting data.

3.1.8 Main IC

The black thing with all the metal legs is an IC, or Integrated Circuit. The main IC on the Arduino is slightly different from board type to board type, but is usually from the AT mega line of IC’s from the ATMEL Company. This can be important, as may need to know the IC type before loading up a new program from the Arduino software. This information can usually be found in writing on the top side of the IC.

3.1.9 Voltage Regulator

The voltage regulator is not actually something interacting with on the Arduino. But it is potentially useful to know that it is there and what it’s for. It controls the amount of voltage that is let into the Arduino board. It will turn away an extra voltage that might harm the circuit.

3.2.0 Technical Specifications

Microcontroller	ATmega328P
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

3.2.1 Schematic Diagram

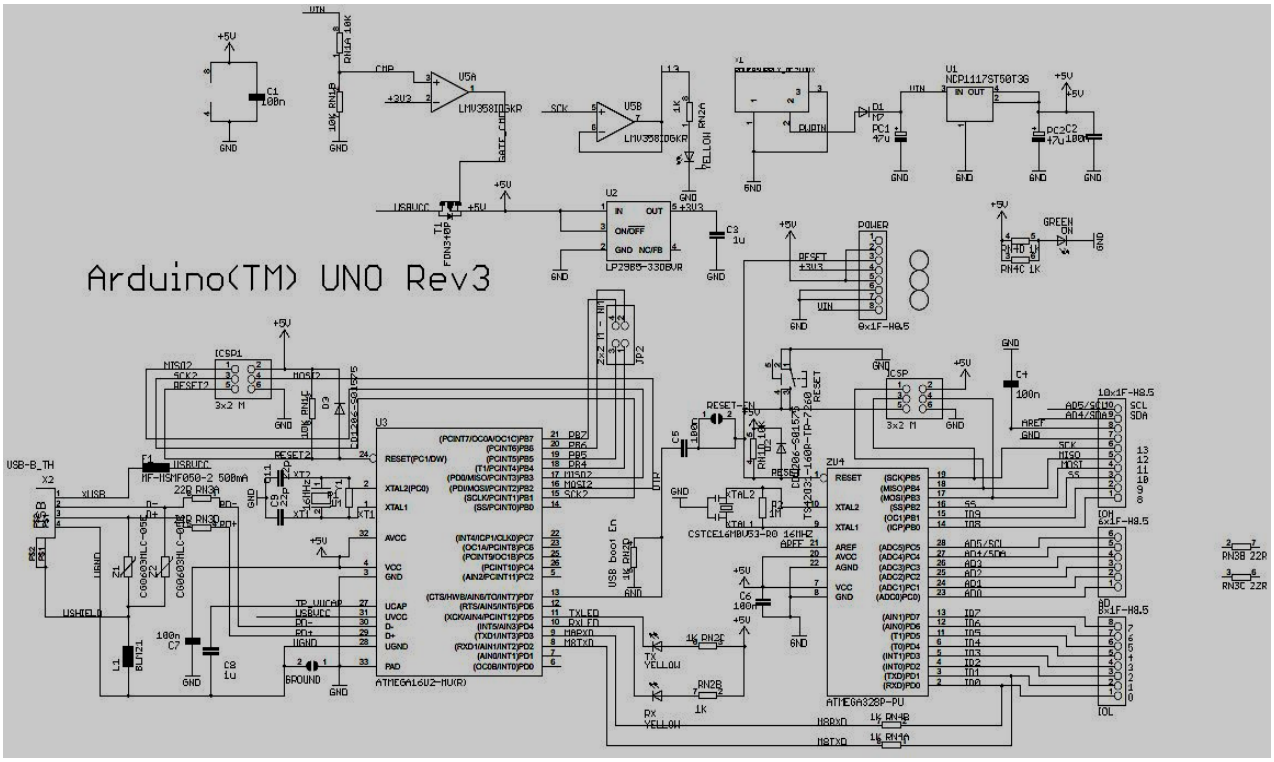


Fig 3.2.1: Schematic Diagram of Arduino-Uno

3.2 Getting started with Arduino Software

We have to download and install the Arduino for Mac, Linux or Windows from arduino.cc. Windows users also need to install a driver. Connect your board via USB, launch the Arduino application and select Arduino-Uno from the tools to board menu. Open the sketch File.
Examples: 01. Basics: Blink. Click the toolbar button to upload it to your board.

3.2.1 The Integrated Development Environment (IDE)

Every microcontroller needs software to be programmed. The Arduino board is not a case apart. It has its own integrated development environment (IDE).It is free and everyone can download it from its official website using either the Windows, Mac OS X or Linux platform. That allows Arduino Board to gain more users and it also helps it to grow.

3.2.2 IDE Parts

- Compile: Before program "code" can be sent to the board, it needs to be converted into instructions that the board understands. This process is called Compiling.
- Stop: This stops the compilation process.
- Create new Sketch: This opens a new window to create news ketch.
- Open Existing Sketch: This loads a sketch from a file on our computer.
- Save Sketch: This saves the changes to the sketch.
- Upload to Board: This compiles and then transmits over the USB cable to our board.
- Serial Monitor: Until this point when our programs (sketches) didn't work, we just pulled out our hair and tried harder.
- Tab Button: This lets you create multiple files in your sketch. This is for more advanced programming than we will do in this class.
- Sketch Editor: This is where write or edit sketches
- Text Console: This shows you what the IDE is currently doing and is also where error messages display if make a mistake in typing program.
- Line Number: This shows what line number your cursor is on.

Fig 3.2.2: Arduino-Uno

3.3 7-SEGMENT DISPLAY

3.3.1 Introduction

A seven-segment display (SSD) is a widely used electronic display device for displaying decimal numbers from 0 to 9. They are most commonly used in electronic devices like digital clocks, timers and calculators to display numeric information. As its name indicates, it is made of seven different illuminating segments which are arranged in such a way that it can form the numbers from 0-9 by displaying different combinations of segments. It is also able to form some alphabets like A, B, C, H, F, E, etc.

3.3.2 History

Seven-segment representation of figures can be found in patents as early as 1903 (in U.S. Patent 1,126,641), when Carl Kinsley invented a method of telegraphically transmitting letters and numbers and having them printed on tape in a segmented format. In 1908, F W Wood invented an 8-segment display, which displayed the number 4 using a diagonal bar (U.S. Patent 974,943). In 1910, a seven-segment display illuminated by incandescent bulbs was used on a power-plant boiler room signal panel.[8] They were also used to show the dialed telephone number to operators during the transition from manual to automatic telephone dialing.[9] They did not achieve widespread use until the advent of LEDs in the 1970s.

3.3.3 Figure

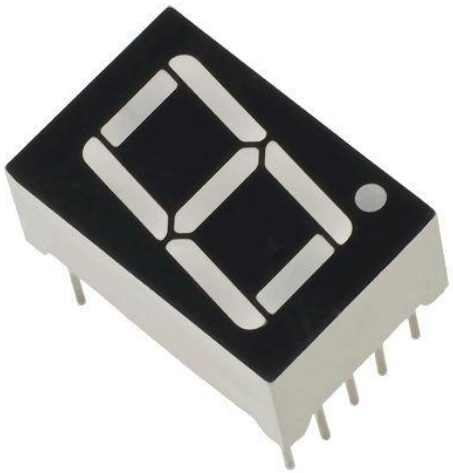


Fig 3.3.3: 7-Segment Display

3.3.4 Truth Table

Table 2.4.2											
Decimal	BCD Inputs				7 Segment Outputs						
	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
10	X	X	X	X	0	0	0	0	0	0	0
11	X	X	X	X	0	0	0	0	0	0	0
12	X	X	X	X	0	0	0	0	0	0	0
13	X	X	X	X	0	0	0	0	0	0	0
14	X	X	X	X	0	0	0	0	0	0	0
15	X	X	X	X	0	0	0	0	0	0	0

Fig 3.3.4: Truth Table

3.4 BCD to 7-Segment Display Decoder Driver

3.4.1 Introduction

The 74xx47 chip is used to drive 7 segment display. We must use the 74xx47 with a common anode 7-segment display. The input to the 74xx47 is a binary number DCBA where D is 8s, C is 4s, B is 2s and A is 1s. The inputs DCBA often come from a binary counter. The display is only sensible if the binary number is between DCBA=0000 (0) and DCBA=1001 (9); this is called Binary Coded Decimal or BCD for short. If the number is larger than 9 you get a strange output on the display. Try this out by moving your mouse over the truth table.
The inputs $\overline{BI/RBO}$, $\overline{RBI}$ and $\overline{LT}$ are usually connected to 5v.

#Function of $\overline{LT}$ & $\overline{BI/RBO}$, $\overline{RBI}$:

$\overline{BI/RBO}$ Stands for Blanking Input & Ripple Blanking Output.

$\overline{RBI}$ Stands for Ripple Blanking Input.

$\overline{LT}$ Stands for Lamp Test.

1. BI/RBO is wire-OR logic serving as blanking input (BI) and/or ripple-blanking output (RBO). The blanking input must be open or held at a high logic level when output functions 0 through 15 are desired, and ripple-blanking input (RBI) must be open or at a high logic level during the decimal 0 output. X = input may be high or low.
2. When a low logic level is applied to the blanking input (forced condition) all segment outputs go to a low logic level regardless of the state of any other input condition.
3. When ripple-blanking input (RBI) is at a low logic level, lamp test input is at high logic level and

A = B = C = D = low logic level, all segment outputs go to a low logic level and the ripple-blanking output goes to a low logic level (response condition).
4. When blanking input/ripple-blanking output is open or held at a high logic level, and a low logic level is applied to lamp test input, all segment outputs go to a high logic level.

3.4.2 Connection

Simple use:

1. Connect V_{cc} [pin 16], $\overline{LT}$ [pin 3], $\overline{BI/RBO}$ [pin 4] and $\overline{RBI}$[pin 5] to 5v.
2. Connect Gnd [pin 8] to 0v.
3. connect DCBA [pins 1, 2, 6 and 7] to DCBA on your counter.
4. Connect $\overline{a}$, $\overline{b}$, $\overline{c}$, $\overline{d}$, $\overline{e}$, $\overline{f}$, $\overline{g}$ [pins 9-15] to a b c d e f g on the common anode 7-segment display.

3.4.3 Figure

Fig 3.4.3: BCD to 7-segment display decoder driver

3.4.4 Connection with 7-segment display

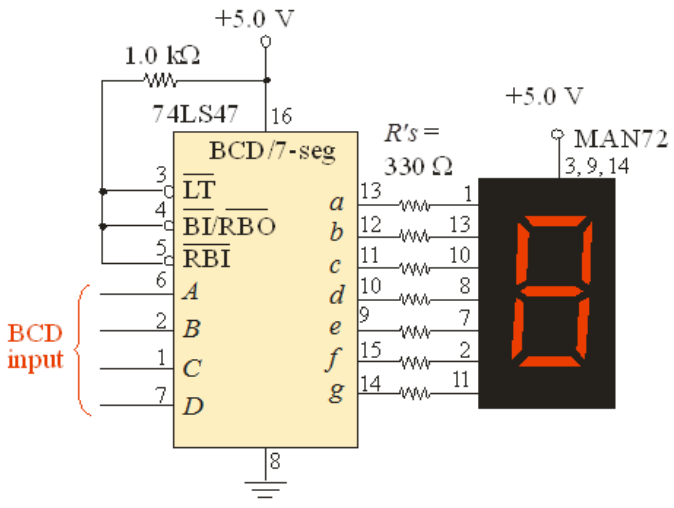
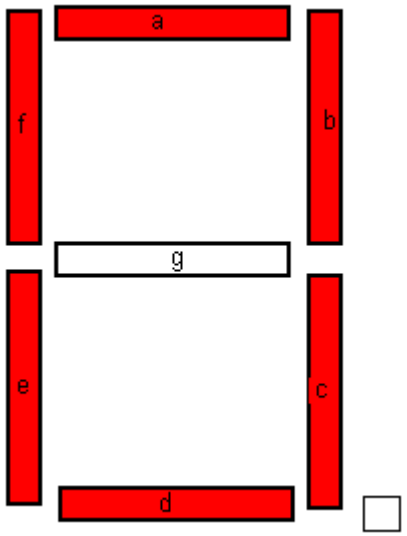


Fig 3.4.4: Connection with 7-segment display

3.4.5 Datasheet of 7447 decoder

Inputs							Outputs							
D	C	B	A	$\overline{BI}/\overline{RBO}$	$\overline{RBI}$	$\overline{LT}$	$\overline{a}$	$\overline{b}$	$\overline{c}$	$\overline{d}$	$\overline{e}$	$\overline{f}$	$\overline{g}$	Display
0	0	0	0	1	1	1	0	0	0	0	0	0	1	
0	0	0	1	1	1	1	1	0	0	1	1	1	1	
0	0	1	0	1	1	1	0	0	1	0	0	1	0	
0	0	1	1	1	1	1	0	0	0	0	1	1	0	
0	1	0	0	1	1	1	1	0	0	1	1	0	0	
0	1	0	1	1	1	1	0	1	0	0	1	0	0	
0	1	1	0	1	1	1	1	1	0	0	0	0	0	
0	1	1	1	1	1	1	0	0	0	1	1	1	1	
1	0	0	0	1	1	1	0	0	0	0	0	0	0	
1	0	0	1	1	1	1	0	0	0	1	1	0	0	
1	0	1	0	1	1	1	1	1	1	0	0	1	0	
1	0	1	1	1	1	1	1	1	0	0	1	1	0	
1	1	0	0	1	1	1	1	0	1	1	1	0	0	
1	1	0	1	1	1	1	0	1	1	0	1	0	0	
1	1	1	0	1	1	1	1	1	1	0	0	0	0	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	



3.5 4-bit Parallel Register(74LS175):

3.5.1 Introduction:

74LS175 is a high speed Quad D Flip-Flop. Thedevice is useful for general flip-flop requirements where clock and clear inputs are common. The information on the D inputs is stored during the LOW to HIGH clock transition. Both true and complemented outputs of each flip-flop are provided. A Master Reset input resets all flip-flops, independent of theClock or D inputs, when LOW.
The LS175 is fabricated with the Schottky barrier diode process for high
speed and is completely compatible with all Motorola TTL families.

- Edge-Triggered D-Type Inputs
- Buffered-Positive Edge-Triggered Clock
- Clock to Output Delays of 30 ns
- Asynchronous Common Reset
- True and Complement Output
- Input Clamp Diodes Limit High Speed Termination Effects

3.5.2 Logic Diagram:

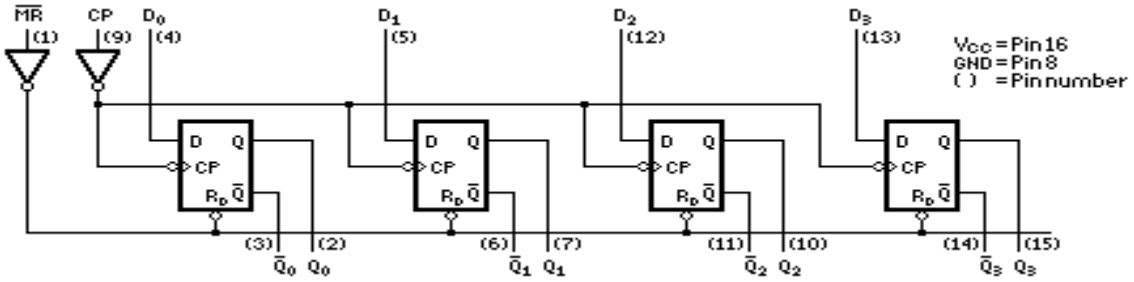


Fig 3.5.2: Logic Diagram of 4-bit parallel register

3.5.3 Figure

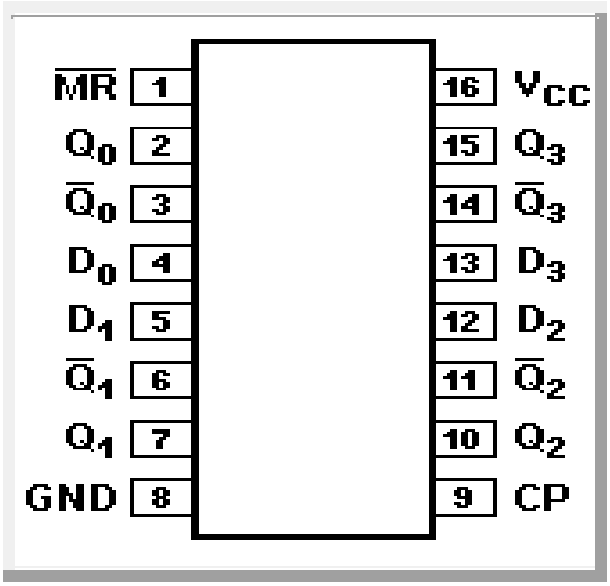


Fig 3.5.3: 4-bit Parallel Register

3.5.4 Truth Table:

D	CK	Q	$\overline{Q}$
0	High	0	1
1	High	1	0
X	0	Q_0	$\overline{Q_0}$
X	1	Q_0	$\overline{Q_0}$

Fig 3.5.4: Truth Table

Chapter-4

Design & Development of the System

4.1 Hardware Design

The whole system design is divided into two parts to design a queue control system. One is the design the smart system in the breadboard and controls the designed system. Another part is the display part design to count the value in smart system. Finally, the queue control system is formed a complete integrated system. In this project Arduino development board is more efficient.

4.2 Block Diagram:

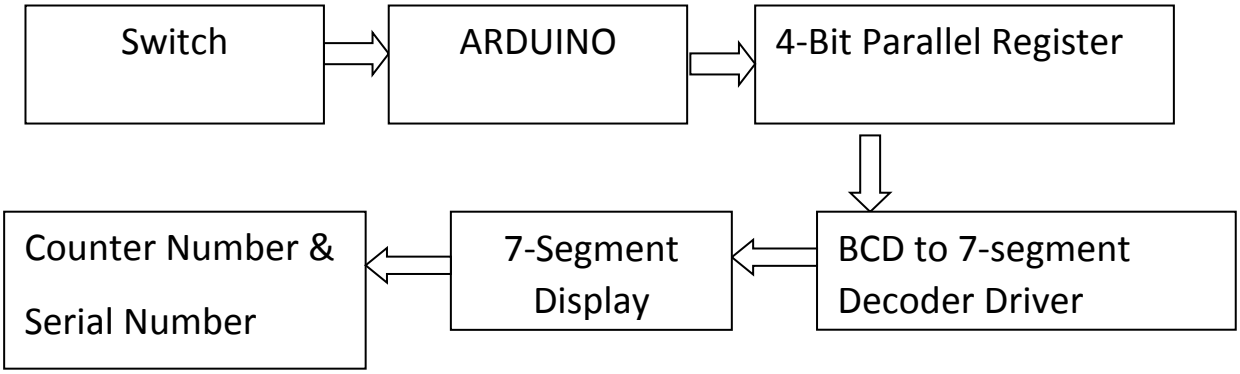


Fig 4.2: Block Diagram

4.3 Full Circuit Diagram

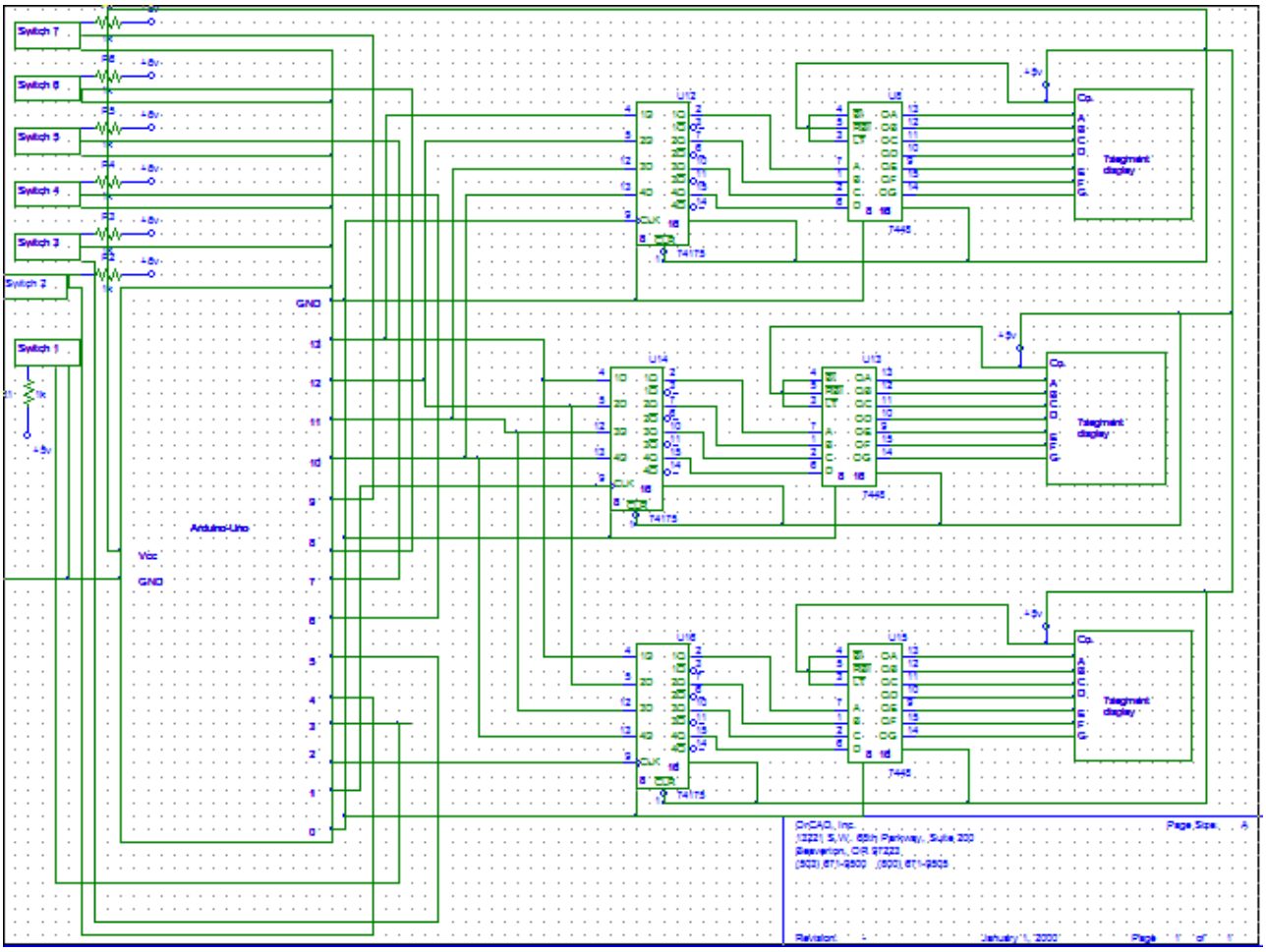


Fig 4.3: Full Circuit Diagram

4.3.1 Components Used in this System:

- **7-Segment Display**
- **BCD to 7-Segment Display Decoder Driver**
- **4-Bit Parallel Register**
- **Switch**
- **Arduino-Uno**

4.4 7-Segment Display

4.4.1 Introduction

Seven-segment displays are commonly used in digital clocks, clock radios, timers, wristwatches, and calculators. They can also be found in motor-vehicle odometers, speedometers, radio frequency indicators, and practically any other display that makes use of alphanumeric characters alone (without the need for graphics). Some seven-segment displays produce an "italicized" (slanted) set of characters.

4.4.2 Schematic Diagram of 7-Segment Display

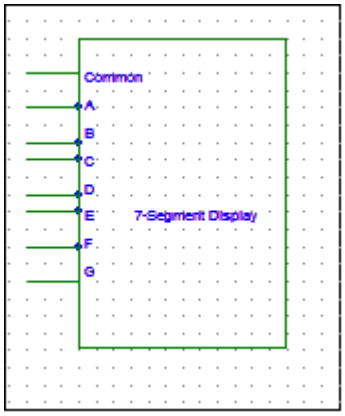


Fig 4.4.2: 7-Segment Display

4.4.3 Schematic Diagram of connection between 7-Segment Display & 7447

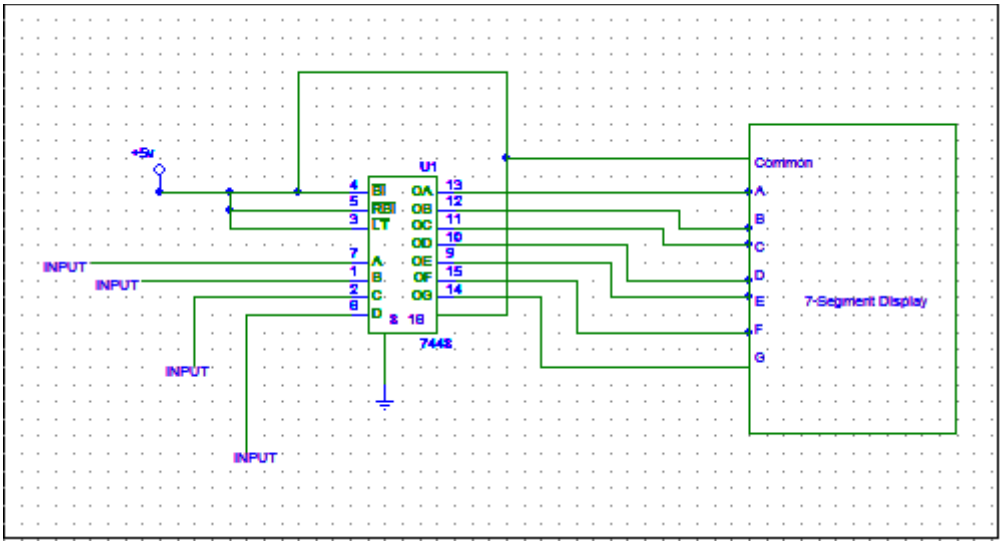


Fig 4.4.3: Schematic Diagram of connection between 7-Segment Display & 7447

4.4.4 Description

Seven-segment displays may use a liquid crystal display (LCD), a light-emitting diode (LED) for each segment, or other light-generating or controlling techniques such as cold cathode gas discharge (Panaplex), vacuum fluorescent, incandescent filaments (Numitron), and others. For gasoline price totems and other large signs, vane displays made up of electromagnetically flipped light-reflecting segments (or "vanes") are still commonly used. An alternative to the 7-segment display in the 1950s through the 1970s was the cold-cathode, neon-lamp-like nixie tube. Starting in 1970, RCA sold a display device known as the *Numitron* that used incandescent filaments arranged into a seven-segment display [10].

- 7-segment display has Seven Segment & a Common point.
- Common Point connected to Vcc/+5V.
- Seven Segment connected to the seven pin of 7447(BCD to 7-Segment decoder driver).
- A-segment connected to 13 no. Pin of 7447.
- B-segment connected to 12 no. Pin of 7447.
- C-segment connected to 11 no. Pin of 7447.
- D-segment connected to 10 no. Pin of 7447.
- E-segment connected to 9 no. Pin of 7447.
- F-segment connected to 15 no. Pin of 7447.
- G-segment connected to 14 no. Pin of 7447.

4.5 BCD to 7-Segment Decoder Driver

4.5.1 Introduction

74LS47 (BCD to 7-Segment display) accepts four lines of BCD (8421) input data and generates their complements internally. The data is decoded with seven AND/OR gates to drive indicator LEDs of the seven segment directly. The outputs correspond to Common anode (CA) configuration of seven segment[11].

4.5.2 Schematic Diagram of 74LS47

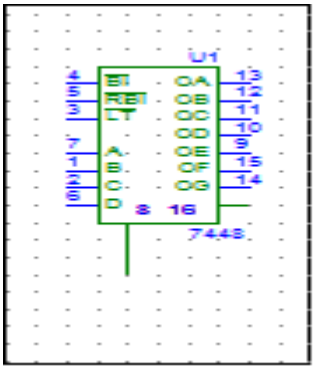


Fig 4.5.2: Schematic Diagram of 7447

4.5.3 Schematic Diagram of connection between 7-Segment Display & 7447

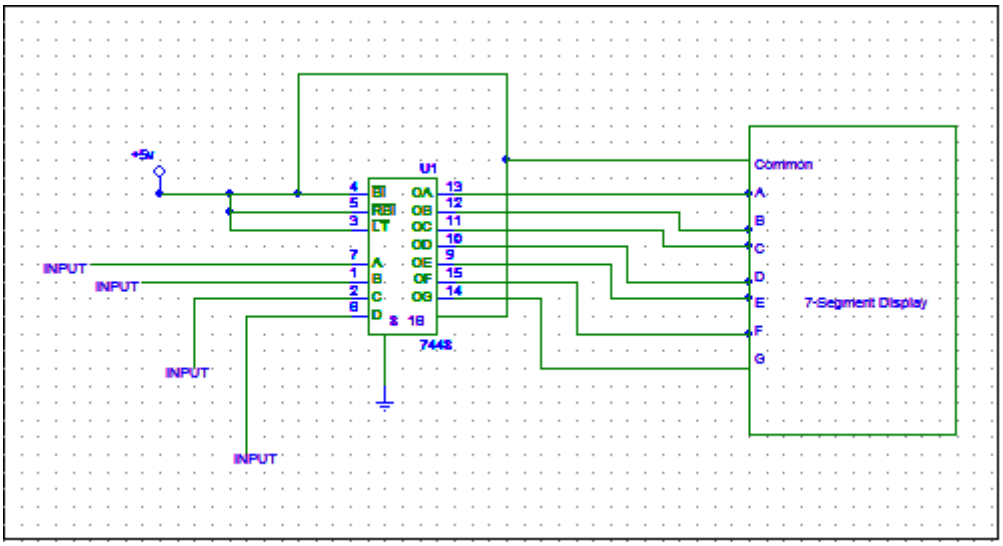


Fig 4.5.3: Schematic Diagram of connection between 7447 & 7-Segment Display

4.5.4 Description

74LS47 is a BCD to 7-Segment display decoder driver. It decodes the binary input. It has 16 pin. The outputs correspond to Common anode (CA) configuration of seven segment.

- $\overline{LT}$ [pin 3], $\overline{BI/RBO}$ [pin 4] , $\overline{RBI}$[pin 5] are connected to Vcc.
- 13 no. Pin of 7447 connected to A-segment.
- 12 no. Pin of 7447 connected to B-segment.
- 11 no. Pin of 7447 connected to C-segment.
- 10 no. Pin of 7447 connected to D-segment.
- 9 no. Pin of 7447 connected to E-segment.
- 14 no. Pin of 7447 connected to G-segment.
- 15 no. Pin of 7447 connected to F-segment.
- Pin no. 1,2,6,7 connected as Output of 74LS175(4-bit Parallel Register).

4.6 Connection of Switches

4.6.1 Introduction

A push-button or simply button is a simple switch mechanism for controlling some aspect of a machine or a process. Buttons are typically made out of hard material, usually plastic or metal. The surface is usually flat or shaped to accommodate the human finger or hand, so as to be easily depressed or pushed. Buttons are most often biased switches, though even many un-biased buttons (due to their physical nature) require a spring to return to their un-pushed state. Different people use different terms for the "pushing" of the button, such as press, depress, mash, hit, and punch.

4.6.2 Figure of Switch:

Fig 4.6.2: Switch

4.6.3 Schematic Diagram of Switch

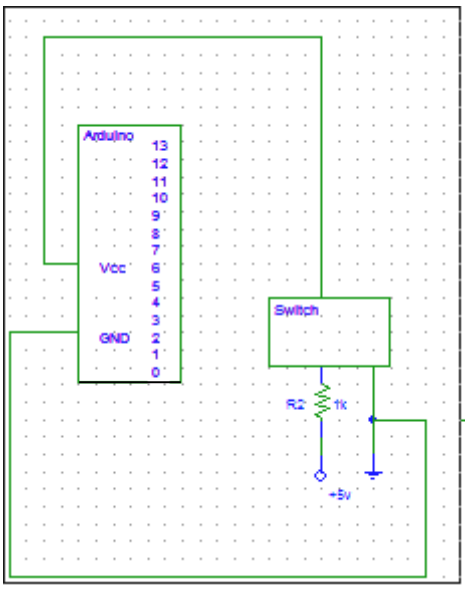


Fig 4.6.3: Schematic Diagram of Switch With Arduino

4.6.4 Description

The "push-button" has been utilized in calculators, push-button telephones, kitchen appliances, and various other mechanical and electronic devices, home and commercial. In industrial and commercial applications, push buttons can be connected together by a mechanical linkage so that the act of pushing one button causes the other button to be released. In this way, a stop button can "force" a start button to be released. This method of linkage is used in simple manual operations in which the machine or process have no electrical circuits for control. Pushbuttons are often color-coded to associate them with their function so that the operator will not push the wrong button in error. Commonly used colors are red for stopping the machine or process and green for starting the machine or process.

4.7 4-bit Parallel Register (74LS175)

4.7.1 Introduction

These circuits are positive clock-enable input.
Information at the D inputs meeting the setup time requirements is transferred to the Q outputs on the positive-going edge of the clock pulse if the clock-enable input (CLK) is low. Clock triggering occurs at a particular voltage level and is not directly related to the transition time of the positive-going pulse. When the clock input is at either the high or low level, the data (D) input signal has no effect at the output. The circuits are designed to prevent false clocking by transitions at the clock-enable (CLK) input.

4.7.2 Schematic Diagram of 4-bit parallel register(74LS175)

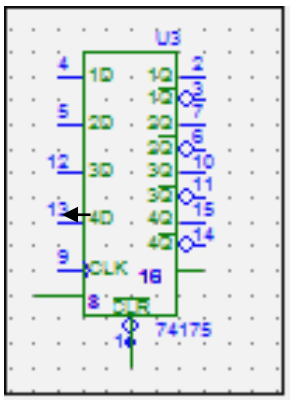


Fig 4.7.2: 4-bit parallel register(74LS175)

4.7.3 Schematic Diagram of 74LS175 with 7447 & 7-segment display

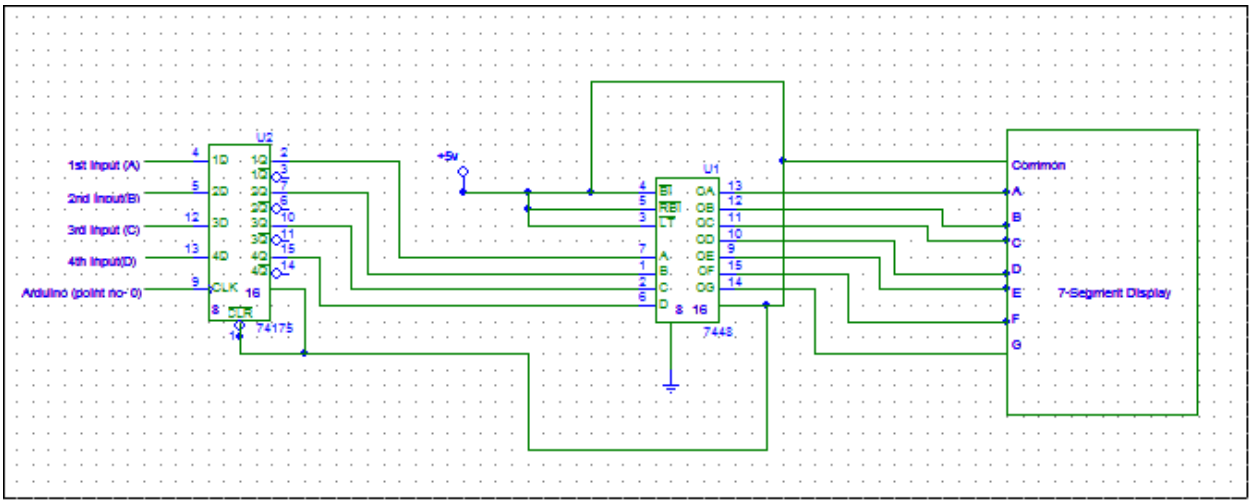


Fig 4.7.3: Schematic Diagram of 74LS175 with 7447 & 7-segment display

4.7.4 Description

The 74LS175 consists of four edge-triggered D flip-flops with individual D inputs and Q and Q outputs. The Clock and Master Reset are common. The four flip-flops will store the state of their individual D inputs on the LOW to HIGH clock (CP) transition, causing individual Q and Q outputs to follow . A LOW input on the Master Reset (MR) will force all Q outputs LOW and Q outputs HIGH independent of Clock or Data inputs[12].

- Master Reset [pin 1] & [pin 16]connected to Vcc.
- 1st output [pin 2] connected to 7447[pin7].
- 2nd output [pin 7] 1st connected to 7447[pin 1].
- 3rd output [pin 10] connected to 7447[pin2].
- 4th output [pin 15] connected to 7447[pin 6].
- Pin 3,6,11,14 has no connection.
- 1st Input [pin 4] connected to Arduino[pin13].
- 2nd Input [pin 5] connected to Arduino[pin12].
- 3rd Input [pin 12] connected to Arduino[pin11].
- 4th Input [pin 13] connected to Arduino[pin10].

4.8 Software Design

Software design is divided into two parts. First we write the Arduino program in Arduino software. Then we compile it to the Arduino hardware. This Arduino command is control the Arduino hardware and other circuit connection. Then check the output in 7-segment display.

4.8.1 Installing Arduino

Arduino runs on Windows. Then we go to the Arduino software web site at http://arduino.cc/en/ Main/Software and download the version of the software compatible with our system. We use Arduino 1.0.5 version[6].

4.8.2 Verifying the Hardware

Now that we have the Arduino IDE software installed, let's connect the computer to the Arduino board, load a small program, and verify that all components are working together. First, need to connect the USB cable to our mc board and then plug the other end of the USB cable into our computer.

4.8.3 Arduino Language

The Arduino language is implemented in C/C++ and based in Wiring. When we write an Arduino sketch, we are implicitly making use of the Wiring library, which is included with the Arduino IDE. This allows us to make run able programs by using only two functions: setup () and loop (). As mentioned, the Wiring language is inspired by Processing, and the Arduino language structure is inherited from the Processing language, where the equivalent functions are called setup (). We need to include both functions in every Arduino program, even if we don't need one of them. Let's analyze the structure of a simple Arduino sketch using again the Blink example.

4.9 Programming Code With Arduino

```
 */
// the setup function runs once when you press reset or power the board
int turn;
bool c1,c2,c3,c4,c5,c6,c7,c8,c9;


// the loop function runs over and over again forever
void digdis(int u,int d){
  digitalWrite(10,u%2);
  u=u/2;
  digitalWrite(11,u%2);
  u=u/2;
  digitalWrite(12,u%2);
  u=u/2;
  digitalWrite(13,u);
  digitalWrite(d,HIGH);
  delay(50);
```

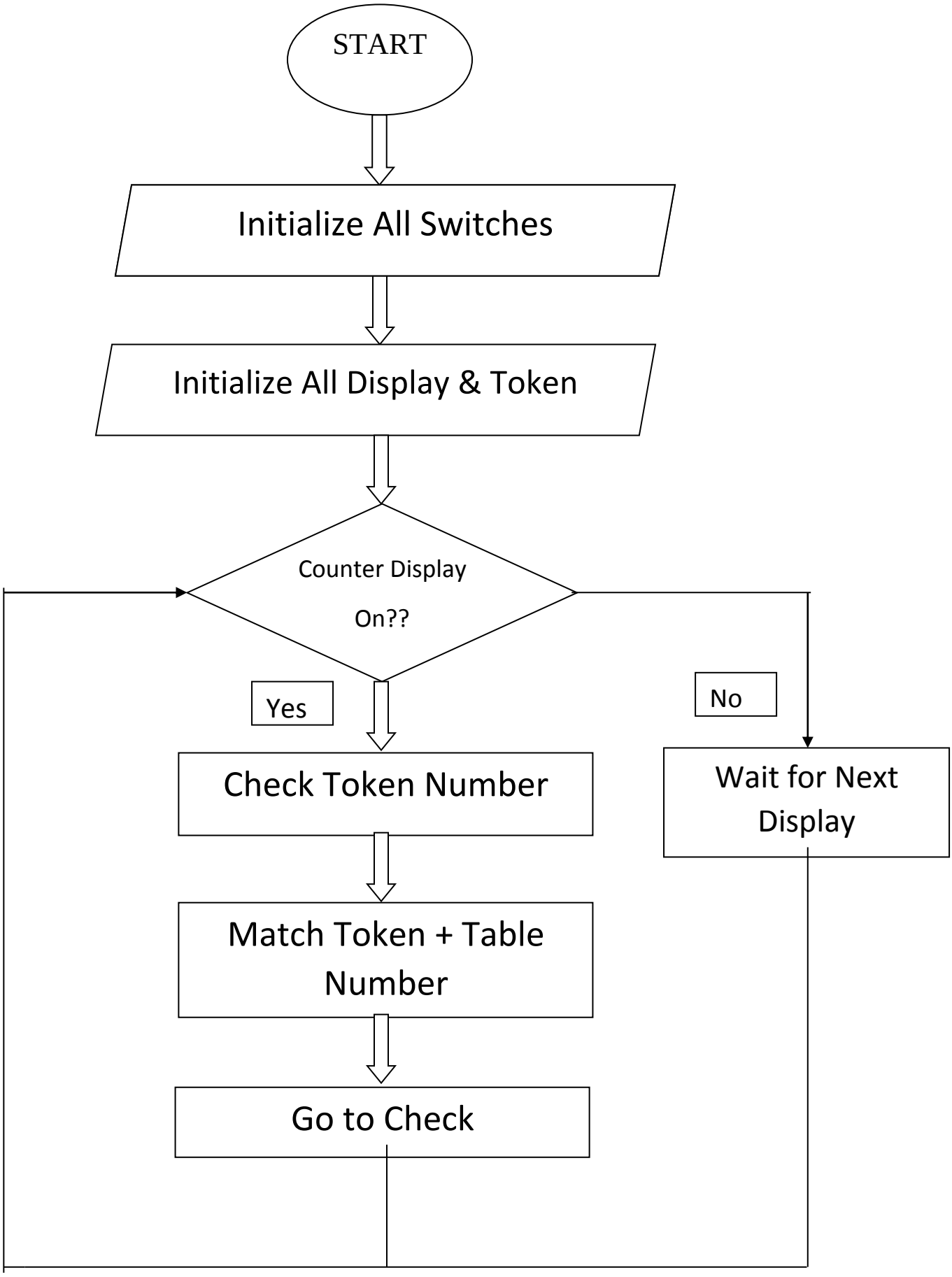
```
  digitalWrite(d,LOW);
}

void trndisp() {
  int u,t;
  u=turn%10;
  t=turn/10;
  digdis(t,1);
  digdis(u,2);
}

void setup() {
  // initialize digital pin 13 as an output.
  turn =0;
  c1=c2=c3=c4=c5=c6=c7=c8=c9=1;
  //display outputs
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  //clock pulses
  pinMode(0, OUTPUT);
  pinMode(1, OUTPUT);
  pinMode(2, OUTPUT);
  //input switches
  pinMode(3, INPUT);
  pinMode(4, INPUT);
  pinMode(5, INPUT);
  pinMode(6, INPUT);
  pinMode(7, INPUT);
```

```
 pinMode(8, INPUT);
 pinMode(9, INPUT);
 //iniate clock pins LOW
 digitalWrite(0,LOW);
 digitalWrite(1,LOW);
 digitalWrite(2,LOW);
 trndisp();
 digdis(0,0);
}

void loop(){
 //cheking switch 1
 if(digitalRead(3)==0) {
   delay(10);
   if(digitalRead(3)==0);c1=0;
   }
 if(digitalRead(3)==1 & c1==0){
   c1=1;turn = (turn+1)%100; digdis(1,0); trndisp();
   }

 //cheking switch 2
 if(digitalRead(4)==0) {
   delay(10);
   if(digitalRead(4)==0);c2=0;
   }
 if(digitalRead(4)==1 & c2==0){
   c2=1;turn = (turn+1)%100; digdis(2,0); trndisp();
   }

  //cheking switch 3
```

```
 if(digitalRead(5)==0) {
   delay(10);
   if(digitalRead(5)==0);c3=0;
   }
 if(digitalRead(5)==1 & c3==0){
   c3=1;turn = (turn+1)%100; digdis(3,0); trndisp();
   }

 //cheking switch 4
 if(digitalRead(6)==0) {
   delay(10);
   if(digitalRead(6)==0);c4=0;
   }
 if(digitalRead(6)==1 & c4==0){
   c4=1;turn = (turn+1)%100; digdis(4,0); trndisp();
   }

//cheking switch 5
 if(digitalRead(7)==0) {
   delay(10);
   if(digitalRead(7)==0);c5=0;
   }
 if(digitalRead(7)==1 & c5==0){
   c5=1;turn = (turn+1)%100; digdis(5,0); trndisp();
   }

  //cheking switch 6
 if(digitalRead(8)==0) {
   delay(10);
   if(digitalRead(8)==0);c6=0;
```

```
    }
  if(digitalRead(8)==1 & c6==0){
    c6=1;turn = (turn+1)%100; digdis(6,0); trndisp();
    }
  //cheking switch 7
 if(digitalRead(9)==0) {
    delay(10);
    if(digitalRead(9)==0);c7=0;
    }
  if(digitalRead(9)==1 & c7==0){
    c7=1;turn = (turn+1)%100; digdis(7,0); trndisp();
    }

}
```

4.10Flow Chart Diagram:



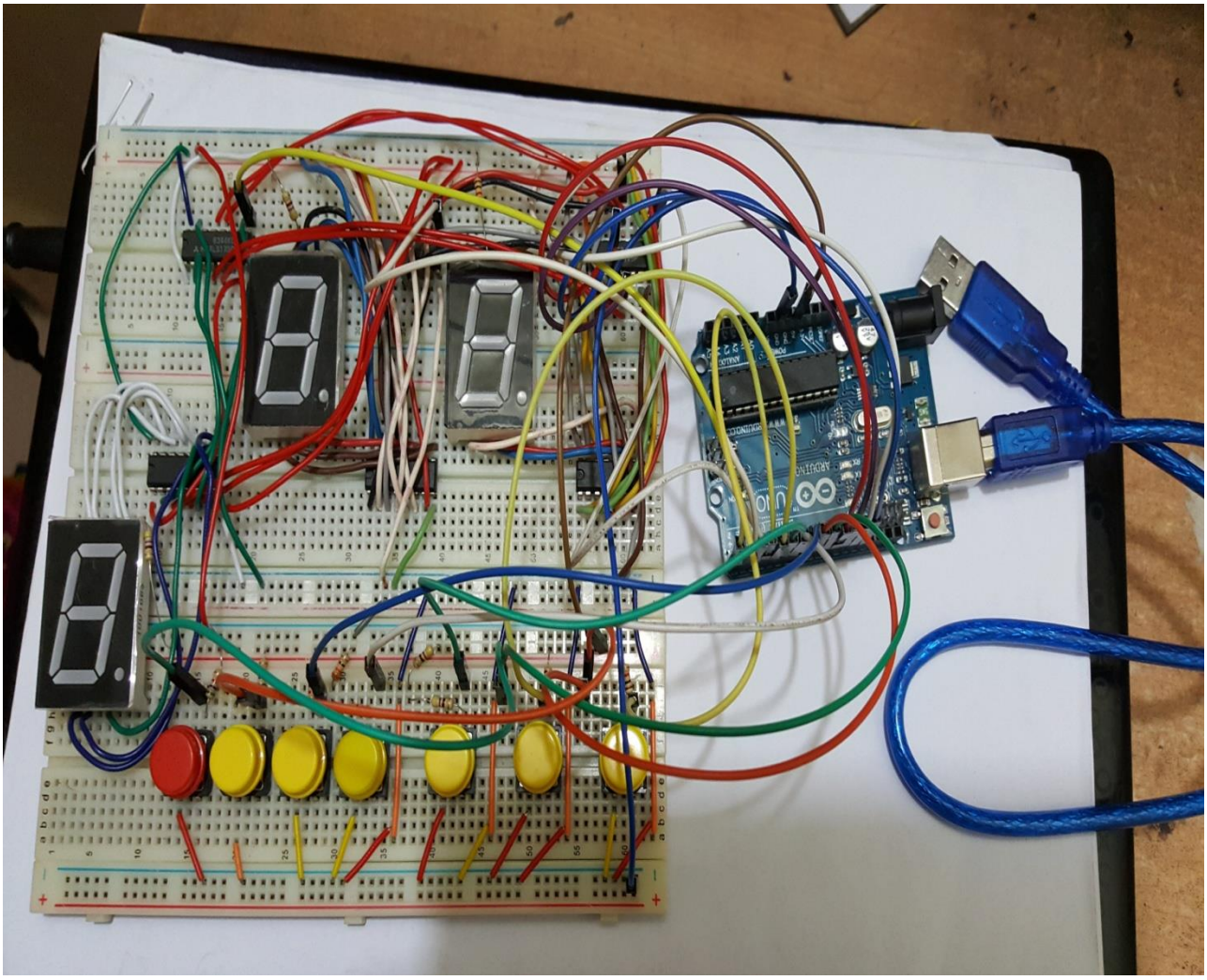
Chapter 5

IMPLEMANTATION AND RESULT

5.1 Implementation

All the parts are connected as the designed circuit in a bread board. Then we upload the programming code into the Arduino and tested the performance of the whole system. It works properly according to our design.

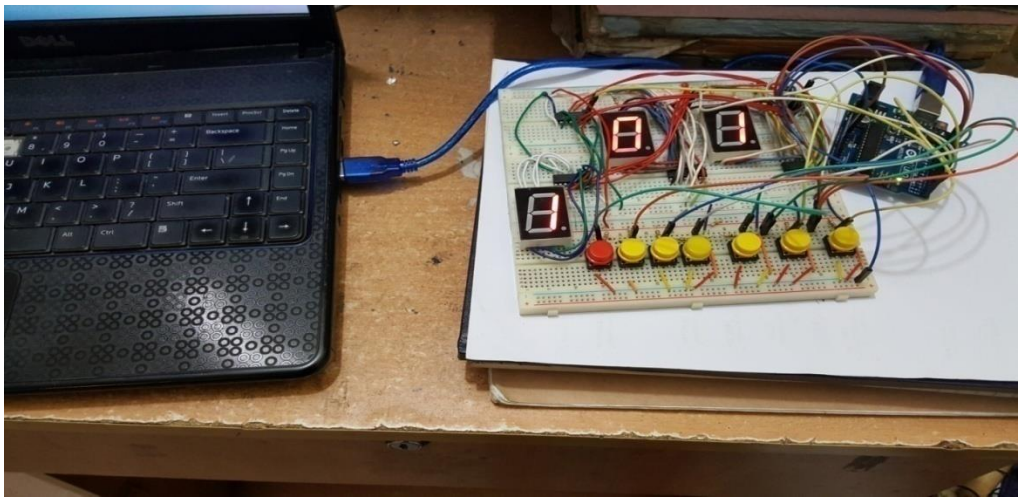
5.2 Connection of the whole system



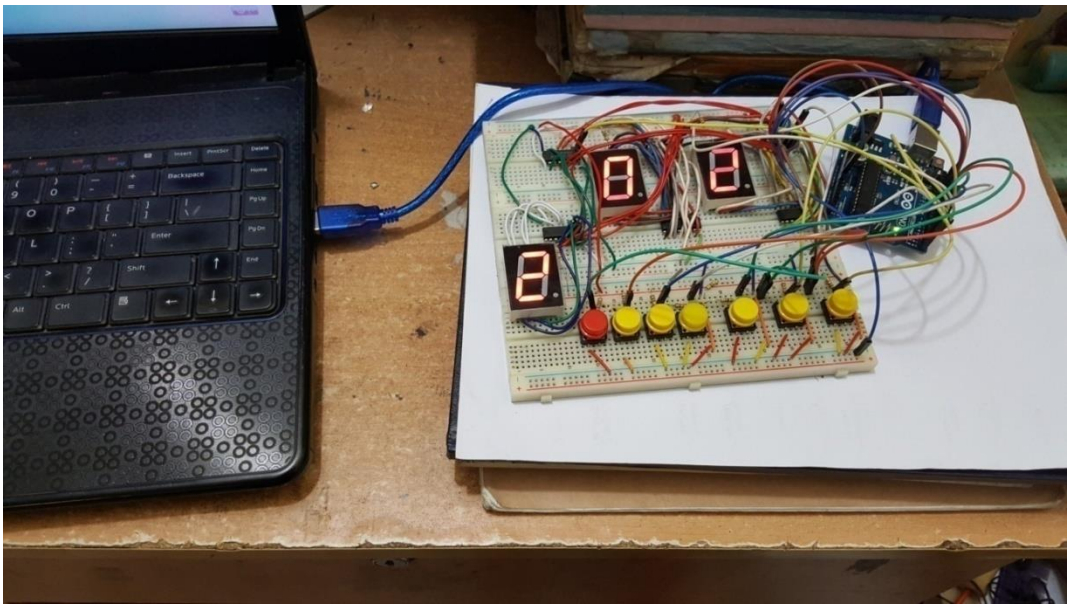
5.3 Result

Description

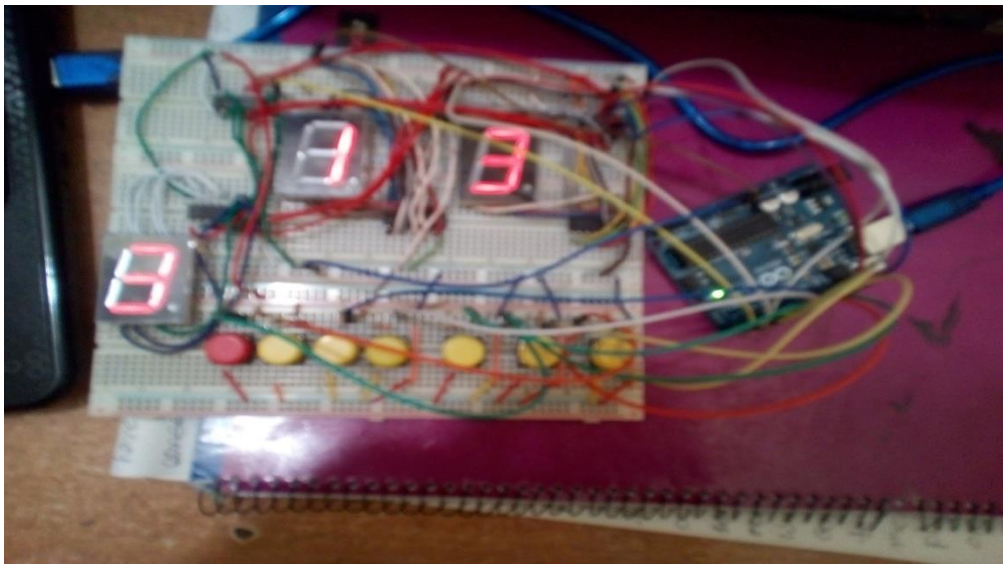
- After Completeing our code and project successfully, we connect the system with power supply and then tested. For a trial test, the button no. 1 has been pressed. When the button is not released, there is no change in the display. When the button is released, the counter display shows “1” and the serial no. disply shows “01”.
- Counter-1
- Serial Number- 01



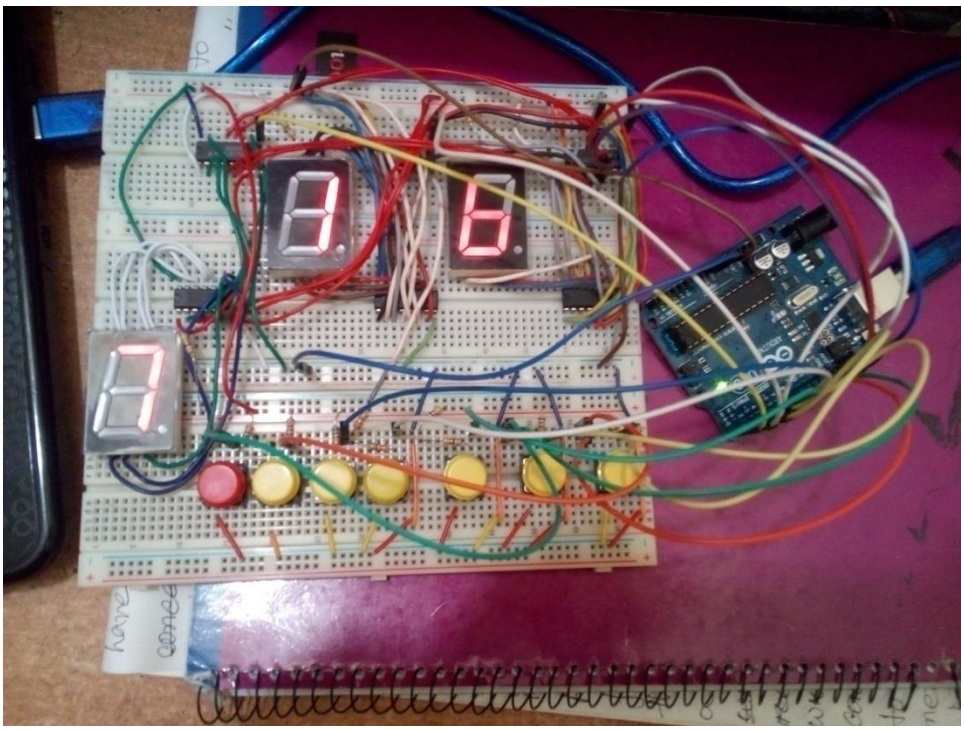
- For a another trial test, the button no. ‘2’ has been pressed. When the button is not released, there is no change in the display. When the button is released, the counter display shows “2” and the serial no. disply shows “02”.
- Counter-2
- Serial Number-02



- For a 3rd trial test, the button no. 3 has been pressed. When the button is not released, there is no change in the display. When the button is released, the counter display shows “3” and the serial no. disply shows “13”.
- Counter-3
- Serial No.-13



- For a 4th trial test, the button no. 7 has been pressed. When the button is not released, there is no change in the display. When the button is released, the counter display shows “7” and the serial no. disply shows “16”.
- Counter-7
- SerialNo. -16



5.4 Avoiding Unwanted Increase of Turn Number

Here we have used seven push button switches to display the counter no. and serial no. The switches are always kept in the “High” positions. When we press a button it produces “Low” output and the system understand the position but there will be no change in the display. When the button is released then the display shows the new results. For this feature, if anyone presses a button for some time the output will update once.

Another source of unwanted triggering of the system is switch bouncing effect. We know, when any mechanical switch is closed it will first produce some transient spikes that may miss-trigger the system. To avoid this miss-triggering, contact debouncing has been incorporated in the software.

Chapter 6

CONCLUSION

6.1 Conclusion

The Queue Control System Using 7-segment display has been designed and developed for making our life more easy and secured. We use 5V from Arduino board and BCD to 7-segment display decoder driver and 4-bit parallel register. Finally, we have designed and developed the whole control system and tested using different switches. We fix all the problems encountered during the design and testing of the system. Finally, we successfully achieved our goals. The developed Queue control system is efficient and the production cost is low. So, our work is suitable for commercial use.

6.2 Future Work Scope

We have used a small subset of each of these technologies. This project gives us an opportunity to do a big project in future. The applications stated above are some demo applications that are absolutely possible with its future development. Initially for the limitation of time and required fund we were able to develop just a Queue Control System.

REFFERENCE

1. The Electronics Handbook by J. C. Whitaker, 1996, CRC Press
2. Introduction to Arduino by Alan G. Smith, September 30, 2011
3. Arduino projects by Enrique Ramos Melgar and Ciriaco Castro Diez
4. Beginning Arduino by Michael McRoberts ,2nd Edition
5. https://upload.wikimedia.org/wikipedia/commons/3/38/Arduino_Uno_-_R3.jpg
6. https://www.arduino.cc/en/Guide/Introduction
7. https://upload.wikimedia.org/wikipedia/commons/3/38/Arduino_Uno_-_R3.jpg
8. https://www.arduino.cc/en/Main/ArduinoBoardUno
9. www.datasheetcatalog.com/datasheets_pdf/S/N/7/4/SN74175.shtml
10. http://thanglong.ece.jhu.edu/Course/137/Lab4-7segment.pdf
11. https://www.scribd.com/doc/42776362/BCD-to-7-segment-Display
12. http://www.engineersgarage.com/electronic-components/74ls47-datasheet