

DESIGN & DEVELOPMENT OF AN OPTICAL MARK
READER



Developed by:

Md. Abdul Khalek Sujon
ID: 2012-2-55-024

Md. Mashuk E-lahi
ID: 2012-2-55-048

Project Supervisor:

Dr. Md. Habibur Rahman
Professor

*Department of Electrical and Electronic Engineering,
Faculty of Engineering and Technology
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. HabiburRahman**. We also declare that no part of this work has been submitted to elsewhere partially or fully for the award of any degree or publication. Any material reproduced in this project has been properly acknowledged.

Signature:

.....
Md. Mashuk E-lahi
ID: 2012-2-55-048

.....
Md. Abdul Khalek Sujon
ID: 2012-2-55-024

APPROVAL

The project titled as “**Design & Developed of an Optical Mark Reader**” submitted to the respected 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:

Md. Mashuk E-lahi
ID: 2012-2-55-048

Md. Abdul Khalek Sujon
ID: 2012-2-55-024

Dr. Md. Habibur Rahman

Professor
Department of Electrical and Electronics Engineering
Faculty of Engineering and Technology
University of Dhaka

Dr. M. Mofazzal Hossain

Professor & Chairperson
Department of Electronics and Communications
Engineering
East West University

ACKNOWLEDGEMENT

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. We want to express our thanks and deep appreciation to our advisor Dr. Md. HabiburRahman as 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. We have special acknowledgement for our group members for their understanding each other and hard working from the beginning up to the end. Finally, we would like to thank all persons who were involved with this project for their valuable help and professionalism during this project.

ABSTRACTION

We know technology is a blessing in our daily life. Technology is making our life easy to easier. A man cannot think without technology even for a single day. In this project, we have designed and developed an Arduino based Optical Mark Reader machine. We have used an Arduinouno Uno board and the main controlling unit and some other devices like- MUXs, switches, LM324 ICs and an Infrared based reflective sensor. The switches worked as answers from the OMR sheet & the MUX's were used for selecting the switches and fetched them into the Arduino board. We also developed a program for proper functioning of the whole system using the Arduino-based software. The whole system has been designed and implemented properly. Finally the performance of the system has been studied and it is found that without some limitation of the sensor section the developed system works satisfactorily. It can read the answers of the OMR sheet with 100% accuracy.

Table of Contents

Short Content

Declarationi
Approvalii
Acknowledgementiii
Abstractiv

Chapter-1 Introduction

1.1.Motivation of the work2
1.2.Survey of literature2
1.2.1 Smart phone based OMR2
1.2.2 OMR developed by C and C++3
1.3.Organization of the report3

Chapter-2 Theoretical Background

2.1. Introduction5
2.2. OMR5
2.3. Multiplexer6
2.3.1. Types of Multiplexing7
2.3.1.1. FDM7
2.3.1.2. WDM7
2.3.1.3.TDM8
2.3.2 List of MUX ICs9
2.3.3 Advantage of MUX9
2.4 Arduino Uno9
2.4.1 Architecture of Arduino-board10
2.4.2 Technical Space11
2.4.3 Power12
2.4.4 Pins12
2.4.5 Reset Button13
2.4.6 TX RX LEDs13
2.4.7 Main IC13

Chapter-3 Design and Development

3.1. Introduction	15
3.2. Circuit Diagram	15
3.3. Hardware Configuration	16
3.3.1. 74151 Multiplexer	16
3.3.2. Connection Diagram.....	16
3.3.3. Pin Numbers and Name.....	16
3.3.4. Logic Diagram.....	17
3.3.5. Functional Descriptions	18
3.3.6. Truth table.....	18
3.4. LM324 IC.....	18
3.4.1. Connection Diagram.....	19
3.4.2. Features.....	19
3.4.3. General Descriptions	20
3.4.4. Internal Diagram.....	20
3.5. Switch	21
3.6 Sensor	21
3.6.1 Reflective Object sensor OPB704.....	21
3.6.2 Color difference white/black	22
3.6.3 Features.....	22
3.6.4 Specifications.....	22
3.7 Software Configuration.....	23
3.7.1 Connect to the board	23
3.7.2 Open the blink example.....	23
3.7.3 Select your board	24
3.7.4 Select your serial port.....	24
3.7.5 Upload the program	24
3.7.6 Flow chart.....	25
3.7.7 Programming Code.....	26

Chapter-4 Implementation and Result

4.1. Introduction.....	32
4.2. Implementation.....	32
4.3 Interfacing between Arduino and Mux.....	32
4.4 Results.....	33

Chapter-5 Conclusion

5.1. Conclusion.....	36
5.2. Future work scope.....	36

Reference.....	37
----------------	----

Chapter 1

Introduction

1.1 Motivation of the work

In this fast and furious period worldwide registering is developing so widely and each manual works are moving towards mechanized work. People don't want to invest their time in processing they just want to give an input and take an output immediately. And by the help of machines we can reduce the processing time. The OMR sheet can be checked by any straightforward scanner and by the utilization of this Framework it can be effectively surveyed additionally the outcome can be put into the database of the framework. The project is designed to serve that entire organization who conducts their exam on OMR sheet.

- ❖ This OMR machine hasn't been made in Bangladesh by any one yet & it is a very expensive machine to buy. If organization needs it then they'll have to import it from abroad with a large scale of money.
- ❖ Our main purpose was to reduce the cost of making it & make it available in our country in very low cost.
- ❖ This is main purpose for us to get motivated for making an OMR machine.

1.2 survey of literature

There are some organizations who have been working with OMR. Some of the examples are given below.

1.2.1 Smart phone based OMR [1]

- Umar Hassan (A Pakistani developer) developed a wonder full code on C sharp programming. He made an application. A smart phone with medium resolution camera with auto focus option can work as an Optical Mark Reader in his word.

The main feature of his work is given below:

- Image detection, analysis is now more precise, much reliable and faster.
- Added a whole class to manage OMR sheet in more like Object-oriented approach.
- Image color correction method changed from hit-and-trial to absolute.
- Ability to process images in a sync threads.
- XML sheet storage changed to access 2007 database
- added OMR.helpers.dbOps class for managing all database operations in one place

- Improved image detection algorithm to consume less memory
- Parameter based output quality and performance adjustment
- I nice GUI to create your own OMR sheets.

1.2.2 OMR developed by C and C++ [2]

Another group named **Eclipse Development Process** has developed an OMR program

The OMR project consists of a highly integrated set of open source C and C++ components that can be used to build robust language runtimes that will support many different hardware and operating system platforms. These components include but are not limited to: memory management, threading, platform port (abstraction) library, diagnostic file support, monitoring support, garbage collection, and native Just In Time compilation.

The long term goal for the OMR project is to foster an open ecosystem of language runtime developers to collaborate and collectively innovate with hardware platform designers, operating system developers, as well as tool and framework developers and to provide a robust runtime technology platform so that language implementers can much more quickly and easily create more fully featured languages to enrich the options available to programmers.

Alongside this project, we will be open sourcing our CRuby implementation that leverages the OMR technology, and we have a CPython implementation also that leverages some of the OMR technology. As we contribute the underlying OMR technology to the OMR project, we'll also open source the implementations to leverage that OMR technology for CRuby and eventually CPython

1.3 Organization of the report

An optical mark reader (OMR) is a data capturing technology used for reading 'tick box' information from printed forms using a scanner. An OMR is an ideal machine for the rapid and accurate scoring of multiple-choice questions (MCQs). The theoretical background of OMR is given in chapter two. Hardware and software analysis are in chapter three. After that we ran our project and got successful result according to our designs.

Chapter 2

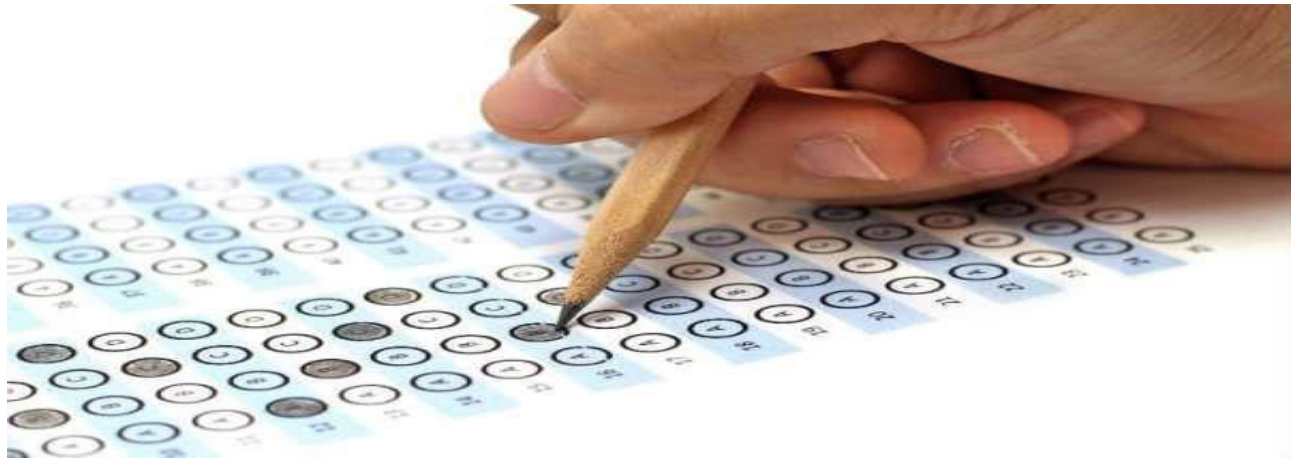
Theoretical Background

2.1 Introduction

The project include with four MUX, LM324 IC, Arduino Uno board, power supply, Switches and sensors. We develop the connection by upper connection. First we work with MUX and switches for manual connection and then we used sensor to make it into digitalize form with arduino. Finally we have got success in the project.

2.2 OMR

OMR stands for Optical Mark Reading or Optical Mark Recognition, OMR is the procedure of get-together data from individuals by perceiving blemishes on a report. OMR is expert by utilizing an equipment device (scanner) that distinguishes a reflection or restricted light transmittance on or through bit of paper. This technology is a very fast and accurate way to translate marks on paper into electronic data.



OMR allows for the processing of hundreds or thousands of physical documents per hour. For example, students may recall taking tests or surveys where they filled in bubbles on paper with pencil. Once the form had been completed, a teacher or teacher's assistant would feed the cards into a system that grades or gathers information from them.

2.3 Multiplexer

A Multiplexer is a device that permits one of a few analog or digital information signals which are to be chosen and transmits the data that is chosen into a solitary medium. Multiplexer is otherwise called Data Selector.

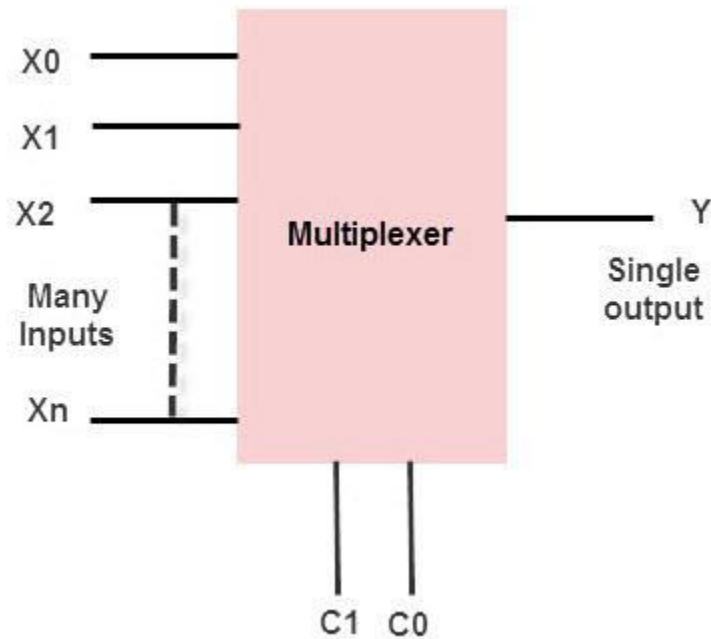
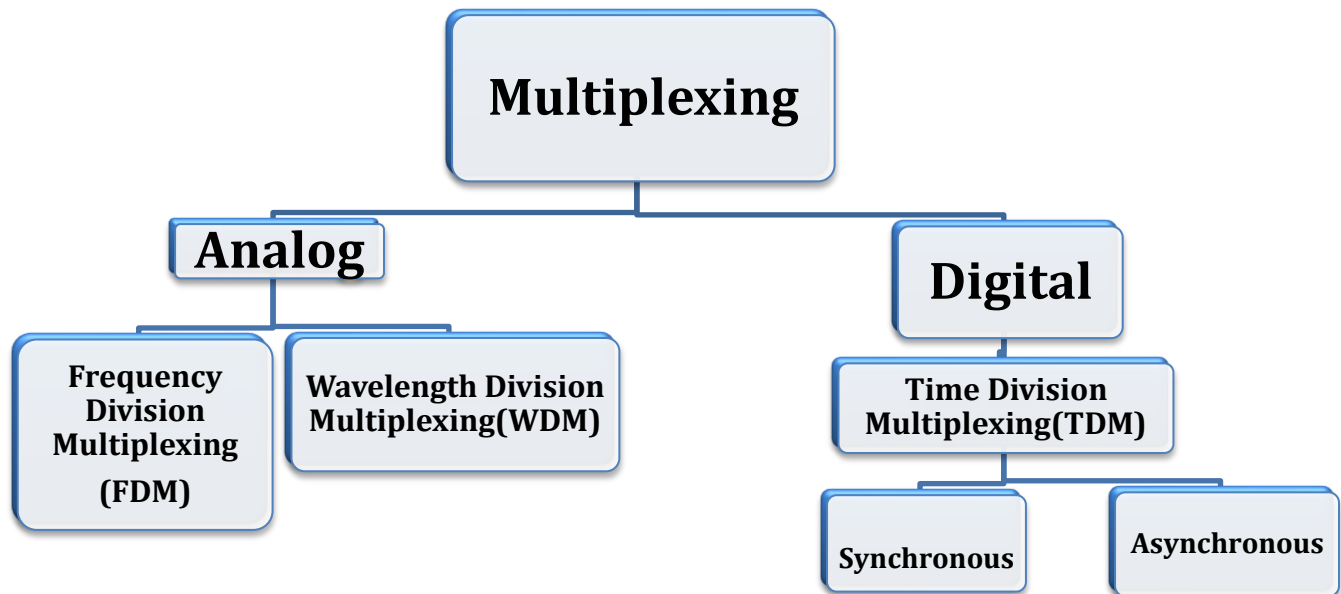


Fig: Multiplexer

A multiplexer of 2^n inputs has n select lines that will be utilized to select input line to send to the output. Multiplexer is contracted as Mux. MUX sends computerized or simple signs at higher pace on a single line in one shared device. It recoups the different signs at the less than desirable end. The Multiplexer supports or intensifies the data that later exchanged over system inside a specific transmission capacity and time.

2.3.1 Types of Multiplexing



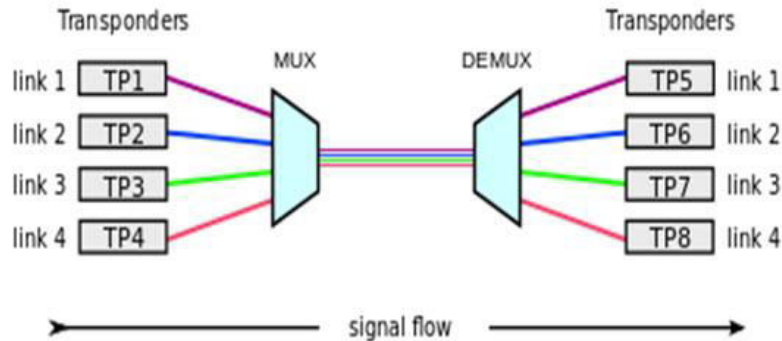
2.3.1.1 Frequency Division Multiplexing

Frequency Division Multiplexing is a technique which uses various frequencies to combine many streams of data for sending signals over a medium for communication purpose. It carries frequency to each data stream and later combines various modulated frequencies to transmission. Television Transmitters are the best example for FDM, which uses FDM to broad cast many channels at a time.

2.3.1.2 Wavelength Division Multiplexing

Wavelength Division Multiplexing (WDM) is analog multiplexing technique and it modulates many data streams on light spectrum. This multiplexing is used in optical fiber. It is FDM optical equivalent. Various signals in WDM are optical signal that will be light and were

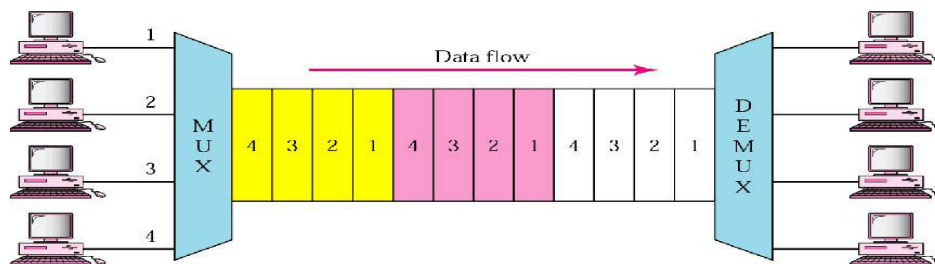
transmitted through optical fiber. WDM similar to FDM as it mixes many signals of different frequencies into single signal and transfer on one link. Wavelength of wave is reciprocal to its frequency, if wavelength increases then frequency decreases. Several light waves from many sources are united to get light signal which will be transmitted across channel to receiver.



Wavelength Division Multiplexing

2.3.1.3 Time Division Multiplexer

TDM is one of types of multiplexers which join data streams by allotting every stream different time slot in a set. It frequently transfers or sends various time slots in an order over one transmission channel. TDM attaches PCM data streams.



Time Division Multiplexer

TDM has also two types:

- **Synchronous Time Division Multiplexing:** In this technique, a single device gets only a fixed time slot to communicate, that can't be changed. No other device can take that slot, and if the slot is missed, it has to wait for the next cycle.
- **Asynchronous Time Division Multiplexing:** This technique is the opposite of the above, with flexible time slots. It's far more efficient because time slots are assigned to active channels or devices, so usually they don't go to waste, and priorities can be changed.

2.3.2 List of Mux ICs

The 7400 series has several ICs that contain multiplexers

Serial No.	IC No.	Functions	Output State
1.	74157	Quad 2:1 Mux	quad 2-line to 1-line data selector/multiplexer, no inverting
2.	74158	Quad 2:1 Mux	quad 2-line to 1-line data selector/multiplexer, inverting
3.	74153	Dual 4:1 Mux	dual 4-line to 1-line data selector/multiplexer
4.	74352	Dual 4:1 Mux	dual 4-line to 1-line data selectors/multiplexers with inverting outputs
5.	74151A	16:1 Mux	Both output available
6.	74151	8:1 Mux	8-line to 1-line data selector/multiplexer
7.	74150	16:1 Mux	16-line to 1-line data selector/multiplexer

2.3.3 Advantage of Mux

1. It decreases number of wires.
2. It decreases circuit complexity and expense.
3. It simplifies logic design.
4. We can execute numerous combinational circuits utilizing MUX.
5. It does not require k-maps and simplification.

2.4 Arduino Uno

Arduino is an open-source stage taking into account simple to-use equipment and programming. Arduino boards can read inputs - light on a sensor, a finger on a button, or a Twitter message - and transform it into an output - enacting an engine, turning on a LED, distributed something on the web. To do as such you utilize the Arduino programming language and run on your computer.

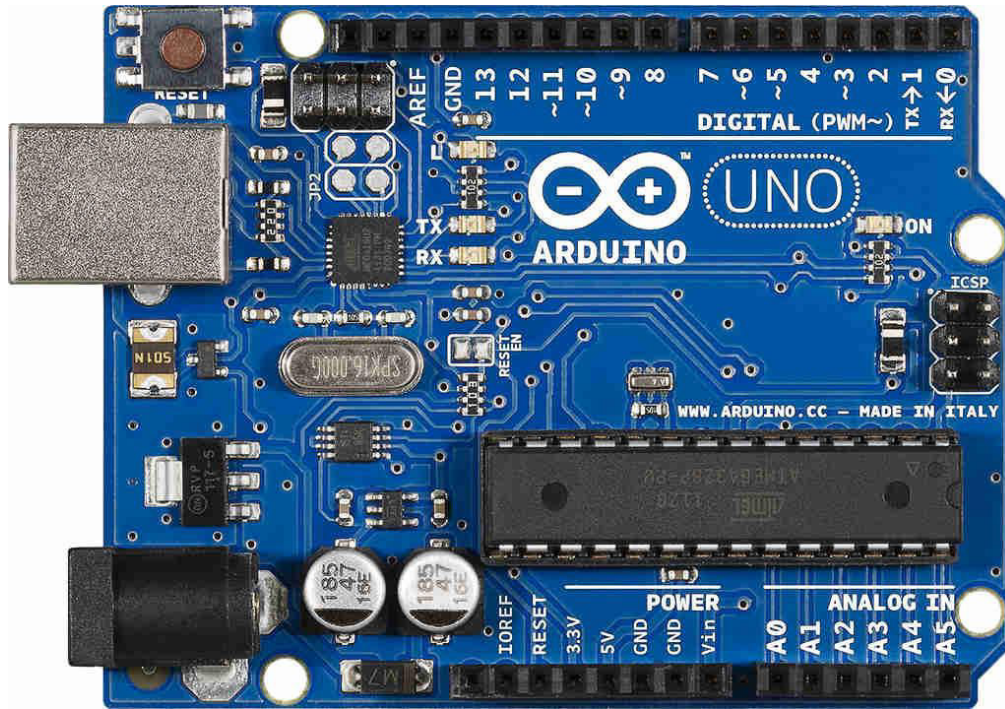


Fig: Arduino Uno

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 (called a programmer) in order to load new code onto the board – you can simply use a USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making it easier to learn to program. Finally, Arduino provides a standard form factor that breaks out the functions of the micro-controller into a more accessible package.

2.4.1 Architecture of Arduino-Uno board

An Arduino board is a one kind of microcontroller based pack. The main Arduino innovation was created in the year 2005 by David Cuartielles and Massimo Banzi. The originators thought to give simple and minimal effort board for understudies, specialists and experts to assemble devices. The Uno is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

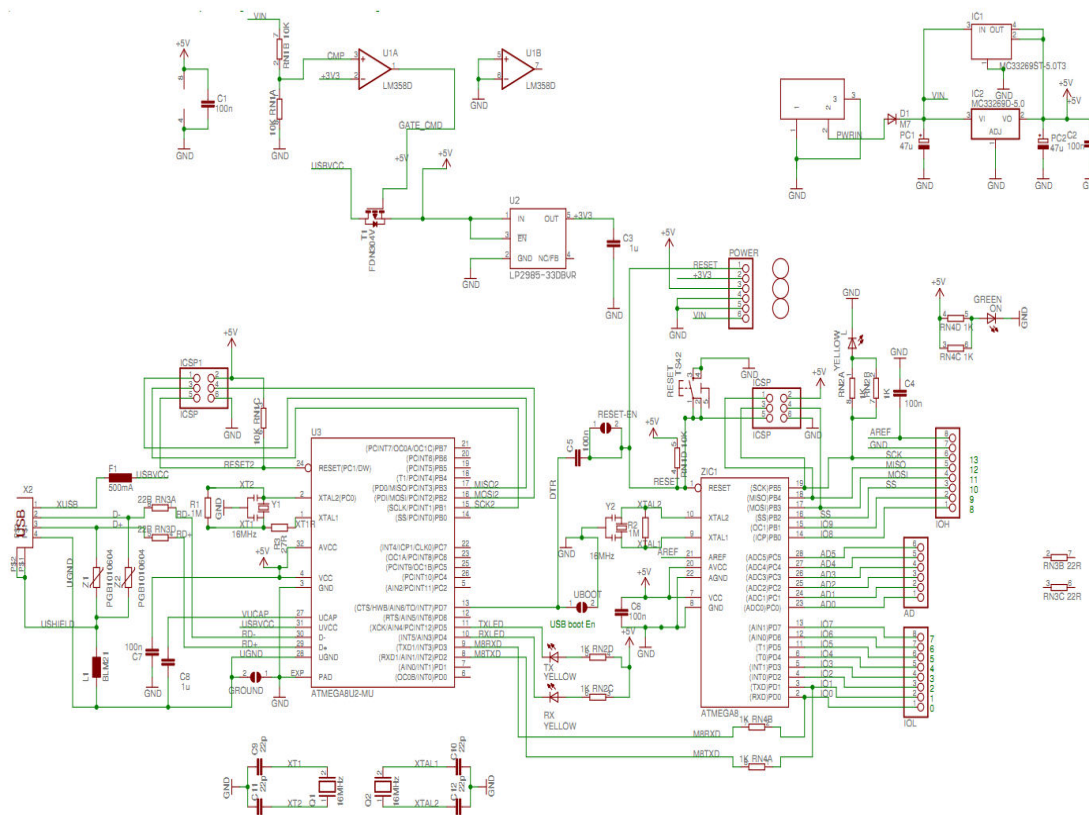


Fig: Schematic Diagram of Arduino Uno

2.4.2 Technical specs

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 boot-loader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

2.4.3 Power

Each Arduino board needs an approach to be associated with a power source. The Arduino UNO can be controlled from a USB link originating from your PC or a divider power supply that is ended in a barrel jack. The USB association is additionally how you will stack code onto your Arduino board. More on the best way to program with Arduino can be found in our [Introducing and Programming Arduino](#) instructional exercise. Remember that don't utilize a power supply more than 20 Volts as you will overpower your Arduino. The prescribed voltage for most Arduino models is somewhere around 6 and 12 Volts.

2.4.4 Pins

The pins on your Arduino are the spots where you interface wires to develop a circuit. They more often than not have dark plastic "headers" that permit you to simply connect a wire right to the board. The Arduino has a few various types of pins, each of which is named on the board and utilized for various capacities.

- **GND (3):** Short for ‘Ground’. There are several GND pins on the Arduino, any of which can be used to ground your circuit.
- **5V (4) & 3.3V (5):** As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.
- **Analog (6):** The area of the pins is under the ‘Analog In’ label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.
- **Digital (7):** Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).
- **PWM (8):** You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM). We have a tutorial on PWM, but for now, think of these pins as being able to simulate analog output (like fading an LED in and out).
- **AREF (9):** Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

2.4.5 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.

2.4.6 TX RX LEDs

TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. 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.

2.4.7 Main IC

The black thing with all the metal legs is an IC, or Integrated Circuit (13). Consider it the brains of our Arduino. The fundamental IC on the Arduino is marginally not the same as board type to board type, yet is as a rule from the ATmega line of IC's from the ATMEL organization. This can be essential, as you may need to know the IC type before stacking up another system from the Arduino programming.

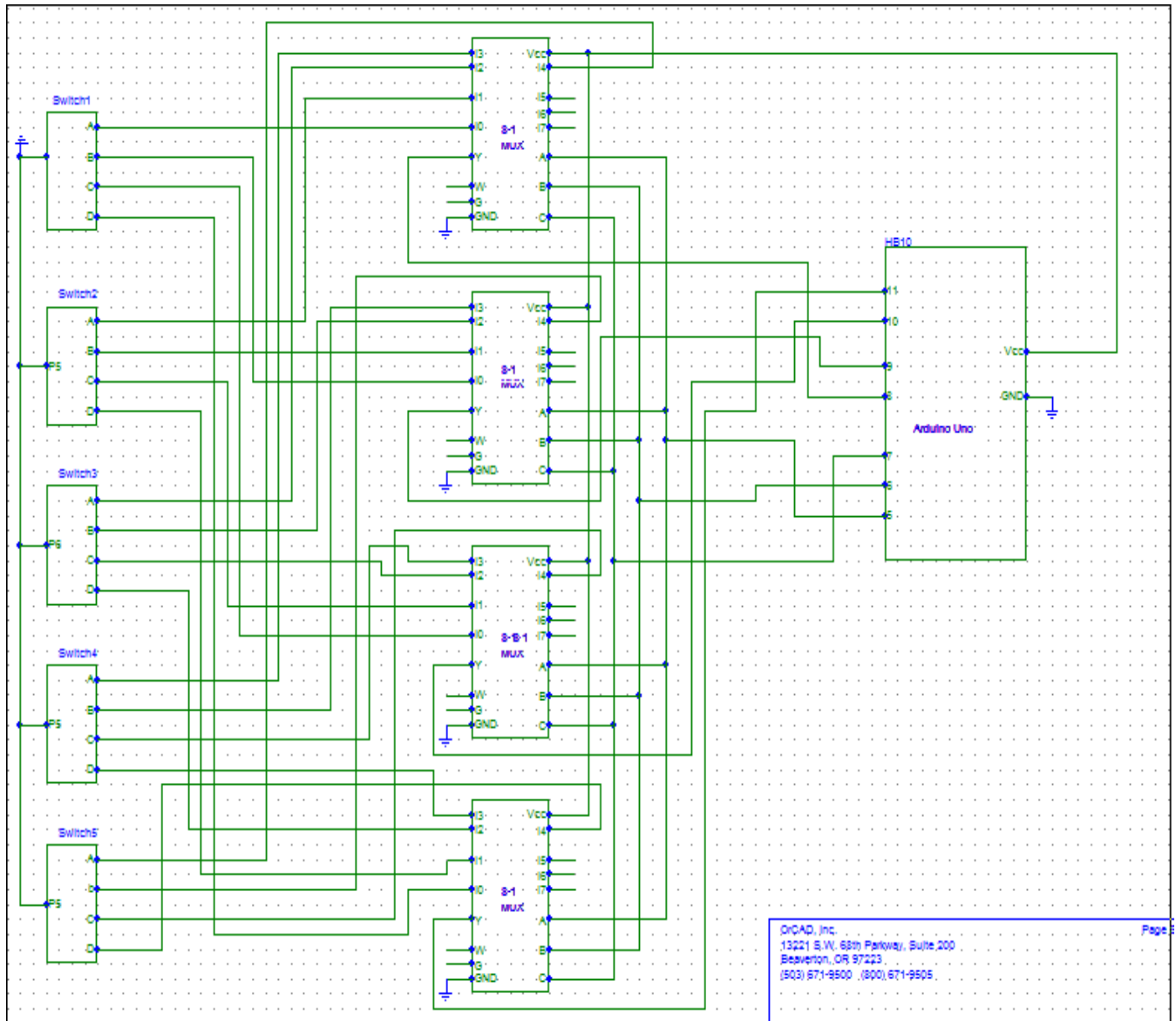
Chapter 3

Design & Development

3.1 Introductions

Optical Mark Reader (OMR), also called “mark sensing”, is a method of scanning technology in which data is input via marks made in predefined positions on a form and entering data into a computer system. Therefore, OMR is best for handling discrete data, where values fall into a limited number of values. In this chapter we would know about hardware and software configuration.

3.2 Circuit Diagram



3.3 Hardware Configuration

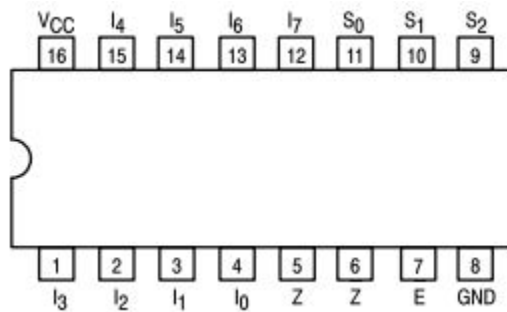
In this project we utilize two ICs. One is 74151 multiplexer which as known as 8-1 MUX and another is LM324. Here we use four MUX including five switches and also a reflective object sensor which make it digitalis.

3.3.1 74LS151 Multiplexer

The TTL/MSI SN54/74LS151 is a high speed 8-input digital Multiplexer. It gives, in one bundle, the capacity to choose one bit of bit from up to eight sources. The LS151 can be utilized as an all-inclusive capacity generator to create any rationale capacity of four variables. Both declaration and nullification output are given.

- ❖ Schottky Process for Rapid
- ❖ Multifunction Capacity
- ❖ On-Chip Select Rationale Disentangling
- ❖ Completely Supported Corresponding outputs
- ❖ Info Clasp Diodes Constrain Fast End Impacts

3.3.2 Connection Diagram



74LS151 IC

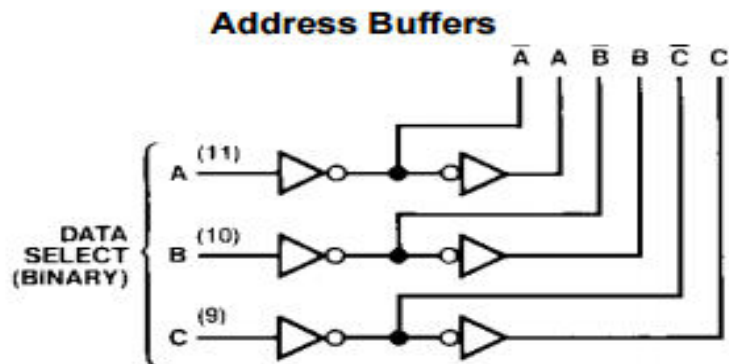
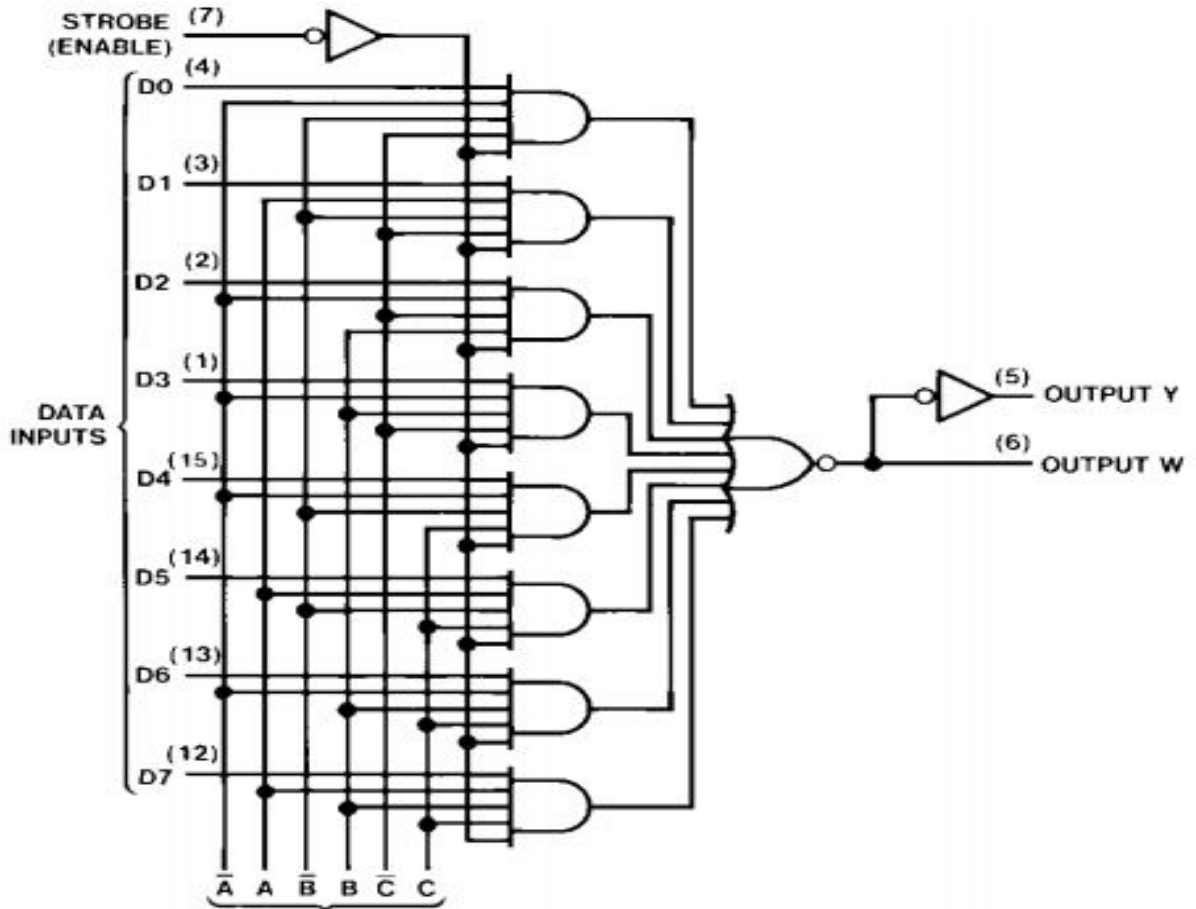
3.3.3 Pin Numbers and Name:

Pin Number	Pin Name
S0–S2	Select Inputs
E	Enable (Active LOW) Input
I0–I7	Multiplexer Inputs
Z	Multiplexer Output
Z	Complementary Multiplexer Output

NOTES:

- ✓ 1 TTL Unit Load (U.L.) = 40 μ A HIGH/1.6 mA LOW.
- ✓ The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Range

3.3.4 Logic Diagram



3.3.5 Functional Descriptions

The LS151 is a legitimate execution of a solitary shaft, 8-position switch with the switch position controlled by the condition of three Select inputs, S0, S1, and S2. The Enable input (E) is active LOW. When it is not activated, the negation output is HIGH and the assertion output is LOW regardless of all other inputs. The LS151 gives the capacity, in one bundle, to choose from eight sources of information or control data. By appropriate control of the inputs, the LS151 can give any rationale capacity of four variables and its refutation.

3.3.6 Truth Table

TRUTH TABLE

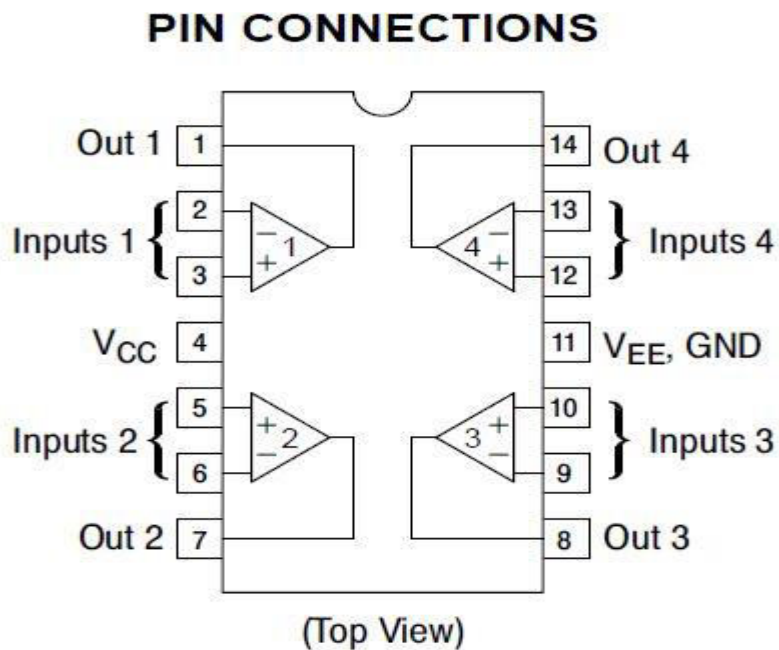
\bar{E}	S ₂	S ₁	S ₀	I ₀	I ₁	I ₂	I ₃	I ₄	I ₅	I ₆	I ₇	\bar{Z}	Z
H	X	X	X	X	X	X	X	X	X	X	X	H	L
L	L	L	L	L	X	X	X	X	X	X	X	H	L
L	L	L	L	H	X	X	X	X	X	X	X	L	H
L	L	L	H	X	L	X	X	X	X	X	X	H	L
L	L	L	H	X	H	X	X	X	X	X	X	L	H
L	L	H	L	X	X	L	X	X	X	X	X	H	L
L	L	H	L	X	X	H	X	X	X	X	X	L	H
L	L	H	H	X	X	X	L	X	X	X	X	H	L
L	L	H	H	X	X	X	H	X	X	X	X	L	H
L	H	L	L	X	X	X	X	L	X	X	X	H	L
L	H	L	L	X	X	X	X	H	X	X	X	L	H
L	H	L	H	X	X	X	X	X	L	X	X	H	L
L	H	L	H	X	X	X	X	X	H	X	X	L	H
L	H	H	L	X	X	X	X	X	X	L	X	H	L
L	H	H	L	X	X	X	X	X	X	H	X	L	H
L	H	H	H	X	X	X	X	X	X	X	L	H	L
L	H	H	H	X	X	X	X	X	X	X	H	L	H

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

3.4 LM324 IC

The LM324 series are low-cost, quad operational amplifiers with true differential inputs. They have several distinct advantages over standard operational amplifier types in single supply applications. The quad amplifier can operate at supply voltages as low as 3.0 V or as high as 32 V with quiescent currents about one-fifth of those associated with the MC1741 (on a per amplifier basis). The common mode input range includes the negative supply, thereby eliminating the necessity for external biasing components in many applications. The output voltage range also includes the negative power supply voltage.

3.4.1 Connection Diagram



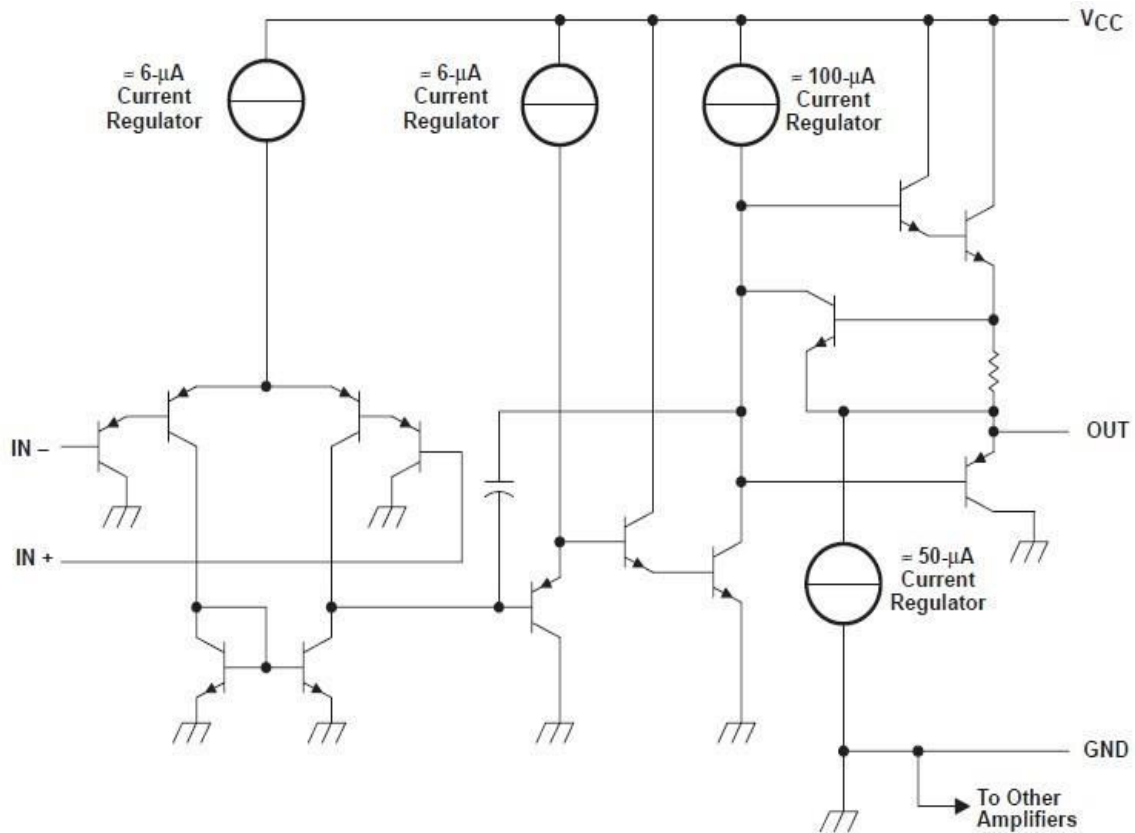
3.4.2 Features

- Short Circuited Protected Outputs
- True Differential Input Stage
- Single Supply Operation: 3.0 V to 32 V
- Low Input Bias Currents: 100 mA Maximum (LM324A)
- Four Amplifiers Per Package
- Internally Compensated
- Common Mode Range Extends to Negative Supply
- Industry Standard Pin-outs
- ESD Clamps on the Inputs Increase Ruggedness without Affecting Device Operation
- NCV Prefix for Automotive and Other Applications Requiring Unique Site and Control Change Requirements; AEC-Q100
- Qualified and PPAP Capable
- These Devices are Pb-Free, Halogen Free/BFR Free and are RoHS Compliant

3.4.3 General Description

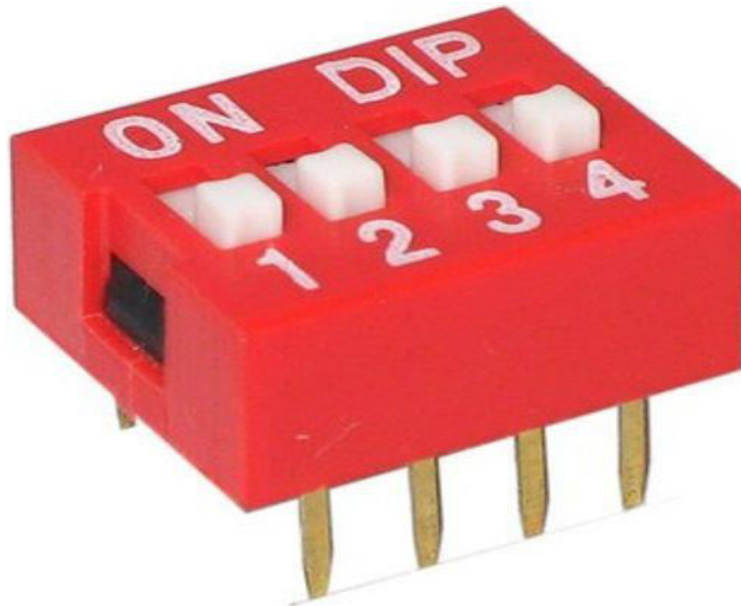
The LM124 series consists of four independent, high gain, internally frequency compensated operational amplifiers which were designed specifically to operate from a single power supply over a wide range of voltages. Operation from split power supplies is also possible and the low power supply current drain is independent of the magnitude of the power supply voltage. Application areas include transducer amplifiers, DC gain blocks and all the conventional op amp circuits which now can be more easily implemented in single power supply systems. For example, the LM124 series can be directly operated off of the standard +5V power supply voltage which is used in digital systems and will easily provide the required interface electronics without requiring the additional $\pm 15V$ power supplies.

3.4.4 Internal Diagram



3.5 Switch

A DIP switch is a manual electric switch that is packaged with others in a group in a standard dual in-line package (DIP). The term may refer to each individual switch, or to the unit as a whole. In this project we use five switches for manual configuration.

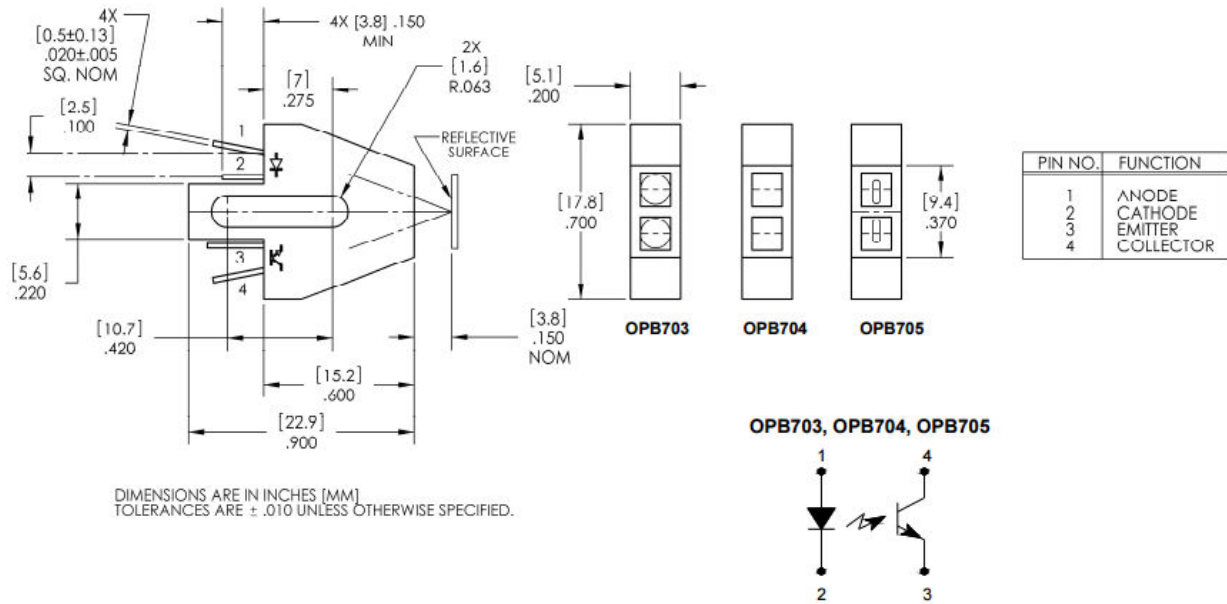


3.6 Sensor

A sensor is a device that detects and responds to some type of input from the physical environment. The specific input could be light, heat, motion, moisture, pressure, or any one of a great number of other environmental phenomena. The output is generally a signal that is converted to human-readable display at the sensor location or transmitted electronically over a network for reading or further processing. There are many different types of sensor but here we only use reflective object sensor which detects the presence of visible light using infrared transmission (IR).

3.6.1 Reflective object sensor OPB704

The OPB704 IR Reflective Object Sensor is a focused reflect sensor that is used for the detection of changes in a surface. The OPB704 is a direct pin-for-pin replacement for the once popular, but now discontinued, QRB1114. One minor difference between the two is the length of leads. The OPB704 is shorter and it has a slightly lower forward current but otherwise the form factor and specs are the same if not better. The OPB704 is great for line detection with robots; it is a 3.8mm focused reflective Infrared Photo Detector. This sensor has a very narrow range of detection also making it ideal for wheel and RPM counters and non-contact surface sensing.



3.6.2 Color Differences White/Black

If you want to detect the difference between white or black surfaces use an ADC (analog-to-digital convertor) pin on a microcontroller (such as an Arduino) or some other device that can utilize variable voltage levels. A black surface will give a voltage somewhere near the upper voltage (4~5V, when using a pull-up resistor), and white surfaces will give a voltage near ground (due to the phototransistor pulling the voltage down).

3.6.3 Features

- ❖ Phototransistor output
- ❖ Non-contact surface sensing
- ❖ Focused for sensing specular reflection
- ❖ Small size (18 x 23 x 5.5 mm)
- ❖ Daylight filter on sensor

3.6.4 Specifications

- ✓ Type: Infrared Phototransistor
- ✓ Detection Distance: 3.8mm
- ✓ Wavelength: 890 nm
- ✓ CE Voltage-Max: 30 V
- ✓ Forward (Drive) Current: 40 mA
- ✓ Power Dissipation: 100 mW

3.7 Software Configuration

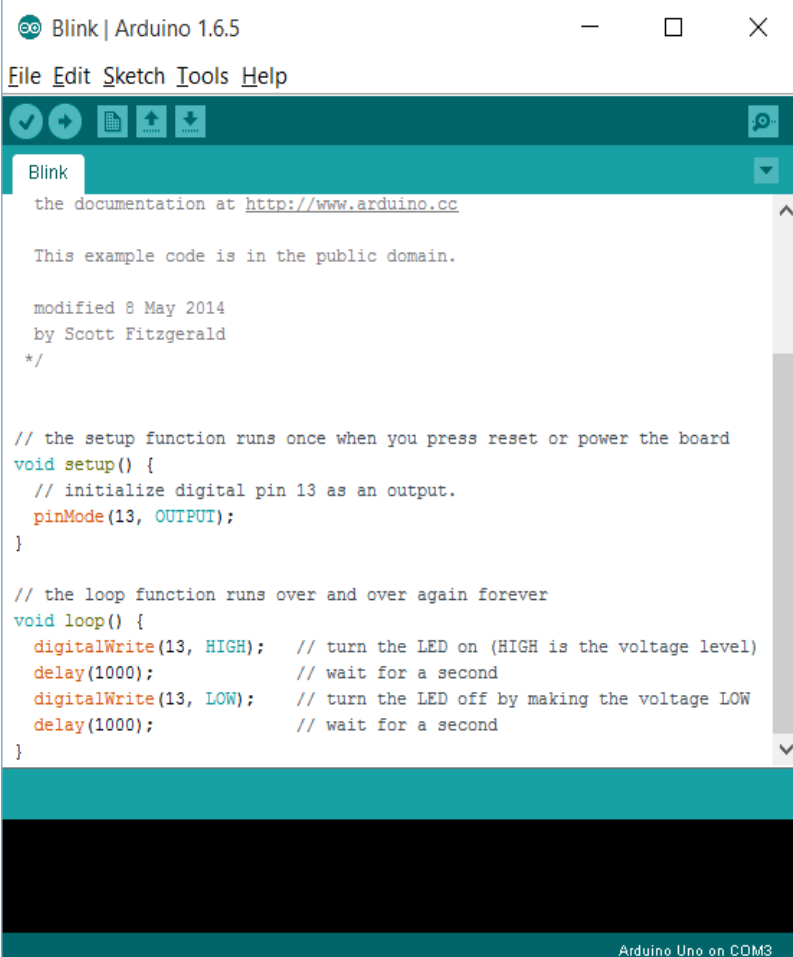
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. We get Arduino software from the official website of Arduino and downloaded windows versions because of our operating system.

3.7.1 Connect the board

The USB connection with the PC is necessary to program the board and not just to power it up. The Uno and Mega automatically draw power from either the USB or an external power supply. Connect the board to your computer using the USB cable. The green power LED (labeled PWR) should go on. Double-click the Arduino icon (arduino.exe) created by the installation process. (Note: if the Arduino Software loads in the wrong language, you can change it in the preferences dialog)

3.7.2 Open the blink example

Open the LED blink example sketch: File > Examples > 01.Basics > Blink.

A screenshot of the Arduino IDE window titled "Blink | Arduino 1.6.5". The window has a menu bar with "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for opening files, saving, and uploading. The main text area shows the code for the Blink sketch. The code includes a comment about the documentation at <http://www.arduino.cc>, a note that the code is in the public domain, and the date and author (modified 8 May 2014 by Scott Fitzgerald). The code defines a setup function to initialize digital pin 13 as an output and a loop function that turns the LED on (HIGH) for one second, then off (LOW) for one second. The status bar at the bottom right indicates "Arduino Uno on COM3".

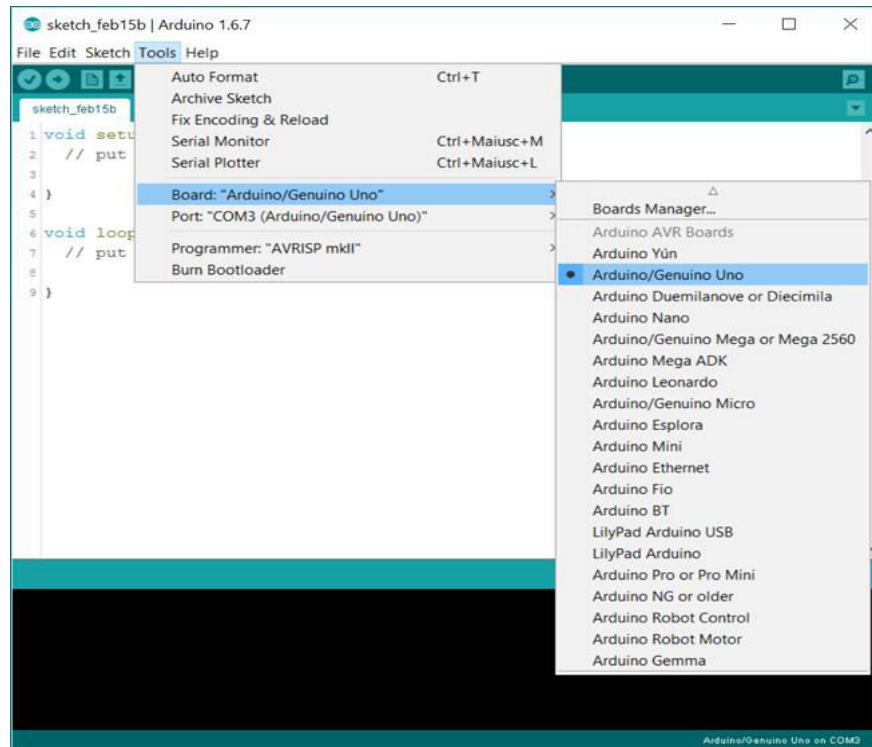
```
Blink | Arduino 1.6.5
File Edit Sketch Tools Help
the documentation at http://www.arduino.cc
This example code is in the public domain.
modified 8 May 2014
by Scott Fitzgerald
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);           // wait for a second
  digitalWrite(13, LOW); // turn the LED off by making the voltage LOW
  delay(1000);           // wait for a second
}
Arduino Uno on COM3
```

3.7.3 Select your board

You'll need to select the entry in the Tools > Board menu that corresponds to your Arduino or Genuino board.



Selecting an Arduino/Genuino Uno

3.7.4 Select your serial port

Select the serial device of the board from the 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 board and re-open the menu; the entry that disappears should be the Arduino or Genuino board. Reconnect the board and select that serial port.

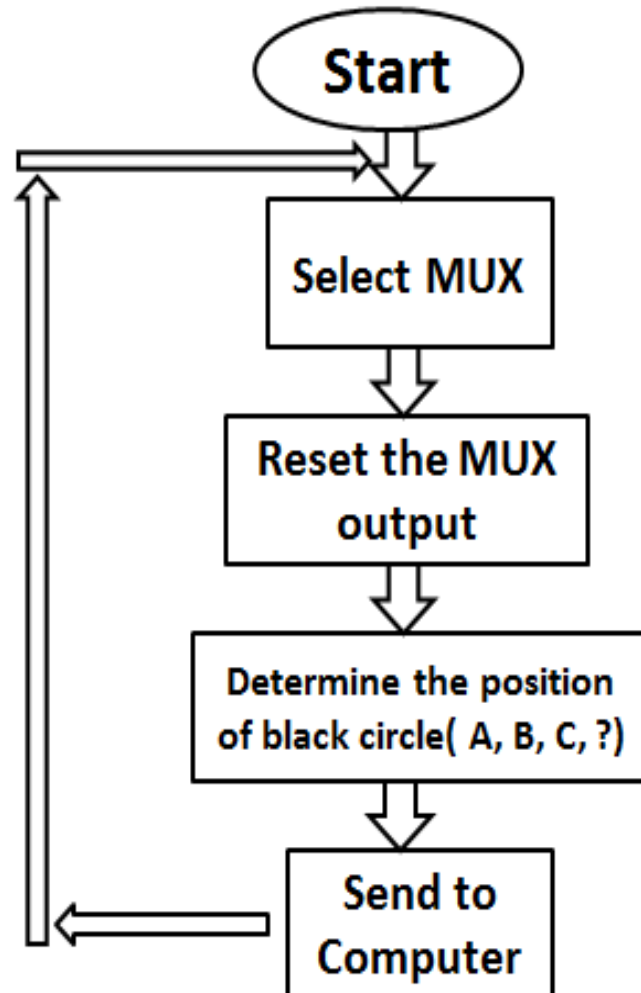
3.7.5 Upload the program

Now, simply click the "Upload" button in the environment. Wait a few seconds - you should see the RX and TX led are on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar.



A few seconds after the upload finishes, you should see the pin 13 (L) LED on the board start to blink (in orange). If it does, congratulations! You've gotten Arduino or Genuino up-and-running.

3.7.6 Flow Chart



3.7.7 Programming Code

/*

Button

Turns on and off a light emitting diode(LED) connected to digital pin 13, when pressing a pushbutton attached to pin 2.

The circuit:

- * LED attached from pin 13 to ground
- * push button attached to pin 2 from +5V
- * 10K resistor attached to pin 2 from ground

* Note: on most Arduinos there is already an LED on the board attached to pin 13.

created 2005

byDojoDave<<http://www.0j0.org>>

modified 30 Aug 2011

by Tom Igoe

This example code is in the public domain.

<http://www.arduino.cc/en/Tutorial/Button>

*/

// constants won't change. They're used here to

// set pin numbers:

constint SA = 5; // the number of the pushbutton pin

constint SB = 6;

constint SC = 7;

```
constint O0 =8 ;
constint O1 =9 ;
constint O2 =10 ;
constint O3 =11 ;
intva,vb,vc,vd;
void setup() {
    // initialize the LED pin as an output:
    pinMode(SA, OUTPUT);
    pinMode(SB, OUTPUT);
    pinMode(SC, OUTPUT);
    pinMode(O0, INPUT);
    pinMode(O1, INPUT);
    pinMode(O2, INPUT);
    pinMode(O3, INPUT);
    Serial.begin(9600);

}

void loop() {
    // read the state of the pushbutton value:
    digitalWrite(SA,LOW);
    digitalWrite(SB,LOW);
    digitalWrite(SC,LOW);
    va=digitalRead(O0);
    vb=digitalRead(O1);
```

```
vc=digitalRead(O2);
vd=digitalRead(O3);
if(va==0 &vb==1 &vc == 1 &vd==1)Serial.print("A");
else if (va==1 &vb==0 &vc == 1 &vd==1)Serial.print("B");
else if (va==1 &vb==1 &vc == 0 &vd==1)Serial.print("C");
else if (va==1 &vb==1 &vc == 1 &vd==0)Serial.print("D");
else if (va==1 &vb==1 &vc == 1 &vd==1)Serial.print("-");
elseSerial.print("?");
```

```
digitalWrite(SA,HIGH);
```

```
digitalWrite(SB,LOW);
```

```
digitalWrite(SC,LOW);
```

```
va=digitalRead(O0);
```

```
vb=digitalRead(O1);
```

```
vc=digitalRead(O2);
```

```
vd=digitalRead(O3);
```

```
if(va==0 &vb==1 &vc == 1 &vd==1)Serial.print("A");
```

```
else if (va==1 &vb==0 &vc == 1 &vd==1)Serial.print("B");
```

```
else if (va==1 &vb==1 &vc == 0 &vd==1)Serial.print("C");
```

```
else if (va==1 &vb==1 &vc == 1 &vd==0)Serial.print("D");
```

```
else if (va==1 &vb==1 &vc == 1 &vd==1)Serial.print("-");
```

```
elseSerial.print("?");
```

```
digitalWrite(SA,LOW);
```

```
digitalWrite(SB,HIGH);
```

```
digitalWrite(SC,LOW);  
va=digitalRead(O0);  
vb=digitalRead(O1);  
vc=digitalRead(O2);  
vd=digitalRead(O3);  
if(va==0 &vb==1 &vc == 1 &vd==1)Serial.print("A");  
else if (va==1 &vb==0 &vc == 1 &vd==1)Serial.print("B");  
else if (va==1 &vb==1 &vc == 0 &vd==1)Serial.print("C");  
else if (va==1 &vb==1 &vc == 1 &vd==0)Serial.print("D");  
else if (va==1 &vb==1 &vc == 1 &vd==1)Serial.print("-");  
elseSerial.print("?");
```

```
digitalWrite(SA,HIGH);  
digitalWrite(SB,HIGH);  
digitalWrite(SC,LOW);  
va=digitalRead(O0);  
vb=digitalRead(O1);  
vc=digitalRead(O2);  
vd=digitalRead(O3);  
if(va==0 &vb==1 &vc == 1 &vd==1)Serial.print("A");  
else if (va==1 &vb==0 &vc == 1 &vd==1)Serial.print("B");  
else if (va==1 &vb==1 &vc == 0 &vd==1)Serial.print("C");  
else if (va==1 &vb==1 &vc == 1 &vd==0)Serial.print("D");  
else if (va==1 &vb==1 &vc == 1 &vd==1)Serial.print("-");  
elseSerial.print("?");
```

```
digitalWrite(SA,LOW);
digitalWrite(SB,LOW);
digitalWrite(SC,HIGH);
va=digitalRead(O0);
vb=digitalRead(O1);
vc=digitalRead(O2);
vd=digitalRead(O3);
if(va==0 &vb==1 &vc == 1 &vd==1)Serial.print("A");
else if (va==1 &vb==0 &vc == 1 &vd==1)Serial.print("B");
else if (va==1 &vb==1 &vc == 0 &vd==1)Serial.print("C");
else if (va==1 &vb==1 &vc == 1 &vd==0)Serial.print("D");
else if (va==1 &vb==1 &vc == 1 &vd==1)Serial.print("-");
elseSerial.print("?");
Serial.println("");
//Serial.print
delay(1000);
}
```

Chapter 4

Implementation & Results

4.1 Introduction

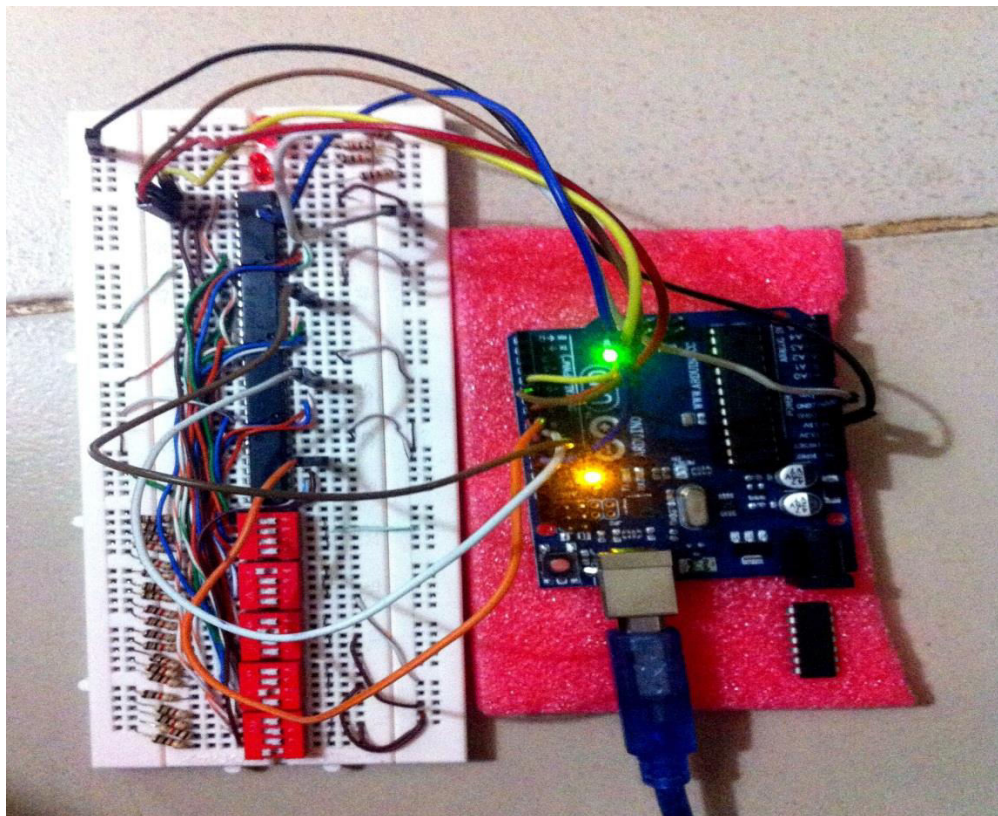
In the previous chapter we described about hardware and software configuration. After configuration we implement this project, run the software with desired code and we get satisfactory performance. In this chapter we will implement the code and we will see the result.

4.2 Implementation

All components are connected as circuit design. Then we upload the programming code in the Arduino and we get positive result. It works properly according to our plan.

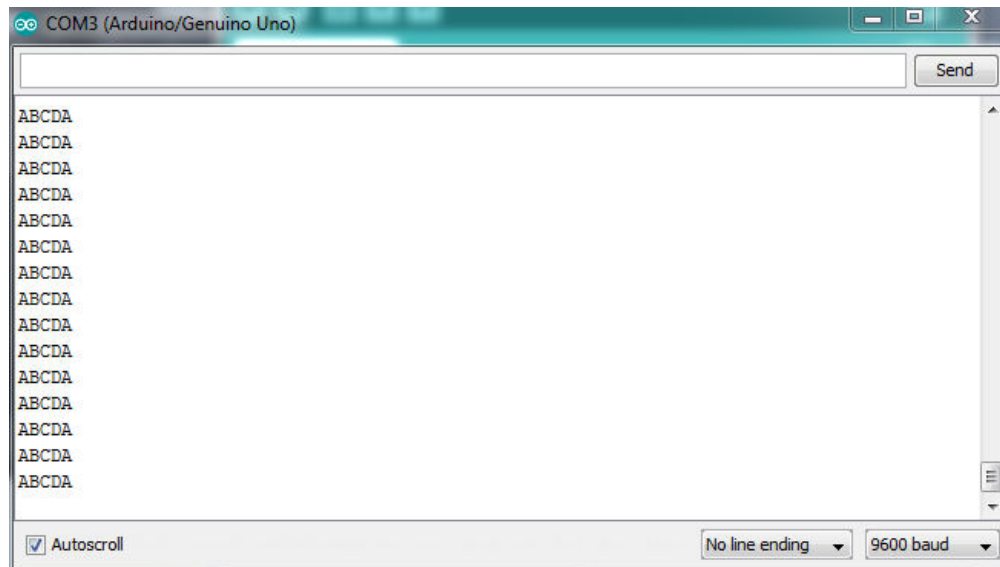
4.3 Interfacing between Arduino and Mux

Here we have used a Arduino-Uno board, bread board, 5 MUX, 4 switches. The switches worked as answers from the OMR sheet & the MUX's were used for selecting the switches and fetched them with the Arduino board. We used some Arduino codes for interfacing with circuit.



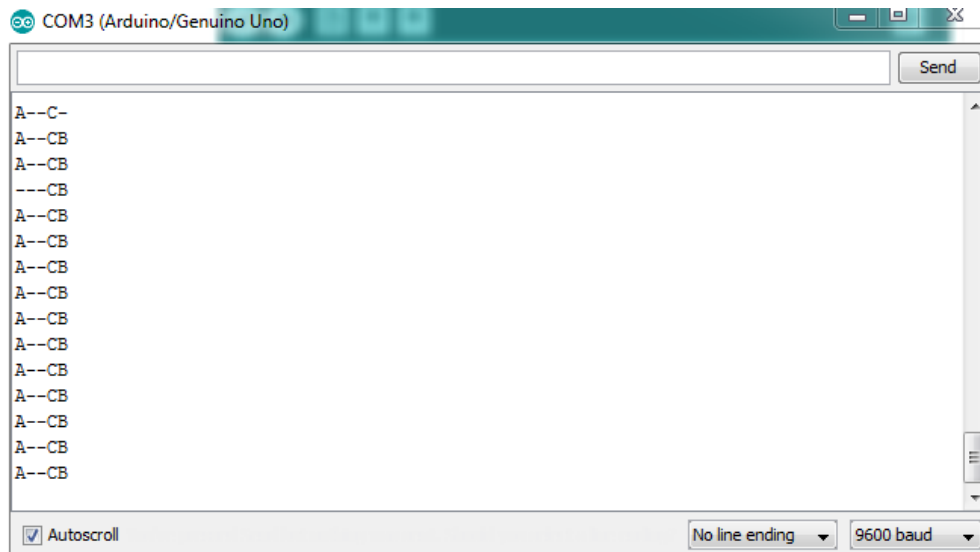
4.4 Results

If the OMR user gives all the answers according to rules then we see the following output.



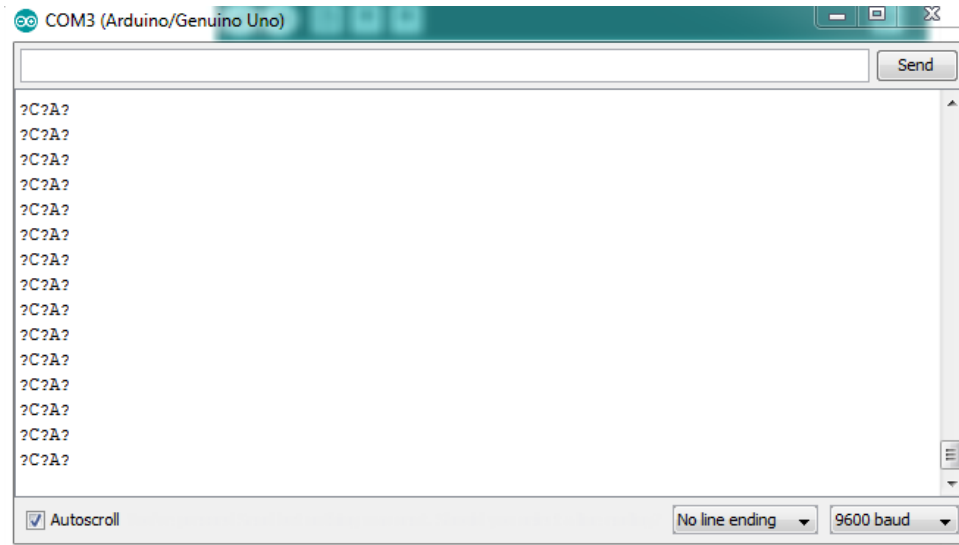
Output 1

If student isn't able to give all answers but gives some answers. Then the output result will be like the following figure. Here '-' means the question is not attempted by the student.



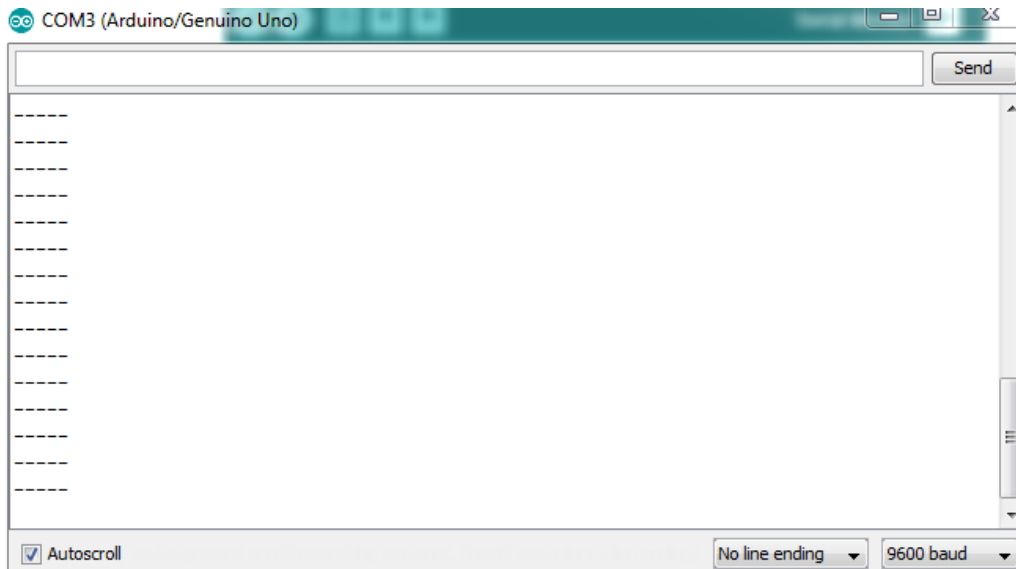
Output 2

If student gives more than one answer in a particular question then a question (?) symbol will be seen in the output of that particular answer.



Output 3

If the student doesn't give any answer then a dash (-) symbol will appear in the output.



Output 4

Chapter 5

Conclusion

5.1 Conclusion

We have done our best to show the design & developed of an Optical Mark Reader machine. In the market an Optical Mark Reader machine is very expensive. Its market value is almost around BDT 4, 00, 000- 12, 00,000. Our main target was to reduce the cost of making it & we are highly successful to achieve our goal. Our total project cost was around BDT 5,000. The mechanical portion has been designed in this project.

In this project we've used switches, MUX's, bread board, LED light, LM324 IC & an Arduino board for completing the project. As for detection, we have used switches for providing us with the answers. But we have also use an infrared based light reflective sensor to originally detect the circle filled-up or blank circle. It shows a voltage difference for filled-up on blank circle.

5.2 Future work scope

In future our plan is to use infrared based light reflective sensor instead of switches for providing us with all the answers. And also give it a proper shape for detection process using automatic OMR sheet movement. An automated system can be developed in future for detection & output through this project.

References

1. <http://www.codeproject.com/Articles/884518/Csharp-Optical-Marks-Recognition-OMR-Engine-a-M>
2. <https://projects.eclipse.org/proposals/omr>
3. <http://efxkits.com/blog/what-is-multiplexer-and-types/> ---
4. <http://www.edgefxkits.com/blog/know-all-about-multiplexing-in-mobile-network/>
5. <https://www.arduino.cc/>
6. <http://www.micropik.com/PDF/74ls151.pdf>
7. <http://www.futurlec.com/74LS/74LS151.shtml>
8. http://www.onsemi.com/pub_link/Collateral/LM324-D.PDF
9. <https://solarbotics.com/product/opb704/>
10. http://courses.cs.washington.edu/courses/csep567/10wi/labs/OPB703-705_70A-70D-B.pdf
11. <http://whatis.techtarget.com/definition/sensor>
12. <https://www.arduino.cc/en/Guide/Windows>